INTERNET OF THINGS-GROUP4

PUBLIC TRANSPORTATION OPTIMIZATION

PHASE-5

STUDENT NAME	:MOHAMMED HAPIPU.B
REGISTER NUMBER	: 822621106302
COLLEGE NAME	:ARIFA INSTITUTE OF TECHNOLOGY
COLLEGE CODE	8226
NM ID	:aut2282260005
EMAIL ID	:mohammedhapipu@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. Based on that information a graph was used to visualize the number of passengers in the certain time of the day in different cities. This project was a prototype and could be further improved to develop a working device that could be deployed in a working environment. The data gathered in this project gave clear information about the number of people using public transportation service in different routes. This information can be crucial when making important decisions, such as, managing timetables for the public vehicles, deploying more or reducing the number of public vehicles from certain routes of the city.

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 networkNowadays,

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 interne

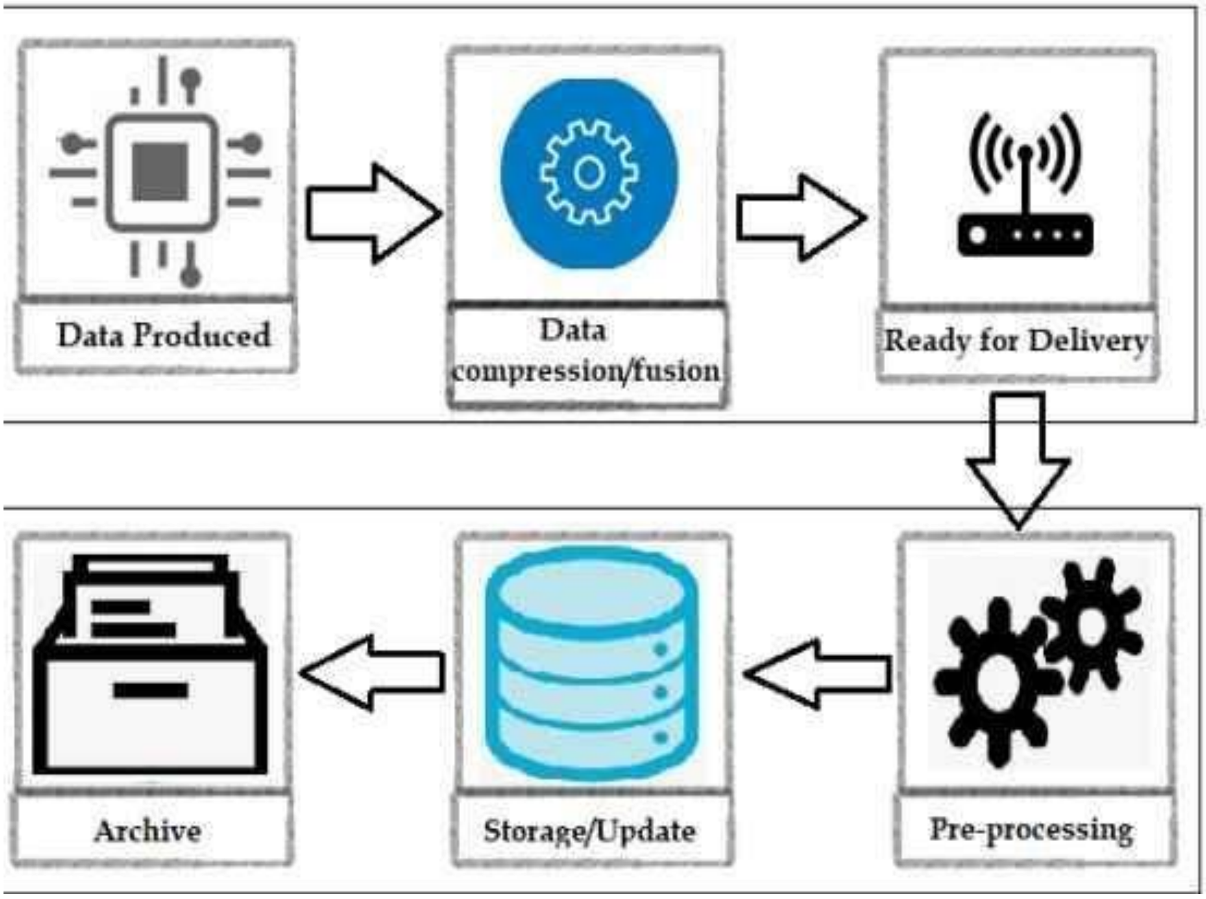
In IoT, devices are equipped with sensors, actuators, processors and hardware for wireless connection. Sensors are used to collect data from the surrounding physical environment. There are different kinds of sensors that have unique properties and are used in different circumstances, for example, temperature sensors, proximity sensors, IR sensors, accelerometer, pressure sensors and light sensors. Sensors are chosen depending on the application of IoT

