Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

SI - Sistemas Autónomos

Sensorização e Actuação em Ambientes

Grupo 2
André Gonçalves, A75625
Miguel Miranda, A74726
Rogério Moreira, A74634
Tiago Sá, A71835

Braga, 24 de Abril de 2018

Conteúdo

1. Introdução

Tendo por base os fundamentos da *Internet of Things*, surge com esta rede a possibilidade de conectar um conjunto enorme de objetos do quotidiano. Estes dispositivos podem encontrar-se, por exemplo, em *smartphones*, veículos, eletrodomésticos ou sensores integrados nos mais diversos sistemas informáticos.

O grande foco da *IoT* encontra-se exatamente na comunicação destes sensores para com a rede e entre si, gerando uma enorme troca de informações, praticamente em tempo real. Estando a *IoT* inevitavelmente envolvida em diversos contextos do dia a dia, torna-se relevante integrar estas técnicas de monitorização/sensorização com atuadores.

No contexto deste projeto, os dados em análise são recolhidos através de um sensor de leitura de *RFID tags*. Como estrutura do relatório, o capitulo 3 descreve o conjunto de procedimentos necessários para recolher informações de sensores *RFID*, focando ainda de que forma é possivel criar ações deliberativas com o tratamento posterior dos dados recolhidos; o capitulo 4 foca a partilha das informações recolhidas na rede, através da plataforma *AdaFruitIO*; o capitulo 5 analisa a plataforma IFTTT, apresentando a criação de regras que interligam os dados enviados para a *IoT*, através do *AdaFruitIO*, com o desempenho de ações predefinidas no ambiente.

Por fim, o capitulo 6 apresenta as conclusões finais do trabalho desenvolvido.

2. Motivação

Muitos dos equipamentos que suportam a *IoT* são capazes de controlar e atuar num determinado contexto do ambiente onde se inserem, de forma autónoma e "inteligente". Esta atuação pode ser algo tão simples como controlar a luminosidade de uma sala, com base em parâmetros como as medições de um sensor de luz, a hora do dia e a presença, ou não, de pessoas na divisão.

Contudo esta interpretação e ação num ambiente pode ir além de arquiteturas reativas. Integrando estes sistemas de sensorização/atuação em módulos deliberativos ou híbridos, é possível analisar os comportamentos e padrões de uso, "personalizando"os serviços desempenhados.

Nestes contextos, a atuação não é apenas uma análise de informação e relações entre parâmetros registados pelos sensores, mas uma aquisição de conhecimento do meio, através da interiorização de padrões de utilização do mesmo.

Numa vertente comum e reativa, a identificação de objetos ou pessoas com base em *RFID tags* permite, por exemplo, controlar o roubo de objetos em lojas de vestuário ou, o desbloqueio de torniquetes quando um identificador válido é lido.

Contudo, através de uma análise mais profunda dos registos é possível extrapolar ações/notificações mais complexas. Por exemplo, caso um sensor leia várias vezes o mesmo id num curto espaço de tempo, pode avaliar a presença de uma possível tentativa de fraude ou extravio do cartão a outros indivíduos, com base nas horas e no historio de utilização daquela *tag*.

As vantagens destes sistemas são assim imensas e associadas aos mais diversos contextos. Sendo que atualmente as plataformas que permitem a integração com IoT são extremamente simples de utilizar, justifica-se ainda mais a exploração de agentes autónomos nestes ambientes monitorizados.

3. Recolha Dados do Sensor

Como referido anteriormente, o desenvolvimento deste projeto baseia-se na utilização e exploração de um sensor capaz de ler as informações contidas em tags *RFID*.

Estes identificadores armazenam no seu circuito uma string única, permitindo assim identificar o objeto ou indivíduo quando a sua tag é registada pelo sensor. Vulgarmente esta tecnologia encontra-se em cartões, identificação de animais, controlo de inventários em lojas ou segurança no acesso a determinados locais, desbloqueando o acesso a pessoas com tags autorizadas.

