

PROJECT REPORT

ON

Development of the MCU (aTmega328) based servo stabilised platform to maintain predetermined angle.

Training Duration: 1st July 2022 – 15th Aug 2022

HELD AT



**AERIAL DELIVERY RESEARCH & DEVELOPMENT ESTABLISHMENT
DEFENCE RESEARCH & DEVELOPMENT ORGANIZATION**

AGRA CANTT – 282001

SUBMITTED BY

Mohammad Irfan Quraishi

B.Tech(3rd year)

Instrumentation and Control Engineering

Institute of Engineering and Rural Technology Prayagraj

Under the supervision of:

Mrs. Mohini Kumari

Technical Officer 'B'

ADRDE (DRDO)

Reviewed by:

Mr. Mahboob Ahmed

Scientist 'E'

ADRDE (DRDO)

ACKNOWLEDGEMENT

Foremost I would like to thank Dr. Manoj Kumar (Director ADRDE) for conducting this training program, providing appropriate learning atmosphere and EI (Electronics and Instrumentation) laboratory with proper facilities without which this training could not have been completed.

I am extremely grateful to, Mr. Mahboob Ahmed, Scientist 'E' and Alok Kumar Jain, Technical Officer 'B' for their revered guidance and supervision which led to the completion of this project. They were always there to help, providing us with all the necessary resources and guidance which helped in successful completion of this training.

I would also like to thank, Mr. Puneet Parihar, Scientist 'B', Mrs. Mohini Kumari, Technical Officer 'B', Mr. Sachin Rajput, Technical Officer 'A', Mr. Ram Balwan Meena STA 'B' for giving their precious time and support in EI lab during the project work. Finally, I would like to thank all my friends from different institutes who had gathered here for various summer projects. The time we spent together has been a great knowledgeable experience.

Mohammad Irfan Quraishi

ABOUT ADRDE

Aerial Delivery Research and Development Establishment (ADRDE), is a pioneer establishment in the country engaged in design and development of flexible aerodynamics decelerators commonly known as parachutes and the allied technologies. This establishment has been recognized to improve its technical and administrative efficiency as well as to increase the work output and to have better interaction between the scientific communities of this establishment. Four technology groups in the field of aerodynamics, mechanical, instrumentation and textile for assistance in design support their project divisions. ADRDE is an ISO 9001:2015 certified lab. This establishment has been responding well to meet the ever-increasing requirements of the Services & Space organisation by developing state-of-the-art systems. An independent information technology division has also been treated to control and monitor the program on the project as well as the carry out micro and macro planning, to meet the user requirements effectively.

CONTENTS

| | |
|--|-----------|
| Abstract | 5 |
| 1. Introduction..... | 6 |
| • Robotics..... | 6 |
| 2. Arduino UNO..... | 8 |
| • Arduino IDE..... | 10 |
| 3. PWM (Pulse Width Modulation) | 12 |
| • Duty Cycle of PWM..... | 12 |
| • Frequency of a PWM..... | 13 |
| • PWM Applications..... | 13 |
| 4. Sensor..... | 14 |
| • Classification of sensor..... | 14 |
| • Inclinator..... | 15 |
| 5. Servo..... | 17 |
| 6. Simple Controller..... | 19 |
| • Proportional Controller..... | 19 |
| 7. Assignment..... | 21 |
| • Assignment 1..... | 21 |
| • Assignment 2..... | 23 |
| • Assignment 3..... | 25 |
| • Assignment 4..... | 27 |
| • Assignment 5..... | 29 |
| • Assignment 6..... | 31 |

ABSTRACT

The purpose of this project is "Development of the MCU (aTmega328) based servo stabilised platform to maintain predetermined angle".

To get the results at the end, the following topics-

- To study about the servomotor
- Inform about the Arduino UNO
 - LED Interfacing with Arduino
 - Controlling servo action through sensor
 - Interfacing of the Servo motor with Arduino Uno and generate PWM vs angle data
 - Interfacing of the inclinometer and servo motor with the Arduino UNO to make a servo stabilised platform.

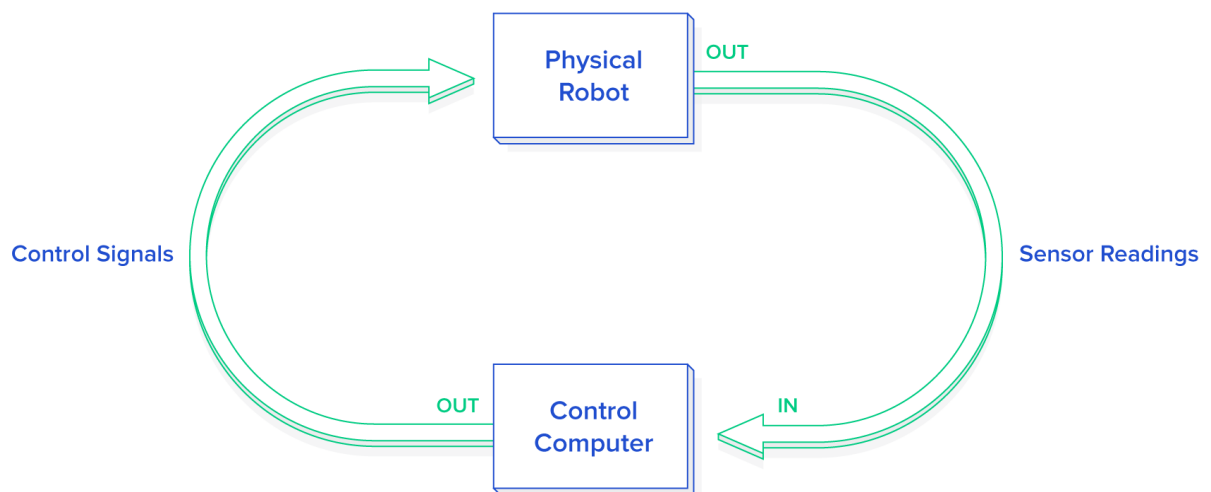
INTRODUCTION

A robot is a machine that gathers information about its environment and uses that information to follow instructions to do some sort of work. The robot sensory system gathers specific information needed by the control system. The robot control system directs the motion and sensory processing of a robot. The robot's mechanical system then produces motion that results in manipulation or locomotion.

Robotics is the science and engineering concerned with the design, manufacture and application of robots, and computer systems for their control, sensory feedback, and information processing. The many types of robotic systems include robotic manipulators, robotic hands, mobile robots, walking robots, aids for disabled persons, tele-robots, and microelectro-mechanical systems. Robotics Engineers study electronics, computer science, artificial intelligence, mechatronics, nanotechnology, and bioengineering.

