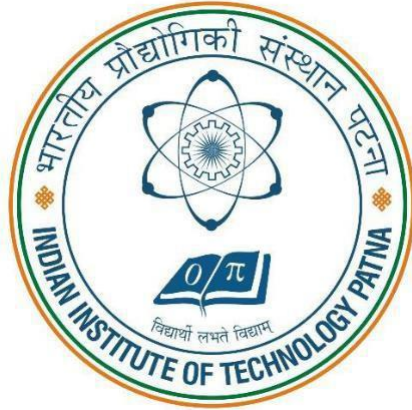


Indian Institute of Technology Patna



Mechatronics, Instrumentation And Controls Laboratory Lab 8 Report

Topic:

Basic operation of Arduino
Microcontroller and control of servo
Motor

Submitted by: Aditya Shah(2011mt02)
(MTech-Mechatronics)

Lab In-charge:

Dr. Atul Thakur

(Assistant Professor)

Department of Mechanical Engineering
IIT Patna

1. Aim of the Experiments.

1. Control LED brightness using PWM by reading Serial commands 0 -9. So “0” means lowest brightness, “9” means highest brightness.
2. Oscillate LED brightness in a sinusoidal wave which is shifted
3. Do task 1 and task 2 without AnalogWrite()
4. Using LDR as a lux sensor
 - a. Signal processing of LDR as a sensor
 - b. ADC of the signal
 - c. Digital filtering –smoothing (Low pass digital filter)
 - d. Calibration using Android phones lux sensor.

2. Pre-Requisites/Components Required

1.1,1.2,1.3:

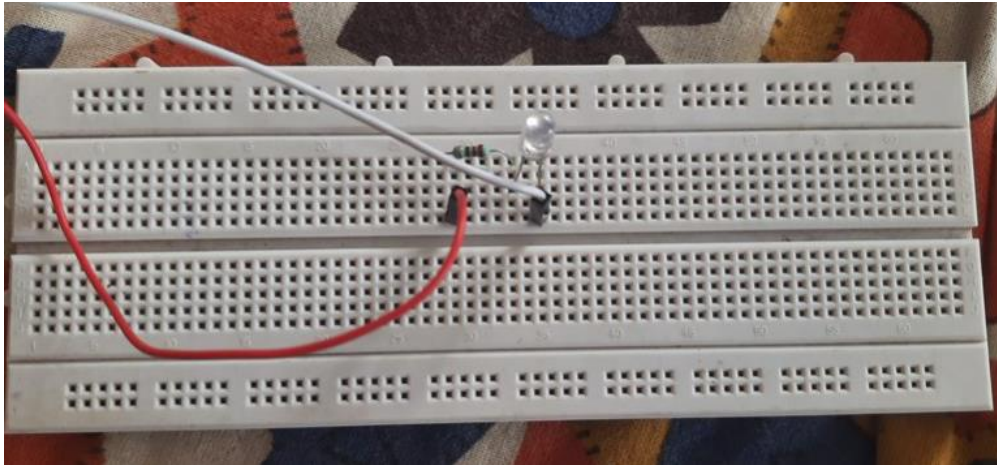
- Led: White
- Resistor: 220 Ω
- Arduino Uno with Programming Cable
- Arduino IDE 1.8.15
- Breadboard and Jumper wires.

1.4:

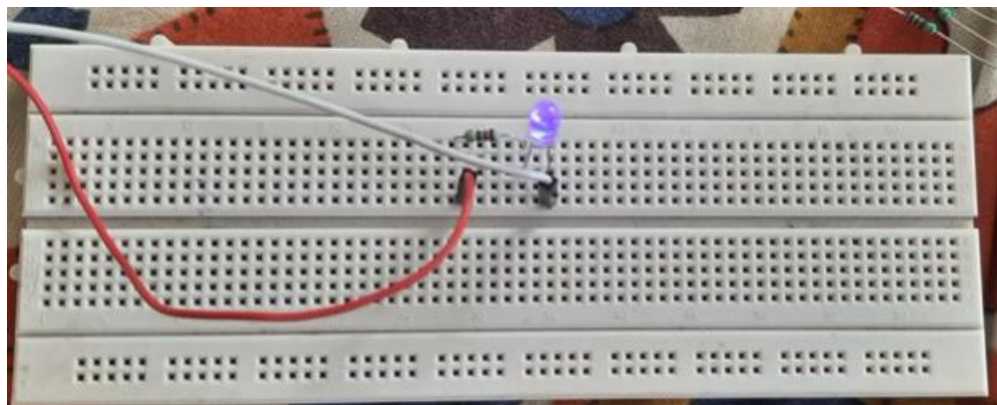
- LDR: 5mm
- Resistor: 10K Ω
- Lux Meter (Android APP)
- Arduino Uno with Programming Cable
- Arduino IDE 1.8.15
- Breadboard and Jumper wires.

