LPG GAS MONITORING SYSTEM

A mini project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

By

D.MEGHANA	**(19K41A04A6)**
G. THARUN	**(19K41A0470)**
K. SHIVASAI	**(19K41A04A3)**
B. ABHISHEK	**(18K41A0463)**

Under the guidance of

Mr. Y. Srikanth

Assistant Professor, Department of ECE.

SR Engineering College

Innovation . Creativity . Entrepreneurship

S R ENGINEERING COLLEGE

Ananthasagar, Warangal-506371.

CERTIFICATE

This is to certify that this project entitled **"GAS MONITORING SYSTEM**" is the bonafied work carried out by **D. MEGHANA, G. THARUN, K. SHIVASAI and B. ABHISHEK** as a mini project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **ELECTRONICS & COMMUNICATIONENGINEERING** during the academic year 2021-2022 under our guidance and Supervision.

Mr. Y. Srikanth
Assistant professor (ECE)
S R Engineering College,
Ananthasagar, Warangal.

Dr. Sandip Bhattacharya
Assoc. Prof. & HOD (ECE),
S R Engineering College,
Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Mr. Y. Srikanth, Assistant Professor** as well as project coordinators **Dr. K. Raj Kumar, Associate Professor, Dr. P. Anuradha, Sr. Assistant Professor** for guiding us from the beginning through the end of the Capstone project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive, encouraging and ultimately drove us to the right direction.

We express our thanks to Head of the ECE Department **Dr. Sandip Bhattacharya Associate Professor,** for his/her encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Principal, **Dr. V. Mahesh,** for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

These days the usage of LPG is widely used in many fields especially in house hold purposes. As the usage of LPG increases, the accidents due to explosion of LPG is increasing, which became a threaten to human life. As well as there's another problem faced while using LPG gas, we will be unaware of the amount of gas present in the cylinder. In this system the gas sensor detects the leakage of the LPG and alerts the user by giving a buzzer, the cylinder is automatically turned off. Later the gas can be turned on or off again with the help of Bluetooth module from different place. In addition to this the system continuously monitors the level of the LPG in the cylinder using load sensor and if the level is below the threshold limit the system intimates the user with the help of OLED display. So that the user has an idea about the max time the LPG lasts.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Gas leakage is a serious problem and nowadays it is observed in many places like residences, industries, and vehicles like Compressed Natural Gas (CNG), buses, cars, etc. It is noticed that due to gas leakage, dangerous accidents occur. The Liquefied petroleum gas (LPG), or propane, is a flammable mixture of hydrocarbon gases used as fuel in many applications like homes, hostels, industries, automobiles, and vehicles because of its desirable properties which include high calorific value, less smoke, less soot, and meager harm to the environment. Liquid petroleum gas (LPG) is highly inflammable and can burn even at some distance from the source of leakage. This energy source is primarily composed of propane and butane which are highly flammable chemical compounds. These gases can catch fire easily. In homes, LPG is used mainly for cooking purposes. When a leak occurs, the leaked gases may lead to an explosion. Gas leakage leads to various accidents resulting in both material loss and human injuries. Home fires have been occurring frequently and the threat to human lives and properties has been growing in recent years. The risks of explosion, fire, suffocation are based on their physical properties such toxicity, flammability, etc. The number of deaths due to the explosion of gas cylinders has been increasing in recent years. The Bhopal gas tragedy is an example of accidents due to gas leakage. The reason for such explosions is due to substandard cylinders, old valves, no regular checking of gas cylinders, worn out regulators and a lack of awareness of handling gas cylinders. Therefore, the gas leakage should be detected and controlled to protect people from danger. An odorant such as ethane thiol is added to LPG, so that leaks can be detected easily by most people. However, some people who have a reduced sense of smell may not be able to rely upon this inherent safety mechanism. A gas monitoring system becomes vital and helps to protect people from the dangers

When you take normal case such as a gas cylinder, the user is more negligent about it, although the user is cautious about the usage of gas in the kitchen, because the user frequently needs to on the stove whenever relatives or friends came to the home at unpredicted times. The user who is in the kitchen loses patience and will not care about the usage of the stove and requires using the stove frequently. At many instances, we are observing a few cases about someone who is injured or died because of exposure to gas in the kitchen. To predict the disaster in advance, then alternatives could be taken to avoid such cases. an IoT device is designed that

will detect gas leakages intimate the user with t and automatically turns off the gas regulator and the status of the gas to be known whenever the gas cylinder is empty and intimate the user to book their gas in advance in order to satisfy the need of the user.

1.2 EXISTING SYSTEMS

In the existing method, different gas sensing technology is used. The LPG gas leakage is detected by the semiconductor sensor. Nowadays LPG accidents occur very common. The main reason of these accidents is due to the leakage of LPG. This leakage of LPG starts when we forget to close the main regulator valve. This is the basis of these kinds of accidents. Already there are some sorts of remedial measures such as when the leakage is detected, alert message International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.org NCACS-2015 Conference Proceedings Volume 3, Issue 15 Special Issue - 2015 1 is sent to the fire station and the owner. The other remedial measure is that when the leakage is detected, exhaust fan is switched on. The first mentioned method has the disadvantage that there is no control action taken, it needs a manual controlling which puts human into direct risk. The second method has the disadvantage that if the wiring of the exhaust fan is not proper then it will cause immediate explosion due to the flow of AC. In all these mentioned methods above, there is only detection no control action is taken.

DRAWBACKS OF EXISTING SYSTEM

- The system doesn't turn off the regulator when there is leakage it just intimates the user
- The system doesn't indicate the gas level
- The system doesn't have Bluetooth module to control the gas regulator

1.3 PRESENT WORK

In order to overcome the problems faced by the user due to lpg gas cylinder we proposed a "Gas monitoring system" where it detects the leakage of the gas in addition to this it intimates the user to book their gas based on the gas level and later whenever the user is willing turn on/off the cylinder it can be accessed within the house. Whenever there is gas leakage in the house the sensor detects the gas leakage and intimates the user with the help of buzzer and automatically turn offs the regulator with the help of the servomotor and displays the message on oled. If the user is willing to turn on or off the gas cylinder. The user can access it with the help of the Bluetooth control app within the house. Many of the users face difficulty in knowing

the level of the gas and face problem in measuring it. our system measures the weight of the cylinder with the help of load cell sensor and intimates the user by displaying a message on the oled to book their gas cylinder in advance in order to not to face any problem

1.4 LITERATURE SURVEY

A.Mahalingam et.al. proposed a gas leak detector that meets the UK occupational and health standards. Gas leakage is a major concern with residential, commercial premises and gas powered transportation vehicles. One of the preventive measures to avoid the danger associated with gas leakage is to install a gas leakage detector at vulnerable locations. The objective of this work is to present the design of a cost effective automatic alarming system, which can detect liquefied petroleum gas leakage in various premises.

Sunithaa.J et al. designed a wireless LPG leakage monitoring system for home safety. The proposed system detects the leakage of the LPG and alerts the consumer using GSM about the leakage and it will switch on the exhaust fan. This system also has a feature that the consumption is approximately indicated in terms of the total weight. Whenever the system detects the increase in the concentration of the LPG leakage it immediately alerts by activating an alarm and simultaneously sending message to the particular mobile phones. The fan is switched on to exhaust gas and an LPG safe valve fitted to the cylinder is closed through signals to avoid further leakage. The device assures safety and prevents explosion

Jolhe et al. have designed a microcontroller-based system where a gas sensor (MQ6) is used in detection of LPG leakage. This unit is also integrated with an alarm unit, to sound an alarm or give a visual indication of the leakage. The sensor has high sensitivity with quick response time at affordable cost. If leakage is detected, message to the particular user or to family member using cellular network called GSM is sent automatically. It also measures the weight of LPG cylinder and displayed in LCD display. A gas quantity of less or equal to10kg,it requests for the new cylinder by automatically sending text message to a distributor. Also when cylinder weighs less than or equal to 0.5 Kg, it informs the consumer by sending a message to refill the cylinder.

CHAPTER 2

HARDWARE OR SOFTWARE TOOLS

2.1 HARDWARE TOOLS

2.1.1 ARDUINO MEGA 2560

The microcontroller board like “Arduino Mega” depends on the ATmega2560 microcontroller. It includes digital input/output pins-54, where 16 pins are analog inputs, 14 are used like PWM outputs hardware serial ports (UARTs) – 4, a crystal oscillator-16 MHz, an ICSP header, a power jack, a USB connection, as well as an RST button. This board mainly includes everything which is essential for supporting the microcontroller. So, the power supply of this board can be done by connecting it to a PC using a USB cable, or battery or an AC-DC adapter. This board can be protected from the unexpected electrical discharge by placing a base plate.

