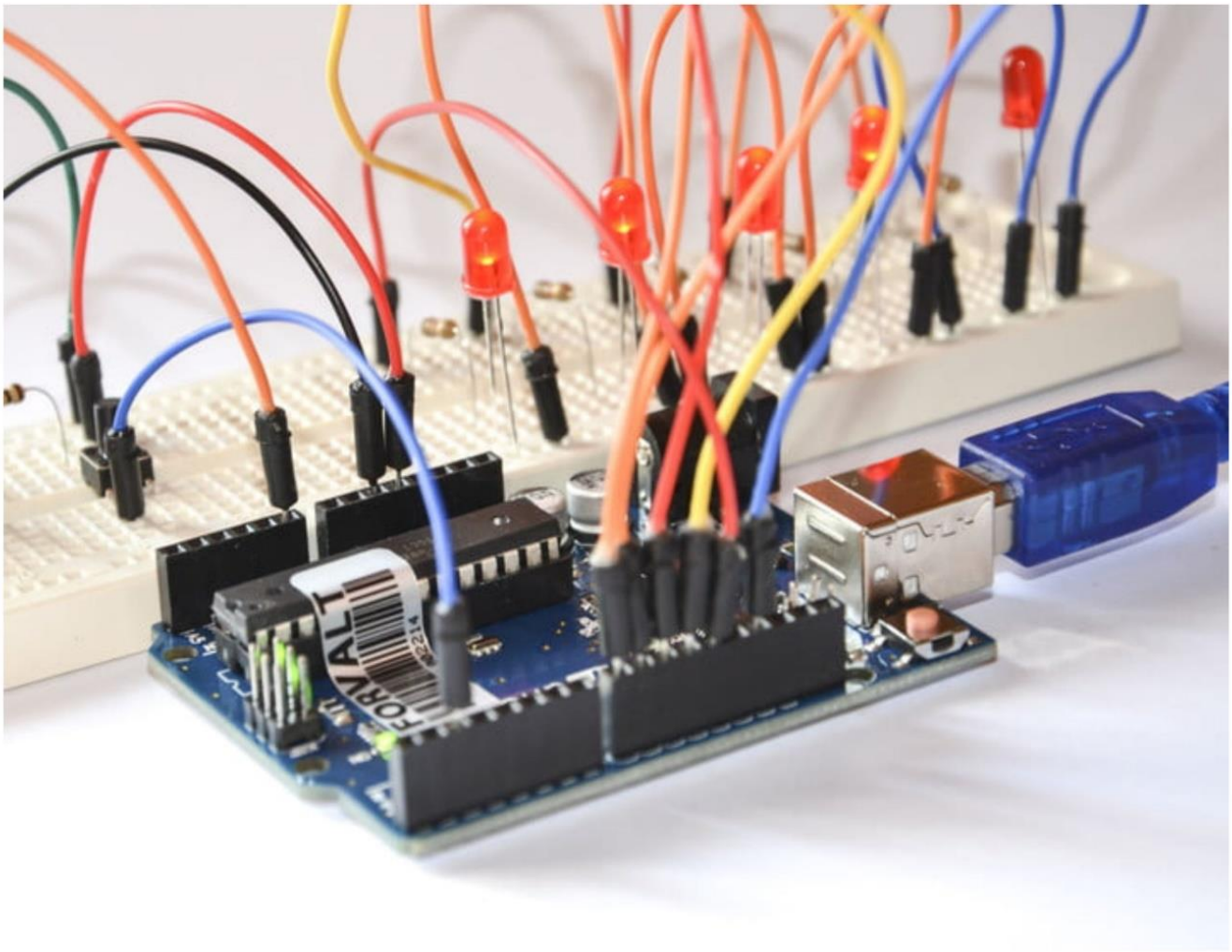
SUN ROBOTICS

Show Your Expertise With Arduino

Ultimate Kit Of Arduino



www.sunrobotics.co.in

SunRobotics Technologies
A/302, Swaminarayan Avenue, Vasna – A'bad – Gujarat – India
Ph no: 079 26608128 : support@sunrobotics.co.in

Preface

Sun Robotics is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

This is an Advanced Level for Arduino kit. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

Contents

- Lesson-17 Control Servo Motor
- Lesson-18 DC Motor Control
- Lesson-19 L293-DC Motor Control
- Lesson-20 Interfacing tilt sensor with arduino
- Lesson-21 Arduino Flame Sensor
- Lesson-22 IR_Receiver using arduino
- Lesson-23 LED Matrix display 8 x 8 dots MAX7219
- Lesson-24 Showing the temperature and humidity sensor in using arduino
- Lesson-25 Using a 74HC595 Shift Register with an Arduino
- Lesson-26 RC522 RFID Module
- Lesson-27 Control LED using clap system
- Lesson-28 Keypad Module
- Lesson-29 Analog Joystick Module
- Lesson-30 Water Level Detection Sensor Module
- Lesson-31 Real Time Clock Module
- Lesson-32 HC-SR04 Project
- Lesson-33 Interface Ultrasonic Sensor
- Lesson-34 PIR Sensor
- Lesson-35 PNP transistor
- Lesson-36 Spinning DC Motor and Control Speed
- Lesson-37 Arduino IR Obstacle Sensor

Web: www.sunrobotics.co.in
Email:support@sunrobotics.co.in

Getting Started with Arduino

What is an Arduino?

Arduino is an open-source physical computing platform designed to make experimenting with electronics more fun and intuitive. Arduino has its own unique, simplified programming language, a vast support network, and thousands of potential uses, making it the perfect platform for both beginner and advanced DIY enthusiasts.

www.arduino.cc

A Computer for the Physical World:

The friendly blue board in your hand (or on your desk) is the Arduino. In some ways you could think of Arduino as the child of traditional desktop and laptop computers. At its roots, the Arduino is essentially a small portable computer. It is capable of taking inputs (such as the push of a button or a reading from a light sensor) and interpreting that information to control various outputs (like a blinking LED light or an electric motor). That's where the term "physical computing" is born - an Arduino is capable of taking the world of electronics and relating it to the physical world in a real and tangible way. Trust us - this will all make more sense soon.

Arduino UNO SMD R3

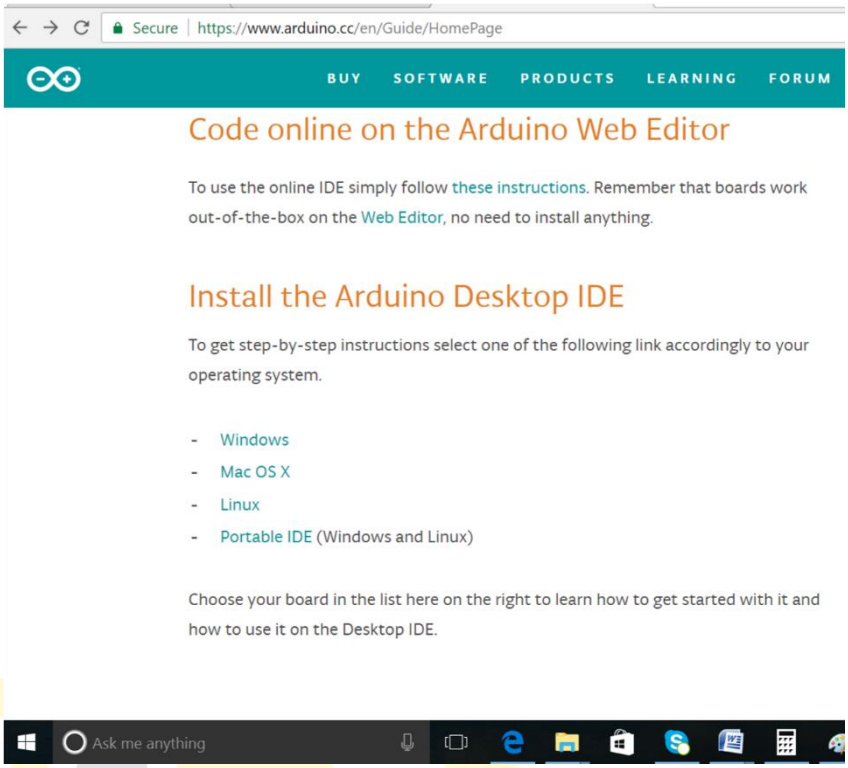
The Arduino Uno is one of several development boards based on the ATmega328. We like it mainly because of its extensive support network and its versatility. It has 14 digital input/output pins (6 of which can be PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Don't worry, you'll learn about all these later.

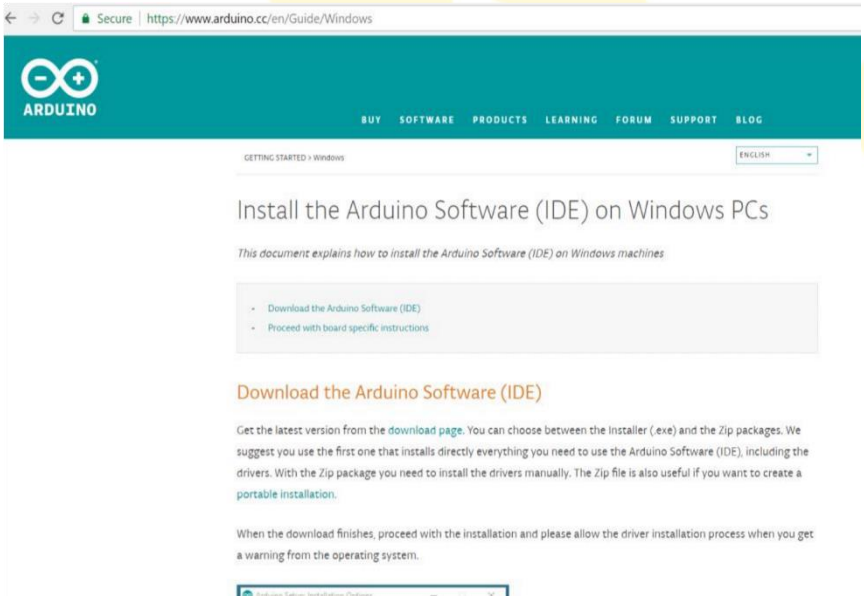
Installing Arduino IDE and Using Uno R3 board

STEP-1: Download the Arduino IDE (Integrated Development Environment)

Access the Internet: In order to get your Arduino up and running, you'll need to download some software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the Arduino to do exactly what you want. It's like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

www.arduino.cc/en/M,ain/Software


Secure | https://www.arduino.cc/en/Guide/HomePage
BUY SOFTWARE PRODUCTS LEARNING FORUM
Code online on the Arduino Web Editor
To use the online IDE simply follow these instructions. Remember that boards work out-of-the-box on the Web Editor, no need to install anything.
Install the Arduino Desktop IDE
To get step-by-step instructions select one of the following link accordingly to your operating system.
- Windows
- Mac OS X
- Linux
- Portable IDE (Windows and Linux)
Choose your board in the list here on the right to learn how to get started with it and how to use it on the Desktop IDE.
Ask me anything


Secure | https://www.arduino.cc/en/Guide/Windows
ARDUINO
BUY SOFTWARE PRODUCTS LEARNING FORUM SUPPORT BLOG
GETTING STARTED > Windows
ENGLISH
Install the Arduino Software (IDE) on Windows PCs
This document explains how to install the Arduino Software (IDE) on Windows machines
Download the Arduino Software (IDE)
Proceed with board specific instructions
Download the Arduino Software (IDE)
Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.

For different operating system platforms, the way of using Arduino IDE is different. Please refer to the following links: Windows User ： http://www.arduino.cc/en/Guide/Windows Mac OS
User:http://www.arduino.cc/en/Guide/MacOSXLinuxUserhttp://playground.arduino.cc/Learning/Linux For more detailed information about Arduino IDE, please refer to the following link: **http://www.arduino.cc/en/Guide/HomePage**

STEP-2: Connect your Arduino Uno to your Computer:

Use the USB cable provided in the kit to connect the Arduino to one of your computer's USB inputs.

STEP-3: Install Drivers

Depending on your computer's operating system, you will need to follow specific instructions. Please go to the URLs below for specific instructions on how to install the drivers onto your Arduino Uno.

Windows Installation Process: Go to the web address below to access the instructions for installations on a Windows-based computer.

http://arduino.cc/en/Guide/Windows

Macintosh OS X Installation Process: Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

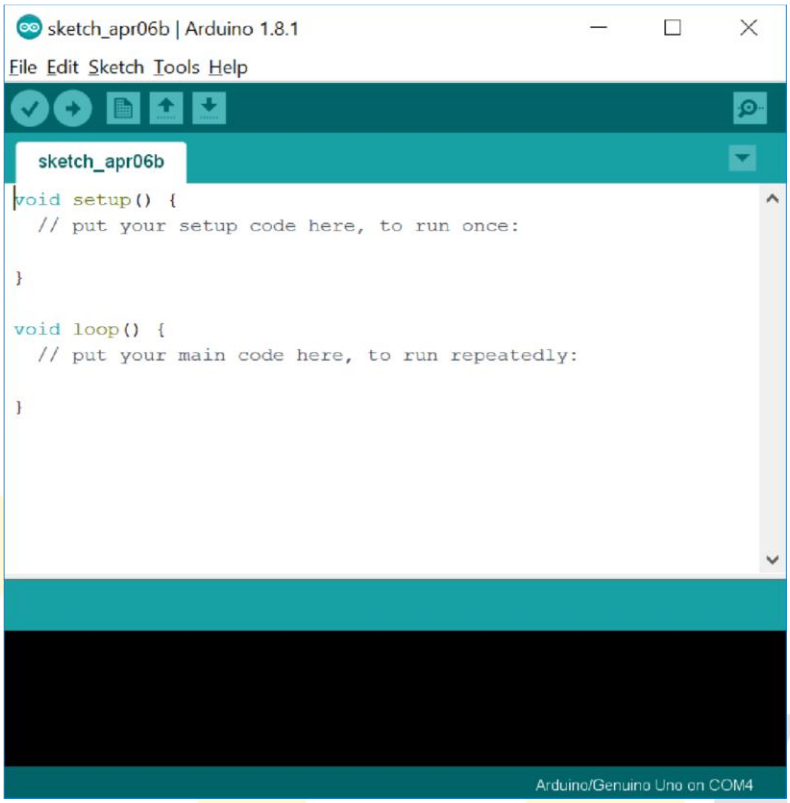
http://arduino.cc/en/Guide/MacOSX

Linux: 32 bit / 64 bit, Installation Process Go to the web address below to access the instructions for installations on a Linux-based computer.

http://www.arduino.cc/playground/Learning/Linux

STEP-4: Open the Arduino IDE

Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away, this is just an introduction. The step is to set your IDE to identify your Arduino Uno.



GUI (Graphical User Interface)

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for Details .Note: If you are using an external programmer with your board, you can hold Down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer" New Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within The current window overwriting its content. Note: due to a bug in Java, this menu Doesn't scroll; if you need to open a sketch late in the list, use the File | Sketch book Menu instead.

Save

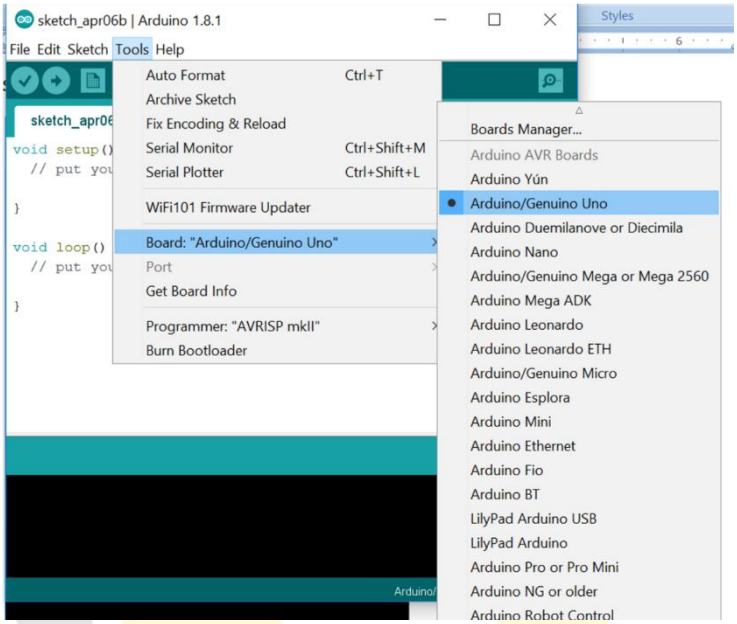
Saves your sketch.

Serial Monitor

Opens the serial monitor.

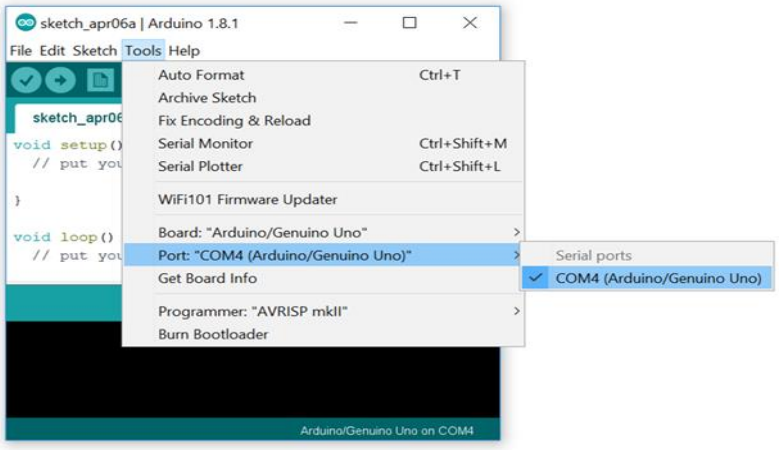
Web: www.sunrobotics.co.in
Email:support@sunrobotics.co.in

STEP-5: Select your board: Arduino Uno



STEP-6: Select your Serial Device

Windows: Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be com3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



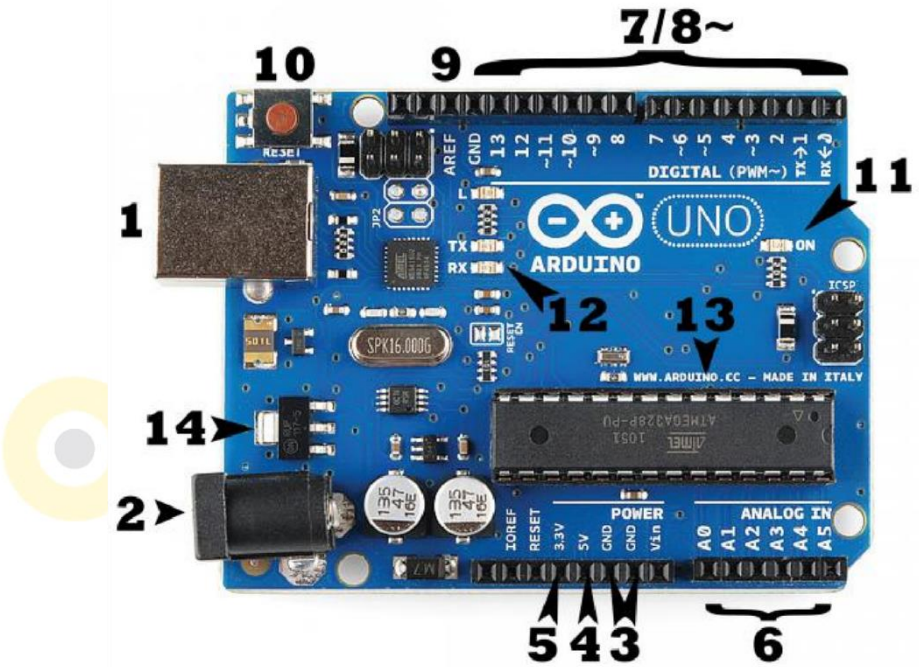
Mac OS: Select the serial device of the Arduino board from the Tools > Serial Port menu. On the Mac, this should be something with /dev/tty.usbmodem (for the Uno or Mega 2560) or /dev/tty.usbserial (for older boards) in it.

Linux: http://playground.arduino.cc/Learning/Linux

About Arduino Uno R3 board

What's on the board?

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common:



Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2).** .

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts. Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3):** Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the AT mega line of IC's from the ATMEL company. This can be

important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

Lesson 1 - Blinking LED

Overview:

In this practical, we will start the journey of learning Arduino UNO .To begin, let's learn how to make an LED blink.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x 220Ω Resistor
- 1 x LED
- 1 x BreadBoard
- 2 x Jumper Wires

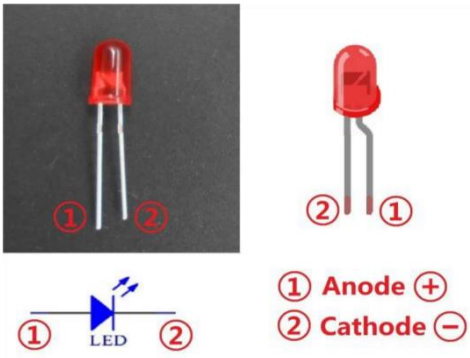
Principle:

In this lesson, we will program the Arduino's GPIO output high level (+5V) and low level (0V), and then make the LED which is connected to the Arduino's GPIO flicker with a certain frequency.

1. What is the LED?

The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes: a positive electrode and a negative one. It lights up only when a forward current passes, and it can flash red, blue, green, yellow, etc. The color of the light depends on the material it is made.

In general, the drive current for LED is 5-20mA. Therefore, in reality it usually needs an extra resistor for current limitation so as to protect the LED.

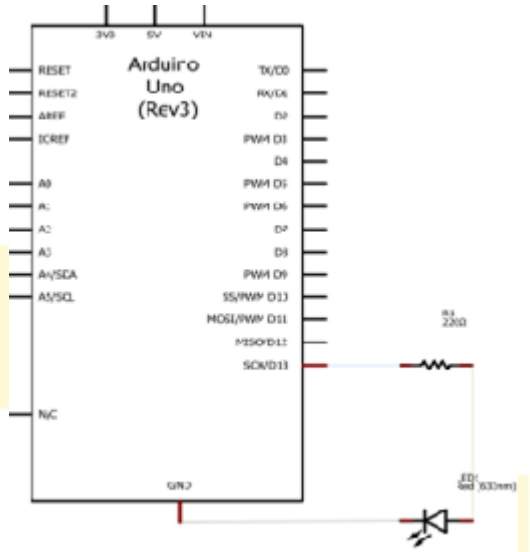


2. What is resistor?

The main function of the resistor is to limit currents. In the circuit, the character 'R' represents resistor, and the unit of resistor is ohm (Ω).

A band resistor is used in this experiment. It is a resistor with a surface coated with some particular color through which the resistance can be identified directly.

3. **Schematic :**



4. **Key functions:**

●setup ()

The setup () function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the

Arduino board.

●loop ()

After creating a setup () function, which initializes and sets the initial values, the loop () function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

●pinMode()

Configures the specified pin to behave either as an input or an output.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

●digitalWrite()

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode (), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

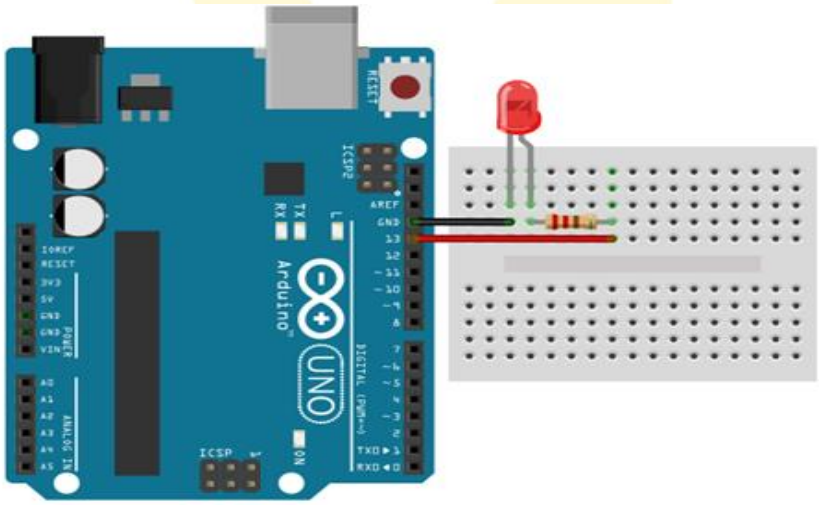
If the pin is configured as an INPUT, digitalWrite () will enable (HIGH) or disable (LOW) the internal pull up on the input pin. It is recommended to set the pin Mode () to INPUT_PULLUP to enable the internal pull-up resistor

●delay ()

Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Procedure:

Step 1: Build the circuit as below:



Step 2: Program: You can copy paste the below program in the IDE or open the code directly from the “CODE” folder that comes with the DVD/from the downloaded Zip folder.

```
/*************************************************************
File name:   01_blinkingLed.ino Description:
Let, LED blinks. ************************************************************/

int ledPin = 13;// LED connected to digital pin 13
void setup()   {
 pinMode(ledPin,OUTPUT);
 }

void loop()    {
digitalWrite(ledPin,HIGH);  // set the LED on
 delay(1000);  // wait for a second
 digitalWrite(ledPin,LOW);// set the LED off
 delay(1000); // wait for a second
 }
```

3: Compile the program and upload to Arduino UNO board. Now you can see the LED blinking.

