SMART PARKING

TANMAY PATHAK - 2018102023
UTKARSH MISHRA - 2018102020
HASIR MUSHTAQ - 2018102049

INDEX

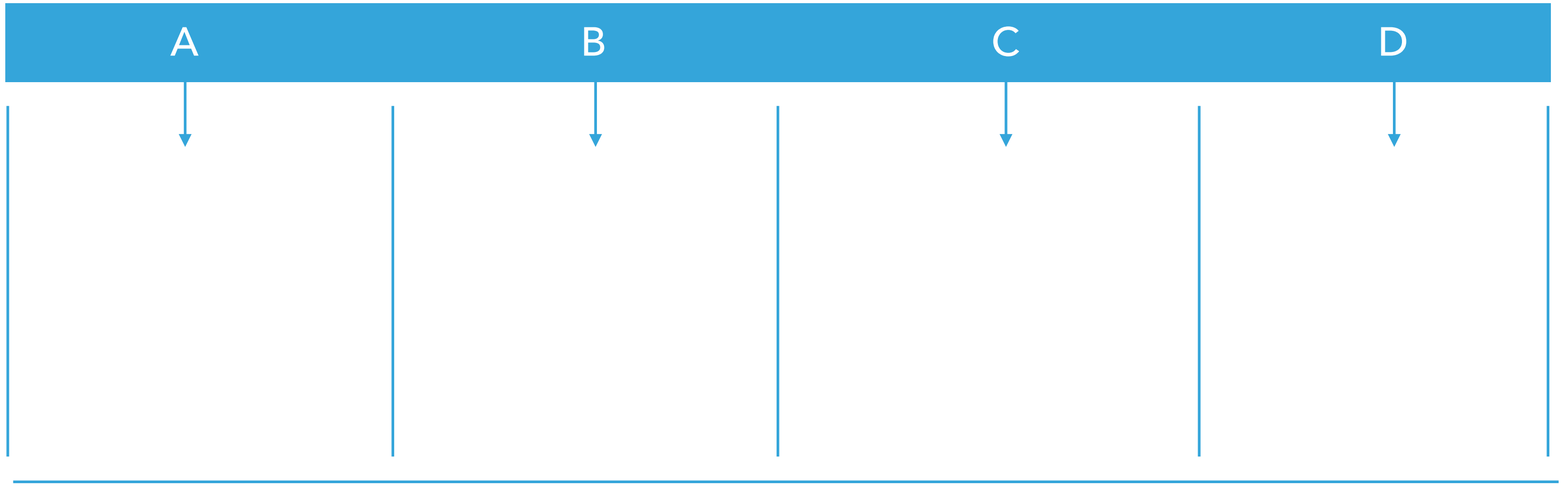
ISSUES WITH CURRENT SYSTEM

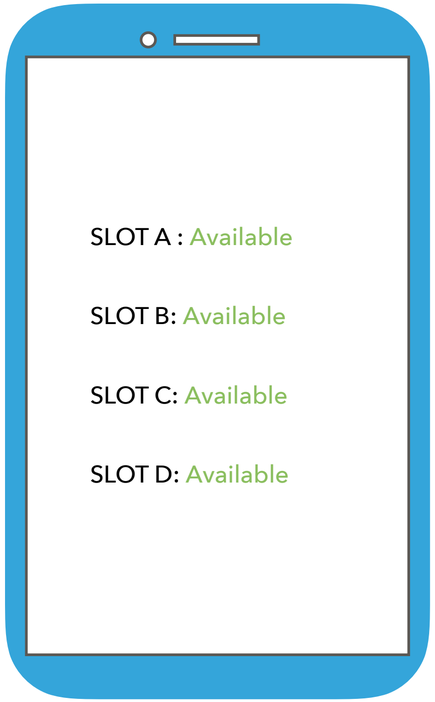
- No information about availability at entrance of parking lot
- Parking counters don’t exactly specify **where** slots are available
- Light Indicators don’t fully resolve the problem
- Absence of autonomous billing

PROPOSED SYSTEM

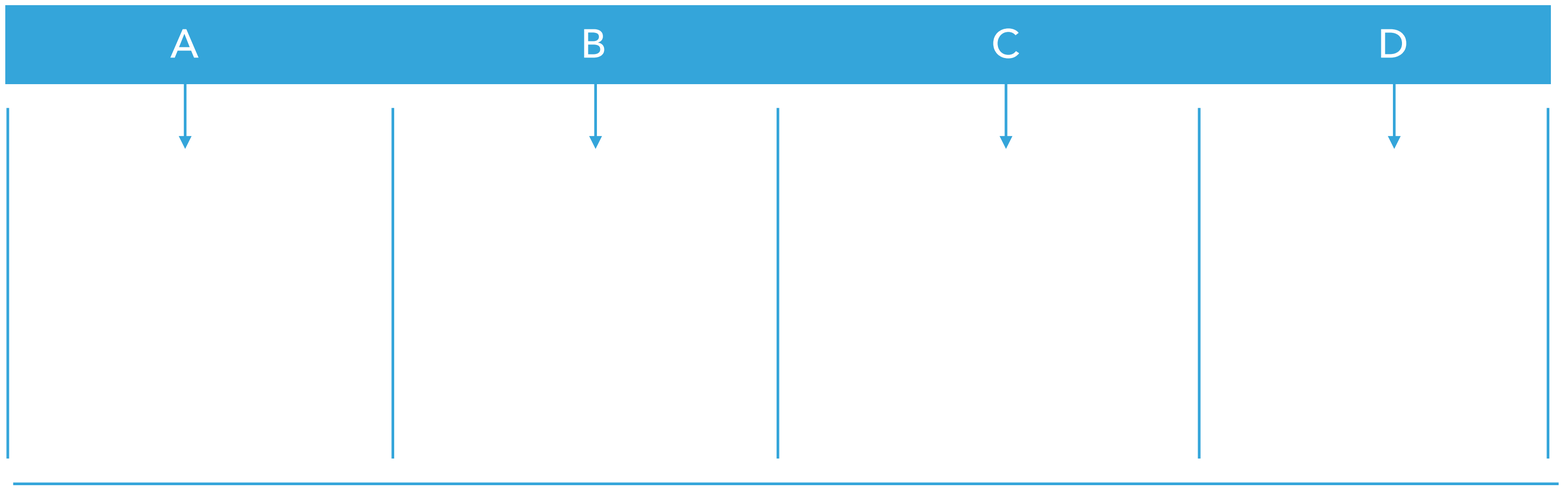
- Visitors will be able to access information about **each** parking slots via the internet
- **Live availability** information will help find parking spots faster
- Autonomous billing (using timers at the parking spot) will further ease the process