Robots consist of some sort of mechanical construction. The mechanical aspect of a robot helps it complete tasks in the environment for which it's designed. Robots need electrical components that control and power the machinery.

Robots contain at least some level of computer programming. Without a set of code telling, it what to do, a robot would just be another piece of simple machinery. Inserting a program into a robot gives it the ability to know when and how to carry out a task.



There are many different ways a robot may be equipped to monitor its environment. These can include anything from proximity sensors, light sensors, bumpers, cameras, and so forth. In addition, robots may communicate with external sensors that give them information that they themselves cannot directly observe.

Our reference robot is equipped with sensors—arranged in every direction. There are more sensors facing the front of the robot than the back because it is usually more important for the robot to know what is in front of it than what is behind it.

Robot control is an essential technology that enables a robot to move precisely and adaptively. The problem of robot control is formulated to include determination of the joint torques to be generated by the joint actuators, so as to guarantee the execution of the robot task while satisfying given transient and steady state requirements. In view of constraints within the environment.

The Robot Control are categorized –

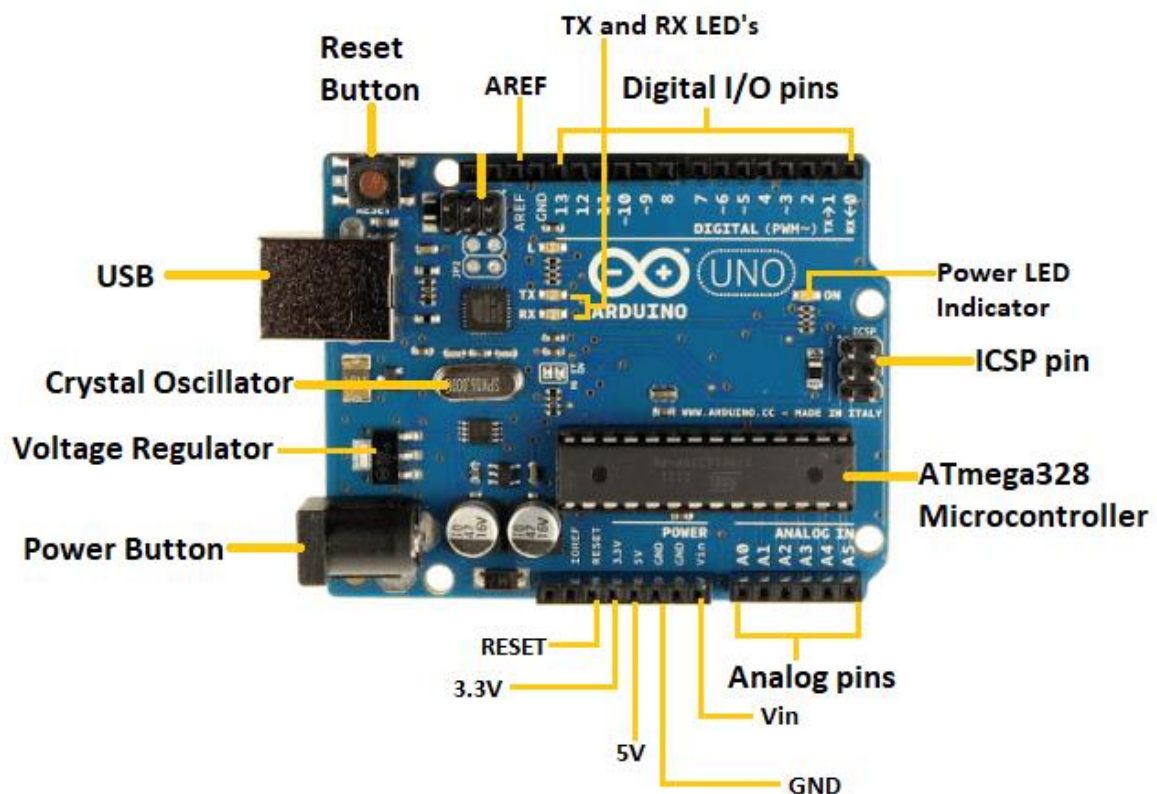
- 1.Motion Control
- 2.Dynamic Control
- 3.Adaptive Control
- 4.Force Control
- 5.Servo Control

Robot control software can only guess the state of the real world based on measurements returned by its sensors. It can only attempt to change the state of the real world through the generation of control signals.

The software that generates the control signals (the “controller”) is required to run at a very high speed and make complex computations. This affects the choice of which robot programming languages are use: Usually, C++ is used for these kinds of scenarios, but in simpler robotics applications, Python is a very good compromise between execution speed and ease of development and testing.

ARDUINO UNO

The Uno is a huge option for initial Arduino. This Arduino board depends on an ATmega328P based microcontroller. As compared with other type of boards, it is very simple to use. It consists of 14-digital I/O pins, where 6-pins can be used as PWM (pulse width modulation outputs), 6 analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), etc. It includes everything required to hold up the microcontroller, simply attach it to a PC with the help of USB cable and give the supply with an AC to DC adapter or battery to get started.



Pin Description

| Pin Category | Pin Name | Details |
|---------------------|--|---|
| Power | Vin, 3.3V, 5V, GND | <p>Vin: Input voltage to Arduino when using an external power source.</p> <p>5V: Regulated power supply used to power microcontroller and other components on the Board.</p> <p>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.</p> <p>GND: ground pins.</p> |
| Reset | Reset | Resets the microcontroller |
| Analog Pins | A0 –A5 | Used to provide analog input in the range of 0-5V |
| Input/output Pins | Digital Pins 0 – 13 | Can be used as input or output pins. |
| Serial | 0(Rx), 1(Tx) | Used to receive and transmit TTL serial data. |
| External Interrupts | 2,3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 10, 11 | Provides 8-bit PWM output. |
| SPI | 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK) | Used for SPI communication. |
| Inbuilt LED | 13 | To turn on the inbuilt LED. |
| TWI | A4 (SDA), A5 (SCA) | Used for TWI communication. |
| AREF | AREF | To provide reference voltage for input voltage. |

Technical Specification

- Microcontroller: ATmega328P – 8-bit AVR family microcontroller.
- Operating voltage: 5V
- Recommended input voltage: 7-12V
- Input voltage limit: 6-20V
- Analog input pins: 6(A0 – A5)
- Digital I/O pins: 14
- DC current on I/O pins: 40mA
- DC current on 3.3V pin: 50mA
- Flash memory: 32 KB
- SRAM: 2KB
- EEPROM: 1KB
- Frequency (clock speed): 14MHz

