



UNIVERSIDADE
LUSÓFONA

Sensores de Temperatura para HACCP

Trabalho Final de curso

Relatório Final

Roberto Rafael Luzindro: 20065449

Nome do Orientador: Sérgio Ferreira

Trabalho Final de Curso | LEI | 30 de junho de 2023

Aluno332 - Sensores de Temperatura para HACCP

Direitos de cópia

(Sensores de Temperatura para HACCP), Copyright de *(Roberto Rafael Luzindro)*, ULHT.

A Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Agradecimentos

Com a finalização deste Relatório de Estágio não posso deixar de agradecer a algumas pessoas que, direta ou indiretamente, me ajudaram neste percurso tão importante da minha vida pessoal e profissional. E a decisão de voltar a estudar, passados 16 anos, para concluir a licenciatura foi a mais acertada.

A todos(as) os(as) professores(as) que no decorrer do meu percurso académico, transmitiram-me experiências e saberes imprescindíveis, para a minha formação como pessoa.

No entanto devo salientar, um especial agradecimento, ao professor e orientador de projeto Sérgio Ferreira, pelo apoio, orientação, sugestões e disponibilidade ao longo do respetivo trabalho.

Ao meu sócio, pela confiança para a realização e implementação do projeto na nossa empresa.

Por fim e de extrema importância, à minha esposa e aos meus 3 filhos, expresso a minha gratidão pelas palavras de incentivo, dedicação, compreensão e amor incondicional, que me proporcionaram na minha formação académica. Sem eles, este projeto não teria sido possível.

Obrigado

Resumo

O trabalho final de curso proposto é o desenvolvimento e configuração de vários microcontroladores Arduino, mais propriamente o ESP32, para monitorização e registo de temperatura para HACCP¹ no restaurante da empresa Northspot.

Sendo esse registo feito para todos os sistemas de refrigeração existente na empresa, para cumprir com os requisitos definidos na parte do HACCP.

O objetivo deste trabalho final de curso é solucionar o proposto de forma barata e simples, para que no futuro chegue a mais empresas do ramo.

Palavras-chave: Arduino, IoT, sensores temperatura, monitorização, ESP32.

¹ O HACCP é uma sigla internacionalmente reconhecida para Hazard Analysis and Critical Control Point ou Análise de Perigos e Controlo de Pontos Críticos.

Abstract

The proposed final course work is the development and configuration of several Arduino microcontrollers, specifically the ESP32, for monitoring and recording temperature for HACCP at the Northspot company restaurant.

This register is made for all existing refrigeration systems in the company, to comply with the requirements defined in the part of HACCP.

The goal of this final course work is to solve the proposed in a cheap and simple way, so that in the future it can reach more companies in the industry.

Keywords: Arduino, IoT, temperature sensors, monitoring, ESP32.

Índice

Agradecimentos.....	iv
Resumo	v
Abstract	vi
Índice	vii
Lista de Figuras	ix
Lista de Tabelas	x
1 Identificação do Problema	1
2 Viabilidade e Pertinência.....	3
3 Benchmarking.....	4
4 Solução Proposta.....	7
4.1 Arduino	8
4.2 ESP32.....	8
4.2.1 FireBeetle 2 ESP32-E.....	9
4.2.2 Pinagem	11
4.2.3 Configuração da IDE do Arduino	12
4.2.1 Atualização de firmware pela biblioteca esp32FOTA.....	15
4.3 Sensores de Temperatura DS18B20	17
4.4 Bateria.....	19
4.5 Sensores Protótipos	20
4.5.1 Esquema da montagem dos componentes	20
4.5.2 Versão Inicial.....	21
4.5.3 Versão 2	21
4.5.4 Versão para equipamentos de congelção	21
4.6 Preço custo dos materiais.....	22
4.7 MQTT	23
4.7.1 Broker MQTT com Rasperry Pi.....	25
4.7.2 Instalação do Broker MQTT Mosquitto	26
4.7.3 Tópicos MQTT	29

4.8	Base de Dados.....	32
4.9	NODE.JS.....	33
4.10	Node-RED.....	34
4.10.1	Esquemas de Configuração no NodeRed	37
4.10.2	MQTT no Node-Red.....	37
4.10.3	Página de monitorização	38
4.11	Twilio.....	43
4.11.1	Whatsapp.....	43
5	Calendário	46
	Bibliografia.....	47
	Apêndice 1 – Requisitos	48
	Apêndice 2 – Vídeo do TFC.....	50
	Apêndice 3 – Código do programa	51
	Glossário	52
6	Testes	53
6.1	MQTT	53
6.2	Base de Dados.....	53
6.3	Envio de Email.....	54

Lista de Figuras

Figura 1 - Formato atual de registo de temperaturas	2
Figura 2 - Solução Proposta	7
Figura 3 - Arduino Uno	8
Figura 4 - ESP WROOM 32	8
Figura 5 - Firebeetle ESP32-E	9
Figura 6 - Esquema de Pinos do Firebeetle	11
Figura 7 - Software Arduino IDE	12
Figura 8 - URL Adicional do Gestor de Placas	13
Figura 9 - Escolha do Tipo de Placa	13
Figura 10 - Seleção da Porta Serial	14
Figura 11 - Complicação do Sketch para o ESP32	14
Figura 12 - Carregamento Completo do Sketch	15
Figura 13 - Exportação ficheiro binário	16
Figura 14 - Compilação do ficheiro binário	16
Figura 15 - Pasta do Servidor Web com o ficheiro binário e JSON	16
Figura 16 - Sensor DS18B20	17
Figura 17 - Esquema do Sensor DS18B20	18
Figura 18 - Sensor DS18B20 com fio à prova de água	18
Figura 19 - Bateria 3,7v de 2000mAh	19
Figura 20 - Esquema dos Componentes	20
Figura 21 - Versão 1 do Sensor	21
Figura 22 - Versão 2 do Sensor	21
Figura 23 - Sensor para Equipamentos de Congelação	22
Figura 24 - MQTT Broker Exemplo	23
Figura 25 - Raspberry Pi 3(Fonte https://www.raspberrypiportugal.pt/)	25
Figura 26 - Configuração do Mosquitto	27
Figura 27 - Servidor Mosquitto Ativo	28
Figura 28 - Lista de Tópicos MQTT	30
Figura 29 - Modelo Entidade Relação	32
Figura 30 - Modelo Funcional de um Servidor Tradicional vs Servidor Node.js	34
Figura 31 - Edição de fluxo baseada no navegador	35
Figura 32 - Desenho dos Módulos no Node-Red	37
Figura 33 - Configuração do MQTT no Node-Red	37
Figura 34 - Estado da ligação ao Broker	38
Figura 35 - Menu barra lateral	39
Figura 36 - Página Inicial	39
Figura 37 – Visualização dos Estados dos Sensores na Cozinha	40
Figura 38 - Visualização dos Estados dos Sensores na Sala	40
Figura 39 - Visualização dos Estados dos Sensores no Armazém	40
Figura 40 - Visualização dos Estados dos Sensores na Esplanada	41
Figura 41 -Alteração do Estado do Sensor	42
Figura 42 – Listagem de todas as zonas dos últimos 24 registos	42
Figura 43 - Callmebot Whatsapp	44
Figura 44 - Exemplos de Mensagem Whatsapp	45
Figura 45 - Query Select à Tabela Registos	53

Lista de Tabelas

Tabela 1 - Lista de Equipamentos de Frio..... 1

Tabela 2 - Alternativas existentes 4

Tabela 3 - Previsão de vida estimada da Bateria 20

Tabela 4 – Custo Total por sensor 22

Tabela 5 - Tipos de Mensagens MQTT 24

Tabela 6 - Identificador e Atributos da Base de Dados 32

Tabela 7 - Tabela de Requisitos 48

1 Identificação do Problema

O trabalho final de curso a que me proponho é o desenvolvimento e criação de hardware com microprocessadores e uma página web para monitorização e registo de temperatura para HACCP no meu restaurante.

Este processo é feito atualmente com o registo em papel de cada sistema refrigerado existente na empresa e por secção. No caso em questão sala, esplanada, armazém e cozinha, correspondendo entre 10 e 15 sensores.

Este deveria ser feito duas vezes ao dia e muitas das vezes esse registo não é feito pelo colaborador responsável pelos registos.

Esta solução terá de ser o mais barata possível, mas ou mesmo tempo fiável.

Tabela 1 - Lista de Equipamentos de Frio

Zona	Equipamentos	Quantidade	Temperatura Ideal
Cozinha	Vertical de Refrigeração	3	2x – 0º C a 8ºC
			1x – -2º C a 4ºC
	Vertical de Congelação	1	≤18º C
Sala	Balcão Vertical	2	2º C a 8º C
	Balcão Horizontal	1	2º C a 8º C
Armazém	Câmara Frigorífica	1	≤18º C
	Vertical de Refrigeração	3	2º C a 8º C
Esplanada	Balcão Horizontal	2	2º C a 8º C
	Vertical de Refrigeração	2	2º C a 8º C
	Arca Congelação	1	≤18º C

CERVEJARIA ARIMAR				FICHA DE CONTROLO				FC - 01												
MONITORIZAÇÃO DAS TEMPERATURAS NOS EQUIPAMENTOS DE FRIO								Página: 1 de 2 Data Emissão: 02/09/2022 Revisão: 02												
Mês: <u>Outubro</u> de <u>2022</u>				Armazém				Responsável:												
1C	Câmara Ultracong. (≤ -18°C)		2C		Eq. refrig. Peixe (0 a 3°C)		3C		Eq. refrig. Pão (0 a 5°C)		4C		Eq. refrig. diversos (0 a 5°C)		5C		Arca gelados (≤ -18°C)		Observ.	
DATA	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)	Hora	T (°C)		
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12	10.00	-18°	18.00	-18°	10.00	3°C	18.00	3°	10.00	5°C	18.00	5°C	10.00	5°C	18.00	5°C	10.00	-18°	18.00	-18°
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				
29																				
30																				
31																				

Elaborado por: (Garantia Ltda.)	Aprovado por:	Auditado Por:	Verificado Por:
(A Técnica Alimentar)	(A Gerência)	(A Técnica Alimentar)	(A Gerência)
Data: / /	Data: / /	Data: / /	Data: / /

Figura 1 - Formato atual de registo de temperaturas

2 Viabilidade e Pertinência

Este projeto terá um grande impacto na empresa no que diz respeito à modernização empresarial que temos vindo a desenvolver ao longo do tempo.

Com esta monitorização dos registos todo o processo, que até aqui tinha sido adotado em papel, será substituído por um mais assertivo e fiável. Eliminasse, o que por vezes acontece que é, a inserção de registo muito depois da hora a que deveria ser registada.

Para evitar esta falha proponho desenvolver, um mecanismo automático de registo de temperatura, para todos os equipamentos de frio existentes no restaurante.



A introdução do conceito da Internet das coisas, associando a monitorização e o registo da temperatura de cada sensor será o ponto a ter em consideração.



3 Benchmarking

Ao longo dos anos o facto dos registos não serem feitos com o rigor que deveriam, levou-nos a ir à procura de alternativas, mas algumas delas eram bastante dispendiosas.

Existem também alguns sensores, mais propriamente chamados de Data Logger que guardam dados para que possam ser transferidos, via USB, para o computador. Os valores dos dispositivos existentes são muito caros e alguns deles cobram mensalidade o que para a maioria das empresas não consegue suportar e a nossa empresa também não.

Tabela 2 - Alternativas existentes

Sensor	Caraterísticas
Termómetro Analógico	
	<p>Indicado monitoriza a temperatura em frigoríficos ou arcas congeladoras. Com indicação de zonas de congelação para fácil leitura. O termómetro para frigorífico ou congelador é circular de fácil leitura. Indica a temperatura na gama de medição de -30 até +30°C, em divisões de 1°C. Pequeno e prático. Sem certificado de calibração.</p> <p>Prós:</p> <ul style="list-style-type: none"> Muito barato, ronda os 5 Euros. <p>Contras:</p> <ul style="list-style-type: none"> A leitura é visual e implica o seu registo posterior.
Termómetro Digital com LCD	
	<p>O termómetro de frigorífico ou arca congeladora permite memorizar as temperaturas máxima e mínima registadas e indica a temperatura do frigorífico ou congelador entre -40,0°C até +69,9°C com uma resolução de 0,1°C e uma precisão de +/-1°C.</p> <p>Prós:</p> <ul style="list-style-type: none"> Barato, ronda os 20 Euros.

	<p>Contras:</p> <ul style="list-style-type: none"> • A leitura é visual e implica o seu registo posterior.
<p>DataLogger Wifi – Testo Saveris 2-T1</p>	
	<p>O sistema de registo de dados sem fios testo Saveris 2 é a solução moderna para a monitorização dos valores de humidade e temperatura em armazéns e salas de trabalho. O sistema consiste em registadores de dados sem fios e um acesso à nuvem Testo. Sem necessidade de instalar qualquer software, pode começar a colocar em funcionamento e configurar de forma rápida e fácil através da nuvem. Os registadores de dados sem fios registam, de forma fiável, a temperatura e a humidade em intervalos ajustáveis e transmitem as leituras diretamente para a nuvem através da WLAN.</p> <p>Prós:</p> <ul style="list-style-type: none"> • Leituras armazenadas em Cloud. Envio de Mensagens de Alerta <p>Contras:</p> <ul style="list-style-type: none"> • Preço ronda os 155 € por dispositivo.
<p>Sensores E-pack</p>	
	<p>Monitorização da temperatura em tempo real</p> <p>Discretas e sem fios, as sondas instaladas nas suas câmaras de refrigeração medem, instantânea e continuamente, a temperatura das mesmas.</p> <p>Não há necessidade de controlar a temperatura das suas câmaras de refrigeração de manhã e à noite; as sondas conectadas automatizam as leituras de temperatura 24 horas por dia, 7 dias por semana.</p> <p>Sempre que o limite de temperatura é excedido, o sistema gera um alerta, informando-o imediatamente via sms ou e-mail.</p> <p>Prós:</p> <ul style="list-style-type: none"> • Leituras armazenadas em Cloud. Envio de Mensagens de Alerta <p>Contras:</p> <ul style="list-style-type: none"> • Preço ronda os 99 € mensalidade, mas estão agregadas mais opções.

