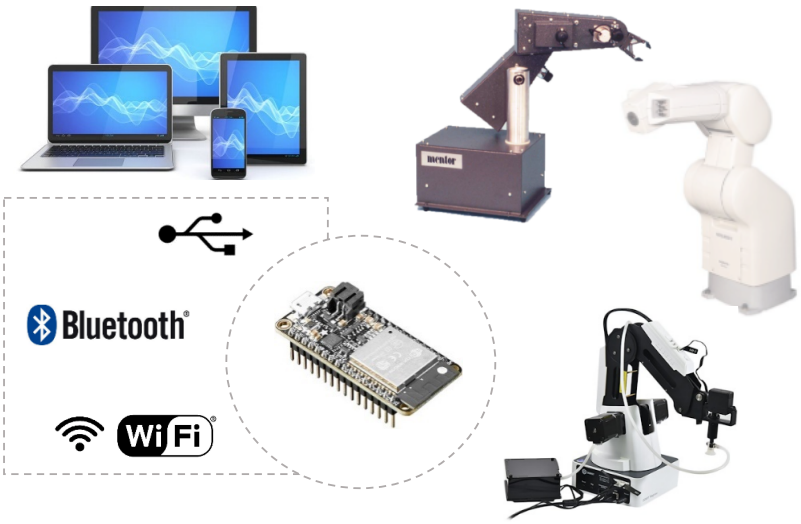
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrotécnica Energia e Automação



Interface Remota Genérica para Braços Robóticos

André Filipe Balsas Mira
Licenciado em Engenharia Eletrotécnica

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia Eletrotécnica – ramo de Automação e Eletrónica Industrial

Orientadores:
Professora Doutora Maria da Graça Vieira de Brito Almeida (ISEL)
Professor Doutor Pedro Miguel Florindo Miguens Matutino (ISEL)

Júri:
Presidente: Professor Doutor Luís Manuel dos Santos Redondo (ISEL)

Vogais:
Professor Doutor Armando José Leitão Cordeiro (ISEL)
Professora Doutora Maria da Graça Vieira de Brito Almeida (ISEL)

Janeiro 2021

Resumo

A automação industrial não se restringe aos limites físicos de uma fábrica ou instalação industrial de produção, pois cada vez mais os processos são comandados e supervisionados remotamente. Nos dias modernos e com a evolução tecnológica, através de aplicações em dispositivos eletrónicos, pode-se facilmente realizar um controlo remoto, sem fios, de diferentes periféricos.

A presente dissertação de Projeto Final de Mestrado, Interface Remota Genérica para Braços Robóticos, tem como objetivo, fazer o estudo teórico, a simulação e um protótipo experimental de uma interface remota genérica, desenvolvida para realizar o controlo de braços robóticos, com recurso a um microcontrolador. O microcontrolador utilizado, possibilita o controlo de braços robóticos através de três tipos de comunicações distintas, a comunicação série e duas comunicações sem fios, Bluetooth e Wi-Fi.

O trabalho iniciou-se com o estudo do microcontrolador e de dois braços robóticos, o Mentor, um braço robótico de 1985 e o Dobot Magician, um braço robótico mais recente, comercializado a partir de 2016. Através da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather, com um microcontrolador ESP-32 de baixo custo, desenvolvido pela Espressif Systems, foi criada a interface do sistema para controlo de braços robóticos.

O controlo dos braços robóticos adotados, permite demonstrar o conceito da interface remota genérica desenvolvida. Além disto, foi também desenvolvida, uma interface utilizador-máquina distinta, para cada uma das três comunicações.

Palavras Chave: Automação Industrial, Robótica, Controlo Remoto, Microcontrolador, Braço Robótico, Interface, Comunicação Série, Bluetooth, Wi-Fi.

Abstract

Industrial automation is not restricted to the physical limits of a factory or industrial production facility, as more and more processes are remotely controlled and supervised. In modern days technological evolution allows, through applications accessed in any electronic device, one can easily perform a wireless remote control of many peripherals.

The present dissertation of Final Master's Project, Generic Remote Interface for Robotic Arms, aims to make the theoretical study, simulation and an experimental prototype of a generic remote interface, developed to perform the control of robotic arms, using a microcontroller.

The microcontroller used, allows the control of robotic arms through three distinct communications, serial communication and two wireless communications, Bluetooth and Wi-Fi.

The work began with the study of the microcontroller and two robotic arms. The Mentor, a 1985 robotic arm and the Dobot Magician, a newer robotic arm, marketed from 2016. Through the development platform Adafruit Huzzah32 - ESP32 Feather, with a low-cost ESP-32 microcontroller, developed by Espressif Systems, the system interface for robotic arm control was created.

The control of the adopted robotic arms allows to demonstrate the concept of the generic remote interface developed. In addition, a distinct user-machine interface has been developed for each available communication.

Keywords: Industrial Automation, Robotics, Remote Control, Microcontroller, Robotic Arm, Interface, Serial Communication, Bluetooth, Wi-Fi.

Agradecimentos

Sem hierarquizar ninguém, agradeço aos meus pais que nunca deixaram de me guiar e de me passar os valores certos. Ao meu pai, agradeço que me tenha sempre encorajado com a maior força e à minha mãe, que sempre me ensinou que conseguimos ser realmente bons naquilo em que investimos com muita dedicação.

Ao meu irmão que, entre discussões e momentos de alegria, nunca me deixou caminhar sozinho.

À minha namorada, que esteve comigo incondicionalmente, pelas suas palavras de grande carinho e preocupação, fica para sempre um sentimento de gratidão imenso.

O presente trabalho só foi possível com o contributo de algumas pessoas que de alguma forma me ajudaram neste percurso académico.

Agradeço aos meus orientadores, Doutora Graça Almeida e Doutor Pedro Miguens por toda a orientação, apoio, disponibilidade e disponibilização de recursos em todas as fases do desenvolvimento desta dissertação.

Ao departamento de Engenharia Mecânica, pela disponibilização de um dos braços robóticos, utilizado no projeto sem o qual não teria sido possível a sua implementação prática.

Por último e não menos importante, a todos os amigos e colegas que me foram acompanhando ao longo desta jornada académica e sem os quais, chegar aqui teria sido muito mais difícil.

Agradeço toda a ajuda e companheirismo, e tomo consciência que esta passagem me brindou com pessoas incríveis.

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

µC	-	Microcontrolador
µP	-	Microprocessador
µs	-	Microsegundo
ADC	-	Analog-to-Digital Converter
AES	-	Advanced Encryption Standard
AI	-	Analog Input
AP	-	Access Point
APAS	-	Adaptable-Programmable Assembly System
APT	-	Automatically Programmed Tool
ASCII	-	American Standard Code for Information Interchange
BPS	-	Bits Per Second
BSS	-	Basic Service Set
CCS	-	Cascading Style Sheet
CLK	-	Clock
CPU	-	Central Processing Unit
DAC	-	Digital-to-Analog Converter
DARPA	-	Defense Advanced Research Projects Agency
DEC	-	Decimal
GPIO	-	Global Purpose Input Output
HEX	-	Hexadecimal
HTML	-	Hyper Text Markul Language
I^2C	-	Inter-Integrated Circuit
ICAM	-	Integrated Computer-Aided Manufacturing
IDE	-	Integrated Development Environment
IRGBR	-	Interface Remota Genérica para Braços Robóticos
LED	-	Light Emitting Diode
MIT	-	Massachusetts Institute of Technology
NASA	-	National Aeronautics and Space Administration
NC	-	Numeric Control
OE	-	Output Enable
PC	-	Portable Computer
PCI	-	Peripheral Component Interconnect
PLC	-	Programmable Logic Controller

PUMA	-	Programmable Universal Machine for Assembly
PWM	-	Pulse With Modulation
RCC	-	Remote Center Compliance
RCLK	-	Register Clock
RD	-	Received Data
RIA	-	Robot Institute of America
SCARA	-	Selective Compliance Arm for Robotic Assembly
SER	-	Shift Enable Register
SPI	-	Serial Peripheral Interface
SRCLK	-	Shift Register Clock
SRCLR	-	Shift Register Clear
SRI	-	Stanford Research Institute
SSID	-	Service Set Identifier
STA	-	Station
TD	-	Transmitted Data
TKIP	-	Temporal Key Integrity Protocol
UART	-	Universal Asynchronous Receiver-Transmitter
USB	-	Universal Serial Bus
WEP	-	Wired Equivalent Privacy
WLAN	-	Wireless Local Area Network
WPA	-	Wired Protected Access

1

Introdução

“ One machine can do the work of a hundred ordinary men,

but no machine can do the work of one extraordinary man. ”

Elbert Hubbard

1. Introdução

Neste Capítulo são apresentados sucintamente os tópicos relevantes para a concretização de todo o trabalho da dissertação no âmbito do mestrado em Engenharia Eletrotécnica, no ramo de Automação e Eletrónica Industrial. São apresentados o enquadramento, a motivação, os objetivos, a organização e estrutura deste documento.

1.1 Enquadramento

A automação industrial funciona como uma ligação de diferentes campos de desenvolvimento tecnológico e tem-me proporcionado a necessidade de adquirir sempre novos conhecimentos em outras áreas científicas.

No contexto industrial cada vez mais se automatizam os processos. Em automação industrial o uso de braços robóticos é recorrente, pois realizam eficazmente e rapidamente o trabalho pretendido, permitindo assim a realização de tarefas repetitivas e com elevada precisão.

A robótica é uma disciplina com múltiplas áreas de aplicação e atuação que acompanha a evolução tecnológica. Os processos de controlo de braços robóticos acompanham essa mesma evolução, simplificando cada vez mais os processos de controlo complexos através de interfaces mais simples e intuitivas para o utilizador.

1.2 Motivação

Com uma procura enorme no mercado, de inovações tecnológicas e soluções mais acessíveis e fáceis de usar, surge a ideia de conciliar os meus gostos pessoais, nomeadamente smartphones, robótica e automação industrial, com toda a aprendizagem e desenvolvimento da licenciatura em Engenharia Eletrotécnica e no mestrado em Engenharia Eletrotécnica ramo Automação e Eletrónica Industrial, e proponho assim conceber uma interface genérica para realizar o controlo de braços robóticos através de qualquer equipamento eletrónico, como por exemplo, um laptop, smartphone ou tablet.

Dada a diversidade de braços robóticos existentes atualmente, conceber um controlo, que recorra a várias interfaces de comunicação e que as mesmas possam ser usadas perante os diversos robôs, mantendo a interface para o utilizador, será o caminho a percorrer.

1.3 Objetivos

O objetivo do trabalho de dissertação de mestrado é conceber, desenvolver e implementar uma Interface Genérica de Controlo Remoto para Braços Robóticos. De modo a concretizar a funcionalidade de controlo remoto, a interface será implementada, tendo por base um controlador, que permitirá, comunicação Bluetooth e comunicação Wi-Fi. Na Figura 1, está representada a arquitetura do sistema proposto para desenvolvimento do projeto.

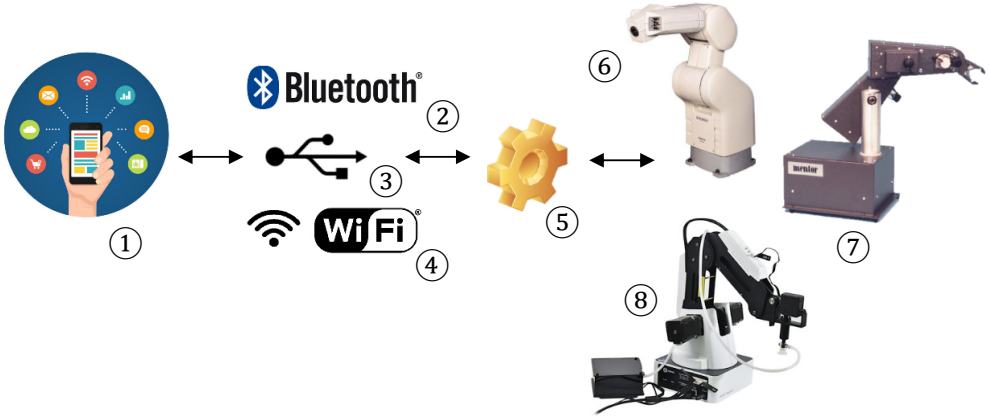


Figura 1 - Modelo proposto para desenvolvimento do projeto.

Em ①, na Figura 1, está representada a interface genérica a que o utilizador tem acesso para o controlo dos braços robóticos adotados. O controlador ⑤, proporciona o uso dos protocolos de comunicação série ③, Bluetooth ② e Wi-Fi ④ e permite estabelecer ligações, para uma diversidade de braços robóticos a controlar. Os pontos ⑥⑦⑧, representam a diversidade de manipuladores que a interface pode controlar, como por exemplo o braço robótico Mitsubishi RV-E3J ⑥, o Mentor ⑦ e o Dobot Magician ⑧.

O desenvolvimento deste projeto, passa pelo estudo e compreensão do controlador escolhido, da análise dos braços robóticos que serão controlados de forma a tirar partido das suas funcionalidades, e para cada comunicação utilizada, terá de ser criada uma interface apropriada e ajustada ao controlo dos braços robóticos.

Para realizar o controlo de um braço robótico, a solução é recorrer a uma aplicação, que poderá ser acedida a partir de qualquer dispositivo, com o devido acesso a comunicação sem fios, Bluetooth ou Wi-Fi. O controlador através dos protocolos de comunicação Série, Bluetooth e/ou Wi-Fi, recebe comandos que o utilizador introduz na aplicação, faz o processamento, e comunica com o braço robótico, estabelecendo assim o ciclo de controlo.

Os pontos chave e desafiantes para a concretização deste projeto passam pelo desenvolvimento da interface para o utilizador, na forma como se faz o envio e receção de comandos na interface e o seu processamento, tarefa de grau acrescido devido ao objetivo de concretizar três comunicações distintas para realizar o controlo dos braços robóticos. Outro desafio será analisar os diferentes braços robóticos utilizados neste projeto, para estabelecer os protocolos necessários de ligação e comunicação e compreender as funcionalidades de cada um.

O objetivo principal do trabalho é a concretização e implementação de uma interface genérica, que ofereça as três possibilidades de comunicação, Série, Bluetooth e Wi-Fi, e esteja adaptada para receber qualquer braço robótico, que se pretenda adicionar, tendo apenas que ser realizada a adaptação do módulo de software que terá o protocolo específico do robô, mantendo os restantes módulos de software.

1.4 Organização e estrutura do documento

O documento está organizado em cinco capítulos.

O primeiro e presente capítulo, introduz o tema abordado, apresentando o enquadramento, a motivação e os objetivos.

O segundo capítulo, descreve o estado de arte, abordando temas fundamentais, sobre a origem da robótica, o seu desenvolvimento histórico, os fundamentos dos robôs manipuladores, possibilidades de controlo dos mesmos, métodos de comunicação e um estudo de dois braços robóticos.

No terceiro capítulo, é descrito o trabalho realizado para o desenvolvimento da interface remota genérica, é abordada a arquitetura proposta, a solução adotada, os comandos implementados e as comunicações desenvolvidas.

No capítulo quatro, são apresentados e analisados os resultados obtidos, a controlar os braços robóticos adotados para validação do sistema proposto, através da utilização experimental da interface remota genérica implementada.

No último capítulo, são apresentadas as conclusões gerais provenientes do desenvolvimento de todo o sistema e são abordadas as perspetivas e ideias de possíveis desenvolvimentos futuros.

2

Estado de Arte

“ If I had nine hours to chop down a tree,

I’d spend the first six sharpening my axe. ”

Abraham Lincoln

2. Estado de arte

Neste capítulo são abordados os temas mais relevantes para o desenvolvimento do trabalho realizado, bem como algum trabalho relevante já desenvolvido no âmbito do controlo de braços robóticos. Destacando-se a origem da robótica e os seus desenvolvimentos históricos, os fundamentos dos robôs, alguns exemplos de controlo de manipuladores, métodos de comunicação e a análise de dois braços robóticos aqui considerados para validação do sistema proposto.

2.1 Origem e desenvolvimento da Robótica

A necessidade de aumentar a produtividade e fornecer produtos finalizados com qualidade motiva a indústria a procurar cada vez mais soluções de automação com base em computadores. Atualmente, a maioria das tarefas automatizadas são realizadas por máquinas, com finalidades especificas, projetadas para executar funções predeterminadas num processo de fabrico. A inflexibilidade e o custo, geralmente, alto destas máquinas, designados de sistemas de automação fixos, levam a um amplo interesse no uso de robôs capazes de executar uma diversidade de funções, num ambiente de trabalho flexível e com custos de produção reduzidos.

O *Robot Institute of America* (RIA), fornece uma designação para robôs industriais [1], designando que um robô é um manipulador multifuncional reprogramável projetado para mover materiais, peças, ferramentas ou dispositivos especializados, através de movimentos programados para o desempenho de uma variedade de tarefas. Em resumo, um robô é um manipulador de uso geral reprogramável com sensores externos que podem executar várias tarefas. Com esta definição, um robô tem inteligência, muito devido a programas de computador, associadas aos seus sistemas de controlo e deteção.

Com o decorrer da história, a ciência passa por uma constante evolução. Este processo origina o aperfeiçoamento de tecnologia que revolucionou e revoluciona a produção e o fabrico no setor industrial. A humanidade sempre sonhou em produzir mais e melhor, e com as transformações velozes no setor industrial, a partir do século XVIII, ficaram registadas 4 etapas [2], designadas como revoluções industriais, caracterizando determinados momentos históricos que originaram mudanças significativas na forma como o homem realiza o processo de fabrico, com melhorias bastante significativas na quantidade produzida, na qualidade da produção e na sua autonomia. A primeira revolução industrial, ocorre no final do século XVIII e fica caraterizada pelo uso do carvão como fonte de energia dando origem à máquina a vapor. Neste século a produção que antes

era artesanal deu origem à automação de processos com a inserção de máquinas no processo de produção.

A segunda revolução industrial, ocorre no fim do século XIX com a introdução da energia elétrica e do petróleo como novas fontes de energia, permitindo a criação de linhas de produção automatizadas melhorando os tempos e os custos dos processos de fabrico.

A terceira revolução industrial, ocorre na década de 1970 marcada pelos avanços nos sistemas eletrónicos programáveis, desenvolvendo matérias como a informática, a robótica e as telecomunicações impulsionadas com a elevada otimização e aumento da produtividade industrial.

A quarta revolução industrial, evolui nos dias de hoje e pretende fazer a ligação entre o mudo digital e o mundo real. A flexibilidade é o objetivo e o desenvolvimento de sistemas inteligentes são a solução. A indústria está em constante evolução, a automatização dos processos de produção é cada vez maior, e a indústria tem como futuro próximo, a implementação de fábricas cada vez mais inteligentes. A indústria moderna quer satisfazer as exigências do consumo moderno com produtos personalizados de custos reduzidos, só possível com uma produção mais eficiente.

A Robótica está em permanente desenvolvimento, como tudo tem um início, o conceito de robô por exemplo, está presente no início da história, quando os antepassados fazem referência a mecanismos que ganham vida. Na civilização grega os primeiros conceitos de robôs encontrados, foram figuras com aparência humana ou animal. *Lenoardo DaVinci* contribuiu para o complexo mundo da robótica desenvolvendo uma extensa investigação no domínio da anatomia humana, que deu origem aos conceitos fundamentais para a criação de articulações mecânicas. A palavra robô [3], foi introduzida na linguagem inglesa em 1921 pelo escritor *Karel Capek* no seu livro de drama satírico, *Rossum's Universal Robots*. Neste trabalho, os robôs são máquinas que se assemelham a pessoas, mas trabalham incansavelmente. Inicialmente, estes robôs eram fabricados com fins lucrativos para substituir trabalhadores humanos, mas, no final, os robôs voltaram-se contra os seus criadores e aniquilaram toda a raça humana. A peça de *Karel Capek* é responsável pelas antigas ideias populares sobre robôs, incluindo a perceção de robôs como máquinas humanas dotadas de inteligência e personalidades individuais. Esta imagem foi reforçada pelo filme alemão *Metropolis* de 1926, pelo robô que anda, *Electro* e o seu cão *Sparko*, exibido em 1939 na Feira Mundial de Nova York, e pelo robô *C3PO* apresentado no filme *Star Wars* de 1977.

O termo robótica [3], refere-se ao estudo e utilização de robôs e foi pela primeira vez usado pelo cientista e escritor *Isaac Asimov* para descrever a ciência que estuda os robôs, em 1942, numa história intitulada *Runaround*. *Isaac* propôs a existência de três leis aplicáveis à robótica. A primeira lei, seria que um robô não pode ferir um ser humano nem permitir que sofra algum mal, na segunda lei, um robô deve obedecer às ordens que lhe sejam dadas pelos seres humanos,

exceção, no caso de contrariar a primeira lei. Na terceira lei, um robô deve proteger a sua própria existência, desde que essa proteção não entre em conflito com a primeira e segunda leis. *Issac Asimov* concebeu estas leis para que os robôs conseguissem conviver com os seres humanos sem que os robôs se revoltassem contra os seres humanos distinguindo o bem e o mal. Mais tarde acrescentou ainda a lei zero na qual dizia que um robô não pode fazer mal à humanidade nem permitir que tal aconteça. Estas leis são uma perspetiva ficcional e nos tempos de hoje o desenvolvimento de robôs não tem como principal objetivo serem imitações humanas muito menos ser outra forma de vida. A ficção científica teve um papel importante na contribuição de ideias para sistemas robóticos, no entanto o real desenvolvimento destes sistemas foi motivado pelo esforço de automatizar as operações industriais tendo como objetivo a realização de tarefas especificas ou programáveis.

O *Numeric Control* (NC), foi desenvolvido no final da década de 1940 e no início da década de 1950 [3]. Como o nome sugere, envolve o controlo das ações de uma máquina ferramenta por meio de números. Este sistema de controlo é baseado no trabalho original de *John Parsons*, que concebeu o uso de cartões perfurados que continham dados de posição para controlar os eixos de uma máquina ferramenta. O seu conceito foi demonstrado para a Força Aérea dos Estados Unidos, que passou a apoiar o projeto de pesquisa e desenvolvimento no, *Massachusetts Institute of Technology* (MIT). O projeto do MIT usou uma fresadora de três eixos para demonstrar o controlo desta através do NC em 1952. Os trabalhos subsequentes no MIT levaram ao desenvolvimento do *Automatically Programmed Tool* (APT), uma linguagem de programação para a máquina ferramenta.

Outro modo de controlo é realizado com base num teleoperador e é designado por *telerobotics,* que se baseia, num manipulador remoto, um dispositivo mecânico que traduz os movimentos do operador humano em movimentos correspondentes, realizados remotamente. Um uso comum de teleoperador é no trabalho de substâncias perigosas, como materiais radioativos. O ser humano pode permanecer seguro e orientar os movimentos do braço remotamente. Os trabalhos de desenvolvimento de teleoperação para manuseio de materiais radioativos remonta aos anos de 1940. Os dispositivos *telerobotics* foram usados pela Comissão de Energia Atômica a partir da mesma época. É a combinação de controlo numérico e de *telerobotics* que formam a base do robô moderno. O reconhecimento da criação destas duas tecnologias é atribuído a dois inventores [3], o britânico *Cynl Walter Kenward*, e o americano *George Devol*.

Um robô, é um manipulador mecânico, cujos movimentos são controlados por técnicas de programação muito semelhantes às usadas no controlo numérico.

O físico *Joseph Engelberger*, era fascinado pelos livros de *Asimov*. O desenvolvimento do primeiro robô industrial, aconteceu em 1959, por *George Devol* e *Joseph Engelberger* [3] [4].

Figura 2 - Desenvolvimento do primeiro robô industrial por Devol e Engelberger [4].

Figura 3 - Primeira instalação industrial de um robô pela *Unimation* [4].

Engelberger e *Devol* começaram a desenvolver planos e protótipos para um robô ajudante universal, o *Unimate,* Figura 2. Em 1962, a *Unimation Company* foi fundada e *Engelberger* tornou-se presidente da empresa. Na Figura 3, está representado o primeiro robô industrial usado numa linha de produção na fábrica *General Motors* em *Trenton*, *New Jersey*, que fez maçanetas para portas e janelas, luminárias e outros equipamentos para interiores de automóveis.

De entre muitas contribuições valiosas para o campo da robótica, podem-se salientar os trabalhos pioneiros da *Stanford University* e *do Stanford Research Institue* sobre linguagens de robôs orientadas por computador. Em 1973, desenvolveram a linguagem experimental *WAVE* [3], precedida pela linguagem AL [3], em 1974, outra linguagem projetada essencialmente para investigação e desenvolvimento. A linguagem do primeiro robô comercial foi a VAL [3], desenvolvida por *Victor Scheinman* e *Bruce Simano* para a *Unimation*. Esta linguagem foi utilizada pela primeira vez para programar o primeiro robô comercial *Programmable Universal Machine for Assembly* (PUMA) da *Unimation*, um robô de braço articulado relativamente pequeno, cujo desenvolvimento foi baseado em estudos de automação de montagem realizados pela *General Motors* (GM).

Na Figura 4, está representado o braço robótico desenvolvido em Stanford, para a montagem de conjuntos de pequenas peças. O construtor deste braço, foi *Victor Scheinman* e formou a empresa *Vicarm* para comercializar uma versão do braço para aplicações industriais.

Em 1978 a GM concluiu que 90% de todas as peças manuseadas durante montagens, pesavam cinco quilos ou menos. Na Figura 5, está representado o robô PUMA, um braço robótico adaptado às especificações da GM para uso em linhas de manuseamento de peças pequenas, que ocupava o mesmo espaço de trabalho que um operador humano.

Figura 4 - Braço de Stanford [4].

Figura 5 - PUMA [4].

O trabalho de Stanford sobre linguagens de robôs, e grande parte do trabalho subsequente realizado em robótica, é amplamente baseado em desenvolvimentos de tecnologia de computadores. Embora os computadores estivessem disponíveis no nascimento da indústria da robótica, somente no final da década de 1970 a economia estava apta para o uso de um pequeno computador, como controlador de robô.

Em 1982, a *International Business Machines* (IBM) desenvolve *a Manufacturing Language* (AML), uma linguagem de programação adaptada especificamente para aplicações robóticas.

Em 1984, nos Estados Unidos da América, é apresentado o *AdeptOne* (Figura 6), um robô do tipo *Selective Compliance Arm for Robotic Assembly* (SCARA) e a ABB nesse mesmo ano produz o robô IRB 1000 (Figura 7), que superava a velocidade de montagem da época, sendo até 50% mais rápido que os braços robóticos desenvolvidos até ao momento [4].

Figura 6 - Robô AdeptOne SCARA [4].

Figura 7 - Robô ABB IRB 1000 [4].

Figura 8 - Sistema de controlo ABB S4 [4].

Em 1992, a ABB, lançou o sistema de controlo aberto S4 (Figura 8), concebido para melhorar a interface homem-máquina e o desempenho dos robôs.

Em 1996, a KUKA, lançou o primeiro sistema de controlo de robôs baseado em *Portable Computer* (PC), em que foi possível, pela primeira vez, mover robôs em tempo real usando um

rato 6D num dispositivo de controlo. Este sistema apresentava uma interface de utilizador *Windows* para as tarefas de controlo e programação.

Em 1998, a ABB desenvolve o *FlexPicker* (Figura 9), um robô capaz de escolher 120 objetos por minuto ou pegar e soltar objetos a uma velocidade de 10 metros por segundo, ambas usando a tecnologia de processamento de imagem incorporada.

Em 1999, a KUKA [4], realiza os primeiros diagnósticos de robôs via Internet e a Reis Robotics [4] introduz um feixe laser integrado nos robôs que substitui a necessidade de um dispositivo de orientação externo para evitar possíveis colisões.

Em 2006, a Comau, introduziu o primeiro *Wireless Teach Pendant* (WiTP) (Figura 10), que realiza a programação sem as restrições das ligações por cabo aos robôs.

Figura 9 - Robô ABB IRB 360 FlexPicker [5].

Figura 10 - Comau *Wireless Teach Pendant* [4].

Figura 11 - Robô ABB IRB120 [6].

Em 2008, a FANUC, lançou um robô com capacidade de manipulação de uma carga útil de até 1200Kg [4]. Em 2009 a ABB, lançou o menor robô industrial multiusos, o IRB120 (Figura 11) [6], que pesava apenas 25kg e conseguia suportar uma carga útil de 3kg com um alcance de 580mm.

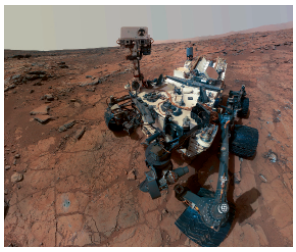
Hoje, quase todos os robôs introduzidos no mercado são controlados por computador, sendo muitas vezes auxiliados por diversos sensores e visão artificial. Os robôs colaborativos, atualmente, marcam cada vez mais presença na robótica. O seu sucesso instantâneo numa vasta gama de indústrias impulsionou a rápida inovação. Estes robôs incluem todos os tipos destinados a alguma interação humana durante a operação. Nem todos são usados em colaboração constante, mas possuem uma série de capacidades de segurança que permitem a interação entre ambos (ISO/TS 15066) [7], sendo que a segurança humana sempre foi uma das principais preocupações da robótica industrial.

2.2 Classificação dos robôs

A robótica pode ser descrita como uma ciência que lida com movimento inteligente de vários mecanismos robô, que podem ser classificados como: *i*) robôs manipuladores; *ii*) robôs veículo; *iii*) sistemas homem-robô; e *iv*) robôs de inspiração biológica.

Os robôs no geral, podem substituir os trabalhadores humanos em trabalhos difíceis, onde são confrontados com condições perigosas, ou até mesmo em trabalhos de elevada precisão. Os robôs manipuladores ou braços robóticos são representados por uma cadeia em série de corpos rígidos, chamados elos, ligados por juntas, responsáveis pelo seu movimento. Na Figura 12 a), está representado um braço robótico moderno da ABB [8], de seis eixos, projetado para alta produção, neste caso, para a indústria alimentar. Os robôs veículo podem ser concebidos para o meio terrestre, aquático ou aéreo. Na Figura 12 b), está representado o robô da *National Aeronautics and Space Administration* (NASA), *Curiosity* [8], usado em missões espaciais em Marte. Na Figura 12 c), está um sistema homem-robô, concretamente um operador a usar uma *haptic glove* em que a mão robótica imita o movimento da mão do operador e este recebe *feedback* ao toque [9]. Na Figura 12 d), está representado um robô de inspiração biológica, neste caso um robô de inspiração no ser humano, o robô Hubo, que ganhou o desafio de robótica *Defense Advanced Research Projects Agency* (DARPA), em 2015 [8].

a) Robôs Manipuladores.



b) Robôs Veículos.

c) Sistemas Homem-Robô.

d) Inspiração Biológica.

Figura 12 - Classificação de robôs.

2.2.1 Automação e robótica

A automação e a robótica são dois termos relacionados. Num contexto industrial, podemos definir automação, como uma tecnologia que se preocupa com o uso de sistemas mecânicos, eletrónicos e baseados em computador na operação e controlo de produção [3].

Exemplos desta tecnologia incluem linhas de operações, máquinas de montagem, sistemas de controlo e robôs.

A robótica é uma forma de automação industrial. Existem três grandes classes de automação industrial, a automação fixa, a automação programável e a automação flexível. Dos três tipos de automação, a robótica caracteriza-se mais com a automação programável.

Um robô industrial, é uma máquina programável de uso geral que possui certas características antropomórficas ou humanas. A característica humana mais típica dos robôs atuais são os seus braços móveis. O robô pode ser programado para mover o seu braço através de uma sequência de movimentos, a fim de executar uma tarefa. Repete-se o padrão de movimento até ser reprogramado para executar outra tarefa. Esta capacidade de reprogramação permite que os robôs sejam usados para uma enorme diversidade de operações.

Embora os próprios robôs sejam exemplos de automação programável, por vezes são usados em automação flexível, ou até em sistemas de automação fixa. Esses sistemas consistem em várias máquinas e/ou robôs a trabalhar em cooperação, sendo geralmente controlados por um computador ou por um controlador programável. Uma linha de produção que realiza soldas automóveis é um exemplo desse tipo de sistema. A linha de soldagem pode consistir em duas dúzias de robôs ou mais, sendo capaz de realizar centenas de soldas separadas em dois ou três locais diferentes de corpo. Neste caso cada robô recebe os comandos de um computador ou de um controlador programável, permitindo definir por estação que parte do processo de soldagem é da sua responsabilidade. Esta linha pode ser considerada um sistema de automação flexível de alta produção.

2.2.2 Configurações dos robôs manipuladores

A anatomia de um braço robótico, é semelhante à configuração física de um ser humano, constituído por corpo, braço e punho. A maioria dos braços robóticos industriais são montados numa base fixa, na qual o corpo está ligado à base e este ao braço, na extremidade do braço temos o punho. Os movimentos relativos entre estes componentes são assegurados por juntas e a uni-las estão os elos, membros rígidos, como representado na Figura 13. Os elos formam assim uma cadeia cinemática sendo as juntas responsáveis pelos movimentos dos elos.

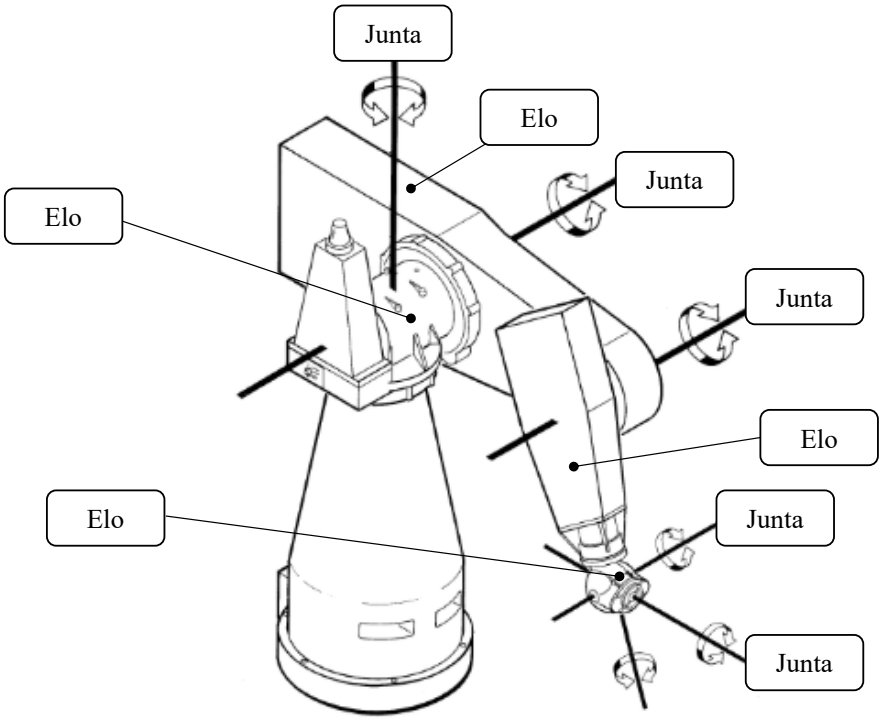


Figura 13 - Juntas e elos de um manipulador.

O número de movimentos individuais que um robô pode realizar, identifica a versatilidade do robô, denominando-se graus de liberdade e correspondendo ao número de juntas que um robô apresenta.

Os movimentos são assim assegurados por juntas que podem ser lineares (L), revolventes (V), torcionais (T) e rotacionais (R), como representado na Tabela 1.

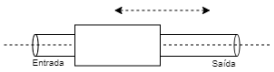
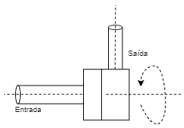
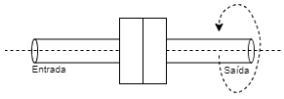
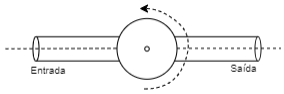
Tipo de junta		Notação	
Junta Linear		L	*Linear*
Junta Revolvente		V	*Revolving*
Junta Torcional		T	*Twisting*
Junta Rotacional		R	*Rotational*

Tabela 1 - Notações usadas para os diversos tipos de juntas de braços robóticos.

Na Tabela 2, está representada a constituição de um punho de robô com três graus de liberdade e uma analogia entre o punho robô e a punho do ser humano. É comum o punho de um robô estar associado a três graus de liberdade.

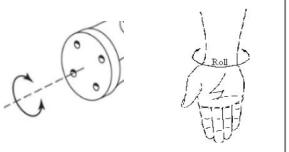
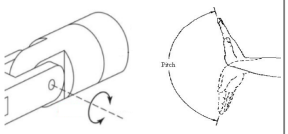
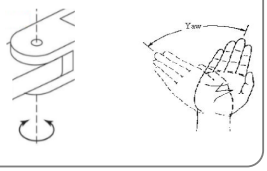
Eixo	Movimento	Descrição	Analogia
Roll	Rotacional	Rotação do punho em torno do eixo do braço do robô.	
Pitch	Inclinação	Orientação vertical ou inclinação do punho, se o eixo rotacional estiver na posição central corresponde à rotação do punho para cima e para baixo.	
Yaw	Orientação Horizontal	Se o eixo rotacional estiver na sua posição central corresponde à rotação do punho para a esquerda e para a direita.	

Tabela 2 - Descrição de 3 graus de liberdade num punho de robô.

Os movimentos dos robôs podem ser divididos em duas categorias, os movimentos do braço/corpo e os movimentos do punho. O espaço dentro do qual o robô pode movimentar a extremidade do punho é o seu volume de trabalho. Para definir o volume de trabalho, usa-se somente a extremidade do punho, desta forma evita-se a confusão que se estabeleceria na sua definição com órgãos terminais variados.

Na Tabela 3, estão representadas cinco configurações base de robôs relativamente à movimentação do corpo/braço bem como o seu volume de trabalho e notação utilizada na literatura.

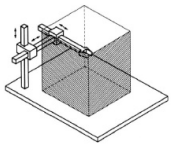
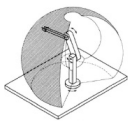
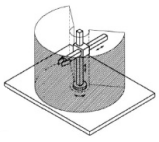
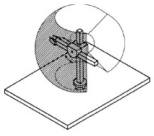
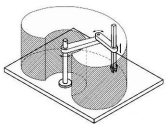
Cartesiana	Articulada	Cilíndrica	Polar	SCARA
				
LLL	TRR	TLL, LTL, LVL	TRL	RRT

Tabela 3 - Notações e volumes de trabalho de configurações de braço/corpo de robô.

Na Tabela 4, estão representados dois punhos de robôs com diferentes graus de liberdade e a respetiva notação aqui considerada.

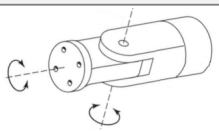
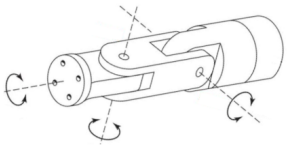
Configuração do punho	Notação
Punho de 2 eixos 	:RT
Punho de 3 eixos 	:TRT

Tabela 4 - Notações de configurações de punhos de robô com diferentes graus de liberdade.

Para a notação da configuração física de um braço robótico, consideram-se os movimentos do corpo/braço, começando pela junta mais próxima da base, continuando até à junta que liga ao punho. Este esquema de notação de juntas também pode incluir os movimentos do punho, nesse caso, começa-se na junta mais próxima da base e continua-se até ao suporte para o órgão terminal.

No caso do manipulador representado na Figura 13, tem a configuração TRR:TRT, ou seja, tem três graus de liberdade para o corpo e outros três graus de liberdade para o punho.

2.2.3 Sistemas de acionamento

A capacidade de movimentação de um robô, quer do seu corpo, braço e punho é fornecida pelo seu sistema de acionamento. Os sistemas de acionamento são compostos por atuadores, que são elementos capazes de converter energia, como elétrica, pneumática ou hidráulica em energia cinética, capacitando assim movimentos, em resposta a comandos que podem ser manuais ou automáticos. Os elementos motrizes usados nos sistemas robóticos podem ser classificados em três tipos: *i*) acionamento hidráulico; *ii*) acionamento pneumático; e *iii*) acionamento elétrico.

Os sistemas de acionamento hidráulicos são apropriados em sistemas que requerem grande capacidade de carga e velocidade, permitindo além disso uma boa precisão de regulação. Os sistemas de acionamento elétricos não permitem tanta força e velocidade como os hidráulicos, mas, possuem uma melhor precisão e repetibilidade, ocupando habitualmente menos espaço, sendo utilizados tipicamente em trabalhos de montagem. Os sistemas de acionamento pneumáticos são usados em robôs pouco sofisticados com poucos graus de liberdade, que executam movimentos simples, rápidos e de pouca precisão em comparação com os anteriores.

2.2.4 Sistemas de controlo

Os sistemas de controlo para robôs têm a tarefa de executar a sequência planeada de movimento. Um dos métodos possíveis [10], é o controlo de posição que se baseia apenas na posição do elemento terminal como parâmetro. Este controlo é o mais básico e a atuação pode ser ponto a ponto, ou em modo contínuo, designando-se neste caso de controlo contínuo de trajetória. Se para além da posição, considerarmos a velocidade, estamos perante um controlo cinemático, em que a posição e a velocidade são os parâmetros de controlo. O controlo dinâmico, realiza-se quando o sistema de controlo também tem em conta as forças exercidas pelos elementos associados ao robô, tendo como parâmetros de controlo a posição, a velocidade e a força. O controlo adaptativo, atualiza o modelo do processo com base nos resultados de ações anteriores e controla a posição, a velocidade e as propriedades dinâmicas dos elementos do robô, considerando as suas variações com a posição.

2.2.5 Estabilidade e precisão de movimentos

A estabilidade é definida como uma medida das oscilações que ocorrem quando o braço robótico se movimenta de um ponto para outro. Um robô com elevada estabilidade, movimenta-se com um tempo de resposta curto e sem oscilações durante ou no fim do movimento.

A precisão de movimentos é definida em função de três parâmetros, resolução espacial, exatidão e repetibilidade. A resolução espacial diz respeito ao mais pequeno incremento de movimento em que o robô pode dividir o seu volume de trabalho. A exatidão refere-se à capacidade de o robô ser programado para atingir um ponto alvo e a repetibilidade diz respeito à capacidade que o robô tem de voltar ao ponto programado.

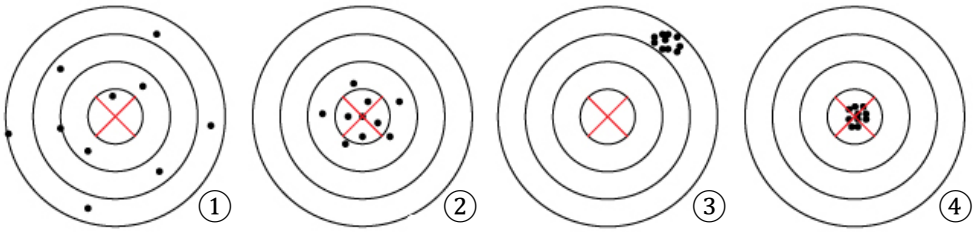


Figura 14 - Diferentes casos de repetibilidade e precisão.

Na Figura 14, no caso ①, representa-se um exemplo de uma fraca repetibilidade e de fraca precisão. Em contraste, no caso ④, representa-se uma elevada repetibilidade e precisão. No caso

②, é representada uma situação de boa precisão, mas fraca repetibilidade e no caso ③, é representada uma boa repetibilidade, mas fraca precisão.

2.2.6 Sensores e elementos terminais

A utilização de sensores é importante para um funcionamento eficiente e preciso dos robôs. Os sensores para sistemas robóticos podem ser geralmente divididos em duas categorias [11], *i*) *proprioceptive sensors*, sensores que avaliam os estados internos dos mecanismos dos robôs como posições, velocidades e forças nas juntas dos robôs; e em *ii*) *exteroceptive sensors,* sensores que fornecem ao controlador dados e informação, por exemplo sobre o ambiente onde o robô atua, sensores de força, tácteis, de proximidade e distância, de voz e visão.

Os sensores convertem a variável física medida num sinal elétrico que pode ser avaliado por computador. Na robótica as variáveis, posição, velocidade, força e binário são as mais controladas e com a utilização de transdutores estas variáveis podem ser convertidas em sinais elétricos. Com base neste princípio os sensores podem ser ainda divididos em sensores elétricos, em que a variável física é transformada diretamente numa medida elétrica, sensores eletromagnéticos, que usam o campo magnético para conversão da variável física e sensores óticos, que utilizam luz na conversão dos sinais.

Da mesma forma que os robôs manipuladores são semelhantes ao braço humano, as garras, usadas como elemento terminal nos robôs recriam as funções da mão humana. Na maioria dos casos, as garras de robôs manipuladores são consideravelmente mais simples do que a mão humana, que englobando o pulso e os dedos, têm ao todo 22 graus de liberdade [11].

Embora estejam comercialmente disponíveis várias garras, por vezes e para tarefas específicas pode ser necessário desenvolver uma garra especial de modo a satisfazer os requisitos dessa tarefa. Assim, o elemento terminal de um robô pode então ser do tipo garra ou ferramenta e permite a realização de uma ou várias tarefas específicas.

2.2.7 Aplicações de manipuladores industriais

A indústria nos dias de hoje, não é imaginada sem manipuladores industriais pois são um elemento essencial na maioria dos processos de fabrico, como os representados na Figura 15 e na Figura 16. Estes braços robóticos podem ser divididos em três grupos [11]. No primeiro grupo são classificados os robôs industriais que têm o papel de *Master*. No segundo grupo temos os robôs

que são classificados como *Slaves*. E no terceiro grupo incluímos os robôs industriais que são usados em todas as aplicações especializadas.

Os robôs *Master*, podem ser encontrados nos processos de produção, como soldadura, pintura, maquinação e montagem. A soldadura com robôs representa uma das aplicações mais frequentes, caracteriza-se pela velocidade e precisão e encontramos o maior número destes robôs na indústria automóvel. São também frequentemente usados em ambientes agressivos, como a pintura a spray, nas aplicações de maquinação e usados para montagem industrial. As linhas de produção e montagem são cada vez mais dominadas por robôs, e não só na indústria automóvel, mas em muitas outras indústrias o uso de manipuladores tornou-se indispensável.

A *General Motors* [12] utiliza aproximadamente 16 000 robôs para trabalhos como soldadura, pintura, carregamento de máquinas, transferência de peças e montagem.

Figura 15 - Utilização de manipuladores na indústria automóvel [13].

Figura 16 - Manipulador usado para paletização [14].

Os robôs que desempenham o papel de *Slave* são utilizados em aplicações industriais como o manuseamento de peças ou materiais, paletização, fundição e fixação. Nestas aplicações, o papel de *Master* é desempenhado por uma máquina de controlo numérico e os manipuladores desempenham tarefas como o manuseamento de materiais, onde as tarefas são repetitivas ou potencialmente perigosas. A paletização, é um exemplo de uma tarefa repetitiva com dimensões industriais, entregue a manipuladores.

Os manipuladores industriais são também utilizados em aplicações especializadas como a garantia de qualidade, inspeção, testes, manutenção, reparação, indústria alimentar, têxtil e construção.

2.3 Possibilidades e exemplos de controlo de manipuladores

Os *Programmable Logic Controller* (PLC), são sistemas que desempenham uma função de controlo, lógico, aritmético, comunicação, temporização, contagem, sequência, realizam as decisões baseadas em informações de entrada e atuam a manipulação lógica das saídas. Estes controladores são autómatos, podem ser constituídos por diversos módulos e têm um custo superior, comparativamente a microcontroladores. Os autómatos variam em tamanho e hardware, mas a sua programação incide maioritariamente no uso de uma ou mais formas de programação, tais como, linguagem *ladder*, diagrama de contatos, lista de instruções, bloco de funções e texto estruturado (norma IEC 61131-3) [15].

Os autómatos são então uma opção para realizar o controlo de braços robóticos tendo, no entanto, um custo elevado. As suas grandes vantagens são a modularidade, robustez, grande imunidade a ruídos de origem eletromagnética, fiabilidade e facilidade de programação no ambiente industrial.

Os microcontroladores, são circuitos integrados programáveis semelhantes a um computador com CPU, memória, portas de entrada, portas de saída, conversores, entre outros periféricos, tendo vários modelos com diversos periféricos [16]. Têm como vantagem um custo bastante mais reduzido em comparação com os PLC.

Concretamente o uso de qualquer uma das opções, autómatos ou microcontroladores pode-se observar na concretização de vários projetos já realizados numa diversidade de soluções. Um dos projetos, foi a automatização de um sistema de separação de peças com recurso a um tapete rolante, um robô e processamento de imagem. Este projeto foi realizado com recurso a um microcontrolador Raspberry PI e concilia o processamento de imagem ao robô para separar peças em movimento [17]. Outro projeto recorreu a um autómato para simular um armazém automático onde o PLC e visão artificial controlam um robô de acionamento pneumático de 3 eixos para armazenar automaticamente peças [18]. O processamento de imagem em paralelo com o controlo de um manipulador é uma solução em constante desenvolvimento e com bastantes vantagens no controlo dinâmico. Para realizar o controlo de um manipulador com recurso a visão artificial são necessárias câmaras e/ou sensores. De igual modo pode-se utilizar algoritmos específicos de redes neuronais [19].

2.4 Métodos de Comunicação

Os protocolos de comunicação são um conjunto de regras estabelecidas que possibilitam a transmissão, receção e formatação de dados entre diversos dispositivos independentemente das

suas diferenças. Na transmissão de informação digital uma sequência de bits pode ser enviada através de um canal de comunicação, em série ou em paralelo.

Na comunicação via canal série, é enviado um bit de cada vez, ou seja, a informação é enviada e recebida sequencialmente. Na comunicação via canal em paralelo, são enviados conjuntos de bits em simultâneo, que resultam numa maior velocidade de transmissão da informação, no entanto requer um número superior de fios. Admitindo que os canais necessitam do mesmo tempo para transmitir 1 bit, a comunicação em paralelo necessita de tantos fios de ligação quanto o número de bits que se quer transmitir.

Cada protocolo de comunicação tem características específicas. A taxa de comunicação tem como unidade, bits por segundo e representa a velocidade da comunicação. Por exemplo uma comunicação assíncrona com 9600 bps envia um bit em 0,0001 segundos. Todos os dispositivos, ligados na mesma linha de comunicação, devem estar com a mesma taxa de comunicação, principalmente comunicações assíncronas que não usam *clock* como referência. Com uma taxa de comunicação diferente nos dispositivos a comunicar, dificilmente os dados são transmitidos e recebidos corretamente.

O método de transmissão síncrono, é um método de comunicação que depende de um sinal de *clock*, um sinal de relógio que define a cadência por medição de tempos, ou seja, cada bit ou conjunto de bits enviado depende de um pulso do *clock*, tendo como principal vantagem a velocidade de transmissão de dados, no entanto é necessário um fio extra para este sinal. O método de transmissão assíncrono, não precisa de um sinal de *clock*, portanto o número de fios necessários é menor. O envio dos dados, neste método é mais suscetível a erros, por isso são necessários alguns parâmetros para garantir o envio correto de uma mensagem. Um parâmetro essencial da comunicação assíncrona, é o *Baud Rate* que especifica a velocidade de receção e envio, por isso é muito importante que os dispositivos a comunicar utilizem o mesmo ritmo.

Outro ponto importante é a codificação usada para estabelecer e reconhecer as sequências binárias, para assegurar uma correta extração dos dados a transmitir. Sinais binários com sequências longas de zeros ou uns tornam difícil uma sincronização constante, visto que o recetor só deteta as transições. A codificação a dois níveis, *High* e *Low*, designa-se por *Non Return to Zero* (NRZ). A codificação *Return to Zero* (RZ), define claramente onde começa cada bit, no entanto requer três níveis de sinal, *High, Zero* e *Low* e maior largura de banda. Uma outra codificação é a *Manchester*, uma solução em que o sinal codificado é conseguido através de um sinal de *clock* auxiliar com o dobro da frequência do sinal a enviar, sendo o sinal codificado, obtido através de um *exclusive or* (XOR), da sequência a enviar com o *clock* auxiliar. Torna-se assim fácil o recetor extrair o *clock* da mensagem recebida.

A transmissão pode ainda ser classificada de três maneiras diferentes, transmissão *simplex, half-duplex e full-duplex*. A transmissão *simplex*, indica que a transmissão é unidirecional, apenas efetua a transmissão ou receção da mensagem. A transmissão *half-duplex*, caracteriza uma transmissão bidirecional, em que a transmissão e a receção não são efetuadas em simultâneo. A transmissão *full-duplex* indica que a transmissão é feita em ambos os sentidos, em simultâneo e ambos os dispositivos podem enviar e receber mensagens.

Os transmissores são os dispositivos da interface que apenas transmitem a informação, os que apenas recebem informação têm o nome de recetores. Os dispositivos que realizam ambas as funções, a transmissão e a receção, são designados de transmissores-recetores.

Os valores de tensão usados nos protocolos são outra característica importante na comunicação, pois identificam os estados lógicos da informação, balizando determinados valores de tensão, para o nível alto, *High* e para o nível baixo, *Low*. As recomendações RS-232, RS-422 e RS-485 clarificam diversos parâmetros em interfaces de transmissão digital série entre equipamentos, referidas na Secção 2.4.2.

2.4.1 Controlo de erros

Na transmissão de informação digital a deteção e correção de erros é essencial para assegurar uma boa comunicação, as informações transmitidas e recebidas têm de ser coerentes para assegurar uma comunicação entre os dispositivos.

Os mecanismos para controlo de erros consistem num controlo em antecipação ou no controlo por retroação. O controlo em antecipação baseia-se na deteção, na localização e correção dos erros, no entanto não são usados em sistemas industriais uma vez que exige muitos bits extra de dados. O controlo por retroação deteta erros na informação recebida e desencadeia um pedido de retransmissão ao emissor da mensagem, sendo o mais usado em sistemas de automação industriais.

Um dos métodos de deteção de erros é o Bit de Paridade. Este método consiste na avaliação da paridade do número de uns de uma palavra. Se a palavra for formada por um número par de uns o bit de paridade a acrescentar no fim da palavra toma o valor de 0, formando uma nova palavra com um bit extra. Pelo contrário se a palavra for formada por um número ímpar de uns, o bit de paridade a acrescentar toma o valor de 1. Este método permite apenas a deteção de erro num só bit.

Para erros em 2 ou conjuntos de mais bits utiliza-se o *Block Check Character* (BCC) ou palavra de teste longitudinal [20], que consiste no acrescento da palavra BCC formada pelo bit de paridade

proveniente dos bits organizados em colunas. Estas duas técnicas de deteção de erros conjugadas, falham se forem registados erros simultâneos nas mesmas linhas e colunas.

O *Cyclic Redundancy Check* (CRC) [20], ou teste de redundância cíclica permite uma maior eficácia para a deteção de erros. A palavra CRC com n bits é construída para ser anexada à mensagem que se pretende transmitir. A palavra CRC constrói-se começando por anexar n zeros à direita da mensagem a transmitir, depois divide-se por um polinómio gerador de CRC e dessa divisão, o resto representa a palavra CRC a acrescentar à mensagem a transmitir. Na receção da mensagem o teste é feito pela divisão da mensagem recebida, pelo gerador de CRC, em que se o resto for nulo não existem erros e caso contrário, existem.

Consoante os protocolos de comunicação, pode-se ter um destes métodos de controlo de erros ou uma combinação deles.