Lesson 2 - Four Blinking LED

Overview:

In the first lesson, we have learned how to make an LED blink by programming the Arduino. Today, we will use the Arduino to control 4 LEDs to make the LEDs show the effect of flowing.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 4 x 220Ω Resistor
- 4 x LED
- 1 x BreadBoard
- Several Jumper Wires

Principle:

The principle of this experiment is very simple and is quite similar with that in the first lesson.

1 **. Key function**

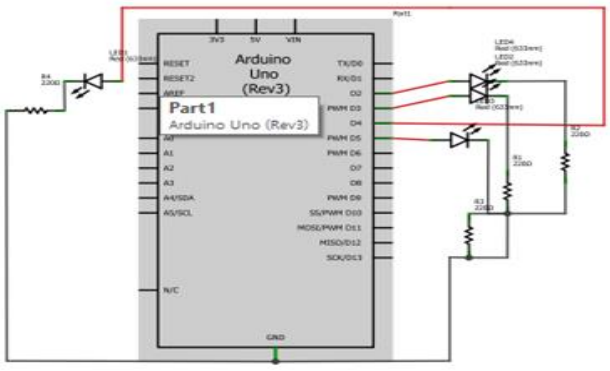
The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

There are three parts to the for loop header:

For (initialization; condition; increment) { //statement(s);
}

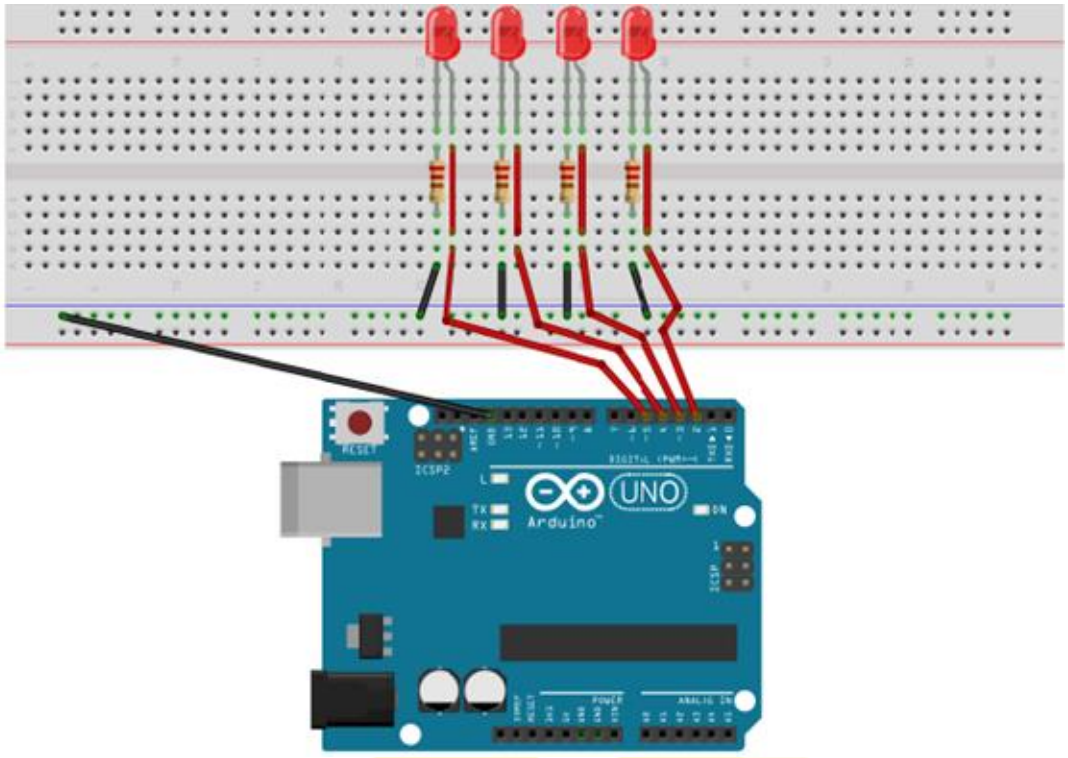
The initialization happens first and exactly once. Each time through the loop, the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

2. Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program Code

```
/*************************************************************
File name:   02_Four blinkingLed.ino Description:
Let, four LED blinks. ************************************************************/

int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
void setup() {
    pinMode(led1,OUTPUT);
    pinMode(led2,OUTPUT);
    pinMode(led3,OUTPUT);
    pinMode(led4,OUTPUT);
 }
  void loop() {
  digitalWrite(led1,HIGH);
  delay(50);
  digitalWrite(led1,LOW);
  digitalWrite(led2,HIGH);
  delay(50);
  digitalWrite(led2,LOW);
  digitalWrite(led3,HIGH);
  delay(50);
  digitalWrite(led3,LOW);
```

```
  digitalWrite(led4,HIGH);
  delay(50);
  digitalWrite(led4,LOW);
  delay(200);
}
```

Step 3: Compile the program and upload to Arduino UNO board
Now, you can see 8 LEDs light up in sequence from the green one on the right side to others on the left, and next from the left to the right. The LEDs flash like flowing water repeatedly in a circular way.

Lesson 3 – RGB Blinking LED

Overview:

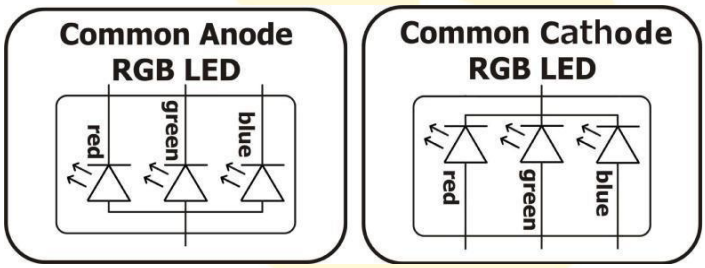
In this lesson, we will program the Arduino for RGB LED control, and make RGB LED emits several of colors of light.

Components:

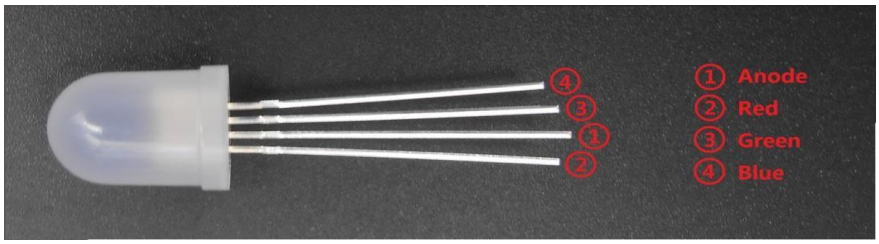
- 1 X Arduino UNO
- 1 X USB Cable
- 1 X RGB LED
- 3 X 220Ω Resistor
- 1 X Breadboard
- Several jumper wires

Principle:

RGB LEDs consist of three LEDs: red, green and blue. These three colored LEDs are capable of producing any color. Tri-color LEDs with red, green, and blue emitters, in general using a four-wire connection with one common lead (anode or cathode).

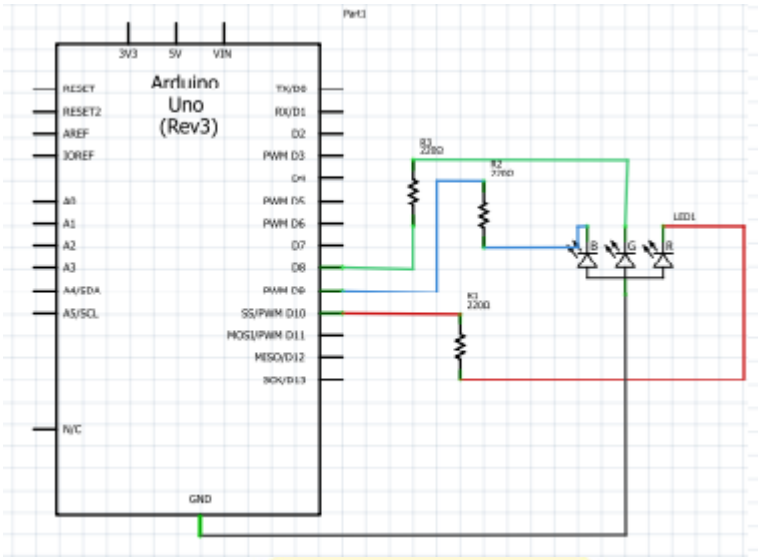


What we use in this experiment is a common anode RGB LED. The longest pin is the common anode of the three LEDs. The pin is connected to the +5V pin of the Arduino, and the rest pins are connected to pin D8, D9, and D10 of the Arduino with a current limiting resistor between.



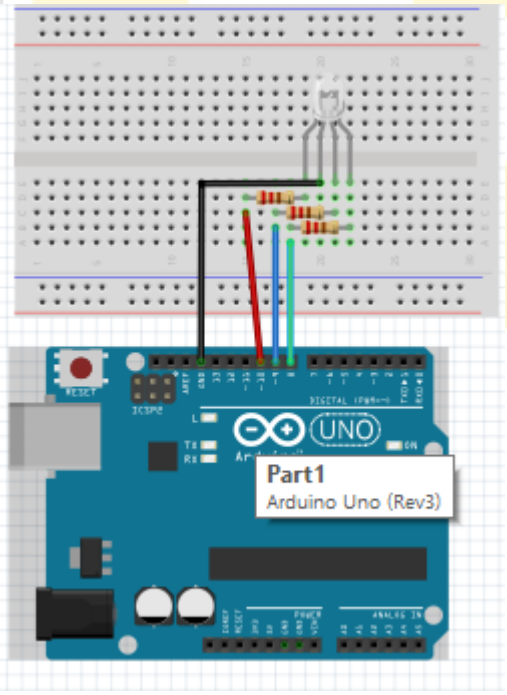
In this way, we can control the color of an RGB LED by 3-channel PWM signals.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program code

```
/*************************************************************
File name:   03_RGB blinkingLed.ino Description:
Let, RGB LED blinks. ************************************************************/

void setup() {
pinMode (8,OUTPUT);
pinMode (9,OUTPUT);
```

```
pinMode (10,OUTPUT);
}

Void loop() {
digitalWrite (8,HIGH);
digitalWrite (10,LOW);
delay(1000);
digitalWrite (9,HIGH);
digitalWrite (8,LOW);
delay(1000);
digitalWrite (10,HIGH);
digitalWrite (9,LOW);
delay(1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Now, you can see the RGB LED flash red, green, blue, yellow, white and purple light, and then go out. Each state lasts for 1s each time, and the LED flashes colors repeatedly in such sequence.

Lesson 4 -Push button – LED Blinking

Overview

In this lesson, we will learn how to detect the state of a button, and then toggle the state of the LED based on the state of the button.

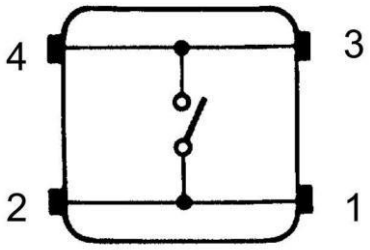
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x 220Ω Resistor
- 1 x 10KΩ Resistor
- 1 x LED
- 1 x Push Button
- 1 x Breadboard
- Several jumper wires

Principle

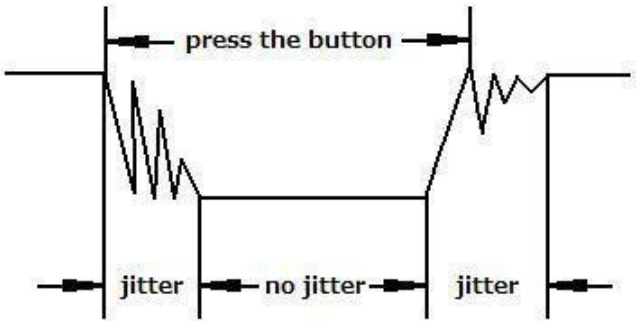
1. **Button**

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used in this experiment will be a 12mm button as shown below.



The button we used is a normally open type one. The two contacts of a button are in the off state under the normal conditions; only when the button is pressed they are closed.

The button jitter must happen in the process of using. The jitter waveform is as the flowing picture:



Each time you press the button, the Arduino will regard you have pressed the button many times due to the jitter of the button. You should deal with the jitter of buttons before using. You can eliminate the jitter through software programming. Besides, you can use a capacitor to solve the issue. Take the software method for example. First, detect whether the level of button interface is low level or high level. If it is low level, 5~10ms delay is needed. Then detect whether the level of button interface is low or high. If the signal is low, you can infer that the button is pressed once. You can also use a 0.1uF capacitor to avoid the jitter of buttons.

2. Interrupt

Hardware interrupts were introduced as a way to reduce wasting the processor's valuable time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem.

. Key functions:

●attach Interrupt (interrupt, ISR, mode)

Specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3).

Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be ignored (turned off) until the current one is finished. As delay () and Millis () both rely on interrupts, they will not work while an ISR is running. delayMicroseconds (), which does not rely on interrupts, will work as expected.

Syntax:

attachInterrupt (pin, ISR,mode)

Parameters: pin: the pin number

ISR: the ISR will be called when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

-LOW to trigger the interrupt whenever the pin is low,

-CHANGE to trigger the interrupt whenever the pin changes value -RISING to trigger When the pin goes from low to high, -FALLING for when the pin goes from high to low.

●digitalRead ()

Reads the value from a specified digital pin, either HIGH or LOW. Syntax:

digitalRead (pin)

Parameters:

Pin: the number of the digital pin you want to read (int) Returns:

HIGH or LOW ●delayMicroseconds (us)

Pauses the program for the amount of time (in microseconds) specified as parameter. There are a thousand microseconds in a millisecond, and a million microseconds in a second.

Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay () instead.

Syntax:

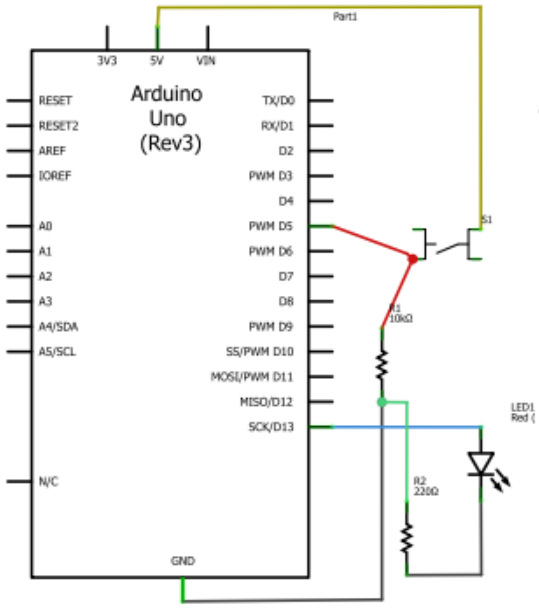
delayMicroseconds(us)

Parameters:

us: the number of microseconds to pause (unsigned int) Returns:

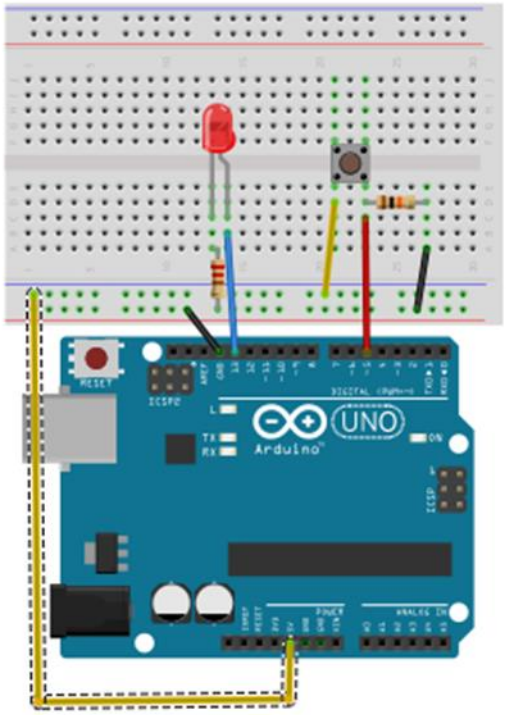
None

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   04_  Push button – LED Blinking.ino Description:
Let, Using push button LED blinks. ************************************************************/

int ledpin=13;          // definition digital interface
```

```
int inpin=5;          //Define the number 7 Interface
int val;              //Define the variable val
void setup()   {
  pinMode(ledpin,OUTPUT);//Define led as Output
  pinMode(inpin,INPUT);//Button interface is defined as input
}
void loop()    {
val=digitalRead(inpin);//Read digitalpin 7 level value assigned to val
if(val==HIGH)//Test button is pressed
{
  digitalWrite(ledpin,LOW);
}
else {
  digitalWrite(ledpin,HIGH);
}
}
```

Step 3: Compile the program and upload to Arduino UNO board

Now press the button, and you can see the state of the LED will be toggled between ON and OFF.

Lesson 5 – 7 Segment Display Interface

Overview:

In this lesson, we will program the Arduino to achieve the controlling of a segment display.

Components

- 1 x Arduino UNO
- 1 x USB Cable
- 8 x 220Ω Resistor
- 1 x 7-segment Display
- 1 x Breadboard
- Several jumper wires

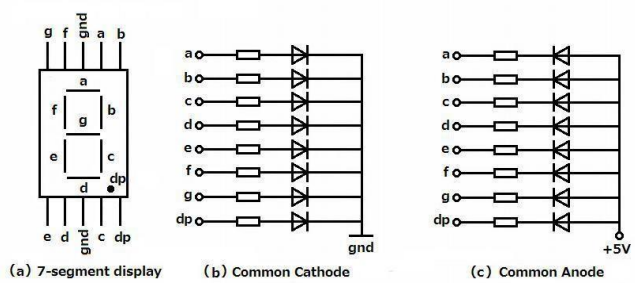
Principle

The seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The seven-segment display is an 8-shaped LED display device composed of eight LEDs (including a decimal point). The segments respectively named a, b, c, d, e, f, g, and dp.

The segment display can be divided into two types: common anode and common cathode segment displays, by internal connections.

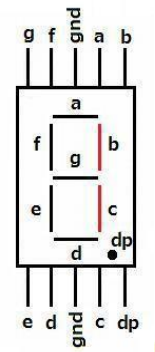


(a) 7-segment display (b) Common Cathode (c) Common Anode

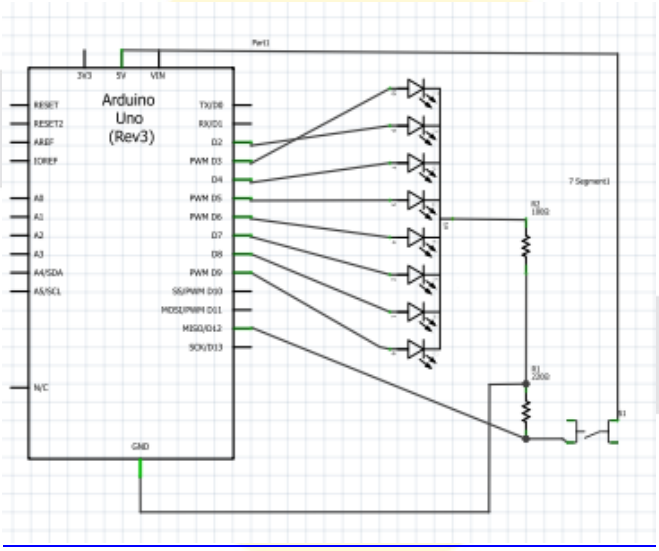
For a common-anode LED, the common anode should be connected to the power supply (VCC); for a common-cathode LED, the common cathode should be connected to the ground (GND).

Each segment of a segment display is composed of an LED, so a resistor is needed for protecting the LED.

A 7-segment display has seven segments for displaying a figure a done more for displaying a decimal point. For example, if you want to display a number '1', you should only light the segment b and c, as shown below.

