

INTERNET OF THINGS-GROUP4
PUBLIC TRANSPORTATION OPTIMIZATION
PHASE-5

STUDENT NAME :ANUSHA R
REGISTER NUMBER : 822621106006
COLLEGE NAME :ARIFA INSTITUTE OF TECHNOLOGY
COLLEGE CODE :8226
NM ID :au822621106006
EMAIL ID :anusharavichandran8@gmail.com

The purpose of this thesis was to use Internet of Things (IoT) in the public transportation system . The information gathered using IoT devices could be used when making decisions to improve the public transportation system. Through IoT one would be able to track the locations of public vehicles and see the number of passengers in that vehicle. It could be used to notify drivers when the number of passengers exceed the limit of the vehicle. The thesis was divided into two parts: theory and practical. The theory part defined the general idea about IoT, its use cases and technologies involved in an IoT system. In the practical part IoT devices such as the Raspberry Pi, infrared sensors and LED lights were used to demonstrate the working idea of the project. The Infrared sensors were used to count the number of passengers in the vehicle. The LEDs were used to warn drivers if the vehicle exceeded the passenger limit

INTRODUCTION:

Internet of Things, also known as IoT, is the term used when devices or objects are embedded with the ability to communicate through the internet. Here, the devices can collect data from the environment and send that data either to cloud or share it with other related devices in the network. Nowadays,

IoT is grabbing a lot of attention, with the terminologies like smart homes and smart cities. Everyday appliances like coffee makers, refrigerators and TVs are already equipped with the capability of connecting to the internet.

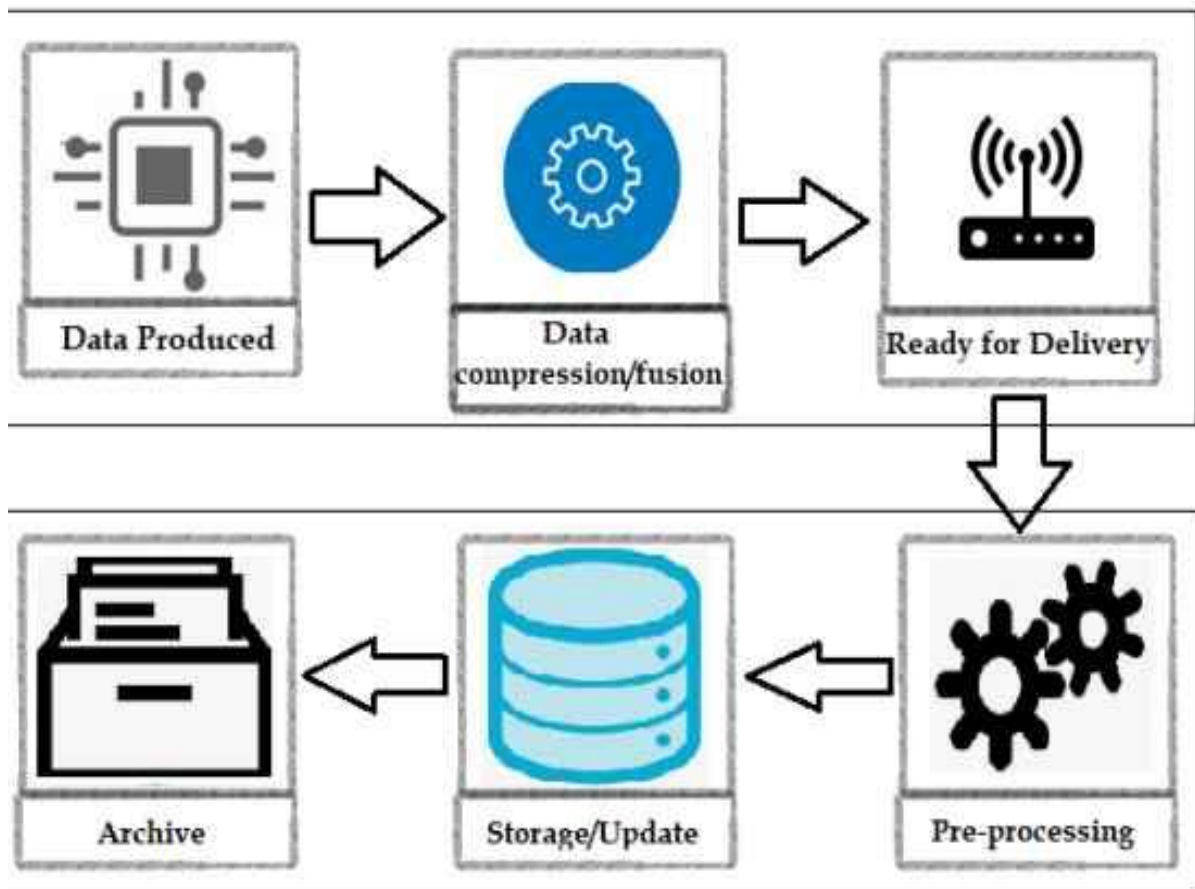
Use of IoT in the Public Transportation :

