**1. What is IoT?**

The Internet of Things is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction

**2. Application of IoT.**

**Consumer applications**

A growing portion of IoT devices is created for consumer use, including connected vehicles, [home automation](https://en.wikipedia.org/wiki/Home_automation), [wearable technology](https://en.wikipedia.org/wiki/Wearable_technology), connected health, and appliances with remote monitoring capabilities.

#### Smart home

IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems. Long-term benefits could include energy savings by automatically ensuring lights and electronics are turned off.

#### Eldercare

One key application of a smart home is to provide [assistance for those with disabilities and elderly individuals](https://en.wikipedia.org/wiki/Home_automation_for_the_elderly_and_disabled). These home systems use assistive technology to accommodate an owner's specific disabilities. Voice can assist users with sight and mobility limitations while alert systems can be connected directly to [cochlear implants](https://en.wikipedia.org/wiki/Cochlear_implant) worn by hearing-impaired users. They can also be equipped with additional safety features. These features can include sensors that monitor for medical emergencies such as falls or seizures. Smart home technology applied in this way can provide users with more freedom and a higher quality of life.

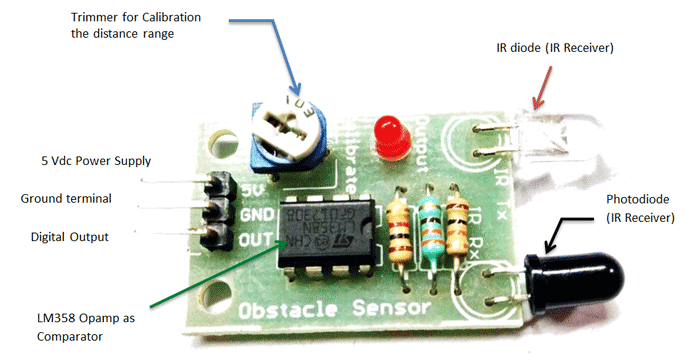
#### Transportation

The IoT can assist in the integration of communications, control, and information processing across various [transportation systems](https://en.wikipedia.org/wiki/Intelligent_transportation_system). Application of the IoT extends to all aspects of transportation systems (i.e. vehicle the infrastructure, and the driver or user). Dynamic interaction between these components of a transport system enables inter- and intra-vehicular communication, [smart traffic control](https://en.wikipedia.org/wiki/Smart_traffic_light), smart parking, [electronic toll collection systems](https://en.wikipedia.org/wiki/Electronic_toll_collection), [logistic](https://en.wikipedia.org/wiki/Logistics_management)s and [fleet management](https://en.wikipedia.org/wiki/Fleet_management), [vehicle control](https://en.wikipedia.org/wiki/Autonomous_cruise_control_system), safety, and road assistance. In Logistics and Fleet Management, for example, an IoT platform can continuously monitor the location and conditions of cargo and assets via wireless sensors and send specific alerts when management exceptions occur (delays, damages, thefts, etc.). This can only be possible with the IoT and its seamless connectivity among devices. Sensors such as GPS, Humidity, and Temperature send data to the IoT platform and then the data is analyzed and then sent to the users. This way, users can track the real-time status of vehicles and can make appropriate decisions. If combined with [Machine Learning](https://en.wikipedia.org/wiki/Machine_learning), then it also helps in reducing traffic accidents by introducing [drowsiness](https://en.wikipedia.org/wiki/Driver_drowsiness_detection) alerts to drivers and providing self-driven cars too.

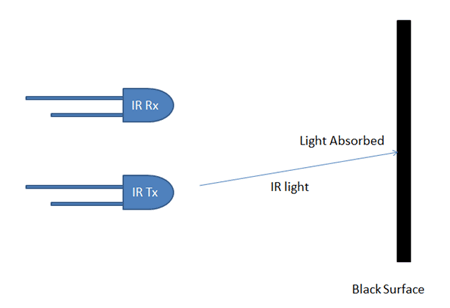
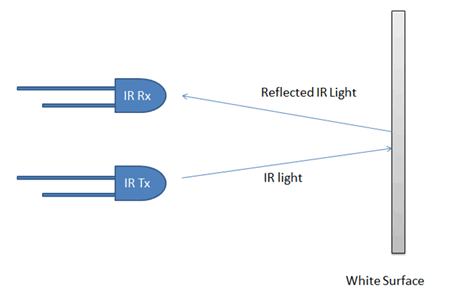
#### Energy management

Significant numbers of energy-consuming devices (e.g. switches, power outlets, bulbs, televisions, etc.) already integrate Internet connectivity, which can allow them to communicate with utilities to balance [power generation](https://en.wikipedia.org/wiki/Electricity_generation) and energy usage and optimize energy consumption as a whole. These devices allow for remote control by users, or central management via a [cloud](https://en.wikipedia.org/wiki/Cloud_computing)-based interface, and enable functions like scheduling (e.g., remotely powering on or off heating systems, controlling ovens, changing lighting conditions, etc.).[]](https://en.wikipedia.org/wiki/Internet_of_things#cite_note-CoMAN-46) The [smart grid](https://en.wikipedia.org/wiki/Smart_grid) is a utility-side IoT application; systems gather and act on energy and power-related information to improve the efficiency of the production and distribution of electricity. Using [advanced metering infrastructure (AMI)](https://en.wikipedia.org/wiki/Smart_meter) Internet-connected devices, electric utilities not only collect data from end-users, but also manage distribution automation devices like transformers