3. Circuit Diagram

1.1

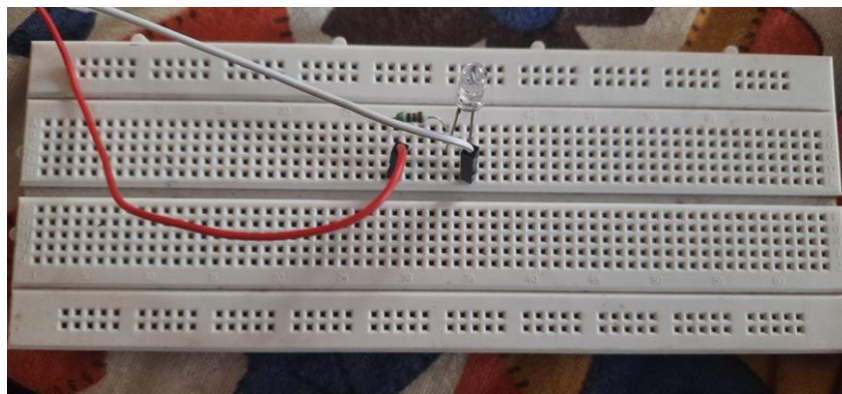


1.2

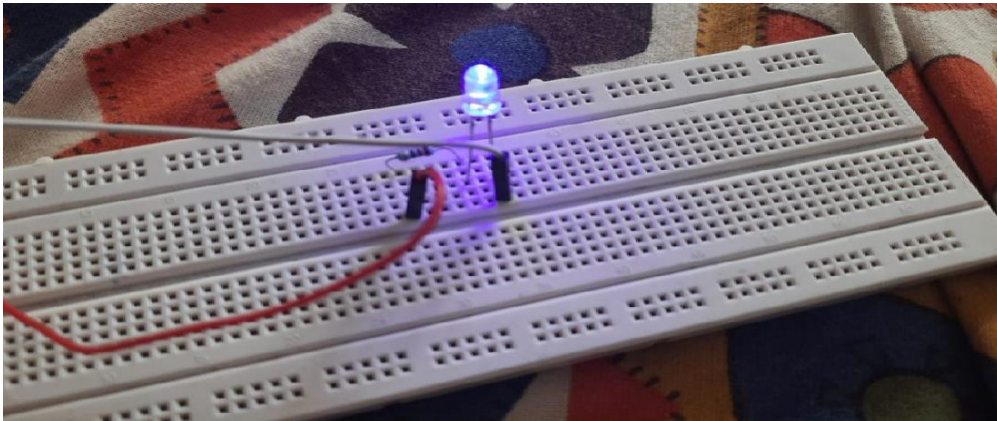




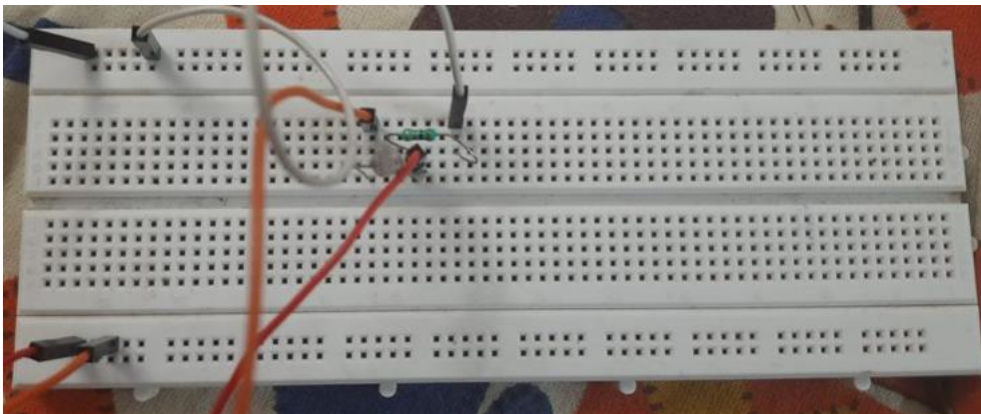
1.3 Task-1:

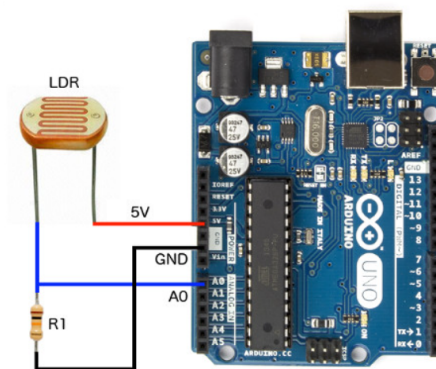


Task-2:



1.4





Schematics Source: <http://cactus.io/hooks/sensors/light/ldr/hookup-arduino-to-ldr-sensor>

4. Procedure

1.1

- Connect the circuit as shown in circuit diagram section.
- Then code was written in IDE, to control LED brightness, by reading Serial commands 0 -9, given by user.
- “0” means lowest brightness, “9” means highest brightness.
- Comments are included in the code.
- Demonstration video is made. Link given at last.

1.2

- Connect the circuit as shown in circuit diagram section.
- Then code was written in IDE, to Oscillate LED brightness in a sinusoidal wave which is shifted above by 127.5.
- Shifting is done because PWM range for writing analog values is in 0-255.
- Comments are included in the code.
- Demonstration video is made. Link given at last.

1.3

Task-1

- Connect the circuit as shown in circuit diagram section.
- Then code was written in IDE, to control LED brightness, by reading Serial commands 0 -9, given by user.
- “0” means lowest brightness, “9” means highest brightness.
- Here, I have used `delayMicroseconds()` instruction, to emulate the PWM concept. With this instruction, according to required duty cycle, T_{on} and T_{off} is calculated and then using this delay function, we got averaging affect, thus getting varying voltage output, which is basically the PWM concept.
- Comments are included in the code.
- Demonstration video is made. Link given at last.

Task-2

- Connect the circuit as shown in circuit diagram section.
- Then code was written in IDE, to Oscillate LED brightness in a sinusoidal wave which is shifted above by 127.5.
- Here, I have used `delayMicroseconds()` instruction, to emulate the PWM concept. With this instruction, according to required duty cycle, T_{on} and T_{off} is calculated and then using this delay function, we got averaging affect, thus getting varying voltage

output, which is basically the PWM concept.

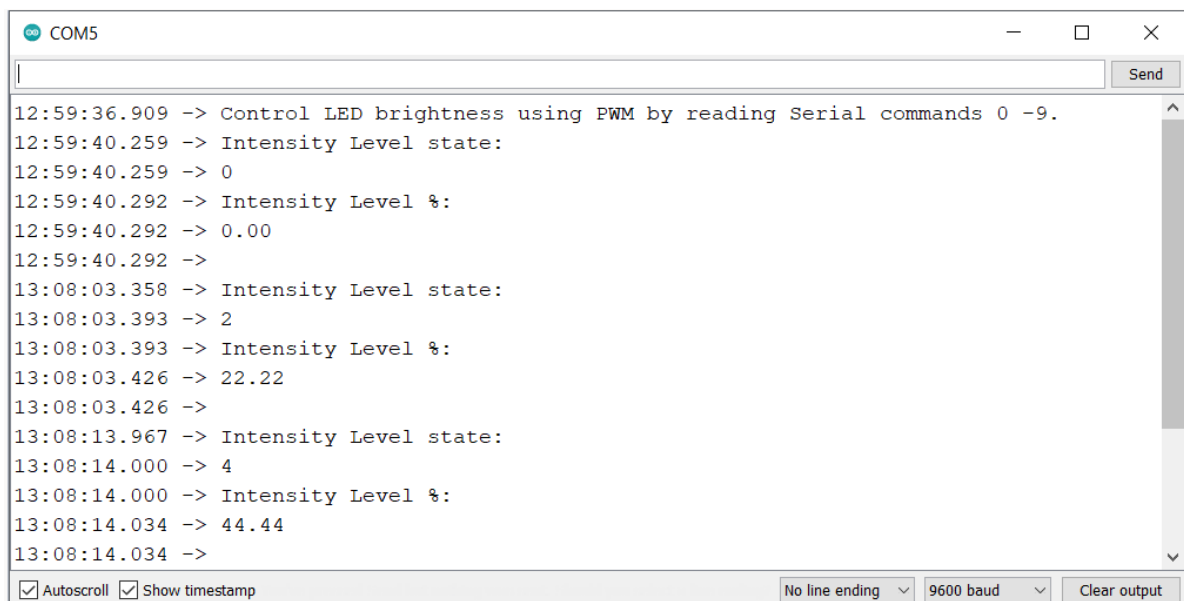
- Comments are included in the code.
- Demonstration video is made. Link given at last.

1.4

- Connect the circuit as shown in circuit diagram section.
- The light intensity was measured using LDR, being a passive element, we have to give an external supply and the other terminal of LDR, we gave it as i/p to Arduino A0 as shown in circuit diagram section..
- Then code was written in IDE, to convert the analog value to digital values, and then Low Pass filter is applied by a smoothening function, to get smooth values and then it is calibrated using Android phones lux sensor, taken as standard.
- Comments are included in the code.
- Demonstration video is made. Link given at last.

