U.C. de Projeto Integrado de Telecomunicações

Ano Letivo: **2021/2022**

Relatório da Fase A

Grupo 2

- Catarina Neves, a93088
- Eduardo Cardoso, a89627
- José Gomes, a93083
- Luís Oliveira, a89380

23/03/2022

Universidade do Minho
Mestrado Integrado em Engenharia de Telecomunicações e Informática

Índice

Índice de figuras

Lista de siglas e acrónimos

API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
IDE	*Integrated Development Environment*
MP3	MPEG *Layer* 3
SI	Sistema Internacional
UUID	*Universally Unique Identifier*
OLED	*Organic light-emitting diode*
JSON	*JavaScript Object Notation*
NTP	Network Time Protocol

1. Introdução

Serve o presente relatório como síntese do trabalho desenvolvido e implementado no decorrer desta primeira fase. Este contém a descrição das estratégias e algoritmos adotados pelo grupo, tal como os respetivos testes realizados. Além dos tópicos acima citados, faz-se referência a trabalhos ou projetos similares, que se enquadram na ótica deste projeto.

O relatório abordará cada etapa desta fase de forma detalhada, ou seja, será apresentada a resposta ou proposta de solução empregue pelo grupo em junção com as ferramentas que foram necessárias para a sua construção.

O diagrama seguinte ilustra o sumário das diferentes tarefas propostas nesta fase:

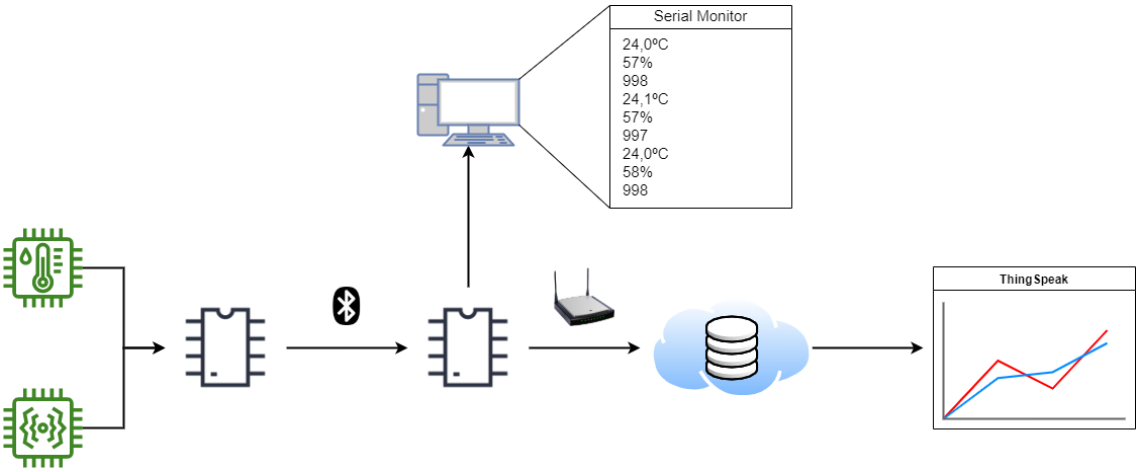

Figura 1 - Tarefas propostas pela fase A.

Os dois sensores (DHT11 e BME280) são ligados a um módulo *Arduino* (placa ESP32 DevKit v1), que servirá como server da conexão BLE (*Bluetooth Low Energy)*. Este estabelece a dita conexão com um outro módulo que, posteriormente, enviará os dados via *Wi-Fi* para um servidor, que tratará dos dados.

2. Trabalho Relacionado

Relativamente aos trabalhos relacionados, enquadram-se os projetos seguintes, que serviram de base à construção deste projeto.

2.1 ESP32 BLE Server and Client

Este projeto, da autoria de Rui Santos [1], consiste num sistema de monitorização de estações meteorológicas, que assenta no modelo Cliente-Servidor.

O Sistema Sensor/servidor é conectado ao sensor BME280 [2], que lida com os sinais seguintes: temperatura ambiente e humidade relativa do ar. Os sinais referidos são atualizados a cada 30 segundos.

O cliente é conectado ao servidor BLE, que é notificado sempre que o Sistema Sensor adquire um novo conjunto de valores (temperatura e humidade). Além das propriedades referidas, a placa ESP32 [3] (Cliente) é ligada a um OLED (*Organic light-emitting diode)*, que imprime a amostra de valores recebida.

A figura seguinte ilustra o circuito implementado pelo respetivo autor.

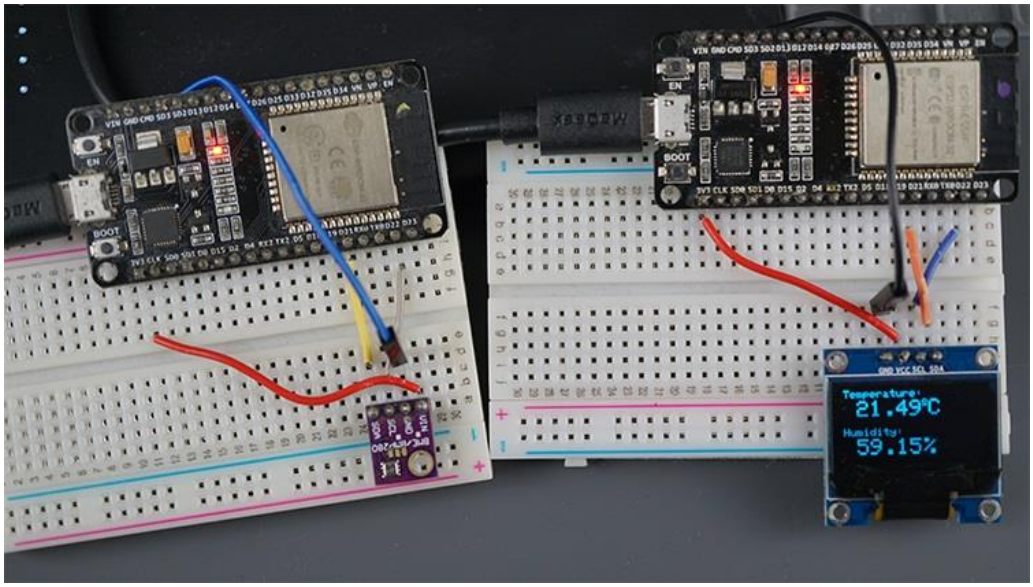

Figura 2 - Circuito implementado pelo autor.

