Instituto Politécnico de Leiria - Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Eletrotécnica e de Computadores

SISTEMA INTELIGENTE DE MONITORIZAÇÃO DE UMA HABITAÇÃO

Unidade Curricular:	Sistema para Internet das coisas – 3ºAno, 2º Semestre
Alunos:	João Oliveira Parracho, Nº: 2160780
	Ricardo Anastácio, Nº: 2162144
Docente:	Nuno Miranda

Leiria, 4 de julho de 2019

Índice de figuras

Índice de tabelas

Não foi encontrada nenhuma entrada do índice de ilustrações.

João Parracho
Ricardo Anastácio

Introdução

No âmbito da unidade curricular de Sistemas Eletrónicos para a Internet das coisas foi proposto a realização de um sistema de monotorização para uma habitação

O presente relatório tem como objetivo descrever as tecnologias de comunicação utilizadas, estudar os sensores e fazer uma descrição técnica de todos os subsistemas implementados e desenvolvidos.

No final é realizado uma visão critica sobre o sistema desenvolvido.

1. Internet das Coisas

1.1. Evolução da *Internet das Coisas*

O objetivo de ter os sistemas ligados em rede é promover as operações remotas, ou seja, qualquer sistema permite a sua monitorização remotamente a qualquer distância via uma aplicação ou qualquer outra plataforma digital.

Atualmente surgem cada vez mais conceitos que visam apoiar a revolução da indústria 4.0 , como por exemplo o IoT (Internet das Coisas) [1] , a interoperabilidade entre máquinas M2M (Máquina para Máquina) [2] e o desenvolvimento de IA (Inteligência Artificial) [3].

De forma a cumprir os objetivos referidos anteriormente é necessário a recolha do máximo de informação possível em tempo real e que estes possam ser processados. Devido á exigência do mercado e à necessidade de recolha de informação de forma eficiente é fundamental desenvolver sistemas IoT de monitorização de alta fidelidade

2. Material utilizado

- 2xLDR
- 2xEsp8266
- 1xRaspeberry Pi
- 1xArduino Mega 2560
- 1xArduíno uno
- 1xPic18F25k50
- 1xMotor
- 1xVentoinha
- 1xAltimetro *MPL3115A2*
- 1x*ADXL345 Digital Adafruit* (acelerómetro)
- 1xPIR
- 1xSonar HC-SR04
- 1xLCD ST7735R

3. Tecnologias de comunicação

Durante a execução do trabalho surgiu a necessidade de integrar todos os subsistemas, para tal foi necessário recorrer a diversos protocolos de comunicação.

3.1. Serial

Este protocolo de comunicação consiste numa transmissão bit a bit entre os dispositivos transmissor/recetor. Para tal é definida uma taxa de comunicação "Baud Rate" que especifica a quantidade de bits a transmitir por segundo. Definindo o mesmo Baud Rate em ambos os dispositivos conseguimos realizar uma comunicação assíncrona entre os dispositivos sem necessitar de um pulso relógio para sincronização. A Figura 3-1 que se segue exemplifica a transmissão do caracter 'S'.

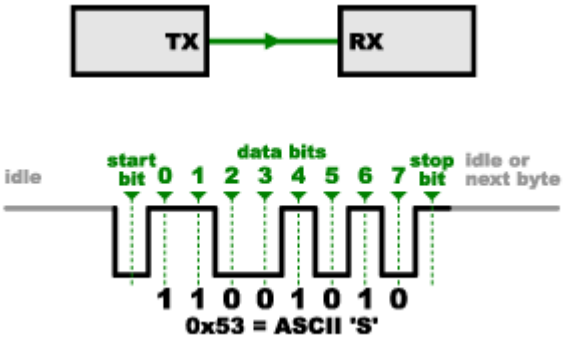

Figura 3-1 - Exemplo da transmissão do caracter 's'

3.2. *I2C*

O protocolo I2C (*Inter-Integrated Circuit*) [4] permite múltiplos dispositivos "*slaves*" comunicarem com um ou mais "*master*". Este protocolo utiliza duas linhas, uma linha (SCL) para o relógio série para poder sincronizar todos os dispositivos e outra para transmitir dados série (SDA). A linha de dados é uma linha bidirecional, desta forma os dispositivos "master" conseguem comunicar com os "*slave*" e vice-versa.

O facto das linhas serem "*open-drain*" [5] requer um pull -externo o que permite que existam vários "bus masters".

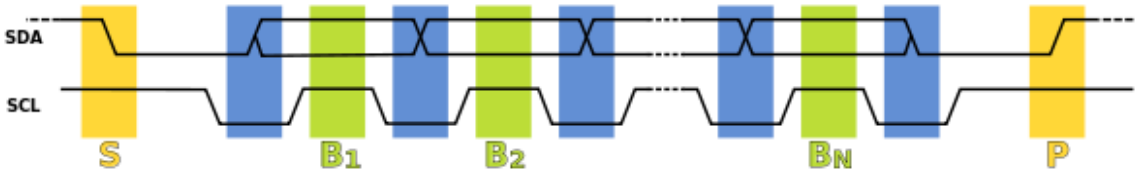

Figura 3-2- Diagrama Temporal de uma ligação I2C

Este barramento utiliza endereçamento de 7-bit ou de 10 bit e tem uma velocidade de 100 kbits/s no modo padrão.

Como podemos observar na Figura 3-2, no início de cada transação I2C é enviado inicialmente um bit "*s*", a linha de dados que esta no estado logico alto é colocada no estado logico baixo e é transmitido um endereço de 7 bits. Cada dispositivo lê o endereço e se este for coincidente com o seu, este responde. No final da transmissão de dados e enviado um bit "P" para terminar a transmissão.

Existe também um protocolo de mensagens para identificar se o "master" lê ou escreve dados para o "*slave*". A Figura 3-3 demonstra uma ligação I2C entre um dispositivo "*master*" e vários "*slaves*".

