

Practical Report Group 71: Parking System using Raspberry Pi and Eclipse Leshan Framework

Internet of Things (2IMN15) Academic Year 2018-2019 Eindhoven University of Technology

Group Members

No	Name	Student ID	Email	Master Program
1	Anoop Krishna Ravattu	1336444	a.k.ravattu@student.tue.nl	Embedded Systems
2	Keyshav Mor	1237978	k.s.mor@student.tue.nl	Embedded Systems

Table of Contents

Group Members.....	1
System Description	3
System Interfaces.....	5
Implementation	6
Testing.....	11
Discussion of Extra-functional Properties.....	16
Evaluation	17
Reflection	17
Contribution.....	18

System Description

A Smart Parking system has been attempted to be developed using Raspberry Pi and Eclipse LWM2M Leshan framework. The system consists of the following features:

- Communication and data exchange between Leshan LWM2M Server and Leshan LWM2M Client to reserve the spot or update the state of the spot.
- Eclipse Leshan is an OMA Lightweight M2M (LWM2M) implementation in Java.
- Leshan relies on the Eclipse IoT Californium project for the CoAP and DTLS implementation
- Constrained Application Protocol, CoAP as defined under RFC 7252, is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. • In the project, the parking spot (implemented/symbolized by Raspberry Pi) is implemented as the Leshan client whereas the parking server acts as Leshan server (which is implemented on the Ubuntu Virtual Machine). Server acts as broker for the services provided by the client.
- Registration and service discovery by the parking spot uses JmDNS, a *Java* implementation of multicast DNS.
- The communication between the Leshan server and the Leshan client uses CoAP.
- A database is created using SQLite on the server side to implement the operations of database, such as insert, delete, read, etc. Integrate the operation in the server-client CoAP application. We have tried to integrate this database in the server-client interaction to streamline read and store data back and forth from the user interface.
- Plate recognition is done by Raspberry Pi (with its camera module) on the client side and the image processing of the number plate is also done on the client side.
- Additional HTML page has been integrated with the Leshan server and is used as Graphical User Interface (GUI) for user convenience.
- XML object files are created to ensure registration and visualization of features for the services provided by the client on the server.
- Open Automatic License Plate Recognition (ALPR) and Splinter are used to do license plate recognition and filtering out text from the image to send the data to the server.
- The deployment diagram of the parking system in general is given as Fig.1.

