

Multiple Motion Control of Robotic Car based on Internet of Things

Swati Jadhao^{1}, R. J. Bhiwani²*

¹*Department of Electronics and Telecommunication Engineering, Babasaheb Naik college of Engineering, Pusaad Maharashtra, India*

²*Guide, Department of Electronics and Telecommunication Engineering, Babasaheb Naik college of Engineering, Pusaad Maharashtra, India*

***Corresponding Author**

E-Mail Id: swatikjadhao@gmail.com

ABSTRACT

Control is an interesting field that has burst with new technologies, bringing the Internet of Things (IoT) concept to life. This paper presents a multiple motion control method for a robotic car that uses the Raspberry Pi. The controlling software uniquely identifies each device, which is the essential principle of IoT. The client can control the car's operations from afar through the internet using voice commands and a Universal Windows application, as well as receive data and feedback. The key contribution of this work is that it takes use of the robot's motion control system's efficiency by allowing the robotic car to receive direct commands from numerous sources at the same time, making the navigating system more efficient. It is not necessary for both the device and the client to be online at the same time. The commands and data are saved in the cloud and delivered when the device is ready to accept them. The car is equipped with a GPS system, allowing customers to track it. For avoiding impediments in its route, the device features an IR sensor. We show how to drive the car using commands and applications, as well as the architecture and design of the Raspberry Pi and connection software.

Keywords: *Raspberry Pi, GPS, motion control, cloud service, IoT, Pi camera module*

INTRODUCTION

An automation is preferable because it has fewer constraints, is more accurate, and is more dependable. These systems are distinguished by their controlling mechanism. Users perceive a collection of independent computers as a single controlling system while using a multiple control system.

To regulate distributed operations, it employs decentralized parts or subsystems. When employed in an industrial setting, they provide flexibility, extended equipment life, ease of new equipment integration, and centralised maintenance. Several complex robot control systems have been constructed using current control techniques or novel

control techniques designed for specific objectives. As a result, the multiple control mechanism is more than a requirement for efficient and flexible processing. The accessibility and availability of low-cost single-board computers the size of a credit card, such as the Raspberry Pi, has enabled the development of a slew of automated and controlled systems with low power consumption and faster processing capabilities at a lower cost. This work proposes a multifunctional control system for robots that incorporates the use of low-cost instruments, connectivity, wireless communication, and the efficiency of the controlling mechanism.

LITERATURE REVIEW OR BACKGROUND

This section gives a rundown of some of the approaches that have been created and tested for using Raspberry Pi and Arduino to control robots and devices. Vladimir Vujovic, *et al.* proposed a Raspberry Pi home automation system in which the Raspberry Pi serves as a sensor web node for controlling home appliances, making it the ideal platform for communicating with a wide range of devices. In this case, the Raspberry Pi serves as both a sensor and a controller [1].

However, the controlling mechanism is limited to data gathering and updating and only works in a controlled setting.

A Raspberry Pi-based home automation system is described in a Raspberry Pi based home automation system through Email [2]. This paper's contribution is that Raspberry Pi can read out user commands via E-mail, and the devices to be controlled are interfaced with Raspberry Pi via relay driver [2].

Clients, on the other hand, can only control the appliance's switching state; no other control system is included.

Jaroslav Sobota, *et al.* [3] present an extremely low-cost and adaptable control platform based on Raspberry Pi and Arduino, which runs the REX control system, an open system for embedded control [3].

The REX platform, on the other hand, isn't standard enough and can't control a huge number of devices at once. Another real-time monitoring system has been developed utilizing Raspberry Pi and Arduino in the development of a fire alarm system [4].

This paper explains how the Raspberry Pi uses sensors to control the situation. It does not, however, include any user-controlled interactions and is solely a sensor-based module.

In a research study [7], Anita Sabo *et al.* described a robotic arm control mechanism using Raspberry Pi and the internet. Despite its benefits, it does have certain drawbacks. It solely uses a web service to control the device, and the client is unable to determine its location. Furthermore, because there is no feedback system, the client has no way of knowing whether the command was executed correctly, which is a must-have feature in any system connected to the internet.

The above-mentioned challenges can be solved by developing a multiple controlling system that allows clients to operate robots from a distance using voice commands and a client application over the internet. Here, a wireless connection is taken into account. The motion control system of a robotic car is examined in this research.

Move forward, backward, turn left, right, rotate left, rotate right, activate obstacle detection, and disable obstacle detection are the first commands. Voice commands and/or user applications can be used to issue these orders. The car can be tracked in the UI at all times, and feedback and statistics about it may be obtained.[5,6]

The ultrasonic distance sensor also assists the robot in avoiding collisions with items that come in its way.

PROPOSED METHODOLOGY

In this section the system workflow described in details. The working procedure is divided into eight major parts and they are as follows Step by step detail work flow of proposed System:

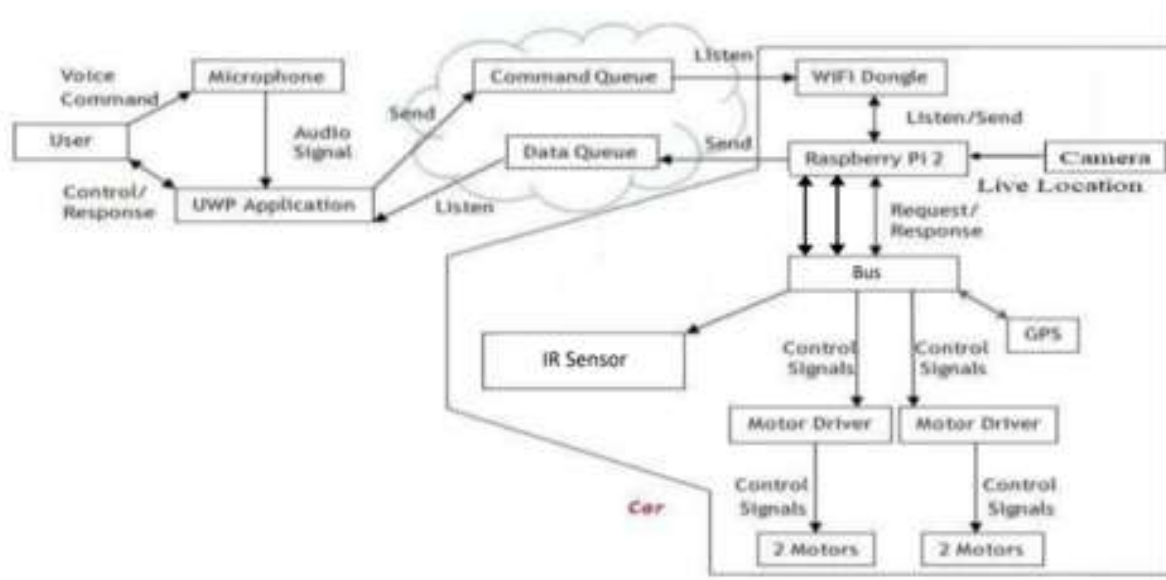


Fig. 1: System Architecture.

