Introduction to Robotics and Intelligent Systems Lab

Lab 3: 3D Transforms using Accelerometter

Authors of the report: Franc Papa, NurmukhammadAbdurasulov

Course instructors

Dr. Fangning Hu, Francesco Maurelli

Lab conducted by : Franc Papa, Nurmukhammad Abdurasulov

Date of execution: 17.03.2021

Jacobs University Bremen, March 17, 2021.

Introduction

Theoretical portion:

The theoretical part of this lab mostly consists of understanding and modelling the behaviour of linear circuit elements such as resistors, inductors, and capacitors.

Arduino Uno:

The arduino uno is a microcontroller that is used for most of the practical tasks of this experiment. The Arduino comes equipped with numerous ports and sources, namely digital and analog ports that can be connected to other circuits, as well as a few sources of voltage.

Processing

Processing is an "extension" of the java language used for graphical design and visual design. Processing can also be connected to an arduino via the arduino API (establishing a port). Processing can be used to draw shapes, or model data retrieved from arduino to display it.

Accelerometer

An accelerometer is a device used to measure g-force acceleration. The Adafruit MMA8451 Accelerometer Breakout used in this experiment also shows what way it is facing.

Resistor:

A resistor is a circuit element that resists the flow of an electric current. The behaviour of a resistor and its voltage-current relationship is best described by Ohm’s Law, which states that:
$V = IR.$
Reading: Voltage is equal to current times resistance, where voltage is in volts, and currents are in ampères.

Capacitor:

The capacitor is another circuit element that is used in this lab, a capacitor is a device used to store electric charge, the voltage-charge relationship of a capacitor is given by:
$Q = CV.$
Where Q = charge, in coulombs, C = capacitance, in Farads, and V = voltage
Derivating both sides of the equation with respect to time gives:

$i = C\frac{dV}{dt}$. One of the important properties of the capacitor for understanding this lab is that capacitors resist sudden changes in voltage.

Inductor

In many ways, the inductor behaves as the "dual" of a capacitor. A capacitor is a device that stores electric energy in the form of a magnetic field. The Voltage-Current relation of an inductor is given by the equation:

$V = L\frac{di}{dt}$, where L is the inductance, measured in henrys.

Laplace transform

The Laplace transform is a very useful tool used to solve ordinary differential equations. By definition, the Laplace Transform is given as:

$\mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}u(t)dt$. Where *u(t)* is known as the step function, a function that is 1

for t≥0 and 0 for t < 0. The result of the transformation is a function that is not dependent on time, but on a complex variable *s*. The result can then be transformed back to the time domain mostly by using tables with common Laplace Transforms. The usefulness of the Laplace transform comes in the form of being able to generalize ohm's law for capacitors and inductors by using impedance, defined as:

$z = \frac{V}{R}$,

For a resistor. this is the Ohm

For a capacitor: $z = \frac{1}{sC}$

For an inductor $z = sL$

This allows for the modelling of capacitors and inductors as resistors in the laplace domain to obtain a result that can then be transformed back into the time domain.

Transfer Function

A transfer function is the ratio of an output function over an input function (assuming zero initial conditions) . There are many kinds of transfer functions that can be defined, but they mostly take the form:

$H(s) = \frac{Y(s)}{X(s)}$, where *Y* = output of an element voltage or current, and *X* = an input of a current of voltage source.

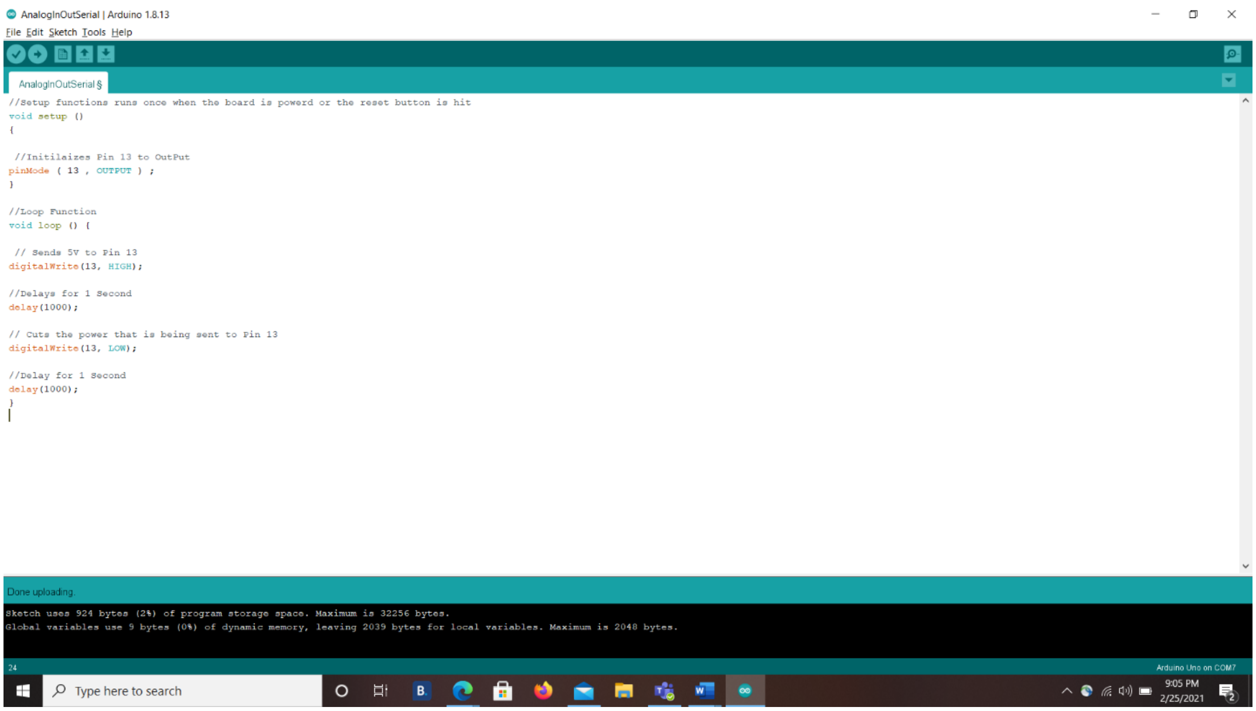
Simulink

Simulink is a programming environment developed by mathworks that is based on MATLAB for analyzing physical and electrical systems. In this lab, it is used to model electrical outputs of RC and RL circuits.

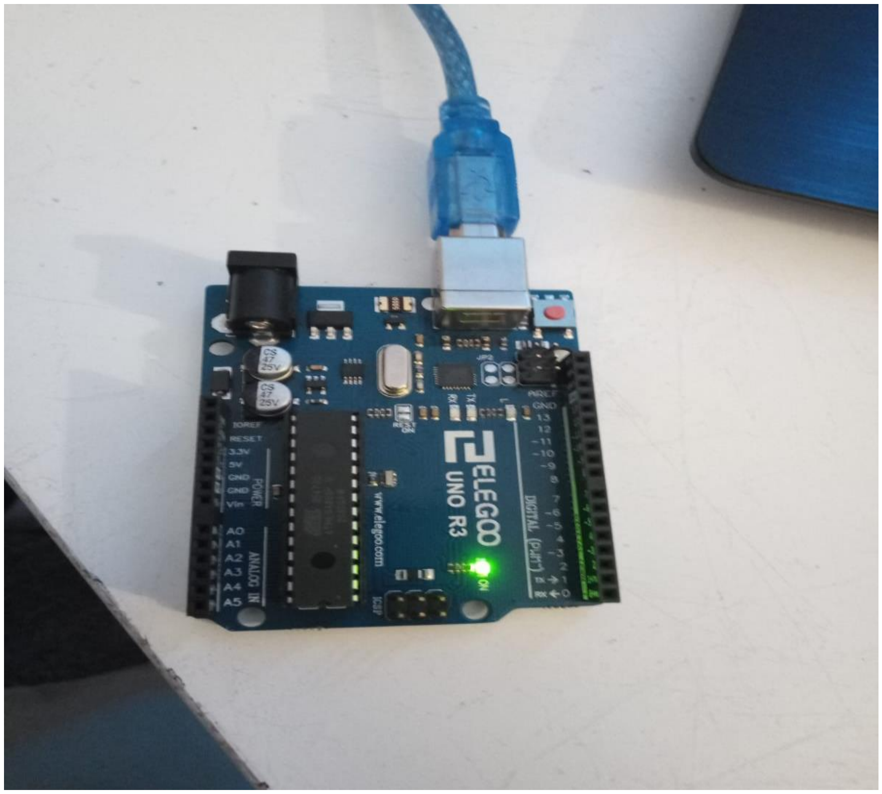
Lab Session 1

Task 1.1 First Program in Arduino

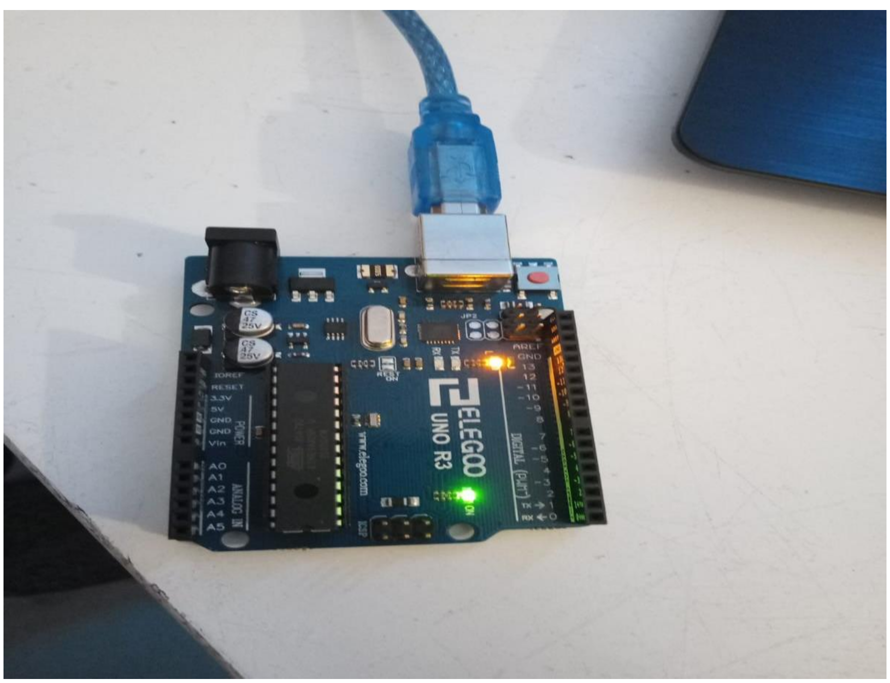
We were asked to be familiar with the arduino IDE and upload our first program on it.The code was as follows:



We uploaded our code in Arduino by using USB cable, before which 2we configured Serial Port. as in Figure 1(On/High State) and Figure 2(OFF/Low State).