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK –
2IMN15

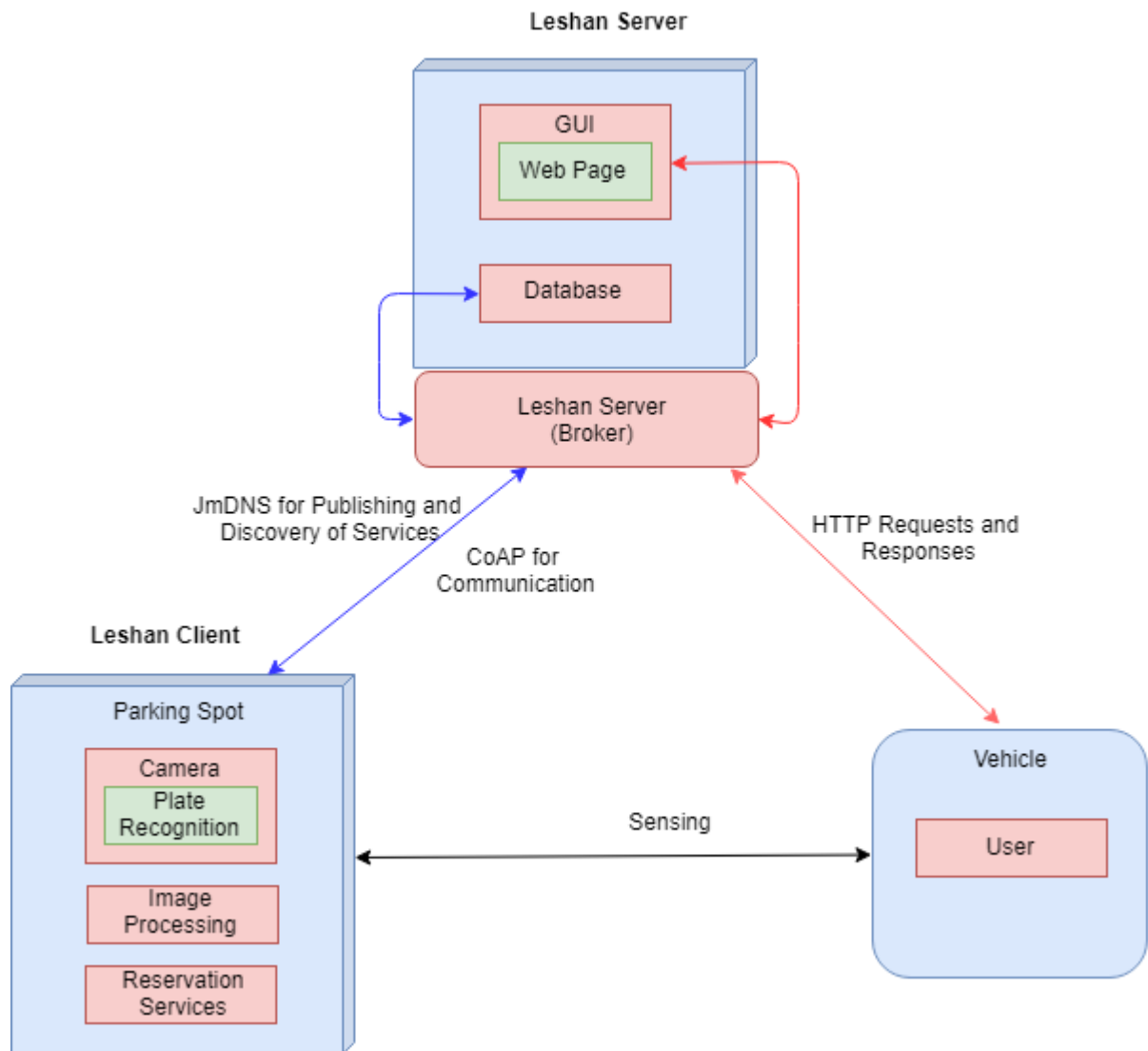


Fig.1

System Interfaces

Our Parking system consists of three main parts:

1. Server
2. Parking Spot
3. Vehicle

The Parking Spot has certain features like Spot ID, Spot State, Vehicle ID, Billing Rate and Vehicle Number Plate Detection. These are the services available from the client and are registered on the server using JmDNS, a Java version of multicast DNS.

The Server acts as a broker for these services. The server also hosts a webpage for the convenience of user to access these services.

The user simulates the driver of the vehicle and interacts with the web page to access these services.

The webpage offers the client, the opportunity to access these services and also make changes to the SQLite database. The actions of the user on the client side triggers some additional processes on the client from the Raspberry Pi side.

The client acts as an interface for image processing and number plate matching actions. Open ALPR is the framework used to process text from the image of the number plate of the vehicle.

Implementation

For the implementation of this project we have used the Eclipse Leshan framework based on Java, CoAP protocol for communication, HTML pages for User interaction, SQL databases for storing and managing data, OpenALPR framework for License Plate recognition and Python for extracting relevant data from License plate JSON file.

We also tried and were partially successful in using Splinter framework for matching the license plate number in the Dutch Number Plate database, but due to the restrictions in time and problems in integrating the retrieved information in the current data flow framework of the project, we have not included it as an additional implementation point of the project.

The initial task was to setup the Raspberry Pi. We first allowed the Raspberry Pi to access the internet connection of the laptop by enabling the Wi-Fi connection on the laptop to share it with the Raspberry Pi over ethernet. Following this, we used the Angry IP scanner to scan the IP address that is currently used by the Raspberry Pi and opened an SSH connection with it. Using the login credentials given to us by TU/e, we logged in and modified the password and the hostname to “kingjulien” and “RaspberryPi121IoTParking” respectively.

We extensively used Tightvncserver and VNCviewer to access the GUI of the Raspberry Pi to program Raspberry Pi specific software.

Following this we downloaded the Leshan repository from the GitHub and built it to generate the leshan-server-demo and leshan-client-demo JAR files. After running these, we understood how the registration of services is taking place and how the objects registered by the client are interacting with the user through the webpage hosted by the server.

We then modified the JSON object files for the parking spot to XML format so that we can include this within the Leshan server and client environments in a way that these services are integrated in the client and are visible on the webpage of the server. Supporting Java files were written and included in the client code to ensure that these services interact smoothly over CoAP.

The Server was also modified to include the SQL database and create a table readable and modifiable by Java environment. This helped the user to make changes to the database when and as required. While the database could be created, the read and modified interactions were not always successful and did give us some Timeout errors once in a while.

Following this we created an HTML page in order to create a dedicated webpage for the Parking Spot reservation which will interact with the client through the server over Leshan’s framework. We were not entirely successful with this attempt and thus resorted to use the original webpage which comes inbuilt with Leshan.

We then installed OpenALPR library on the Raspberry Pi for License Plate recognition and extract textual information from the number plate of the car. We tried reading several number plates using this and were satisfied with the results. We used ALPR to detect the number plate of the

Vehicle ID which has been already added by the user on the Webpage. ALPR then generated a JSON file which had to be read using a Python code to extract the Highest Likely Candidate (usually data with confidence higher than 90%). This Number plate was then stored in a text file.

The text file was then read by the Java code and when the User clicked “Read” on Vehicle Number Plate Image on the Webpage and displayed the text it reads from the text file. The text matched the Vehicle ID entered by the user.

We also tried and were partially successful in using Splinter to check database of Dutch license plates, but sharing and parsing data was getting complicated and restrictions of time was the reason we could not integrate it in a full-fledged manner. However, we found all the process extremely interesting and thoroughly enjoyed working with it.

A detailed functioning of the project has been demonstrated extensively in a video which can be found on the following web-link :

The steps on the interaction are as follows and have been elaborated with the screenshots which are given in the Testing section :

1. Connect Raspberry Pi to Laptop over SSH and Log-In
2. Open Ubuntu Virtual Machine
3. Start Leshan Server on Virtual Machine
4. Start Leshan Client on Raspberry Pi
5. Open Web browser and start <http://localhost:8080>
6. Check if the client “RaspberryPi121IoTParkingSpot” is registered on the server
7. Click on the registered client
8. Check for Parking Spot Service
9. Look for Parking Spot ID, Read Spot State, Modify State to Reserve/Free, Modify Vehicle ID, Billing Rate & Vehicle Image Number Plate.
10. The action of “Reserve” triggers image capture and OpenALPR filters out text from this captured image of the number plate. For convenience, we have used a downloaded image due to lack of Raspberry Pi camera clarity (using raspistill command) and also assume that Vehicle occupies the spot immediately after reserving. We are aware realistic scenarios are slightly different and that in a case of an ideal camera, the image can be captured in real time.
11. OpenALPR outputs a JSON file. Python code filters out High Likely Candidate text from the JSON file and stores in a text file.