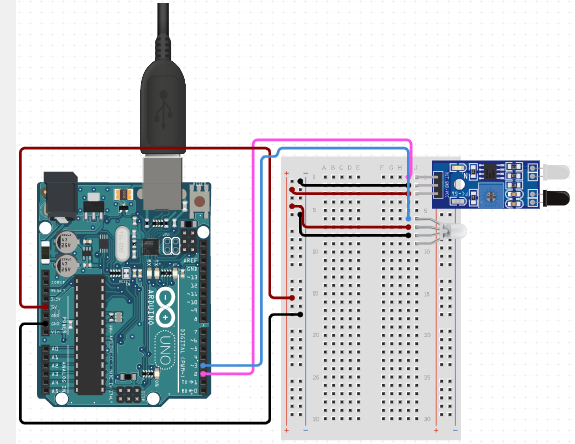
**IR sensor (infrared sensor)**



IR Sensor or Infrared Sensor has two main parts. IR Transmitter and IR Receiver. The work of an IR transmitter or Infrared transmitter is to transmit the infrared waves whereas the work of the IR receiver is to receive these infrared waves. IR receiver constantly sends digital data in the form of 0 or 1 to V Out pin of the sensor.



**Code and Circuit diagram (Arduino)**



int IR\_pin = 2;

int LED\_pin = 3;

void setup() {

pinMode(IR\_pin,INPUT);

pinMode(LED\_pin, OUTPUT);

Serial.begin(9600);

}

void loop() {

if(digitalRead(IR\_pin)==HIGH) {

digitalWrite(LED\_pin, LOW);

}

else {

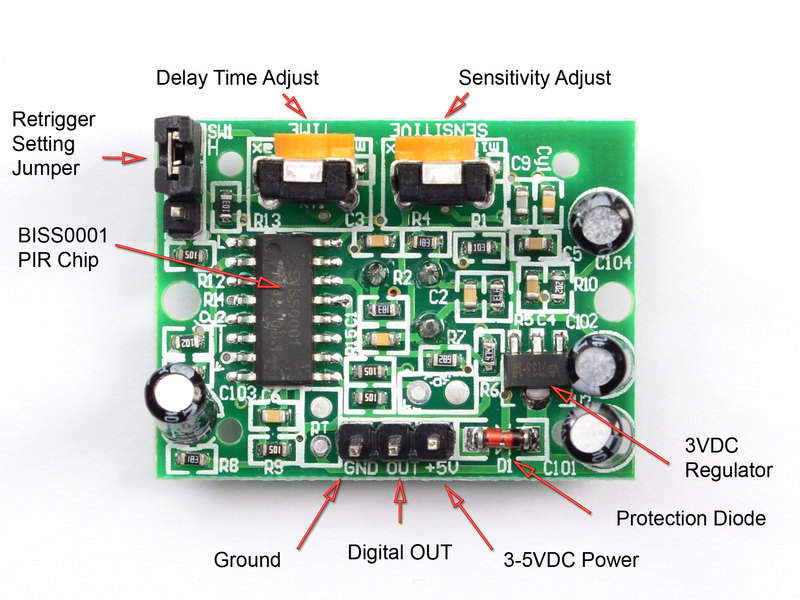
digitalWrite(LED\_pin, HIGH);

}

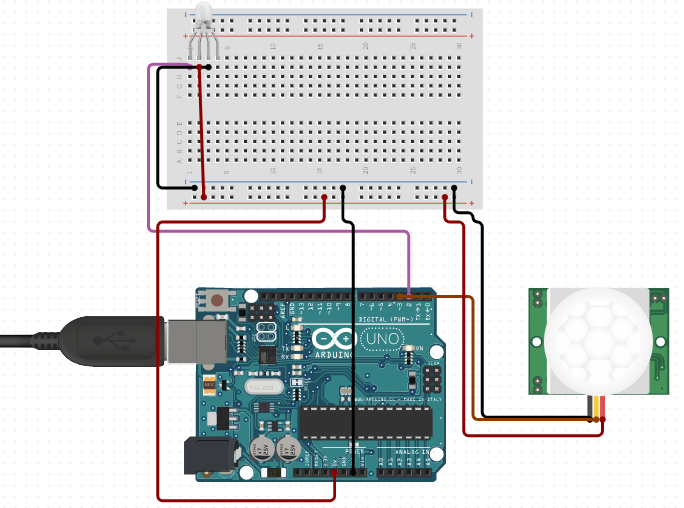
}

**PIR sensor (passive infrared sensor)**

A passive infrared sensor (PIR sensor) is an electronic [sensor](https://en.wikipedia.org/wiki/Sensor) that measures [infrared](https://en.wikipedia.org/wiki/Infrared) (IR) light radiating from objects in its field of view. They are most often used in PIR-based [motion detectors](https://en.wikipedia.org/wiki/Motion_detector). PIR sensors are commonly used in security alarms and automatic lighting applications. PIR sensors detect general movement, but do not give information on who or what moved. For that purpose, an [active IR sensor](https://en.wikipedia.org/wiki/Active_IR_sensor) is required.PIR sensors are commonly called simply "PIR", or sometimes "PID", for "passive infrared detector". The term *passive* refers to the fact that PIR devices do not radiate energy for detection purposes. They work entirely by detecting [infrared radiation](https://en.wikipedia.org/wiki/Infrared_radiation) (radiant heat) emitted by or reflected from objects.



