

The Internet of Things - Theory and Implementation

Liam Russell - 15071057

December 8, 2017

Task 1: Sensor Recording/Actuation

The first task of the assignment required a program that would record the values of at least two different sensors and store the sensor name and value on a database or an MQTT server. For this first task my program records data from a Slider and Rotator sensor, storing the data into a mysql database. Figure 1 shows the PhidgetServer up and running, awaiting data to be passed to it.



Figure 1: Sensor server awaiting data

Figure 2 and 3 show the sensorToServer class gathering data for 15 seconds from both the Slider and Rotator sensors. Each time the Slider or Rotator sensor would move, the sensor name and value would be recorded and put into a URL that sends the data to the database.

```
STS-Opening and waiting for 15 seconds

Gathering data for 15 seconds

Sending new sensor value : 179
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=rotate&sensorvalue=179
```

Figure 2: sensorToServer gathering data for 15 seconds

```
Sending new sensor value : 170
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=rotate&sensorvalue=170
Sending new sensor value : 126
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&sensorvalue=126
Sending new sensor value : 179
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=rotate&sensorvalue=179
Sending new sensor value : 137
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&sensorvalue=137
Sending new sensor value : 139
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&sensorvalue=139
Closed Slider and Rotate Voltage Ratio Input
```

Figure 3: sensorToServer gathering data and closing

Figure 4 below shows the sql code that is used to update the table sensorUsage, using userID, the sensor name and sensor value passed from the sensors and a time stamp to show when the data was added the table. Figure 5 shows examples of records in the database, showing that data from the Slider and Rotator can be sent simultaneously.

```

DEBUG: Update: insert into sensorUsage(UserID, SensorName, SensorValue, TimeInserted) values('15071057','rotate','75', now());
DEBUG: Update: insert into sensorUsage(UserID, SensorName, SensorValue, TimeInserted) values('15071057','slider','15', now());
DEBUG: Return response for receiving data OK
DEBUG: Return response for receiving data OK
DEBUG: Update: insert into sensorUsage(UserID, SensorName, SensorValue, TimeInserted) values('15071057','rotate','78', now());
DEBUG: Return response for receiving data OK

```

Figure 4: Updating sensorUsage

15071057	rotate	97	2017-12-07 12:24:29
15071057	slider	30	2017-12-07 12:24:29
15071057	rotate	79	2017-12-07 12:24:29
15071057	slider	18	2017-12-07 12:24:29
15071057	rotate	75	2017-12-07 12:24:29
15071057	slider	15	2017-12-07 12:24:29
15071057	rotate	78	2017-12-07 12:24:29

Figure 5: Recorded data in sensorUsage

The data given by the sensors was output using JSON, Figure 6 shows the code that was used to create GSON and sensorData objects. The sensorData object is passed the sensor name and value, using GSON the sData object is parsed into a JSON object. The JSON object is then output using a PrintWriter.

```

Gson gson = new Gson();
sensorData sData = new sensorData(sensorNameStr,sensorValueStr);
String json = gson.toJson(sData);
response.setContentType("application/json");
PrintWriter out = response.getWriter();
System.out.println("DEBUG: json return: "+json);
out.print(json);
out.close();

```

Figure 6: Outputting data as JSON

Along with the Slider and Rotator sensors, a Motor actuator is connected to the system. In a separate class the Motor is asking the server for data that has been sent to it from the Slider, Figure 7 shows data being received from the server and the Motor moving to the Sliders value.

```

MFS-Opening and waiting for 15 seconds
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&getdata=true
Retrieved data from server: {"name":"slider","value":"120"}
Target Position Reached: 120
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&getdata=true
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&getdata=true
Retrieved data from server: {"name":"slider","value":"86"}
Target Position Reached: 86.0

