P2009-Parâmetros de qualidade da água [Projeto IOT]

———— (————-), ————— (UTAD/ECT), ———— (UTAD/ECT)

***Abstract*—Nowadays , the concept of IoT is known quite everywhere and by everyone ,but it's practical implementation leaves much to be desired. Despite of it's name, the main purpose of the project is to integrate sensors and micro-sensors with cloud services , show how data is treated automatically and how to take the full advantage from communication protocol LoRaWAN.**

Index Terms - IoT , LoRa , platform , Arduino , ttn

***Abstract*—Hoje em dia o conceito da IoT é conhecido quase por toda a gente mas até a data de hoje, numa escala mundial , poucas soluções (práticas) para a sua implementação foram apresentadas. Este projeto visa mostrar uma dessas soluções , bem como a tecnologia e serviços que podem ser usados para a implementação da mesma. Apesar do seu nome , o projeto tem como finalidade realizar a integração de sensores de análise de parâmetros qualitativos da agua com o protocolo de comunicação LoRaWAN e outras plataformas em causa.**

Palavras-Chave - IoT , LoRa , plataforma , Arduino , ttn

I. Introdução

Fig. 1: IoT

ESTE presente projeto serve como exemplo para uma possível integração de vários sensores de medição que permitem avaliar a qualidade de água com serviços em cloud que oferecem serviços como a visualização e posteriormente o tratamento dos dados não sendo necessária a presença física para efeitos do tal. O projeto surgiu no âmbito da **IoT** (Internet of Things) ou em português a Internet das Coisas cujo conceito é integrar a vida quotidiana ou simplesmente a tecnologia do dia a dia com a internet.

Artigo submetido em 9 de Setembro de 2020.

Além do IoT existe o **IIoT** que parte do mesmo conceito que a Internet das Coisas mas que tem o alvo qualquer tipo de industria em vez do uso pessoal. A ideia é usar uma ferramenta industrial onde serão ligadas várias máquinas e permitir a intercomunicação das mesmas dentro de um ecossistema , criando assim uma fluidez e a eliminação do erro humano na produção deixando o ultimo responsável apenas pela monitorização do sistema e avaliação do produto final.

O presente artigo foi elaborado em LaTeX [1] sendo a parte do desenvolvimento executada em C/C++ bem como JavaScript com base na plataforma Arduino na qual existem milhares de projetos até à data de hoje. Para a integração dos dados com a cloud foi escolhida a tecnologia de rede LoRa e posteriormente usado o protocolo de comunicação LoRaWan para enviar os mesmos para a plataforma **TheThingsNetwork** (TTN). Para o tratamento dos dados foi integrada a plataforma **ThingSpeak** na TTN. Para tornar este projeto mais interativo foi escolhida a plataforma Twitter para a divulgação dos dados através de uma rede social.

Nas seções seguintes explicar-se-á porque foi escolhida uma plataforma ou outra, porquê foi implementada uma solução ou outra e quais foram as dificuldades encontradas durante a realização do projeto.

II. Desenvolvimento

Entende-se por **plataforma** , um lugar na internet onde podem ser executadas aplicações com tarefas diferentes do modo que sejam satisfeitas várias necessidades.

Começar-se-á assim com a explicação de cada uma.

A. Arduino

A principal e a mais importante é claro a **Arduino**. A história da mesma parte em 2005 numa instituição Italiana com o intuito de baixar os custos de prototipagem que na altura eram considerados caros por muitos alunos (+-50 USD). Num tempo sem precedentes quando os micro-controladores ganhavam muita popularidade, quando houve o boom micro-tecnológico, a arquitetura Arduino não podia falhar. Ganhou rapidamente a popularidade e em 2008 já tinham vendido mais de 60.000 unidades. O dispositivo em si é constituído por 2 componentes: a componente elétrica e a componente programável. A parte életrica enquadra módulos elétromecânicos previamente instalados que torna o Arduino praticamente um dispositivo PLUG-IN e é por isso que trabalhar com ele torna-se uma tarefa muito fácil deixando a cargo do utilizador apenas a componente programavél. Baseia-se o Arduino em micro-controladores de 8 , 16 e 32 bits da marca Atmel com arquitetura AVR e contém 3 tipos de memória : FLASH , SRAM

(memória operativa) e EEPROM. Destaca-se o Arduino das outras plataformas/placas pelo numero de saídas/entradas que dispõe, uma vez que através destas ligações consegue-se interagir com os sensores ou outros dispositivos semelhantes. Todos os modelos têm ligações de dois tipos: digitais e analógicas variando apenas o numero de cada uma de acordo com o modelo em causa. As ligações digitais tem duas posições lógicas: 0 e 1 , correspondendo ao LOW e ao HIGH . Também tem duas posições:INPUT e OUTPUT ou seja conseguem receber ou enviar sinais que contem dados.

Listagem 1: Definição dos pins

```
pinMode(13, OUTPUT);      //pin 13 como saida
digitalWrite(13, HIGH); //liga o pin 13
```

Os pins analógicos conseguem apenas receber sinais , ou seja servem apenas para entradas do sinal. Para por em funcionamento é preciso definir apenas no inicio do código o pin necessário. A amplitude de medição é de 0 a 5V o que em bits corresponde de 0 a 1023, ou seja ,permite realizar a leitura com uma precisão de 0.005 Volts o que garante uma boa sensibilidade para leitura de sensores como termistores e foto-resistências. Atualmente a Arduino é uma das plataformas mais vendidas , mais utilizadas e principalmente mais baratas do mundo para realizar projetos de automação e de integração de tecnologia na internet (IoT). No projeto atual será usada como plataforma mãe através da qual vamos ligar os sensores de acidez da água e o de condutividade elétrica , ambos analógicos e com a mesma finalidade: efetuar a medição dos parâmetros que permite avaliar a qualidade da água.

