كلية الهندسة - جامعة المنصورة

Lab

C1118

معمل القياسات و المستشعرات C1118

Computers and Control Systems Engineering Department

قسم هندسة الحاسبات و نظم التحكم

Laboratory Book

Computers and Control Systems Engineering Department

2021

معمل القياسات والمستشعرات C1118

Laboratory Book

Table of Contents

Part 1

1: Laboratory Basic Information

أولا: البيانات الأساسية للمعمل

اسم المعمل	معمل القياسات و المستشعرات (الرقم الكودي: C1118)
القسم العلمي	قسم هندسة الحاسبات و نظم التحكم
مشرفي المعمل	د/ جون فايز د/ محمود سعفان
الهيئة المعاونة	م/ الزهراء عبدالعليم م/ دعاء عرفة م/ هبة سليم
أخصائي المعمل	م/ عبير صلاح الإمام
التليفون	داخلي1305
الموقع بالنسبة للكلية	الناحية البحرية
مساحة المعمل	120 متر مربع

Part 2

2: Laboratory Instruments

ثانيا: قائمة بالأجهزة و المعدات الموجودة في المعمل

محتويات دولاب 1/

م	الكود	الاسم	العدد
		رف 1	
1	4	Raspberry Pi3	10
2	5	HDMI 5" LCD	10
3	6	Arduino –Raspberry Pi Shield Arduino	10
4	7	Raspberry Pi Camera	5
5	67	UTD 2025LUNI 100M HZ (Oscilloscope)	4
		رف 2	
6	20	Liquid Flow Meter Sensor	10
7	34	PIR-TDL9958 Light Control Module	10
8	54	4KG Stepper Motor	10
9	64	PIC Kit ICSP FEPSP5	15
		رف 3	
10	17	PH50 1*6 Female long	100
11	18	PH50 1*8 Female long	100
12	24	Kit HX711 A/D Converter 24 bit	10
13	26	3 Axis Compass and 3 Axis accelerator	10
14	31	Piezo Element SF10293	10
15	37	DN6838 Hall Effect sensor	10
16	42	Rotary Encoder	10
17	45	DS18B20 Temperature sensor	10
18	46	DHT11 Temperature and Humidity Sensor	10
19	58	Press Switch 4 Pins	10
20	59	Press Switch 2 Pins	10
21	60	Micro Switch Big (MS2)	10
22	61	Micro Switch Big (MS3)	10
23	62	Red LED	10
24	66	Function Generator UTG9005C	6
25	68	0804 ADC	10
26	69	0808 DAC	10

10	7805 Voltage Regulator	70	27
10	NE555 IC	71	28
10	Fixed resistor 0.25 W-Mixed values	72	29
10	74HC175 IC	73	30
10	74HC04 IC	74	31
10	74HC08 IC	75	32
10	74HC32 IC	76	33
10	74LS83 IC	77	34
10	4015 IC	78	35
10	74LS90 IC	79	36
10	IRF540 N Channel MOSFET	80	37
10	741 IC	81	38
10	74LS47 IC	82	39
10	74HC76 IC	83	40
10	Crystal OSC (4 MHZ)	84	41
10	Seven Segment common anode	85	42
10	Small LDR	86	43
5	P 6000 1x -10x Oscilloscope Probe	87	44
10	POT Resistor 10K	88	45
10	Jumpers (male _male) short and long	89	46

محتويات دولاب 2/

رف 1			
م	الكود	الاسم	العدد
1	9	Hall Switch Sensor Kit	10
2	25	BMP 180 Pressure Sensor Kit	10
3	30	RTC DS1307 12C kit(Real Time Clock Module)	10
4	32	LDR Module Kit	10
5	38	MPU6050-GY521 IMU Kit	10
6	48	Rotary Motor (toy)	10
7	90	DSPIC30F4013	10
8	91	DSPIC33F64	10
9	92	PIC18F46K20	10
10	93	PIC Kit 3 Programmer	5
11	94	Wi-Fi Module	3
12	95	Bluetooth Module	3
13	96	45 in 1 Tools Box for Arduino	1

رف 2	
Sensor Bag 1	
الكود	الاسم
CS-200	External I/O Board Module
CS-201	Temperature Sensor
CS-202	Gas, Smoke and Ethanol Sensor
CS-203	A/D Hall sensor
CS-204	Thermo couple sensor
CS-205	Photo cell sensor
CS-206	infrared TX-RX sensor
CS-207	Pressure sensor
CS-208	Voltage to frequency converter
CS-209	frequency to voltage converter
CS-210	Pyro Electric thermistor sensor
Sensor Bag 2	
CS-211	Ultrasonic Sensor
CS-212	Photo IR and Photo Interrupter Sensor
CS-213	PT100 Sensor
CS-214	Humidity Sensor
CS-215	Strain Gauge Sensor
CS-216	Magnetic Resistor Sensor

Crystal Temperature Sensor	CS-217
2x Weights Set 5 KG	____
Connectors	____

Content of Tools Box for Arduino	
Name	**Number**
RGB LED(3 Colors)	2
SMD RGB LED	1
Two Color LED	1
Mini Two Color LED	1
7 Color LED	1
Passive Buzzer	1
Active Buzzer	1
DHT Sensor	2
Analog Temperature Sensor	1
Analog/Digital Temperature Sensor	1
Temperature Sensor(DS18B20)	1
Rotary Encoder	1
Heartbeat sensor	2
Knock/Tap Sensor	1
Reed Sensor	1
Mini Reed Sensor	1
Mercury Tilt Switch	1
Ball Switch Module	1
Shock Switch	1
Metal Touch Sensor	1
LDR Module	1
Laser Emitter Module	1
Big Sound Sensor	1
Small Sound Sensor	1
Linear Hall Sensor	1
Digital Hall Sensor	1
Analog Hall Sensor	1
Ultrasonic Sensor	1
Water Level Sensor	1
Soil Moisture Sensor	1
Flame Sensor	2
Light cup module	1
IR Receiver	1
IR Transmitter	1
Obstacle avoidance sensor	1
Line Follow Sensor	1

light blocking Module	1
Joy stick	1
SD CARD Reader	1
Step Down Power Module (MP1584EN)	1
Breadboard power supply module	1

محتويات دولاب 3/

م	الكود	الاسم	العدد
رف 1			
1	11	Arduino Uno Rev.3_China	10
2	12	Arduino SH-L293D	10
3	13	DC Motor Driver 43A-BTS7960	5
4	14	2xLCD +16x Keypad-Arduino Shield	10
5	15	4 Relay Module Kit	10
6	16	L298 Red Board	10
7	19	MQ2 Sensor Kit	10
8	21	Water Level sensor Kit	10
9	22	Water Flow Sensor	10
10	27	Finger Print Module Serial SM12	10
11	28	Color Sensor TCS3200	10
12	29	Tracking Line Digital Kit	10
13	40	Motor Encoder MY37 3PPM	10
14	41	Photo Encoder Kit	10
15	44	NFC PN5832- Blue Kit	10
16	47	DG01D - TT Motor F-Wheel RW002	10
17	49	Micro Servo Motor 180 Degree	10
18	50	L298 Red Board Kit	10
19	51	2WD 2 floor -Ro Base	10
20	53	Pump 12V DC	10
21	57	KP 200 -Keypad 4x4	10
رف 2			
22	8	USB Adapter 5V-2A	10
23	23	Load Cell Kit 20Kg	10
24	33	Flame Sensor Kit	10

10	Ultrasonic Sensor 2cm-4m	35	25
10	Sound Sensor Digital Out mode	36	26
10	Door Magnetic Sensor	39	27
10	Current Sensor 30A DC	43	28
60	BB0 Bread Board	63	29
30	Digital Multimeter-Model 9205	65	30
10	Arduino Mega 2560- Development Kit	10	31
10	OW007-Small Wheel	52	32
30	Soldering Iron 30Watt	55	33
30	Small disordering pump	56	34

Part 3

3: Laboratory Student Beneficiaries and Activities

ثالثا: الخدمات الطلابية التي يؤديها المعمل و الأنشطة

الأقسام العلمية المستفيدة من المعمل	قسم هندسة الحاسبات و نظم التحكم برنامج هندسة الحاسبات و الاتصالات برنامج الهندسة الطبية الحيوية برنامج الميكاترونكس
المقررات الدراسية التي تستفيد من المعمل	مقرر قياسات و مستشعرات (الفرقة الثانية) مقرر مستشعرات و مؤثرات (برنامج الهندسة الطبية و برنامج ميكاترونكس) مقرر نظم تحكم بالحاسبات 1(الفرقة الثالثة) مقرر نظم تحكم بالحاسبات 2(الفرقة الرابعة) مقرر نمذجة و محاكاة النظم (الفرقة الثانية)
الأنشطة الطلابية داخل المعمل	مشاريع القسم لفرق القسم في المواد الدراسية مشاريع التخرج للفرقة الرابعة مشاريع الفرق العلمية و التعليمية
المسابقات العملية التي شارك فيها طلاب من المستفيدين من المعمل	Roborace, Minesweeper, Somo robot

Part 4

4: Laboratory Experimental

رابعا: التجارب المعملية

Arduino Board :

Arduino is an open source physical computing platform based on a simple input/output (I/O) board and a development environment that implements the Processing language. Arduino can be used to develop standalone interactive objects or can be connected to software on your computer.

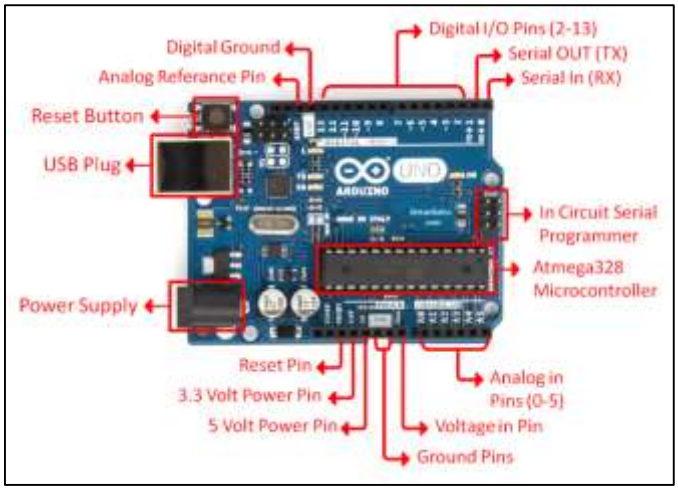


Fig 1.ARDUINO UNO

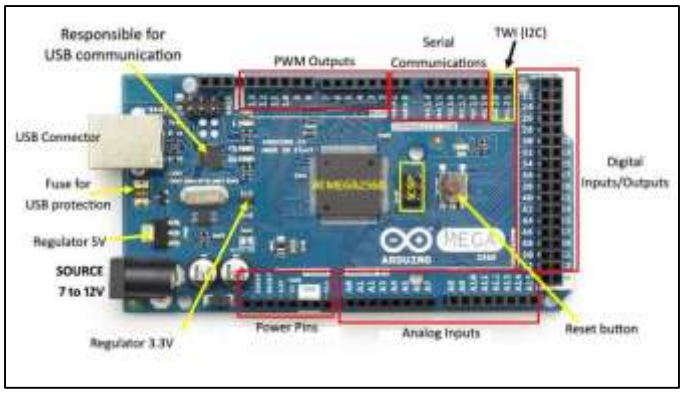


Fig 2. ARDUINO MEGE

التجارب التالية تطبق باستخدام بوردة اردوينو Arduino Uno (code: 11) أو Arduino Mega 2560- Development Kit (code: 10) مع استخدام البوكس المحتوي على 45 مستشعر 45 Sensors for ARDUINO Box.