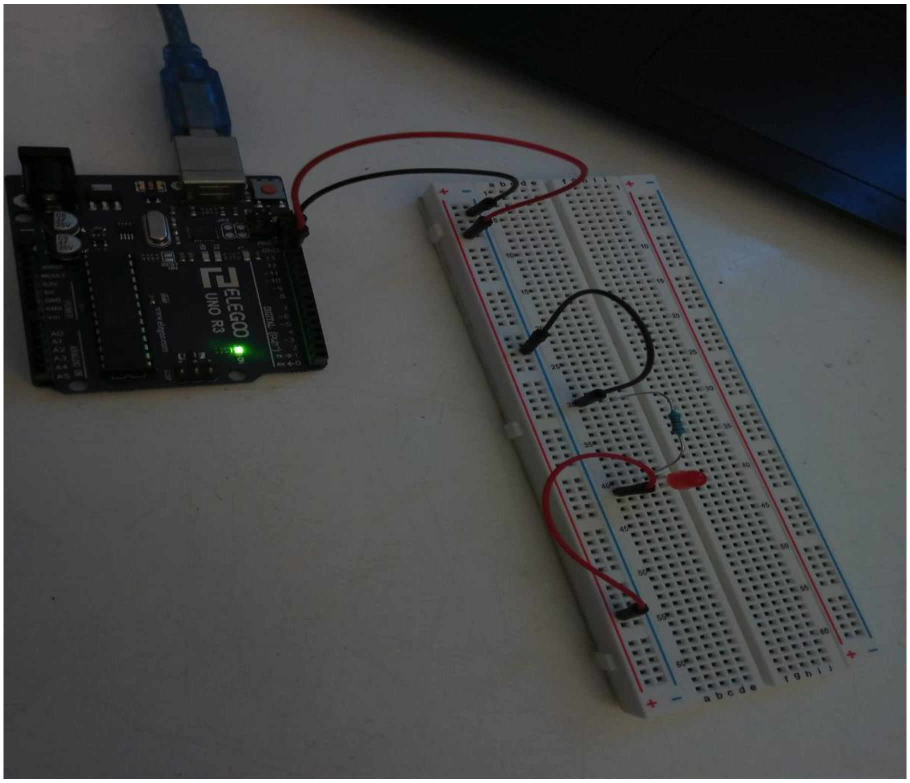


Picture Captured When LED Blinked ’OFF’ on Arduino board

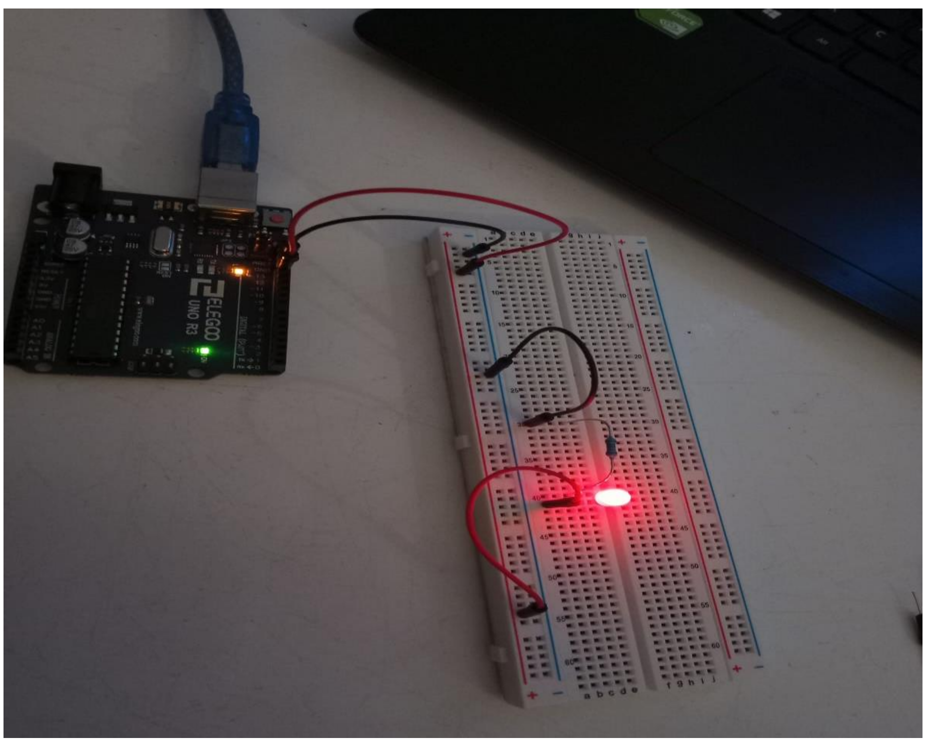


Picture Captured When LED Blinked ’ON’ on Arduino board

Task 1.2 Experiment as on 1.1 but by using external LED on breadboard

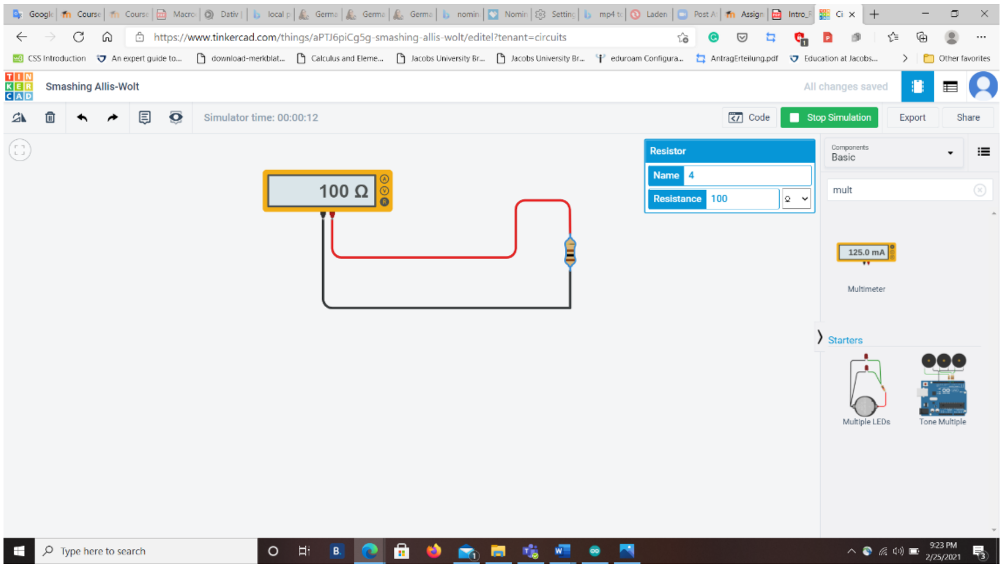


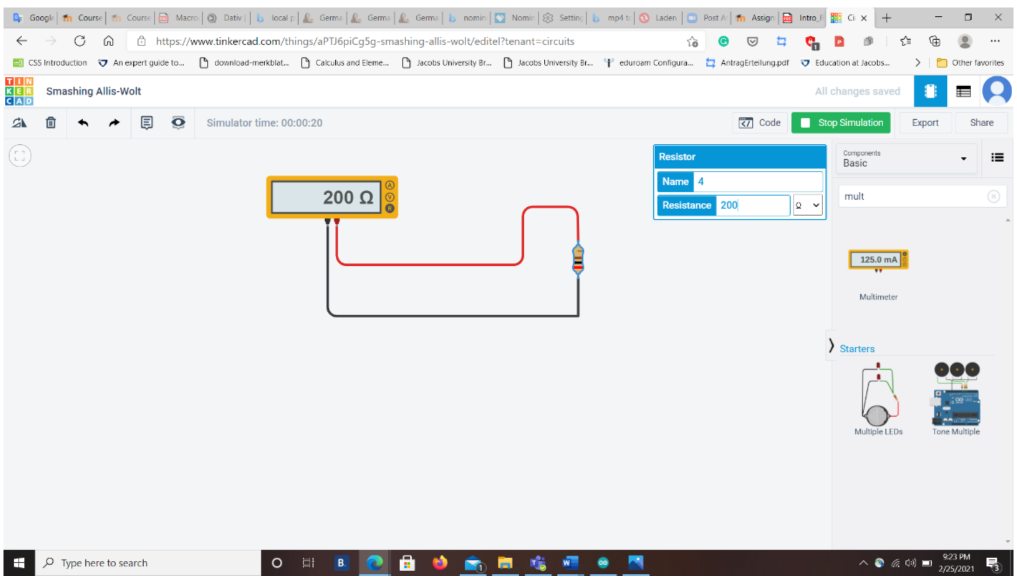
Picture Captured When LED Blinked ’OFF’ on Arduino Board

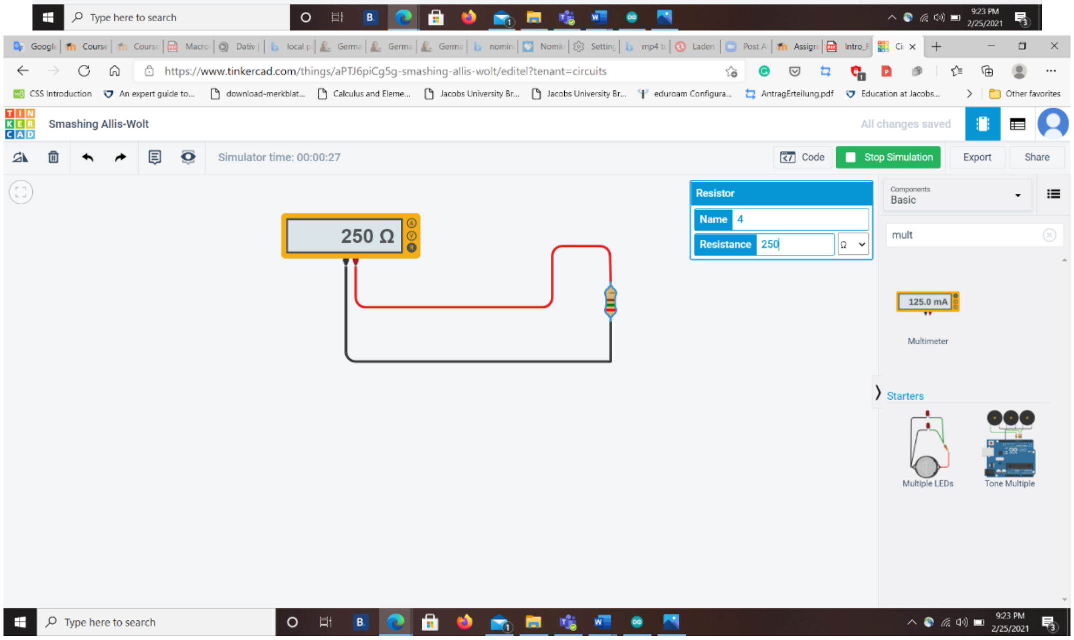


Picture Captured When LED Blinked ’ON’ on Arduino Board

Task 1.3 Measuring the resistance of resistors and recording them

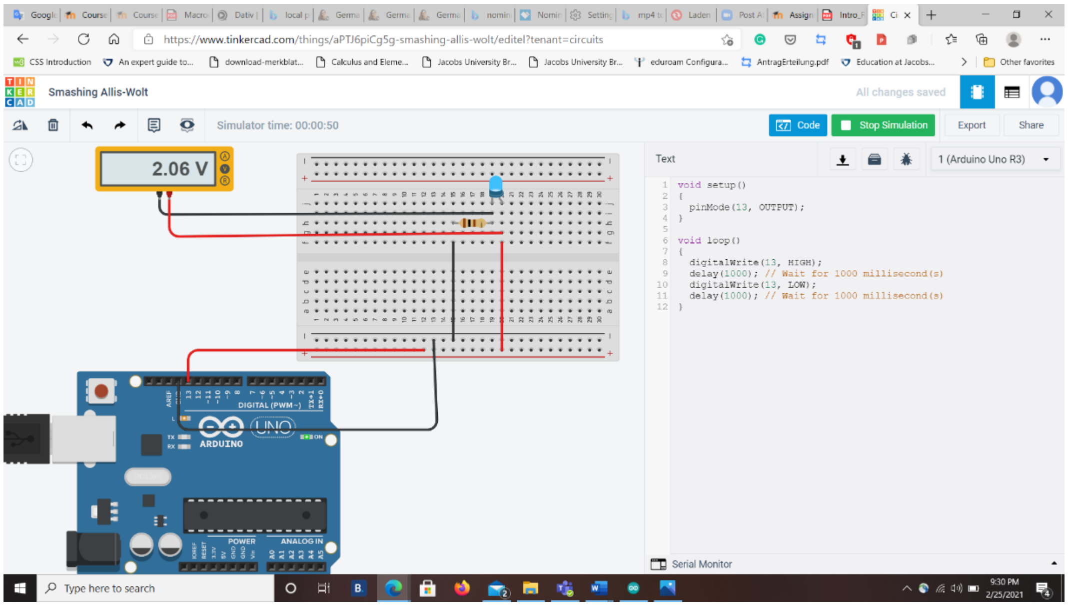






Measuring Resistance of Various Resistor using Multimeter

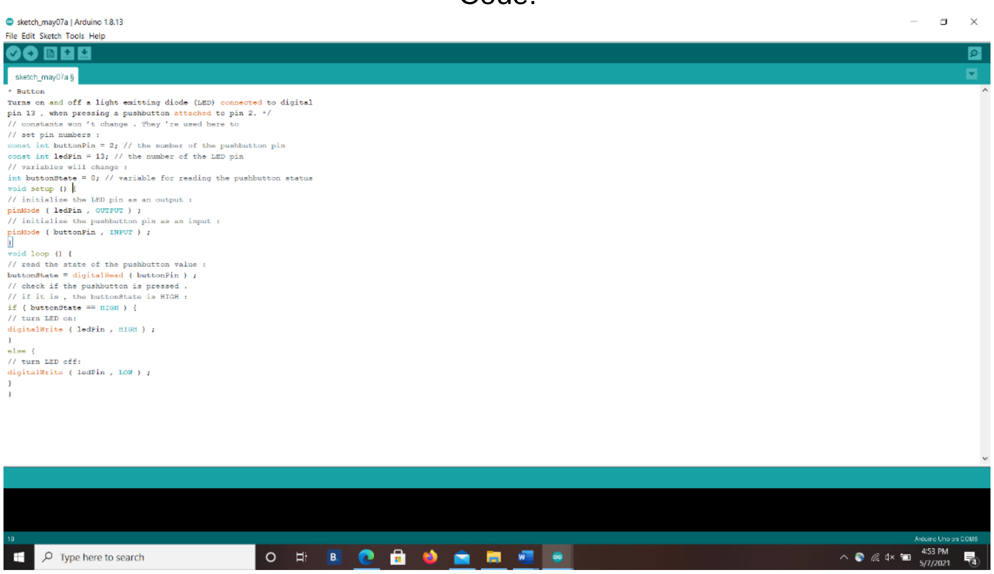
Task 1.4 Measuring the Voltage of the LED