Public transportation is a transport service available for general public that operates on fixed routes and charges a certain fee for each trip. Public transportation helps to reduce air pollution and traffic congestion by providing the service to many people at the same time. Public transportation also influences the quality of life in urban city. (Clean Energy Nepal 2014.) In order to afford living in the urban city people have to work and they are required to arrive at work on time. Sometimes they might also have meetings that are related to work. If the public transportation they are using are unreliable, they will be in constant stress when heading out to reach their destination.

Application Protocols:

In IoT, end devices are connected to the back-end servers using different communication technologies. Depending on their applications, IoT devices are either connected through a gateway, that forms its own LAN, or through cellular networks that covers wider range. After a secured connection is made, the devices can send their data to the server or to a cloud using the internet. For IoT applications, it is important to use those protocols that can work with devices running on battery power with lower computing capacity. (Collina et al. 2014.) Some of the popular application layer protocols used in

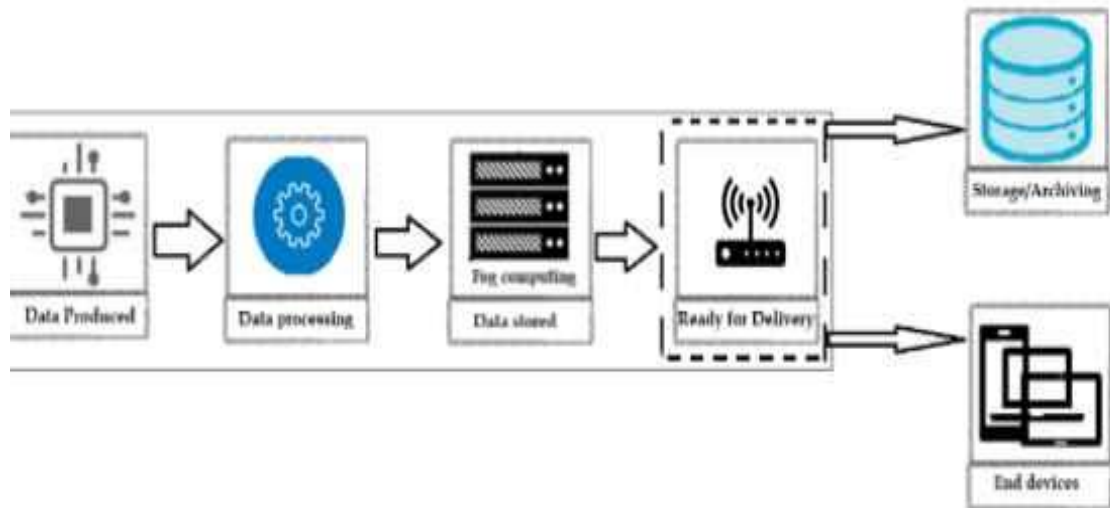
IoT applications are: MQTT, CoAP, AMQP, XMPP and DDS.



Lifecycle of data in an IoT system

Lifecycle of a data in IoT starts from its production. Data is produced using sensors and devices that are used in an IoT network. IoT devices can be programmed to produce data periodically or they are programmed to produce data only when a query is sent. Some IoT devices are able to store the data for a short period of time before sending it to the gateway of the network. The devices use short range communication technologies to send these data to the gateways. There can be more than one device connected in an IoT network gateway and all the data from these devices are collected in that gateway. (Sb.) After the data are collected in the gateway
The lifecycle of data in an autonomous IoT system

working in real-time is different from that of other non-autonomous applications. An autonomous IoT system can refer to IoT devices used in factories to monitor the condition of machines and to check the air quality and humidity levels inside the buildings, sensors that are used in self-driving vehicles and smart devices used in hospitals that constantly monitor the condition of a patient. In such situations, after the data are produced they are instantly processed within the network. The device itself can also process the data without having to send them further up the network, provided that the device is equipped with enough computing power. Otherwise, the processing can take place at the edge of the network, in the gateways. This way the response time is quicker and affective. The processed data are then stored within the network. One can access these data using end-devices that request the data from the network. This type of setup is suitable for applications where it is not necessary to send all the data further up the network for in-depth analysis and storage. The gateway can filter the data and only send the summarized version of the collected data. It is also important to note that the storage in these types of setups is limited, meaning that older data gets deleted when new data are updated. (Sb.) The lifecycle of data in an autonomous IoT system can be seen in Figure



