OPERATIONS OF TCS-3200, MPU-6050, HC-05, LOAD CELL AND SHARP IR

SET-5

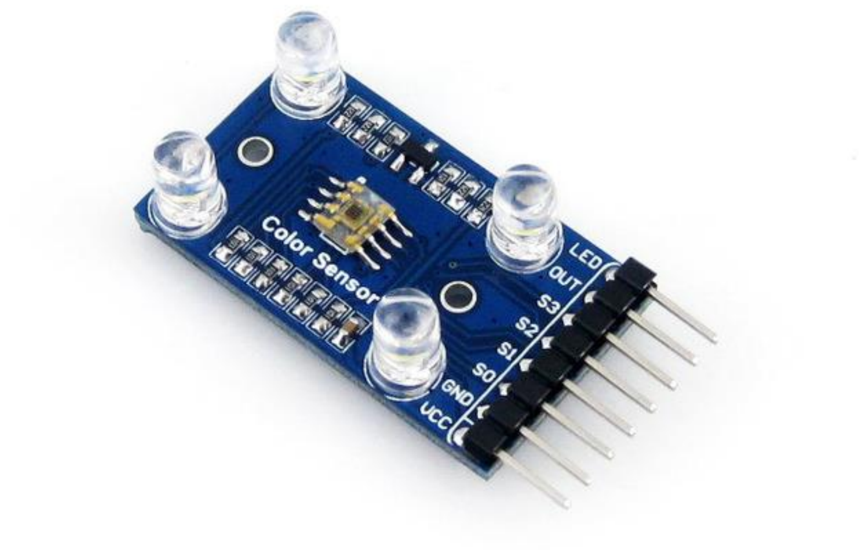


Figure 1. TCS-3200

By

AAYUSH KUMAR (Electrical)

TCS-230/ TCS-3200

The TCS230 senses color light with the help of an 8 x 8 array of photodiodes. Then using a Current-to-Frequency Converter the readings from the photodiodes are converted into a square wave with a frequency directly proportional to the light intensity. Finally, using the Arduino Board we can read the square wave output and get the results for the color.

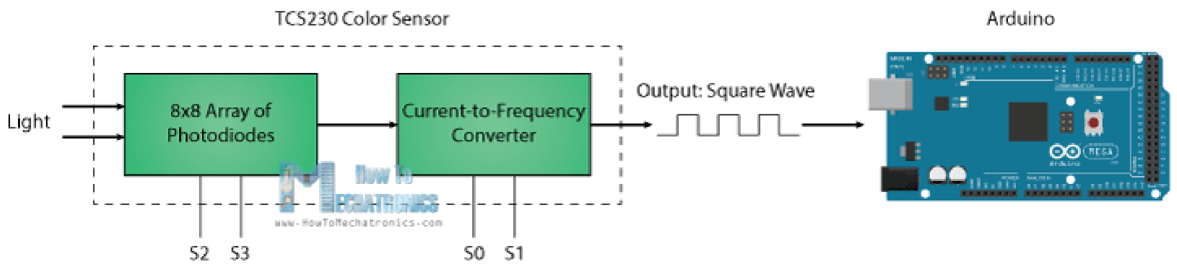


Figure 2. Working of TCS-3200

If we take a closer look at the sensor we can see how it detects various colors. The photodiodes have three different color filters. Sixteen of them have red filters, another 16 have green filters, another 16 have blue filters and the other 16 photodiodes are clear with no filters.

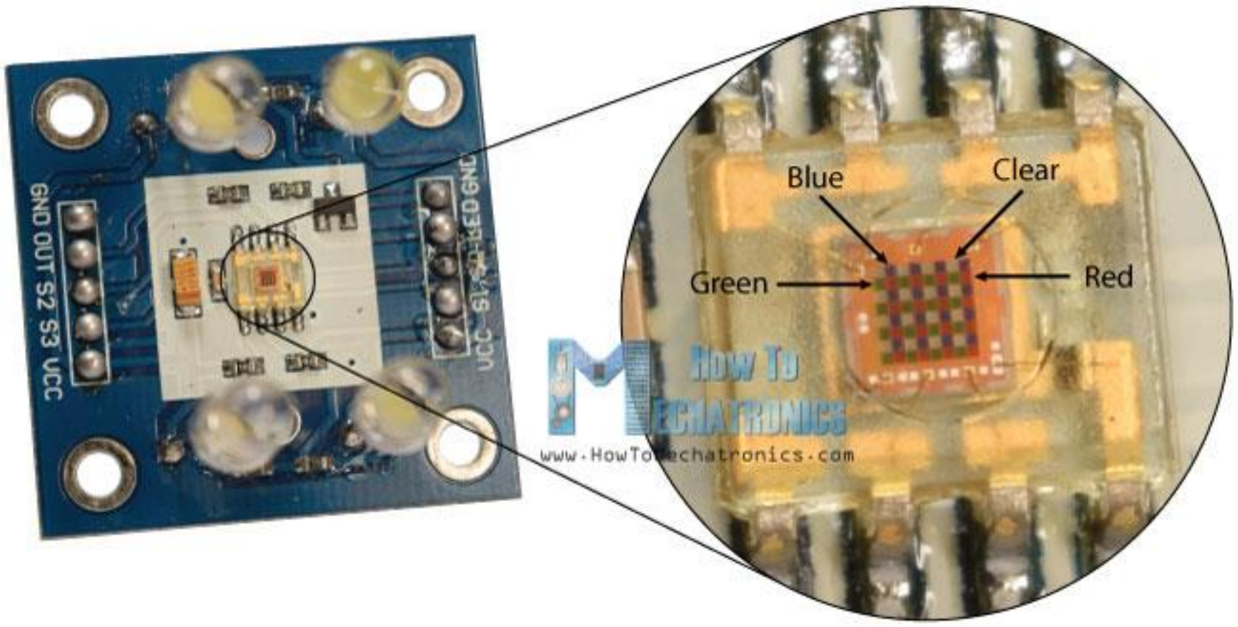
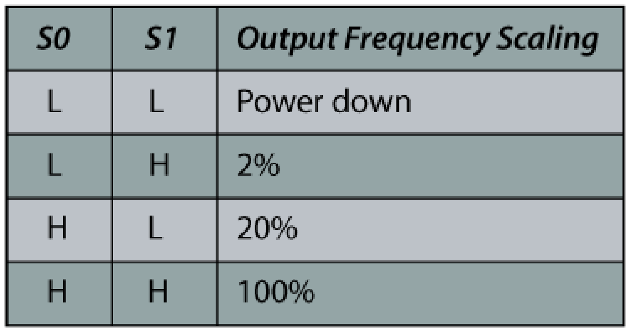
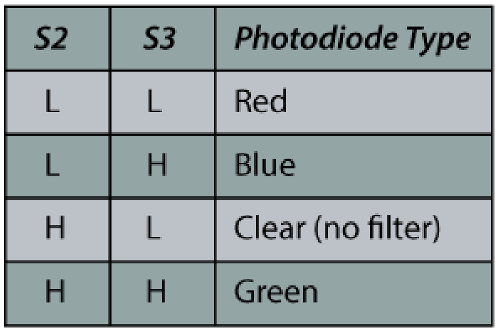


Figure 3. Zoomed view of TCS-3200

Each 16 photodiodes are connected in parallel, so using the two control pins S2 and S3 we can select which of them will be read. So for example, if we want to detect red color, we can just use the 16 red filtered photodiodes by setting the two pins to low logic level according to the table.



