Politecnico di Torino

"An Integrated Web Platform for Remote Control and Monitoring of Diverse Embedded Devices: A Comprehensive Approach to Secure Communication and Efficient Data Management"

Master's Degree in Computer Engineering

Roberto Caschetto (302723)

Supervisors:

Prof. Maurizio Rebaudengo
Dr. Edoardo Giust
Ing. Paolo Doz

abinsula

Politecnico di Torino

Table of Contents

1 Introduction

- Introduction
- Frontend
- Backend
- Data Persistence
- Devices
- Performance Evaluation
- Conclusions

Problems to overcome

Problems to overcome:

- Control a variety of embedded systems spread across the country
- Monitoring of the collected data of the different devices
- Update control flow of the single devices
- Single frontend interface to perform the different control and monitoring operations

Solution's Architecture Composition

The solution's architecture has been divided into different main components:

- Frontend interface: The web page allowing the users to interact with the entire system in an abstract and simplified way.
- Backend servers: The core of the architecture, handling all the communication between the different components of the system, allowing also the retrieval and storage of data to/from the databases.
- Databases: The data persistence location where all the data collected and other information is stored.
- Devices: The main actors collecting data used for generating statistics and visualizations to the frontend.

Table of Contents

2 Frontend

- ▶ Introduction
- ▶ Frontend
- ▶ Backend
- ▶ Data Persistence
- ▶ Devices
- ▶ Performance Evaluation
- ▶ Conclusions

Objectives

2 Frontend

The frontend provide a web interface to:

- Track the information of the single company devices, models and products.
- Show analytics of the data collected by each sensor installed in the devices.
- Handle device firmware updates.
- Remote control of the device using custom-made commands.

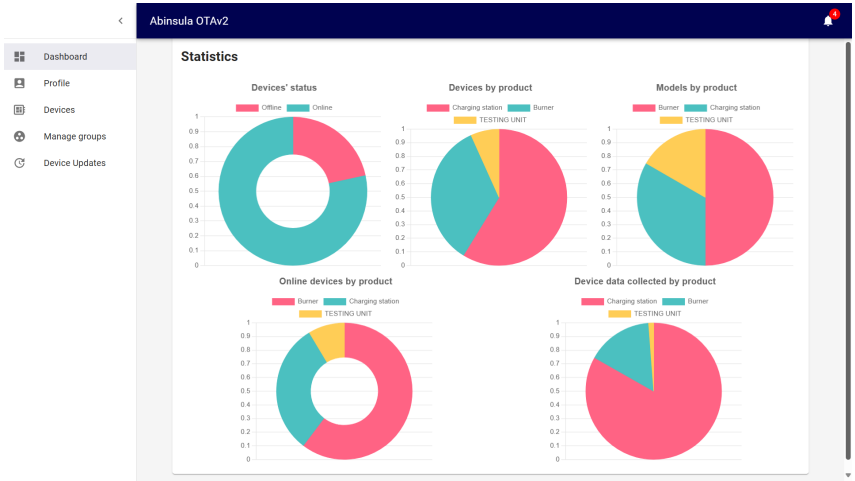
Challenges and solutions

2 Frontend

Challenge	*Solution*
Show different information of all the devices, models and products available	Provide a frontend interface grouping and visualizing different information
Create new updates with specific information for only compatible devices	Provide a Page to create updates specifying compatible versions and devices
Accessible way to update groups of compatible devices	Provide a Page to create groups to be selected in the updates' page

