FIT5140 Advanced Mobile Systems
Assignment 3b - IoT Project Documentation

Due Date: 4/11/2018
Tutor: Josh Olsen
Studio time: Tue, 2:00pm - 4:00pm
Student 1: Hanfang Zhao - 28127641
Student 2: Yilei Pan - 28583280
Project Name: Parking System

1. Overall Architecture

The system can be divided into three modules, and in 2 aspects.

	Server	Database	Client
Hardware	1. PIR Sensor Module 2. Raspberry Pi 8MP Camera V2 3. 3 sets of LED + 330 Ω Resistor 4. Raspberry Pi	Firebase	IOS Device
Software & Service	1. Python(on Raspberry Pi) 2. OpenALPR	Firebase	Xcode(Swift 4)

This system contains three main modules, which are illustrated by *Diagrams 1,2,3*.

1. Monitor and display car enter/exits into/from parking lot
2. Monitor and display parking plots
3. Book available parking spaces

Shown by *Diagram 4*, the circuit diagram has some minor adjustment.

2. Design Decisions

Server side

Due to the limitation of hardware, we use one group of sensors in three situation: **entry monitor, exit monitor** and **plot monitor**.

Entry monitor should perform the motion detection and plate recognition. Here is part of design details because all the steps are depicted in *Diagram 1*.

- When a motion is detected, system will keep read status of motion sensor in 3 seconds until the motion stopped which means the system will give 3 second for the car to stop. If it stopped, plate recognition will perform, or the system will back to initial state.
- RPI needs to read history from firebase to judge if this car has already entered or not.
- About the plate recognition, the configuration of the openalpr library we used here is “auwide”. We only take the one with highest confidence of all the recognition results and upload it depends on its length(we assume plate’s length must equal to 6).
- Leds are controlled by RPI, green means no motion and red means motion detected.
- The data uploads to firebase not only include the plate, but also the time and status as shown in *Diagram 5*.

Exit monitor should perform the motion detection and plate recognition same as entry monitor. Here is part of design details because all the steps are also depicted in *Diagram 1*.

- The data uploads to firebase not only include the plate, but also the time and status as in *Diagram 6*.
- Almost all the things are similar to entry monitor except the status it monitored is “out”, which means only the car with “in” in its latest history can exit.

Plot monitor should monitor the situation of this plot includes car parking and car leaving.
Here is part of design details because all the steps are also depicted in *Diagram 2*.

- Server retrieves the latest reservation of this plot from Firebase in every iteration, then upload its current state only if any change happened.

- We define the status of a plot includes 3 states: empty 0(green LED), reserved 1(orange LED) and parked 2(red LED).
- If any motion is detected, system will execute next step by judging current state:
    - If current state is 0, and result plate is valid(not null and length equals to 6). Its state will change to 2 and this plate will be saved.
    - If current state is 2, and result plate is Null(empty string). Its state will change to 0 and its plate should set to Null.
    - If current state is 1, the RPI will try to match the plate with the one retrieve from Firebase. If successfully matched, the state of this plot will change to 2(parked). In another case, RPI will upload a “invalid” attribute with recognised plate to firebase to notice the owner of this plot that the car with this plat occupy this plot.
- System keep tracking the status of each reserved plot because we set the timeout of a reservation request is 30 minutes. It means that the server will check the reservation time and update plot’s status if it is out of time.
- If current plot has an “invalid” attribute, which means some car has already occupied this reserved plot, this expiration of reservation will move this “invalid” plate to “plate” attribute and set it to 2(parked), or set it to 0(empty). Its data structure in Firebase is shown in *Diagram 7*.

Client side
IOS application is design in following modules: **login/registration, car management, my information** and **plot reservation.**
Login/registration manages authentication and account registration.
- Login: The email and password user input uses Firebase authentication to help with login, the format of input for email will be validated.
- Register: The email and password will be used to create a user in Firebase authentication. All other information will be stored in the (users/<userid>) as shown by *Diagram 8*. All input are validated real time.