2.4.2 Comunicação com fios

A *Electronics Industries Association* (EIA), realiza a normalização para interfaces de comunicação série na forma de recomendações, *Recommended Standards* (RS). Estas recomendações são adotadas pelos fabricantes a nível mundial e funcionam quase como normas que apresentam especificações a nível das ligações físicas. Para a comunicação apenas entre dois equipamentos recorre-se à interface de transmissão digital série RS-232. Com as interfaces RS-422 e RS-485 pode-se aumentar o número de recetores, aumentando as distâncias entre equipamentos. Na interface RS-232 a transmissão é feita através de condutores, num sistema de comunicação série apenas entre dois participantes, com sinais elétricos binários em tensão, em que o número total de condutores para a ligação pode ser superior a vinte. Os circuitos desta interface podem ser circuitos de dados, circuitos de controlo e circuitos de cadência para transmissão assíncrona. Nesta interface o valor lógico 0, para sinais de dados, corresponde a um nível de tensão de 3V a 15V e para sinais de controlo e cadência, corresponde ao nível de tensão de -3V a -15V. O valor lógico 1, para sinais de dados, corresponde a um nível de tensão de -3V a -15V e para sinais de controlo e cadência, corresponde ao nível de tensão de 3V a 15V. Os parâmetros de transmissão como o número de bits de dados, bit de paridade e stop bit são assegurados na lógica interna dos equipamentos terminais através de unidades U*niversal Asynchronous Receiver-Transmitter* (UART). Estas unidades consistem em circuitos integrados ou periféricos dos microcontroladores ou microprocessadores para o controlo de equipamentos.

A interface com a recomendação RS-422, define para o valor lógico 1, o nível de -2V a -6V e para o valor lógico 0, o nível de 2V a 6V. A interface RS-485, define para o valor lógico 0, o nível

de -1,5V a -6V e para o valor lógico 1, o nível de 1,5V a 6V. Em ambas são usadas tensões diferenciais.

Na comunicação série através da UART, o funcionamento é simples, o pino de transmissão, *Tx*, envia um pacote de bits que será interpretado bit a bit pelo recetor, na entrada *Rx*. Cada pacote enviado pode conter bits que indicam o início da mensagem, stop bits para indicar o final da mensagem, bits de dados e bits de paridade para detetar a receção de erros. A ligação, é feita pelos pinos *Rx*, *Tx* e *GND*. A interligação dos pinos *Rx* e *Tx* entre o transmissor e o recetor pode ser variada o que classifica a transmissão como podendo ser *simplex*, *half-duplex* ou *full-duplex*.

Para protocolos síncronos, pode-se usar a comunicação *Inter-Integrated Circuit* (I^2C), que funciona com a hierarquia *Master/Slave* em que o dispositivo mestre coordena os dispositivos escravos. Este protocolo usa os pinos *Serial Clock* (SCL) e *Serial Data* (SDA). Neste protocolo em modo mestre é gerado um *clock* e é iniciada a comunicação com os escravos, em modo escravo o *clock* é recebido e o escravo pode responder quando endereçado por um mestre.

Outra comunicação síncrona é a *Serial Peripheral Interface* (SPI), funcionando também com a hierarquia *Master/Slave*. Este protocolo usa os pinos *Master Output Slave Input* (MOSI), *Master Input Slave Output* (MISO), *Serial Clock* (SCLK) e *Slave Select* (SS). Neste protocolo o mestre gera um *clock* e seleciona através do pino SS com qual dispositivo será efetuada a comunicação. De seguida os dados são enviados para o dispositivo de destino pelo pino MOSI e então o dispositivo escravo envia uma resposta ao mestre pelo pino MISO.

2.4.3 Comunicação sem fios

A transmissão de dados via radio constitui redes que se designam por *Wireless Local Area Networks* (WLAN). Estas redes consistem na transferência de dados sem a utilização de cabos. As distâncias envolvidas podem ser curtas ou longas dependendo do protocolo e da infraestrutura. O termo comunicação sem fio foi utilizado com duas formas na história da comunicação, inicialmente, foi utilizado para definir o sistema que passou a ser chamado de rádio transmissor, em 1920, e em 1980, o termo ressurgiu, sendo utilizado para qualquer tipo de tecnologia que fizesse comunicações sem utilizar fios.

A comunicação Wi-Fi, é um conjunto de especificações, para redes locais sem fio, baseada no padrão IEEE 802.11. Com a tecnologia Wi-Fi, é possível implementar redes que ligam computadores e outros dispositivos compatíveis. Estas redes não exigem o uso de cabos sendo a transmissão de dados feita por radiofrequência. As especificações IEEE 802.11, Ethernet, não são muito diferentes das especificações IEEE 802.3, as tradicionais redes com fio, essencialmente, o

que muda de um padrão para o outro são as suas características de ligação física, em que um tipo funciona com cabos e o outro, por radiofrequência. O padrão 802.11 estabelece normas para a criação e para o uso de redes sem fio. Como existem inúmeros serviços que podem utilizar sinais de rádio, é necessário que cada um opere de acordo com as exigências estabelecidas.

Os dispositivos, chamados *station* (STA), conectam-se a *Access Point* (AP), que fornecem o acesso. Quando um ou mais STA se conectam a um AP, tem-se, uma rede, que é denominada, *Basic Service Set* (BSS). Por questões de segurança e pela possibilidade de haver mais de um BSS em determinado local, é importante que cada um receba uma identificação denominada, *Service Set Identifier* (SSID). O SSID, é o nome dado a cada rede sem fios. Nas redes Wi-Fi, basta a qualquer dispositivo ter compatibilidade com a tecnologia para se conectar à rede. Para evitar problemas de segurança estes tipos de redes têm vários sistemas de ligação. Um deles é o *Wired Equivalent Privacy* (WEP) e consiste num mecanismo de autenticação que funciona, de forma aberta ou restrita por um uso de chaves de rede. Na forma aberta, a rede aceita qualquer dispositivo que solicite conexão. Na forma restrita, é necessário que cada dispositivo solicitante forneça uma chave de rede. O *Wired Protected Access* (WPA), tal como o WEP, também se baseia na autenticação dos dados da rede, mas de maneira mais segura. A base está num protocolo chamado *Temporal Key Integrity Protocol* (TKIP), que ficou também conhecido como WEP2. Uma chave de 128 bits é utilizada pelos dispositivos da rede e combinada com o *MAC Address*, um endereço único atribuído a cada dispostivo. A chave é trocada periodicamente, ao contrário do WEP, que é fixo, e a sequência definida na configuração da rede é usada, para o estabelecimento da conexão.

O WPA2 utiliza o *Advanced Encryption Standard* (AES), sendo ainda mais segura que as anteriores.

Outra comunicação sem fios é o Bluetooth, um padrão global de comunicação, de baixo consumo de energia, que permite a transmissão de dados entre dispositivos, desde que estejam próximos. A transmissão de dados é através de radiofrequência e permite que um dispositivo detete o outro, sendo necessário apenas que ambos estejam dentro do limite de proximidade [21].

2.5 Braços Robóticos Adotados

De forma a validar a Interface Remota Genérica para Braços Robóticos (IRGBR) a ser construída, serão necessários braços robóticos para controlar. No laboratório de automação, estava disponível um braço robótico, não usado durante bastante tempo, que foi uma primeira abordagem para a realização do projeto. Este braço robótico é o Mentor, um robô comercializado em 1985.

De forma a analisar, estudar e validar mais braços robóticos surgiu a oportunidade de colaboração com o Departamento de Mecânica do ISEL, ao ceder um robô multifuncional mais atual, comercializado a partir de 2016, o Dobot Magician.

Estes dois braços robóticos são analisados nas próximas Secções e serão adaptados para validação do projeto a desenvolver.

2.5.1 Braço Robótico Mentor

O Mentor, é um robô manipulador ou braço robótico, comercializado em 1985, com uma garra fixa como elemento terminal, representado na Figura 17. Este braço robótico tem uma configuração base articulada e é semelhante ao braço humano, devido à sua constituição.

O Mentor é constituído por cinco juntas, que representam cinco graus de liberdade como representado na Figura 18. A junta um (J1), corresponde ao movimento da cintura do braço robótico e é uma junta de torção. A junta dois (J2) e a junta três (J3), correspondem respetivamente ao movimento do ombro e do cotovelo, sendo ambas juntas rotacionais. A junta quatro (J4) e a junta cinco (J5) representam o movimento do pulso, sendo a junta quatro rotacional e a junta cinco de torção. A notação da configuração articulada do braço robótico Mentor é TRR:RT, de acordo com a Tabela 1.

Figura 17 - Mentor (no laboratório de robótica no ISEL).

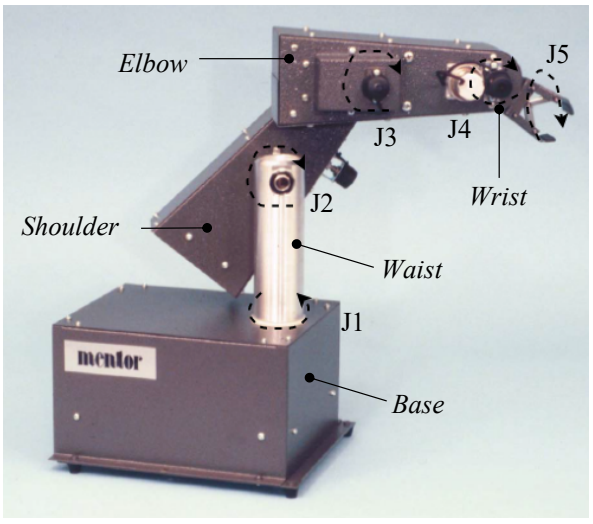


Figura 18 - Constituição do braço robótico Mentor.

O Mentor é composto por base, cintura (*waist*), ombro (*shoulder*), cotovelo (*elbow*) e pulso (*wrist*). Estes elementos rígidos, são os elos que unem as 5 juntas responsáveis pelas movimentações do Mentor. Na Tabela 5, pode-se observar as especificações técnicas, resumidas, mais relevantes do Mentor.

Nome	Mentor
Ano	1985
Movimento dos Eixos	(Junta: Eixo, Alcance, Comprimento elo) Junta 1 (J1): Cintura, movimento angular 210º, 185mm Junta 2 (J2): Ombro, movimento angular 180º, 165mm Junta 3 (J3): Cotovelo, movimento angular 180º, 150mm Junta 4 (J4): Pulso, movimento angular 140º Junta 5 (J5): Pulso, movimento angular 320º
Garra terminal	Abertura máxima: 45mm; Força máxima: 10N
Carga útil máxima	1000g
Alcance máximo	428mm
Repetibilidade de posição	2mm
Dimensões da Base	320 x 270 x 189 mm
Fonte de energia	230V, 50Hz
Sistema de controlo	8bit
Comunicação	20 pinos dedicados

Tabela 5 - Especificações técnicas do braço robótico Mentor.

A garra fixa acrescenta um movimento extra, ao conjunto, podendo abrir e fechar. A garra é o elemento terminal fixo e permite ao Mentor carregar no máximo 1Kg, podendo abrir 45mm para agarrar o que pretender. A garra tem uma força máxima de 10N e pode alcançar 428mm medidos a partir da sua base. O Mentor é um braço robótico de acionamento elétrico. O campo de ação é delimitado pelas movimentações angulares das suas juntas. Este braço robótico funciona com a passagem de dados a 8 bits, por isso para cada junta e para a abertura e fecho da garra só estão disponíveis o conjunto de valores de 0 a 255 como descrito na Tabela 6.

Eixo	Movimento	8 bits
Junta 1 (J1)	Movimento angular máx. 210º	0 a 255
Junta 2 (J2)	Movimento angular máx. 180º	0 a 255
Junta 3 (J3)	Movimento angular máx. 180º	0 a 255
Junta 4 (J4)	Movimento angular máx. 140º	0 a 255
Junta 5 (J5)	Movimento angular máx. 320º	0 a 255
Garra	145mm - Aberta/Fechada - 0mm	0 a 255

Tabela 6 - Amplitude máxima de movimento de cada eixo do Mentor com 8 bits.

A repetibilidade de posição do Mentor é de 2mm, ou seja, reflete a capacidade que o braço robótico tem de voltar ao ponto programado.

Na base do braço robótico, podemos encontrar a interface de comunicação disponível do Mentor, situada na parte traseira. Este manipulador esteve parado mais de uma década e com vista à sua recuperação e utilização com novos controladores e interfaces foram feitas alterações de modo a substituir os sinais da placa *Peripheral Component Interconnect* (PCI) original. Devido ao tempo de inatividade e às alterações de operação feitas no Mentor, algumas das características do Mentor podem não ser atuais, sendo sempre preciso calibrar o Mentor para o seu uso. Isto afeta a amplitude angular dos movimentos do Mentor.

O conjunto original [22], consistia numa ligação do Mentor à placa PCI que era alimentada e fornecia a ligação ao computador e a um simulador. Esta placa, com a recuperação feita, pode ser substituída por outro periférico, respeitando os 20 pinos dedicados para comunicação e funções correspondentes do Mentor.

A comunicação com o Mentor obedece a uma ligação de 20 pinos dedicados via *flat cable*. Na Figura 19, está representada a comunicação disponível com o Mentor.

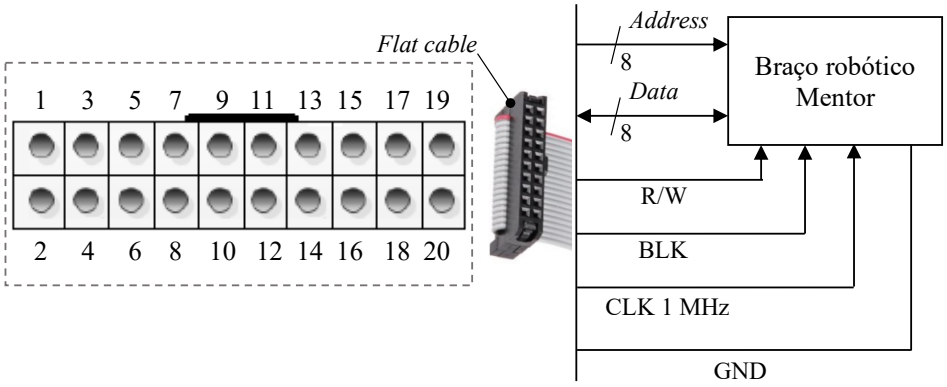


Figura 19 - Interface de comunicação com o Mentor.

A ligação com o Mentor tem dedicadas, 11 entradas digitais, 1 saída digital e 8 bidirecionais, como resumido na Tabela 7.

O Mentor é controlado a partir de incrementos ou decrementos atribuídos às suas juntas, ou seja, pela entrada direta de dados, sendo o seu sistema de controlo apenas da posição adotada.

Para o comando do Mentor, com a substituição da placa PCI, cada função do braço robótico tem um endereço específico na zona de memória, em que por exemplo para se movimentar o Mentor, é necessário escrever o endereço (*address*) e os 8 bits de dados (*data*) para a correspondente movimentação da junta. Para todos os comandos é necessário escrever o endereço e os dados, que são delimitados pelos sinais RW e BLK, com RW no nível lógico a '1' inicia-se uma leitura e

termina com o valor lógico a ‘0’. O sinal BLK controla a escrita. O sinal de *clock* de 1 MHz é responsável pela comunicação síncrona.

Mentor	Nome	Pinos								
Inputs	*Address*	*address*	0	1	2	3	4	5	6	7
		flat cable	2	4	6	8	10	14	16	18
	R/W	*flat cable*	17							
	BLK	*flat cable*	12							
	CLK 1 MHz	*flat cable*	20							
Outputs	GND	*flat cable*	19							
Inputs/Outputs	*Data*	*data*	0	1	2	3	4	5	6	7
		flat cable	1	3	5	7	9	11	13	15

Tabela 7 - Entradas e saídas da Interface de comunicação do Mentor.

Para comandar o Mentor, com a adaptação feita, são enviados comandos via porta série/USB desde o PC para uma interface, que por sua vez gera os correspondentes endereços e valores de posição em binário (8 bits) e aplica-os nos barramentos de dados e endereços da eletrónica de controlo do Mentor. Qualquer programa capaz de enviar informação em caracteres *American Standard Code for Information Interchange* (ASCII) via porta série, pode ser utilizado.

2.5.2 Braço Robótico Dobot Magician

O Dobot Magician, é um robô manipulador ou braço robótico, multifuncional, comercializado a partir de 2016, que permite a utilização de diferentes elementos terminais. Este braço robótico tem uma configuração base, articulada, como o Mentor. O equipamento terminal do Dobot permite a realização de tarefas distintas como impressão 3D, gravação laser, escrita, desenho e sucção a vácuo, consoante o elemento terminal em uso.

O Dobot Magician, representado na Figura 20 e Figura 21, é constituído por quatro juntas, que representam quatro graus de liberdade. A junta um (J1), corresponde ao movimento da base do braço robótico e é uma junta de torção. A junta dois (J2) e a junta três (J3), correspondem respetivamente ao movimento do braço traseiro e do antebraço, sendo ambas juntas rotacionais. A junta quatro (J4), corresponde ao movimento de torção do acessório terminal. A notação da configuração articulada do braço robótico Dobot Magician é TRR:T, segundo a Tabela 1.

Figura 20 - Dobot Magician (no laboratório de automação no ISEL).

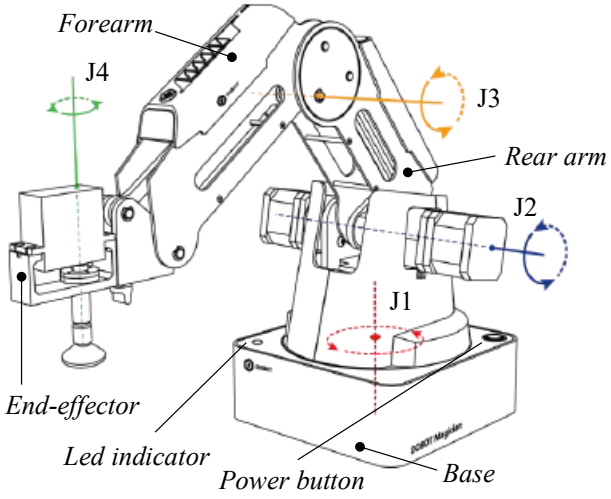


Figura 21 - Constituição do braço robótico Dobot Magician.

O Dobot Magician é composto por base, braço traseiro (*rear Arm*), antebraço (*forearm*) e um acessório terminal removível (*end-effector*). Estes elementos rígidos são os elos que unem as quatro juntas responsáveis pelas movimentações do Dobot Magician. Com a ventosa a vácuo como elemento terminal, o Dobot Magician, consegue carregar no máximo 500g e pode alcançar 320 mm medidos a partir da sua base.

Nome	Dobot Magician.
Ano	2016
Movimento dos Eixos	(Junta: Eixo, Alcance, Velocidade máxima) Junta 1 (J1): Base, -90º a 90º, 320º/s Junta 2 (J2): Braço traseiro, 0º a 85º, 320º/s Junta 3 (J3): Antebraço, -10º a 90º, 320º/s Junta 4 (J4): Acessório terminal, -90º a 90º, 480º/s
Acessório Terminal	Ventosa a vácuo, Garra, Laser, Caneta, Impressora 3D
Carga útil máxima	500g
Alcance máximo	320mm
Repetibilidade de posição	0.2mm
Dimensões da base	158mm x 158mm
Fonte de energia	230V, 50Hz
Sistema de controlo	Controlador integrado Dobot, DobotStudio
Comunicação	UART, USB, Wi-Fi, Bluetooth
Peso líquido	3.4Kg

Tabela 8 - Especificações técnicas do Dobot Magician.

Na Tabela 8, pode-se observar as especificações técnicas, resumidas, mais relevantes do Dobot Magician e na Tabela 9 encontra-se uma lista do equipamento terminal disponível.

Acessórios terminais	Especificações
Impressora 3D	Tamanho máximo de impressão: (LxWxH) 150x150x150mm. Material de impressão 3D: PLA; Resolução: 0.1mm.
Laser	Consumo: 500mw; Tipo: 405nm (*blue laser*); Energia: 12V
Caneta	Diâmetro: 10mm
Ventosa a vácuo	Diâmetro ventosa de sucção: 20mm; Pressão: -35Kpa
Garra	Alcance: 27.5mm; Tipo: Pneumática; Força: 8N

Tabela 9 - Acessórios terminais do Dobot Magician [23].

O Dobot Magician é um braço robótico de acionamento elétrico com elementos terminais removíveis de acionamento pneumático, como a ventosa a vácuo ou a garra.

As coordenadas de origem (*Origin Coordinate*) e a localização do ponto terminal (*End-effector Center*), estão representadas na Figura 22.

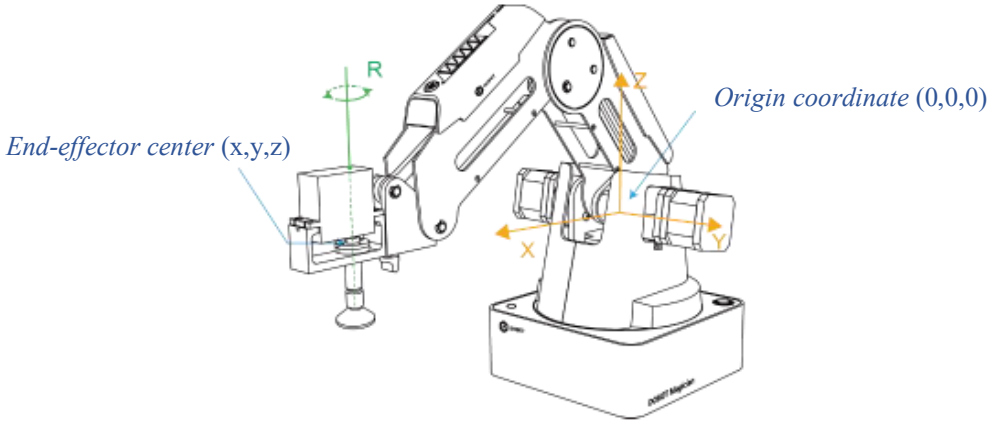
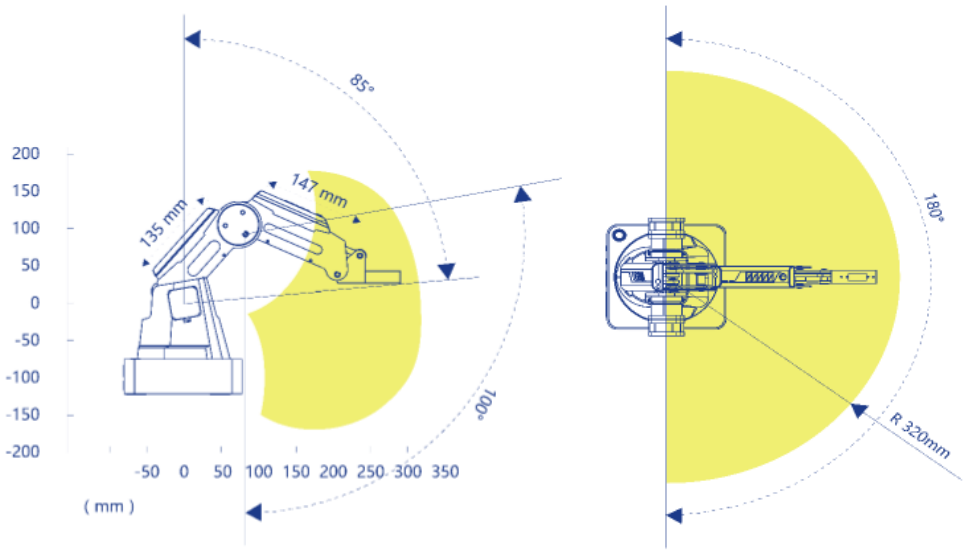


Figura 22 - Localização das coordenadas de origem e do ponto terminal do Dobot Magician no plano XYZ [24].

Uma vez observadas as localizações da origem e do ponto terminal de atuação, podemos definir o campo de atuação do Dobot Magician.

A Figura 23, representa o campo de ação do Dobot Magician em relação às coordenadas x e z, em a), e o campo de atuação em relação à coordenada y, em b). O Dobot Magician tem uma repetibilidade de 0.2 mm.



a) Campo de ação coordenadas x e z. b) Campo de ação coordenada y.

Figura 23 - Campo de ação do Dobot Magician [24].

Na base do Dobot Magician, encontramos o Botão de Ligar/Desligar (*Power Button*) e uma indicação luminosa (*LED indicator*). Este indicador *Light-Emmiting Diode* (LED), representa vários estados de acordo com a cor apresentada, descritos na Tabela 10.

Indicador LED	Descrição
Verde	Dobot Magician em funcionamento normal.
Amarelo	Dobot Magician a iniciar funcionamento.
Azul	Dobot Magician em modo offline.
Azul intermitente	Dobot Magician a correr a posição *home* ou em processo de nivelamento automático.
Vermelho	Dobot Magician excedeu os limites de posição, alarme não foi limpo ou a ligação do kit de impressora 3D é anormal.

Tabela 10 - Descrição dos estados do indicador LED, presente na base do Dobot Magician [24].

Ainda na base do braço robótico, podemos encontrar as interfaces de comunicação disponíveis do Dobot Magician, situadas no painel traseiro, dispostas conforme a representação na Figura 24.

O ponto ① corresponde à chave de *reset*, *Reset Key* e reinicia o Dobot Magician. O ponto ② *Functional Key*, tem dois modos, um pressionar rápido, executa o programa em modo offline; e um pressionar de dois segundos, executa o procedimento de "*homing*" do Dobot Magician. A Inteface de comunicação ③, *Communication Interface/UART interface*, permite a comunicação

série, e a ligação de módulos acessórios do Dobot que disponibilizam comunicação Bluetooth e Wi-Fi. O protocolo de comandos do Dobot é adotado em qualquer uma destas ligações [25].

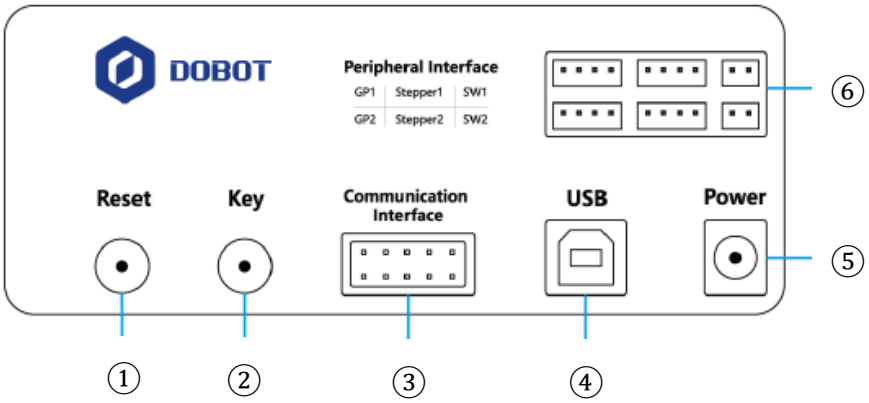


Figura 24 - Interfaces disponíveis na base do Dobot Magician [24].

A ligação USB ④, *USB interface*, permite a ligação ao PC. A alimentação ⑤, *Power interface* é a fonte de energia. A interface ⑥, permite a interligação de periféricos (*Peripheral interface)*, capacitando por exemplo o controlo pneumático da ventosa de sucção.

Na Tabela 11, estão representados os níveis de tensão da interface de comunicação da UART do Dobot Magician.

Pino	5V	RX	nRST	E2	5V
Level Output	5V/1A	-	-	3.3V/20mA	5V/1A
Level Input	-	3.3V/5V 20mA	-	-	-

Pino	GND	TX	Stop	E1	GND
Level Output	-	3.3V/5V 20mA	-	-	-
Level Input	-	-	3.3V/5V 20mA	3.3V/5V 20mA	-

Tabela 11 - Níveis de tensão da interface de comunicação UART com o Dobot Magician [26].

O Dobot Magician pode ser controlado através de Windows/Android/iOS, conseguindo a transmissão de dados através de qualquer uma das interfaces de comunicação disponíveis, através do seu software de controlo específico, o *DobotStudio* [27] ou através do protocolo de comando adotado em todas as suas interfaces. Neste software dedicado, é possível realizar o controlo de

posição e velocidade. A comunicação *USB to serial port,* tem como parâmetros, 115200 bps, 8 bits de dados, 1 *Stop bit* e sem paridade.

O protocolo de comando, baseia-se na construção de uma mensagem constituída por cabeçalho (*Header*), um pacote de carregamento (*Payload*), e uma soma de verificação, similar ao BCC, contabilizando apenas os dados do campo *payload* na soma (denominado nos manuais do Dobot Magician de *Checksum*). A constituição da mensagem, respeitando o protocolo de comando do Dobot Magician, está representada na Tabela 12.

Header		Len	Payload				Checksum
Header	Header		ID	Ctrl		Parameters	
				rw	isQueued		
0xAA	0xAA	len	id	rw	queue	parameters	payload checksum

Tabela 12 - Constituição do protocolo de comunicação do Dobot Magician.

Os dois primeiros parâmetros, são sempre o *Header* e representam o início da mensagem. O número de parâmetros do *Payload* é o Len (*Lenght*). Payload é o conjunto de instruções a serem enviadas ou recebidas, começando sempre pelo ID (*Identification Description*), seguido do Ctrl e depois os parâmetros da mensagem (*Parameters*).

O ID identifica o tipo de comando. Os comandos do Dobot Magician são divididos nas seguintes categorias de acordo com as suas funções de implementação: *Queue execution control command*, *Related command of device information*, *Common parameter command*, *Home function command*, *Handholdteaching command*, *Jog mode command*, *PTP mode command*, *CP mode command*, *TRACK mode command*, *WAIT mode command*, *TRIG trigger related command* e *IO control command*. Estas funções são classificadas pelos respetivos IDs, representados na Tabela 13.

Classificação	ID	Classificação	ID
ProtocolFunctionDeviceInfoBase	[0,10[	*ProtocolFunctionCPBase*	[90,100[
ProtocolFunctionPoseBase	[10,20[	*ProtocolFunctionARCBase*	[100,110[
ProtocolFunctionALARMBase	[20,30[	*ProtocolFunctionWAITBase*	[110,120[
ProtocolFunctionHOMEBase	[30,40[	*ProtocolFunctionTRIGBase*	[120,130[
ProtocolFunctionHHTBase	[40,50[	*ProtocolFunctionEIOBase*	[130,140[
ProtocolFunctionArmOrientationBase	[50,60[	*ProtocolFunctionCALBase*	[140,150[
ProtocolFunctionEndEffectorBase	[60,70[	*ProtocolFunctionWIFIBase*	[150,160[
ProtocolFunctionJOGBase	[70,80[	*ProtocolFunctionQueuedCmdBase*	[240,250[
ProtocolFunctionPTPBase	[80,90[	*ProtocolMax*	256

Tabela 13 - Classificação dos comandos por ID [25].

O campo *Ctrl* é composto por dois parâmetros, *rw* e *isQueue* e indica respetivamente se o comando está a escrever ou a ler informações, e/ou se está em fila. O parâmetro *rw* no valor lógico '1' representa comandos que escrevem informação e no valor lógico '0' corresponde a comandos que obtêm informação. O parâmetro *isQueued* com o valor lógico '1', indica que a instrução é um comado para a fila e devolve uma resposta de 64bits com a dimensão de 2+8, em que 2 representa o ID e Ctrl preenchidos, e 8 é o número de bits presentes no campo *parameters*. O parâmetro *isQueued* com o valor lógico '0' indica que a instrução é um comando imediato e não tem retorno tendo uma dimensão de 2+0, em que 0 significa que o campo *parameters* está vazio.

O campo de *Checksum,* representa o fim do comando e assegura o controlo de erros da comunicação. O *checksum* é calculado em dois passos, *i*) Soma do *payload*, produzindo o resultado R; e *ii*) Complemento para 2 do resultado R. A verificação da mensagem é realizada através da soma binária do *payload* com o *checksum,* e se este resultado for igual a 0, então a mensagem foi recebida corretamente.

Resumindo, o protocolo de comando do Dobot Magician, adotado para todas as suas interfaces de comunicação disponíveis, tem um tamanho variável, dependendo do comando pretendido. Todas as mensagens são enviadas inicialmente pelo utilizador e em todas essas comunicações, o Dobot Magician devolve uma resposta. Todos os comandos estão divididos em comandos imediatos e comandos de fila, em que todas as operações de leitura são comandos imediatos e podem ser executados imediatamente e os comandos de fila são colocados numa fila ordenada para execução em série. Para escrever ou definir um conjunto de operações, os comandos devem ser comandos de fila. Os comandos com parâmetros de movimento por exemplo, não são apenas comandos imediatos, mas também comandos de fila. Antes de enviar comandos de fila para o Dobot, o utilizador deve averiguar o espaço restante na fila de comandos.

3

Trabalho Desenvolvido

“ I have been impressed with the urgency of doing.

Knowing is not enough; we must apply.

Being willing is not enough; we must do. ”

Leonardo da Vinci

3. Trabalho desenvolvido

Neste capítulo, são apresentados os requisitos para a elaboração e desenvolvimento de todo o projeto bem como a solução implementada. É analisado o controlador escolhido e os braços robóticos para validar o funcionamento do projeto. São explicadas as comunicações implementadas para a interface remota genérica e os comandos implementados para controlo dos braços robóticos.

3.1 Arquitetura proposta

O projeto a desenvolver, tem por base a implementação e desenvolvimento de uma interface remota, que para além da comunicação série, permite estabelecer comunicações Bluetooth e Wi-Fi. A interface é genérica, de forma a adaptar-se ao controlo de diferentes braços robóticos. A arquitetura proposta para desenvolvimento do projeto é composta por três módulos, o braço robótico a controlar (C), o processamento de comandos para o controlo (B) e a interface utilizada pelo utilizador (A), como representado na Figura 25.

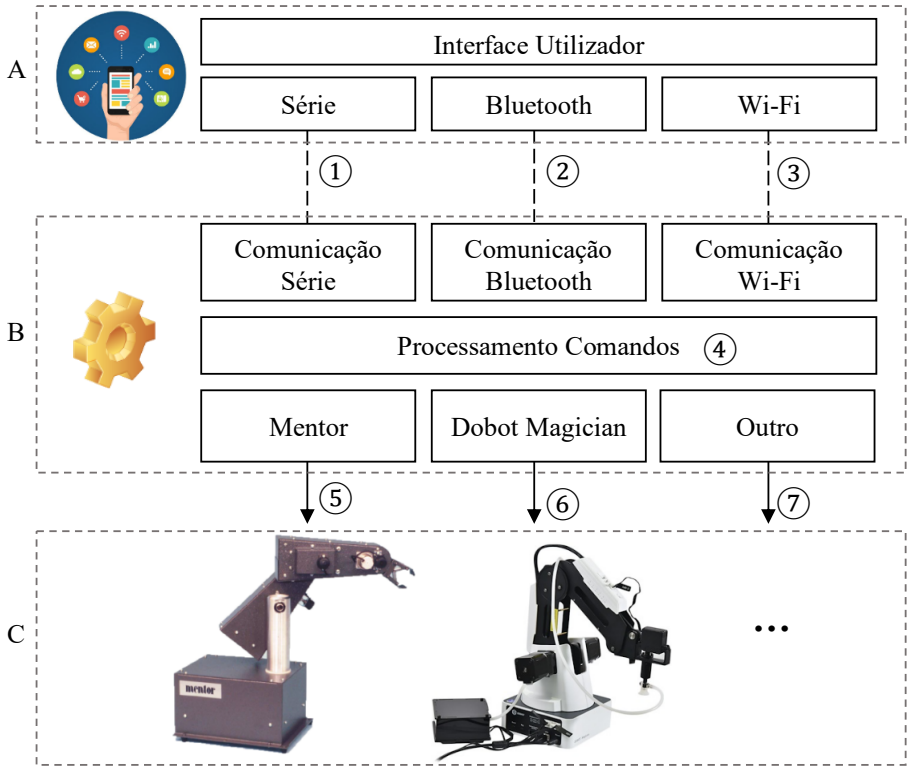


Figura 25 - Arquitetura para desenvolvimento do projeto.

A Figura 25 (C), representa os braços robóticos adotados para a validação de todo o projeto. Os braços robóticos adotados são o Mentor e o Dobot Magician, analisados anteriormente na Secção 2.5.

Em (B), estão representados todos os processos que o controlador escolhido tem de realizar. O controlador, tem de ter acesso às comunicações série ①, Bluetooth ② e Wi-Fi ③. Tem de ter disponíveis, múltiplas possibilidades de ligações físicas a braços robóticos e capacidade de programação e processamento de comandos para o controlo dos mesmos. Os pontos ⑤⑥⑦, representam as ligações físicas que o controlador terá de satisfazer de forma a poder ligar-se aos braços robóticos e o processamento de comandos ④, destinado a cada robô.

Em (A), está representada a interface remota genérica traduzida na interface utilizador-máquina para controlar um braço robótico, como o Mentor ou o Dobot Magician, com três interfaces de comunicação distintas, série, Bluetooth e Wi-Fi.

A escolha do controlador para elaborar o projeto passa pela decisão entre um autómato programável ou um microcontrolador. Entre estes dois controladores a escolha rapidamente recaiu sobre um microcontrolador, pois o seu preço é significantemente mais reduzido e mais fácil de transportar.

Existem muitos microcontroladores, por isso o que cumpre os requisitos elaborados para desenvolvimento do projeto com o menor preço é a escolha adotada. O microcontrolador selecionado foi um ESP32 disponível na plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather, que tem disponíveis os vários meios de comunicação estabelecidos como requisitos para o trabalho.

3.2 Solução implementada: Microcontrolador

A implementação da Interface Remota Genérica para controlo de Braços Robóticos (IRGBR), tem por base o microcontrolador ESP-WROOM-32 [28], na plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather [29]. Na Figura 26, estão representadas, uma vista frontal e uma vista traseira da plataforma de desenvolvimento, sem os pinos soldados.

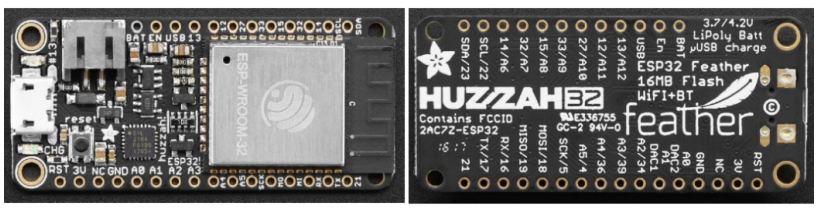


Figura 26 - Vista frontal (esquerda) e traseira (direita) da plataforma de desenvolvimento.

Alguns dos componentes desta plataforma estão assinalados na Figura 27. Em ①, está representado o LED *built in, um LED* associado ao pino 13. A plataforma conta ainda com um segundo LED, o *charging status* LED ③, que fica ligado, sempre que a placa é alimentada. Tem um botão de *reset* ④, bastante útil para reiniciar execuções do código carregado no microcontrolador ESP-32 ⑤. O microcontrolador, pode ser alimentado na ligação a um computador via USB/serial *port* ou a uma bateria. Na ligação cabo Micro USB ②, a placa regulará o USB de 5V para 3,3V. Esta plataforma de desenvolvimento, via micro USB, permite a interligação com o software Arduino IDE.

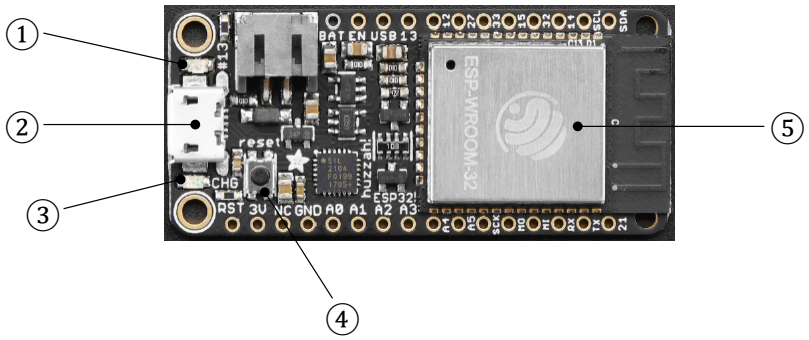


Figura 27 - Detalhe de alguns periféricos na plataforma de desenvolvimento.

O Arduino IDE [30] é uma plataforma de desenvolvimento de código aberto multiplataforma, disponível para os sistemas operativos Windows, Macintosh OSX e Linux. A linguagem de programação utilizada foi a linguagem C [31].

Na Tabela 14, estão representadas as especificações principais do microcontrolador ESP32.

Nome	Adafruit Huzzah32 - ESP32 Feather
Preço	21,95$, aproximadamente 18,13€
Memória dados	520 KB SRAM
Memória código	4 MByte flash
Cristal interno	8 MHz
Pinos	18 GPIO, 4 GPI, 11 AI, 2 AO
Interfaces	SPI, I^2C, UART, USB, Bluetooth, Wi-Fi
Dimensões	51 x 22.7 x 7.3 mm
Peso	6.8g

Tabela 14 - Especificações do microcontrolador ESP32.

O microcontrolador disponibiliza bastantes interfaces de comunicação e inclui pinos de ligação. Podem-se organizar os pinos do microcontrolador em duas categorias, pinos de alimentação e pinos lógicos. Os pinos lógicos podem ainda ser divididos em quatro categorias, sendo elas, *serial*

pins, *Inter-Integrated Circuit e Serial Peripheral Interface pins* (I^2C e SPI), *Global Purpose Input Output pins* (*GPIO)* e *Analog Input pins* (AI).

A plataforma de desenvolvimento disponibiliza cinco pinos de alimentação, GND, BAT, USB, EN e 3V. Os pinos lógicos são os pinos de entrada/saída de uso geral, definidos para o microcontrolador e toda a lógica funciona a 3,3V. Os pinos lógicos serial, RX e TX, são os pinos *Serial1*. Estes pinos são usados para conexão a dispositivos, transmissores recetores assíncronos universais (UART).

Os pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather podem ser consultados com mais detalhe no Anexo 1.

3.2.1 Ligação do microcontrolador ao Mentor

Para a ligação ao Mentor, são necessárias 11 entradas digitais e 8 bidirecionais como representado na Figura 28 e visto anteriormente na Secção 2.5.1.

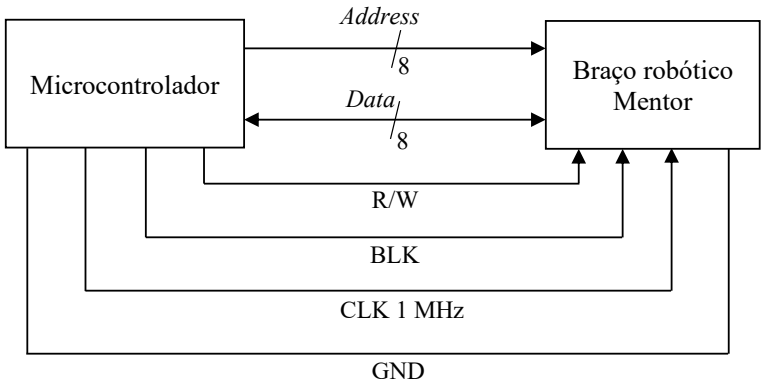


Figura 28 - Ligação para comunicação com o Mentor.

A comunicação entre o microcontrolador e o braço robótico Mentor caracteriza-se como sendo uma transmissão de informação digital através de um canal de comunicação paralelo e é síncrona porque baseia-se num sinal de *clock* de 1MHz. Nesta comunicação o microcontrolador é o *master* e o Mentor é o *slave*.

O microcontrolador adotado só tem 18 pinos bidirecionais de uso geral, ou seja o microcontrolador precisaria de 11 saídas digitais e 8 bidirecionais que totalizam 19 saídas. De forma a conseguir-se realizar a ligação foi utilizado um *shift register*, que permite expandir o número de saídas digitais até oito saídas, à custa de três saídas digitais do microcontrolador. Na Figura 29, está representado o diagrama de blocos do *shift register* SN74HC595 usado.

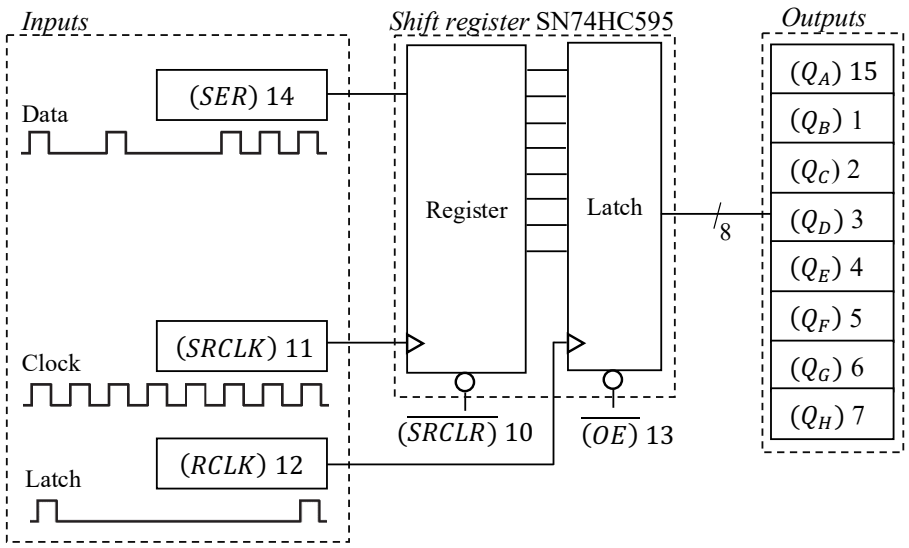


Figura 29 - Shift register SN74HC595.

Este *shift register* através de 3 *inputs*, SER, SRCLK e RCLK, recebe os 8 bits, cada bit numa saída binária, de (Q_A) a (Q_H). O pino *Shift Enable Serial Input Register* (SER), recebe a sequência de bits que se pretende transmitir. O pino S*hift Register Clock* (SRCLK), é o sinal de *clock* do *shift register*. O terceiro pino marca o início e o fim da sequência de 8 bits que são postos nas oito saídas binárias, o R*egister Clock* (RCLK). De modo a ter-se em permanência as oito saídas binárias da mensagem de 8 bits, liga-se o *Output Enable* ($\overline{OE}$) ao valor lógico '0', neste caso ao *ground* (GND). O *Shift Register Clear* ($\overline{SRCLR}$), coloca-se inativo, ou seja, no valor lógico '1', ligando neste caso no pino 3V do µC.

A ligação adotada entre o microcontrolador e o Mentor está representada na Figura 30.

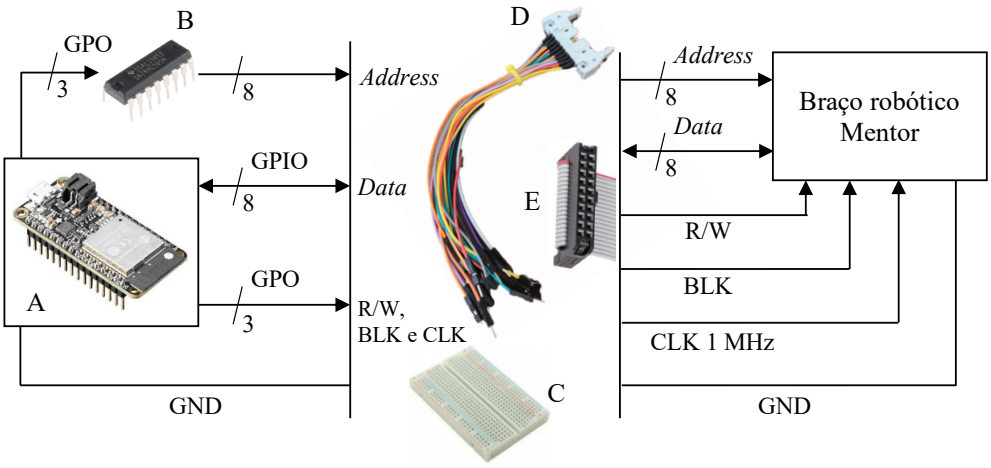


Figura 30 - Ligação entre o microcontrolador e o Mentor.

Com a utilização do *Shift Register* SN SN74HC595 [32], representado na Figura 30 em (B), o microcontrolador (A) usa 6 pinos de saída digital (3GPO + 3GPO), 8 bidirecionais e 1 para o *ground* para realizar toda a comunicação com o Mentor. O microcontrolador e o *shift register* são ligados numa *breadboard* (C) em conjunto com o adaptador (D) para ligação ao *flat cable* (E).

De forma a implementar-se o sinal de *clock* de 1 MHz recorre-se a um sinal *Pulse With Modulation* (PWM), no microcontrolador, no pino 21. A parametrização do PWM é realizada através da função *ledcAttachpin*, associando o pino 21 ao canal 1 com *i*) ledcAttachpin(21, 1); em seguida, com a instrução *ii*) ledcSetup(1, 1000000, 6), para parametrizar o canal 1 com a frequência de 1 000 000 Hz, com uma resolução de 6 bits. Considerando uma frequência máxima disponível no microcontrolador de 80 MHz e a resolução de 6 bits, obtém-se uma frequência máxima disponível para uso de, $80\,000\,000/2^6 = 1250000\,Hz$. Sendo assim os 1 000 000Hz requisitados ao canal 1 são possíveis, pois estão dentro da gama de funcionamento. Finalmente com a instrução *iii*) ledcWrite(1, 32); define-se, um *duty cicle* de 50% ($2^6 \times 0{,}5 = 32$) para o canal 1.

As ligações realizadas ao adaptador de *Flat Cable* (FC) para o Mentor, estão representadas na Tabela 15, onde as cores apresentadas, correspondem às cores associadas nas ligações dos fios usados no adaptador para ligação ao *flat cable*.

Ligação	Adaptador Flat Cable								Ligação
(GPIO 13) µC 13		Data 0	FC-01	FC-02	Address 0				(Q_A) SR 15
(GPIO 27) µC 27		Data 1	FC-03	FC-04	Address 1				(Q_B) SR 1
(GPIO 33) µC 33		Data 2	FC-05	FC-06	Address 2				(Q_C) SR 2
(GPIO 15) µC 15		Data 3	FC-07	FC-08	Address 3				(Q_D) SR 3
(GPIO 32) µC 32		Data 4	FC-09	FC-10	Address 4				(Q_E) SR 4
(GPIO 14) µC 14		Data 5	FC-11	FC-12	BLK				(GPIO) 19 µC MI
(GPIO 22) µC SCL		Data 6	FC-13	FC-14	Address 5				(Q_F) SR 5
(GPIO 23) µC SDA		Data 7	FC-15	FC-16	Address 6				(Q_G) SR 6
(GPIO 18) µC MO		R/W	FC-17	FC-18	Address 7				(Q_H) SR 7
µC GND		GND	FC-19	FC-20	CLK				(GPIO 21) µC 21

Tabela 15 - Constituição e ligações do adaptador de Flat-Cable (Mentor).

A consulta dos pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather pode ser consultada no Anexo 1. O esquema de ligações detalhado com os pinos e com as ligações realizadas do microcontrolador ao Mentor, podem ser consultadas no Anexo 2.

3.2.2 Ligação microcontrolador Dobot Magician

A comunicação entre o microcontrolador e o braço robótico Dobot Magician caracteriza-se como sendo uma transmissão de informação digital através de um canal série, assíncrona, em modo *half-duplex*, uma vez que apenas o microcontrolador representa o papel de *master* e o Dobot Magician o de *slave*.

Para a ligação ao Dobot Magician, é utilizada a interface UART. Nesta ligação são necessários apenas 3 pinos do microcontrolador: Um de transmissão TX; um de receção RX; e o sinal de referência *ground*. As ligações entre o microcontrolador e a interface de ligação UART do Dobot Magician estão descritas na Tabela 16.

Ligação	Microcontrolador
Dobot UART GND	GND
Dobot UART TX	RX
Dobot UART RX	TX

Tabela 16 - Ligações do microcontrolador (Dobot Magician).

Note-se que o pino transmissor do microcontrolador TX, liga na interface do Dobot em RX, para receber a transmissão e o pino receptor do microcontrolador RX, liga na interface do Dobot em TX, para receber a resposta do Dobot Magician. Os pinos de ligação Tx e Rx do microcontrolador, correspondem a *Serial1*.

A consulta dos pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather pode ser consultada no Anexo 1. O esquema de ligações detalhado com os pinos e com as ligações realizadas do microcontrolador ao Dobot Magician, podem ser consultadas no Anexo 2.

3.3 Implementação de comandos

A interface remota genérica desenvolvida permite o controlo de diversos braços robóticos. Cada robô tem as suas particularidades e os comandos são ajustados a cada braço robótico dentro das suas funcionalidades.

3.3.1 Comandos implementados para o Mentor

O Mentor é um braço robótico em que a programação funciona por entrada direta de dados. Nesse sentido, os comandos mais básicos de movimentação para o Mentor funcionam escolhendo o valor das suas juntas dentro dos limites pré-estabelecidos.

Os comandos implementados para o Mentor são **W**, **S**, **E**, **EV**, **WR** que representam a movimentação de cada uma das suas cinco juntas respetivamente a J1, J2, J3, J4, J5 e **G** que representa o controlo da garra. Ambos vistos anteriormente na Secção 2.5.1 na Tabela 6.

O comando **W xxx**, permite a movimentação da junta J1, *waist*, balizado entre 0 a 250, correspondendo a uma movimentação de torção da cintura.

O comando **S xxx**, permite a movimentação da junta J2, *shoulder*, balizado entre 0 a 250, correspondendo a uma movimentação de rotação do ombro.

O comando **E xxx**, permite a movimentação da junta J3, *elbow*, balizado entre 0 a 250, que correspondendo a uma movimentação de rotação do cotovelo.

O comando **EV xxx**, permite a movimentação da junta J4, *elevation wrist*, balizado entre 77 a 178, correspondendo a uma elevação do punho.

O comando **WR xxx**, permite a movimentação da junta J5, *rotation wrist*, balizado entre 0 a 255, correspondendo a uma torção do punho.

O comando **G xxx**, permite abrir e fechar a garra, sendo balizado entre 151 e 196, em que sendo G 151 é o estado inicial da garra, aberta e G 196 representa a garra fechada.

Foram ainda implementados os comandos **H**, **R** e **IP**.

O comando **H**, realiza o posicionamento do Mentor na *home position*, uma posição que corresponde a W 128, S 128, E 128, EV 128, WR 128 e G 151. Basicamente a *home postion* é a posição inicial do Mentor que no estado inicial começa com a garra aberta e com todas as suas cinco juntas com um valor central.

O comando **R**, permite realizar a leitura dos valores nos eixos W, S, E, EV, WR e do valor G da garra do Mentor.

O comando **IP**, foi concebido para receber um endereço IP, para acesso ao servidor Web desenvolvido, para controlo do Mentor.

As instruções para comando do braço robótico Mentor, bem como as respostas da interface na utilização dos vários comandos implementados, encontram-se resumidos, para as três comunicações desenvolvidas, no Anexo 3.

Os resultados do controlo do braço robótico Mentor com o código desenvolvido, são ilustrados no capítulo 4.1. No Anexo 5, pode ser consultado o código desenvolvido para a Interface Remota Genérica para controlo do Mentor.

3.3.2 Comandos implementados para o Dobot Magician

O Dobot Magician possui um protocolo de comandos [25], descrito na Secção 2.6. O controlo implementado para este braço robótico, foi realizado através da definição de coordenadas no plano xyz e o elemento terminal aqui considerado é a ventosa a vácuo que tem um movimento de torção representado por R como representado na Figura 31.

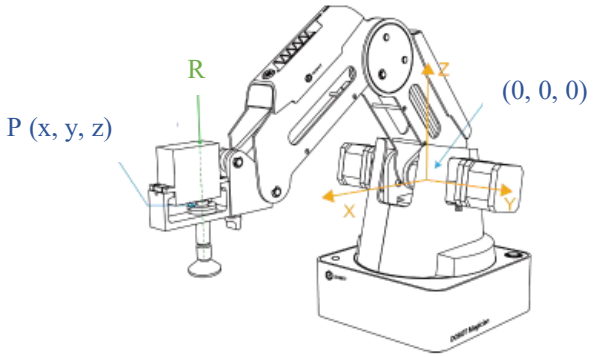


Figura 31 - Coordenadas do Dobot Magician no plano xyz.

Os comandos implementados para o Dobot Magician são **P**, **H**, **R**, **A** e **IP**.

