IOT KIT MANUAL

Cognifront

Using IOT kit:

1. In IOT kit, all the components are already mounted in the boxes. So student do not need to handle them, and risk of damaging the component is very less.
2. Each pin of the microcontroller board is given on the board using colored sockets and having proper numbering. So no risk of connecting wrong pin number.
3. Also, for each component its power supply and data pins are given on the board using colored sockets.
4. The patch cords are given in the kit for the connecting the devices with the microcontroller.
5. These patch cords are very sturdy, and very easy to connect.
6. Due to the sockets of the microcontroller, devices and patch cords, it is very easy to build the circuit.
7. Troubleshooting of the circuit is also very easy.

Interfacing and Programming Arduino UNO board with PC

Aim/Objectives:

- To study the layout of the Arduino UNO Board.
- To Identify the microcontroller IC (ATMEGA328) on the Arduino UNO board
- To study the installation and interface of the Arduino IDE.
- To learn basic commands to write the program in the Arduino IDE.
- To write a simple program in the Arduino IDE to interface the Arduino board with PC.

Software:

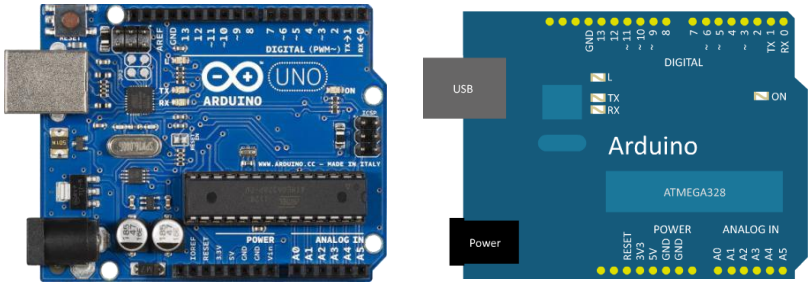
- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino UNO Board
- PC / Laptop

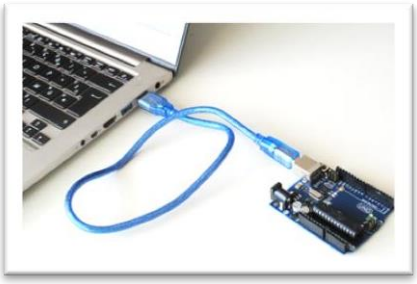
Theory:

Introduction to Arduino Board



1. The Arduino Board is an Open Source hardware.
2. The brain of the Arduino Board is the **microcontroller IC ATMEGA328** (3rd version) present at the bottom right corner of the board.
3. There are total 20 GPIO pins mounted on the board.
4. From these, on the top side of the board, 14 are digital input and output pins named as 0, 1….13. These pins can be used for digital input or digital output (digital read or write).
5. The remaining 6 pins, at the bottom right side of the board are the Analog Input pins, named as A0, A1….A5. These pins are used to read only the analog input (analog read).
6. From the 14 digital pins, 6 pins are denoted by '~' sign, called as PWM pin. These pins are used to write the analog signal (analog write). That means these pins are used to send the analog output.
7. Arduino board can be connected to the PC using USB cable. The cable has two different connectors at its two ends. One of type-A(PC side)) and the other is type-B(Arduino side).

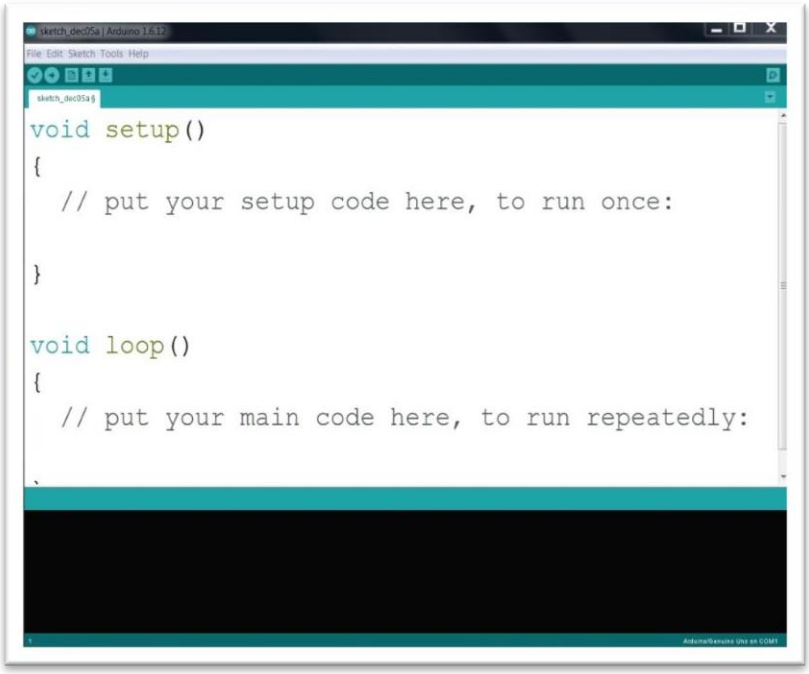
8. Power is given to Arduino Board by two ways,
    a. Through the USB cable
    or
    b. By external adapter or battery using the barrel power connector given on the Arduino board (Here no need of PC/Laptop).
9. A debugging led is given on the Arduino Board for debugging the board and the program. This led is connected internally to the digital pin number 13. It is also called as inbuilt led.



Introduction to Arduino IDE (Integrated Development Environment)

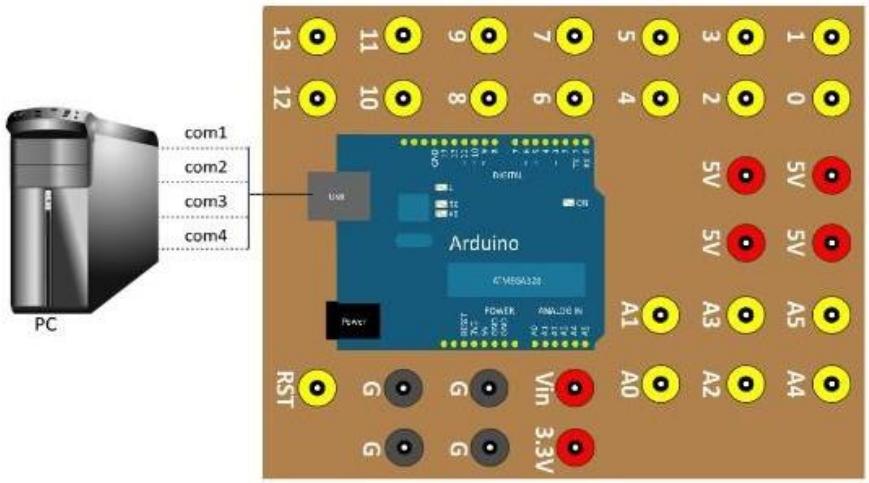
1. Download the latest version of the Arduino IDE from the website Arduino.cc. The download is free.
2. Arduino IDE is available for windows, Linux, and mac-OS. Choose the proper IDE as per the operating system installed in your PC or laptop.
3. It is an exe file.
4. After downloading, run it to install on the PC.

Using Arduino IDE:



1. Start IDE
2. In order to exchange data and information between PC and Arduino board, the type of the board and the port number is to be selected. (This selection is available under 'Tool' menu)
3. Type the 'c' program in Arduino IDE
4. Save and compile the program (tick mark)
5. If the program compiles properly, it can be transferred (uploaded) to the microcontroller IC in the Arduino board. For this use the symbol 'right arrow'.
6. When the program is being transferred to the Arduino board, the LEDs on the Arduino board which indicates serial communication starts blinking.
7. After this the microcontroller should start working on its own.
8. In real life applications Arduino is removed from the PC by disconnecting the USB cable and separate power supply is given by the adaptor or the power bank.

Interface diagram:



Procedure:

Write the program as per the algorithm given below.
Save and compile the program.
Connect the Arduino board to PC/Laptop using USB cable.
Upload the compiled program and check the output.

Algorithm:

Start IDE
Configure the pin number '13' as 'OUTPUT' pin
Make the 'OUTPUT' as 'HIGH'
Give delay of one second
Make the 'OUTPUT' as 'LOW'
Give delay of one second

Observation:

Observe the LED connected to the pin number ‘13’, it starts blinking as soon as the program is uploaded.

Interfacing LED bar with the Arduino UNO board

Aim/Objectives:

- To study the working of LED.
- To study the configurations used to create a LED bar.
- To study the pin configuration of LED and to interface it with the Arduino board using resistor.
- To interface the LED bar with the Arduino board using IOT kit
- To interface the LED bar with the Arduino board using breadboard
- To write a program to generate different patterns (Festival lighting) on LED bar.

Software:

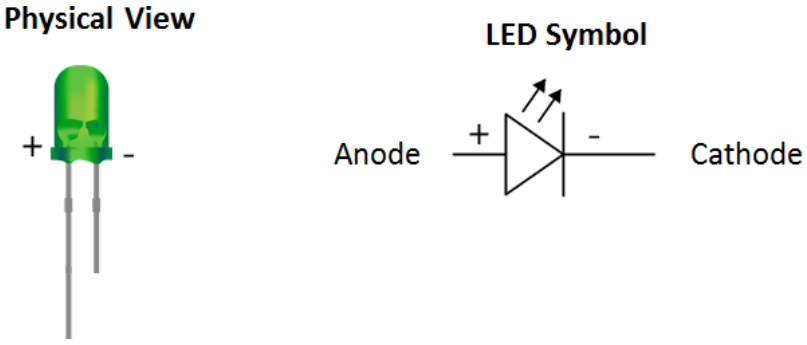
- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino UNO Board
- PC / Laptop
- LED bar module

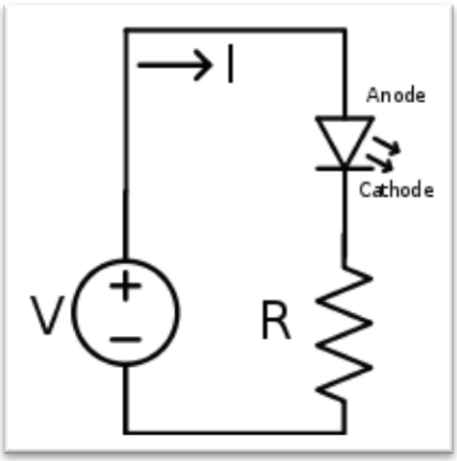
Theory:

Introduction to "LED"

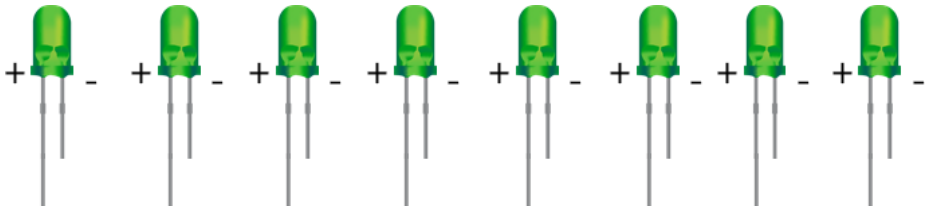


- The full of the LED is Light emitting diode.
- LED is a p-n junction diode.
- It has two terminals named as 'anode (+ve)' and 'cathode (-ve)'.
- When LED is forward biased, it emits light.
- Required Voltage: 1.7 V
- Required Current: 5 to 20 mA

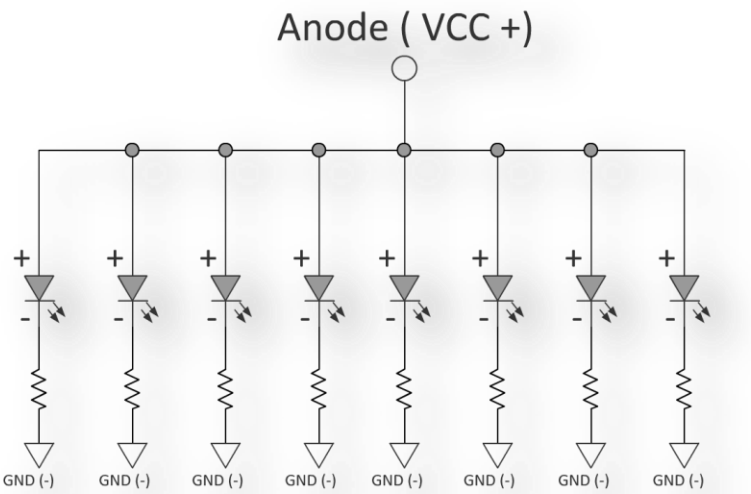
- The Arduino Uno board output current is 30 mA
- So current limiting resistor is connected between the led and the Arduino board while interfacing the led with the Arduino board



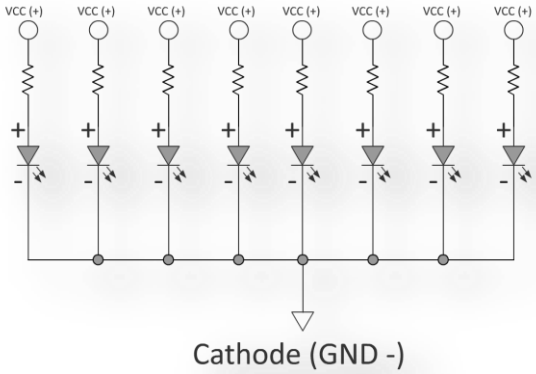
- In LED bar number of LEDs are connected in series (in our case 8 LEDs are connected)



- LED bar has two configurations as
    a. **Common Anode**: anode terminal of all the LEDs are made common and connected to the VCC (+5v). By controlling cathode terminal we can make LED ON or OFF (current sourcing).



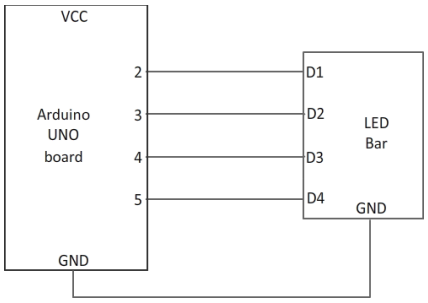
b. **Common Cathode**: cathode terminal of all the LEDs are made common and connected to the Ground (0v). By controlling anode terminal we can make LED ON or OFF (current sinking).



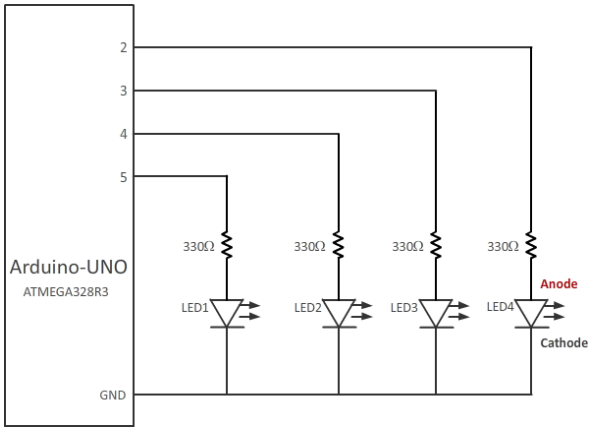
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

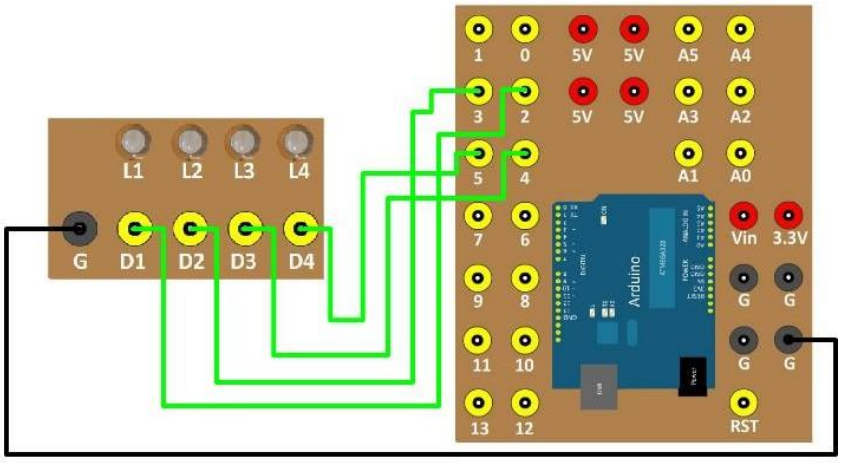
Schematic diagram kit:



Interfacing diagram using breadboard:



Interfacing diagram using IOT kit:



Steps for assembling the circuit:

- Connect the four data pins of the LED bar to any four digital pins of the Arduino board.
- Connect the ground pin (G) of the LED bar module to the ground pin of the Arduino

board

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Declare the integer variables for the four data pins of the led bar.
- In the void setup(), set all the led pins in the OUTPUT mode.
- In the void loop(),
- Make all the pins ‘HIGH’.
- Give delay of one second.
- Make all the pins ‘LOW’.
- Give delay of one second.

