ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

FEEDBACK CONTROL OF ROBOTIC ARM EDGE OWI-535 WITH ARDUINO AND MATLAB

Contents

Introduction

This project aims to control the Robotic arm through Arduino & Matlab, replacing the manual control.
Specifically, this Robotic arm consists of 5 degrees of freedom (d.o.f), Base, 3 Joints and the Grip.
Briefly, we will connect the base and the joints in an Arduino uno board in which a motor-shield is attached, and make it move through Arduino & Matlab program.
We shall guide you step by step how to make the wiring, soldering, the programs and of course how it looks as a final product.

Construction

The materials we need:

1) Arduino Uno

2) Motor-Shield (OEM Motor Shield L293D)

3) Potentiometers (and some paper-clips)

4) Jumpers for the wiring

6) Robotic Arm(ROBOTIC ARM EDGE(OWI-535))

Connection of Arduino Uno Board with the Robotic Arm

Before we proceed, let's take a look at main parts which we'll make the connections.

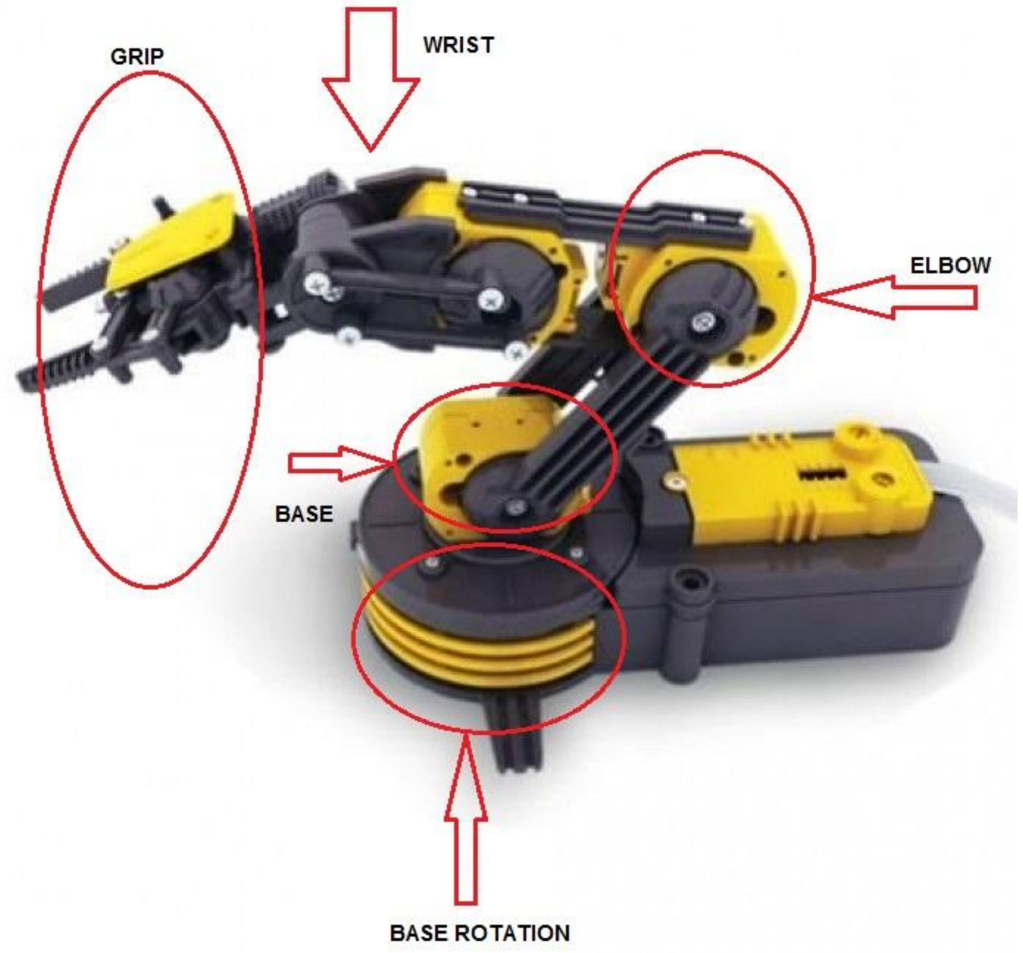


Image 6, Robotic Arm of 5 d.o.f.

