

Departamento de Engenharia Eletrotécnica

Smart Lab Check: Sistema de rastreamento de equipamentos, materiais e componentes em laboratório

Relatório final da Unidade Curricular de Projeto em Engenharia Eletrotécnica de Computadores

Autores:

André Filipe Silva Ventura

Rafael José Fonseca Ferreira

Orientadores:

Professor Doutor Hugo Miguel Cravo Gomes

Professor Doutor Lino Miguel Moreira Ferreira

Professor Mestre Rui Miguel Baptista Peixoto

Leiria, julho de 2022

Agradecimentos

Queremos agradecer aos nossos professores orientadores, Doutores Hugo Miguel Cravo Gomes e Lino Miguel Moreira Ferreira e ao Mestre Rui Miguel Baptista Peixoto pela disponibilidade, flexibilidade, partilha de conhecimentos e soluções para alguns problemas. Agradecemos também ao Marco Santos e à Sofia Gualdino, responsável pelo centro de eletrónica, pela disponibilização de material para a produção do projeto e também pela ajuda na produção da placa do circuito impresso. Por último, queríamos agradecer aos nossos colegas que em algumas ocasiões partilharam conhecimentos connosco.

Resumo

O presente relatório pretende demonstrar todo o trabalho realizado no desenvolvimento do projeto *Smart Lab Check*, realizado no âmbito da unidade curricular de Projeto em Engenharia Eletrotécnica e de Computadores da Escola Superior de Tecnologias e Gestão do Instituto Politécnico de Leiria.

O projeto *Smart lab Check* consiste num sistema que permite a deteção e identificação de equipamentos, materiais e componentes presentes e ausentes de qualquer laboratório.

Para a realização deste sistema de rastreamento de equipamentos foram desenvolvidos dois sistemas com funcionalidades diferentes. Estes sistemas denominam-se como Check Box e Check IN/OUT. O Check Box realiza a verificação e gestão de materiais do laboratório, enquanto o sistema Check IN/OUT pretende detetar a saída ou entrada em laboratório de equipamentos eletrónicos de maior custo/importância (osciloscópio, fontes de alimentação ou sinal) e ainda os componentes e caixas.

O presente projeto centrou-se na realização de duas metodologias: uma verifica se as caixas do laboratório contêm as ferramentas, indica quais estão em falta. A segunda é a verificação e sinalização de equipamentos eletrónicos que saiam do laboratório. No fim pode ser tudo visto através da plataforma online Node-RED.

Abstract

The present report aims to demonstrate all the work carried out in the development of the Smart Lab Check project, carried out within the scope of the Project in Electrical and Computer Engineering curricular unit of the Escola Superior de Tecnologias e Gestão of the Instituto Politécnico de Leiria.

The Smart lab Check project consists of a system that allows the detection and identification of equipment, materials and components present and absent from any laboratory.

To implement this equipment tracking system, two systems with different functionalities were developed. These systems are called Check Box and Check IN/OUT. The Check Box performs the verification and management of laboratory materials, while the Check IN/OUT system intends to detect the exit or entry into the laboratory of electronic equipment of higher cost/importance (oscilloscope, power supplies or signal) and also components and boxes.

The present project focused on the realization of two methodologies: one checks if the laboratory boxes contain the tools, if not, it indicates which ones are missing. The second is the verification and signaling of electronic equipment leaving the laboratory. In the end, everything can be seen through the online platform Node-RED.

Índice

1.	Introdução	1
1.1.	Motivação e objetivos	1
1.2.	Cronograma Geral	2
1.3.	Estrutura do relatório.....	3
2.	Estado de arte	4
2.1.	Tecnologias utilizadas em deteção/identificação de objetos/comunicação.....	5
2.1.1.	RFID.....	5
2.1.2.	UWB	9
2.1.3.	GPS	10
2.1.4.	BLE	11
2.1.5.	WiFi.....	12
2.1.6.	UART	13
2.1.7.	I2C.....	15
2.1.8.	HTTP.....	16
3.	Desenvolvimento e implementação.....	18
3.1.	Requisitos do Sistema	18
3.2.	Estrutura Funcional do projeto.....	19
3.2.1.	Diagramas de Blocos.....	19
3.2.2.	Esquemas elétricos	20
3.3.	Componentes do Projeto	21
3.3.1.	Sistema Check Box	21
3.3.1.1.	Microcontrolador ESP32 WROOM 32	22

3.3.1.2.	Módulo de leitura RFID	23
3.3.1.3.	RTC DS3231	25
3.3.1.4.	Teclado Matrix	26
3.3.1.5.	Display OLED.....	27
3.3.1.6.	MicroSD adapter	29
3.3.1.7.	Raspberry Pi 3B	30
3.3.2.	Sistema Check IN/OUT	31
4.	Desenvolvimento da PCB	34
5.	Software	37
5.1.	Funcionalidade do Sistema Check Box.....	37
5.1.1.	Ficheiros cartao microSD.....	39
5.1.2.	Verificação da caixa	42
5.1.3.	Adicionar <i>tag</i> de caixa.....	43
5.1.4.	Adicionar <i>tag</i> componente	45
5.2.	Raspberry Pi 3B (WEBSERVER).....	46
5.2.1.	Processamento de dados.....	46
5.2.1.1.	Criação da base de dados phpMyAdmin	46
5.2.2.	Interface Node-RED.....	48
5.2.2.1.	Conexões entre nodes para caixa.....	52
5.2.2.2.	Conexões entre nodes para componentes	53
5.2.2.3.	Conexões entre nodes para o estado da caixa.....	55
5.2.2.4.	Conexões para adicionar caixas componentes na base de dados.....	55
5.2.2.5.	Conexões entre nodes dos botões	57
5.2.2.6.	Sistema Check IN/OUT Node-RED.....	58
6.	Testes de funcionamento	60
7.	Conclusão e trabalhos futuros	64
8.	Referências	65
9.	ANEXO 1	68

Chave de abreviaturas/siglas

AP- Access point

API - Application Programming Interfaces

BLE- Bluetooth Low Energy

ESTG - Escola Superior Tecnologia e Gestão

GPIO - General Ptheriperal Input Output

GPS - Global Positioning System

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

I/O - Input/Output

I2C - Inter-Integrated Circuit

ID – Identification

IDE - Integrated Development Envirnoment

IoT – Internet of Things

IP - Internet Protocol

IPLeiria - Instituto Politécnico de Leiria

JSON - JavaScript Object Notation

LAN - Local Area Network

LCD - Liquid Crystal Display

LED - Light-Emitting Diode

NC - Normally Connected

NO - Normally Open

OLED - Organic Light-Emitting Diodes

PCB - Printed Circuit Board

PHP - Hypertext Preprocessor

RFID - Radio-Frequency Identification

RTC - Real-Time clock

SDA - Serial Data

SPI - Serial Peripheral Interface

SyO - System on Chip

TCP - Transmission Control Protocol

UART - Universal Asynchronous Receiver/Transmitter

URL - Uniform Resource Locator

USB - Universal Serial Bus

UWB - Ultra Wide Bando

Lista figuras

Figura 1 - Diagrama de blocos do projeto.....	2
Figura 2 - Sistema RFID [4].....	6
Figura 3 - Tags de UHF RFID EPCglobal Gen2 and ISO/IEC 18000-6c [5].	7
Figura 4 - Tag ativa [6].	8
Figura 5 - Tag semi-passiva [7].	8
Figura 6 - Relação de âncoras de tags [10].	10
Figura 7 - Conceito de cálculo do GPS [12].	11
Figura 8 - Modulo Bluetooth [14].....	12
Figura 9 - Access Point [17].....	13
Figura 10 - Transmissão/Receção UART [18].....	14
Figura 11 - Paridade de bits [19].....	14
Figura 12 - Bits da comunicação I2C.....	15
Figura 13 - Comunicação I2C entre master e slave [20].....	16
Figura 14 - Diagrama geral do projeto.	19
Figura 15 - Hardware do sistema Check Box.	20
Figura 16 – Hardware do sistema Check IN/OUT.	21
Figura 17 – Modulo interno do microcontrolador [22].	23
Figura 18 - RFID JRD-4035 [23].....	24
Figura 19 - Esquema interno do leitor para comunicação com outro dispositivo [23].	24
Figura 20 – RTC DS3231.	25

Figura 21 - Esquemático interno da RTC [24].	26
Figura 22 . Esquema teclado matrix [25].	27
Figura 23 - Teclado matrix.	27
Figura 24 - Display OLED.	28
Figura 25 – Registo de modulo para comunicação I2C [26].	28
Figura 26 - MicroSD adapter.	29
Figura 27 - Raspberry Pi 3B [28].	30
Figura 28 - Esquemático completo da PCB.	34
Figura 29 - PCB no Proteus.	35
Figura 30 – PCB final com o hardware no interior da caixa.	36
Figura 31 - Caixa final Check Box	36
Figura 32 - Ficheiro excel das tags caixas.....	39
Figura 33 - Ficheiro excel das Tags caixas e Componentes.....	40
Figura 34 - Tag registada.	40
Figura 35 - Ficheiro das tags.....	41
Figura 36 - Conexão do componente ESP com o computador [30].	41
Figura 37 - Informação de uma verificação de caixa através da porta série.	42
Figura 38 - Diagrama de adicionar tag da caixa.....	44
Figura 39 - Diagrama de adicionar tag de componentes.	45
Figura 40 - Base de dados smartlab no phpmyadmin.	46
Figura 41 - Diagrama de blocos da Base de dados.	47
Figura 42 - Criação da tabela componentes.	48
Figura 43 - Criação da tabela leituras na base de dados.....	48
Figura 44 – Menu principal da Dashboard.....	49
Figura 45 – Aba dashboard Componentes em Falta.	50
Figura 46 – Aba dashboard Registo de leituras.....	50
Figura 47 – Aba dashboard Leituras.	50
	IX

Figura 48 – Aba dashboard adicionar.	51
Figura 49 – Aba dashboard Check IN/OUT.....	51
Figura 50 - Flow do nomecaixa e insert_leitura_caixa.	52
Figura 51 - Querie efetuada para obter o nome e laboratório da caixa.	52
Figura 52 - Querie para inserir na tabela leituras.	53
Figura 53 - Flow de como obter o nome e laboratório dos componentes.	53
Figura 54 - Modo de passar tags lidas entre funções.	54
Figura 55 - Nome e laboratório dos componentes.	54
Figura 56 - Flow para inserir componentes e verificar componentes em falta.	54
Figura 57 - Flow que indica o estado da caixa.	55
Figura 58 - Diagrama de adicionar caixa, componente e componente Mesa.	56
Figura 59 - Flow para adicionar caixa.....	56
Figura 60 – Aba da dashboard adicionar.....	57
Figura 61 - Inserir caixa na base de dados.	57
Figura 62 - Botões no Node-RED.....	58
Figura 63 - Flow Check IN/OUT.	58
Figura 64 - Inserir componente para tabela leituras2.	59
Figura 65 - Ficheiro Excel das tags dos componentes e das caixas	60
Figura 66 - Informação da porta Serial de uma leitura de testes.....	61
Figura 67 - Teste da verificação da caixa.....	62
Figura 68 - Teste de verificar caixa incompleta.....	63
Figura 69 - Informação do LCD referente à Leitura testada.....	63

Lista de Tabelas

Tabela 1 - Cronograma do sistema proposto.....	3
Tabela 2 - Exemplos de sugestões existentes [3].	4
Tabela 3 - Faixas de frequência [8].	9
Tabela 4 - Pinos de um cartão microSD [27].	29
Tabela 5 - Ligação do microSD ao ESP32.....	30
Tabela 6 - Lista de componentes do sistema Check Box.	35

1. Introdução

Atualmente, o controlo do material disponível nos laboratórios da Escola Superior de Tecnologia e Gestão de Leiria (ESTG) é efetuado por alunos, o que pode levar a erros de gestão de stock. Para automatizar este processo, tornando-o menos propício a erros, foi criado o projeto *Smartl Lab Check*. O projeto foi dividido em dois sistemas: Check Box e Check IN/OUT. A finalidade global dos sistemas é permitir a identificação e deteção perante a existência ou ausência de equipamentos ou materiais do laboratório. De modo a implementar os objetivos definidos foi necessário elaborar um estudo acerca das tecnologias a utilizar. Para a resolução do projeto foram escolhidas as tecnologias *Radio-Frequency Identification* (RFID), para detetar o material do laboratório e WiFi, para comunicar entre o utilizador e os dois sistemas.

O sistema Check Box faz a verificação do material existente em cada caixa pertencente aos postos de trabalho do laboratório. No final de cada aula, um aluno de cada posto, arrumará o material utilizado e passará a caixa junto do dispositivo para respetiva verificação. Esta verificação vai indicar se a caixa se encontra completa (contém o material todo) ou se tem algum material em falta (caixa incompleta). Caso a caixa tenha um material que não lhe pertence, é indicado que não pertence e a qual caixa pertence. Esta verificação é registada numa base de dados que pode ser acedida e visualizada através da interface criada com o serviço Node-RED.

O sistema Check IN/OUT verifica se algum material do laboratório sai ou entra no laboratório, e caso seja detetado algum movimento de entrada ou de saída de material é possível visualizar na plataforma do Node-RED o nome do material, laboratório a que pertence e a hora em que ocorreu o acontecimento. O registo é guardado na tabela da base de dados, sendo depois possível ver o registo no Node-RED.

1.1. Motivação e objetivos

Considerando este um projeto inovador, criado de raiz, que visa a ser utilizado no Instituto Politécnico de Leiria (IPLeiria)-ESTG para contribuir na gestão do material utilizado nos laboratórios do IPLeiria,

consideramos altamente motivador a possibilidade de contribuir para a melhoria dos laboratórios da instituição de ensino que frequentamos.

A motivação que se destaca para a implementação deste projeto foi a necessidade de constante observação e supervisionamento dos equipamentos e materiais no laboratório por parte dos docentes. Sendo a ESTG uma escola de tecnologia podem e devem ser criadas soluções para esta problemática. O objetivo consiste em desenvolver um dispositivo que promove a verificação e deteção autónoma de diversos materiais e equipamentos dentro do laboratório, sem necessidade de interação humana direta.

O diagrama de blocos relativo a este projeto está representado na Figura 1.

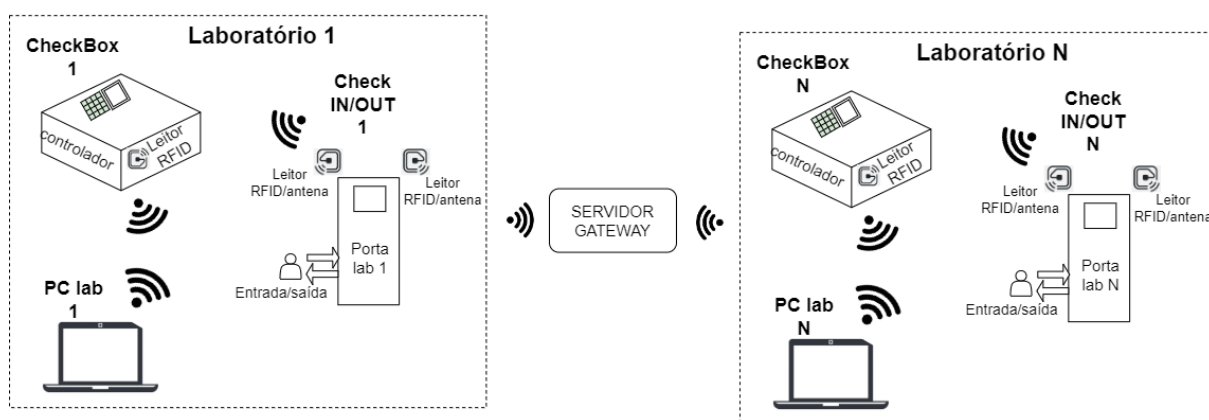


Figura 1 - Diagrama de blocos do projeto

1.2. Cronograma Geral

Para o desenvolvimento do presente projeto foi realizado um cronograma de tarefas a seguir, como se pode observar na Tabela 1.

Inicialmente foi feito um estudo das tecnologias a utilizar para uma boa adequação e realização do projeto. Após estas pesquisas foram considerados vários cenários possíveis, nomeadamente o poder de alcance do RFID, o tipo de tag, entre outros, de modo que o projeto fosse bem planeado inicialmente e com as melhores condições possíveis.

Tarefas	outubro	novembro	dezembro	janeiro	fevereiro	março	abril	maio	junho	Avaliação Final
1- Estudo e investigação das tecnologias e equipamento a utilizar.	x									
2- Estudo dos cenários de aplicação, equipamentos e componentes a ser rastreados, localização dos "módulos localizadores".	x	x								
3- Construção e testes de cenários de aplicação.		x	x	x						
4- Identificação de limitações dos cenários testados e processamento dos dados recolhidos.				x	x					
5- Implementação em hardware e firmware do módulo checkbox .						x	x			
6- Implementação do sistema de monitorização web.							x	x		
7- Implementação em hardware e firmware do módulo check-in e check-out .							x	x		
8- Desenvolvimento do sistema final.								x	x	
9- Conceção de testes e experimentação do sistema no "terreno".									x	
10- Desenvolvimento de documentação técnica.	x	x		x	x	x	x	x	x	

Tabela 1 - Cronograma do sistema proposto

De seguida deu-se a implementação do hardware, tendo sido utilizado algum equipamento experimental para averiguar possibilidades de implementação diferentes que era necessário serem estudadas e definidas. O passo seguinte foi a aquisição do hardware principal necessário para realizar o projeto, e implementação do software adequado. De seguida realizámos todos os testes de cenários possíveis com base potenciais cenários reais.

Com base nos testes realizados, e com hardware e software adquiridos prosseguiu-se para a definição e otimização global do sistema.

Por fim foi concluído o sistema com a funcionalidade desejada e procedeu-se à colocação de um dos sistemas em *Printed Circuit Board* (PCB).

1.3. Estrutura do relatório

Este relatório está dividido em sete capítulos onde no capítulo 1 é apresentado o problema e sumariada a solução proposta para o mesmo e o capítulo 2 consiste no estado de arte das tecnologias utilizadas na realização do projeto. No capítulo 3 é descrita a fase de desenvolvimento e implementação do projeto e no capítulo 4 consta o desenvolvimento da PCB. O capítulo 5 refere-se ao software utilizado na realização do projeto e os testes de funcionamento encontram-se descritos no capítulo 6. Por último, no capítulo 7 são efetuadas as conclusões finais e sugerido o trabalho futuro.

2. Estado de arte

Após diversos estudos de tecnologias que pudessem ser usadas e considerando os objetivos do projeto foram decididas quais as tecnologias a utilizar na realização deste projeto sendo que cada uma tem o seu propósito e funcionalidade no projeto.

Existem já alguns sistemas ou serviços implementados na atualidade, no qual têm um objetivo semelhante ao que se propõe neste projeto, no entanto nenhuma das implementações explicita diretamente o uso de RFID passivo para tal.

Deste modo os sistemas são explicados na generalidade devido à ocultação de informação de implementação por parte das entidades e empresas [1] [2].

Uns pequenos levantamentos acerca de sistemas em comum tenham funcionalidades ou tecnologias semelhantes com o que se propõe neste projeto e que servem de referência para o que se pretende, serão descritos na Tabela 2 .



Sistema	Tecnologias e constituição	Função	Vantagens
 Eliko UWM RTLS	Ultra Wide Band (UWB), Servidor Web	Sistema de localização em tempo real para ou rastreamento de objetos dentro de um espaço fechado com ligação e acesso da informação para a internet	Alta precisão, versátil, tempo real, suporte empresarial, monitorização através de objetos e obstáculos
 Sentrax	Sensores ambientais, Beacons, localizadores e pontos de acesso, Software próprio	Multi-sensorização e rastreamento em tempo real da posição, usado para diversas aplicações e setores como indústria, cidades, armazéns de manufatura e edifícios	Versátil, múltiplas aplicações, múltiplas tecnologias, suporte empresarial

Tabela 2 - Exemplos de sugestões existentes [3].