5. Results

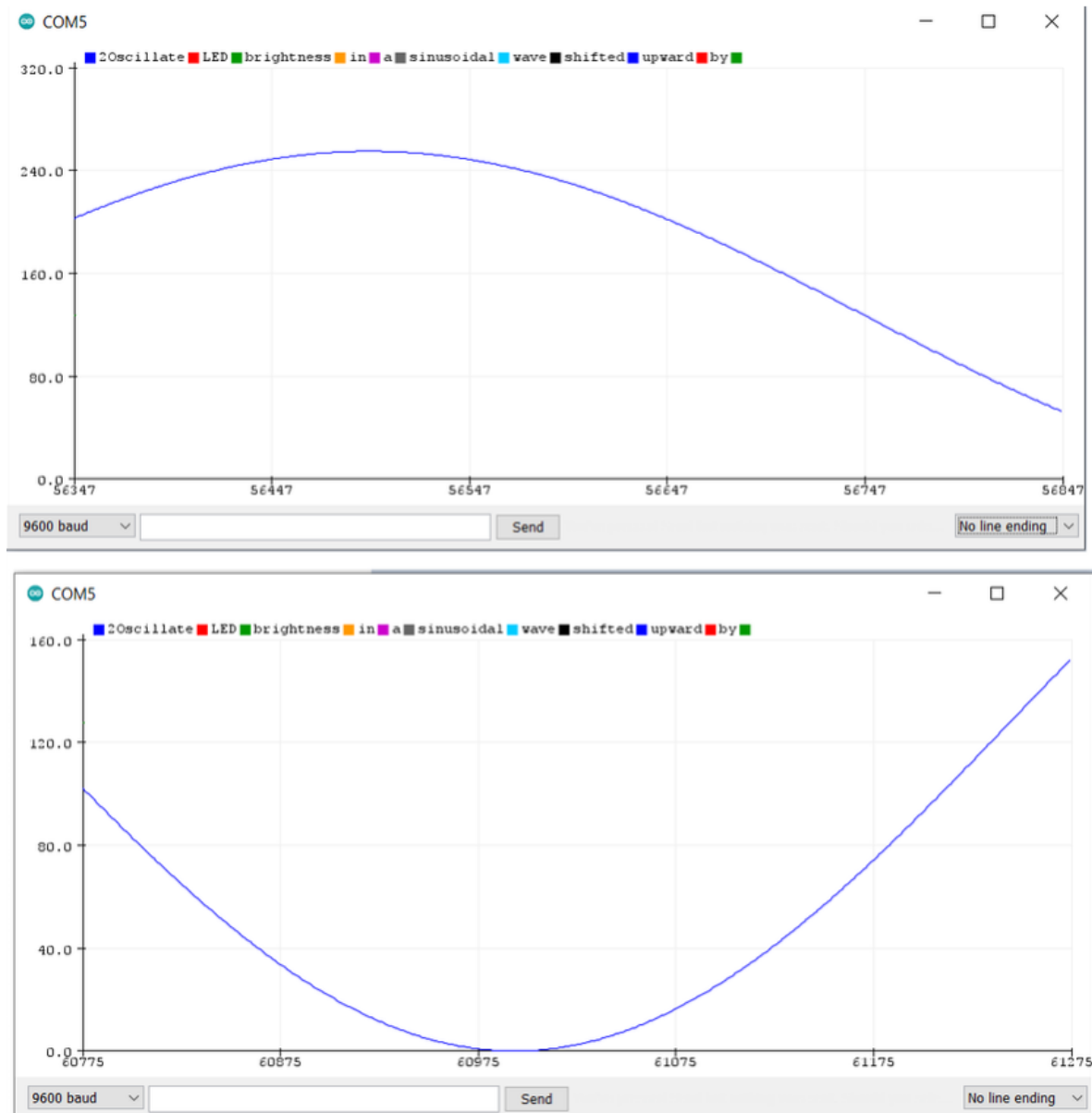
1.1



```
COM5
12:59:36.909 -> Control LED brightness using PWM by reading Serial commands 0 -9.
12:59:40.259 -> Intensity Level state:
12:59:40.259 -> 0
12:59:40.292 -> Intensity Level %:
12:59:40.292 -> 0.00
12:59:40.292 ->
13:08:03.358 -> Intensity Level state:
13:08:03.393 -> 2
13:08:03.393 -> Intensity Level %:
13:08:03.426 -> 22.22
13:08:03.426 ->
13:08:13.967 -> Intensity Level state:
13:08:14.000 -> 4
13:08:14.000 -> Intensity Level %:
13:08:14.034 -> 44.44
13:08:14.034 ->
☒ Autoscroll ☒ Show timestamp
No line ending 9600 baud Clear output
```

Led Voltage variation(0-9)

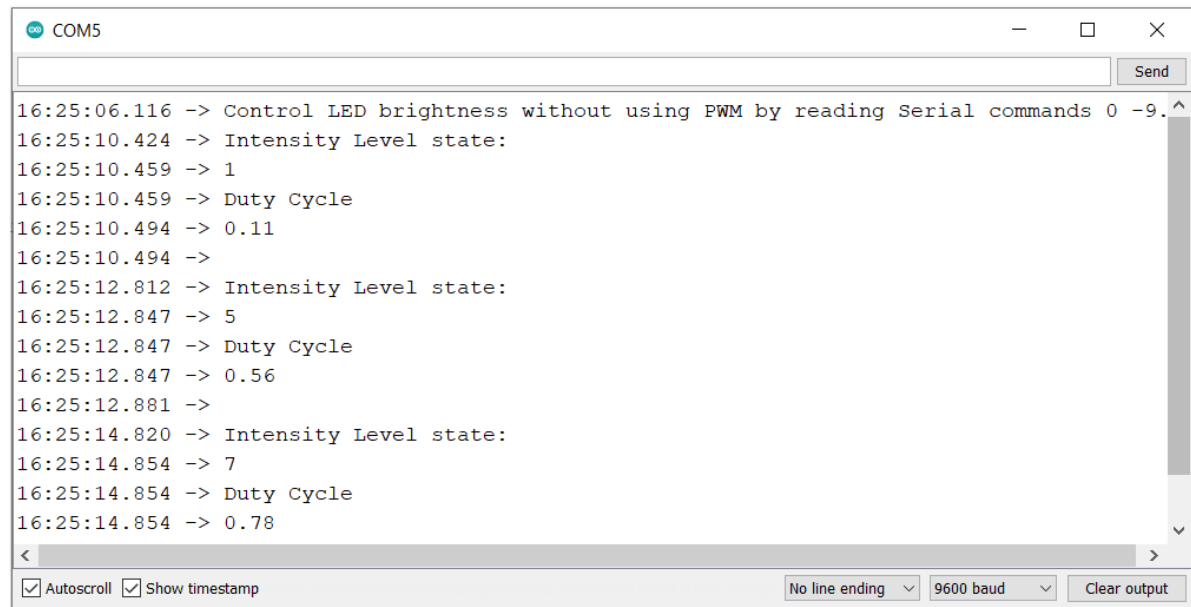
1.2



Led Voltage variation (Sinusoidal)

