**Siddaganga Institute of Technology, Tumakur**

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi, Approved by AICTE, New Delhi, Accredited by NAAC and ISO 9001:2015 certified)

A Report on Project titled

“Smart Stick for Blind using Raspberry Pi”

***Submitted***

***in the partial fulfillment of the requirements for Industrial Training***

### BACHELOR OF ENGINEERING

in

### ELECTRONICS AND COMMUNICATION ENGINEERING

by

**Hanumesha 1SI21EC404**

**Sanketh S Govankop 1SI21EC408**

**Neha B 1SI20EC058**

**Hruthik Gowda G 1SI21EC405**

**Ravi kumar E 1SI20EC073**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**2021-22**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

## **Siddaganga Institute of Technology, Tumakuru**

*(An Autonomous Institution Affiliated to vishveshvaraya technological university, Belagavi , Approved by AICTE , NewDelhi*

*,Accredited by NAAC and ISO 9001:2015 certified)*



##### CERTIFICATE

This is to certify that project titled ” Smart Stick for Blind using Raspberry Pi” is a bonafide work carried out **by Hanumesha(1SI21EC404), Sanketh S Govankop(1SI21EC408), Neha B(1SI20EC058), Hruthik Gowda G(1SI21EC405), Ravi kumar E(1SI20EC073)** of IV semester **Bachelor of** **Engineering in Electronics and Communication** of the **SIDDAGANGA INSTITUTE OF TECHNOLOGY**(An Autonomous Institution, affiliated to VTU, Belagavi , Approved by AICTE, New Delhi, Accredited by NAAC and ISO 9001:2015 certified) during the academic year **2021-2022**.

Guided by

**Saikiran Singarao** Signature and date

**ABSTRACT**

A Smart stick system concept is devised to provide a smart electronic aid for blind people. Blind and visually impaired people find difficulties in detecting obstacles during walking in the street. The system is intended to provide object detection, real time assistance via GPS by making use of Raspberry Pi. The system consists of ultrasonic sensors, GPS module, and the feedback is received through ultrasonic sensors. The proposed system detects an object around them and sends feedback in the form of beep, warning messages via MQTT. The aim of the overall system is to provide a low cost and efficient obstacle detection aid for blind which gives a sense of artificial vision by providing information about the environmental scenario of static and dynamic object around them, so that they can walk independently.

INDEX

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **CONTENT** | **PAGE NO** |
| **1** | ABSTRACT | **3** |
| **2** | INTRODUCTION | **5** |
| **3** | SYSTEM OVERVIEW | **6** |
| **4** | RASPBERRY PI | **7** |
| **5** | ULTRASONIC SENSOR | **12** |
| **6** | GPS MODULE | **13** |
| **7** | MQTT PROTOCOL | **15** |
| **8** | SOFTWARE USED | **16** |
| **9** | METHODOLOGY | **17** |
| **10** | CODE | **18** |
| **11** | WORKING | **26** |
| **12** | RESULT | **27** |
| **13** | LAYOUT SPECIFICATION | **28** |
| **14** | ANALYSIS | **29** |
| **15** | CONCLUSION AND RECOMMENDATION | **30** |
| **16** | REFERENCE | **31** |

**INTRODUCTION**

Eye sight plays a major role in collecting most of the information from the real world and that information will be processed by brain, visually impaired people suffer inconveniences in their daily and social life. Blindness or visual impairment is a condition that affects many people around the world. This condition leads to the loss of the valuable sense of vision. Worldwide there are millions of people who are visually impaired, where many of them are blind. The need for assistive devices was and will be continuous. There is a wide range of navigation systems and tools existing for visually impaired individuals. The blind person truly requires an identifying object.

**OBJECTIVE**

The main objective of this is to provide an application for blind people to detect the obstacles in various directions, detecting pits and manholes on the ground to make free to walk. in an innovative stick is designed for the visually disabled people for their easy navigation.

**PROBLEM STATEMENT**

* Blind people can’t easily recognize obstacles or stairs while using normal blind stick .
* No safety features on the normal blind stick.
* Can’t locate the location of the normal blind stick user when they are having an emergency problem or lost in a public area.

**SYSTEM OVERVIEW**

POWER SUPPLY

BUZZER

MQTT SERVER

GPS MODULE

ULTRASONIC SENSOR

RASPBERRY PI

Hardware Requirements:

* Raspberry Pi
* Ultrasonic sensor
* GPS Module
* Buzzer
* Power supply

**RASPBERRY PI**

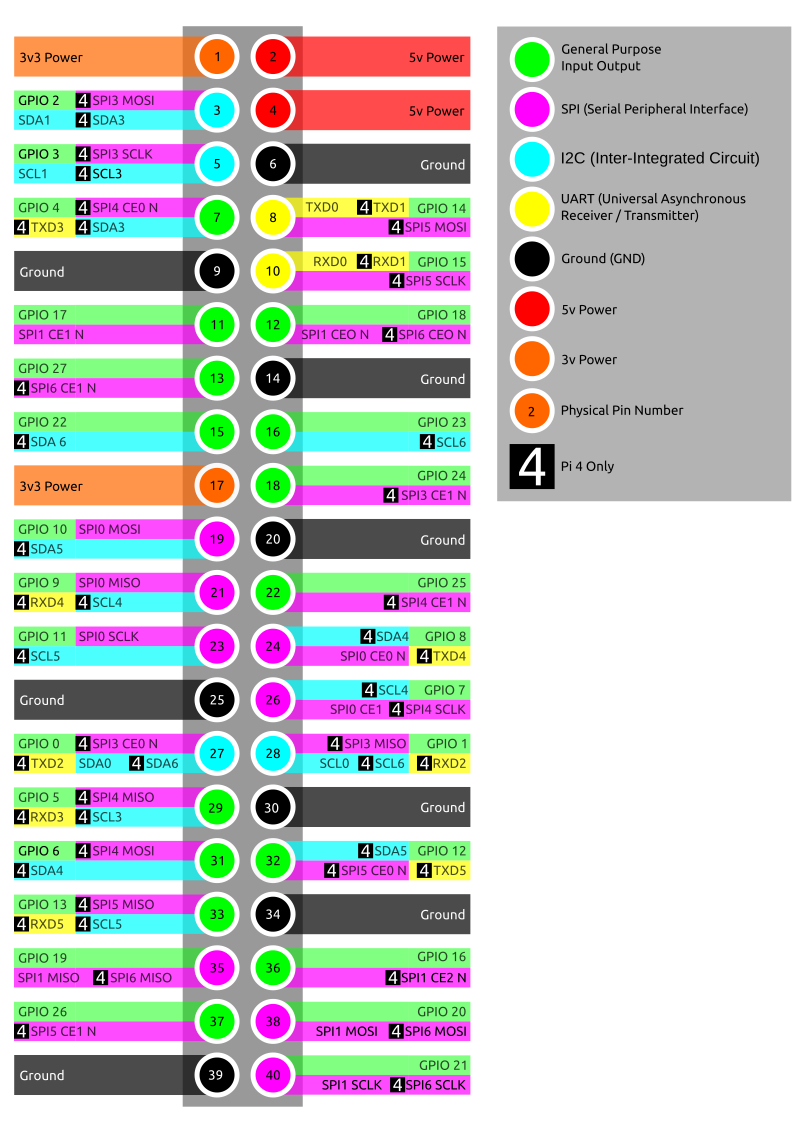
The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