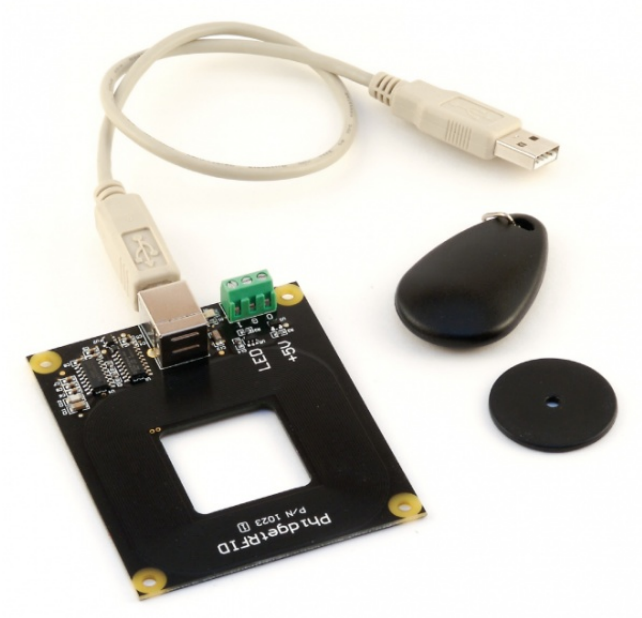

Figura 3.1: Exemplo de um sensor de leitura de tags *RFID*.

O sensor utilizado, semelhante ao da figura 3.1, encontra-se associado à empresa *Phidgets*. Apesar de proprietária, a companhia oferece todas as ferramentas necessárias para operar e integrar estes equipamentos em qualquer projeto.

Para tal, é necessário [1]:

1. Realizar o *download* da biblioteca que permite reconhecer e instalar os drivers das placas dos sensores, sendo que existem instaladores para qualquer tipo de sistema operativo;

2. Selecionar a linguagem de programação sobre a utilizar.

   A plataforma permite programar os equipamentos em C, C#, Python e Java, sendo neste último ambiente que se realiza este estudo.

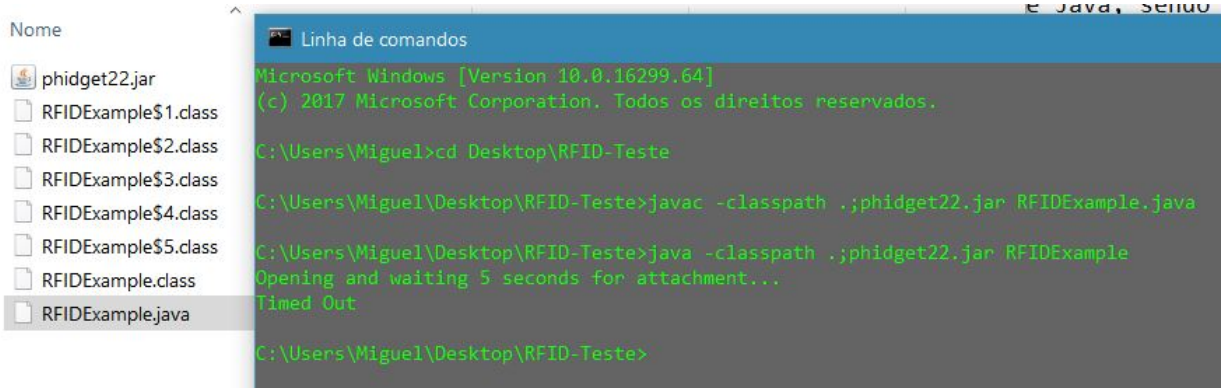

Figura 3.2: Execução da versão inicial do Applet que monitoriza os eventos do sensor RFID.

```
INFO [phidget22][2018-04-24T15:18:10 usbmac.c+287 PhidgetUSBOpenHandle()]: Using Control Endpoint for Host->Device communication.
INFO [phidget22][2018-04-24T15:18:10 usbmac.c+291 PhidgetUSBOpenHandle()]: Attach: 0x00006803
INFO [phidget22][2018-04-24T15:18:10 usb.c+397 PhidgetUSBReadThreadFunction()]: PhidgetRFID(1023) (6964): ReadThread starting
INFO [phidget22][2018-04-24T15:18:10 rfiddevice.c+563 tagTimerThreadFunction()]: tagTimerThread running
channel 0 on device 6964 attached
Tag read: 01022f89ca
Tag lost: 01022f89ca
Tag read: 550011ed25
Tag lost: 550011ed25
Tag read: 01022f89ca
Tag lost: 01022f89ca
```

Figura 3.3: Exemplo de valores lidos pelo sensor, utilizando cartões com RFID.