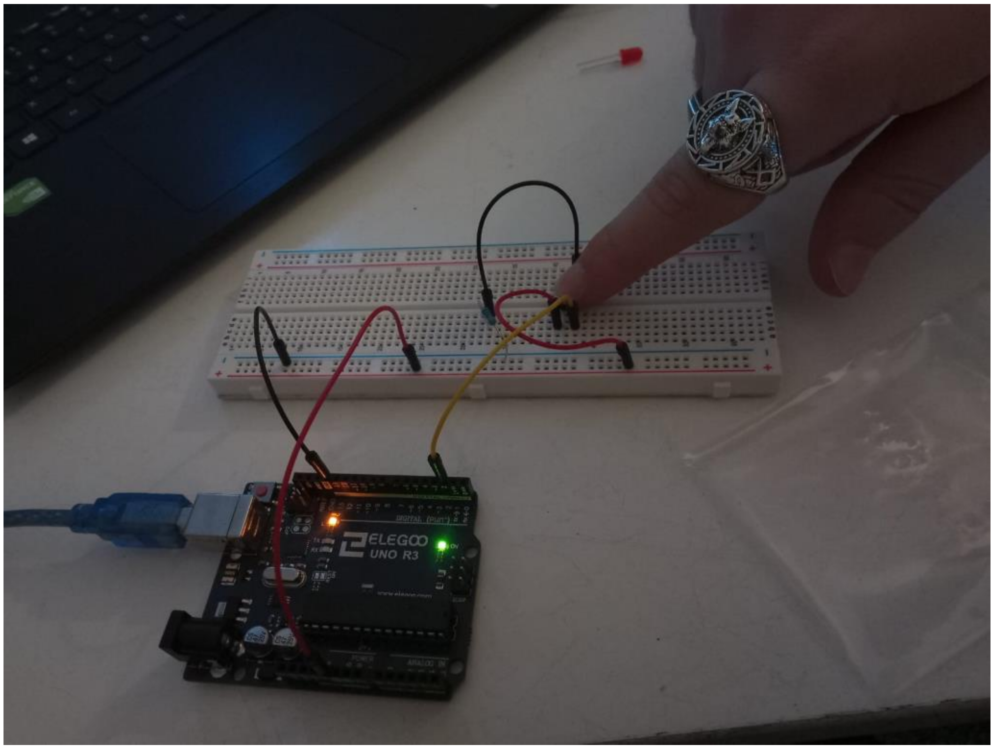


Measurement of Voltage of LED on circuit

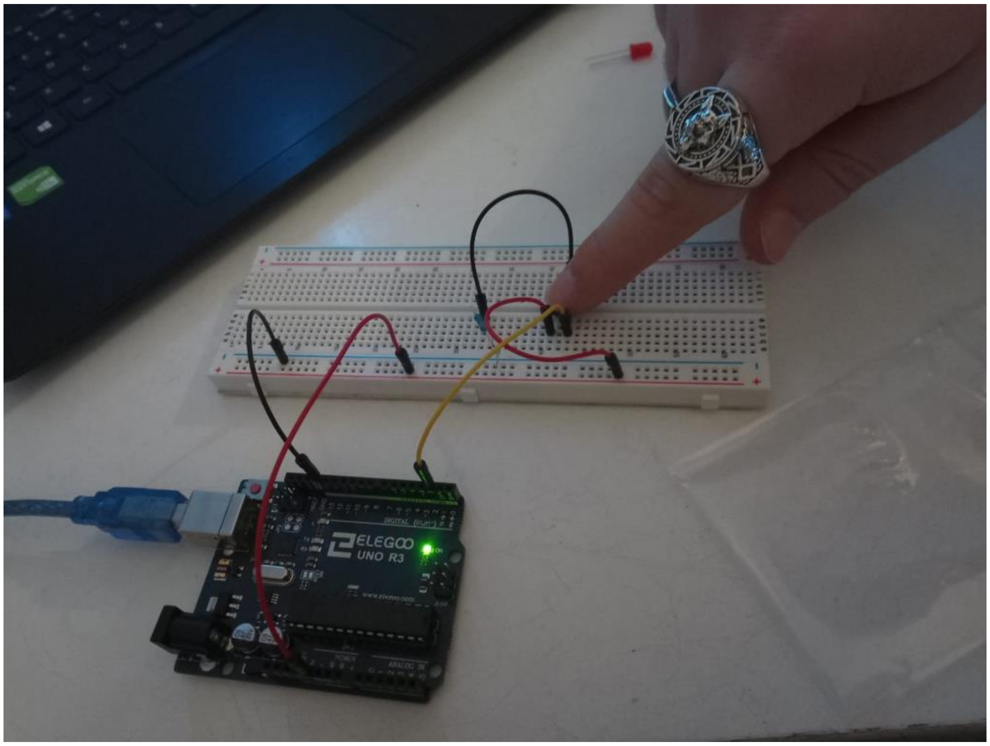
Task 1.6 Keeping the Button on the circuit so it only glows when pressed

Code:



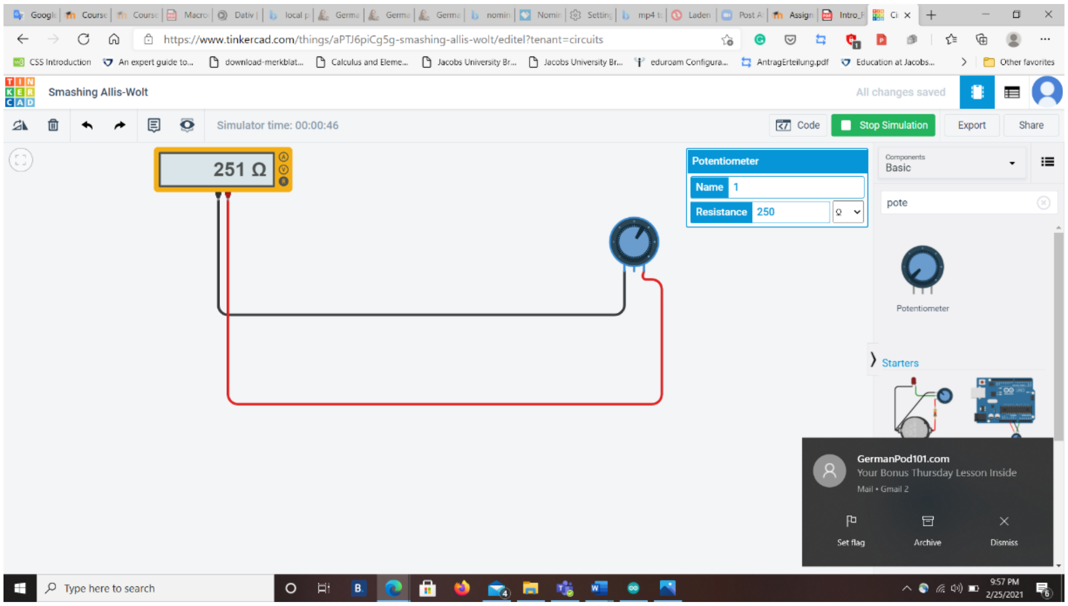


Bulb Glowing while Button is Pressed



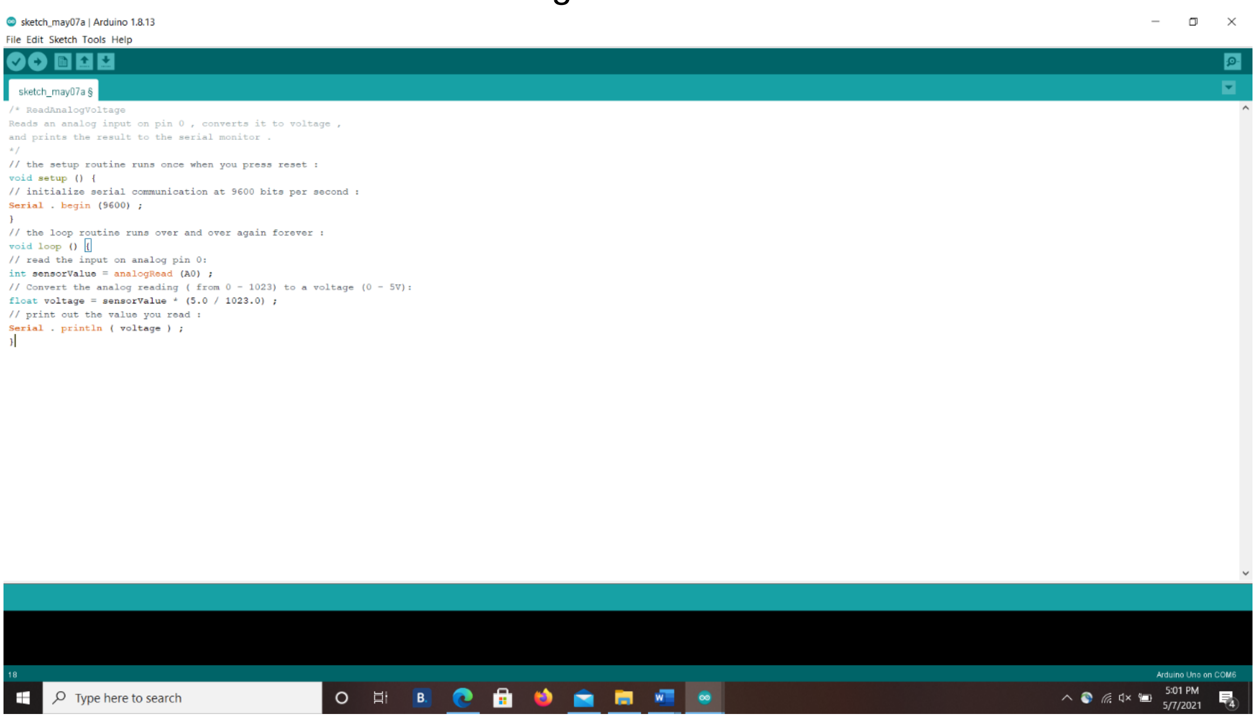
Bulb didn't glow while Button is not Pressed

