|  |
| --- |
| Robotics and Mechatronics engineering - Bachelor degree - Monash University |
| DC MOTOR PARAMETER ESTIMATION  Mechatronics system analyses and modeling |
| |  |  |  | | --- | --- | --- | |  |  | **MTE403** | |

**Course**

Mechatronics system analyses and modeling

**MTE403**

**Supervisor**

Dr. Bahaa Eldin Mohamed Naser

Eng. Waleed El-Badry

Eng. Eslam Ibrahim

**Submitted by**

Student ID

Abdallah Sayed Shahat 91645

Mahmoud Tawfik Mahmoud 95933

Ammar Abdelnasser Taghian 38243

David Maged Youssef 91489

Remon Lamey Lemby Batrous 95893

**Table of content**

* **Simulation 1……………………………………………….…….………….3**
  + Project objective…………………………………………....................
  + Project overview……………………………………………………....
  + Main used components(HW&SW) ………….…….…………………
  + Controlling the system and testing ……………………………………
  + Arduino Code...………………………………………………………..
* **Simulation 2…………………………………………………….………….4**
* System mathematical model……….………………………………….
* System model using MATLAB……….………………………………
* **Simulation 3………………………………………….………………..…...6**
* HW overview & connection…………….……………………………
* Full control using Arduino……………………………………………
* Getting measurements (actual data) …………………………….……
* Arduino Code...……………………………………………………….
* **Simulation 4 (Parameter estimation) …………………………………….8**
* Explain the concept……………………………….……………………
* Prepare logging data………………………………….……….……….
* Loging the data (MATLAB)…………………………………………..
* Getting your parameter………………………………………………...
* Arduino Code…………………………………………………………..

**List of figures**

|  |  |
| --- | --- |
| Figure 1.a | 4 |
| Figure 2.a | 6 |
| Figure 2.b | 8 |
| Figure 3.a | 12 |
| Figure3.b | 13 |
| Figure 4.a | 15 |
| Figure 4.b | 15 |
| Figure 4.c | 15 |
| Figure 4.d | 16 |

**List of tables**

|  |  |
| --- | --- |
| Table 1 | 13 |

**List of Codes**

|  |  |
| --- | --- |
| code.1 | 5,6 |
| code.2 | 8~11 |
| code.3 | 17~21 |

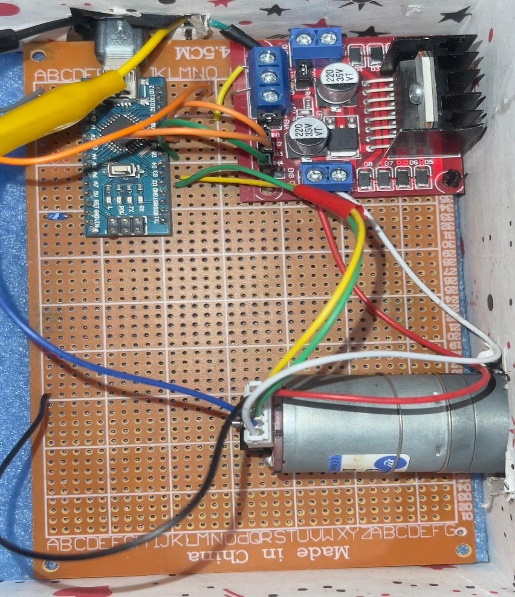
**Simulation one: -**

Project main objective

-in this project we aim to measure and analyze the parameters of an unknown Mechatronics system.

Project overview

-in this project the procedure of analyzing and measuring the parameters is spilt into two parts part one the connection of the DC-Motor with the other hardware components and part two the software using a series of programs.

-List of hardware components

1- DC-MOTOR with encoder (GM-25-370-CE).

2- ARDUINO NANO.

3- H-BRIDGE(LM-298).

4- A 12-VOLTS POWER SUPPLY.

5- A PCB BOARD.

List of used software programs

1. Arduino IDE-(2.0.3).
2. Matlab-R2021a.
3. serialplot-0.12.0-win32.
4. Proteus.

Figure 1.a

Controlling and testing the system

First we started by uploading the Arduino code on the Arduino Nano chip This code main function is that there are a set of variables that stands for the pins connection on the Arduino where pin number seven and pin number eight are used for setting the direction either clock-wise or antilock-wise and pin number two and pin number three are used for the encoder phases phase-A and Phase-B respectively and pin number five as the Motor Enable and the 3.3v pin is connected to the positive terminal of the encoder and all is connected to a common ground. The H-bridge is connected with pins number seven and eight and is also connected with the motor supplying it with voltage and ground throw two pins. Where pins number seven and eight are responsible for sending the direction of motion from the serial monitor throw the Arduino and then to the motor throw the H-bridge and sending the value of the PWM throw pin number five to the H-bridge then to the motor to control the speed of the motor and it is sent in the form of voltage. Where pin number two and pin number three support interrupt peripheral and pins seven and eight are digital input and output pins and pin five support PWM.

