

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

[1. Introdução 5](#_Toc98785418)

[2. Trabalho Relacionado 6](#_Toc98785419)

[2.1 ESP32 BLE Server and Client 6](#_Toc98785420)

[2.2 ESP32 BLE – Connecting to Fitness Band to Trigger a Bulb 7](#_Toc98785421)

[3. Etapas do trabalho desenvolvido 8](#_Toc98785422)

[3.0. Montagem Eletrónica 8](#_Toc98785423)

[3.1. Aquisição das amostras dos sensores para o sistema sensor 9](#_Toc98785424)

[3.2. Conversão e processamento dos valores obtidos nos sensores 11](#_Toc98785425)

[3.3. Apresentação dos dados recolhidos em tempo real 12](#_Toc98785426)

[3.4. Transmissão dos dados via BLE para o gateway 12](#_Toc98785427)

[3.5. Envio dos dados por Wi-Fi para visualização e armazenamento online 13](#_Toc98785428)

[4. Análise de resultados e testes efetuados 15](#_Toc98785429)

[5. Conclusão 16](#_Toc98785430)

[5.1. Contribuição de cada aluno 16](#_Toc98785431)

[6. Lista de Referências 17](#_Toc98785432)

Índice de figuras

Figura 1 - Circuito implementado pelo autor 6

Figura 2 - Esquema eletrónico desenhado pelo autor 7

Figura 3 - Circuito implementado pelo autor 7

Figura 4 - Diagrama eletrónico dos componentes 8

Figura 5 - Sistema sensor: sensores e variáveis associadas 9

Figura 6 - Bibliotecas importadas pelo Sistema Sensor 9

Figura 7 – Algoritmo da função é responsável pela aquisição dos dados meteorológicos 10

Figura 8 - Processamento dos dados obtidos na etapa anterior 11

Figura 9 - Acesso à humidade e temperatura através da variável event 11

Figura 10 - Apresentação dos dados no terminal Serial Monitor do Arduino 12

Figura 11 - Comunicação via BLE entre Sensor e Gateway 12

Figura 12 - Configuração das propriedades do BLE 12

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

Figura 14 - Bibliotecas relativas ao gateway 13

Figura 15 - Algoritmo da função responsável pelo envio de dados para o ThingSpeak 14

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*

# Introdução

Serve o presente relatório como síntese do trabalho desenvolvido e implementado no decorrer desta fase. Este contém a descrição das estratégias e algoritmos adotados pelo grupo como também os 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 à sua construção.

# 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, consiste num sistema de monitorização de estações meteorológicas, que assenta no modelo Cliente-Servidor.

O sistema sensor/servidor é conectado ao sensor BME280, que lida com os seguintes sinais: temperatura ambiente e humidade relativa do ar. Os sinais referidos são atualizados a cada trinta segundos.

O cliente é conectado ao servidor BLE (*Bluetooth Low Energy)*, que é notificado sempre que o sistema sensor adquire um novo conjunto de valores (temperatura e humidade). Além das propriedades referidas, a placa ESP32 (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.

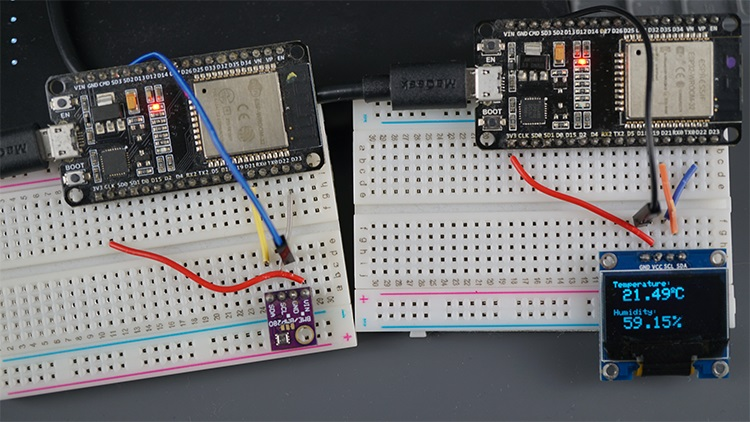


Figura 1 - Circuito implementado pelo autor

Para testar o sistema sensor e a fiabilidade da respetiva 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, como também, 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.

