

**University of Global Village (UGV), Barishal**  
**Dept. of Electrical and Electronic Engineering (EEE)**



## Lab Manual

# Electronics Shop Practice

**Noor Md Shahriar**

BSc in EEE, [RUET](#)

[Senior Lecturer](#)

Co-chairman, Dept. of EEE

[University of Global Village \(UGV\)](#)

[874/322, C&B Road, Barishal, Bangladesh.](#)



Contact: +8801743500587



[Facebook](#)



[LinkedIn](#)



[Twitter](#)



## Table of Contents

Course Rationale .....	5
Course Objectives .....	5
Course Learning Outcomes (CLOs).....	5
Course Content .....	5
Course Schedule.....	6
PREFERRED TOOLS .....	6
TEXT/REFERENCE BOOKS .....	7
Experiment no: 01 .....	8
Experiment no: 02 .....	19
Experiment no: 03 .....	28
Experiment no: 04.....	32
Experiment no: 05 .....	35
Experiment no: 06 .....	40
Experiment no: 07 .....	46
Experiment no: 08.....	48
Experiment no: 09 .....	52

## Course Rationale

This hands-on course introduces students to microcontroller programming and sensor interfacing using Arduino UNO. It covers fundamental concepts of embedded systems, including input/output interfacing, sensor integration, motor control, and data visualization. The course emphasizes practical skills in prototyping circuits, debugging, and writing efficient code for real-world applications.

## Course Objectives:

- Understand the architecture of Arduino UNO and its programming environment.
- Develop skills in circuit design, soldering, and prototyping.
- Interface sensors (LDR, ultrasonic, IR, temperature) and actuators (motors, servos, buzzers) with Arduino.
- Learn to read sensor data and display it on LCD/serial monitor.
- Build mini-projects integrating multiple components.

## Course Learning Outcomes (CLOs):

After the successful completion of this course, students are expected to be able to:

CLO1	<b>Understand</b> and <b>operate</b> microcontroller systems, including the 8086-trainer board, EMU8086 emulator, and Keil C51 Evaluation Kit.
CLO2	<b>Develop</b> and <b>debug</b> assembly-level programming for the 8086 microprocessor and 8051 microcontroller.
CLO3	<b>Implement</b> interfacing techniques for various sensors, actuators, and displays with Arduino.
CLO4	<b>Design</b> and <b>test</b> microcontroller-based applications, including Programmable Peripheral Interfaces.
CLO5	<b>Analyze</b> and <b>troubleshoot</b> microcontroller-based systems for real-world applications.

## Course Content:

Sl. No.	Topic & Details	Class Hours	CLO Mapping
1	Soldering & Desoldering: Techniques and safety.	2	CLO2
2	Arduino UNO & IDE: Pinout, functions, and basic sketches.	2	CLO1
3	LED Blink & Keypad Interfacing: Digital I/O operations.	3	CLO3
4	Potentiometer & Buzzer: Analog input and sound output.	3	CLO3
5	Servo & DC Motors: PWM control with motor drivers (L298N).	3	CLO5

6	Sensors: LDR, Ultrasonic (HC-SR04), Temperature (LM35).	4	CLO4
7	LCD Interfacing: 16x2 display with I2C module.	2	CLO6
8	IR Sensors: Active/Passive IR obstacle detection.	2	CLO4
9	Stepper Motor: Control using ULN2003 driver.	3	CLO5
10	Mini-Project: Integrated system (e.g., smart fan with temperature control).	4	CLO6

## Course Schedule

Week	Topic & Detail	Teaching-Learning Strategy	Assessment Strategy	CLO Mapping
1	Soldering practice + Arduino UNO introduction.	Demo + Hands-on	Lab Journal Check	CLO1, CLO2
2	Arduino IDE setup + LED blink.	Coding demo + Simulation	Code Submission	CLO1, CLO3
3	4x4 keypad interfacing.	Circuit building + Debugging	Practical Test	CLO3
4	Potentiometer + Serial Monitor + Buzzer.	Analog input labs	Quiz + Output Demo	CLO3
5	Servo motor + DC motor control.	PWM explanation + Motor driver lab	Lab Report	CLO5
6	LDR + Ultrasonic sensor.	Data logging + Threshold analysis	Sensor Data Submission	CLO4
7	LM35 Temperature sensor + LCD display.	Calibration + I2C LCD	Project Draft	CLO4, CLO6
8	IR sensors (Active/Passive).	Obstacle detection lab	Circuit Viva	CLO4
9	Stepper motor interfacing.	Driver wiring + Half/Full step	Motor Control Demo	CLO5
10	Mini-Project Submission.	Guided prototyping	Final Evaluation (Demo + Report)	CLO6
11	Capstone Project Work: End-to- end microcontroller application development	Group work, mentoring sessions	Weekly progress review	CLO 5
12	Final Project Presentation: Showcase of microcontroller- based applications	Presentation and evaluation by a panel	Evaluation of final project	CLO 5

## PREFERRED TOOLS

- Arduino IDE,
- Proteus 8 Professional (or higher versions),
- TinkerCad,
- MDA-8086 Kit.

## **TEXT/REFERENCE BOOKS**

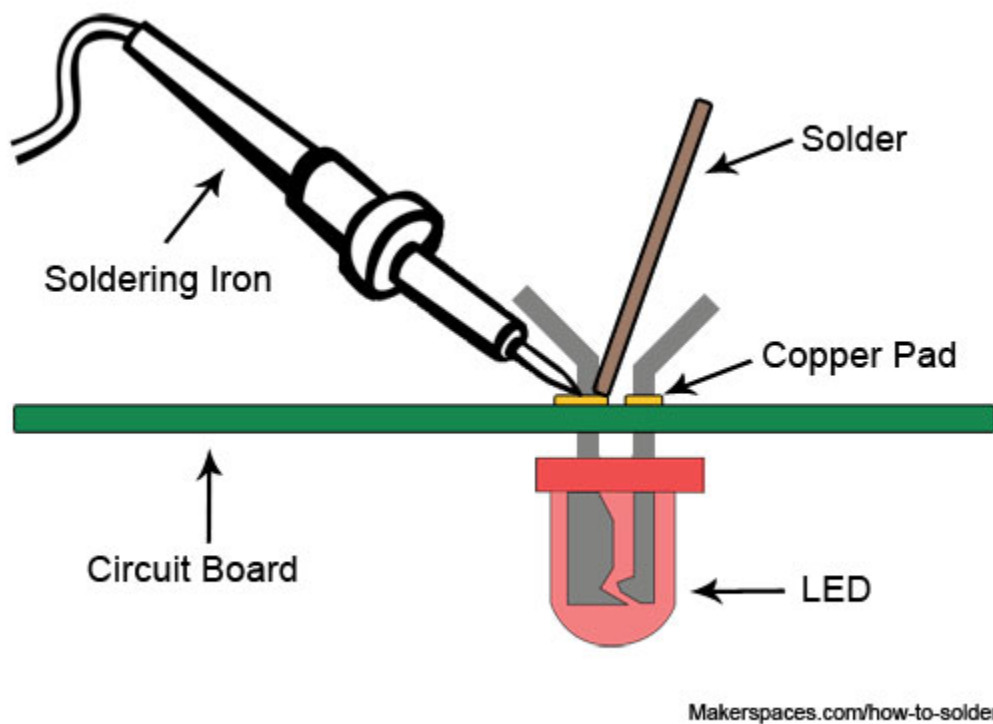
- a) Microprocessors and Microcomputer-Based System Design by Mohamed Rafiquzzaman.
- b) The 8051 Microcontroller and Embedded systems: using Assembly and C by Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. Mckinla.
- c) The AVR Microcontroller and Embedded Systems Using Assembly and C: Using Arduino Uno and Atmel Studio by Sepehr Naimi, Sarmad Naimi, Muhammad Ali Mazidi (2nd ed.)
- d) Microprocessor and Interfacing Programming and Hardware by Douglas V. Hall

## Experiment no: 01

### Experiment Title: Experiment on Soldering & Desoldering

#### Theory:

Learning how to solder w/ proper soldering techniques is a fundamental skill every maker should master. If you were to take apart any electronic device that contains a circuit board, you'll see the components are attached using soldering techniques. Soldering is the process of joining two or more electronic parts together by melting solder around the connection. Solder is a metal alloy and when it cools it creates a strong electrical bond between the parts. Even though soldering can create a permanent connection, it can also be reversed using a Desoldering tool as described below.



#### Soldering Tools

##### 1. Soldering Iron

A [soldering iron](#) is a hand tool that plugs into a standard 120v AC outlet and heats up in order to melt solder around electrical connections. This is one of the most important tools used in soldering and it can come in a few variations such as pen or gun form. For beginners, it's recommended that you use the pen style soldering iron in the 15W to 30W range. Most soldering irons have interchangeable tips that can be used for different soldering applications. Be very cautious when using any type of soldering iron because it can heat up to 896° F which is extremely hot.



## 2. Soldering Station

A [soldering station](#) is a more advanced version of the basic standalone soldering pen. If you are going to be doing a lot of soldering, these are great to have as they offer more flexibility and control. The main benefit of a soldering station is the ability to precisely adjust the temperature of the soldering iron which is great for a range of projects. These stations can also create a safer workspace as some include advanced temperature sensors, alert settings and even password protection for safety.



## 3. Soldering Iron Tips

At the end of most soldering irons is an interchangeable part known as a soldering tip. There are many variations of this tip and they come in a wide variety of shapes and sizes. Each tip is used for a specific purpose and offers a distinct advantage over another. The most common tips you will use in electronics projects are the [conical tip](#) and the [chisel tip](#).

**Conical Tip** – Used in precision electronics soldering because of the fine tip. Because of its pointed end, it's able to deliver heat to smaller areas without affecting its surroundings.

**Chisel Tip** – This tip is well-suited to soldering wires or other larger components because of its broad flat tip.



#### 4. Brass or Conventional Sponge

Using a sponge will help to keep the soldering iron tip clean by removing the oxidation that forms. Tips with oxidation will tend to turn black and not accept solder as it did when it was new. You could use a conventional wet sponge but this tends to shorten the lifespan of the tip due to expansion and contraction. Also, a wet sponge will drop the temperature of the tip temporarily when wiped. A better alternative is to use a [brass sponge](#) as shown on the left.



#### 5. Soldering Iron Stand

A [soldering iron stand](#) is very basic but very useful and handy to have. This stand helps prevent the hot iron tip from coming in contact with flammable materials or causing accidental injury to your hand. Most soldering stations come with this built in and also include a sponge or brass sponge for cleaning the tip.





## 6. Solder

Solder is a metal alloy material that is melted to create a permanent bond between electrical parts. It comes in both lead and lead-free variations with diameters of .032" and .062" being the most common. Inside the solder core is a material known as flux which helps improve electrical contact and its mechanical strength.

For electronics soldering, the most commonly used type is [lead-free rosin core solder](#). This type of solder is usually made up of a Tin/Copper alloy. You can also use leaded 60/40 (60% tin, 40% lead) rosin core solder but it's becoming less popular due to health concerns. If you do use lead solder, make sure you have proper ventilation and that you wash your hands after use.



When buying solder, make sure NOT to use acid core solder as this will damage your circuits and components. Acid core solder is sold at home improvement stores and is mainly used for plumbing and metal working.

As mentioned earlier, solder does come in a few different diameters. The thicker diameter solder (.062") is good for soldering larger joints more quickly but it can make soldering smaller joints difficult. For this reason, it's always a good idea to have both sizes on hand for your different projects.