Lifecycle of an IoT data in an autonomous IoT system

Hardware Setup The components used in this project are:

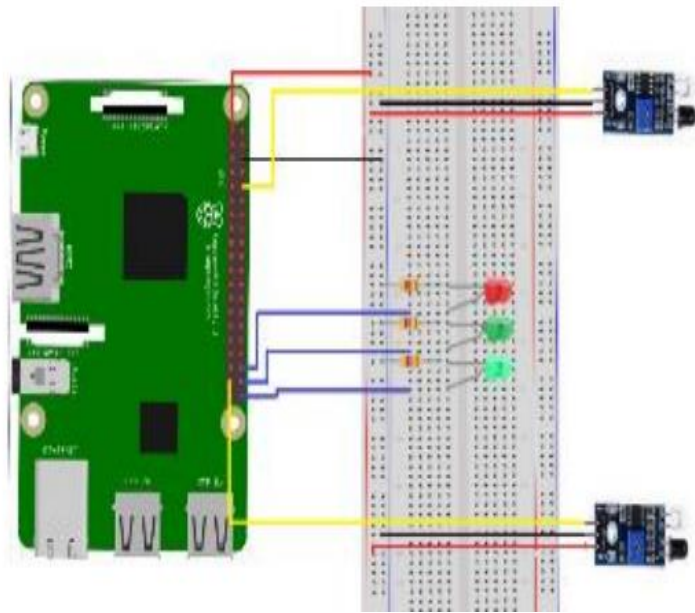
- One Raspberry Pi 3
- SD card
- Two IR Obstacle sensors
- Three LEDs
- Three resistors
- Wires
- Breadboard

The Raspberry Pi was mounted with a SD card which had Raspbian OS installed. Using the breadboard, the LEDs were connected to GPIO pins with protecting resistors. In this case GPIO17 was connected to the green LED, GPIO22 to the blue LED and GPIO20 to the red LED. The IR obstacle sensor has three pins: SIG, VCC and GND, as shown in figure

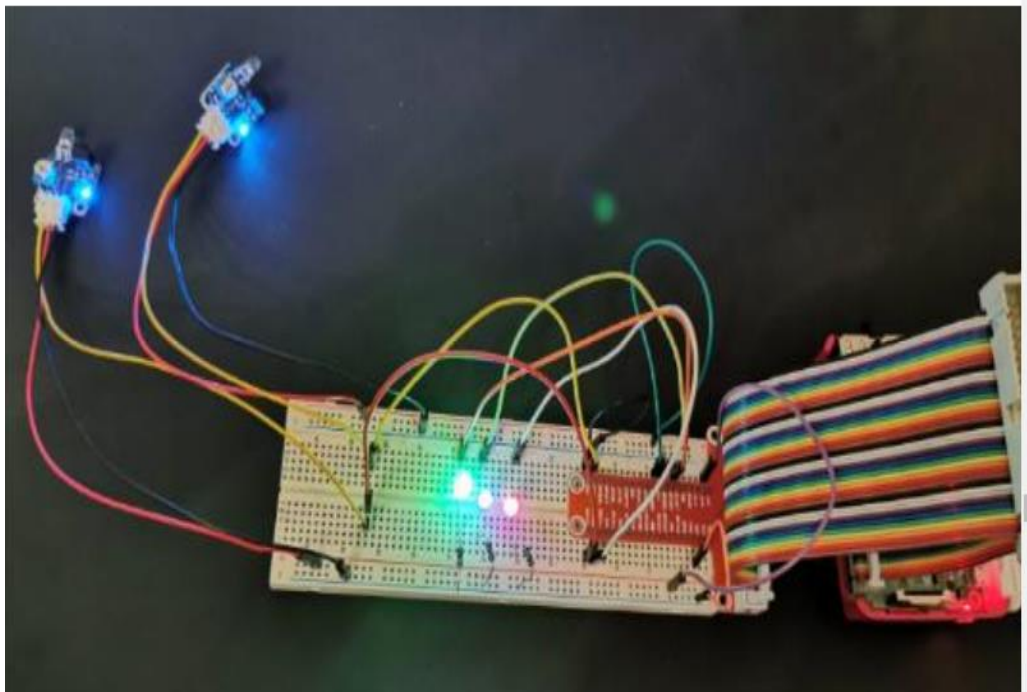


IR Obstacle Sensor

In the sensor SIG is for the signal, VCC and GND are for power and ground respectively. The SIG pin of the first IR obstacle sensor was connected to GPIO21 which would detect a passenger boarding the bus. The SIG pin of the second IR obstacle sensor was connected to GPIO26. This will detect a passenger exiting the bus. The power and ground pins were connected to their appropriate pins in the Raspberry Pi.