Advantages of using the Arduino includes:

- Easy to use and perfect for beginners.
- Cross-Platform
- Wide Variety

Applications of Arduino Boards:

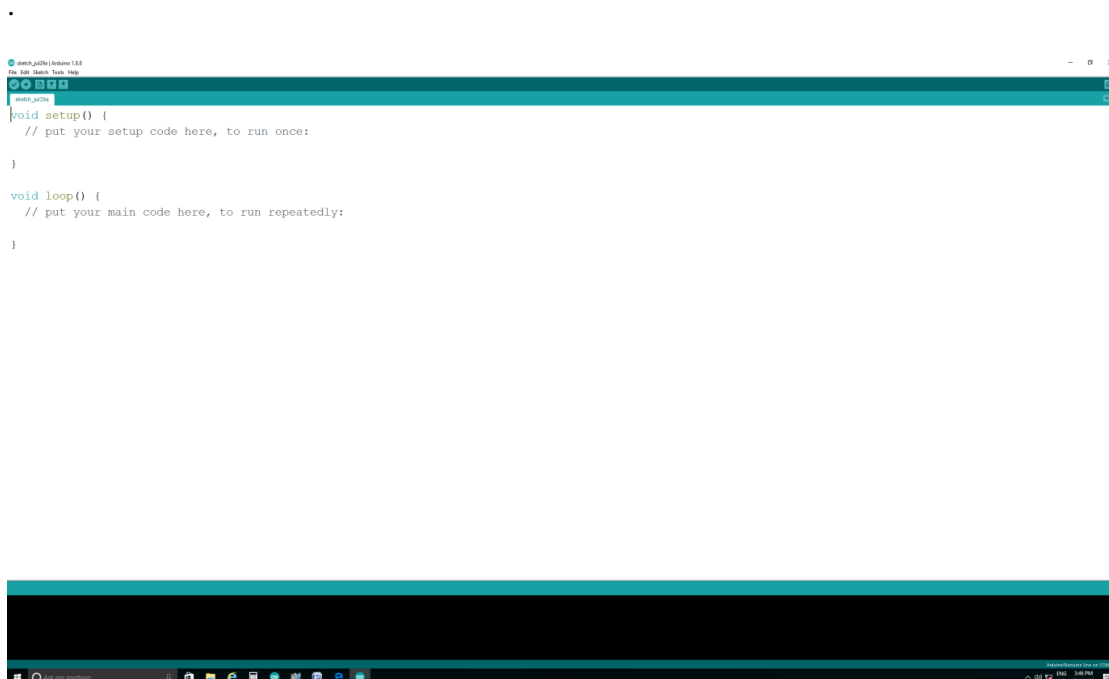
Arduino boards are playing very important roles in many disciplines today-

- Smart Homes
- Defence
- Industries
- Traffic Signal Control
- Medical
- Laboratories
- Aerospace
- Automatic Vehicle Control

ARDUINO IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuine hardware to upload programs and communicate with them.

The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU tool chain, also included with the IDE distribution. The Arduino IDE employs the program *argued* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, argued is used as the uploading tool to flash the user code onto official Arduino boards.



Sketches

- **Verify/Compile** Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- **Upload** Compiles and loads the binary file onto the configured board through the configured Port.
- **Upload Using Programmer** This will overwrite the boot loader on the board; you will need to use Tools > Burn Boot loader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for

your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Boot loader command must be executed.

- Export Compiled Binary Saves a .hex file that may be kept as archive or sent to the board using other tools.
- Show Sketch Folder Opens the current sketch folder.
- Include Library Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- Adds a supplemental file to the sketch (it will be copied from its current location). The file is saved to the data subfolder of the sketch, which is intended for assets such as documentation. The contents of the data folder are not compiled, so they do not become part of the sketch program.

Tools

- Auto Format - To formats the programme code i.e., indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- Archive Sketch- Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- Fix Encoding & Reload Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- Serial Monitor -Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- Board -Select the board that you're using. See below for descriptions of the various boards.

- Port -This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- Programmer- For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a boot loader to a new microcontroller, you will use this.
- Burn Boot loader -The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino board but is useful if you purchase a new AT mega microcontroller (which normally come without a boot loader). Ensure that you've selected the correct board from the Boards menu before burning the boot loader on the target board. This command also set the right fuses.

SENSOR

Sensor is a device that gives an output by detecting the changes in physical quantities or events. In general, sensors are termed as the devices that generate an electrical signal or optical output signal corresponding to the variations in the level of inputs.

All types of sensors can be basically classified into analog sensors and digital sensors.

Classification of Sensors:

i) Based on the quantity being measured-

- Temperature: Resistance Temperature Detector (RTD), Thermistor, Thermocouple
- Pressure: Bourdon tube, manometer, diaphragms, pressure gauge
- Force/ torque: Strain gauge, load cell
- Speed/ position: Tachometer, encoder, LVDT
- Light: Photo-diode, Light dependent resistor

ii) Active and Passive sensors-

Based on power requirement, sensors can be classified as active and passive. Active sensors are those which do not require external power source for their functioning. They generate power within themselves to operate and hence called as self-generating type. The energy for functioning is derived from the quantity being measured. Passive sensors require external power source for their functioning. Most of the resistive, inductive and capacitive sensors are passive (just as resistors, inductors and capacitors are called passive devices).

iii) Analog and Digital sensor-

An analog sensor converts the physical quantity being measured to analog form (Continuous in time). A digital sensor produces output in the form of pulse.

iv) Inverse sensors:

There are some sensors which are capable of sensing a physical quantity to convert it to other form and also sense the output signal form to get back the quantity in original form. This property makes them suitable to use in microphone and speakers

INCLINOMETER SENSOR

The DAS series inclinometers are range of high-performance low-cost dual axis tilt sensors for measurement of angle in both the pitch and roll axes. They utilise a very high-performance MEMS sensor which exhibits low long term & temperature drift compared with many competitive devices. Each sensor is packaged in a small, robust, sealed Aluminium housing and is supplied with a 2m screened PUR cable terminated with a 4 pin M12 connector. The high-resolution output voltage varies from 0.5-4.5V over the range of the sensor. There are three measurement range options available: $\pm 10^\circ$, $\pm 30^\circ$ & $\pm 90^\circ$, and a further two supply voltage level options: 5V (DC), or 7V to 32V (DC). These sensors are CE certified, and are manufactured and individually tested in our UK factory to guarantee performance to the stated specifications.