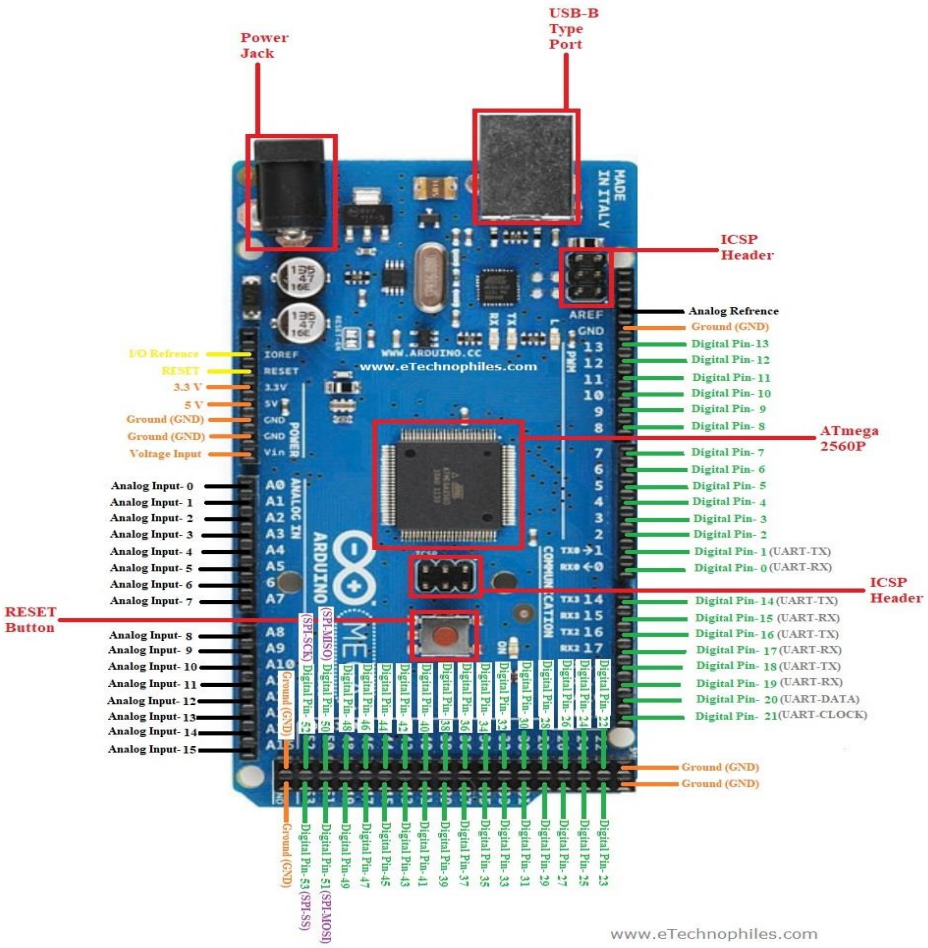


Fig.1. Arduino mega 2560

The SCL & SDA pins of Mega 2560 R3 board connects to beside the AREF pin. Additionally, there are two latest pins located near the RST pin. One pin is the IOREF that

permit the shields to adjust the voltage offered from the Arduino board. Another pin is not associated & it is kept for upcoming purposes. These boards work with every existing shield although can adjust to latest shields which utilize these extra pins.

Arduino Mega Specifications

The specifications of Arduino Mega include the following.

- The ATmega2560 is a Microcontroller
- The operating voltage of this microcontroller is 5volts
- The recommended Input Voltage will range from 7volts to 12volts
- The input voltage will range from 6volts to 20volts
- The digital input/output pins are 54 where 15 of these pins will supply PWM o/p.
- Analog Input Pins are 16
- DC Current for each input/output pin is 40 mA
- DC Current used for 3.3V Pin is 50 mA
- Flash Memory like 256 KB where 8 KB of flash memory is used with the help of bootloader
- The static random-access memory (SRAM) is 8 KB
- The electrically erasable programmable read-only memory (EEPROM) is 4 KB
- The clock (CLK) speed is 16 MHz
- The USB host chip used in this is MAX3421E
- The length of this board is 101.52 mm
- The width of this board is 53.3 mm
- The weight of this board is 36 g

Arduino Mega Pin Configuration

The pin configuration of this Arduino mega 2560 board is shown below. Every pin of this board comes by a particular function which is allied with it. All analog pins of this board can be used as digital I/O pins. By using this board, the Arduino mega projected can be designed. These boards offer flexible work memory space is the more & processing power that permits to work with different types of sensors without delay. When we compare with other types of Arduino boards, these boards are physically superior.

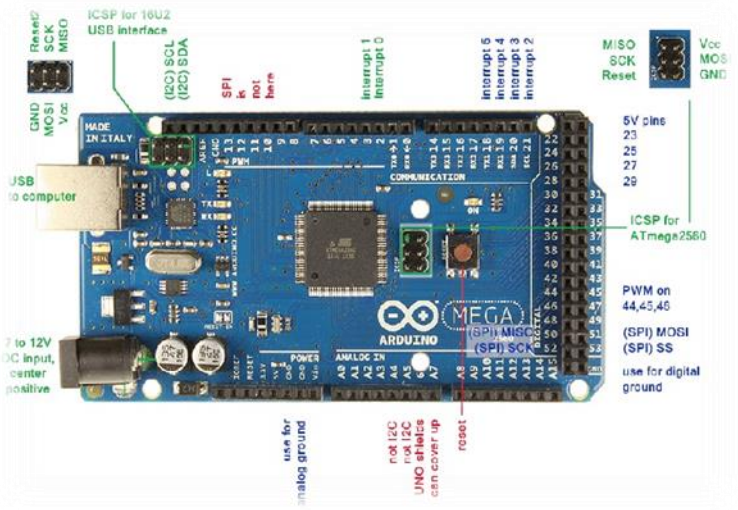


Fig.2.Arduino-mega 2560-board-pin-diagram

Pin 3.3V & 5V

These pins are used for providing o/p regulated voltage approximately 5V. This RPS (regulated power supply) provides the power to the microcontroller as well as other components which are used over the Arduino mega board. It can be attained from Vin-pin of the board or one more regulated voltage supply-5V otherwise USB cable, whereas another voltage regulation can be offered by 3.3V0-pin. The max power can be drawn by this is 50mA.

GND Pin

The Arduino mega board includes 5-GND pins where one of these pins can be used whenever the project requires.

Reset (RST) Pin

The RST pin of this board can be used for rearranging the board. The board can be rearranged by setting this pin to low.

Vin Pin

The range of supplied input voltage to the board ranges from 7volts to 20volts. The voltage provided by the power jack can be accessed through this pin. However, the output voltage through this pin to the board will be automatically set up to 5V.

Serial Communication

The serial pins of this board like TXD and RXD are used to transmit & receive the serial data. Tx indicates the transmission of information whereas the RX indicates receive data. The serial pins of this board have four combinations. For serial 0, it includes Tx (1) and Rx (0), for serial 1, it includes Tx(18) & Rx(19), for serial 2 it includes Tx(16) & Rx(17), and finally for serial 3, it includes Tx(14) & Rx(15).

External Interrupts

The external interrupts can be formed by using 6-pins like interrupt 0(0), interrupt 1(3), interrupt 2(21), interrupt 3(20), interrupt 4(19), interrupt 5(18). These pins produce interrupts by a number of ways i.e. Providing LOW value, rising or falling edge or changing the value to the interrupt pins.

LED

This Arduino board includes a LED and that is allied to pin-13 which is named as digital pin 13. This LED can be operated based on the high and low values of the pin. This will give you to modify the programming skills in real time.

AREF

The term AREF stands for Analog Reference Voltage which is a reference voltage for analog inputs

Analog Pins

There are 16-analog pins included on the board which is marked as A0-A15. It is very important to know that all the analog pins on this board can be utilized like digital I/O pins. Every analog pin is accessible with the 10-bit resolution which can gauge from GND to 5 volts. But, the higher value can be altered using AREF pin as well as the function of analog Reference

I2C

The I2C communication can be supported by two pins namely 20 & 21 where 20-pin signifies Serial Data Line (SDA) which is used for holding the data & 21-pin signifies Serial Clock Line (SCL) mostly utilized for offering data synchronization among the devices

SPI Communication

The term SPI is a serial peripheral interface which is used to transmit the data among the controller & other components. Four pins like MISO (50), MOSI (51), SCK (52), and SS (53) are utilized for the communication of SPI.