Para testar o Sistema Sensor e a fiabilidade da comunicação BLE com o respetivo cliente, o autor recorreu ao terminal *Serial Monitor* e a uma aplicação para *smartphone,* isto é, um BLE *scanner*.

O terminal citado imprimiu as várias amostras recolhidas e mensagens de diagnóstico da ligação.

2.2 ESP32 BLE – Connecting to Fitness Band to Trigger a Bulb

Este projeto, da autoria de Aswinth Raj, é baseado na seguinte lógica: qualquer pessoa que carregue ou esteja a usar uma pulseira *fitness* e que passe relativamente perto de uma placa ESP32, caso a gama de frequências da pulseira se enquadre no espetro da placa, esta deteta-a e por consequência acende uma luz.

Este sistema é baseado no modelo Cliente-Servidor: a pulseira *fitness* desempenha o papel de servidor e a placa referida atua como cliente.

A figura seguinte ilustra o esquema eletrónico planeado pelo autor.

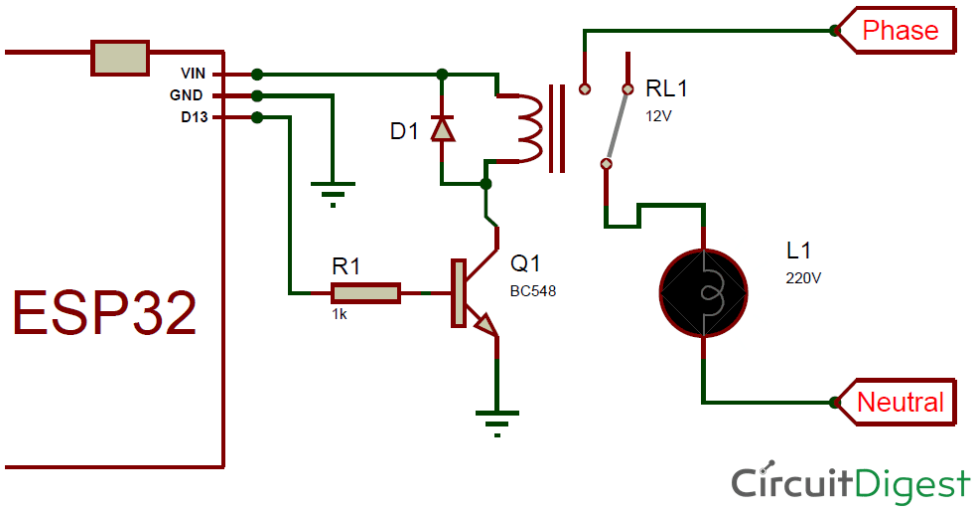

Figura 3 - Esquema eletrónico desenhado pelo autor.

O *hardware* utilizado é simples, pelo que, grande parte da "magia" ocorre dentro do código. A placa ESP32 acende ou desliga a lâmpada, respetivamente, quando o sinal *Bluetooth* é detetado ou perdido.

Os estados da lâmpada (aceso ou desligado) são coordenados através de um relé. O relé, utilizado pelo autor, opera com uma tensão de 5 volts.

Tal como o projeto referido anteriormente, para obter o serviço e o respetivo UUID do servidor, o autor recorreu a uma aplicação para *smartphone,* nomeadamente, um BL*E scanner*. Os parâmetros recolhidos são necessários à implementação dos códigos servidor e cliente.

Através do esquema eletrónico, o autor construiu o circuito seguinte, que é a versão final do sistema.

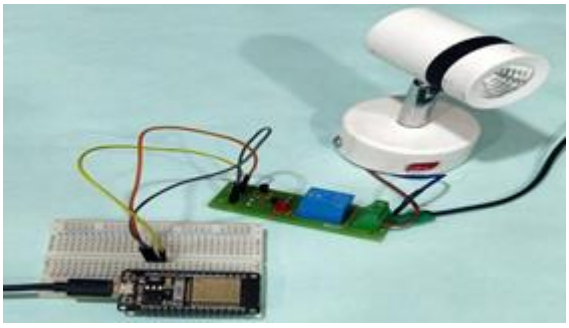

Figura 4 - Circuito implementado pelo autor.

3. Etapas do trabalho desenvolvido

Esta secção contém todas as etapas necessárias ao desenvolvimento desta fase, que se encontram descritas pelos mecanismos, ferramentas utilizadas e algoritmos implementados pelo grupo.

3.0. Montagem Eletrónica

A figura seguinte ilustra a montagem eletrónica efetuada pelo grupo. As ligações entre os componentes foram efetuadas com apoio aos *datasheets* [3] [4] [2] *correspondentes*. Como demonstrado abaixo, é possível identificar os componentes eletrónicos utilizados: placa ESP32 DevKit v1 e os sensores: DHT11 e BME280.