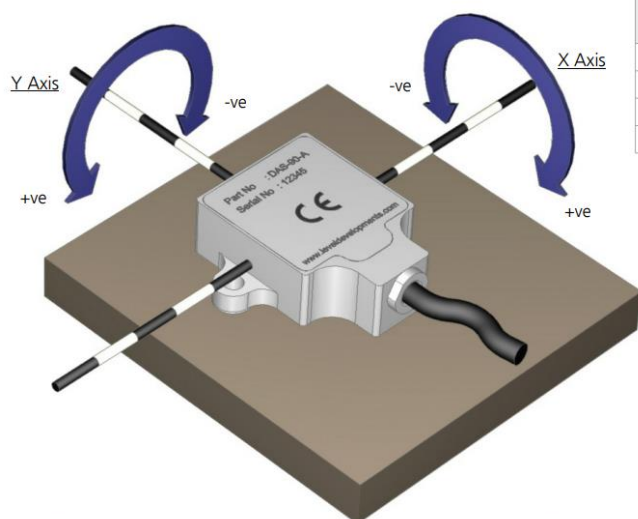


Features

- Dual axis, measuring ranges: $\pm 10^\circ$, $\pm 30^\circ$ or $\pm 90^\circ$
- Input supply options: 5Vdc or 7-32Vdc
- Sealed to IP67 · Solid state MEMS sensor
- High resolution 0.5-4.5V voltage output
- Low-cost relative to performance
- Small size, 46 x 43.5 x 13.5mm
- 2-meter cable with braided screen & M12 connector

Axis Direction and Mounting Orientation and Wiring Details

Horizontal surface is the zero plane



| Pin # | Internal Wire Colour | Function |
|-------|----------------------|---------------|
| 1 | Brown | +ve Supply |
| 2 | White | Y Axis Output |
| 3 | Blue | Gnd (0V) |
| 4 | Black | X Axis Output |

SERVO MOTOR

A servomotor is a linear actuator or rotary actuator that allows for precise control of linear or angular position, acceleration, and velocity. It consists of a motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.



Principle of Servo Motor

Servo motor works on the **PWM** (Pulse Width Modulation) principle, which means its angle of rotation, is controlled by the duration of pulse applied to its control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears

Working of Servo Motor

Servo motors control position and speed very precisely. Now a potentiometer can sense the mechanical position of the shaft. Hence it couples with the motor shaft through gears. The current position of the shaft is converted into electrical signal by potentiometer, and is compared with the command input signal. In modern servo motors, electronic encoders or sensors sense the position of the shaft.

Usually, a servomotor turns 90 degrees in either direction hence maximum movement can be 180 degrees.

A servo motor is control by sending a pulse width modulated (PWM) signal through the control wire. A pulse is sent every 20 milliseconds. Width of the pulses determines the position of the shaft.

Applications of Servo Motor

- Robotics: The robot arm.
- Conveyor belts: servo motors move, stop, and start conveyor belts.
- Camera auto focus.
- Solar tracking system.

PWM (Pulse Width Modulation):

While ADC is used to read Analog signals by a digital device like microcontroller, a PWM can be considered as an exact opposite of it i.e., PWM is used to produce Analog signals from a digital device like microcontroller. PWM stands for Pulse Width Modulation which can be understood as a type of signal which can be produced from a digital IC such as microcontroller or 555 timers. The signal thus produced will have a train of pulses. That is, at any given instance of time the wave will either be high or will be low. For example, let us consider a 5V PWM signal, in this case the PWM signal will either be 5V (high) or at ground level 0V (low). The duration at which the signals stay high is called the “on time” and the duration at which the signal stays low is called as the “off time”.

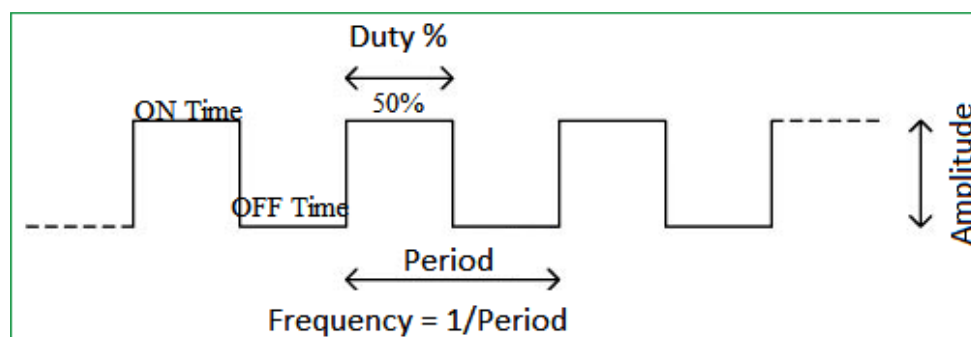
Two important parameters associated with PWM are-

- 1) Duty cycle of the PWM
- 2) Frequency of the PWM

Duty Cycle of PWM

As a PWM signal stays on for a particular time and then stays off for the rest of the period. The thing which makes a PWM signal special and more useful is that we can set for how long it should stay on by controlling the duty cycle of the PWM signal. The percentage of time in which the PWM signal remains HIGH (on time) is called as duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle. The formulae to calculate the duty cycle is shown below-

$$\text{Duty Cycle} = \frac{\text{Turn ON time}}{(\text{Turn ON time} + \text{Turn OFF time})}$$



A PWM signal with 50% Duty Cycle

By controlling the Duty cycle from 0% to 100% we can control the “on time” of PWM signal and thus the width of signal. Since we can modulate the width of the pulse, it got its iconic name “Pulse width Modulation”.

Frequency of a PWM

The frequency of a PWM signal determines how fast a PWM completes one period. One Period is the complete ON and OFF time of a PWM signal as shown in the above figure. The formulae to calculate the Frequency is given below-

Frequency = $1/\text{Time Period}$

Time Period = On time + Off time

Normally the PWM signals generated by microcontroller will be around 500 Hz, such high frequencies will be used in high-speed switching devices like inverters or converters. But not all applications require high frequency. For example, to control a servo motor we need to produce PWM signals with 50Hz frequency, so the frequency of a PWM signal is also controllable by program for all microcontrollers.

Control System

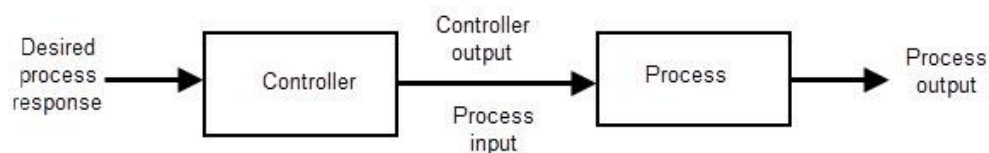
A control system is defined as a system of devices that manages, commands, directs, or regulates the behaviour of other devices or systems to achieve a desired result. A control system achieves this through control loops, which are a process designed to maintain a process variable at a desired set point.