Dimensions

The dimension of Arduino Mega 2560 board mainly includes the length as well as widths like 101.6mm or 4 inch X 53.34 mm or 2.1 inches. It is comparatively superior to other types of boards which are accessible in the marketplace. But, the power jack and USB port are somewhat expanded from the specified measurements.

Shield Compatibility

Arduino Mega is well-suited for most of the guards used in other Arduino boards. Before you propose to utilize a guard, confirm the operating voltage of the guard is well-suited

with the voltage of the board. The operating voltage of most of the guards will be 3.3V otherwise 5V. But, guards with high operating voltage can injure the board.

2.1.2 GAS SENSOR MQ2

The MQ-2 Gas sensor can detect or measure gasses like LPG, Alcohol, Propane, Hydrogen, CO, and even methane. The module version of this sensor comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. When it comes to measuring the gas in ppm the analog pin has to be used, the analog pin also TTL driven and works on 5V and hence can be used with most common microcontrollers.

Fig.3. MQ-2 Gas Sensor

FEATURES

- Operating Voltage is +5V
- Can be used to Measure or detect LPG, Alcohol, Propane, Hydrogen, CO and even methane
- Analog output voltage: 0V to 5V
- Digital Output Voltage: 0V or 5V (TTL Logic)
- Preheat duration 20 seconds
- Can be used as a Digital or analog sensor
- The Sensitivity of Digital pin can be varied using the potentiometer

The sensor is actually enclosed in two layers of fine stainless-steel mesh called Anti-explosion network. It ensures that heater element inside the sensor will not cause an explosion, as we are sensing flammable gases. It also provides protection for the sensor and filters out suspended particles so that only gaseous elements are able to pass inside the chamber. The mesh is bound to rest of the body via a copper plated clamping ring.

When tin dioxide (semiconductor particles) is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is adsorbed on the surface of the sensing material. This prevents electric current flow.

In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor.

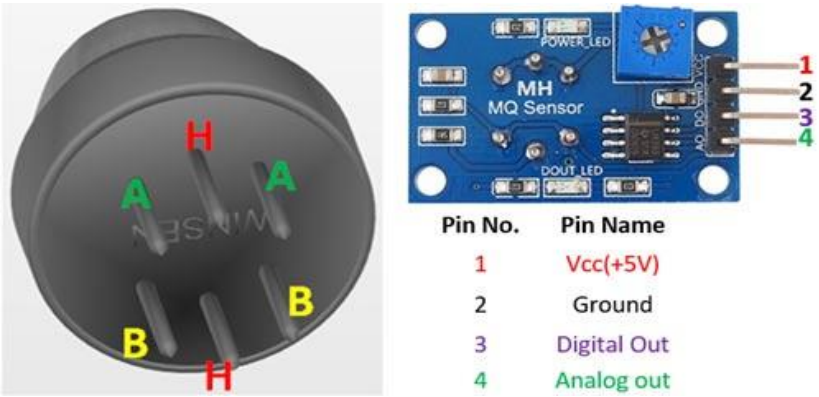


Fig.4. MQ2 Sensor Pinout

- VCC supplies power for the module. You can connect it to 5V output from your Arduino.
- GND is the Ground Pin and needs to be connected to GND pin on the Arduino.
- D0 provides a digital representation of the presence of gas.
- AO provides analog output voltage in proportional to the concentration of gas.

2.1.3 BUZZER

A buzzer or beeper is a signalling device, usually electronic, typically used in automobiles, household appliances such as a microwave oven, or game shows.

Fig.5. Buzzer

It most commonly consists of a number of switches or sensors connected to a control unit that determines if and which button was pushed or a present time has lapsed, and usually illuminates a light on the appropriate button or control panel, and sounds a warning in the form of a continuous or intermittent buzzing or beeping sound. Initially this device was based on an electromechanical system which was identical to an electric bell without the metal gong. Often these units were anchored to a wall or ceiling and used the ceiling or wall as a sounding board. Another implementation with some AC-connected devices was to implement a circuit to make the AC current into a noise loud enough to drive a loudspeaker and hook this circuit up to a cheap 8-ohm speaker. Nowadays, it is more popular to use a ceramic-based piezoelectric sounder like a son alert which makes a high-pitched tone. Usually these were hooked up to "driver" circuits which varied the pitch of the sound or pulsed the sound on and off.

In game shows it is also known as a "lockout system," because when one person signals ("buzzes in"), all others are locked out from signalling. Several game shows have large buzzer buttons which are identified as "plungers".

The word "buzzer" comes from the rasping noise that buzzers made when they were electromechanical devices, operated from stepped-down AC line voltage at 50 or 60 cycles. Other sounds commonly used to indicate that a button has been pressed are a ring or a beep.

2.1.4 SERVO MOTOR

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism. If motor is powered by a DC power supply then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the DC servo motor working. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc.

Fig.6. servo motor

It consists of three parts:

- Controlled device
- Output sensor
- Feedback system

It is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to the control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and reference output signal. So, the main task of servomechanism is to maintain the output of a system at the desired value at presence of noises.

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes,

its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

Servo motor works on PWM (Pulse width modulation) principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically, servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.

Fig.7. Servomotor pinout

FEATURES:

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC

2.1.5 OLED

OLED stands for Organic Light Emitting Diode. The OLED displays are very small and have high resolution. These displays have no back light and they makes their own light. That's why these are very low power devices. These OLED modules are driven by SSD1306 IC which is a driver IC for 128x64 Dot Matrix OLED segments. The SSD1306 has its own controller and supports both SPI and I2C communication protocols. Hence, there are various OLED modules in the market, some that support only SPI communication, some that support only I2C communication, and some that support both I2C and SPI communication. (Different number of pins for different modules) Since the driver IC supports 128x64 resolution, there are some variants that have lesser resolution like 128x32.

Different modules support different colors like blue, yellow, white. Some modules support multiple colors as well. You will need to check the specifications of your display module to know which colors are supported. We are using 4-pin I2C supported 128x64 OLED module similar to the one shown in the above image.

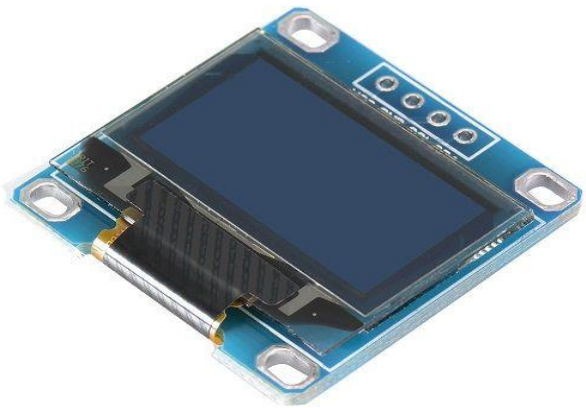


Fig.8. OLED

When the voltage is applied to the OLED. The current flows from the cathode to anode through the organic layers of the OLED. The cathode gives the electrons to the emissive layer of organic molecules and the anode removes electrons from the conductive layer of organic molecules.

At the boundary between the conductive and emissive layer, electron holes are created. These holes are filled by the electrons and the OLED emits light. The color of the OLED depends upon the organic molecules used.

Features

- Resolution: 128 x 64
- Full Compatible with Arduino
- Backlight : OLED self light, no backlight
- Support wide voltage : 3.3V-5V DC
- GND: Connected to the ground of the circuit
- Supply (Vdd,Vcc,5V): Can be powered by either 3.3V or 5V
- SCK (D0,SCL,CLK): The display supports both IIC and SPI, for which clock is supplied through this pin
- SDA (D1,MOSI): This is the data pin of the both, it can either be used for IIC or for SPI

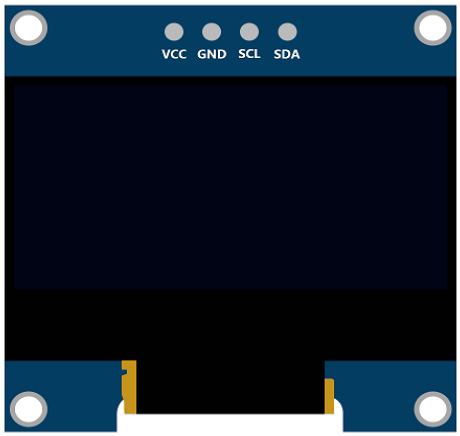


Fig.9. OLED pinout

- VCC: This is the power pin for the module. A supply of 3.3V or 5V can be provided to this pin to power the display.
- GND: This is the ground pin for the module.
- SCL and SDA: These are the serial clock and serial data pins for I2C communication.