Observation:

- Observe the output on LED bar as per the program.

Interfacing the Piezo buzzer with the Arduino board for Generating sound and Music

Objectives:

- To study the working of Piezo buzzer
- To study the pin configuration of Piezo buzzer
- To interface the Piezo buzzer with Arduino board using IOT kit
- To interface the Piezo buzzer with Arduino board using breadboard
- To write a program to generate a beep sound from Piezo buzzer
- To write a program to generate 7 tones of music from Piezo buzzer. For this learn the 'tone' function

Software:

Arduino IDE 1.6.9 or higher

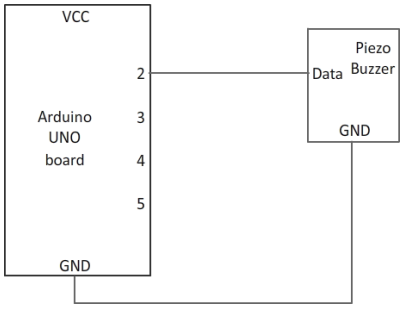
Hardware Modules:

- Arduino Board
- PC / Laptop
- Piezo buzzer

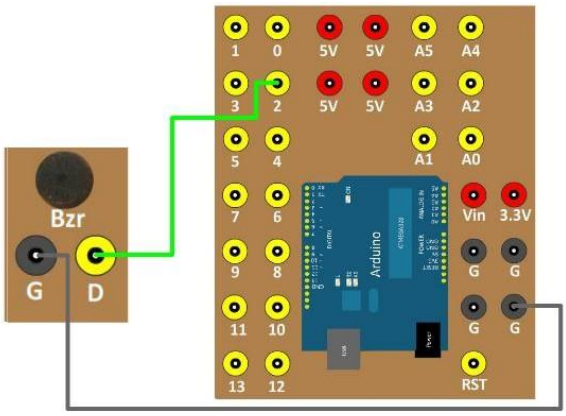
Theory:

- The Piezoelectric buzzer is an electronic device which is commonly used to produce sound/music.
- The Piezoelectric buzzer works on inverse principle of Piezo electricity.
- This buzzer consists of Piezo crystal which consists of a diaphragm.
- When voltage is applied, the diaphragm gets either stretched or compressed and signals of different frequency are generated i.e. sound is generated.
- By using high-performance piezoelectric elements, loud, clear, and pleasant electronic tone can be generated.
- Using this buzzer, we can generate exact musical tone by providing appropriate notes.
- This buzzer is commonly used in refrigerators, typewriters, automotive instruments and security equipment.
- Piezo buzzer has two pins. The pin having '+' sign is called the data pin. It is to be connected to any digital pin of the Arduino board. The pin having '-' sign is called the ground pin. It is to be connected to the ground pin of the Arduino board.

Schematic diagram:



Interfacing diagram using IOT kit:



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Steps for assembling the circuit:

- Connect the Arduino board pins from pin no. 2 to Data pin (red wire) of Piezo buzzer module.
- Connect the ground pin (G) of Arduino board to the ground pin (Black wire) of Piezo buzzer module.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

For interfacing Piezo buzzer

- Start IDE.
- Declare a variable to assign a pin no. of Arduino board.
- In void setup() function, declare that pin as OUTPUT.
- In void loop() function, make that pin digitally HIGH and LOW alternately with appropriate delay.

Algorithm:

For changing frequency through Tone function

- Start IDE.
- Declare a variable to assign a pin no. 2 of Arduino board.
- In void setup() function, declare that pin as OUTPUT.
- In void loop() function, use Tone() function and set the frequency and duration to that pin as “tone(pin no, frequency, duration);”.
- For example, to set the tones as ‘sa, re, ga, ma, pa, dha, ni, sa’ we have to set the tone functions as follows,

  tone(2, 3000, 500), delay (1000)

  tone(2, 3100, 500), delay (1000)

  tone(2, 3200, 500), delay (1000)

  tone(2, 3300, 500), delay (1000)

  tone(2, 3400, 500), delay (1000)

  tone(2, 3500, 500), delay (1000)

  tone(2, 3600, 500), delay (1000)

  tone(2, 3700, 500), delay (1000)

Observation:

Listen the sound as per the program.

Using Serial Monitor of the Arduino IDE

Aim/Objectives:

- To study the concept of Serial Monitor of the Arduino IDE.
- To write a program to display the output of the Arduino Program on the Serial monitor
- To write a program to send input from Serial monitor to the Arduino program and control the actuator.

Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop

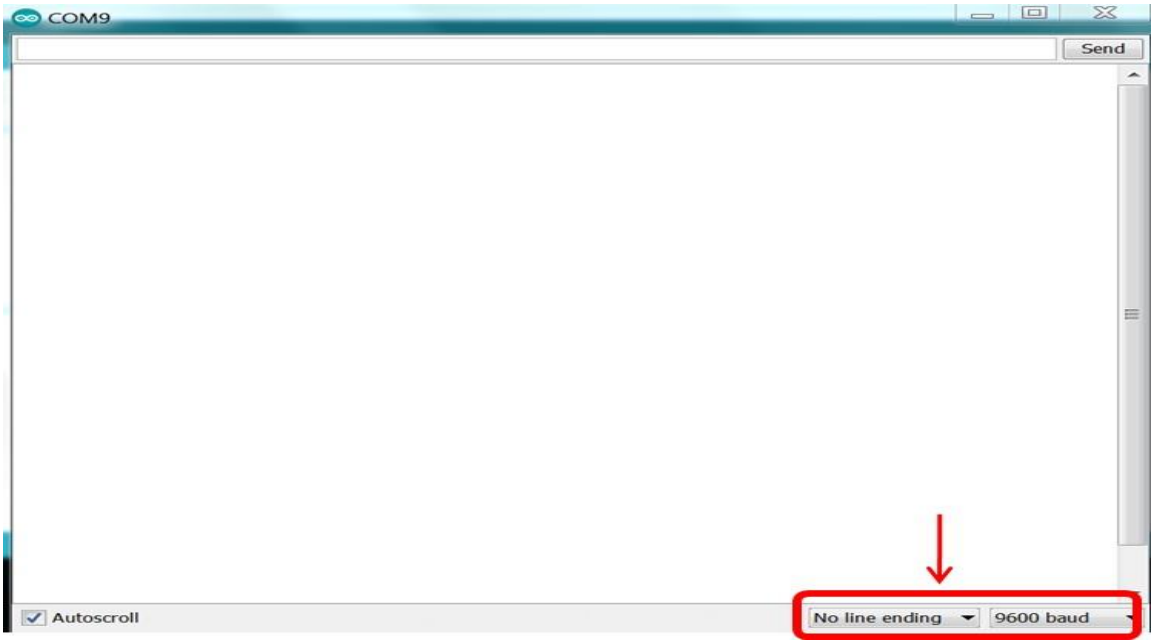
Introduction to "Serial Monitor function"

Introduction to "Serial Monitor function"

- This is a special function of Arduino IDE.
- The Serial monitor is a popup window.
- To open the Serial monitor window from the Arduino IDE, click on the icon given in the right top corner of the Arudino IDE window.

- Data can be sent in both directions.
- Serial data is sent using the USB cable from the Arudino board to Serial monitor and vice versa .
- By using serial monitor, we can debug the Arduino program by displaying the output on the Serial monitor and also can send input from keyboard to the running program.
- To activate the serial monitor function, the Arduino board must be connected to the computer using the USB cable.

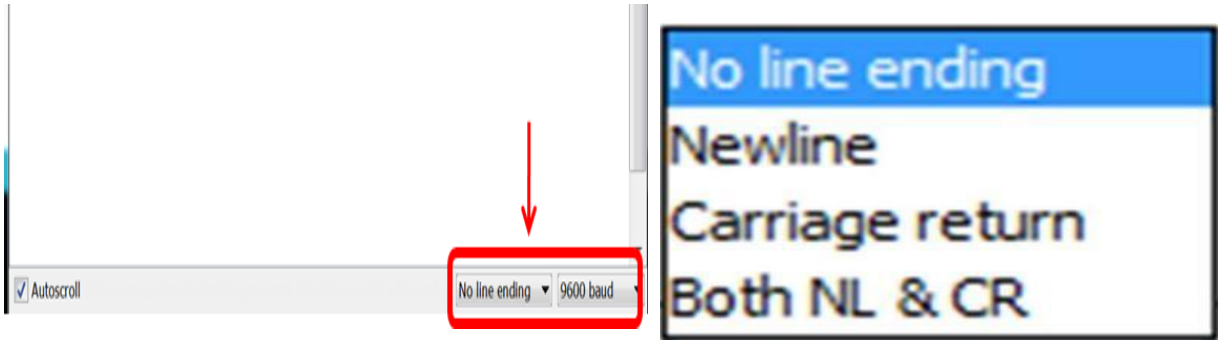
- It is compulsory to set the baud rate on the serial monitor as well as in the program. Otherwise the Serial monitor function will not work.



- For using the serial monitor function, special functions are provided by the Arduino IDE. Each function starts with the word 'Serial'. These functions are,

    a. Serial.begin(): this function is used to set the 'Baud rate'. Standard baud rate used for serial communication is '9600'.
    b. Serial.read(): this function is used to read the data typed on monitor.
    c. Serial.Available(): this function is used to check the buffer on USB driver of Arduino board.
    d. Serial.print(): this function is used to display the message on monitor.
    e. Serial.println(): this function is used to display the message on new line on monitor .
    f. To see the proper output on serial monitor, do the following settings on serial monitor

- One more option is given on the serial monitor to arrange the data on the screen.
- In serial communication, most of the time, at the end of the data we send the characters like "Newline" or "Carriage Return".
- To properly display this type of data on the serial monitor, we have to select one of the options given here. Like,



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Interface diagram:

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the compiled program and check the output.

Algorithm:

Program 1 : To send data from the Arduino board to the Serial monitor and display on it

- Start IDE
- In the void setup(), set the baud rate to '9600'.

- In the void loop(), use the function Serial.println() to display any string on the serial monitor. Pass the string to this function and write it in the double quotes.

Program 2 : To receive data from the Serial monitor in Arduino board program and control the actuator.

- Start IDE,
- Declare a character variable to store the input received from the Serial monitor.
- In the void setup(), set the baud rate to '9600'. Set the pin number 13 in OUTPUT mode.
- In the void loop(), use the function Serial.available() in the 'if' condition to check the buffer of the Arduino board.
- Use the function Serial.read() to read the input from the Arduino board buffer and store the input in the variable.
- Compare the variable with '1' and if the condition is true then make the inbuilt led ON.
- Compare the variable with '0' and if the condition is true then make the inbuilt led OFF.

Observation:

- For the first program, observe the output on monitor
- For the second program, observe the output on the inbuilt led.

Interfacing Push buttons with the Arduino UNO board

Objectives:

- To study the concept of Push button.
- To study the pull-up resistor and pull-down resistor configurations which are used for interfacing the Push button with the Arduino board.
- To interface the Push button with Arduino board.
- To write a program to control the inbuilt LED and the buzzer connected to the Arduino board using Push buttons

Software:

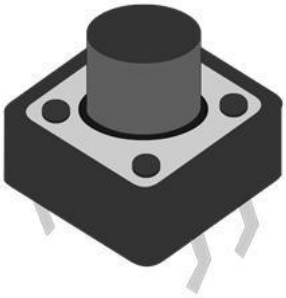
- Arduino IDE 1.6.9 or higher

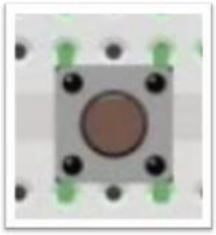
Hardware Modules:

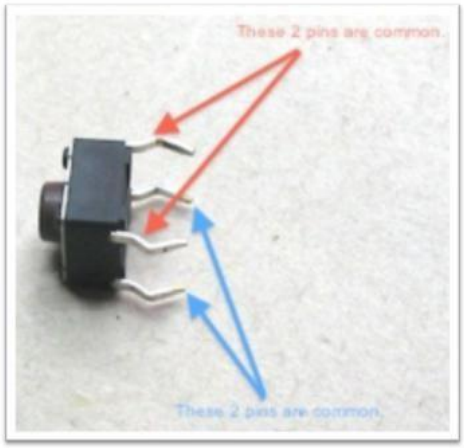
- Arduino Board
- PC / Laptop
- Push buttons, LED, Buzzer

Theory:

Introduction to "Push buttons"

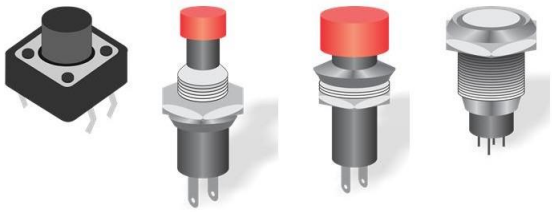




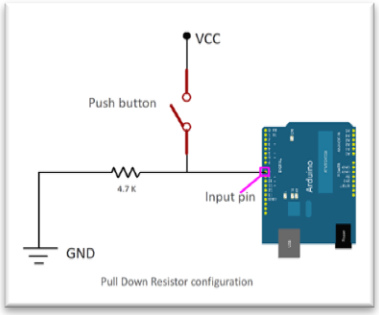


- A Push button or simply button is a simple switch mechanism for controlling some aspects of machine or process.

- Push button is used to make or break the circuit.
- When the push button is pressed, the circuit is closed and current starts flowing through it. Due to this, the devices inserted in the circuit are turned ON. For example, the doorbell, horn, calculators etc.
- In this session we will use the push button, called Momentary Tactile. When this push button is pressed, the circuit temporarily closes and the device is turned ON temporarily. When the push is released, the circuit breaks and the device is turned OFF immediately. The example is doorbell.
- Push button has four pins. But only two pins are used to interface it with the Arduino board. One is for VCC and other is for ground.