```

Figure 7: Actuator receiving data

As the data is received from the server in the JSON format, a sensorData object is needed to pass the JSON data into so that only the sensor value element can be selected. Figure 8 shows that after the data is read in and passed into sData, an integer is created parsing

the value that is passed from `sData.getValue()`. That value was then used to move the motors position.

```
String motorPosStr = getFromServer("slider");
Gson gson = new Gson();
sensorData sData = gson.fromJson(motorPosStr, sensorData.class);
int motorPos = Integer.parseInt(sData.getValue());
```

Figure 8: Selecting sensor value

Task 2: Mobile Phone Application

For the second task of the assignment I created a mobile phone application using Android Studio. The application allows the user to read the values of both sensors, send sensor data to the database and activate the motor with the click of a button. Figure 9 shows the UI that was created for the mobile application.

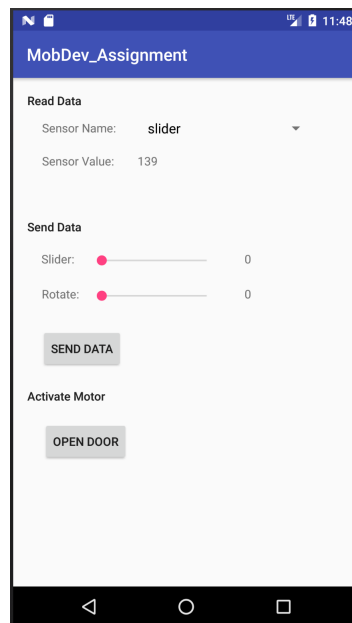


Figure 9: Mobile Application GUI

Figure 10 and 11 shows the application reading data from the server, the sensor name is selected from the drop down box and the the sensors value is shown in the text view below.

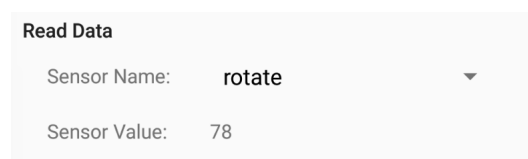


Figure 10: Reading slider value

Read Data


Sensor Name: slider ▼


Sensor Value: 15

Figure 11: Reading rotator value

Data can be sent to the database from the application, this can be done by using the seek bars to select a value for both the Slider and Rotator sensors. The maximum value that can be selected is 180 as it is the maximum value that the Motor can turn to. Clicking the send data button will send the values in the text views to the database along with the name of the corresponding sensor. This can be seen in figure 12.

Send Data

Slider:  126

Rotate:  63

SEND DATA

Figure 12: Sending data

Figure 13 and 14 show that the database has been updated and the application now shows the values that were sent to the database in figure 12.

Read Data

Sensor Name: slider ▼

Sensor Value: 126

Figure 13: Updated slider value

Read Data

Sensor Name: rotate ▼

Sensor Value: 63

Figure 14: Updated slider value

Figure 15 shows the data that was sent by the mobile application in figure 12.

15071057	slider	126	2017-12-07 12:27:39
15071057	rotate	63	2017-12-07 12:27:39
+-----+-----+-----+-----+			

Figure 15: Sent data in database

Figure 16 shows the eclipse program responding when the "Open Door" button is clicked in the mobile application, the button sends the slider value 200. This value is sent as the physical sensors are unable to send this value, this means the door unlocking process won't happen accidentally. When the eclipse project receive the door unlock value, the motor moves and waits 3 seconds before closing to simulate a door locking system giving the user a few seconds to open the door.

```
Sending data to: http://localhost:8080/PhidgetServer/sensorToDB?sensorname=slider&getdata=true

Opening lock to position to 180 for 3 seconds

Target Position Reached: 180

Setting lock closed, position 0
```