S0	*S1*	*Output Frequency Scaling*
L	L	Power down
L	H	2%
H	L	20%
H	H	100%



S2	*S3*	*Photodiode Type*
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

The sensor has two more control pins, S0 and S1 which are used for scaling the output frequency. The frequency can be scaled to three different preset values of 100 %, 20 % or 2%. This frequency-scaling function allows the output of the sensor to be optimized for various frequency counters or microcontrollers.

Now we are ready to move on and connect the TCS230 sensor to the Arduino board. Here's the circuit schematics.

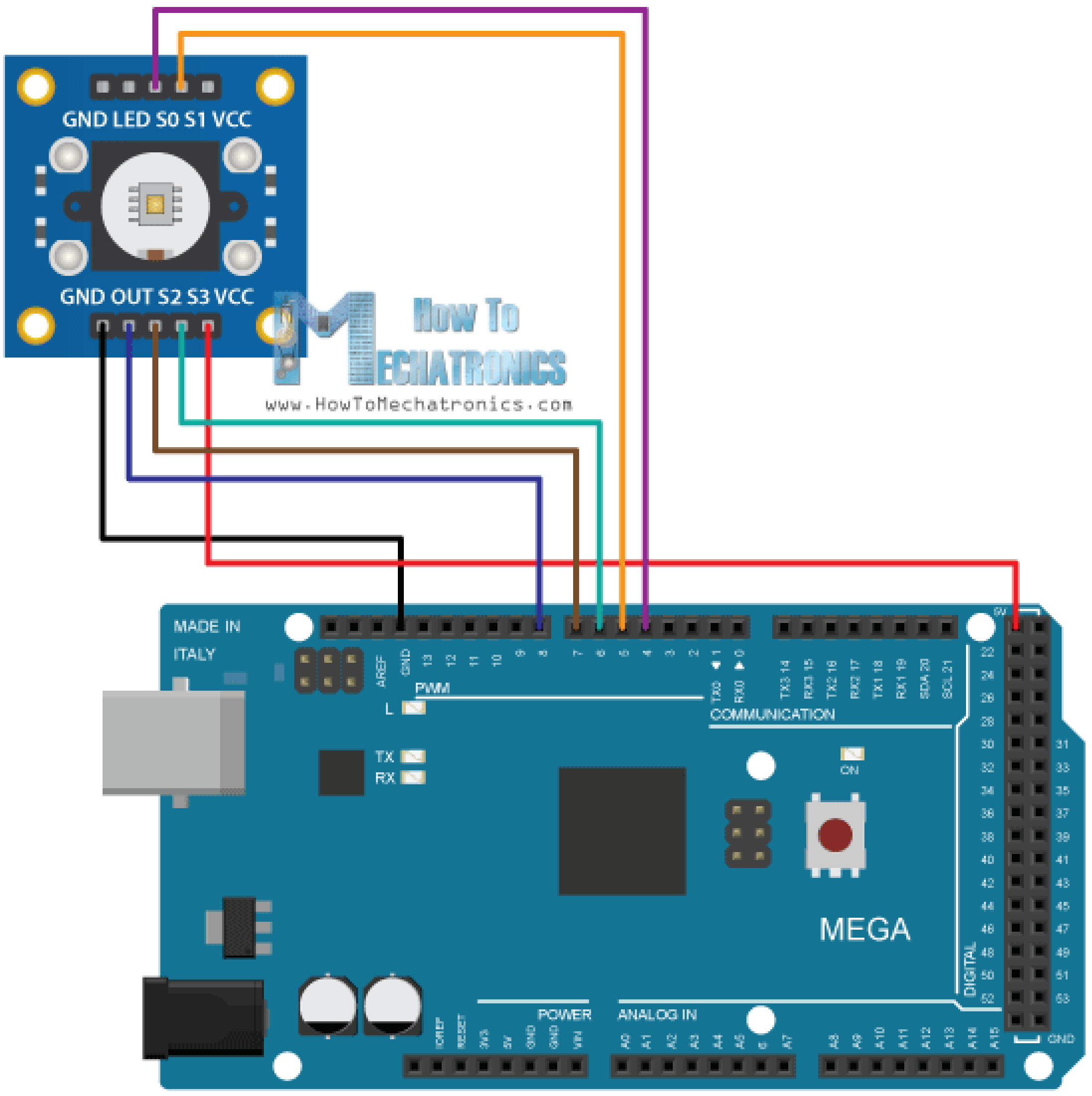


Figure 4. Connections

TCS230 Color Sensor Source Code

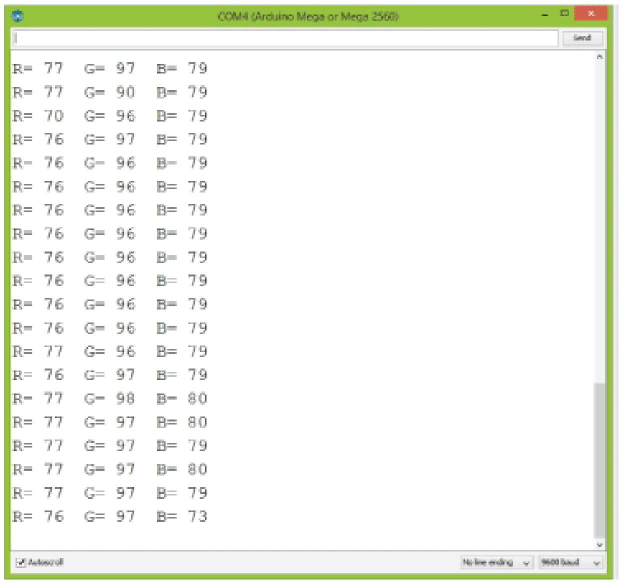
Description: First we need to define the pins to which the sensor is connected and define a variable for reading the frequency. In the setup section we need to define the four control pins as outputs and the sensor output as an Arduino input. Here we also need to set the frequency-scaling, for this example I will set it to 20%, and start the serial communication for displaying the results in the Serial Monitor.

In the loop section, we will start with reading the red filtered photodiodes. For that purpose we will set the two control pins S2 and S3 to low logic level. Then using the “pulseIn()” function we will read the output frequency and put it into the variable “frequency”. Using the Serial.print() function we will print the result on the serial monitor. The same procedure goes for the two other colors, we just need to adjust the control pins for the appropriate color.