Helping Hand (Third Hand)

A [helping hand](#) is a device that has 2 or more alligator clips and sometimes a magnifying glass/light attached. These clips will assist you by holding the items you are trying to solder while you use the soldering iron and solder. A very helpful tool to have in your makerspace.

**Procedure:**

- Soldering irons can reach temperatures of 800' F so it's very important to know where your iron is at all times. We always recommend you use a soldering iron stand to help prevent accidental burns or damage.



Make sure you are soldering in a well-ventilated area. When solder is heated, there are fumes released that are harmful to your eyes and lungs. It's recommended to use a [fume extractor](#) which is a fan with a charcoal filter that absorbs the harmful solder smoke you can visit sites like [Integrated Air Systems](#) for air filtration systems.

It's always a good idea to wear protective eye wear in case of accidental splashes of hot solder.

Lastly, make sure to wash your hands when done soldering especially if using lead solder.

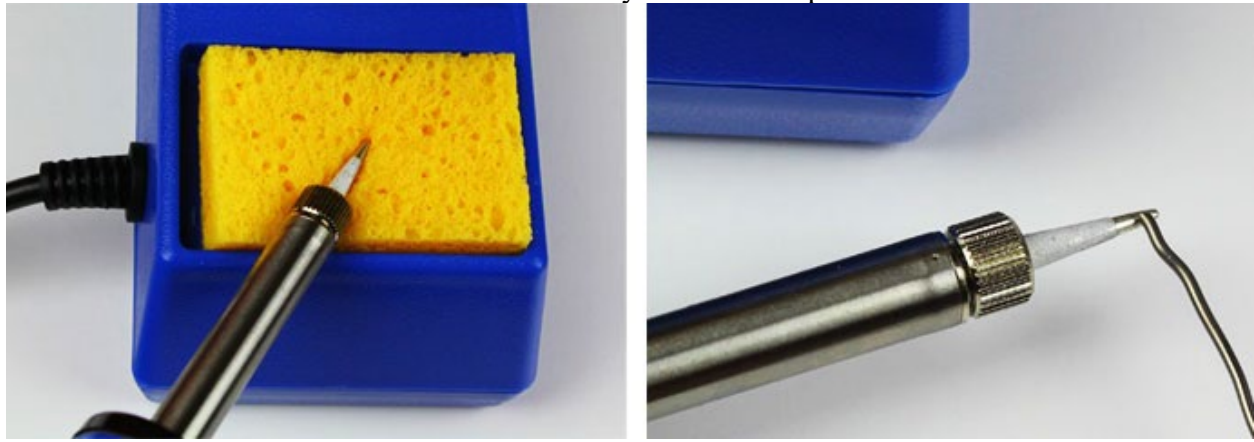
- Tinning The Tip:

**Step 1:** Begin by making sure the tip is attached to the iron and screwed tightly in place.

**Step 2:** Turn on your soldering iron and let it heat up. If you have a soldering station with an adjustable temp control, set it to 400° C/ 752° F.

**Step 3:** Wipe the tip of the soldering iron on a damp wet sponge to clean it. Wait a few seconds to let the tip heat up again before proceeding to step 4.

**Step 4:** Hold the soldering iron in one hand and solder in the other. Touch the solder to the tip of the iron and make sure the solder flows evenly around the tip.

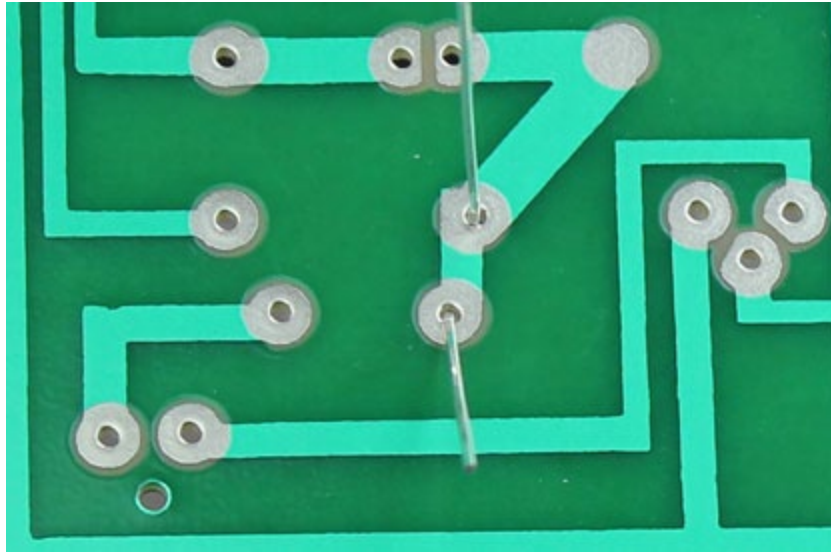


You should tin the tip of your iron before and after each soldering session to extend its life. Eventually, every tip will wear out and will need replacing when it becomes rough or pitted.

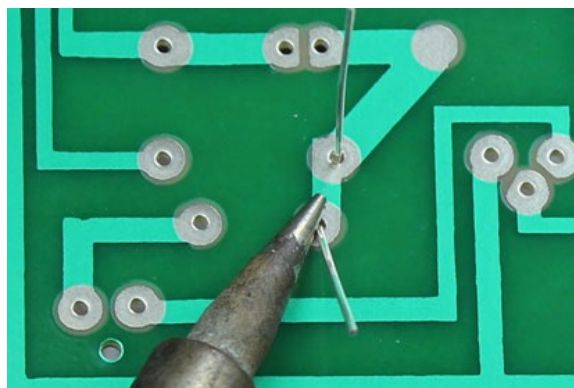
- How To Solder

To better explain how to solder, we're going to demonstrate it with a real-world application. In this example, we're going to solder an LED to a circuit board.

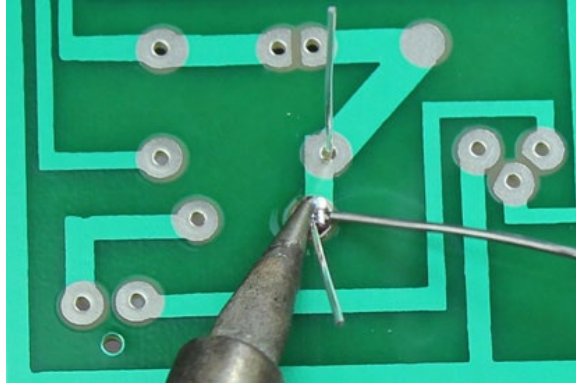
**Step 1: Mount the Component** – Begin by inserting the leads of the LED into the holes of the circuit board. Flip the board over and bend the leads outward at a 45° angle. This will help the component make a better connection with the copper pad and prevent it from falling out while soldering.



**Step 2: Heat the Joint** – Turn your soldering iron on and if it has an adjustable heat control, set it to 400°C. At this point, touch the tip of the iron to the copper pad and the resistor lead at the same time. You need to hold the soldering iron in place for 3-4 seconds in order to heat the pad and the lead.

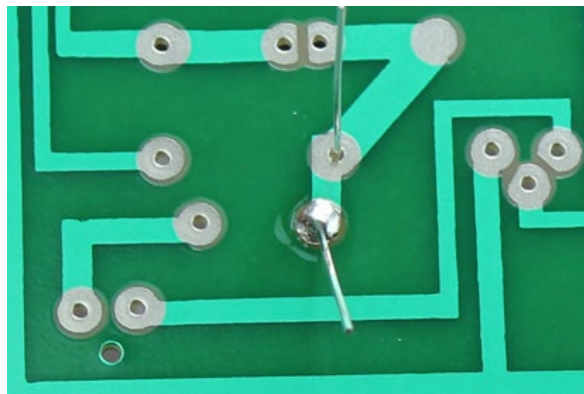


**Step 3: Apply Solder to Joint** – Continue holding the soldering iron on the copper pad and the lead and touch your solder to the joint. **IMPORTANT** – Don't touch the solder directly to the tip of the iron. You want the joint to be hot enough to melt the solder when it's touched. If the joint is too cold, it will form a bad connection.



**Step 4: Snip the Leads** – Remove the soldering iron and let the solder cool down naturally. Don't blow on the solder as this will cause a bad joint. Once cool, you can snip the extra wire from leads.

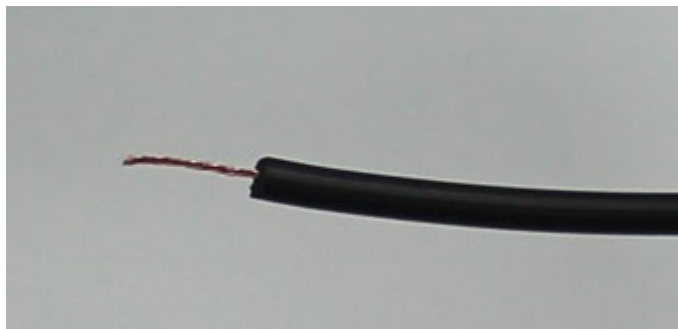
A proper solder joint is smooth, shiny and looks like a volcano or cone shape. You want just enough solder to cover the entire joint but not too much so it becomes a ball or spills to a nearby lead or joint.



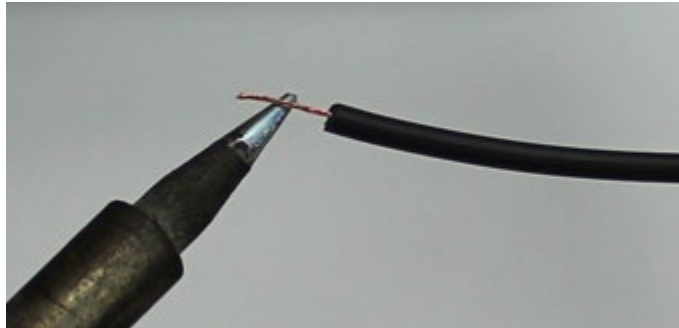
- How To Solder Wires

Now it's time to show you how to solder wires together. For this process, it's recommended to use helping hands or other type of clamp device.

Begin by removing the insulation from the ends of both wires you are soldering together. If the wire is stranded, twist the strands together with your fingers.



Make sure your soldering iron is fully heated and touch the tip to the end of one of the wires. Hold it on the wire for 3-4 seconds.



Keep the iron in place and touch the solder to the wire until it's fully coated. Repeat this process on the other wire.



Hold the two tinned wires on top of each other and touch the soldering iron to both wires. This process should melt the solder and coat both wires evenly.



Remove the soldering iron and wait a few seconds to let the soldered connection cool and harden. Use heat shrink to cover the connection.