**Code and Circuit diagram (Arduino)**



int PIR\_pin = 3;

int LED\_pin = 2;

void setup() {

pinMode(PIR\_pin,INPUT);

pinMode(LED\_pin,OUTPUT);

Serial.begin(9600);

}

void loop() {

int var1 = digitalRead( PIR\_pin);

Serial.println(var1);

if (var1==1){

digitalWrite(LED\_pin, LOW);

}

else{

digitalWrite(LED\_pin, HIGH);

}

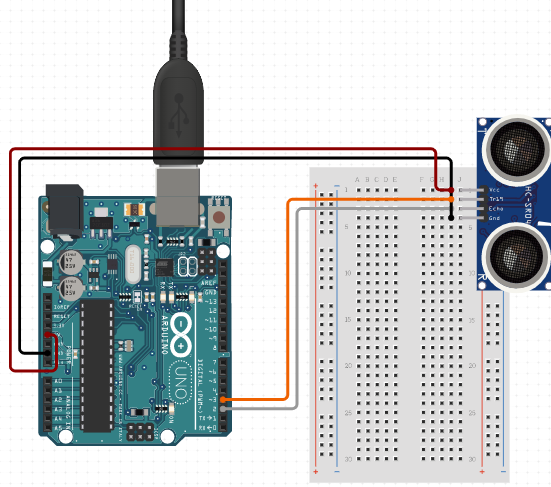
}

**ultrasonic sensors**

ultrasonic sensors are almost completely insensitive to interfering factors (such as extraneous light, dust, smoke, mist, vapor, lint, oily air, etc.). They are best suited for the detection of transparent and dark objects, reflective surfaces, shiny objects and of bulk materials and liquids. Ultrasonic sensors allow for the reliable detection and measurement of objects, independent of their material, color, transparency and texture.



**Code and Circuit diagram (Arduino)**



const int trigger=3;

const int echo=2;

long duration,inches,cm;

void setup()

{

pinMode(trigger,OUTPUT);

pinMode(echo,INPUT);

Serial.begin(9600);

}

void loop()

{

digitalWrite(trigger,LOW);

delayMicroseconds(100);

digitalWrite(trigger,HIGH);

delayMicroseconds(100);

digitalWrite(trigger,LOW);

pinMode(echo,INPUT);

duration=pulseIn(echo,HIGH);

cm=(duration/2)/29.1;

inches=(duration/2)/74;

Serial.print("Duration:");

Serial.print(inches);

Serial.print(" inches ");

Serial.print(cm);

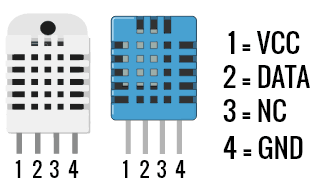
Serial.println(" cm");

delay(1000);

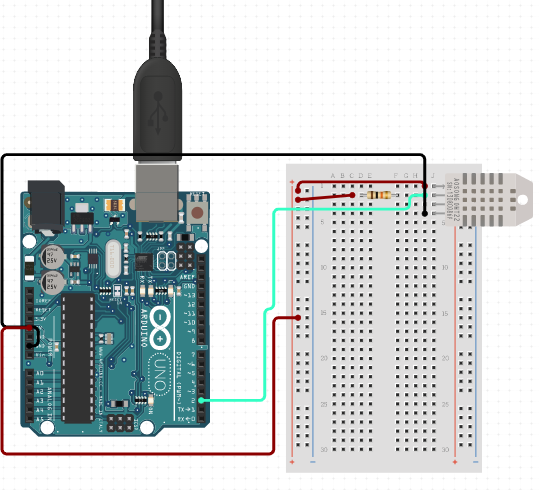
}

**DHT22**  **Sensor** (Digital Temperature And Humidity)

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).Simply connect the first pin on the left to 3-5V power, the second pin to your data input pin and the right most pin to ground. Although it uses a single-wire to send data it is not Dallas One Wire compatible! If you want multiple sensors, each one must have its own data pin.Compared to the DHT11, this sensor is more precise, more accurate and works in a bigger range of temperature/humidity, but its larger and more expensive.Comes with a 4.7K - 10K resistor, which you will want to use as a pullup from the data pin to VCC



**Code and Circuit diagram (Arduino)**



/\*Install the libraries before starting

||Tools->Manage libraries

|->Adafruit unifed Sensor

|->DHT22 Sensor Library \*/

#include "DHT.h";