THE CODE

```
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

int frequency = 0;

void setup() {
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(sensorOut, INPUT);

// Setting frequency-scaling to 20%
digitalWrite(S0,HIGH);
digitalWrite(S1,LOW);

Serial.begin(9600);
}

void loop() {
// Setting red filtered photodiodes to be read
digitalWrite(S2,LOW);
digitalWrite(S3,LOW);
// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("R= ");//printing name
Serial.print(frequency);//printing RED color frequency
Serial.print(" ");
delay(100);

// Setting Green filtered photodiodes to be read
digitalWrite(S2,HIGH);
digitalWrite(S3,HIGH);
// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("G= ");//printing name
Serial.print(frequency);//printing RED color frequency
Serial.print(" ");
delay(100);

// Setting Blue filtered photodiodes to be read
digitalWrite(S2,LOW);
digitalWrite(S3,HIGH);
// Reading the output frequency
frequency = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("B= ");//printing name
Serial.print(frequency);//printing RED color frequency
Serial.println(" ");
delay(100);
}
```

Now if we run the Serial Monitor we will start getting some values. These values depend on the selected frequency-scaling, as well as from the surrounding lighting.



```
R= 77   G= 97   B= 79
R= 77   G= 90   B= 79
R= 70   G= 96   B= 79
R= 76   G= 97   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 76   G= 96   B= 79
R= 77   G= 96   B= 79
R= 76   G= 97   B= 79
R= 77   G= 98   B= 80
R= 77   G= 97   B= 80
R= 77   G= 97   B= 79
R= 77   G= 97   B= 80
R= 77   G= 97   B= 79
R= 76   G= 97   B= 73
```

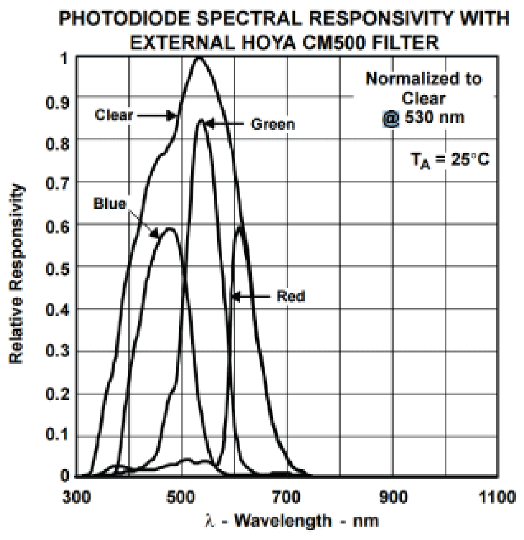


*Figure takan from the TCS230Sensor datasheet

Figure 5. Serieal Monitor view

Note here that three values differ due to the different sensitivity of each photodiode type, as seen from the photodiode spectral responsivity diagram from the datasheet of the sensor.

Nevertheless, now let's see how the values react when we will bring different colors in front of the sensor. So for example, if we bring red color, the initial value will drop down, in my case from around 70 to around 25.

So now if we want to represent the detected colors with the RGB Model which has values from 0 to 255, we will use the map() function to map or convert the readings to the values from 0 to 255.

```
//Remaping the value of the frequency to the RGB Model of 0 to 255
frequency = map(frequency, 25,70,255,0);
```

The value of 70 will be mapped to 0, and the value of 25 to 255. The same procedure goes for the two other colors.

MPU-6050

The MPU 6050 is a sensor based on MEMS (micro electro mechanical systems) technology. Both the accelerometer and the gyroscope are embedded inside a single chip. This chip uses I2C (inter-integrated circuit) protocol for communication.

How Does an Accelerometer Work?

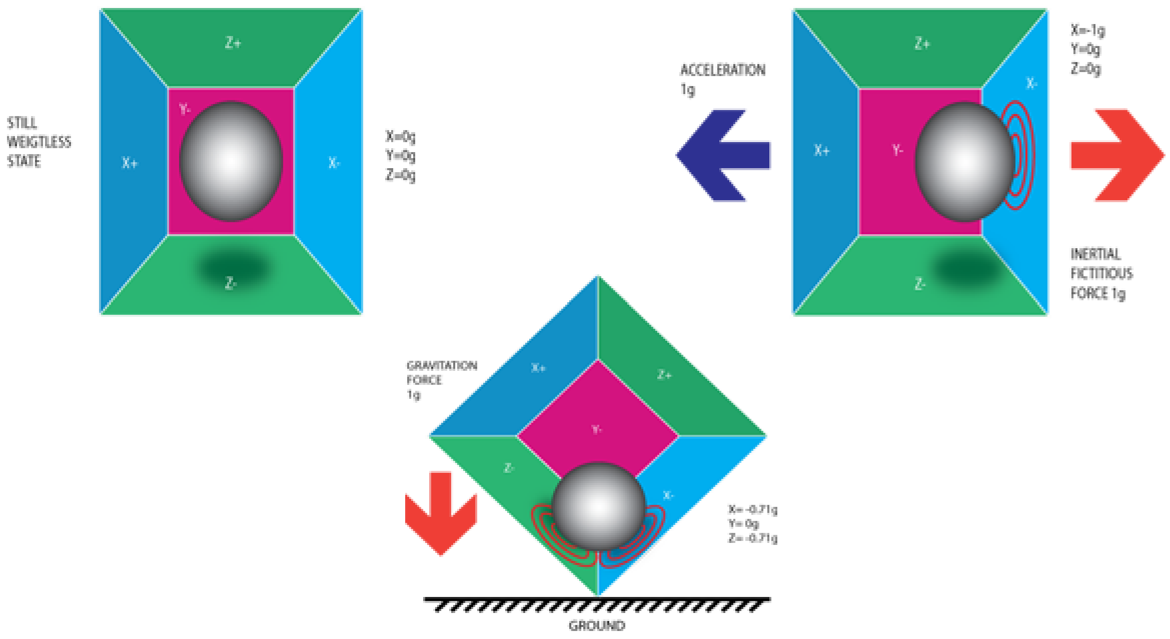


Figure 6. Piezo Electric Accelerometer

An accelerometer works on the principle of the piezoelectric effect. Imagine a cuboidal box with a small ball inside it, like in the picture above. The walls of this box are made with piezoelectric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination due to gravity. The wall that the ball collides with creates tiny piezoelectric currents. There are three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y, and Z axes. Depending on the current produced from the piezoelectric walls, we can determine the direction of inclination and its magnitude.

How Does a Gyroscope Work?

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork-like structure that is in a constant back-and-forth motion. It is held in place using piezoelectric crystals. Whenever you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezoelectric effect, and this current is amplified. The values are then refined by the host microcontroller. Check this short video that explains how a MEMS gyroscope works.

Interfacing the Arduino MPU 6050

The MPU 6050 communicates with the Arduino through the I2C protocol. The MPU 6050 is connected to Arduino as shown in the following diagram. If your MPU 6050 module has a 5V pin, then you can connect it to your Arduino's 5V pin. If not, you will have to connect it to the 3.3V pin. Next, the GND of the Arduino is connected to the GND of the MPU 6050.