- Desoldering

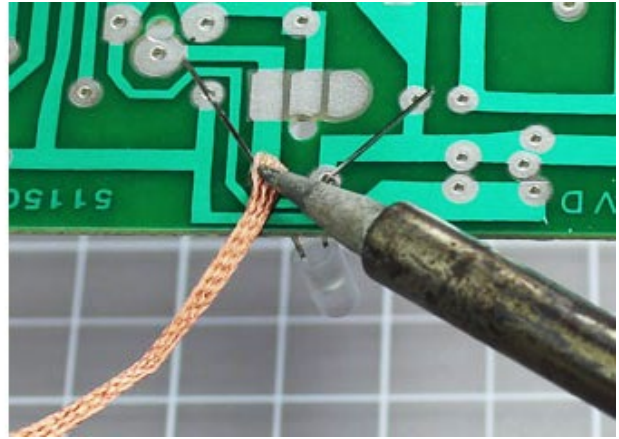
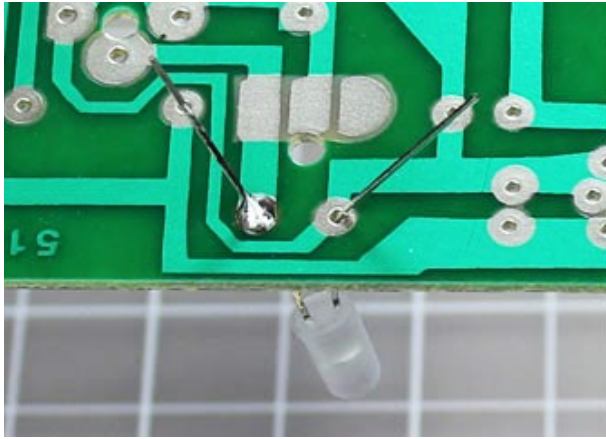
The good thing about using solder is the fact that it can be removed easily in a technique known as Desoldering. This comes in handy if you need to remove a component or make a correction to your electronic circuit.

To desolder a joint, you will need solder wick which is also known as [Desoldering braid](#).



**Step 1** – Place a piece of the Desoldering braid on top of the joint/solder you want removed.

**Step 2** – Heat your soldering iron and touch the tip to the top of the braid. This will heat the solder below which will then be absorbed into the Desoldering braid. You can now remove the braid to see the solder has been extracted and removed. Be careful touching the braid when you are heating it because it will get hot.



**Optional** – If you have a lot of solder, you want removed, you may want to use a device called a solder sucker. This is a handheld mechanical vacuum that sucks up hot solder with a press of a button.

To use, press the plunger down at the end of the solder sucker. Heat the joint with your soldering iron and place the tip of the solder sucker over the hot solder. Press the release button to suck up the liquid solder. In order to empty the solder sucker, press down on the plunger.



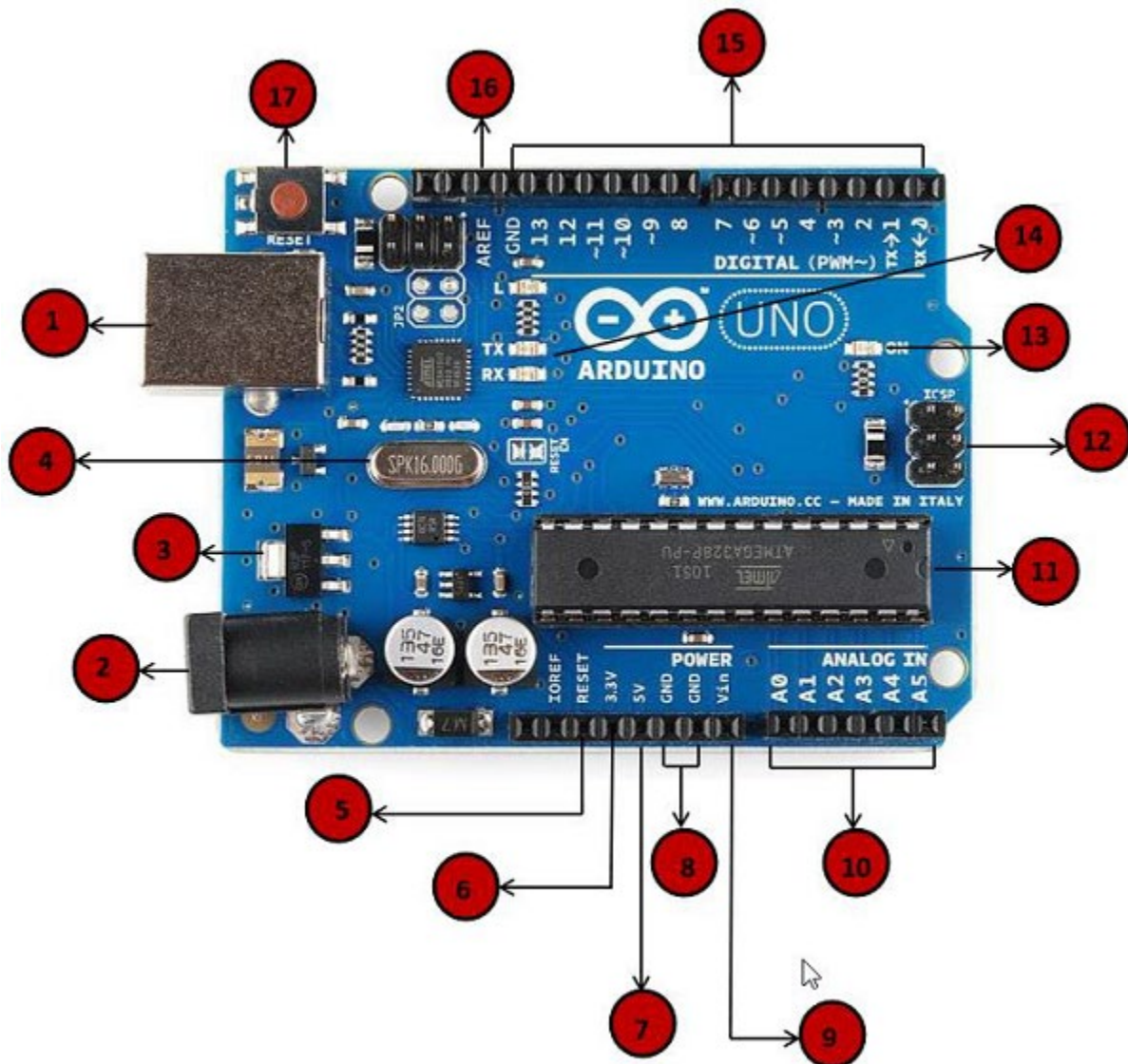



## Experiment no: 02









### Experiment Title: Introduction to Arduino UNO Hardware & Arduino IDE Software





#### Theory:

In this chapter, we will learn about the different components on the Arduino board. We will study the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.



	<p><b>Power USB</b></p> <p>Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).</p>
---	---

	<p><b>Power (Barrel Jack)</b></p> <p>Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).</p>
	<p><b>Voltage Regulator</b></p> <p>The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.</p>
	<p><b>Crystal Oscillator</b></p> <p>The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.</p>
	<p><b>Arduino Reset</b></p> <p>You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).</p>
	<p><b>Pins (3.3, 5, GND, Vin)</b></p> <p>3.3V (6) – Supply 3.3 output volt  5V (7) – Supply 5 output volt</p> <p>Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.</p> <p>GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.</p> <p>Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.</p>
	<p><b>Analog pins</b></p> <p>The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>
	<p><b>Main microcontroller</b></p> <p>Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.</p>
	<p><b>ICSP pin</b></p> <p>Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.</p>

	<p><b>Power LED indicator</b> This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
	<p><b>TX and RX LEDs</b> On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
	<p><b>Digital I/O</b> The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.</p>
	<p><b>AREF</b> AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

## Arduino - Installation

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

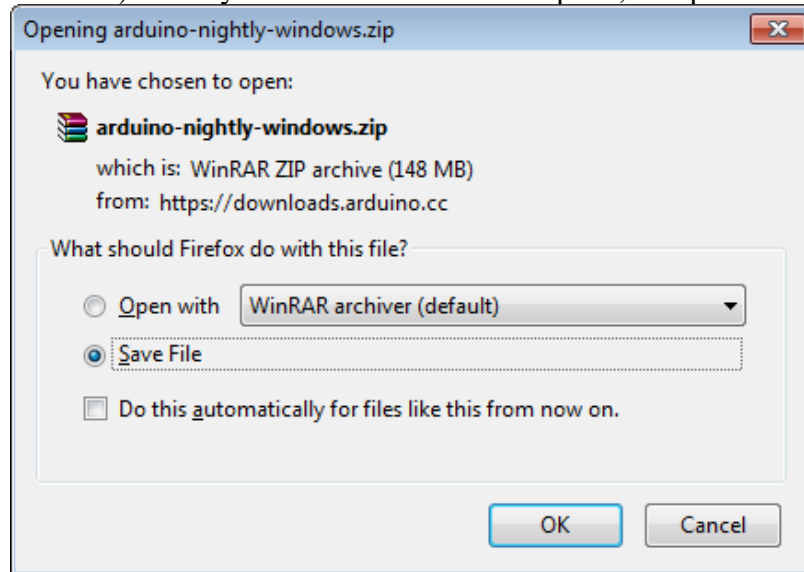
In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

- **Step 1** – First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



- Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the [Download page](#) on the Arduino Official website. You must select your software which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



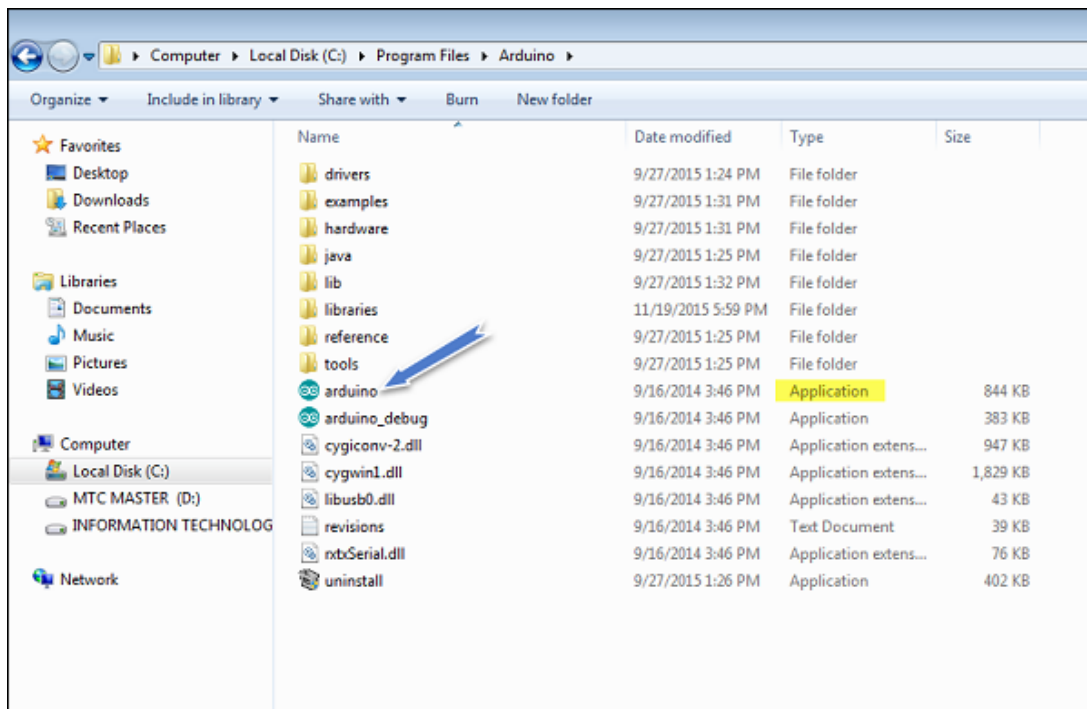
- Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 – Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