Raspberry Pi circuit diagram



Raspberry Pi circuit

shows the circuit diagram of the project and Figure 9 shows how all of the devices are connected with the Raspberry Pi.

Python Code

In the first part of the Python code different modules were imported. To control the LEDs and GPIO channels of the Raspberry Pi, `gpiozero` and `RPi.GPIO` modules were imported. To manipulate time and date, according to the needs, `time` and `datetime` modules were imported. For connecting to the MariaDB server a `mysql.connector` module was used. Finally, a `csv` module was imported to write the data obtained through the sensor to a csv file in the local directory. The Python code used for importing these modules are as following:

```
import RPi.GPIO as GPIO
from gpiozero import LED
import time
from datetime import datetime
import mysql.connector
import csv
```

After importing the necessary modules, the Python code goes on to assign the GPIO pins to its respective LEDs and sensors. For the numbering of the pins the BCM numbering system was used. The Python codes are as following:

```
passengerCountIn = 21 # detects when passenger enter the bus
```

```
LedIn = LED(17) # lights up to indicate passenger in  
passengerCountOut = 26 # detects when passenger leaves the bus  
LedOut = LED(22) # lights up to indicate passenger exit  
LedMax = LED(20) # lights up to indicate the bus is full
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(passengerCountIn, GPIO.IN)  
GPIO.setup(passengerCountOut, GPIO.IN)
```

The next part of the code is used for connecting to the MariaDB server. This code contains the name of the server host, username, password and the database, as shown below:

```
mydb = mysql.connector.connect(host="localhost",  
user="gaurab",passwd="gaurav",  
db="CSVFile") # connect to MariaDB server  
cur = mydb.cursor()
```

The rest of the codes defines the function which are as following:

```
count = 0  
dbMsg = "0"  
def date_now():  
    now = datetime.now()  
    dateString = now.strftime("%Y-%m-%d") # shows date  
    return(dateString)
```

```
def time_now():
    now=datetime.now()
    timeString = now.strftime("%H%M") # shows time
    return(timeString)
def one_passenger():
    LedIn.on()
    time.sleep(0.1)
    LedIn.off()
    return("{} Passenger In!".format(count))
def more_passenger():
    LedIn.on()
    time.sleep(0.1)
    LedIn.off()
    return("There are {} Passengers in the bus!".format(count))
def max_passenger():
    LedIn.on()
    LedMax.on()
    time.sleep(0.1)
    LedIn.off()
    return("PASSENGER LIMIT REACHED!")
def one_passenger_leave():
    LedOut.on()
    LedMax.off()
    time.sleep(0.1)
    LedOut.off()
    return("Passenger Out! {} Passengers remaining!".format(count))
def zero_passenger():
    LedOut.on()
    LedMax.off()
    time.sleep(0.1)
```

```

    LedOut.off()
    return("Nb Passengers remaining!")
def overload_message():
    LedOut.on()
    time.sleep(0.1)
    LedOut.off()
    return("Passenger Out! {} Passengers remaining! PASSENGER LIMIT
    REACHED!".format(count))
def write_csv(bb): # create csv file to save the data
    with open('/home/pi/Documents/Server/passenger_data1_2.csv',
    mode='a') as
    passenger_data:
        sensor_data = csv.writer(passenger_data, delimiter=',',
    quotechar='"', quoting=csv.QUOTE_MINIMAL)
        write_data = sensor_data.writerow([date_now(), time_now(), count,
    bb])
    return(write_data)
def databaseConnection(): # connect to sql database
    sql = "INSERT INTO Passenger_Data (DATE, TIME, STATUS, MESSAGE)
    VALUES (%s, %s, %s, %s)"
    val = (date_now(), time_now(), count, dbMsg)
    cur.execute(sql, val)
    mydb.commit()
    print(cur.rowcount, "record inserted.")
    while True:

        if 0 == GPIO.input(passengerCountIn): # when passenger enters
            count += 1
            if count == 1:
                print(one_passenger())