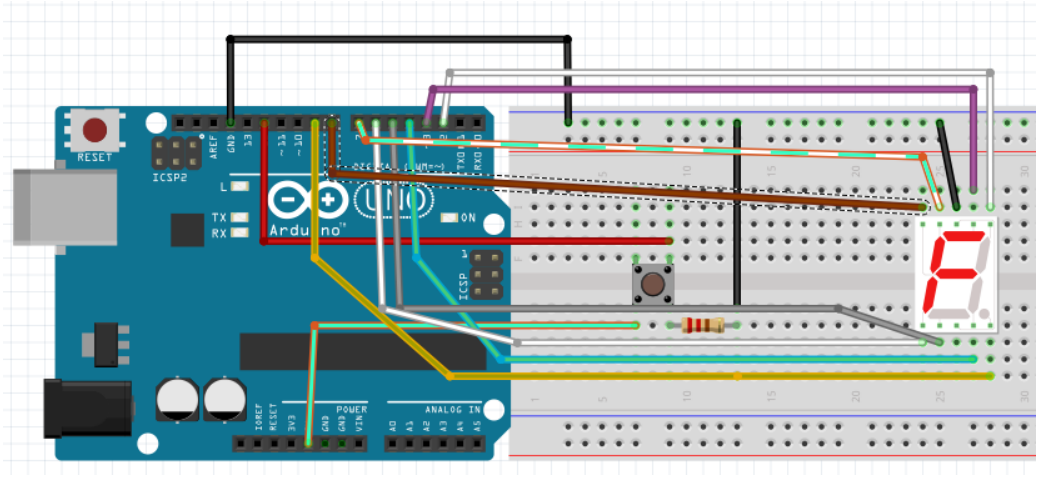


Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program Code:

```
/*************************************************************
File name:   05_7 Segment Display Interface .ino Description:
Let, 7 segment interface. ************************************************************/

int pushButton = 12;
unsigned int buttonState = 0;
int seven_seg_digits[10][7] = { { 1,1,1,1,1,1,0 },  // = 0
                                { 1,0,1,0,0,0,0 },  // = 1
                                { 1,1,0,1,1,0,1 },  // = 2
                                { 1,1,1,1,0,0,1 },  // = 3
                                { 1,0,1,0,0,1,1 },  // = 4
                                { 0,1,1,1,0,1,1 },  // = 5
                                { 0,1,1,1,1,1,1 },  // = 6
                                { 1,1,1,0,0,0,0 },  // = 7
                                { 1,1,1,1,1,1,1 },  // = 8
                                { 1,1,1,0,0,1,1 }   // = 9
                                };
void setup()      {
 Serial.begin(9600);
 pinMode(pushButton, INPUT);
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
}
void loop() {
  if (digitalRead(pushButton) == 1)  {
   buttonState = buttonState + 1;
   while (digitalRead(pushButton) == 1)   {
   delay (5);
   }
 }
 Serial.println(buttonState);
 delay(1);       // delay in between reads for stability
 sevenSegWrite(buttonState);
 if (buttonState == 9)     {
  hringur();
  buttonState = 0;
 }
}
 void hringur()   {
 while (buttonState == 9)   {
  Serial.println(buttonState);
  delay(1);       // delay in between reads for stability
  if (digitalRead(pushButton) == 1)   {
  buttonState = buttonState + 1;
  while (digitalRead(pushButton) == 1)   {
  delay (5);
  }
```

```
  }
 }
 return;
}
void sevenSegWrite(unsigned int digit) {
 unsigned int pin = 2;
 for (unsigned int segCount = 0; segCount < 8; segCount++) {
  digitalWrite(pin, seven_seg_digits[digit][segCount]);
  pin++;
  digitalWrite(9, 1);
 }
 return;
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Now, you should see the number 0~9 and characters A~F displayed in turn on the segment display.

Web: www.sunrobotics.co.in
Email:support@sunrobotics.co.in

Lesson 6 - Four Digital Seven Segment Display

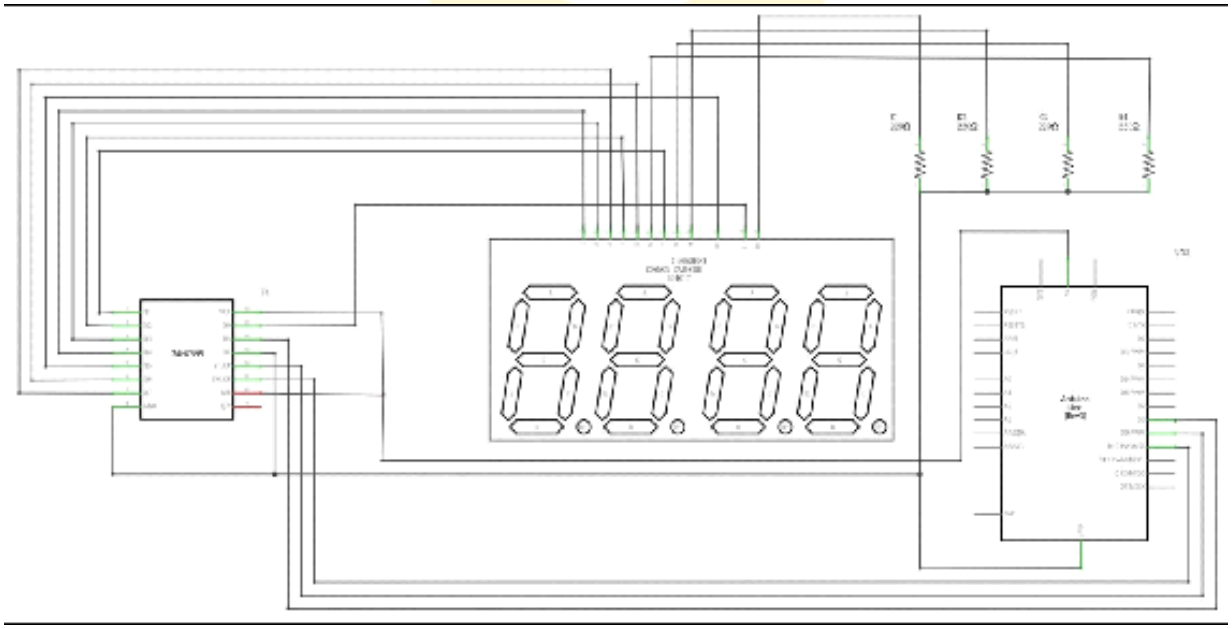
Overview:
In this lesson, you will learn how to use a 4-digit 7-segment display.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x 74HC595
- 1 x Push Button
- 1 x 4 Digit 7-segment Display
- 4 x 220Ω Resistor
- 1 x Breadboard
- Several jumper wires

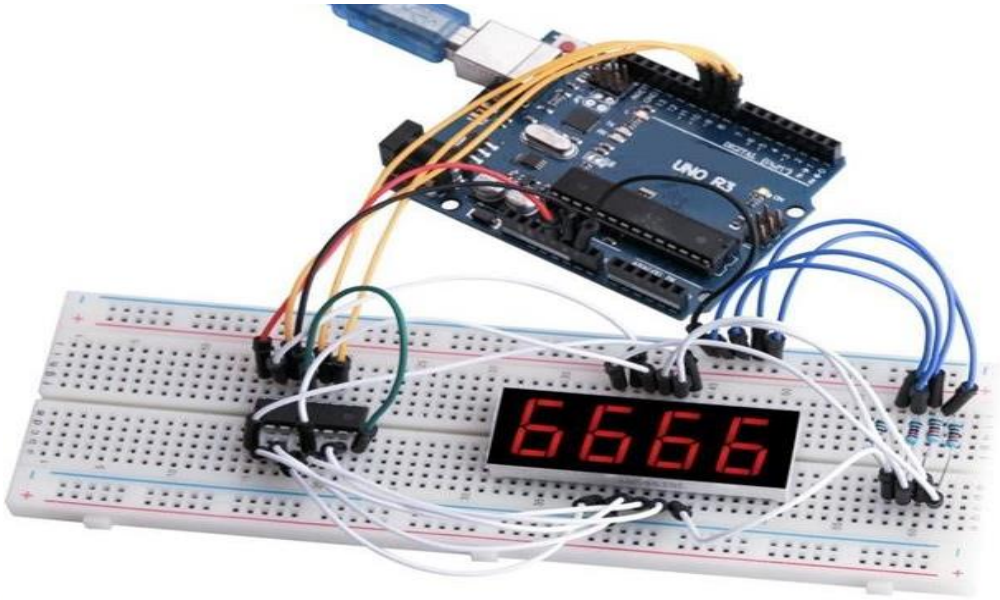
Principle
When using 1-digit 7-segment display and it is common anode, the common anode pin connects to the power source; if it is common cathode, the common cathode pin connects to the GND.
When using 4-digit 7-segment display, the common anode or common cathode pin is used to control which digit is displayed. Even though there is only one digit working, the principle of Persistence of Vision enables you to see all numbers displayed because each the scanning speed is so fast you hardly notice the intervals.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program code

```
/*************************************************************
File name:   06_  Four Digital Seven Segment Display .ino Description:
Let, four digital segment display. ************************************/

int latch=9;  //74HC595  pin 12 STCP
int clock=10; //74HC595  pin 11 SHCP
int data=8;   //74HC595  pin 14 DS
unsigned char table[]=
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c
,0x39,0x5e,0x79,0x71,0x00};
void setup() {
  pinMode(latch,OUTPUT);
  pinMode(clock,OUTPUT);
  pinMode(data,OUTPUT);
}
void Display(unsigned char num)
{
  digitalWrite(latch,LOW);
  shiftOut(data,clock,MSBFIRST,table[num]);
  digitalWrite(latch,HIGH);
}
void loop() {
  Display(1);
  delay(500);
  Display(2);
  delay(500);
```

```
  Display(3);
  delay(500);
  Display(4);
  delay(500);
  Display(5);
  delay(500);
  Display(6);
  delay(500);
  Display(7);
  delay(500);
  Display(8);
  delay(500);
  Display(9);
  delay(500);
  Display(10);
  delay(500);
  Display(11);
  delay(500);
  Display(12);
  delay(500);
  Display(13);
  delay(500);
  Display(14);
  delay(500);
  Display(15);
  delay(500);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 7 – 3 Way Traffic Light Controller

Overview:

In the second lesson, we have learned how to make a LEDs blink by programming the Arduino. Today, we will use the Arduino to control different LEDs different color to make the LEDs show the effect of flowing in traffic light system.

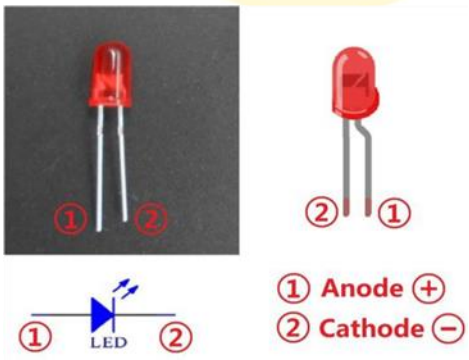
Components:

- 1 X Arduino UNO
- 1 X USB Cable
- 3 x 200Ω Resistor
- 3 X 5mm RED LED
- 3 x 5mm YELLOW LED
- 3 x 5mm GREEN LED
- 1 X Breadboard
- Several jumper wires

Principle:

The principle of this experiment is very simple and is quite similar with that in the second lesson.

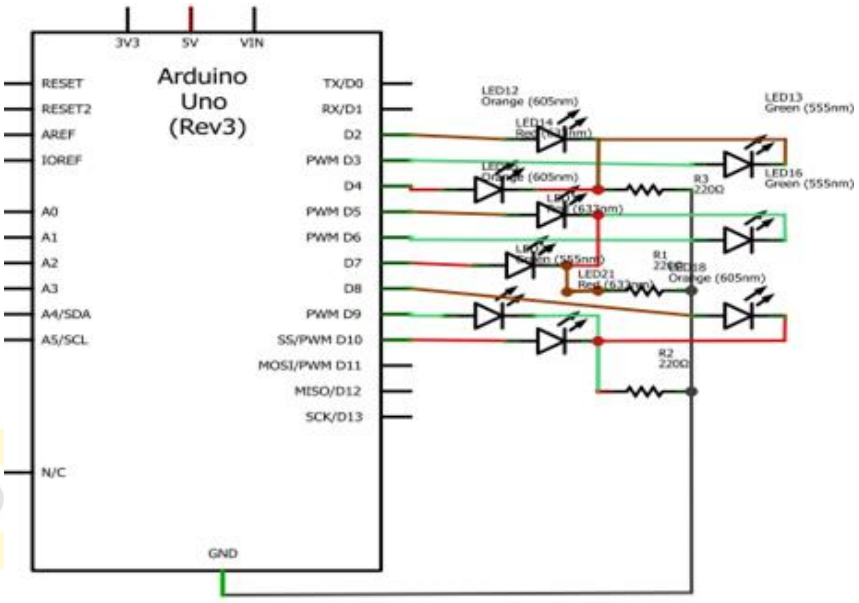
This **Arduino based 3-Way Traffic Light Controller** is a simple project which is useful to understand the working of traffic lights which we see around us. We have covered a simpler version of traffic lights in this traffic light circuit. Here have demonstrated it for 3 sides or ways. Now let's get into the project.



The code for this Arduino Traffic Light Controller Project is simple and can be easily understood. Here we have demonstrated Traffic lights for the 3 ways road and the code glows LED's on all the three sides in a particular sequence, in which the actual Traffic Lights works. Like, at a time, there will be two Red signals on any of the two sides and one Green light on

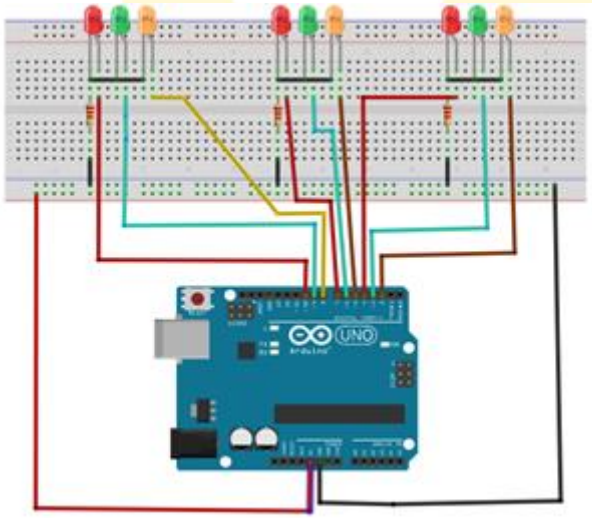
the remaining side. And yellow light will also glow, for 1 second each time, in between transition from Red to Green, means first red light glows for 5 second then yellow light glows for 1 second and then finally green light will be turned on.

1 . Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program code

```
/*************************************************************
File name:  07_3 WAY TRAFFIC LIGHT .ino Description:
Let, traffic light using LED blinks. ************************************************************/
```

```
void setup() {
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
}
 void loop()   {
  digitalWrite(2,1); //enables the 1st set of signals
  digitalWrite(7,1);
  digitalWrite(10,1);
  digitalWrite(4,0);
  digitalWrite(3,0);
  digitalWrite(6,0);
  digitalWrite(8,0);
  digitalWrite(9,0);
  digitalWrite(5,0);
  delay(4000);

  digitalWrite(3,1); //enables the yellow lights
  digitalWrite(6,1);
  digitalWrite(2,0);
  digitalWrite(7,0);
  delay(1000);

  digitalWrite(4,1); //enables the 2nd set of signals
  digitalWrite(5,1);
  digitalWrite(10,1);
  digitalWrite(2,0);
  digitalWrite(3,0);
  digitalWrite(6,0);
  digitalWrite(8,0);
  digitalWrite(9,0);
  digitalWrite(7,0);
  delay(4000);

  digitalWrite(9,1); //enables the yellow lights
  digitalWrite(6,1);
  digitalWrite(10,0);
  digitalWrite(5,0);
  digitalWrite(4,0);
  delay(1000);

  digitalWrite(8,1); //enables the 3rd set of signals
```

```
  digitalWrite(4,1);
  digitalWrite(7,1);
  digitalWrite(2,0);
  digitalWrite(3,0);
  digitalWrite(5,0);
  digitalWrite(6,0);
  digitalWrite(9,0);
  digitalWrite(10,0);
  delay(4000);

  digitalWrite(9,1); //enables the yellow lights
  digitalWrite(3,1);
  digitalWrite(7,0);
  digitalWrite(8,0);
  digitalWrite(4,0);
  delay(1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Now, you can see the 3 way traffic light for the 3 ways road and the code glows LED's on all the three sides in a particular sequence, in which the actual Traffic Lights works. Like, at a time, there will be two Red signals on any of the two sides and one Green light on the remaining side. And yellow light will also glow, for 1 second each time, in between transition from Red to Green, means first red light glows for 5 second then yellow light glows for 1 second and then finally green light will be turned on.

Lesson 8 - Control traffic lights with a push button

Overview:

This project will imitate cross pedestrian traffic light control with push button switch. We will use three LEDs(RED,YELLOW,GREEN) to imitate traffic lights in main road, use two LEDs(GREEN,RED) to imitate pedestrian cross light, use a push button to imitate pedestrian cross button.

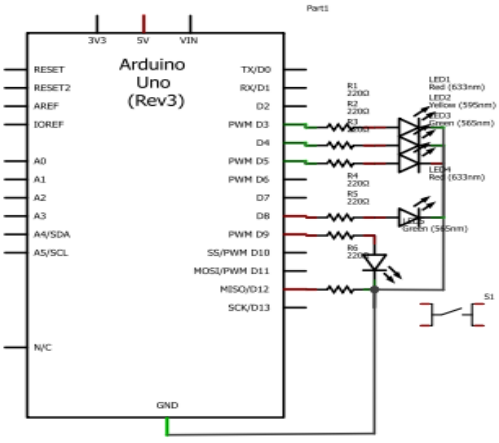
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 6 x 220Ω Resistor
- 2 x RED LED
- 1 x YELLOW LED
- 2 x GREEN LED
- 1 x Breadboard
- 1 x Push Button
- Several jumper wires

Principle

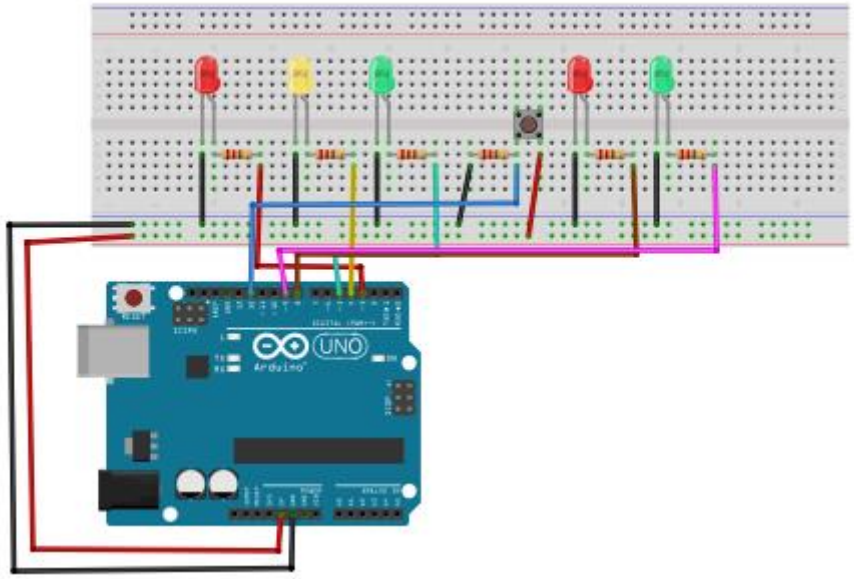
This project will imitate cross pedestrian traffic light control with push button switch. We will use three LEDs(RED,YELLOW,GREEN) to imitate traffic lights in main road, use two LEDs(GREEN,RED) to imitate pedestrian cross light, use a push button to imitate pedestrian cross button. When push button is pressed, the GREEN light in main road will turn off and yellow light will turn on for 2 seconds, then RED light will turn on, GREEN pedestrian light will turn on. After 4 seconds, the Green pedestrian light will become flashing for 5 seconds and turn off. Red pedestrian will on and main road Red light will off and Main road green light will on.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   08_Control traffic lights with a push button.ino Description:
Let, traffic control using push button to LED blinks***********************************/
int mainRoadRed = 3;
int mainRoadYellow = 4;
int mainRoadGreen = 5;
int crossRed = 8;
int crossGreen = 9;
int button = 12;
int crossTime = 4000;
unsigned long buttonGap ; // collects the time since the button was last pressed

void setup () {
  pinMode(mainRoadRed, OUTPUT);
  pinMode(mainRoadYellow, OUTPUT);
  pinMode(mainRoadGreen, OUTPUT);
  pinMode(crossRed, OUTPUT);
  pinMode(crossGreen, OUTPUT);
  pinMode(button, INPUT);
  digitalWrite(mainRoadGreen,HIGH); //start with green car light on
  digitalWrite(crossRed, HIGH); //start with red ped light on
}

void loop(){
  int state = digitalRead(button);
  if(state==HIGH && (millis() - buttonGap) > 5000) {
  switchLights();
```

```
  }
}
void switchLights() {
  digitalWrite(mainRoadGreen,LOW);  //turn off green light in main road
  digitalWrite(mainRoadYellow,HIGH); // turn on yellow light in main road
  delay(2000); //wait 2 seconds
  digitalWrite(mainRoadYellow,LOW); // turn off yellow light in main road
  digitalWrite(mainRoadRed,HIGH); //turn on red light in main road
  delay(1000); //wait 1s before turning on cross pedestrian green light
  digitalWrite(crossRed,LOW); //turn off red  pedestrian light
  digitalWrite(crossGreen,HIGH); //turn on green  pedestrian light
  delay(crossTime); //delay preset time of 4 seconds
  for (int x=0; x<10; x++){
  digitalWrite(crossGreen,HIGH);
  delay(250);
  digitalWrite(crossGreen,LOW);
  delay(250);
  }
 digitalWrite(crossRed, HIGH); //turn on red  pedestrian light
 delay(100);
 digitalWrite(mainRoadGreen,HIGH); //turn on green light in main road
 digitalWrite(mainRoadRed,LOW); //turn off red light in main road
 buttonGap = millis(); //remember time since recent light switch
 }
```

Step 3: Compile the program and upload to Arduino UNO board
Push the button, the main road red traffic light will turn on and pedestrian green light will turn on.

Lesson 9 - RGB LED push button color change

Overview:

In this lesson, we will program the Arduino to change RGB LED color with the press of a button.

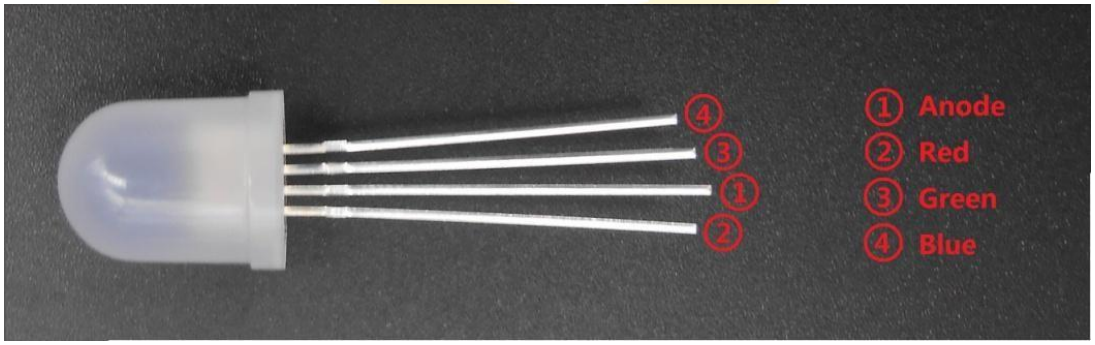
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x RGB LED
- 3 x 220Ω Resistor
- 1 x BreadBoard
- Several jumper wires

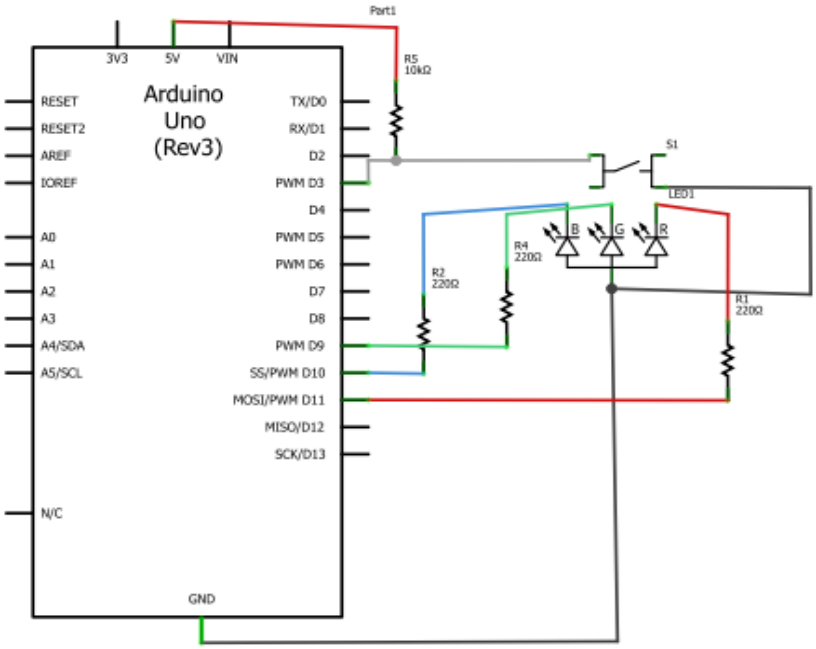
Principle

RGB LEDs consist of three LEDs: red, green and blue. These three colored LEDs are capable of producing any color. Tri-color LEDs with red, green, and blue emitters, in general using a four-wire connection with one common lead (anode or cathode).

What we use in this experiment is a common anode RGB LED. The longest pin is the common anode of the three LEDs. The pin is connected to the +5V pin of the Arduino, and the rest pins are connected to pin D9, D10, and D11 of the Arduino with a current limiting resistor between using then push button.

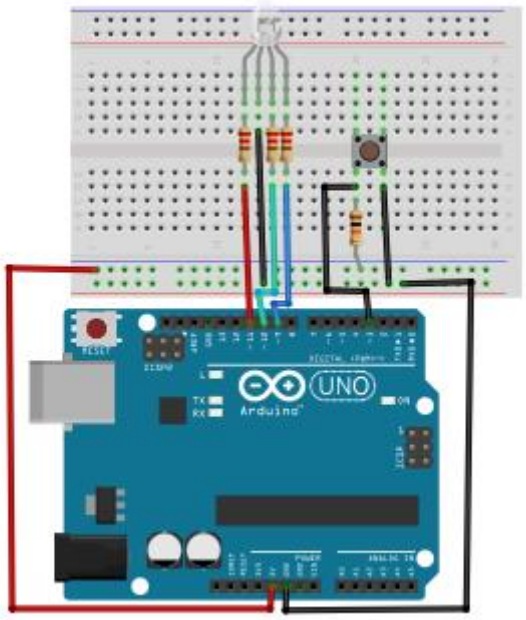


Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
```

```
File name:  09 RGB LED push button color change.ino Description:
Let,RGB LED blinks using push button*****************************************/
const int buttonPin = 3;
const int redPin = 11;
const int greenPin = 10;
const int bluePin = 9;
int counter = 0;