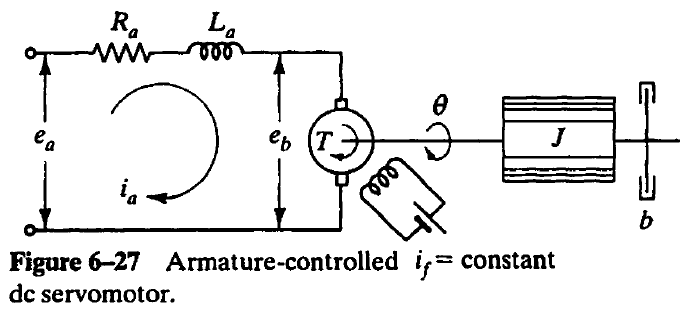
**Arduino Code:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   COURSE  : MTE504 MECHATRONICS II  
   PROJECT : SPEED CHANGER  
   DATE    : 04/08/2021  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
// Serial Port Used  
#**define** ser Serial  
  
// Motor PWM Pin  
**byte** pinMotor = 5;  
  
// Direvction Pin  
**byte** pinDir1 = 7;  
**byte** pinDir2 = 8;  
  
**void** setup() {  
  // Setup serial baudrate  
  ser.begin(9600);  
  
  // Set pins as output  
  pinMode(pinMotor, OUTPUT);  
  pinMode(pinDir1, OUTPUT);  
   pinMode(pinDir2, OUTPUT);  
  
  // Default to 0  
  digitalWrite(pinMotor, LOW);  
  digitalWrite(pinDir1, LOW);  
   digitalWrite(pinDir2, LOW);  
}  
  
**void** loop() {  
}  
  
**void** serialEvent() {  
  if (ser.available()) {  
    **char** cmd = ser.read();  
    switch (cmd) {  
      case 's':  
        // Set Speed to 100%  
        analogWrite(pinMotor, 1.0 \* 255.0);  
        break;  
  
      case 'c':  
        // Clockwise  
        digitalWrite(pinDir1, HIGH);  
        break;  
  
      case 'a':  
        // Anti Clockwise  
        digitalWrite(pinDir2, HIGH);  
        break;  
  
      case 'p':  
        // Set Speed to 0%  
        analogWrite(pinMotor, 0.0);  
          digitalWrite(pinDir1, LOW);  
   digitalWrite(pinDir2, LOW);  
          
        break;  
  
      default:  
        break;  
    }  
  }  
}

Code.1

**Simulation two:**

**System mathematical model**

 For any system of any type getting it to your paper is easier and cheaper to deal with so this gives you more space to try it with true response and results and that’s the beauty of mathematics and this operation is the spirit of modeling and in this field a powerful technology like MATLAB is very useful.

Mathematical Derivation  
First: the electrical system analysis

Using Kirchhoff's voltage law which states that at any given instant of time the algebraic sum of the voltages around any loop in an electrical circuit is zero.

Figure.2. a

And according to this law we will get the equation which represent our electrical system and its Laplace transform (Assuming that all initial conditions are zero) which will be used later to get the transfer function of the whole system as following:

* (the DE of the electrical system)
* a voltage proportional to the product of the flux and angular velocity)
* To get the transfer function which represent energy transformation from one form two anther, the meaning of what we mentioned previously that we will transfer the differential equation from time domain to (s) domain for doing this we will use a popular method in the world of math which called Laplace transform
* Laplace transfer:
* To connect between the electrical and mechanical systems from 1 in 2
* Second: the mechanical system
* the torque that is applied to the inertia and friction; and we can represent this in the equation in the beside picture.
* Knowing that the torque *T* developed by the motor is proportional to the product of the armature current and the air gap flux ψ*,* which in tum is proportional to the field current, or ψwhere is a constant.
* (the DE of the Mechanical system)
* Laplace transfer:
* Considering as the input and as the output and eliminating and from 4 in 3

Getting the transfer function is in mathematical meaning is ratio between output and input of any system

And for our system the output is the angular displacement (Ɵ) and input is the applied voltage