B. LoRa

Surgida em 2009 , **LoRa** (Long Range ou Longo Alcance) é uma tecnologia de comunicação com topologia em estrela que permite aos dispositivos alimentados a baixa potencia comunicar entre si ou com dispositivos semelhantes. Com uma camada física situada no modelo OSI, LoRa é uma tecnologia de uma área muito ampla que tem na sua base a modulação radio em frequência linear de banda larga (Chirp Spread Spectrum ou CSS) com uma correção integrada de erros (Forward Error Correction ou FEC). A desmodulação do sinal é feita abaixo dos 20 dB o que representa o limite superior do nível de ruido enquanto outras tecnologias que usam FSK (tecnologia que modula em frequência não linear) não manipulam sinais abaixo dos 8-10 dB do nível do ruido. A tal tecnologia de modulação (CSS) foi usada durante vários anos em equipamentos militares, mas foi a LoRa que ofereceu uma implementação a baixo custo no mercado global. LoRa aumenta a sensibilidade do recetor e utiliza toda a largura de banda do canal o que a torna numa tecnologia muito estável ao ruído e o que faz com que seja uma tecnologia a considerar quando se quer uma ”comunicação limpa” e sem interrupções entre os dispositivos que operam com pequenas quantidades de dados a baixo fluxo. O grande conveniente desta tecnologia é a distância que se pode alcançar a um custo relativamente baixo (pouco uso da largura de banda) quando comparado com outras tecnologias. O tal pode ser conferido na figura 2 onde podemos ver que LoRa chega a ter um alcance superior e com uso muito inferior da largura da banda quando comparado com outras tecnologias. Contudo, a velocidade máxima assegurada é de 5 kbps a uma largura de banda máxima de 500 kHz dependendo da área em que se operam os módulos. Tudo isso é influenciado pelo poder de transmissão (tx power) , largura da banda (bandwith) e pelo fator de propagação(SF). Quando se diminui o poder de transmissão a autonomia vai aumentar mas ao custo do alcance. A taxa de transferência é constituída pelos outros 2 parâmetros ou seja são eles que determinam quão rápido os dados serão transferidos e o tempo que eles vão permanecer no ar. Ao aumentar a largura de banda ou diminuir o fator SF poderá conseguir-se o envio de duas vezes mais dados no mesmo período de tempo , mas o sistema ficará mais susceptível ao ruido e menos estável. De mencionar que cada parâmetro mencionado acima é adaptado a cada sensor em causa dependendo da topologia do terreno e a distância em que o dispositivo se encontra.

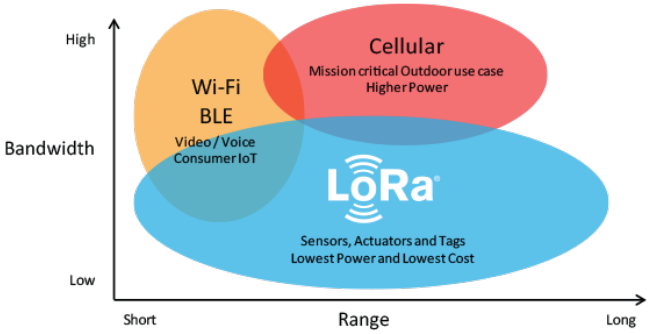

Fig. 2: LoRa Range

C. LoRaWAN

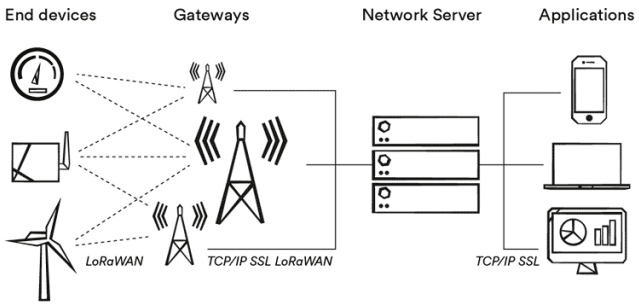

Fig. 3: Exemplo LoRaWAN

Muitas das vezes a definição de LoRaWAN [2] é confundida com a da LoRa, uma vez que **LoRaWAN** é o tipo de protocolo de comunicação construído em cima da camada física LoRa , enquanto a LoRa é o tipo de arquitetura. LoRaWAN é um tipo de rede LPWAN (Low Power Wide Area Network) desenvolvido e apoiado por varias entidades como: IBM , Semtech, Cisco, Swisscom etc. Para a identificação dos dispositivos é utilizado o endereçamento MAC , o mesmo que se usa em Wi-Fi, Bluetooth etc. Este protocolo é adaptado e otimizado para dispositivos terminais (node modules) que

funcionam a baixa potência garantindo assim uma boa relação entre a velocidade de entrega e o tempo de funcionamento quando se alimenta um dispositivo a partir de uma bateria ou uma fonte de alimentação a baixa potencia como por exemplo um painel solar. Num LoRaWAN típico , as gateways retransmitem os dados encriptados dos end-nodes para o servidor central e posteriormente do servidor da aplicação para o utilizador final. Cada país têm parâmetros pré-estabelecidos que definem a frequência de uso de acordo com a região em causa. Embora os parâmetros não são obrigatórios e podem ser alterados à medida , os consumidores são mesmo assim obrigados a estudar as regulações do pais em que se vai usar a tecnologia e caso for necessário adquirir uma licença de uso para a mesma. Para além de ser muito útil quanto à distância de transmissão , LoRa garante também 3 ferramentas de seguridade imprescindíveis. Estas são: NwkSKey, AppSKey e AppKey. Todas as chaves tem 128 bits , similar aos que se usam no estandarte 802.15.4 (WPAN). Quando um dispositivo end-node abre a janela de transmissão as chaves AppSKey e NWkSKey são geradas automaticamente e unicamente em cada sessão(OTAA) sendo também possível a alocação estática das duas (ABP) . Enquanto o NwkSKey é partilhada com o servidor , a AppSKey é mantida em privado. Esta chave é usada para encriptar e decriptar o PAYLOAD. O PAYLOAD é totalmente encriptado entre o node module e o componente do servidor garantindo assim a integridade e privacidade da mensagem. Para além das chaves mencionadas existe uma medida adicional de proteção chamada Frame Counters. Como estamos perante um protocolo rádio , existe a probabilidade de captação e retransmissão dos dados. Este fenómeno tem o nome de reply attack (ataque de resposta). Para o evitar , são colocados dois contadores zerados , downlinkcounter e uplinkcounter respetivamente. Cada vez que o dispositivo faz uplink ou downlink os contadores são incrementados e caso a mensagem recebida ou enviada tenha um numero inferior ao último downlink ou uplink a mensagem será ignorada e no caso do ABP será bloqueado o acesso total de trasmissão sendo necessário a reinserção e reativação do módulo em causa.

Na figura 3 estão presentes 4 elementos chave. LoRa Node module (dispositivos de medição/ recolha de dados) , LoRaWAN Gateway , LoRa Server e LoRa Aplication Client.

