



AMERICAN UNIVERSITY OF SHARJAH

College of Engineering

Fall 2023

COE 410 Project Report

Smart Toll Gate System

Name	ID
Warda Ul Hasan	g00090299
Hanaa Khalil	g00083777
Batool Azzam	g00092174

Instructors Name: Dr. Abdurahman Al-Ali and Mr. Ahmad Al Nabulsi

Submission date: 11/10/23

System Diagrams

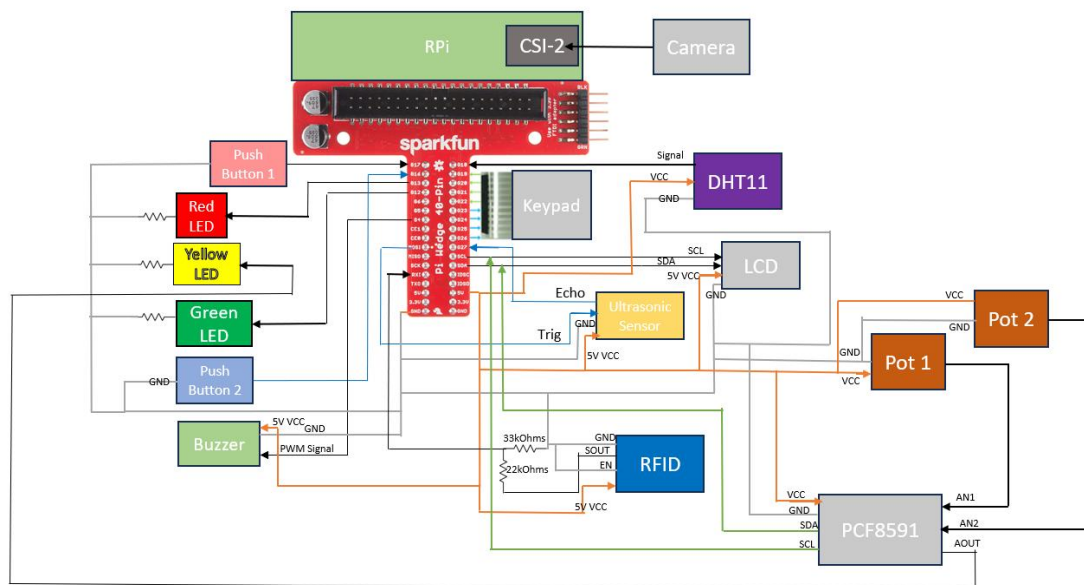


Figure 1 Hardware Diagram

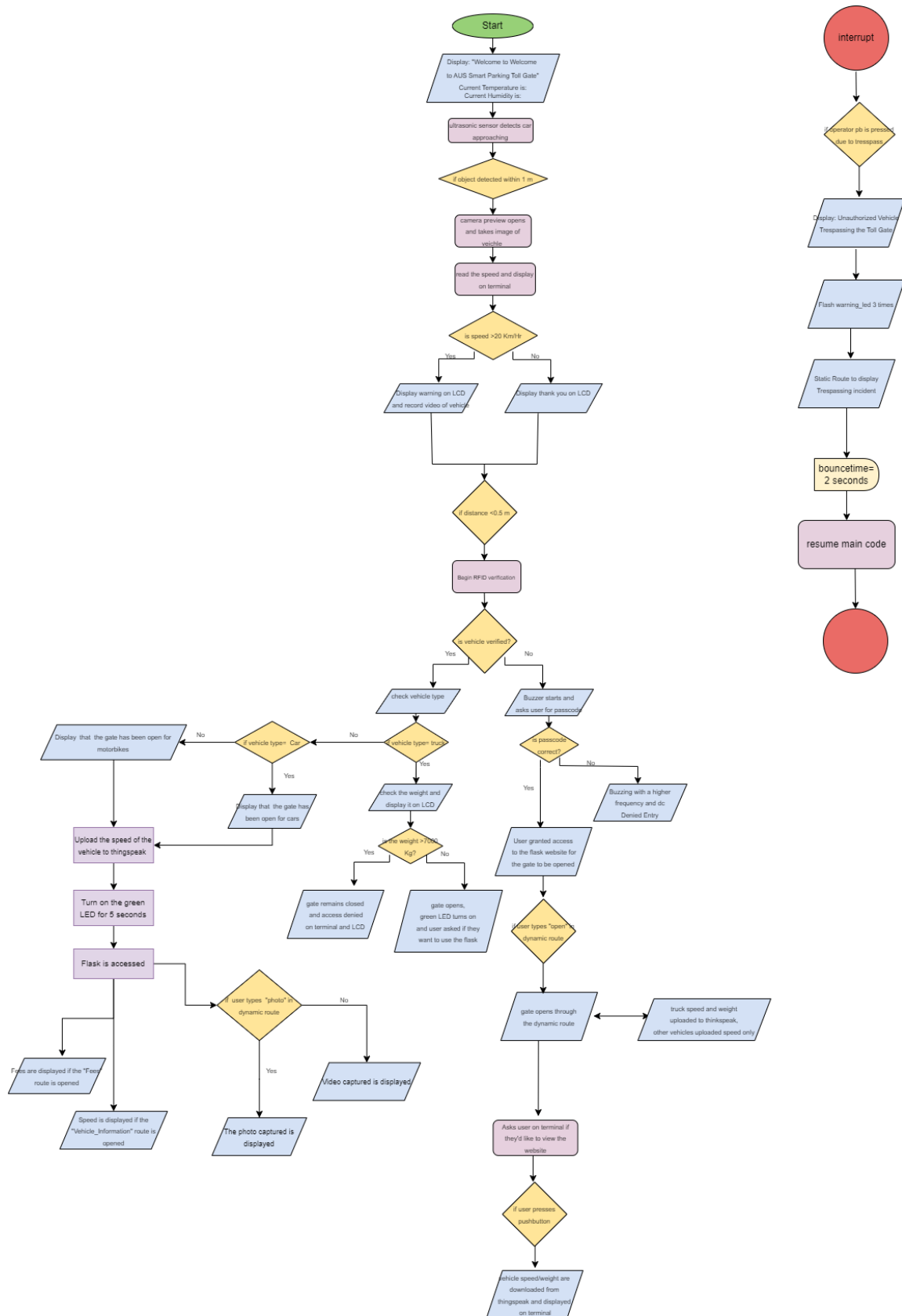


Figure 2 Software Flowchart

Program Code

```
#import GPIO module to enable input and output ports
import RPi.GPIO as GPIO

#import time library to implement sleep functions
import time

#import temperature and humidity sensor, DT11 library
import DHT11

#import the PCF8591 module to perform Analog to Digital Conversion
import PCF8591 as ADC

#import the urllib.request library
import urllib.request

#import the flask library to set up flask server
from flask import Flask

#import send_file(photo_path, imageType) method from flask
from flask import send_file

#import datetime module to add timestamp
from datetime import datetime

#import picamera library to access PiCamera functions
from picamera import PiCamera

#to ignore warnings
GPIO.setwarnings(False)

#import LCD1602 library to display on the LCD
import LCD1602 as LCD

#initialize the LCD
LCD.init(0x27, 1)

#import keypad library to use keypad function
from keypadfunction import keypad

#import serial port library
import serial
SERIAL_PORT = '/dev/ttyS0'
```

```

#setting the serial port according to the Parallax reader's datasheet
ser = serial.Serial(baudrate = 2400, bytesize = serial.EIGHTBITS,
parity = serial.PARITY_NONE, port = SERIAL_PORT, stopbits =
serial.STOPBITS_ONE, timeout = 1)

#using PiCamera() class an object called Mycamera is created
Mycamera=PiCamera()

#Write Key, Channel ID values for ThingSpeak channel
WRITE_KEY = "B840552VU4JUOV8K"
Ch_ID = "2328945"

#ThingSpeak field number
Speed_Field = 4
Weight_Field = 2

#number of values to read
No_of_readings = 10
#to read 5th value
element_number = 5
#empty lists to store actual values
speed_values = []
weight_values = []

#function to validate RFID tag - if the tag is valid the code will be
returned else False
def validate_rfid(code):
    s = code.decode("ascii")
    if (len(s) == 12) and (s[0] == "\n") and (s[11] == "\r"):
        return s[1:11]
    else:
        return False

#setup I2C of PCF8591
ADC.setup(0x48)
#setting the BCM mode
GPIO.setmode(GPIO.BCM)
#setting the buzzer pin as an output
buzzer = 4
GPIO.setup(buzzer,GPIO.OUT)

# setting buzzer to produce a PWM signal with a frequency of 200 Hz
buzz = GPIO.PWM(buzzer,200)