Experiment No.	Name
1	RGB LED
2	Two Color LED
3	7 Color LED
4	Active/Passive Buzzer Module
5	Digital Temperature and Humidity Sensor
6	Analog/Digital Temperature Sensor
7	Temperature Sensor(DS18B20)
8	Rotary encoder module
9	Heartbeat Sensor
10	Knock/Tap Sensor module
11	Reed and Mini Reed Switch Module
12	Mercury Tilt Switch Module
13	Ball Switch Module
14	Metal Touch Sensor
15	Shock Switch
16	LDR Module
17	Laser transmitter Module
18	Sound Sensor
19	Ultrasonic Sensor
20	Water Level Sensor
21	Soil Moisture Sensor
22	Hall Sensor
23	Obstacle avoidance sensor module
24	Magic Light Cup Module
25	Flame Sensor Module
26	XY-axis Joystick Module
27	Line Follower Sensor Module
28	IR Transmitter Module and 38 KHz IR Receiver Module
29	SD Card Reader
30	Photo Interrupter Module
31	Step Down Power Module (mp1584EN)

Experiment 1. RGB LED:

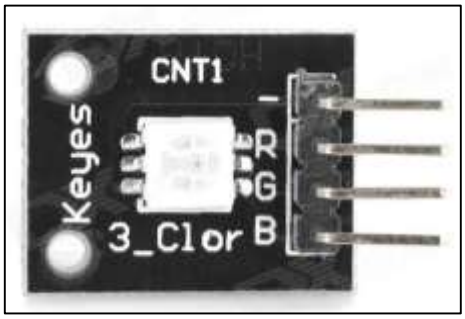


Fig 3. SMD RGB LED Module

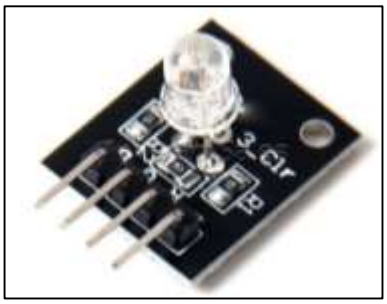
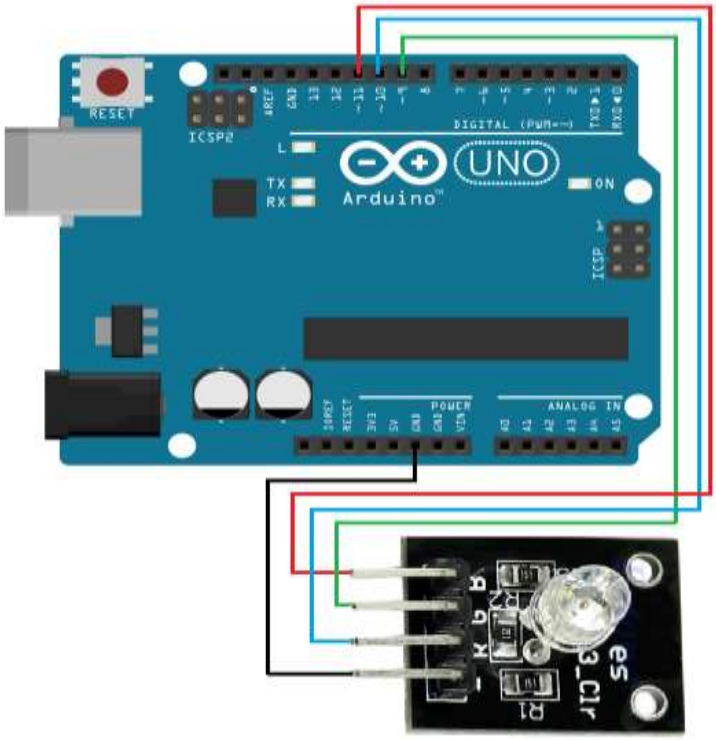


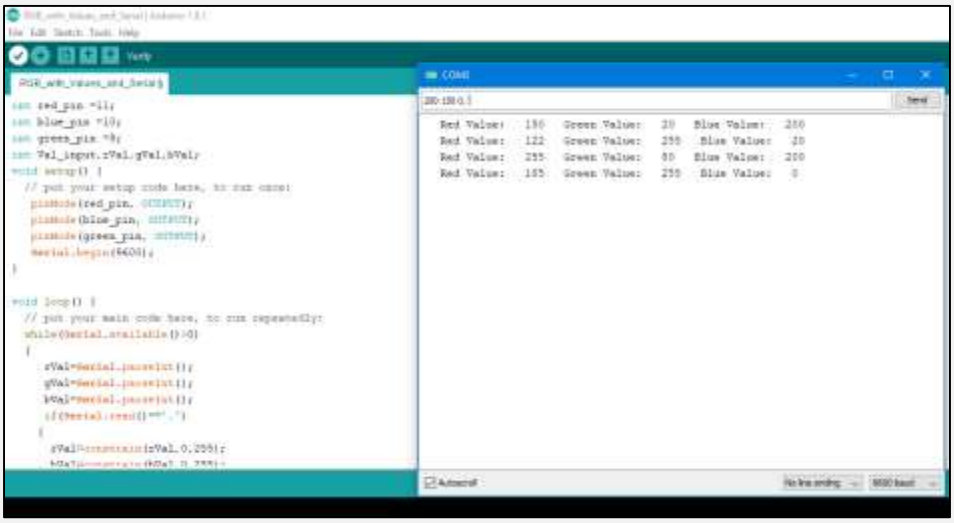
Fig 4. RGB LED Module




```
int red_pin =11;
int blue_pin =10;
int green_pin =9;
int Val_input,rVal,gVal,bVal;
void setup() {
// put your setup code here, to run once:
pinMode(red_pin, OUTPUT);
pinMode(blue_pin, OUTPUT);
pinMode(green_pin, OUTPUT);
```

```
Serial.begin(9600);
}
void loop() {
// put your main code here, to run repeatedly:
while(Serial.available()>0)
{
rVal=Serial.parseInt();
gVal=Serial.parseInt();
bVal=Serial.parseInt();
if(Serial.read()=='.')
{
rVal=constrain(rVal,0,255);
bVal=constrain(bVal,0,255);
gVal=constrain(gVal,0,255);
analogWrite(red_pin,rVal);
analogWrite(green_pin,gVal);
analogWrite(blue_pin,bVal);
Serial.print("   Red Value:   ");
Serial.print(rVal);
Serial.print("   Green Value:   ");
Serial.print(gVal);
Serial.print("   Blue Value:   ");
Serial.print(bVal);
Serial.println();
}
}
```

}


Red Value: 130 Green Value: 20 Blue Value: 200
Red Value: 122 Green Value: 255 Blue Value: 20
Red Value: 255 Green Value: 80 Blue Value: 200
Red Value: 165 Green Value: 255 Blue Value: 0

Experiment 2. Two Color LED(RG LED):

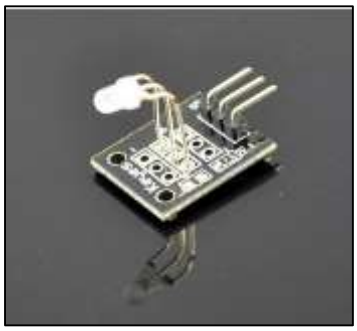


Fig 5. Two Color LED Module (small)

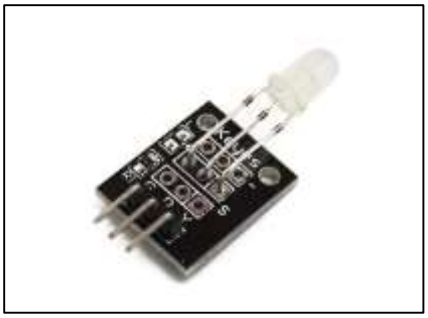
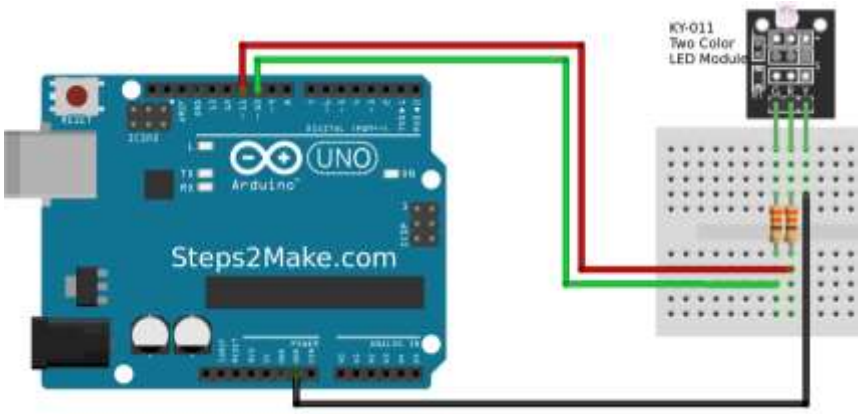


Fig 6. Two Color LED Module.




```
int red_pin =11;
int green_pin =10;
int val;
void setup()
{  pinMode(red_pin, OUTPUT);
   pinMode(green_pin, OUTPUT); }
void loop() {
  for (val=255;val>0;val--)
  { analogWrite(green_pin,val);
    analogWrite(red_pin,255-val);
    delay(20); }
  delay(1000);
  for (val=0;val<255;val++)
  {  analogWrite(green_pin,val);
     analogWrite(red_pin,255-val);
     delay(20);}
  delay(1000);}
```

Experiment 3. 7 Color LED:

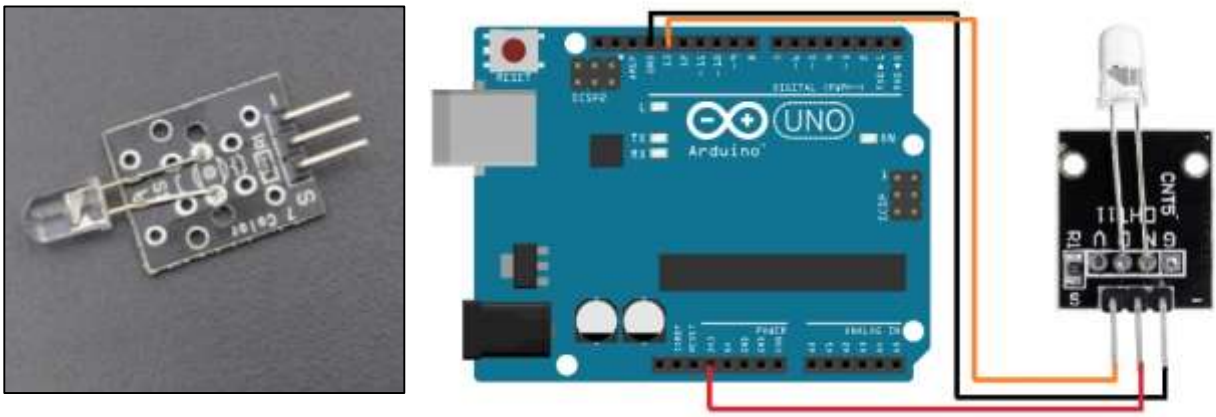


Fig 7. 7 Color LED Module.

```
void setup ()
{
  // Initialize the digital pin as an output.
  pinMode (13, OUTPUT);
}

void loop ()
{
  digitalWrite (13, HIGH); // set the LED on
  delay (2000); // wait for a second
  digitalWrite (13, LOW); // set the LED off
  delay (2000); // wait for a second
}
```

Experiment 4. Active/Passive Buzzer Module:

Active Buzzer Module produces a single-tone sound when signal is high. To produce different tones use the passive buzzer module.

