Instituto Superior Técnico

Engenharia Eletrotécnica e de Computadores

Redes Móveis e Sem Fios

Sistema de Agricultura IoT

Autores:
Paulo Eusébio 81607
Renato Henriques 81588
Grupo 3

Corpo Docente:
Prof. António Grilo

3 de Junho de 2018

Conteúdo

1 Introdução

Este relatório aborda detalhadamente o projeto *Sistema de Agricultura IoT* no âmbito da disciplina *Redes Móveis e Sem Fios*. De inicio apresenta-se uma arquitetura global do sistema com a interação entre todos os componentes relevantes, seguida de uma descrição das componentes de hardware e software utilizadas, quais as suas implicações e características. São também definidos e justificados os protocolos e formas comunicação do sistema. Por fim avalia-se a solução desenvolvida e os passos para tornar o projeto ainda mais completo.

2 Arquitetura do Sistema

2.1 Descrição

Este projeto consiste numa solução de controlo e monitorização inteligente de um sistema de rega de um campo agrícola. Este sistema consiste num conjunto de sensores que efetua leituras de temperatura, humidades do solo e do ar, incluindo ainda uma bomba de água que atua automaticamente consoante a necessidade de rega, esta necessidade é definida pelo utilizador.

A monitorização é feita com recurso a uma aplicação em Android a partir da qual é possível visualizar o estado atual do sistema. O controlo também é feito pela aplicação através da definição de valores de atuação de uma bomba de água ou pela sua ativação direta por parte do utilizador.

2.2 Arquitetura Global

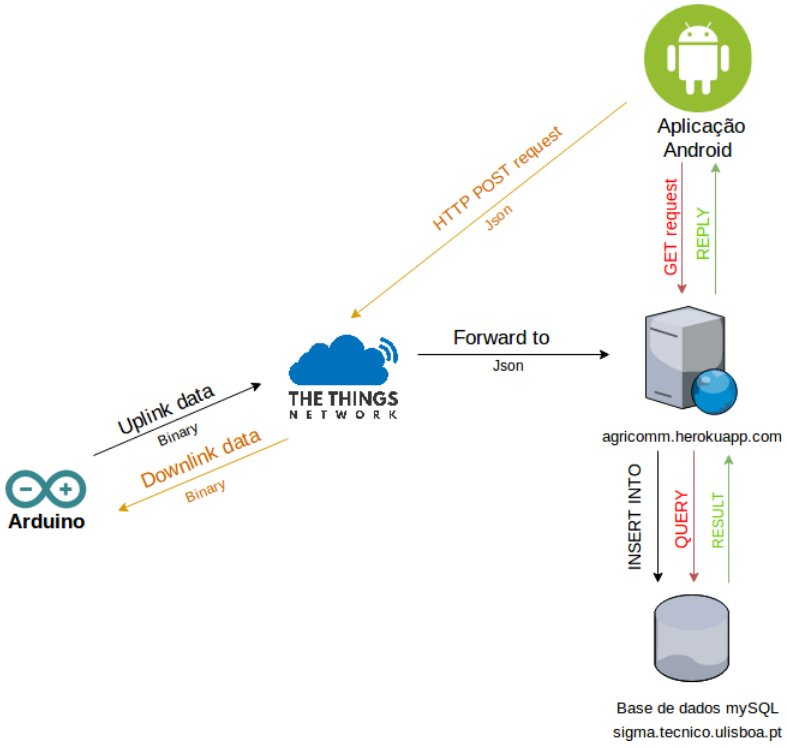


Figura 1: Esquema do funcionamento do sistema desenvolvido.

O esquema global de funcionamento segue a Figura 1. É necessário todo um processamento e reencaminhamento de dados entre o sistema e a aplicação controlada pelo utilizador, em que importa salientar a adição de um base de dados intermédia que permite o armazenamento a longo prazo de dados do sistema, de forma a poder efetuar cálculos ou consultas no futuro.

3 Hardware

3.1 Sensor de Humidade do Solo

O sensor de humidade de solo é o SEN0114 fabricado pela DFRobot. O sensor utiliza duas sondas para passar corrente entre si pelo solo e assim medindo a sua resistência. Desta forma é possível obter o nível de humidade do solo, já que quanto mais molhado estiver o solo mais facilmente conduz eletricidade.