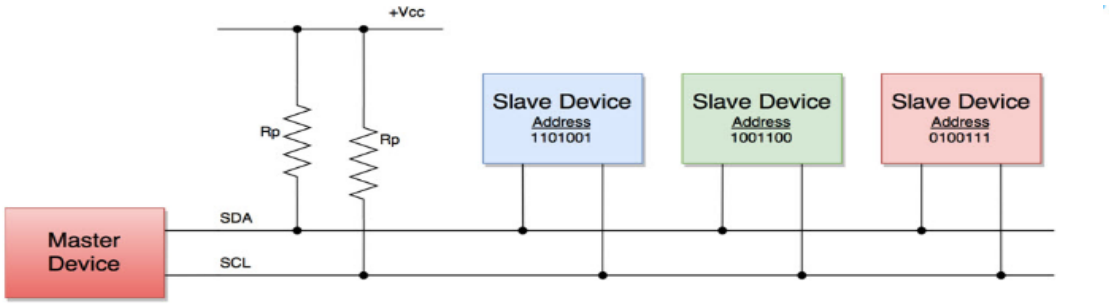

Figura 3-3-Diagram de uma ligação I2C

3.3. SPI

O método SPI (Serial Peripheral Interface) permite uma comunicação full-duplex, síncrona de elevada velocidade. Fisicamente esta interface apresenta quatro ligações o SCLK (pulso relógio de sincronização), MOSI (linha de transmissão master-slave), MISO (linha de receção slaver-master), SS (linha que define o slave com o qual se estabelece a comunicar)Figura 3-4.

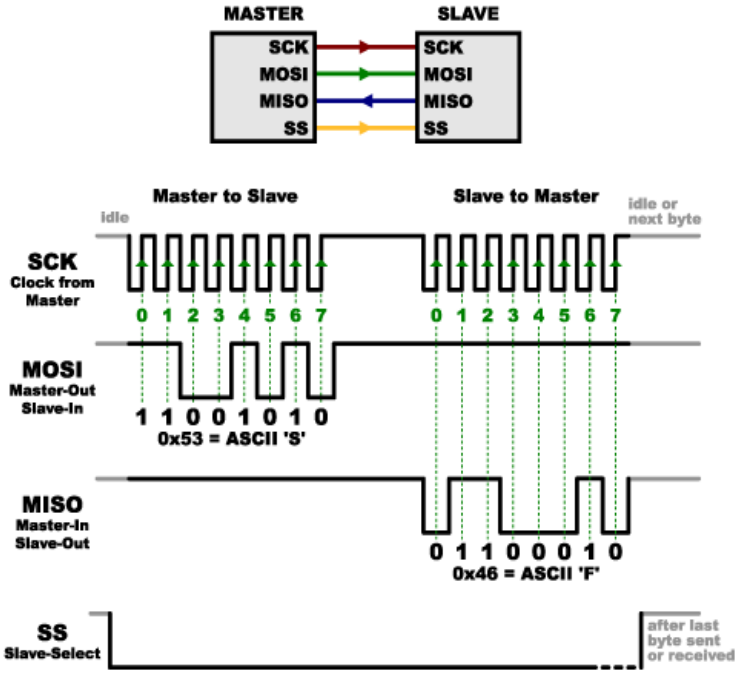

Figura 3-4 - Protocolo SPI

3.4. *MQTT*

Com o crescimento exponencial da tecnologia e da Internet das coisas surge a necessidade de um protocolo que interligue diversos dispositivos (sensores, microcontroladores) numa rede sem fios.

O MQTT (Message Queuing Telemetry Transport) [6] surge com o objetivo de criar um protocolo de comunicação simples, fácil de configurar e que permita a interligação de vários nós de sensores/controladores.

Este protocolo é baseado em publicações/subscrições e foi projetado para dispositivos de largura de banda reduzida e de alta latência o que o torna ideal para comunicações *machine-to-machine* (M2M) [7].

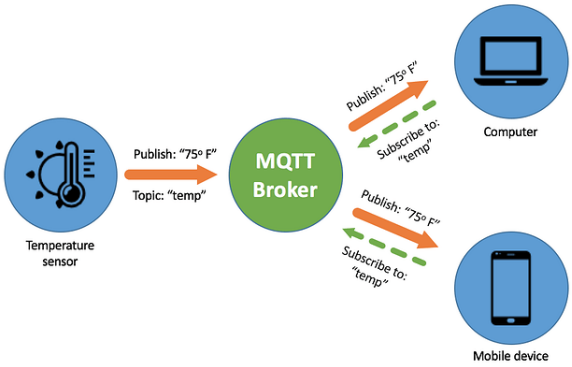

Figura 3-5 - Sistema MQTT

Um sistema MQTT é baseado no TCP/IP e constituído por um *broker*, o servidor responsável pela gestão de informação dos clientes, e por tópicos.

A Figura 3-5 mostra um sensor de temperatura e dois dispositivos ligados a um *broker MQTT*. O objetivo dos dispositivos é obter a temperatura do sensor, para tal é necessário que este publique para um determinado tópico do *broker MQTT* os valores medidos e de forma a poderem ser visualizados por todos subscritores do respetivo tópico.

Em suma como os dois dispositivos são subscritores do tópico do sensor de temperatura sempre que este publica informação, os dois dispositivos recebem a informação atualizada através do *broker MQTT*.

Uma das vantagens de utilizar o protocolo *MQTT* é que a cada conexão está associado uma qualidade de serviço (QoS), ou seja, conseguimos tornar o nosso sistema de comunicação mais eficiente e robusto uma vez que é possível fazer uma distinção da importância das mensagens enviadas/recebidas.

3.5. *Node-RED*

O Node-RED **Erro! A origem da referência não foi encontrada.** é uma plataforma Open Source de d esenvolvimento criada pela IBM que conta com um editor de fluxo baseado em JavaScript e NodeJS acessível através de um browser. Esta ferramenta tem como objetivo atuar, interligar e controlar diversos nós de uma rede de sensores de uma forma simples e intuitiva. Através do Node-RED é possível comunicar e interagir com base em diversos protocolos tais como MQTT, Serial, TCP/UDP e HTTP entre outros. É possível através de plugins desenvolvidos pela comunidade e IBM adicionar novas funcionalidades a plataforma tais como dashboards, GeoMaps, Clouds, etc.