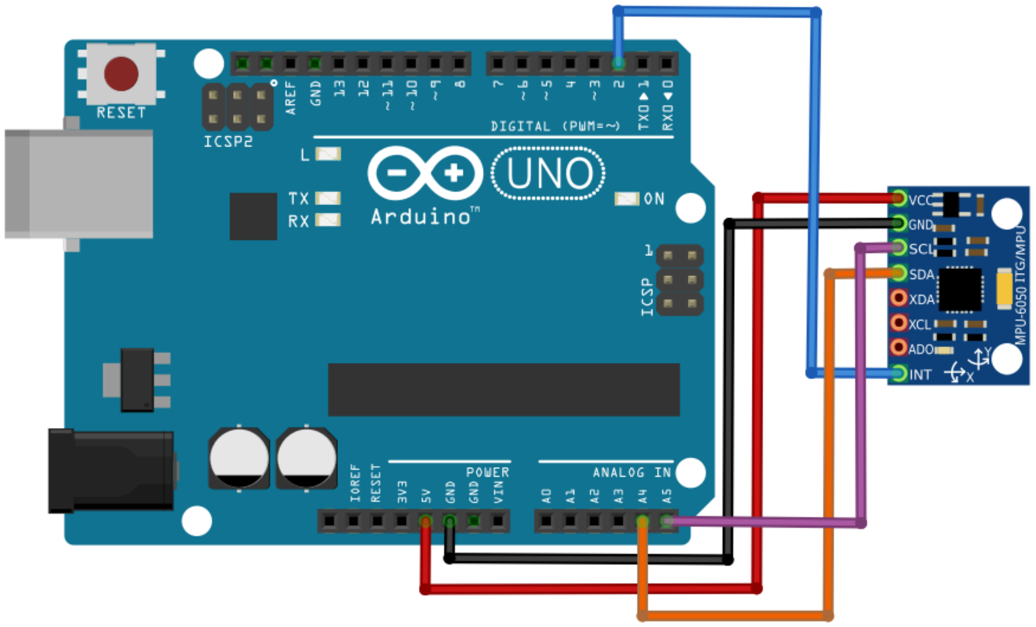


Figure 7. Arduino MPU 6050 connections

The program we will be running here also takes advantage of the Arduino's interrupt pin. Connect your Arduino's digital pin 2 (interrupt pin 0) to the pin labeled as INT on the MPU 6050.

Next, we need to set up the I2C lines. To do this, connect the pin labeled SDA on the MPU 6050 to the Arduino's analog pin 4 (SDA), and the pin labeled as SCL on the MPU 6050 to the Arduino's analog pin 5 (SCL). That's it! You have finished wiring up the Arduino MPU 6050.

Uploading the Code and Testing the Arduino MPU 6050

To test the Arduino MPU 6050, first download the Arduino library for MPU 6050, developed by Jeff Rowberg. You can find the library here. Next, you have to unzip/extract this library, take the folder named "MPU6050", and paste it inside the Arduino's "library" folder. To do this, go to the location where you have installed Arduino (Arduino --> libraries) and paste it inside the libraries folder. You might also have to do the same thing to install the I2Cdev library if you don't already have it for your Arduino. Do the same procedure as above to install it. You can find the file here.

If you have done this correctly, when you open the Arduino IDE, you can see "MPU6050" in File --> Examples. Next, open the example program from File --> Examples --> MPU6050 --> Examples --> MPU6050_DMP6.

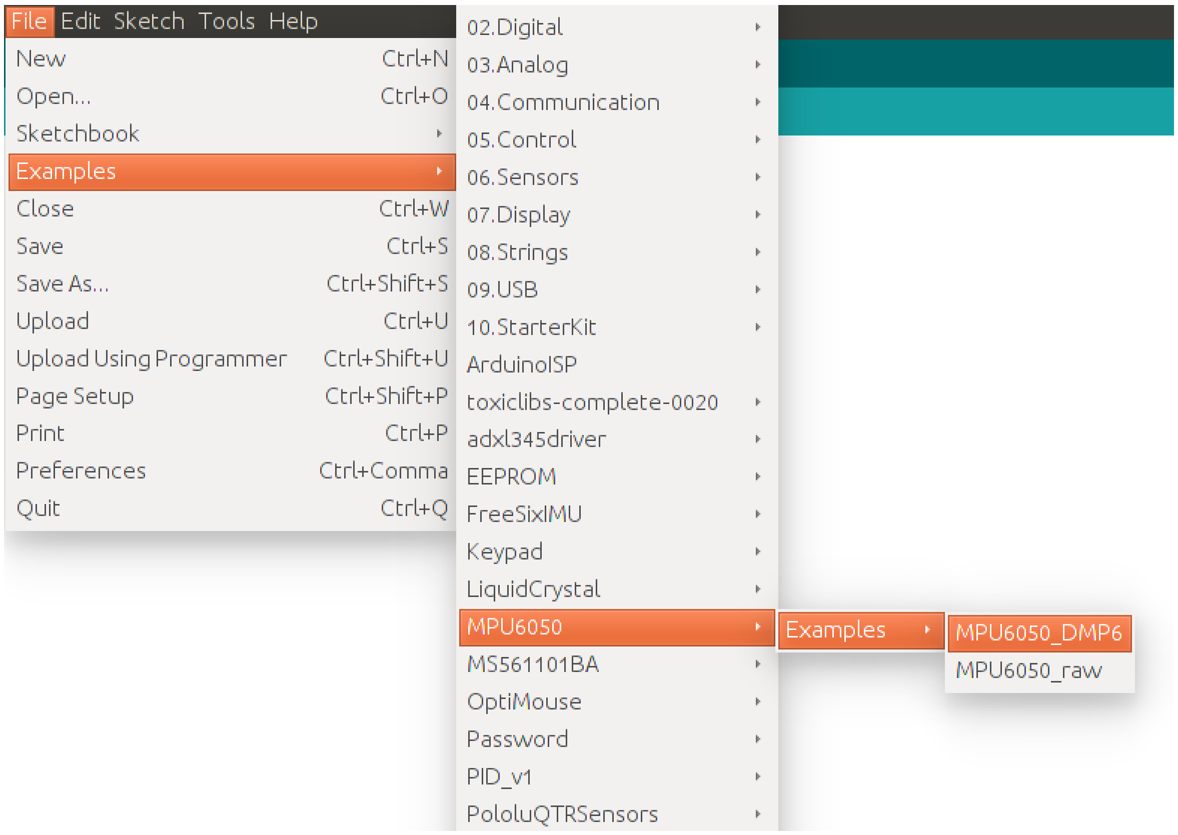
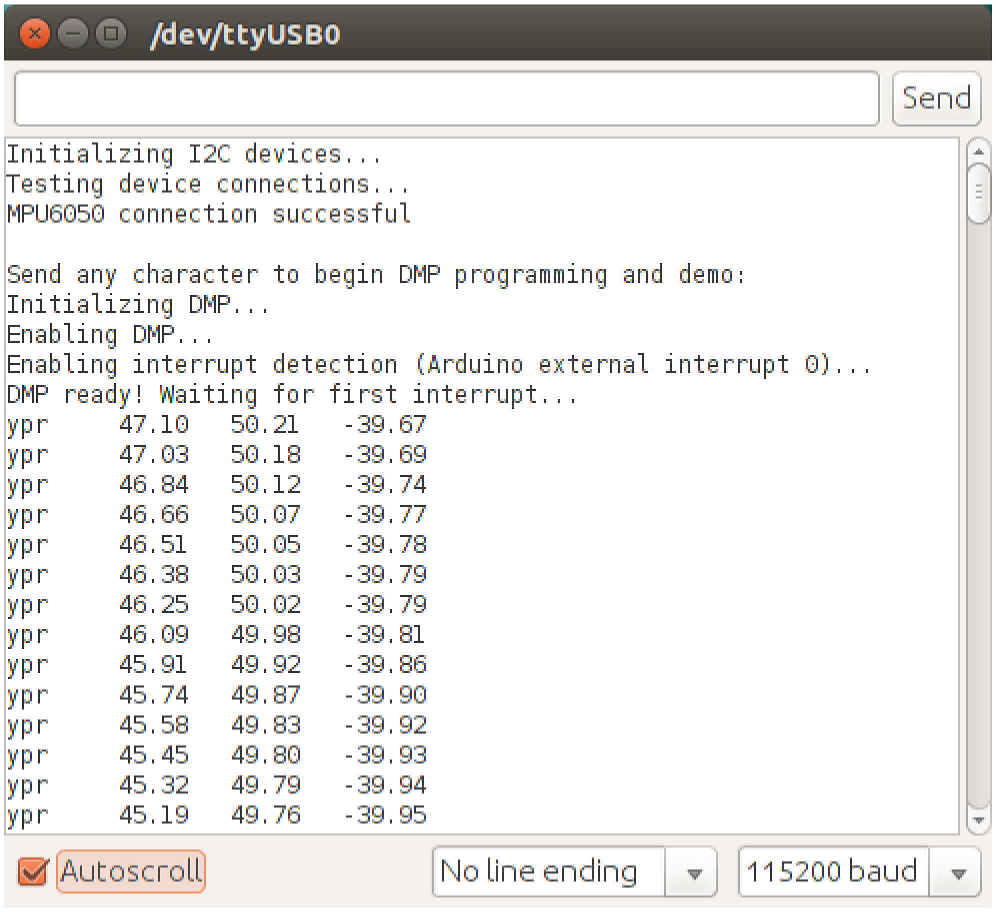