#define DHTPIN 4

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

Serial.println("DHTxx test!");

dht.begin(); //Start

}

void loop()

{

delay(2000);

float h = dht.readHumidity();

float t = dht.readTemperature();

// if (isnan(h) || isnan(t) )

// {

// Serial.println("Failed to read from DHT sensor!");

// return;

// }

// Serial.print("Humidity: ");

Serial.print(h);

// Serial.print(" %\t");

// Serial.print("Temperature: ");

Serial.print(t);

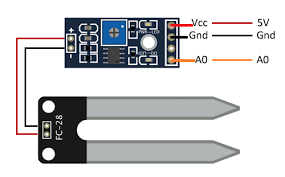
// Serial.print(" \*C ");

// Serial.print("\n ");

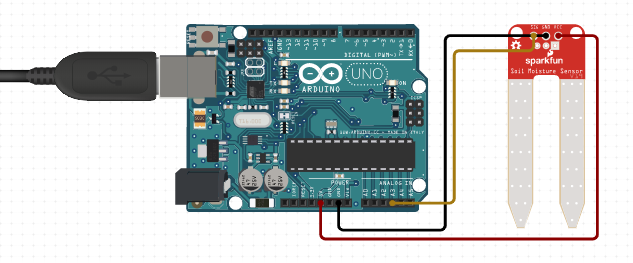
}

**Moisture**

The soil moisture sensor consists of two probes that are used to measure the volumetric content of water. The two probes allow the current to pass through the soil and then it gets the resistance value to measure the moisture value. When there is more water, the soil will conduct more electricity which means that there will be less resistance. Therefore, the moisture level will be higher. Dry soil conducts electricity poorly, so when there will be less water, then the soil will conduct less electricity which means that there will be more resistance. Therefore, the moisture level will be lower. This sensor can be connected in two modes; Analog mode and digital mode. First, we will connect it in Analog mode and then we will use it in Digital mode.



**Code and Circuit diagram (Arduino)**



const int moistpin=A0;

void setup()

{

pinMode(moistpin,INPUT);

Serial.begin(9600);

}

void loop()

{

// put your main code here, to run repeatedly:

long int per=analogRead(moistpin);

per=map(per,1023,0,0,100) ;

Serial.print(per);

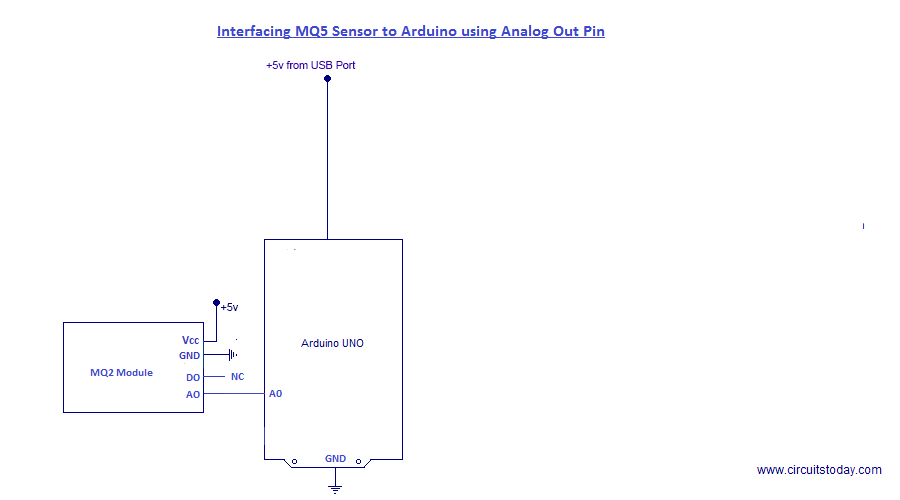
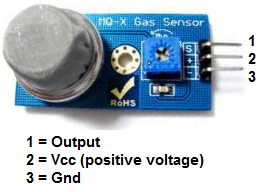
Serial.println("%");

delay(1000);

