

IOT WORKSHOP

22SC1209

A.Y: 2022 –23

Semester: 1-2



(DEEMED TO BE **U N I V E R S I T Y**)

DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING

Vaddeswaram, Guntur District

Index of Experiments

S. No.	Name of the Experiment	Page No.	Marks	Faculty Sign
A	Introduction of Arduino UNO, ESP32 development board and Basic Electronics Component			
B	Arduino IDE Software Installation and Programming Concepts			
1	LED interfacing With Arduino			
2	Controlling LED Interface With Arduino using PUSH button			
3	Automatic Street Light Controlling Using Arduino			
4	Smoke Detection Using MQ-2 GAS Sensor			
5	Interfacing PIR Sensor with Arduino			
6	Interfacing Ultrasound Sensor with Arduino			
7	Interfacing Temperature Sensor			
8	Interfacing Relay to Arduino			
9	Interfacing Servo Motor			
10	LCD Display Interfacing with Arduino			
11	ESP 32 SETUP & Controlling the GPIO pins in ESP32			
12	Controlling the GPIO pins in ESP32 using Touch Inputs			
13	Interfacing Servo motor to ESP32			

2022-23 EVEN SEMESTER LAB CONTINUOUS EVALUATION

S No:	Date	Experiment Title	Pre-Lab (5M)	In Lab				Post-Lab (5M)	Viva-voce (5M)	Total (50M)	Sign of Faculty
				Logic (10M)	Execution (10M)	Result (10M)	Analysis (5M)				
1											
2											
3											
4											
5											
6											
7											

2022-23 EVEN SEMESTER PROJECT CONTINUOUS EVALUATION											
S No:	Date	Project Title	Abst ract (5M)	Project Work				Rep ort (5M)	Viva- voce (5M)	Total (50M)	Sign of Faculty
				Logic (10M)	Execut ion (10M)	Result (10M)	Analysis (5M)				

2022-23 EVEN SEMESTER LAB CONTINUOUS EVALUATION

S No:	Date	Experiment Title	Pre-Lab (5M)	In Lab				Post-Lab (5M)	Viva-voce (5M)	Total (50M)	Sign of Faculty
				Logic (10M)	Execution (10M)	Result (10M)	Analysis (5M)				
8											
9											
10											
11											
12											
13											
7											

2022-23 EVEN SEMESTER PROJECT CONTINUOUS EVALUATION									
S No:	Date	Project Title	Abstract (5M)	Project Work				Report (5M)	Viva-voce (5M)
				Logic (10M)	Execution (10M)	Result (10M)	Analysis (5M)	Total (50M)	Sign of Faculty

Expt No: A

Date:

Introduction of Arduino UNO, ESP32 development board and Basic Electronics Component

Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.



Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

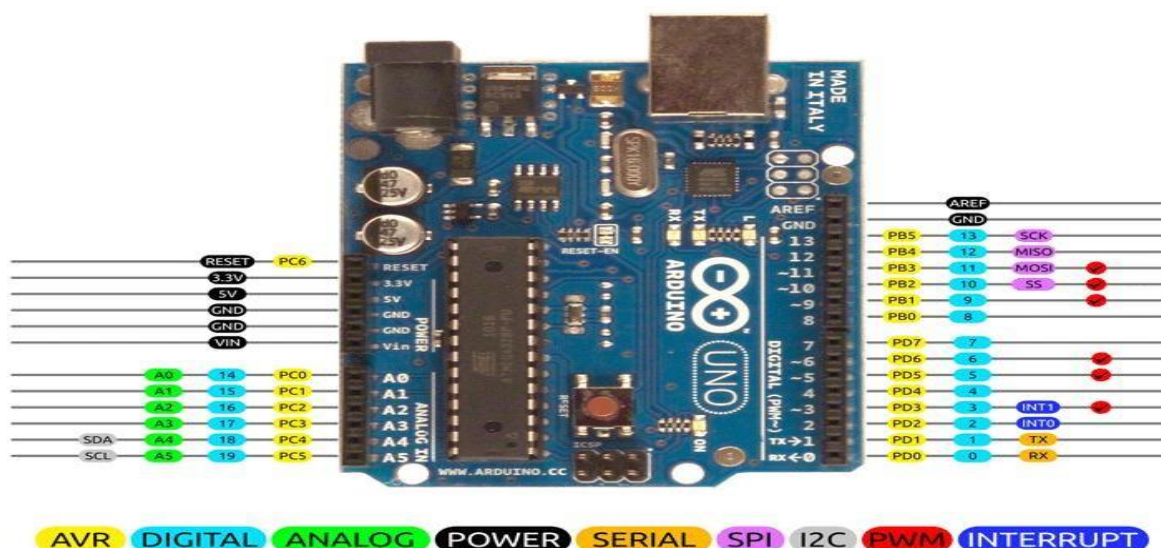
Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without

Microcontroller	<u>ATmega328P</u>
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

worrying too much about doing something wrong, worst case scenario you can replace the chip for a few rupees and start over again.



THE BOARD

Power connector

This is how you power your Arduino when it's not plugged into a USB port for power. Can accept voltages between 7-12V.

USB port

Used for powering your Arduino Uno, uploading your sketches to your Arduino, and for communicating with your Arduino sketch (via Serial.println() etc.)

Reset Button

Resets the ATmega microcontroller.

TX and RX LEDs

These LEDs indicate communication between your Arduino and your computer. Expect them to flicker rapidly during sketch upload as well as during serial communication. Useful for debugging.

Digital pins

Use these pins with digitalRead(), digitalWrite(), and analogWrite(). analogWrite() works only on the pins with the PWM symbol.

Pin 13 LED

The only actuator built-in to your Arduino Uno. Besides being a handy target for your first blink sketch, this LED is very useful for debugging.

GND and 5V pins

Use these pins to provide +5V power and ground to your circuits.

Analog in

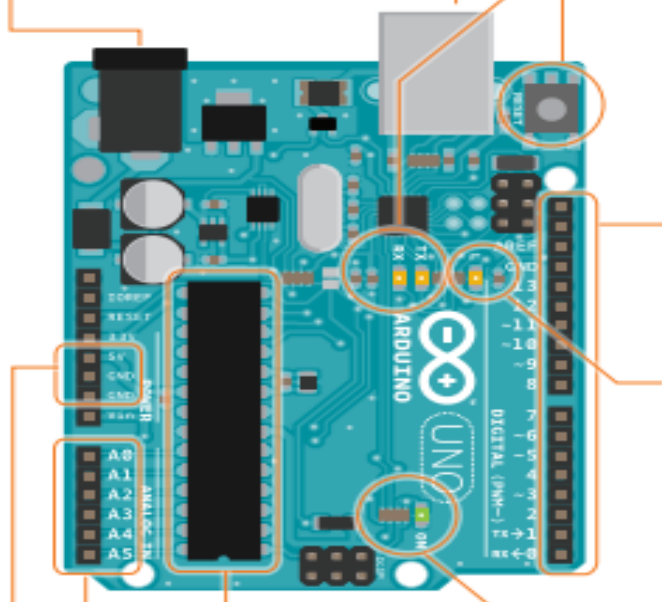
Use these pins with analogRead().

ATmega microcontroller

The heart of your Arduino Uno.

Power LED

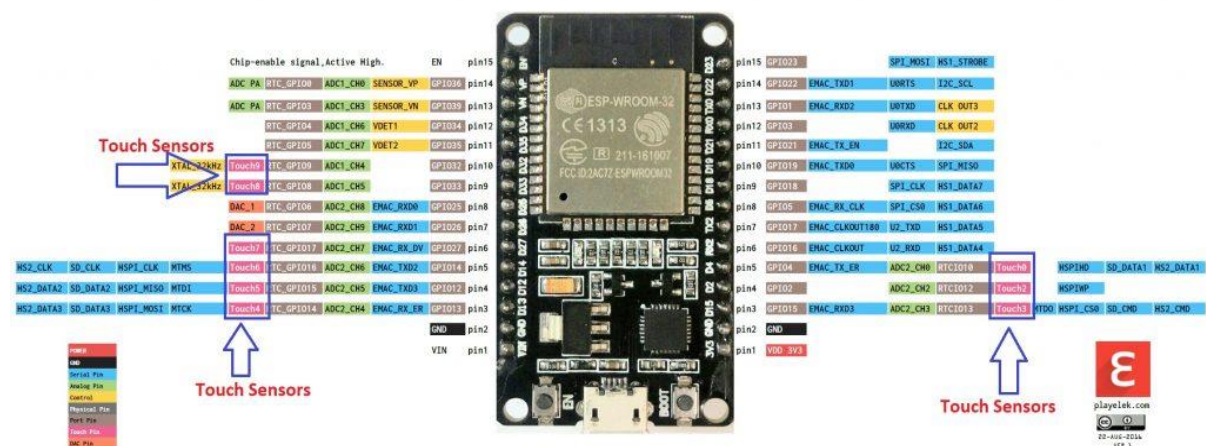
Indicates that your Arduino is receiving power. Useful for debugging.



INTRODUCTION to ESP32 development board

ESP32 is in the series of low power and low cost on chip microcontroller. It comes up with already integrated dual mode Bluetooth and Wi-Fi. It is especially aimed to provide versatility, robustness and reliability in a large number of applications. Some applications in which this microcontroller is extensively used are MP3 decoding, voice encoding and music streaming. Best RF and power performance can be easily achieved using this microcontroller. ESP32 comes up with a USB port so we can say that it is a plug and play device i.e. just plug in cable and your device is turned on and you are able to program it just like Arduino development boards.

DOIT ESP32 DEVKIT V1 PINOUT



COMPARISON OF ESP32 WITH ARDUINO

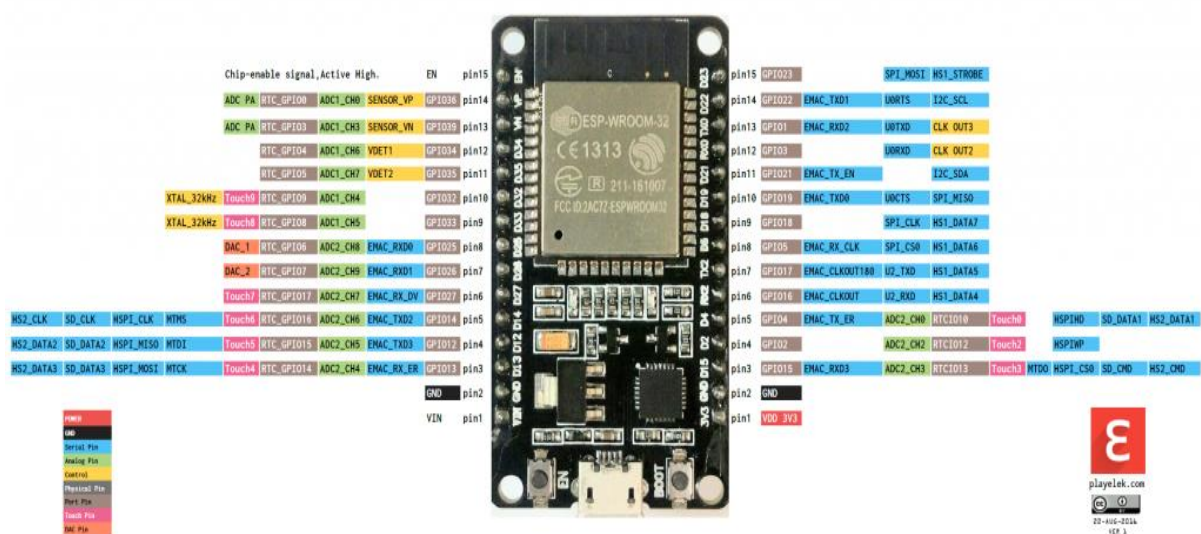
If we compare this microcontroller with Arduino boards then we have a big advantage of this microcontroller over Arduino development boards and that is of Wi-Fi. Although Wi-Fi can be used with many Arduino boards but this feature comes up in form of shields / adapters. If we attach Wi-Fi shield with our Arduino board then we can access internet in our Arduino board otherwise it's not possible. ESP32 comes up with integrated Wi-Fi module. Same is the case with Bluetooth which is available in Arduino in form of some modules and when it comes to ESP32, this feature is already integrated. So if we are interested in using Wi-Fi and Bluetooth then this board is less expensive as compared to Arduino boards as these adapters for Wi-Fi and Bluetooth are expensive.

KEY FEATURES OF ESP32

- Built in Wi-Fi module of standard 802.11
- Wi-Fi module operate in range of 2.4 GHz – 2.5 GHz
- Three modes of operation: 1. Access point. 2. Client. 3. Access point + station
- Dual core microprocessor of 32 – bit
- Operating voltage is 3.3 V
- Clock frequency from 80 MHz and goes up to 240 MHz
- SRAM memory is of 512 KB
- ROM memory is 448 KB
- External flash memory is supported and is up to 32 Mb i.e. 4MB
- Maximum current in each pin is 12 mA but 6 mA is recommended to use
- It has 36 general purpose input / output pins
- General purpose input / output pins comes up with PWM / I2C and SPI functionality
- Bluetooth version 4.2 is available and Bluetooth low energy (BLE) version
- 2 to 3.6 V operating voltage
- Deep sleep current of 2.5 μ A
- 10 electrode capacitive touch support
- Hardware supported encryption for AES, ECC, RSA – 4096, SHA2
- On board PCB antenna or IPEX connector which act as external antenna
- Operating temperature lies in range of -40°C to +125°C

PINS SPECIFICATIONS OF ESP32

DOIT ESP32 DEVKIT V1 PINOUT



ESP32 supports pin multiplexing feature i.e. we can decide that from 28 broken input / output pins which peripherals can be connected. It simply means that it's up to us that we have to decide for these pins and they can act as MISO, RX, SCLK, MOSI, TX, SCL, SDA and many more. However pins for analog to digital convertor and digital to analog convertor are static

- Analog to digital convertor (ADC) channels are 18
- SPI interfaces are 3
- UART interfaces are 3
- I2C interfaces are 2
- PWM outputs are 16
- Digital to analog convertors (DAC) are 2
- I2S interfaces are 2
-

SENSORS ON ESP32

- Hall sensors
- 32 kHz crystal oscillator
- Ultra low noise analog amplifier
- 10x capacitive touch interfaces

POWERING ESP32 DEVELOPMENT BOARD

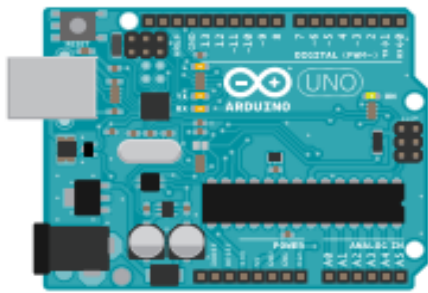
In order to turn on your ESP32 development board we can either make use of USB port or LiPo battery. If we use these both devices in our board then the charge controller that is already present on board will charge our LiPo battery. There is also a 3.3 V voltage regulator present on board which will deliver current of 600 mA. During the RF transmission our board can pull up as much as 250 mA. Our general purpose input output pins cannot tolerate 5 V. So if we need to interface our board with 5 V then we have to do level shifting.

PROGRAMMING OF ESP32

Arduino IDE can be used for programming of this board. But we have to install some drivers and libraries in order to make it compatible in Arduino IDE and ready to use just like we use other Arduino boards in Arduino IDE. So a brief explanation of drivers and libraries that are needed is given below.



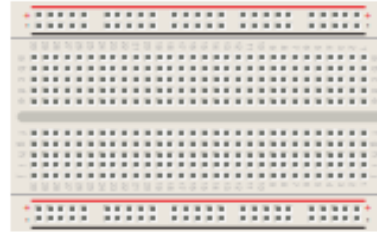
PARTS IN YOUR KIT



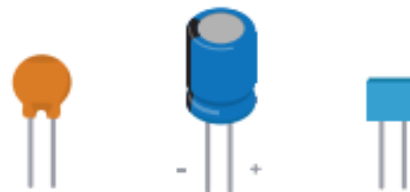
Arduino Uno - The microcontroller development board that will be at the heart of your projects. It's a simple computer, but one that has no way for you to interact with it yet. You will be building the circuits and interfaces for interaction, and telling the microcontroller how to interface with other components.



Battery Snap - Used to connect a 9V battery to power leads that can be easily plugged into a breadboard or your Arduino.



Breadboard - A board on which you can build electronic circuits. It's like a patch panel, with rows of holes that allow you to connect wires and components together. Versions that require soldering are available, as well as the solder-less type used here.



Capacitors - These components store and release electrical energy in a circuit. When the circuit's voltage is higher than what is stored in the capacitor, it allows current to flow in, giving the capacitor a charge. When the circuit's voltage is lower, the stored charge is released. Often placed across power and ground close to a sensor or motor to help smooth fluctuations in voltage.



DC motor - Converts electrical energy into mechanical energy when electricity is applied to its leads. Coils of wire inside the motor become magnetized when current flows through them.

These magnetic fields attract and repel magnets, causing the shaft to spin. If the direction of the electricity is reversed, the motor will spin in the opposite direction.



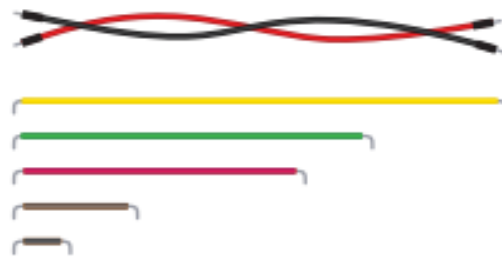
Diode - Ensures electricity only flows in one direction. Useful when you have a motor or other high current/voltage load in your circuit. Diodes are polarized, meaning that the direction that they're placed in a circuit matters. Placed one way, they allow current to pass through. Placed the other way, they block it. The anode side generally connects to the point of higher energy in your circuit. The cathode typically connects to the point of lower energy, or to ground. The cathode is usually marked with a band on one side of the component's body.



Gels (red, green, blue) - These filter out different wavelengths of light. When used in conjunction with photoresistors, they cause the sensor to only react to the amount of light in the filtered color.



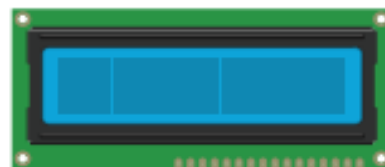
H-bridge - A circuit that allows you to control the polarity of the voltage applied to a load, usually a motor. The H-bridge in the kit is an integrated circuit, but it could also be constructed with a number of discrete components.



Jumper wires - Use these to connect components to each other on the breadboard, and to the Arduino.



Light Emitting Diodes (LEDs) - A type of diode that illuminates when electricity passes through it. Like all diodes, electricity only flows in one direction through these components. You're probably familiar with these as indicators on a variety of electronic devices. The anode, which typically connects to power, is usually the longer leg, and the cathode is the shorter leg.



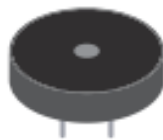
Liquid Crystal Display (LCD) - A type of alphanumeric or graphic display based on liquid crystals. LCDs are available in a many sizes, shapes, and styles. Yours has 2 rows with 16 characters each.



Male header pins - These pins fit into female sockets, like those on a breadboard. They help make connecting things much easier.



Optocoupler - This allows you to connect two circuits that do not share a common power supply. Internally there is a small LED that, when illuminated, causes a photoreceptor inside to close an internal switch. When you apply voltage to the + pin, the LED lights and the internal switch closes. The two outputs replace a switch in the second circuit.



Piezo - An electrical component that can be used to detect vibrations and create noises.



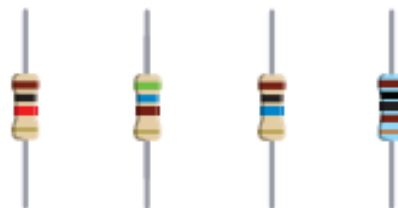
Photoresistor - (also called a photocell, or light-dependent resistor). A variable resistor that changes its resistance based on the amount of light that falls on its face.



Potentiometer - A variable resistor with three pins. Two of the pins are connected to the ends of a fixed resistor. The middle pin, or wiper, moves across the resistor, dividing it into two halves. When the external sides of the potentiometer are connected to voltage and ground, the middle leg will give the difference in voltage as you turn the knob. Often referred to as a pot.



Pushbuttons - Momentary switches that close a circuit when pressed. They snap into breadboards easily. These are good for detecting on/off signals.



Resistors - Resist the flow of electrical energy in a circuit, changing the voltage and current as a result. Resistor values are measured in ohms (represented by the Greek omega character: Ω). The colored stripes on the sides of resistors indicate their value (see resistor color code table).