Uma vez que as tecnologias e sistemas apresentados faltam com alguma informação detalhada da sua implementação, e são implementados por empresas privadas ou entidades prestadoras de serviços, no qual têm total ou parcial controlo dos seus sistemas. Ao pensar nos recursos possíveis facultar para implementação do projeto, nas tecnologias e sistemas estudados, bem como nos objetivos deste projeto, decidiu-se a implementação dos Sistemas Check IN/OUT e Check Box, no qual é utilizado a tecnologia

de RFID e feito a conectividade dos dois sistemas através da implementação e uso de tecnologias e protocolos.

2.1. Tecnologias utilizadas em deteção/identificação de objetos/comunicação

De um modo inicial estudou-se as várias tecnologias tendo em conta os requisitos explícitos e funcionalidades pretendidas no projeto, refletiu-se sobre o conceito de (*Internet of Things*) IoT, na qual possibilita comunicação entre o utilizador e entre os dois sistemas futuramente designados Check IN/OUT Check Box.

Surgiram várias tecnologias que poderiam ir de enquadramento com o nosso trabalho como RFID, o UWB, o *Global Positioning System* (GPS) e *Bluetooth Low Energy* (BLE).

Após vários estudos optou-se pelo uso da tecnologia RFID pelas suas vantagens notórias em relação às outras tecnologias e que foi explicado anteriormente neste relatório.

A tecnologia UWB é extremamente precisa e é usada para distâncias curtas igualmente como o RFID, serve maioritariamente na parte dos casos para rastreamento de objetos em tempo real sendo que não se enquadra na totalidade no nosso sistema, também pelas razões de que é uma tecnologia mais cara que o RFID, e as *tags* que são usadas não se enquadram por serem do tipo ativas, deste modo precisariam de constante fornecimento de energia para alimentar as baterias.

2.1.1. RFID

O RFID é um protocolo de Identificação por Radiofrequência e surgiu como um sistema de rastreamento e controlo de acesso. Esta tecnologia permite a leitura e escrita de dados através de sinais de rádio. Um sistema RFID é geralmente composto pelo sistema de recolha/armazenamento de dados, pelo leitor e pelas *tags* RFID, como está apresentado na Figura 2.

Normalmente as *tags* estão agregadas fisicamente a objetos. Assim, o leitor faz a leitura do *Identification* (ID) presente nas *tags* através de um sinal de radiofrequência, sendo que o alcance do sistema varia com a frequência e a tipologia utilizada, ou seja, depende do tipo de *tag*. As *tags* podem ser ativas, semi-passivas ou passivas.

Esta tecnologia tem várias vantagens tais como, maior confiabilidade, pois são capazes de funcionar em maus ambientes e com clima extremos, alta velocidade, otimização de processos, facilidade de leitura,

rastreabilidade e capacidade de armazenamento e maior durabilidade como as *tags* passivas. Também tem as suas desvantagens como interferências por metais e custo elevado comparativamente a outras tecnologias, como por exemplo código de barras.

Este foi o tipo de tecnologia que decidimos utilizar na implementação do projeto, uma vez que esta tecnologia apresenta ter baixo custo, e atende às necessidades exigidas para completção do projeto, escolhendo-se para tal o uso de *tags* passivas que irá ser explicado adiante.

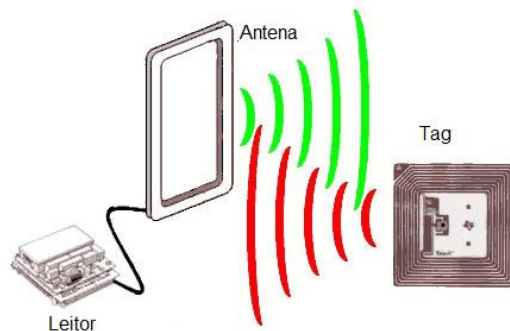


Figura 2 - Sistema RFID [4].

Um leitor RFID é responsável pela leitura das *tags* e pela gestão das colisões das leituras. Contém a interface para comunicar com o sistema de recolha de dados.

A *tag* é alimentada, envia dados de volta ao leitor, como acontece com a bobina do leitor, alternando o campo magnético, mas o campo magnético é tão pequeno que interfere com o campo magnético do leitor. Geralmente os leitores leem uma gama específica de frequências de funcionamento das *tags*, que está descrito na Tabela 3.

A *tag* é o dispositivo eletrónico que contém a informação de cada elemento a identificar. É constituído por uma antena e um microchip com o respetivo ID. Existem três tipos de *tags*, sendo elas as *tags* passivas, ativas e semi-passivas, que serão descritas mais abaixo. De seguida serão descritos os três tipos de *tags*:

Tags passivas: Este é o tipo de *tag* que não possui fonte de alimentação própria. Quando entra no raio de ação do leitor, é alimentada pela energia de ondas rádio emitidas pela antena do mesmo, deste jeito, a *tag* através da energia recebida do leitor gera um campo eletromagnético próprio e responde de volta para o leitor com uma onda rádio contendo a informação da *tag*. Este modo de funcionamento RFID é chamado de *backscatter*, para enviar uma resposta de volta.

É possível em certas circunstâncias que as *tags* RFID não sejam ativadas e não ocorram leituras, tal acontece caso não seja gerado um campo eletromagnético por parte da *tag* através do fenómeno *backscatter*, este contratempo pode acontecer por diversos motivos, sendo os mais comuns interferências de rádio frequência entre múltiplas *tags* sobrepostas, por serem lidas em simultâneo e gerando colisões na comunicação, uma vez que esta tecnologia utiliza uma única faixa de frequências com pouca largura de banda. Também pode ocorrer a reflexão ou refração em objetos físicos que anulam totalmente ou parcialmente as ondas geradas pela propagação de ondas rádio. Outra vantagem da utilização das *tags* deste tipo é o facto de possibilitar o uso de *tags* diversificadas e de tamanho reduzido, com baixo custo de aquisição, desde que sejam compatíveis com os protocolos de comunicação estabelecidos e use gamas de frequência comuns ao leitor RFID. A *tag* utilizada e que é compatível com o leitor RFID, no qual é usado no sistema Check Box está representada na Figura 3.

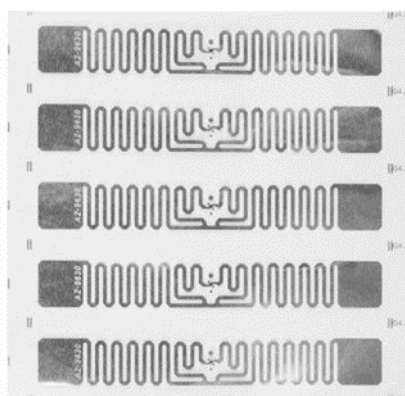


Figura 3 - *Tags* de UHF RFID EPCglobal Gen2 and ISO/IEC 18000-6c [5].

Vantagens: Tamanho pequeno, leve, baixo custo, longa vida até mais de dez anos, sem manutenção, formatos diversos, muitas aplicações disponíveis.

Desvantagens: Como não há fonte de alimentação interna, a distância de leitura é limitada. Geralmente, são necessários leitores RFID com grande potência, dependendo também da frequência de operação das *tags* conforme a gama espectral de frequências.

Esta tipologia de *tags* foi o que se decidiu utilizar no projeto, tendo em conta as vantagens relativas às outras tipologias de *tags* passivas RFID e a outras tecnologias principais pesquisadas neste capítulo, nomeadamente a distância de leitura entre *tags* e leitor é suficiente para o correto funcionamento dos sistemas, tem baixo custo monetário, e inexistência de necessidade de manutenção relativamente à alimentação das *tags*.

Tags ativas – As *tags* ativas, Figura 4, possuem fonte de alimentação integrada, normalmente pilhas, e funcionam de maneira ligeiramente diferente das *tags* passivas. As *tags* ativas, por não dependerem das ondas rádio do reader para se alimentar, podem ser programadas para comunicar periodicamente e enviar uma maior quantidade de informação, visto terem a possibilidade de possuir uma maior memória. O raio de alcance das *tags* ativas é superior ao das *tags* passivas.



Figura 4 - Tag ativa [6].

Vantagens: Longa distância de leitura, até dezenas de metros ou mesmo centenas de metros.

Desvantagens: Grande volume, alto custo, o tempo de serviço é limitado pela vida útil da bateria.

Tags semi-passivas – Este tipo de *tag* é um misto entre *tag* passiva e ativa visto que funciona do mesmo modo da *tag* passiva, ou seja, apenas responde mediante solicitação do leitor, no entanto também tem fonte de alimentação própria que permite funcionar caso a distância de operação exija maiores distâncias. Pode observar-se um exemplo de *tag* semi-passiva na Figura 5.



Figura 5 - Tag semi-passiva [7].

Vantagens: Estes tipos têm velocidade de resposta mais rápida e melhor eficiência, consomem menos energia.

Desvantagens: Tamanho grande e alto custo

A *tag* recebe a energia por meio de ondas eletromagnéticas provenientes da antena do leitor. Uma vez que as ondas alcançam a *tag*, a energia viaja através da antena interna da *tag* e ativa o *chip*, ou circuito integrado. A energia restante é modulada com os dados do chip e flui de volta através da antena da *tag* para a antena do leitor na forma de ondas eletromagnéticas.

Faixa de frequência	Banda	Alcance entre o leitor e a <i>tag</i>	Vantagens	Desvantagens	Aplicação
LF	125 KHz – 134 KHz	Menos de 0.5 metro	Boa operação próximo a metais e água	Curto alcance de leitura	Rastreamento de animais, controle de acesso, imobilização de veículos, autenticação de produtos, identificação de itens, bagagens em linhas aéreas, <i>smart cards</i> e bibliotecas
HF	13.56 MHz	Menos de 1 metro	Baixo custo das <i>tags</i> , boa interação e boa qualidade de transmissão	Necessita de potência elevada nos leitores	Identificação de itens, bagagens em linhas aéreas, <i>smart cards</i> e bibliotecas
UHF	860 MHz – 960 MHz	Até 9 metros	Baixo custo, <i>tags</i> com tamanho reduzido	Não opera bem próximo a metais e líquidos	Controlo de fornecimento logístico
Microondas	2.45 GHz – 5.8 GHz	Acima de 10 metros	Velocidade de transmissão de dados	Não opera bem próximo a metais e líquidos, maior custo	Controlo de fornecimento logístico

Tabela 3 - Faixas de frequência [8].

2.1.2. UWB

UWB é uma tecnologia de pequeno alcance, com propagação através de ondas rádio, semelhante ao RFID. Pode ser usada para monitorização de posições em ambientes fechados e o seu maior benefício em relação a outras tecnologias é a sua precisão na leitura de objetos denominadas etiquetas UWB, assim como um desempenho acrescido no aspeto de propagação Multipercurso sem interferências, comparativamente ao WiFi ou BLE.

Contudo têm uma bateria própria pouco duradoura que é necessário ser recarregada de tempo a tempo e apenas permitem ser acoplados a objetos de grande dimensão.

São usados por norma três recetores denominados âncoras que permitem a identificação das posições das etiquetas através do método de diferenciação de tempo referente a cada âncora, Figura 6.

Esta tecnologia torna-se dispendiosa e complexa a sua implementação para um laboratório, pelo que não existe realmente a necessidade de precisão que esta tecnologia oferece [9].

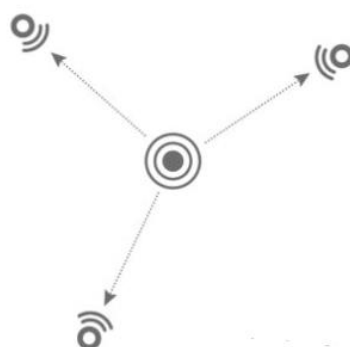


Figura 6 - Relação de âncoras de tags [10].

2.1.3. GPS

GPS é um sistema de navegação baseada no uso de satélites, que na atualidade tem bastante uso no mercado e diversas funcionalidades úteis para fins que exijam determinação de posições, uso de navegação mapeamento de mapas.

O seu funcionamento reside na identificação da posição de um objeto recetor através dos satélites e de uma técnica referida como trilateração, Figura 7.

A trilateração utiliza operações matemáticas e satélites para determinação da velocidade, posição e elevação do objeto recetor rastreado.

Esta técnica incide na receção de sinais rádio provenientes do objeto recetor para cada satélite, deste modo aplica vários algoritmos matemáticos para cálculo dos parâmetros de posição, a informação proveniente de cada satélite contribui para a precisão no rastreamento de posição [11].

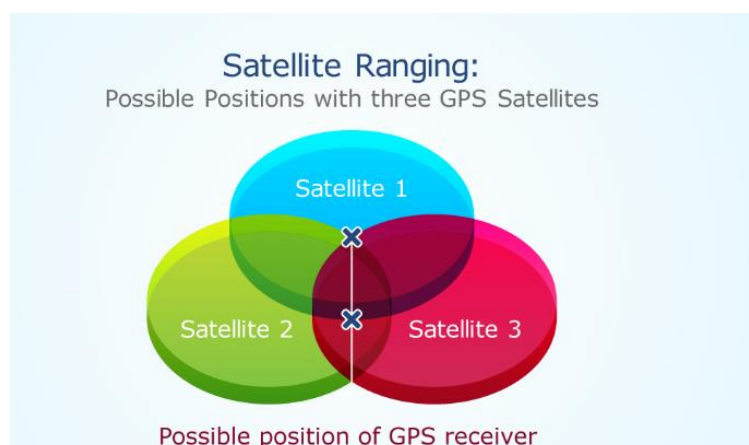


Figura 7 - Conceito de cálculo do GPS [12].

Apesar da tecnologia GPS ser de altamente versátil e ter uma grande utilidade para o quotidiano e para inumeráveis setores industriais, não vai ao encontro de ser uma ferramenta útil para o desenvolvimento deste projeto.

Para a área que é necessário rastrear, neste caso o laboratório e pretendido obter a localização, presença ou ausência de material, o GPS não é viável. Uma vez que o GPS tem uma precisão pequena para identificação da posição e rastreamento em situações de pequenas áreas. Sendo que acoplamento dos recetores de GPS em material do laboratório seria bastante difícil, uma vez que o material do laboratório pode ter pequenas ou médias dimensões, para além disso teria de ser fornecido constante manutenção aos recetores de GPS devido a não terem uma fonte de energia própria.

2.1.4. BLE

BLE é uma tecnologia de baixo consumo que provém da tecnologia Bluetooth convencional consistindo na propagação de ondas rádio na banda de 2.4 GHz de frequências, permitindo a ligação ponto a ponto entre dois dispositivos que incorporem esta tecnologia.

Cada conexão de dispositivos Bluetooth ocupa um canal na banda de radiofrequência específica para estabelecer uma comunicação e assim transmitir e receber dados de um dispositivo para outro.

No entanto o BLE utiliza um sistema de modulação para transmissão e receção menos complexo do que o Bluetooth, deste modo gasta menos energia de consumo para os dispositivos. Este método apenas permite iniciar uma conexão entre os dois dispositivos ponto a ponto caso seja necessária a transmissão ou receção de dados, caso contrário fica num estado de poupança de energia.

O BLE não permite a transferência de tanta informação e a ligação entre dispositivos não está constantemente ativa como no Bluetooth.

Esta tecnologia em relação ao tradicional Bluetooth e GPS que tem maior gasto de energia, os módulos recetores BLE podem conter pequenas dimensões, e a sua distância de comunicação para um forte sinal de receção pode ir até vinte e cinco metros, assim enquadrava-se na perfeição para o cenário proposto, no entanto o seu problema principal é o custo e quantidade de dispositivos Beacon de receção BLE necessários para acoplar em cada material do laboratório, para além disso a sua precisão de localização não é a melhor das outras tecnologias referidas, pelo que pode dar uma localização até cinco metros diferente da realidade [13]. Um exemplo de módulo de Bluetooth está representado na Figura 8.



Figura 8 - Modulo Bluetooth [14].

2.1.5. WiFi

A tecnologia WiFi é uma das tecnologias mais usadas e permite que se criem ligações de dados fechados entre dispositivos ou em rede, ou seja, possibilita o envio de dados de forma segura entre os dispositivos.

Para estender a comunicação o WiFi utiliza *Access point* (AP), pega na largura de banda proveniente de um router e estende para que outros dispositivos possam entrar na rede.

Um ponto de acesso fornece dados sobre os dispositivos na rede, segurança entre outros. Os AP (Figura 9) são elementos responsáveis pela ligação sem fios dos vários dispositivos terminais entre eles e à rede *Local Area Network* (LAN). São também responsáveis pelo aumento de alcance da rede, ao mesmo tempo que podem ser interligados com outras redes. Algumas das grandes vantagens deste tipo de tecnologia são a facilidade de instalação e a economia de meios de transmissão. Deste modo é fácil e rápido criar uma ligação sem fios, segura e com largura de banda considerável, entre dispositivos [15].

O padrão IEEE 802.11 define os protocolos que permitem comunicações com dispositivos sem fio habilitados para Wi-Fi atuais, incluindo roteiros sem fio e pontos de acesso sem fio.

IEEE 802 é uma seleção de padrões de rede que cobrem as especificações da camada física e de união de dados para tecnologias como Ethernet e wireless. As redes IEEE 802.11b e IEEE 802.11g usam a frequência 2.4 GHz para seus canais, que são espaçados a cada 5 MHz. As redes IEEE 802.11n usam frequências de 2.4 ou 5 GHz. O padrão mais utilizado usa 5GHz, IEEE 802.11ax [16].

As gamas de frequências atribuídas ao Wi-Fi são de 2.400-2.4835GHz e 5.725-5.850GHz.

Deste modo esta tecnologia apenas serve como meio de comunicação entre o microcontrolador ESP32 e o Raspberry Pi, sem quaisquer necessidades de uso de cabos, sendo apenas necessário os dispositivos terem cobertura de uma rede WiFi, que neste caso para o projeto a rede WiFi é a rede privada dos labs da ESTG.

Esta tecnologia não é indicada para o rastreamento do material em laboratório, uma vez que seria bastante difícil a sua implementação devido ao elevado consumo de energia para sua alimentação, bem como a dificuldade de agregação destes módulos nos para pequenos e médios tamanhos de material de laboratório.

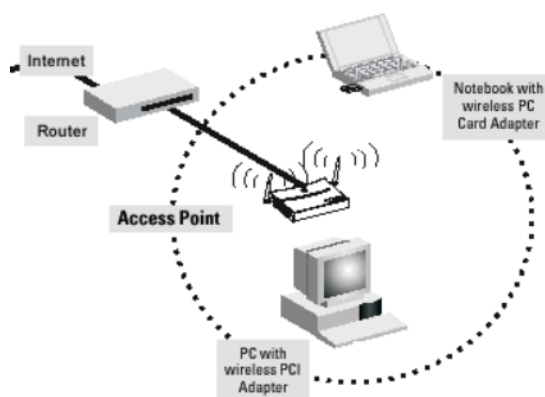


Figura 9 - Access Point [17].

2.1.6. UART

Universal Asynchronous Receiver/Transmitter (UART) não é um protocolo de comunicação série tal como o *Serial Peripheral Interface* (SPI) e *Inter-Integrated Circuit* (I2C), sendo este um circuito à parte

embutido dentro outro hardware que serve para transmitir dados com outro dispositivo que também tenha este circuito embutido.

Este método de comunicação é assíncrono na qual a transmissão e receção de um dispositivo para outro não depende de um relógio de sincronização entre o dispositivo emissor e o recetor, porém numa transmissão de um dispositivo para outro existem dois portos para cada um, cada um com funções de transmissão e receção, como mostra na Figura 10.

