Instituto Superior Técnico

Redes Móveis e Sem Fios

Entrega Final

Controlar a Temperatura da Água Dentro de um Aquário

Alunos	*Número de aluno*
Carina FERNANDES	84019
Sophie TABOADA	84187

2 de Junho 2019

Índice

1 Introdução

O projecto desenvolvido consiste sucintamente no desenvolvimento de um mecanismo de detecção e monitorização da temperatura da água num aquário através da implementação de comunicações sem fios entre vários dispositivos. Para atingir este fim são usados e desenvolvidos conhecimentos em várias áreas, nomeadamente na electrónica, na programação de aplicações, servidores *web*, e bases de dados. O maior objectivo deste trabalho é então conseguir estabelecer comunicação entre módulos de cada uma destas áreas e interliga-los entre si através do uso da Internet. A área em questão, também conhecida por *Internet of Things* é predominante no mundo tecnológico hoje em dia que está constantemente a evoluir.

2 Arquitectura Geral

A arquitectura geral da solução mantém-se bastante semelhante relativamente à apresentada no relatório intermédio. Isto é, as comunicações são feitas maioritariamente através do servidor, não existindo nenhuma comunicação directamente entre o ESP32 e a aplicação móvel. A aplicação é utilizada para criar utilizadores, representar as temperaturas dos sensores associados e mudar as temperaturas mínimas e máximas da água do aquário, assim como o estado *default* da bomba de água.

Por sua vez o servidor encarrega-se de guardar todas as informações fornecidas na base de dados mas também de fornecer ao ESP32 as temperaturas mínimas e máximas definidas pelo utilizador na aplicação. O ESP32 recebe esta informação, transmite as temperaturas medidas pelo sensor ao servidor e detecta se a temperatura excede algum destes limites. Se este for o caso, acciona a bomba caso a temperatura seja acima da máxima permitida e só a desliga quando a mesma baixar abaixo da temperatura mínima. Uma representação da arquitetura geral do projecto está disponível na Figura 1.

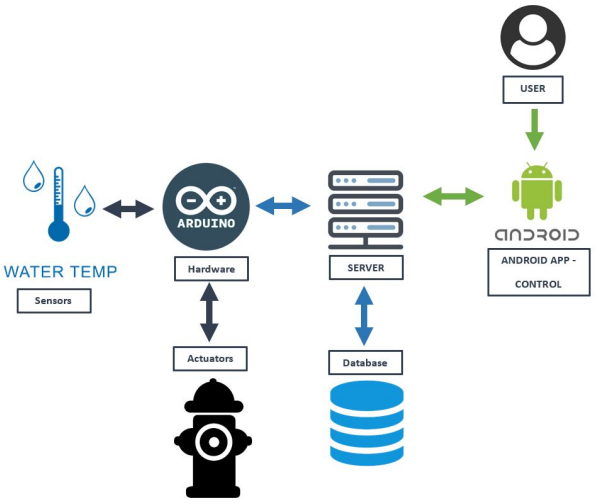


Figura 1: Arquitectura geral da solução implementada.

3 Hardware

Na Figura 2 apresentam-se as alterações feitas ao circuito inicial, nomeadamente a placa de Arduino Uno Rev2 foi substituída por outro componente devido a algumas complicações em transmitir os resultados do sensor de temperatura por falha técnica do aparelho. Optámos então por um modelo parecido, uma placa ESP-WROOM-32. Esta placa tem também um modulo WiFi integrado e as comunicações entre a placa e o servidor são simples e assemelham-se às que seriam realizadas com o Arduino Uno Rev2.

O sinal do sensor é extraído a partir do pino D15 da placa (fio amarelo) e o estado da bomba é enviado a partir da saída D19 da placa (fio roxo).

Uma segunda alteração foi feita a nível da alimentação da bomba de água. Uma vez que 5V não foram suficientes para por esta última a funcionar, decidiu-se substituir o transformador por uma pilha de 9V de maneira a fornecer à bomba a voltagem necessária para o seu correcto funcionamento.

Para que o sinal do sensor de temperatura seja lido correctamente, utilizou-se uma estratégia de *Pull Up* do sinal com a ajuda de três resistências (formando uma resistência de 4,7kΩ), ligadas em série entre os 3.3V provenientes do ESP32 e o sinal de saída do sensor.