2.1.6 LOADCELL

A load cell is a sensor or Transducer then converts electrical signal proportional to the force applied to it.Load cells are commonly used in weigh Scales. One of the most common type of Load Cell is Strain Gauge Type. Strain Gauge type load cells are particularly stiff, have very good resonance values, and tend to have long life cycles in application.

Strain gauge load cells work on the principle that the strain gauge (a planar resistor) deforms/stretches/contracts when the material of the load cells deforms appropriately. These values are extremely small and are relational to the stress and/or strain that the material load cell is undergoing at the time. The change in resistance of the strain gauge provides an electrical value change that is calibrated to the load placed on the load cell.

The output electrical signal generated by the load cell is in millivolts and cannot be sensed by your regular 8 bit or 12 bit ADC. Special Instrumentation amplifiers are required to amplify this low voltage signals before it can be used. Special modules based on IC's which are used in weigh scales are also available. One such module is based on HX711 24-Bit ADC Weigh Scale Module which eliminates the need for costly instrumentation Amplifiers.This is a standard load cell for measuring weight upto 40 Kg. The most common use of this weight sensor is in weighing machine. Every weighing machine which shows weight has a load cell as sensing element.

This conversion is indirect and happens in two stages. Through a mechanical arrangement, the force being sensed deforms a strain gauge. The strain gauge measures the deformation (strain) as an electrical signal, because the strain changes the effective electrical resistance of the wire. A load cell usually consists of four strain gauges in a Wheatstone bridge configuration. Load cells of one strain gauge (quarter bridge) or two strain gauges (half bridge) are also available.

The electrical signal output is typically in the order of a few millivolts and requires amplification by an instrumentation amplifier before it can be used. The output of the transducer is plugged into an algorithm to calculate the force applied to the transducer.Load cells are used in several types of measuring instruments such as weighing scales, universal testing machines.

Fig.10.Loadcell

Weight Sensor (Load Cell) 40KG

- Rated load: 40 Kg.
- Output: 2mv / v.
- Temperature zero drift: 0.1% F.S.
- Output sensitivity: ± 0.15mv / v.
- Temperature sensitivity: 0.05% F.S.
- Insulation resistance: ≥ 2000MΩ
- Excitation voltage: 5-10VDC.

Load cell amplifier -Hx711

This Weight Sensor amplifier is based on HX711, which consists of an amplifier and a precision 24-bit analog-to-digital convertor designed for weigh scale and industrial control applications to interface directly with a bridge sensor. The HX711 uses a two wire interface (Clock and Data) for communication. Compared with other chips, HX711 has added advantages such as high integration, fast response, immunity, and other features improving the total performance and reliability. Finally it's one among the best choices for electronic enthusiasts.

Features:

- Operation Voltage: 2.7V-5V
- Operation Current: < 1.5mA
- Selectable 10SPS or 80SPS output data rate
- Simultaneous 50 and 60Hz supply rejection

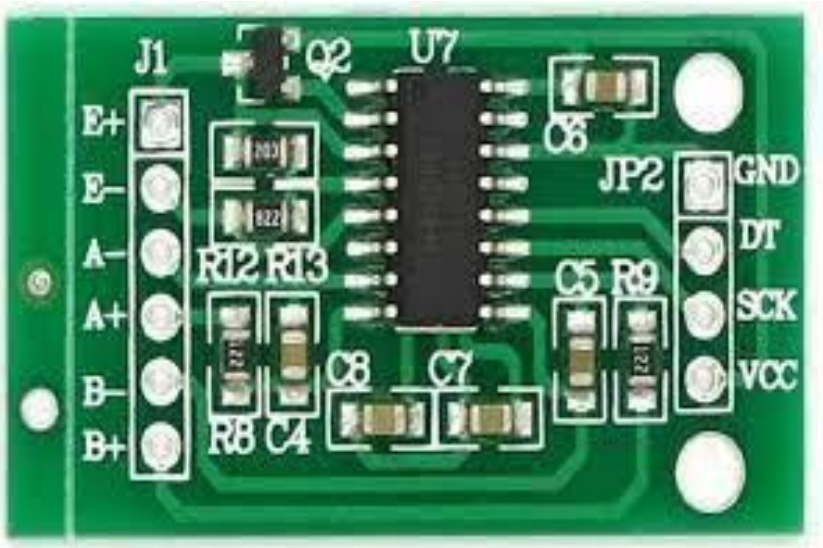


Fig.11.Hx711 Amplifier

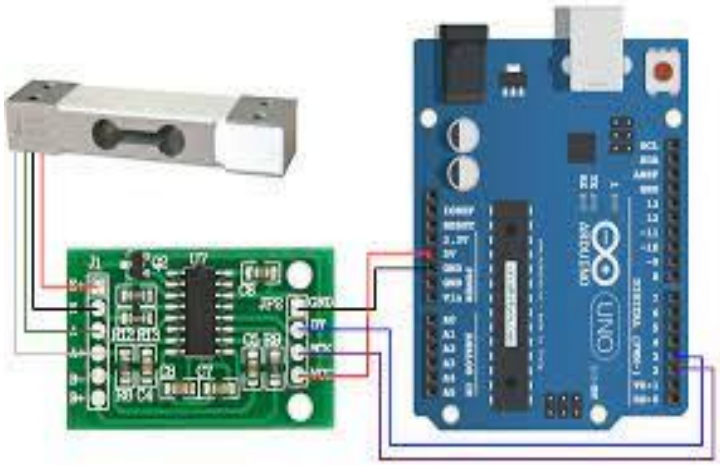
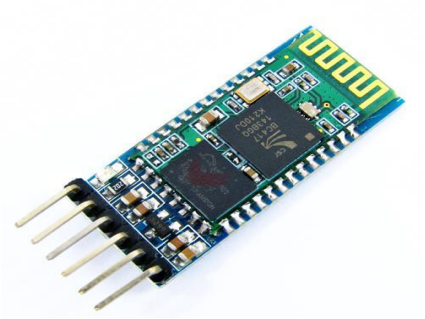


Fig.12.loadcell pinout

2.1.7 BLUETOOTH MODULE HC-05

HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration. HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.

This module works on 3.3 V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator. As HC-05 Bluetooth module has 3.3 V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module



. Fig.13.Bluetooth module (hc-05)

Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.

It has 6 pins,

1. **Key/EN:** It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

   1. **Data mode:** Exchange of data between devices.
   2. **Command mode:** It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

2. **VCC:** Connect 5 V or 3.3 V to this Pin.
3. **GND:** Ground Pin of module.
4. **TXD:** Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)
5. **RXD:** Receive data serially (received data will be transmitted wirelessly by Bluetooth module).
6. **State:** It tells whether module is connected or not.

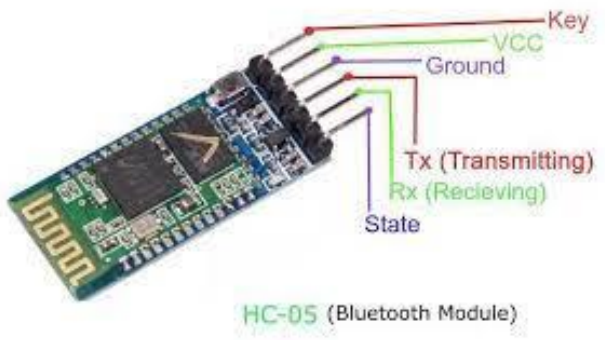


Fig.14.Bluetoothmodule(hc-05) pinout

Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol

- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

2.2 SOFTWARE TOOLS

2.2.1 Arduino IDE

The program code written for Arduino is known as a sketch. The software used for developing such sketches for an Arduino is commonly known as the Arduino IDE. This IDE contains the following parts in it:

- Text editor: This is where the simplified code can be written using a simplified version of C++ programming language.