void setup()  {
 pinMode(buttonPin, INPUT);
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);
}
void loop() {
 int buttonState;
 buttonState = digitalRead(buttonPin);
 if (buttonState == LOW) {
  counter++;
  delay(150);
 }
  else if (counter == 0) {
  digitalWrite(red, LOW);
  digitalWrite(green, LOW);
  digitalWrite(blue, LOW);
 }
 else if (counter == 1) {
 digitalWrite(red, HIGH);
 digitalWrite(green, LOW);
 digitalWrite(blue, LOW);
 }
  else if (counter == 2) {
  digitalWrite(red, LOW);
  digitalWrite(green, HIGH);
  digitalWrite(blue, LOW);
 }
  else if (counter == 3) {
  digitalWrite(red, LOW);
  digitalWrite(green, LOW);
  digitalWrite(blue, HIGH);
 }
  else {
  counter = 0;
 }
}
```

Step 3: Compile the program and upload to Arduino UNO board

Now, you can see press the push button the RGB LED will be change red, green, and blue. Each state lasts for 1s each time, and the LED flashes colors repeatedly in such sequence

Lesson 10 - Multiple tones with one Piezo Buzzer

Overview:

In this lesson, you will learn how to use a buzzer. The purpose of the experiment is to generate eight different sounds, each sound lasting 0.5 seconds: from Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz).

COMPONENTS

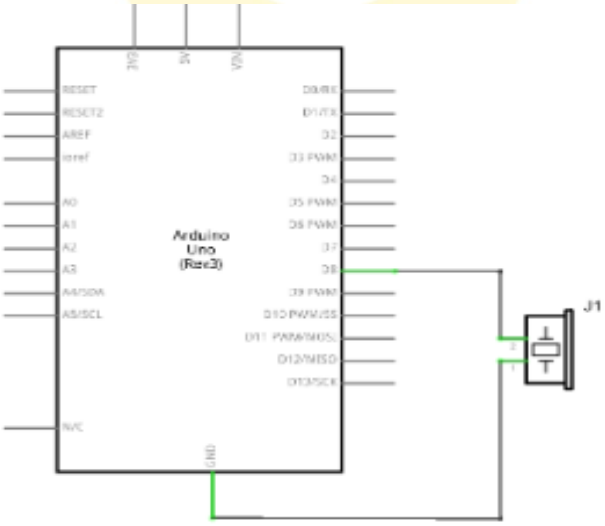
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Buzzer
- 1 x BreadBoard
- 2 x Jumper Wires

Principles

The working principle of buzzer it to use PWM generating audio to make the air to vibrate. Appropriately changed as long as the vibration frequency, it can generate different sounds. For example, sending a pulse of 523Hz, it can generate Alto Do, pulse of 587Hz, it can generate midrange Re, pulse of 659Hz, it can produce midrange Mi. By the buzzer, you can play a song.

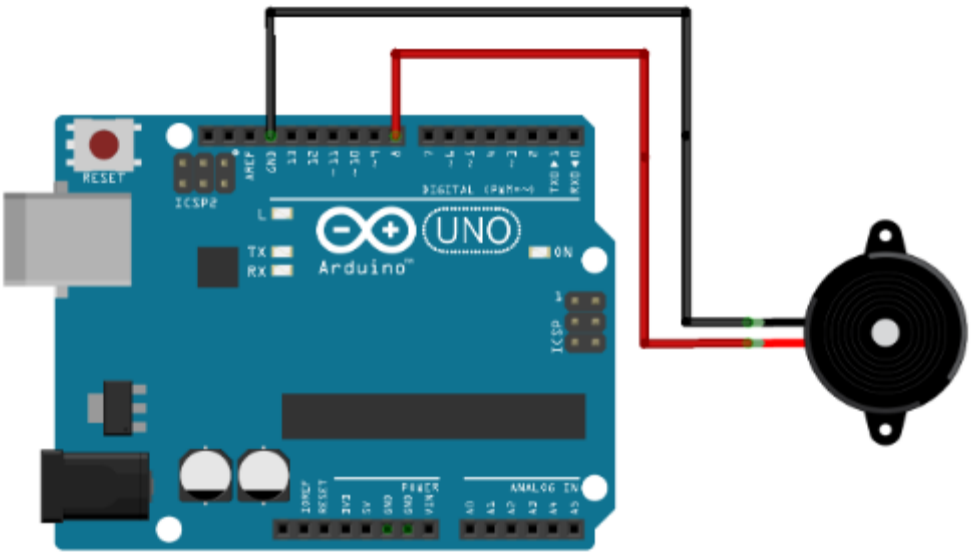
We should be careful not to use the UNO R3 board analog Write () function to generate a pulse to the buzzer, because the pulse output of analog Write () is fixed (500Hz).

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:  10 Buzzer Arduino Circuit  .ino Description:
Let, buzzer Multiple tones. ***************************************/

const int buzzer= 8;
void setup(){
pinMode (buzzer,OUTPUT);
 }

void loop(){
tone(buzzer,1000);
delay(1000);
noTone(buzzer);
delay(1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 11 - Digital thermometer using arduino and LM35 Temperature sensor

Overview:

In this project we have made an Arduino based digital thermometer to display the current ambient temperature and temperature changes on a LCD unit in real time.

Components

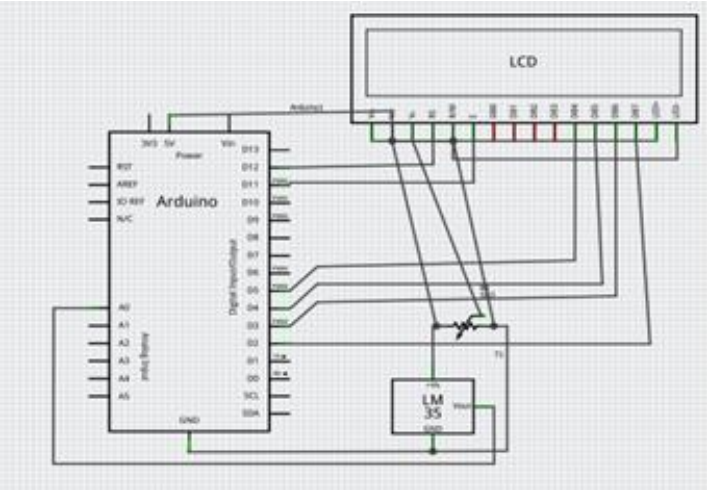
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Lm35
- 1 x LCD(16 x 2)Display
- 1 x potentiometer
- 1 x BreadBoard
- Jumper Wires

Principle

In this Arduino LM35 temperature sensor interfacing, Arduino Uno is used to control the whole process. An LM35 temperature sensor is used for sensing environment temperature which gives 1 degree temperature on every 10mV change at its output pin. You can easily check it with voltmeter by connecting Vcc at pin 1 and Ground at pin 3 and output voltage at pin 2 of LM35 sensor. For an example if the output voltage of LM35 sensor is 250m volt that means the temperature is around 25 degree Celsius.

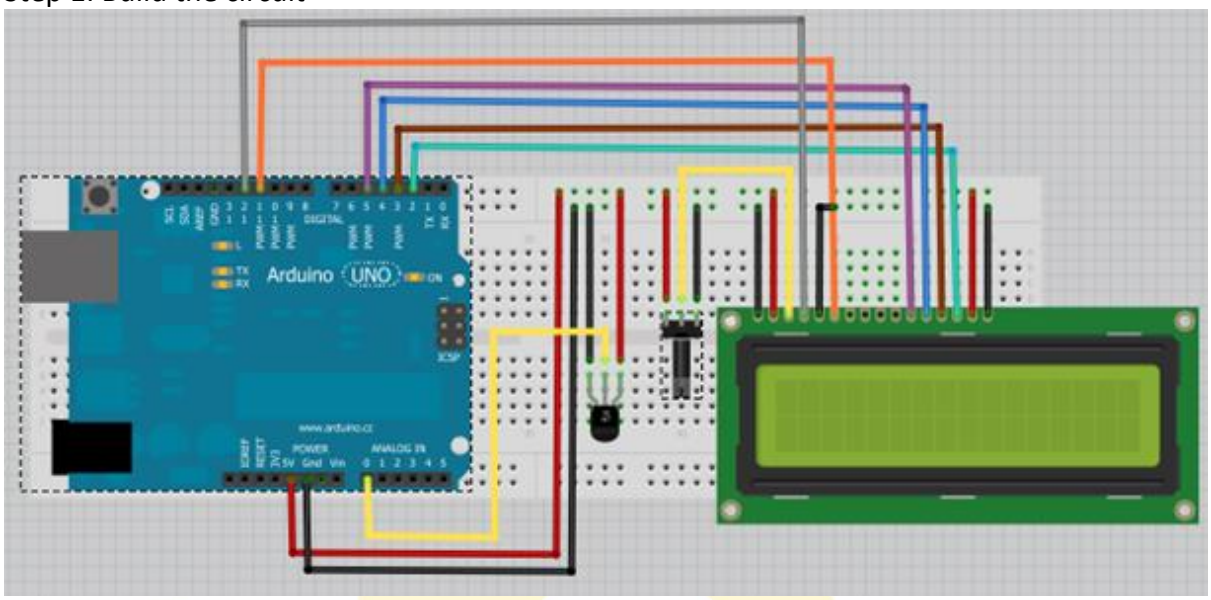
Arduino reads output voltage of temperature sensor by using Analog pin A0 and performs the calculation to convert this Analog value to a digital value of current temperature. After calculations arduino sends these calculations or temperature to 16x2 LCD unit by using appropriate commands of LCD.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   11 Digital thermometer using arduino and LM35 Temperature sensor.ino Description:
Let, LM35 using LCD display. ******************************************/

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int inPin = 0;
void setup()
 {
   lcd.begin(16, 2);
 }
void loop()
 {
   int value = analogRead(inPin);
   lcd.setCursor(0, 1);
   float millivolts = (value / 1024.0) * 5000;
   float celsius = millivolts / 10;
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print(celsius);
   lcd.print(" Celsius");
   lcd.setCursor(0,1);
   lcd.print((celsius * 9)/ 5 + 32);
   lcd.print(" Fahrenheit");
   delay(10000);
 }
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 12 - Seeing the light using Photo resistor with an arduino

Overview:

In this lesson, we will learn how to measure the light intensity by photo resistor.

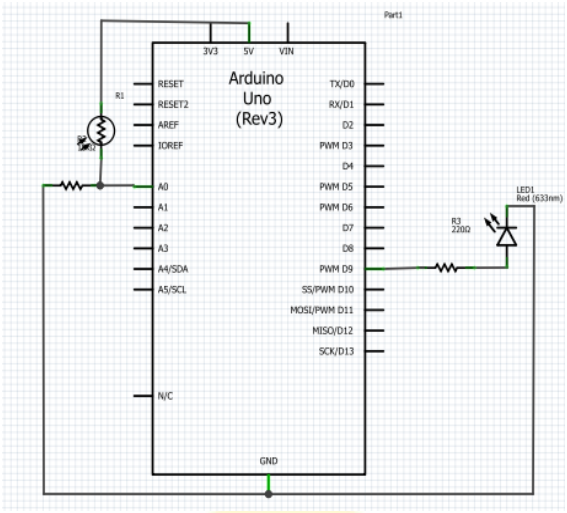
Components

- 1 x Arduino UNO
- 1 x USB cable
- 1 x 10kΩ Resistor
- 1 x 220Ω Resistor
- 1 x LED
- 1 x Photo Resistor
- 1 x BreadBoard
- Jumper wires

Principle:

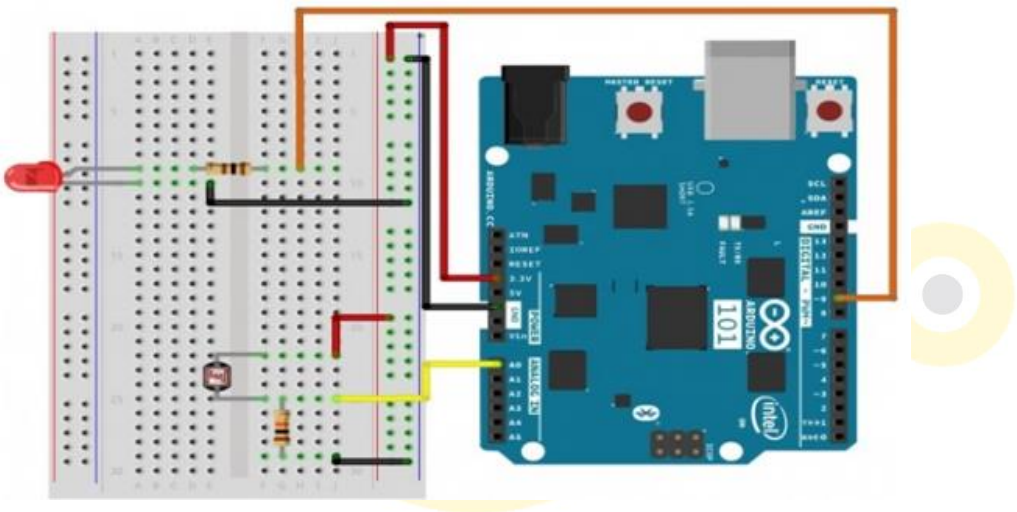
A photo resistor is a light-controlled variable resistor. The resistance of a photo resistor decreases with the increasing incident light intensity; in other words, it exhibits photoconductivity. A photo resistor can be applied in light-sensitive detector circuits.
A photo resistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megohms (MΩ), while in the light, a photo resistor can have a resistance as low as a few hundred ohms. If incident light on a photo resistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their whole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photo resistor can substantially differ among dissimilar devices. Moreover, unique photo resistors may react substantially differently to photons within certain wavelength bands.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:  12 Seeing the light using photo resistor with an arduino Description:
Let, photoresistor using LED blinks. ***************************************/

const int sensorPin = 0;
const int ledPin = 9;
int lightCal;
int lightVal;

void setup()   {
 pinMode(ledPin, OUTPUT);
```

```
 lightCal = analogRead(sensorPin);
  }
void loop()    {
  lightVal = analogRead(sensorPin);
  if (lightVal < lightCal - 50)   {
  digitalWrite(9, HIGH);
  }  else    {
    digitalWrite(9, LOW);
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 13 – LCD 1602 Display

Overview:

In this lesson, we will learn how to use a character display device - LCD1602 on the Arduino platform. We first make the LCD1602 display a string "Hello world!"

Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x potentiometer
- 1 x LCD(16 x 2) Display
- 1 x BreadBoard
- Jumper Wires

Principle

LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

● A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

● A Read/Write (R/W) pin that selects reading mode or writing mode

● An Enable pin that enables writing to the registers

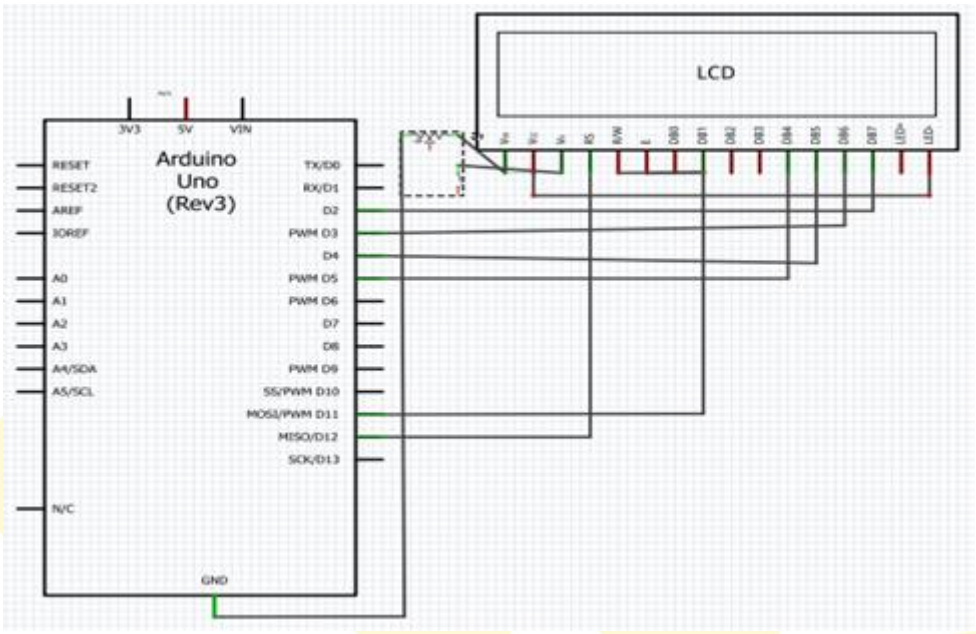
● 8 data pins (D0-D7). The state of these pins (high or low) is the bits that you're writing to a register when you write, or the values when you read.

● There are also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

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

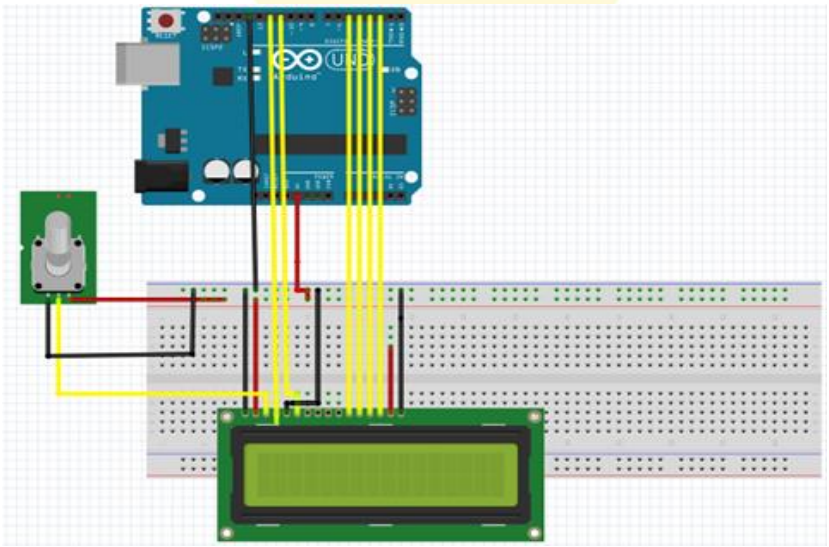
A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/************************************************************
File name:   13  LCD1602 Display arduino.ino Description:
Let, LCD display print HELLO WORLD. ****************************************/
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print ("HELLO WORLD :)");
}
void loop() {
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 14 - Control LED Blink Rate With Potentiometer

Overview:
In this tutorial we have study about that control LED blink rate with using potentiometer.

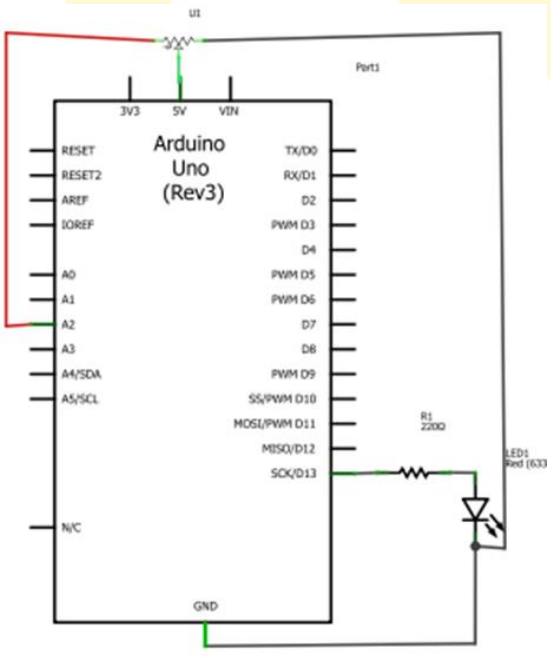
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Breadboard
- 1 x 5mm RED LED
- 1 x 220Ω Resistor
- 1 x 10kΩ Potentiometer
- Jumper wires

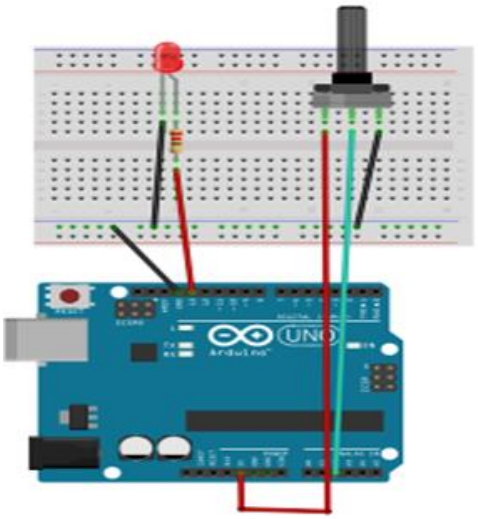
Principle

Connect one pin from your pot to 5V, the center pin to analog pin A2 and the remaining pin to ground. Next, connect a 220 ohm current limiting resistor to digital pin 13, with an LED in series. The long, positive leg (the anode) of the LED should be connected to the output from the resistor, with the shorter, negative leg (the cathode) connected to ground.

Schematic



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   14 Control LED Blink Rate With Potentiometer. Ino Description:
Let, LED Using potentiometer. ************************************/

int potPin = A2;    // select the input pin for the potentiometer
int ledPin = 13;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the sensor
void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}
void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  digitalWrite(ledPin, HIGH);  // turn the ledPin on
  delay(val);                  // stop the program for some time
  digitalWrite(ledPin, LOW);   // turn the ledPin off
  delay(val);                  // stop the program for some time
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 15 - Relay Module

Overview:

In this lesson, you will learn how to use a relay module.

Components

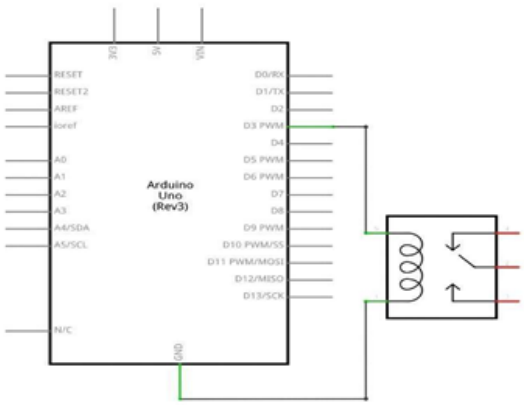
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x BreadBoard
- 1 x Relay Module
- Jumper wire

Principle

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used as in solid state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long-distance telegraph circuits as amplifiers. They repeated the signal coming in from one circuit and retransmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

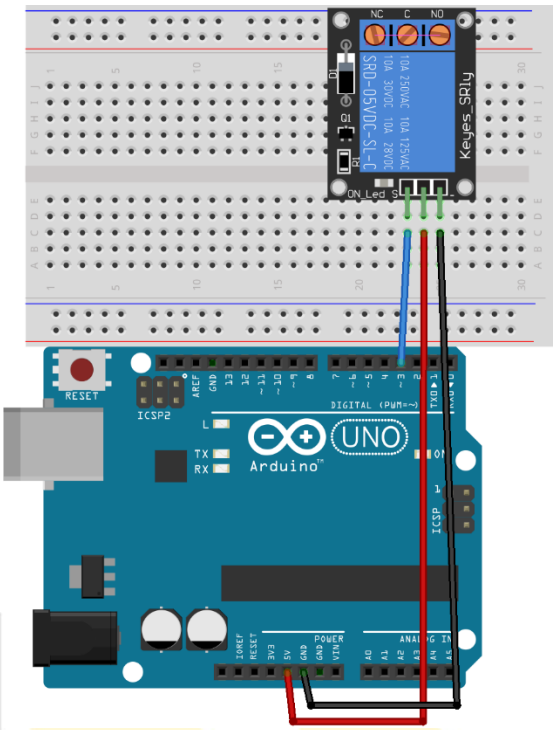
A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform the switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults. In modern electric power systems, these functions are performed by digital instruments called "protective relays".

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   15 Relay Module. ino Description:
Let, Relay use. ***************************************************/

int relayPin = 3;
void setup()   {
  pinMode(relayPin, OUTPUT);
}
 void loop()  {
   digitalWrite(relayPin, HIGH);   // turn the relay on (HIGH is the voltage level)
  delay(2000);            // wait for a second
  digitalWrite(relayPin, LOW);    // turn the relay off by making the voltage LOW
  delay(2000);            // wait for a second
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 16 - DC Motor Direction Control

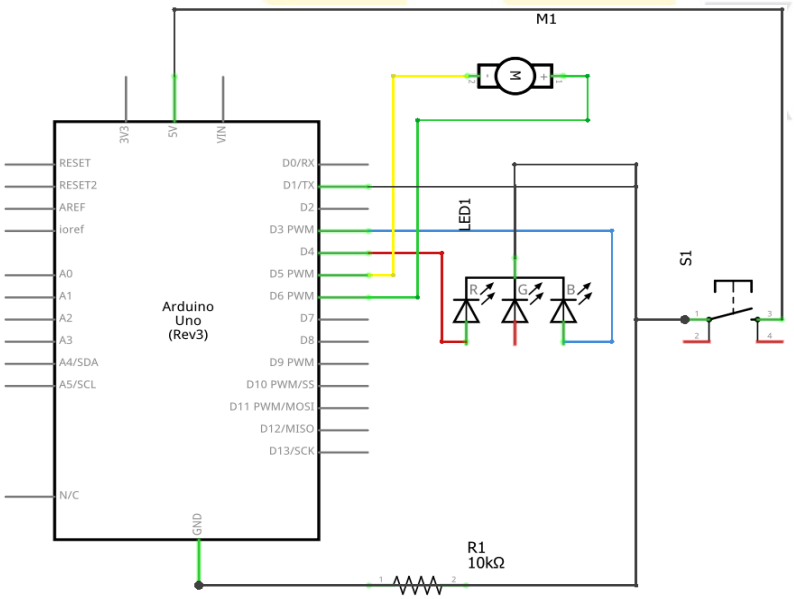
Overview:
In this tutorial we have study about DC Motor control forward direction or revere direction using RGB LED.

COMPONENTS

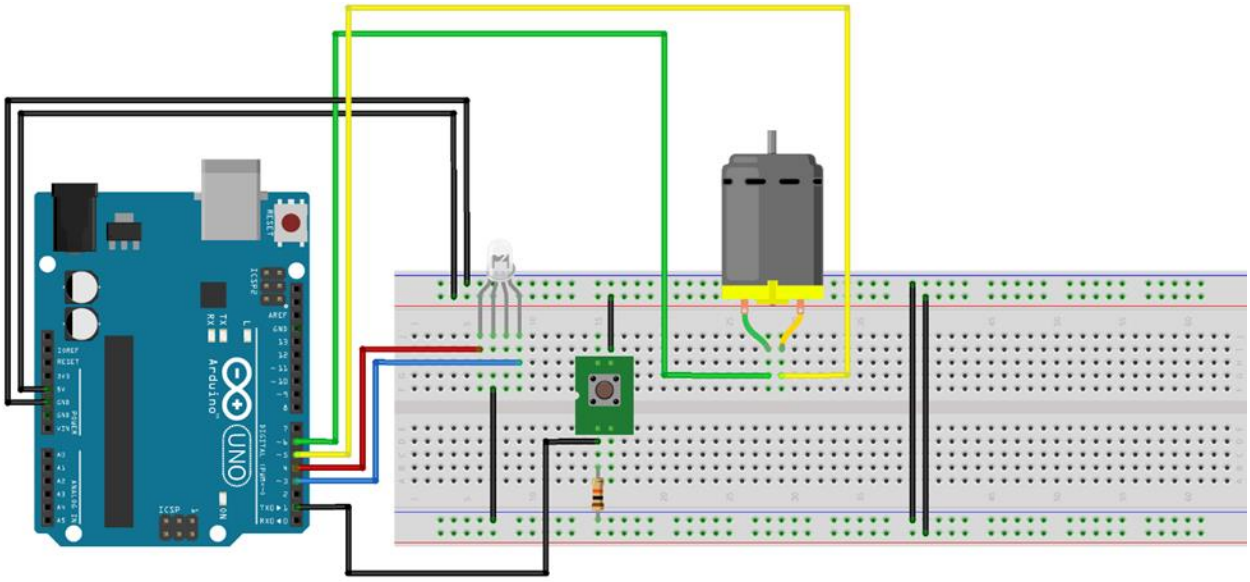
- 1 x Arduino UNO
- 1 X USB Cable
- 1 x BreadBoard
- 1 x RGB LED
- 1 x Push Button
- 1 x 5v Dc Motor
- 1 x 10kΩ Resistor
- Jumper wires

Principle:
Control the rotation of the DC Motor either clockwise or counter clockwise thru the switch and make the RGB LED as indicator e.g. Blue for clockwise and Red for counter clockwise.

Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   16 DC Motor Direction Control.ino Description:
Let, DC motor control by RGB LED. ***************************************/
const int inputPin=1;
const int blue=3;
const int red=4;
const int motorPin1=5, motorPin2=6;
int dir = LOW;
int prevState=0, currentState=0;

void setup()   {
pinMode(inputPin, INPUT);
pinMode(motorPin1, OUTPUT);
pinMode(motorPin2, OUTPUT);
pinMode(blue, OUTPUT);
pinMode(red, OUTPUT);
}
void loop ()   {
currentState=digitalRead(inputPin);
if (currentState!= prevState)
 {
  if (currentState == HIGH)
   {
     dir = !dir;
   }
 }
 prevState = currentState;
  if (dir==HIGH)
```

```
{
 digitalWrite(motorPin1,HIGH);
 digitalWrite(motorPin2,LOW);
 digitalWrite(blue,HIGH);
 digitalWrite(red,LOW);
}
 else     {
 digitalWrite(motorPin1,LOW);
 digitalWrite(motorPin2,HIGH);
 digitalWrite(blue,LOW);
 digitalWrite(red,HIGH);
}
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 17 - Control Servo Motor

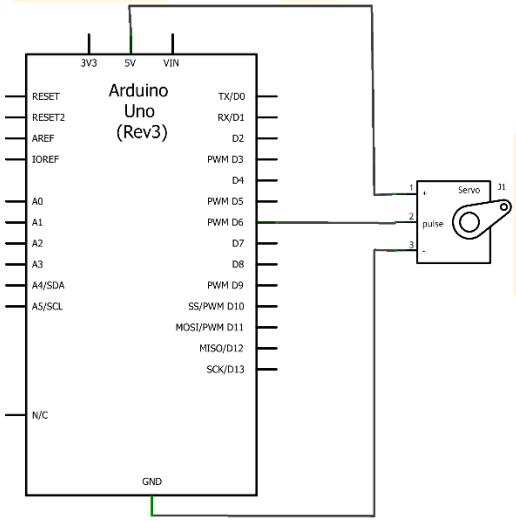
Overview:

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your UNO R3 board. These pulses tell the servo what position it should move to. A servo has three wires, the brown wire is GND, the red one is VCC, and the orange one is signal line.

COMPONENTS:

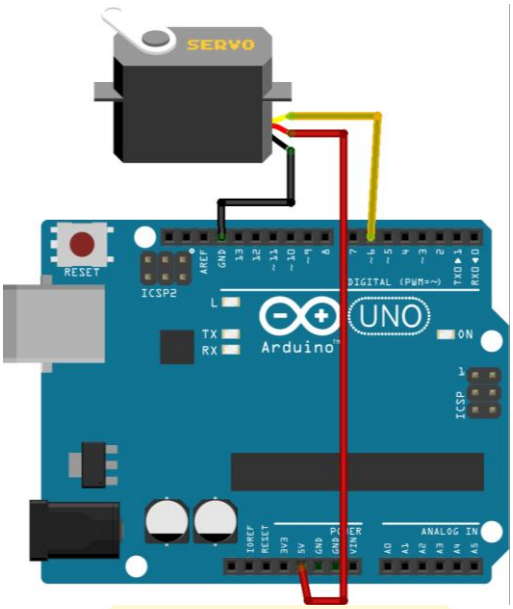
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Servo Motor
- 1 x BreadBoard
- 2 x Jumper Wires

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   17   Control Servo Motor  .ino Description:
Let, servo motor rotate ************************************************************/

# include <Servo.h>
Servo myservo;
int pos = 0;
void setup()  {
myservo.attach(6);
}
void loop()   {
  for (pos = 0; pos <= 180; pos += 5) {
  myservo.write(pos);
  delay(15);
}
  for (pos = 180; pos >= 0; pos -= 5) {
  myservo.write(pos);
  delay(15);
}
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 18 - Stepper Motor Control

Overview:
In this lesson, we will learn how to control a stepper motor.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Stepper Motor
- 1 x Motor Driver Module
- 1 x Battery
- Jumper wires

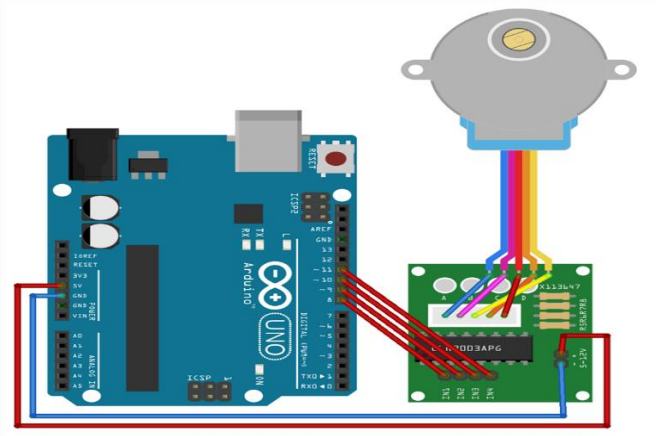
Principle
Most **stepper motors** will operate only with the help of a driver module. This is because the controller module (In our case Arduino) will not be able to provide enough current from its I/O pins for the motor to operate. So we will use an external module like **ULN2003** module as **stepper motor driver**. There are a many types of driver module and the rating of one will change based on the type of motor used.

The circuit Diagram for the **arduino stepper motor control project** is shown above. We have used the 28BYJ-48 Stepper motor and the ULN2003 Driver module. To energize the four coils of the stepper motor we are using the digital pins 8,9,10 and 11. The driver module is powered by the 5V pin of the Arduino Board.

But, power the driver with External Power supply when you are connecting some load to the steppe motor. Since I am just using the motor for demonstration purpose I have used the +5V rail of the Arduino Board. Also remember to connect the Ground of the Arduino with the ground of the Diver module.

Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   18  stepper Motor Control  .ino Description:
Let, Control stepper motor. ******************************************/
#include <Stepper.h>
#define STEPS 64
Stepper stepper(STEPS, 8, 9, 10, 11);
int previous = 0;
void setup()  {
 stepper.setSpeed(30);
}

void loop()   {
  int val = analogRead(0);
  stepper.step(val - previous);
  previous = val;
}
```

