Embedded Systems - Project Report

Luke Fetin and Dylan Mitri

Introduction

In week 6, we were tasked with developing a project that applied the knowledge of embedded systems we had acquired so far over the semester, as well as strengthen our understanding of the core concepts and applications.

We could select from a list of basic premises of embedded systems projects or invent or our own. The methods of implementation and our prototypes functionality was up to the groups, however there were general marking guidelines to follow to ensure that our project remained within the bounds of an embedded system.

After careful consideration, we chose the "Wireless Car Park Reservation/Billing System" as our project. The main goal was to create a system in which users could remotely interact with a car park via a wireless connection, either Bluetooth or a Radio, communicating with the Arduino to reserve and pay for a space in a car park. We felt that this project has a simple concept yet had great potential to be an example of an embedded system.

After analysing the theoretical system provided to us, we decided to first implement a base functionality that we could expand off. We settled on creating a GUI interface in which users could connect to an Arduino (the intercom) via Bluetooth, where they could reserve a car space and the processing be done by the Arduino due. We settled on Bluetooth as we had functional Bluetooth shield modules at our disposal, as well as knowledge on how to use them.

We aimed to build off this as much possible by implementing additional functionality, such as peripherals like an LCD screen to display the information about various fields, sensors to read in information in the environment, a back-up/ off-the-grid power source and a database so users could check availability of car parks remotely and admins could check conditions of the Arduino by checking the status of the sensors remotely. We set up our base functionality such that we could easily implement any additional components to the existing prototype when we so desired. We had various resources available at our disposable courtesy of the Electrical Engineering department, as well as $50 to spend on any components we want or need from selected electronic retailers. We were also able to loan quintessential components such as an Arduino and breadboard from the department.

From the 4th of May, we were given roughly 5 Weeks lab time to work on our projects, as well as the Milestones we also had to meet. Given this we laid out a timeline for our project, hypothesising how much functionality we could realistically accomplish with our available time, allocating days where we would definitely work on the project and creating a reference/ guideline to show if we're staying on track.

Implementation

Prototype

As mentioned before, we aimed to get a base design for our prototype. Hardware wise, this consisted of an Arduino Due and the Bluetooth module. The Bluetooth shield was designed to be situated on top of an Arduino, which made connection very simple. We initially used this set up with a ‘off-the-shelf’ Bluetooth terminal, to prototype the Bluetooth communication with the user device. The implemented functionality was sending and receiving commands and data. After ensuring this basic functionality of the Bluetooth communication, we were then able to further implement additional components to the prototype. We removed the shield from on top of the board and situated it on a bread board, connecting to the Arduino via various wires. This was necessary to allow the LCD Display to connect to the SPI pins.

Hardware

From then, we implemented the other peripherals on a separate breadboard, to avoid interfering with the Bluetooth shield. Starting with the LCD screen, connecting it to the Arduino was simple as the instructions were provided on the Arduino website. We tested it was operational using some simple code.

The other peripherals were simply implemented. The point of the temperature sensor was to monitor the heat of the hardware, and prompt the software to react if the components begin to overheat. Analog data is read in from the sensor via the analog pins on the board, the sensor is also connected to the power supply and ground.

Similarly, the purpose of the accelerometer was to monitor the status of the hardware and the environment. The accelerometer was programmed to detect wild movements and thus, instability. In turn this information would also be sent to the software, which would prompt a course of action such as securing the information or suspending action until the system is stable. Like the temperature sensor, the data is read in via the analog pins on the board, and is also connected to power and ground.

A simple power supply was also connected straight into the board via a port on the Arduino. The power supply was a 9v DC battery pack. This gave us sufficient current and power to power our components, and ensure power stability, as most components were low consumption. Also related was another peripheral we added, a simple voltage divider circuit. This allowed us to measure the voltage read in from our power supply. Like our other peripherals, we could use the data read in to signal the software to do something. In our case, we monitored the power level of the board and would raise a warning if levels fell below a threshold.

Software

The core/ back-end of the user device application was built using Python which consisted on three parts: Bluetooth Communication, Database and GUI.

For the implementation of the Bluetooth communication we utilized PyBluez library. PyBluez library is a Bluetooth Python extension module to allow Python developers to use system Bluetooth resources, the PyBluez library allowed us to easily connect the user's device to the Arduino and send and receive data. Making use of our prototype the Bluetooth Python code was modelled from the output of the Bluetooth terminal in our original prototype in Stage 1.

The database itself is a PGSQL Server and is hosted on a USYD Server, for each car park a table is stored with an attribute for max capacity of car park, availability, cost per hour, battery level, temperature level and motion level. We used the Python library PG8000 to connect to the server, execute SQL queries and retrieve their results. Every time a user connects to an intercom (Arduino) it receives the appropriate car park data from the Arduino and updates the server

The front-end of the GUI was built using bootstrap, bootstrap is a free html, CSS and JavaScript framework for building responsive front-end websites. Flask, another Python library, is a web framework that allows easy dynamic rendering of web pages. Flask is the binder between the back-end and the front-end.

Problem Solving Approach

To gain a basis to have something to work on top of, we first implemented a prototype that demonstrated the basic functionality of the project, as specified in the previous section. From there we categorized our work into two independent groups, hardware and software, to allow us to develop the project concurrently.

The software group consisted of all things Web App GUI and Bluetooth and the hardware group consisted of everything else. We utilized our different skill sets to leverage further the efficiency of the implementation, Luke –Computer Science 3rd year and Electrical 2nd year, Dylan – Electrical 3rd.

Concerning the hardware components, the LCD screen was implemented first, as we could use the screen to output the data from the other components, thus testing the functionality of both at the same time. The implementation of the other components was straight forward, however interpreting the raw data read in required us to investigate what the values represented, and in turn how we could modify them to represent something meaningful.

Specifically, for the batter sensory, as we were not able to find an equivalent discharge curve or experiment and calculate it ourselves, we estimated by using a discharge curve for a single AA Alkaline battery (our battery pack consists of 6 AA Alkaline batteries). From there, knowing that the discharge curve initially has a steep decline, then a long plateau and again a steep decline, we categorised the voltage level in to 3 distinct levels low, medium and high.

The problem-solving approach to Implementing the software group was to follow a Block Method implementation: Implementing and then testing each component, systematically combining the components and testing at each state.
Another problem-solving approach for the software group was to utilize existing web frameworks to allow speedy development and sleek design, building a GUI from scratch could be very time consuming.

Discussion

The functionality we implemented into our final prototype covered all our ideas proposed at the design phase, however we had to factor in our choices to the timeline, as well as our practical knowledge and skills needed to implement our choices. There were various other ideas we had hypothesised implementing to further showcase our projects main purpose, as well as other features that will have aided in the core functionally as well.