A tensão de alimentação do sensor é 3.3V ou 5V e a tensão de saída corresponde a um intervalo entre 0 e 4.2V.Segundo a documentação, caso a tensão convertida nos canais ADC esteja entre 0 e 300 então pode-se considerar o solo seco, caso esteja entre 700 e 950 então pode-se considerar que o sensor está em água. Entre estes valores considera-se que o sensor está em solo húmido.

Apesar deste sensor ser suficiente para ter uma noção da humidade do solo, vale a pena comentar que a precisão do sensor não é boa e por isso só deve ser utilizado para projetos em que esta caraterística não tenha impacto.

3.2 Sensor de Temperatura e humidade relativa do ar

O sensor DHT11 é composto por uma componente resistiva de medição de humidade e termístor para medir a temperatura do ar. Para além disso, estes componentes são ligados a uma unidade de processamento de 8 bits que permite comunicações digitais viável, rápida e resistente a interferências. Como os fios que serão usados para ligar o sensor ao *Arduino* serão de curto comprimento, é necessário utilizar uma resistência de pull-up para garantir que a entrada tem sempre o estado lógico desejado.

Quanto às especificações técnicas do sensor, este consegue medir desde 20% a 90% da humidade relativa do ar com uma precisão de $\pm 5\%$ e temperaturas dos 0ºC aos 50ºC com uma precisão de ± 2ºC. Apesar da precisão não ser muito relevante para o sistema agrícola, o facto do sensor não conseguir ler temperaturas negativas é um pouco limitativo.

A comunicação *Serial* entre o sensor e o MCU é feito com apenas um canal mas duas direções e dura aproximadamente 4ms. O formato dos dados enviados e o processo de comunicação está descrito da documentação do sensor [1].

3.3 Atuador

O sistema de atuação consiste apenas na (des)ativação de uma bomba de água assim que os *thresholds* são ultrapassados ou a ordem é dada pelo utilizador. A bomba de água é um motor que trabalha a uma tensão entre os 4V e os 12V e que consome uma corrente de 0.8A. Para alimentar o motor é utilizada uma pilha de 9V. Como a bomba não tem um pino de controlo, foi necessário utilizar um circuito com uma configuração de Coletor aberto adicionando um transístor MOSFET para ser possível controlar o estado da bomba com um pino digital do *Arduino*.

3.4 Módulo LoRa

A transmissão e receção de sinais rádio é feita pelo módulo RFM95 [7] que usa a técnica de modulação LoRa™ [8]. Este módulo oferece a possibilidade de configurar parâmetros como a potência de transmissão e frequência. Este módulo é alimentado a uma tensão de 3.3V. A comunicação entre o RFM95 e o microcontrolador é feita com o protocolo *Serial Peripheral Interface* (SPI).

3.5 Integração com o controlador Arduino

O Arduino Nano é um placa de prototipagem compacta baseada no processador *ATmega328*. É alimentado por ligação mini-usb (onde também são carregados os programas a executar). A placa é de pequenas dimensões e de baixo consumo o que a torna bastante conveniente para este projeto.

Para além disso, esta placa contém 6 pinos PWM I/O de um total de 14 pinos digitais I/O, tal como 8 pinos analógicos. O processador trabalha a uma frequência de 16MHz e tem uma memória flash de 32kB, que em principio é suficiente para o programa que será usado neste trabalho.

O Arduino permite fazer a controlo dos vários elementos descritos nesta secção e as ligações entre o controlador e os diferentes módulos estão apresentadas nas tabelas da Figura 2.

RFM95	Arduino
DI02	D4
DI01	D3
DI00	D2
MISO	D12
MOSI	D11
SCK	D13
nSS	D6
RST	D5
GND	GND
VCC	3V3
Moisture Sensor	-
VCC	5V
GND	GND
Control	A6

DHT11	Arduino
Pin 1	5V
Pin 2	D9
Pin 4	GND
Water Pump Circuit	-
Digital Pin	D7

Figura 2: Tabela com as ligações do Arduino para os elementos do resto da montagem.

