ABSTRACT

Security is a major concern in many parts of the world. From petty thefts to full-fledged heists, the increase in thefts has been a cause for concern. To combat these problems, many security measures have been implemented. From alarm systems to CCTV cameras and human patrols, there are different levels of security systems in the world. The most common of these is a CCTV camera system. Recent developments in IoT has made it possible to access these systems remotely, However, they have also increased the price dramatically.

The cheapest and most commonly used CCTV camera systems consist of 4 cameras per room that stream a live feed to a PC which is monitored by a human guard. These are used in most small shops and medium-large workplaces. Despite being relatively cheap, these security systems do not account for human error. In underpaid workplaces and small shops, the guard monitoring CCTV often falls asleep on the night shift. This results in burglaries despite having a security system in place.

In this paper, we discuss the design and implementation of an auxiliary security system that is used in tandem with the standard CCTV setup that is present in most work settings. The system consists of 1-4 units of rover type robots depending on the number of walls with openings that can be turned on at night when the workplace is empty. These WiFi-enabled robots will patrol the room at night and use the principles of IoT to send an alert upon detection of an intruder. They also cost a fraction of the price of a standard CCTV system.

INDEX

SR NO.	TITLE	PAGE NO.
1	**INTRODUCTION**	5
2	**LITERATURE SURVEY**	13
3	**SYSTEM DESCRIPTION**	14
4	**SYSTEM DESIGN**	17
5	**WORKING**	25
6	**TEST RESULTS**	25
7	**ADVANTAGES AND LIMITATIONS**	34
8	**CONCLUSION**	35
9	**FUTURE WORK**	35

10	**REFERENCES**	36

LIST OF FIGURE

Fig no .	Figure Title
1	Illustration of Detection algorithm
2	Illustration of API keys
3	thingtweet Authorization
4	Thingtweet configuration
5	Thingspeak MQTT configuration
6	MQTTX subscriber configuration
7	. Algorithm for working (1)
8	Algorithm for working (2)
9	. Circuit Diagram on Proteus
10	Front View
11	Top view
12	Side View
13	Arduino MEGA
14	.Battery Source
15	. Motor Driver
16	. Ultrasonic Sensors
17	Infrared sensor
18	. HDPP wheel
19	. BO motors
20	BreadBoard
21	MQTT
22	Code sent from Microcontroller
23	Email alert
24	Twitter alert
25	MQTT alert
26	Email spam error
27	**Circuit Diagram**
28	**Serial Monitor**

LIST OF TABLES

Sr no	Table title
1	Modes of operation of MEGA
2	Calculation of Error Percentage of Ultrasonic sensors

INTRODUCTION

System Description

The system consists of 1-4 bots each monitoring one side of a room and uploading alert data to a cloud IoT Platform known as ThingSpeak.

The overall system can be broadly classified into three categories; The mechanical system, The electronic system and The IoT system.

Mechanical system:

The chassis of the bot is made of two laser-cut acrylic sheets joined together by 50mm spacers. Acrylic is used because it is durable, cost-effective and easily replaceable. It has also been painted matte black to blend in with the darkness at night.

The lower sheet houses the four battery-operated motors connected to a gearbox for extra torque. It also houses the common power and ground circuit as well as the comparator circuit for the Infrared Sensors.

Four plastic wheels have been attached to the motors mounted on the lower sheet as they are easily available and very cheap. The rims of the wheel have also been made black to make the bot inconspicuous.

The Upper sheet houses the microcontroller unit, the motor driver and the battery. The microcontroller is on the top sheet because the design makes it easy to access and reprogram the controller with proper IoT Credentials and Wifi SSID and password. The battery is placed on top so it can easily be replaced and the motor driver is put on top so that the heatsink can dissipate the heat effectively. This is also the reason we have not implemented a cover in the design.

Electronic System:

The microcontroller used is the modified version of the ATmega 2560 microcontroller. This modified board is similar to the Arduino mega in that it consists of 54 I/O pins. Although we have not used all the pins, the remaining pins may be used to implement more sensors in the future. This modified ATmega board also has an ESP8266 wifi module built-in and connected internally via i2c serial communication protocol on channel 3. Thus, the commands to communicate with ESP8266 must be administered on Serial 3.

The board also has a Ch340 USB TTL converter which allows uploading of code directly via MicroUSB. The serial connection between these 3 vital components (CH340, ATmega 2560, ESP8266) can be configured via an 8 bit DIP switch. The switches may be configured in various ways to make only certain chips work together or all work independently. This means that we can program the ESP8266 and the atmega separately to run their own programs. This allows for a contingency code which can send a message in case the atmega

malfunctions.However, we are running the board in MODE 4 according to the table below. We have also used MODE 2 to upload AT Firmware to the ESP8266 which allows us to control the ESP8266 with the ATmega 2560 by sending AT commands. An Arduino Library, known as “WiFiESP”, has been used to do the same. This board provides an expansive yet compact method to implement large scale WiFi based projects with ease.

Mode No	Description	1	2	3	4	5	6	7	8
1	CH340 connect to ESP8266 (upload sketch)	OFF	OFF	OFF	OFF	ON	ON	ON	NoUSE
2	CH340 connect to ESP8266 (connect)	OFF	OFF	OFF	OFF	ON	ON	OFF	NoUSE
3	CH340 connect to ATmega2560 (upload sketch)	OFF	OFF	ON	ON	OFF	OFF	OFF	NoUSE
4	CH340 connect to Mega2560 COM3 connect to ESP8266	ON	ON	ON	ON	OFF	OFF	OFF	NoUSE
5	Mega2560+ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF	NoUSE
6	All modules work independent	OFF	OFF	OFF	OFF	OFF	OFF	OFF	NoUSE

Table 1 . Modes of operation of MEGA

The other components of the system include an L298N dual motor drive. However, each side is driving two motors hence the right wheels act simultaneously and the left wheels act simultaneously.

There are two IR sensors which help the bot follow a path defined by a black line. To make the black line distinct in the dark at night, we have attached a standard white LED which will supplement the line following system.

The final and most important part of the electronic system is the ultrasonic sensor detection algorithm. The ultrasonic sensors used are rated as having a max detection range of 4m. However, we have determined experimentally that the actual range of detection is 10m. The details of the experiment are given below.

within the program internally, However, these will not matter as the reference ranges will also have the same calculation error.

Conclusion: We must subtract 5 cm from the max detection range to ensure no false alarms.

The detection system consists of 2 phases; The Calibration Phase and The Detection Phase

Calibration Phase:

The bot at first needs to be placed at the starting black line facing the opposite wall which also has a finishing black line. In this phase, the bot will determine its range of detection and the minimum distance for it to detect an intruder on the line. This is done by an algorithm that updates the maximum side detection range if it is less than the previous maximum detection range. This ensures that the bot does not mistake furniture for an intruder and send an alert. This feature comes with the disadvantage that the bot will not be as effective as possible in places with extruding furniture. An illustration is provided below

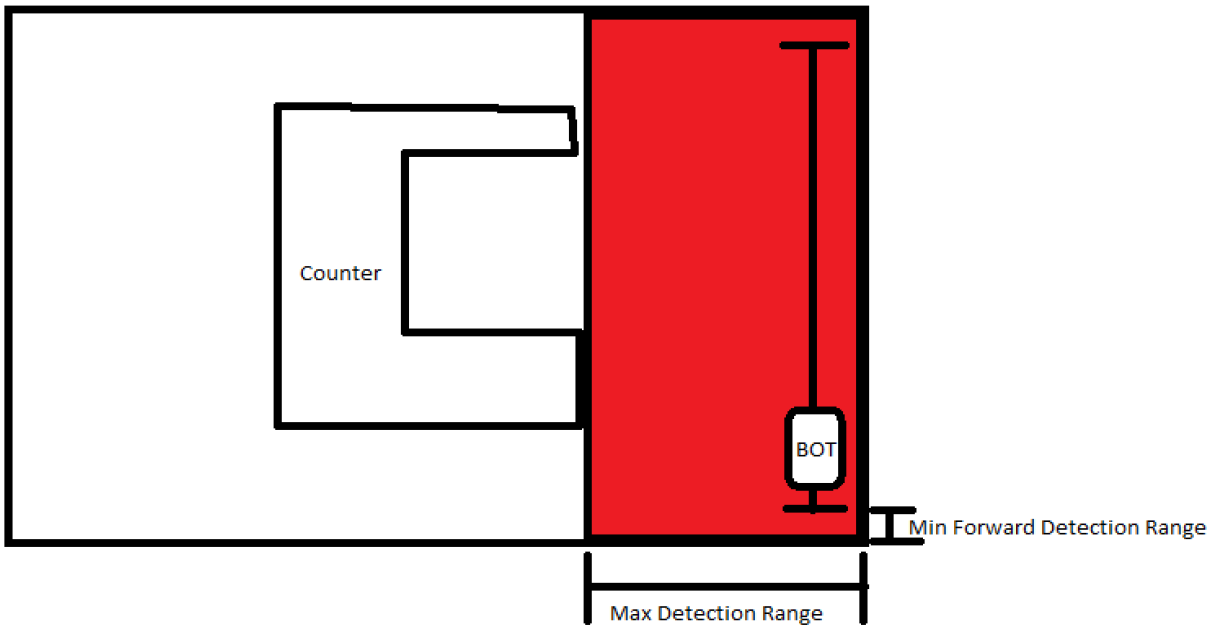

Fig 1. Illustration of Detection algorithm

After determining the maximum detection range and the minimum forward detection range, the calibration phase will end.

The Detection Phase:

In this phase, the bot will continuously take distance measurements and compare them with the max detection distance and min forward distance it determined in calibration. If any distance comes out to be less than calibrated distances, a flag message is triggered and sent to ThingSpeak IoT Platform.

IoT System:

To implement the IoT system we have used ThingSpeak IoT Platform. ThingSpeak is a free and open-source software written in Ruby which allows users to communicate with internet-enabled devices. It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites. ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications.

ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

ThingSpeak has multiple apps which are used to set up the Alert System. The apps used in Thingspeak include React, ThingTweet, MATLAB Analysis and MQTT Communication. Each has been explained below

To Get Started with ThingSpeak, we first created a new public channel and added a field in that channel known as the EMW alert system. After the creation of the channel and field, we noted the write API key, since we would be writing a flag value to this channel from the microcontroller and we also noted the channel ID from the API key Section. The illustration of the same has been listed below. Although the channel ID is not used, it is required by the Thingspeak Arduino library for writing to a channel field.