Figure 16: Moving actuator from mobile application

Figure 17 below shows the data being received by the application in a JSON format. Similar to Figure 8 in task one a Sensor object is created, using fromJson the sensor name and value are passed to it. The text view's text is then set by using mySensor.getValue().

```
sensorNameField.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        if (String.valueOf(sensorNameField.getSelectedItem()).equals("slider")){
            String valueDB = getSensorData("slider");
            Sensor mySensor = gson.fromJson(valueDB, Sensor.class);
            sensorValueField.setText(mySensor.getValue());
        }
        else
        {
            String valueDB = getSensorData("rotate");
            Sensor mySensor = gson.fromJson(valueDB, Sensor.class);
            sensorValueField.setText(mySensor.getValue());
        }
    }
})
```

Figure 17: Selecting sensor value

Task 3: Smart Home System Diagrams

For task 3 of the assignment I created diagrams for all four of the smart home nodes, these diagrams include the sensors, actuators, hubs and routers. The diagrams have also been annotated with the protocols that the devices use for communication, the power supplies used and useful information about the devices.

Figure 18: Philips Hue System Diagram

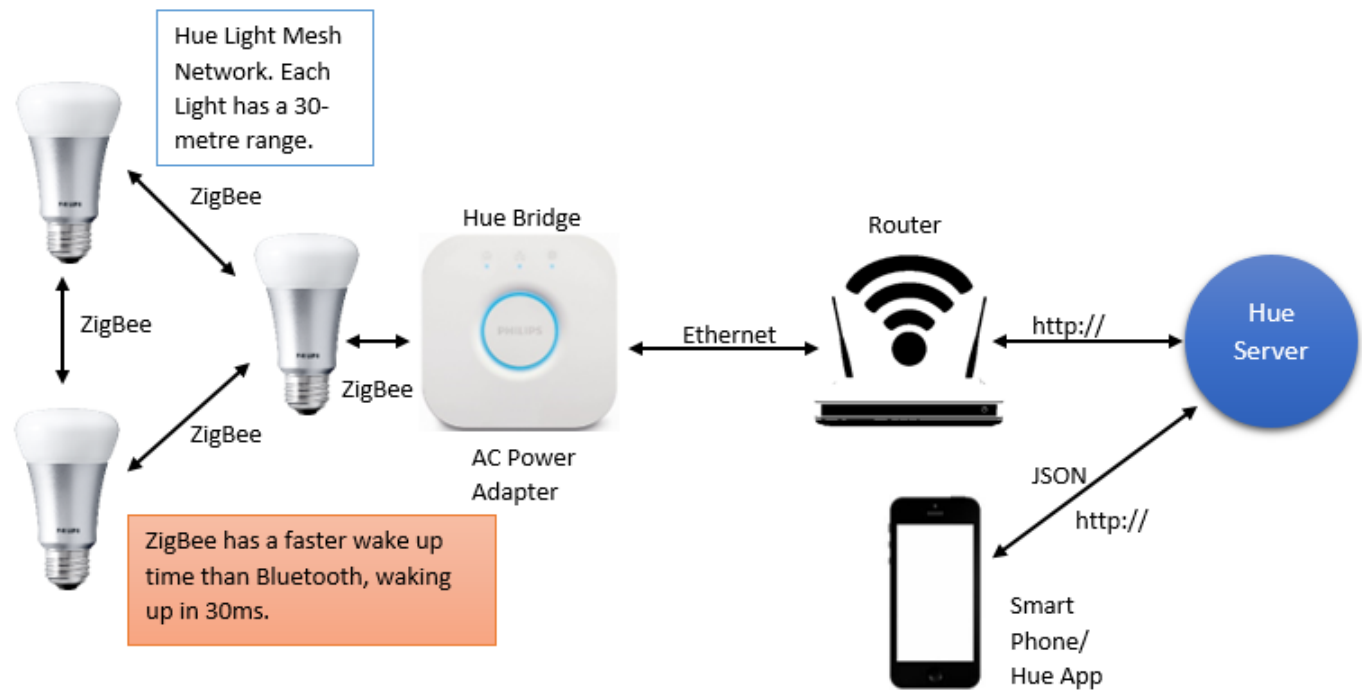


Figure 19: Flic System Diagram

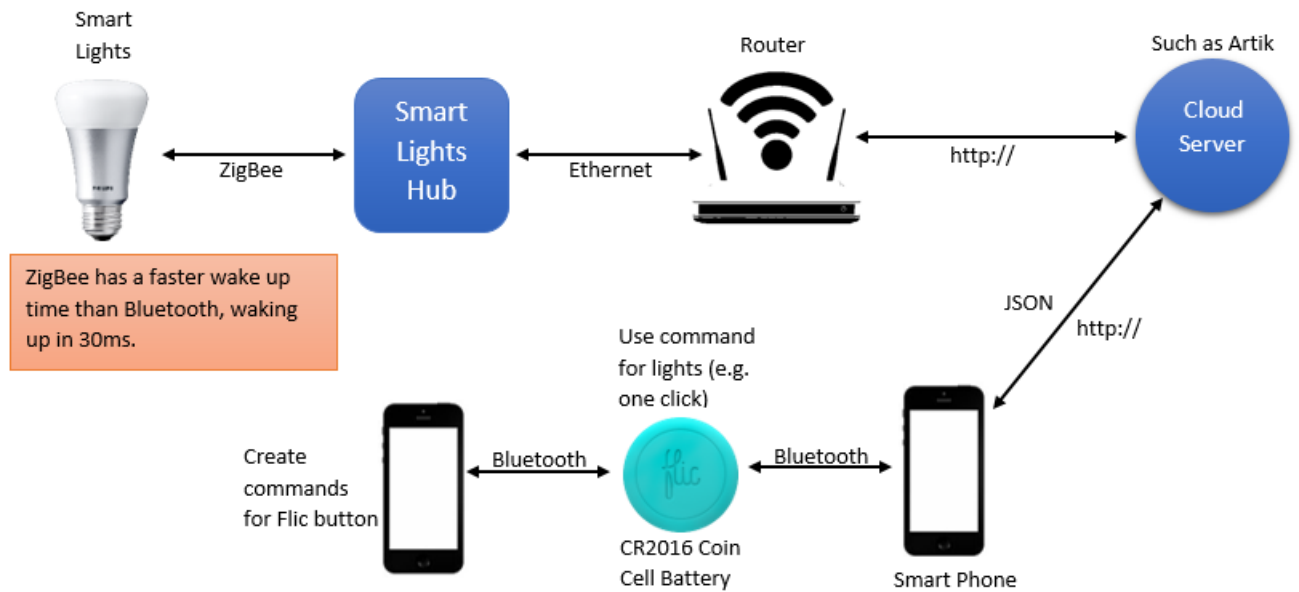


Figure 20: Amazon Echo System Diagram

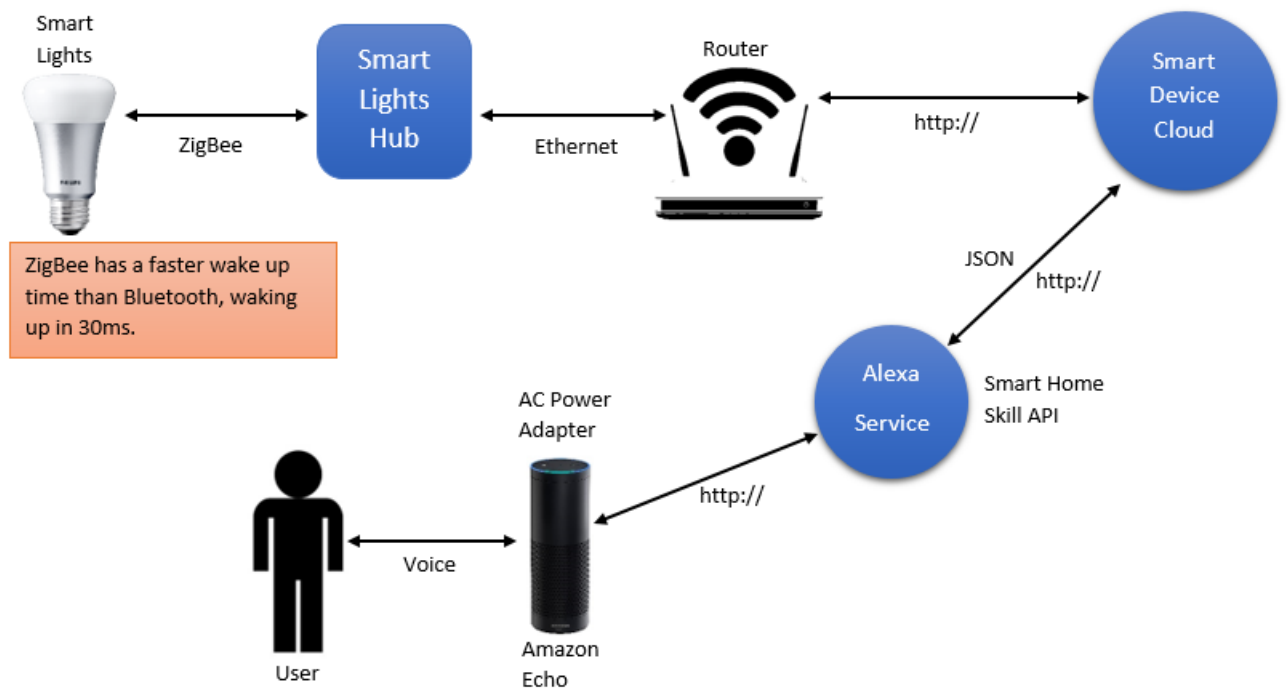
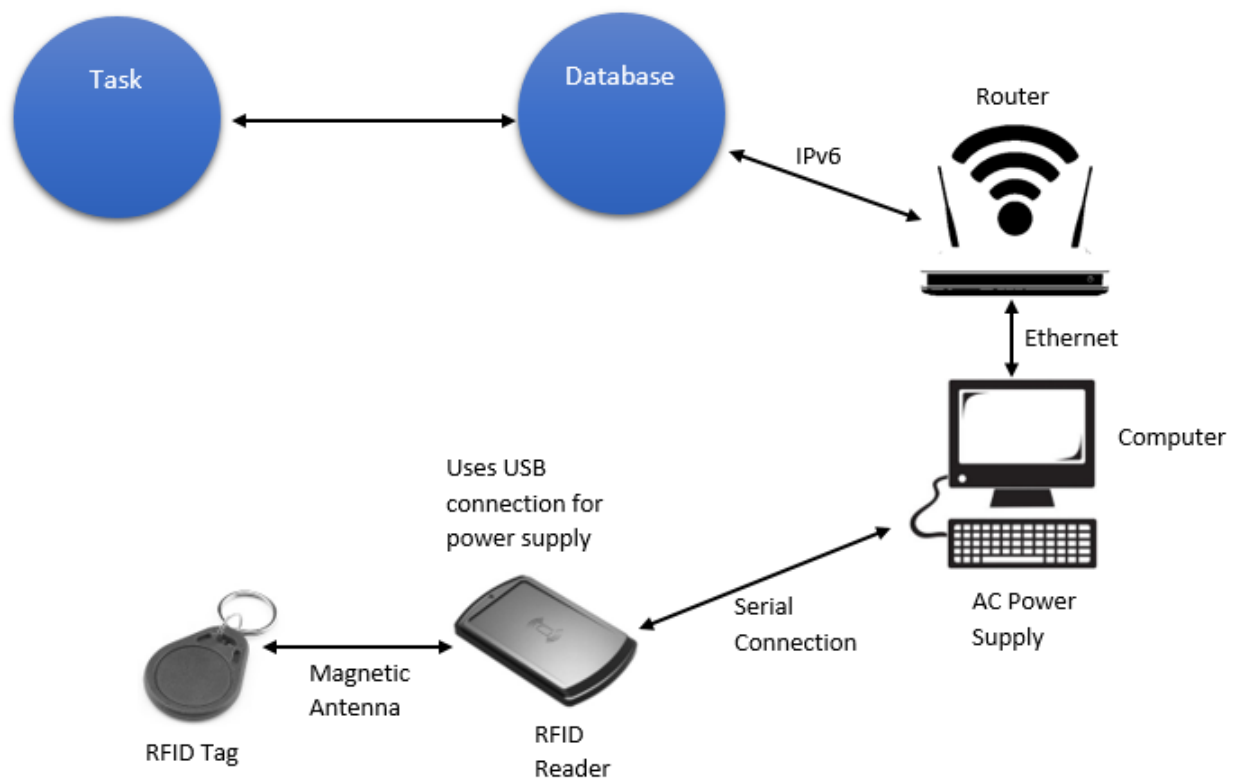