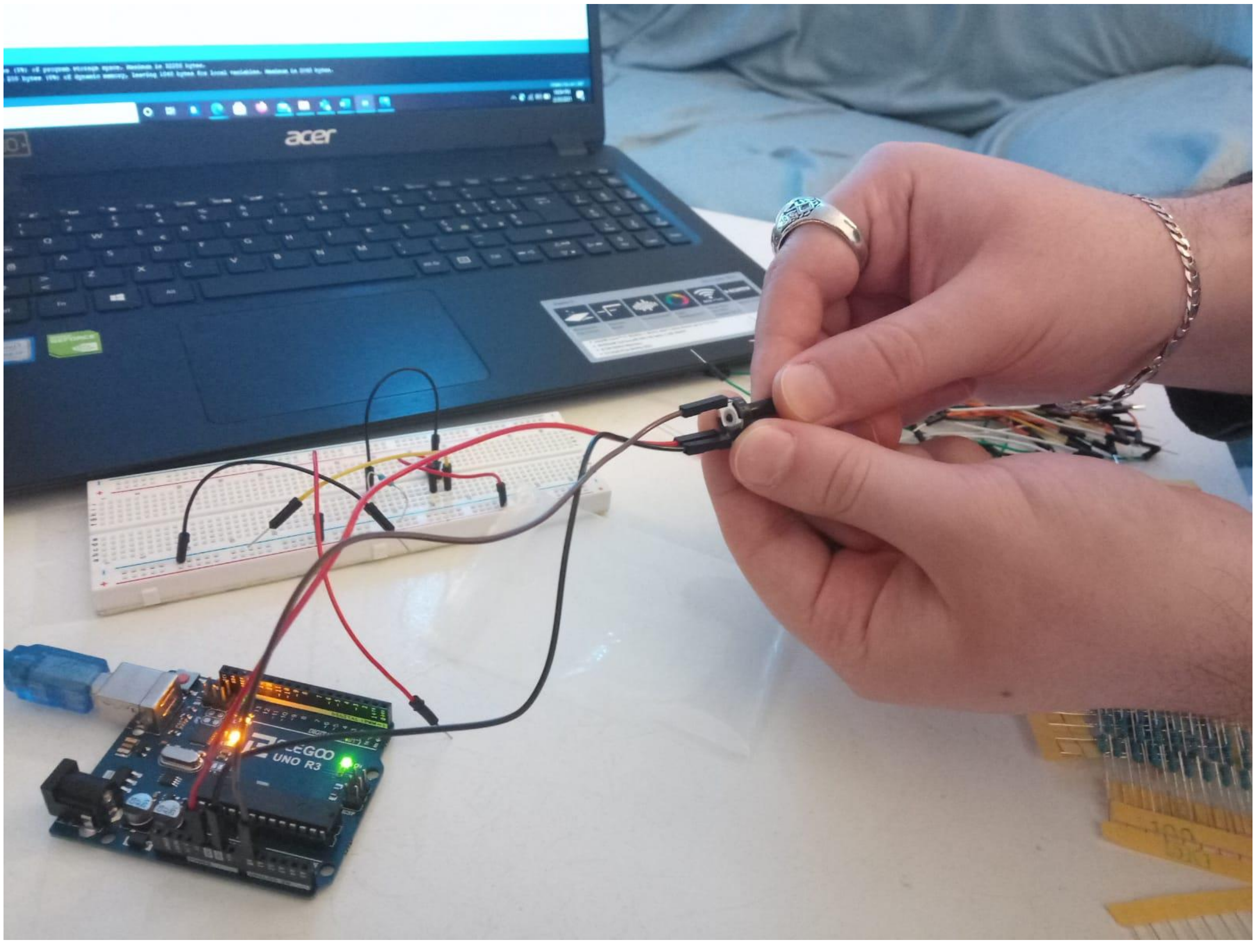
Task 1.7 Measuring the Total resistance across the outer pins of potentiometer



Measuring Resistance of Potentiometer

Task 1.8 Changing the amount of resistance using potentiometer and observing in serial monitor

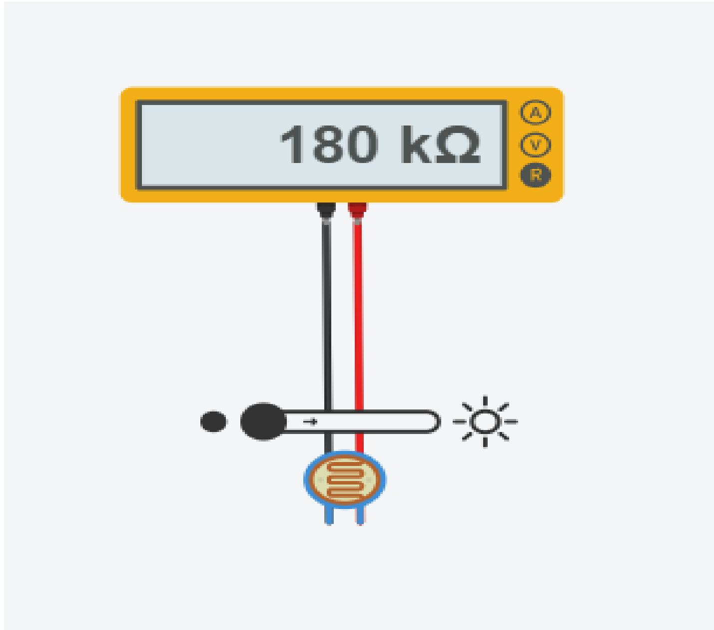




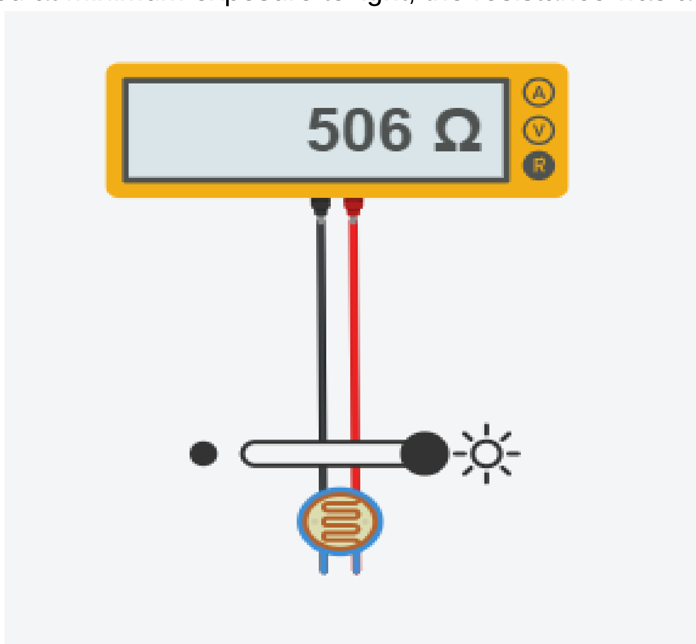
Lab Session 2

Task 2.1 Measuring Resistance Range Manually

We were asked to measure the range of resistance values exhibited by the LDR provided under various light conditions.



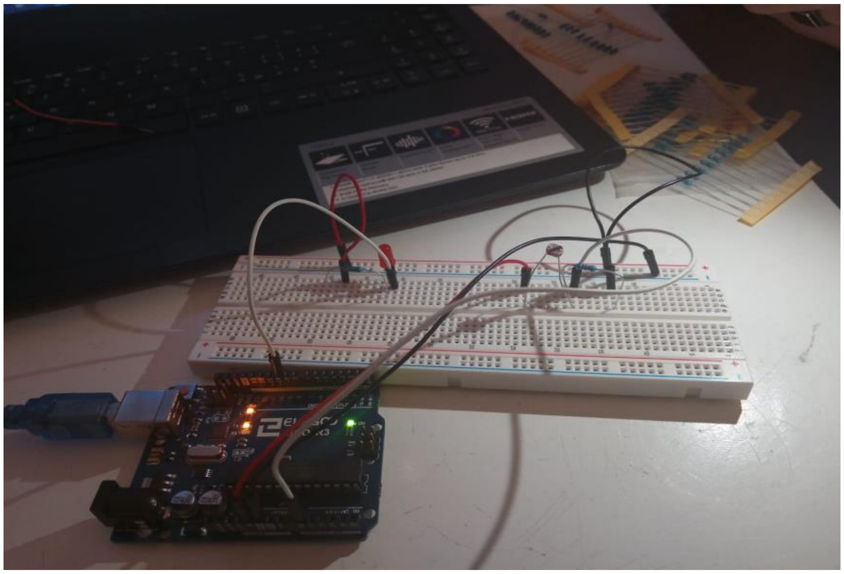
Picture captured at minimum exposure to light, the resistance was at 180 kilo ohms.



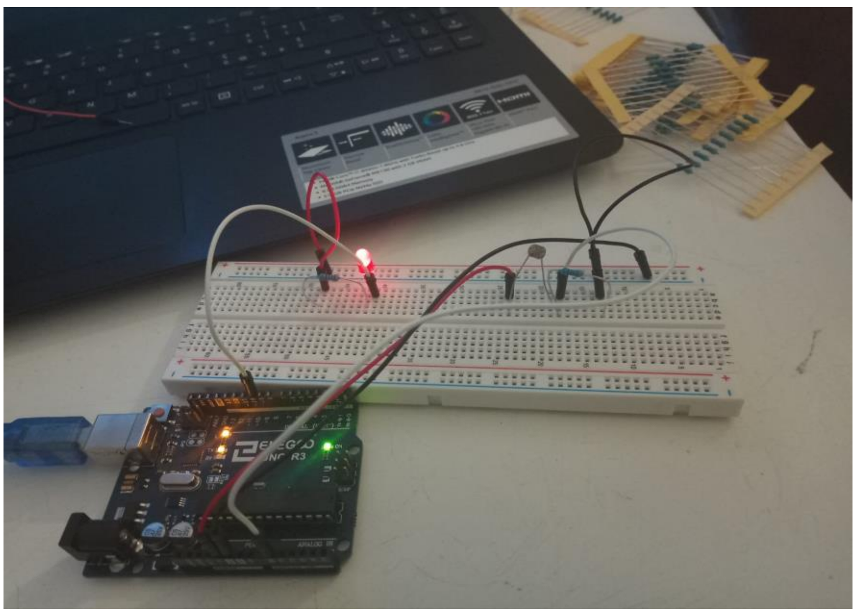
Picture captured at maximum exposure to light, the resistance was at 506 ohms.

Task 2.2 LDR in a Voltage-Divider

In task 2.2 we set up the circuit as shown in below pictures so as to ensure the LED turns on as soon as the LDR detects that it has become dark.

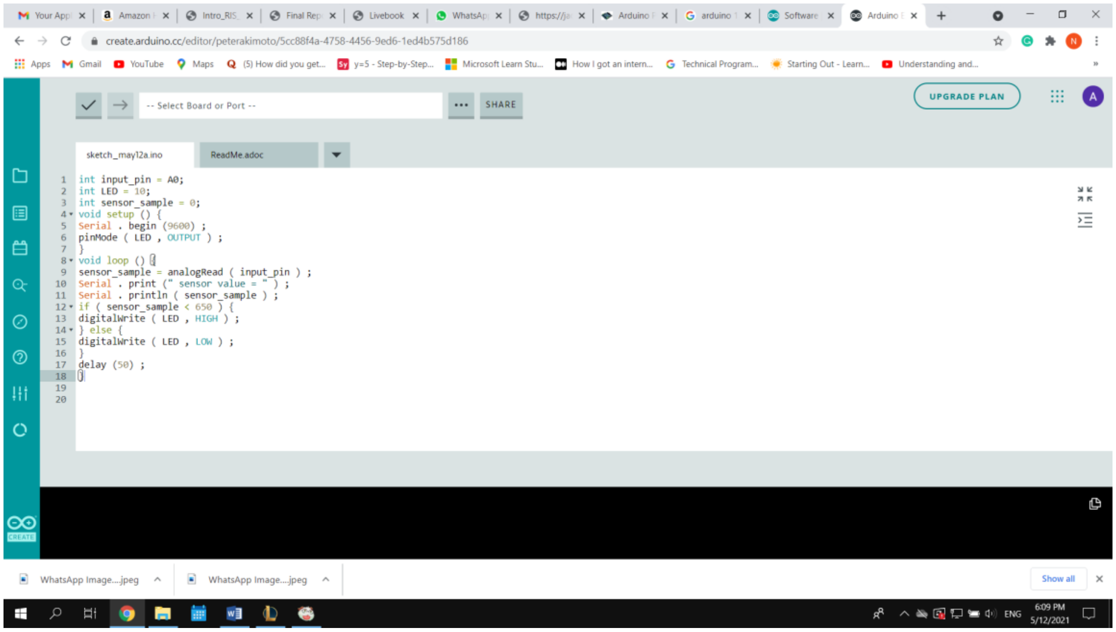


Picture captured when sensor value is greater than 650 hence LED is off.