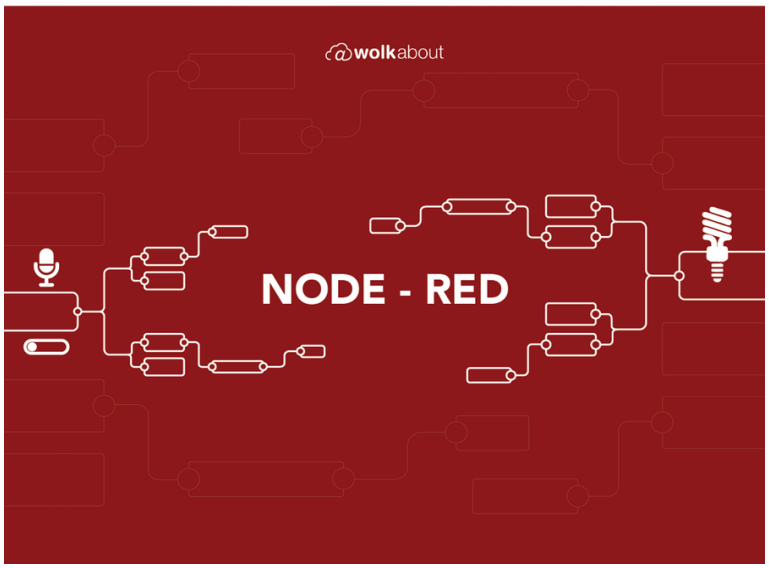

Figura 3-6 - Node-Red

4. Sensores

4.1. *PIR*

O sensor eletrónico PIR é sensor infravermelho passivo. O principal objetivo é detetar a presença movimento. Esta deteção é feita com base na emissão de enérgica térmica proveniente de objetos, pessoas e animais esta energia térmica emite radiação com comprimentos de onda no espetro do infravermelho que é posteriormente detetada pelo sensor. Desta forma é possível assinalar a presença de algum objeto termicamente radiante Figura 4-1, no entanto não temos informação sobre quem ou o que se moveu.

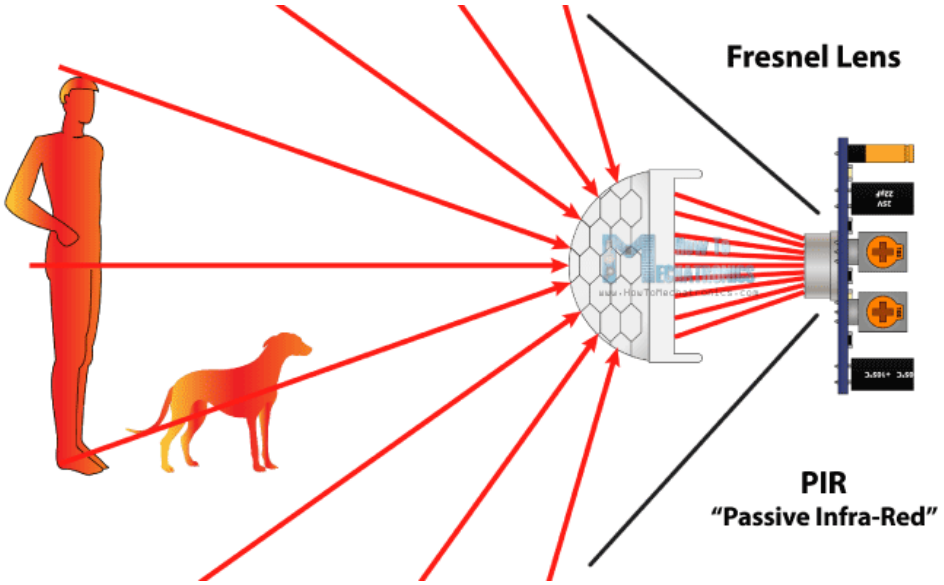

Figura 4-1 - Sensor PIR

4.2. *DHT11*

O DHT11 [8] é um sensor digital de temperatura e humidade simples, de alta fidelidade e resposta rápida.

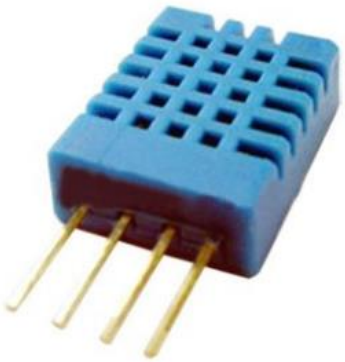

Figura 4-3-DHT11

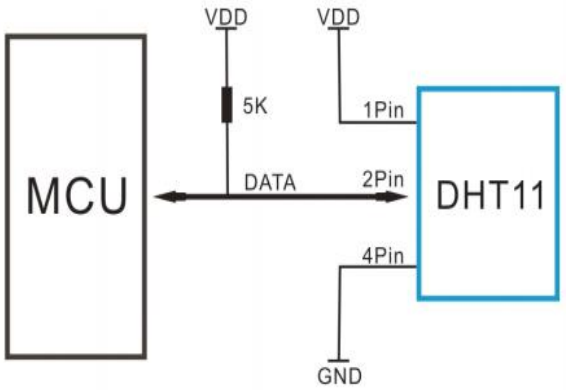

Figura 4-2-Aplicação típica DHT11

Como podemos observar na Figura 4-2 a comunicação é feita apenas por uma ligação de dados.

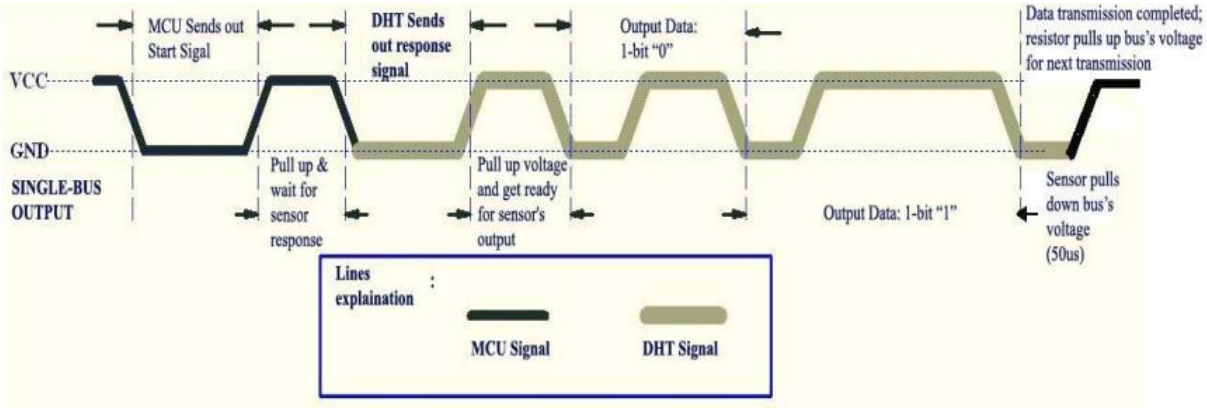

Figura 4-4 - Comunicação Com DHT11

A informação é, portanto, transmitida num single-bus de forma síncrona entre o microcontrolador (MCU) e o DHT11, onde cada processo de comunicação demora tipicamente 4ms. Numa fase inicial o MCU envia um sinal para iniciar a comunicação e aguarda que o DHT11 responda, posteriormente o DHT11 responde ao MCU a dizer que vai iniciar a transmissão de dados. São transmitidos uma trama de dados de 40 bits, 8 bits para a parte integral do valor de humidade, 8 bits para a parte decimal do valor de humidade, repete a sequencia, mas para os valores de temperatura e termina com o envio de 8 bits para informar o MCU que a transmissão terminou.