Figure 8. Arduino MPU 6050 DMP code

Next, you have to upload this code to your Arduino. After uploading the code, open up the serial monitor and set the baud rate as 115200. Next, check if you see something like "Initializing I2C devices ..." on the serial monitor. If you don't, just press the reset button. Now, you'll see a line saying, "Send any character to begin DMP programming and demo." Just type in any character on the serial monitor and send it, and you should start seeing the yaw, pitch, and roll values coming in from the MPU 6050.



```
Initializing I2C devices...
Testing device connections...
MPU6050 connection successful

Send any character to begin DMP programming and demo:
Initializing DMP...
Enabling DMP...
Enabling interrupt detection (Arduino external interrupt 0)...
DMP ready! Waiting for first interrupt...
ypr     47.10   50.21   -39.67
ypr     47.03   50.18   -39.69
ypr     46.84   50.12   -39.74
ypr     46.66   50.07   -39.77
ypr     46.51   50.05   -39.78
ypr     46.38   50.03   -39.79
ypr     46.25   50.02   -39.79
ypr     46.09   49.98   -39.81
ypr     45.91   49.92   -39.86
ypr     45.74   49.87   -39.90
ypr     45.58   49.83   -39.92
ypr     45.45   49.80   -39.93
ypr     45.32   49.79   -39.94
ypr     45.19   49.76   -39.95
```

Figure 9. Arduino MPU 6050 Serial Monitor

DMP stands for Digital Motion Processing. The MPU 6050 has a built-in motion processor. It processes the values from the accelerometer and gyroscope to give us accurate 3D values.

Also, you will need to wait about 10 seconds before you get accurate values from the Arduino MPU 6050, after which the values will begin to stabilize.

HC-05

For this tutorial I made two example, controlling the Arduino using a smartphone and controlling the Arduino using a laptop or a PC. In order not to overload this tutorial, in my next tutorial we will learn how we can configure the HC-05 Bluetooth module and make a Bluetooth communication between two separate Arduino Boards as master and slave devices.

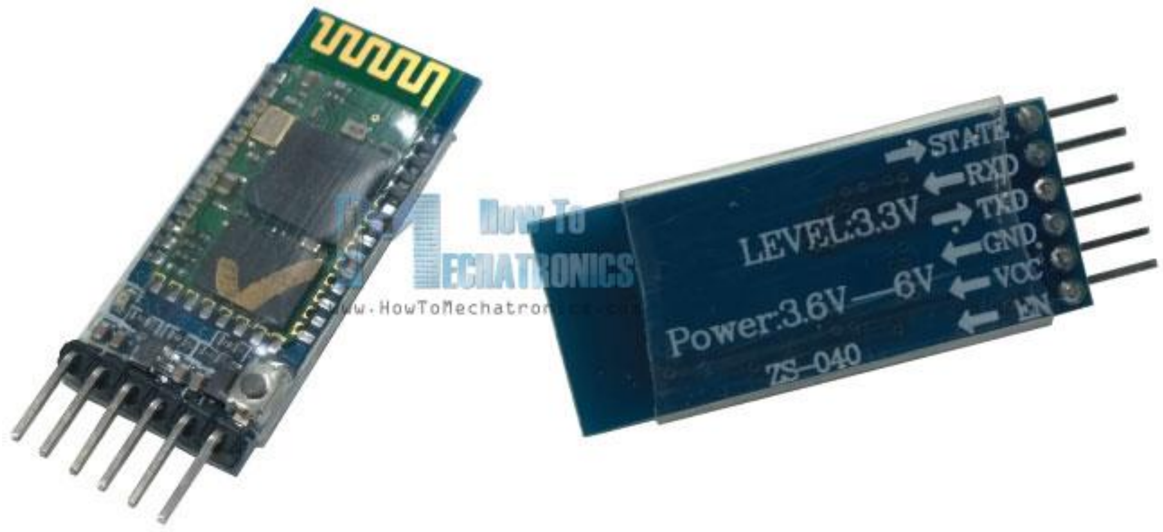


Figure 10. HC-05

Before we start with the first example, controlling an Arduino using a smartphone, let's take a closer look at the HC-05 Bluetooth module. Comparing it to the HC-06 module, which can only be set as a Slave, the HC-05 can be set as Master as well which enables making a communication between two separate Arduino Boards. There are several different versions of this this module but I recommend the one that comes on a breakout board because in that way it's much easier to be connected. The HC-05 module is a Bluetooth SPP (Serial Port Protocol) module, which means it communicates with the Arduino via the Serial Communication.

Circuit Schematics

Here's how we need to connect the module to the Arduino Board.