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK –
2IMN15

12. User clicks on Read feature of Vehicle Number Plate Image object on the Webpage of the Server showing the registered client services.
13. This “Read” action triggers the action of the Java code to read the text file generated by the Python code. This text then appears on the Webpage. The text matches the Vehicle Number plate on the image.

The sequential diagram for the Parking System in general is given below as fig.1 and the sequential diagram for the image processing is given as fig.2.

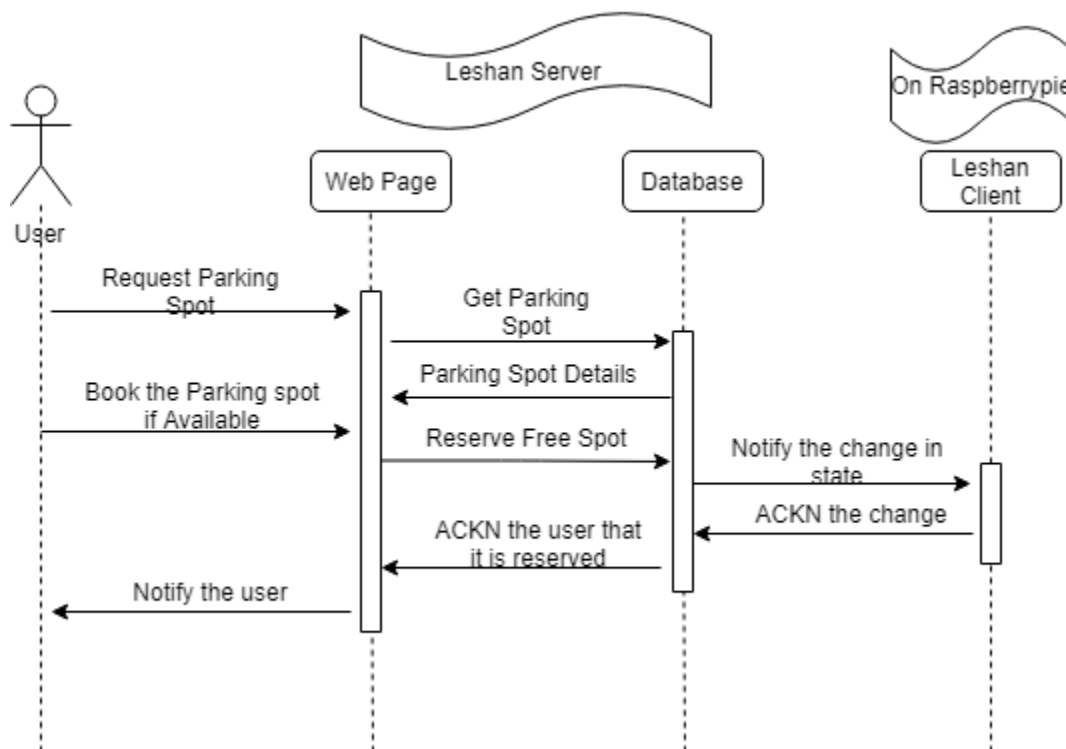


Fig.1

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK –
2IMN15

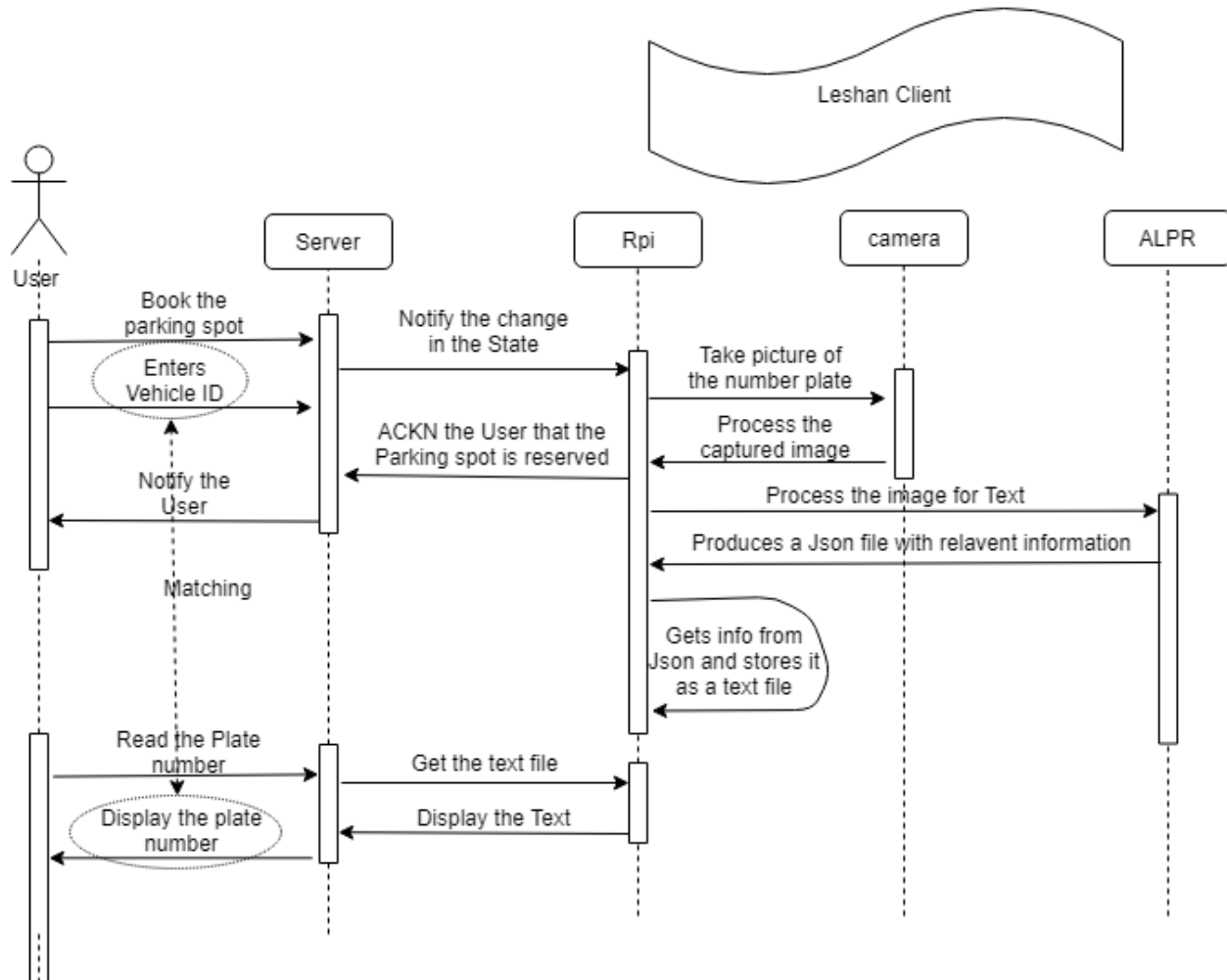


Fig.2

Difficulties:

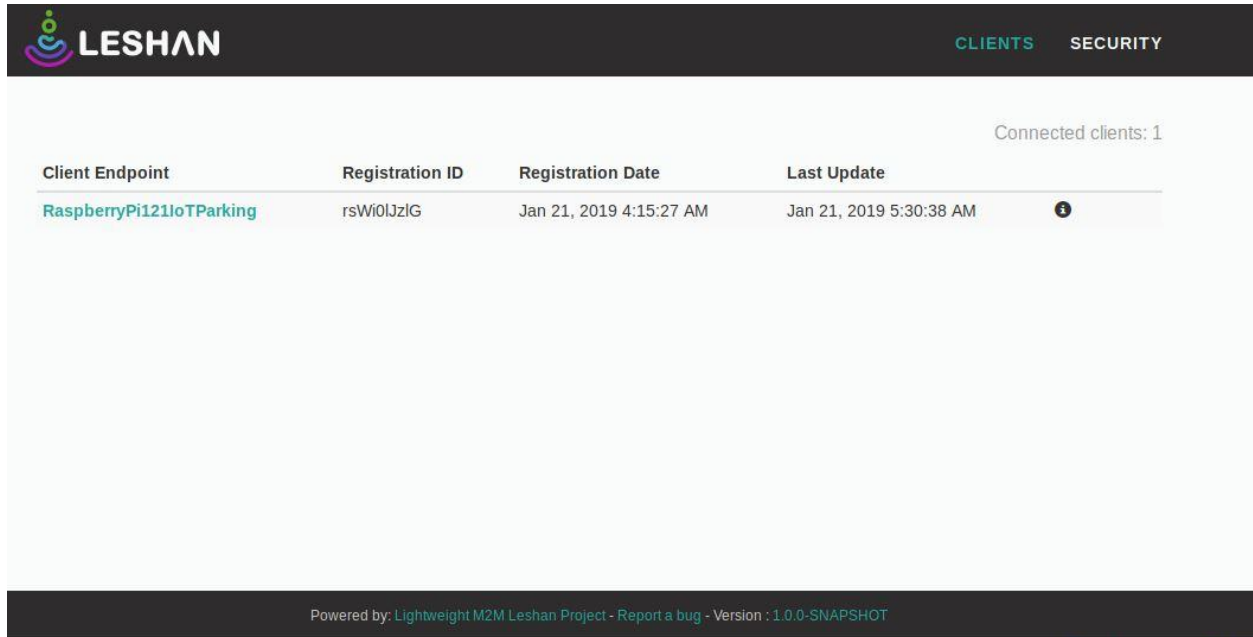
There were quite a few difficulties that we faced during the implementation of the project. We have listed down a few of them.

1. There were certain troubles faced initially to setup and establish communication between CoAP server and CoAP client which came with txThings. The tutorials provided were not really helpful to troubleshoot this problem. This made us switch to Leshan.
2. It was difficult to integrate the SQLite database seamlessly as expected by us. But this was not a major hurdle.
3. It was difficult to understand the interaction of Leshan Server and Client codes with supplementing HTML, XML and Java Servlet files. A lot of time was spent on understanding these. A tutorial on explaining how Leshan interacts with its various subcomponents and its dependencies would have been helpful to make an even better project.
4. Inexperience with Java, SQL and HTML was also a big problem in making them all work seamlessly together.
5. Splinter was interesting to use and partially successful too, but we could not implement it and include it in our final demonstration due to lack of time and complications in triggering script and sharing data. We would continue exploring this to make it a part of the improved project.
6. Writing the server, client, image processing and data retrieval all are extremely interesting experiences but time consuming. The sheer scale of the project was too big to be completed by 2 people in the given time.
7. For some reason, it was difficult to install new packages on Raspberry Pi which took long to be corrected. Also it was difficult to build Leshan on Raspberry Pi and we spent a lot of time in resolving this issue. Without these problems we could have made a slightly more complex project.