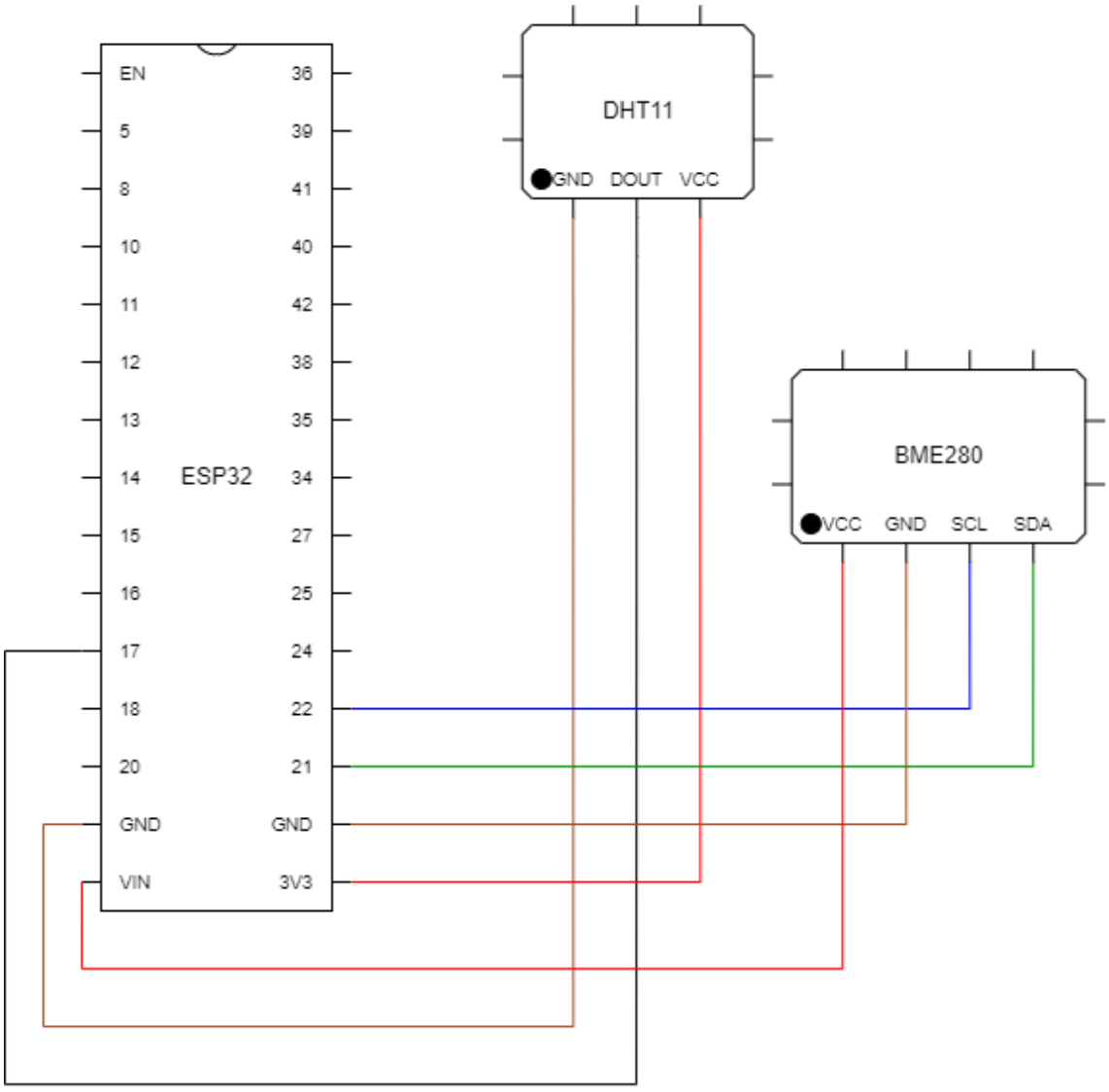

Figura 5 - Diagrama eletrónico dos componentes.

3.1. Aquisição das amostras dos sensores para o Sistema Sensor

A figura seguinte ilustra os diferentes dados meteorológicos utilizados, em junção com os sensores, responsáveis pelo seu processamento.

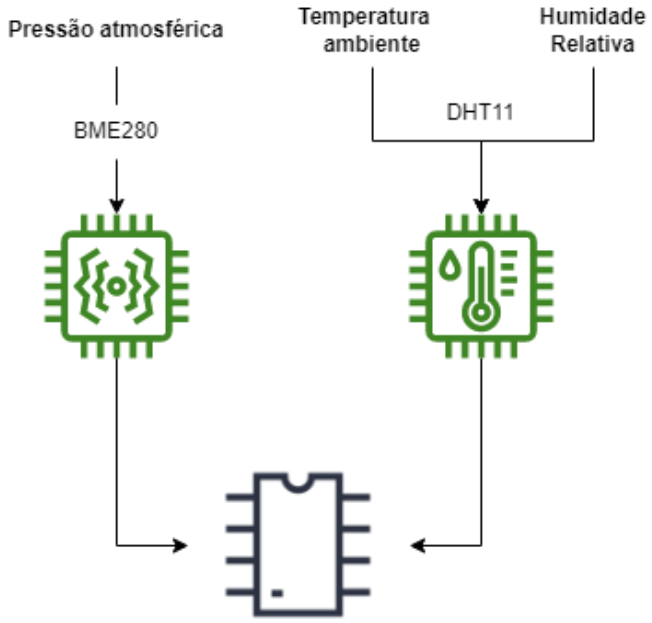

Figura 6 - Sistema Sensor: sensores e variáveis associadas.