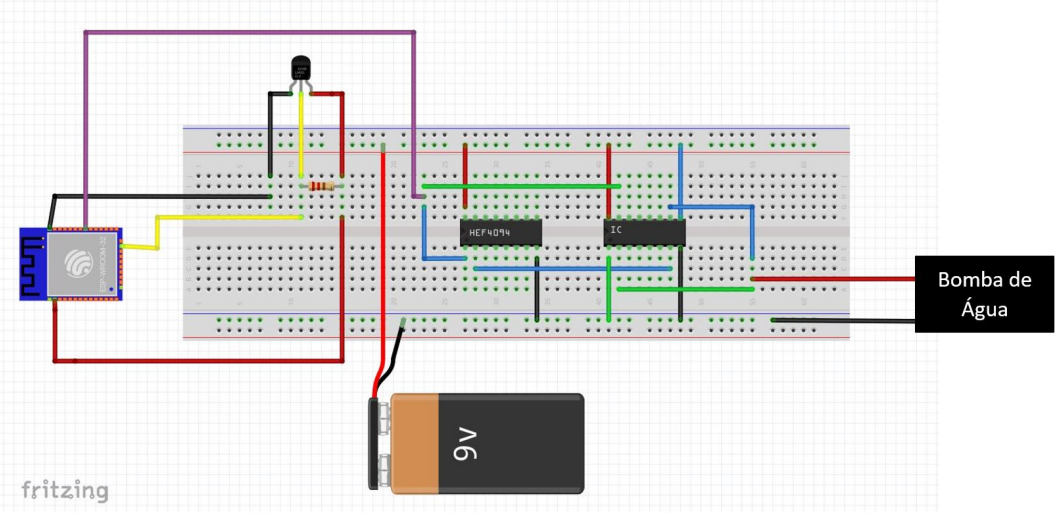


Figura 2: Diagrama de Fritz dos componentes de Hardware.

3.1 Funcionamento da Bomba de Água

De modo a ligar a bomba apenas quando o sinal do pino D19 do ESP32 fica activo, é utilizado um multiplexer construído a partir de portas de passagem CMOS e portas NOT, como se apresenta na Figura 3. O sinal gerado pelo ESP32 é usado directamente como controlador de um dos CMOS, com entrada ligada ao VCC. O segundo CMOS recebe como controlo o sinal invertido do sinal de saída do ESP32 e tem como entrada o sinal GND. Evita-se assim que os dois CMOS estejam a transmitir sinal ao mesmo tempo. Ambas as saídas dos CMOS estão ligadas em

curto circuito com a entrada VCC da bomba, sendo que a entrada GND está ligada a GND.

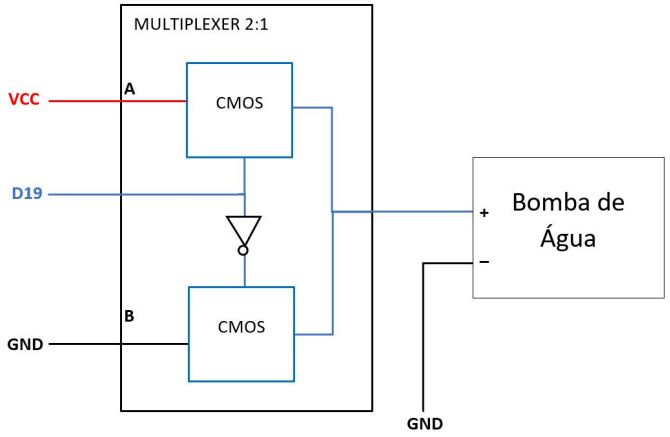


Figura 3: Multiplexer responsável pelo controlo da bomba de água.

3.2 Sensores/Actuadores

Correspondem aos mesmos indicados no relatório intermédio:

- Dallas Temperature Sensor DS18B20;

  A leitura de dados realizada pelo sensor pode retornar um resultado com precisão de 9, 10, 11 ou 12 bits, originando uma resolução de 0.5 °C, 0.25 °C, 0.125 °C e 0.0625 °C, respectivamente.

  Por sua vez a base de dados no servidor, guarda temperaturas como valores *float* de 5 dígitos de precisão, com 32 bits, (considerando leituras do sensor de 10 bits) e *username* e *password* como *char* de 8 caracteres, o que corresponde a 64 bits. Decidiu-se, que se usaria uma precisão até à décima do valor fornecido pelo ESP32 ao servidor. A leitura da Temperatura pelo ESP32 é feita a cada segundo.

- Uma Bomba de Água.

  Funciona a cerca de 9V e caso a leitura de temperatura seja inferior ao valor definido pelo utilizador para a temperatura mínima, a bomba será inactivada. Por sua vez, se o valor for superior ao de temperatura máxima, esta será activada.

4 Software

4.1 Software - Aplicação

Foi construída uma aplicação *Android* para fornecer informação ao utilizador acerca da temperatura do seu aquário e para que se pudesse mudar os limites de

temperatura associados a este.

A aplicação foi desenvolvida no *Android Studio* utilizando a linguagem XML para desenvolver a parte gráfica e Java para criar a parte interactiva entre o utilizador e a aplicação, sendo também responsável pela comunicação da aplicação com o Servidor.

O protocolo utilizado para a comunicação entre a aplicação e o servidor *web* é o protocolo HTTP. Isto significa que existe uma transferência de informação pela rede, que é acessível à aplicação através da conexão com um URL especifico. Por este meio, a aplicação consegue fornecer informação ao servidor.

