SRI KRISHNA COLLEGE OF TECHNOLOGY

(Accredited by NAAC with A Grade)
Kovaipudur, Coimbatore-641 042

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

18IT067
INTERNET OF THINGS
LAB MANUAL

PREPARED BY	VERIFIED BY
Ms. G. Sandhya	Dr. P. Tamije Selvy
Ms. S. Soundarya	HOD/CSE

2021-22 ODD

18IT067 INTERNET OF THINGS 3 0 2 4

OBJECTIVES

- To impart knowledge on design and development of IoT based system design.

COURSE CONTENTS

1. Design and development of CO2 Emission Identification System.
2. Design and development of Automatic Irrigation System.
3. Design and Development of Theft Identification Alert System.
4. Design and Development of Health Abnormal Alert System.
5. Design and Development of Alcohol Detector.
6. Design and Development of Water Tank Level Monitoring System.
7. Design and Development of Rain Fall Level Detection System.
8. Design and Development of Gas Leakage Monitoring System.

LIST OF EXPERIMENTS

S.NO.	TITLE	CO/PO MAPPING
1.	CO2 EMISSION IDENTIFICATION SYSTEM	CO1, CO2, CO3, CO4
2.	AUTOMATIC IRRIGATION SYSTEM	CO1, CO2, CO3, CO4
3.	THEFT IDENTIFICATION ALERT SYSTEM	CO1, CO2, CO3, CO4
4.	HEALTH ABNORMAL ALERT SYSTEM	CO1, CO2, CO3, CO4
5.	ALCOHOL DETECTOR	CO1, CO2, CO3, CO4
6.	WATER TANK LEVEL MONITORING SYSTEM	CO1, CO2, CO3, CO4
7.	RAIN FALL LEVEL DETECTION SYSTEM	CO1, CO2, CO3, CO4
8.	GAS LEAKAGE MONITORING SYSTEM	CO1, CO2, CO3, CO4

CO2 EMISSION IDENTIFICATION SYSTEM

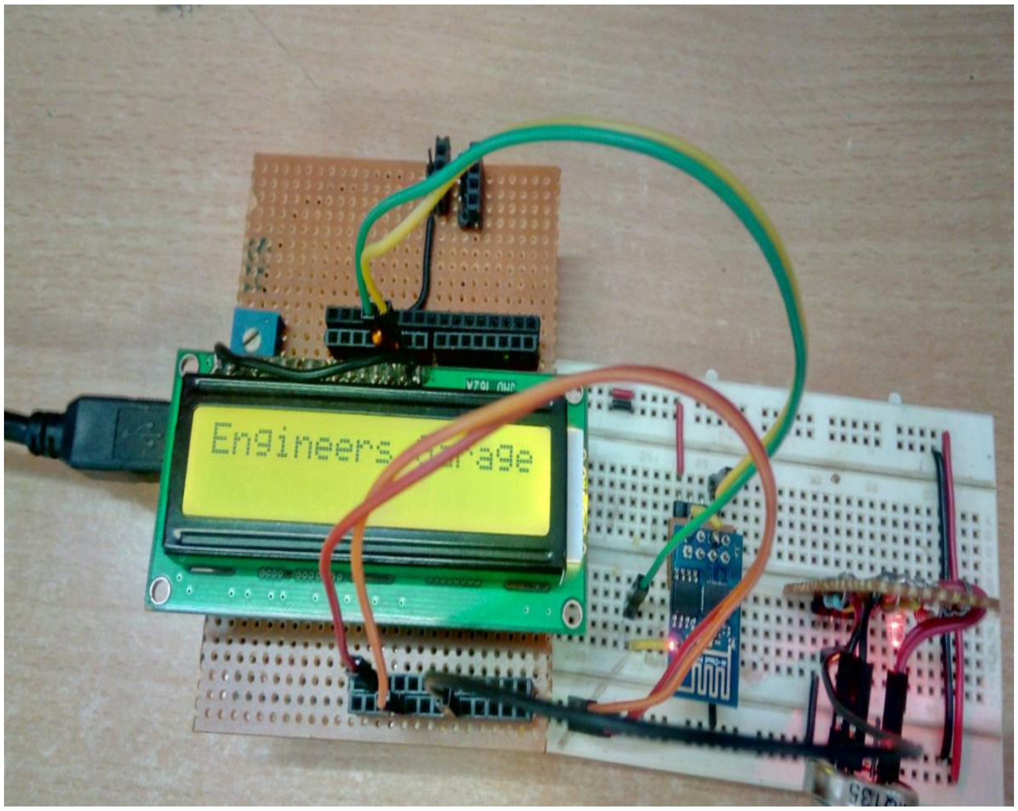
OBJECTIVE:

To develop an IOT Based CO2 Emission Identification System in which we will monitor the Air Quality over a web server using internet and will trigger an alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO2, smoke, alcohol, benzene and NH3.

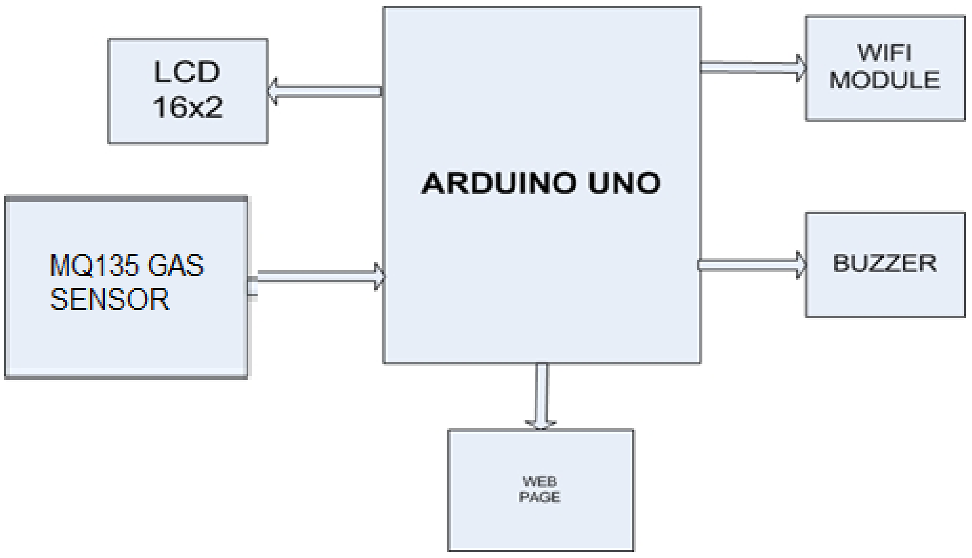
REQUIRED COMPONENTS:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- Connecting wires

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

- The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases, so it is perfect gas sensor for our Air Quality Monitoring Project.
- When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million).
- MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in “Code Explanation” section below.
- Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM.
- When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”.
- Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”

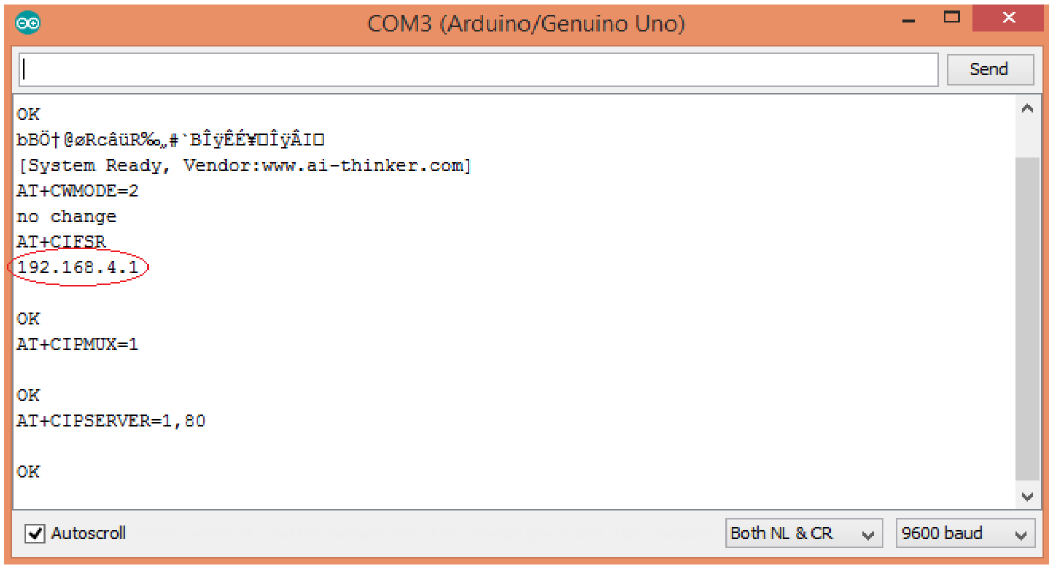
PROGRAM:

```
#include "MQ135.h"
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10); // This makes pin 9 of Arduino as RX pin and pin 10 of Arduino
as the TX pin
const int sensorPin= 0;
int air_quality;
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11, 5, 4, 3, 2);
void setup() {
pinMode(8, OUTPUT);
lcd.begin(16,2);
lcd.setCursor (0,0);
lcd.print ("circuitdigest ");
lcd.setCursor (0,1);
lcd.print ("Sensor Warming ");
delay(1000);
Serial.begin(115200);
esp8266.begin(115200); // your esp's baud rate might be different
sendData("AT+RST\r\n",2000,DEBUG);  // reset module
sendData("AT+CWMODE=2\r\n",1000,DEBUG);  // configure as access point
sendData("AT+CIFSR\r\n",1000,DEBUG);  // get ip address
sendData("AT+CIPMUair_quality=1\r\n",1000,DEBUG); // configure for multiple connections
sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG);  // turn on server on port 80
pinMode(sensorPin, INPUT);        //Gas sensor will be an input to the arduino
lcd.clear();
}
void loop() {
MQ135 gasSensor = MQ135(A0);
float air_quality = gasSensor.getPPM();
```

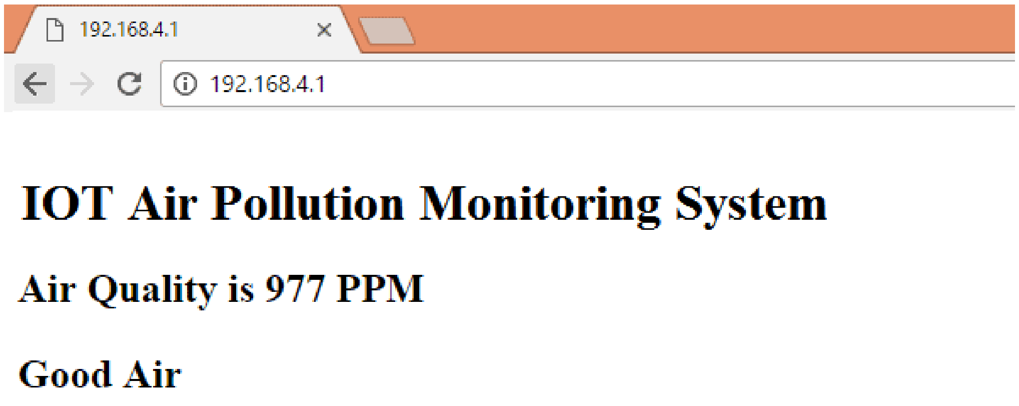
```
if(esp8266.available()) // check if the esp is sending a message
{
if(esp8266.find("+IPD,"))
{
delay(1000);
int connectionId = esp8266.read()-48; /* We are subtracting 48 from the output because the
read() function returns the ASCII decimal value and the first decimal number which is 0 starts at
48*/
String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";
webpage += "<p><h2>";
webpage+= " Air Quality is ";
webpage+= air_quality;
webpage+=" PPM";
webpage += "<p>";
if (air_quality<=1000)
{
webpage+= "Fresh Air";
}
else if(air_quality<=2000 && air_quality>=1000)
{
webpage+= "Poor Air";
}
else if (air_quality>=2000 )
{
webpage+= "Danger! Move to Fresh Air";
}
webpage += "</h2></p></body>";
String cipSend = "AT+CIPSEND=";
cipSend += connectionId;
cipSend += ",";
cipSend +=webpage.length();
cipSend +="\r\n";
sendData(cipSend,1000,DEBUG);
sendData(webpage,1000,DEBUG);
cipSend = "AT+CIPSEND=";
cipSend += connectionId;
cipSend += ",";
cipSend +=webpage.length();
cipSend +="\r\n";
String closeCommand = "AT+CIPCLOSE=";
closeCommand+=connectionId; // append connection id
closeCommand+="\r\n";
sendData(closeCommand,3000,DEBUG);
}  }
lcd.setCursor (0, 0);
lcd.print ("Air Quality is ");
lcd.print (air_quality);
lcd.print (" PPM ");
lcd.setCursor (0,1);
if (air_quality<=1000)
{
lcd.print("Fresh Air");
digitalWrite(8, LOW);
}
else if( air_quality>=1000 && air_quality<=2000 )
{
lcd.print("Poor Air, Open Windows");
digitalWrite(8, HIGH );
}
else if (air_quality>=2000 )
{
lcd.print("Danger! Move to Fresh Air");
```

```
digitalWrite(8, HIGH);    // turn the LED on
}
lcd.scrollDisplayLeft();
delay(1000);
}
String sendData(String command, const int timeout, boolean debug)
{
String response = "";
esp8266.print(command); // send the read character to the esp8266
long int time = millis();
while( (time+timeout) > millis())
{
while(esp8266.available())
{
// The esp has data so display its output to the serial window
char c = esp8266.read(); // read the next character.
response+=c;
}      }
if(debug)
{
Serial.print(response);
}
return response;
}
```