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK – 2IMN15

Testing

Following are some screenshots which pertain to our demonstration and testing of the project :

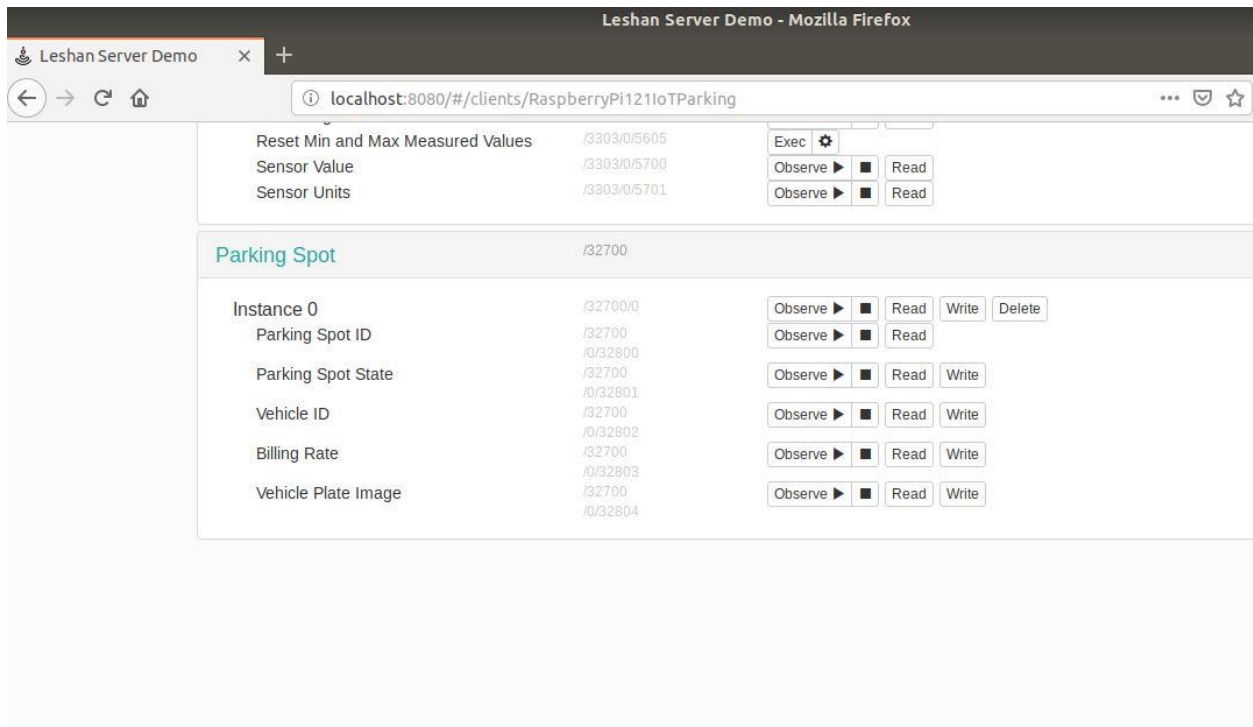


The screenshot shows the LESHAN web interface. At the top, there is a header with the LESHAN logo and navigation links for CLIENTS and SECURITY. Below the header, a status bar indicates "Connected clients: 1". A table lists the client details:

Client Endpoint	Registration ID	Registration Date	Last Update
RaspberryPi121IoTParking	rsWi0JzIG	Jan 21, 2019 4:15:27 AM	Jan 21, 2019 5:30:38 AM

At the bottom of the interface, a footer states: "Powered by: Lightweight M2M Leshan Project - Report a bug - Version : 1.0.0-SNAPSHOT".

Registration of Client



The screenshot shows the Leshan Server Demo web interface in a Mozilla Firefox browser. The address bar displays "localhost:8080/#/clients/RaspberryPi121IoTParking". The interface is divided into two main sections:

Client Registration Details:

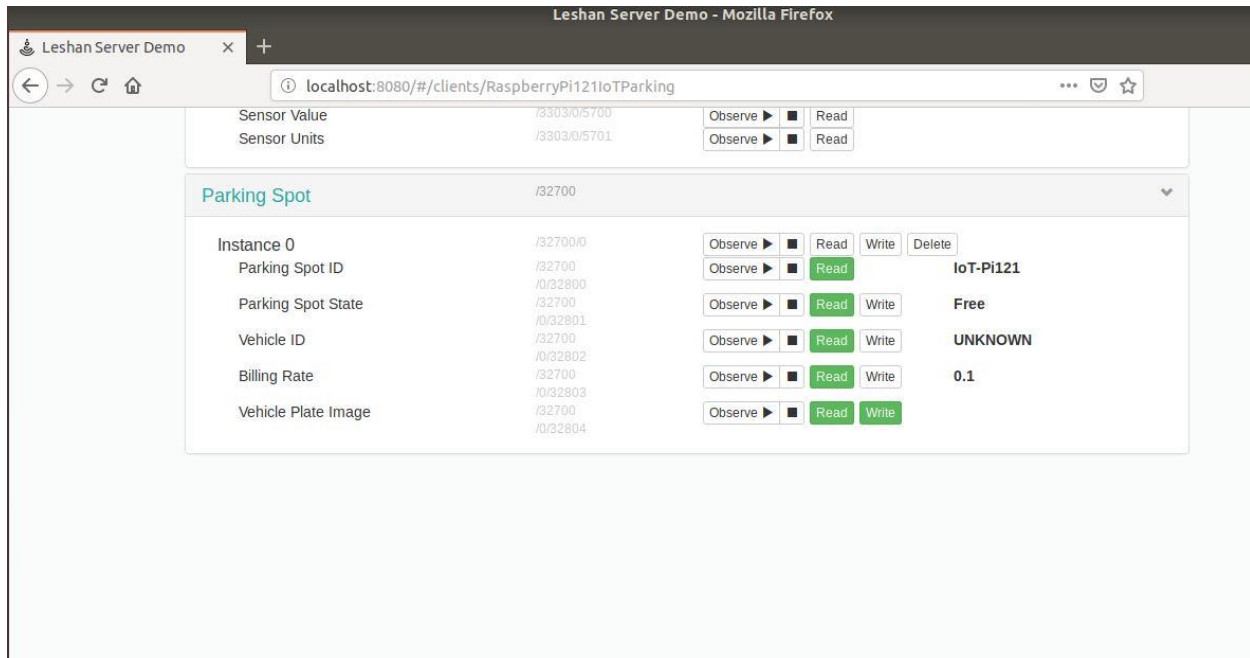
Property	Value	Actions
Reset Min and Max Measured Values	/3303/0/5605	Exec [gear icon]
Sensor Value	/3303/0/5700	Observe [play icon] [stop icon] Read
Sensor Units	/3303/0/5701	Observe [play icon] [stop icon] Read

Parking Spot Details:

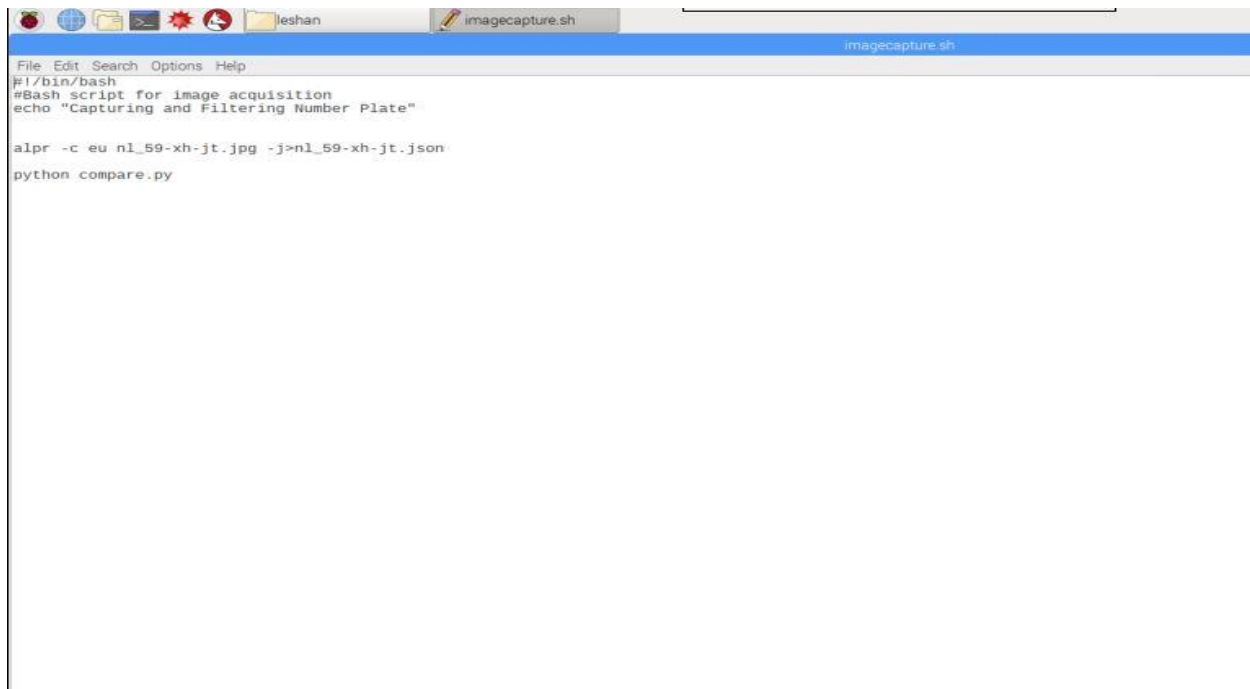
Property	Value	Actions
Instance 0	/32700/0	Observe [play icon] [stop icon] Read Write Delete
Parking Spot ID	/32700	Observe [play icon] [stop icon] Read
Parking Spot State	/0/32800	
	/32700	Observe [play icon] [stop icon] Read Write
	/0/32801	
Vehicle ID	/32700	Observe [play icon] [stop icon] Read Write
	/0/32802	
Billing Rate	/32700	Observe [play icon] [stop icon] Read Write
	/0/32803	
Vehicle Plate Image	/32700	Observe [play icon] [stop icon] Read Write
	/0/32804	