```

```

#setting a download push button as an input

download_pb = 16
GPIO.setup(download_pb,GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

#setting an operator push button as an input
operator_pb = 17
GPIO.setup(operator_pb,GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

#red LED - setting a warning LED an an output
warning_led = 13
GPIO.setup(warning_led,GPIO.OUT)

#green LED - setting a entry gate LED an an output
entrygate_led = 12
GPIO.setup(entrygate_led,GPIO.OUT)

#yellow LED - setting a LED to varies intensity with varying weight as
an output
weight_led = 5
GPIO.setup(weight_led,GPIO.OUT)

#setting the port for the echo pin of the ultrasonic sensor as an
input
echo = 27
GPIO.setup(echo,GPIO.IN)

#setting the port for the trig pin of the ultrasonic sensor as an
output
trig = 10
GPIO.setup(trig,GPIO.OUT)

# function to read humidity and temperature

def readTempHum():
    result=""
    while (not result):
        result = DHT11.readDht11(18)

```

```

        if result:
            humidity, temperature = result
        return result

# function to measure the distance in cm using the ultrasonic sensor
def distance():
    # initially making the trig signal low
    GPIO.output(trig, GPIO.LOW)
    time.sleep(0.000002)
    # setting the trig pin high for 10 us
    GPIO.output(trig, 1)
    time.sleep(0.000001)
    # stop the trig signal from generating
    GPIO.output(trig, 0)
    # reading the echo pin signal and calculate the distance in cm
    while GPIO.input(echo) == 0:
        a=0
    # capturing time 1
    time1 = time.time()
    while GPIO.input(echo) == 1:
        a=0
    # capturing time 2
    time2 = time.time()
    # duration of the echo signal being received
    duration = time2 - time1
    # returning the distance in cm
    return duration*1000000/58

# function to read the speed of the vehicle from a potentiometer
def readspeed():
    # read ADC units from channel 1
    ADC_units = ADC.read(1)
    # convert the units into volts
    ADC_volts=(ADC_units*3.3)/256
    # calculate the speed using the ADC units read from the potentiometer
    # attached to channel 0
    speed = (ADC_units/255)*200
    vehicle_speed = int(speed)
    return vehicle_speed

# function to read the weight of a truck
def readWeight():
    # read ADC units from channel 2

```

```

    ADC_units = ADC.read(2)
# using the DAC, the digital units read from channel 2 is converted to
analog out in AOUT
# in our scenario we are reading weight as the digital data and
converting it as analog output which changes the light intensity of
the LED connected to AOUT pin
    ADC.write(ADC_units)
# convert the units into volts
    ADC_volts=(ADC_units*3.3)/256
# calculate the speed using the ADC units read from the potentiometer
attached to channel 0
    weight = (ADC_units/255)*8000
    vehicle_weight = int(weight)
    return vehicle_weight

#setting up the default route path: creating the index page with a
welcome message to the toll gate
myapp = Flask(__name__)

@myapp.route('/')

def index ():
    return 'Welcome to the Smart Toll Gate system'

#Route path '/Fees' - Static Route
@myapp.route('/Fees')
#function to display the fees for each vehicle type in the client
browser terminal
def DisplayFees():
    fee = "Car fees = 15 Dhs | Motorbike fees 10 Dhs | Truck fees 20
Dhs"
    return fee

#Route path '/Vehicle_Information' - Static Route
@myapp.route('/Vehicle_Information')
#function to display vehicle's speed in the client browser terminal
along with the date and time in ISO format
def vehicle_info():
    timestamp = datetime.now().isoformat()
#if the speed of the vehicle is greater than or equal to 20, a warning
message prints on the static route
#if the speed is less than 20, just the speed is printed on the screen
    speed = readspeed()

```



```

    if(speed>20):
        buf = "Speed  = {}km/Hr. Slow Down!!".format(speed)
    else:
        buf = "Speed  = {}km/Hr".format(speed)
    return buf

#ISR function: in the case of an unauthorized vehicle trespassing the
toll gate, the operator presses the operator_pb and an event is
detected
trespass = 0
def warning(self):
    #warning message printed on the computer screen
    print("Unauthorized Vehicle Trespassing the Toll Gate\n")
    global trespass
    trespass =1
    #turning the red LED ON and OFF 4 times
    for x in range(4):
        GPIO.output(warning_led, GPIO.HIGH)
        time.sleep(1)
        GPIO.output(warning_led, GPIO.LOW)
        time.sleep(1)
    if __name__ == '__main__':      # running flask to display when the
trespassing occurs
        myapp.run(host='0.0.0.0',port= 5020)

#interrupt callback definition
GPIO.add_event_detect(operator_pb, GPIO.FALLING, callback = warning,
bounce time = 2000)

#Route path '/Trespassing' - Static Route
#this static route warns if there is any vehicle that had gained
unauthorized access
#this Static route is called through an ISR
@myapp.route('/Trespassing')
def tresspass_detect():
    if(trespass == 1):
        timestamp = datetime.now().isoformat()
        msg = "Warning! Unauthorised access detected at
{}".format(timestamp)
    else:
        msg = "No unauthorised access detected"

    return msg

```

```

#turning the entry LED OFF
GPIO.output(entrygate_led,GPIO.LOW)

# Route path '/Open_Gate/open' or '/Open_Gate/close'- Dynamic Route to
open or close the gate
@myapp.route('/Open_Gate/<x>')
def Open_Gate(mode):
    if(mode == "open"):
        GPIO.output(entrygate_led,GPIO.HIGH)
        return 'The Gate is Open'
    elif(mode == "close"):
        GPIO.output(entrygate_led,GPIO.LOW)
        return 'The Gate is closed'
    else:
        return 'Enter open or close'

# dynamic route to capture an image or record a video
@myapp.route('/Vehicle_Capture/<capture>')
def vehicle_capture(capture):
    # if the variable in the dynamic route is photo, an image is sent to
    the client browser
    if (capture == "photo"):

response=send_file('/home/pi/Desktop/Vehiclepic.jpg',mimetype="image/j
peg")
    # if the variable in the dynamic route is video, a video is sent to
    the client browser
    elif(capture == "video"):
        response =
send_file('/home/pi/Desktop/video.h264',mimetype="video/h264")
    else:
        response="Enter photo or video"

    return response

# storing the RFID tag codes values attached on a truck, car, and a
motorbike
truckcode = '5400652FA3'
carcode = '46003BA4AD'
motorbikecode = '010FB3C21B'

x = 0    #dummy variable