4 Software

4.1 Arduino

Na placa Arduino, para além das bibliotecas padrão, importou-se a biblioteca IBM LMIC (LoRaMAC in C) adaptada para Arduino [4], compatível com o modulo LoRa RFM95 e com as especificações das *gateways* suportadas pela comunidade de *The Things Network* (TTN). Esta biblioteca contém as funções que tratam todo o processo de envio (modulação, potência, frequência, etc.) e receção de sinais por parte do módulo, tal como toda a comunicação entre este e o micro-controlador..

4.1.1 Envio e receção de dados

Depois de cada transmissão, o envio de dados seguinte é agendado com indicação da função a ser ativada (*do_send()*) pelo disparo do temporizador (que espera um intervalo de tempo TX_INTERVAL) que procede à recolha de dados e à sua transmissão, Figura 3.

```
// Schedule next transmission
os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
```

Figura 3: Função de calendarização de transmissão da biblioteca LMIC

A biblioteca também define a estrutura (*struct lmic_t*) onde são guardados os dados recebidos pelo módulo, a posição onde começa a informação e o seu tamanho.

Para a aquisição de dados do DHT11 e comunicação por Serial com o Arduino usa-se biblioteca DHT-Sensor-Library [3]. O sensor SEN0114 não necessita de nenhuma biblioteca adicional já que a aquisição de dados é feita diretamente pela entrada analógica a que o sensor estiver ligado.

O micro-controlador segue o esquema de trabalhos representado na Figura 4 em que o parâmetro *Timeout* é o período de amostragem de dados, controlado pelo utilizador.

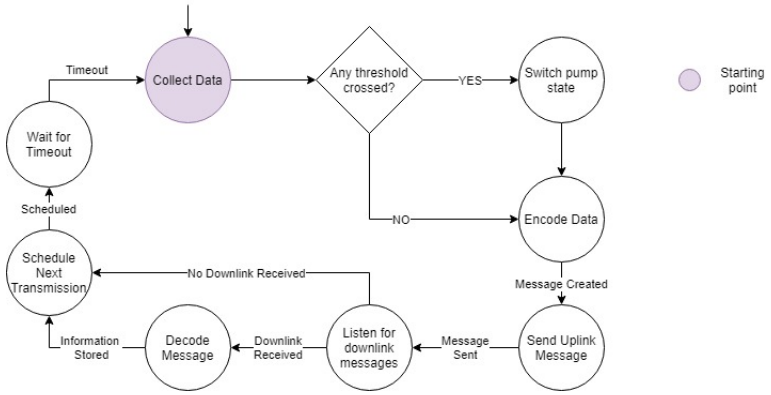


Figura 4: Fluxograma de trabalho do Arduino

4.2 Interface *The Things Network*

O serviço oferecido pelo *The Things Network* (TTN), permite a receção e armazenamento das mensagens enviadas (*uplink*) pelo módulo RFM95. Para além disso, a plataforma permite ainda enviar mensagens de volta por downlink para o módulo RFM95, o que será conveniente no envio de ordens e parâmetros a partir da aplicação *Android*.

Esta plataforma tem um sistema muito conveniente de *Encoding* e *Decoding* em Javascript de mensagens que recebe, conforme sejam provenientes de *downlink* ou *uplink*, respetivamente. Quando as mensagens *uplink* em formato hexadecimal vindas do Arduino chegam ao TTN são convertidas numa mensagem em formato JSON para ser enviada para o servidor HTTP (Decoding). Como a este nível não há preocupações do consumo energético, não há necessidade de enviar mensagens pequenas entre estes dois servidores e por isso utiliza-se o objeto JSON por ser mais conveniente. Pela mesma lógica, o Android envia mensagens em formato JSON que são transformadas no TTN para uma mensagem de apenas 2 Bytes (Encoding). O sistema de mensagens desenvolvido é discutido na secção 5

Como a plataforma *The Things Network* não oferece uma forma de guardar as mensagens durante um período extenso (7 dias no máximo), foi necessário arranjar uma forma de direcionar os dados recebidos para outro servidor (secção 4.3) que tratasse desta tarefa.