O objetivo desta etapa é a recolha de dados através dos sensores e a respetiva transferência para o Sistema Sensor, como é representado na figura 6.

Os dados acima apresentados foram obtidas com recurso aos componentes eletrónicos (DHT11 e BME280) dentro do código do Sistema Sensor, através da importação de um conjunto de bibliotecas, que operando com os sensores referidos, é possível recolher as várias amostras de valores necessários.

A tabela seguinte ilustra todas as bibliotecas necessárias à implementação do Sistema Sensor.

Tabela 1 - Bibliotecas importadas pelo Sistema Sensor.

Sistema Sensor	
Bluetooth Low Energy	Componentes: DHT11 e BME280
<BLEDevice.h>	<DHT.h>
<BLEServer.h>	<DHT_U.h>
<BLEUtils.h>	<Adafruit_Sensor.h>
<BLE2902.h>	<Adafruit_BME280.h>

A aquisição dos dados meteorológicos é realizada dentro de uma determinada função; o algoritmo implementado pela mesma encontra-se traduzido de acordo com o fluxograma seguinte:

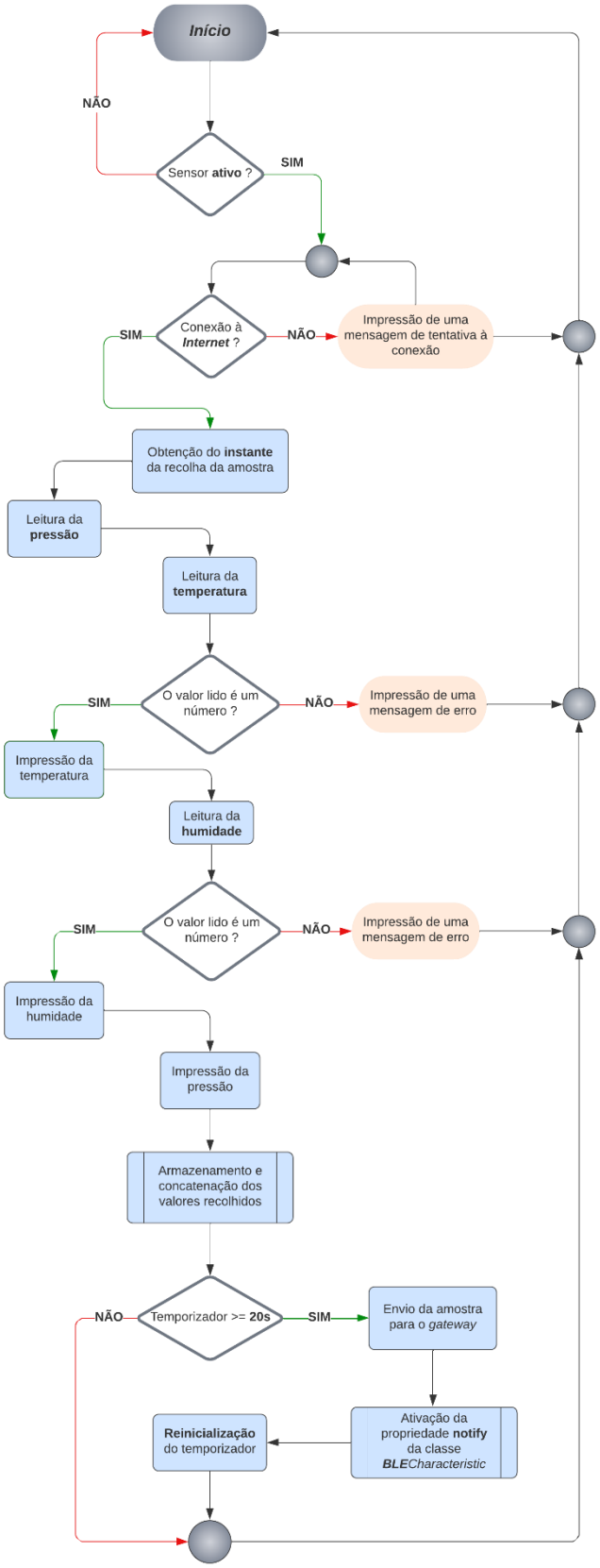

Figura 7 – Fluxograma da função responsável pela aquisição dos dados meteorológicos.

3.2. Conversão e processamento dos valores obtidos nos sensores

O diagrama seguinte representa a sequência de passos necessários ao processamento e conversão dos valores recolhidos nos sensores.

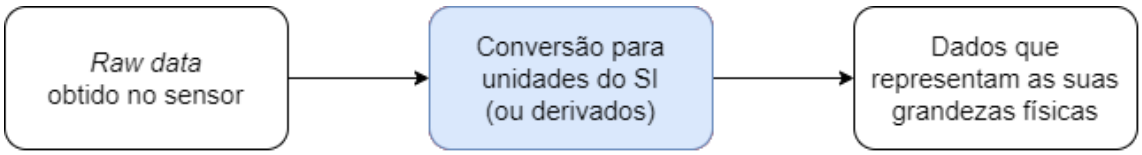

Figura 8 - Processamento dos dados obtidos na etapa anterior.

A variável *event* é responsável pela conversão dos valores lidos para a unidade do SI (Sistema Internacional) associada, ou seja, equivale ao bloco azul representado na figura 8. É de realçar que a variável referida faz a conversão de forma automática.

Para aceder à temperatura ou humidade obtida, é necessário evocar o parâmetro pretendido, que é intrínseco a esta variável (circunferências a vermelho).

```
  Serial.print(F("Temperature: "));
  Serial.print(event.temperature);

  dtostrf(event.temperature,4,1,tempStringC);
  Serial.println(F("°C"));
}
// Get humidity event and print its value.
dht.humidity().getEvent(&event);
if (isnan(event.relative_humidity)) {
  Serial.println(F("Error reading humidity!"));
}
else {
  Serial.print(F("Humidity: "));
  Serial.print(event.relative_humidity);
  Serial.println(F("%"));
```

Figura 9 - Acesso à humidade e temperatura através da variável event.

