UNITED INTERNATIONAL UNIVERSITY (UIU)

Dept. of Computer Science & Engineering

Course No: CSE 4326

Course Title: Microprocessors and Microcontrollers Laboratory

Experiment 1:An introduction to Arduino and Interfacing of Gas Sensor using Arduino & Showing the Sensor Data in OLED Display

Objectives:

In this lab, we will learn how to interface a gas sensor with an Arduino and display the sensor data on an OLED display. The experiment is divided into two parts. First, we will learn details about the arduino uno board and how arduino can be used to perform simple hardware projects. In the second part, we will use an important sensor called, gas sensor which detects presence of different gases in the environment. This sensor will be used to detect the presence of gas in the environment, and the Arduino will read the data from the sensor and display it on the OLED display. This lab will teach you how to work with Arduino, sensors and digital displays, as well as how to interface them together to create a complete system. This will definitely give you an idea on how to do mini projects using an Arduino board. Final outcomes of this experiment will be-

- To learn about Arduino boards and Arduino IDE.
- Examples of hardware and software interface using Arduino board.
- Examples of emulating arduino projects in Proteus software.
- Interfacing gas sensor with arduino.
- Using an OLED display to show sensor data.

Components required:

Hardware:

1) Arduino Uno board
2) LED
3) MQ-2 Gas Sensor
4) OLED Display (SSD1306)
5) Breadboard
6) Jumper wires
7) 10k ohm Resistor

8) Push button

Software:

1) Proteus
2) Arduino IDE

Introduction to Arduino:

(a) Arduino Board:

Arduino is an open-source hardware and software platform designed for building digital devices and interactive objects. It provides a user-friendly way for makers, hobbyists, and professionals to create and prototype various electronic projects. The core of Arduino is the **Arduino board**, which typically consists of a microcontroller, input and output pins, and a programming interface. There are various types of Arduino boards such as Arduino Nano, Uno, Mega Boards. Arduino Uno is sufficient to perform mid level projects and that's why in this course, we are mainly focusing on the Arduino Uno board. Fig. 1 shows a typical image of the Arduino Uno board.

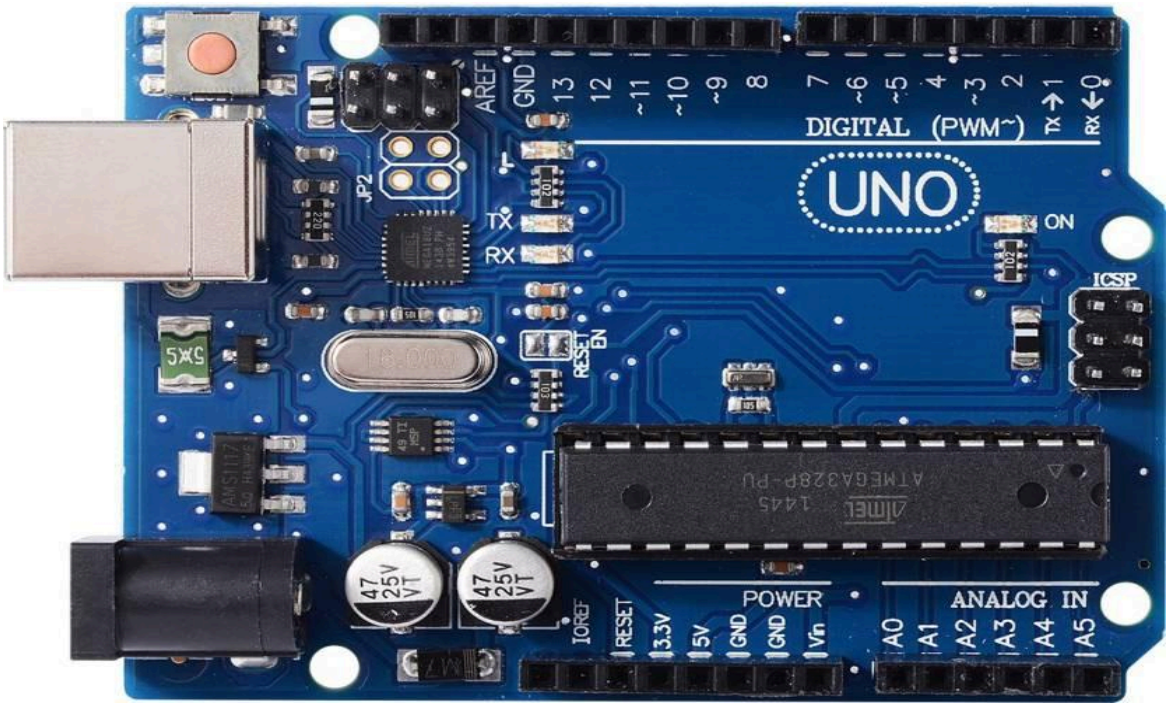


Fig 1: Arduino Uno Board

The Arduino Uno board (Fig. 1) has both digital and analog pins that can be used for various input and output tasks. Here's a breakdown of these pins:

Digital Pins: These pins can be used for digital input or output. They support digital signals, meaning they can be in either a HIGH (5V) or LOW (0V) state.

Analog Pins: Analog Pins A0 to A5 (also referred to as Analog Inputs)- these pins can read analog voltage levels, typically ranging from 0 to 5 volts, and provide 10-bit resolution (0-1023) using the built-in ADC. They are often used for reading sensors that provide analog outputs.

Additionally, it's worth noting that some pins on the Arduino Uno have special functions:

- **Pins 0 (RX) and 1 (TX):** These pins are used for serial communication (UART) with other devices, like your computer for programming and serial debugging.

Applications: There are many hardware and software applications of Arduino. For example, home automation, robotics, IoT, environmental monitoring, security systems, weather monitoring, smart agriculture, healthcare monitoring and many more.You can use arduino to implement your innovative ideas and projects.

(b) Arduino IDE: Figure 2 shows interface of Arduino IDE.

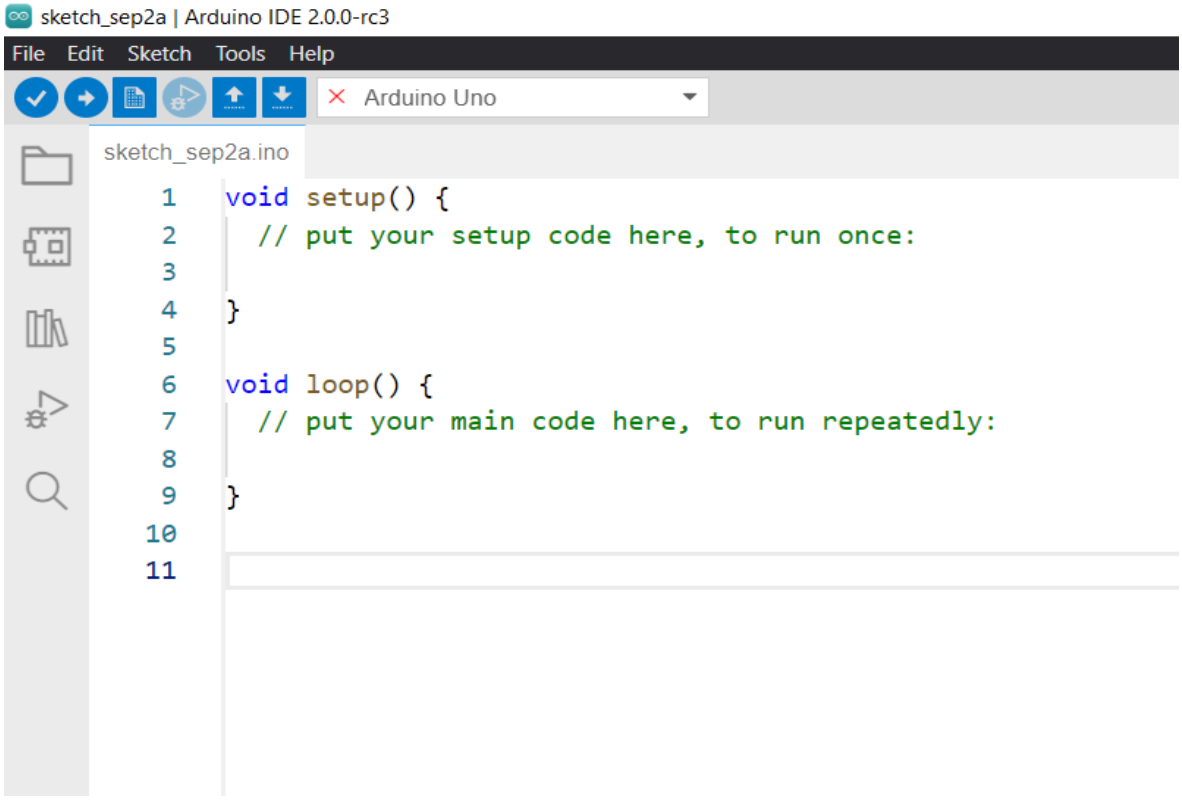


Fig 2: Arduino IDE

Two required functions / methods / routines:

```
void setup()

{

// runs once

}

void loop()

{

// repeats  }
```