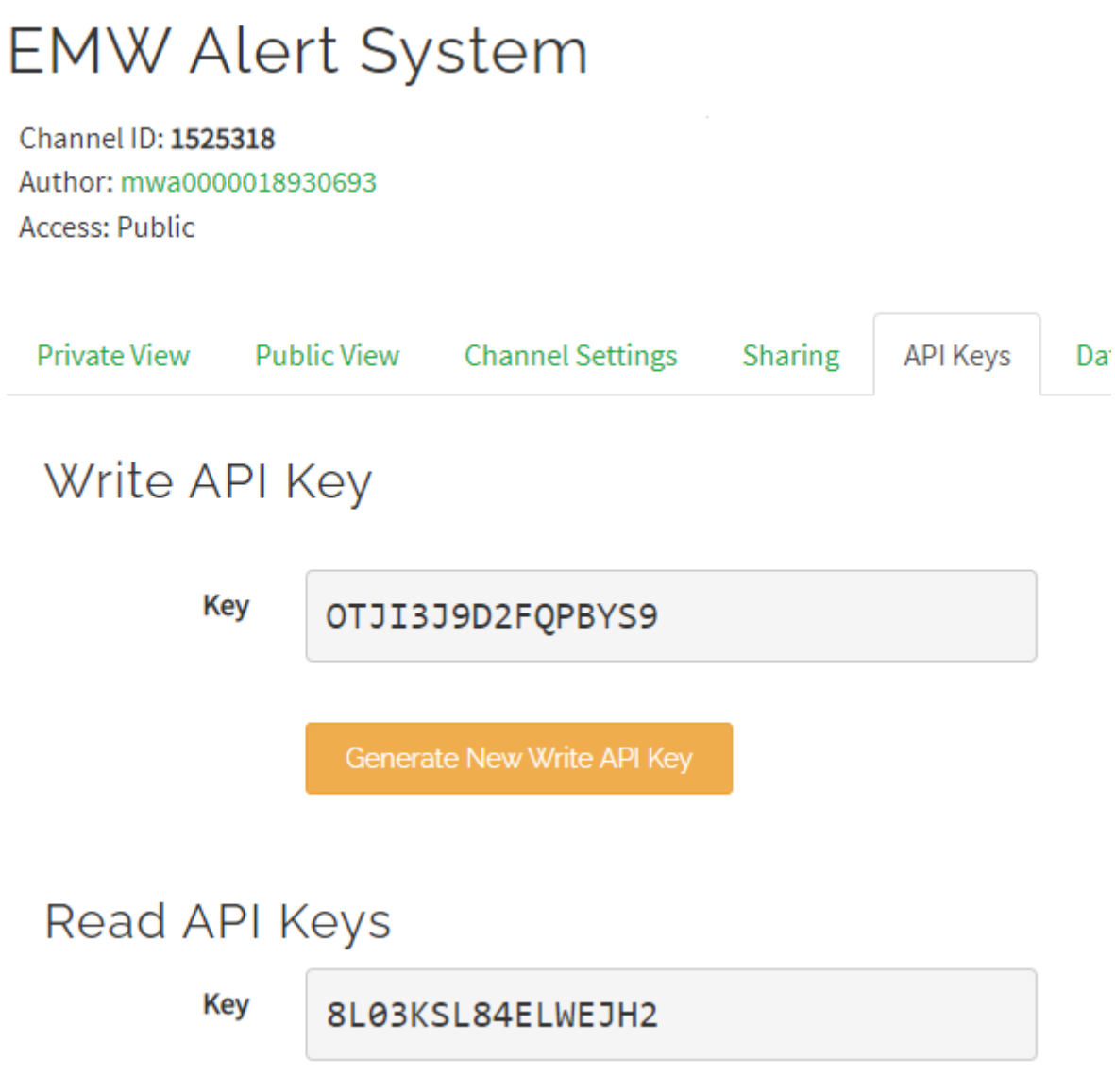

Fig 2. Illustration of API keys

The flag value can be written to this channel by sending the following GET request to Port 80 (common internet port):

GET https://api.thingspeak.com/update?api_key=”<write api key>”&field<number>=<value>

ThingSpeak allows sending of up to 3,000,000 messages per year on a free account in this way.

Once this is set up, we can send alerts using React App which sends a tweet via ThingTweet and email via MATLAB Analysis. We also use ThingSpeak MQTT to send an instantaneous desktop alert.

ThingTweet:

The ThingTweet app is used to link a Twitter account to the user's ThingSpeak account. The devices can then send alerts via Twitter using ThingTweet. Once we click on link account, we are redirected to a Twitter page where, upon logging in, we have to authorize ThingSpeak to send a Twitter alert. An illustration is provided below

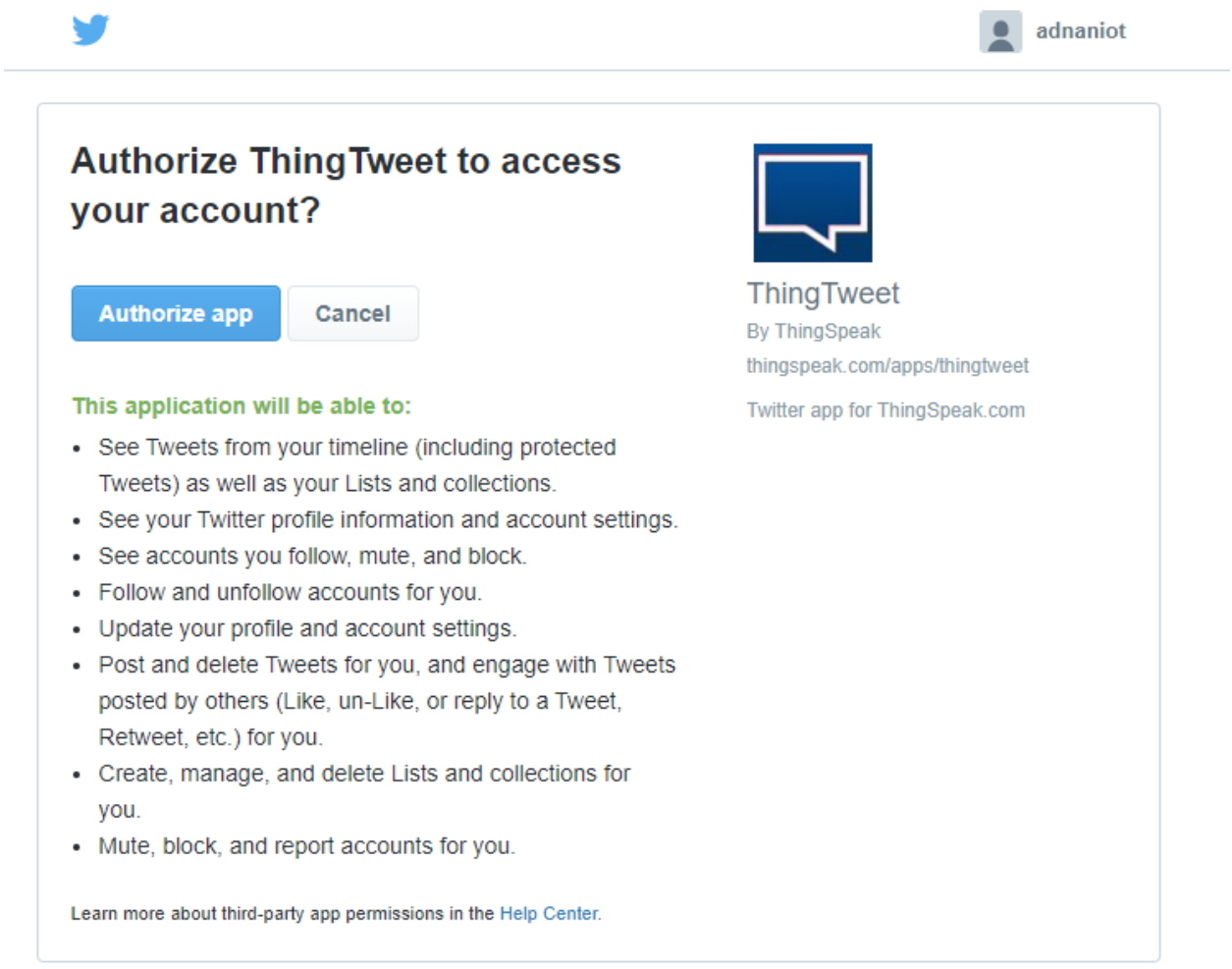

Fig 3. thingtweet Authorization

Following this, an API key for twitter will be generated. This may be used in a POST request sent to port 80 as follows:

POST https://api.thingspeak.com/apps/thingtweet/1/statuses/update

api_key=<twitter API key>

status=<body of tweet>

However, this is done in this system via React App.

MATLAB Analysis:

This app allows the use of running MATLAB code and is used in conjunction with React app.

This requires Thingspeak Alert API Key which can be obtained from the main account page in ThingSpeak. It begins with “TAK”.

The following code was used to send an email to the google account used to register with ThingSpeak. The system employed uses a Java payload system to send the e-mail. Following code was used:

```
alert_body = 'Please Check CCTV';
alert_subject = 'Intruder Alert';
alert_api_key = 'TAKG5N7ZH1Q7TQ31Z3PUG';
alert_url= "https://api.thingspeak.com/alerts/send";
jsonmessage = sprintf(['{"subject": "%s", "body": "%s"}'],
alert_subject,alert_body);
options = weboptions("HeaderFields", {'Thingspeak-Alerts-API-Key',
alert_api_key; 'Content-Type','application/json'});
result = webwrite(alert_url, jsonmessage, options);
```

ThingSpeak allows sending of 800 emails per year on a free account.

React:

React works with ThingHTTP, ThingTweet, and MATLAB Analysis apps to perform actions when channel data meets a certain condition. In our case, React posts a tweet and an email to send an alert when the EMW alert field receives a flag value “1”

To send the tweet, we use the ThingTweet react in the following configuration

Name:	Twitter EMW
Condition Type:	String
Test Frequency:	On data insertion
Last Ran:	2021-10-24 20:37
Channel:	EMW Alert System
Condition:	Field 1 (Flag) ends with "1"
ThingTweet:	adnaniot: Intruder Alert!! Please Check The CCTV!
Run:	Each time the condition is met
Created:	2021-10-03 5:18 pm

Fig 4. Thingtweet configuration

This sends a tweet once we receive a flag message.

Similarly, we add a new react in which MATLAB Analysis is also run under the same configuration.

Desktop Notifications (MQTTX)

MQTT is a lightweight, publish-subscribe network protocol that transports messages between devices. The single most amazing advantage of MQTT protocol is that it is practically instantaneous since it does not require the Thingspeak Server's response which is normally required in the other two alerting methods.

ThingSpeak allows the addition of one MQTT device for free. We have added a device with the following configurations.