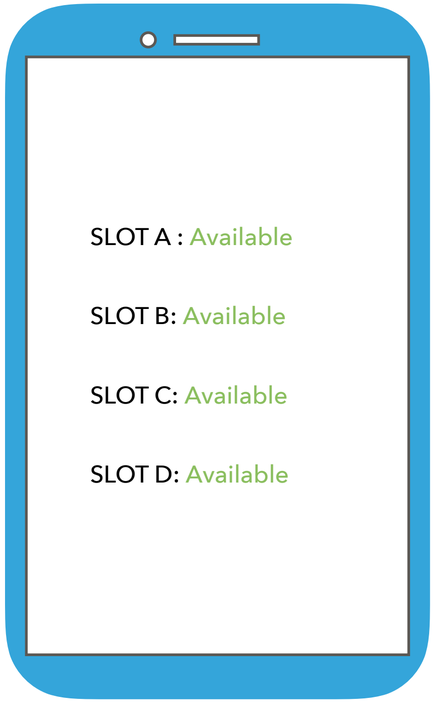
DEMONSTRATION



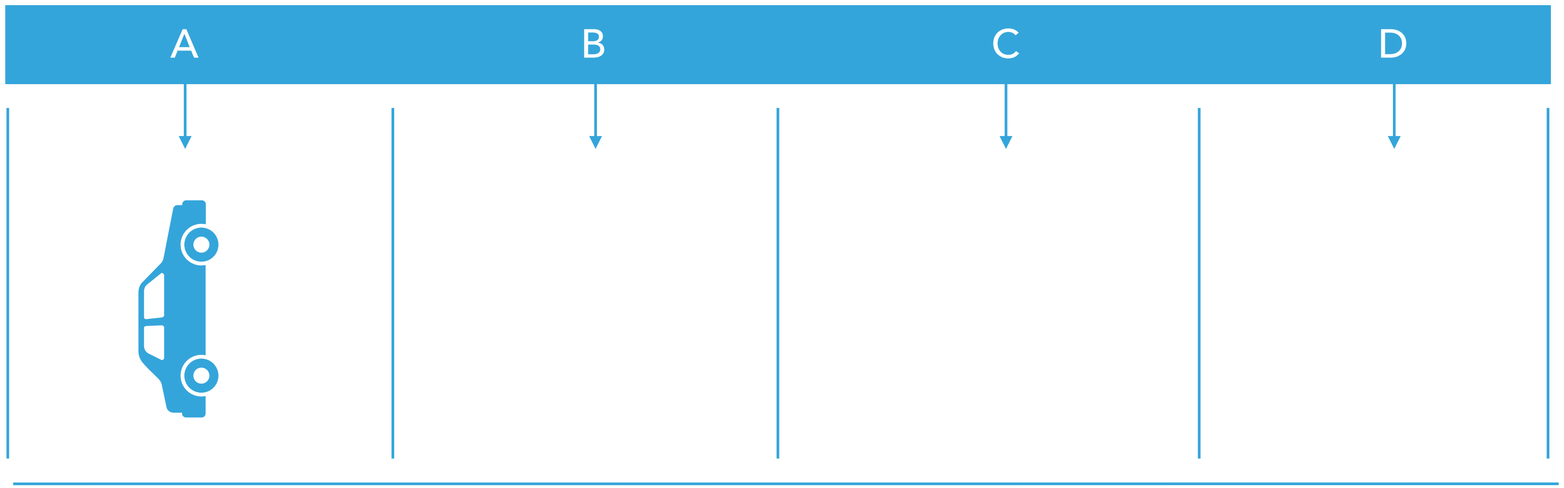


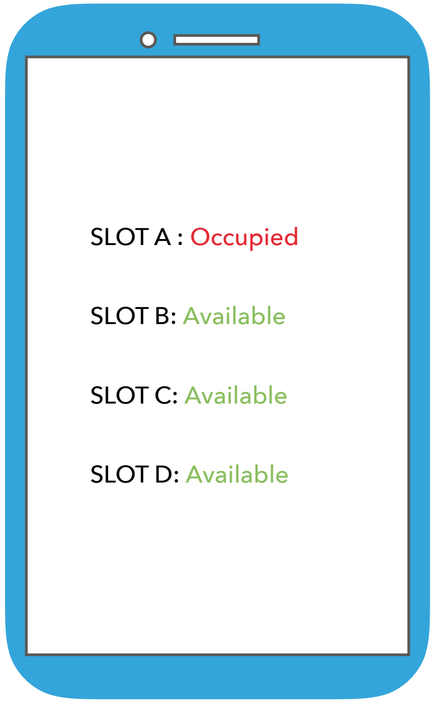
DEMONSTRATION



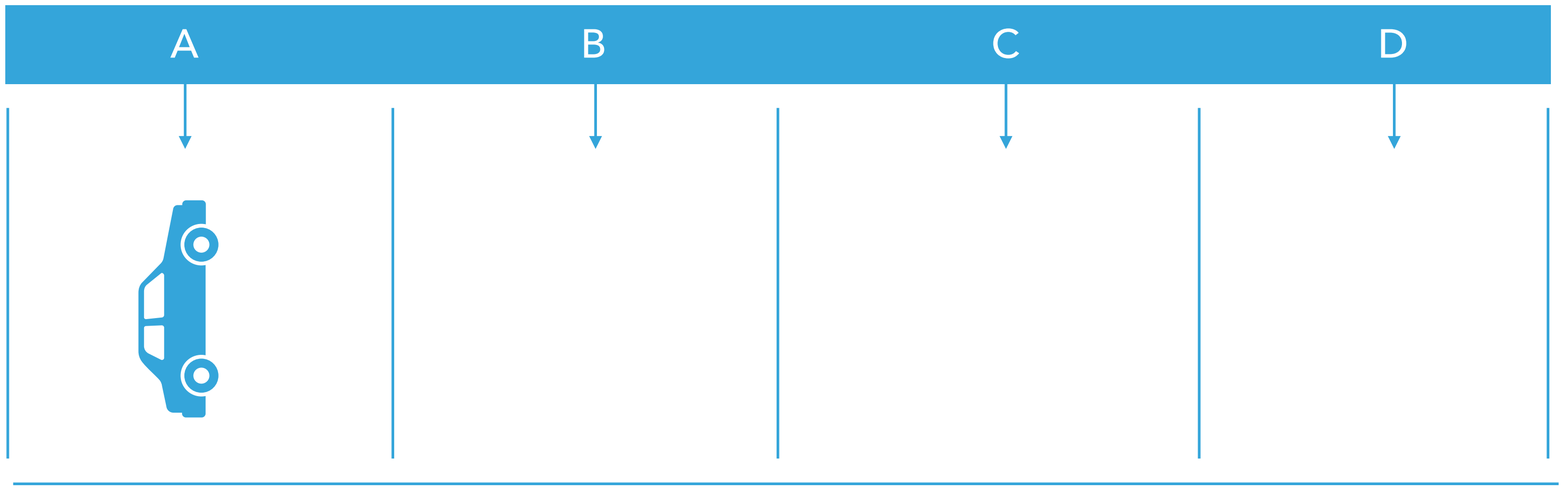


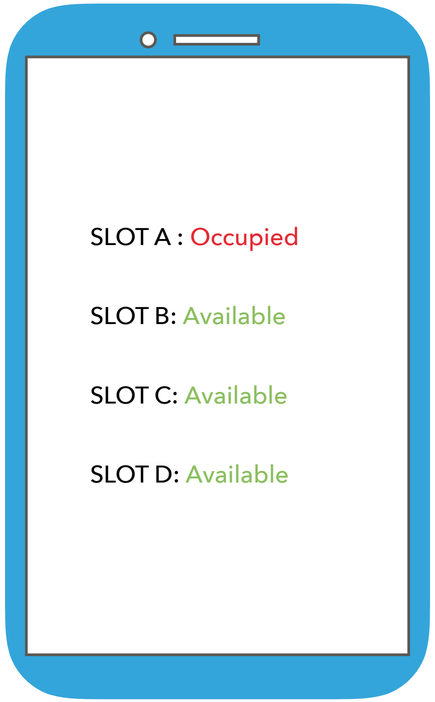
DEMONSTRATION



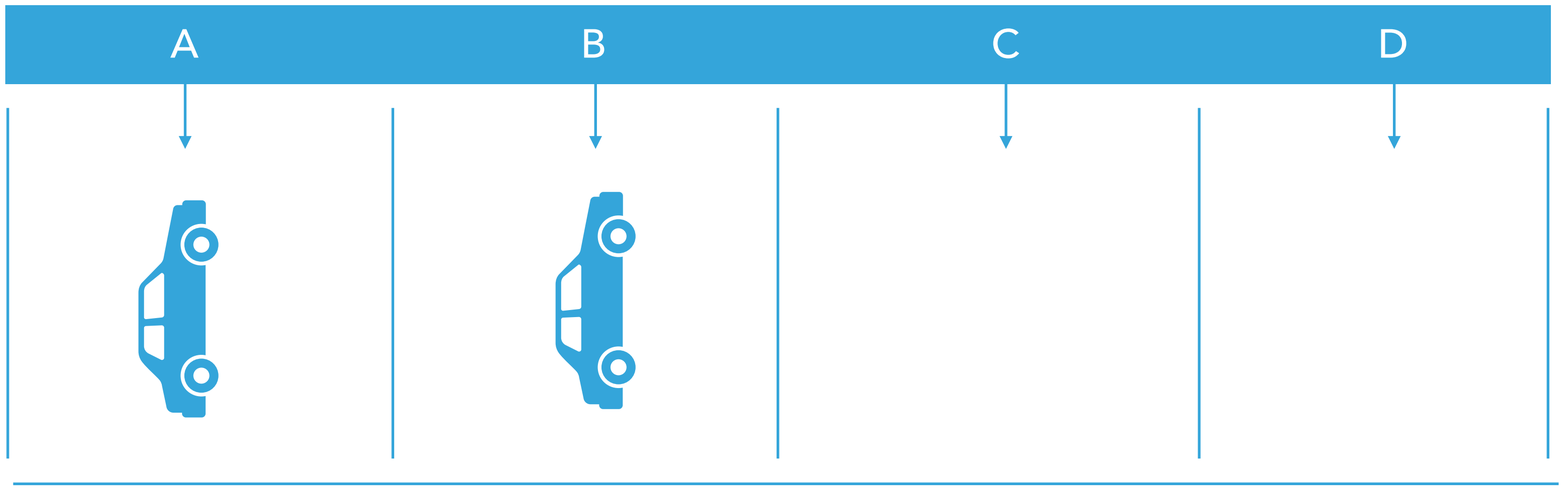


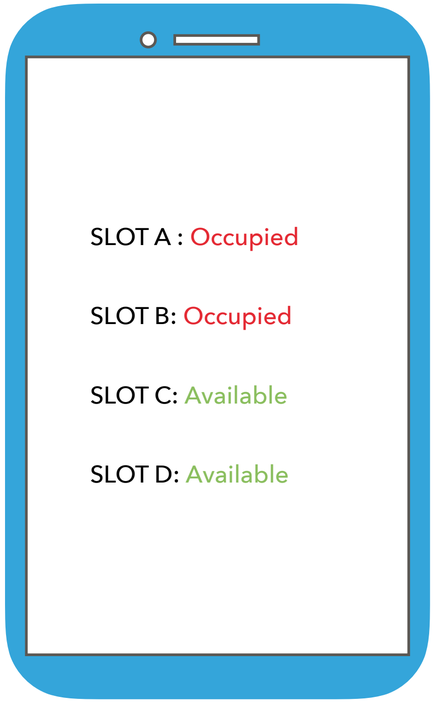