Step 3: Compile the program and upload to Arduino UNO board

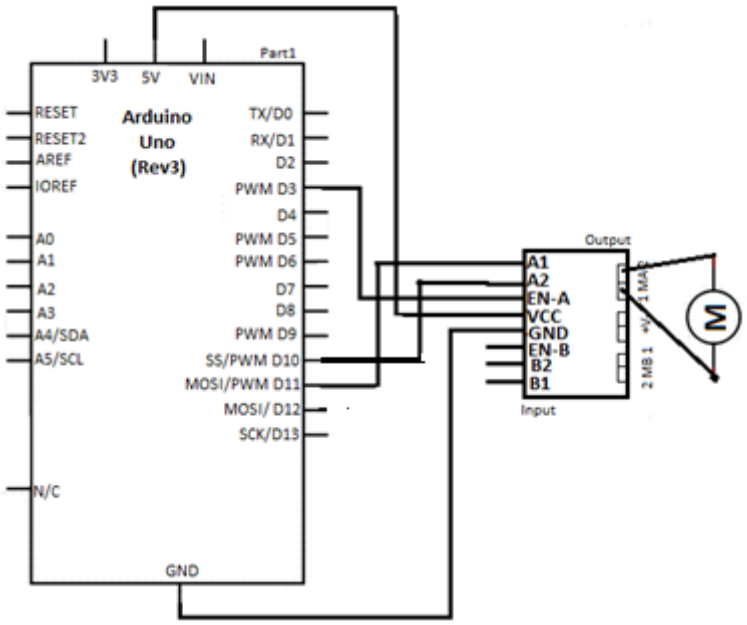
Lesson 19 – L293-DC Motor Control

Overview:
In this lesson, we will learn how to control a L293 Driver Module Control DC Motor
Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x L293 Driver Module
- 1 x DC Motor
- Jumper wires

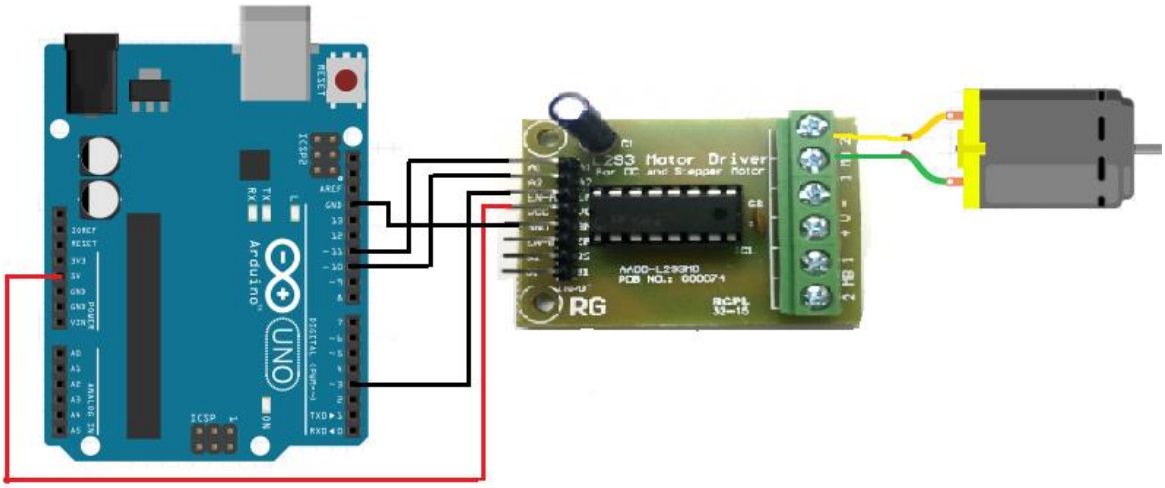
Principle
A motor driver is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver act as an interface between Arduino and the motors .The most commonly used motor driver IC's are from the L293D. L293D consist of two H-bridge. H-bridge is the simplest circuit for controlling a low current rated motor.
There are 2 INPUT pins, 2 OUTPUT pins and 2 ENABLE pin for each motor. L293D consist of two H-bridge. H-bridge is the simplest circuit for controlling a low current rated motor

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   19 L293 Driver Module-DC Motor Control .ino Description:
Let, L293 Driver Module Control DC motor. ***************************/
int enableA = 3;
int MotorA1 = 11;
int MotorA2 = 10;
void setup() {
  Serial.begin (9600);
  //configure pin modes
  pinMode (enableA, OUTPUT);
  pinMode (MotorA1, OUTPUT);
  pinMode (MotorA2, OUTPUT);
 }
 void loop() {
  //enabling motor A
  Serial.println ("Enabling Motors");
  digitalWrite (enableA, HIGH);
   delay (1000);
  //do something
  Serial.println ("Motion Forward");
  digitalWrite (MotorA1, LOW);
  digitalWrite (MotorA2, HIGH);
   //3s forward
  delay (3000);
  Serial.println ("Motion Backwards");
  //reverse
  digitalWrite (MotorA1,HIGH);
  digitalWrite (MotorA2,LOW);
    //5s backwards
  delay (3000);
   Serial.println ("Stoping motors");
  //stop
  digitalWrite (enableA, LOW);
  delay (3000);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 20 - Interfacing tilt sensor with arduino

Overview:

In this lesson, we will learn how to use the tilt switch and change the state of an LED by changing the angle of tilt switch.

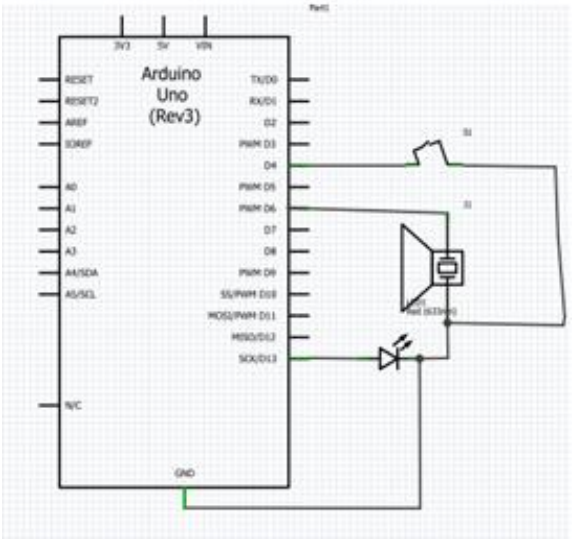
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Tilt Switch (SW-520D)
- 1 x LED
- 1 x Buzzer
- 1 x Breadboard
- Several jumper wires

Principle

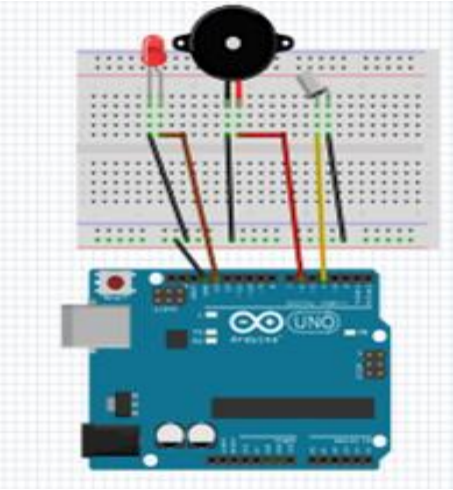
The tilt switch is also called the ball switch. When the switch is tilted in the appropriate direction, the contacts will be connected, tilting the switch then buzzer will operated the opposite direction causes the metallic ball to move away from that set of contacts, thus breaking that circuit.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   20 Interfacing tilt sensor with arduino. ino Description:
Let, Tilt sensor using buzzer tone the LED will blinks. ***************************/
int inPin = 4;                    //Establishes the number of the input pin (tilting sensor)
int reading;                      //Establishes the current reading from the input pin (tilting sensor)
int RedLedPin = 13;          //Establishes the number of the Red LED output pin
const int SpeakerPin = 6; // Establishes the number of the Speaker/Buzzer pin

void setup(){
  pinMode (inPin, INPUT);              //This will set inPin as INPUT
  pinMode (RedLedPin, OUTPUT); //This will set RedLedPin as OUTPUT
}

void loop () {
  reading = digitalRead(inPin);     //This will read the output from the inPin(tilt sensor)
  if (reading == 1) {                    //If the reading is 1 or True it will process the codes under it
    digitalWrite(RedLedPin, HIGH);   // if the tilt sensor is tilted, it will turn the red LED ON

    tone(SpeakerPin, 494, 500);     // if the tilt sensor is tilted, turn the Speaker ON
   delay(500);                  //pause for 0.5 seconds

  } else {
    digitalWrite(RedLedPin, LOW);    //Turns off LED if no there is no tilt
 }
  delay(200);   // pause 200 milliseconds between readings
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Now, when you lean the breadboard at a certain angle, you will see the state of LED is changed.

Lesson 21 - Arduino Flame Sensor

Overview:
In this project we have to study about the project is a simple Arduino flame sensor.

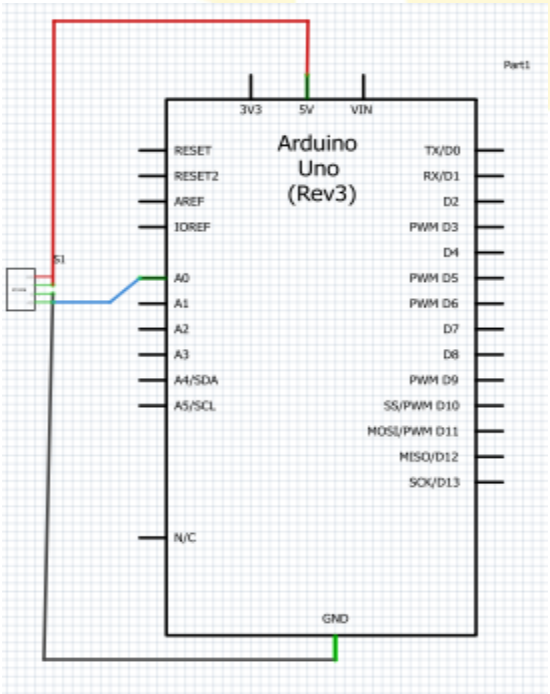
Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Flame Sensor
- 1 x Breadboard
- Jumper wire

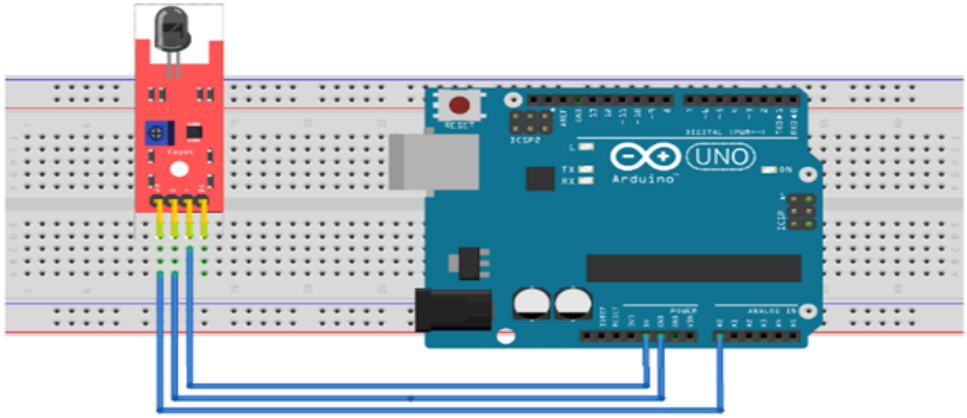
Principle:
The digital and analog interfaces on the KY-026, use a lighter or a candle to interact with the flame detector module.

The digital interface will send a HIGH signal when fire is detected by the sensor, turning on the LED on the Arduino (pin A0). Turn the potentiometer clock-wise to increase the detection threshold and counter-clockwise to decrease it.

Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:  21 Arduino  Flame Sensor.ino Description:
Let, fire is detected by the sensor turning on the LED ***************************/
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup()  {
   Serial.begin(9600);
}
void loop() {
            int sensorReading = analogRead(A0);
            int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
   switch (range) {
  case 0:    // A fire closer than 1.5 feet away.
   Serial.println("** Close Fire **");
   break;
  case 1:    // A fire between 1-3 feet away.
   Serial.println("** Distant Fire **");
   break;
  case 2:    // No fire detected.
   Serial.println("No Fire");
   break;
  }
  delay(1);  // delay between reads
}
```

Step 3: Compile the program and upload to Arduino UNO board

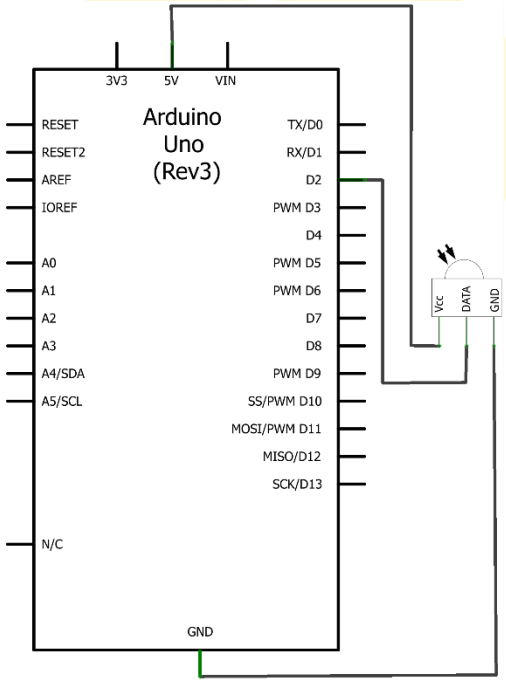
Lesson 22 - IR_Receiver using arduino

Overview:
Using an IR Remote is a great way to have wireless control of your project.

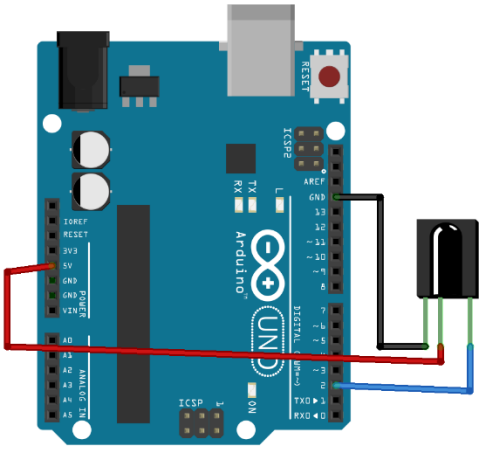
Components:

- 1 x Arduino UNO
- 1 x USB cable
- 1 x IR sensor
- 1 x Remote
- 1 x Breadboard
- Jumper wires

Principle
IR detectors are little microchips with a photocell that are tuned to listen to infrared light. They are almost always used for remote control detection - every TV and DVD player has One of these in the front to listen for the IR signal from the clicker. Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels. IR light is not visible to the human eye, which means it takes a little more work to test a setup.
Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   22- IR_Receiver using arduino .ino Description:
Let,IR Remote is a great way to have wireless control *********************************/
#include <IRremote.h>      // Include the lib
int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 23- LED Matrix display 8 x 8 dots (MAX7219)

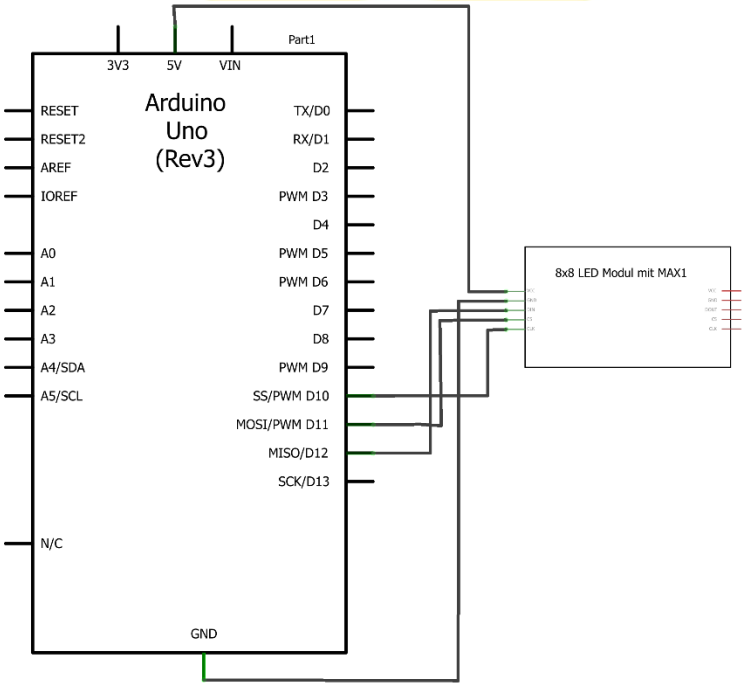
Overview:
In this project, we will learn about LED Matrix Displays 8×8 dots (MAX7219) Interface with Arduino.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Dot Matrix(MAX7219)
- 1 x Breadboard
- Jumper wires

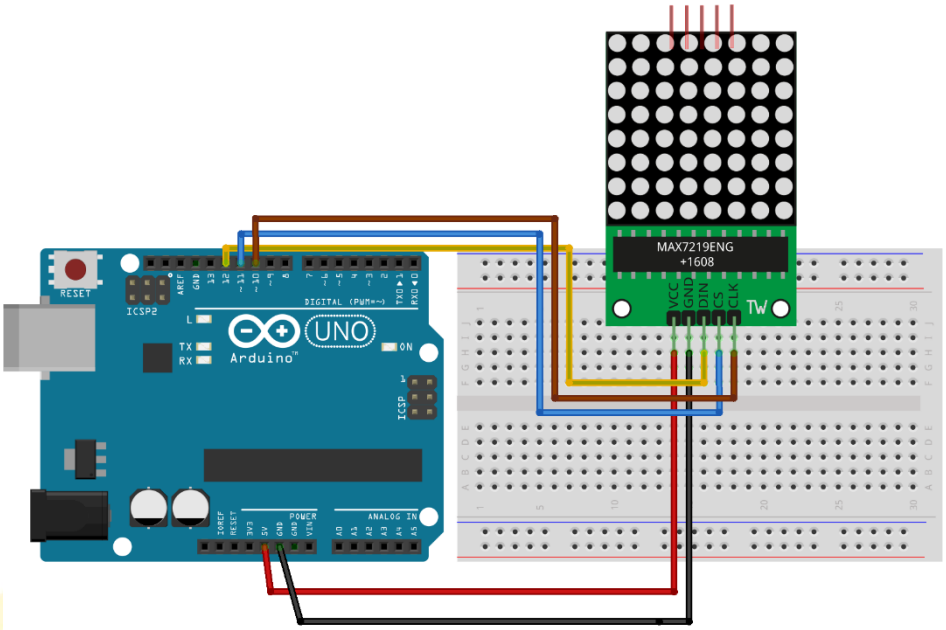
Principle:
The MAX7219/MAX7221 are compact, serial input/output common-cathode display drivers that interface to microcontrollers and microprocessors to control 7-segment numeric LED displays of up to 8 digits, bar-graph displays, or 64 individual LEDs. Included on the MAX7219 chip is a BCD code-B decoder, a multiplex scan circuitry, a segment and digit drivers, and an 8×8 static RAM that stores each digit. Only one external resistor is required to set the segment current for all LEDs. The MAX7221 is compatible with SPI, QSPI, and MICROWIRE communication protocols, and has slew-rate-limited segment drivers to reduce EMI. Individual digits of the connected LED display may be addressed and updated without rewriting the entire display. The MAX7219/MAX7221 also allow the user to select code-B decoding or no-decode for each digit.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:  23- LED Matrix display 8 x 8 dots (MAX7219).ino Description:
Let, LED Matrix Displays 8×8 dots (MAX7219) Interface with Arduino.**************************/

#include <LedControl.h>
int DIN = 12;
int CS =  11;
int CLK = 10;
byte e[8]=    {0x7C,0x7C,0x60,0x7C,0x7C,0x60,0x7C,0x7C};
byte d[8]=    {0x78,0x7C,0x66,0x66,0x66,0x66,0x7C,0x78};
byte u[8]=    {0x66,0x66,0x66,0x66,0x66,0x66,0x7E,0x7E};
byte c[8]=    {0x7E,0x7E,0x60,0x60,0x60,0x60,0x7E,0x7E};
byte eight[8]= {0x7E,0x7E,0x66,0x7E,0x7E,0x66,0x7E,0x7E};
byte s[8]=    {0x7E,0x7C,0x60,0x7C,0x3E,0x06,0x3E,0x7E};
byte dot[8]=  {0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18};
byte o[8]=    {0x7E,0x7E,0x66,0x66,0x66,0x66,0x7E,0x7E};
byte m[8]=    {0xE7,0xFF,0xFF,0xDB,0xDB,0xDB,0xC3,0xC3};
LedControl lc=LedControl(DIN,CLK,CS,0);
void setup()  {
 lc.shutdown(0,false);      //The MAX72XX is in power-saving mode on startup
 lc.setIntensity(0,15);     // Set the brightness to maximum value
 lc.clearDisplay(0);        // and clear the display
}
void loop()    {
   byte smile[8]=   {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C};
   byte neutral[8]= {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C};
```

```
    byte frown[8]=   {0x3C,0x42,0xA5,0x81,0x99,0xA5,0x42,0x3C};
    printByte(smile);
    delay(1000);
    printByte(neutral);
    delay(1000);
    printByte(frown);
    delay(1000);
    printEduc8s();
    lc.clearDisplay(0);
    delay(1000);
}
void printEduc8s() {
  printByte(e);
  delay(1000);
  printByte(d);
  delay(1000);
  printByte(u);
  delay(1000);
  printByte(c);
  delay(1000);
  printByte(eight);
  delay(1000);
  printByte(s);
  delay(1000);
  printByte(dot);
  delay(1000);
  printByte(c);
  delay(1000);
  printByte(o);
  delay(1000);
  printByte(m);
  delay(1000);
}
void printByte(byte character [])  {
  int i = 0;
  for(i=0;i<8;i++)
  {
    lc.setRow(0,i,character[i]);
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board

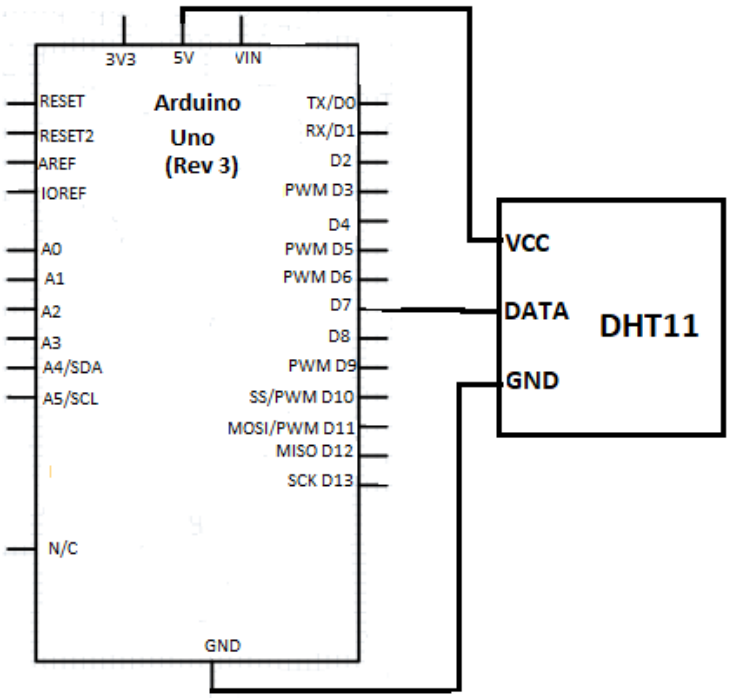
Lesson 24 - Showing the temperature and humidity sensor in using arduino

Overview:
In this tutorial we have to learn about the data from DHT11 sensor send to Arduino UNO and then the humidity and temperature.

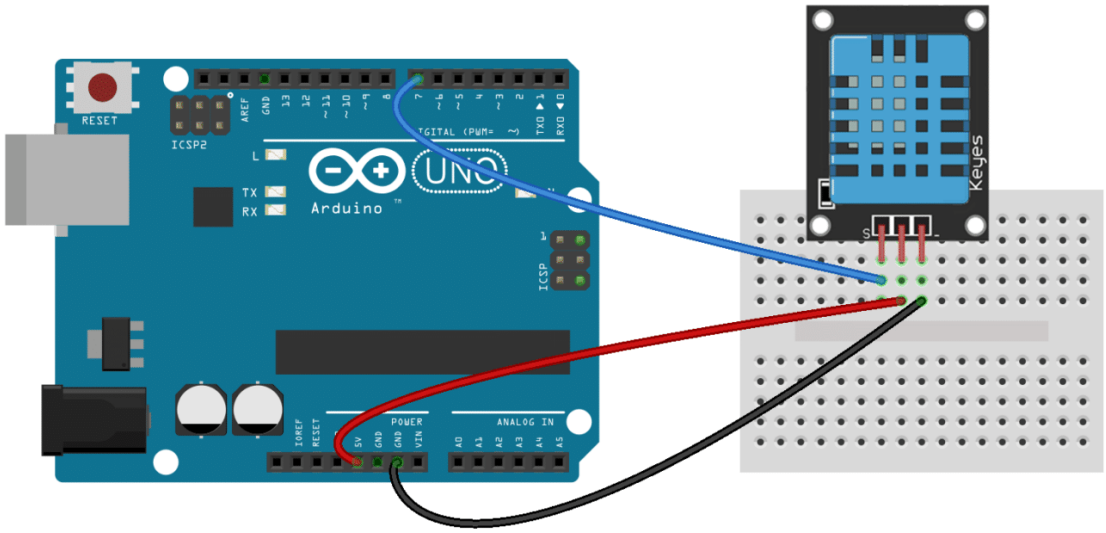
COMPONENTS

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x DTH11 Temperature Sensor
- 1 x BreadBoard
- Jumper Wires

Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/************************************************************
File name:   24  Showing the temperature and humidity sensor in display using arduino.ino
Description:
Let, Showing the temperature using DHT. ****************************/
#include <SimpleDHT.h>
int pinDHT11 = 7;
SimpleDHT11 dht11;

void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println("==================================");
  Serial.println("Sample DHT11...");

  byte temperature = 0;
  byte humidity = 0;
  byte data[40] = {0};
  if (dht11.read(pinDHT11, &temperature, &humidity, data)) {
    Serial.print("Read DHT11 failed");
    return;
  }
    Serial.print("Sample RAW Bits: ");
  for (int i = 0; i < 40; i++) {
    Serial.print((int)data[i]);
    if (i > 0 && ((i + 1) % 4) == 0) {
      Serial.print(' ');
    }
  }
  Serial.println("");
```

```
  Serial.print ("Sample OK: ");
  Serial.print((int)temperature); Serial.print(" *C, ");
  Serial.print((int)humidity); Serial.println(" %");
 delay(1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 25 - Using a 74HC595 Shift Register with an Arduino

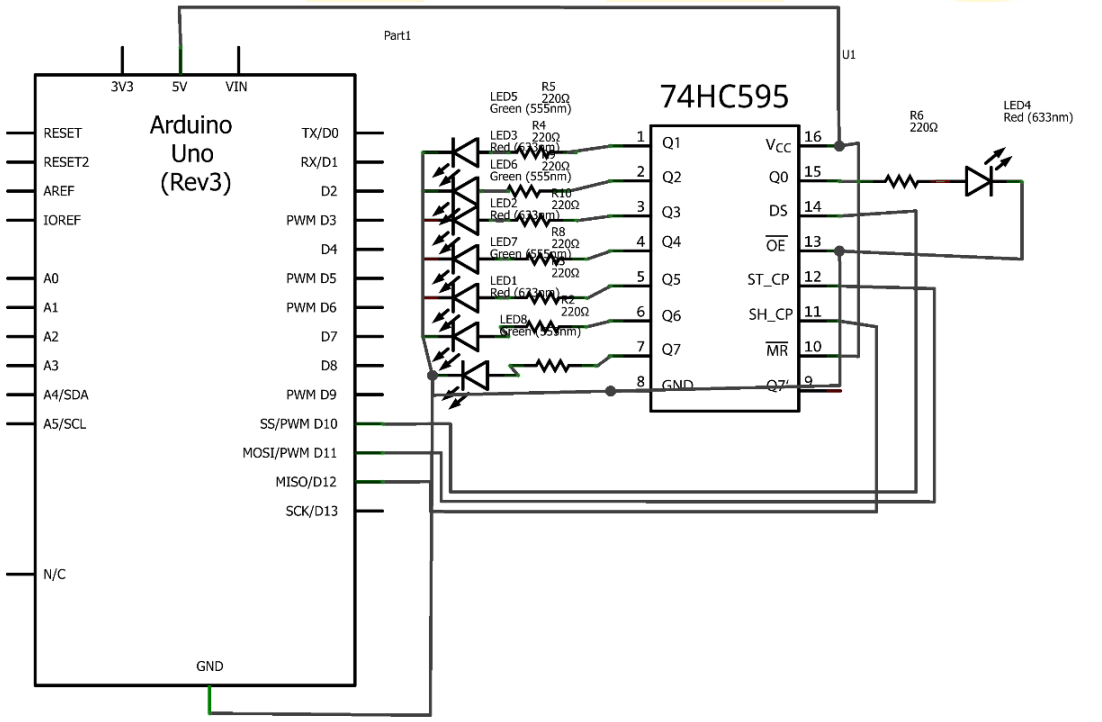
Overview:
In this tutorial you will practical by using the shift register with Arduino Uno to control 8 LEDs.

Components:

- 1 x Arduino UNO
- 1 x USB cable
- 1 x IC 74HC595
- 4 x 5mm RED LED
- 4 x 5mm GREEN LED
- 7 x Resistor
- 1 x Breadboard
- Jumper wires

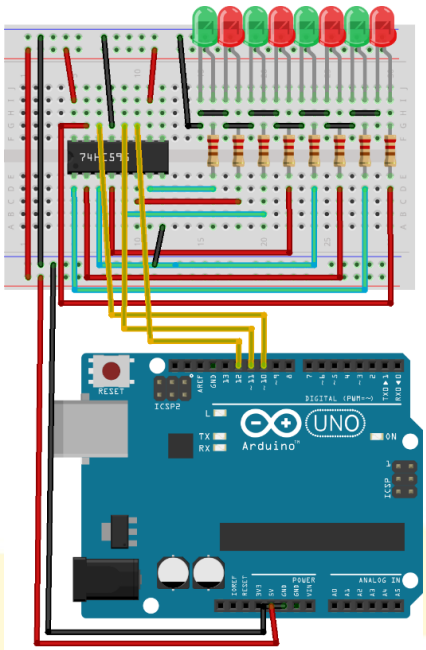
Principle:
Here we are going to send data in eight bit size through a single channel to shift register. The shift register takes the data serially and stores that data in its memory. Once the data is sent by the controller, we are going to send a command to shift register to show the data at the output, with this command the shift register puts out data parallel.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name:   25  using a 74HC595 shift register with an arduino  .ino Description:
Let, IC 74HC595 BLINK LED. ***************************************************/
#include <ShiftRegister74HC595.h>
int data = 10;  // 74HC595 data input pin 14
int clock = 12; // 74hc595 clock (SCK) line pin 11
int latch = 11;   //  74hc595latch line pin 12
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
void setup ()  {
pinMode (data, OUTPUT);
pinMode (clock, OUTPUT);
pinMode (latch, OUTPUT);
}
void loop ()  {
for (int i = 0; i <256; i ++)  {
updateLEDs (i);
delay (500);
}
}
void updateLEDs (int value)  {
digitalWrite (latch, LOW);
shift Out (data, clock, MSBFIRST, ~ value); // serial data output
digitalWrite (latch, HIGH); // latch
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 26 - RC522 RFID Module

Overview:

In this tutorial, you will learn to how to apply the RC522 RFID Reader Module on UNO R3.This module uses the Serial Peripheral Interface (SPI) bus to communicate with controllers such as Arduino, Raspberry Pi, beagle board, etc.

Components:

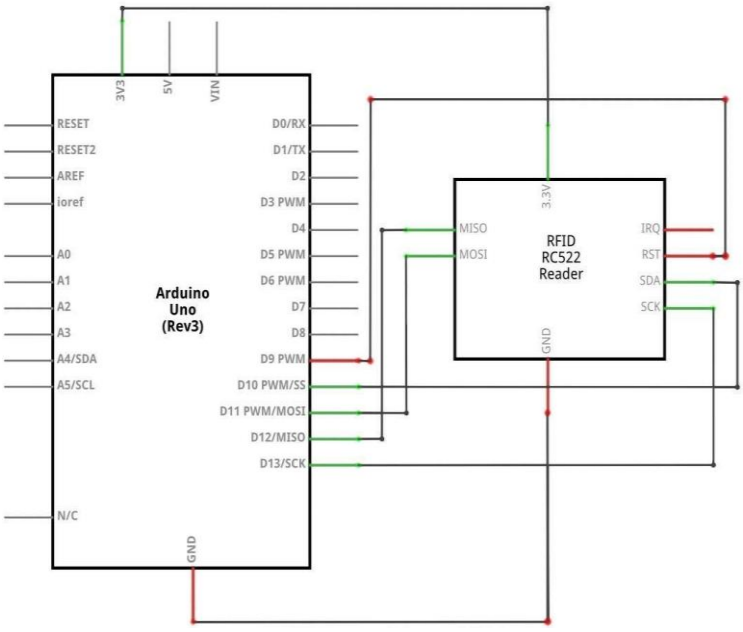
- 1 x Arduino Uno
- 1 x USB cable
- 1 x Rc522 module
- 1 x Breadboard
- Jumper wires

Principle:

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz .The MFRC522 reader supports ISO 14443A / MIFARE® mode.

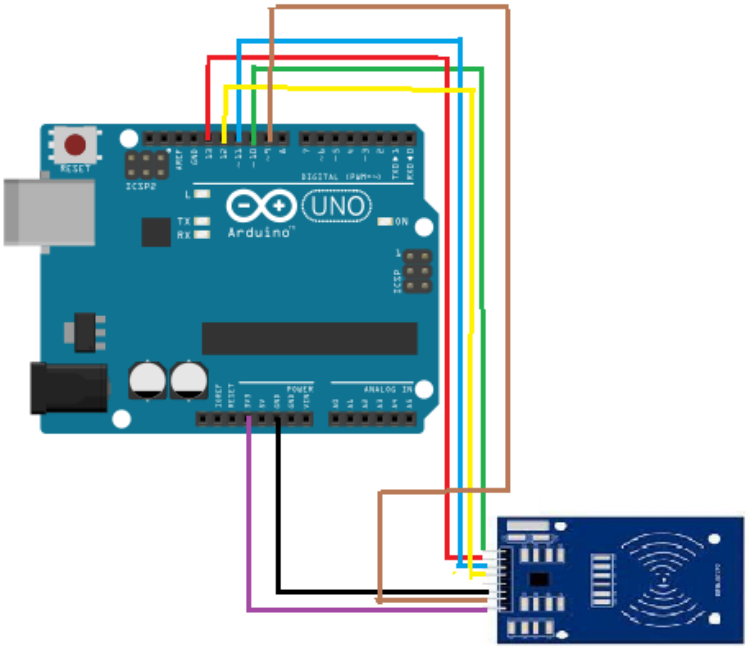
The MFRC522‟s internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE®Classic (e.g. MIFARE® Standard) products. The MFRC522 supports contactless communication using MIFARE® higher transfer speeds up to 848 Kbit/s in both directions.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************
File name: 26- RC522 RFID Module.ino Description:
Let, RFID module. **********************************/
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN   9     // Configurable, see typical pin layout above
#define SS_PIN    10    // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
MFRC522::MIFARE_Key key;
void setup() {
  Serial.begin(9600);  // Initialize serial communications with the PC
  while (!Serial);     // Do nothing if no serial port is opened (added for Arduinos based on
ATMEGA32U4)
  SPI.begin();        // Init SPI bus
  mfrc522.PCD_Init();  // Init MFRC522 card
  Serial.println(F("Warning: this example overwrites the UID of your UID changeable card, use with
care!"));
    for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}
void loop() {
if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    delay(50);
    return;
```

```
  }
   Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
   Serial.print (mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
   Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();
  byte newUid[] = NEW_UID;
  if ( mfrc522.MIFARE_SetUid(newUid, (byte)4, true) ) {
   Serial.println (F ("Wrote new UID to card."));
  }
  mfrc522.PICC_HaltA();
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   return;
  }
  Serial.println(F("New UID and contents:"));
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
  delay(2000);
 }
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 27 – Control LED using clap system

Overview:
In this tutorial we have study about that clap and make sound Led will blink using sound sensor**.**

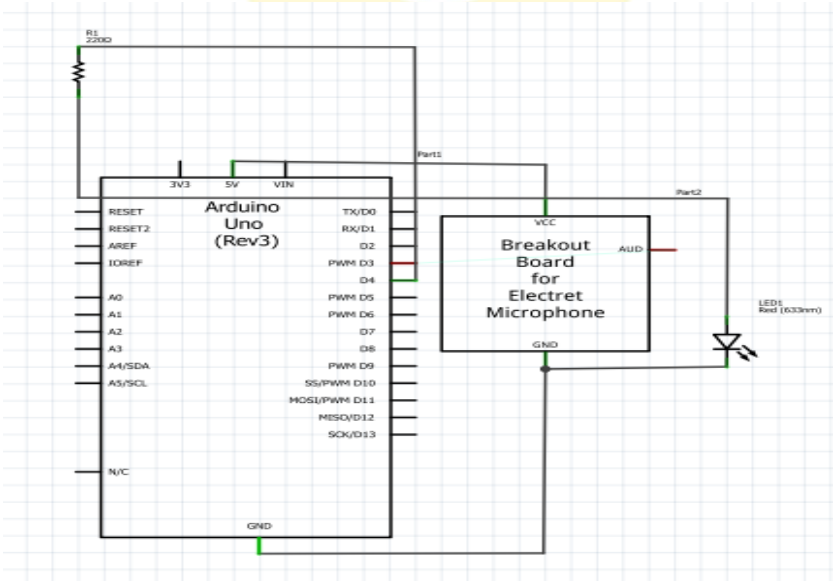
Components:

- 1 x Arduino Uno
- 1 x USB cable
- 1 x Sound Sensor
- 1 x LED
- 1 x 220Ω Resistor
- 1 x Breadboard
- Jumper wires

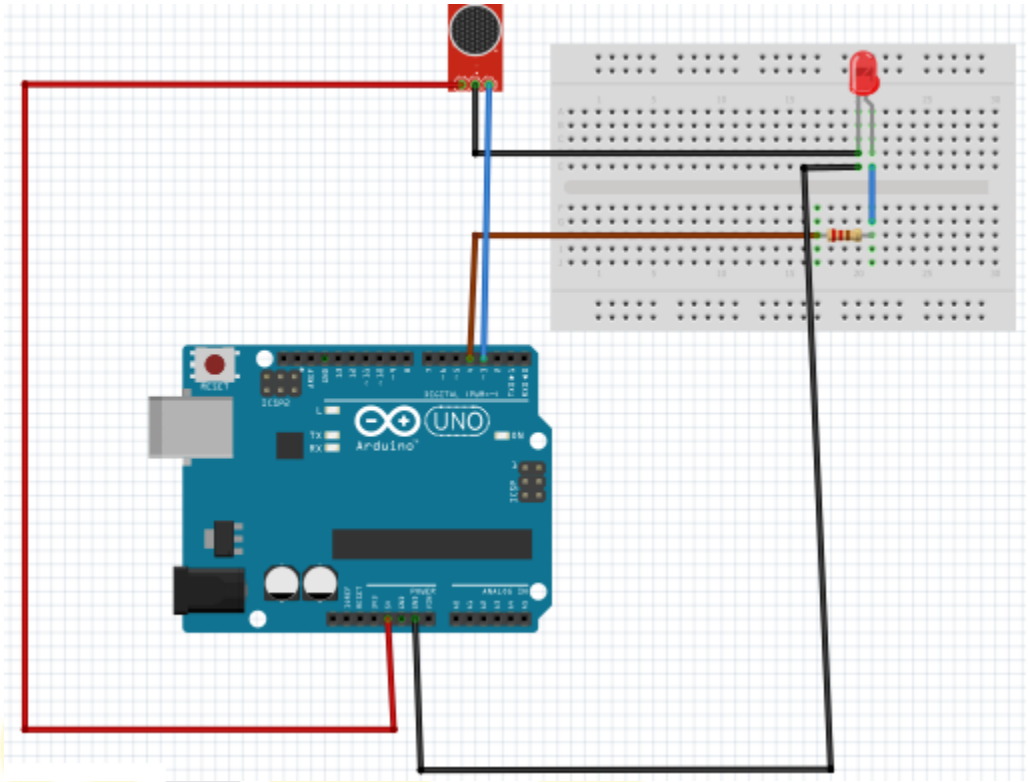
Principle:
KY-038 Sound Sensor using Arduino. It also shows how you can control LED by clap with the help of Arduino and Sound Sensor. We have shown only to control LED, but by using the same concept you can control any electronic device.
KY-038 Sound Sensor having 4 Pins:

- AO – Analog Output
- G – Ground
- + – VCC
- DO – Digital Output

Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 27 - Control led using clap system. No Description:
Let, control led using  clap*************************************/
int soundSensor =3;
int LED = 4 ;
boolean LEDStatus = false;
void setup() {
pinMode(soundSensor,INPUT);
pinMode(LED,OUTPUT);
}
void loop() {
int SensorData=digitalRead(soundSensor);
if(SensorData==1){
if (LEDStatus==false){
LEDStatus=true;
digitalWrite(LED,HIGH);
}
 else  {
LEDStatus=false;
digitalWrite(LED,LOW);
}
}
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 28 – Keypad Module

Overview:
In this project, we will go over how to integrate a keyboard with an UNO R3 board so that the UNO R3 can read the keys being pressed by a user.

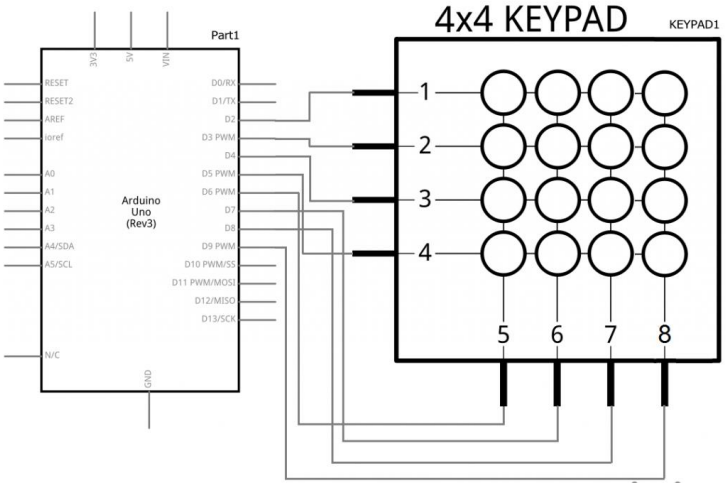
Components:

- 1 x Arduino Uno
- 1 x USB cable
- 1 x Membrane switch module
- 1 x Breadboard
- Jumper wires

Principle

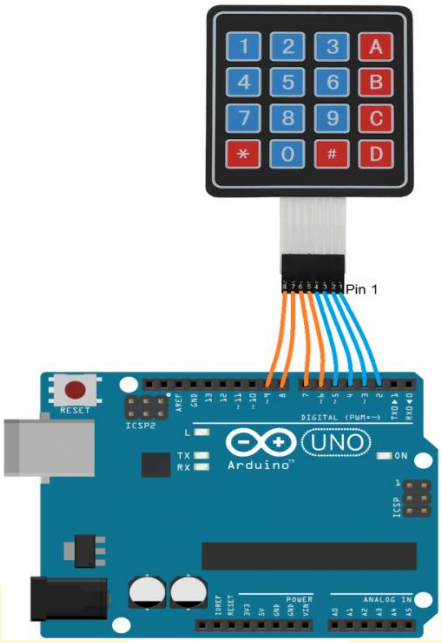
Keypads are used in all types of devices, including cell phones, fax machines, microwaves, ovens, door locks, etc. They're practically everywhere. Tons of electronic devices use them for user input.
So knowing how to connect a keypad to a microcontroller such as an UNO R3 board is very valuable for building many different types of commercial products.
At the end when all is connected properly and programmed, when a key is pressed, it show up at the Serial Monitor on your computer. Whenever you press a key, it shows up on the Serial Monitor. Later, in another project, we will connect the keypad circuit, so that it will get displayed on an LCD. But for now, for simplicity purposes, we start at simply showing the key pressed on the computer.
For this project, the type of keypad we will use is a matrix keypad. This is a keypad that follows an encoding scheme that allows it to have much less output pins than there are keys. For example, the matrix keypad we are using has 16 keys (0-9, A-D, *, #), yet only 8 output pins. With a linear keypad, there would have to be 17 output pins (one for each key and a ground pin) in order to work. The matrix encoding scheme allows for less output pins and thus much less connections that have to make for the keypad to work. In this way, they are more efficient than linear keypads, being that they have less wiring.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 28 - Keypad Module.ino Description:
Let, keypad  ************************************************/
#include <Keypad.h>
const byte ROWS = 4;    //four rows
const byte COLS = 4;    //four columns
char hexaKeys[ROWS][COLS] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the Rows of the keypad pin 8, 7, 6, 5 respectively
byte colPins[COLS] = {5, 4, 3, 2}; //connect to the Columns of the keypad pin 4, 3, 2, 1 respectively
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
void setup()  {
 Serial.begin(9600);
}
void loop(){
 char customKey = customKeypad.getKey();
  if (customKey){
  Serial.println(customKey); // Send the pressed key value to the arduino serial monitor
 }
}
```