O comando **P x y z r**, realiza a movimentação do Dobot inserindo as coordenadas x, y, z e r em que o valor x, y e z corresponde às coordenadas do elemento terminal no plano xyz, e r corresponde ao valor de torção do elemento terminal, em que os quatro valores são decimais. Os limites das coordenadas possíveis para os eixos foram definidos para cada eixo através de verificação experimental. Para o eixo x o Dobot está preparado para uma movimentação de -90º a 90º completando um movimento angular máximo de 180º, no entanto de forma a não realizar movimentações excessivas, balizaram-se os valores de 160mm a 250mm para o valor da coordenada x. Para o eixo y o Dobot está preparado para uma movimentação de 0º a 85º completando um movimento angular máximo de 85º, no entanto de forma a não realizar movimentações excessivas balizaram-se os valores de -150mm a 150mm para o valor da coordenada y. Para o eixo z o Dobot está preparado para uma movimentação de -10º a 90º completando um movimento angular máximo de 100º, no entanto de forma a não realizar movimentações excessivas, balizaram-se os valores de -50mm a 75mm para o valor da coordenada z. Para a torção do elemento terminal o Dobot consegue realizar uma movimentação

angular de -90º a 90º, que representa um movimento de torção máxima de 180º, assim adotaram-se os valores no intervalo de [-50º; +50º] que correspondem a uma torção de -50º a 50º.

Para implementar o comando P x y z r, é necessário implementar o protocolo do Dobot Magician *ProtocolFunctionPTPBase > SetPTPCmd* [25]. Na Tabela 17, está representada a constituição da trama para construção do comando P.

	Header		Len	Payload			Checksum
	Header 1	Header 2		ID	Ctrl	Parameters	
Comando	170	170	2+17	84	3 ou 1	parameters	checksum
Resposta	170	170	2+8 ou 2+0	84	3 ou 1	parameters	checksum

Tabela 17 - Constituição da trama do comando P para o Dobot Magician.

No comando de movimentação para a posição desejada, o utilizador introduz as coordenadas que deseja como um *float*. O protocolo do Dobot Magician utiliza o modo *little-endian* na representação do *float,* enquanto que no microcontrolador o modo utilizado é *big-endian*. Em *little-endian* os bytes são guardados por ordem crescente do seu peso numérico em endereços sucessivos da memória (extremidade menor primeiro ou *little-endian*), enquanto que em *big-endian*, os bytes são guardados por ordem decrescente do seu peso numérico em endereços sucessivos da memória (extremidade maior primeiro ou *big-endian*).

O processamento de comandos para envio e receção das coordenadas, tem em consideração a conversão entre os modos. Assim, para enviar de um comando de posição P para o Dobot Magician, é necessário realizar o seguinte processamento:

① Introduzir as coordenadas em *float* (b*ig endian*);

② Converter as coordenadas *float* para hexadecimal, em seguida realizar um *Swap Endianess,* obtendo-se a coordenada hexadecimal em modo *little endian*;

③ Converter a coordenada hexadecimal para decimal.

Na Tabela 18, está representado um exemplo do processamento anteriormente descrito para envio de um comando P com os valores x:250.0, y:150.0, z:75.0 e r:50.0.

	x				y				z				r				
Float	250.0				150.0				75.0				50.0				①
	x1	x2	x3	x4	y1	y2	y3	y4	z1	z2	z3	z4	r1	r2	r3	r4	
Hex	00	00	7A	43	00	00	16	43	00	00	96	42	00	00	48	42	②
Dec	0	0	122	67	0	0	22	67	0	0	150	66	0	0	72	66	③

Tabela 18 - Conversão numérica no envio de um comando P 250.0 150.0 75.0 50.0 para o Dobot.

Na Tabela 19 e na Tabela 20, está representada a trama enviada no envio do comando P 250.0 150.0 75.0 50.0, como exemplo. As coordenadas (x, y, z, r) em *float* são convertidas para hexadecimal e depois convertidas para decimal, como analisado na Tabela 18. O *checksum* é calculado automaticamente em cada comando enviado.

	Header			Payload			
	Header	Header	Len	ID	Ctrl	Parameters	Checksum
Dec	170	170	19	84	3	parameters	checksum

Tabela 19 - Exemplo da trama enviada no envio do comando P 250.0 150.0 75.0 50.0.

	parameters															
	Mode	x				y				z				r			
		x1	x2	x3	x4	y1	y2	y3	y4	z1	z2	z3	z4	r1	r2	r3	r4
Dec	1	0	0	122	67	0	0	22	67	0	0	150	66	0	0	72	66

Tabela 20 - Pormenor do campo *parameters* no envio do comando P 250.0 150.0 75.0 50.0.

Para o envio do comando P - Posição Desejada, são realizadas as seguintes ações:

i) Introdução das coordenadas para movimentação do Dobot em *Float* ①, no envio do comando P, pelo utilizador;

ii) As coordenadas introduzidas para movimentação do Dobot em *Float* ①, são convertidas para Hex, *swap endianness* do número Hex ② e por último são convertidas para Dec ③, como visto na Tabela 18;

iii) Cálculo do checksum;

iv) Envio da trama completa para o Dobot Magician;

v) Recepção da resposta.

O comando **R**, permite realizar a leitura das coordenadas onde o Dobot se encontra posicionado, seguindo o protocolo Dobot Magician *ProtocolFunctionPoseBase* > *GetPose* [25]. Após o envio deste comando, a resposta, retorna as coordenadas x y z e r do Dobot. A constituição do comando R e a respetiva resposta estão representadas na Tabela 21.

	Header			Payload			
	Header 1	Header 2	Len	ID	Ctrl	Parameters	Checksum
Comando	170	170	2+0	10	0	empty	246
Resposta	170	170	2+32	10	0	parameters	checksum

Tabela 21 - Constituição da trama do comando R para o Dobot Magician.

Segue-se uma análise das tramas enviadas e recebidas na utilização do comando R, depois de ter sido enviado o comando P 250.0 150.0 75.0 50.0, descrito anteriormente.

O comando R, tem como função realizar a leitura das coordenadas do Dobot Magician no plano xyz e a torção da ventosa a vácuo. A trama enviada na utilização do comando R está representada na Tabela 22.

	Header		Len	Payload			Checksum
	Header 1	Header 2		ID	Ctrl	Parameters	
Dec	170	170	2	10	0	empty	246

Tabela 22 - Exemplo da trama enviada no envio do comando R para o Dobot Magician.

Na Tabela 23 e na Tabela 24, tem-se o exemplo da resposta recebida, proveniente do envio do comando R.

	Header		Len	Payload			Checksum
	Header 1	Header 2		ID	Ctrl	Parameters	
Dec	170	170	34	10	0	parameters	checksum

Tabela 23 - Trama recebida na utilização do comando R.

	parameters															
	x				y				z				r			
	x1	x2	x3	x4	y1	y2	y3	y4	z1	z2	z3	z4	r1	r2	r3	r4
Dec	0	0	122	67	0	0	22	67	0	0	150	66	0	0	72	66

Tabela 24 - Pormenor do campo *parameters* da trama recebida na utilização do comando R.

Na receção de tramas com coordenadas de posição respondidas pelo Dobot Magician acontece o seguinte processamento:

① Receção da trama em decimal (*Little Endian*);

② Conversão da trama para hexadecimal, *Swap Endianess* para converter as coordenadas em hexadecimal no modo *Big Endian*;

③ Conversão das coordenadas em hexadecimal para *float*.

Na Tabela 25, está representada esta lógica de conversão, na receção da trama proveniente do envio de um comando R.

	x1	x2	x3	x4	y1	y2	y3	y4	z1	z2	z3	z4	r1	r2	r3	r4	
Dec	0	0	122	67	0	0	22	67	0	0	150	66	0	0	72	66	①
Hex	00	00	7A	43	00	00	16	43	0	0	96	42	00	00	48	42	②
	x				y				z				r				
Float	250.0				150.0				75.0				50.0				③

Tabela 25 - Conversão numérica na receção da trama de resposta do comando R.

Para o envio do comando R - Ler coordenadas, são realizadas as seguintes ações:

i) Envio da trama completa associada ao comando R para o Dobot Magician;

ii) Receção e processamento da resposta, como representado na Tabela 25.

O comando **H**, realiza o posicionamento do Dobot Magician para a sua *home position.* Este comando calibra e reinicia o Dobot, eliminando todos os erros e alarmes, usando o protocolo Dobot Magician *ProtocolFunctionHOMEBase* > *SetHOMECmd* [25]. A *home position* do Dobot Magician, equivale às seguintes coordenadas: x:260,5; y:0; z:-8,5; e r:0.

	Header		Len	Payload			Checksum
	Header 1	Header 2		ID	Ctrl	Parameters	
Comando	170	170	2+4	31	3 ou 1	parameters	222
Resposta	170	170	2+8 ou 2+0	31	3 ou 1	parameters	checksum

Tabela 26 - Constituição da trama do comando H para o Dobot Magician.

Para o envio do comando H - *home position*, são realizadas as seguintes ações:

i) Envio da trama completa associada ao comando H para o Dobot Magician;

ii) Receção da resposta.

O comando **A**, permite receber o código de alarme do Dobot Magician, recorrendo ao protocolo Dobot Magician *ProtocolFunctionALARMBase* > *GetAlarmsState* [25].

	Header		Len	Payload			Checksum
	Header 1	Header 2		ID	Ctrl	Parameters	
Comando	170	170	2+0	20	0	empty	236
Resposta	170	170	2+16	20	0	parameters	checksum

Tabela 27 - Constituição da trama do comando A para o Dobot Magician.

O protocolo de comando do Dobot Magician apresenta algumas incongruências, nomeadamente ao desenvolver-se uma interface de comando própria, a falta de informação disponibilizada por parte do fabricante sobre os códigos que representam os vários alarmes relativos ao Dobot foram um impedimento de atribuir o motivo de alarme a cada código que recebia. A solução adotada foi a de que se o Dobot apresentar alarme, então recebe-se o código de alarme, apresentando-o ao utilizador, sendo eliminado através do comando de calibração, *home position* (comando H), sendo um dos métodos sugeridos no manual [25].

Para o envio do comando A - Receção do código de alarme, são realizadas as seguintes ações:

i) Envio da trama associada ao comando A;

ii) Receção e processamento da resposta.

Para além destes comandos foi ainda implementado mais um comando com a designação, **IP**, comando que permite receber o endereço IP, do servidor Web, desenvolvido para controlo do Dobot Magician.

O cálculo do *checksum*, por exemplo, no envio do comando P 250.0 150.0 75.0 50.0, pode-se realizar em quatro passos:

$i)\ Soma\ Payload_{Decimal}\colon\ ID + Ctrl + Parameters$

$$84 + 3 + 1 + 0 + 0 + 122 + 67 + 0 + 0 + 22 + 67 + 0 + 0 + 150 + 66 + 0 + 0 + 72 + 66 = 720$$

$ii)\ Soma\ Payload_{Binário}\colon\ 720_{Decimal} = 0010\ 1101\ 0000_{Binário}$

$iii)\ Soma\ Payload_{Binário\ 8LSB}\colon$

$$0010\ 1101\ 0000_{Binário} = 1101\ 0000_{Binário\ 8LSB} = 208_{Decimal}$$

$iv)\ Checksum = 2'scomplement_{Decimal}\colon$

$$256 - 208 = 48$$

Conclui-se assim, que no envio do comando P 250.0 150.0 75.0 50.0, obtém-se um *checksum* de 48 em decimal.

As instruções para comando do braço robótico Dobot Magician, bem como as respostas da interface na utilização dos vários comandos implementados encontram-se resumidos para as três comunicações desenvolvidas, no Anexo 4.

Os resultados do controlo do braço robótico Dobot Magician com o código desenvolvido, são ilustrados no Capítulo 4.2. No Anexo 5 pode ser consultado o código desenvolvido para a Interface Remota Genérica para controlo do Dobot Magician.

3.4 Interface remota genérica

A interface para controlo de braços robóticos, foi desenvolvida em linguagem C. Além de ambiente de desenvolvimento, o software Arduino permite o carregamento do código desenvolvido para o microcontrolador. O microcontrolador está ligado fisicamente ao braço robótico a controlar, como visto nas Secções 3.2.1 e 3.2.2.

A Interface Remota Genérica para controlo de Braços Robóticos (IRGBR), foi desenvolvida, a pensar na adaptação de vários braços robóticos, assim sendo, o código principal, está preparado para receber sempre uma nova camada correspondente ao braço robótico que se pretenda controlar, não tendo de constituir um código total de raiz. O *commandhandler* é o mecanismo usado na adaptação e criação de comandos para os braços robóticos controlados pela interface remota genérica para robôs analisado nas Secções 3.4.1 e 3.4.2.

A interface remota genérica desenvolvida tem módulos para os três tipos de comunicações consideradas para o controlo dos braços robóticos, analisadas separadamente, nas Secções 3.4.3, 3.4.4 e 3.4.5.

3.4.1 CommandHandler Mentor

Para o braço robótico Mentor, o *commandhandler* considera dois formatos. Esses formatos são classificados por conterem ou não argumento. Assim, para o controlo do Mentor existem três comandos sem argumento e seis comandos com argumento, representados na Tabela 28.

CommandHandler

Comando	Argumento	
H	Sem argumento	
IP		
R		
W xxx	Com 1 argumento	0 a 250
S xxx		0 a 250
E xxx		0 a 250
EV xxx		77 a 178
WR xxx		0 a 255
G xxx		151 a 196

Tabela 28 - CommandHandler comandos Mentor.

Um dos comandos sem argumento é o comando H, responsável pelo Mentor adotar a posição inicial, que basicamente é colocar o valor 128 em todas as suas juntas e colocar a garra no valor 151, ou seja, garra aberta. Quando o utilizador envia o comando H, recebe uma resposta com a posição inicial, "Robô Mentor - Posição Inicial.\nW:128 S:128 E:128 EV:128 WR:128 G:151".

O comando IP, outro comando sem argumento, devolve ao utilizador o endereço IP do sistema implementado, permitindo assim obter o endereço do servidor Web, para controlar por Wi-Fi o Mentor, via *browser*.

O comando R, solicita a leitura dos eixos do Mentor e como resposta devolve ao utilizador, "Robô Mentor - Leitura de Eixos.\nW:xxx S:xxx E:xxx EV:xxx WR:xxx G:xxx ", em que W xxx representa o valor *waist* presente na junta J1, S xxx representa o valor *shoulder* presente na junta J2, E xxx representa o valor *elbow* na junta J3, EV xxx representa o valor *elevation wrist* da junta J4, WR xxx representa o valor *rotation wrist* da junta J5 e G xxx representa o valor associado à abertura ou fecho da garra terminal do Mentor.

Para os comandos com argumento tem-se cada movimentação individual das 5 juntas do Mentor mais a garra. Tem-se o comando W, com argumento disponível de 0 a 250 para movimentação da cintura do Mentor. O comando S, com argumento de 0 a 250 para movimentação do ombro e o comando E, com argumento disponível de 0 a 250 para movimentação do cotovelo. O comando EV, para movimentação do pulso num movimento de elevação, com argumento de 77 a 178 e o comando WR, para movimentação do pulso, mas neste comando para um movimento de rotação, com argumento de 0 a 255. E por fim, o comando G com argumento de 151 que representa a garra aberta até 196, que representa a garra fechada.

Caso seja enviado um comando inválido o sistema retorna uma mensagem de erro, informando o utilizador através da mensagem, "Comando Inválido! Insira um novo comando."

Nos comandos com argumento para movimentação do Mentor, W, S, E, WR, EV e G, caso a gama de valores exceda os valores permitidos, a interface informa o utilizador que o argumento é inválido através da mensagem "Comando X: xxx, Argumento Inválido! Repita.". Ainda nestes comandos de movimentação, se o utilizador apenas enviar o comando sem o argumento a interface devolve a mensagem de aviso "Comando X sem argumento, insira um novo comando.".

3.4.2 CommandHandler Dobot Magician

Para o braço robótico Dobot Magician, o *commandhandler* considera dois formatos. Esses formatos são separados por conterem quatro argumentos ou nenhum argumento e são representados na Tabela 29.

CommandHandler

Comando		Argumento		
H IP R A		Sem argumento		
P x y z r	Com 4 argumentos	Argumento 1	x	160 a 250
		Argumento 2	y	-150 a 150
		Argumento 3	z	-50 a 75
		Argumento 4	r	-50 a 50

Tabela 29 - CommandHandler comandos Dobot Magician.

Para o Dobot existem quatro comandos sem argumento, o comando H, que faz o Dobot retornar a posição inicial, as coordenadas, x:260.5, y:0 z:-8.5 e r:0. Neste caso o utilizador usa o comando H e recebe "Robô Dobot - Posição Inicial.\n X:260.5 Y:0 Z:-8.5 R:0". O comando IP também não tem argumento e serve para devolver ao utilizador o endereço IP para acesso ao servidor Web que permite controlar por Wi-Fi o Dobot. Outro comando sem argumento, o comando R, faz a leitura das coordenadas do Dobot e devolve ao utilizador, "Robô Dobot - Leitura Coordenadas.\n X:xxx Y:xxx Z:xxx R:xxx". O último comando sem argumento, permite ao utilizador ultrapassar o estado de alarme que o Dobot apresenta, realizando a calibração no retorno à posição *home* eliminando o alarme existente, recebendo o código de alarme. Neste comando o utilizador recebe o código de alarme, que se existir, será diferente de 0 e o Dobot Magician retorna à posição inicial.

Para os comandos com argumento tem-se a movimentação do braço robótico Dobot com o comando P seguido das suas quatro coordenadas de movimentação. O primeiro argumento diz respeito ao eixo x e está balizado entre 160mm a 250mm, que representa a coordenada x no plano xyz. O segundo argumento diz respeito ao eixo y e está balizado entre -150mm a 150mm que representa a coordenada y no plano xyz. O terceiro argumento diz respeito ao eixo z e está balizado ente -50mm a 75mm que representa a coordenada z no plano xyz. Por fim, o quarto argumento diz respeito à coordenada r que representa a torção do acessório terminal, a ventosa a vácuo que está balizado entre -50º a 50º.

A letra representa a função, e nos argumentos são passados os valores das coordenadas. Se for enviado um comando diferente dos anteriormente descritos, é um comando inválido, retornando uma mensagem de "Comando Inválido! Insira um novo comando."

No comando com 4 argumentos para movimentação do Dobot, P x y z e, caso se falhe a gama de valores permitida a interface avisa o utilizador qual ou quais os argumentos que são inválidos recebendo a mensagem "Robô Dobot (deteção argumentos inválidos) Sem 4 argumentos válidos,

repita o comando.”. Ainda neste comando, se por acaso o utilizador apenas enviar a letra, sem argumento, a interface devolve a mensagem de aviso, “Comando P sem argumentos! Repita.”.

3.4.3 Comunicação Série

A comunicação série, resulta no envio e receção de comandos a partir de um monitor série. Para validação do projeto recorre-se ao monitor série disponível no Arduino IDE.

Na comunicação série para controlo de um braço robótico, a partir de um monitor série, o utilizador tem de escrever e enviar os comandos de controlo disponíveis, com ou sem argumento e recebe as respostas aos comandos realizados.

Na Figura 32, estão representadas resumidamente, as etapas da comunicação série, em que o utilizador envia os comandos via monitor série e recebe as respostas no monitor série e pode também realizar uma monitorização do que é feito na comunicação série, num monitor Bluetooth se estiver ativa a comunicação Bluetooth (A). O microcontrolador realiza o processamento de todos os comandos recebidos (B) e retorna ao utilizador as respostas dos braços robóticos aos comandos realizados. O utilizador apenas tem o controlo de um braço robótico (C), em cada execução de código.

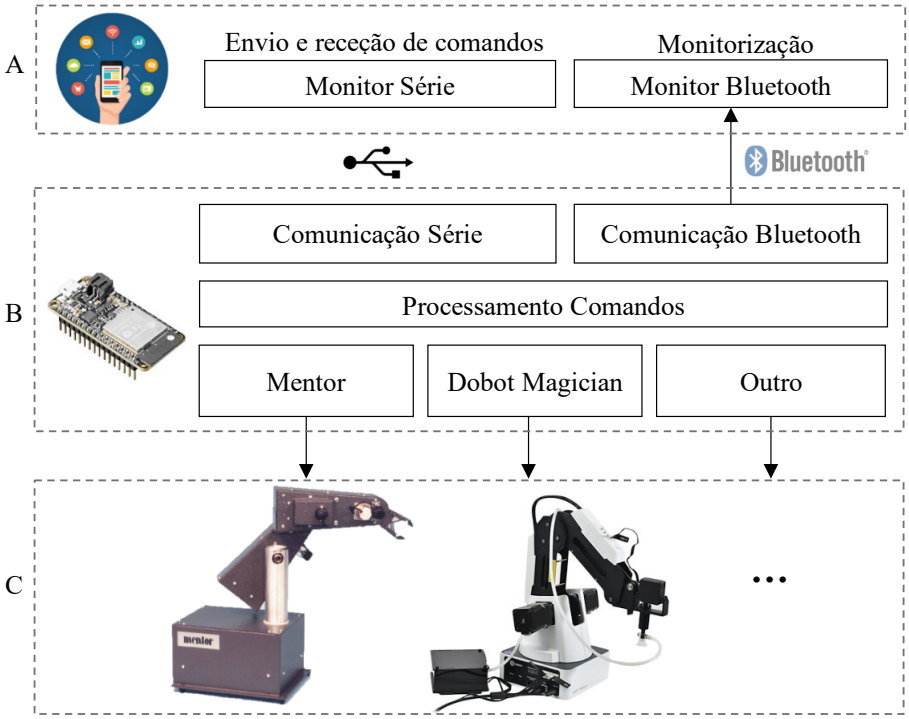


Figura 32 - Comunicação Série da interface remota genérica.

3.4.4 Comunicação Bluetooth

A comunicação Bluetooth, é assegurada por qualquer dispositivo que tenha Bluetooth. O funcionamento é similar ao da comunicação série, pois funciona através de um monitor Bluetooth implementando um canal série. Assim, o utilizador escreve e envia os comandos de controlo disponíveis, com ou sem argumento e recebe as respetivas respostas. Tal como no caso da comunicação série as ações e repostas do controlo podem ser monitorizadas num monitor série. Na Figura 33, estão representadas resumidamente as etapas da comunicação Bluetooth, em que o utilizador envia os comandos via monitor Bluetooth e recebe as respostas no monitor Bluetooth, e pode também realizar a monitorização do sistema num monitor série se estiver ativa a comunicação série (A). O microcontrolador realiza o processamento de todos os comandos recebidos (B) e retorna ao utilizador as respostas dos braços robóticos aos comandos solicitados. O utilizador apenas tem o controlo de um braço robótico (C) em cada execução do código nesta comunicação.

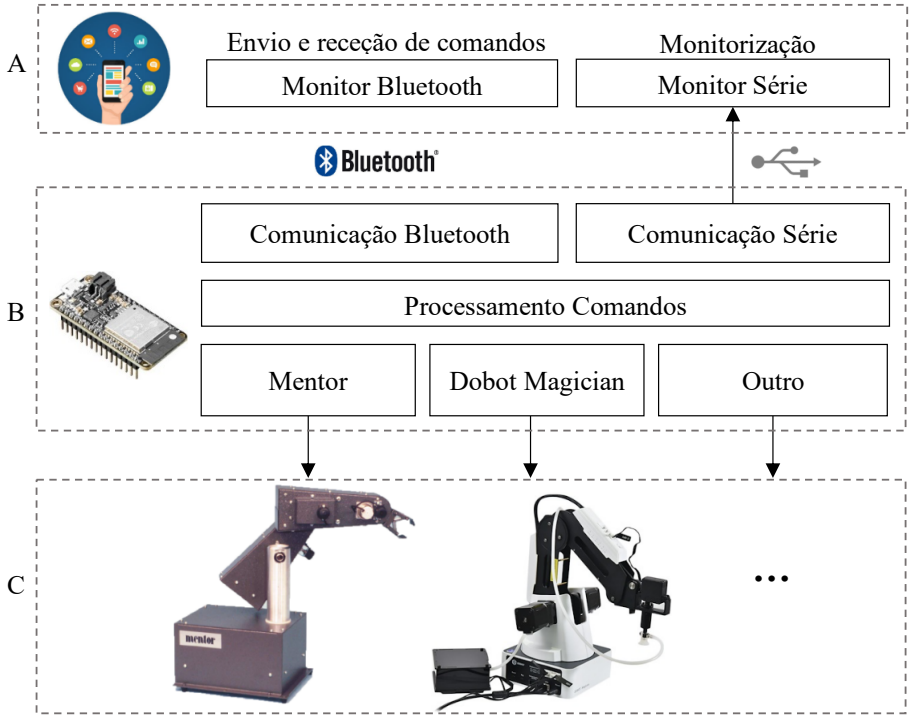


Figura 33 - Comunicação Bluetooth da interface remota genérica.

O monitor Bluetooth utilizado no projeto, para a comunicação Bluetooth, foi a aplicação Serial Bluetooth Terminal [33], disponível na *playstore* para Android e representado na Figura 34.

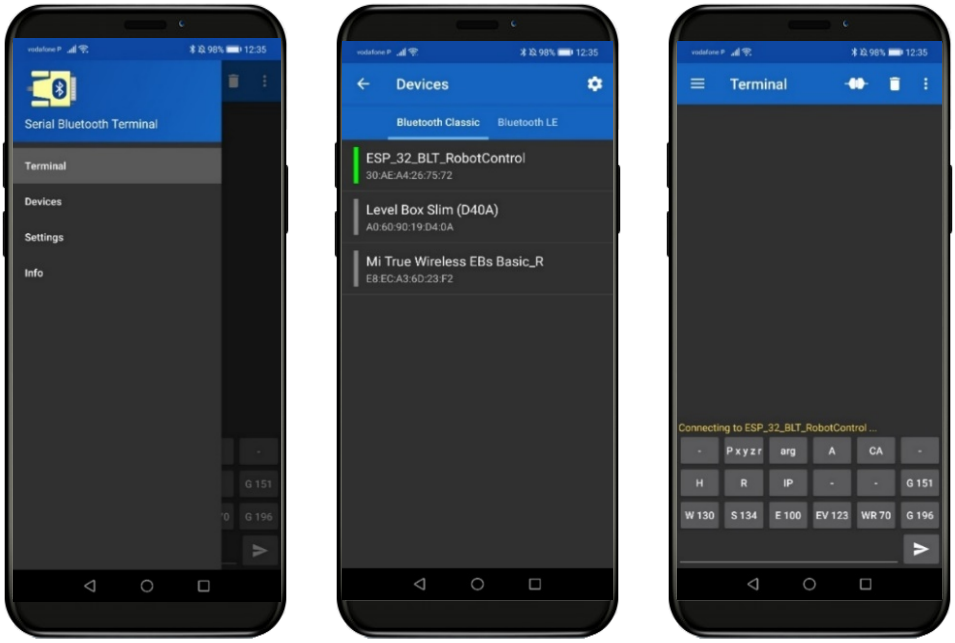


Figura 34 - Procedimento para emparelhamento Bluetooth na aplicação Serial Bluetooth Terminal.

A aplicação Serial Bluetooth Terminal, é uma aplicação de terminal/consola específica para microcontroladores, arduinos e outros dispositivos com interface série/UART, ligados por Bluetooth, que realiza uma função de conversor série para dispositivos android.

Para se usar o monitor Bluetooth da aplicação Serial Bluetooth Terminal, basta ligar o Bluetooth do dispositivo e emparelhar com o microcontrolador. Acedendo à aplicação basta realizar, Terminal → Devices → ESP_32_BLT_RobotControl, como ilustrado na Figura 34.

3.4.5 Comunicação Wi-Fi

A comunicação Wi-Fi é realizada através de um servidor Web. Este servidor Web é acedido através do endereço IP devolvido pelo microcontrolador. O servidor Web resulta de um conjunto de páginas construídas e desenvolvidas em *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CCS) e JavaScript, otimizadas para vários tamanhos de ecrã, adaptando-se aos diversos dispositivos, habitualmente usados como interface, desde *tablets*, *smartphones* e *laptops*. O servidor Web desenvolvido tem por base a arquitetura de páginas representada na Figura 35.

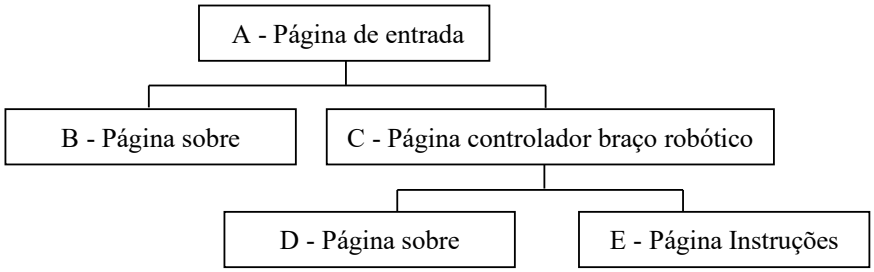


Figura 35 - Arquitetura do servidor Web desenvolvido.

A página de entrada (A) é acedida através do endereço IP. Uma vez na página de entrada, é solicitado ao utilizador que realize o *login* com as credenciais de entrada corretas, nome de utilizador e palavra-passe, para aceder à página do controlador do braço robótico em uso. Na página de entrada o utilizador pode também aceder a uma página explicativa do sistema (B).

Com o *login* corretamente efetuado o utilizador é redirecionado para a página do controlador do braço robótico (C). A página de controlo desenvolvida tem a mesma disposição gráfica (*layout*) independente dos braços robóticos adotados, conforme representado na Figura 36, com o objetivo de ser genérica de modo a adaptar-se aos braços robóticos considerados no âmbito deste projeto.

Na Figura 36, está representada a disposição gráfica (*layout*), criada para os controlos dos robôs adotados ou a serem adicionados.

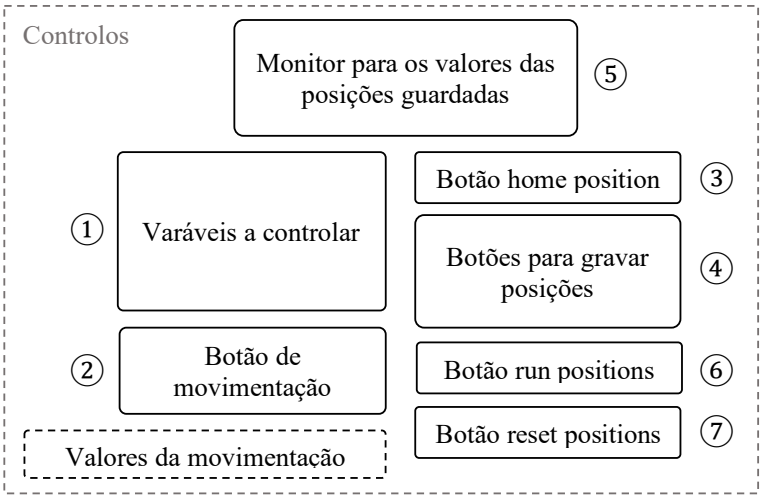


Figura 36 - *Layout* dos controlos dos braços robóticos no servidor Web.

Em ①, têm-se as variáveis a controlar. Em ②, depois de escolhidos os valores de ①, com o Botão de movimentação "Enviar," o braço robótico executa a movimentação pretendida. O botão ③, faz o braço robótico executar a sua *home position*, uma característica comum a todos os manipuladores. O conjunto ④, contém os botões para gravar várias posições de movimentação,

que podem ser observadas no monitor ⑤. O botão *run positions* ⑥, faz o braço robótico executar as posições gravadas no sistema controlador. O botão *reset positions* ⑦, elimina as posições anteriormente armazenadas em ⑤.

Este *layout* através de vários elementos interativos permite realizar o controlo remoto do braço robótico adotado. Na página de controlo do braço robótico (C), o utilizador tem ainda acesso a uma página que contém as instruções de utilização (E) e a uma página explicativa (D).

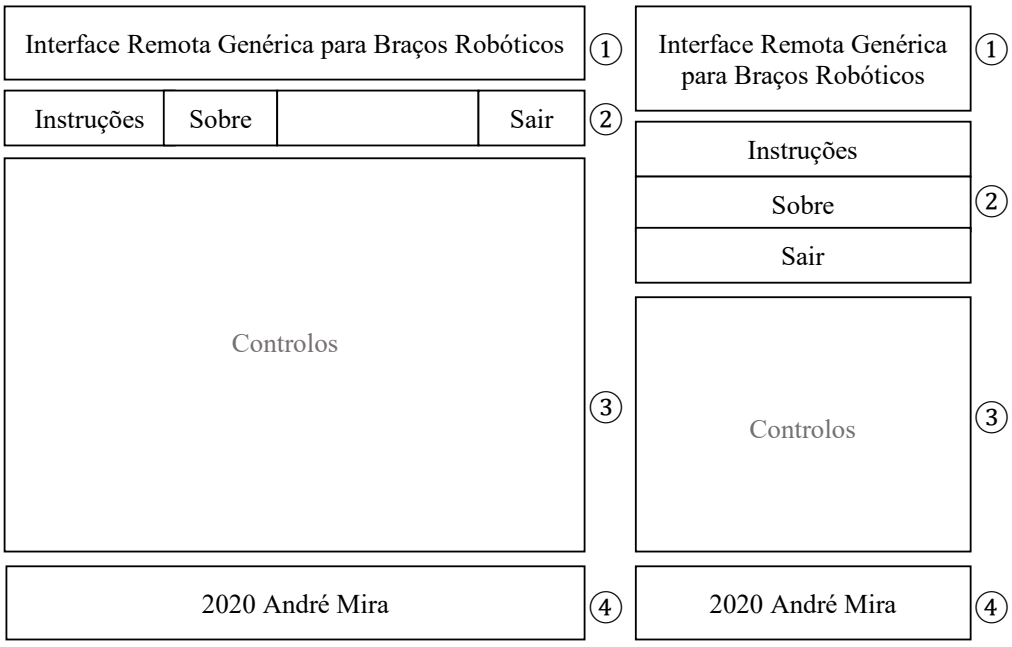
Na Figura 37, está representada a página de controlo do braço robótico com o *layout* comum a todas as páginas do servidor Web, constituídas por:

① Cabeçalho;

② Barra de navegação superior;

③ Corpo da página;

④ Rodapé.



a) *Landscape*. b) *Portrait*.

Figura 37 - Layout adaptativo do servidor Web desenvolvido.

A Figura 37 a), representa o layout da página de controlo dos braços robóticos, no modo panorama (*landscape*) e em b) está representado o layout em modo retrato (*portrait*).

As páginas constituintes do servidor Web, apenas têm a diferença, dos botões presentes na barra de navegação superior.

Toda a construção do servidor foi realizada em HTML, contando com botões de submissão de formulário. No servidor Web implementou-se também um controlo de acessos para limitar o acesso ao controlo dos robôs somente a utilizadores com as credenciais válidas. Com recurso a CCS e a Java Script implementaram-se as funções de estilo para o cabeçalho, para a barra de navegação superior, para o corpo da página e para o rodapé, em que todos são adaptativos a todas as dimensões de ecrãs e por isso o *layout* é adaptado a qualquer equipamento eletrónico.

Os acessos na barra de navegação têm dois estados, o de repouso e o de *click* e todos os botões desenvolvidos têm três estados, o de repouso, o de ser carregado, e o de passar em cima com o rato. No rodapé foram implementados dois botões para acesso aos contactos do autor do projeto. A página HTML do controlador do braço robótico recorre ainda ao uso de *cookies* para fixar o valor das variáveis.

Na Figura 38, estão representadas resumidamente as etapas da comunicação Wi-Fi, em que o utilizador envia os comandos através da página de controlo acedida através do servidor Web e recebe as respostas dos controlos num monitor série e/ou num monitor Bluetooth (A).

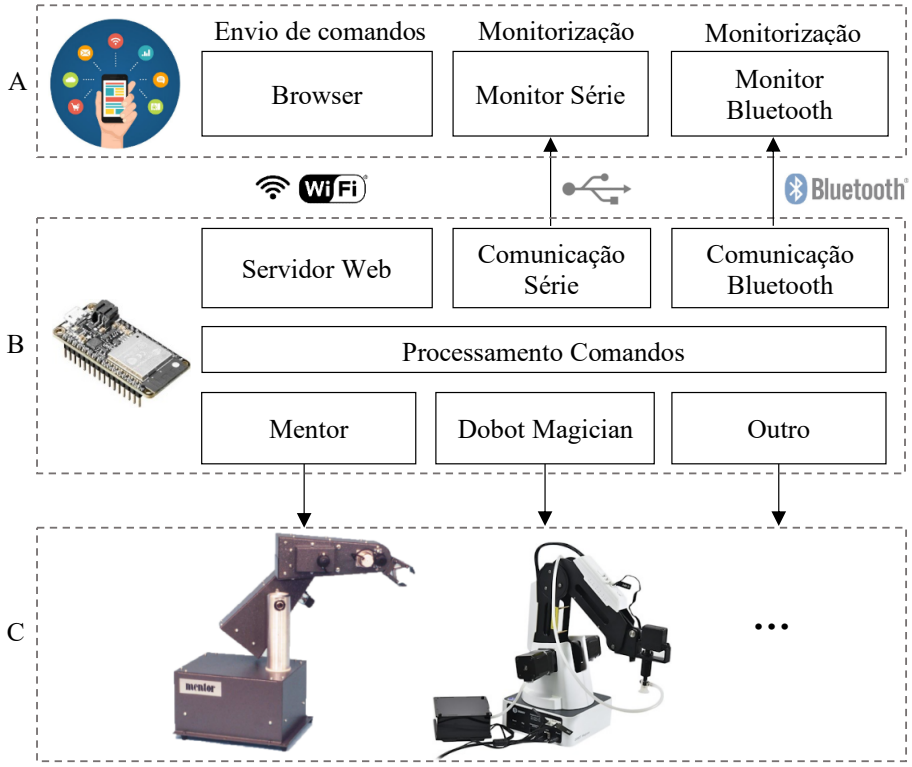


Figura 38 - Comunicação Wi-Fi da interface remota genérica.

O microcontrolador realiza o processamento de todos os comandos recebidos (B) e retorna ao utilizador as respostas dos braços robóticos aos comandos realizados através dos monitores série e/ou Bluetooth se ligados. O utilizador apenas tem o controlo de um braço robótico (C) em cada execução do código. A comunicação Wi-Fi, é então assegurada por qualquer dispositivo que tenha acesso ao protocolo de comunicação Wi-Fi. O utilizador, através do servidor Web, tem acesso a uma página de controlo interativo para o braço robótico. Este controlo realizado via Wi-Fi, pode ainda, se ligados, ser monitorizado pelos monitores série e /ou Bluetooth registando as ações e repostas do controlo.

4

Resultados

“ I have not failed,

but found 1000 ways to not make a light bulb. ”

Thomas Edison

4. Resultados

Neste capítulo, é apresentado o funcionamento da Interface Remota Genérica para Braços Robóticos (IRGBR) desenvolvida, analisando o controlo dos braços robóticos, Mentor e Dobot Magician, com as comunicações Série, Bluetooth e Wi-Fi.

Na Figura 39, está representado o funcionamento da IRGBR. O utilizador tem disponíveis os monitores Série e Bluetooth para envio e receção de comandos e através do acesso a um servidor Web tem uma página para controlo do braço robótico, controlo este que pode ser monitorizado a partir de um ou de ambos os monitores, Série e Bluetooth, como representado em (A).

O microcontrolador realiza o processamento de todos os comandos recebidos (B) e retorna ao utilizador as respostas dos braços robóticos aos comandos usados. O utilizador apenas tem o controlo de um braço robótico (C) em cada execução do código desenvolvido.

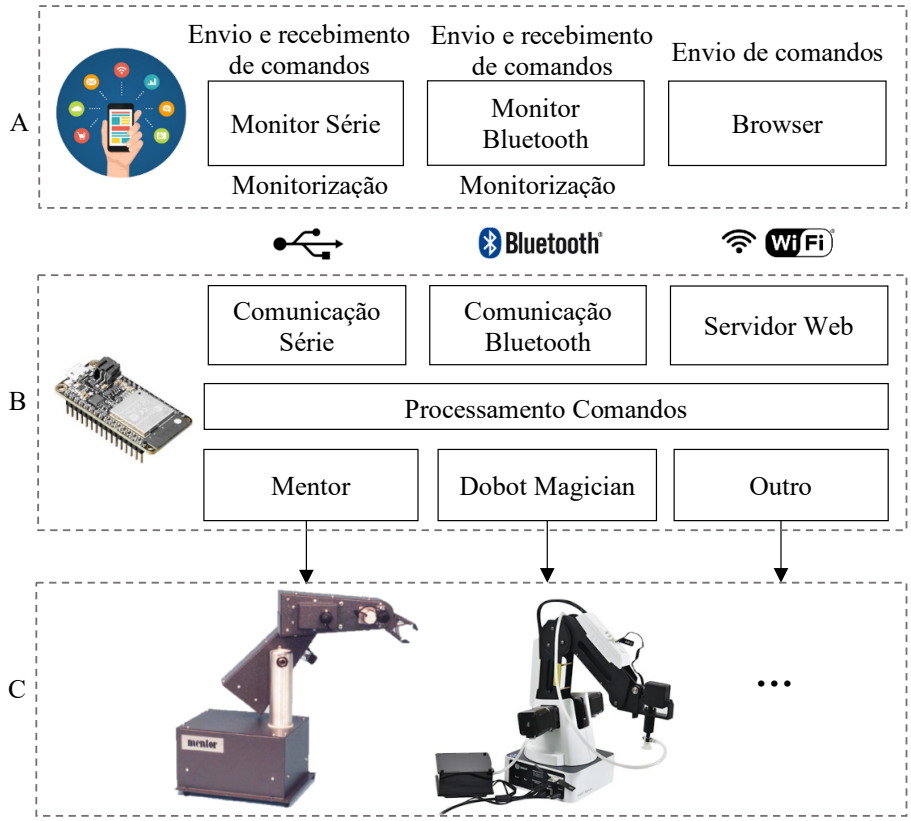


Figura 39 - Utilização da IRGBR com as três comunicações em simultâneo.

Na Secção 4.1 e 4.2 é analisado o controlo particular, do Mentor e do Dobot Magician, pela IRGBR desenvolvida.

4.1 Interface remota genérica: controlo do Mentor

A IRGBR para o Mentor funciona alimentando o microcontrolador a um portátil ou a uma *powerbank* e ligando-o ao braço robótico. Depois de descarregado o código para o microcontrolador a interface está pronta a utilizar. O utilizador tem disponível três comunicações para realizar o controlo do Mentor, comunicação série, comunicação Bluetooth e comunicação Wi-Fi.

A comunicação série, acontece ligando o microcontrolador ao software Arduino e acedendo ao monitor série. A partir deste, pode-se enviar e receber comandos para o Mentor.

Na Figura 40, está representada a janela do Monitor série do Arduino, com o exemplo de execução de vários comandos enviados na interface, para controlo do Mentor, via comunicação série.

No controlo do Mentor, o monitor série, tem de se configurar para operar a 9600 baud e com avanço automático de linha. Neste monitor, o envio de comandos funciona escrevendo o comando pretendido e carregando no botão enviar no topo da janela.

A execução do programa começa com o premir do botão *reset* no microcontrolador. No monitor série o utilizador recebe o cabeçalho da comunicação série como representado na Figura 40, em (A). O cabeçalho da interface na comunicação série, delimita o uso do botão *reset* entre várias execuções do código. Assim que o código começa, a primeira ação é movimentar o Mentor para a sua *home position*. A seguir a IRGBR devolve ao utilizador o endereço IP para acesso ao servidor Web, assim que completar a ligação Wi-Fi. Ainda no cabeçalho o utilizador tem uma mensagem com os comandos disponíveis, “Comunicação Série (Mentor) \n Comandos disponíveis: W, S, E, EV, WR, G, R e IP”.

Em (B), está representado o envio de vários comandos pelo utilizador, para comunicação série com o Mentor. No exemplo de execução representado na Figura 40, o primeiro comando enviado foi W 250, comando que faz o Mentor realizar uma torção do seu eixo *waist,* e no qual o utilizador obtém como resposta, “Movimentação Waist, W:250.” como confirmação no monitor série.

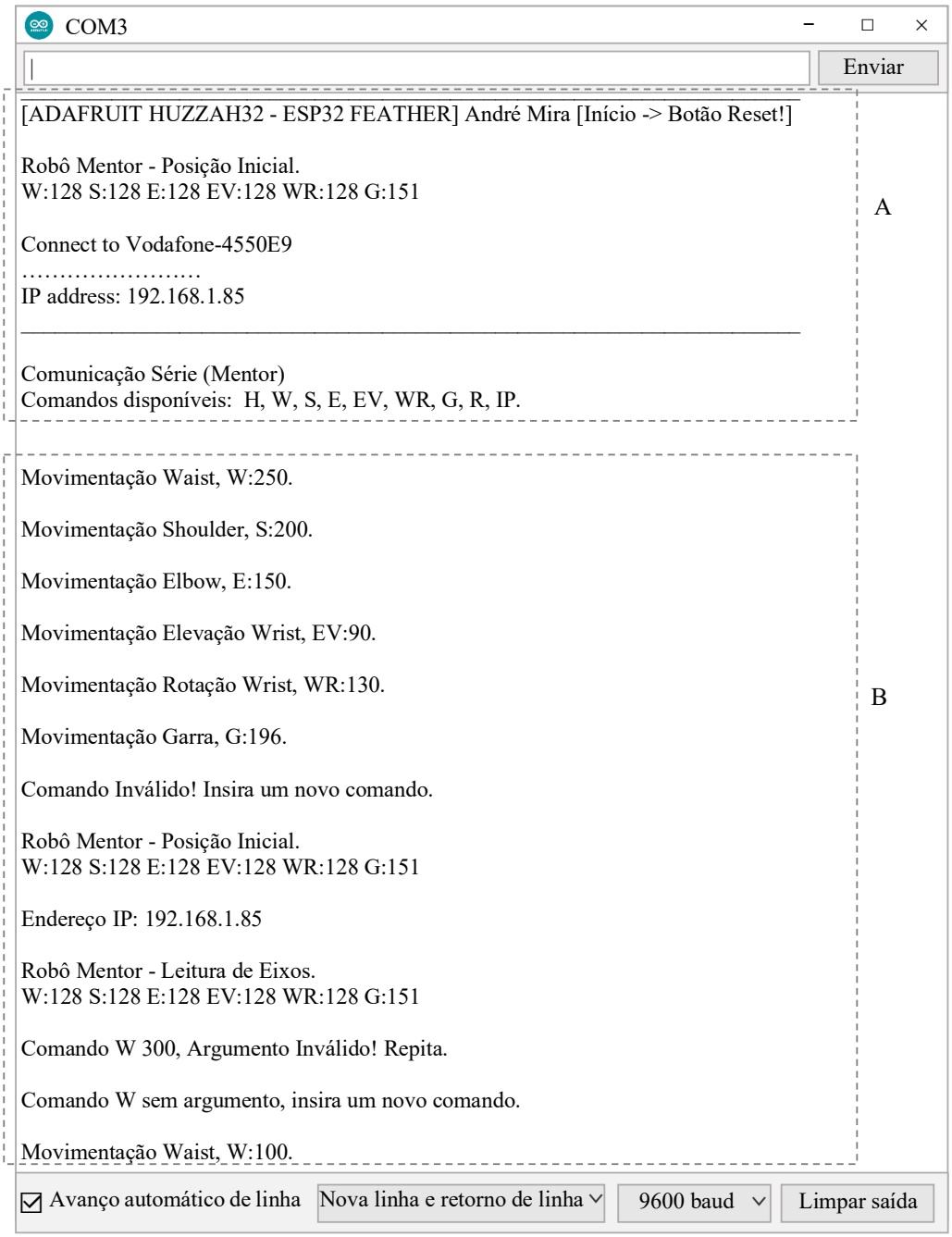


Figura 40 - Exemplo de comunicação série para controlo do Mentor (envio via Monitor Série Arduino).

Os comandos W, S, E, EV, WR e G são semelhantes, no sentido em que ambos depois de enviados, resultam em mensagens de confirmação do comando efetuado para o utilizador no monitor série. No exemplo em análise, seguiu-se o envio dos comandos, S 200, para uma movimentação de rotação do eixo *shoulder*, E 150, para uma movimentação de rotação do eixo

elbow, EV 90, para uma movimentação de elevação do eixo *wrist elevation*, WR 130, para uma movimentação de torção do eixo *wrist rotation* e G 196, para fechar a garra.

Depois do envio dos comandos básicos de movimentação do Mentor W, S, E, EV, WR e G, o utilizador enviou o comando "A", no qual obteve como resposta, "Comando Inválido! Insira um novo comando.". Este exemplo serve para demonstrar a resposta na introdução de um comando inválido, em que todos os comandos introduzidos, diferentes de W, S, E, EV, WR, G, H, R e IP, resultam num comando inválido, solicitando ao utilizador que introduza um novo comando válido.

No envio do comando H, comando que permite posicionar o Mentor na sua *home position*, o utilizador obtém como resposta "Robô Mentor - Posição Inicial. \n W:128 S:128 E:128 EV:128 WR:128 G:151.". Neste caso, a posição *home* do Mentor, são todos os seus eixos com o valor médio 128 e a garra aberta (G 151).

A qualquer altura pode-se executar o comando IP, para receber o endereço IP de acesso ao servidor Web, analisado mais à frente, no qual se obtém como resposta "Endereço IP: 192.168.1.85".

No envio do comando R, comando que permite ler os valores nos eixos do Mentor, o utilizador obtém como resposta "Robô Mentor - Leitura de Eixos. \n W:128 S:128 E:128 EV:128 WR:128 G:151."

No envio do comando W 300, o utilizador obtém como resposta, "Comando W 300, Argumento Inválido! Repita", pois trata-se do envio de um comando válido, mas com um argumento inválido pois não se encontra na gama de valores disponíveis para o comando. Todos os comandos válidos enviados, mas com argumento inválido serão sinalizados como no exemplo.

No envio do comando W, o utilizador obtém como resposta "Comando W sem argumento, insira um novo comando.", pois trata-se do envio de um comando válido, mas sem argumento. Todos os comandos válidos, mas sem argumento são também sinalizados pela interface. Em ambos estes casos, de comando válido com argumento inválido ou sem argumento, o Mentor não realiza qualquer movimentação.

Os comandos enviados a partir do monitor série do Arduino podem também ser monitorizados a partir de um monitor Bluetooth, como por exemplo o da aplicação *Bluetooth Serial Terminal*. A monitorização dos comandos enviados por comunicação série, a partir do monitor série do Arduino, vistos anteriormente, está representada na Figura 41, recorrendo ao monitor Bluetooth da aplicação *Bluetooth Serial Terminal*.

Para ativar esta monitorização, o utilizador apenas tem de se emparelhar via Bluetooth com o dispositivo ESP_32_BLT_RobotControl. Uma vez concluído o emparelhamento, o utilizador obtém a resposta "*Conected*" a amarelo, que assinala o emparelhamento concluído com sucesso.

Na aplicação, as mensagens a amarelo representam o emparelhamento, o início e o fim da conexão. As mensagens a verde representam todas as respostas recebidas.

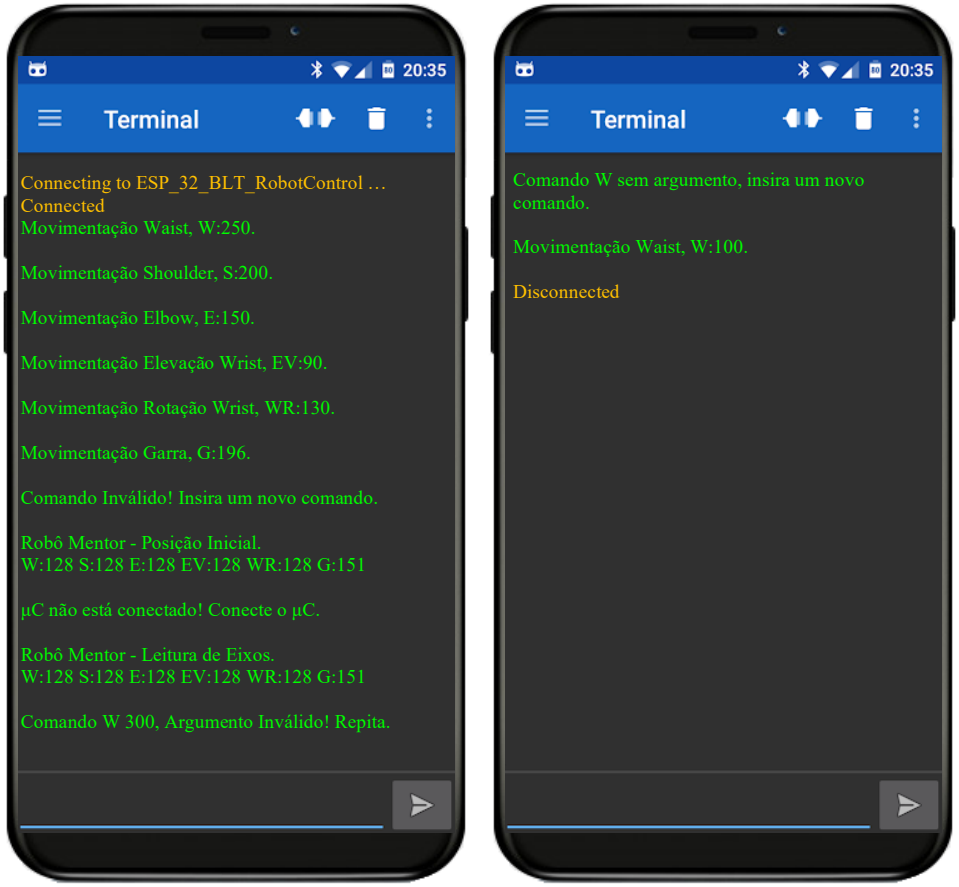


Figura 41 - Exemplo de monitorização da comunicação Série para controlo do Mentor (via monitor Bluetooth).

Na Figura 41, estão representadas as respostas recebidas, no monitor Bluetooth da aplicação *Bluetooth Serial Terminal*, provenientes do envio dos comandos vistos anteriormente, via monitor Série.

A comunicação Bluetooth funciona também a partir da aplicação *Bluetooth Serial Terminal* que disponibiliza um monitor Bluetooth, semelhante ao monitor série. Os comandos que o utilizador desejar são enviados via Bluetooth para o microcontrolador e podem também ser monitorizados num monitor série se ativo.

Na Figura 42, está o exemplo de uma execução de envio de comandos via Bluetooth para controlo do Mentor. Neste exemplo é enviada uma sequência de comandos semelhante à analisada anteriormente na comunicação série.

Os comandos introduzidos e enviados pelo utilizador, aparecem a azul e a resposta a esse comando enviado encontra-se a verde. Neste teste foram executados os comandos W 130, para movimentar o eixo *Waist*, S 134, para movimentar o eixo *shoulder*, E 100, para movimentar o eixo *elbow*, EV 123, para movimentar o eixo *wrist elevation*, WR 70, para movimentar o eixo *wrist rotation*, G 151, para manter a garra aberta. Seguiu-se o envio do comando A, para perceber a resposta à introdução de um comando inválido. O comando H, para posicionar o Mentor na posição *home* e o comando IP, para receber o endereço IP que permite o acesso ao servidor Web.