Sending Command

Once the system starts Voice command or manual clicking of buttons visible in the user interface are the two ways to deliver commands to the car. A XML grammar file lists all of the conceivable phrases or commands that users might say. Users can also control the car straight through the UWP application's UI and transmit whatever command they want, just like before.

Checks for Command Validation

A Speech Recognizer object is constructed in order to recognize speech. After proper processing, an XML grammar file is supplied to that object, which utilizes it to decode speech from the signal. The designated event handlers execute the rest of the task after successful decoding. However, if decoding fails, the client is asked to generate any command from a list of permitted commands. This request is actually a message that appears on the application's user interface. [8]

Stores Commands in a Cloud Service

The UWP app saves commands in a cloud service provided by Thingier.io Queue. This system's queue provides a well-defined and adaptable service. Because

both automobile information and commands required to be delivered at the same time to the desired locations or devices, two queues were used: one for data and another for commands. The Raspberry Pi in the car listens for commands and transmits information to the Data Queue. The UWP application on the controller end, on the other hand, listens to the Data Queue and delivers commands to the Command Queue. Because both end system components do not need to be linked to the Thingier.io at the same time, the system's performance is ensured.

Raspberry Pi Collects the Command

Every 0.5 seconds, the Raspberry Pi checks for commands and retrieves them from the cloud service's command queue. The Raspberry Pi connect with each other via the I2C (Inter Integrated Circuit) communication protocol. To identify them, they each have a different unique address. The Raspberry Pi gets three different types of command signals from the 1) GPS sensor values acquired from the GPS, 2) data received from the obstacle detector, and 3) manipulate the car's direction of motion or state in

response to the Raspberry Pi's command signal.

Raspberry Pi Captures the Images and stream it over the internet

Once the image get capture from the camera, the images get stream over the internet by using Raspberry Pi Webcam Server. For streaming the images over the internet, firstly we need to do the setup of router by adding new virtual server by providing the proper Service port, IP address etc. After this setting all the traffic will route towards the destined port to the webcam server located at the IP address and port which is provided.

Raspberry Pi Takes Action According to the Command

Raspberry Pi responds according to the command it receives. For example, obtaining the value of the GPS sensor, obtaining the reading of the obstacle distance sensor, and manipulating the car's direction of motion or state. The GPS sensor continuously pings to determine the car's current location. The IR sensor is additionally pinged by Raspberry Pi to detect the obstacle ahead of the car. Raspberry Pi uses the motor controllers to adjust the direction and speed of the motors based on the commands.

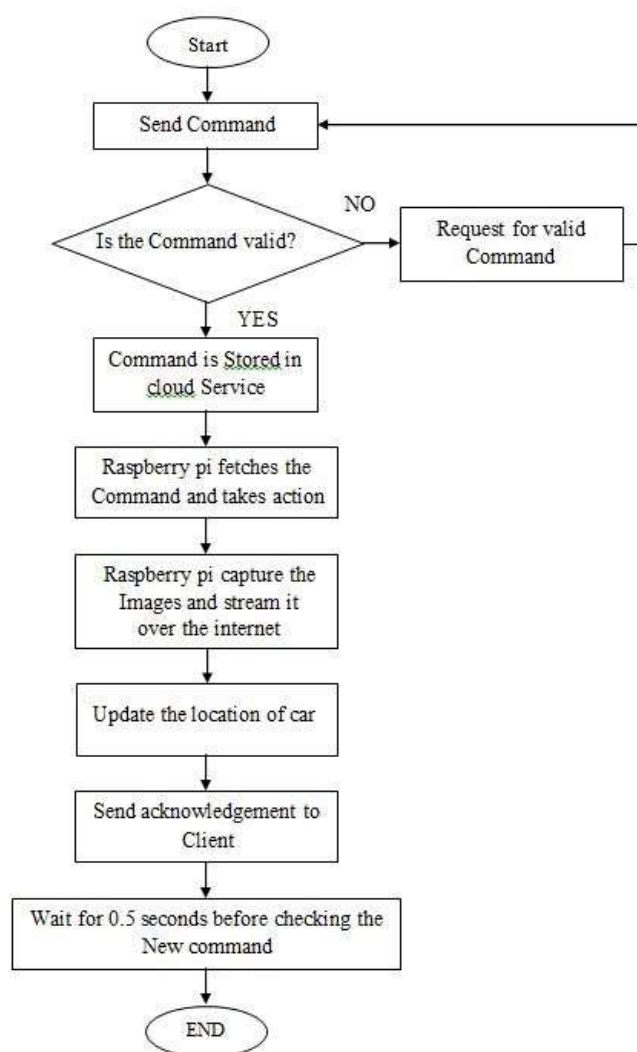


Fig. 2: Workflow of the Proposed System.

Updates GPS Position of the Car

When the Robotic Car is instructed to alter its position, the Raspberry Pi polls the GPS sensor to receive the most up-to-date GPS position, and when it is commanded to communicate the GPS position, it is sent to the Cloud Service Bus Data queue. The UWP application receives this data later and adjusts the UI accordingly.

Sends Acknowledgement to User

Whatever command is given to the car, it will respond with a precise response that either indicates that the command has been completed or that it has not been fulfilled. The acknowledgement is usually represented as a 1 or a 0. This is a critical feature in systems like this that are built on the Internet of Things paradigm, because without it, the client will be unable to determine whether the system has worked as expected.

SOFTWARE AND HARDWARE DESCRIPTION**What is Thinger.io?**

Thinger.io is a cloud IoT Platform that provides every needed tool to prototype, scale and manage connected products in a very simple way. Our goal is to democratize the use of IoT making it accessible to the whole world, and streamlining the development of big IoT projects.

Free IoT platform: Thinger.io provides a lifetime freemium account with only few limitations to start learning and prototyping when your product becomes ready to scale, you can deploy a Premium Server with full capacities within minutes.

Simple but powerful: Just a couple code lines to connect a device and start retrieving data or controlling its functionalities with our web-based Console, able to connect and manage thousands of devices in a simple way.

Hardware agnostic: Any device from any manufacturer can be easily integrated with Thinger.io's infrastructure.

Extremely scalable & efficient infrastructure: Thanks to our unique communication paradigm, in which the IoT server subscribes device resources to retrieve data only when it is necessary, a single Thinger.io instance is able to manage thousands of IoT devices with low computational load, bandwidth and latencies.

Open-Source: Most of the platform modules, libraries and APP source code are available in our Github repository to be downloaded and modified with MIT license.

Thinger.io Main Features

Thinger.io platform is formed by two main products a Backend (which is the actual IoT server) and a web-based Frontend that simplifies working with all the features using any computer or smartphone. The image below shows the main features provides by this platform to create IoT projects.

Every kind of device, no matter the processor, the network or the manufacturer. Thinger.io allows to create bidirectional communications with Linux, Arduino, Raspberry Pi, and even with edge technologies like Sigfox or LoRaWAN or other internet API data resources.

Store Device Data: Just a couple clicks to create a Data Bucket a store IoT data in a scalable, efficient and affordable way that also allows real-time data aggregation.

Display Real-time or Stored Data in multiple widgets such as time series, donut charts, gauges, or even custom

made representations to create awesome dashboards within minutes.