In Arduino, the **setup()** and **loop()** functions are fundamental to how a program runs on the microcontroller. These functions define the behavior of your Arduino sketch (program). Here's what each of them does:

setup() function:

- The setup() function is called once when the Arduino board is powered on or reset.
- It is typically used for initializing variables, setting the pin modes (input or output), and any other one-time setup tasks.
- You can use pinMode(), Serial.begin(), and other initialization commands in this function.

loop() function:

- The loop() function is where your main code resides. It runs continuously after the setup() function completes.
- Anything placed in the loop() function will be executed repeatedly as long as the Arduino is powered on or until you manually reset it.
- You can use this function to read sensors, control outputs, and implement your desired functionality.

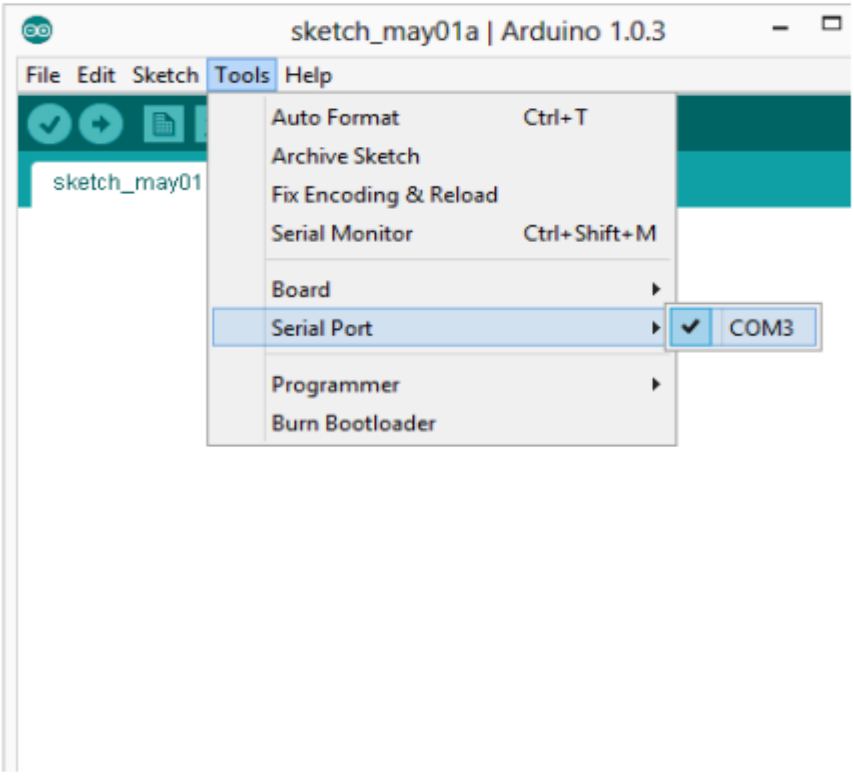


Figure: Settings: Tools → Serial Port

Fig 3: Arduino IDE connection with serial port

Your computer communicates to the Arduino microcontroller via a serial port through a USB-Serial adapter(figure 3).Check to make sure that the drivers are properly installed.

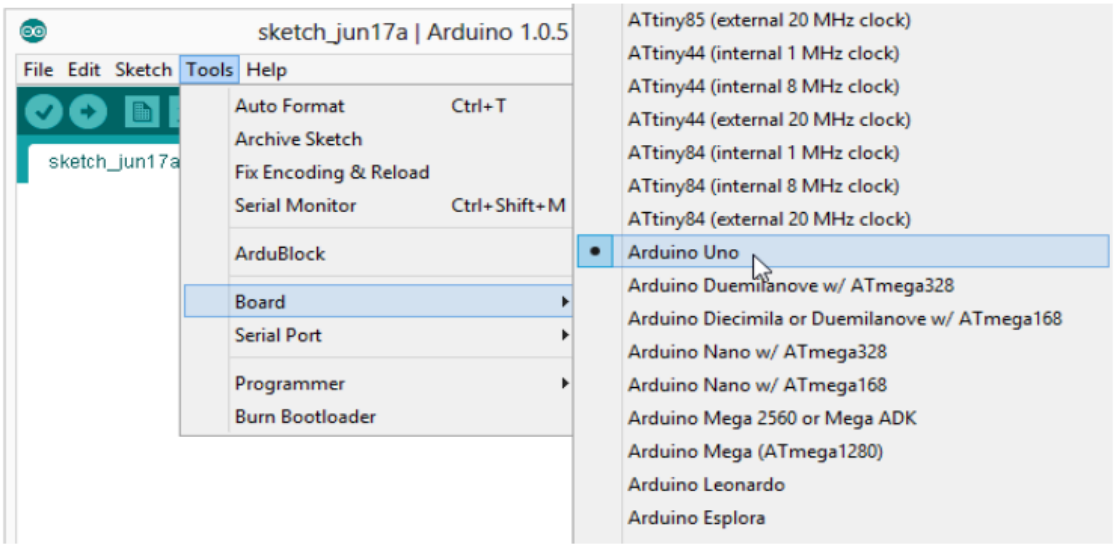


Figure : Settings: Tools → Board

Fig 4: Arduino board selection

Next, double-check that the proper board is selected under the Tools Board menu as shown in figure 4.

(c)Simple hardware connection:

(i)Toggling an LED:

Here,we will turn on an LED for some time(1000 ms) and then turn off the LED for a definite time (1000 ms) and keep doing it, which is known as **"Toggling".**

Basic Syntax Description:
In Arduino, the 'pinMode' function is used to set the mode of a digital I/O (input/output) pin.
For example,
pinMode(2, INPUT); // Configures pin 2 as an input
You can use this mode when you want to **read** the state of a sensor or external device connected to the pin.
pinMode(13, OUTPUT); // Configures pin 13 as an output
This mode configures the specified pin as an output, allowing you to **send** digital signals from the Arduino to external components like LEDs, motors, or relays.
Note that, **"//"** is used to denote a comment.
delay (1000) ; // introduces a time delay of 1000 mili seconds = 1 second
In Arduino, the **digitalWrite** function is used to set the state of a digital I/O (input/output) pin to either HIGH (5V or the supply voltage) or LOW (0V or ground). This function is commonly used to control external components such as LEDs, relays, motors, and more.