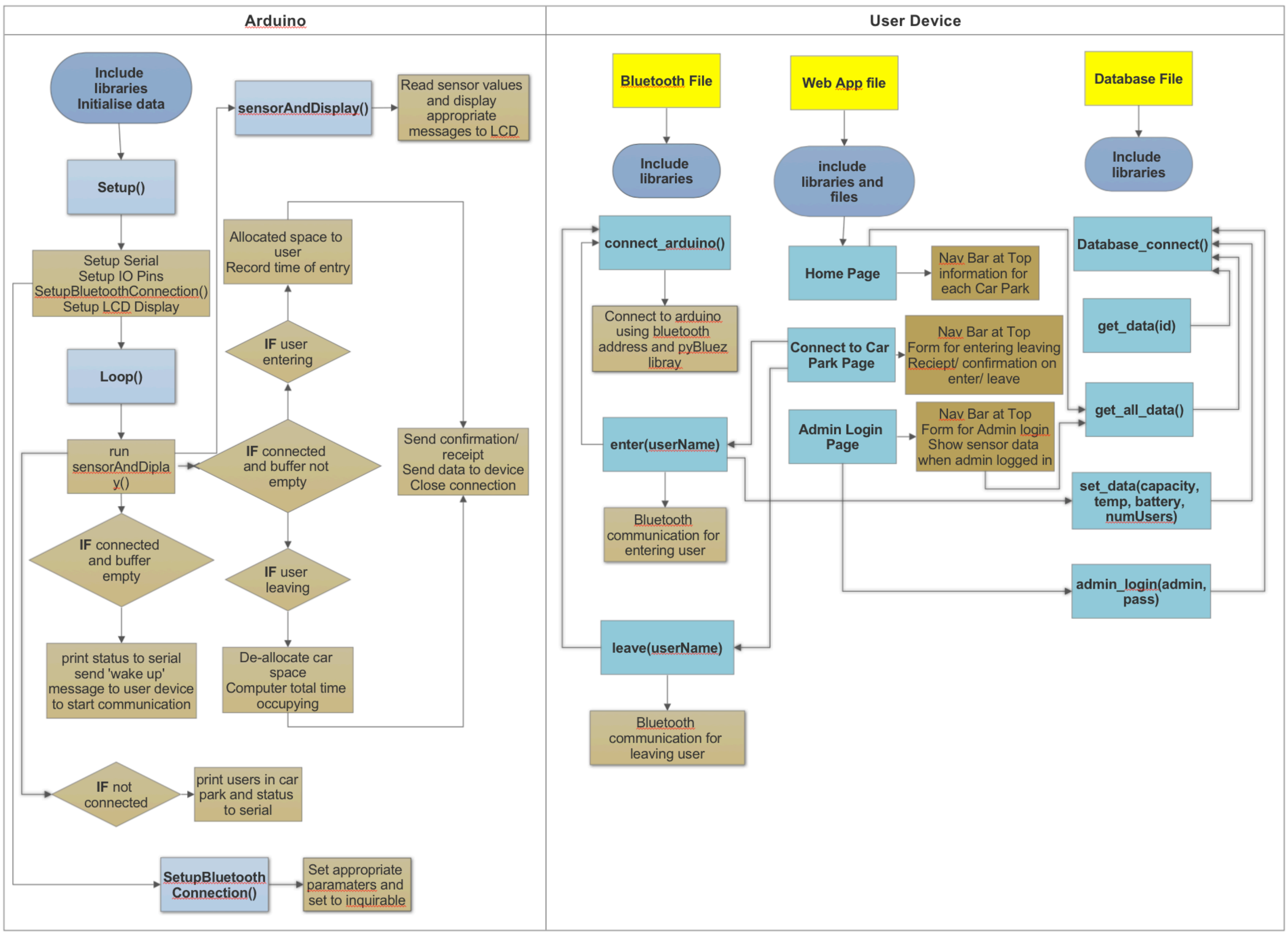
Such features hypothesised included model of a car park, with motion sensors, motorised doors and LEDs. We felt something like this would have helped demonstrate the connection between the user interfacing with our software, and the response the inputs into the system have on the actual carpark. We however did not peruse this as we instead wanted to focus on the software side of the system, creating a fully functional GUI. Another factor in deciding not to peruse this was the timeframe of the project, as well resources available, including our availability to work on such a design.

Conclusion

In conclusion, we believe that our final prototype fully showcases all the functionality we proposed initially. Having said that we also believe that there is much more room to improve our prototype, by example, adding in a prototype car park, however we believe that our key concepts of our embedded system have been visualised.

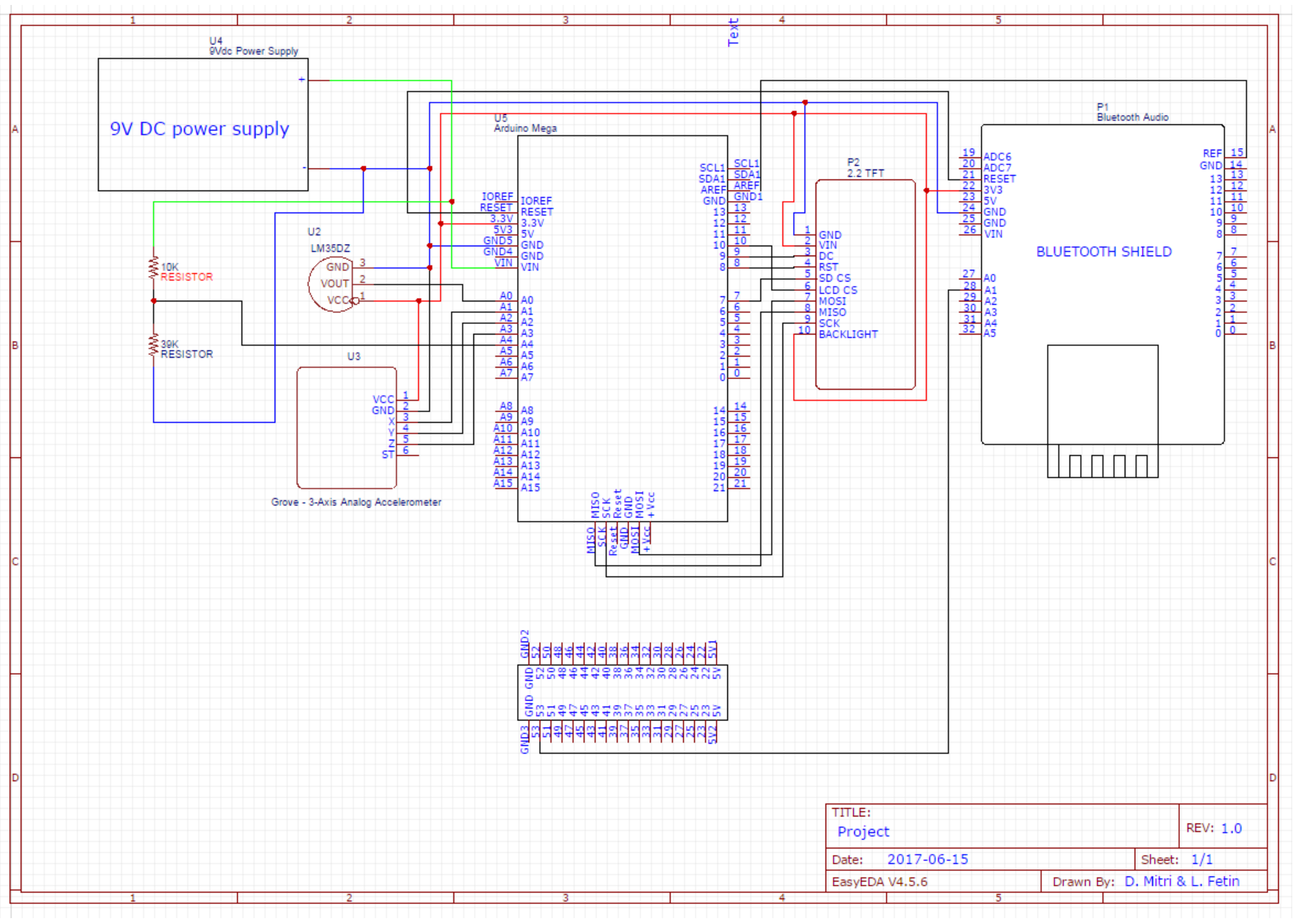
Our knowledge and understanding of embedded systems has also improved, as we now both understand more the concepts of an embedded system, its use, and how to implement and debug one.