DEMONSTRATION



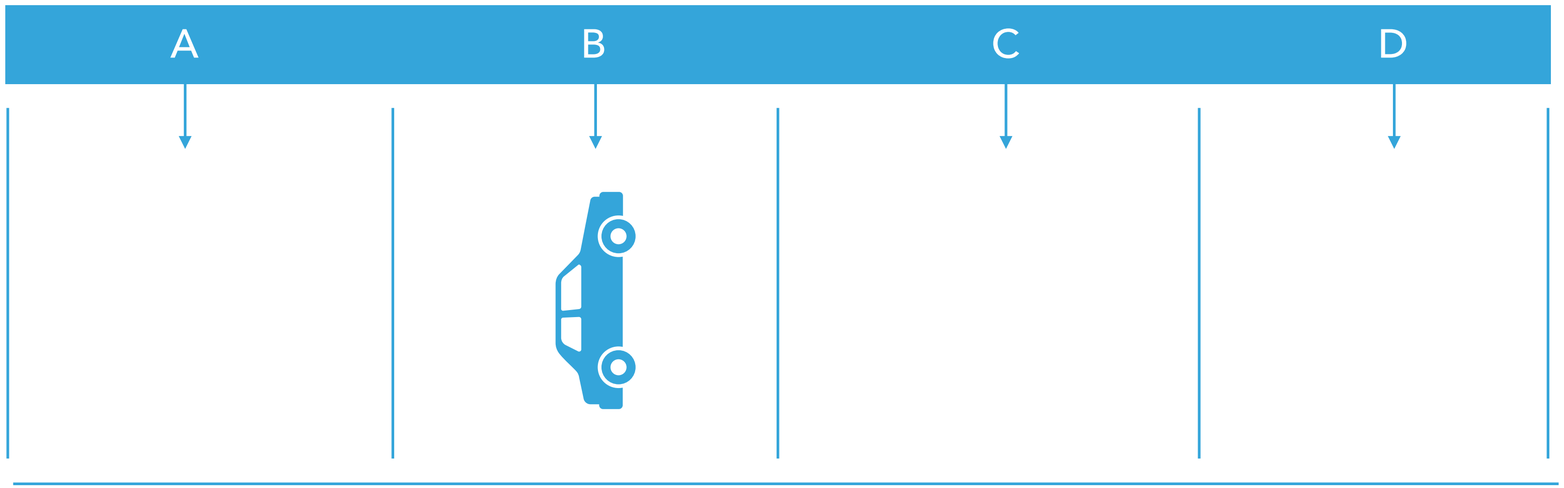


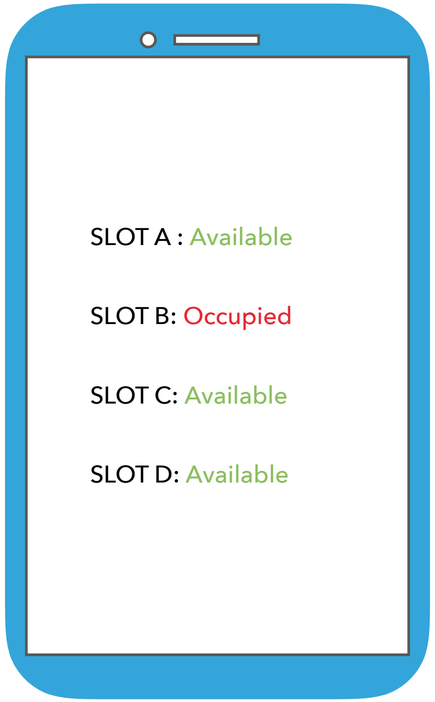
DEMONSTRATION





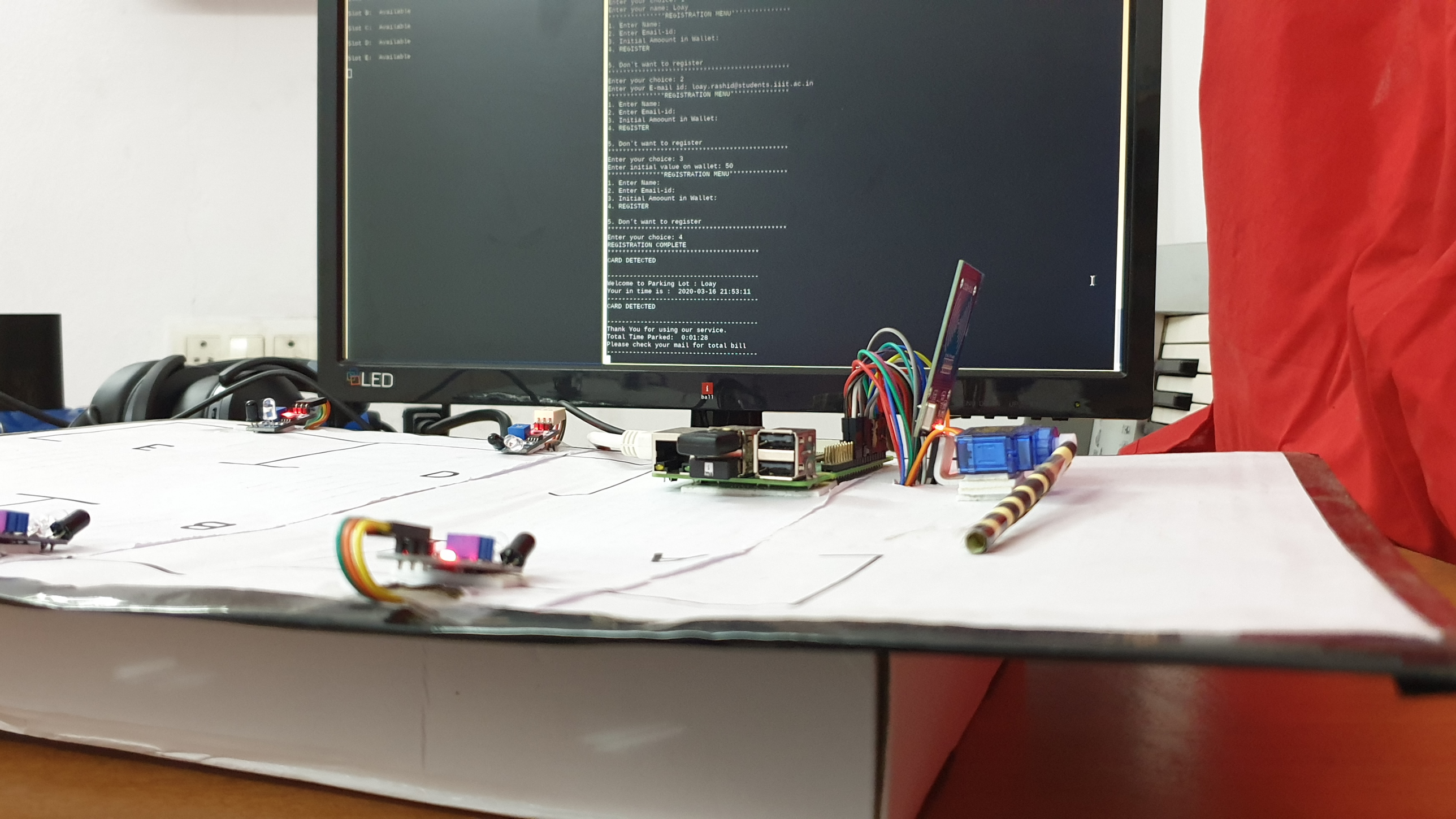
DEMONSTRATION





PROJECT PICTURES

Final Project Hardware

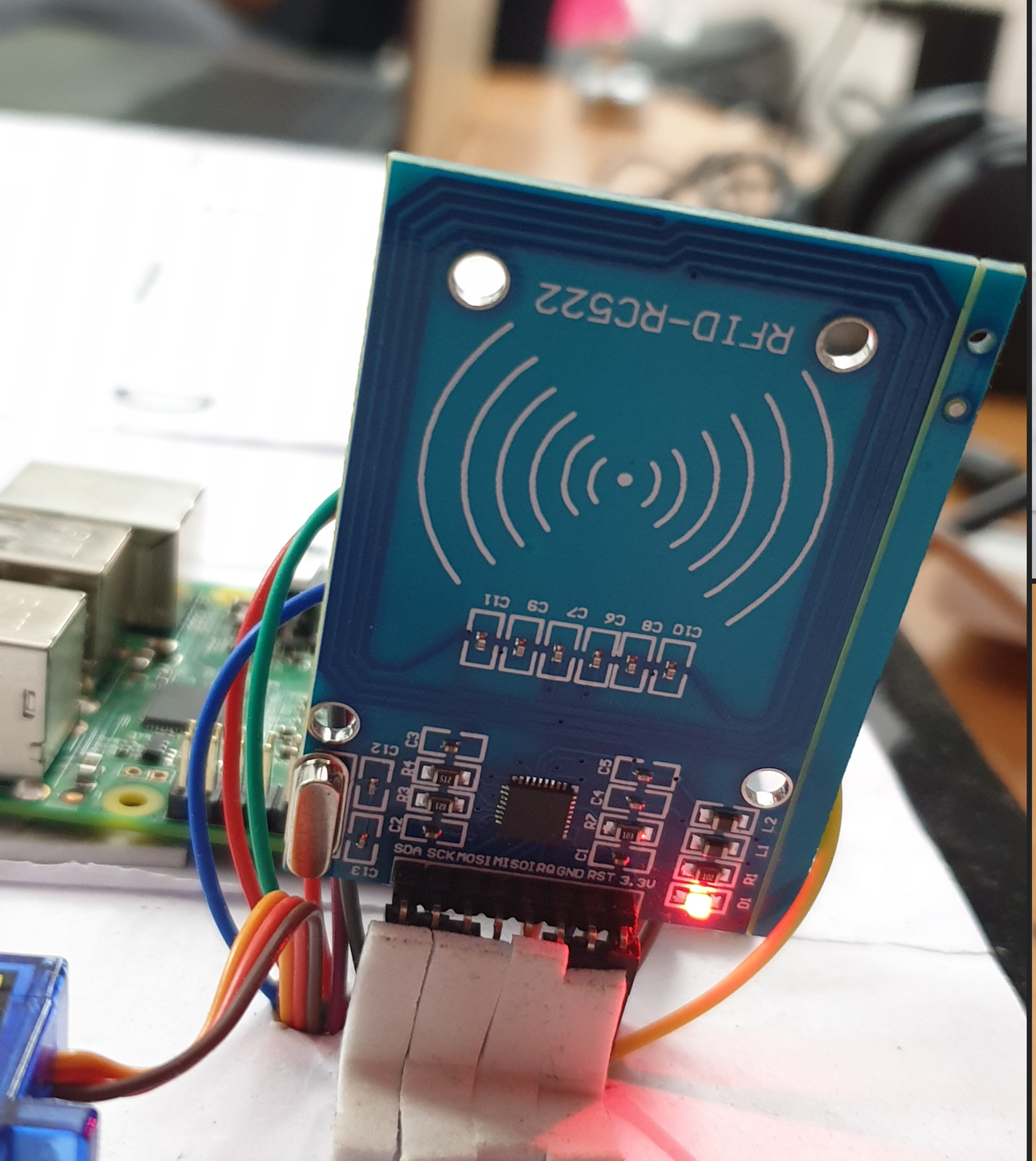

Slot B: Available
Slot C: Available
Slot D: Available
Slot E: Available
Enter your choice: 1
Enter your name: Loay
**********REGISTRATION MENU**********
1. Enter Name:
2. Enter Email-id:
3. Initial Amoount in Wallet:
4. REGISTER
5. Don't want to register
****
Enter your choice: 2
Enter your E-mail id: loay.rashid@students.iiit.ac.in
**********REGISTRATION MENU**********
1. Enter Name:
2. Enter Email-id:
3. Initial Amoount in Wallet:
4. REGISTER
5. Don't want to register
****
Enter your choice: 3
Enter initial value on wallet: 50
**********REGISTRATION MENU**********
1. Enter Name:
2. Enter Email-id:
3. Initial Amoount in Wallet:
4. REGISTER
5. Don't want to register
****
Enter your choice: 4
REGISTRATION COMPLETE
****
CARD DETECTED