There are two main types of control systems. They are as follow

- 1 Open- loop control system
- 2 Closed- loop control system

Open Loop Control System

A control system in which the control action is totally independent of the output of the system then it is called an open-loop control system.

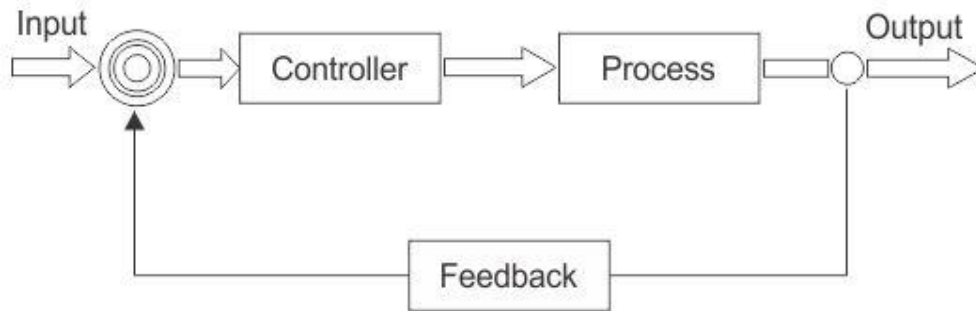


Advantages of open-loop control systems:

1. Simple in construction and design
2. Easy to maintain
3. Generally stable

Closed Loop Control System

Control systems in which the output has an effect on the input quantity in such a manner that the input quantity will adjust itself based on the output generated is called a closed-loop control system.



$$1+G(s)H(s)$$

Where,

$G(s)$ =Controller

$H(s)$ = feedback

Advantages of closed-loop control systems include:

- 1.The bandwidth range is large
- 2.Facilitates automation
- 3.It is less affected by noise

| Open Loop Control System | Closed Loop Control System |
|-----------------------------------|---|
| The feedback element is absent. | The feedback element is always present. |
| An error detector is not present. | An error detector is always present. |
| It is a stable one. | It may become unstable. |
| Having a small bandwidth. | Having a large bandwidth |
| Easy to construct. | Complicated construction. |
| It is inaccurate. | It is accurate. |
| Less maintenance. | More maintenance. |

SIMPLE CONTROLLER

Proportional controller

PROPORTIONAL CONTROLLER-

P controller is mostly used in first order processes with single energy storage to stabilize the unstable process. The main usage of the P controller is to decrease the steady state error of the system. As the proportional gain factor K increases, the steady state error of the system decreases. However, despite the reduction, P control can never manage to eliminate the steady state error of the system. As we increase the proportional gain, it provides smaller amplitude and phase margin, faster dynamics satisfying wider frequency band and larger sensitivity to the noise. We can use this controller only when our system is tolerable to a constant steady state error. In addition, it can be easily concluded that applying P controller decreases the rise time and after a certain value of reduction on the steady state error, increasing K only leads to overshoot of the system response. P control also causes oscillation if sufficiently aggressive in the presence of lags and/or dead time. The more lags (higher order), the more problem it leads. Plus, it directly amplifies process noise.

$$P_{out} = K_p e(t) + p_0$$

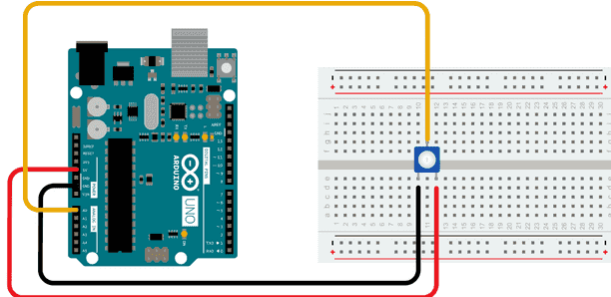
Where

- p_0 : Controller output with zero error.
- P_{out} : Output of the proportional controller
- K_p : Proportional gain
- $e(t)$: Instantaneous process error at time t . $e(t) = SP - PV$
- $e(t) = SP - PV$
- SP : Set point
- PV : Process variable

ASSIGNMENT – 1

AIM- Write a code to read analog, analog input using Arduino UNO MCU.

Circuit Diagram-

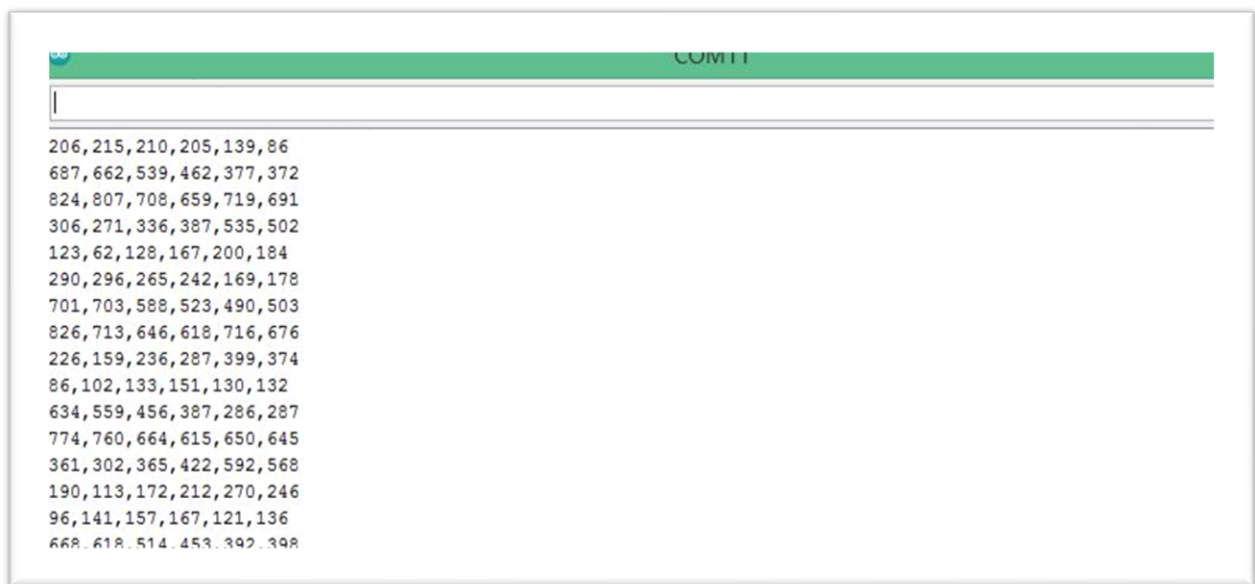


PROGRAM-

```
void setup (){  
  Serial.begin(9600);  
}  
  
void loop(){  
  int val1 = analogRead(A0);  
  Serial.print(val1);  
  Serial.print(",");  
  int val2 = analogRead(A1);  
  Serial.print(val2);  
  Serial.print(",");  
  int val3 = analogRead(A2);  
  Serial.print(val3);  
  Serial.print(",");  
  int val4 = analogRead(A3);  
  Serial.print(val4);  
  Serial.print(",");  
  int val5 = analogRead(A4);  
  Serial.print(val5);
```

```
Serial.print(",");  
int val6 = analogRead(A5);  
Serial.println(val6);  
delay(1000);  
}  
}
```