3.3. Apresentação dos dados recolhidos em tempo real

Como forma de validação e avaliação dos valores recolhidos, a cada segundo (o intervalo de impressão enunciado é de 1 segundo), é impressa uma amostra no terminal *Serial Monitor.*

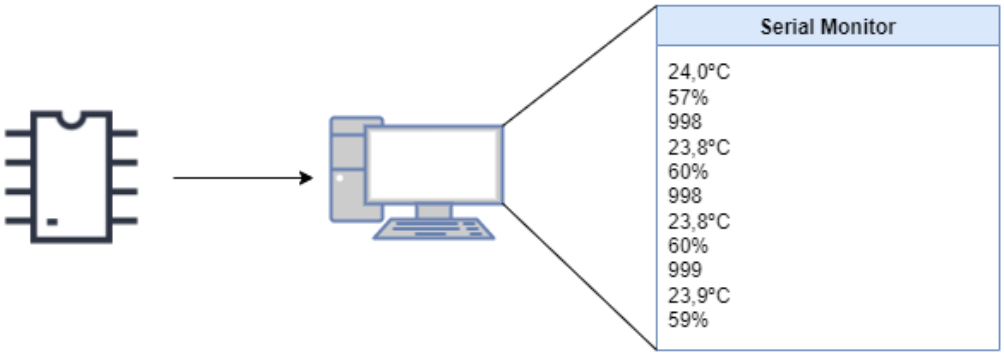

Figura 10 - Apresentação dos dados no terminal Serial Monitor do Arduino.

A apresentação dos dados recolhidos encontra-se num formato visualmente acessível, de forma local e em tempo real. Deste modo, a cada segundo (período de impressão), o Arduino irá imprimir no terminal *Serial Monitor* todos os dados meteorológicos, para que possam ser analisados, posteriormente, pelos utilizadores.

3.4. Transmissão dos dados via BLE para o Gateway

Nesta sub-etapa, é assegurada a transmissão de dados entre os dois Arduinos via BLE, sendo exemplificada pela figura seguinte.

Figura 11 - Comunicação via BLE entre Sensor e Gateway.

Através das bibliotecas referidas na tabela 1, foi possível declarar e inicializar um conjunto de objetos responsáveis pela comunicação BLE. Cada objeto atua e opera com cada propriedade/parâmetro do BLE.

Configurou-se o Sistema Sensor de forma a operar de acordo com a propriedade *notify,* que quando ativada, permite ao servidor usar a operação *Handle Value Notification* nos atributos do BLE. Também se definiu um UUID (*Universally Unique Identifier*), que é partilhado pelo Sistema Sensor e *Gateway*.

```
 BLECharacteristic::PROPERTY_NOTIFY);

#define SERVICE_UUID "1aa3d607-8465-4476-a592-40a6b0f14efb"
```

Figura 12 - Configuração das propriedades do BLE.

3.5. Envio dos dados via Wi-Fi para visualização e armazenamento online

Nesta etapa, o *Gateway* usufrui de comunicação Wi-Fi, para enviar toda informação recolhida, de forma periódica, para um servidor online *ThingSpeak* [5]. Este servidor serve de armazenamento e ferramenta de apresentação e desenho de gráficos, sendo assim possível visualizar de forma mais pormenorizada os dados obtidos pelos sensores.

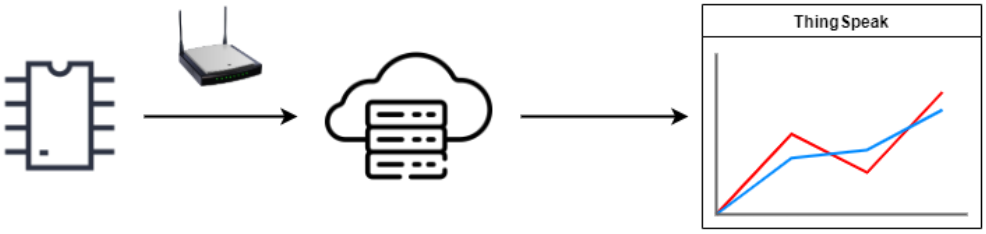

Figura 13 - Envio de dados pelo Gateway para o servidor online ThingSpeak.

Dentro do código do *Gateway*, através da importação de um conjunto de bibliotecas, apresentadas na tabela 2, é possível enviar toda a informação recolhida pelos sensores para o respetivo servidor, tal como foi citado no parágrafo anterior.

Da mesma forma que a comunicação para o Sistema Sensor, a comunicação para o Gateway possui as suas especificações por *design*: o Sistema Sensor só enviará os dados, passados 20 segundos, período este que será usado também para o envio posterior do Gateway para o *ThingSpeak*.

A tabela seguinte ilustra todas as bibliotecas necessárias à implementação do *Gateway*, onde os diferentes mecanismos empregues, encontram-se organizados de acordo com as respetivas colunas.

Tabela 2 - Bibliotecas importadas pelo Gateway.