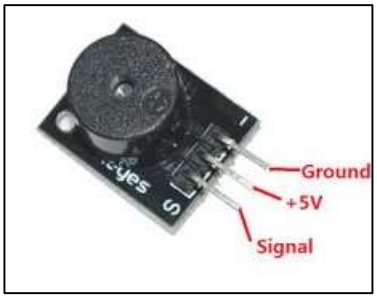
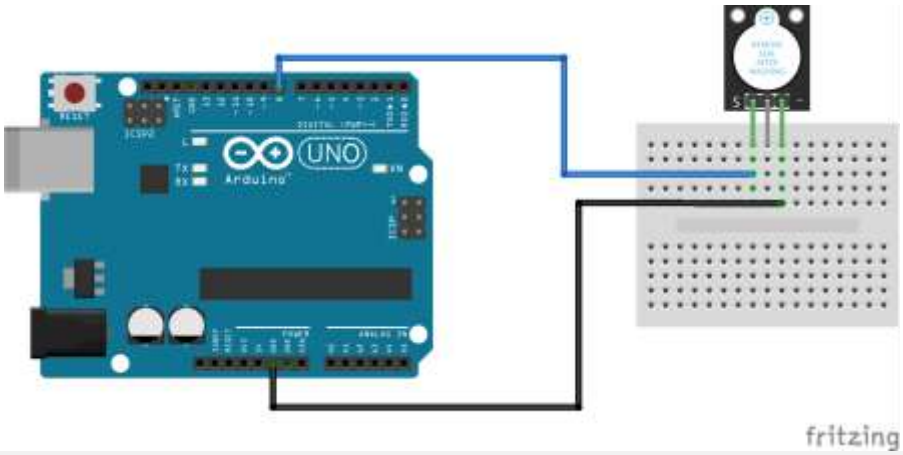


Fig 8. Active Buzzer Module.

Fig 9.Passive Buzzer Module.




```
// using Active Buzzer
int buzzerPin = 8;
void setup()
{  pinMode (buzzerPin, OUTPUT); }
void loop()
{ digitalWrite (buzzerPin, HIGH);
  delay(500);
  digitalWrite (buzzerPin, LOW);
  delay(500);  }
```

```
//using Passive buzzer
const int buzzer = 8; //buzzer to arduino pin 9
void setup()
{
  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output
}
void loop()
{ tone(buzzer,1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(1000);  }      // ...for 1sec
```

Experiment 5. Digital Temprature and Humidity Sensor:

The DHT11 humidity and temperature sensor makes it really easy to add humidity and temperature data to projects. It's perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.
Before you can use the DHT11 on the Arduino, you'll need to install the **DHTLib** library. It has all the functions needed to get the humidity and temperature readings from the sensor.

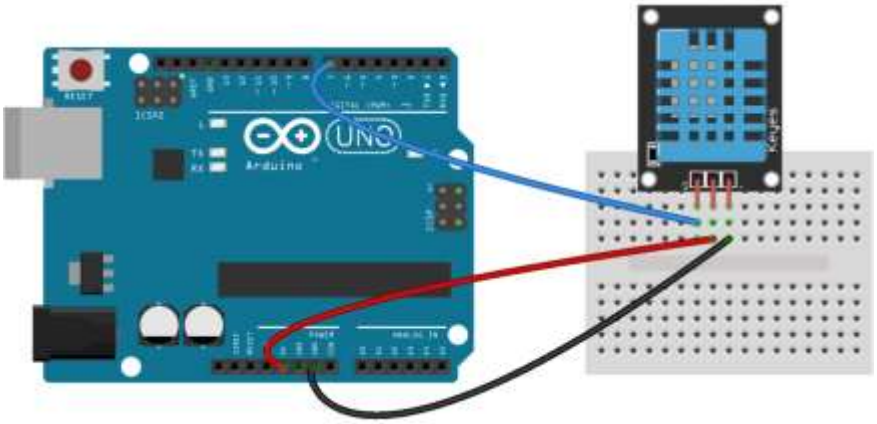


Fig 10. DHT Sensor

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 7
void setup(){
Serial.begin(9600);
}

void loop()
{
int chk = DHT.read11(DHT11_PIN);
Serial.print("Temperature = ");
Serial.println(DHT.temperature);
Serial.print("Humidity = ");
Serial.println(DHT.humidity);
delay(1000);}
```

Experiment 6. Analog/Digital Temperature Sensor:

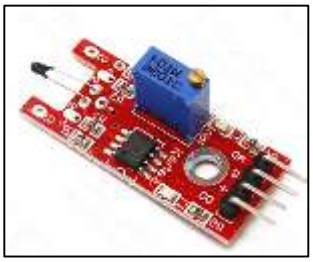
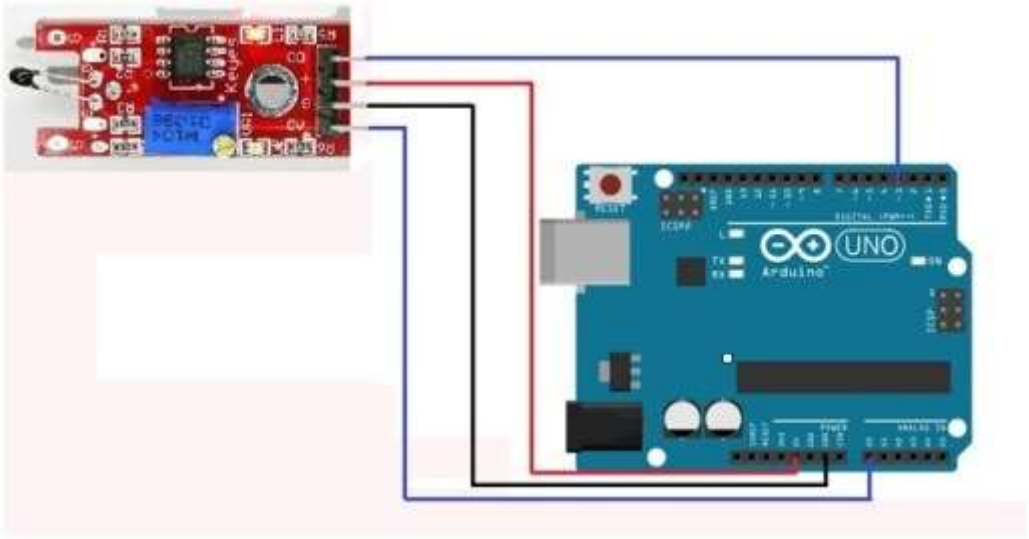


Fig 11.Analog/ Digital Temperature Sensor

Fig 12. Analog Temperature Sensor

The analog temperature sensor is based on the thermistor (resistance increases with the ambient temperature changes), sending the data to the analog IO, thereby widely used in gardening, home alarm systems and other devices. The temperature Range of sensor is from -55°C to +125°C. Another temperature sensor module is used also to measure the temperature and it gives output both at *analog* and *digital* pin. This module has many components like thermistor, 100k ohm potentiometer, and lm393 comparator. Potentiometer is basically used for the digital output. The digital output can be changed by changing the value of potentiometer.



```
#include <math.h>
#define LED_PIN 13
#define DIGITAL_INPUT 3
#define ANALOG_INPUT   A0
int digital_output ;// This will read the digital value
int analog_output ;// This will read the analog value
int revised_output;// variable to store the corrected value
float temp_C ;// Variable for storing the temperature
float temp_f ;// Variable for storing the Fahrenheit

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

```
  pinMode ( DIGITAL_INPUT, INPUT ) ;
  pinMode ( ANALOG_INPUT, INPUT )  ;
  Serial.begin ( 9600 ) ;
  Serial.println ("The data is:  " ) ;
}

void loop()
{ analog_output = analogRead ( ANALOG_INPUT ) ;
  Serial.print ( " Analog value of the module is =  " ) ;
  Serial.print ( analog_output ) ,  DEC ;
  // The module has thermistor connection reversed
  revised_output= map ( analog_output, 0, 1023, 1023, 0 ) ;
  temp_C    = Thermistor ( revised_output ) ;
  temp_f =( temp_C * 9.0 ) / 5.0 + 32.0 ;
  // Reading the digital data
  digital_output = digitalRead ( DIGITAL_INPUT ) ;
  Serial.print( "   Digital value of the module is =  " ) ;
  Serial.println ( digital_output ) , DEC ;
  Serial.print( " LED is =" ) ;
  if ( digital_output == HIGH )
  // The LED will turn on When the sensor value will exceed the set point
  { digitalWrite ( LED_PIN, HIGH ) ;
   Serial.print( "ON " ) ; }
  else
  { digitalWrite( LED_PIN, LOW ) ;
    Serial.print( "OFF " ) ;   }
    Serial.print( " Measured Temperature = " ) ;
    Serial.print( temp_f, 1 ) ;//display the temperature in Fahrenheit
    Serial.print(" F  " ) ;
    Serial.print(temp_C, 1 );// display the temperature in Celsius
    Serial.println(" C " ) ;
    Serial.println() ;
    delay(1000) ; }
double Thermistor ( int RawADC )
{
double Temp ;
Temp = log(((10240000/RawADC)-10000));
Temp = 1 / ( 0.001129148 + ( 0.000234125 * Temp ) + ( 0.0000000876741 *
Temp * Temp * Temp ) ) ;
Temp = Temp - 273.15 ;            // This will Convert Kelvin to Celsius
return Temp ;
}
```

Experiment 7. Temprature Sensor(DS18B20):

These 3-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog converter. They work great with any microcontroller using a single digital pin, and you can even connect multiple ones to the same pin, each one has a unique 64-bit ID burned in at the factory to differentiate them. Usable with 3.0-5.0V systems. Usable temperature range: -5° to 90°C (-23°F to +194°F)

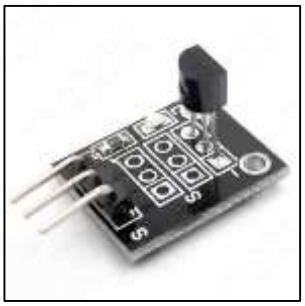
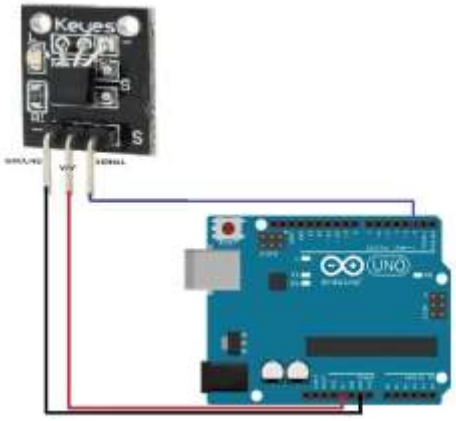