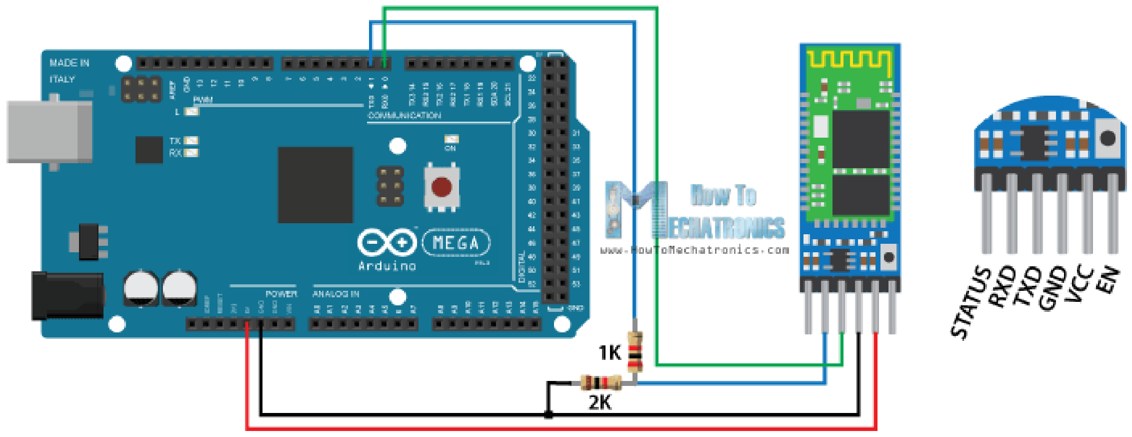


Figure 11. Connections

The particular module that I have can be powered from 3.6 to 6 volts, because it comes on breakout board which contains a voltage regulator. However, the logic voltage level of the data pins is 3.3V. So, the line between the Arduino TX (Transmit Pin, which has 5V output) and the Bluetooth module RX (Receive Pin, which supports only 3.3V) needs to be connected through a voltage divider in order not to burn the module. On the other hand, the line between the Bluetooth module TX pin and the Arduino RX pin can be connected directly because the 3.3V signal from the Bluetooth module is enough to be accepted as a high logic at the Arduino Board.

Arduino Source Code

So, now we are ready to make the Arduino code for enabling the communication between the Arduino board and the smartphone. We will make a simple example, just turning on and off a LED but it will be good enough for understanding the communication.

```
#define ledPin 7
int state = 0;

void setup() {
pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, LOW);
Serial.begin(38400); // Default communication rate of the Bluetooth module
}

void loop() {
if(Serial.available() > 0){ // Checks whether data is comming from the serial port
state = Serial.read(); // Reads the data from the serial port
}

if (state == '0') {
digitalWrite(ledPin, LOW); // Turn LED OFF
Serial.println("LED: OFF"); // Send back, to the phone, the String "LED: ON"
state = 0;
}
else if (state == '1') {
digitalWrite(ledPin, HIGH);
Serial.println("LED: ON");;
state = 0;
}
}
```

Description: First we need to define the pin to which our LED will be connected and a variable in which we will store the data coming from the smartphone. In the setup section we need to define the LED pin as output and set it low right away. As mention previously, we will use the serial communication so we need to begin the serial communication at 38400 baud rate, which is the default baud rate of the Bluetooth module. In the loop section with the Serial.available() function we will check whether there is available data in the serial port to be read. This means that when we will send data to the Bluetooth module this statement will be true so then using the Serial.read() function we will read that data and put it into the "state" variable. So if the Arduino receive the character '0' it will turn the LED off and using the Serial.println() function it will send back to the smartphone, via the serial port, the String "LED: OFF". Additionally we will reset the "state" variable to 0 so that the two above lines will be executed only once. Note here that the "state" variable is integer, so when we receive the character '0' that comes from smartphone, the actual value of the integer "state" variable is 48, which corresponds to character '0', according to the ASCII table.. That's why in the "if" statement we are comparing the "state" variable to a character '0'. On the other hand, if the received character is '1', the LED will light up and the String "LED: ON" will be sent back.

Now the code is ready to be uploaded but in order to do that we need to unplug the TX and RX lines because when uploading the Arduino uses the serial communication so the pins RX (digital pin 0) and TX (digital pin1) are busy. We can avoid this step if we use the other TX and RX pins of the Arduino Board, but in that case we will have to use the SoftwareSerial.h library for the serial communication.

Connecting the Smartphone to the HC-05 Bluetooth Module and the Arduino

Now we are ready to connect the smartphone to the Bluetooth module and the Arduino. What we need to do here is to activate the Bluetooth and the smartphone will find the HC-05 Bluetooth module.

Then we need to pair the devices and the default password of the HC-05 module is 1234. After we have paired the devices we need an application for controlling the Arduino. There are many application in the Play Store for this purpose which will work with the Arduino code that we wrote. However, we can make our own custom application for this tutorial using the MIT App Inventor online application. This is a great and easy to use application for building Android application and in my next tutorial you can find a detailed step by step guide how to build your own custom Android application for your Arduino Project.