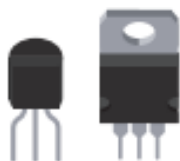
Servo motor - A type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your Arduino. These pulses tell the motor what position it should move to.



Temperature sensor - Changes its voltage output depending on the temperature of the component. The outside legs connect to power and ground. The voltage on the center pin changes as it gets warmer or cooler.



Tilt sensor - A type of switch that will open or close depending on its orientation. Typically they are hollow cylinders with a metal ball inside that will make a connection across two leads when tilted in the proper direction.

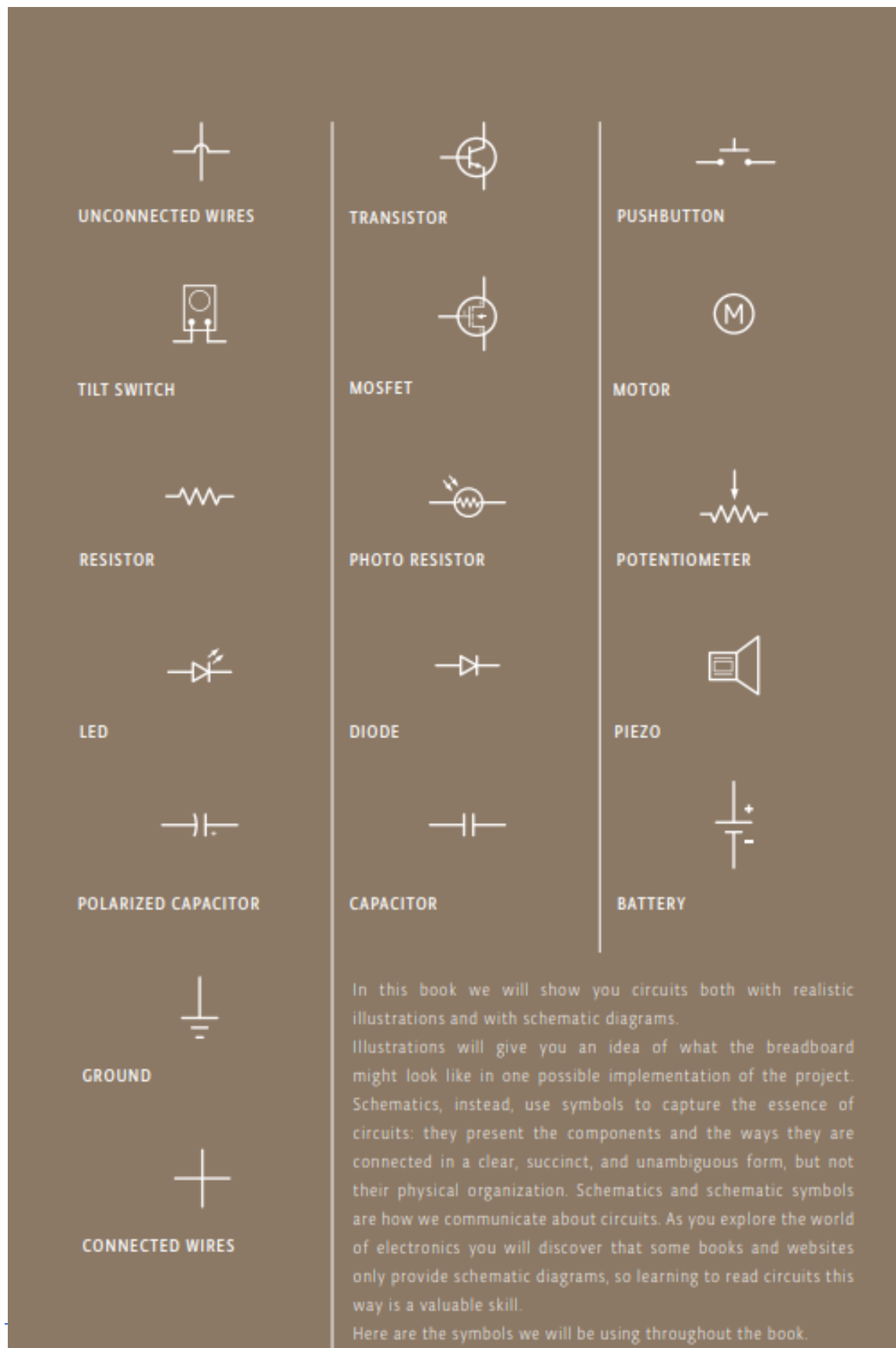


Transistor - A three legged device that can operate as an electronic switch. Useful for control-

ling high current/high voltage components like motors. One pin connects to ground, another to the component being controlled, and the third connects to the Arduino. When the component receives voltage on the pin connected to an Arduino, it closes the circuit between the ground and the other component.



USB Cable - This allows you to connect your Arduino Uno to your personal computer for programming. It also provides power to the Arduino for most of the projects in the kit.



YOUR FIRST COMPONENTS



An **LED**, or light-emitting diode, is a component that converts electrical energy into light energy. LEDs are polarized components, which means they only allow electricity to flow through them in one direction. The longer leg on the LED is called an anode, it will connect to power. The shorter leg is a cathode and will connect to ground. When voltage is applied to the anode of the LED, and the cathode is connected to ground, the LED emits light.



A **resistor** is a component that resists the flow of electrical energy (see the components list for an explanation on the colored stripes on the side). It converts some of the electrical energy into heat. If you put a resistor in series with a component like an LED, the resistor will use up some of the electrical energy and the LED will receive less energy as a result. This allows you to supply components with the amount of energy they need. You use a resistor in series with the LED to keep it from receiving too much voltage. Without the resistor, the LED would be brighter for a few moments, but quickly burn out.

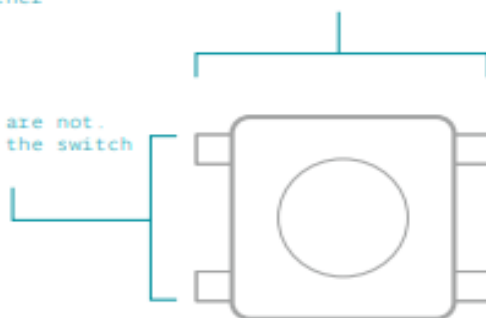


A **switch** interrupts the flow of electricity, breaking the circuit when open. When a switch is closed, it will complete a circuit. There are many types of switches. The ones in your kit are called momentary switches, or pushbuttons, because they are only closed when pressure is applied.

SWITCH CONNECTIONS

These two pins of a switch are connected to each other

These two are not. They form the switch



The switch
Fig. 7

SWITCH SCHEMATIC VIEW



A - Toggle switch symbol



B - Pushbutton symbol

Expt No: B**Date:****Arduino IDE Software Installation and Programming Concepts****Setup & Installation****A. Arduino IDE (For Windows)**

This part will guide you through the set up and installation process of the Integrated Development Environment (IDE) that will be used throughout the exercises.

1. Open your default internet browser and access the Arduino website. Download the latest Arduino IDE version. The software is compatible with Linux, Mac and Windows so just choose the one that matches your OS. The Arduino download page is at <http://arduino.cc/en/Main/Software> [2].



Figure 2.1 A part of Arduino Website's download page. The current version at this time was 0022. Arduino allows you to install its IDE on several platforms (see encircled)

2. After downloading the compressed file, extract its contents to your preferred directory (*C:\Program Files*, your *Desktop* or *etc...*). Note that the whole folder size is around 200MB when completely extracted [2].

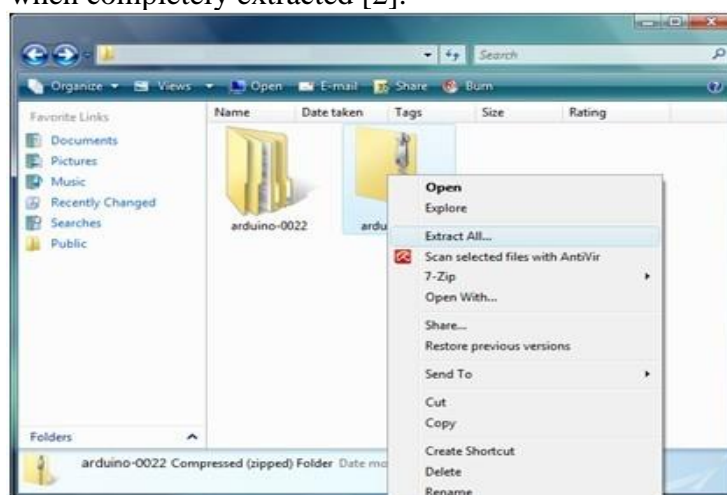


Figure 2.2 Screenshot of attempt to extract the zipped Arduino folder. Make sure you have an archive utility such as 7zip or WinRAR.

3. Congratulations! Arduino IDE is installed on your computer. To use it, just navigate to your main folder directory and run the **Arduino** application [2].

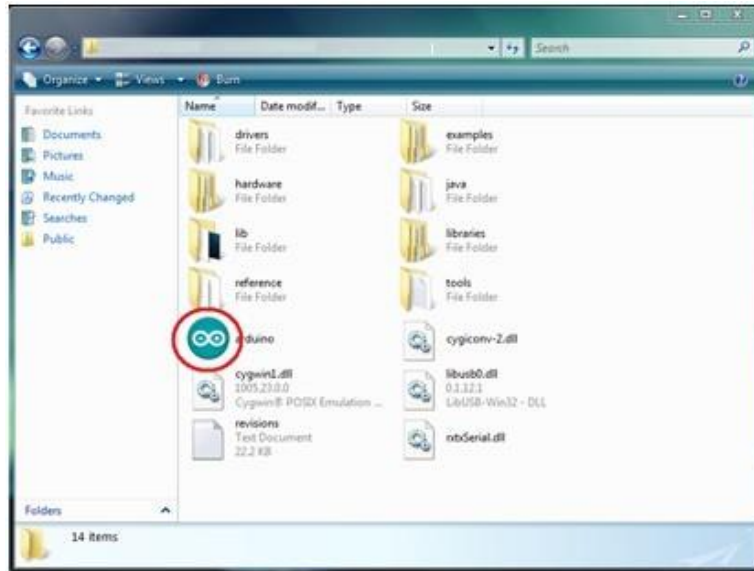


Figure 2.3 Screenshot of what's inside the *Arduino-0022* folder. The application icon looks like an infinity symbol.

B. Arduino Uno board driver (For Windows)

This part will guide you through the set-up and installation process of the Arduino Uno board driver for the device to be recognized by the IDE.

1. Connect the Arduino UNO to the computer via USB Cable (A on fig. 4) [2]. Check if it is properly connected by observing the green LED labeled ON (B on fig. 4) on the board.



Figure 2.4 Photo of Arduino Uno board connected to a Computer. Note that the board's USB-B port.

2. Wait for Windows to try and install the device's driver until it fails. Navigate to the **Device Manager** through **Start > Control Panel > Device Manager**. Locate the Arduino Uno Device. Right-click it to choose **Update Driver Software** [2].

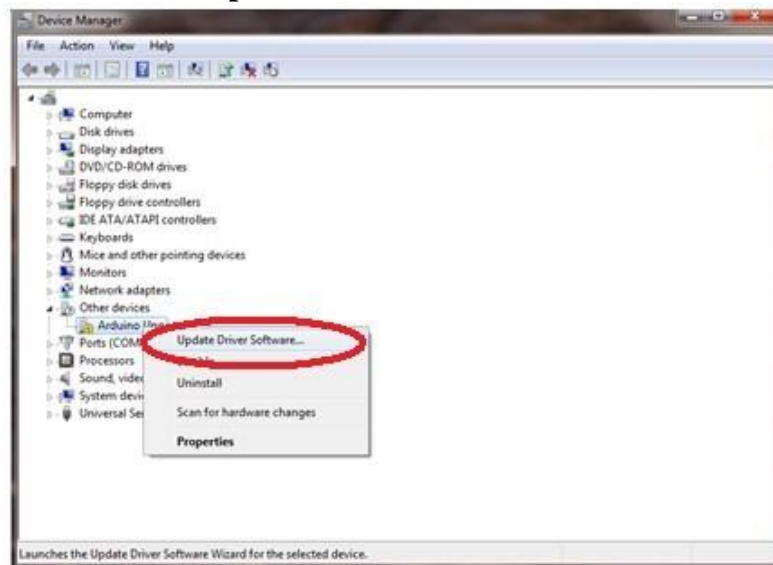


Figure 2.5 Screenshot of the Device Manager. The Arduino Uno should have an exclamation point.

3. Choose to browse your computer for the driver by clicking **Browse my computer for driver software** [2].

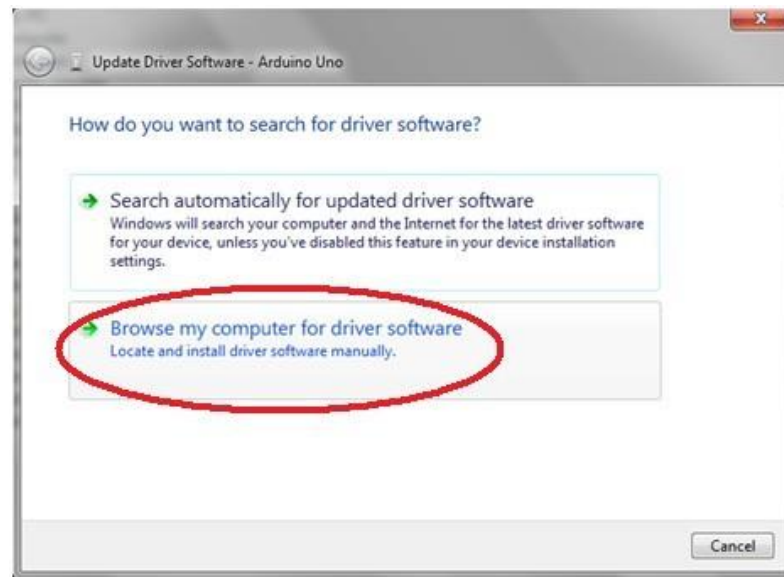


Figure 2.6 Screenshot of the options for searching the device driver. Choose the second option so that you can look for it in your hard disk.

4. A new window will open for you to indicate the location of the driver. Click **Browse...**



Figure 2.7 Screenshot of the browse option menu. Choose the first option which is to look manually for the folder that contains the Arduino Uno board's driver.

5. Navigate to your Arduino folder and choose the driversfolder. Click **OK** upon selection [2].

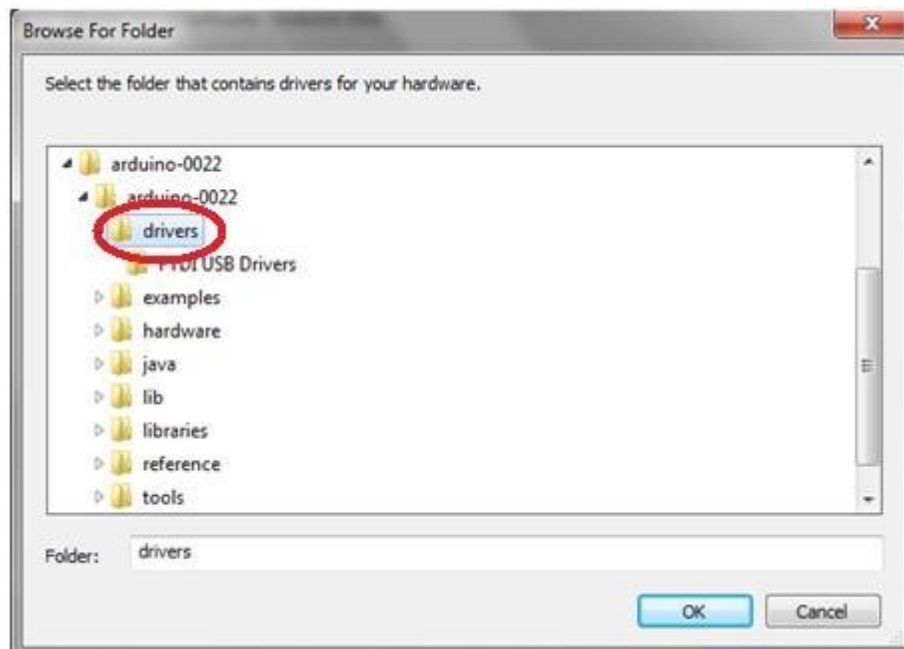


Figure 2.8 Screenshot of navigating through the Arduino software folder. Note that the **drivers** folder was chosen rather than the FTDI USB Drivers (It was mentioned earlier that only preceding models use this)

6. A Windows Security window sometimes pops up to confirm if you want to continue the installation. Just click, **Install this driver software anyway**.



Figure 2.9 Screenshot of pop-up window. Windows can't verify the publisher of the device software but we know that the software's publisher is Arduino.

7. Wait for Windows to finish installing the device driver. Upon completion, you should see an installation successful message. Congratulations and click **Close**. You are ready to

start programming using Arduino!

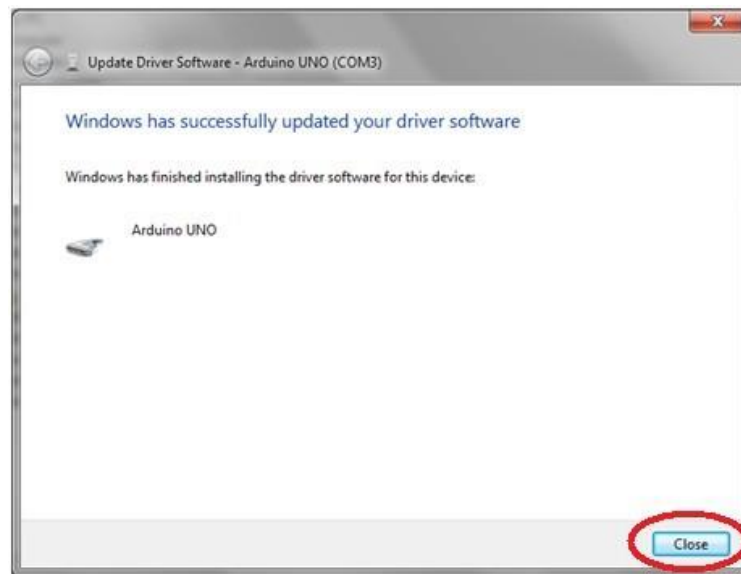


Figure 2.10 Screenshot of successful driver installation of the Uno board. The next step is to start doing the exercises.

WINDOWS INSTALLATION

Online version
arduino.cc/windows

INSTRUCTION FOR:
WINDOWS 7, VISTA,
AND XP

- 1 When the download of the IDE finishes, unzip the downloaded file. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.
- 2 Connect the Arduino to your computer with the USB cable. Your Arduino will automatically draw power from either the USB connection to the computer or an external power supply. The green power light (labeled PWR) should turn on.
- 3 Windows should initiate its driver installation process when the board is plugged in. Your computer won't be able to find the drivers by itself, so you'll need to tell it where they are located.
 - Click on the Start Menu and open the Control Panel.
 - Navigate to "System and Security". Open the Device Manager.
 - In Windows XP, look for the listing named "Ports (COM & LPT)" and right click on the "USB device" port; in Vista and Windows 7, right click on "Unknown device" under "Other devices".
 - Choose "Update Driver Software".
 - On Windows XP and Windows 7, you will be asked whether to install automatically or "with a path". Chose the second option, "with a path". On Windows Vista proceed directly to the next step.
 - Select the "Browse my computer for Driver software" option.
 - Navigate to the folder you unzipped in the earlier step. Locate and select the "Drivers" folder in the main Arduino folder (not the "FTDI USB Drivers" sub-directory). Press "OK" and "Next" to proceed.
 - If you are prompted with a warning dialog about not passing Windows Logo testing, click "Continue Anyway".
 - Windows now will take over the driver installation.

In the Device Manager, you should now see a port listing similar to "Arduino UNO (COM4)".

Congratulations! You've installed the Arduino IDE on your computer.

MAC OS X INSTALLATION

Online version
arduino.cc/mac

INSTRUCTION FOR:
OS X 10.5 AND
LATER

- 1 When the download of the IDE finished, double-click the .zip file. This will expand the Arduino application.
- 2 Copy the Arduino application into the Applications folder, or wherever else you wish to install the software.
- 3 Connect the board to the computer with the USB cable. The green power LED (labeled PWR) should turn on.
- 4 You do not need to install any drivers to work with the board. Depending on the version of OS X that you are running, you might get a dialog box asking if you wish to open the "Network Preferences". Click the "Network Preferences..." button, and then click "Apply".
- 5 The Uno will show up as "Not Configured", but it is still working. You can quit the System Preferences.