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK – 2IMN15

Parking Spot Registration



Observation of Parking Spot States



Bash script to execute image processing

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK –
2IMN15

```

pi@RaspberryPi121IoTarking: ~/leshan
pi@RaspberryPi121IoTarking:~/leshan $ ./imagecapture.sh
Capturing and Filtering Number Plate
{u'data_type': u'alpr_results',
 u'epoch_time': 1548096406031L,
 u'img_height': 288,
 u'img_width': 440,
 u'processing_time_ms': 348.15271,
 u'regions_of_interest': [],
 u'results': [{u'candidates': [{u'confidence': 94.443581,
                               u'matches_template': 0,
                               u'plate': u'S9XHJT'},
                              {u'confidence': 84.954819,
                               u'matches_template': 0,
                               u'plate': u'S9XHJT'},
                              {u'confidence': 79.121521,
                               u'matches_template': 0,
                               u'plate': u'S9XMJT'},
                              {u'confidence': 69.632767,
                               u'matches_template': 0,
                               u'plate': u'S9XMJT'}]},
              u'confidence': 94.443581,
              u'coordinates': [{u'x': 124, u'y': 133},
                               {u'x': 304, u'y': 134},
                               {u'x': 304, u'y': 169},
                               {u'x': 124, u'y': 167}],
              u'matches_template': 0,
              u'plate': u'S9XHJT',
              u'plate_index': 0,
              u'processing_time_ms': 199.157791,
              u'region': u'',
              u'region_confidence': 0,
              u'requested_topn': 10}],
 u'version': 2}
59XHJT

```

Activating Script to capture number plate and filter out text

```

1 import json
2 from pprint import pprint
3
4 with open('nl_59-xh-jt.json') as f:
5     data = json.load(f)
6     pprint(data)
7     string=data["results"][0]["plate"]
8
9
10 print(string)
11
12 f=open("matched.txt","w+")
13
14 f.write(string)
15
16 f.close()
17

```

Python script to filter out correct number plate from JSON file

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK – 2IMN15

```

{ try {
    BufferedReader in = new BufferedReader(
        new InputStreamReader(
            new FileInputStream("PIPE.txt"), "UTF8"));

    String str;

    while ((str = in.readLine()) != null) {
        if(!str.equals("UNKNOWN"))
            updatevehnum(str);
        //FLAG=false;
    }
    in.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

try { String[] cmd={"sh","/home/pi/leshan/imagecapture.sh"};
    Runtime.getRuntime().exec(cmd);
    LOG.info("Will Scan Plate as Vehicle in sensed in the vicinity");
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

```

Java Code snippet to activate image capture script from the Client

```

case PARKINGSPOTSTATE:
    return ReadResponse.success(resourceId,ParkState);
case VEHICLEID:
    return ReadResponse.success(resourceId,ParkSpotID);
case BILLING:
    return ReadResponse.success(resourceId,0.1);
case VEHICLENUMBERPLATE:
    try { FileInputStream fis = new FileInputStream("/home/pi/leshan/matched.txt");
        Scanner in = new Scanner(fis);
        while(in.hasNext())
        {
            PlateState=in.nextLine();
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

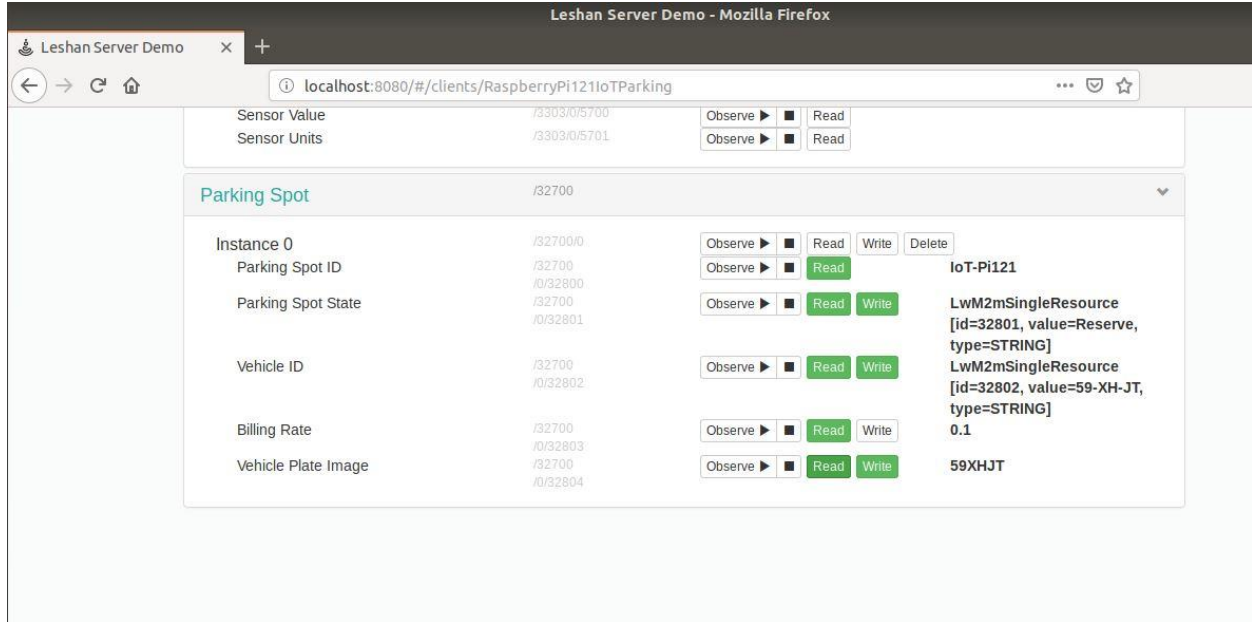
```

Java Code snippet to access the text file generated by Python code for reading JSON file



Image of the Number Plate to be Read.