**Modeling using MATLAB**

Using this app gives you big benefits with modeling and simulating a big and complex systems by using Simulink we can simulate our system in form of blocks as shown in figure 2. b

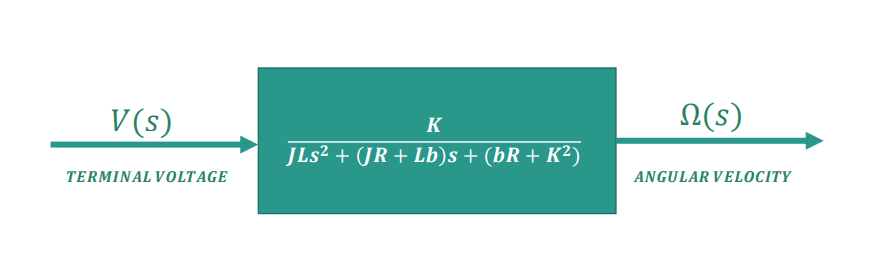
****

Figure 2.b

So we can consider this green block as our real motor that enable as interact with it take results and plotting it and this make the optimizing and analysis processes easier and faster

**Arduino Code:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   COURSE  : MTE504 MECHATRONICS II  
   PROJECT : MONITOR SPEED  
   DATE    : 10/08/2021  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*\*\*\* LIBRARIES \*\*\*\*\*/  
#**include** <Encoder.h>  
#**include** <TimerOne.h>  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*\*\*\* VARIABLES \*\*\*\*\*/  
  
// Serial Port Used  
#**define** PORT Serial  
  
// PWM value  
**int** valuePWM = 0;  
  
// Motor PWM Pin  
**byte** pinMotor = 5;  
  
// Direvction Pin  
**byte** pinDir1 = 7;  
**byte** pinDir2 = 8;  
  
// ENCODER  
**byte** pinCHA = 2;  
**byte** pinCHB = 3;  
  
// Encoder number  of pulses per revolution  
**float** PulsesPerRevolution = 380;  
  
// Calculated speed in RPM  
**volatile** **float** speedRPM = 0;  
  
// Current count of incoming encoder pulses  
**volatile** **float** PulseCount = 0.0;  
  
// TIMER  
**float** timerInterval = 0.1;  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
// Timer Interrupt Service Routine  
**void** tmrISR()  
{  
  // Disable interrupt  
  noInterrupts();  
  speedRPM = ((PulseCount / timerInterval) / PulsesPerRevolution) \* 60.0;  
  PORT.print(valuePWM);  
  PORT.print(",");  
  PORT.println(speedRPM);  
  PulseCount = 0;  
  // Enable Interrupt  
  interrupts();  
}  
  
// External Interrupt Service Routine  
**void** externalISR()  
{  
  // Increment pulse count  
  PulseCount++;  
}  
  
**void** setup()  
{  
  // Setup serial baudrate  
  PORT.begin(115200);  
  
  // Set pins as output  
  pinMode(pinMotor, OUTPUT);  
  pinMode(pinDir1, OUTPUT);  
  pinMode(pinDir2, OUTPUT);  
  
  // Default to 0  
  digitalWrite(pinMotor, LOW);  
  digitalWrite(pinDir1, LOW);  
  digitalWrite(pinDir2, LOW);  
  // Encoder PIN  
  pinMode(pinCHA, INPUT); // Set encoder pin as input to receive pulse train  
}  
  
**void** loop()  
{  
}  
  
**void** serialEvent()  
{  
  // Read command characters until \n is received  
  **auto** IncomingCommand = PORT.readStringUntil('\n');  
  **const** **auto** Command = IncomingCommand.substring(0, IncomingCommand.indexOf(','));  
  **auto** StringData = IncomingCommand.substring(IncomingCommand.indexOf(',') + 1, IncomingCommand.length());  
  PORT.flush();  
  // Parsing command  
  if (Command == "GO")  
  {  
    // Start streaming data  
    Timer1.initialize(timerInterval \* 1000000);  
    //Attach ISR  
    Timer1.attachInterrupt(tmrISR);  
  
    //Enable external interrupt (Rising Edge)  
    attachInterrupt(digitalPinToInterrupt(pinCHA), externalISR, RISING);  
  }  
  if (Command == "END")  
  {  
    // Stop Motor  
    analogWrite(pinMotor, 0.0);  
    //Detach ISR  
    Timer1.detachInterrupt();  
    detachInterrupt(digitalPinToInterrupt(pinCHA));  
  }  
  