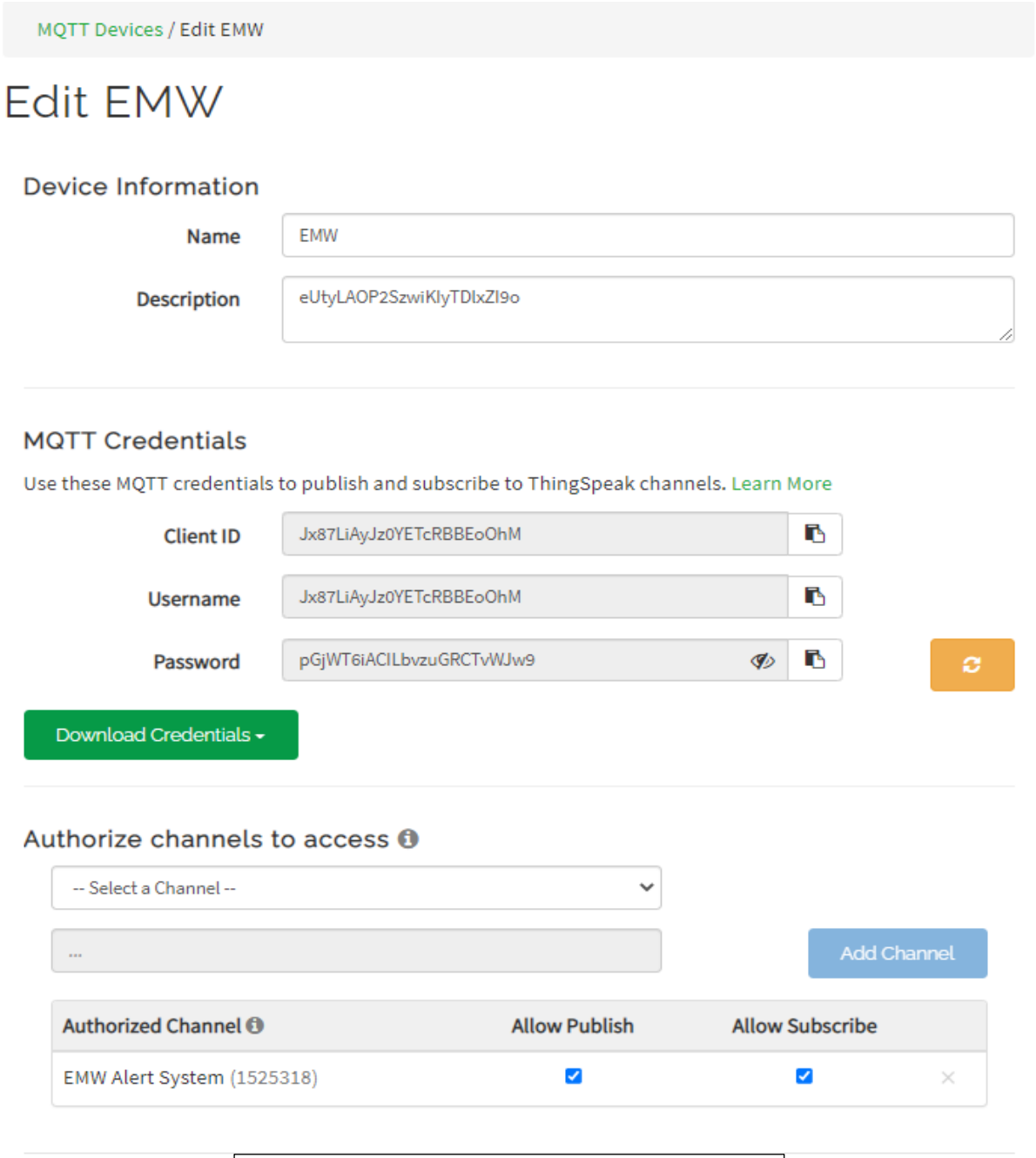

Fig 5. Thingspeak MQTT configuration

Once the device is cre erated. This allows desktop apps such as MQTTx to subscribe to the MQTT device on thingspeak which is publishing data from the authorised channel (the alert channel) continuously.

We have used MQTTx to push desktop notifications. To make MQTTx a subscriber, we first add thingspeak as a broker using the following configurations

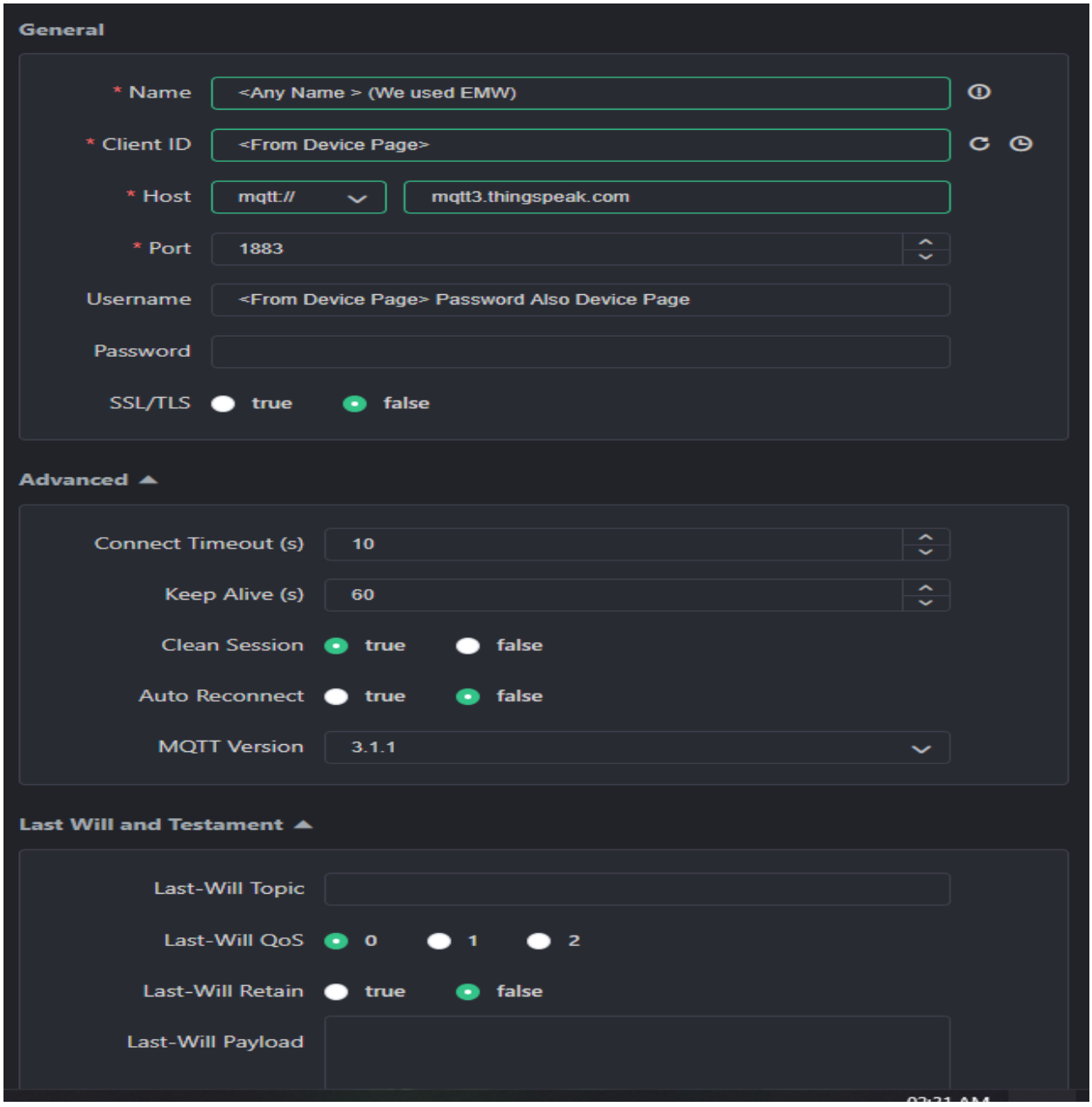

Fig 6. MQTTX subscriber configuration

Then we subscribe to a field by adding a new subscription with topic

channels/<channelID>/subscribe/fields/field<field number>

This sets up the desktop notification.

LITERATURE REVIEW

To prepare for the undertaking of this project, certain steps were followed to ascertain the proper flow of this project from the design phase to the implementation phase. One of the

most important steps that were taken was the literature survey in which numerous papers were studied and listed below are our understandings of each of them

From the paper ‘Vehicle Accident Alert System’ by Authors Neel Patel; Keyur Patel; Rohan Thosar; Darshan Tank; Prof. Rashmi Adatkar 2019, helped us gain an insight into usage and programming the NodeMCU effectively to be used in an alert system which helped set up our base for the IOT domain of our project. Although we did not use the NodeMCU in the final project, this paper gave us insights into working with an ESP8266 module, which we did use.

The paper ‘A line follower robot from design to implementation: Technical issues and problems’ by authors Mehran pakdaman, M. Mehdi Sanaatiyan, Mahdi Rezaei Ghahroudi Talks About Various Parts Required to make a line follower , its Sensor configuration and also Provides Good Code Structure . this paper gave as un understanding into positioning the sensors to reduce effect of ambient light and about the various states of motor while providing us with a good base to build our code on

‘Design and Implementation of the Mobile Fire Alarm System Using Wireless Sensor Networks’ by Karwan MUHEDEN, Ebubekir ERDEM, Sercan VANÇIN helped us to map out our Integration of multiple sensors a and understand how to Use Wi-Fi to send data over to a aggregation point. It was also hugely beneficial for our design stage as it provides a Flowchart for the alarm system. Which was instrumental in understanding the working of the entire alert system

The paper by ‘Internet of Things Based Indoor Smart Surveillance and Monitoring System using Arduino and Raspberry Pi’ by N C Sendhilkumar , C Malarvizhi2, M Anand, and K Periyarselvam suggested a new and efficient system instead of complex algorithms for motion detection . This paper also provided us with an innovative way for the movement algorithm for the patrol robot which gave us different options to program our patrol bot in case of any errors. This was the paper which initially gave us the idea of sending data over Wi-Fi to a smart phone and how to implement it successfully using Arduino and raspberry pi.

The Paper ‘Border Security System using Arduino & Ultrasonic Sensors’ by AMIT KUMAR, ANCHAL BARANWAL, ARUN KUMAR, BRIJESH KUMAR KUSHWAHA, DHANAJAY MISHRA, DEEPU KUMAR, VARUN SINGHAL helped us get a deeper understanding of ultrasonic sensors and as to why they are the backbone of this project. This entire system uses the technique of echolocation which is based on ultrasonic wave radar, RF communication & Infra- Red technology which is separately controlled by different Arduino boards which controls the different sensors, motors & line following robot according to the

desire. We took inspiration from this idea and implemented it by using an Arduino mega instead of a normal Arduino. This robot works on the principle of Infra-Red Line Tracking. Which can be white or black line based on the programming. This paper showed us the different applications depending upon the sensors such as: Accurate Distance Measurement, Close Range Device, Level Detection, Vehicle Detection, Blind Person Support and Robotics Barrier

'Temperature and Humidity Monitoring and Control System with Thing Speak' by Khin Kyawt Kyawt Khaing Talks about viability of Thingspeak as an IOT server for small scale projects. And the t process of IOT integration over Wi-Fi . it also introduced us to the MTTQ which we implemented successfully. This paper helped us set our entire operation for the IOT protocol as it provided us with a Description on How to use Thingspeak and also the use of MQTT protocol to route to an app which we have implemented successfully. It further helped us in our implementation as it provided entire flowchart for sending data collected from sensor to getting alert on phone.