4.3 Servidor HTTP

Para fazer a ponte entre o serviço do *The Things Network* e um sistema de registo de dados foi desenvolvido um servidor HTTP que tem o propósito de não só encaminhar dados do *TTN* para a Base de dados, como também de funcionar como uma REST API para aceder ao seu conteúdo. O servidor está a correr na plataforma *Heroku* e pode ser acedido pelo link: `agricomm.herokuapp.com`.

Na página inicial do website é possível ver um histórico de todas as mensagens que já foram enviadas para a base de dados. A REST API desenvolvida permite fazer pedidos *GET* de vários parâmetros da base de dados, nomeadamente:

- **agricomm.herokuapp.com/data**: retorna, em formato *json*, todo o conteúdo da base de dados;
- **agricomm.herokuapp.com/data/:parametro**: retorna, em formato *json*, todo o registo do parâmetro introduzido (que pode ser: temperature, humidity, moisture ou pump) e os respetivos instantes em que a informação foi obtida;
- **agricomm.herokuapp.com/data/:parametro/:intevalo**: retorna, em formato *json*, o registo do parâmetro introduzido (que pode ser: temperature, humidity, moisture ou pump) e os respetivos instantes em que a informação foi obtida, durante um intervalo de tempo (que pode ser: last_year, last_month, last_week ou last_day);

- **agricomm.herokuapp.com/device/:deviceid/data/:parametro/:intevalo**: retorna, em formato *json*, o registo do parâmetro introduzido (que pode ser: temperature, humidity, moisture ou pump) e os respetivos instantes em que a informação foi obtida, durante um intervalo de tempo (que pode ser: last_year, last_month, last_week ou last_day) para um certo identificador do dispositivo.

Decidiu-se fazer uma REST API por ser uma forma simples de alcançar a informação e que pode ser usado tanto por uma aplicação Android como por um serviço de *Home Assistant*, caso o utilizador assim o deseje. O servidor foi desenvolvido em Node.JS, uma framework de *Javascript* própria para o desenvolvimento de *Backend* de aplicações.

4.4 Base de Dados

Guardar dados durante períodos de tempo mais longos permite fazer análises mais elaboradas e por isso decidiu-se utilizar uma base de dados externa ao TTN para guardar as mensagens, já que este só guarda durante um intervalo máximo de 7 dias. A base de dados *mySQL* está alojada no serviço Sigma do Instituto Superior Técnico.

Os campos que são guardados na tabela da base dados são: o identificador do aparelho (para o caso de querermos correr vários aparelhos em simultâneo), a temperatura, a humidade do ar, a humidade do solo, o estado da bomba de rega (ligada ou desligada) e um timestamp com o momento em que a mensagem foi enviada. A descrição detalhada da tabela pode ser observada na Figura 5.

Field	Type	Null	Key	Default
devID	varchar(255)	YES		NULL
temperature	decimal(5,2)	YES		NULL
humidity	decimal(5,2)	YES		NULL
moisture	decimal(5,2)	YES		NULL
pump	tinyint(1)	YES		NULL
arrived_time	datetime	NO	PRI	0000-00-00 00:00:00

Figura 5: Descrição da tabela utilizada na Base de dados *MySQL*.

As operações na base de dados estão prevenidas para tentativas de ataques de segurança como *SQL Injection*.

4.5 Aplicação Móvel para Android

4.5.1 Design e Objetivos

A aplicação móvel para Android permite controlar e definir o funcionamento do sistema em tempo real. Com a aplicação é possível definir os limites pelos quais a bomba de água é ativada, isto é, a partir de qual valor temperatura e humidade deve ser usado o atuador, e também definir o período de amostragem desses valores. É ainda possível observar graficamente o estado do sistema, com a possibilidade de filtrar os resultados por tipo (temperatura, humidades do ar ou solo e estado da bomba), num espaço temporal (último dia, semana, mês ou ano).