```

```

# function to click a picture with annotated date and time in ISO
format
def takePicture():
#setting the sharpness of the camera to 30
    Mycamera.sharpness = 30
#setting the brightness of the camera to 50
    Mycamera.brightness= 50
#setting the resolution of the camera to 640 x 480
    Mycamera.resolution = (640,480)
# getting the current date and time in ISO format
    timestamp = datetime.now().isoformat()
# setting the photo path where photo should be saved after capturing
    photo_path="/home/pi/Desktop/Vehiclepic.jpg"
# starts the camera preview on screen
    Mycamera.start_preview()
# insert the date and time in ISO format
    Mycamera.annotate_text="Picture taken at time %s" %timestamp
    time.sleep(4)
# capture the image
    Mycamera.capture(photo_path)
# stop the camera preview
    Mycamera.stop_preview()

# function to record a 5 second video with annotated date and time in
ISO format
def takeVideo():
    Mycamera.sharpness = 30
    Mycamera.brightness= 50
    Mycamera.resolution = (640,480)
    timestamp = datetime.now().isoformat()
    Mycamera.start_preview()
    Mycamera.annotate_text="Video taken at time %s" %timestamp
    time.sleep(5)
    Mycamera.start_recording('/home/pi/Desktop/Vehiclevideo.h264')
    time.sleep(5)
    Mycamera.stop_recording()
    Mycamera.stop_preview()

# loops forever - main system
while True:

    # reading the current Temperature and Humidity
    humidity, temperature = readTempHum()

```

```

    # prints on the computer screen a welcome message to Smart Parking
Toll Gate
    # also prints current temperature and humidity
    if(x==0):
        print("Welcome to AUS Smart Parking Toll Gate\n")
        print ("Current Temperature is:{}C".format(temperature))
        print ("Current Humidity is:{}%".format(humidity))
        print("\n")
        x+=1

# measuring the distance of any approaching vehicles using the
ultrasonic sensor
    dist = distance()/100

    # if the distance is less than 1 meter, prints on the computer screen
that vehicle is approaching less than 1 meter away
    if(dist < 1):
        print("Vehicle approaching is less than 1 meter away\n")
        takePicture()
        # reading the speed of the vehicle
        vehicle_speed = readspeed()
        # print on screen the approaching vehicle's speed
        print("Vehicle speed = {:0.2f} km/hr".format(vehicle_speed))
        print("\n")

        # if the vehicle's speed is greater than 20 km/hr, a warning
message prints on the computer screen and the LCD display to slow down
        if( vehicle_speed > 20):
            print("Driver's speed is greater than 20 km/hr\n")
            LCD.write(0,0, "Slow Down      ")
            LCD.write(0,1, "Drive below 20 ")
            #records a video for 2 seconds is the vehicle approaching
has a speed greater than 20
            takeVideo()
            time.sleep(2)

        # if the vehicle's speed is less than or equal to 20 km/hr, a
thank you message prints on the LCD screen
        else:
            LCD.write(0,0, "Thank you for  ")
            LCD.write(0,1, "driving safely  ")

```

```

        # when the distance of the approaching vehicle is less than
        0.5 meters the validation of the RFID tag embedded on the bodies of
        the vehicle takes place
        if(dist<0.5):
# prints on the screen vehicle is less than 0.5 meters away
        print("Vehicle approaching is less than 0.5 meters
away\n")
# prints on the screen and displays on the LCD that vehicle is being
validated
        print("Validating Vehicle\n")
        LCD.write(0,0, "Validating      ")
        LCD.write(0,1, "Vehicle      ")
        time.sleep(3)
        ser.flushInput()
        ser.flushOutput()
# read 12 bytes from the serial port
        data=ser.read(12)
        # if a code is returned by validate_rfid, then the vehicle
        is validated
        code = validate_rfid(data)
        if code:
# if the code is not false and has a string returned the message for
vehicle validation is printed on the screen and display on the LCD
screen
        print("Vehicle Validated\n")
        LCD.write(0,0, "Vehicle      ")
        LCD.write(0,1, "Validated      ")

# if the code returned is the same value as the code of a car, the
vehicle is recognized as a car, provides direct entry for cars
displayed on the computer screen and LCD display and the entry LED
lights turn ON
        if(code == carcode):
            print("The gate has been opened!\n")
# the car's speed is uploaded to ThingSpeak
            x=
urllib.request.urlopen("https://api.thingspeak.com/update?api_key={}&f
ield4={}".format(WRITE_KEY,vehicle_speed))
            LCD.write(0,0, "Direct Entry      ")
            LCD.write(0,1, "for Cars      ")
            GPIO.output(entrygate_led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(entrygate_led,GPIO.LOW)
            flask = input("Would you like to access flask?
(Enter 'Yes' or 'No')")

```

```

        if (flsk == "Yes"):
            if __name__ == '__main__':
                myapp.run(host='0.0.0.0',port= 5020)

        # if the code returned is the same value as the code
of a motorbike, the vehicle is recognized as a motorbike, provides
direct entry for motorbikes displayed on the computer screen and LCD
display and the entry LED lights turn ON

        elif(code == motorbikecode):
            print("The gate has been opened!\n")
# the motorbike's speed is uploaded to ThingSpeak
            x=
urllib.request.urlopen("https://api.thingspeak.com/update?api_key={}&f
ield4={}".format(WRITE_KEY,vehicle_speed))
            LCD.write(0,0, "Direct Entry  ")
            LCD.write(0,1, "for Motorbikes ")
            GPIO.output(entrygate_led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(entrygate_led,GPIO.LOW)
            flsk = input("Would you like to access flask?
(Enter 'Yes' or 'No')")
            if (flsk == "Yes"):
                if __name__ == '__main__':
                    myapp.run(host='0.0.0.0',port= 5020)

        # if a truck is recognized from the code returned, the
truck's weight needs to be measured first

        elif(code == truckcode):
            print("Please measure the weight of the TRUCK")
            # reading the car's weight
            weight = readWeight()
            # updating the ThingSpeak channel with the weight
and speed value of the truck
            x=
urllib.request.urlopen("https://api.thingspeak.com/update?api_key={}&f
ield2={}&field4={}".format(WRITE_KEY,weight,vehicle_speed))
            print("Truck weight: ",weight, " kg")
            weight1 = str(weight)
            weight1 = weight1 + " kg"
            LCD.write(0,0, " ")
            LCD.write(0,1, " ")
            LCD.write(0,0, "Truck's weight:")
            LCD.write(0,1, weight1)