Trigger events and data values using an embedded Node-RED rule engine

Extend with custom features with multiple plugins to integrate IoT projects into your company's software or any other third party Internet service.

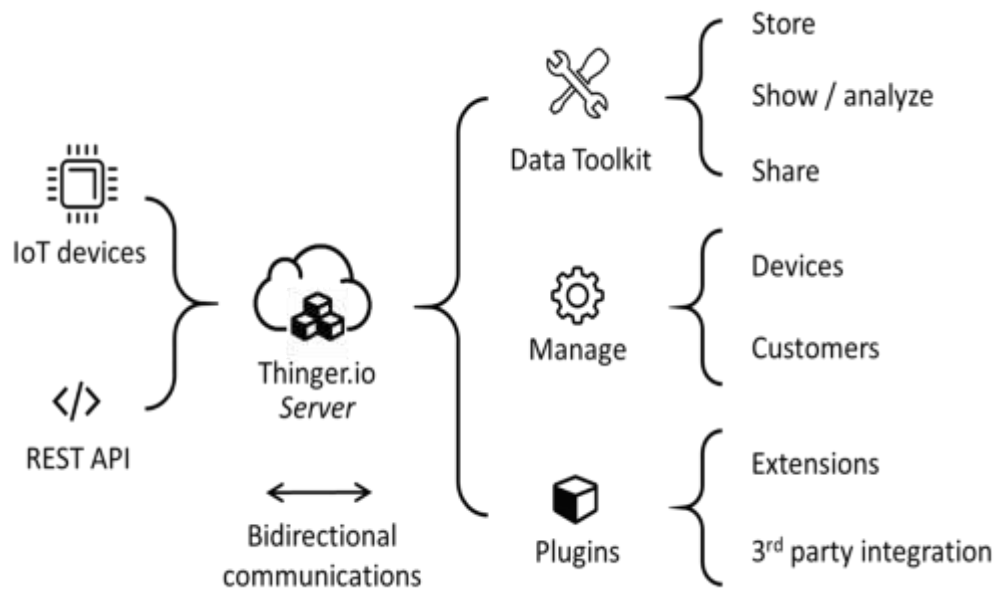


Fig. 3: Thingier.io system.

Connect devices: Fully compatible with

Control a DC Motor with an L298 Controller and Raspberry Pi

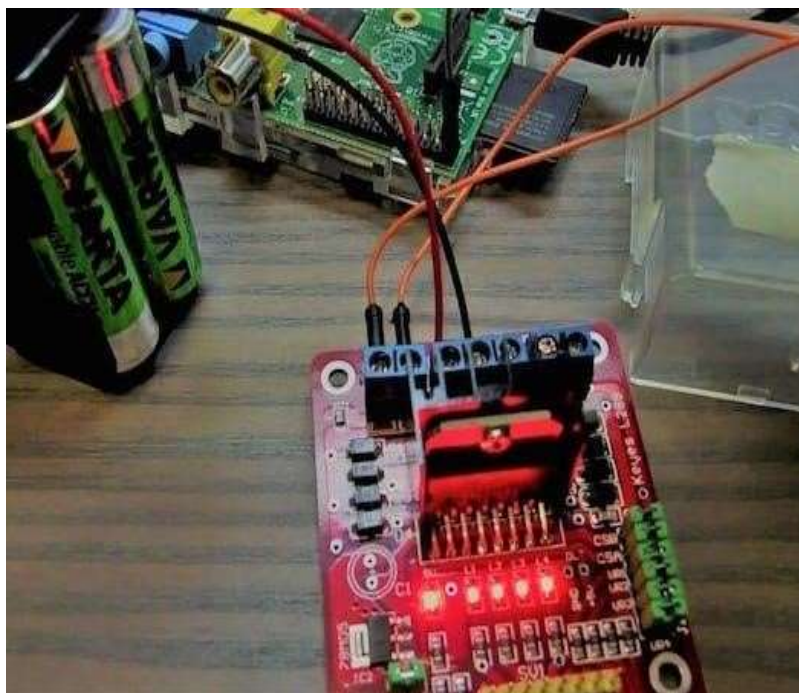


Fig. 4: DC Motor Controller.

Raspberry Pi 3 Model B**What is an L298?**

L298 is known as a dual bidirectional motor driver which is based on dual H-Bridge Motor driver IC. This circuit allows you to control two DC motors independently in either direction.

It is a commonly used component for prototypes and hobbyist projects, as it is easy to use and interface the L298 with a Raspberry Pi or an Arduino. Other than its minimal design, it also provides an onboard 5V regulator that you can use to power your 5V circuits very conveniently.

There are many L298 based motor driver modules out there in the market and you can use any of the locally available L298 based motor drivers because they all are essentially the same. I used an L298 breakout board, which makes setup a little easier.

Required materials

- Raspberry Pi (Any Pi should work)

- LM298 motor driver or LM298 breakout board
- DC Motor
- 9V Battery

Connecting an L298 with a Raspberry Pi

Controlling a DC Motor is easy with a Raspberry Pi. We use an L298 motor driver to control the DC motor, which allows the motor to move forward or backward.

For communication, we will use a simple serial communication over USB cable.

- Connect IN1 on the L298 to the Raspberry Pi's pin number 26.
- Connect IN2 on the L298 to Raspberry Pi's pin number 20.
- Connect the ENA and 12-volt pin to a 9-volt battery.

Make sure the grounds of the battery, Raspberry Pi, and L298 are common.