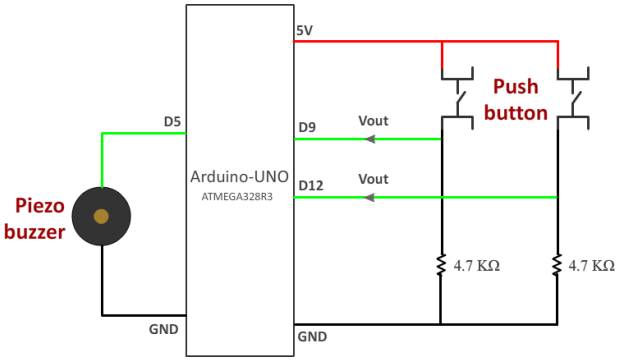


- Push button is a digital sensor.
- So input is sent from push button to the Arduino board.
- So from push button, one more wire is connected to the digital pin of the Arduino board.
- This pin of the Arduino board is declared as the Input pin.
- When push button is not pressed, what input this pin should receive? Yes '0' or low input.
- But in such connection, the input of this pin is not fixed when push button is not pressed. This state of the input pin of any microcontroller is called as floating or undetermined state
- To avoid this floating state of the input pin, when the push button is not pressed, we have the solution of pull-up or pull-down resistor configurations.
- While interfacing the push button with Arduino board, we are going to use the pull-down resistor configuration.
- In this configuration, a resistor is connected between the input pin and the ground pin of the push button. The value of this resistor is approximately 4.7 kilo ohm.

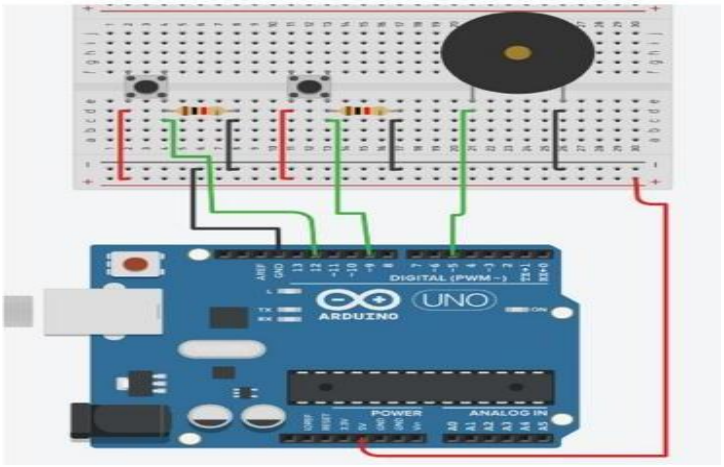


- So, in this configuration, when push button is not pressed, no current flows from VCC terminal to the input pin. So the input pin shows LOW status.
- And when push button is pressed, the input pin directly gets connected to the VCC pin. The current starts flowing from VCC terminal to the input pin. So the input pin shows the 'HIGH' status.

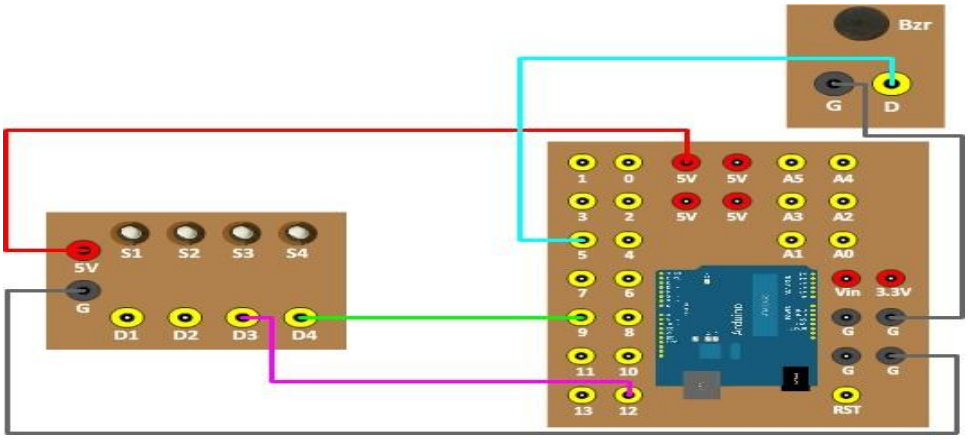
Schematic diagram:



Interfacing diagram using breadboard:



Interfacing diagram using the IOT kit:



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Steps for assembling the circuit:

- Connect the data pin of the 1st Push button to the pin no. 9 of the Arduino board.
- Connect the data pin of the 2nd Push button to the pin no. 12 of the Arduino board.
- Connect the data pin of the buzzer to the pin no. 5 of the Arduino board.
- Connect the VCC pin of the Push buttons to the 5v of the Arduino board.
- Connect the ground pin of the Push buttons to the ground of the Arduino board.
- Connect the ground pin of the buzzer to the ground of the Arduino board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Set the baud rate to '9600'
- Declare four variables as integer for storing the pin numbers of push buttons, buzzer and led.
- Declare two variables as integer for storing the input values of the push buttons.

- Set the push button pins in INPUT mode.
- Set the buzzer and led pins in OUTPUT mode.
- Read the input from both the push buttons using the function digitalRead(). Store the inputs in corresponding variables.
- Compare the first input value with the HIGH value and if the condition is true then make the led ON else make the led OFF.
- Compare the second input value with the HIGH value and if the condition is true then make the buzzer ON else make the buzzer OFF.

Observation:

- Observe the output on the inbuilt led and the buzzer.

Interfacing Temperature & Humidity Sensor (DHT11) with the Arduino UNO board

Objectives:

- To study different types of Temperature and Humidity sensors.
- To study the working of the Temperature and Humidity sensor named DHT11.
- To study the pin configuration of the Temperature and Humidity sensor named DHT11.
- To interface the DHT11 sensor with the Aruduino board.
- To write a program to read the Temperature and Humidity values from the DHT11 sensor and display them on Serial monitor.

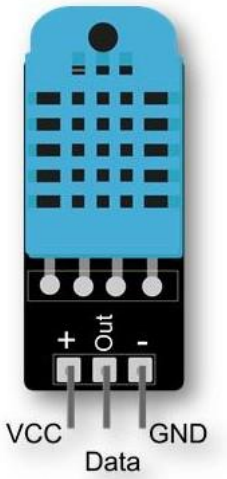
Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop
- Temperature & Humidity sensor (DHT11)

Theory:



DHT11

DHT22

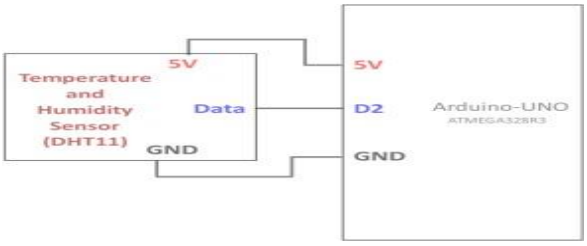
- Physical quantities like Humidity, temperature, pressure etc. are monitored to get information about the environmental conditions.
- Temperature is basically amount of heat present in environment. Humidity is the presence of water vapors in air.

- The Temperature & amount of water vapor in air can affect human comfort as well as many manufacturing processes in industries.
- The presence of water vapor also influences various physical, chemical, and biological processes.
- The sensor which we are going to study measures both Temperature and Humidity of the environment. Such two types of sensors are there. One is DHT11 and other is DHT22.
- The range of the DHT11 sensor is less, so it is used only in school and college projects, while the range of the DHT22 sensor is large, so it is used in Industrial projects for more accuracy.
- The DHT11 sensor has a resistive-type humidity measurement component in which resistivity of semiconductor material changes as per humidity in environment changes.
- This sensor has NTC temperature measurement component which detects the change in temperature.
- This sensor has only one data pin which is used to send both temperature and humidity values to the microcontroller.
- So the One Wire protocol is used to read the values from the sensor to the microcontroller.

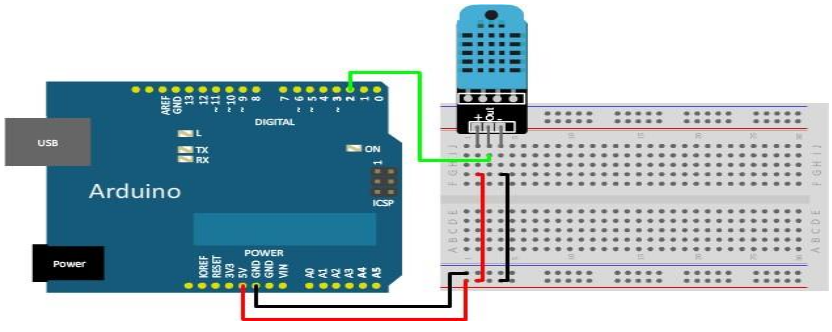
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

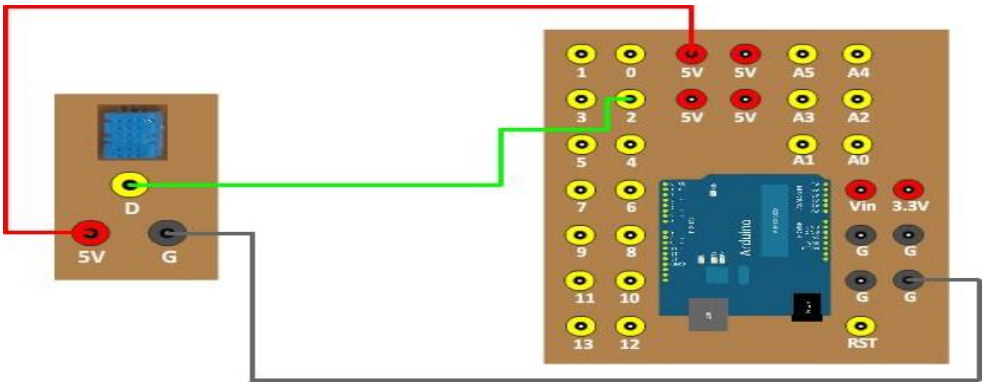
Schematic diagram:



Interfacing diagram using breadboard:



Interfacing diagram using the IOT kit:



Steps for assembling the circuit:

- Connect the data pin of the DHT11 sensor to the digital pin no. 2 of the Arduino board.
- Connect the ground pin (G) of the DHT11 sensor to the ground pin(G) of the Arduino board.
- Connect the VCC pin (5v) of the DHT11 sensor to the 5V pin of the Arduino board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Download DHT.h library. To add this zip library in the Arduino IDE, follow these steps,
    a. Click on the Sketch menu,
    b. Here, click on Include library option,
    c. In the submenu, click on Add zip library option. The windows explorer window opens.
    d. Here select the particular library from the location in your PC and click on the 'Open' button.
    e. The library will be added in the Arduino IDE.
- Include DHT.h library.
- Declare a variable to store the DHT11 sensor pin.
- Use the statement, "#define DHTTYPE DHT11" to declare which type of DHT sensor we are using.

- Use the statement, “DHT dht(sensorPin, DHTTYPE)” to create the object of the DHT library.
- Declare the variable “t” of data type float to store the temperature value
- Declare the variable “h” of data type float to store the humidity value
- In void setup, set the baud rate to “9600”.
- In void loop,
    f. Read and store the temperature value in variable “t” by using function “dht.readTemperature()”
    g. Print the temperature value on serial monitor
    h. Read and store the temperature value in variable “h” by using function “dht.readHumidity()”
    i. Print the humidity value on serial monitor

Observation:

Observe the output on serial monitor to check if the Temperature and Humidity values are shown correct.

Measuring Distance of an object using the Ultrasonic Sensor

Objectives:

- To study the concept of Ultrasound waves.
- To study the working of Ultrasonic sensor HC-SR04
- To study the function pulseIn, which is to be used in the program of Ultrasonic sensor.
- To interface the Ultrasonic sensor with the Arduino board.
- To write a program to measure the distance using the Ultrasonic sensor and display it on serial monitor.

Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop
- Ultrasonic Sensor (HC-SR04 model)

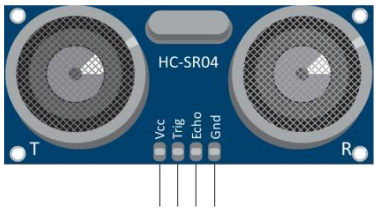
Theory:

- Ultrasonic sensor is one of the most commonly used sensor for distance measurement.
- It works on the principal of ultrasound i.e. it emits the waves whose frequency is greater than 20KHz which is not audible to human ear.
- Hence, ultrasonic waves are used in the measurement of distance just to avoid the interference from surrounding noise which usually falls within 20KHz range.
- In Ultrasonic sensor, it consists of two lenses, one is the transmitter and another is receiver.
- The transmitter (Trigger pin) transmits the ultrasound wave, the wave reaches till the object and reflects back.
- The receiver lens (Echo pin) receives the reflected wave and the duration of this travelling of waves is measured by the Arduino board program.
- Now we know that the velocity of sound (speed) is 340 m/s.
- To use this value we convert it into cm/microsecond (cm/ µs). Therefore $340\text{m/s} = 0.034\ \text{cm/µs}$.
- Using the equation
    - $Speed = \frac{Distance}{Time}$ we derive $Distance = Speed \times Time\ (by\ Arduino)$

- As Arduino measures the duration of the travelling of wave from "transmitter to object + object to receiver", so when we calculate the distance using this $Time\ (by\ Arduino)$,

it is double of the distance of the object from Ultrasound Sensor.

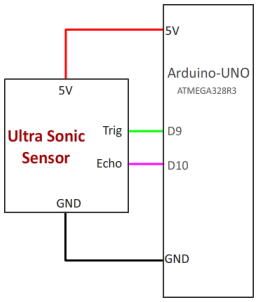
- So to get the actual distance of the object from Ultrasound Sensor, we have to divide the above calculated Distance by 2. Therefore [Distance=(0.034 x time)/2]
- For e.g., we get the value of Time=300 µs, then we get, Distance=(0.034 x 300)/2=5.1 cm
- The range of the Ultrasonics sensor HC-SR04 to measure the distance is **minimum 2 cm and maximum 4 meters (400cm).**
- This means that if your target is within 2cm or beyond 4 meters then this sensor cannot measure the distance. In such case, the output will be shown as negative values.



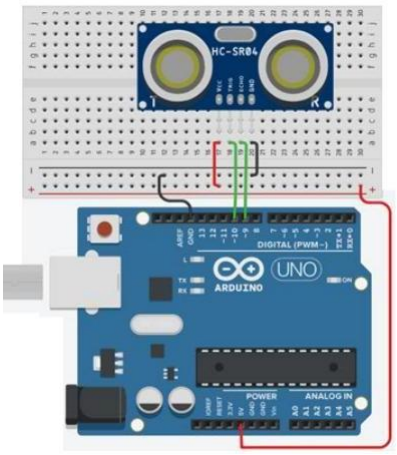
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop using USB cable.

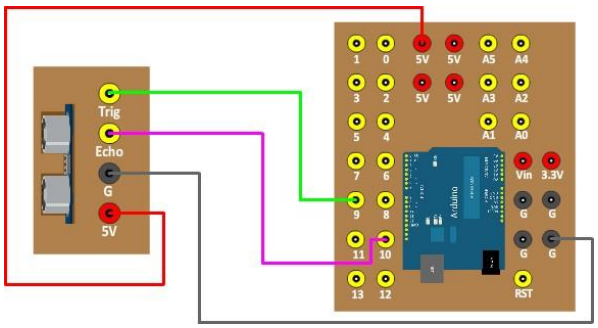
Schematic diagram:



Interface diagram using breadboard:



Interface diagram using IOT kit:



Steps for assembling the circuit:

- Connect the Trig pin of Ultrasonic sensor module to the pin no 9 of the Arduino board.
- Connect the Echo pin of Ultrasonic sensor module to the pin no 10 of the Arduino board.
- Connect the ground pin (G) of Ultrasonic sensor module to the ground pin (G) of the Arduino board.
- Connect the VCC pin (5v) of Ultrasonic sensor module to the VCC pin (5v) of the Arduino board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Initialize two variables ‘TrigPin’ and ‘EchoPin’ and assign pin no. 9 and 10 to them respectively.
- In void setup() function, initialize the baud rate for serial monitor.
- Define pinMode for TrigPin as OUTPUT and EchoPin as INPUT.
- In void loop() function, using digitalWrite(TrigPin, Low), make the Trig pin low.
- In void loop() function, give delay of 2 microseconds.
- In void loop() function, using digitalWrite(TrigPin,HIGH), make the trig pin high.
- In void loop() function, give delay of 10 microseconds
- In void loop() function, using digitalWrite(TrigPin,LOW), make the trig pin low
- Declare the integer variable ‘duration’ and store the value returned by the pulseIn(echo,HIGH) function in it.
- Calculate the distance using formula [Distance=(0.034 x duration)/2]
- Using Serial.println() function display the distance in cm.