```

```
        # if the weight of the truck is greater than 7000,
the truck is denied to use the gate
```

```
        if(weight > 7000):
            print("Denied Access, the weight is greater
than 7000 kg\n")
```

```
            LCD.write(0,0, "
            LCD.write(0,1, "
            LCD.write(0,0, "Access Denied: ")
            LCD.write(0,1, "Weight>7000 kg")
```

```
        # else trucks with less than 7000 kg weight, the
truck will be granted access to the gate
```

```
        else:
            print("The gate has been opened!\n")
            GPIO.output(entrygate_led,GPIO.HIGH)
            time.sleep(5)
            GPIO.output(entrygate_led,GPIO.LOW)
            flask = input("Would you like to access flask?
(Enter 'Yes' or 'No')")
```

```
            if (flask == "Yes"):
                if __name__ == '__main__':
                    myapp.run(host='0.0.0.0',port= 5020)
```

```
        #if the vehicles were not able to be validated by the RFID
reader, a buzzer will start and the LCD and computer screen will
display RFID is not valid
```

```
        elif(not(code)):
            print("RFID not valid!\n")
            buzz.start(30)
            LCD.write(0,0, "RFID not valid ")
            LCD.write(0,1, "
            time.sleep(4)
            # then the driver will be asked to enter their
passcode to gain access to the website to open gate using a keypad
            LCD.write(0,0, "Enter Passcode ")
            LCD.write(0,1, "to open website")
```

```
        # the values from the keypad is read
        key1 = keypad()
        time.sleep(1)
```

```

        key2 = keypad()
        time.sleep(1)
        key3 = keypad()
        time.sleep(1)
        key4 = keypad()
        time.sleep(1)
        # concatenating the keypad values entered by the user as a
string
        key_final = str(key1) + str(key2) + str(key3)+
str(key4)

        # if the passcode entered is first four characters of
any of the vehicle's RFID tag code, then the driver will get access to
the website
        if(key_final == truckcode[0:4] or key_final ==
carcode[0:4] or key_final == motorbikecode[0:4]):
            print("Successful! Please use flask to open
gate\n")
            buzz.ChangeDutyCycle(0)
            LCD.write(0, 0, "Use flask to      ")
            LCD.write(0,1, "Open Gate      ")
            if __name__ == '__main__':
                myapp.run(host='0.0.0.0',port= 5020)

        # if the passcode is wrong the driver is unauthorized
and is denied to gain access to the website, and the buzzer start
alarming with a greater frequency
        else:
            print("Access Denied")
            LCD.write(0, 0, "Passcode is  ")
            LCD.write(0,1, "wrong      ")
            buzz.ChangeFrequency(600)
            buzz.ChangeDutyCycle(30)
            time.sleep(5)
            buzz.ChangeDutyCycle(0)

# if the download pushbutton is pressed, the speed of the vehicles is
downloaded from ThingSpeak
        if(GPIO.input(download_pb) == 1):
            print("Downloading Speed of Vehicles: ")
# reading 10 vehicles speeds
            s=
urllib.request.urlopen("https://api.thingspeak.com/channels/{}/fields/
{}.csv?results={}".format(Ch_ID,Speed_Field,No_of_readings))
# decoding the read data to ascii

```



```

        data1 =s.read().decode('ascii')
        print(data1)
# convert the ascii data to comma separated string
        data1 = ",".join(data1.split("\n"))

# creating a list of speed values
        for i in range(5, No_of_readings*3+3, 3):
            speed_values.append(data1.split(",")[i])

        speedElement= float(speed_values[element_number])
        vehicle_speed = readspeed()
# if the current vehicle speed is greater than the speed of the 5th
vehicle a warning message is displayed
        if(vehicle_speed > speedElement):
            print("Warning your speed is greater than the fifth
element")

# downloading the weight of the vehicles, following the same steps as
the download of the speed of the vehicles
        print("Downloading Weight of Vehicles: ")
        w=
urllib.request.urlopen("https://api.thingspeak.com/channels/{}/fields/
{}.csv?results={}".format(Ch_ID,Weight_Field,No_of_readings))
        data2 =w.read().decode('ascii')
        print(data2)

        data2 = ",".join(data2.split("\n"))

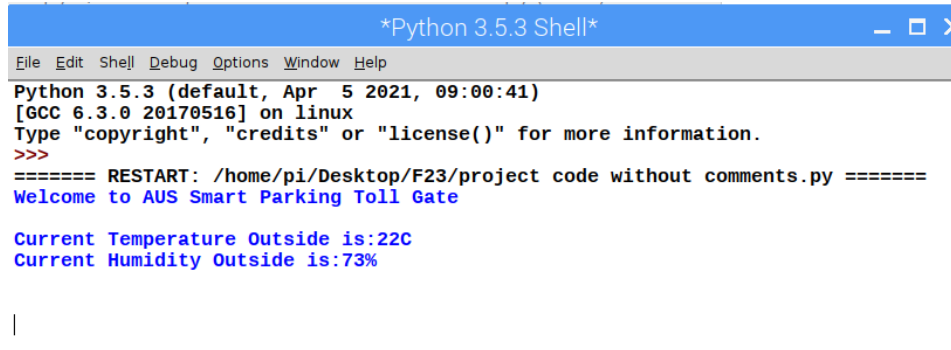
        for j in range(5, No_of_readings*3+3, 3):
            weight_values.append(data1.split(",")[j])

        weightElement= float(weight_values[element_number])
        weight = readWeight()
        if(weight > weightElement):
            print("Warning your weight is greater than the fifth
element")

```

Execution Scenarios

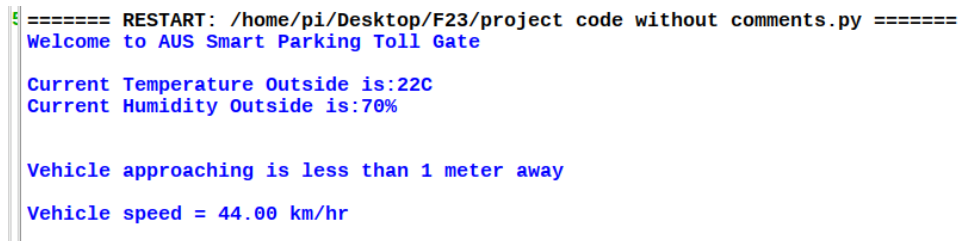
The system starts by displaying on the terminal a welcome message, along with the temperature and humidity recorded from the DHT11 sensor. Afterwards, the Ultrasonic sensor detects any approaching vehicles. If a vehicle is approaching in less than 1 meter the camera will open and capture, along with reading the speed that is acquired from the potentiometer and displaying it on the terminal.



```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature Outside is:22C
Current Humidity Outside is:73%
```

Figure 3. System Start Welcome Message



```
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature Outside is:22C
Current Humidity Outside is:70%

Vehicle approaching is less than 1 meter away
Vehicle speed = 44.00 km/hr
```

Figure 4. Distance and Speed

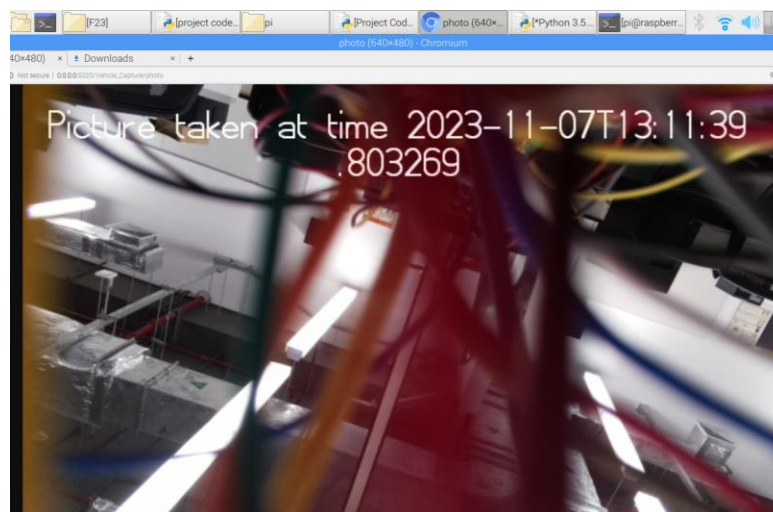


Figure 5. Camera Preview

If the speed of the vehicle is > 20 Km/Hr, a warning message will be displayed on the LCD and the camera will open to record a video of the vehicle. If it is under 20 Km/Hr, a thank you message is displayed on the LCD.

```
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====
Welcome to AUS Smart Parking Toll Gate
Current Temperature Outside is:23C
Current Humidity Outside is:70%

Vehicle approaching is less than 1 meter away
Vehicle speed = 153.00 km/hr
```