Fig 13. DS18B20 Temperature Sensor

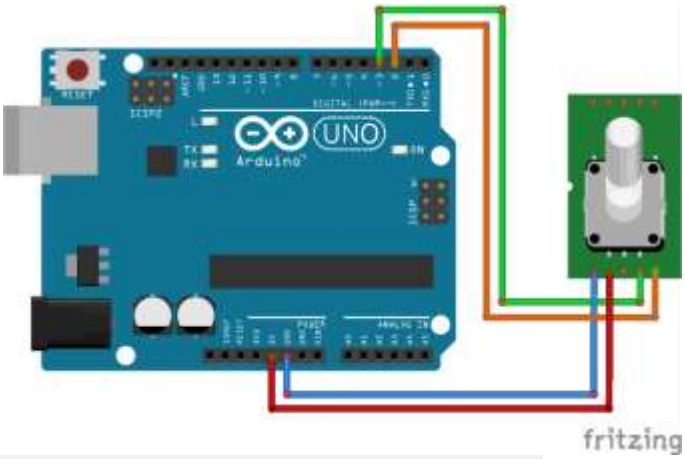


```
#include <OneWire.h>
#include <DallasTemperature.h>
// Data wire is plugged into digital pin 2 on the Arduino
#define ONE_WIRE_BUS 2
// Setup a oneWire instance to communicate with any OneWire device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library
DallasTemperature sensors(&oneWire);
void setup(void)
{
  sensors.begin();  // Start up the library
  Serial.begin(9600);
}
void loop(void)
{
  // Send the command to get temperatures
  sensors.requestTemperatures();
  //print the temperature in Celsius
  Serial.print("Temperature: ");
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print((char)176);//shows degrees character
  Serial.print("C  |  ");
  //print the temperature in Fahrenheit
  Serial.print((sensors.getTempCByIndex(0) * 9.0) / 5.0 + 32.0);
  Serial.print((char)176);//shows degrees character
  Serial.println("F");
  delay(500); }
```

Experiment 8. Rotary encoder module:

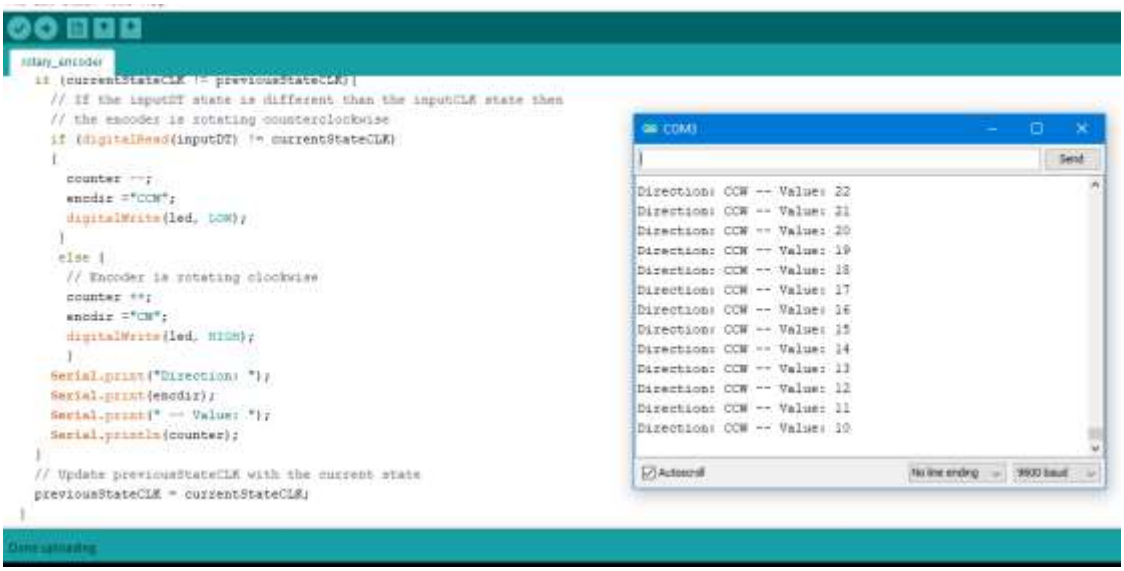
By rotating the rotary encoder can be counted in the positive direction and the reverse direction during rotation of the output pulse frequency. That starts counting from 0.

Fig 14. .Rotary Encoder.




```
#define inputCLK 2
#define inputDT 3
#define led 13

int counter = 0;
int currentStateCLK;
int previousStateCLK;

String encdir ="";

void setup() {
  // Set encoder pins as inputs
  pinMode (inputCLK,INPUT);
  pinMode (inputDT,INPUT);
  pinMode (led,OUTPUT);
  Serial.begin (9600);
  // Read the initial state of inputCLK and assign to previousStateCLK
  previousStateCLK = digitalRead(inputCLK);
}

void loop() {
 // Read the current state of inputCLK
  currentStateCLK = digitalRead(inputCLK);
  // If the previous and the current state of the inputCLK are different
then a pulse has occured
  if (currentStateCLK != previousStateCLK){
    // If the inputDT state is different than the inputCLK state then
    // the encoder is rotating counterclockwise
    if (digitalRead(inputDT) != currentStateCLK)
    {
      counter --;
```

```
      encdir ="CCW";
      digitalWrite(led, LOW);
     }
     else {
      // Encoder is rotating clockwise
      counter ++;
      encdir ="CW";
      digitalWrite(led, HIGH);
      }
    Serial.print("Direction: ");
    Serial.print(encdir);
    Serial.print(" -- Value: ");
    Serial.println(counter);
  }
  // Update previousStateCLK with the current state
  previousStateCLK = currentStateCLK;
}
```



Experiment 9. Heartbeat Sensor:

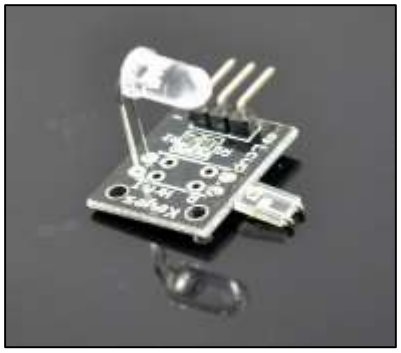


Fig 15. Heartbeat sensor

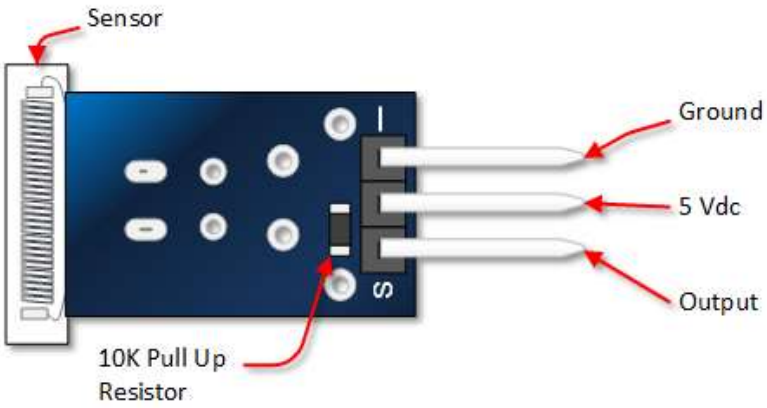
This sensor module is used to detect the pulse of the finger using the bright infrared (IR) LED and a photo resistor. Pulse monitor works as follows: The LED is the light side of the finger, and phototransistor on the other side of the finger, phototransistor used to obtain the flux emitted, when the blood pressure pulse by the finger when the resistance of the photo transistor will be slightly changed. The project's schematic circuit as shown, we chose a very high resistance resistor R1, because most of the light through the finger is absorbed, it is desirable that the phototransistor is sensitive enough. Resistance can be selected by experiment to get the best results. The most important is to keep the shield stray light into the phototransistor. For home lighting that is particularly important because the lights at home mostly based 50HZ or 60HZ fluctuate, so faint heartbeat will add considerable noise.

```
int sensorPin = 0;
double alpha = 0.75;
int period = 100;
double change = 0.0;
double minval = 0.0;
void setup ()
{
  Serial.begin (9600);
}
void loop ()
{
    static double oldValue = 0;
    static double oldChange = 0;
    int rawValue = analogRead (sensorPin);
    double value = alpha * oldValue + (1 - alpha) * rawValue;
    Serial.print (rawValue);
    Serial.print (",");
    Serial.println (value);
    oldValue = value;
    delay (period);
}
```

Experiment 10. Knock/Tap Sensor module:

The knock sensor, detects the knocks and the taps. It can work like a switch. The sensor sends data momentarily to the board.

Fig 16. Knock/tap Sensor




```
int Led = 13 ;
int Knock = 3 ;
int val ;
void setup ()
{
  pinMode (Led, OUTPUT) ;
  pinMode (Knock, INPUT) ;
}
void loop ()
{
  val = digitalRead (Knock) ;
  if (val == HIGH) // When the percussion when the sensor detects a
signal, LED ON
  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }}
```

Experiment 11. Reed and Mini Reed Switch Module:

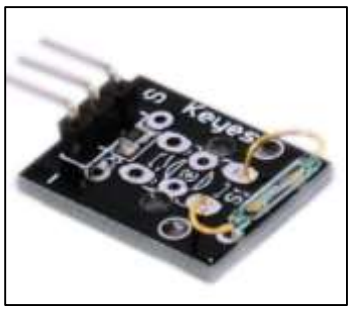
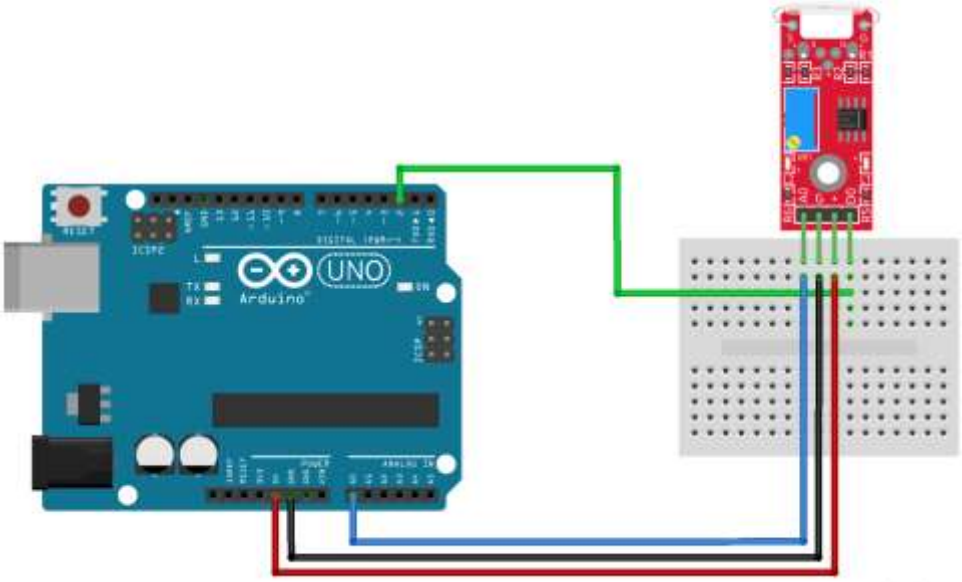


Fig 17. Mini Reed Switch

Fig 18. Reed Switch

A reed switch is a switch that needs a magnet in front of it to switch on or off. This works the same for the Arduino module.



```
int Led = 13 ;
int reedSw = 2;
int val ;
void setup ()
{
 pinMode (Led, OUTPUT) ;
 pinMode (reedSw, INPUT) ;
}
void loop ()
{ val = digitalRead (reedSw) ;
 if (val == HIGH)
 { digitalWrite (Led, HIGH); }
  else
{ digitalWrite (Led, LOW);}
}
```

Experiment 12. Mercury Tilt Switch Module:

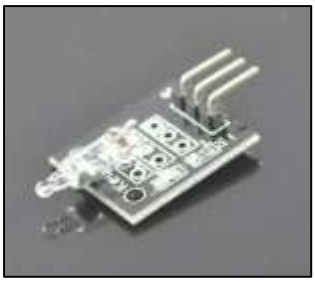
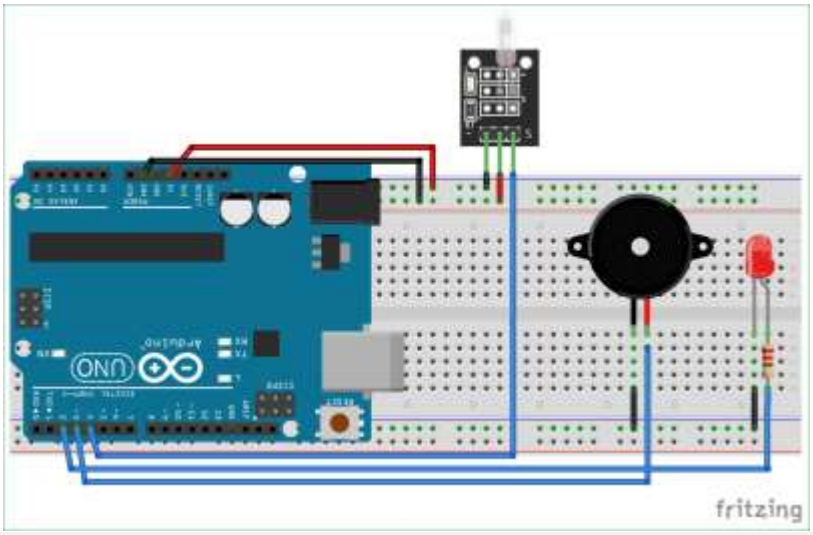


Fig 19. Mercury Tilt Module.