Na subsecção Leitura DHT11Subsistema *PIC* será demonstrado o código utilizado para a comunicação do PIC com o DHT11.

4.3. *ADXL345 Digital Adafruit* (acelerómetro)

O ADXL345 é um acelerómetro de 3 eixos, de alta resolução (13bits) que consegue medir aceleração na ordem dos +-16g. O seu sinal de saída é apresentado em 16bits formatados em complementos para dois. A informação disponível no sensor pode ser acessível tanto por SPI como I2C.

Este sensor é adequado para uma medição estática da aceleração da gravidade por exemplo na inclinação (tilt) ou na medição da aceleração dinâmica resultante do movimento ou choque. A sua elevada resolução (4 mg / LSB) permite a medição de alterações de inclinação inferiores a 1,0 °.

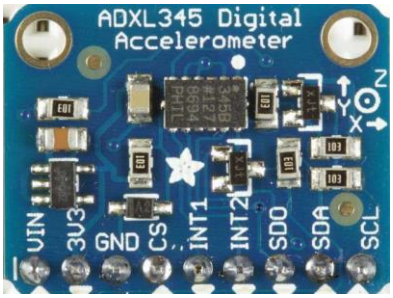

Figura 4-5 - Acelerómetro ADXL345

4.4. Altímetro *MPL3115A2*

O Altímetro MPL3115A2 [9] é um sensor piezoresistivo de pressão, temperatura e altitude que comunica através do protocolo *I2C*.

Figura 4-6-Altimeter MPL3115A2

Possui 20 bits para a leitura de pressão e altitude, para a leitura da temperatura utiliza apenas 12 bits e possui uma memória FIFO interna. Consegue ler valores de pressão na ordem dos 20 a 110 kPa, altitude na ordem dos -698 até as 11775 m e por fim temperaturas que podem variar dos -40ºC até aos 85ºC.

A Figura 4-7 mostra o diagrama de blocos do altímetro e podemos observar que este possui um bloco "*sensor configuration and output data registers*", que possui todas as configurações necessárias para comunicar com o sensor, como por exemplo o endereço *I2C* que permite aos dispositivo "master" identificar o altímetro numa comunicação *I2C*., como foi descrito na secção anterior

Os pinos INT1 e INT2 permitem, através da interface *I2C*, ao utilizador ajustar a configurações do sensor como o *threshold* e o tempo de interrupção

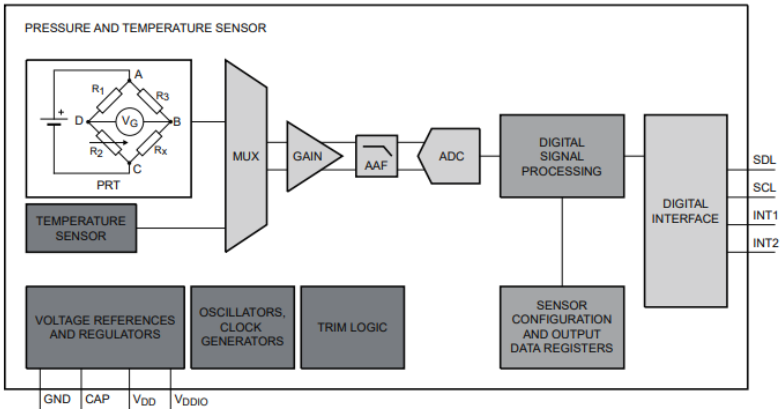

Figura 4-7- Diagrama de blocos do MPL3115A2

Na subsecção Subsistema *Arduíno Mega* 2560 será demonstrado o código utilizado para a comunicação do Arduíno com o *MPL3115A2*.

4.5. *Sonar HC-SR04*

O sensor ultrassónico Sonar HC-SR04 [10] ´utiliza o sonar para determinar a sua distancia a um objeto perante a sua linha de vista.

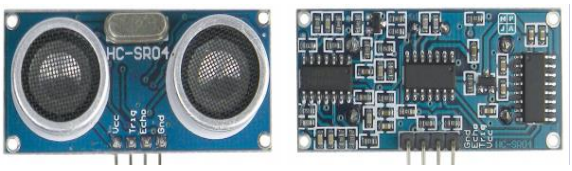

Figura 4-8-Sonar HC-SR04

Como podemos observar na Figura 4-10, o funcionamento deste sensor é muito simples. Inicialmente emite um sinal de alta frequência pelo pino *trig*, que será refletida (ou não) por um objeto e recebida pelo pino *echo* do sonar. Uma vez que a velocidade do som no ar é conhecida, através do intervalo de tempo entre a transmissão e a receção é possível determinar a distância. O diagrama temporal do sonar pode ser visualizado na Figura 4-10

Este sensor permite medir distâncias entre os 2cm e os 4m.