Numa transmissão a informação é transferida do porto transmissor para o porto recetor no circuito UART de cada dispositivo, como demonstrado abaixo, no qual os bits de informação para transmitir são internamente transferidos do processador do dispositivo em modo paralelo, ou seja, todos em simultâneo, para o circuito interno UART, sendo estes são agrupados em pacotes de dados e são adicionados bits extra com bit de inicialização, paridade e de paragem.

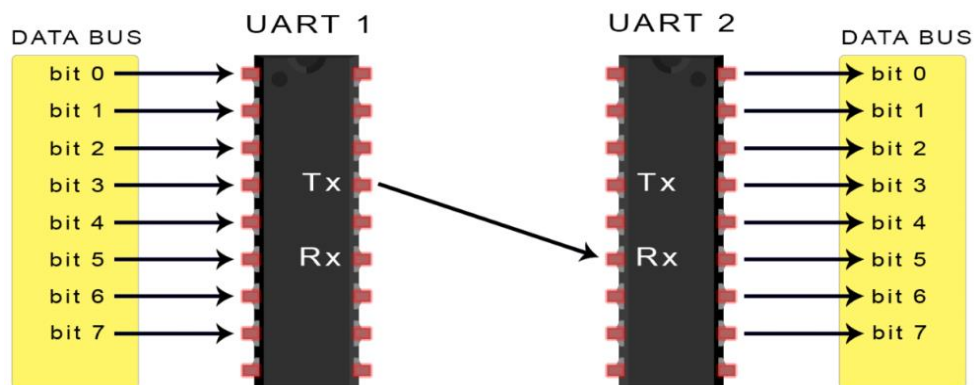


Figura 10 - Transmissão/Receção UART [18].

O circuito recetor reconhece que a informação é transmitida através da leitura do estado da linha de transmissão do outro dispositivo, na qual simboliza o bit de inicialização. Os bits de informação são seguidos com bits de paridade (Figura 11) e de paragem, nos quais servem para sinalizar bits de informação alterados durante a transmissão, por fim os bits de paragem estabelecem o fim de uma transmissão.

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)
------------------------	------------------------------------	-------------------------------	------------------------------

Figura 11 - Paridade de bits [19].

2.1.7. I2C

O protocolo I2C transfere informação bit a bit através de uma única linha, sendo que transmite e recebe dados nesta mesma, mas sem as comunicações se sobreporem uma em relação à outra (Figura 12).

O dispositivo de controlo que inicializa a comunicação é denominado *Master* e o recetor da informação é denominado *Slave*. Neste caso o *Master* é sempre a unidade de processamento e o *Slave* um sensor que recebe ordens de operação provenientes de um microcontrolador ou outra entidade de processamento.

Para este protocolo são apenas necessárias duas linhas de transmissão de dados, uma linha como referida para passagem de dados e outra para sincronização entre o *Slave* e o *Master* de modo a garantir que não haja falhas na passagem de dados nem sobreposição.

É ainda possível vários dispositivos *Slave* estarem conectados ao mesmo para apenas um dispositivo *Master*, tal acontece devido ao facto de todos os dispositivos *Slave* usarem um endereço único diferente, no qual o *Master* começa por enviar informação de inicialização para indicar que pretende estabelecer uma comunicação.

Todos os endereços *Slave* disponíveis e conectados ao *Master* recebem esse sinal, no entanto apenas os dispositivos *Slave* cujo endereço seja igual ao endereço frame enviado pelo dispositivo *Master* enviam um bit de confirmação de volta para sinalizar que este é válido para comunicação, em caso contrário a linha *Serial Data* (SDA) do *Slave* é colocada a nível alto e assim não estabelece comunicação com o dispositivo *Master*.

A informação é transmitida ou recebida entre *Master* e *Slave* em conformidade com um bit em cada frame de informação enviado pelo *Master*. Caso a informação chegue corretamente ao dispositivo *Slave* é enviado de volta para o *Master* um bit ACK de confirmação que os bits de informação foram recebidos corretamente.

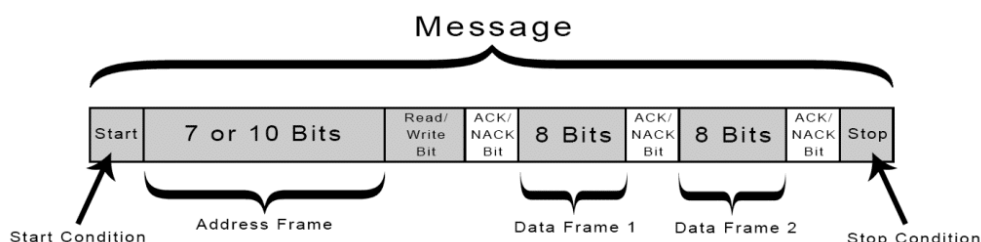


Figura 12 - Bits da comunicação I2C.

SPI utiliza ligações físicas para transmitir informação entre dispositivos em que é utilizado uma ligação série entre eles, ou seja a informação é passada ao longo do tempo e não instantaneamente sobre a forma de impulsos, no qual originam bits de informação ao longo do tempo, este tempo é delimitado pelo relógio que é inicializado pela entidade que transmite os dados, o relógio sincroniza a informação passada entre o emissor denominado *Master* o recetor denominado *Slave*, deste modo a informação é transmitida e recebida simultaneamente sem interrupções no tempo, Figura 13.

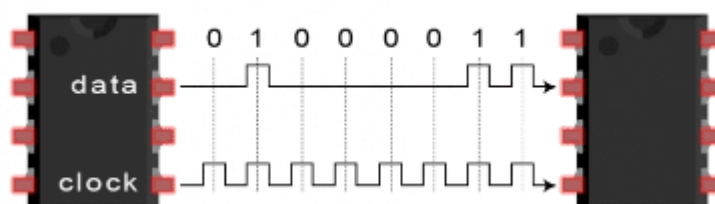


Figura 13 - Comunicação I2C entre *master* e *slave* [20].

2.1.8. HTTP

HyperText Transfer Protocol (HTTP) é um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores, principalmente na *World Wide Web* (Internet). No nosso projeto decidimos utilizar este tipo de comunicação devido ao fato de nunca termos trabalhado com ele, de modo a trabalhar com algo novo.

Este protocolo é utilizado para transferência de páginas *HyperText Markup Language* (HTML) do computador para a Internet, por este motivo, os endereços dos websites *Uniform Resource Locator* (URL), ou seja, é o endereço de qualquer site na internet) utilizam no início a expressão "http://". Esta informação é necessária para estabelecer a comunicação entre a URL e o *webserver* que armazena os dados. Para realizar a transferência de dados o protocolo HTTP, necessita de estar agregado a outros dois protocolos de rede: *Transmission Control Protocol* (TCP) e *Internet Protocol* (IP). Estes dois protocolos fazem o modelo TCP/IP, um conjunto de protocolos de comunicação entre computadores em rede [21]. O HTTP usa métodos de requisição (*http request*) e resposta (*http response*). *Http request* serve para indicar que ação é executada e o *http response* espera por essa ação ser executada e informa o que aconteceu, se foi enviada com sucesso ou não. O *http request* usa dois métodos, sendo eles a função *get* que é utilizada para o utilizador obter recursos e função *post* que é realizada quando o utilizador deseja enviar dados.

No projeto esta comunicação é feita entre o microcontrolador ESP32 e o Node-RED, que está no *webserver*, Raspberry Pi 3B, através das funções get e post. O ESP32 envia os dados para o Node-RED, onde este faz a função post dos dados enviados, neste caso as *tags*. Caso seja necessário o Node-RED fazer um pedido de dados ao ESP32, é utilizada a função get, neste caso para a realização dos botões.

3. Desenvolvimento e implementação

Este capítulo apresenta todo o desenvolvimento e implementações realizadas no projeto.

3.1. Requisitos do Sistema

Numa fase inicial, uma vez que o projeto poderia tomar vários rumos, definiu-se inicialmente um objetivo concreto para o sistema, uma vez que o projeto tem por base o rastreamento e deteção de *tags* dentro de laboratório, o que o abre inúmeras possibilidades de funcionalidades e de propósitos. Concluímos, após considerar e refletir, de que não seria necessário rastrear o equipamento dentro do espaço fechado, apenas garantir a presença dos materiais e componentes dentro da sala. Pensámos em duas funcionalidades distintas, a verificação de componentes presentes nas caixas de material e monitorização quanto à saída do material da sala.

Relativamente à fase inicial, os objetivos e finalidade do projeto até esta fase mantiveram-se inalterados, sendo que mantivemos o mesmo objetivo de desenvolver dois sistemas distintos, mas com comunicação e interligação entre eles.

Apenas foram mudados alguns detalhes durante o seu desenvolvimento, foram usadas novas metodologias como o uso de hardware, Software e Firmware nos, quais se destacam: periféricos conectados ao microcontrolador ESP32, computador Raspberry Pi, ferramenta de programação Web como o Node-RED e também uso de base de dados.

Mais concretamente houve grandes progressos relativamente ao sistema inicial Check Box, no qual numa primeira fase só constava a funcionalidade de verificação de caixas de material, não estando otimizada da melhor maneira em termos de código para um bom interface para com o utilizador.

Deste modo o sistema foi melhorado em relação ao antigo, e adicionadas mais funcionalidades como a inserção de informação e visualização por parte do utilizador através da interface deste com o microcontrolador ESP32.

Também foram feitas várias otimizações de código e manipulação da informação, como também se procedeu à integração entre a ferramenta de software. Node-RED. Igualmente como o microcontrolador ESP32, este é feito através de conexão WIFI e uso de protocolos de comunicação como é o caso do HTTP, nos quais serão explicados mais adiante.

3.2. Estrutura Funcional do projeto

Neste capítulo serão apresentadas as estruturas do Check Box e Check IN/OUT com diagramas.

3.2.1. Diagramas de Blocos

O esquema geral do funcionamento do projeto encontra-se na Figura 14, engloba os dois sistemas locais tanto o designado Check Box como o Check IN/OUT, juntamente com a sua constituição. A informação adquirida pelos sistemas tem como principal função a aquisição de *tags* referentes ao material de laboratório. Posteriormente é enviada a informação para o servidor Raspberry Pi que contém um servidor Node-RED que possibilita interligação da interface na Web e apresentação de informação e interação para com os utilizadores, para além é o servidor da base de dados.

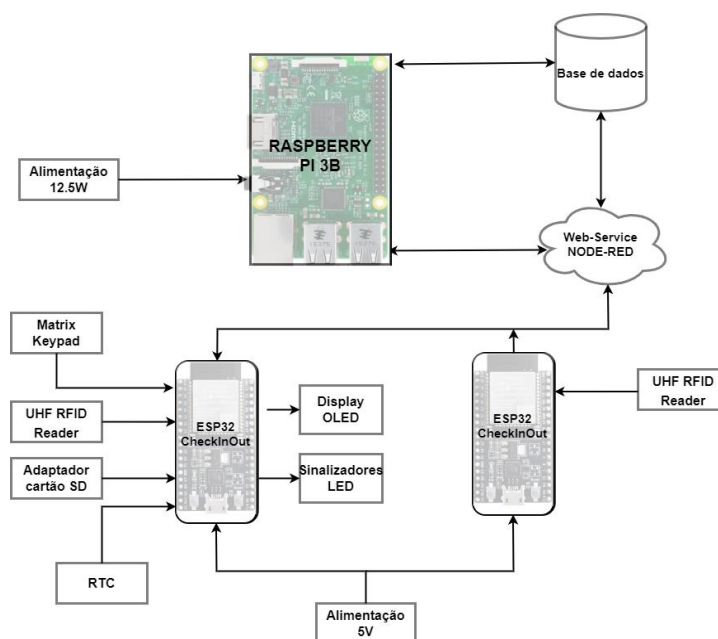


Figura 14 - Diagrama geral do projeto.

3.2.2. Esquemas elétricos

Check Box

O Sistema Check Box tem obrigatoriamente de ser constituído por um conjunto de periféricos que permitam na execução as tarefas pretendidas, deste modo utilizou-se uma unidade de processamento denominado microcontrolador, que permite estabelecer ligação e controlar os dispositivos, deste modo, é o elemento do projeto onde se desenvolve funcionalidades interligando e programando o hardware implementando assim o sistema pretendido.

mostra as ligações do sistema Check Box feitas, juntamente com os diversos protocolos de comunicação usados, componentes e ligações entre dispositivos.

Na Figura 15 mostra as ligações do sistema Check Box feitas, juntamente com os diversos protocolos de comunicação usados, hardware e ligações entre dispositivos.

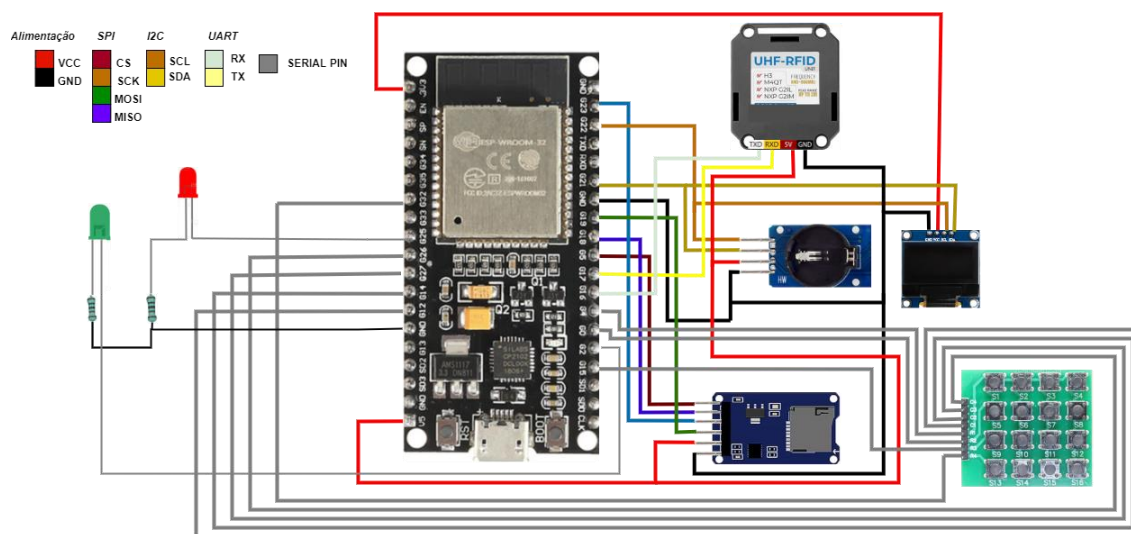


Figura 15 - Hardware do sistema Check Box.

Check IN/OUT

O esquema elétrico do sistema Check IN/OUT foi alterado na implementação relativamente ao ilustrado, a Figura 16, esquematiza o que seria suposto implementar, mais abaixo este será explicado juntamente com a razão de não ser implementado e quais problemas obtidos na tentativa da sua execução.

Sendo o sistema final Check IN/OUT apenas é constituído por um leitor RFID UHF JRD 4035, posteriormente é feita uma comunicação WiFi para o uso do HTTP e assim enviar remotamente a informação das *tags* de materiais ou caixas de laboratório.

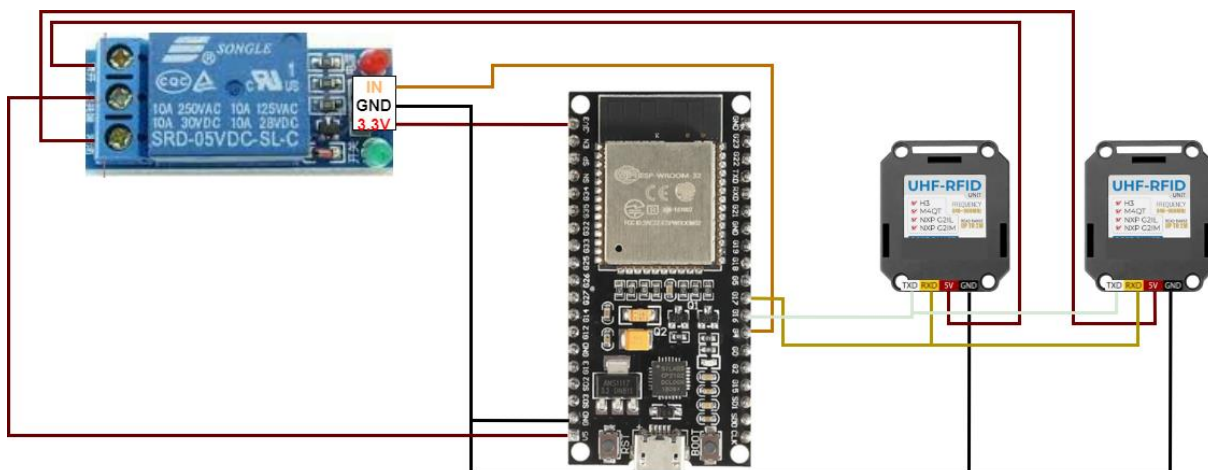


Figura 16 – Hardware do sistema Check IN/OUT.

3.3. Componentes do Projeto

Neste capítulo serão descritas as principais características de cada componente a escolha criteriosa dos mesmos.

3.3.1. Sistema Check Box

O Sistema Check Box foi testado no ESP32 e por ser de mais fácil implementação e familiaridade com *Integrated Development Environment* (IDE) Arduino logo é usado para operar o leitor de RFID neste sistema.

O sistema Check Box, baseia o seu funcionamento na utilização do microcontrolador ESP32, ao qual estão interligados os outros módulos de hardware.

Os componentes auxiliares são um teclado numérico matricial, um adaptador de cartões microSD, *Liquid Crystal Display* (LCD) *Organic Light-Emitting Diodes* (OLED), módulo de relógio em tempo real, leitor de *tags*, RFID e *Light-Emitting Diodes* (LEDs) de sinalização.

O código em ambiente Arduíno é baseado em tipo de linguagem C e C++, pelo que é de mais baixo nível comparativamente com a linguagem python3 usada no Raspberry Pi, deste modo, é mais contextualizada para executar instruções de mais baixo nível, facilitando a manipulação de instruções para o hardware, assim o código pode ser compilado e executado mais rapidamente, facilitando no encontro e resolução de problemas ao nível de programação para o hardware. Tornando-se assim a linguagem de programação mais usada para o desenvolvimento de aplicações em Hardware.

3.3.1.1. Microcontrolador ESP32 WROOM 32

O microcontrolador ESP32 é um *System on Chip* (SoC) no qual possui uma unidade de processamento acoplado a uma placa composto com múltiplo hardware embutido auxiliar que em conjunto forma em um sistema complexo em que executa diversas tarefas a nível de sistemas embebidos, programação e processamento. Tem a função de interagir e interligar diferentes dispositivos num único sistema central possibilitando a conectividade e operação de instrução com vários subsistemas que conectados ou embutidos na placa. Como referido anteriormente é o componente do sistema responsável pelas ligações entre dispositivos, programação e execução de instruções.

Tendo inúmeras características e que levaram à sua escolha em relação a outros SoC disponíveis no mercado. Nomeadamente os principais razões da sua aquisição são o seu custo baixo, a sua constituição interna, por fim a facilidade e apoio em software disponível já existente.

Em termos de características de hardware este é constituído por dois processadores que têm uma velocidade consideravelmente rápida, podendo alcançar até 600 instruções por segundo o que o torna bastante eficiente e rápido para a sua função, também possui uma memória volátil (*Sram*) e de armazenamento (*Flash*) suficiente. Também tem módulos de comunicação embutidos internamente imprescindíveis para auxílio e implementação neste sistema, nomeadamente I2C, SPI, WiFi, UART, entre outros, que o tornam bastante completo. Adicionalmente tem pinos configuráveis denominados *GPIO* (*General Ptheriperal Input Output*), nos quais podem ser programados de diferentes maneiras para serem ligados a outros componentes ou periféricos.