Car management manage all the car that this user owned.
- All the car information are retrieved from Firebase. Data structure in Firebase is (users/<userid>/plates): as shown in *Diagram 9*.
- We assume that only one of these cars can enter in(be tracked in) the parking lot. “Current” attribute represents the index of car plate which has already entered the parking lot (-1 means no car is currently inside)
- IOS application displays the information of “current” car with its lasting time from latest “in” entry history.
- User can add new car by typing its plate. He/She can also modify saved car plate. All the changes will upload to Firebase.

My information helps users to view/edit their personal profile
- The view page retrieves the user profile from Firebase (users/<userid>) and display them. The Edit options is provided.
- The edit page retrieves the user profile from Firebase and puts them into text fields accordingly. All input are validated real time. Once ‘Save’ button is clicked, the information will be upload to Firebase to update user profile.

Plot reservation is the page that user can view and make reservation on all the plots.
- Similar to the server side, car status and color includes same 3 couples: empty 0(green car), reserved 1(orange car) and parked 2(red car). But there is additional special couple: reserved 1(red car) which represents this plot is reserved by another user that you cannot make any change on it.
- User can make a reservation on an empty plot. Then a order attribute with current time is set to Firebase. User can also cancel a reservation of a reserved 1(orange car) plot, then if there is an invalid car, its state will change to parked 2, or it will change to empty 0.

- User will receive an alert if his reserved plot is occupied by another car by retrieving the “invalid” attribute. User will also receive an alert if his reservation is expired by retrieving “order” attribute.

3. Libraries

1. Openalpr, a python library is used. It is implemented on RPI, and is used after a picture is taken. It contributes the functionality of plate recognition from photos.
   Ref: https://github.com/openalpr/openalpr
2. Firebase, libraries for swift and python are used. The libraries are used when there are needs to interact with Firebase (e.g. login, signup, update data etc.)
3. LGButton, a button library contributes to UI optimization.
   Ref: https://github.com/loregr/LGButton
4. SkyFloatingLabelTextField, a TextField library contributes to UI optimization.
   Ref: https://github.com/Skyscanner/SkyFloatingLabelTextField

4. Notes

1. The functionalities of payment and fee calculation are not implemented because they are not the main parts of this IoT project and requires too much time.
2. The Camera and PIR sensor are used for both vehicle entry/exit and parking spot monitor. In actual use of this project, there should be 1 set for entry, 1 set for exit, and 1 set for each parking spot.

5. Appendix

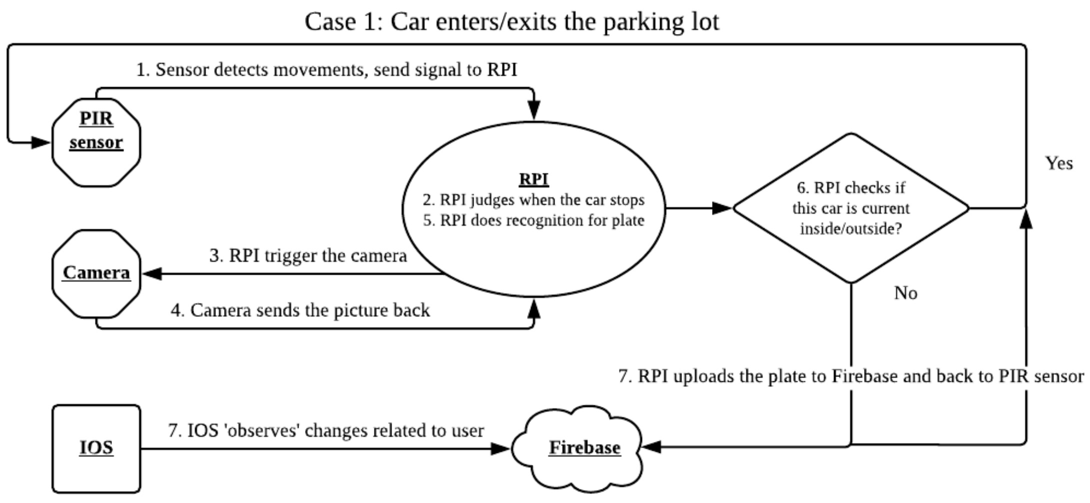

Case 1: Car enters/exits the parking lot
1. Sensor detects movements, send signal to RPI
PIR sensor
RPI
2. RPI judges when the car stops
5. RPI does recognition for plate
6. RPI checks if this car is current inside/outside?
Yes
No
Camera
3. RPI trigger the camera
4. Camera sends the picture back
7. RPI uploads the plate to Firebase and back to PIR sensor
IOS
7. IOS 'observes' changes related to user
Firebase

