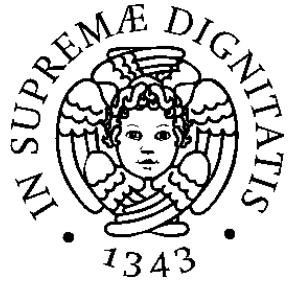


UNIVERSITÀ DI PISA

SCUOLA DI INGEGNERIA
Dipartimento Ingegneria dell'Informazione
Corso INTERNET OF THINGS
MSc in ARTIFICIAL INTELLIGENCE AND DATA ENGINEERING

REPORT ACCADEMIC PROJECT FOR IMPLEMENTING A IoT SYSTEM

PRESENTATO DA:

CALABRESE Pietro

Anno Accademico
2022/2023

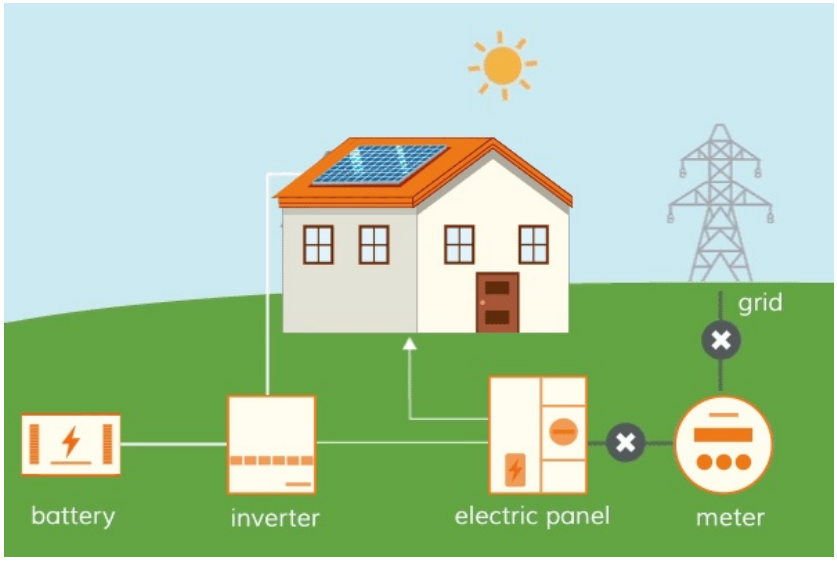
Contents

Chapter 1

Introduction

A home battery might be called "smart" because it connects to an app that allows the homeowner to track their usage. We take a look at how we can make a battery smarter by connecting it to external inputs in order to enhance its performance and preserve its life.
A typical home battery will be charged when a household produces more energy than it consumes, e.g. when solar panels are producing electricity. The battery will then be used when there is enough load available to replace the direct consumption from the grid. This is an important task, especially nowadays we are moving towards an increasingly intense use of renewable funds, even for small plants. attention must also be paid to the entire energy production and consumption process. this is why batteries play an important role. Extending their life allows us to reduce the exploitation of rare raw materials for their production. the most important parameters are the state of charge and the internal temperature of the battery. starting from these two parameters it is possible to implement a system to manage the charging and discharging cycle in an optimal way.
Furthermore, with the use of increasingly low-cost information and sensor technologies, it is possible to create a completely automated solution that is able to manage itself autonomously and with minimal effort on the part of the user.

The aim of this project is to create a smart system that autonomously manages a smart battery, this through the use of IoT technologies.



As mentioned previously, there is a need to acquire all the values that describe the environment and the physical conditions in which plants live, for this project the following sensors were used for educational purposes:

- **Battery Sensor**: it samply collects all the internal parameters from the battery, in this case SoC and Internal Temperature.
- **Rele Actuator**: it is a smart switch that enable the energy flow from an external source.