O principal hardware e eletrónica que constitui um SoC pode ser ilustrado na Figura 17, no qual em particular o ESP32 WROOM também integra estes na sua estrutura e ainda alguns outros como o caso de um módulo de comunicação WiFi.

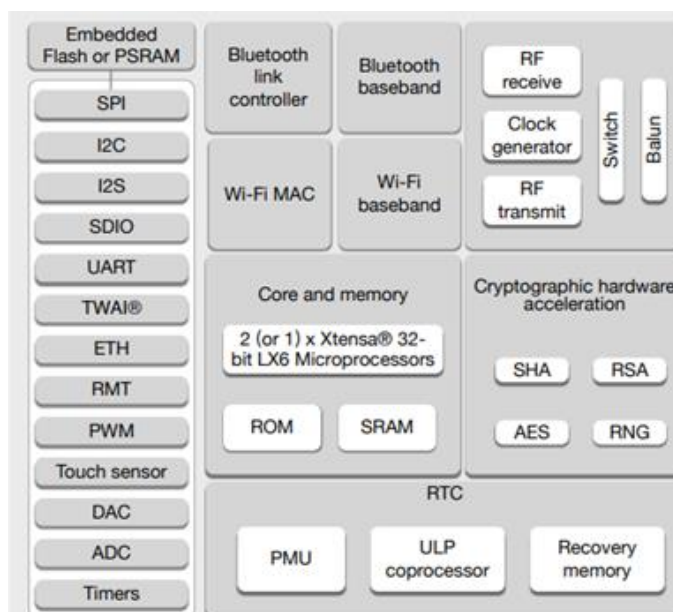


Figura 17 – Módulo interno do microcontrolador [22].

3.3.1.2. Módulo de leitura RFID

Esta é uma parte fundamental neste projeto, tal como já referenciado, consiste na deteção /identificação de materiais através de *tags* RFID. Sendo assim, a escolha do leitor é de extrema importância, já que deveremos ter em conta as frequências utilizadas, a distância de deteção, o número de *tags* lidas em simultâneo, entre outras.

A Figura 18 apresenta o módulo RFID JRD-4035 tem a funcionalidade de aquisição de dados no sistema Check Box, no qual é um Leitor de *tags* RFID que opera na banda de frequências UHF na gama espectral de frequências de 840 MHz a 960 MHz, tendo como características técnicas teóricas a possibilidade de leitura até 200 *tags* em simultâneo utilizando o protocolo de interface aéreo ISO 18000-6C E EPCglobal UHF class 1 Gen 2 que especificam um conjunto de medidas de operação e espalhamento de ondas Radio-freqüência na de faixas de frequências [23].



Figura 18 - RFID JRD-4035 [23].

Este leitor é conectado ao ESP32 fisicamente através do protocolo UART de comunicação, como se pode observar na Figura 19.

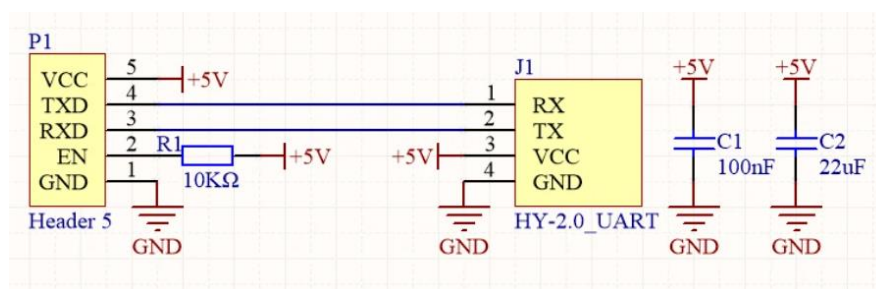


Figura 19 - Esquema interno do leitor para comunicação com outro dispositivo [23].

Devido ao facto de o leitor RFID não ter software próprio para auxiliar a sua utilização para leituras através da porta série do computador ou outros dispositivos diretamente. Foi necessário pesquisar fontes de código e referências do leitor, foi encontrado o repositório oficial disponibilizado no GitHub pelo seu fabricante, que fornece algum apoio para operar com este, no entanto não é apropriado para o Hardware que usamos no projeto nomeadamente o ESP32.

Deste modo foi necessário adaptar o código fonte encontrado, para ser executado no microcontrolador ESP32 o leitor. A própria marca apenas disponibilizava o código fonte para interagir com o hardware próprio “M5STACK core”.

Para este efeito e operar corretamente com o leitor através do ESP32, foi necessário retirar várias bibliotecas e partes de código que não seriam necessárias, e que não permitiam a execução de compilação para o ESP32 diretamente.

3.3.1.3. RTC DS3231

Na *Real-Time Clock* (RTC) (Figura 20) é usado um relógio digital de tempo real de alta precisão e de baixo consumo de potência, tem incorporado na sua constituição um sensor de temperatura e um oscilador de modo a melhorar a sua precisão de informação.

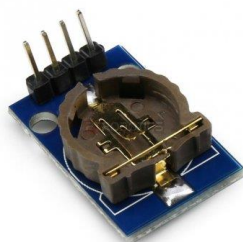


Figura 20 – RTC DS3231.

A Figura 21 mostra o esquemático interno da RTC e as ligações de funcionamento do Chip DS3231 com o exterior.

Tal como referido, este módulo tem um único objetivo neste sistema de recolher informação de hora e data atuais no momento de uma verificação de *tag* de caixa. É necessária uma configuração inicial através de software para configurar a informação atual de data e hora.

Caso o microcontrolador ESP32 do sistema Check Box se desligue da corrente, as informações internas no Chip da RTC no dispositivo não são perdidas, pois tem uma pilha que serve de bateria na ausência de alimentação interna por parte dos pinos de alimentação.

A leitura e escrita da informação referente ao calendário e relógio é obtida através da leitura de bytes concretos do registo, no qual estes inicialmente são escritos e sob a forma de binário decimal.

Apercebeu-se a importância de registo de data e hora em que é realizado o rastreio de material das caixas através do sistema Check Box, permitindo, não só, perceber em que momentos os materiais são utilizados, mas também, quando é detetada a falta de algum desses materiais.

Após análise do diagrama de blocos do ESP32, verificamos a existência de um bloco de relógio a tempo de real (RTC), no entanto, segundo informação recolhida, este é pouco preciso, o que seria prejudicial para a fiabilidade da informação recolhida. Desta forma, tornou-se necessário adicionar um módulo RTC externo.

No projeto definiu-se para cada leitura de caixa de componentes deveríamos ter um segundo registo de segurança, para em caso de perda de ligação WiFi não seja possível o registo de verificação na base de dados. Esta é outra das razões de necessidade de utilização de um relógio em tempo real que escrevesse a hora e data de cada verificação para o cartão MicroSD.

Para comunicar utiliza o protocolo de comunicação I2C para interagir com o ESP32.

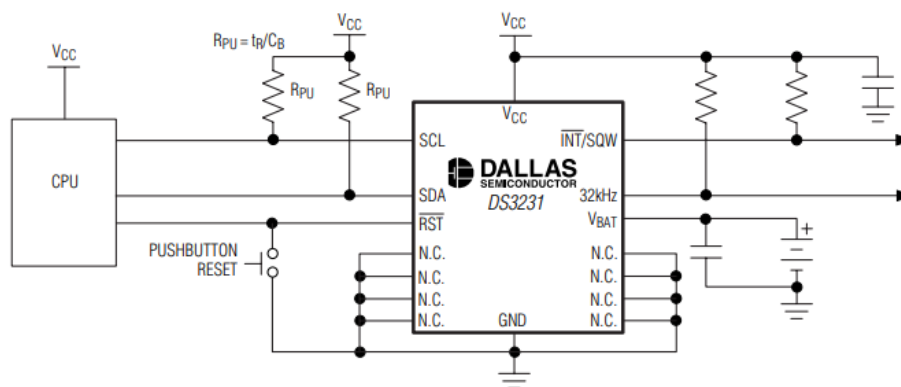


Figura 21 - Esquemático interno da RTC [24].

3.3.1.4. Teclado Matrix

Optou-se por utilizar um teclado uma vez que se achou necessária a introdução de ordens por parte do utilizador manualmente, a informação do sistema Check Box introduzida através deste, apenas passa para o cartão o novo registo de informação, sendo apenas local e não introduz nova informação remotamente na base de dados, sendo possível fazer verificação de caixas, novos registos de *tags* designadas como caixa ou componentes.

Teclado matricial da Figura 22, que é usado no projeto tem dezasseis botões, no qual o seu circuito interno é composto por 4 x 4 linhas de contacto em linhas e colunas respetivamente, cada botão representa um ponto de ligação entre uma das linhas com uma das colunas, quando um botão é pressionado, a linha e a coluna mudam de estado lógico sendo que estão conectadas a pinos diferentes do ESP32, assim assinalando consoante a programação e associação feitos de cada pino do microcontrolador para cada linha matricial do teclado é possível detetar que um botão específico foi pressionado.

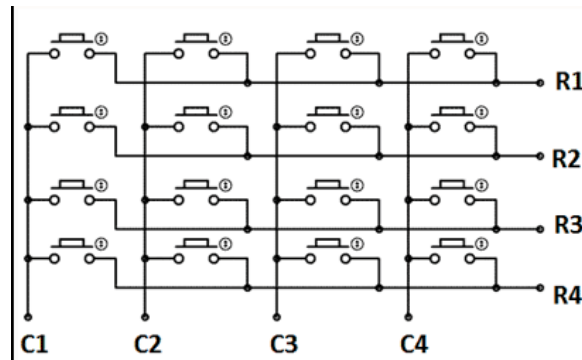


Figura 22 . Esquema teclado matrix [25].

O microcontrolador ESP32 tem resistências internas acopladas próprias nos pinos *Input/Output* (I/O), pelo que não houve necessidade de usar resistências externas com o circuito para ligação do teclado com ESP32.

O teclado utilizado para no nosso sistema, Figura 23 tem a função de interface entre o utilizador e microcontrolador, daí o utilizador pode navegar pelo menu do programa, dando instruções de resposta.



Figura 23 - Teclado matrix.

3.3.1.5. Display OLED

Achou-se necessária a visualização local através de um display LCD que auxilia na seleção e recolha de informação no módulo local, em que é possível visualizar os diferentes menus e submenus, bem como informação para execução de instruções para o microcontrolador ESP32. É também imprescindível o seu uso para visualização de informação e instruções feitas localmente, assim não é necessário a visualização ou execuções de instruções através do *dashboard* Node-red, caso assim seja mais prático.

Modulo de display OLED monocromático branco com dimensão de 0.96 polegadas e resolução de 128x64 pixels, tornando bastante económico em termos de potência consumida, tem um alto contraste de imagem e excelente nitidez, como se apresenta na Figura 24.



Figura 24 - Display OLED.

Este módulo serve para interface de visualização para com o utilizador sendo que opera como os módulos RTC através de I2C com o microcontrolador, com um endereço de *Slave* diferente.

Cujo um módulo I2C é externamente conectado ao MCU do módulo do LCD, deste modo permite a conexão entre o microcontrolador ESP32 e o display LCD através de I2C, sendo a ponte de ligação entre interface MCU do display LCD e o microcontrolador.

No qual o I2C apenas conecta com os pinos D [0:2] da interface MCU e utiliza os pinos de controlo RES# E D/C#, representado na Figura 25.

Pin Name Bus Interface	Data/Command Interface								Control Signal				
	D7	D6	D5	D4	D3	D2	D1	D0	E	R/W#	CS#	D/C#	RES#
I ² C	Tie LOW					SDA _{OUT}	SDA _{IN}	SCL	Tie LOW			SA0	RES#

Figura 25 – Registo de modulo para comunicação I2C [26].

Pode haver duas instruções de funcionamento, tendo em conta o estado do pino D/C#, consequentemente são executados comandos nas linhas D [0:2] ou estas linhas serem usadas para escrita de dados.

Numa situação de transmissão de dados os pinos D1 e D2 correspondem à linha SDA do protocolo I2C, sendo que D1 é usado para a transmissão e D2 é usado para a receção de dados.

A linha SCL ocupa o pino D0 que sincroniza a transmissão da linha SDA durante o processo. Os pinos de controlo D/C# tem o endereço *Slave* e a sua seleção é feita consoante o estado do bit SA0. O pino RES# é utilizado para inicialização e reset do dispositivo.

3.3.1.6. MicroSD adapter

É utilizado um adaptador de cartão MicroSD, como o da Figura 26, que faz comunicação com os pinos do ESP32 através do protocolo de comunicação SPI.

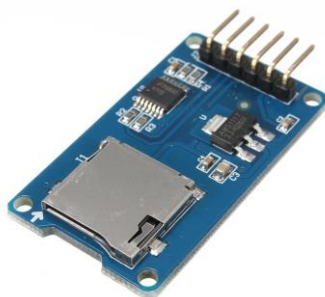


Figura 26 - MicroSD adapter.

A Tabela 4 representa a atribuição dos pinos de um cartão microSD.

Pin #	Name	Type	Micro SD Description
1	RSV		Reserved
2	CS	I	Chip Select (neg true)
3	DI	S	Data In
4	VDD	S	Supply Voltage
5	SCLK	I	Clock
6	VSS	S	Supply Voltage Ground
7	DO	O	Data Out
8	RSV	I	Reserved



Tabela 4 - Pinos de um cartão microSD [27].

Na Tabela 5 representa a atribuição dos pinos do adaptador de cartão microSD para o os pinos do microcontrolador.

Este cartão guarda as listas de *tags* no sistema, nos quais dois ficheiros são apenas de registo de informação em que outros dois ficheiros têm o ID das *tags* do sistema, num desses ficheiros constam apenas as *tags* de caixa, e no outro ficheiro estão todas as *tags* dos componentes que estão associados a cada caixa.

Pinos adaptador cartão SD	Pinos ESP32 WROOM 38	Informação dos pinos
CS	5	Linha para seleção do <i>Slave</i>
SCK	18	Linha com relógio de sincronização
MOSI	23	Linha de transmissão de dados do <i>Master</i> para <i>Slave</i>
MISO	19	Linha de transmissão de dados do <i>Slave</i> o <i>Master</i>

Tabela 5 - Ligação do microSD ao ESP32.

3.3.1.7. Raspberry Pi 3B

O Raspberry Pi 3b é como um minicomputador de tamanho reduzido, mas que contém componentes integrados. É possível ligar um teclado, display, rato, um cabo de internet entre outros ao Raspberry, fazendo com que se pareça um computador normal, sendo que também consegue fazer o papel de microcontrolador, no entanto não utilizamos esse método. Podem ser visualizadas na Figura 27 todas as características que o Raspberry Pi 3b contém. O Raspberry Pi necessita de um sistema operativo como todos os computadores, que possibilita aplicações usarem o hardware do próprio Raspberry Pi. O sistema operativo do Raspberry Pi é baseado em Debian, Linux.

Raspberry Pi também pode ser usado como *webserver* numa rede local ou na internet. No nosso projeto o Raspberry Pi vai ser usado como *webserver*, que vai estar ligado à internet do IPLeiria (labs), fazendo host ao *webserver*.

Foi instalado no Raspberry Pi a base de dados phpMyAdmin e a plataforma Node-RED, para dar host ao *webserver*.

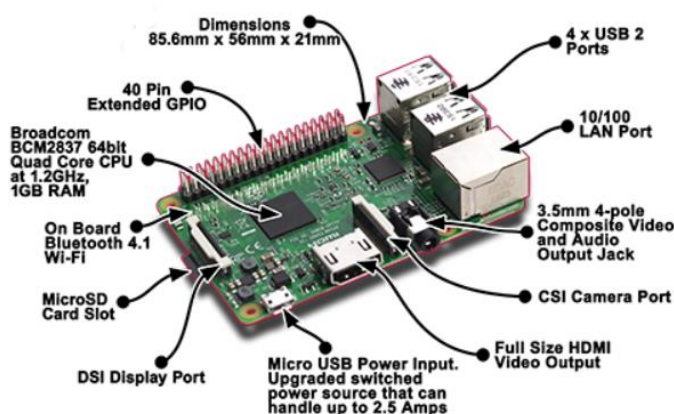


Figura 27 - Raspberry Pi 3B [28].

3.3.2. Sistema Check IN/OUT

Este Sistema foi planeado para ser implementado em paralelo com o sistema Check Box, no entanto, não foi feita pesquisa nem proposto um método de implementação detalhado na fase inicial de implementação do projeto, no entanto o seu conceito foi ponderado pelo uso de dois leitores ou antenas para deteção e verificação de passagens de material pela porta, correspondendo a situações de saída ou entrada do laboratório. Deste modo deu-se prioridade e mais atenção ao sistema Check Box, pelo que foi deixado a implementação do Check IN/OUT para depois da completção do sistema Check Box.

Apesar de este sistema final ter sido usado um leitor RFID UHF JRD-4035 igual ao do sistema Check Box, o leitor indicado para este sistema teria de ser de outro tipo específico, nomeadamente com mais distância de leitura e capacidade para suportar duas antenas externas para colocação, deteção e diferenciação de situações de entrada ou saída, um exemplo teria de ser do tipo UHF com frequências na ordem de 2.4GHz, em que tivesse um alcance mínimo em volta dois metros, para que quando colocado na entrada e saída da porta do laboratório fosse capaz de cobrir na totalidade a área da porta, assim garantir uma leitura caso ocorra uma entrada ou saída de material identificado.

Este sistema foi implementado com o leitor incorreto para apenas testagens do cenário real em que se usasse um leitor indicado como já referenciado e colocado ambas as antenas na entrada e saída da porta.

Houve falta de tempo e incapacidade de pesquisa e planeamento para adquirir um leitor RFID adequado para o efeito, este sistema experimentalmente foi implementado com um leitor RFID conectado a um microcontrolador ESP32.

Foram também feitas tentativas para implementar dois leitores RFID em simultâneo no microcontrolador ESP32, deste modo pretender simular duas antenas de o leitor RFID desejado como o cenário real como é ilustrado na Figura 16.

Foi totalmente pensado as possibilidades e feitas várias tentativas para a implementação dos dois leitores simultaneamente no mesmo microcontrolador ESP32, pelo que no final não foi possível executar corretamente o cenário experimental pretendido.

As tentativas feitas tiveram os seguintes procedimentos:

1º-Tentativa: Os leitores UHF JRD-4035 usados para simulação e como referido anteriormente utilizam o protocolo UART de comunicação, pelo que o ESP32 Wroom apenas tem uma interface de comunicação UART disponível, sendo a outra usada para comunicação série com o Serial Port do IDE Arduino.

Deste modo não é um método acessível e direto a configurar internamente o microcontrolador para manipulação de uma interface UART ou alterar o existente, assim não é possível ter duas interfaces UART agregadas a dois leitores distintos. Foram também procuradas diversas soluções que pudessem ir ao encontro à solução, no entanto apenas foi descoberta a possibilidade de alterar a associação de diferentes pinos para uma interface UART, porém de nada valeu neste caso, uma vez que para uma correta e distinta leitura dos dois leitores RFID. Apesar de ser possível a comutação da associação de pinos diferentes para uma interface UART, estes de qualquer modo não poderiam usar a mesma interface em simultâneo, pois haveria colisão nas leituras entre leitores e não haveria uma distinção coerente entre os leitores, logo não se poderia identificar qual o leitor de entrada ou saída a proceder uma leitura.

2º-Tentativa:

Uma segunda tentativa envolve o uso de uma *relay* de comutação de sinal, juntamente com os dois leitores RFID ligados diretamente aos pinos de interface UART do ESP32. A alimentação de +5V dos leitores é ligada a cada saída da *relay* que comutam de estado entre ON e OFF.