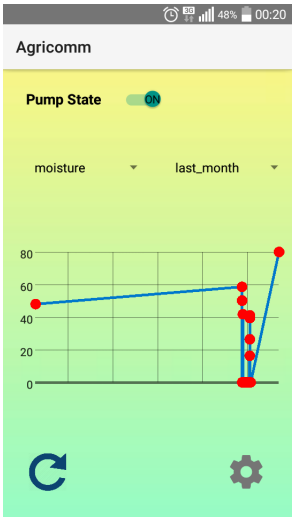
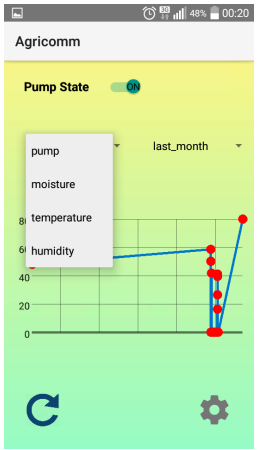
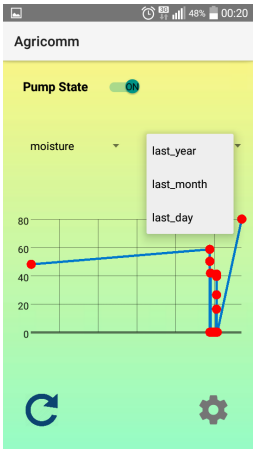


Figura 6: Main Activity

A aplicação apresenta duas atividades, a principal (figura 6, *Main Activity*), permite ao utilizador visualizar e controlar o gráfico do estado atual do sistema através de dois menus *DropDown* onde se escolhe o filtro de resultados desejado, figuras 7a e 7b, o gráfico pode ser atualizado, mantendo os filtros, pelo botão de *refresh* do canto inferior esquerdo [1].



(a) Tipo de dados



(b) Período dos dados

Figura 7: Dropdowns usados para recolha de dados

O eixo Y do gráfico mostra os valores medidos (por exemplo temperatura) e o eixo X, a data em que foi feita essa medição, clicando em qualquer ponto do gráfico é possível observar os valores exatos, figura 8. O *switch* no topo da atividade controla manualmente o estado da bomba (ativa ou desativa) e a sua posição está de acordo com a ultima leitura do estado da bomba. O último botão, do canto inferior direito, ao clicar, muda para a segunda atividade (*settings*).

[1] Alguns valores mostrados no gráfico não correspondem medições reais, mas sim a testes efetuados durante o desenvolvimento do projeto.

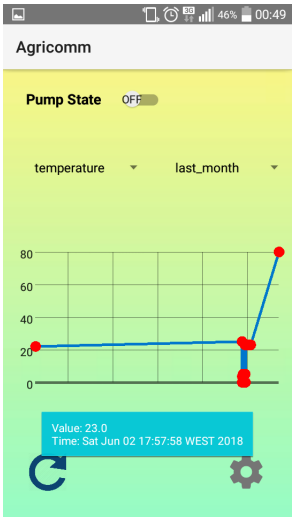


Figura 8: Mostra de valores exatos após clique em ponto do gráfico

A segunda atividade, *settings* figura 9, permite ao utilizador a definição dos limites de atuação do sistema e frequência de amostragem de dados, com quatro caixas de textos correspondentes a temperatura, humidade do ar, humidade do solo e período de amostragem, cada caixa tem a seu lado o botão que permite o envio do valor inserido nela. Há também um botão *home* no canto inferior direito que permite voltar à atividade principal.

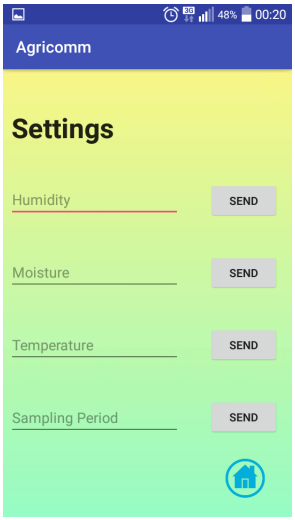


Figura 9: Actividade *Settings*

4.5.2 Bibliotecas e Modo de Funcionamento

O envio de dados *threshold* definidos pelo utilizador na actividade *settings* ou o estado da bomba na actividade *Main Activity* é feito pelo método *HTTP POST* ao servidor The Things Network, que contém uma integração HTTP.

```
https://integrations.thethingsnetwork.org/ttn-eu/api
/v2/down/my-app-id/my-process-id?key=ttn-account-
v2.secret
```