PRACTICAL REPORT GROUP 19: PARKING SYSTEM USING RASPBERRY PI AND ECLIPSE LESHAN FRAMEWORK – 2IMN15



Reading Image Plate Captured by RPi client

Connected clients: {{clients.length}}

```
var xhttp = new XMLHttpRequest(); var client = "http://localhost:8080/api/clients?"; var clientstatus = "http://localhost:8080/api/clients?"; var vehicleID = "/32700/0/32802" var objectid = "/32700/0/32801" var parkingspotID = "/32700/0/32800" var vehiclenumberplate = "/32700/0/32804" var parkingspotname = []; var parkingspotstatus = [];
```

Parking Reservation system

See parking spaces:

ParkingSpot ID **Vehicle ID**
 {{client.endpoint}} {{objectid}} {{}} {{client.lastUpdate | date:'medium'}}
 {{error}}

The unsuccessful HTML page which could not be integrated successfully with Leshan Server

Discussion of Extra-functional Properties

Here we discuss about the Extra Functional Properties namely Performance and QoS of the system.

Due to the usage of LWM2M protocol which is on top of COAP protocol, which is a replacement for HTTP for constrained devices (where energy matters a lot). HTTP packets are heavier and sits over TCP. Thus, increasing the size and count of message packets over the air. On the other hand, COAP which has shorter packets and sits over UDP layer. Thus, decreasing the number of packets and increasing the throughput with lesser processing time. Parking spot which is considered here as an IOT device needs to have better energy factor and good processing time. This is obtained by using LWM2M as an application layer in the OSI model. Further, the latency calculation and performance evaluation can be calculated here. And due to smaller and lesser packets with constrained meanings over the air, the latency value will decrease to a larger extent. [1]

For the discussion of extra functional properties, apart from the lecture slides on “Architecture” from the course Internet of Things(2IMN15) we also took a reference from one of the slides in Architecture of Distributed Systems (2IMN10) [2]

Manageability: Although there is only one parking spot in our system, we can add additional parking spots to our system. Our parking system is designed in such a way that it can manage different parking spots at once

Expandability: the database is created in such a way that it can accommodate more than one parking spot.

Consistency: The results we obtain from our system are consistent and do not change or get corrupted as time progresses. As long as the client and server are active and no reset happens, data consistency is maintained.

Correctness: The system is designed in such a way that it will always produce the correct results. For example, if the parking spot is not reserved or occupied, the state of the parking spot in the in the Webpage is always shown as free.

Reliability: The system retains all its functionalities and meets expectations of the user in all scenarios.

Modularity: All the components that are designed for the mentioned parking system are modular. For example, if there is a malfunction with the parking spot(Raspberry Pi) it can be replaced. If the created database is at its full capacity and cannot accommodate any more new parking spots, a new database with more capacity can easily replace the old one.

Evaluation

Overall, we are quite satisfied with our efforts invested in the project and also the results obtained from it. We believe that had the difficulties faced by us not been prolonged or had they not been present, we could have made an even more complex and aesthetically appealing demonstration of the project. We are also quite pleased that whatever we have implemented in working and giving reliable results. We believe that without the problems faced and with some more knowledge of Java, XML, HTML and Splinter framework, we could have implemented a couple more points from the specified tasks.

Reflection

We learnt deeply about CoAP and how it is implemented for constrained devices in an IoT architecture. We understood how defining a data and control flow is extremely important before writing the code of any software and more importantly and IoT application.

We learnt how IoT applications interact with one another, how services interact with the backend and how these services work seamlessly to give a unified experience to the user.

We learnt how complex it is to write a complex IoT application and how small mistakes in any of the dependencies of the code can lead to error in functioning of the code.

We learnt new tricks and techniques which are useful in a Linux environment be it Ubuntu or Raspbian.

We got a chance to polish our Python and programming skills to write clean software.

We learnt how to do simple image processing and the experience of working with Splinter framework was really interesting and most insightful part of the project apart from the Leshan framework.

We also learnt how important team-work is and how projects can be unsuccessful if there is no collaborative spirit among team members.

Contribution

Keyshav Mor :

- Setting up of Raspberry Pi
- Leshan Server, interaction with SQL Database, Object specification in XML, HTML code for Webpage
- Writing supporting Java servlet files for Parking Spot for Leshan Client
- Bash scripting.
- Image Processing, OpenALPR , Python Code to extract Number Plate from JSON file
- Testing and experimenting with Splinter framework
- Video Demonstration
- Report

Anoop Krishna Ravattu :

- Leshan Client
- Report
- SQL Database

Link for the Demo Video and Demo Code:

1. <https://drive.google.com/file/d/1C02429uniwe2G6XeCNbLBS6A2i01KuKv/view>
2. <https://drive.google.com/open?id=1GV4fl6lYl4k6zIKuPLuUJs-yzqQ53867>

References:

[1]. <https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap.pdf>

[2]. https://www.win.tue.nl/~wsinmak/Education/2IMN10/Q1_1819/ADS.IntroArch.pdf