D. Sensores(Node Modules)

Os end-nodes LoRa são elementos da LoRaWAN que realizam medições ou controlam dispositivos eletrónicos. Eles controlam-se remotamente e são alimentados por norma a partir de baterias. Os dados na Rede LoRaWAN podem ser transmitidos em ambas as direções , tanto a partir dos end-node para o servidor , como inversamente. Os nós não enviam os dados permanentemente , ou seja eles abrem a janela de transmissão por um curto período de tempo (1 a 6 segundos) ao fim de qual o período de envio termina , mas o de receção é mantido aberto até se esgotar o período. Após isso o sistema entra em hibernação (sleep-mode) até receberem outro pedido de transmissão. Isso depende da classe do dispositivo que está a transmitir. Há 3 classes de dispositivos: A , B ou C.

1) Classe A:
Este end-node abre a janela de transmissão conforme um gráfico pré-estabelecido. Quem inicia o transmissão é o módulo em si e não necessita de confirmação de receção apesar de existir uma ”confirmação” quando necessário (ACK). Este módulo está pronto para receber dados logo após o envio sendo esta classe a mais usada e a mais económica em termos de energia elétrica.

2) Classe B:
Esta classe de módulos abre a janela de transmissão conforme um gráfico estabelecido pelo servidor. O servidor envia os dados de acordo com este gráfico também. Os dispositivos desta classe sincronizam o tempo com o servidor com a ajuda dos ”beacons” que recebe a partir do servidor. Esta classe é caracterizada por ter um tempo de resposta rápida e com uma janela de transmissão prolongada e flexível. Também tem as características dos dispositivos da classe A.

3) Classe C:
Estes dispositivos estão prontos para receber sempre ou seja mantém a janela de transmissão aberta. Os dados são recebidos logo após o envio prejudicando assim a eficiência energética mas garantindo assim a receção quase instantânea dos dados (low latency). Tal como a classe anterior , esta tem as características das outras duas mencionadas acima.

E. Gateways

Fig. 4: Exemplo Gateways

Dispositivo responsável pelo envio dos dados para o servidor e para outro módulo qualquer ligado à este dispositivo. Existem 2 tipos de gateways: colocadas no interior e no exterior. Diferenciam-se as duas pelo tipo de antenas colocadas bem como pela forma como são construídas e pela alimentação das mesmas. Normalmente as gateways colocadas no interior

tem as antenas incorporadas na placa mãe bem como precisam de uma fonte de alimentação exterior. As gateways do tipo OUTDOOR destacam-se pelo poder de penetração e alcance maior devido ao tamanho e a disposição das antenas no exterior. A alimentação muitas das vezes é do tipo PoE. Como este tipo de antenas são colocadas no exterior o fabricante vê-se necessário de adicionar um extra como o indíce de proteção contra poeiras e líquidos. Um exemplo das tais gateways pode se ver na figura 4. O alcance das mesmas é definido pelos parâmetros mencionados anteriormente como o power tx , a largura da banda e pelo fator SF. Quando o fator SF é definido na posição 12 (posição máxima) é possível chegar ao alcance máximo. Normalmente o alcance de uma gateway exterior é de 10-15 km o que permite cobrir uma cidade inteira de dimensões medias. Apesar disso a 16 de Abril de 2020 foi conseguido um envio de dados a mais de 800 km (832 para ser exato) usando uma gateway com um poder de transmissão de 19 dBm (25 mW). O dispositivo foi pendurado num balão de ar e foi lançado várias vezes até ser conseguida uma altura de 38km . A periodicidade de transmissão foi a cada 60000 ms alternando entre os fatores SF7 , SF9 e SF 11 (SF12 não foi usado devido ao movimento instável bo balão) com a ajuda das antenas bipolares do dispositivo com 2dbi de 8.4 cm cada. As mensagens do dispositivo foram captadas por varias gateways entre as quais uma se localizava a 832 km numa montanha na Republica Checa. O tal alcance deve-se a um fenómeno de condutas de evaporação na atmosfera na qual as ondas radio são dobradas e levadas para distancias muito longas criando assim um efeito de guia de onda.

Fig. 5: Transm. Record

F. TheThingsNetwork

Nascida em 2015 é uma plataforma relativamente recente. Realiza o papel do servidor e da aplicação mencionados anteriormente. Tem como finalidade comunicar com os sensores através das gateways, receber ou enviar dados, fazer o processamento na cloud e enviar para o tratamento posterior dos mesmos. Baseia-se exclusivamente no protocolo LoRaWAN e atua como uma plataforma que disponibiliza vários serviços. Atualmente **TTN** é mais uma comunidade que uma plataforma uma vez com a passagem dos anos torna-se muito popular devido aos entusiastas que estão em número crescente.

G. ThingSpeak

Integrado com o MatLab, ThingSpeak proporciona ferramentas de análise de dados muito incríveis. Permite-nos a visualização dados em tempo real de várias formas , sendo eles gráficos , tabelas ou simplesmente na forma numerica através de subtração de dados usando os protocolos HTTP e MQTT.

H. Twitter

Rede social que visa a partilha rápida de mensagens. As tais mensagens têm o nome de tweets com dimensão máxima de até 140 caracteres. A grande utilidade do twitter é a rapidez com que as notícias/avisos tem capacidade de se espalhar devido a uma ferramenta chamada re-tweet. Funciona em cima de várias linguagens de programação como Phyton/C/JavaScript etc.

I. Funcionamento

Os sensores são ligados ao Arduino e colocados em funcionamento. Cada um deles lê os dados obtidos. Como os dois sensores são analógicos há a necessidade de passar os valores obtidos para valores digitais. A conversão é feita de acordo com a escala dimensionada no Arduino e os dados são enviados para TheThingsNetwork. Os dados recebidos pela aplicação da plataforma e enviados ThingSpeak. Aqui, os dados serão tratados de acordo com os parâmetros especificados . Como mencionado anteriormente que se quer a divulgação dos dados através de plataformas sociais integrar-se-á ThingSpeak com Twitter e assim sempre que uma condição for verificada o ThingSpeak irá fazer um Tweet avisando os seguidores.