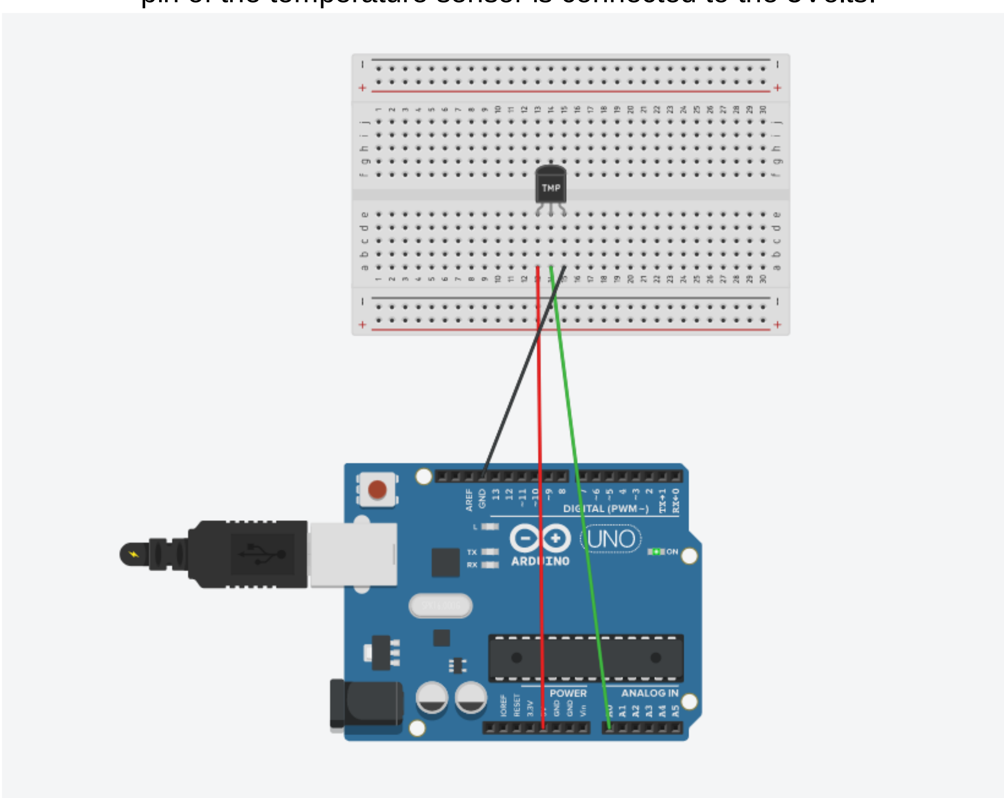
Picture captured when sensor value is less than 650 hence LED is on.

The code below was used to turn on the LED as soon as the LDR detects that it has become dark:



Task 2.3 Measuring the average-temperature and its mean

The middle pin is connected to A0 AND the other pin with a blue wire is connected to the ground. The middle pin is meant to read temperature and return the value after conversion to voltage. The pin of the temperature sensor is connected to the 5Volts.



Code used for measure temperature:

```
int TMP36 = A0; // The middle lead is connected to analog -in 0.
int temperature = 0;
int wait_ms = 20; // wait time betweem measurements in millisec .
# define NR_SAMPLES 10
int samples [ NR_SAMPLES ]; // array of samples
void setup (){
Serial . begin (9600) ;
}
void loop (){
float sum= 0.0;
for (int i=0; i< NR_SAMPLES ; ++i){
// map values from range [0, 410] to [-50, 150]
samples [i]= map ( analogRead ( TMP36 ), 0, 410 , -50, 150) ;
sum += samples [i];
delay ( wait_ms );
}
float mean = sum/ NR_SAMPLES ;
float sum_square_deviation = 0.0;
for (int i=0; i< NR_SAMPLES ; ++i){
sum_square_deviation += ( samples [i] - mean )*( samples [i] - mean );
}
float standard_deviation = sqrt ( sum_square_deviation / NR_SAMPLES );
Serial . print (" mean : ");
Serial . print (mean , 3);
Serial . print (" C, \t std: ");
Serial . println ( standard_deviation );
}
```

COM5		COM5	
mean : 23.800 C,	std: 0.60	mean : 30.000 C,	std: 0.63
mean : 23.700 C,	std: 0.64	mean : 29.700 C,	std: 0.78
mean : 23.200 C,	std: 0.75	mean : 29.700 C,	std: 0.46
mean : 23.700 C,	std: 0.90	mean : 30.000 C,	std: 0.45
mean : 23.800 C,	std: 0.87	mean : 30.000 C,	std: 0.45
mean : 23.400 C,	std: 0.66	mean : 29.900 C,	std: 0.54
mean : 23.700 C,	std: 0.78	mean : 29.800 C,	std: 0.40
mean : 24.000 C,	std: 0.63	mean : 30.000 C,	std: 0.45
mean : 23.700 C,	std: 0.64	mean : 30.000 C,	std: 0.63
mean : 23.800 C,	std: 0.60	mean : 29.800 C,	std: 0.87
mean : 23.700 C,	std: 0.64	mean : 29.900 C,	std: 0.54
mean : 23.900 C,	std: 0.54	mean : 30.000 C,	std: 0.77
mean : 23.500 C,	std: 0.67	mean : 29.900 C,	std: 0.30
mean : 24.000 C,	std: 0.45	mean : 29.800 C,	std: 0.40
mean : 23.700 C,	std: 0.64	mean : 29.900 C,	std: 0.54
mean : 23.400 C,	std: 0.49	mean : 29.800 C,	std: 0.87

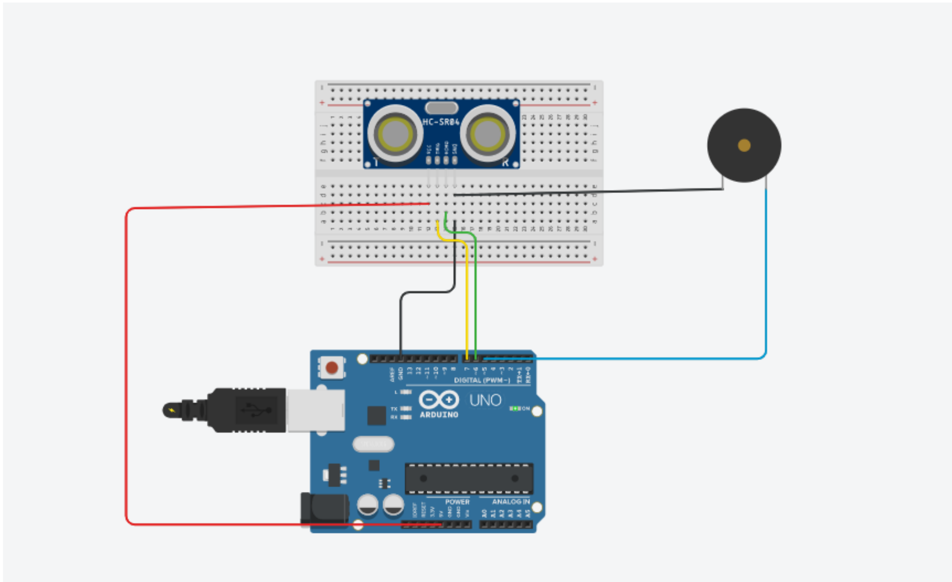
:

Picture captured before touching the sensor and after

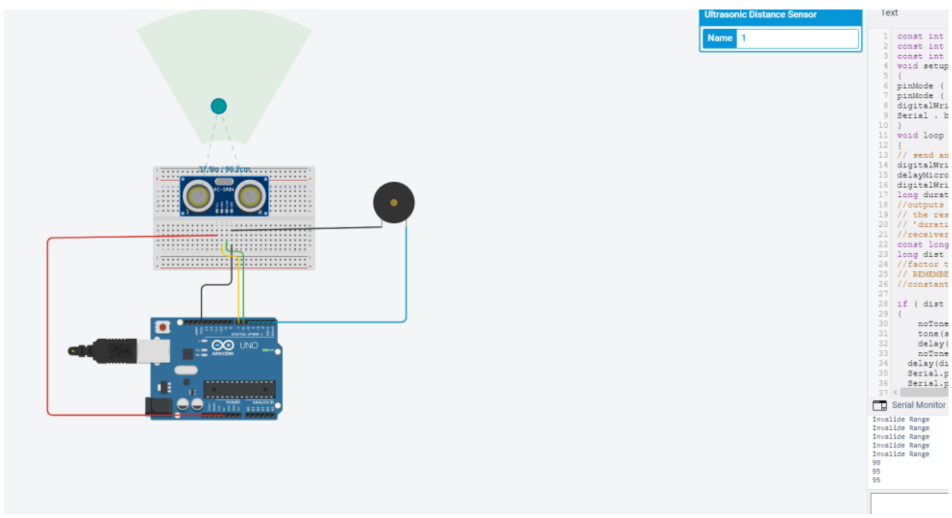
Task 2.4

The reason we use the value 410 in the call to the map function is because when it comes to the temperature sensor the middle pin gives out a voltage between 0-2 Volts. A value of 0 corresponds to the temperature of -50 degrees Celsius, and the value of 2 volts corresponds to a temperature of 150 degrees Celsius which is equivalent to 410

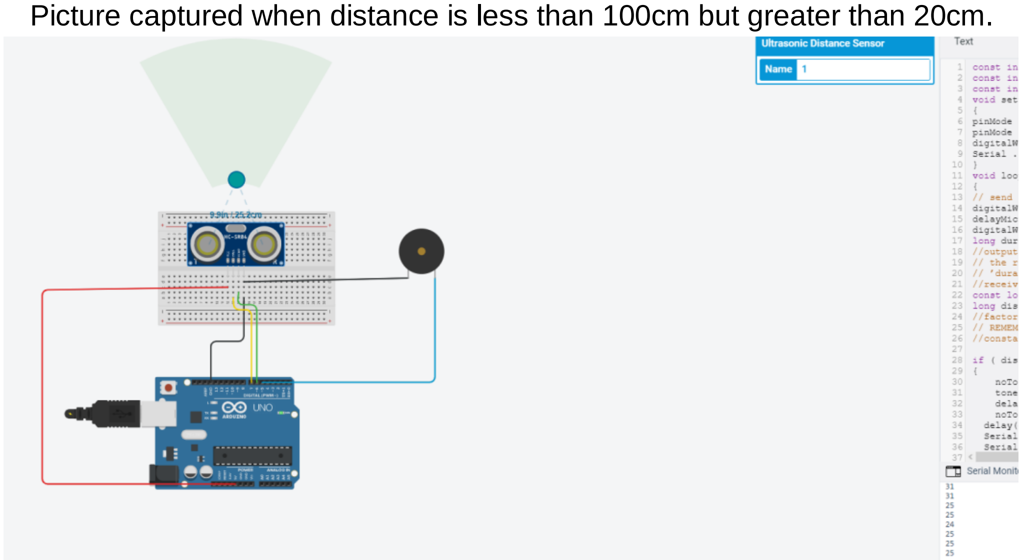
Task 2.5 Creating a car parking sensor using the Ultrasonic distance sensor



Picture of the circuit used for car parking sensor



Picture captured when distance is less than 100cm but greater than 20cm.