Observation:

- Observe the output on serial monitor. Check if the distance shown is correct.

Interfacing LCD with the Arduino UNO board and displaying text on it

Objectives:

- To study the concept of LCD.
- To identify the 16 pins of LCD and study their use
- To study the two registers of LCD
- To interface the LCD with the Arduino board
- To write a program to show the output on LCD at specific place. For this study various functions of LCD.

Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

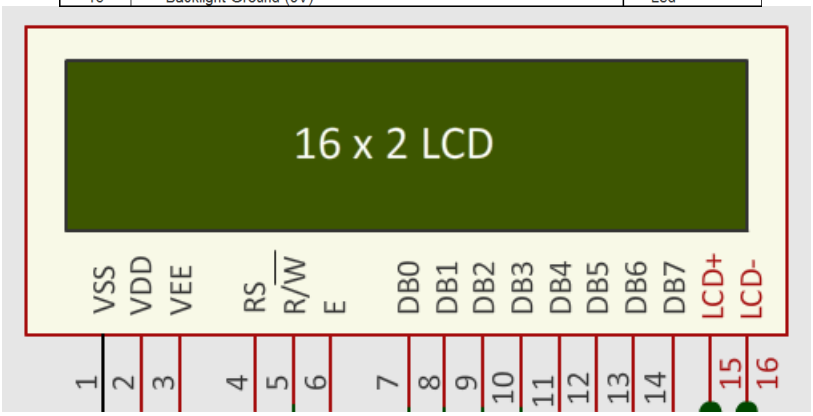
- Arduino Board
- PC / Laptop
- LCD module

Theory:

- A Liquid Crystal Display (LCD) is flat panel display that uses the light-modulating properties of liquid crystals.
- A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits.
- LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.
- A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix.
- This LCD has two registers, namely, Command register and Data register.
- The command register stores the command instructions given to the LCD.
- A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc.
- The Data register used to store the Data which to be displayed on LCD.
- RS pin is used to select one of the register. When the RS pin is low, the command register is selected and when the RS pin is high, the data register is selected.
- The data is the ASCII value of the character to be displayed on the LCD.
- The various pins of LCD are as shown in Pin Description table.

- The 3rd pin is VEE. It is used to adjust the contrast of the LCD to display the output properly on the LCD. A variable resistor or a potentiometer is needs to be connected to this pin while interfacing it to the Arduino board.

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-



- Instead of using 8 data pins to send data from the Arduino board to the LCD, we are using only 4 data pins to reduce the connections.
- The total 8 bits data is divided into two nibbles and then sent to LCD through 4 data pins.
- The **Liquid Crystal Library.h library is used in the LCD program** to control LCD
- Basic functions used from the LCD library are shown in the table below,

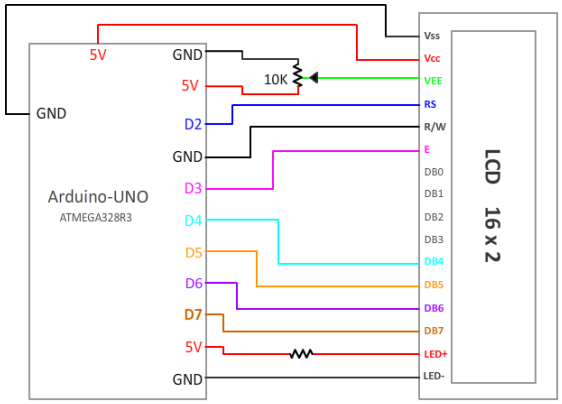
Name of Function	Description	Syntax
LiquidCrystal()	Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. 8 data lines are used for fast displays. In practice usually 4 data lines are used.	LiquidCrystal(rs, enable, d4, d5, d6, d7)
begin()	This function is used to initialize the LCD and also to mention which type of LCD we are using. So we have to pass the parameters as total no. of columns of the LCD and total no. rows of the LCD which we are using.	lcd.begin(cols, rows)
clear()	Clears the LCD screen and positions the cursor in the upper-left corner.	*lcd.*clear()

setCursor()	Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.	lcd.setCursor(col, row)
write()	Write a character to the LCD.	lcd.write(data)
print()	Prints text to the LCD.	lcd.print(data) lcd.print(data, BASE)
scrollDisplayLeft()	Scrolls the contents of the display (text and cursor) one space to the left.	lcd.scrollDisplayLeft()
scrollDisplayRight()	Scrolls the contents of the display (text and cursor) one space to the right.	lcd.scrollDisplayRight()

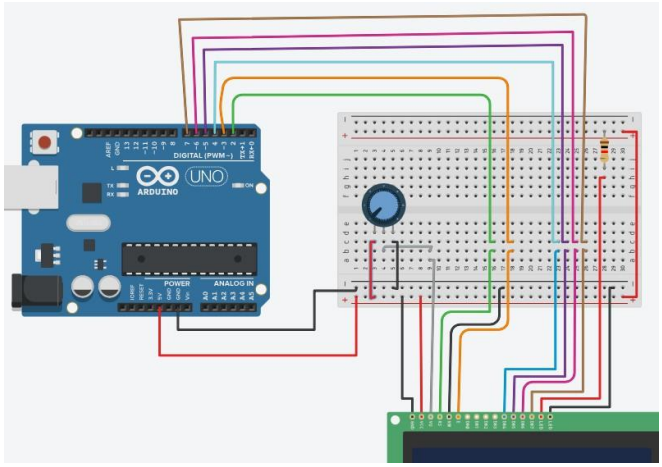
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

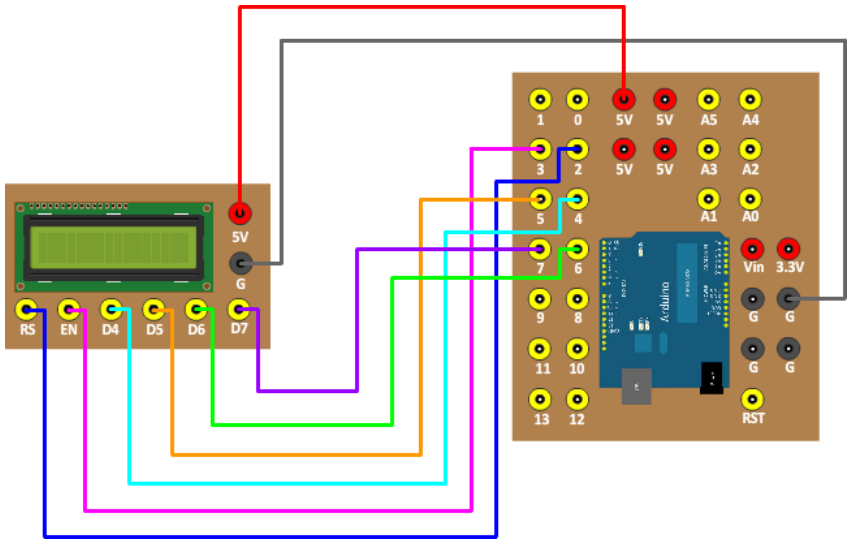
Schematic diagram:



Interfacing diagram using breadboard:



Interfacing diagram using the IOT kit:



Steps for assembling the circuit:

- Connect the Arduino board pins from pin no. 2 to pin no. 7 to the LCD Module pins, RS, Enable (E), D4, D5, D6, D7 respectively.
- Connect the ground pin (G) of the Arduino board to the ground pin (G) of LCD Module.
- Connect the VCC pin (5v) of the Arduino board to the VCC pin (5v) of LCD Module.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the LCD module to the Arduino board.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Include the library 'LiquidCrystal.h'
- Create the object LCD of the class LiquidCrystal and assign all the pin number to the object as LiquidCrystal lcd(2,3,4,5,6,7);
- In void setup() function, initialize the 16x2 LCD by using the function lcd.begin(16,2);
- In void loop() function, clear the screen by using the function lcd.clear();
- In void loop() function, set the cursor at position 1st row and 1st column position (i.e. 0,0), by using the function lcd.setCursor(0,0);

- Enter the string which is to be displayed on LCD, by using the command Lcd.print(“Welcome All”);

Observation:

Observe the output LCD module as per the program.

Interfacing Light sensor (LDR) with the Arduino UNO board

Objectives:

- To study the working of the LDR sensor
- To interface the LDR sensor with the Arduino board.
- To write a program to control the LED using the LDR sensor.

Software:

- Arduino IDE 1.6.9 or higher

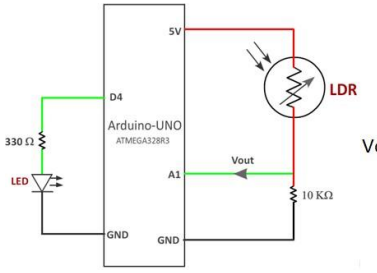
Hardware Modules:

- Arduino UNO Board
- PC / Laptop
- LDR sensor module

Theory:

- The full form of the LDR is Light Dependent Resistor.
- It consists of a material called cadmium sulfide. It is shown by the zigzag lines.
- LDR sensor has two leads or pins.
- LDR sensor works on the principle of “Photo-Conductivity”.
- As the intensity of the light falling on the LDR sensor increases, the resistance of the sensor decreases.
- Inversely, as the intensity of the light falling on the LDR sensor decreases, the resistance of the sensor increases.
- This happens due to the principle of “photo-conductivity”.
- As per this principle, the light has photon energy.
- More the light intensity, more the photon energy.

- So when the light having more intensity falls on the LDR sensor, then due to more photon energy, the charged carriers of the LDR sensor increase.
- And as the charged carriers increase, the resistance of the LDR sensor decreases. •
- LDR is an analog sensor. So we connect it to any one of the analog input pins (A0 to A5) of Arduino board.
- In LDR sensor, as per the change in light intensity, the value of its resistance changes. So the LDR sensor gives the output, in the form of resistance. But in Arduino, on its GPIO pins, it can receive the input only in the form of voltage. So the input coming from LDR sensor, which is in the form of resistance has to be converted into the voltage form. To do this conversion, while interfacing the LDR sensor with Arduino, a voltage divider circuit is build.



Voltage divider circuit

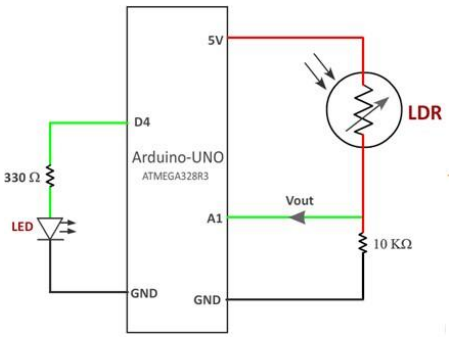
- In LDR sensor, as per the change in light intensity, the value of its resistance changes. So the LDR sensor gives the output, in the form of resistance. But in Arduino, on its GPIO pins, it can receive the input only in the form of voltage. So the input coming from LDR sensor, which is in the form of resistance has to be converted into the voltage form. To do this conversion, while interfacing the LDR sensor with Arduino, a voltage divider circuit is build.
- As shown in the diagram, one terminal of the LDR is directly connected to the ground pin of Arduino.
- To the other terminal of LDR, one 10Kohm resistor is connected in series with LDR. The other end of the resistor is connected to the 5V supply of the Arduino.
- Between the resistor and the LDR, one wire is connected and it is further connected to the analog input pin of Arduino. This is the output of the LDR in the form of voltage. Itis called as V-out. The value of V-out depends on the value of resistance of resistor andthe LDR.
- In this circuit, if the intensity of the light falling on LDR increases, the V-out decreases.
- And if the intensity of the light falling on LDR decreases, the V-out increases.
- The ATMEGA328 IC of Aduino board has 10 bit Analog to Digital converter(ADC). So the analog input recived on these analog input pins (A0 to A5) is converted into digital form. So we get digital output of these analog sensors.

- As the ADC is of 10 bits, the total digital values are 210=1024. But the digital value starts from 0. So we get the digital values from 0 to 1023.
- So here, for 0 volt signal, we get digital value 0.
- For 5 volt signal, we get digital value 1023
- For 2.5 volt signal, we get digital value 512.
- In this way we get different digital values for the analog signal varying from 0V to 5V.
- So from LDR sensor, we get output value ranging from 0 to 1023.

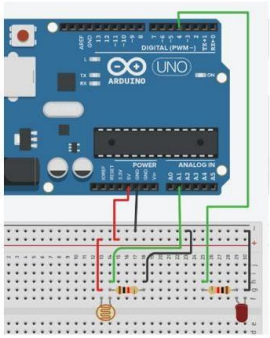
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

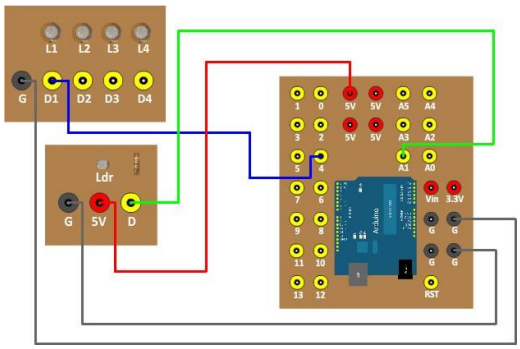
Schematic diagram:



Interfacing diagram using breadboard:



Interfacing diagram using the IOT kit:



Steps for assembling the circuit:

- Connect the A1 pin of the Arduino board to the Data pin (D) pin of the LDR module.
- Connect the ground pin (G) of the Arduino board to the ground pin (G) of the LDR module.
- Connect the VCC pin (5v) of Arduino board to the VCC pin (5v) of the LDR module.
- Connect the digital pin no 4 of the Arduino board to the LED pin of the LED bar.
- Connect the ground pin (G) of the Arduino board to the ground pin (G) of the LED bar.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Declare a variable to store the analog pin that is LDR pin of the Arduino board.
- Declare a variable to store the analog value from LDR module.
- Declare a variable to store the threshold value for LDR module.
- Declare a variable to store the digital pin that is LED pin of the Arduino board.
- In void setup() function, set the baud rate to 9600 using function Serial.begin(9600) for serial monitor.
- In void setup() declare the Pinmode of the LDR pin as INPUT and declare the pinmode of the LED pin as OUTPUT.

- In void loop() function, to read the analog value from LDR, use the function ‘analogRead(pinno)’.
- In void loop() function, to print the sensor value on serial monitor.
- In void loop() function, use the if loop to compare the sensorvalue with the threshold value. If sensorvalue is greater than threshold then make the ledpin HIGH so as to make the LED ON, else make the ledpin LOW so as to make the LED OFF.
- Give some delay.

Observation:

- Observe the output on serial monitor. The output should be between the values 0 to 1023. Also observe the output on the LED.

Interfacing and controlling Direction of Servo Motor using Arduino board

Objectives:

- To study the working of the Servo motor
- To interface the servo motor with the Arduino board.
- To write a program to rotate the servo motor in specific angle.

