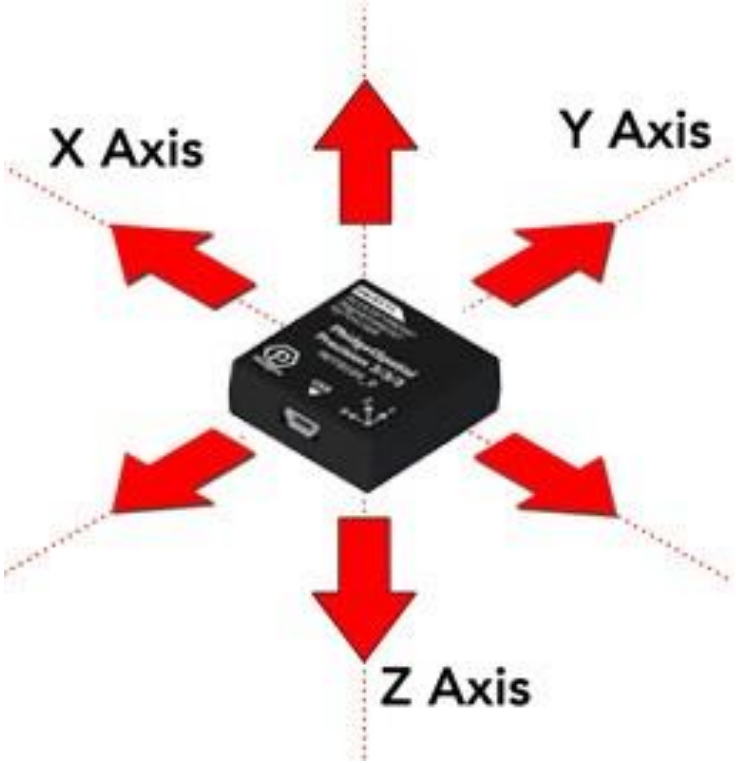
LECTURE 10
Accelerometer Sensor and Application

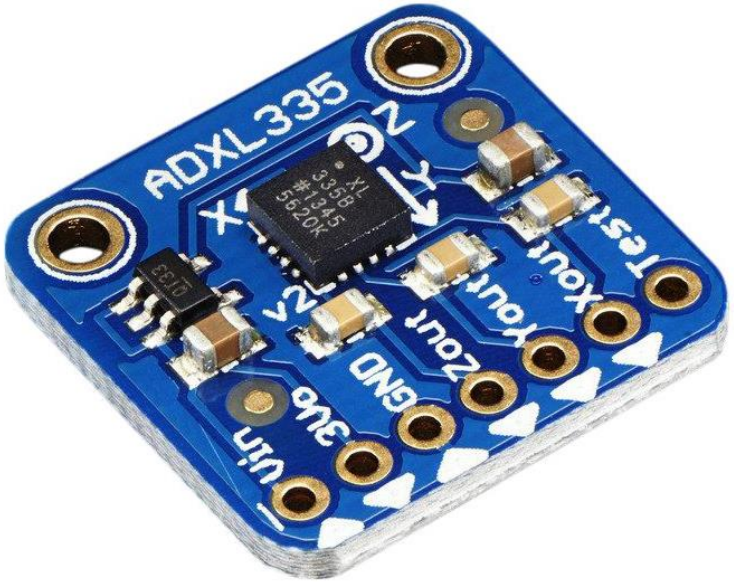
Instructor: Osman Gul, Ph.D. Candidate
Department of Mechanical Engineering
Korea Advanced Institute of Science and Technology (KAIST)

Syllabus:

Week	Date& Time*	Content
1		No Class (Course Add/Drop Period)
2	March 6 (Mon) 20-22	Orientation and Course Overview
3	March 13 (Mon) 20-22	Sensor Fundamentals
4	March 20 (Mon) 20-22	Arduino Fundamentals
5	March 27 (Mon) 20-22	Arduino Programming and Applications
6	April 3 (Mon) 20-22	Force Sensor and Application: Control Multiple LEDs
7	April 10 (Mon) 20-22	Light Sensor and Application: Solar Tracker
8	April 17	No Class (Midterm)
9	April 24 (Mon) 20-22	Humidity and Temperature Sensor and Application: Temperature Controlled Fan
10	May 1 (Mon) 20-22	Gas Sensor and Application: Smoke Detector
11	May 8 (Mon) 20-22	Sound Sensor and Application: Control LED by Clapping
12	May 15 (Mon) 20-22	Accelerometer Sensor and Application: Ping Pong Game
13	May 22 (Mon) 20-22	Ultrasonic Sensor and Application: Flappy Bird Game
14	May 29 (Mon) 20-22	Course Wrap-up
15	June 5	No Class (Final)
16	June 12	No Class (Final)

Accelerometer Sensor



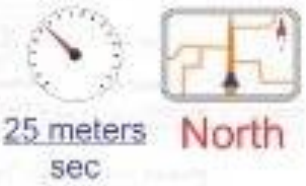


Velocity Vs. Speed

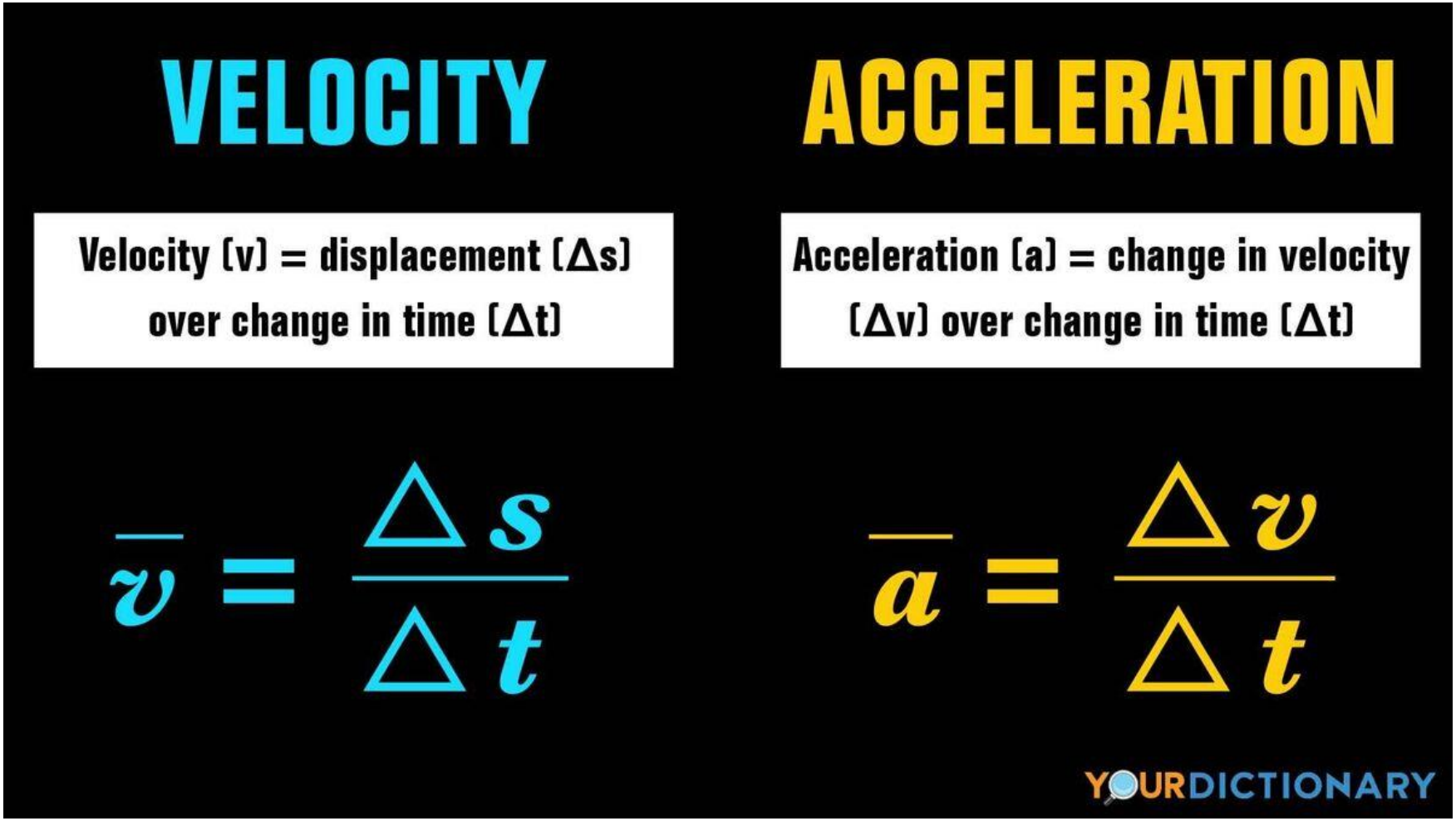
Velocity:
Velocity is the vector quantity that signifies the magnitude of the rate of change of position and also the direction of an object's movement.