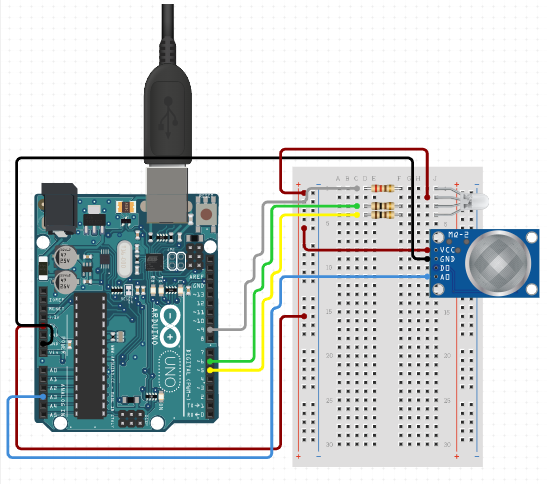
}

**MQ2 Sensor**

MQ2 is basically a general-purpose gas sensor (similar to MQ5) which can sense a broad range of gases like LPG, Butane, Methane(CH4), Hydrogen and in addition to these gases MQ2 is sensitive to smoke as well. we have written a tutorial on [**Interfacing MQ5 Gas sensor to Arduino**](http://www.circuitstoday.com/interfacing-mq5-lpg-sensor-to-arduino). Both MQ5 and MQ2 are basically gas sensors but their range of sensing different gas levels vary. For example, MQ5 can sense LPG in a broader range of 200 ppm to 10000 ppm, whereas the range of MQ2 for LPG is short and is from 5000 ppm to 10000 ppm. Similarly, MQ2 is sensitive to smoke whereas MQ5 is not that sensitive to smoke. So we can not choose the MQ5 gas sensor to design a smoke alarm or smoke involving applications. MQ2 can sense methane(CH4) up to 20000 ppm whereas MQ5 can sense CH4 only up to 10000 ppm. The difference between MQ5 and MQ2, therefore, lies in its range of values. We choose the right sensor based on the application requirement; say for example – We can not choose MQ2 for sensing low levels of LPG in the range of 700 ppm because MQ2 is insensitive to low levels of LPG (its range begins at 5000 ppm and extends to 10000 ppm). So for applications to sense low levels of LPG, MQ5 is the ideal choice as it can sense values starting from 200 ppm.



**Code and Circuit diagram (Arduino)**



int sensor=7;

int gas\_value;

void setup()

{pinMode(sensor,INPUT);

Serial.begin(9600);

}

void loop(){

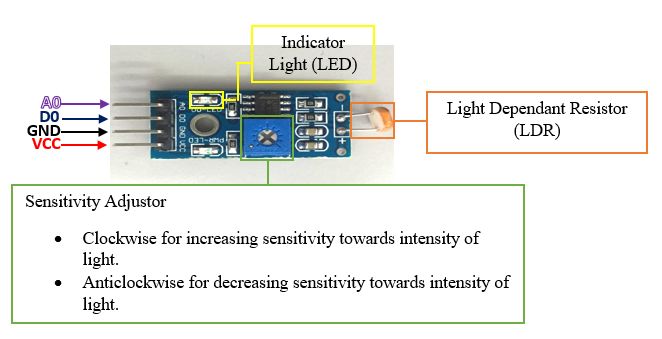
gas\_value=digitalRead(sensor);

Serial.println(gas\_value);

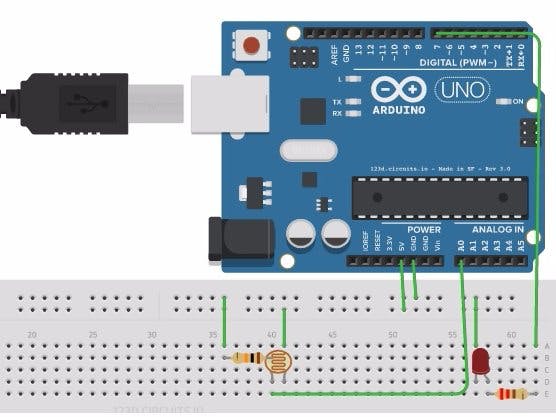
}

## LDR (Light Dependent Resistor)

This system works by sensing the intensity of light in its environment. The sensor that can be used to detect light is an [LDR](https://en.wikipedia.org/wiki/Photoresistor). It's inexpensive, and you can buy it from any local electronics store or online. The LDR gives out an analog voltage when connected to VCC (5V), which varies in magnitude in direct proportion to the input light intensity on it. That is, the greater the intensity of light, the greater the corresponding voltage from the LDR will be. Since the LDR gives out an analog voltage, it is connected to the analog input pin on the Arduino. The Arduino, with its built-in ADC (analog-to-digital converter), then converts the analog voltage (from 0-5V) into a digital value in the range of (0-1023). When there is sufficient light in its environment or on its surface, the converted digital values read from the LDR through the Arduino will be in the range of 800-1023.



**Code and Circuit diagram (Arduino)**



const int lamp = 7;

boolean x = true;

void setup() {

Serial.begin(9600);

pinMode(lamp , OUTPUT);

}

void loop() {

int c = analogRead(A0);

delay(500);

if ( c<300 && x == true){

digitalWrite(7,HIGH);

x = false;

delay(1000);

}

else if ( c <300 && x == false){

x = true;

digitalWrite(7,LOW);

delay(1000);