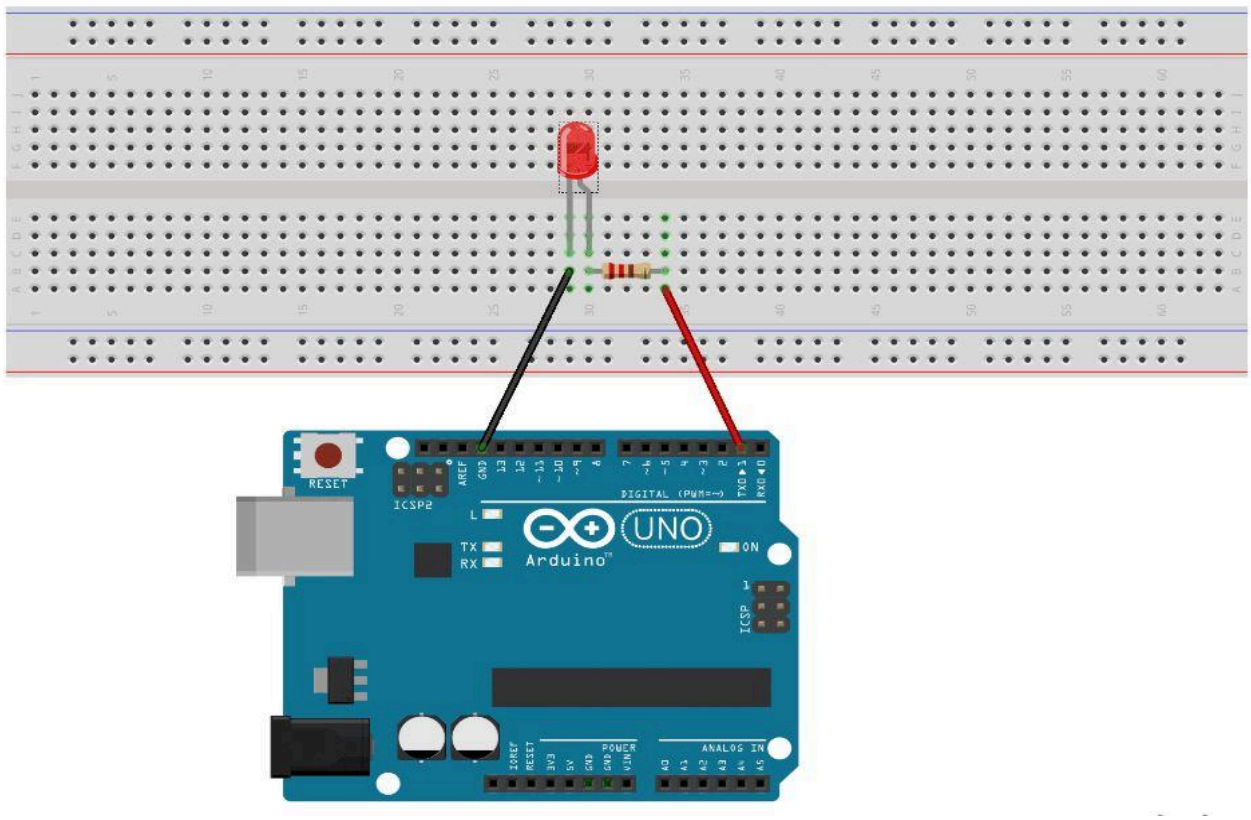


Fig 5: Schematic Diagram of c(i)

Code:

```
// the setup function runs once when you press reset or power the board

void setup() //this is a comment

{

// initialize digital pin 1 as an output.

pinMode(1, OUTPUT);//the resistor here just controls the current through LED

}

// the loop function runs over and over again forever

void loop()

{
```

```
digitalWrite(1, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(1, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}
```

(d)Adding Arduino Library into Proteus:

Download Arduino → https://www.arduino.cc/en/Main/Software

Download Arduino library for Proteus → ELMS course website Proteus library Arduino library for proteus ARDUINO.LIB

Step 1: Copy ARDUINO.LIB in the in to Library folder

If you are using Proteus 8 then Library folder will be within the data Folder (figure 6).

(Proteus 8 Professional\Data\LIBRARY)

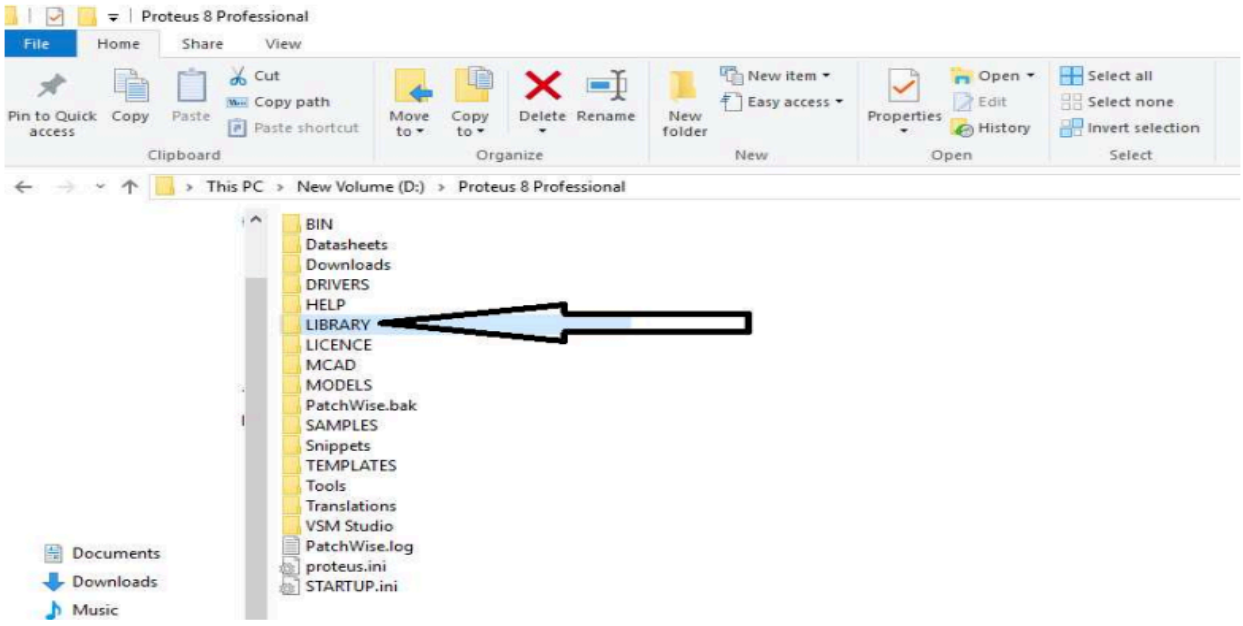


Fig 6: Navigating through library folder

Step 2: Now open Your Proteus and Search Arduino as shown in figure 7.

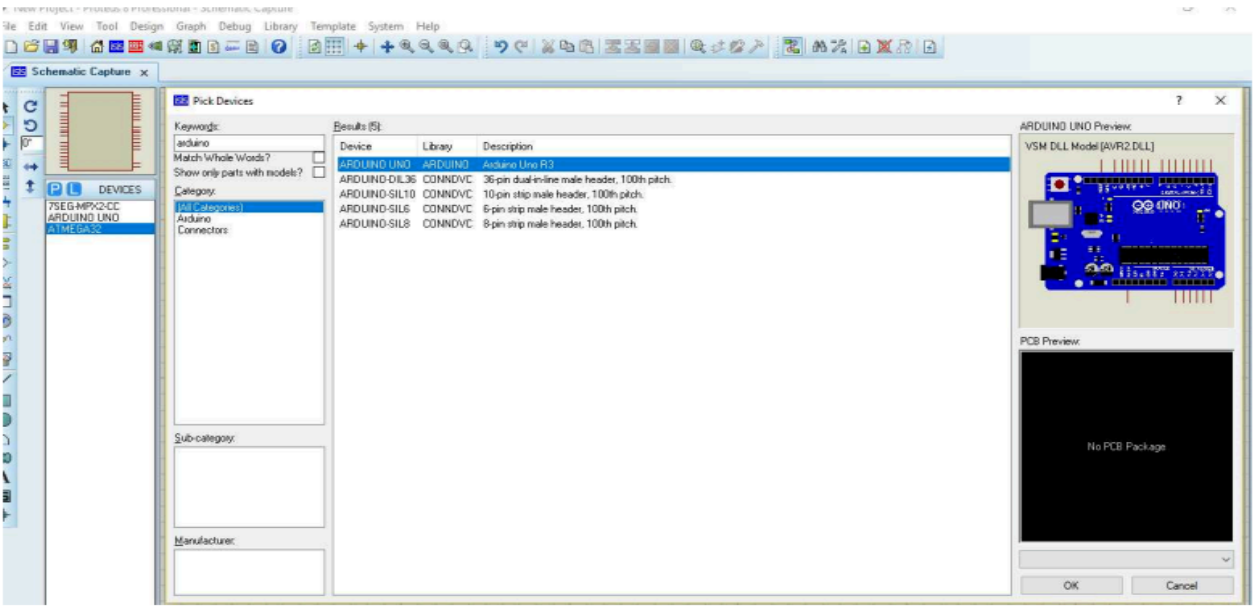
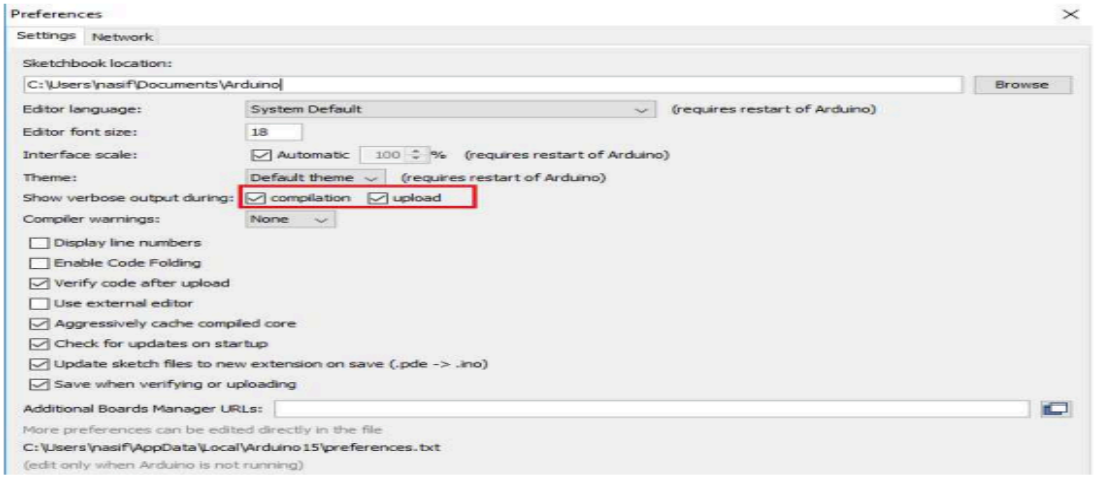