1.3

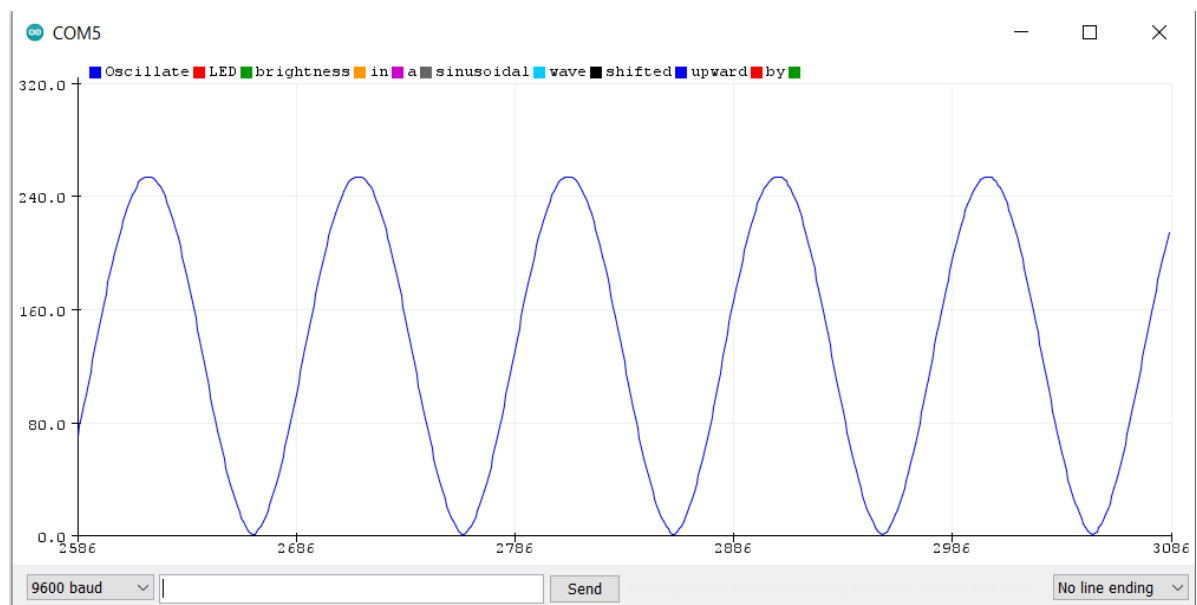
Task-1



```
COM5
16:25:06.116 -> Control LED brightness without using PWM by reading Serial commands 0 -9.
16:25:10.424 -> Intensity Level state:
16:25:10.459 -> 1
16:25:10.459 -> Duty Cycle
16:25:10.494 -> 0.11
16:25:10.494 ->
16:25:12.812 -> Intensity Level state:
16:25:12.847 -> 5
16:25:12.847 -> Duty Cycle
16:25:12.847 -> 0.56
16:25:12.881 ->
16:25:14.820 -> Intensity Level state:
16:25:14.854 -> 7
16:25:14.854 -> Duty Cycle
16:25:14.854 -> 0.78
☒ Autoscroll ☒ Show timestamp
No line ending 9600 baud Clear output
```

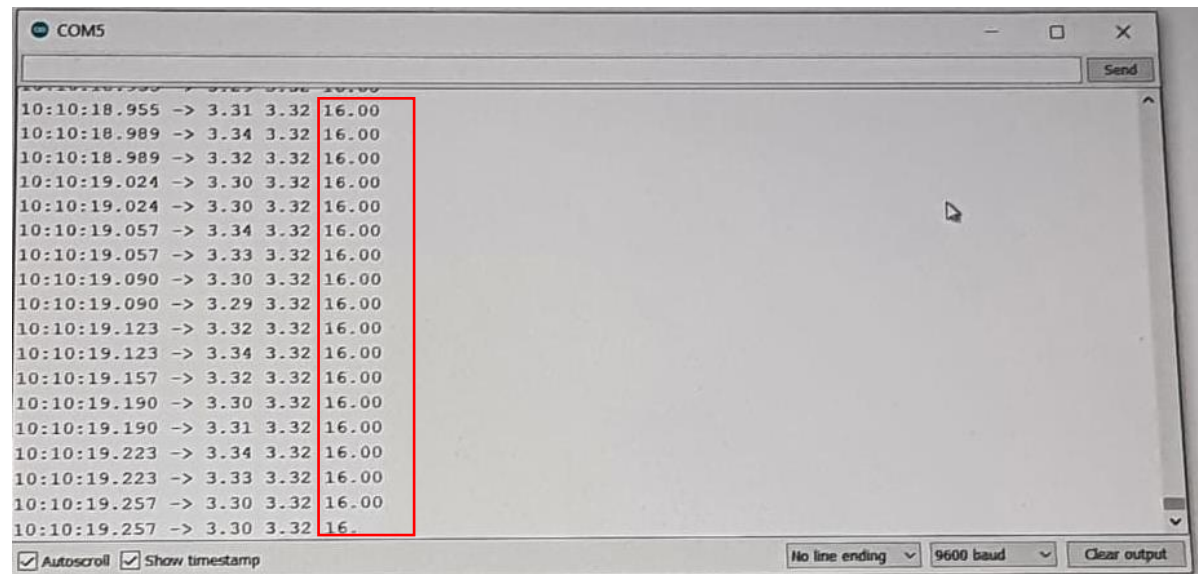
Led Voltage variation(0-9)