Appendix 1: Software Flowchart


Arduino
User Device
Include libraries Initialise data
Setup()
Setup Serial
Setup IO Pins
SetupBluetoothConnection()
Setup LCD Display
Loop()
run sensorAndDiplay()
IF connected and buffer empty
print status to serial send 'wake up' message to user device to start communication
IF not connected
print users in car park and status to serial
SetupBluetooth Connection()
Set appropriate paramaters and set to inquirable
sensorAndDisplay()
Read sensor values and display appropriate messages to LCD
Allocated space to user Record time of entry
IF user entering
IF connected and buffer not empty
IF user leaving
De-allocate car space Computer total time occupying
Send confirmation/ receipt Send data to device Close connection
Bluetooth File
Include libraries
connect_arduino()
Connect to arduino using bluetooth address and pyBluez libray
enter(userName)
Bluetooth communication for entering user
leave(userName)
Bluetooth communication for leaving user
Web App file
include libraries and files
Home Page
Nav Bar at Top information for each Car Park
Connect to Car Park Page
Nav Bar at Top Form for entering leaving Reciept/ confirmation on enter/ leave
Admin Login Page
Nav Bar at Top Form for Admin login Show sensor data when admin logged in
Database File
Include libraries
Database_connect()
get_data(id)
get_all_data()
set_data(capacity, temp, battery, numUsers)
admin_login(admin, pass)

The diagram shows the control for the two devices. Dark blue circles represent the code in the global space, light blue squares represent functions, light tan squares represent corresponding code blocks to these functions, light tan diamonds show control logic for these corresponding functions and dark tan squares represent rendered web pages.
For the Arduino, the global space code is run first, then setup and then loop. Depending on what events occur after this will determine the procedure of the Arduino as specified in the flowchart.
For the User Device, Web App file is run to start the GUI interface, the code in the global space is run then the function to render the home page. The user is now able to navigate the GUI. The Web App file will use functions from the 2 other files Bluetooth and Database when needed as specified in the diagram.

Appendix 2: Hardware Schematic



Hardware Schematic Components:

Arduino due: Acting as the control unit for the Bluetooth and all the other peripherals, the board will process the information coming from the user, via the Bluetooth connection.

Bluetooth shield: The Bluetooth hardware, a small module that is connected to the Arduino. Generates a Bluetooth signal which users can connect to and access the car park system.

LCD Screen: A basic TFT LCD screen, used to display information about the carpark as well as the environment, and status of the hardware.

Integrated breadboard: A simple breadboard used to facilitate the hardware components and assist in the connection between all parts.

Power supply: A 9 volt DC power supply used to power the components independently.

Temperature sensor: Simple temperature sensor, values read in through analog pins.

Accelerometer: 3-axis accelerometer, values read in through analog pins also.

Resistors: Used to check the voltage of the power supply.

Hardware Schematic Description

Our schematic shows all the connections for all components in our prototype. All of our components are connected to the Arduino via a breadboard or directly connecting into the pins.

The board is powered by our 9V battery pack, which is connected directly to the boards power jack. This power can then be accessed via the 3.3v pin, and the Vin pin. The rails on the bread board were connected to the 3.3v pin as that was the required voltage for most components. The boards were also grounded via the ground pin on the Arduino.

The Bluetooth shield, temperature sensor, voltage divider, and accelerometer all feed in analog data directly into the analog pins on the board. All of our components are powered via the power supply, by connecting to the rails on the breadboard. Our LCD screen required direct connections to the Arduino’s SPI pins, which was the reason we could not situate our Bluetooth shield on top of the Arduino.