4 Solução Proposta

A solução discutida com o orientador do projeto foi que o registro fosse feito pela plataforma Arduino onde é possível enviar e receber informações de sistemas de captação de dados, como sensores.

A plataforma é composta essencialmente por duas partes. O Hardware e o Software. Sendo o hardware baseado em simples placas com microcontroladores e um ambiente de desenvolvimento para programar a respetiva placa. O Software é desenvolvimento web de manipulação simples e prática.

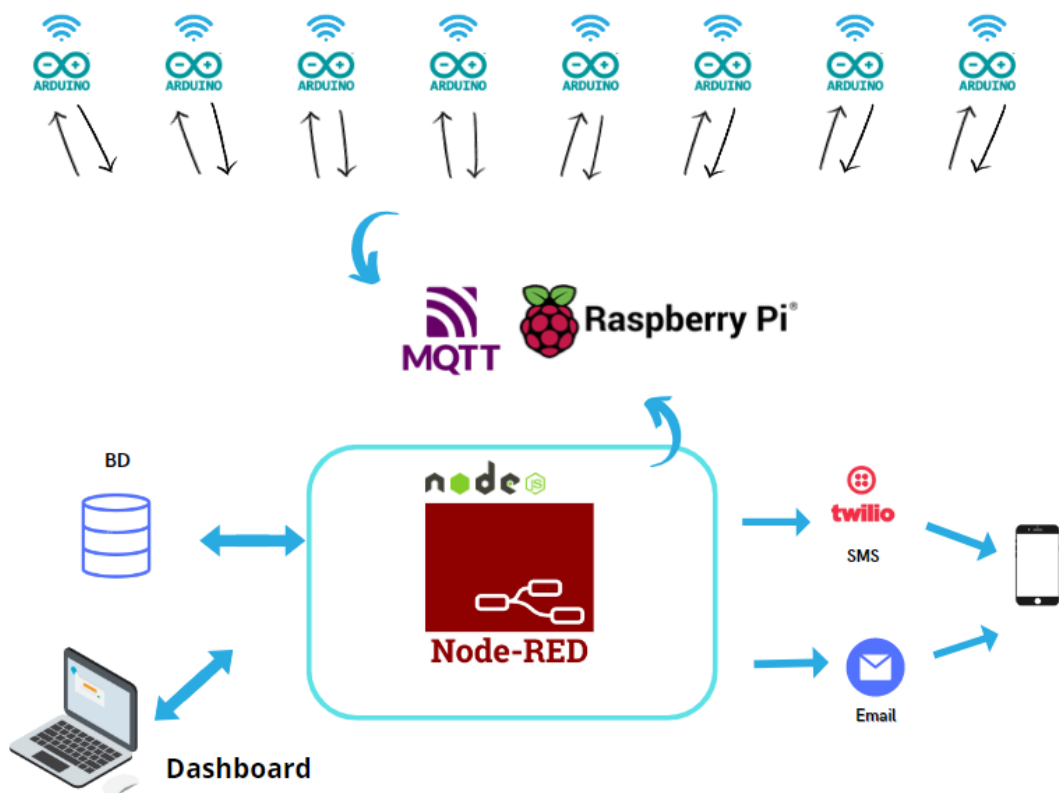


Figura 2 - Solução Proposta

4.1 Arduino



Figura 3 - Arduino Uno

Arduino é uma plataforma de prototipagem eletrônica, criado por Massimo Banzi e David Cuartielles, em 2005, com objetivo de permitir o desenvolvimento de controlo de sistemas interativos, de baixo custo e acessível a todos.

Com o Arduino é possível enviar e receber informações de praticamente qualquer outro sistema eletrónico. Desta forma, é possível construir por exemplo, um sistema de captação de dados de sensores, como temperatura, controlo de iluminação, processar e enviar esses dados para um sistema remoto, ou demonstrá-los num ecrã.

Outra característica importante é que todo material (software, bibliotecas, hardware) é open-source, ou seja, pode ser usado por todos, sem a necessidade de pagamento de royalties ou direitos de autor.

4.2 ESP32



Figura 4 - ESP WROOM 32

ESP32 é uma série de microcontroladores de baixo custo e baixo consumo de energia. Também é um sistema-em-um-chip com microcontrolador integrado, WiFi e Bluetooth. A série ESP32 emprega um microprocessador Tensilica Xtensa LX6 com duas variações dual-core e single-core e inclui uma antena integrada RF tipo balun, amplificador de potência, recetor de baixo ruído amplificado, filtros, gerenciamento de energia dos módulos.

4.2.1 FireBeetle 2 ESP32-E

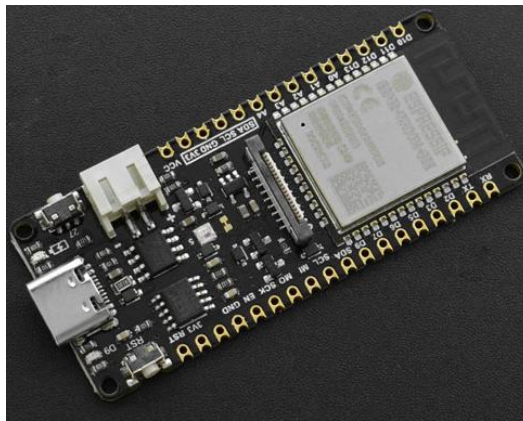


Figura 5 - Firebeetle ESP32-E

O FireBeetle ESP32-E, especialmente projetado para IoT, é uma placa controladora principal baseada em ESP-WROOM-32E com chips dual-core. Suporta comunicação de modo duplo WiFi e Bluetooth e apresenta tamanho pequeno, consumo de energia ultrabaixo, circuito de carregamento integrado e interface fácil de usar, que pode ser usado para IoT doméstico inteligente ou aplicações IoT industriais.

Especificações

- Tensão operacional: 3,3 V
- Tensão de entrada: 3,3V~5,5V
- Suporte de baixa potência: 10uA
- Corrente máxima de descarga: 600mA@3,3V LDO

- Corrente máxima de carga: 500mA
- Suporta Carregamento USB
- Processador: processador Tensilica LX6 dual-core (um para conexão de alta velocidade; outro para desenvolvimento independente)
- Frequência principal: 240MHz
- RAM: 520KB
- Flash: 32Mbits
- Padrão Wi-Fi: FCC/CE/TELEC/KCC
- Protocolo Wi-Fi: 802.11 b/g/n/d/e/i/k/r (802.11n, velocidade de até 150 Mbps), A-MPDU e A-MSDU Aggregation, suporta intervalo de guarda de 0,4 us)
- Faixa de frequência: 2,4 ~ 2,5 GHz
- Protocolo Bluetooth: compatível com o padrão Bluetooth v4.2 BR/EDR e BLE
- Áudio Bluetooth: áudio CVSD e SBC
- Corrente operacional: 80mA (média)
- Suporte ao download do Arduino com uma chave
- Suporta MicroPython
- Relógio no chip: cristal de 40MHz, cristal de 32,768KHz
- Digital I/O x10(Arduino default)
- Entrada analógica x5 (padrão do Arduino)
- SPI x1 (Padrão do Arduino)
- IIC x1 (Padrão do Arduino)
- I2S x1 (Arduino Padrão)
- RGB_LED: 5/D8
- Temperatura de operação: -40°C~+85°C
- Tamanho do módulo: 25,4 × 60 (mm)
- Tamanho do furo de montagem: Furo de montagem M2 com diâmetro de 2,0 mm

4.2.2 Pinagem

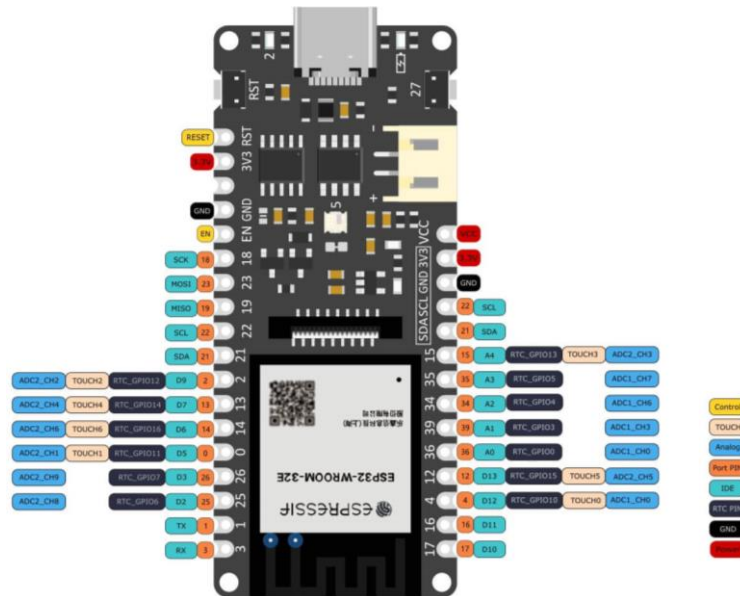


Figura 6 - Esquema de Pinos do Firebeetle

Características:

- Tipo-c: Interface USB: 4,75v-5,5v
- Conector da bateria de íons de lítio PH2.0: 3,5-4,2 V
- LED 2/D9: LED de controle via pino 2/D9
- Indicador de carregamento: LED vermelho para indicar o status do carregamento:
 - 1. Desligado quando totalmente carregado ou não carregado;
 - 2. Ligado durante o carregamento;
 - 3. Flash rápido quando alimentado por USB e sem bateria conectada.
- RST Reset Pin: clique no botão reset para redefinir o programa
- 5/D8 Indicador WS2812: controle LED WS2812 RGB via pino 5/D8
- Almofada de baixo consumo de energia: esta almofada é especialmente projetada para baixo consumo de energia. Ativo por padrão. pode cortar o fio fino no meio com uma faca para desconectá-lo. Após a desconexão, o

consumo de energia estática pode ser reduzido em 500 μ A. O consumo de energia pode ser reduzido para 13 μ A depois de controlar o controlador principal para entrar no modo de hibernação através do programa.

4.2.3 Configuração da IDE do Arduino

Arduino IDE é uma aplicação de plataforma cruzada, escrito em funções de C e C ++. É usado para escrever e fazer upload de programas em placas compatíveis com Arduino, mas também, com a ajuda de núcleos de terceiros, outras placas de desenvolvimento como é o caso da FireBeetle ESP32-E

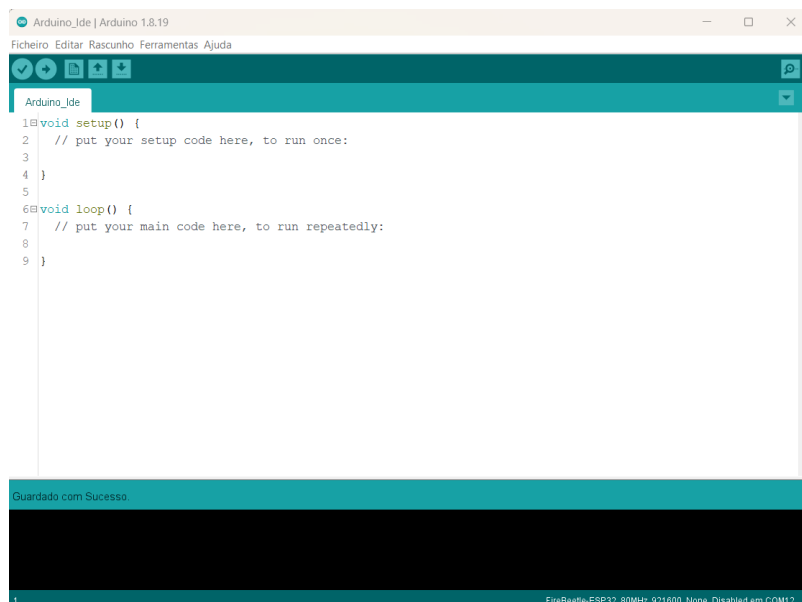


Figura 7 - Software Arduino IDE

Os programas feitos no Arduino são chamados de *sketches*. Antes de enviar os *sketches* para o ESP32 precisamos de informar ao IDE que tipo de placa estamos a usar.

Para tal é necessário instalar o núcleo do controlador através de um URL específico para esta placa.