Figure 6. Speed > 20 Km/Hr



Figure 7. LCD “Slow Down” Display

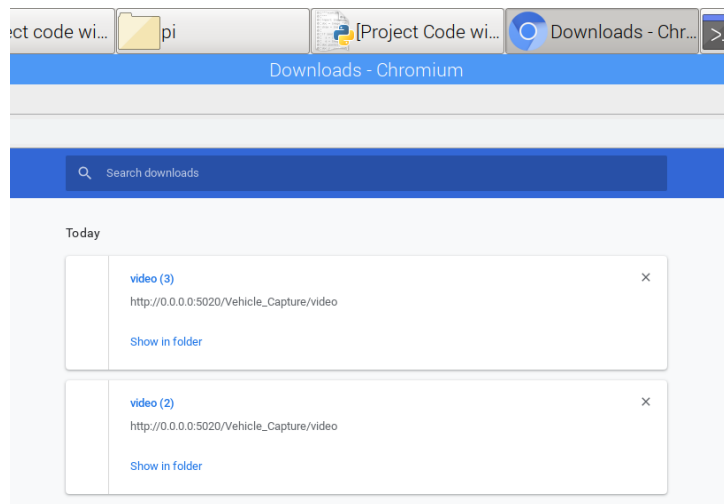


Figure 8. Recorded Video Download

When the ultrasonic sensor detects a vehicle approaching in less than 0.5 m, the process of the RFID validation begins. The vehicle is supposed to validate itself by scanning the tag assigned to them. This process is displayed on the LCD and on the terminal as well, as can be seen below. If the vehicle is validated through the RFID tag the gate will open without the need for any more actions and the green LED turns on. Each vehicle type is assigned to an RFID tag. There are three types in this system: Cars, Trucks and Motorcycles. Verified cars and motorcycles are given instant entry.

```
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====  
Welcome to AUS Smart Parking Toll Gate  
  
Current Temperature Outside is:23C  
Current Humidity Outside is:70%  
  
Vehicle approaching is less than 1 meter away  
Vehicle speed = 153.00 km/hr  
  
Vehicle approaching is less than 0.5 meters away  
Validating Vehicle  
Vehicle Validated  
The gate has been opened!
```

Figure 9. RFID Validation Terminal



Figure 10. RFID Validation LCD

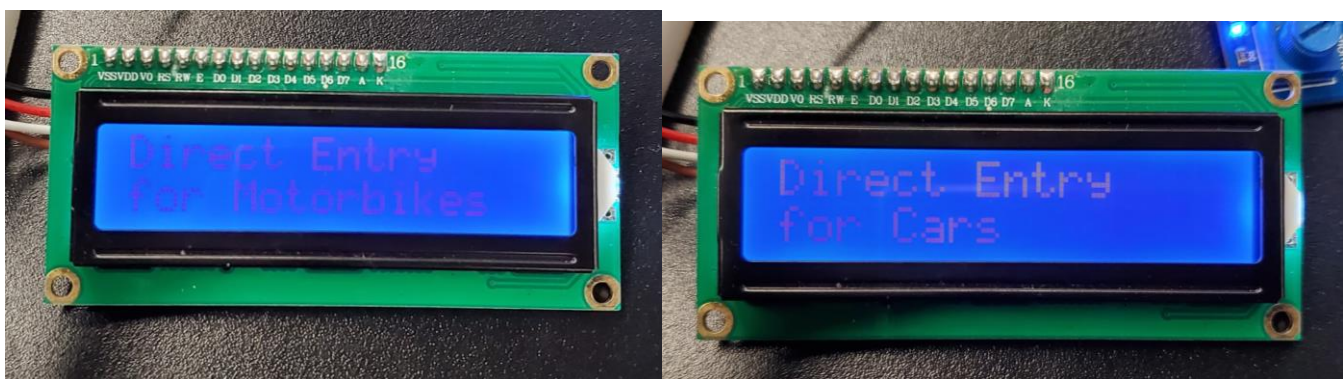


Figure 11. Direct Entry LCD

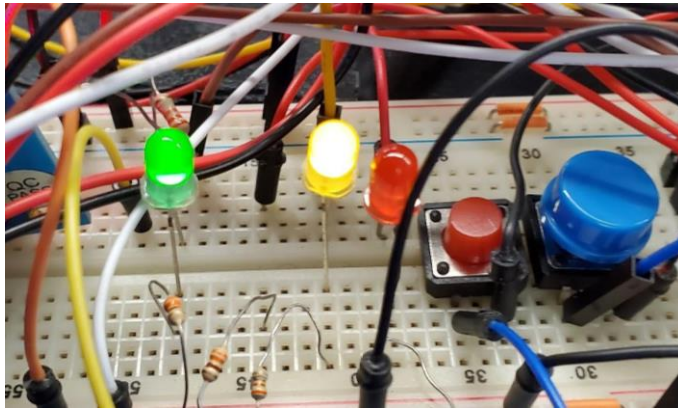


Figure 12. Green LED For Open Gate

For trucks, the weight of the truck is first measured. If it is above 7000Kg, it is denied entry as it exceeds the maximum weight set for the system. The weight is displayed on the LCD and if it is denied entry a message is also displayed. This will additionally be displayed on the terminal. The weight of the truck is demonstrated using an LED. The greater the weight, the brighter the yellow LED.



Figure 13. Truck Weight on LCD



Figure 14. Access Denied on LCD


```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/Project Code with comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature is:23C
Current Humidity is:69%

Vehicle approaching is less than 1 meter away
Vehicle speed = 167.00 km/hr

Driver's speed is greater than 20 km/hr
Vehicle approaching is less than 0.5 meters away
Validating Vehicle
Vehicle Validated

Please measure the weight of the TRUCK
Error: Device address: 0x48
name 'f' is not defined
Truck weight: 7090 kg
Denied Access, the weight is greater than 7000 kg
```

Figure 15. Truck Access Denied on Terminal

```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature Outside is:22C
Current Humidity Outside is:70%

Vehicle approaching is less than 1 meter away
Vehicle speed = 44.00 km/hr

Vehicle approaching is less than 0.5 meters away
Validating Vehicle
Vehicle Validated

Please measure the weight of the TRUCK
Error: Device address: 0x48
name 'f' is not defined
Truck weight: 1223 kg
The gate has been opened!

* Running on http://0.0.0.0:5020/ (Press CTRL+C to quit)
```

Figure 16. Truck Entry Access on Terminal

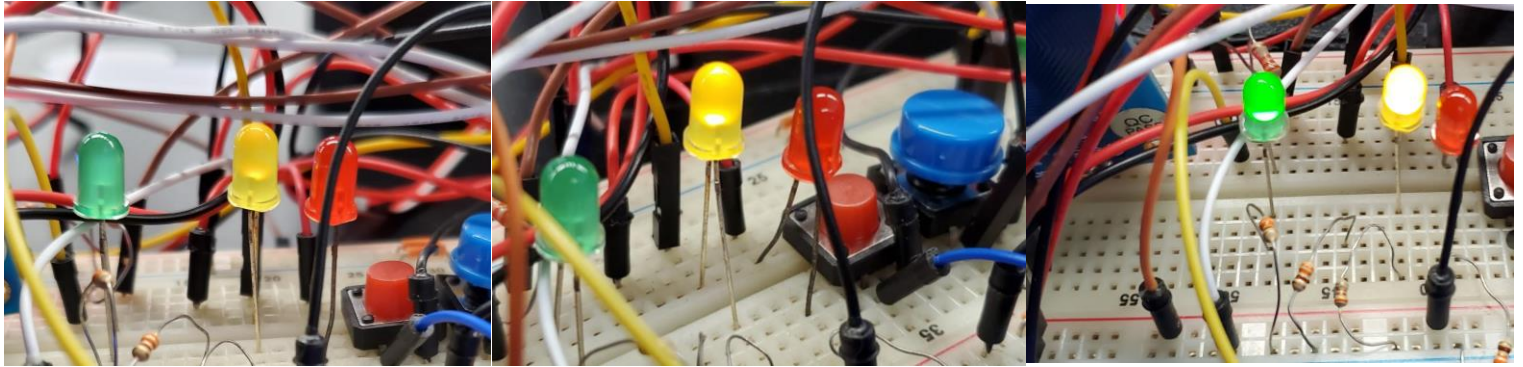


Figure 17. LED Brightness Based on Weight

If the vehicle validation through RFID tag is failed, the buzzer starts buzzing and the LCD displays a message stating that the RFID is not valid and then asking the vehicle user to enter a password to access the website. This password is supposed to be the first 4 digits of the unique tag ID. If even that fails, the buzzer will change the frequency of its buzzing to a higher pitch and they are denied entry.



