IoT 19/20 - Final Project Report

Marcello Cellina 10709506 and Giulio Mario Martena 10437884 - MSC in Automation and Control Engineering.

Introduction

Our final project for the Internet of Things course revolves around the development of a remote control for a heater via internet. The control action will be carried out through an internet connected relay.
Furthermore, a separate room's temperature will be monitored through sensors and the heater will be activated automatically if said temperature falls below a certain threshold.

The customer is not highly technologically educated, so the HMI needs to be easy, safe and intuitive.

Methodologies

First of all, we defined the needed devices and wirings:

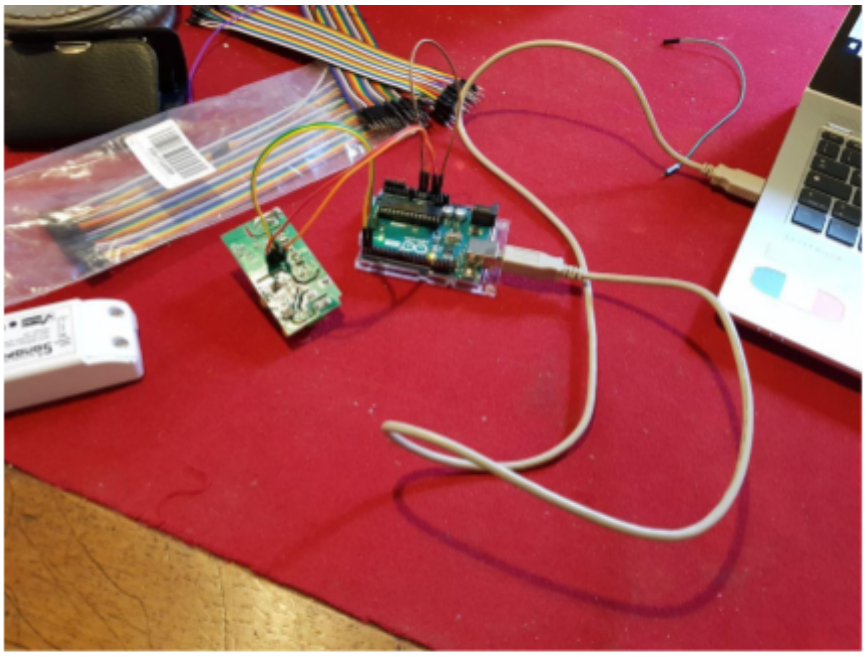
- The HMI interface of choice is a **Telegram bot**;
- The relay of choice is the **Sonoff Basic**, a smart switch based on ESP which can communicate via WiFi;
- The communication protocol will be **MQTT**;
- The monitoring of the room is done through sensors controlled by an **Arduino Uno** which really does not need any introduction; the communication happens through serial port.

Server setup

The physical machine mounts Ubuntu Server 18.04. It will act as a broker, so we need to install Mosquitto on it.
First of all, we installed Mosquitto.
and we enabled the relative daemon to automatically startup using *systemd*; the port used by said daemon is the 1883.
Then we installed node-red; *NodeJS* and its package manager *NPM* are essential for this stage.
Thus, we enabled node-red to startup automatically as well; as a result, node-red is operating on port 1880.
Next, we installed the Telegram palette and then added it to node-red through its builtin node manager. The full code can be found in the *Server Setup* section of the attached source code.

Device setup

The Sonoff Basic is not out-of-the-box compatible with MQTT, but only supports its own cloud platform; luckily, the open source community developed many alternative firmwares which free the device from the proprietary constraints and implement solutions for MQTT. The **Sonoff Tasmota** firmware was chosen for this task.
After downloading the *tasmota.bin* from this Github page, we flashed the device using the Arduino Uno, which in this scenario acts as a USB/TTY converter.



The Sonoff flashing stage. Notice the Arduino Uno board acting as a USB/TTY converter

Arduino communication

The missing piece of the puzzle is the Arduino. Its main task is to publish the temperature on the serial port with a frequency of **1 msg/minute** and to go to sleep in the meantime.
The main issue here has been to link together node-red and the Arduino hardware; to resolve this issue, we installed the **serial port** node from this repository.

Node Red Flow

The bot shows a predetermined keyboard with the main commands (in the image on the left).
The two nodes on top are command nodes, and define the behavior of the two main commands:

- */accendi* sets the topic *cmnd/tasmota/power* to 1
- */spegni* sets the topic *cmnd/tasmota/power* to 0

/start, quite self-explicably, starts the interaction between user and bot, while */stato* allows the user to retrieve the state of the boiler (ON or OFF).

The receiver block right underneath blocks the user from sending unwanted commands to the bot.
Finally, the block at the bottom of the page listens for state changes and sends them to the user through the bot.
The full node-red flow is attached to this report and a screenshot of it reported in the source code attachment.

Results

We carried out the first test on a lamp. We were able to successfully control it from the Telegram bot, which correctly answered every command. Then we deployed it on the real system which, as mentioned above, was already correctly configured to communicate with the Arduino through MQTT. Further details on the Tasmota firmware can be found in its doc page.

Conclusion

We were able to finish the project and had quite fun with it.
This project let us discover some interesting technologies out there; by developing this little system, we were able to appreciate how many things can be done with the tools we saw during classes.

The customer was very pleased with the result, and overall it has been a growing opportunity which made us put our hands on things and appreciate both the engineering process and the era we live in, which allows us to dramatically simplify our lives.

Attachments

- *bot screenshotX.png*: a screenshot of the working Telegram bot;
- *log.pdf*: a self-explainatory log file for the node-red flow;
- *Node-RED Debug Tools.html*: an html version of the log (better formatted for browser visualization);
- *node-red flow.json*: the working node-red flow for the bot;
- *Physical System.jpg*: a picture of the deployed relay;
- *serversetup.sh*: a shell script for the automation of the Ubuntu Server setup;
- Activity report and source code.