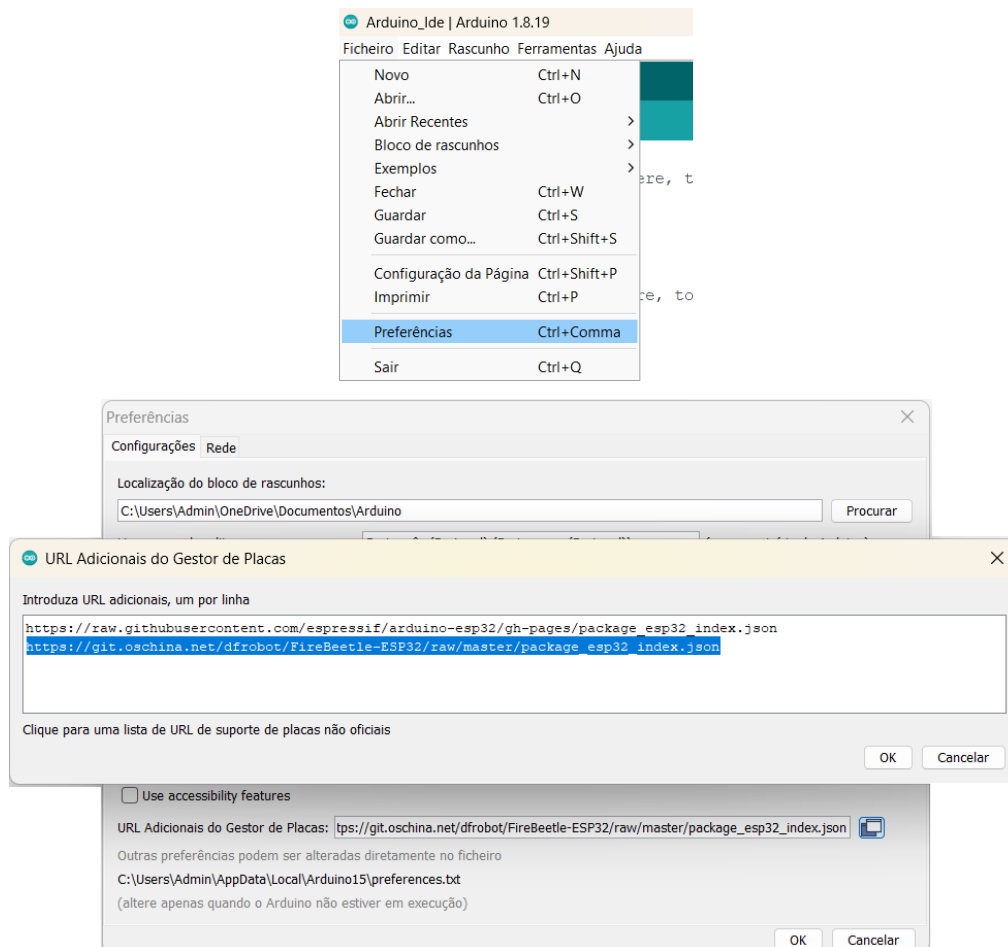


Figura 8 - URL Adicional do Gestor de Placas

https://git.oschina.net/dfrobot/FireBeetle-ESP32/raw/master/package_esp32_index.json

Com esta informação já podemos selecionar a placa a utilizar para upload dos *sketches* e escolher qual a porta serial para o envio.

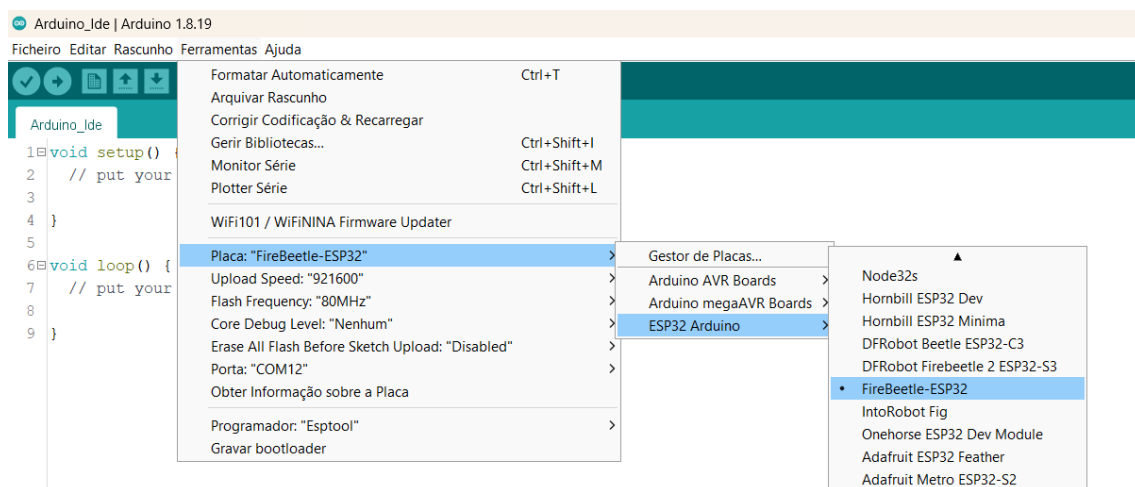


Figura 9 - Escolha do Tipo de Placa

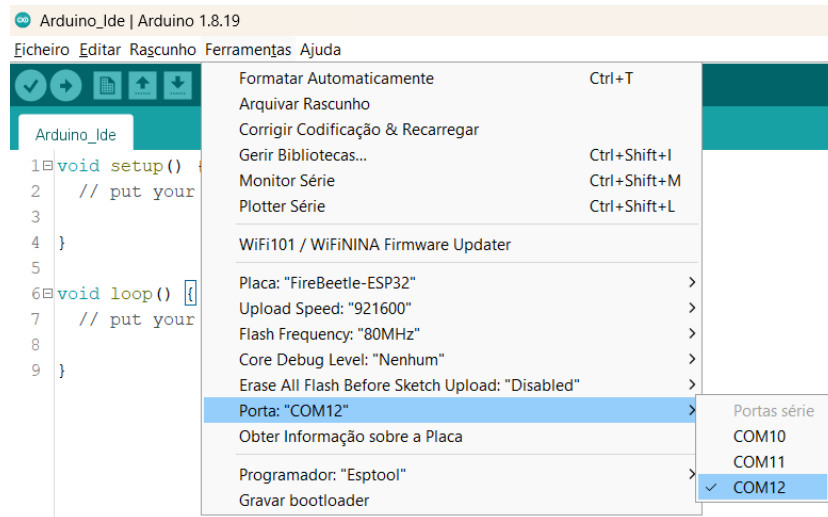
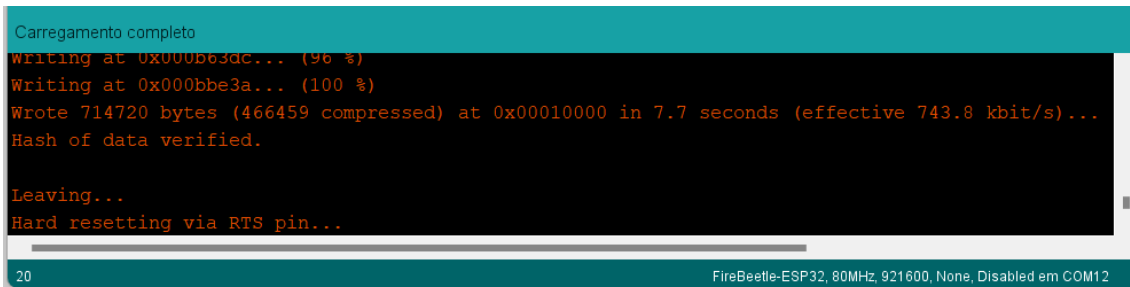


Figura 10 - Seleção da Porta Serial

Com isto, ao clicar no botão do upload, algumas coisas deverão acontecer. Primeiro, uma barra a indicar o progresso da operação enquanto o IDE compila o sketch (isto significa que está a converter o sketch para uma forma adequada para a transferência). Se existir erros de compilação serão mostrados. A seguir, no esp32, os Leds identificados por Rx e Fx irão piscar por breves segundos indicando a transferência do sketch. E por último, quando completo o carregamento, mostra se o upload foi bem sucedido, quantos bytes estão utilizados e disponíveis na memória flash para programas no ESP32.



Figura 11 - Complicação do Sketch para o ESP32



```
Carregamento completo
Writing at 0x000b63dc... (96 %)
Writing at 0x000bbe3a... (100 %)
Wrote 714720 bytes (466459 compressed) at 0x00010000 in 7.7 seconds (effective 743.8 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...

20 FireBeetle-ESP32, 80MHz, 921600, None, Disabled em COM12
```

Figura 12 - Carregamento Completo do Sketch

4.2.1 Atualização de firmware pela biblioteca esp32FOTA

Permite que o firmware seja atualizado a partir do servidor da empresa, o sensor verifica se há atualizações sempre que há envio da temperatura. Usa um arquivo JSON simples para descrever se um novo firmware está disponível.

A biblioteca acede a um ficheiro JSON hospedada no servidor e verifica se existe alguma versão nova para instalar. Caso existe nova versão faz o download e instala.

```
{
  "type": "esp32-fota-http",
  "version": "1",
  "url": "https://.../Cozinha/Sensor_4/Cozinha_4.ino.bin"
}
```

Neste caso, e utilizando o servidor web da empresa, foi criado um domínio ao qual foi chamado de ***sensores.northspot.pt***. Nele foram criadas 4 pastas, uma por cada zona. Dentro de cada pasta uma outra com o nome do sensor. E por último o ficheiro binário com o código para a atualização.

Esta atualização é feita com a gravação do sketch do ESP32 em ficheiro binário que será alojado no servido web.



Figura 13 - Exportação ficheiro binário

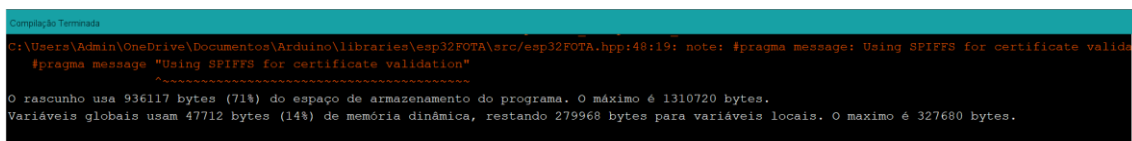


Figura 14 - Compilação do ficheiro binário

Em seguida, altera-se o nome do ficheiro para o nome que irá ser chamado no servidor para atualização. Faz-se o upload do ficheiro bin e do ficheiro JSON para o servidor. Assim, por cada vez que o ESP32 se liga ao WiFi e durante a sua execução, verifica se existe uma nova versão.

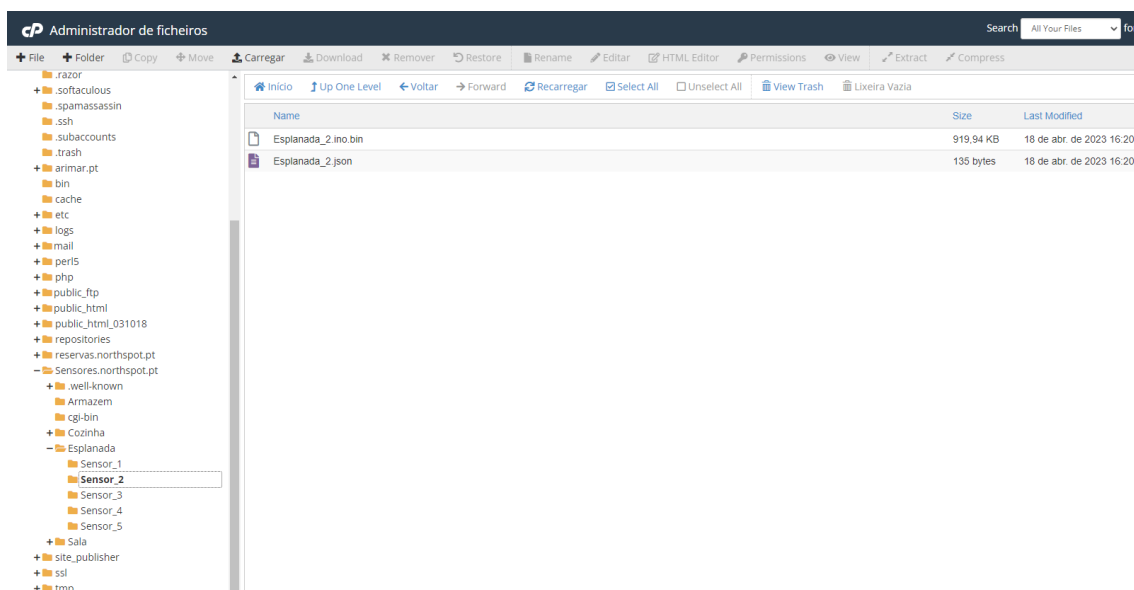


Figura 15 - Pasta do Servidor Web com o ficheiro binário e JSON

4.3 Sensores de Temperatura DS18B20



Figura 16 - Sensor DS18B20

O sensor DS18B20 é ideal para utilização em protoboard ou placas de circuitos definitivos.

É muito utilizado no desenvolvimento de projetos eletrônicos e robóticos através de placas microcontroladoras, entre elas, Arduino, Raspberry, ARM, AVR, PIC, etc pois possui apenas 1 pino com saída digital.

Fornece temperatura em graus Celcius com uma precisão 9-12 bits. A sua gama de temperaturas é -55°C a 125°C (+/-0.5°C).

Como cada chip DS18B20 contém um número de série único, podem ser usados vários sensores DS18B20 no mesmo barramento 1-wire. Isto permite a colocação de sensores de temperatura em vários locais. Aplicações onde este recurso é útil incluem HVAC, controlo de climatização ambiental, controlo de temperaturas no interior dos edifícios, equipamentos ou máquinas, etc.

Pode ser alimentado com 3.0~5.5V.

