CEN 415
Open-Ended IOT System

Omar Aoun 1059398
Mahmoud Bakir 1060398
Ahmad Hashi 1046819
Omar Altawil 1047982

Supervised By: Eng. Yasmina Al Khalil

جامعة أبوظبي
ABU DHABI UNIVERSITY

Submitted: June 5, 2018

Contents

List of Figures

List of Tables

Abstract

In this project, we are creating a smart home system that lets the person monitor the status of a room from an app. First thing we did is the connections, we added four sensors which were IR and we connected it to the Raspberry Pi. In this project we had to use our IOT system to identify the number of current people in Room A. We used Firebase to maintain a Real-Time feedback of the current status of Room A.

1 Motivation

Our motivation in this project was to make an embedded system that can detect the number of people who were in a certain place. The system should recognize the individual who's entering the room and the one who's leaving in order to keep a clean count of the current status of the selected place or room. We were asked to use A Raspberry Pi as our micro-controller and add up whatever circuitry needed in order to make the system functioning.

2 Introduction

In this project, we are trying to make a smart system that can monitor the activity of people in a specified room. The main idea is to have a group of sensors that can produce a variety of outputs and then collect these outputs and analyze it.Based on the acquired outputs, the system has to tell the user about the status of the specified room, meaning that the system must know the following:-

1. The number of people who left Room A.
2. The number of people who entered Room A.
3. The number of current people in Room A.

Our smart system must then present the data in a Mobile Application with real-time data about the status of Room A. This smart system requires the use of sensors and a micro-controller.

3 Experimental Setup

The list of all of everything that we have used in our project:-

- **Raspberry Pi:** The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python [1]

- **IR Sensor:** IR Sensor works by using a specific light sensor to detect a select light wavelength in the Infra-Red (IR) spectrum. The sensor has two main parts, one part sends the IR while the other part senses the reflected light, so it is possible to have a sensor that can return the value of the reflected light and it can be used to measure how ”bright” the object is [3]

- **Jumper wires:** Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering [2].

- **GoPro Camera:** A camera type that we used throughout our project.

4 Procedure

4.1 The Hardware & the State Chart

Our hardware circuit had the components that were mentioned in the equipments section above.We started by setting up the hardware connection of our system, we used Raspberry Pi and two IR sensors. We wanted to use a Laser sensor due to its high accuracy but we couldn’t obtain any Laser sensor, so we went with the IR which is less accurate but it does the job. Figure 1 shows the connection. We assumed that our Raspberry Pi consumes 1500 mAH when using Simple CV, we also assumed that the Pi’s LSD screen needs 200 mAH to work. The IR sensor takes around 50 mAH (2 takes 100 mAH) and the camera takes around 200 mAH. So, the total of our hypothetical power shecme for the project is around 2000 mAH.We picked a BQY Power 11.1V Lipo Battery with a capacity of 2200 mAH for our project, we can operate for approximately one hour using one battery. Figure 2shows the Battery. After that, we designed our State Chart. Figure 3 shows the State Chart.

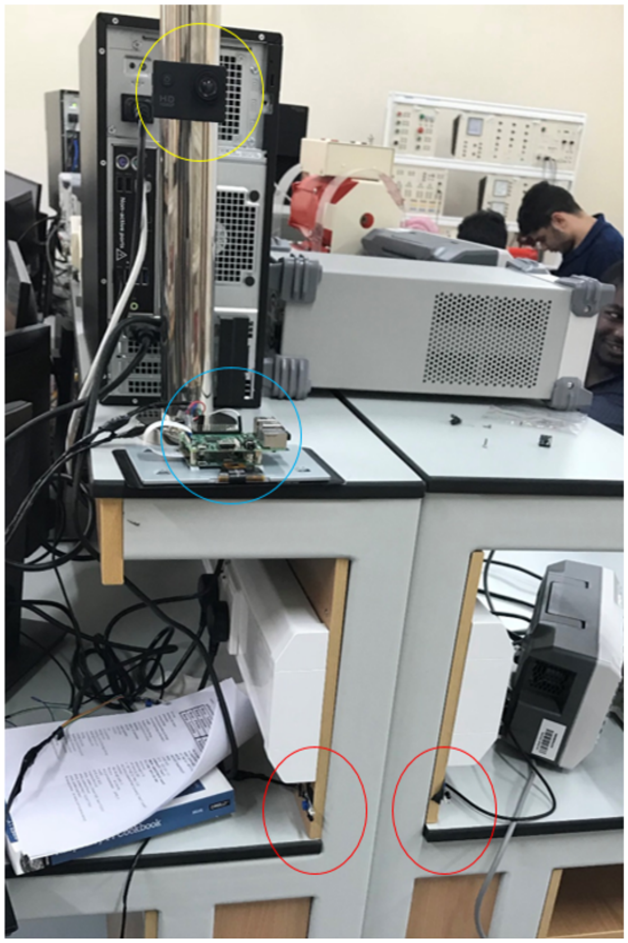


Figure 1: The Circuit Connections

BQY
11.1V
2200mAh
Li-Poly
Rechargeable
Battery
Caution:
Please read charging instruction before use.
Improper use can cause explosion or fire.
Use Specified charger only.
Do not disassemble or put the batteries in the fire.