Fig 7: Choosing Arduino board from proteus parts

Step 3: Make a Hex file as shown in figure 8.



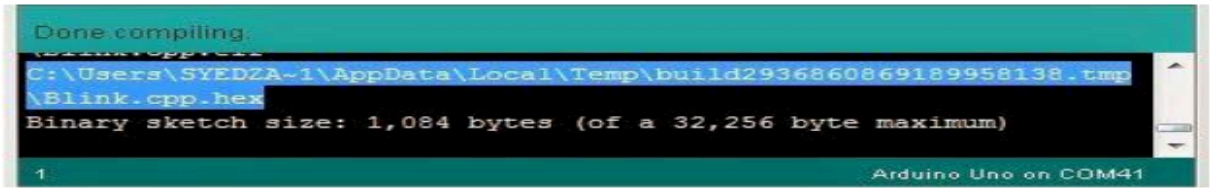


Fig 8: Compilation and upload of code

Open Arduino Software and Click on file and then Preference and tick both of the Option "Compilation and Upload " After writing your code, click on compile then you will get the Link of Your Hex file at the Output go to that place and get your Hex file.

Step 4: Upload Code
It is Time to upload Code to Arduino as shown in figure 9. After uploading,just run your proteus file to see results!

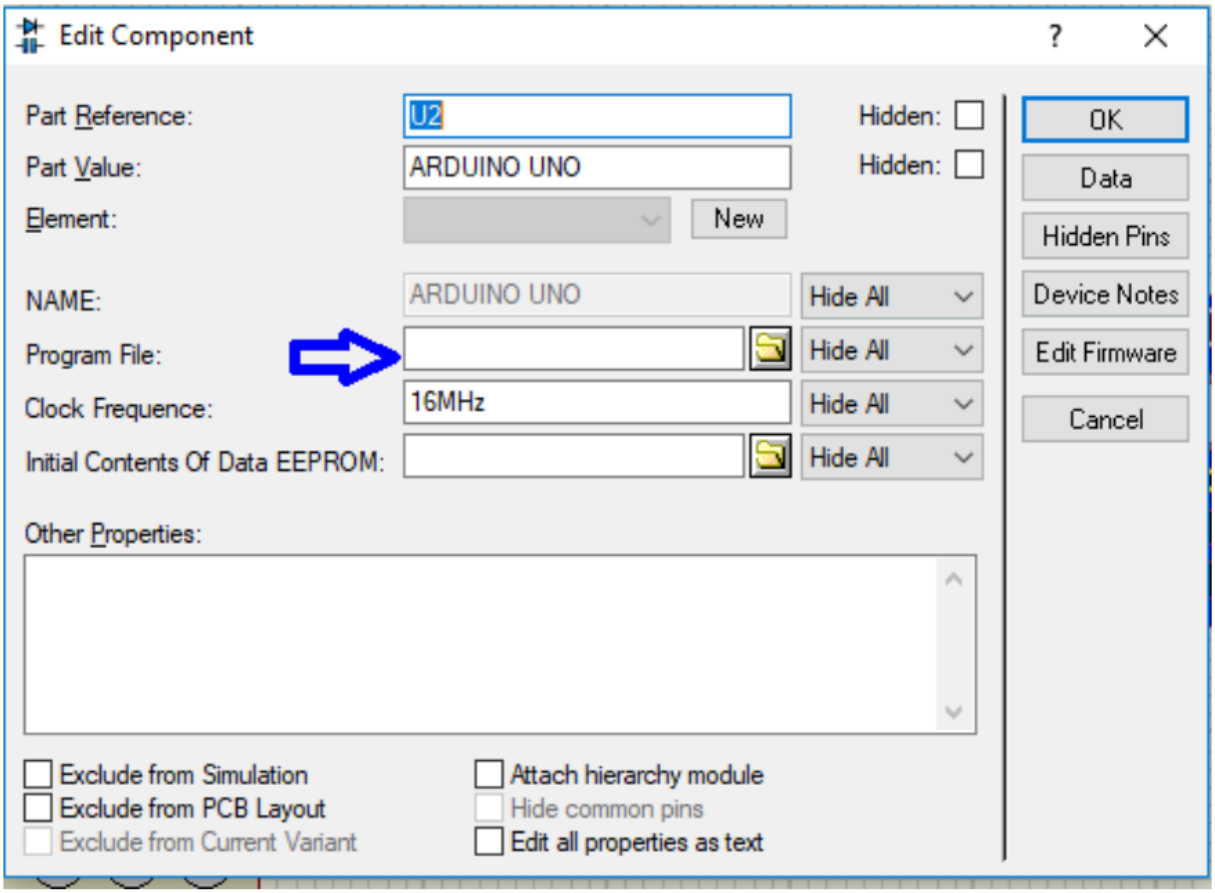


Fig 9: Uploading HEX file in proteus

This is how an arduino code can be uploaded in Proteus for simulation purposes.

You may wonder why we need Proteus simulation. This is due to the fact that, while working on a big project,doing trial and error with hardware is troublesome.Finding out results using simulation softwares like Proteus can save time and resources.

Proteus simulation tutorial:

(i) Print a single string: Connect arduino to virtual terminal as shown in fig.10. Note that,**RX of**

arduino is connected to TXD of virtual terminal and TX of arduino is connected to RXD of virtual terminal.

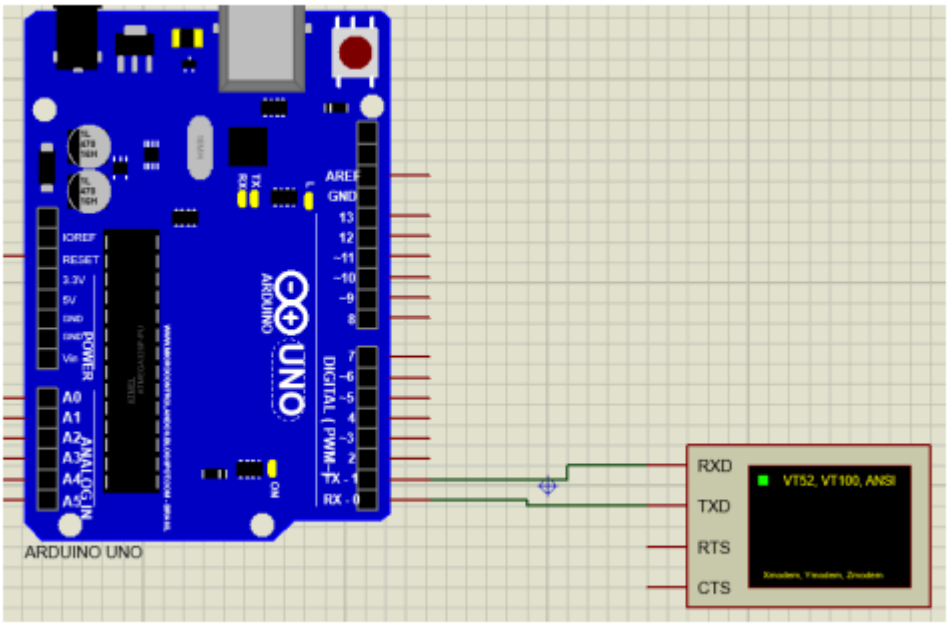


Figure: Proterus "virtual terminal" serial communication simulation

Fig 10: Connecting virtual terminal with Arudino in Proteus

Code:

```
void setup()

{

Serial.begin(9600); // used to initialize the communication between the Arduino
//board and a computer or other devices via the serial communication interface.Baud
//rate is the number of bits transmitted per second and determines  how fast data is
//sent and received. 9600 here is baud rate.

}

void loop()

{

Serial.println("Hello from Arduino !!! ");

//prints Hello from Arduino!! In serial monitor

delay(300); // a delay of 300 ms
```

```
}
```