Características:

- Interface 1-Wire® requiere somente um pino para comunicação
- Cada dispositivo possui uma identificação única de 64bits armazenada na sua memória ROM
- Capacidade "Multidrop" simplifica aplicações de medição de temperatura distribuída
- Não necessita de componentes externos

- Pode ser alimentado pela linha de dados. Alimentação de 3.0V a 5.5V
- Gama de temperaturas de -55°C a $+125^{\circ}\text{C}$ (-67°F a $+257^{\circ}\text{F}$)
- Precisão $\pm 0.5^{\circ}\text{C}$ de -10°C a $+85^{\circ}\text{C}$
- Resolução de 9 a 12 bits configurada pelo utilizador
- Converte a temperatura numa word de 12-bit em 750ms (max.)
- Para ser usado em aplicações de controlo termostático, sistemas industriais, produtos de consumo, termómetros ou qualquer outro sistema termicamente sensível

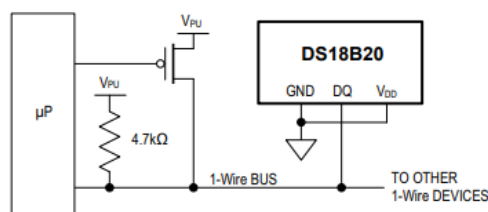


Figura 17 - Esquema do Sensor DS18B20

Para os equipamentos de congelação será usado o sensor de temperatura DS18B20.

Componente eletrónico digital desenvolvido para ser aplicado nos mais diversos ambientes, pois é capaz de medir a temperatura em locais húmidos, incluindo de baixo de água.



Figura 18 - Sensor DS18B20 com fio à prova de água

4.4 Bateria



Figura 19 - Bateria 3,7v de 2000mAh

A bateria utilizada para dar energia aos microprocessadores é uma bateria de 3,7v.

Foi escolhida por ter um desempenho confiável, sem fugas, excelente rentabilidade e longa vida útil.

Em termos de segurança é tem como pontes fortes a sobre carga, sobre descarga, sobre corrente, proteção contra curto-circuitos, proteção contra sobre temperatura.

Em termos de sobre descarga, nos testes efetuados ela corta a corrente aos 3,1v. Na sobre carga suporta aproximadamente $\pm 4,3v$ sem problemas. A previsão de duração da bateria, com a programação de paragens forçadas, será de sensivelmente 1 ano.

Características:

- 2000 mAh, 3,7 V
- Capacidade nominal e tensão: 2000 mAh, 3,7 V
- Corrente de carga padrão: 0,2C (400 mA)
- Tensão máxima de carga: 4.200 ± 0.020 V
- Corrente de descarga padrão: de 0,2 C (400 mA)
- Tensão de corte: $2,750 \pm 0,005$ V
- Corrente máxima de carga: 1C (2000 mA)
- Corrente máxima de descarga: 2C
- Temperatura de carregamento: $0 \sim 45$ °C
- Temperatura de descarga: $-20 \sim 60$ °C
- Temperatura de armazenamento: $-20 \sim 25$ °C
- Peso: 40G

Tabela 3 - Previsão de vida estimada da Bateria

Nº de Baterias	1
Tipo de Bateria	Li-Ion
Capacidade da Bateria	2000 mAh
Consumo de Corrente durante o repouso	0,05 mA
Consumo de Corrente durante a Execução	80 mA
Nº de execuções por hora	1
Duração da Execução	4000 ms
Vida Estimada da Bateria	607,4 Dias
	19,9 meses
	1,66 anos

4.5 Sensores Protótipos

4.5.1 Esquema da montagem dos componentes

Para a montagem dos componentes foi utilizado o Fritzing, que é um software de desenvolvimento Open Source para design de hardware eletrônico, com ele podemos construir um circuito mais permanente com uma Placa de Circuito Impresso.

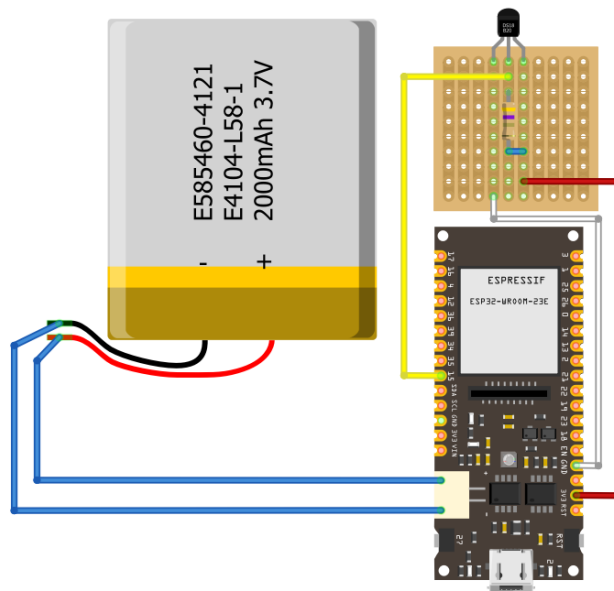


Figura 20 - Esquema dos Componentes

4.5.2 Versão Inicial



Figura 21 - Versão 1 do Sensor

A primeira versão do sensor foi feita com uma protoboard de 170 furos onde o microprocessador está ligado ao sensor, à resistência de 4,7 Ω e a vários jumpers.

4.5.3 Versão 2

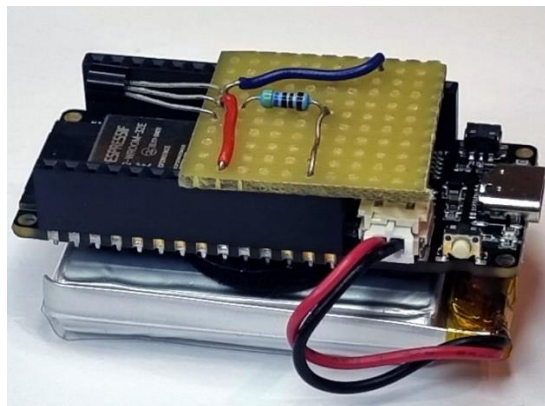


Figura 22 - Versão 2 do Sensor

Nesta segunda versão foi cortada uma placa de circuito impresso de uma face, com cerca de 250x300mm de 100 furos. Com esta alteração o protótipo ficou mais compacto aproximando-se do que se pretende para a versão final.

4.5.4 Versão para equipamentos de congelação



Figura 23 - Sensor para Equipamentos de Congelação

Este sensor para equipamentos de congelação foi construído com um cabo plano de 1mm. O microprocessador irá ficar na parte exterior do equipamento de congelação e o sensor no interior.

4.6 Preço custo dos materiais

Tabela 4 – Custo Total por sensor

	Custo
Microprocessador	9,36 €
Bateria	10,29 €
Sensor	0,64 €
Resistência	0,03 €
Fios/Jumpers	0,04 €
Placa de Circuito Impresso	0,34 €
Caixa Plástica	1,70 €
	22,40 €

Como podemos verificar o sensor fica a um preço bastante atrativo, estamos a falar num custo total para o sistema completo a $15 \times 22,40 \text{ €} = 336 \text{ €}$. Valor muito significativo dada a dimensão da estrutura implementada.

Se somarmos o Raspberry com caixa com o custo aproximado de 150€ teremos no total um sistema implementado por menos de 500€.

4.7 MQTT

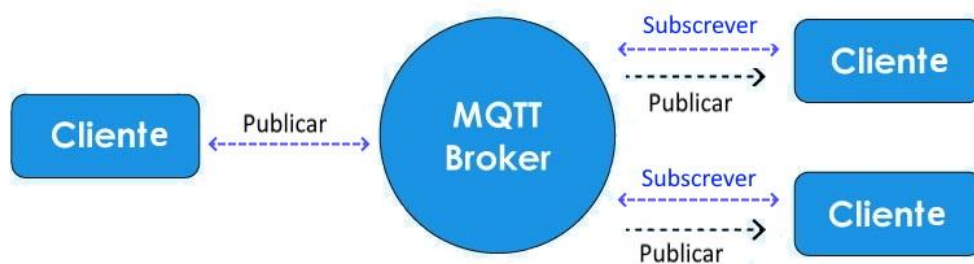


Figura 24 - MQTT Broker Exemplo

MQTT, é um protocolo de mensagens leve para sensores e pequenos dispositivos móveis otimizado para redes TCP/IP. O esquema de troca de mensagens é fundamentado no modelo Publicador-Subscritor, extremamente simples e leve. Os princípios arquitetónicos são minimizar o uso de banda de rede e uso de recursos dos equipamentos enquanto garantindo confiabilidade e algum nível de garantia de entrega. Estes princípios tornam este protocolo ideal para as comunicações emergentes (M2M) “machine-to-machine” e para as aplicações “Internet of Things” (Internet das coisas) um mundo de equipamentos conectados.

O MQTT define três níveis de Qualidade de Serviço (QoS). A QoS define o quanto o broker/cliente tentará garantir que uma mensagem seja recebida. As mensagens podem ser enviadas em qualquer nível de QoS e os clientes podem tentar se inscrever em tópicos em qualquer nível de QoS. Isso significa que o cliente escolhe a QoS máxima que receberá. Por exemplo, se uma mensagem for publicada em QoS 2 e um cliente estiver inscrito com QoS 0, a mensagem será entregue a esse cliente com QoS 0. Se um segundo cliente também estiver inscrito no mesmo tópico, mas com QoS 2, então ele receberá a mesma mensagem, mas com QoS 2. Para um segundo exemplo, se um cliente for inscrito com QoS 2 e uma mensagem for publicada em QoS 0, o cliente a receberá em QoS 0.

Níveis mais altos de QoS são mais confiáveis, mas envolvem maior latência e têm requisitos de largura de banda mais altos.

- 0: O broker/cliente entregará a mensagem uma vez, sem confirmação.
- 1: O corretor/cliente entregará a mensagem pelo menos uma vez, sendo necessária a confirmação.
- 2: O corretor/cliente entregará a mensagem exatamente uma vez usando um handshake de quatro etapas.

Tabela 5 - Tipos de Mensagens MQTT

Valor	Nome	Direção	Descrição
0	Reservado	Proibido	Reservado
1	CONNECT	Cliente para Servidor	Requisição do cliente para conectar ao servidor
2	CONNACK	Servidor para Cliente	Reconhecimento da conexão
3	PUBLISH	Cliente para Servidor ou Servidor para cliente	Publicar mensagem
4	PUBACK	Cliente para Servidor ou Servidor para cliente	Reconhecimento da publicação

5	PUBREC	Publicação recebida	Publicação recebida (parte 2 do QoS=1)
6	PUBREL	Cliente para Servidor ou Servidor para cliente	Publicação lançada (parte 2 do QoS=2)
7	PUBCOMP	Cliente para Servidor ou Servidor para cliente	Publicação completa (parte 3 do QoS=2)
8	SUBSCRIBE	Cliente para Servidor	Pedido de inscrição
9	SUBACK	Servidor para cliente	Reconhecimento de inscrição
10	UNSUBSCRIBE	Cliente para Servidor	Pedido de desinscrição
11	UNSUBACK	Servidor para cliente	Reconhecimento desinscrição
12	PINGREQ	Requisição	Requisição PING
13	PINGRESP	Servidor para cliente	Resposta PING
14	DISCONNECT	Cliente para Servidor	Cliente está desconectado
15	Reservado	Proibido	Reservado

4.7.1 Broker MQTT com Raspberry Pi

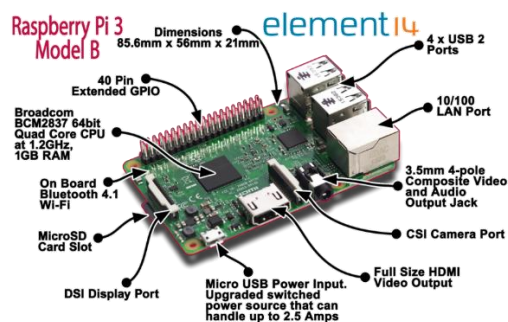


Figura 25 - Raspberry Pi 3(Fonte <https://www.raspberrypiportugal.pt/>)

Raspberry Pi é um computador de placa única que cabe na mão, onde alguns dos seus modelos são praticamente do tamanho de um cartão de crédito. Oferece todos os

recursos necessários para utilizar esse tipo de equipamento, podendo conectar na sua placa os dispositivos: mouse, teclado e monitor, funcionando assim como um computador desktop!

O Raspberry Pi foi desenvolvido pela Fundação Raspberry Pi, no Reino Unido, tendo o seu primeiro modelo lançado em 2012. Passou por várias atualizações, mais ou menos quinze no total, mas sempre evoluindo e aumentando as suas habilidades através de novos recursos de acordo com os avanços da tecnologia. Hoje são comercializados quatro modelos: ZeroW, 3 A+, 3 B+ e 4 B, sendo recomendado a utilização dos dois últimos modelos em uma automação residencial.