```

```

dbMsg = one_passenger()
write_csv(one_passenger()) # writes to csv file
databaseConnection() # writes to sql database
elif count >= 15:
    print(max_passenger())
    dbMsg = max_passenger()
    write_csv(max_passenger())
    databaseConnection()
else:
    print(more_passenger())
    dbMsg = more_passenger()
    write_csv(more_passenger())
    databaseConnection()
else:
    if 0 = GPIO.input(passengerCountOut): # when passenger exits
    if count <= 0:
        print(zero_passenger())
        dbMsg = zero_passenger()
        write_csv(zero_passenger())
        databaseConnection()
    elif 0 < count <= 15:
        count -= 1
        print(one_passenger_leave())
        dbMsg = one_passenger_leave()
        write_csv(one_passenger_leave())
        databaseConnection()

    else:
        count -= 1
        print(overload_message())

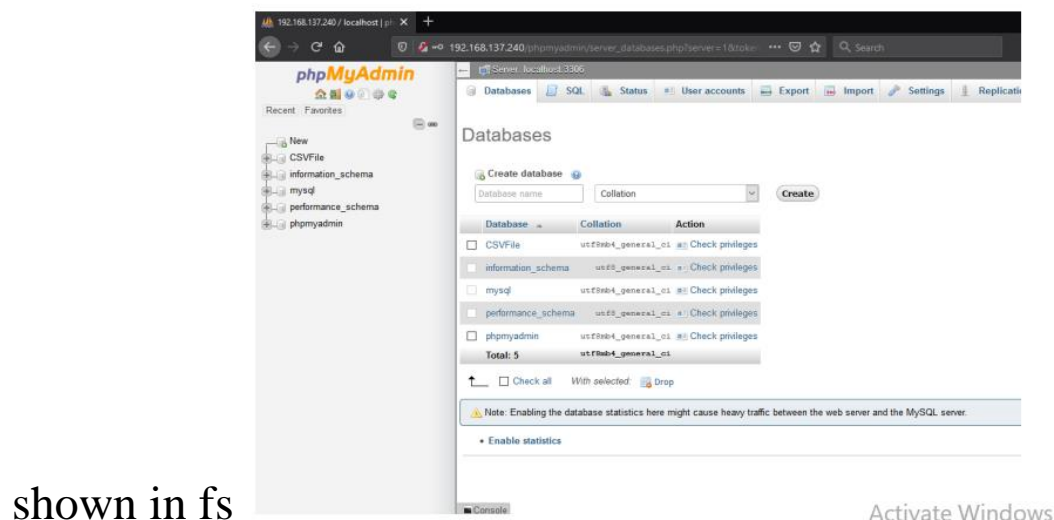
```

```
dbMsg = overload_message()
write_csv(overload_message())
databaseConnection()
```

Through these codes LEDs are turned on and off according to the data from the sensors. The write_csv(bb) function was used for creating a csv file where all the data from the sensor are collected in the local directory. The databaseConnection() function inputs the value in the sql database in its respective columns.

Data Collection:

For data collection in the MariaDB database I first had to create a database and a table. First, through the web browser I connected to PHPMYAdmin. Using the username and password that I created I was able log in to PHPMYAdmin as



shown in fs

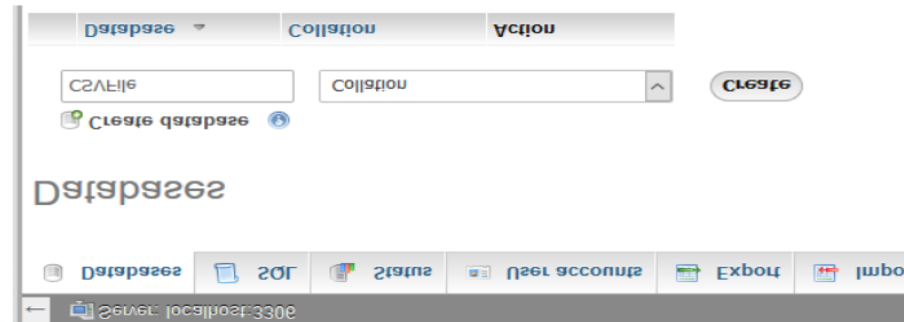
PHPMYAdmin server

Inside the server I created a new database called CSVFile using Create database as shown in Figure

is introduced in Figure 14.