OUTPUT:



```
OK
bBÖ†@øRcâüR‰„#`BÎÿÊÉ¥□ÎÿÂI□
[System Ready, Vendor:www.ai-thinker.com]
AT+CWMODE=2
no change
AT+CIFSR
192.168.4.1

OK
AT+CIPMUX=1

OK
AT+CIPSERVER=1,80

OK
```



IOT Air Pollution Monitoring System

Air Quality is 977 PPM

Good Air

RESULT:

Thus, an IOT Based CO2 Emission Identification System has been designed to monitor the quality of air in the environment with the help of Arduino uno kit.

AUTOMATIC PLANT IRRIGATION SYSTEM

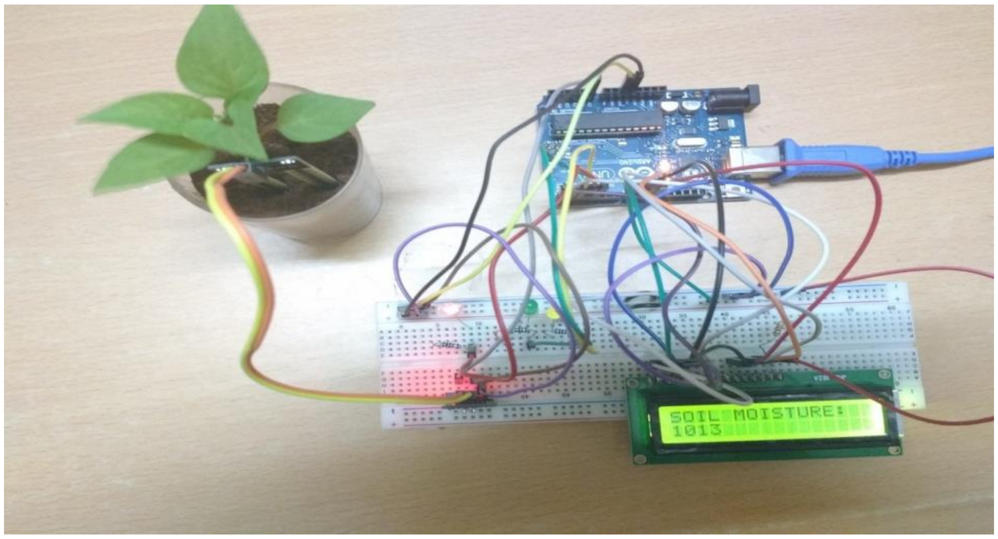
OBJECTIVE:

To develop an Automatic Plant Irrigation System using Arduino Uno, which automatically provides water to your plants and keep us updated by sending message.

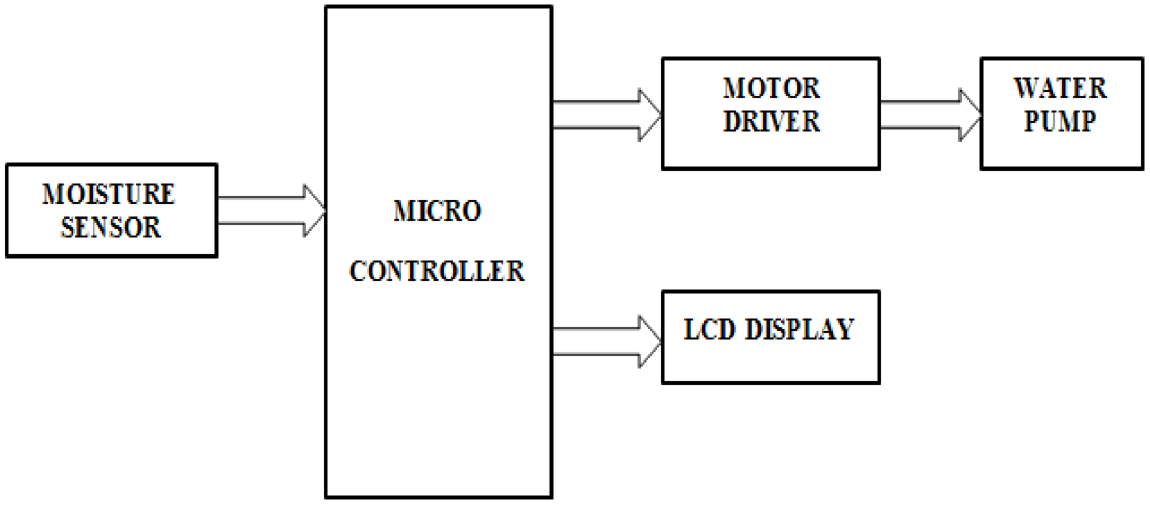
REQUIRED COMPONENTS

- Arduino Uno
- GSM Module
- Transistor BC547 (2)
- Connecting wires
- 16x2 LCD (optional)
- Power supply 12v 1A
- Relay 12v
- Water cooler pump
- Soil Moisture Sensor
- Resistors (1k, 10k)
- Variable Resister (10k, 100k)
- Terminal connector
- Voltage Regulator IC LM317

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

- In this System, Soil Moisture Sensor checks the moisture level in the soil and if moisture level is low then Arduino switches on a water pump to provide water to the plant.
- Water pump gets automatically off when system finds enough moisture in the soil.
- Whenever system switched on or off the pump, a message is sent to the user via GSM module, updating the status of water pump and soil moisture.
- This system is very useful in Farms, gardens, home etc. This system is completely automated and there is no need for any human intervention.

PROGRAM:

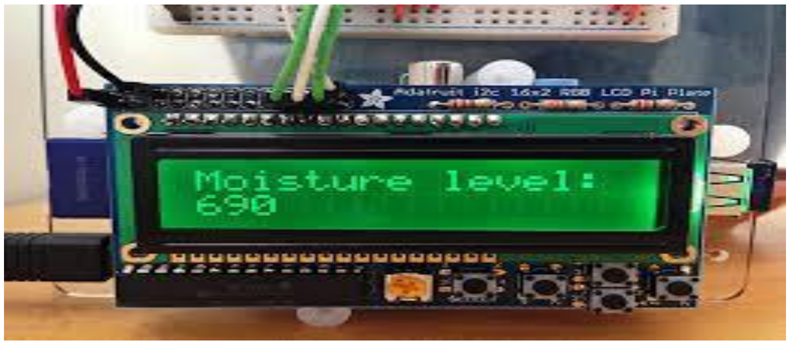
```
#include<SoftwareSerial.h>
SoftwareSerial Serial1(2,3);
#include<LiquidCrystal.h>
LiquidCrystal lcd(14,15,16,17,18,19);
int led=13;
int flag=0;
String str="";
#define motor 11
#define sensor 7
```

```
void setup()
{
lcd.begin(16,2);
Serial1.begin(9600);
Serial.begin(9600);
pinMode(led, OUTPUT);
pinMode(motor, OUTPUT);
pinMode(sensor, INPUT_PULLUP);
lcd.print("Water Irrigaton");
lcd.setCursor(4,1);
delay(2000);
lcd.clear();
lcd.print("Circuit Digest");
lcd.setCursor(0,1);
lcd.print("Welcomes You");
delay(2000);
gsmInit();
lcd.clear();
lcd.print("System Ready");
}
void loop()
{
lcd.setCursor(0,0);
lcd.print("Automatic Mode    ");
if(digitalRead(sensor)==1 && flag==0)
{
delay(1000);
if(digitalRead(sensor)==1)
{
digitalWrite(led, HIGH);
sendSMS("Low Soil Moisture detected. Motor turned ON");
lcd.begin(16,2);
lcd.setCursor(0,1);
lcd.print("Motor ON     ");
digitalWrite(motor, HIGH);
delay(2000);
flag=1;
}}
else if(digitalRead(sensor)==0 && flag==1)
{
delay(1000);
if(digitalRead(sensor)==0)
{
digitalWrite(led, LOW);
sendSMS("Soil Moisture is Normal. Motor turned OFF");
digitalWrite(motor, LOW);
lcd.begin(16,2);
lcd.print("Motor OFF");
lcd.setCursor(0,1);
lcd.print("Motor OFF");
delay(2000);
flag=0;
}}}
void sendSMS(String msg)
{
lcd.clear();
lcd.print("Sending SMS");
Serial1.println("AT+CMGF=1");
delay(500);
Serial1.print("AT+CMGS=");
Serial1.print('"');
Serial1.print("+919610126059");    // number
```

```
Serial1.print("");
Serial1.println();
delay(500);
Serial1.println(msg);
delay(500);
Serial1.write(26);
delay(1000);
lcd.clear();
lcd.print("SMS Sent");
delay(1000);
lcd.begin(16,2);
}
void gsmInit()
{
lcd.clear();
lcd.print("Finding Module..");
boolean at_flag=1;
while(at_flag)
{
Serial1.println("AT");
while(Serial1.available()>0)
{
if(Serial1.find("OK"))
at_flag=0;
}
delay(1000);
}
Serial1.println("ATE0");
lcd.clear();
lcd.print("Finding Network..");
boolean net_flag=1;
```

```
while(net_flag)
{
Serial1.println("AT+CPIN?");
while(Serial1.available()>0)
{
if(Serial1.find("READY"))
net_flag=0;
break;
}
delay(1000);
}
Serial1.println("AT+CNMI=2,2,0,0,0");
delay(1000);
Serial1.println("AT+CMGF=1");
delay(1000);
Serial1.println("AT+CSMP=17,167,0,0");
lcd.clear();
Serial1.flush();
}
```

OUTPUT:



RESULT:

Thus, an Automatic Plant Irrigation System using Arduino has been designed and implemented with help of Soil Moisture Sensor, that checks the moisture level in the soil and if moisture level is low then Arduino switches on a water pump to provide water to the plant.