SYSTEM DESCRIPTION

The system consists of 1-4 units each uploading alerts to the same Thingspeak Channel. The robot determines the presence of an intruder via comparing detected distance and the calibrated detection range. If an intruder steps in the detection range of the bot, the detected distance is less than the normal range and the bot sends an alert flag of value 1 to thingspeak. Thingspeak has an app that utilizes Thingtweet app to send a twitter alert message and a Matlab script to send an email to the registered thingspeak ID. It also alerts desktop via MQTT protocol using MQTTX App

ALGORITHM FOR WORKING

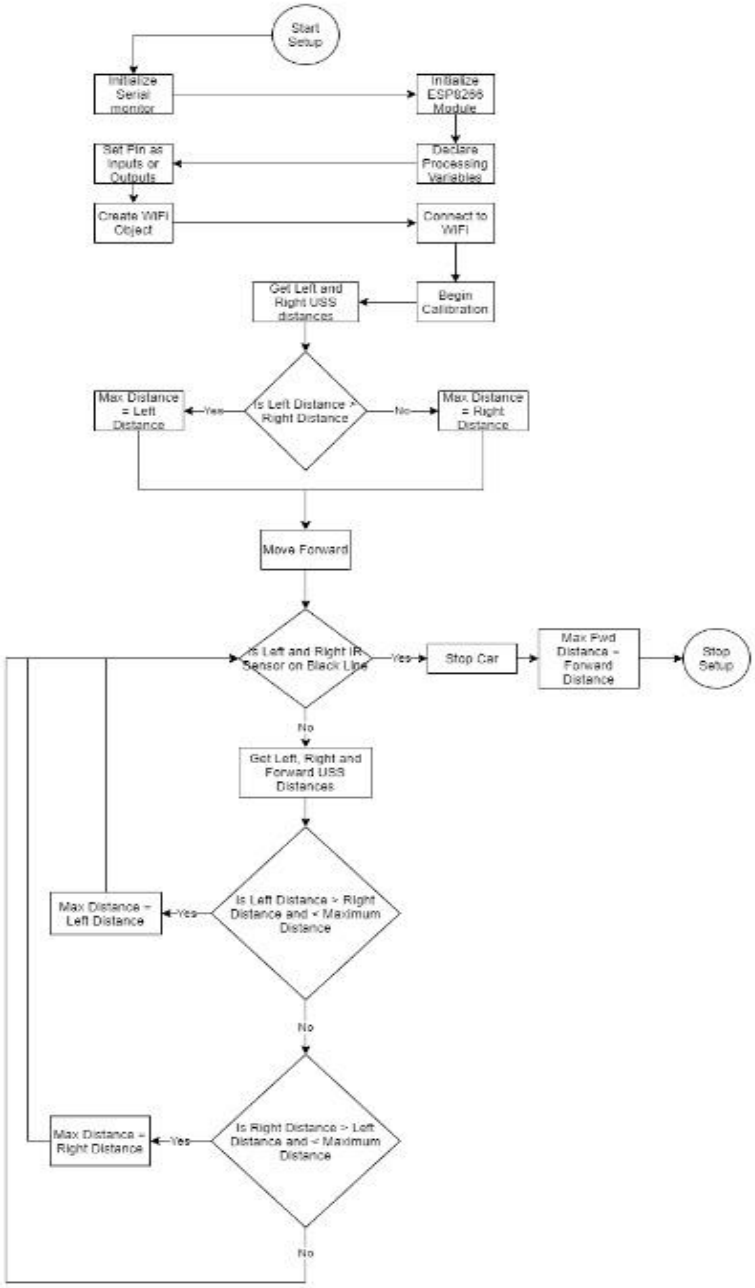

Fig7. Algorithm for working (1)

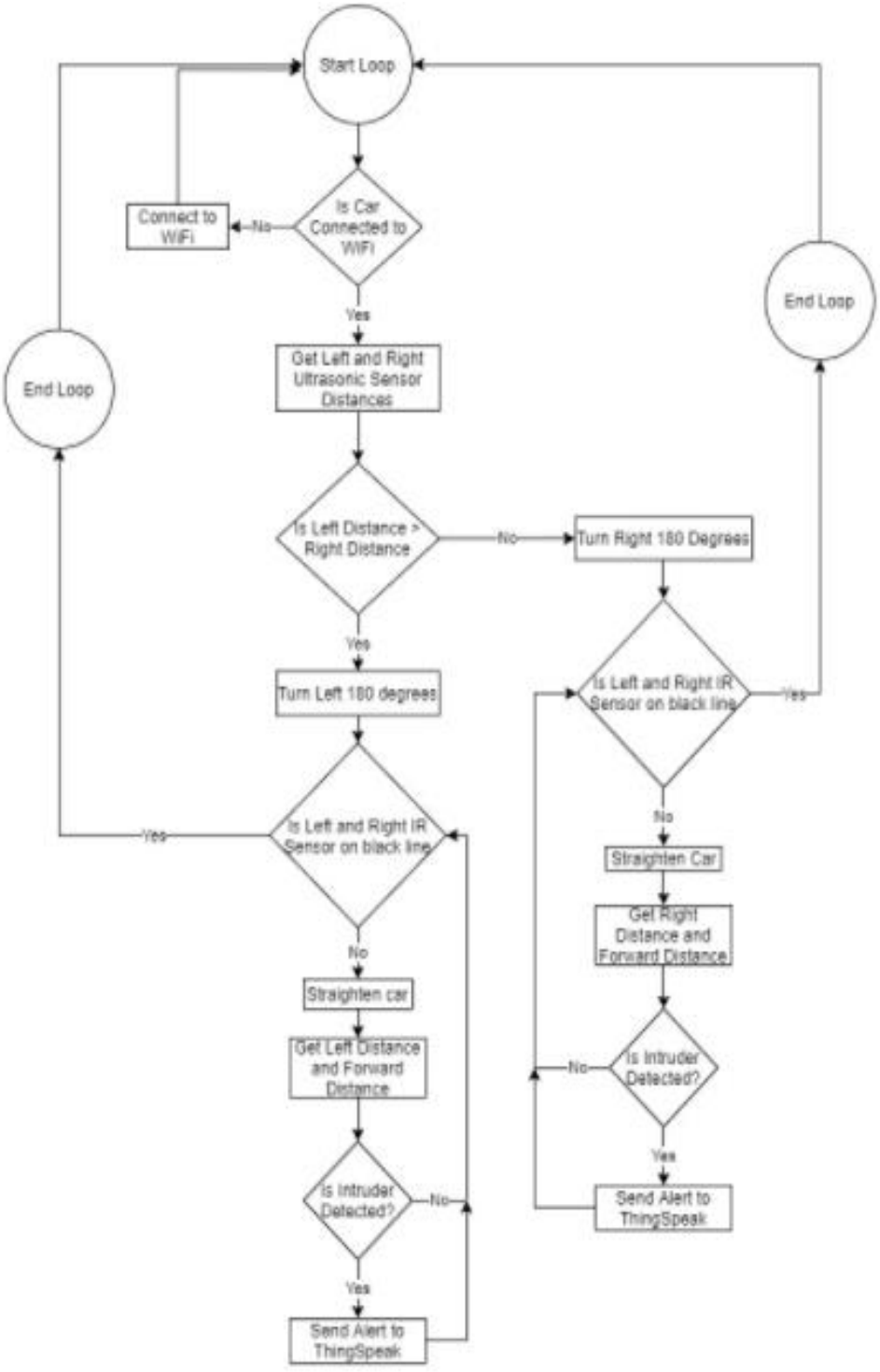

Fig8. Algorithm for working (2)