Speed:
Speed is the scalar quantity that Signifies only the magnitude of the rate of change of an object's movement.

Example:



Example:


VELOCITY
ACCELERATION
Velocity (v) = displacement (Δs) over change in time (Δt)
Acceleration (a) = change in velocity (Δv) over change in time (Δt)
$\overline{v} = \frac{\Delta s}{\Delta t}$
$\overline{a} = \frac{\Delta v}{\Delta t}$
YOURDICTIONARY

- An **accelerometer** is an electronic sensor that measures the **acceleration forces** acting on an object, in order to determine the **object's position** in space and monitor the **object's movement**.

- Acceleration, which is a vector quantity, is the **rate of change of an object's velocity** (velocity being the displacement of the object divided by the change in time).

- There are two types of acceleration forces**: static forces** and **dynamic forces**.

- Static forces are forces that are **constantly** being applied to the object (such as friction or gravity). Dynamic forces are **"moving" forces** applied to the object at various rates (such as vibration, or the force exerted on a cue ball in a game of pool). This is why accelerometers are used in automobile collision safety systems, for example. When a car is acted on by a powerful dynamic force, the accelerometer (sensing a rapid deceleration) sends an electronic signal to an embedded computer, which in turn deploys the airbags.

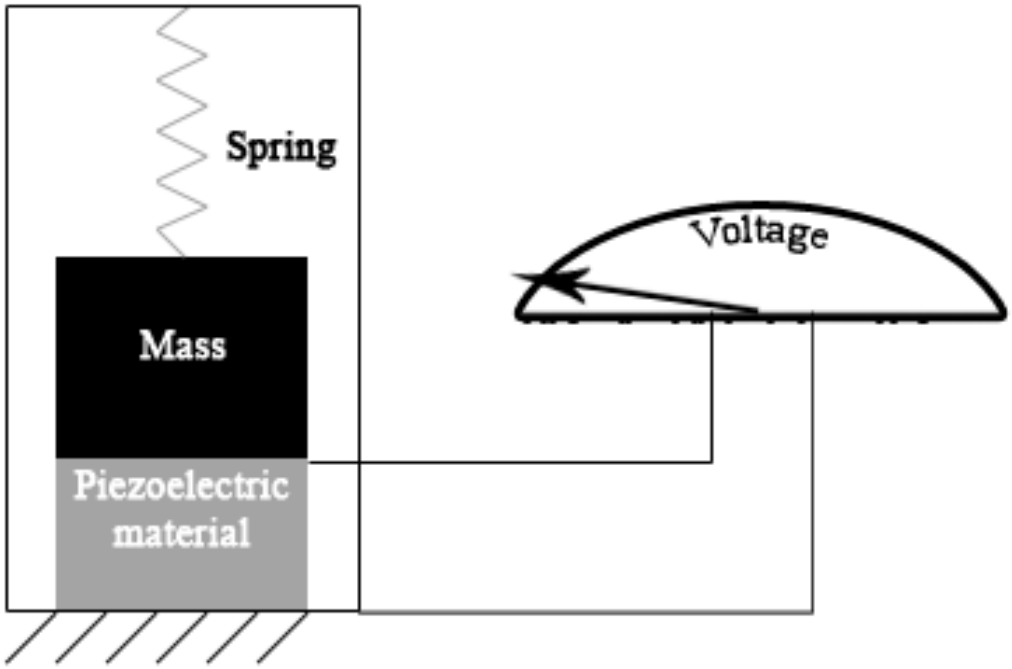
There are three different types of accelerometers, and they are each designed to efficiently function in their intended environments. The three types are: piezoelectric, piezoresistance and capacitive.

- A piezoelectric accelerometer utilizes the **piezoelectric effect** (piezoelectric materials produce electricity when put under physical stress) to sense change in acceleration. Piezoelectric accelerometers are most commonly used in vibration and shock measurement.

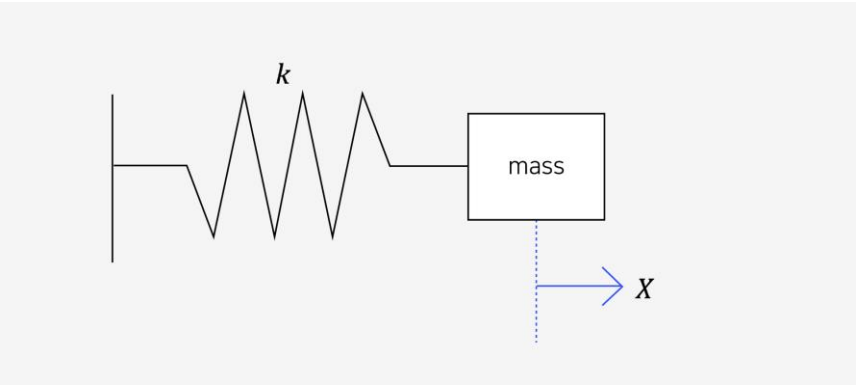
- Piezoresistance accelerometers are much less sensitive than piezoelectric accelerometers, and they are better suited to vehicle crash testing. A piezoresistance accelerometer increases its resistance in proportion to the amount of pressure applied to it.