The potentiometers will be placed in the wrist, elbow, base and base rotation of the arm. It will help us read their values in order to make it move. Unfortunately, the grip will stay idle due to a restriction of motor-shield which can move only 4 motors.

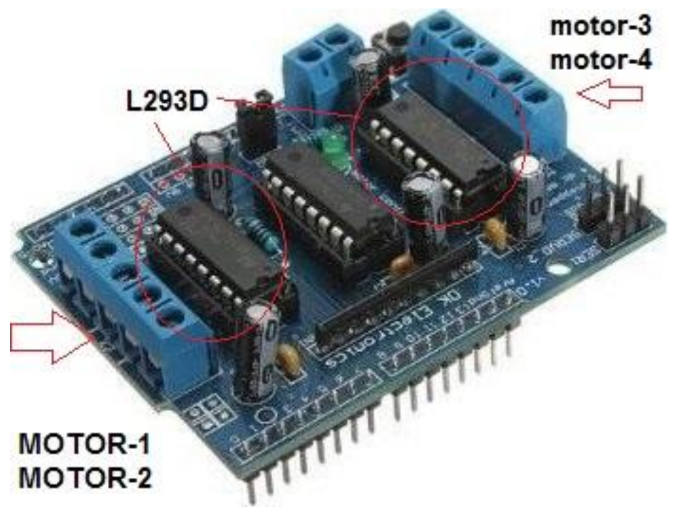


Image 7, Motor-Shield

In order to place the potentiometers in the joints we make a small preparation.

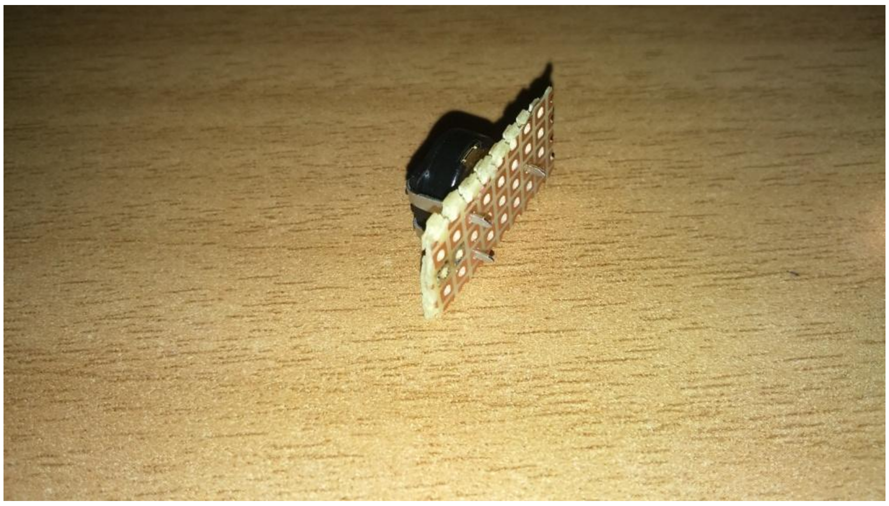
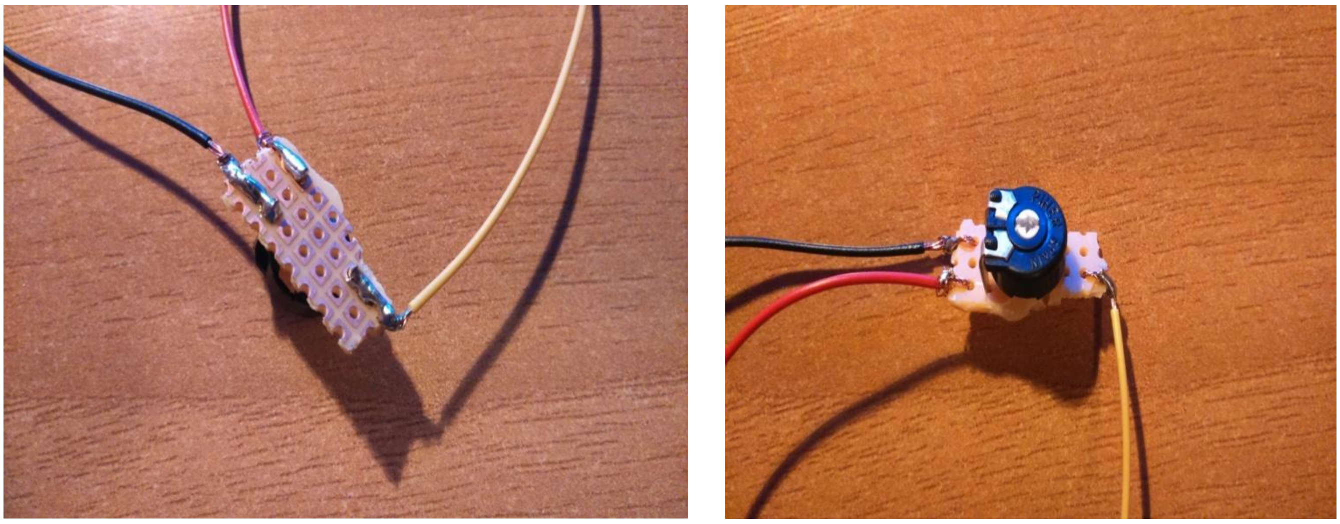