Tomando como exemplo o caso da interface *Register* na aplicação, como se observa na Figura 4, em que a aplicação deseja mandar as informações de um novo utilizador para o servidor para que este o adicione de seguida na base de dados.

```
String link = "http://web.ist.utl.pt/~ist425319/register.php?username=" + username+ "&password="+
URL url = new URL(link);


"&password="+ password + "&minTemp=" + minTemp+ "&maxTemp=" + maxTemp + "&WBdefault=" + WBstatus;
```

Figura 4: Conexão com o URL do servidor fornecendo a este, os dados do novo utilizador.

De modo a saber se a operação foi realizada com sucesso e que os dados foram registados na base de dados, o servidor responde na mesma página *web* a indicar se houve erro, incluindo uma mensagem de esclarecimento. Na Figura 5 está um exemplo da mensagem fornecida pelo servidor quando a temperatura mínima indicada pelo utilizador está em vazio.

Figura 5: Mensagem do utilizador no URL depois do utilizador submeter uma temperatura mínima em vazio.

A aplicação também pode requisitar informação da base de dados ao servidor. O procedimento é feito da mesma maneira, é feita a conexão com servidor usando o URL especifico e juntando os dados necessários para a resposta. O servidor imprime então a informação na respectiva página*web*, seguindo-se o exemplo do requisito das temperaturas medidas pelo sensor na Figura 6, em que a aplicação apenas manda o

nome do utilizador para que o servidor consiga identificar as temperaturas respectivas presentes na base de dados.

```
String username = getUsername();
String link = "http://web.ist.utl.pt/~ist425319/getTs.php?username=" + username;
URL url = new URL(link);

HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.connect();

InputStream response = urlConnection.getInputStream();
String TempArray = getTempFromResponse(response);
```

Figura 6: Conexão com URL indicando o `username` e processamento da resposta.

Para ter acesso às respostas do servido, é efectuado um *GET request*, que permite só e apenas, obter dados numa página especificada. Desta maneira assegura-se que não há nenhuma alteração feita na base de dados, já que se trata de uma operação de leitura apenas. A resposta fornecida pelo do servidor é do tipo *JSON Object*. Para que a aplicação consiga descodificar esta mensagem, criou-se uma função que passa essa informação para uma *String*. No caso da Figura 6, a função é `getTempFromResponse()` que recebe como parâmetro a resposta obtida quando é efectuada a conexão ao URL. Estes procedimentos encontram-se envolvidos dentro de uma *thread* chamada *AsyncTask*, permitindo assim correr a conexão com o servidor em paralelo com o resto do programa. Isto permite ao utilizador continuar a mexer na aplicação sem ter de esperar pelo sucesso da ligação. Outra vantagem corresponde ao facto da aplicação não ir abaixo caso a ligação não seja sucedida.

A comunicação com o servidor é conseguida utilizando as seguintes classes:

- `MainActivity`: corresponde ao *login*, a aplicação envia os dados (`username` e `password`) inseridos pelo utilizador nos campos para preencher, e o servidor confirma se o utilizador existe e se as credenciais estão correctas.
- `ResisterActivity`: a aplicação envia os dados inseridos pelo utilizador nos campos a preencher, e o servidor confirma se o utilizador não já está na base de dados e se os dados são válidos. Caso esteja tudo correcto, os dados são inseridos na base de dados.
- `GraphActivity`: a aplicação pede informação acerca das temperaturas medidas pelo sensor ao longo do tempo.
- `SettingsActivity`: alteração dos valores dos limites de temperatura e estado da bomba sendo os novos valores desejados enviados para o servidor.

4.2 Software - ESP32

De modo a exemplificar o funcionamento geral do projecto, com especial foco no desempenho do ESP32, foi criado o fluxograma apresentado na Figura 7.