Diagram 1

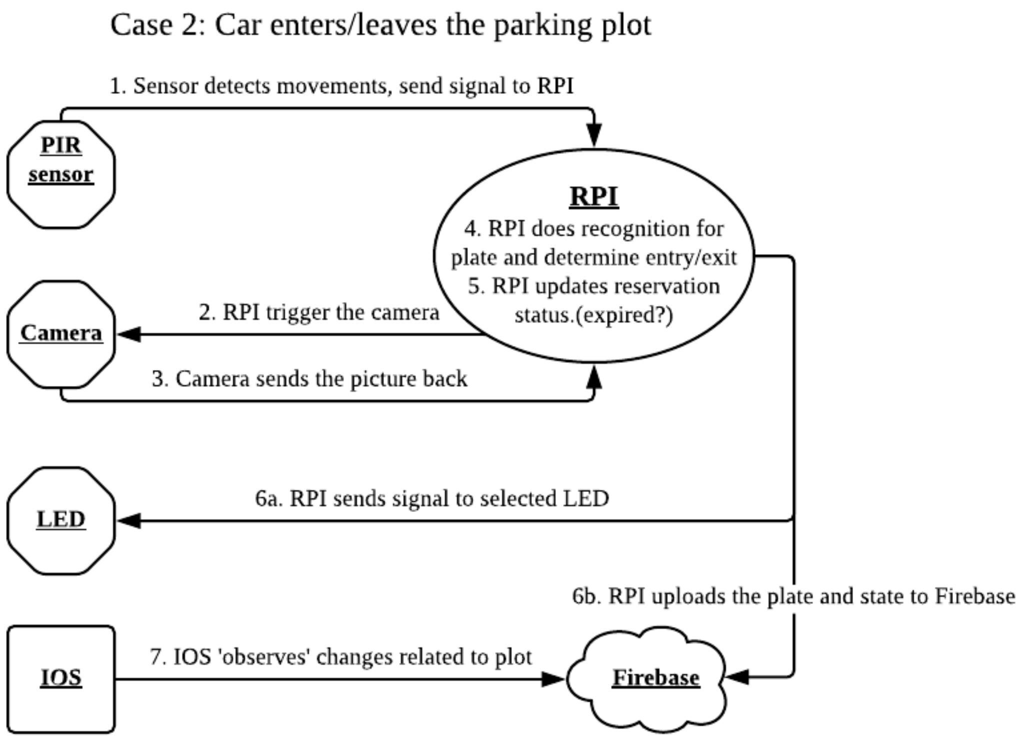

Case 2: Car enters/leaves the parking plot
1. Sensor detects movements, send signal to RPI
PIR sensor
RPI
4. RPI does recognition for plate and determine entry/exit
5. RPI updates reservation status.(expired?)
Camera
2. RPI trigger the camera
3. Camera sends the picture back
LED
6a. RPI sends signal to selected LED
6b. RPI uploads the plate and state to Firebase
IOS
7. IOS 'observes' changes related to plot
Firebase