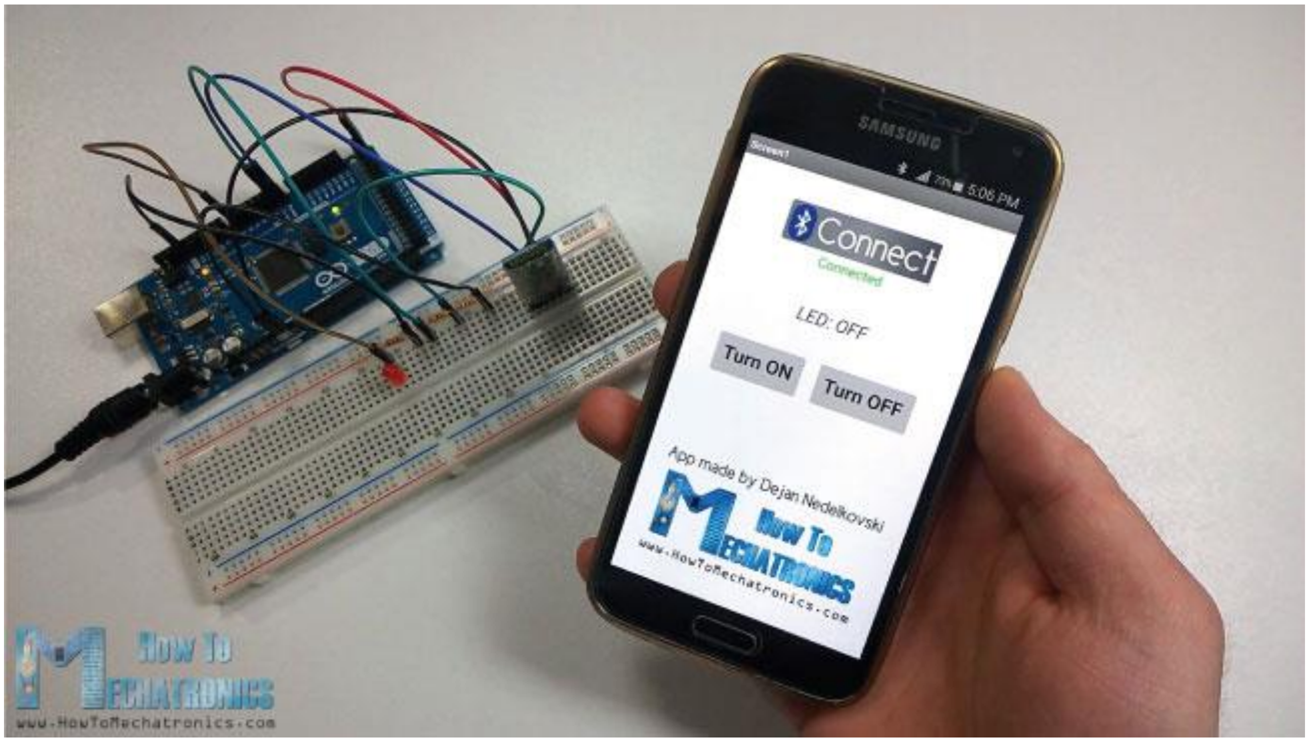


Figure 12. Connecting with Bluetooth app

Link for Bluetooth app:

https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/#

With the connect button we will connect the smartphone to the Bluetooth module and the status text below the button will tell us whether we have successfully connected. Using the “Turn ON” and “Turn OFF” buttons we can turn on and off the LED. The text above the buttons is the one that the Arduino is sending back to the smartphone when a particular button is pressed.

LOAD CELL

Load cell is a sensor or a transducer that converts a load or force acting on it into an electronic signal. This electronic signal can be a voltage change, current change or frequency change depending on the type of load cell and circuitry used.

There are many different kinds of load cells.

Resistive load cells work on the principle of piezo-resistivity. When a load/force/stress is applied to the sensor, it changes its resistance. This change in resistance leads to a change in output voltage when a input voltage is applied.

Capacitive load cells work on the principle of change of capacitance which is the ability of a system to hold a certain amount of charge when a voltage is applied to it. For common parallel plate capacitors, the capacitance is directly proportional to the amount of overlap of the plates and the dielectric between the plates and inversely proportional to the gap between the plates.

How does a resistive load cell works ?

A load cell is made by using an elastic member (with very highly repeatable deflection pattern) to which a number of strain gauges are attached.

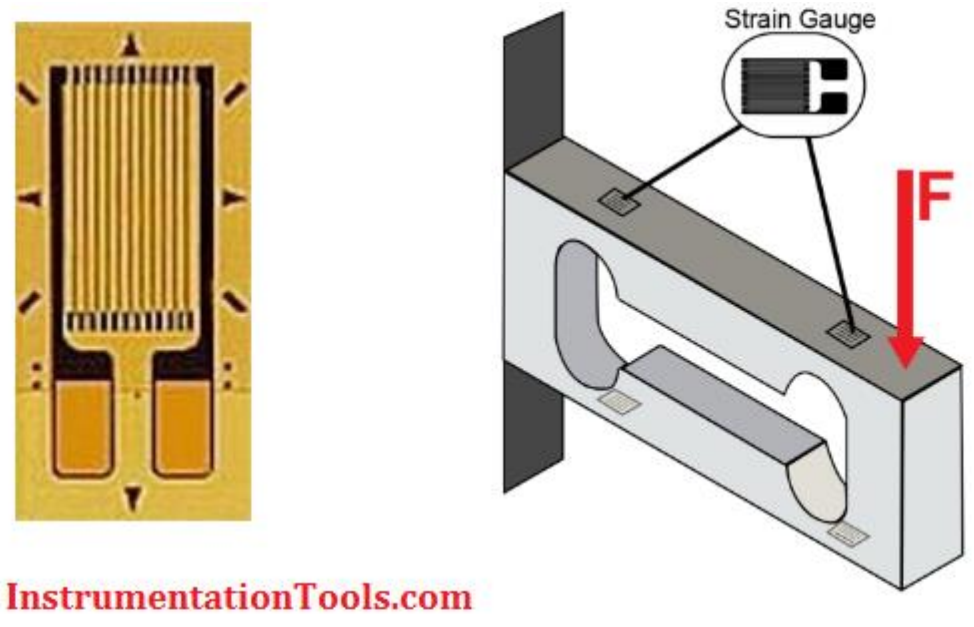


Figure 13. Resistive Load cell Principle

In this particular load cell shown in above figure, there are a total of four strain gauges that are bonded to the upper and lower surfaces of the load cell.

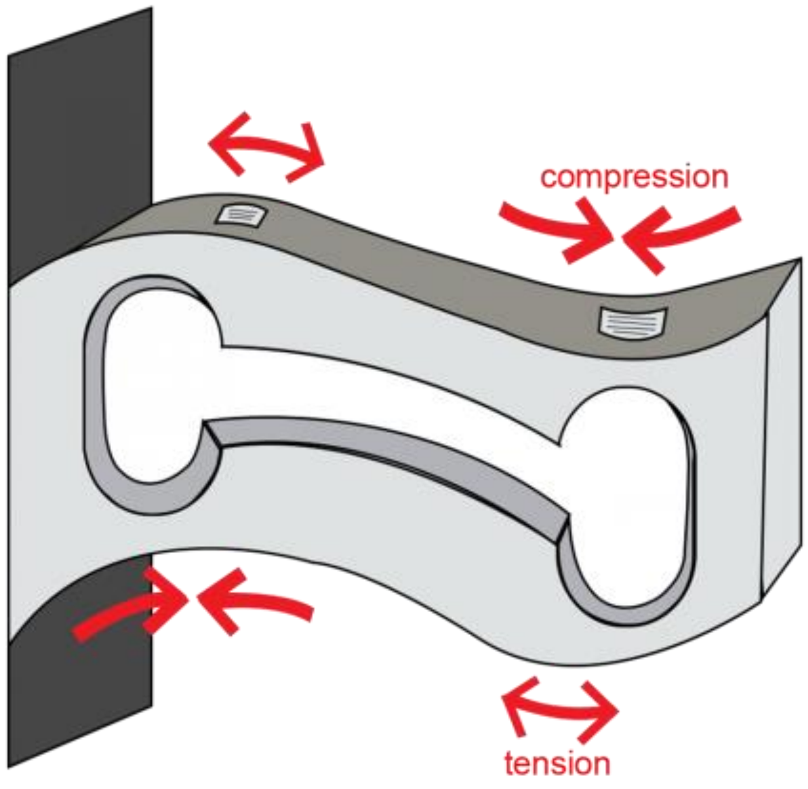


Figure 14. Effect of Load Application

When the load is applied to the body of a resistive load cell as shown above, the elastic member, deflects as shown and creates a strain at those locations due to the stress applied. As a result, two of the strain gauges are in compression, whereas the other two are in tension as shown in below animation.

During a measurement, weight acts on the load cell's **metal spring element** and causes **elastic deformation**.

This strain (positive or negative) is converted into an electrical signal by a **strain gauge (SG)** installed on the spring element. The simplest type of load cell is a bending beam with a strain gauge.