Figure 18. RFID Invalid Display on LCD



Figure 19. Enter Password Prompt on LCD



Figure 20. Password is Wrong on LCD

```

Welcome to AUS Smart Parking Toll Gate
:~e
Current Temperature is:22C
Current Humidity is:70%

:S
Vehicle approaching is less than 1 meter away
-o
Vehicle speed = 165.00 km/hr
f
id
:h
:d
Vehicle approaching is less than 0.5 meters away
Validating Vehicle
S
> RFID not valid!
$
Access Denied
m
..

```

Figure 21. Access Denied on Terminal

If the password entered is correct, the buzzer will stop, and the user will be asked on the LCD to use the flask to open the gate and on the terminal

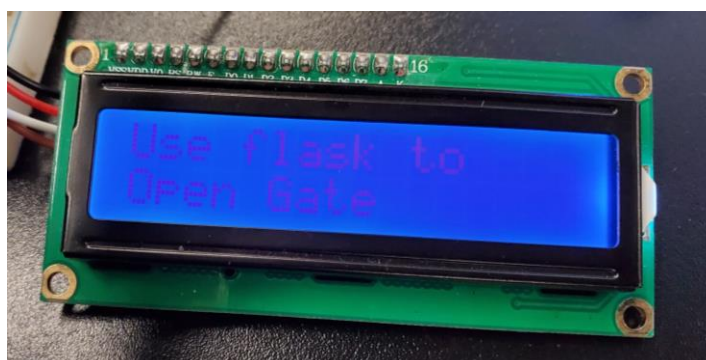


Figure 22. Flask Prompt on LCD


```

Welcome to AUS Smart Parking Toll Gate

Current Temperature is:22C
Current Humidity is:71%

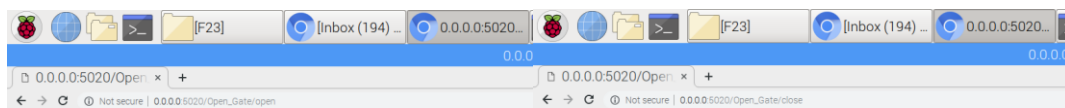
Vehicle approaching is less than 1 meter away
Vehicle speed = 30.00 km/hr

Driver's speed is greater than 20 km/hr
Vehicle approaching is less than 0.5 meters away
Validating Vehicle
RFID not valid!
Successful! Please use flask to open gate
* Running on http://0.0.0.0:5020/ (Press CTRL+C to quit)

```

Figure 23.Successful Password Entry Terminal

Once the user accesses the flask, they will be able to open or close the gate using it. With the green LED turning on when the gate opens.

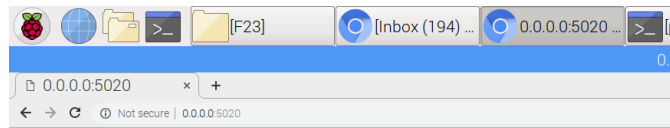


The Gate is Open

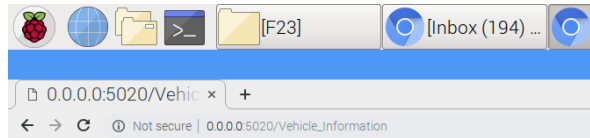
The Gate is closed

Figure 24. Flask Routes Open/Close Gate

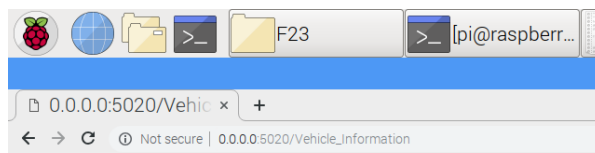
Using the flask the user will also be able to view the toll gate fees and information regarding their vehicle. They will first be greeted by the index page with a welcome message



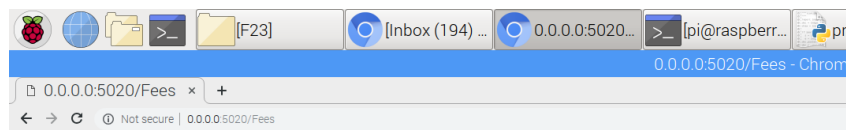
Welcome to the Smart Toll Gate system



Speed = 47km/Hr. Slow Down!!