Especificações:

- Raspberry Pi 3 Model B Anatel
- Processador Broadcom BCM2837 64bit ARMv8 Cortex-A53 Quad-Core Clock 1.2 GHz
- Memória RAM: 1GB
- Adaptador Wifi 802.11n integrado (trabalha na frequência de 2.4 Ghz)
- Bluetooth 4.1 BLE integrado
- Conector de vídeo HDMI
- 4 portas USB 2.0
- Conector Ethernet
- Interface para câmara (CSI)
- Interface para display (DSI)
- Slot para cartão microSD
- Conector de áudio e vídeo
- GPIO de 40 pinos
- Dimensões: 85 x 56 x 17mm

4.7.2 Instalação do Broker MQTT Mosquitto

A instalação do Mosquitto é feita com o seguinte comando:

sudo apt install -y mosquitto mosquitto-clients

Em seguida com os seguintes comandos ativamos, iniciamos e paramos o serviço do Mosquitto:

sudo systemctl enable mosquitto.service

sudo systemctl start mosquitto.service

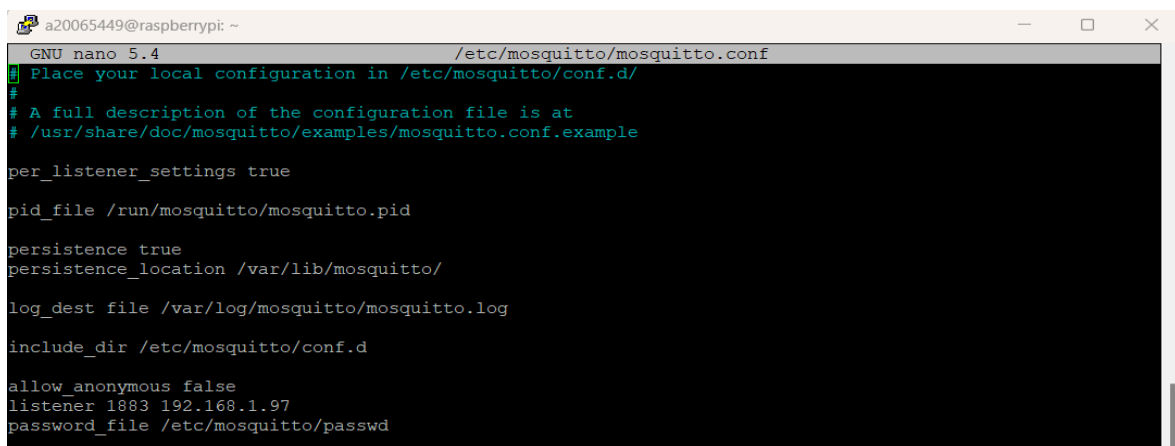
sudo systemctl stop mosquitto.service

Definir um user e password para acesso ao Broker

sudo mosquitto_passwd -c /etc/mosquitto/passwd a20065449

Para garantir a segurança no acesso ao Broker, no ficheiro *mosquitto.conf* foi definido que o acesso não poderá ser feito anónimo. Também definir a porta de acesso, o IP e o ficheiro onde definimos a password.

Sudo nano /etc/mosquitto/mosquitto.conf



```
GNU nano 5.4 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

per_listener_settings true

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

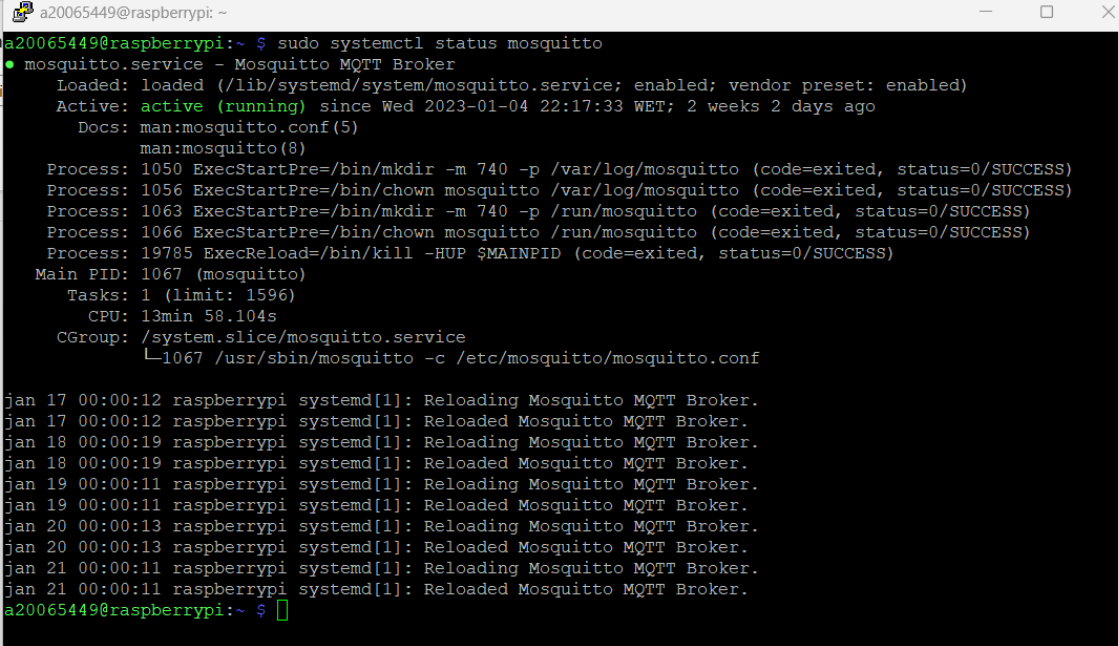
include_dir /etc/mosquitto/conf.d

allow_anonymous false
listener 1883 192.168.1.97
password_file /etc/mosquitto/passwd
```

Figura 26 - Configuração do Mosquitto

Com o comando abaixo podemos verificar o estado do servidor Mosquitto.

sudo systemctl status mosquitto



```
a20065449@raspberrypi: ~
a20065449@raspberrypi:~ $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-01-04 22:17:33 WET; 2 weeks 2 days ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Process: 1050 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 1056 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 1063 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 1066 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 19785 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
  Main PID: 1067 (mosquitto)
    Tasks: 1 (limit: 1596)
         CPU: 13min 58.104s
   CGroup: /system.slice/mosquitto.service
           └─1067 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

jan 17 00:00:12 raspberrypi systemd[1]: Reloading Mosquitto MQTT Broker.
jan 17 00:00:12 raspberrypi systemd[1]: Reloaded Mosquitto MQTT Broker.
jan 18 00:00:19 raspberrypi systemd[1]: Reloading Mosquitto MQTT Broker.
jan 18 00:00:19 raspberrypi systemd[1]: Reloaded Mosquitto MQTT Broker.
jan 19 00:00:11 raspberrypi systemd[1]: Reloading Mosquitto MQTT Broker.
jan 19 00:00:11 raspberrypi systemd[1]: Reloaded Mosquitto MQTT Broker.
jan 20 00:00:13 raspberrypi systemd[1]: Reloading Mosquitto MQTT Broker.
jan 20 00:00:13 raspberrypi systemd[1]: Reloaded Mosquitto MQTT Broker.
jan 21 00:00:11 raspberrypi systemd[1]: Reloading Mosquitto MQTT Broker.
jan 21 00:00:11 raspberrypi systemd[1]: Reloaded Mosquitto MQTT Broker.
a20065449@raspberrypi:~ $
```

Figura 27 - Servidor Mosquitto Ativo

Para garantir a inicialização Automática Mosquitto

Para fazer com que o Mosquitto comece automaticamente depois do sistema inicializar ou reinicializar foi criada uma pasta.

/home/pi/.config/autostart.

Dentro desta pasta foi criado um ficheiro mosquitto.desktop com o seguinte comando:

sudo nano /home/a20065449/.config/autostart/mosquitto.desktop

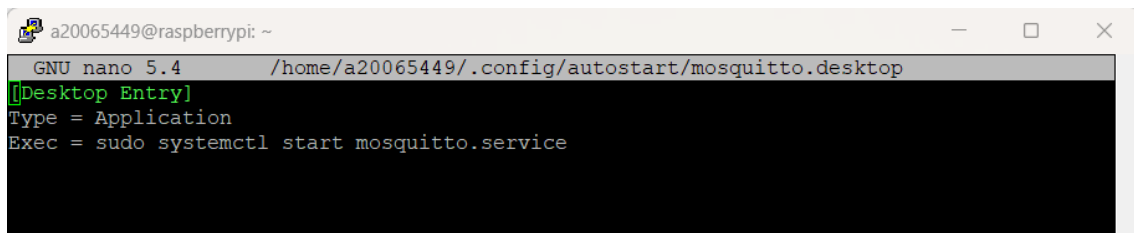
O conteúdo deste ficheiro terá as seguintes linhas de código.

[Desktop Entry]

Type = Application

Exec = sudo nano /home/a20065449/.config/autostart/mosquitto.desktop

Será muito útil caso haja uma falha de corrente elétrica.

A screenshot of a terminal window on a Raspberry Pi. The window title is 'a20065449@raspberrypi: ~'. The terminal shows the GNU nano 5.4 editor editing the file '/home/a20065449/.config/autostart/mosquitto.desktop'. The content of the file is:

```
[[Desktop Entry]
Type = Application
Exec = sudo systemctl start mosquitto.service
```

4.7.3 Tópicos MQTT

4.7.3.1 Um único tópico por mensagem

Utilizar um único tópico por mensagem, fica mais fácil identificar e gerenciar as mensagens. Também ajuda a garantir que cada mensagem seja entregue ao destinatário certo. Se vários tópicos forem usados para uma mensagem, há uma hipótese grande de que alguns dos destinatários não recebam a mensagem devido ao roteamento incorreto.

Usar um único tópico por mensagem também ajuda a manter o broker MQTT organizado. Quando todas as mensagens têm seu próprio tópico exclusivo, é muito mais fácil localizá-las no futuro, se necessário. Isso pode ser especialmente útil ao solucionar um problema.

4.7.3.2 Tópico curto e simples

O protocolo MQTT foi projetado para ser leve e eficiente, por isso é importante que os tópicos sejam o mais curtos possível. Tópicos mais longos podem causar sobrecarga desnecessária na rede, o que pode levar a um desempenho mais lento e maior latência. Além disso, tópicos mais longos podem ser mais difíceis de ler e entender, tornando a depuração e a solução de problemas mais desafiadoras. Para manter tópicos curtos e simples, uma estrutura hierárquica com vários níveis de subtópicos.

4.7.3.3 Tópicos utilizados

TFC/Zona/Sensor_N

Onde:

TFC – Neste caso o nome da disciplina.

Zona – Local onde se encontra o sensor, 4 no total.

Sensor_N – Sensor número N de N.

Mantendo assim os tópicos curtos e simples com 3 níveis de hierarquia.

<p><i>TFC/Cozinha/Sensor1</i></p> <p><i>TFC/Cozinha/Sensor2</i></p> <p><i>TFC/Cozinha/Sensor3</i></p> <p><i>TFC/Cozinha/Sensor4</i></p>	<p><i>TFC/Armazem/Sensor1</i></p> <p><i>TFC/Armazem/Sensor2</i></p> <p><i>TFC/Armazem/Sensor3</i></p> <p><i>TFC/Armazem/Sensor4</i></p>
<p><i>TFC/Sala/Sensor1</i></p> <p><i>TFC/Sala/Sensor2</i></p> <p><i>TFC/Sala/Sensor3</i></p>	<p><i>TFC/Esplanada/Sensor1</i></p> <p><i>TFC/Esplanada/Sensor2</i></p> <p><i>TFC/Esplanada/Sensor3</i></p> <p><i>TFC/Esplanada/Sensor4</i></p> <p><i>TFC/Esplanada/Sensor5</i></p>

Figura 28 - Lista de Tópicos MQTT

4.7.3.4 Estrutura da mensagem enviada

A mensagem dos tópicos é enviada no formato de Json com a seguinte estrutura.

```
{
  "Sensor_ID": "28e3e75d013c25",
  "Temperatura": 21.375,
  "Bateria": 4.153999805,
  "WIFI": -45,
  "Tipo": 2,
  "Ativo": "S",
  "ESP_IP": "192.168.1.104",
  "VERSAO": 2,
  "Nome": "Esplanada - Arca dos Gelados"
}
```

```
{
    "Sensor_ID":"281b5d75d013c2d": Tipo String.

    "Temperatura":21.4375: Tipo Float

    "Bateria":4.226: Tipo Float

    "WIFI":-53: Tipo Int

    "Tipo":1: Tipo Int

    "Ativo":"S": Tipo Char

    "ESP_IP":"192.168.1.127": Tipo String

    "VERSAO":2, Tipo Int

    "Nome":"Arca Congeladora 4": Tipo String
}
```

Sensor_ID – Indica um número único de cada sensor de temperatura DS18B20 ligado no ESP32.

Temperatura – Valor da temperatura lido através do sensor.

Bateria – Valor bateria lido através do modulo da bateria incorporado no ESP32.

WiFi – Indica o nível de intensidade do sinal de WiFi.

Tipo – Corresponde ao tipo de equipamento em que estão a ser lidos os registos.

Ativo – Todos os Sensores estão inicialmente ativos, posteriormente podem ser

ESP_IP – Indica qual o endereço IP do ESP32.

Versão – Corresponde à versão do código instalado no ESP32.

Nome – Indica o nome do equipamento onde está o sensor.

