Egyptian Aviation Academy
الأكاديمية المصرية لعلوم الطيران

Control Engineering

Ministry of Higher Education
وزارة التعليم العالي

Institute of Aviation Engineering and Technology
معهد هندسة وتكنولوجيا الطيران

Speed control of dc motor

Course Name	ECE 310 Control 1-B
Title	Speed control of dc motor
Name	Ahmed Thabit Ali Eldeen Ahmed Mohamed Atef Mostafa M.sayed
Grade	3rd Year – Second Term
Department	Electronics&Communication Departmant
Academic Year	2022/ 2023

Under Supervision Of

Prof. Mohammed Essa

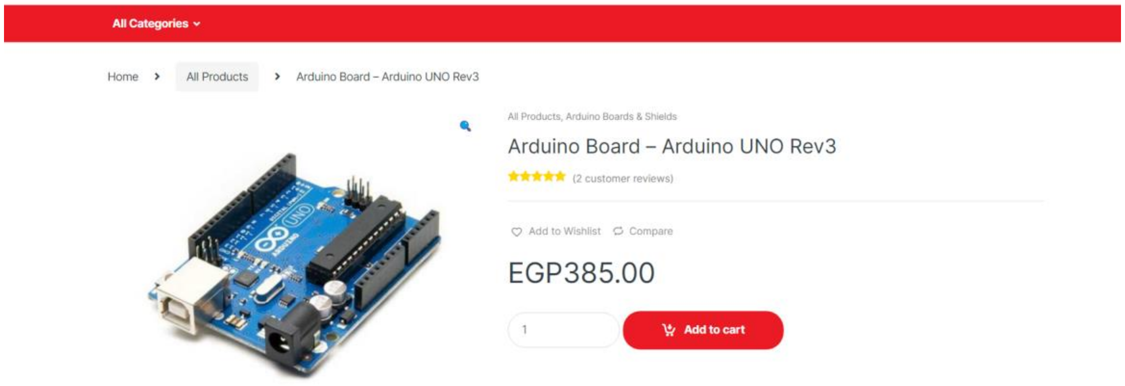
Eng. Shaimaa Helmy

2022- 2023

1. Introduction

The objective of this project is to achieve speed control of a DC motor through the integration of Arduino programming platform and MATLAB's Simulink coder. The real-time system is implemented using the hardware Arduino support package with MATLAB, while the System Identification Toolbox is employed to model the DC motor. The control design process utilizes an Arduino board and Simulink PID controller in a closed loop system, with the evaluation of the controllers based on various criteria such as settling time, system overshoot, rise time, steady-state error, and cross-correlation. Moreover, this project suggests the incorporation of Artificial Intelligence (AI) and Internet of Things (IoT) technologies to enhance the control and monitoring systems of DC motors

2-Select the hardware component



All Categories

Home › All Products › Arduino Board – Arduino UNO Rev3

All Products, Arduino Boards & Shields

Arduino Board – Arduino UNO Rev3

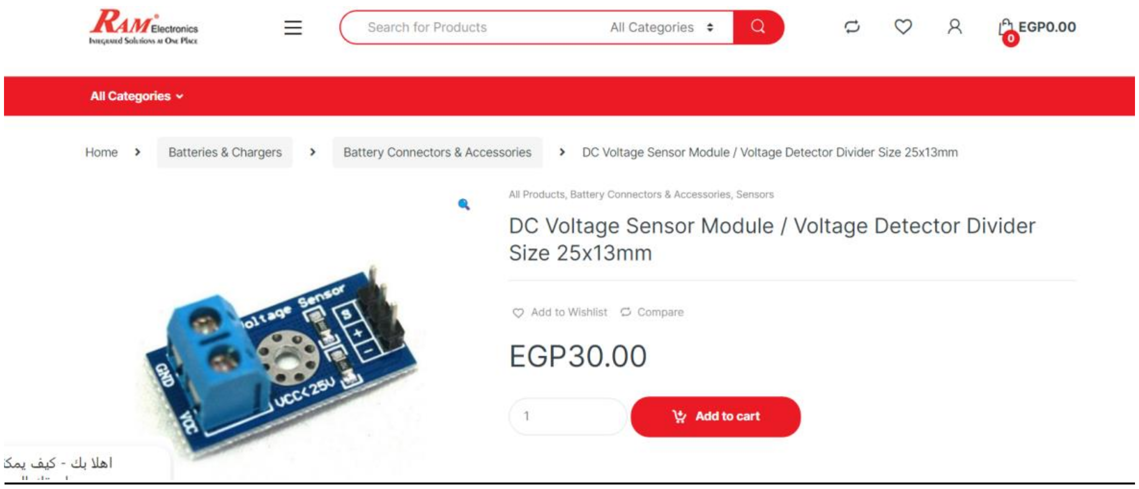
★★★★★ (2 customer reviews)

Add to Wishlist Compare

EGP385.00

1 Add to cart

Figure1 arduino



RAM Electronics Integrated Solutions at One Place

Search for Products All Categories

EGP0.00

All Categories

Home › Batteries & Chargers › Battery Connectors & Accessories › DC Voltage Sensor Module / Voltage Detector Divider Size 25x13mm

All Products, Battery Connectors & Accessories, Sensors

DC Voltage Sensor Module / Voltage Detector Divider Size 25x13mm

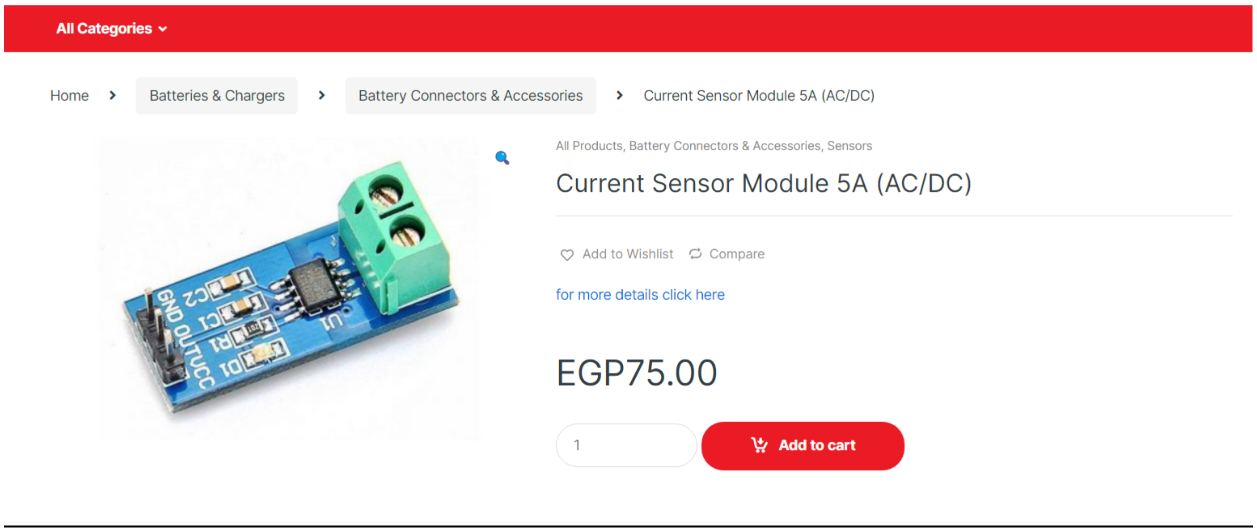
Add to Wishlist Compare

EGP30.00

1 Add to cart

اهلا بك - كيف يمك

Figure 2 voltage sensor



All Categories

Home › Batteries & Chargers › Battery Connectors & Accessories › Current Sensor Module 5A (AC/DC)

All Products, Battery Connectors & Accessories, Sensors

Current Sensor Module 5A (AC/DC)

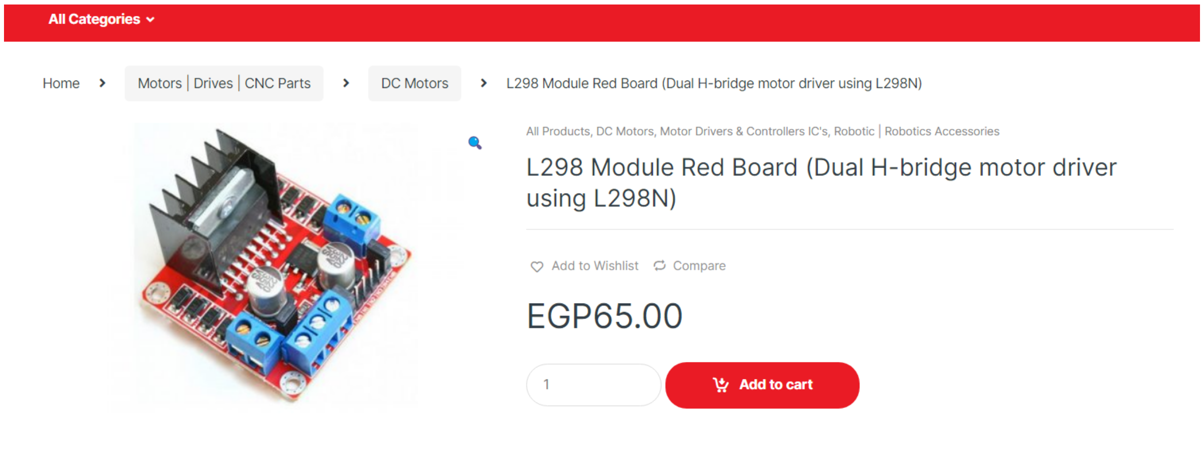
Add to Wishlist Compare

for more details click here

EGP75.00

1 Add to cart

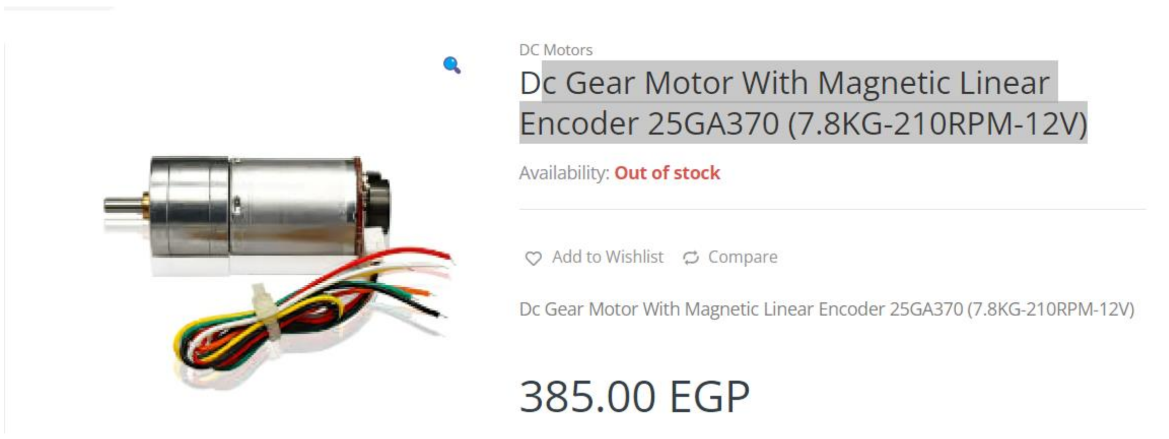
Figure 3 current sensor


All Categories
Home
Motors | Drives | CNC Parts
DC Motors
L298 Module Red Board (Dual H-bridge motor driver using L298N)
All Products, DC Motors, Motor Drivers & Controllers IC's, Robotic | Robotics Accessories

L298 Module Red Board (Dual H-bridge motor driver using L298N)

EGP65.00

Figure 4 H –bridge



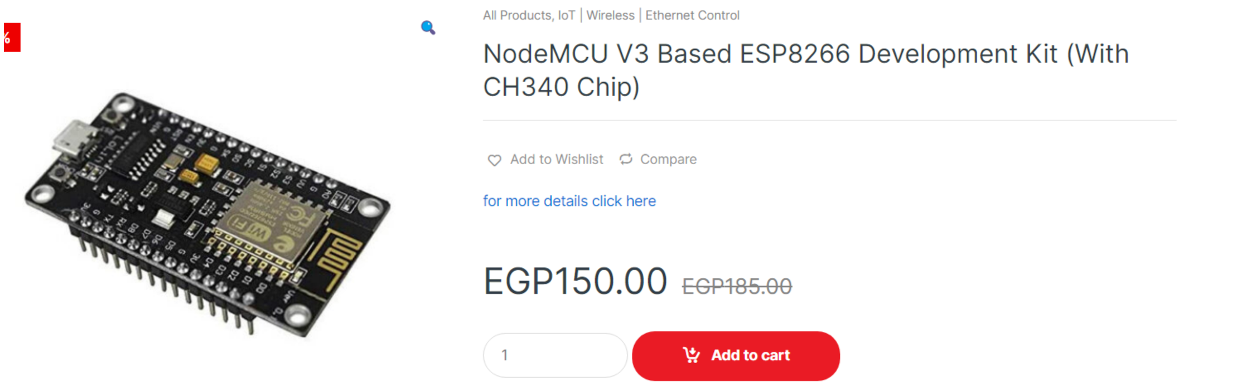
Dc Gear Motor With Magnetic Linear Encoder 25GA370 (7.8KG-210RPM-12V)

Availability: Out of stock

Dc Gear Motor With Magnetic Linear Encoder 25GA370 (7.8KG-210RPM-12V)

385.00 EGP

Figure 5 Dc Gear Motor With Magnetic Linear Encode



NodeMCU V3 Based ESP8266 Development Kit (With CH340 Chip)

for more details click here

EGP150.00 ~~EGP185.00~~

Figure 6 ESP

Total Cost :

- Arduino => 480 £
- Current sensor => 65 £
- Voltage sensor => 25 £
- H bridge => 90 £
- Metal Gear DC 12V = 385 £
- ESP 8266 =>150 E

COMPONENTS

Contains of :

- Arduino
- Current sensor
- Voltage sensor
- H bridge
- Metal gear Dc 12v

Arduino :

The Arduino Uno is an opensource microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010. The board is equipped with sets of digital and analog input/output pins that may be interfaced to various expansion boards and other circuits.

Figure 7 Arduino uno

Current sensor :

Sensing and controlling current flow is a fundamental requirement in a wide variety of applications including, over-current protection circuits, battery chargers, switching mode power supplies, digital watt meters, programmable current sources, etc. This ACS721 current module is based on ACS712 sensor, which can accurately detect AC or DC current. The maximum AC or DC that can be detected can reach 5A, and the present current signal can be read via analog I / O port of Arduino.

Figure 8 current sensor

Voltage sensor : A simple but very useful module which uses a potential divider to reduce any input voltage by a factor of 5. This allows you to use the analog input of a microcontroller to monitor voltages much higher than it capable of sensing. For example, with a 0-5V analog input range you are able to measure a voltage up to 25V. The module also includes convenient screw terminals for easy and secure connection of wires. This module is based on the principle of resistive voltage divider design, can make the red terminal connector input voltage to 5 times smaller Arduino analog input voltages up to 5 v, the voltage detection module input voltage not greater than 5Vx5=25V (if using 3.3V systems, input voltage not greater than 3.3Vx5=16.5V). Arduino AVR chips have 10-bit AD, so this module simulates a resolution of 0.00489V (5V/1023), so the minimum voltage of input voltage detection module is 0.00489Vx5=0.02445 v

Figure 9 voltage sensor

H-bridge :

L298N Dual H Bridge DC Stepper Motor Drive Controller Board Module For Arduino.The L298 Stepper Controller makes it easy to drive either two DC motors or a bipolar stepper motor. This is a very high quality board and is very compact for designs where space really matters.

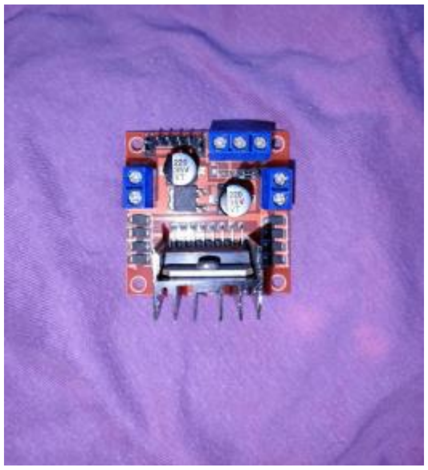


Figure 10 H-bridge

Metal gear DC 12 v :

This is a DC Mini Metal Reduction Motor, perfect for DIY engine, electric lock, robot model All-metal material, extremely durable and fine craftsmanship, motor speed can only be reduced, can not be raised Voltage: DC 12V Retarder Reduction Ratio: 1:110 Specification Material: Metal No-load current: 50mA Locked-rotor current: 1A .

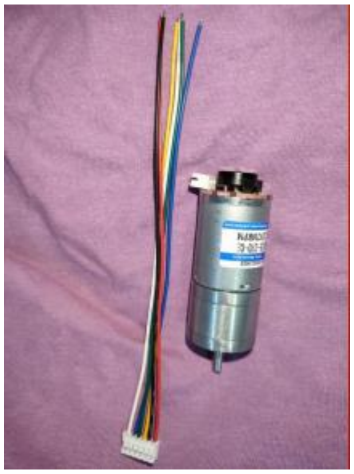


Figure 11 DC motor 12V

ESP

Extra-Sensory Perception, refers to the ability to perceive information through means other than the five physical senses of sight, hearing, touch, taste, and smell. This can include abilities such as telepathy (the ability to read minds), clairvoyance (the ability to see things beyond the range of normal vision), precognition (the ability to perceive future events), and psychokinesis (the ability to move objects with the mind).

ESP has been a topic of much debate and skepticism within the scientific community, as there is little empirical evidence to support its existence. However, many people still believe in the existence of ESP and claim to have experienced it themselves or witnessed it in others.

Research into ESP has been ongoing for decades, with some studies indicating that certain individuals may possess limited psychic abilities. However, the validity and reliability of these studies have been called into question by critics, and the scientific consensus remains that there is no conclusive evidence to support the existence of ESP.

Figure 12 ESP

Building a MATLAB model

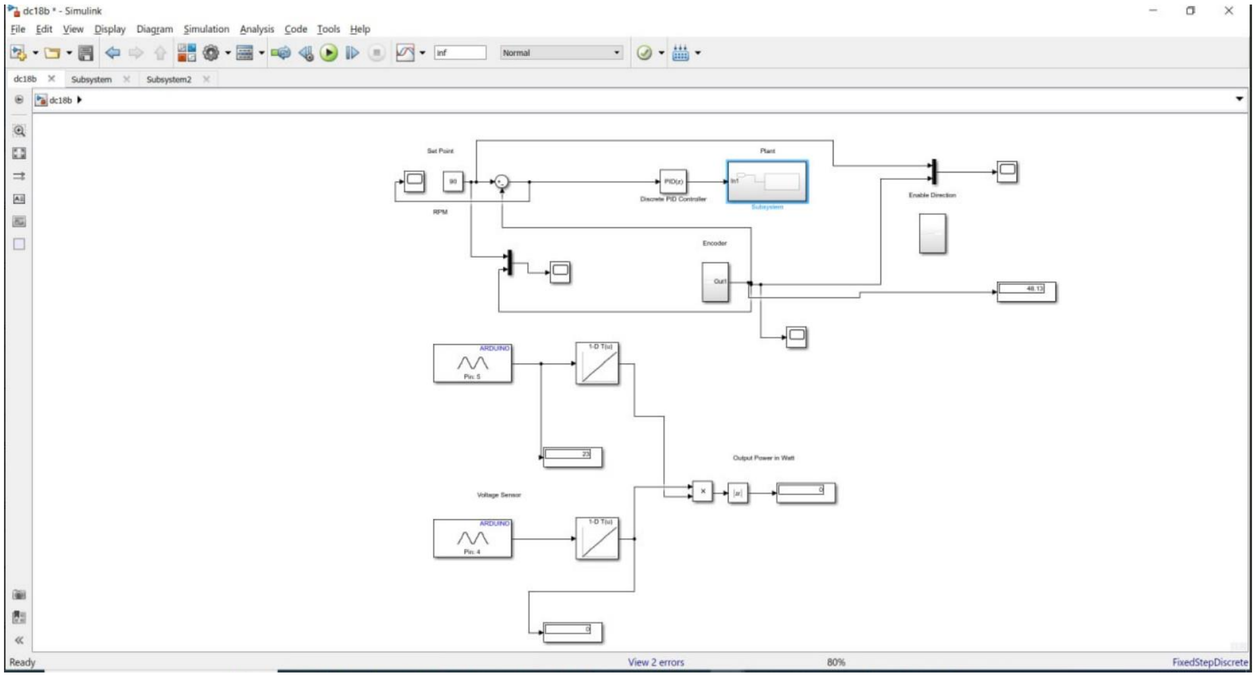


Figure 13 Model (1)

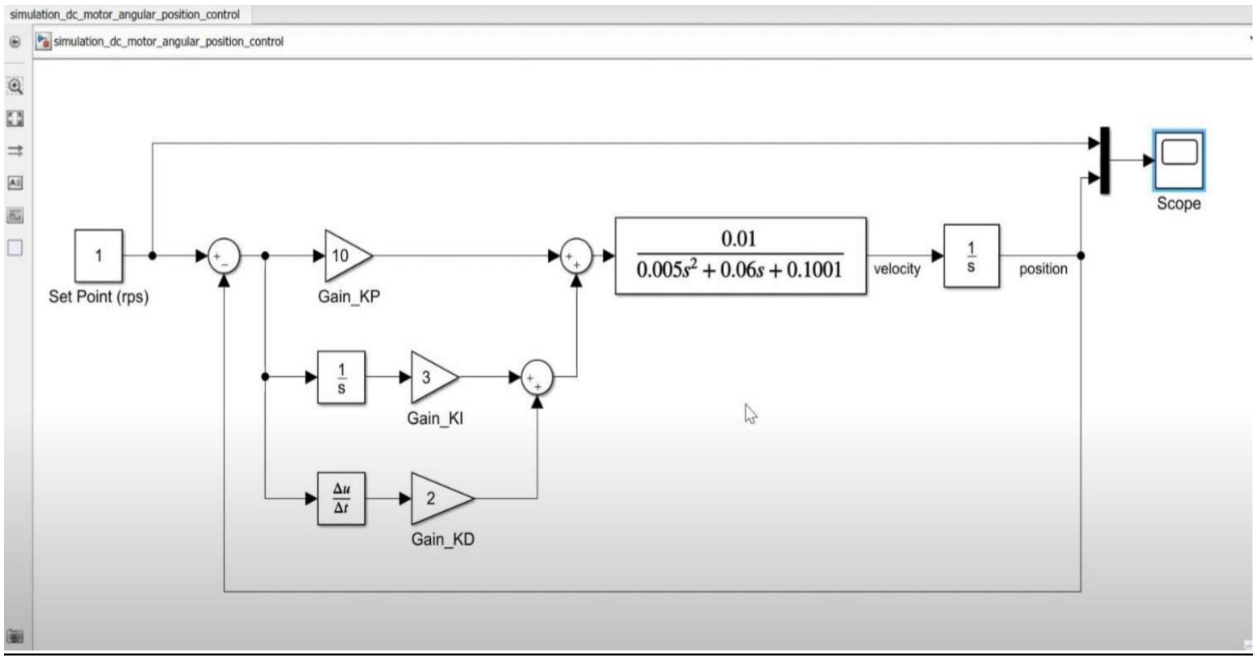
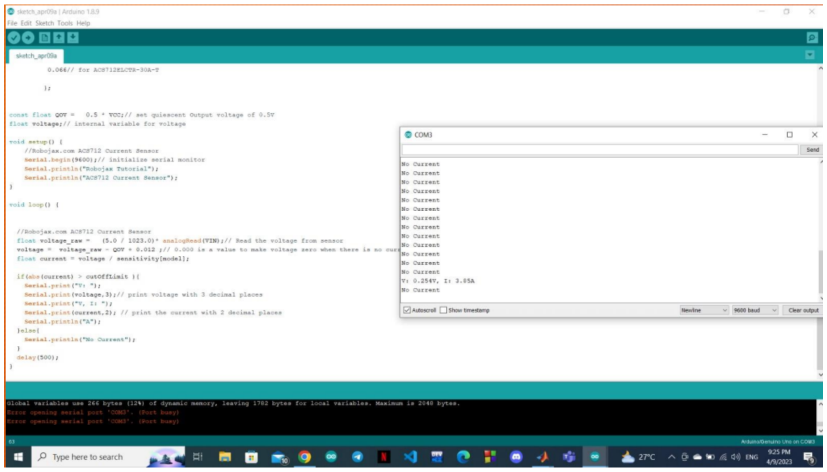

simulation_dc_motor_angular_position_control
1
Set Point (rps)
10
Gain_KP
3
Gain_KI
2
Gain_KD
$\frac{1}{s}$
$\frac{\Delta u}{\Delta t}$
$\frac{0.01}{0.005s^2+0.06s+0.1001}$
velocity
position
Scope

Figure 14 Model (2)

Test components

1. Test Current sensor
2. Test Voltage sensor
3. Test Dc motor / Test the whole system

Test 1 :

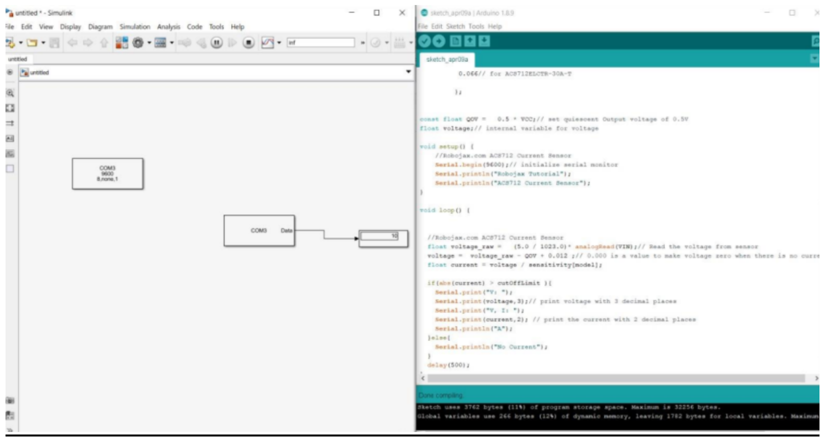


```
         0.066// for ACS712ELCTR-30A-T

        };


const float QOV =   0.5 * VCC;// set quiescent Output voltage of 0.5V
float voltage;// internal variable for voltage

void setup() {
    //Robojax.com ACS712 Current Sensor
    Serial.begin(9600);// initialize serial monitor
    Serial.println("Robojax Tutorial");
    Serial.println("ACS712 Current Sensor");
}

void loop() {


  //Robojax.com ACS712 Current Sensor
  float voltage_raw =   (5.0 / 1023.0)* analogRead(VIN);// Read the voltage from sensor
  voltage =  voltage_raw - QOV + 0.012 ;// 0.000 is a value to make voltage zero when there is no curr
  float current = voltage / sensitivity[model];

  if(abs(current) > cutOffLimit ){
    Serial.print("V: ");
    Serial.print(voltage,3);// print voltage with 3 decimal places
    Serial.print("V, I: ");
    Serial.print(current,2); // print the current with 2 decimal places
    Serial.println("A");
  }else{
    Serial.println("No Current");
  }
  delay(500);
}
```

```
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
No Current
V: 0.254V, I: 3.85A
No Current
```

```
Global variables use 266 bytes (12%) of dynamic memory, leaving 1782 bytes for local variables. Maximum is 2048 bytes.
Error opening serial port 'COM3'. (Port busy)
Error opening serial port 'COM3'. (Port busy)
```

Figure 15 Arduino IDE with matlab



untitled * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
untitled
COM3
9600
8,none,1
COM3
Data
10
sketch_apr09a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_apr09a

```
         0.066// for ACS712ELCTR-30A-T

        };


const float QOV =   0.5 * VCC;// set quiescent Output voltage of 0.5V
float voltage;// internal variable for voltage

void setup() {
    //Robojax.com ACS712 Current Sensor
    Serial.begin(9600);// initialize serial monitor
    Serial.println("Robojax Tutorial");
    Serial.println("ACS712 Current Sensor");
}

void loop() {


  //Robojax.com ACS712 Current Sensor
  float voltage_raw =   (5.0 / 1023.0)* analogRead(VIN);// Read the voltage from sensor
  voltage =  voltage_raw - QOV + 0.012 ;// 0.000 is a value to make voltage zero when there is no curre
  float current = voltage / sensitivity[model];

  if(abs(current) > cutOffLimit ){
    Serial.print("V: ");
    Serial.print(voltage,3);// print voltage with 3 decimal places
    Serial.print("V, I: ");
    Serial.print(current,2); // print the current with 2 decimal places
    Serial.println("A");
  }else{
    Serial.println("No Current");
  }
  delay(500);
```

```
Sketch uses 3762 bytes (11%) of program storage space. Maximum is 32256 bytes.
Global variables use 266 bytes (12%) of dynamic memory, leaving 1782 bytes for local variables. Maximum
```

Figure 15 Arduino IDE with matlab

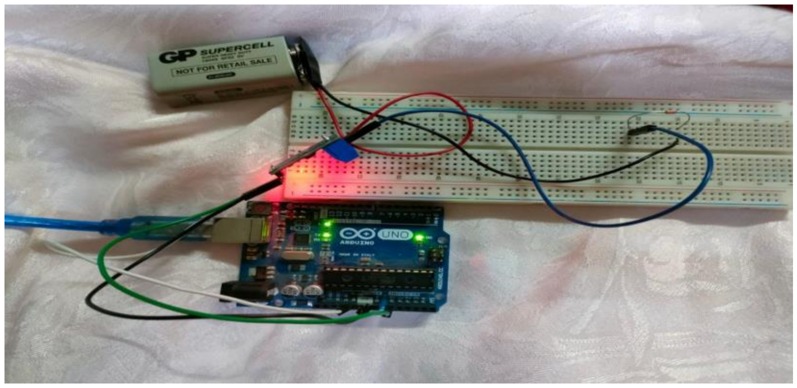
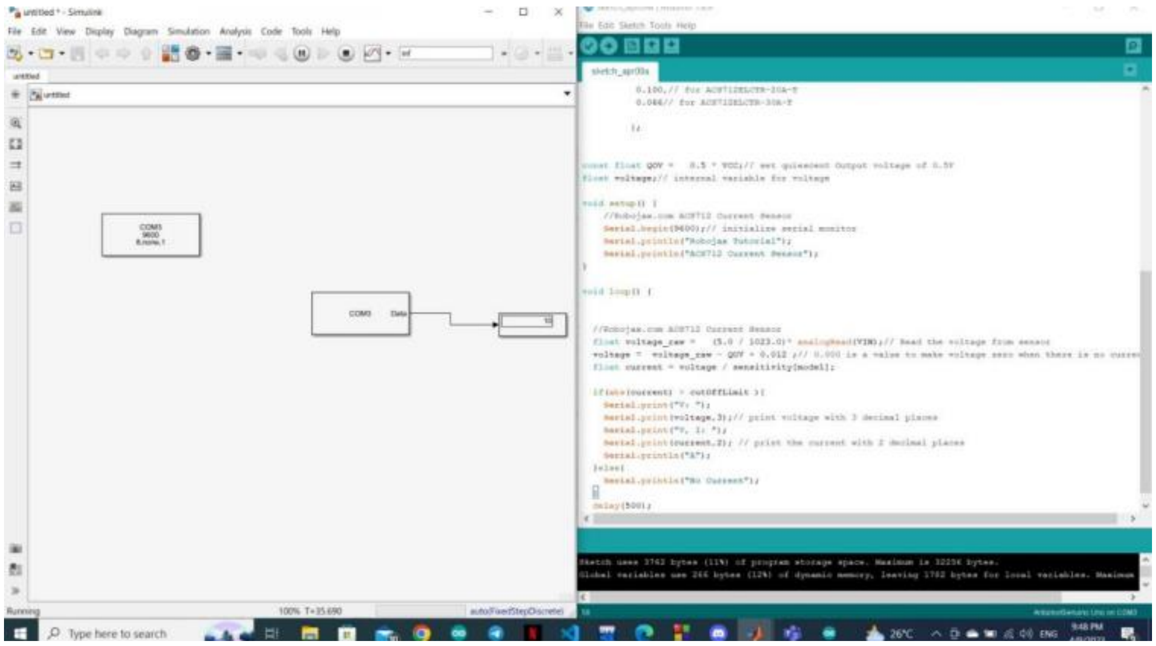


Figure 16 connect arduino with current sensor

Test 2 :



```
        0.100,// for ACS712ELCTR-20A-T
        0.066// for ACS712ELCTR-30A-T

       };


const float QOV =   0.5 * VCC;// set quiescent Output voltage of 0.5V
float voltage;// internal variable for voltage

void setup() {
    //Robojax.com ACS712 Current Sensor
    Serial.begin(9600);// initialize serial monitor
    Serial.println("Robojax Tutorial");
    Serial.println("ACS712 Current Sensor");
}

void loop() {


  //Robojax.com ACS712 Current Sensor
  float voltage_raw =   (5.0 / 1023.0)* analogRead(VIN);// Read the voltage from sensor
  voltage =  voltage_raw - QOV + 0.012 ;// 0.000 is a value to make voltage zero when there is no curre
  float current = voltage / sensitivity[model];

  if(abs(current) > cutOffLimit ){
    Serial.print("V: ");
    Serial.print(voltage,3);// print voltage with 3 decimal places
    Serial.print("V, I: ");
    Serial.print(current,2); // print the current with 2 decimal places
    Serial.println("A");
  }else{
    Serial.println("No Current");
  }
  delay(500);
```

```
Sketch uses 3762 bytes (11%) of program storage space. Maximum is 32256 bytes.
Global variables use 266 bytes (12%) of dynamic memory, leaving 1782 bytes for local variables. Maximum
```

Figure 17 Arduino uno with matlab

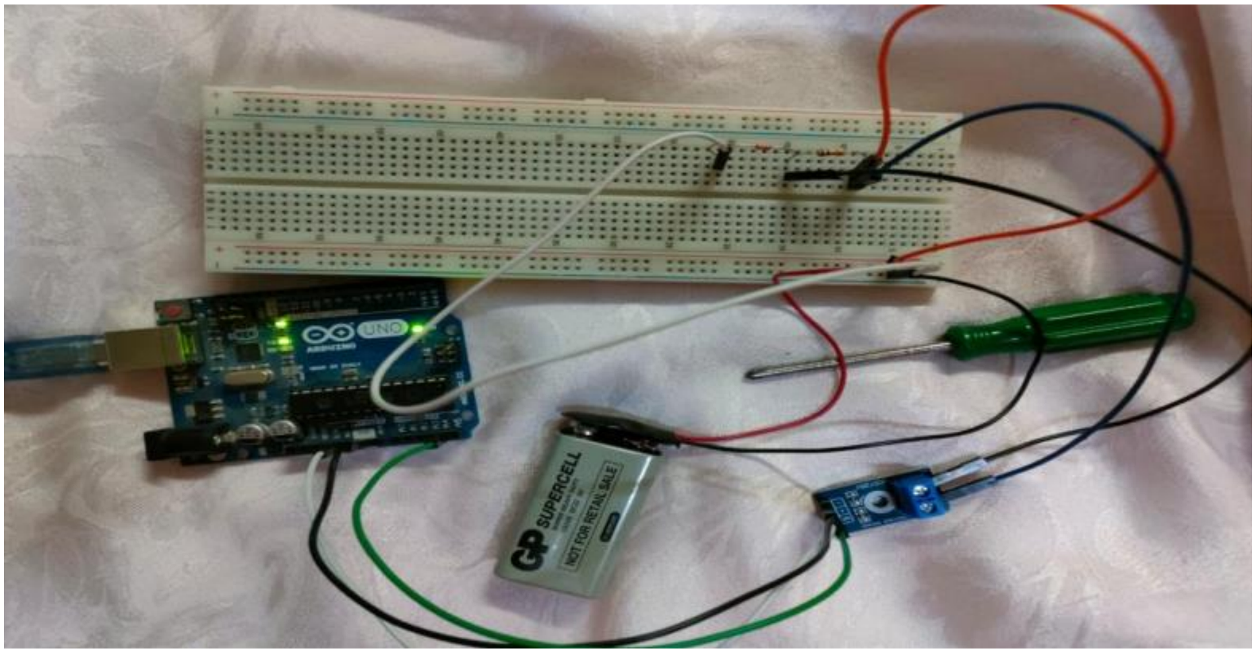
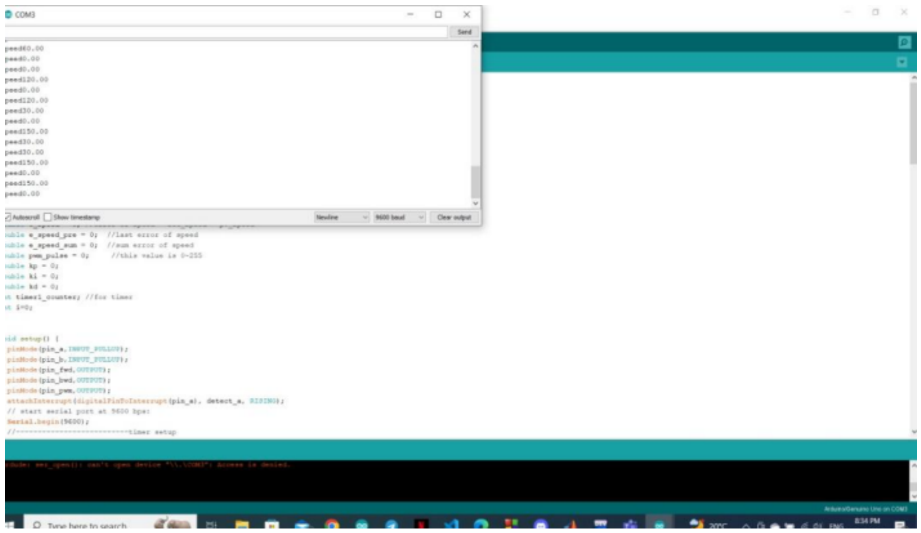


Figure 18 connections arduino uno and voltage sensor

Test 3 :

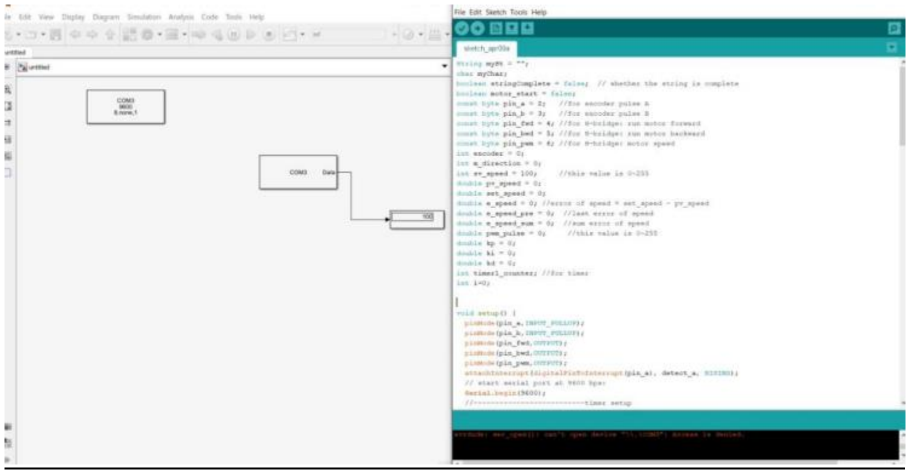


```
COM3
peed60.00
peed0.00
peed0.00
peed120.00
peed0.00
peed120.00
peed30.00
peed0.00
peed150.00
peed30.00
peed30.00
peed150.00
peed0.00
peed150.00
peed0.00
```

```
uble e_speed_pre = 0;  //last error of speed
uble e_speed_sum = 0;  //sum error of speed
uble pwm_pulse = 0;     //this value is 0-255
uble kp = 0;
uble ki = 0;
uble kd = 0;
t timer1_counter; //for timer
t i=0;

id setup() {
pinMode(pin_a,INPUT_PULLUP);
pinMode(pin_b,INPUT_PULLUP);
pinMode(pin_fwd,OUTPUT);
pinMode(pin_bwd,OUTPUT);
pinMode(pin_pwm,OUTPUT);
attachInterrupt(digitalPinToInterrupt(pin_a), detect_a, RISING);
// start serial port at 9600 bps:
Serial.begin(9600);
//---------------------------timer setup
```

```
rdude: ser_open(): can't open device "\\.\COM3": Access is denied.
```

Figure19 Arduino IDE



File Edit View Display Diagram Simulation Analysis Code Tools Help
untitled
COM3
9600
8,none,1
COM3
Data
100
File Edit Sketch Tools Help
sketch_apr03a

```
String myS = "";
char myChar;
boolean stringComplete = false;  // whether the string is complete
boolean motor_start = false;
const byte pin_a = 2;   //for encoder pulse A
const byte pin_b = 3;   //for encoder pulse B
const byte pin_fwd = 4; //for H-bridge: run motor forward
const byte pin_bwd = 5; //for H-bridge: run motor backward
const byte pin_pwm = 6; //for H-bridge: motor speed
int encoder = 0;
int m_direction = 0;
int sv_speed = 100;     //this value is 0-255
double pv_speed = 0;
double set_speed = 0;
double e_speed = 0; //error of speed = set_speed - pv_speed
double e_speed_pre = 0;  //last error of speed
double e_speed_sum = 0;  //sum error of speed
double pwm_pulse = 0;     //this value is 0-255
double kp = 0;
double ki = 0;
double kd = 0;
int timer1_counter; //for timer
int i=0;

void setup() {
  pinMode(pin_a,INPUT_PULLUP);
  pinMode(pin_b,INPUT_PULLUP);
  pinMode(pin_fwd,OUTPUT);
  pinMode(pin_bwd,OUTPUT);
  pinMode(pin_pwm,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(pin_a), detect_a, RISING);
  // start serial port at 9600 bps:
  Serial.begin(9600);
  //---------------------------timer setup
```

```
avrdude: ser_open(): can't open device "\\.\COM3": Access is denied.
```

Figure 20 Arduino with Matlab (test 1)

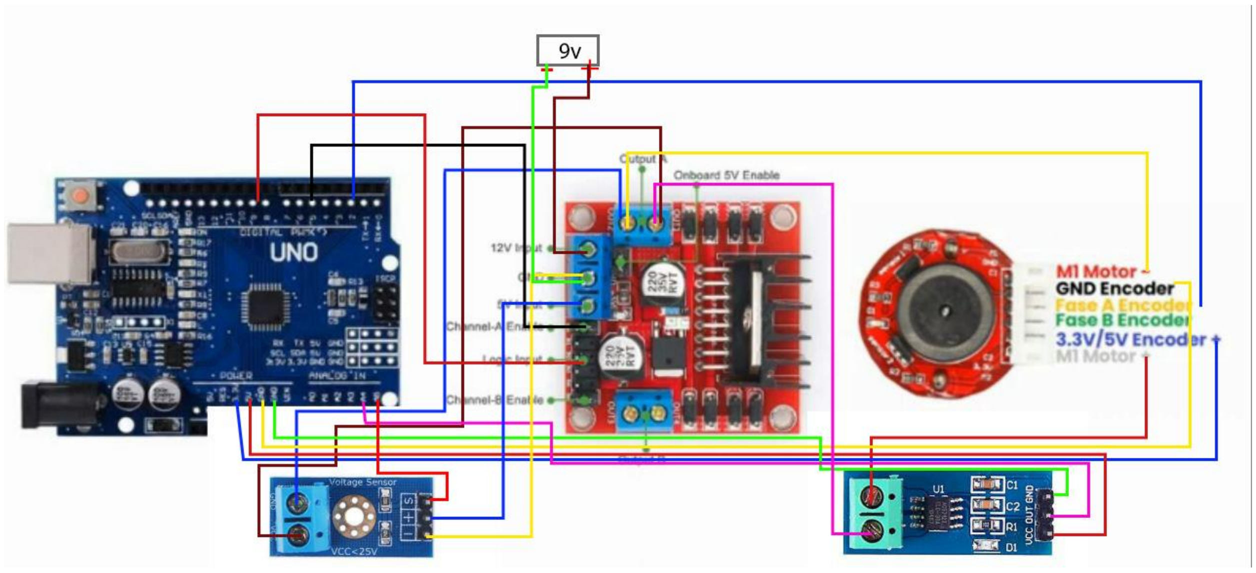


Figure 21 connections

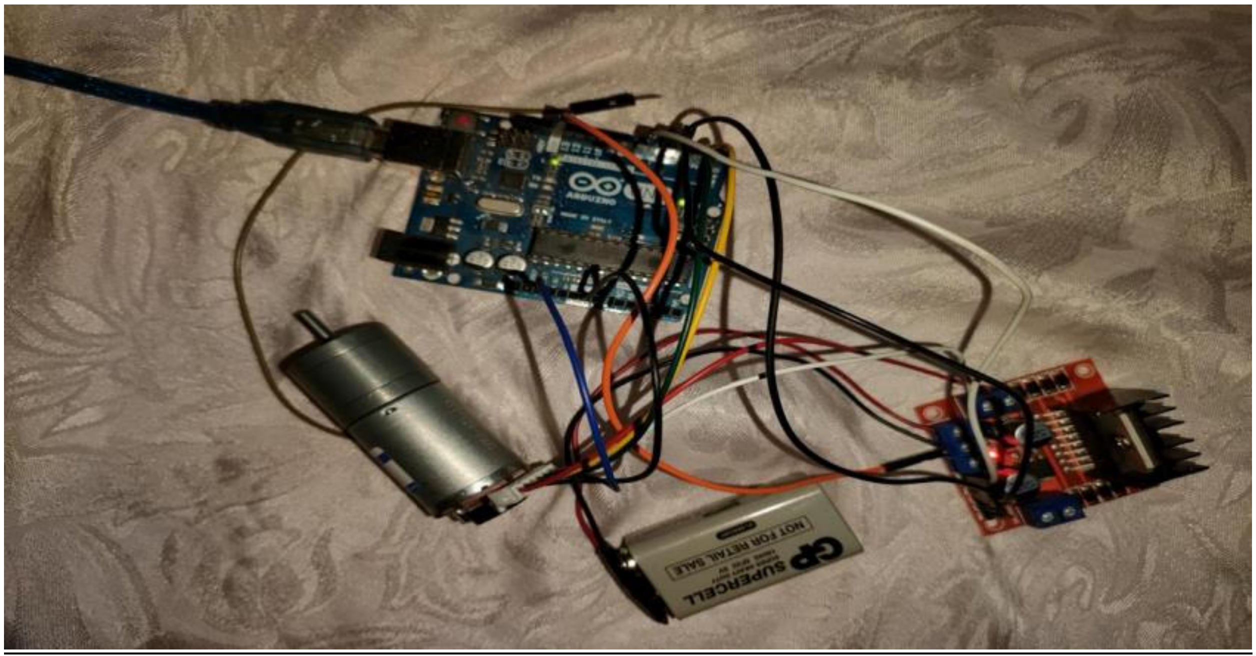
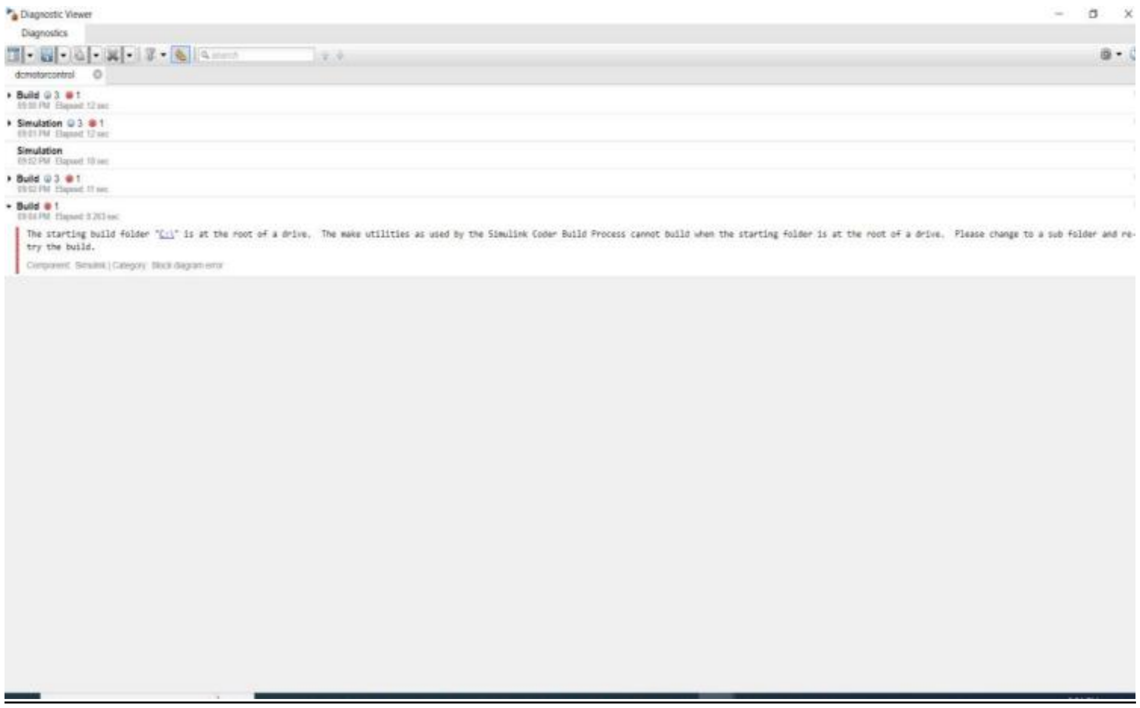


Figure 22 hardware



```
Build 3 1
09:00 PM  Elapsed: 12 sec
Simulation 3 1
09:01 PM  Elapsed: 12 sec
Simulation
09:02 PM  Elapsed: 10 sec
Build 3 1
09:02 PM  Elapsed: 11 sec
Build 1
09:04 PM  Elapsed: 3.263 sec
The starting build folder "C:\" is at the root of a drive.  The make utilities as used by the Simulink Coder Build Process cannot build when the starting folder is at the root of a drive.  Please change to a sub folder and re-
try the build.
Component: Simulink | Category: Block diagram error
```

<u>Figure 23 problem</u>

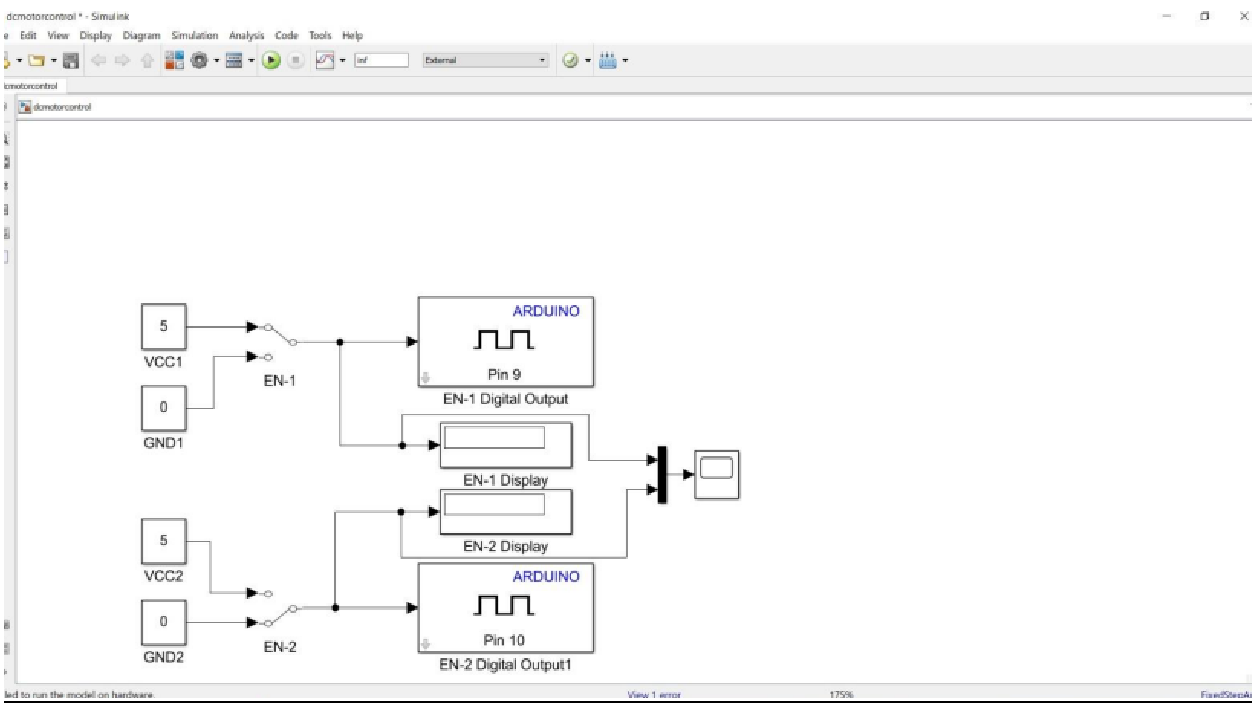


Figure 24 test 2

System identification

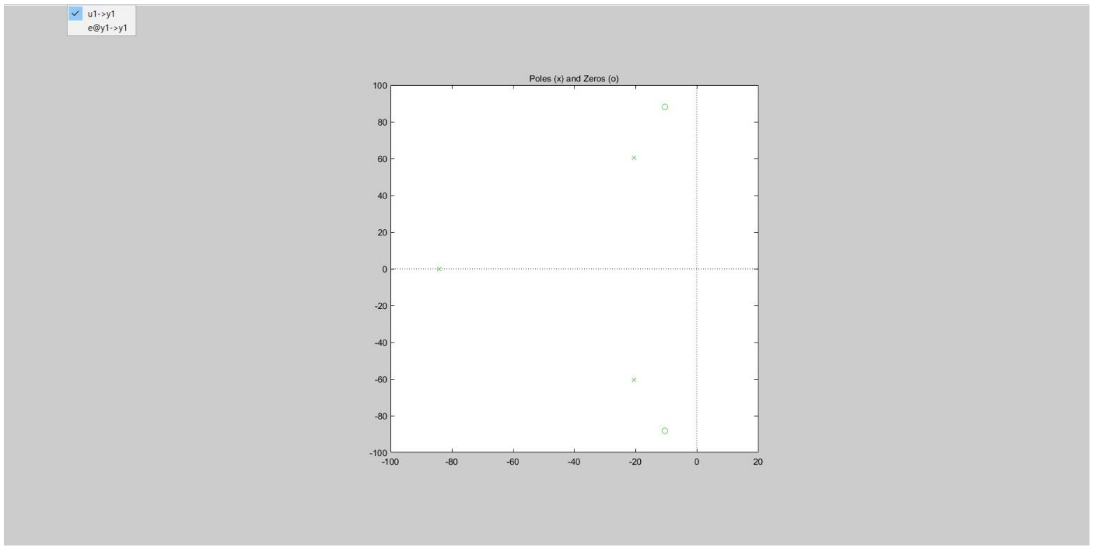


Figure 25 poles& zeros

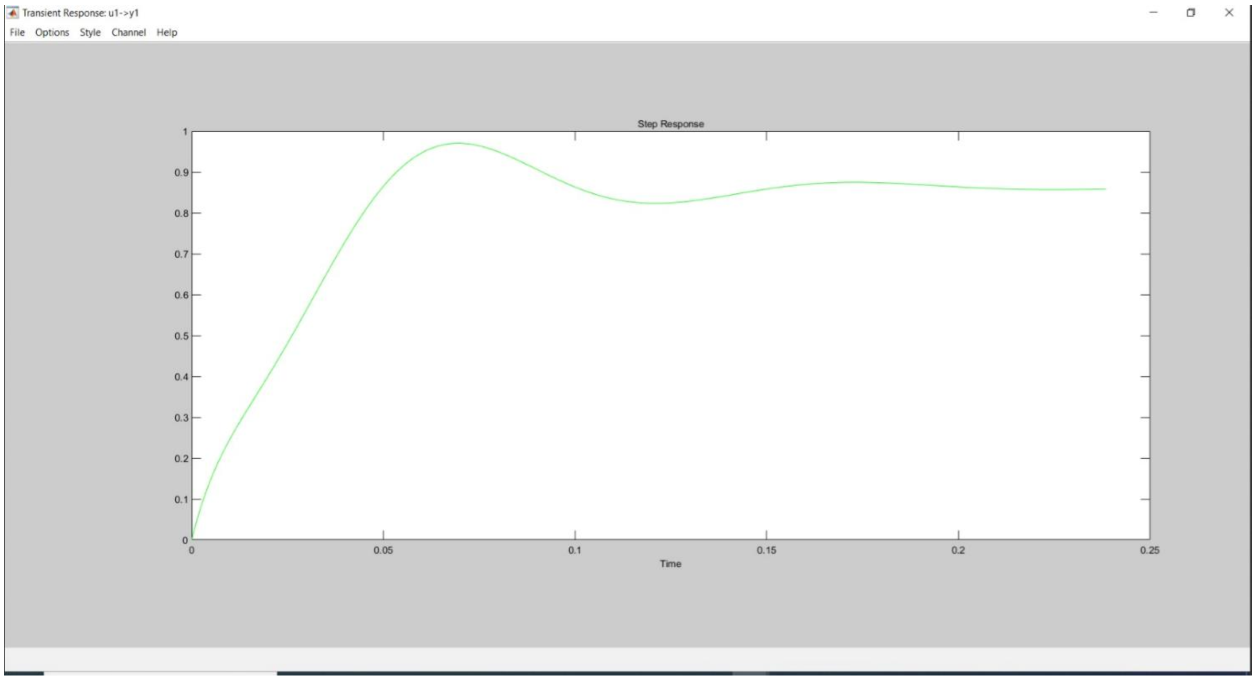


Figure 26 transient response

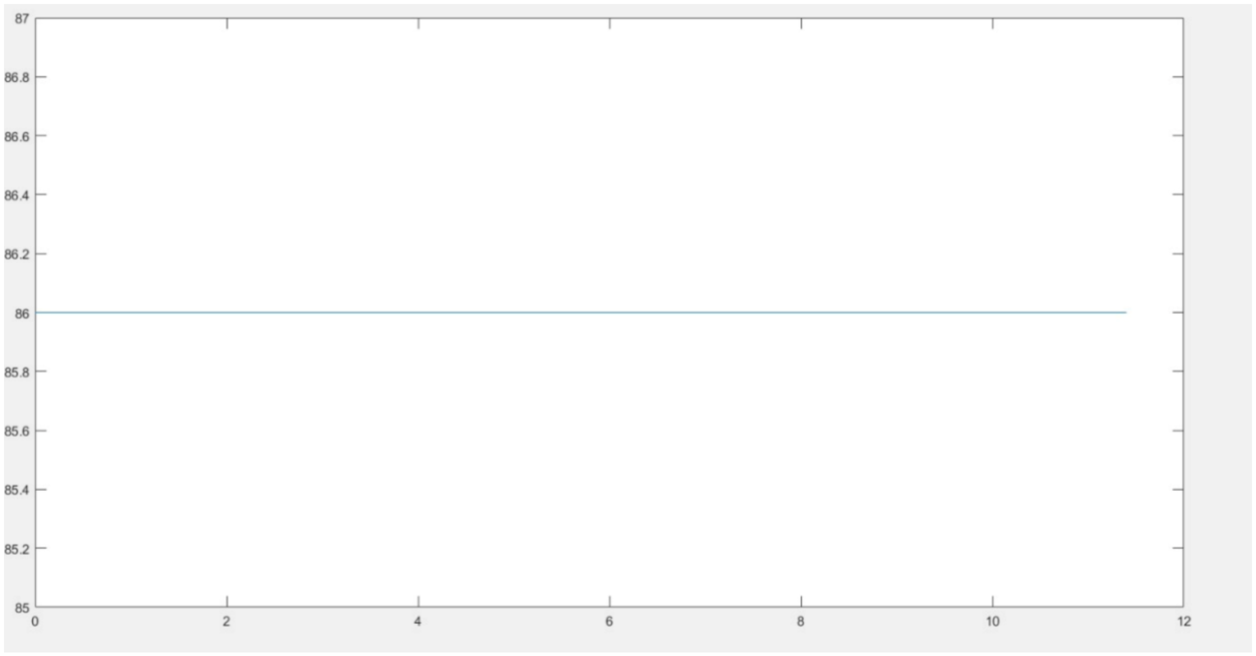


figure 27 system input

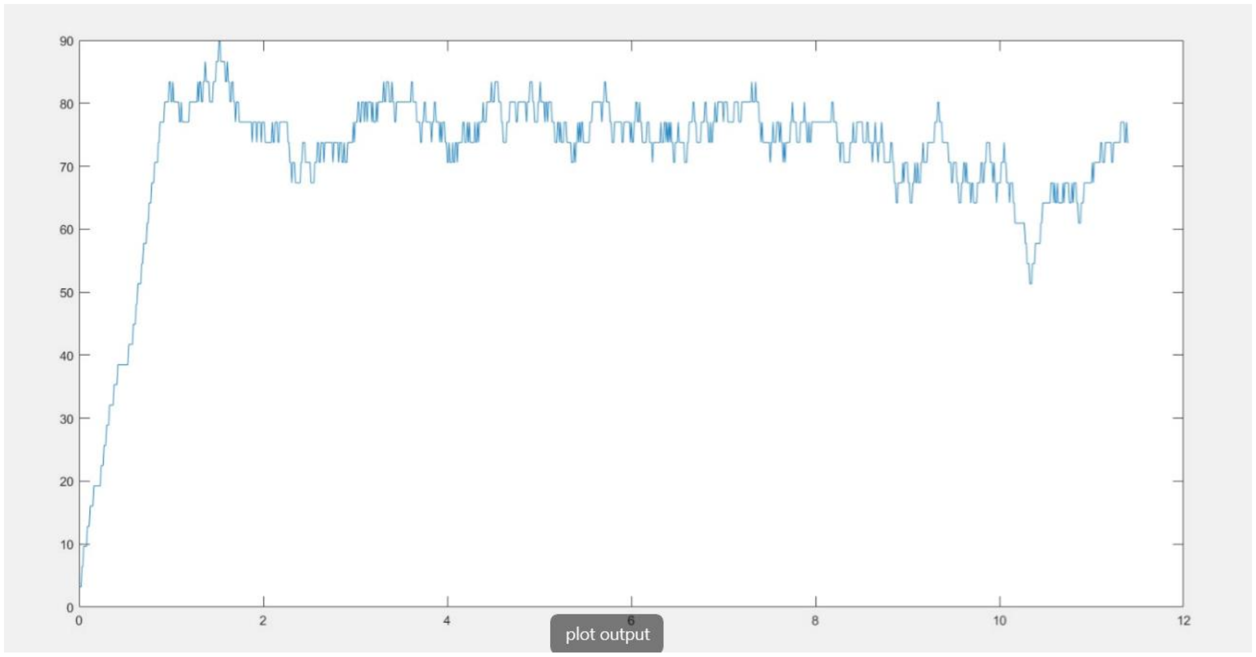
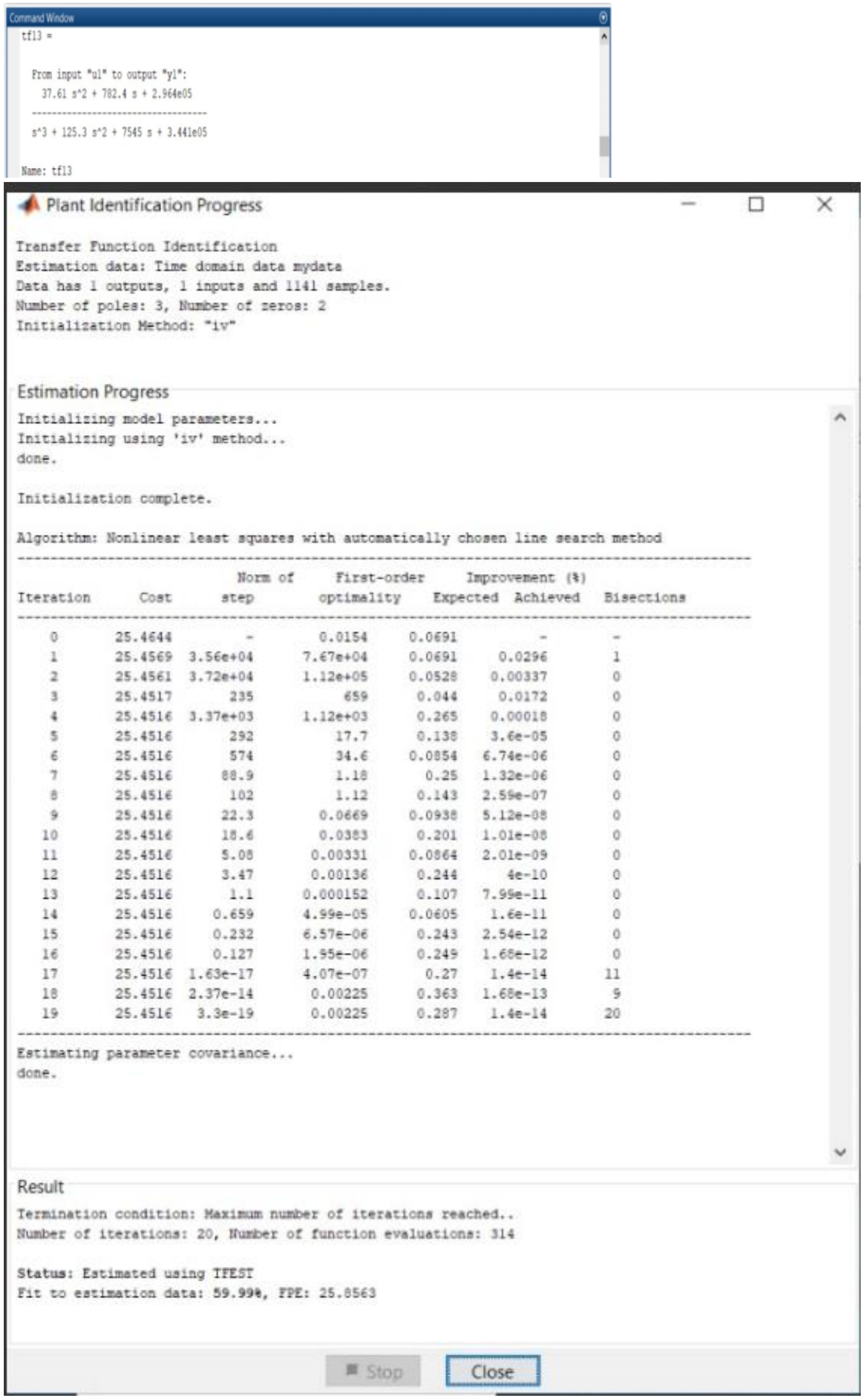


figure 28 system output



```
tf13 =

  From input "u1" to output "y1":
    37.61 s^2 + 782.4 s + 2.964e05
  ----------------------------------
  s^3 + 125.3 s^2 + 7545 s + 3.441e05

Name: tf13
```

```
Transfer Function Identification
Estimation data: Time domain data mydata
Data has 1 outputs, 1 inputs and 1141 samples.
Number of poles: 3, Number of zeros: 2
Initialization Method: "iv"
```

Estimation Progress

```
Initializing model parameters...
Initializing using 'iv' method...
done.

Initialization complete.

Algorithm: Nonlinear least squares with automatically chosen line search method
------------------------------------------------------------------------------------------
                         Norm of      First-order     Improvement (%)
Iteration      Cost       step         optimality    Expected  Achieved    Bisections
------------------------------------------------------------------------------------------
    0       25.4644         -            0.0154     0.0691         -           -
    1       25.4569  3.56e+04         7.67e+04     0.0691     0.0296          1
    2       25.4561  3.72e+04         1.12e+05     0.0528    0.00337          0
    3       25.4517       235              659      0.044     0.0172          0
    4       25.4516  3.37e+03         1.12e+03      0.265    0.00018          0
    5       25.4516       292             17.7      0.138    3.6e-05          0
    6       25.4516       574             34.6     0.0854   6.74e-06          0
    7       25.4516      88.9             1.18       0.25   1.32e-06          0
    8       25.4516       102             1.12      0.143   2.59e-07          0
    9       25.4516      22.3           0.0669     0.0938   5.12e-08          0
   10       25.4516      18.6           0.0383      0.201   1.01e-08          0
   11       25.4516      5.08          0.00331     0.0864   2.01e-09          0
   12       25.4516      3.47          0.00136      0.244      4e-10          0
   13       25.4516       1.1         0.000152      0.107   7.99e-11          0
   14       25.4516     0.659         4.99e-05     0.0605    1.6e-11          0
   15       25.4516     0.232         6.57e-06      0.243   2.54e-12          0
   16       25.4516     0.127         1.95e-06      0.249   1.68e-12          0
   17       25.4516  1.63e-17         4.07e-07       0.27    1.4e-14         11
   18       25.4516  2.37e-14          0.00225      0.363   1.68e-13          9
   19       25.4516   3.3e-19          0.00225      0.287    1.4e-14         20
------------------------------------------------------------------------------------------
Estimating parameter covariance...
done.
```

Result

```
Termination condition: Maximum number of iterations reached..
Number of iterations: 20, Number of function evaluations: 314

Status: Estimated using TFEST
Fit to estimation data: 59.99%, FPE: 25.8563
```

Figure 29 transfer function

Design PID controller

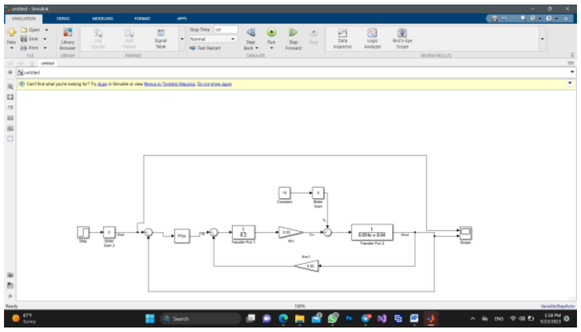
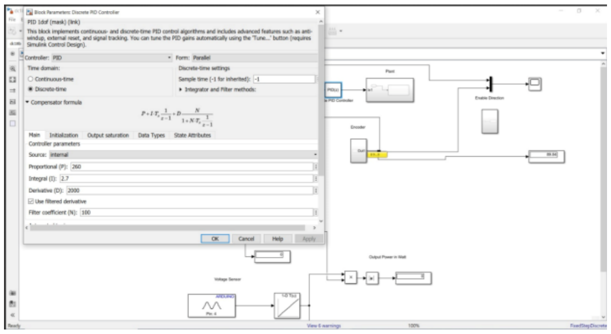


Figure 30 software pid



Block Parameters: Discrete PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID
Form: Parallel
Time domain: ( ) Continuous-time [✓ Discrete-time]
Discrete-time settings
Sample time (-1 for inherited): -1
Integrator and Filter methods:

Compensator formula

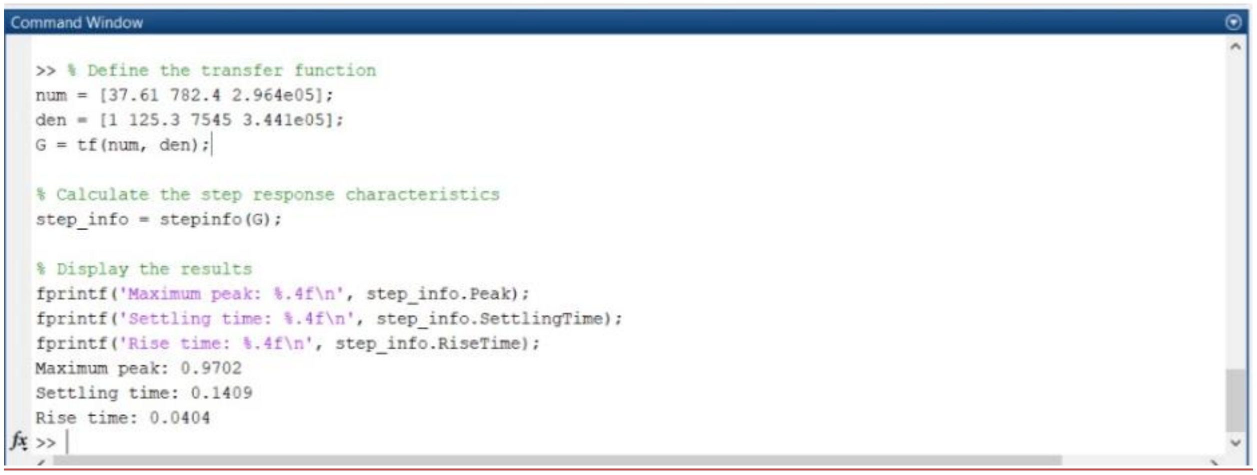
$$P + I \cdot T_s \frac{1}{z-1} + D \frac{N}{1 + N \cdot T_s \frac{1}{z-1}}$$

Main | Initialization | Output saturation | Data Types | State Attributes

Controller parameters
Source: internal
Proportional (P): 260
Integral (I): 2.7
Derivative (D): 2000
[✓] Use filtered derivative
Filter coefficient (N): 100

Figure 31 true values

Evaluate the performance



```
>> % Define the transfer function
num = [37.61 782.4 2.964e05];
den = [1 125.3 7545 3.441e05];
G = tf(num, den);

% Calculate the step response characteristics
step_info = stepinfo(G);

% Display the results
fprintf('Maximum peak: %.4f\n', step_info.Peak);
fprintf('Settling time: %.4f\n', step_info.SettlingTime);
fprintf('Rise time: %.4f\n', step_info.RiseTime);
Maximum peak: 0.9702
Settling time: 0.1409
Rise time: 0.0404
>>
```

<u>Figure 32 step response</u>

Maximum peak: 0.9702

Settling time: 0.1409

Rise time: 0.0404

Root locus

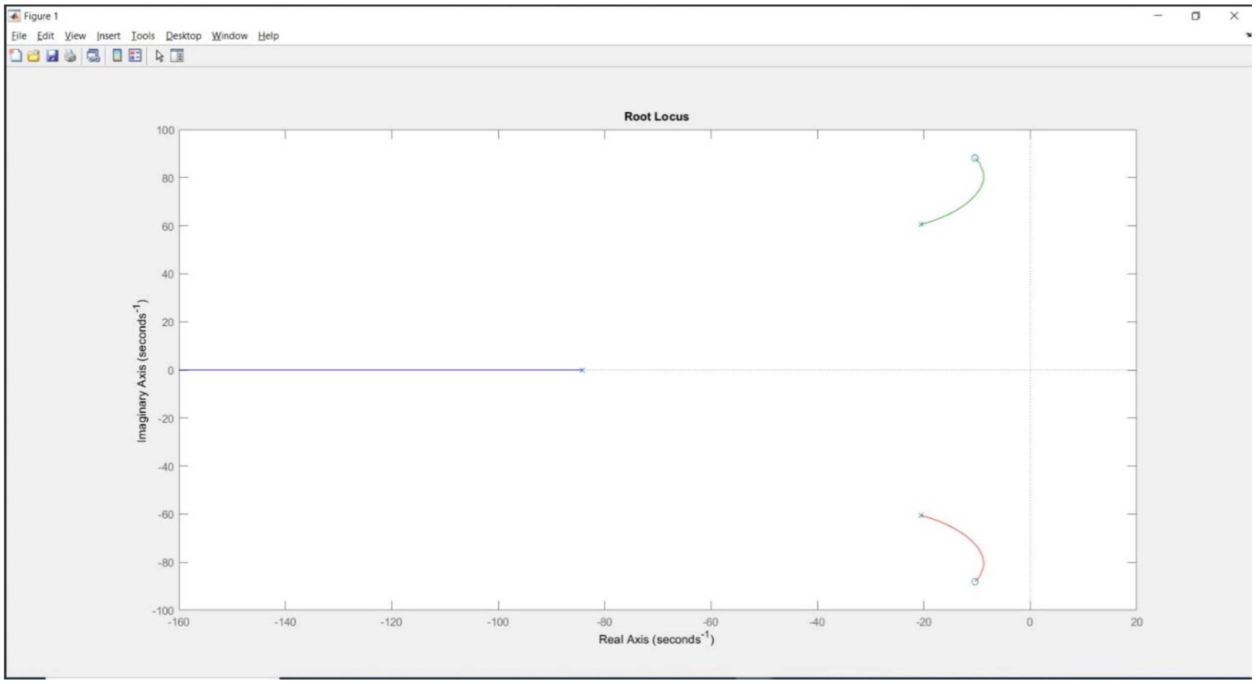
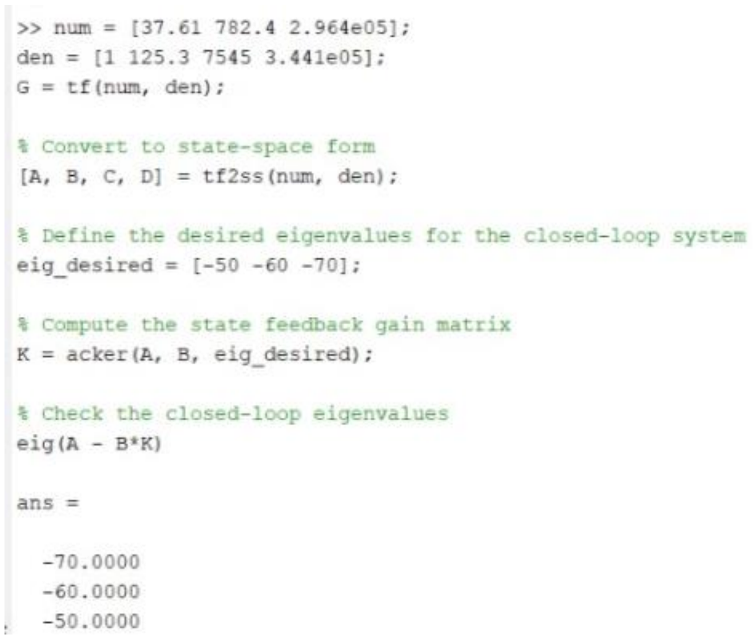


Figure 33 root locus

State feedback



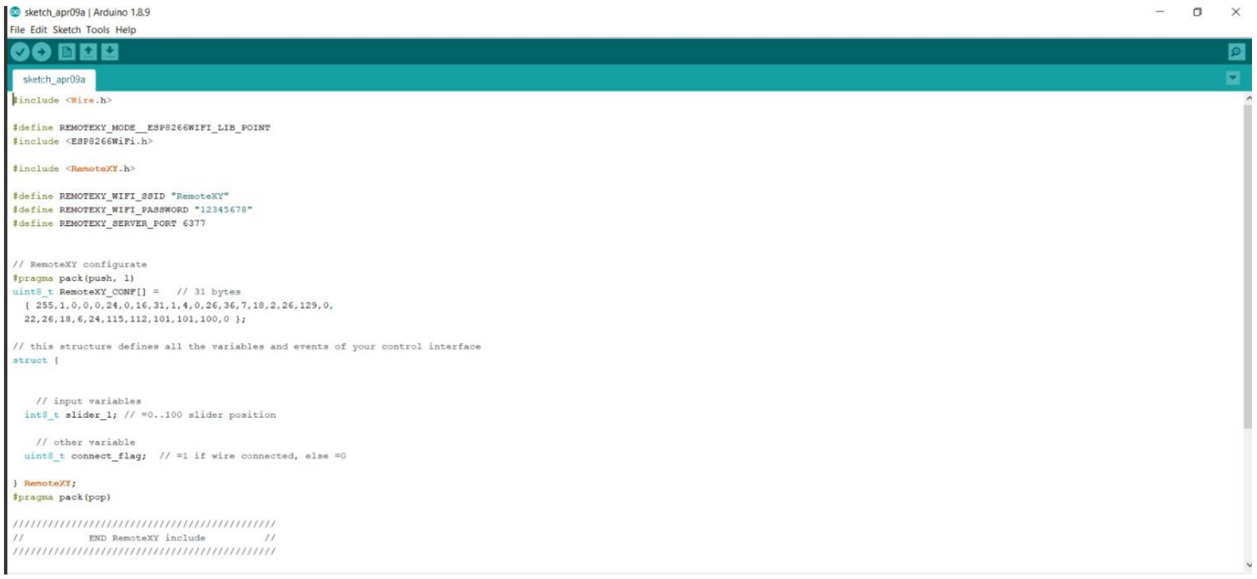
```
>> num = [37.61 782.4 2.964e05];
den = [1 125.3 7545 3.441e05];
G = tf(num, den);

% Convert to state-space form
[A, B, C, D] = tf2ss(num, den);

% Define the desired eigenvalues for the closed-loop system
eig_desired = [-50 -60 -70];

% Compute the state feedback gain matrix
K = acker(A, B, eig_desired);

% Check the closed-loop eigenvalues
eig(A - B*K)

ans =

  -70.0000
  -60.0000
  -50.0000
```

Figure 34 state feedback(eigen value)

Eigen values

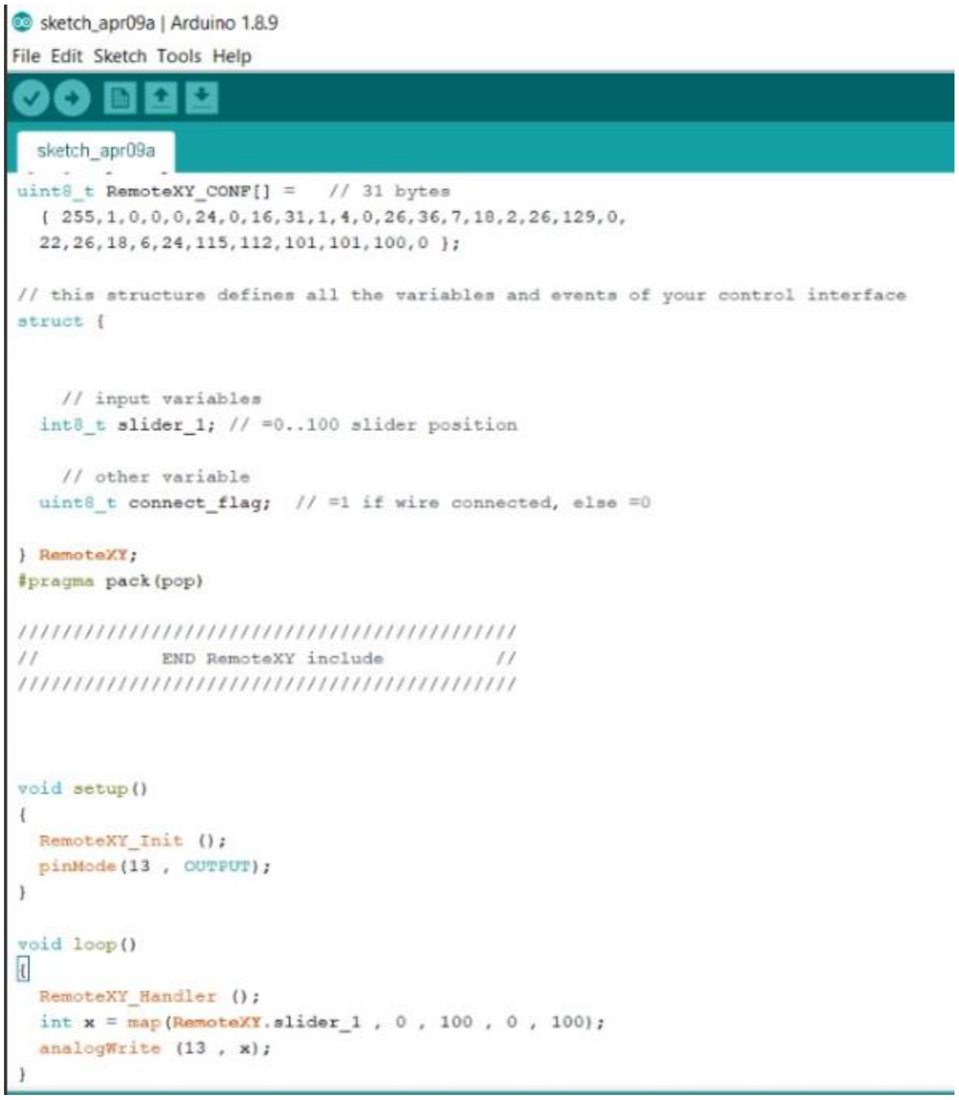
-70 -60 -50

Test ESP8266



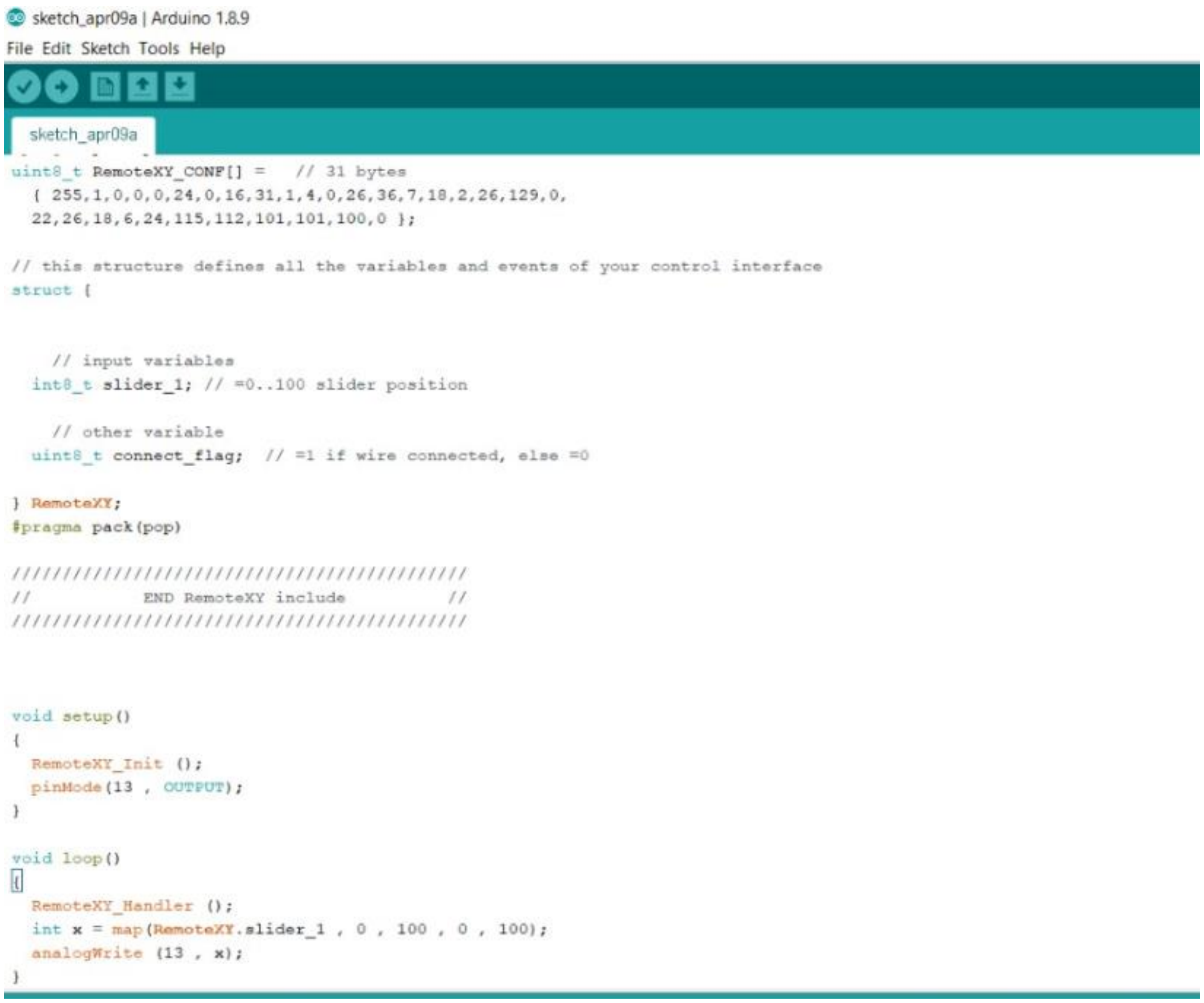
```
#include <Wire.h>

#define REMOTEXY_MODE__ESP8266WIFI_LIB_POINT
#include <ESP8266WiFi.h>

#include <RemoteXY.h>

#define REMOTEXY_WIFI_SSID "RemoteXY"
#define REMOTEXY_WIFI_PASSWORD "12345678"
#define REMOTEXY_SERVER_PORT 6377


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =   // 31 bytes
  { 255,1,0,0,0,24,0,16,31,1,4,0,26,36,7,18,2,26,129,0,
  22,26,18,6,24,115,112,101,101,100,0 };

// this structure defines all the variables and events of your control interface
struct {


    // input variables
  int8_t slider_1; // =0..100 slider position

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//            END RemoteXY include          //
/////////////////////////////////////////////
```

Figure 35 code ESP



```
uint8_t RemoteXY_CONF[] =   // 31 bytes
  { 255,1,0,0,0,24,0,16,31,1,4,0,26,36,7,18,2,26,129,0,
  22,26,18,6,24,115,112,101,101,100,0 };

// this structure defines all the variables and events of your control interface
struct {


    // input variables
  int8_t slider_1; // =0..100 slider position

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//           END RemoteXY include          //
/////////////////////////////////////////////



void setup()
{
  RemoteXY_Init ();
  pinMode(13 , OUTPUT);
}

void loop()
{
  RemoteXY_Handler ();
  int x = map(RemoteXY.slider_1 , 0 , 100 , 0 , 100);
  analogWrite (13 , x);
}
```

Figure 36 code ESP



```
uint8_t RemoteXY_CONF[] =   // 31 bytes
  { 255,1,0,0,0,24,0,16,31,1,4,0,26,36,7,18,2,26,129,0,
  22,26,18,6,24,115,112,101,101,100,0 };

// this structure defines all the variables and events of your control interface
struct {


    // input variables
  int8_t slider_1; // =0..100 slider position

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//           END RemoteXY include          //
/////////////////////////////////////////////



void setup()
{
  RemoteXY_Init ();
  pinMode(13 , OUTPUT);
}

void loop()
{
  RemoteXY_Handler ();
  int x = map(RemoteXY.slider_1 , 0 , 100 , 0 , 100);
  analogWrite (13 , x);
}
```

Figure 37 code ESP

IoT platform

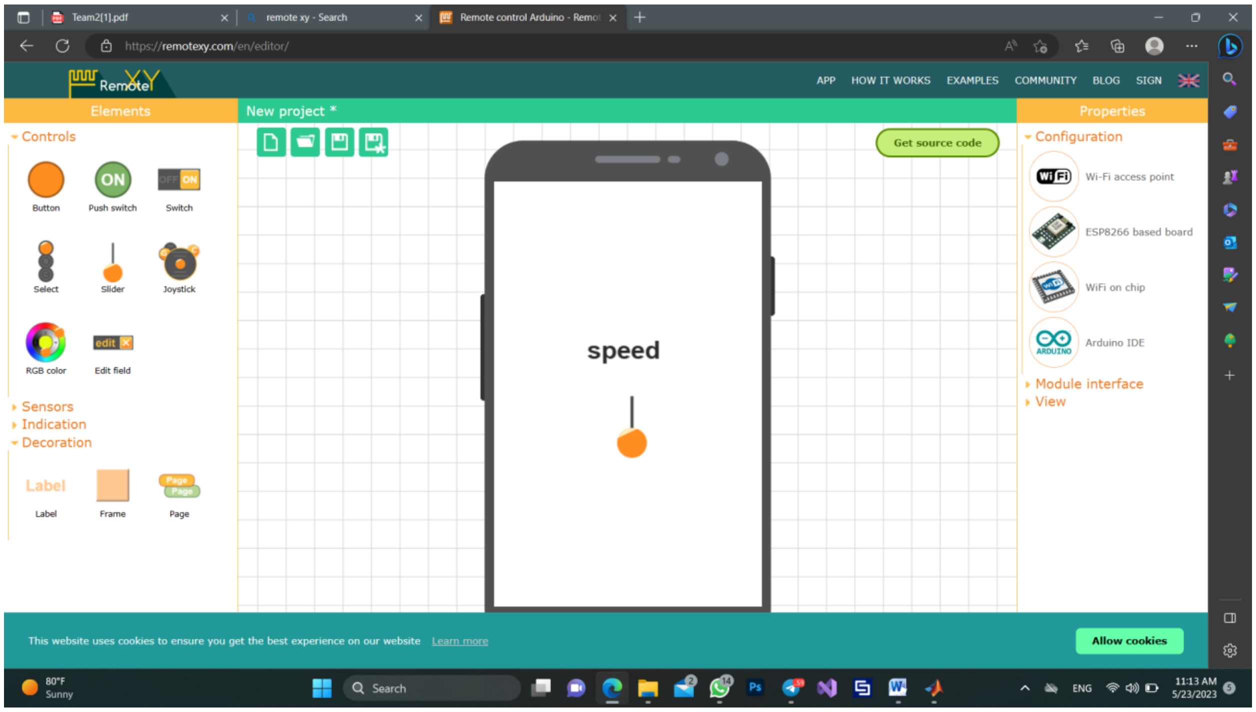


Figure 38

Conclusion

In conclusion, the speed control of a DC motor using Arduino Uno and a microcontroller PID offers a precise and efficient way to regulate the motor’s speed. The combination of the Arduino Uno and microcontroller PID algorithm allows for real-time adjustments to the motor’s speed and helps maintain a constant speed even under varying loads.

The Arduino Uno provides an easy-to-use platform for programming and controlling the motor, while the PID algorithm provides accurate and stable control of the motor’s speed. Together, they form a powerful tool for controlling DC motors in a wide variety of applications, from robotics to industrial automation.

Overall, the speed control of a DC motor using Arduino Uno and a microcontroller PID offers a cost-effective and reliable solution for precise motor speed control. As technology continues to evolve and improve, the possibilities for this type of control will only increase, making it an exciting area for further exploration and development.

References

[1]. Peerzada, P., Larik, W. H., & Mahar, A. A. (2021). DC Motor Speed Control Through Arduino and L298N Motor Driver Using PID Controller. International Journal of Electrical Engineering & Emerging Technology, 4(2), 21-24.

[2]. Elsrogy, W. M., Fkirin, M. A., & Hassan, M. M. (2013, May). Speed control of DC motor using PID controller based on artificial intelligence techniques. In 2013 International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 196-201). IEEE.

[3]. Gavran, M., Fruk, M., & Vujisić, G. (2017, May). PI controller for DC motor speed realized with Arduino and Simulink. In 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1557-1561). IEEE.

[4]. Ma'arif, A., & Setiawan, N. R. (2021). Control of DC motor using integral state feedback and comparison with PID: simulation and arduino implementation. Journal of Robotics and Control (JRC), 2(5), 456-461.

5]. Hat, M. K., Ibrahim, B. S. K. K., Mohd, T. A. T., & Hassan, M. K. (2015). Model based design of pid controller for bldc motor with implementation of embedded arduino mega controller. ARPN Journal of Engineering and Applied Sciences, 10(19), 8588-8594.