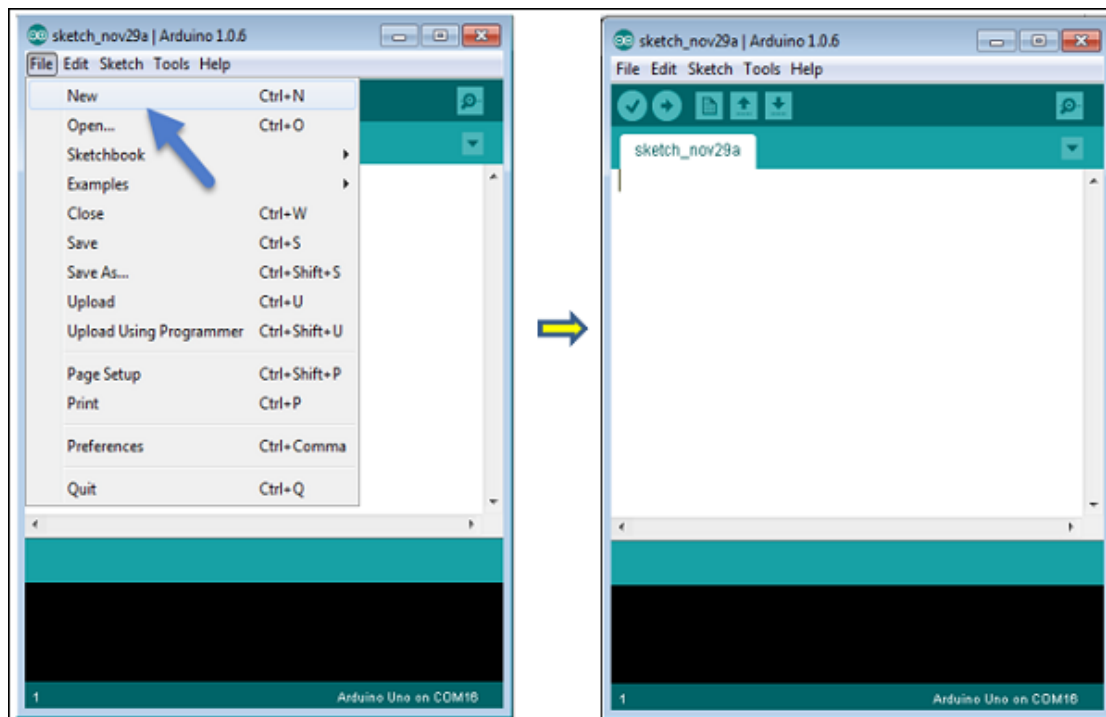
- Step 5 – Open your first project.

Once the software starts, you have two options –

Create a new project.

Open an existing project example.

To create a new project, select File → **New**.



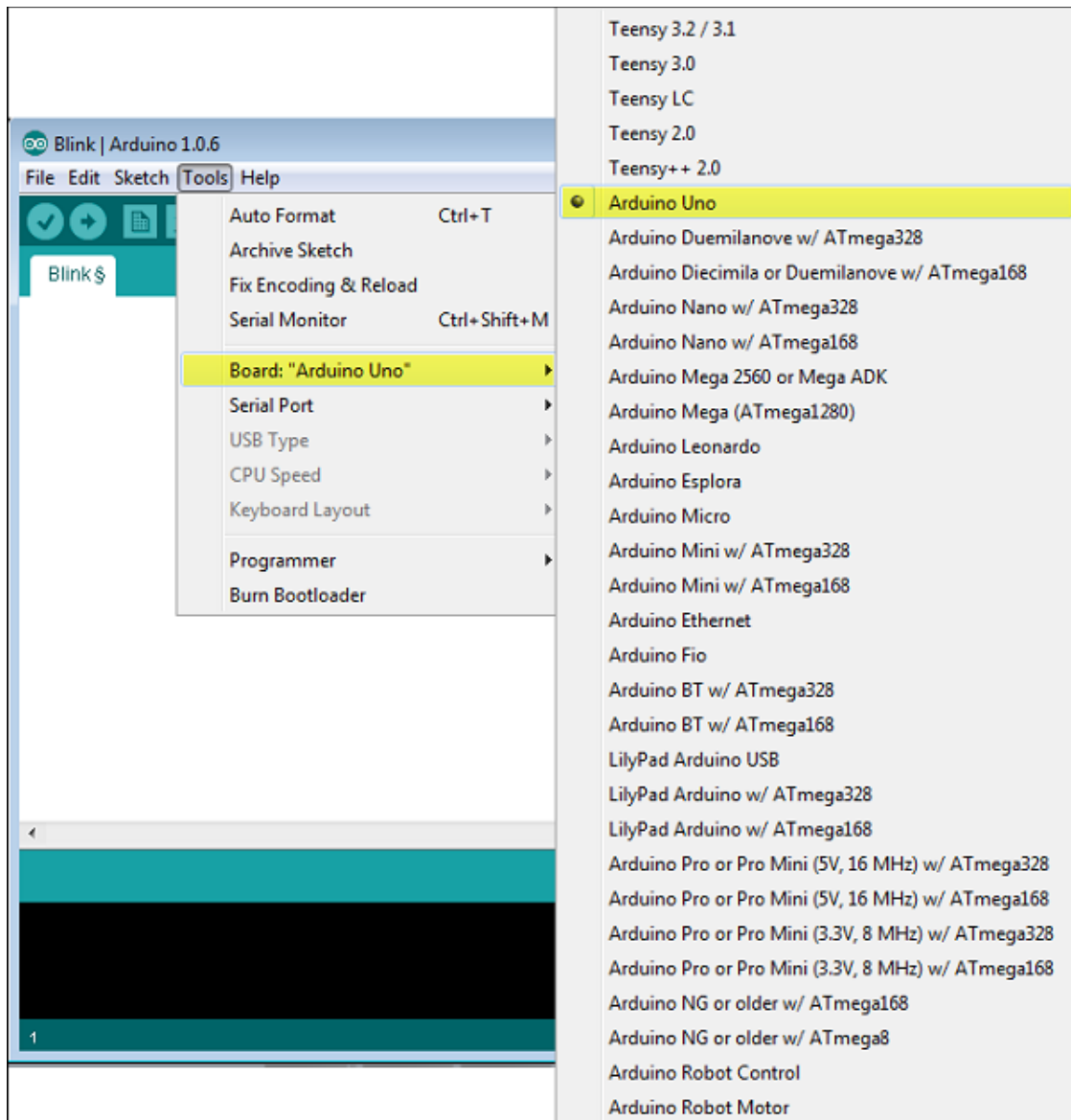
To open an existing project example, select File → Example → Basics → Blink.

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

- Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

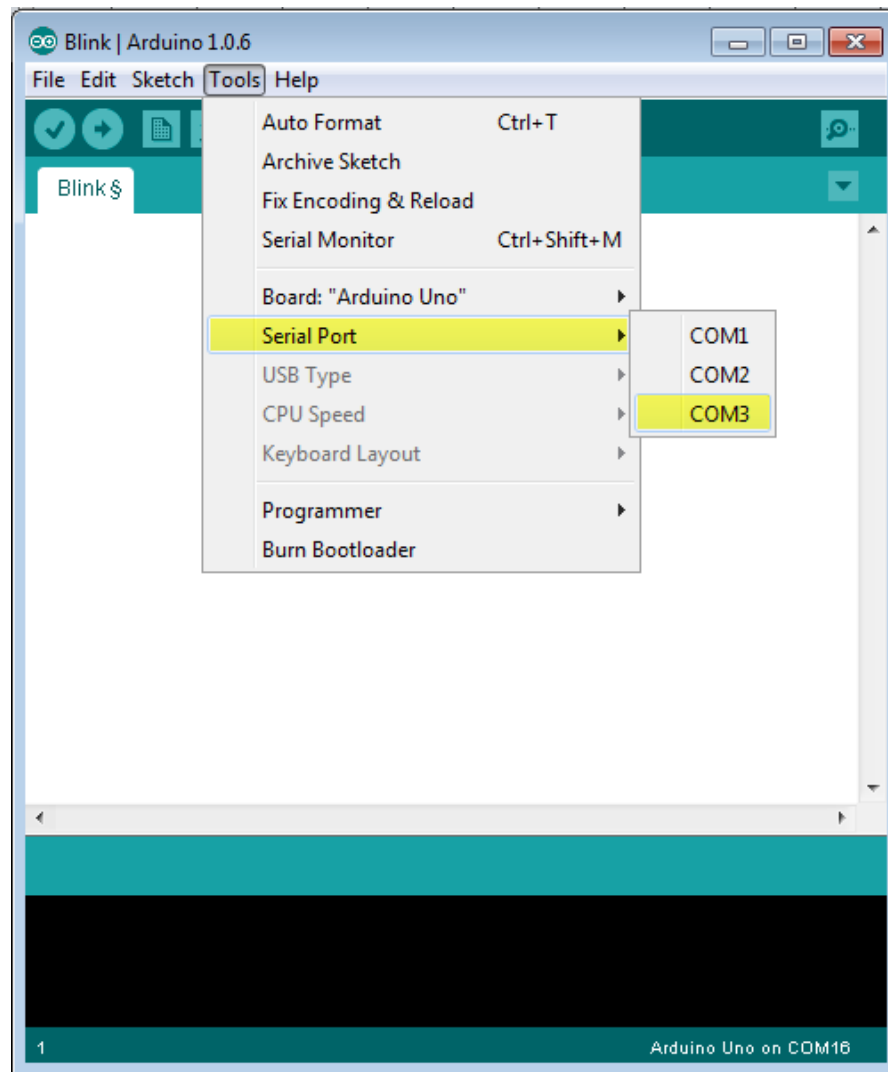
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

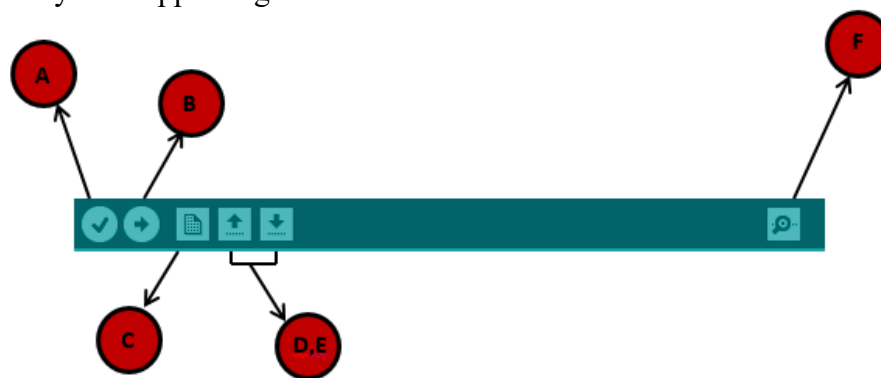
- Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**A** – Used to check if there is any compilation error.

**B** – Used to upload a program to the Arduino board.

**C** – Shortcut used to create a new sketch.



**D** – Used to directly open one of the example sketch.

**E** – Used to save your sketch.

**F** – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

**Note** – If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

## Experiment no: 03

**Experiment Title:** Interfacing basic LED blink and basic 4x4 keypad with Arduino.

### Objective:

- Interfacing Arduino analog and digital ports.
- Interfacing Arduino digital ports with 4x4 keypad and use of serial monitor for value cross checking.

### Required Apparatus:

1. Arduino Uno board
2. USB cable (for power and programming)
3. LED (optional, as pin 13 has a built-in LED)
4. 220Ω resistor (if using an external LED)
5. Breadboard and jumper wires (if using an external LED)

### Description:

The pins on the Arduino can be configured as either inputs or outputs. Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 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), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors. Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller, but the remaining chip will still function adequately. For this reason, it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular application.