Figura 10: URL oferecido pela TTN para pedidos HTTP

Esse pedido é feito para o link exibido na figura 10 enviando no seu corpo a informação do aparelho correspondente e os dados que se querem enviar por downlink ao micro-controlador seguindo o formato JSON e os campos definidos pela The Things Network, figura 11.

```
Copy
{
  "dev_id": "my-dev-id", // The device ID
  "port": 1,             // LoRaWAN FPort
  "confirmed": false,    // Whether the downlink should
be confirmed by the device
  "payload_fields": {    // The JSON object to be
encoded by your encoder payload function
    "on": true,
    "color": "blue"
  }
}
```

Figura 11: Exemplo, disponibilizado pelo TTN do formato requerido, para o envio de downlinks

A requisição de dados do sistema é feito ao servidor *Heroku* pelo método *HTTP GET*, em que toda a informação necessária para o pedido segue no URL, como discutido na secção 4.3. Após a receção da resposta, uma lista de objetos em formato JSON, constrói-se o gráfico em que cada ponto corresponde a um objeto que contém um valor medido (eixo Y) e o momento da medição (eixo X).

Para além de todas as bibliotecas utilitárias incluídas no ambiente *Android Studio*, são as mais importantes *java.net* [10], para os métodos HTTP, *org.json* [11] para o processamento de dados em formato JSON e *com.jjoe64.graphview* [9] para exibição e controlo do gráfico.

4.6 Compatibilidade do sistema para múltiplos dispositivos ligados

Tendo em conta a sua arquitetura, o sistema criado é facilmente adaptável para funcionar com vários dispositivos. Ter uma rede de dispositivos *IoT* é de grande interesse porque permite controlar os vários nós da rede independentemente e também obter dados de várias localizações em simultâneo, o que garante um conhecimento mais geral do que está a acontecer, por exemplo, num jardim de maiores dimensões.

Para este projeto ser totalmente compatível com vários dispositivos, a única alteração que teria que ser feita seria na aplicação *Android*. Uma vez que a base de dados guarda o identificador do dispositivo que enviou a respetiva informação, seria necessário ter um *Activity* adicional que permitisse escolher o dispositivo do qual se pretende obter os dados e assim fazer um pedido *GET* adequado ao servidor *HTTP* para os obter.

5 Comunicação

Este tipo de sistemas são caracterizados pelo baixo custo, o que implica uma necessidade de reduzir o consumo energético. Todas as comunicações envolvem gasto de energia, logo é importante reduzir o número de comunicações e a sua duração de forma a minimizar esse consumo. Neste sistema há necessidade de comunicação entre o micro-controlador e um servidor, para acesso e controlo pelo utilizador.

5.1 Forma de comunicação

As comunicações estão assentes no protocolo LoRaWAN [6], que permite a troca de mensagens por via de ondas rádio com frequência dentro da banda espectral *Industrial, Scientific and Medical (ISM)*, que na Europa opera entre 863 e 870 MHz. Todo o processamento de baixo nível (escolhas de potência, modulação, frequência, etc..) necessário para operar o módulo LoRa é feito recorrendo à biblioteca LMIC que aplica o protocolo LoRaWAN em conformidade com as gateways usados pelo *The Things Network*.

Apenas há a necessidade de definir o protocolo de mensagens enviadas entre a unidade de processamento e o servidor TTN.

5.2 Unidade de processamento - Servidor TTN (Uplink)

Tendo sempre como objetivo a minimização do consumo energético, são enviadas mensagens com uma periodicidade definida pelo utilizador, com o estado atual do sistema, isto é, temperatura, humidades relativas do ar e do solo, e estado da bomba de água (ON/OFF). Esta informação está comprimida no formato hexadecimal numa única mensagem de 6 bytes. As mensagens seguem o formato da Figura 12.