THEFT IDENTIFICATION ALERT SYSTEM

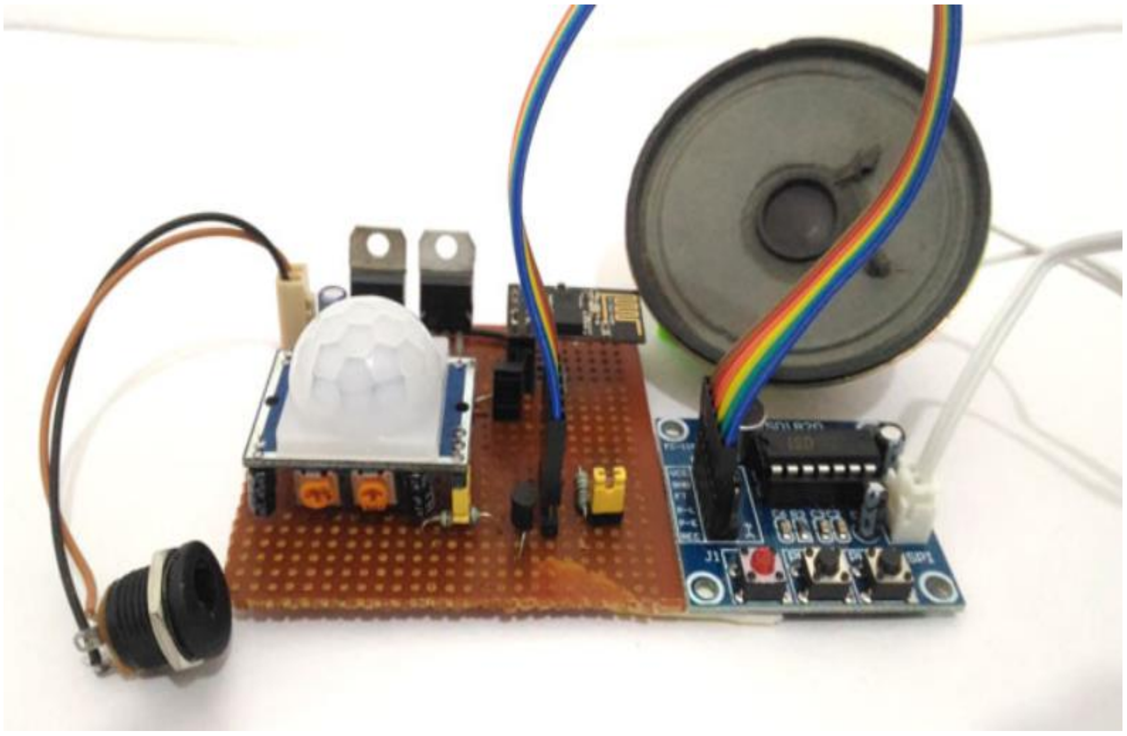
OBJECTIVE:

To build a Security system which can trigger an E-mail with Date and time to a particular E-mail ID stating the intrusion and gives alert by your own audio clip when it detects someone. This system is a fully functional Security system which can be Armed/Disarmed (Activated/De-activated) remotely via internet.

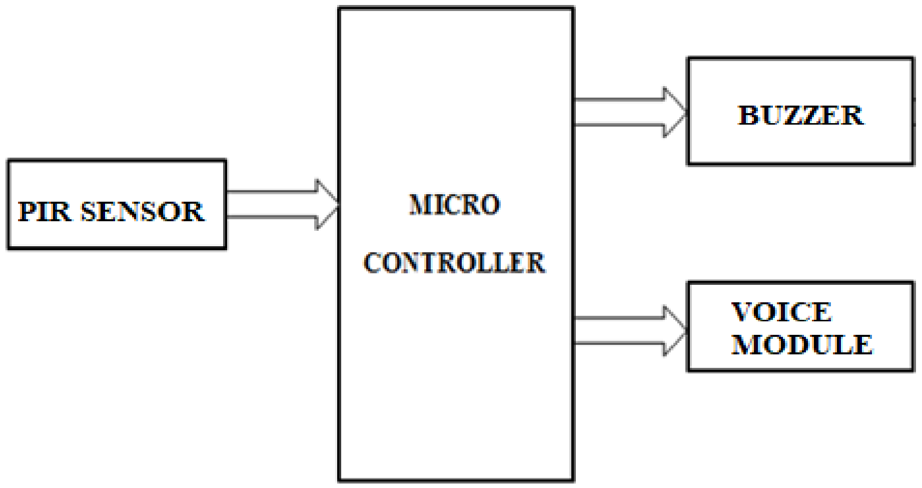
REQUIRED COMPONENTS:

- ESP8266
- PIR sensor
- ISD1820 Voice Module
- LM317, LM7805
- BC547 (2Nos)
- 1K, 200ohm, 330ohm resistors
- 10uf and 0.1uf Capacitors
- 12V adapter / 9V battery to power the setup

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



IoT Based Security System for Theft Identification

WORKING EXPLANATION:

Creating an API to send E-mail using IFTTT:

Once the hardware is ready lets create an API (Application Program Interface) that can send E-mail to a particular E-mail ID. This can be easily done with the help of a website called IFTTT.com. I have also covered a project that could send SMS using ESP8266 and Email using PIC Microcontroller which uses the same IFTTT services.

If you have not yet used IFTTT visit the video at the end of this tutorial, if you are familiar simply follow the below steps

1. Log In into your IFTTT account
2. Search for “Maker Web hooks” and click on connect
3. Now search for “Gmail” and click on connect and follow the steps to give access
4. Then, create an Applet by clicking on My Applet-> New Applet.
5. Here, the “This” function will be for web makerhooks service and “that” function will be Gmail Services
6. So click on “This”, search and select Web maker hooks. It will ask for event name i have named my event as “123” you can name yours anything

7. Then click on "That", search and select Gmail and enter the subject and body of the mail.
8. Once all the required details are entered your Apple should be ready.
9. Now, search and get into Web Maker Hooks again and click on "Documentation". Then under event name enter the event name we used in the Applet. In my case it is "123" and copy the URL since we will be needing it in our Arduino program.

Once you are ready with the Hardware and the codes you can upload the program to your ESP8266 module by using a FTDI board. After uploading the program click on serial monitor and you should see something like IP address in that page. If not reset your ESP8266 module and try again.

Once you have verified all these you can now transfer your ESP8266 to the Perf board and power it ON and then short the switches.

After powering on you can use the IP to get into the above shown webpage and enable the Security system. After enabling wait for 50-60 seconds for the PIR sensor to calibrate.

Now you project is ready for action, you can leave it in a place you wish and if anyone crosses that places and falls within the range of PIR sensor a voice message will be triggered and an E-mail will be sent to your E-mail ID with the Date and time he/she crossed.

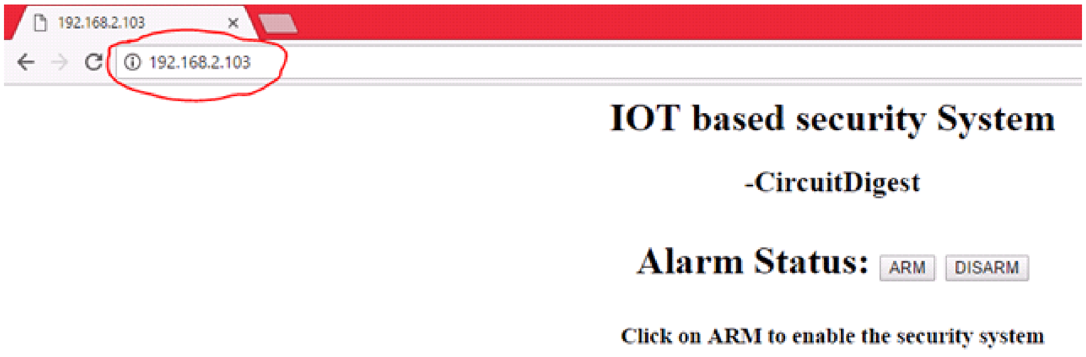
PROGRAM:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiClientSecure.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
MDNSResponder mdns;
boolean trigger_sms = false;
boolean power_module = false;
const char* ssid = "BPAS home";  //Enter you Wifi SSID here
const char* password = "cracksen"; //Enter your password here
ESP8266WebServer server(80);
String mainPage = ""; //The default page
String feedback = ""; //Gives staus of the switch
String currentPage = ""; //Combines main and feedback page
int GPIO_0 = 0; //Pin defanition
int GPIO_2 = 2; //Pin defanition
void setup(void){ //HTML code for webpage//
mainPage += "<h1 align=\"center\">IOT based security System</h1><h2 align=\"center\">-
CircuitDigest</h2><h1 align=\"center\"><p>Alarm Status: <a
href=\"switch2On\"><button>ARM</button></a>&nbsp;<a
href=\"switch2Off\"><button>DISARM</button></a></p>";
feedback = "<h3 align=\"center\">Click on ARM to enable the security system</h3>";
//End of HTML code//
currentPage = mainPage+feedback;
// preparing GPIOs
pinMode(GPIO_0, INPUT); //feeded by PIR sensor
pinMode(GPIO_2, OUTPUT); //used to turn on PIR and Voice module
digitalWrite(GPIO_2, LOW);
delay(1000);
Serial.begin(115200);
WiFi.begin(ssid, password);
Serial.println("");
// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
} // config static IP
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP()); //Serial monitor will give the IP addrss of your ESP module
if (mdns.begin("esp8266", WiFi.localIP())) {
Serial.println("MDNS responder started");
}
server.on("/", [](){
```

```
currentPage = mainPage+feedback;
server.send(200, "text/html", currentPage);
currentPage = "";
});
server.on("/switch2On", [](){
feedback = "<h3 align=\"center\">Alarm is up and running </h3>"; //HTML code modification
currentPage=mainPage+feedback;
server.send(200, "text/html", currentPage);
currentPage="";
digitalWrite(GPIO_2, HIGH); //Turn on PIR and Voice module
power_module=true;
delay(1000);
});
server.on("/switch2Off", [](){
feedback = "<h3 align=\"center\">Alarm is down</h3>"; //HTML code modification
currentPage=mainPage+feedback;
server.send(200, "text/html", currentPage);
currentPage="";
digitalWrite(GPIO_2, LOW); //Turn off PIT and Voice Module
power_module=false;
delay(1000);
});
server.begin();
Serial.println("IOT security system is up and running");
} //Function to send an E-mail//
void send_Email()
{
const char* host = "maker.ifttt.com"; //The host of API URL will be the same for all
const int httpsPort = 443;
WiFiClientSecure client;
Serial.print("connecting to ");
Serial.println(host);
if (!client.connect(host, httpsPort)) {
Serial.println("connection failed");
return;
}
String url = "/trigger/123/with/key/mDsoOV_EERS3xRfrh3_UQBhbcx0qlRHns-z2qXXXXX";
//Must change it to your API URL
Serial.print("requesting URL: ");
Serial.println(url);
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" +
"User-Agent: BuildFailureDetectorESP8266\r\n" +
"Connection: close\r\n\r\n");
Serial.println("request sent");
while (client.connected()) {
String line = client.readStringUntil('\n');
if (line == "\r") {
Serial.println("headers received");
break;
}  }
String line = client.readStringUntil('\n');
Serial.println("reply was:");
Serial.println("==========");
Serial.println(line);
Serial.println("==========");
Serial.println("closing connection");
} //Function to send an E-mail
void loop(void){
server.handleClient();
if (digitalRead(GPIO_0==HIGH)  && power_module==true)
trigger_sms=true;
```