3. Como forma de inicialização, a plataforma oferece ainda um conjunto de projetos exemplo, para cada tipo de equipamento disponibilizado.

   Neste caso foi feito o download do projeto *RFID* com a opção "Multiple"no sistema operativo e do pacote *jar* com a API do *Phidget*. Esta API é normalmente genérica partilhando o mesmo nome para os métodos e eventos, independente do dispositivo/sensor utilizado.

4. Colocando o ficheiro jar e a classe java descarregados na mesma diretoria, é apenas necessário compilar e executar o programa criada.

   Novamente, todo o processo encontra-se explicitamente documentado e ilustrado no site da plataforma *Phidget*.

   A figura 3.2 demonstra a tarefa de execução do programa base que controla os eventos do sensor.

   Por omissão, o código descarregado espera 5 segundos para conectar/reconhecer o sensor e 10 segundos para receber informações do sensor, através da passagens de tags *RFID*. No exemplo da figura o processo apresenta timeout uma vez que nenhum sensor estava conectado ao computador (já tinha sido devolvido à equipa docente).

Sempre que se aproximar um RFID tag do sensor, é acionado um evento que permite reconhecer a sua presença e recolher as suas informações (Figura 3.3). No caso desta tecnologia e, como já referido anteriormente, cada tag regista em si uma string única que a identifica inequivocamente.

3.1 Criação de ações Deliberativas

No excerto de código seguinte é apresentada uma forma de registar a tag lida pela sensor, mantendo um histórico simples de todas as medições utilizadas, registando o número de ocorrências e hora de cada registo.

Para uma utilização efetivamente útil destas informações, podem ser criados métodos que analisem estes registos ao longo de um período de tempo maior, observando assim padrões de utilização por parte dos utilizadores, como a sua rotina ou assiduidade de acesso ao serviço que requer o uso da RFID tag.

```
// Inicialização de um canal para receber informações
RFID ch = new RFID();

// variável para armazenas todos os registos lidos pelos
↪  sensor
HashMap<String, ClientTAG> clientes = new HashMap<String,
↪  ClientTAG>();

    // (...)

// Evento disparado caso seja identificada uma tag pelo sensor
ch.addTagListener(new RFIDTagListener() {
    public void onTag(RFIDTagEvent e) {
        String tagId = e.getTag(); // get readed tag id
        Long time = System.currentTimeMillis();
        if(this.clientes.containsKey(tagId))
            this.clientes.get(tagId).update(time);
        else{
            ClientTAG new_cli = ClientTAG(tagId, time);
            this.clientes.add(tagId, new_cli);
        }
    }
});

    // (...)

// classe para registar um historico simples de um
↪  identificador
private class ClientTAG{
    private String tagID;
    private int nr_ocorrencias = 0;
    private ArrayList<Long> horas_ocorrencias = new
    ↪  ArrayList<Long>();

    ClientTAG(String tagID, long time){
        this.tagID = tagID;
        this.nr_ocorrencias++;
        this.horas_ocorrencias.add(time);
    }

    public void update(Long time){
        this.nr_ocorrencias++;
        this.horas_ocorrencias.add(time);
    }
}
```

Tendo este registo gradual dos valores lidos pelo sensor é possível criar algumas regras mais complexas. Cada regra exige um comportamento/ação especifico. Por este motivo, sempre que estas situações excecionais se verifiquem, deve ser

enviada em acréscimo uma mensagem com as informações recolhidas para uma outra Feed do AdaFruit, criada especificamente para cada caso. Posteriormente existirá no IFTTT um trigger para cada um desses Feeds adicionais. Exemplos de casos/situações irregulares, no contexto de RFIDs, serão por exemplo:

- O utilizador aceder ao serviço numa hora que não pertence à sua rotina de trabalho;
- O utilizador utilizar o seu cartão um número elevado de vezes num curto período de tempo (10 passagens pelo sensor em menos de 10 minutos). Neste caso o sensor pode assumir uma utilização fraudulenta ou extravio do cartão, estando a ser utilizado por outras pessoas que não o titular;

Como é possível de observar, mesmo para um utilizador que não compreenda profundamente de programação são facilmente compreendidos os métodos que recolhem as informações destes sensores. Novamente, a empresa *Phidget* disponibiliza no seu site uma API com todos os métodos e eventos possíveis de programar sobre as suas placas e sensores.