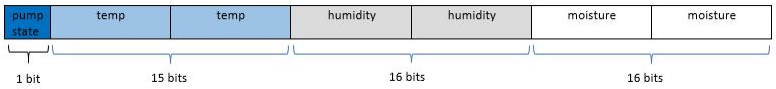


Figura 12: Formato padrão da mensagem *Uplink*, com 6 bytes

5.3 Servidor TTN - Unidade de processamento (Downlink)

O utilizador tem a possibilidade de configurar limites de funcionamento do sistema, daí a necessidade de enviar mensagens por meio do sistema TTN para a unidade de controlo. As mensagens enviadas por *downlink* contêm dois campos, um com uma etiqueta para o parâmetro a alterar e outro com o valor para o qual será atualizado. Os parâmetro atualizáveis são: *thresholds* de temperatuda, humidades do solo e água; estado da bomba (On/Off); frequência de amostragem de dados.

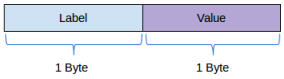


Figura 13: Formato da mensagem *Downlink*.

Os valores são codificados de forma a serem enviados por dois bytes, Figura 13, em que um byte tem a label do campo a ser alterado e o segundo byte o valor a ser usado. Após a receção da mensagem o microcontrolador retira o valor do primeiro byte e conforme a letra correspondente (valor ASCII) atualiza o campo pedido. O protocolo segue a tabela 1.

Label	ASCII	Valor a alterar
p	112	Estado da bomba
t	116	Threshold temperatura
m	109	Threshold humidade do solo
h	104	Threshold humidade do ar
i	105	Periodo de amostragem

Tabela 1: Protocolo de mensagens Downlink

Devido ao forte impacto destas mensagens no funcionamento do sistema, é necessário garantir a sua entrega por meio de um *acknowledgment* (ACK).

5.4 Limitações das comunicações

Devido ao uso de frequências de livre acesso, há limitações regulamentais para o uso destas. As limitações variam entre 0.1% e 10% do *duty-cycle*, para além dessa limitação a rede *The Things Network* estabelece um limite justo de acesso de 30 segundos de *uplink air time* por dispositivo por dia, e de 10 mensagens downlink por dispositivo por dia. De forma a cumprir as limitações as mensagens enviadas em uplink são os mais pequenas possível e não requerem *acknowledgment*, dado que a perda de uma mensagem não quebra o funcionamento normal do programa. As únicas mensagens de downlink enviadas são para definição de limites de funcionamento, logo é expectável que a sua frequência seja baixa.

Dentro do protocolo LoRaWAN usa-se a classe A [12]. Esta classe permite comunicações bi-direcionais, contudo o envio de mensagens downlink pela gateway só é feito imediatamente a seguir à receção de uma mensagem uplink, isto porque há duas janela de receção de mensagens por parte do módulo LoRa abertas apenas após um envio (duas janelas de receção para prevenir atrasos de propagação). As classes B e C permitiriam um envio downlink mais dinâmico, contudo o módulo teria um maior consumo energético devido a uma elevada frequência de leitura do meio. Neste projeto não há a necessidade de ouvir constantemente o meio, dado que o sistema não é critico e a demora da receção de mensagens não compromete o seu funcionamento, diminuindo assim o consumo.

6 Discussão de resultados

6.1 Análise critica da solução

A solução implementada permite ao utilizador o controlo desejado sobre um sistema de agricultura e a sua automatização. Contudo, podem haver formas de melhorar o serviço e deve-se avaliar a solução apresentada.

6.1.1 Consumo energético

Ao nível do consumo, é possível baixar ainda mais o consumo energético. Dado que se está a utilizar uma placa de prototipagem Arduino, uma grande percentagem do consumo dessa placa não está a ser utilizado pelo sistema implementado. Para uma utilização desta solução em larga escala e com elevada autonomia seria preferível desenvolver um sistema embebido dedicado apenas às funções desejadas. Para além disso poder-se-ia utilizar modos *Sleep* para reduzir o consumo energético e só acordar o sistema quando é preciso responder a certas interrupções como o envio de uma mensagem.