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. This negatively influences the quality of life. Also, arriving late to work or in a meeting can negatively affect a person’s reputation. A poor public transportation can also refrain families from going out together, for example to a movie or to a park. Enjoying time with family can positively influence one’s quality of life. When people have access to a public transportation that they know they can rely on and feel safe when using it, they might not consider buying their own personal transportation device, for instance, a car or a motorcycle. This can result in less traffic on the road. In addition, people would not have to spend their income on fuel and maintenance of their vehicle. This can reduce the financial burden, especially for a family that has a low-income source. Public transportation can also provide employment opportunities to many people. Employment also plays a major role that affects the influence on the quality of a person’s life. Public transportation service needs to be affordable, well-organized and it should use the urban space efficiently (Clean Energy Nepal 2014). However, due to rapid urbanization and poor management of public transport service, the transportation system in Nepal is in dire need of improvement. Currently, most of the operation of public transport is carried out by privately owned organizations. There can be more than two different organizations running the public transportation service in the same route. In addition, these organizations lack proper coordination and they do not run on schedules. Due to which the roads in Nepal are congested with public vehicles and the public transportation service has become unreliable, resulting in poor service. This has led to big traffic congestion, increased pollution and increased road accidents. Also, the unreliability and poor transportation service has forced the general public to invest in their own personal transportation device. This has further increased the number of vehicles on the road. It is also difficult to collect the records of number of people that use the public transportation service as there are no proper ticket system in place.

Application Protocols:

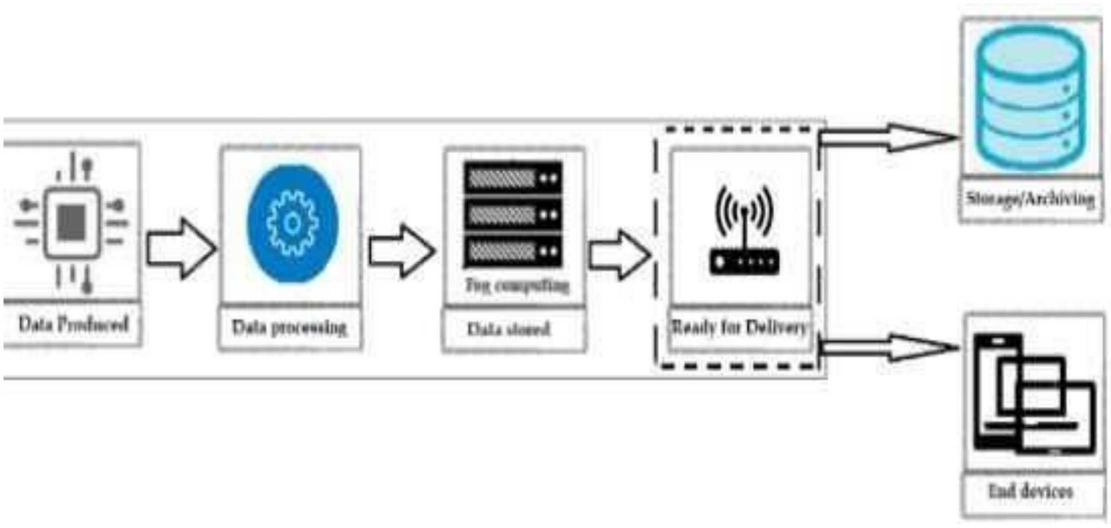
In IoT, end devices are connected to the back-end servers using different communication technologies. Depending on their applications, IoT devices are either connected through a gateway, that forms its own LAN, or through cellular networks that covers wider range. After a secured connection is made, the devices can send their data to the server or to a cloud using the internet. The server has to update its data every time a new value is produced by the end devices. Application layer protocols carry these new values from the end devices and deliver them to the servers. Application layer protocols are also responsible for delivering messages from the servers to users' applications. In addition, users are able to access and control the end devices through these protocols. 18 (Karagiannis et al. 2015, 1–2.) Popular application layer protocols such as HTTP, SMTP and FTP requires higher computing power to process them. 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 gateway performs filtration, data aggregation and compression of the raw data. This is done, because cost-wise it is very expensive to transmit a large amount of data through the network. They also have limited bandwidth, and constantly sending raw uncompressed data through the network creates latency. This can result in a life threatening situation in applications that require quick and real-time response and actions, for example in self-driving vehicles and health monitoring devices in a hospital. (Sb.)