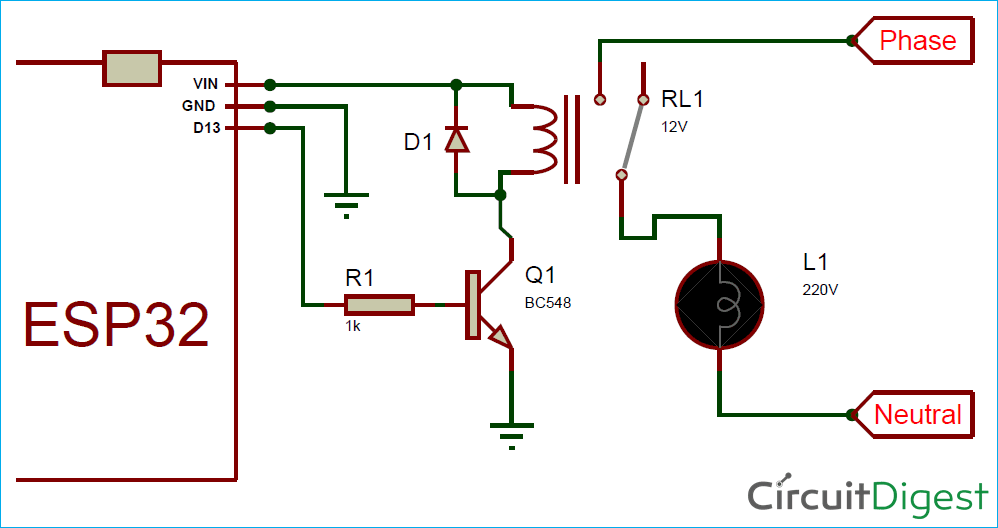
**

Figura 2 - 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 cinco 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.

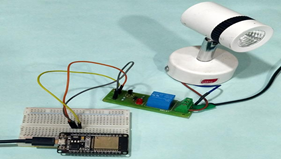


Figura 3 - Circuito implementado pelo autor

# 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* [1][2][3]correspondentes.

Através da sua análise, é possível identificar os componentes eletrónicos utilizados: placa ESP32 e os sensores: DHT11 e BME280.

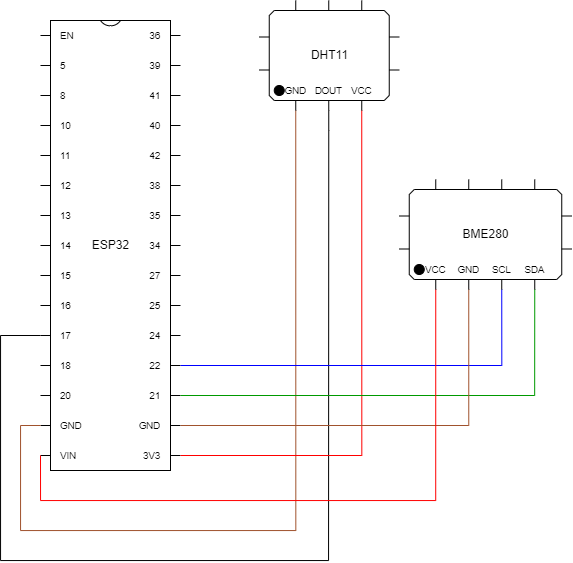


Figura 4 - Diagrama eletrónico dos componentes

## 3.1. Aquisição das amostras dos sensores para o sistema sensor

A figura seguinte ilustra as diferentes variáveis meteorológicas utilizadas, em junção com os sensores, responsáveis pelo seu processamento.

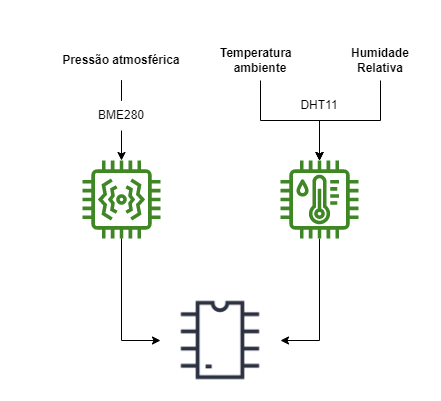


Figura 5 - Sistema sensor: sensores e variáveis associadas

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 5.

As variáveis acima apresentadas 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 operam com os sensores descritos, é possível recolher várias amostras de valores associados a essas variáveis.