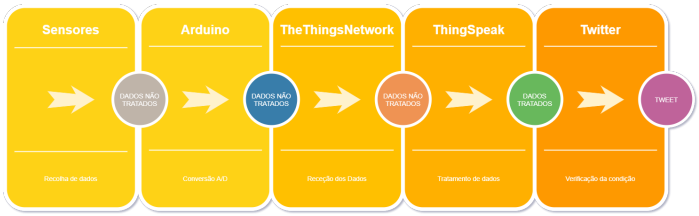

Fig. 6: Diagrama Funcionamento

Equipamentos utilizados:

J. Plataformas electrónicas:

Modelo: Arduino MKR1300

K. Sensores:

1) Sensor ácidez:
Modelo: DFRobots [3]. Leitura: Analogica

2) Sensor Condutividade Elétrica:
Modelo: DFRobots. Leitura: Analogica

3) Sensor Temperatura e Humidade:
Modelo: DHT11 , com marca desconhecida. Leitura: Digital

III. Protótipo

A. No Arduino:

O prototipo é constituido pelo Arduino MKR1300 , o sensor de pH , o da condutividade eletrica e o da temperatura e humidade no ar. Dois dos sensores , respetivamente o do pH e da condutividade eléctrica são da marca DFRobots e o ultimo , o DHT11 de marca desconhecida sendo este um sensor não obrigatório colocado apenas para fins estatísticos da zona em que opera o modulo.

A seguir pode-se ver o código fonte a colocar no Arduino para que possamos ler, converter e enviar os dados para o servidor LoRa. O grande inconveniente do MKR1300 é a falta do tipo de memória **EEPROM** que é uma memória programável não volátil (Real Only Memory) usada em dispositivos eletrónicos para armazenar pequenas quantidades de dados que podem servir para executar comandos neste caso servindo para realizar a calibração do sensor pH e condutividade elétrica. Apesar do facto da memória EEPROM poder ser lida um numero ilimitado de vezes , o número de vezes que ela pode ser reprogramada foi limitado internamente pelo que a maneira que se avança com a reprogramação ,o chip deteriora-se devido ao requerimento de uma tensão de uso mais elevada do que usados normalmente em circuitos eléctricos. O tipo de leitura/escrita é feita de byte por byte ou seja é muito mais lenta que outro tipo de memórias.

Atualmente existe a memória do tipo flash que deriva da EEPROM, muito mais moderna , mais económica, mas com uma vida útil inferior. Apesar de não ser recomendado , devido ao numero limitado de ciclos leitura/escrita que pode prejudicar a memória flash , para poder o usar os sensores que requerem a tal memória EEPROM, foi decidido atribuir uma parte da memória flash neste sentido e usá-la para guardar os valores ou seja emular a memória EEPROM usando bibliotecas especificas. Essas bibliotecas atribuem blocos aleatórios da memória (divididas em contadores) para leitura/escrita para as nossas necessidades. Como no caso deste projeto é preciso guardar valores a fim de realizar calibração o numero de vezes que se vai aceder à memória atribuída é muito raro , ou seja faz mais sentido aceder aos dados diretamente da memória flash sem manter uma cópia dos mesmos em SRAM (static RAM - memória volátil embutida no processador) poupando assim capacidade de processamento do CPU . Embora nem o tempo disponibilizado , nem o conhecimento foi suficiente para garantir o funcionamento correto a 100 porcento do sensor com a memória emulada , mas conseguiu-se ,pelo menos, o cumprimento básico de leitura do mesmo.

1) Componente programável:

Listagem 2: Iniciação das variáveis

```
#include <DFRobot_EC.h>

#include <SimpleDHT.h>
#define EC_PIN A2
int pinDHT11 = 7;

SimpleDHT11 dht11(pinDHT11);

DFRobot_EC ec;
uint8_t txBuffer[8];

float voltage,ecValue,temperature = 25; // 25 -> Valor inicialmente dado para efeitos de calibracao


#define SensorPin A1
#define Offset 0.00
#define LED 13
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40     //quantas vezes os dados s o coletados

double averagearray(int* arr, int number);
int pHArray[ArrayLenth];    //Guarda o valor medio da medicao
int pHArrayIndex=0;
```

Declaração das variáveis , definição de parâmetros de leitura dos sensores e inclusão das bibliotecas necessárias para um correto funcionamento do módulo.

Listagem 3: Get EUI

```
#include <MKRWAN.h>
const char *appEui = "70B3D57ED0030FEA";
const char *appKey = "71114
    EA43603A768424D9013136EFE43";
LoRaModem modem(Serial);
// Escolha a regi o (AS923, AU915, EU868, KR920, IN865, US915, US915_HYBRID)
_lora_band region = EU868;
 void setup()
  {
    while (!Serial);
    if (!modem.begin(region)) {
      Serial.println("Erro ao iniciar!");

      while (1) {}
    };

      Serial.print("A vers o do modulo e: ");
      Serial.println(modem.version());
      Serial.print("O EUI do dispositivo e: ");
      Serial.println(modem.deviceEUI());

    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
      Serial.println("Ha algo de errado. Por favor , verifique a cobertura!");
      while (1) {}
    }
    //O MODEM PERMITE EVIAR UMA MENSAGEM A CADA 2 MINUTOS!?
    modem.minPollInterval(60);
 }
```

Para começar é necessário obter o device EUI. No inicio foi definido o "AppEUI" e a respetiva AppKey onde se vai ligar o MKR 1300 (os credenciais de acesso) para que o modem efetue a ligação com o servidor. Após executar o código e caso a ligação for efetuada com sucesso , vai ser devolvido o mesmo, juntamente com a versão do módulo para poder registar e usar o dispositivo na aplicação do servidor LoRa. Caso a ligação falhar, o MKR vai tentar efetuar a ligação a cada 2 minutos até obter alguma resposta por parte do servidor.

Listagem 4: Código Fonte Arduino

```

    static unsigned long samplingTime = millis();
    static unsigned long printTime = millis();
    static float pHValue,voltage;
```

```
    static unsigned long timepoint = millis();
    if(millis()-timepoint>1000U)  //intervalo de tempo: 1s
    {
      timepoint = millis();
      voltage = analogRead(EC_PIN)/1024.0*5000;  // ler o valor da tensao
      ecValue =  ec.readEC(voltage,temperature);  // converter a tensao em EC com compensa ao da temperatura ( Nao necessario (pode ser eliminado ))
      Serial.print("EC:");
      Serial.print(ecValue,2);
      Serial.println("ms/cm");
}
```

O código recolhe amostras para todos os intervalos acima de 1 segundo. O sinal analógico lido é transformado em bits através de uma regra de 3 simples e depois o valor é imprimido no Serial do Arduino.