Task-2

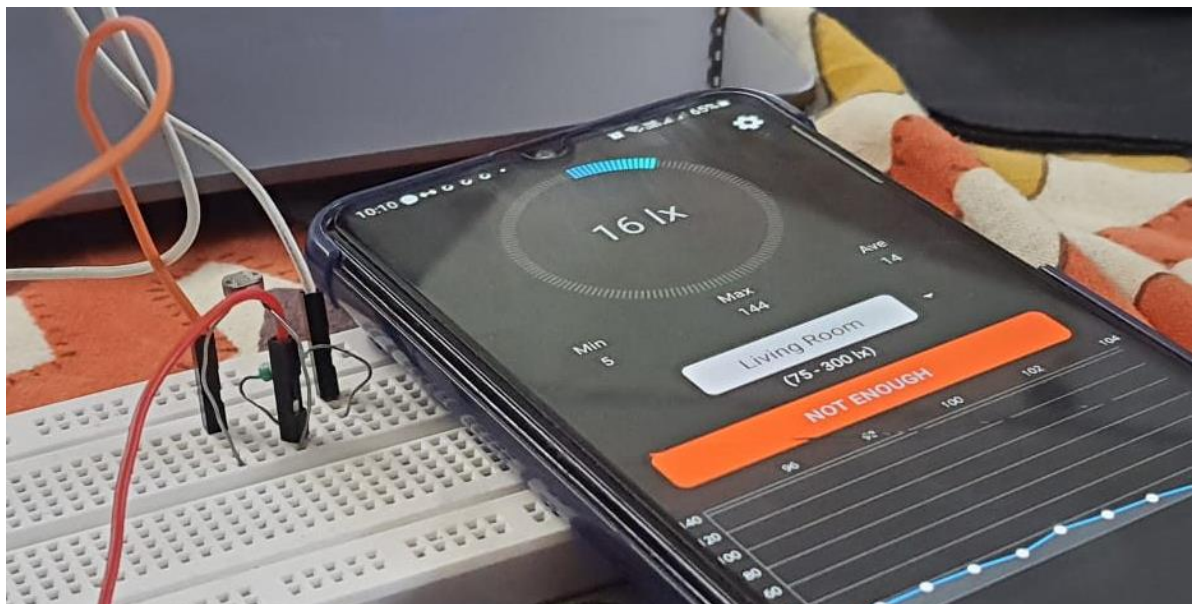


Led Voltage variation(Sinusoidal)

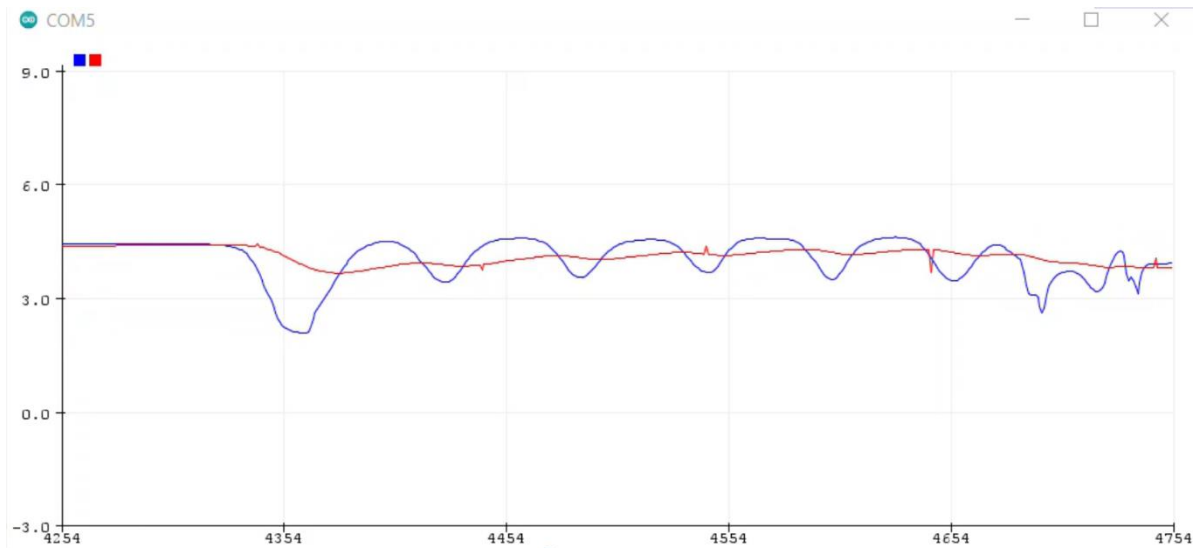
1.4



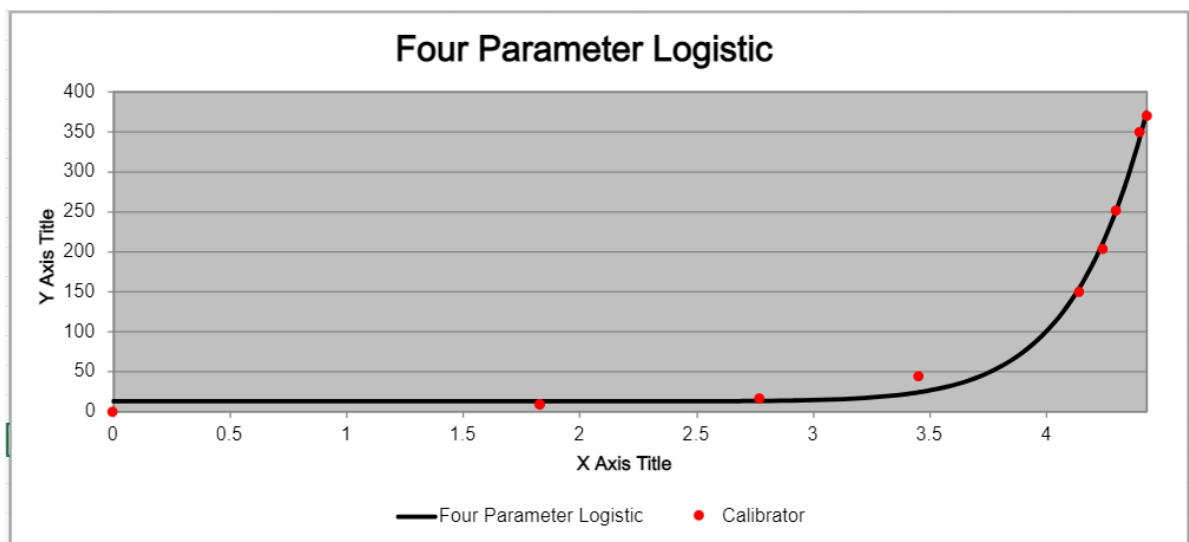
Calibrated results



Phone Lux meter app results



Smoothing Effect (Red Color)



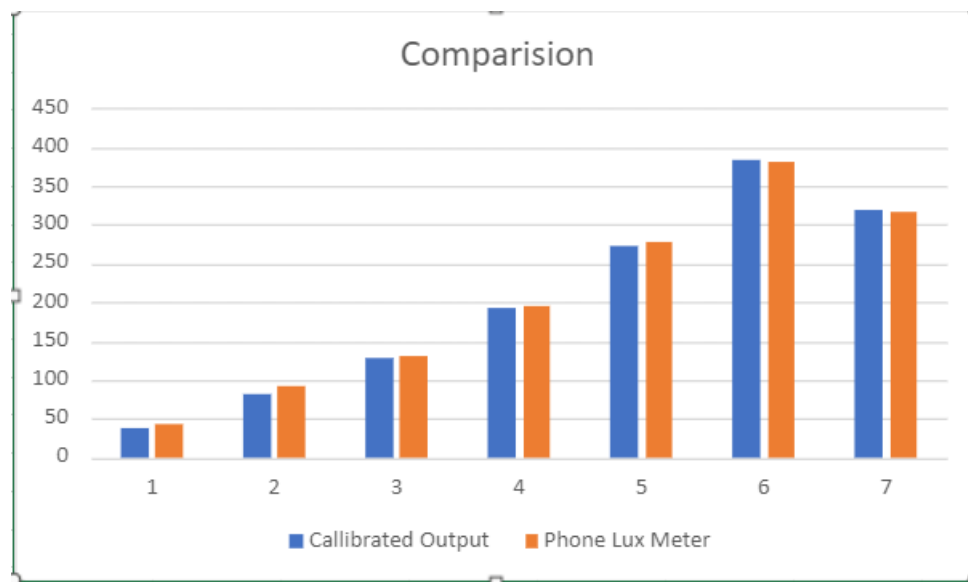
Smoothed sensor value versus android lux sensor reading