****

* The first-generation **Raspberry Pi Model B** was released in February 2012, followed by the simpler and cheaper **Model A**.
* In 2014, the Foundation released a board with an improved design, **Raspberry Pi Model B+**. These first-generation boards feature [ARM11 processors](https://en.wikipedia.org/wiki/ARM11).
* The **Raspberry Pi 2** was released in February 2015 and initially featured a 900 MHz 32-bit quad-core [ARM Cortex-A7](https://en.wikipedia.org/wiki/ARM_Cortex-A7) processor with 1 GB RAM. Revision 1.2 featured a 900 MHz [64-bit](https://en.wikipedia.org/wiki/64-bit_computing) quad-core [ARM Cortex-A53](https://en.wikipedia.org/wiki/ARM_Cortex-A53) processor (the same as that in the Raspberry Pi 3 Model B, but underclocked to 900 MHz).
* The **Raspberry Pi 3 Model B** was released in February 2016 with a 1.2 GHz 64-bit [quad core](https://en.wikipedia.org/wiki/Multi-core_processor) [ARM Cortex-A53](https://en.wikipedia.org/wiki/ARM_Cortex-A53) processor, on-board [802.11n](https://en.wikipedia.org/wiki/802.11n) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi), [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth) and USB boot capabilities.
* Its CPU is 700Mhz single core ARM1176JZF-S,
* It has 4 USB ports
* It has dual core video core iv multimedia coprocessor
* Size of its RAM is 512mb
* It has micro SDHC plot for storage

Power rating of raspberry pi is 600mA i.e., 3.0W

## **General Purpose Input Output (GPIO) Pins**

****

The GPIO is the most basic, yet accessible aspect of the Raspberry Pi. GPIO pins are digital which means they can have two states, off or on. They can have a direction to receive or send current (input, output respectively) and we can control the state and direction of the pins using programming languages such as Python, JavaScript, node-RED etc.

The operating voltage of the GPIO pins is 3.3v with a maximum current draw of 16mA. This means that we can safely power one or two LEDs (Light Emitting Diodes) from a single GPIO pin, via a resistor (see [resistor color codes](https://www.tomshardware.com/how-to/resistor-color-codes)). But for anything requiring more current, a DC motor for example, we will need to use external components to ensure that we do not damage the GPIO.

Controlling a GPIO pin with Python is accomplished by first importing a library of pre-written code. The most common library is RPi.GPIO (https://pypi.org/project/RPi.GPIO/) and it has been used to create thousands of projects since the early days of the Raspberry Pi. In more recent times a new library called GPIO Zero (https://pypi.org/project/gpiozero/)has been introduced, offering an easier entry for those new to Python and basic electronics. Both of these libraries come pre-installed with the Raspbian operating system.

GPIO pins have multiple names; the first most obvious reference is their “physical” location on the GPIO. Starting at the top left of the GPIO, and by that we mean the pin nearest to where the micro-SD card is inserted, we have physical pin 1 which provides 3v3 power. To the right of that pin is physical pin 2 which provides 5v power. The pin numbers then increase as we move down each column, with pin 1 going to pin 3, 5,7 etc until we reach pin 39. You will quickly see that each pin from 1 to 39 in this column follows an odd number sequence. And for the column starting with pin 2 it will go 4,6,8 etc until it reaches 40. Following an even number sequence. Physical pin numbering is the most basic way to locate a pin, but many of the tutorials written for the Raspberry Pi follow a different numbering sequence.

Broadcom (BCM) pin numbering (aka GPIO pin numbering) seems to be chaotic to the average user. With GPIO17, 22 and 27 following on from each other with little thought to logical numbering. The BCM pin mapping refers to the GPIO pins that have been directly connected to the System on a Chip (SoC) of the Raspberry Pi. In essence we have direct links to the brain of our Pi to connect sensors and components for use in our projects.

You will see the majority of Raspberry Pi tutorials using this reference and that is because it is the officially supported pin numbering scheme from the Raspberry Pi Foundation. So, it is best practice to start using and learning the BCM pin numbering scheme as it will become second nature to you over time. Also note that BCM and GPIO pin numbering refer to the same scheme. So, for example GPIO17 is the same as BCM17.

Certain GPIO pins also have alternate functions that allow them to interface with different kinds of devices that use the I2C, SPI or UART protocols. For example, GPIO3 and GPIO 4 are also SDA and SCL I2C pins used to connect devices using the I2C protocol. To use these pins with these protocols we need to enable the interfaces using the Raspberry Pi Configuration application found in the Raspbian OS, Preferences menu.

## **I2C, SPI and UART:** Which Do You Use?

We'll get into the specific differences between I2C, SPI and UART below, but if you're wondering which one you need to use to connect to given device, the short answer is to check the spec sheet. For example, one tiny LED screen might require SPI and another might use I2C (almost nothing uses UART). If you read the documentation that comes with a product (provided it has some), it will usually tell you which Pi pins to use.