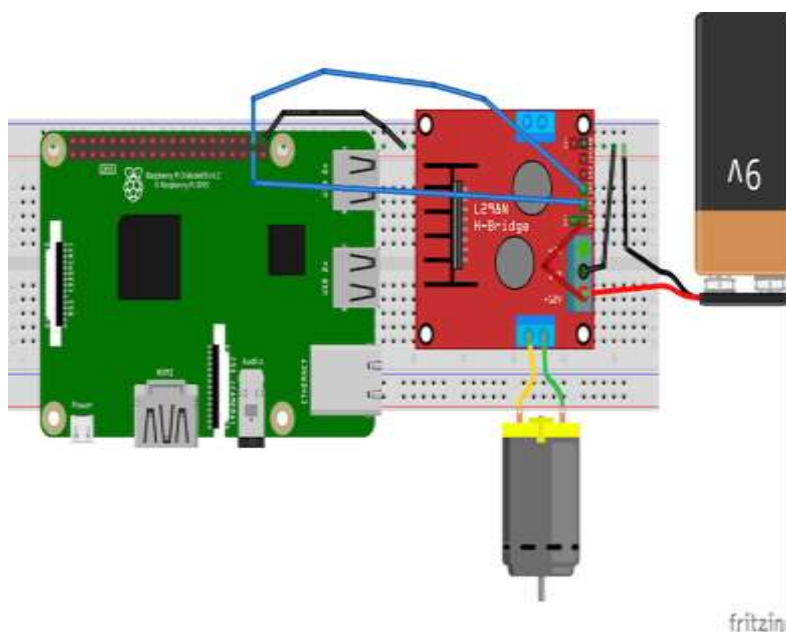


Fig. 5: Interfacing L298 with a Raspberry Pi

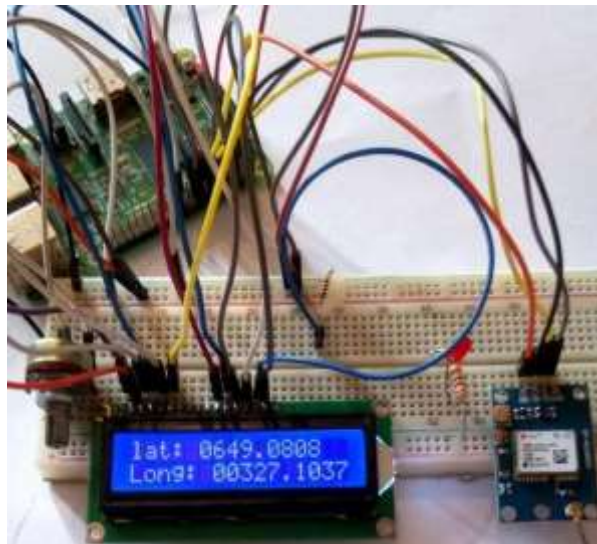


Fig. 6: Raspberry Pi 3 GPS Module Interfacing.

Raspberry Pi 3 GPS module interfacing tutorial

One of the coolest embedded platforms like the Arduino has given makers and DIYers the ability to get location data easily using GPS module and thus build things that rely on location. With the amount of power packed by the Raspberry Pi, it certainly will be quite awesome to build GPS based projects with the same cheap GPS modules. We will Interface GPS module with Raspberry Pi 3.

The goal of this project is to collect location data (longitude and latitude) via UART from a GPS module and display them on a 16x2 LCD.

Required Components

- Raspberry Pi 3
- Neo 6m v2 GPS Module
- 16 × 2 LCD
- Power source for the Raspberry Pi
- LAN cable to connect the pi to your PC in headless mode
- Breadboard and Jumper cables
- Resistor / potentiometer to the LCD
- Memory card 8 or 16Gb running Raspbian Jessie

Other than that we need to install GPS Daemon (GPSD) library, 16x2 LCD Adafruit library. Here we are using Raspberry Pi 3 with Raspbian Jessie OS.

GPS Module and Its Working:

GPS stands for Global Positioning System and used to detect the Latitude and Longitude of any location on the Earth, with exact UTC time (Universal Time Coordinated). GPS module is the main component in our vehicle tracking system project. This device receives the coordinates from the satellite for each and every second, with time and date.

GPS module sends the data related to tracking position in real time, and it sends so many data in NMEA format. NMEA format consist several sentences, in which we only need one sentence. This sentence starts from \$GPGGA and contains the coordinates, time and other useful information. This GPGGA is referred to Global Positioning System Fix Data. Know more about Reading GPS data and its strings here.

We can extract coordinate from \$GPGGA string by counting the commas in the string. Suppose you find \$GPGGA string and stores it in an array, then

Latitude can be found after two commas and Longitude can be found after four commas. Now these latitude and longitude can be put in other arrays.

Preparing the Raspberry Pi to communicate with GPS

The first thing we have to do to get this project underway is to prepare our Raspberry Pi 3 to be able to communicate with the GPS module via UART. Here we have used Neo 6m v2 GPS Module.

To dive in, here's a little explanation of How the Raspberry Pi 3 UART Works. The Raspberry Pi has two

built-in UARTs, a PL011 and a mini UART. They are implemented using different hardware blocks so they have slightly different characteristics. On the raspberry pi 3 however, the wireless/Bluetooth module is connected to the PLO11 UART, while the mini UART is used for the Linux console output.

Connections for Raspberry Pi GPS module interfacing

Connect the GPS Module and LCD to the Raspberry Pi as shown in the Circuit Diagram below.

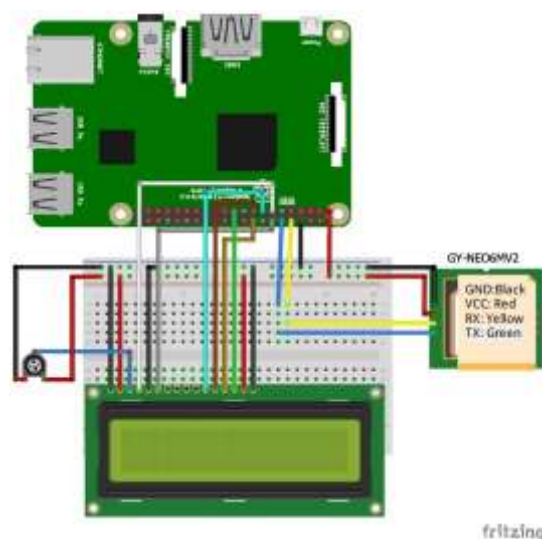


Fig. 7: Connections for Raspberry Pi GPS module Interfacing

9600 is the baud rate at which the GPS module communicates. This may be used for once we are sure of data communication between the GPS and the RPI.

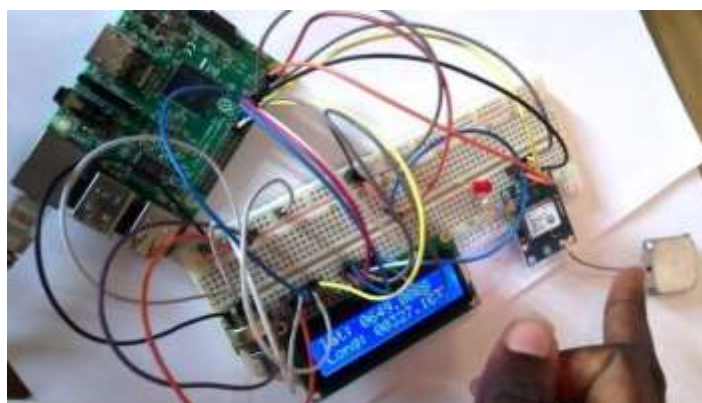


Fig. 8: The Location in latitude and longitude on LCD using GPS and Raspberry Pi.

Distance Sensor**Interfacing IR sensor with Raspberry Pi (proximity sensor – obstacle detector)**

Here we will learn about Infrared Sensors, simply known as IR Sensor and how to interface an IR Sensor with Raspberry Pi. By interfacing this IR Sensor with Raspberry Pi, you can implement a Proximity Sensor Application (Obstacle Detection).

Overview

Infrared Sensors or IR Sensors are one of the frequently used sensor modules by electronics hobbyists and makers. They are often used as Obstacle Detecting Sensors or Proximity Sensors.

IR Sensors are also used in Contactless Digital Tachometers. Some of the other applications where IR Sensors are implemented are Line Follower Robots, Obstacle Avoiding Robots, Edge Avoiding Robots and many more.

Brief Note on IR Sensor (IR Proximity Sensor)

IR Sensors emit and receive Infrared radiation. They are often used as Proximity Sensors i.e. detect and alarm if an object is close to the sensor.

Let me help you understand better about IR Sensors by giving two real life applications of IR Sensors. The first one is Mobile Phones.

Almost all mobile phones nowadays have IR Sensors in them. Usually, they will be placed near the earpiece on the phone.