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

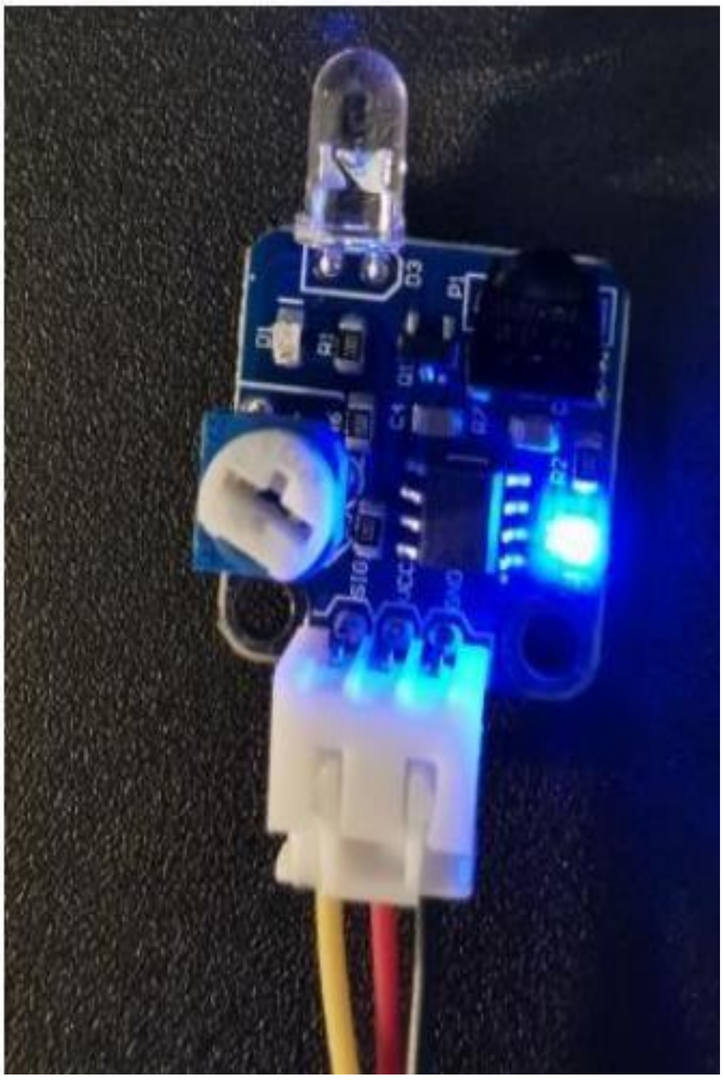


Lifecycle of an IoT data in an autonomous IoT system

Hardware Setup The components used in this project are:

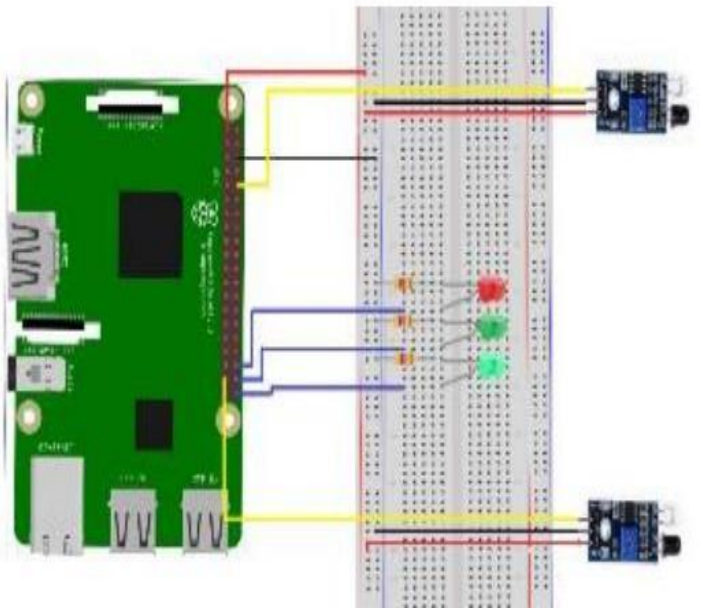
- One Raspberry Pi 3
- SD card
- Two IR Obstacle sensors
- Three LEDs
- Three resistors
- Wires
- Breadboard