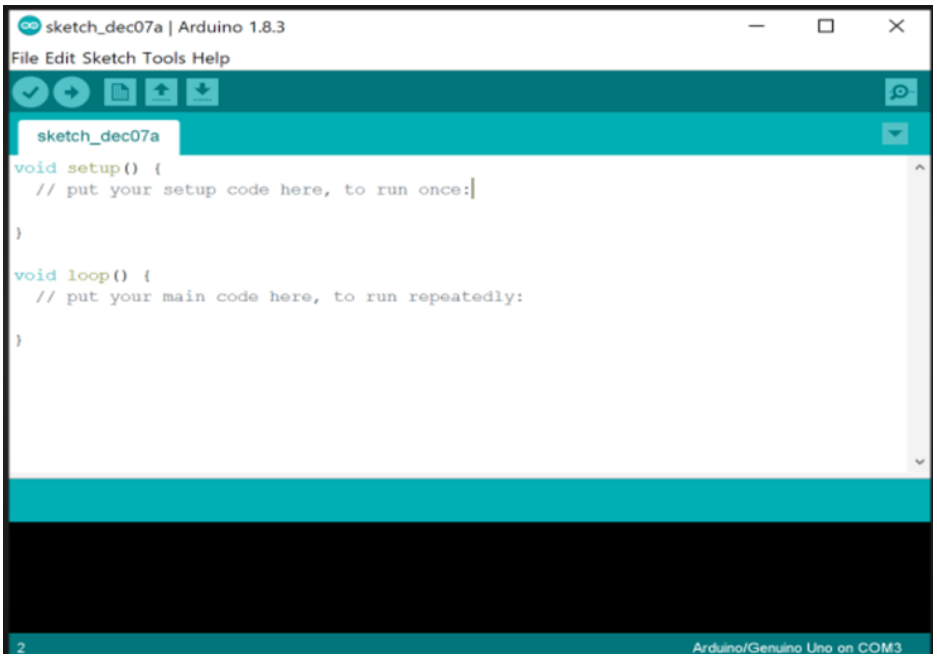


Fig.15. Arduino IDE

- Message area: It displays error and also gives feedback on saving and exporting the code.
- Text: The console displays text output by the Arduino environment including complete error messages and other information
- Console Toolbar: This toolbar contains various buttons like Verify, Upload, New, Open, Save and Serial Monitor. On the bottom right-hand corner of the window there displays the Development Board and the Serial Port in use.

Features of Arduino IDE

- The project file or the sketches for a project are saved with the file extension. ino

- Features such as cut / copy / paste are supported in this IDE.
- There also is a facility for finding a particular word and replacing it with another by pressing the Ctrl + F buttons on the keyboard
- The most basic part or the skeleton of all Arduino code will have two functions

Programming basics

Programming techniques of Arduino sketch in the Arduino IDE. There are two main parts every sketch will always have, they are:

- void setup ()
- void loop ()

1) void setup(): This is the first routine that begins when the Arduino starts functioning. This function is executed only once throughout the entire program functioning.

The setup function contains the initialization of every pin we intend use in our project for input or output. Here is an example of how it should be written:

```
void setup()
  {
     Serial.begin(9600);
  }
```

Here the pin is the no. of the pin that is to be defined. INPUT / OUPUT correspond to the mode in which the pin is to be used.

```
void setup()
  {
    pinMode(pin, INPUT);
    pinMode(pin, OUTPUT);
  }
```

It also contains the initialization of the Serial Monitor. A serial monitor is used to know the data that are being sent serially to any peripheral device. Before using any variables for programming it is necessary to define them above the function “void setup()”

2) void loop (): This function is the next important function in the Sketch. It consists of that part of the code that needs to be continuously executed unlike the part of the code written in the setup function. An example of a void loop is as follows:

```
void loop()
  {
     digitalWrite(pin, HIGH);
  }
```

Here digital Write is a function that writes a high or a low value to a digital pin. If the pin has been configured as an OUTPUT with pin Mode (), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

Similarly, if there is a need for delay in the sketch then there is another function that creates a delay in the execution of the code

```
delay(1000);   //delay for a second
```

This creates a delay in the execution of the program for the time period specified (in milliseconds).

2.2.2 Arduino Bluetooth control application

Arduino Bluetooth Control is an application that allows you to control your Arduino board (and similar boards) via Bluetooth, and so to create awesome and fully customized projects, with the new features available within the app.

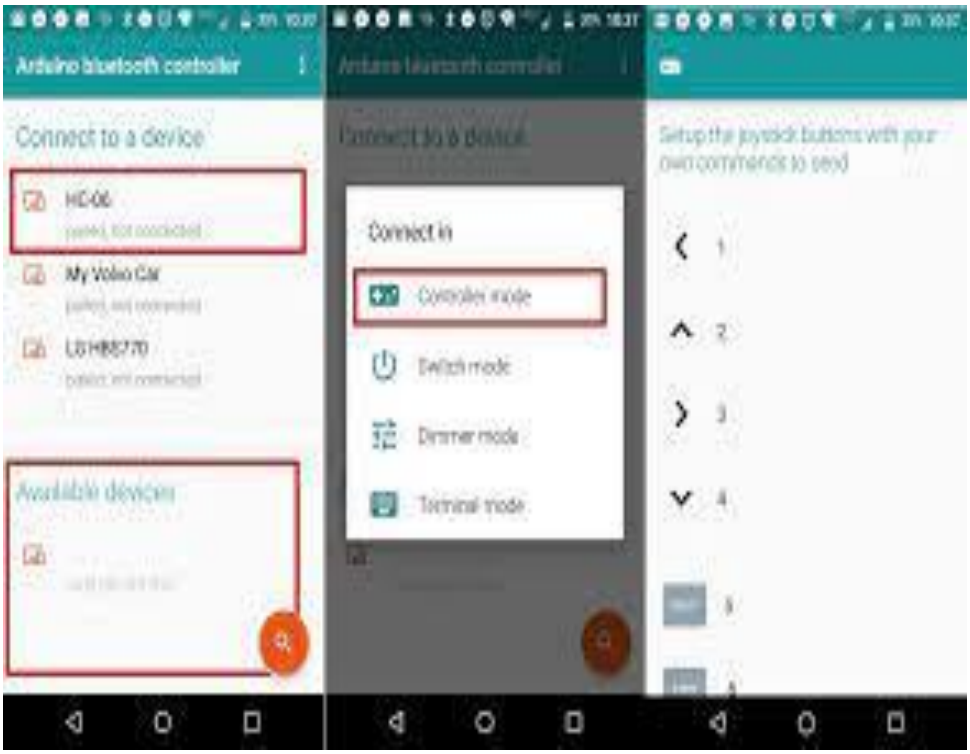


Fig.16.bluetooth control application

CHAPTER 3

PROECT IMPLEMENTATION

3.1 DESCRIPTION OF PROPOSED PROJECT

The block diagram includes MQ2 gas sensor which detects the gas leakage if there is a gas leakage it sends message to the Arduino mega 2560. later the Arduino intimates the user by giving a buzzer and displaying smoke detected on oled, in addition to this the regulator is turned off automatically with the help of servo motor. nextly, loadcell which measures the weight of the cylinder and sends message to the Arduino mega 2560 .in case if the cylinder weight is less the Arduino intimates the user with the help of oled by giving a message book your gas. then Bluetooth module which interfaces Bluetooth control app and Arduino mega 2560 to turn on/off the regulator whenever required

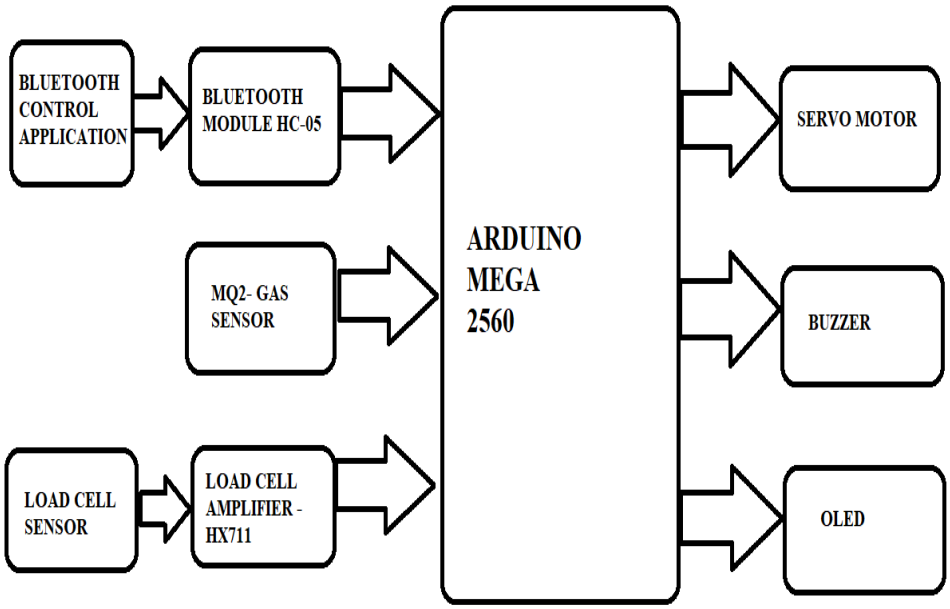


Fig.17. Block Diagram of proposed project

We have included three things in our project

1. Gas leakage detection
2. Turn on/off gas regulator with application
3. Gas level indicator

1. Gas leakage detection

Gas leakage detection the mq2 gas sensor detects the leakage of the lpg gas sends the message to Arduino mega 2560. the Arduino mega 2560 intimates the user by turning on the buzzer as well as automatically turns offs the regulator with the help of servo motor in addition to this it displays a msg as smoke detection on the oled