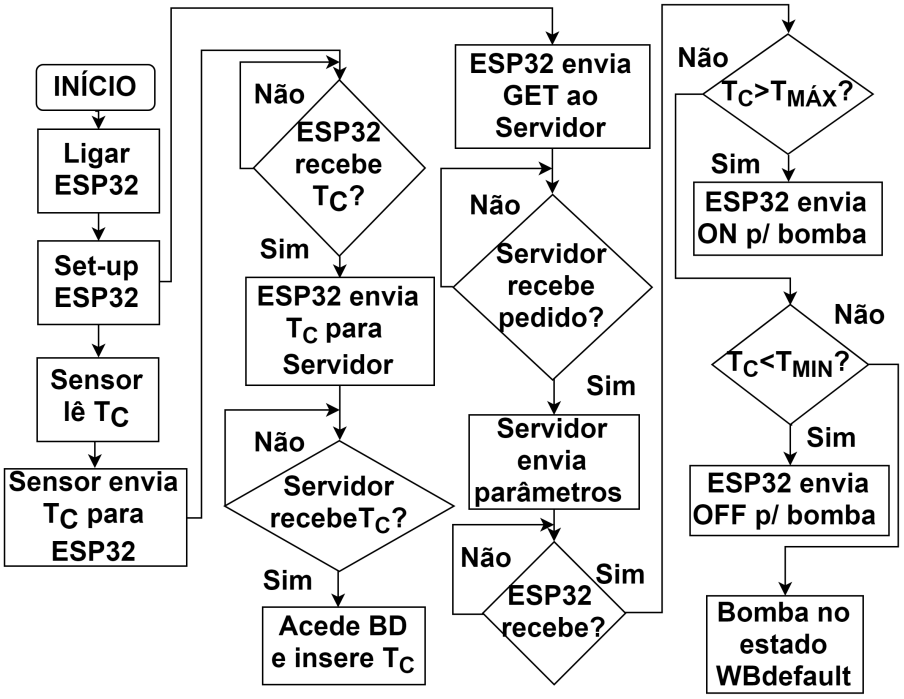


Figura 7: Fluxograma do funcionamento principal do circuito implementado.

- **Ligação ESP32 - Bomba de Água**

Tal como foi referido na secção 3.1, a actuação sobre a bomba é realizada conforme o valor colocado no pino D19 sendo esta simplesmente realizada através de um `digitalWrite`.

- **Ligação ESP32 - Sensor de Temperatura**

Para interagir com o sensor o ESP32 utiliza as bibliotecas `DallasTemperature` [1] e `OneWire`[2] que o permite efetuar leituras dos valores de temperatura medidos.

Esta primeira biblioteca permite a definição de uma variável `sensors` do tipo `DallasTemperature` sobre o qual são chamados os métodos apresentados na Tabela 1.

É ainda criada uma variável `sensor` do tipo `DeviceAddress` que guarda os vários endereços dos pinos do ESP32 disponíveis que obtém graças à biblioteca `OneWire`.

Tabela 1: Métodos usados da biblioteca `DallasTemperature`.

Método	Descrição
`begin()`	Inicializa o barramento de endereços.
`requestTemperatures()`	Comando para realizar medidas de temperatura.
`getTempC(sensor)`	Devolve a medição em graus *Celcius*.

- **Ligação ESP32 - Base de Dados**

O ESP32 precisa de comunicar com a base de dados de modo a saber o utilizador em questão e as temperaturas de água admissíveis por este de modo a decidir se deve ligar ou não a bomba de água. Para além disso é preciso que os valores lidos de temperatura sejam guardados na base de dados de forma a permitir o desenho de um gráfico de evolução de temperatura.

Visto que as interacções com a base de dados foram codificadas para serem realizadas através de páginas *web* é preciso que o ESP32 seja capaz de enviar e receber informação das mesmas, o que será feito de modo semelhante ao apresentado na secção 4.1.

Portanto é utilizada a biblioteca *HttpClient* [3] que permite realizar pedidos *GET* a páginas *web* que por sua vez comunicam com o servidor onde está guardada a base de dados.

Estes pedidos são realizados através da criação do objecto `http` do tipo `HttpClient`, sendo os métodos utilizados sobre este objecto apresentados na Tabela 2.

Tabela 2: Métodos usados da biblioteca `HttpClient`.

Método	Descrição
`begin(URL)`	Inicializa o objecto para esse URL.
`GET()`	Realiza pedido através de uma ligação TCP.
`getString()`	Devolve a *payload* do obtido ao efetuar o pedido como uma *string*.
`end()`	Fechar porta TCP.

Para realizar a ligação TCP é necessário que o ESP32 esteja ligado à Internet, sendo isto feito através de uma ligação Wi-Fi. Para tal, é usada a biblioteca `WiFi` [4], podendo a ligação ser realizada a qualquer rede desde que sejam configurados o nome (`ssid`) e a *password* (`password`) da mesma.

Os métodos utilizados desta bibliotca estão apresentados na Tabela 3.

Tabela 3: Métodos usados da biblioteca `WiFi`.

Método	Descrição
`begin(ssid, password)`	Inicializa os parâmetros da rede e inicia a ligação.
`status()`	Devolve o estado da ligação.

4.3 Software - Servidor

Para facilitar o armazenamento de informação e a comunicação entre com o ESP32 e a aplicação móvel, foram utilizados os servidores `db.tecnico.ulisboa.pt` e `web.ist.utl.pt` respectivamente.

- **Ligação Web - Aplicação Móvel**

Tal como foi dito na secção 4.1, a comunicação é realizada através de métodos definidos pelo protocolo HTTP.

Quando a aplicação realiza um pedido à página *web*, esta última realiza um `GET` de forma a receber a informação necessária para interagir com a base de dados.