}

}

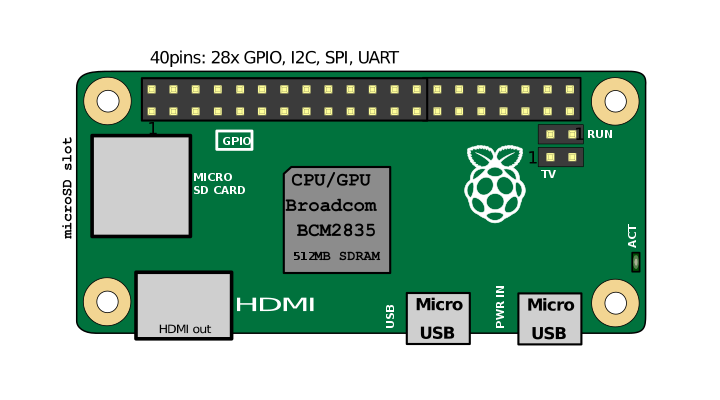
**Raspberry Pi**

What is Raspberry Pi

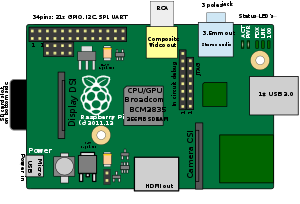
The Raspberry Pi is a series of small [single-board computers](https://en.wikipedia.org/wiki/Single-board_computer) developed in the [United Kingdom](https://en.wikipedia.org/wiki/United_Kingdom) by the [Raspberry Pi Foundation](https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation) to promote the teaching of basic [computer science](https://en.wikipedia.org/wiki/Computer_science) in schools and in [developing countries](https://en.wikipedia.org/wiki/Developing_countries). The original model became far more popular than anticipated, selling outside its [target market](https://en.wikipedia.org/wiki/Target_market) for uses such as [robotics](https://en.wikipedia.org/wiki/Robotics). It does not include peripherals (such as [keyboards](https://en.wikipedia.org/wiki/Keyboard_(computing)) and [mice](https://en.wikipedia.org/wiki/Mouse_(computing))) or [cases](https://en.wikipedia.org/wiki/Computer_case). However, some accessories have been included in several official and unofficial bundles. The organization behind the Raspberry Pi consists of two arms. The first two models were developed by the Raspberry Pi Foundation. After the Pi Model B was released, the Foundation set up Raspberry Pi Trading, with [Eben Upton](https://en.wikipedia.org/wiki/Eben_Upton) as CEO, to develop the third model, the B+. Raspberry Pi Trading is responsible for developing technology while the Foundation is an educational charity to promote the teaching of basic computer science in schools and in developing countries. According to the Raspberry Pi Foundation, more than 5 million Raspberry Pi was sold by February 2015, making it the best-selling [British computer](https://en.wikipedia.org/wiki/British_computer). By November 2016 they had sold 11 million units, and 12.5m by March 2017, making it the third best-selling "general-purpose computer”. In July 2017, sales reached nearly 15 million.[[14]](https://en.wikipedia.org/wiki/Raspberry_Pi#cite_note-14) In March 2018, sales reached 19 million. Most Pis are made in a [Sony](https://en.wikipedia.org/wiki/Sony) factory in [Pencoed](https://en.wikipedia.org/wiki/Pencoed), Wales.[[16]](https://en.wikipedia.org/wiki/Raspberry_Pi#cite_note-16) Some are made in China and Japan.

**Type of Model**

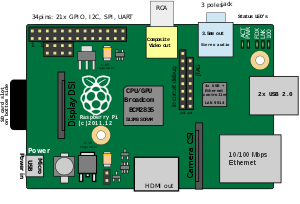
#### Pi Zero



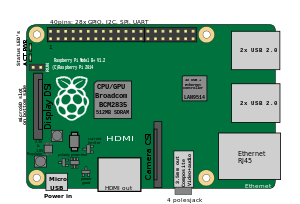
#### Model A



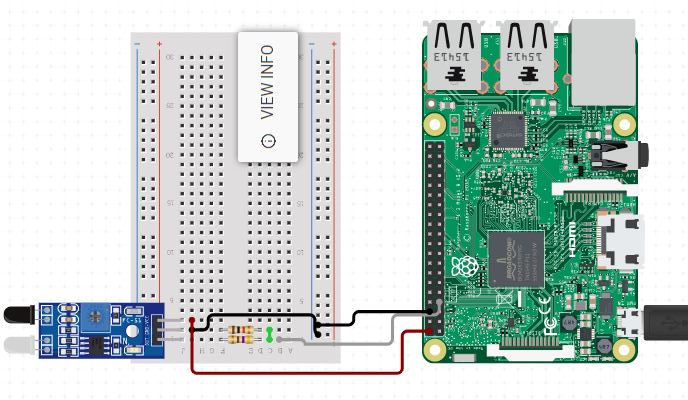
1. Model B



1. Raspberry Pi 1 Model B revision 1.2



**IR Sensor**



import RPi.GPIO as IO

import time

IO.setwarnings(False)

IO.setmode(IO.BCM)

IO.setup(2,IO.OUT) #GPIO 2 -> Red LED as output

IO.setup(3,IO.OUT) #GPIO 3 -> Green LED as output

IO.setup(14,IO.IN) #GPIO 14 -> IR sensor as input

while 1:

if(IO.input(14)==True):

IO.output(2,True)