No capitulo seguinte será abordado de que forma é possível enviar estes dados registados para a *IoT*, através da plataforma *AdaFruitIO*.

4. Serviço *AdaFruitIO*

Depois de descrito como observar e armazenar os registos medidos pelo sensor RFID, este capitulo aborda uma forma simples de partilhar este dados com a rede, num contexto de *Internet of Things*. O envio de informações para a rede permite, por exemplo, que os dados recolhidos sejam trocados com outros equipamentos, que pertencem ao contexto de funcionamento do sensor.

Por exemplo, sempre que um sensor reconheça a saída de um aluno de uma escola, através do seu cartão de identificação, poderia ser acionado um sistema de notificações que instantaneamente avise o encarregado de educação sobre a saída do aluno, indicando a hora do acontecimento.

Esta partilha de informações dentro de um sub sistema da rede, permite assim aumentar o alcance de ação destes equipamentos de sensorizacão e agentes autónomos.

Como plataforma para receber e armazenar as informações na rede, foram utilizados os serviços disponibilizados pelo *AdaFruit*. Este sistema procura ser extremamente fácil de utilizar, permitindo recolher dados através de conexões simples e sem necessidade de programar código complexo. A troca de mensagens com este sistema baseia-se no protocolo de comunicação MQTT (Message Queue Telemetry Transport). Este protocolo foi desenvolvido no contexto de *IoT*, focando a redução do tamanho das mensagens e do *overhead* de comunicação.

A plataforma AdaFruit representa um MQTT Broker, que recebe e armazena as informações enviadas por parte dos sensores. As informações são armazenadas em *Feeds*, que podem posteriormente ser distribuídas para outros equipamentos que subscrevam esses mesmos *Feeds*.

O AdaFruit é assim vantajoso pelo facto de além de ser simples, abstrair todas as tarefas necessárias para construir um servidor e conexões entre os sensores e o nodo onde se armazenam os registos enviados para o Broker.

4.1 Conexão com AdaFruit

O seguinte excerto de código modelo o envio das informações lidas por um sensor para um servidor no serviço AdaFruit.

Para que esta conexão seja possível de acontecer é necessário criar previamente um Feed no AdaFruit, onde se irão armazenar os dados conforme o seu tipo ou intenções do programador.

```
String clientId = "Mikeeee";
String content = e.getTag();
String topic = "MikeAdaFruitIO";
int qos = 0;
boolean retained = false;
String broker = "tcp://io.adafruit.com:1883";
String feed = "/feeds/registocliente";
```

```
MemoryPersistence persistence = new MemoryPersistence();

try {
    MqttClient client = new MqttClient(broker, clientId,
    ↪  persistence);
    MqttConnectOptions connOpts = new MqttConnectOptions();
    connOpts.setCleanSession(true);
    connOpts.setUserName("MikeAdaFruitIO");
    String password = "bd826fb9065a404798aef61c4e4b04b2";
    connOpts.setPassword(password.toCharArray());
    client.connect(connOpts);
    System.out.println("Connected");
    MqttTopic registoTopic =
    ↪  client.getTopic("MikeAdaFruitIO/feeds/registocliente");
    System.out.println("Publishing message: "+content);
    registoTopic.publish(new MqttMessage(content.getBytes()));
    System.out.println("Published data. Topic: " +
    ↪  registoTopic.getName() + "   Message: " + content);
    client.disconnect();
    System.out.println("Disconnected");
    System.exit(0);
} catch(MqttException me) {
    System.out.println("reason "+me.getReasonCode());
    System.out.println("msg "+me.getMessage());
    System.out.println("loc "+me.getLocalizedMessage());
    System.out.println("cause "+me.getCause());
    System.out.println("excep "+me);
    me.printStackTrace();
}
```

Um exemplo do acesso à informação de um destes *Feeds*, será abordado no capitulo 5. Recorrendo ao IFTTT será possivel criar triggers que disparem após a entrada de dados, num determinado *Feed* em "escuta".