### Procedure:

- If using the **built-in LED** (on pin 13), no external connections are needed.
- If using an **external LED** Connect the **anode (long leg)** of the LED to **pin 13** via a **220Ω resistor**.
- Connect the **cathode (short leg)** to the **GND (ground)** pin on the Arduino.
- Open the **Arduino IDE** on your computer.

- Ensure the correct board is selected:  
Tools > Board > Arduino Uno.
- Select the correct port:  
Tools > Port > COMx (where x matches your Arduino's port).
- **Upload** the Code to Arduino Uno.

### Circuit Connection:

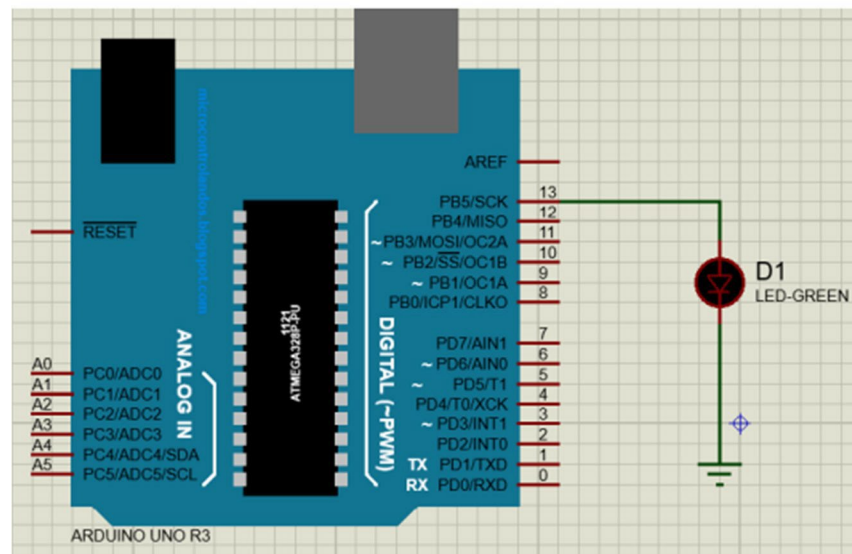


Figure 3.1: Implementation of simple Arduino based circuits with LED.

### Code:

```
// Define pin number
const int ledPin = 13;

void setup() {
  // Set pin 13 as output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Turn LED on
  digitalWrite(ledPin, HIGH);
  delay(1000); // Wait 1 second

  // Turn LED off
  digitalWrite(ledPin, LOW);
}
```

```

    delay(1000); // Wait 1 second
}

```

## Exercises:

- Design Arduino based switch-controlled LED system.

## Procedure:

1. Wire the 4×4 keypad:
  - Connect rows (R1–R4) → Arduino pins 2–5
  - Connect columns (C1–C4) → Arduino pins 6–9
2. Upload the code to Arduino Uno.
3. Open Serial Monitor (Ctrl+Shift+M or Tools > Serial Monitor).
  - Set baud rate to 9600
4. Press any key:
  - Verify each button shows correct character in Serial Monitor
  - Example output: Key Pressed: 5

## Circuit Connection:

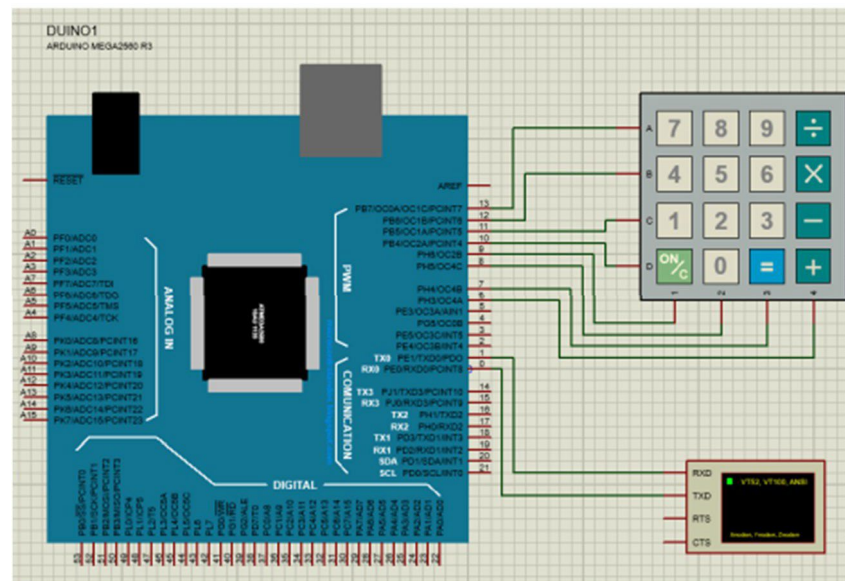


Figure 3.2: Interfacing basic 4x4 keypad with Arduino.

**Code:**

```
#include <Keypad.h>

// Define keypad size
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns

// Define keypad keymap (matches image layout)
char keys[ROWS][COLS] = {
  {'7', '8', '9', '+'},
  {'4', '5', '6', '-'},
  {'1', '2', '3', 'x'},
  {'0', '0', '=', '/'}
};

// Connect keypad ROW0-ROW3 to Arduino digital pins 2-5
byte rowPins[ROWS] = {2, 3, 4, 5};

// Connect keypad COL0-COL3 to Arduino digital pins 6-9
byte colPins[COLS] = {6, 7, 8, 9};

// Create the Keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  // Initialize serial communication for output
  Serial.begin(9600);
  Serial.println("Keypad Test Started...");
}

void loop() {
  char key = keypad.getKey(); // Read key

  if (key) {
    Serial.print("Key Pressed: ");
    Serial.println(key);
  }
}
```

**Exercises:**

- Design a simple calculator using Arduino and available keypad.

## Experiment no: 04

**Experiment Title:** Printing Potentiometer value to Serial Monitor and Interfacing Piezo Buzzer with Arduino.

### Objective:

- Interfacing Arduino analog port with potentiometer and use of serial monitor for observation of value for cross checking.
- Interfacing Arduino digital port with a simple buzzer.

### Description:

The analog pins on the Arduino can be configured as either inputs or outputs. Pins configured as INPUT with pinMode() can be used to take input from devices that generate an analog value. Potentiometers are devices that can produce variable resistance. In this experiment we shall connect the middle wire (wiper) of the potentiometer to an analog port of an Arduino, the other two terminals are connected to power and ground ports of Arduino respectively. Thus when the wiper is rotated, variable resistance is provided as input to the analog pin of Arduino, which is read and output is shown correspondingly to serial monitor.

### Piezo buzzer

Piezo buzzer is an electronic device commonly used to produce sound. Piezo buzzer is based on the inverse principle of piezo electricity discovered in 1880 by Jacques and Pierre Curie. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezo electric materials. Piezoceramic is class of manmade piezoelectric material, which poses piezo electric effect and is widely used to make “disc”, the heart of piezo buzzer. When subjected to an alternating electric field they stretch or compress, in accordance with the frequency of the signal thereby producing sound. Built-in functions in the Arduino library delay(), tone() and noTone() may be used.

### Procedure:

- Upload the code to Arduino Uno.
- Open **Serial Monitor** (Ctrl+Shift+M or Tools > Serial Monitor).
- Set baud rate to **9600**.
- Rotate the potentiometer and observe value range from **0 to 1023**.

### Circuit Connection:

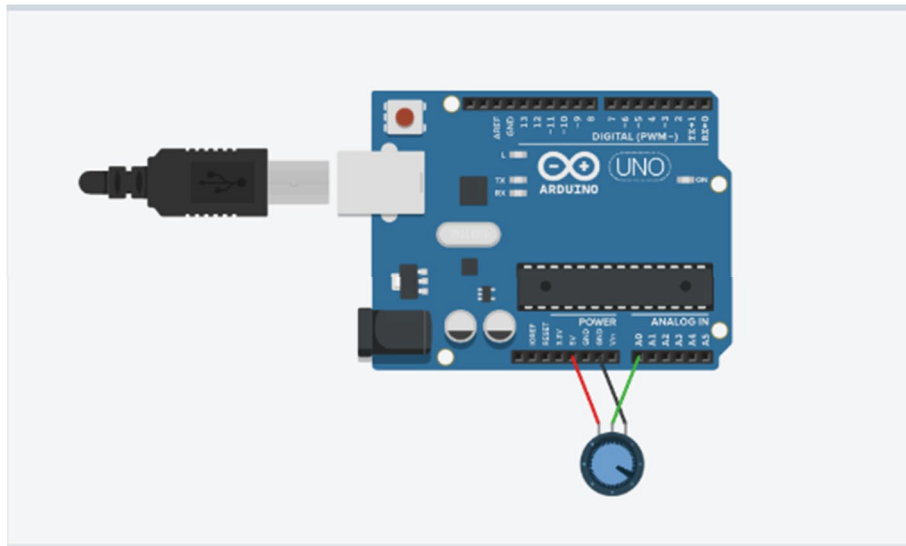


Figure 3.3: Interfacing Arduino analog port with potentiometer

Code:

```
// Define analog pin
const int potPin = A0;

void setup() {
  // Start serial communication
  Serial.begin(9600);
}

void loop() {
  // Read analog value (0 to 1023)
  int potValue = analogRead(potPin);

  // Print the value to Serial Monitor
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);

  // Small delay to stabilize output
  delay(200);
}
```

### Exercises:

- Design a light with dimming/brightening effect using Potentiometer and LED.

### Procedure:

1. Wire the buzzer:
  - Connect buzzer (+) pin → Arduino D8 through 100Ω resistor
  - Connect buzzer (-) pin → Arduino GND
2. Upload the code to Arduino Uno
3. Change 1000 in tone(8, 1000) to adjust frequency (Hz)

### Circuit Connection:

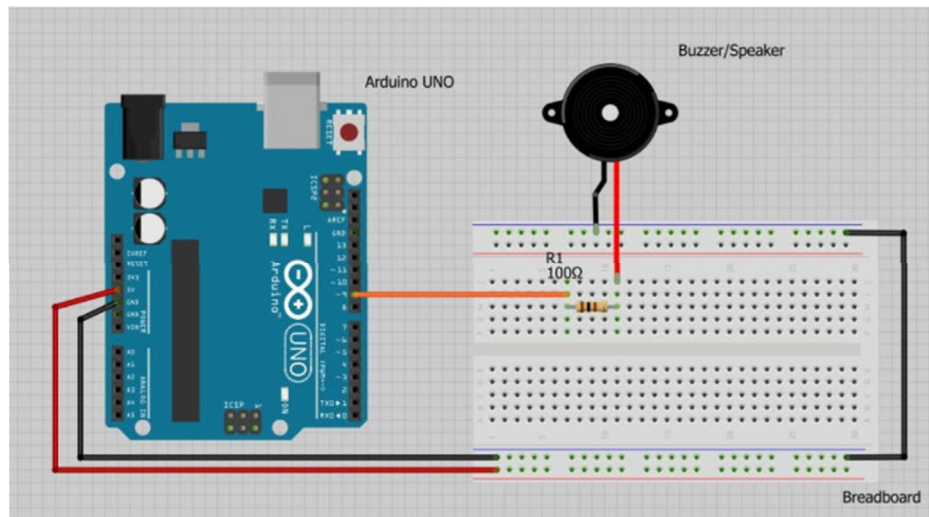


Figure 3.4: Interfacing Piezo Buzzer with Arduino.