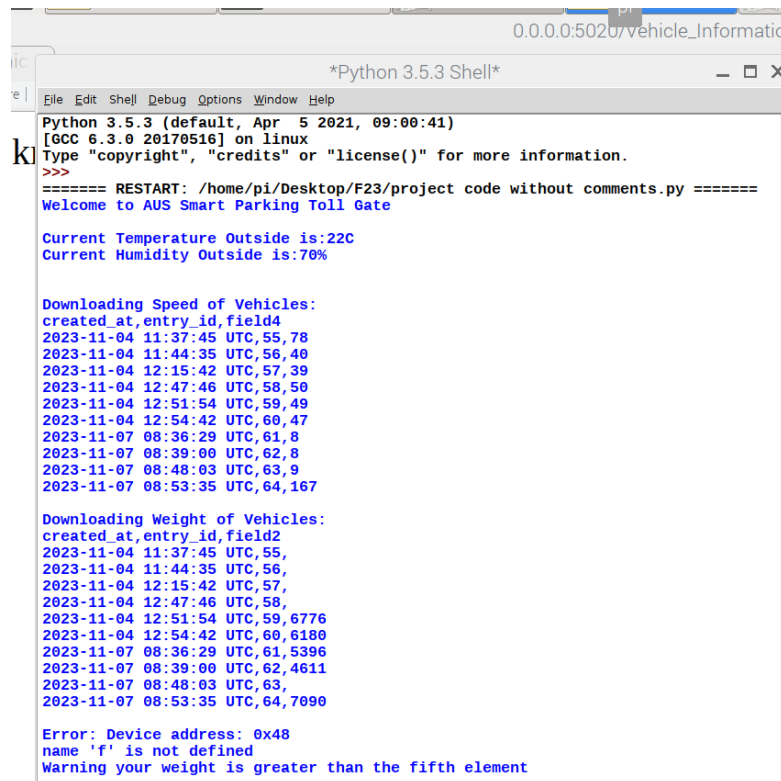
Speed = 10km/Hr



Car fees = 15 Dhs | Motorbike fees 10 Dhs | Truck fees 20 Dhs

Figure 25. Various Static Route Displays

Using thingspeak, the user will be able to download data regarding the vehicle's speed and weight by pressing on the pushbutton set for the user. This will be displayed on the terminal. The obtained data is compared to the 5th element. This is done because it's important to ensure the data received from external sources (thingspeak) is consistent so by comparing the incoming data to this specific element, the system can verify that it is receiving the latest and relevant information and the system can monitor changes in variables over time.



```
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/project code without comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature Outside is:22C
Current Humidity Outside is:70%

Downloading Speed of Vehicles:
created_at,entry_id,field4
2023-11-04 11:37:45 UTC,55,78
2023-11-04 11:44:35 UTC,56,40
2023-11-04 12:15:42 UTC,57,39
2023-11-04 12:47:46 UTC,58,50
2023-11-04 12:51:54 UTC,59,49
2023-11-04 12:54:42 UTC,60,47
2023-11-07 08:36:29 UTC,61,8
2023-11-07 08:39:00 UTC,62,8
2023-11-07 08:48:03 UTC,63,9
2023-11-07 08:53:35 UTC,64,167

Downloading Weight of Vehicles:
created_at,entry_id,field2
2023-11-04 11:37:45 UTC,55,
2023-11-04 11:44:35 UTC,56,
2023-11-04 12:15:42 UTC,57,
2023-11-04 12:47:46 UTC,58,
2023-11-04 12:51:54 UTC,59,6776
2023-11-04 12:54:42 UTC,60,6180
2023-11-07 08:36:29 UTC,61,5396
2023-11-07 08:39:00 UTC,62,4611
2023-11-07 08:48:03 UTC,63,
2023-11-07 08:53:35 UTC,64,7090

Error: Device address: 0x48
name 'f' is not defined
Warning your weight is greater than the fifth element
```

Figure 26. ThingSpeak Download on Terminal

As part of the system, an operator pushbutton was assigned in order to deal with trespassing attempts. In the code this is set as the interrupt. When the operator presses the pushbutton, and the warning red LED is turned on and it flashes three times, the incident is run in the flask and on terminal to show an attempt of unauthorized trespassing.

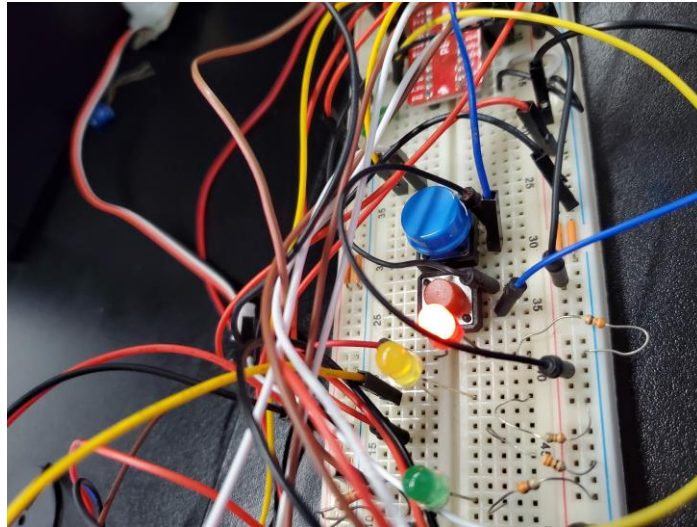


Figure 27. Red LED For Trespassing



No unauthorised access detected

Warning! Unauthorised access detected at 2023-11-07T13:01:44.098062

Figure 28. Flask Trespassing Displays

```

1 -- carcode[0:4] or key final == motorbikecode[0:4] --
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Apr 5 2021, 09:00:41)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/Desktop/F23/Project Code with comments.py =====
Welcome to AUS Smart Parking Toll Gate

Current Temperature is:22C
Current Humidity is:70%

Unauthorized Vehicle Trespassing the Toll Gate

* Running on http://0.0.0.0:5020/ (Press CTRL+C to quit)

```

Figure 29. Trespassing Event on Terminal

Final System Connections:

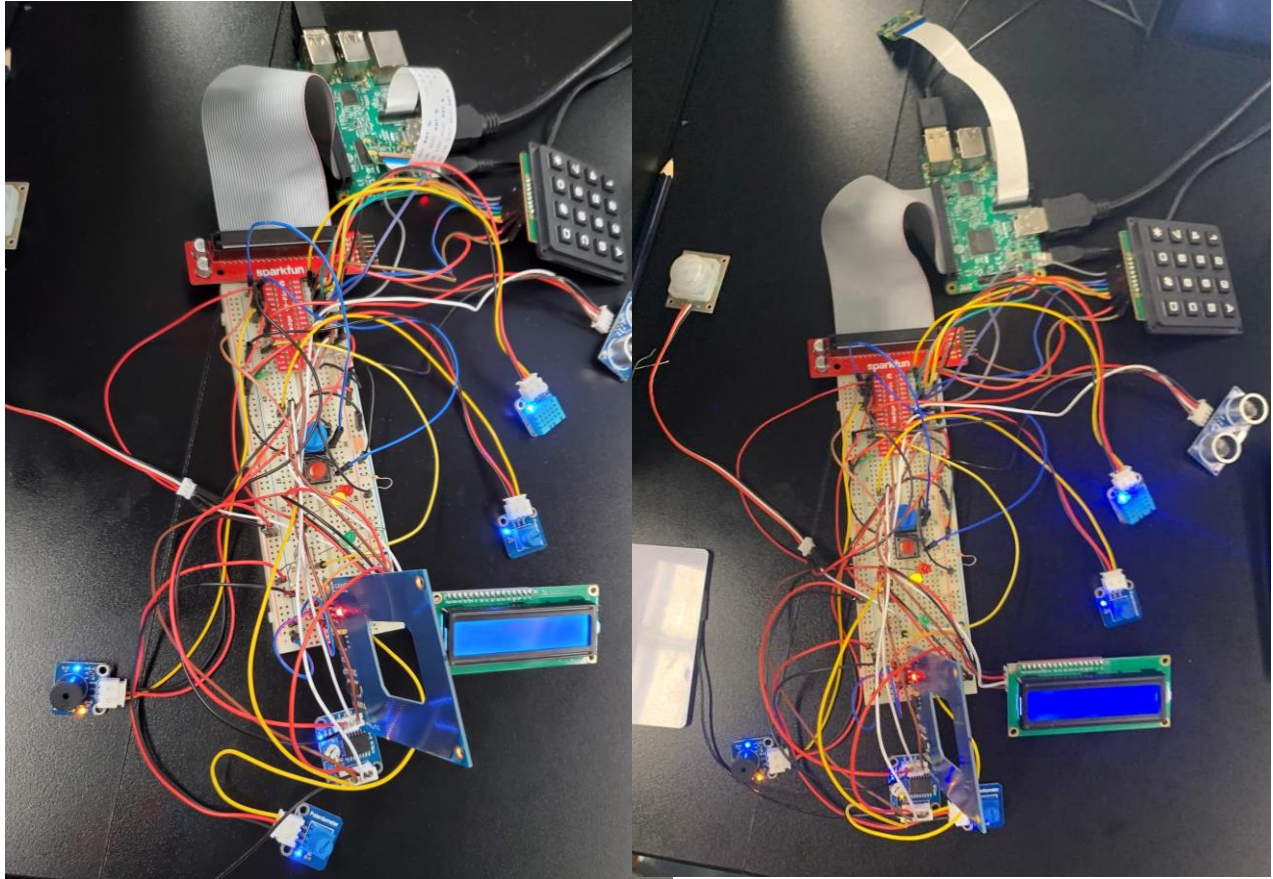


Figure 30. Final System Connections

Additional Executions/ Screenshots:

```
pi@raspberrypi: ~  
File Edit Tabs Help  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.25.34.151 netmask 255.255.248.0 broadcast 10.25.39.255  
    inet6 fe80::ba27:ebff:fe36:6bae prefixlen 64 scopeid 0x20<link>  
    ether b8:27:eb:36:6b:ae txqueuelen 1000 (Ethernet)  
    RX packets 10815790 bytes 1853015663 (1.7 GiB)  
    RX errors 0 dropped 336553 overruns 0 frame 0  
    TX packets 340343 bytes 83842935 (79.9 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@raspberrypi:~$ scrot  
pi@raspberrypi:~$ i2c -y detect  
bash: i2c: command not found  
pi@raspberrypi:~$ scrot -d 4  
pi@raspberrypi:~$ i2cdetect -y 1  
    0 1 2 3 4 5 6 7 8 9 a b c d e f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- 27 -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- 48 -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Figure 31 I2C Command