De seguida, de acordo com o resultado dessa interacção, este é guardado numa variável `response`.

De modo a verificar a correcta implementação das várias funcionalidades, é utilizado o comando `echo` de modo a imprimir a variavél PHP `response` nas páginas *web*.

- **Ligação Web - Base de Dados**

A comunicação entre as páginas *web* e a base de dados é realizada através da criação de ligações do tipo PDO (*PHP Data Objects*).

A criação de cada instância desta classe permite estabelecer uma ligação ao servidor `db.tecnico.ulisboa.pt` sendo então possível aceder às bases de dados do utilizador `ist425319`.

É através destas ligações que serão realizados *queries* à base de dados, isto é comandos `INSERT`, `SELECT` e `UPDATE` permitindo assim inserir, aceder e atualizar informação na mesma.

5 Servidor

5.1 Servidor - Base de Dados

De forma a guardar a informação necessária para o funcionamento do projecto são utilizadas duas tabelas que estão guardadas na base de dados `ist425319`. Esta base de dados está inserida no servidor `db.ist.utl.pt` que corre, entre outras, a aplicação de sistema de gestão de base de dados MySQL. Este é o sistema que suporta a linguagem SQL que é a utilizada neste projecto para interagir com a base de dados.

Nesta implementação existem duas tabelas armazenadas na base de dados, que são criadas, consultadas e actualizadas conforme o necessário.

A primeira armazena informação que o utilizador consegue alterar através da mobile App, nomeadamente o *username*, a respectiva *password*, a mínima e máxima temperatura de água admissíveis, *minTemp* e *maxTemp* respectivamente, assim como o estado *default* da bomba de água, *WBdefault*. Esta tabela encontra-se apresentada na Tabela 4.

Tabela 4: Tabela *users* que guarda dados sobre o utilizador.

username	password	minTemp	maxTemp	WBdefault
-	-	-	-	-

A segunda tabela armazena as leituras de temperatura realizadas pelo sensor de temperatura conforme o actual utilizador e tempo de medição, estando representada na Tabela 5.

Tabela 5: Tabela *temp* que guarda leituras do sensor de temperatura.

temperature	username	time_st
-	-	-

O acesso ao acima referido servidor `db.ist.utl.pt` é feito através do *cluster* `sigma` do Instituto Superior Técnico.

5.2 Servidor - Web

É também através do *cluster* `sigma` que é acedido o servidor *web*, `web.ist.utl.pt` onde estarão os ficheiros de PHP desenvolvidos para implementação das várias páginas *web* necessárias.

Cada página desenvolvida e a respectiva descrição está apresentada na Tabela 6.

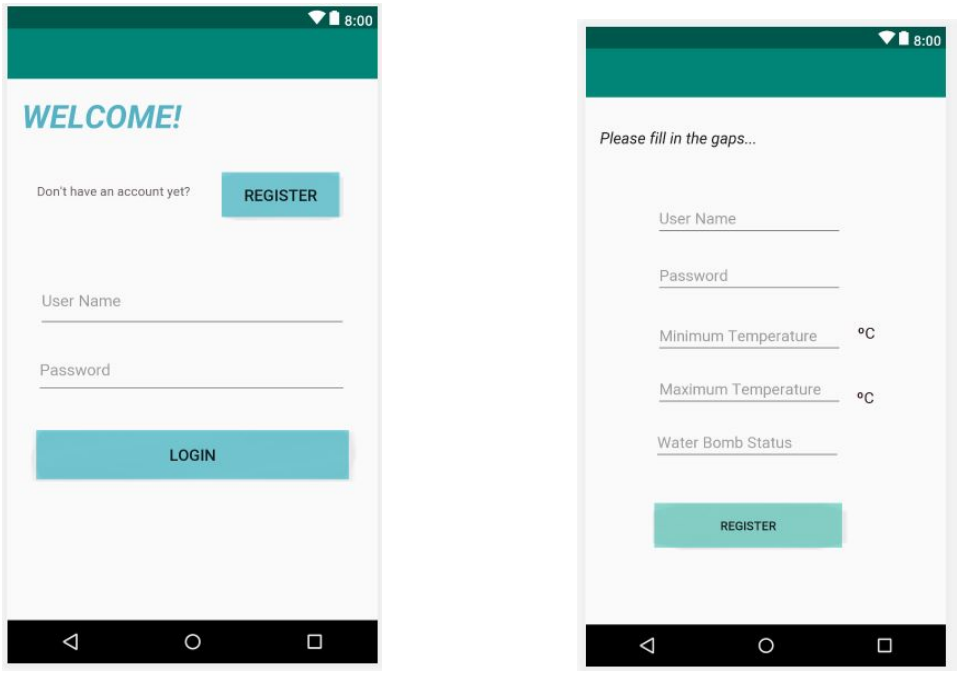
Tabela 6: Ficheiros PHP desenvolvidos.