CIRCUIT DIAGRAM OF PATROL BOT

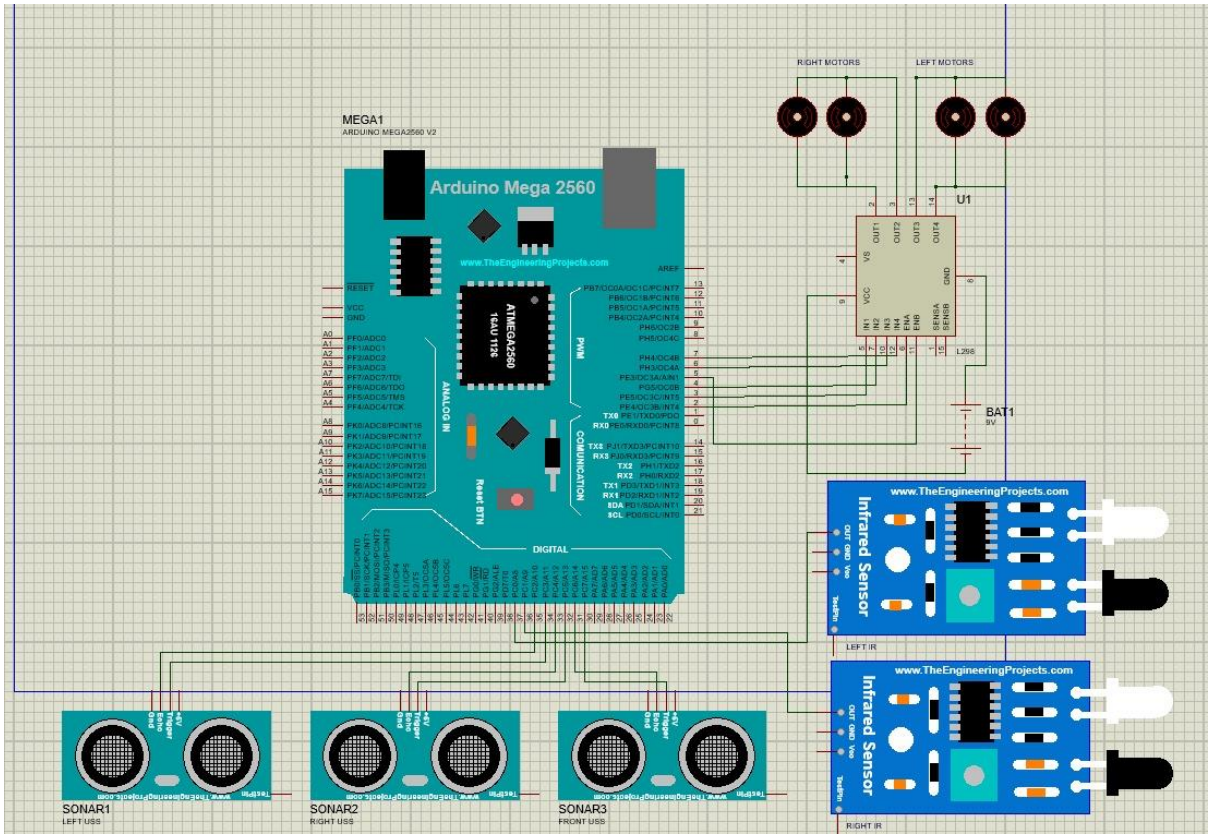

Fig9. Circuit Diagram on Proteus

SYSTEM DESIGN

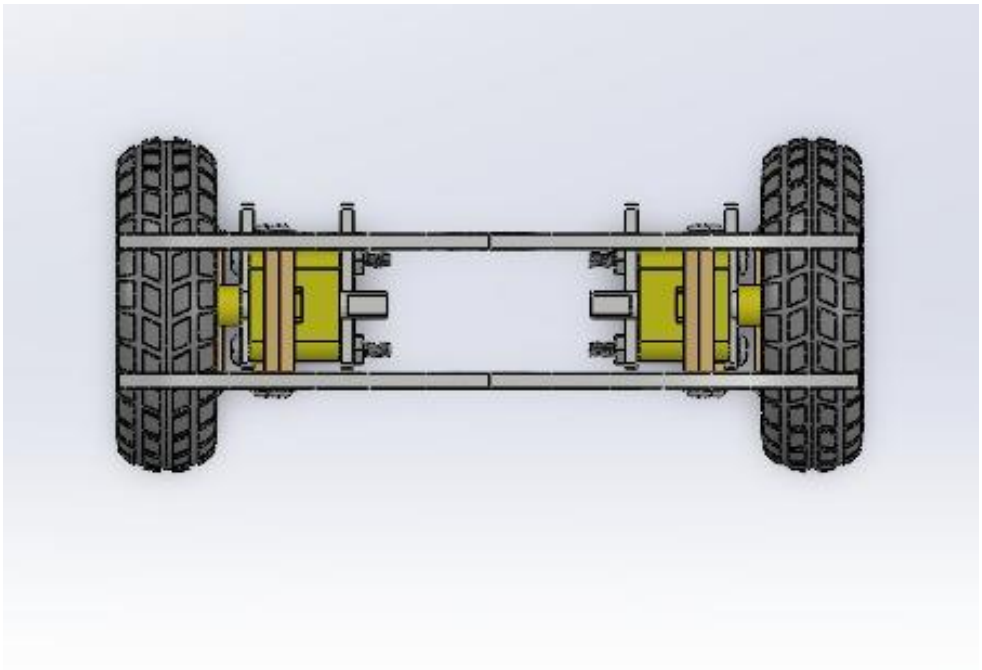

Fig 10. Front View

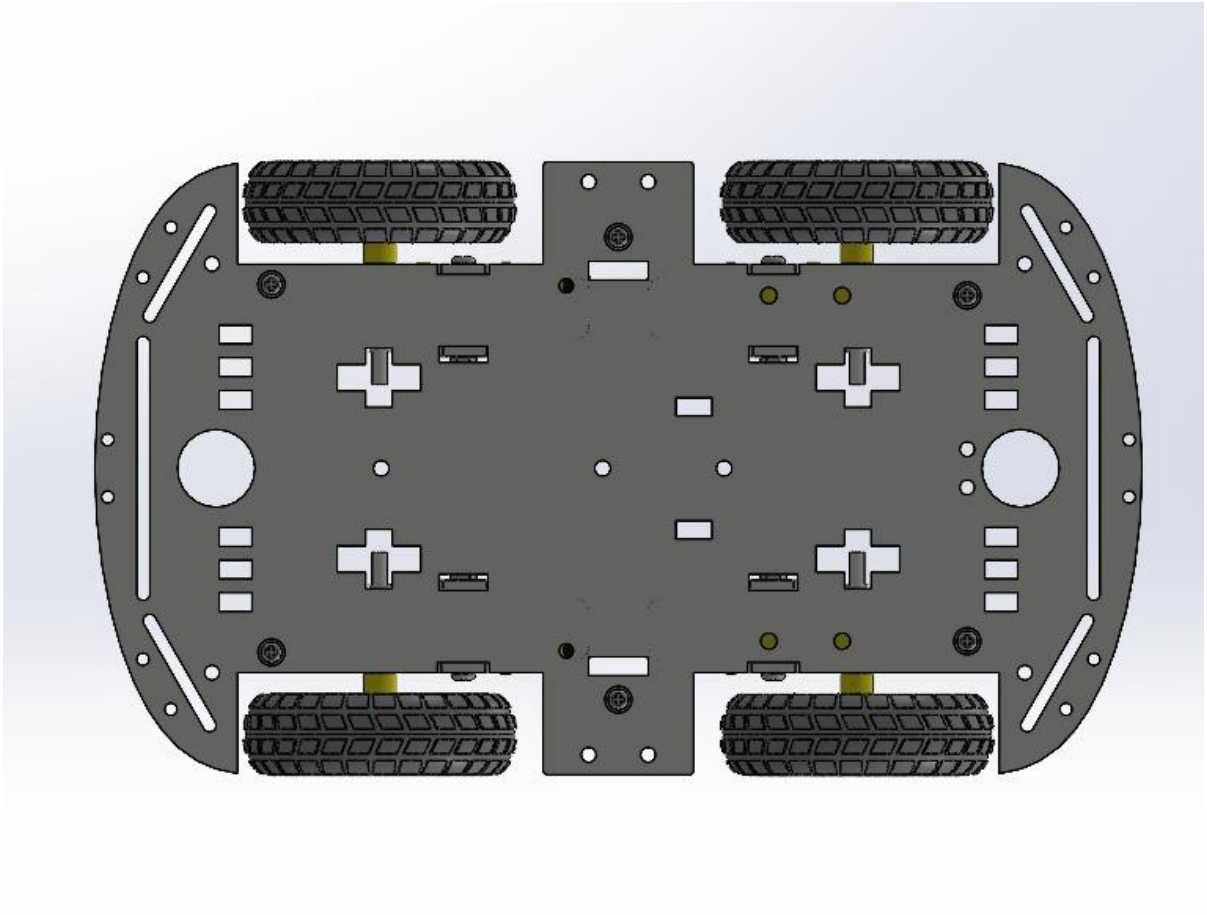

Fig 11. Top View

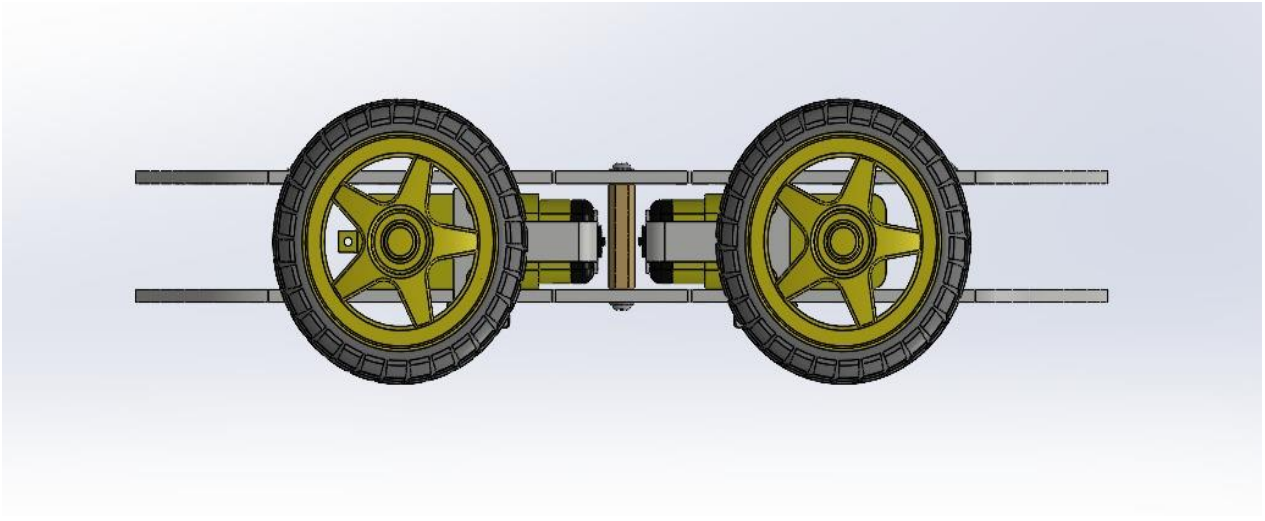

Fig 12. Side View

CODE TO IMPLEMENT THE SYSTEM

```
#include "WiFiEsp.h"
#include "ThingSpeak.h"

//Defining IR sensor Values
#define black 1
#define white 0
// Front Ultrasonic Sensor
#define frontTrig 30
#define frontEcho 31
// Right Ultrasonic Sensor
#define rTrig 32
#define rEcho 33
// Left Ultrasonic Sensor
#define lTrig 34
#define lEcho 35
//IR Sensor
#define senR 36
#define senL 37
// Right Motor connections
#define rmspd 2
#define rmp 3
#define rmm 4
// Left Motor connections
#define lmspd 5
#define lmp 6
#define lmm 7
//Wifi Details
char ssid[] = "ssid";     //  your network SSID (name)
char pass[] = "pass";   // your network password
int status = WL_IDLE_STATUS; // Initial wifi status
// Creating Wifi module to connect to
WiFiEspClient  client;
//Thingspeak Channel Details
unsigned long myChannelNumber = 1525318;
const char * myWriteAPIKey = "OTJI3J9D2FQPBYS9";
const int field = 1;
int flag=0;
int x;
// Right IR Sensor value
int Rread= 0;
// Left IR Sensor value
int Lread= 0;
//USS distsnces
double ld=0;
double rd=0;
double fd=0;
double fwdduration=0;
double fwddistance=0;
double lduration=0;
double ldistance=0;
double rduration=0;
double rdistance=0;
//Max Distance
double md=0;
double mfd=0;
void setup()
{
    //Initialize serial for debugging
    Serial.begin(115200);
```

```
    // initialize serial for ESP module
    Serial3.begin(115200);
    // initialize Wifi module
    WiFi.init(&Serial3);
    // Initialize ThingSpeak
    ThingSpeak.begin(client);
    //connect to wifi
    startwifi();
    // Set all the motor control pins to outputs
    pinMode(rmp, OUTPUT);
    pinMode(rmm, OUTPUT);
    pinMode(rmspd, OUTPUT);
    pinMode(lmp, OUTPUT);
    pinMode(lmm, OUTPUT);
    pinMode(lmspd, OUTPUT);
    //Set IR Sensor pins to input.(IN ACTUAL IR SENSOR)
    pinMode(senR, INPUT);
    pinMode(senL, INPUT);
    //Set Ultrasonic Echo Pins to input
    pinMode(frontEcho, INPUT);
    pinMode(lEcho, INPUT);
    pinMode(rEcho, INPUT);
    //Set Ultrasonic Trigger Pins to output
    pinMode(frontTrig, OUTPUT);
    pinMode(lTrig, OUTPUT);
    pinMode(rTrig, OUTPUT);
    // Turn off motors - Initial state
    digitalWrite(rmp, LOW);
    digitalWrite(rmm, LOW);
    digitalWrite(lmp, LOW);
    digitalWrite(lmm, LOW);
    //Speed Control
    analogWrite(rmspd , 100);
    analogWrite(lmspd , 100);

   //Callibration Phase
    Serial.println("Callibration Phase Initiate....");
    ld=getLeftDistance();
    rd=getRightDistance();
    if(ld>rd)
      md=ld;
    else
      md=rd;
    fwd();
    while(digitalRead(senR)!=black && digitalRead(senL)!=black)
    {
      ld=getLeftDistance();
      rd=getRightDistance();
      fd=getFwdDistance();
      if(digitalRead(senR)==black && digitalRead(senL)==black)
      {
        mfd=fd;
      }
      if(ld>rd && ld<md)
      {
        md=ld;
      }
      else if (rd>ld && rd<md)
      {
        md=rd;
      }
```

```
      else continue;
    }
    md-=5;
    Serial.println("Callibration Complete\nMax Detection Distance in cm:");
    Serial.print(md);
    stopcar();
    delay(1000);
}
void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
    ld=getLeftDistance();
    rd=getRightDistance();
    if (ld>rd)
    {
      left();
      while(digitalRead(senL) != black)
      {
        left();
      }
      stopcar();
      straighten();
      while (digitalRead(senR)!=black && digitalRead(senL)!=black)
      {
        straighten();
        ld=getLeftDistance();
        fd=getFwdDistance();
        if (ld<md || fd<mfd+1 || fd<mfd-1)
        {
          flag=1;
          alert();
        }
        else continue;
      }
    }
    if (rd>ld)
    {
      right();
      while(digitalRead(senR) != black)
      {
        right();
      }
      stopcar();
      straighten();
      while (digitalRead(senR)!=black && digitalRead(senL)!=black)
      {
        straighten();
        rd=getRightDistance();
        fd=getFwdDistance();
        if (rd<md-1 || fd<mfd+1 || fd<mfd-1)
        {
          flag=1;
          alert();
        }
        else continue;
      }
    }
  }
  else
  {
```

```
    startwifi();
  }
}
void alert()
{
  // Write to ThingSpeak.
  int x = ThingSpeak.writeField(myChannelNumber, field, flag,
myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " +
String(x));
  }
  flag=0;
}
void startwifi()
{
  Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while(WiFi.status() != WL_CONNECTED)
    {
      WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network.
      Serial.print(".");
      delay(5000);
    }
}
void fwd() //All wheels spinning in Anti - Clockwise direction
{
    digitalWrite(rmp, HIGH);
    digitalWrite(rmm, LOW);
    digitalWrite(lmp, HIGH);
    digitalWrite(lmm, LOW);
}
void stopcar() //All wheels stop spinning
{
    digitalWrite(rmp, LOW);
    digitalWrite(rmm, LOW);
    digitalWrite(lmp, LOW);
    digitalWrite(lmm, LOW);
}
void left() //Right wheels spin Anti - Clockwise, left wheels spin
Clockwise
{
    digitalWrite(rmp, HIGH);
    digitalWrite(rmm, LOW);
    digitalWrite(lmp, LOW);
    digitalWrite(lmm, HIGH);
}
void right() //left wheels spin Anti - Clockwise, right wheels spin
Clockwise
{
    digitalWrite(rmp, LOW);
    digitalWrite(rmm, HIGH);
    digitalWrite(lmp, HIGH);
    digitalWrite(lmm, LOW);
}
void straighten()
{
  if (digitalRead(senR)!= white)
```

```
  {
    while (digitalRead(senR)!= white && digitalRead(senL)!= white)
    {
      right();
    }
    fwd();
   }
   else if (digitalRead(senL)!= white)
   {
     while (digitalRead(senR)!= white && digitalRead(senL)!= white)
     {
       left();
     }
       fwd();
     }
}
double getFwdDistance()
{
  digitalWrite (frontTrig, LOW); //write trigger pin low to get a clean
high pulse
  delayMicroseconds(2); //Low for 2 microseconds
  digitalWrite (frontTrig, HIGH);// Set to high for 10 microseconds
  delayMicroseconds(10);
  digitalWrite (frontTrig, LOW);// set back to low
  fwdduration = pulseIn(frontEcho, HIGH); //get the time for which echopin
was high (US ray did not bounce back)
  fwddistance = fwdduration * 0.034 / 2; //s=vt (v= speed of sound,
t=duration) we multiply by 2 because time corresponds to US ray travellin
the distance twice
  return fwddistance; //return value of distance calculated
}
double getRightDistance()
{
  digitalWrite (rTrig, LOW);
  delayMicroseconds(2);
  digitalWrite (rTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite (rTrig, LOW);
  rduration = pulseIn(rEcho, HIGH);
  rdistance = rduration * 0.034 / 2;
  return rdistance;
}
double getLeftDistance()
{
  digitalWrite (lTrig, LOW);
  delayMicroseconds(2);
  digitalWrite (lTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite (lTrig, LOW);
  lduration = pulseIn(lEcho, HIGH);
  ldistance = lduration * 0.034 / 2;
  return ldistance;
}
```