  // New command for PWM  
  if (Command == "PWM")  
  {  
    // Send PWM value  
    valuePWM = StringData.toInt();  
    analogWrite(pinMotor, valuePWM);  
  }  
  
  // New command for direction  
  if (Command == "DIR1")  
  {  
  digitalWrite(pinDir1, HIGH);  
  digitalWrite(pinDir2, LOW);      
    }  
     if (Command == "DIR2")  
  {  
  digitalWrite(pinDir2, HIGH);  
  digitalWrite(pinDir1, LOW);      
    }  
  }

Code.2

**Simulation three:**

Hardware overview: -

1. The DC-Motor with encoder: we can integrate with them throw six terminals where two terminals are responsible for supplying the motor with voltage and ground and two terminals are for supplying the encoder with voltage and ground and the last two terminals are for phase A and phase B of the encoder where there are many types of encoders (mechanical, optical, magnetic, and electromagnetic induction types) put in this project we are using the magnetic encoder. The encoder is a sensor responsible for the feedback of the system.
2. The H-bridge(LM-298): - is the main communication system in the project where it communicates with the Arduino and give commends to the motor

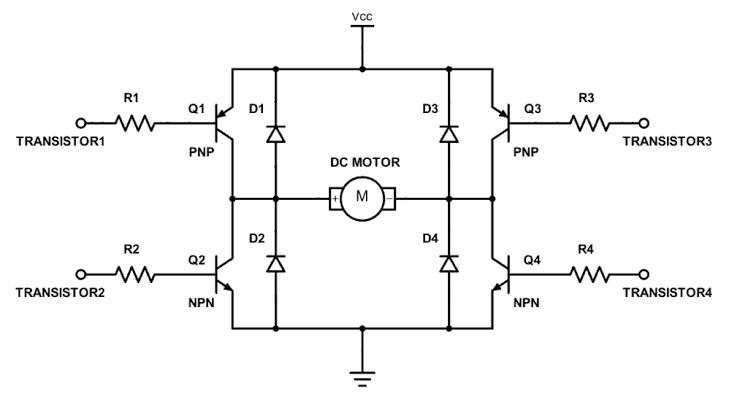


Figure 3.a

1. **Arduino Nano: -** is an Arduino kit that uses AT-MEGA chip as its microcontroller
2. **Power supply:** we used a 12-volt DC-power supply
3. **PCB board:** is used for connecting all of the mentioned components

**Full control using Arduino:**

Using Arduino ide by including a set of libraries ( timerone.h ,encoder.h)

Where timerone: is one of the libraries written to take advantage of the 16-bit counter that comes with the Atmega328.

Encoder library: allows the code to integrate with the encoder where the Encoder counts pulses from quadrature encoded signals, which are commonly available from rotary knobs, motor or shaft sensors and other position sensors.

Then we defined the (port serial) to make the code editable.

Then we initialized a set of variables that the code would use as locations representing the connections between the system to draw the pass for the signals that we would sent throw the serial monitor to study the behavior of the motor.

Then there are a set of variables which are (plus count, plus per revolution, time)which are used to calculate the speed in RPM using this equation

SpeedRPM = ((PulseCount / timerInterval) / PulsesPerRevolution) \* 60.0

And these values are always updated throw out the program.

And then we assigned the input and output pins where we deduced by the process of trial and error when we sent a high signal to one direction we must sent low to the other direction for the motor to operate

|  |  |  |
| --- | --- | --- |
| Dir-1 | Dir-2 | Motor rotation |
| 0 | 0 | Stop rotating |
| 0 | 1 | Clock-wise |
| 1 | 0 | Anticlock-wise |
| 1 | 1 | Stop rotating |

**Table 1**

A family of application-defined functions that are called whenever there is data to be read from a serial peripheral. Then we initialized a variable called command to integrate between the system and the serial plotter.

And by using switch case it allowed us to control the starting and stopping of the motor and its speed by changing the PWM and its direction.