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

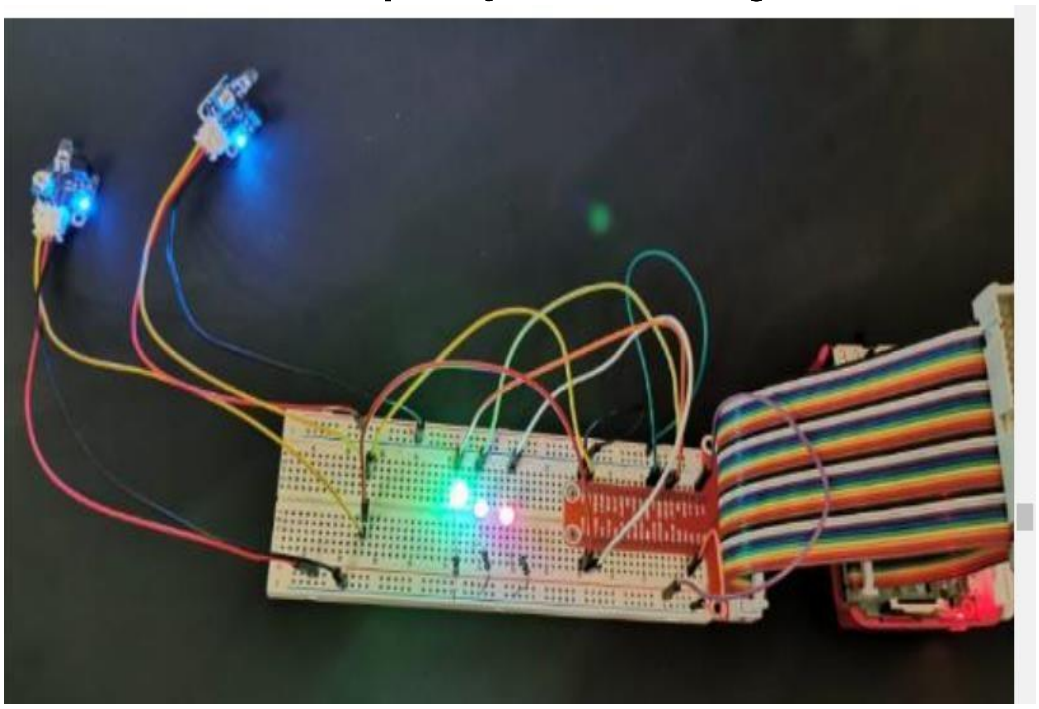


IR Obstacle SensOR

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 9shows 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) # lightsuptoindicatepassenger in
passengerCountOut =26 # detectswhenpassengerleavesthebus
LedOut =LED(22)# lightsup toindicatepassenger exit
LedMax =LED(20) #   lightsuptoindicatethebusisfull

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 toMariaDBserver
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")# showsdate
 return(dateString)
```

```
def time_now():
 now=datetime.now()
 timeString = now.strftime("%H%M") # show time
 return(timeString)
def one_passenger():
 LedIn.on()
 time.sleep(0.1)
 LedIn.off()
 return("D} Passenger In!".format(count))
def more_passenger():
 LedIn.on()
 time.sleep(0.1)
 LedIn.off()
 return("There are D} 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! D} Passengers remaining!".format(count))
def zero_passenger():
 LedOut.on()
 LedMax.off()
 time.sleep(0.1)
 LedOut.off()
return("No Passengers remaining!")
def overload_message():
 LedOut.on()
 time.sleep(0.1)
 LedOut.off()
 return("Passenger Out! D} Passengersremaining! PASSENGERLIMIT
REACHED!".format(count))
def write_csv(bb): # createcsv filetosavethedata
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 tosql 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, "recordinserted.")
whileTrue:

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

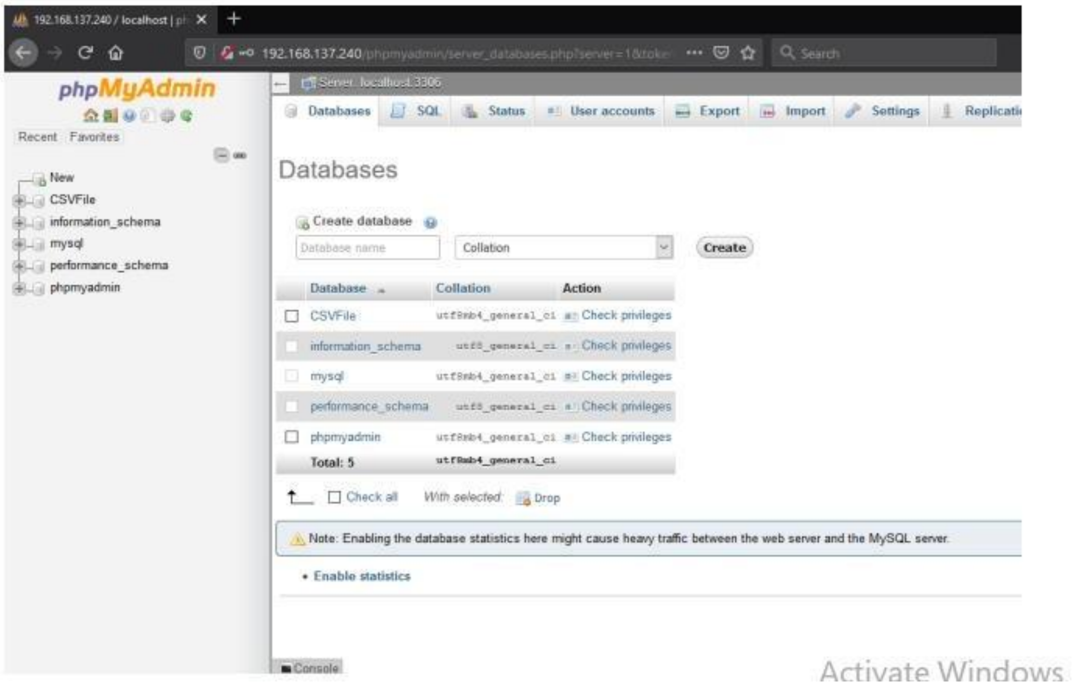
```
dbMsg = one_passenger()
write_csv(one_passenger()) # writestocsv file
databaseConnection() # writestosql database
elif count >=15:
print(max_passenger())
dbMsg = max_passenger()
write_csv(max_passenger())
databaseConnection()
else:
print(more_passenger())
dbMsg = more_passenger()
write_csv(more_passenger())
databaseConnection()
else:
if 0==GPIO.input(passengerCountOut): #whenpassenger 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 fig


192.168.137.240/phpmyadmin/server_databases.php?server=1&token

Server: localhost:3306

Databases | SQL | Status | User accounts | Export | Import | Settings | Replicati

Databases

Create database

Database name | Collation | Create