5. Criação ações em *IFTTT*

A plataforma *IFTTT (If This Then That)* apresenta-se como um serviço web gratuito, capaz de integrar alterações ou eventos que ocorram em equipamentos, aplicações ou serviços web, como o *AdaFruit*.

Este sistema permite a criação de ações explicitamente condicionais, designadas por *applets*. Um applet é assim desencadeado por eventos que ocorram num nos serviços que se encontra a ser "observado". Quando uma alteração é verificada, é desempenhada a ação especificada na condição.

Considerando que os serviços disponibilizados pelos *AdaFruit* funcionam como uma espécie de servidor capaz de organizar a informação que recebe de sensores em *Feeds* especificas, a plataforma *IFTTT* funciona como uma especie de interface abstrata, capaz de detetar a entrada de novos dados numa Feed à escuta e, caso se verifiquem condições, executar ações.

Estas ações podem ser a partilha/reenvio das informações para outros dispositivos como, a tomada efetiva de ações num determinado equipamento do ambiente.

5.1 Integração com *AdaFruit*

O sistema AdaFruit oferece já um serviço que permite a interligação fácil com o IFTTT, de forma a criar regras que se relacionem com a entrada de valores numa feed. Para tal é necessário:

1. Ter uma conta AdaFruit, com um Feed a receber informações de um determinado sensor, num formato conhecido. Considerando os tópicos abordados no capitulo 4, a criação de uma conta e de um Feed de dados já está realizado;

2. No website do IFTTT deve procurar-se o serviço AdaFruit, que permite interligar as contas das duas plataformas;

3. Uma regra/applet é composta por um trigger (condição do if - "This") e uma ação (tarefa a executar - "That");

4. Tendo os dois serviços interligados, na criação de um novo *applet* será possível escolher o serviço AdaFruit e, em seguida, escolher o trigger "Monitor a feed on AdaFruit IO".

   A interface é de tal forma automatizada que fornece já um conjunto pré criado de opções válidas. Nomeadamente, são listados apenas os nomes das Feeds que existem na conta AdaFruti associada ao IFTTT, agilizando assim a sua escolha.

   No campo relação é indicado o operador sobre o qual se realiza a comparação entre o valor lido da Feed e o valor sobre o qual se quer desempenhar uma ação, podendo ser a condição <, <=, =, >, >= ou !=.