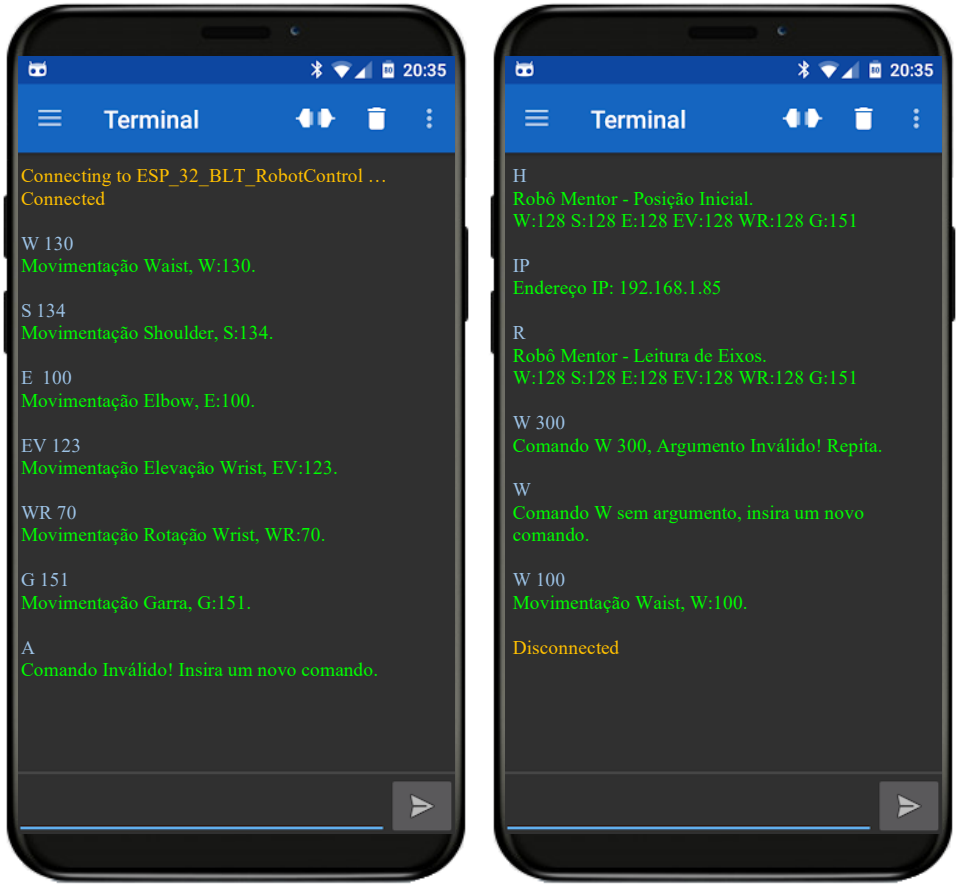


Figura 42 - Exemplo de comunicação Bluetooth para controlo do Mentor (envio via Monitor Bluetooth).

O envio do comando R, permite ler os eixos do Mentor. O comando W 300, permite demonstrar a resposta na introdução de um comando válido, mas com argumento inválido e W para demonstrar a resposta de um comando válido sem argumento.

Na realização da comunicação série, o utilizador envia e recebe respostas no monitor série e pode monitorizar o que está a realizar no monitor Bluetooth. Na realização da comunicação Bluetooth, o utilizador à semelhança com a comunicação série, pode monitorizar o envio e receção de comandos não só no monitor Bluetooth, mas também no monitor série, como representado na Figura 43.

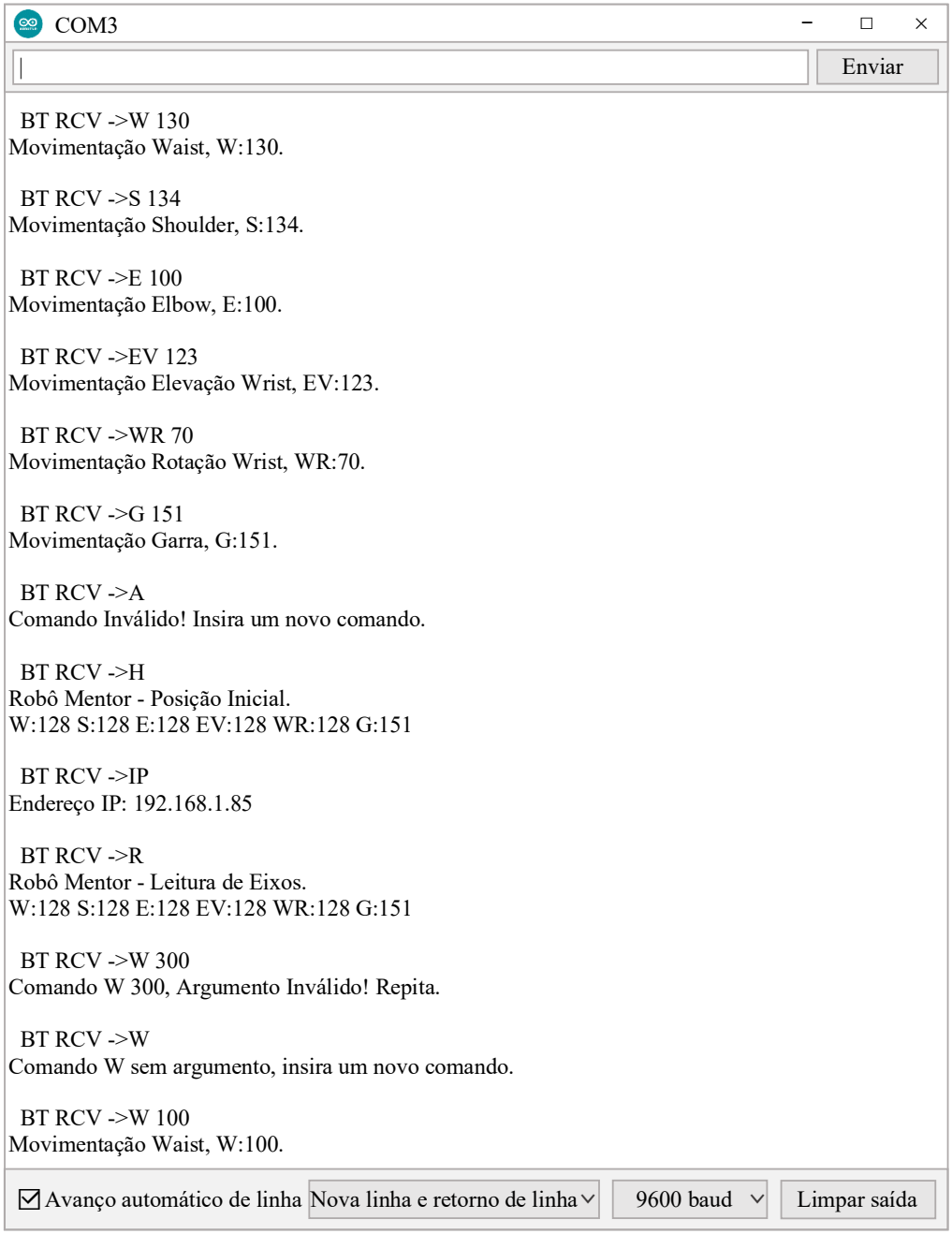


Figura 43 - Exemplo de monitorização da comunicação Bluetooth para controlo do Mentor (via monitor Série Arduino).

Na Figura 43, estão representados os comandos enviados por Bluetooth para controlo do Mentor, a ser monitorizados a partir do monitor série do Arduino. Quando o comando é enviado a partir do monitor Bluetooth, o monitor série assinala o envio do comando com o distintivo "BT RCV ->".

Na comunicação Wi-Fi, o utilizador realiza o controlo do Mentor acedendo a um servidor Web através de um endereço IP. Este servidor Web é implementado no microcontrolador, e o endereço IP é fornecido via comunicação Série ou comunicação Bluetooth através da resposta obtida no envio do comando "IP", isto claro, se o sistema estiver ligado a uma rede Wi-Fi disponível, ou também disponível logo no cabeçalho, no início de uma execução, com o premir do botão *reset*, onde é visível o endereço IP assim que a ligação *wireless* for estabelecida.

Quando o utilizador aceder ao servidor Web através do endereço IP recebido, é redirecionado para a página de entrada. O servidor Web desenvolvido para o Mentor, é constituído por cinco páginas distintas: A - Página de Entrada (Servidor Web Mentor), Fig.44; B - Página de Entrada Sobre (Servidor Web Mentor), Fig.45; C - Página de Controlo (Servidor Web Mentor), Fig.46; D - Página de Instruções Mentor (Servidor Web Mentor), Fig.47; e E - Página Sobre Mentor (Servidor Web Mentor), Fig.48.

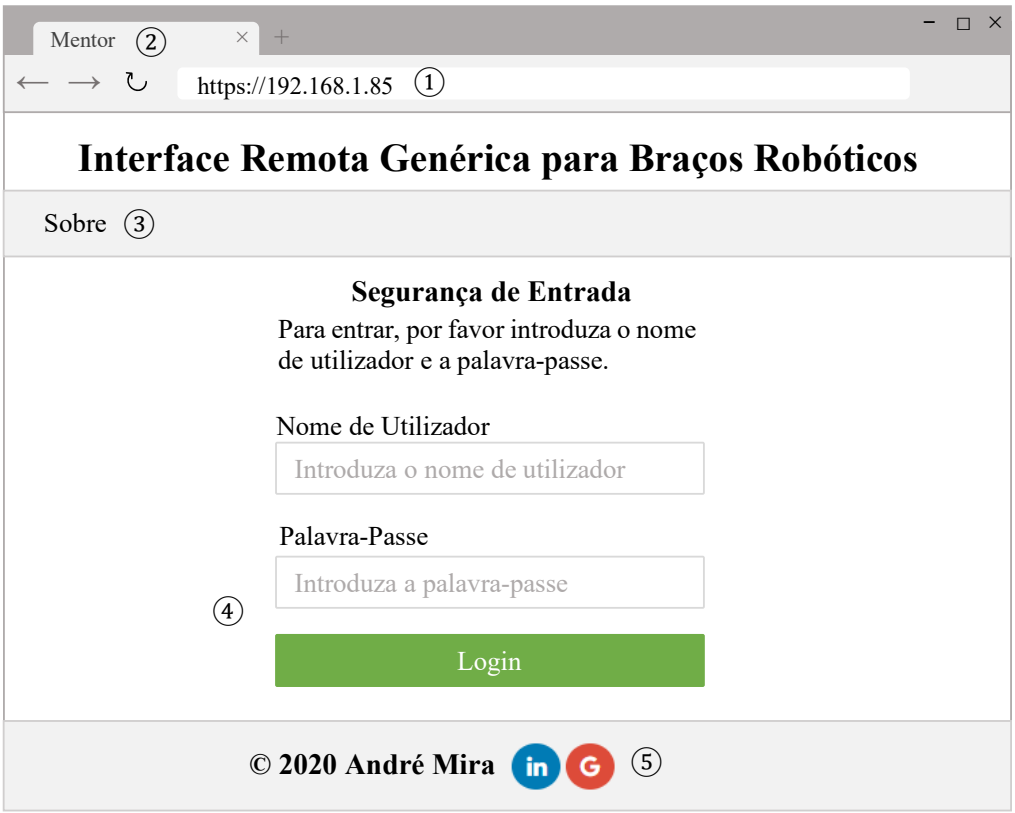


Figura 44 - Página de Entrada do Servidor Web (A).

Na Figura 44, podemos observar a página de entrada do servidor Web desenvolvido para o Mentor. O acesso à página de entrada, resulta do endereço IP recebido através do comando IP, acedido num *browser* como representado em ①.

Em ② está representado o nome que fica na aba em todas as páginas do sevidor Web dedicado ao Mentor. Em ③, sem efetuar login, o utilizador tem acesso, a uma página informativa sobre o servidor Web, que sucintamente tem a explicação do que é a interface para a comunicação Wi-Fi, representada na Figura 45. Através do acesso na barra de navegação superior "Voltar", permite ao utilizador retornar à página de entrada, representada na Figura 44.

O rodapé ⑤, é constante em todas as páginas do servidor Web desenvolvido e tem dois botões de acessos rápidos que permitem a consulta dos contatos do autor do projeto construído. Para aceder à página principal do servidor Web, a página para controlo do Mentor, o utilizador tem de inserir as credenciais de acesso corretas ④. O nome de utilizador é Mentor e a palavra-passe é Mentor.

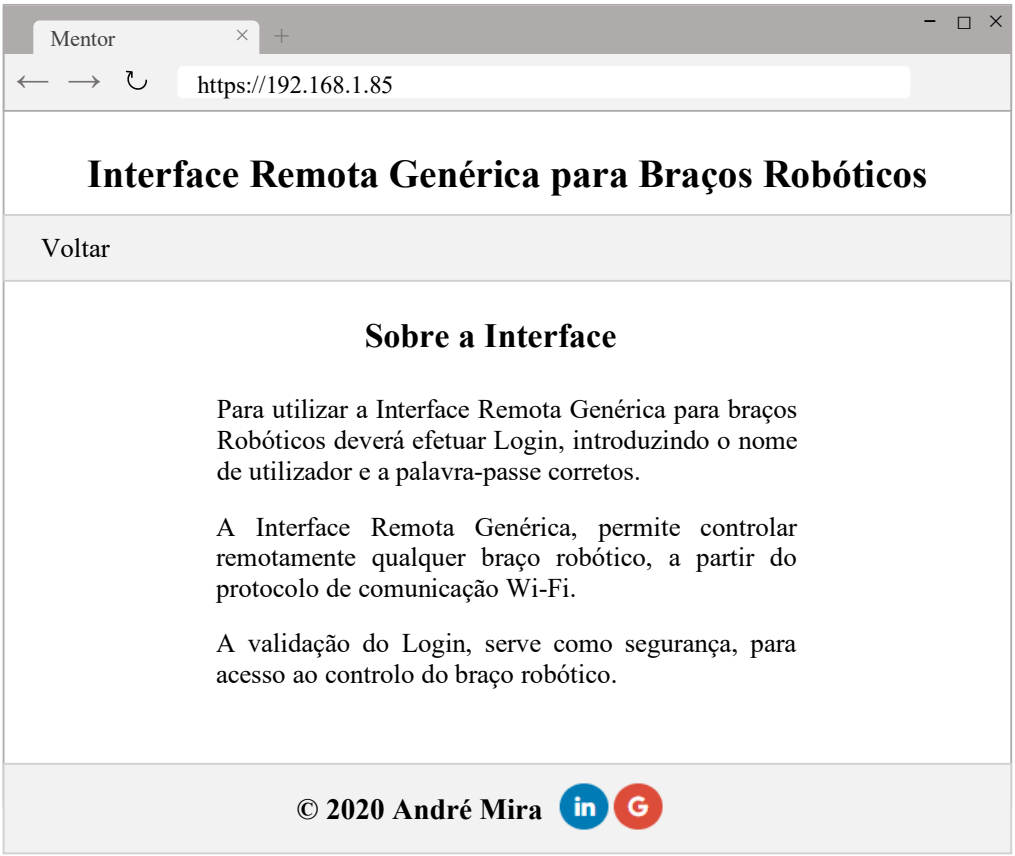


Figura 45 - Informação sobre o Servidor Web (B).

A página de Entrada (A), Figura 44 e a página de Entrada Sobre (B), Figura 45, são independentes do braço robótico a ser controlado. Estas páginas limitam o acesso ao controlador principal do braço robótico, que requerem, o preenchimento de nome de utilizador e palavra-passe.

Efetuado corretamente o login na página de entrada, o utilizador acede à página de controlo do braço robótico Mentor, representada na Figura 46.

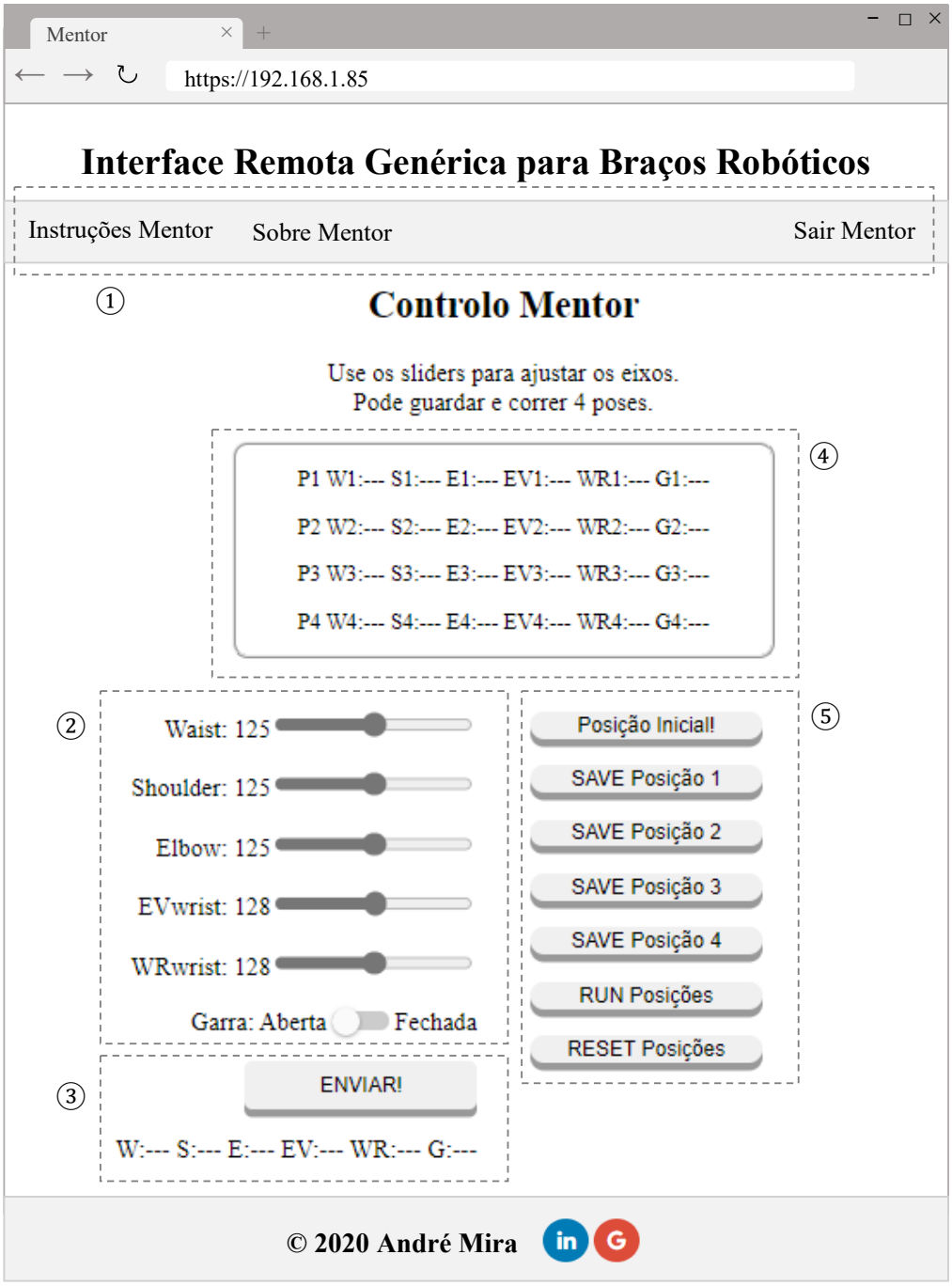


Figura 46 - Página de Controlo do Servidor Web para o Mentor (C).

Nesta página de controlo do Mentor, Figura 46, o utilizador pode aceder, através da barra de navegação superior ①, a uma página de instruções para uso do controlador para o Mentor, através do acesso "Instruções Mentor". Esta página informativa com as instruções para controlo do Mentor, está representada na Figura 47. Com o acesso "Voltar" o utilizador retorna à página de controlo, Figura 46.

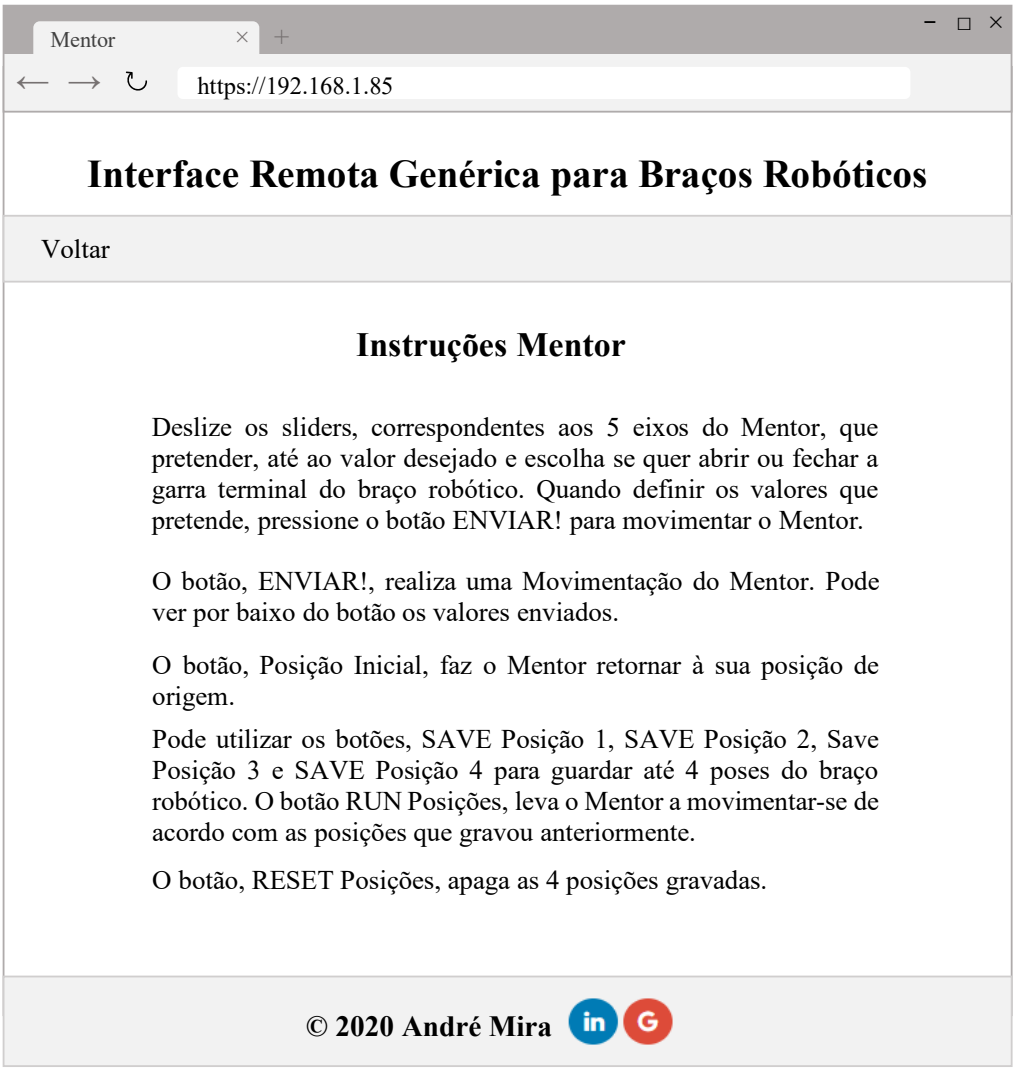


Figura 47 - Página de informação sobre o controlo do Mentor (D).

Através da barra de navegação superior ① da Figura 46, o utilizador pode ainda aceder, a uma página informativa sobre o Mentor, através do acesso "Sobre Mentor". Esta página informativa está representada na Figura 48. Com o acesso "Voltar" o utilizador retorna à página de controlo, Figura 46.

Na página de controlo, Figura 46, com o acesso “Sair” o utilizador volta para a página de entrada, representada na Figura 44.

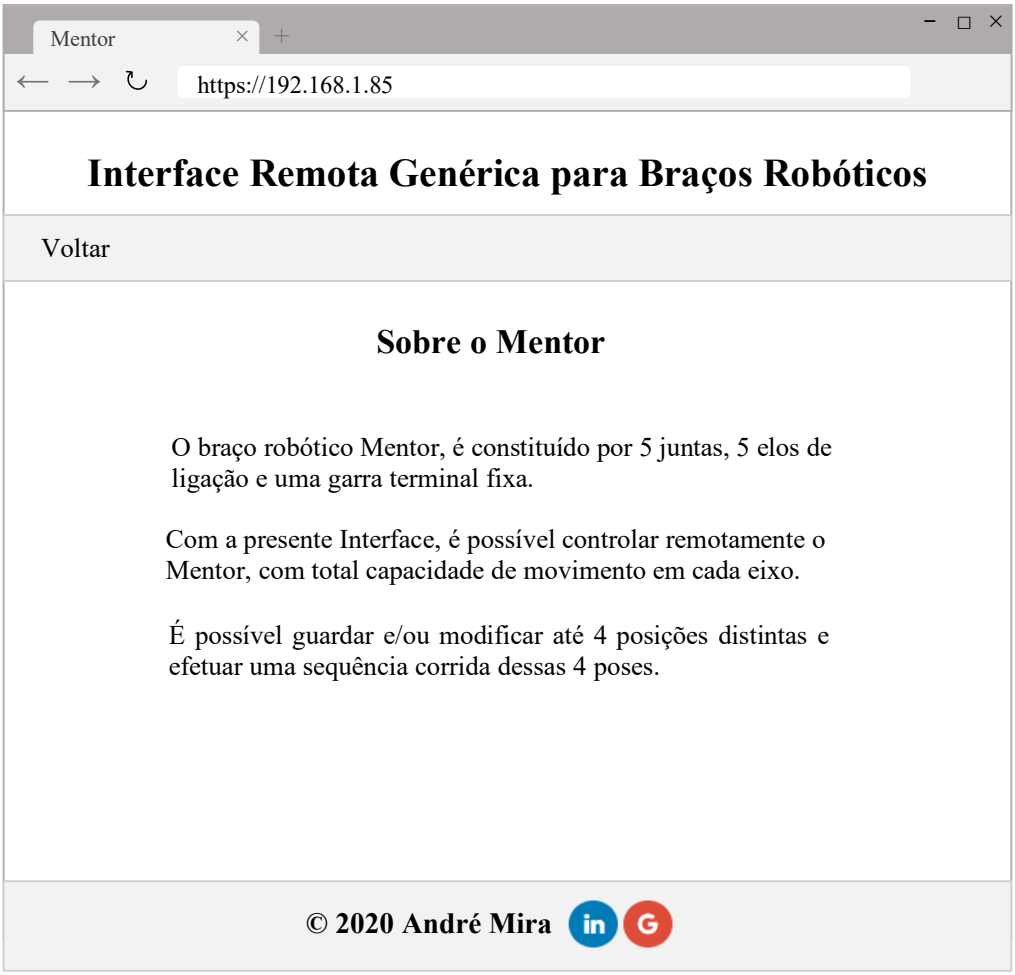


Figura 48 - Página de informação sobre o Mentor (E).

Na página de controlo do Mentor todos os botões e *sliders* são entradas para realizar o controlo do braço robótico. Na Figura 46, em ②, estão representados os cinco *sliders* correspondentes aos cinco graus de liberdade do braço robótico Mentor, permitindo controlar o valor individual de cada um. Tem um botão com dois estados, para a abertura e fecho da garra. Este botão para controlo da garra, tem como estado de repouso a garra aberta e se ativado permite o fecho da garra terminal do Mentor.

Em ③, está o botão “ENVIAR!”, que permite realizar a movimentação do Mentor de acordo com os seis valores estabelecidos em ②. Ao usar o botão “ENVIAR!” para a movimentação do Mentor, os valores enviados são visualizados por baixo do botão e só voltam a ser atualizados com um novo pressionar do mesmo.

Em ⑤, está uma coluna de sete botões específicos. O primeiro botão, "Posição Inicial!", comanda o Mentor para a sua *home position.* Os seguintes botões, são os botões "SAVE Posição" que

permite gravar individualmente até quatro posições, em que cada uma, tem os seis valores de movimentação associados. Os valores de cada posição gravada podem ser monitorizados no ecrã ④. Através do botão "RUN Posições", o Mentor executa as quatro poses previamente gravadas, sempre que pressionado. O botão "RESET Posições", apaga os valores das quatro posições gravadas no ecrã ④.

Na comunicação Wi-Fi, tudo o que for selecionado para controlo do Mentor pode ser monitorizado no monitor Série ou no monitor Bluetooth, desde que ativos. Os *sliders* definem o valor pretendido em cada eixo e o botão "ENVIAR!" dá início à movimentação do Mentor. O botão "RUN Posições" faz o Mentor correr as quatro poses gravadas.

Na Tabela 30, estão resumidas as funções de todos os comandos presentes na página de controlo do servidor Web, para o Mentor.

Inputs servidor Web	Função
Slider Waist	Valor da junta um (J1, waist).
Slider Shoulder	Valor da junta dois (J2, shoulder).
Slider Elbow	Valor da junta três (J3, elbow).
Slider EVwrist	Valor da junta quatro (J4, elevation wrist).
Slider WRwrist	Valor da junta cinco (J5, wrist rotation).
Toogle button Garra	Ação da garra [Repouso-Aberta a Atuado-Fechada].
ENVIAR!	Movimentação do Mentor com os valores selecionados.
Posição Inicial!	Braço Robótico Mentor na posição Home.
SAVE Posição 1	Valores da posição 1 gravados no ecrã.
SAVE Posição 2	Valores da posição 2 gravados no ecrã.
SAVE Posição 3	Valores da posição 3 gravados no ecrã.
SAVE Posição 4	Valores da posição 4 gravados no ecrã.
RUN Positions	Movimentações seguidas do Mentor na P1, P2, P3 e P4.
RESET Positions	Valores das posições 1, 2, 3 e 4 apagadas no ecrã.

Tabela 30 - Comandos da comunicação Wi-Fi no servidor Web para o Mentor.

Na Figura 49 , está representada a monitorização das ações realizadas na página de controlo do servidor Web para o Mentor, no monitor série do Arduino e uma representação dos botões acionados, que desencadeiam os comandos, assinalados, de um a oito.

A execução começa com o botão *reset* do microcontrolador ①, dando início à execução do programa.

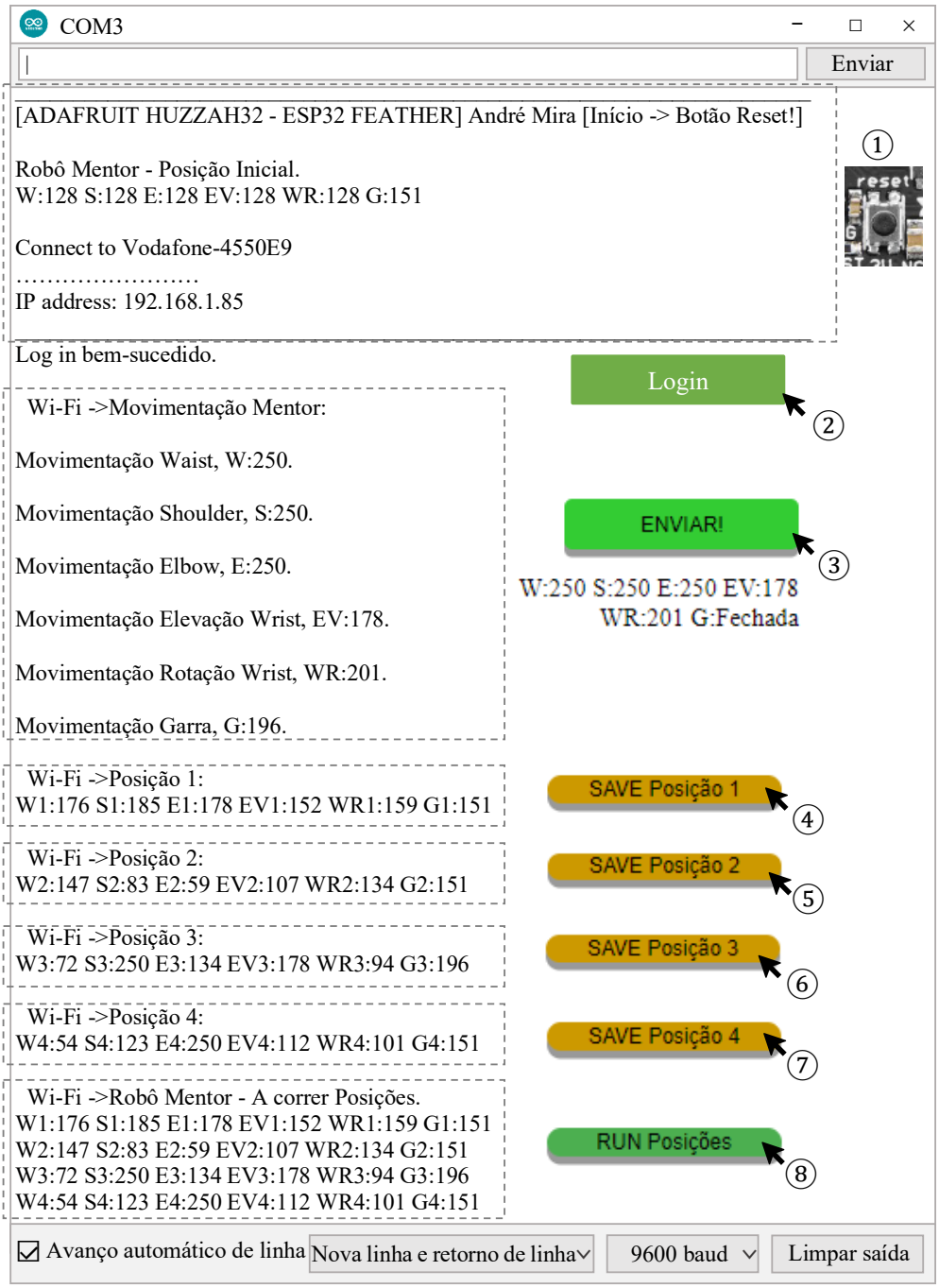


Figura 49 - Exemplo de monitorização da comunicação Wi-Fi para controlo do Mentor (via Monitor série).

O Mentor começa na sua *home position* e no monitor série tem-se acesso ao endereço IP do servidor Web desenvolvido assim que a ligação à internet é concluída, podendo-se observar a que SSID está feita a ligação, sendo neste caso a "Vodafone-4550E9".

Caso o utilizador não estivesse ligado a um monitor série, o endereço IP podia ser pedido pelo monitor Bluetooth com o comando IP. Acedendo ao endereço IP, foi realizado o *login* com as credenciais corretas e obtém-se a resposta nos dois monitores, de *log in* bem-sucedido ②.

Uma vez na página de controlo, foi escolhido o valor dos *sliders* e carregado o botão "ENVIAR!" ③ que comanda a movimentação do Mentor. Depois foram gravadas quatro posições distintas para o Mentor em ④⑤⑥⑦ e carregado no botão "RUN Posições" ⑧, que faz o Mentor executar as quatro posições pré-gravadas.

Na Figura 50, pode-se observar a monitorização do processo de envio de comandos via servidor Web, no monitor Bluetooth.

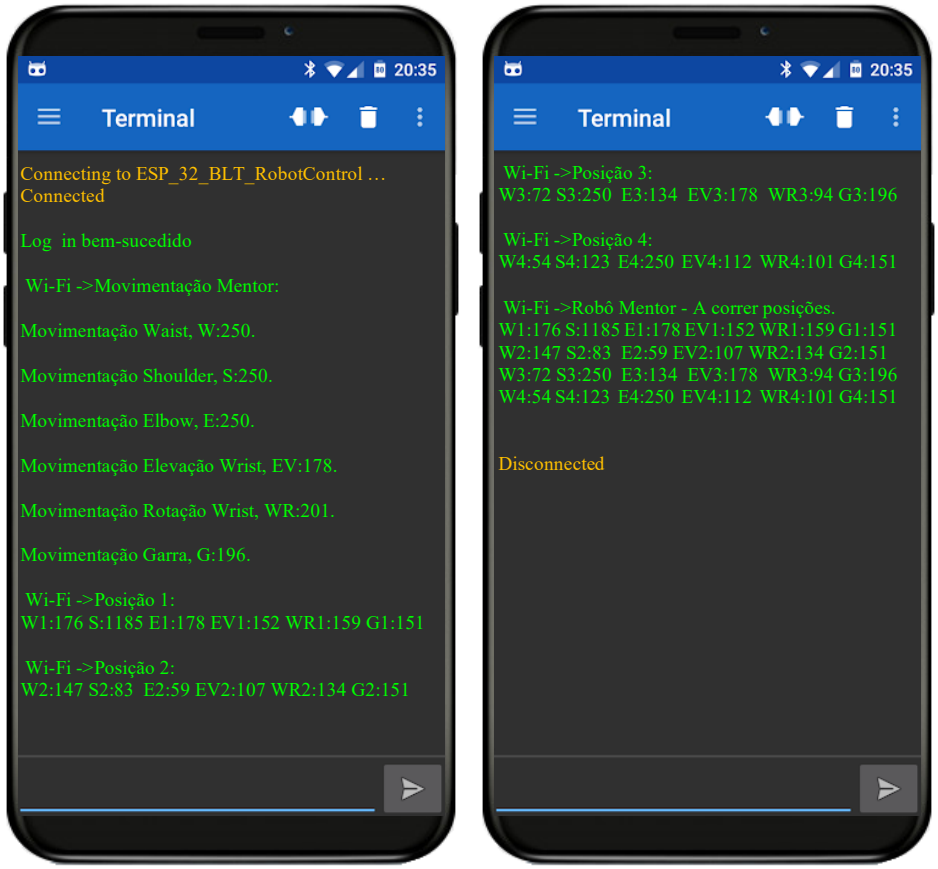


Figura 50 - Exemplo de monitorização da comunicação Wi-Fi para controlo do Mentor (via Monitor Bluetooth).

Os comandos usados pelo utilizador no servidor Web, são registados para monitorização no monitor Série e/ou no monitor Bluetooth como representado na Figura 49 e na Figura 50.

4.2 Interface remota genérica: controlo do Dobot Magician

A IRGBR para o Dobot Magician funciona alimentando o microcontrolador a um portátil ou a uma *powerbank* e ligando-o ao braço robótico Dobot Magician. Com o código descarregado para o microcontrolador, a interface está pronta a utilizar. O utilizador tem disponível três comunicações para realizar o controlo do Dobot Magician, comunicação Série, comunicação Bluetooth e comunicação Wi-Fi. O Dobot Magician é um braço robótico controlado em coordenadas tridimensionais no plano xyz.

A comunicação série, é estabelecida ligando o microcontrolador ao software Arduino e acedendo ao monitor série. A partir do monitor, pode-se enviar e receber comandos para o Dobot Magician.

Na Figura 51, está representada a janela do Monitor série do Arduino, com o exemplo de execução de vários comandos, na interface de controlo do Dobot Magician via comunicação série. Para controlo do Dobot Magician, o monitor série, tem de ser configurarado para operar a 115200 baud e com avanço automático de linha ativo.

A execução do programa começa ao premir o botão *reset* no microcontrolador. No monitor série o utilizador recebe o cabeçalho da comunicação como representado na Figura 51 em (A). O cabeçalho da interface na comunicação série, delimita o uso do botão *reset* entre várias execuções do código. Assim que o código inicia, a primeira ação é movimentar o Dobot Magician para a sua *home position.* Depois é realizada a ligação Wi-Fi, assim que a ligação é estabelecida, o endereço IP fica disponível, para acesso ao servidor Web do Dobot Magician. Ainda no cabeçalho o utilizador tem uma mensagem com os comandos disponíveis para o Dobot, “Comunicação Série (Dobot) \n Comandos Disponíveis: H, P, R, A e IP”.

Em (B), está representado o envio de vários comandos para comunicação série com o Dobot Magician. Neste exemplo o primeiro comando enviado foi P 250.0 150.0 75.0 50.0, comando que faz o Dobot Magician movimentar-se para as coordenadas (250.0; 150.0; 75.0) e realizar uma torção do elemento terminal, obtendo como resposta, “Robô Dobot \n P 250.000 150.000 75.000 50.000”, que confirma o posicionamento do Dobot Magician.

Segue-se o envio do comando H, um comando para posicionar o Dobot na posição *Home*, no qual se obtém a resposta “Robô Dobot - Posição Inicial. \n X:260.5 Y:0 Z:-8.5 R:0”.

No envio do comando R, comando para obter a posição atual do Dobot no espaço e da torção do elemento terminal, o utilizador obtém como resposta “Robô Dobot - Leitura Coordenadas X:260.5 Y:0 Z:-8.5 R:0”.

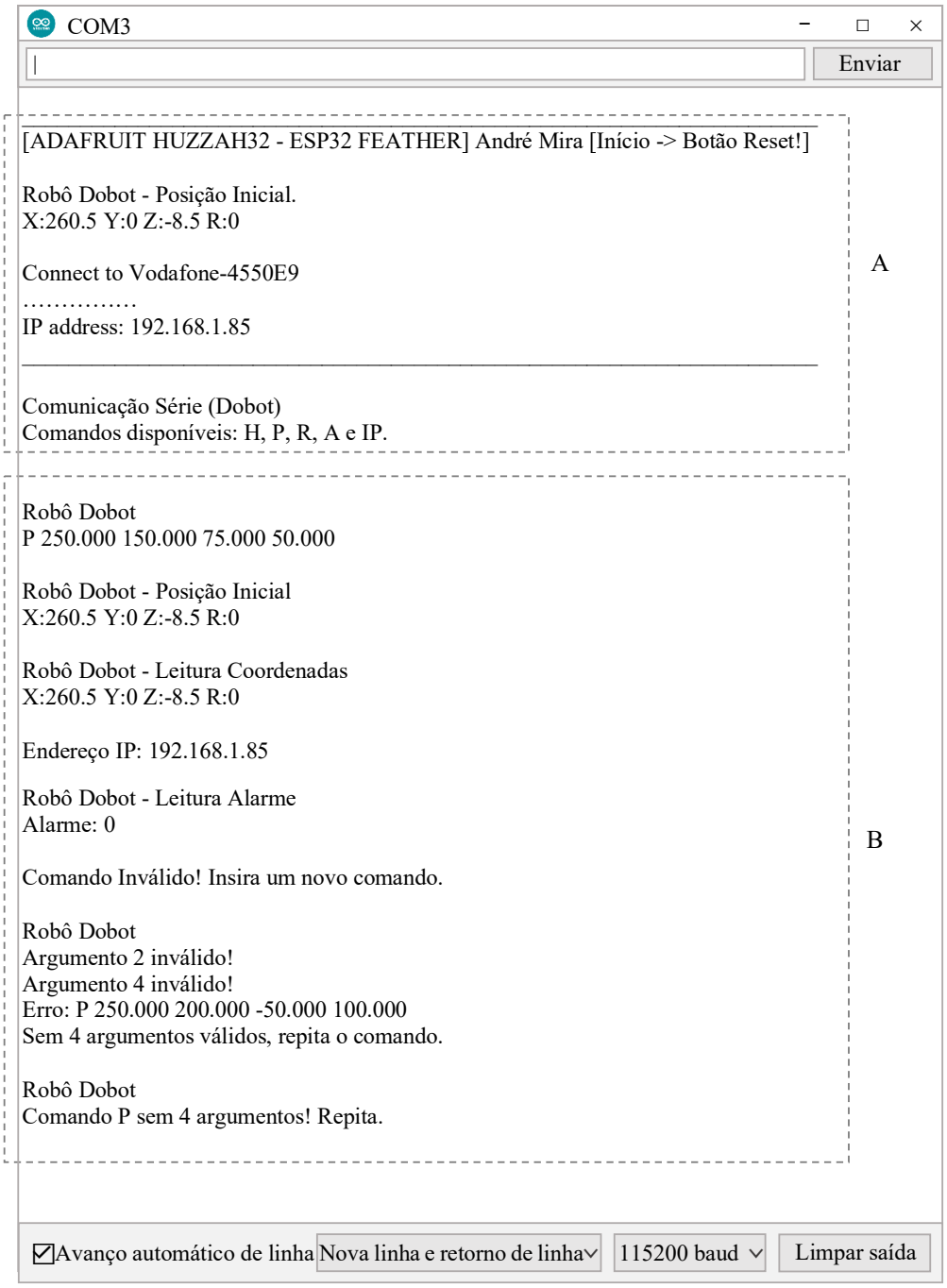


Figura 51 - Exemplo de comunicação série para controlo do Dobot Magician (envio via Monitor Série Arduino).

O envio do comando IP, serve para receber o endereço IP que permitirá o acesso ao servidor Web, tendo-se o obtido a resposta, “Endereço IP: 192.168.1.85”.

O envio do comando A, permite ao utilizador ler o alarme do Dobot Magician, no exemplo o Dobot não tinha nenhum alarme ativo por isso obteve-se como resposta, “Robô Dobot - Leitura Alarme \n Alarme: 0”. De seguida foi enviado o comando G, para se demonstrar a resposta na introdução de um comando inválido, em que o utilizador obtém como resposta “Comando Inválido! Insira um novo comando.”. No envio do comando P 250 200 -50 100, o segundo e quarto argumentos não se encontram dentro dos limites estabelecidos para o correto funcionamento do Dobot, assim valida-se a resposta no envio de um comando válido mas com argumentos inválidos, obtendo-se a resposta, “Robô Dobot \n Argumento 2 inválido! \n Argumento 4 inválido! \n Erro: P 250.000 200.000 -50.000 100.000 \n Sem 4 argumentos válidos, repita o comando.”, demonstrando que a IRGBR assinala quais os argumentos fora dos limites, assinalando como erro e não movimentando o braço robótico. O envio do comando P permite obter a resposta no envio de um comando válido mas sem quatro argumentos, no qual o utilizador recebe “Robô Dobot \n Comando P sem 4 argumentos!Repita.”. Os comandos introduzidos e enviados a partir do monitor série do Arduino podem também ser monitorizados a partir do monitor Bluetooth da aplicação *Serial Bluetooth Terminal* como representado na Figura 52.

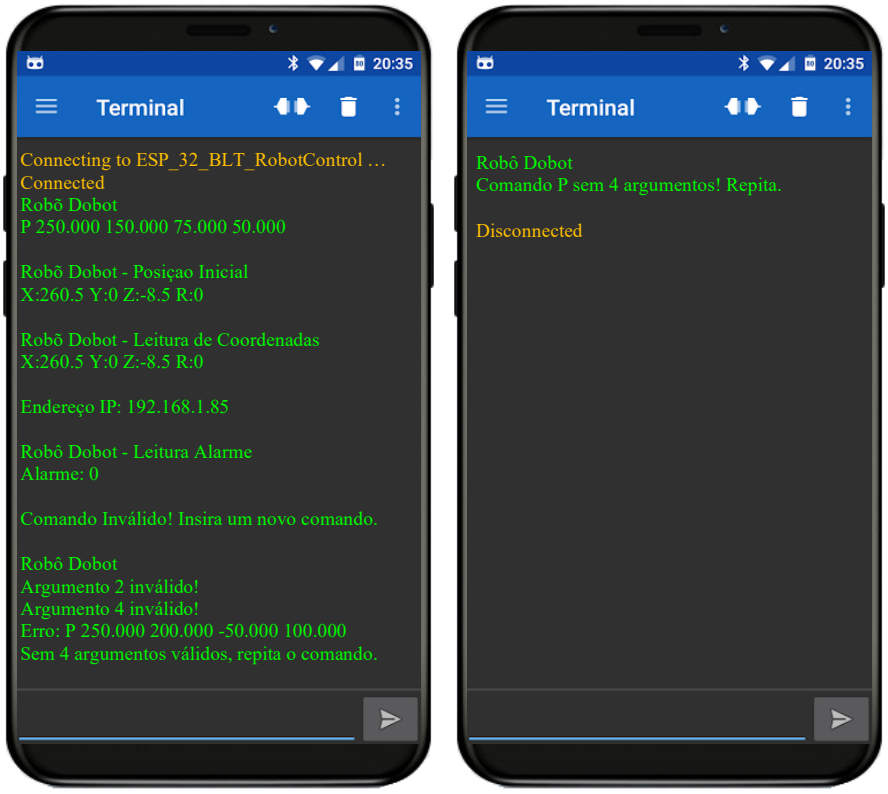


Figura 52 - Exemplo de monitorização da comunicação série para controlo do Dobot Magician (via Monitor Bluetooth).

A comunicação Bluetooth funciona a partir da aplicação *Serial Bluetooth Terminal* que disponibiliza um Monitor Bluetooth semelhante ao monitor série. Os comandos que o utilizador desejar são enviados pelo canal Bluetooth para o microcontrolador e são escritos no monitor série, se estiver ativo.

Na Figura 53, está representado o exemplo de uma execução de envio de comandos, via Bluetooth pelo *Serial Bluetooth Terminal*.

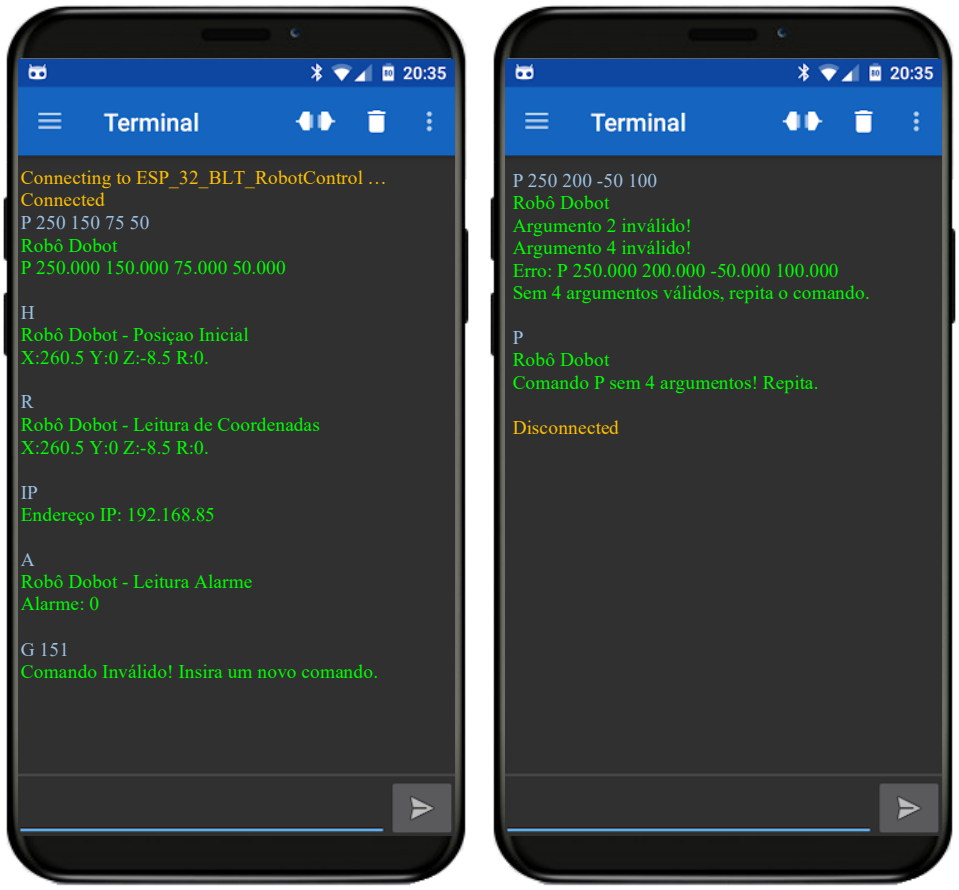


Figura 53 - Exemplo de comunicação Bluetooth para controlo do Dobot Magician (envio via Monitor Serial Bluetooth Terminal).

A ligação com o dispositivo Bluetooth, é iniciada e escrita a amarelo. Os comandos enviados pelo utilizador, representados a azul, e a resposta a um comando enviado é representada a verde. Os comandos enviados no monitor Bluetooth da Figura 53, são os mesmos enviados via monitor série, descritos anteriormente.

Na Figura 54, estão representados os comandos enviados por Bluetooth monitorizados no monitor série do Arduino. Quando o comando é enviado para o Dobot Magician a partir do monitor Bluetooth o monitor série assinala o envio do comando com o distintivo "BT RCV ->".

Figura 54 - Exemplo de monitorização da comunicação Bluetooth para controlo do Dobot Magician (via Monitor Série Arduino).

Na comunicação Wi-Fi, o utilizador pode aceder a um servidor Web através do endereço IP. Este servidor Web é implementado no microcontrolador, o endereço é através do mesmo procedimento do Mentor descrito anteriormente.

Quando o utilizador aceder ao servidor Web através do endereço IP, é redirecionado para a página de entrada. O servidor Web para o Dobot Magician é constituído por cinco páginas distintas: A -

Página de Entrada (Servidor Web Dobot Magician); B - Página de Entrada Sobre (Servidor Web Dobot Magician); C - Página de Controlo (Servidor Web Dobot Magician), Fig.56; D - Página de Instruções Dobot (Servidor Web Dobot Magician); E - Página Sobre Dobot (Servidor Web Dobot Magician).

As páginas de Entrada e sobre da página de entrada, são independentes do braço robótico controlado e já foram descritas anteriormente na Secção 4.1, representadas na Figura 44 e na Figura 45. As credenciais de acesso para *login* na página de controlo do Dobot Magician são, nome de utilizador Dobot e palavra-passe Dobot, como representado na Figura 55.

Figura 55 - Credenciais de acesso para controlo do Dobot Magician.

Depois de efetuado corretamente o *login* na página de entrada, o utilizador tem acesso à página de controlo do Dobot Magician, representada na Figura 56. Esta página é muito semelhante à anteriormente analisada, página de controlo do Mentor. O *layout* para qualquer braço robótico é semelhante, representado na Figura 56 por ①. Através da barra de navegação superior, os acessos a uma página de instruções para uso do controlador do Dobot, através do Botão "Instruções Dobot", o acesso a uma página que explica resumidamente o Dobot, através do botão "Sobre Dobot", ou sair da página de controlo, com o Botão "Sair Dobot", voltando para a página de entrada.

Na página de controlo para o Dobot Magician todos os botões e *sliders* são entradas para realizar o controlo do braço robótico. Em ②, estão representados os quatro *sliders* correspondentes aos valores das três coordenadas espaciais e da torção do elemento terminal, permitindo controlar o valor individual de cada um. Em ③, está o botão "ENVIAR!", que permite realizar a movimentação do Dobot de acordo com os valores estabelecidos em ②. Ao usar o botão "ENVIAR!" para a movimentação do Dobot, os valores enviados são afixados por baixo do respetivo botão, e só voltam a ser atualizados quando for novamente pressionado.

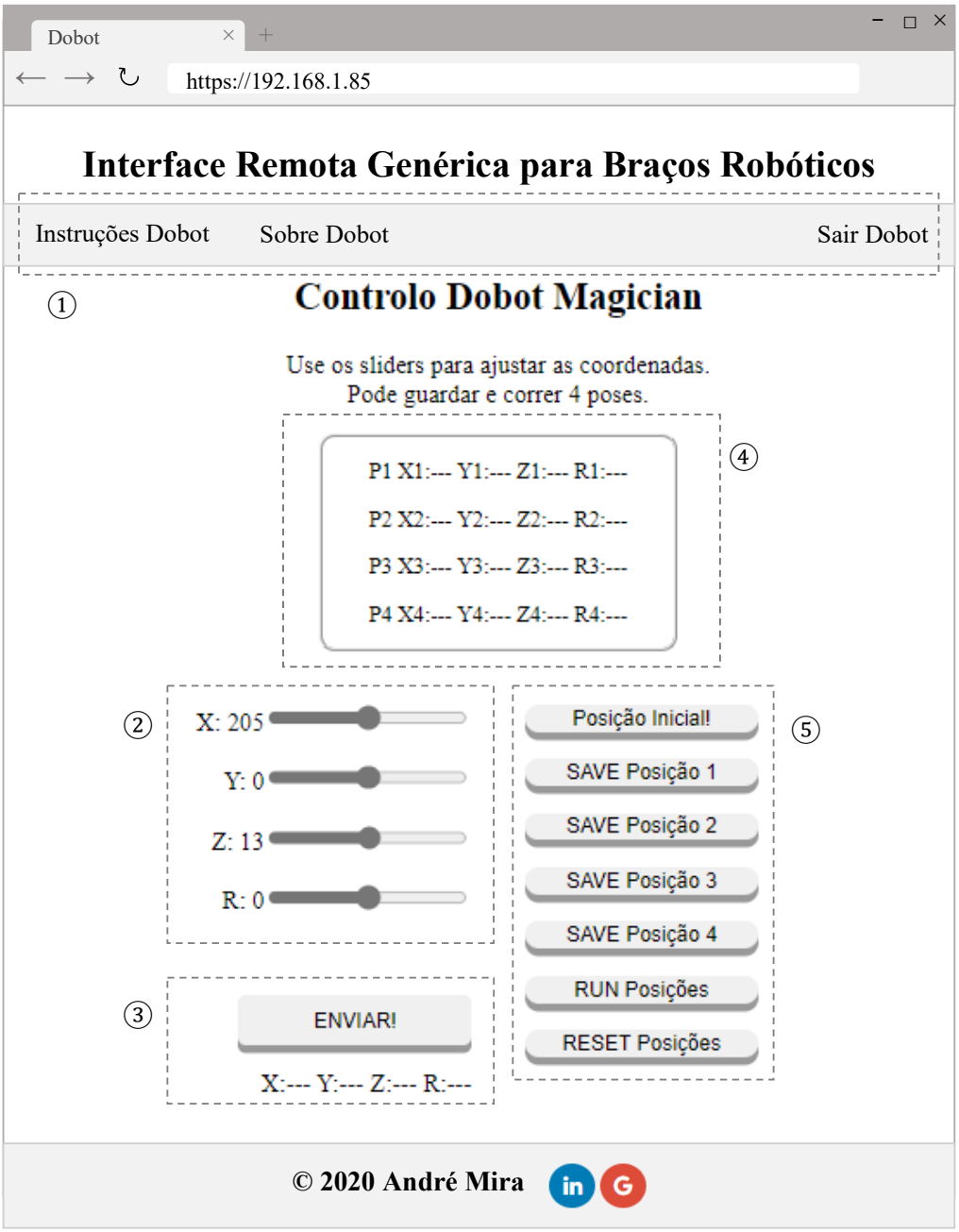


Figura 56 - Página de Controlo do Servidor Web para o Dobot Magician (C).