Listagem 5: Código Fonte Arduino

```
 if(millis()-samplingTime > samplingInterval)
{
    pHArray[pHArrayIndex++]=analogRead(SensorPin);
    if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
    voltage = averagearray(pHArray, ArrayLenth)*5.0/1024; //Transforme os valores da tensao em valores de pH (ajuta a escala)
    pHValue = 3.5*voltage+Offset;
    samplingTime=millis();
}

  if(millis() - printTime > printInterval)   // Cada 800 milisegundos imprima o valor do pH
{

  Serial.print("pH value: ");
  Serial.println(pHValue,2);
  printTime=millis();
}
```

Quando a diferença entre o tempo atual e o tempo decorrido desde a iniciação da variável "milis" for maior que a do intervalo de recolha da amostra (definido no inicio do código) o valor da tensão é transformado na escala do pH, armazenado e imprimido também no Serial do Arduino.

Listagem 6: Código Fonte Arduino

```
 byte temperature1 = 0;
 byte humidity = 0;
 int err = SimpleDHTErrSuccess;
 if ((err = dht11.read(&temperature1, &humidity, NULL)) != SimpleDHTErrSuccess) {
 Serial.print("Read DHT11 failed, err="); Serial.print(err);delay(1000);
 return;
}
```

Após a definição das variáveis temperatura e humidade o sensor é lido. Caso os valores forem nulos , uma mensagem de erro é enviada para o Serial.

Listagem 7: DEC para HEX

```
    txBuffer[0] = ((int)temperature1 >> 8) & 0xff;
    txBuffer[1]  = (int)temperature1 & 0xff;

    txBuffer[2] = ((int)humidity >> 8) & 0xff;
    txBuffer[3]  = (int)humidity & 0xff;

    pHValue = pHValue* 3000;

    txBuffer[4] = ((int)pHValue >> 8) & 0xff;
    txBuffer[5]  = (int)pHValue & 0xff;

    ecValue = ecValue * 3000;
    txBuffer[6] = ((int)ecValue >> 8) & 0xff;
    txBuffer[7]  = (int)ecValue & 0xff;
```

Os valores guardados de cada sensor são transformados de decimal para hexadecimal (base 8) através de um operador chamado shifting. Trata-se de uma operação de manipulação de bits em que os bits são mudados de posição em função da direção apontada. Como neste caso é feito o shift para a direita o valor decimal é transformado em binário e a sequência dos bits em binário é alterada adicionando 0's e bits significativos no inicio no número. Surge a necessidade de fazer shifting uma vez que se quer o envio de uma quantidade muita pequena de dados e para efeitos de encriptação e segurança. Na linha 8 bem como na linha 12 as variáveis são redefinidas e multiplicadas por 3000, passando assim para números inteiros os valores inicialmente guardados. O tal é feito uma vez que não é possível usar o operando shift para números em float.

Listagem 8: Código Fonte Arduino

```
    Serial.println("HEX Temperatura");
    Serial.println(txBuffer[0], HEX);
    Serial.println(txBuffer[1], HEX);
    Serial.println("HEX Humidade");
    Serial.println(txBuffer[2], HEX);
    Serial.println(txBuffer[3], HEX);
    Serial.println("HEX pH");
    Serial.println(txBuffer[4], HEX);
    Serial.println(txBuffer[5], HEX);
    Serial.println("HEX eC");
    Serial.println(txBuffer[6], HEX);
    Serial.println(txBuffer[7], HEX);

    modem.beginPacket();
    modem.write(txBuffer[0]);
    modem.write(txBuffer[1]);
    modem.write(txBuffer[2]);
    modem.write(txBuffer[3]);
    modem.write(txBuffer[4]);
    modem.write(txBuffer[5]);
    modem.write(txBuffer[6]);
    modem.write(txBuffer[7]);
```

O valor de cada sensor é imprimido no Serial após a conversão para Hexadecimal garantindo assim a integridade e monitorização dos dados. A seguir os dados são enviados para a gateway. A função de envio pode ser reduzido para modem.write(txBuffer, HEX) enviando o vetor por inteiro e não só as posições especificadas.

Listagem 9: Código Fonte Arduino

```
    err = modem.endPacket(txBuffer);
    if (err > 0) {
      Serial.println("Sucesso! A mensagem foi enviada corretamente!");
      } else {
      Serial.println("Erro ao enviar a mensagem :(");
      Serial.println("(s  consegues enviar uma quantia limitada de mensagens por minuto dependendo da intensidade do sinal)");
      Serial.println("(a tal pode variar de uma mensagem por segundo ate uma por minuto)");
```

```
    }
    delay(10000);
  }
```

Caso o pacote de dados não for entregue com sucesso ou o sinal for fraco , um erro é mostrado sinalizando assim que os dados não foram entregues. A quantidade de dados que pode ser enviada vária em função da intensidade do sinal que o end-node está a receber. Quanto maior for o coeficiente SF menos dados será possível enviar por minuto mas o alcance do envio será maior.

Listagem 10: Função Average pH

```
double averagearray(int* arr, int number){
int i;
int max,min;
double avg;
long amount=0;
if(number<=0){
  Serial.println("Erro ao fazer a media!/n");
  return 0;
}
if(number<5){   //calcular diretamente a
  estatistica
  for(i=0;i<number;i++){
    amount+=arr[i];
  }
  avg = amount/number;
  return avg;
}else{
  if(arr[0]<arr[1]){
    min = arr[0];max=arr[1];
  }
  else{
    min=arr[1];max=arr[0];
  }
  for(i=2;i<number;i++){
    if(arr[i]<min){
      amount+=min;          //arr<min
      min=arr[i];
    }else {
      if(arr[i]>max){
        amount+=max;      //arr>max
        max=arr[i];
      }else{
        amount+=arr[i];  //min<=arr<=max
      }
    }//if
  }//for
  avg = (double)amount/(number-2);
}//if
return avg;
}
```

Função de calibração que calcula o valor médio de pH lido num pequeno intervalo de tempo. Usada para obter mais precisão de medição.