	Database	Collation	Action
☐	CSVFile	utf8mb4_general_ci	Check privileges
☐	information_schema	utf8_general_ci	Check privileges
☐	mysql	utf8mb4_general_ci	Check privileges
☐	performance_schema	utf8_general_ci	Check privileges
☐	phpmyadmin	utf8mb4_general_ci	Check privileges
	Total: 5	utf8mb4_general_ci	

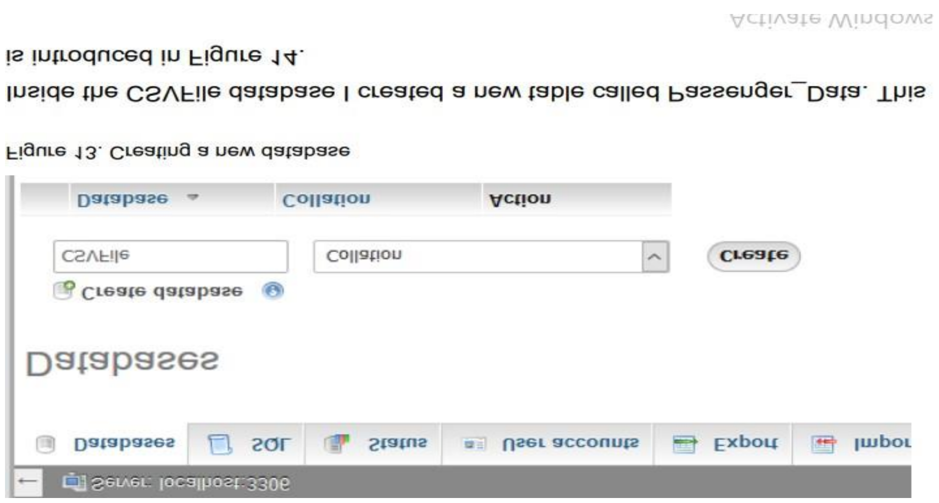
☐ Check all With selected: Drop

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

- Enable statistics

PHPMyAdmin server

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

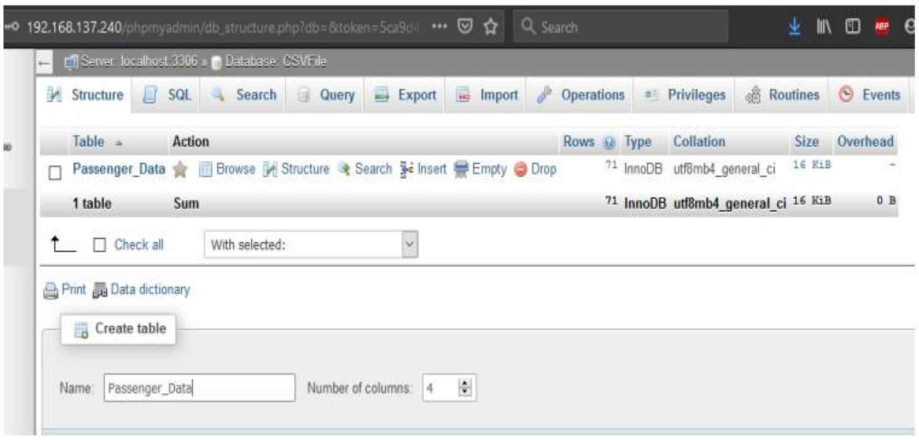

is introduced in Figure 14.

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



Server: localhost:3306 » Database: CSVFile

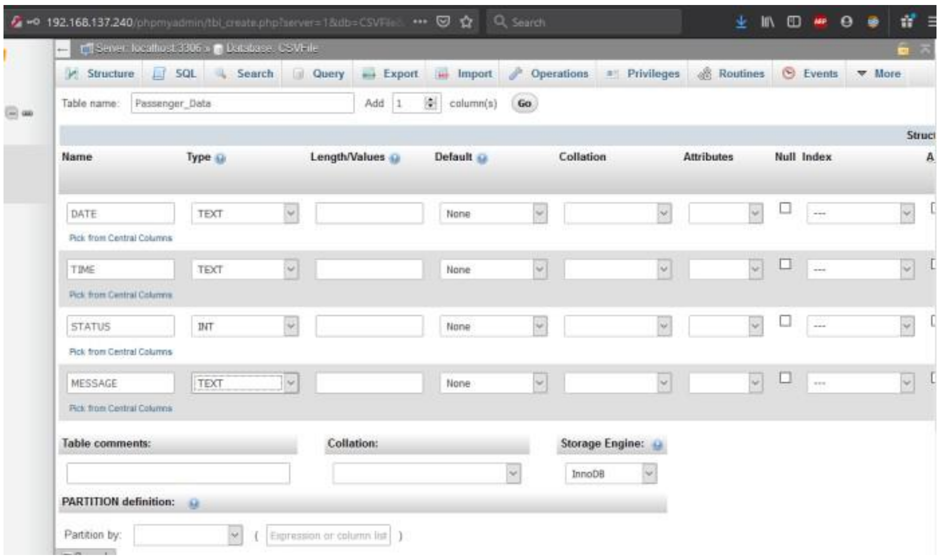
Table	Action	Rows	Type	Collation	Size	Overhead