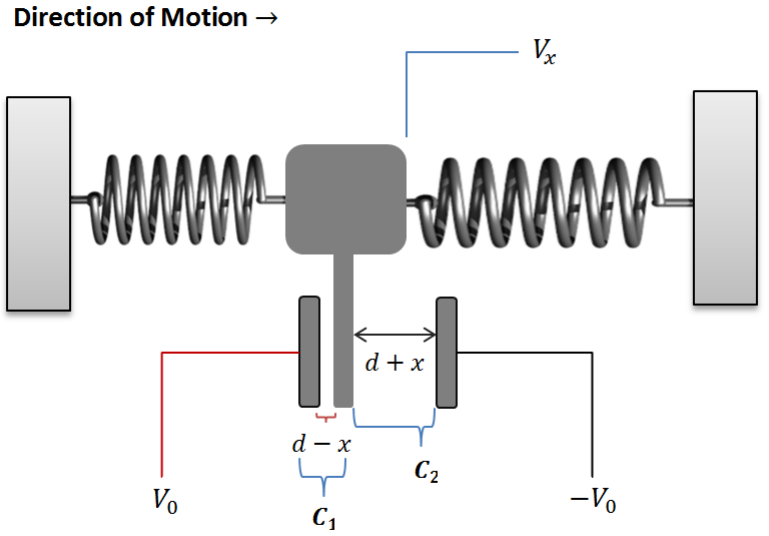
- The third and most commonly used type of accelerometer is the capacitive accelerometer. Capacitive accelerometers use change **in electrical capacitance** to determine an object's acceleration. When the sensor undergoes acceleration, the distance between its capacitor plates changes as the diaphragm of the sensor moves.

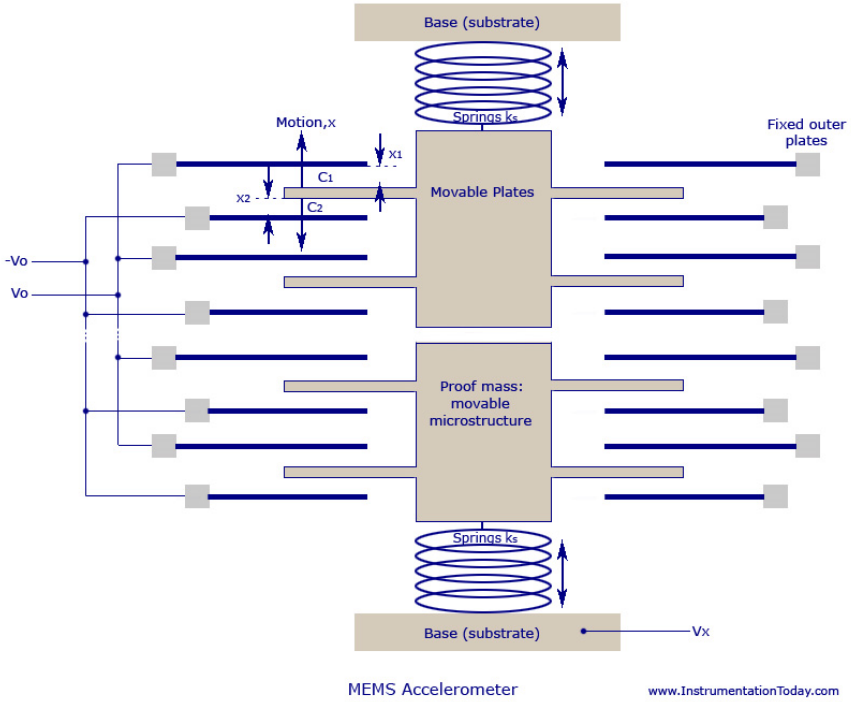
An example of the inside of a piezoelectric accelerometer

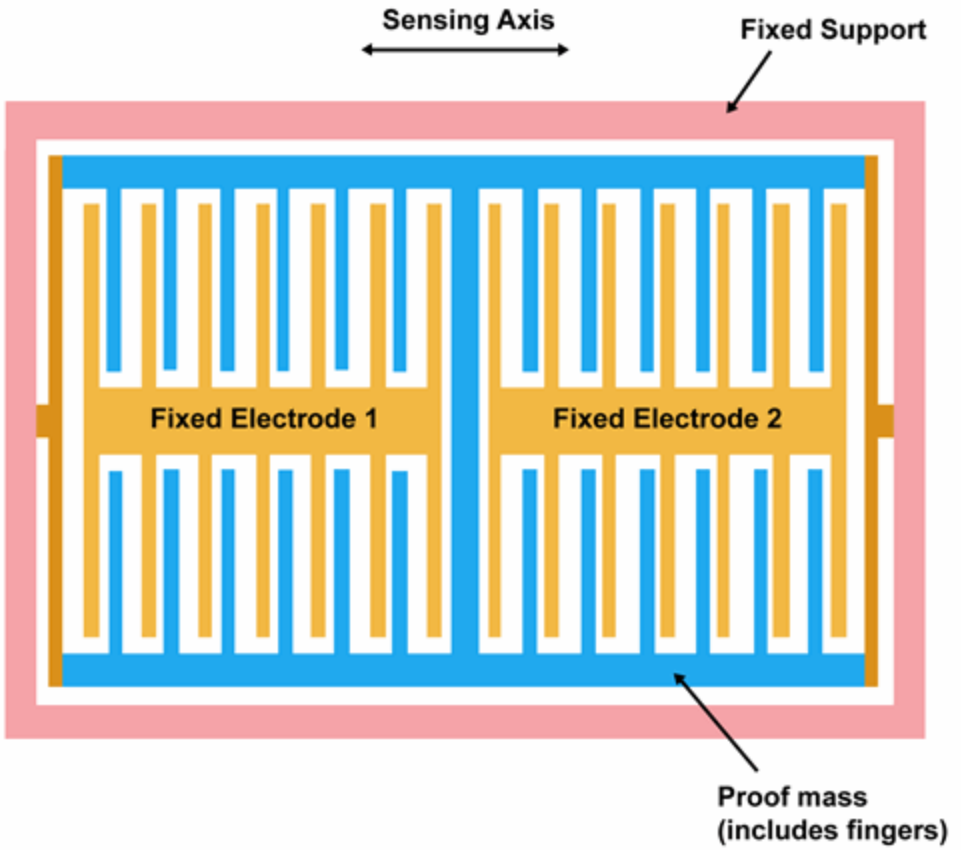


- All accelerometers work on the principle of a mass on a spring, when the thing they are attached to accelerates then the mass wants to remain stationary due to its inertia and therefore the spring is stretched or compressed, creating a force which is detected and corresponds to the applied acceleration.