2. Turn on/off gas regulator with application

when the user is willing to turn on/off the regulator. The user can turn on/off the regulator with the help of Bluetooth control application. whenever the Bluetooth module(hc-05) receives the command either on/off from the application it passes the message to the Arduino at mega 2560. the Arduino mega 2560 intimates the user by displaying the gas on/off on the oled and with the help of servo motor it turns on/off the regulator

3. Gas level indicator

Finally, the gas level indicator the weight of the gas cylinder is measured by the load cell and the output is given to the hx711 which amplifies and digitally converts the load cell output. this output is given to the Arduino mega 2560. the Arduino mega 2560 intimates the user if the level is low and display book your gas on oled

3.2 FLOW CHART

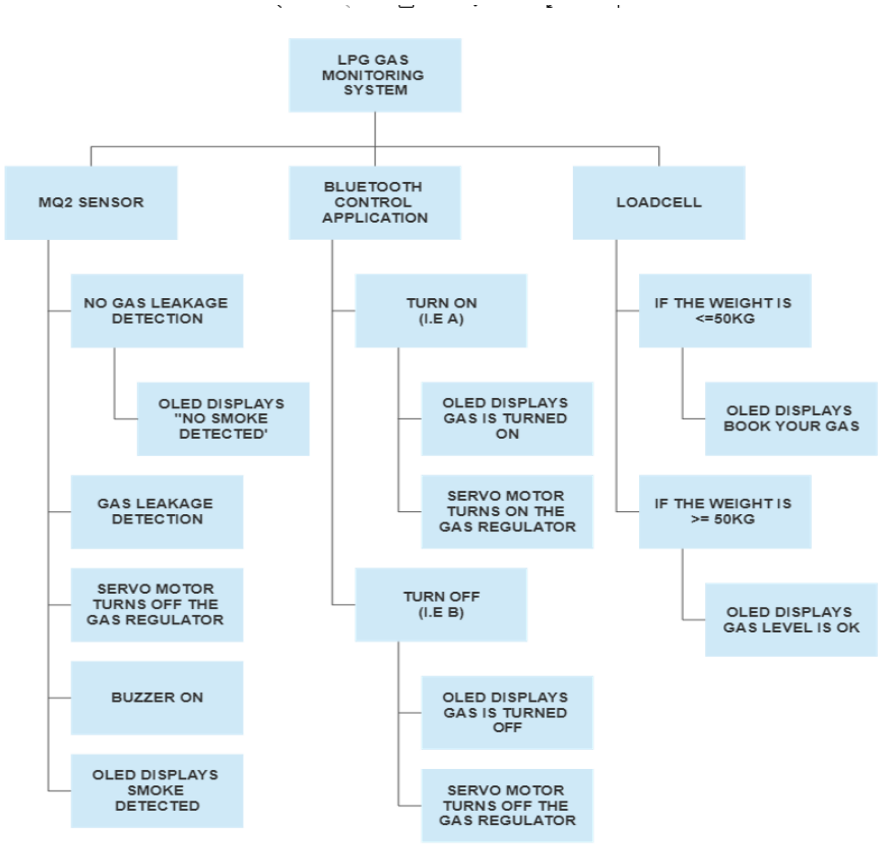


Fig.18. Flow Chart of proposed project

Description of flow chart:

- when there is gas leakage following operations are done to intimate the user there is a buzzer, automatically servomotor turns off the regulator and oled displays smoke detection.
- When there is no leakage of gas the oled displays no smoke detection
- Loadcell measures the weight of the cylinder if the weight is less than 16kg it intimates the user by displaying bk your gas on oled
- if the weight of the cylinder is greater than 16kg it intimates the user by displaying gas level is ok
- if the user is willing to turn on/off the gas cylinder the user can access it with the help of Bluetooth control application

CHAPTER 4

RESULTS AND ANALYSIS

The below figures show the prototype of our proposed system. We have taken MQ2 gas sensor, servo motor, oled, loadcell, buzzer, Bluetooth module (hc-05) all the sensor are connected the Arduino mega 2560

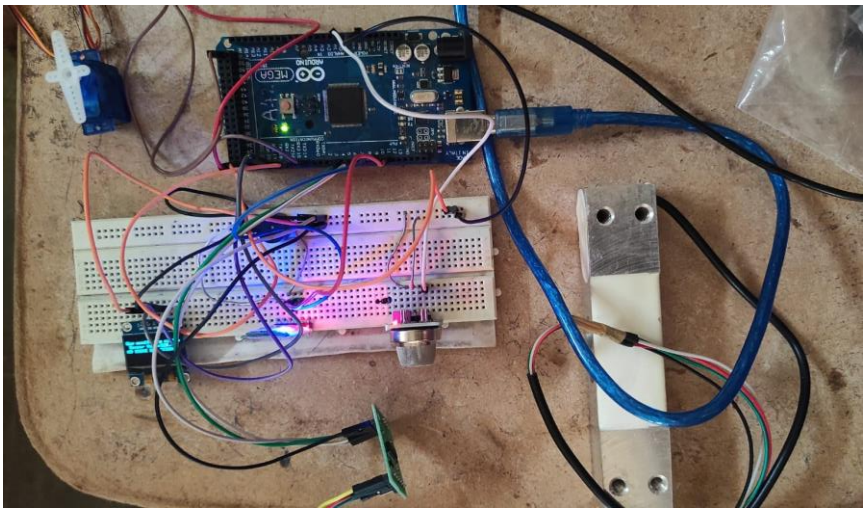


Fig.19. prototype for lpg gas monitoring system

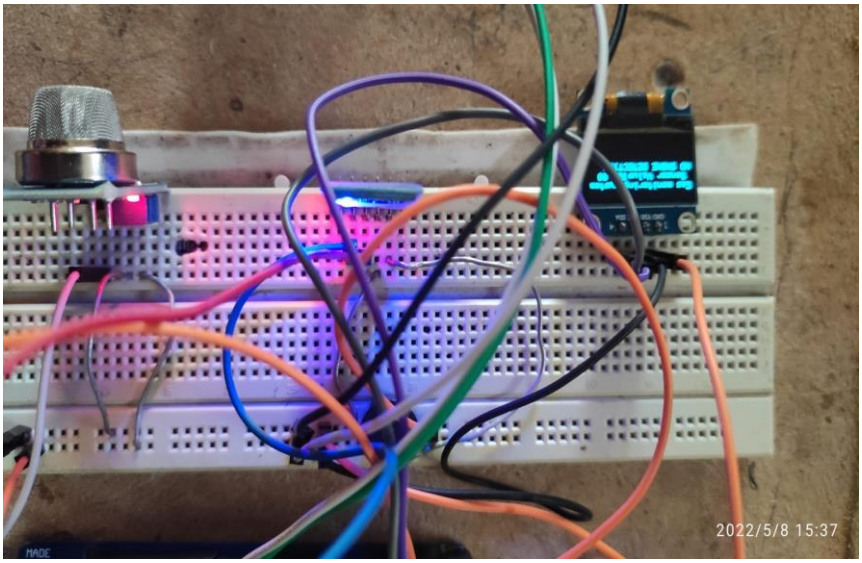


Fig.20.output of the oled display when there is no leakage of gas

The microcontroller is connected to the Laptop/Computer through a USB cable. This cable allows us to connect to the Arduino mega 2560 and upload code into it. We now upload the code we have written in Arduino IDE software to the microcontroller.