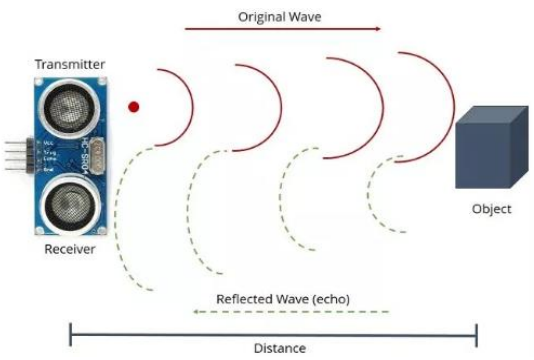

Figura 4-10-Funcionamento do sensor Sonar

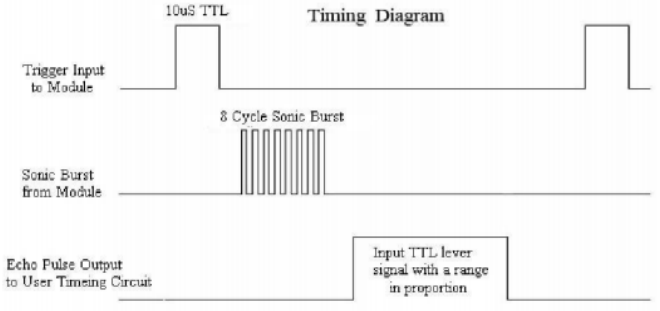

Figura 4-9-Diagrama temporal do Sonar

4.6. *LDR*

O LDR (Light Dependent Resistor), é nada mais que uma resistência variável consoante a intensidade de luz do meio que o rodeia. Quando colocado num local bem iluminado a sua resistência encontra-se na ordem dos KΩ, a medida que o local vai escurecendo a sua resistência vai aumentado de uma forma exponencial atingindo os MΩ Figura 4-11. Esta variação de resistência é causada pela incidência de fotões na superfície do componente o que leva a libertação dos eletrões que constituem o sensor, com isto a condutividade do sensor aumenta diminuindo a resistência do componente.

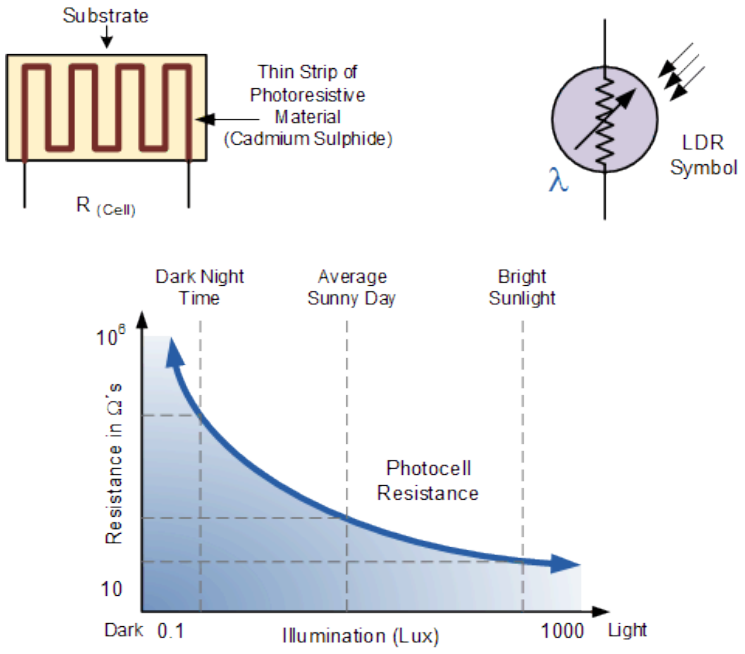

Figura 4-11 - LDR

5. Sistema

5.1. Descrição do Sistema *IoT* implementado

Como referido anteriormente, no âmbito da unidade curricular de Sistemas para Internet das coisas foi

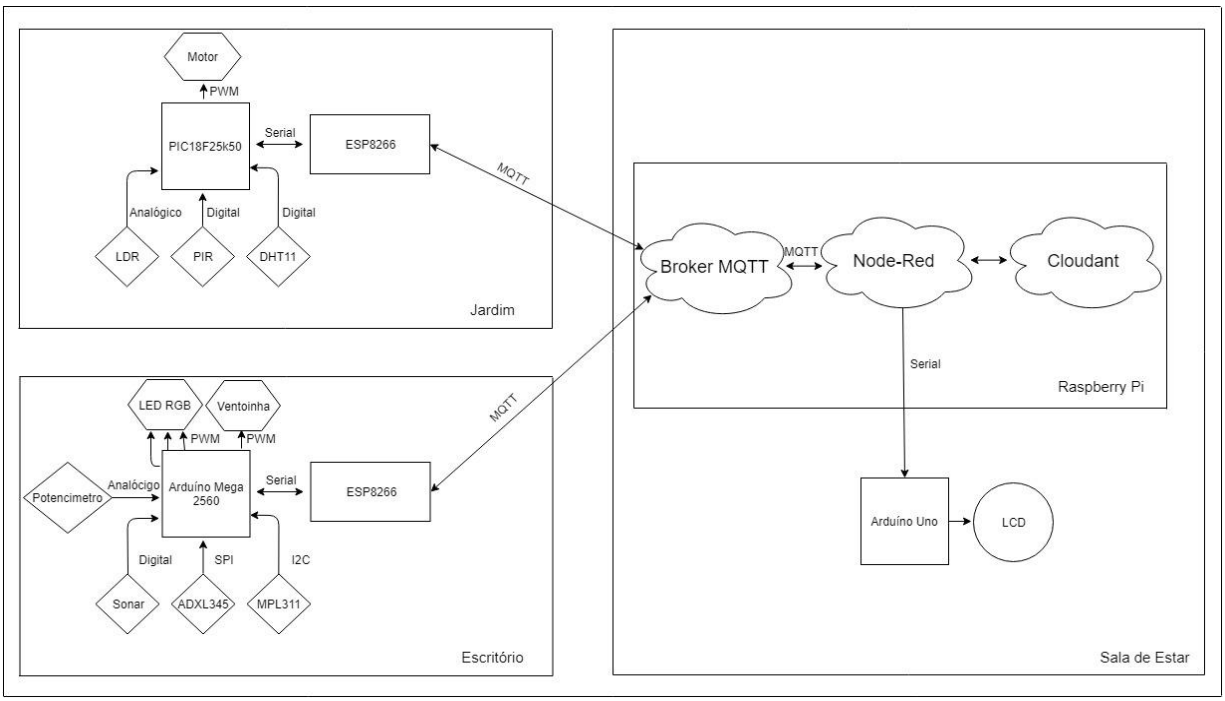

Figura 5-1 - Diagrama de blocos do Sistema IoT desenvolvido

desenvolvido um sistema de monitorização inteligente de uma habitação. Uma habitação é constituída por vários sectores, sendo que o nosso sistema foca-se em apenas dois, o jardim e o escritório.