Welcome to Parking Lot : Loay
Your in time is : 2020-03-16 21:53:11

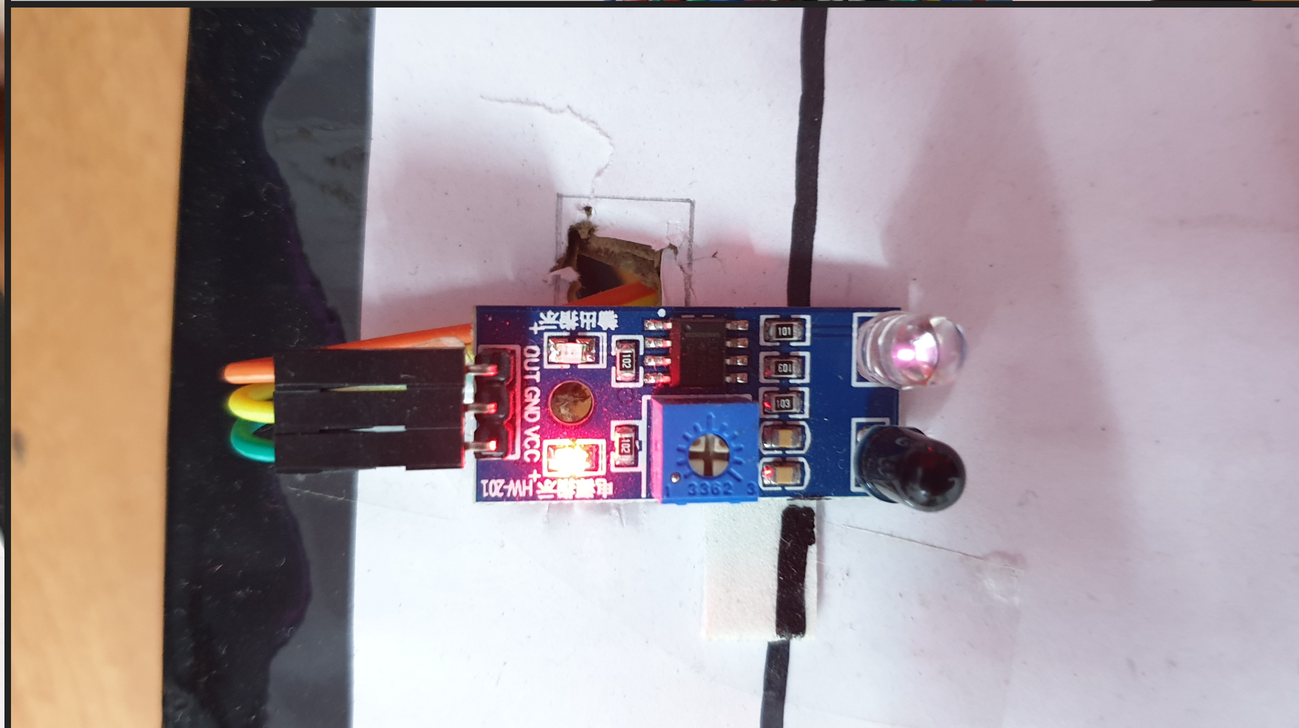
CARD DETECTED

Thank You for using our service.
Total Time Parked: 0:01:28
Please check your mail for total bill