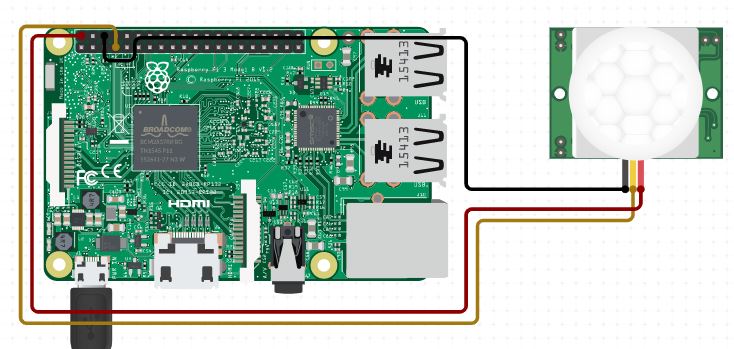
IO.output(3,False)

if(IO.input(14)==False):

IO.output(3,True)

IO.output(2,False)

**PIR Sensor**



import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

PIR\_PIN=40

LIGHT\_PIN=26

GPIO.setup(LIGHT\_PIN, GPIO.OUT)

GPIO.output(LIGHT\_PIN, False)

def readPIR(pin):

GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, False)

time.sleep(0.1)

GPIO.setup(pin, GPIO.IN)

i = GPIO.input(pin)

if i == False:

print("NO INTRUDERS", i)

GPIO.output(LIGHT\_PIN, False)

elif i == True:

print("INTRUDERS DETECTED", i)

GPIO.output(LIGHT\_PIN, True)

try:

while True:

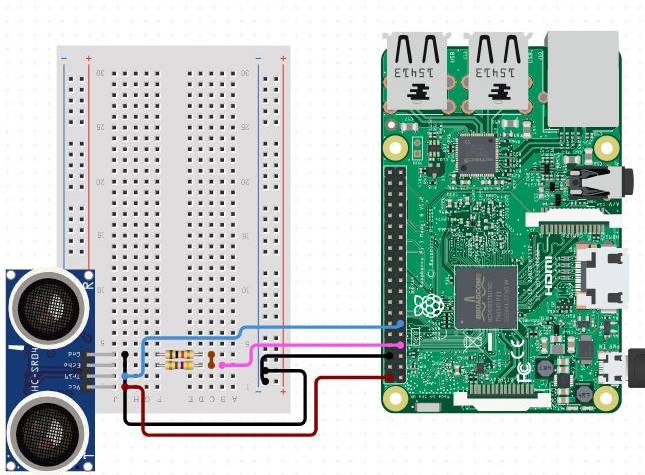
readPIR(PIR\_PIN)

except KeyboardInterrupt:

GPIO.cleanup()

exit()

**Ultrasonic sensors**



import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

TRIG = 16

ECHO = 18

i=0

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)

print ("Calibrating.....")

time.sleep(2)

print ("Place the object......")

try:

while True:

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:

pulse\_start = time.time()

while GPIO.input(ECHO)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance+1.15, 2)

if distance<=20 and distance>=5:

print ("distance:",distance,"cm")

i=1

if distance>20 and i==1:

print ("place the object....")

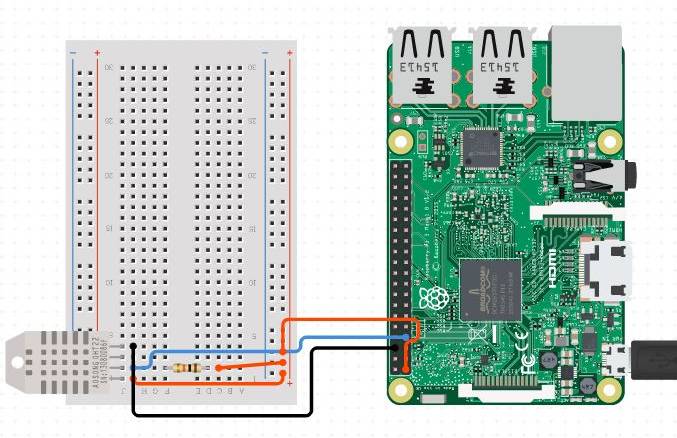
i=0

time.sleep(2)

except KeyboardInterrupt:

GPIO.cleanup()

**DHT22**  **Sensor**



import sys

import Adafruit\_DHT

sensor=Adafruit\_DHT.DHT22

pin=4

humidity,temperature=Adafruit\_DHT.read\_retry(sensor,pin)

if humidity is not None and temperature is not None:

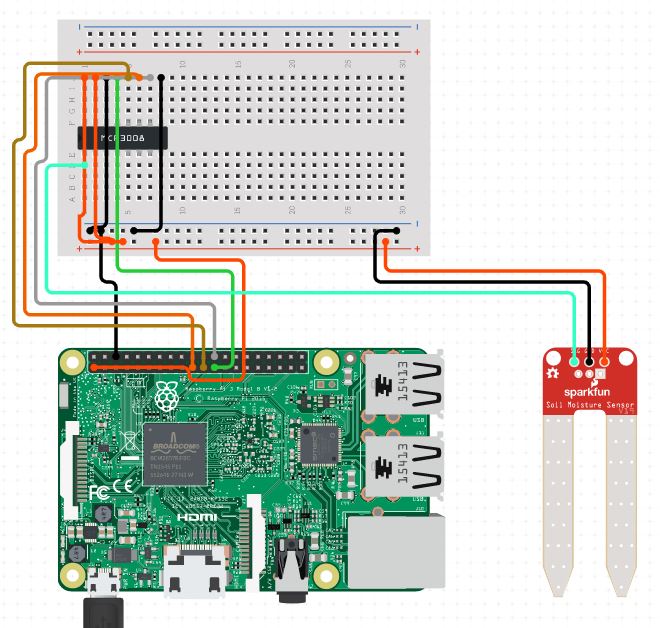
print('Temp={0:0.1f}\* Humidity={1:0.1f}%'.format(temperature,humidity))

else:

print('Failed to get reading. Try again')

sys.exit(1)