Software:

Arduino IDE 1.6.9 or higher

Hardware Modules:

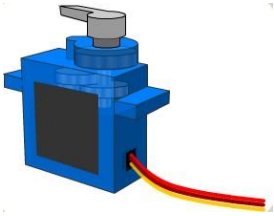
Arduino Board

PC / Laptop

Servo Motor module

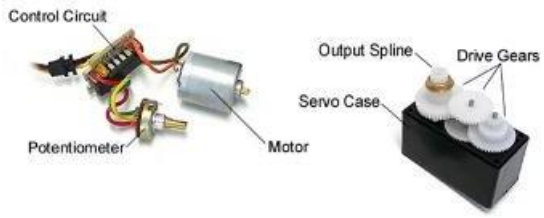
Theory:

- A Servo motor is an actuator which is linear or rotary in motion.
- A Servo motor allows a precise control of angular or linear position, velocity and acceleration.
- It consists of gears and output shaft.
- The servo motor is controlled by sending a signal of Pulse Width Modulation (PWM) and its output shaft can be rotated between angle $0^0 to\ 180^0$.
- To rotate the output shaft, we give the specific value of angle in degrees as input to servo motor. The shaft moves at the same given angle, whatever may be the current position.
- The 'servo.h' libray is used in the program to send the output in degrees to the servo motor.



- The external parts of the servo motor are as follows,
    a. Output shaft,
    b. Data wire in orange color
    c. VCC wire in red color,
    d. Ground wire in brown color.

- The internal parts of the servo motor are as follows,

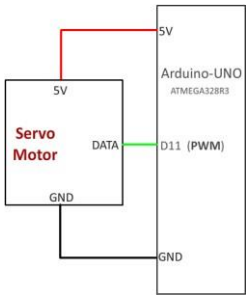


e. DC motor,
f. Drive gear system,
g. Output shaft,
h. Control circuit, and
i. Potentiometer or Position sensor. This sensor is connected to the output shaft.

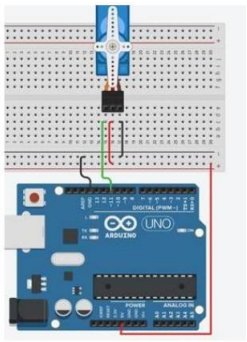
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop using USB cable.

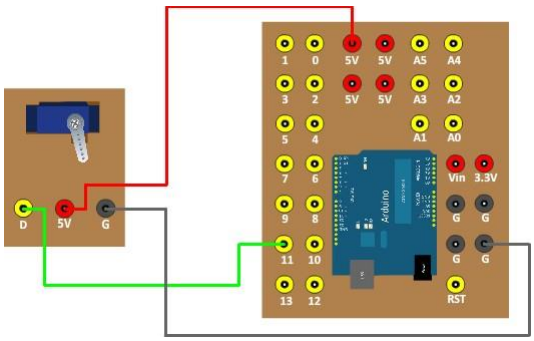
Schematic diagram:



Interfacing diagram using breadboard:



Interfacing diagram using the IOT kit:



Steps for assembling the circuit:

- Connect the DATA pin of Servo Motor to pin no.11 (PWM) of the Arduino board.
- Connect the ground pin (G) of the Servo Motor to the ground pin (G) of the Arduino board.
- Connect the VCC pin (5V) of the Servo Motor to the 5V pin of the Arduino board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Include the library Servo.h. It is given in the Arduino IDE. So need to download it.
- Create an object of class Servo by naming as 'myservo'.
- In void setup() function, to initialize the servo motor's data pin, use the function myservo.attach(pin no.).
- In void loop() function, to give input angle to servo motor, use the function myservo.write(angle in degree).
- Give delay of appropriate time.
- Again give input angle to the servo motor.
- Give delay

Observation:

Observe the movement of the output shaft of the Servo motor as per the given angle.

Interfacing the RFID Reader with Arduino board and read the UID of the RFID Tag

Objectives:

- To study the concept of the RFID system.
- To study the different types of RFID tags.
- To interface the RFID reader with the Arduino board.
- To write a program to read the RFID tag and display its Unique ID on the Serial monitor. Also, compare the Unique ID with the Unique ID which is stored in the program and if they match, turn the led ON

Software:

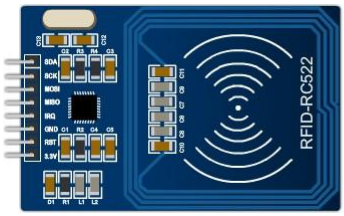
- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop
- RFID Reader
- RFID Tag (2)

Theory:

- The full form of the RFID is Radio Frequency Identification.
- In the RFID system, the wireless communication takes place between the RFID reader and RFID tag.

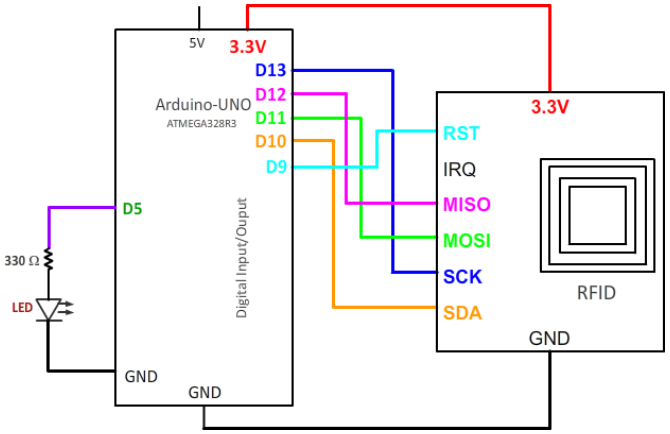


- For wireless communication, radio waves of different frequencies are used.
- Depending on the frequency used, the RFID system is classified into three different types as
    - Low Frequency RFID system
    - High Frequency RFID system
    - Ultra-high frequency RFID system
- We are using the High Frequency RFID system with 13.56 megahertz (MHz) frequency
- RFID Tags are classified into three types as
    - Active RFID Tag
    - Passive RFID Tag
    - Semi-passive RFID Tag
- We are going to study the Passive RFID tag
- This passive RFID tag works on 13.56 mega hertz frequency
- So in this system, the RFID readers and RFID tags are tuned to the 13.56 mega hertz frequency.
- Each RFID tag contains a unique ID. The size of this ID is 4 bytes
- The RFID Reader contains an IC named MFRC522
- The RFID Reader requires only maximum 3.3 volt supply. If more than 3.3 volts supply is given, then the RFID Reader will be damaged
- The range of this RFID reader to read the tag is 3 cm.
- This is a passive RFID system. So when the tag comes in the range of the RFID Reader, the tag gets power from the reader and becomes active. Then the tag sends its unique ID to the RFID Reader.
- RFID Reader reads the unique Id (UID) and sends it to the Arduino board.
- To establish the proper communication between the RFID Reader and the Arduino board, the **SPI protocol** is used.
- This protocol is also called as '**four wires protocol'** because it requires four wires for communication
- On each microcontroller board, the **SPI protocol pins are fixed**
- So when the SPI protocol is to be used for the communication between the Arduino UNO board and the any SPI device then the Arduino UNO board pin and the SPI device pins must be connected as follows,
    - The **MOSI** pin must be connected to the digital pin number **11** of the Arduino UNO board.
    - The **MISO** pin must be connected to the digital pin number **12** of the Arduino UNO board.
    - The **SCK** pin must be connected to the digital pin number **13** of the Arduinov board.
    - The CS (SS) pin must be connected to the digital pin number 10 of the Arduino UNO board. If more slaves are there then more CS pin will be required.

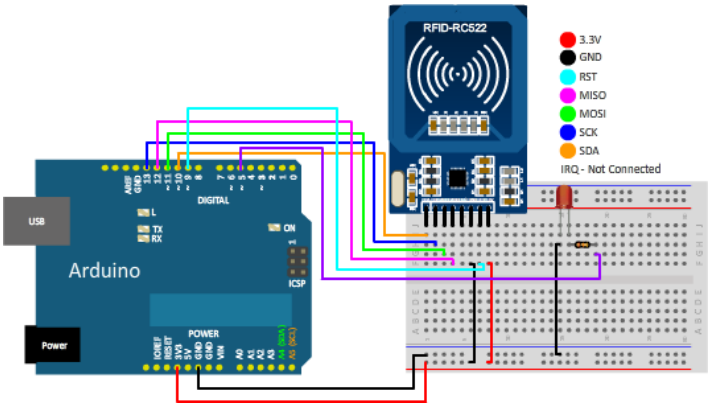
Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

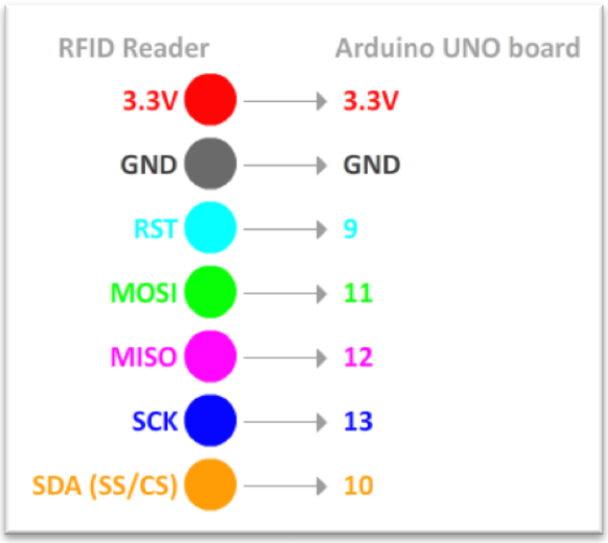
Schematic diagram (SPI protocol):

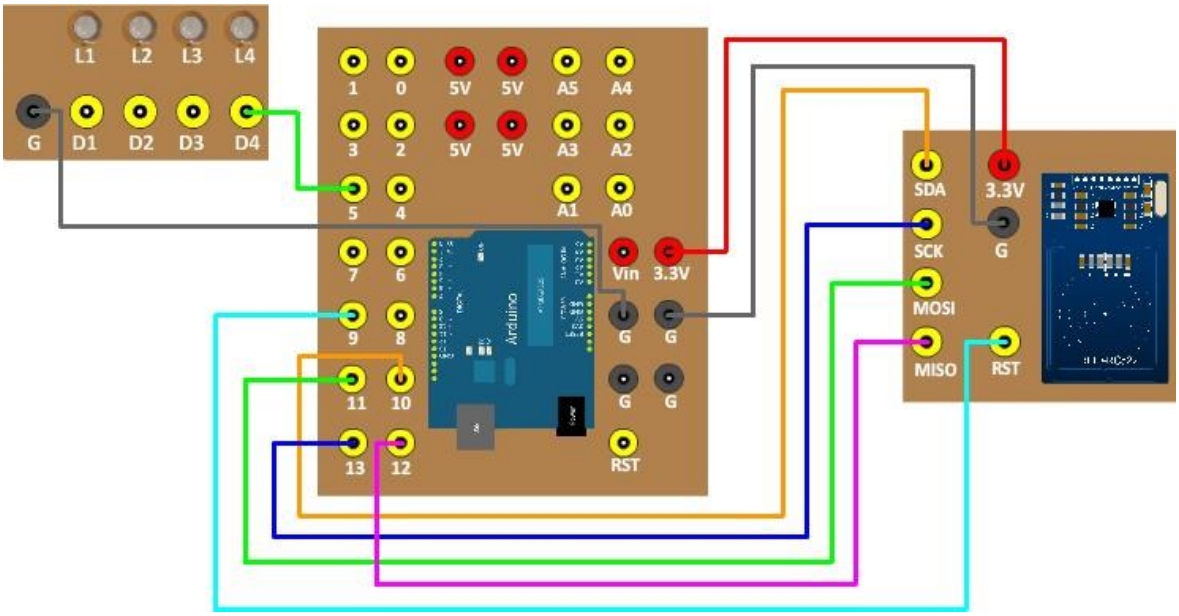


Interfacing diagram using the breadboard (SPI protocol):



Interfacing diagram using the IOT kit (SPI protocol):





Steps for assembling the circuit:

- The RFID reader has total 8 pins.
- The first pin of the RFID reader is SDA. It is the CS or SS pin of the SPI protocol. So it is connected to the digital pin number 10 of the Arduino board.
- The second pin of the RFID reader is SCK. So it is connected to the digital pin number 13 of the Arduino board.
- The third pin is MOSI. So it is connected to the digital pin number 11 of the Arduino board.
- The fourth pin is MISO. So it is connected to the digital pin number 12 of the Arduino board.
- The fifth pin is IRQ. Right now it is not used. So it is not connected.
- The sixth pin is Ground. It is not connected to the ground pin of the Arduino board.
- The 7th pin is RST means Reset. After reading the RFID tag from the Reader, the Arduino board resets the RFID reader using this RST pin. Due to this reset, the RFID reader becomes ready to read the next tag. This pin is to be connected to any digital pin of the Arduino board. Here it is connected to the digital pin number 9 of the Arduino board.
- The 8th pin is VCC. Always remember that the RFID reader works on maximum 3.3 volt supply. So connect this VCC pin to 3.3 volt pin of the Arduino board. By mistake, if you connect it to the 5 volt pin then the RFID reader will be damaged.
- In this practical, we have also used a led.
- The data pin of the led is connected to the digital pin number 5 of the Arduino board.
- The ground pin of the led is connected to the ground pin of the Arduino board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Firstly, download the 'rfid master' library from the website, 'github.com'. This library is downloaded in the zip format. It contains the file MFRC522.h. So include the MFRC522.h library.
- Include the SPI.h library. It is given in the Arduino IDE. So no need to download it.
- Declare three variables to store the pin numbers of SDA, RST, and led pins.
- Create an object of the class MFRC522. Pass the SDA and RST pins as the parameters to this object.
- Declare an integer variable to use in the 'For loop'.
- In the void setup(),

- Set the baud rate as 9600 to use the Serial monitor.
- Call the SPI.begin() function to initialize the SPI communication.
- Use the function , “mfrc522.PCD_Init()” to initialize the RFID Reader.
- Set the mode of the ledpin as OUTPUT.
- In the void loop() function,
- Use the function, “!mfrc522.PICC_IsNewCardPresent” in the ‘if’ condition to check if the RFID tag is present near the RFID reader.
- When the tag is detected, again use the if condition and the, “!mfrc522.PICC_ReadCardSerial()” in it to read all the 4 bytes of the tag.
- Use the ‘For loop’ to display all the 4 bytes on the Serial monitor.
- In the for loop, compare the integer variable with “mfrc522.uid.size” to print all the 4 bytes.
- In the Serial.print function, pass the parameter. ‘mfrc522.uid.uidByte*i+’.
- After displaying each byte give space so as to separate two bytes.
- For example, the Unique ID of the RFID Tag will be displayed as follows,
- 220 space 182 space 122 space 37
- In the next program, compare the each byte with the bytes of your tag.
- If all the bytes match then turn the LED ON for 5 seconds and then make it OFF.

Observation:

- See the UID of your RFID Tag on the Serial monitor.
- Compare multiple RFID tags and check for which tag the led turns ON.

Interfacing Proximity (IR) sensor with Arduino Uno

Objectives:

- To study the Proximity Sensor.
- To study how to Interface Proximity sensor using Arduino board.