Used below formulation for calibration: -

$$y = d + \frac{a - d}{1 + \left(\frac{x}{c}\right)^b}$$

$$y = 141520040 + \frac{12.60103 - 141520040}{1 + \left(\frac{x}{11.179369}\right)^{13.91121}}$$

X Axis Title	Y Axis Title
0	0
1.83	9
1.83	9
2.77	17
3.45	44
4.14	150
4.24	204
4.3	252
4.4	350
4.43	370



6. Inference.

In 1.1,1.2 and 1.3, the result obtained was desired and also checked with multimeter, and thus using PWM concept, we were able to vary the LED brightness. Also I learnt about various Instruction, and how to emulate PWM concept using delayMicroseconds() instruction. And the results were verified via Serial Plotter.

In the 1.4, I learnt how to use LDR as a LUX meter. I found difficulty in calibration part. The LDR variation with LIGHT intensity variation was highly nonlinear. To tackle this ,I used an curve fitting technique, for better calibration. I used Non-Linear 4PL technique for calibration, which give good result. During experimentation, it was found for low LUX value, the calibrated value were little lower than phone lux meter app and opposite was observed for higher LUX value case. Also, since the ambient light sensor of phone is located at top, near the camera. For different light intensity, there was little deviation of distance, which may be one of the reasons for the deviation of reading of the two. And since the ADC is of 10 bits, we can get a very good resolution, which is capable to sense $5/1023 \text{ V}=4.88\text{mV}$ variation coming due to light intensity variation recorded by LDR.

Question: -

- 1. Write features of Atmega328p and ADC of Atmega328p.**
- 2. Write a note on calibration, give two examples?**

Answer: -

1. ATMEGA328P is high performance, low power controller from Microchip. ATMEGA328P is an 8-bit microcontroller based on AVR RISC architecture. It is the most popular of all AVR controllers as it is used in ARDUINO boards. It is a 28 pin chip.

ATMEGA328P – Features	
CPU	8-bit AVR
Number of Pins	28
Operating Voltage (V)	+1.8 V TO +5.5V
Number of programmable I/O lines	23
Communication Interface	Master/Slave SPI Serial Interface(17,18,19 PINS) [Can be used for programming this controller] Programmable Serial USART(2,3 PINS) [Can be used for programming this controller] Two-wire Serial Interface(27,28 PINS)[Can be used to connect peripheral devices like Servos, sensors and memory devices]
JTAG Interface	Not available
ADC Module	6channels, 10-bit resolution ADC
Timer Module	Two 8-bit counters with Separate Prescaler and compare mode, One 16-bit counter with Separate Prescaler,compare mode and capture mode.
Analog Comparators	1(12,13 PINS)
DAC Module	Nil
PWM channels	6
External Oscillator	0-4MHz @ 1.8V to 5.5V 0-10MHz @ 2.7V to 5.5V 0-20MHz @ 4.5V to 5.5V
Internal Oscillator	8MHz Calibrated Internal Oscillator
Program Memory Type	Flash
Program Memory or Flash memory	32Kbytes[10000 write/erase cycles]
CPU Speed	1MIPS for 1MHz
RAM	2Kbytes Internal SRAM
EEPROM	1Kbytes EEPROM
Watchdog Timer	Programmable Watchdog Timer with Separate On-chipOscillator
Program Lock	Yes
Power Save Modes	Six Modes[Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby]
Operating Temperature	-40°C to +105°C(+105 being absolute maximum, -40 being absolute minimum)

Features: - Some notable features are given below:

ADC Features: -

The Atmel ATmega328P microcontroller used on the Arduino Uno has an analog-to-digital conversion (ADC) module capable of converting an analog voltage into a 10-bit number from 0 to 1023. The input to the module can be selected to come from any one of six inputs on the chip. While this gives the microcontroller the ability to convert the signals from six different analog sources into 10-bit values only one channel can be converted at a time. The ADC can convert signals at a rate of about 15 kSPS (samples per second.)

10 bit ADC of ATmega328p has the following features: -

- **10 bit resolution (0-1023 steps)**
- **Sampling rate of upto 15k SPS**
- **6 Multiplexed single ended channels in 28 pin DIP version.**
- **2 additional single ended channels in 32 pin TQFP and QFN packages .**
- **Built in temperature sensor.**
- **ADC Input Voltage Range of 0 -Vcc (0-5V)**
- **Selectable 1.1V ADC Reference Voltage**
- **Sleep Mode Noise Canceler**

2.) Formally, calibration is the documented comparison of the measurement device to be calibrated against a traceable reference device.

The reference standard may be also referred as a “calibrator.” Logically, the reference is more accurate than the device to be calibrated.

When we make a calibration and compare two devices, we may find out there is some difference between the two. So, we have to adjust the device under test to measure correctly. This process is often called adjustment or trimming.

Formally, calibration does not include adjustment, but is a separate process. In everyday language the word calibration sometimes also includes possible adjustment. But as mentioned, the adjustment is a separate process according to most formal sources.

Thus, in simple manner we can say Calibration is a comparison between a known measurement (the standard) and the measurement

using your instrument.

Calibration of our measuring instruments has two objectives:

- It checks the accuracy of the instrument
- It determines the traceability of the measurement.