Diagram 2

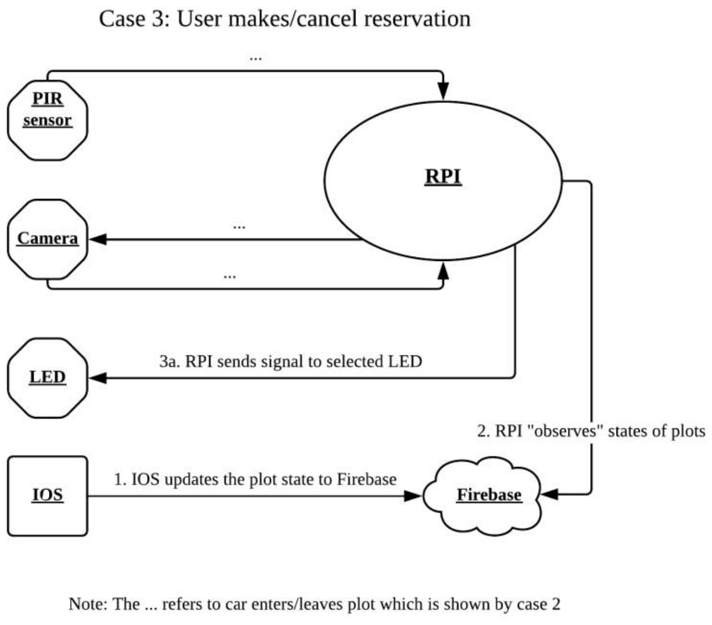

Case 3: User makes/cancel reservation
...
PIR sensor
RPI
Camera
...
...
LED
3a. RPI sends signal to selected LED
2. RPI "observes" states of plots
IOS
1. IOS updates the plot state to Firebase
Firebase
Note: The ... refers to car enters/leaves plot which is shown by case 2

Diagram 3

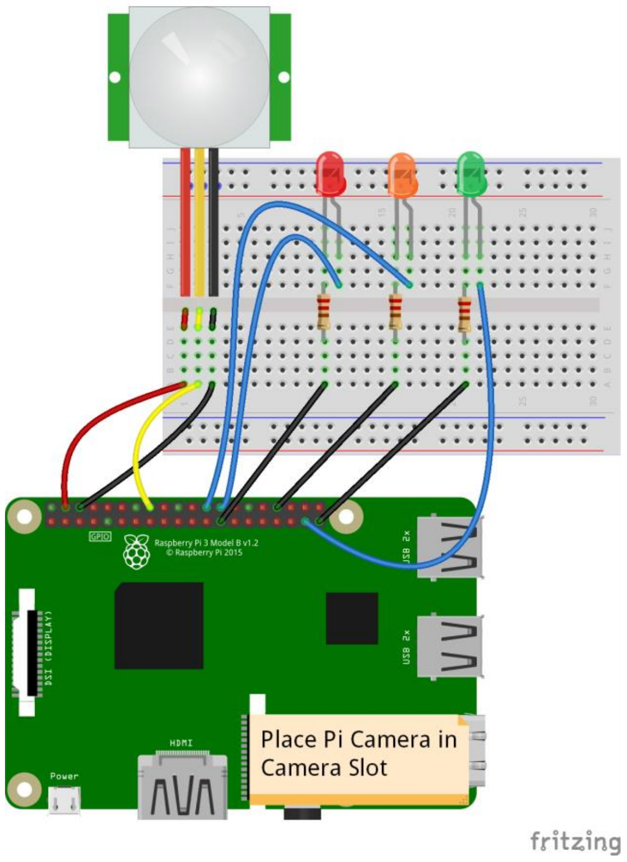
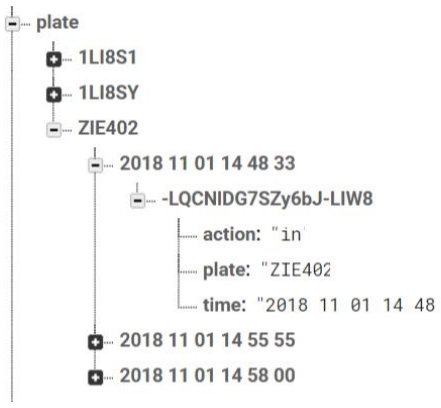
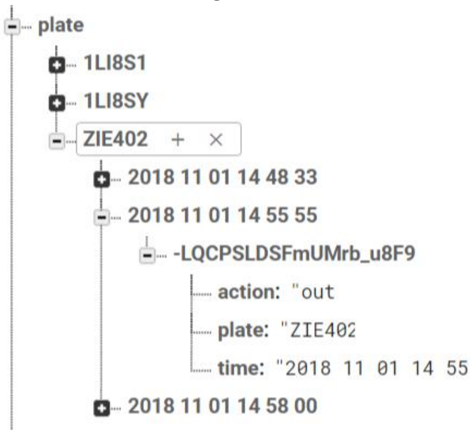


