INSTALLATION GUIDE

The installation of the whole project begins in the CCU of the remote testbed, the Raspberry Pi 4 Model B. There are several configurations to take into account in order to achieve the correct behavior in the server.

- **installationCCU.sh:** It contains all the installations needed to obtain the desired behavior in the server. It does not require any argument, just to be executed in the device that will act as the system's server (in our case, in the Raspberry Pi 4 Model B).

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop/thesisCode$
./installationCCU.sh
```

Figure A.1. Installation central server instruction.

Once the installation of the previous packages has been done correctly, we have to compute several changes and additions.

1. Create the MySQL user that identifies our scripts when running the project. Therefore, once the previous step was done, in one terminal of the server:

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop/thesisCode$ mysql -u root
-p
```

Figure A.2. MySQL access command.

* It is necessary to follow the implementation chosen by the customer while the implementation was taking place.

```
mysql> CREATE USER ‘testbedControl’@’localhost’ IDENTIFIED BY ‘raspberrypi’
```

Figure A.3. MySQL user creation command.

* If any of the data is changed when creating the user, it also needs to be changed in the scripts used to store the data.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'testbedControl'@'localhost' WITH
GRANT OPTION;
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

Figure A.4. MySQL user providing privileges command.

The user used to identify the requests done to the database has been created, therefore, the user needs to log in the MySQL system to create the database that will store all the data.

```
mysql> mysql -u testbedControl -p  # Insert the password: raspberrypi
mysql> CREATE DATABASE sysMonitorDB;
```

Figure A.5. MySQL database creation command.

* Now, it is necessary to run the script that generates all the tables that form the database previously created. There is a specific folder called "*testbed*" that includes all the different scripts and elements that will construct the website. At the moment, we need the script called "database.php", in order to execute it, in a terminal of the raspberry :

```
raspberrypi@raspberrypi:/var/www/html$ php database.php
```

Figure A.6. MySQL database table's creation command.

The database is already organized and ready to receive information. However, there are two more steps to take into account.

It is necessary to include the scripts and elements that compose the Website to the specific path that will allow the other devices to interact with it. There is a script responsible to do that:

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop/thesisCode$ ./website.sh
```

Figure A.7. Website content copy command..

The last step is ensuring the server listens to the specific port used to send the requests from the Odorids M1S. It is required to be listening to the port 5053, therefore:

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop$ sudo nano
/etc/apache2/ports.conf
```

Figure A.8. Server port modifications.

Add or modify the Listen directive to specify the port you want Apache to listen on. For example, to listen on port 5053, add:

```
Listen 5053
<IfModule ssl_module>
      Listen 5053
</IfModule>
```

Figure A.9. Port modification content to ensure the data receiving.

* Execute ^X to save the changes.

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop$ sudo systemctl restart
apache2
```

Figure A.10. Server restart command.

Once all the different steps have been done, the server is ready to receive information, the user can continue with the Odroid M1S configuration.

The installation process for the Odroid M1S begins in the Raspberry Pi 4 Model B. Several steps have to be taken into account to obtain the correct functionality:

1. Coding transmission to the Odroid M1S

The Raspberry Pi has to send each Odroid M1S the folder that contains all the necessary installations and scripts to constitute the remote testbed. In order to send all this content, a specific script will be executed in a terminal of the Raspberry Pi:

- **installation.sh:** It requires one argument, the IP address of the Odroid M1S in which it is necessary to install the system. An example of how to run it is (considering the IP address of an Odroid M1S: 192.168.0.103 and the IP address of the Raspberry Pi: 192.168.0.130):

```
raspberrypi@raspberrypi:/home/raspberrypi/Desktop/thesisCode$
./installation.sh 192.168.0.103
```

Figure B.1. Installation in the Odroid M1S instruction.

The Odroid M1S has to be in the same network as the Raspberry Pi to receive the data, if not, the communication will not be established and no content will be transmitted.

Once the transmission is completely done, a new folder will appear in the path:

```
odroid@server:/home/odroid$ ls
odroid@server:/home/odroid$ odroidConfiguration
```

Figure B.2. Access to the configuration folder of the Odroid M1S .

"ls" : command used to see what is in that specific path.

2. Odroid configuration

Once the Odroid M1S has all the scripts needed to run the system, it is first required to install a series of packages and programs that will be useful to execute the other scripts. Therefore, the next step is to run in the Odroid M1S terminal and while being connected to WiFi:

```
odroid@server:/home/odroid/odroidConfiguration$ ./configuration.sh
```

Figure B.3. Installation Odroid M1S packages and dependencies.

Please accept all the necessary conditions to install all the different packages.

3. Odroid scripts running

All the packages and dependencies of the scripts are downloaded and installed in the Odroid M1S, therefore, it is prepared to play its role in the system.

Inside the previously folder, there are different scripts:

- **registration.sh:** It is used to register in the database which nRF Dongle is connected to each Odroid M1S. Each Odroid M1S can be associated with different nRF Dongles but this information has to be sent to the server. It accepts multiple arguments that correspond to the MAC addresses of each nRF Dongle. For example:

Considering the Odroid M1S is connected to two different nRF Dongles whose MAC addresses are: a9:B7:c5:66:5f:Y8 and 4p:HI:2f:j9:P8:2s. If all the devices are connected to the same network as the server and by executing:

```
odroid@server:/home/odroid/odroidConfiguration$ ./registration.sh
a9:B7:c5:66:5f:Y8 4p:HI:2f:j9:P8:2s
```

Figure B.4. Registration of the nRF Dongles and Odroid M1S example execution.

The script automatically obtains the Odroid's MAC and sends to the server the specific information so it can process it and store it in the system's database.

- **read.sh:** It is used to read all the information received from the nRF Dongles so it can be sent to the server for its processing and storing step. Once the customer wants to save the data received it can be executed by computing in a terminal:

```
odroid@server:/home/odroid/odroidConfiguration$ ./read.sh
```

Figure B.5. Enable Odroid M1S to receive data from the nRF Dongles script.

If there is no nRF Dongle sending information to the Odroid, the customer will receive that announcement and the script will have to be run again once a nRF Dongle has been connected to the Odroid M1S in which this script was executed.

- **voltageMeasurement.py:** It is used to obtain the current consumed by the nRF Dongles every 10 seconds, a value that can be changed by the customer. Once it is executed, every 10 seconds it obtains the voltage value and sends it to the server to calculate the corresponding current value. In order to execute it, in the terminal it is required to compute:

```
odroid@server:/home/odroid/odroidConfiguration$ sudo python3
voltageMeasurement.py
```

Figure B.6. Enable Odroid M1S to obtain the current data script.

* Root permissions are required to access the GPIO pins.

* Press ^C to stop the execution of the script.

* While the previous command is executed, please run the next command to send all the computed data to the server (Raspberry Pi).

```
odroid@server:/home/odroid/odroidConfiguration$ ./readCurrent.sh
```

Figure B.7. Enable Odroid M1S to send the current data to the server script.

* Press ^C to stop the execution of the script.

- **reset.py:** It requires one specific argument, a binary file that is sent to the nRF connected to the Odroid M1S so it is possible to execute the reset of their configuration. This script should be executed by the customer. It requires a binary file (for example: newConfiguration.bin) and it can be run by:

```
odroid@server:/home/odroid/odroidConfiguration$ sudo python3 reset.py
newConfiguration.bin
```

Figure B.8. Sending binary configuration to nRF Dongle script.

Finally the website will be available in any browser (recommended Firefox) by searching for the URL= “*http://191.168.0.130:5053/*”.