LED


RFID-RC522
SDA SCK MOSI MISO IRQ GND RST 3.3V

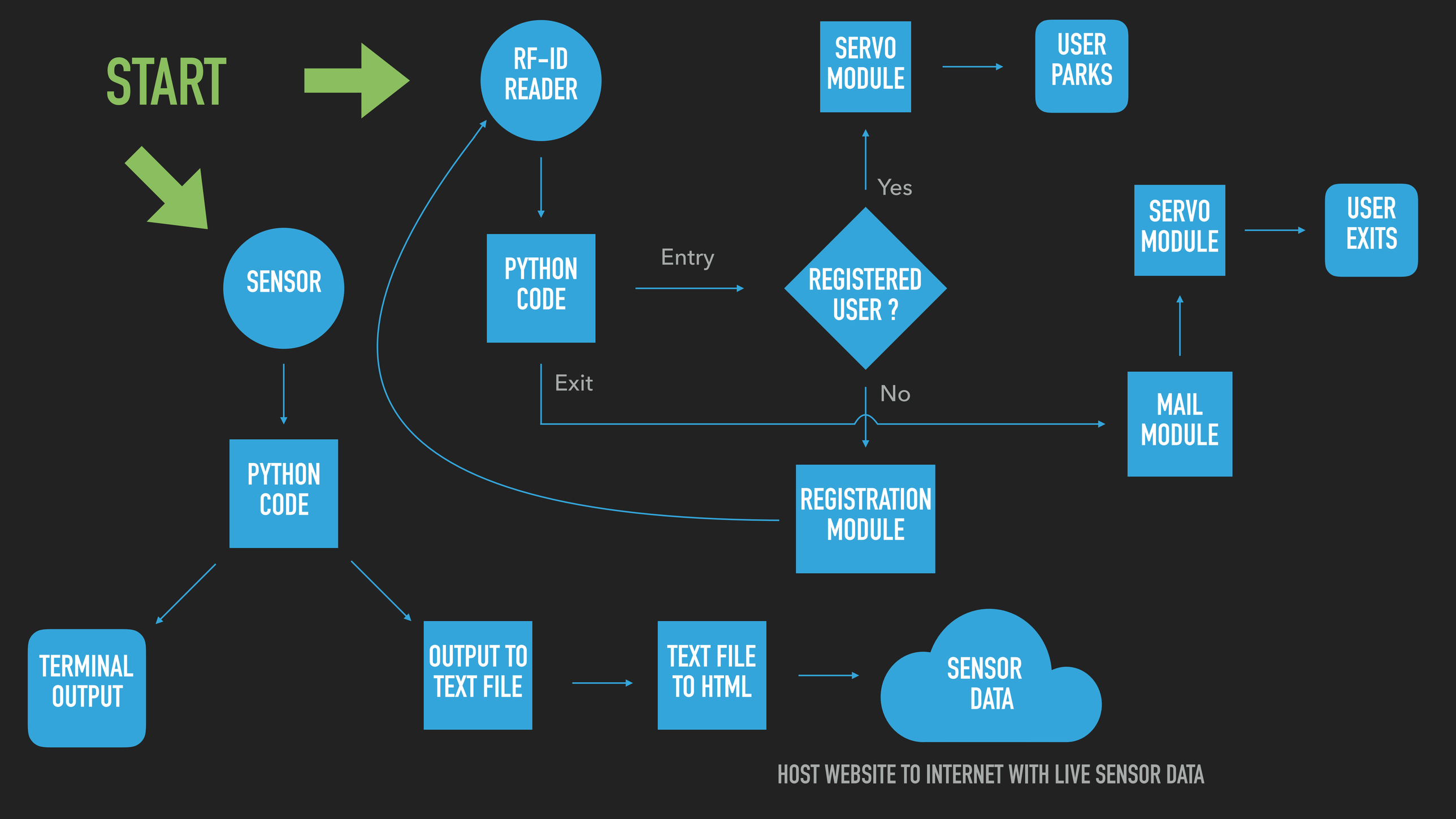

Microsoft
GCT
GCT
HDMI
ETHERNET
Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015


OUT GND VCC
HW-201

PROJECT VIDEO

Click here to watch Video

FLOW CHART


START
RF-ID READER
SERVO MODULE
USER PARKS
SENSOR
PYTHON CODE
Entry
REGISTERED USER ?
Yes
SERVO MODULE
USER EXITS
Exit
No
MAIL MODULE
PYTHON CODE
REGISTRATION MODULE
TERMINAL OUTPUT
OUTPUT TO TEXT FILE
TEXT FILE TO HTML
SENSOR DATA
HOST WEBSITE TO INTERNET WITH LIVE SENSOR DATA

CODE SNIPPETS

```
268             else:
269
270                 if entry[(uid[0] + uid[1] + uid[2] + uid[3])] == 0 :
271                     print 40 * "-"
272                     print "Welcome to Parking Lot : "+name[(uid[0] + uid[1] + uid[2] + uid[3])]
273                     entry[(uid[0] + uid[1] + uid[2] + uid[3])] = 1
274                     time_e_in = datetime.datetime.now().replace(microsecond = 0 )
275                     print "Your in time is : ",time_e_in
276                     print 40 * "-"
277                     servo()
278                 else:
279                     email_e = email[(uid[0] + uid[1] + uid[2] + uid[3])]
280                     time_e_out = datetime.datetime.now().replace(microsecond = 0 )
281                     time_e = time_e_out - time_e_in
282                     print 40 * "-"
283                     print "Thank You for using our service.\nTotal Time Parked: ",time_e,"\nPlease check your mail for total bill"
284                     print 40 * "-"
285                     entry[(uid[0] + uid[1] + uid[2] + uid[3])] = 0
286                     name_m = name[(uid[0] + uid[1] + uid[2] + uid[3])]
287                     mail(name_m, time_e_in, time_e_out, time_e, email_e,(uid[0] + uid[1] + uid[2] + uid[3]))
288                     servo()
```

RF-id: Accessing name, email-id and wallet amount corresponding to RF-id number

```
235
236         else :
237             if  card [ (uid[0] + uid[1] + uid [2] + uid[3]) ] == 0 :
238
239                 print "NOT A REGISTERED USER - Pls complete the registration process below\n"
240                 choice = 0
241                 while ( choice != 5):
242                     print "***************REGISTRATION MENU***************"
243                     print "1. Enter Name: "
244                     print "2. Enter Email-id: "
245                     print "3. Initial Amoount in Wallet:  "
246                     print "4. REGISTER"
247                     print "\n5. Don't want to register"
248                     print "***********************************************"
249
250                     choice = input("Enter your choice: ")
251
252                     if choice == 1 :
253                         enter_name = raw_input("Enter your name: ")
254                         name [ (uid[0] + uid[1] + uid[2] +uid[3]) ] = enter_name
255
256                     if choice == 2 :
257                         enter_email = raw_input("Enter your E-mail id: ")
258                         email[ (uid[0] + uid[1] + uid [2] + uid[3] ) ] = enter_email
259
260                     if choice == 3:
261                         enter_wallet = input("Enter initial value on wallet: ")
262                         wallet[ (uid[0] + uid[1] + uid[2] + uid[3]) ] = enter_wallet
263                     if choice == 4:
264                         card [ (uid[0] + uid[1] + uid[2] + uid[3]) ] = 1
265                         print "REGISTRATION COMPLETE"
266                         print 40 * "*"
267                         break
```

New User Registration: A menu driven program to enter values of name, email id and wallet amount to corresponding RF-id cards

```
50 def mail(naame, time_in, time_out, time_total, eemail,loc):
51
52                 c = str(time_total)
53                 cost_time = ( (3600*int(c[0])) + (((int(c[2])*10) + int(c[3]))*60) + ((int(c[5])*10) + int(c[6])) )
54                 cost = cost_time * 0.01
55
56                 wallet[loc] = wallet[loc] - cost
57
58                 #print "Cost is = ",cost
59
60                 msg = MIMEMultipart()
61                 msg['From'] = 'smartparkingiiit@outlook.com'
62                 msg['To'] = eemail
63                 msg['Subject'] = 'Smart Parking Bill'
64                 bill = "Dear "+naame+",\n\nThis is your bill for the parking services.\n\nIn-time : "+str(time_in)+".\nOut-time : "+str(time_out)+".\n\nBased on your parked t
   ime and charges price as 36 Rupees/hour,your total bill is: Rs."+str(cost)+"\n\nRemaining balance in Wallet: Rs."+str(wallet[loc])+"\n\nThank You."
65                 message = bill
66                 msg.attach(MIMEText(message))
67
68                 server = smtplib.SMTP('smtp.outlook.com',587) #Connects to SMTP sever at timeout 587sec
69                 server.ehlo()
70                 server.starttls() #Puts SMTP in TLS ( transport layer security) mode for encryption
71                 server.ehlo()
72                 server.login('smartparkingiiit@outlook.com', 'Utkarsh@2001') #Username and Password mentioned
73                 server.sendmail("smartparkingiiit@outlook.com",eemail,msg.as_string())
74                 server.quit()
75
```