Figure 2: The Lipo Batery

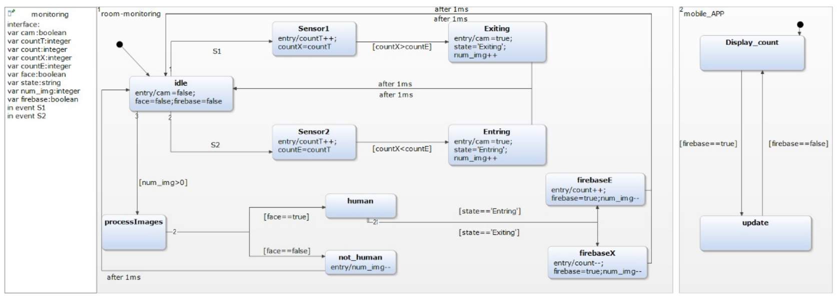


Figure 3: The State Chart

4.2 The Raspberry Pi code

We wrote our Python code for the Raspberry Pi. We had written a total of two Algorithm.The first Algorithm's main idea was to capture a picture when the IR reacts to a motion an then process the picture using Simple CV.By using feature detection, we can determine whether the motion is traced back to a person, also we can know whether the person is entering or leaving the room by using the face feature for identifying entering and the full body feature for identifying exiting using the help of Simple CV. Although the idea behind this Algorithm was simple,the problem with this Algorithm was the inaccuracy that we faced when trying to detect the full body feature which made the Algorithm kind of faulty when trying to identify the exiting state of a person.The code of the first Algorithm was not included in the report because we didn't use it, but its going t be included in the Appendix.

The second Algorithm used two IR sensor's. It's main idea was to use the two sensors to identify whether the state of the person is entering or exiting.This Algorithm was more accurate comparing to the first Algorithm and the entering and exiting states doesn't depend on the camera, the drawbacks of this Algorithm was it's complexity comparing to the first Algorithm, the positioning of the camera has to be perfect and the usage of two IR sensors instead of one.The code of the second Algorithm is listed downward.

```python
#Algorithm 2
#Here, we import all the librarys that we will use in this project
from SimpleCV import Image
from SimpleCV import Color, ColorCurve, Camera, Image, pg, np, cv
from SimpleCV.Display import Display
import RPi.GPIO as GPIO
import time
from time import gmtime, strftime
from firebase import firebase
from google.cloud import storage
import os

#Here, we initialize our global variables
feature=0
countEnt=0
countExi=0
countT=0
```