Figure 21: RFID Reader System Diagram



Task 4: Internet of Things Analysis

The title of my chosen topic is "Parking Availability Prediction for Sensor-Enabled Car Parks in Smart Cities". This work highlights that due to low costing, low power sensor technology, the Internet of Things has become more popular and is now being used in smart cities. More specifically the work looks at the use of sensor technology in car parks located in San Francisco, USA and Melbourne, Australia. Using data gathered from these sensor-enabled car parks, Zheng et al. (2015) produces a system to predict the occupancy rate of car parks. This is done using three sets of parameters that are later analysed on their effectiveness of prediction.

Bélissent et al. (2010) states that population in urban areas is projected to grow by 2.9 billion in 2050, this means nearly 70 percent of the worlds population will be living in cities and their surrounding areas. With increasing populations, it is vital that the smart city becomes more popular, this allows management and passing of information about public services more efficient. Zheng et al. (2015) mentions that one of the key felicitities to manage is a car park, as a study shows that 30 percent of congestion in a city is caused by cars that are searching for a place to park. Smart cities giving drivers information on the availability of car park spaces near their destination can help lower and control the congestion in a city.

The paper states that due to the increased development of sensor technology, various cities are now using IoT systems and devices in their cities. Zheng et al. (2015) then gives examples of IoT cities such as the City of Melbourne in Australia and San Francisco, USA who have made real-time parking information available to the public. This is where the proposed prediction system will be useful, the system will take use the sensor- enabled car parks and public data available together to make an efficient way for people to plan their trip and parking well ahead of time.

Three types of non-parametric algorithms were used in the papers research, these are the regression tree, neutral network and support vector regression. The input for these algorithms is the parking data from the cities of Melbourne and San Francisco, parameters were taken from this data and used to create three feature sets, to be input to the algorithms. The feature sets and algorithms were testing in all possible combinations to produce a parking occupancy rate for all of the scenarios, Zheng et al. (2015) says that the regression tree algorithm used with a the feature set that contained observation history along with the day of the week and the time of that day provided the best performance. Its also stated that the prediction system can be used in mobile applications or car navigation systems, to give drivers information during their trip and guide them to their nearest car park. This section relates back to work I have done and lectures that I have received during the IoT subject, for example there was a task called How Far Away? this task uses location to help when to decide when to active sensors. This can be used in a car park application, to find out how far away the users phone is from the car park and offer alternative car parks if a high number of other users are close to the same car park. GeoHash was talked about in the week 8 lecture, a use for this in a mobile application or car navigation system would be to pin car park locations onto the user map allowing them to see all close car parks at a quick glance, allowing them to keep focused on the

road ahead.

When the occupancy rate of a car park is to be determined, the output is between the ranges of 0 to 1 and the input for the algorithms are data sets. The input for data set one used the time and days of the week and gave an output for the occupancy rate at a given time for each day of the week. The second data set used the occupancy rate at a given time, the number of observations before the current one and how far ahead the prediction of occupancy was to be. Finally, feature set three combined the two previous two feature sets.