Em ⑤, está uma coluna de sete botões específicos. O primeiro botão, "Posição Inicial!", faz o Dobot movimentar-se para a sua *home position.* Os seguintes botões, são os botões "SAVE Posição" que permitem gravar individualmente até quatro posições, cada uma com os quatro valores de movimentação associados. Os valores de cada posição armazenada podem ser monitorizados no ecrã ④. Através do botão "RUN Posições", o Dobot executa as quatro

posiçõess previamente armazenadas, sempre que pressionado. O botão “RESET Posições”, apaga os valores das quatro posições armazenadas.

Na comunicação Wi-Fi, todos os comandos do Dobot podem ser monitorizados no monitor série ou no monitor Bluetooth, desde que ativos. As entradas do servidor Web estão detalhadas na Tabela 31.

Inputs servidor Web	Função
Slider X	Valor da coordenada x no plano xyz.
Slider Y	Valor da coordenada y no plano xyz.
Slider Z	Valor da coordenada z no plano xyz.
Slider R	Valor da torção do acessório terminal.
ENVIAR!	Movimentação do Mentor com os valores selecionados.
Posição Inicial!	Braço Robótico Mentor na posição Home.
SAVE Posição 1	Valores da posição 1 gravados no ecrã.
SAVE Posição 2	Valores da posição 2 gravados no ecrã.
SAVE Posição 3	Valores da posição 3 gravados no ecrã.
SAVE Posição 4	Valores da posição 4 gravados no ecrã.
RUN Positions	Movimentações seguidas do Mentor na P1, P2, P3 e P4.
RESET Positions	Valores das posições 1, 2, 3 e 4 apagados no ecrã.

Tabela 31 - Comandos da comunicação Wi-Fi para o Dobot Magician.

Na Figura 57, está representada a monitorização das ações realizadas no servidor Web, no monitor série do Arduino e os botões acionados pelo utilizador que desencadeiam os comandos do Dobot.

A execução começa com o botão *reset* do microcontrolador, Figura 57 ①, dando início à execução do programa. O Dobot começa na sua *home position* e no monitor série tem-se acesso ao endereço IP do servidor Web. Acedendo ao endereço IP, é realizado o login com as credenciais e obteve-se a resposta nos dois monitores, de *log in* bem-sucedido ②.

Uma vez na página de controlo, foi selecionado o valor dos *sliders* e premido o botão ENVIAR! ③. Depois foram armazenadas quatro posições distintas para o Mentor em ④⑤⑥⑦ e premido o botão “RUN Posições” ⑧, que faz o Dobot Magician executar as quatro posições.

Ao usar o botão “RESET Posições” ⑨, as quatro posições são apagadas. Por último com o botão “Posição Inicial” ⑩, o Dobot Magician volta à sua *home position*.

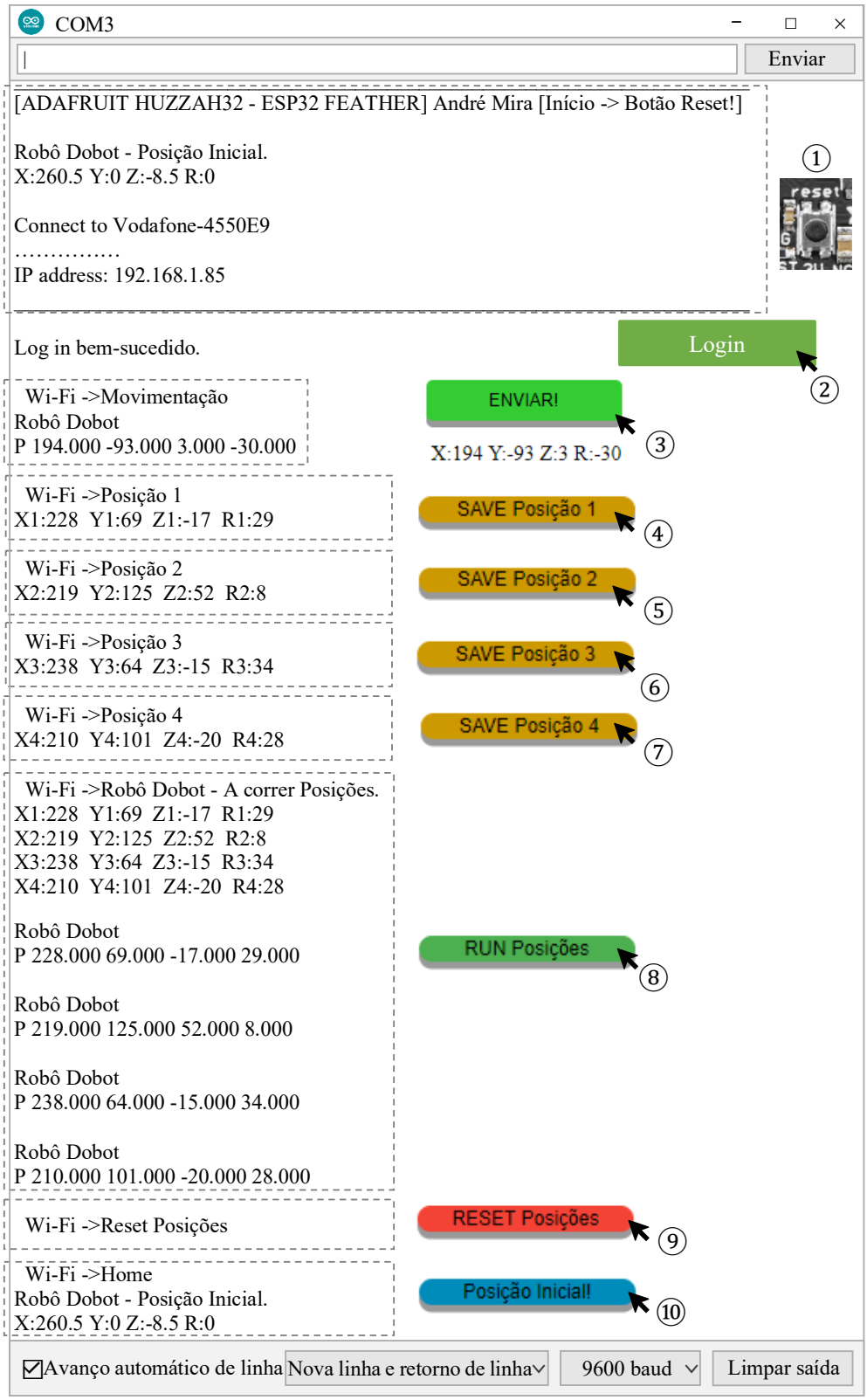


Figura 57 - Exemplo de monitorização da comunicação Wi-Fi para controlo do Dobot Magician (via Monitor Série Arduino).

Na Figura 58, está representada a monitorização dos comandos acionados na página de controlo do Dobot Magician do servidor Web, no monitor Bluetooth.

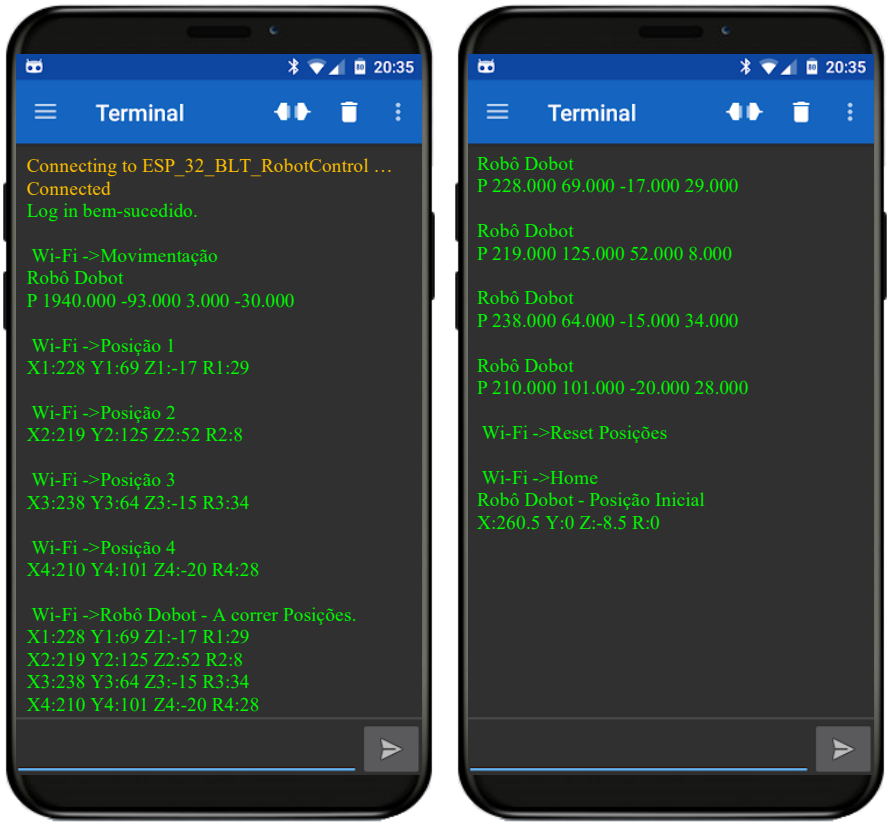


Figura 58 - Exemplo de monitorização da comunicação Wi-Fi para controlo do Dobot Magician (via Monitor Bluetooth).

Os comandos enviados pelo utilizador no servidor Web são enviados para as monitorizações série, e Bluetooth. Ambos os monitores registam os comandos usados no servidor Web sendo possível ter ambos em simultâneo.

O acesso do utilizador à página de instruções do Dobot e à página sobre o Dobot, a partir da página de controlo do Dobot Magician, é idêntico ao descrito anteriormente para o servidor Web do Mentor, alterando apenas o texto descritivo em cada uma das páginas.

5

Conclusões e Trabalho Futuro

“ We can’t solve problems by using the same kind of thinking

we used when we created them. ”

Albert Einstein

5. Conclusões e Trabalho Futuro

Neste Capítulo são referidas as ideias e aprendizagens principais consequentes do trabalho desenvolvido no âmbito deste projeto. São também equacionados os tópicos de maior relevo, para desenvolvimentos futuros que poderão complementar este projeto.

5.1 Conclusões

A presente dissertação de projeto final de Mestrado, teve por base o estudo teórico, a simulação e o desenvolvimento de um protótipo experimental de uma interface remota genérica, concebida para o controlo de braços robóticos. O trabalho foi iniciado com a seleção e estudo do controlador, e dos braços robóticos que a interface desenvolvida permite controlar.

A interface remota genérica para controlo de braços robóticos foi desenvolvida com sucesso, conforme documentado no Capítulo 4. Conseguiu-se desenvolver uma interface através de uma única camada de software correspondente ao braço robótico, que possibilita o controlo do mesmo através de três comunicações distintas: *i)* comunicação série; *ii)* comunicação Bluetooth; e *iii)* comunicação Wi-Fi. A interface desenvolvida demonstra que é possível controlar remotamente com sucesso os braços robóticos Mentor [34] e Dobot Magician [24], através das três comunicações (série, Bluetooth e Wi-Fi). Considera-se genérica, pois esta foi desenvolvida tendo por base uma construção por camadas de software, que se inserem conforme o braço robótico a controlar, ou seja, está preparada para a inserção de novos braços robóticos e está demonstrado o seu funcionamento com o controlo bem-sucedido dos dois braços robóticos aqui considerados, o Mentor e o Dobot Magician. Para a interface remota genérica desenvolvida, o procedimento de introduzir novos manipuladores é simples, basta implementar os comandos de novos braços robóticos, mantendo toda a estrutura existente, e esses novos braços robóticos poderão ser controlados com recurso às três comunicações desenvolvidas.

Na IRGBR desenvolvida a comunicação série resulta no envio e receção de comandos através de um monitor série. Na comunicação série o envio e a receção de comandos podem ser monitorizados em simultâneo num monitor Bluetooth conforme descrito no Capítulo 4.

A comunicação Bluetooth da IRGBR resulta no envio e receção de comandos através de um monitor Bluetooth, tendo sido usado neste projeto, a aplicação *Bluetooth Serial Terminal* [33], como descrito no quarto capítulo. Na comunicação Bluetooth o envio e a receção de comandos funcionam a partir do monitor Bluetooth, à semelhança da comunicação série, estes podem ser monitorizados em simultâneo num monitor série.

A comunicação Wi-Fi realiza-se através do acesso a um servidor Web. São necessárias credenciais de acesso para aceder à página dedicada ao controlo do braço robótico adotado. O endereço IP do servidor pode ser obtido através do monitor Série ou Bluetooth. As páginas foram desenvolvidas de forma a adaptarem-se a todos os ecrãs modernos de várias dimensões, ou seja, adapta-se a um portátil, *tablet*, *smartphone* ou outro dispositivo eletrónico.

A interface desenvolvida tem a particularidade de possibilitar o uso destes três modos de comunicações em simultâneo sem afetar as interações com o controlo do braço robótico. O utilizador pode ter um monitor série e um monitor Bluetooth ativos bem como a página Web de controlo do braço robótico e alternar entre as três possibilidades, no envio de comandos, e mantendo o acesso às monitorizações nos monitores série e Bluetooth.

Os braços robóticos adotados, serviram para aperfeiçoar e comprovar o funcionamento da IRGBR desenvolvida. Conseguiu-se assim o controlo de dois braços robóticos, ambos manipuladores, com características diferentes. O Mentor um braço robótico mais antigo, que apesar de não ser utilizado há mais de 20 anos, funciona, tendo sido recondicionado e adaptado com sucesso para este projeto. Para o Dobot Magician, um braço robótico mais recente, o seu protocolo de comando possibilita um maior número de comandos, no entanto, apenas os mais relevantes foram considerados neste projeto, uma vez que o objetivo não era o desenvolvimento do software dedicado ao Dobot, mas sim implementar uma interface simples e genérica para vários braços robóticos. Na implementação do comando para obter os alarmes do Dobot Magician por falta de informação disponível por parte do fabricante, não foi possível ter acesso a todos os códigos de alarme disponíveis no próprio software do mesmo, e por isso foi realizada uma adaptação na eliminação dos alarmes.

Para ambos estes braços robóticos foram testadas as suas movimentações e os valores disponíveis quer para movimentação das juntas do Mentor ou das coordenadas para o Dobot, tendo sido balizadas de forma a que ambos os braços evitassem situações impossíveis e não se danificassem.

5.2 Trabalho Futuro

A interface remota genérica desenvolvida, demonstrou que consegue controlar pelo menos os dois braços robóticos aqui considerados, o Mentor e o Dobot Magician, com recurso às comunicações Série, Bluetooth e Wi-Fi.

Como trabalho futuro, podem ser analisados novos braços robóticos, para serem acrescentados em novas camadas de software para a interface remota genérica realizar o seu controlo. De notar

que o trabalho está preparado para receber novos braços, em que facilmente são implementados novos comandos recorrendo às interfaces de comunicação desenvolvidas.

Além das três interfaces de comunicação desenvolvidas neste projeto, será interessante desenvolver novas possibilidades de comunicação. O acesso a outros protocolos de comunicação, além dos já desenvolvidos, como *Controller Area Network* (CAN) [35] ou o I^2C [36] introduziriam novas possibilidades no controlo de braços robóticos. Como por exemplo, a cooperação de vários robôs em simultâneo numa única rede, em que o utilizador poderia controlar uma linha com vários robôs através de uma só interface. A implementação do protocolo CAN, serviria por exemplo, para o conceito de controlar vários braços robóticos através da continuação do desenvolvimento da presente IRGBR. Outro protocolo interessante a desenvolver e implementar poderia ser o I^2C, que permitiria o controlo de vários braços robóticos sendos *slaves,* sendo o microcontrolador *o master* e tendo assim o utilizador acesso a uma linha com vários braços robóticos ou na cooperação de vários *masters* possibilitando o controlo de um ou mais braços robóticos, *slaves*.

Com o desenvolvimento de novos protocolos, com intuito de comandar vários braços robóticos em simultâneo, com vista ao objetivo de formar uma linha de várias tarefas sequenciais atribuídas a cada robô, seria interessante desenvolver e implementar mecanismos de exclusão associados à presente IRGBR. Estes mecanismos de exclusão basicamente controlariam o acesso aos braços robóticos. Uma vez que um utilizador tem o controlo de toda a linha de robôs, na ligação de um segundo ou mais utilizadores com credenciais de acesso, teria que ser estabelecida uma hierarquia ou um mecanismo de exclusão capaz de selecionar qual o utilizador que permaneceria no controlo de todos os braços robóticos, em que numa primeira abordagem, mais simples, o primeiro utilizador ligado teria acesso ao controlo e todos os seguintes ficariam em espera podendo aceder ao controlo apenas um de cada vez.

Bibliografia

[1] K. S. Fu, R. C. Gonzalez e C. S. G. Lee, “Robotics: Control, Sensing, Vision and intelligence”, McGraw-Hill International Editions, Singapore, 1987.

[2] M. Boettcher, “Revolução Industrial “ [Online] Available: https://www.linkedin.com/pulse/revolu%C3%A7%C3%A3o-industrial-um-pouco-de-hist%C3%B3ria-da-10-at%C3%A9-boettcher/ [Acedido em 25.11.2020].

[3] Mikell P. Groover, Mitchell Weiss, Roger N. Nagel e Nicholas G. Odrey, “Industrial Robotics: Technology, Programming, and Applications”, McGraw-Hill International Editions, Singapore, 1986.

[4] International Federation of Robotics, Robot History, “Timeline” [Online] Available: https://ifr.org/robot-history [Acedido em 25.11.2020].

[5] ABB Robotics, Industrial Robots, “IRB 120” [Online] Available: https://new.abb.com/products/robotics/industrial-robots/irb-120 [Acedido em 25.11.2020].

[6] ABB Robotics, Industrial Robots, “IRB 360 FlexPicker” [Online] Available: https://new.abb.com/products/robotics/industrial-robots/irb-360 [Acedido em 05.11.2020].

[7] ISO, “ISO/TS 15066:2016 - Robots and robotic devices - Collaborative robots.”

[8] Peter Corke, Robotics, “Vision and control: Fundamental Algorithms in Matlab” (second edition), Springer, 2017.

[9] Shadow Robot Company, “Tactile Telerobot” [Online] Available: https://www.shadowrobot.com/telerobots/ [Acedido em 27.11.2020].

[10] Secção de Automação e Robótica, Sebenta Sistemas Robóticos, Instituto Superior de Engenharia de Lisboa, 2019.

[11] Matjaž Mihelj, Tadej Bajd, Aleš Ude, Jadran Lenarčič, Aleš Stanovnik, Marko Munih, Jure Rejc, Sebastjan Šlajpah, “Robotics” (second edition), Springer, 2019.

[12] “Aplicações da Robótica Industrial” [Online] Available: https://blog.infaimon.com/pt/aplicaciones-de-la-robotica-industrial/ [Acedido em 25.11.2020].

[13] O Globo, “Robôs” [Online] Available: https://oglobo.globo.com/economia/em-mundo-repleto-de-robos-montadoras-contratam-mais-humanos-23082727 [Acedido em 25.11.2020].

[14] Fersiltec Automação Industrial [Online] Available: https://fersiltec.com.br/blog/robotica/impacto-robotica-industria/ [Acedido em 25.11.2020].

[15] “IEC” [Online] Available: https://www.iec.ch/homepage. [Acedido em 05.11.2020].

[16] Sebenta Circuitos Eletrónicos Embebidos, Eletrónica Industrial, Instituto Superior de Engenharia de Lisboa, 2016.

[17] Jorge Manuel De Figueiredo Costa, “Automatização de sistema de separação de peças com recurso a um tapete rolante e um robot e processamento de imagem”, ISEL, 2019.

[18] Daniel Jorge Ribeiro Carvalho, “Processamento de imagem num simulador de armazenamento automático”, ISEL, 2016.

[19] Maria de Graça Vieira de Brito Almeida, “Controlo de um manipulador usando visão”, IST, 2004.

[20] João Palma, “Introdução às redes de campo de automação”, ISEL, 2004.

[21] “Bluetooth” [Online] Available: https://www.bluetooth.com/. [Acedido em 05.11.2020].

[22] Mentor User Manual, Cybernetic Applications, 1987.

[23] Dobot Magician Specifications [Online] Available: https://www.dobot.cc/dobot-magician/specification.html [Acedido em 25.11.2020].

[24] Dobot Magician User Guide, ShenZhen Yuejiang Technology, V1.7.0, 2019.01.09.

[25] Dobot Magician Communication Protocol, ShenZhen Yuejiang Technology, V1.1.5, 2019.08.05.

[26] Dobot Magician Interface Description (V2), ShenZhen Yuejiang Technology, V2, 2019.07.01.

[27] DobotStudio [Online] Available: https://www.dobot.cc/downloadcenter.html [Acedido em 25.11.2020].

[28] ESP32-WROOM-32 Datasheet, Espressif Systems, V2.9, 2019.

[29] Lady ADA, Manual Adafruit HUZZAH32 - ESP32 Feather, Adafruit lerning system, 2019.

[30] Arduino [Online] Available: https://www.arduino.cc/en/Guide/Introduction [Acedido em 25.11.2020].

[31] Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language.

[32] Datasheet Shift Register SN74HC595, Texas Instruments, 2015.

[33] Google Play, Serial Bluetooth Terminal [Online] Available: https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=pt_PT&gl=US [Acedido em 04.11.2020].

[34] Mentor Desktop Robot 35-001-USB, Feedback Intruments Limited.

[35] Vector, "CAN Protocol" [Online] Available: https://www.vector.com/int/en/know-how/technologies/networks/can/ [Acedido em 04.11.2020].

[36] Circuit Basics, "I^2C Protocol" [Online] Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ [Acedido em 04.11.2020].

Anexos

No complemento do documento escrito, da presente dissertação de mestrado apresento os seguintes anexos, que contém informações importantes sobre toda a construção e desenvolvimento do projeto final concluído.

Anexo 1. Pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather.

Anexo 2. Ligações Adafruit Huzzah32 - ESP32 Feather aos braços robóticos Mentor e Dobot Magician.

Anexo 3. Instruções para comando do braço robótico Mentor.

Anexo 4. Instruções para comando do braço robótico Dobot Magician.

Anexo 5. Dossier de projeto da IRGBR.

Anexo 1. Pinos da Plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather

Podem-se organizar os pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather em duas categorias, pinos de alimentação e pinos lógicos. Os pinos lógicos podem ainda ser divididos em quatro categorias, sendo elas, *serial pins*, *Inter-Integrated Circuit e Serial Peripheral Interface pins* (I2C e SPI), *global purpose input output pins* (*GPIO)* e *Analog pins* (AI). Estes pinos, estão esquematizados na figura 1.

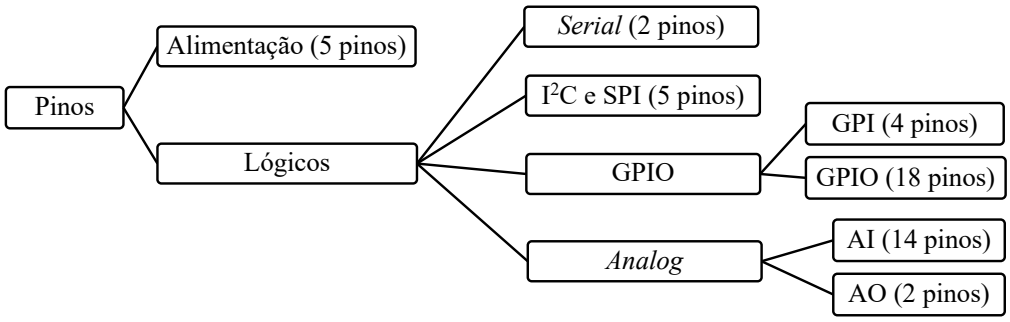


Figura 1 - Pinos do microcontrolador Adafruit Huzzah32 - ESP32 Feather.

A plataforma de desenvolvimento disponibiliza cinco pinos de alimentação, GND, BAT, USB, EN e 3V, dispostos na tabela 1.

Alimentação	Descrição dos pinos de alimentação
GND	Ponto comum para toda a potência e lógica.
BAT	Tensão positiva de/para o conector JST, para a bateria opcional.
USB	Tensão positiva de/para o conector micro USB, se ligado.
EN	Pino de habilitação do regulador de 3,3 V. Ligar ao GND para desativar o regulador de 3,3 V.
3V	Saída do regulador de 3,3V.

Tabela 1 - Pinos de Alimentação do μC Adafruit Huzzah32 - ESP32 Feather.

Os pinos lógicos são os pinos de entrada/saída de uso geral, definidos para o microcontrolador. Toda a lógica funciona a 3,3V. Na tabela 2, estão representados os pino*s* lógicos do microcontrolador para comunicação série.

Pinos Lógicos Serial	Descrição dos pinos serial
RX	O pino RX é a entrada no módulo.
TX	O pino TX é a saída do módulo.

Tabela 2 - Pinos Lógicos *Serial* do µC Adafruit Huzzah32 - ESP32 Feather.

Os pinos lógicos serial, RX e TX, são os pinos *Serial1*. Estes pinos são usados para conexão a dispositivos, transmissores recetores assíncronos universais (UART). A transmissão das sequências de bits obriga à aplicação dos sinais segundo cadências bem definidas. Só deste modo um recetor poderá reconhecer a informação presente no sinal que ele recebe. A cadência é definida por medição de tempo através de relógio (*clock*). Quando existe um sinal de relógio comum no transmissor e no recetor e os ciclos de comunicação estão sincronizados e diz-se que a comunicação é feita em modo síncrono. Se não existir um sinal permanente de sincronismo, o recetor tem que detetar quando o transmissor inicia a transmissão de uma sequência e sincronizar-se automaticamente com ela, neste caso a comunicação realiza-se em modo assíncrono.

Com a plataforma de desenvolvimento na horizontal, podemos designar uma linha de pinos superior e uma inferior. Os pinos lógicos GPIO e *Analog* da linha inferior do microcontrolador estão dispostos na tabela 3.

GPIO e *Analog*	Descrição dos pinos GPIO e analógicos da linha inferior
A0	Entrada analógica A0; ADC #2; Saída analógica DAC2; GPIO #26.
A1	Entrada analógica A1; ADC #2; Saída analógica DAC1; GPIO #25.
A2	Entrada analógica A2; ADC #1; GPI #34.
A3	Entrada analógica A3; ADC #1; GPI #39.
A4	Entrada analógica A4; ADC #1; GPI #36.
A5	Entrada analógica A5; ADC #2; GPIO #4.
21	GPIO #21.

Tabela 3 - Pinos Lógicos GPIO e Analógicos do µC Adafruit Huzzah32 - ESP32 Feather (linha inferior).

Na tabela 4, encontram-se representados os pinos lógicos *GPIO* e *Analog*, da linha superior do microcontrolador.

GPIO e *Analog*	Descrição dos pinos GPIO e analógicos da linha superior
13	Entrada analógica A12; ADC #2; GPIO #13; LED built in.
12	Entrada analógica A11; ADC #2; GPIO #12.
27	Entrada analógica A10; ADC #2; GPIO #27.
33	Entrada analógica A9; ADC #1; GPIO #33; Pode ser usado para conectar um cristal de 32 KHz.
15	Entrada analógica A8; ADC #2; GPIO #15.
32	Entrada analógica A7; ADC #1; GPIO #32; Pode ser usado para conectar um cristal de 32 KHz.
14	Entrada analógica A6; ADC #2; GPIO #14.
A13	Entrada analógica A13; Entrada de uso geral #35.

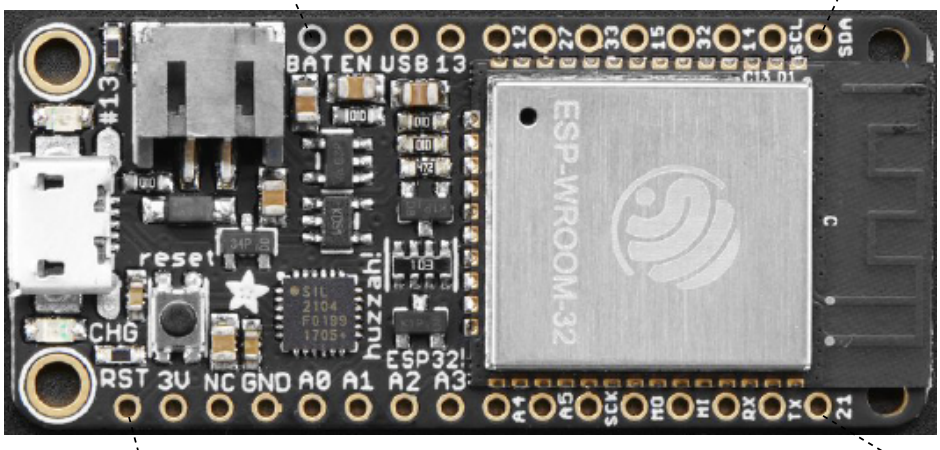
Tabela 4 - Pinos lógicos GPIO e Analógicos do µC Adafruit Huzzah32 - ESP32 Feather (linha superior).

No Anexo 1.1 encontram-se representados numa figura todos os pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather.

Anexo 1.1. Pinos da plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather

AI	A13						A12 ADC#2	A11 ADC#2	A10 ADC#2	A9 ADC#1	A8 ADC#2	A7 ADC#1	A6 ADC#2										
GPI	35																						
GPIO							13	12	27	33	15	32	14	22	23								
Back							A12	A11	A10	A9	A8	A7	A6	22	23								
Front	BAT	EN	USB	13	12	27	33	15	32	14	SCL	SDA											
Numeração Breadboard	5	6	7	8	9	10	11	12	13	14	15	16											
JP3	1	2	3	4	5	6	7	8	9	10	11	12											

(Linha Superior)



(Linha Inferior)

JP1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Numeração Breadbord	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Back / Front	RST	3V	NC	GND	A0	A1	A2	A3	A4	A5	SCK	MOSI	MISO	RX	TX	21
					DAC2	DAC1	34	39	36	4	5	18	19	16	17	
GPIO					26	25				4	5	18	19	16	17	21
GPI							34	39	36							
AI					A0 ADC#2	A1 ADC#2	A2 ADC#1	A3 ADC#1	A4 ADC#1	A5 ADC#2						
AO					DAC2 ADC#2	DAC1 ADC#2										

Legenda:
GPIO - Global Purpose Input Output
GPI - Global Purpose Input
AI - Analogic Input
AO - Analogic Output

Anexo 2. Ligações do Microcontrolador ao Mentor e Dobot Magician

Ligação Microcontrolador - Mentor

As ligações do *shift register sn74hc595* e do adaptador *flat cable* para o Mentor, feitas ao microcontrolador, estão representadas na tabela 1.

Ligações	Microcontrolador		Ligações
(V_{CC}) SR 16; $(\overline{SRCLR})$ SR 10	3V	(GPIO 13) 13	(Data 0) FC-01
(GND) SR 8; $(\overline{OE})$ SR 13	GND	(GPIO 27) 27	(Data 1) FC-03
(SER) SR 14	(GPIO 26) A0	(GPIO 33) 33	(Data 2) FC-05
$(RCLK)$ SR12	(GPIO 25) A1	(GPIO 15) 15	(Data 3) FC-07
$(SRCLK)$ SR 11	(GPIO 4) A5	(GPIO 32) 32	(Data 4) FC-09
(R/W) FC-17	(GPIO 18) MO	(GPIO 14) 14	(Data 5) FC-11
(BLK) FC-12	(GPIO 19) MI	(GPIO 22) SCL	(Data 6) FC-13
(CLK 1 MHz) FC-20	(GPIO 21) 21	(GPIO 23) SDA	(Data 7) FC-15

Tabela 1 - Ligações do Microcontrolador (Mentor).

As ligações realizadas ao adaptador de *flat cable* (FC) para o Mentor, estão representadas na tabela 2, onde as cores apresentadas, correspondem às cores associadas nas ligações dos fios usados no adaptador.

Ligações	Adaptador Flat Cable								Ligações
(GPIO 13) µC 13		Data 0	FC-01	FC-02	Adress 0				(Q_A) SR 15
(GPIO 27) µC 27		Data 1	FC-03	FC-04	Adress 1				(Q_B) SR 1
(GPIO 33) µC 33		Data 2	FC-05	FC-06	Adress 2				(Q_C) SR 2
(GPIO 15) µC 15		Data 3	FC-07	FC-08	Adress 3				(Q_D) SR 3
(GPIO 32) µC 32		Data 4	FC-09	FC-10	Adress 4				(Q_E) SR 4
(GPIO 14) µC 14		Data 5	FC-11	FC-12	BLK				(GPIO) 19 µC MI
(GPIO 22) µC SCL		Data 6	FC-13	FC-14	Adress 5				(Q_F) SR 5
(GPIO 23) µC SDA		Data 7	FC-15	FC-16	Adress 6				(Q_G) SR 6
(GPIO 18) µC MO		R/W	FC-17	FC-18	Adress 7				(Q_H) SR 7
µC GND		GND	FC-19	FC-20	CLK				(GPIO 21) µC 21

Tabela 2 - Ligações do adaptador de Flat-Cable (Mentor).

As ligações realizadas no shift register sn74hc595, são representadas na tabela 3.

Ligações	Shift Register SN74HC595		Ligações
(Address 1) FC-04	(Q_B) 1	(V_{CC}) 16	µC 3V
(Address 2) FC-06	(Q_C) 2	(Q_A) 15	(Address 0) FC-02
(Address 3) FC-08	(Q_D) 3	(SER) 14	(GPIO 26) µC A0
(Address 4) FC-10	(Q_E) 4	$\overline{(OE)}$ 13	µC GND
(Address 5) FC-14	(Q_F) 5	$(RCLK)$ 12	(GPIO 25) µC A1
(Address 6) FC-16	(Q_G) 6	$(SRCLK)$ 11	(GPIO 4) µC A5
(Address 7) FC-18	(Q_H) 7	$(\overline{SRCLR})$ 10	µC 3V
µC GND	(GND) 8	$(Q_{H'})$ 9	-

Tabela 3 - Ligações do Shift Register (Mentor).

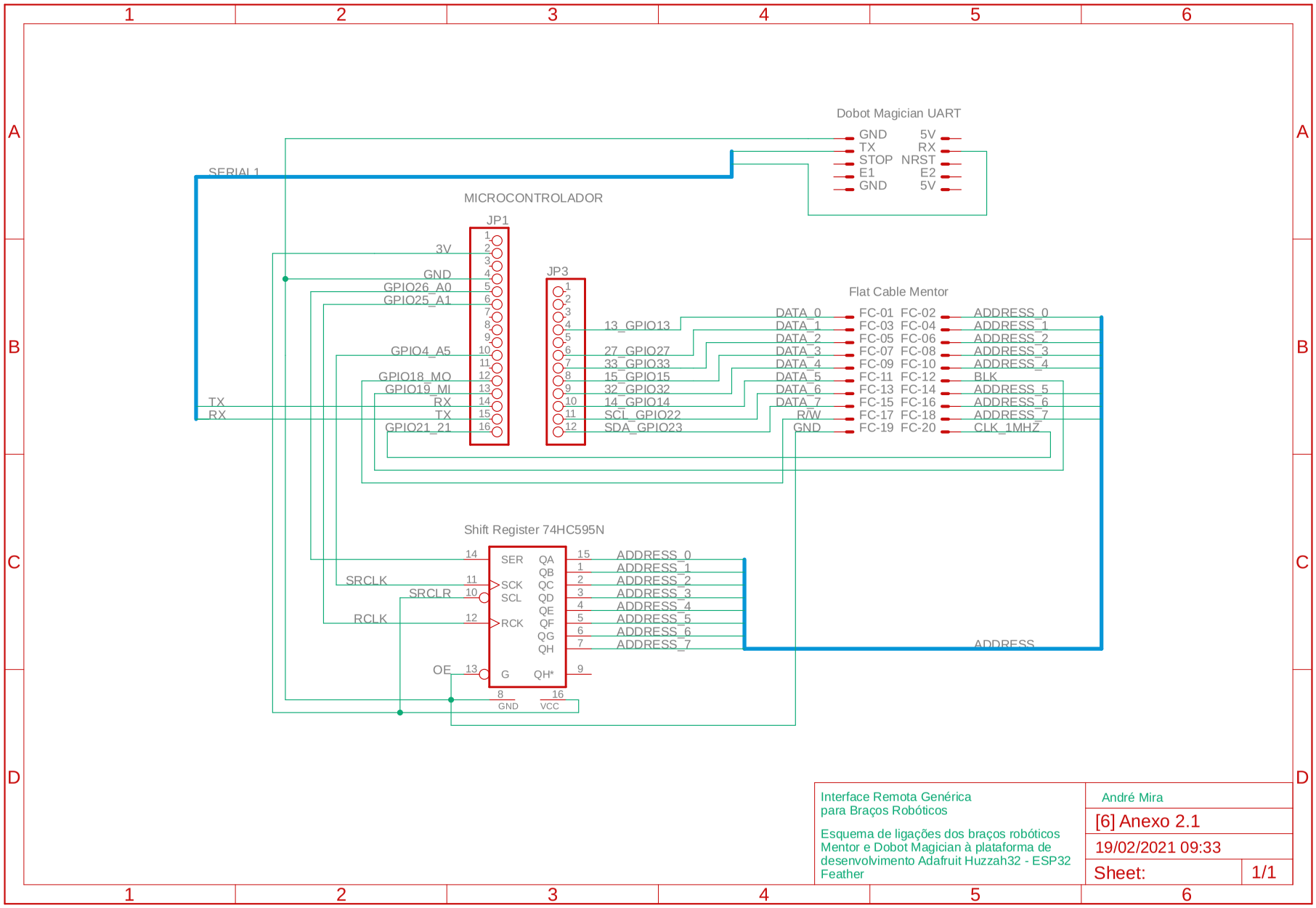
Ligação Microcontrolador - Dobot Magician

As ligações do microcontrolador ao Dobot Magician, estão representadas na tabela 4.

Ligações	Microcontrolador
Dobot UART GND	GND
Dobot UART TX	RX
Dobot UART RX	TX

Tabela 4 - Ligações do microcontrolador (Dobot Magician).

O esquema de ligações ao microcontrolador com as ligações apresentadas em todas as tabelas do presente anexo, encontra-se no Anexo 2.1.



Interface Remota Genérica para Braços Robóticos Esquema de ligações dos braços robóticos Mentor e Dobot Magician à plataforma de desenvolvimento Adafruit Huzzah32 - ESP32 Feather	André Mira [6] Anexo 2.1 19/02/2021 09:33 Sheet: 1/1

Anexo 3. Instruções para comando do braço robótico Mentor

Comandos da **COMUNICAÇÃO SÉRIE**, da Interface Remota Genérica para controlo do braço robótico **MENTOR**.

	FUNÇÃO	**COMANDO**	**VALORES**	**RETORNO**
Comandos de Movimentação	Movimentação posição Home	H	-	Robô Mentor - Posição Inicial. W:128 S:128 E:128 EV:128 WR:128 G:151
	Movimentação Waist	W xxx	0 a 250	Movimentação Waist, W:xxx.
	Movimentação Shoulder	S xxx	0 a 250	Movimentação Shoulder, S:xxx.
	Movimentação Elbow	E xxx	0 a 250	Movimentação Elbow, E:xxx.
	Movimentação Elevation wrist	EV xxx	77 a 178	Movimentação Elevação Wrist, EV:xxx.
	Movimentação Wrist Rotation	WR xxx	0 a 255	Movimentação Rotação Wrist, WR:xxx.
	Ação da garra	G xxx	151 a 196	Movimentação Garra, G:xxx.
	Comando com argumento errado. Comando sem Argumento	W, S, E, EV, WR, G		Comando X: xxx , Argumento Inválido! Repita. Comando X sem argumento, insira um novo comando.
	Leitura dos eixos	R	-	Robô Mentor - Leitura de Eixos. W:xxx S:xxx E:xxx EV:xxx WR:xxx G:xxx
	Endereço Web Server	IP	-	Com Ligação Wi-Fi Endereço IP: xxx.xxx.x.xx Sem Ligação Wi-Fi µC não está conectado! Conecte o µC.
	Comando não reconhecido	Other	-	Comando Inválido! Insira um novo comando.

Nota: O envio e a receção de comandos da comunicação série, funciona a partir do monitor série via software Arduino IDE.
Os retornos da comunicação Série podem ser lidos no monitor Série e/ou no monitor Bluetooth.
Dispositivo para conexão Bluetooth: ESP_32_BLT_RobotControl.

Comandos da **COMUNICAÇÃO BLUETOOTH**, da Interface Remota Genérica para controlo do braço robótico **MENTOR**.				
FUNÇÃO		**COMANDO**	**VALORES**	**RETORNO**
Comandos de Movimentação	Movimentação posição Home	H	-	Robô Mentor - Posição Inicial. W:128 S:128 E:128 EV:128 WR:128 G:151
	Movimentação Waist	W xxx	0 a 250	Movimentação Waist, W:xxx.
	Movimentação Shoulder	S xxx	0 a 250	Movimentação Shoulder, S:xxx.
	Movimentação Elbow	E xxx	0 a 250	Movimentação Elbow, E:xxx.
	Movimentação Elevation wrist	EV xxx	77 a 178	Movimentação Elevação Wrist, EV:xxx.
	Movimentação Wrist Rotation	WR xxx	0 a 255	Movimentação Rotação Wrist, WR:xxx.
	Ação da garra	G xxx	151 a 196	Movimentação Garra, G:xxx.
Comando com argumento errado.		W, S, E, EV, WR, G		Comando X: xxx , Argumento Inválido! Repita.
Comando sem Argumento				Comando X sem argumento, insira um novo comando.
Leitura dos eixos		R	-	Robô Mentor - Leitura de Eixos. W:xxx S:xxx E:xxx EV:xxx WR:xxx G:xxx
Endereço Web Server		IP	-	Com Ligação Wi-Fi Endereço IP: xxx.xxx.x.xx Sem Ligação Wi-Fi µC não está conectado! Conecte o µC.
Comando não reconhecido		Other	-	Comando Inválido! Insira um novo comando.
Nota: O envio e a receção de comandos da comunicação Bluetooth, funciona a partir da aplicação “Bluetooth Serial Terminal”. Os retornos na comunicação Bluetooth podem ser lidos no monitor Bluetooth e/ou no monitor Série. Dispositivo para conexão Bluetooth: ESP_32_BLT_RobotControl.				

Comandos da **COMUNICAÇÃO WI-FI (1)**, da Interface Remota Genérica para controlo do braço robótico **MENTOR**.			
FUNÇÃO	**INPUTS**	**RETORNO**	
Acesso à página de controlo	Submit button (Login)	Credenciais corretas Credenciais erradas	Log in bem-sucedido. Log in falhado.
Valor waist	Slider (Waist, 0 a 250)	-	
Valor shoulder	Slider (Shoulder, 0 a 250)	-	
Valor elbow	Slider (Elbow, 0 a 250)	-	
Valor elevation wrist	Slider (EVwrist, 77 a 178)	-	
Valor wrist rotation	Slider (WRwrist, 0 a 255)	-	
Valor da garra	Toggle button (Garra aberta ou fechada)	-	
Movimentação do Mentor	Submit button (ENVIAR!)	Movimentação Mentor: Movimentação Waist, W:xxx. Movimentação Shoulder, S:xxx. Movimentação Elbow, E:xxx. Movimentação Elevação Wrist, EV:xxx. Movimentação Rotação Wrist, WR:xxx. Movimentação Garra, G:xxx.	
Nota: A manipulação dos inputs da comunicação Wi-Fi, funcionam a partir do servidor web, acedido por um endereço IP. Credenciais para login no servidor web: ⎪Nome de Utilizador: Mentor ⎪ Palavra-Passe: Mentor ⎪. Os retornos da comunicação Wi-Fi podem ser lidos no monitor série e/ou no monitor Bluetooth.			

Comandos da **COMUNICAÇÃO WI-FI (2)**, da Interface Remota Genérica para controlo do braço robótico **MENTOR**.

FUNÇÃO	INPUTS	RETORNO
Movimentação para a posição Home.	Button (Posição Inicial!)	Robô Mentor - Posição Inicial. W:128 S:128 E:128 EV:128 WR:128 G:151
Valores da J1, J2, J3, J4, J5 e G, guardados na Posição 1.	Button (SAVE Posição 1)	Posição 1: W1:xxx S1:xxx E1:xxx EV1:xxx WR1:xxx G1:xxx
Valores da J1, J2, J3, J4, J5 e G, guardados na Posição 2.	Button (SAVE Posição 2)	Posição 2: W2:xxx S2:xxx E2:xxx EV2:xxx WR2:xxx G2:xxx
Valores da J1, J2, J3, J4, J5 e G, guardados na Posição 3.	Button (SAVE Posição 3)	Posição 3: W3:xxx S3:xxx E3:xxx EV3:xxx WR3:xxx G3:xxx
Valores da J1, J2, J3, J4, J5 e G, guardados na Posição 4.	Button (SAVE Posição 4)	Posição 4: W4:xxx S4:xxx E4:xxx EV4:xxx WR4:xxx G4:xxx
Movimentação do Mentor para as Posições 1, 2, 3 e 4.	Button (RUN Posições)	Robô Mentor - A correr Posições. W1:xxx S1:xxx E1:xxx EV1:xxx WR1:xxx G1:xxx W2:xxx S2:xxx E2:xxx EV2:xxx WR2:xxx G2:xxx W3:xxx S3:xxx E3:xxx EV3:xxx WR3:xxx G3:xxx W4:xxx S4:xxx E4:xxx EV4:xxx WR4:xxx G4:xxx
Reset dos valores das Posições 1, 2, 3 e 4.	Button (RESET Posições)	-

Nota: A manipulação dos inputs da comunicação Wi-Fi, funcionam a partir do servidor web, acedido por um endereço IP.

Anexo 4. Instruções para comando do braço robótico Dobot Magician

Comandos da **COMUNICAÇÃO SÉRIE**, da Interface Remota Genérica para controlo do braço robótico **DOBOT MAGICIAN**.

FUNÇÃO	COMANDO	VALORES	RETORNO
Movimentação posição Home.	H	-	Robô Dobot - Posição Inicial. X:260.5 Y:0 Z:-8.5 R:0
Movimentação no plano xyz.	P x y z r	x: 160 a 250 y: -150 a 150 z: -50 a 75 r: -50 a 50	Comando correto Robô Dobot P xxx xxx xxx xxx Comando com argumentos inválidos Robô Dobot (deteção argumentos inválidos) Sem 4 argumentos válidos, repita o comando. Sem 4 argumentos Comando P sem 4 argumentos! Repita.
Leitura das coordenadas x, y, z e r.	R	-	Robô Dobot - Leitura Coordenadas. X:xxx Y:xxx Z:xxx R:xxx
Leitura código de alarme	A	-	Robô Dobot - Leitura Alarme. Alarme: xxxx
Endereço Web Server	IP	-	Com Ligação Wi-Fi Endereço IP: xxx.xxx.x.xx Sem ligação Wi-Fi µC não está conectado! Conecte o µC.
Comando não reconhecido.	Other	-	Comando Inválido! Insira um novo comando.

Nota: O envio e a receção de comandos da comunicação série, funciona a partir do monitor série via software Arduino IDE.
Os retornos da comunicação Série podem ser lidos no monitor Série e/ou no monitor Bluetooth.
Dispositivo: ESP_32_BLT_RobotControl.

Comandos da **COMUNICAÇÃO BLUETOOTH**, da Interface Remota Genérica para controlo do braço robótico **DOBOT MAGICIAN**.				
FUNÇÃO	**COMANDO**	**VALORES**	**RETORNO**	
Movimentação posição Home.	H	-	Robô Dobot - Posição Inicial. X:260.5 Y:0 Z:-8.5 R:0	
Movimentação no plano xyz.	P x y z r	x: 160 a 250 y: -150 a 150 z: -50 a 75 r: -50 a 50	Comando correto Comando com argumentos inválidos Sem 4 argumentos	Robô Dobot P xxx xxx xxx xxx Robô Dobot (deteção argumentos inválidos) Sem 4 argumentos válidos, repita o comando. Comando P sem 4 argumentos! Repita.
Leitura das coordenadas x, y, z e r.	R	-	Robô Dobot - Leitura Coordenadas. X:xxx Y:xxx Z:xxx R:xxx	
Leitura código de alarme	A	-	Robô Dobot - Leitura Alarme. Alarme: xxxx	
Endereço Web Server	IP	-	Com Ligação Wi-Fi Sem ligação Wi-Fi	Endereço IP: xxx.xxx.x.xx µC não está conectado! Conecte o µC.
Comando não reconhecido.	Other	-	Comando Inválido! Insira um novo comando.	
Nota: O envio e a receção de comandos da comunicação Bluetooth, funciona a partir da aplicação “Serial Bluetooth Terminal”. Os retornos da comunicação Bluetooth podem ser lidos no monitor Bluetooth e/ou no monitor série. Dispositivo: ESP_32_BLT_RobotControl				

Comandos da **COMUNICAÇÃO WI-FI (1)**, da Interface Remota Genérica para controlo do braço robótico **DOBOT MAGICIAN**.

FUNÇÃO	**INPUTS**	**RETORNO**	
Acesso à página de controlo	Submit button (Login)	Credenciais corretas	Log in bem-sucedido.
		Credenciais erradas	Log in falhado.
Valor da coordenada X	Slider (x: 160 a 250)	-	
Valor da coordenada Y	Slider (y: -150 a 150)	-	
Valor da coordenada Z	Slider (z: -50 a 75)	-	
Valor de R	Slider (r: -50 a 50)	-	
Movimentação do Dobot	Submit button (ENVIAR!)	Movimentação Robô Dobot P xxx xxx xxx xxx	

Nota: A manipulação dos inputs da comunicação Wi-Fi, funcionam a partir do servidor web, acedido por um endereço IP.
Credenciais para login no servidor web: │Nome de Utilizador: Dobot │ Palavra-Passe: Dobot │.
Os retornos da comunicação Wi-Fi podem ser lidos no monitor série e/ou no monitor Bluetooth.

Comandos da **COMUNICAÇÃO WI-FI (2)**, da Interface Remota Genérica para controlo do braço robótico **DOBOT MAGICIAN**.		
FUNÇÃO	**INPUTS**	**RETORNO**
Movimentação para a posição Home.	Button (Posição Inicial!)	Robô Dobot - Posição Inicial. X:260,5 Y:0 Z:-8,5 R:0
Valores de x, y, z, r, guardados na Posição 1.	Button (SAVE Posição 1)	Posição 1. X1:xxx Y1:xxx Z1:xxx R1:xxx
Valores de x, y, z, r, guardados na Posição 2.	Button (SAVE Posição 2)	Posição 2. X2:xxx Y2:xxx Z2:xxx R2:xxx
Valores de x, y, z, r, guardados na Posição 3.	Button (SAVE Posição 3)	Posição 3. X3:xxx Y3:xxx Z3:xxx R3:xxx
Valores de x, y, z, r, guardados na Posição 4.	Button (SAVE Posição 4)	Posição 4: X4:xxx Y4:xxx Z4:xxx R4:xxx
Movimentação do Dobot para as Posições 1, 2, 3 e 4.	Button (RUN Posições)	Robô Dobot - A correr posições. X1:xxx Y1:xxx Z1:xxx R1:xxx X2:xxx Y2:xxx Z2:xxx R2:xxx X3:xxx Y3:xxx Z3:xxx R3:xxx X4:xxx Y4:xxx Z4:xxx R4:xxx
Reset dos valores das Posições 1, 2, 3 e 4.	Button (RESET Posições)	-
Nota: A manipulação dos inputs da comunicação Wi-Fi, funcionam a partir do servidor web, acedido por um endereço IP.		

Anexo 5. Dossier de projeto da IRGBR

Constituição do código desenvolvido para a IRGBR

A constituição do código da Interface Remota Genérica para Braços Robóticos (IRGBR), está descrita na tabela 1.

Separador	(A) Mentor	(B) Dobot Magician	
Principal	InterfaceM	InterfaceD	①
Biblioteca	CommandHandler.cpp	CommandHandler.cpp	②
Biblioteca	CommandHandler.h	CommandHandler.h	③
Comunicação Bluetooth	ComunicacaoBLT	ComunicacaoBLT	④
Comunicação Série	ComunicaçaoSerie	ComunicaçaoSerie	⑤
Comunicação Wi-Fi	ComunicacaoWIFI	ComunicacaoWIFI	⑥
Programação Braço Robótico	Mentor	Dobot	⑦
Servidor Web Braço Robótico	htmlsM.h	htmlsD.h	⑧

Tabela 1 - Constituição do Código da Interface Remota Genérica para Braços Robóticos.

No separador principal do código desenvolvido ①, pode-se desativar e ativar facilmente as comunicações série, Bluetooth e Wi-Fi. Na comunicação Wi-Fi pode-se alternar entre várias ligações que se queiram guardar, como por exemplo uma ligação em casa ou uma ligação na rede do ISEL. O commandHandler ②③, é a biblioteca usada para a manipulação dos comandos para controlo do braço robótico através do microcontrolador.

Os separadores ④⑤⑥ dizem respeito às comunicações série, Bluetooth e Wi-Fi desenvolvidas. No separador ⑧ está a construção do servidor web desenvolvido, para o controlo do braço robótico adotado. No separador ⑦, entra o protocolo de comando do braço robótico adotado.

A - IRGBR: InterfaceMentor

O código desenvolvido para o Mentor é composto por 8 separadores.