Chapter 2

IoT Architecture

2.1 Schema IoT Architecture

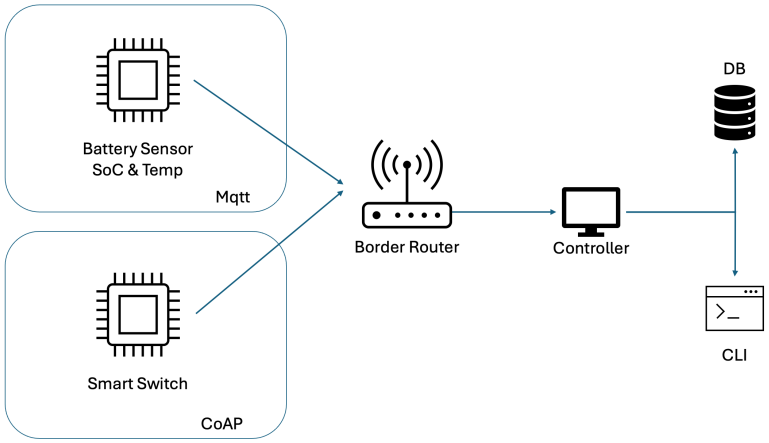


Figure 2.1: Architecture of the System

In the figure above, Figure 2.1, is represented the IoT architecture. We have a series of sensors that exploit different protocols to communicate and receive data. These data is received/sent also by a Python Application. It collects the received data and sends these data to a database and to a CLI.

2.2 Sensor and Application Protocols

The system uses two different networks for the IoT devices. The first is the one that uses MQTT to retrive data, the other one uses CoAP to communicate with the devices.

All system is made to be deployed on real sensors, specifically I used the nRF52840 Dongle. This sensor has not all sensors to collect the data, but using the led and the switch on the board I tryed to mimic the changements and the interaction of the system.

The broker MQTT is deployed on the machine. While CoAP network, is deployed on the sensors that expose some resource and whit the application can communicate directly.

The Collector is the main module of the system. It is implemented in python using the libraries CaAPThon 3 and Paho. The Collector collects data from CoAP and MQTT sensors and stores in SQL DB, implemented using MySQL. It also manages the logic of the system, it changes the state of the valve actuators based on the data it collects from the sensors and acquires the data from the sensors at pre-established intervals. This also provides the user with a CLI through which they can view data and system status, as well as set parameters with which the system manages its logic.

In a real environment, the data acquisition interval could be set between 30 and 60 minutes. For the academic case of this project it is set to 30 seconds.

2.3 Data Encoding

For the data exchange we used the well known JSON data format. The main reason is that, being in an IoT environment, therefore with constrained devices, the JSON format allows us to send less and faster data, which translates

into lower energy consumption and the possibility of real-time monitoring. However, the energy constraint can be overcome since in the environment where the sensors work, we foresee the possibility of a continuous power supply. Furthermore, the data we exchange is primitive, simple data and does not require validation given the environment lacking requirements such as data integrity and security.

The Bettery Sensor sends its data using the following format:

```
{
  "node": "node_id",
  "SoC": soc,
  "Temp": temp
}
```

Chapter 3

Data Storing

3.1 Database

All the data collected by the client is stored on the "electromodule" database using the well-known relational database management system MySQL.

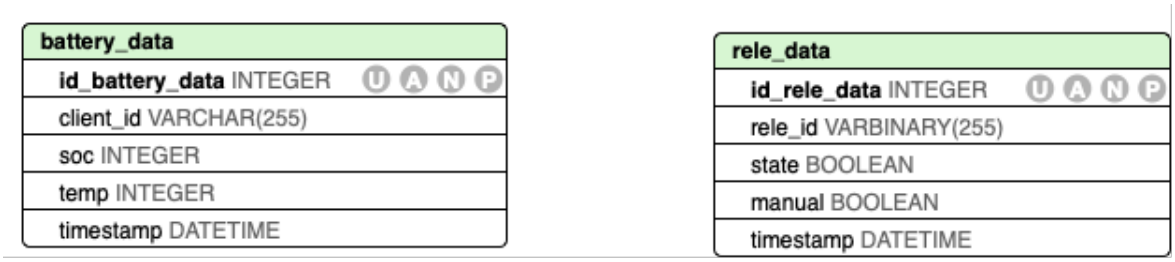


Figure 3.1: Database

To maintain a history of the acquired data within the database, I used a fictitious ID for each record within the entity. Since it is a very simple application, there are no dependencies between the various database entities.

There are 2 entities:

- battery_data: which stores all data produced by the sensor inside the battery

- rele_data: Maintains the state of the smart switch.