No jardim existe um subsistema constituído por um controlador (pic18F25k40), um módulo wi-fi (esp8266), três sensores e um motor no qual o controlador atua. O objetivo deste subsistema é realizar a monotorização de forma eficaz e de alta fidelidade, controlando aspetos como a temperatura, humidade, a luminosidade e todo o sistema de rega através de um motor.

No Escritório existe um subsistema constituído por um controlador (arduino mega 2560), um módulo wi-fi (esp8266), três sensores, um potenciómetro para simular o aquecimento de um determinado dispositivo e uma ventoinha que é atuada através do controlador.

O diagrama de blocos pode ser observado na Figura 5-1.

Os dados de todos os sensores são guardados numa cloud (cloudant IBM).

5.2. Servidor *Raspberry Pi*

A *Raspberry PI* desemprenha um papel fundamental no nosso sistema *IOT*, interligando todos os subsistemas.

Na raspberry existem três serviços diferentes a correr, o Node-red , o servidor broker MQTT e o *ngrok*.

O Node-Red como referido anteriormente permite a monitorização remota dos sistemas através de uma interface WEB.

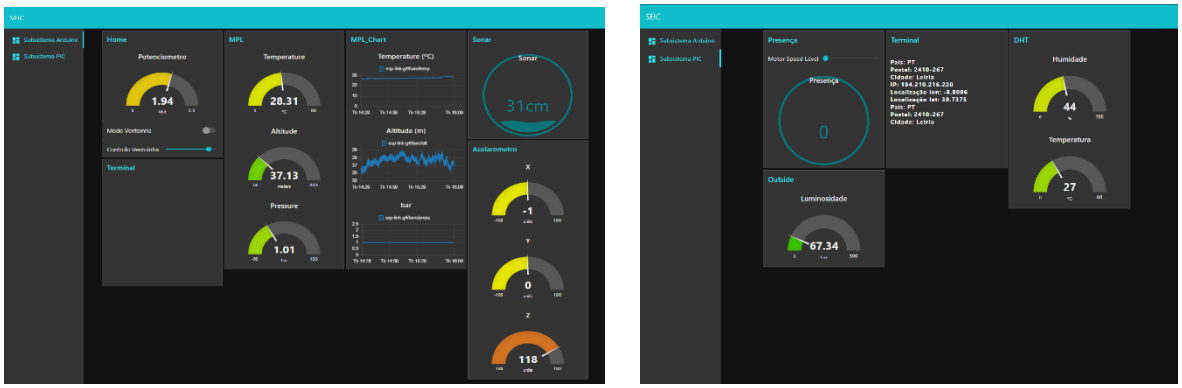

Figura 5-2-Dashboard

Através desta plataforma é possível criar uma *dashboard* que inicialmente só pode ser acedida através da rede local o servidor está a correr. De maneira a contornar esta limitação, existe um serviço de ngrok [11] a correr e sempre que o *raspberry pi* arranca é criado um túnel seguro, com uma ligação cifrada, para o servidor na pi onde o node red está a correr e assim que o túnel estiver criado, a pi notifica o utilizador via email com o dns gerado para aceder ao node-red. Desta forma, não só é possível aceder a *dashboard* em qualquer parte do mundo, mas também é possível aceder a plataforma e configurar.

Com o node-red é possível criar alarmes consoante a informação proveniente dos subsistemas via MQTT e notificar o(s) utilizador(es) via email e/ou Twitter. Como o Node-Red já esta preparado com blocos de MQTT é possível atuar sobre os subsistemas.

Como o Node-Red é baseado em *javascript* e possui blocos de funções e blocos pré-programados é possível fazer umdo processamento dos sensores no servidor, diminuindo a carga nos controladores. Neste sistema foram implementados blocos de áudio, para alertar o utilizador caso algum alarme dispare. Desta forma o utilizador não tem que estar constantemente atento ao email ou à dashboard, basta ter o som alto e a dashboard aberta que consegue ouvir os alarmes sonoros.

Na Figura 5-2 podemos observar a *dashboard* desenvolvida.

O servido broker MQTT foi implementado utilizando o *mosquitto* [12].

A raspberry pi, irá ter associada um LCD com a informação de vários sensores, desta forma caso o utilizador não tenho meio de aceder ao node-red, consegue visualizar em tempo real o parte da monitorização.

Como o node-red foi desenvolvido pela IBM, já possui blocos de função que facilitam a conexão com o serviço da *cloudant* . As informações relativas aos sensores são guardadas neste serviço de cloud num período de 5 minutos.

A fim de saber a localização do servidor foi realizado um pedido HTTP a uma API externa, os dados provenientes deste webserver foram o ip externo do servidor, a sua localização, o país onde se encontra bem como o código postal da localidade. Esta informação é revelada no sector terminal e ainda revelada num mapa acessível através do endereço http://10.0.1.1:1880/worldmap/ Figura 5-3.

Em suma o sistema desenvolvido está preparado para ser acedido/programado remotamente de qualquer lugar.