Ficheiro	*Input*	Descrição
`getTs.php`	`username`	Obtém os valores de temperatura presentes na base de dados para um dado utilizador.
`getvals.php`	`username`	Obtém os valores de `minTemp`, `maxTemp`, `WBdefault` do utilizador.
`insertTc.php`	`username/temp/time_s`	Insere leitura de temperatura na base de dados.
`login.php`	`username/password`	Verifica se o utilizador e a passe existem na base de dados. Username atual é guardado num ficheiro *username.txt* no servidor *web* que será acedido pelo ESP32.
`newpass.php`	`username/password/newpass`	Altera a passe de um utilizador na base de dados.
`register.php`	`username/password/minTemp/maxTemp/WBdefault`	Regista um novo utilizador na base de dados.
`updateTsW.php`	`username/newTmin/newTmax/newWd`	Atualiza os parâmetros de um utilizador.
`utils.php`		Possui funções auxiliares aos restantes ficheiros.

6 Aplicação Móvel

A aplicação é constituída por 6 interfaces. As duas primeiras sendo a do *Login*, na Figura 8a, e do *Register*, na Figura 8b, ambas apresentadas na Figura 8.

Os dados necessários para fazer o *login* são o `username` e a `password` do utilizador. Estes dados são depois confirmados pela base de dados, caso o `username` exista e a `password` esteja correcta, a aplicação passa para a interface de *Menu*. Caso as credenciais estejam erradas então a aplicação informa o utilizador de que algo está incorrecto. Para o caso do *Register* o utilizador tem de preencher as diferentes partes do registo. A informação é depois enviada para o servidor que por sua vez coloca os dados na base de dados, criando uma conta para este novo utilizador.



(a) Janela Inicial (b) Janela Register

Figura 8: Imagens do Login e do Register da Aplicação

Uma vez iniciada a sessão, apresenta-se o menu principal, presente na Figura 9, em que o utilizador pode selecionar observar o gráfico das temperaturas medidas pelo sensor, Figura 10a, ou alterar as *settings* definidas durante o registo à aplicação, Figura 10b. O utilizador também pode optar por alterar a sua palavra passe, Figura 10c.

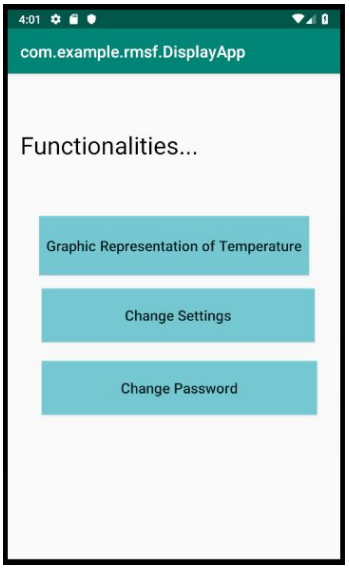
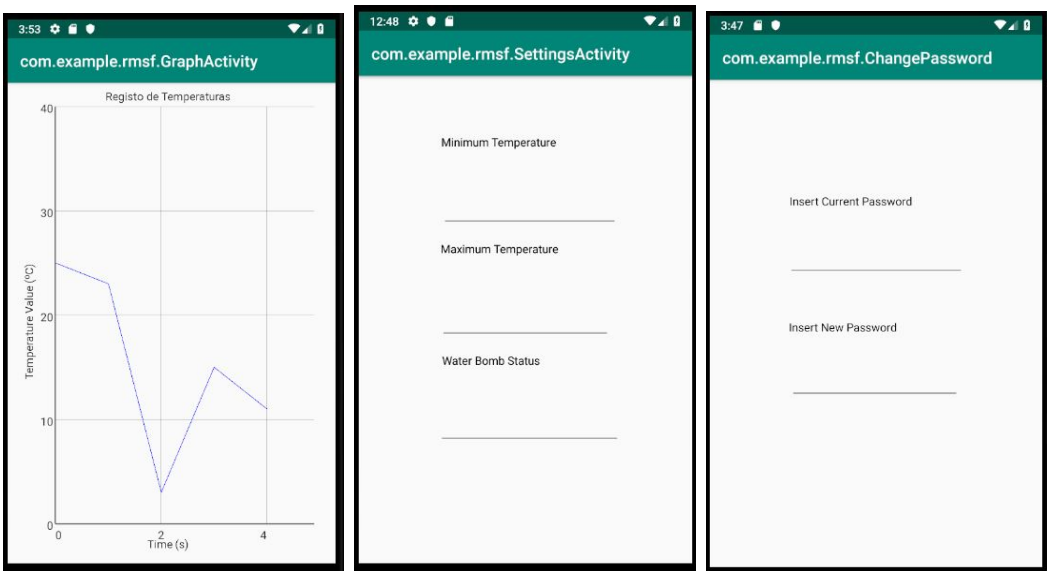


Figura 9: Menu principal da aplicação



(a) Janela *Graph*. (b) Janela *Change Settings*. (c) Janela *Change Password*.