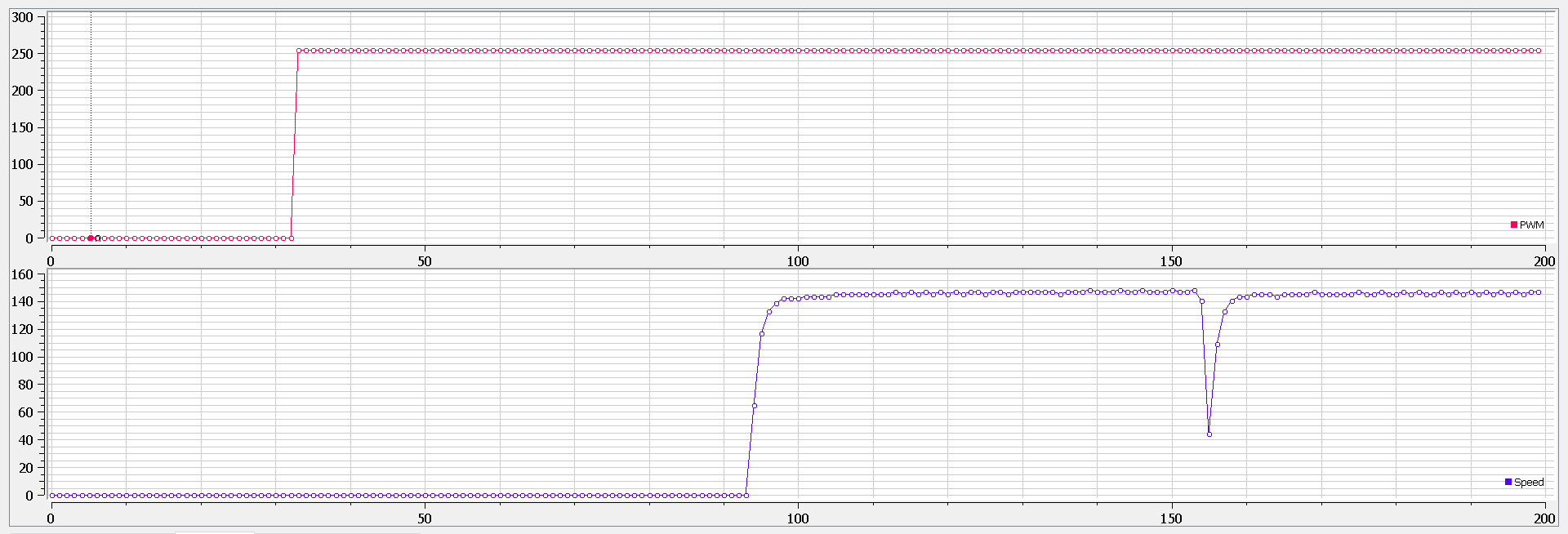
Getting measurements:

Figure 3.b

**Simulation four:**

**Explain the concept of estimating Parameter**

When you know nothing for a system Parameter you can estimate their parameters if you Know some of their features which you deduce by analyzing the system and modeling it mathematically or by using MATLAB. One of this important features is the transfer function of the system which we mention in details in simulation 2 partition and we can use anther similar system to get its knowing parameter as initial assumed parameter and by getting a set of measured data from our actual output which is a result for a set of input data then we can use parameter estimator which is an app in Simulink to estimate our real system parameters

**Prepare the logging**

for this purpose, we control our system in particular way to get this data and using the code mentioned in the previous partition with additional point to fit this way. The way we are talking about is to turn the motor on then off for equal periods of time

and the additional point for the code is to control the number of cycle and the number of samples send by the microcontroller and for this we define some variables to show (the sample count, the switching period, the number of samples, the overall sample count, the overall number of samples) and by using these variables in our code we could manage how long the on period will take and how many of this period will happen by using the following calculations

switchingPeriod = 7; (determine how long the half period time)

numOfSamples = switchingPeriod / timerInterval;

overallNumOfSamples = numOfSamples \* switchingPeriod \* 2 + numOfSamples;

And the main concept for our program is the interaction between the encoder which represents the feedback of the system and the timer peripheral and the interrupt peripheral so when we attach an interrupt to the pulse count which we get from the encoder we can use it to calculate the RPM and when we pass this RPM to function which is attached to an ISR and use the timer peripheral to control when this ISR will fire

So this way we could plot our RPM values which we consider the output set of data and this output set of data changes by the change of the input set of data which are the PWM values which are translated by the H-bridge to voltage and then transmit it to the motor. As we get this data we could now call them a logging data that we will export in the form of a csv file from serial plotter which we will use in the next step for parameters estimation. The following figures show how the serial plotter show the data.