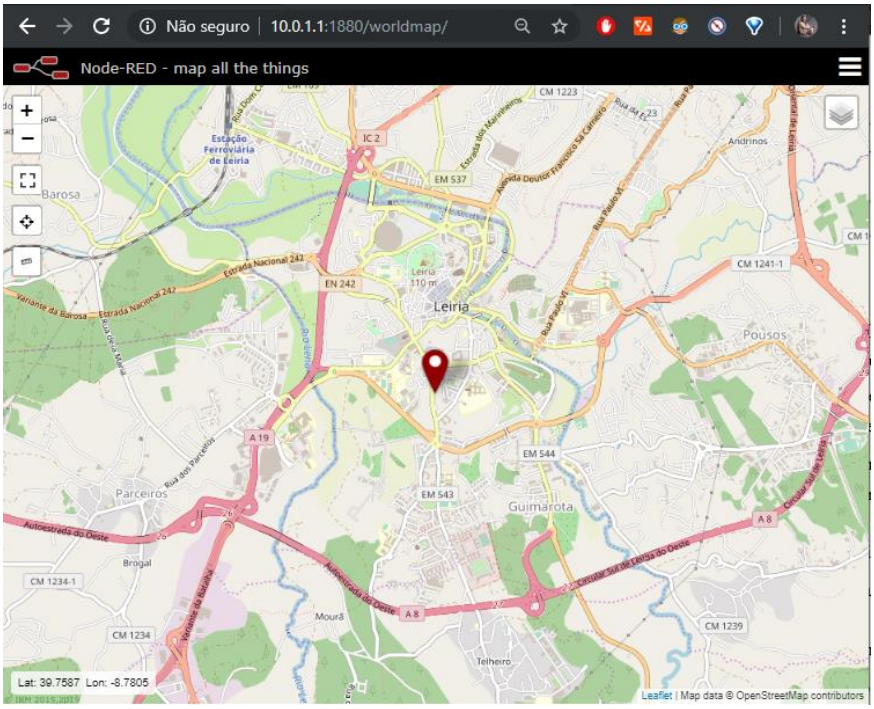

Figura 5-3 - Localização do servidor

5.3. Subsistema *PIC*

O subsistema colocado no jardim foi implementado no pic18F25k40 é usado para ler a luminosidade atual por intermedio do LDR, assinalar a presença de algum corpo junto do sensor PIR, recolher informação acerca da temperatura e humidade com o sensor DTH11 e ainda controlar a velocidade de um pequeno motor DC.

Para podermos comunicar entre o pic e o node-RED foi necessário encontrar elemento intermediário que se ligue a uma rede wifi e por fim subscreva e publique no respetivo tópico MQTT neste caso escolhemos o ESP8266. Surge então a necessidade de comunicar entre o pic e o esp e para este efeito foi utilizada uma comunicação serial assíncrona os dados são solicitados pelo esp através do envio de caracteres escolhidos para o efeito aos quais o pic vai responder ou atuar consoante o pretendido.

O motor deste sistema pretende simular a atuação de um sistema de rega onde se pode controlar a velocidade de rotação do mesmo por meio de um *slider button* na *dashboard*.

5.3.1. Leitura PIR

Este sensor foi o implementado a fim de detetar a presença de alguém no jardim a fim de se necessário interromper o atuador de rega. Na seguinte imagem Figura 5-4 é possível visualizar o envio do valor de estado do sensor PIR a partir do pic.

```
sprintf(msg1,"SecondMQ_%d\0",PIR_GetValue());
UARTSendString(msg1,MAX_LENGTH_UART);
flag2=0;
```

Figura 5-4 - Código Sensor PIR

5.3.2. Leitura LDR

A fim de evitar que o sistema de rega trabalhe durante as horas de maior calor que correspondem as horas de alto índice de radiação luminosa foi implementado o sensor LDR para analisar o índice radiação. O código que se segue mostra a leitura, conversão e envio da intensidade luminosa detetada neste sensor. Figura 5-5

```
ADC_SelectChannel(AN0);
ADC_StartConversion();
Vout=  ADC_GetConversionResult()*0.00488;
lux =500/(10.0*((5-Vout)/Vout));
sprintf(msg1,"FirstMQ6_%.2f\0",lux);
UARTSendString(msg1,MAX_LENGTH_UART);
flag1=0;
```

Figura 5-5 - Código Sensor LDR

5.3.3. Leitura DHT11

Os índices de humidade e temperatura também são extremamente relevantes para a monitorização adequada de um jardim moderno então foi também solicitada esta informação ao sensor DTH11. Na figura **Erro! A origem da referência não foi encontrada.** segue o código que sugere esta leitura.

```
DHT11_Start(); /* send start pulse to DHT11 module */
/* read 40-bit data from DHT11 module */
RH_Integral = DHT11_ReadData(); /* read Relative Humidity's integral value */
RH_Decimal = DHT11_ReadData(); /* read Relative Humidity's decimal value */
T_Integral = DHT11_ReadData(); /* read Temperature's integral value */
T_Decimal = DHT11_ReadData(); /* read Relative Temperature's decimal value */
Checksum = DHT11_ReadData(); /* read 8-bit checksum value */
sprintf(msg1, "ThirdMQ_%d\0",T_Integral );
UARTSendString(msg1, MAX_LENGTH_UART);
flag3 = 0;
```

```
void DHT11_Start() {
    DHT_TRIS = 0;
    DHT_LAT = 0; /* send low pulse of min. 18 ms width */
    __delay_ms(18);
    DHT_LAT = 1; /* pull data bus high */
    __delay_us(40);
    DHT_TRIS = 1; /* set as input port */
    while (DHT_PORT & 1); /* wait till bus is High */
    while (!(DHT_PORT & 1)); /* wait till bus is Low */
    while (DHT_PORT & 1); /* wait till bus is High */
}
```

```
char DHT11_ReadData() {
    char i = 0;
    char data = 0;
    for (i = 0; i < 8; i++) {
        while (!(DHT_PORT & 1));
        __delay_us(30);
        if (DHT_PORT & 1) data = ((data << 1) | 1);
        else data = (data << 1);
        while (DHT_PORT & 1);
    }
    return data;
}
```

Figura 5-6 - Código Sensor DHT11

5.3.4. Comunicação com ESP8266

Tal como referido a comunicação serie é solicitada pelos esp, isto é, através do envio de determinados carateres é pedido ao PIC que execute determinadas funções tais como ler sensores ou atuar dispositivos. A forma como estes pedidos são realizados é mostrada na Figura 5-7.

```
void EUSART1_RxDataHandler(void){
    // use this default receive interrupt handler code
    eusart1RxBuffer[eusart1RxHead++] = RCREG1;
    if(sizeof(eusart1RxBuffer) <= eusart1RxHead)
    {
        eusart1RxHead = 0;
    }
    eusart1RxCount++;
    switch (RCREG1){
        case 'A':flag1=1;break;
        case 'B':flag2=1;break;
        case 'C':flag3=1;break;
        case 'D':flag4=1;break;
        case '0': PWM2_LoadDutyValue(0); break;
        case '1': PWM2_LoadDutyValue(500); break;
        case '2': PWM2_LoadDutyValue(750); break;
        case '3': PWM2_LoadDutyValue(999); break;
    }
}
```

```
void loop() {
  client.loop();
  Serial.flush();
  if (Serial.available() > 0) {
    str = Serial.readString();

    if (str.indexOf("Hey") >= 0) {Serial.print("A\n");}
    else if (str.indexOf("FirstMQ") >= 0) {
      publishSerialData("/nodeMCU/", str);
      Serial.print("B\n");
    }
    else if (str.indexOf("SecondMQ") >= 0) {
      publishSerialData("/nodeMCU2/", str);
      Serial.print("C\n");
    }
    else if (str.indexOf("ThirdMQ") >= 0) {
      publishSerialData("/nodeMCUT/", str);
      Serial.print("D\n");
    }
    else if (str.indexOf("FourthdMQ") >= 0) {
      publishSerialData("/nodeMCUH/", str);
      Serial.print("A\n");
    }
    else {Serial.print("Ups" + str);}
  }
}
```

Figura 5-7 - Comunicação PIC18-ESP

5.4. Subsistema *Arduíno Mega* 2560

O subsistema Arduíno é usado para monitorizar o escritório e revela também algumas informações consideradas interessantes para isso dispõem de um sensor MPL que nos devolve pressão, altitude e temperatura, um sonar que nos indica proximidade de algum objeto ao sensor, um acelerómetro onde retiramos informações de posição do equipamento e ainda um potenciómetro que simula disparar um sistema de arrefecimento da divisão através de uma ventoinha que representa um atuador.