(ii)Read a string:

Code:

```
void setup()

{

Serial.begin(9600);

}



void loop()

{

if(Serial.available()>0)

{

String s = Serial.readString(); // read a single 'string'

Serial.print("Reply from Arduino: ");// Serial.print() prints in the same line

Serial.println(s);} //Serial.println() prints in a new line

}
```

(iii) Push button interfacing with Arduino:

Internal to the Arduino are pullup resistors with a value around 50k-ohm. These resistors can be optionally connected internally using INPUT_PULLUP. This is functionally (and electrically) equivalent to connecting a 50k-ohm resistor between the pin and +5V, the only difference is that it requires no external components and you can turn it on and off in software during the execution of the program. The pin mode of INPUT_PULLUP means that the pin is to be used as an input, but that if nothing else is connected to the input it should be 'pulled up' to HIGH. In other words, the default value for the input is HIGH, unless it is pulled LOW by the action of pressing the button.

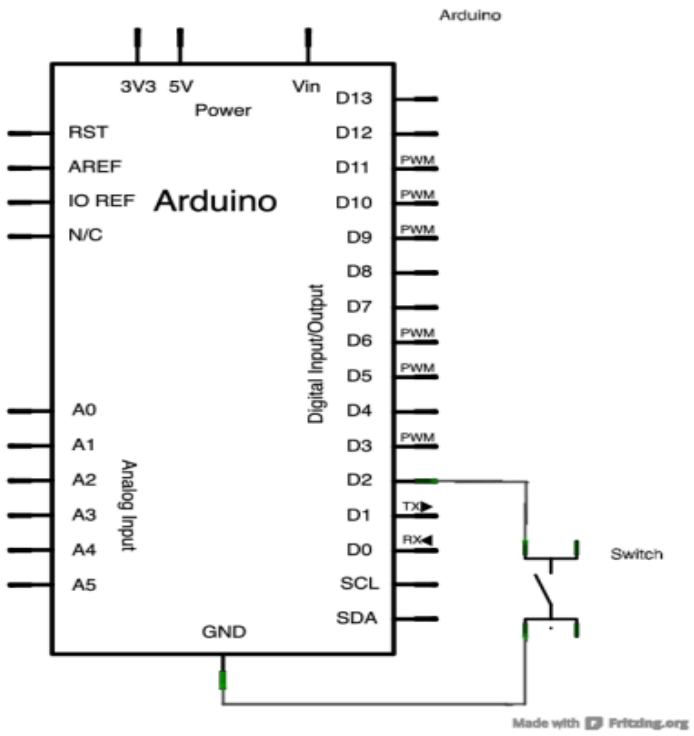


Fig 11: Push button with Arudino

Code:

```
void setup()
{
//start serial connection
Serial.begin(9600);
//configure pin2 as an input and enable the internal pull-up resistor
pinMode(2, INPUT_PULLUP);
pinMode(13, OUTPUT);// connect an LED at pin13.

}

void loop() {
//read the pushbutton value into a variable
int sensorVal = digitalRead(2);
//print out the value of the pushbutton
Serial.println(sensorVal);
// Keep in mind the pullup means the pushbutton's
// logic is inverted. It goes HIGH when it's open,
// and LOW when it's pressed. Turn on pin 13 when the
// button's pressed, and off when it's not:
if (sensorVal == LOW)
{
digitalWrite(13, HIGH);
}
else {
digitalWrite(13, LOW);
```

```
}
}
```
This is how you can control an LED with a push button to turn it ON/OFF.

Basics of MQ-2 gas Sensor:

The MQ-2 gas sensor is a metal oxide semiconductor (MOS) type gas sensor. It is a low-cost, widely used gas sensor that detects multiple gas (but cannot identify them) including Liquefied petroleum gas (LPG), Methane (CH_4), Carbon Monoxide(CO), Alcohol, Smoke (CO_2), Hydrogen(H_2) and Propanein the air concentrations ranging from 200 to 10000 ppm.
The MQ-2 gas sensor works by measuring the change in electrical resistance of a sensing element when it is exposed to a gas. The change in resistance of the sensing element is measured using a simple voltage divider circuit. The output voltage of the voltage divider circuit is proportional to the concentration of the gas present in the air.

Pinout Overview:

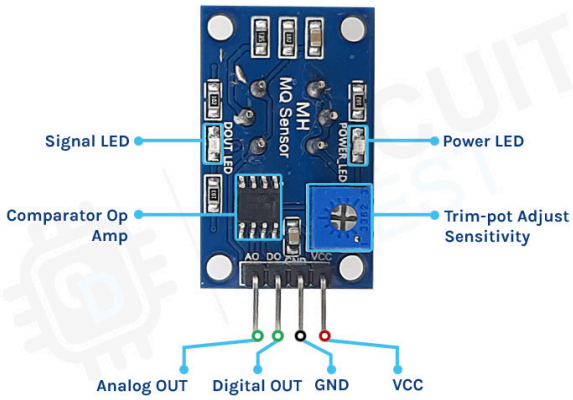


Fig 12: MQ-2 gas Sensor Pinout Diagram

MQ-2 Gas and smoke sensor module has four pins:

VCC supplies power to the module. Connect it to the 5V output of your Arduino.

GND is the ground pin.

D0 indicates the presence of combustible gasses. D0 becomes LOW when the gas concentration exceeds the threshold value (as set by the potentiometer), and HIGH otherwise.

A0 produces an analog output voltage proportional to gas concentration, so a higher concentration results in a higher voltage and a lower concentration results in a lower voltage.

To power the circuit, the 5V pin of the arduino is used because the operating voltage range of this module is 5V with ±0.1% tolerance.

The module also has two onboard LEDs.

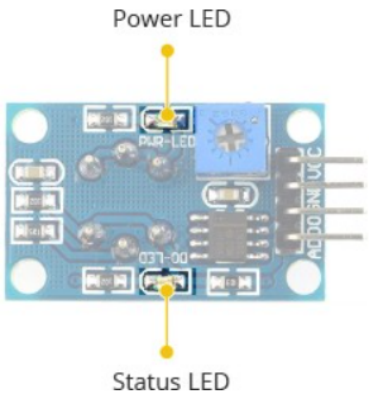


Fig 13: LEDs in MQ-2 gas Sensor

The power LED turns on when power is applied to the board and the Dout LED turns on when the trigger value set by the potentiometer is reached.

This board also has a comparator OP-Amp onboard that is responsible for converting the incoming analog signal from the gas sensor to a digital signal.

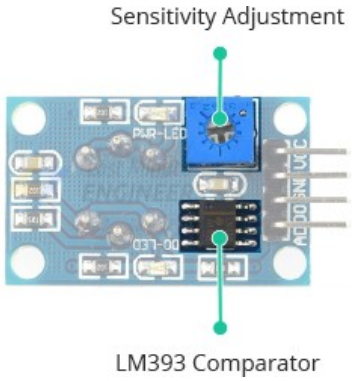


Fig 14: Comparator and Sensitivity Adjustment Potentiometer in MQ-2 gas Sensor

It also has a sensitivity adjustment Trim-pot, with that we can adjust the sensitivity of the device. Finally we have some resistor capacitors for decoupling and filtering.