COMPONENTS:

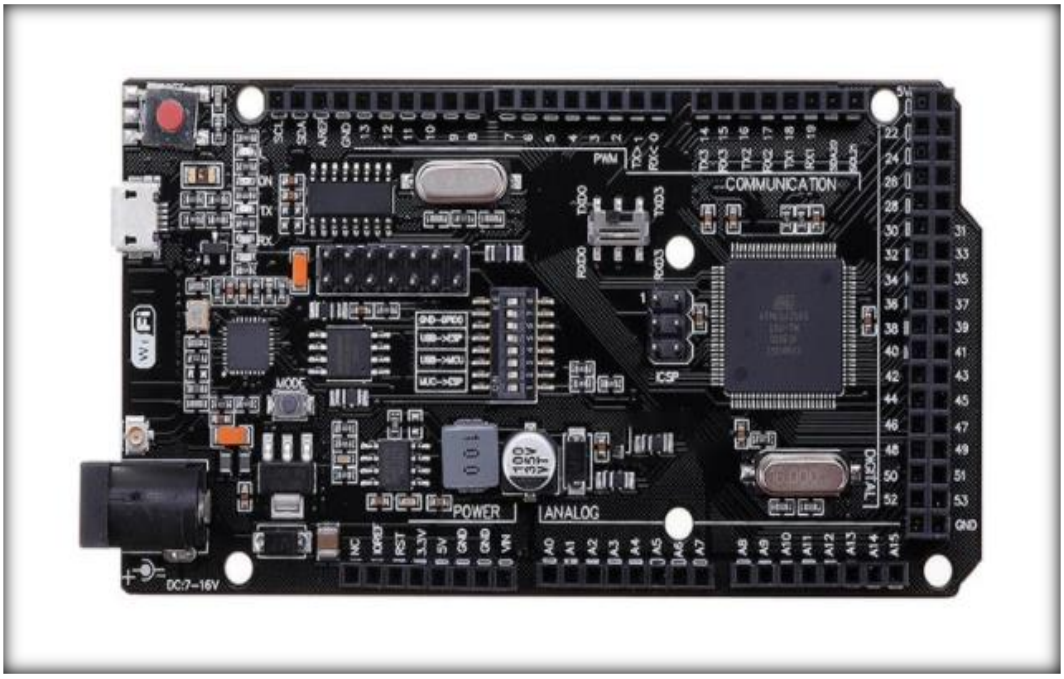

Fig 13. Arduino MEGA

1. It is a customized version of the classic ARDUINO MEGA R3 board. Full integration of Atmel ATmega2560 microcontroller and ESP8266 Wi-Fi IC, with 32 Mb (megabits) of flash memory, and CH340G USB-TTL converter on a single board! All components can be set up to work together or independently. All of the modules can work together or separately. And everyone has their own pinout headers. The convenient solution for the development of new projects requiring Uno and Wi-Fi. Via USB you can update sketches and firmware for ATmega328 and for ESP8266.For this on-board have the USB-serial converter CH340G.

Fig 14 .Battery Source

2. Power Supply: An alkaline battery (IEC code: L) is a type of primary battery that derives its energy from the reaction between zinc metal and manganese dioxide. Compared with zinc–carbon batteries of the Leclanché cell or zinc chloride types, alkaline batteries have a higher energy density and longer shelf life, yet provide the same voltage. The alkaline battery gets its name because it has an alkaline electrolyte of potassium hydroxide (KOH) instead of the acidic ammonium chloride (NH4Cl) or zinc chloride (ZnCl2) electrolyte of the zinc–carbon batteries. Other battery systems also use alkaline electrolytes, but they use different active materials for the electrodes.

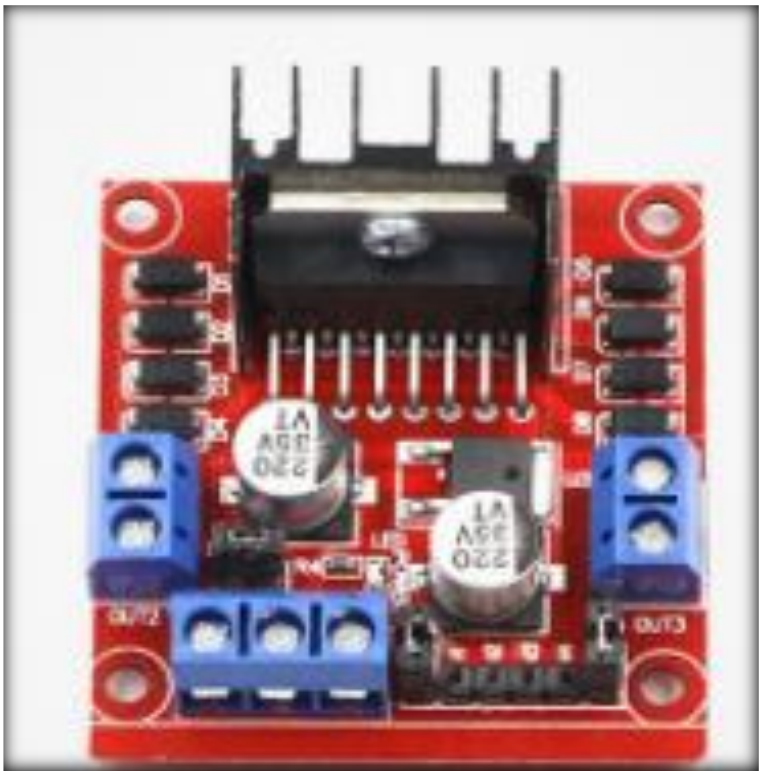

Fig 15. Motor Driver

3. H-bridge is a circuit which allows the voltage to be flown in either direction Dual H-bridge Motor Driver integrated circuit IC). This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions. It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes

Fig 16. Ultrasonic Sensors

4. Ultrasonic Sensor: An ultrasonic sensor sends out an ultrasonic pulse of a prefixed frequency. If there is an obstacle in the path of the pulse, the waves gets reflected back and are again imposed on the receiver of the sensor. Depending on the time taken for the waves to reach back to the sensor, the range of the obstacle is calculated.

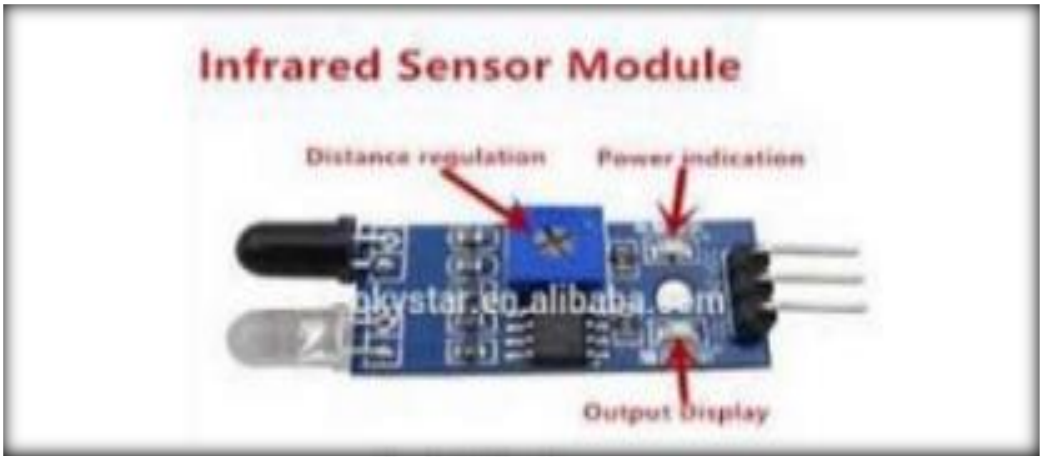

Fig 17. Infrared Sensors

5. IR infrared sensor IR infrared sensor module operates on operational infrared sight programmed control options. it has high dependability , low power consumption , sensitivity variable , task mode low voltage . generally used as part of various electrical gears for automated detection , especially for battery powered and controlled systems

Fig 18. HDPP wheel

6. Wheels The hubs are HDPP (High density polypropylene) and the tyres are real rubber for good grip. The tyres have foam inserts to provide support and keep the tyres round for smooth running. These are the same wheels used for 2wd and 4wd Magician Chassis. These wheels are designed to be a press fit to the shafts of our plastic gearboxes and our plastic gearbox brackets.