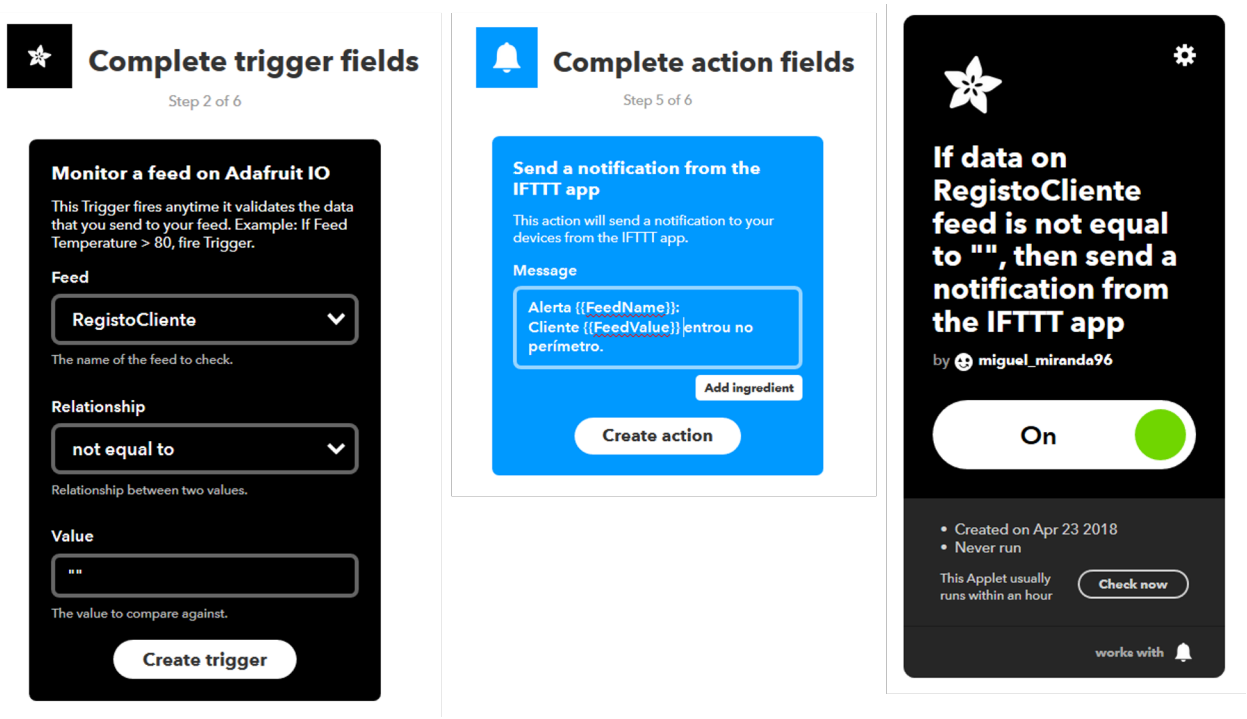

Figura 5.1: Criação de uma regra na plataforma IFTTT, analisando os dados de uma Feed em AdaFruit.

Conceptualmente, o trigger pode ser compreendido da seguinte forma:

```
if(FeedName has new value):
    if (FeedName.FeedValue is {Operator} {TriggerValue}):
        Do_action("that")
```

5. Na figura 5.1 é descrito um exemplo de um trigger para o caso abordado nos capítulos anteriores.

   - Um sensor de RFID envia para uma Feed com o nome "RegistoCliente"todos os identificadores que lê.
   - Sobre essa Feed está à escuta um trigger do IFTTT.
   - Sempre que entrar um valor não nulo, informa um administrador da entrada de um cliente, através de uma notificação no seu smartphone.

6. A ação a desempenhar usa o serviço de notificações do IFTTT, que envia a notificação para os dispositivos com a aplicação instalada e associada à conta em utilização.

   A notificação pode ainda ser personalizada a nível do texto e informações que são partilhadas.

   A figura 5.2 e 5.3 demonstra a entrada de dados no Feed do AdaFruit e a receção quase imediata das respetivas notificações, no dispositivo que se encontra a usar a aplicação com a conta onde foi criada o trigger.

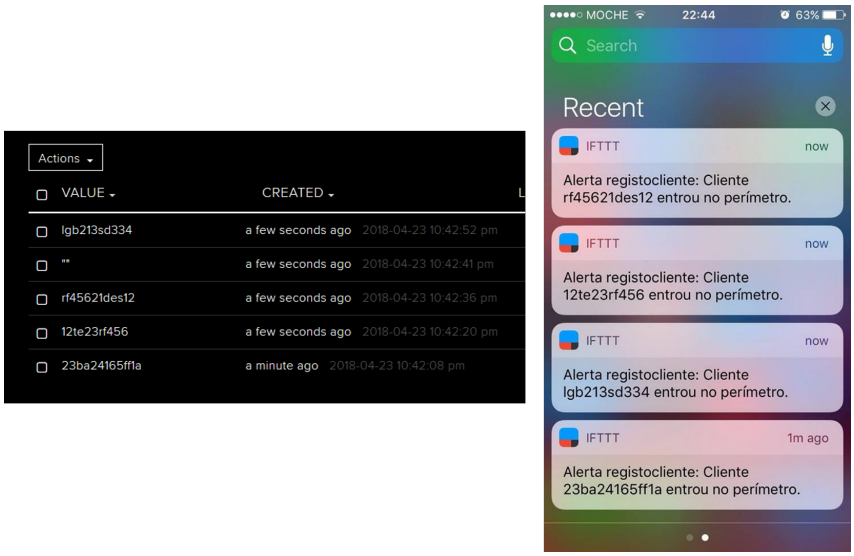

Figura 5.2: Exemplo de notificações recebidas por exemplos inseridos manualmente na Feed à escuta.