Passenger_Data	Browse Structure Search Insert Empty Drop	71	InnoDB	utf8mb4_general_ci	16 KiB	-
1 table	Sum	71	InnoDB	utf8mb4_general_ci	16 KiB	0 B

Create table

Name: Passenger_Data Number of columns: 4

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



Server: localhost:3306 » Database: CSVFile

Table name: Passenger_Data

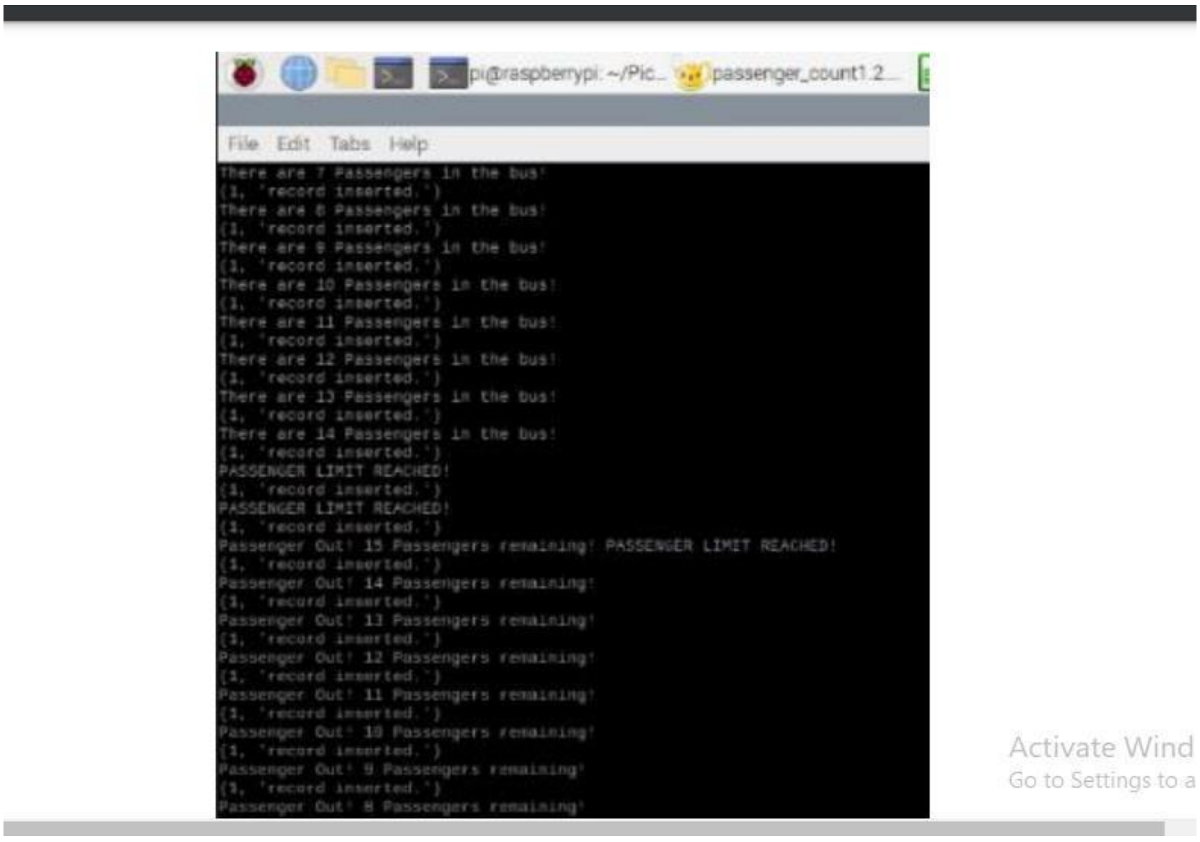
Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
DATE	TEXT		None			☐	---
TIME	TEXT		None			☐	---
STATUS	INT		None			☐	---
MESSAGE	TEXT		None			☐	---

Storage Engine: InnoDB

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.

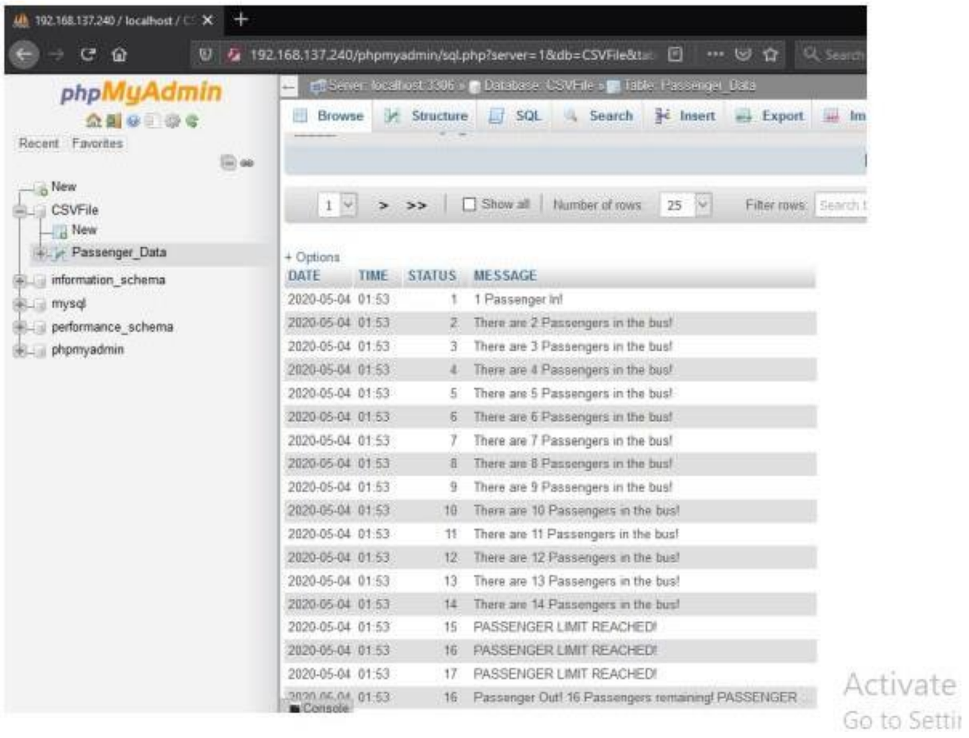
After successfully uploading the data to the database it prints out a message “1, ‘record inserted.’, as shown in Figure