For Raspberry Pi 4 users note that there are now many more I2C, SPI and UART pins available to you. These extra interfaces are activated using device tree overlays and can provide four extra SPI, I2C and UART connections.

## **I2C - Inter-Integrated Circuit**

I2C is a low speed two wire serial protocol to connect devices using the I2C standard. Devices using the I2C standard have a master slave relationship. There can be more than one master, but each slave device requires a unique address, obtained by the manufacturer from NXP, formerly known as Philips Semiconductors. This means that we can talk to multiple devices on a single I2C connection as each device is unique and discoverable by the user and the computer using [Linux commands](https://www.tomshardware.com/reviews/raspberry-pi-command-line-commands,6159.html) such as i2cdetect.

As mentioned earlier I2C has two connections: SDA and SCL.  They work by sending data to and from the SDA connection, with the speed controlled via the SCL pin. I2C is a quick and easy way to add many different components, such as LCD / OLED screens, temperature sensors and analog to digital converters for use with photoresistors etc to you project. While proving to be a little trickier to understand than standard GPIO pins, the knowledge gained from learning I2C will serve you well as you will understand how to connect higher precision sensors for use in the field.

The Raspberry Pi has two I2C connections at GPIO 2 and 3 (SDA and SCL) are for I2C0 (master) and physical pins 27 and 28 are I2C pins that enable the Pi to talk to compatible HAT (Hardware Attached on Top) add on boards.

## **SPI - Serial Peripheral Interface**

SPI is another protocol for connecting compatible devices to your Raspberry Pi. It is similar to I2C in that there is a master slave relationship between the Raspberry Pi and the devices connected to it.

Typically, SPI is used to send data over short distances between microcontrollers and components such as shift registers, sensors and even an SD card. Data is synchronised using a clock (SCLK at GPIO11) from the master (our Pi) and the data is sent from the Pi to our SPI component using the MOSI (GPIO GPIO10) pin. MOSI stands for Master Out Slave In. If the component needs to reply to our Pi, then it will send data back using the MISO pin (GPIO9) which stands for Master In Slave Out.

## **UART - Universal Asynchronous Receiver / Transmitter**

Commonly known as “Serial,” the UART pins (Transmit GPIO14, Receive GPIO15) provide a console / terminal login for headless setup, which means connecting to the Pi without a keyboard or pointing device. Normally, the easiest way to do a headless Raspberry Pi setup is simply to control the Pi over a network or direct USB connection (in the case of Pi Zero).

But, if there's no network connection, you can also control a headless Pi using a serial cable or USB to serial board from a computer running a terminal console. UART is exceptionally reliable and provides access to a Pi without the need for extra equipment. Just remember to enable the Serial Console in the Raspberry Pi Configuration application. Chances are that you won't want to do this, but the UART support is there if you need it.

## **Ground (gnd)**

Ground is commonly referred to as GND, gnd or - but they all mean the same thing. GND is where all voltages can be measured from and it also completes an electrical circuit. It is our zero point and by connecting a component, such as an LED to a power source and ground the component becomes part of the circuit and current will flow through the LED and produce light.

When building circuits, it is always wise to make your ground connections first before applying any power as it will prevent any issues with sensitive components. The Raspberry Pi has eight ground connections along the GPIO and each of these ground pins connects to one single ground connection. So, the choice of which ground pin to use is determined by personal preference, or convenience when connecting components.

## **5v(VCC)**

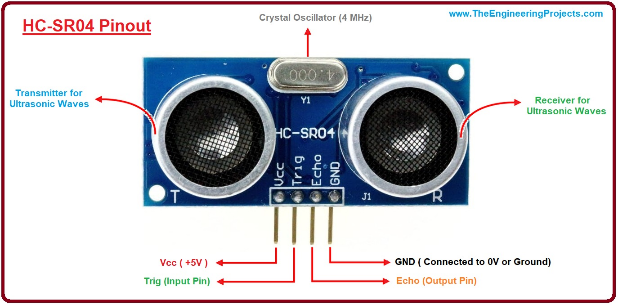
The 5v pins give direct access to the 5v supply coming from your mains adaptor, less power than used by the Raspberry Pi itself. A Pi can be powered directly from these pins, and it can also power other 5v devices. When using these pins directly, be careful and check your voltages before making a connection because they bypass any safety features, such as the voltage regulator and fuse which are there to protect your Pi. Bypass these with a higher voltage and you could render your Pi inoperable.

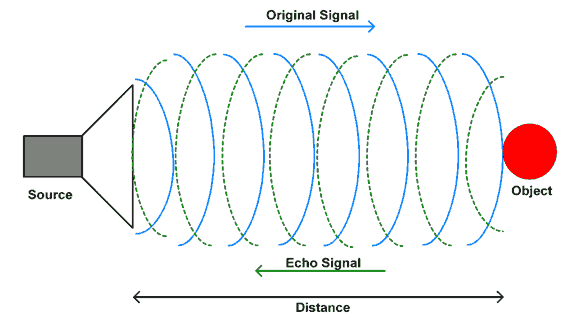
## **3.3V**

The 3v pin is there to offer a stable 3.3v supply to power components and to test LEDs. In reality, it will be rare that you factor this pin into a build, but it does have a special use. When connecting an LED to the GPIO, we first need to make sure that the LED is wired up correctly and that it lights up. By connecting the long leg of the LED, the anode to the 3.3v pin via a resistor, and the shorter leg, the cathode to any of the Ground (gnd) pins we can check that our LED lights up and is working. This eliminates a hardware fault from the project and enables us to start building our project with confidence.