Fig.21.output of the oled display when there is gas leakage

In the first step the we need to tare the offset to get accurate measurement then place the mass on the load cell and send the mass value to the arduino to set the cabilartion factor as shown in the below figure

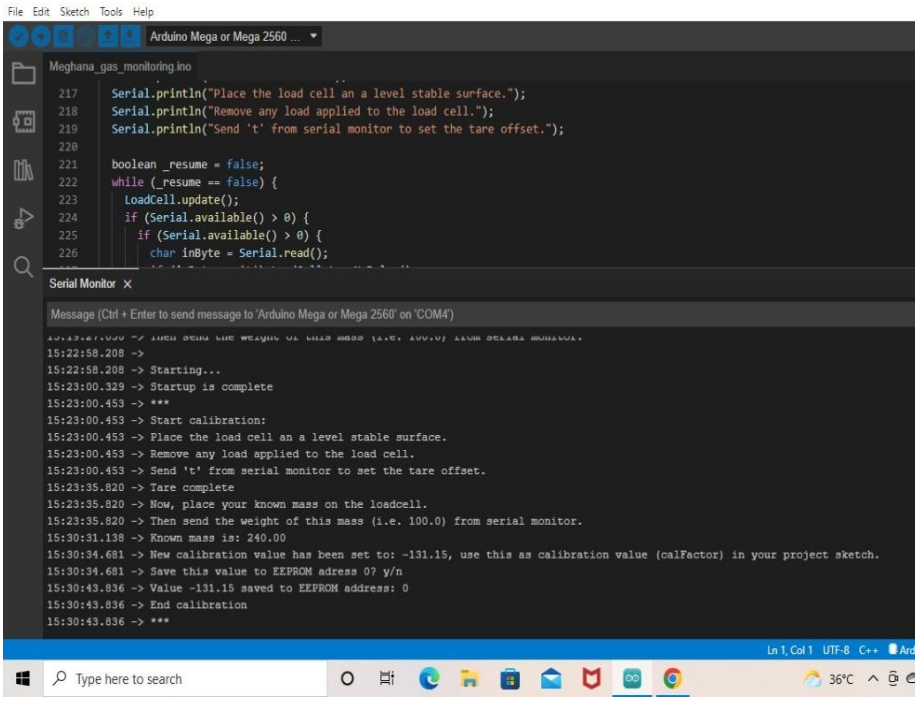


Fig.22.Serial monitor output

Then the load cell starts measuring the weight of the cylinder in case if the load cell weight is less than or equal to 50kg it displays a message “book your gas” else it displays “gas level is ok” as shown in the below figures

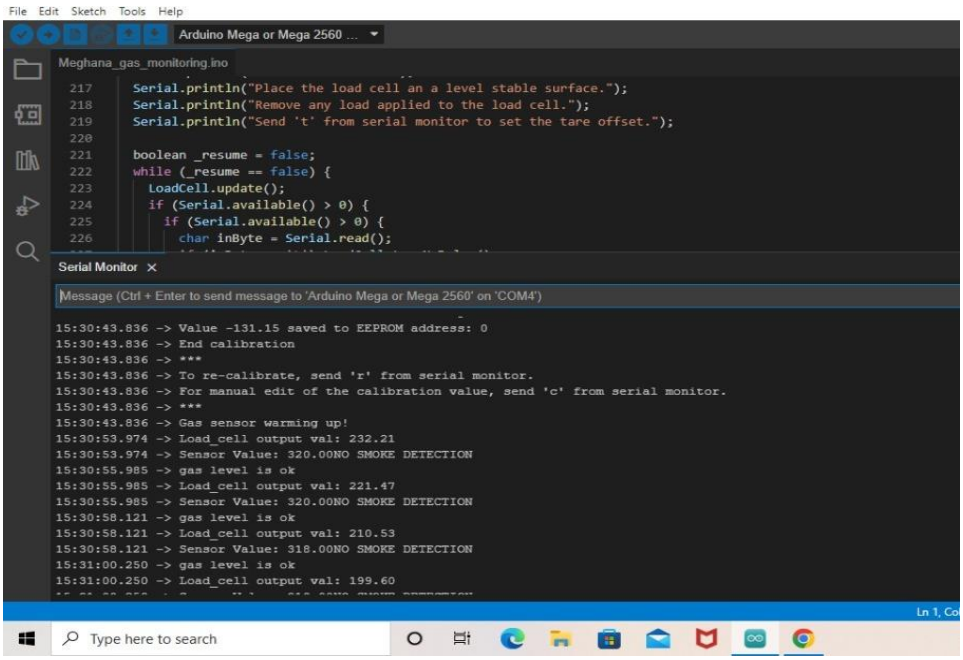


Fig.23.Serial monitor output

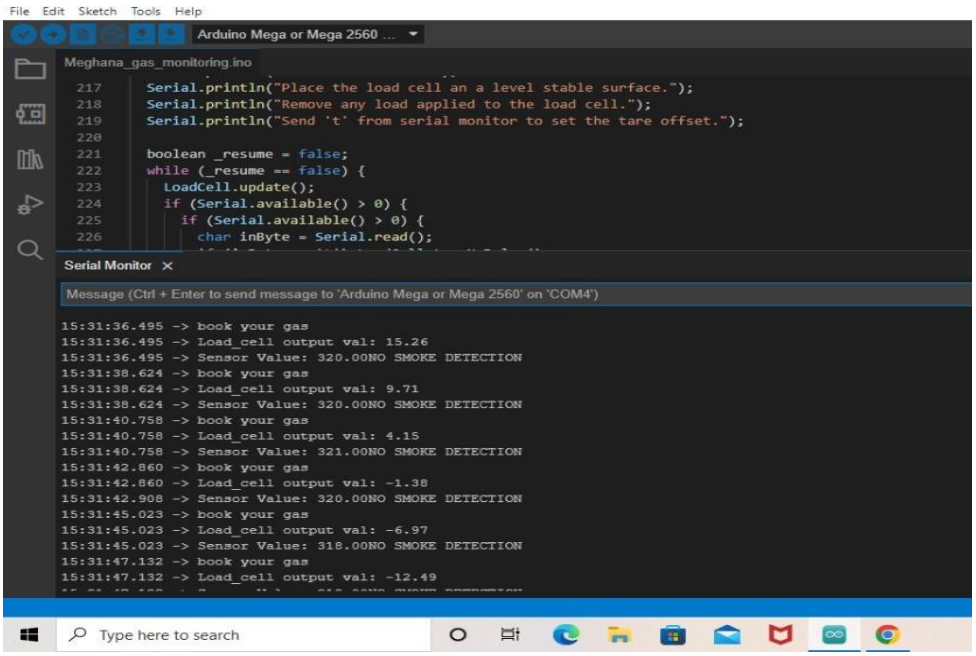


Fig.24.Serial monitor output

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

On the whole, we could say our project “LPG GAS MONITORING SYSTEM” will detect the gas leakage and intimate the user which helps in reducing the accidents due to explosion of LPG gas as well as most of them are unaware of the amount of gas present in the cylinder our system helps in knowing the level of gas in the cylinder and intimates the user accordingly. later we can turn on/off the gas with the help of Bluetooth module

4.2 FUTURE SCOPE

Further this proposed system can be enhanced by controlling the system from anywhere outside the house and also automatically book the gas if the gas level is low without any user involvement

BIBLIOGRAPHY

[1] Jolhe, B. D., P. A. Potdukhe, and N. S. Gawai. "Automatic lpg booking, leakage detection and real time gas measurement monitoring system." International Journal of Engineering Research & Technology (IJERT) 2, no. 4 (2013): 1192-1195.

[2] Soundarya, T. and Anchitaalagammai, J.V., 2014. Control and Monitoring System For Liquefied Petroleum Gas (LPG) Detection And Prevention. International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), 3(3), pp.696-700.

[3] Macker, A., Shukla, A.K., Dey, S. and Agarwal, J., 2018, May. ARDUINO based LPG gas monitoring… automatic cylinder booking with alert system. In 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1209-1212). IEEE.

[4] Kusriyanto, M., Yulianto, A. and Kurniawan, S., 2018. Early detection of LPG gas leakage based Wireless Sensor Networking. In MATEC web of Conferences (Vol. 154, p. 01045). EDP Sciences.