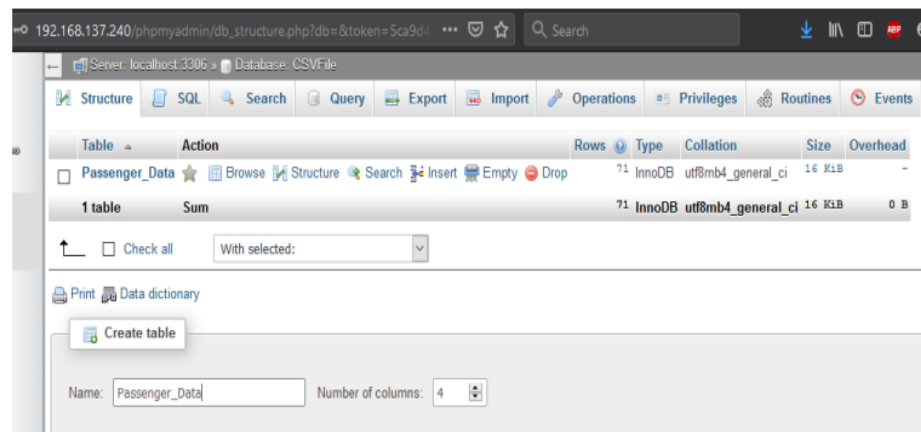
Inside the CSVFile database I created a new table called Passenger_Data. This

Figure 13: Creating a new database



Creating a new database

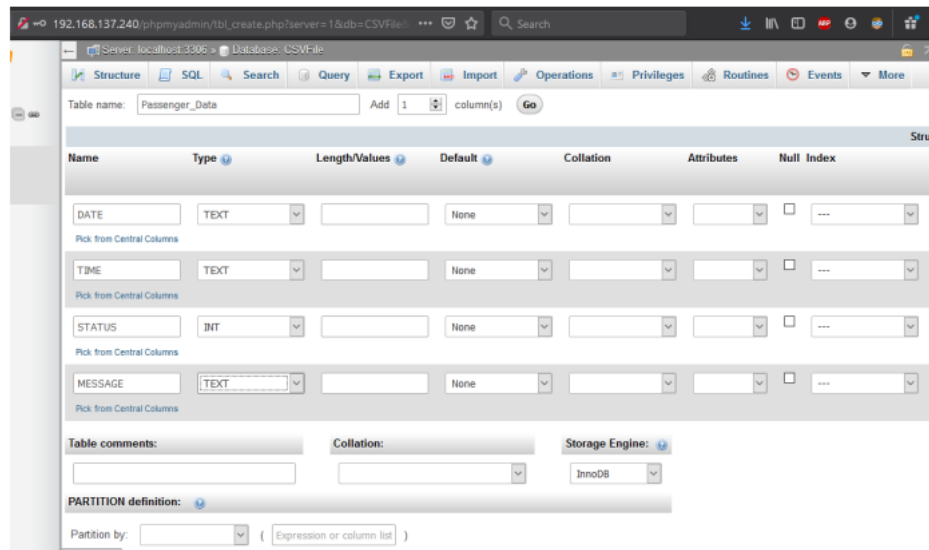
Inside the CSVFile database I created a new table called Passenger_Data. This is introduced in Figure



Creating a new table

Inside the table I created four columns: DATE, TIME, STATUS, MESSAGE. The value for all of these columns will be filled using python code which is why I put TEXT as a Type for DATE, TIME and MESSAGE, which is shown in