**ULTRASONIC SENSOR**

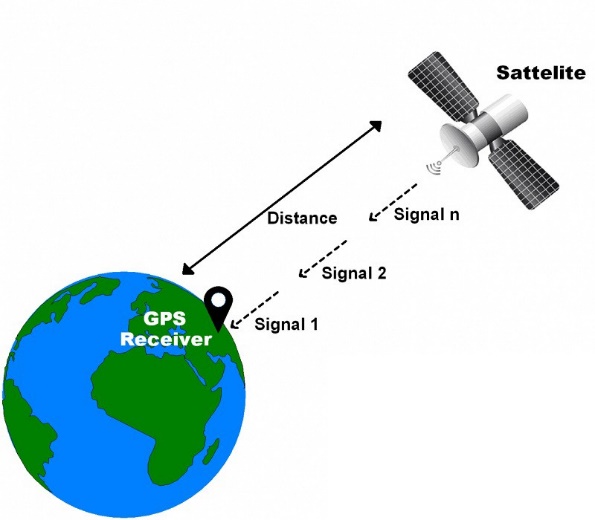
High frequency sound waves are generated by ultrasonic sensor. It evaluates the echo which is received back by the sensors. The time interval between sending the signal and receiving the echo is calculated by sensor to determine the distance to an object. Ultrasonic is like an infrared where it will reflect on a surface in any shape, but ultrasonic has a better range detection compared to infrared. In robotic and automation industry, ultrasonic has been highly accepted because of its usage. In our Project the Ultrasonic sensor distance measurement Module deals with the distance measurement between the obstacle and the blind person. This module starts the process when the user turns on the device using power supply. Firstly, when the device turns on, the ultrasonic sensor will automatically send the distance measurement of the obstacle in front of the blind.





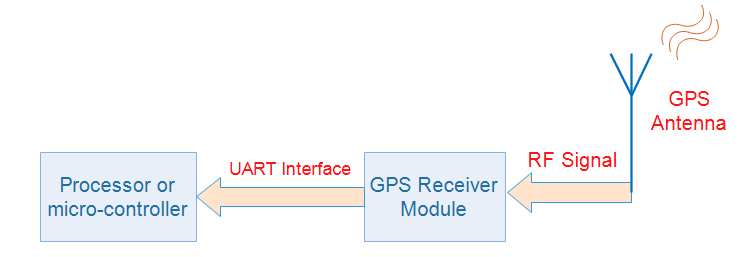
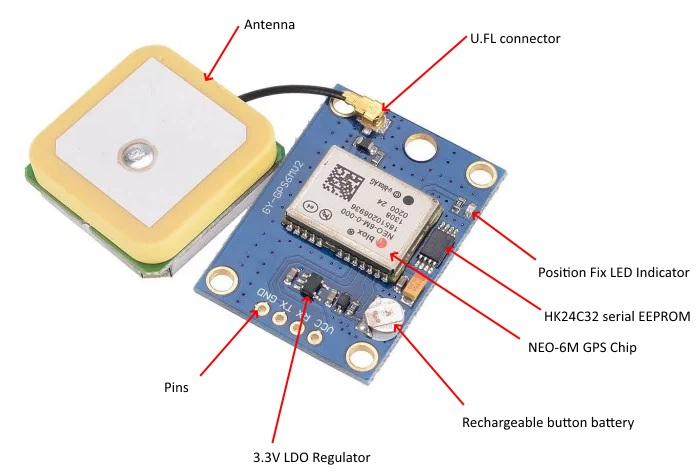
**GPS MODULE**

GPS (Global Positioning System) is a satellite-based navigation system. It provides time and location-based information to a GPS receiver, located anywhere on or near the earth's surface. GPS works in all weather conditions. GPS works through a technique called trilateration. Trilateration is the process of determining your position based on the intersection of spheres. When a receiver receives a signal from one of the satellites, it calculates its distance from the satellite considering a 3-D sphere with the satellite located at the centre of the sphere. Once the receiver does the same with 3 other GPS satellites, the receiver then proceeds to find the intersection point of the 3 spheres to calculate its location. Used to calculate location, velocity, and elevation, **Trilateration** collects signals from satellites to output location information.

The GPS module receives a timestamp from each of the visible satellites, along with data on where in the sky each one is located (among other pieces of data). From this information, the GPS receiver now knows the distance to each satellite in view. **If the GPS receiver’s antenna can see at least 4 satellites, it can accurately calculate its position and time**

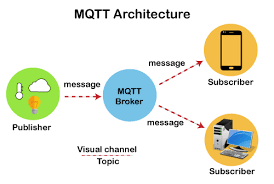
A single satellite broadcasts a microwave signal which is picked up by a GPS device and used to calculate the distance from the GPS device to the satellite. Since a GPS device only gives information about the distance from a satellite, a single satellite cannot provide

much location information. Satellites do not give off information about angles, so the location of a GPS device could be anywhere on a sphere’s surface area.  
  
When a satellite sends a signal, it creates a circle with a radius measured from the GPS device to the satellite.  
  
When we add a second satellite, it creates a second circle, and the location is narrowed down to one of two points where the circles intersect.  
  
With a third satellite, the device’s location can finally be determined, as the device is at the intersection of all three circles.