Nomeadamente estas saídas são designadas *Normally Open* (NO) e *Normally Connected* (NC), significa que, consoante o estado do pino do microcontrolador que está ligado à entrada da *relay*, a sua saída de referência, neste caso +5V comuta entre NO e NC, fazendo com que cada leitor ligado à saída estivesse alimentado desfasadamente, consoante as instruções no ESP32, procedeu-se a tal para apenas um leitor estivesse alimentado consequentemente cada um faz leituras desfasadamente e sem interferência.

O algoritmo de implementação é o seguinte:

A alimentação da *relay* está constantemente a comutar entre os leitores um e leitores dois.

Caso um dos leitores faça uma leitura, essa leitura é registada, sendo de seguida uma instrução dada pelo ESP32 para comutação do estado da *relay*, colocando o leitor atual sem alimentação e o outro alimentado, deste modo uma segunda leitura é feita pelo leitor agora alimentado, em caso de uma leitura bem sucedida pelos dois leitores, é feito o registo de todas as *tags* lidas por eles e registado a ordem com que passou pelos leitores, sendo as situações referentes a estado de saída ou entrada consoante a sua ordem leitura, posteriormente essa informação de *tags* de componentes detetados, bem como o seu estado de saída ou entrada são enviadas para a plataforma Node-RED e inseridas na base de dados.

No entanto esta implementação também não foi bem sucedida, uma vez que cada leitor tem na sua constituição interna condensadores.

Este facto leva a que quando haja uma comutação na alimentação dos leitores estes ainda tenham energia armazenada, fazendo com a energia de alimentação para o chip dos leitores RFID não desvanece de imediato num pequeno período de tempo, causando que o leitor desligado da alimentação direta ainda

esteja a ser alimentado pela carga do condensador e assim a possibilidade de executar uma leitura até este se encontrar totalmente descarregado, levando ao problema inicial de identificação de qual os leitores a executar leituras.

Solução proposta:

Uma solução que resolveria por completo este problema, cujo as duas tentativas não resolveram é o facto de utilizar dois microcontroladores para dois leitores, assim não haveria o problema de identificação nas leituras RFID e cada interface UART, seria ligada a dois dispositivos diferentes e colocados em cada sitio, estes dois microcontroladores poderiam ser ligados pelo método *peer-to-peer*, através de protocolos de comunicação como o BLE, ou até mesmo WiFi, comunicando e dando enviando informação bidireccionalmente, após executada uma leitura em um dos, posteriormente o outro microcontrolador estaria a aguardar uma segunda leitura, através deste método é possível proceder à leitura dos componentes das *tags* identificando caso seja uma entrada ou saída .

4. Desenvolvimento da PCB

Com a conclusão dos testes em breadboard, seguiu-se a fase do desenho da PCB do sistema Check Box, esquematizado. O Software utilizado tanto para o desenho dos esquemáticos como para os das PCBs foi o Proteus8 profissional, onde foi realizado um estudo prévio e adaptação ao programa.

Na primeira fase da construção da PCB foi feito um esquemático com todo o hardware do sistema Check Box, semelhante ao que estava na breadboard, onde foi adicionado um condensador para cada componente, apresentado na Figura 28.

Na Figura 29, encontra-se o circuito completo para a PCB, em Proteus, onde foram colocados todos os componentes criados no esquemático, com as medidas corretas dos componentes de modo a calcular o tamanho da PCB, com o objetivo de minimizar o tamanho da mesma. Foram desenhadas as pistas desde os componentes até ao ESP32, corretamente como tinha sido planeado no esquemático.

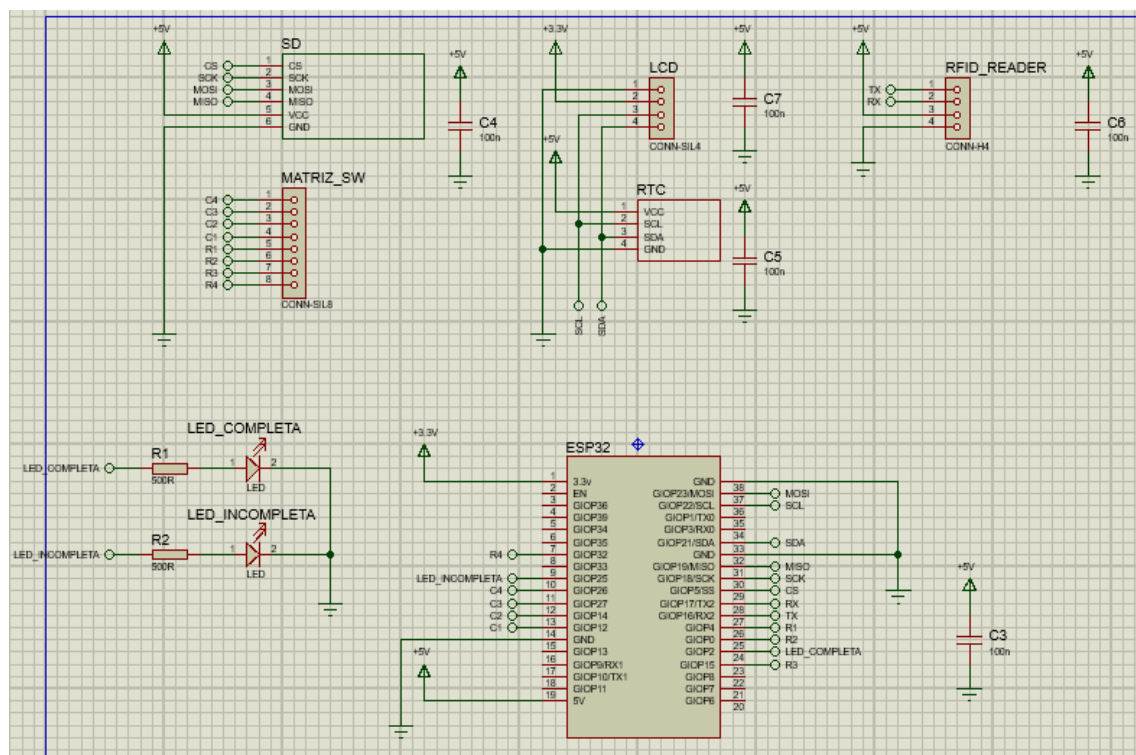


Figura 28 - Esquemático completo da PCB.

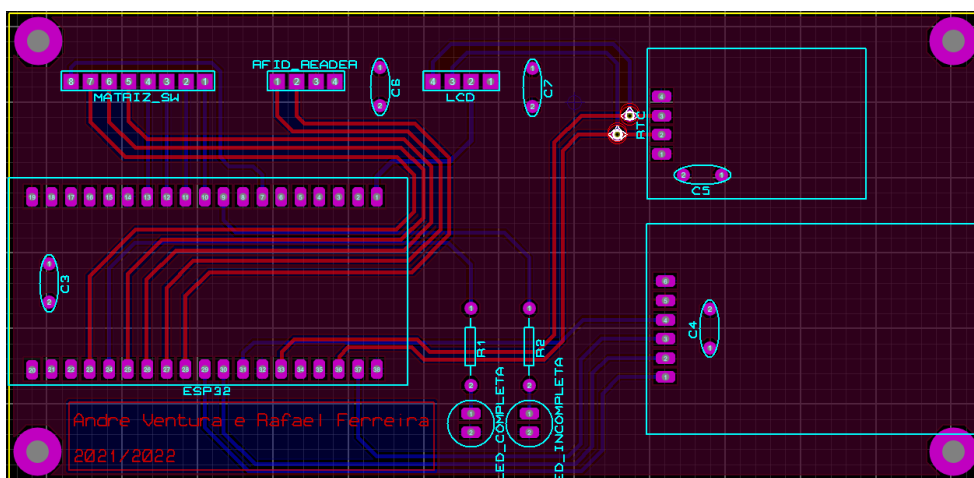


Figura 29- PCB no Proteus.

O hardware utilizado para a construção da PCB está sumarizado na Tabela 6.

Designação	Quantidade	Informação
Header ESP32	1	Soldados na PCB consoante o <i>Package</i> do ESP32 para um bom encaixe na PCB
Header RTC	1	Headers onde é conectado o modulo da RTC
Header LCD	1	Headers onde são ligados fios ao LCD
Header Teclado Matrix	1	Headers onde são ligados os fios do Teclado
Header RFID_READER	1	Headers onde são ligados os fios do leitor
Header microSD Adapter	1	Header onde é conectado o microSD
Header Leds	2	Headers onde são ligados fios aos leds
Condensadores	5	C1=C2=C3=C4=C5= 100nF
Resistências	2	500Ω

Tabela 6 - Lista de componentes do sistema Check Box.

Com o desenho da PCB finalizado, foram efetuadas algumas alterações em relação ao desenho dos proteus, para ficar de acordo com as condições desejadas. Foi necessário furar, colocar vias e soldar todos os componentes, por fim conectar os componentes na PCB de forma desejada, como está representado na Figura 30. O ESP32 é alimentado a 5V com um cabo Micro USB. O passo seguinte passou por testar todos os componentes na placa e verificar se a PCB estava a funcionar como previsto.

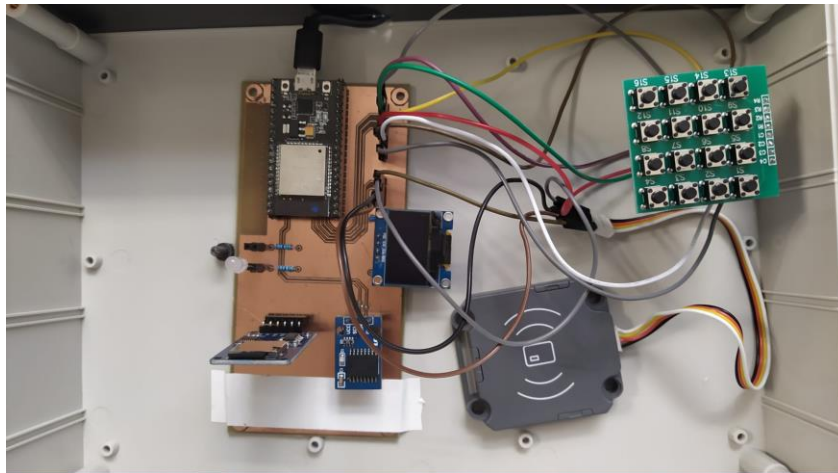


Figura 30 – PCB final com o hardware no interior da caixa.

Por fim, de forma a tornar o projeto mais apresentável, robusto e a englobar todas as ligações, foi comprada uma caixa (Figura 31) com as medidas necessárias para colocar o LCD, teclado matrix, dois leds e uma caixa de um laboratório. Inicialmente o teclado matrix e o LCD eram para ficar de lado, mas optou-se por colocar em cima da caixa para ser mais fácil a visualização do LCD, devido ao seu tamanho ser reduzido.



Figura 31 - Caixa final Check Box

5. Software

Neste capítulo é explicado como foi realizada toda a parte do software, *webserver* e IoT do projeto, e todas as suas funcionalidades.

5.1. Funcionalidade do Sistema Check Box

Para implementação deste sistema foi utilizado o IDE Sloeber Eclipse que utiliza a plataforma de programação em Arduino.

Um IDE permite que programadores consolidem diferentes aspetos de implementação e escrita de código para um programa de computador, sendo que permite a edição de código de um modo mais fácil, favorecendo no tempo de programação e deteção de erros, sendo que as maiores vantagens são verificação automática de erros de sintaxe do código através de compiladores incorporados, ferramentas extra executivos que permitem automaticamente após a programação do código incorporar diretamente na memória do computador, neste caso o ESP32.

Procede-se à inclusão de bibliotecas que está representado na (Figura 1 do Anexo1), existentes em código, disponibilizando código extra e funções fundamentais que nos são úteis, sendo uma excelente ferramenta de auxílio na programação e interação com o hardware.

RFID_command.h – Esta biblioteca contém as funções de execução de comandos do Leitor RFID, no qual as suas variáveis internas e certas funções removidas, juntamente com algumas bibliotecas que esta incluía. Feito deste modo para que o código pudesse ser compilado sem erros para com a board ESP32 DEV Module.

Stdio.h - É um cabeçalho declarado no programa que faz parte da biblioteca de C na qual tem à sua posse um conjunto de funções com determinados tipos de variáveis e ferramentas auxiliares de entrada e saída e definições de variáveis

Wire.h - É um header que é necessário chamar para estabelecer comunicação com os dispositivos que utilizam protocolo I2C, sendo que **RTClib.h**, **Adafruit_SSD1306.h** necessitam desta biblioteca para funcionar corretamente.

SPI.h - Biblioteca desenvolvida para boards Arduino e usada para estabelecer comunicação SPI com o adaptador de leitura microSD.

SD.h - Esta biblioteca é utilizada como base e ferramenta auxiliar de manipulação de objetos dentro dos ficheiros do cartão micro-SD sendo que esta é fundamental integrar pois é a principal funcionalidade deste programa, a biblioteca **FS.h** também necessita da integração desta.

FS.h - Esta biblioteca igualmente com a **SD.h** integra funcionalidades e ferramentas usadas para comunicar com o cartão microSD.

String.h - Este cabeçalho faz parte da biblioteca *strings* da biblioteca de C++. no qual é usado para definição de variáveis do tipo *String* a partir de variáveis base tipo *char* e mais alguma manipulação de código, as *tags* associadas no nosso programa são variáveis do tipo *String*.

RTClib.h - É a biblioteca utilizada para definir horário e data da RTC e também faz a leitura destes dados para o programa.

Wifi.h - Esta biblioteca permite suporte na ligação à rede Wi-Fi.

HTTPClient.h - Esta biblioteca necessita de uma conexão a uma rede Wi-Fi e assiste nas comunicações através do protocolo HTTP para envio de mensagens para o nosso servidor Web

Adafruit_GFX.h - É a biblioteca base para todo o tipo de livrarias de displays gráficos, e é necessário o uso desta com a biblioteca específica do nosso display **Adafruit_SSD1306.h**

Adafruit_SSD1306.h - Biblioteca especificada para operar com o display OLED, no qual a partir de uma chamada de simples através de funções é possível imprimir informação no seu ecrã.

Existem várias funções que implementámos para organização do programa, iremos de um modo geral destacar as mais importantes, deste modo explicaremos também o seu funcionamento e integração no projeto, está representado na Figura 2 do Anexo 1.

O programa é inicializado pela declaração de variáveis de inicialização globais, funções, e inicialização de livrarias incorporadas.

Devido à grande diversidade de hardware utilizado, o que previsivelmente gera uma grande quantidade de instruções no programa, foram criadas diversas funções para maximizar a sua organização.

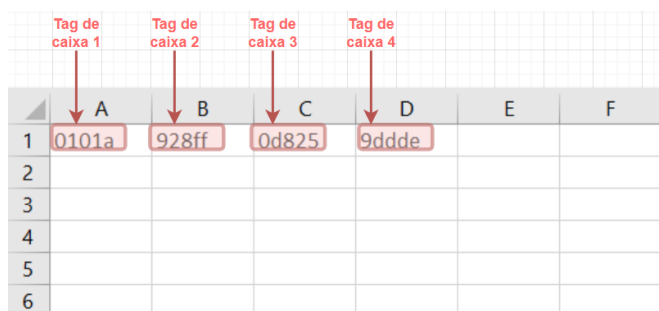
O programa foi efetuado da seguinte forma:

A primeira instrução realizada no decorrer do programa é a leitura de ficheiros no cartão microSD na qual tem armazenado o ID de todas as *tags* do sistema Check Box em formato de ficheiros. Cada ficheiro tem um propósito diferente, no qual existem dois ficheiros para associação de *tags* e outros dois para registo de informação.

Existem duas designações diferentes para *tags* no nosso projeto, no qual existem *tags* de caixa e *tags* de componente. Uma *tag* caixa é referente às *tags* acopladas na própria caixa que acomoda material. Uma *tag* de componente é referente a *tags* acopladas aos componentes inseridos na caixa. Cada componente no laboratório tem uma caixa de material associada, pelo que cada *tag* de componente tem uma *tag* de caixa associada.

5.1.1. Ficheiros cartao microSD

Ficheiro **taglist.CSV** - Contem apenas IDs das *tags* de caixa do sistema e estão sempre representados dentro do ficheiro na 1ª linha da horizontal e separadas em diferentes colunas. O ficheiro excel referente as *tags* caixas, pode ser observado na Figura 32.



	Tag de caixa 1	Tag de caixa 2	Tag de caixa 3	Tag de caixa 4		
	A	B	C	D	E	F
1	0101a	928ff	0d825	9ddde		
2						
3						
4						
5						
6						

Figura 32- Ficheiro excel das *tags* caixas.

O **Ficheiro boxlist.CSV** contém os IDs das *tags* de caixas e componentes existentes no sistema I, em que na 1ª coluna estão apenas as *tags* de caixa e nas seguintes colunas para a mesma linha de uma determinada *tag* de caixa, estão as *tags* dos componentes que lhe estão associados. O ficheiro excel referente as *tags* caixas e componentes, está na Figura 33.

	A	B	C	D	E
Tag de caixa 1	1	0101a	017bc	0c97c	0c4ff
Tag de caixa 2	2	928ff	11003	00d57	
Tag de caixa 3	3	0d825	4768y	64795	
Tag de caixa 4	4	9ddde	89zy6	k7yt4	
	5				
	6				

Figura 33 - Ficheiro excel das Tags caixas e Componentes.

A partir dos ficheiros **taglist.CSV** e **Boxlist.CSV** são lidas e copiadas as *tags* de caixa para variáveis no programa, de modo a auxiliar nas operações de manipulação de informação e não operar constantemente com o cartão, abrindo e fechando os mesmos ficheiros. Bastaria apenas a leitura do ficheiro **taglist.CSV**, mas foi implementado a leitura do segundo para comparação da informação se encontra coerente, Figura 34.

```

ultima tag lida:64676
Tags main registadas no ficheiro 1:
64674
92900
0e52a
64676
lendo do ficheiro2 as tags que sao de caixa:
Ultima posicao do ficheiro2:64676
:Posicao:58
Tags main registadas no ficheiro 2:

64674
92900
0e52a
64676

```

Figura 34 - Tag registada.

Ficheiro registos.CSV- Contém todos os registos referentes a uma leitura de verificação, na qual é registada a hora, data, *tag* de caixa principal, *tags* presentes na leitura e estado da caixa referente à falta de material, Figura 35.

12:53:16	30/6/2022	Caixa:6467	tags presentes:nenhuma	Estado da caixa: Incompleta
12:53:59	30/6/2022	Caixa:6467	tags presentes:nenhuma	Estado da caixa: Incompleta
12:54:48	30/6/2022	Caixa:6467	tags presentes:nenhuma	Estado da caixa: Incompleta
12:59:44	30/6/2022	Caixa:9290	tags presentes:nenhuma	Estado da caixa: Incompleta

Figura 35 - Ficheiro das *tags*.

Utilizou-se um sistema de seleção de opção por parte do utilizador para interagir com o programa de um modo facilitado, no qual o programa é iniciado através da chamada de um menu principal programa que por defeito está sempre a correr, até que seja inserida uma tecla de seleção de menu, que está representado na Figura 3 do Anexo 1.

A seleção de opção de menu pode ser realizada de duas formas, em modo de utilizador nomeadamente o teclado matricial ou a plataforma Web Node-RED que irá ser explicado adiante, pelo que também utilizamos uma terceira interface, nomeadamente o UART0 do ESP32 permite estabelecer comunicação série entre IDE e esta interface por defeito do IDE sendo que não pode ser utilizada para outro propósito. Posteriormente o programa IDE permite a adaptação e associação da interface série através do protocolo UART com uma porta *Universal Serial Bus* (USB) ligada a um PC (Porta COM), adicionalmente sendo por este método que é possível proceder também ao upload do código devidamente compilado pelo programa para o PC Figura 36.