① InterfaceM

```
#include <Arduino.h>          // Comunicação AP
#include <WiFi.h>             // Comunicação Wi-Fi
#include <WiFiClient.h>       // Comunicação Wi-Fi
#include <WebServer.h>        // Comunicação Wi-Fi
#include "htmls.h"            // Comunicação Wi-Fi
#include <esp_wpa2.h>         // Comunicação Wi-Fi
WebServer server(80);         // Comunicação Wi-Fi
```

```
#include "CommandHandler.h"           // Comunicação Série
CommandHandler cmdHdl(" ", '\n');     // Comunicação Série (default are: delim="," and term=';')
#include <ShiftRegister74HC595.h>     // Shift Register
ShiftRegister74HC595<1> sr(26, 4, 25);// SR <nº shift registers> (data pin, clock pin, latch pin)

#include "BluetoothSerial.h"   // Comunicação Bluetooth
BluetoothSerial ESP_BT;        // Comunicação Bluetooth


// [Partition Scheme > No OTA (Large APP)]
void setup(void) {
  Serial.begin(9600);          // Monitor Série 9600
  Serial.println("\n\n_____________________________________________________________________________");
  Serial.println("[ADAFRUIT HUZZAH32 - ESP32 FEATHER] André Mira [Início -> Botão Reset!]\n");
  inicializarPins();           // Mentor  Inicialização Pins
  PosicaoInicial();            // Posição Inicial do Mentor
  startSERIE();                // Comunicação Série
  startBLT();                  // Comunicação Bluetooth
  startWIFIcasa();             // Comunicação Wi-Fi Casa
  //startWIFIisel();           // Comunicação Wi-Fi Isel
  //startAP();                 // Comunicação AP
  startWebServer();            // Comunicação Wi-Fi WebServer
  Serial.println("_____________________________________________________________________________\n");
  Serial.println("Comunicação Série (Mentor)");
  Serial.println("Comandos disponíveis: H, W, S, E, EV, WR, G, R, IP.\n\n");
}

void loop(void) {
  cmdHdl.processSerial(Serial);     // Comunicação Série
  blueTooth();                      // Comunicação Bluetooth
  server.handleClient();            // Comunicação Wi-Fi
}
```

② CommandHandler.cpp

```
/**
 * CommandHandler - A Wiring/Arduino library to tokenize and parse commands
 * received in different forms, serial, string, or char.
 *
 * Copyright (C) 2015 Cronin Group http://www.chem.gla.ac.uk/cronin/
 * Copyright (C) 2012 Stefan Rado
 * Copyright (C) 2011 Steven Cogswell <steven.cogswell@gmail.com>
 *                    http://husks.wordpress.com
 *
 * Version 20151029
 *
 * This library is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this library.  If not, see <http://www.gnu.org/licenses/>.
 */

#include "CommandHandler.h"

/**
 * Constructor allowing to change default delim and term
 * Example: SerialCommand sCmd(" ", ';');
 * Default are COMMANDHANDLER_DEFAULT_DELIM and COMMANDHANDLER_DEFAULT_TERM
 */
CommandHandler::CommandHandler(const char *newdelim, char newterm)
  : commandList(NULL),
    commandCount(0),
    relayList(NULL),
    relayCount(0),
    defaultHandler(NULL),
    pt2defaultHandlerObject(NULL),
```

```
    wrapper_defaultHandler(NULL),
    term(newterm),          // asssign new terminator for commands
    last(NULL),
    delim(newdelim) // assign new delimitor
{
  inCmdStream = &Serial;
  outCmdStream = &Serial;

  commandHeader = String("");
  commandDecimal = 2;

  clearBuffer();
}

/**
 * Adds a "command" and a handler function to the list of available commands.
 * This is used for matching a found token in the buffer, and gives the pointer
 * to the handler function to deal with it.
 */
void CommandHandler::addCommand(const char *command, void (*function)()) {
  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Adding command (");
    Serial.print(commandCount);
    Serial.print("): ");
    Serial.println(command);
  #endif

  commandList = (CommandHandlerCallback *) realloc(commandList, (commandCount + 1) * sizeof(CommandHandlerCallback));
  strncpy(commandList[commandCount].command, command, COMMANDHANDLER_MAXCOMMANDLENGTH);
  commandList[commandCount].function = function;
  commandCount++;
}

/**
 * Adds a "command" and a handler function to the list of available relay.
 * This is used for matching a found token in the buffer, and gives the pointer
 * to the handler function to deal with the remaining of the command
 */
void CommandHandler::addRelay(const char *command, void (*function)(const char *, void*), void* pt2Object) {
  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Adding relay (");
    Serial.print(relayCount);
    Serial.print("): ");
    Serial.println(command);
  #endif

  relayList = (RelayHandlerCallback *) realloc(relayList, (relayCount + 1) * sizeof(RelayHandlerCallback));
  strncpy(relayList[relayCount].command, command, COMMANDHANDLER_MAXCOMMANDLENGTH);
  relayList[relayCount].pt2Object = pt2Object;
  relayList[relayCount].function = function;
  relayCount++;
}

/**
 * This sets up a handler to be called in the event that the receveived command string
 * isn't in the list of commands.
 */
void CommandHandler::setDefaultHandler(void (*function)(const char *)) {
  defaultHandler = function;
}

void CommandHandler::setDefaultHandler(void (*function)(const char *, void*), void* pt2Object) {
  pt2defaultHandlerObject = pt2Object;
  wrapper_defaultHandler = function;
}

/**
 * Assign the default serial
 */
void CommandHandler::setInCmdSerial(Stream &inStream) {
  inCmdStream = &inStream;
}

/**
 * Check the default Serial
 */
```

```
void CommandHandler::processSerial() {
  processSerial(*inCmdStream);
}

/**
 * This checks the Serial stream for characters, and assembles them into a buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processSerial(Stream &inStream) {
  while (inStream.available() > 0) {
    char inChar = inStream.read();   // Read single available character, there may be more waiting
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("Serial: ");
      Serial.println(inChar);   // Echo back to serial stream
    #endif
    processChar(inChar);
  }
}

/**
 * This iterate on a String char by char, and push them into a buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processString(const char *inString) {
  for (int i = 0; i < strlen(inString); i++){
    char inChar = inString[i];
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("String: ");
      Serial.println(inChar);   // Echo back to serial stream
    #endif
    processChar(inChar);
  }
}

/**
 * This add a characters to the buffer, and analyse the buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processChar(char inChar) {
  if (inChar == term) {     // Check for the terminator (default '\r') meaning end of command
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("Received: ");
      Serial.println(buffer);
    #endif

    char *command = strtok_r(buffer, delim, &last);   // Search for command at start of buffer
    if (command != NULL) {
      boolean matched = false;
      // searching in commands
      for (int i = 0; i < commandCount; i++) {
        #ifdef COMMANDHANDLER_DEBUG
          Serial.print("Comparing [");
          Serial.print(command);
          Serial.print("] to [");
          Serial.print(commandList[i].command);
          Serial.println("]");
        #endif

        // Compare the found command against the list of known commands for a match
        if (strncmp(command, commandList[i].command, COMMANDHANDLER_MAXCOMMANDLENGTH) == 0) {
          #ifdef COMMANDHANDLER_DEBUG
            Serial.print("Matched Command: ");
            Serial.println(command);
          #endif

          // Execute the stored handler function for the command
          (*commandList[i].function)();
          matched = true;
          break;
        }
      }
      // searching in relays
      for (int i = 0; i < relayCount; i++) {
```

```
      #ifdef COMMANDHANDLER_DEBUG
        Serial.print("Comparing [");
        Serial.print(command);
        Serial.print("] to [");
        Serial.print(relayList[i].command);
        Serial.println("]");
      #endif

      // Compare the found command against the relay list of known commands for a match
      if (strncmp(command, relayList[i].command, COMMANDHANDLER_MAXCOMMANDLENGTH) == 0) {
        #ifdef COMMANDHANDLER_DEBUG
          Serial.print("Matched Relay: ");
          Serial.println(command);
        #endif

        // Execute the stored handler function for the command
        (*relayList[i].function)(remaining(), relayList[i].pt2Object);
        matched = true;
        break;
      }
     }
     if (!matched){
      if (defaultHandler != NULL) {
       (*defaultHandler)(command);
      } else if (pt2defaultHandlerObject != NULL) {
       (*wrapper_defaultHandler)(command, pt2defaultHandlerObject);
      }
     }
    }
    clearBuffer();
  }
  else if (isprint(inChar)) {     // Only printable characters into the buffer
   if (bufPos < COMMANDHANDLER_BUFFER) {
    buffer[bufPos] = inChar;  // Put character into buffer
    buffer[bufPos+1] = STRING_NULL_TERM;      // Null terminate
    bufPos++;
   } else {
    #ifdef COMMANDHANDLER_DEBUG
     Serial.println("Line buffer is full - increase COMMANDHANDLER_BUFFER");
    #endif
   }
  }
}

/*
 * Clear the input buffer.
 */
void CommandHandler::clearBuffer() {
  buffer[0] = STRING_NULL_TERM;
  bufPos = 0;
}

/**
 * Retrieve the next token ("word" or "argument") from the command buffer.
 * Returns NULL if no more tokens exist.
 */
char *CommandHandler::next() {
  return strtok_r(NULL, delim, &last);
}

/**
 * Returns char* of the remaining of the command buffer (for getting arguments to commands).
 * Returns NULL if no more tokens exist.
 */
char *CommandHandler::remaining() {

  //reinit the remains char
  remains[0] = STRING_NULL_TERM;

  char str_term[2];
  str_term[0] = term;
  str_term[1] = STRING_NULL_TERM;

   // Search for the remaining up to next term
  char *command = strtok_r(NULL, str_term, &last);
```

```
  // forge term in string format
  strcpy(remains, command);
  strcat(remains, str_term);

  // clear the buffer now, we emptied the current command
  // the remaining is might be given to another handler
  // or it might used by the same commandHandler instance
  // hence the buffer should be emptied now
  clearBuffer();

  return remains;
}



/****************************************
 * Helpers to read args and cast them into specific type, strongly inspired by CmdMessenger
 ****************************************/

/**
 * Read the next argument as int16
 */
int CommandHandler::readIntArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return atoi(arg);
  }
  argOk = false;
  return 0;
}

/**
 * Read the next argument as int32
 */
long CommandHandler::readLongArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return atol(arg);
  }
  argOk = false;
  return 0L; // 'L' to force the constant into a long data format
}

/**
 * Read the next argument as bool
 */
bool CommandHandler::readBoolArg() {
  return (readIntArg() != 0) ? true : false;
}

/**
 * Read the next argument as float
 */
float CommandHandler::readFloatArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return strtod(arg, NULL);
  }
  argOk = false;
  return 0;
}

/**
 * Read the next argument as double
 */
double CommandHandler::readDoubleArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
```

```
    return strtod(arg, NULL);
  }
  argOk = false;
  return 0;
}

/**
 * Read next argument as string.
 */
char* CommandHandler::readStringArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return arg;
  }
  argOk = false;
  return STRING_NULL_TERM;
}

/**
 * Compare the next argument with a string
 */
bool CommandHandler::compareStringArg(const char *stringToCompare) {
  char *arg;
  arg = next();
  if (arg != NULL) {
    return (strcmp(stringToCompare, arg) == 0) ? true : false;
  }
  return false;
}

/****************************************
 * Forging and sending output commands
 ****************************************/


/**
 * Set an header for the output command
 */
void CommandHandler::setCmdHeader(const char *cmdHeader, bool addDelim) {

  commandHeader = String(cmdHeader);

  if (addDelim == true) {
    commandHeader = commandHeader + String(delim);
  }

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out Command Header is now ");
    Serial.println(commandHeader);
  #endif
}

void CommandHandler::initCmd() {

  commandString = commandHeader;

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdDelim() {

  commandString = commandString + String(delim);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}
//
void CommandHandler::addCmdTerm() {
```

```
  commandString = commandString + String(term);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdBool(bool value) {

  commandString = commandString + String(value);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdInt(int value) {

  commandString = commandString + String(value, DEC);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdLong(long value) {

  commandString = commandString + String(value, DEC);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}


void CommandHandler::setCmdDecimal(byte decimal) {
  commandDecimal = decimal;
}

void CommandHandler::addCmdFloat(double value) {
  addCmdFloat(value, commandDecimal);
}

void CommandHandler::addCmdFloat(float value, byte decimal) {


  commandString = commandString + String(value, decimal);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdDouble(double value) {
  addCmdDouble(value, commandDecimal);
}

void CommandHandler::addCmdDouble(double value, byte decimal) {

  commandString = commandString + String(value, decimal);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdString(const char *value) {

  commandString = commandString + String(value);
```

```
  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif

}

char* CommandHandler::getOutCmd() {

  commandString.toCharArray(command, COMMANDHANDLER_BUFFER + 1);

  return command;
}

void CommandHandler::setOutCmdSerial(Stream &outStream) {
  outCmdStream = &outStream;
}

void CommandHandler::sendCmdSerial() {
  sendCmdSerial(*outCmdStream);
}

void CommandHandler::sendCmdSerial(Stream &outStream) {
  outStream.print(commandString);
}
```

③ CommandHandler.h

```
/**
 * CommandHandler - A Wiring/Arduino library to tokenize and parse commands
 * received in different forms, serial, string, or char.
 *
 * Copyright (C) 2015 Cronin Group http://www.chem.gla.ac.uk/cronin/
 * Copyright (C) 2012 Stefan Rado
 * Copyright (C) 2011 Steven Cogswell <steven.cogswell@gmail.com>
 *                    http://husks.wordpress.com
 *
 * Version 20151029
 *
 * This library is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this library.  If not, see <http://www.gnu.org/licenses/>.
 */

#ifndef CommandHandler_h
#define CommandHandler_h

#if defined(WIRING) && WIRING >= 100
  #include <Wiring.h>
#elif defined(ARDUINO) && ARDUINO >= 100
  #include <Arduino.h>
#else
  #include <WProgram.h>
#endif
#include <string.h>

// Size of the input buffer in bytes (maximum length of one command plus arguments)
#define COMMANDHANDLER_BUFFER 64
// Maximum length of a command excluding the terminating null
#define COMMANDHANDLER_MAXCOMMANDLENGTH 8
// Default delimitor and terminator
#define COMMANDHANDLER_DEFAULT_DELIM ","
#define COMMANDHANDLER_DEFAULT_TERM ';'
// The null term for string
#define STRING_NULL_TERM '\0'
```

```
// Uncomment the next line to run the library in debug mode (verbose messages)
// #define COMMANDHANDLER_DEBUG


class CommandHandler {
  public:
    CommandHandler(const char *newdelim = COMMANDHANDLER_DEFAULT_DELIM, const char newterm = COMMANDHANDLER_DEFAULT_TERM);   // Constructor
    void addCommand(const char *command, void(*function)());  // Add a command to the processing dictionary.
    void addRelay(const char *command, void (*function)(const char *, void*), void* pt2Object = NULL);  // Add a command to the relay dictionary. Such relay are given the remaining of the command. pt2Object is the reference to the instance associated with the callback, it will be given as the second argument of the callback function, default is NULL
    void setDefaultHandler(void (*function)(const char *));   // A handler to call when no valid command received.
    void setDefaultHandler(void (*function)(const char *, void*), void* pt2Object);    // A handler to call when no valid command received.

    void setInCmdSerial(Stream &inStream); // define to which serial to send the read commands
    void processSerial();  // Process what on the in stream
    void processSerial(Stream &inStream);  // Process what on the designated stream
    void processString(const char *inString); // Process a String
    void processChar(char inChar); //Process a char
    void clearBuffer();   // Clears the input buffer.
    char *remaining();        // Returns pointer to remaining of the command buffer (for getting arguments to commands).
    char *next();         // Returns pointer to next token found in command buffer (for getting arguments to commands).

    // helpers to cast next into different types
    bool argOk; // this variable is set after the below function are run, it tell you if thing went well
    bool readBoolArg();
    int readIntArg();
    long readLongArg();
    float readFloatArg();
    double readDoubleArg();
    char *readStringArg();
    bool compareStringArg(const char *stringToCompare);

    //helpers to create a message
    void setCmdHeader(const char *cmdHeader, bool addDelim = true); // setting a char to be added at the start of each out message (default "")
    void initCmd(); // initialize the command buffer  to build next message to be sent

    void clearCmd(); // clear the output command
    void addCmdDelim();
    void addCmdTerm();

    void addCmdBool(bool value);
    void addCmdInt(int value);
    void addCmdLong(long value);

    void setCmdDecimal(byte decimal);
    void addCmdFloat(double value);
    void addCmdFloat(float value, byte decimal);
    void addCmdDouble(double value);
    void addCmdDouble(double value, byte decimal);

    void addCmdString(const char *value);

    char* getOutCmd(); // get pointer to command buffer

    void setOutCmdSerial(Stream &outStream); // define to which serial to send the out commands
    void sendCmdSerial(); //send current command thought the Stream
    void sendCmdSerial(Stream &outStream); //send current command thought the Stream

  private:

    // Command/handler dictionary
    struct CommandHandlerCallback {
      char command[COMMANDHANDLER_MAXCOMMANDLENGTH + 1];
      void (*function)();
    };                              // Data structure to hold Command/Handler function key-value pairs
    CommandHandlerCallback *commandList;   // Actual definition for command/handler array
    byte commandCount;

    // Relay/handler dictionary
    struct RelayHandlerCallback {
      char command[COMMANDHANDLER_MAXCOMMANDLENGTH + 1];
```

```
    void* pt2Object;
    void (*function)(const char *, void*);
  };                              // Data structure to hold Relay/Handler function key-value pairs
  RelayHandlerCallback *relayList;   // Actual definition for Relay/handler array
  byte relayCount;

  // Pointer to the default handler function
  void (*defaultHandler)(const char *);
  void* pt2defaultHandlerObject;
  void (*wrapper_defaultHandler)(const char *, void*);

  const char *delim; // null-terminated list of character to be used as delimeters for tokenizing (default " ")
  char term;     // Character that signals end of command (default '\n')

  char buffer[COMMANDHANDLER_BUFFER + 1]; // Buffer of stored characters while waiting for terminator character
  byte bufPos;                        // Current position in the buffer
  char *last;                         // State variable used by strtok_r during processing

  char remains[COMMANDHANDLER_BUFFER + 1]; // Buffer of stored characters to pass to a relay function

  char command[COMMANDHANDLER_BUFFER + 1];
  String commandString; // Out Command
  String commandHeader; // header for out command
  byte commandDecimal;


  // in and out default strem
  Stream *inCmdStream;
  Stream *outCmdStream;
};

#endif //CommandHandler_h
```

④ ComunicacaoBLT

```
char incoming;
byte startedText = 1;
int txtIdx = 0;
const int BUFFER_SIZE = 80;
char rcvBuffer[BUFFER_SIZE];
void startBLT()
{
 ESP_BT.begin("ESP_32_BLT_RobotControl");
}
void blueTooth(){
 if (ESP_BT.available())
 {
  incoming = ESP_BT.read();
  switch (incoming)
  {
   case '\n':
    if (startedText)
    {
     startedText = 1;
     rcvBuffer[txtIdx] = '\n';
     for (int i = txtIdx + 1; i < BUFFER_SIZE; i++) {
      rcvBuffer[i] = '\0';
     }
     Serial.print(" BT RCV ->");
     Serial.print(rcvBuffer);
     cmdHdl.processString(rcvBuffer);
     txtIdx = 0;
    }
    else
    {
     startedText = 1;
     txtIdx = 0;
    }
    break;
   default:
    if (startedText)
    {
     rcvBuffer[txtIdx++] = incoming;
    }
```

```
      if (txtIdx >= BUFFER_SIZE)
      {
       Serial.println("Buffer overflow");
       txtIdx = 0;
      }
      break;
    }
  }
}
```

⑤ ComunicaçaoSerie

```
void startSERIE()
{
 cmdHdl.addCommand("W", W);     // Movimentar W  Waist
 cmdHdl.addCommand("S", S);     // Movimentar S  Shoulder
 cmdHdl.addCommand("E", E);     // Movimentar E  Elbow
 cmdHdl.addCommand("EV", EV);   // Movimentar EV Elevação Wrist
 cmdHdl.addCommand("WR", WR);   // Movimentar WR Rotação Wrist
 cmdHdl.addCommand("G", G);     // Abrir/Fechar  Garra

 cmdHdl.addCommand("R", RD);    // Ler posições

 cmdHdl.addCommand("H", H);     // Retornar á posição inicial

 cmdHdl.addCommand("IP", avisoWiFiBLT);       // Escrever IP por bluetooth

 cmdHdl.setDefaultHandler(unrecognized);      // Handler for command that isn't matched
}

void unrecognized(const char *command)
{
 Serial.println("Comando Inválido! Insira um novo comando.\n");
 ESP_BT.println("Comando Inválido! Insira um novo comando.\n");    // escrever na app Serial Bluetooth
}
```

⑥ ComunicacaoWIFI

```
#define EAP_IDENTITY "****Email_Aluno_ISEL****"
#define EAP_PASSWORD "********PassWord********"

String w;     String s;     String e;      String ev;      String wr;      String g;
String w1;    String s1;    String e1;     String ev1;     String wr1;     String g1;
String w2;    String s2;    String e2;     String ev2;     String wr2;     String g2;
String w3;    String s3;    String e3;     String ev3;     String wr3;     String g3;
String w4;    String s4;    String e4;     String ev4;     String wr4;     String g4;

void startWIFIcasa()
{
 const char* ssid = "********RedeCasa****";
 const char* password = "****PassWord****";

 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);

 Serial.print("Connect to "); Serial.println(ssid);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 } Serial.println("");

 Serial.print("IP address: ");  Serial.println(WiFi.localIP());
}

void startWIFIisel()
{
 const char* ssid =  "eduroam";

 Serial.println();
 Serial.printf("A estabelecer ligação com a rede: %s", ssid);
 WiFi.disconnect( true );
 WiFi.mode( WIFI_STA );
 // Provide identity
```

```
  esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
  // Provide username (identity and username are the same)
  esp_wifi_sta_wpa2_ent_set_username((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
  // Provide password
  esp_wifi_sta_wpa2_ent_set_password((uint8_t *)EAP_PASSWORD, strlen(EAP_PASSWORD));
  // Set config settings to default
  esp_wpa2_config_t config = WPA2_CONFIG_INIT_DEFAULT();
  // Set config settings to enable function
  esp_wifi_sta_wpa2_ent_enable(&config);
  WiFi.begin(ssid );
  Serial.println();
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".. ");
  }
  Serial.println();
  Serial.printf("Ligação estabelecida com sucesso com a rede: %s", ssid);
  Serial.println();
  Serial.print  ("IP Address:                         ");
  Serial.println(WiFi.localIP());
}

void startAP()
{
  const char* ssid     = "ESP32-Access-Point";
  const char* password = "Mentor";

  Serial.print("Setting AP (Access Point)…");
  // Remove the password parameter, if you want the AP (Access Point) to be open
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP(); Serial.print("AP IP address: "); Serial.println(IP);
  // Print ESP Local IP Address
  Serial.print("WiFi Local IP:"); Serial.println(WiFi.localIP());
}

void avisoWiFiBLT()
{
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.println("µC não está conectado! Conecte o µC.\n");
    ESP_BT.println("µC não está conectado! Conecte o µC.\n");    // escrever na app Serial Bluetooth
  }
  else
  {
    Serial.print("Endereço IP: "); Serial.print(WiFi.localIP()); Serial.println("\n");
    ESP_BT.print("Endereço IP: "); ESP_BT.print(WiFi.localIP()); ESP_BT.println("\n"); // escrever na app Serial Bluetooth o ip
  }
}

void startWebServer()
{
  server.on("/", handleRoot);
  server.on("/login", handleLogin);
  server.on("/sobre", []() {
    String s = P2M_Sobre;
    server.send(200, "text/html", s);
  });
  server.on("/instrucoesM", []() {
    String s = P4M_InstrucoesM;
    server.send(200, "text/html", s);
  });
  server.on("/sobreM", []() {
    String s = P5M_SobreM;
    server.send(200, "text/html", s);
  });

  //here the list of headers to be recorded
  const char * headerkeys[] = {"User-Agent", "Cookie"} ;
  size_t headerkeyssize = sizeof(headerkeys) / sizeof(char*);

  //ask server to track these headers
  server.collectHeaders(headerkeys, headerkeyssize);

  server.begin();
  //Serial.println("HTTP server started");
```

```
}

bool is_authentified()
{
  //Serial.println("Enter is_authentified");
  if (server.hasHeader("Cookie"))
  {
    //Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    //Serial.println(cookie);
    if (cookie.indexOf("ESPSESSIONID=1") != -1) {
      //Serial.println("Authentification Successful");Serial.println("");
      return true;
    }
  }
  //Serial.println("Authentification Failed"); Serial.println("");
  return false;
}

//login page, also called for disconnect
void handleLogin()
{
  String msg;
  if (server.hasHeader("Cookie")) {
    //Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    //Serial.println(cookie);
  }
  if (server.hasArg("DISCONNECT"))
  {
    //Serial.println("Disconnection");
    server.sendHeader("Location", "/login");
    server.sendHeader("Cache-Control", "no-cache");
    server.sendHeader("Set-Cookie", "ESPSESSIONID=0");
    server.send(301);
    return;
  }
  if (server.hasArg("USERNAME") && server.hasArg("PASSWORD"))
  {
    if (server.arg("USERNAME") == "Mentor" &&  server.arg("PASSWORD") == "Mentor")
    {
      server.sendHeader("Location", "/");
      server.sendHeader("Cache-Control", "no-cache");
      server.sendHeader("Set-Cookie", "ESPSESSIONID=1");
      server.send(301);
      Serial.println("Log in bem-sucedido.\n");
      ESP_BT.println("Log in bem-sucedido.\n");
      return;
    }
    msg = "Wrong username/password! try again.";
    Serial.println("Log in falhado.\n");
    ESP_BT.println("Log in falhado.\n");
  }
  String s = P1M_Entrada;              // Read HTML contents
  server.send(200, "text/html", s); // Send web page
}

//root page can be accessed only if authentification is ok
void handleRoot()
{
  //Serial.println("Enter handleRoot");
  String header;
  if (!is_authentified())
  {
    server.sendHeader("Location", "/login");
    server.sendHeader("Cache-Control", "no-cache");
    server.send(301);
    return;
  }

  String a = P3M_ControloM; server.send(200, "text/html", a);

  if (server.hasArg("enviarValores"))
  {
    //Serial.println("Envio Valores Mentor.");
    w = server.arg("sliderWaist");     //Serial. print("w:"); Serial.println(w);
```

```
  s = server.arg("sliderShoulder");  //Serial. print("s:"); Serial.println(s);
  e = server.arg("sliderElbow");     //Serial. print("e:"); Serial.println(e);
  ev = server.arg("sliderEVwrist");  //Serial. print("ev:"); Serial.println(ev);
  wr = server.arg("sliderWRwrist");  //Serial. print("wr:"); Serial.println(wr);
  if (server.hasArg("GarraFechada"))
  {
    g = 196;  // Garra Fechada
    //Serial. print("g:");Serial.println(g);
  }
  else
  {
    g = 151; // Garra Aberta
    //Serial. print("g:");Serial.println(g);
  }
  //Serial.println("");

  Serial.println(" Wi-Fi ->Movimentação Mentor:\n");
  ESP_BT.println(" Wi-Fi ->Movimentação Mentor:\n");
  char buffer[40];
  sprintf(buffer, "W %s\n\0", w); //Serial.print(buffer);
  cmdHdl.processString(buffer); delay(1000);

  sprintf(buffer, "S %s\n\0", s); //Serial.print(buffer);
  cmdHdl.processString(buffer); delay(1000);

  sprintf(buffer, "E %s\n\0", e); //Serial.print(buffer);
  cmdHdl.processString(buffer); delay(1000);

  sprintf(buffer, "EV %s\n\0", ev); //Serial.print(buffer);
  cmdHdl.processString(buffer); delay(1000);

  sprintf(buffer, "WR %s\n\0", wr); //Serial.print(buffer);
  cmdHdl.processString(buffer); delay(1000);

  sprintf(buffer, "G %s\n\0", g); //Serial.print(buffer);
  cmdHdl.processString(buffer);
  //Serial.println("\n"); //3
}

if (server.hasArg("PI"))
{
  Serial.println(" Wi-FI ->Home");
  ESP_BT.println(" Wi-FI ->Home");
  PosicaoInicial();            // Posição Inicial do Mentor
}

if (server.hasArg("P1"))
{
  Serial.println(" Wi-Fi ->Posição 1:");
  ESP_BT.println(" Wi-Fi ->Posição 1:");

  w1 = server.arg("sliderWaist");
  Serial.print("W1:"); Serial.print(w1);
  ESP_BT.print("W1:"); ESP_BT.print(w1);

  s1 = server.arg("sliderShoulder");
  Serial.print(" S1:"); Serial.print(s1);
  ESP_BT.print(" S1:"); ESP_BT.print(s1);

  e1 = server.arg("sliderElbow");
  Serial.print(" E1:"); Serial.print(e1);
  ESP_BT.print(" E1:"); ESP_BT.print(e1);

  ev1 = server.arg("sliderEVwrist");
  Serial.print(" EV1:"); Serial.print(ev1);
  ESP_BT.print(" EV1:");  ESP_BT.print(ev1);

  wr1 = server.arg("sliderWRwrist");
  Serial.print(" WR1:"); Serial.print(wr1);
  ESP_BT.print(" WR1:"); ESP_BT.print(wr1);

  if (server.hasArg("GarraFechada"))
  {
    g1 = 196;  // Garra Fechada
    Serial.print(" G1:"); Serial.print(g1);
    ESP_BT.print(" G1:"); ESP_BT.print(g1);
```

```
  }
  else
  {
    g1 = 151; // Garra Aberta
    Serial.print(" G1:"); Serial.print(g1);
    ESP_BT.print(" G1:"); ESP_BT.print(g1);
  }
  Serial.println("\n");
  ESP_BT.println("\n");
}

if (server.hasArg("P2"))
{
  Serial.println(" Wi-Fi ->Posição 2:");
  ESP_BT.println(" Wi-Fi ->Posição 2:");

  w2 = server.arg("sliderWaist");
  Serial.print("W2:"); Serial.print(w2);
  ESP_BT.print("W2:"); ESP_BT.print(w2);

  s2 = server.arg("sliderShoulder");
  Serial.print(" S2:"); Serial.print(s2);
  ESP_BT.print(" S2:"); ESP_BT.print(s2);

  e2 = server.arg("sliderElbow");
  Serial.print(" E2:"); Serial.print(e2);
  ESP_BT.print(" E2:"); ESP_BT.print(e2);

  ev2 = server.arg("sliderEVwrist");
  Serial.print(" EV2:"); Serial.print(ev2);
  ESP_BT.print(" EV2:"); ESP_BT.print(ev2);

  wr2 = server.arg("sliderWRwrist");
  Serial.print(" WR2:"); Serial.print(wr2);
  ESP_BT.print(" WR2:"); ESP_BT.print(wr2);

  if (server.hasArg("GarraFechada"))
  {
    g2 = 196;  // Garra Fechada
    Serial.print(" G2:"); Serial.print(g2);
    ESP_BT.print(" G2:"); ESP_BT.print(g2);
  }
  else
  {
    g2 = 151; // Garra Aberta
    Serial.print(" G2:"); Serial.print(g2);
    ESP_BT.print(" G2:"); ESP_BT.print(g2);
  }
  Serial.println("\n");
  ESP_BT.println("\n");
}

if (server.hasArg("P3"))
{
  Serial.println(" Wi-Fi ->Posição 3:");
  ESP_BT.println(" Wi-Fi ->Posição 3:");

  w3 = server.arg("sliderWaist");
  Serial.print("W3:"); Serial.print(w3);
  ESP_BT.print("W3:"); ESP_BT.print(w3);

  s3 = server.arg("sliderShoulder");
  Serial.print(" S3:"); Serial.print(s3);
  ESP_BT.print(" S3:"); ESP_BT.print(s3);

  e3 = server.arg("sliderElbow");
  Serial.print(" E3:"); Serial.print(e3);
  ESP_BT.print(" E3:"); ESP_BT.print(e3);

  ev3 = server.arg("sliderEVwrist");
  Serial.print(" EV3:"); Serial.print(ev3);
  ESP_BT.print(" EV3:"); ESP_BT.print(ev3);

  wr3 = server.arg("sliderWRwrist");
  Serial.print(" WR3:"); Serial.print(wr3);
  ESP_BT.print(" WR3:"); ESP_BT.print(wr3);
```

```
  if (server.hasArg("GarraFechada"))
  {
    g3 = 196;  // Garra Fechada
    Serial.print(" G3:"); Serial.print(g3);
    ESP_BT.print(" G3:"); ESP_BT.print(g3);
  }
  else
  {
    g3 = 151; // Garra Aberta
    Serial.print(" G3:"); Serial.print(g3);
    ESP_BT.print(" G3:"); ESP_BT.print(g3);
  }
  Serial.println("\n");
  ESP_BT.println("\n");
}

if (server.hasArg("P4"))
{
  Serial.println(" Wi-Fi ->Posição 4:");
  ESP_BT.println(" Wi-Fi ->Posição 4:");

  w4 = server.arg("sliderWaist");
  Serial.print("W4:"); Serial.print(w4);
  ESP_BT.print("W4:"); ESP_BT.print(w4);

  s4 = server.arg("sliderShoulder");
  Serial.print(" S4:"); Serial.print(s4);
  ESP_BT.print(" S4:"); ESP_BT.print(s4);

  e4 = server.arg("sliderElbow");
  Serial.print(" E4:"); Serial.print(e4);
  ESP_BT.print(" E4:"); ESP_BT.print(e4);

  ev4 = server.arg("sliderEVwrist");
  Serial.print(" EV4:"); Serial.print(ev4);
  ESP_BT.print(" EV4:"); ESP_BT.print(ev4);

  wr4 = server.arg("sliderWRwrist");
  Serial.print(" WR4:"); Serial.print(wr4);
  ESP_BT.print(" WR4:"); ESP_BT.print(wr4);

  if (server.hasArg("GarraFechada"))
  {
    g4 = 196;  // Garra Fechada
    Serial.print(" G4:"); Serial.print(g4);
    ESP_BT.print(" G4:"); ESP_BT.print(g4);
  }
  else {
    g4 = 151; // Garra Aberta
    Serial.print(" G4:"); Serial.print(g4);
    ESP_BT.print(" G4:"); ESP_BT.print(g4);
  }
  Serial.println("\n");
  ESP_BT.println("\n");
}

if (server.hasArg("RUN"))
{
  Serial.println(" Wi-Fi ->Robô Mentor - A correr Posições.");
  ESP_BT.println(" Wi-Fi ->Robô Mentor - A correr Posições.");

  Serial.print("W1:"); Serial.print(w1); Serial.print(" S1:"); Serial.print(s1); Serial.print(" E1:"); Serial.print(e1); Serial.print(" EV1:"); Serial.print(ev1); Serial.print(" WR1:"); Serial.print(wr1); Serial.print(" G1:"); Serial.println(g1);
  ESP_BT.print("W1:"); ESP_BT.print(w1); ESP_BT.print(" S1:"); ESP_BT.print(s1); ESP_BT.print(" E1:"); ESP_BT.print(e1); ESP_BT.print(" EV1:"); ESP_BT.print(ev1); ESP_BT.print(" WR1:"); ESP_BT.print(wr1); ESP_BT.print(" G1:"); ESP_BT.println(g1);

  Serial.print("W2:"); Serial.print(w2); Serial.print(" S2:"); Serial.print(s2); Serial.print(" E2:"); Serial.print(e2); Serial.print(" EV2:"); Serial.print(ev2); Serial.print(" WR2:"); Serial.print(wr2); Serial.print(" G2:"); Serial.println(g2);
  ESP_BT.print("W2:"); ESP_BT.print(w2); ESP_BT.print(" S2:"); ESP_BT.print(s2); ESP_BT.print(" E2:"); ESP_BT.print(e2); ESP_BT.print(" EV2:"); ESP_BT.print(ev2); ESP_BT.print(" WR2:"); ESP_BT.print(wr2); ESP_BT.print(" G2:"); ESP_BT.println(g2);

  Serial.print("W3:"); Serial.print(w3); Serial.print(" S3:"); Serial.print(s3); Serial.print(" E3:"); Serial.print(e3); Serial.print(" EV3:"); Serial.print(ev3); Serial.print(" WR3:"); Serial.print(wr3); Serial.print(" G3:"); Serial.println(g3);
```

```
   ESP_BT.print("W3:"); ESP_BT.print(w3); ESP_BT.print(" S3:"); ESP_BT.print(s3); ESP_BT.print(" E3:"); ESP_BT.print(e3); ESP_BT.print(" EV3:"); ESP_BT.print(ev3); ESP_BT.print(" WR3:"); ESP_BT.print(wr3); ESP_BT.print(" G3:"); ESP_BT.println(g3);

   Serial.print("W4:"); Serial.print(w4); Serial.print(" S4:"); Serial.print(s4); Serial.print(" E4:"); Serial.print(e4); Serial.print(" EV4:"); Serial.print(ev4); Serial.print(" WR4:"); Serial.print(wr4); Serial.print(" G4:"); Serial.print(g4); Serial.println("\n");
   ESP_BT.print("W4:"); ESP_BT.print(w4); ESP_BT.print(" S4:"); ESP_BT.print(s4); ESP_BT.print(" E4:"); ESP_BT.print(e4); ESP_BT.print(" EV4:"); ESP_BT.print(ev4); ESP_BT.print(" WR4:"); ESP_BT.print(wr4); ESP_BT.print(" G4:"); ESP_BT.print(g4); ESP_BT.println("\n");


   char buffer[40];
   sprintf(buffer, "W %s\n\0", w1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "S %s\n\0", s1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "E %s\n\0", e1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "EV %s\n\0", ev1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "WR %s\n\0", wr1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "G %s\n\0", g1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   //Serial.println("");
   delay(2000);

   sprintf(buffer, "W %s\n\0", w2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "S %s\n\0", s2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "E %s\n\0", e2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "EV %s\n\0", ev2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "WR %s\n\0", wr2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "G %s\n\0", g2); //Serial.println(buffer);
   cmdHdl.processString(buffer);
   //Serial.println("");
   delay(2000);

   sprintf(buffer, "W %s\n\0", w3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "S %s\n\0", s3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "E %s\n\0", e3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "EV %s\n\0", ev3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "WR %s\n\0", wr3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "G %s\n\0", g3); //Serial.println(buffer);
   cmdHdl.processString(buffer);
   //Serial.println("");
   delay(2000);

   sprintf(buffer, "W %s\n\0", w4); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "S %s\n\0", s4); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "E %s\n\0", e4); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "EV %s\n\0", ev4); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "WR %s\n\0", wr4); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "G %s\n\0", g4); //Serial.println(buffer);
   cmdHdl.processString(buffer);
   //Serial.println("");
 }

 if (server.hasArg("RESET"))
 {
   Serial.println(" Wi-Fi ->Reset Posições\n");
   ESP_BT.println(" Wi-Fi ->Reset Posições\n");
```

```
  }
}
```

⑦ Mentor

```
//_____________________________________________________________________________________________________
__Declaração de variáveis  flat cable (20 pinos)
//                                                                    [ Microcontrolador -> FlatCable -> Mentor ]
// Shift Register
int latchPin = 26;     // Latch pin of 74HC595
int clockPin = 4;      // Clock pin of 74HC595
int dataPin  = 25;     // Data  pin of 74HC595

// Flat Cable
const int B_0 = 13;    // FC1
const int B_1 = 27;    // FC3
const int B_2 = 33;    // FC5
const int B_3 = 15;    // FC7
const int B_4 = 32;    // FC9
const int B_5 = 14;    // FC11
const int B_6 = 22;    // FC13
const int B_7 = 23;    // FC15
const int RW  = 18;    // FC17
const int BK  = 19;    // FC12

//_____________________________________________________________________________________________________
___________________Declaração de Variáveis
int wpos  = 0;     int spos  = 0;      int epos  = 0;     int lmpos  = 0;      int rmpos  = 0;    int gpos  = 0;       // W S E Lm Rm G (para
guardar leitura)
int datarot = 128;  int dataelev = 128;   int datalm = 128;  int datarm = 128;                                  // Rotação Elevação Lm Rm
char mode = OUTPUT; int data = 128;  int datar = 0; int address = 224;
// data1 = 128; data2=128;



void inicializarPins()
{
 //Pins Shift Register 74HC595
 pinMode(latchPin, OUTPUT);
 pinMode(dataPin,  OUTPUT);
 pinMode(clockPin, OUTPUT);

 pinMode(B_0, mode);
 pinMode(B_1, mode);
 pinMode(B_2, mode);
 pinMode(B_3, mode);
 pinMode(B_4, mode);
 pinMode(B_5, mode);
 pinMode(B_6, mode);
 pinMode(B_7, mode);

 pinMode(RW, OUTPUT);
 pinMode(BK, OUTPUT);

 //_____________________________ PWM de 1MHz  [f = 1 000 000Hz| T=1us= 0,00001]
 pinMode(21, OUTPUT);        //Definimos o pino 21 como saída.
 ledcAttachPin(21, 1);       //Atribuimos o pino 21 ao canal 1.
 ledcSetup(1, 1000000, 6);   //Atribuimos ao canal 1 a frequencia de 1 000 000 Hz com resolucao de 6bits.
 ledcWrite(1, 32);           //Escrevemos um duty cycle de 50% no canal 1.
 // 80 000 000/2e6 = 1 250 000
 digitalWrite(RW, LOW);
 digitalWrite(BK, LOW);
}



// Comandar robo Mentor
//_____________________________________________________________________________________________________
__
void R()
{
 digitalWrite(RW, HIGH);
}
void WRI()
```

```
{
  digitalWrite(RW, LOW);
}
void BO()
{
  digitalWrite(BK, HIGH);
}
void BF()
{
  digitalWrite(BK, LOW);
}
void AD()
{
  // Shift Register 74HC595
  // 15A 1B 2C 3D 4E 5F 6G 7H
  // 0   1  2  3  4  5  6  7
  sr.set(0, HIGH && (address & B00000001));              // Shift Register 15     //FC2
  sr.set(1, HIGH && (address & B00000010));              // Shift Register 1      //FC4
  sr.set(2, HIGH && (address & B00000100));              // Shift Register 2      //FC6
  sr.set(3, HIGH && (address & B00001000));              // Shift Register 3      //FC8
  sr.set(4, HIGH && (address & B00010000));              // Shift Register 4      //FC10
  sr.set(5, HIGH && (address & B00100000));              // Shift Register 5      //FC14
  sr.set(6, HIGH && (address & B01000000));              // Shift Register 6      //FC16
  sr.set(7, HIGH && (address & B10000000));              // Shift Register 7      //FC18
  //  sr.setAll(address)

  //  uint8_t address_v[] = { address };
  //  sr.setAll(address_v);
}
void DA()
{
  digitalWrite(B_0, HIGH && (data & B00000001));
  digitalWrite(B_1, HIGH && (data & B00000010));
  digitalWrite(B_2, HIGH && (data & B00000100));
  digitalWrite(B_3, HIGH && (data & B00001000));
  digitalWrite(B_4, HIGH && (data & B00010000));
  digitalWrite(B_5, HIGH && (data & B00100000));
  digitalWrite(B_6, HIGH && (data & B01000000));
  digitalWrite(B_7, HIGH && (data & B10000000));
}

void readpins()
{
  mode = INPUT;
  setpins();
  datar = B11111110 | digitalRead(B_0);
  datar = datar & (B11111101 | (digitalRead(B_1) << 1));
  datar = datar & (B11111011 | (digitalRead(B_2) << 2));
  datar = datar & (B11110111 | (digitalRead(B_3) << 3));
  datar = datar & (B11101111 | (digitalRead(B_4) << 4));
  datar = datar & (B11011111 | (digitalRead(B_5) << 5));
  datar = datar & (B10111111 | (digitalRead(B_6) << 6));
  datar = datar & (B01111111 | (digitalRead(B_7) << 7));
}

void setpins()
{
  pinMode(B_0, mode);
  pinMode(B_1, mode);
  pinMode(B_2, mode);
  pinMode(B_3, mode);
  pinMode(B_4, mode);
  pinMode(B_5, mode);
  pinMode(B_6, mode);
  pinMode(B_7, mode);
}

void ALE()
{
  address = 230;
  BO();
  DA();
  AD();
  BF();
}
void SC()
```

```
{
 address = 231;
 data = 0;
 BO();
 DA();
 AD();
 BF();
 delay (5);
}
void OEF()
{
 address = 17;
 BO();
 AD();
 BF();
}
void OEL()
{
 address = 16;
 BO();
 AD();
 BF();
}
void OEI()
{
 address = 18;
 BO();
 AD();
 BF();
}
//_____________________________________________________________________________________________
__


//_____________________________________________________________________________________________
__COMANDO DE ENVIO DO VALOR DE POSIÇÃO
//_____________________________________________________________________________________________
__COMANDOS DE MOVIMENTAÇÃO
void W()
{
 int wString;
 wString = cmdHdl.readIntArg();
 if (cmdHdl.argOk)
  {
   data = wString;
   if (data < 0)
   {
    Serial.print("Comando W "); Serial.print(wString ); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando W "); ESP_BT.print(wString);  ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else if (data > 250)
   {
    Serial.print("Comando W "); Serial.print(wString); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando W "); ESP_BT.print(wString); ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else
   {
    Serial.print("Movimentação Waist, W:"); Serial.print(wString); Serial.println(".\n");
    ESP_BT.print("Movimentação Waist, W:"); ESP_BT.print(wString); ESP_BT.println(".\n");
    BO();
    address = 224;
    DA();
    AD();
    BF();
   }
  }
  else
  {
   Serial.println("Comando W sem argumento, insira um novo comando.\n");
   ESP_BT.println("Comando W sem argumento, insira um novo comando.\n");
  }
}
void S()
{
 int sString;
```

```
  sString = cmdHdl.readIntArg();
  if (cmdHdl.argOk)
  {
   data = sString;
   if (data < 0)
   {
    Serial.print("Comando S "); Serial.print(sString); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando S "); ESP_BT.print(sString); ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else if (data > 250)
   {
    Serial.print("Comando S "); Serial.print(sString); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando S "); ESP_BT.print(sString); ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else
   {
    Serial.print("Movimentação Shoulder, S:"); Serial.print(sString); Serial.println(".\n");
    ESP_BT.print("Movimentação Shoulder, S:"); ESP_BT.print(sString); ESP_BT.println(".\n");
    BO();
    address = 225;
    DA();
    AD();
    BF();
   }
  }
  else
  {
   Serial.println("Comando S sem argumento, insira um novo comando.\n");
   ESP_BT.println("Comando S sem argumento, insira um novo comando.\n");
  }
}
void E()
{
  int eString;
  eString = cmdHdl.readIntArg();
  if (cmdHdl.argOk)
  {
   data = eString;
   if (data < 0)
   {
    Serial.print("Comando E "); Serial.print(eString); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando E "); ESP_BT.print(eString); ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else if (data > 250)
   {
    Serial.print("Comando E "); Serial.print(eString); Serial.println(", Argumento Inválido! Repita.\n");
    ESP_BT.print("Comando E "); ESP_BT.print(eString); ESP_BT.println(", Argumento Inválido! Repita.\n");
   }
   else
   {
    Serial.print("Movimentação Elbow, E:"); Serial.print(eString); Serial.println(".\n");
    ESP_BT.print("Movimentação Elbow, E:"); ESP_BT.print(eString); ESP_BT.println(".\n");
    BO();
    address = 226;
    DA();
    AD();
    BF();
   }
  }
  else
  {
   Serial.println("Comando E sem argumento, insira um novo comando.\n");
   ESP_BT.println("Comando E sem argumento, insira um novo comando.\n");
  }
}
void LM()
{
  BO();
  address = 227;
  DA();
  AD();
  BF();
}
void RM()
{
  BO();
```

```
  address = 228;
  DA();
  AD();
  BF();
}
void EV()
{
  int evString;
  evString = cmdHdl.readIntArg();
  if (cmdHdl.argOk)
  {
    data = evString;
    if (data < 77)
    {
      Serial.print("Comando EV "); Serial.print(evString); Serial.println(", Argumento Inválido! Repita.\n");
      ESP_BT.print("Comando EV "); ESP_BT.print(evString); ESP_BT.println(", Argumento Inválido! Repita.\n");
    }
    else if (data > 178)
    {
      Serial.print("Comando EV "); Serial.print(evString); Serial.println(", Argumento Inválido! Repita.\n");
      ESP_BT.print("Comando EV "); ESP_BT.print(evString); ESP_BT.println(", Argumento Inválido! Repita.\n");
    }
    else
    {
      Serial.print("Movimentação Elevação Wrist, EV:"); Serial.print(evString); Serial.println(".\n");
      ESP_BT.print("Movimentação Elevação Wrist, EV:"); ESP_BT.print(evString); ESP_BT.println(".\n");

      // after
      //      data = data - dataelev;  dataelev = dataelev + data;
      //      datalm = datalm + data;
      //      datarm = datarm + data;
      //
      //      BO();
      //
      //      //datalm = data2;
      //      data = datalm;
      //      LM();
      //
      //      //datarm = data1;
      //      data = datarm;
      //      RM();
      //
      //      BF();

      // Solução Elevação +1
      BO();
      data = evString + 1;
      LM();
      data = evString + 1;
      RM();
      BF();
    }
  }
  else
  {
    Serial.println("Comando EV sem argumento, insira um novo comando.\n");
    ESP_BT.println("Comando EV sem argumento, insira um novo comando.\n");
  }
}

void WR()
{
  int wrString;
  wrString = cmdHdl.readIntArg();
  if (cmdHdl.argOk)
  {
    data = wrString;
    if (data < 0)
    {
      Serial.print("Comando WR "); Serial.print(wrString); Serial.println(", Argumento Inválido! Repita.\n");
      ESP_BT.print("Comando WR "); ESP_BT.print(wrString); ESP_BT.println(", Argumento Inválido! Repita.\n");
    }
    else if (data > 255)
    {
      Serial.print("Comando WR "); Serial.print(wrString); Serial.println(", Argumento Inválido! Repita.\n");
      ESP_BT.print("Comando WR "); ESP_BT.print(wrString); ESP_BT.println(", Argumento Inválido! Repita.\n");
```

```
    }
    else
    {
     Serial.print("Movimentação Rotação Wrist, WR:"); Serial.print(wrString); Serial.println(".\n");
     ESP_BT.print("Movimentação Rotação Wrist, WR:"); ESP_BT.print(wrString); ESP_BT.println(".\n");

     // after
     //      data = data - datarot;     datarot = datarot + data;
     //
     //      datalm = datalm + data;
     //      datarm = datarm - data;
     //
     //      BO();
     //
     //      data = datalm;
     //      LM();
     //      data = datarm;
     //      RM();
     //
     //      BF();

     // Solução Rotação +1
     BO();
     data = round(- (wrString / 2));
     LM();
     data = round (wrString / 2);
     RM();
     BF();
    }
   }
   else
   {
    Serial.println("Comando WR sem argumento, insira um novo comando.\n");
    ESP_BT.println("Comando WR sem argumento, insira um novo comando.\n");
   }
}

void G()
{
  int gString;
  gString = cmdHdl.readIntArg();
  if (cmdHdl.argOk)
  {
    data = gString;
    if (data < 151)
    {
     Serial.print("Comando G "); Serial.print(gString); Serial.println(", Argumento Inválido! Repita.\n");
     ESP_BT.print("Comando G "); ESP_BT.print(gString); ESP_BT.println(", Argumento Inválido! Repita.\n");
    }
    else if (data > 196)
    {
     Serial.print("Comando G "); Serial.print(gString); Serial.println(", Argumento Inválido! Repita.\n");
     ESP_BT.print("Comando G "); ESP_BT.print(gString); ESP_BT.println(", Argumento Inválido! Repita.\n");
    }
    else
    {
     Serial.print("Movimentação Garra, G:"); Serial.print(gString); Serial.println(".\n");
     ESP_BT.print("Movimentação Garra, G:"); ESP_BT.print(gString); ESP_BT.println(".\n");
     BO();
     address = 229;
     DA();
     AD();
     BF();
    }
  }
  else
  {
    Serial.println("Comando G sem argumento, insira um novo comando.\n");
    ESP_BT.println("Comando G sem argumento, insira um novo comando.\n");
  }
}
//_____________________________________________________________________________________________
__
```

```
//_____________________________________________________________________________________________
__COMANDOS DE LEITURA DE POSIÇÕES
// Ler as posições actuais de todos os eixos do braço.
void RD()
{
  Serial.println("Robô Mentor - Leitura de Eixos.");
  ESP_BT.println("Robô Mentor - Leitura de Eixos.");
  WF();
  SF();
  EF();
  LMF();
  RMF();
  GF();

  Serial.println("\n");
  ESP_BT.println("\n");
}

void WF()
{
  data = 0;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay (100);
  wpos = datar;
  Serial.print("W:"); Serial.print(wpos);
  ESP_BT.print("W:"); ESP_BT.print(wpos);
  WRI();
  mode = OUTPUT;
  setpins();
}
void SF()
{
  data = 16;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay (100);
  spos = datar;
  Serial.print(" S:"); Serial.print(spos);
  ESP_BT.print(" S:"); ESP_BT.print(spos);
  WRI();
  mode = OUTPUT;
  setpins();
}
void EF()
{
  data = 32;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay (100);
  epos = datar;
  Serial.print(" E:"); Serial.print(epos);
  ESP_BT.print(" E:"); ESP_BT.print(epos);
  WRI();
  mode = OUTPUT;
  setpins();
}
void LMF()
{
  data = 48;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay(100);
  lmpos = datar;
```

```
  Serial.print(" Lm:"); Serial.print(lmpos);  //Lm
  ESP_BT.print(" Lm:"); ESP_BT.print(lmpos);
  WRI();
  mode = OUTPUT;
  setpins();
}
void RMF()
{
  data = 64;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay(100);
  rmpos = datar;
  Serial.print(" Rm:"); Serial.print(rmpos);  //Rm
  ESP_BT.print(" Rm:"); ESP_BT.print(rmpos);
  WRI();
  mode = OUTPUT;
  setpins();
}
void GF()
{
  data = 80;
  ALE();
  SC();
  R();
  OEF();
  readpins();
  delay(100);
  gpos = datar;
  Serial.print(" G:"); Serial.print(gpos);
  ESP_BT.print(" G:"); ESP_BT.print(gpos);
  WRI();
  mode = OUTPUT;
  setpins();
}
//_____________________________________________________________________________________________________
__


//_____________________________________________________________________________________________________
__Posição Incial do Mentor
void H()
{
  PosicaoInicial();
}

void PosicaoInicial()
{
  Serial.println("Robô Mentor - Posição inicial.");
  Serial.println("W:128 S:128 E:128 EV:128 WR:128 G:151\n");
  ESP_BT.println("Robô Mentor - Posição inicial.");
  ESP_BT.println("W:128 S:128 E:128 EV:128 WR:128 G:151\n");

  data = 128;
  //  W
  BO();
  address = 224;
  DA();
  AD();
  BF();
  delay (100);

  data = 128;
  //  S
  BO();
  address = 225;
  DA();
  AD();
  BF();
  delay (100);

  data = 128;
```

```
  //  E
  BO();
  address = 226;
  DA();
  AD();
  BF();
  delay (100);

  data = 128;
  //  LM
  BO();
  address = 227;
  DA();
  AD();
  BF();
  delay (100);

  data = 128;
  //  RM
  BO();
  address = 228;
  DA();
  AD();
  BF();
  delay (100);


  data = 151;
  //  G();
  BO();
  address = 229;
  DA();
  AD();
  BF();
  delay (100);
}
//_____________________________________________________________________________________________
__
```