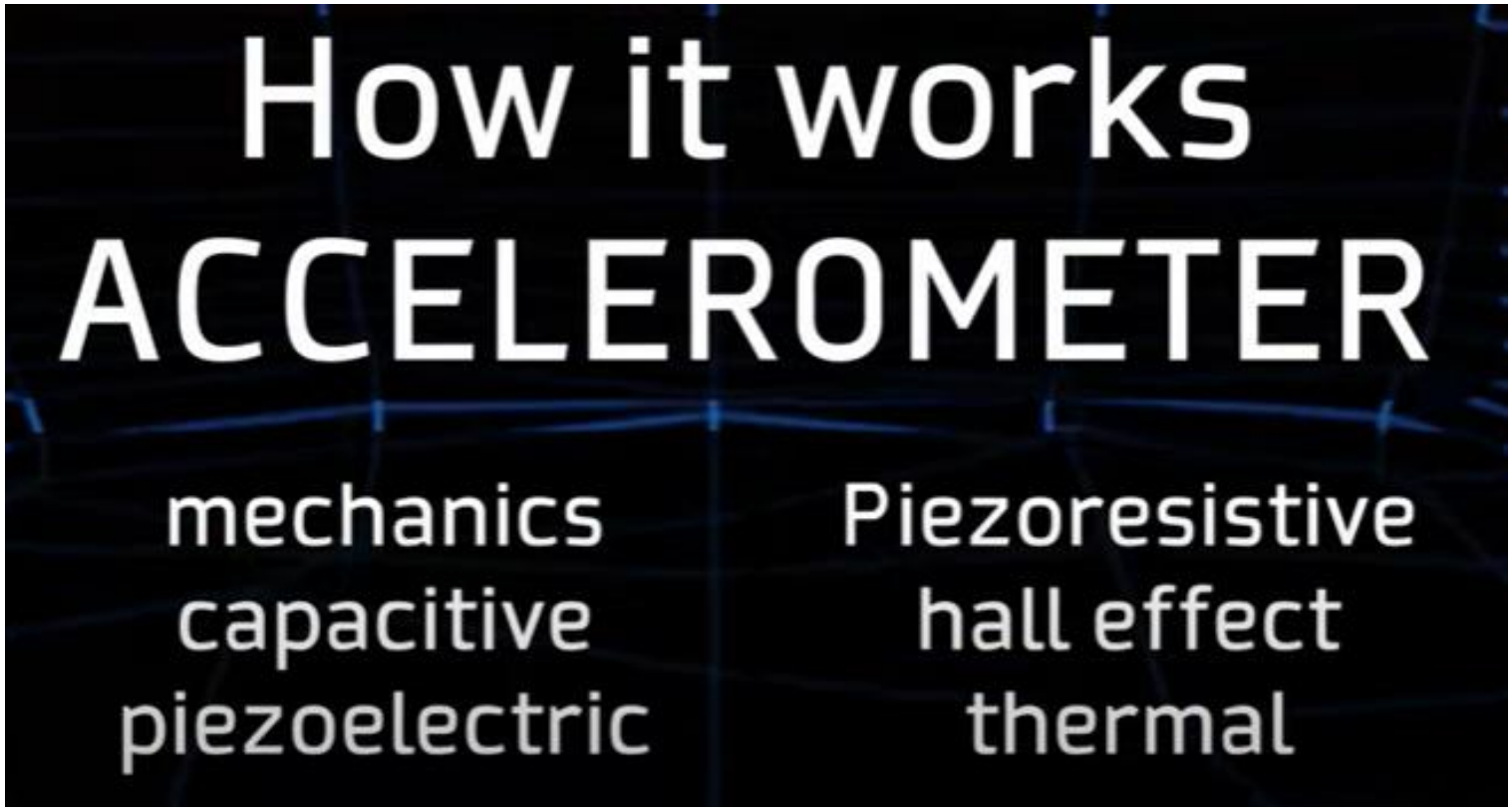




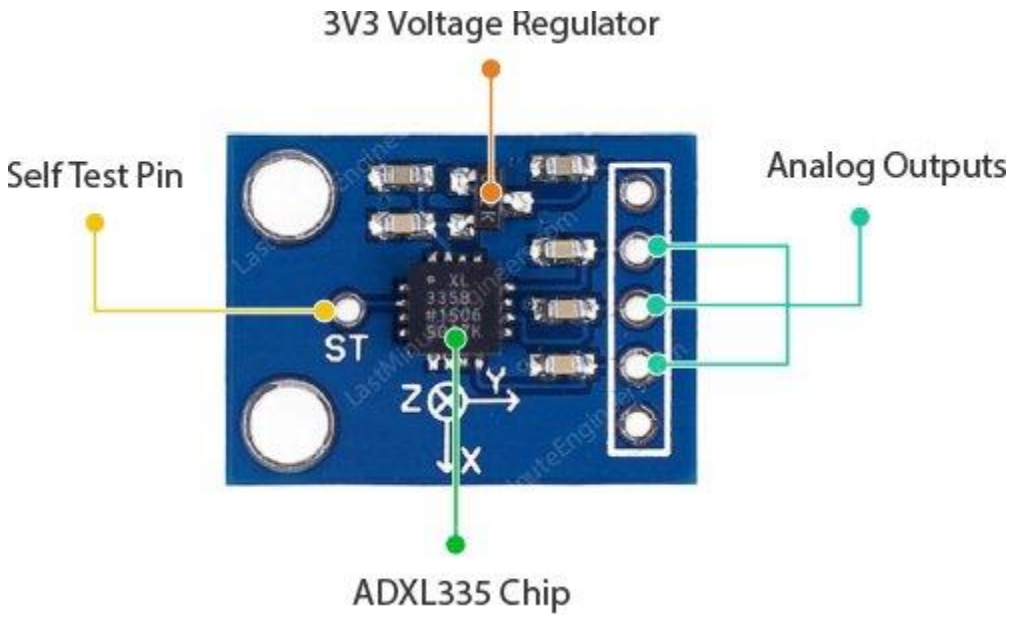


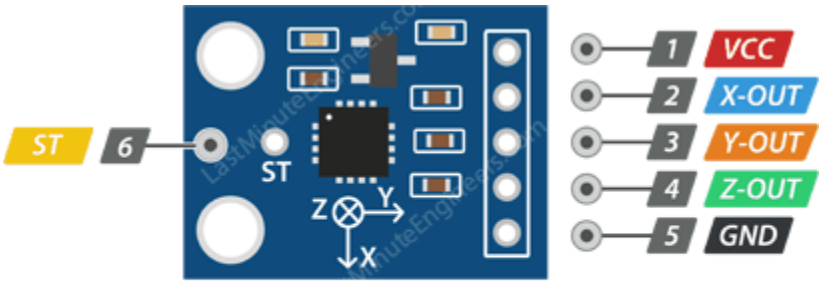


As a result of deflection, the capacitance between fixed plates and plates attached to the suspended structure changes. This change in capacitance is proportional to the acceleration along that axis.