A tabela seguinte ilustra todas as bibliotecas necessárias à implementação do 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> |

Figura 6 - Bibliotecas importadas pelo Sistema Sensor

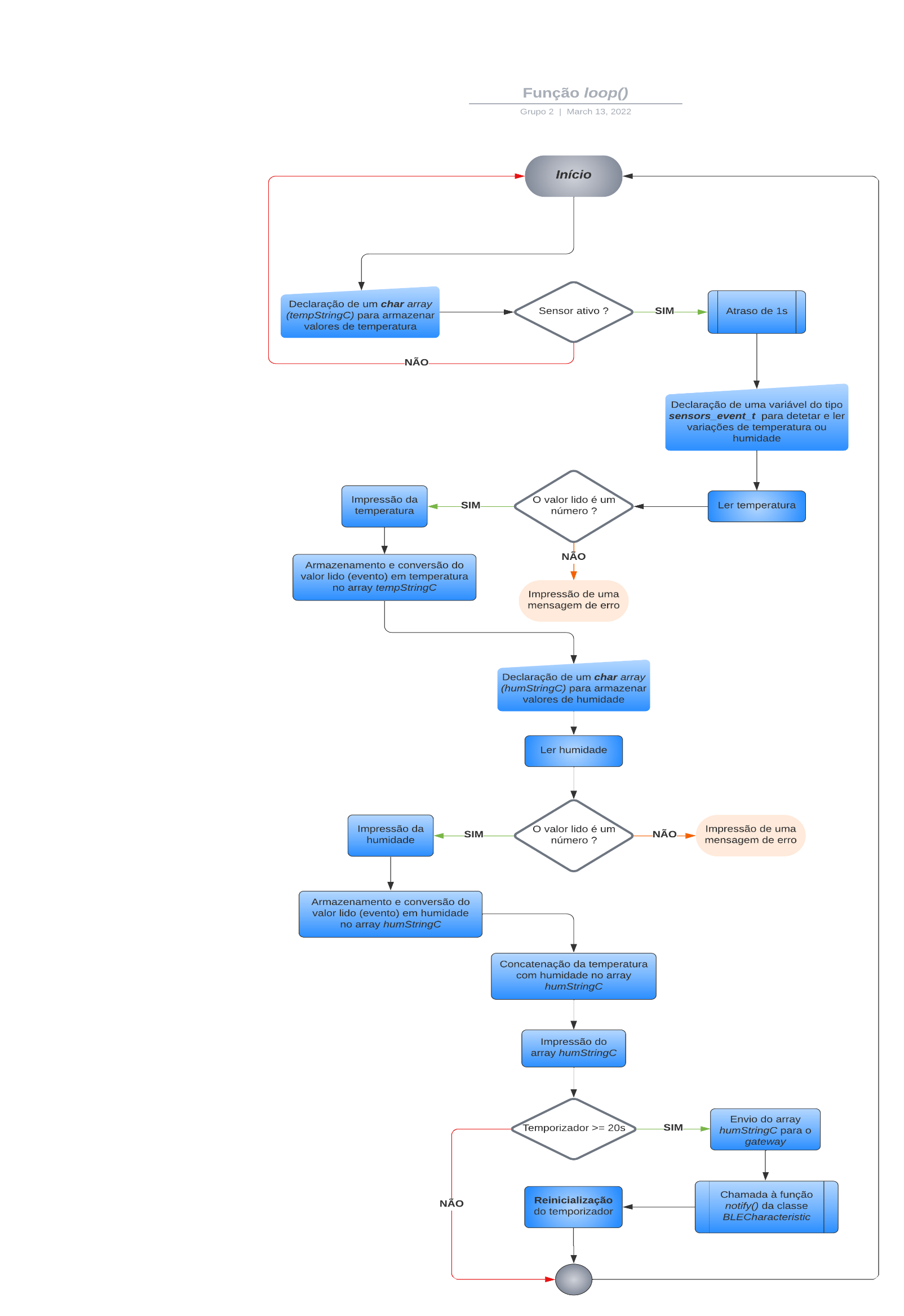
A aquisição dos dados meteorológicos é realizada dentro de uma determinada função; o algoritmo implementado pela mesma ser traduzido de acordo com o fluxograma seguinte:

Figura 7 – Algoritmo da função é 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.

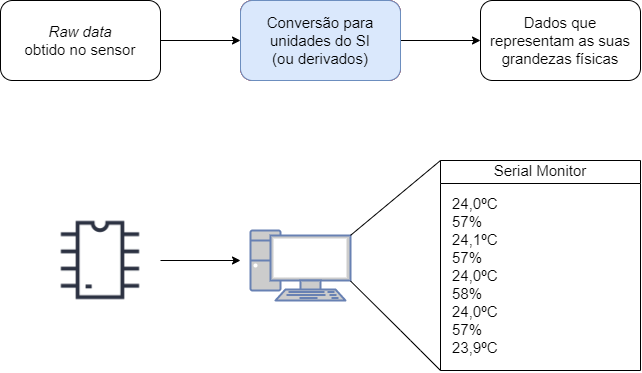


Figura 8 - Processamento dos dados obtidos na etapa anterior

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).