Gateway		
Bluetooth Low Energy	Wi-Fi	*ThingSpeak*
<BLEDevice.h>	<WiFi.h>	<ThingSpeak.h>
	<WiFiAP.h>	"ThingSpeak.h"
	<WiFiClient.h>	
	<WiFiMulti.h>	
	<WiFiScan.h>	
	<WiFiServer.h>	
	<WiFiSTA.h>	
	<WiFiType.h>	
	<WiFiUdp.h>	

Relativamente ao envio dos dados obtidos, este é realizado dentro de uma determinada função; o algoritmo implementado por esta pode ser traduzido de acordo com o fluxograma seguinte:

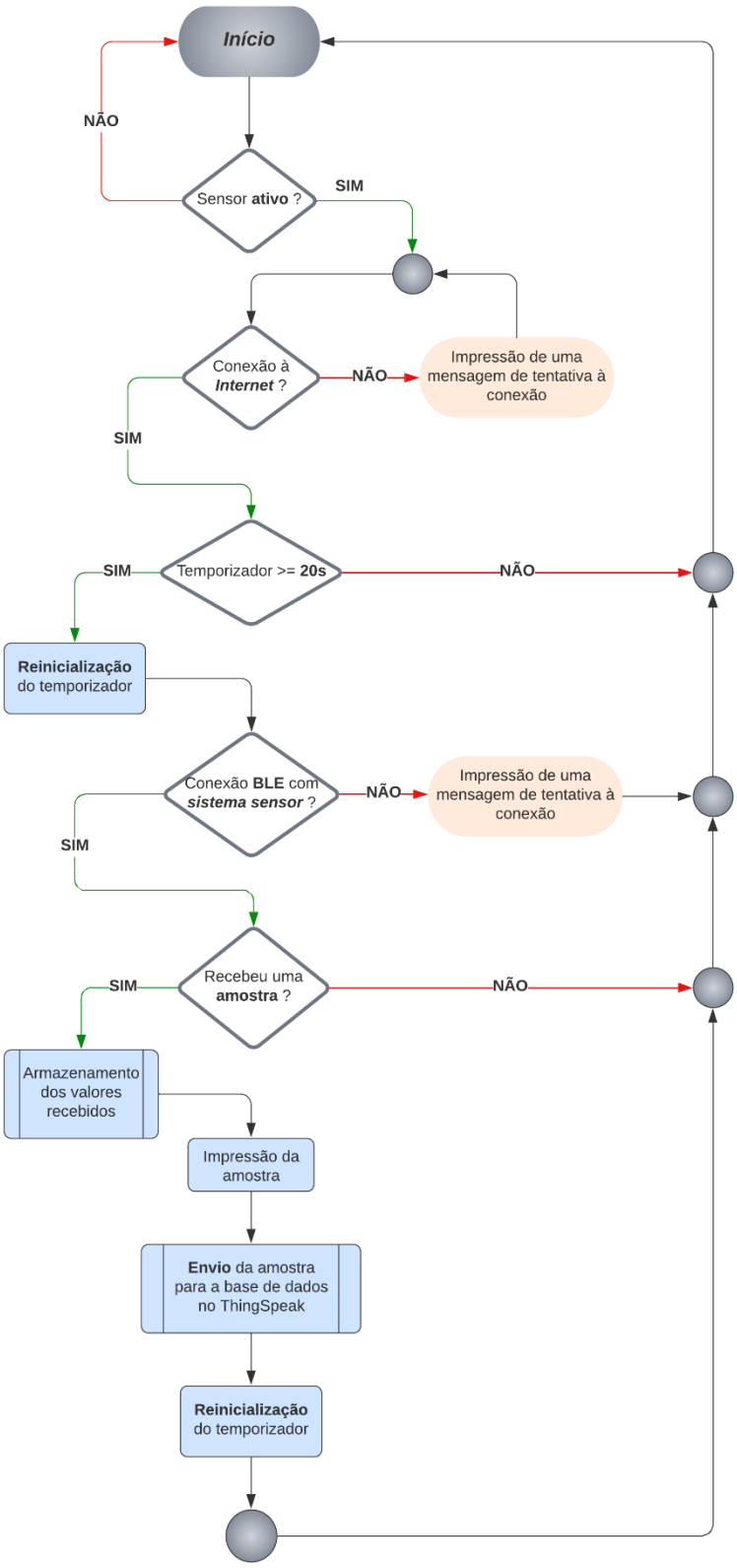

Figura 14 - Fluxograma da função responsável pelo envio de dados para o ThingSpeak.

4. Análise de resultados e testes efetuados

De forma a retificar o funcionamento e o comportamento dos protocolos implementados, colocou-se o Sistema Sensor e o *Gateway* em comunicação, com o intuito de simular a transição de dados entre estas entidades.

Os resultados obtidos encontram-se demonstrados nas figuras 15 e 16. Para cada amostra, é impresso o instante de recolha da amostra em junção com os valores das variáveis recolhidas, na grandeza física correspondente.

É de notar que os valores do *timestamp* são obtidos usufruindo de uma conexão a um servidor NTP (Network Time Protocol) [6], inicializada em ambos os sistemas dos módulos *Arduinos.*

A figura 15 ilustra a execução do Sistema Sensor; através da análise da mesma, é possível visualizar o envio de dados para o *Gateway,* a cada segundo.