When gathering the data for the research San Francisco's 8200 parking sensors would be the source for this data, though not all data from the 8200 sensors is available due to some sensors being in off-street car parks that do not give real-time information. Data was collected by the team every 15 minutes during the dates 12/08/2013 and 22/09/2013, during this time 3948 records were collected. This data gave the team the number of spaces currently in use at each time and the number of operational spaces, to calculate the occupancy rate for San Francisco the number of slots in use was divided by the amount of slots operational. During research there were problems such as connection failures and sensors malfunctioning, IoT problems were highlighted in the week 9 lecture on critical issues. For example to this day there is no universal network that is used to manage all types of sensors across different networks, this can lead to connection issues like the research encountered. Some pieces of IoT equipment can be small and flimsy, increasing the chance of the kit being damaged or broken, harming data collection.

In the case of Melbourne the city had 7114 in-ground sensors in 23 areas if the city and Zheng et al. (2015) didn't gather data the same way they did for San Francisco. The team collected data that had already been recorded over the period between 01/10/2011 and 30/09/2012, giving them 12,208,178 records. The occupancy rate for this data set would have to be calculated a different way, as the records were arrival times and departure times. 1 was added to their formula for arrival times and 1 was deducted for departure times, after data for each street block was collected the number given would tell the team the number of spots currently occupied, the same calculation as used for San Francisco could then be used. After the data collected by both cities had been put through the data sets and algorithms, the research found that the regression tree algorithm, using data sets including the time, day of the week and history of occupancy rates was the best at predicting parking availability.

The research that was conducted has its advantages and disadvantages, the main advantage of the work is that it will help reduce the amount of congestion in cities when it is implemented into applications. This will become more and more useful over the years when the population of cities grow, as it will become a useful tool for more people. Though the research is very useful it also has some downsides, the first main one is that for the prediction algorithm to be used a lot of data has to be gathered first. This means that for other cities it may be a while before they are able to use this algorithm, especially if the city isn't utilising IoT systems yet. Another downside to the work is that although the prediction updates in real-time - depending on the day and the time to give the most accurate information available at the time, it is only a prediction and the system can be wrong especially when there are events on days that the system doesn't account for.

As cities around the world begin to use more IoT systems to increase their efficiency, a system can be created that updates in real-time and updates the amount of car park spaces in nearby car parks. There are a few ways this information can be displayed, the first would be to display the results on digital billboards that already exist in multiple cities, another would be to integrate LED displays onto road signs and show the car park spaces for each particular road. There is then also the potential for mobile phone applications to be linked to this system and for the LED displays to display different car park space values depending on how many people are being routed to the area by their mobile phone application.

In conclusion the research done in the paper is a good start to increasing the efficiency of smart cities. Though the research may become outdated in years to come if the population and amount of cars in the two cities increases by a substantial, this research can be used as a starting point for developers looking to create more car park systems in the future.

Bibliography

Jennifer Bélissent et al. Getting clever about smart cities: new opportunities require new business models. *Cambridge, Massachusetts, USA*, 2010.

Yanxu Zheng, Sutharshan Rajasegarar, and Christopher Leckie. Parking availability prediction for sensor-enabled car parks in smart cities. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.

Task 4 Report

ORIGINALITY REPORT

6%

SIMILARITY INDEX

3%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1	Yanxu Zheng, , Sutharshan Rajasegarar, and Christopher Leckie. "Parking availability prediction for sensor-enabled car parks in smart cities", 2015 IEEE Tenth International Conference on Intelligent Sensors Sensor Networks and Information Processing (ISSNIP), 2015. Publication	2%
2	Submitted to University of Melbourne Student Paper	1%
3	Submitted to Manchester Metropolitan University Student Paper	1%
4	www.irjet.net Internet Source	1%
5	groups.open.org.nz Internet Source	1%
6	zenodo.org Internet Source	1%