Appendix 3: Arduino Code

```
#include <ArduinoSTL.h> /* Author: Mike Matera, Andy Brown - C++ STL Library */
#include <TFT.h>  // Arduino LCD library
#include <SPI.h>

#define cs   10
#define dc   9
#define rst  8

#define blueToothSerial Serial2

// Initialise User Data
int numCarSpaces = 250;
int numUsers = 50;
std::vector<String> userOccupySpace;
int loopCounter = 0;
std::vector<unsigned long> enterTotalMinutes;
int counter = 0;

// Initialise LCD and Sensor Data
int batteryLevel = 0;
int motionLevel = 0;
int temperatureLevel = 0;

TFT TFTscreen = TFT(cs, dc, rst);
char battery_level_array[50];
char high_array[10];
char medium_array[10];
char low_array[10];
char caution_array[20];
char motion_array[50];
char temp_array[50];

String battery_level_string = "Power level:\n";
String high_string = "High\n";
String medium_string = "Medium\n";
String low_string = "Low\n";
String caution_string = "CAUTION\n";
String temp_string = "Board is overheating!\n";
String motion_string = "Board is unstable!\n";

int x_stable = 494;
int y_stable = 512;
int z_stable = 616;

int stable_9v = 528;

int prev_temp = 0;

int cycle = 0;



void setup()
{
      /*
       *  Setup for bluetooth module and user device communication
       */
      Serial.begin(9600);
      pinMode(53,INPUT); // Set digital pin 53 to input from A1 of bluetooth
shield, 1 indicates bluetooth shield connected and 0 not connected
      setupBlueToothConnection();

      /*
```

```
        *  Setup for Sensors and LCD
        */
       // Required at the beginning of every sketch that uses the GLCD:
       TFTscreen.begin();

       // clear the screen with a black background
       TFTscreen.background(0, 0, 0);

       battery_level_string.toCharArray(battery_level_array,50);
       high_string.toCharArray(high_array,10);
       medium_string.toCharArray(medium_array,10);
       low_string.toCharArray(low_array,10);
       caution_string.toCharArray(caution_array,10);
       motion_string.toCharArray(motion_array,50);
       temp_string.toCharArray(temp_array,50);
}

void loop()
{
   int recvInt = 0;
    while(1)
    {

        int isConnected = digitalRead(53);

         sensorAndDisplay(); // Display information on LCD

        // If device connected and buffer available
        if((isConnected != 0) && blueToothSerial.available())
        {

              String s = "";

              blueToothSerial.println("Entering or Leaving?"); // Ask user device
if entering or leaving

               // Read in Entering or Leaving request from bluetooth buffer
               // While loop enforces synchronisation between user device and
arduino, polling
               while(s != "Entering" && s != "Leaving"){


                  blueToothSerial.println("Entering or Leaving?");
                  s = blueToothSerial.readString();

                  isConnected = digitalRead(53);
                  if(isConnected == 0) break; // If connection drops, break

                 }

              // If user is entering
              if(s == "Entering"){

                 blueToothSerial.println("Enter User ID");
                 String iD = "";

                 // Read in User Id
                 while(iD == ""){


                  blueToothSerial.println("Enter User ID");
                  iD = blueToothSerial.readString();

                  isConnected = digitalRead(53);
                  if(isConnected == 0) break;
```

```
                }


                bool newID = true;

                //Check to see if user already in car park
                for(int i = 0; i < userOccupySpace.size(); i++){

                   if(userOccupySpace[i] == iD){
                     newID = false;
                   }

                }

                // If new user, add to vector data structure and record time of
entry
                if(newID){

                  // Compute time entered
                  unsigned long totalMinutes = millis() / 1000;
                  enterTotalMinutes.push_back(totalMinutes);
                  unsigned long currentMinute = totalMinutes % 60;
                  unsigned long currentHour = totalMinutes / 60;
                  userOccupySpace.push_back(iD);

                  // print to serial and user device confirmation of entry
                  Serial.println("User: " + iD + " entered at time, H: " +
currentHour + " M: " + currentMinute);
                   blueToothSerial.println("User: " + iD + " entered at time, H: "
+ currentHour + " M: " + currentMinute);
                   numUsers++;

                } else{

                   Serial.println("User: " + iD + " already in use.");
                   blueToothSerial.println("User: " + iD + " already in use.");

                }

                String recievedData = "";

                // Send Car Park data to user device
                while(recievedData != "true"){

                  // Format data string for user device
                   String s = "Sending Data:";
                   s += numUsers;
                   s += ",";
                   s += batteryLevel;
                   s += ",";
                   s += motionLevel;
                   s += ",";
                   s += temperatureLevel;

                   blueToothSerial.println(s);
                   recievedData = blueToothSerial.readString();
                   Serial.println(recievedData);

                   isConnected = digitalRead(53);
                   if(isConnected == 0) break;

                }

                // End and close connection
                blueToothSerial.end();
                delay(500);
                setupBlueToothConnection();
                delay(500);

             // If user is leaving
             } else if(s == "Leaving"){

                 blueToothSerial.println("Enter User ID");
                String iD2 = "";

                 // Read in user Id from bluetooth buffer
                while(iD2 == ""){


                 blueToothSerial.println("Enter User ID");
                 iD2 = blueToothSerial.readString();

                 isConnected = digitalRead(53);
                 if(isConnected == 0) break; // If connection drops, break
                }

               bool userHere = false;
               unsigned long leaveM = 0;
               unsigned long leaveH = 0;

                // Check if user is in car park
                for(int i = 0; i < userOccupySpace.size(); i++){


                    if(userOccupySpace[i] == iD2){

                      // calculate time user occupied car space if user in car
park
                      unsigned long totalMinutes = millis() / 1000;
                      unsigned long leaveMinutes = abs(totalMinutes -
enterTotalMinutes[i]);
                      leaveM = leaveMinutes % 60;
                      leaveH =  leaveMinutes/60;
                      userOccupySpace.erase(userOccupySpace.begin() + i);
                      enterTotalMinutes.erase(enterTotalMinutes.begin() + i);
                      userHere = true;
                      numUsers--;

                    }

                }

                if(userHere){

                   // print reciept to serial and send to user device
                   Serial.println("User: " + iD2 + " stayed in car park for, H: "
+ leaveH + " M: " + leaveM + " ");
                   blueToothSerial.println("User: " + iD2 + " stayed in car park
for, H: " + leaveH + " M: " + leaveM + " ");


                }else{

                   // else print to serial and send to user device user is not in
car park
                   Serial.println("User: " + iD2 + " not in car park");
                   blueToothSerial.println("User: " + iD2 + " not in car park ");

                }

             String recievedData = "";
```

```
                // Send Car Park Data to user device
                while(recievedData != "true"){

                   String s = "Sending Data:";
                   s += numUsers;
                   s += ",";
                   s += batteryLevel;
                   s += ",";
                   s += motionLevel;
                   s += ",";
                   s += temperatureLevel;

                   blueToothSerial.println(s);
                   recievedData = blueToothSerial.readString();
                   Serial.println(recievedData);
                   isConnected = digitalRead(53);

                   if(isConnected == 0) break;

                }

             // End and close connection
              blueToothSerial.end();
              delay(500);
              setupBlueToothConnection();
              delay(500);

              }

        }

        // If device connect but bluetooth buffer empty
        else if((isConnected != 0) && !blueToothSerial.available()){

            // Counter to intermitently print message to serial
            if(counter > 5){

              Serial.println("connected: empty buffer");
              counter = 0;

            }

            // Send data to user device buffer so that communication can start
            blueToothSerial.println("Wake up");
            counter++;

        }

        // If device not connected
        else
        {

          // If atleast one user is in car park
          if(userOccupySpace.size() > 0 && counter > 4){

            Serial.println("Disconnected");

            // Print to serial details of users logged in
            Serial.println("Number of user of users: " +
String(userOccupySpace.size()));

            for(int i =0 ;i < userOccupySpace.size(); i++){

               Serial.println("User: " + userOccupySpace[i]);

            }
            counter = 0;
          }

          // If no users in car park
          else
          {

            if(counter > 4){

              Serial.println("Disconnected");
              counter = 0;

            }

            counter++;

          }
       }
    }
}

/*
*  Setup for Sensors and LCD
*/
void sensorAndDisplay() {
  // Read in the analog values
  int reading_temp = analogRead(A0);
  int reading_voltage = analogRead(A1);

  int x_reading = analogRead(A2);
  int y_reading = analogRead(A3);
  int z_reading = analogRead(A4);

  int temperature = reading_temp / 5;
  temperatureLevel = temperature;
  int voltage = reading_voltage;

  // Check if board stable, if unstable print warning to LCD Display
  if( (x_reading > x_stable + 20) || (x_reading < x_stable - 20) ||
  (y_reading > y_stable + 20) || (y_reading < y_stable - 20) ||
  (y_reading > y_stable + 20) || (y_reading < y_stable - 20) ) {

    motionLevel = 1;

    // LCD Display controls
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(2);
    TFTscreen.text(caution_array, 0, 0);
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(1.5);
    TFTscreen.text(motion_array, 0, 20);
    delay(5000);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(caution_array, 0, 0);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(motion_array, 0, 20);

    TFTscreen.background(0, 0, 0);

  }
  else {
    motionLevel = 0;
  }

  // Check temperature around board, if too hot display warning on LCD Display
```

```
if(prev_temp != 0 && cycle > 3) {
  if(reading_temp > prev_temp + 3) {

    TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(2);
    TFTscreen.text(caution_array, 0, 0);
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(1.5);
    TFTscreen.text(temp_array, 0, 20);
    delay(3000);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(caution_array, 0, 0);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(temp_array, 0, 20);

    TFTscreen.background(0, 0, 0);
  }
}

// If no warning is displayed, the level of the battery will be displayed

// Medium lebel
if(voltage <= (stable_9v / 2)) {
  batteryLevel = 0;
  TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(2);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.text(medium_array, 0, 20);
    delay(2500);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(medium_array, 0, 20);
}
// Low Level
else if(voltage <= (stable_9v / 4)) {
  batteryLevel = -1;
  TFTscreen.stroke(255, 255, 255);
    TFTscreen.setTextSize(2);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.text(low_array, 0, 20);
    delay(1000);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(low_array, 0, 20);
}
// High Level
else {
  TFTscreen.stroke(255, 255, 255);
    batteryLevel = 1;
    TFTscreen.setTextSize(2);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(255, 255, 255);
    TFTscreen.text(high_array, 0, 20);
    delay(1000);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(battery_level_array, 0, 0);
    TFTscreen.stroke(0, 0, 0);
    TFTscreen.text(high_array, 0, 20);
}

prev_temp = reading_temp;
cycle++;
```

```
}

/*
 *  Setup for bluetooth module
 */
void setupBlueToothConnection()
{
    blueToothSerial.begin(38400);                            // Set BluetoothBee
BaudRate to default baud rate 38400
    blueToothSerial.print("\r\n+STWMOD=0\r\n");              // set the bluetooth
work in slave mode
    blueToothSerial.print("\r\n+STNA=CarPark\r\n");          // set the bluetooth
name as "Car Park"
    blueToothSerial.print("\r\n+STOAUT=1\r\n");              // Permit Paired device
to connect me
    blueToothSerial.print("\r\n+STAUTO=0\r\n");              // Auto-connection
should be forbidden here
    delay(2000);                                             // This delay is
required.

    while(blueToothSerial.available()) {
      Serial.write(blueToothSerial.read());
    }

    blueToothSerial.print("\r\n+INQ=1\r\n");                 // make the bluetooth
inquirable
    Serial.println("The slave bluetooth is inquirable!");
    blueToothSerial.flush();
    delay(2000);                                             // This delay is
required.

    // Ensure bluetooth serial is flushed
    while(blueToothSerial.available()) {

      Serial.write(blueToothSerial.read());

    }

}
```