Example: -

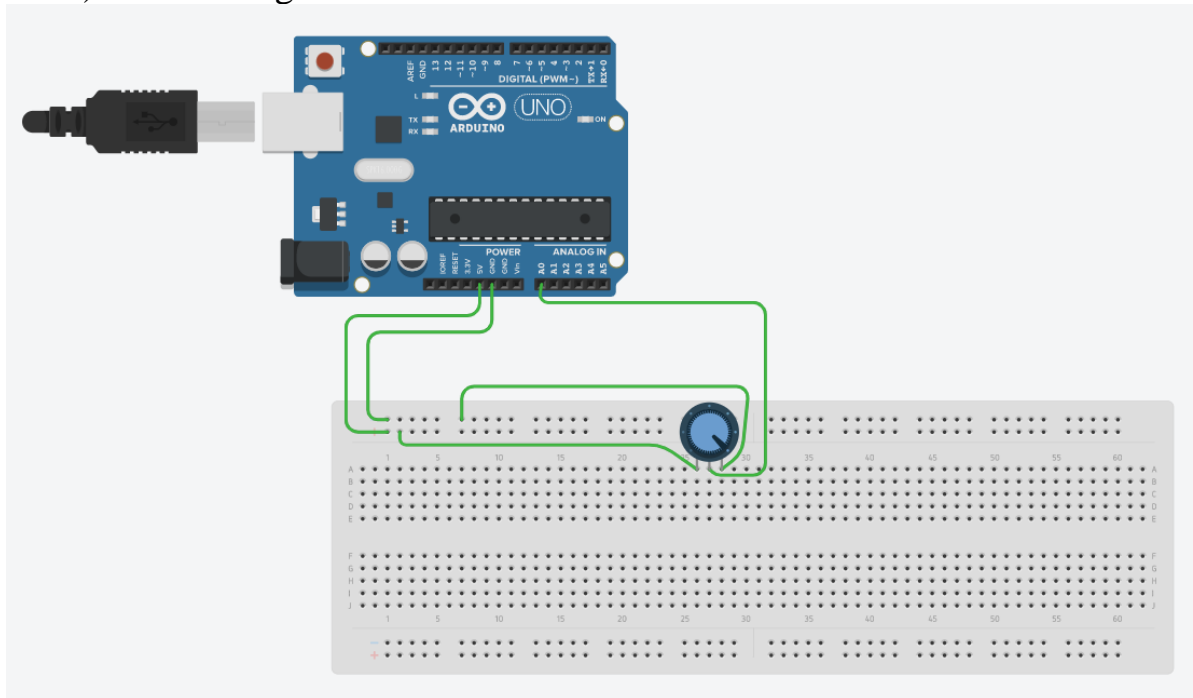
- A thermometer could be calibrated against a standard device (boiling point method or the freezing point method) so the error of indication or the correction is determined, and adjusted (e.g. via calibration constants) so that it shows the true temperature in Celsius at specific points on the scale.
- In experiment 1.4, we used LDR as a LUX meter and after smoothening the result, we calibrated the same with the help of Phone Lux Meter App (Taken as Standard). Thus, after calibration, we make use of an curve fitting method, to correct the reading obtained from the test device.
- Load Cell Calibration is an adjustment or set of corrections that are performed on a load cell, or instrument (amplifier), to make sure that the sensor operates as accurately, or error-free, as possible. The calibration tasks involve comparing the accuracy of the sensors with the recognised standard value weight. The sensor elements need to be adjusted when the accuracy falls below the standard.

Extra Question: -

1. Calibrate the rotary potentiometer to get angles in degree ?
2. Control DC motor speed using PWM.
3. Attach the DC motor & potentiometer, and control the angle of the DC motor.
 - Proportional control = error * Kp
 - Use the calibrated potentiometer code and control dc motor code to get this task done.

Answer

1.) Circuit Diagram: -



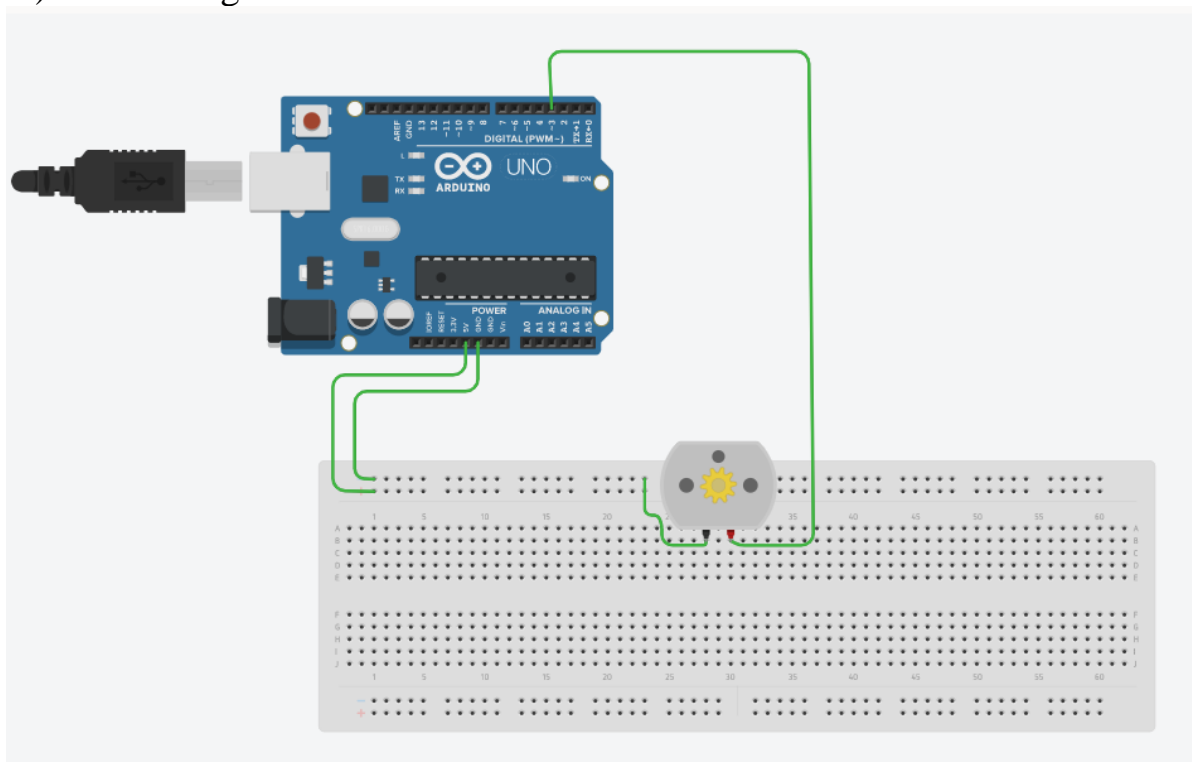
Components: -

Component List			Download CSV
Name	Quantity	Component	
U1	1	Arduino Uno R3	
Rpot1	1	250 k Ω Potentiometer	

Code: -

```
1 // C++ code
2 //
3 int potpin = 0; // analog pin used to connect the potentiometer
4 int val; // variable to read the value from the analog pin
5
6 void setup()
7 {
8   Serial.begin(9600);
9 }
10
11 void loop()
12 {
13   val = 1023 - analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023)
14   val = map(val, 0, 1023, 0, 270); // Angle starting from left to right side.(each segment=45 deg)
15   Serial.println(val);
16 }
17
18 }
```