4.8 Base de Dados

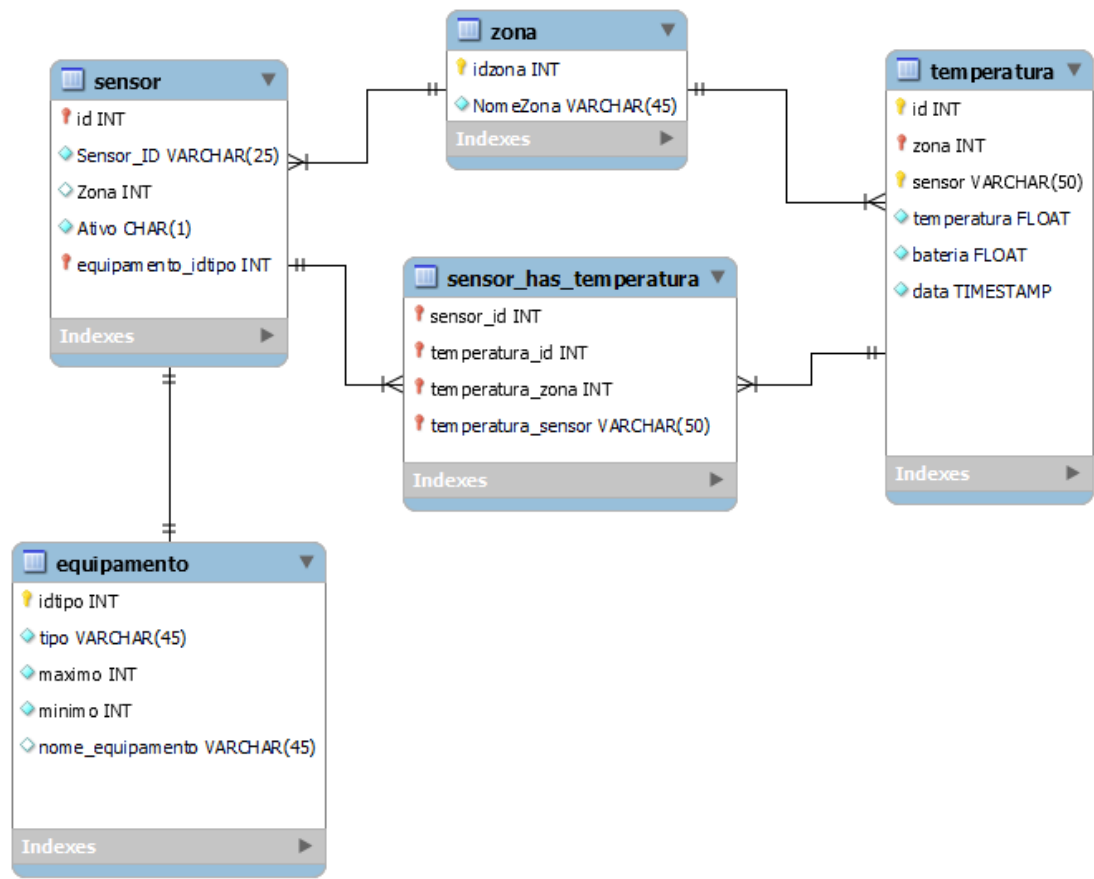


Figura 29 - Modelo Entidade Relação

Foi feito um levantamento do tipo de equipamento e zonas onde instalar os sensores. Deram origem às seguintes tabelas.

Tabela 6 - Identificador e Atributos da Base de Dados

Identificador	Atributos	Grau
Zona	Idzona, NomeZona	2
Sensores	Id, Sensor_ID, Zona, Ativo, equipamento_idtipo	5
Equipamento	idTipo, Tipo, maximo, minimo, nome_equipamento	5

Temperatura	Id, zona, sensor, temperatura, bateria, data	6
-------------	--	---

4.9 NODE.JS

De acordo com o site (<https://nodejs.org/en>), o *Node.js* é um ambiente de runtime de Javascript baseado no motor de Javascript V8 (este é o motor de Javascript usado pelo browser Chrome). Do ponto de vista arquitetural, o *Node.js* caracteriza-se por utilizar um modelo direcionado a eventos, onde as operações de I/O são efetuadas de forma assíncrona para evitar bloqueios desnecessários. Atualmente, o seu ecossistema de pacotes disponibiliza o maior número de bibliotecas open source existentes no mundo, que podem ser utilizadas em qualquer aplicação através do uso do utilitário *npm*.

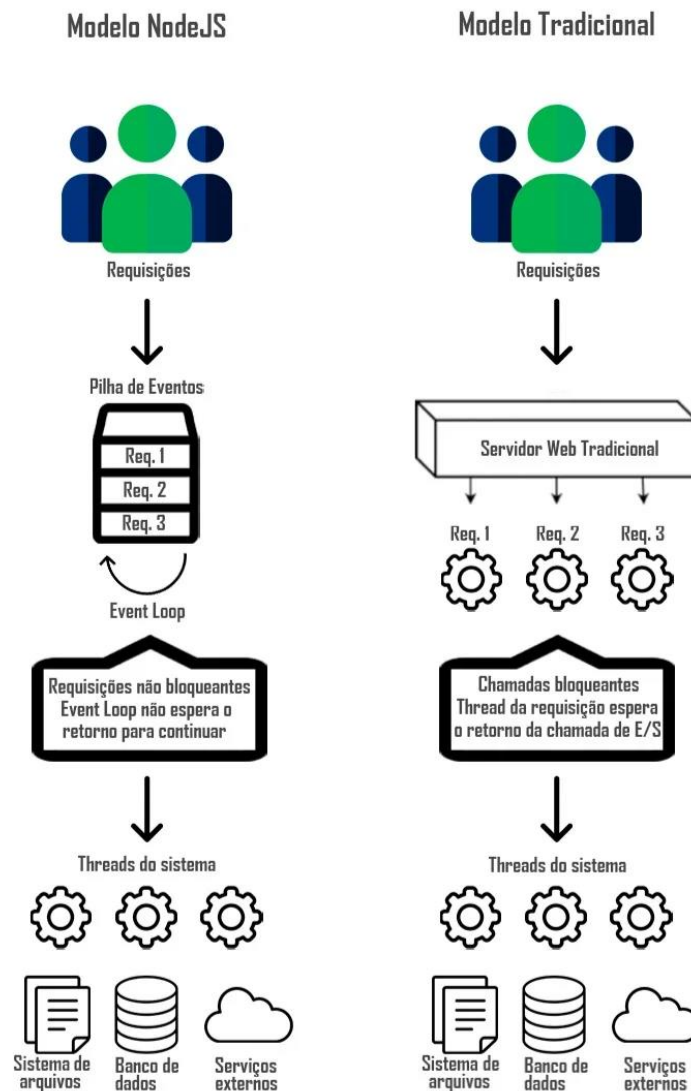


Figura 30 - Modelo Funcional de um Servidor Tradicional vs Servidor Node.js

4.10 Node-RED

O Node-RED, é uma ferramenta visual de ambiente de código aberto, que inicialmente foi desenvolvida para implementar, criar e/ou conectar dispositivos de IoT, tendo sido estendida posteriormente para hardwares, APIs e web services. Assim sendo, por meio dos nodes ou nós é possível ler arquivos CSV, escutar eventos http, tcp, websocket, twitter, mqtt entre outros. O editor é baseado no navegador que além de simples e compatível com todos os browsers, facilita a conexão de fluxos usando os nós (nodes). E assim como o protocolo MQTT o Node-RED também foi criado pela IBM Emerging Technology.

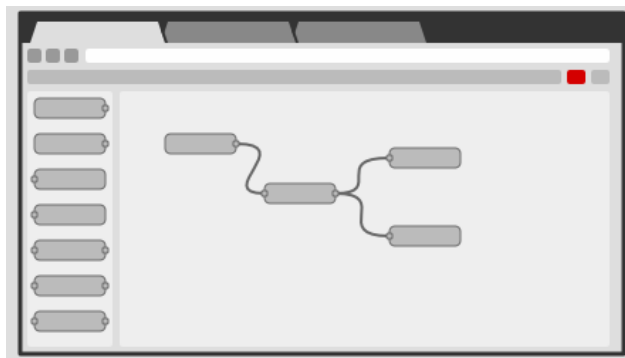


Figura 31 - Edição de fluxo baseada no navegador

O Node-RED fornece um editor de fluxo baseado em navegador que facilita a conexão de fluxos usando a ampla variedade de nós. Os fluxos podem ser implantados no tempo de execução com um único clique. As funções JavaScript podem ser criadas no editor usando um editor de rich text.

O tempo de execução leve é construído em Node.js, aproveitando ao máximo o modelo sem bloqueio e orientado a eventos. Isso o torna ideal para execução na borda da rede em hardware de baixo custo, como o Raspberry Pi, bem como na nuvem.

Com mais de 225.000 módulos no repositório de pacotes do Node, é fácil estender a gama de nós da paleta para adicionar novos recursos.

Uma biblioteca integrada permite salvar funções, modelos ou fluxos úteis para reutilização.

Entre as características do Node-red destaca-se a possibilidade de poder ser desenvolvido em qualquer Sistema Operacional. Entretanto, para que seja possível proceder com a instalação da dependência do Node-RED, dependência esta que pode ser entendida como um pacote de módulos, é preciso que já esteja instalado o Node.js.

Uma das grandes vantagens do Node-RED é que ele não funciona apenas para a parte lógica, mas conseguimos criar telas, interfaces gráficas feitas em angular através de um módulo muito utilizado e cuidado pela comunidade, o dashboard. O módulo de dashboard uma caixa de ferramentas composta de botões, sliders, caixas de texto, formulários, gráficos para construirmos uma tela e obter informações importantes, isso

é excelente para aqueles produtos que precisam de um protótipo rápido para validação de pessoas em massa.

Também é possível se criar telas de forma comum usando HTML e CSS através de nós simples de template e requisições http.

O fato de ser uma plataforma de programação em blocos, não limita o Node-RED somente a isso, o Node-RED nos entrega diversos módulos de programação e conversão de linguagens como Python, .NET Core para serem utilizadas dentro dele, bem como acesso a diversos tipos de banco de dados tanto relacionais como não relacionais. Exemplo: mysql, sql Server, MongoDB, DynamoDB dentre tantos outros.

4.10.1 Esquemas de Configuração no NodeRed

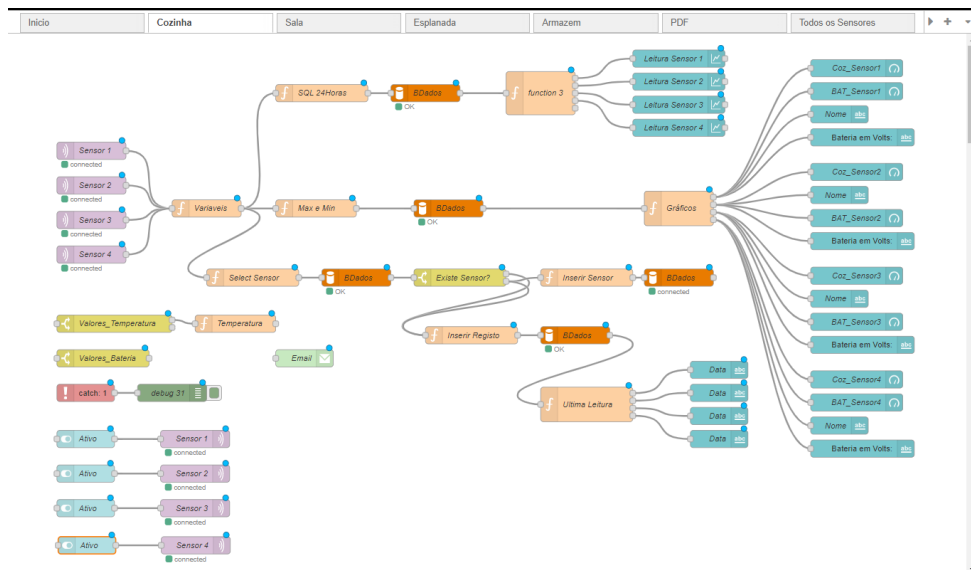


Figura 32 - Desenho dos Módulos no Node-Red

4.10.2 MQTT no Node-Red

Figura 33 - Configuração do MQTT no Node-Red

A configuração do MQTT é definida indicando o IP e a porta do Broker, bem como o utilizador e password.

O nó dos sensores indicam qual o estado em que está a ligação ao MQTT, indicando se está conetado ou desconetado. Como demonstra a figura abaixo, a ligação ao broker de todos os sensores, neste caso na cozinha, estão ligados.

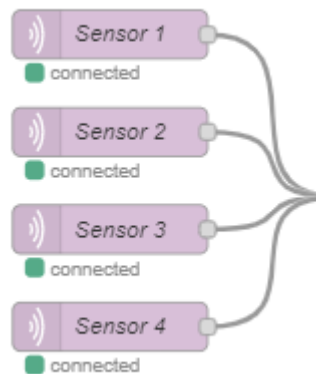


Figura 34 - Estado da ligação ao Broker

Sempre que chega, uma mensagem JSON através do MQTT, ela é separada por várias variáveis que serão utilizadas para a representação gráfica, dos valores recebidos, no dashboard.

Durante este processo é verificado se o ID do Sensor já existe, caso seja um novo insere na tabela.

É feito uma query para ler os valor anteriores para serem exibidos no gráfico.

4.10.3 Página de monitorização

A página terá um menu onde é possível navegar e seleccionar outra zona para visualização dos dados existentes dos sensores.

O menu é composto pelas quatro zonas e por uma listagem dos últimos 24 dos registos das temperaturas por cada zona.

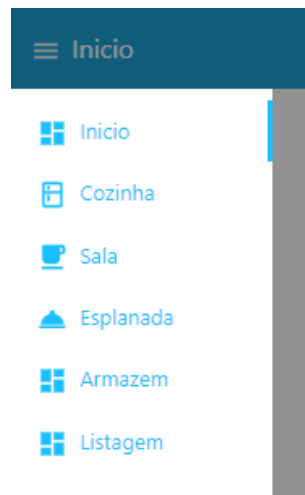


Figura 35 - Menu barra lateral

Foi criado um DDNS para se aceder à pagina dos sensores externamente. É validado o acesso ao site através de user e password.