### Code:

```
void setup() {  
  pinMode(8, OUTPUT); // Set pin 8 as output  
}  
  
void loop() {  
  tone(8, 1000); // Play 1kHz tone  
  delay(1000); // Play for 1 second  
  noTone(8); // Stop tone  
  delay(1000); // Wait 1 second  
}
```

### Exercises:

- Students can test out how various frequency changes in the tone() function changes the output of the buzzer.



## **Experiment no: 05**

**Experiment Title:** Interfacing Servo motor and DC motors with motor driver with Arduino.

### **Objective:**

- Interfacing Arduino digital port with a servo motor.
- Interfacing simple DC motors with motor driver L293D and Arduino
- Creating the working procedures of a two-wheeler car.

### **Required Apparatus**

- Arduino UNO
- 1 × L298N Motor Driver Module
- 2 × DC Motors (typically 6V–12V)
- 1 × Servo Motor (e.g., SG90 or MG996R)
- 1 × External Power Supply (e.g., 9V battery or 12V adapter)
- Jumper Wires
- Breadboard (optional, for prototyping)
- Screwdriver (for terminal connections)

### **Description:**

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration in a mechanical system. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servo motors. The motors used in the lab have a rotational range of 180°. It consists of a DC motor, gear system, Position sensor and Control circuit. Degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo library and its associated functions such as attach(), write() may be used.

- **DC Motor:**

A direct current, or DC motor is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction. The L293D has two +V pins (8 and 16).

The pin '+Vmotor (8) provides the power for the motors, and +V

(16) for the chip's logic. We have connected both to the Arduino 5V pin. However, if you were using a more powerful motor, or a higher voltage motor, you would provide the motor with a separate power supply using pin 8 connected to the positive power supply and the ground of the second power supply is connected to the ground of the Arduino. Motor drivers acts as an

interface between the motors and the control circuits. Motors require high amount of current whereas the controller circuit works on low current signals. So the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor. We should not drive the motor directly from Arduino board pins. This may damage the board. Built-in functions in the Arduino library `digitalWrite()` and `delay()` may be used.

#### Procedure:

1. Connect the Servo Motor to Arduino:
  - Yellow (Signal) → Arduino Pin 9
  - Red (VCC) → Arduino 5V
  - Black or Brown (GND) → Arduino GND
2. Upload Code via Arduino IDE:
  - Connect the Arduino to your computer using the USB cable.
  - Open the Arduino IDE and paste the code below.
  - Select the correct board and port, then upload.
3. Observe the Servo Movement:
  - The servo will sweep back and forth between  $0^\circ$  and  $180^\circ$ .

#### Circuit Connection:

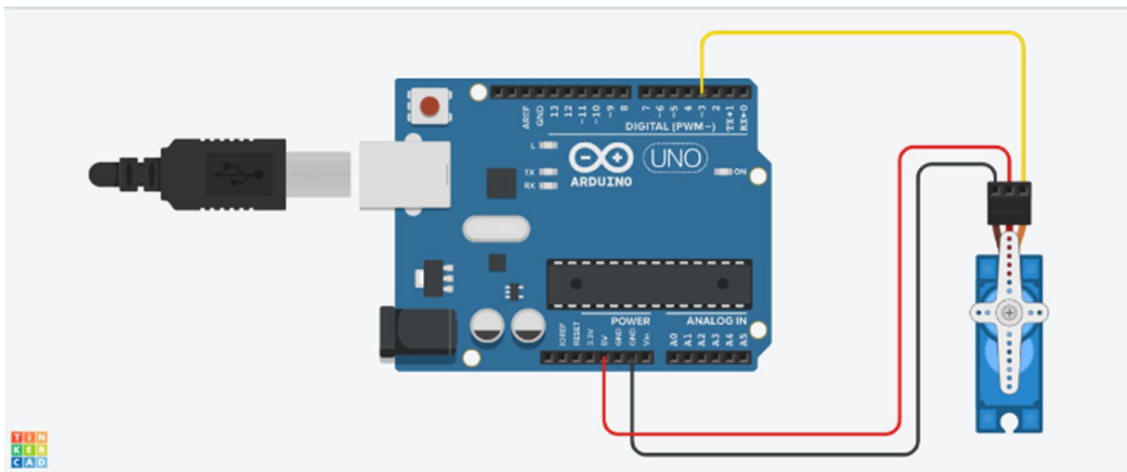


Figure 3.5: Interfacing Servo motor with Arduino.

#### Code:

```
#include <Servo.h>

Servo myServo; // Create a servo object
```

```

void setup() {
  myServo.attach(9); // Attach servo to pin 9
}

void loop() {
  // Sweep from 0 to 180 degrees
  for (int pos = 0; pos <= 180; pos++) {
    myServo.write(pos);
    delay(15); // Wait for the servo to reach the position
  }

  // Sweep back from 180 to 0 degrees
  for (int pos = 180; pos >= 0; pos--) {
    myServo.write(pos);
    delay(15);
  }
}

```

### Exercises:

- Students can test out how various delay changes in code change the output of the motor.
- Students may experiment how to connect both Servo motor and Buzzer to the same Arduino and combine the separate coding functions to make a cohesive program.

### Procedure

- 1. Connect the Motors to L298N:**
  - Motor A to OUT1 and OUT2
  - Motor B to OUT3 and OUT4
- 2. Connect L298N to Arduino:**
  - IN1 → Arduino pin 2
  - IN2 → Arduino pin 3
  - IN3 → Arduino pin 4
  - IN4 → Arduino pin 5
  - ENA → Arduino pin 9 (PWM)
  - ENB → Arduino pin 10 (PWM)
- 3. Power Connections:**
  - L298N +12V → External power supply
  - L298N GND → Arduino GND and power supply GND
  - L298N 5V → Arduino 5V (if jumper is set)
- 4. Upload the Code** (see below)
- 5. Power the System** and observe motor behavior.

### Design:

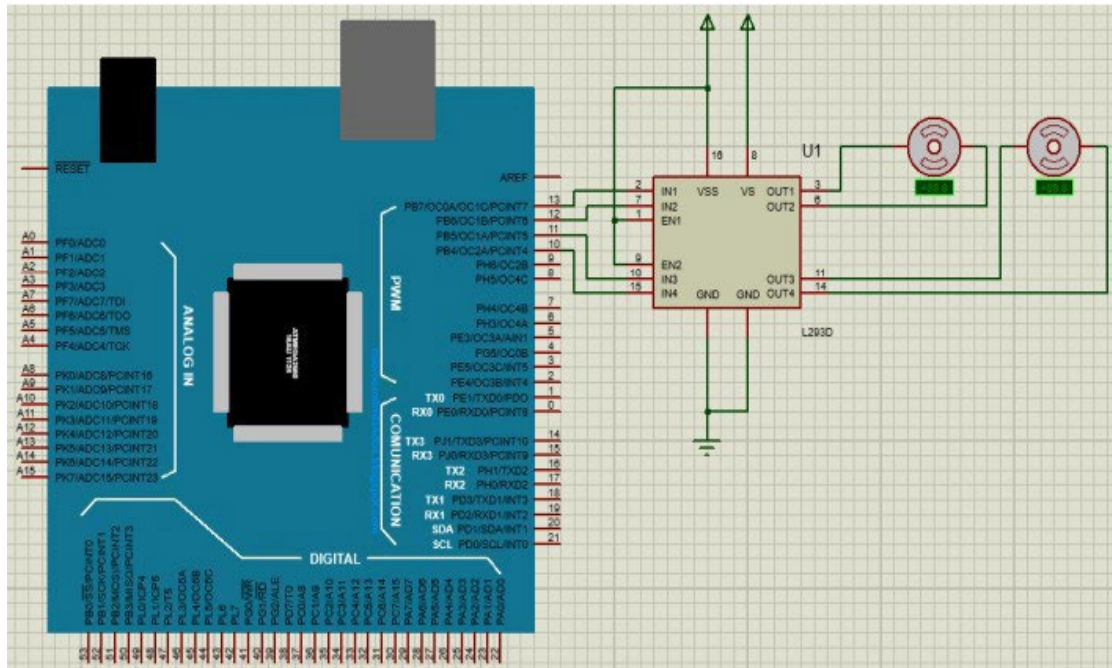


Figure 3.9: Interfacing DC motors with motor driver and Arduino.

**Code:**

```
// Motor A pins
int motor1pin1 = 2;
int motor1pin2 = 3;
int enableA = 9;

// Motor B pins
int motor2pin1 = 4;
int motor2pin2 = 5;
int enableB = 10;

void setup() {
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(enableA, OUTPUT);

  pinMode(motor2pin1, OUTPUT);
  pinMode(motor2pin2, OUTPUT);
  pinMode(enableB, OUTPUT);
}

void loop() {
```

```
// Set speed (0-255)
analogWrite(enableA, 150);
analogWrite(enableB, 150);

// Forward
digitalWrite(motor1pin1, HIGH);
digitalWrite(motor1pin2, LOW);
digitalWrite(motor2pin1, HIGH);
digitalWrite(motor2pin2, LOW);
delay(3000);

// Reverse
digitalWrite(motor1pin1, LOW);
digitalWrite(motor1pin2, HIGH);
digitalWrite(motor2pin1, LOW);
digitalWrite(motor2pin2, HIGH);
delay(3000);
}
```

**Exercises:**

- Students can test out how to design a working scenario of four-wheeler car using Arduino.

## **Experiment no: 06**

**Experiment Title:** Interfacing LDR, Ultrasonic & Temperature Sensor with Arduino.

### **Objective:**

- Interfacing Arduino digital port with an Ultrasonic Sensor.
- Interfacing Arduino digital port with an LM-35 temperature sensor.
- Interfacing Arduino digital port Light Dependent Resistor with Arduino.

### **Required Apparatus**

- **1 × Arduino Uno**
- **1 × HC-SR04 Ultrasonic Sensor**
- **Jumper Wires**
- **USB Cable** (for programming and power)
- **Arduino IDE**

### **Description:**

- **Ultrasonic Sensor**

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect from boundaries to produce distinct echo patterns. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. The ultrasonic sensors used in the lab, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. The working principle is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated. Built-in functions in the Arduino library `digitalWrite()` and `delay()` may be used.

- **Temperature Sensor**

LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of LM35 over thermistor is it does not require any external calibration. The coating also protects it from self-heating. LM35 can measure from -55 degrees centigrade to 150-degree centigrade. The accuracy level is very high if operated at optimal temperature and humidity levels. The conversion of the output voltage to centigrade is also easy and straight forward. The input voltage to LM35 can be from +4 volts to 30 volts. It consumes about 60 microamperes of current. In order to understand the working principle of LM35 temperature sensor we have to understand the linear scale factor. In the features of LM35 it is given to be +10 mills volt per degree centigrade. It means that with increase in output of 10 mills volt by the sensor `vout` pin the temperature value increases by one. For example, if the sensor is

outputting 100 mills volt at vout pin the temperature in centigrade will be 10-degree centigrade. The same goes for the negative temperature reading. If the sensor is outputting -100 mills volt the temperature will be -10 degrees Celsius. Built-in functions in the Arduino library `analogRead()` and `delay()` may be used.

- **LDR**

An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. Light Dependent Resistors (LDR) are also called photoresistors. They are made of high resistance semiconductor material. The resistance values of LDR in darkness are several megaohms whereas in bright light it will be dropped to hundred ohms. Built-in functions in the Arduino library `analogRead()` and `delay()` may be used.