```
pi@raspberrypi:~$ ifconfig  
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether b8:27:eb:63:3e:fb txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 92782 bytes 34166912 (32.5 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 92782 bytes 34166912 (32.5 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.25.34.151 netmask 255.255.248.0 broadcast 10.25.39.255  
    inet6 fe80::ba27:ebff:fe36:6bae prefixlen 64 scopeid 0x20<link>  
    ether b8:27:eb:36:6b:ae txqueuelen 1000 (Ethernet)  
    RX packets 10815790 bytes 1853015663 (1.7 GiB)  
    RX errors 0 dropped 336553 overruns 0 frame 0  
    TX packets 340343 bytes 83842935 (79.9 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@raspberrypi:~$ scrot
```

Figure 32 ifconfig Command



Figure 33. Camera Capture :)

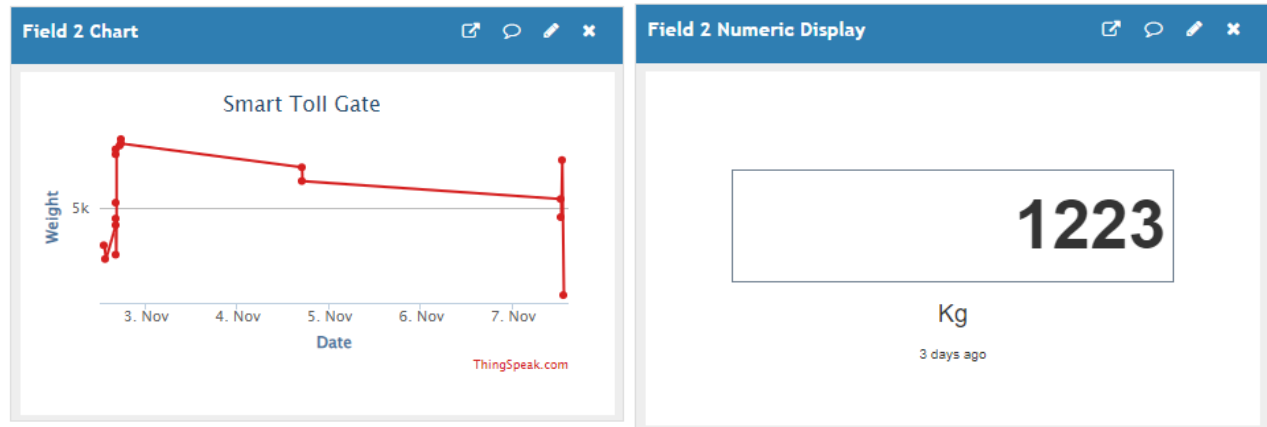


Figure 34. ThingSpeak Weight Widgets

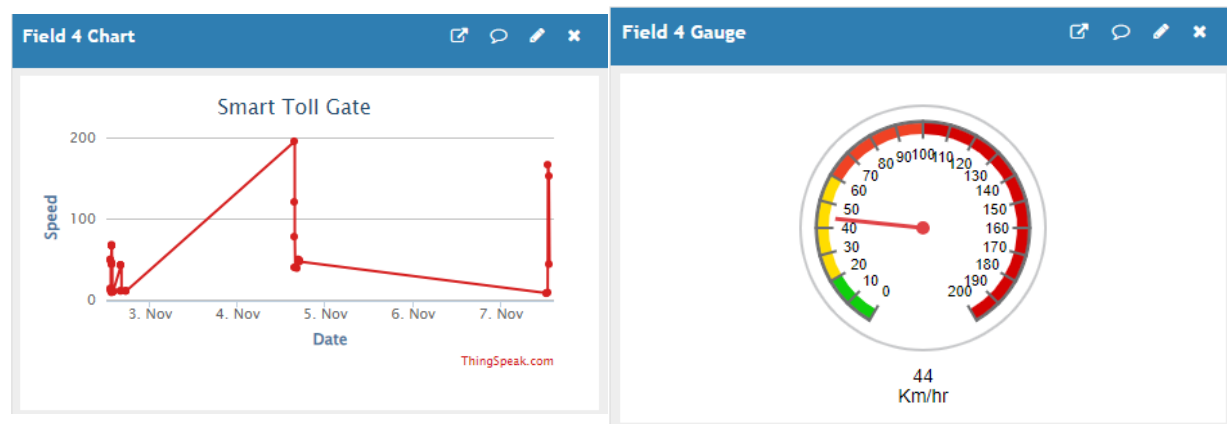


Figure 35. ThingSpeak Speed Widgets

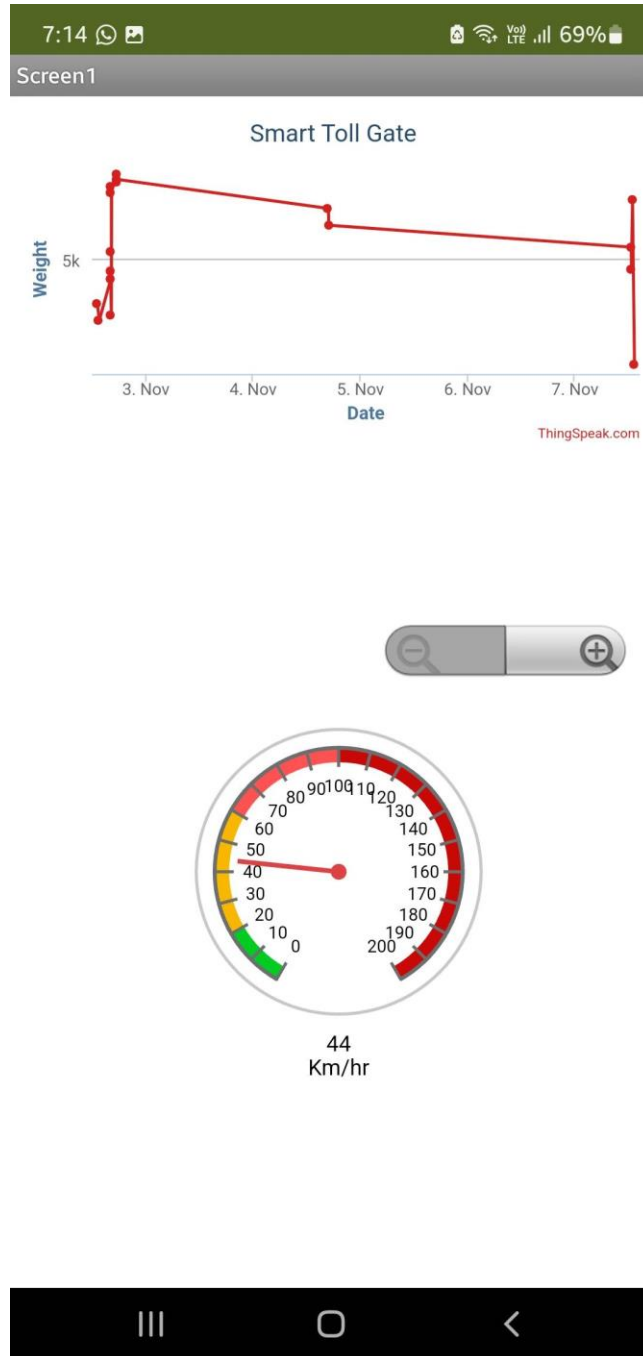


Figure 36. MIT App Widgets