The MQ-2 gas sensor can be used in two different modes:

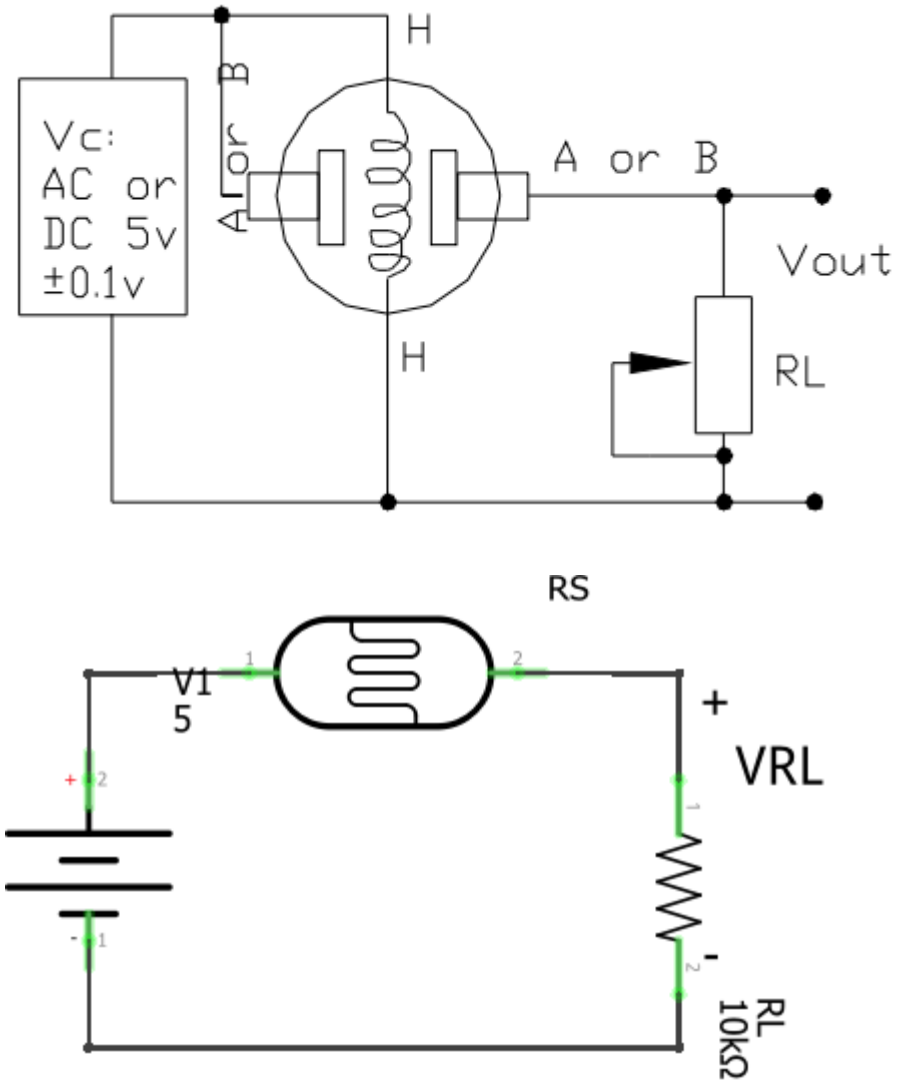
- Analog mode : In analog mode, the sensor output voltage is proportional to the concentration of gas present in the air.
- Digital mode : In digital mode, the sensor output voltage is a high or low signal depending on whether the gas concentration exceeds a certain threshold.

Analog output voltage (at the A0 pin) varies in proportion to the concentration of smoke/gas. The higher the concentration, the higher the output voltage; the lower the concentration, the lower the output voltage. In clean air, the output voltage is 0v. In the presence of smoke, the output voltage > 0v. If the concentration is ≥10000 ppm, the output voltage is 5v. To use the MQ-2 gas sensor in analog mode, the A0 pin is connected to an analog input pin of a microcontroller. The microcontroller can then read the analog voltage on the A0 pin to determine the concentration of gas present in the air.

The analog signal is digitized by an LM393 High Precision Comparator and made available at the Digital Output (D0) pin. A potentiometer is used for adjusting threshold and the sensitivity of the digital output (D0). When the gas concentration exceeds the threshold value, the module outputs LOW otherwise HIGH. To use the MQ-2 gas sensor in digital mode, the D0 pin is connected to a digital input pin of a microcontroller. The microcontroller can then read the digital signal on the D0 pin to determine whether the gas concentration exceeds a certain threshold.

Pre-Heat Time for MQ-2 gas sensor:

While working with this type of gas sensor, there is a pre heat time or stabilization time that is required for this device to work properly. Given the small size of the sensor, thermal equilibrium will be almost surely reached **within 1.00 hr.**



<u>Fig 15:</u> Simplified Internal diagram of MQ-2 gas Sensor

There is a heating element that is responsible for maintaining a constant temperature across the ceramic sensing element (H). This heating element is usually a small electrically powered resistor.

Add the appropriate delay in the Arduino code to the void setup section after powering the MQ-2 gas sensor to allow for preheating.

<u>What is ppm for gas?</u>

Parts-per-million, or “ppm”, is commonly used as a fractional unit of measure for concentration. As an example, a methane (molecular) concentration of 2% means that 2 out of every 100 air molecules is methane. Similarly, a methane concentration of 2 ppm means that 2 out of every 1 million air molecules is methane.

Procedure:

Step 1: Preheat the gas sensor for at least 30 mins for precise measurement.

Step 2: Connect the Gas Sensor:

a. Connect the MQ-2 gas sensor to the breadboard.
b. Connect the VCC pin of the gas sensor to the 5V pin of the Arduino.
c. Connect the GND pin of the gas sensor to the GND pin of the Arduino.
d. Connect the D_0 pin of the gas sensor to digital **pin 2** of the Arduino.
e. Connect the A_0 pin of the gas sensor to Analog pin, A_0 of the Arduino.

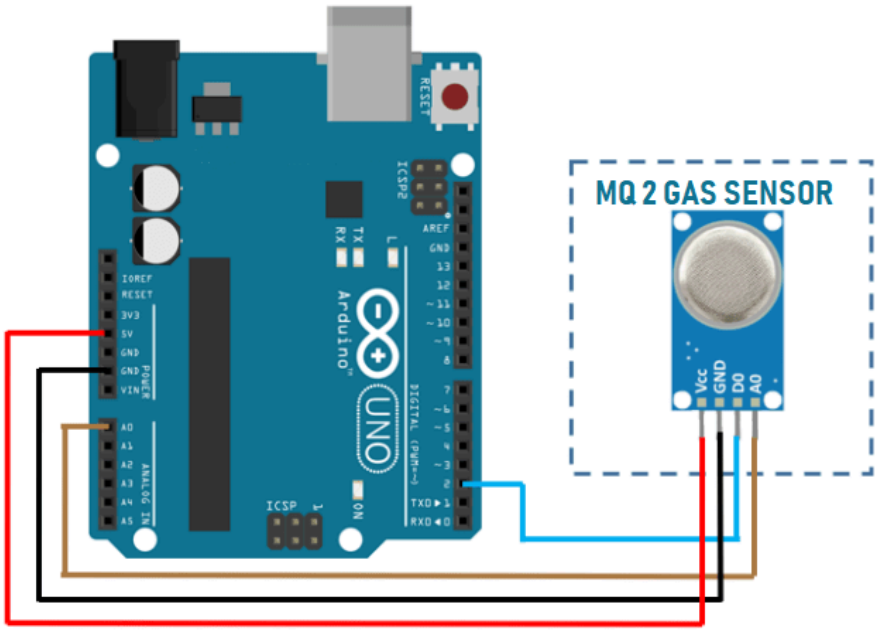


Fig 16: MQ-2 gas sensor and arduino connection diagram.

Step 3: Connect the OLED Display:

a. Connect the OLED display to the breadboard.
b. Connect the VCC pin of the OLED display to the 5V pin of the Arduino.
c. Connect the GND pin of the OLED display to the GND pin of the Arduino.
d. Connect the **SDA** pin of the OLED display to **analog pin 4** of the Arduino.
e. Connect the **SCL** pin of the OLED display to **analog pin 5** of the Arduino.