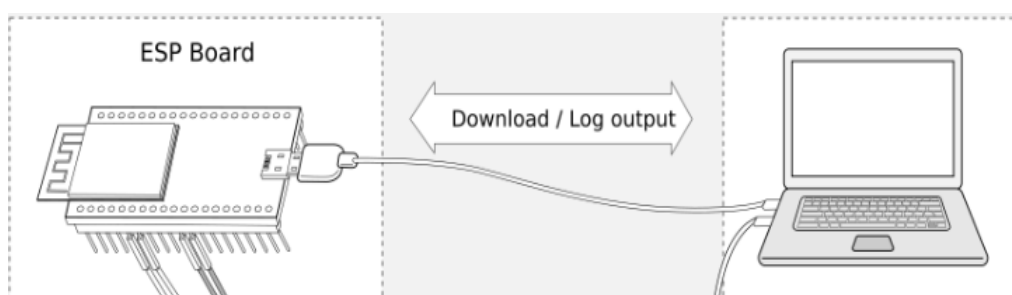


Figura 36 - Conexão do componente ESP com o computador [30].

Após a introdução de uma opção válida por parte do utilizador na seleção do menu, as funções correspondentes à seleção irão ativar.

No seguinte tópico serão abordadas quais as opções principais e explicado o seu funcionamento.

5.1.2. Verificação da caixa

Sendo este um dos principais requisitos, foi a primeira funcionalidade a ser implementada. Tem como principal função a verificação de material caso haja deteção de uma caixa, neste caso, procede à leitura dos seus componentes caso estejam presentes na leitura.

No modo programador é possível visualizar a informação completa de uma verificação de caixa nomeadamente:

- *Tags* que não constem nos ficheiros *taglist.CSV* ou *Boxlist.CSV* são consideradas como *tags* não registadas.
- *Tags* de caixa detetadas para além da lida são também detetadas como *tags* de outra caixa.
- *Tags* detetadas que não pertencem à *tag* de caixa lida mas que estejam associadas a outra *tag* de caixa são também reconhecidas como sendo de outra caixa.
- *Tags* que estão em falta
- O estado da caixa lida: completa ou falta de componentes
- A Figura 37 mostra a deteção de várias *tags* a partir de uma verificação e a sua associação no sistema.

```
Informacao de todas as tags lidas encontra-se abaixo:
TAG:64676 STATUS:
Registada no cartao como sendo caixa
-----
TAG:92900 STATUS:
Registada no cartao como sendo caixa
-----
TAG:64674 STATUS:
Registada no cartao como sendo caixa
-----
TAG: 0101aSTATUS: Nao registada no sistema
-----
TAG: 64613STATUS: Registada no cartaoTAG DE CAIXA ASSOCIADA: 92900
-----
TAG: 01636STATUS: Nao registada no sistema
-----
TAG: 017c0STATUS: Nao registada no sistema
-----
TAG: 64769STATUS: Registada no cartaoTAG DE CAIXA ASSOCIADA: 92900
```

Figura 37 - Informação de uma verificação de caixa através da porta série.

O acesso à porta Serial monitor do é uma mais valia como ferramenta de auxílio para verificação do correto funcionamento do sistema, tendo sido dimensionado para a pessoa programadora, deste modo através de informação extra na porta Serial para o PC, é possível verificar pormenorizadamente se a associação de ficheiros no sistema está a funcionar corretamente, após inserção de novas *tags* no sistema, quer sejam inseridas internamente através do menu, quer através da inserção de *tags* externamente fora do sistema através da inserção de *tags* manualmente nos ficheiros no cartão SD.

Modo utilizador

O modo utilizador é apenas uma forma simplificada de recolher informação de uma verificação de caixa de material feita de uma caixa, e é esta a interface direta que o utilizador tem com o Check Box fisicamente.

O modo utilizador apenas imprime informação relativa a uma caixa, sendo que não entra em tantos detalhes como no modo programador. Porém provém da mesma fonte de código e da mesma função de verificação, no entanto decidiu-se não o fazer porque o módulo Display local apenas serviria como *tag*agem de material rápida e não necessitaria de informação para além estado do material da caixa, assim sendo no display é possível após uma verificação:

- Imprimir a caixa lida
- Imprimir o estado da caixa relativo à presença de material
- Sinalização luminosa a partir de LEDs consoante o estado da caixa relativo à presença de material.
- Opção de informação extra após leitura
 - Componentes presentes na leitura da caixa
 - Componentes em falta na leitura da caixa

O funcionamento geral da funcionalidade de verificação de material de uma caixa está representado num fluxograma na Figura 4 do Anexo 1.

5.1.3. Adicionar *tag* de caixa

Esta opção permite adicionar IDs de *tags* no sistema e associá-las a registo de caixas, deste modo o seu método pode ser ilustrado na Figura 38.

Sempre que seja executada a função de adicionar uma nova caixa ou componente na base de dados, é fundamental o ID da *tag* desse componente estar presente no sistema local, ou seja no cartão microSD nos ficheiros, caso contrário se o ID de uma *tag* não esteja registado localmente, também não será enviado para a base de dados em uma leitura de verificação de caixa, deste modo não é impressa nesta.

O algoritmo do sistema global está feito deste modo, em que é necessário a existência do registo local de uma determinada *tag* também ser reconhecida na base de dados. Logo é necessário adicionar localmente um ID de uma *tag*, caso se pretenda registar também na base de dados.

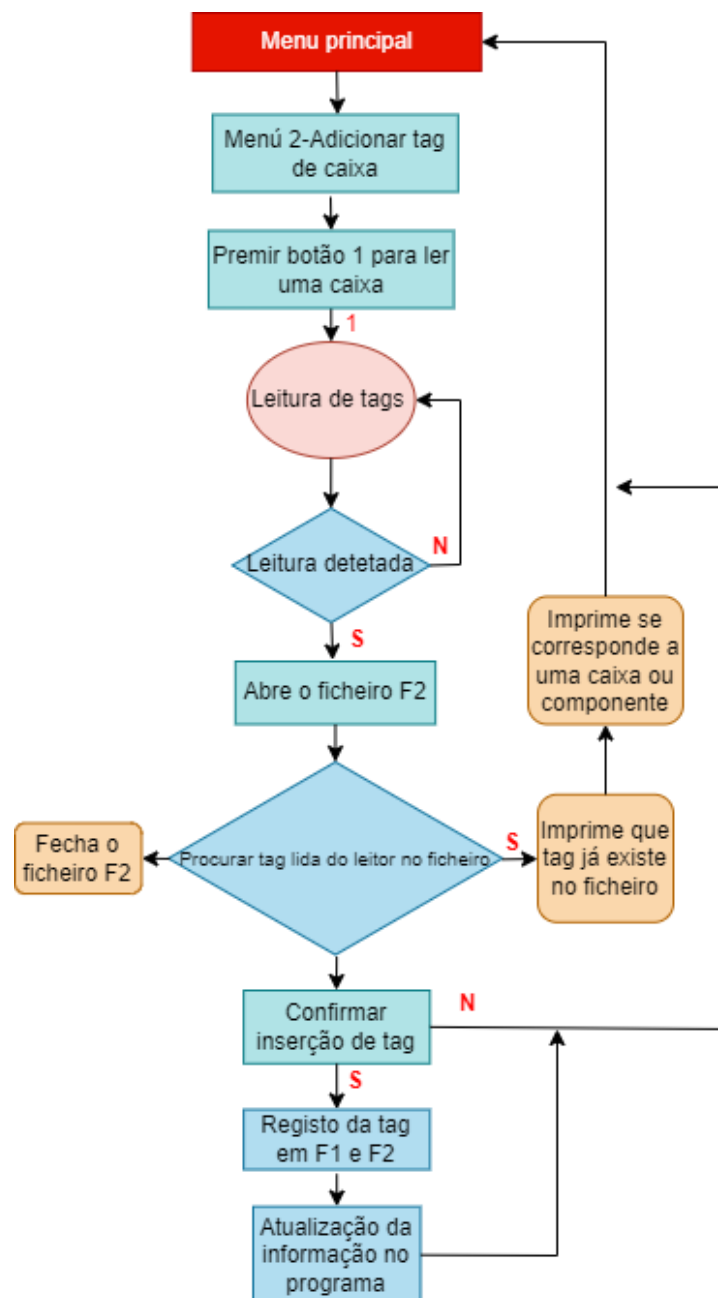


Figura 38 - Diagrama de adicionar *tag* da caixa.

5.1.4. Adicionar *tag* componente

Esta funcionalidade é simples, permite a leitura de uma *tag* singular e verificar qual a sua designação no sistema local, inexistente, *tag* de caixa, *tag* de componente. O diagrama de blocos encontra-se representado na Figura 39.

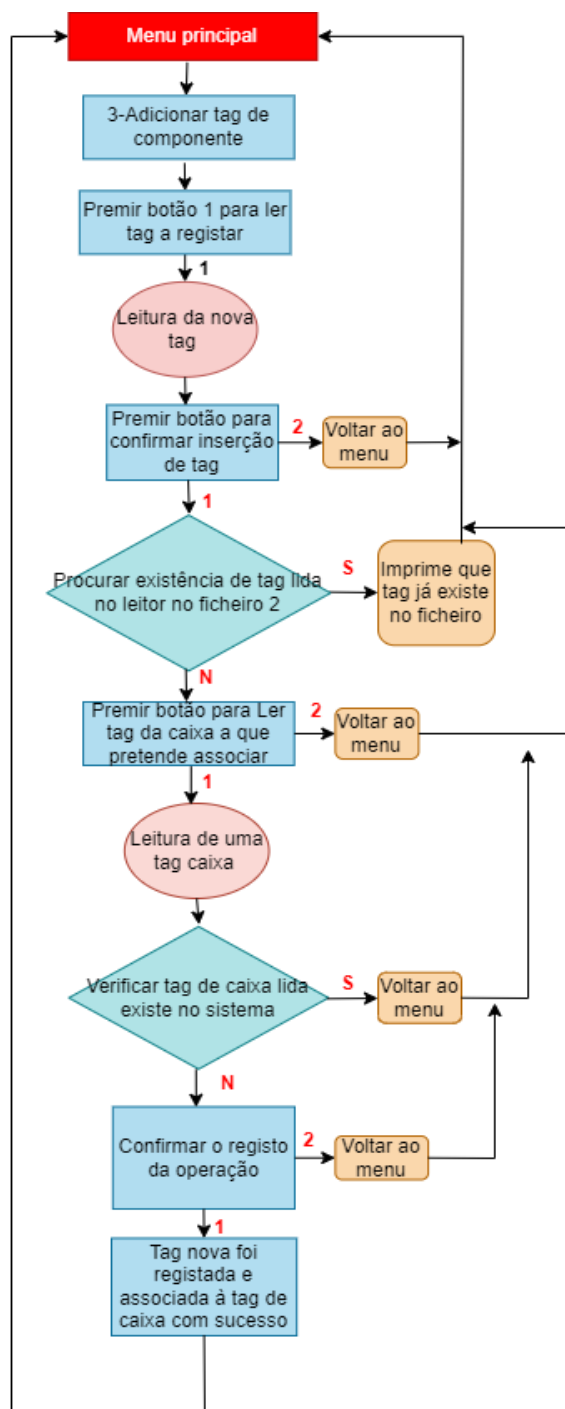


Figura 39 - Diagrama de adicionar *tag* de componentes.

5.2. Raspberry Pi 3B (WEBSERVER)

5.2.1. Processamento de dados

Processamento de dados consiste em realizar operações específicas num conjunto de dados ou informação, de acordo com o objetivo do utilizador, formando uma base de dados que reúne toda a informação. Neste projeto foi utilizada uma base de dados para guardar toda a informação das *tags* registadas e ao mesmo tempo é possível verificar, guardar todos os registos efetuados do projeto. Esta base de dados foi instalada no *webserver*, ou seja, localmente no Raspberry Pi 3B. Para aceder à base de dados é necessário indicar o *Internet Protocol* (IP) do *webserver* (Raspberry Pi) e está protegida com uma *password*. (o *login* é: root e a *password* é: smartlab)

A base de dados escolhida para o projeto foi a MySQL phpMyAdmin, como se verifica na Figura 40. O phpMyAdmin é uma ferramenta de administração gratuita e de código aberto para MySQL e MariaDB. Como um aplicativo da web escrito principalmente em *Hypertext Preprocessor* (PHP), tornou-se uma das ferramentas de administração MySQL, especialmente para serviços de hospedagem na web.

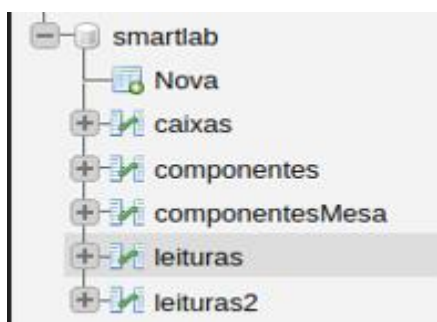


Figura 40 - Base de dados smartlab no phpmyadmin.

5.2.1.1. Criação da base de dados phpMyAdmin

Neste projeto foi criada uma base de dados, instalada no *webserver*, Raspberry Pi, que contem toda a informação das *tags*. O diagrama de blocos da base de dados criada está representado na Figura 41. Foram criadas três tabelas, onde a informação foi organizada por colunas, para registar as *tags* das caixas, componentes e componentesMesa do laboratório. As colunas têm toda a informação necessária de cada tabela. Foram criadas mais duas tabelas denominadas de leituras e leituras2, onde a tabela de leitura

guarda todos os registos efetuados, ou seja, todas as leituras do sistema Check Box. Por sua vez, a tabela leituras2 guarda todos os registos do sistema Check IN/OUT. Estes registos podem ser visualizados na plataforma Node-RED, na *dashboard* Check IN/OUT e em Registo de Leituras.

Existe uma condição nos componentes, pois esta tabela não tem a coluna laboratório, mas tem uma coluna que indica a *tag.caixa*. Cada componente está associado a uma caixa através da *tag*, da caixa, que é única. Por sua vez, cada caixa está associada a um laboratório, desta forma os componentes associados a essa caixa estão automaticamente associados ao laboratório da caixa. Assim é possível saber a que laboratório pertence qualquer componente.

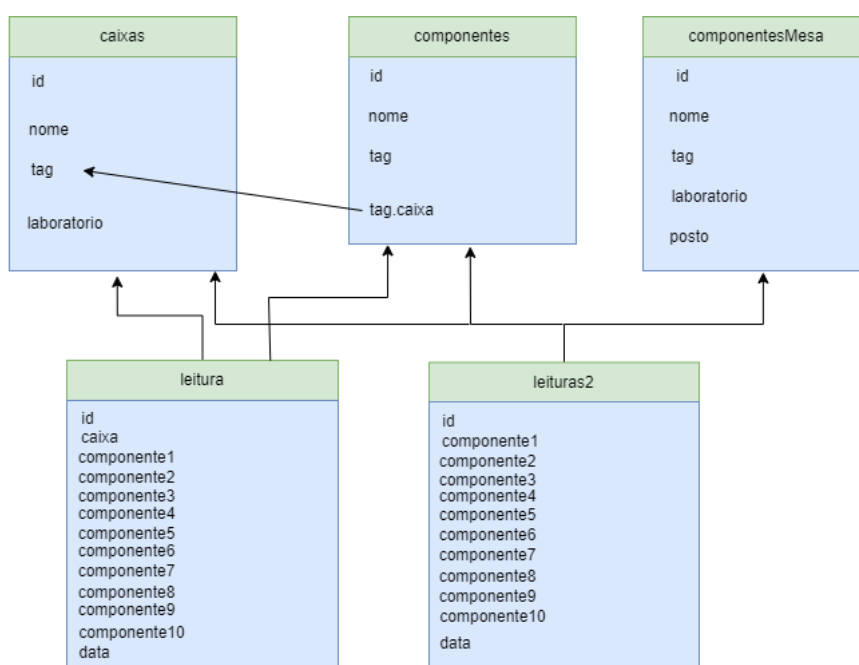


Figura 41 - Diagrama de blocos da Base de dados.

Na Figura 42, está representado um exemplo de como se cria uma tabela na base de dados, neste caso a tabela componentes. Foram criadas quatro colunas pois é o necessário para toda a informação dos componentes. A primeira coluna é o id, sendo este o que identifica o registo que é único para cada componente, neste caso. Cada componente tem o seu próprio id. Todas as tabelas criadas têm id que é um INT incremental. De seguida foram criadas mais três colunas, sendo elas para o nome do componente, a *tag* do componente e a *tag.caixa*, e são todas VARCHARS, ou seja, alfanumérico. Esta ultima coluna a *tag.caixa* está relacionada com a *tag* da caixa, pois cada componente esta associado a uma caixa através da *tag* da caixa.

```
CREATE TABLE componentes (  
  id INT PRIMARY KEY,  
  tag VARCHAR(100) DEFAULT '',  
  nome VARCHAR(200) DEFAULT '',  
  tagcaixa VARCHAR(100) DEFAULT '')
```

Figura 42 - Criação da tabela componentes.

A criação da tabela leituras, como se pode observar na Figura 43, é composta por treze colunas que são: o id INT incremental, caixa, dez componentes e a data(datetime). Nas tabelas leituras e leituras2 o id é bastante útil, pois através do id conseguimos facilmente ir buscar o último registo realizado. Uma caixa tem até oito componentes, sendo que foi realizada uma tabela com dez componentes para o caso de existirem componentes a mais na caixa e serem registados na mesma. Os componentes a mais podem ser vistos na *dashboard* Leituras no Node-RED, juntamente com toda a informação sobre eles, de modo a saber a qual caixa e laboratório a que pertencem.

```
CREATE TABLE leituras (  
  id INT PRIMARY KEY,  
  caixa VARCHAR(100) DEFAULT '',  
  componente1 VARCHAR(100) DEFAULT '',  
  componente2 VARCHAR(100) DEFAULT '',  
  componente3 VARCHAR(100) DEFAULT '',  
  componente4 VARCHAR(100) DEFAULT '',  
  componente5 VARCHAR(100) DEFAULT '',  
  componente6 VARCHAR(100) DEFAULT '',  
  componente7 VARCHAR(100) DEFAULT '',  
  componente8 VARCHAR(100) DEFAULT '',  
  componente9 VARCHAR(100) DEFAULT '',  
  componente10 VARCHAR(100) DEFAULT '',  
  data DATETIME DEFAULT '19000101');
```

Figura 43 - Criação da tabela leituras na base de dados.

5.2.2. Interface Node-RED

Node-RED, criado pela IBM *Emerging Technology*, é uma ferramenta de desenvolvimento *Open-Source*. Node-RED é uma plataforma, baseada em fluxos que foi desenvolvida para interagir com dispositivos de hardware, *Application Programming Interfaces* (APIs) e serviços online com o intuito de simplificar os sistemas IoT [31].

O Node-RED permite a programação através de uma interface gráfica. Para desenvolver uma aplicação, temos de ligar nós uns aos outros, onde cada um deles tem as suas próprias funções. No final esses Flows são armazenados em *JavaScript Object Notation* (JSON) simplificando a partilha do código. O formato JSON é utilizado para estruturar dados em formato de texto e permitir a troca de dados entre aplicações de forma simples e rápida devido a utilização de JavaScript. JavaScript é uma linguagem de programação de script de alto nível. Node-RED também usa node.js que é um software de código aberto que permite a execução de JavaScript. Node.js é ideal para ser executado em hardware de baixo custo, como por exemplo Raspberry Pi.

Na Figura 44 pode observar-se a *dashboard* principal do projeto no Node-RED, com todos os tipos de menu. No menu Principal estão os botões para verificar caixa, adicionar componente e adicionar caixa. Foram criadas várias abas de *dashboard* para um bom controlo do sistema.