Uma imagem com texto

Descrição gerada automaticamente

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, sempre que o sensor obtém uma nova amostra, esta é impressa no terminal *serial monitor.* O intervalo de tempo entre as diversas impressões, dep [1]ende do período de amostragem, que pode ser ajustado, sendo de momento, *hardcoded*.

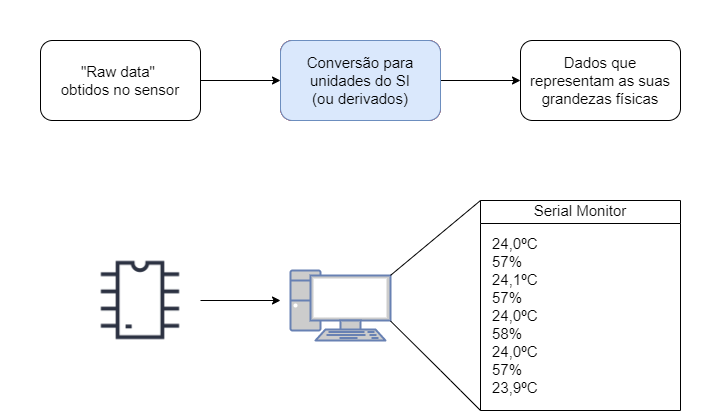


Figura 10 - Apresentação dos dados no terminal Serial Monitor do Arduino

A apresentação dos dados recolhidos encontra-se num formato visualmente acessível, de forma local e em tempo real. Deste modo, a cada um segundo (período de amostragem) 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 (*Bluetooth Low Energy*), sendo exemplificada pela figura seguinte.

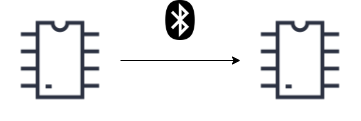


Figura 11 - Comunicação via BLE entre Sensor e Gateway

Através das bibliotecas referidas na figura dois, 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* e definiu-se um UUID (*Universally Unique Identifier*), que é partilhado pelo sistema sensor e *gateway*.

Uma imagem com texto

Descrição gerada automaticamente



Figura 12 - Configuração das propriedades do BLE

## 3.5. Envio dos dados por 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*). Este servidor serve de armazenamento e ferramenta de apresentação e desenho de gráficos, assim, é possível visualizar de forma mais pormenorizada os dados obtidos pelos sensores.

Uma imagem com texto

Descrição gerada automaticamente

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, que operam com a comunicação Wi-Fi, BLE e com o servidor *online* é possível enviar toda a informação recolhida pelos sensores para o respetivo servidor, tal como foi citado no parágrafo anterior.

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.