Picture captured when the distance was less than 20cm

the distance was less than 20cm and the beep was continuous without any interruptions

Code used:

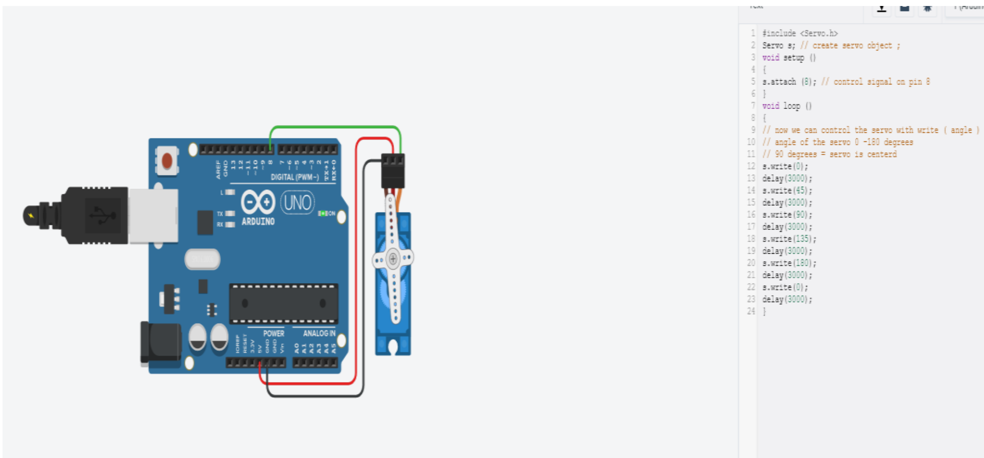
```
const int trig = 7;
const int echo = 6;
const int speaker=5;
void setup ()
{
pinMode ( trig , OUTPUT ) ;
pinMode ( echo , INPUT ) ;
digitalWrite ( trig , LOW) ;
Serial . begin (9600) ;
}
void loop ()
{
// send an impulse to trigger the sensor start the measurement
digitalWrite ( trig , HIGH ) ;
delayMicroseconds (15) ; // minimum impulse width required by HC -SR4 sensor
digitalWrite ( trig , LOW) ;
long duration = pulseIn ( echo , HIGH ) ; // this function waits until the sensor
//outputs the result
// the result is encoded as the pulse width in microseconds
// 'duration ' is the time it takes sound from the transmitter back to the
//receiver after it bounces off an obstacle
const long vsound = 340; // [m/s]
long dist = ( duration / 2L ) * vsound / 10000L ; // 10000 is just the scaling
//factor to get the result in [cm]
// REMEMBER : when doing operations with 'long ' variables , always put 'L' after
//constants otherwise you will have bugs !

if ( dist < 100L && dist > 20L )
{
    noTone(speaker);
    tone(speaker,1000);
    delay(dist*10);
    noTone(speaker);
  delay(dist*10);
  Serial.print(dist);
  Serial.print("\n");
}
else if(dist<20L)
{
  tone(speaker,1000);
  Serial.print(dist);
  Serial.print("\n");
}
else{
    Serial.print("Invalide Range");
    Serial.print("\n");
  }
```

Task 2.6

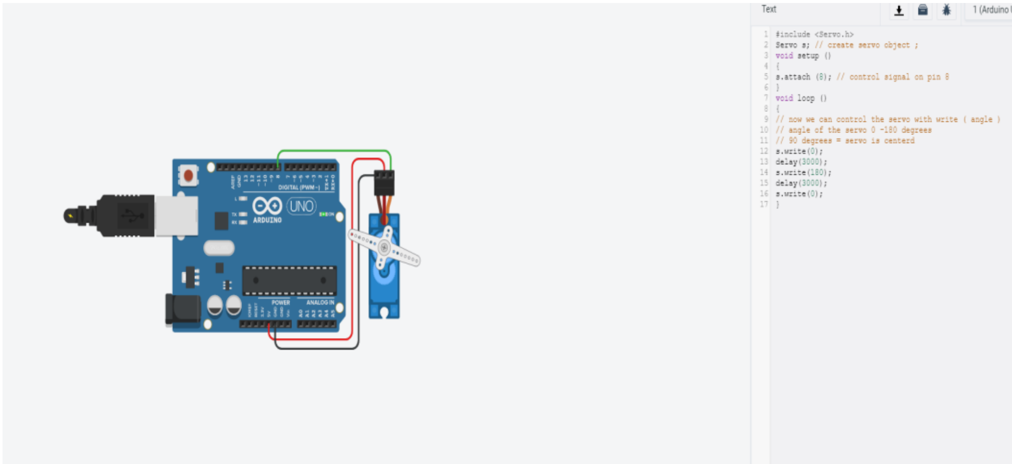
Pins 9 and 10 cannot be used on the Uno when using the servo library

Task 2.7 Changing the code to stop the servo at 0 degrees

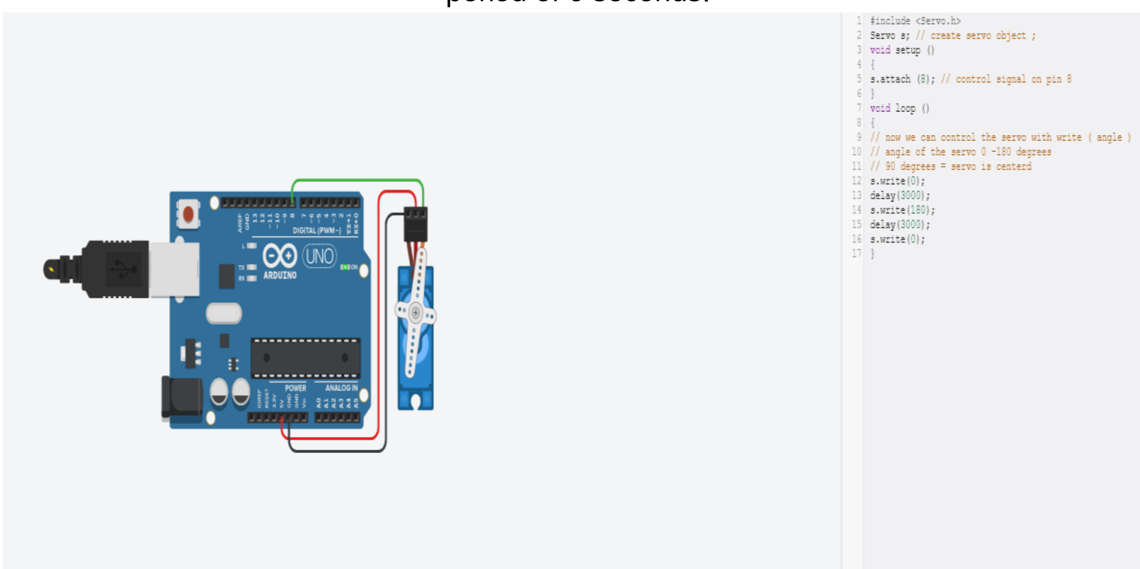


Picture captured when the servo starts at 0 degrees

Task 2.8 Making the servo transition smoothly from 0 to 180 and back in a continuous cycle



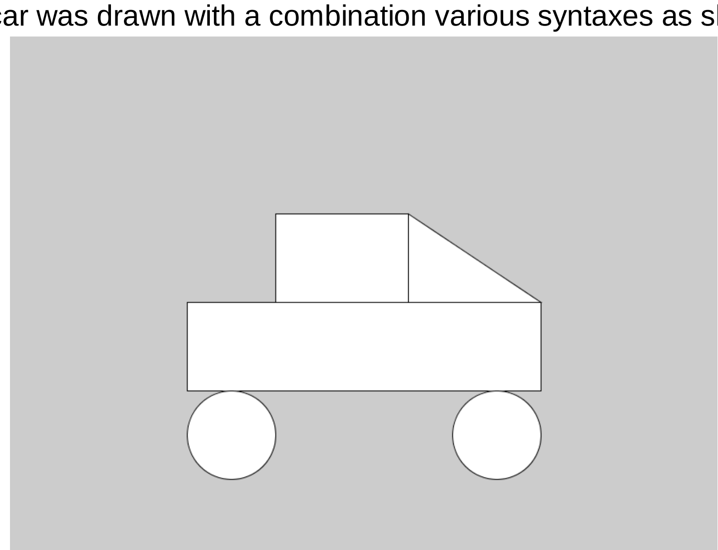
The code written ensures that the servo transition smoothly from 0 to 180 degrees and back in a period of 6 seconds.

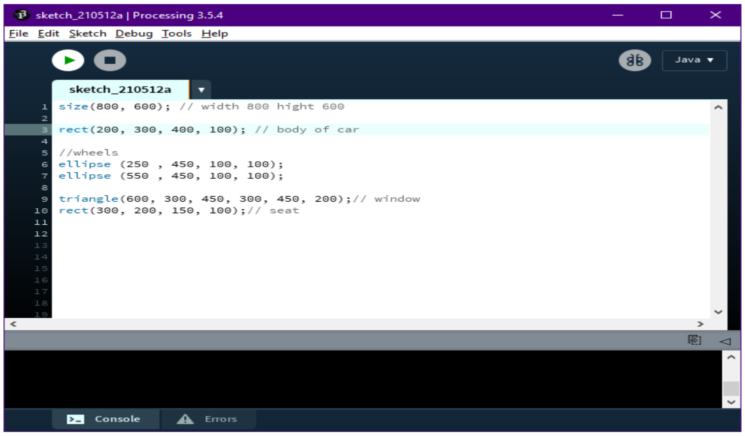


Picture captured when the servo moves from 180 to 0 degrees

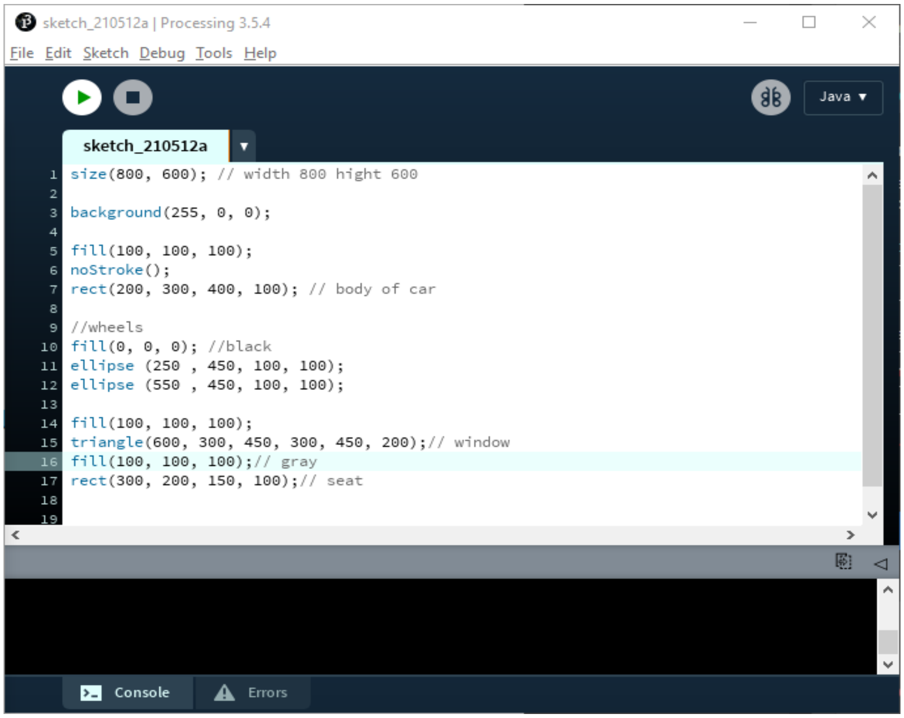
Task 2.9

2D car was drawn with a combination various syntaxes as shown




sketch_210512a | Processing 3.5.4
File Edit Sketch Debug Tools Help
Java
sketch_210512a
size(800, 600); // width 800 hight 600
rect(200, 300, 400, 100); // body of car
//wheels
ellipse (250 , 450, 100, 100);
ellipse (550 , 450, 100, 100);
triangle(600, 300, 450, 300, 450, 200);// window
rect(300, 200, 150, 100);// seat
Console
Errors

Task 2.10



Lab Session 3

Task 3.1

In Task 3.1 we are going to use the "Processing" Programming Language to perform 3D transformation using accelerometer. We use the P3D renderer for drawing in 3D.

Run the code listed in the lab manual Listing 3.1. The XYZ coordinate-system (CS) in Processing is left-handed CS. The origin is at the top-left, X-axis points to the right of the screen,Y-axis points to the bottom of the screen, and the Z-axis is coming out of the screen.

For the task 3.1 we should complete the following:

In the code, the variables 'rotx' and 'roty' are the rotation angles corresponding to the X-axis and Y-axis. Be familiar with the commands rotateX() and rotateY() and explain theusage of these two commands.

```
float rotx , roty ;
void setup () {
size (400 ,400 , P3D);
rotx = 0.0;
roty = 0.0;
}
void draw () {
background (255) ;
pushMatrix ();
translate ( width /2.0 , height /2.0 , -50);
rotateX ( rotx );
rotateY ( roty );
draw_axes ();
popMatrix ();
}
void draw_axes () {
fill (100 , 200 , 100 , 127) ;
stroke (0);
box (90 , 60, 30) ;
stroke (255 , 0, 0);
line (0, 0, 0, 100 , 0, 0); // Red X- Axis
stroke (0, 255 , 0);
line (0, 0, 0, 0, 100 , 0); // Green Y- Axis
stroke (0, 0, 255) ;
line (0, 0, 0, 0, 0, 100) ; // Blue Z- Axis
}
/** This is an event - handler function .
*/
void mouseDragged () {
float rate = 0.01;
rotx += ( pmouseY - mouseY ) * rate ;
roty += ( mouseX - pmouseX ) * rate ;
}
```

Figure1: Task 3.1 Code.