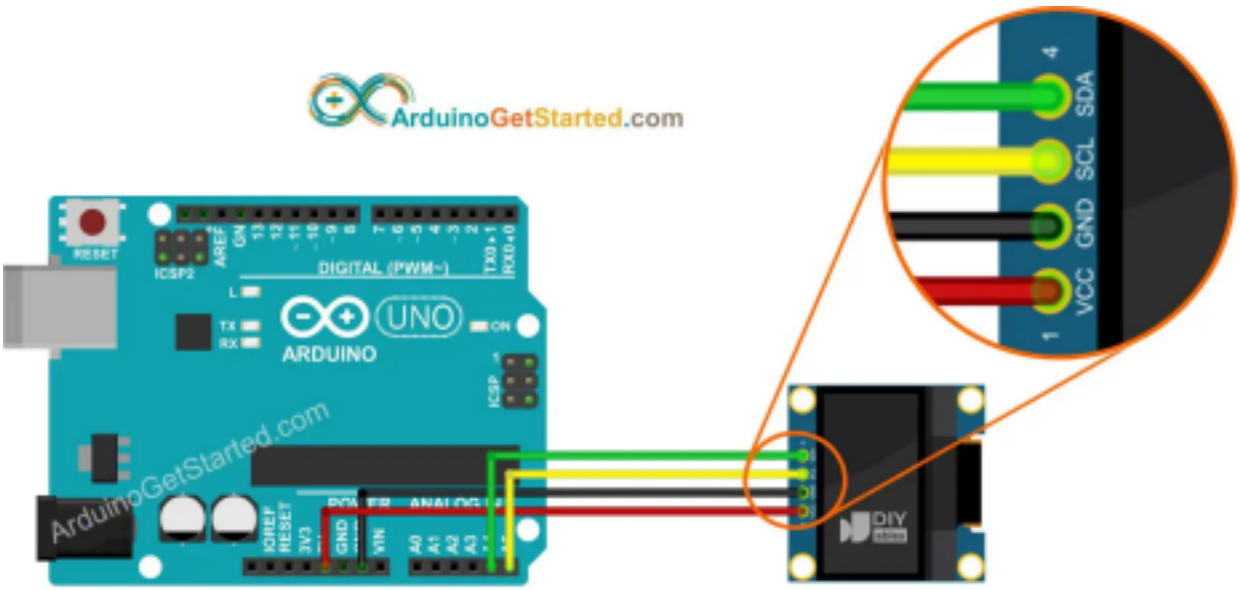


Fig 17: OLED and arduino connection diagram.

Step 4: Connect the 10K ohm Resistor:

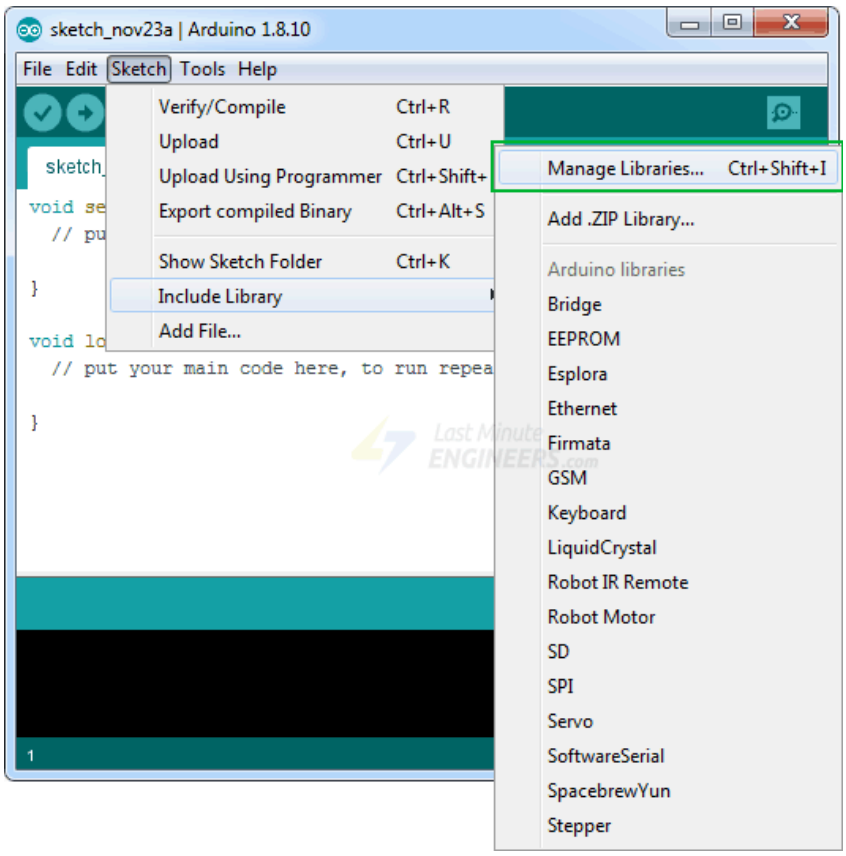
Resistors limit the current flowing through the LED by creating a voltage drop across them. To protect the LED from excessive current flow which can damage the LED.

a. Connect one leg of the 10K ohm resistor to the DO pin of the gas sensor
b. Connect the other leg of the resistor to the VCC pin of the gas sensor

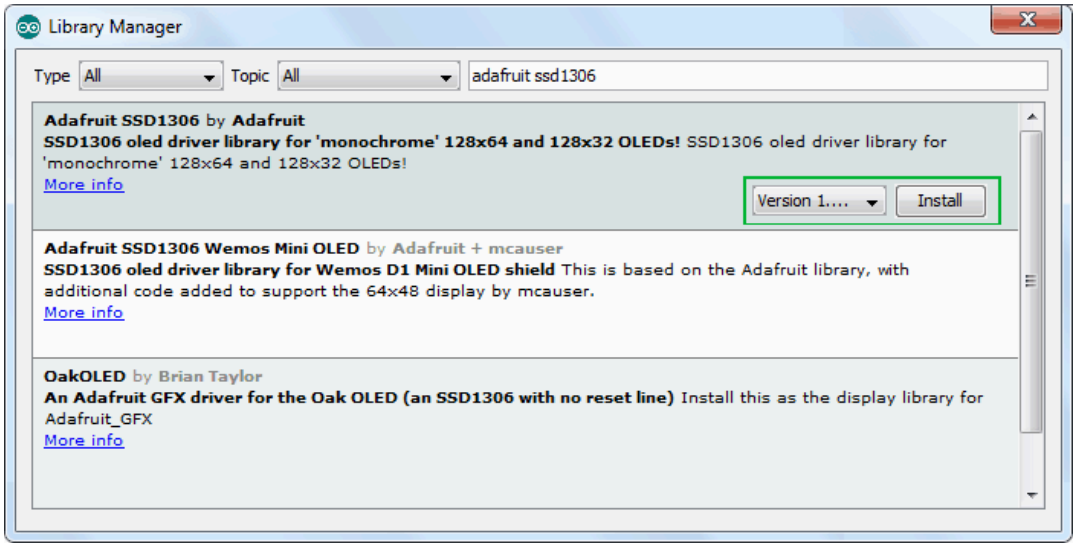
Step 5: Upload the Code:

a. Open the Arduino IDE and create a new sketch
b. Include Adafruit_SSD1306.h , Adafruit_GFX.h and busIO libraries into the Arduino IDE by following the instructions:

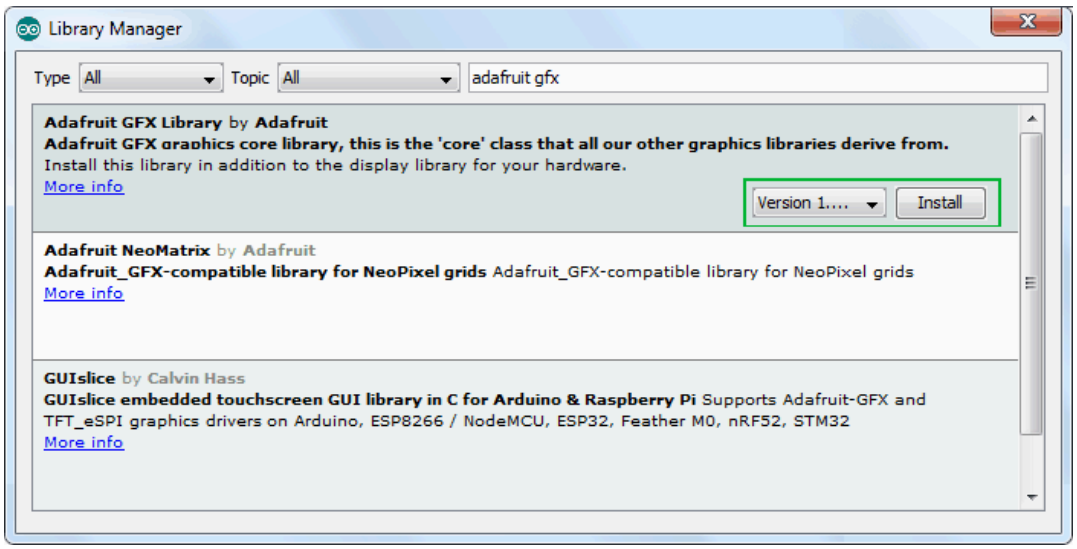
i.