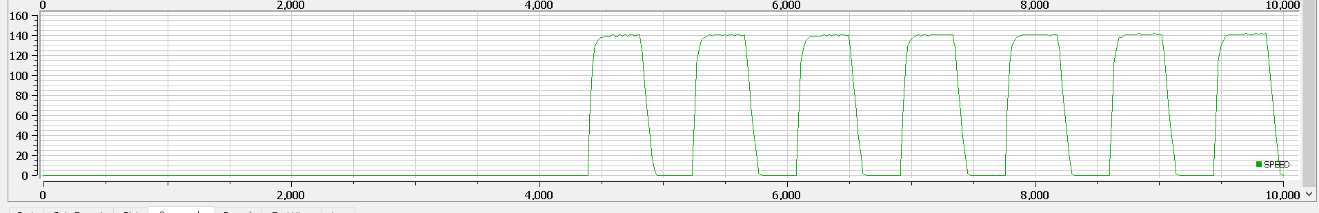


Figure 4.a(speed)

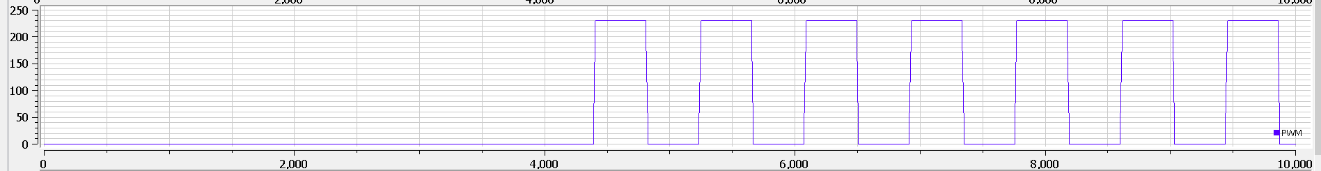


Figure 4.b(PWM)

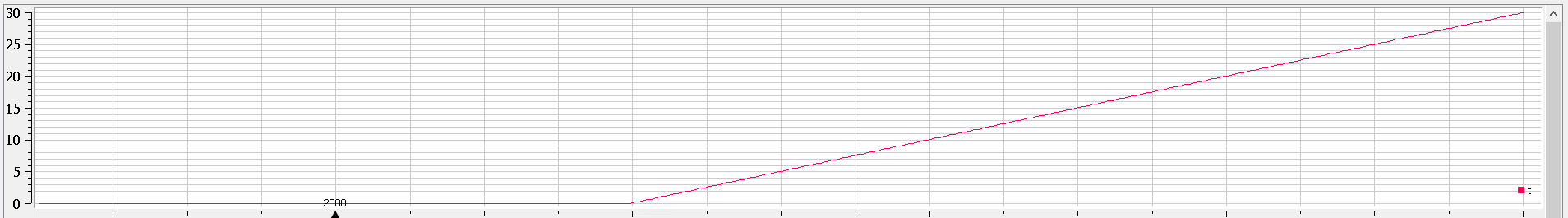


Figure 4.c(time)

**Logging data to MATLAB using this script**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%% SCRIPT : Import DC motor log file to MATLAB workspace %%  
%% BY : Waleed El-Badry %%  
%% DATE : 13/08/2021 D:\laby\teams\modeling\Simulation 4\serialplot\ %%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
   
%% Read log csv file  
file\_path='D:\laby\teams\modeling\Simulation4\serialplot\DATA\_MOTOR\_4.csv';  
motor\_log = readtable(file\_path);  
   
%% Store each column in a variable  
t = motor\_log.t;  
speed = motor\_log.SPEED;  
pwm = motor\_log.PWM;  
   
%% export to mat file  
save('motor\_log','t','pwm','speed');  
R= 2.0 % Ohms  
L= 0.5 % Henrys  
K = .015 % torque constant   
B = 0.2 % Nms  
J= 0.02 % kg.m^2

Using this lines, we imported the dc motor log file to the MATLAB by writing the pass for the csv file that contains the measured data that we took from the motor and also adding the assumed parameters (R, K, L, B, J).