A Tilt Sensor switch is an electronic device that detects the orientation of an object and gives its output High or Low accordingly. Basically, it has a mercury ball inside it which moves and makes the circuit. So tilt sensor can turn on or off the circuit based on the orientation.




```
int mercurySw=4;
int buzzer=3,led=2;
void setup()
{ pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(mercurySw, INPUT); }
void loop()
{if (digitalRead(mercurySw) == 1)
 { digitalWrite(led, HIGH);
   digitalWrite(buzzer, HIGH);
   delay(300);
   digitalWrite(led, LOW);
   digitalWrite(buzzer, LOW);
   delay(300); }}
```

Experiment 13. Ball Switch Module:

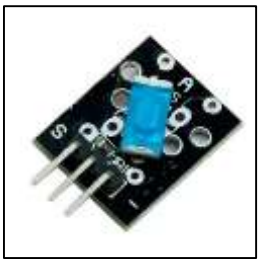
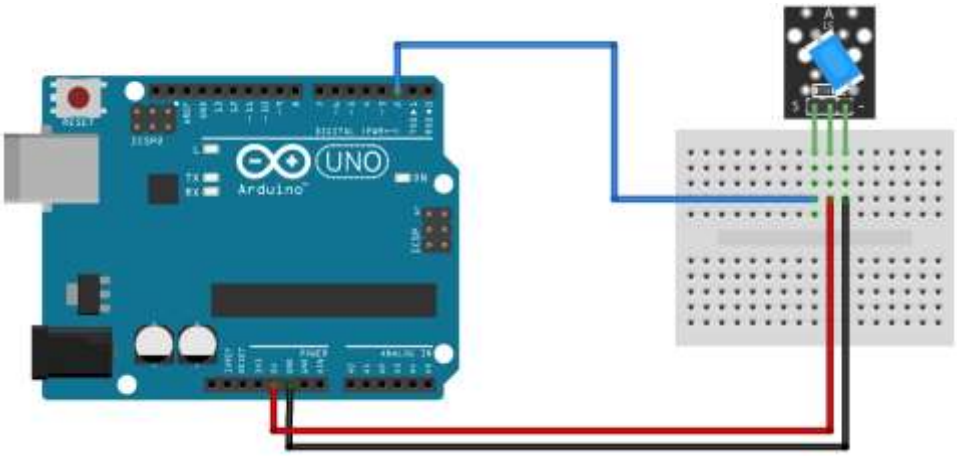


Fig 20. Ball Switch Module.

This sensor contains a small metal ball which will complete a circuit depending on the position in the sensor. Because the sensor is very basic, it can only detect large changes when its tilt, and cannot measure the angle of its tilt.




```
int Led = 13 ;//built-in Led
int ball_Sw = 2;
int val;
void setup ()
{
  pinMode (Led, OUTPUT)
    pinMode (ball_Sw, INPUT);
}
void loop ()
{
  val = digitalRead (ball_Sw)
  if (val == HIGH)  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }
}
```

Experiment 14. Metal Touch Sensor:

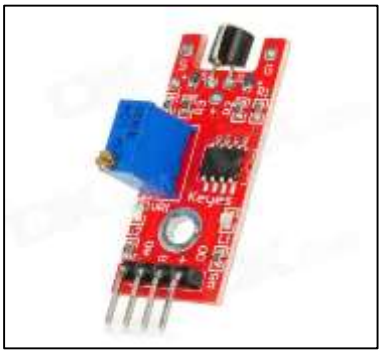
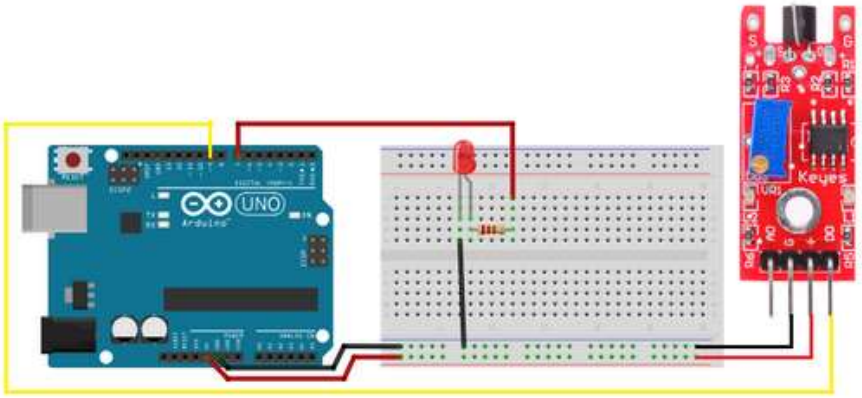


Fig 21.Touch Sensor.

Metal touch sensor is a type of switch that triggers whenever the metal spike of the sensor is touched by a conducting body like our body. This module has 4 header pins, these are allocated for the +5V, GND, D0 / (Digital Output), A0 / (Analog Output).

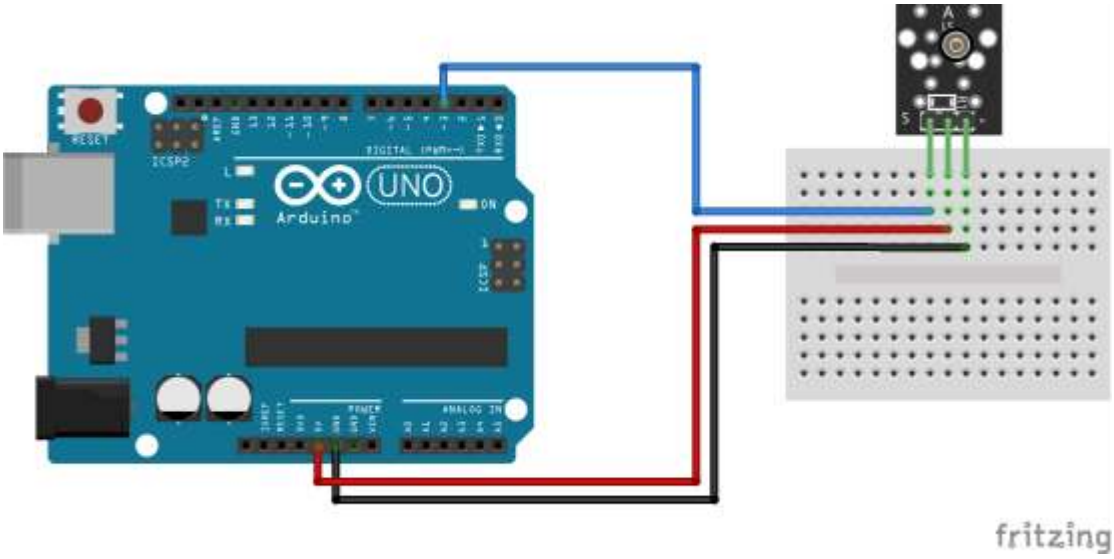


```
int ledpin = 7 ; // sets the LED @pin 7
int touchpin = 9; // sets the KY-036 metal touch sensor @pin 9
int value ;          // defines the numeric variables as value
void setup ()
{
  pinMode (touchpin, INPUT) ; // sets the metal touch sensor as INPUT
  pinMode (ledpin, OUTPUT) ; // sets LED as the OUTPUT
}
void loop ()
{
  value = digitalRead (touchpin) ; // reads the value of the touchpin
  if (value == HIGH)       // If the value is HIGH
  {
    digitalWrite (ledpin, HIGH);   // It will turn the LED ON, indicating
that the sensor has been triggered
  }
  else        //otherwise
  {
    digitalWrite (ledpin, LOW);  // LED is turned off if sensor is not
triggered  }}
```

Experiment 15. Shock Switch:

Fig 21.Shock Sensor.

The switch primarily consists of a terminal that forms a center post and a second terminal that is a spring that surrounds the center post. When a sufficient force is transferred to the switch, the terminal consisting of the spring moves and shorts both terminals together. The connection between the terminals is momentary and will require a little thought as you implement it in your Arduino project. Positioning of the switch is also important.




```
int shockPin = 3;
int shockVal = HIGH; // This is where we record our shock measurement
boolean bAlarm = false;
unsigned long lastShockTime; // Record the time that we measured a shock
int shockAlarmTime = 250; // Number of milli seconds to keep the shock alarm high
void setup ()
{
  Serial.begin(9600);
  pinMode (shockPin, INPUT);}

void loop ()
```

```
{
  shockVal = digitalRead (shockPin) ; // read the value from our sensor
  if (shockVal == LOW) // If we're in an alarm state
  {
    lastShockTime = millis(); // record the time of the shock
    // The following is so you don't scroll on the output screen
    if (!bAlarm){
      Serial.println("Shock Alarm");
      bAlarm = true;
    }
  }
  else
  {
    if( (millis()-lastShockTime) > shockAlarmTime  &&  bAlarm){
      Serial.println("no alarm");
      bAlarm = false;
    }
  }
}
```

Experiment 16. LDR Module:

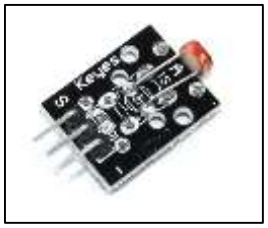


Fig 22. LDR Module.