Software:

- Arduino IDE 1.6.9 or higher

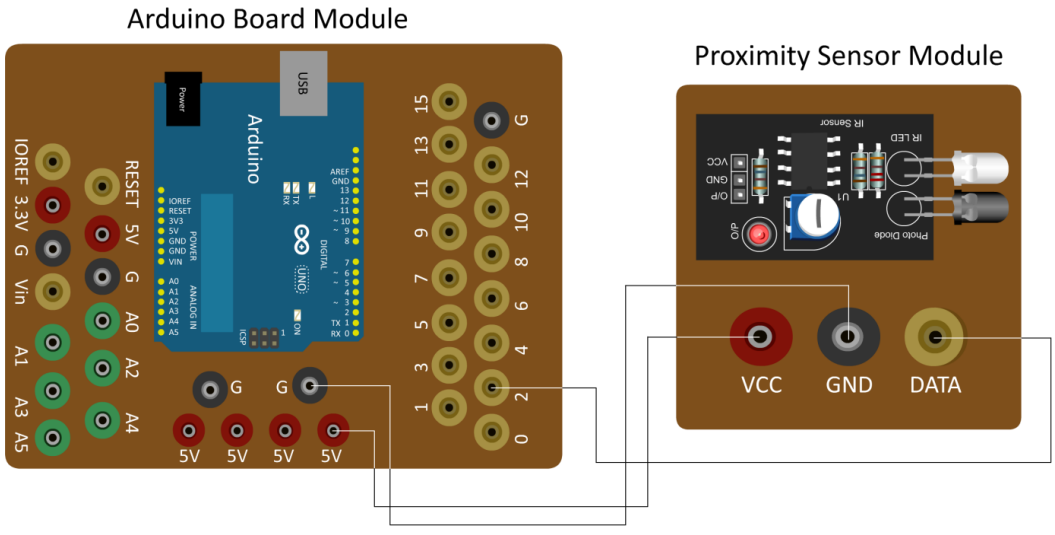
Hardware Modules:

- Arduino Board
- PC / Laptop
- Proximity IR Sensor (module)

Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Interfacing diagram:



Steps for assembling the circuit:

- Connect the digital pin no 2 of Arduino board to the DATA pin of Proximity sensor module.
- Connect the ground pin (G) of Arduino board to the ground pin (G) of Proximity sensor module.
- Connect the VCC pin (5v) of Arduino board to the VCC pin (5v) of Proximity sensor module.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Initialize a variable 'sensorPin' and assign pin no. 2 of Arduino (digital pin) to it.
- Initialize a variable 'led1Pin' and assign pin no. 13 of Arduino (digital pin) to it
- Initialize a variable 'sensorValue' to store the proximity sensor value.
- In void setup() function, initialize the baud rate for serial monitor.
- Define pinMode for sensorPin as INPUT.
- Define pinMode for led1Pin as OUTPUT
- In void loop() function, using digitalread(sensorPin), read the value of proximity sensor.
- Print the value on serial monitor.
- Use if condition to compare the sensorValue. When the value is high, make the led1Pin HIGH so as to ON the LED and if the value is low, make the led1pin LOW so as to OFF the LED.

Observation:

- Observe the output on serial monitor as per the program.

Interfacing and controlling Direction of Stepper Motor using Arduino UNO board.

Objectives:

- To study how to operate Stepper motor using Arduino board
- To study how to control Speed and Direction of Stepper Motor

Software:

- Arduino IDE 1.6.9 or higher

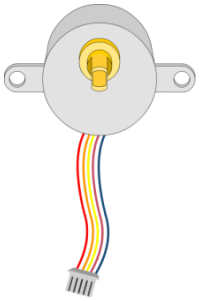
Hardware Modules:

- Arduino Board
- PC / Laptop
- Stepper motor
- Driver circuit

Theory:

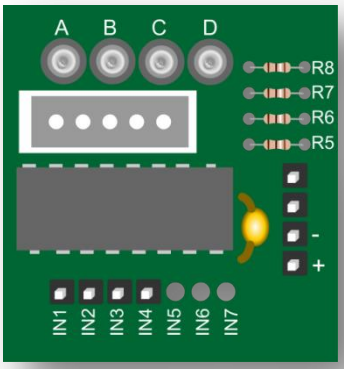
- Stepper motor is an electromechanical device which converts electrical energy into mechanical movements.
- Stepper motor is a brushless DC electric motor that divides a full rotation into a number of equal steps.
- Due to unique design of stepper motor, it can be controlled to a high degree of accuracy without any feedback mechanisms.
- The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils.
- The coils are charged positively and negatively in a specific sequence, precisely moving the shaft forward or backward in small "steps".
- Typical types of stepper motors can rotate $2^0, 2.5^0, 5^0, 7.5^0 and\ 15^0$ per input electrical pulse.
- The inner magnet of stepper motor is effectively divided into many separate sections, which look like teeth on a gear wheel.
- The electromagnets and the output shaft of stepper motor are arranged in such a way that when we give train of 8 pulses, the output shaft completes its one rotation.
- The speed of the motor shafts rotation is directly related to the frequency of the input pulses.

- The length of rotation is directly related to the number of input pulses applied.
- A stepper motor can be a good choice whenever controlled movement is required. They can be used to advantage in applications where you need to control rotation angle, speed, position and synchronism.
- Some of these include Robotics, printers, plotters, high-end office equipment, hard disk drives, medical equipment, fax machines, automotive and many more.



1. Blue wire: Coil 4
2. Pink wire: Coil 3
3. Yellow wire: Coil 2
4. Orange wire: Coil 1
5. Red wire: 5v

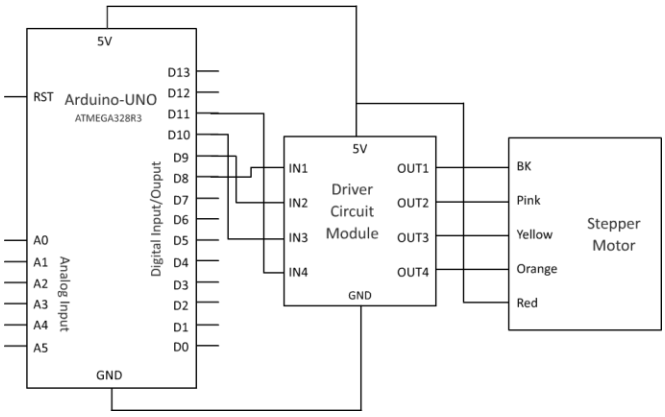
- To drive Stepper motor it requires high current (more than 150m amp).
- So we connect the driver circuitry between Raspberry-Pi board and the stepper to boost the current that passes through the stepper motor.
- And as per the change in current, the speed of stepper motor changes.



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Schematic diagram:



Steps for assembling the circuit:

- Connect the pin no 8 to 11 of Arduino board to the IN1, IN2, IN3 and IN4 pins of Driver circuitry respectively.
- Connect the ground pin (G) of Arduino board to the ground pin (G) of Driver circuitry.
- Connect the VCC pin (5v) of Arduino board to the VCC pin (5v) of Driver circuitry.
- Connect the Stepper motor to the Driver circuitry.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

INPUT: Bit pattern or (Set of pulses).

OUTPUT: Rotation of stepper Motor (stepwise).

- Start IDE.
- Declare the variables ‘motorPin1’, ‘motorPin2’, ‘motorPin3’, ‘motorPin4’ and assign pin no. 8 to 11 respectively.
- Set the pinMode of all the pins as OUTPUT.
- For clockwise rotation of the motor,
    a. First make the motorPin1 as HIGH and make remaining 3 pins as LOW.
    b. Give delay of 2 milisec
    c. Then make the motorPin2 as HIGH and make remaining 3 pins as LOW.

d. Give delay of 2 milisec
e. Then make the motorPin3 as HIGH and make remaining 3 pins as LOW.
f. Give delay of 2 milisec
g. Then make the motorPin4 as HIGH and make remaining 3 pins as LOW.
h. Give delay of 2 milisec

- For Anticlockwise rotation, make the pins HIGH in reverse order.
- Stop.

Observation:

- Observe the output on serial monitor as per the program.

Interfacing Mechanical relay and controlling high voltage devices

Objectives:

- To study how to interface relay with Arduino.
- To study how to control high voltage devices using relay and Arduino

Software:

- Arduino IDE 1.6.9 or higher

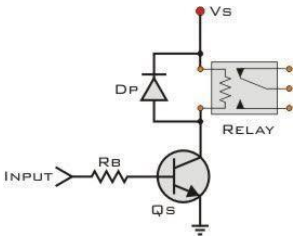
Hardware Modules:

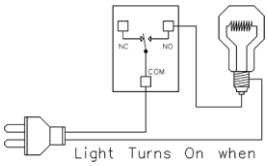
- Arduino Board
- PC / Laptop
- Relay module

Theory:

- A Relay is used to control the high voltage devices using Arduino.
- It is an electrically operated switch. Relays mostly use electromagnetic mechanism and operate many circuits from single signal.
- The relay can handle high power required to directly control electric circuits. (for e.g. Fan, Bulb)
- There are various types of relay, for e.g. Mechanical, Solid state, SCR etc.
- Out of these we are using Mechanical relay.
- As shown in the dig., there are total 6 pins on relay module. They are as Input, Ground, VCC, Normally Open (NO), Common, Normally Closed (NC).
- Practically there are two requirements. In one requirement, the device is always OFF and it is required to make it ON through Arduino. In such case we use the Normally Open Circuit.
- In the second requirement, the device is always ON and it is required to make it OFF through Arduino. In such case Normally Closed circuit is used.
- In our practical we are using Normally Open (NO) circuit. Means the Lamp is always OFF and when required we are making it ON through Arduino.
- To make the Lamp ON, one terminal of Lamp is connected to NO pin. Second terminal of the Lamp is connected to 230V supply.
- There is a lever which is connected to the Common terminal of the relay. To make the Lamp ON, the lever should be connected to the Normally Open (NO) terminal.

- To control the lever the Transistor is used, it is shown in black colour on the relay board. Emitter of the transistor is connected to ground. Base of the transistor is connected to the Data pin of the Arduino through RB register. Collector of the Transistor is connected to one terminal of the coil.
- Internally there is a coil in the Relay. Internally one terminal of the coil is connected to VCC and other terminal is connected to the collector of transistor. A freewheeling diode is connected across the coil in the reverse biased mode.
- When input is given to the Base terminal of the Transistor, current starts flowing through the coil, magnetic field is generated around it. Due to this magnetic field the lever is pulled down towards the NO terminal and the circuit gets completed and the Lamp gets ON.

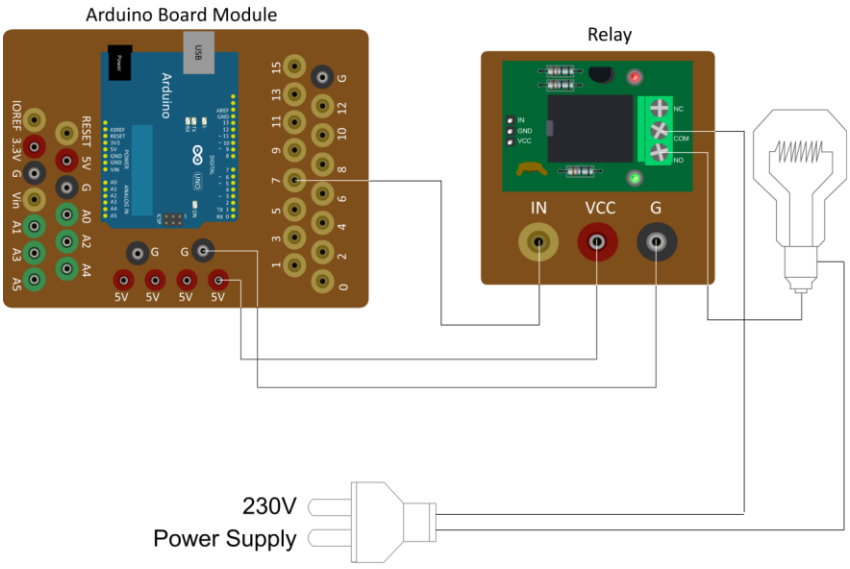




Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Interfacing diagram:



Steps for assembling the circuit:

1. Connect the pin no. 7 of Arduino board to the INPUT pin of Relay module.
2. Connect the ground pin (G) of Arduino board to the ground pin (G) of Relay Module.
3. Connect the VCC pin (5v) of Arduino board to the VCC pin (5v) of Relay Module.
4. Connections of the Lamp module and Relay module are as follows,
    a. One terminal of the Battery (230 V) is directly connected to one terminal (A) of Lamp module.
    b. Other terminal of the Battery is connected to the Common terminal of the Relay module.
    c. Normally Open (NO) terminal of Relay is connected to the other terminal (B) of the Lamp module.

Procedure:

5. Write the program as per the algorithm given below.
6. Save and compile the program.
7. Connect the Relay module to the Arduino board.
8. Connect the Lamp module to the Relay module
9. Connect the Arduino board to PC/Laptop using USB cable.
10. Upload the program and check the output.

Algorithm:

11. Start IDE
12. Declare the variable ‘lamp1Pin’ to assign the pin number 7
13. Configure the pin number ‘lamp1Pin’ as ‘OUTPUT’ pin
14. Make the ‘OUTPUT’ as ‘HIGH’
15. Give delay of ‘10’ second
16. Make the ‘OUTPUT’ as ‘LOW’
17. Give delay of ‘10’ second

Observation:

18. Observe the ON and OFF states of the Lamp as per the program.

Interfacing Seven Segment Display with Arduino UNO board

Objectives:

- To study the interfacing of Seven Segment Display (SSD) with Arduino and display numbers 0 and 1 on it.

Software:

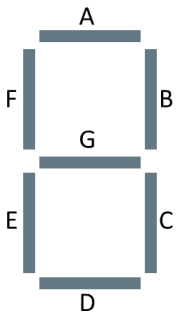
- Arduino IDE 1.6.9 or higher

Hardware Modules:

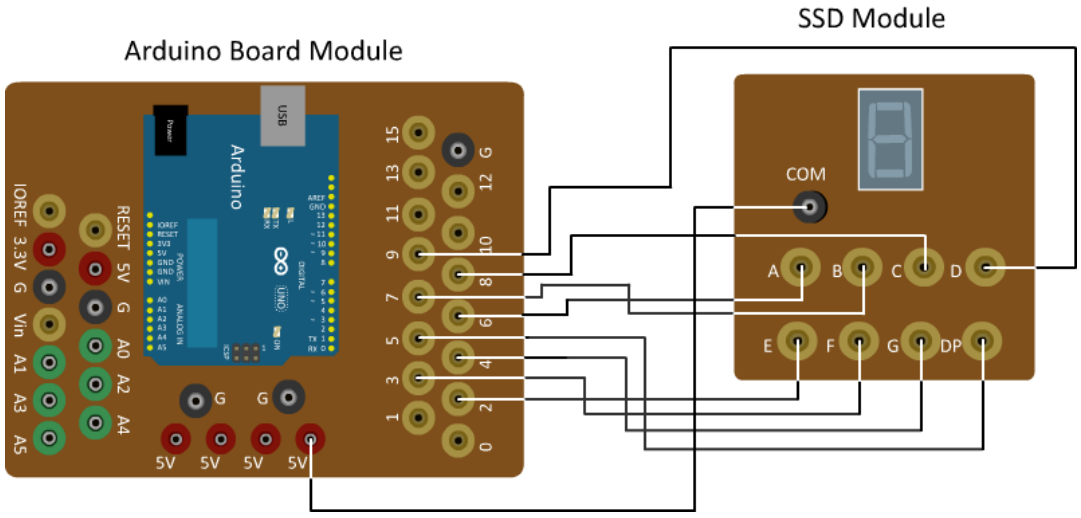
- Arduino Board
- PC / Laptop
- SSD module
- Switch (Push button) Module

Theory:

- A Seven Segment Display is a form of electronic display device for displaying Decimal numerals i.e. an alternative to the more complex dot matrix displays.
- It can be arranged so that different combinations can be used to make alphanumeric characters.
- The Seven Segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle and bottom.
- The segments of Seven Segment Display are referred to by the letters A to G where the optional decimal point is used for the display of non-integer numbers.



Interfacing diagram:



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Steps for assembling the circuit:

- Connect the Arduino board pins from pin no. 2 to pin no. 9 to the SSD module pins from pin no. A to pin no. DP (or H) serially
- Connect the VCC pin (5v) of Arduino board to the 'COM' pin of SSD module.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Declare variables for all the pins of Seven Segment Display (SSD).
- Set Arduino board pins from pin no. 2 to pin no. 9 as OUTPUT
- To display the number '0' on SSD, make the pins A To F as HIGH and make the pin G as LOW.