Congratulations! You have Arduino all set up and you're ready to start making projects.

LINUX INSTALLATION

If you're using Linux, please visit the website for instructions:
arduino.cc/linux

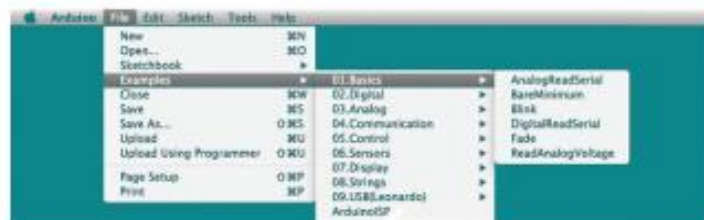
COMMUNICATING WITH THE ARDUINO

Now that you've installed the Arduino IDE and made sure your computer can talk to the board, it's time to make sure you can upload a program.

- 1 Double-click the Arduino application to open it. If the IDE loads in the wrong language, you can change this in the application preferences. Look for "Language Support" on this page for details: arduino.cc/ide

- 2 Navigate to the LED blink example sketch ('sketch' is what Arduino programs are called). It's located under:

FILE > EXAMPLES > 01.BASICS > BLINK



- 3 A window with some text in it should have opened. Leave the window be for now, and select your board under:

TOOLS > BOARD menu



- 4 Choose the serial port your Arduino is connected to from the **TOOLS > SERIAL PORT** menu.

— **On Windows.** This is likely to be the COM with the highest number. There is no harm in guessing wrong, and if it doesn't work, try the next one. To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

— **On Mac.** This should be something with /dev/tty.usbmodem in it. There are usually two of these; select either one.



Fig. 1

- 5 To upload the Blink sketch to your Arduino, press the **UPLOAD** toggle in the top left corner of the window. See Fig. 1.

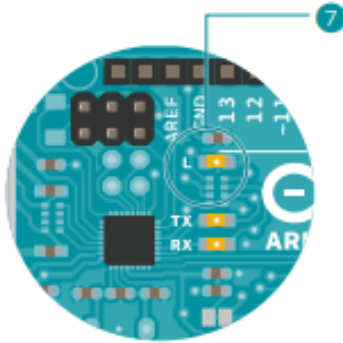


Fig. 2

- 6 You should see a bar indicating the progress of the upload near the lower left corner of the Arduino IDE, and the lights labeled TX and RX on the Arduino board will be blinking. If the upload is successful, the IDE will display the message **DONE UPLOADING**.

- 7 A few seconds after the upload has completed, you should see the yellow LED with an **L** next to it start blinking. See Fig. 2.

If this is the case, congratulations! You've successfully programmed the Arduino to blink its onboard LED!

Sometimes your brand new Arduino is already programmed with the Blink sketch, so you can't tell if you are truly in control. If this is the case, change the **delay** time by changing the number in the parenthesis to 100, and upload the Blink sketch again. Now the LED should blink much faster.

Congratulations! You really are in control! Now it's time to move on to Project 1. (You needn't save any changes you have made.)

ADDITIONAL INFORMATION

If you have problems with any of the steps outlined above, please see the troubleshooting suggestions:

arduino.cc/trouble

While you're getting ready to build your projects, you can look at the following page for additional information about the Arduino's programming environment:

arduino.cc/ide

You might also want to look at:

— the examples for using various sensors and actuators

arduino.cc/tutorial

— the reference for the Arduino language

arduino.cc/examples

Arduino Programming

Click on the Arduino executable which has the Arduino logo



The following screen comes up:



Figure 2.11 Screenshot of Sketches

The programs written for Arduino are called sketches. For the sketch to work on the Arduino Uno, there are two hardware related settings need to be done in the Arduino IDE – Board Serial Port.

For selecting the board, go to the Tools tab and select Board. From the menu select Uno.

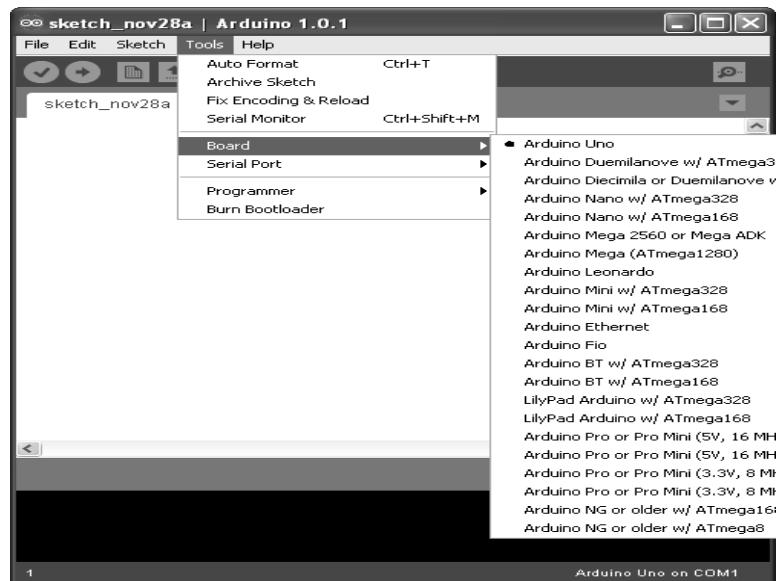


Figure 2.12 Screenshot of Connect the Arduino Uno to the USB port of PC

When you connect your Arduino Uno to the USB port of your PC, it will be mapped as a serial port. To know the serial port to which your Arduino is mapped, follow the following procedure:

Right click on My Computer

Select the Manage option

In the pop up screen for Computer Management, select the Device Manager

Expand the Ports item; the Arduino Uno will appear as one of the drop down items

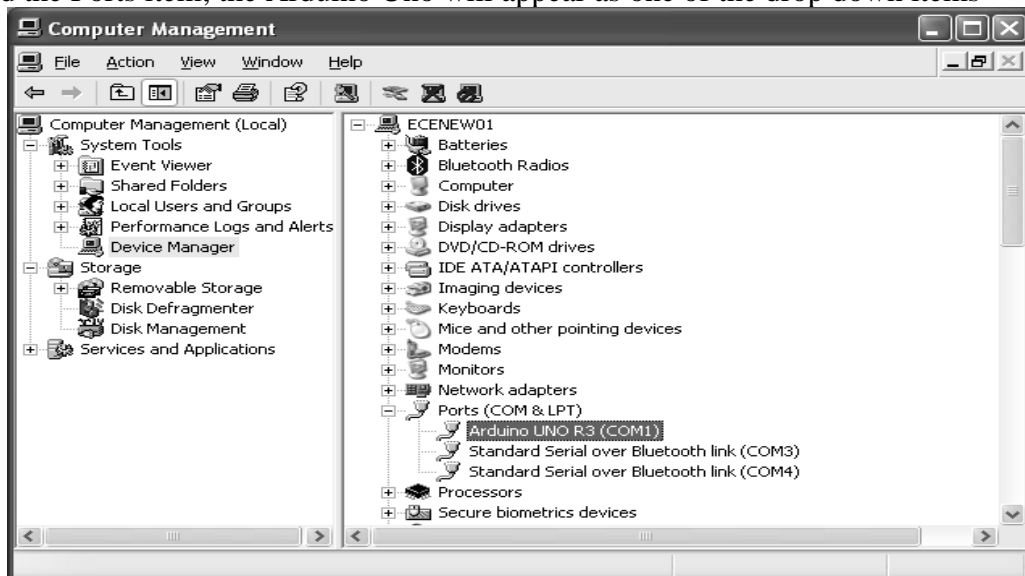


Figure 2.13 Screenshot of Port selection

In the Arduino IDE, select the Serial Port as the port to which the Arduino is mapped.

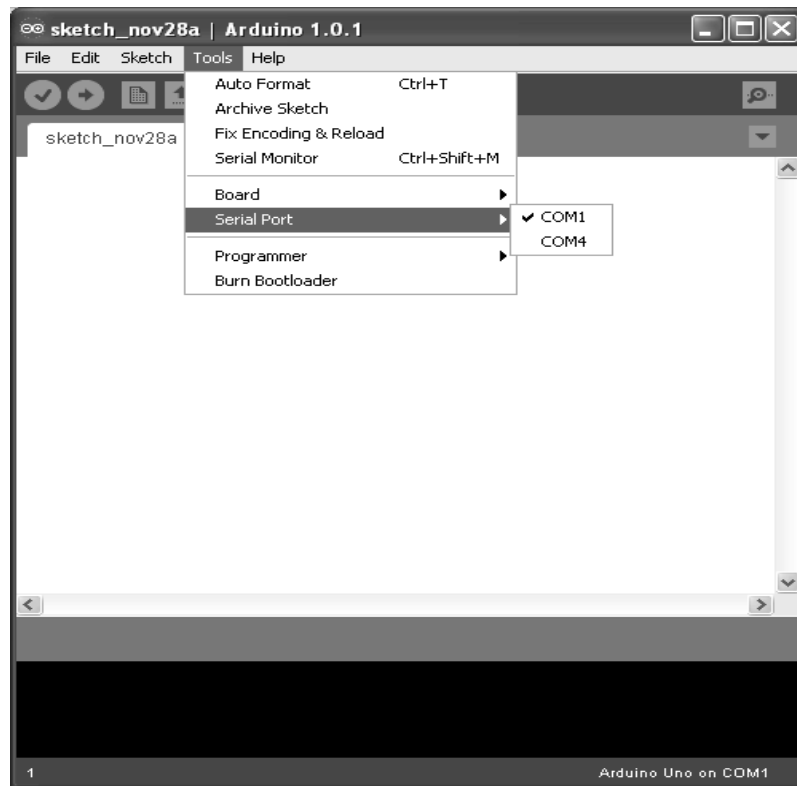


Figure 2.14 Screenshot of Mapping the serial port

The basic structure of the Arduino sketch is fairly simple and has two required functions:

```
void setup()
{
  statements;
}
void loop()
{
  statements;
}
```

Where `setup()` is the preparation, `loop()` is the execution. Both functions are required for the program to work. The setup function should follow the declaration of any variables at the very beginning of the program. It is the first function to run in the program, is run only once, and is used to set pin Mode or initialize serial communication.

The loop function follows next and includes the code to be executed continuously reading inputs, triggering outputs, etc. This function is the core of all Arduino programs and does the bulk of the work

setup()

The setup() function is called once when your program starts. Use it to initialize pin modes, or begin serial. It must be included in a program even if there are no statements to run.

```
void setup()
{
  pinMode(pin, OUTPUT); // sets the 'pin' as output
}
```

loop()

After calling the setup() function, the loop() function does precisely what its name suggests, and loops consecutively, allowing the program to change, respond, and control the Arduino board.

```
void loop()
{
  digitalWrite(pin, HIGH); // turns 'pin' on delay(1000); // pauses for one second
  digitalWrite(pin, LOW); // turns 'pin' off delay(1000); // pauses for one second
}
```

pinMode(pin, mode)

Used in void setup() to configure a specified pin to behave either as an INPUT or an OUTPUT.

```
pinMode(pin, OUTPUT); // sets 'pin' to output
```

There are also convenient pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed in the following manner:

```
Pin Mode (pin, INPUT); // set 'pin' to input digitalWrite(pin, HIGH); // turn on pullup resistors
```

Pullup resistors would normally be used for connecting inputs like switches. Notice in the above example it does not convert pin to an output, it is merely a method for activating the internal pull-ups.

Pins configured as OUTPUT can provide 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), but not enough current to run most relays, solenoids, or motors. Short circuits on Arduino pins and excessive current can damage or destroy the output pin, or damage the entire Atmega chip. It is often a good idea to connect an OUTPUT pin to an external device in series with a 470Ω or 1KΩ resistor.

digitalRead(pin)

Reads the value from a specified digital pin with the result either HIGH or LOW. The pin can be specified as either a variable or constant (0-13).

`value = digitalRead(Pin); // sets 'value' equal to the input pin`

digitalWrite(pin, value)

Outputs either logic level HIGH or LOW at (turns on or off) a specified digital pin. The pin can be specified as either a variable or constant (0-13).
`digitalWrite(pin, HIGH); // sets 'pin' to high`

The following example reads a pushbutton connected to a digital input and turns on an LED

connected to a digital output when the button has been pressed:

```
int led = 13; // connect LED to pin 13
int pin = 7; // connect pushbutton to pin 7
int value = 0; // variable to store the read value void setup()
{
  pinMode(led, OUTPUT); // sets pin 13 as output pinMode(pin, INPUT); // sets pin 7 as input
}
void loop()
{
  value = digitalRead(pin); // sets 'value' equal to the input pin digitalWrite(led, value); // sets 'led' to the button's value
}
```

analogRead(pin)

Reads the value from a specified analog pin with a 10-bit resolution. This function only works on the analog in pins (0-5). The resulting integer values range from 0 to 1023.
`value = analogRead(pin); // sets 'value' equal to 'pin'`

Note: Analog pins unlike digital ones, do not need to be first declared as INPUT or OUTPUT.
analogWrite(pin, value)

Writes a pseudo-analog value using hardware enabled pulse width modulation (PWM) to an output pin marked PWM. On Uno, this function works on pins 3, 5, 6, 9, 10, and 11. The value can be specified as a variable or constant with a value from 0-255. `analogWrite(pin, value); // writes 'value' to analog 'pin'`

A value of 0 generates a steady 0 volts output at the specified pin; a value of 255 generates a steady 5 volts output at the specified pin. For values in between 0 and 255, the pin rapidly alternates between 0 and 5 volts - the higher the value, the more often the pin is HIGH (5 volts). For example, a value of 64 will be 0 volts three-quarters of the time, and 5 volts one quarter of the time; a value of 128 will be at 0 half the time and 255 half the time; and a value of 192 will be 0 volts one quarter of the time and 5 volts three-quarters of the time.

Because this is a hardware function, the pin will generate a steady wave after a call to `analogWrite` in the background until the next call to `analogWrite` (or a call to `digitalRead` or `digitalWrite` on the same pin).

Note: Analog pins unlike digital ones do not need to be first declared as INPUT or OUTPUT.

The following example reads an analog value from an analog input pin, converts the value by dividing by 4, and outputs a PWM signal on a PWM pin:

```
int led = 10; // LED with 220 resistor on pin 10
int pin = A0; // potentiometer on analog pin 0
int value; // value for reading
void setup(){ } // no setup needed
void loop()
{
  value = analogRead(pin); // sets 'value' equal to 'pin' value /= 4; // converts 0-1023 to 0-255
  analogWrite(led, value); // outputs PWM signal to led
}
```

delay(ms)

Pauses a program for the amount of time as specified in milliseconds, where 1000 equals 1 second.

```
delay(1000); // waits for one second
```

millis()

Returns the number of milliseconds since the Arduino board began running the current program as an unsigned long value.

```
value = millis(); // sets 'value' equal to millis()
```

Note: This number will overflow (reset back to zero), after approximately 9 hours.

Serial.begin(rate)

Opens serial port and sets the baud rate for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported.

```
void setup()
{
  Serial.begin(9600); // opens serial port
} // sets data rate to 9600 bps
```

Note: When using serial communication, digital pins 0 (RX) and 1 (TX) cannot be used at the same time.

Serial.println(data)

Prints data to the serial port, followed by an automatic carriage return and line feed. This command takes the same form as `Serial.print()`, but is easier for reading data on the Serial

Monitor.

`Serial.println(analogValue);` // sends the value of

// 'analogValue'

Note: For more information on the various permutations of the `Serial.println()` and `Serial.print()` functions please refer to the Arduino website.

The following simple example takes a reading from analog pin0 and sends this data to the computer every 1 second.

```
void setup()
{
  Serial.begin(9600); // sets serial to 9600bps
}
void loop()
{
  Serial.println(analogRead(A0)); // sends analog value
  delay(1000); // pauses for 1 second
}
```

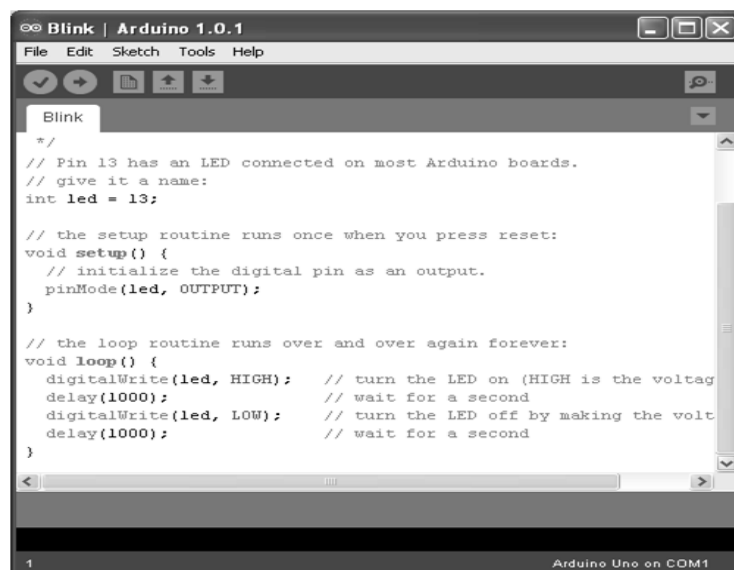


Figure 2.15 Screenshot of Arduino program compilations

After entering your program, click on the Verify button for compilation. If there are errors, the line numbers of the errors are shown in the bottom window. Correct the errors. After successful verification, upload your program to the Arduino using the Upload button. A common cause for failure in uploading is that your Arduino is not connected to a different COM port than the one shown in the Arduino IDE.

Expt.NO:1**Date:****LED Interface with Arduino****AIM:**

The main objective of this experiment is

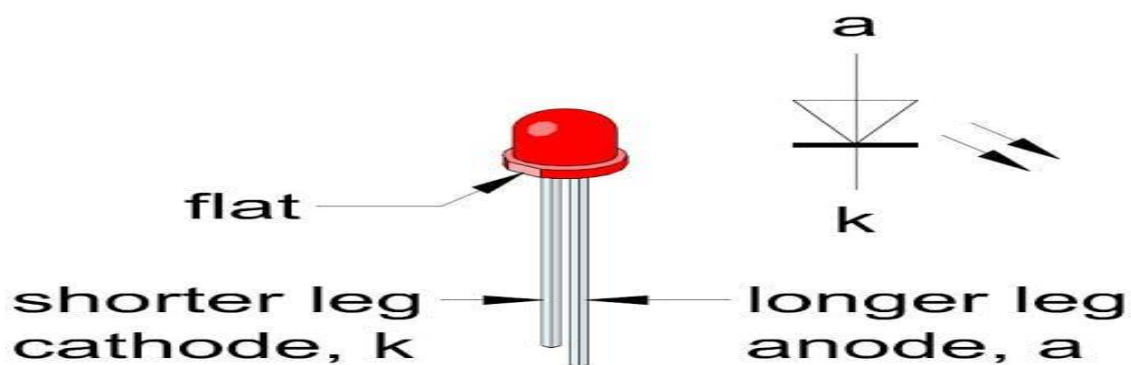
- To Blink an LED with delay of 2 seconds.
- To Blinking 2 LED'S alternatively with delay of second.
- To Blinking 2 LED'S Together with delay of 1 second.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
LED	1
Resistor (220 Ω)	1
Bread board	1
Jumper wires	2

Description:

Led blinking is the most beginner & easy step to start your experiment with arduino. Lets get started Firstly identify anode(+ve) & cathode (-ve) leg of LED. Following diagram gives a clear idea of LED



Now connect a resistor to positive leg and other end of resistor to 5V. Cathode will be connected to Arduino GPIO, in our case pin 13. Final circuit will look as below :

CONNECTION DIAGRAM:

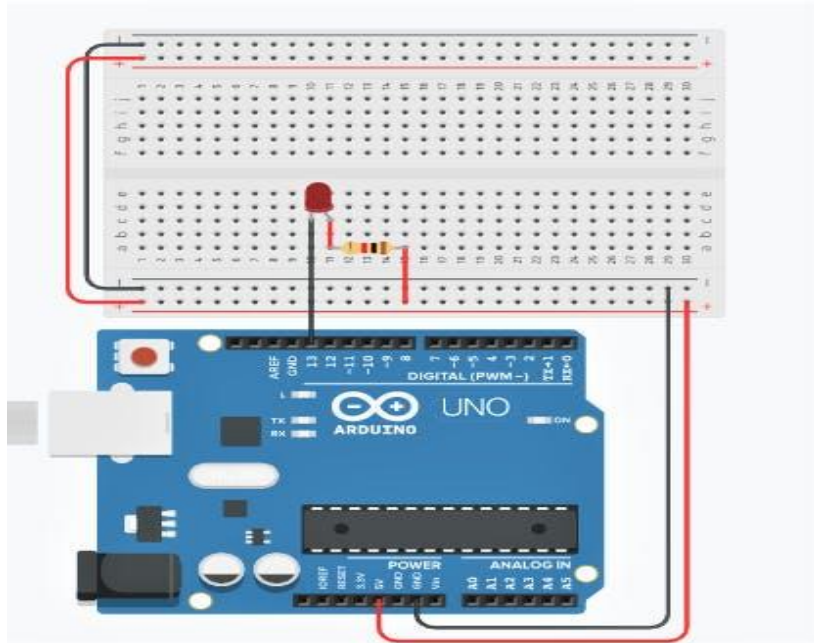
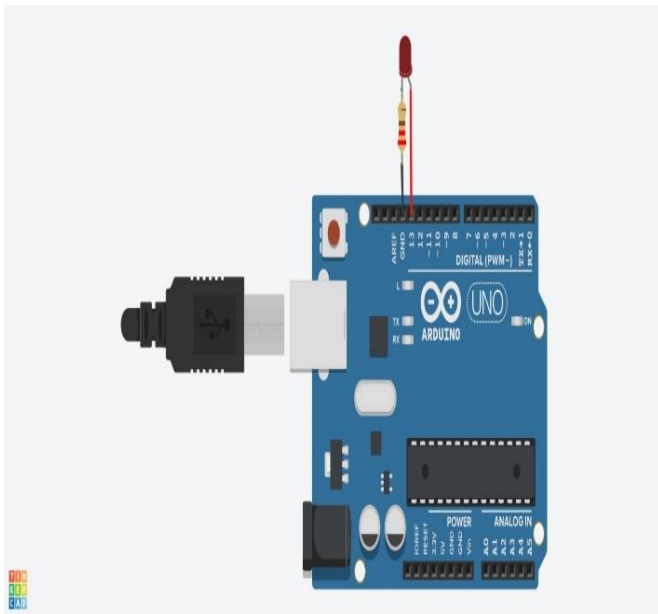
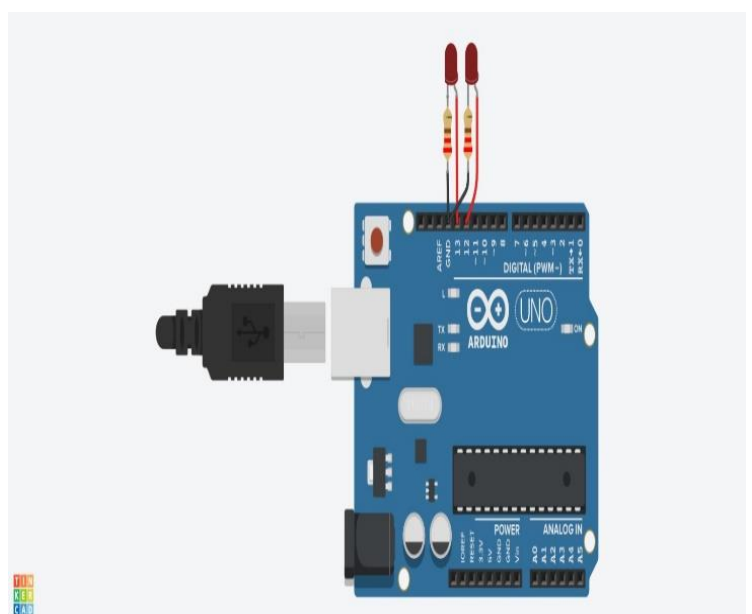


Fig.1.LED Interface with Aurdino



a.Blink an LED with delay of 2 seconds



b.Blink an 2 LED's alternatively

PRE-LAB SESSION

1. Draw the Circuit Diagram of LED interface with Arduino

2. Draw the Circuit Diagram of 2 LED interface with Arduino

3. How many analog pins in the Arduino Boards?

4. List out the types of LED's?

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--🔗 New
4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.
6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
7. After connecting, go to **Tools** ----🔗 **Board** ---🔗 Select **Arduino/Genuino Uno** option
8. After selecting board, go to **Tools** ----🔗 **Port** ---🔗 Select **Arduino Uno COM port 3** (name may appear differently for other computers).

****Note that this port option will be displayed only when board is connected to computer**

9. Now upload the program to the Arduino board by clicking **Upload** option.
10. Observe the output.

PROGRAM: A.Blink an LED with delay of 2 seconds

PROGRAM: B.Blink an 2 LED's alternatively

PROGRAM:C. Blinking 2 LED'S Together

INFERENCES/ANALYSIS

POST-LAB ASSIGNMENTS

- 1.What is the Command used to declare the inputs/outputs in Arduino
- 2.How many Analog and Digital pins in the Arduino Board
- 3.What is the Use of resistor in connection Diagram(Fig.1)?

Results:

Expt No No.: 2**Date:****Controlling LED Interface With Arduino using PUSH button****AIM**

The main objective of this experiment is

- A. Read the Push button State Using Serial Monitor
- B. Turn ON an LED when a button is pressed, OFF when button is released.
- C. Blinking LED using Push Button with delay 1s

SOFTWARES USED

Arduino UNO (Arduino IDE)

APPARATUS/ COMPONENTS REQUIRED

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
LED	1
Resistor (220Ω)	2
Push button	1
Bread board	1
Jumper wires	5

THEORY:

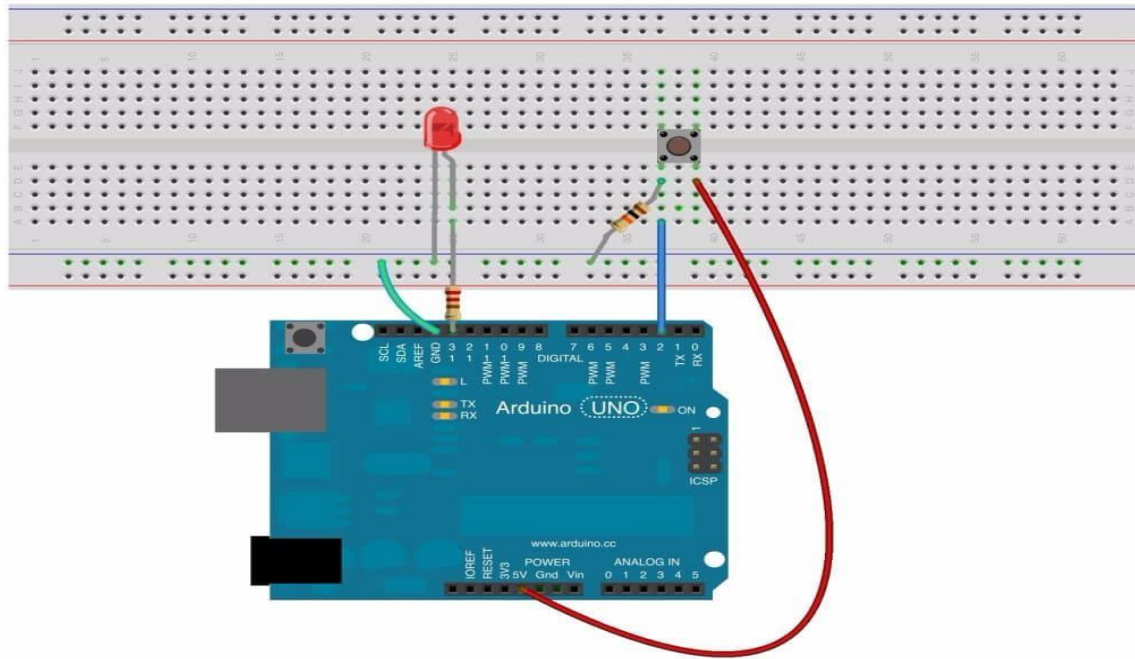
In this experiment we simply hook up 5 volts to one side of a button and to the other side of the button we connect pin 2. When you press the button it completes an electrical connection, pin 2 will “see” the 5 volts and if we digitalRead() at pin 2, it will report HIGH. During the times the button is not being pressed, pin 2 reports LOW.

To turn on an LED by pressing the button, we simply make an if statement whose condition says something like “...if the voltage at pin 2 is HIGH, turn on the LED at pin 13...” It is really that easy.

We also have pin 2 connected to ground at all times through a resistor. This is because when we read values at pin 2, we want to get either a HIGH or a LOW reported. If pin 2 is not connected to ground, then when the button is not being pressed it becomes what is called a floating pin – it’s not connected to anything. Floating pins on the Arduino are fine for the most

part – unless you are trying to record an input from them – then they are bad, and can give you spurious information.

CONNECTION DIAGRAM:



PRE-LAB

1. Draw the Circuit Diagram for the Connection Diagram

2. Draw the Pin diagram for Push-Button

PROCEDURE:

1. Connect one of the Arduino GND pins to one of the long power rails on the breadboard – this will be the ground rail.
2. Connect the short leg of the LED to this same ground rail on the breadboard then connect the long leg to a row on the breadboard.
3. Connect the 220-ohm resistor from pin 13 to the same row that you have the long leg of the LED attached.
4. Place the pushbutton on the breadboard. Most buttons will straddle the center trench on the breadboard.
5. Connect a jumper wire from the 5-volt pin to one side of the pushbutton.
6. Connect a jumper wire from pin 2 to the other side of the pushbutton.
7. Connect one side of the 10k resistor from the ground rail on the breadboard to the other side to the pushbutton – on the same side that pin 2 connects.
8. Plug the Arduino board into your computer with a USB cable.
9. Open the Arduino IDE.
10. Open the sketch for this section.
11. Click the Verify button on the top left. It should turn orange and then back to blue.
12. Click the Upload button. It will also turn orange and then blue once the sketch has finished uploading to your Arduino board.
13. Press the button a couple times and see how the LED at pin 13 reacts.

PROGRAM:

A.Turn ON an LED when a button is pressed, OFF when button is released.

B. Blinking LED using Push Button with delay 1s

PROGRAM:

INFERENCES/ANALYSIS

POST-LAB

- 1.How many PWM Pins Available in Arduino UNO?
- 2.Listout the PIN Terminals in PUSH -Button
- 3.Is that 5V Supply is mandatory for Push Button?Why?

RESULT:

Expt No.: 3

Date:

Automatic Street Light Controlling Using Arduino**AIM**

The main objective of this experiment is to design a Automatic Street Light Controller Using LDR Sensor and Arduino UNO

SOFTWARES USED

Arduino UNO (Arduino IDE)

APPARATUS/ COMPONENTS REQUIRED

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
Resistor (1K Ω)	1
LDR	1
Bread board	1
Jumper wires	3

THEORY:

In our experiment instead of using street light we using LED. Basically when there is darkness the led will glow and when there is sufficient light led will stop glowing. This a simple circuit for of interface Arduino uno with LDR sensor.

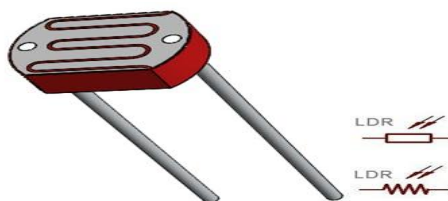


Fig.1.LDR Sensor and Symbol

LDR (light dependent resistor) also called photoresistors are responsive to light. Photoresistors are used to indicate the intensity or the presence or the absence of light. When there is darkness the resistance of photoresistor increases and when there is sufficient light it dramatically decreases.

LDR (light dependent resistor) which has two terminals. Terminal one is the signal pin which should be connected according to the code. Another terminal is considered as the ground pin

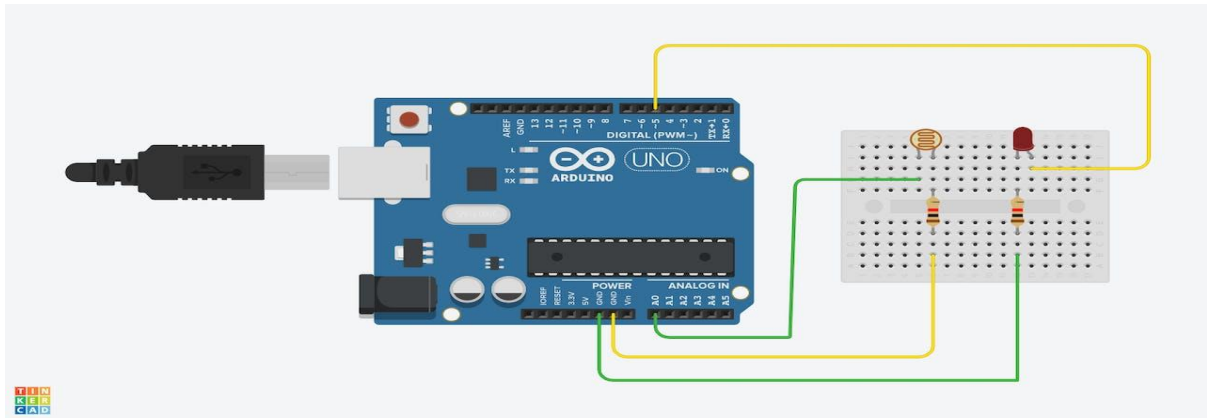
which should be connected to the ground of the system.

In the simplest terms, a light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current (known as electrons and holes) combine together within the semiconductor material. Led has two terminals : positive and negative.

Connections of LDR sensor : First terminal should be connected to analog pin 0 (A0) of Arduino. Second terminal should be connected any one led pf the resistor. Another leg of resistor should be connected to Gnd of Arduino.

Led connections : Positive pin should be connected to digital pin 5 of Arduino. Negative pin should be connected any one led pf the resistor. Another leg of resistor should be connected to Gnd of Arduino.

CONNECTION DIAGRAM:



PRE-LAB

1. Draw the Circuit Diagram for the Connection Diagram

2. Draw the Pin diagram for LDR sensor Module

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
 2. Open Arduino IDE in the computer
 3. Create new file File_--→ New
 4. Type your program and Save it in appropriate location in your computer.
 5. Compile your program by clicking Verify option in the menu.
 6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
 7. After connecting, go to Tools ----→ Board ----→ Select Arduino/Genuino Uno option
 8. After selecting board, go to Tools ----→ Port ----→ Select Arduino Uno COM port 3 (name may appear differently for other computers).
- **Note** that this port option will be displayed only when board is connected to computer
9. Now upload the program to the Arduino board by clicking Upload option.
 10. Once the program uploaded successfully, open serial monitor window in Arduino IDE to see the value of LDR. When you change the brightness of LDR, observe the LED glows when LDR values reaches threshold “200”.

CODE/PROGRAM:-

INFERENCE ANALYSIS

POST-LAB

- 1.What is the Principle of LDR sensor?
- 2.List out the pin in LDR Sensor module?
- 3.What is the Difference between Analog and Digital Sensor?
- 4.Is that LDR is Analog or Digital Sensor?

RESULT:

Exp No.: 4

Date:

Smoke Detection Using MQ-2 GAS Sensor**AIM**

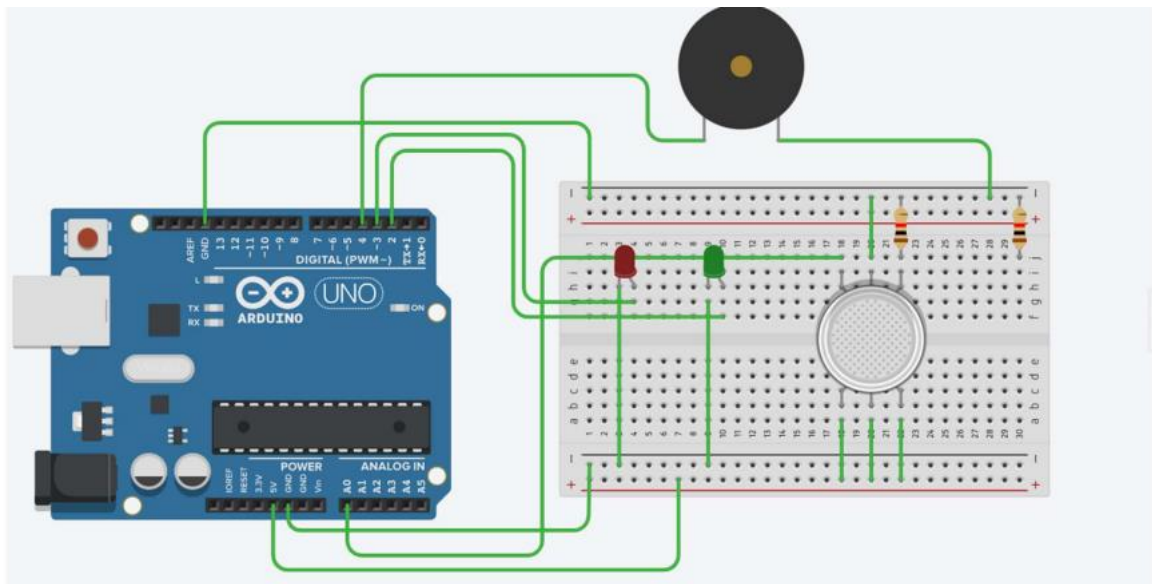
The main objective of this experiment is to read the MQ-2 sensor value and Turn on Buzzer and LED if analogue sensor value exceeds its threshold value.

SOFTWARES USED

Arduino UNO (Arduino IDE)

APPARATUS/ COMPONENTS REQUIRED

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
Buzzer ,LED	1 /2
Resistor (220Ω, 1KΩ)	2
Smoke Sensor(MQ2)	1
Bread board	1
Jumper wires	6

CONNECTION DIAGRAM For Tinkercad

PRE-LAB

1. Draw the Circuit Diagram for the Connection Diagram

2. Draw the Pin diagram for LDR sensor Module

THEORY:

In this Experiment , you will read the sensor analog output voltage and when the smoke reaches a certain level, it will make sound a buzzer and a red LED will turn on.

When the output voltage is below that level, a green LED will be on.

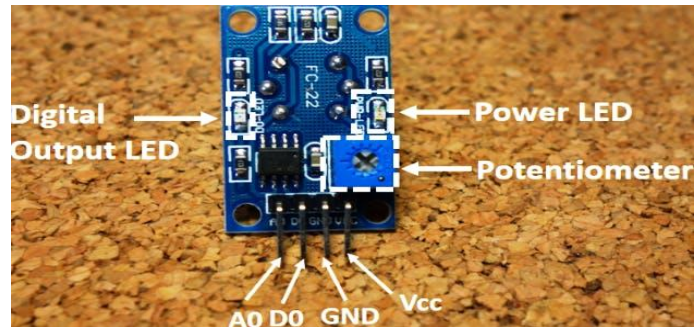
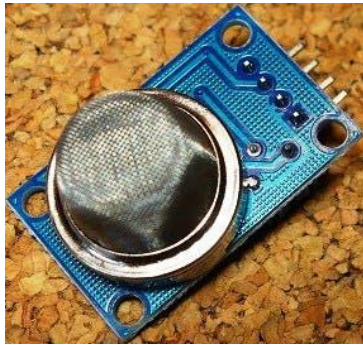
MQ-2 Smoke Sensor

The MQ-2 smoke sensor is sensitive to smoke and to the following flammable gases:

- LPG
- Butane
- Propane
- Methane
- Alcohol
- Hydrogen

The resistance of the sensor is different depending on the type of the gas.

The smoke sensor has a built-in potentiometer that allows you to adjust the sensor sensitivity according to how accurate you want to detect gas.

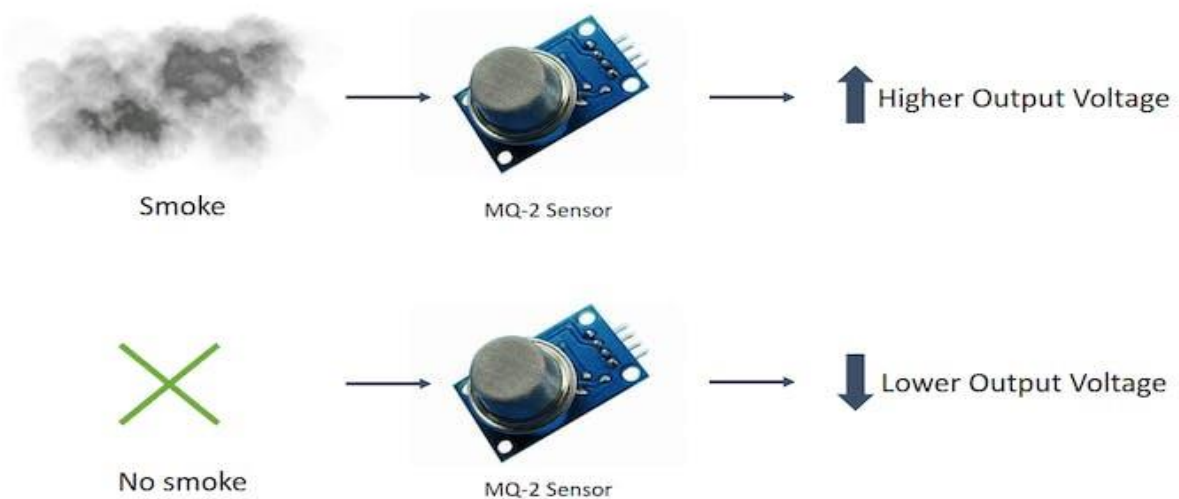


MQ2 Sensor Pin-out details

The voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. The sensor outputs a voltage that is proportional to the concentration of smoke/gas.