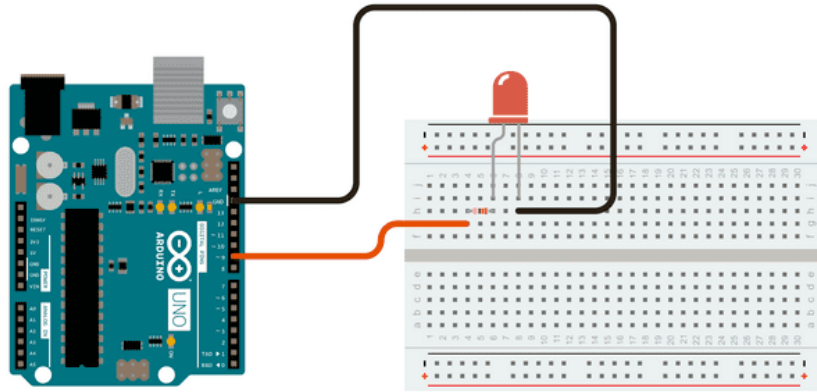
Output



ASSIGNMENT-2

AIM- Interfacing of the LED to the Arduino UNO and write a code for to make it blink.

Circuit diagram-



PROGRAM-

```
void setup() {  
  Serial.begin(9600);  
  pinMode(6,OUTPUT);  
}
```

```
void loop() {  
  digitalWrite(6,HIGH);  
  delay(500);  
  digitalWrite(6,LOW);  
  delay(500);  
}
```

Output:

- 1) If we'll give the LOW as output then the LED turns OFF.
- 2) If we'll give the HIGH as output then the LED turns ON.

ASSIGNMENT-3

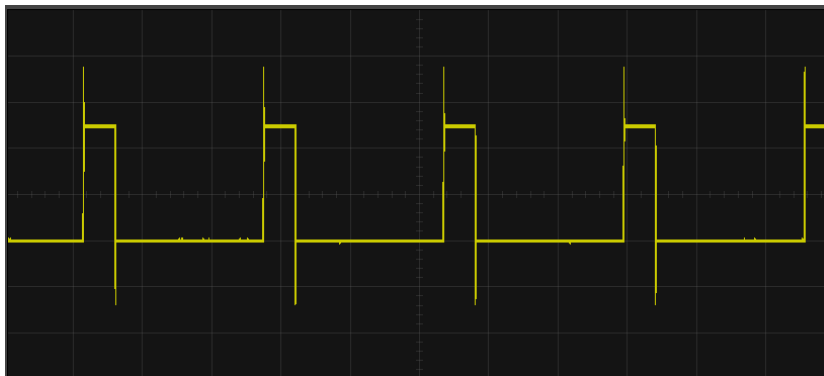
AIM- Write a code to generate PWM signal of desired Duty cycle and time period.

THEORY- By varying the width of PWM we generate d PWM signal of range 0.9 to 2.1 microsec as per the graphs shown below. While ADC is used to read Analog signals by a digital device like microcontroller, a PWM can be considered as an exact opposite of it i.e., PWM is used to produce Analog signals from a digital device like microcontroller. As a PWM signal stays on for a particular time and then stays off for the rest of the period. The thing which makes a PWM signal special and more useful is that we can set for how long it should stay on by controlling the duty cycle of the PWM signal. The percentage of time in which the PWM signal remains HIGH (on time) is called as duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle.

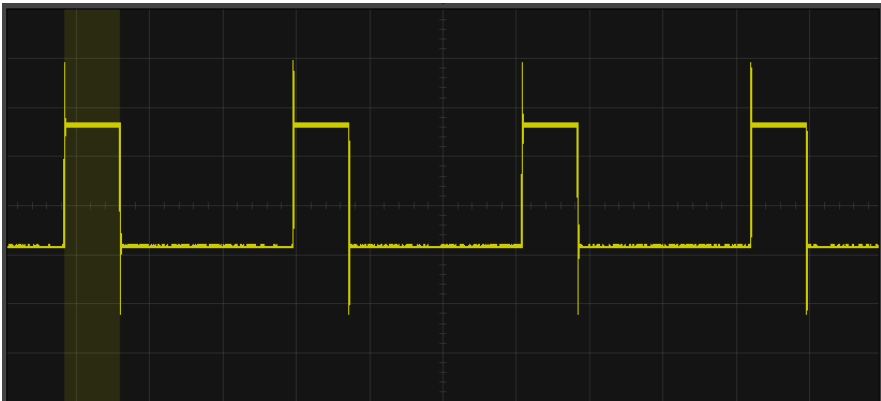
PROGRAM-

```
void setup() {  
  int timeperiod =20000;  
  
  Serial.begin(9600);  
  pinMode(6,OUTPUT);  
}  
  
void loop() {  
  for( int servo =900 ; servo <=2100; servo ++){  
    digitalWrite(6,HIGH);  
    delayMicroseconds(servo);  
    digitalWrite(6,LOW);  
    delayMicroseconds((timeperiod- servo));  
  }  
}
```

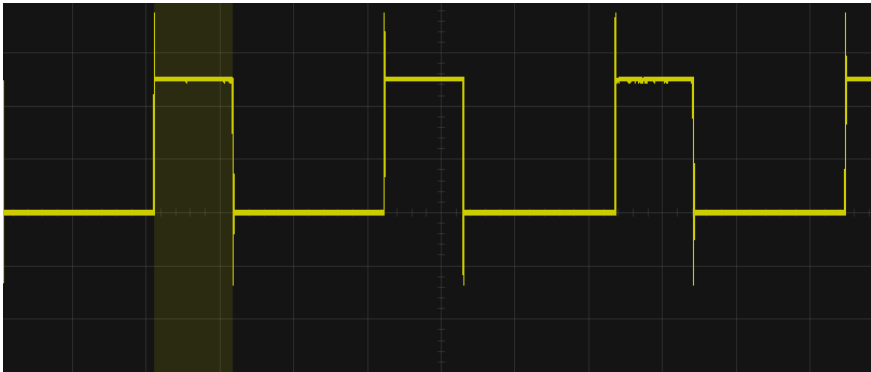
Observed Output:



Pulse width 0.9 microsec



Pulse width 1.5 microsec

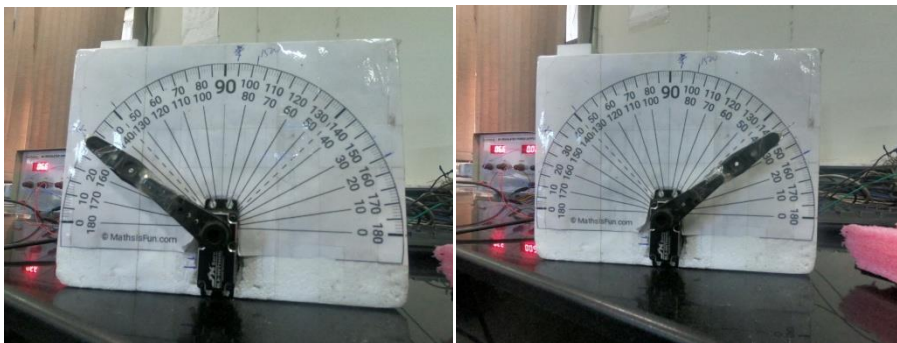
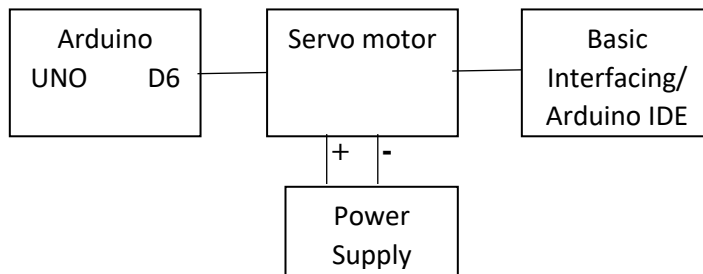


Pulse width 2.1 microsec

ASSIGNMENT-4

AIM- Interfacing of the Servo motor with Arduino Uno and generate PWM vs angle data.

CIRCUIT DIAGRAM-



THEORY- Servo motor moves to range between 900 to 2100.

PROGRAM-

```
void setup() {
```

```
  Serial.begin(9600);{  
  pinMode(6,OUTPUT);  
  }}
```

```
void loop() {
```

```
  for( int servo =900 ; servo <=2100; servo =servo+10){  
    digitalWrite(6,HIGH);  
    delayMicroseconds(servo);  
    digitalWrite(6,LOW);
```

```

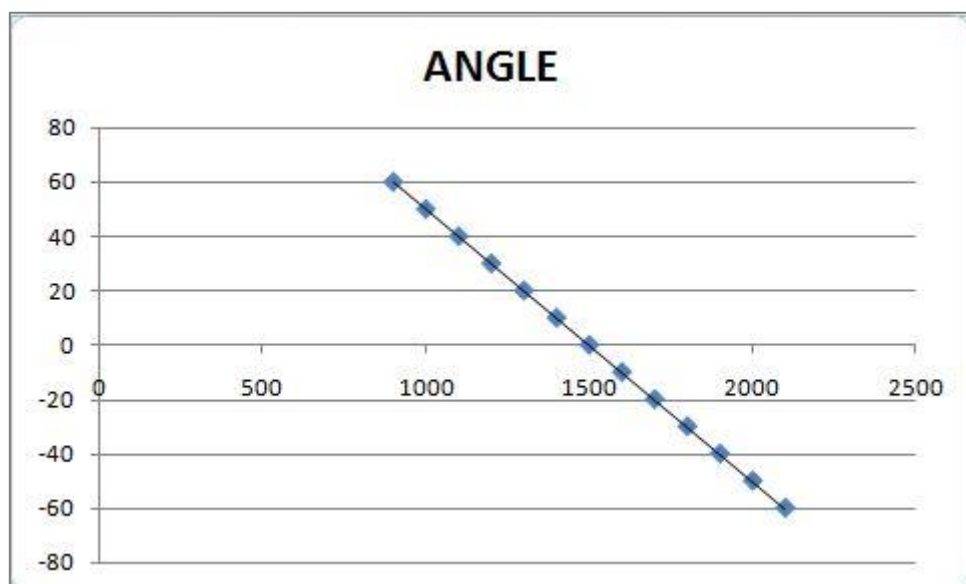
delayMicroseconds((20000- servo));
}
Delay(5000)
}

```

OUTPUT:

| RANGE | OUTPUT | ANGLE |
|-------|--------|-------|
| 900 | 150 | 60 |
| 1000 | 140 | 50 |
| 1100 | 130 | 40 |
| 1200 | 120 | 30 |
| 1300 | 110 | 20 |
| 1400 | 100 | 10 |
| 1500 | 90 | 0 |
| 1600 | 80 | -10 |
| 1700 | 70 | -20 |
| 1800 | 60 | -30 |
| 1900 | 50 | -40 |
| 2000 | 40 | -50 |
| 2100 | 30 | -60 |

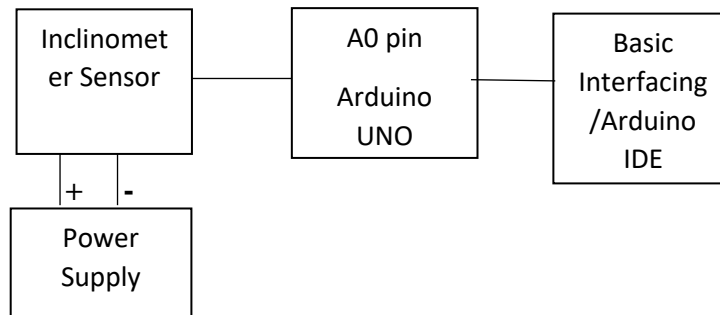
Excel graph



ASSIGNMENT-5

AIM –Interfacing of the integral of inclinometer with Arduino UNO.

CIRCUIT DIAGRAM



THEORY- If I'll rotate the sensor, I'll get the output (ADC count).

PROGRAM –

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int sensor = analogRead(A0);
  Serial.print(sensor);
  Serial.print(",");
  Serial.print(angle);
  Serial.println(",");
  delay(200);
}
```

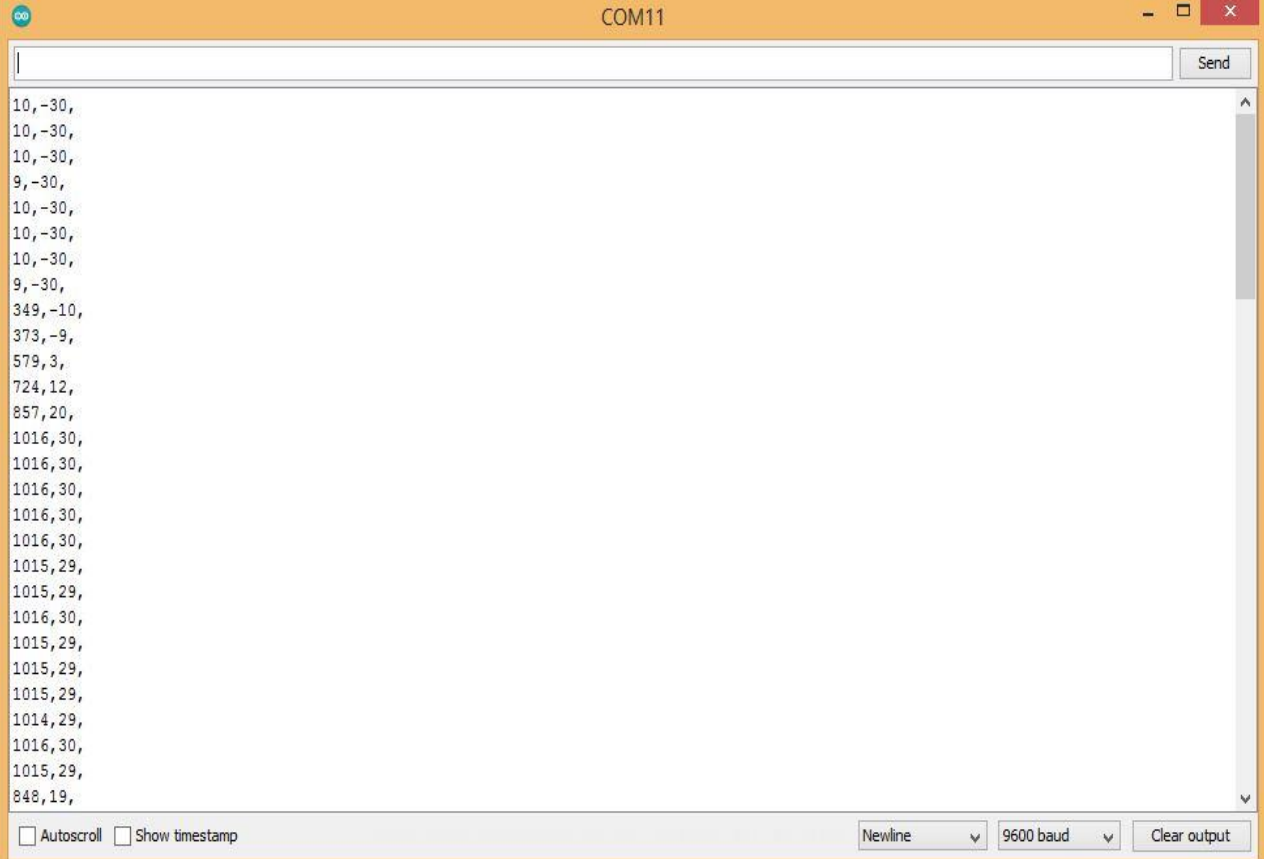
Output

| angle | ADC count |
|-------|-----------|
| -30 | 10 |
| +30 | 1016 |

Count to angle conversion factor $= (1016 - 10) / 60$ count/deg

In Arduino the same could be implemented as `map (sensor,10,1016, -30,30)`

OUTPUT-

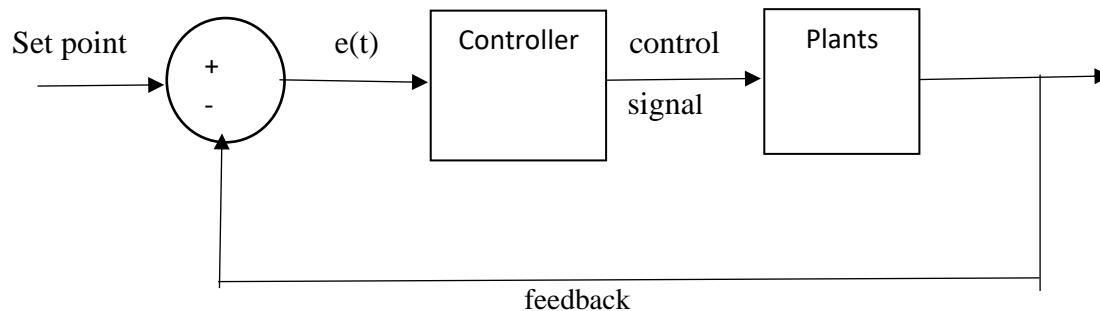


```
10,-30,  
10,-30,  
10,-30,  
9,-30,  
10,-30,  
10,-30,  
10,-30,  
9,-30,  
349,-10,  
373,-9,  
579,3,  
724,12,  
857,20,  
1016,30,  
1016,30,  
1016,30,  
1016,30,  
1016,30,  
1015,29,  
1015,29,  
1016,30,  
1015,29,  
1015,29,  
1015,29,  
1015,29,  
1014,29,  
1016,30,  
1015,29,  
848,19,
```

ASSIGNMENT-6

AIM – Interfacing of the inclinometer and servo motor with the Arduino UNO to make a servo stabilised platform.

CIRCUIT DIAGRAM -



BRIEF DETAIL

Read the output of data from arduino pwm pin and connect it to the signal of servo motor and give it the range from 900 to 2100 the servo motor move the range and connect it to the supply and then add inclinometer sensor to it sensor Balance the pin of the servo motor according to the angle.

PROGRAM-

```
int gain=10;
int set_angle=0;
int error1;
int cmd;
void setup() {
  Serial.begin(9600);
  pinMode(6,OUTPUT);
}
void loop() {
  // reads setup angle
  int sensor = analogRead(A0);
  int angle= map( sensor,10,1016,-30,30);
  //Serial.println(sensor);
  //delay(200);
  error1=angle-set_angle;
```

```

if(error1!=0)
{
cmd=gain*error1+1500;
digitalWrite(6,HIGH);
delayMicroseconds(cmd);
digitalWrite(6,LOW);
delayMicroseconds((20000-cmd));
Serial.print(set_angle);
Serial.print(",");
Serial.print(angle);
Serial.print(",");
Serial.print(cmd);
Serial.println(",");
}
delay(200);
}

```

OUTPUT-