Getting the parameters

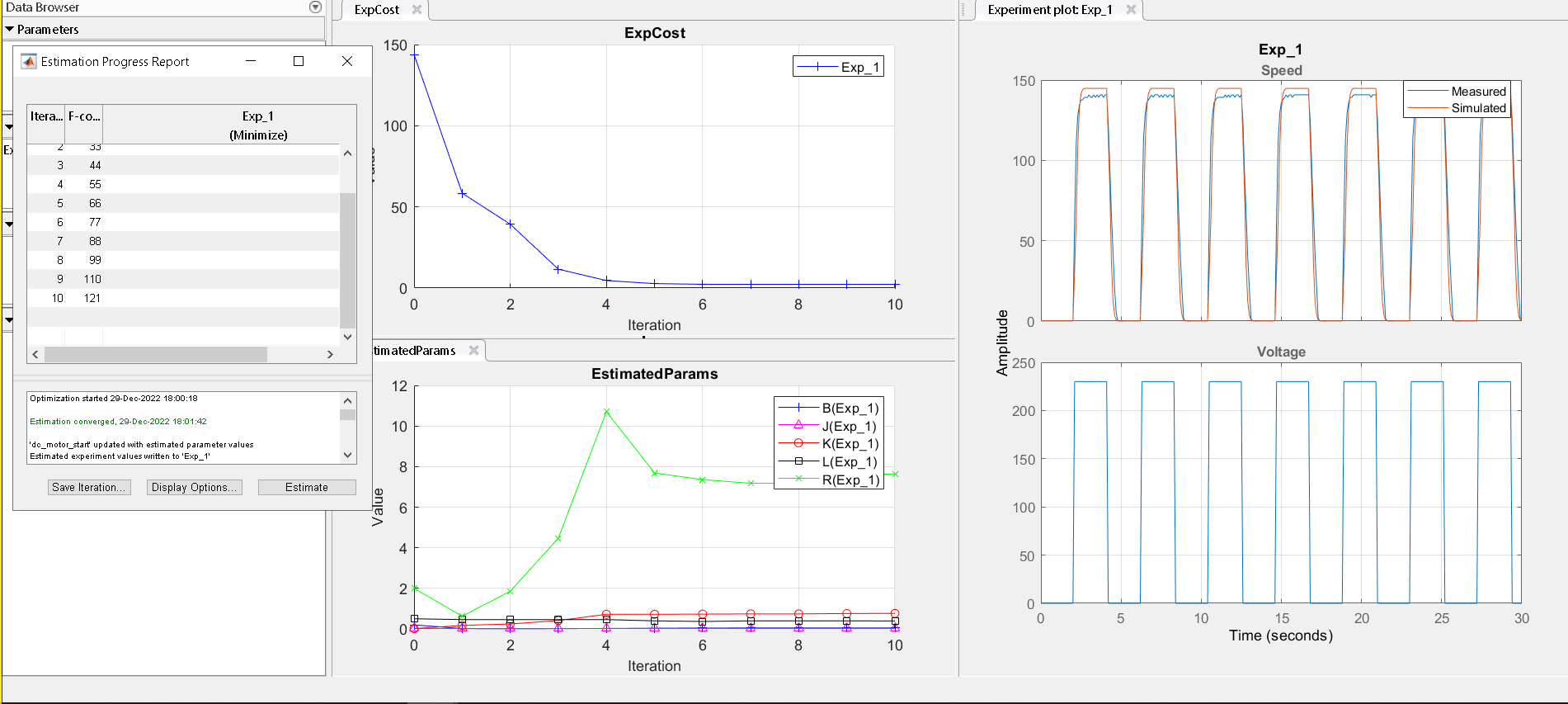
Getting the parameters throw the process of tuning where the matlab starts to tune the assumed data to reach as close as possible to the measured data as shown in the figure below.

Figure 4.d

**Arduino Code:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   COURSE  : MTE504 MECHATRONICS II  
   PROJECT : MONITOR SPEED  
   DATE    : 10/08/2021  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*\*\*\* LIBRARIES \*\*\*\*\*/  
#**include** <Encoder.h>  
#**include** <TimerOne.h>  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*\*\*\* VARIABLES \*\*\*\*\*/  
  
// Serial Port Used  
#**define** PORT Serial  
  
// PWM value  
**int** valuePWM = 0;  
  
// Motor PWM Pin  
**byte** pinMotor = 5;  
  
// Direvction Pin  
**byte** pinDir1 = 7;  
**byte** pinDir2 = 8;  
  
// ENCODER  
**byte** pinCHA = 2;  
**byte** pinCHB = 3;  
  
// Encoder number  of pulses per revolution  
**float** PulsesPerRevolution =375 ;  //actual ppr = ppr\*reduction ratio   visit : http://xytmotors.com/encoder/153.html  for motor data sheet4\*12.3  
  
// Calculated speed in RPM  
**volatile** **float** speedRPM = 0;  
  
// Current count of incoming encoder pulses  
**volatile** **float** PulseCount = 0.0;  
  
// TIMER  
**float** timerInterval = 0.1;  
  