When the user make or receives a phone call, the IR Sensor detects how far the phone is from the user's ear. If it is close to the ear, the phone's display will be turned off so that you do not touch anything on the screen accidentally.

Another important application is in automobiles. All modern cars are equipped with reverse parking sensor that sense how far you can reverse your car without hitting anything. These reverse sensors are implemented using IR Sensors.

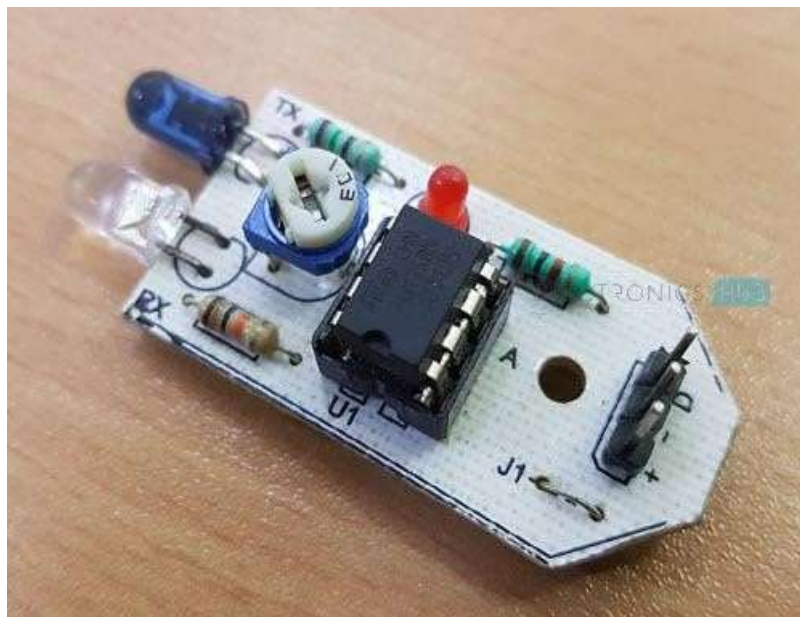


Fig. 9: IR Module.

An IR Sensor Module basically consists of three parts: an IR Transmitter, an IR Detector and a control circuit.

Usually, an IR LED is used as an IR Transmitter and a Photo Diode or a Photo Transistor (less often) is used as an IR Detector. The control circuit consists of a Comparator IC with necessary components.

Based on the application and requirement, IR Sensors can be implemented in two ways. In the first method, both the IR Transmitter and the IR Detector are placed side-by-side. In the second setup, the IR Transmitter and the IR Detector are placed facing each other.

The first way of implementation is known as Reflective Type IR Sensor. In this setup, the IR Transmitter continuously emits infrared light and if there is any obstacle/object in front of the sensor, the infrared light hits the object and bounces back. The reflected signal is captured by the IR Detector and the control circuit will reflect a Logic HIGH on its output.

The second way of implementation, where the IR Transmitter and Detector are positioned face-t-face, is known as Transmissive Type IR Sensor. Here, the infrared light from the IR Transmitter always falls on the Detector.

If there is an object in between the Transmitter and Detector, then there will be an obstruction to the infrared light and the control circuit will detect this and produces appropriate output.

The IR Sensor used in this project is a Reflective Type IR Sensor. You can easily build this type of IR Sensor as a DIY Project as the circuit is very simple.

Schematic of IR Sensor Module

The following image shows the circuit diagram of the IR Sensor Module. It consists of the following components.

- IR LED
- Photo Diode
- 150 Ω Resistor
- 10 K Ω Resistor
- 10 K Ω Potentiometer
- LM358
- LED
- 1 K Ω Resistor

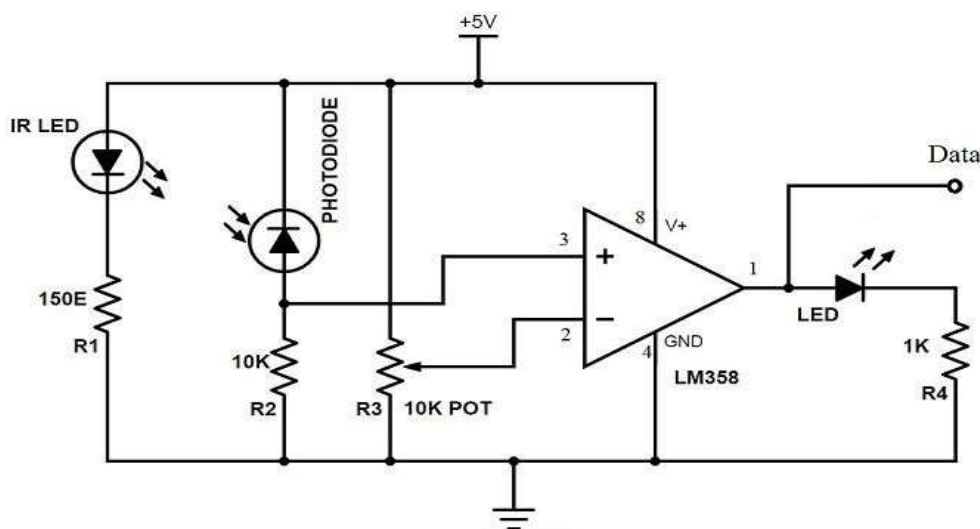


Fig. 10: Schematic of IR Module.

Raspberry Pi IR Sensor Interface

Now that we have seen a little bit about the IR Sensor Module and its connections, we will proceed with interfacing IR Sensor with Raspberry Pi. The Raspberry Pi IR Sensor Interface can be converted into a Proximity

Detector, where the application will detect if the object is too close to the sensor.

Circuit Diagram

The following image shows the connection diagram of Interfacing IR Sensor with Raspberry Pi. You have already seen the circuit diagram of the IR Sensor Module.

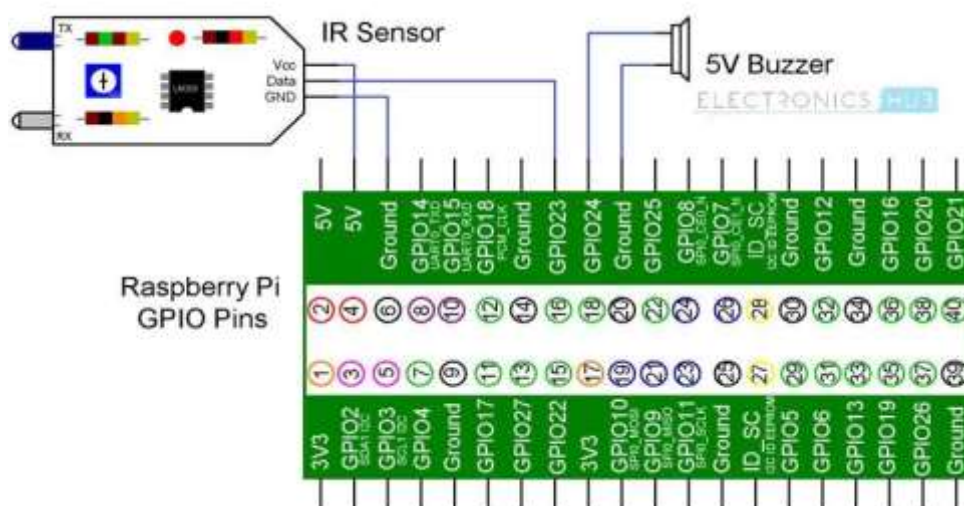


Fig. 11: Circuit Diagram IR Sensor.