Appendix 4: User Device - Bluetooth Python Code

```
import serial
import time
import database

#####################################################
##  Connect to Arduino
#####################################################

def connect_arduino():

   print("Start")
   port="/dev/tty.CarPark11-DevB"  #Connect to Arduino using bluetooth address
   bluetooth=serial.Serial(port, 9600)  #Start communications with the bluetooth
unit
   print("Connected")
   bluetooth.flushInput()

   return bluetooth

#####################################################
##  Request to Enter
#####################################################

def enter(userName):

    # Connect to Arduino
   bluetooth = connect_arduino()

   input_data = ''

   #  Wait to Arduino request to recieve Entering or Leaving request
   while ('Entering or Leaving' not in input_data):
      input_data = bluetooth.readline().decode()[:20]

   # Send entering request
   bluetooth.write(b'Entering')
   time.sleep(2) # Delays used for synchronisation
   bluetooth.write(b'Entering')

   time.sleep(0.1)

   #  Wait to Arduino request to recieve User ID
   while (input_data != 'Enter User ID'):
      input_data = bluetooth.readline().decode()[:13]

   # Send User ID
   bluetooth.write(b""+str.encode(userName))

   time.sleep(0.1)

   # Wait to recieve confirmation from Arduino
   while ('User: ' not in input_data):
      input_data = bluetooth.readline().decode()

   message = input_data

   print(message)

   # Wait to Arduino request to recieve Data
   while ('Data' not in input_data):
      input_data = bluetooth.readline().decode()
   # Send data recieved confirmation to Arduino
   bluetooth.write(b"true")

   # Get data from message
   data = input_data.split(':')[1]
   data = data.split(',')
   data[3] = data[3].split('\r')[0]

   # Update databse
   database.set_capacity(int(data[0]),1)
   database.set_battery(int(data[1]), 1)
   database.set_motion(int(data[2]), 1)
   database.set_temp(int(data[3]), 1)


   bluetooth.close()  # Close bluetooth conenction
   print("Done")

   return message

#####################################################
##  Request to Leave
#####################################################

def leave(userName):

   # Connect to Arduino
   bluetooth = connect_arduino()

   #  Wait to Arduino request to recieve Entering or Leaving request
   input_data = ''
   while (input_data != 'Entering or Leaving?'):
      input_data = bluetooth.readline().decode()[:20]

   # Send entering request
   bluetooth.write(b'Leaving')
   time.sleep(2)
   bluetooth.write(b'Leaving')

   time.sleep(0.1)

   #  Wait to Arduino request to recieve User ID
   while (input_data != 'Enter User ID'):
      input_data = bluetooth.readline().decode()[:13]


   # Send User ID
   bluetooth.write(b""+str.encode(userName))
   time.sleep(1)
   bluetooth.write(b"" + str.encode(userName))

   time.sleep(0.1)

   # Wait to recieve confirmation from Arduino
   while ('User: ' not in input_data):
      input_data = bluetooth.readline().decode()

   message = input_data

   print(message)

   time.sleep(2)

   # Wait to Arduino request to recieve Data
   while ('Data' not in input_data):
      input_data = bluetooth.readline().decode()
```

```
# Send data recieved confirmation to Arduino
bluetooth.write(b"true")

# Get data from message
data = input_data.split(':')[1]
data = data.split(',')
data[3] = data[3].split('\r')[0]

# Update databse
database.set_capacity(int(data[0]), 1)
database.set_battery(int(data[1]), 1)
database.set_motion(int(data[2]), 1)
database.set_temp(int(data[3]), 1)

bluetooth.close()  # Close bluetooth connection
print("Done")

return message
```