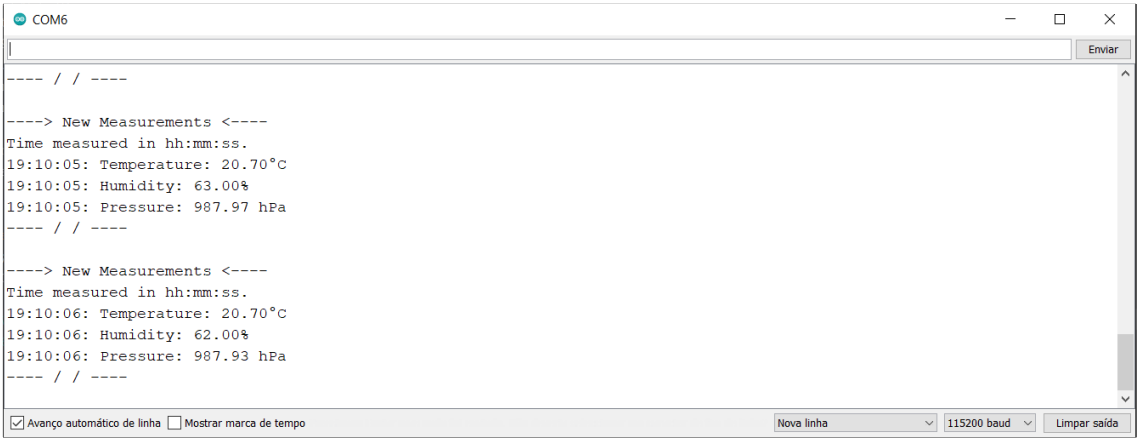

Figura 15 - Serial Monitor do Sistema Sensor.

Relativamente ao Gateway, a cada 20 segundos, recebe uma amostra, que foi enviada pelo Sistema Sensor, tal como demonstrado na figura 16.

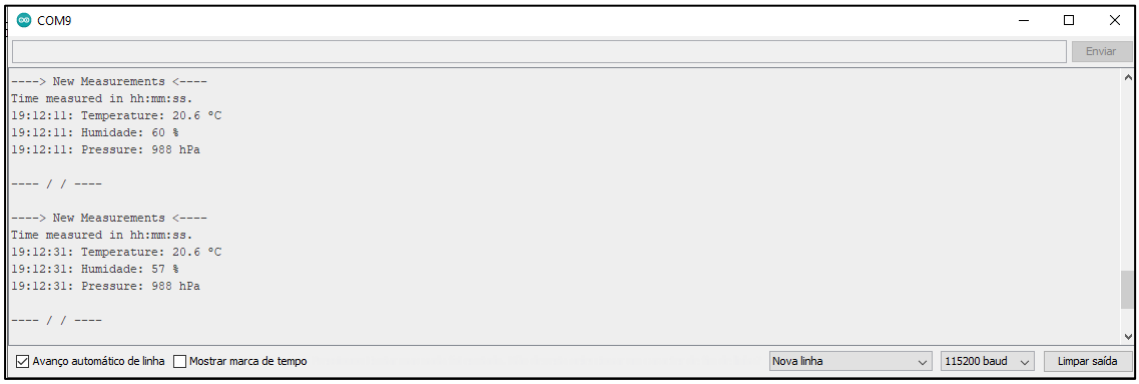

Figura 16 - Serial Monitor do Gateway.

Numa fase mais avançada, estabelecida a ligação com o *ThingSpeak,* as amostras recebidas pelo *Gateway* são redirecionadas para a base de dados *online* (alocada no servidor referido).

As figuras 17, 18 e 19 retratam a evolução temporal dos valores recolhidos, associados à variável meteorológica respetiva (temperatura, humidade e pressão), no dia 22 de março.

A figura seguinte demonstra a evolução temporal da temperatura. A escala da temperatura encontra-se disposta no eixo vertical e a escala temporal no eixo horizontal. Através da análise da figura, conclui-se que a temperatura evolui de forma linear e crescente (a partir do dia 22 de março).

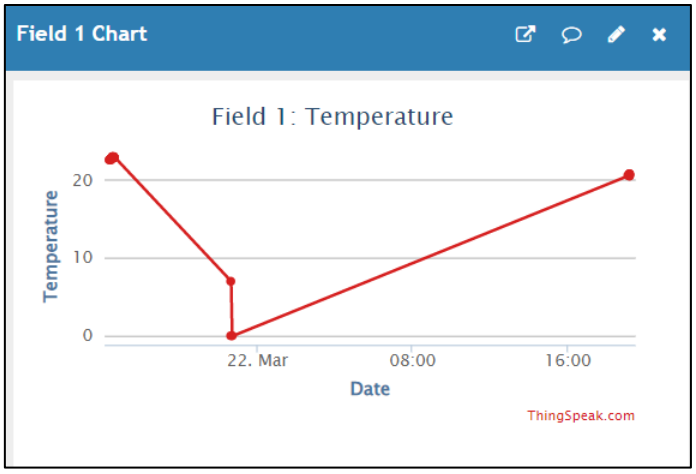

Figura 17 - Variação da temperatura.

Em contraste com a figura 17, a figura 18 ilustra a evolução temporal da humidade. A escala da humidade encontra-se disposta no eixo vertical e a escala temporal no eixo horizontal. Através da análise da figura, conclui-se que a humidade evolui de forma, aproximadamente linear, contudo, de forma decrescente.

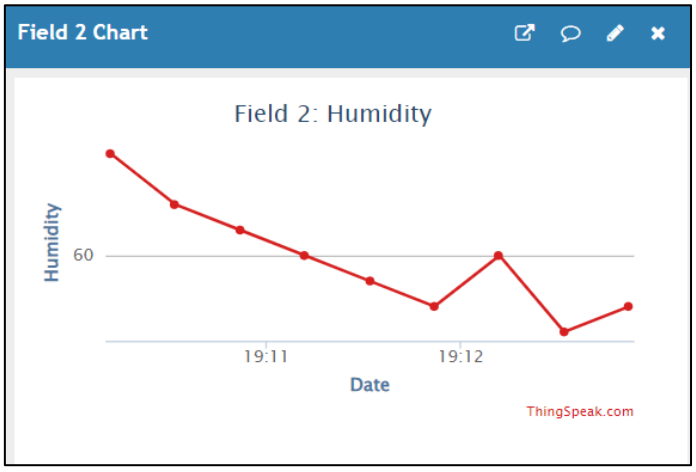

Figura 18 - Variação da humidade.