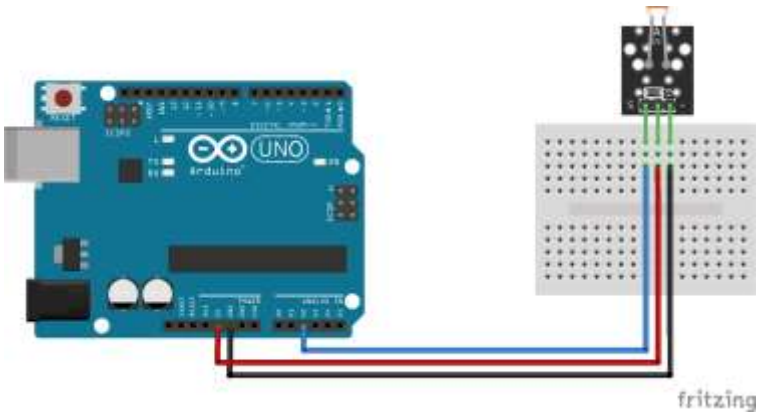


Photo resistors, also known as light dependent resistors (LDR), are light sensitive devices most often used to indicate the presence or absence of light, or to measure the light intensity. In the dark, their resistance is very high, sometimes up to 1MΩ, but when the LDR sensor is exposed to light, the resistance drops dramatically, even down to a few ohms, depending on the light intensity. LDRs have a sensitivity that varies with the wavelength of the light applied and are nonlinear devices. They are used in many applications but are sometimes made obsolete by other devices such as photodiodes and phototransistors. Some countries have banned LDRs made of lead or cadmium over environmental safety concerns.

```
const int photocellPin = A2;
int photocellReading;
void setup(void)
{ Serial.begin(9600); }
void loop(void)
{
photocellReading = analogRead(photocellPin);
Serial.print("Analog reading = ");
Serial.print(photocellReading);
if (photocellReading < 20) { Serial.println(" - Dark");}
else if (photocellReading < 100) { Serial.println(" - Dim");}
else if (photocellReading < 400) {Serial.println(" - Light"); }
else if (photocellReading < 600) { Serial.println(" - Bright"); }
else {Serial.println(" - Very bright"); }
delay(1000);
}
```

Experiment 17. Laser transmitter Module:

Laser transmitter module, 650 nm (red), gives a small intense beam. Take care of your eyes, do not look direct into the beam. Power consumption: 30 mA at 5 V

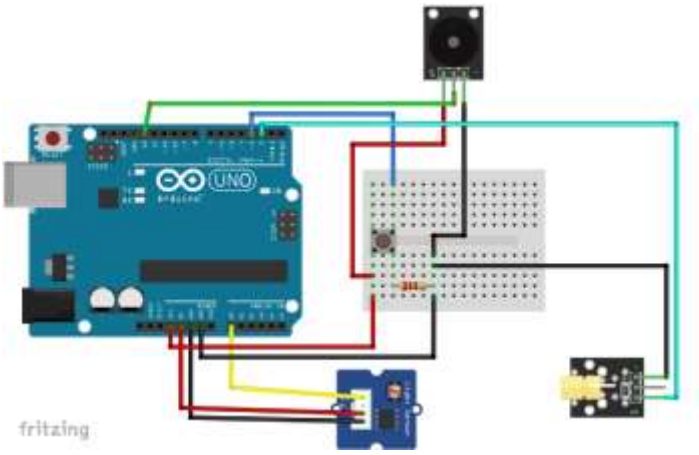


Fig 23. Laser transmitter Module.

```
#define Rec 0      //LDR output
#define Laser 2    //Laser module
#define Button 3   //Push button input
bool detection;
void setup() {
  pinMode(Laser, OUTPUT);
  digitalWrite(Laser, HIGH); //Turning on the laser
  delay(2000);
}
void loop() {
short Detect = analogRead(Rec); //Constantly reading the module value
bool  Button_state = digitalRead(Button);//And the button value (1-0)
//The Max value is 760, if someone passes it goes below that (every value
lower than 700 can do the work)
if(Detect < 500)
{ detection = true;}
 if(detection==true)
{ tone(13,2000);//Alarm sequence will go on as long as the detection is
true
  delay(50);//This alarm has two sounds 2kHz and 1Khz delayed by 50ms
```

```
  tone(13,1000);
  delay(50);}
if(Button_state == HIGH)  //If the button is pressed the buzzer is turned
off and the detection too
{     detection = false;
      noTone(13);
 }
}
```

Experiment 18. Sound Sensor:

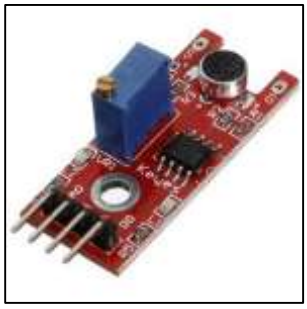


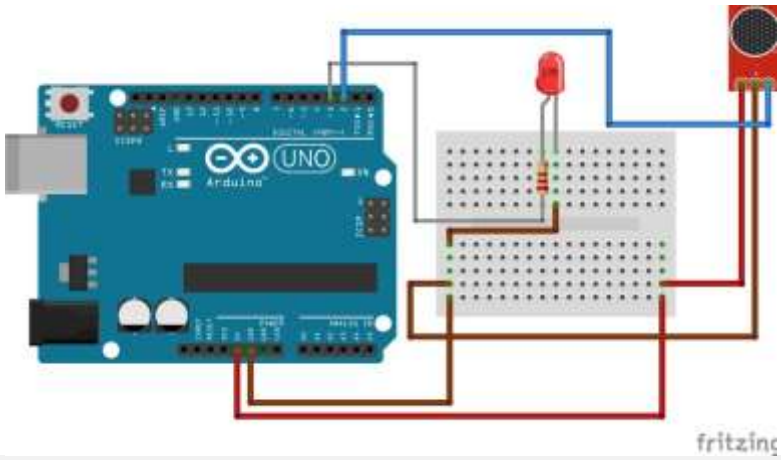
Fig 24. Small Sound Sensor.

Fig 25. Big Sound Sensor.

For sound detection Module has two outputs:

1. AO, analog output, real-time output voltage signal of the microphone.
2. DO, when the sound intensity reaches a certain threshold, the output high and low signal.

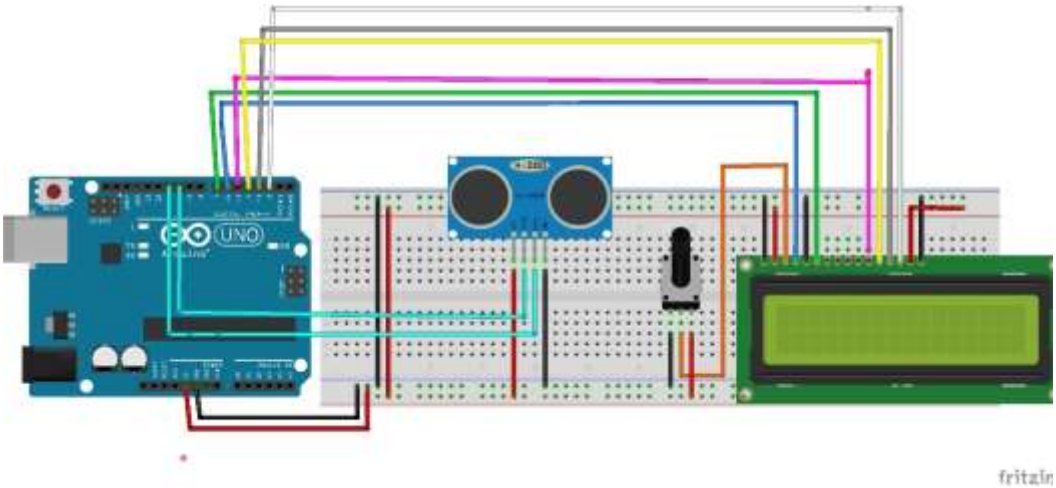
The threshold-sensitivity can be adjusted via potentiometer on the sensor




```
int soundSensor=2;
int LED=3;
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);}}}
```

Experiment 19. Ultrasonic Sensor:

Fig 26. Ultrasonic Sensor.

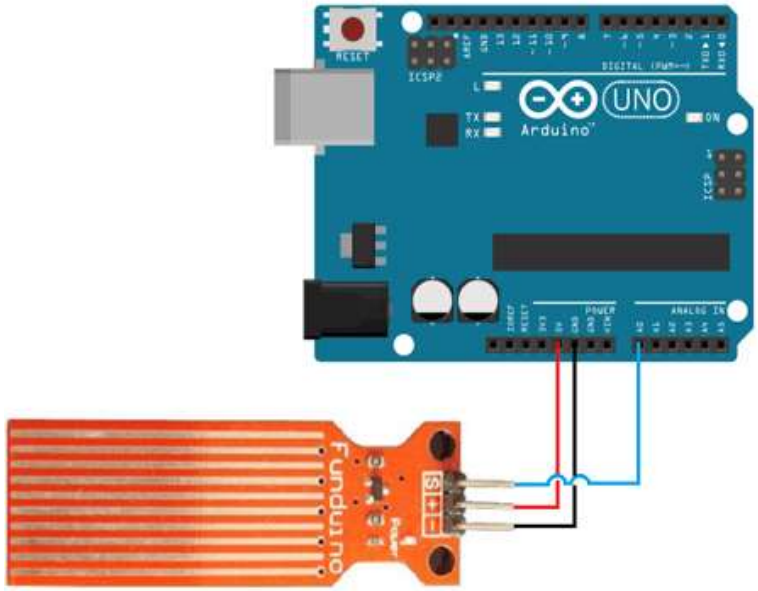



```
#include <LiquidCrystal.h> // includes the LiquidCrystal Library
LiquidCrystal lcd(6, 7, 5, 4, 3, 2); // Creates an LCD object.
Parameters: (rs, enable, d4, d5, d6, d7)
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;
void setup()
{
lcd.begin(16,2); // Initializes the interface to the LCD screen, and
specifies the dimensions (width and height) of the display
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(53,OUTPUT); //back light for lcd
digitalWrite(53,HIGH);
}
void loop()
```

```
{
delay(500);
lcd.clear();
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
lcd.setCursor(0,0); // Sets the location at which subsequent text written
to the LCD will be displayed
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distance); // Prints the distance value from the sensor
lcd.print(" cm");
delay(10);
}
```

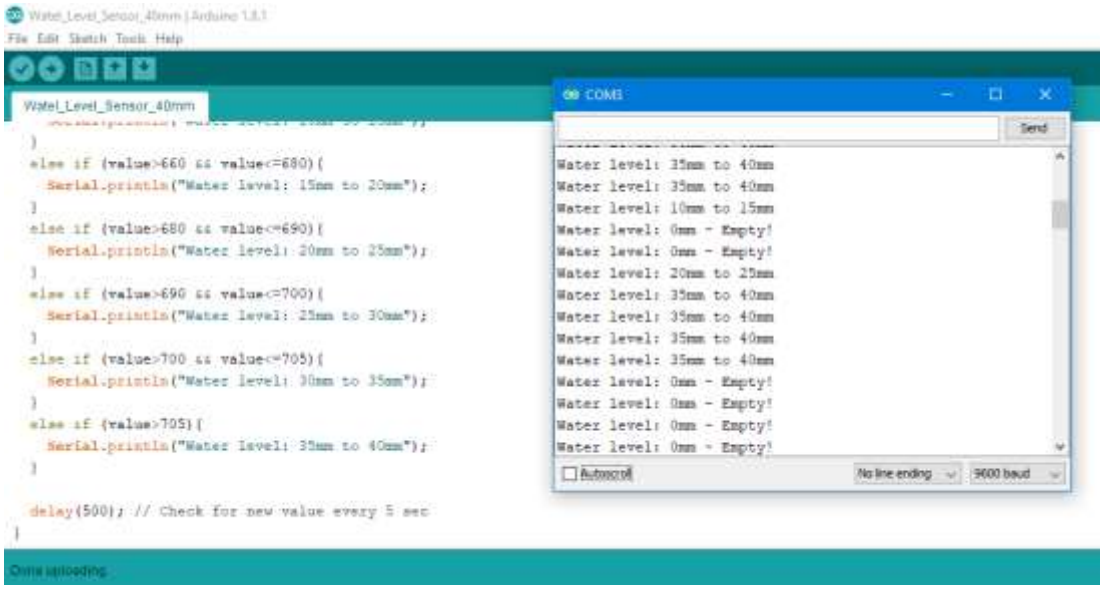
Experiment 20. Water Level Sensor:

Fig 27. Water Level Sensor.



```
const int angVal = A0; //Sensor AO pin to Arduino pin A0
int value;          //Variable to store the incomming data
void setup()
{  Serial.begin(9600); }
void loop()
{ value = analogRead(angVal); //Read data from analog pin and store it to value
variable
 if (value<=480)
 {Serial.println("Water level: 0mm - Empty!");}
  else if (value>480 && value<=530){
    Serial.println("Water level: 0mm to 5mm");
  }
  else if (value>530 && value<=615){
    Serial.println("Water level: 5mm to 10mm");
  }
  else if (value>615 && value<=660){
    Serial.println("Water level: 10mm to 15mm");
  }
  else if (value>660 && value<=680){
    Serial.println("Water level: 15mm to 20mm");
  }
  else if (value>680 && value<=690){
    Serial.println("Water level: 20mm to 25mm");
  }
  else if (value>690 && value<=700){
    Serial.println("Water level: 25mm to 30mm");
  }
  else if (value>700 && value<=705){
    Serial.println("Water level: 30mm to 35mm");
```

```
  }
  else if (value>705){
    Serial.println("Water level: 35mm to 40mm");
  }

  delay(500); // Check for new value every 5 sec
}
```



Experiment 21. Soil Moisture Sensor:

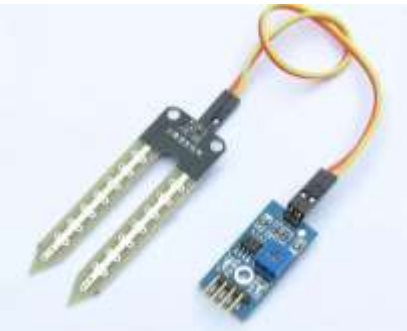
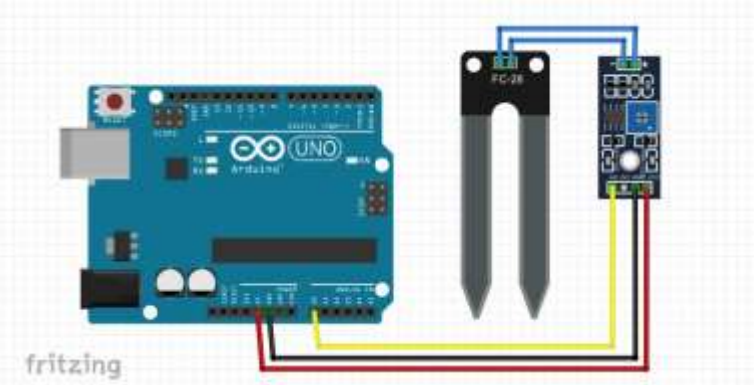


Fig 28.Soil Moisture Sensor.




```
const int hygrometer = A0;  //Hygrometer sensor analog pin output at pin
A0 of Arduino
int value;
void setup(){
  Serial.begin(9600);
}
void loop(){
  // When the plant is watered well the sensor will read a value 380~400
  value = analogRead(hygrometer);   //Read analog value
  value = constrain(value, 400, 1023);  //Keep the ranges!
  value = map(value,400,1023,100,0);  //Map value : 400 will be 100 and
1023 will be 0
  Serial.print("Soil humidity: ");
  Serial.print(value);
  Serial.println("%");
  delay(2000); //Read every 2 sec.
}
```

Experiment 22. Hall Sensor:

Fig 29.Analog Magnetic Hall Sensor.

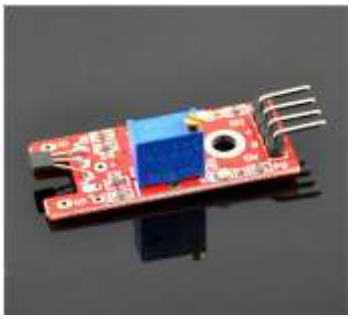


Fig 30.Linear Magnetic Hall Sensor.

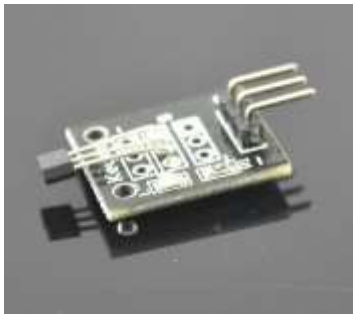


Fig 31.Digital Magnetic Hall Sensor.

Analog magnetic field sensor module measures strength of the field and given it by an analog voltage at the signal pin of the module. The sensor is connected to GND and 5V of the Arduino board. The output voltage is measured by analog pin A5 on the Arduino board. Digital hall magnetic is a magnetic switch. If no magnetic field is present, the signal line of the sensor is HIGH (3.5 V). If a magnetic field is presented to the sensor, the signal line goes LOW, at the same time the LED on the sensor lights up. The polarity of the magnetic field is of influence to the switching action. The front side of the sensor needs the opposite polarity as the back of the sensor to switch on.

```
//analog hall sensor
int sensorPin = A5;
int ledPin = 13;
int sensorValue = 0;
void setup () {
  pinMode (ledPin, OUTPUT);
  Serial.begin (9600);
}
void loop () {
  sensorValue = analogRead (sensorPin);
  digitalWrite (ledPin, HIGH);
  delay (sensorValue);
  digitalWrite (ledPin, LOW);
  delay (sensorValue);
  Serial.println (sensorValue, DEC);
}
```

```
//Digital hall sensor
int Led = 13 ;
int SENSOR = 10 ;
void setup()
{
  pinMode(Led, OUTPUT);
  pinMode(SENSOR, INPUT);
}
```

```
void loop()
{
if (digitalRead(SENSOR)== LOW)
// when the Hall sensor detects a magnetic field, Arduino LED lights up
  {
   digitalWrite(Led, HIGH);
  }
 else
  {
   digitalWrite (Led, LOW);
  }
}
```

Experiment 23.Obstacle avoidance sensor module:

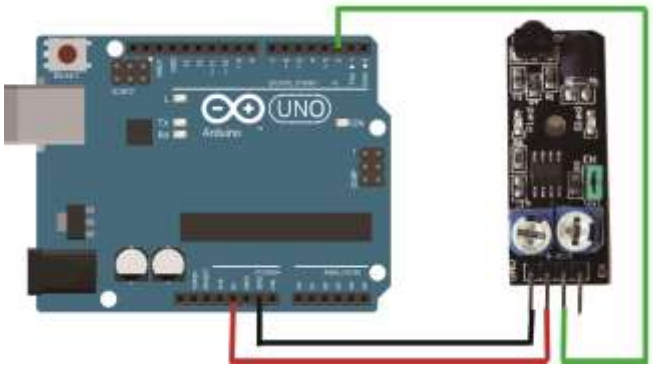


Fig 32. Obstacle avoidance sensor

This Infrared Obstacle Avoidance Sensor returns a signal when it detects an object in range. The range of the sensor is around 2-40 cm is distance. It operates at 3.5 to 5 volts at around 20 milliamps. Infrared obstacle avoidance sensor is designed to detect obstacles or the difference in reflective services. One application is to help a wheeled robot avoid obstacles with a sensor to react to adjustable distance settings. This device has an infrared transmitter and receiver, which forms the sensor pair. The transmitter LED emits a certain frequency of infrared, which the receiver LED will detect. The receiving LED will detect some of the signal back and will trigger the digital on/off "signal" pin when a specific threshold "distance" has been detected. Most boards will have 2 potentiometers, one of which is to adjust how sensitive the sensor is. You can use it to adjust the distance from the object at which the sensor detects it. Typically, the other potentiometer, which changes the transmitter IR frequency is not adjusted.

```
void setup ()
{ pinMode(13, OUTPUT);
  pinMode (2, INPUT);// define the obstacle avoidance sensor interface
}
void loop ()
{   if (digitalRead (2)== HIGH) // When the obstacle avoidance sensor
detects a signal, LED flashes
  {digitalWrite (13, HIGH); }
  else {digitalWrite (13, LOW); }}
```

Experiment 24.Magic Light Cup Module:

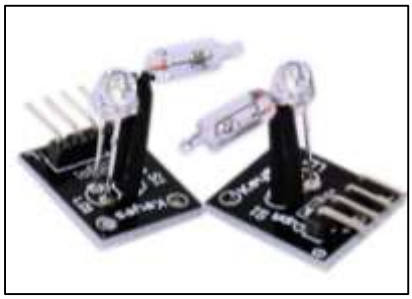
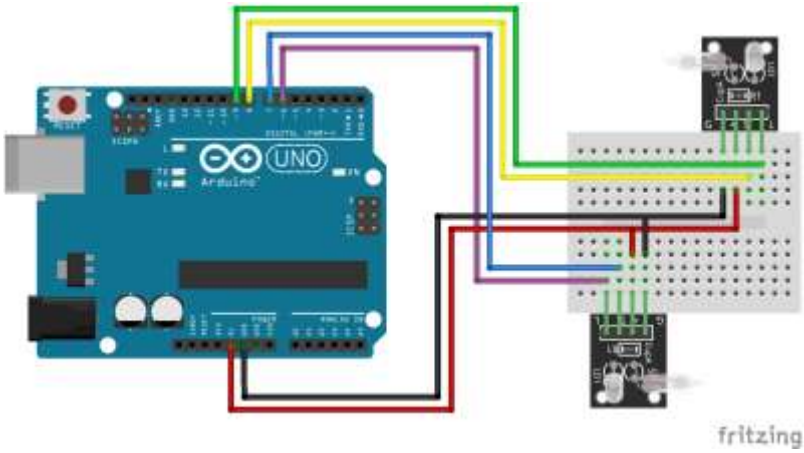


Fig 33. Magic light cup module

Magic Light Cup modules are easy to Interactive Technology Division developed a can and ARDUINO interactive modules, PWM dimming principle is to use the principle of two modules brightness changes. Mercury switches provide a digital signal that triggers the PWM regulator, through the program design, we can see the light like two cups filled with the effect of shuffling back and forth.

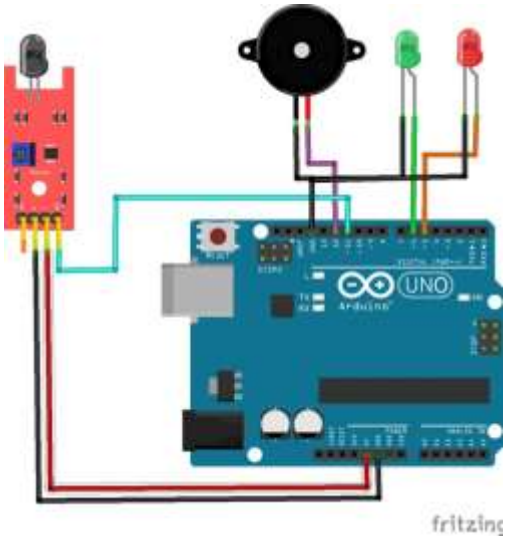



```
// G connect to GND
// + connect to 5V
// L connect pin 6(1st module) & 9(2nd module)
// S connect pin 7(1st module) & 8(2nd module)
int LedPinA = 9;
int LedPinB = 6;
int ButtonPinA = 7;
int ButtonPinB = 8;
int buttonStateA = 0;
int buttonStateB = 0;
int brightness = 0;
void setup ()
```

```
{
  pinMode (LedPinA, OUTPUT);
  pinMode (LedPinB, OUTPUT);
  pinMode (ButtonPinA, INPUT);
  pinMode (ButtonPinB, INPUT);
}
void loop ()
{
  buttonStateA = digitalRead (ButtonPinA);
  if (buttonStateA == HIGH && brightness != 255)
  {
    brightness += 1;
  }
  buttonStateB = digitalRead (ButtonPinB);
  if (buttonStateB == HIGH && brightness != 0)
  {
    brightness -= 1;
  }
  analogWrite(LedPinA, brightness);
  analogWrite(LedPinB, 255 - brightness);
  delay (25);
}
```