https://www.youtube.com/watch?v=XhBHp8tUWPQ


3V3 Voltage Regulator
Self Test Pin
Analog Outputs
ST
Z
Y
X
ADXL335 Chip



ADXL335 Pinout

Last Minute ENGINEERS.com

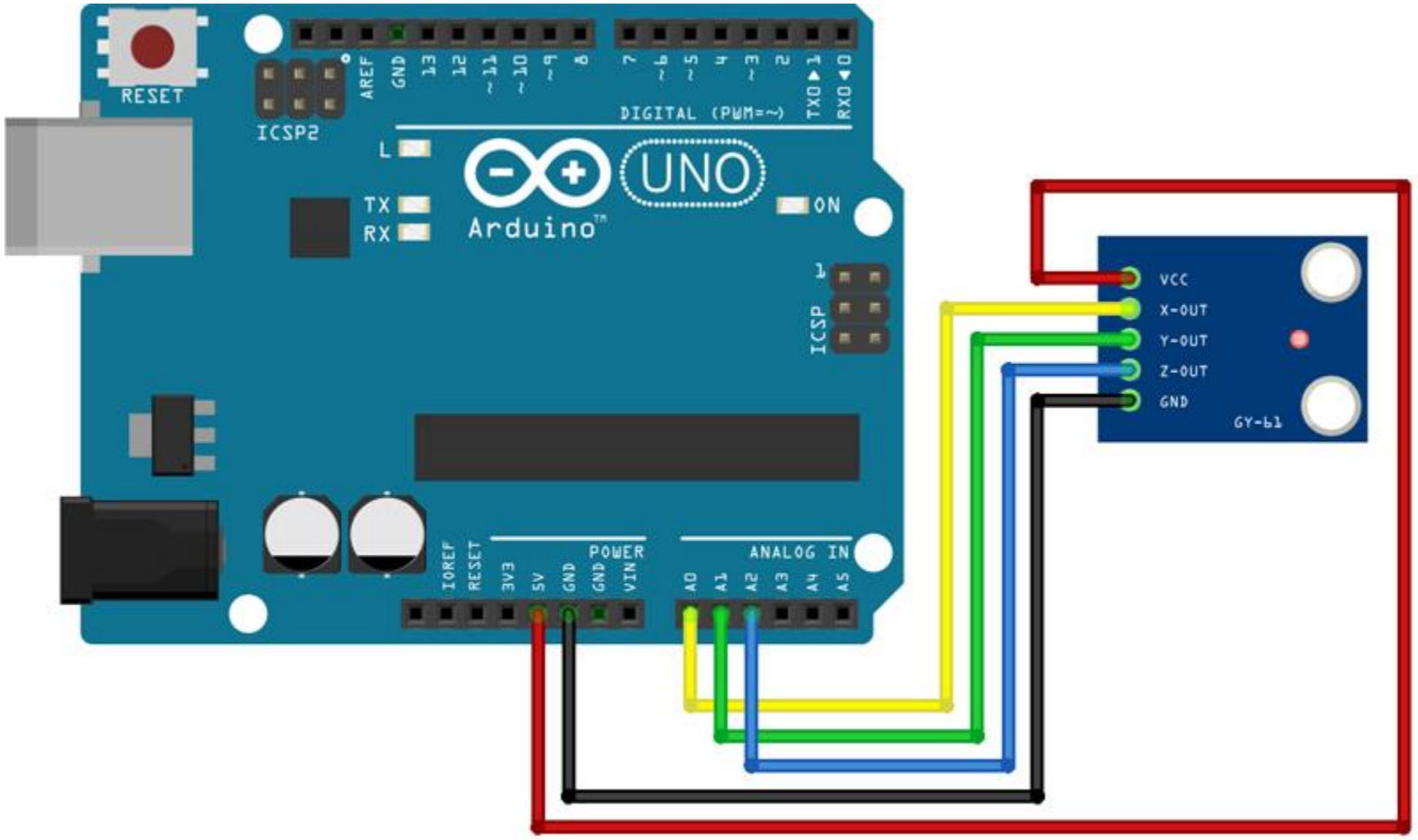
VCC supplies power to the module. Connect it to the 5V output of your Arduino.

X-Out outputs an analog voltage proportional to acceleration along the X axis.

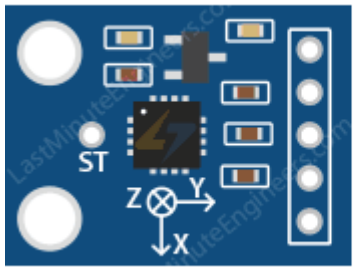
Y-Out outputs an analog voltage proportional to acceleration along the Y axis.

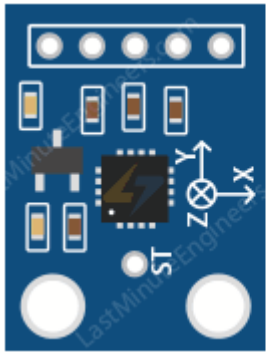
Z-Out outputs analog voltage proportional to acceleration along the Z axis.

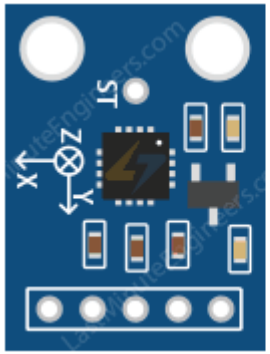
GND is the ground pin.

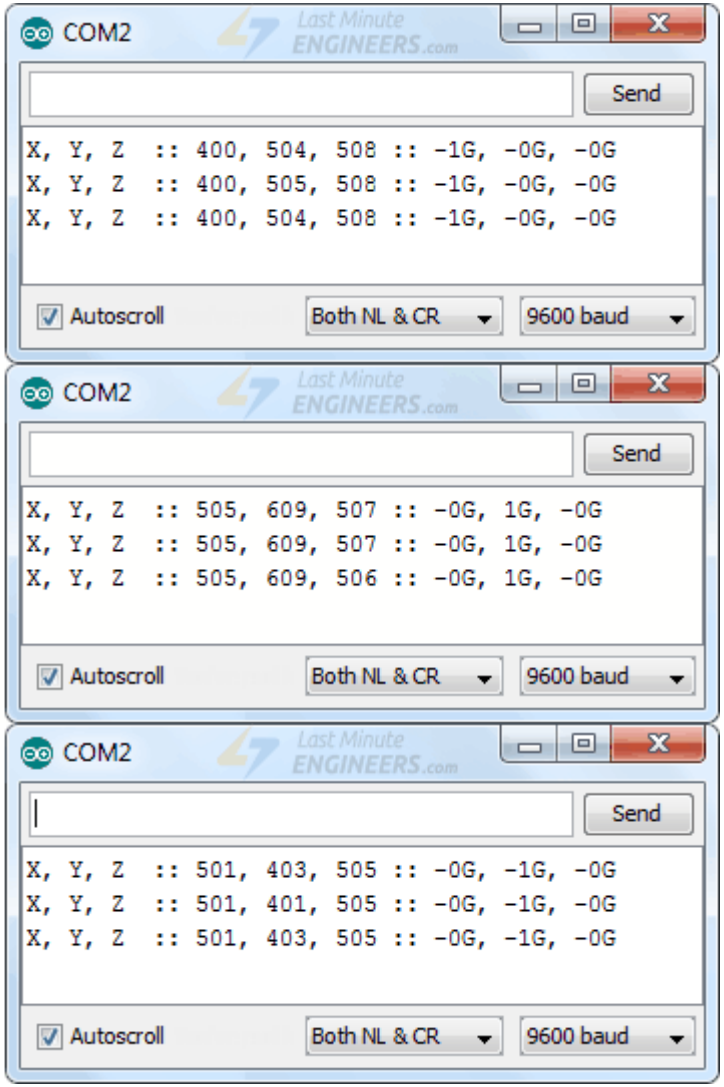

RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5
VCC
X-OUT
Y-OUT
Z-OUT
GND
GY-61

```
const int xInput = A0;
const int yInput = A1;
const int zInput = A2;

// initialize minimum and maximum Raw Ranges for each axis
int RawMin = 0;
int RawMax = 1023;

// Take multiple samples to reduce noise
const int sampleSize = 10;

void setup()
{
		analogReference(EXTERNAL);
		Serial.begin(9600);
}

void loop()
{
		//Read raw values
		int xRaw = ReadAxis(xInput);
		int yRaw = ReadAxis(yInput);
		int zRaw = ReadAxis(zInput);

		// Convert raw values to 'milli-Gs"
		long xScaled = map(xRaw, RawMin, RawMax, -3000, 3000);
		long yScaled = map(yRaw, RawMin, RawMax, -3000, 3000);
		long zScaled = map(zRaw, RawMin, RawMax, -3000, 3000);

		// re-scale to fractional Gs
		float xAccel = xScaled / 1000.0;
		float yAccel = yScaled / 1000.0;
		float zAccel = zScaled / 1000.0;

		Serial.print("X, Y, Z  :: ");
		Serial.print(xRaw);
		Serial.print(", ");
		Serial.print(yRaw);
		Serial.print(", ");
		Serial.print(zRaw);
		Serial.print(" :: ");
		Serial.print(xAccel,0);
		Serial.print("G, ");
		Serial.print(yAccel,0);
		Serial.print("G, ");
		Serial.print(zAccel,0);
		Serial.println("G");

		delay(200);
}

// Take samples and return the average
int ReadAxis(int axisPin)
{
		long reading = 0;
		analogRead(axisPin);
		delay(1);
		for (int i = 0; i < sampleSize; i++)
		{
		reading += analogRead(axisPin);
		}
		return reading/sampleSize;
}
```