Components Required

- Raspberry Pi 3 Model B
- IR Sensor
- 5V Buzzer
- Mini Breadboard
- Connecting Wires
- Power Supply
- Computer

Circuit Design

The IR Sensor Module has only three Pins: VCC, GND and Data. Connect the VCC and GND pins of the IR Sensor to +5V and GND pins of the Raspberry Pi. Then connect the Data pin of the IR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi.

In order to indicate the alarm, I have used a simple 5V Buzzer. Connect one terminal of the buzzer to GND of Raspberry Pi and the other terminal

(usually marked +) to GPIO24 i.e. Physical Pin 18 of Raspberry Pi.

Working

We have learned how to interface an IR Sensor with Raspberry Pi. I'll now explain the working of the project.

All the magic happens in the IR Sensor Module. As it is a Reflective type IR Sensor, whenever an object is placed in front of the sensor, the Infrared light from the IR LED gets reflected back after hitting the object and falls on the Photo Diode.

The photo diode then starts conducting. As a result, the voltage at the noninverting input of the LM358 will be greater than that at the inverting input. Since the LM358 is acting as a Comparator, its output will become

HIGH and the on-board LED glows. The HIGH on the Data Pin is detected by the Raspberry Pi and it activates the buzzer.

Using the 10 K Ω Potentiometer, you can adjust how far the object can be placed in front of the sensor in order to detect.

Applications

As mentioned in the earlier sections, Proximity Sensor or Obstacle Detection is the main application of interfacing IR Sensor with Raspberry Pi. Some of the common applications include

Contactless Tachometer
Line Follower Robot
Obstacle Avoiding Robot
Car Reverse Sensor
Mobile Proximity Sensor

Working of I2C Communication Protocol

What is an I2C?

An Inter-Integrated Circuit (I2C) or Two-Wire Interface (TWI) is a synchronous serial protocol originally developed by Philips Semiconductors (now

NXP). It's a multi-master, multi-slave serial bus for low-speed devices that only requires two wires for serial data

communication between multiple devices. It can easily be implemented with two digital input/output channels on a device. An I2C bus has just two wires over which hundreds of devices can communicate serial data.

As a master-slave type communication standard, at least one device connected to the bus should be the master. It's the master device that generates a clock signal for synchronous serial communication.

The slave devices can transfer data to and from the master device(s), which access slave devices by their I2C addresses. The address of each slave device on an I2C bus must be unique. However, the I2C slave devices still must obtain their addresses from NXP.

There are only two wires in I2C where devices can be connected:

1. Serial data (SDA) – the line over which the master and slave devices communicate serial data
2. Serial clock (SCK) – the line over which master device(s) generate the clock signal.

Writing the device

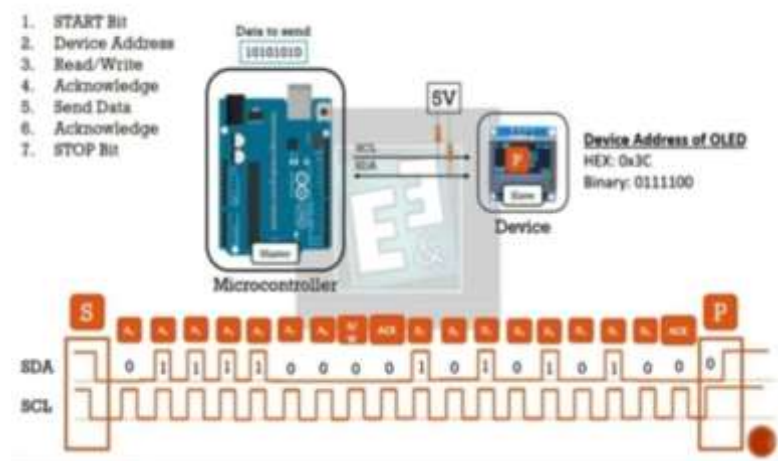


Fig. 12: Writing the Device.

Steps for writing to the device:

- Master send start bit 0 to slave.
- Master send device address to slave.
- Master send read/write bit to slave, 0 for write and 1 for read.
- Slave send acknowledge to master, 0 for busy and 1 for ready.
- Master send data to slave.
- Slave send acknowledge 0 to master.
- Master send stop bit 0 to slave.
- Then SCL and SDA both high again.

Reading the device

Steps for reading the device:

- Master send start bit 0 to slave.
- Master send device address to slave.
- Master send read/write bit to slave, 0 for write and 1 for read.
- Slave send acknowledge to master, 0 for busy and 1 for ready.
- Master receives data from slave.
- Master send acknowledge 1 to slave.
- Master send stop bit 0 to slave.

- Then SCL and SDA both goes high.

I2C in Raspberry Pi:

For serial communication over the I2C protocol, the Broadcom processor of Raspberry Pi has Broadcom Serial Controller (BSC). This standard-mode master BSC controller is NXP Semiconductor's I2C compliant and supports a data transfer rate of 400 kbps.

The BSC controller supports both 7-bit as well as 10-bit addressing. This I2C interface is accessible at pins GPIO2 (Board Pin No. 3) and GPIO3 (Board Pin No. 5). GPIO2 is Serial Data (SDA) line, and GPIO3 is a Serial Clock (SCL) line of the I2C1. These I2C pins are internally pulled up to 3.3V via 1.8 k ohms resistors. That is why these pins cannot be used for general-purpose I/O where pull-up is not required.

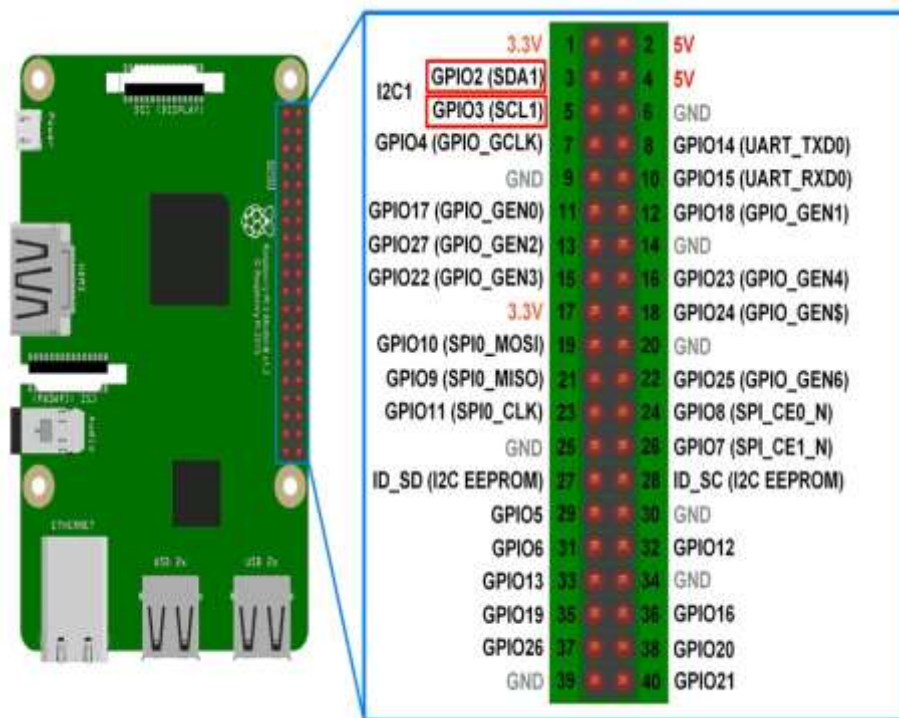


Fig. 13: I2C in Raspberry Pi.

