INSTITUTO SUPERIOR TÉCNICO

REDES MÓVEIS E SEM FIOS

Projeto

Development of Internet of Things sensor monitoring based on IEEE 802.15.4, MICAz and Android

Grupo 9

75847 – José Dias

76090 – Diogo Ferreira

Prof. António Grilo

22 de Maio de 2016

Índice

1. Motivação

A *Internet of Things* consiste na interligação de vários sensores à rede de *Internet*. Os dados recebidos de sistemas de baixa potência são enviados e armazenados em servidores permanentemente ligados à rede *(Cloud*) que permitem a mobilidade no acesso por qualquer dispositivo portátil com capacidade de ligação, em particular, dos *smartphones*.

Pretende-se desenvolver um sistema de segurança composto por sensores de movimento que alerte o seu utilizador em caso deteção. No processo, serão aprofundados conhecimentos em redes de computadores, comunicação entre máquinas e funcionamento em rede, aperfeiçoamento e aprendizagem de novas linguagens de programação. Prevê-se um benefício pedagógico pela junção de temas aprendidos na área de Telecomunicações e Redes de Comunicação e em especial pela verificação do objetivo de cada camada do modelo OSI no funcionamento de um sistema de comunicação.

2. Arquitetura

O sistema a implementar consiste na ligação de um conjunto de sensores de movimento à rede de *Internet*. O sistema a implementar pode-se dividir em 3 etapas: aquisição, armazenamento e acesso.

Na aquisição, os dados serão recolhidos e enviados para o servidor via Internet. Será usado o sistema MICAz e MIB520. Este sistema consiste em várias placas com comunicação via rádio (norma IEEE 802.15.4) que correm o sistema operativo orientado a eventos *TinyOS*. Define-se uma para receber os dados e as restantes serão ligadas uma a cada sensor. Os dados recebidos pela estação de base são passados à placa MIB520 através de um conector de expansão de 51 pinos e para o computador via USB. O computador corre o sistema operativo Linux, versão *Ubuntu* ou *Mint*. As tramas são processadas de acordo com a sua origem através de um identificador do nó, e é colocado um carimbo temporal através do computador. Só serão transmitidos dados entre as placas quando à atividade num dos sensores.

Os dados são armazenados num servidor *Web*, implementado em PHP. Este comunicará com o computador do qual receberá a informação relativa aos sensores de movimento.

O acesso aos dados será implementado a partir de uma aplicação que vai ser instalada num *smartphone* ou *tablet* que corra o sistema operativo *Android*, onde será possível verificar o

histórico dos alarmes anteriores ou ouvir com pouco atraso um som caso um dos sensores seja ativado nesse momento. Este som poderá ser modificado consoante a preferência do utilizador.

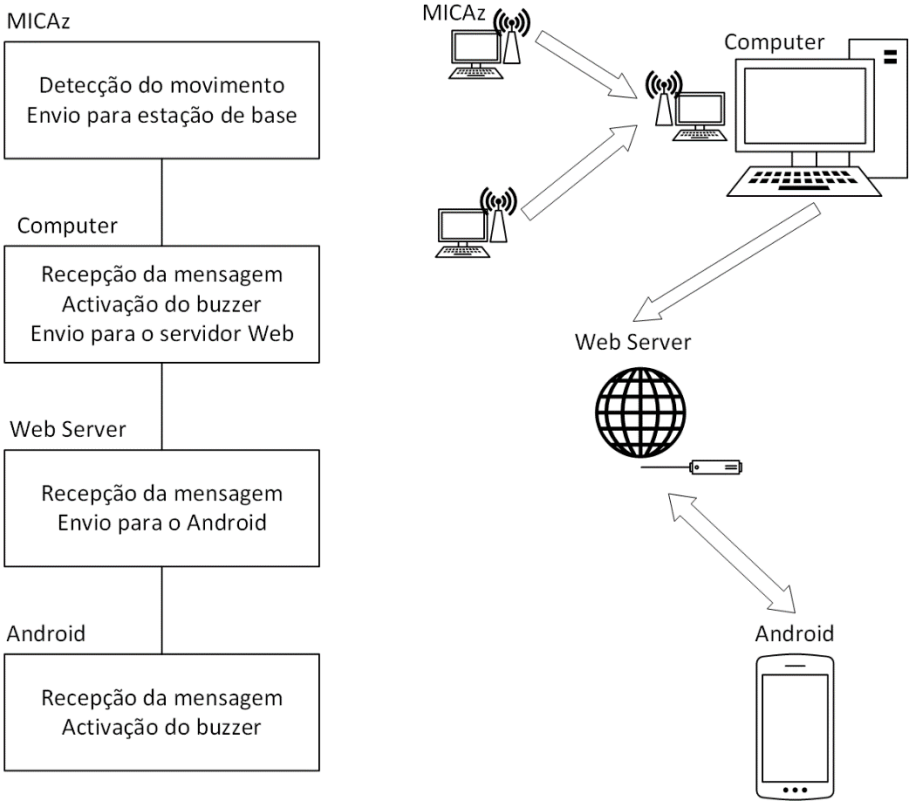

Figura 1 - Arquitetura *high level* do sistema e respetivo fluxograma.

3. Objetivos iniciais

- Familiarização com a linguagem de programação *nesC* orientada a eventos e com o sistema operativo *TinyOS* que corre nas placas MICAz e que fazem a ligação entre o sensor e a estação de base.
- Interpretação das tramas enviadas pela estação de base ao computador (via USB) e implementação de *software* para tratamento automático da informação.
- Implementação de um servidor Web, em PHP, com ligação ao computador associado à estação de base e acessível através de um *browser* e da aplicação, onde serão guardados todos os dados.
- Desenvolvimento da aplicação *Android* para consulta em tempo real dos alarmes.

4. Trabalho Desenvolvido

4.1. MICAz, MTS310, MIB520

No sistema implementado consideram-se 4 tipos de nós com funções distintas. A estação de base, ligada à placa de comunicação (MIB520) com o computador, os sensores de movimento, as sirenes (MTS300) e as sirenes com detetor de som (MTS310). Cada nó é programado de acordo com a sua função e todos os nós têm conhecimento da estrutura das mensagens que trocam via rádio através de um ficheiro *.h e que será descrito na secção dos protocolos.

Cada sensor tem um parâmetro único que o identifica dentro da sua rede – Node ID. Ao programar cada sensor, este parâmetro foi alterado de sensor para sensor ao começar em 1 e incrementar o seu valor.

Utilizou-se uma interface já desenvolvida pelo INESC, *BinarySensor*, para detetar um evento de existência de movimento. Detetado o movimento, usaram-se as interface de comunicação via rádio disponíveis nos exemplos mais simples do *TinyOS* para comunicar a mensagem aos restantes nós. A mensagem segue com identificação do nó que lhe deu origem e deverá ser recebida por sirenes dentro do raio de comunicação do nó e pela estação de base. A mensagem é ignorada pelos restantes sensores de movimento pois só receberão mensagens da estação de base. Para sirenes escondidas do detetor de movimento, a estação de base difunde uma mensagem de alerta de deteção de movimento e que as fará tocar. Para os nós equipados com a placa MTS310, o detetor de som na faixa dos 4 kHz é usado para detetar outras sirenes a tocar.

Idealmente todas as sirenes teriam também detetor de som, ou seja, todas as placas seriam MTS310. Contudo, devido a um problema de interferência entre o módulo de rádio e o detetor de som não foi possível implementar a opção de configuração do efeito de propagação estando este sempre ativo.

O conjunto das sirenes tem um identificador bem definido (100) e igual entre si para que a opção de ligar ou desligar a sirene seja difundida com este identificador e interpretada por cada uma.

4.2. Estação de Base

A estação de base envia e recebe mensagens através do módulo de rádio e através da porta USB. A sua execução está associada a um programa Java que é responsável por fazer a gestão das mensagens que chegam do servidor PHP central e das mensagens que deverão ser enviadas via rádio pela estação de base.

Assim que é iniciado, o programa requer informação ao servidor PHP acerca do estado da rede e das suas configurações internas. Assim que o pedido é respondido, a informação é extraída da *String* é colocada, byte a byte, no vetor que contém o pacote a ser enviado por USB para a estação de base. A estação de base limita-se a difundir este pacote e a alertar cada nó das suas configurações. A cada identificador de nó diferente é dedicada uma mensagem de configurações.

A gestão de pedidos é realizada através de duas *threads*: uma para enviar pedidos periódicos ao servidor de 3 em 3 segundos e receber as configurações dos nós, e outra para enviar alarmes assíncronos dos sensores.

4.3. Servidor PHP

O servidor PHP é implementado de maneira a receber pedidos e a lidar com vários clientes em simultâneo: Utilizadores (*Android*) ou Estações de Base. Toda a comunicação via Internet é realizada através de *Sockets*, em que as mensagens trocadas são *Strings*.

Quando se executa o servidor este cria uma interface de comunicação bidireccional entre processos (*Socket*), seguido de um *Bind* do endereço IP no qual se encontra e do porto desejado e o Listen de modo a receber os clientes. Estes clientes são aceites através do *Accept*, a cada cliente é realizada uma chamada de sistema (*fork*) que cria um novo processo e este novo processo é que realiza a manipulação do cliente que acabou de aceitar. A base do servidor é um conjunto de verificações nas mensagens de protocolo, primeiro para decidir se se trata de um cliente *Android* ou de uma estação de base e posteriormente para decidir que tipo de operação ou *query* executar na base de dados.

4.4. Base de Dados

A base de dados tem 5 tabelas e a sua construção teve como a escalabilidade de todo o sistema sem que fossem necessárias quaisquer alterações à base de dados.

Na tabela Pessoa (*Person*) é possível guardar os dados de cada utilizador, como o seu E-mail (E-mail), nome (*Name*), password *(Password*).

Na tabela PAN é possível guardar o ID de cada PAN (*idPan*) e correspondente Serial Key (Serial_key), guardar se esta está ativa (*Enable*) ou se tem a propagação (Propagation) ou sirene (Buzzer) ativos.

A partir das tabelas anteriores é derivada uma tabela de relação entre elas, que vai associar cada pessoa através do seu e-mail a uma certa PAN definida pelo seu ID, ainda nesta tabela é dada a indicação de Notificação (Notify). Esta indicação é necessária pois vários membros podem aceder à mesma PAN e ser necessário saber quais destes é que já foram notificados pelo seu alarme.

Na tabela Node guarda-se o valor do ID correspondente a cada nó (*idNode*), se este se encontra ativado (*Activated*). Devido a existir uma relação entre os nós e a sua PAN e a esta relação ser de muitos para um (*Many-to-One*) não é criada uma tabela nova mas sim acrescentado um parâmetro à tabela Node com a chave primária correspondente à tabela PAN de modo a identificar a que PAN este nó pertence.

Na tabela NodeReadings guarda-se um *Timestamp*. Pela mesma explicação apresentada para a tabela Node deve ser também acrescentado um parâmetro à tabela *NodeReadings* acerca do nó a que está associado (NodeidNode).

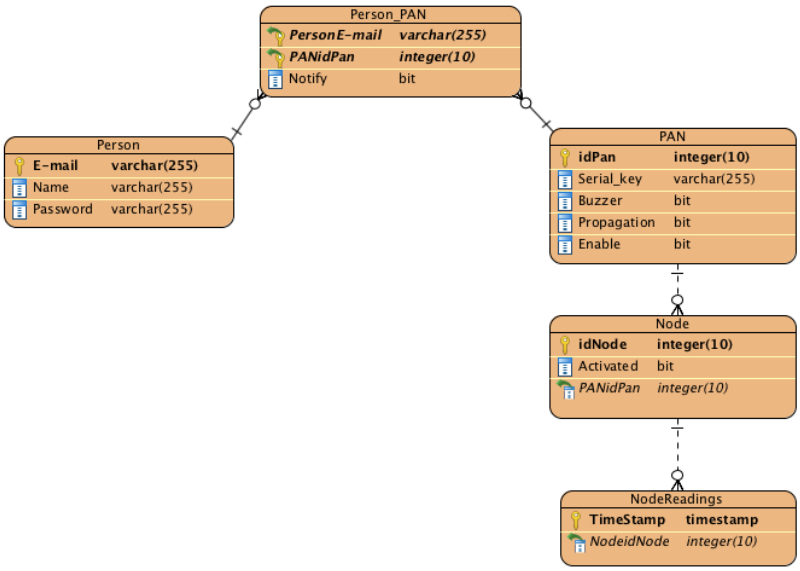

Figura 2 Modelo E-R da base de dados.

4.5. Android

No *Android* desenvolveu-se uma página inicial, sobre o qual se deve realizar o login (*Sign In)* ou realizar o registo (*Register*), este é apresentado na figura 3. Este é implementado de modo a só ser possível o acesso remoto aos dispositivos que pertencem ao utilizador registado que os detém. Independentemente da opção escolhida (*Sign* In ou *Register*) deverá ser dado um endereço IP válido para o servidor Note-se que este campo foi apenas inserido devido aos autores não terem acesso a um servidor com endereço IP constante e público. Em caso de implementação no mercado, seria comprado endereço IP externo e constante e a opção não deveria ser acessível ao utilizador.

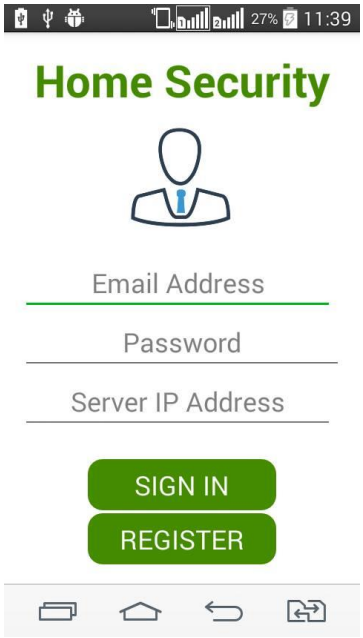

Figura 3 - Página Inicial.

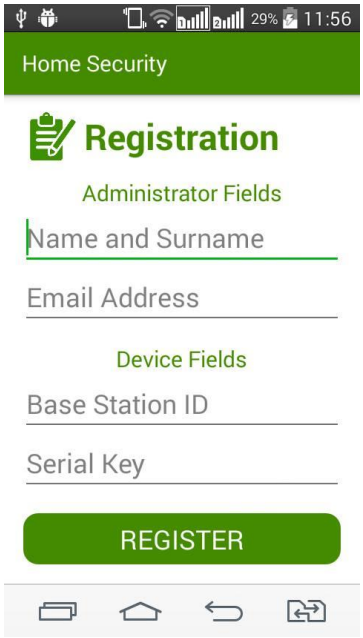

Figura 4 - Página de Registo

Caso a opção escolhida seja de registo, a página é alterada - figura 4. O ID da estação de base e a sua *serial key* são conhecidos pelo utilizador, imaginando a analogia com os routers quando são comprados e contêm algures na sua estrutura um selo com estes parâmetros.

Assim que o utilizador prime o botão de registo é verificado se o ID inserido existe, em caso afirmativo, verifica se corresponde à *Serial Key* indicada. Caso um dos parâmetros esteja incorreto é mostrado um aviso com a indicação de qual dos parâmetros está incorreto. Em caso de sucesso o servidor PHP enviará o *mail* com a *password* de acesso à aplicação.

Caso a opção seja de login, envia-se o login para o servidor. Se este for aceite pelo servidor, a aplicação muda para a página *Home* onde se pode iniciar ou terminar pedidos periódicos de 5 segundos ao servidor para receber notificações de novos alarmes e aceder à página das configurações (*Settings*) ou à página dos registos (*Records*). Esta página está representada na figura 5. Quando o utilizador recebe uma notificação da aplicação, *o smartphone* vibra e toca um som a avisando o utilizador. Se o utilizador carregar na notificação é redirecionado para a página de *Login* da aplicação para que possa, por exemplo, desligar a sirene.

Caso o *login* esteja incorreto surge um *pop-up* com a informação relativa ao campo que se encontra errado.

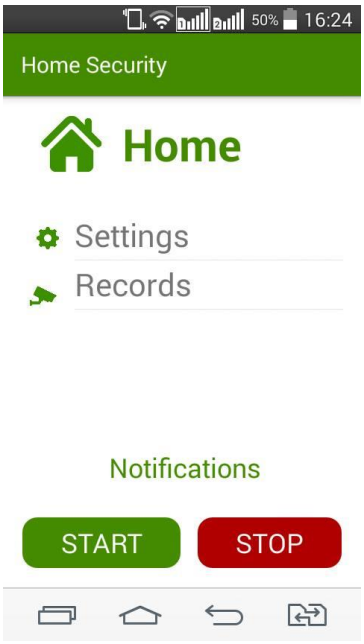

Figura 5 - Página da Casa

Ao entrar nas configurações (*Settings*), é enviado um pedido ao servidor a requerer as configurações dos nós para correcta apresentação da página. Apresentam-se três botões de alternância (*toggle button*) e uma lista dos sensores de movimento associados a uma única PAN. Através dos botões é possível ativar ou desativar o sistema todo (*Enabled*), apenas a sirene (*Buzzer*), ou um sensor de movimento em específico ao carregar por cima do elemento da lista que se pretende ativar ou desativar e que irá alterar a sua cor de fundo caso esteja ativo (Verde) ou descativo (branco).

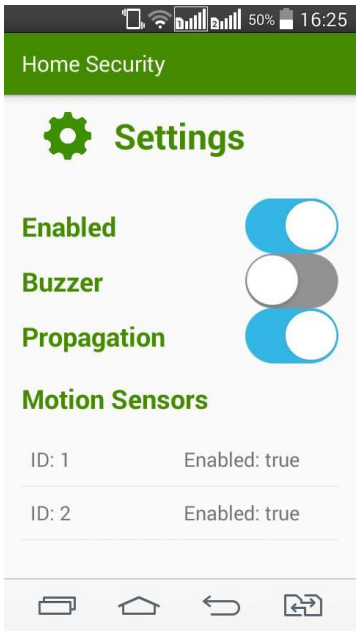

Figura 6 - Página das Configurações.

Figura 7 - Página do histórico dos alarmes

Na página de registos (*Records*) é apresentada uma lista dos alarmes que ocorreram no passado, ver exemplo na figura 7 e é indicado num *pop-up* o número total de registos transferidos.

Todo o código da aplicação foi desenvolvido no ambiente *Android Studio*. Utilizou-se uma atividade para cada uma das páginas e um serviço com uma *thread* a correr em segundo plano para realizar os pedidos periódicos. Todas as ligações à rede requerem uma *thread* diferente da *thread* de interface com o utilizador.

5. Protocolos Utilizados

5.1. Comunicação da Estação para Base e Servidor

Na comunicação da Estação de Base para Servidor envia-se BASE no início do pedido de maneira a que o servidor consiga perceber o remetente do pedido, este deve ser precedido de NOTIFICATION ou REQSETS consoante seja o pedido periódico de configurações dos nós ou uma notificação de alarme para o servidor.

Após um NOTIFICATION surge dois campos com o ID do nó detetor e o ID da PAN. O servidor deverá aplicar um carimbo temporal sobre o registo.

Depois de um REQSETS deve aparecer o ID da PAN que requer.

5.2. Comunicação do Android para o Servidor

Na comunicação do Android para Servidor envia-se ANDROID no início do pedido de maneira a que o servidor consiga perceber o remetente do pedido, este deve ser precedido de:

- LOGIN, cliente que tenta a sua autenticação, é acompanhado do EMAIL e da PASSWORD.
- REGISTER, cliente que tenta o seu registo, acompanhado do NOME, EMAIL, PAN ID e PAN SK.
- MODIFY, modifica os parâmetros dos nós da rede, acompanhado do PAN ID, do parâmetro alterado (por exemplo BUZZER, ou SENSOR), o VALOR do parâmetro e o nó do sensor caso seja um sensor a ser modificado.
- RETRIEVE, carrega as configurações atuais dos nós da rede, acompanhado do PAN ID.
- NOTIFICATION, averigua se a pessoa deve ser notificada, seguido do EMAIL da pessoa.

5.3. Comunicação do Servidor para Estação de Base

Quando o servidor rebebe uma notificação da estação de base com um novo alarme, insere o registo na base de dados, actualiza o campo de notificação de todas as pessoas associadas à PAN de onde foi proveniente o alarme. Responde OK.

Aos pedidos periódicos (REQSETS) responde:

OK [Enabled] [Buzzer] [Propagation] [NodeID1] [Enable1] ... [NodeIDN] [EnableN]

Em que os campos assinalados entre parêntesis retos têm os respetivos valores.

5.4. Comunicação do Servidor para Android

Aos pedidos do Android o servidor apenas responde com OK caso as verificações dos Queries tenham sido bem sucedidas ou NOK seguido dos parâmetros incorretos (caso do Registo e do Login). As situações mais importantes apresentam-se de seguida.

- Pedido de lista de registos: OK [Nº Registos encontrados]///[1ºReg [Data e hora] [ID Sensor]]// … [NºReg [Data e hora] [ID Sensor]]].

- Pedido de configurações dos nós: OK [System Enab.] [Buzzer Enab.] [Prop. Enab.]

5.5. Comunicação entre MICAz

Apresenta-se na figura 8, uma tabela com a estrutura definida em todos os MICAz para comunicação via IEEE 802.15.4 e a descrição do seu propósito. Os campos Node ID e Counter são inteiros de 16 bits e os restantes são booleanos.

	Node ID	Counter	Base Station	Buzzer Enabled	Motion Enabled	Propagation Enabled	Motion Detected	Tone Detected	System Enabled
Bytes	2	2	1	1	1	1	1	1	1

Figura 8 –Formato das mensagens entre MICAz.

- **Node:** Identificador do nó de destino ou origem da mensagem. 100 para todos os nós associados à placata MTS300/310. 1,2,3..etc para todos os nós associados a um sensor de movimento.
- **Counter:** Um contador que indica o número do pacote.
- **Base Station:** Indica se o pacote é originário da base station. Em caso afirmativo, o NodeID indica o nó de destino. Todos os nós guardam os parâmetros de configuração existentes numa mensagem com este campo a TRUE.
- **Buzzer Enabled:** Indica se as sirenes estão activas.
- **Motion Enabled:** Indica se o a detecção está activa para o nó com identificador NodeID.
- **Propagation Enabled:** Indicaria se o modo de propagação de sirene estaria activa, isto é se quando fosse detetado um tom de 4 kHz, a sirene deste nó deveria tocar.
- **Motion Detected:** Indica se foi observado movimento pelo sensor NodeID.
- **Tone Detected:** Indicaria se foi detectado um som.
- **System Enabled:** Indica se o sistema está activo. Em caso de FALSE, nenhuma sirene tocará nem nenhum pacote será enviado por sensores de movimento.

6. Configuração para envio do *e-mail*

Para a função mail() do servidor PHP funcionar foi necessário utilizar um servidor SMTP. Para tal criou-se um conta de email na Google e utilizou-se o seu servidor SMTP. Contudo foi ainda necessário configurar um programa que reencaminha o pedido de envio do email da máquina local onde corre o servidor PHP para o servidor SMTP da Google. Para tal utilizou-se o SSMTP que em Linux pode ser obtido através do comando:

$ sudo apt-get install ssmtp

Depois da sua instalação é necessário configurar o ficheiro ssmtp.conf acessível em /etc/ssmtp.conf com os dados de autenticação da conta criada como representado na figura 9.

```
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=postmaster

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=noreply.rmsf2016@gmail.com

# The full hostname
hostname=localhost

# Username/Password
AuthUser=noreply.rmsf2016@gmail.com
AuthPass=
UseSTARTTLS=YES

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
FromLineOverride=YES
```

Figura 9 – Edição do ficheiro ssmtp.conf.

De seguida apenas é necessário fornecer ao PHP o caminho onde deverá procurar o programa. Para tal acede-se ao ficheiro /etc/php5/cli/php.ini e modifica-se como representado na figura 10.

```
; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
sendmail_path = sendmail_path = /usr/sbin/ssmtp -t
```

Figura 10 – Edição do ficheiro php.ini.

Na figura 11 encontra-se um exemplo do *email* enviado ao utilizador com a sua password de acesso.

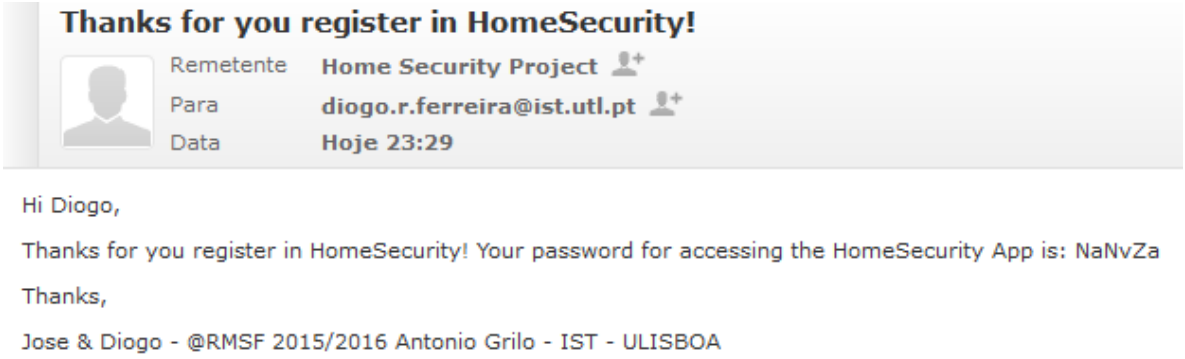

Figure 11 – Email recebido pelo utilizador.

Exemplo com Digrama Temporal

A figura 12 representa o digrama temporal do Login seguido da receção de notificações, primeiramente sem qualquer alarme e na segunda requisição com um aviso. A figura seguinte representa a entrada na página de *Settings* e a modificação de um parâmetro.

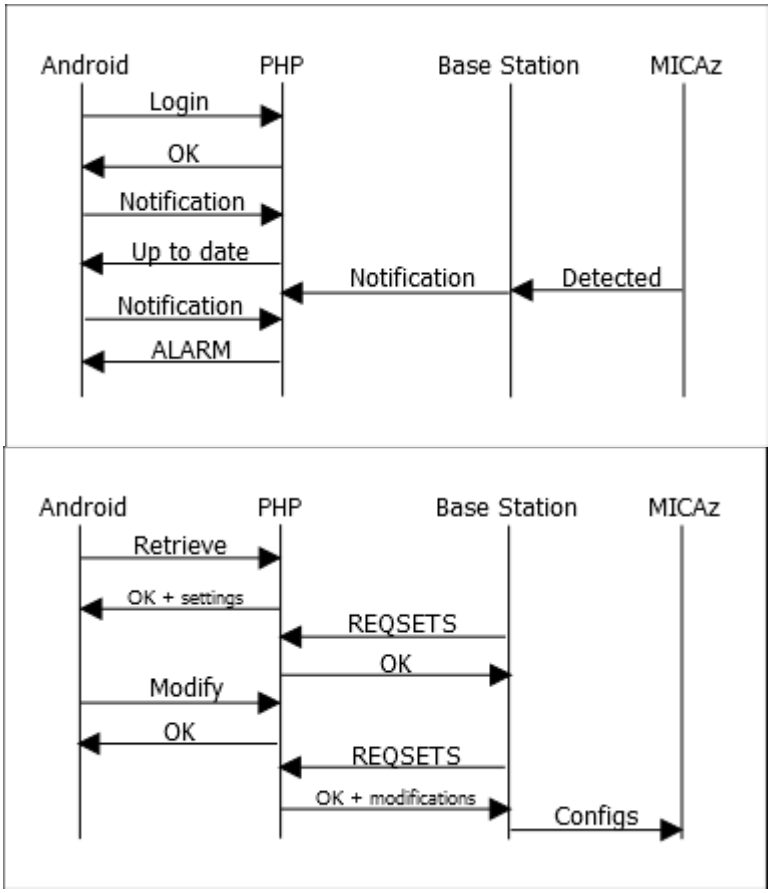

Figure 12 – Digrama temporal.

7. Considerações finais

Uma vertente que pode ser mais explorada neste trabalho é a área da segurança devido às implicações que podem existir caso o sistema não se encontre protegido contra eventuais utilizadores mal-intencionados. Assim sendo, pode ser implementado o envio de mensagens cifrado de modo a manter a confidencialidade entre os vários pontos do sistema, no envio das mensagens pode ser adicionado ainda um função que indique a integridade das mensagens (por exemplo *checksum*) e outra que dê a sua autenticação (por exemplo MAC). Na base de dados pode ser ainda utilizado uma função que cria uma *hash* de modo a que os valores das *password* não fiquem em *plain text* sendo acessíveis a qualquer pessoa que consiga aceder à base de dados.

Os pedidos periódicos efetuados pelo Android e pela estação de base representam um consumo energético que podia ser evitado se os endereços IP de ambos os pontos terminais fossem públicos e bem conhecidos.