Experiment 25. Flame Sensor Module:

Fig 34. Flame Sensor Module



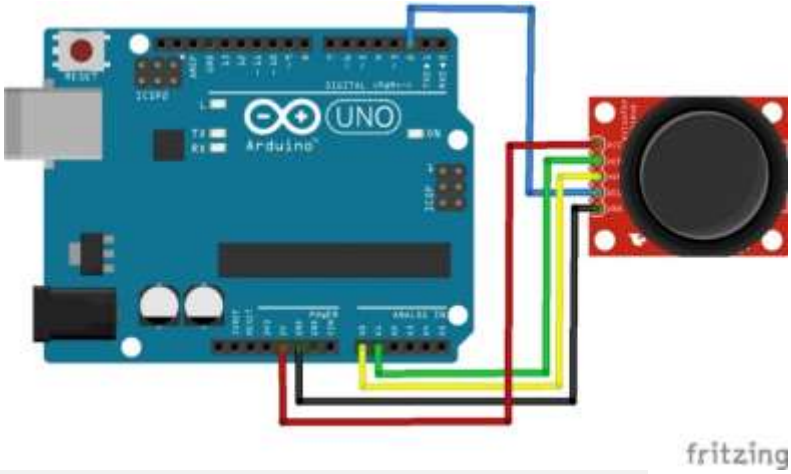
Flame sensor module is sensitive to the flame and radiation. It also can detect ordinary light source in the range of a wavelength 760nm-1100 nm infrared. It can be used as a flame alarm or in firefighting robots. Two outputs mode:
-AO: analog output real-time output voltage signal on the thermal resistance
-DO: when the temperature reaches a certain threshold the output high and low signal threshold adjustable via potentiometer

```
const int buzzerPin = 12;
const int flamePin = 11;
int Flame = HIGH;
int redled = 5;
int greenled = 6;
void setup()
{
pinMode(buzzerPin, OUTPUT);
pinMode(redled, OUTPUT);
pinMode(greenled, OUTPUT);
pinMode(flamePin, INPUT);
Serial.begin(9600);
}
void loop()
{ Flame = digitalRead(flamePin);
if (Flame== LOW)
{
digitalWrite(buzzerPin, HIGH);
digitalWrite(redled, HIGH);
digitalWrite(greenled, LOW);
}
else
{ digitalWrite(buzzerPin, LOW);
  digitalWrite(greenled, HIGH);
  digitalWrite(redled, LOW);
}}
```

Experiment 26. XY-axis Joystick Module:

The joystick is a combination of 2 analog potentiometer and a digital switch.

Fig 35. XY-axis Joystick Module




```
int JoyStick_X = A0; // x
int JoyStick_Y = A1; // y
int JoyStick_Z = 2; // key
void setup()
{ pinMode(JoyStick_X, INPUT);
  pinMode(JoyStick_Y, INPUT);
  pinMode(JoyStick_Z, INPUT_PULLUP);
  Serial.begin(9600); // 9600 bps
}
void loop()
{ int x, y, z;
  x = analogRead(JoyStick_X);
  y = analogRead(JoyStick_Y);
  z = digitalRead(JoyStick_Z);
  Serial.print(x, DEC);
  Serial.print('P');
  Serial.print(y, DEC);
  Serial.print(",");
  Serial.println(z, DEC);
  delay(100);}
```

Experiment 27. Line Follow Sensor Module:

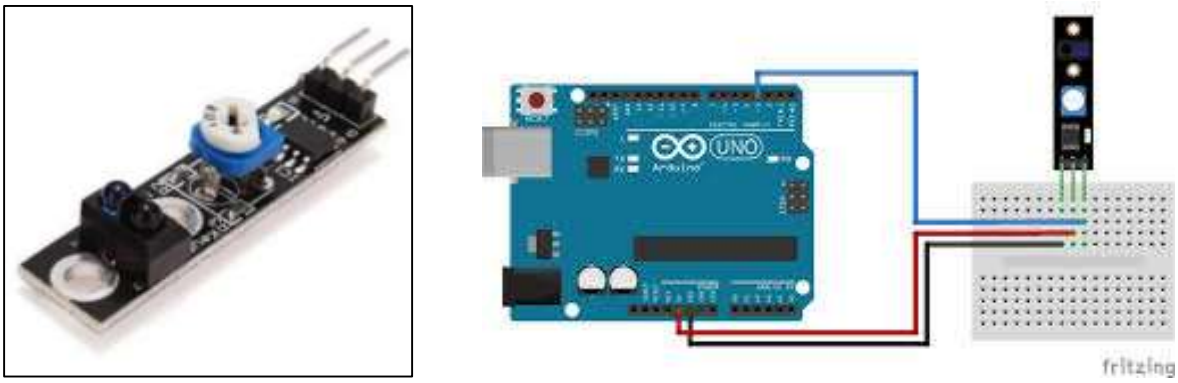


Fig 36. Line Follow Sensor Module

The line follow sensor module has a reflective IR sensor for determining if the surface is light or dark. The sensor can also be used to sense if an object is close or far away. It is used in robots to follow the line or keep inside the restricted area. If no object is in close proximity to the sensor or the surface is dark, the output is high. If a reflective object is near or the surface is light, the output will go high.

```
int led = 13;//LED pin
int sensor = 3; //sensor pin
int val; //numeric variable
void setup()
{
pinMode(led, OUTPUT); //set LED pin as output
pinMode(sensor, INPUT); //set sensor pin as input
}
void loop()
{
val = digitalRead(sensor); //Read the sensor
if(val == HIGH) { digitalWrite(Led, HIGH); }
else { digitalWrite(Led, LOW); }}
```

Experiment 28. IR Transmitter Module and 38 KHz IR Receiver Module:

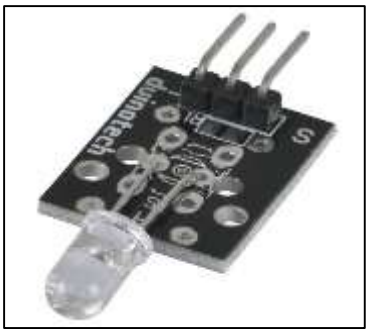


Fig 37.IR Transmitter

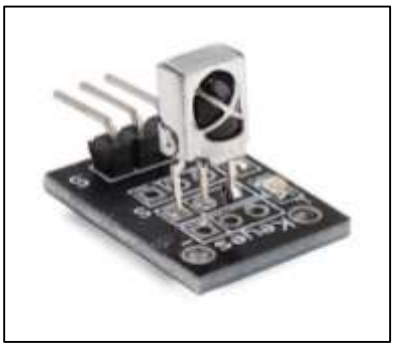
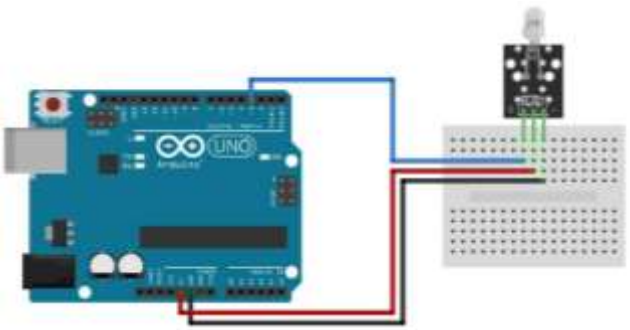
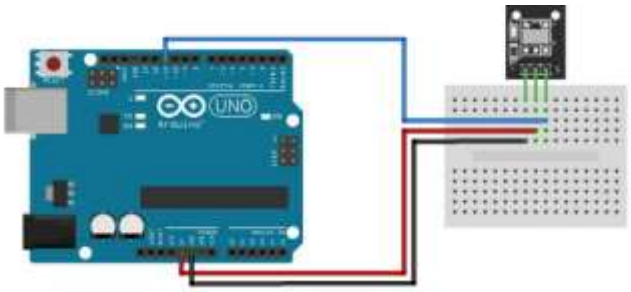


Fig 38.IR Receiver

The Infrared Transmitter Module consists of just a 5mm IR LED. It works together with any 38 kHz receiver and almost any device using regular IR remote control. IR receiver module reacts to 38 kHz modulated infrared light. This module consists of a 38 kHz IR receiver, a 1kΩ resistor and an LED. It works together with a 38 kHz IR transmitter or any regular IR remote control.

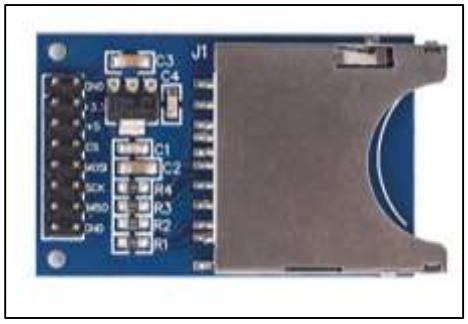




```
//IR Transmitter code
#include <IRremote.h>
IRsend irsend;
void setup()
{
Serial.begin(9600);
}
void loop()
{
 for (int i = 0; i < 50; i++){
 irsend.sendSony(0xa90, 12); // Sony TV power code
 delay(40);
 }
}
```

```
//IR Receiver
#include <IRremote.h>
int RECV_PIN = 11; // define input pin on Arduino
IRrecv irrecv(RECV_PIN);
decode_results results; // decode_results class is defined in IRremote.h
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); // small delay to prevent reading errors
}
```

Experiment 29.SD Card Reader:



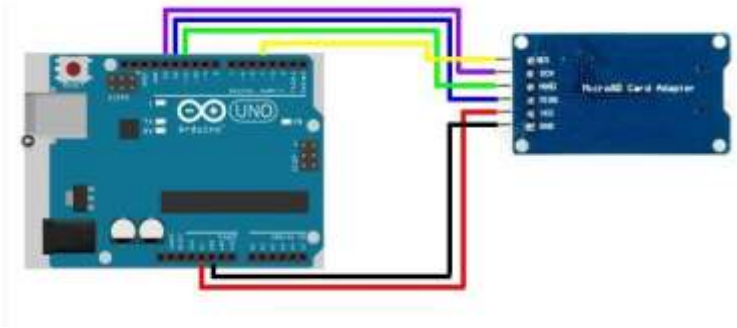


Fig 39.SD Card Reader

This SD Card is used for transferring data to and from a standard SD card. The pin out is directly compatible with Arduino and also can be used with other microcontrollers. It allow us to add mass storage and data logging to our projects.

```
/*
  SD card read/write
 This example shows how to read and write data to and from an SD card
file
 The circuit:
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)
 created   Nov 2010
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe
 This example code is in the public domain.
 */

#include <SPI.h>
#include <SD.h>
File myFile;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
     // wait for serial port to connect. Needed for native USB port only
  }
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;}
  Serial.println("initialization done.");
  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("test.txt", FILE_WRITE);
  // if the file opened okay, write to it:
```

```
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.");
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}
void loop() {
  // nothing happens after setup
}
```

Experiment 30.Photo Interrupter Module:

Fig 40. Photo Interrupter Module

Photo Interrupter Module for Arduino, will trigger a signal when light between the sensor gaps is blocked. The photo Interrupter module consists of an optical emitter/detector in the front and two resistors (1 kΩ and 33 Ω) in the back. The sensor uses a beam of light between de emitter and detector to check if the path between both is being blocked by an opaque object.

Experiment 31. Step Down Power Module (mp1584EN):

Features:

1. Input Voltage: 4.5~28V.
2. Output Voltage: 0.8V~20V.
3. Output Current: 3A (Max).
4. Efficiency: 92 % (Max).
5. Switch Frequency: 1.5MHz (Max), 1MHz (Typical).