- Give delay.
- Make all the pins as LOW.
- To display the number ‘1’ on SSD, make the pins A, D, E, F, G as HIGH and make the pin B and C as LOW.
- Give delay.
- Make all the pins as LOW.

Observation:

- Observe the output on SSD module as per the program.

Interfacing and Reading a Potentiometer using Arduino

Objectives:

- To study the interfacing of Potentiometer.

Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop
- Potentiometer module
- LED bar Module

Theory:

- A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value.
- By turning the shaft of the potentiometer, we change the amount of resistence on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" that pin to 5 volts and ground, giving us a different analog input.
- When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and we read 0 & when the shaft is turned all the way in the other direction, there are 5 volts going to the pin and we read 1023.
- So we get the output voltage (that we see on Serial monitor) from potentiometer in numbers ranging from 0 to 1023 that is proportional to the amount of voltage being applied to the pin between 0 to 5V.
- We can convert this number to Volt by the formula

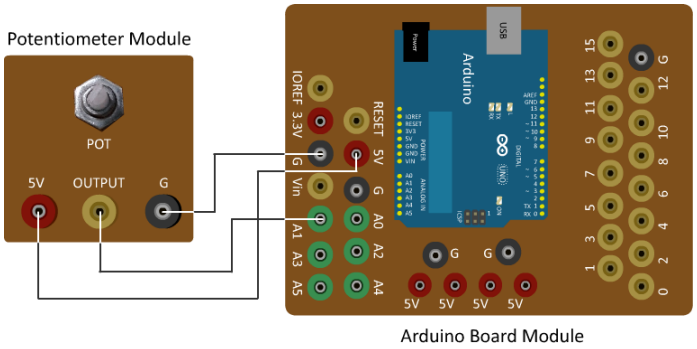
$$v = \frac{5}{1023} \times p$$

Where, v is the value in Volt, P is the analog reading from potentiometer in numbers between 0 to 1023.

Now we get, $\frac{5}{1023} = 0.00489$

So we can rewrite the above formula as $v = 0.00489 \times p$

Schematic diagram:



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Steps for assembling the circuit:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Start IDE.
- Declare required variables.
- Set the baud rate to 9600.
- Set Arduino board pin A1 as OUTPUT.
- Read the OUTPUT from potentiometer by varying the potentiometer.
- Display the above reading on serial monitor.

Observation:

- Observe the output on RGB module as per the program.

Write an application using Arduino for traffic signal monitoring and control system.

Objectives:

- Understand the working of traffic signal.
- Interface 12 LED's to Arduino board.
- Program the Arduino board to control the traffic signal.

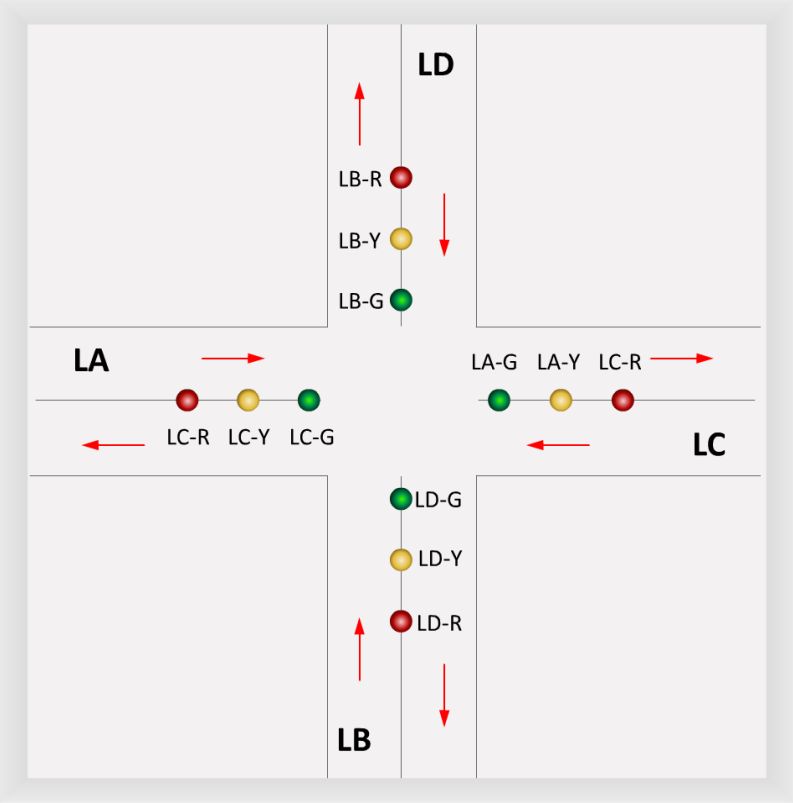
Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino Board
- PC / Laptop
- 12 LED's
- 12 resistors of 220 ohm

Theory:

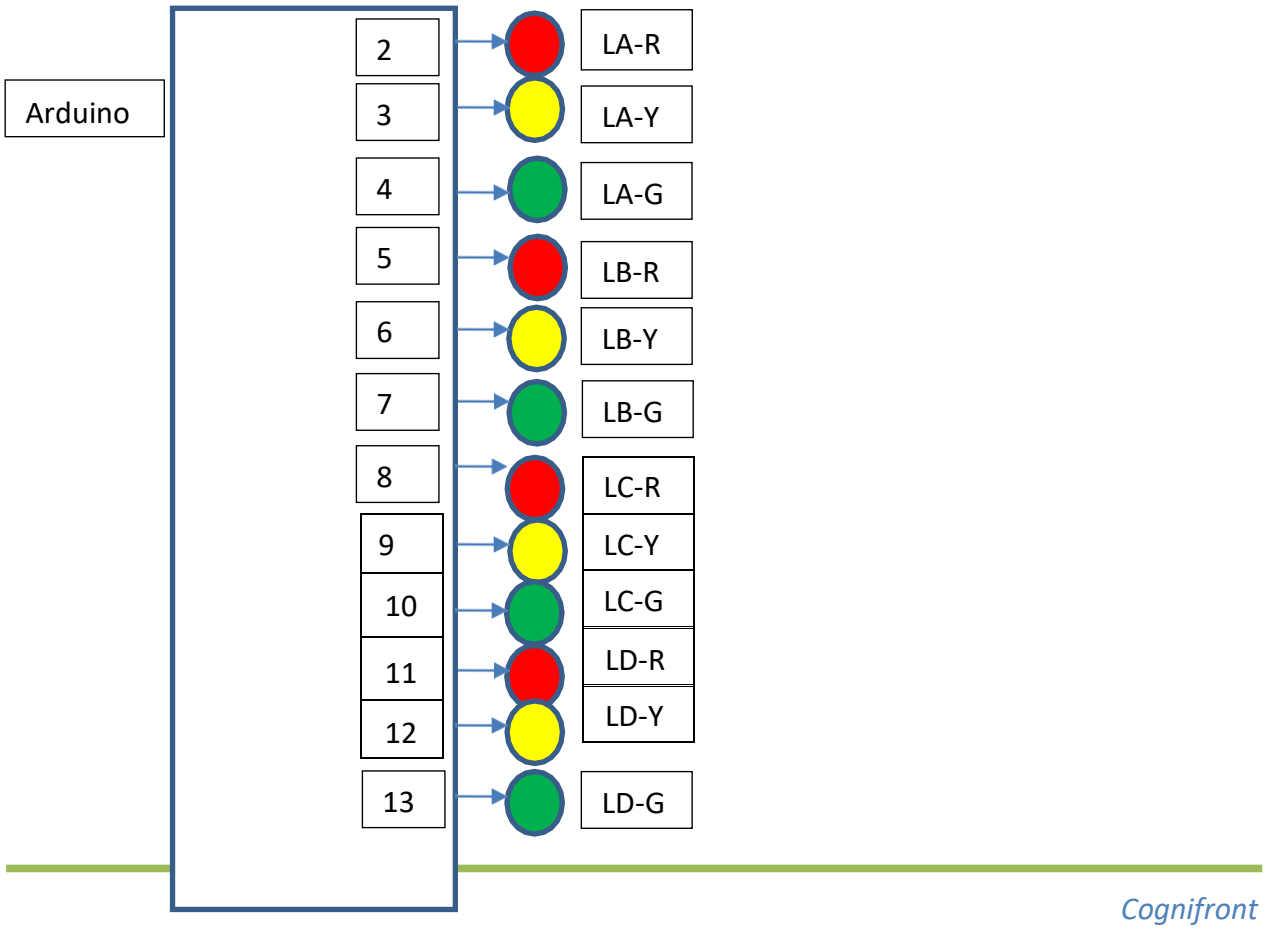


- A simple traffic light system for a 4 way intersection is implemented using Raspberry pi where the traffic is controlled in a pre-defined timing system.
- There are 4 lanes LA, LB, LC and LD going towards the signal.
- At the cross road there are 4 sets of Traffic lights opposite to each lane.
- These sets are LA(LA-G, LA-Y, LA-R), LB(LB-G, LB-Y, LB-R), LC(LC-G, LC-Y, LC-R), LB(LD-G, LD-Y, LD-R),
- Traffic from any lane moves when its corresponding Green light is ON.
- "ON time" of any Red light is dependent on the "ON time" of Yellow light and Green light of other 3 signal lights.
- "ON time" of Yellow light is same for all lanes.
- User can specify and change the "ON time" of the Green light and Red light of each signal separately.
- The Traffic light pattern keeps on repeating till the next change made by the user.

Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Schematic diagram:



Steps for assembling the circuit:

- Connect the R, Y, G pins of “Lane A” to 2, 3, 4 pins of Arduino Board respectively.
- Connect the R, Y, G pins of “Lane B” to 5, 6, 7 pins of Arduino Board respectively.
- Connect the R, Y, G pins of “Lane C” to 8, 9, 10 pins of Arduino Board respectively.
- Connect the R, Y, G pins of “Lane D” to 11, 12, 13 pins of Arduino Board respectively.
- Connect the “COM” pin of the Traffic Signal module to the GND pin of Arduino Board.

Procedure:

- Write the program as per the algorithm given below.
- Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Declare the 12 leds data pins.
- Declare pinmode of 12 leds as output.

- Firstly for the Lane-A, the signal becomes Green.
- Hence, for all other Lanes, the corresponding Red signal is on.
- As a warning indicator, the Yellow light in Lane-A is tuned on indicating that the red light is about to light up. Similarly, the yellow light in the Lane-C is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-C, the signal becomes Green. So at the same time the signal for Lane-A becomes Red.
- As a warning indicator, the Yellow light in Lane-C is tuned on indicating that the red light is about to light up. Similarly, the yellow light in the Lane-D is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-D, the signal becomes Green. So at the same time the signal for Lane-C becomes Red.
- As a warning indicator, the Yellow light in Lane-D is tuned on indicating that the red light is about to light up. Similarly, the yellow light in the Lane-B is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-B, the signal becomes Green. So at the same time the signal for Lane-D becomes Red.
- As a warning indicator, the Yellow light in Lane-B is tuned on indicating that the red light is about to light up. Similarly, the yellow light in the Lane-A is also turned as an indication that the green light about to be turned on.
- The system then loops back to Lane 1 where the process mentioned above will be repeated all over again.

Observation:

Observe the output on LEDs.

Program:

```
byte R1 = 2;

byte Y1 = 3;

byte G1 = 4;


byte R2 = 5;

byte Y2 = 6;

byte G2 = 7;


byte R3 = 8;

byte Y3 = 9;

byte G3 = 10;


byte R4 = 11;

byte Y4 = 12;

byte G4 = 13;


int Lane_A[] = {R1, Y1, G1}; // Lane 1 Red, Yellow and Green

int Lane_B[] = {R2, Y2, G2}; // Lane 2 Red, Yellow and Green

int Lane_C[] = {R3, Y3, G3}; // Lane 3 Red, Yellow and Green

int Lane_D[] = {R4, Y4, G4}; // Lane 4 Red, Yellow and Green
```

```
void setup()
{
 for (int i = 0; i < 3; i++)
 {
   pinMode(Lane_A[i], OUTPUT);
   pinMode(Lane_B[i], OUTPUT);
   pinMode(Lane_C[i], OUTPUT);
   pinMode(Lane_D[i], OUTPUT);
 }
 for (int i = 0; i < 3; i++)
 {
   digitalWrite(Lane_A[i], LOW);
   digitalWrite(Lane_B[i], LOW);
   digitalWrite(Lane_C[i], LOW);
   digitalWrite(Lane_D[i], LOW);
 }
}

void loop()
{
 digitalWrite(Lane_A[2], HIGH);//LaneA Green ON
 digitalWrite(Lane_A[0], LOW); //LaneA Red OFF
 digitalWrite(Lane_C[0], HIGH);//LaneA Red OFF
 digitalWrite(Lane_D[0], HIGH);//LaneA Red OFF
```

```
digitalWrite(Lane_B[0], HIGH);//LaneA Red OFF
delay(7000);

digitalWrite(Lane_A[2], LOW); //LaneA Green OFF
digitalWrite(Lane_A[1], HIGH);//LaneA Yellow ON
delay(3000);

digitalWrite(Lane_A[0], HIGH);//LaneA Red ON
digitalWrite(Lane_A[1], LOW); //LaneA Yellow OFF
digitalWrite(Lane_B[0], LOW); //LaneB Red OFF
digitalWrite(Lane_B[2], HIGH);//LaneB Green ON
delay(7000);

digitalWrite(Lane_B[2], LOW); //LaneB Green OFF
digitalWrite(Lane_B[1], HIGH); //LaneB Yellow ON
delay(3000);

digitalWrite(Lane_B[0], HIGH); //LaneB Red ON
digitalWrite(Lane_B[1], LOW);  //LaneB Yellow OFF
digitalWrite(Lane_C[0], LOW); //LaneC Red OFF
digitalWrite(Lane_C[2], HIGH); //LaneC Green ON
delay(7000);

digitalWrite(Lane_C[2], LOW);  //LaneC Green OFF
```

```
  digitalWrite(Lane_C[1], HIGH); //LaneC Yellow ON
  delay(3000);


  digitalWrite(Lane_C[0], HIGH); //LaneC Red ON
  digitalWrite(Lane_C[1], LOW);  //LaneC Yellow OFF
  digitalWrite(Lane_D[0], LOW);//LaneD Red OFF
  digitalWrite(Lane_D[2], HIGH);//LaneD Green ON
  delay(7000);


  digitalWrite(Lane_D[2], LOW); //LaneD Green OFF
  digitalWrite(Lane_D[1], HIGH); //LaneD Yellow ON
  delay(3000);
  digitalWrite(Lane_D[1], LOW); //LaneD Yellow OFF
}
```

Interfacing GSM (SIM900A) module with Arduino UNO board

Objectives:

- To study the interfacing of GSM (SIM900A) module with Arduino UNO board.
- To write a program to send SMS to mobile from the GSM module.
- To write a program to receive SMS from mobile to the GSM module.
- To send the DHT11 sensor data to the ThingSpeak Cloud using GSM module.