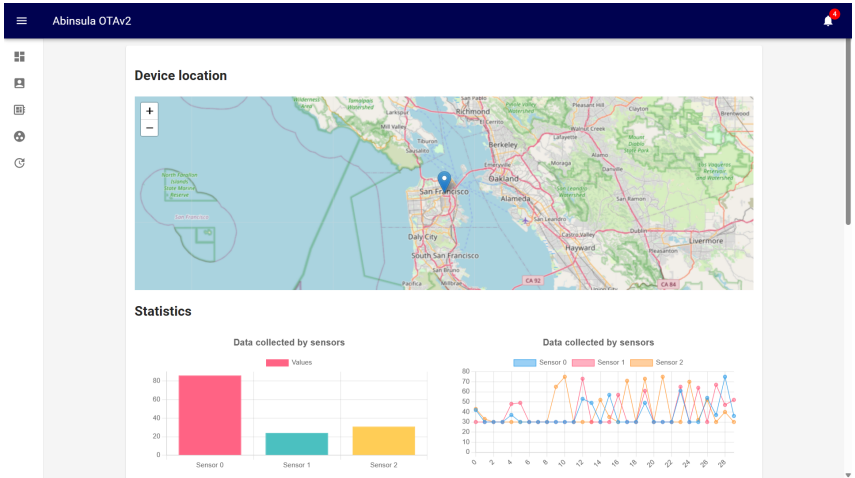
Frontend dashboard's view

2 Frontend



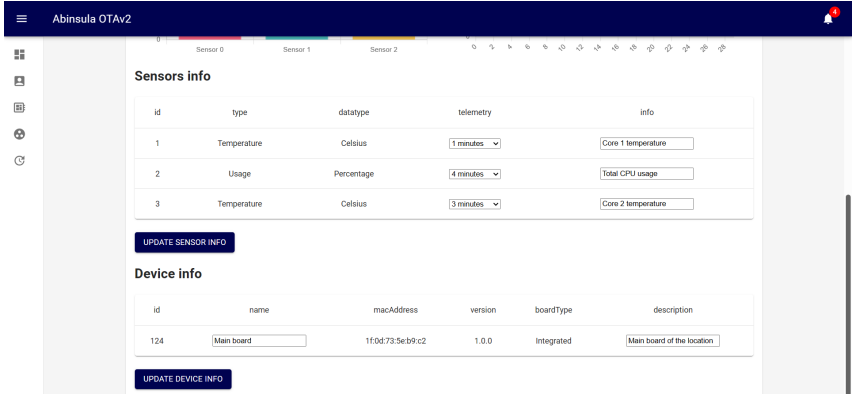
Frontend device's statistics view

2 Frontend



Frontend device's statistics view 2

2 Frontend



Frontend device's statistics view 3

2 Frontend

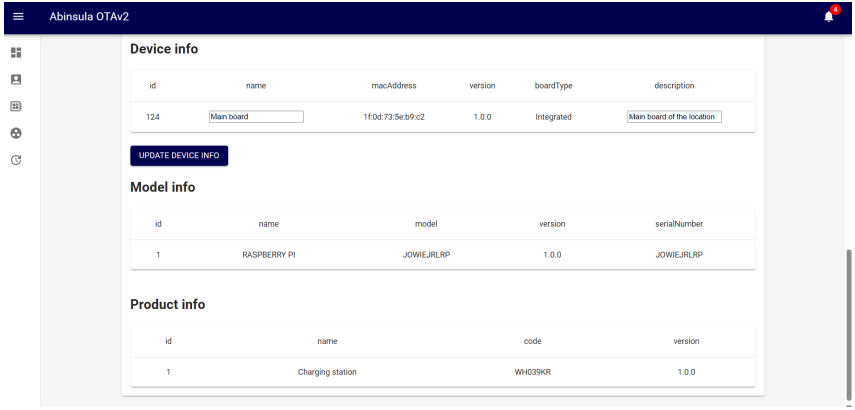


Table of Contents

3 Backend

- Introduction
- Frontend
- Backend
- Data Persistence
- Devices
- Performance Evaluation
- Conclusions

Challenges and solutions

3 Backend

Challenge	*Solution*
Transmition of custom commands from frontend to devices	Communication via Event Emitter between ApiServer and TCPServer
High load of requests	Split the workload between RestAPI server and TCPServer
High computational requirements	Retrieve the required data and leave the computation to the frontend
Recognition of real devices from possible intruders	Authentication methods with temporary authorization keys

Backend's server composition

Backend composition:

- RestfulAPI Server: For handling API requests to store/retrieve data from/to databases.
- TCP Server: For handling the direct connection between the server and devices for custom command's exeuction.

RestApi Interaction Architecture

3 Backend

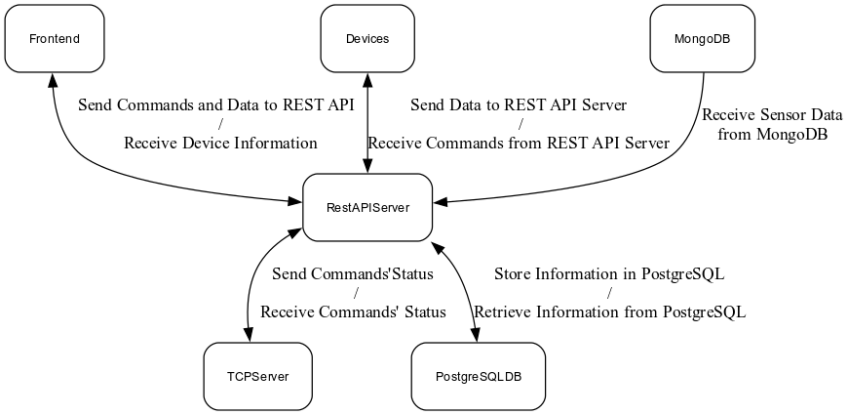


Table of Contents

4 Data Persistence

- Introduction
- Frontend
- Backend
- Data Persistence
- Devices
- Performance Evaluation
- Conclusions

Challenges and solutions

4 Data Persistence

Challenge	*Solution*
Storage of structured data with many join operations	Use of PostgresSQL database
Storage of unstructured data with different and changing fields	Use of MongoDB database
High load of read/write operations for some information	Use of Redis for in-RAM caching of information

Data persistence

4 Data Persistence

The architecture uses different storage units to get the best performances and scalability:

- SQL Database (PostgreSQL): Used for storing structured data.
- NoSQL Database (MongoDB): Used for storing unstructured data.
- In-RAM Database (Redis): Used for storing transient and frequently requested data.