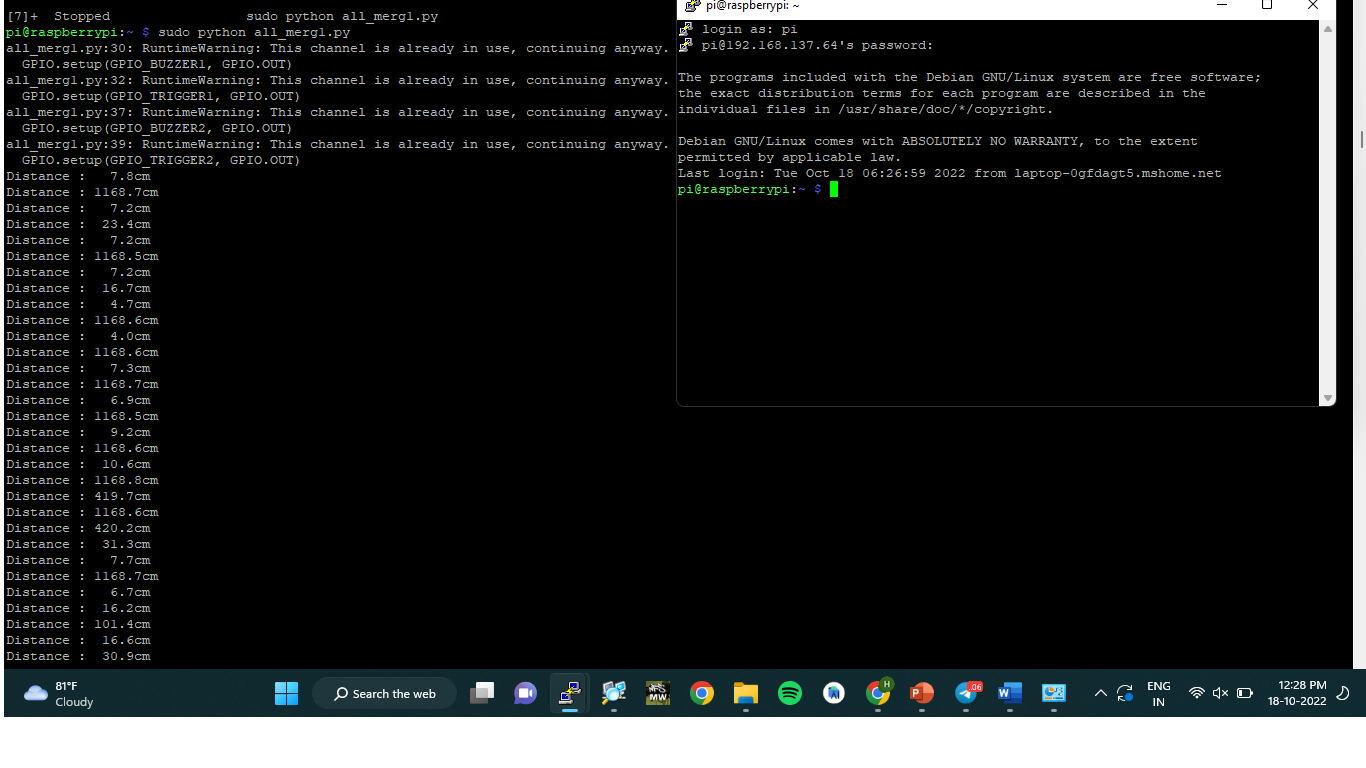
******

**MQTT PROTOCOL**

MQTT stands for **Message Queuing Telemetry Transport**. MQTT is a machine-to-machine internet of things connectivity protocol. It is an extremely lightweight and publish-subscribe messaging transport protocol. This protocol is useful for the connection with the remote location where the bandwidth is a premium. These characteristics make it useful in various situations, including constant environment such as for communication machine to machine and internet of things contexts. It is a publish and subscribe system where we can publish and receive the messages as a client. It makes it easy for communication between multiple devices. It is a simple messaging protocol designed for the constrained devices and with low bandwidth, so it's a perfect solution for the internet of things applications.



**SOFTWARE USED:**

**PUTTY:** is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port.

**MQTT:**

MQTT (Message Queue Telemetry Transport) is a lightweight, publish-subscribe, machine to machine network protocol for Message queue/Message queuing service. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically. The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device (from a micro controller up to a fully-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

**METHODOLOGY**

YES

NO

NOO

YES

SEND CURRENT LOCATION TO MQTT APP

BUZZER ON

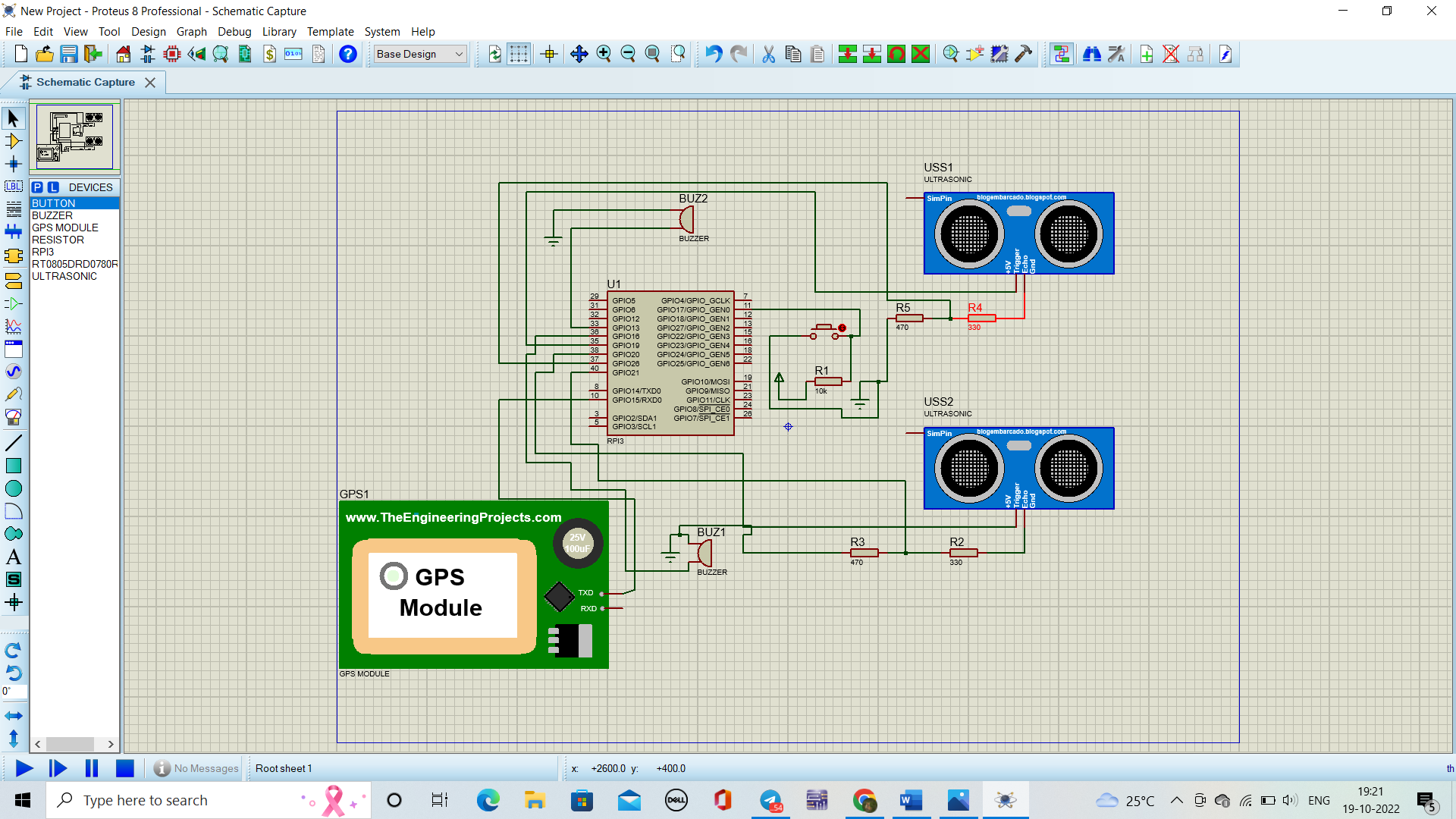
IF DETECTED

IF PRESSED

CHECKS IF BUTTON IS PRESSED

CHECKS FOR OBSTACLE

**FLOW CHART:**

CIRCUIT DIAGRAM:

CODE:

#!/usr/bin/python

# Import required Python libraries

import multiprocessing

import os

import RPi.GPIO as GPIO

import time

import paho.mqtt.client as mqtt

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.IN)

#### setting up MQTT #########

mqttc = mqtt.Client()

mqttc.connect("broker.hivemq.com", 1883,60)

mqttc.loop\_start()

# Select which GPIOs you will use

GPIO\_BUZZER1 = 33

GPIO\_TRIGGER1 = 35

GPIO\_ECHO1 = 37

#for 2nd ultrasonic

GPIO\_BUZZER2 = 36

GPIO\_TRIGGER2 = 38

GPIO\_ECHO2 = 40

# Set BUZZER to OUTPUT mode

GPIO.setup(GPIO\_BUZZER1, GPIO.OUT)

# Set TRIGGER to OUTPUT mode

GPIO.setup(GPIO\_TRIGGER1, GPIO.OUT)

# Set ECHO to INPUT mode

GPIO.setup(GPIO\_ECHO1, GPIO.IN)

# Set BUZZER to OUTPUT mode

GPIO.setup(GPIO\_BUZZER2, GPIO.OUT)

# Set TRIGGER to OUTPUT mode

GPIO.setup(GPIO\_TRIGGER2, GPIO.OUT)

# Set ECHO to INPUT mode

GPIO.setup(GPIO\_ECHO2, GPIO.IN)

# Measures the distance between a sensor and an obstacle and returns the measured value

def distance1():

# Send 10 microsecond pulse to TRIGGER

GPIO.output(GPIO\_TRIGGER1, True) # set TRIGGER to HIGH

time.sleep(0.00001) # wait 10 microseconds

GPIO.output(GPIO\_TRIGGER1, False) # set TRIGGER back to LOW

# Create variable start and assign it current time

start = time.time()

# Create variable stop and assign it current time

stop = time.time()

# Refresh start value until the ECHO goes HIGH = until the wave is send

while GPIO.input(GPIO\_ECHO1) == 0:

start = time.time()

# Assign the actual time to stop variable until the ECHO goes back from HIGH to LOW = the wave came back

while GPIO.input(GPIO\_ECHO1) == 1:

stop = time.time()

# Calculate the time it took the wave to travel there and back

measuredTime1 = stop - start

# Calculate the travel distance by multiplying the measured time by speed of sound

distanceBothWays = measuredTime1 \* 33112 # cm/s in 20 degrees Celsius

# Divide the distance by 2 to get the actual distance from sensor to obstacle

distance1 = distanceBothWays / 2

# Print the distance to see if everything works correctly

print("Distance : {0:5.1f}cm".format(distance1))

# Return the actual measured distance

return distance1

def distance2():

# Send 10 microsecond pulse to TRIGGER

GPIO.output(GPIO\_TRIGGER2, True) # set TRIGGER to HIGH

time.sleep(0.00001) # wait 10 microseconds

GPIO.output(GPIO\_TRIGGER2, False) # set TRIGGER back to LOW

# Create variable start and assign it current time

start = time.time()

# Create variable stop and assign it current time

stop = time.time()

# Refresh start value until the ECHO goes HIGH = until the wave is send

while GPIO.input(GPIO\_ECHO2) == 0:

start = time.time()

# Assign the actual time to stop variable until the ECHO goes back from HIGH to LOW = the wave came back

while GPIO.input(GPIO\_ECHO2) == 1:

stop = time.time()

# Calculate the time it took the wave to travel there and back

measuredTime2 = stop - start

# Calculate the travel distance by multiplying the measured time by speed of sound

distanceBothWays = measuredTime2 \* 33112 # cm/s in 20 degrees Celsius

# Divide the distance by 2 to get the actual distance from sensor to obstacle

distance2 = distanceBothWays / 2

# Print the distance to see if everything works correctly

print("Distance : {0:5.1f}cm".format(distance2))

# Return the actual measured distance

return distance2

# Calculates the frequency of beeping depending on the measured distance

def beep\_freq1():

# Measure the distance

dist = distance1()

# If the distance is bigger than 50cm, we will not beep at all

if dist > 50:

return -1

# If the distance is between 50 and 30 cm, we will beep once a second

elif dist <= 50 and dist >=30:

return 1

# If the distance is between 30 and 20 cm, we will beep every twice a second

elif dist < 30 and dist >= 20:

return 0.5

# If the distance is between 20 and 10 cm, we will beep four times a second

elif dist < 20 and dist >= 10:

return 0.25

# If the distance is smaller than 10 cm, we will beep constantly

else:

return 0

# Calculates the frequency of beeping depending on the measured distance

def beep\_freq2():

# Measure the distance

dist = distance2()

# If the distance is bigger than 50cm, we will beep constantly

if dist > 50:

return 0

# If the distance is between 50 and 30 cm, we will beep four times a second

elif dist <= 50 and dist >=30:

return 0.25

# If the distance is between 30 and 20 cm, we will beep every twice a second

elif dist < 30 and dist >= 20:

return 0.5

# If the distance is between 20 and 10 cm, we will beep once a second

elif dist < 20 and dist >= 10:

return 1

# If the distance is smaller than 10 cm, we will not beep at all

else:

return -1

# Main function

def main():

try:

import time

# Repeat till the program is ended by the user

while True:

Data=GPIO.input(11)

if (Data > 0):

import serial

import time

import string

import pynmea2

port="/dev/ttyAMA0"

ser=serial.Serial(port, baudrate=9600, timeout=0.5)

dataout = pynmea2.NMEAStreamReader()

newdata=ser.readline()

if newdata[0:6] == "$GPRMC":

newmsg=pynmea2.parse(newdata)

lat=newmsg.latitude

lng=newmsg.longitude

gps = "Latitude=" + str(lat) + "and Longitude=" + str(lng)

print(gps)

(result,mid) = mqttc.publish("paho/location",gps,2)

print("Switch is pressed ")

time.sleep(1)

# Get the beeping frequency

freq1 = beep\_freq1()

# No beeping

if freq1 == -1:

GPIO.output(GPIO\_BUZZER1, False)

time.sleep(0.25)

# Constant beeping

elif freq1 == 0:

GPIO.output(GPIO\_BUZZER1, True)

time.sleep(0.25)

# Beeping on certain frequency

else:

GPIO.output(GPIO\_BUZZER1, True)

time.sleep(0.2) # Beep is 0.2 seconds long

GPIO.output(GPIO\_BUZZER1, False)

time.sleep(freq1)

# Pause between beeps = beeping frequency

freq2 = beep\_freq2()

if freq2 == -1:

GPIO.output(GPIO\_BUZZER2, False)

time.sleep(0.25)

elif freq2 == 0:

GPIO.output(GPIO\_BUZZER2, True)

time.sleep(0.25)

else:

GPIO.output(GPIO\_BUZZER2, True)

time.sleep(0.2) # Beep is 0.2 seconds long

GPIO.output(GPIO\_BUZZER2, False)

time.sleep(freq2)

# If the program is ended, stop beeping and cleanup GPIOs

except KeyboardInterrupt:

GPIO.output(GPIO\_BUZZER1, False)

GPIO.output(GPIO\_BUZZER2, False)

GPIO.cleanup()

# Run the main function when the script is executed

if \_\_name\_\_ == "\_\_main\_\_":

main()

**WORKING**

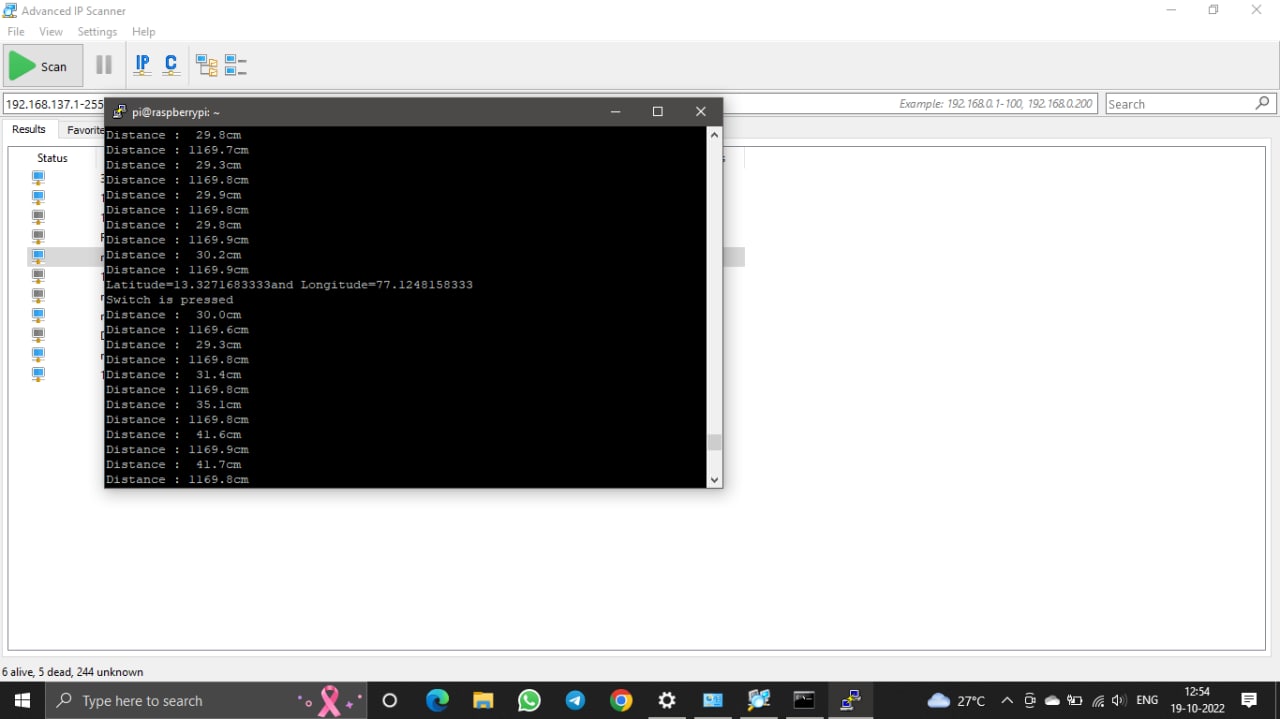
CONNECTIONS:

Trigger of the ultrasonic sensor 1 is connected to the pin no.35 of raspberry pi, and trigger of the ultrasonic 2 is connected to the pin no.38. Echo of ultrasonic sensor 1 is connected to pin no.37, and echo of ultrasonic sensor 2 is connected to pin no.45. GPS Module transmitter pin is connected to the receiver pin of raspberry pi i.e pin no.10. Two buzzers are connected to pin number 33 and 36. A switch is connected to pi no.11.