Software:

- Arduino IDE 1.6.9 or higher

Hardware Modules:

- Arduino UNO Board
- PC / Laptop
- GSM (SIM900A) module

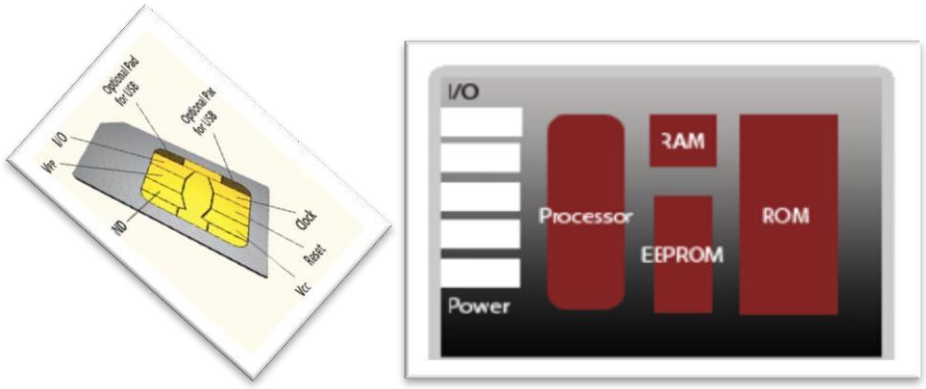
Theory:

- The full form of GSM is Global System for Mobile.
- It is a second generation cellular standard. That means we can insert a SIM card in this GSM module, which we insert in our mobile phone and can do all the activities which we do using mobile phone.
- It includes the integration of voice services and data delivery using digital modulation.
- Using this module and the microcontroller like Aruduino and Raspberry-Pi, we can,
    1. Make a call to the mobile or receive a call from the mobile.
    2. Send SMS to the mobile or receive SMS from mobile.

3. Send data to the cloud or receive data from the cloud using GPRS facility.

- Nowadays we use ESP8266 or ESP32 or any other wifi Modules to send any sensor data to the Internet wirelessly. Hence Wifi comes into action and thus we need Wifi Connection for wireless communication with any server. But the disadvantage of using Wifi is, it is not available everywhere. The wifi signal is limited to certain locations and to a certain range up to a few meters. For example, in order to use IoT Connectivity and to get data from the farmer's fields, we can't rely on Wifi due to unavailability. Similarly forest, river zone, mountains are the areas where wifi connection is not available.
- So, GSM GPRS is the only alternative left as per the present scenario and current technology. GSM GPRS Module allows you to add location-tracking, voice, text, SMS, and data to your application. The big advantage of GSM/GPRS Connectivity is, it covers a wide area and signal/connectivity is available almost everywhere.
- **General Packet Radio Services (GPRS) is a best-effort packet-switching protocol for wireless and cellular network communication services.**
- SIM900A Modem works on frequencies 900/ 1800 MHz. It can search these two bands automatically. The frequency bands can also be set by AT Commands. The baud rate is configurable from 1200-115200 through AT command.
- This SIM900A GSM module has three parts as
    a. SIM card
    b. GSM module and
    c. Antenna

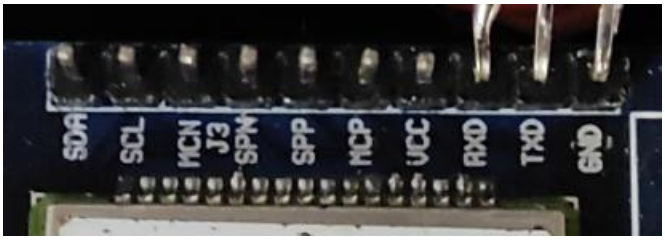
SIM card



- **Subscriber Identification Module (SIM)** is a smart card included in every cell phone, this same card is to be used in this GSM module.
- There is a flat connector having 6 or 8-pin embedded on the top of the card.
- It also has a full-fledged microcomputer with an OS.

- A SIM Cardholder is given on the GSM module to insert the SIM card as shown in the diagram.
-  

GSM Module



- This module has total Ten pins. Out of them, we use three pins to interface it with the microcontroller. These pins are
    a. Transmitter (TXD) pin
    b. Receiver (RXD) Pin and
    c. Ground Pin
- To give power supply to this module, a separate black barrel connector is given. Connect this connector to 12V-2A power supply.
- Here, pin number 4 is the VCC which can be connected with the Arduino's 5volts if we want to give the power supply from Arduino board.
- But here we have given the external power supply, so we have left this pin as unconnected.
- So we will connect,
- Pin number 1 that is the ground pin to the ground pin of the Arduino board.

- Pin number 2 that is the TXD pin to the RXD pin of the Arduino board.
- Pin number 3 that is the RXD pin to the TXD pin of the Arduino board
- The UART protocol is used to interface this GSM module with the microcontroller, so for serial communication, the TXD and RXD pins of this module are connected to the RXD and TXD pins of the Arduino board.
- Note that, there is a cross connection between the RXD and TXD pins of the GSM module and the Arduino board. This is because when Arduino board transmits data using TXD pin, the GSM board receives this data using RXD pin and vice versa.

Antenna

- This GSM module has SMA connector to connect the external antenna.

Insert a SIM card in the GSM module

- This board works with networks that means it requires 2G SIM card.
- So, A (2G) SIM Card, or a 4G or 5G SIM Card with 2G backward compatible is to be inserted.
- **Jio SIM Cards won't work with this GSM board as they are not compatible with 2G network. So do not use Jio SIM card.**

Network status LED

- Three LEDs are given on this GSM module.
- The LED named as “NWK” is used to show the network connectivity of the SIM card.
- If the SIM is not connected to the network then this LED blinks fast that is after every second. And when SIM is connected to the network, then this LED blinks slowly that is after every 3 seconds.

AT commands used in the program of GSM module

Basic AT Command

There some basic AT commands which are used as follows,

Command	Description	Response
AT	Checking communication	OK
ATE0 or ATE1	Turn off /ON echo	OK
ATI	Product Identification	e.g.: SIM900 R11.0 OK
AT+GSN	Product serial number identification (IMEI)	IMEI no.
AT+GMI	Manufacturer Name	Manufacturer Name E.g. SIMCOM_LTD
AT+GMM	Model Identification	Model no. E.g. SIMCOM_900A
AT+CSMINS?	SIM Inserted Status Reporting	+CSMINS: <n>, <SIM inserted> 0: not inserted 1: inserted
AT+CSPN?	Get Service Provider Name from SIM	Service Provider Name E.g. +CSPN: “!dea”, 1

Calling

To use calling service following AT commands are used.

Command	Description	Response
ATD<Mob. No.>;	Dial / call a number	OK (if successful), BUSY (if busy), NO CARRIER (if no connection)
ATA	Answer a call	OK
ATH	Hang up call	OK
ATDL	Redial Last Telephone Number Used	OK (if successful), BUSY (if busy),NO CARRIER (if no connection)

If you want to send the SMS to mobile phone from GSM without program, then

1. **Insert the SIM card in the GSM board.**
2. **Connect the GSM to the microcontroller board; also connect to the proper power supply.**
3. **Wait till the SIM gets connected to the network.**
4. **Then type following commands on the serial monitor.**

AT+CMGF=1 <ENTER>

AT+CMGS="1638740161" <ENTER> //Replace the phone number

The modem will respond with: >

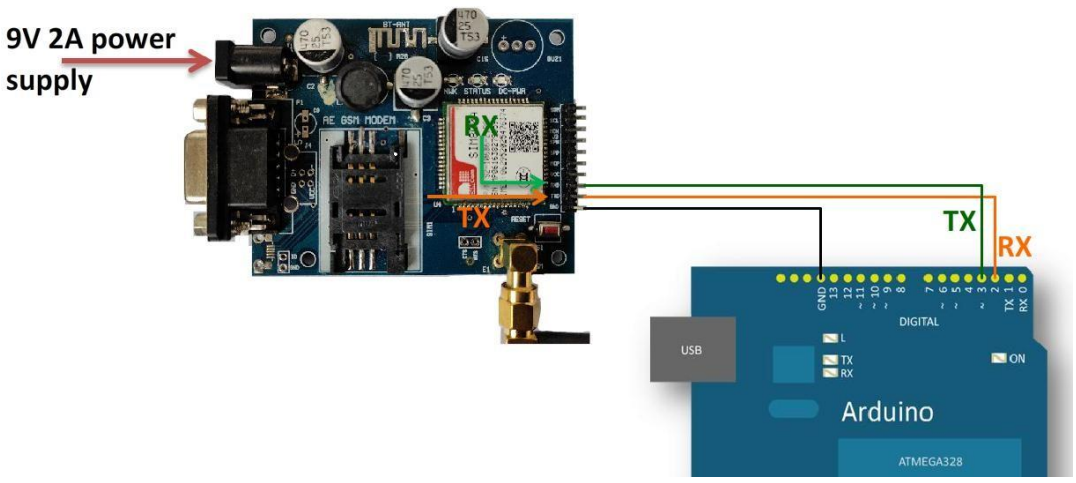
You can now type the message text and send the message using the <CTRL>-<Z> key combination:

Hello World ! <CTRL-Z>

After some seconds the modem will respond with the message ID of the message, indicating that the message was sent correctly: **+CMGS: 62**

Else upload the given program in the Arduino board and send sms .

Interfacing diagram:



Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

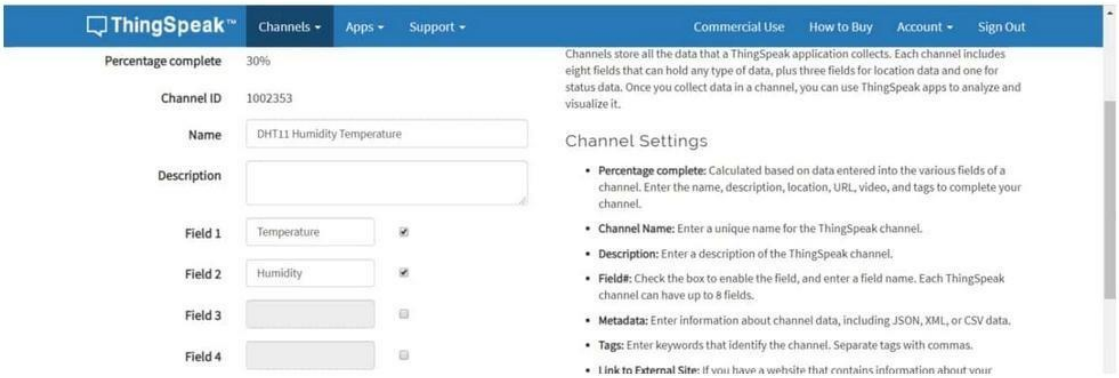
Program:

- Following programs are given with this manual, run them one by one and check the output
    a. Program to call mobile phone from GSM module
    b. Program to send SMS to the mobile phone from GSM module
    c. Program to receive SMS in the GSM module from mobile phone

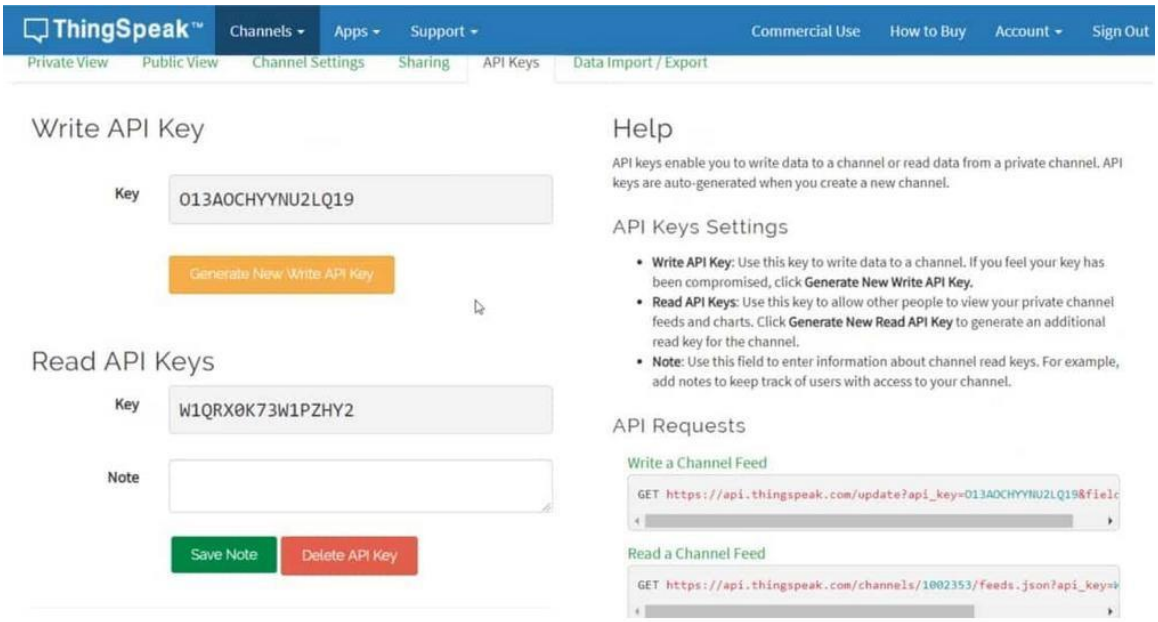
Remotely monitor the data using General Packet Radio Services (GPRS) facility of GSM board and ThingSpeak cloud

- ThingSpeak is an IOT cloud platform. Some of its facilities are free to use.
- By using ThingSpeak, we can monitor our data and control our system over the Internet, using the Channels and web pages provided by ThingSpeak.
- Visit https://thingspeak.com and create an account.

- Then create a new channel and name the fields for the data which you want to send to the cloud. For example, name the first field as Temperature and second field as Humidity.



-
- Now see the tab – API keys. These keys are to be written in the Arduino board program.



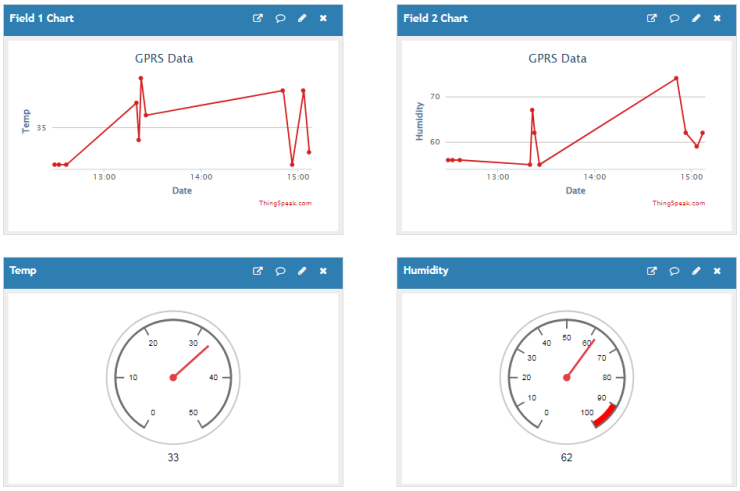
- The Write API is used when you want to send data to the ThingSpeak cloud from the Arduino board
- The Read API key is used when you want to read data from the ThingSpeak cloud in the Arduino board.
- Now, first see the program to send data to ThingSpeak cloud. In this program,
- Replace the “www” name by the APN (Access Point Name) of your SIM card.

```
gprsSerial.println("AT+CSTT=\"airtelgprs.com\"");//start task and setting the APN,
delay(1000);
```

- To check the APN of your SIM card, put your SIM card in your mobile. Then go to Settings > SIM cards & Mobile Network > Select the SIM > Access point names. In the GNERAL tab see what is written.
- **For VI, it is “www”**
- **For BSNL, it is “bsnlnet”**
- **For Jio, it is “jionet”**
- **For Airtel, it is “airtelgprs.com”**
- **For Idea, it is “internet”**
- Also Replace the Write API key by your Write API key, as here we are sending data to the ThingSpeak cloud.

```
String str="GET https://api.thingspeak.com/update?api_key=013AOCHYYNU2LQ19&field1=" + String(t) +"&field2="+String(h);
Serial.println(str);
```

- Now put the SIM card in the GSM module. Connect the power supply. Check the network LED.
- Now upload the program. See the output on the ThingSpeak web page.



- In this program, we are sending random value for Temperature and humidity. You can connect the DHT11 sensor and see the values.
- For the free version of ThingSpeak, there is a delay of 15 seconds. So the values are updated after 15 seconds.

Observation:

- Observe the output on the web page.