Pin configuration of Raspberry Pi:

As shown in fig below we can connect the devices with raspberry pi.



A Raspberry Pi camera module is shown, featuring a green printed circuit board (PCB) with a black lens and a yellow micro-USB port. A white ribbon cable is attached to the module, with the text "1-25A 25 1-25A 25" printed on it. The module is labeled "Raspberry Pi Camera" and "Rev 1.3".

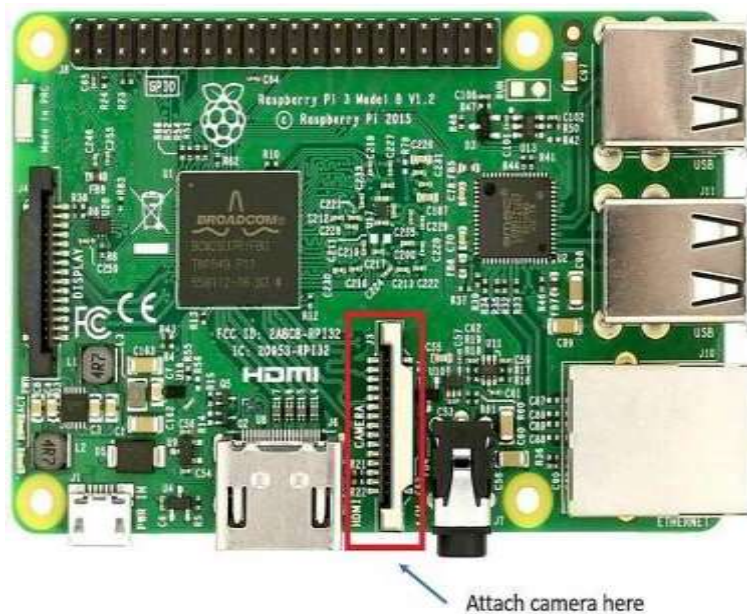
Fig. 15: Pi Camera Module (v1.3).

Pi Camera module is a camera which can be used to take pictures and high definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach Pi Camera module directly. This Pi Camera module can attach to the Raspberry Pi's CSI port using 15-pin ribbon cable.

Here, we have used Pi camera v1.3. Its features are listed below,

- HD Video recording –1080p @30fps, 720p @60fps, 960p @45fps and so on.
- It can capture wide, still (motionless) images of resolution 2592×1944 pixels
- CSI Interface enabled.

Connect Pi Camera to CSI interface of Raspberry Pi board as shown below,



Now, we can use Pi Camera for capturing images and videos using Raspberry Pi. Before using Pi Camera, we need to enable camera for its working.

How to enable camera functionality on RASPBERRY Pi

For enabling camera in Raspberry Pi, open raspberry pi configuration using

following command, `sudo raspi-config` then select **Interfacing options** in which select **camera** option to enable its functionality.

Reboot Raspberry Pi.

Now we can access camera on Raspberry Pi.

Now we can capture images and videos using Pi Camera on Raspberry Pi.

PERFORMANCE EVALUATION AND RESULT

Table 1: Performance Analysis.

Distance (m)	Time (ms)			
	Commands received by car in	Stored in cloud	Perform actions in	Client gets updates
10	5 ms	5 ms	8ms	11 ms
20	5 ms	5 ms	9 ms	12ms
35	5 ms	5 ms	8ms	11 ms
45	6 ms	6 ms	10ms	13 ms
60	6 ms	6 ms	10ms	12ms
95	6 ms	6 ms	8ms	11 ms
100	7 ms	7 ms	10ms	13 ms

For creating wireless communication, a wireless router with a broadband connection was used. The experiment was conducted in a normal environment with sound level ranging from 60dB to 80dB. From the experimental result it is seen that the signal of router reaches up to 100m. The results are based on all eight commands listed in the XML grammar file. Depending on the distance between the robotic car and the router, there are variations in time of receiving the commands. On an average it takes only 6 ms to receive a command. At the same time, commands are stored in the

cloud service. According to the mode of commands, the Arduino take actions and users get updated with an average timing of 9 ms and 12 ms respectively. But the system is crucially dependent on the performance of router, the broadband connection and the Wi-Fi dongle attached to the Raspberry Pi. The performance is comparatively better than previous research results of any motion controlling systems of robotic car. The working environment, surface of robotic car and signal availability are kept in consideration here.

Table 2: Differential Steering Method.

Left Motors	Right Motors	Outcome
Forward	Forward	Forward
Forward	Static	Left
Static	Forward	Right
Backward	Backward	Backward
Forward	Backward	Rotate Right
Backward	Forward	Rotate Left



Fig. 17: Picture of robotic car.

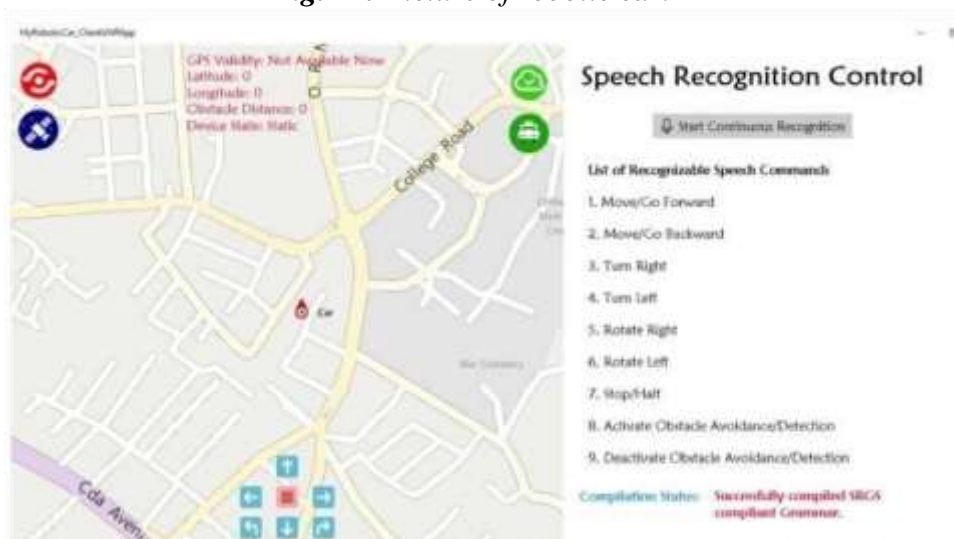


Fig. 18: UWP Application's User-Interface 750.