Mail Module: Using the SMTP function, we will send out mail to users upon exiting, which will contain in-time, out-time and cost corresponding to total time parked. It will also contain the remain balance in the wallet

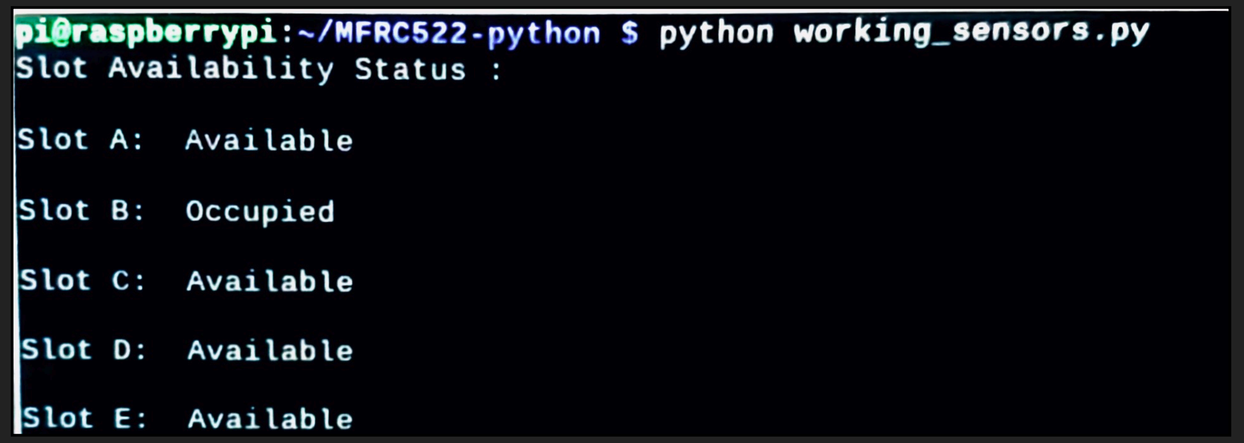
```
25 import RPi.GPIO as GPIO
26 import MFRC522
27 import signal
28 import time
29 import datetime
30 import smtplib
31 continue_reading = True
32
33 from email.MIMEMultipart import MIMEMultipart
34 from email.MIMEText import MIMEText
35
36 def servo():
37     GPIO.setmode(GPIO.BOARD)
38     GPIO.setup(11,GPIO.OUT)
39     servo1 = GPIO.PWM(11,50)
40     servo1.start(0)
41     servo1.ChangeDutyCycle(2)
42     time.sleep(3)
43     servo1.ChangeDutyCycle(7)
44     time.sleep(3)
45     servo1.ChangeDutyCycle(2)
46     servo1.ChangeDutyCycle(0)
47     servo1.stop()
48     GPIO.cleanup()
```

Servo Motor: This module controls the gate at the entrance of the parking lot. This gate opens only when a registered user scans their card.

```
1 import RPi.GPIO as GPIO
2 import time
3 import sys
4
5
6 #CURSOR_UP_ONE = '\x1b[1A'
7 #ERASE_LINE = '\x1b[2k'
8
9 sensor_a = 8
10 sensor_b = 10
11 sensor_c = 12
12 sensor_d = 16
13 sensor_e = 18
14
15 GPIO.setmode(GPIO.BOARD)
16 GPIO.setup(sensor_a,GPIO.IN)
17 GPIO.setup(sensor_b,GPIO.IN)
18 GPIO.setup(sensor_c,GPIO.IN)
19 GPIO.setup(sensor_d,GPIO.IN)
20 GPIO.setup(sensor_e,GPIO.IN)
21
22 print "Slot Availability Status :\n"
23 #print "IR Sensor Ready....."
24 #print " "
25
26 # 0 for occupied
```

Initialize Proximity Sensor: Setting up the proximity so that the output from the sensor (high or low) can be read by the python program

```
73      print"Slot E: ",
74      file_txt += "Slot E: "
75      if GPIO.input(sensor_e) == 1:
76          print "Available\n"
77          file_txt += "Available\n"
78
79      if GPIO.input(sensor_e) == 0:
80          print "Occupied\n"
81          file_txt += "Occupied\n"
82 #*****************************************************************************************#
83      f.write(file_txt)
84      f.close()
85      time.sleep(0.5)
86
87      for i in range(10):
88          sys.stdout.write("\033[F")
89        sys.stdout.write("\033[K")
90
91 except KeyboardInterrupt:
```



Working of Proximity Sensor: The current status of the sensors is written on to a file continuously with the gap of 0.5 seconds. The output in the terminal seems dynamic as the output is cleared an reprinted every 0.5 second (notice the same in the video)

```
<!DOCTYPE html>
<html>
        <head>
                <meta http-equiv = "refresh" content = "6.2" >
        </head>
        <body>
                <h1> Slot Availability Status </h1>
                <object width = "300" height = "300" type = "text/plain" data = "output.txt" border = "0" >
                </object>
        </body>

</html>
```

HTML File: This html file takes the data from output.txt (contents of this file is written down by the python code running the sensors. There is an addition command to enable auto refresh the webpage. This html file is hosted on the internet using a third party host : ngrok

SCALABILITY

- Started with miniature implementation
- Scalable in parking lots with numbered/labeled slots
- Larger scale implementation will require additional infrastructure (Wireless Proximity Sensors)

THANK YOU