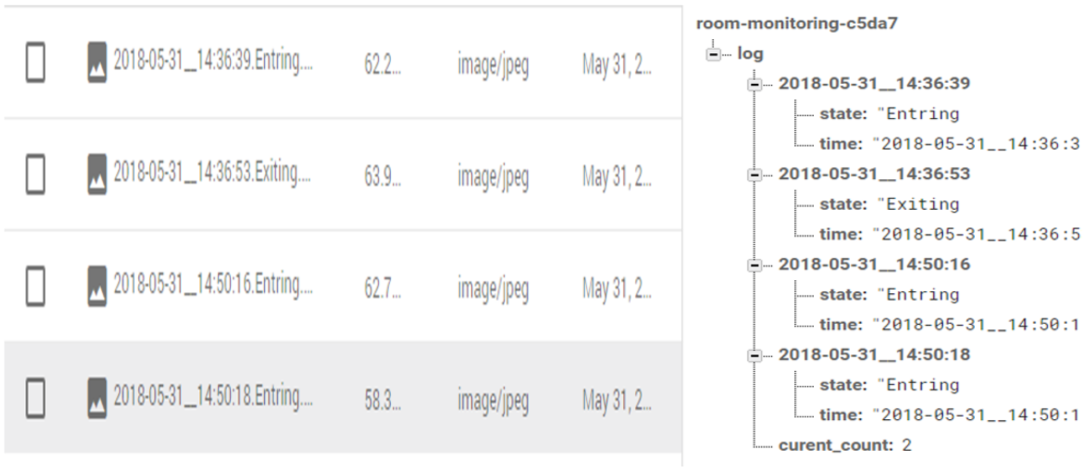
```
count=0
#Here, we initialize the Camera instants and select the resolution
cam = Camera()
cam.resolution = (1920, 1080)
#Here, the getimge() function will tack a picture and save it to
   vision directory with the name we pass
def getimge(s):
      img = cam.getImage().save('/home/pi/Desktop/vision/'+s+'.jpg')
  return Image('/home/pi/Desktop/vision/'+s+'.jpg')
#Here, the exiting() function is the interrupt Function for the
   first IR sensor, it will increment the global #variables countT
   and save it in countExi then call the function check()
def exiting(channel):
    global countExi
    global countT
    countT+=1
    countExi=countT
    check()
#Here, the entering() function is the interrupt function for the
   second IR sensor, it will increment the #global variables
   countT and save it in countEnt then call the function check()
def entering(channel):
    global countEnt
    global countT
    countT+=1
    countEnt=countT
    check()
#Here, the check() function will be called every time an interrupt
   happens to check if the two sensors #have started so it can
   determine the direction of movement (enter or exit).It does
   that by comparing #the value of countEnt and countExi, if one
   of them is 0 that means thats one of the sensors did not #start
   so it will exit.
#if countEnt is greater than than countExi, that means the
   entering sensor have been started after the #exiting sensor so
   the state will be entering. On the other hand, if countExi is
   greater than countEnt #that means the exiting sensor have been
   started after the entering sensor so the state will be exiting.
#If we enter one of these two states we will get the exact time
   using gmtime() function and save it in a #local variable called
```

```
    nowtime and we sprat it and the state by a(.)
#Then we will order the camera to take an image which it name will
    be the value of nowtime and save it #in the vision directory so
    it can be processed later.
def check():
    global countExi
    global countEnt
    if countExi==0 or countEnt ==0:
      return
    elif countExi > countEnt:
      nowtime=strftime("%Y-%m-%d__%H:%M:%S.Exiting", gmtime())
      img=getimge(nowtime)
      print('exiting')
      countExi=0
      countEnt=0
    elif countExi < countEnt:
      nowtime=strftime("%Y-%m-%d__%H:%M:%S.Entring", gmtime())
      img=getimge(nowtime)
      print('entring')
      countExi=0
      countEnt=0
#Here, we initialize the location of the json file that hold
    CREDENTIALS key for our google storage and #we pass the name of
    project that we want to read from.
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] =
    '/home/pi/.config/gcloud/application_default_credentials.json'
client = storage.Client('room-monitoring-c5da7')
bucket = client.get_bucket('room-monitoring-c5da7.appspot.com')
#Here, we connect to firebase real-time database
firebase =
    firebase.FirebaseApplication('https://room-monitoring-c5da7.firebaseio.com/',
    None)
#Here, we initialize the pins as input pin with a callback
    function as interrupt
#Here, the bouncetime=500 so we can give the sensor enough time to
    reset before calling the interrupt #again
GPIO.setmode(GPIO.BCM)
GPIO.setup(15, GPIO.IN)
GPIO.setup(18, GPIO.IN)
GPIO.add_event_detect(15,GPIO.FALLING,
    callback=exiting,bouncetime=500)
GPIO.add_event_detect(18,GPIO.FALLING,
    callback=entering,bouncetime=500)
#this will loop forever, it will keep checking the file vision for
    any new images to process(if interrupt #happen it will execute
    the interrupt then come back to continue executing the loop).
while True:
#Here, we iterate for each image inside the vision file
    for filenames in os.listdir('/home/pi/Desktop/vision'):
          time.sleep(0.01)
#Here, we initialize the file as image so we can process it using
    SimpleCV library
          img=Image('/home/pi/Desktop/vision/'+filenames)
         #Here, we split the time and the state(first part will
             contain the time and the second will contain #the
             state[entring,exiting])
          state=filenames.split('.')

          #Here, we process the Image for face detection, if face
              is found we increment feature
          #We use multiple feature to increase the face detection
              accuracy.
          faces = img.findHaarFeatures('face.xml', min_neighbors=5)
          if faces:
              feature+=1
          face2 = img.findHaarFeatures('face2.xml', min_neighbors=3)
          if face2:
              feature+=1
          face3 = img.findHaarFeatures('face3.xml', min_neighbors=3)
          if face3:
              feature+=1
          face4 = img.findHaarFeatures('face4.xml', min_neighbors=3)
          if face4:
              feature+=1
          face_cv2 = img.findHaarFeatures('face_cv2.xml',
              min_neighbors=3)
          if face_cv2:
              feature+=1
          print(feature)
#If we find at least 3 features out of 6 then a human face is
```

```
    detected.
#we first check number of faces in the images(in case multiple
    people enter/exit at once)
#We will add all his/her data to the real-time database and
    increment or decrement the count(number of #people on the room)
    depending on the state.
#Finally, we will delete the image from the file so we dont waste
    time in processing it again.
          if feature>2:
             if state[1] == 'Entring':
              count=couny+len(faces)
             elif state[1] =='Exiting':
              count=count-len(faces)
             firebase.patch('/log/' ,{'curent_count': count})
             firebase.patch('/log/'+
                 state[0],{'state':state[1],'time':state[0]})
             Blob = bucket.blob(filenames)
             Blob.upload_from_filename(filename='/home/pi/Desktop/vision/'+filenames)
             print('entring')
             feature=0
             os.remove('/home/pi/Desktop/vision/'+filenames)

#If we found less than 3 features, then that means a human is not
    in the picture, so we delate the #image.
          else:
             os.remove('/home/pi/Desktop/vision/'+filenames)
```