Table of Contents

5 Devices

- ▶ Introduction
- ▶ Frontend
- ▶ Backend
- ▶ Data Persistence
- ▶ Devices
- ▶ Performance Evaluation
- ▶ Conclusions

Challenges and solutions

5 Devices

Challenge	*Solution*
Communicate safely with the backend	Use encrypted TLS channel
Reduce the packet size for improved performances in data transmission	Use of Message-Packs
Identify the device	Use of a configuration file with device's info
Remember the already identified devices	Use of a temporary key generated at first connection
Check periodically for updates and keep TCP connection with the backend alive	Use of threads for specific periodic operations

Device Capabilities

5 Devices

The client has been developed in C to run on different kinds of systems and allowing the device to perform different operations:

- Authenticate and register the device via the RestfulAPI server.
- Look for new updates via the RestfulAPI server.
- Transmit the data collected by the sensors via the RestfulAPI server.
- Receive custom commands and transmit the commands' results via the TCP server.
- Transmit pings to update the device's connection status via the TCP server.

Device's Configuration File

5 Devices

```
{
  "serialnumber": "124",
  "product": "1",
  "shared_password": "secretpassword",
  "model": "JOWIEJRLRP",
  "version": "1.0.0",
  "location": {
    "latitude": 37.7749,
    "longitude": -122.4194
  },
  "authkey": "secretkey",
  "update_frequency": 1000,
  "heartbeat_frequency": 1000,

  "sensors": [
    {
      "sensor_id": 1,
      "sensor_type": "Temperature",
      "data_type": "Celsius",
      "sensor_path": "/sys/class/thermal/thermal_zone0/temp",
      "max_frequency_of_transmission": 480,
      "actual_frequency": 60
    },
    {
      "sensor_id": 2,
      "sensor_type": "Usage",
      "data_type": "Percentage",
      "sensor_path": "/proc/stat",
      "max_frequency_of_transmission": 420,
      "actual_frequency": 240
    },
```

Table of Contents

6 Performance Evaluation

- Introduction
- Frontend
- Backend
- Data Persistence
- Devices
- Performance Evaluation
- Conclusions

Performances

6 Performance Evaluation

In the end, many performance tests have been performed during the different phases of development.
The results were always satisfying also on the final versions as depicted in the figure on the right.
The final performances were extracted during a stress test on medium load with few hundreds devices sending data to the server at the same time.

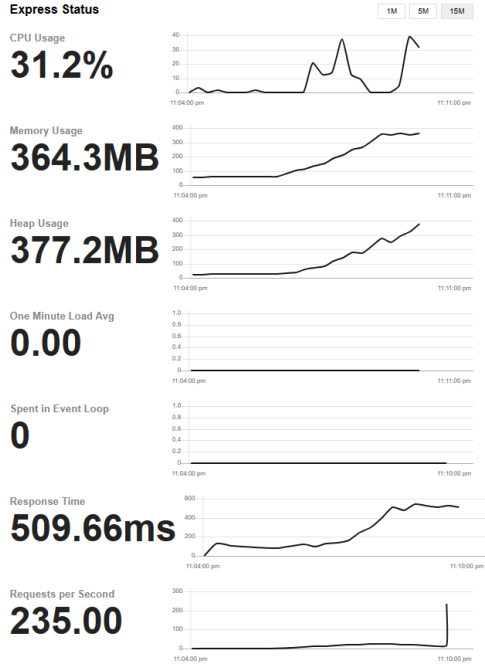


Table of Contents

7 Conclusions

- ▶ Introduction
- ▶ Frontend
- ▶ Backend
- ▶ Data Persistence
- ▶ Devices
- ▶ Performance Evaluation
- ▶ Conclusions

Conclusion

7 Conclusions

Overall, the architecture achieved the control and monitoring of the different devices connected and spread around the country, overcoming the different difficulties encountered. It is providing a user-friendly interface to the user, abstracting the complex operations performed by the different components allowing a direct interaction among them, in an easy way.

Thank you for listening!