2) Explicação do código inteiro::
O código começa pela inclusão das bibliotecas necessárias para a execução sem erros do código. A seguir declaram-se todas as variáveis que vão ser usadas globalmente e definida o endereço da aplicação no servidor "AppEUI" e a respeitiva chave onde vai se ligar o MKR. O modem vai tentar efetuar a ligação com o servidor , caso for efetuada com sucesso , vai ser devolvido o device EUI , juntamente com a versão do módulo , caso contrário o MKR vai tentar efetuar a ligação a cada 2 minutos. Para a leitura dos sensores , o intervalo no arduino será a cada 10 minutos mas enviado será apenas um valor a cada 24 horas , ou seja em 24 horas teremos 24 medições da qualidade da agua. O tal foi feito para otimizar e reduzir o consumo de energia , mas ao mesmo tempo para ter uma leitura de dados constante e em tempo real. Neste prototipo foram efetuadas medições seguidas para efeitos de demonstração. Para uma perceção melhor do código a seguir será disponibilizado a figura 7 com o respetivo fluxograma do código.

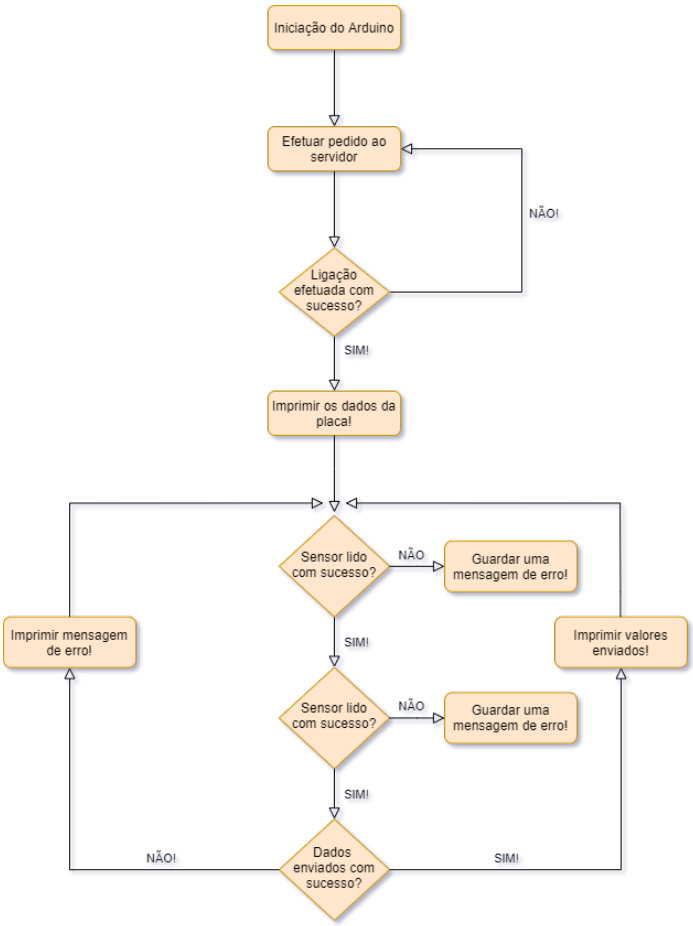

Fig. 7: Fluxograma

3) Componente Elétrica:
Como se pode verificar na figura numero 8, cada sensor tem 3 ligações. Uma que serve para efetuar a ligação a terra (GND) , outra que serve para alimentação (VCC 5V) e a terceira para efetuar o envio de dados para o Arduino. Como mencionado anteriormente 2 dois sensores são analógicos e um digital.

B. Servidor:

Para começar foi necessário efetuar o registo nas plataformas que futuramente irão processar os dados. A plataforma TheThingsNetwork é grátis para usar enquanto o ThingSpeak requer uma conta MatLab com um plano anual de pagamento.

1) TheThingsNetwork: Após efetuar o registo foi necessário criar uma Aplicação na divisão Console onde introduziu-se o ID da aplicação , juntamente com a descrição da mesma e a região na qual pretendemos registar a aplicação. A seguir foi necessário correr o código da Listagem 3 para obter o device

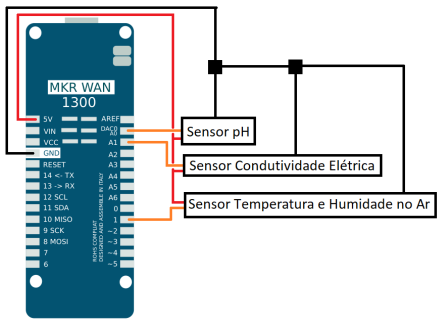

Fig. 8: Prototipo Circuito

EUI e registar o MKR. Na figura 9 está o resultado com todos os credenciais para efetuar o envio de dados.

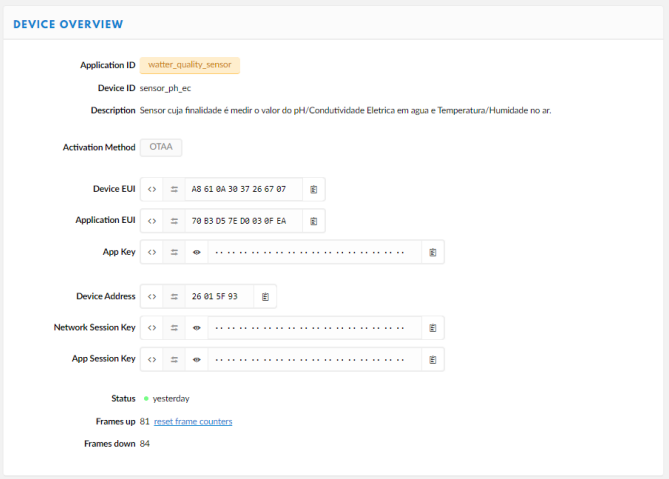

Fig. 9: Registo de uma aplicação

Para que os dados recebidos sejam recebidos corretamente e que se apresentem na forma decimal foi necessário ir a divisão PAYLOAD e definir a função decoder onde foram invocados 4 metodos com o objeto ”decoded”. Os bytes que fizeram right shift no Arduino agora passam novamente para a posição anterior (left shift) e o numero em hexadecimal é passado para decimal. De mencionar que para as variáveis pHValue e ecValue foram efetuadas operações inversas de modo que os dados possam ser apresentados em float (ver Listagem 4).