Figure 15. The status will show how many passengers are in the bus



Creating columns and their types

After creating the database and a table inside the database, I added that information in my Python code. Through that code the data from the sensor was copied to the sql database table. After successfully uploading the data to the database it prints out a message “1, ‘record inserted.’”, as shown in Figure

```

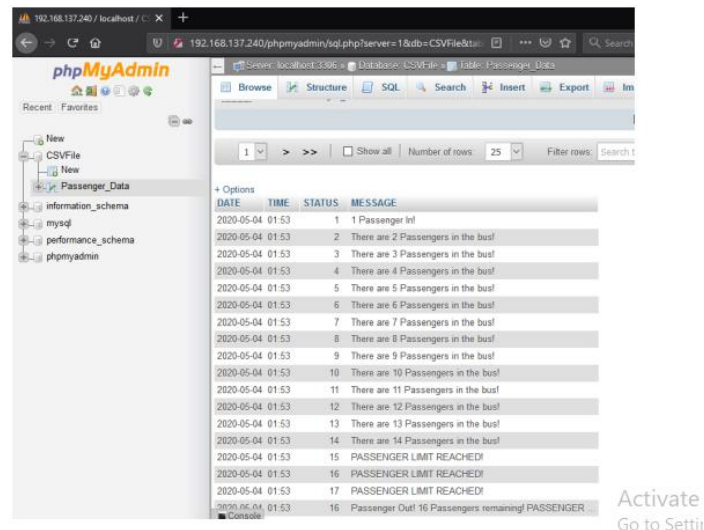
pi@raspberrypi: ~/Pic... passenger_count1.2...
File Edit Tabs Help
There are 7 Passengers in the bus!
(1, 'record inserted.')
There are 8 Passengers in the bus!
(1, 'record inserted.')
There are 9 Passengers in the bus!
(1, 'record inserted.')
There are 10 Passengers in the bus!
(1, 'record inserted.')
There are 11 Passengers in the bus!
(1, 'record inserted.')
There are 12 Passengers in the bus!
(1, 'record inserted.')
There are 13 Passengers in the bus!
(1, 'record inserted.')
There are 14 Passengers in the bus!
(1, 'record inserted.')
PASSENGER LIMIT REACHED!
(1, 'record inserted.')
PASSENGER LIMIT REACHED!
(1, 'record inserted.')
Passenger Out! 15 Passengers remaining! PASSENGER LIMIT REACHED!
(1, 'record inserted.')
Passenger Out! 14 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 13 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 12 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 11 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 10 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 9 Passengers remaining!
(1, 'record inserted.')
Passenger Out! 8 Passengers remaining!
(1, 'record inserted.')

```

Activate Wind
Go to Settings to a

Sensor sending data to the database

In the PHPMyAdmin server, inside the Passenger_Data we can click refresh to see the new data that has been copied to the database as Figure



The screenshot shows the PHPMyAdmin web interface. On the left, the database structure is visible, with 'Passenger_Data' selected. The main panel displays a table with the following data:

DATE	TIME	STATUS	MESSAGE
2020-05-04	01:53	1	1 Passenger In!
2020-05-04	01:53	2	There are 2 Passengers in the bus!
2020-05-04	01:53	3	There are 3 Passengers in the bus!
2020-05-04	01:53	4	There are 4 Passengers in the bus!
2020-05-04	01:53	5	There are 5 Passengers in the bus!
2020-05-04	01:53	6	There are 6 Passengers in the bus!
2020-05-04	01:53	7	There are 7 Passengers in the bus!
2020-05-04	01:53	8	There are 8 Passengers in the bus!
2020-05-04	01:53	9	There are 9 Passengers in the bus!
2020-05-04	01:53	10	There are 10 Passengers in the bus!
2020-05-04	01:53	11	There are 11 Passengers in the bus!
2020-05-04	01:53	12	There are 12 Passengers in the bus!
2020-05-04	01:53	13	There are 13 Passengers in the bus!
2020-05-04	01:53	14	There are 14 Passengers in the bus!
2020-05-04	01:53	15	PASSENGER LIMIT REACHED!
2020-05-04	01:53	16	PASSENGER LIMIT REACHED!
2020-05-04	01:53	17	PASSENGER LIMIT REACHED!
2020-05-04	01:53	16	Passenger Out! 16 Passengers remaining! PASSENGER

New data in the database

We can also access these data from the Raspberry Pi terminal. To do that we can use the following commands: `$sudo mysql -u root -p`; `$show databases;` `$use CSVFile;` `$show tables;` `$SELECT*FROM Passenger_Data;` The output of these commands is shown in Figure

We can also access these data from the Raspberry Pi terminal. To do that we

can use the following commands:

```
$sudo mysql -u root -p
```

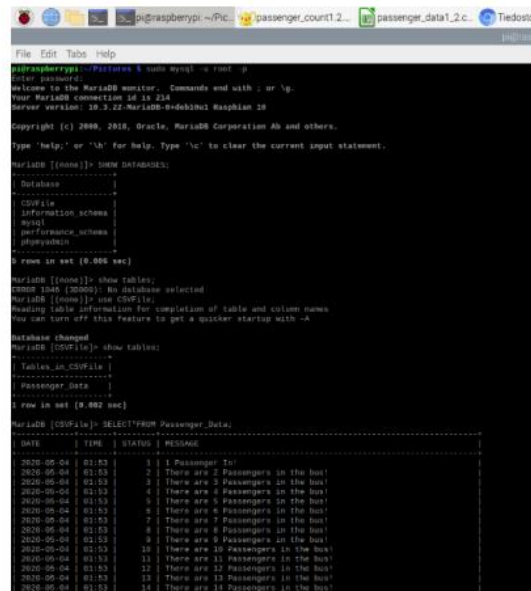
```
$show databases;
```

```
$use CSVFile;
```

```
$show tables;
```