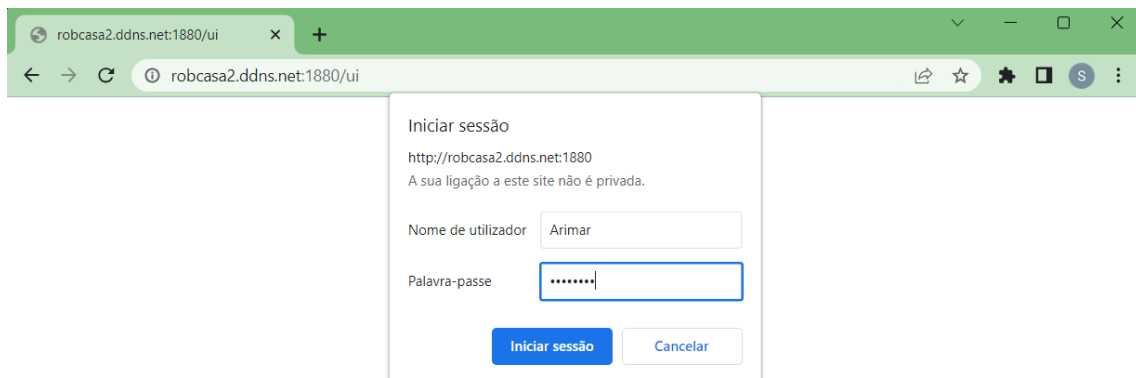


Figura 36 - Página Inicial

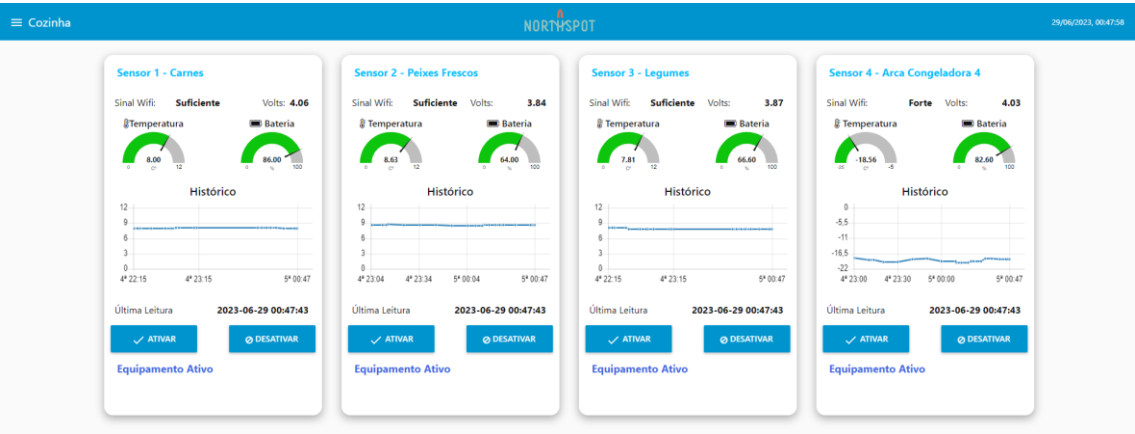


Figura 37 – Visualização dos Estados dos Sensores na Cozinha

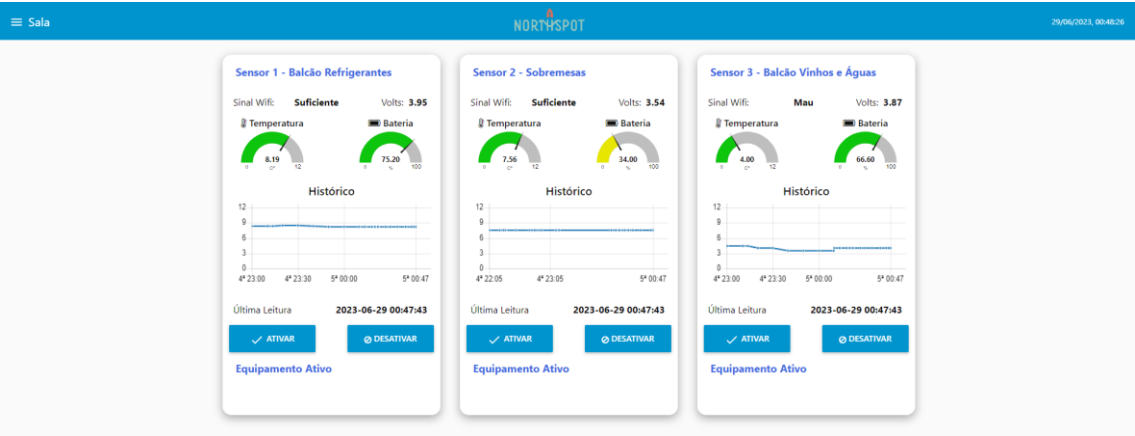


Figura 38 - Visualização dos Estados dos Sensores na Sala



Figura 39 - Visualização dos Estados dos Sensores no Armazém

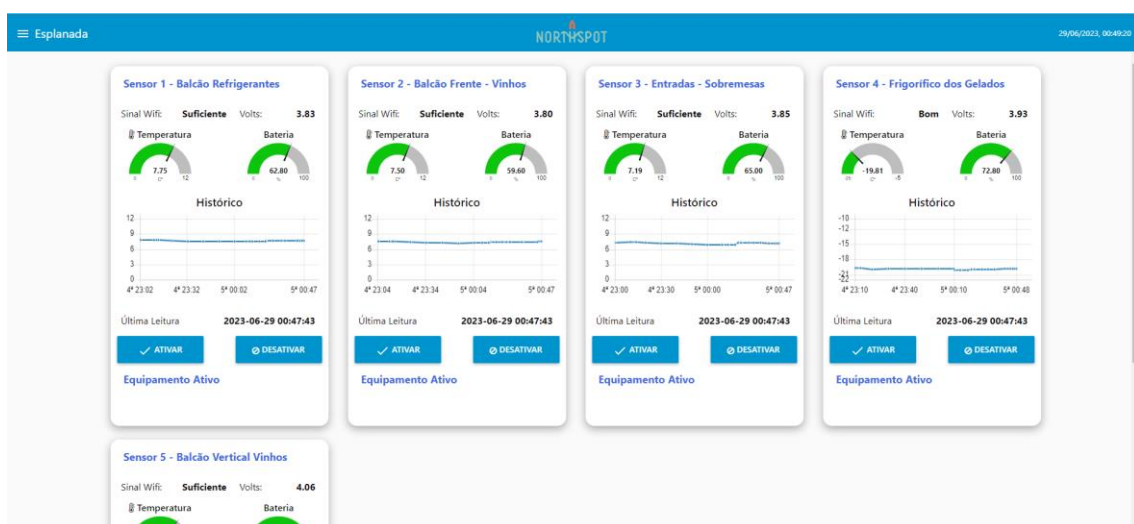


Figura 40 - Visualização dos Estados dos Sensores na Esplanada

Existe a possibilidade de alteração do estado do sensor, em que, pode estar ativo ou inativo. Em modo ativo, regista normalmente as temperaturas, o estado da bateria e envia alertas se for caso disso.

Para se alterar o estado para inativo existem 3 possibilidades, Avaria, Carregar Bateria ou Desativar, o sensor continua a enviar as mensagens para o Broker mas não emite qualquer alerta.

1. **Avaria:** O equipamento de frio tem uma avaria, e precisa de ser reparado.
2. **Carregar Bateria:** O sensor está a carregar a bateria.
3. **Desativar:** O equipamento de frio está desligado.

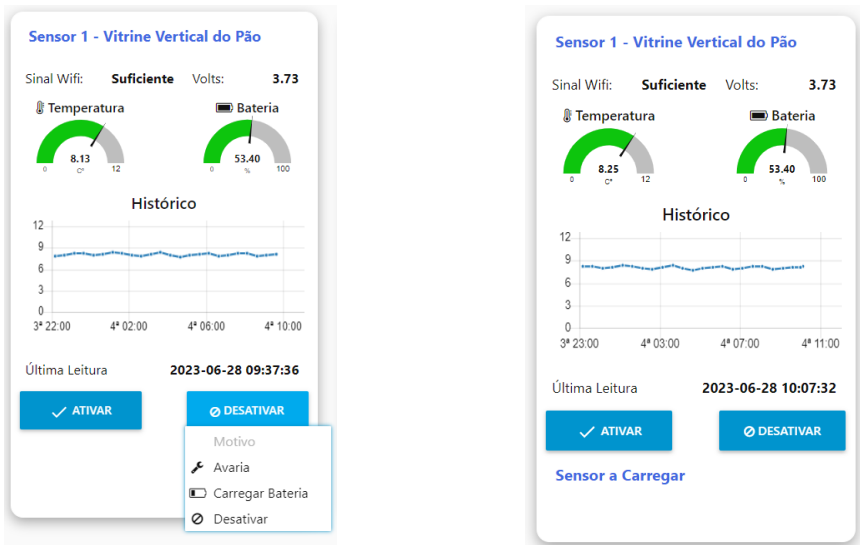


Figura 41 -Alteração do Estado do Sensor

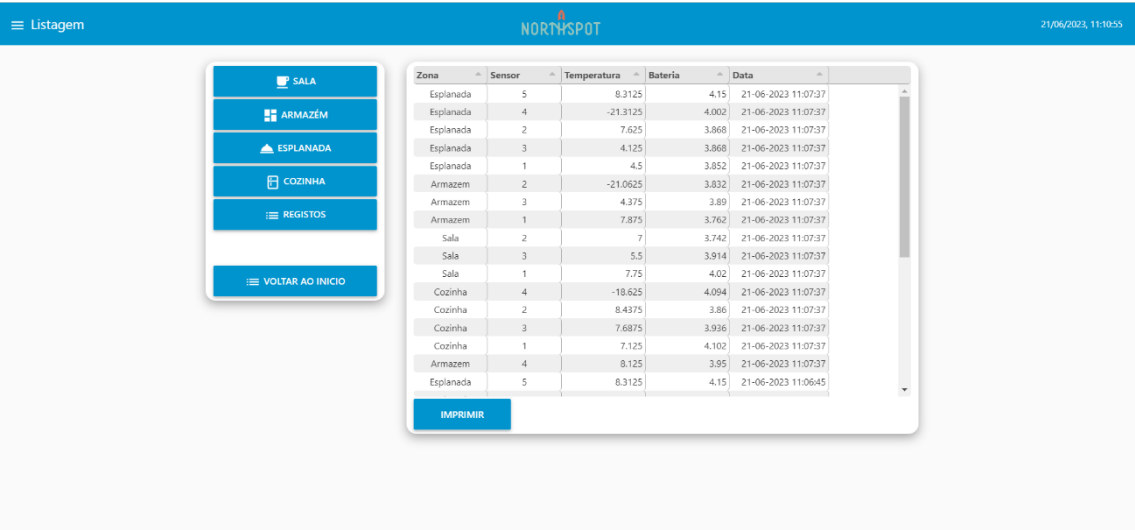


Figura 42 – Listagem de todas as zonas dos últimos 24 registos

A listagem temperaturas é feita, inicialmente para todas as zonas e todos os sensores. Podendo ser escolhida as zonas na pesquisa bem como o selecionar sensor. Desta forma, consegue-se a listagem para todos os sensores existentes no sistema.

4.11 Twilio

A Twilio é a plataforma de comunicação na nuvem líder mundial que permite envolver os clientes em todos os canais: SMS, voz, vídeo, e-mail, WhatsApp, Facebook Messenger, Push Notification e muito mais. Para empresas que precisam de serviços de comunicação para integrar com Sites, App, Sistemas Web, CRM e muito mais.

A Twilio tem diversas APIs que permitem que desenvolvedores consigam criar aplicações e integrações com suas soluções. Sem sombra de dúvida uma das melhores plataformas de comunicação do mercado.

Com esta ferramenta será possível controlar o estado dos dispositivos, saber ao receber um SMS qual o dispositivo em questão e qual o problema do alerta.

4.11.1 Whatsapp

No programa foi utilizado uma API para envio dos alertas por mensagem. No caso, foi utilizado o callmebot para esse mesmo envio

Para utilizar este serviço é necessário obter uma chave do bot antes de utilizar a API:

Passos para configurar o callmebot:

1. Adicionar o número de telefone +34 644 41 87 20 aos contactos telefónicos.
2. Enviar esta mensagem "**I allow callmebot to send me messages**" para o novo contato.
3. Aguardar até receber a mensagem "**API Activated for your phone number. A sua APIKEY é xxxxxx**" do bot.
4. A mensagem do WhatsApp do bot contem a apikey necessária para enviar mensagens usando a API.
5. Só depois de receber a confirmação pode começar a enviar mensagens.