Listagem 11: Código PAYLOAD TTN

```
function Decoder(bytes, port) {
var decoded = {};

decoded.Temperature = (((bytes[0]) << 8| bytes
[1])).toFixed(2);
decoded.Humidity = (((bytes[2]) << 8| bytes[3]))
.toFixed(2);
decoded.pHValue = (((bytes[4]) << 8| bytes[5])
/3000).toFixed(2);
decoded.ecValue = (((bytes[6]) << 8| bytes[7])
/3000).toFixed(2);

return {
  field1: decoded.Temperature,
  field2: decoded.Humidity,
  field3: decoded.pHValue,
  field4: decoded.ecValue

}
}
```

2) ThingSpeak:: Na divisão integrations na aplicação da TTN foi necessário fazer link entre as contas das duas plataformas. O TTN assim enviará automaticamente os dados de uma plataforma para outra através do objeto return definido no PAYLOAD. Como se pode ver na listagem anterior cada field tem o respetivo método que invoca o objeto e envia posteriormente os dados para o ThingSpeak.

Foi necessário criar um canal onde preencheu-se o nome/descrição e o numero de field’s que pretendemos usar (no nosso caso 8 , 4 em forma de gráfico e 4 em forma numérica).

Resultado:

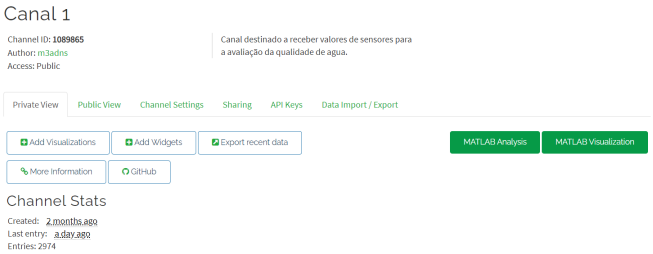

Fig. 10: Canal ThingSpeak

IV. Resultados e discussão

A ideia inicial do projeto era integrar o prototipo feito no tanque de água presente no Polo I das Engenharias da UTAD e observar alguma alteração na qualidade da água, mas dado às circunstâncias e impossibilitados de efetuar as medições devido à falta de energia elétrica, bem como na ausência duma gateway no lugar em causa, foi escolhido simular o tanque com um recipiente com água a colocar dentro. Assim , foram compradas águas de várias marcas, desconhecidas, com propriedades diferentes, e foi simulada a alteração da qualidade da agua, elevando ou diminuindo a acidez e a condutividade da mesma. Na ausência de Portugal , recorreu-se ao site TTNMapper para encontrar a gateway mais próxima. A seguir usou-se uma ferramenta no site , chamada Colour Radar para analisar o raio do alcance da mesma e encontrar a posição ideal para ligar. Os dados da gateway usada para transmitir os dados pode ser acedida através da figura 11 bem como pode ser visualizada acedendo ao link seguinte: LoraGw-Roof. Como se pode ver na figura 11 as ondas rádio viajam em linha reta a não ser que haja reflexão. As ondas de uma cor mais quente , como vermelha, indicam uma intensidade de sinal mais fraca , abaixo dos 100 dbm enquanto as cores mais frias significam um sinal mais potente , acima dos 120 dbm. Após a analise foi decidido enviar os dados a partir de uma localização à 1 km de distância da gateway. Os resultados serão partilhados nas figura 12 , 13 , 14 , 15 e 16.

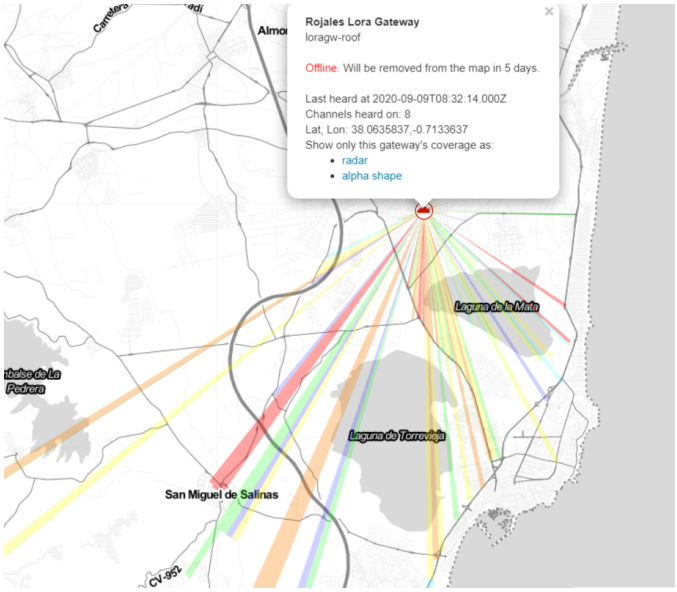

Fig. 11: Diagrama Funcionamento

```
EC:1.19ms/cm
pH value: 0.30
HEX Temperatura
0
1A
HEX Humidade
0
42
HEX pH
3
85
HEX eC
D
F4
Sucesso! A mensagem foi enviada corretamente!
```

Fig. 12: Diagrama Funcionamento

Na figura 12 está um exemplo da Serial do Arduino com valores aleatórios durante a medição de EC e de pH em valor decimal bem como os valores de todos os sensores em hexadecimal.

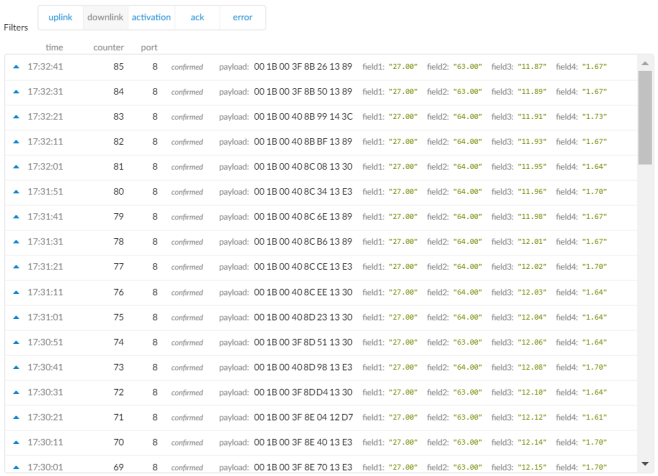

Fig. 13: Downlink TTN

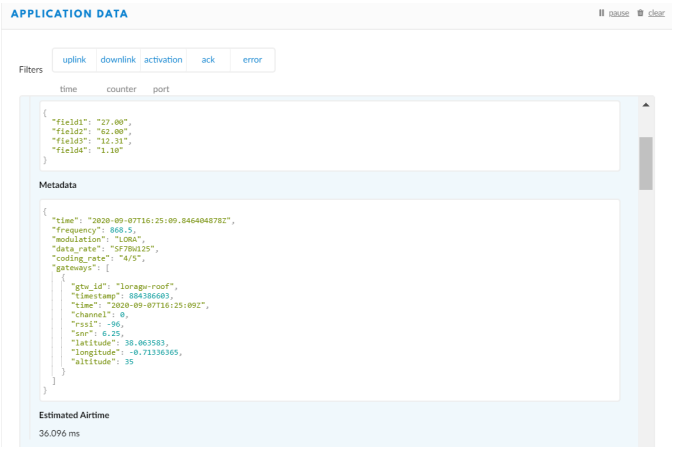

Fig. 14: Detalhes TTN

Na figura 13 estão presentes os downlinks realizados pelo servidor vindos do MKR. Estão presentes dados como , o frame counter , o tempo em que se realizou o envio , a porta , o PAYLOAD (valor hexadecimal recebido) e os respetivos fields recebidos.