\$SELECT*FROM Passenger_Data;

The output of these commands is shown in Figure



```
mysql> show databases;
Database
+-----+
CSVFile
information_schema
mysql
performance_schema
phpmyadmin
5 rows in set (0.004 sec)

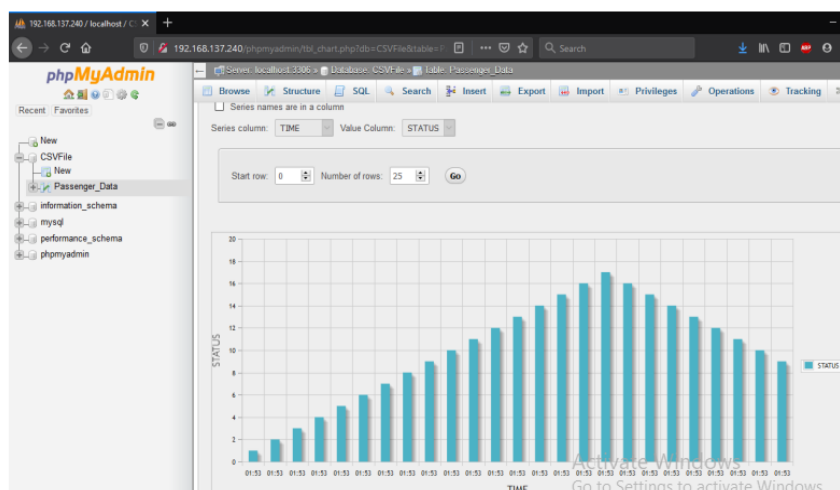
mysql> use CSVFile;
mysql> show tables;
Tables_in_CSVFile
+-----+
Passenger_Data
1 row in set (0.002 sec)

mysql> SELECT * FROM Passenger_Data;
+-----+-----+-----+-----+
DATE      | TIME | STATUS | MESSAGE
+-----+-----+-----+-----+
2020-05-04 | 01:53 | 1      | 1 Passenger in the boat!
2020-05-04 | 01:53 | 2      | 2 There are 2 Passengers in the boat!
2020-05-04 | 01:53 | 3      | 3 There are 3 Passengers in the boat!
2020-05-04 | 01:53 | 4      | 4 There are 4 Passengers in the boat!
2020-05-04 | 01:53 | 5      | 5 There are 5 Passengers in the boat!
2020-05-04 | 01:53 | 6      | 6 There are 6 Passengers in the boat!
2020-05-04 | 01:53 | 7      | 7 There are 7 Passengers in the boat!
2020-05-04 | 01:53 | 8      | 8 There are 8 Passengers in the boat!
2020-05-04 | 01:53 | 9      | 9 There are 9 Passengers in the boat!
2020-05-04 | 01:53 | 10     | 10 There are 10 Passengers in the boat!
2020-05-04 | 01:53 | 11     | 11 There are 11 Passengers in the boat!
2020-05-04 | 01:53 | 12     | 12 There are 12 Passengers in the boat!
2020-05-04 | 01:53 | 13     | 13 There are 13 Passengers in the boat!
2020-05-04 | 01:53 | 14     | 14 There are 14 Passengers in the boat!
```

Ac

Accessing the table through command line

Another advantage of using a PHPMyAdmin server is that we can easily convert the sql data into visual representation



Column graph

Figure shows the visual representation of the data that was collected earlier using the Raspberry Pi.

Conclusion:

The aim of the project was to use IoT in the public transportation system of Nepal through which the Department of Transport Management could efficiently monitor, control and improve the public transportation services. In the practical part of the project IoT sensors were used to monitor the passenger data of a bus. It could also provide a visual warning for the driver, if the number of passengers in the bus exceeded its limit. Furthermore, the project goes on to show the possibility of monitoring this data through remote access.

The project also introduced IoT and its technologies. It briefly discussed the applications of IoT and how they were used in different sectors of the society. It explained how the data collected from IoT have become more crucial when making important decisions and when planning for the future. Businesses are using IoT to stay ahead in the market, cities are using IoT to save energy and resources. At the same time IoT is used in agriculture, transportation and healthcare. Some of the security features of IoT were also discussed.