WORKING:

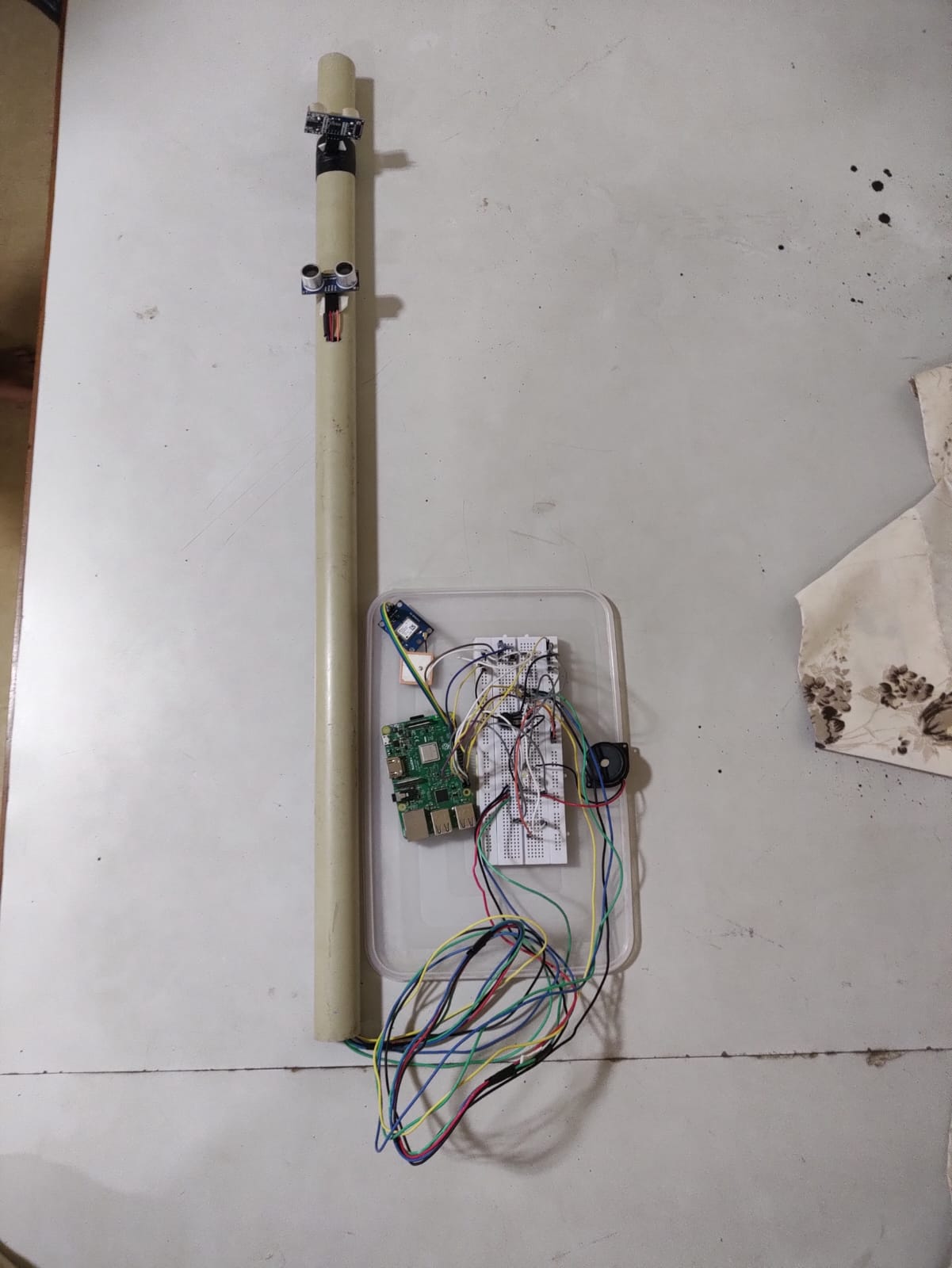
As soon as the power supply is turned on and raspberry pi is logged in the ultrasonic sensor starts to send waves. If any object is present the waves bounce back. Total time will be calculated and it will be halved to get the time required to reach the obstacle. Time is then multiplied with factor to get the distance. Based on the distance we have programmed the buzzer for different frequencies. For front obstacle detection if the obstacle is present in the range of distance of 50cm-30cm the frequency will be low. If the obstacle is in the range of less than 10cm the frequency will be high. For path hole detection we have used the reverse principle used in front obstacle detection. If no obstacle is present then buzzer frequency will be high or else frequency will be low.

Consider a situation if the blind person is lost and he doesn’t have any idea regarding the place where he is? Keeping in mind such situation we have installed gps module and added a switch to it. In such situation if he presses the switch, automatically his live location will be shared by mqtt server to his care taker.

**RESULT**

**LAYOUT SPECIFICATION**

FRONT VIEW

****

**ANALYSIS**

After we designed and programmed the Modern Blind Stick to the needed standard, this innovation product has been functioning well. We tested the Modern Blind Stick with certain obstacles, hazard, range and distance and it work successfully as we programed the Modern Blind Stick. Analysis about the point of view about the visually impaired person walk at the public. First analysis is the visually impaired person to walk at the public independently. Secondly, the real time location of the visually impaired person at the public. Third, the safety of the visually impaired person at the public. The analysis shows the problems as the visually impaired person the risk when walking at the public.

**CONCLUSION AND RECOMMENDATION**

CONCLUSION:

In the end of our project, we can conclude that our project can reduce the number of risk and injuries for the visually impaired person when walking at public. Nowadays, even at young age experience the visually impairment. This thing cannot be taken so lightly as they know how much risk could it be. If the number of risk and injuries increasing rapidly, the kid or the person will lose their spirit to walk independently. The Smart stick acts as a basic platform for the coming generation of more aiding devices to help the visually impaired to navigate safely both indoor and outdoor. It is effective and affordable. It leads to good results in detecting the obstacles on the path of the user in a range of two meters. Though the system is hard-wired with sensors and other components, it's light in weight. Further aspects of this system can be improved via wireless connectivity between the system components, thus, increasing the range of the ultrasonic sensor and implementing a technology for determining the speed of approaching obstacles.

RECOMMENDATION:

In the future, we hope that our project can be commercialize as there are many benefits such as to reduce the number of risk and injuries for the visually impaired people. Our life is very priceless and cannot be replace. Because we all just live only once, so seize our life with positive vibes. We hope we can improvise our project if there is a thing that can make our product more quality than before.

**REFERENCE**

1. <https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/>
2. <https://sparklers-the-makers.github.io/blog/robotics/use-neo-6m-module-with-raspberry-pi/>
3. <https://www.electronicwings.com/raspberry-pi/gps-module-interfacing-with-raspberry-pi>
4. <https://mqtt.org/>
5. <https://www.raspberrypi.org/>