sketch_nov23a | Arduino 1.8.10
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
Keyboard
LiquidCrystal
Robot IR Remote
Robot Motor
SD
SPI
Servo
SoftwareSerial
SpacebrewYun
Stepper
// put your main code here, to run repea

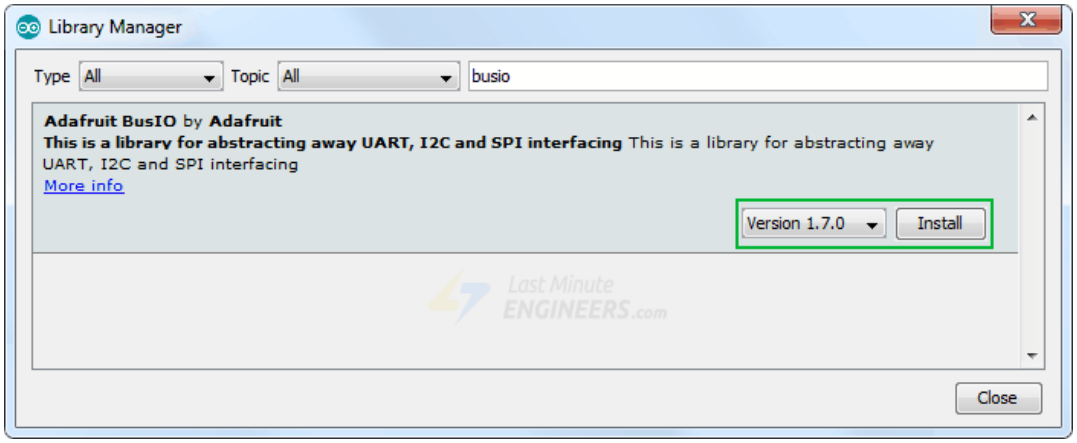
ii.


Library Manager
Type All Topic All adafruit ssd1306
Adafruit SSD1306 by Adafruit
SSD1306 oled driver library for 'monochrome' 128x64 and 128x32 OLEDs! SSD1306 oled driver library for 'monochrome' 128x64 and 128x32 OLEDs!
More info
Version 1.... Install
Adafruit SSD1306 Wemos Mini OLED by Adafruit + mcauser
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to support the 64x48 display by mcauser.
More info
OakOLED by Brian Taylor
An Adafruit GFX driver for the Oak OLED (an SSD1306 with no reset line) Install this as the display library for Adafruit_GFX
More info

iii.



iv.



c. Copy and paste the following codes one by one in the IDE sketch and see what results are shown:

Code:

```
// Pin connected to the MQ-2 sensor

const int sensorPin = A0;

// Define the load resistance value (in ohms) used in the circuit

#define RL 10  //Load resistance

#define m -0.263 //Calculated Slope

#define b 0.42 //Calculated intercept
```

```
#define Ro 20 // Resistance on fresh air



void setup() {

  Serial.begin(9600);

  Serial.println("MQ2 warming up!");

  delay(20000); // allow the MQ2 to warm up

  pinMode(sensorPin, INPUT);

  pinMode(Buzz, OUTPUT);

  pinMode(relay, OUTPUT);

}

void loop() {

float VRL; //Voltage drop across the MQ sensor
float Rs; //Sensor resistance at gas concentration
float ratio; //Define variable for ratio

float sensorValue = analogRead(sensorPin);

Serial.println(sensorValue);


VRL = sensorValue * (5.0/1023.0); //Measure the voltage drop and convert to 0-5V


Rs = ((5.0*RL)/VRL)-RL; //Use formula to get Rs value


ratio = Rs/Ro;  // find ratio Rs/Ro

float ppm = pow(10, ((log10(ratio)-b)/m)); //use formula to calculate ppm



  Serial.print("PPM: ");
```

```
  Serial.println(ppm);

}
```

Verify and upload the code to the Arduino board. The above code performs the following:

1. Reads Analog Data from the MQ-2 Sensor and prints the analog data in the serial monitor of the Arduino IDE.
2. Converts the Analog data to voltage (0-5V) in Arduino and prints it on the serial monitor.
3. Converts the voltage level from the MQ-2 sensor to the PPM value and prints it on the serial monitor.
4. Also, the ppm data can be serial plotted in the Arduino by following these instructions:

   Arduino IDE > Tools > Serial Plotter

So far, we worked with a gas sensor only. Now let's look at the OLED Display. We will show you how to display simple texts on the OLED Display. You have already connected the OLED Display to Arduino. The OLED Display we are using in this lab works as I2C Devices (Master-slave devices, you will learn about such devices in the theory classes). So, each OLED Display has a screen address which we first need to find out. After connecting the display you need to scan and set the I2C address of the OLED display. You can find the I2C address of the display by following these steps:

- Navigate to Arduino IDE > File > Examples > Wire > i2c_scanner
- Upload and check the Serial monitor for the I2C Address.

Now, upload the following codes which will simply display "Hello World" on the OLED Display:

```
#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN_ADDRESS 0×3C ///< See datasheet for Address; 0×3D for 128×64, 0×3C for 128×32

Adafruit_SSD1306 display(128, 64, &Wire, OLED_RESET);

void setup() {

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {

    Serial.println(F("SSD1306 allocation failed"));
```

```
    for (;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.display();
//display.clearDisplay();
}
void loop() {
 display.clearDisplay();
 display.setCursor(0, 12);
  display.println("Hello World!");
 display.display();
delay(100);
}
```

- **Change 0x3C with your I2C address #define SCREEN_ADDRESS 0x3C**

Lab Tasks:

1) Display the Gas Sensor data you measured using MQ-2 sensor on the OLED like this:

   "Gas sensor data: ppm value".

2) Turn on a buzzer if your gas sensor detects ppm above a certain threshold using Arduino.
3) Setup a gas sensing system so that if it detects gas, a message - "Gas is detected" is shown on the OLED screen.

Conclusion:

In this lab,we learned how to operate an Arduino board, how to upload code and run a system.Also,we learned how to interface a gas sensor with an Arduino and display the sensor data on an OLED display. This can be used to detect the presence of gas in the environment and display it in real-time. This system can be useful in industrial applications, homes, and offices to ensure safety in the presence of gas.

Tasks for Lab Report:

1. Use a push button to toggle an LED using Proteus.
2. **LDR** sensors can sense the change of intensity of light.Suppose,you have an LDR sensor and an LED . Interface an LDR sensor with Arduino in Proteus and turn on the LED when the intensity of light is low and turn it off when the intensity is high. Use the link given in references to understand how the LDR sensor works.
3. Imagine you are doing a project on obstacle avoidance.You want to measure the distance of an object from your device. Interface a **SONAR s**ensor with Arduino and measure distance using it.Show the distance in meter.Use the link given in references to understand how sonar sensor works. Also, download the ultrasonic library for proteus.
4. A gas sensing based system should be implemented such that if gas is detected, an alarm/buzzer is triggered instantly to notify the incident.
5. A gas sensing based system should be implemented such that if gas is detected, a motorized exhaust fan is triggered to keep out the fumes.

Reference links:

1) In-Depth: How MQ2 Gas/Smoke Sensor Works? & Interface it with Arduino (lastminuteengineers.com)
2) calculation of ppm value using mq sensor and arduino - Using Arduino / Project Guidance - Arduino Forum
3) Push button interfacing with arduino
4) In-Depth: Interface OLED Graphic Display Module with Arduino (lastminuteengineers.com)
5) Ultrasonic sensor interfacing with Arduino & Ultrasonic library for proteus
6) LDR interfacing with Arduino
7) MQ sensor Datasheet