⑧ htmlsM.h

```
//*****************************************************************************************************
*************************************************************************
const char P1M_Entrada[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Mentor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 35%;}
/* Middle column */
.column.middle {width: 30%; background-color: none;}
```

```
/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


input[type=text], input[type=password] {width: 100%;padding: 12px 20px;margin: 8px 0;display: inline-block;border: 1px solid #ccc;box-sizing: border-box;}
button {background-color: #4CAF50;color: white;padding: 14px 20px;margin: 8px 0;border: none;cursor: pointer;width: 100%;}
button:hover {opacity: 0.8;}
.container {padding: 16px;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/sobre">Sobre</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

   <div class="container">

  <h2 style="text-align:center">Segurança de entrada</h2>
  <p>Para entrar, por favor introduza o nome de utilizador e a palavra-passe.</p>
  <br><br>
  <form action='/login' method='POST'>

     <label for="USERNAME"><b>Nome de Utilizador</b></label>
     <input type="text" placeholder="Introduza o nome de utilizador" name="USERNAME" >

     <label for="PASSWORD"><b>Palavra-Passe</b></label>
    <input type="password" placeholder="Introduza a palavra-passe" name="PASSWORD" >

    <button type="submit">Login</button>
  <br><br><br>
  </form>
  </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
  <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin" target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)======";
```

```
//*****************************************************************************************************************
*************************************************************************
const char P2M_Sobre[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Mentor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}


/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}

</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/login">Voltar</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

  <h2 style="text-align:center">Sobre a Interface</h2>
  <div class="a">
```

```
 <br>
 Para utilizar a Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos deverá efetuar Login, introduzindo o nome
 de utilizador e a palavra-passe corretos.</p>
 A Interface Remota Gen&#233rica, permite controlar remotamente qualquer braço rob&#243tico, a partir do
 protocolo de comunicação Wi-Fi.</p>
 A validação do Login, serve como segurança, para o acesso ao controlo do braço robótico.</p>
 <br><br><br><br><br>

 </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
 <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin"
target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)=====";


//*************************************************************************************************************
*************************************************************************
const char P3M_ControloM[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Mentor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
ul.topnav                    {list-style-type: none;margin: 0;padding: 0;overflow: hidden; border: 1px solid #cccccc; background-color:
#f2f2f2;}
ul.topnav li                 {float: left; } /*border-right:1px solid white; border-bottom: 1px solid white;*/
ul.topnav li a               {display: block;color: black;text-align: center;padding: 14px 16px;text-decoration: none;}
ul.topnav li a:hover:not(.active) {background-color:  #ddd;  color: white;}
ul.topnav li a.active          {background-color: #4CAF50;}
ul.topnav li.right             {float: right;}
@media screen and (max-width: 600px) {ul.topnav li.right, ul.topnav li {float: none;}}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 25%;}
/* Middle column */
.column.middle {width: 50%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}



/* coluna do meio com 2 boxs */
```

```
.box {float: left; width: 50%; padding: 17px;}
.clearfix::after {content: "";clear: both;display: table;}

/* Botões */
.button {display: inline-block; padding: none; font-size: none; cursor: pointer; text-align: center; text-decoration: none; outline: none;
color: black; background-color: #F0F0F0; border: none; border-radius: 10px; box-shadow: 0 5px #999; width: 150px;}
.button1:hover {background-color: #008CBA;}
.button1:active { background-color: #008CBA; box-shadow: 0 3px #666; transform: translateY(4px);}
.button2:hover {background-color: #cc9900;}
.button2:active { background-color: #e7e7e7; box-shadow: 0 3px #666; transform: translateY(4px);}
.button3:hover {background-color: #4CAF50;}
.button3:active { background-color: #4CAF50; box-shadow: 0 3px #666; transform: translateY(4px);}
.button4:hover {background-color: #f44336;}
.button4:active { background-color: #f44336; box-shadow: 0 3px #666; transform: translateY(4px);}
.buttonE {display: inline-block; padding: 8px 22px; font-size: none; cursor: pointer; text-align: center; text-decoration: none; outline:
none; color: black; background-color: #F0F0F0; border: none; border-radius: 5px; box-shadow: 0 5px #999; width: 150px;}
.buttonE:hover {background-color: #33cc33;}
.buttonE:active { background-color: #e7e7e7; box-shadow: 0 3px #666; transform: translateY(4px);}
.slider:hover {opacity: 3;}

/* Switch Slide============================*/
.input-toggle {line-height: 0;font-size: 0;display: inline-block;margin: 0;}
/* Hide original checkbox, but don't use 'display: none' to allow focus on it using keyboard*/
.input-toggle input {display: block;position: absolute;opacity: 0;width: 0;height: 0;}
.switch {display: inline-block;position: relative;height: 0.75rem;width: 2.25rem;border-radius: 0.5rem;background-color:
#CCC;cursor: pointer;  transition: all 200ms ease-in-out;}
.switch:before {content:"";display: block;height: 1.1rem;width: 1.1rem;border-radius: 100%;background-color: #FAFAFA;box-
shadow: 0 1px 3px rgba(0,0,0,0.5);position: absolute;top: -15%;left: 0;transition: all 200ms ease-in-out;}
.input-toggle input:checked + .switch{  background-color: green;transition: all 200ms ease-in-out;}
.input-toggle input:checked + .switch:before {left: 1.25rem;transition: all 200ms ease-in-out;}

/* TV */
.centerTV {background-color: none; font-size: 14px; margin-left:auto; margin-right:auto;justify-content: center;align-items:
center;height: 140px;width: 350px;border: 3px solid #e7e7e7; border-top-left-radius: 10px;border-top-right-radius: 10px;border-
bottom-left-radius: 10px;border-bottom-right-radius: 10px; border-style: ridge;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<ul class="topnav">
  <li><a href="/instrucoesM">Instruções Mentor</a></li>
  <li><a href="/sobreM">Sobre Mentor</a></li>
  <li class="right"><a href="/login?DISCONNECT=YES">Sair Mentor</a></li>
</ul>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:center">
    <h2 >Controlo Mentor</h2>
    <p>Use os sliders para ajustar os eixos.<br>Pode guardar e correr 4 poses.</p>

    <div class="centerTV" >
<p>
 <span id="demoW1"></span><span id="demoSW1"></span>
 <span id="demoS1"></span><span id="demoSS1"></span>
 <span id="demoE1"></span><span id="demoSE1"></span>
 <span id="demoEV1"></span><span id="demoSEV1"></span>
 <span id="demoWR1"></span><span id="demoSWR1"></span>
 <span id="demoG1"></span><span id="demoSG1"></span>
</p>
<p>
```

```
 <span id="demoW2"></span><span id="demoSW2"></span>
 <span id="demoS2"></span><span id="demoSS2"></span>
 <span id="demoE2"></span><span id="demoSE2"></span>
 <span id="demoEV2"></span><span id="demoSEV2"></span>
 <span id="demoWR2"></span><span id="demoSWR2"></span>
 <span id="demoG2"></span><span id="demoSG2"></span>
</p>
<p>
 <span id="demoW3"></span><span id="demoSW3"></span>
 <span id="demoS3"></span><span id="demoSS3"></span>
 <span id="demoE3"></span><span id="demoSE3"></span>
 <span id="demoEV3"></span><span id="demoSEV3"></span>
 <span id="demoWR3"></span><span id="demoSWR3"></span>
 <span id="demoG3"></span><span id="demoSG3"></span>
</p>
<p>
 <span id="demoW4"></span><span id="demoSW4"></span>
 <span id="demoS4"></span><span id="demoSS4"></span>
 <span id="demoE4"></span><span id="demoSE4"></span>
 <span id="demoEV4"></span><span id="demoSEV4"></span>
 <span id="demoWR4"></span><span id="demoSWR4"></span>
 <span id="demoG4"></span><span id="demoSG4"></span>
</p>
   </div>

  <div class="clearfix">

<form action="/" method="post">
   <div class="box" style="text-align:right">
  <p><div >Waist: <span id="demo1"></span><input type="range"       min="0"  max="250"     class="slider"  id="myRange1"
name="sliderWaist"></div></p>
  <p><div >Shoulder: <span id="demo2"></span><input type="range"  min="0"    max="250"     class="slider"  id="myRange2"
name="sliderShoulder"></div></p>
  <p><div >Elbow: <span id="demo3"></span><input type="range"      min="0"    max="250"    class="slider"  id="myRange3"
name="sliderElbow"></div></p>
  <p><div >EVwrist: <span id="demo4"></span><input type="range"   min="77"    max="178"   class="slider"  id="myRange4"
name="sliderEVwrist"></div></p>
  <p><div >WRwrist: <span id="demo5"></span><input type="range"   min="0"    max="255"   class="slider"  id="myRange5"
name="sliderWRwrist"></div></p>

<p>Garra: Aberta <label class="input-toggle" for="switch">
<input type="checkbox" id="switch" name="GarraFechada"><span class="switch"></span>
</label>  Fechada</p>

  <p><button       class="button       buttonE"       onclick="showEnviar()"       style="vertical-align:middle"       type="submit"
name="enviarValores">ENVIAR!</button></p>
<p>
 <span id="demoW"></span><span id="demoSW"></span>
 <span id="demoS"></span><span id="demoSS"></span>
 <span id="demoE"></span><span id="demoSE"></span>
 <span id="demoEV"></span><span id="demoSEV"></span>
 <span id="demoWR"></span><span id="demoSWR"></span>
 <span id="demoG"></span><span id="demoSG"></span>
</p>
 </div>


   <div class="box" style="text-align:left">
  <p><button class="button button1" name="PI" value="valoresPI" type="submit">Posi&#231&#227o Inicial!</button></p>
  <p><button class="button button2" onclick="updateP1()" name="P1" value="valoresP1" type="submit">SAVE Posi&#231&#227o
1</button> </p>

  <p><button class="button button2" onclick="updateP2()" name="P2" value="valoresP2" type="submit">SAVE Posi&#231&#227o
2</button> </p>

  <p><button class="button button2" onclick="updateP3()" name="P3" value="valoresP3" type="submit">SAVE Posi&#231&#227o
3</button> </p>

  <p><button class="button button2" onclick="updateP4()" name="P4" value="valoresP4" type="submit">SAVE Posi&#231&#227o
4</button> </p>

 <p><button class="button button3" name="RUN" value="valoresP1234" type="submit">RUN Posições</button> </p>
 <p><button       class="button       button4"       onclick="Reset()"       name="RESET"       value="valoresP1234"       type="submit">RESET
Posições</button></p>
   </div>
</form>
```

```
   </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
   <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin" target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


<script>
var slider1 = document.getElementById("myRange1");
var output1 = document.getElementById("demo1");
    slider1.value = getCookie("demoSW","--- ");
output1.innerHTML = slider1.value;
slider1.oninput = function() {output1.innerHTML = this.value;}

var slider2 = document.getElementById("myRange2");
var output2 = document.getElementById("demo2");
    slider2.value = getCookie("demoSS","--- ");
output2.innerHTML = slider2.value;
slider2.oninput = function() {output2.innerHTML = this.value;}

var slider3 = document.getElementById("myRange3");
var output3 = document.getElementById("demo3");
    slider3.value = getCookie("demoSE","--- ");
output3.innerHTML = slider3.value;
slider3.oninput = function() {output3.innerHTML = this.value;}

var slider4 = document.getElementById("myRange4");
var output4 = document.getElementById("demo4");
    slider4.value = getCookie("demoSEV","--- ");
output4.innerHTML = slider4.value;
slider4.oninput = function() {output4.innerHTML = this.value;}

var slider5 = document.getElementById("myRange5");
var output5 = document.getElementById("demo5");
   slider5.value = getCookie("demoSWR","--- ");
output5.innerHTML = slider5.value;
slider5.oninput = function() {output5.innerHTML = this.value;}


document.getElementById("demoW").innerHTML = "W:";
document.getElementById("demoSW").innerHTML = getCookie("demoSW","--- ");
document.getElementById("demoS").innerHTML = "S:";
document.getElementById("demoSS").innerHTML = getCookie("demoSS","--- ");
document.getElementById("demoE").innerHTML = "E:";
document.getElementById("demoSE").innerHTML = getCookie("demoSE","--- ");
document.getElementById("demoEV").innerHTML = "EV:";
document.getElementById("demoSEV").innerHTML = getCookie("demoSEV","--- ");
document.getElementById("demoWR").innerHTML = "WR:";
document.getElementById("demoSWR").innerHTML = getCookie("demoSWR","--- ");
document.getElementById("demoG").innerHTML = "G:";
document.getElementById("demoSG").innerHTML = getCookie("demoSG","--- ");


document.getElementById("demoW1").innerHTML = "P1  W1:";
document.getElementById("demoSW1").innerHTML = getCookie("demoSW1","--- ");
document.getElementById("demoS1").innerHTML = "S1:";
document.getElementById("demoSS1").innerHTML = getCookie("demoSS1","--- ");
document.getElementById("demoE1").innerHTML = "E1:";
document.getElementById("demoSE1").innerHTML = getCookie("demoSE1","--- ");
document.getElementById("demoEV1").innerHTML = "EV1:";
document.getElementById("demoSEV1").innerHTML = getCookie("demoSEV1","--- ");
document.getElementById("demoWR1").innerHTML = "WR1:";
document.getElementById("demoSWR1").innerHTML = getCookie("demoSWR1","--- ");
document.getElementById("demoG1").innerHTML = "G1:";
document.getElementById("demoSG1").innerHTML = getCookie("demoSG1","--- ");

document.getElementById("demoW2").innerHTML = "P2  W2:";
document.getElementById("demoSW2").innerHTML = getCookie("demoSW2","--- ");
```

```
document.getElementById("demoS2").innerHTML = "S2:";
document.getElementById("demoSS2").innerHTML = getCookie("demoSS2","--- ");
document.getElementById("demoE2").innerHTML = "E2:";
document.getElementById("demoSE2").innerHTML = getCookie("demoSE2","--- ");
document.getElementById("demoEV2").innerHTML = "EV2:";
document.getElementById("demoSEV2").innerHTML = getCookie("demoSEV2","--- ");
document.getElementById("demoWR2").innerHTML = "WR2:";
document.getElementById("demoSWR2").innerHTML = getCookie("demoSWR2","--- ");
document.getElementById("demoG2").innerHTML = "G2:";
document.getElementById("demoSG2").innerHTML = getCookie("demoSG2","--- ");

document.getElementById("demoW3").innerHTML = "P3  W3:";
document.getElementById("demoSW3").innerHTML = getCookie("demoSW3","--- ");
document.getElementById("demoS3").innerHTML = "S3:";
document.getElementById("demoSS3").innerHTML = getCookie("demoSS3","--- ");
document.getElementById("demoE3").innerHTML = "E3:";
document.getElementById("demoSE3").innerHTML = getCookie("demoSE3","--- ");
document.getElementById("demoEV3").innerHTML = "EV3:";
document.getElementById("demoSEV3").innerHTML = getCookie("demoSEV3","--- ");
document.getElementById("demoWR3").innerHTML = "WR3:";
document.getElementById("demoSWR3").innerHTML = getCookie("demoSWR3","--- ");
document.getElementById("demoG3").innerHTML = "G3:";
document.getElementById("demoSG3").innerHTML = getCookie("demoSG3","--- ");

document.getElementById("demoW4").innerHTML = "P4  W4:";
document.getElementById("demoSW4").innerHTML = getCookie("demoSW4","--- ");
document.getElementById("demoS4").innerHTML = "S4:";
document.getElementById("demoSS4").innerHTML = getCookie("demoSS4","--- ");
document.getElementById("demoE4").innerHTML = "E4:";
document.getElementById("demoSE4").innerHTML = getCookie("demoSE4","--- ");
document.getElementById("demoEV4").innerHTML = "EV4:";
document.getElementById("demoSEV4").innerHTML = getCookie("demoSEV4","--- ");
document.getElementById("demoWR4").innerHTML = "WR4:";
document.getElementById("demoSWR4").innerHTML = getCookie("demoSWR4","--- ");
document.getElementById("demoG4").innerHTML = "G4:";
document.getElementById("demoSG4").innerHTML = getCookie("demoSG4","--- ");

 function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

function delCookie(cname) {
  document.cookie = cname + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
}

function getCookie(cname,defaultValue) {
  var name = cname + "=";
  var decodedCookie = decodeURIComponent(document.cookie);
  var ca = decodedCookie.split(';');
  for(var i = 0; i <ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return defaultValue;
}

function showEnviar()
{
  document.getElementById("demoW").innerHTML = "W:";
  var w = document.getElementById("myRange1").value;
  document.getElementById("demoSW").innerHTML = w;
  setCookie("demoSW", w, 1);

  document.getElementById("demoS").innerHTML = "S:";
  var s = document.getElementById("myRange2").value;
  document.getElementById("demoSS").innerHTML = s;
  setCookie("demoSS", s, 1);
```

```
 document.getElementById("demoE").innerHTML = "E:";
 var e = document.getElementById("myRange3").value;
 document.getElementById("demoSE").innerHTML = e;
  setCookie("demoSE", e, 1);

 document.getElementById("demoEV").innerHTML = "EV:";
 var ev = document.getElementById("myRange4").value;
 document.getElementById("demoSEV").innerHTML = ev;
  setCookie("demoSEV", ev, 1);

 document.getElementById("demoWR").innerHTML = "WR:";
 var wr = document.getElementById("myRange5").value;
 document.getElementById("demoSWR").innerHTML = wr;
  setCookie("demoSWR", wr, 1);

 document.getElementById("demoG").innerHTML = "G:";
 var g = document.getElementById("switch").checked;
 if (g ==true)
 {
  g ="Fechada";
  document.getElementById("demoSG").innerHTML = g;
 }
 if (g ==false)
 {
  g ="Aberta";
  document.getElementById("demoSG").innerHTML = g;
 }
  setCookie("demoSG", g, 1);



}
function updateP1()
{
 document.getElementById("demoW1").innerHTML = "P1  W1:";
 var w1 = document.getElementById("myRange1").value;
 document.getElementById("demoSW1").innerHTML = w1;
  setCookie("demoSW1",w1,1);

 document.getElementById("demoS1").innerHTML = "S1:";
 var s1 = document.getElementById("myRange2").value;
 document.getElementById("demoSS1").innerHTML = s1;
  setCookie("demoSS1",s1,1);

 document.getElementById("demoE1").innerHTML = "E1:";
 var e1 = document.getElementById("myRange3").value;
 document.getElementById("demoSE1").innerHTML = e1;
  setCookie("demoSE1",e1,1);

 document.getElementById("demoEV1").innerHTML = "EV1:";
 var ev1 = document.getElementById("myRange4").value;
 document.getElementById("demoSEV1").innerHTML = ev1;
  setCookie("demoSEV1",ev1,1);

 document.getElementById("demoWR1").innerHTML = "WR1:";
 var wr1 = document.getElementById("myRange5").value;
 document.getElementById("demoSWR1").innerHTML = wr1;
  setCookie("demoSWR1",wr1,1);

 document.getElementById("demoG1").innerHTML = "G1:";
 var g1 = document.getElementById("switch").checked;
 if (g1 ==true)
 {
  g1 ="196";
  document.getElementById("demoSG1").innerHTML = g1;
 }
 if (g1 ==false)
 {
  g1 ="151";
  document.getElementById("demoSG1").innerHTML = g1;
 }
 setCookie("demoSG1",g1,1);
}
function updateP2()
{
 document.getElementById("demoW2").innerHTML = "P2  W2:";
```

```
  var w2 = document.getElementById("myRange1").value;
  document.getElementById("demoSW2").innerHTML = w2;
    setCookie("demoSW2",w2,1);

  document.getElementById("demoS2").innerHTML = "S2:";
  var s2 = document.getElementById("myRange2").value;
  document.getElementById("demoSS2").innerHTML = s2;
    setCookie("demoSS2",s2,1);

  document.getElementById("demoE2").innerHTML = "E2:";
  var e2 = document.getElementById("myRange3").value;
  document.getElementById("demoSE2").innerHTML = e2;
    setCookie("demoSE2",e2,1);

  document.getElementById("demoEV2").innerHTML = "EV2:";
  var ev2 = document.getElementById("myRange4").value;
  document.getElementById("demoSEV2").innerHTML = ev2;
    setCookie("demoSEV2",ev2,1);

  document.getElementById("demoWR2").innerHTML = "WR2:";
  var wr2 = document.getElementById("myRange5").value;
  document.getElementById("demoSWR2").innerHTML = wr2;
    setCookie("demoSWR2",wr2,1);

  document.getElementById("demoG2").innerHTML = "G2:";
  var g2 = document.getElementById("switch").checked;
  if (g2 ==true)
  {
    g2 ="196";
    document.getElementById("demoSG2").innerHTML = g2;
  }
  if (g2 ==false)
  {
    g2 ="151";
    document.getElementById("demoSG2").innerHTML = g2;
  }
  setCookie("demoSG2",g2,1);
}
function updateP3()
{
  document.getElementById("demoW3").innerHTML = "P3  W3:";
  var w3 = document.getElementById("myRange1").value;
  document.getElementById("demoSW3").innerHTML = w3;
    setCookie("demoSW3",w3,1);

  document.getElementById("demoS3").innerHTML = "S3:";
  var s3 = document.getElementById("myRange2").value;
  document.getElementById("demoSS3").innerHTML = s3;
    setCookie("demoSS3",s3,1);

  document.getElementById("demoE3").innerHTML = "E3:";
  var e3 = document.getElementById("myRange3").value;
  document.getElementById("demoSE3").innerHTML = e3;
    setCookie("demoSE3",e3,1);

  document.getElementById("demoEV3").innerHTML = "EV3:";
  var ev3 = document.getElementById("myRange4").value;
  document.getElementById("demoSEV3").innerHTML = ev3;
    setCookie("demoSEV3",ev3,1);

  document.getElementById("demoWR3").innerHTML = "WR3:";
  var wr3 = document.getElementById("myRange5").value;
  document.getElementById("demoSWR3").innerHTML = wr3;
    setCookie("demoSWR3",wr3,1);

  document.getElementById("demoG3").innerHTML = "G3:";
  var g3 = document.getElementById("switch").checked;
  if (g3 ==true)
  {
    g3 ="196";
    document.getElementById("demoSG3").innerHTML = g3;
  }
  if (g3 ==false)
  {
    g3 ="151";
    document.getElementById("demoSG3").innerHTML = g3;
```

```
  }
  setCookie("demoSG3",g3,1);
}
function updateP4()
{
  document.getElementById("demoW4").innerHTML = "P4  W4:";
  var w4 = document.getElementById("myRange1").value;
  document.getElementById("demoSW4").innerHTML = w4;
    setCookie("demoSW4",w4,1);

  document.getElementById("demoS4").innerHTML = "S4:";
  var s4 = document.getElementById("myRange2").value;
  document.getElementById("demoSS4").innerHTML = s4;
    setCookie("demoSS4",s4,1);

  document.getElementById("demoE4").innerHTML = "E4:";
  var e4 = document.getElementById("myRange3").value;
  document.getElementById("demoSE4").innerHTML = e4;
    setCookie("demoSE4",e4,1);

  document.getElementById("demoEV4").innerHTML = "EV4:";
  var ev4 = document.getElementById("myRange4").value;
  document.getElementById("demoSEV4").innerHTML = ev4;
    setCookie("demoSEV4",ev4,1);

  document.getElementById("demoWR4").innerHTML = "WR4:";
  var wr4 = document.getElementById("myRange5").value;
  document.getElementById("demoSWR4").innerHTML = wr4;
    setCookie("demoSWR4",wr4,1);

  document.getElementById("demoG4").innerHTML = "G4:";
  var g4 = document.getElementById("switch").checked;
  if (g4 ==true)
  {
    g4 ="196";
    document.getElementById("demoSG4").innerHTML = g4;
  }
  if (g4 ==false)
  {
    g4 ="151";
    document.getElementById("demoSG4").innerHTML = g4;
  }
  setCookie("demoSG4",g4,1);
}

function Reset()
{
  document.getElementById("demoW1").innerHTML = "P1  W1:";
  var w1 = document.getElementById("myRange1").value;
  document.getElementById("demoSW1").innerHTML = "--- ";
    delCookie("demoSW1");

  document.getElementById("demoS1").innerHTML = "S1:";
  var s1 = document.getElementById("myRange2").value;
  document.getElementById("demoSS1").innerHTML = "--- ";
    delCookie("demoSS1");

  document.getElementById("demoE1").innerHTML = "E1:";
  var e1 = document.getElementById("myRange3").value;
  document.getElementById("demoSE1").innerHTML = "--- ";
    delCookie("demoSE1");

  document.getElementById("demoEV1").innerHTML = "EV1:";
  var ev1 = document.getElementById("myRange4").value;
  document.getElementById("demoSEV1").innerHTML = "--- ";
    delCookie("demoSEV1");

  document.getElementById("demoWR1").innerHTML = "WR1:";
  var wr1 = document.getElementById("myRange5").value;
  document.getElementById("demoSWR1").innerHTML = "--- ";
    delCookie("demoSWR1");

  document.getElementById("demoG1").innerHTML = "G1:";
  var g1 = document.getElementById("switch").value;
  document.getElementById("demoSG1").innerHTML = "--- ";
    delCookie("demoSG1");
```

```
document.getElementById("demoW2").innerHTML = "P2  W2:";
var w2 = document.getElementById("myRange1").value;
document.getElementById("demoSW2").innerHTML ="--- ";
  delCookie("demoSW2");

document.getElementById("demoS2").innerHTML = "S2:";
var s2 = document.getElementById("myRange2").value;
document.getElementById("demoSS2").innerHTML = "--- ";
  delCookie("demoSS2");

document.getElementById("demoE2").innerHTML = "E2:";
var e2 = document.getElementById("myRange3").value;
document.getElementById("demoSE2").innerHTML = "--- ";
  delCookie("demoSE2");

document.getElementById("demoEV2").innerHTML = "EV2:";
var ev2 = document.getElementById("myRange4").value;
document.getElementById("demoSEV2").innerHTML = "--- ";
  delCookie("demoSEV2");

document.getElementById("demoWR2").innerHTML = "WR2:";
var wr2 = document.getElementById("myRange5").value;
document.getElementById("demoSWR2").innerHTML = "--- ";
  delCookie("demoSWR2");

document.getElementById("demoG2").innerHTML = "G2:";
var g2 = document.getElementById("switch").value;
document.getElementById("demoSG2").innerHTML = "--- ";
  delCookie("demoSG2");


document.getElementById("demoW3").innerHTML = "P3  W3:";
var w3 = document.getElementById("myRange1").value;
document.getElementById("demoSW3").innerHTML = "--- ";
  delCookie("demoSW3");


document.getElementById("demoS3").innerHTML = "S3:";
var s3 = document.getElementById("myRange2").value;
document.getElementById("demoSS3").innerHTML = "--- ";
  delCookie("demoSS3");

document.getElementById("demoE3").innerHTML = "E3:";
var e3 = document.getElementById("myRange3").value;
document.getElementById("demoSE3").innerHTML = "--- ";
  delCookie("demoSE3");

document.getElementById("demoEV3").innerHTML = "EV3:";
var ev3 = document.getElementById("myRange4").value;
document.getElementById("demoSEV3").innerHTML = "--- ";
  delCookie("demoSEV3");

document.getElementById("demoWR3").innerHTML = "WR3:";
var wr3 = document.getElementById("myRange5").value;
document.getElementById("demoSWR3").innerHTML = "--- ";
  delCookie("demoSWR3");

document.getElementById("demoG3").innerHTML = "G3:";
var g3 = document.getElementById("switch").value;
document.getElementById("demoSG3").innerHTML = "--- ";
  delCookie("demoSG3");


document.getElementById("demoW4").innerHTML = "P4  W4:";
var w4 = document.getElementById("myRange1").value;
document.getElementById("demoSW4").innerHTML = "--- ";
   delCookie("demoSW4");

document.getElementById("demoS4").innerHTML = "S4:";
var s4 = document.getElementById("myRange2").value;
document.getElementById("demoSS4").innerHTML = "--- ";
   delCookie("demoSS4");

document.getElementById("demoE4").innerHTML = "E4:";
```

```
  var e4 = document.getElementById("myRange3").value;
  document.getElementById("demoSE4").innerHTML = "--- ";
    delCookie("demoSE4");

  document.getElementById("demoEV4").innerHTML = "EV4:";
  var ev4 = document.getElementById("myRange4").value;
  document.getElementById("demoSEV4").innerHTML = "--- ";
    delCookie("demoSEV4");

  document.getElementById("demoWR4").innerHTML = "WR4:";
  var wr4 = document.getElementById("myRange5").value;
  document.getElementById("demoSWR4").innerHTML = "--- ";
    delCookie("demoSWR4");

  document.getElementById("demoG4").innerHTML = "G4:";
  var g4 = document.getElementById("switch").value;
  document.getElementById("demoSG4").innerHTML = "--- ";
    delCookie("demoSG4");
}


</script>

</body>
</html>
)=====";


//*************************************************************************************************************
**************************************************************************
const char P4M_InstrucoesM[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Mentor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
```

```
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/">Voltar</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

  <h2 style="text-align:center">Instruções Mentor</h2>
  <div class="a">
  <br>
  Deslize os sliders, correspondentes aos 5 eixos do Mentor, que pretender, até ao valor desejado e escolha se quer
  abrir ou fechar a garra terminal do braço robótico. Quando definir os valores que pretende, pressione o botão ENVIAR! para movimentar o Mentor.</p>
  O botão, ENVIAR!, realiza uma Movimentação do Mentor. Pode ver por baixo do botão os valores enviados.</p>
  O botão, Posição Inicial, faz o Mentor retornar á sua posição de origem.</p>
  Pode utilizar os botões, SAVE Posição 1, SAVE Posição 2, Save Posição 3 e SAVE Posição 4 para guardar até 4 poses do braço robótico. O botão RUN Posições, leva o Mentor a movimentar-se de acordo com as posições que gravou anteriormente.</p>
  O botão, RESET Posições, apaga as 4 posições gravadas.</p>
  <br><br><br>

  </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
 <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin" target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)=====";


//*************************************************************************************************************
*****************************************************************************
const char P5M_SobreM[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Mentor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
```

```
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}

 /* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/">Voltar</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

  <h2 style="text-align:center">Sobre o Mentor</h2>
  <div class="a">
  <br>
  O braço robótico Mentor, é constituído por 5 juntas, 5 elos de ligação e uma garra terminal fixa.</p>
  Com a presente Interface, é possível controlar remotamente o Mentor, com total capacidade de movimento em cada eixo.</p>
  É possível guardar e/ou modificar até 4 posições distintas e efetuar uma sequência corrida dessas 4 poses.</p>
  <br><br><br><br><br>

  </div>
</div>

<div class="column side">
</div>
</div>

<div class="footer">
  <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin"
target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)======";
```

B - IRGBR: InterfaceDobotMagician.

O código desenvolvido para o Dobot Magician é composto por 8 separadores.

① InterfaceD

```
#define SERIAL_TX_BUFFER_SIZE 256
#define SERIAL_RX_BUFFER_SIZE 256
#define _SS_MAX_RX_BUFF 256
#include <Arduino.h>                  // Comunicação AP
#include "CommandHandler.h"           // Comunicação Série
CommandHandler cmdHdl(" ", '\n');     // Comunicação Série  default are delim="," and term=';'
#include <WiFi.h>                     // Comunicação Wi-Fi
#include <WiFiClient.h>               // Comunicação Wi-Fi
#include <WebServer.h>                // Comunicação Wi-Fi
#include "htmlsD.h"                   // Comunicação Wi-Fi
#include <esp_wpa2.h>                 // Comunicação Wi-Fi
WebServer server(80);                 // Comunicação Wi-Fi
#include "BluetoothSerial.h"          // Comunicação Bluetooth
BluetoothSerial ESP_BT;               // Comunicação Bluetooth

void setup(void){
  Serial.begin(115200);               // Monitor Série Serial.begin(115200,8,none,1);
  Serial1.begin(115200);              // Comunicação RX TX GND | Serial1 PINS | Serial1.begin(115200,8,none,1);
  Serial.println("\n\n_____________________________________________________________________________"); delay(100);
  Serial.println("[ADAFRUIT HUZZAH32 - ESP32 FEATHER] André Mira [Início -> Botão Reset!]\n");
  PosicaoHome();                      // Posição Inicial do Dobot
  startSERIE();                       // Comunicação Série
  startBLT();                         // Comunicação Bluetooth

  startWIFIcasa();                    // Comunicação Wi-Fi Casa
  //startWIFIisel();                  // Comunicação Wi-Fi Isel
  //startAP();                        // Comunicação AP
  startWebServer();                   // Comunicação Wi-Fi WebServer
  Serial.println("_____________________________________________________________________________\n");delay(100);
  Serial.println("Comunicação Série (Dobot)");
  Serial.println("Comandos disponíveis: H, P, R, A e IP.\n\n");
}

void loop(void){
  cmdHdl.processSerial(Serial);       // Comunicação Série
  server.handleClient();              // Comunicação Wi-Fi
  blueTooth();                        // Comunicação Bluetooth
}
```

② CommandHandler.cpp

```
/**
 * CommandHandler - A Wiring/Arduino library to tokenize and parse commands
 * received in different forms, serial, string, or char.
 *
 * Copyright (C) 2015 Cronin Group http://www.chem.gla.ac.uk/cronin/
 * Copyright (C) 2012 Stefan Rado
 * Copyright (C) 2011 Steven Cogswell <steven.cogswell@gmail.com>
 *                    http://husks.wordpress.com
 *
 * Version 20151029
 *
 * This library is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this library.  If not, see <http://www.gnu.org/licenses/>.
 */
```

```
#include "CommandHandler.h"

/**
 * Constructor allowing to change default delim and term
 * Example: SerialCommand sCmd(" ", ';');
 * Default are COMMANDHANDLER_DEFAULT_DELIM and COMMANDHANDLER_DEFAULT_TERM
 */
CommandHandler::CommandHandler(const char *newdelim, char newterm)
  : commandList(NULL),
    commandCount(0),
    relayList(NULL),
    relayCount(0),
    defaultHandler(NULL),
    pt2defaultHandlerObject(NULL),
    wrapper_defaultHandler(NULL),
    term(newterm),           // asssign new terminator for commands
    last(NULL),
    delim(newdelim) // assign new delimitor
{
  inCmdStream = &Serial;
  outCmdStream = &Serial;

  commandHeader = String("");
  commandDecimal = 2;

  clearBuffer();
}

/**
 * Adds a "command" and a handler function to the list of available commands.
 * This is used for matching a found token in the buffer, and gives the pointer
 * to the handler function to deal with it.
 */
void CommandHandler::addCommand(const char *command, void (*function)()) {
  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Adding command (");
    Serial.print(commandCount);
    Serial.print("): ");
    Serial.println(command);
  #endif

  commandList = (CommandHandlerCallback *) realloc(commandList, (commandCount + 1) * sizeof(CommandHandlerCallback));
  strncpy(commandList[commandCount].command, command, COMMANDHANDLER_MAXCOMMANDLENGTH);
  commandList[commandCount].function = function;
  commandCount++;
}

/**
 * Adds a "command" and a handler function to the list of available relay.
 * This is used for matching a found token in the buffer, and gives the pointer
 * to the handler function to deal with the remaining of the command
 */
void CommandHandler::addRelay(const char *command, void (*function)(const char *, void*), void* pt2Object) {
  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Adding relay (");
    Serial.print(relayCount);
    Serial.print("): ");
    Serial.println(command);
  #endif

  relayList = (RelayHandlerCallback *) realloc(relayList, (relayCount + 1) * sizeof(RelayHandlerCallback));
  strncpy(relayList[relayCount].command, command, COMMANDHANDLER_MAXCOMMANDLENGTH);
  relayList[relayCount].pt2Object = pt2Object;
  relayList[relayCount].function = function;
  relayCount++;
}

/**
 * This sets up a handler to be called in the event that the receveived command string
 * isn't in the list of commands.
 */
void CommandHandler::setDefaultHandler(void (*function)(const char *)) {
  defaultHandler = function;
}

void CommandHandler::setDefaultHandler(void (*function)(const char *, void*), void* pt2Object) {
```

```
  pt2defaultHandlerObject = pt2Object;
  wrapper_defaultHandler = function;
}

/**
 * Assign the default serial
 */
void CommandHandler::setInCmdSerial(Stream &inStream) {
  inCmdStream = &inStream;
}

/**
 * Check the default Serial
 */
void CommandHandler::processSerial() {
  processSerial(*inCmdStream);
}

/**
 * This checks the Serial stream for characters, and assembles them into a buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processSerial(Stream &inStream) {
  while (inStream.available() > 0) {
    char inChar = inStream.read();   // Read single available character, there may be more waiting
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("Serial: ");
      Serial.println(inChar);   // Echo back to serial stream
    #endif
    processChar(inChar);
  }
}

/**
 * This iterate on a String char by char, and push them into a buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processString(const char *inString) {
  for (int i = 0; i < strlen(inString); i++){
    char inChar = inString[i];
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("String: ");
      Serial.println(inChar);   // Echo back to serial stream
    #endif
    processChar(inChar);
  }
}

/**
 * This add a characters to the buffer, and analyse the buffer.
 * When the terminator character (default COMMANDHANDLER_DEFAULT_TERM) is seen, it starts parsing the
 * buffer for a prefix command, and calls handlers setup by addCommand() member
 */
void CommandHandler::processChar(char inChar) {
  if (inChar == term) {     // Check for the terminator (default '\r') meaning end of command
    #ifdef COMMANDHANDLER_DEBUG
      Serial.print("Received: ");
      Serial.println(buffer);
    #endif

    char *command = strtok_r(buffer, delim, &last);   // Search for command at start of buffer
    if (command != NULL) {
      boolean matched = false;
      // searching in commands
      for (int i = 0; i < commandCount; i++) {
        #ifdef COMMANDHANDLER_DEBUG
          Serial.print("Comparing [");
          Serial.print(command);
          Serial.print("] to [");
          Serial.print(commandList[i].command);
          Serial.println("]");
        #endif

        // Compare the found command against the list of known commands for a match
```

```
      if (strncmp(command, commandList[i].command, COMMANDHANDLER_MAXCOMMANDLENGTH) == 0) {
        #ifdef COMMANDHANDLER_DEBUG
          Serial.print("Matched Command: ");
          Serial.println(command);
        #endif

        // Execute the stored handler function for the command
        (*commandList[i].function)();
        matched = true;
        break;
      }
    }
    // searching in relays
    for (int i = 0; i < relayCount; i++) {
      #ifdef COMMANDHANDLER_DEBUG
        Serial.print("Comparing [");
        Serial.print(command);
        Serial.print("] to [");
        Serial.print(relayList[i].command);
        Serial.println("]");
      #endif

      // Compare the found command against the relay list of known commands for a match
      if (strncmp(command, relayList[i].command, COMMANDHANDLER_MAXCOMMANDLENGTH) == 0) {
        #ifdef COMMANDHANDLER_DEBUG
          Serial.print("Matched Relay: ");
          Serial.println(command);
        #endif

        // Execute the stored handler function for the command
        (*relayList[i].function)(remaining(), relayList[i].pt2Object);
        matched = true;
        break;
      }
    }
    if (!matched){
      if (defaultHandler != NULL) {
        (*defaultHandler)(command);
      } else if (pt2defaultHandlerObject != NULL) {
        (*wrapper_defaultHandler)(command, pt2defaultHandlerObject);
      }
    }
  }
  clearBuffer();
 }
 else if (isprint(inChar)) {     // Only printable characters into the buffer
  if (bufPos < COMMANDHANDLER_BUFFER) {
    buffer[bufPos] = inChar;  // Put character into buffer
    buffer[bufPos+1] = STRING_NULL_TERM;      // Null terminate
    bufPos++;
  } else {
    #ifdef COMMANDHANDLER_DEBUG
      Serial.println("Line buffer is full - increase COMMANDHANDLER_BUFFER");
    #endif
  }
 }
}

/*
 * Clear the input buffer.
 */
void CommandHandler::clearBuffer() {
  buffer[0] = STRING_NULL_TERM;
  bufPos = 0;
}

/**
 * Retrieve the next token ("word" or "argument") from the command buffer.
 * Returns NULL if no more tokens exist.
 */
char *CommandHandler::next() {
  return strtok_r(NULL, delim, &last);
}

/**
 * Returns char* of the remaining of the command buffer (for getting arguments to commands).
```

```
 * Returns NULL if no more tokens exist.
 */
char *CommandHandler::remaining() {

  //reinit the remains char
  remains[0] = STRING_NULL_TERM;

  char str_term[2];
  str_term[0] = term;
  str_term[1] = STRING_NULL_TERM;

   // Search for the remaining up to next term
  char *command = strtok_r(NULL, str_term, &last);

  // forge term in string format
  strcpy(remains, command);
  strcat(remains, str_term);

  // clear the buffer now, we emptied the current command
  // the remaining is might be given to another handler
  // or it might used by the same commandHandler instance
  // hence the buffer should be emptied now
  clearBuffer();

  return remains;
}



/**************************************
 * Helpers to read args and cast them into specific type, strongly inspired by CmdMessenger
 **************************************/

/**
 * Read the next argument as int16
 */
int CommandHandler::readIntArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return atoi(arg);
  }
  argOk = false;
  return 0;
}

/**
 * Read the next argument as int32
 */
long CommandHandler::readLongArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return atol(arg);
  }
  argOk = false;
  return 0L; // 'L' to force the constant into a long data format
}

/**
 * Read the next argument as bool
 */
bool CommandHandler::readBoolArg() {
  return (readIntArg() != 0) ? true : false;
}

/**
 * Read the next argument as float
 */
float CommandHandler::readFloatArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
```

```
    return strtod(arg, NULL);
  }
  argOk = false;
  return 0;
}

/**
 * Read the next argument as double
 */
double CommandHandler::readDoubleArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return strtod(arg, NULL);
  }
  argOk = false;
  return 0;
}

/**
 * Read next argument as string.
 */
char* CommandHandler::readStringArg() {
  char *arg;
  arg = next();
  if (arg != NULL) {
    argOk = true;
    return arg;
  }
  argOk = false;
  return STRING_NULL_TERM;
}

/**
 * Compare the next argument with a string
 */
bool CommandHandler::compareStringArg(const char *stringToCompare) {
  char *arg;
  arg = next();
  if (arg != NULL) {
    return (strcmp(stringToCompare, arg) == 0) ? true : false;
  }
  return false;
}

/****************************************
 * Forging and sending output commands
 ****************************************/


/**
 * Set an header for the output command
 */
void CommandHandler::setCmdHeader(const char *cmdHeader, bool addDelim) {

  commandHeader = String(cmdHeader);

  if (addDelim == true) {
    commandHeader = commandHeader + String(delim);
  }

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out Command Header is now ");
    Serial.println(commandHeader);
  #endif
}

void CommandHandler::initCmd() {

  commandString = commandHeader;

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
```

```
}

void CommandHandler::addCmdDelim() {

  commandString = commandString + String(delim);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}
//
void CommandHandler::addCmdTerm() {

  commandString = commandString + String(term);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdBool(bool value) {

  commandString = commandString + String(value);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdInt(int value) {

  commandString = commandString + String(value, DEC);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdLong(long value) {

  commandString = commandString + String(value, DEC);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}


void CommandHandler::setCmdDecimal(byte decimal) {
  commandDecimal = decimal;
}

void CommandHandler::addCmdFloat(double value) {
  addCmdFloat(value, commandDecimal);
}

void CommandHandler::addCmdFloat(float value, byte decimal) {


  commandString = commandString + String(value, decimal);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdDouble(double value) {
  addCmdDouble(value, commandDecimal);
}
```

```
void CommandHandler::addCmdDouble(double value, byte decimal) {

  commandString = commandString + String(value, decimal);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif
}

void CommandHandler::addCmdString(const char *value) {

  commandString = commandString + String(value);

  #ifdef COMMANDHANDLER_DEBUG
    Serial.print("Out command is now ");
    Serial.println(commandString);
  #endif

}

char* CommandHandler::getOutCmd() {

  commandString.toCharArray(command, COMMANDHANDLER_BUFFER + 1);

  return command;
}

void CommandHandler::setOutCmdSerial(Stream &outStream) {
  outCmdStream = &outStream;
}

void CommandHandler::sendCmdSerial() {
  sendCmdSerial(*outCmdStream);
}

void CommandHandler::sendCmdSerial(Stream &outStream) {
  outStream.print(commandString);
}
```

③ CommandHandler.h

```
/**
 * CommandHandler - A Wiring/Arduino library to tokenize and parse commands
 * received in different forms, serial, string, or char.
 *
 * Copyright (C) 2015 Cronin Group http://www.chem.gla.ac.uk/cronin/
 * Copyright (C) 2012 Stefan Rado
 * Copyright (C) 2011 Steven Cogswell <steven.cogswell@gmail.com>
 *                    http://husks.wordpress.com
 *
 * Version 20151029
 *
 * This library is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this library.  If not, see <http://www.gnu.org/licenses/>.
 */

#ifndef CommandHandler_h
#define CommandHandler_h

#if defined(WIRING) && WIRING >= 100
  #include <Wiring.h>
#elif defined(ARDUINO) && ARDUINO >= 100
  #include <Arduino.h>
```

```
#else
  #include <WProgram.h>
#endif
#include <string.h>

// Size of the input buffer in bytes (maximum length of one command plus arguments)
#define COMMANDHANDLER_BUFFER 64
// Maximum length of a command excluding the terminating null
#define COMMANDHANDLER_MAXCOMMANDLENGTH 8
// Default delimitor and terminator
#define COMMANDHANDLER_DEFAULT_DELIM ","
#define COMMANDHANDLER_DEFAULT_TERM ';'
// The null term for string
#define STRING_NULL_TERM '\0'

// Uncomment the next line to run the library in debug mode (verbose messages)
// #define COMMANDHANDLER_DEBUG


class CommandHandler {
  public:
    CommandHandler(const char *newdelim = COMMANDHANDLER_DEFAULT_DELIM, const char newterm = COMMANDHANDLER_DEFAULT_TERM);   // Constructor
    void addCommand(const char *command, void(*function)());  // Add a command to the processing dictionary.
    void addRelay(const char *command, void (*function)(const char *, void*), void* pt2Object = NULL);  // Add a command to the relay dictionary. Such relay are given the remaining of the command. pt2Object is the reference to the instance associated with the callback, it will be given as the second argument of the callback function, default is NULL
    void setDefaultHandler(void (*function)(const char *));   // A handler to call when no valid command received.
    void setDefaultHandler(void (*function)(const char *, void*), void* pt2Object);    // A handler to call when no valid command received.

    void setInCmdSerial(Stream &inStream); // define to which serial to send the read commands
    void processSerial();  // Process what on the in stream
    void processSerial(Stream &inStream);  // Process what on the designated stream
    void processString(const char *inString); // Process a String
    void processChar(char inChar); //Process a char
    void clearBuffer();   // Clears the input buffer.
    char *remaining();        // Returns pointer to remaining of the command buffer (for getting arguments to commands).
    char *next();         // Returns pointer to next token found in command buffer (for getting arguments to commands).

    // helpers to cast next into different types
    bool argOk; // this variable is set after the below function are run, it tell you if thing went well
    bool readBoolArg();
    int readIntArg();
    long readLongArg();
    float readFloatArg();
    double readDoubleArg();
    char *readStringArg();
    bool compareStringArg(const char *stringToCompare);

    //helpers to create a message
    void setCmdHeader(const char *cmdHeader, bool addDelim = true); // setting a char to be added at the start of each out message (default "")
    void initCmd(); // initialize the command buffer  to build next message to be sent

    void clearCmd(); // clear the output command
    void addCmdDelim();
    void addCmdTerm();

    void addCmdBool(bool value);
    void addCmdInt(int value);
    void addCmdLong(long value);

    void setCmdDecimal(byte decimal);
    void addCmdFloat(double value);
    void addCmdFloat(float value, byte decimal);
    void addCmdDouble(double value);
    void addCmdDouble(double value, byte decimal);

    void addCmdString(const char *value);

    char* getOutCmd(); // get pointer to command buffer

    void setOutCmdSerial(Stream &outStream); // define to which serial to send the out commands
    void sendCmdSerial(); //send current command thought the Stream
    void sendCmdSerial(Stream &outStream); //send current command thought the Stream
```

```
 private:

   // Command/handler dictionary
   struct CommandHandlerCallback {
     char command[COMMANDHANDLER_MAXCOMMANDLENGTH + 1];
     void (*function)();
   };                              // Data structure to hold Command/Handler function key-value pairs
   CommandHandlerCallback *commandList;   // Actual definition for command/handler array
   byte commandCount;

   // Relay/handler dictionary
   struct RelayHandlerCallback {
     char command[COMMANDHANDLER_MAXCOMMANDLENGTH + 1];
     void* pt2Object;
     void (*function)(const char *, void*);
   };                              // Data structure to hold Relay/Handler function key-value pairs
   RelayHandlerCallback *relayList;   // Actual definition for Relay/handler array
   byte relayCount;

   // Pointer to the default handler function
   void (*defaultHandler)(const char *);
   void* pt2defaultHandlerObject;
   void (*wrapper_defaultHandler)(const char *, void*);

   const char *delim; // null-terminated list of character to be used as delimeters for tokenizing (default " ")
   char term;     // Character that signals end of command (default '\n')

   char buffer[COMMANDHANDLER_BUFFER + 1]; // Buffer of stored characters while waiting for terminator character
   byte bufPos;                    // Current position in the buffer
   char *last;                     // State variable used by strtok_r during processing

   char remains[COMMANDHANDLER_BUFFER + 1]; // Buffer of stored characters to pass to a relay function

   char command[COMMANDHANDLER_BUFFER + 1];
   String commandString; // Out Command
   String commandHeader; // header for out command
   byte commandDecimal;


   // in and out default strem
   Stream *inCmdStream;
   Stream *outCmdStream;
};

#endif //CommandHandler_h
```

④ ComunicacaoBLT

```
char incoming;
byte startedText = 1;
int txtIdx = 0;
const int BUFFER_SIZE = 40;
char rcvBuffer[BUFFER_SIZE];
void startBLT(){
  ESP_BT.begin("ESP_32_BLT_RobotControl");
}
void blueTooth(){
  if (ESP_BT.available())
  {
    incoming = ESP_BT.read();
    switch (incoming)
    {
      case '\n':
        if (startedText)
        {
          startedText = 1;
          rcvBuffer[txtIdx] = '\n';
          for (int i = txtIdx+1; i < BUFFER_SIZE; i++) {
            rcvBuffer[i] = '\0';
          }
          //Serial.print("\n");
          Serial.print(" BT RCV ->");
          Serial.print(rcvBuffer);
```

```
        cmdHdl.processString(rcvBuffer);
        txtIdx = 0;
       }
       else
       {
        startedText = 1;
        txtIdx = 0;
       }
       break;
     default:
       if (startedText)
       {
        rcvBuffer[txtIdx++] = incoming;          // passar 1 a frente
       }
       if (txtIdx >= BUFFER_SIZE)
       {
        Serial.println("Buffer overflow bt");
        txtIdx = 0;
       }
       break;
   }
  }
}
```

⑤ ComunicaçaoSerie

```
void startSERIE()
{
 cmdHdl.addCommand("H", PosicaoHome);       // Posição Home
 cmdHdl.addCommand("P", PosicaoDesejada);   // Posicao X Y Z r
 cmdHdl.addCommand("IP", avisoWiFiBLT);     // Escrever IP por bluetooth

 cmdHdl.addCommand("R", GETPOSITION);       // Receber coordenadas do dobot
 cmdHdl.addCommand("A", GETALARM);          // Receber código de alarme
 //cmdHdl.addCommand("CA", CLEARALARM);           // Limpar alarme

 cmdHdl.setDefaultHandler(unrecognized);    // Handler for command that isn't matched
}

void unrecognized(const char *command)
{
 Serial.println("Comando Inválido! Insira um novo comando.\n");
 ESP_BT.println("Comando Inválido! Insira um novo comando.\n");
}
```