Appendix 5: User Device – Database Python Code

```
#!/usr/bin/env python3

from modules import pg8000
import configparser

#####################################################
##  Database Connect
#####################################################

'''
Connects to the database using the connection string
'''
def database_connect():

    # Read the config file
    config = configparser.ConfigParser()
    config.read('config.ini')

    # Create a connection to the database
    connection = None

    try:

        # Parses the config file and connects using the connect string
        connection = pg8000.connect(database=config['DATABASE']['user'],
                                    user=config['DATABASE']['user'],
                                    password=config['DATABASE']['password'],
                                    host=config['DATABASE']['host'])

    except pg8000.OperationalError as e:

        print("""Error, you haven't updated your config.ini or you have a bad
        connection, please try again. (Update your files first, then check
        internet connection)
        """)
        print(e)

    # return the connection to use
    return connection

#####################################################
##  Get Data for Specific Car Park
#####################################################

def get_data(id):

    # Ask for the database connection, and get the cursor set up
    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try execute SQL query to retrieve data for specified Car Park
        sql = """SELECT *
                 FROM carpark1
                 WHERE id="""+str(id)
        cur.execute(sql)
        r = cur.fetchone()              # Fetch the first row
        cur.close()                         # Close the cursor
        conn.close()                        # Close the connection to the db

        return r

    except:

        # If there were any errors, return a NULL row printing an error to the 
debug
        print("Error Invalid Login")

    cur.close()                         # Close the cursor
    conn.close()                        # Close the connection to the db

    return None

#####################################################
##  Get Data for all Car Parks
#####################################################

def get_all_data():

    # Ask for the database connection, and get the cursor set up
    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try execute SQL query to retrieve data of all car parks

        sql = """SELECT *
                 FROM carpark1
                 ORDER BY id"""

        cur.execute(sql)
        r = cur.fetchall()                  # Fetch all rows
        cur.close()                         # Close the cursor
        conn.close()                        # Close the connection to the db

        # Format data for GUI
        data = [{
            'id': row[0],
            'capacity': str(row[1]).replace('-', '/'),  # Format for start_date
            'max': row[2],
            'batteryLevel': row[3],
            'motionLevel': row[4],
            'temperature': row[5],
            'price': row[6]
        } for row in r]

        return data

    except:

        # If there were any errors, return a NULL row printing an error to the 
debug
        print("Error Invalid Login")

    cur.close()                         # Close the cursor
    conn.close()                        # Close the connection to the db

    return None
```

```
#####################################################
##  Set Data
#####################################################

def set_capacity(newCapacity,id):

    # Ask for the database connection, and get the cursor set up

    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try execute SQL query to update Car Park Capacity

        sql = """UPDATE carpark1
                    SET capacity = %d
                    WHERE id = %d;""" % (newCapacity,id)

        cur.execute(sql)
        conn.commit()                   # Commit update on data
        cur.close()                     # Close the cursor
        conn.close()                    # Close the connection to the db

        return True

    except:

        # If there were any errors, return a NULL row printing an error to the
debug
        print("Error Invalid Login")

    cur.close()                     # Close the cursor
    conn.close()                    # Close the connection to the db

    return None

def set_battery(newValue,id):

    # Ask for the database connection, and get the cursor set up
    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try executing the SQL to update Battery Level
        sql = """UPDATE carpark1
                    SET batterylevel = %d
                    WHERE id = %d;""" % (newValue,id)

        cur.execute(sql)
        conn.commit()                   # Commit update to data
        cur.close()                     # Close the cursor
        conn.close()                    # Close the connection to the db

        return True
    except:

        # If there were any errors, return a NULL row printing an error to the
debug
        print("Error Invalid Login")

    cur.close()                        # Close the cursor
    conn.close()                       # Close the connection to the db

    return None

def set_motion(newValue,id):

    # Ask for the database connection, and get the cursor set up

    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try executing the SQL to update Motion Level

        sql = """UPDATE carpark1
                    SET motionlevel = %d
                    WHERE id = %d;""" % (newValue,id)

        cur.execute(sql)
        conn.commit()                      # Commit updated data
        cur.close()                        # Close the cursor
        conn.close()                       # Close the connection to the db

        return True

    except:

        # If there were any errors, return a NULL row printing an error to the
debug
        print("Error Invalid Login")

    cur.close()                        # Close the cursor
    conn.close()                       # Close the connection to the db

    return None

def set_temp(newValue,id):

    # Ask for the database connection, and get the cursor set up
    conn = database_connect()

    if(conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try executing the SQL to update Temperature Level
        sql = """UPDATE carpark1
                    SET temperature = %d
                    WHERE id = %d;""" % (newValue,id)

        cur.execute(sql)
        conn.commit()                   # Commit updated data
        cur.close()                     # Close the cursor
        conn.close()                    # Close the connection to the db

        return True

    except:

        # If there were any errors, return a NULL row printing an error to the
debug
        print("Error Invalid Login")

    cur.close()                     # Close the cursor
    conn.close()                    # Close the connection to the db

    return None

    #####################################################
    ##  Get Data
    #####################################################

def admin_login(user,password):

    # Ask for the database connection, and get the cursor set up

    conn = database_connect()

    if (conn is None):
        return None

    cur = conn.cursor()

    try:

        # Try executing the SQL to check for Admin User
        sql = """SELECT *
                 FROM admins
                 WHERE username='%s' AND password ='%s' """ % (user,password)

        cur.execute(sql)
        r = cur.fetchone()  # Fetch the first row
        cur.close()  # Close the cursor
        conn.close()  # Close the connection to the db

        return r

    except:

        # If there were any errors, return a NULL row printing an error to the
debug
        print("Error Invalid Login")

    cur.close()  # Close the cursor
    conn.close()  # Close the connection to the db

    return None
```

Appendix 6: User Device – Database Configuration File

```
##############################
##     Database Details     ##
##--------------------------##
##############################

[DATABASE]
host = soit-db-pro-2.ucc.usyd.edu.au
user = y17i2x20_lfet7715
password = 450299812
```