```
else
trigger_sms=false;
if (trigger_sms==true) {
send_Email(); //If the PIR module is powered and if a person is detected send a E-mail
trigger_sms=false;
delay(2000);
}}
```

OUTPUT:

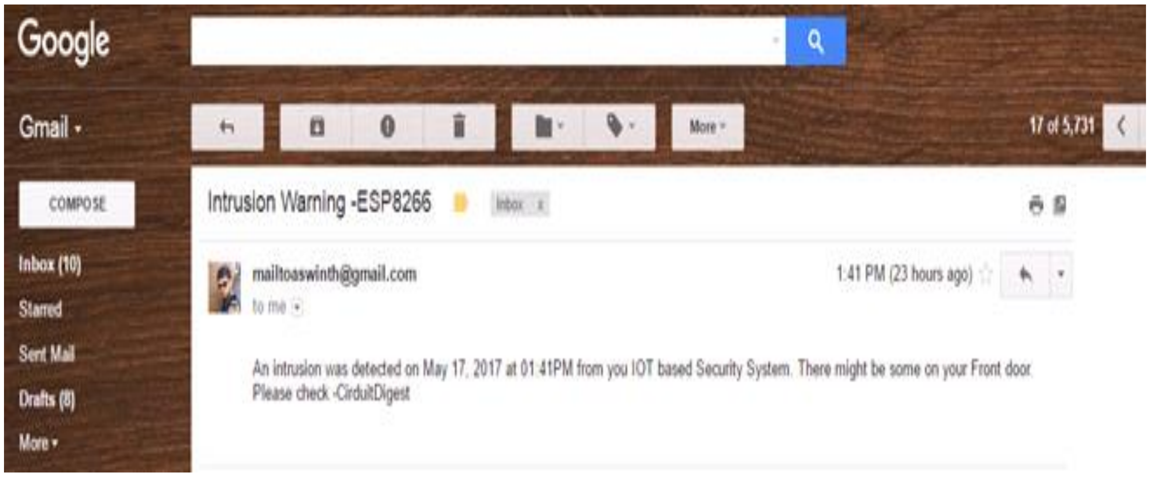


IOT based security System

-CircuitDigest

Alarm Status: ARM DISARM

Click on ARM to enable the security system



Intrusion Warning -ESP8266

mailtoaswinth@gmail.com

to me

1:41 PM (23 hours ago)

An intrusion was detected on May 17, 2017 at 01:41PM from you IOT based Security System. There might be some on your Front door. Please check -CircuitDigest

RESULT:

Thus, an IOT based Security system for Theft identification has been done, by sending E-mail and voice alert to the user when it detects someone with the help of ESP8266 module using Ardunio IDE.

HEALTH ABNORMAL ALERT SYSTEM

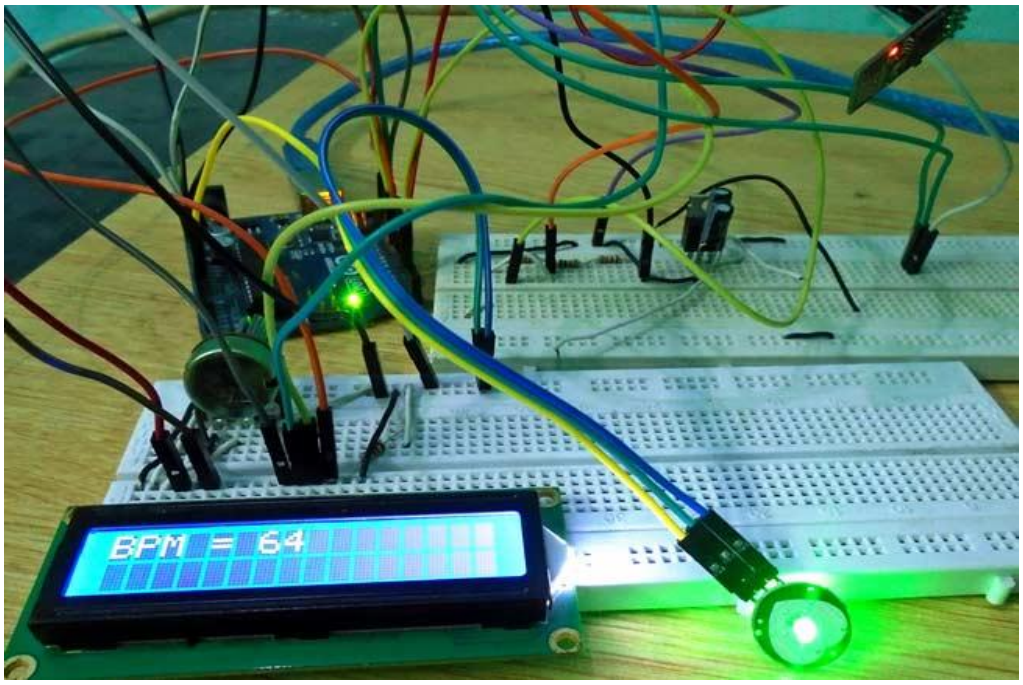
OBJECTIVE:

To build a Health monitoring system based on Internet of Things that uses Sensors to track patient health and uses internet to inform their loved ones in case of any issues.

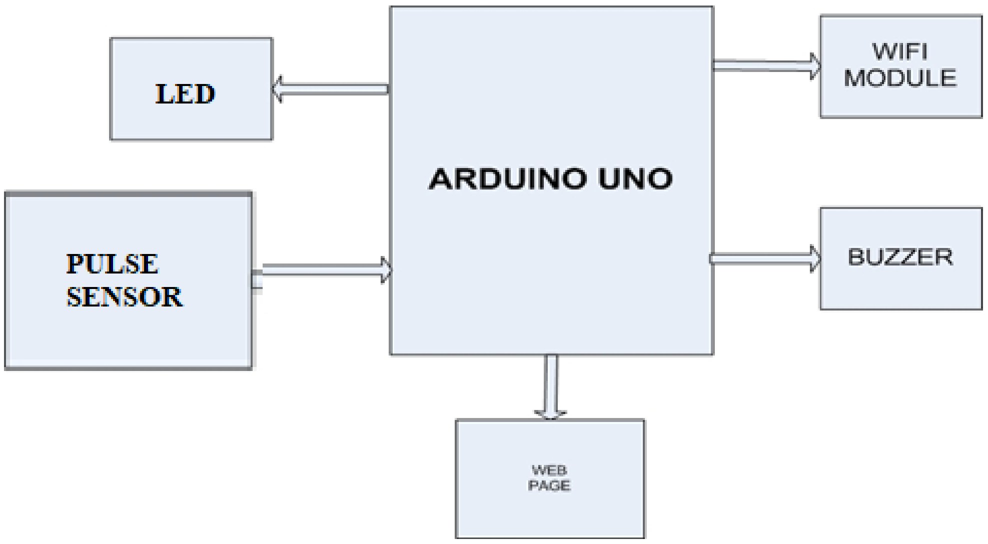
REQUIRED COMPONENTS:

- Pulse sensor
- Wi-Fi module ESP8266
- Arduino Uno
- LCD
- Bread Board
- 10k potentiometer
- 1k resistors
- 220-ohm resistors
- LED
- Connecting wires

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

Monitoring your beloved ones becomes a difficult task in the modern-day life. Keeping track of the health status of the patient at home is a difficult task. Especially old aged patients should be periodically monitored and their loved ones need to be informed about their health status from time to time while at work. So, an innovative system is proposed that automated this task with ease. Our system puts forward a smart patient health tracking system that uses Sensors to track patient health and uses internet to inform their loved ones in case of any issues. Our system uses temperature as well as heartbeat sensing to keep track of patient health. The mobile application for the patient and doctors contains a very simple GUI Interface for reading all the parameters in the mobile or at anywhere in the world by using internet connectivity. In this project we are using various sensors and modules for performing a different type of functions and the "Thingspeak" Cloud service is used for storing all the data in the cloud, it provides security and facility of accessing all the parameters at any time which is very useful for the doctors at the time of treatment. This system also generates an alert when it required that means at the time of any critical conditions and notifications about the medicines, location change, conditions etc.

PROGRAM:

```
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10);
#include <LiquidCrystal.h>
```

```
#include <stdlib.h>
LiquidCrystal lcd(12,11,5,4,3,2);
#define SSID "Your Wifi Name"     // "SSID-WiFiname"
#define PASS "Your Wifi Password"       // "password"
#define IP "184.106.153.149"// thingspeak.com ip
String msg = "GET /update?key=9YS21NU0HY5YS1IKU";  //change it with your api key like
"GET /update?key=Your Api Key"
//Variables
float temp;
int hum;
String tempC;
int error;
int pulsePin = 0;                  // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 13;                  // pin to blink led at each beat
int fadePin = 5;
int fadeRate = 0;
// Volatile Variables, used in the interrupt service routine!
volatile int BPM;                    // int that holds raw Analog in 0. updated every 2mS
volatile int Signal;                 // holds the incoming raw data
volatile int IBI = 600;              // int that holds the time interval between beats! Must be seeded!
volatile boolean Pulse = false;     // "True" when heartbeat is detected. "False" when not a "live
beat".
volatile boolean QS = false;        // becomes true when Arduino finds a beat.
// Regards Serial OutPut  -- Set This Up to your needs
static boolean serialVisual = true;   // Set to 'false' by Default.  Re-set to 'true' to see Arduino
Serial Monitor ASCII Visual Pulse
volatile int rate[10];                    // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find IBI
volatile int P =512;                      // used to find peak in pulse wave, seeded
volatile int T = 512;                     // used to find trough in pulse wave, seeded
volatile int thresh = 525;                // used to find instant moment of heart beat, seeded
volatile int amp = 100;                   // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable
BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with reasonable
BPM
void setup()
{
lcd.begin(16, 2);
lcd.print("circuitdigest.com");
delay(100);
lcd.setCursor(0,1);
lcd.print("Connecting...");
Serial.begin(9600); //or use default 115200.
esp8266.begin(9600);
Serial.println("AT");
esp8266.println("AT");
delay(5000);
if(esp8266.find("OK")){
connectWiFi();
}
interruptSetup();
}
void loop(){
lcd.clear();
start: //label
error=0;
lcd.setCursor(0, 0);
lcd.print("BPM = ");
lcd.print(BPM);
delay (100);
```

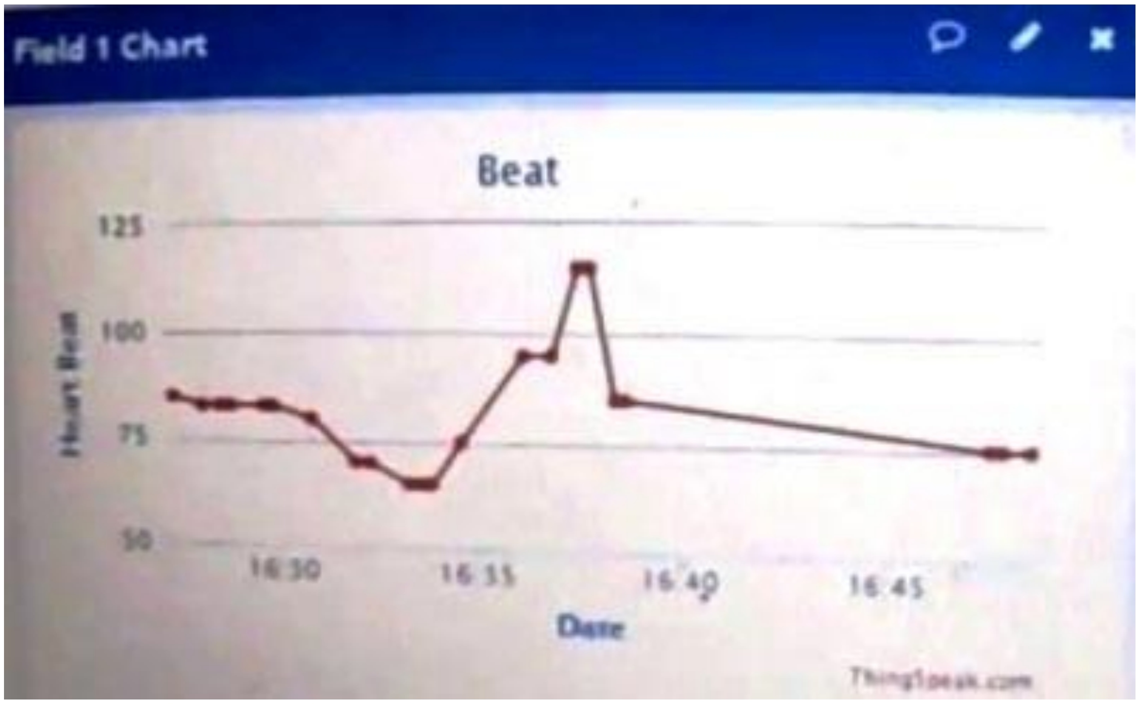
```
lcd.setCursor(0, 1); // set the cursor to column 0, line 2
delay(1000);
updatebeat();
//Resend if transmission is not completed
if (error==1){
goto start; //go to label "start"
}
delay(1000);
}
void updatebeat(){
String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += IP;
cmd += "\",80";
Serial.println(cmd);
esp8266.println(cmd);
delay(2000);
if(esp8266.find("Error")){
return;
}
cmd = msg ;
cmd += "&field1=";
cmd += BPM;
cmd += "\r\n";
Serial.print("AT+CIPSEND=");
esp8266.print("AT+CIPSEND=");
Serial.println(cmd.length());
esp8266.println(cmd.length());
if(esp8266.find(">")){
Serial.print(cmd);
esp8266.print(cmd);
}
```