ST
Z
Y
X


Y
X
Z
ST


ST
Z
X
Y


COM2
Send
X, Y, Z :: 400, 504, 508 :: -1G, -0G, -0G
X, Y, Z :: 400, 505, 508 :: -1G, -0G, -0G
X, Y, Z :: 400, 504, 508 :: -1G, -0G, -0G
Autoscroll
Both NL & CR
9600 baud
COM2
Send
X, Y, Z :: 505, 609, 507 :: -0G, 1G, -0G
X, Y, Z :: 505, 609, 507 :: -0G, 1G, -0G
X, Y, Z :: 505, 609, 506 :: -0G, 1G, -0G
Autoscroll
Both NL & CR
9600 baud
COM2
Send
X, Y, Z :: 501, 403, 505 :: -0G, -1G, -0G
X, Y, Z :: 501, 401, 505 :: -0G, -1G, -0G
X, Y, Z :: 501, 403, 505 :: -0G, -1G, -0G
Autoscroll
Both NL & CR
9600 baud

- Because the Arduino has a 10-bit ADC (2^{10} = 1024), it will map the output voltages of the ADXL335, which range from 0 to 3.3 volts, into integer values between 0 and 1023. That is why RawMin is set to 0 and RawMax is set to 1023.

```
// initialize minimum and maximum Raw Ranges for each axis
int RawMin = 0;
int RawMax = 1023;
```

- The sampleSize variable specifies the number of samples that should be taken by the Arduino during each conversion. In our case, we set sampleSize to 10 to achieve more accurate results.

```
// Take multiple samples to reduce noise
const int sampleSize = 10;
```

- In the setup section, we first set the analog reference to EXTERNAL by calling analogReference(EXTERNAL). We then initiate serial communications with the PC.

```
analogReference(EXTERNAL);
Serial.begin(9600);
```

- In the loop section, we read analog outputs from the sensor every 200ms. Note that we are calling the ReadAxis() custom function instead of the analogRead() function. This function simply takes ten ADC conversion samples and returns the average.

```
//Read raw values
int xRaw = ReadAxis(xInput);
int yRaw = ReadAxis(yInput);
int zRaw = ReadAxis(zInput);
```

Converting ADXL335 Output to Acceleration(g)

- The mapping is handled by the IDE's built-in map() function. When we call map(xRaw, RawMin, RawMax, -3000, 3000), the value of RawMin is mapped to -3000, the value of RawMax is mapped to 3000, and values in-between are mapped to values in-between.

For example,

- If the sensor outputs 0 volts on x-axis, that is xRaw=0, the map() function will return -3000, which corresponds to -3g.
- If the sensor outputs 1.65 volts on x-axis, that is xRaw=511, the map() function will return 0, which corresponds to 0g.
- If the sensor outputs 3.3 volts on x-axis, that is xRaw=1023, the map() function will return 3000, which corresponds to +3g.

The term Ratiometric makes more sense now that the output voltage increases linearly with acceleration across the range.

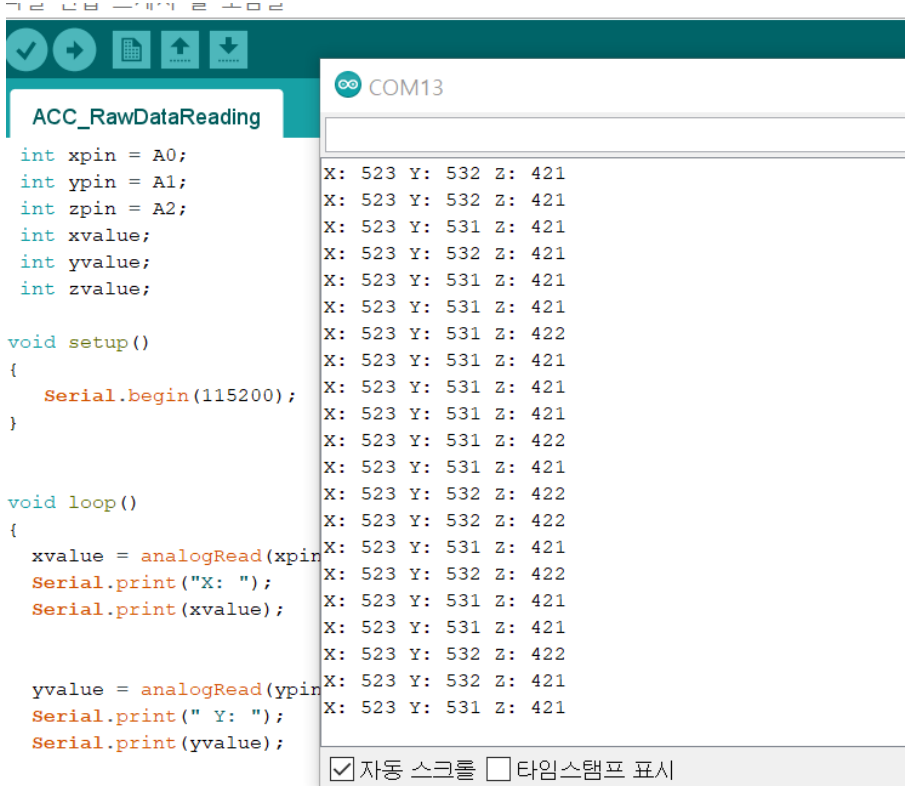
```
// Convert raw values to 'milli-Gs"
long xScaled = map(xRaw, RawMin, RawMax, -3000, 3000);
long yScaled = map(yRaw, RawMin, RawMax, -3000, 3000);
long zScaled = map(zRaw, RawMin, RawMax, -3000, 3000);
```