Step 3: Compile the program and upload to Arduino UNO board.
When connecting the pins to the UNO R3 board, we connect them to the digital output pins, D9-D2. We connect the first pin of the keypad to D9, the second pin to D8, the third pin to D7, the fourth pin to D6, the fifth pin to D5, the sixth pin to D4, the seventh pin to D3, and the eighth pin to D2. These are the connections in a table:

Keypad Pin	Connects to Arduino Pin...
1	D9
2	D8
3	D7
4	D6
5	D5
6	D4
7	D3
8	D2

Lesson 29 - Analog Joystick Module

Overview:

In this tutorial we will learn how to use the analog joystick module. Analog joysticks are a great way to add some control in your projects.

Components:

- 1 x Arduino Uno
- 1 x USB cable
- 1 x Joystick Module
- 1 x Breadboard
- Jumper wires

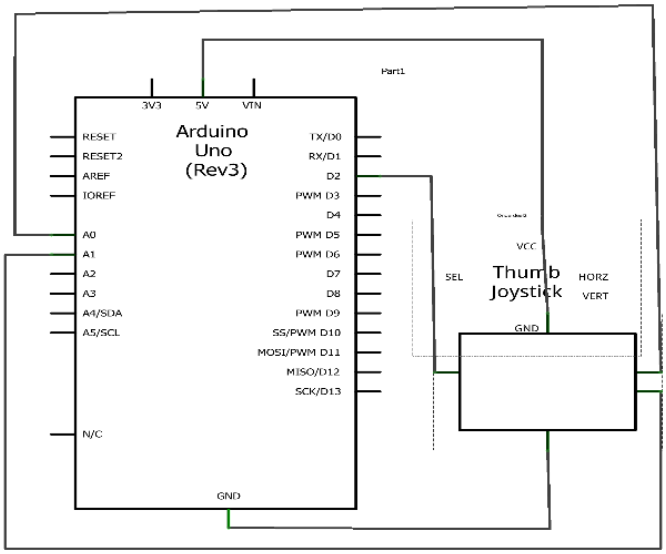
Principle

Joystick

The module has 5 pins: Vcc, Ground, X, Y, Key. Note that the labels on yours may be slightly different, depending on where you got the module from. The thumb stick is analog and should provide more accurate readings than simple „directional" joysticks tact use some forms of buttons, or mechanical switches. Additionally, you can press the joystick down (rather hard on mine) to activate a „press to select" push-button.

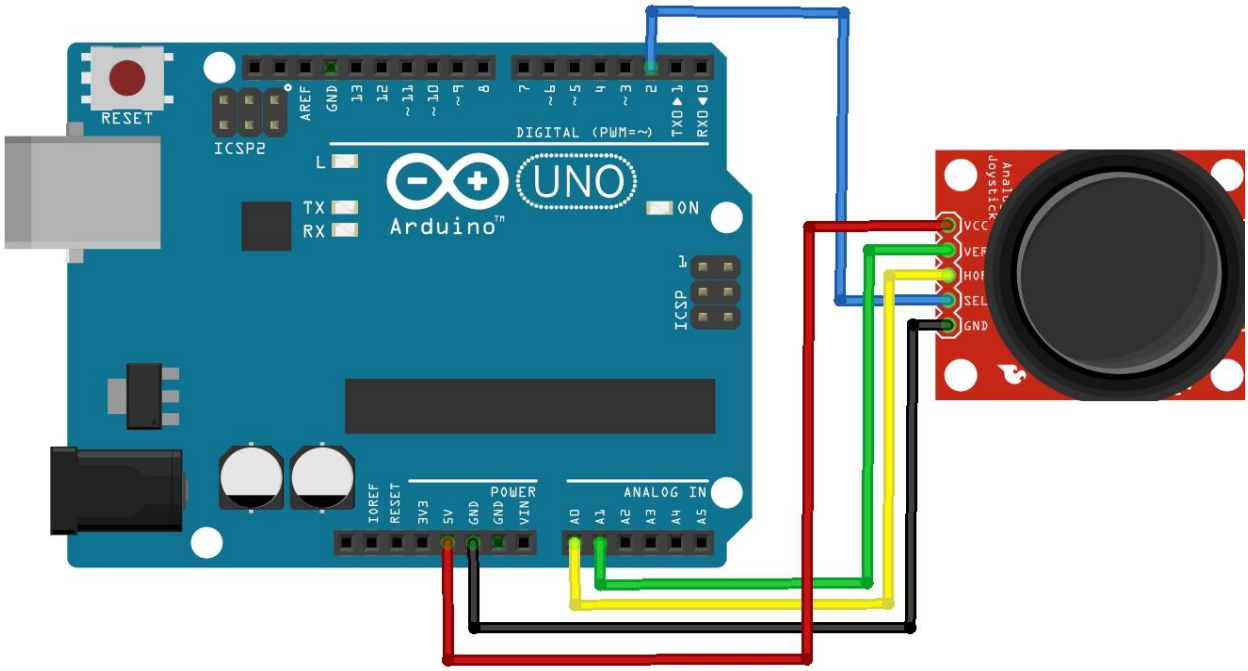
We have to use analog Arduino pins to read the data from the X/Y pins, and a digital pin to read the button. The Key pin is connected to ground, when the joystick is pressed down, and is floating otherwise. To get stable readings from the Key /Select pin, it needs to be connected to Vcc via a pull-up resistor. The built in resistors on the Arduino digital pins can be used. For a tutorial on how to activate the pull-up resistors for Arduino pins, configured as inputs.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 29 - Analog Joystick Module.ino Description:
 Let ,Analog Joystick Module*************************************************/
const int SW_pin = 2; // digital pin connected to switch output
const int X_pin = A0; // analog pin connected to X output
const int Y_pin = A1; // analog pin connected to Y output

void setup() {
  pinMode(SW_pin, INPUT);
  digitalWrite(SW_pin, HIGH);
  Serial.begin(9600);
}
void loop() {
  Serial.print("Switch:  ");
  Serial.print(digitalRead(SW_pin));
  Serial.print("\n");
  Serial.print("X-axis: ");
  Serial.print(analogRead(X_pin));
  Serial.print("\n");
  Serial.print("Y-axis: ");
  Serial.println(analogRead(Y_pin));
  Serial.print("\n\n");
  delay(500);
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 30-Water Level Detection Sensor Module

Overview

In this lesson, you will learn how to use a water level detection sensor module.

Component Required:

- 1 x Arduino Uno
- 1 x USB cable
- 1 x Water level detection sensor module
- 1 x Breadboard
- Jumper wires

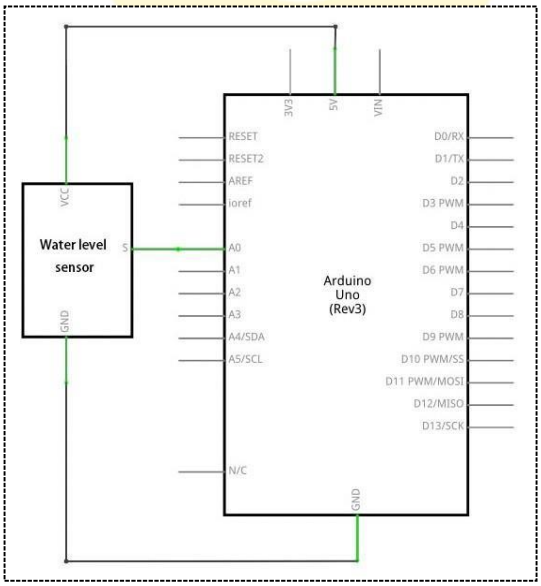
Principle

Water sensor:

A water sensor brick is designed for water detection, which can be widely used in sensing the rainfall, water level, even the liquate leakage. The brick is mainly composed of three parts: an electronic brick connector, a 1 MΩ resistor, and several lines of bare conducting Wires. This sensor works by having a series of exposed traces connected to ground. Interlaced between the grounded traces are the sense traces.

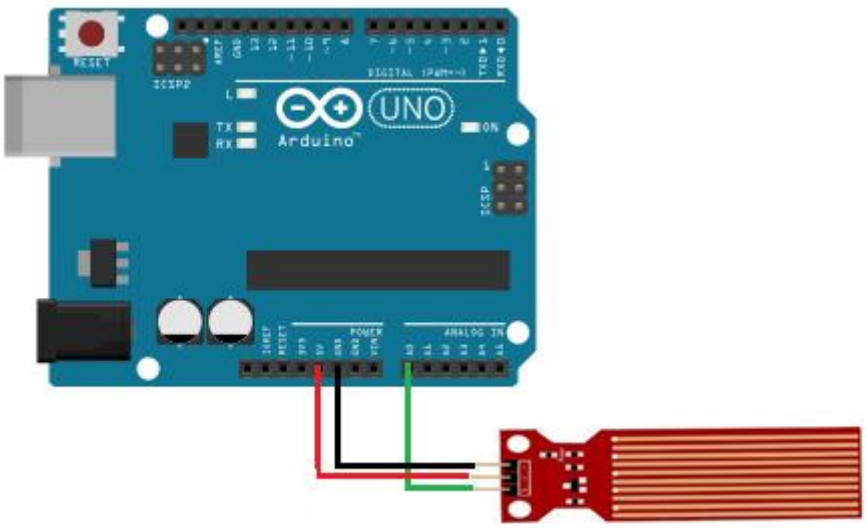
The sensor traces have a weak pull-up resistor of 1 MΩ. The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace. Believe it or not this circuit will work with the digital I/O pins of your UNO R3 board or you can use it with the analog pins to detect the amount of water induced contact between the grounded and sensor traces.

Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 30 - Water Level Detection Sensor Module.ino Description:
Let, Water Level Detection*****************************************/

int adc_id = 0;
int HistoryValue = 0;
char printBuffer[128];
void setup() {
  Serial.begin(9600);
}
void loop()  {
    int value = analogRead(adc_id); // get adc value
    if(((HistoryValue>=value) && ((HistoryValue - value) > 10)) || ((HistoryValue<value)
&& ((value - HistoryValue) > 10)))
    {
      sprintf(printBuffer,"ADC%d level is %d\n",adc_id, value);
      Serial.print(printBuffer);
      HistoryValue = value;
    }
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 31 - Real Time Clock Module

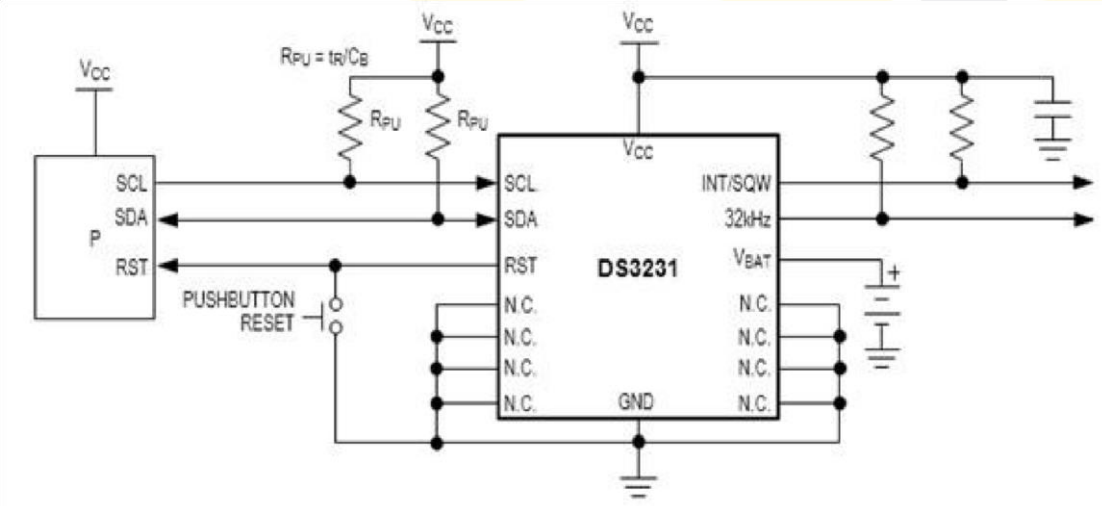
Overview:
In this lesson, you will learn how to use the DS3231, clock module that displays the year, month, day, hour, minute, second and week. Support is via a backup battery trickle charger, which can be used unless being connected to UNO with only three data cables.

Component:

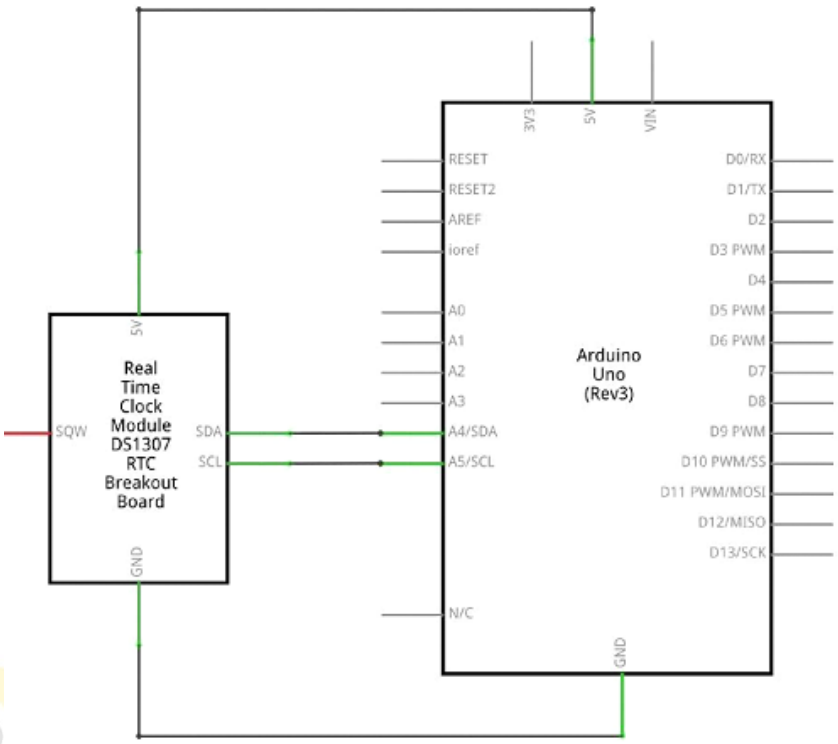
- 1 x Arduino Uno
- 1 x USB cable
- 1 x DS3231 real time clock module
- 1 x Breadboard
- Jumper wires

Principle:

DS3231
The DS3231 is a simple time-keeping chip. It has an integrated battery, so the clock can continue keeping time even when unplugged.

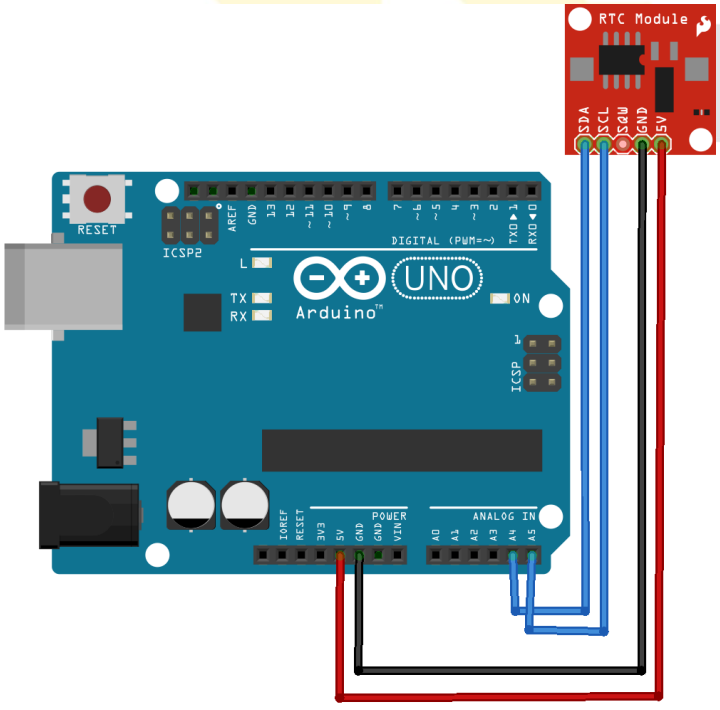


Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 31 - Real Time Clock Module .ino Description:
Let, Real Time Clock. ********************************************/

#include <Wire.h>
#include <DS3231.h>
  DS3231 clock;
  RTCDateTime dt;
  void setup()  {
  Serial.begin(9600);
  Serial.println("Initialize DS3231");;
  clock.begin();
  clock.setDateTime(__DATE__, __TIME__);
}
void loop()  {
  dt = clock.getDateTime();
  Serial.print("Raw data: ");
  Serial.print(dt.year);   Serial.print("-");
  Serial.print(dt.month);  Serial.print("-");
  Serial.print(dt.day);    Serial.print(" ");
  Serial.print(dt.hour);   Serial.print(":");
  Serial.print(dt.minute); Serial.print(":");
  Serial.print(dt.second); Serial.println("");
  delay(1000);
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 32 – HC-SR04 Project

Overview:
In this lesson, we will learn how to measure the distance by the ultrasonic distance sensor.

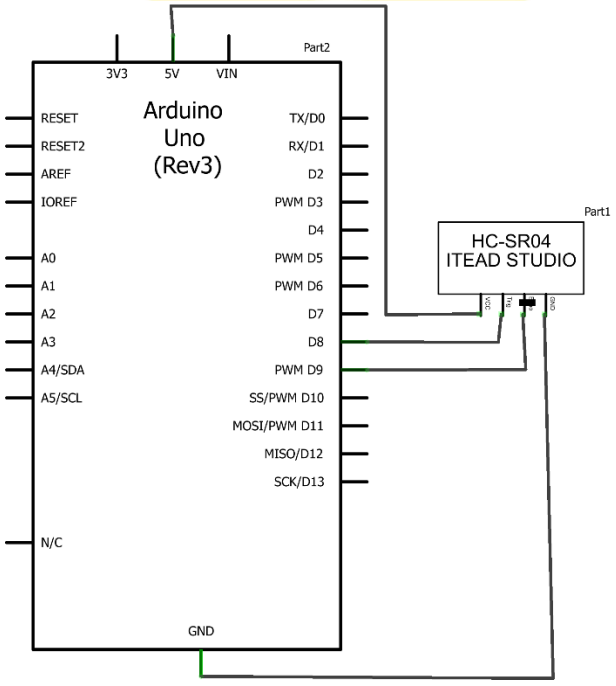
Component:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Ultrasonic sensor
- 1 x Breadboard
- Jumper wire

Principle:

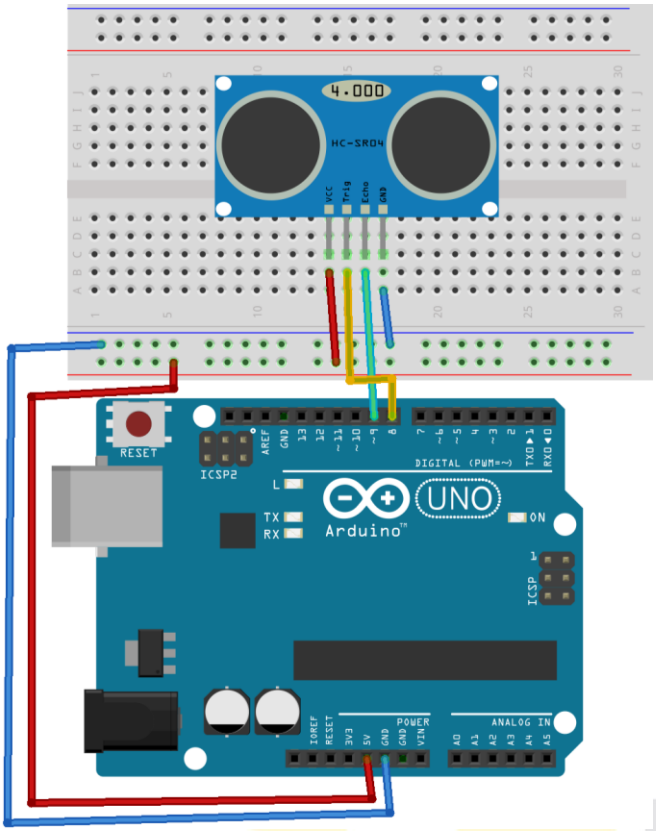
The HC-SR04 uses sound instead of light as the GP2D12 module does. The HC-SR04 sends a ping as a submarine does and measure the time between sending and receiving anything back when an object is in front of the sensor. Because using sound for its measurements we can reach up to 4 meters. The module is about 45x20x15 mm in size and has a 4 pin connection. Two pins are needed to power the module with 5 Volts. The working current is about 15 mA. One pin is the trigger ping and the last one is used to read the result of the measurements, the echo pin. The measuring angle from the HC-SR04 is 15 degree. At 4 meter distance this should be a beam of about 1 meter. At 1 meter this is 26 cm so we have to keep this in mind when using this information.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 32 – HC-SR04 Project .ino Description:
Let, Distance measure with ultrasonic sensor.************************/

#include <HCSR04.h>
const int TriggerPin = 8;     //Trig pin
const int EchoPin = 9;        //Echo pin
long Duration = 0;
void setup() {
  pinMode(TriggerPin,OUTPUT);  // Trigger is an output pin
  pinMode(EchoPin,INPUT);      // Echo is an input pin
  Serial.begin(9600);          // Serial Output
}
void loop()  {
  digitalWrite(TriggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH);         // Trigger pin to HIGH
  delayMicroseconds(10);                  // 10us high
  digitalWrite(TriggerPin, LOW);          // Trigger pin to HIGH
  Duration = pulseIn(EchoPin,HIGH);       // Waits for the echo pin to get high
```

```
  long Distance_mm = Distance(Duration);   // Use function to calculate the distance

  Serial.print("Distance = ");           // Output to serial
  Serial.print(Distance_mm);
  Serial.println(" mm");
   delay(1000);                        // Wait to do next measurement
}
long Distance(long time)
{
   long DistanceCalc;                  // Calculation variable
   DistanceCalc = ((time /2.9) / 2);     // Actual calculation in mm
   //DistanceCalc = time / 74 / 2;        // Actual calculation in inches
   return DistanceCalc;                // return calculated value
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 33- Interface ultrasonic sensor

Overview:
The HC-SR 04 is famous ultrasonic range sensor, and it's very easy to use with many microcontrollers. This article will give your idea about to interface ultrasonic sensor with arduino board.