```
else{
Serial.println("AT+CIPCLOSE");
esp8266.println("AT+CIPCLOSE");
//Resend...
error=1;
}}
boolean connectWiFi(){
Serial.println("AT+CWMODE=1");
esp8266.println("AT+CWMODE=1");
delay(2000);
String cmd="AT+CWJAP=\"";
cmd+=SSID;
cmd+="\",\"";
cmd+=PASS;
cmd+="\"";
Serial.println(cmd);
esp8266.println(cmd);
delay(5000);
if(esp8266.find("OK")){
Serial.println("OK");
return true;
}else{
return false;
}}
void interruptSetup(){
TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC
MODE
TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND
OCR2A
```

```
sei();            // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
ISR(TIMER2_COMPA_vect){                       // triggered when Timer2 counts to 124
cli();                                  // disable interrupts while we do this
Signal = analogRead(pulsePin);              // read the Pulse Sensor
sampleCounter += 2;                         // keep track of the time in mS
int N = sampleCounter - lastBeatTime;       // monitor the time since the last beat to avoid noise
//  find the peak and trough of the pulse wave
if(Signal < thresh && N > (IBI/5)*3){       // avoid dichrotic noise by waiting 3/5 of last IBI
if (Signal < T){                        // T is the trough
T = Signal;                             // keep track of lowest point in pulse wave
}  }
if(Signal > thresh && Signal > P){          // thresh condition helps avoid noise
P = Signal;                             // P is the peak
}                                       // keep track of highest point in pulse wave
//  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
// signal surges up in value every time there is a pulse
if (N > 250){                                   // avoid high frequency noise
if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
Pulse = true;                               // set the Pulse flag when there is a pulse
digitalWrite(blinkPin,HIGH);                // turn on pin 13 LED
IBI = sampleCounter - lastBeatTime;         // time between beats in mS
lastBeatTime = sampleCounter;               // keep track of time for next pulse
if(secondBeat){                        // if this is the second beat
secondBeat = false;                  // clear secondBeat flag
for(int i=0; i<=9; i++){             // seed the running total to get a realistic BPM at startup
rate[i] = IBI;
}      }
if(firstBeat){                       // if it's the first time beat is found
firstBeat = false;                   // clear firstBeat flag
secondBeat = true;                   // set the second beat flag
sei();                          // enable interrupts again
return;                         // IBI value is unreliable so discard it
}
word runningTotal = 0;                  // clear the runningTotal variable
for(int i=0; i<=8; i++){                // shift data in the rate array
rate[i] = rate[i+1];                // and drop the oldest IBI value
runningTotal += rate[i];              // add up the 9 oldest IBI values
}
rate[9] = IBI;                      // add the latest IBI to the rate array
runningTotal += rate[9];                // add the latest IBI to runningTotal
runningTotal /= 10;                   // average the last 10 IBI values
BPM = 60000/runningTotal;               // how many beats can fit into a minute? that's BPM!
QS = true;                          // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
} }
if (Signal < thresh && Pulse == true){   // when the values are going down, the beat is over
digitalWrite(blinkPin,LOW);            // turn off pin 13 LED
Pulse = false;                      // reset the Pulse flag so we can do it again
amp = P - T;                        // get amplitude of the pulse wave
thresh = amp/2 + T;                   // set thresh at 50% of the amplitude
P = thresh;                         // reset these for next time
T = thresh;
}
if (N > 2500){                        // if 2.5 seconds go by without a beat
thresh = 512;                       // set thresh default
P = 512;                            // set P default
T = 512;                            // set T default
lastBeatTime = sampleCounter;          // bring the lastBeatTime up to date
firstBeat = true;                     // set these to avoid noise
secondBeat = false;                   // when we get the heartbeat back
}
```

```
sei();
// enable interrupts when youre done!
}// end isr
```

OUTPUT:



RESULT:

Thus, an IoT based patient health abnormal alert system effectively uses internet to monitor patient health stats and save lives on time.

ALCOHOL DETECTOR

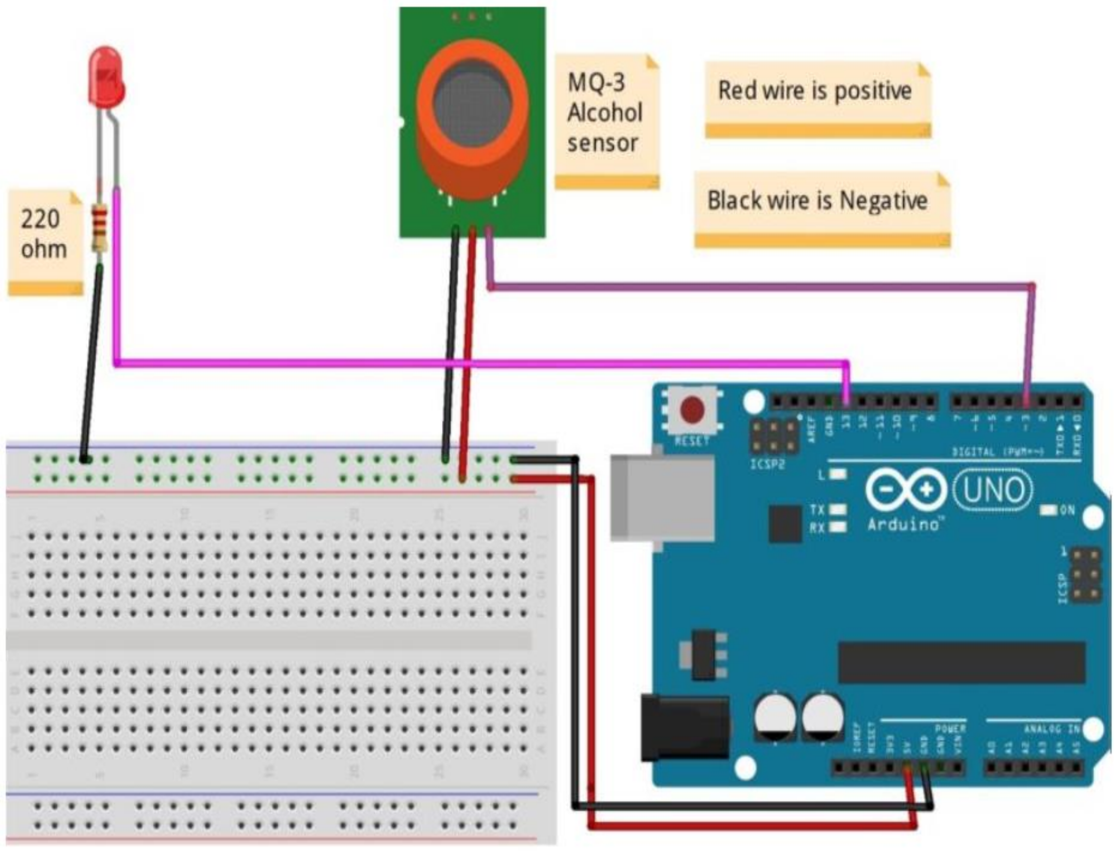
OBJECTIVE:

To develop an alcohol detector in which we will monitor the drunk and drive using internet and will trigger an alarm when the air quality goes down beyond a certain level, means when there is sufficient amount of alcohol present in the air.

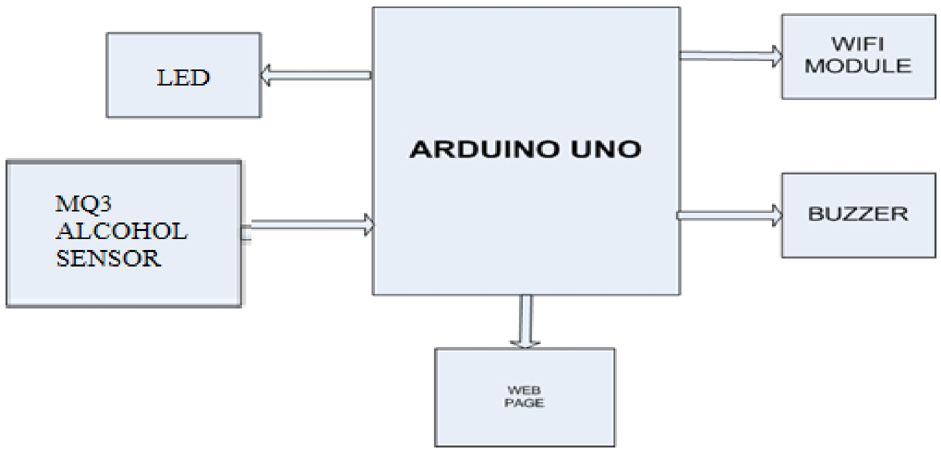
REQUIRED COMPONENTS:

- MQ3 Alcohol sensor
- Arduino Uno
- LED or Buzzer
- Wi-Fi Module
- Breadboard
- Connecting wires

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



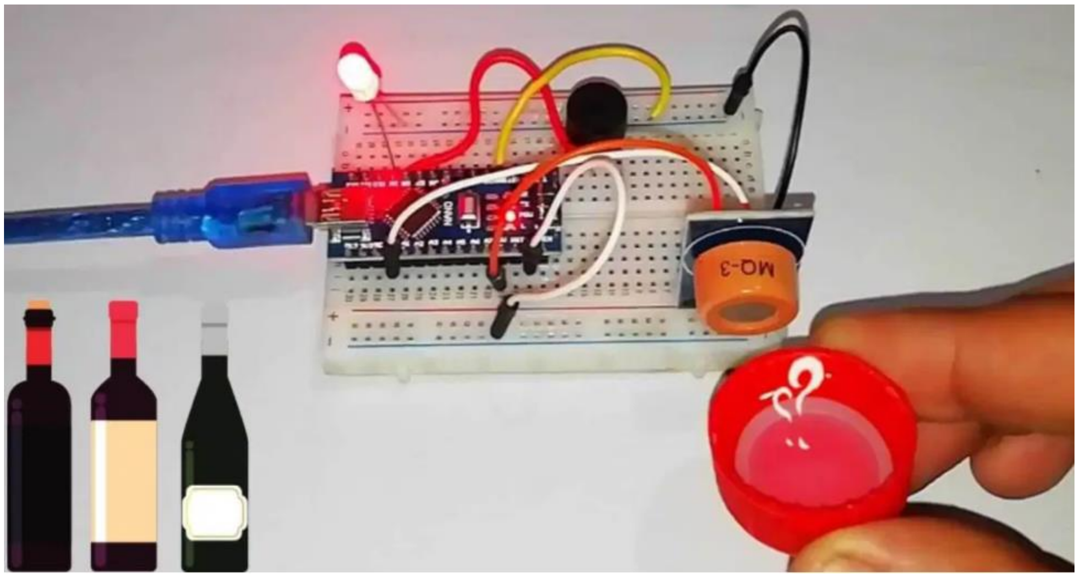
WORKING EXPLANATION:

- The MQ3 sensor can sense alcohol, so it is perfect alcohol sensor for our alcohol detector Project.
- When we will connect it to Arduino then it will sense the alcohol, and we will get the alcohol level in PPM (parts per million).
- MQ3 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So, for converting the output in PPM, here we have used a library for MQ3 sensor.
- Sensor will detect whether alcohol is there in the air and it will alert us with the “Alcohol Detected” message. Otherwise, it will send a “No Alcohol Detected” message.
- Also, if alcohol is detected LED will be turned on and Buzzer sounds.

PROGRAM:

```
#include "MQ3.h"
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10); // This makes pin 9 of Arduino as RX pin and pin 10 of Arduino
as the TX pin
const int sensorPin= 0;
int air_quality;
//Code for LED in the IDE of arduino from this line
int LED = 12;
int ALCOHOL_sensor = 3;// MQ-6 SENSOR
int ALCOHOL_detected;
void setup() {
Serial.begin(9600);
pinMode(LED, OUTPUT);
pinMode(ALCOHOL_sensor, INPUT);
}
void loop()  {
ALCOHOL_detected = digitalRead(ALCOHOL_sensor);
Serial.println(ALCOHOL_detected);
if (ALCOHOL_detected == 1)
{
Serial.println("ALCOHOL detected...");
digitalWrite(LED, HIGH);
}
else
{
Serial.println("No ALCOHOL  detected ");
digitalWrite(LED, LOW);
}  }
// Code for Buzzerin the IDE of arduino from this line
int LED = 12;
int BUZZER = 13;
int ALCOHOL_sensor = 3;// MQ-3 SENSOR
int ALCOHOL_detected;
void setup()
{
Serial.begin(9600);
pinMode(LED, OUTPUT);
pinMode(BUZZER, OUTPUT);
pinMode(ALCOHOL_sensor, INPUT);
}
void loop()
{
ALCOHOL_detected = digitalRead(ALCOHOL_sensor);
Serial.println(ALCOHOL_detected);
if (ALCOHOL_detected == 1)
{
Serial.println("ALCOHOL detected...");
digitalWrite(LED, HIGH);
digitalWrite(BUZZER, HIGH);
}
else
{
Serial.println("No ALCOHOL  detected.");
digitalWrite(LED, LOW);
digitalWrite(BUZZER, LOW);
}  }
```

OUTPUT:



RESULT:

Thus, an IOT Based alcohol detector has been designed to monitor the drunk and drive with the help of Arduino uno kit.

WATER TANK LEVEL MONITORING SYSTEM

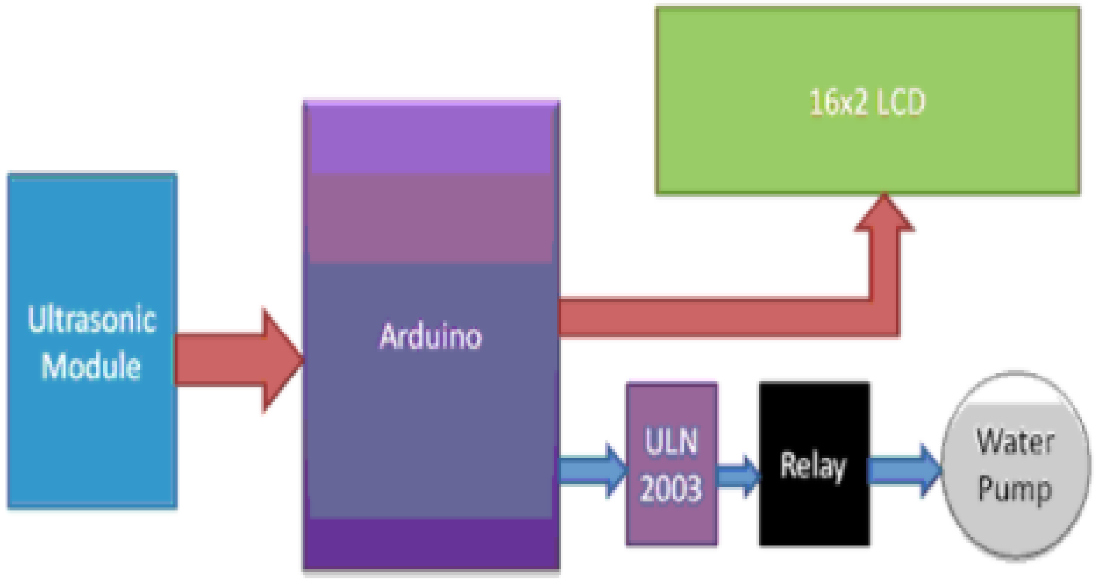
OBJECTIVE:

To develop an Arduino based automatic water level monitoring system and we are going to measure the water level by using ultrasonic sensors. Once the water level is measured, if it reaches the particular level automatically alarm will turn on.

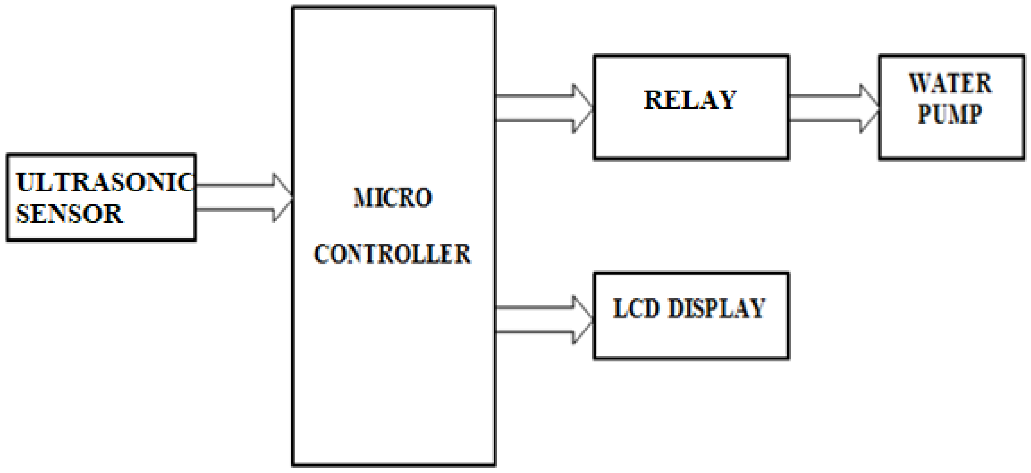
REQUIRED COMPONENTS:

- Arduino Uno
- Ultrasonic sensor Module
- 16x2 LCD
- Breadboard
- Buzzer
- Copper wire
- Connecting wires
- Water Pump
- Relay

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

Working of this project is very simple we have used Ultrasonic sensor module which sends the sound waves in the water tank and detects reflection of sound waves that is ECHO. First of all we needs to trigger the ultrasonic sensor module to transmit signal by using Arduino and then wait to receive ECHO. Arduino reads the time between triggering and received ECHO. We know that speed of sound is around 340 m/s. so we can calculate distance by using given formula:

Distance= (travel time/2) * speed of sound

Where speed of sound is approximately 340m per second.

By using this methods we gets distance from sensor to water surface. After it we need to calculate water level.

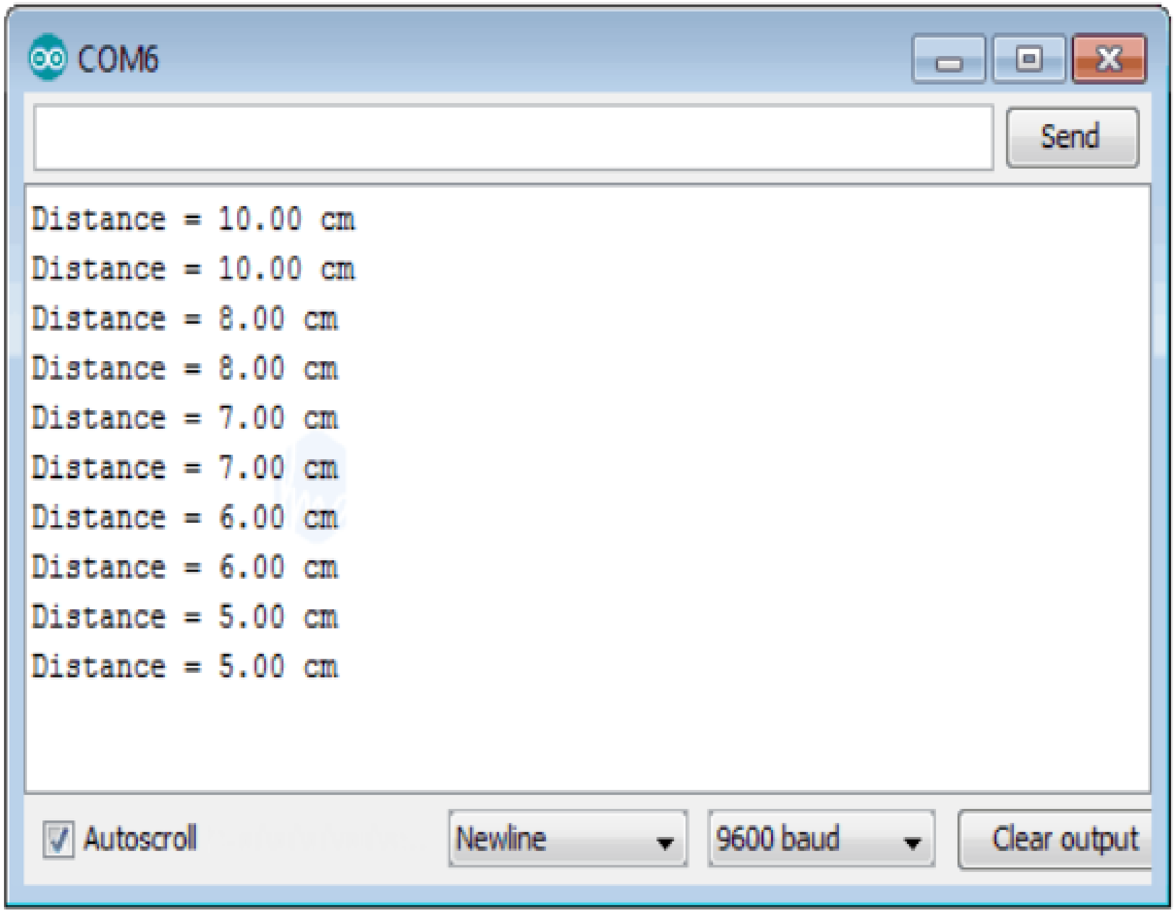
Now we need to calculate the total length of water tank. As we know the length of water tank then we can calculate the water level by subtracting resulting distance coming from ultrasonic from total length of tank. And we will get the water level distance. Now we can convert this water level in to the percent of water, and can display it on LCD. The working of the complete water level indicator project is shown in below block diagram.

PROGRAM:

```
#include <LiquidCrystal.h>

#define trigger 10

#define echo 11

#define motor 8
```

```
#define buzzer 12
LiquidCrystal lcd(7,6,5,4,3,2);
float time=0,distance=0;
int temp=0;
void setup()
{
lcd.begin(16,2);
pinMode(trigger,OUTPUT);
pinMode(echo,INPUT);
pinMode(motor, OUTPUT);
pinMode(buzzer, OUTPUT);
lcd.print("  Water Level ");
lcd.setCursor(0,1);
lcd.print("   Indicator  ");
delay(2000);
}
void loop()
{
lcd.clear();
digitalWrite(trigger,LOW);
delayMicroseconds(2);
digitalWrite(trigger,HIGH);
delayMicroseconds(10);
digitalWrite(trigger,LOW);
delayMicroseconds(2);
time=pulseIn(echo,HIGH);
distance=time*340/20000;
lcd.clear();
lcd.print("Water Space In  ");
lcd.setCursor(0,1);
lcd.print("Tank is: ");
lcd.print(distance);
lcd.print("Cm");
delay(2000);
if(distance<12 && temp==0)
{
digitalWrite(motor, LOW);
digitalWrite(buzzer, HIGH);
lcd.clear();
lcd.print("Water Tank Full ");
lcd.setCursor(0,1);
lcd.print("Motor Turned OFF");
delay(2000);
digitalWrite(buzzer, LOW);
delay(3000);
temp=1;
}
else if(distance<12 && temp==1)
{
digitalWrite(motor, LOW);
lcd.clear();
lcd.print("Water Tank Full ");
lcd.setCursor(0,1);
lcd.print("Motor Turned OFF");
delay(5000);
}
else if(distance>30)
{
digitalWrite(motor, HIGH);
lcd.clear();
lcd.print("LOW Water Level");
lcd.setCursor(0,1);
lcd.print("Motor Turned ON");
delay(5000);
temp=0;
}
}
```

OUTPUT:



```
Distance = 10.00 cm
Distance = 10.00 cm
Distance = 8.00 cm
Distance = 8.00 cm
Distance = 7.00 cm
Distance = 7.00 cm
Distance = 6.00 cm
Distance = 6.00 cm
Distance = 5.00 cm
Distance = 5.00 cm
```

RESULT:

Thus, an Automatic water tank level monitoring system using ultrasonic sensor successfully implemented using arduino Uno.

RAIN FALL LEVEL DETECTION SYSTEM

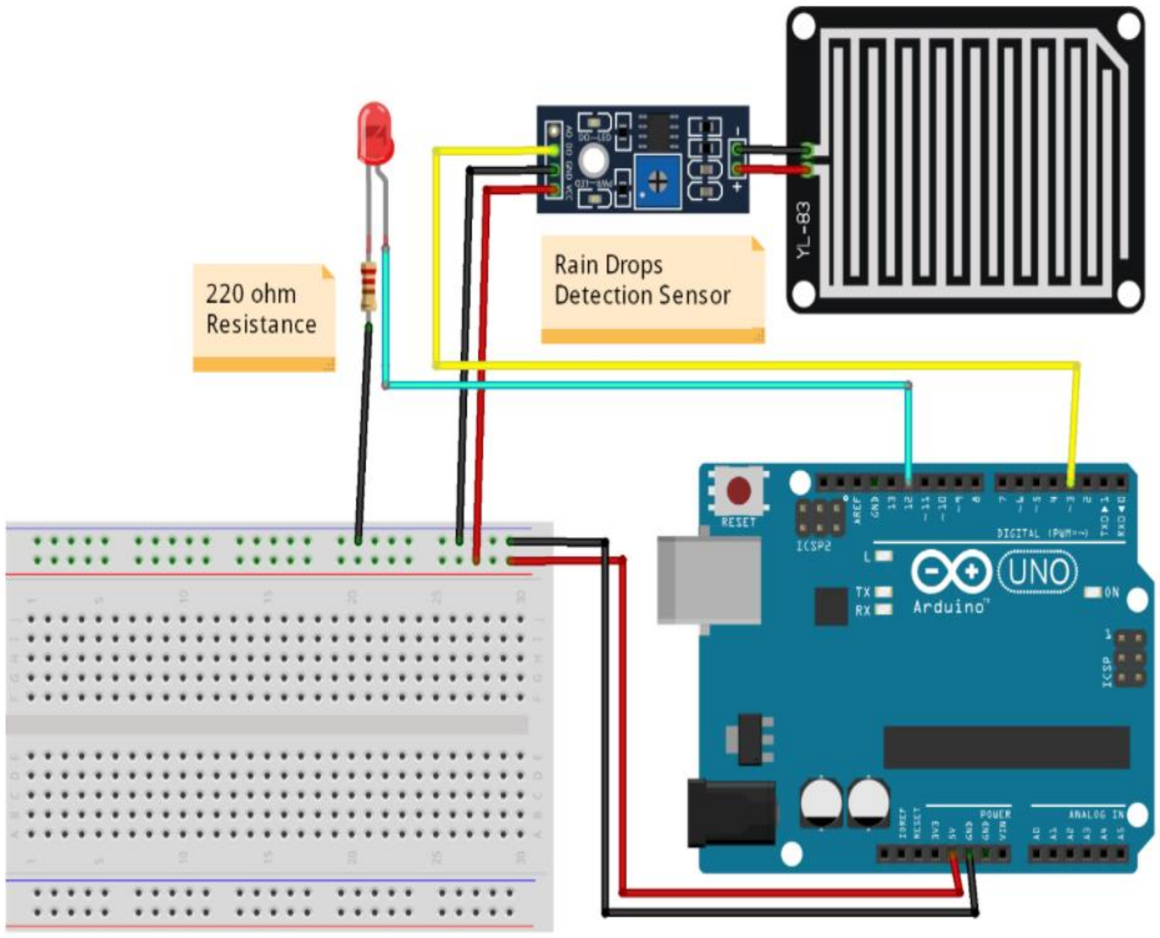
OBJECTIVE:

To build a Rain Fall Level Detection System which will monitor the rainfall using Arduino and will trigger an alarm when it detects rainfall.

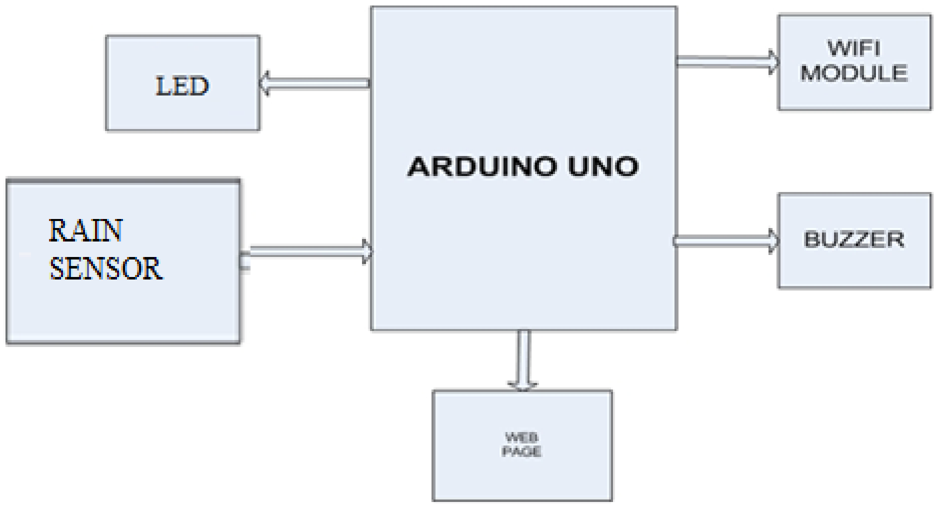
REQUIRED COMPONENTS:

- Arduino UNO
- Rain sensor
- Buzzer or LED
- Breadboard
- Connecting wires

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

Working of the **rain sensor module** is simple to understand. During a sunny day, due to the dryness on the rain board module, it offers high resistance to the supply voltage. This voltage appears on the output pin of the rain board module as 5V. This 5V is read as 1023 if read by an analog pin of the Arduino. During rain, the rainwater causes an increase in the wetness on the rain board, which in turn results in the decrease in the resistance offered for the supply. As the resistance decreases gradually, the output voltage starts to decrease.

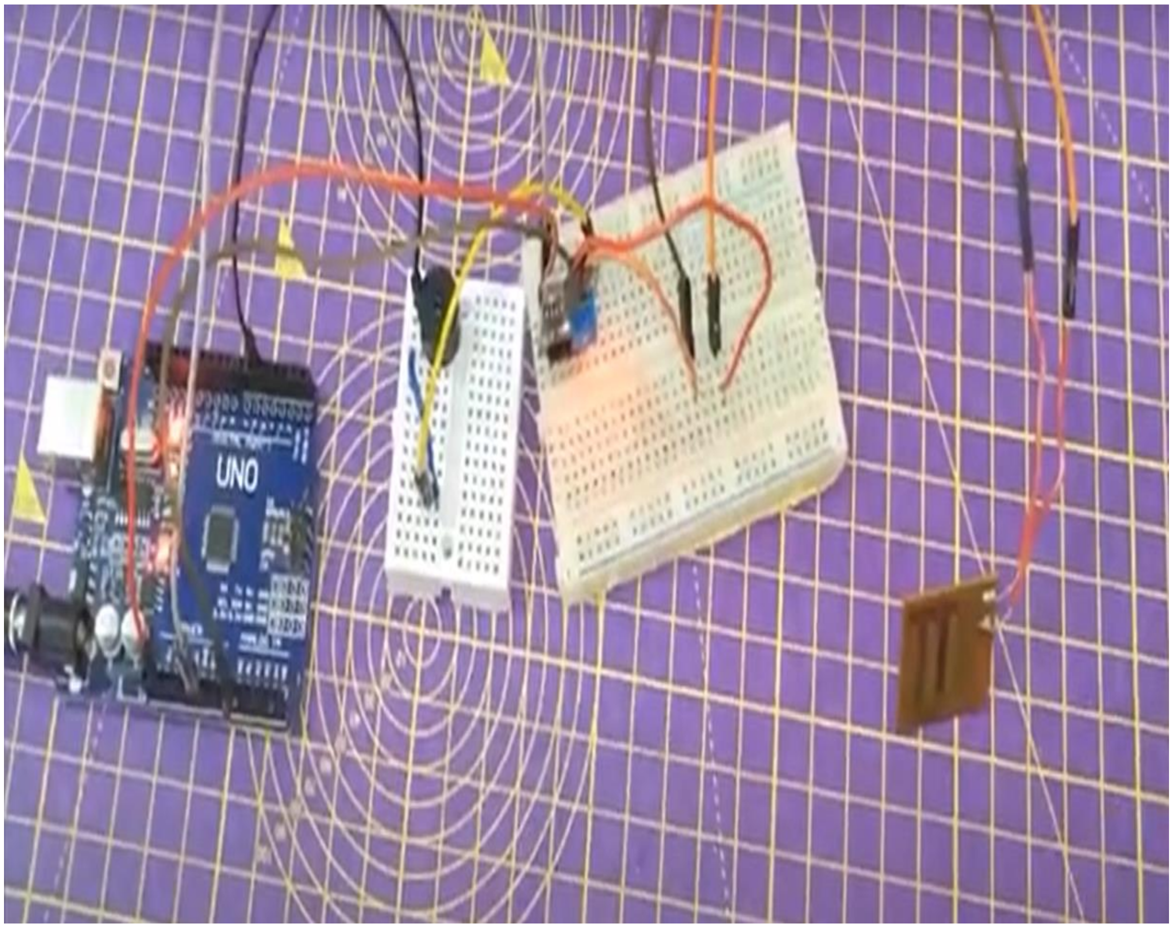
When the rain board is fully wet, and the resistance offered by it is minimum, the output voltage will be as low as possible (approx. 0). This 0V is read as 0 value if read by an analog pin of the Arduino. If the rain board module is partially wet, the output of this rain board module will be with respect to the resistance it offers. If the resistance offered by the rain board module is in such a way that the output is 3V the read analog value will be 613. Formula to find ADC can be given by, **ADC = (analog voltage value X 1023)/5**. By using this formula you can convert any analog voltage to t Arduino analog read value.

The rain gauge module, which is shown in the circuit diagram is connected to the control board. The control board’s VCC pin is connected to the 5V supply. The ground pin is connected to ground. If needed, the D0 pin is connected to any digital pin of the Arduino, and that pin must be declared as an output pin in the program. The problem we face with the D0 pin is that we can’t get the exact value of the output voltage. If the output crosses the threshold voltage, then the control module can sense the change in the output. We need to operate the buzzer, even if there is a considerable change in the output voltage in the rain board module. Due to these reasons, the A0 pin is connected to the analog pin of Arduino, which makes monitoring the change in output easy. The buzzer, which is used as a signal to the user, can be connected to any digital pin of the Arduino. If the buzzer needs more than 5V, then try to connect a relay circuit or a transistor and then connecting the load to it.

PROGRAM:

```
#define rainfall A0
#define buzzer 5
int value;
int set=10;
void setup()
{
Serial.begin(9600);
pinMode(buzzer,OUTPUT);
pinMode(rainfall,INPUT);
}
void loop()
{
value = analogRead(rainfall);
Serial.println("LOL");
Serial.println(value);
value = map(value,0,1023,225,0);
Serial.println(value);
if(value>=set)
{
Serial.println("rain detected");
digitalWrite(buzzer,HIGH);
}
else
{
digitalWrite(buzzer,LOW);
}
delay(200);
}
```

OUTPUT:



RESULT:

Thus, a Rain Fall Level Detection System is implemented which will monitor the rainfall using Arduino and will trigger an alarm when it detects rainfall.

GAS LEAKAGE MONITORING SYSTEM

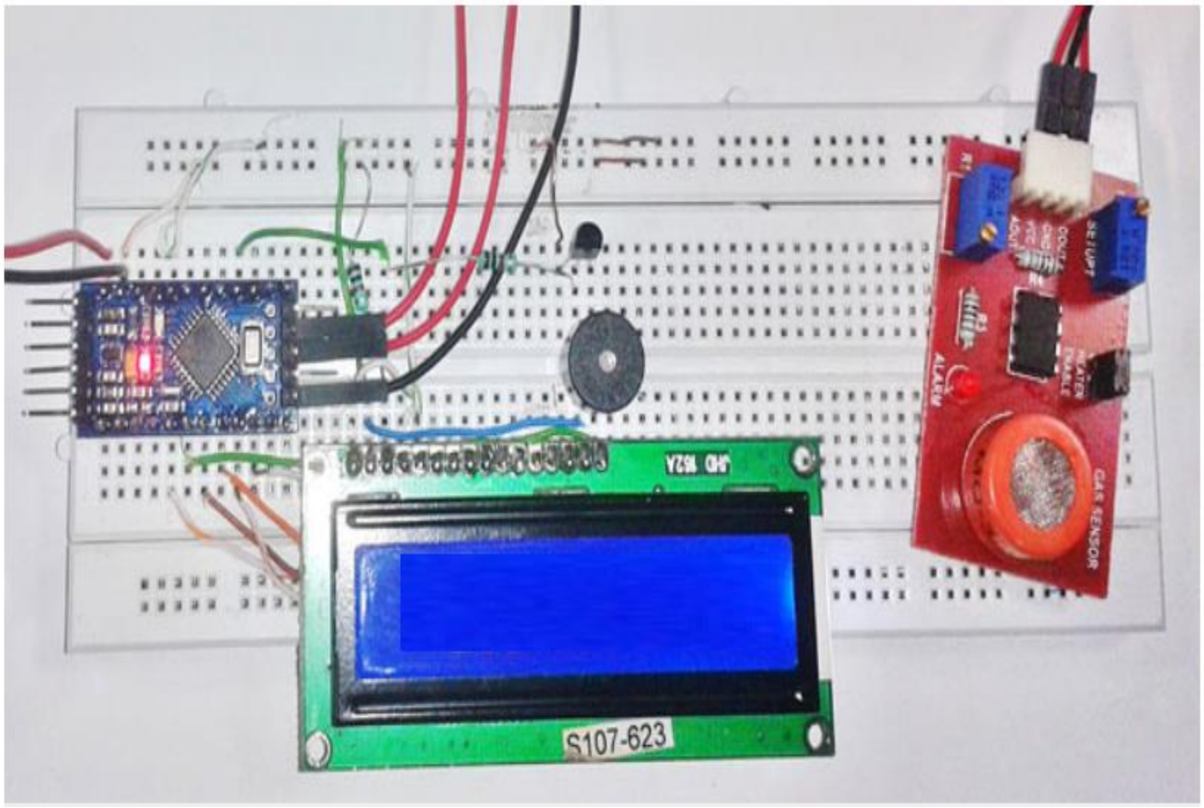
OBJECTIVE:

To develop a Gas Leakage Monitoring System which will monitor the leakage of LPG gas using internet and will trigger an alarm when there is leakage of LPG gas.

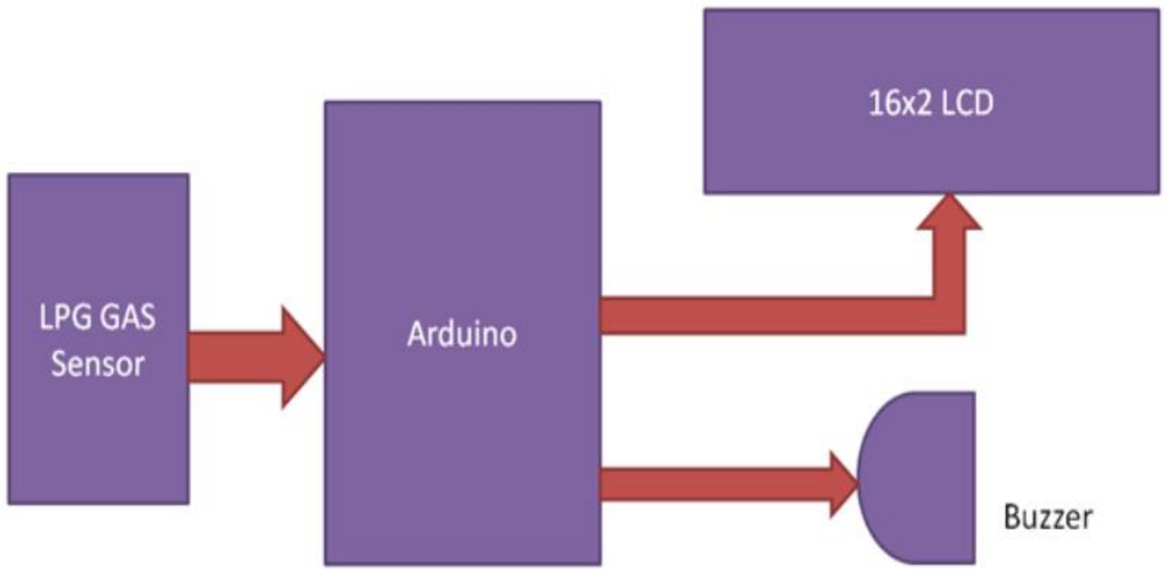
REQUIRED COMPONENTS:

- Arduino Pro Mini/Arduino UNO
- LPG Gas sensor Module/MQ6 Gas Sensor
- Buzzer
- BC 547 Transistor
- 16x2 LCD
- 1K resistor
- Bread board
- 9-volt battery
- Connecting wires

CIRCUIT DIAGRAM:



BLOCK DIAGRAM:



WORKING EXPLANATION:

While LPG is an essential need of every household, its leakage could lead to a disaster. To alert on LPG leakage and prevent any mishappening there are various products to detect the leakage. Here we have developed an Arduino based LPG gas detector alarm. If gas leakage occurs, this system detects it and makes an alert by buzing the buzzer attached with the circuit. This system is easy to build and anyone who have some knowledge of electronics and programing, can build it.. We have used a LPG gas sensor module to detect LPG Gas. When LPG gas leakage occurs, it gives a HIGH pulse on its DO pin and arduino continuously reads its DO pin. When Arduino gets a HIGH pulse from LPG Gas module it shows “LPG Gas Leakage Alert” message on 16x2 LCD and activates buzzer which beeps again and again until the gas detector module doesn't sense the gas in environment. When LPG gas detector module gives LOW pulse to arduino, then LCD shows “No LPG Gas Leakage” message.

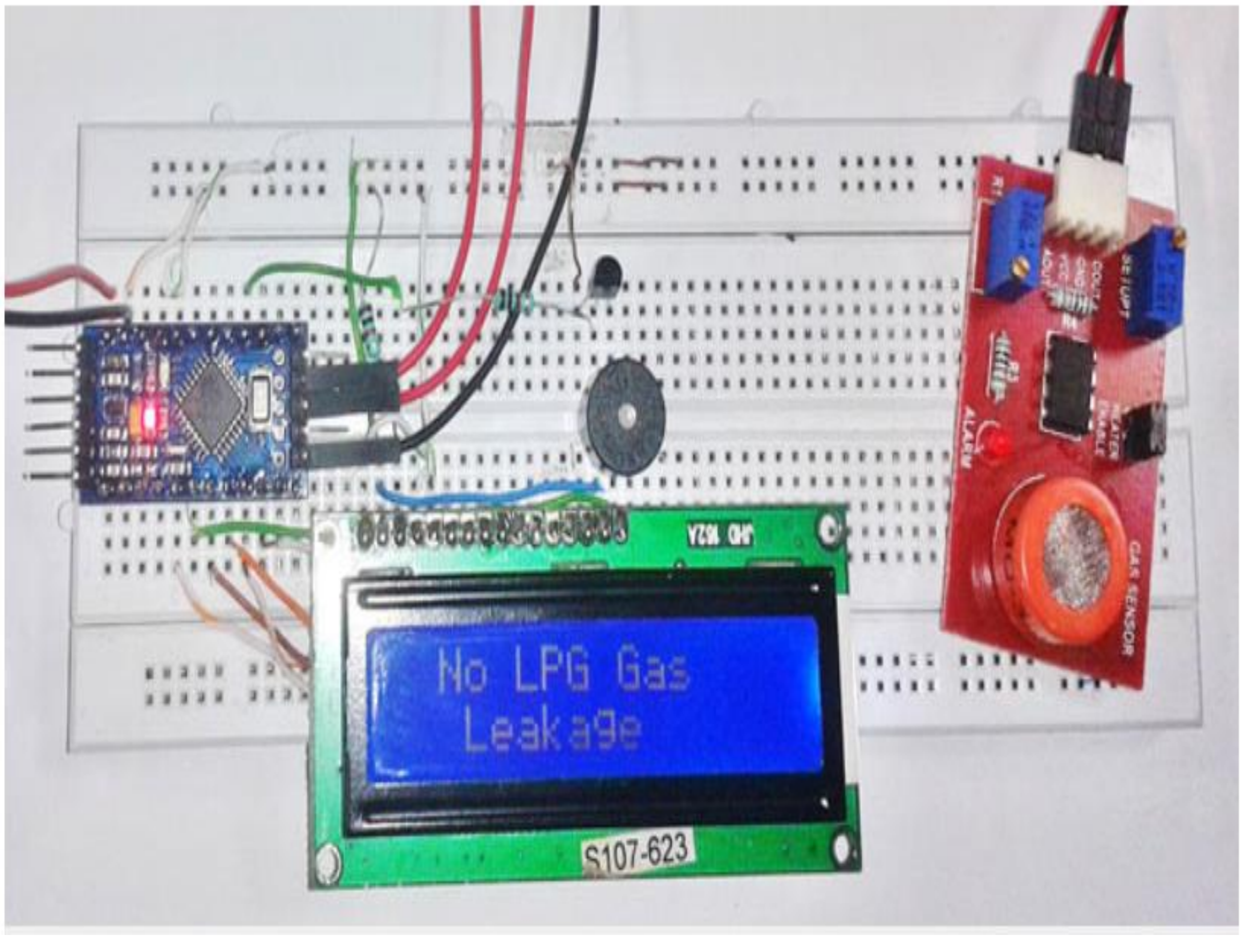
LPG Gas Sensor Module

This module contains a MQ3 sensor which actually detects LPG gas, a comparator (LM393) for comparing MQ3 output voltage with reference voltage. It gives a HIGH output when LPG gas is sensed. A potentiometer is also used for controlling sensitivity of gas sensing. This module is very easy to interface with microcontrollers and arduino and easily available in market by name “LPG Gas Sensor Module”. We can also build it by using LM358 or LM393 and MQ3.

PROGRAM:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(3, 2, 4, 5, 6, 7);
#define lpg_sensor 18
#define buzzer 13
void setup()
{
pinMode(lpg_sensor, INPUT);
pinMode(buzzer, OUTPUT);
lcd.begin(16, 2);
lcd.print("LPG Gas Detector");
lcd.setCursor(0,1);
lcd.print("Circuit Digest");
delay(2000);
}
void loop()
{
if(digitalRead(lpg_sensor))
{
digitalWrite(buzzer, HIGH);
lcd.clear();
lcd.print("LPG Gas Leakage");
lcd.setCursor(0, 1);
lcd.print("     Alert     ");
delay(400);
digitalWrite(buzzer, LOW);
delay(500);
}
else
{
digitalWrite(buzzer, LOW);
lcd.clear();
lcd.print("  No LPG Gas ");
lcd.setCursor(0,1);
lcd.print("   Leakage   ");
delay(1000);
}
}
```

OUTPUT:



RESULT:

Thus, a Gas Leakage Monitoring System which will monitor the leakage of LPG gas is implemented using Arduino and an alarm is triggered when there is leakage of LPG gas.