// Logging  
**float** sampleCount = 0.0;  
**float** switchingPeriod =2 ;  
**float** numOfSamples = switchingPeriod / timerInterval;  
**float** overallSampleCount = 0;  
**float** overallNumOfSamples = numOfSamples \* 7 \* 2 + numOfSamples;  
**float** t = 0.0;  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
// Timer Interrupt Service Routine  
**void** tmrISR()  
{  
  // Disable interrupt  
  noInterrupts();  
  
  speedRPM = ((PulseCount / timerInterval) / PulsesPerRevolution) \* 60.0;  
  PORT.print(t);  
  PORT.print(",");  
  PORT.print(valuePWM);  
  PORT.print(",");  
  PORT.println(speedRPM);  
  PulseCount = 0;  
    
  // For logging  
  if (sampleCount == numOfSamples)  
  {  
    if (valuePWM > 0)  
    {  
      valuePWM = 0;  
      analogWrite(pinMotor, valuePWM);  
   digitalWrite(pinDir1,LOW);  
   digitalWrite(pinDir2, LOW);   
    }  
    else  
    {  
      valuePWM = 230;  
      analogWrite(pinMotor, valuePWM);  
   digitalWrite(pinDir1, HIGH);  
   digitalWrite(pinDir2, LOW);        
    }  
    sampleCount = 0;  
  }  
  else  
    // Increment pulse count  
    sampleCount++;  
//for stop after overall sampele count   
  if (overallSampleCount >= overallNumOfSamples)  
  {  
    // End connection  
    // Stop Motor  
    analogWrite(pinMotor, 0.0);  
    sampleCount = 0;  
    PulseCount = 0;  
    overallSampleCount = 0;  
    t = 0;  
      
    //Detach ISR  
    Timer1.detachInterrupt();  
    detachInterrupt(digitalPinToInterrupt(pinCHA));  
  }  
  else  
    overallSampleCount++;  
  
  // Increment timestamp  
  t+=timerInterval ;  
  
  // Enable Interrupt  
  interrupts();  
}  
  
// External Interrupt Service Routine  
**void** externalISR()  
{  
  PulseCount++;  
}  
  
**void** setup()  
{  
  // Setup serial baudrate  
  PORT.begin(115200);  
  
  // Set pins as output  
  pinMode(pinMotor, OUTPUT);  
  pinMode(pinDir1, OUTPUT);  
 pinMode(pinDir2, OUTPUT);  
  // Default to 0  
  digitalWrite(pinMotor, LOW);  
  digitalWrite(pinDir1, LOW);  
   digitalWrite(pinDir2, LOW);  
  
  // Encoder PIN  
  pinMode(pinCHA, INPUT); // Set encoder pin as input to receive pulse train  
}  
  
**void** loop()  
{  
}  
  
**void** serialEvent()  
{  
  // Read command characters until \n is received  
  **auto** IncomingCommand = PORT.readStringUntil('\n');  
  **const** **auto** Command = IncomingCommand.substring(0, IncomingCommand.indexOf(','));  
  **auto** StringData = IncomingCommand.substring(IncomingCommand.indexOf(',') + 1, IncomingCommand.length());  
  PORT.flush();  
  // Parsing command  
  if (Command == "GO")  
  {  
    // Start streaming data  
    Timer1.initialize(timerInterval \* 1000000);  
    //Attach ISR  
    Timer1.attachInterrupt(tmrISR);  
  
    //Enable external interrupt (Rising Edge)  
    attachInterrupt(digitalPinToInterrupt(pinCHA), externalISR, RISING);  
  }  
  if (Command == "END")  
  {  
    // Stop Motor  
    analogWrite(pinMotor, 0.0);  
    //Detach ISR  
    Timer1.detachInterrupt();  
    detachInterrupt(digitalPinToInterrupt(pinCHA));  
  }  
  
  // New command for PWM  
  if (Command == "PWM")  
  {  
    // Send PWM value  
    valuePWM = StringData.toInt();  
    analogWrite(pinMotor, valuePWM);  
  }  
  
  // New command for direction  
  if (Command == "DIR1")  
  {  
     digitalWrite(pinDir1, HIGH);  
   digitalWrite(pinDir2, LOW);  
  
  }  
if (Command == "DIR2")  
  {  
     digitalWrite(pinDir2, HIGH);  
   digitalWrite(pinDir1, LOW);  
 }  
  if (Command == "GOLOG")  
  {  
    // Start streaming data  
    Timer1.initialize(timerInterval \* 1000000);  
    //Attach ISR  
    Timer1.attachInterrupt(tmrISR);  
  
    //Enable external interrupt (Rising Edge)  
    attachInterrupt(digitalPinToInterrupt(pinCHA), externalISR, RISING);  
  
    // Set PWM to 0  
    analogWrite(pinMotor, 0);  
  }  
}