Appendix 7: User Device – GUI Python Code

```
# import modules and files

from modules import flask
from flask import * # Flask library used to connected python with web app
import bluetooth
import database

# Initilialise flask app
app = Flask(__name__)

carParkData = {()}

# First page shown when user accesses website
@app.route('/')
def index():
    global carParkData
    carParkData = database.get_all_data() # Retrieve data from database

    return render_template('first.html', carParkData = carParkData) # Render Web
App template

# Connect to Car Park Page
@app.route('/connect', methods=['GET', 'POST'])
def connect():

    # If user just viewing page
    if(request.method == 'GET'):
        return render_template('connect.html')

    # If user submits form to connect to car park form

    if (request.form['carpark'] != '1'):
        return render_template('reciept.html', message='Can not connect to Car
Park', leaving=False)


    # Call function to communicate request to Arduino
    if(request.form['entering'] == 'true'):
        message = bluetooth.enter(request.form['userid'])
    else:

        message = bluetooth.leave(request.form['userid'])

        payment = ''

        # Calculate cost using price for car park and time spent (encoded in
```

```python
message)
        if 'not in car park' not in message:
            start = 39 + len(request.form['userid'])
            minute = int(message[start:start + 2])
            start = 34 + len(request.form['userid'])
            hour = int(message[start:start + 2])
            price = int(carParkData[0]['price'])

            leaving = True

            cost = hour * price

            if( minute > 0 ):
                cost += price

            payment = 'Amount Due: $' + str(cost)


        return render_template('reciept.html', message = message, payment =
payment, leaving = True)

    return render_template('reciept.html', message=message, leaving = False)

# Admin page
@app.route('/admin', methods=['GET', 'POST'])
def admin():

    # Retrieve most recent data from database
    global carParkData
    carParkData = database.get_all_data()

    # If user is on login page
    if(request.method == 'GET'):
        return render_template('admin_login.html')

    # If user has submitted login form

    # Check database for admin user
    login_request =
database.admin_login(request.form['username'],request.form['password'])

    if login_request is None:
        message = 'Incorrect username or password'

        return render_template('reciept.html', message=message, leaving=False)

    # Format data for display
    for row in carParkData:
        if(row['batteryLevel'] > 0):
            row['batteryLevel'] = 'High'
        elif (int(row['batteryLevel']) == 0):
            row['batteryLevel'] = 'Medium'
        elif (int(row['batteryLevel']) < 0):
            row['batteryLevel'] = 'Low'
        if (row['motionLevel'] == 0):
            row['motionLevel'] = 'Stable'
        else:
            row['motionLevel'] = 'UNSTABLE'

    return render_template('admin_view.html',  carParkData = carParkData)


# Start App

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Appendix 8: User Device – Home Web Page

```
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Sydney Car Parks</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="{{ url_for('static', filename='scripts/jquery.js') }}"></script>
    </head>
    <body>
      <nav class="navbar navbar-default">
        <div class="container-fluid">
          <!—Nav bar and toggle get grouped for better mobile display -->
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">CPS</a>
          </div>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
              <li class="active"><a href="#">Check Car Park Availability <span
class="sr-only">(current)</span></a></li>
              <li><a href="{{ url_for('connect')}}">Connect to Car Park</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <li><a href="{{ url_for('admin')}}">Admin Login</a></li>
            </ul>
          </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
      </nav>
      <div class="row">
        <div class="col-xs-6 col-md-3 col-lg-offset-1">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 1 </b></h1></a>
            <p> Address: 31 Abercrombie St Chippendale </p>
            <p> Total Capcity: {{carParkData[0]['max']}} </p>
            <p> Availability: {{carParkData[0]['capacity']}} </p>
            <p> Price per hour: {{carParkData[0]['price']}}$ </p>
          </div>
        </div>
        <div class="col-xs-6 col-md-3">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 2 </b></h1></a>
            <p> Address: 21 Nowhere Av Glebe </p>
```

```html
          <p> Total Capcity: {{carParkData[1]['max']}} </p>
          <p> Availability: {{carParkData[1]['capacity']}} </p>
          <p> Price per hour: {{carParkData[1]['price']}}$ </p>
        </div>
      </div>
      <div class="col-xs-6 col-md-3">
        <div  class="thumbnail">
          <a href="#"> <h1><b> Car Park 3 </b></h1></a>
          <p> Address: 31 Somewhere Rd Ultimo </p>
          <p> Total Capcity: {{carParkData[2]['max']}} </p>
          <p> Availability: {{carParkData[2]['capacity']}} </p>
          <p> Price per hour: {{carParkData[2]['price']}}$ </p>
        </div>
      </div>
      <div class="col-xs-6 col-md-3 col-lg-offset-4">
        <div  class="thumbnail">
          <a href="#"> <h1><b> Car Park 4 </b></h1></a>
          <p> Address: 31 Righthere Cl Darlington </p>
          <p> Total Capcity: {{carParkData[3]['max']}} </p>
          <p> Availability: {{carParkData[3]['capacity']}} </p>
          <p> Price per hour: {{carParkData[3]['price']}}$ </p>
        </div>
      </div>
    </div>
  </body>
</html>
```

Appendix 9: User Device – Connect to Car Park Web Page

```
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Sydney Car Parks</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    </head>
    <body>
      <nav class="navbar navbar-default">
        <div class="container-fluid">
          <!-- Navbar and toggle get grouped for better mobile display -->
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">CPS</a>
          </div>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
              <li><a href="{{ url_for('index')}}">Check Car Park Availability
</a></li>
              <li class="active"><a href="#">Connect to Car Park<span class="sr-
only">(current)</span></a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <li><a href="{{ url_for('admin')}}">Admin Login</a></li>
            </ul>
          </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
      </nav>
      <div class="col-xs-4 col-xs-offset-4">
        <form method = "POST" class="form-horizontal"
action="{{url_for('connect')}}" >
          <div class="form-group">
            <label for="action" class="control-label">  Entering or Leaveing ?
</label>
            <select name = "entering" class="form-control">
              <option value = "true">Entering</option>
              <option value = "false" >Leaving</option>
            </select>
          </div>
          <div class="form-group">
            <label for="carpark" class="control-label"> Select Car Park</label>
            <div class="col-sm-10">
              <select name ="carpark" multiple class="form-control">
                <option value = "1">1</option>
                <option value = "2">2</option>
                <option value = "3">3</option>
                <option value = "4">4</option>
              </select>
            </div>
          </div>
          <div class="form-group">
            <label for="userid" class="control-label">User Id</label>
            <input type="text" name = "userid" class="form-control" 
placeholder="User Id">
          </div>
          <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
              <button type="submit" class="btn btn-default">Connect</button>
            </div>
          </div>
        </form>
      </div>
    </body>
```

Appendix 10: User Device – Admin Login Web Page