```
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!
```

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



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:

$sudomysql-uroot-p

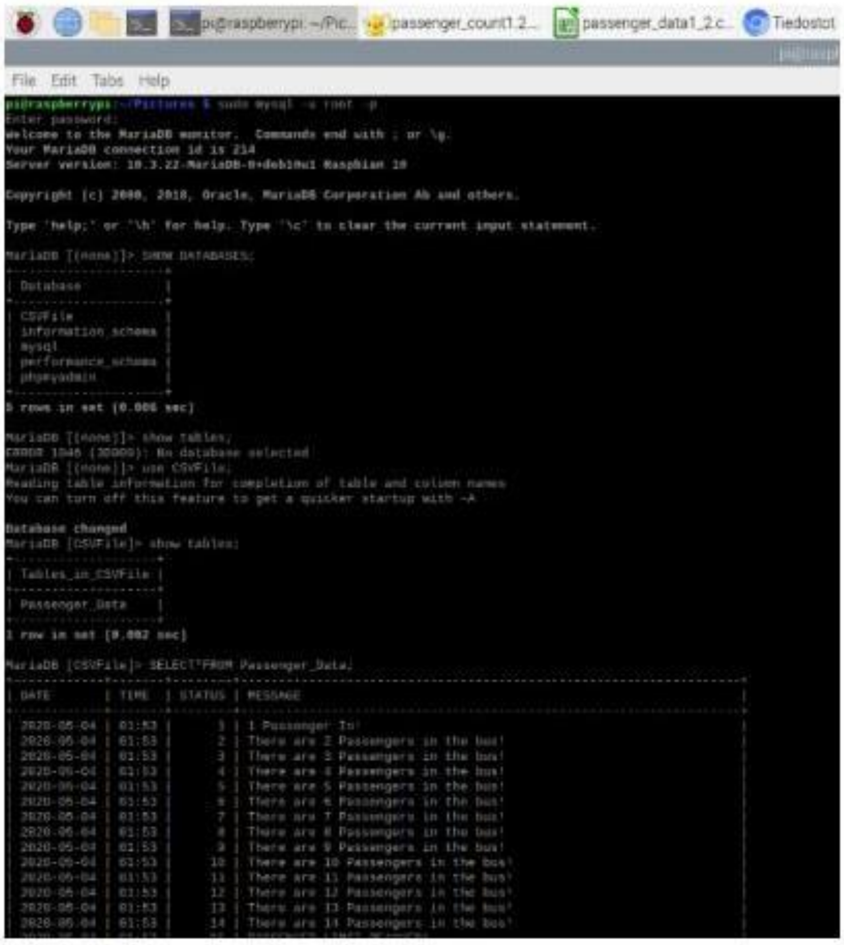
$showdatabases;

$useCSVFile;

$showtables;

$SELECT*FROMPassenger_Data;

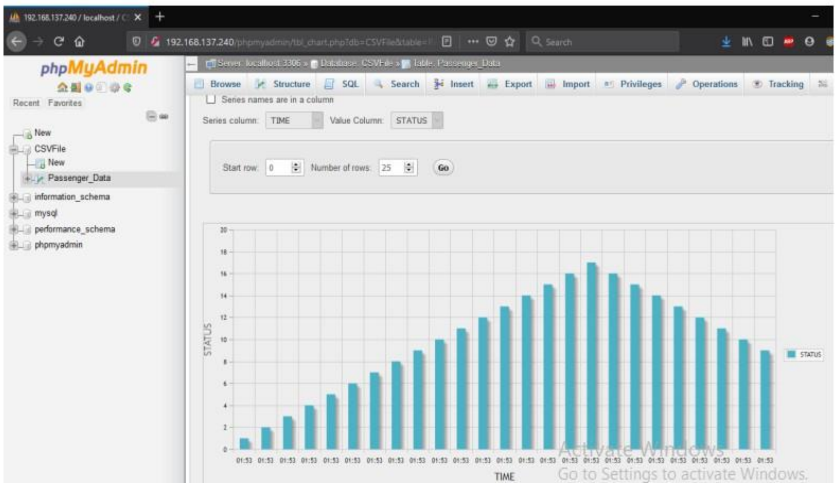
The output of these commands is shown in Figure



```
pi@raspberrypi:~/Pictures $ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 214
Server version: 10.3.22-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| CSVFile            |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
+--------------------+
5 rows in set (0.006 sec)

MariaDB [(none)]> show tables;
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> use CSVFile;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [CSVFile]> show tables;
+-------------------+
| Tables_in_CSVFile |
+-------------------+
| Passenger_Data    |
+-------------------+
1 row in set (0.002 sec)

MariaDB [CSVFile]> SELECT*FROM Passenger_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!               |
```

. **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.