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

Figura 14 - Bibliotecas relativas ao gateway

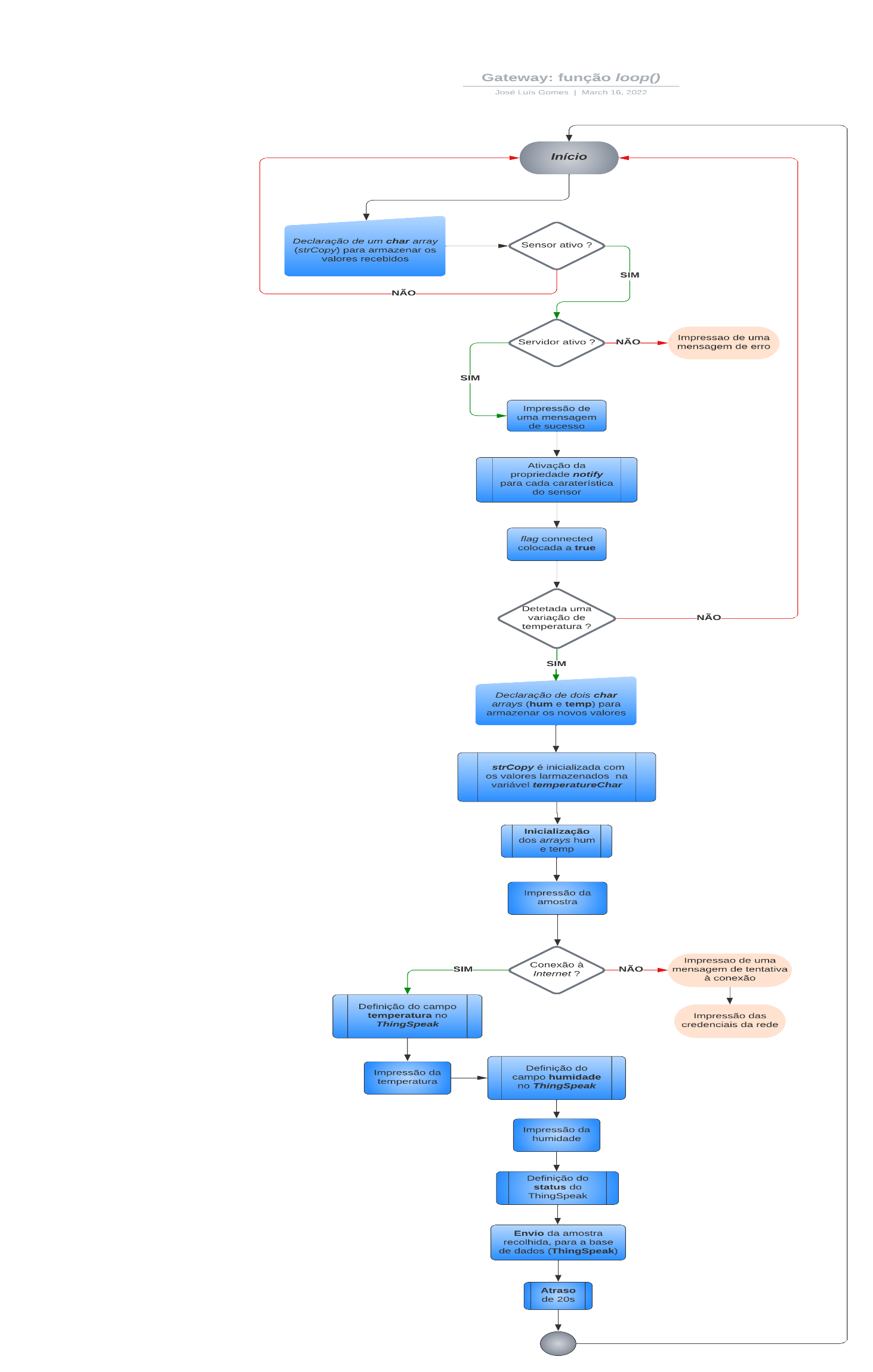
Relativamente ao envio dos dados obtidos, este é realizado dentro de uma determinada função; o algoritmo implementado pela mesma pode ser traduzido de acordo com o fluxograma seguinte:

Figura 15 - Algoritmo da função responsável pelo envio de dados para o ThingSpeak

# Análise de resultados e testes efetuados

# Conclusão

Concluída esta exposição, na opinião do grupo, esperamos desta fase um nível de complexidade moderada, especialmente na fase de *design* inicial, visto que queremos, desde o início, eficiência e modularidade para que o resto do projeto seja mais simples.

Assim sendo, acreditamos que o planeamento apresentado irá resultar numa aplicação eficaz do hardware e do software, para que demonstremos adequadamente conhecimentos a adquirir no decorrer da unidade curricular.

## 5.1. Contribuição de cada aluno

Catarina Neves

* A
* B
* C

Eduardo Cardoso

* A
* B
* C

José Gomes

* Conceção do algoritmo
* Tradução dos algoritmos implementados em fluxogramas
* Desenvolvimento do relatório

Luís Oliveira

* Desenho e comunicação com o Thingspeak
* Desenho de esquemas e gráficos
* Desenvolvimento do relatório

# Lista de Referências

|  |  |
| --- | --- |
| [1] | Espressif Systems, “ESP32 Series Datasheet,” 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf. |
| [2] | Bosch, “BMP280 Digital Pressure Sensor,” 8 janeiro 2018. [Online]. Available: https://www.digikey.pt/en/datasheets/bosch-sensortec/bosch-sensortec-bst-bmp280-ds001-19. |
| [3] | Aosong, “Temperature and humidity module DHT11 Product Manual,” 14 maio 2015. [Online]. Available: https://www.waveshare.com/w/upload/c/c7/DHT11\_datasheet.pdf. |
| [4] | R. Santos, “Random Nerd Tutorials,” 16 5 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/. |
| [5] | R. Aswinth, “Circuit Digest,” 5 10 2018. [Online]. Available: https://circuitdigest.com/microcontroller-projects/esp32-ble-client-connecting-to-fitness-band-to-trigger-light. |
| [6] | R. Teja, “Eletronics Hub,” 24 3 2021. [Online]. Available: https://www.electronicshub.org/esp32-ble-tutorial/. |
| [7] | afan31, “Github,” 15 12 2020. [Online]. Available: https:/github.com/mathworks/thingspeak-arduino. |
| [8] | Aosong, “Temperature and humidity module,” 14 maio 2015. [Online]. Available: https://www.waveshare.com/w/upload/c/c7/DHT11\_datasheet.pdf. |