In the beginning the *background(255)* to set the color of the background, then the command *translate* is used to specify the amount to displace the object within the display window. In the*draw_axes()* function, we are drawing the *x (red), y(green) and z(blue)* axes going through thefaces of the cube. *mouseDragged*() function determined amount by which the object should move when we move the mouse by a certain distance. *draw ()* function rotates everything by respective amount using *rotateX()* and *rotateY()*. We ran the code in processing and the following results were obtained. We rotated the 3D object in different positions, each position isshown below.

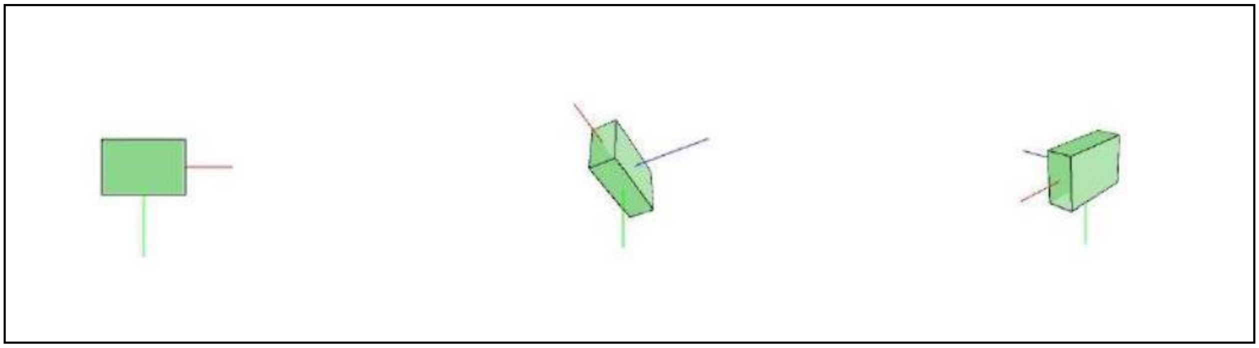
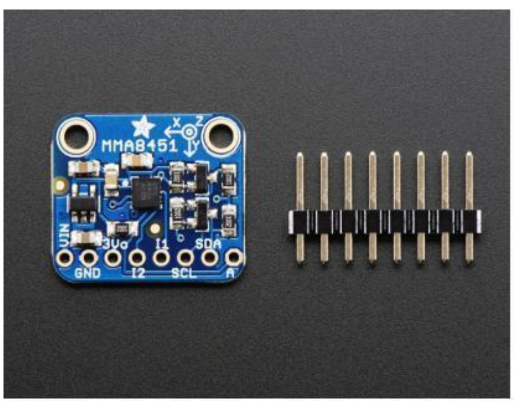


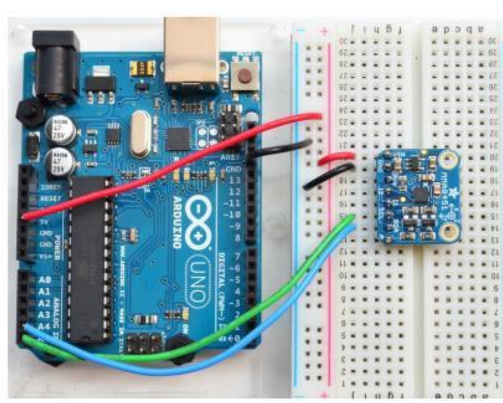
Figure2: Object visualization in Processing

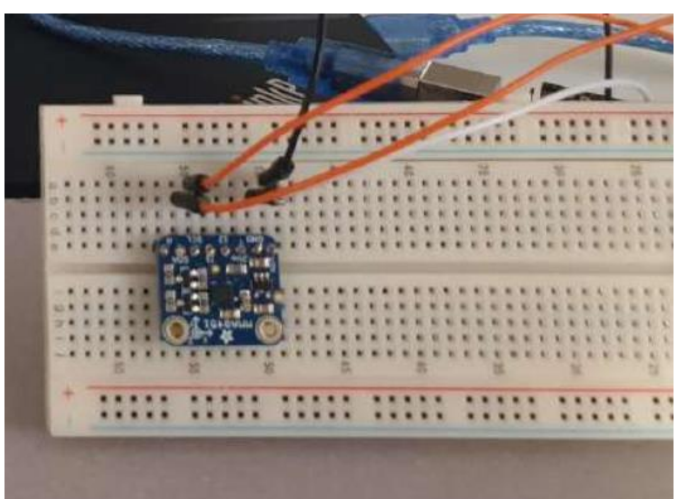
<u>*Task 3.2*</u>

In task 3.2 we are going to measure the acceleration using the accelerometer which is a devicethat measures the vibration, or acceleration of motion of a structure. The force caused by vibration or a change in motion (acceleration) causes the mass to "squeeze" the piezoelectric material which produces an electrical charge that is proportional to the force exerted upon it.

In our lab we use the accelerometer MMA8451. From Freescale with a 14-bit ADC resolution.We built the circuit as in the figure below.







We then download the file Adafruit MMA8451 Library-master.zip. In the Arduino IDE, went tothe menu item Sketch, Include Library, Add .ZIP Library, and in the file-dialog, and selected thezip file.

We downloaded the file Adafruit Sensor-master.zip to a folder of your choice. In the ArduinoIDE, went to the menu item Sketch, Include Library, Add .ZIP Library, and in the file-dialog, selected the zip file.

Then we run the code In the Arduino IDE, opened File, Examples, Adafruit MMA8451 Library, MMA8451demo, compiled it, and uploaded it to Arduino. Then we opened the serial monitor and the following output appeared.

```
X:      -50     Y:      396     Z:      4026
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
Portrait Up Front

X:      -40     Y:      404     Z:      4010
X:      -0.01   Y:      0.09    Z:      0.98    m/s^2
Portrait Up Front

X:      -44     Y:      388     Z:      3998
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
Portrait Up Front

X:      -40     Y:      412     Z:      4006
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
Portrait Up Front

X:      -36     Y:      404     Z:      3996
X:      -0.01   Y:      0.09    Z:      0.98    m/s^2
Portrait Up Front

X:      -36     Y:      380     Z:      3998
X:      -0.01   Y:      0.09    Z:      0.99    m/s^2
Portrait Up Front

X:      -44     Y:      398     Z:      4002
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
Portrait Up Front

X:      -40     Y:      400     Z:      4024
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
Portrait Up Front

X:      -46     Y:      408     Z:      3988
X:      -0.01   Y:      0.10    Z:      0.98    m/s^2
```

Figure3: Accelerometer output in the serial monitor measuring the G-force acceleration whichshows 0.98 m/s2 in the free fall state.

The unit m/s2 is not really the unit the accelerometer measures, it measures the force which applies in our sensor. It uses electromechanical sensor that is designed to measure either static ordynamic acceleration. Static acceleration is the constant force acting on a body, like gravity or friction. These forces are predictable and uniform to a large extend. For example, the acceleration due to gravity is constant at 9.8m/s, and the gravitation force is almost the same at every point on earth. The value shown in the serial monitor is off by a factor of 10, the real valueshould be 9.8m/s2 while the serial monitor shows 0.98m/s2.

Then we show the output of the accelerometer in different positions:

0: Portrait Up Front
1: Portrait Up Back
2: Portrait Down Front
3: Portrait Down Back
4: Landscape Right Front

5: Landscape Right Back
6: Landscape Left Front
7: Landscape Left Back

```
15:56:04.399 -> X:      -9.73   Y:      0.19    Z:      -0.68   m/s^2
15:56:04.433 -> Landscape Left Back
15:56:04.466 ->
15:56:04.908 -> X:      -4068   Y:      74      Z:      -294
15:56:04.942 -> X:      -9.74   Y:      0.16    Z:      -0.76   m/s^2
15:56:04.976 -> Landscape Left Back
15:56:04.976 ->
15:56:05.420 -> X:      -4086   Y:      6       Z:      -258
15:56:05.455 -> X:      -9.78   Y:      0.03    Z:      -0.57   m/s^2
15:56:05.489 -> Landscape Left Back
15:56:05.522 ->
15:56:05.932 -> X:      -4074   Y:      78      Z:      -302
15:56:05.965 -> X:      -9.76   Y:      0.17    Z:      -0.73   m/s^2
15:56:05.999 -> Landscape Left Back
15:56:06.033 ->
```

Figure4: LANDSCAPE LEFT BACK

```
15:54:35.301 -> X:      9.62    Y:      -0.49   Z:      -0.16   m/s^2
15:54:35.335 -> Landscape Right Back
15:54:35.369 ->
15:54:35.816 -> X:      4010    Y:      -134    Z:      102
15:54:35.816 -> X:      9.62    Y:      -0.34   Z:      0.23    m/s^2
15:54:35.849 -> Landscape Right Back
15:54:35.885 ->
15:54:36.328 -> X:      4032    Y:      6       Z:      186
15:54:36.328 -> X:      9.65    Y:      -0.02   Z:      0.40    m/s^2
15:54:36.397 -> Landscape Right Back
15:54:36.397 ->
15:54:36.843 -> X:      4034    Y:      -6      Z:      158
15:54:36.877 -> X:      9.65    Y:      -0.02   Z:      0.41    m/s^2
15:54:36.912 -> Landscape Right Back
15:54:36.912 ->
```

Figure5: LANDSCAPE RIGHT BACK

```
15:54:17.596 -> X:      9.58    Y:      0.68    Z:      0.60    m/s^2
15:54:17.631 -> Landscape Right Front
15:54:17.665 ->
15:54:18.075 -> X:      4030    Y:      244     Z:      180
15:54:18.108 -> X:      9.63    Y:      0.62    Z:      0.42    m/s^2
15:54:18.143 -> Landscape Right Front
15:54:18.177 ->
15:54:18.588 -> X:      4058    Y:      572     Z:      976
15:54:18.622 -> X:      9.42    Y:      1.04    Z:      0.91    m/s^2
15:54:18.655 -> Landscape Right Front
15:54:18.689 ->
15:54:19.136 -> X:      4028    Y:      162     Z:      276
15:54:19.136 -> X:      9.63    Y:      0.41    Z:      0.68    m/s^2
15:54:19.170 -> Landscape Right Front
15:54:19.205 ->
```

Figure6: LANDSCAPE RIGHT FRONT

```
15:55:30.475 -> X:       -9.72   Y:      0.45    Z:      2.04    m/s^2
15:55:30.510 -> Landscape Left Front
15:55:30.545 ->
15:55:30.988 -> X:       -3994   Y:      38      Z:      954
15:55:30.988 -> X:       -9.58   Y:      0.11    Z:      2.31    m/s^2
15:55:31.057 -> Landscape Left Front
15:55:31.057 ->
15:55:31.502 -> X:       -3978   Y:      212     Z:      988
15:55:31.536 -> X:       -9.49   Y:      0.43    Z:      2.41    m/s^2
15:55:31.571 -> Landscape Left Front
15:55:31.571 ->
15:55:32.018 -> X:       -3998   Y:      190     Z:      952
15:55:32.053 -> X:       -9.52   Y:      0.49    Z:      2.35    m/s^2
15:55:32.086 -> Landscape Left Front
15:55:32.121 ->
```