We use wheatstone bridge circuit to convert this change in strain/resistance into voltage which is proportional to the load.

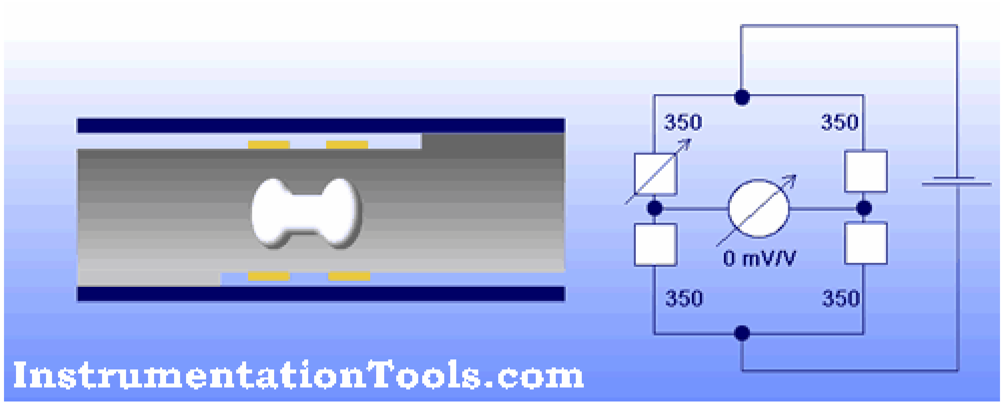


Figure 15. Using of Wheatstone Bridge

Wheatstone Bridge Circuit

The four strain gauges are configured in a Wheatstone Bridge configuration with four separate resistors connected as shown in what is called a Wheatstone Bridge Network. An excitation voltage – usually 10V is applied to one set of corners and the voltage difference is measured between the other two corners. At equilibrium with no applied load, the voltage output is zero or very close to zero when the four resistors are closely matched in value. That is why it is referred to as a balanced bridge circuit.

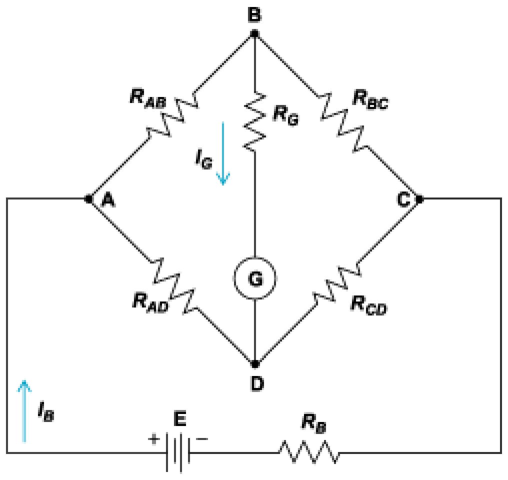


Figure 16. Wheatstone Bridge

When the metallic member to which the strain gauges are attached, is stressed by the application of a force, the resulting strain – leads to a change in resistance in one (or more) of the resistors. This change in resistance results in a change in output voltage. This small change in output voltage (usually about 20 mVolt of total change in response to full load) can be measured and digitized after careful amplification of the small milli-volt level signals to a higher amplitude 0-5V or 0-10V signal.

These load cells have been in use for many decades now, and can provide very accurate readings but require many tedious steps during the manufacturing process

SHARP IR

IR Sensors work by using a specific light sensor to detect a select light wavelength in the Infra-Red (IR) spectrum. By using an LED which produces light at the same wavelength as what the sensor is looking for, you can look at the intensity of the received light. When an object is close to the sensor, the light from the LED bounces off the object and into the light sensor. This results in a large jump in the intensity, which we already know can be detected using a threshold.

Since the sensor works by looking for reflected light, it is possible to have a sensor that can return the value of the reflected light. This type of sensor can then be used to measure how "bright" the object is. This is useful for tasks like line tracking.

The Circuit

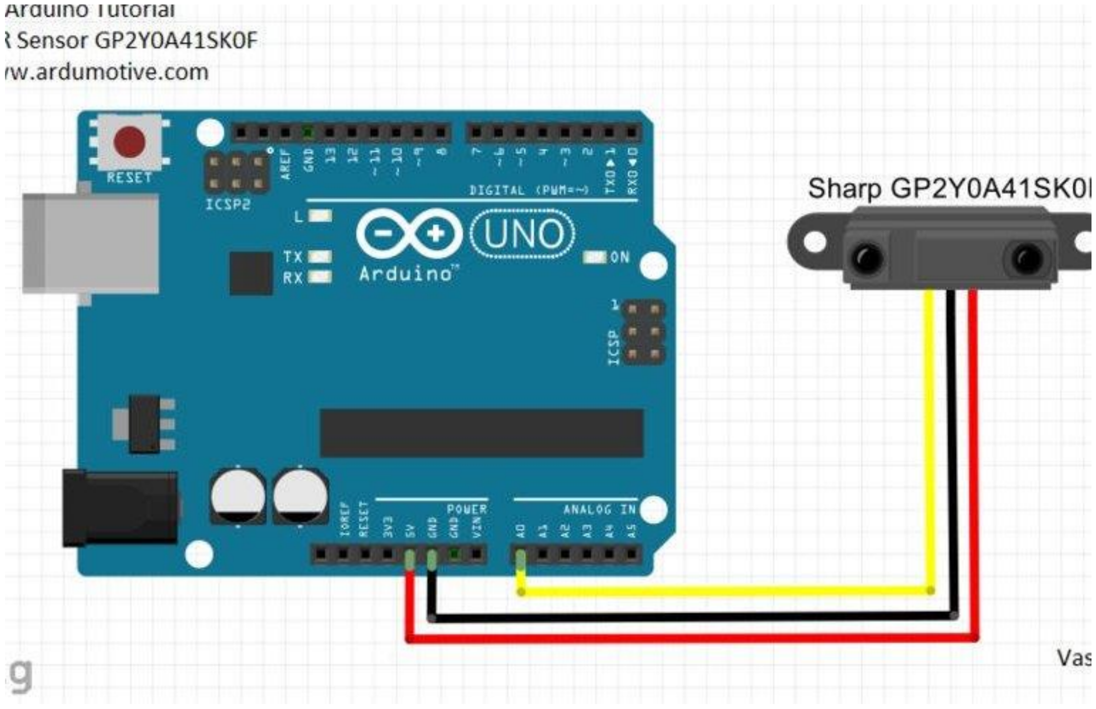


Figure 17. Connections

The connections are pretty easy, see the above image with the breadboard circuit schematic.

CODE

```
#define sensor A0 // Sharp IR GP2Y0A41SK0F (4-30cm, analog)


void setup() {
  Serial.begin(9600); // start the serial port
}
void loop()
{
  // 5v

  float volts = analogRead(sensor)*0.0048828125;  // value from sensor *
(5/1024)

  int distance = 13*pow(volts, -1); // worked out from datasheet graph

  delay(1000); // slow down serial port



  if (distance <= 30){

    Serial.println(distance);   // print the distance

  }

}
```

Now, we can view our output in the serial monitor.