Tal como o sistema acima mencionado (Subsistema *PIC*) o arduíno não dispõe de um modulo wifi que permita publicações MQTT para o node-RED então recorremos novamente a um esp8266 no entanto desta vês usamos o firmware esp-link desta forma não foi necessária qualquer programação adicional neste equipamento sendo o arduíno a ter o controle das funções do esp através de comandos AT enviados e recebidos via serial.

5.4.1. Leitura MPL

Surgiu a necessidade de efetuar uma leitura rigorosa da temperatura do escritório para tal foi usado o sensor MPL que estava à disposição, este consegue fornecer ao utilizador também a pressão e altitude algo que achamos também interessante para colocar à disposição. A Figura 4-6-Altimeter MPL3115A2 que se segue da a conhecer como os dados pedidos ao sensor.

```
Adafruit_MPL3115A2 baro = Adafruit_MPL3115A2();

void wifiCb(void* response) {
  ELClientResponse *res = (ELClientResponse*)response;
  if (res->argc() == 1) {
    uint8_t status;
    res->popArg(&status, 1);

    if(status == STATION_GOT_IP) Serial.println("WIFI CONNECTED");
    else {
      Serial.print("WIFI NOT READY: ");
      Serial.println(status);
    }
  }
}

void publishData(){
  double pot;
  float pascals = baro.getPressure();
  float altm    = baro.getAltitude();
  float tempC   = baro.getTemperature();
```

Figura 5-8 - Código Sensor MLP

5.4.2. Leitura Acelerómetro

O acelerómetro foi usado apenas como um objetivo informativo dando a possibilidade ao utilizador de analisar a influencia da sua ação ao movimentar o sensor entre diversas posições. O código que se segue Figura 5-9 mostra como é realizada a leitura deste sensor.

```
readRegister(DATAX0, 6, values);
x = ((int)values[1]<<8)|(int)values[0];
y = ((int)values[3]<<8)|(int)values[2];
z = ((int)values[5]<<8)|(int)values[4];

if(lastx!=x || lasty!= y || lastz != z){
  s = String(x,DEC) + "_" + String(y,DEC) + "_" + String(z,DEC);
  Serial.print("XYZ: " + s + '\n');
  mqtt.publish("esp-link-g4/acl",&s[0]);
}
```

```
void writeRegister(char registerAddress, char value){
  digitalWrite(CS, LOW);
  SPI.transfer(registerAddress);
  SPI.transfer(value);
  digitalWrite(CS, HIGH);
}
void readRegister(char registerAddress, int numBytes, unsigned char * values){
  char address = 0x80 | registerAddress;
  if(numBytes > 1)address = address | 0x40;
  digitalWrite(CS, LOW);
  SPI.transfer(address);
  for(int i=0; i<numBytes; i++){values[i] = SPI.transfer(0x00);}
  digitalWrite(CS, HIGH);
}
```

Figura 5-9 - Código Sensor Acelerómetro

5.4.3. Leitura Sonar

A fim saber a proximidade de alguém perto do escritório foi colocado o Sonar, este permite saber até uma distância máxima de 7 metros se esta alguém próximo si. A Figura 5-10 seguinte da a conhecer como a distancia é medida através do sonar.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
```

Figura 5-10 - Código Sensor Sonar

6. Conclusão

O desenvolvimento de este sistema foi imprescindível para o culminar da UC de Sistemas Eletrónicos para a Internet das Coisas com sucesso. A realização deste trabalho implicou a aplicação de conhecimentos importantes adquiridos em aula e proporcionou ainda o desejo e pesquisa por novos recursos a fim de implementar novas funcionalidades.

Por vezes surgiram alguns problemas que atrasaram o processo de desenvolvimento, estes serviram para aprimorar técnicas e melhorar métodos e aprender novas formas de resolver estas situações.

Desta forma concluímos que o trabalho foi realizado com sucesso o cumprindo todos os objetivos propostos e ainda adicionando outros que achamos pertinentes.

7. Referencias

[1] Wikipédia, “Internet das Coisas,” [Online]. Available: https://pt.wikipedia.org/wiki/Internet_das_coisas.

[2] Wikipédia, “Máquina para Máquina,” [Online]. Available: https://pt.wikipedia.org/wiki/Machine_to_Machine.

[3] Wikipédia, “Inteligência Artificial,” [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence.

[4] Wikipédia, “I2C,” [Online]. Available: https://pt.wikipedia.org/wiki/I%C2%B2C.

[5] Wikipedia, “Open-Drain,” [Online]. Available: https://en.wikipedia.org/wiki/Open_collector.

[6] Wikipédia, “MQTT,” [Online]. Available: https://en.wikipedia.org/wiki/MQTT.

[7] Wikipédia, “Machine to Machine,” [Online]. Available: https://pt.wikipedia.org/wiki/Machine_to_Machine.

[8] M. Eletronics, “DHT11 Humidity & Temperature Sensor,” [Online]. Available: https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf.

[9] nxp, “Altímetro MPL3115A2,” [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MPL3115A2.pdf.

[10] sparkfun, “Ultrasonic Ranging Module HC-SR04,” [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[11] ngrok, “ngrok,” [Online]. Available: https://ngrok.com/.

[12] E. Mosquitto, “Eclipse Mosquitto,” [Online]. Available: https://mosquitto.org/.

[13] Wikipédia, "Indústria 4.0," [Online]. Available: https://pt.wikipedia.org/wiki/Ind%C3%BAstria_4.0.

8. Anexos