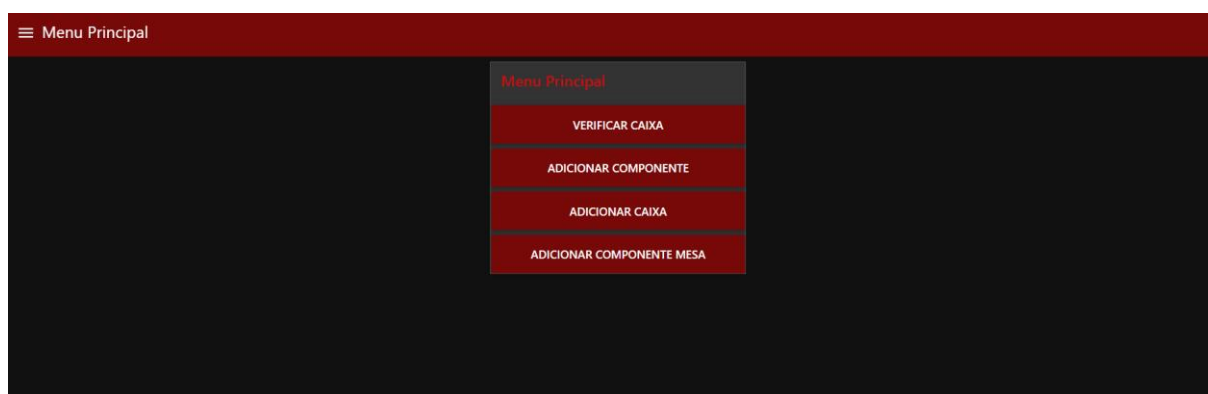


Figura 44 – Menu principal da *Dashboard*

Foi criada uma aba de *dashboard* com uma tabela que mostra todos os componentes que estão em falta (Figura 45), depois de verificar a caixa, caso existam, indica quais os componentes em falta. A aba da *dashboard* Registo de Leituras (Figura 46) mostra uma tabela com os registos da última verificação da caixa com a hora e data atual, do *webserver*, Raspberry Pi.

Material em falta	
Componentes em Falta	
nome	
Beardboard	
Cabo V	
Cabo P	
Cabo Crocodilo	
Alicato 1	
Alicato 2	

Figura 45 – Aba *dashboard* Componentes em Falta.

Registo de Leituras													
Registo de Leituras													
Caixa	Labora...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Comp...	Data
Caixa1	Laboratório 1	Multímetro 1	Multímetro 2										29-06-2022 ...
Caixa1	Laboratório 1	Beardboard	Multímetro 2	Alicato 2									27-06-2022 ...
Caixa1	Laboratório 1	Beardboard	Multímetro 2	Alicato 2									27-06-2022 ...
Caixa1	Laboratório 1	Beardboard	Multímetro 2	Alicato 2									27-06-2022 ...
Caixa1	Laboratório 1	Beardboard	Multímetro 2	Alicato 2									27-06-2022 ...
Caixa1	Laboratório 1	Beardboard	Multímetro 2	Alicato 2									27-06-2022 ...

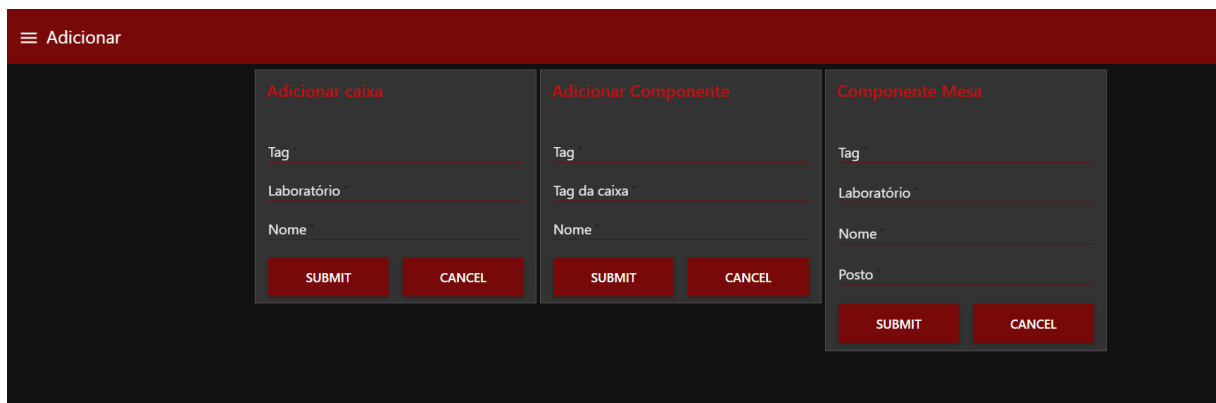
Figura 46 – Aba *dashboard* Registo de leituras.

Na aba da *dashboard* Leituras (Figura 47), é indicada qual a caixa e os componentes que foram lidos pelo leitor no momento da verificação da caixa e o estado da caixa, ou seja, se a caixa está completa ou incompleta.

Leituras	
Leituras	
Caixa: Caixa1,Laboratório 1	
Estado: Incompleta	
Componente1: Multímetro 1,Laboratório 1,Caixa1	
Componente2: Multímetro 2,Laboratório 1,Caixa1	

Figura 47 – Aba *dashboard* Leituras.

Foi criada uma aba da *dashboard* para adicionar, onde tem o formulário para adicionar uma caixa ou um componente. É necessário pressionar o botão de adicionar no menu Principal para obter a *tag* lida pelo leitor, de seguida ir à *dashboard* adicionar (Figura 48) e preencher o formulário para adicionar a caixa ou componentes para a base de dados. Por último, a aba da *dashboard* do sistema Check IN/OUT (Figura 49) apresenta uma tabela com todos os registos do material (nome, laboratório e data atual) que saia ou entre na sala, podendo ser caixas, componentes ou componentes de mesa.



The screenshot shows a dashboard titled 'Adicionar' with three distinct forms for adding items:

- Adicionar caixa:** Includes input fields for 'Tag', 'Laboratório', and 'Nome', with 'SUBMIT' and 'CANCEL' buttons.
- Adicionar Componente:** Includes input fields for 'Tag', 'Tag da caixa', and 'Nome', with 'SUBMIT' and 'CANCEL' buttons.
- Componente Mesa:** Includes input fields for 'Tag', 'Laboratório', 'Nome', and 'Posto', with 'SUBMIT' and 'CANCEL' buttons.

Figura 48 – Aba *dashboard* adicionar.



The screenshot shows a dashboard titled 'Check in/out' with a table displaying material check-in/out records. The table has columns for 'Nome', 'Laboratório', and 'Data'.

Nome	Laboratório	Data
Caixa1	Laboratório 1	29-06-2022 18:13:57
Multímetro 2	Laboratório 1	29-06-2022 18:13:57
Multímetro 1	Laboratório 1	29-06-2022 18:13:57
Osciloscópio	Laboratório 2	29-06-2022 18:13:57

Figura 49 – Aba *dashboard* Check IN/OUT.

5.2.2.1. Conexões entre nodes para caixa

No flow da Figura 50 é obtido o nome da caixa que foi lida pelo leitor e inserido no registo de leituras. Através da comunicação HTTP, é feito um POST da *tag* da caixa lida pelo leitor(*main_tag*). De seguida, essa *tag* é enviada através de uma *String* para duas funções diferentes, “nomecaixa” e “insert_leitura_caixa”, como se pode observar na Figura 50. No Node-RED a forma de passar os objetos, neste caso as *tags*, entre as funções é através de um método *msg.payload*. Para requisitar ou consultar a informação da base de dados são utilizadas *queries*. As *queries* permitem adicionar, remover e modificar qualquer tipo de dados.

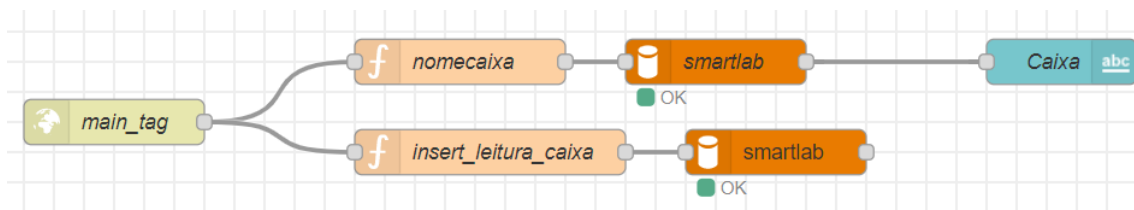


Figura 50 - Flow do nomecaixa e insert_leitura_caixa.

A primeira função (“nomecaixa”) faz a *querie*, representada na Figura 51, à base de dados que vai a tabela das caixas, verifica se existe alguma caixa com a *tag* que foi lida e devolve o nome e o laboratório desse caixa. Esta função pode ser visualizada na aba da *dashboard* Leituras.

```
var str = msg.payload;

msg.topic = "SELECT nome,laboratorio from caixas where caixas.tag='"+str+"'";

return msg;
```

Figura 51 - *Querie* efetuada para obter o nome e laboratório da caixa.

A segunda função (“insert_leitura_caixa”), representada na Figura 52, insere a caixa lida, a data e hora atual, ou seja, a data e hora do *webserver*, Raspberry Pi, na tabela leituras da base de dados. Esta função pode ser visualizada na aba da *dashboard* Registo de leituras.

```
var str = msg.payload;

msg.topic = "INSERT INTO leituras (caixa,data) VALUES ('"+str+"',NOW())";

return msg;
```

Figura 52 - *Query* para inserir na tabela leituras.

5.2.2.2. Conexões entre nodes para componentes

Na Figura 53 está representado o flow para obter o nome e o laboratório dos componentes. É recebido um post com as *tags* dos componentes numa *String* (“*tags_presentes*”).



Figura 53 - Flow de como obter o nome e laboratório dos componentes.

Na função é feito um *split* num *array*, para tirar as vírgulas e atribuir a cada valor da msg uma posição da array, como se observa na Figura 54. De seguida a função faz a *query* à base de dados, onde irá buscar o nome do componente, o nome e o laboratório da caixa à qual o componente está associado (Figura 55). Este processo é repetido para oito componente devido ao facto de cada caixa ter oito componentes.

```
var str = msg.payload;

var parts = str.split(",");

msg.value1 = parts[1];
msg.value2 = parts[2];
msg.value3 = parts[3];
msg.value4 = parts[4];
msg.value5 = parts[5];
msg.value6 = parts[6];
msg.value7 = parts[7];
msg.value8 = parts[8];

return msg;
```

Figura 54 - Modo de passar *tags* lidas entre funções.

```
msg.topic = `Select nome,(Select laboratorio from caixas where caixas.tag=componentes.tagcaixa) as laboratorio,
(Select nome from caixas where caixas.tag=componentes.tagcaixa) as caixa
from componentes where tag='"+componente1+"'`;
```

Figura 55 - Nome e laboratório dos componentes.

A função “insert_leitura_comp” é responsável por fazer a *querie* que vai atualizar os componentes lidos pelo leitor no registo criado pela leitura da caixa, na tabela leituras, ou seja, primeiro é lida a caixa e inserido o registo na tabela leitura, de seguida atualiza esse registo com os componentes na tabela leituras. O flow desta função está representado na Figura 56.



Figura 56 - Flow para inserir componentes e verificar componentes em falta.

Após concluir o inserir e atualizar o registo de leitura, na função (“registo de leituras”), é realizada a *querie* onde são devolvidos todos os registos da tabela de leituras, ordenados por ordem decrescente da data, ou seja, da leitura mais recente para a leitura mais antiga. Os dados que são devolvidos são o nome da caixa, laboratório da caixa, nome de cada componente e a data e hora do *webserver*, Raspberry Pi, no momento da leitura.

Na função componentes em falta é feita uma comparação entre os componentes da caixa que foi lida e os componentes que foram recebidos do post. Isto irá devolver o nome dos componentes daquela caixa que estão em falta. Vai buscar o nome dos componentes da caixa que foi lida na última leitura, que é o max(id) da tabela leituras, e verifica se a *tag* é diferente dos componentes lidos na última leitura. Esta função compara até dez componentes.

5.2.2.3. Conexões entre nodes para o estado da caixa

Na Figura 57, é recebido o post do estado da caixa (“state”), se está completa ou incompleta. Este flow é apresentado na *dashboard* Leituras, juntamente com o nome e laboratório da caixa e dos seus componentes lidos.

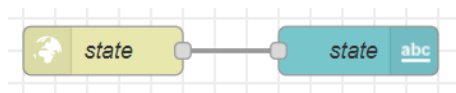


Figura 57 - Flow que indica o estado da caixa.

5.2.2.4. Conexões para adicionar caixas componentes na base de dados

Para adicionar uma nova caixa é necessário ir ao menu principal e pressionar o botão adicionar caixa. De seguida é necessário ir ao menu da *dashboard* e mudar para a opção adicionar, onde está o formulário para preencher com os requisitos da caixa, enviando esses requisitos para a base de dados, sendo eles o nome, laboratório e a *tag* que já está registada, este processo pode ser observado na Figura 58. Este processo é igual para as caixas, componentes e componentes de Mesa.

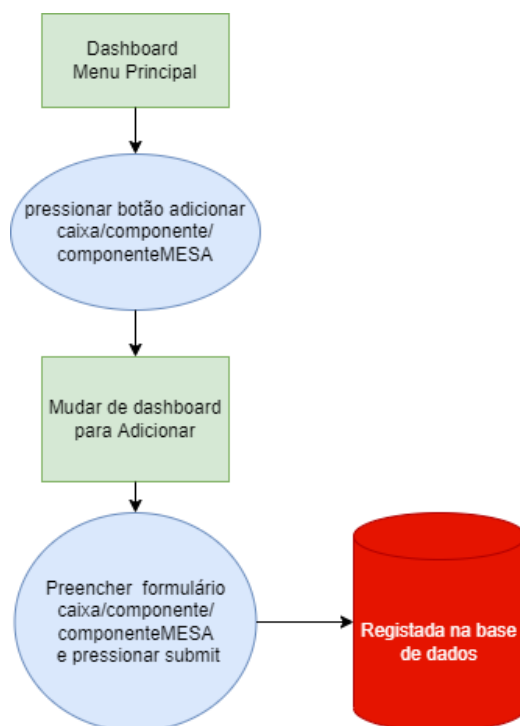


Figura 58 - Diagrama de adicionar caixa, componente e componente Mesa.

Na realização deste processo (Figura 59) é recebido o post de uma *tag*("request_tag2/addcaixa"), onde na função é imputado o valor recebido no campo do formulário "Tag", depois é preenchido o formulário e no submit é feita a *querie* para inserir a caixa na base de dados.

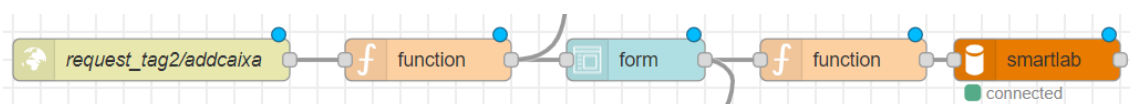


Figura 59 - Flow para adicionar caixa.

Os dados do formulário, como se pode observar na Figura 60, são a *tag*, nome e laboratório. É feito o mesmo processo para adicionar componentes, mas ao preencher o formulário, é necessário preencher o nome e a *tag* da caixa, ou seja, colocar a *tag* da caixa à qual queremos adicionar este componente. A *querie* para adicionar a caixa para a base de dados está representada na Figura 61.

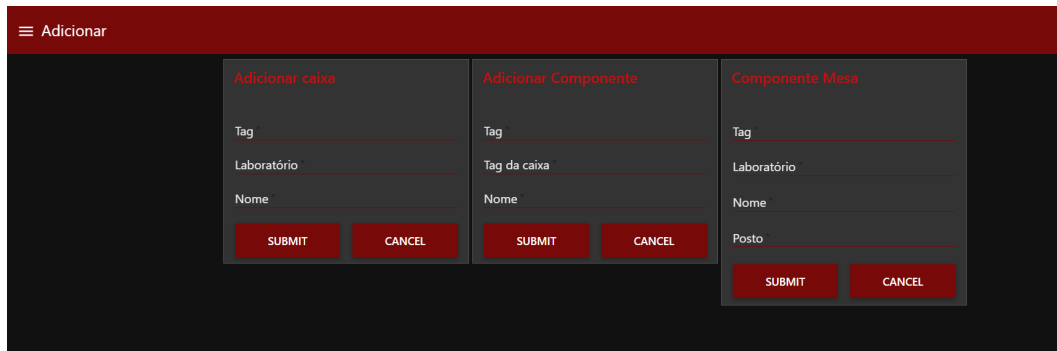


Figura 60 – Aba da *dashboard* adicionar.

```

msg.topic = "INSERT INTO caixas (tag,laboratorio,nome) VALUES ('"+msg.payload.tag+"','"+msg.payload.laboratorio+"','"+msg.payload.nome+"')";
return msg;

```

Figura 61 - Inserir caixa na base de dados.

5.2.2.5. Conexões entre nodes dos botões

No menu principal temos três botões (Figura 62), sendo que têm os três a mesma função, sendo eles designados para adicionar caixa, adicionar componente e adicionar componente de Mesa. O outro botão `button_1` serve para fazer a verificação da caixa.

O ESP32 manda constantemente um pedido de reposta(request) para o servidor Node-RED, no qual uma resposta do servidor é enviada de volta, por defeito se nenhum botão for premido a mensagem de resposta é ignorada pelo ESP32, caso um botão seja pressionado a mensagem de resposta para o microcontrolador é validada e uma funcionalidade é executada, quer seja adicionar caixa, adicionar componente, adicionar componente de Mesa ou verificar uma caixa.

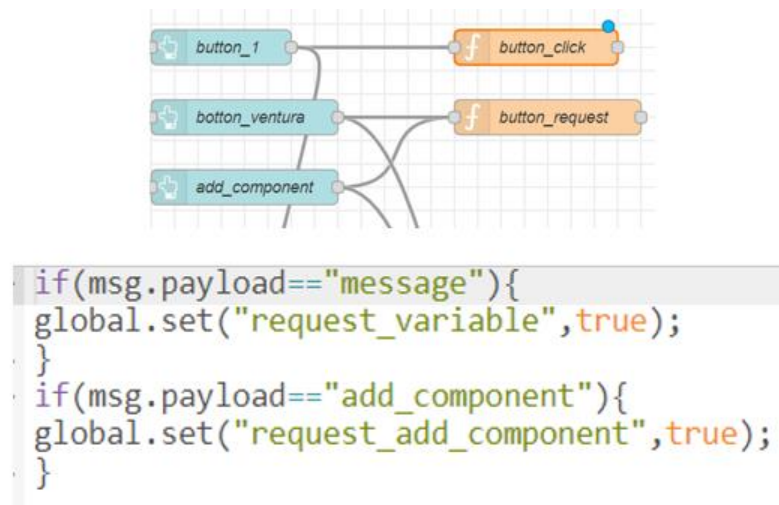


Figura 62 - Botões no Node-RED.

5.2.2.6. Sistema Check IN/OUT Node-RED

Neste flow (Figura 63) qualquer *tag* lida pelo leitor é inserida na tabela leituras2, o último registo da tabela leituras2 é apresentado na aba da *dashboard* Check IN/OUT. Esta tabela vai apresentar o nome, laboratório de qualquer caixa, componente ou componente de mesa, e a data e hora do servidor no momento da leitura. A tabela leituras2 regista até dez componentes, caixas ou componentes de mesa, para caso um aluno sair da sala com uma caixa completa mais algum componente, a tabela leituras2 conseguir guardar todas as *tags* no registo.