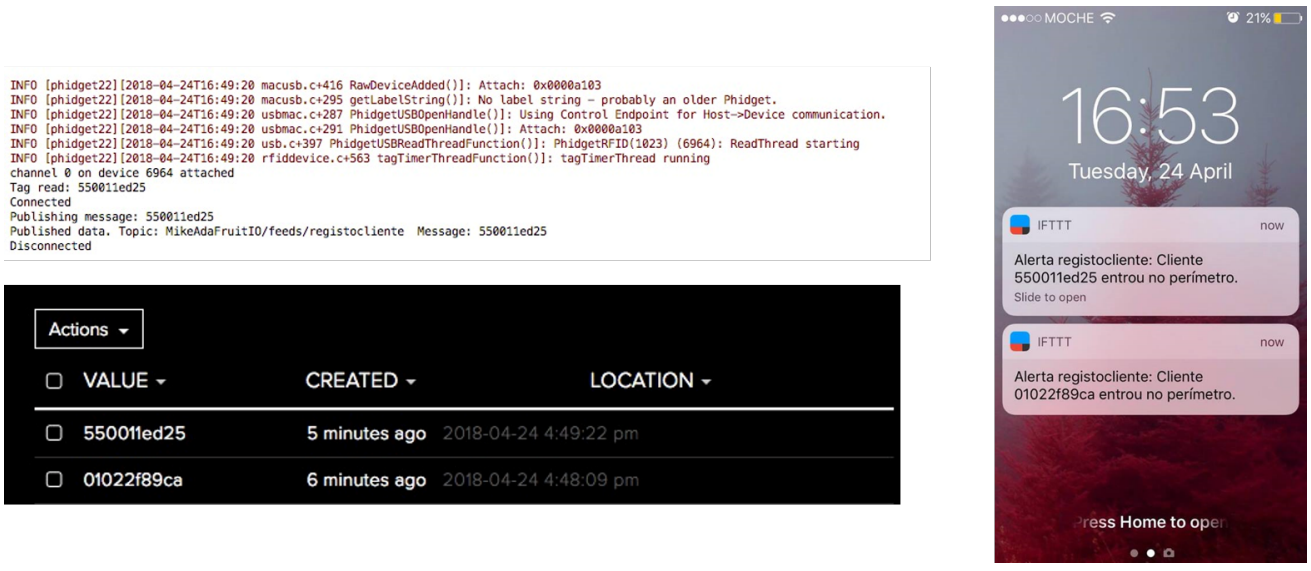

Figura 5.3: Exemplo de notificações recebidas diretamente do sensor, disparando o trigger à escuta do Feed.

6. Conclusão

Estando a *IoT* cada vez mais presente nos dispositivos do dia-a-dia, as vantagens deste domínio tornam-se cada vez mais acessíveis ao utilizadores finais, que utilizam serviços baseados nesta rede. O trabalho desenvolvido neste projeto foca assim um introdução breve aos serviços que permitem de forma simples e rápida programar, gerir e interligar dispositivos baseados em *IoT*.

Num contexto especifico, foram apresentadas regras semi-deliberativas possíveis de construir com base em dados simples, recolhidos através de um leitor de tags RFID. A capacidade de retirar conhecimento e padrões de um registo de dados tão simples revela-se o principal desafio num projeto de monitorização, que procure ser verdadeiramente autónomo e personalizável ao ambiente onde se insere.

Devido à imensa utilização de sensores e serviços baseados em *IoT* no presente, todas as questões relacionadas com um armazenamento de informações num servidor ou acesso às mesmas, por parte de outros dispositivos da rede, revela-se relativamente simples. Respetivamente, os serviços da plataforma *AdaFruitIO* e *IFTTT* oferecem uma interface abstrata e de alto nível, permitindo criar serviços e funcionalidades praticamente sem necessitar de conhecimentos em programação.

Em suma, o presente relatório destaca os passos conseguidos no projeto, levando à criação de um sistema de sensorização e monitorização de um ambiente, baseado em sistemas autónomos.

Bibliografia

[1] Phidget support for windows. `https://www.phidgets.com/docs/OS_-_Windows`. [Online; accessed 23-Abril-2018].