**Moisture**



import RPi.GPIO as GPIO

import time

#GPIO SETUP

channel = 21

GPIO.setmode(GPIO.BCM)

GPIO.setup(channel, GPIO.IN)

def callback(channel):

if GPIO.input(channel):

print ("Water Detected!")

else:

print ("No Water Detected!")

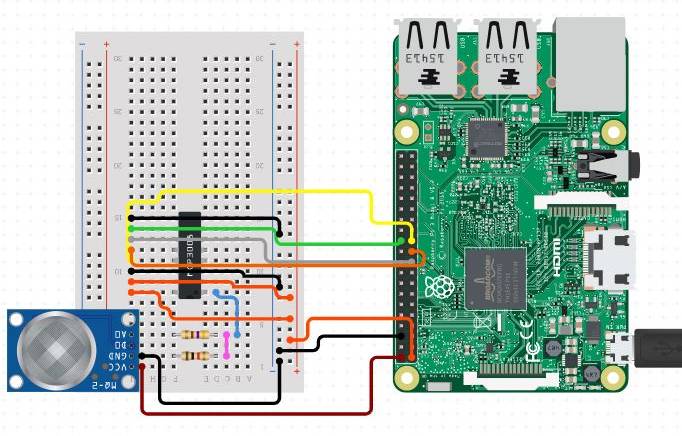
GPIO.add\_event\_detect(channel, GPIO.BOTH, bouncetime=300)

GPIO.add\_event\_callback(channel, callback)

while True:

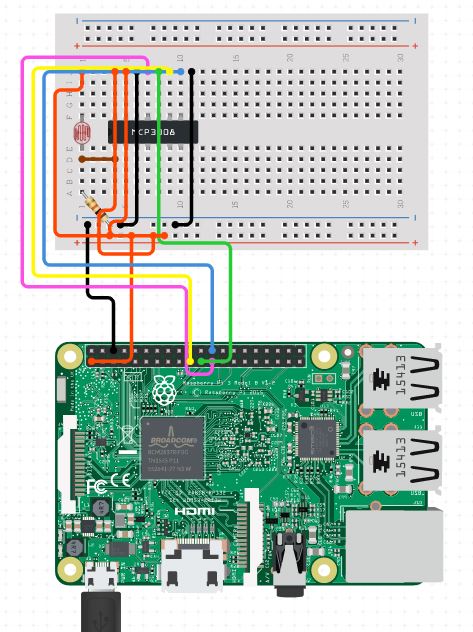
time.sleep(1)

**MQ2 Sensor**



|  |
| --- |
| from spidev import SpiDev |
|  |  |
|  | class MCP3008: |
|  | def \_\_init\_\_(self, bus = 0, device = 0): |
|  | self.bus, self.device = bus, device |
|  | self.spi = SpiDev() |
|  | self.open() |
|  | self.spi.max\_speed\_hz = 1000000 # 1MHz |
|  |  |
|  | def open(self): |
|  | self.spi.open(self.bus, self.device) |
|  | self.spi.max\_speed\_hz = 1000000 # 1MHz |
|  |  |
|  | def read(self, channel = 0): |
|  | cmd1 = 4 | 2 | (( channel & 4) >> 2) |
|  | cmd2 = (channel & 3) << 6 |
|  |  |
|  | adc = self.spi.xfer2([cmd1, cmd2, 0]) |
|  | data = ((adc[1] & 15) << 8) + adc[2] |
|  | return data |
|  |  |
|  | def close(self): |
|  | self.spi.close() |

LDR Sensor



import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

ldr\_thrhold = 100000

LDR\_PIN=18

LIGHT\_PIN=26

GPIO.setup(LIGHT\_PIN,GPIO.OUT)

GPIO.output(LIGHT\_PIN,False)

def readLDR(pin):

reading=0

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin,False)

time.sleep(0.1)

GPIO.setup(pin,GPIO.IN)

while(GPIO.input(pin)==False):

reading+=1

if reading>ldr\_thrhold:

break

return reading

def switchon(pin):

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin,True)

def switchoff(pin):

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin,False)

try:

while True:

ldr\_reading = readLDR(LDR\_PIN)

print(ldr\_reading)

if ldr\_reading<ldr\_thrhold:

switchoff(LIGHT\_PIN)

else:

switchon(LIGHT\_PIN)

time.sleep(0.5)

except KeyboardInterrupt:

GPIO.cleanup()

exit()