Na figra 14 , estão todos os detalhes de envio da gateway. Pode-se comprovar que um dos envios foi realizado , às 16 horas e 25 minutos , à uma frequência de 865.5 mHz com um fator spread SF7. Isso quer dizer que o módulo que enviou os dados estava relativamente perto da gateway e não foi necessário usar um coeficiente SF de alcance máximo. Da mesma maneira pode-se verificar que os dados foram recebidos pela gateway com o id ”loragw-roof”, à uma altura de 35 metros numa latitude ”38.063583” e longitude ”-0.71336365” , a mesma que está presente na figura 11. O tempo que os dados recebidos permaneceram no ar foi de 36.096 ms.

As duas figuras , 15 e 16 respetivamente , pertencem à plataforma ThingSpeak. Cada field definido anteriormente tem 2 posições , uma na forma gráfica e outra na digital para facilitar a leitura e interpretação dos dados. Os dados são atualizados em tempo real à medida que os dados são recebidos.

Fig. 15: Gráficos ThingSpeak

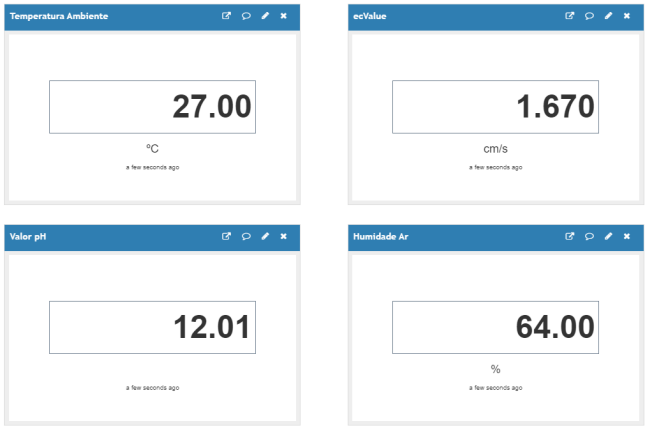

Fig. 16: Leitura digital

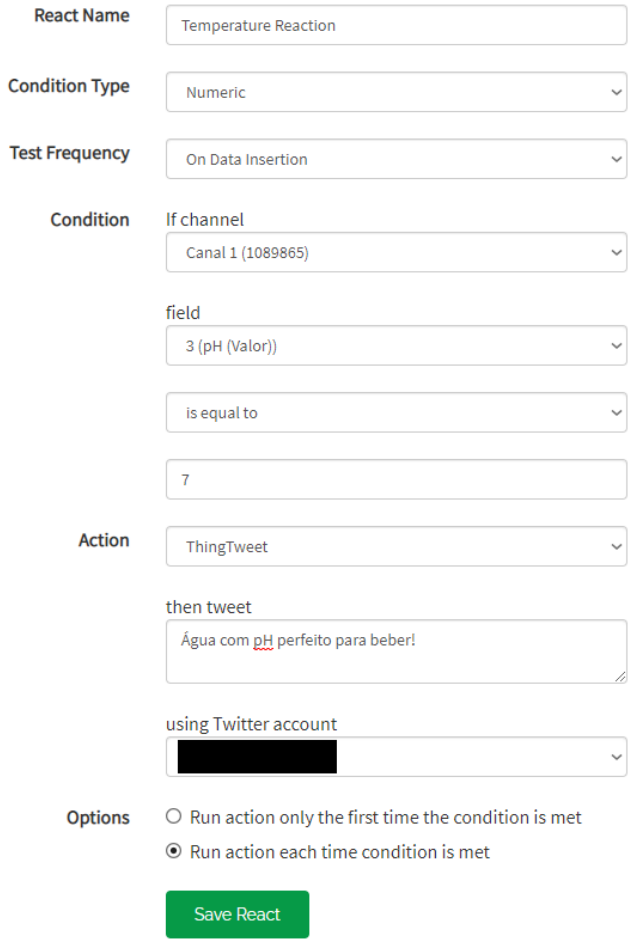

Fig. 17: Twitter react

No entanto ainda falta a integração do Twitter no ThingSpeak. Para realizar isso é necessário fazer o link da conta do Twitter inserindo a API Key no ThingSpeak. Após isso criou-se um React onde criou-se uma condição para quando o pH for igual à 7 o ThingSpeak irá fazer um tweet com a mensagem "Água com pH perfeito para beber!".

V. Conclusões

Durante a realização do trabalho verificaram-se vários lapsos no que diz respeito ao conhecimento bem como à aplicação prática do que foi pretendido. O alvo deste projeto não foi efetuar a medição dos parâmetros da água , mas sim mostrar de que forma é possível automatizar, enviar e efetuar o tratamento posterior dos dados com a tecnologia disponível nos dias de hoje e como facilitar a nossa vida bem como a vida profissional. No entanto durante a realização do trabalho o resultado pretendido foi alcançado , com recurso à internet/livros e ao conhecimento de outras pessoas e assim permite concluir que LoRa é uma solução viável , muito barata e com alto rendimento para implementar neste tipo de projetos quando se quer retirar o rendimento máximo. Desta forma dá-se por concluído o projeto em causa. Obrigado pela leitura!

Agradecimentos

Agradeço ao professor xxxxxxxxxx pelo tempo, equipamento e conhecimento fornecido durante o semestre do ano letivo presente , sem o qual seria impossível para mim realizar o pretendido neste projeto. O professor xxxxxxxxx não só proporcionou ajuda mas também motivação e dedicação ao projeto.

Referências bilbiográficas

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.
[2] LoRaWAN Allience site, *2015*.
[3] DFRobots