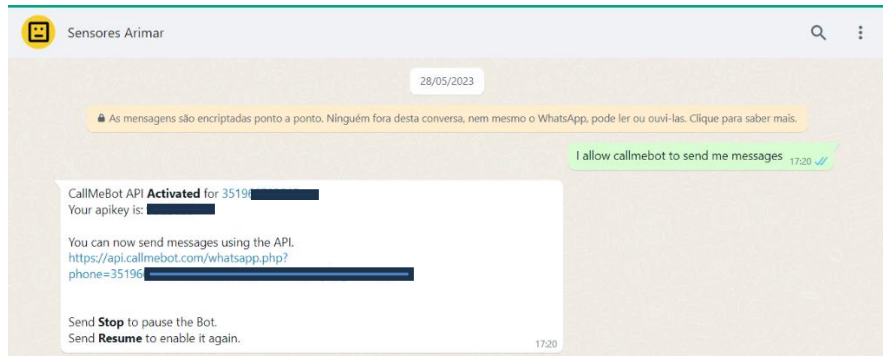
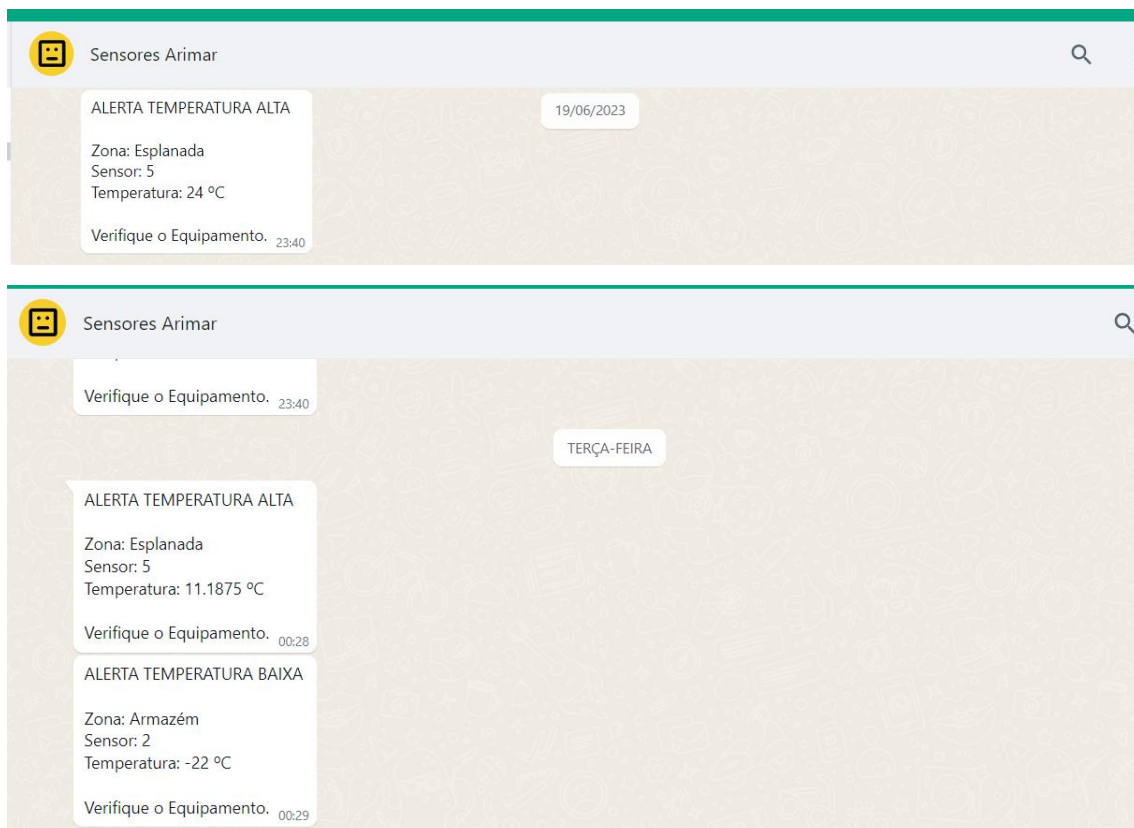


Figura 43 - Callmebot Whatsapp

Quando os valores estão fora dos parâmetros definidos o sistema envia por Whatsapp.



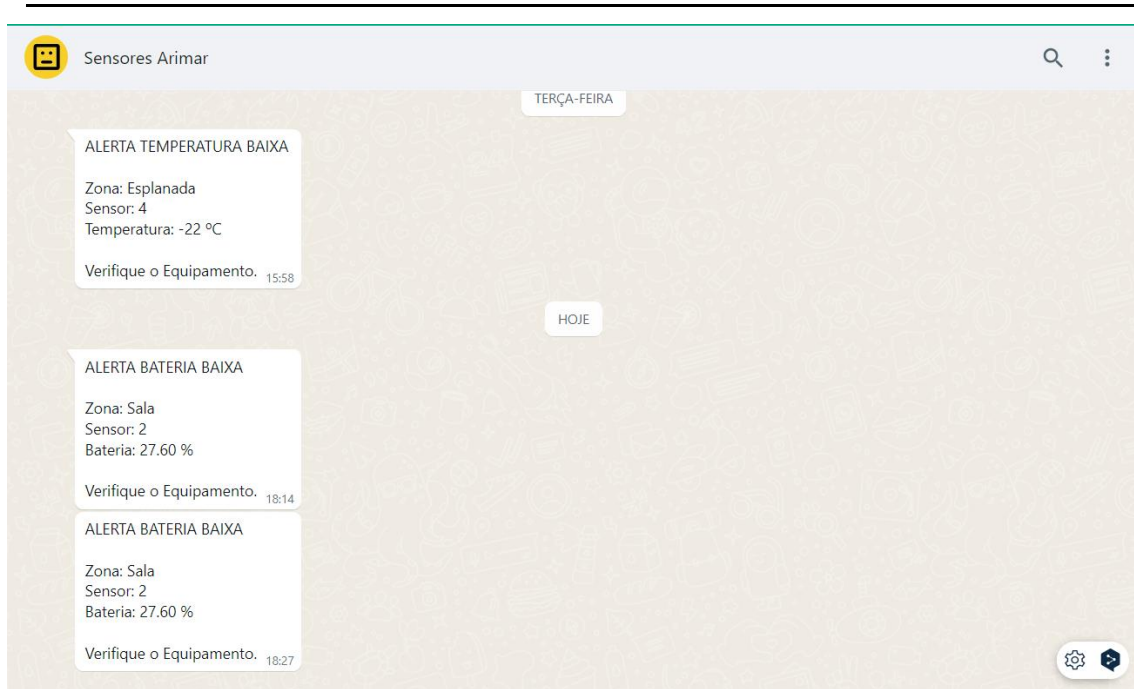


Figura 44 - Exemplos de Mensagem Whatsapp

5 Calendário

TAREFA	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul
ESP32										
Identificar o Arduino a utilizar										
Sensor Temperatura										
Configurar ESP32 para utilização										
Montagem dos componentes										
Wifi Manager para os ESP32										
Atualização OTA										
MQTT										
Configurar o Raspberry Pi										
Configurar MQTT Local										
Segurança										
Base de Dados	-	-	-	-	-	-	-	-	-	-
Criação da Base de Dados										
Node Red										
Configurar Node										
Configurar Envio de Email										
Dashboard de Monitorização										
Envio de SMS										
Impressão Pdf										
Testes										
Testar Arduino e Sensores										
Relatórios										
Relatório Intercalar 1º Semestre										
Relatório Intermédio										
Relatório Intercalar 2º Semestre										
Relatório Final										

Bibliografia

[DEISI22]

DEISI, Regulamento de Trabalho Final de Curso, Set. 2022.

[ULHT22]

Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, Out. 2022.

[MQTT]

MQTT Header, <http://www.rfwireless-world.com/Tutorials/MQTT-tutorial.html>, outubro de 2022.

[Whatsapp]

Callmebot, <https://www.callmebot.com/>, maio de 2023.

[NODE-RED]

Node-Red, <https://nodered.org/>, janeiro de 2023.

[Mosquitto]

Mosquitto, <https://mosquitto.org/>, outubro de 2022.

[MYSql]

MYSql, <https://www.mysql.com/>, fevereiro de 2023.

[Tutoriais]

Varios tutoriais sobre arduinos e node-red,. <https://randomnerdtutorials.com/>, outubro de 2022.

Apêndice 1 – Requisitos

Tabela 7 - Tabela de Requisitos

Tabela de requisitos			
Título	Descrição	Critério de aceitação	Implementado
ESP32 Arduino			
Ligar à WIFI	Todos os microprocessadores devem ligar ao WiFi para haver comunicação.	Obrigatório	Sim
Ligar ao MQTT	Comunicação ao protocolo MQTT através dos tópicos definidos inicialmente.	Obrigatório	Sim
Poupar Bateria	O Microprocessador deverá poupar ao máximo energia.	Obrigatório	Sim
Configurar ESP32 na primeira ligação à Wifi	O ESP32 é configurado na primeira ligação à rede a que foi conectado.	Opcional	Não
ESP32 OTA	Enviar código para ESP32 via wifi e não com cabo USB	Opcional	Sim
Sensores			
DS18B20	Testar todos os sensores para verificação de captura de temperatura.	Obrigatório	Sim
MQTT			
Criar Broker	Criação de um broker local para envio de mensagens através do protocolo.	Obrigatório	Sim
Raspberry Pi	Configurar o Raspberry Pi como Broker	Obrigatório	Sim

Tópicos	Tópicos com os nomes dos sensores.	Obrigatório	Sim
Base de Dados			
Criação	Implementação da base de dados.	Obrigatório	Sim
Acesso	Acesso à base de dados	Obrigatório	Sim
Notificações			
SMS	Envio de SMS(whatsapp) sempre que 1 Sensor envia dados fora dos parâmetros definidos.	Obrigatório	Sim
Email	Envio de email sempre que 1 Sensor envia dados fora dos parâmetros definidos.	Obrigatório	Sim
Bateria	O sistema deverá alertar para que a bateria do dispositivo seja carregada.	Obrigatório	Sim
Node-Red			
Criação de Fluxos	Criação de fluxos para as comunicações recebidas e enviadas para o ESP32.	Obrigatório	Sim
Dashboard	Implementação da Dashboard para todos os sensores.	Obrigatório	Sim
Impressão de Relatório	Botão para imprimir os relatórios de temperatura por sensor.	Obrigatório	Sim

Apêndice 2 – Vídeo do TFC

Vídeo 1 Parte - <https://youtu.be/LlkvtCKdIRA>

Vídeo 2 Parte - <https://www.youtube.com/watch?v=BAzV6uTLDok>

Vídeo 3 Parte - <https://www.youtube.com/watch?v=PMZ0a8ygDvc>

Apêndice 3 – Código do programa

<https://github.com/a20065449/TFC>

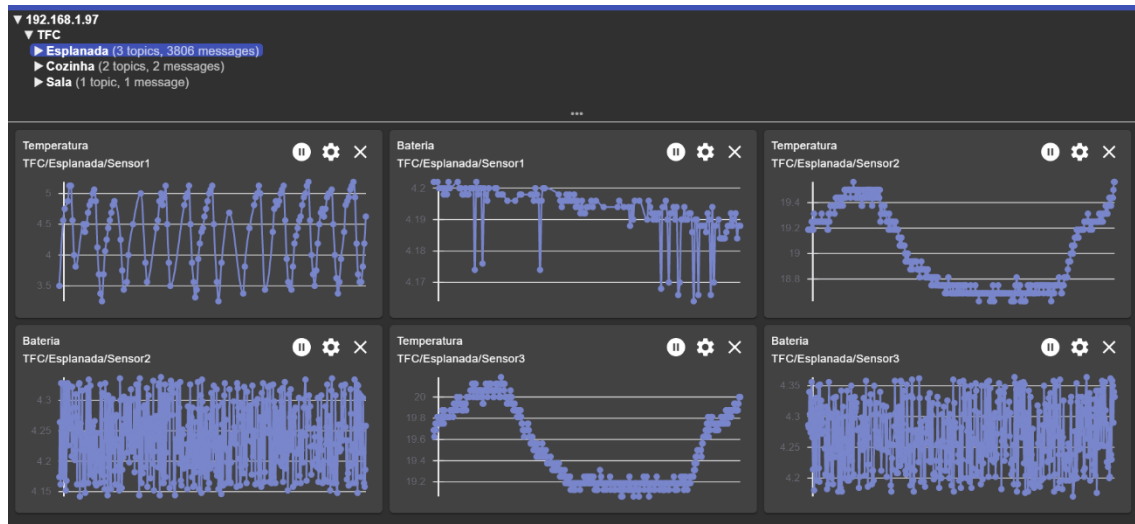
Glossário

LEI	Licenciatura em Engenharia Informática
TFC	Trabalho Final de Curso
MQTT	Message Queuing Telemetry Transport
IDE	Integrated Development Environment
URL	Uniform Resource Locator
DDNS	Dynamic Domain Name System

6 Testes

6.1 MQTT

Mensagens MQTT recebidas



Durante o desenvolvimento verificou-se que o envio das mensagens através dos ESP32 estava a decorrer de forma natural.

6.2 Base de Dados

O começo dos registos gravados na base de dados iniciaram-se em fevereiro e neste momento tem mais de 1.000.000 registos só de testes. Lembrar que, em muitos processadores, a leitura pode ser de 10 minutos e alguns de 30 minutos e outros de 1h30. No final todos estarão com 1h30 entre cada leitura.

```
1 • SELECT * FROM sensores.registos;
```

id	zona	sensor	temperatura	bateria	data	ativo
1044042	4	283a5975d013cb3	7.5	3.83	2023-06-28 11:11:02	S
1044041	3	28a4075d013c77	7.3125	3.81	2023-06-28 11:10:58	N
1044040	4	284fdd81e3513c26	-19.9375	3.936	2023-06-28 11:09:52	S
1044039	2	289e9d9b59201f8	3.5	3.866	2023-06-28 11:09:51	N
1044038	3	28b6cb574ec3c1c	-20.4375	3.8	2023-06-28 11:08:37	S
1044037	2	28e7f975d013ce	7.625	3.578	2023-06-28 11:08:34	S
1044036	4	28aee075d013c62	6.875	3.852	2023-06-28 11:08:10	N
1044035	1	2829667ed001e	-18.1875	4.026	2023-06-28 11:08:09	S
1044034	2	28778b75d013c51	8.125	3.97	2023-06-28 11:08:09	S
1044033	3	288bee75d013c9d	7.8125	3.734	2023-06-28 11:07:19	S
1044032	3	28b6cb574ec3c1c	-20.4375	3.804	2023-06-28 11:03:35	S
1044031	2	28e7f975d013ce	7.5625	3.578	2023-06-28 11:03:32	S
1044030	4	28e3e75d013c25	7.1875	3.802	2023-06-28 11:03:26	S
1044029	3	28d23c75d013cf8	3.625	3.876	2023-06-28 11:03:26	N
1044028	1	283f4c75d013cf6	8.4375	3.842	2023-06-28 11:03:23	S
1044027	4	283a5975d013cb3	7.375	3.83	2023-06-28 11:01:01	S

Figura 45 - Query Select à Tabela Registos

6.3 Envio de Email