2018-05-31__14:36:39.Entring....	62.2...	image/jpeg	May 31, 2...
2018-05-31__14:36:53.Exiting....	63.9...	image/jpeg	May 31, 2...
2018-05-31__14:50:16.Entring....	62.7...	image/jpeg	May 31, 2...
2018-05-31__14:50:18.Entring....	58.3...	image/jpeg	May 31, 2...

```
room-monitoring-c5da7
  log
    2018-05-31__14:36:39
      state: "Entring
      time: "2018-05-31__14:36:3
    2018-05-31__14:36:53
      state: "Exiting
      time: "2018-05-31__14:36:5
    2018-05-31__14:50:16
      state: "Entring
      time: "2018-05-31__14:50:1
    2018-05-31__14:50:18
      state: "Entring
      time: "2018-05-31__14:50:1
    curent_count: 2
```

Figure 4: Real-Time Database & Cloud Storage

We used Firebase(real-time database,storage) to save all the acquired data and images. Figure 4 above shows a screen shot of the real-time database and the storage.

4.3 The Mobile Application

After finishing the code of the Raspberry Pi, we start in making the mobile application that can be used to monitor our IOT system. The following code are for the Monitor part of the mobile application. The java code of the Android application is shown downward.

```java
public class MainActivity extends AppCompatActivity {
    private TextView count; // Display list

    // Fire base code
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference log = database.getReference("log");
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        count = (TextView) findViewById(R.id.count);

        // Read from the database
        log.child("curent_count").addValueEventListener(new
            ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                int value = dataSnapshot.getValue(Integer.class);
                count.setText(Integer.toString(value));
            }

            @Override
            public void onCancelled(DatabaseError error) {
            }
        });
    }
}
```

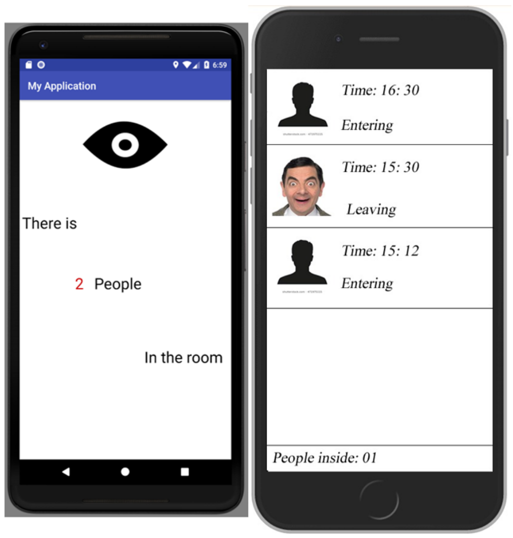


Figure 5: The Mobile Application

We got something simple as shown in figure 5. As we can see, we have two pictures of our app. The picture on the right represents the way our mobile interface should have looked like, but due to not having enough time, we made it simpler which is shown in the left part of the picture.

5 Conclusion

In the end, we learned a lot doing this project. We learned the basis of how IOT works in general and how to implement on a fundamental level, and how we can use it for us to monitor a smart home. Even though we only did a small project compared to other Open-Ended IOT appliances, but still we learned a handful number of tricks to start. We learned how to connect the Raspberry Pi with sensors, also learned how to create a mobile application and connect it to the sensors and the server which helped us in monitoring the data using the app.

6 Appendix

1. The code of Algorithm one.

2. Mobile Application source code

References

[1] What is a Raspberry Pi? (n.d.). Retrieved from https://www.raspberrypi.org/help/what- is-a-raspberry-pi/

[2] Hemmings, M. (2018). What is a Jumper Wire?. Retrieved from http://blog.sparkfuneducation.com/what-is-jumper-wire

[3] IR Sensor — What is an IR Sensor?. (2018). Retrieved from http://education.rec.ri.cmu.edu/content/electronics/boe/ir/sensor/1.html