Figura 10: Imagens do *display* do gráfico das temperaturas e das páginas de modificação de limites da aplicação.

7 Diferenças face à entrega intermédia

A principal diferença entre o proposto na Entrega Intermédia e a implementação final é a utilização do módulo ESP32 em vez do ESP32 Uno previamente proposto.

Após realizados testes foi verificado que o ESP32 em questão tinha problemas em transmitir as leituras realizadas pelo sensor de temperatura e foi então substituído pelo ESP32. O transformador de tensão foi substituído por uma pilhar de 9V.

Para além disso não foi utilizada a biblioteca `WiFiNINA` mas sim a biblioteca `WiFi`.

Em termos de actuação sobre a bomba de água, a decisão de mudança de estado ON/OFF passou a ser realizada no ESP32.

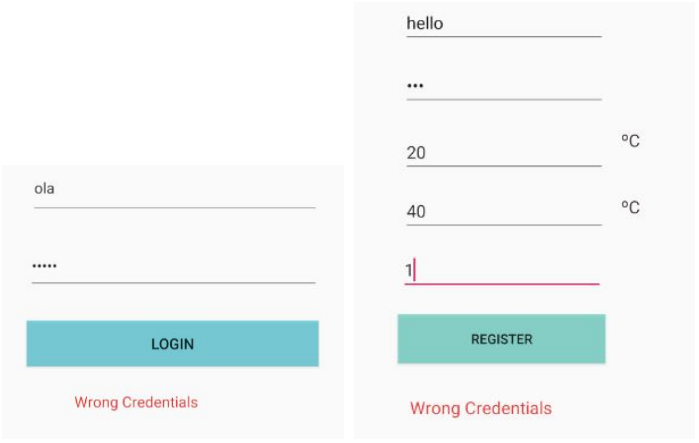
Neste momento já são efectuadas todas as comunicações necessárias entre os dispositivos. As restantes diferenças são detalhes de funcionamento principalmente da aplicação, tal como, por exemplo, a adição da obtenção do gráfico de evolução de temperatura.

8 Testes de Sistema

De modo a verificar o correcto funcionamento do projecto, foram realizados vários testes ao longo do seu desenvolvimento. De modo a exemplificá-lo serão apresentados alguns testes que demonstram as funcionalidades básicas.

- **Testes de Login e Register**

O Login é efectuado caso o utilizador exista na base de dados e a password inserida corresponda à password do utilizador. Se os dados não coincidirem, uma mensagem em vermelho é afixada no *écran*. O mesmo acontece quando na interface Register o utilizador tenta inserir um nome de um utilizador já existente ou temperaturas ou estado da bomba inválidos. Estes exemplos estão ilustrados na Figura 11.

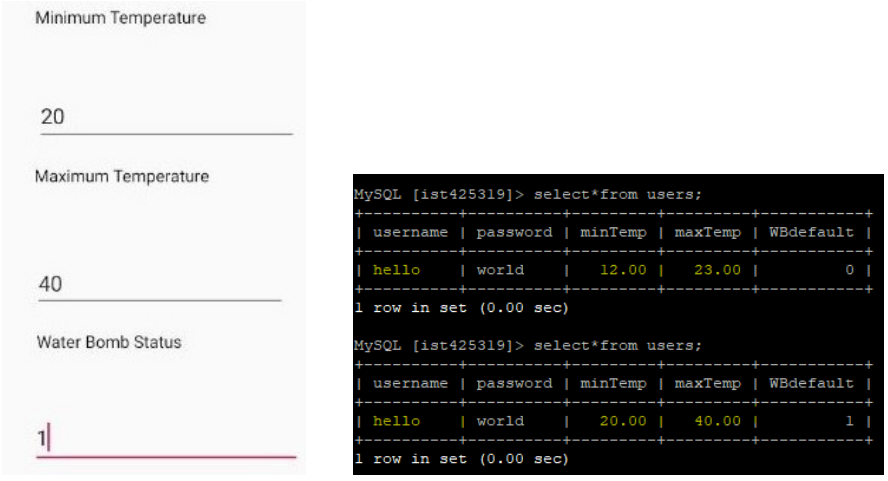


(a) Janela do Login falhado (b) Janela do Registo falhado

Figura 11: Imagens das interfaces iniciais da aplicação em caso de erro.