In other words, the relationship between voltage and gas concentration is the following:

- The greater the gas concentration, the greater the output voltage
- The lower the gas concentration, the lower the output voltage



Working Mechanism

The output can be an analog signal (A0) that can be read with an analog input of the Arduino or a digital output (D0) that can be read with a digital input of the Arduino.

Pin Wiring

The MQ-2 sensor has 4 pins.

Pin-----Wiring to Arduino Uno

A0-----Analog pins

D0-----Digital pins

GND-----GND

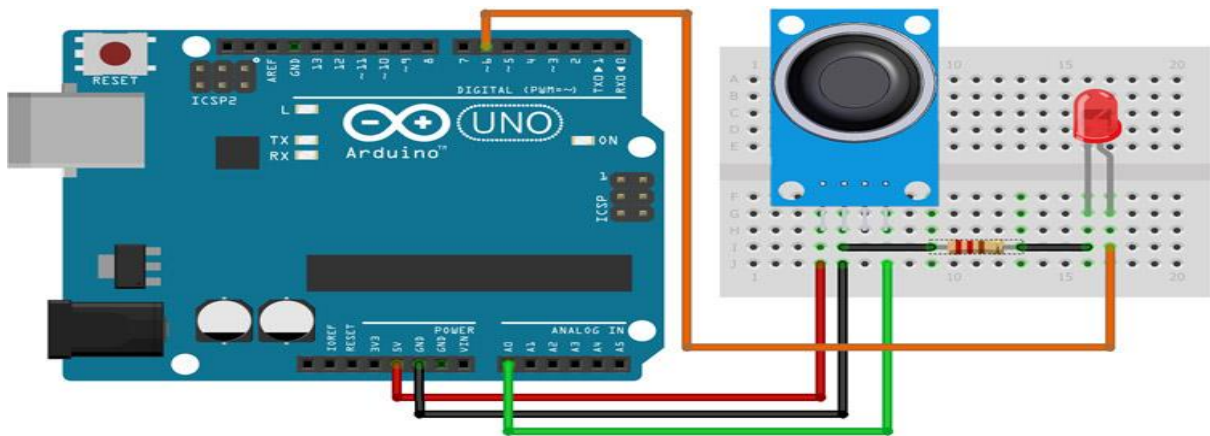
VCC-----5V

So, before jumping into the coding part, let's check whether we've assembled all the necessary hardware components.

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--⑦ New
4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.
6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
7. After connecting, go to **Tools** ----⑦ **Board** ---⑦ Select **Arduino/Genuino Uno** option
8. After selecting board, go to **Tools** ----⑦ **Port** ---⑦ Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.
10. Once the program uploaded successfully, open serial monitor window in Arduino IDE to see the value of Gas Sensor. When you change the Smoke content of MQ2, observe the buzzer beeps when Sensor values reaches threshold "300" and according to that LED glow.

PROGRAM for Tinkercad Connection

Program for Interface MQ2 Sensor With Arduino:

INFERENCES/ANALYSIS

POST-LAB

- 1.What is the Principle of MQ2 Gas sensor?
- 2.List out the pin in MQ2 Sensor module?
- 4.Why that MQ2 Gas Sensor is Analog Sensor?

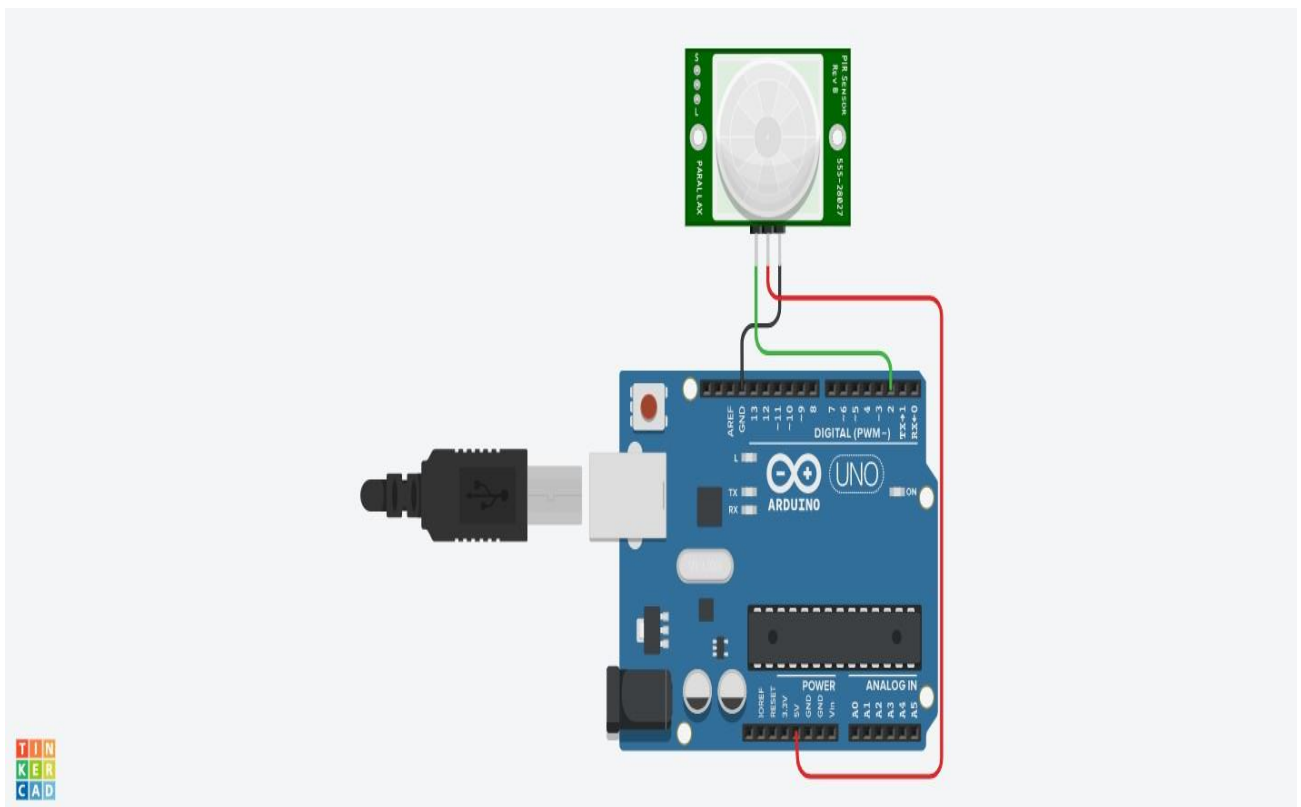
Results

ExptNo No.: 5**Date:****Interfacing PIR Sensor****AIM:**

The main objective of this experiment is to read the digital sensor value and display the sensor value in serial monitor.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
Digital PIR/PING/Motion Sensor	1
Bread board	1
Jumper wires	3

CONNECTION DIAGRAM:

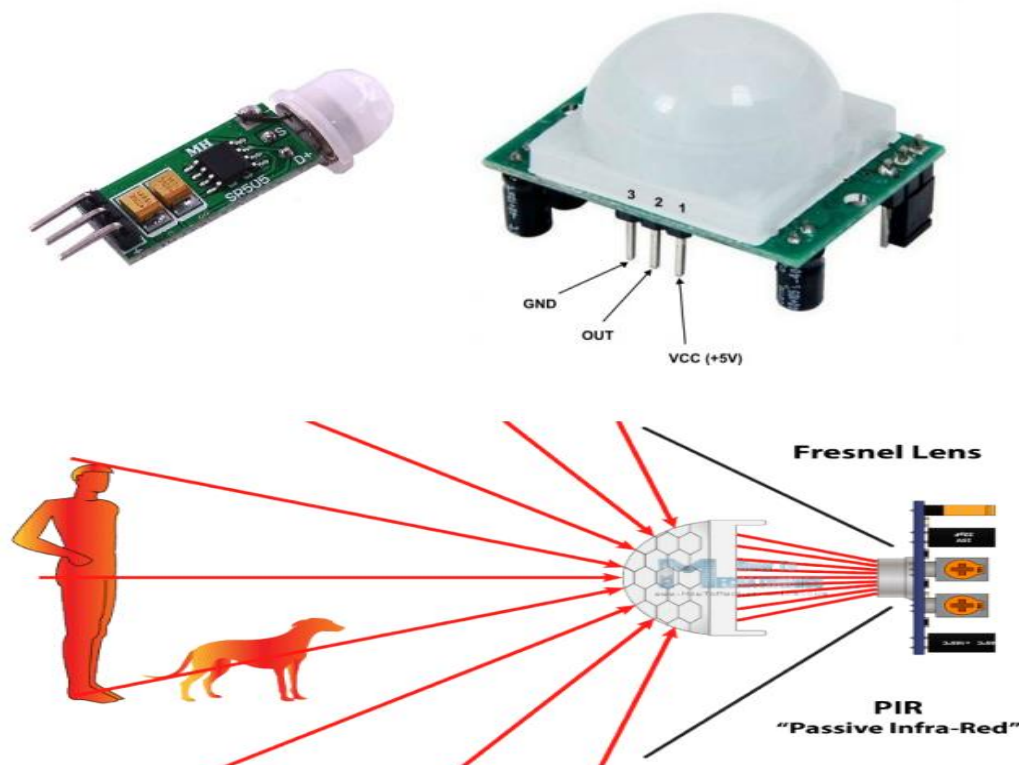
PRE-LAB

1. Draw the Circuit Diagram for the Connection Diagram

2. Draw the Pin diagram for PIR sensor Module

THEORY

Descriptions: • PIR stands for Passive Infrared Sensor. • It is called passive sensor because it does not emit the IR rays. It only receives the IR rays. • PIR sensor allows to detect or sense the motion. • It is small, Inexpensive and low power device. • It is non wearable device for that reason they are commonly found in appliances and gadgets used in homes or businesses.



• It is having 3 pin structure where , o Pin 1 is Vcc (+5v) o Pin 2 is Output o Pin 3 is ground • All pins

must be connected with arduino pins. • PIR sensor can detect animal/human movement in a requirement range. • For Indoor passive infrared Detection distances range from 25 cm to 20 m • For Outdoor passive infrared, the detection distance ranges from 10 meters to 150 meters. • It detects infrared radiation from the environment. Once there is infrared radiation from the human body particle with temperature, focusing on the optical system causes the device to generate a sudden electrical signal.

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--⑦ New
4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.
6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
7. After connecting, go to **Tools** ----⑦ **Board** ---⑦ Select **Arduino/Genuino Uno** option
8. After selecting board, go to **Tools** ----⑦ **Port** ---⑦ Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.
10. Once the program uploaded successfully, open serial monitor window in Arduino IDE to see the value of PIR sensor. When you give motion Infront of sensor, observe the value changes in serial monitor.

PROGRAM:

INFERENCES/ANALYSIS

POST-LAB

- 1.What is the Principle of PIR sensor?
- 2.List out the pin in PIN Sensor module?
- 4.Is that PIR Sensor is Analog Sensor or Digital Sensor

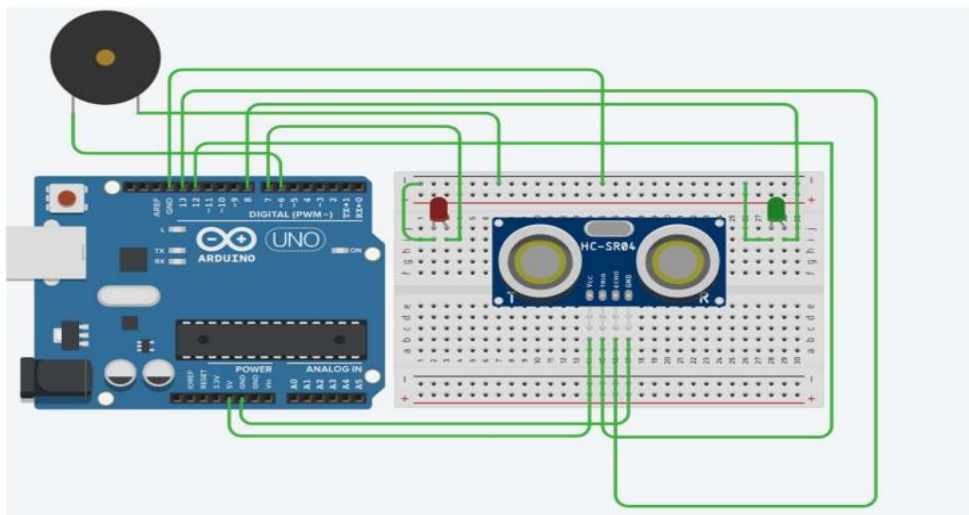
RESULT:

Expt No.:6**Date:****Interfacing Ultrasound Sensor with Arduino****Aim:**

To Interface an Ultrasonic sensor to activate a buzzer and RED LED if the distance between the sensor and object is less than 35 inches and activate Green LED and OFF the buzzer when distance is more than 35 inches.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
Ultrasound Sensor	1
Bread board	1
Jumper wires	3

CONNECTION DIAGRAM:**DESCRIPTION**

This application is based upon the reflection of sound waves. Sound waves are defined as longitudinal pressure waves in the medium in which they are travelling. Subjects whose dimensions are larger than the wavelength of the impinging sound waves reflect them; the

reflected waves are called the echo. If the speed of sound in the medium is known and the time taken for the sound waves to travel the distance from the source to the subject and back to the source is measured, the distance from the source to the subject can be computed accurately. In this distance measurement system we had ultrasonic sensor HC-SR04 interfaced with Arduino UnoR3. Programming and hardware part of ultrasonic sensor interfacing with arduino UnoR3 and use a proper display unit for measurement of distance.

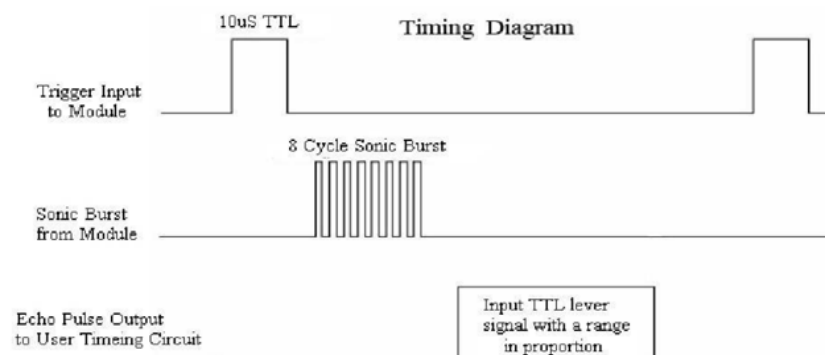
Ultrasonic sensor HC-SR04 is used here to measure distance in range of 2cm-400cm with accuracy of 3mm. The sensor module consists of ultrasonic transmitter, receiver and the control circuit. The working principle of ultrasonic sensor is as follows:

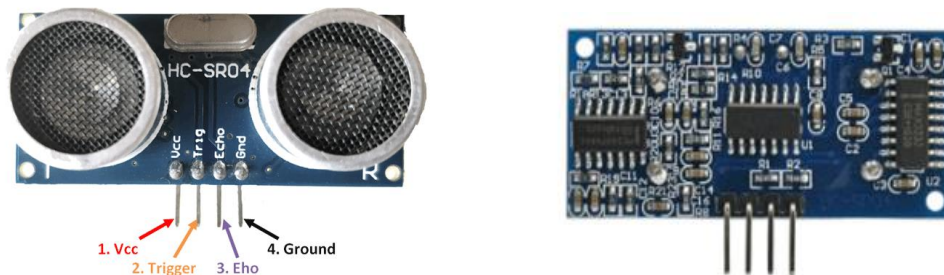
- i. High level signal is sent for 10 μ s using Trigger.
- ii. The module sends eight 40 KHz signals automatically, and then detects whether pulse is received or not.
- iii. If the signal is received, then it is through high level. The time of high duration is the time gap between sending and receiving the signal.

Thus the distance of the obstacle from the sensor is calculated by the formula given as:

$$\text{Distance} = (\text{Time} \times \text{Speed of Sound in Air (340 m/s)}) / 2$$

Here we have divided the product of speed and time by 2 because the time is the total time it took to reach the obstacle and return back. Thus the time to reach obstacle is just half the total time taken.





Specifications:

The HC-SR04 Ultrasonic Sensor contains the following specifications

Working Voltage: DC 5 V

Working Current: 15mA

Working Frequency Min Range 2cm: 40Hz Measuring Angle 15 degree

Max Range: 4m

Min Range: 2cm

Measuring Angle: 15 degree

Trigger Input Signal: 10uS TTL pulse

Echo Output Signal: Input TTL level signal and the range in proportion

Dimension: 45*20*15mm

Ultrasonic Sensor Pin Configuration:

Ultrasonic Sensor HC-SR04	Arduino	Description
VCC	5V	The Vcc pin powers the sensor, typically with +5V
Trig	Pin 11	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
Echo	Pin 12	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
GND	GND	This pin is connected to the Ground of the system.

Pre-lab Session:

1. Draw the Block Diagram for Connection Diagram

2. Draw the Pin Diagram For UltraSound Sensor

3. What is the Relationship between wavelength and Frequency?

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
 2. Open Arduino IDE in the computer
 3. Create new file File_--➤ New
 4. Type your program and **Save** it in appropriate location in your computer.
 5. Compile your program by clicking **Verify** option in the menu.
 6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
 7. After connecting, go to **Tools** ----➤ **Board** ---➤ Select **Arduino/Genuino Uno** option
 8. After selecting board, go to **Tools** ----➤ **Port** ---➤ Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.
 10. Once the program uploaded successfully, observe the distance measurement in serial monitor according to that LEDs will glow and Buzzer will blow.

PROGRAM:

INFERENCES/ANALYSIS

First of all we need 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. PulsIn () function will wait for the echo pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 or 1/29.1 and divide it by 2. At the end we will print the value of the distance on the Serial Monitor. We can calculate distance by using given formula:

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

The speed of sound is: $343\text{m/s} = 0.0343\text{ cm/uS} = 1/29.1\text{ cm/uS}$

Applications

- Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
- Used to measure the distance within a wide range of 2cm to 400cm
- Can be used to map the objects surrounding the sensor by rotating it
- Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water

Post Lab Session:

1. How many pins for ultrasonic sensor? What are they?
2. What are i/p and o/p pins and purpose?
- 3) Is it mandatory to connect only to 11,12 pins of arduino?
- 4) What is the use of serial.print?
- 5) Explain the function of digitalwrite?

RESULT:

ExptNo No.: 7

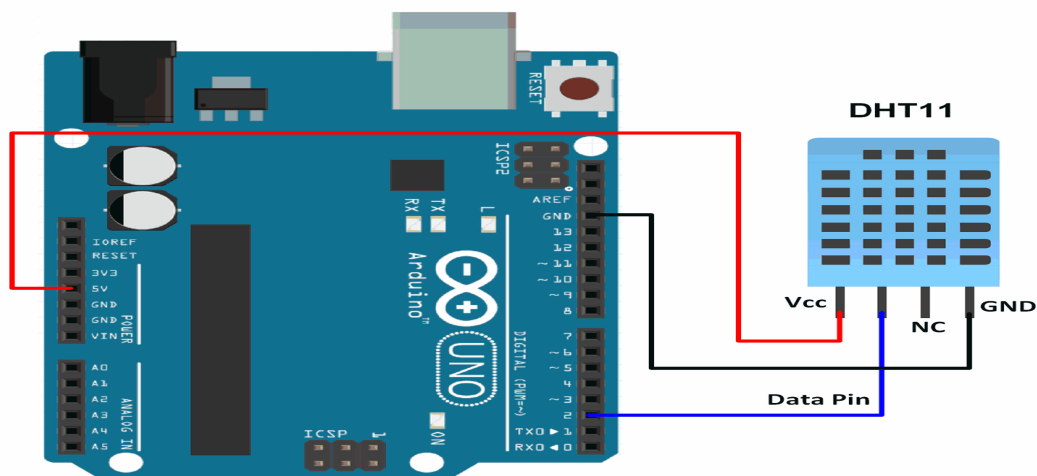
Date:

Interfacing Temperature Sensor**AIM:**

The main objective of this experiment is to read the DHT11 sensor value and display the sensor value in serial monitor.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
DHT11 Sensor	1
Bread board	1
Jumper wires	3

CONNECTION DIAGRAM:**PRE-LAB**

1. Draw the Circuit Diagram for the Connection Diagram

2. Draw the Pin diagram for DHT11 Module

THEORY

In this experiment you will learn how to set up the DHT11 Humidity and Temperature sensor on your Arduino UNO. And learn about how the Humidity sensor works, and how to check output readings from the Serial monitor

Description:

The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

- DHT11 sensor measures and provides humidity and temperature values serially over a single wire.
- It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.
- It has 4 pins; one of which is used for data communication in serial form.
- Pulses of different TON and TOFF are decoded as logic 1 or logic 0 or start pulse or end of a frame.