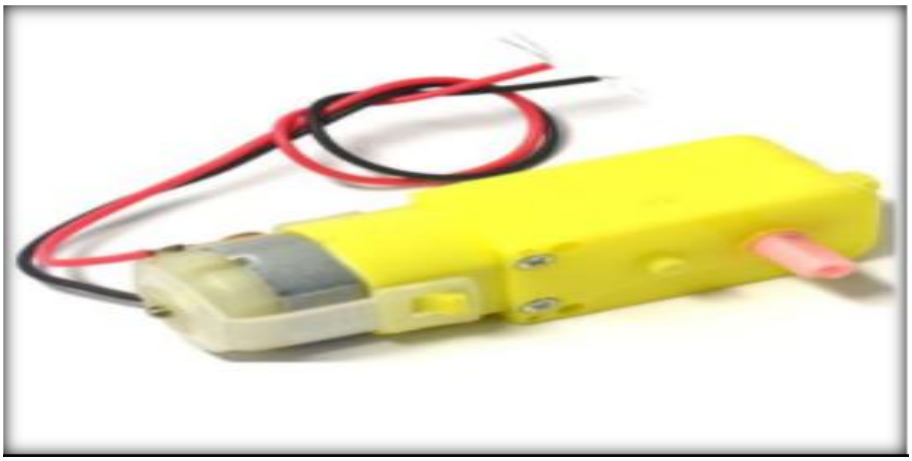

Fig 19. BO motors

7. BO Motors This BO Motor is made up of silicon gear and it is light-weight. It gives good torque. BO Motors are called as Battery Operated motors. Hence it is used in many DIY Robotic applications that run on batteries. It can run on a 6v-9v battery. If you are using more than 1 motor in your project then you can use it with 12V Battery also.
   The line follower robots and few of the project needs a slow rpm motor. Because it will be sturdy when following the line. So you may have to choose a slower RPM BO Motors like a 60 RPM BO Motor. It will have good torque also.

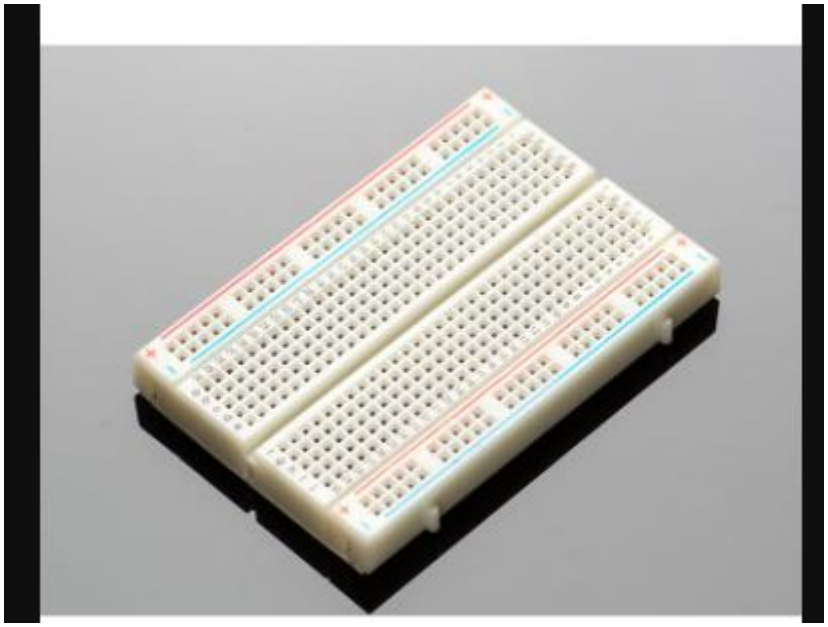

Fig 20. BreadBoard

8. A breadboard, or protoboard, is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used when slicing bread In the 1970s the solderless breadboard (a.k.a. plug board, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these. Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also popular with students and in technological education. Compared to more permanent circuit connection methods, modern breadboards have high parasitic capacitance, relatively high resistance, and less reliable connections, which are subject to jostle and physical degradation. Signalling is limited to about 10 MHz, and not everything works properly even well below that frequency.

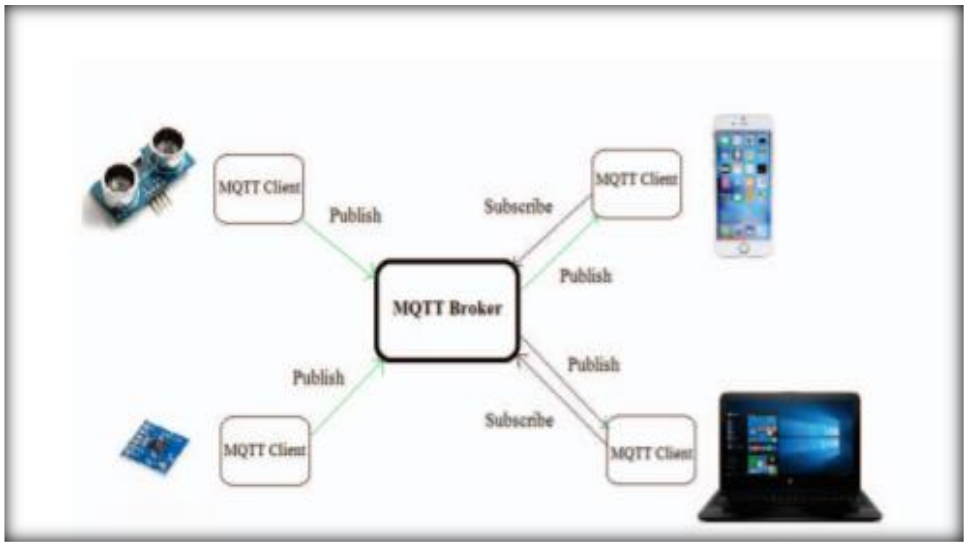

Fig 21. MQTT

WORKING

TEST RESULTS

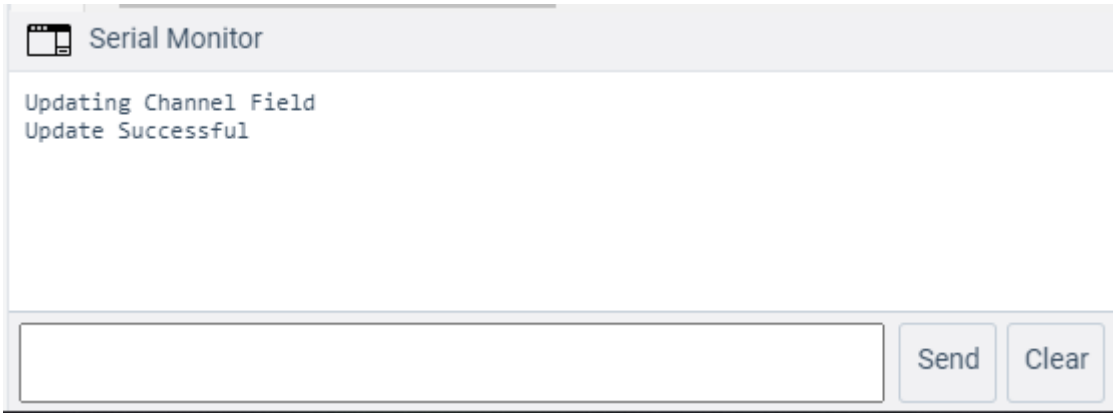

Fig 22 . Code sent from Microcontroller

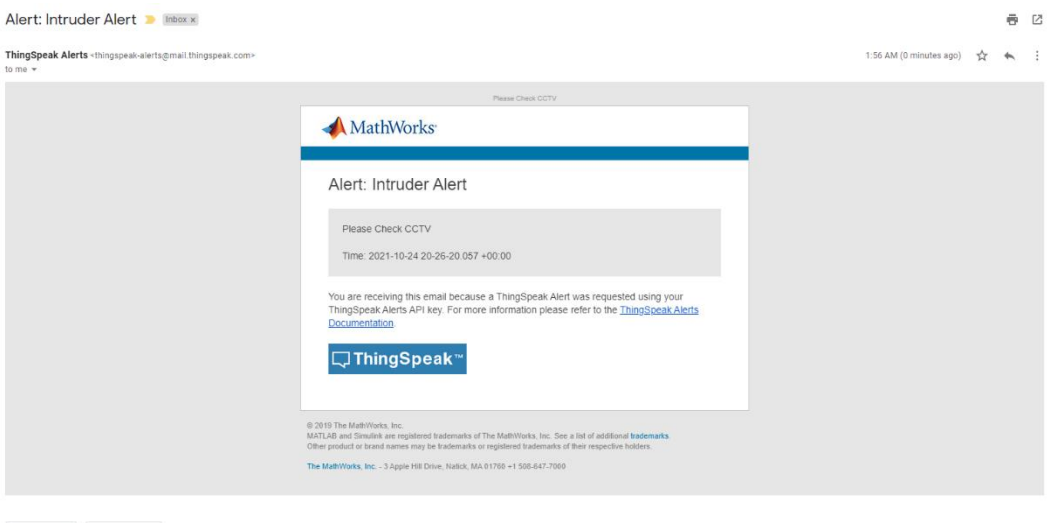

Fig 23 . Email alert

Fig 24 . twitter alert

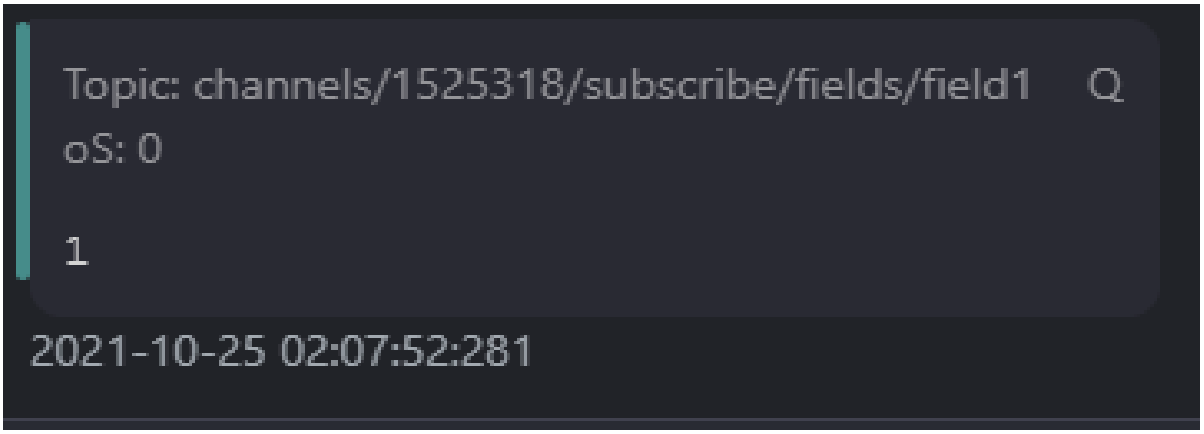

Fig 25 . MQTT alert

Error using
matlab.internal.webservices.HTTPConnector/copyContentToByteArr
ay (line 373)

The server returned the status 429 with message "Too Many
Requests" in response to the request to URL
https://api.thingspeak.com/alerts/send.