- **Alteração de Parâmetros+ Leitura Temperatura + Atuação da Bomba**

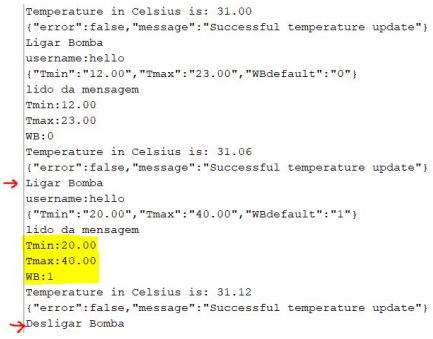
De forma a demonstrar a correta alteração de parâmetros através da interface *Register* da aplicação, está apresentado na Figura 12, a introdução dos novos valores na aplicação, na Figura 12a, e a sua posterior alteração na base de dados, na Figura 12b.



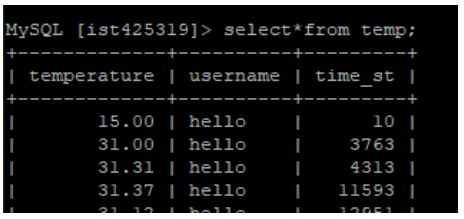
(a) Alteração dos parâmetros. (b) Estado da tabela users antes e após alteração.

Figura 12: Alteração dos parâmetros na aplicação e efeito na base de dados.

Em relação à leitura das medições de temperatura, dos actuais parâmetros e eventual actuação sobre a bomba, na Figura 13 é possível observar que o ESP32 realiza leituras dos parâmetros actualizados da base de dados e das medições de temperatura realizadas pelo sensor, Figura13a, sendo estas inseridas na base de dados como se pode observar na Figura13b.



(a) Leitura actualizada dos parâmetros e da temperatura no ESP32.



(b) Inserção de valores de temperatura na base de dados.

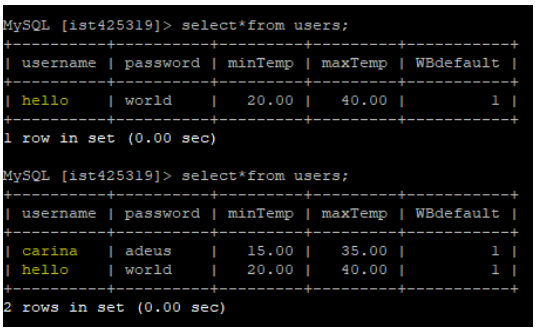
Figura 13: Alteração dos parâmetros na aplicação e efeito na ESP32 e na base de dados.

Para além do mais, na Figura13a é possível verificar a correcta actuação sobre a bomba de água de acordo com os parâmetros da base de dados e a última leitura de temperatura.

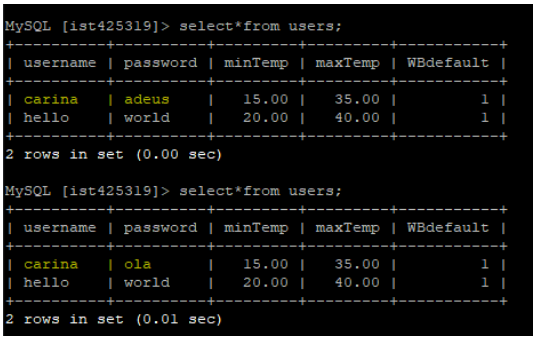
- **Novo utilizador + Alteração de Passe**

A verificação das funcionalidades implementadas nas interfaces *Register* e *Change Password* da aplicação, está apresentada na Figura 14.

Na Figura 14a verifica-se a criação de uma nova linha na tabela de utilizadores, enquanto que na Figura 14b observa-se a alteração da sua palavra passe.



(a) Adição de novo utilizador à base de dados.



(b) Alteração da passe do utilizador.

Figura 14: Efeitos na base de dados de ações feitas na aplicação.

9 Conclusões

A bem sucedida implementação deste projecto foi de facto um desafio que exigiu a utilização e desenvolvimento de capacidades em várias áreas, desde a programação de aplicação móveis, de microcontroladores, à gestão de bases de dados incluindo ainda bases de electrónica. Neste âmbito, era crucial então garantir a correcta utilização dos vários mecanismos e protocolos de comunicação entre os diversos módulos utilizados.

Para além do mais, ao longo deste projecto tornando-se bastante claro o potencial de dispositivos que utilizam o conceito de *Internet of Things* uma vez que este facilita bastante a comunicação entre diversos módulos e plataformas, assegurando assim a correcta implementação das funcionalidades desejadas.

Por fim conclui-se que o desenvolvimento do projecto foi bem sucedido tendo sido de facto implementado um mecanismo de controlo e monitorização de temperatura de água, tendo para tal sido aplicados conceitos de comunicação sem fios.

Referências

[1] "Dallas Temperature Library," https://github.com/milesburton/Arduino-Temperature-Control-Library.

[2] "OneWire Library," https://github.com/PaulStoffregen/OneWire.

[3] "HTTPClient Library for Esp32." https://github.com/espressif/arduino-esp32/tree/master/libraries/HTTPClient.

[4] "WiFi Temperature for Esp32," https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi.