COURSE MATERIAL

IV Year B. Tech I- Semester

MECHANICAL ENGINEERING

AY: 2022-23

Automation & Control Engineering Laboratory

R18A0390

PREPARED BY
H DEEPTHI
ASST PROFESSOR

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF MECHANICAL ENGINEERING

(Autonomous Institution-UGC, Govt. of India)
Secunderabad-500100,Telangana State, India.
www.mrcet.ac.in

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

www.mrcet.ac.in

Department of Mechanical Engineering

VISION

To become an innovative knowledge center in mechanical engineering through state-of-the-art teaching-learning and research practices, promoting creative thinking professionals.

MISSION

The Department of Mechanical Engineering is dedicated for transforming the students into highly competent Mechanical engineers to meet the needs of the industry, in a changing and challenging technical environment, by strongly focusing in the fundamentals of engineering sciences for achieving excellent results in their professional pursuits.

Quality Policy

- ✓ To pursuit global Standards of excellence in all our endeavors namely teaching, research and continuing education and to remain accountable in our core and support functions, through processes of self-evaluation and continuous improvement.
- ✓ To create a midst of excellence for imparting state of art education, industry-oriented training research in the field of technical education.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

IV Year B. Tech ME - I Sem

L	P	C
0	3	1.5

(R18A0390) AUTOMATION AND CONTROL ENGINEERING LAB

Course Objectives:

1. To learn the different types of system and process controls in Automation.
2. To identify and study different control system components.
3. To understand simulation on controllers.
4. To demonstrate working of different actuating systems and sensors.
5. To learn and demonstrate IoT.

Any 10 of the following experiments has to be performed

1. Simulation on P Controller.
2. Simulation on P+I Controller.
3. Simulation on P+D Controller.
4. Simulation of Hydraulic Actuation System.
5. Simulation of Pneumatic Actuation System.
6. Simulation on Stepper Motor.
7. Simulation on Logic gates, decoders and flip-flops.
8. Determination of Ambient Temperature using Arduino.
9. Experiment on Photo-resistor and LED using Arduino.
10. Determination of Temperature using IoT.
11. Determination of Humidity using IoT.
12. Experiment on speed control of stepper motor.
13. Obstacle detection using sensors.
14. Experiment on assessment of load characteristics of solar panels in series & parallel connection.

Course Outcomes: Students will be able

1. To apply the knowledge in real time applications.
2. To understand the components of control system.
3. To gain knowledge on simulation softwares.
4. To work on different actuating systems & sensors.
5. To understand technologies like IoT, machine languages.

Determination of Ambient Temperature using Arduino

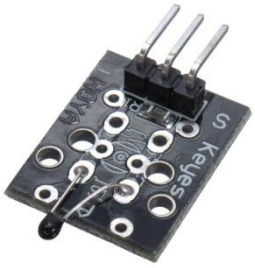
Aim: To sense and determine the Ambient Temperature using Temperature Sensor and Arduino.

Tools and Components required:

1. Arduino UNO
2. Analog Temperature Sensor
3. Connecting Wires
4. Arduino Software
5. Bread Board

Theory:

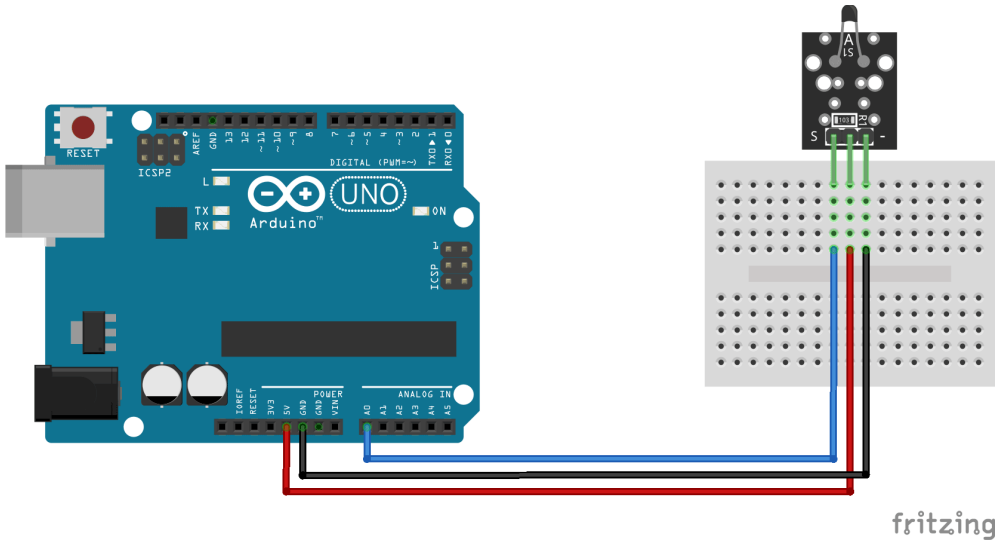
Analog Temperature Sensor module KY-013 for Arduino, measures ambient temperature based on resistance of the thermistor. The KY-013 Analog Temperature Sensor module consists of a NTC thermistor and a 10 kΩ resistor. The resistance of the thermistor varies with surrounding temperature; we'll use the Steinhart–Hart equation to derive precise temperature of the thermistor.



Specifications

1. Operating Voltage : 5V
2. Temperature measurement range : -55°C to 125°C [-67°F to 257°F]
3. Measurement accuracy : ±0.5°C

Circuit:



Program:

```
int ThermistorPin = A0;
int Vo;
float R1 = 10000;
float logR2, R2, T;
float c1 = 0.001129148, c2 = 0.000234125, c3 = 0.0000000876741;
void setup() {
  Serial.begin(9600);
}
void loop() {
  Vo = analogRead(ThermistorPin);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2)); // temperature in Kelvin
  T = T - 273.15;
  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.println(" C");
  delay(500);
}
```

Result: The ambient temperature is determined using the KY-013 temperature sensor and arduino.

Experiment on Photo-resistor and LED using Arduino

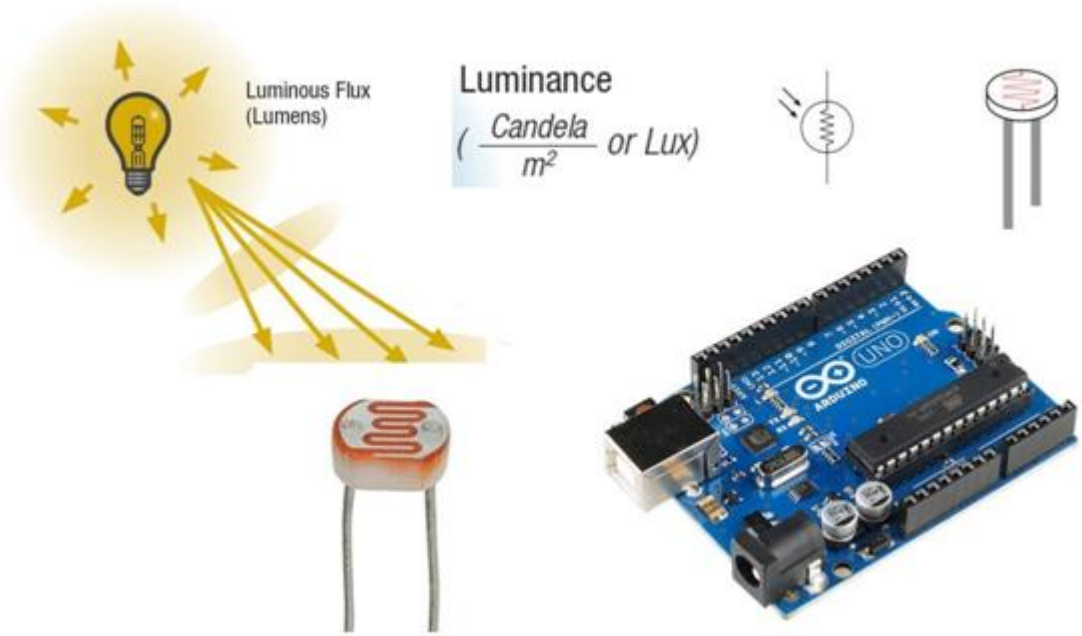
Aim: To detect the light intensity using Arduino UNO.

Tools and Components required:

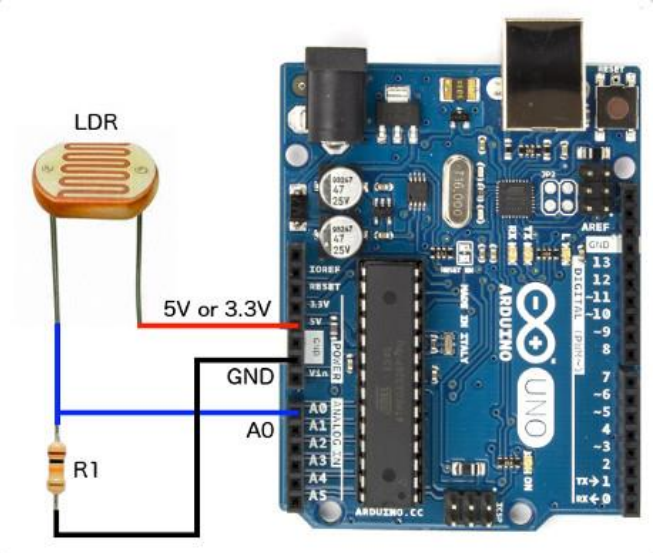
1. Arduino UNO board
2. LED's
3. LDR Sensor (Photo Resistor)
4. Connecting Wires
5. Arduino Software.
6. Resistors
7. Bread Board

Theory:

Photo resistor allows interacting with the external environment, through intensity of light. It is based on light resistance that senses the light and will allow the microcontroller in the arduino to react and change the intensity of LED diode. The photo resistor creates a different resistance based on the intensity or the light. Changing the resistance through intensity changes the voltage too. The microcontroller reads different values and will light up the Led with more or less intensity. A low resistance value will occur when the sensor is well lighted and a high resistance value will occur when it is in darkness.



Circuit:



Program

```
int ldr=A0;//Set A0(Analog Input) for LDR.
int value=0;
void setup() {
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop() {
value=analogRead(ldr);
Serial.println("LDR value is :");
Serial.println(value);
if(value<300)
 {
   digitalWrite(13,HIGH);
 }
 else
 {
   digitalWrite(13,LOW);
 }
}
```

Result: The light intensity is measured successfully with Photo resistor and Arduino UNO.

Determination of Humidity using IoT

Aim : To determine the humidity using Arduino

Tools and Components required:

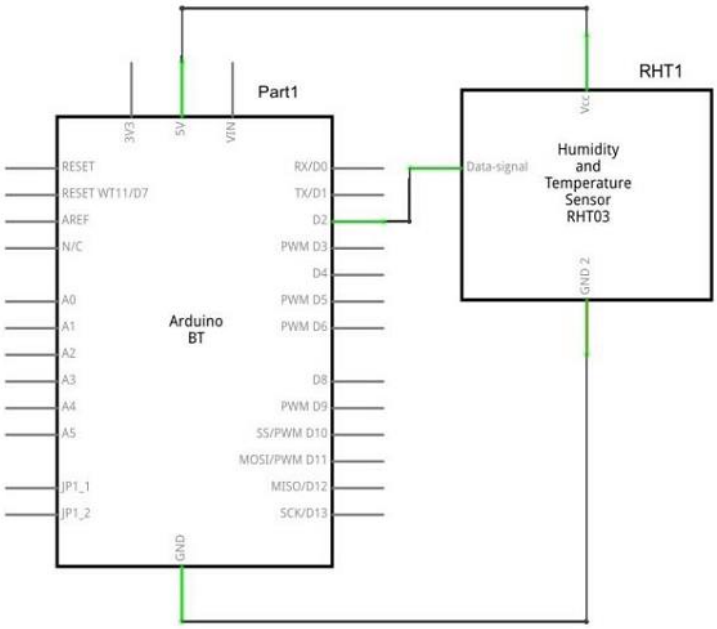
1. Arduino UNO Board
2. DHT-22
3. Bread Board
4. Arduino Software
5. Connecting Cables

Theory:

The DHT-22 (also named as AM2302) is a digital-output, relative humidity, and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends a digital signal on the data pin.

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

Circuit Diagram:



Program:

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print ("Humidity: ");
  Serial.print (h);
  Serial.print (" %\t");
  Serial.print ("Temperature: ");
  Serial.print (t);
  Serial.print (" *C ");
  Serial.print (f);
```

```
  Serial.print (" *F\t");
  Serial.print ("Heat index: ");
  Serial.print (hic);
  Serial.print (" *C ");
  Serial.print (hif);
  Serial.println (" *F");
}
```

Result:

The obstacle are detected using an ultrasonic sensor and arduino UNO.

Experiment on Speed Control of Stepper Motor

Aim: To control the speed of Stepper Motor using Arduino

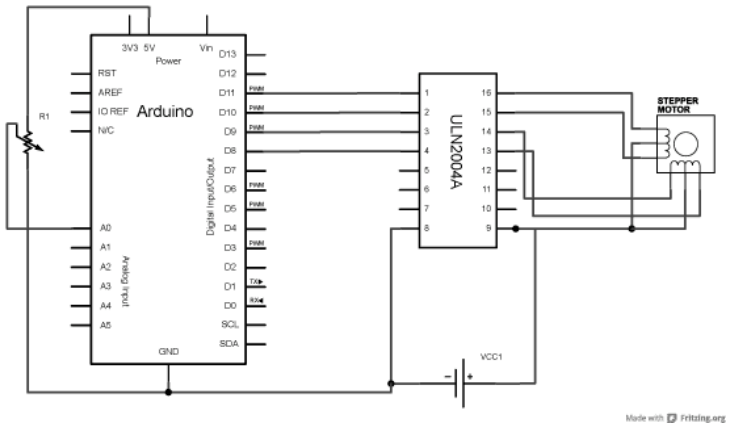
Tools and Components required:

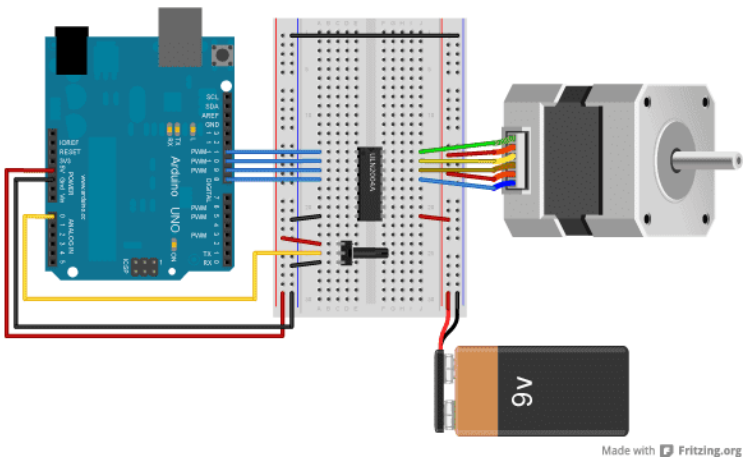
1. Arduino Uno
2. Stepper Motor
3. 10k ohm potentiometer
4. Bread board'
5. Arduino software
6. U2004 Darlington Array (in case of Unipolar stepper motor)
7. SN754410ne H-Bridge (in case of Bipolar stepper motor)

Theory:

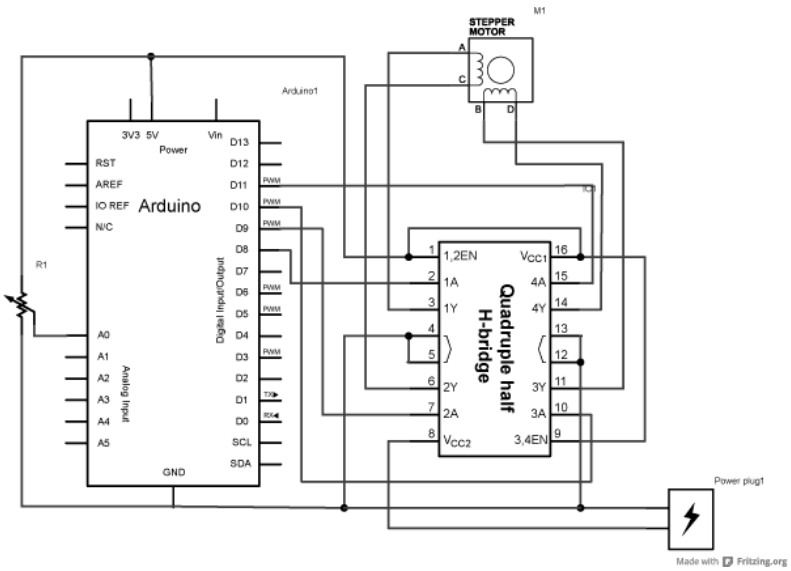
Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps". There are two types of steppers, Unipolars and Bipolars, and it is very important to know which type you are working with. For each of the motors, there is a different circuit.

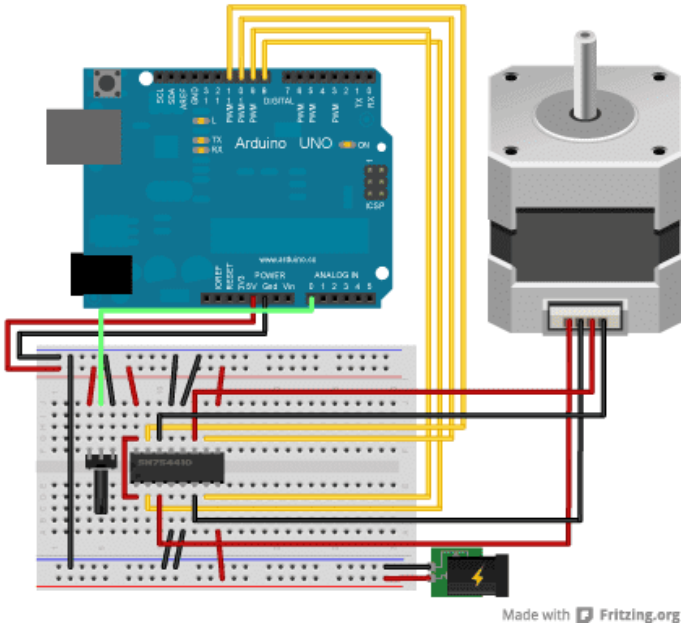
Circuits:





Unipolar Motor Schematic





Bipolar Motor schematic

Program:

```
#include <Stepper.h>

const int stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

int stepCount = 0;  // number of steps the motor has taken

void setup() {

}

void loop() {

  int sensorReading = analogRead(A0);

  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
```

```
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    myStepper.step(stepsPerRevolution / 100);
  }
}
```

Result: The speed of the Uni and Bipolar Stepper motor is controlled using Arduino

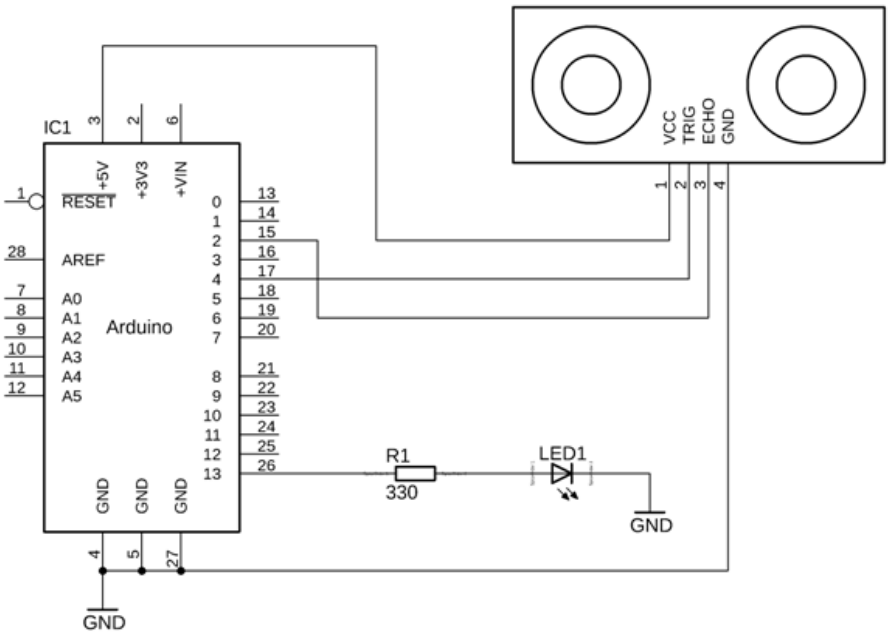
Obstacle detection using Sensors

Aim : To detect the obstacles and alert using Arduino

Tools Required:

1. Arduino UNO Board
2. Ultrasonic sensor
3. Test Board
4. Arduino Software
5. Connecting Cables
6. +5V buzzer/ LED

Circuit Diagram:



Theory:

The ultrasonic sensor transmits sound waves and receives sound reflected from an object. When ultrasonic waves are incident on an object, diffused reflection of the energy takes place over a wide solid angle which might be as high as 180 degrees. Thus some fraction of the incident energy is reflected back to the transducer in the form of echoes. If the object is very close to the sensor, the sound waves returns quickly, but if the object is far away from the sensor, the sound waves takes longer to return. But if objects are too far away from the sensor, the signal takes so long to come back (or is very weak when it comes back) that the receiver cannot detect it. The sensor uses the time it takes for the sound to come back from the object in front to determine the distance of an object.

The distance to the object (L) can then be calculated through the speed of ultrasonic waves (v) in the medium by the relation, where,'t' is the time taken by the wave to reach back to the sensor and 'Θ' is the angle between the horizontal and the path taken. If the object is in motion, instruments based on Doppler shift are used. The ultrasonic sensor can measure distances in centimetres and inches. It can measure from 0 to 2.5 meters, with a precision of 3 cm.

Program:

```
int const trigPin = 10;
int const echoPin = 9;
int const buzzPin = 2;

void setup()
{
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
        pinMode(buzzPin, OUTPUT);
}
void loop()
{
        int duration, distance;
        digitalWrite(trigPin, HIGH);
        delay(1);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance = (duration/2) / 29.1;
  if (distance <= 50 && distance >= 0) {
        digitalWrite(buzzPin, HIGH);
  } else {
        digitalWrite(buzzPin, LOW);
  }
  delay(60);
}
```

Result:

The obstacle are detected using an ultrasonic sensor and arduino UNO.

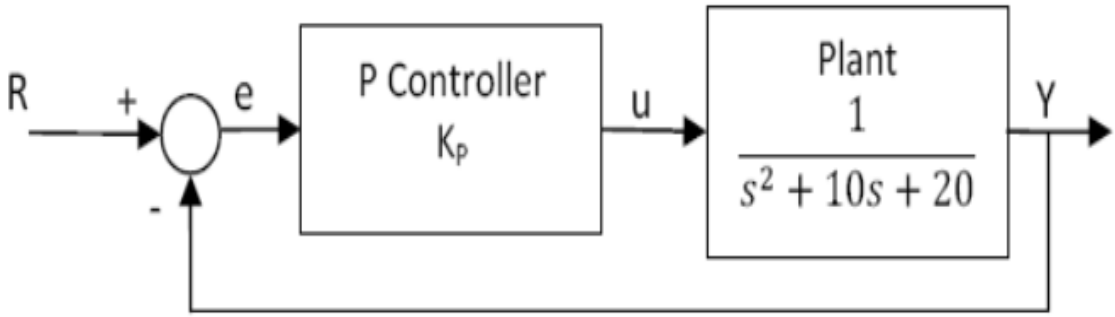
Simulation on P Controller

Aim: To simulate the P Controller for unit step input.

System Configuration:

1. Ram: 4 GB
2. Processor: Intel CORE i3
3. Operating system: Window 7
4. Software: MATLAB

Circuit Diagram:



Plant: A system to be controlled.

Controller: Provides excitation for the plant; Designed to control the overall system behavior.

Proportional control, in engineering and process control, is a type of linear feedback control system in which a correction is applied to the controlled variable which is proportional to the difference between the desired value (setpoint, SP) and the measured value (process variable, PV). In the proportional control algorithm, the controller output is proportional to the error signal, which is the difference between the setpoint and the process variable. In other words, the output of a proportional controller is the multiplication product of the error signal and the proportional gain.

This can be mathematically expressed as

$$P_{out} \propto e(t)$$

$$P_{out} = K_P e(t) + P_0$$

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.

Transfer Function

$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

Let proportional gain $K_p = 300$

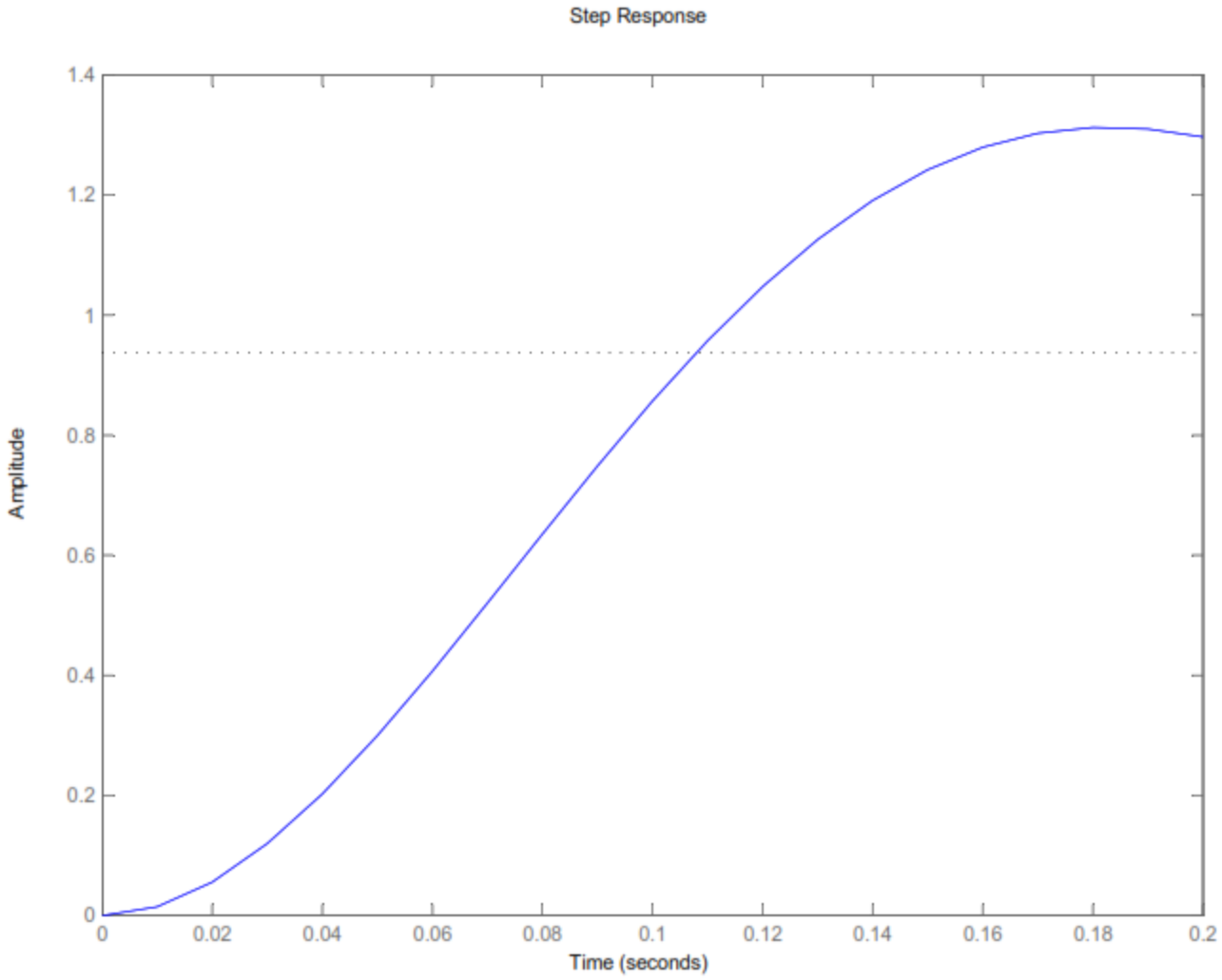
The goal of the problem is to show K_p contribute to obtain

- ✓ Fast rise time
- ✓ Minimum overshoot
- ✓ No steady state error

Program:

```
num = 1;

den = [ 1 10 20];

Plant = tf(num,den);

Step(plant);

Kp = 300;

Contr = kp;

Sys_cl=feedback(contr*plant,1)

t = 0:0.01:0.2;

step(sys_cl,t)
```

Observations:



Result: Simulation of P Controller done successfully by using MATLAB

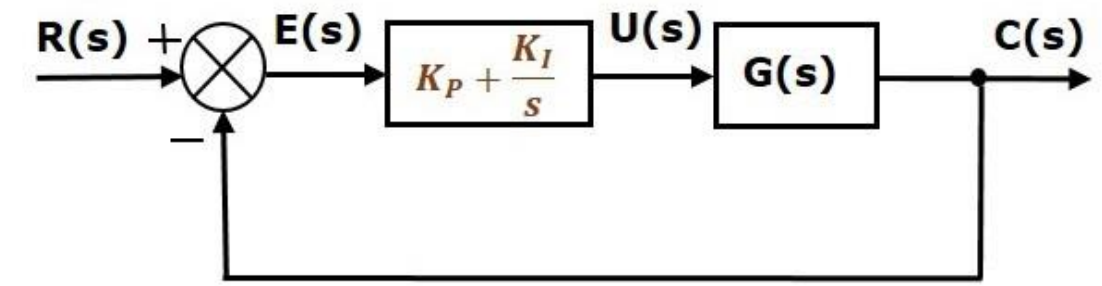
Simulation on P-I Controller

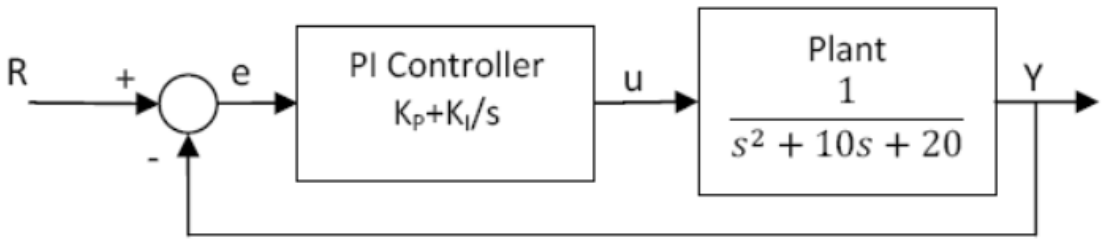
Aim: To simulate the PI Controller for unit step input.

System Configuration:

1. Ram: 4 GB
2. Processor: Intel CORE i3
3. Operating system: Window 7
4. Software: MATLAB

Circuit Diagram:





Plant: A system to be controlled.

Controller: Provides excitation for the plant; Designed to control the overall system behavior.

Proportional Integral Control: Output power equals to the sum of proportion and integration coefficients. The higher the proportion coefficient, the less the output power at the same control error. The higher the integration coefficient, the slower the accumulated integration coefficient. PI control provides zero control error and is insensitive to interference of the measurement channel.

The PI control disadvantage is slow reaction to disturbances. To adjust the PI controller, you should first set the integration time equal to zero, and the maximum proportion time. Then by decreasing the coefficient of proportionality, achieve periodic oscillations in the system. Close to the optimum value of the coefficient of proportionality is twice higher than that at which any hesitation, and close to the optimum value of the integration time constant - is 20% less than the oscillation period.

This can be mathematically expressed as

$$m = K_{\mathrm{p}}\left[e + \frac{1}{T_{\mathrm{i}}}\int_0^t edt\right] + b$$

$$m = K_{\mathrm{p}}e + K_{\mathrm{p}}\frac{1}{T_{\mathrm{i}}}\int_0^t edt + b$$

Transfer Function

$$\frac{X(s)}{F(s)} = \frac{K_p s + K_I}{s^3 + 10s^2 + (20 + K_p)s + K_I}$$

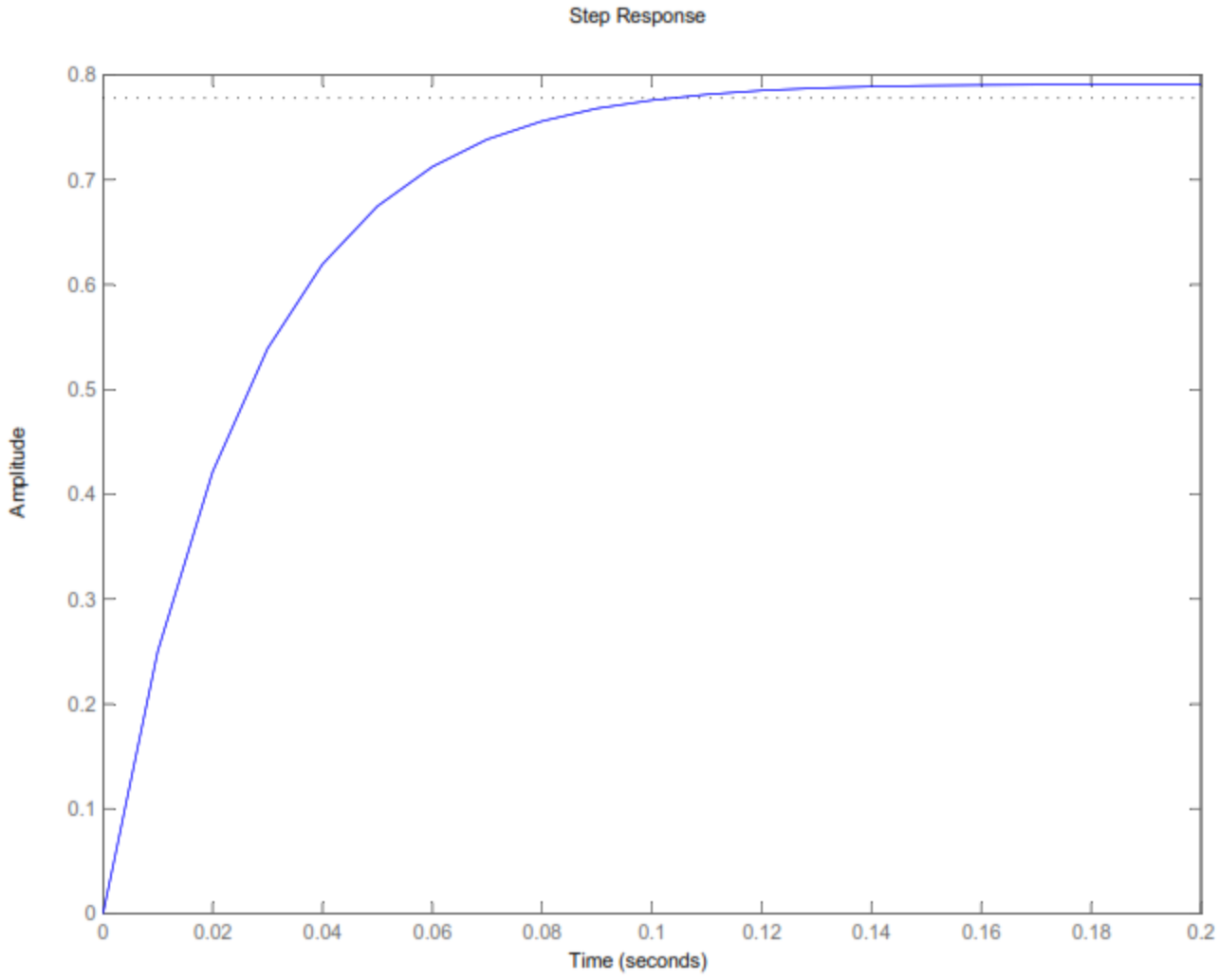
Let reduce the K_p to 30 and K_i equal to 70

Program:

```
num = 1;

den = [ 1 10 20];

Plant = tf(num,den);

Step(plant);

Kp = 30;

Ki  = 70;

Contr = tf([Kp Ki],[1 0]);

Sys_cl=feedback(contr*plant,1)
```

```
t = 0:0.01:0.2;

step(sys_cl,t)
```

Observations:



Result: Simulation of P-I Controller done successfully by using MATLAB

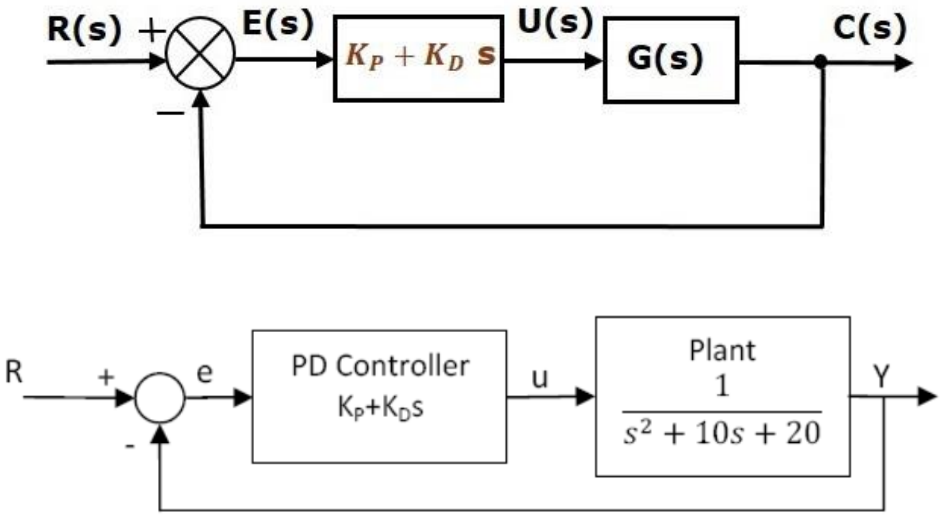
Simulation on P-D Controller

Aim: To simulate the PD Controller for unit step input.

System Configuration:

1. Ram: 4 GB
2. Processor: Intel CORE i3
3. Operating system: Window 7
4. Software: MATLAB

Circuit Diagram:



Plant: A system to be controlled.

Controller: Provides excitation for the plant; Designed to control the overall system behavior.

Proportional Derivative Control: As the name indicates, it is a combination of proportional and derivative control actions. It is widely used in industrial applications. Its control action is mathematically expressed as

$$m = K_p\left[e + T_d \frac{de}{dt}\right] + b$$

$$m = K_p e + K_p T_d \frac{de}{dt} + b$$

This control action sometimes fails to eliminate the offset of proportional controllers. It is also called rate control action. In P-D controller, the output varies with respect to rate of change of actuating error signal.

Transfer Function

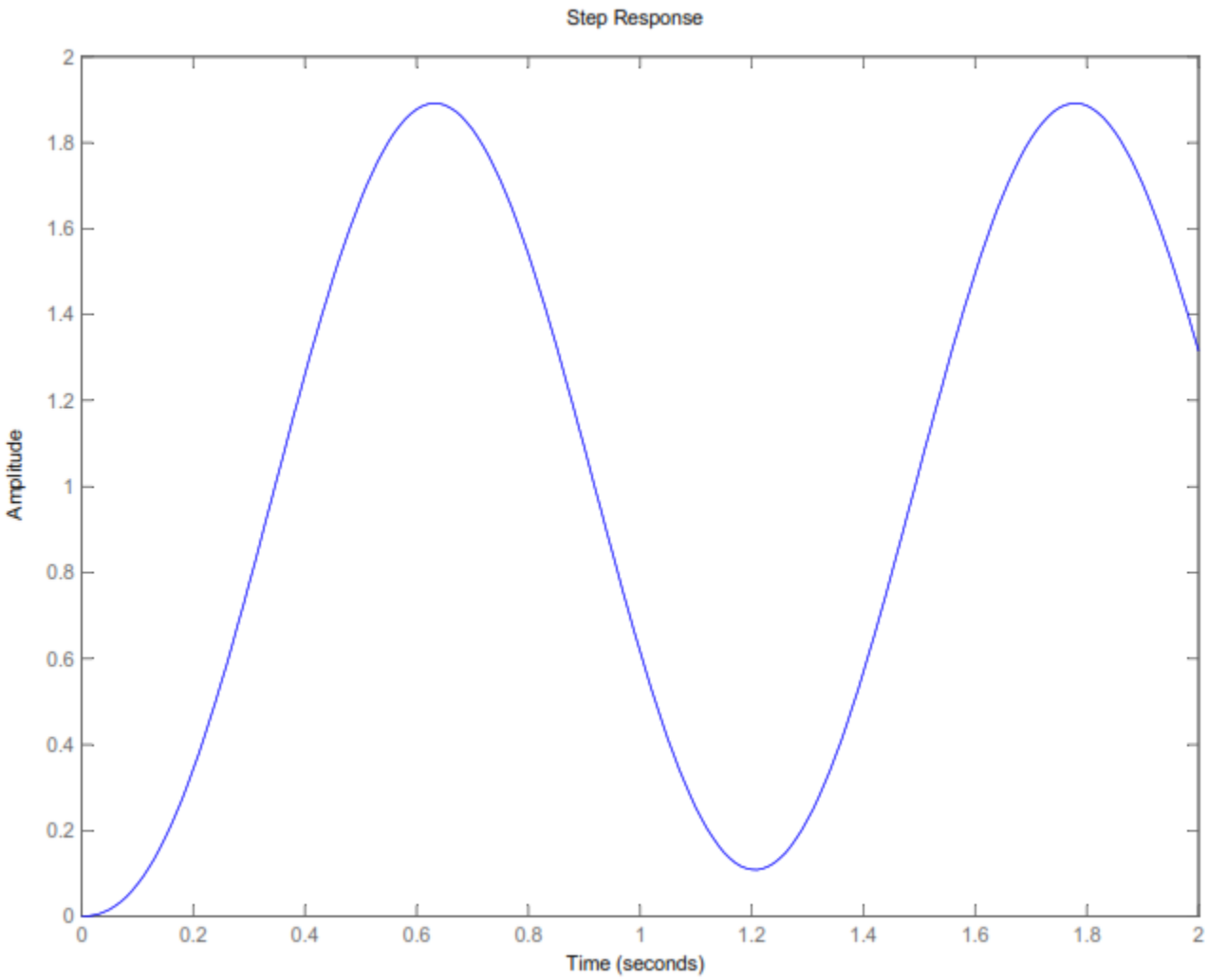
$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Let reduce the K_p to 300 and K_d equal to 10

Program:

```
num = 1;

den = [ 1 10 20];

Plant = tf(num,den);

Step(plant);

Kp = 300;

Kd  = 10;

Contr = tf([Kd Kp],1);

Sys_cl=feedback(contr*plant,1)

t = 0:0.01:0.2;

step(sys_cl,t)
```

Observations:



Result: Simulation of P-D Controller done successfully by using MATLAB