2.)Circuit Diagram: -



Components: -

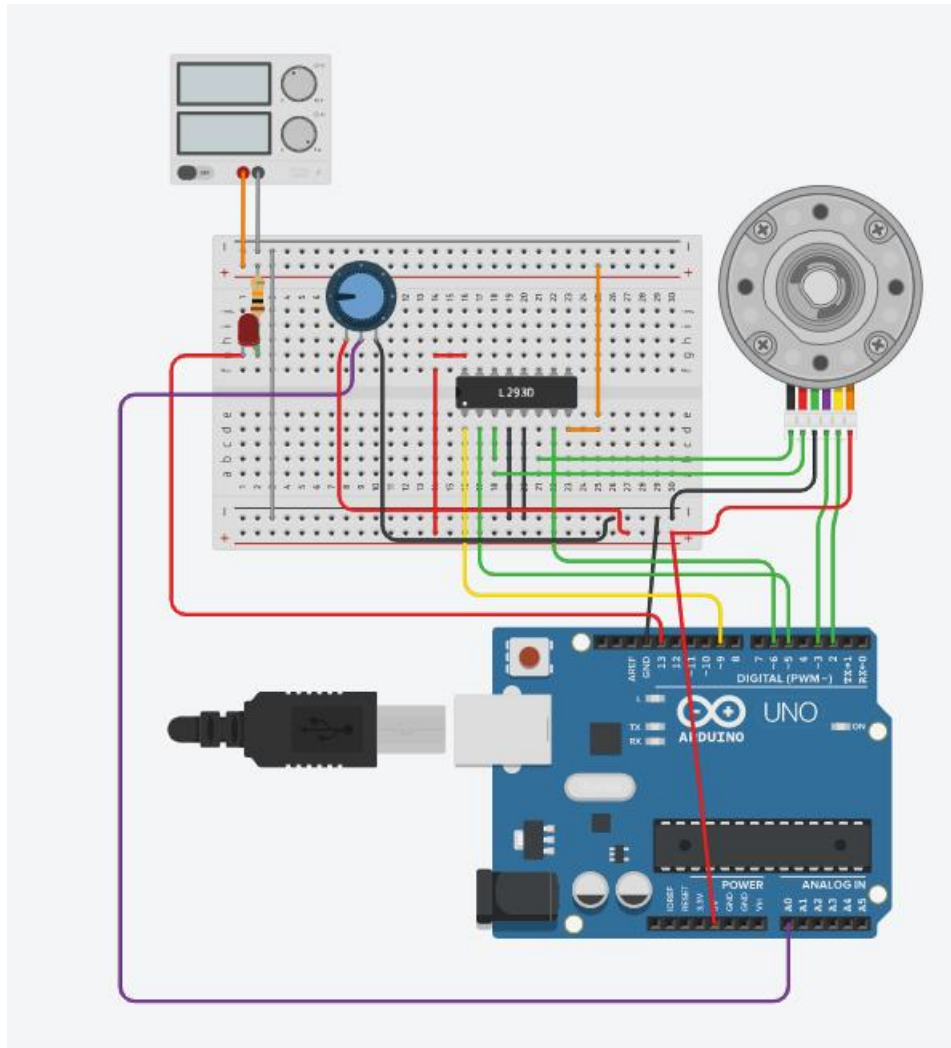
Component List [Download CSV](#)

Name	Quantity	Component
U1	1	Arduino Uno R3
M2	1	DC Motor


Code: -

```
Text [v] [Download] [Save] [Run] 1 (Arduino Uno R3) v
1 // C++ code
2
3 int motor = 3;
4
5 void setup()
6 {
7   Serial.begin(9600);
8   pinMode(motor, OUTPUT);
9   Serial.println("Enter values between 0 - 9");
10 }
11
12 void loop()
13 {
14   if(Serial.available())
15   {
16     int speed = Serial.parseInt(); //Receive Value from serial m
17     val = map(speed, 0, 9, 0, 255);
18
19     Serial.print(speed);
20     Serial.print(" ");
21
22     analogWrite(motor, val); //sets the motors speed
23
24   }
25 }
26
27
```

3.)Circuit Diagram: -



Components: -

Component List			 Download CSV
Name	Quantity	Component	
U1	1	Arduino Uno R3	
M1	1	45 DC Motor with Encoder	
P1	1	12 , 4.9 Power Supply	
U2	1	H-bridge Motor Driver	
R1	1	10 kΩ Resistor	
D1	1	Red LED	
Rpot1	1	250 kΩ Potentiometer	

Code: -

```
Text 1 (Arduino Uno R3)
1  /* 45 RPM HD Premium Planetary Gear Motor w/Encoder */
2
3  // POT
4  int potpin = 0; // analog pin used to connect the potentiometer
5  int val; // variable to read the value from the analog pin
6
7  // motor control pin
8  const int motorDirPin = 5; // Input 1
9  const int motorPWMPin = 6; // Input 2
10 const int EnablePin = 9; // Enable
11 const int LED = 13;
12 // encoder pin
13 const int encoderPinA = 2;
14 const int encoderPinB = 3;
15 int encoderPos = 0;
16 // encoder value change motor turn angles
17 const float ratio = 360./188.611/48.;
18 // 360. -> 1 turn
19 // 188.611 -> Gear Ratio
20 // 48. -> Encoder: Countable Events Per Revolution (Motor Shaft)
21 // testing
22 // P control
23 float Kp = 30;
24 float targetDeg;
25
26 void doEncoderA()
27 {
28   encoderPos += (digitalRead(encoderPinA)==digitalRead(encoderPinB))?1:-1;
29 }
30 void doEncoderB()
31 {
32   encoderPos += (digitalRead(encoderPinA)==digitalRead(encoderPinB))?-1:1;
33 }
34
35 void doMotor(bool dir, int vel)
36 {
37   digitalWrite(motorDirPin, dir);
38   digitalWrite(LED, dir);
39   analogWrite(motorPWMPin, dir?(255 - vel):vel);
40 }
41
42 void setup()
43 {
44   Serial.begin(9600);
45
46   pinMode(encoderPinA, INPUT_PULLUP);
47   attachInterrupt(0, doEncoderA, CHANGE);
48
49   pinMode(encoderPinB, INPUT_PULLUP);
50   attachInterrupt(1, doEncoderB, CHANGE);
51
52   pinMode(LED, OUTPUT);
53   pinMode(motorDirPin, OUTPUT);
54   pinMode(EnablePin, OUTPUT);
55 }
56
57 void loop()
58 {
59
60
61   // POT CALIBRATION
62   val = 1023 - analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023)
63                                     // Angle starting from left to right side.(each segment=45 deg)
64   val = map(val, 0, 1023, 0, 270); // scale it to use it with the servo (value between 0 and 180)
65   targetDeg= val ;
66   Serial.println(val);
67   float motorDeg = float(encoderPos)*ratio;
68   //Control signal = P gain * (target value-present value)
69   float error = targetDeg - motorDeg;
70   float control = Kp*error;
71
72   digitalWrite(EnablePin, 255);
73
74   doMotor((control>=0)?HIGH:LOW, min(abs(control), 255));
75
76   Serial.print("encoderPos : ");
77   Serial.print(encoderPos);
78   Serial.print("    motorDeg : ");
79   Serial.print(float(encoderPos)*ratio);
80   Serial.print("    error : ");
```

To View the Demonstration, Output and Circuit Image's: -
Click on the following Link: -

- [Exp-1.1- Control LED brightness using PWM](#)
- [Exp-1.2- Oscillate LED brightness in a sinusoidal wave](#)
- [Exp-1.3- Emulate PWM using delayMicroseconds\(\) instruction](#)
- [Exp-1.4- Using LDR as a lux sensor](#)
- [Extra-1- Calibrate the rotary potentiometer](#)
- [Extra-2- Control DC motor speed using PWM](#)
- [Extra-3- Control the angle of the DC motor](#)