```
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Sydney Car Parks</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="{{ url_for('static', filename='scripts/jquery.js') }}"></script>
    </head>
    <body>
      <nav class="navbar navbar-default">
        <div class="container-fluid">
          <!-- Navbar and toggle get grouped for better mobile display -->
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">CPS</a>
          </div>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
              <li><a href="{{ url_for('index')}}">Check Car Park
Availability</a></li>
              <li><a href="{{ url_for('connect')}}">Connect to Car Park</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <li class ='active'><a href="{{ url_for('admin')}}">Admin
Login</a></li>
            </ul>
          </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
      </nav>
      <br></br><br></br><br></br>
      <form method ='POST' class="form-horizontal col-xs-offset-4 col-xs-4 col-lg-
offset-5 col-lg-3" action="{{url_for('admin')}}">
        <div class="form-group">
          <label for="username">Username</label><p></p>
          <input name="username" type="text" class="form-control"
id="exampleInputName2" placeholder="Enter Admin UserName">
        </div>
        <div class="form-group">
          <label for="Password">Password</label><p></p>
          <input name ="password" type="email" class="form-control"
id="exampleInputEmail2" placeholder="Password">
        </div>
      <button type="submit" class="btn btn-default">Login</button>
    </form>
  </body>
</html>
```

Appendix 11: User Device – Admin View Page

```html
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Sydney Car Parks</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="{{ url_for('static', filename='scripts/jquery.js') }}"></script>
    </head>
    <body>
      <nav class="navbar navbar-default">
        <div class="container-fluid">
          <!-- Navbar and toggle get grouped for better mobile display -->
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">CPS</a>
          </div>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
              <li class="active"><a href="#">Check Car Park Availability <span
class="sr-only">(current)</span></a></li>
              <li><a href="{{ url_for('connect')}}">Connect to Car Park</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <li><a href="{{ url_for('admin')}}">Admin Login</a></li>
            </ul>
          </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
      </nav>
      <div class="row">
        <div class="col-xs-6 col-md-3 col-lg-offset-1">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 1 </b></h1></a>
            <p> Address: 31 Abercrombie St Chippendale </p>
            <p> Total Capcity: {{carParkData[0]['max']}} </p>
            <p> Availability: {{carParkData[0]['capacity']}} </p>
            <p> Price per hour: {{carParkData[0]['price']}}$ </p>
            <p> Battery Level: {{carParkData[0]['batteryLevel']}} </p>
            <p> Motion Level: {{carParkData[0]['motionLevel']}} </p>
            <p> Temperature: {{carParkData[0]['temperature']}} </p>
          </div>
        </div>
        <div class="col-xs-6 col-md-3">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 2 </b></h1></a>
            <p> Address: 21 Nowhere Av Glebe </p>
            <p> Total Capcity: {{carParkData[1]['max']}} </p>
            <p> Availability: {{carParkData[1]['capacity']}} </p>
            <p> Price per hour: {{carParkData[1]['price']}}$ </p>
            <p> Battery Level: {{carParkData[1]['batteryLevel']}} </p>
            <p> Motion Level: {{carParkData[1]['motionLevel']}} </p>
            <p> Temperature: {{carParkData[1]['temperature']}} </p>
          </div>
        </div>
        <div class="col-xs-6 col-md-3">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 3 </b></h1></a>
            <p> Address: 31 Somewhere Rd Ultimo </p>
            <p> Total Capcity: {{carParkData[2]['max']}} </p>
            <p> Availability: {{carParkData[2]['capacity']}} </p>
            <p> Price per hour: {{carParkData[2]['price']}}$ </p>
            <p> Battery Level: {{carParkData[2]['batteryLevel']}} </p>
            <p> Motion Level: {{carParkData[2]['motionLevel']}} </p>
            <p> Temperature: {{carParkData[2]['temperature']}} </p>
          </div>
        </div>
        <div class="col-xs-6 col-md-3 col-lg-offset-4">
          <div  class="thumbnail">
            <a href="#"> <h1><b> Car Park 4 </b></h1></a>
            <p> Address: 31 Righthere Cl Darlington </p>
            <p> Total Capcity: {{carParkData[3]['max']}} </p>
            <p> Availability: {{carParkData[3]['capacity']}} </p>
            <p> Price per hour: {{carParkData[3]['price']}}$ </p>
            <p> Battery Level: {{carParkData[3]['batteryLevel']}} </p>
            <p> Motion Level: {{carParkData[3]['motionLevel']}} </p>
            <p> Temperature: {{carParkData[3]['temperature']}} </p>
          </div>
        </div>
      </div>
    </body>
</html>
```

Appendix 11: User Device – Receipt / Confirmation Web Page

```html
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Sydney Car Parks</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <!-- jQuery library -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="{{ url_for('static', filename='scripts/jquery.js') }}"></script>
    </head>
    <body>
      <nav class="navbar navbar-default">
        <div class="container-fluid">
          <!—Navbar and toggle get grouped for better mobile display -->
          <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">CPS</a>
          </div>
          <!-- Collect the nav links, forms, and other content for toggling -->
          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
              <li><a href="{{ url_for('index')}}">Check Car Park
Availability</a></li>
              <li><a href="{{ url_for('connect')}}">Connect to Car Park</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <li><a href="{{ url_for('admin')}}">Admin Login</a></li>
            </ul>
          </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
      </nav>
      <div class="well well-lg col-lg-6 col-lg-offset-3">
        <br></br>
            <h3><center>{{message}}</center><h3>
      <br></br>
        <h3><center>{{payment}}</center><h3>
    </div>
  </body>
</html>
```

Appendix 12: Software Library Citation

Our Source Code and Reusable Project Library

Authors: Luke Fetin and Dylan Mitri
Home Page: https://github.com/lfet/car-park-reservation-prototype
Documentation: https://github.com/lfet/car-park-reservation-prototype/blob/master/README.md
Licence: GNU and MitriFetin Public Licence

Arduino C++ STL Library

Authors: Mike Matera **and** Andy Brown'
Home Page: http://andybrown.me.uk/2011/01/15/the-standard-template-library-stl-for-avr-with-c-streams/
Download URL: https://github.com/mike-matera/ArduinoSTL
Licence: GNU

PyBluez Python Bluetooth Library

Author: Piotr Karulis
Maintainer: Piotr Karulis
Home Page: http://karulis.github.io/pybluez/
Download URL: https://github.com/karulis/pybluez
License: GPL

Flask Python Web Framework

Authors: Armin Ronacher and contributors.
Home Page: http://flask.pocoo.org/
Download URL: https://github.com/pallets/flask
License: https://github.com/pallets/flask/blob/master/LICENSE

PG800 Python Library

Author: Mathieu Fenniak
Home Page: https://github.com/mfenniak/pg8000
License: BSD

Bootstrap Web Framework

Authors: Mark Otto and Jacob Thornton
Home Page: https://github.com/twbs/bootstrap
Download URL: https://github.com/twbs/bootstrap
Licence: MIT Licence