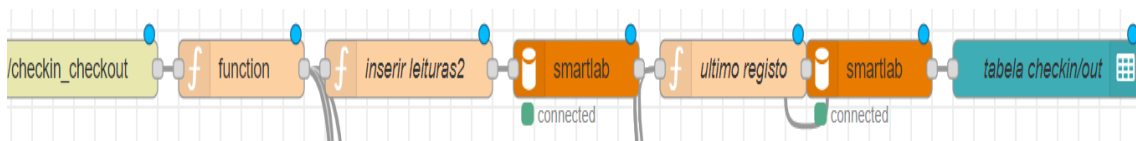


Figura 63 - Flow Check IN/OUT.

No Check IN/OUT recebemos um post das *tags* e entra na função, faz o *split* igual ao Check Box para a divisão das *tags* lidas. Na função seguinte insere a *tag* e data(datetime) na tabela leituras2 de cada *tag* lida (componente, componentes de mesa ou caixa). Devido ao facto de o leitor estar sempre ligado, envia *Strings* vazias até ler *tags*. Para resolver este problema foi feita uma condição para só inserir *String* que sejam *tags* na tabela leituras2, para a tabela não ter *String* de *tags* vazias, como se verifica na Figura 64.

Após inserir na tabela leituras2 é feita a *querie* na função (“último registo”) para obter, o nome, laboratório e data de todos os registos da tabela leituras2, ordenados por ordem decrescente, da leitura mais recente para a mais antiga.

No nome é feita uma sub*querie* às três tabelas para devolver o nome da *tag* lida caso seja caixa, componente ou componente de mesa.

```
var componente1 = msg.value1;  
  
if (componente1 != null) {  
    msg.topic = "Insert into leituras2 (tag,data) VALUES ('"+componente1+"',NOW())";  
    return msg;  
}
```

Figura 64 - Inserir componente para tabela leituras2.

6. Testes de funcionamento

Foram realizados vários cenários durante o projeto para obter o melhor resultado possível na verificação de uma caixa.

Através da informação no cartão microSD referente às *tags* registadas no ficheiro **Boxlist.CSV** que tem o ID das *tags* de componentes associadas ao ID das *tags* de caixa.

É necessário que a caixa a ser verificada tenha uma posição específica para melhor radiação da antena do leitor, na qual a posição mais conveniente é a colocação da caixa frontalmente com o leitor. No entanto, por vezes o Leitor apresenta defeito na leitura e não reconhece na totalidade as *tags* que supostamente estão presentes, pelo que essa é a única vulnerabilidade do Sistema.

O seu funcionamento e implementação em software, no entanto, como demonstra o exemplo possui a mesma eficácia independentemente do número de *tags* registadas nos seus ficheiros, quer o número de leituras e quantidade de deteções feitas em simultâneo, deste modo em termos de processamento e tratamentos de dados o programa é totalmente consistente e na associação de informação não existem quaisquer falhas no sistema local.

Os dados referentes à leitura das *tags* que foram verificadas serão enviadas posteriormente para a plataforma Web Node-Red. (Figura 65).

64674	64809		
92900	64613	64666	64769
0e52a	64798		
64676			

Figura 65 - Ficheiro Excel das *tags* dos componentes e das caixas

Deste modo os testes foram:

- Colocação de uma *tag* de caixa e outra *tag* de componente no leitor, e testar a funcionalidade de verificação de uma caixa, nomeadamente para 64674 que se encontra como *tag* de caixa no

ficheiro que tem a *tag* 64809 associada como componente. Através da leitura da caixa é possível concluir que a leitura referente à caixa está completa.

- De seguida foi efetuado um procedimento similar ao anterior, no entanto foram colocadas *tags* sobrepostas no leitor de modo a testar a fiabilidade e correta associação no sistema, foram colocadas algumas pertencentes aos ficheiros de componentes, nomeadamente: 64769, 64613, 64798.

Outras três não pertencentes de todo ao sistema nomeadamente: 66ac3,0c4ff,64674.

E outras duas *tags* designadas caixas nos ficheiros:0e52a,74674.

Por fim é visualizada a informação referente ao teste feito na Figura 66

```
Informacao de todas as tags lidas encontra-se abaixo:
TAG: 66ac3STATUS: Nao registada no sistema
-----
TAG:0e52a STATUS:
Registada no cartao como sendo caixa
-----
TAG: 64769 STATUS: Registada no cartao TAG DE CAIXA ASSOCIADA: 92900
-----
TAG: 0c4ffSTATUS: Nao registada no sistema
-----
TAG:64674 STATUS:
Registada no cartao como sendo caixa
-----
TAG: 64676STATUS: Nao registada no sistema
-----
TAG: 64613 STATUS: Registada no cartao TAG DE CAIXA ASSOCIADA: 92900
-----
TAG: 64798 STATUS: Registada no cartao TAG DE CAIXA ASSOCIADA: 0e52a
```

Figura 66 - Informação da porta Serial de uma leitura de testes

Na Figura 67, numa fase mais avançada mostra o posicionamento do material numa verificação da caixa de modo a facilitar a leitura correta. É necessário ter em atenção a breadboard pois contem metal por baixo que reflete as ondas radio provenientes do leitor RFID e dificulta muito a leitura, sendo que os testes de maior sucesso foram quando se trocou a breadboard por outra sem metal. Por outro lado, os alicatos que se encontram dentro bolsa não têm quaisquer problemas de leituras.



Figura 67 - Teste da verificação da caixa.

A Figura 68 demonstra a realização um teste com sete componentes onde foi retirado um componente e verifica-se que a caixa fica incompleta, e acende o led vermelho. No dispositivo da Check Box no menu principal é possível verificar toda a informação da última leitura, é possível verificar no submenu quais os componentes que foram lidos pelo leitor e também as *tags* que estão em falta

Estes dados são enviados para o Node-RED onde se pode observar toda a informação sobre a última leitura, quais a *tags* em falta e guardada a informação da leitura numa tabela da base de dados.

Foi realizado um teste onde foram colocadas todas as *tags* referentes a uma caixa completa, ou seja, composta pelos oito componentes. Como se pode observar na Figura 69, o LCD apresenta caixa completa e acende um led verde.



Figura 68 - Teste de verificar caixa incompleta

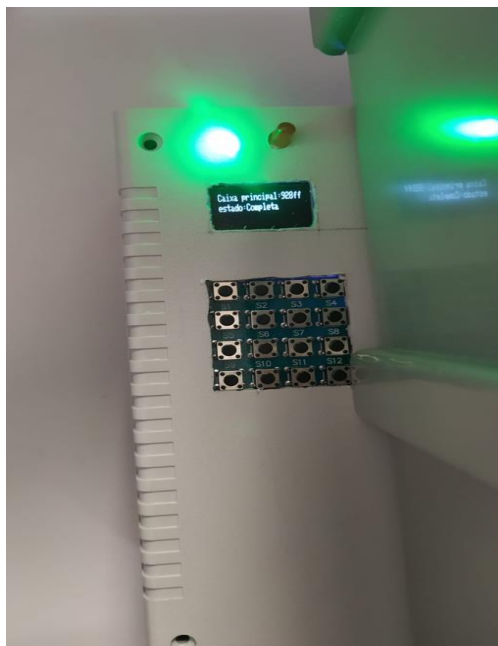


Figura 69 - Informação do LCD referente à Leitura testada

A partir deste exemplo é possível demonstrar o modo de funcionamento do sistema Check Box localmente, referente a uma verificação de caixa, assim como o seu bom funcionamento em termos de Software de implementação.

7. Conclusão e trabalhos futuros

Este projeto foi implementado e desenvolvido por volta de dois sistemas RFID que fazem a leitura local de *tags*, ambos com propósitos diferentes, no entanto estão interligados com uma base de dados externa.

Foram pesquisadas diferentes tecnologias, diferentes protocolos de comunicação e metodologias adotadas perante as várias fases do projeto, houve também algumas falhas em termos experimentais e erros aleatórios ao longo do seu desenvolvimento.

No entanto, o desenvolvimento do projeto foi uma mais valia em termos de conhecimento, experiência e capacidade de dimensionamento. Os resultados finais em termos de funcionalidade do projeto não foram cumpridos na totalidade.

É ainda possível desenvolver muito o projeto, deste modo são recomendados para o sistema global:

- Multiplicação dos sistemas Check IN/OUT e Check Box para todos os laboratórios;
- Interligação e comunicação dos sistemas de diferentes laboratórios.

São recomendados para o sistema Check Box:

- Troca do leitor RFID que aparenta dificuldades na deteção de *tags* de componentes dentro da caixa, sendo o único defeito deste sistema;
- Ter a opção no menu para remover *tags* do cartão microSD;
- Ter a opção para remover *tags* da base de dados através do Node-RED.

São recomendados para o sistema Check IN/OUT:

- Uso de um leitor RFID de alcance maior, que possibilite a incorporação de duas antenas para deteção de entrada ou saída de um laboratório, sendo que para este efeito foram usados dois leitores de curto alcance similares aos do sistema Check Box para apenas simulação.

8. Referências

- [1] “Sentrax,” [Online]. Available: <https://sentrax.com/>
- [2] “Eliko,” [Online]. Available: https://eliko.tech/uwb-rtls-ultra-wideband-real-time-location-system/?gclid=CjwKCAjw5s6WBhA4EiwACGncZYRlYHIUK1vUtGzDuh9nGGkXe9Nx6p2XbRp0esXnpQ2W5qKY74VWSRoCzk0QAvD_BwE#benefits-of-eliko-uwb-rtls
- [3] P. Almeida e J. Arrais, “Solução para rastreio de produtos e validação automática de expedição com base em RFID,” 2015
- [4] “Baú da Eletronica,” [Online]. Available: <https://blog.baudaeletronica.com.br/rfid-com-arduino/?fbclid=IwAR05rgssh5i2z05pGONaBA8Gz4yfHLleAEmYeDiE2qiMOGUBqcJZEnnBLJo>
- [5] “PTROBOTICS,” [Online]. Available: <https://www.ptrobotics.com/rfid/6840-uhf-rfid-tag-set-of-5.html>
- [6] “Indiamart,” [Online]. Available: <https://www.indiamart.com/proddetail/active-uhf-rfid-tags-22301326891.html>
- [7] “RFID4ustore,” [Online]. Available: <https://rfid4ustore.com/caen-uhf-temperature-logger-semi-passive-tag-epc-gen2-with-external-probe/>
- [8] “i3c,” [Online]. Available: <https://i3csolucoes.com.br/como-funciona-o-rfid/>
- [9] “LINK LABS,” [Online]. Available: <https://www.link-labs.com/blog/ultra-wideband-positioning-location-tracking>
- [10] “infsoft,” [Online]. Available: <https://www.infsoft.com/basics/positioning-technologies/ultra-wideband/>
- [11] “Lifewire,” [Online]. Available: <https://www.lifewire.com/trilateration-in-gps-1683341>

- [12] “GEOTAB,” [Online]. Available: <https://www.geotab.com/blog/what-is-gps/>
- [13] “inpixon,” [Online]. Available: <https://www.inpixon.com/technology/standards/bluetooth-low-energy>
- [14] “Ferramentas,” [Online]. Available: <https://ferramentas.pt/modulo-bluetooth-sem-software-gcy-30-4-bosch-1600a00r26.html>
- [15] “Wikipédia,” [Online]. Available: <https://pt.wikipedia.org/wiki/Wi-Fi>
- [16] “Instituto Federal Santa Catarina,” [Online]. Available: <https://moodle.ifsc.edu.br/mod/book/view.php?id=267690&chapterid=60297>
- [17] “redes,” [Online]. Available: <http://redesssss.blogspot.com/2009/11/access-point.html>
- [18] “Circuitbasics,” [Online]. Available: https://www.circuitbasics.com/basics-uart-communication/?fbclid=IwAR3wE8loIUuEHogFYaPHvoBraN9G_3V2bI06GO5j9Vxg5naXsTQez8attrg
- [19] “Analogdevices,” [Online]. Available: https://www.analog.com/media/en/analog-dialogue/volume-54/number-4/uart-a-hardware-communication-protocol.pdf?fbclid=IwAR3Y1tihOxJajrDux15o3C4JBd_jRST9jp2SxBZupzW4hjUS8hNgyWE3yqI
- [20] “Circuit Basics,” [Online]. Available: <https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/>
- [21] “Rockcontent,” [Online]. Available: <https://rockcontent.com/br/blog/http/>
- [22] “espressif,” [Online]. Available: https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- [23] “M5stack,” [Online]. Available: https://docs.m5stack.com/en/unit/uhf_rfid?fbclid=IwAR0T_RaE1sVgnwi4lZRDGgx6u2qSek3E3zajc9SILUDv4Bewes2J_Hms0Iw
- [24] “Tinyosshop,” [Online]. Available: <https://www.tinyosshop.com/datasheet/DS3231%20datasheet.pdf>
- [25] “CircuitDigest,” [Online]. Available: https://circuitdigest.com/microcontroller-projects/keypad-interfacing-with-8051-microcontroller?fbclid=IwAR3QccbhC2b9uTAPENQk51L9iywXro_Q9tqILjsKbHWN0B7m40A1lv357Cc

- [26] “cdn,” [Online]. Available: <https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>
- [27] “Supertalent,” [Online]. Available: http://www.supertalent.com/datasheets/5_112.pdf
- [28] “ptrobotics,” [Online]. Available: <https://www.ptrobotics.com/chassi-de-robo/5418-alphabot2-robot-building-kit-for-raspberry-pi-3-model-b.html>
- [29] “Marktrace,” [Online]. Available: https://www.marktraceiot.com/rfid-uhf-short-range-integrated-reader_p41.html
- [30] “Espressif,” [Online]. Available: https://docs.espressif.com/projects/esp-at/en/latest/ESP32/Get_Started/Hardware_connection.html
- [31] “Nodered,” [Online]. Available: https://nodered.org/?fbclid=IwAR075fotmVNqlUXKupKh4t_-ZiGYivaUiHJgMzVQSAWza2fReW6b-d4DV-c

9. ANEXO 1

```
1 #include "Arduino.h"
2 #include "RFID_command.h"
3 #include <stdio.h>
4 #include <Wire.h>
5 #include <SPI.h>
6 #include <SD.h>
7 #include <FS.h>
8 #include <string.h>
9 #include <stdbool.h>
10 #include "RTCLib.h"
11 #include <WiFi.h>
12 #include <HttpClient.h>
13 #include <Adafruit_GFX.h>
14 #include <Adafruit_SSD1306.h>
15 #include <Keypad.h>
```

Figura 1 – Bibliotecas.

```
void setup() ;
void IRAM_ATTR onTimer2() ;
void loop() ;
void Menu_main() ;
int User_menu_input() ;
void func_verificar_tags() ;
void scan_tags_row(String search_string, String current_reading_array[]) ;
void existing_tags(String search_string) ;
void raw_registo(String present_tags[], String main, String state) ;
void registo(String main, String present_tags[], int count, String state) ;
String func_adicionar_novas_tags() ;
void ler_file_1() ;
void update_file_1() ;
void InserirBox_file_1() ;
void ler_file_2() ;
void update_ler_file2() ;
int ler_end_of_file_2() ;
void inserirBox_file_2(String current_reading) ;
String Inserir_tag_main() ;
void Inserir_tag_nova() ;
void put_on_line(String tag_nova, String tag_main) ;
void imprimir_file1() ;
void imprimir_file2() ;
void write_sd() ;
int http_get_message() ;
void http_get_message_2() ;
void menu_select_keyboard() ;
int menu_select_all() ;
void mostrar_leitura() ;
void send_tags() ;
void mostrar_informacao_tag() ;
```

Figura 2 – Funções do programa Check Box.

```
selection = menu_select_all();  
switch (selection) {  
  
case 1:  
    func_verificar_tags();  
  
    break;  
case 2:  
    InserirBox_file_1();  
  
    break;  
case 3:  
    Inserir_tag_nova();  
  
    break;  
case 4:  
    mostrar_leitura();  
  
    break;  
case 5:  
    mostrar_informacao_tag();  
    break;  
case 6:  
    write_sd();  
    break;  
default:  
  
    break;
```

Figura 3 – Menu Check Box.

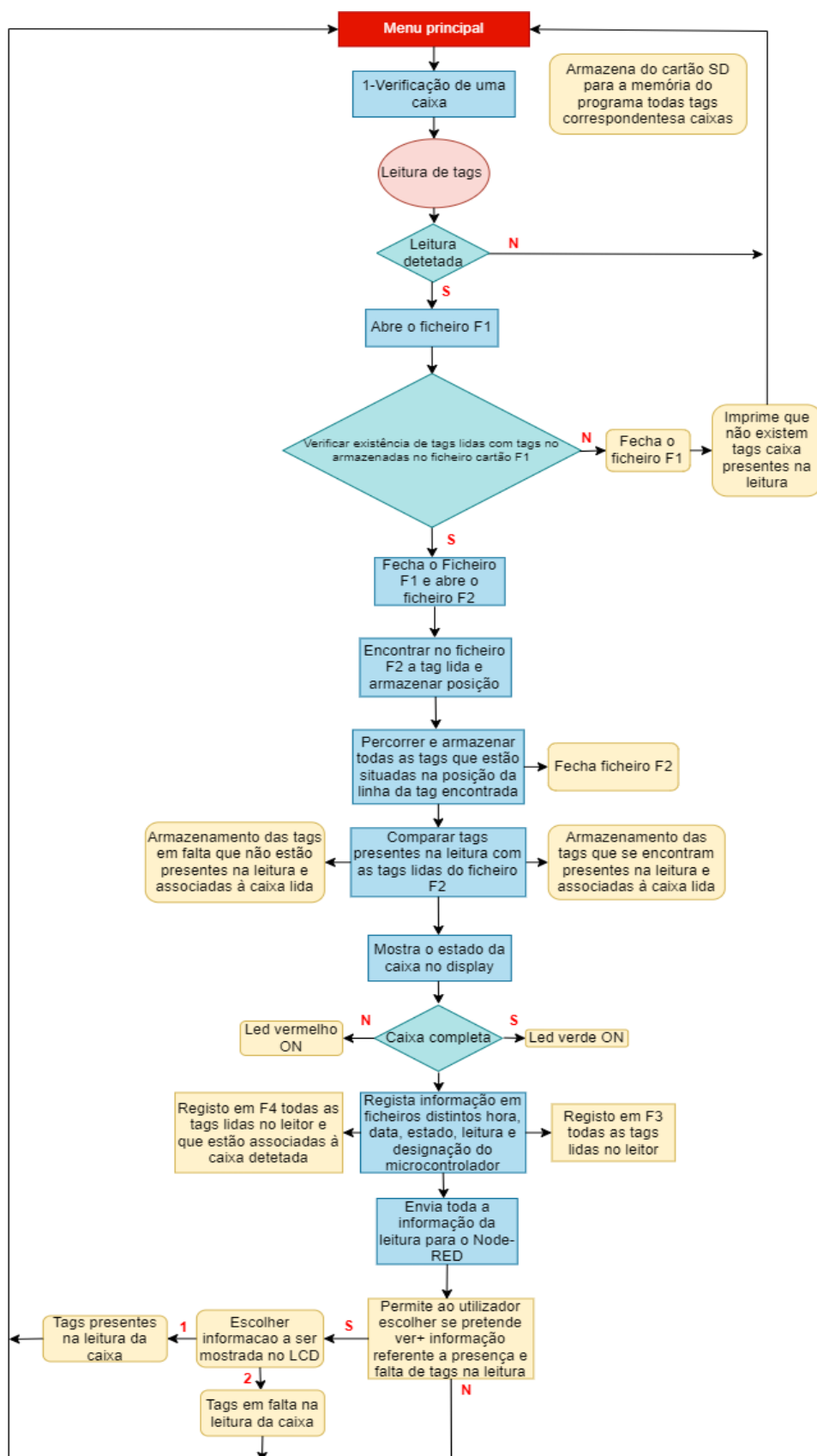


Figura 4 – Fluxograma do código verificar caixa.