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
 2. Open Arduino IDE in the computer
 3. Create new file File_--**7** New
 4. Type your program and **Save** it in appropriate location in your computer.
 5. Compile your program by clicking **Verify** option in the menu.
 6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
 7. After connecting, go to **Tools** ----**7** **Board** ---**7** Select **Arduino/Genuino Uno** option
 8. After selecting board, go to **Tools** ----**7** **Port** ---**7** Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.
 10. Once the program uploaded successfully, open serial monitor window in Arduino IDE to see the value of DHT sensor. When you give motion Infront of sensor, observe the value changes in serial monitor.

PROGRAM:

INFERENCES/ANALYSIS

POST-LAB

- 1.What is the Principle of DHT11 sensor?
- 2.List out the pin in Sensor module?
- 4.Is that DHT11 Sensor is Analog Sensor or Digital Sensor

RESULT:

Expt No.:8

Date:

Interfacing Relay to Arduino

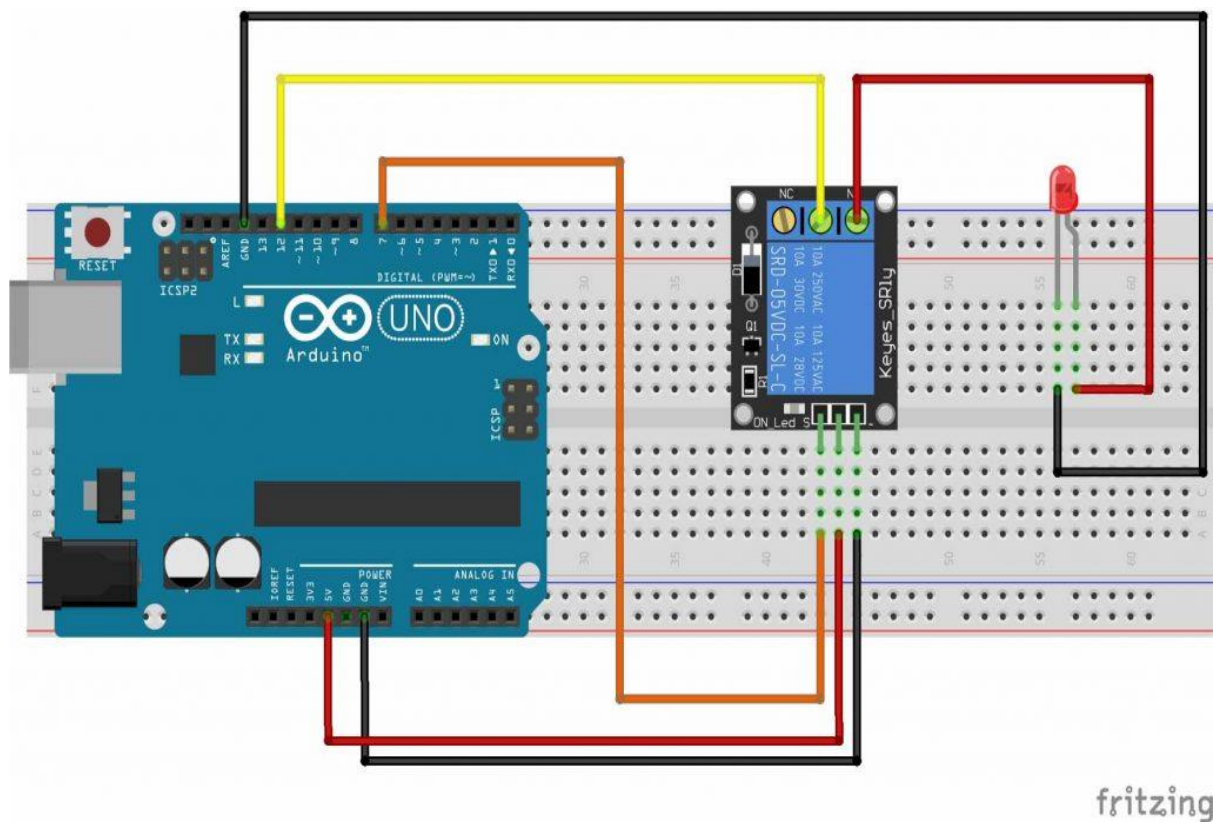
Aim:

The main objective of this experiment is to interface relay to the Arduino Uno board and control its position.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
relay	1
Bread board	1
Jumper wires	3

CONNECTION DIAGRAM:



Pre-lab Session:

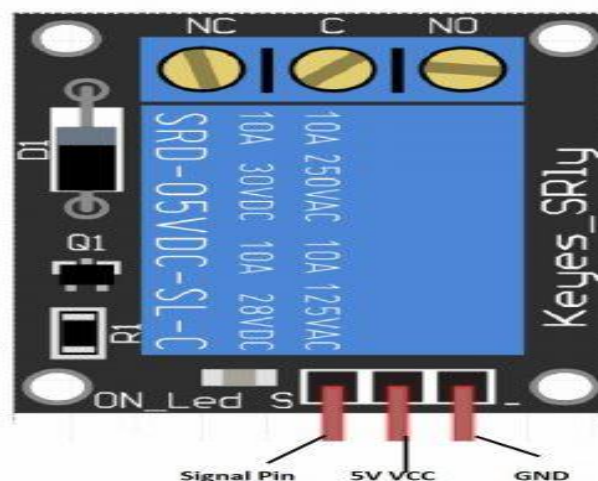
1. Draw the Block Diagram for Connection Diagram
2. Draw the Pin Diagram For Relay

THEORY**Relay Module**

A relay is generally an electrically operated switch. The principle used by the relays is an electromagnet to mechanically operate the switch. So basically it operates a switch using an electromagnet which needs only less power like 5V, 12V or 24V. Different kinds of relays are available in the market like SPDT, DPDT, SPST, 5V, 12V, 24V and with various high current/voltage driving capacity. In this tutorial we are using a 5V relay module.

Pinouts

Let's dive deep into the 5V relay module which we are using. This module has total 6 pins out of which 3 are used for controlling the relay (low voltage side). While other 3 are the output of the relay (usually high voltage side), where we will connect the device which we want to control.



- **NO : Normally Open** – This is the normally open terminal of the relay, means if we don't

energise the relay there won't be any contact with Common terminal. But it will establish electrical contact with Common terminal once the relay is energized.

- **C** : Common terminal
- **NC : Normally Closed** – This is the normally closed terminal of the relay, means it will have electrical contact with common terminal whenever the relay is not energised. And there won't be electrical contact when the relay is energised.
- **GND** : Ground Pin
- **Signal** : Actuation signal to control the relay.
- **5V VCC** : Operating voltage for the relay.

Working

The principle behind the relay is electromagnetism, a switch operated by an electromagnet. So, a low voltage is enough for an electromagnet to get activated. This small voltage will be given to the relay by Arduino Uno or using some intermediate driver if required.

NO, NC and COM Terminals

Usually SPDT (Single Pole Double Throw) relays have 3 output terminals, these are the 3 terminals of internal SPDT electromagnetic switch.

Common (COM)

This is the commonly terminal. This terminal will be connected to either of other 2 terminals (NO or NC) based on the state of relay.

Normally Open (NO)

As the name indicates this is normally open terminal, ie. if the relay is not energized (not ON), this pin will be open. We can say that the switch is OFF by default and when the relay is energized it will become ON.

Normally Closed (NC)

As the name indicates it is normally closed terminal, ie. if the relay is not energized (not ON), this pin will be closed. We can say that the switch is ON by default and when the relay is energized it will become OFF.

PROCEDURE:

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--**7** New
4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.
6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
7. After connecting, go to **Tools** ----**7** **Board** ---**7** Select **Arduino/Genuino Uno** option
8. After selecting board, go to **Tools** ----**7** **Port** ---**7** Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.

10. Once the program uploaded successfully, observe the relay operation.

Description

GND pin of 5V Relay – GND pin of Arduino

Signal (Input) pin of 5V Relay – pin 7 of Arduino

VCC pin of 5V Relay – 5V pin of Arduino

Common pin of 5V Relay – pin 12 of Arduino

NO pin of 5V Relay – Positive pin of the LED

GND pin of LED – GND pin of Arduino

PROGRAM:**PRE-LAB**

1. List out Types of Relay?
2. List out the pin in Relay module?
3. How many PWM pin available in Arduino Board?

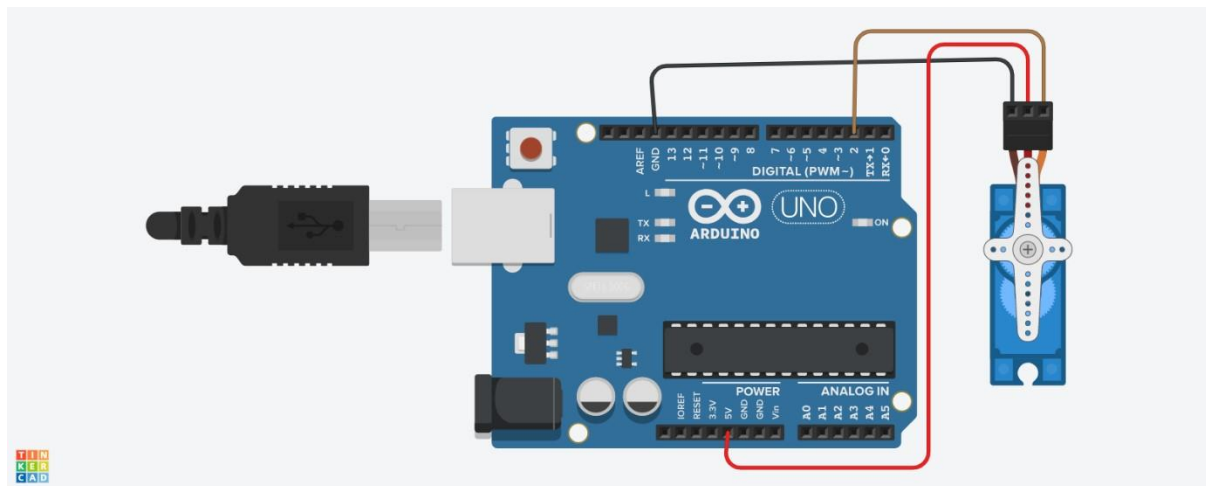
INFERENCE/ANALYSIS:**RESULT:**

Expt No.:9**Date:****Interfacing Servo Motor****Aim:**

The main objective of this experiment is to interface servo motor to the Arduino Uno board and control its position.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
Servo motor	1
Bread board	1
Jumper wires	3

CONNECTION DIAGRAM:**PROCEDURE:**

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--**7** New
4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.

6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
 7. After connecting, go to **Tools** ---- **Board** --- Select **Arduino/Genuino Uno** option
 8. After selecting board, go to **Tools** ---- **Port** --- Select **Arduino Uno COM port 3** (name may appear differently for other computers).
- **Note that this port option will be displayed only when board is connected to computer**
9. Now upload the program to the Arduino board by clicking **Upload** option.
 10. Once the program uploaded successfully, observe the servo motor is changing its position.

PROGRAM:

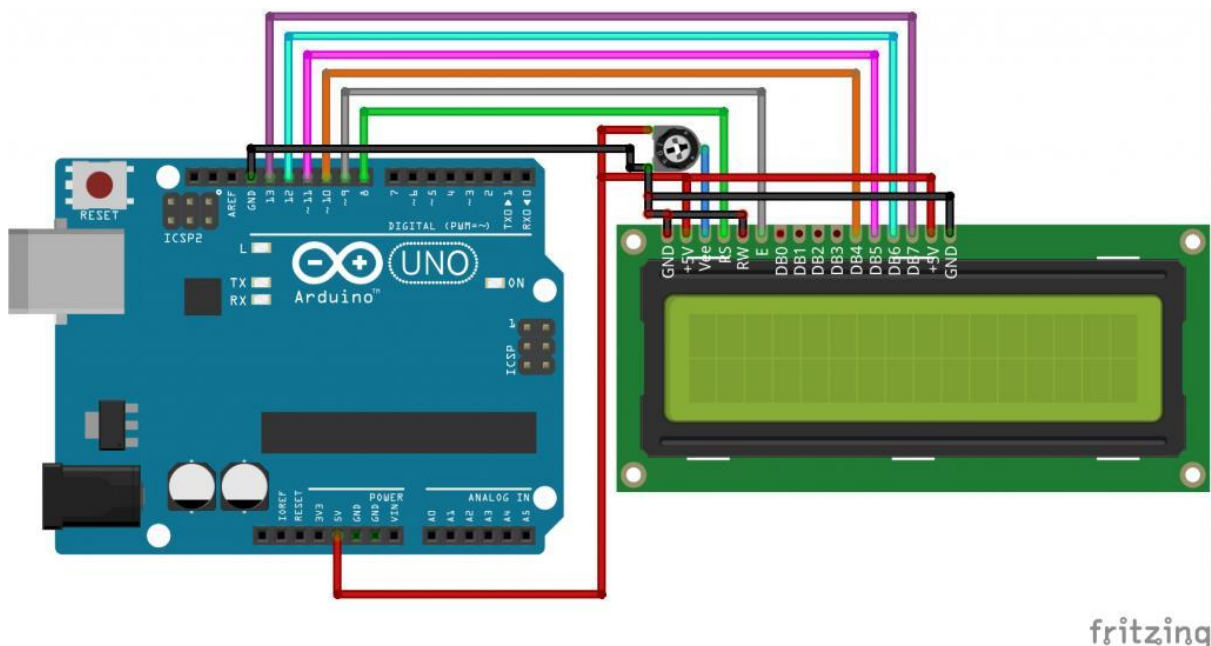
RESULT:

Expt No.:10**Date:****LCD Display Interfacing with Arduino****Aim**

The main objective of this experiment is to interface LCD to the Arduino Uno board and display “Hello, Your Name”.

APPARATUS REQUIRED:

Name of the Component	No of Units
Arduino Uno development board	1
USB cable	1
LCD	1
Potentiometer	1
Bread board	1
Jumper wires	As required

CONNECTION DIAGRAM:**PROCEDURE:**

1. Give the connections to the Arduino board as shown in the connection diagram.
2. Open Arduino IDE in the computer
3. Create new file File_--🔍 New

4. Type your program and **Save** it in appropriate location in your computer.
5. Compile your program by clicking **Verify** option in the menu.
6. Once the program compiled successfully, connect the Arduino board to the computer using USB cable.
7. After connecting, go to **Tools** ----**⌵** **Board** ---**⌵** Select **Arduino/Genuino Uno** option
8. After selecting board, go to **Tools** ----**⌵** **Port** ---**⌵** Select **Arduino Uno COM port 3** (name may appear differently for other computers).

****Note that this port option will be displayed only when board is connected to computer**

9. Now upload the program to the Arduino board by clicking **Upload** option.
10. Once the program uploaded successfully, observe the output on LCD.

PROGRAM:

RESULT:

Expt No.:11**Date:****ESP 32 SETUP & Controlling the GPIO pins in ESP32****AIM**

To setup ESP32 and GPIO Programming using Arduino IDE Programming.

SOFTWARES USED

ESP32 BOARD (Arduino IDE)

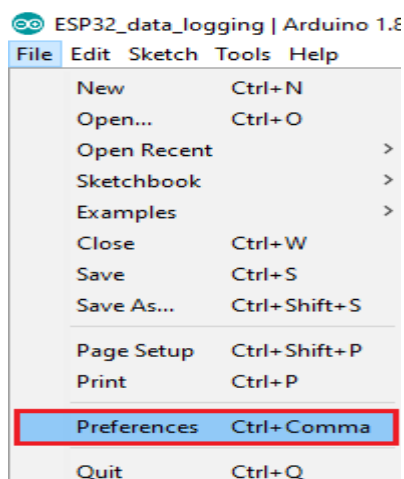
APPARATUS/ COMPONENTS REQUIRED

1. ESP32
2. Connecting Cable
3. DC 5V, 1A Power Adaptor
4. Connecting Wires
5. Bread Board

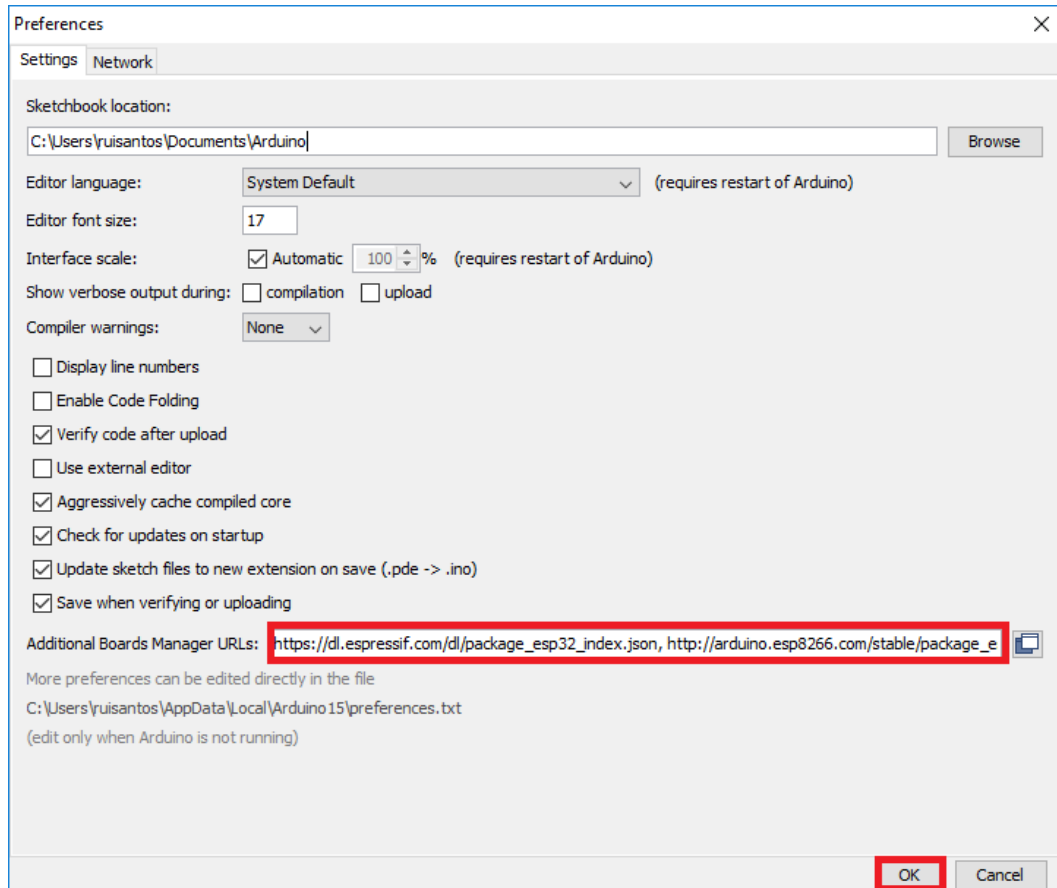
DESCRIPTION: ESP32 INSTALLATION PROCEDURE**Installing ESP32 Add-on in Arduino IDE**