A figura 19 apresenta a evolução temporal da pressão atmosférica. A escala da pressão atmosférica encontra-se disposta no eixo vertical e a escala temporal no eixo horizontal. Através da análise da figura, tal como é evidenciado no gráfico da humidade, a pressão também evolui de forma decrescente.

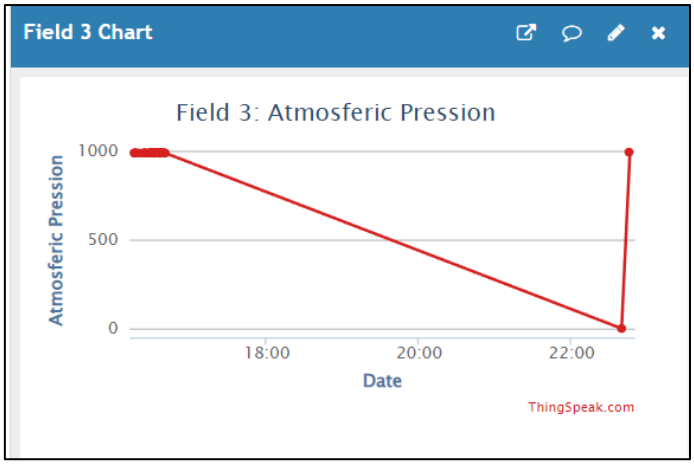

Figura 19 - Variação da pressão atmosférica.

Para efetuar uma análise mais detalhada da amostra recolhida, recorreu-se ao formato JSON (*JavaScript Object Notation*), que é responsável pela organização e indentação do conteúdo da amostra.

A figura 20 mostra os campos e detalhes da base de dados utilizada pelo grupo.


```
"channel": {
  "id": 1666665,
  "name": "Projeto Integrado de Telecomunicações",
  "description": "Sistema de Monitorização de Estações Meteorológicas",
  "latitude": "0.0",
  "longitude": "0.0",
  "field1": "Temperature",
  "field2": "Humidity",
  "field3": "Atmosferic Pression",
  "created_at": "2022-03-03T11:19:13Z",
  "updated_at": "2022-03-14T16:20:47Z",
  "last_entry_id": 108
```


Figura 20 – Campos e detalhes da base de dados.

A figura 21 apresenta os detalhes de uma amostra registada na base de dados. É possível visualizar o registo temporal em junção com os valores de temperatura, humidade e pressão recolhidos.


```
{
  "created_at": "2022-03-21T16:36:09Z",
  "entry_id": 85,
  "field1": "23.0",
  "field2": "41",
  "field3": "988"
},
```


Figura 21 – Detalhes e composição de uma determinada amostra.

5. Conclusão

Concluída esta fase, na opinião do grupo, conseguiu-se atingir todos os objetivos propostos. Esta fase contribuiu para o conhecimento e estudo das tecnologias *Bluetooth Low Energy* e Wi-Fi, tal como a interação com sensores e as suas respetivas bibliotecas.

O grupo pretende, através uma boa e forte base, seja possível atingir as metas definidas para as próximas etapas do projeto enunciado.

5.1. Contribuição de cada aluno

Catarina Neves

- Conceção dos algoritmos;
- Tratamento de dados, tabelas e imagens a estes relacionados;
- Desenho de esquemas e gráficos;
- Recolha e implementação de *timestamps;*
- Desenvolvimento do relatório;

Eduardo Cardoso

- Conceção dos algoritmos;
- Implementação e desenvolvimento dos algoritmos de comunicação BLE;
- Implementação da comunicação Wi-Fi;
- Implementação da comunicação entre *Gateway* e *Thingspeak;*
- Recolha e implementação de *timestamps;*
- Desenvolvimento do relatório;

José Gomes

- Conceção dos algoritmos;
- Tradução dos algoritmos implementados em fluxogramas;
- Implementação dos algoritmos de comunicação BLE;
- Desenvolvimento do relatório;

Luís Oliveira

- Conceção dos algoritmos;
- Criação e configuração da base de dados no *ThingSpeak*;
- Implementação da comunicação entre *Gateway* e *ThingSpeak;*
- Desenho de esquemas e gráficos;
- Desenvolvimento do relatório.

6. Lista de Referências

[1] R. Santos, “Random Nerd Tutorials,” 16 5 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/.

[2] Bosch, “BME280 Combined humidity and pressure sensor,” setembro 2018. [Online]. Available: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf.

[3] Espressif Systems, “ESP32 Series Datasheet,” 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[4] Aosong, “Temperature and humidity module DHT11 Product Manual,” 14 maio 2015. [Online]. Available: https://www.waveshare.com/w/upload/c/c7/DHT11_datasheet.pdf.

[5] ioBridge, “ThingSpeak for IoT Projects,” 2022. [Online]. Available: https://thingspeak.com/.

[6] NTP Pool Project, “Europe — europe.pool.ntp.org,” [Online]. Available: https://www.pool.ntp.org/zone/europe.

[7] R. Aswinth, “Circuit Digest,” 5 10 2018. [Online]. Available: https://circuitdigest.com/microcontroller-projects/esp32-ble-client-connecting-to-fitness-band-to-trigger-light.

[8] R. Teja, “Eletronics Hub,” 24 3 2021. [Online]. Available: https://www.electronicshub.org/esp32-ble-tutorial/.

[9] afan31, “Github,” 15 12 2020. [Online]. Available: https:/github.com/mathworks/thingspeak-arduino.