Fig 26 . Email spam error

Experiment to determine actual range of HC SR04 Ultrasonic Sensor:

Apparatus: Arduino Uno, Breadboard, HC SR04 USS, measuring tape

Circuit Diagram:

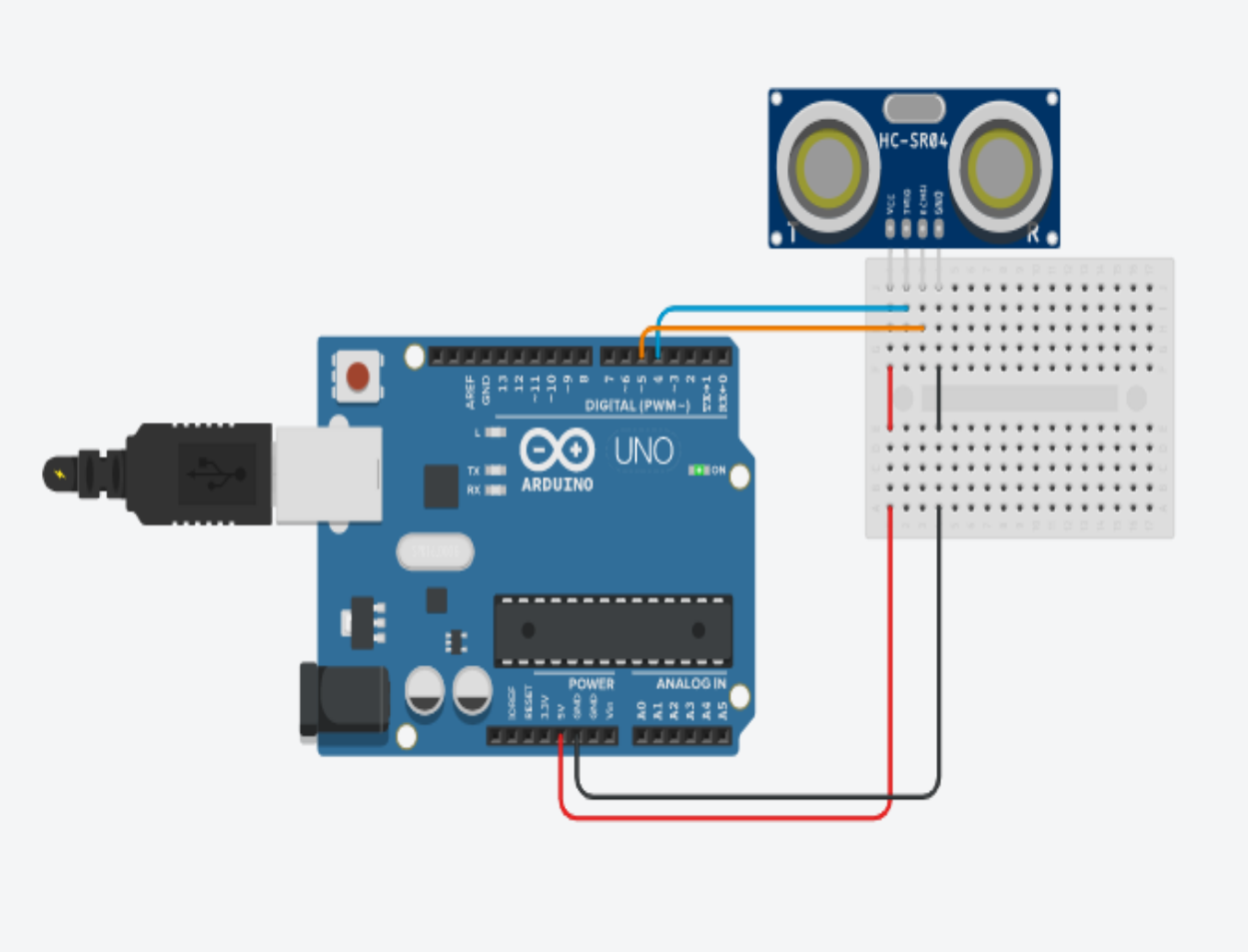

Fig 27. Test circuit diagram

CODE

```
#define trig 4
#define echo 5
double x,duration,distance;
char y;

void setup()
{

  Serial.begin(9600);
  pinMode(echo, INPUT);
  pinMode(trig, OUTPUT);

}

void loop()
{
  while(Serial.available())
  {
    y = Serial.read();
    if (y== 'n')
    {
      x=getDistance();
      Serial.println(x);
    }
  }
}

double getDistance()

{

  digitalWrite (trig, LOW);

//write trigger pin low to get a clean high pulse

  delayMicroseconds(2); //Low for 2 microseconds
  digitalWrite (trig, HIGH);// Set to high for 10 microseconds
  delayMicroseconds(10);
  digitalWrite (trig, LOW);// set back to low
  duration = pulseIn(echo, HIGH);

//get the time for which echopin was high (US ray did not bounce back)

  distance = duration * 0.034 / 2;

//s=vt (v= speed of sound, t=duration) we multiply by 2 because time
corresponds to US ray travellig the distance twice

 return distance; //return value of distance calculated

}
```

Observations

Sr. No	Actual (cm)	Observed (cm)	Error (cm)	Error Percentage (%)
1	100.3	99.18	1.11	1.11
2	200.3	197.80	2.48	1.24
3	301.3	297.33	3.95	1.31
4	400.6	395.27	5.33	1.33
5	500.5	493.74	6.76	1.35
6	600.2	591.86	8.34	1.39
7	700	690.13	9.87	1.41
8	800.3	788.94	11.36	1.42
9	900.8	887.74	13.06	1.45
10	1000.5	985.69	14.8	1.48

Table 2. Calculation of Error Percentage of Ultrasonic sensors

Fig 28. Serial Monitor

Result:

The Ultrasonic sensor retains an accuracy of 98.52% at 10 meters and the accuracy gets better with the decrease in distance. A large contributor to the error are the calculations done in the program. However, These calculation errors may be ignored as the reference distances would also have these errors.

ADVANTAGES AND LIMITATIONS

The Advantages of using this patrol Bot over other options would be:

- **This Patrol Bot is an secondary system and not a primary system so it acts as a multilayer security blanket thus being one of a kind systems**
- **It is an affordable addition to a standard CCTV security system**
- **It runs on an independent power source thus power outages will not effect this security measure**
- **As it is autonomous it has negligible to no error as human error is removed**
- **The small size of this bot paired with the range of the ultrasonic sensors makes the robot almost undetectable to the intruder before setting of the alert system**
- **The alert system uses IOT mechanisms such as thingspeak, MQTT, Matlab Thingtweet and Thingreact to instantly send alerts making the reaction time almost instantaneous**
- **The multiple ways of sending alerts acts as a failsafe as if one service is down the other services will still go through**
- **The bot also has tamper detection**

Limitations:

- **The range of the Patrol Bot decreases as the distance from the furniture/objects decreases ie. extruding parts of the room will decrease the overall range of the bot**
- **The battery life of the patrol robot is limited as it works on a rechargeable lithium ion battery and therefore has to be charged before every patrol session and works on a limited timeline**
- **The alert system heavily relies on Internet service and Wi-Fi connection to send alerts across all its platforms and therefore becomes handicapped during connection issues**
- **If there is a lighting issue in the path it can lead to pathway errors as it will not be able to distinguish between different shades of black and therefore stray off path**
- **If it is knocked of course by manual or natural phenomenon it will be immobilised until human intervention due to its natural code**
- **There is a limitation to the number of alerts it can send at a certain time therefore in-case of alerts extending the threshold it will cease to send alerts**

CONCLUSION

This robot can be a very important addition to existing security systems which include mostly just a CCTV monitoring station. As there is a margin of human error in low Grade security systems such as commercial shops or Buildings ,

This patrol Bot was designed to be a secondary cost effective security system which would integrate very easily with their own current standing Security systems and add a blanket of security.

This project was modelled successfully on Proteus and solidworks and IOT system was implemented using ThingSpeak IOT platform which is again cost effective as it is open sourced.

FUTURE WORK

The Patrol Bot can be implemented with a camera which can be mounted on a motor in order to achieve a 360 degree view of the room which would help to visually assess the situation and use open CV libraries to detect intruders with more accuracy and push out alerts at a faster rate to ensure a quick response can be taken. The navigational system can be further upgraded by the usage of Li-DAR technology which would map put the entire room and set a path for itself while avoiding obstacles through complex navigational paths and use ROS(robot operating system) to automate movement of the patrol robots. We can also unify the alert system by making an autonomous app which pushes notifications on desktop as well as mobile platforms. To aid in the navigational process we can add mecanum wheels which will help the bot to turn and sharp angles in tight spaces further increasing its navigational abilities.

REFERENCES

- Patel, Neel and Tank, Darshan and Patel, Keyur and Adatkar, Rashmi and Thosar, Rohan, Vehicle Accident Alert System (April 8, 2019). 2nd International Conference on Advances in Science & Technology (ICAST) 2019 on 8th, 9th April 2019 by K J Somaiya Institute of Engineering & Information Technology, Mumbai, India, Available at SSRN: https://ssrn.com/abstract=3368089 or http://dx.doi.org/10.2139/ssrn.3368089

- Pakdaman, Mehran & Sanaatiyan, Mohammad Mehdi & Rezaei, Mahdi. (2010). A line follower robot from design to implementation: Technical issues and problems. 5 - 9. 10.1109/ICCAE.2010.5451881.

- K. Muheden, E. Erdem and S. Vançin, "Design and implementation of the mobile fire alarm system using wireless sensor networks," 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI), 2016, pp. 000243-000246, doi: 10.1109/CINTI.2016.7846411.

- Internet of Things Based Indoor Smart Surveillance and Monitoring System using Arduino and Raspberry Pi N C Sendhilkumar *et al* 2021 *J. Phys.: Conf. Ser.* 1964 062083

- Khaing, Khin Kyawt Kyawt. "Temperature and Humidity Monitoring and Control System with Thing Speak." *International Journal of Scientific Research and Engineering Development* (2019).

- Kumar, Amit, et al. "Border Security System using Arduino & Ultrasonic Sensors." *International journal of scientific engineering and technology research* 6 (2017): 2489-2492.