Finally, the sensor's output is scaled down to fractional Gs by dividing it by 1000 and displayed on the serial monitor.

```
// re-scale to fractional Gs
float xAccel = xScaled / 1000.0;
float yAccel = yScaled / 1000.0;
float zAccel = zScaled / 1000.0;

Serial.print("X, Y, Z  :: ");
Serial.print(xRaw);
Serial.print(", ");
Serial.print(yRaw);
Serial.print(", ");
Serial.print(zRaw);
Serial.print(" :: ");
Serial.print(xAccel,0);
Serial.print("G, ");
Serial.print(yAccel,0);
Serial.print("G, ");
Serial.print(zAccel,0);
Serial.println("G");
```

```
#define AccelPin     A0       // A0 is connected to X-axis of Accel
#define Samplesize   13       // filterSample number
int Array1 [Samplesize];        // array for holding raw sensor values for sensor

int rawData1, smoothData1;    // variables for sensor data

int toSend;

void setup(){
 Serial.begin(9600);
}

void loop()
{
 rawData1 = analogRead(AccelPin);                    // read X-axis of accelerometer
 smoothData1 = digitalSmooth(rawData1, Array1);

 toSend = map (smoothData1, 193, 280, 0, 255);          // the data from accelerometer mapped to form a byte
 Serial.write (toSend);
 delay(100);
}

 int digitalSmooth(int rawIn, int *sensSmoothArray){       // "int *sensSmoothArray" passes an array to the function - the asterisk indicates the array name is a pointer
 int j, k, temp, top, bottom;
 long total;
 static int i;
 static int sorted[Samplesize];
 boolean done;

 i = (i + 1) % Samplesize;              // increment counter and roll over if necc. - % (modulo operator) rolls over variable
 sensSmoothArray[i] = rawIn;          // input new data into the oldest slot


 for (j=0; j<Samplesize; j++){         // transfer data array into anther array for sorting and averaging
  sorted[j] = sensSmoothArray[j];
 }

 done = 0;                // flag to know when we're done sorting
 while(done != 1){      // simple swap sort, sorts numbers from lowest to highest
  done = 1;
  for (j = 0; j < (Samplesize - 1); j++){
   if (sorted[j] > sorted[j + 1]){     // numbers are out of order - swap
    temp = sorted[j + 1];
    sorted [j+1] =  sorted[j] ;
    sorted [j] = temp;
    done = 0;
   }
  }
 }

 bottom = max(((Samplesize * 15)  / 100), 1);
 top = min((((Samplesize * 85) / 100) + 1  ), (Samplesize - 1));  // the + 1 is to make up for asymmetry caused by integer rounding
 k = 0;
 total = 0;
 for (j = bottom; j< top; j++){
  total += sorted[j];       // total remaining indices
  k++;

 }

 return total / k;         // divide by number of samples
}
```

```
 int xpin = A0;
 int ypin = A1;
 int zpin = A2;
 int xvalue;
 int yvalue;
 int zvalue;

void setup()
{
  Serial.begin(115200);        // initialize the serial communications:
}

void loop()
{
 xvalue = analogRead(xpin);
//reads values from x-pin & measures acceleration in X direction
 Serial.print("X: ");
 Serial.print(xvalue);


 yvalue = analogRead(ypin);
 Serial.print(" Y: ");
 Serial.print(yvalue);

 zvalue = analogRead(zpin);
 Serial.print(" Z: ");
 Serial.print(zvalue);
 Serial.println();

 delay(100);
}
```



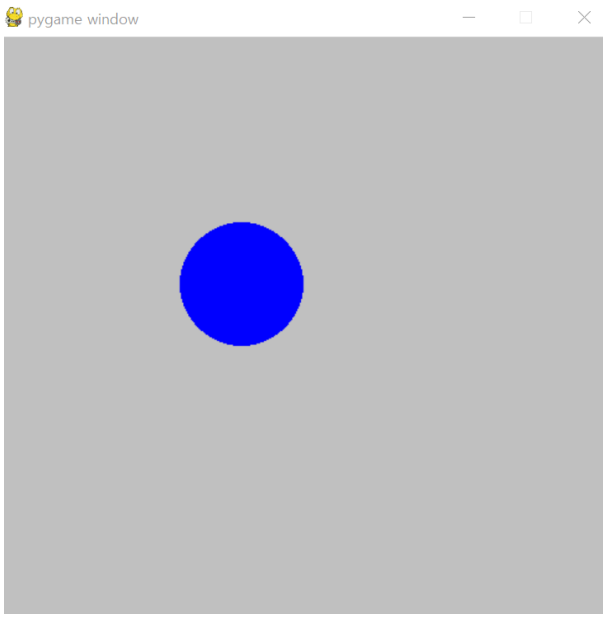
```
 int xpin = A0;
 int ypin = A1;
 int zpin = A2;
 int xvalue;
 int yvalue;
 int zvalue;

void setup()
{
  Serial.begin(115200);        // initialize the serial communications:
}


void loop()
{
  xvalue = analogRead(xpin);                    //reads values from x-pin & measures acceleration in X direction
  int x = map(xvalue, 400, 600, -10, 10);          //maps the extreme ends analog values from -100 to 100 for our understanding
//; you need to replace the 267 & 400 value with your values from calibration
  int xg = (float)x;                  //converts the mapped value into acceleration in terms of "g"
  //Serial.print("X: ");
  //Serial.print(xvalue);
  Serial.print(xg);                           //prints value of acceleration in X direction
  Serial.print(" ,");
//  Serial.print("g   ");                        //prints "g"

 // yvalue = analogRead(ypin);
 // int y = map(yvalue, 400, 400, -100, 100);
 // float yg = (float)y/(-10);
//  Serial.print(" Y: ");
//  Serial.print(yvalue);
  //Serial.print(" ,");
 // Serial.print("\t");
//  Serial.print(yg);
 // Serial.print("g   ");

  zvalue = analogRead(zpin);
  int z = map(zvalue,400, 600, -10, 10);
  int zg = (float)z;
 // Serial.print("\t");
  //Serial.print(" Z: ");
  //Serial.print(zvalue);
 // Serial.println();
  Serial.println(zg);
//  Serial.println("g   ");
  delay(100);
}
```

Name	Color	Code	RGB	HSL
white		#ffffff or #fff	rgb(255,255,255)	hsl(0,0%,100%)
silver		#c0c0c0	rgb(192,192,192)	hsl(0,0%,75%)
gray		#808080	rgb(128,128,128)	hsl(0,0%,50%)
black		#000000 or #000	rgb(0,0,0)	hsl(0,0%,0%)
maroon		#800000	rgb(128,0,0)	hsl(0,100%,25%)
red		#ff0000 or #f00	rgb(255,0,0)	hsl(0,100%,50%)
orange		#ffa500	rgb(255,165,0)	hsl(38.8,100%,50%)
yellow		#ffff00 or #ff0	rgb(255,255,0)	hsl(60,100%,50%)
olive		#808000	rgb(128,128,0)	hsl(60,100%,25%)
lime		#00ff00 or #0f0	rgb(0,255,0)	hsl(120,100%,50%)
green		#008000	rgb(0,128,0)	hsl(120,100%,25%)
aqua		#00ffff or #0ff	rgb(0,255,255)	hsl(180,100%,50%)
blue		#0000ff or #00f	rgb(0,0,255)	hsl(240,100%,50%)
navy		#000080	rgb(0,0,128)	hsl(240,100%,25%)
teal		#008080	rgb(0,128,128)	hsl(180,100%,25%)
fuchsia		#ff00ff or #f0f	rgb(255,0,255)	hsl(300,100%,50%)
purple		#800080	rgb(128,0,128)	hsl(300,100%,25%)