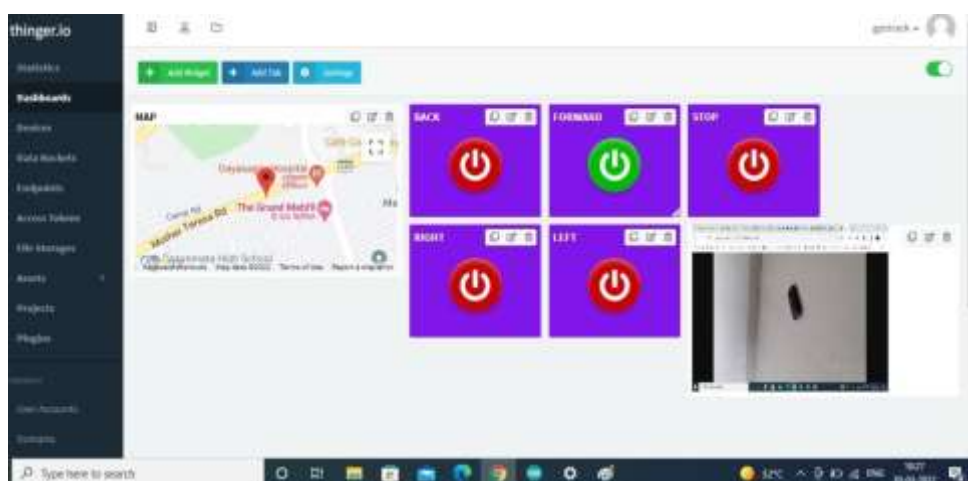


Fig. 19: Image get stream over the internet.

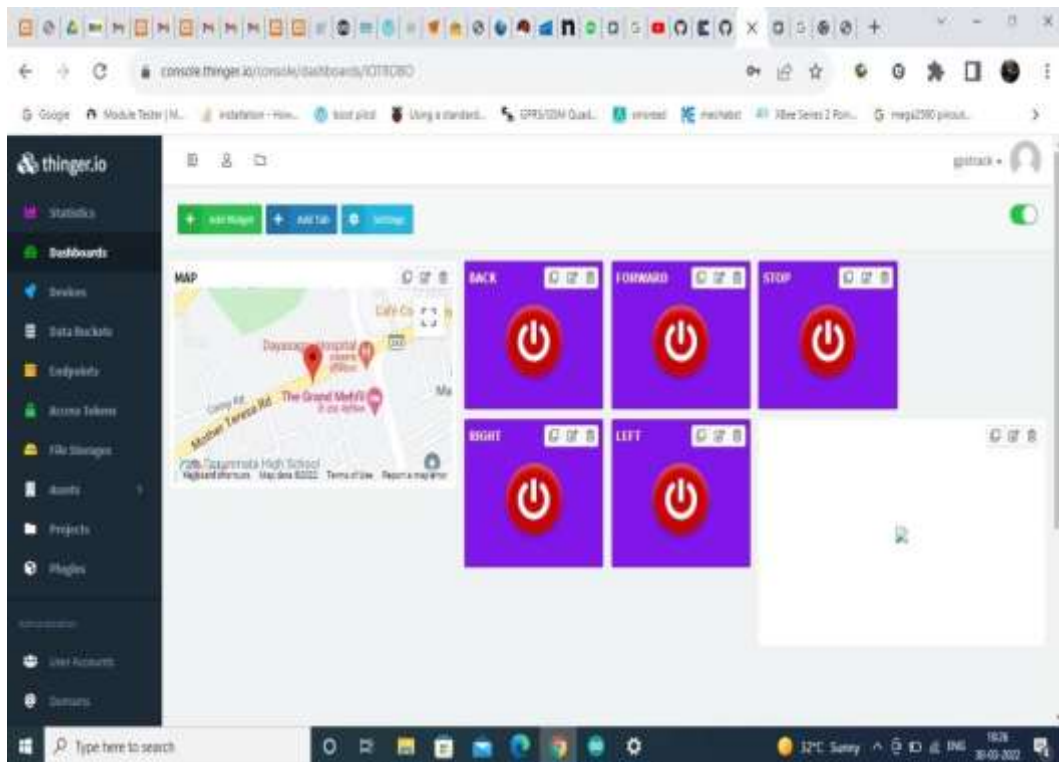


Fig. 20: UWP applications user interface.

ADVANTAGE & DISADVANTAGE

Advantages

- Client can manage the activities of car from remote or distant places over the internet by voice commands or manual clicking of buttons visible in the user interface and Universal window Applications and also able to get data and feedback.
- It leverages the efficiency of Robot motion controlling system because robotic car can receive direct commands at a time from multiple sources which make the maneuvering system more efficient.
- The cloud service helps the system to reduce memory load.
- IR sensor helps the robotic car to avoid collision with objects coming in between its path.
- Live video Streaming is possible by using Raspberry Pi which is not just a sensor node but a controller as well.

Disadvantages

Device and client need not have to be online at the same time, User can leave command and system will work by fetching the commands from the queue in FIFO i.e. First-in First-out order.

CONCLUSION

In this work, IoT is combined with an efficient technique of numerous control systems. Having the ability to control many devices in multiple ways makes it easier to manage a system. The cloud service assists the system in lowering memory usage.

After a given amount of time, stored messages are automatically deleted. Multiple controlling ways have less effect on time and performance than a single manner of control system, according to the performance data, if the incorporation is efficient enough.

REFERENCES

1. Vujović, V., & Maksimović, M. (2015). Raspberry Pi as a Sensor Web node for home automation. *Computers & Electrical Engineering*, 44, 153-171.
2. Jain, S., Vaibhav, A., & Goyal, L. (2014, February). Raspberry Pi based interactive home automation system through E-mail. In *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)* (pp. 277-280). IEEE.
3. Sobota, J., Pišl, R., Balda, P., & Schlegel, M. (2013). Raspberry Pi and Arduino boards in control education. *IFAC Proceedings Volumes*, 46(17), 7-12.
4. Bahrudin, M. S. B., Kassim, R. A., & Buniyamin, N. (2013, December). Development of fire alarm system using Raspberry Pi and Arduino Uno. In *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE)* (pp. 43-48). IEEE.
5. Krasovitskaya, K., Cherkashin, E., & Gorunchik, S. (2016). Expert System for Structural Analysis of Electrocardiograms. *On Applied Internet and Information Technologies*, 220.
6. Huang, X., & Deng, L. (2010). An Overview of Modern Speech Recognition. *Handbook of natural language processing*, 2, 339-366.
7. Krishna, R., Bala, G. S., ASC, S. S., Sarma, B. B. P., & Alla, G. S. (2012). Design and implementation of a robotic arm based on haptic technology. *Int. J. of Eng. Research and Applications*, 2(34).
8. Abdullah, A., Sidek, O., Amran, N. A., Za'Bah, U. N., Nikmat, F., Jafar, H., & Hadi, M. A. (2012, December). Development of wireless sensor network for monitoring global warming. In *2012 International Conference on Advanced Computer Science and Information Systems (ICACSIS)* (pp. 107-111). IEEE.