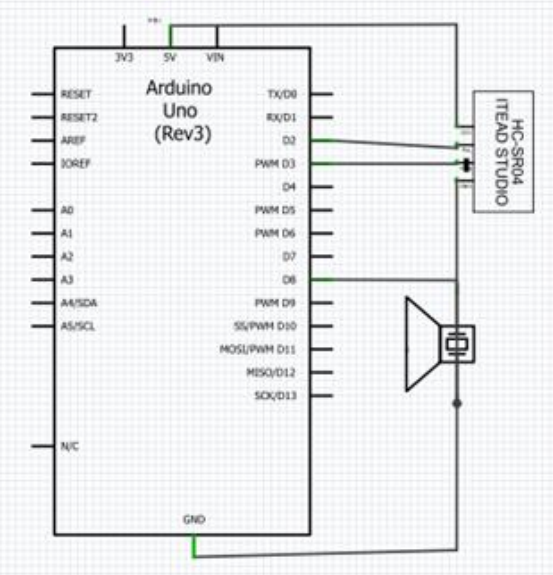
Components:

- 1 x Arduino UNO
- 1 x USB cable
- 1 x Ultrasonic Sensor
- 1 x Buzzer
- 1 x Breadboard
- Jumper wires

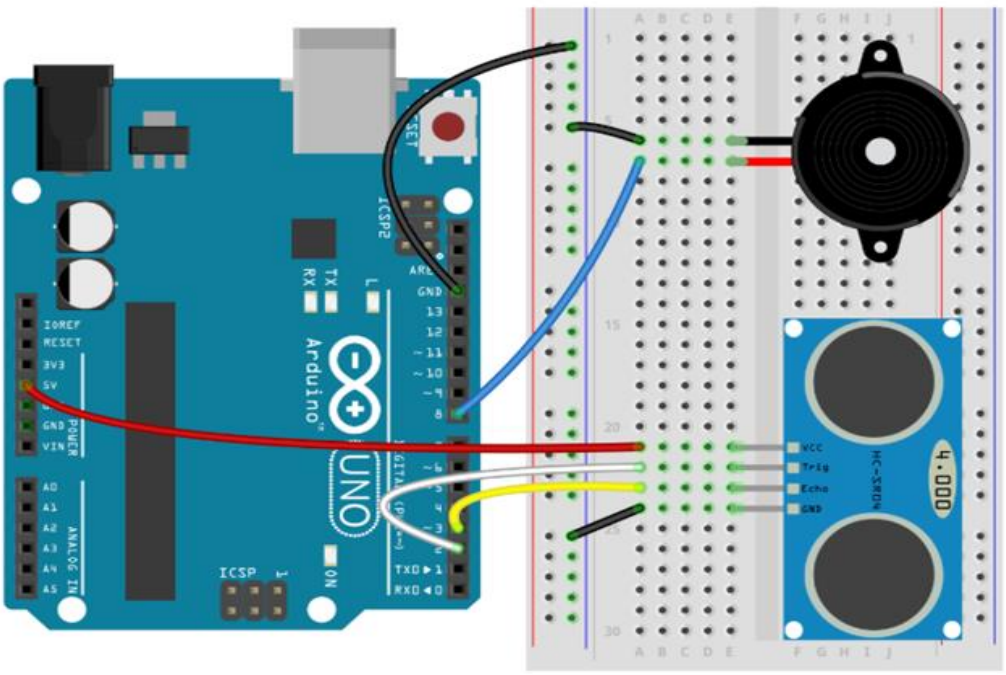
Principle:

This circuit can used for security alarm, invisible fence with alarm or distance measurement (Cm) etc…
The HC-SR04 sensor module works under the principle of ultrasonic wave transmission and ultrasonic wave reflection (echo). The time difference between transmitter waves to echo wave is calculated for to find the distance between sensor module and objects.

Schematic:



Procedure:
Step 1: Build the circuit



Step 2: Program

```
/*************************************************************
File name: 33 Interface ultrasonic sensor.ino Description:
Let, Security alarm or distance measurement**********************/
void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, INPUT);
  }
void loop() {
  digitalWrite(2, HIGH);
  delayMicroseconds(10);
  digitalWrite(2, LOW);
  long duration = pulseIn(3, HIGH);
     if (duration == 0) {
    return;
     }
  long distance = duration / 58.2;
  tone(8, 1000, 20);
  delay(20);
  noTone(8);
  delay(2 * distance);
}
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson – 34 PIR sensor

Overview:

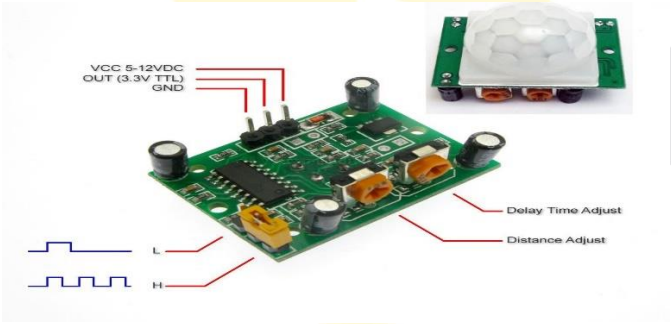
In this tutorial I will show you how to make basic interfacing of PIR sensor with the arduino.

Components

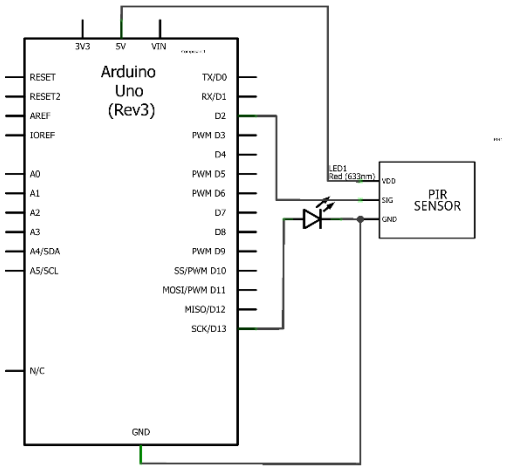
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x PIR sensor
- 1 x BreadBoard
- Several jumper wires

Principle

PIR sensor detects a human being moving around within approximately 10m from the sensor. This is an average value, as the actual detection range is between 5m and 12m.PIR are fundamentally made of a pyro electric sensor, which can detect levels of infrared radiation. This PIR sensor is connected to the arduino pin 2 that is dedicated for interrupt 0.The interrupt will be triggered at the rising edge of the input. And after receiving the interrupt. It will call a function (ISR) alarm ON.

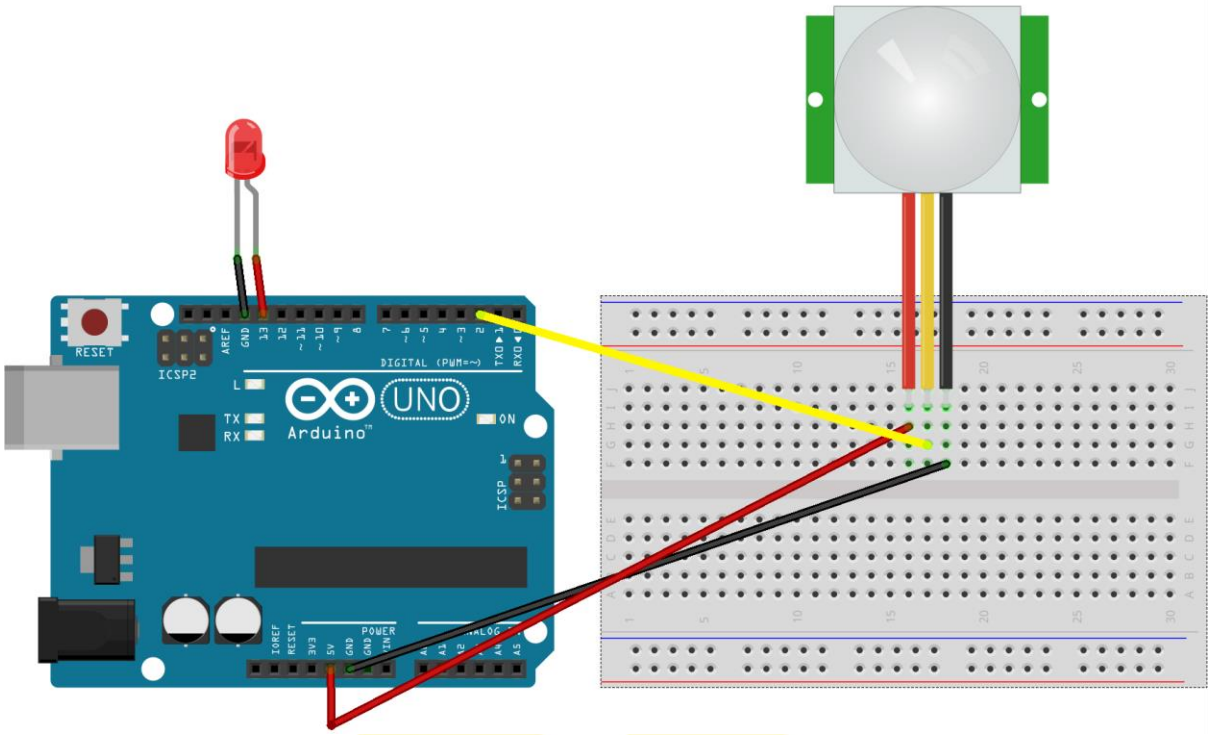


Schematic



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:  34. PIR sensor .ino Description:
Let, PIR sensor motion detected LED will blinks.*****************************/
int led = 13;                // the pin that the LED is atteched to
int sensor = 2;              // the pin that the sensor is atteched to
int state = LOW;             // by default, no motion detected
int val = 0;                 // variable to store the sensor status
(value)
void setup() {
  pinMode(led, OUTPUT);      // initalize LED as an output
  pinMode(sensor, INPUT);    // initialize sensor as an input
  Serial.begin(9600);        // initialize serial
}
void loop(){
  val = digitalRead(sensor);   // read sensor value
  if (val == HIGH) {           // check if the sensor is HIGH
    digitalWrite(led, HIGH);   // turn LED ON
    delay(100);                // delay 100 milliseconds
        if (state == LOW) {
      Serial.println("Motion detected!");
      state = HIGH;       // update variable state to HIGH
    }
  }
  else {
```

```
      digitalWrite(led, LOW); // turn LED OFF
      delay(200);             // delay 200 milliseconds
  if (state == HIGH){
        Serial.println("Motion stopped!");
        state = LOW;       // update variable state to LOW
    }
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board .After compile the code we can see that PIR sensor detected the movement.

Lesson 35 - PNP transistor

Overview:

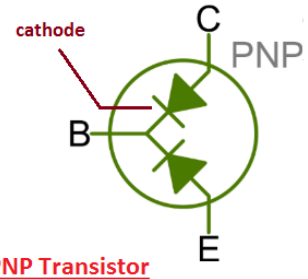
In this tutorial I will show you how to LED blink with interface PNP Transistor using the arduino.

Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x PNP Transistor
- 1 x Red LED
- 1 x 220Ω Resistor
- 1 x 10kΩ Resistor
- 1 x BreadBoard
- Several jumper wires

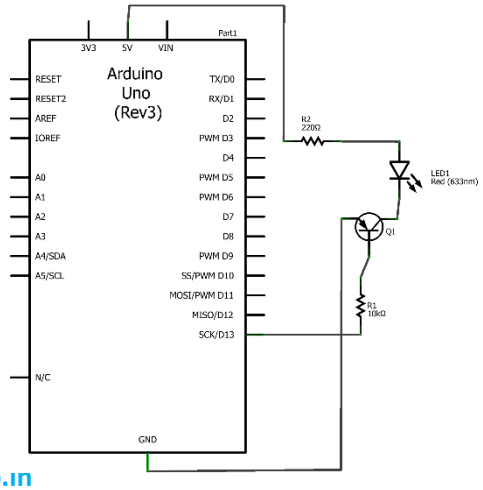
Principle

The **PNP transistor** is a type of bipolar transistor used for amplification and switching purpose and for the designing of the complementary output stage in combination with NPN transistor. It comes with three terminals called emitter, base, and collector where small current at the base terminal is used to control large current at other terminals.
It is a current controlled device also known as *sinking* device where it sinks current into its base terminal and current flows out of the collector.



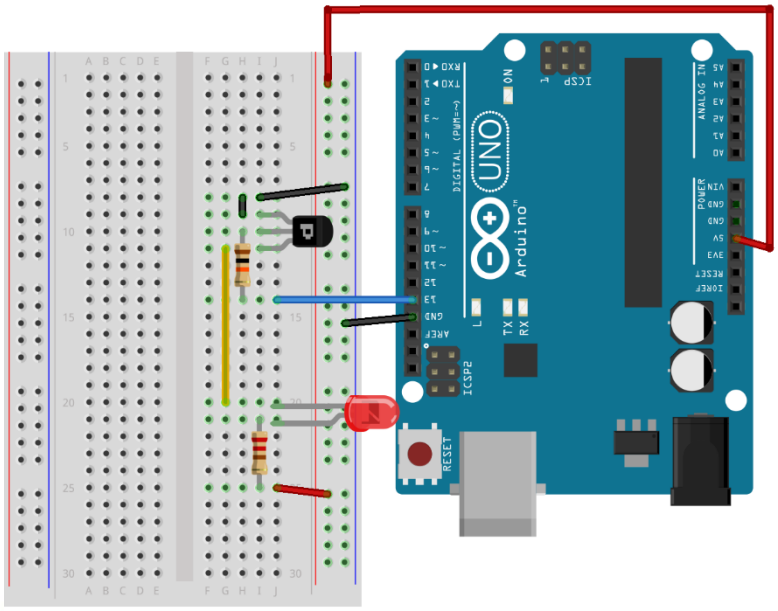
PNP Transistor

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name:   35. PNP transistor .ino Description:
Let, LED blinks using PNP transistor. *********************************/
 void setup()  {
 pinMode(13, OUTPUT);
   }
 void loop() {
 digitalWrite(13, HIGH); // Turn on LED
 delay(5000);
 digitalWrite(13, LOW); // Turn off LED
 delay(5000);
   }
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 36-Spinning DC Motor and Control Speed

Overview:

In this tutorial I will show you how to spinning a motor using the arduino and requires the use of a transistor.

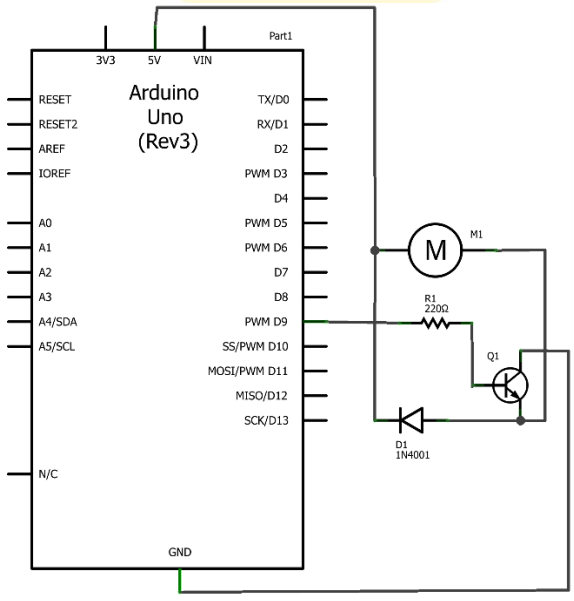
Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x NPN Transistor
- 1 x 220Ω Resistor
- 1 x Diode
- 1 x DC Motor
- 1 x BreadBoard
- Several jumper wires

Principle

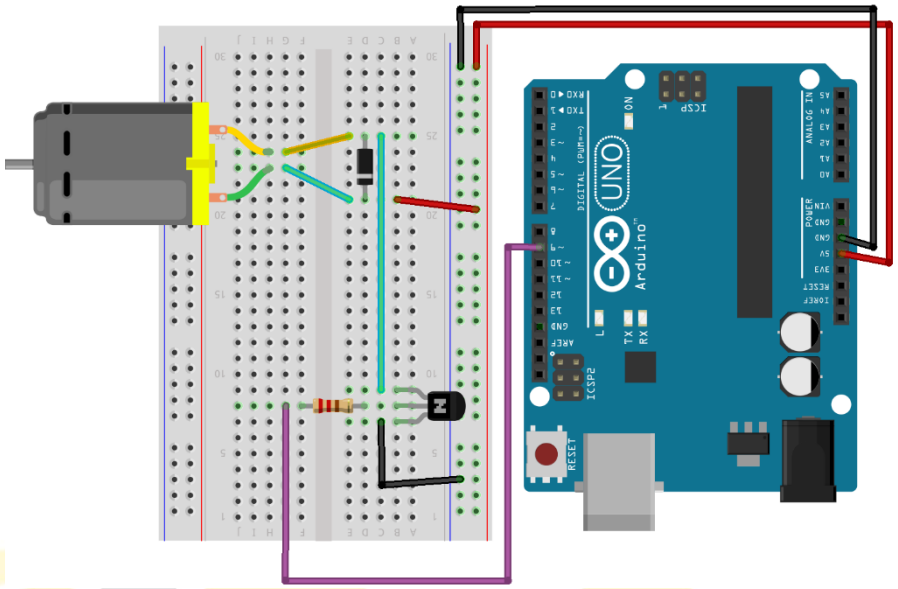
We are going to tackle spinning a motor. This requires the use of a transistor, which can switch a larger amount of current than the Arduino. When using a transistor, you just need to make sure its maximum specs are high enough for your use. The transistor we are using for this circuit is rated at 40V max and 200 milliamps max – perfect for our toy motor. When the motor is spinning and suddenly turned off, the magnetic field inside it collapses, generating a voltage spike. This can damage the transistor. To prevent this, we use a "flyback diode", which diverts the voltage spike around the transistor.

Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*************************************************************
File name: 36 Spinning DC Motor and Control Speed .ino Description:
Let,spinning dc motor and also control speed of the motor. *******************************/

int motorPin = 9;
void setup() {
   pinMode(motorPin, OUTPUT);
   Serial.begin(9600);
}
void loop()
{
  // motorOnThenOff();
  // motorOnThenOffWithSpeed();
  // motorAcceleration();
    serialSpeed();
}
void motorOnThenOff()  {
  int onTime = 3000;  // milliseconds to turn the motor on
  int offTime = 3000; // milliseconds to turn the motor off
  digitalWrite(motorPin, HIGH); // turn the motor on (full speed)
  delay(onTime);              // delay for onTime milliseconds
  digitalWrite(motorPin, LOW);  // turn the motor off
  delay(offTime);             // delay for offTime milliseconds
}
void motorOnThenOffWithSpeed() {
```

```
  int Speed1 = 200;  // between 0 (stopped) and 255 (full speed)
  int Time1 = 3000;  // milliseconds for speed 1
  int Speed2 = 50;   // between 0 (stopped) and 255 (full speed)
  int Time2 = 3000;  // milliseconds to turn the motor off
  analogWrite(motorPin, Speed1);  // turns the motor On
  delay(Time1);                   // delay for onTime milliseconds
  analogWrite(motorPin, Speed2);  // turns the motor Off
  delay(Time2);                   // delay for offTime milliseconds
}
void motorAcceleration() {
  int speed;
  int delayTime = 20; // milliseconds between each speed step
    for(speed = 0; speed <= 255; speed++) {
    analogWrite(motorPin,speed);  // set the new speed
    delay(delayTime);             // delay between speed steps
  }
    for(speed = 255; speed >= 0; speed--) {
    analogWrite(motorPin,speed);  // set the new speed
    delay(delayTime);             // delay between speed steps
  }
}
void serialSpeed() {
  int speed;
  Serial.println("Type a speed (0-255) into the box above,");
  Serial.println("then click [send] or press [return]");
  Serial.println();  // Print a blank line
  while(true)  // "true" is always true, so this will loop forever.
  {
   while (Serial.available() > 0) {
     speed = Serial.parseInt();
     speed = constrain(speed, 0, 255);
     Serial.print("Setting speed to ");
     Serial.println(speed);
     analogWrite(motorPin, speed);
    }
  }
    }
```

Step 3: Compile the program and upload to Arduino UNO board.

Lesson 37 - Arduino IR Obstacle Sensor

Overview:

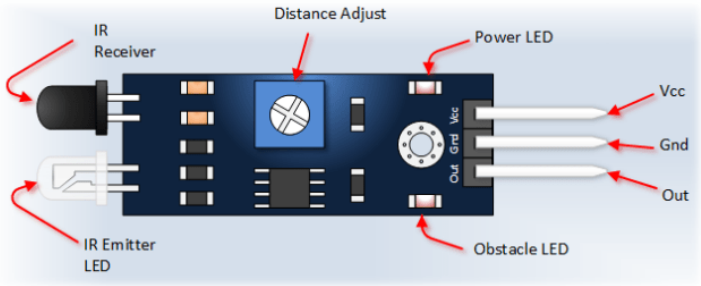
In this tutorial, we are going to use the IR obstacle avoidance sensor with the arduino to cause some actions on an LED.

Components

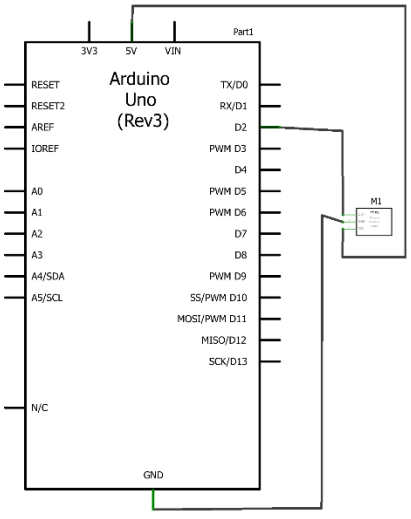
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Infrared sensor
- 1 x BreadBoard
- Several jumper wires

Principle

Infrared Sensors/IR Sensors has two working parts. One IR Transmitter two IR Receiver. IR transmitter is to transmit the infrared waves whereas the work of IR receiver is to receive these infrared waves. IR receiver always sends digital data in the form of 0 or 1 to Vout pin of the sensor.
Here is an object in the front side of the IR sensor, the transmitted infrared waves from the IR sensor transmitter reflects from that object and is received by the IR sensor receiver. IR sensor gives 0 in this condition. Whereas, if there is no object in front of the IR sensor, the transmitted infrared waves from the IR transmitter is not received by the IR receiver.

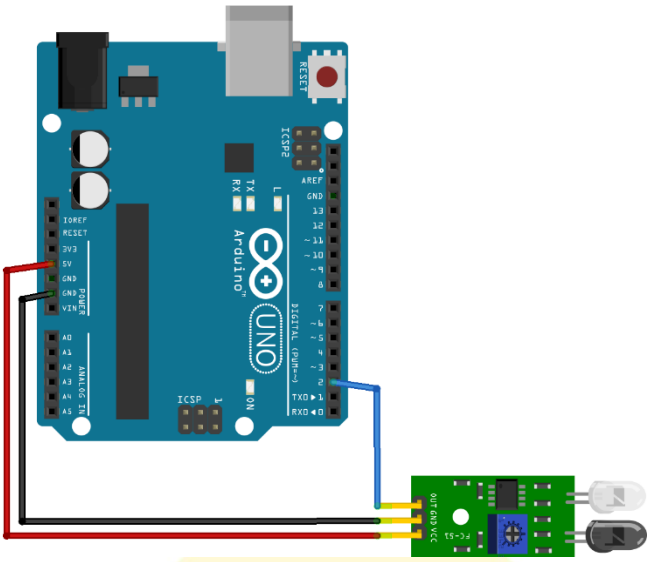


Schematic:



Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/************************************************************
File name: 37 Arduino IR Obstacle Sensor.ino Description:
Let, IR sensor transmitter reflects from that object and is received by the IR sensor receiver.**************/

int LED = 13; // Use the onboard Uno LED
int isObstaclePin = 2;  // This is our input pin
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE
void setup()    {
  pinMode(LED, OUTPUT);
  pinMode(isObstaclePin, INPUT);
  Serial.begin(9600);
  }
void loop() {
  isObstacle = digitalRead(isObstaclePin);
  if (isObstacle == LOW)   {
    Serial.println("OBSTACLE!!, OBSTACLE!!");
    digitalWrite(LED, HIGH);
  }
  else   {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
    }
```

Step 3: Compile the program and upload to Arduino UNO board.

THANK YOU

SUN ROBOTICS

www.sunrobotics.co.in