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File> Preferences**



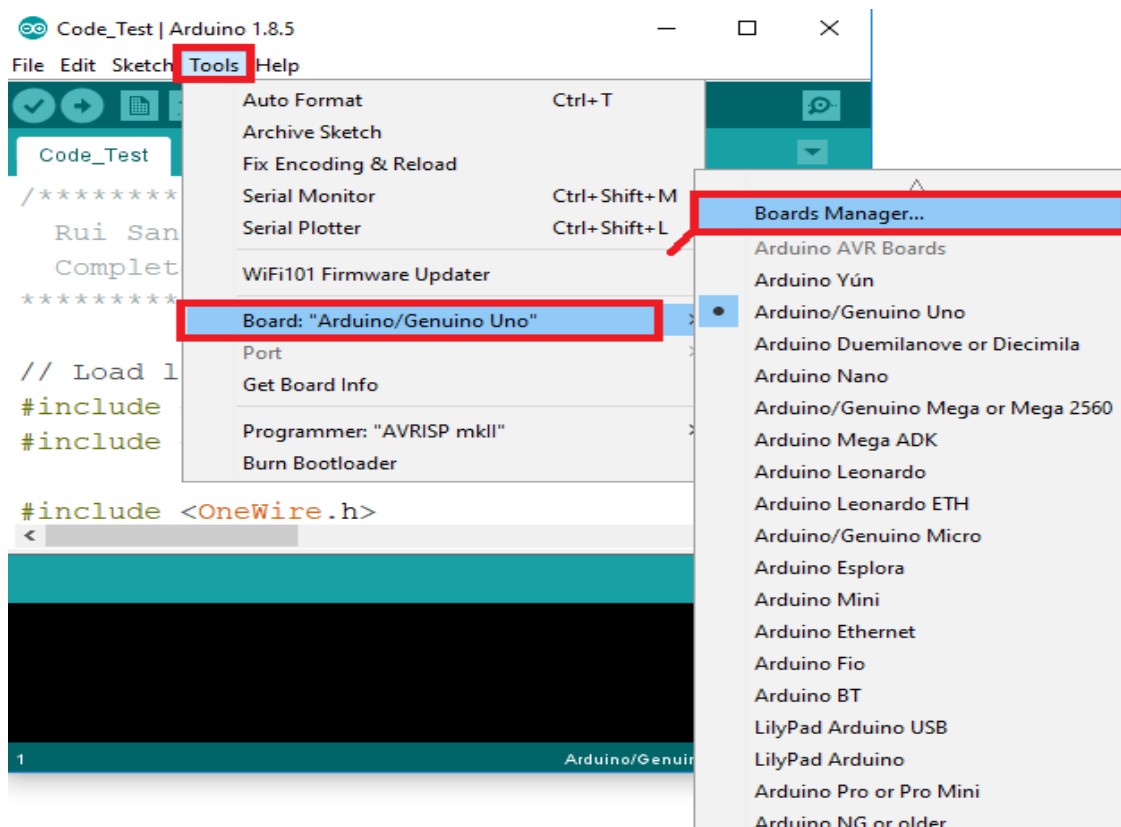
2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the “Additional Board Manager URLs” field as shown in the figure below. Then, click the “OK” button:



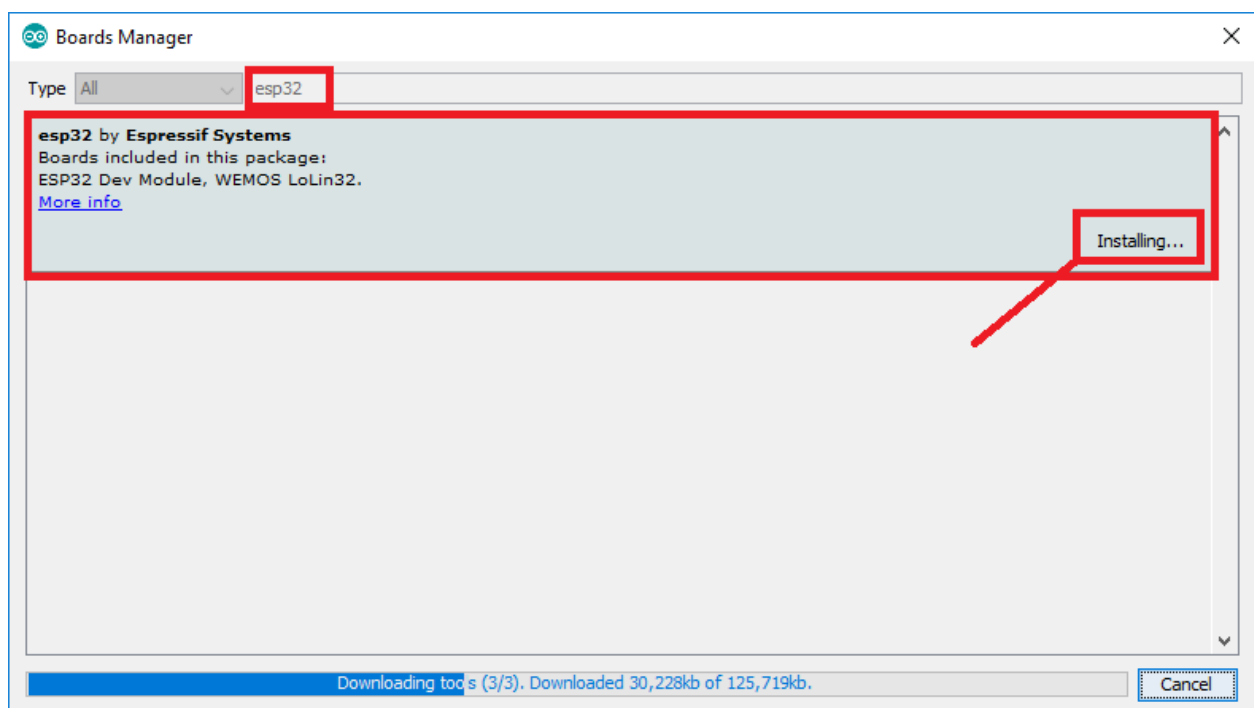
Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:

`https://dl.espressif.com/dl/package_esp32_index.json,`
`http://arduino.esp8266.com/stable/package_esp8266com_index.json`

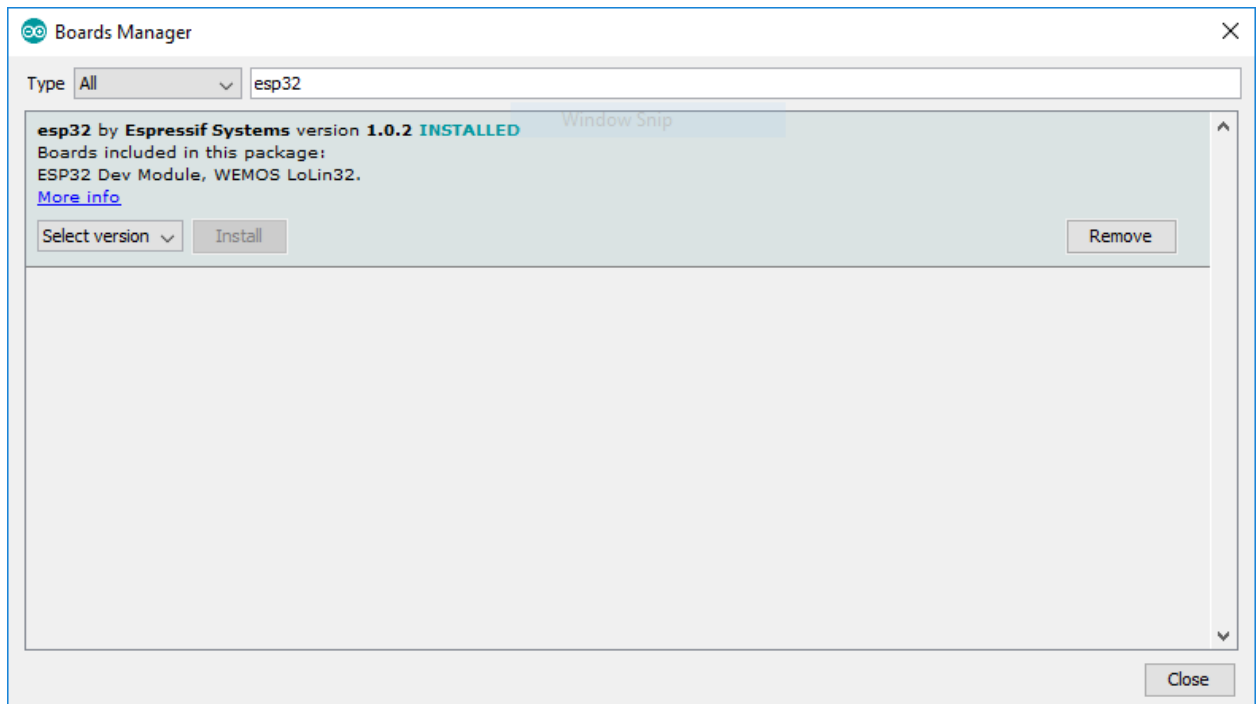
3. Open the Boards Manager. Go to **Tools > Board > Boards Manager...**



4. Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**”:



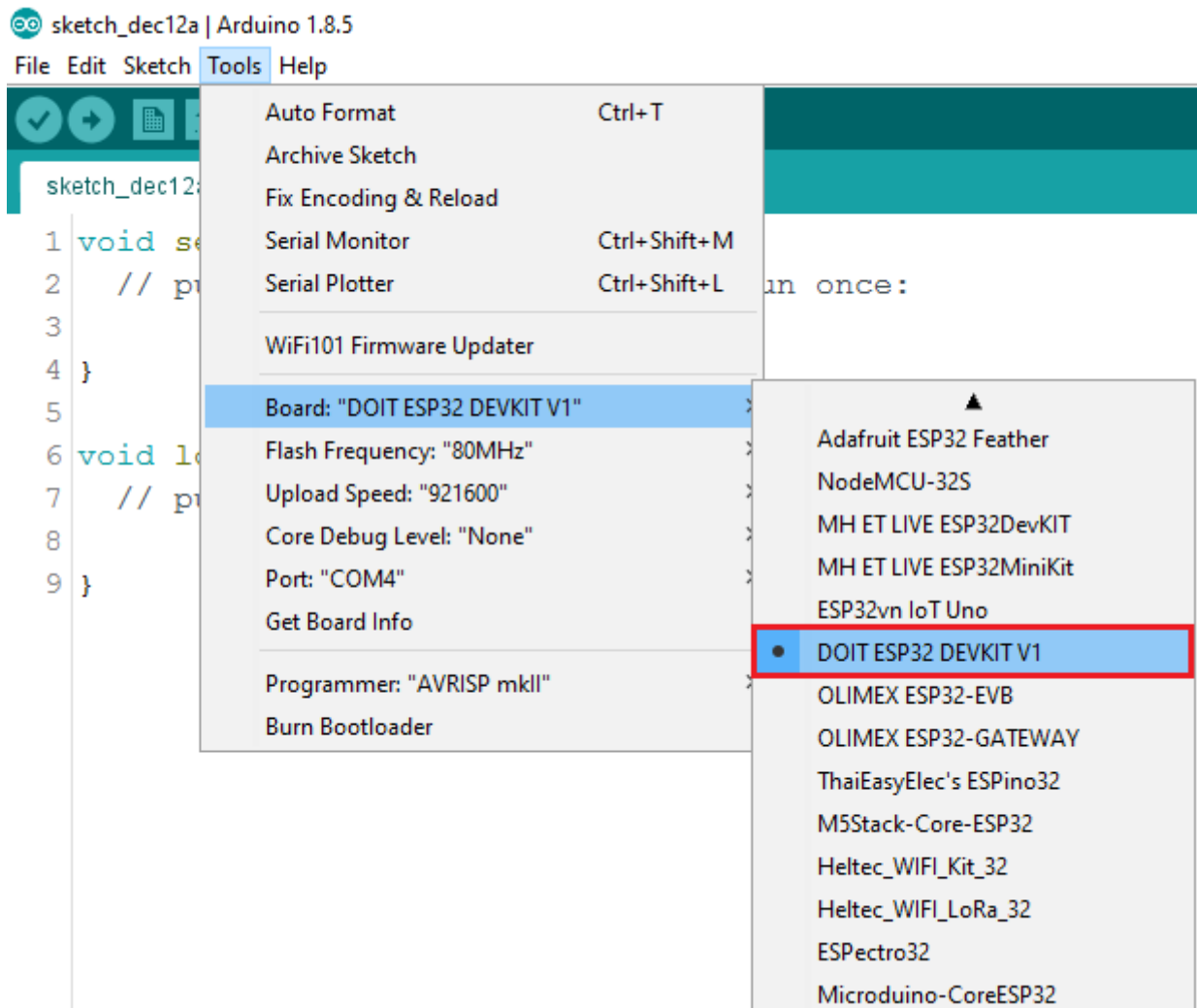
5. That's it. It should be installed after a few seconds.



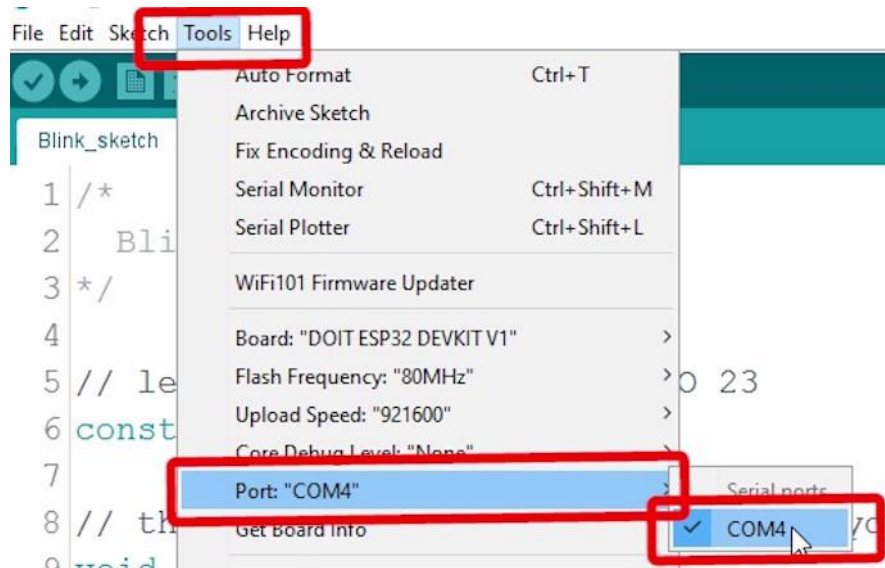
Testing the Installation

Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

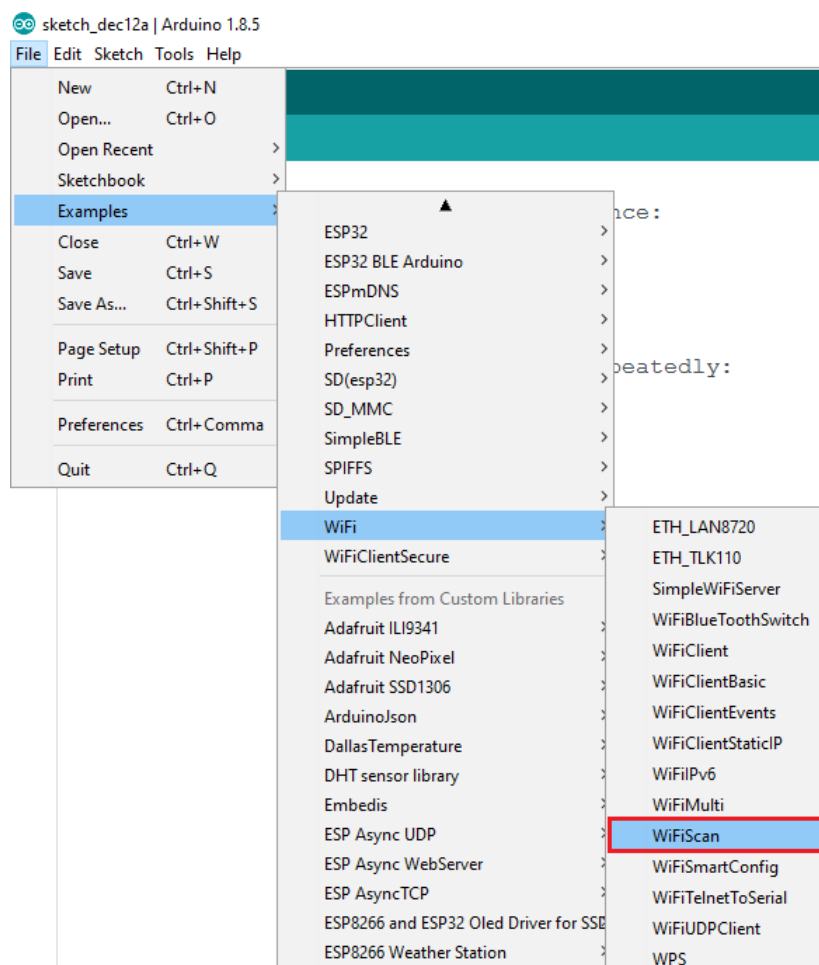
1. Select your Board in **Tools > Board** menu (in my case it's the **DOIT ESP32 DEVKIT V1**)



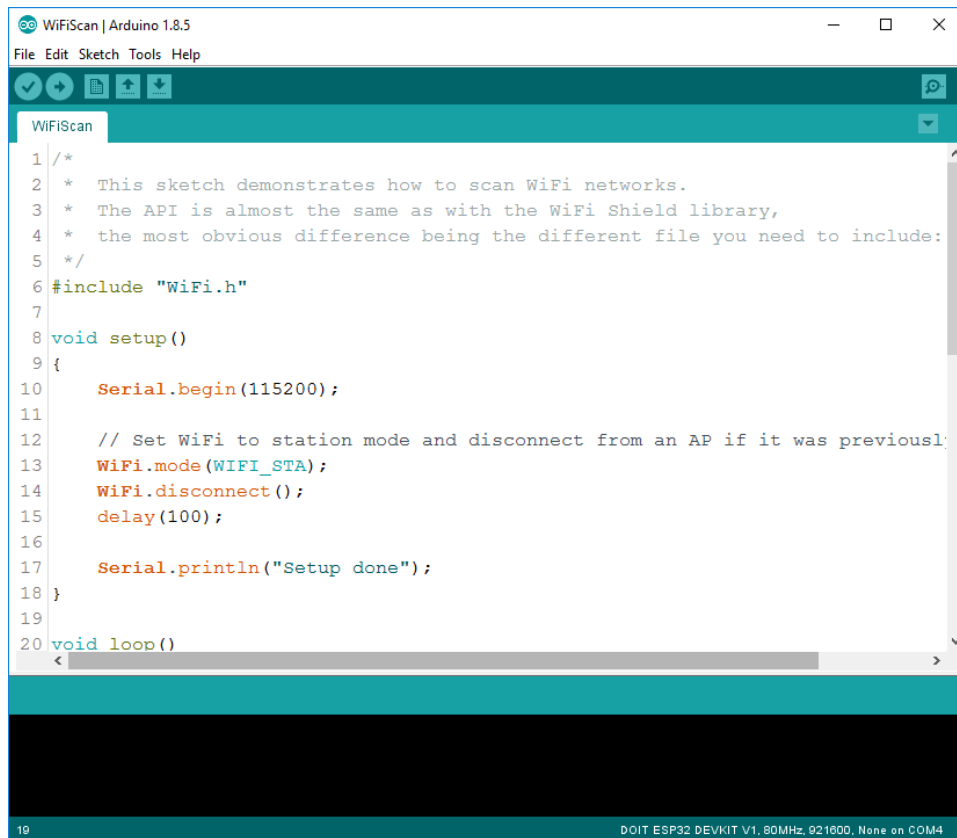
2. Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the [CP210x USB to UART Bridge VCP Drivers](#)):



3. Open the following example under **File > Examples > WiFi (ESP32) > WiFiScan**



4. A new sketch opens in your Arduino IDE:

The screenshot shows the Arduino IDE interface with a new sketch titled 'WiFiScan'. The code is as follows:

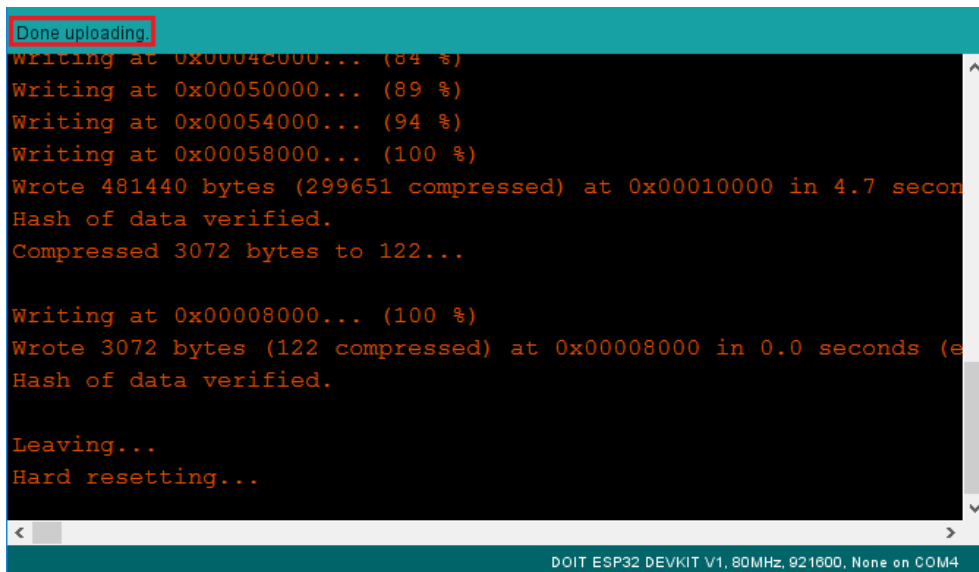
```
1 /*
2  * This sketch demonstrates how to scan WiFi networks.
3  * The API is almost the same as with the WiFi Shield library,
4  * the most obvious difference being the different file you need to include:
5  */
6 #include "WiFi.h"
7
8 void setup()
9 {
10     Serial.begin(115200);
11
12     // Set WiFi to station mode and disconnect from an AP if it was previously
13     WiFi.mode(WIFI_STA);
14     WiFi.disconnect();
15     delay(100);
16
17     Serial.println("Setup done");
18 }
19
20 void loop()
```

The status bar at the bottom indicates 'DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM4'.