⑥ ComunicacaoWIFI

```
#define EAP_IDENTITY "****Email_Aluno_ISEL****"
#define EAP_PASSWORD "********PassWord********"

String x;       String y;       String z;       String r;
String xP1;     String yP1;     String zP1;     String rP1;
String xP2;     String yP2;     String zP2;     String rP2;
String xP3;     String yP3;     String zP3;     String rP3;
String xP4;     String yP4;     String zP4;     String rP4;


void startWIFIcasa()
{
 const char* ssid = "********RedeCasa****";
 const char* password = "****PassWord****";

 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);

 Serial.print("Connect to "); Serial.println(ssid);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 } Serial.println("");

 Serial.print("IP address: ");  Serial.println(WiFi.localIP());
}
```

```
void startWIFIisel()
{
 const char* ssid =  "eduroam";

 Serial.println();
 Serial.printf("A estabelecer ligação com a rede: %s", ssid);
 WiFi.disconnect( true );
 WiFi.mode( WIFI_STA );
 // Provide identity
 esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
 // Provide username (identity and username are the same)
 esp_wifi_sta_wpa2_ent_set_username((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
 // Provide password
 esp_wifi_sta_wpa2_ent_set_password((uint8_t *)EAP_PASSWORD, strlen(EAP_PASSWORD));
 // Set config settings to default
 esp_wpa2_config_t config = WPA2_CONFIG_INIT_DEFAULT();
 // Set config settings to enable function
 esp_wifi_sta_wpa2_ent_enable(&config);
 WiFi.begin(ssid );
 Serial.println();
 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500);
  Serial.print(".. ");
 }
 Serial.println();
 Serial.printf("Ligação estabelecida com sucesso com a rede: %s", ssid);
 Serial.println();
 Serial.print  ("IP Address:                         ");
 Serial.println(WiFi.localIP());
}

void startAP()
{
 const char* ssid     = "ESP32-Access-Point";
 const char* password = "Dobot";

 Serial.print("Setting AP (Access Point)…");
 // Remove the password parameter, if you want the AP (Access Point) to be open
 WiFi.softAP(ssid, password);
 IPAddress IP = WiFi.softAPIP(); Serial.print("AP IP address: "); Serial.println(IP);
 // Print ESP Local IP Address
 Serial.print("WiFi Local IP:"); Serial.println(WiFi.localIP());
}

void avisoWiFiBLT()
{
 if (WiFi.status() != WL_CONNECTED)
 {
  Serial.println("µC não está conectado! Conecte o µC.\n");
  ESP_BT.println("µC não está conectado! Conecte o µC.\n");    // escrever na app Serial Bluetooth
 }
 else
 {
  Serial.print("Endereço IP: "); Serial.print(WiFi.localIP()); Serial.println("\n");
  ESP_BT.print("Endereço IP: "); ESP_BT.print(WiFi.localIP()); ESP_BT.println("\n"); // escrever na app Serial Bluetooth o ip
 }
}

void startWebServer()
{
 server.on("/", handleRoot);
 server.on("/login", handleLogin);
 server.on("/sobre", []() {
  String s = P2D_Sobre;
  server.send(200, "text/html", s);
 });
 server.on("/instrucoesD", []() {
  String s = P4D_InstrucoesD;
  server.send(200, "text/html", s);
 });
 server.on("/sobreD", []() {
  String s = P5D_SobreD;
  server.send(200, "text/html", s);
 });
```

```
  //here the list of headers to be recorded
  const char * headerkeys[] = {"User-Agent", "Cookie"} ;
  size_t headerkeyssize = sizeof(headerkeys) / sizeof(char*);

  //ask server to track these headers
  server.collectHeaders(headerkeys, headerkeyssize);

  server.begin();
  //Serial.println("HTTP server started"); Serial.println("");
}

bool is_authentified()
{
  //Serial.println("Enter is_authentified");
  if (server.hasHeader("Cookie"))
  {
    //Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    //Serial.println(cookie);
    if (cookie.indexOf("ESPSESSIONID=1") != -1)
    {
      //Serial.println("Authentification Successful");Serial.println("");
      return true;
    }
  }
  //Serial.println("Authentification Failed"); Serial.println("");
  return false;
}

//login page, also called for disconnect
void handleLogin()
{
  String msg;
  if (server.hasHeader("Cookie")) {
    //Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    //Serial.println(cookie);
  }
  if (server.hasArg("DISCONNECT"))
  {
    //Serial.println("Disconnection");
    server.sendHeader("Location", "/login");
    server.sendHeader("Cache-Control", "no-cache");
    server.sendHeader("Set-Cookie", "ESPSESSIONID=0");
    server.send(301);
    return;
  }
  if (server.hasArg("USERNAME") && server.hasArg("PASSWORD"))
  {
    if (server.arg("USERNAME") == "Dobot" &&  server.arg("PASSWORD") == "Dobot")
    {
      server.sendHeader("Location", "/");
      server.sendHeader("Cache-Control", "no-cache");
      server.sendHeader("Set-Cookie", "ESPSESSIONID=1");
      server.send(301);
      Serial.println("Log in bem-sucedido.\n");
      ESP_BT.println("Log in bem-sucedido.\n");
      return;
    }
    msg = "Wrong username/password! try again.";
    Serial.println("Log in falhado.\n");
    ESP_BT.println("Log in falhado.\n");
  }
  String s = P1D_Entrada;              // Read HTML contents
  server.send(200, "text/html", s); // Send web page
}

//root page can be accessed only if authentification is ok
void handleRoot()
{
  //Serial.println("Enter handleRoot");
  String header;
  if (!is_authentified())
  {
    server.sendHeader("Location", "/login");
```

```
  server.sendHeader("Cache-Control", "no-cache");
  server.send(301);
  return;
 }

 String a = P3D_ControloD; server.send(200, "text/html", a);

 if (server.hasArg("enviarValores"))
 {
  x = server.arg("sliderX");     //Serial. print("X:"); Serial.println(x);
  y = server.arg("sliderY");     //Serial. print("Y:"); Serial.println(y);
  z = server.arg("sliderZ");     //Serial. print("Z:"); Serial.println(z);
  r = server.arg("sliderR");     //Serial. print("R:"); Serial.println(r);

  Serial.println(" Wi-FI ->Movimentação");
  ESP_BT.println(" Wi-FI ->Movimentação");

  char buffer[60];
  sprintf(buffer, "P %s %s %s %s\n\0", x, y, z, r); cmdHdl.processString(buffer); //Serial.println(buffer);
 }

 if (server.hasArg("PI"))
 {
  Serial.println(" Wi-FI ->Home");
  ESP_BT.println(" Wi-FI ->Home");
  PosicaoHome();
 }

 if (server.hasArg("P1"))
 {
  xP1 = server.arg("sliderX");
  yP1 = server.arg("sliderY");
  zP1 = server.arg("sliderZ");
  rP1 = server.arg("sliderR");
  Serial.println(" Wi-Fi ->Posição 1");
  ESP_BT.println(" Wi-Fi ->Posição 1");
  Serial.print("X1:"); Serial.print(xP1); Serial.print(" Y1:"); Serial.print(yP1); Serial.print(" Z1:"); Serial.print(zP1); Serial.print("
R1:"); Serial.print(rP1); Serial.println("\n");
  ESP_BT.print("X1:");     ESP_BT.print(xP1);     ESP_BT.print("     Y1:");     ESP_BT.print(yP1);     ESP_BT.print("     Z1:");
ESP_BT.print(zP1); ESP_BT.print(" R1:"); ESP_BT.print(rP1); ESP_BT.println("\n");
 }

 if (server.hasArg("P2"))
 {
  xP2 = server.arg("sliderX");
  yP2 = server.arg("sliderY");
  zP2 = server.arg("sliderZ");
  rP2 = server.arg("sliderR");
  Serial.println(" Wi-Fi ->Posição 2");
  ESP_BT.println(" Wi-Fi ->Posição 2");
  Serial.print("X2:"); Serial.print(xP2); Serial.print(" Y2:"); Serial.print(yP2); Serial. print(" Z2:"); Serial.print(zP2); Serial.print("
R2:"); Serial.print(rP2); Serial.println("\n");
  ESP_BT.print("X2:");     ESP_BT.print(xP2);     ESP_BT.print("     Y2:");     ESP_BT.print(yP2);     ESP_BT.     print("     Z2:");
ESP_BT.print(zP2); ESP_BT.print(" R2:"); ESP_BT.print(rP2); ESP_BT.println("\n");
 }

 if (server.hasArg("P3"))
 {
  xP3 = server.arg("sliderX");
  yP3 = server.arg("sliderY");
  zP3 = server.arg("sliderZ");
  rP3 = server.arg("sliderR");
  Serial.println(" Wi-Fi ->Posição 3");
  ESP_BT.println(" Wi-Fi ->Posição 3");
  Serial. print("X3:"); Serial.print(xP3); Serial.print(" Y3:"); Serial.print(yP3); Serial.print(" Z3:"); Serial.print(zP3); Serial.print("
R3:"); Serial.print(rP3); Serial.println("\n");
  ESP_BT.     print("X3:");     ESP_BT.print(xP3);     ESP_BT.print("     Y3:");     ESP_BT.print(yP3);     ESP_BT.print("     Z3:");
ESP_BT.print(zP3); ESP_BT.print(" R3:"); ESP_BT.print(rP3); ESP_BT.println("\n");
 }

 if (server.hasArg("P4"))
 {
  xP4 = server.arg("sliderX");
  yP4 = server.arg("sliderY");
  zP4 = server.arg("sliderZ");
  rP4 = server.arg("sliderR");
```

```
   Serial.println(" Wi-Fi ->Posição 4");
   ESP_BT.println(" Wi-Fi ->Posição 4");
   Serial. print("X4:"); Serial.print(xP4); Serial.print(" Y4:"); Serial.print(yP4); Serial.print(" Z4:"); Serial.print(zP4); Serial.print(" R4:"); Serial.print(rP4); Serial.println("\n");
   ESP_BT. print("X4:"); ESP_BT.print(xP4); ESP_BT.print(" Y4:"); ESP_BT.print(yP4); ESP_BT.print(" Z4:"); ESP_BT.print(zP4); ESP_BT.print(" R4:"); ESP_BT.print(rP4); ESP_BT.println("\n");
  }

  if (server.hasArg("RUN"))
  {
   Serial.println(" Wi-Fi ->Robô Dobot - A correr Posições.");
   ESP_BT.println(" Wi-Fi ->Robô Dobot - A correr Posições.");

   Serial.print("X1:"); Serial.print(xP1); Serial.print(" Y1:"); Serial.print(yP1); Serial.print(" Z1:"); Serial.print(zP1); Serial.print(" R1:"); Serial.println(rP1);
   ESP_BT.print("X1:"); ESP_BT.print(xP1); ESP_BT.print(" Y1:"); ESP_BT.print(yP1); ESP_BT.print(" Z1:"); ESP_BT.print(zP1); ESP_BT.print(" R1:"); ESP_BT.println(rP1);

   Serial.print("X2:"); Serial.print(xP2); Serial.print(" Y2:"); Serial.print(yP2); Serial.print(" Z2:"); Serial.print(zP2); Serial.print(" R2:"); Serial.println(rP2);
   ESP_BT.print("X2:"); ESP_BT.print(xP2); ESP_BT.print(" Y2:"); ESP_BT.print(yP2); ESP_BT.print(" Z2:"); ESP_BT.print(zP2); ESP_BT.print(" R2:"); ESP_BT.println(rP2);

   Serial.print("X3:"); Serial.print(xP3); Serial.print(" Y3:"); Serial.print(yP3); Serial.print(" Z3:"); Serial.print(zP3); Serial.print(" R3:"); Serial.println(rP3);
   ESP_BT.print("X3:"); ESP_BT.print(xP3); ESP_BT.print(" Y3:"); ESP_BT.print(yP3); ESP_BT.print(" Z3:"); ESP_BT.print(zP3); ESP_BT.print(" R3:"); ESP_BT.println(rP3);

   Serial.print("X4:"); Serial.print(xP4); Serial. print(" Y4:"); Serial.print(yP4); Serial.print(" Z4:"); Serial.print(zP4); Serial.print(" R4:"); Serial.print(rP4); Serial.println("\n");
   ESP_BT.print("X4:"); ESP_BT.print(xP4); ESP_BT. print(" Y4:"); ESP_BT.print(yP4); ESP_BT.print(" Z4:"); ESP_BT.print(zP4); ESP_BT.print(" R4:"); ESP_BT.print(rP4); ESP_BT.println("\n");

   char buffer[60];
   sprintf(buffer, "P %s %s %s %s\n\0", xP1, yP1, zP1, rP1); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "P %s %s %s %s\n\0", xP2, yP2, zP2, rP2); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "P %s %s %s %s\n\0", xP3, yP3, zP3, rP3); //Serial.println(buffer);
   cmdHdl.processString(buffer); delay(1000);
   sprintf(buffer, "P %s %s %s %s\n\0", xP4, yP4, zP4, rP4); //Serial.println(buffer);
   cmdHdl.processString(buffer);
  }

  if (server.hasArg("RESET"))
  {
   Serial.println(" Wi-Fi ->Reset Posições\n");
   ESP_BT.println(" Wi-Fi ->Reset Posições\n");
  }

}
```

⑦ Dobot

```
//_____________________________________________________________________________________________________
_____________HOME Position
void PosicaoHome()
{
  Serial.println("Robô Dobot - Posição Inicial");
  Serial.println("X:260.5 Y:0 Z:-8.5 R:0\n");
  ESP_BT.println("Robô Dobot - Posição Inicial");
  ESP_BT.println("X:260.5 Y:0 Z:-8.5 R:0\n");

  // Limpar o canal para nova leitura
  while (Serial1.available() > 0 ) {
    Serial1.read();
  }

  // Enviar comando Home
  int cmd[10] = {170, 170, 6, 31, 3, 0, 0, 0, 0, 222};
  for (int i = 0; i < 10; i++)
  {
    Serial1.write(char(cmd[i]));
    //Serial.println((cmd[i]));
```

```
  }

  //Serial.println("");
}


//_____________________________________________________________________________________________________
_____________Get Pose
void GETPOSITION()
{
  Serial.println("Robô Dobot - Leitura Coordenadas");
  ESP_BT.println("Robô Dobot - Leitura Coordenadas");

  // Limpar o canal para nova leitura
  while (Serial1.available() > 0 ) {
    Serial1.read();
  }

  // Enviar comando ler posição
  int cmdE[6] = {170, 170, 2, 10, 0, 246};
  for (int i = 0; i < 6; i++)
  {
    Serial1.write(char(cmdE[i]));
    //Serial.println((cmdE[i]));
  }

  // Receber comando ler posição
  //Serial.println("'\nRecepção Comando GetPose");
  //int h1P, h2P, lenP, idP, ctrlP, Px1, Px2, Px3, Px4, Py1, Py2, Py3, Py4, Pz1, Pz2, Pz3, Pz4, Pr1, Pr2, Pr3, Pr4, ba1, ba2, ba3, ba4,
ra1, ra2, ra3, ra4, fa1, fa2, fa3, fa4, ea1, ea2, ea3, ea4, PChecksum;
  //int cmdR[38] = {h1P, h2P, lenP, idP, ctrlP, Px1, Px2, Px3, Px4, Py1, Py2, Py3, Py4, Pz1, Pz2, Pz3, Pz4, Pr1, Pr2, Pr3, Pr4, ba1,
ba2, ba3, ba4, ra1, ra2, ra3, ra4, fa1, fa2, fa3, fa4, ea1, ea2, ea3, ea4, PChecksum};

#define PX1_IDX 2
#define PX2_IDX 3
#define PX3_IDX 4
#define PX4_IDX 5

#define PY1_IDX 6
#define PY2_IDX 7
#define PY3_IDX 8
#define PY4_IDX 9

#define PZ1_IDX 10
#define PZ2_IDX 11
#define PZ3_IDX 12
#define PZ4_IDX 13

#define PR1_IDX 14
#define PR2_IDX 15
#define PR3_IDX 16
#define PR4_IDX 17

  int cmdR[40], len;
  int dat, count = 2;
//    while (count != 0)
//    {
//      while (Serial1.available() <= 0);                                    //< ----------------------------------------------------------------------
------- -
//      dat = Serial1.read();                                 // Header1 Header2
//      count = (dat == 0xAA) ? count - 1 : 2;
//    }
  len = Serial1.read();                                 // len
  //Serial.print("Size len:"); Serial.println(len);
  for (int i = 0; i < len; i++)
  {
    //while (Serial1.available() <= 0);                                     //< ----------------------------------------------------------------------
--------- -
    cmdR[i] = Serial1.read();                                 // ID ctrl [2]...[len]
    //Serial.println((cmdR[i]));
  }
  //while (Serial1.available() <= 0);                                     //< ----------------------------------------------------------------------
--------- -
  int checksum = Serial1.read();                                 // checksum
  //Serial.print("Checksum:"); Serial.println(checksum);
  delay(100);
```

```
 // Converter a posição recebida de Px1234 Py1234 Pz1234 Pr1234
 // (Hex -> Float)
 //Serial.println("");

 char numX[60];
 sprintf(numX, "%d%d%d%d\0", cmdR[PX1_IDX], cmdR[PX2_IDX], cmdR[PX3_IDX], cmdR[PX4_IDX]);
 //Serial.print("Px DEC: "); Serial.println(numX);
 sprintf(numX, "%02X%02X%02X%02X\0", cmdR[PX1_IDX], cmdR[PX2_IDX], cmdR[PX3_IDX], cmdR[PX4_IDX]);
 //Serial.print("Px HEX: "); Serial.println(numX);
 //Serial.print("Px Float: "); Serial.println(hex_to_float(numX)); Serial.println("");

 char numY[60];
 sprintf(numY, "%d%d%d%d\0", cmdR[PY1_IDX], cmdR[PY2_IDX], cmdR[PY3_IDX], cmdR[PY4_IDX]);
 //Serial.print("Py DEC: "); Serial.println(numY);
 sprintf(numY, "%02X%02X%02X%02X\0", cmdR[PY1_IDX], cmdR[PY2_IDX], cmdR[PY3_IDX], cmdR[PY4_IDX]);
 //Serial.print("Py HEX: "); Serial.println(numY);
 //Serial.print("Py Float: "); Serial.println(hex_to_float(numY)); Serial.println("");

 char numZ[60];
 sprintf(numZ, "%d%d%d%d\0", cmdR[PZ1_IDX], cmdR[PZ2_IDX], cmdR[PZ3_IDX], cmdR[PZ4_IDX]);
 //Serial.print("Pz DEC: "); Serial.println(numZ);
 sprintf(numZ, "%02X%02X%02X%02X\0", cmdR[PZ1_IDX], cmdR[PZ2_IDX], cmdR[PZ3_IDX], cmdR[PZ4_IDX]);
 //Serial.print("Pz HEX: "); Serial.println(numZ);
 //Serial.print("Pz Float: "); Serial.println(hex_to_float(numZ)); Serial.println("");

 char numR[60];
 sprintf(numR, "%d%d%d%d\0", cmdR[PR1_IDX], cmdR[PR2_IDX], cmdR[PR3_IDX], cmdR[PR4_IDX]);
 //Serial.print("Pr DEC: "); Serial.println(numR);
 sprintf(numR, "%02X%02X%02X%02X\0", cmdR[PR1_IDX], cmdR[PR2_IDX], cmdR[PR3_IDX], cmdR[PR4_IDX]);
 //Serial.print("Pr HEX: "); Serial.println(numR);
 //Serial.print("Pr Float: "); Serial.println(hex_to_float(numR)); Serial.println("");

 float PoseX = hex_to_float(numX); //Serial.println(PoseX);
 float PoseY = hex_to_float(numY); //Serial.println(PoseY);
 float PoseZ = hex_to_float(numZ); //Serial.println(PoseZ);
 float PoseR = hex_to_float(numR); //Serial.println(PoseR);

 //  char Pose[60];
 //  sprintf(Pose, "P %.5f %.5f %.5f %.5f\0", PoseX, PoseY, PoseZ, PoseR);
 //  Serial.println(Pose);

  Serial.print("X:"); Serial.print(PoseX); Serial.print(" Y:"); Serial.print(PoseY); Serial.print(" Z:"); Serial.print(PoseZ); Serial.print("
R:"); Serial.print(PoseR); Serial.println("\n");
  ESP_BT.print("X:");     ESP_BT.print(PoseX);     ESP_BT.print("     Y:");     ESP_BT.print(PoseY);     ESP_BT.print("     Z:");
ESP_BT.print(PoseZ); ESP_BT.print(" R:"); ESP_BT.print(PoseR); ESP_BT.println("\n");
}

float hex_to_float(char* hexValue)
{
  // hex swap litle endiand
  char hexNumber1[9];
  hexNumber1[8] = '\0';

  hexNumber1[0] = hexValue[6];
  hexNumber1[1] = hexValue[7];

  hexNumber1[2] = hexValue[4];
  hexNumber1[3] = hexValue[5];

  hexNumber1[4] = hexValue[2];
  hexNumber1[5] = hexValue[3];

  hexNumber1[6] = hexValue[0];
  hexNumber1[7] = hexValue[1];

  // hex to float conversion
  int num;
  sscanf(hexNumber1, "%8x", &num);  // assuming you checked input
  return *((float*)&num);
}
//_____________________________________________________________________________________________________________
_____________FIM Get Pose
```

```
//_____________________________________________________________________________________________
_____________Get Alarm
void GETALARM()
{
  Serial.println("Robô Dobot - Leitura Alarme");
  ESP_BT.println("Robô Dobot - Leitura Alarme");

  // Limpar o canal para nova leitura
  while (Serial1.available() > 0 ) {
    Serial1.read();
  }

  // Enviar comando Alarme
  //Serial.println("Envio Comando Alarme");
  //  int h1a, h2a, lena, ida, ctrla, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16, aChecksum;
  //  int cmdR[22] = {h1a, h2a, lena, ida, ctrla, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16, aChecksum};
  int cmdE[6] = {170, 170, 2, 20, 0, 236};
  for (int i = 0; i < 6; i++)
  {
    Serial1.write(char(cmdE[i]));
    //Serial.println((cmdE[i]));
  }

  // Receber comando Alarme opçao 1
  //  Serial.println("\nRecepção Comando Alarme");
  //  int cmdR[22];
  //  for (int i = 0; i < 22; i++)
  //  {
  //    //while (Serial1.available() <= 0);                          //<----------------------------------------------------------------------------
  //    cmdR[i] = Serial1.read();
  //    Serial.println((cmdR[i])); delay(100);
  //  }
  //
  //  Serial.println("Alarme: \n");
  //  ESP_BT.println("Alarme: \n");
  //  delay(100);

  // Receber comando Alarme opçao 2                          dat dat len cmdR[0]ID cmdR[1]Ctrl ... cmdR[len]  checksum
  //Serial.println("\nRecepção Comando Alarme");
  int cmdR[40], len;
  int dat, count = 2;
  int somaParameters = 0;
  //  while (count != 0)
  //  {
  //    while (Serial1.available() <= 0);                                  //< ----------------------------------------------------------------
--------- -
  //    dat = Serial1.read();                                 // Header1 Header2
  //    count = (dat == 0xAA) ? count - 1 : 2;
  //  }
  len = Serial1.read();
  //Serial.print("Size len:"); Serial.println(len);  // len
  for (int i = 0; i < len; i++)
  {
    //while (Serial1.available() <= 0);                                  //<----------------------------------------------------------------
-------
    cmdR[i] = Serial1.read();                                 // cmdR[0]ID cmdR[1]Ctrl ... cmdR[len]
    //Serial.println((cmdR[i]));
  }
  for (int i = 2; i < len; i++)
  {
    somaParameters = cmdR[i] + somaParameters;                          // cmdR[0]ID cmdR[1]Ctrl | somaParameters = cmdR[2] +
somaParameters | ... cmdR[len]
    //Serial.print("Soma:"); Serial.println(somaParameters);       // Soma
  }
  //while (Serial1.available() <= 0);                                  //<----------------------------------------------------------------
-----
  int checksum = Serial1.read();
  //Serial.print("Checksum:"); Serial.println(checksum);       // checksum
  delay(100);

  Serial.print("Alarme: "); Serial.print(somaParameters); Serial.println("\n");
  ESP_BT.print("Alarme: "); ESP_BT.print(somaParameters); ESP_BT.println("\n");

  if (somaParameters != 0)
  {
    PosicaoHome();
```

```
 }
 delay(100);
}



//_____________________________________________________________________________________________________
_____________Posição desejada
void PosicaoDesejada()    // P x y z r
{
 Serial.print("Robô Dobot\n");
 ESP_BT.print("Robô Dobot\n");

 // Limpar o canal para nova leitura
 while (Serial1.available() > 0 ) {
  Serial1.read();
 }

 int x1, x2, x3, x4, y1, y2, y3, y4, z1, z2, z3, z4, r1, r2, r3, r4, Checksum;
 int c = 0;

 float value1; value1 = cmdHdl.readFloatArg();
 float value2; value2 = cmdHdl.readFloatArg();
 float value3; value3 = cmdHdl.readFloatArg();
 float value4; value4 = cmdHdl.readFloatArg();

 //  if (cmdHdl.argOk)
 //  {
 //    char numP[60]; sprintf(numP, "P %.3f %.3f %.3f %.3f\0", value1, value2, value3, value4);
 //    Serial.print("Inicial - "); Serial.print(numP); Serial.println("\n");
 //    ESP_BT.print("Inicial - "); ESP_BT.print(numP); ESP_BT.println("\n");
 //  }

 //if (value1 == false || value2 == false || value3 == false || value4 == false)
 if (value1 < 160 || value1 > 250 || value2 < -150 || value2 > 150 || value3 < -50|| value3 > 75 || value4 < -50 || value4 > 50)
 {
  Serial.println("Comando P sem 4 argumentos! Repita.\n");
  ESP_BT.println("Comando P sem 4 argumentos! Repita.\n");
 }
 else
 {
  //_____________________________________________________________________________ X   160 e 250
  if (cmdHdl.argOk & (value1 >= 160 & value1 <= 250) )
  {
    //Serial.print("\n Float number1 was: "); Serial.println(value1);            // FLOAT NUMBER 1

    char hexNumber1[9];
    sprintf(hexNumber1, "%08x", *(unsigned int*)&value1);
    char dest1[20];
    // hex swap litle endiand
    dest1[8] = '\0';

    dest1[0] = hexNumber1[6];
    dest1[1] = hexNumber1[7];

    dest1[2] = hexNumber1[4];
    dest1[3] = hexNumber1[5];

    dest1[4] = hexNumber1[2];
    dest1[5] = hexNumber1[3];

    dest1[6] = hexNumber1[0];
    dest1[7] = hexNumber1[1];

    //Serial.print("Hex number1 is: "); Serial.println(dest1);                // HEX NUMBER 1

    char *CardNumber1 = dest1;
    byte CardNumberByte1[4];
    unsigned long number1 = strtoul( CardNumber1, nullptr, 16);
    for (int i = 3; i >= 0; i--)          // start with lowest byte of number
    {
     CardNumberByte1[i] = number1 & 0xFF;  // or: = byte( number);
     number1 >>= 8;                       // get next byte into position
    }
```

```
  x1 = CardNumberByte1[0];
  x2 = CardNumberByte1[1];
  x3 = CardNumberByte1[2];
  x4 = CardNumberByte1[3];
  //Serial.println("Dec number1 is: ");                          // DEC NUMBER 1
  //Serial.print("x1: "); Serial.println(x1);
  //Serial.print("x2: "); Serial.println(x2);
  //Serial.print("x3: "); Serial.println(x3);
  //Serial.print("x4: "); Serial.println(x4);
  //Serial.println("");
}
else
{
  Serial.println("Argumento 1 inválido!");
  ESP_BT.println("Argumento 1 inválido!");
  c = 1;
}


//_____________________________________________________________________________ Y  -150 a 150
if (cmdHdl.argOk & (value2 >= -150 & value2 <= 150) )
{
  //Serial.print(" Float number2 was: "); Serial.println(value2);            // FLOAT NUMBER 2

  char hexNumber2[9];
  sprintf(hexNumber2, "%08x", *(unsigned int*)&value2);
  char dest2[20];
  // hex swap litle endiand
  dest2[8] = '\0';

  dest2[0] = hexNumber2[6];
  dest2[1] = hexNumber2[7];

  dest2[2] = hexNumber2[4];
  dest2[3] = hexNumber2[5];

  dest2[4] = hexNumber2[2];
  dest2[5] = hexNumber2[3];

  dest2[6] = hexNumber2[0];
  dest2[7] = hexNumber2[1];

  //Serial.print("Hex number2 is: "); Serial.println(dest2);              // HEX NUMBER 2

  char *CardNumber2 = dest2;
  byte CardNumberByte2[4];
  unsigned long number2 = strtoul( CardNumber2, nullptr, 16);
  for (int i = 3; i >= 0; i--)          // start with lowest byte of number
  {
    CardNumberByte2[i] = number2 & 0xFF;  // or: = byte( number);
    number2 >>= 8;                      // get next byte into position
  }

  y1 = CardNumberByte2[0];
  y2 = CardNumberByte2[1];
  y3 = CardNumberByte2[2];
  y4 = CardNumberByte2[3];
  //Serial.println("Dec number2 is: ");                          // DEC NUMBER 2
  //Serial.print("y1: "); Serial.println(y1);
  //Serial.print("y2: "); Serial.println(y2);
  //Serial.print("y3: "); Serial.println(y3);
  //Serial.print("y4: "); Serial.println(y4);
  //Serial.println("");

  //    y = y1 + y2 + y3 + y4;
  //    Serial.println(y);
}
else
{
  Serial.println("Argumento 2 inválido!");
  ESP_BT.println("Argumento 2 inválido!");
  c = 1;
}


//_____________________________________________________________________________ Z   -50 a 75
```

```
if (cmdHdl.argOk & (value3 >= -50 & value3 <= 75) )
{
  //Serial.print(" Float number3 was: "); Serial.println(value3);              // FLOAT NUMBER 3

  char hexNumber3[9];
  sprintf(hexNumber3, "%08x", *(unsigned int*)&value3);
  char dest3[20];
  // hex swap litle endiand
  dest3[8] = '\0';

  dest3[0] = hexNumber3[6];
  dest3[1] = hexNumber3[7];

  dest3[2] = hexNumber3[4];
  dest3[3] = hexNumber3[5];

  dest3[4] = hexNumber3[2];
  dest3[5] = hexNumber3[3];

  dest3[6] = hexNumber3[0];
  dest3[7] = hexNumber3[1];

  //Serial.print("Hex number3 is: "); Serial.println(dest3);                   // HEX NUMBER 3

  char *CardNumber3 = dest3;
  byte CardNumberByte3[4];
  unsigned long number3 = strtoul( CardNumber3, nullptr, 16);
  for (int i = 3; i >= 0; i--)          // start with lowest byte of number
  {
   CardNumberByte3[i] = number3 & 0xFF;  // or: = byte( number);
   number3 >>= 8;                       // get next byte into position
  }

  z1 = CardNumberByte3[0];
  z2 = CardNumberByte3[1];
  z3 = CardNumberByte3[2];
  z4 = CardNumberByte3[3];
  //Serial.println("Dec number3 is: ");                                        // DEC NUMBER 3
  //Serial.print("z1: "); Serial.println(z1);
  //Serial.print("z2: "); Serial.println(z2);
  //Serial.print("z3: "); Serial.println(z3);
  //Serial.print("z4: "); Serial.println(z4);
  //Serial.println("");

  //    z = z1 + z2 + z3 + z4;
  //    Serial.println(z);
}
else
{
  Serial.println("Argumento 3 inválido!");
  ESP_BT.println("Argumento 3 inválido!");
  c = 1;
}


//_____________________________________________________________________________ r   -50 a 50
if (cmdHdl.argOk & (value4 >= -50 & value4 <= 50) )
{
  //Serial.print(" Float number4 was: "); Serial.println(value4);              // FLOAR NUMBER 4

  char hexNumber4[9];
  sprintf(hexNumber4, "%08x", *(unsigned int*)&value4);
  char dest4[20];
  // hex swap litle endiand
  dest4[8] = '\0';

  dest4[0] = hexNumber4[6];
  dest4[1] = hexNumber4[7];

  dest4[2] = hexNumber4[4];
  dest4[3] = hexNumber4[5];

  dest4[4] = hexNumber4[2];
  dest4[5] = hexNumber4[3];

  dest4[6] = hexNumber4[0];
```

```
    dest4[7] = hexNumber4[1];

    //Serial.print("Hex number4 is: "); Serial.println(dest4);                 // HEX NUMBER 4

    char *CardNumber4 = dest4;
    byte CardNumberByte4[4];
    unsigned long number4 = strtoul( CardNumber4, nullptr, 16);
    for (int i = 3; i >= 0; i--)          // start with lowest byte of number
    {
     CardNumberByte4[i] = number4 & 0xFF;  // or: = byte( number);
     number4 >>= 8;                       // get next byte into position
    }

    r1 = CardNumberByte4[0];
    r2 = CardNumberByte4[1];
    r3 = CardNumberByte4[2];
    r4 = CardNumberByte4[3];
    //Serial.println("Dec number4 is: ");                                  // DEC NUMBER 4
    //Serial.print("r1: "); Serial.println(r1);
    //Serial.print("r2: "); Serial.println(r2);
    //Serial.print("r3: "); Serial.println(r3);
    //Serial.print("r4: "); Serial.println(r4);
    //Serial.println("");

    //    r = r1 + r2 + r3 + r4;
    //    Serial.println(r);
   }
   else
   {
    Serial.println("Argumento 4 inválido!");
    ESP_BT.println("Argumento 4 inválido!");
    c = 1;
   }



   if (c < 1)
   {
    //_____________________________________________________________________________________ Checksum
    int somaChecksum1 = 84 + 3 + 1 +  x1 + x2 + x3 + x4 + y1 + y2 + y3 + y4 + z1 + z2 + z3 + z4 + r1 + r2 + r3 + r4;
    //Serial.print("Dec somaChecksum1: ");      Serial.println(somaChecksum1);                    // soma checksum DEC
    //Serial.print("Bin somaChecksum1: ");      Serial.println(somaChecksum1, BIN);               // checksum BIN
    //Serial.print("Bin somaChecksum1 8lsb: "); Serial.println(somaChecksum1 & 0x00FF, BIN);      // checksum BIN
8lsb
    //Serial.print("Dec somaChecksum1 8lsb: "); Serial.println(somaChecksum1 & 0x00FF, DEC);      // checksum
DEC 8lsb
    int lsbDEC = somaChecksum1 & 0x00FF; //Serial.println(lsbDEC);
    Checksum = 256 - lsbDEC;
    //Serial.print("Checksum: "); Serial.println(Checksum);                                     // 2's complement checksum DEC


    //_____________________________________________________________________________________ Comando ENVIADO
    //Serial.println("\nComando Enviado:");
    int comando[23] = {170, 170, 19, 84, 3, 1, x1, x2, x3, x4, y1, y2, y3, y4, z1, z2, z3, z4, r1, r2, r3, r4, Checksum};
    for (int i = 0; i < 23; i++)
    {
     delay(100);
     Serial1.write(char(comando[i]));
     //Serial.println((comando[i]));
    }
    //Serial.println("FIM Comando Enviado!");
   }
   else
   {
    char numP[60]; sprintf(numP, "Erro: P %.3f %.3f %.3f %.3f\0", value1, value2, value3, value4); delay(100);
    Serial.println(numP);
    ESP_BT.println(numP);
    Serial.println("Sem 4 argumentos válidos, repita o comando.\n");
    ESP_BT.println("Sem 4 argumentos válidos, repita o comando.\n");
    delay(100);
   }

   if (cmdHdl.argOk & (c < 1))                                                    // POSIÇÂO DESEJADA
   {
    char numP[60]; sprintf(numP, "P %.3f %.3f %.3f %.3f\0", value1, value2, value3, value4);
    Serial.print(numP); Serial.println("\n");
```

```
    ESP_BT.print(numP); ESP_BT.println("\n");
   }
  }
  //_____________________________________________________________________________ Comando RECEBIDO
  // X

}


//_____________________________________________________________________________________________________
_____________Clear Alarm
//void CLEARALARM()
//{
//  Serial.println("Robô Dobot - Apagar Alarme\n");
//  ESP_BT.println("Robô Dobot - Apagar Alarme\n");
//
//  // Limpar o canal para nova leitura
//  while (Serial1.available() > 0 ) {
//    Serial1.read();
//  }
//
//  // Enviar comando ler posição
//  Serial.println("Envio Comando ClearAlarm");
//  int cmdE[6] = {170, 170, 2, 21, 1, 234};
//  for (int i = 0; i < 6; i++)
//  {
//    Serial1.write(char(cmdE[i]));
//    Serial.println((cmdE[i]));
//  }
//
//  // Receber comando Alarme
//  Serial.println("\nRecepção Comando ClearAlarme");
//  int cmdR[22];
//  for (int i = 0; i < 22; i++)
//  {
//    while (Serial1.available() <= 0);
//    cmdR[i] = Serial1.read();
//    Serial.println((cmdR[i]));
//  }
//  Serial.println("FIM Recepção Comando ClearAlarme");
//  delay(100);
//  Serial.println("");
//}
```

⑧ htmlsD.h

```
//*************************************************************************************************************
*****************************************************************************
const char P1D_Entrada[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Dobot</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
```

```
/* Left and right column */
.column.side {width: 35%;}
/* Middle column */
.column.middle {width: 30%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


input[type=text], input[type=password] {width: 100%;padding: 12px 20px;margin: 8px 0;display: inline-block;border: 1px solid #ccc;box-sizing: border-box;}
button {background-color: #4CAF50;color: white;padding: 14px 20px;margin: 8px 0;border: none;cursor: pointer;width: 100%;}
button:hover {opacity: 0.8;}
.container {padding: 16px;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/sobre">Sobre</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

   <div class="container">

  <h2 style="text-align:center">Segurança de entrada</h2>
  <p>Para entrar, por favor introduza o nome de utilizador e a palavra-passe.</p>
  <br><br>
  <form action='/login' method='POST'>

     <label for="USERNAME"><b>Nome de Utilizador</b></label>
     <input type="text" placeholder="Introduza o nome de utilizador" name="USERNAME" >

     <label for="PASSWORD"><b>Palavra-Passe</b></label>
    <input type="password" placeholder="Introduza a palavra-passe" name="PASSWORD" >

    <button type="submit">Login</button>
  <br><br><br>
  </form>
  </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
  <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin" target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>
```

```
</body>
</html>
)=====";


//*************************************************************************************************************
*************************************************************************
const char P2D_Sobre[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Dobot</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}


/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/login">Voltar</a>
</div>


<div class="row">
<div class="column side">
</div>
```

```
<div class="column middle" style="text-align:left">

 <h2 style="text-align:center">Sobre a Interface</h2>
 <div class="a">
 <br>
 Para utilizar a Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos deverá efetuar Login, introduzindo o nome
 de utilizador e a palavra-passe corretos.</p>
 A Interface Remota Gen&#233rica, permite controlar remotamente qualquer braço rob&#243tico, a partir do
 protocolo de comunicação Wi-Fi.</p>
 A validação do Login, serve como segurança, para o acesso ao controlo do braço robótico.</p>
 <br><br><br><br><br>

 </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
 <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin"
target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)======";



//*************************************************************************************************************
*****************************************************************************
const char P3D_ControloD[] PROGMEM = R"======(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Dobot</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
ul.topnav                    {list-style-type: none;margin: 0;padding: 0;overflow: hidden; border: 1px solid #cccccc; background-color:
#f2f2f2;}
ul.topnav li                 {float: left; } /*border-right:1px solid white; border-bottom: 1px solid white;*/
ul.topnav li a               {display: block;color: black;text-align: center;padding: 14px 16px;text-decoration: none;}
ul.topnav li a:hover:not(.active) {background-color:  #ddd;  color: white;}
ul.topnav li a.active          {background-color: #4CAF50;}
ul.topnav li.right           {float: right;}
@media screen and (max-width: 600px) {ul.topnav li.right, ul.topnav li {float: none;}}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 25%;}
/* Middle column */
.column.middle {width: 50%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
```

```
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}


/* coluna do meio com 2 boxs */
.box {float: left; width: 50%; padding: 17px;}
.clearfix::after {content: "";clear: both;display: table;}

/* Botões */
.button {display: inline-block; padding: none; font-size: none; cursor: pointer; text-align: center; text-decoration: none; outline: none;
color: black; background-color: #F0F0F0; border: none; border-radius: 10px; box-shadow: 0 5px #999; width: 150px;}
.button1:hover {background-color: #008CBA;}
.button1:active { background-color: #008CBA; box-shadow: 0 3px #666; transform: translateY(4px);}
.button2:hover {background-color: #cc9900;}
.button2:active { background-color: #e7e7e7; box-shadow: 0 3px #666; transform: translateY(4px);}
.button3:hover {background-color: #4CAF50;}
.button3:active { background-color: #4CAF50; box-shadow: 0 3px #666; transform: translateY(4px);}
.button4:hover {background-color: #f44336;}
.button4:active { background-color: #f44336; box-shadow: 0 3px #666; transform: translateY(4px);}
.buttonE {display: inline-block; padding: 8px 22px; font-size: none; cursor: pointer; text-align: center; text-decoration: none; outline:
none; color: black; background-color: #F0F0F0; border: none; border-radius: 5px; box-shadow: 0 5px #999; width: 150px;}
.buttonE:hover {background-color: #33cc33;}
.buttonE:active { background-color: #e7e7e7; box-shadow: 0 3px #666; transform: translateY(4px);}
.slider:hover {opacity: 3;}

/* TV */
.centerTV { font-size: 14px; margin-left:auto; margin-right:auto;justify-content: center;align-items: center;height: 140px;width:
230px;border: 3px solid #e7e7e7; border-top-left-radius: 10px;border-top-right-radius: 10px;border-bottom-left-radius: 10px;border-
bottom-right-radius: 10px; border-style: ridge;}

 /* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<ul class="topnav">
  <li><a href="/instrucoesD">Instruções Dobot</a></li>
  <li><a href="/sobreD">Sobre Dobot</a></li>
  <li class="right"><a href="/login?DISCONNECT=YES">Sair Dobot</a></li>
</ul>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:center">
    <h2 >Controlo Dobot Magician</h2>
    <p>Use os sliders para ajustar as coordenadas.<br>Pode guardar e correr 4 poses.</p>

    <div class="centerTV" >
<p>
 <span id="demoX1"></span><span id="demoSX1"></span>
 <span id="demoY1"></span><span id="demoSY1"></span>
 <span id="demoZ1"></span><span id="demoSZ1"></span>
 <span id="demoR1"></span><span id="demoSR1"></span>
</p>
<p>
 <span id="demoX2"></span><span id="demoSX2"></span>
 <span id="demoY2"></span><span id="demoSY2"></span>
 <span id="demoZ2"></span><span id="demoSZ2"></span>
 <span id="demoR2"></span><span id="demoSR2"></span>
</p>
<p>
 <span id="demoX3"></span><span id="demoSX3"></span>
 <span id="demoY3"></span><span id="demoSY3"></span>
 <span id="demoZ3"></span><span id="demoSZ3"></span>
```

```
 <span id="demoR3"></span><span id="demoSR3"></span>
</p>
<p>
 <span id="demoX4"></span><span id="demoSX4"></span>
 <span id="demoY4"></span><span id="demoSY4"></span>
 <span id="demoZ4"></span><span id="demoSZ4"></span>
 <span id="demoR4"></span><span id="demoSR4"></span>
</p>
   </div>

  <div class="clearfix">

<form action="/" method="post">
   <div class="box" style="text-align:right">
  <p><div>X:  <span  id="demo1"></span><input  type="range"  min="160"  max="250"  class="slider"  id="myRange1"
name="sliderX"></div></p>
  <p><div>Y:  <span  id="demo2"></span><input  type="range"  min="-150"  max="150"  class="slider"  id="myRange2"
name="sliderY"></div></p>
  <p><div>Z:  <span  id="demo3"></span><input  type="range"  min="-50"  max="75"  class="slider"  id="myRange3"
name="sliderZ"></div></p>
  <p><div>R:  <span  id="demo4"></span><input  type="range"  min="-50"  max="50"  class="slider"  id="myRange4"
name="sliderR"></div></p>
<br>
  <p><button  class="button  buttonE"  onclick="showEnviar()"  style="vertical-align:middle"  type="submit"
name="enviarValores">ENVIAR!</button></p>
<p>
 <span id="demoX"></span><span id="demoSX"></span>
 <span id="demoY"></span><span id="demoSY"></span>
 <span id="demoZ"></span><span id="demoSZ"></span>
 <span id="demoR"></span><span id="demoSR"></span>
</p>
   </div>


    <div class="box" style="text-align:left">
   <p><button class="button button1" name="PI" value="valoresPI" type="submit">Posi&#231&#227o Inicial!</button></p>

  <p><button class="button button2" onclick="updateP1()" name="P1" value="valoresP1" type="submit">SAVE Posi&#231&#227o
1</button> </p>

  <p><button class="button button2" onclick="updateP2()" name="P2" value="valoresP2" type="submit">SAVE Posi&#231&#227o
2</button> </p>

  <p><button class="button button2" onclick="updateP3()" name="P3" value="valoresP3" type="submit">SAVE Posi&#231&#227o
3</button> </p>

  <p><button class="button button2" onclick="updateP4()" name="P4" value="valoresP4" type="submit">SAVE Posi&#231&#227o
4</button> </p>

  <p><button class="button button3" name="RUN" value="valoresP1234" type="submit">RUN Posições</button> </p>
   <p><button  class="button  button4"  onclick="Reset()"  name="RESET"  value="valoresP1234"  type="submit">RESET
Posições</button></p>

    </div>
</form>

    </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
   <h3><a>&#169  2020  Andr&#233  Mira  </a><a  href="https://www.linkedin.com/in/andré-mira"  class="fa  fa-linkedin"
target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>

<script>
var slider1 = document.getElementById("myRange1");
var output1 = document.getElementById("demo1");
    slider1.value = getCookie("demoSX","--- ");
output1.innerHTML = slider1.value;
slider1.oninput = function() {output1.innerHTML = this.value;}
```

```
var slider2 = document.getElementById("myRange2");
var output2 = document.getElementById("demo2");
   slider2.value = getCookie("demoSY","--- ");
output2.innerHTML = slider2.value;
slider2.oninput = function() {output2.innerHTML = this.value;}

var slider3 = document.getElementById("myRange3");
var output3 = document.getElementById("demo3");
   slider3.value = getCookie("demoSZ","--- ");
output3.innerHTML = slider3.value;
slider3.oninput = function() {output3.innerHTML = this.value;}

var slider4 = document.getElementById("myRange4");
var output4 = document.getElementById("demo4");
slider4.value = getCookie("demoSR","--- ");
output4.innerHTML = slider4.value;
slider4.oninput = function() {output4.innerHTML = this.value;}


document.getElementById("demoX1").innerHTML = "P1  X1:";
   document.getElementById("demoSX1").innerHTML = getCookie("demoSX1","--- ");
document.getElementById("demoY1").innerHTML = "Y1:";
   document.getElementById("demoSY1").innerHTML = getCookie("demoSY1","--- ");
document.getElementById("demoZ1").innerHTML = "Z1:";
   document.getElementById("demoSZ1").innerHTML = getCookie("demoSZ1","--- ");
document.getElementById("demoR1").innerHTML = "R1:";
   document.getElementById("demoSR1").innerHTML = getCookie("demoSR1","--- ");

document.getElementById("demoX2").innerHTML = "P2  X2:";
   document.getElementById("demoSX2").innerHTML = getCookie("demoSX2","--- ");
document.getElementById("demoY2").innerHTML = "Y2:";
   document.getElementById("demoSY2").innerHTML = getCookie("demoSY2","--- ");
document.getElementById("demoZ2").innerHTML = "Z2:";
   document.getElementById("demoSZ2").innerHTML = getCookie("demoSZ2","--- ");
document.getElementById("demoR2").innerHTML = "R2:";
   document.getElementById("demoSR2").innerHTML = getCookie("demoSR2","--- ");

document.getElementById("demoX3").innerHTML = "P3  X3:";
   document.getElementById("demoSX3").innerHTML = getCookie("demoSX3","--- ");
document.getElementById("demoY3").innerHTML = "Y3:";
   document.getElementById("demoSY3").innerHTML = getCookie("demoSY3","--- ");
document.getElementById("demoZ3").innerHTML = "Z3:";
   document.getElementById("demoSZ3").innerHTML = getCookie("demoSZ3","--- ");
document.getElementById("demoR3").innerHTML = "R3:";
   document.getElementById("demoSR3").innerHTML = getCookie("demoSR3","--- ");

document.getElementById("demoX4").innerHTML = "P4  X4:";
   document.getElementById("demoSX4").innerHTML = getCookie("demoSX4","--- ");
document.getElementById("demoY4").innerHTML = "Y4:";
   document.getElementById("demoSY4").innerHTML = getCookie("demoSY4","--- ");
document.getElementById("demoZ4").innerHTML = "Z4:";
document.getElementById("demoSZ4").innerHTML = getCookie("demoSZ4","--- ");
document.getElementById("demoR4").innerHTML = "R4:";
document.getElementById("demoSR4").innerHTML = getCookie("demoSR4","--- ");

document.getElementById("demoX").innerHTML = "X:";
document.getElementById("demoSX").innerHTML = getCookie("demoSX","--- ");
document.getElementById("demoY").innerHTML = "Y:";
document.getElementById("demoSY").innerHTML = getCookie("demoSY","--- ");
document.getElementById("demoZ").innerHTML = "Z:";
document.getElementById("demoSZ").innerHTML = getCookie("demoSZ","--- ");
document.getElementById("demoR").innerHTML = "R:";
document.getElementById("demoSR").innerHTML = getCookie("demoSR","--- ");

function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

function delCookie(cname) {
  document.cookie = cname + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
}

function getCookie(cname,defaultValue) {
```

```
  var name = cname + "=";
  var decodedCookie = decodeURIComponent(document.cookie);
  var ca = decodedCookie.split(';');
  for(var i = 0; i <ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return defaultValue;
}

function showEnviar()
{
  document.getElementById("demoX").innerHTML = "X:";
  var x = document.getElementById("myRange1").value;
  document.getElementById("demoSX").innerHTML = x;
  setCookie("demoSX", x, 1);

  document.getElementById("demoY").innerHTML = "Y:";
  var y = document.getElementById("myRange2").value;
  document.getElementById("demoSY").innerHTML = y;
  setCookie("demoSY", y, 1);

  document.getElementById("demoZ").innerHTML = "Z:";
  var z = document.getElementById("myRange3").value;
  document.getElementById("demoSZ").innerHTML = z;
  setCookie("demoSZ", z, 1);

  document.getElementById("demoR").innerHTML = "R:";
  var r = document.getElementById("myRange4").value;
  document.getElementById("demoSR").innerHTML = r;
  setCookie("demoSR", r, 1);
}
function updateP1()
{
  document.getElementById("demoX1").innerHTML = "P1  X1:";
  var x1 = document.getElementById("myRange1").value;
  document.getElementById("demoSX1").innerHTML = x1;
    setCookie("demoSX1",x1,1);

  document.getElementById("demoY1").innerHTML = "Y1:";
  var y1 = document.getElementById("myRange2").value;
  document.getElementById("demoSY1").innerHTML = y1;
    setCookie("demoSY1",y1,1);

  document.getElementById("demoZ1").innerHTML = "Z1:";
  var z1 = document.getElementById("myRange3").value;
  document.getElementById("demoSZ1").innerHTML = z1;
    setCookie("demoSZ1",z1,1);

  document.getElementById("demoR1").innerHTML = "R1:";
  var r1 = document.getElementById("myRange4").value;
  document.getElementById("demoSR1").innerHTML = r1;
    setCookie("demoSR1",r1,1);
}
function updateP2()
{
  document.getElementById("demoX2").innerHTML = "P2  X2:";
  var x2 = document.getElementById("myRange1").value;
  document.getElementById("demoSX2").innerHTML = x2;
    setCookie("demoSX2",x2,1);

  document.getElementById("demoY2").innerHTML = "Y2:";
  var y2 = document.getElementById("myRange2").value;
  document.getElementById("demoSY2").innerHTML = y2;
    setCookie("demoSY2",y2,1);

  document.getElementById("demoZ2").innerHTML = "Z2:";
  var z2 = document.getElementById("myRange3").value;
  document.getElementById("demoSZ2").innerHTML = z2;
    setCookie("demoSZ2",z2,1);
```

```
 document.getElementById("demoR2").innerHTML = "R2:";
 var r2 = document.getElementById("myRange4").value;
 document.getElementById("demoSR2").innerHTML = r2;
  setCookie("demoSR2",r2,1);
}
function updateP3()
{
 document.getElementById("demoX3").innerHTML = "P3  X3:";
 var x3 = document.getElementById("myRange1").value;
 document.getElementById("demoSX3").innerHTML = x3;
  setCookie("demoSX3",x3,1);

 document.getElementById("demoY3").innerHTML = "Y3:";
 var y3 = document.getElementById("myRange2").value;
 document.getElementById("demoSY3").innerHTML = y3;
  setCookie("demoSY3",y3,1);

 document.getElementById("demoZ3").innerHTML = "Z3:";
 var z3 = document.getElementById("myRange3").value;
 document.getElementById("demoSZ3").innerHTML = z3;
  setCookie("demoSZ3",z3,1);

 document.getElementById("demoR3").innerHTML = "R3:";
 var r3 = document.getElementById("myRange4").value;
 document.getElementById("demoSR3").innerHTML = r3;
  setCookie("demoSR3",r3,1);
}
function updateP4()
{
 document.getElementById("demoX4").innerHTML = "P4  X4:";
 var x4 = document.getElementById("myRange1").value;
 document.getElementById("demoSX4").innerHTML = x4;
  setCookie("demoSX4",x4,1);

 document.getElementById("demoY4").innerHTML = "Y4:";
 var y4 = document.getElementById("myRange2").value;
 document.getElementById("demoSY4").innerHTML = y4;
  setCookie("demoSY4",y4,1);

 document.getElementById("demoZ4").innerHTML = "Z4:";
 var z4 = document.getElementById("myRange3").value;
 document.getElementById("demoSZ4").innerHTML = z4;
  setCookie("demoSZ4",z4,1);

 document.getElementById("demoR4").innerHTML = "R4:";
 var r4 = document.getElementById("myRange4").value;
 document.getElementById("demoSR4").innerHTML = r4;
 setCookie("demoSR4",r4,1);
}

function Reset()
{
 document.getElementById("demoX1").innerHTML = "P1  X1:";
 var x1 = document.getElementById("myRange1").value;
 document.getElementById("demoSX1").innerHTML = "--- ";
  delCookie("demoSX1");

 document.getElementById("demoY1").innerHTML = "Y1:";
 var y1 = document.getElementById("myRange2").value;
 document.getElementById("demoSY1").innerHTML = "--- ";
  delCookie("demoSY1");

 document.getElementById("demoZ1").innerHTML = "Z1:";
 var z1 = document.getElementById("myRange3").value;
 document.getElementById("demoSZ1").innerHTML = "--- ";
  delCookie("demoSZ1");

 document.getElementById("demoR1").innerHTML = "R1:";
 var r1 = document.getElementById("myRange4").value;
 document.getElementById("demoSR1").innerHTML = "--- ";
  delCookie("demoSR1");

 document.getElementById("demoX2").innerHTML = "P2  X2:";
 var x2 = document.getElementById("myRange1").value;
 document.getElementById("demoSX2").innerHTML = "--- ";
  delCookie("demoSX2");
```

```
  document.getElementById("demoY2").innerHTML = "Y2:";
  var y2 = document.getElementById("myRange2").value;
  document.getElementById("demoSY2").innerHTML = "--- ";
    delCookie("demoSY2");

  document.getElementById("demoZ2").innerHTML = "Z2:";
  var z2 = document.getElementById("myRange3").value;
  document.getElementById("demoSZ2").innerHTML = "--- ";
    delCookie("demoSZ2");

  document.getElementById("demoR2").innerHTML = "R2:";
  var r2 = document.getElementById("myRange4").value;
  document.getElementById("demoSR2").innerHTML = "--- ";
    delCookie("demoSR2");

  document.getElementById("demoX3").innerHTML = "P3  X3:";
  var x3 = document.getElementById("myRange1").value;
  document.getElementById("demoSX3").innerHTML = "--- ";
    delCookie("demoSX3");

  document.getElementById("demoY3").innerHTML = "Y3:";
  var y3 = document.getElementById("myRange2").value;
  document.getElementById("demoSY3").innerHTML = "--- ";
    delCookie("demoSY3");

  document.getElementById("demoZ3").innerHTML = "Z3:";
  var z3 = document.getElementById("myRange3").value;
  document.getElementById("demoSZ3").innerHTML = "--- ";
    delCookie("demoSZ3");

  document.getElementById("demoR3").innerHTML = "R3:";
  var r3 = document.getElementById("myRange4").value;
  document.getElementById("demoSR3").innerHTML = "--- ";
    delCookie("demoSR3");

  document.getElementById("demoX4").innerHTML = "P4  X4:";
  var x4 = document.getElementById("myRange1").value;
  document.getElementById("demoSX4").innerHTML = "--- ";
  delCookie("demoSX4");

  document.getElementById("demoY4").innerHTML = "Y4:";
  var y4 = document.getElementById("myRange2").value;
  document.getElementById("demoSY4").innerHTML = "--- ";
      delCookie("demoSY4");

  document.getElementById("demoZ4").innerHTML = "Z4:";
  var z4 = document.getElementById("myRange3").value;
  document.getElementById("demoSZ4").innerHTML = "--- ";
  delCookie("demoSZ4");

  document.getElementById("demoR4").innerHTML = "R4:";
  var r4 = document.getElementById("myRange4").value;
  document.getElementById("demoSR4").innerHTML = "--- ";
  delCookie("demoSR4");
}

</script>

</body>
</html>
)=====";



//*************************************************************************************************************
*************************************************************************
const char P4D_InstrucoesD[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Dobot</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

```
<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}

div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/">Voltar</a>
</div>


<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

  <h2 style="text-align:center">Instruções  Dobot Magician</h2>
  <div class="a">
  <br>
  Deslize os sliders, correspondentes às coordenadas do Dobot, que pretender até aos valores desejados.
  Quando definir os valores, pressione o botão ENVIAR! para movimentar o Dobot Magician.</p>
  O botão, ENVIAR!, realiza uma Movimentação do Dobot Magician. Pode ver por baixo do botão as coordenadas enviadas</p>
  O botão, Posição Inicial, faz o Dobot retornar á sua posição de origem.</p>
  Pode utilizar os botões SAVE Posição 1, SAVE Posição 2, SAVE Posição 3 e SAVE Posição 4 para guardar até 4 poses do braço robótico. O botão RUN Posições, leva o Dobot a movimentar-se de acordo com as posições que gravou anteriormente.</p>
  O botão, RESET Posições, apaga as 4 posições gravadas.</p>
  <br><br><br>

  </div>
</div>

<div class="column side">
</div>
```

```
</div>

<div class="footer">
  <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin" target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)=====";



//*************************************************************************************************************
*****************************************************************************
const char P5D_SobreD[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="en">

<head>
<title>Dobot</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<style>
* {box-sizing: border-box;}
body {margin: 0;}

/* Style the header */
.header{background-color: none; padding: 20px; text-align: center;}

/* Style the top navigation bar */
.topnav {overflow: hidden; border: 1px solid #cccccc; background-color: #f2f2f2;}
/* Style the topnav links */
.topnav a {float: left; display: block; color: #000000; text-align: center; padding: 14px 16px; text-decoration: none;}
/* Change color on hover */
.topnav a:hover {background-color: #ddd; color: #FFFFFF;}

/* Create three unequal columns that floats next to each other */
.column {float: left; padding: 10px;}
/* Left and right column */
.column.side {width: 30%;}
/* Middle column */
.column.middle {width: 40%; background-color: none;}

/* Clear floats after the columns */
.row:after {content: ""; display: table; clear: both;}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {.column.side, .column.middle {width: 100%;}}

/* Style the footer */
.footer {background-color: #f2f2f2; border: 1px solid #cccccc; padding: 8px; text-align: center;}

div.a {text-indent: 50px; text-align: justify;text-justify: inter-word;}

/* redes sociais */
.fa {padding: 10px;width: 40px;text-align: center;text-decoration: none;margin: 0px 2px;border-radius: 50%;}
.fa:hover {opacity: 0.7;}
.fa-google {background: #dd4b39;color: white;}
.fa-linkedin {background: #007bb5;color: white;}
</style>
</head>


<body>

<div class="header">
  <h1>Interface Remota Gen&#233rica para Bra&#231os Rob&#243ticos</h1>
</div>

<div class="topnav">
  <a href="/">Voltar</a>
</div>
```

```
<div class="row">
<div class="column side">
</div>

<div class="column middle" style="text-align:left">

  <h2 style="text-align:center">Sobre o Dobot Magician</h2>
  <div class="a">
  <br>
  O braço robótico Dobot Magician, é constituído por 4 juntas, 4 elos de ligação e uma ventosa a vácuo, como acessório terminal.</p>
  Com a presente Interface, é possível controlar remotamente o Dobot Magician, com total capacidade de movimentos.</p>
  É possível guardar e/ou modificar até 4 posições distintas e efetuar uma sequência corrida dessas 4 poses.</p>
  <br><br><br><br><br>

  </div>
</div>

<div class="column side">
</div>
</div>


<div class="footer">
  <h3><a>&#169 2020 Andr&#233 Mira </a><a href="https://www.linkedin.com/in/andré-mira" class="fa fa-linkedin"
target="_blank"></a><a href="mailto:andremiracontact@gmail.com" class="fa fa-google"></a></h3>
</div>


</body>
</html>
)======";
```