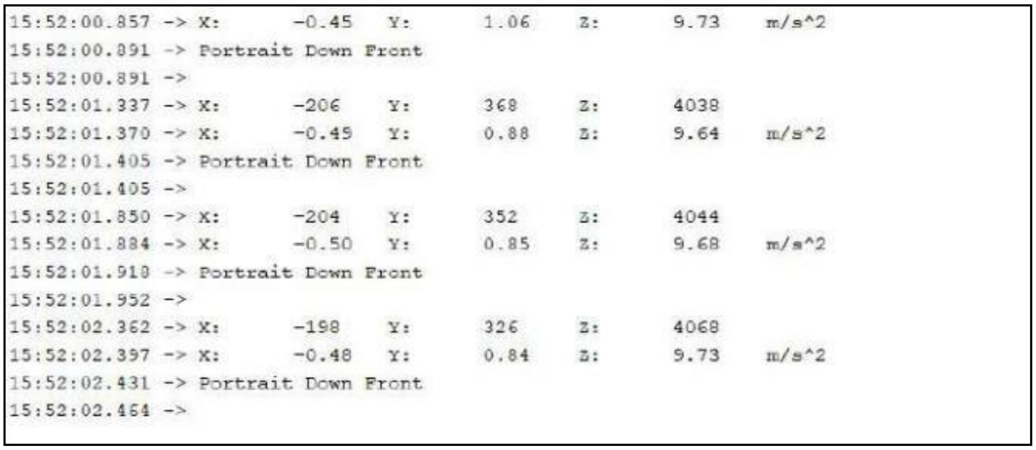
Figure7: LENDSCAPE LEFT FRONT

```
15:53:46.848 -> X:       -0.16   Y:      0.78    Z:      -9.56   m/s^2
15:53:46.884 -> Portrait Down Back
15:53:46.917 ->
15:53:47.360 -> X:       -90     Y:      278     Z:      -4022
15:53:47.395 -> X:       -0.19   Y:      0.64    Z:      -9.66   m/s^2
15:53:47.429 -> Portrait Down Back
15:53:47.429 ->
15:53:47.875 -> X:       -70     Y:      278     Z:      -4006
15:53:47.908 -> X:       -0.20   Y:      0.68    Z:      -9.61   m/s^2
15:53:47.942 -> Portrait Down Back
15:53:47.977 ->
15:53:48.389 -> X:       0       Y:      278     Z:      -3978
15:53:48.422 -> X:       -0.00   Y:      0.65    Z:      -9.50   m/s^2
15:53:48.457 -> Portrait Down Back
15:53:48.491 ->
```

Figure8: PORTRAIT DOWN BACK

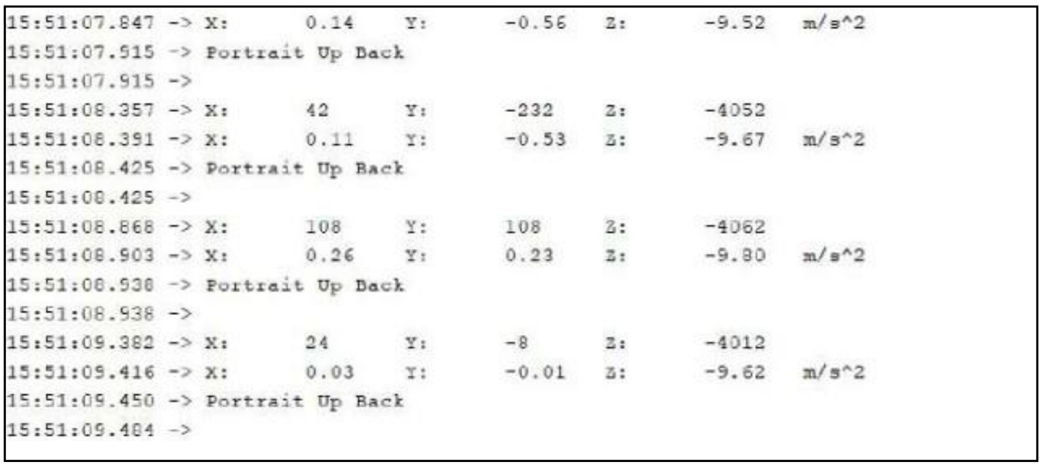
```
15:48:34.139 -> X:       -0.04   Y:      0.20    Z:      9.76    m/s^2
15:48:34.174 -> Portrait Up Front
15:48:34.207 ->
15:48:34.622 -> X:       -28     Y:      74      Z:      4050
15:48:34.655 -> X:       -0.01   Y:      0.21    Z:      9.77    m/s^2
15:48:34.690 -> Portrait Up Front
15:48:34.724 ->
15:48:35.137 -> X:       -36     Y:      102     Z:      4064
15:48:35.170 -> X:       -0.09   Y:      0.23    Z:      9.72    m/s^2
15:48:35.204 -> Portrait Up Front
15:48:35.238 ->
15:48:35.649 -> X:       -26     Y:      86      Z:      4086
15:48:35.683 -> X:       -0.11   Y:      0.16    Z:      9.74    m/s^2
15:48:35.718 -> Portrait Up Front
15:48:35.753 ->
```

Figure9: PORTRAIT UP FRONT



```
15:52:00.857 -> X:       -0.45    Y:      1.06     Z:      9.73    m/s^2
15:52:00.891 -> Portrait Down Front
15:52:00.891 ->
15:52:01.337 -> X:       -206     Y:      368      Z:      4038
15:52:01.370 -> X:       -0.45    Y:      0.88     Z:      9.64    m/s^2
15:52:01.405 -> Portrait Down Front
15:52:01.405 ->
15:52:01.850 -> X:       -204     Y:      352      Z:      4044
15:52:01.884 -> X:       -0.50    Y:      0.85     Z:      9.68    m/s^2
15:52:01.918 -> Portrait Down Front
15:52:01.952 ->
15:52:02.362 -> X:       -198     Y:      326      Z:      4068
15:52:02.397 -> X:       -0.48    Y:      0.84     Z:      9.73    m/s^2
15:52:02.431 -> Portrait Down Front
15:52:02.464 ->
```

Figure10: PORTRAIT DOWN FRONT



```
15:51:07.847 -> X:       0.14     Y:      -0.56    Z:      -9.52   m/s^2
15:51:07.915 -> Portrait Up Back
15:51:07.915 ->
15:51:08.357 -> X:       42       Y:      -232     Z:      -4052
15:51:08.391 -> X:       0.11     Y:      -0.53    Z:      -9.67   m/s^2
15:51:08.425 -> Portrait Up Back
15:51:08.425 ->
15:51:08.868 -> X:       108      Y:      108      Z:      -4062
15:51:08.903 -> X:       0.26     Y:      0.23     Z:      -9.80   m/s^2
15:51:08.938 -> Portrait Up Back
15:51:08.938 ->
15:51:09.382 -> X:       24       Y:      -8       Z:      -4012
15:51:09.416 -> X:       0.03     Y:      -0.01    Z:      -9.62   m/s^2
15:51:09.450 -> Portrait Up Back
15:51:09.484 ->
```

Figure11: PORTRAIT UP BACK

The demo from the examples code is shown below.

```
/**************************************************************/
/*!
@file Adafruit_MMA8451.h
@author K. Townsend (Adafruit Industries)
@license BSD (see license.txt)

This is an example for the Adafruit MMA8451 Accel
breakout board
----> https://www.adafruit.com/products/2019

Adafruit invests time and resources providing this open
source code, please support Adafruit and open-source
hardware by purchasing products from Adafruit!

@section HISTORY

v1.0 - First release
*/
/**************************************************************/

#include <Adafruit_MMA8451.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MMA8451 mma = Adafruit_MMA8451();
```

```
void set up(void)

Serial.begin(9600

);

Serial.println(11 Adafruit MMA8451 test!11 ) ;

i£ (lmma .b e gin())
{
Serial.println(11Couldnt start11 ) ;
while (1)


Serial.println(11 MMA8451 found!11 ) ;
mm a.setR ange (MMA 845 1 RANG
E_2_G);Serial .print(11Range =
11 ) ;
Serial.print(2 << mma .getR ange());
Serial.println(11G11 ) ;


void loop()

// Read the 'raw' data in 14-bit
countsmma.read();
Serial.print(11X:\t11 )
; Serial .pri nt(mma .x);
Serial.print("\tY:\t");
Serial .print(mma.y);
Serial.print("\tZ:\t");
Serial .print(mma.z);
Serial.println();

/* Get a new sensor event
*/sensors_event_t event ;
mma.getEvent(&event );
/* Display the results (accelerati on is meas ured in
* m/s. 2 ) */
Serial.print(11X: \t11 ) ;
Serial .print(event .acce leration.x);
Serial.print(11\t11 ) ;
Serial.print(11Y: \t11 ) ;
Serial .print(event .acce leration.y);
Serial .print(11\t11 ) ;
Serial.print(11Z: \t11 ) ;
Serial .print(event .acce leration.z);
Serial .print(11\t11 ) ;
Serial.println(11m/sA 2 11 ) ;
```

```
/* Get the orientati on of the sens or */
uint8 t o = mma.getOrientation();

switch (o)
{
case MMA8451 PL PUF:
Serial.println(11 Portrait      Up Front 11) ;
break ;
case MMA8451 PL PUB:
Serial.println(11 Portrait      Up Back 11) ;
break ;
case MMA8451 PL PDF:
Serial.println(11Portrait Down Front11) ;
break;
case MMA8451 PL PDB:
Serial.println(11Portrait Down Back11) ;
break;
case MMA8451 PL LRF:
Serial.println(11 Landscape    Right Front11) ;
break ;
case MMA8451 PL LRB:
Serial.println(11 Landscape    Right Back11) ;
break ;
case MMA8451 PL LLF:
Serial.println(11Landscape Left Front11) ;
break;
case MMA8451 PL LLB:
Serial.println(11Landscape Left Back11) ;
break;

Serial .println();
delay(500);
)
```

Task 3.3

Pitch, yaw and roll are three dimensional movement of an object. We can use the movement of an airplane to demonstrate it as shown in the figure below Let θ *2* [-π=2*;* π=2]*;* *2* (-π*;* π]*;* φ *2* (-π*;* π] denote the angles which the object rotates along the axis corresponding to pitch, yaw and roll. We havethe following relation between θ*;* φ and *X; Y; Z* where *X; Y; Z* approximate the measured forces along the *XY Z* axes. We need to first normalize *X; Y; Z* such that:

$$ax = \frac{X}{\sqrt{X^2 + Y^2 + Z^2}}$$

$$ay = \frac{Y}{\sqrt{X^2 + Y^2 + Z^2}}$$

$$az = \frac{Z}{\sqrt{X^2 + Y^2 + Z^2}}$$

$$\theta = asin(ax)$$

$$\phi = atan(\frac{ay}{cos\theta}, \frac{az}{cos\theta})$$

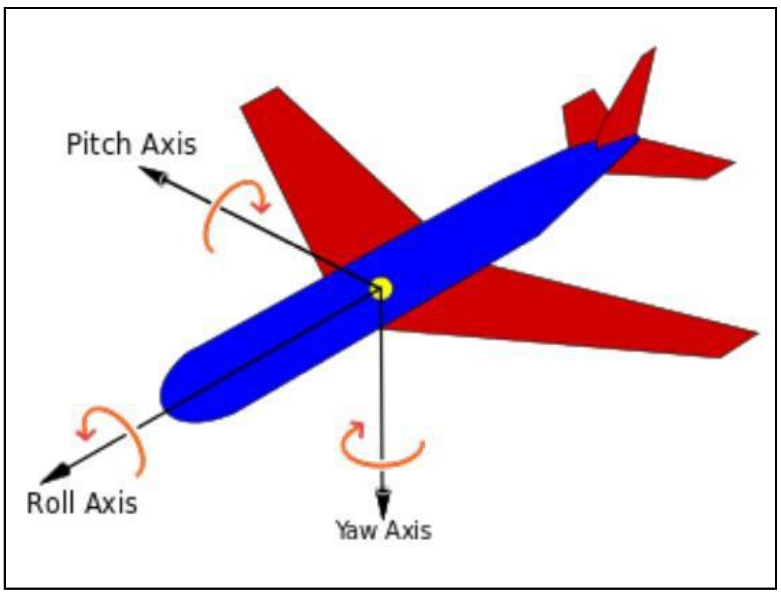


Figure12: Illustration of three dimensional movements.

On the Arduino side, we modified the code of the demo program, so that the Processing programis able to read what it writes on the serial port. On the Processing side we filled the missing partsof the code in order to calculate the pitch and the roll. We ran the GUI to see the output shown inthe figures 13 and 14. The modified code and the output is shown below.