5. Press the **Upload** button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.



6. If everything went as expected, you should see a “**Done uploading.**” message.



```

Done uploading.
Writing at 0x00040000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 seconds
Hash of data verified.
Compressed 3072 bytes to 122...

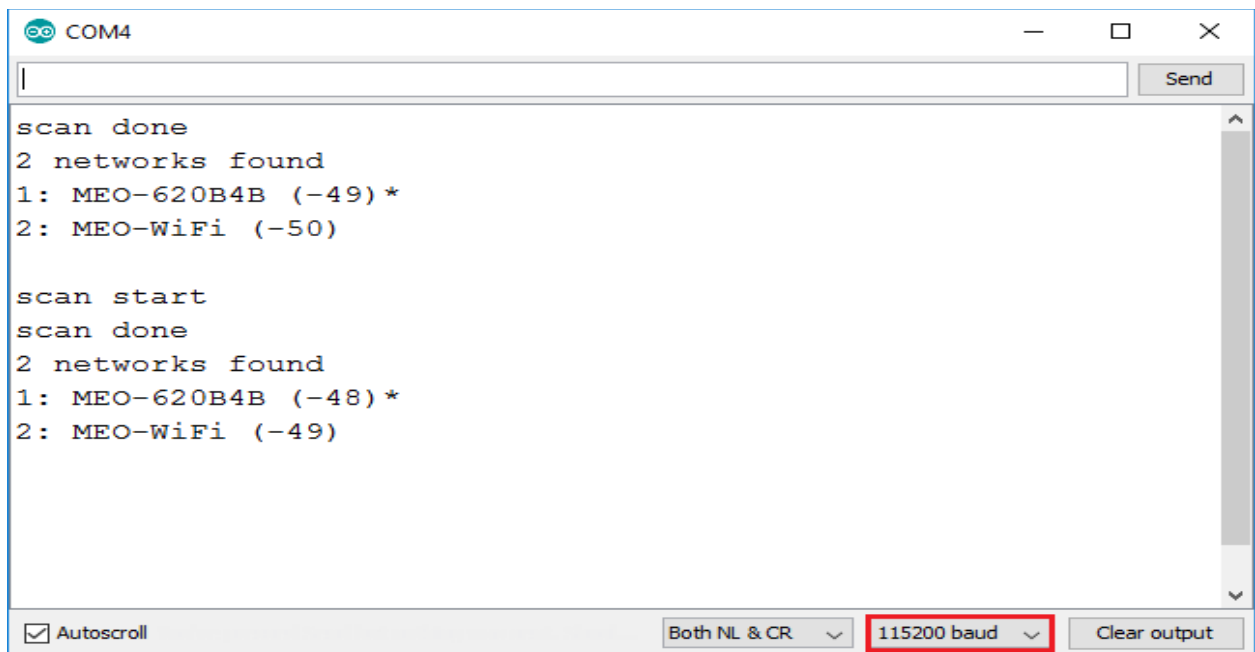
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (effective 115200 bps)
Hash of data verified.

Leaving...
Hard resetting...
  
```

7. Open the Arduino IDE Serial Monitor at a baud rate of 115200:



8. Press the ESP32 on-board **Enable** button and you should see the networks available near your ESP32:



```

COM4
scan done
2 networks found
1: MEO-620B4B (-49) *
2: MEO-WiFi (-50)

scan start
scan done
2 networks found
1: MEO-620B4B (-48) *
2: MEO-WiFi (-49)
  
```

Autoscroll Both NL & CR 115200 baud Clear output

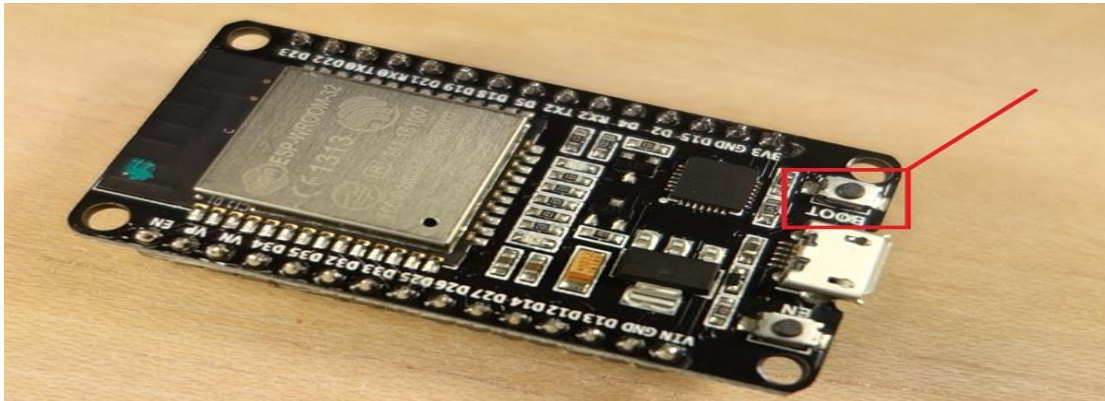
Troubleshooting

If you try to upload a new sketch to your ESP32 and you get this error message “A fatal

error occurred: *Failed to connect to ESP32: Timed out... Connecting...*“. It means that your ESP32 is not in flashing/uploading mode.

Having the right board name and COM port selected, follow these steps:

- Hold-down the “**BOOT**” button in your ESP32 board



- Press the “**Upload**” button in the Arduino IDE to upload your sketch:



- After you see the “**Connecting....**” message in your Arduino IDE, release the finger from the “**BOOT**” button:

```

Uploading...
Archiving built core (caching) in: C:\Users\NULISAN-1\AppData\Local\Temp\arduino_cache_959883\core\core_espressif_esp32_esp32doit-devkit-v1_Flash
Sketch uses 501366 bytes (38%) of program storage space. Maximum is 1310720 bytes.
Global variables use 37320 bytes (12%) of dynamic memory, leaving 257592 bytes for local variables. Maximum is 294912 bytes.
esptool.py v2.1
Connecting.....
Chip is ESP32D0WDQ6 (revision (unknown 0xa))
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Auto-detected flash size: 4MB
Compressed 8192 bytes to 47...

Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 8192.1 kbit/s)...
Hash of data verified.
Compressed 12304 bytes to 8126...

Writing at 0x00001000... (100 %)
  
```

- After that, you should see the “**Done uploading**” message

That’s it. Your ESP32 should have the new sketch running. Press the “**ENABLE**” button to restart the ESP32 and run the new uploaded sketch.

You’ll also have to repeat that button sequence every time you want to upload a new

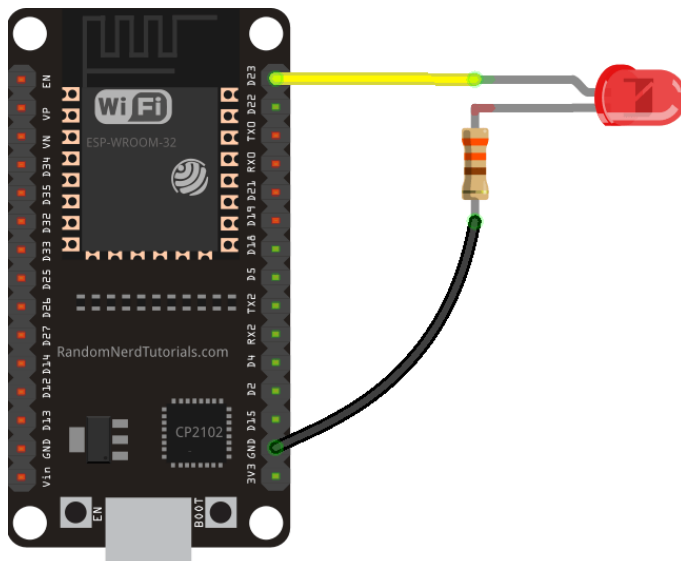
sketch. But if you want to solve this issue once for all without the need to press the **BOOT** button

PRE-LAB SESSION

1. What type of cable should be used for ESP32?
Ans.
2. Specify ROM size of ESP32?
Ans.
3. Mention the software which is to be installed in the laptop for ESP32 board?
Ans.
4. What is the operating voltage of the GPIO pins in ESP32?
Ans.

GPIO PROGRAMMING

How to write Air Quality measurement using ESP32 ?



1. Connect Sensors any GPIO Pin (GPIO 4) as per the Circuit Diagram
2. Connect 1K or 470Ω Resistor for current limiting Purpose.

CODE/ PROGRAM

PRECAUTIONS

1. Use insulation tools while you connect the Circuit.
2. Wear Good Insulating Shoes.
3. Avoid contacting circuits with wet hands or wet materials.
4. Maintain a work space clear of extraneous material such as books, papers, and clothes.
5. Be careful while connecting any actuators in 230V AC Mains.

RESULT

We successfully setup ESP32 along with LED blinking Program using ArduinoIDE .

INFERENCES/ANALYSIS**POST-LAB ASSIGNMENTS**

1. How many GPIO pins are available in the ESP32?
Ans.
2. Mention the number of ADCs available in ESP32?
Ans.
3. How many USB ports are available in the ESP32?
Ans.
4. Mention the number of micro usb ports available in ESP32?
Ans.
5. How many PWM pins are available in the ESP32?
Ans.
6. List out the GND pins available in the ESP32?
Ans.
7. List out the +3.3 V pins available in the ESP32.
Ans.
8. List out the +5V pins available in the ESP32
Ans.

ExptNo.:12**Date:****Controlling the GPIO pins in ESP32 using Touch Inputs****AIM**

To MPU6050 interface with ESP32 and WEBSERVER using Arduino IDE Programming.

SOFTWARES USED

ESP32 BOARD (Arduino IDE)

APPARATUS/ COMPONENTS REQUIRED

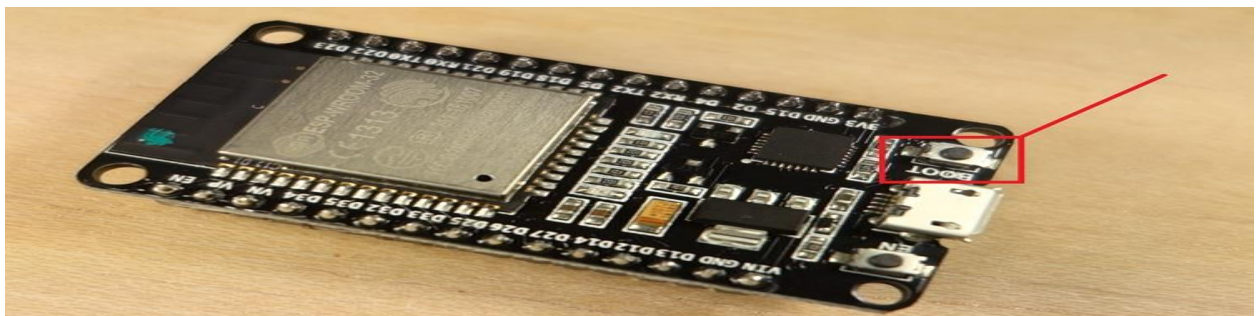
1. ESP32
2. Connecting Cable
3. DC 5V, 1A Power Adaptor

DESCRIPTION:**Troubleshooting**

If you try to upload a new sketch to your ESP32 and you get this error message “*A fatal error occurred: Failed to connect to ESP32: Timed out... Connecting...*“. It means that your ESP32 is not in flashing/uploading mode.

Having the right board name and COM port selected, follow these steps:

- Hold-down the “**BOOT**” button in your ESP32 board



- Press the “**Upload**” button in the Arduino IDE to upload your sketch:



- After you see the “**Connecting....**” message in your Arduino IDE, release the finger from the “**BOOT**” button:

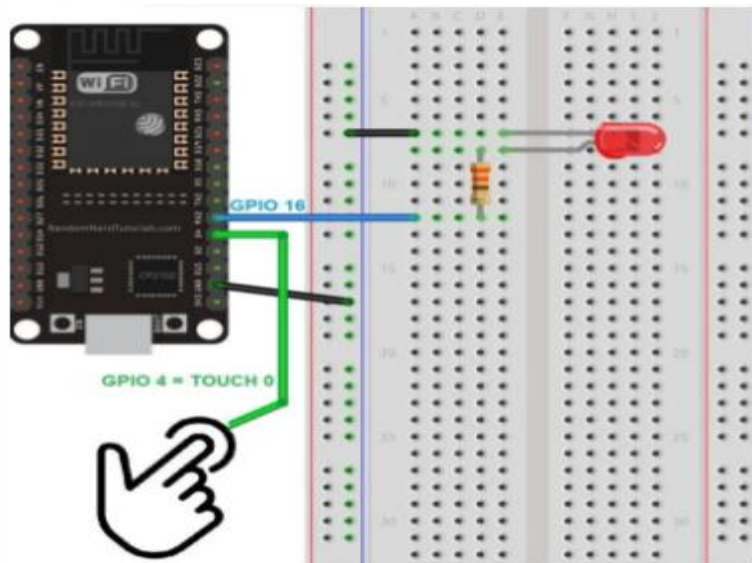
- After that, you should see the “**Done uploading**” message

That’s it. Your ESP32 should have the new sketch running. Press the “**ENABLE**” button to restart the ESP32 and run the new uploaded sketch.

You’ll also have to repeat that button sequence every time you want to upload a new sketch. But if you want to solve this issue once for all without the need to press the **BOOT** button

PRE-LAB SESSION

- What type of cable should be used for ESP32?
Ans.
- Specify ROM size of ESP32?
Ans.
- Mention the software which is to be installed in the laptop for ESP32 board?
Ans.
- What is the operating voltage of the GPIO pins in ESP32?
Ans.

Circuit Diagram:-**CODE/ PROGRAM**

PRECAUTIONS

1. Use insulation tools while you connect the Circuit.
2. Wear Good Insulating Shoes.
3. Avoid contacting circuits with wet hands or wet materials.
4. Maintain a work space clear of extraneous material such as books, papers, and clothes.
5. Be careful while connecting any actuators in 230V AC Mains.

INFERENCES/ANALYSIS

Results

ExptNo No.: 13**Date:****Interfacing Servo motor to ESP32****AIM**

To interface Servo Motor with ESP8266

SOFTWARES USED

ESP32 BOARD (Arduino IDE)

APPARATUS/ COMPONENTS REQUIRED

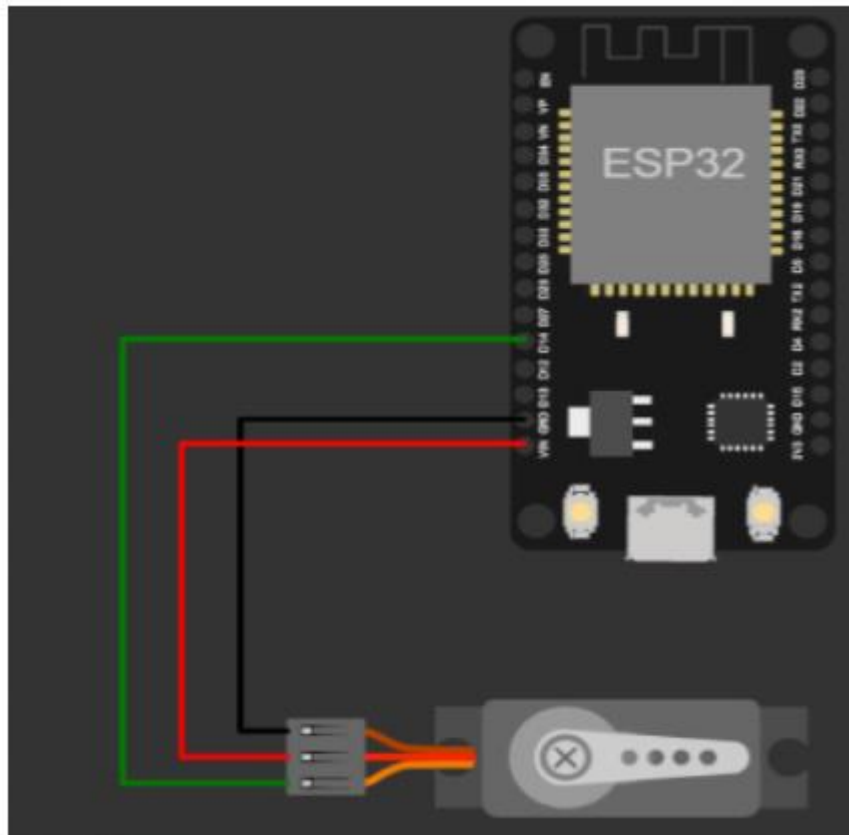
1. ESP32
2. Servo Motor
3. DC 5V, 1A Power Adaptor
4. Connecting Wires
5. Bread Board

THEORY:

A **servomotor** (or servo motor) is a simple electric motor, controlled with the help of servomechanism. If the motor as a controlled device, associated with servomechanism is DC motor, then it is commonly known as a **DC Servo Motor**. If AC operates the controlled motor, it is known as a AC Servo Motor. Low Power Operation Extends Battery-Backup Run Time. A servomotor is a linear actuator or rotary actuator that allows for precise control of linear or angular position, acceleration, and velocity. It consists of a motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

There are some special types of applications of an electric motor where the rotation of the motor is required for just a certain angle. For these applications, we require some special types of motor with some special arrangement which makes the motor rotate a certain angle for a given electrical input (signal). The servo motor is usually a simple DC motor controlled for specific angular rotation with the help of additional servomechanism (a typical closed-loop feedback control system). Nowadays, servo systems are used widely in industrial applications. Servo motor applications are also commonly seen in remote-controlled toy cars for controlling the direction of motion, and it is also very widely used as the motor which moves the tray of a CD or DVD player. Besides these, there are hundreds of servo motor applications we see in our daily life.

HARDWARE CONNECTION:



SG90 Servo Motor Pinouts	
Terminal Color	Brief Description
Red	VCC Connected to 3.5V to 5V
Brown	Ground
Orange/Yellow	Control Signal or PWM Signal to be applied on this terminal

CODE/PROGRAM:-

```
#define COUNT_LOW 0
#define COUNT_HIGH 8888

#define TIMER_WIDTH 16
```

```
#include "esp32-hal-ledc.h"

void setup() {
  Serial.begin(9600);
  ledcSetup(1, 50, TIMER_WIDTH); // channel 1, 50 Hz, 16-bit width
  ledcAttachPin(14, 1); // GPIO 14 assigned to channel 1
}

void loop() {
  for (int i=COUNT_LOW ; i < COUNT_HIGH ; i=i+100)
  {
    ledcWrite(1, i);    // sweep servo 1
    Serial.print(1,i);
    delay(50);
  }

  for (int i=COUNT_HIGH ; i > COUNT_LOW ; i=i-100)
  {
    ledcWrite(1, i);    // sweep servo 1
    Serial.print(1,i);
    delay(50);
  }
}
```

PRECAUTIONS

1. Use insulation tools while you connect the Circuit.
2. Wear Good Insulating Shoes.
3. Avoid contacting circuits with wet hands or wet materials.
4. Maintain a work space clear of extraneous material such as books, papers, and clothes.
5. Be careful while connecting any actuators in 230V AC Mains.

RESULT

INFERENCES/ANALYSIS