### Procedure:

1. Connect the HC-SR04 to Arduino:
  - VCC → Arduino 5V
  - Trig → Arduino Pin 7
  - Echo → Arduino Pin 6
  - GND → Arduino GND
2. Upload the Code:
  - Connect the Arduino to your computer via USB.
  - Open the Arduino IDE, paste the code below, and upload it.
3. Open Serial Monitor:
  - Set the baud rate to 9600 to view distance readings in real time.

### Circuit Connection:

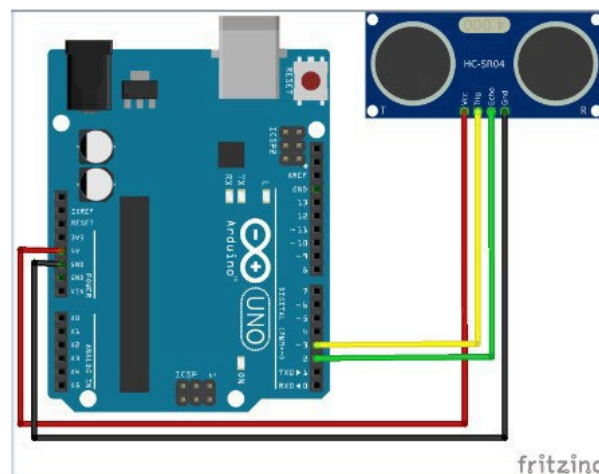


Figure 3.6: Interfacing Ultrasonic Sensor with Arduino.

## Code:

```
const int trigPin = 7;
const int echoPin = 6;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Send a 10µs pulse to trigger the sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo time
  long duration = pulseIn(echoPin, HIGH);

  // Calculate distance in centimeters
  float distance = duration * 0.034 / 2;

  // Print the distance
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(500);
}
```

## Exercises:

- Students can test out how obstacles change the output of the sensor.
- Students may experiment how to connect Servo motor, buzzer and the ultrasonic sensor to the same Arduino and combine the separate coding functions to make a cohesive program.

## Procedure:

### 1. Circuit Assembly

- Connect the **VCC pin** of the LM35 to **5V** on the Arduino.
- Connect the **GND pin** of the LM35 to **GND** on the Arduino.
- Connect the **VOUT pin** of the LM35 to **A0** (Analog 0) on the Arduino.

### 2. Software Setup





```

Serial.print(temperature);
Serial.println(" °C");

delay(1000); // Wait 1 second between readings
}

```

### Exercises:

- Students can test out how to show temperature readings on LCD display.

### Procedure:

- Assemble the components on the breadboard as per the circuit.
- Connect LDR and resistor to form a voltage divider circuit.
- Interface LCD display to Arduino using the specified pins.
- Open **Arduino IDE**, connect the board, and upload the code below.
- Observe light intensity on the **LCD display** as ambient light changes.

### Circuit Connection:

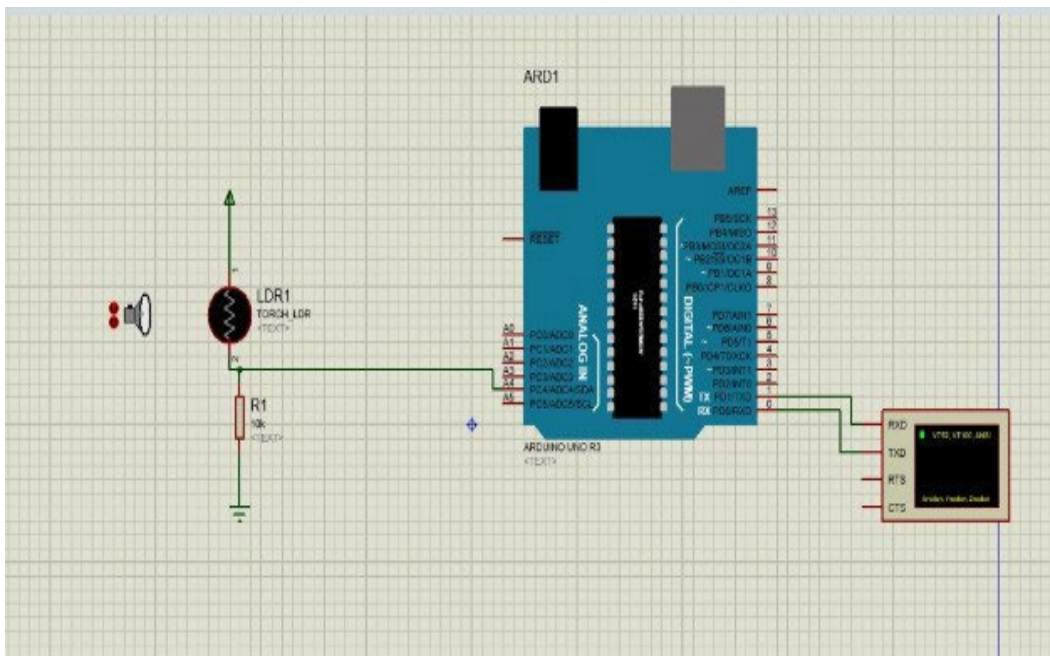


Figure 3.10: Interfacing Light Dependent Resistor with Arduino.

### Code:

```

#include <LiquidCrystal.h>

```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
const int ldrPin = A0;  
  
void setup() {  
  lcd.begin(16, 2);  
  lcd.print("Light Intensity:");  
}  
  
void loop() {  
  int ldrValue = analogRead(ldrPin);  
  
  lcd.setCursor(0, 1);  
  lcd.print("Value: ");  
  lcd.print(ldrValue);  
  delay(1000);  
}
```

**Exercises:**

- Students can test out how to combine LDR with other components to create a cohesive system.

## **Experiment no: 07**

**Experiment Title:** Interfacing basic LCD display with Arduino.

### **Objective:**

- Interfacing 2x20 alphanumeric LCD with Arduino digital ports.

### **Required Apparatus:**

- Arduino Uno R3 — 1
- 16x2 LCD Display (LM032L) — 1
- Potentiometer (10k $\Omega$ ) — 1
- Breadboard — 1
- Jumper wires — several
- USB Cable — 1

### **Description:**

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface. The example sketch provided prints "Hello World!" to the LCD and shows the time in seconds since the Arduino was reset.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and Bklt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions. The Hitachi- compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4- bit mode.

## Procedure:

1. Assemble the LCD and potentiometer on the breadboard and wire according to the table above.
2. Connect the Arduino Uno to your computer using the USB cable.
3. Open **Arduino IDE**, select "**Arduino Uno**" from Tools > Board, and choose the correct **COM port**.
4. Use the built-in LiquidCrystal library to write and upload the sketch.

## Circuit Connection:

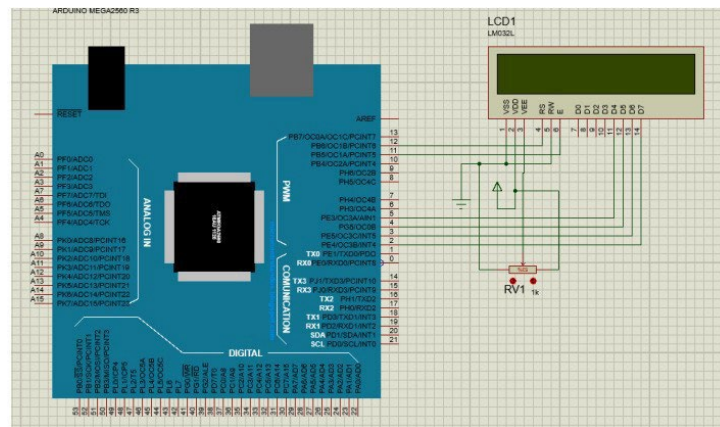


Figure 3.7: Interfacing basic LCD display with Arduino.

## Code:

```
#include <LiquidCrystal.h>

// RS, E, D4, D5, D6, D7 connected to 12, 11, 5, 4, 3, 2
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);          // Initialize 16x2 LCD
  lcd.print("Hello, Noor!");
}

void loop() {
  lcd.setCursor(0, 1);      // Move to second line
  lcd.print("Using Arduino Uno");
  delay(1000);
}
```

## Exercises:

- Students can test out how to design a simple calculator using Arduino, available keypad and 2x20 LCD display.
- Students can test out how to design a system to show values in seven segment display

## Experiment no: 08

**Experiment Title:** Interfacing Infrared Sensor (Active and Passive) with Arduino.

### Objective:

- Interfacing Arduino digital port Infrared Sensor (Active) with Arduino.
- Interfacing Arduino digital port Infrared Sensor (Passive) with Arduino.

### Required Apparatus

- Arduino Uno R3 — 1
- IR Obstacle Sensor Module — 1
- LED (Yellow) — 1
- 220Ω Resistor — 1 (for LED)
- Breadboard — 1
- Jumper wires — several
- USB Cable — 1

### Description:

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, which can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received. This active infrared sensor includes both the transmitter as well as the receiver. In most of the applications, the light-emitting diode is used as a source. LED is used as a non-imaging infrared sensor whereas the laser diode is used as an imaging infrared sensor. Built-in functions in the Arduino library `digitalWrite()` and `delay()` may be used.

### Procedure

1. Assemble the IR sensor and LED on the breadboard.
2. Wire the IR sensor to 5V, GND, and pin 7 (signal).
3. Connect the LED to pin 13 with a resistor in series.
4. Open Arduino IDE → Select **Arduino Uno** → Choose correct **COM port**.
5. Paste and upload the code below.
6. Bring an object near the IR sensor to observe LED feedback and serial messages.

## Circuit Connection:

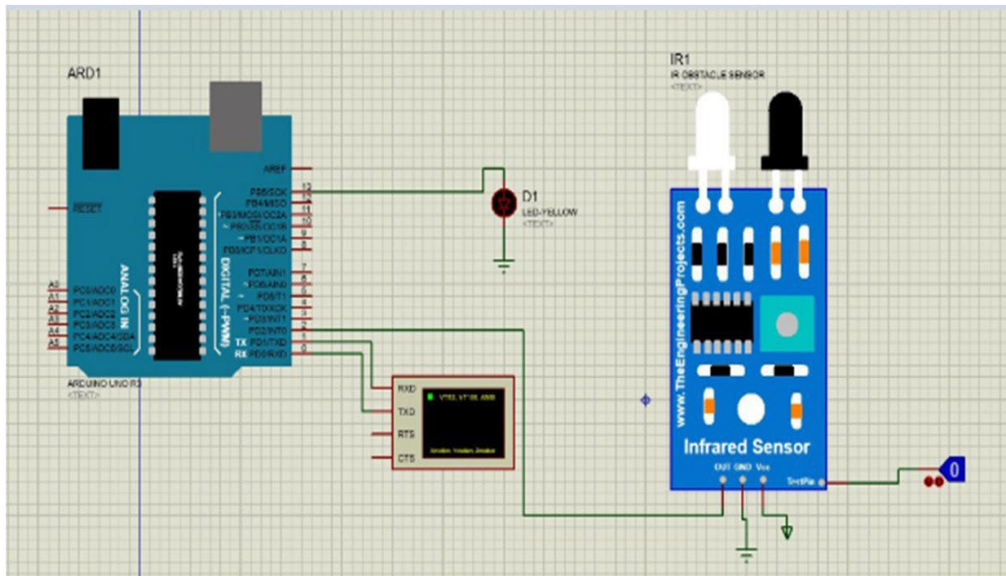


Figure 3.11: Interfacing Infrared Sensor (Active) with Arduino.

## Code:

```
const int irPin = 7;    // IR sensor output pin
const int ledPin = 13;  // LED indicator pin

void setup() {
  pinMode(irPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int obstacle = digitalRead(irPin);

  if (obstacle == LOW) { // LOW = Obstacle detected
    digitalWrite(ledPin, HIGH);
    Serial.println("Obstacle detected!");
  } else {
    digitalWrite(ledPin, LOW);
    Serial.println("Path clear");
  }

  delay(500);
}
```

## Exercises:

- Students can test out how to combine IR Sensor with other components to create a cohesive

system.

### Procedure:

1. Assemble the RFID module, PIR sensor, and LED on a breadboard and wire them to the Arduino as listed above.
2. Install the required **MFRC522 library** in Arduino IDE:
  - Go to **Sketch > Include Library > Manage Libraries**
  - Search for **MFRC522** and install by GithubCommunity
3. Connect your Arduino Uno and upload the provided code.
4. Open **Serial Monitor** and scan your RFID card or tag.
5. When valid ID is scanned and motion is detected, the LED turns ON indicating access granted.

### Circuit Connection:

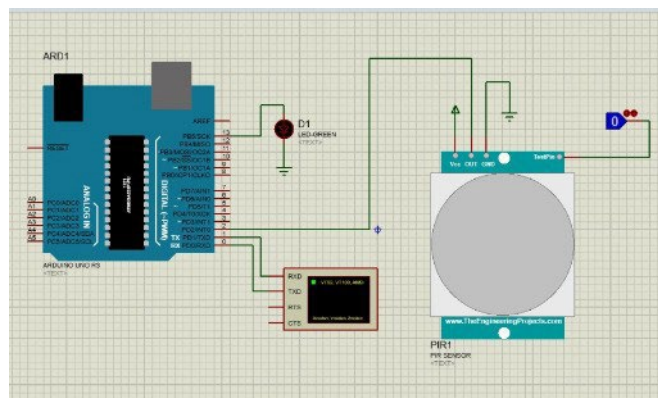


Figure 3.12: Interfacing Infrared Sensor (Passive) with Arduino.

### Code:

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
#define LED_PIN 7
#define PIR_PIN 8

MFRC522 rfid(SS_PIN, RST_PIN);
byte validUID[4] = {0xDE, 0xAD, 0xBE, 0xEF}; // Replace with your RFID tag's UID
```



```

void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  pinMode(LED_PIN, OUTPUT);
  pinMode(PIR_PIN, INPUT);
  Serial.println("Scan RFID and detect motion...");
}

void loop() {
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) return;

  bool isAuthorized = true;
  for (byte i = 0; i < 4; i++) {
    if (rfid.uid.uidByte[i] != validUID[i]) {
      isAuthorized = false;
      break;
    }
  }

  if (isAuthorized && digitalRead(PIR_PIN) == HIGH) {
    Serial.println("Access Granted!");
    digitalWrite(LED_PIN, HIGH);
    delay(2000);
    digitalWrite(LED_PIN, LOW);
  } else {
    Serial.println("Access Denied or No Motion");
  }

  rfid.PICC_HaltA();
}

```

#### Exercises:

- Students can test out how to combine PIR Sensor with other components to create a cohesive system.

## **Experiment no: 09**

### **Experiment Title:** Interfacing with Stepper Motor

#### **Objective:**

- To understand basic theory of Stepper Motor.
- To understand the operation theory and characteristics of Stepper Motor.
- To understand 1-phase, 2-phase and 1-2 phase excitations.

#### **Required Apparatus:**

- Arduino Uno R3 — 1
- 28BYJ-48 Stepper Motor — 1
- ULN2003 Driver Module — 1
- Breadboard — 1
- Jumper wires — several
- External 5V DC Power Supply (for motor) — 1
- USB Cable — 1
- Computer with Arduino IDE — 1

#### **Theory:**

In general, a motor is a device that transforms electrical energy to mechanical energy. It is a synchronous electric motor capable of dividing a complete rotation into many steps. Stepper motors typically consist of a permanent magnet shaft (rotor) encircled by a stator. As long as the motor is not large, the angular position of the motor can be precisely regulated without the use of a feedback device. As a result, it operates in a simple precise open-loop system in which the output is directly proportional to the input.

Permanent Magnet (PM) Stepper Motors consist of permanent magnet rotors with no teeth, which are magnetized perpendicular to the axis of rotation. By energizing the four phases (in sequence), the rotor rotates due to the attraction of magnetic poles. The stepper motor shown in the diagram below will take 90-degree steps as the windings are energized by passing electricity between the coils of the stator in clockwise sequence: X, X', Y, Y' to complete a revolution. Energized a particular phase by DC current will create N pole of the stator (the source part of electricity) and S pole to the other wise, which interact the N pole of the rotor. 45-degree steps are created by energizing consecutive two phases. As example, SA and SB creates 45-degree steps rotate will point to the middle of A and B phases. Anti-clockwise rotation will be created by energized the stators from opposite direction.

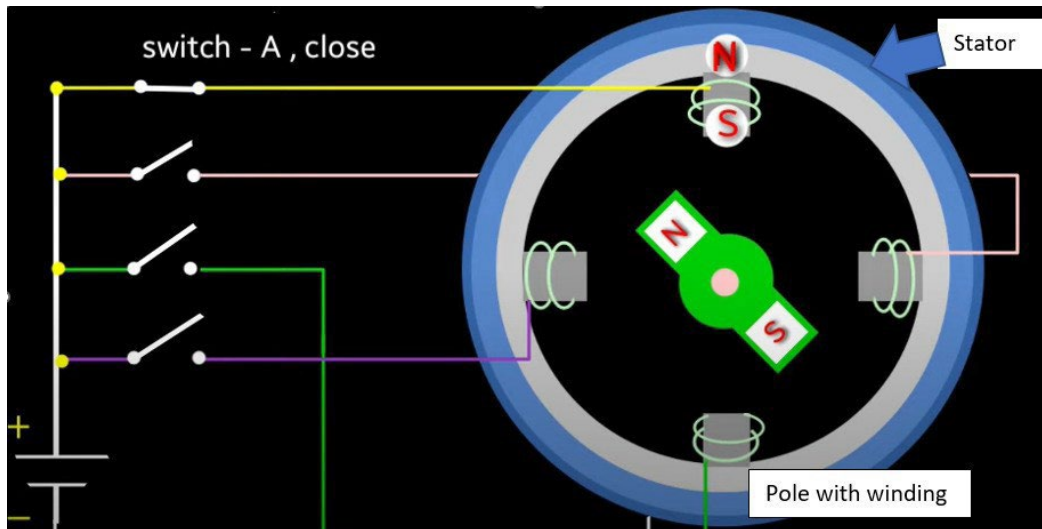
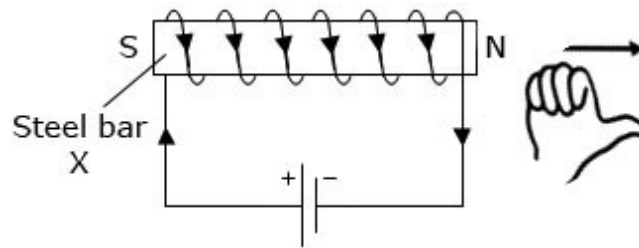


Figure 6.5: Stepper Motor and its Working Strategy



Since stepping motor makes step- by-step movement and each step is equidistant, the rotor and stator magnetic field must be synchronous. During start-up and stopping, the two fields may not be synchronous, so it is suggested to slowly accelerate and decelerate the stepping motor during the start-up or stopping period.

Figure 6.6 is used to explain the operation of simplified stepping motor ( $90^\circ/\text{step}$ ). Here the A coil and B coil are perpendicular to each other. If either A or B coil is excited (a condition which is known as single-phase excitation), the rotor can be moved to  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ ,  $270^\circ$ -degree position depending on the current's ON/OFF conditions in the coils, see Figure 6.6(a). If both coils have current flowing at the same time, then the rotor positions can be  $45^\circ$ ,  $135^\circ$ ,  $225^\circ$ ,  $315^\circ$  degrees as shown in Figure 6.6(b). This is known as two-phase excitation. In Figure 6.6(c), the excitation alternates between 1-phase and 2-phase, then the motor will rotate according to  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ ,  $135^\circ$ ,  $180^\circ$ ,  $225^\circ$ ,  $270^\circ$ ,  $315^\circ$  sequence. This is 1-2 phase excitation; each step distance is only half of step movement of either 1-phase or 2-phase excitation. Stepping motor can rotate in clockwise or counter-clockwise direction depending on the current pulse sequence applied to the excitation coils of the motor. Referring to the truth tables in Figure 6.6(a), (b), (c). If signals are applied to coil A and B according to Step 1,2,3,4,5,6,7,8, then counter-clockwise movement is achieved. And vice-versa is true. If signals are applied according to step 8,7,6,5,4,3,2,1, then clockwise movement is achieved.

### Procedure

1. Assemble the circuit on a breadboard as per the wiring guide above.
2. Connect Arduino Uno to your computer with the USB cable.

3. Install the **Stepper** library (built-in with Arduino IDE).
4. Upload the provided code below to control the motor steps.
5. Observe the motor rotating based on the programmed logic.

### Circuit Diagram:

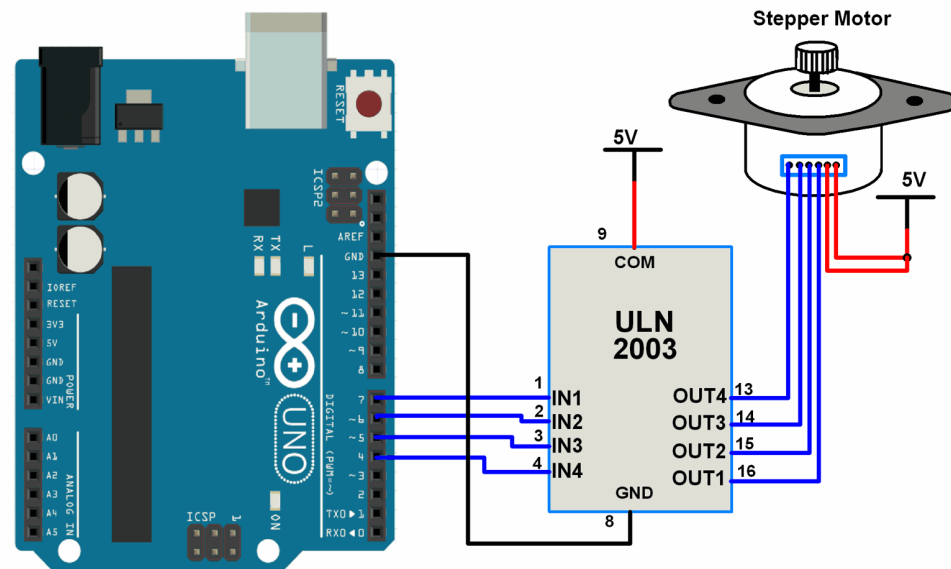


Figure: Interfacing with Stepper Motor

### Code:

```
#include <Stepper.h>

const int stepsPerRevolution = 2048; // For 28BYJ-48 motor
Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);

void setup() {
  myStepper.setSpeed(10); // 10 RPM
  Serial.begin(9600);
}

void loop() {
  Serial.println("Rotating clockwise...");
  myStepper.step(stepsPerRevolution); // Full rotation
  delay(1000);

  Serial.println("Rotating counterclockwise...");
  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```