6.1.2 Conectividade e comunicação

Neste projeto usa-se como forma de comunicação e transmissão de mensagens o protocolo LoRaWan, num ambiente real para largos hectares de terreno há que avaliar se essa solução mantém a fiabilidade, para isso testou-se o alcance de transmissão de módulos LoRa com os resultados apresentados na Figura 14.

Figura 14: Alcance empírico do módulo LoRa.

Apesar da grande vulnerabilidade da qualidade do sinal quando enfrenta muitos obstáculos[2] (como paredes), num ambiente real e livre, conclui-se que, que o alcance de uma *gateway LoRaWan* é suficiente e que um bom posicionamento da mesma pode cobrir vários quilómetros quadrados de terreno, o que oferece boa conectividade. Uma vez que os as transmissões podem ter ritmos baixos e são pouco frequentes, esta solução terá um custo mais baixo que uma rede celular.

A maior desvantagem desta tecnologia será a dependência de redes livres como o *The Things Network* que apesar de oferecer algumas condições de funcionamento, não tem a garantia de cobertura como teria um serviço do tipo *SigFox*.

6.1.3 Interface do utilizador

A aplicação desenvolvida para o *smartphone* responde aos requisitos, é de fácil interação e leve. Numa solução deste tipo poderia ser útil adicionar ainda mais controlo sobre o sistema, por exemplo sobre a luminosidade (no caso de estufas) ou vídeo-acompanhamento. Para garantir uma utilização segura da aplicação também seria importante fazer um sistema de autenticação.

6.2 Próximos passos

Apesar do objetivo do projeto ter sido cumprido, considera-se que existe grande margem para melhorar esta ideia, nomeadamente em torná-la mais inteligente. Utilizando conceitos de *Machine Learning* e análise de dados meteorológicos, poder-se-ia fazer um controlo da bomba de forma mais adequado, como por exemplo, não ligar a bomba de rega se a previsão meteorológica disser que no dia seguinte irá chover. Outra forma de tornar o sistema mais inteligente seria também ter em conta o tipo de flora do terreno no qual o Sistema IoT está.

A nível tecnológico seria interessante experimentar utilizar um dos serviços de *Real-Time Databases* que são atualmente fornecidos por empresas como a Google ou a Amazon. Estes serviços para além de uma base de dados, que guardam uma certa quantidade de informação gratuitamente, também permitem fazer avisos aos vários dispositivos conectados quando existe nova informação.

[2]O marcador vermelho na Torre Norte coincide com a Sala de Computadores de Engenharia Eletrotécnica. Apesar de a nível de localização este sitio ser muito perto da antena da *Gateway*, como existem muitos obstáculos entre estes verificou-se que não se conseguia fazer chegar mensagens, o que mostra que esta tecnologia não é a mais adequada para sistemas no interior de edifícios.

Referências

[1] D-Robotics, *DHT11 Humidity & Temperature Sensor*, em `http://www.micro4you.com/files/sensor/DHT11.pdf`

[2] DFRobot, *Moisture Sensor Fabricator Page*, em `https://www.dfrobot.com/wiki/index.php/Moisture_Sensor_(SKU:SEN0114)`

[3] Arduino, *Biblioteca do DHT11 para Arduino*, em `https://playground.arduino.cc/Main/DHT11Lib`

[4] Matthijs Kooijman, *Lora Library that runs in Arduino Environment*, em `https://github.com/matthijskooijman/arduino-lmic`

[5] The Things Network, *Node.JS Application SDK for The Things Network*, em `https://www.thethingsnetwork.org/docs/applications/nodejs/`

[6] *LoRa Alliance^{TM} Technology*, em `https://www.lora-alliance.org/technology`

[7] *RFM95 datasheet*, em `http://www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf`

[8] *LoRa^{TM} Modulation*, em `https://www.semtech.com/uploads/documents/an1200.22.pdf`

[9] *Jonas Gehring*, em `http://www.android-graphview.org/`

[10] *Biblioteca java.net*, em `https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html`

[11] *Biblioteca JSON*, em `https://stleary.github.io/JSON-java/`

[12] *LoRaWan especificações*, em `https://lora-alliance.org/sites/default/files/2018-04/lorawantm_-specification_-v1.1.pdf`