Diagram 4



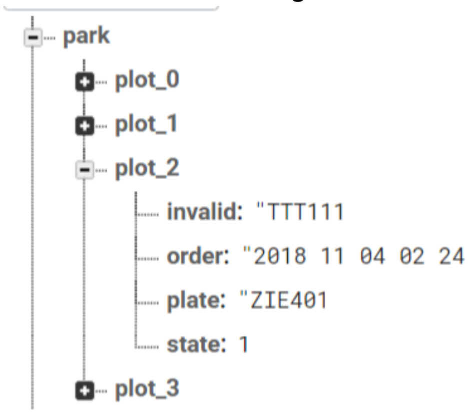
```
plate
  1LI8S1
  1LI8SY
  ZIE402
    2018 11 01 14 48 33
      -LQCNIDG7SZy6bJ-LIW8
        action: "in
        plate: "ZIE402
        time: "2018 11 01 14 48
    2018 11 01 14 55 55
    2018 11 01 14 58 00
```

Diagram 5



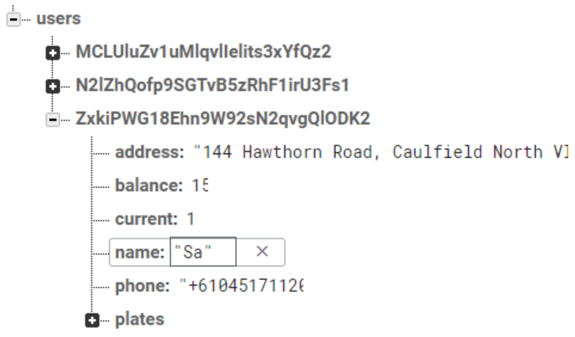
```
plate
  1LI8S1
  1LI8SY
  ZIE402 + ×
    2018 11 01 14 48 33
    2018 11 01 14 55 55
      -LQCPSLDSFmUMrb_u8F9
        action: "out
        plate: "ZIE402
        time: "2018 11 01 14 55
    2018 11 01 14 58 00
```

Diagram 6



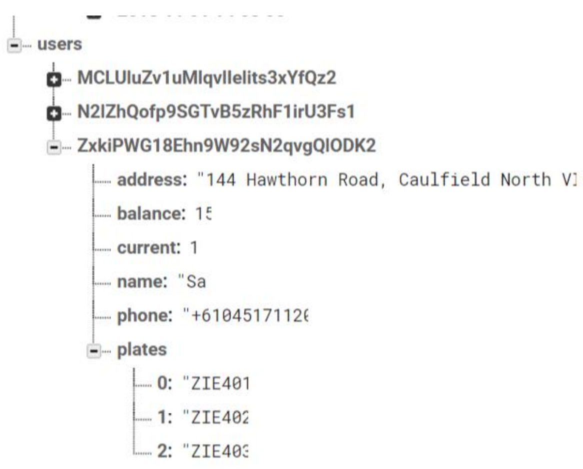
```
park
  plot_0
  plot_1
  plot_2
    invalid: "TTT111
    order: "2018 11 04 02 24
    plate: "ZIE401
    state: 1
  plot_3
```

Diagram 7



```
users
  MCLUluZv1uMlqvlIelits3xYfQz2
  N2lZhQofp9SGTvB5zRhF1irU3Fs1
  ZxkiPWG18Ehn9W92sN2qvgQlODK2
    address: "144 Hawthorn Road, Caulfield North VI
    balance: 15
    current: 1
    name: "Sa" ×
    phone: "+6104517112
    plates
```

Diagram 8



```
users
  MCLUluZv1uMlqvlIelits3xYfQz2
  N2lZhQofp9SGTvB5zRhF1irU3Fs1
  ZxkiPWG18Ehn9W92sN2qvgQIODK2
    address: "144 Hawthorn Road, Caulfield North VI
    balance: 15
    current: 1
    name: "Sa
    phone: "+6104517112
    plates
      0: "ZIE401
      1: "ZIE402
      2: "ZIE403
```

Diagram 9