[5] Zakaria, Z., Idroas, M., Samsuri, A., & Adam, A. A. (2017). Ultrasonic instrumentation system for Liquefied Petroleum Gas level monitoring. Journal of Natural Gas Science and Engineering, 45, 428-435.

[6] Shahadat MM, Mallik A, Islam M. Development of an automated gas-leakage monitoring system with feedback and feedforward control by utilizing IoT. Facta universitatis-series: Electronics and Energetics. 2019;32(4):615-31.

[7] Srivastava, D. and Varshini, A., 2021. LPG Gas Monitoring and Cylinder Booking Alert System. International Journal of Progressive Research in Science and Engineering, 2(5), pp.12-17.

[8] Salmani, R., 2016. LPG gas leakage detection & control system. Bonfring International Journal of Software Engineering and Soft Computing, 6(Special Issue Special Issue on Advances in Computer Science and Engineering and Workshop on Big Data Analytics Editors: Dr. SB Kulkarni, Dr. UP Kulkarni, Dr. SM Joshi and JV Vadavi), pp.73-77.

APPENDIX

```
// gas sensor declaration
#define MQ2pin (0)
float sensorValue;

//buzzer declaration
#define buzzer 16

//servo motor declaration
#include <Servo.h>
Servo myservo;
int GasON=0;
int GasOFF=90;

#include <HX711.h>

/*
  ----------------------------------------------------------------------------
  HX711_ADC
  Arduino library for HX711 24-Bit Analog-to-Digital Converter for Weight Scales
  Olav Kallhovd sept2017
  ----------------------------------------------------------------------------
*/

/*
  This example file shows how to calibrate the load cell and optionally store the calibration
  value in EEPROM, and also how to change the value manually.
  The result value can then later be included in your project sketch or fetched from EEPROM.
```

To implement calibration in your project sketch the simplified procedure is as follow:

```
    LoadCell.tare();
    //place known mass
    LoadCell.refreshDataSet();
    float newCalibrationValue = LoadCell.getNewCalibration(known_mass);
*/
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:
```

```
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;
float newCalibrationValue;

void setup() {

  Serial.begin(9600); delay(10);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 20);
  display.println("Gas monitoring system");
  display.display();

  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
```

```
  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value to positive
  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be performed in the next step
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
    LoadCell.setCalFactor(1.0); // user set calibration value (float), initial value 1.0 may be used for this sketch
    Serial.println("Startup is complete");
  }
  while (!LoadCell.update());
  calibrate(); //start calibration procedure

 //servo motor
 myservo.attach(14);
  Serial.println("Gas sensor warming up!");
  delay(10000); // allow the MQ-6 to warm up
   mySerial.begin(4800);
  myservo.write(0);
  pinMode(buzzer,OUTPUT);
}

void loop() {
  display.clearDisplay();
   delay(3000);
  display.setTextSize(1);
  display.setTextColor(WHITE);
```

```
if (mySerial.available()) {
  Serial.write(mySerial.read());
 }
 if (Serial.available()) {
  mySerial.write(Serial.read());
 }


 static boolean newDataReady = 0;
 const int serialPrintInterval = 0; //increase value to slow down serial print activity

 // check for new data/start next conversion:
 if (LoadCell.update()) newDataReady = true;

 // get smoothed value from the dataset:
 if (newDataReady) {
  if (millis() > t + serialPrintInterval)
  {
    float i = LoadCell.getData();
    Serial.print("Load_cell output val: ");
    Serial.println(i);

    display.setCursor(40, 50);
    display.println(i);
     display.display();
      newDataReady = 0;
     t = millis();
  }
 }
```

```
  // receive command from serial terminal
  if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay(); //tare
    else if (inByte == 'r') calibrate(); //calibrate
    else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value manually
  }

  // check if last tare operation is complete
  if (LoadCell.getTareStatus() == true) {
    Serial.println("Tare complete");
  }
sensorValue = analogRead(MQ2pin); // read analog input pin 0
  //mq2 gas sensor
  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);


  display.setCursor(0, 0);
  display.println(" Sensor Value:");
  display.display();
  display.setCursor(80, 0);
  display.println(sensorValue);
  display.display();

  if(sensorValue > 350)
  {
     myservo.write(GasOFF);
     digitalWrite(buzzer,HIGH);
```

```
    Serial.println(" | Smoke detected!");
    display.clearDisplay();
    display.setTextSize(2);

    display.setTextColor(WHITE);
    display.setCursor(0, 30);
    display.println("  | Smoke detected!");
    display.display();
     delay(3000);
  }
  else
  {
     digitalWrite(buzzer,LOW);
     Serial.println("NO SMOKE DETECTION");
     //display.clearDisplay();
     display.setCursor(0, 10);
     display.println("NO SMOKE DETECTION");
     display.display();
     delay(2000); // wait 2s for next reading
  }
/*int z= analogRead(14);
float Q= map(z,0,1023,0,255);
   if(Q<=0){
     display.setCursor(0, 40);
     display.println("GAS is turned ON");
     display.display();
   }
    else{
     display.setCursor(0, 40);
     display.println("GAS is Turned OFF");
```

```
    display.display();
  }*/
if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 'A')
    {
   myservo.write(0);
   display.setCursor(0, 40);
   display.println("GAS is turned ON");
   display.display();
   Serial.print("GAS is turned ON ");
    }
else if(inByte == 'B')
    {
    Serial.print("GAS is Turned OFF ");
    myservo.write(90);
    display.setCursor(0, 20);
    display.println(" GAS is Turned OFF");
    display.display();
  }
}
 if( LoadCell.getData() <=50)
        {
          Serial.println("book your gas");
          //display.clearDisplay();
         display.setCursor(0, 30);
         display.println(" book your gas");
          display.display();
        }
        else
```

```
        {
        Serial.println("gas level is ok");
         //display.clearDisplay();
        display.setCursor(0, 30);
        display.println(" gas level is ok");
         display.display();
        }

 }

void calibrate() {
  Serial.println("***");
  Serial.println("Start calibration:");
  Serial.println("Place the load cell an a level stable surface.");
  Serial.println("Remove any load applied to the load cell.");
  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      if (Serial.available() > 0) {
        char inByte = Serial.read();
        if (inByte == 't') LoadCell.tareNoDelay();
      }
    }
    if (LoadCell.getTareStatus() == true) {
      Serial.println("Tare complete");
      _resume = true;
    }
```

```
  }

  Serial.println("Now, place your known mass on the loadcell.");
  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");

  float known_mass = 0;
  _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      known_mass = Serial.parseFloat();
      if (known_mass != 0) {
        Serial.print("Known mass is: ");
        Serial.println(known_mass);
        _resume = true;
      }
    }
  }

  LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known mass is measured correct
    newCalibrationValue = LoadCell.getNewCalibration(known_mass); //get the new calibration value

  Serial.print("New calibration value has been set to: ");
  Serial.print(newCalibrationValue);
  Serial.println(", use this as calibration value (calFactor) in your project sketch.");
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
```

```
  _resume = false;
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;

      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

  Serial.println("End calibration");
  Serial.println("***");
  Serial.println("To re-calibrate, send 'r' from serial monitor.");
```

```
  Serial.println("For manual edit of the calibration value, send 'c' from serial monitor.");
  Serial.println("***");
}
void changeSavedCalFactor() {
  float oldCalibrationValue = LoadCell.getCalFactor();
  boolean _resume = false;
  Serial.println("***");
  Serial.print("Current value is: ");
  Serial.println(oldCalibrationValue);
  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
  float newCalibrationValue;
  while (_resume == false) {
    if (Serial.available() > 0) {
      newCalibrationValue = Serial.parseFloat();
      if (newCalibrationValue != 0) {
        Serial.print("New calibration value is: ");
        Serial.println(newCalibrationValue);
        LoadCell.setCalFactor(newCalibrationValue);
        _resume = true;
      }
    }
  }
  _resume = false;
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
```

```
#if defined(ESP8266)|| defined(ESP32)
      EEPROM.begin(512);
#endif
      EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
      EEPROM.commit();
#endif
      EEPROM.get(calVal_eepromAdress, newCalibrationValue);
      Serial.print("Value ");
      Serial.print(newCalibrationValue);
      Serial.print(" saved to EEPROM address: ");
      Serial.println(calVal_eepromAdress);
      _resume = true;
     }
     else if (inByte == 'n') {
      Serial.println("Value not saved to EEPROM");
      _resume = true;
     }
   }
  }
  Serial.println("End change calibration value");
  Serial.println("***");
}
```