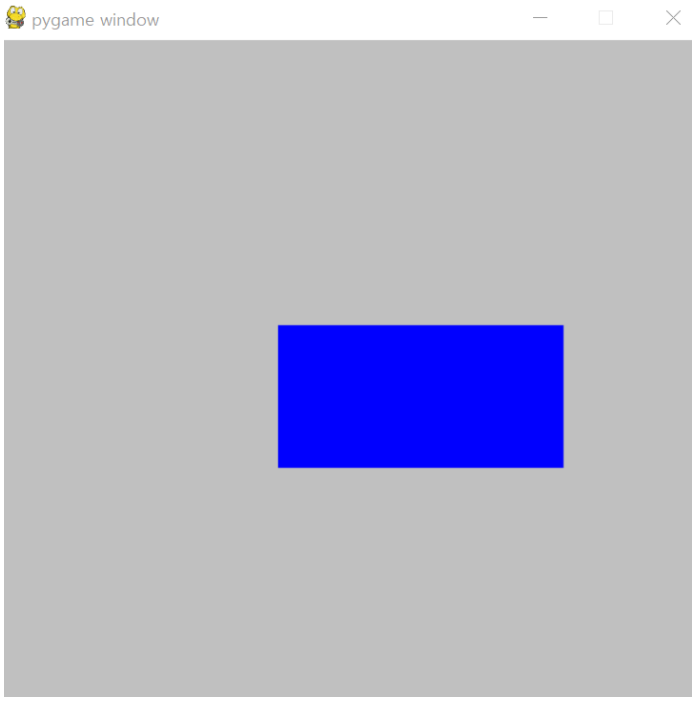
Python IDE Installation:

https://www.jetbrains.com/pycharm/

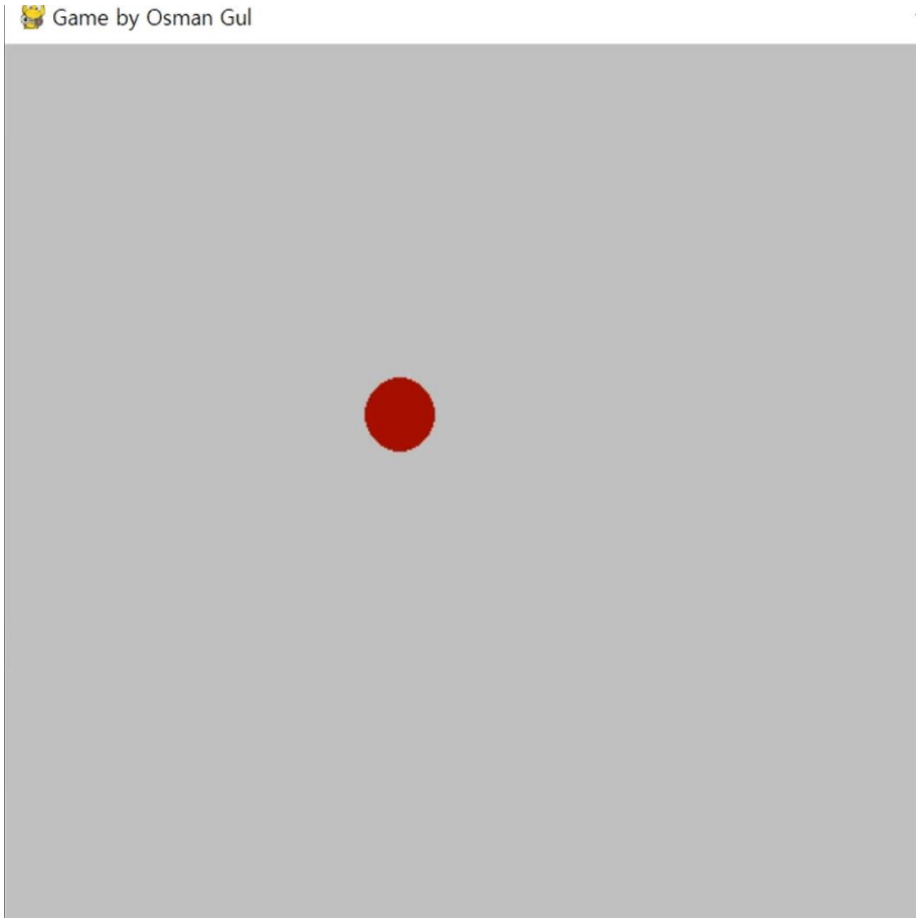
```
import pygame
import sys

pygame.init()

display = pygame.display.set_mode((500, 500))
display.fill((192, 192, 192))

pygame.draw.circle(display, (0, 0, 255), [200,200] , radius=50)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```



```
import pygame
import sys

pygame.init()

display = pygame.display.set_mode((500, 500))
display.fill((192, 192, 192))

pygame.draw.rect(display, (0, 0, 255), [200,200,200,100])
# location of 200,200 #width 200, height 100

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```



```
import serial  # import Serial Library
import numpy  # Import numpy
import pygame
import sys

arduinoData = serial.Serial('COM13', 115200,timeout = 1)  # Creating our serial object named arduinoData

SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480

white = (255, 255, 255)
black = (0, 0, 0)

pos_x = 200
pos_y = 200

color1 = (192, 192, 192)
color2 = (153, 0, 0)


pygame.init()
pygame.display.set_caption("Game by Osman Gul")

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
clock = pygame.time.Clock()

while True:  # While loop that loops forever
    while (arduinoData.inWaiting() == 0):  # Wait here until there is data
        pass  # do nothing
    arduinoString = arduinoData.readline()  # read the line of text from the serial port
    arduinoString = arduinoString.decode('UTF-8').strip()
    dataArray = arduinoString.split(',')  # Split it into an array called dataArray
    print(dataArray)
    x = int(dataArray[0].strip())
    z = int(dataArray[1].strip())

    clock.tick()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()


    if z>0:
        pos_x += 5

    if z < 0:
        pos_x -= 5

    #if x > 0:
    #    pos_y += 5

    #if x < 0:
    #    pos_y -= 5

    screen.fill(color1)
    pygame.draw.circle(screen, color2, (pos_x, pos_y), 20)
    pygame.display.update()
```



```
import serial  # import Serial Library
import numpy  # Import numpy
import pygame
import sys

arduinoData = serial.Serial('COM13', 115200)  # Creating our serial object named arduinoData

SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480

white = (255, 255, 255)
black = (0, 0, 0)

pos_x = 200
pos_y = 200

color1 = (192, 192, 192)
color2 = (153, 0, 0)


pygame.init()
pygame.display.set_caption("Game by Osman Gul")

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
clock = pygame.time.Clock()

while True:  # While loop that loops forever
    while (arduinoData.inWaiting() == 0):  # Wait here until there is data
        pass  # do nothing
    arduinoString = arduinoData.readline()  # read the line of text from the serial port
    arduinoString = arduinoString.decode('UTF-8').strip()
    dataArray = arduinoString.split(',')  # Split it into an array called dataArray
    print(dataArray)
    x = int(dataArray[0].strip())
    z = int(dataArray[1].strip())

    clock.tick()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()


    if z>0:
        pos_x += 5

    if z < 0:
        pos_x -= 5

    if x > 0:
        pos_y += 5

    if x < 0:
        pos_y -= 5

    screen.fill(color1)
    pygame.draw.circle(screen, color2, (pos_x, pos_y), 20)
    pygame.display.update()
```