Image 8, Proto-Board with Potentiometer

Firstly, we take a proto-board, cut it with a cutter and put the potentiometers through it's holes, as shown in Image 8. Secondly, we solder 3 wires in each potentiometer edge. (See Images 9,10)



Images 9,10, Potentiometer Wiring

Now, we place the potentiometers in the centre of each Joint. We stabilize them with hot glue.

Subsequently, we bent and cut the clips as the image below. In addition, in order to read our values from potentiometers precisely, the angles that are created need to be 90°.
Finally, we stabilize the clips with hot glue(See Images 11 - 14).

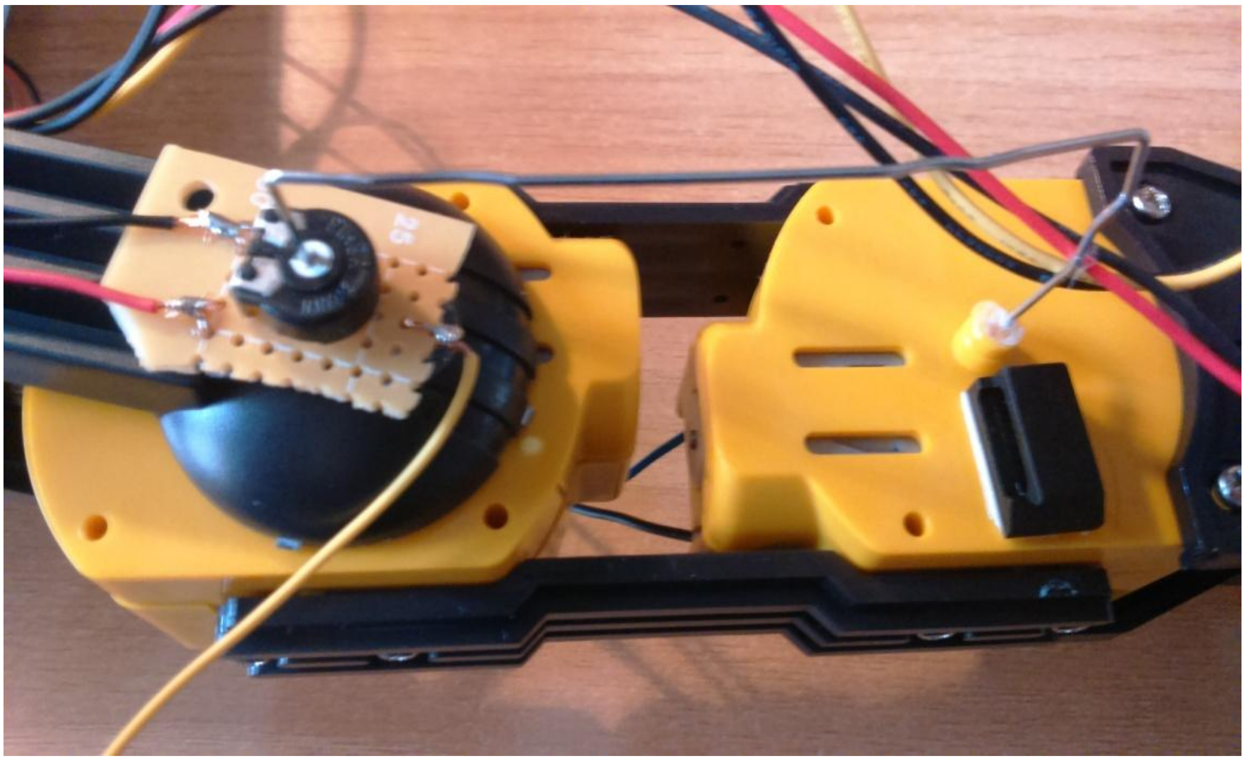


Image 11.Elbow Joint with the potentiometer

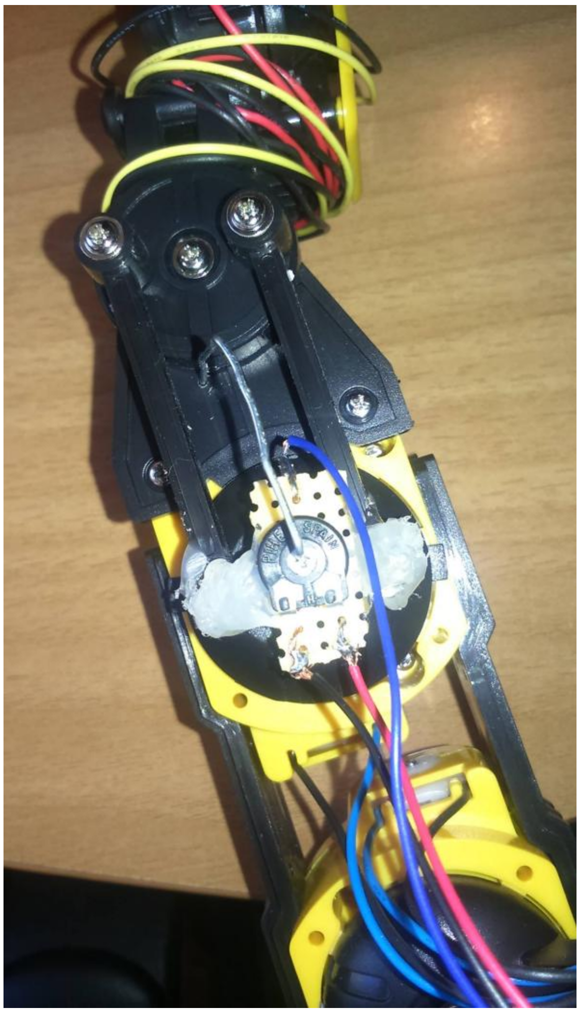


Image 12.Wrist Joint with potentiometer

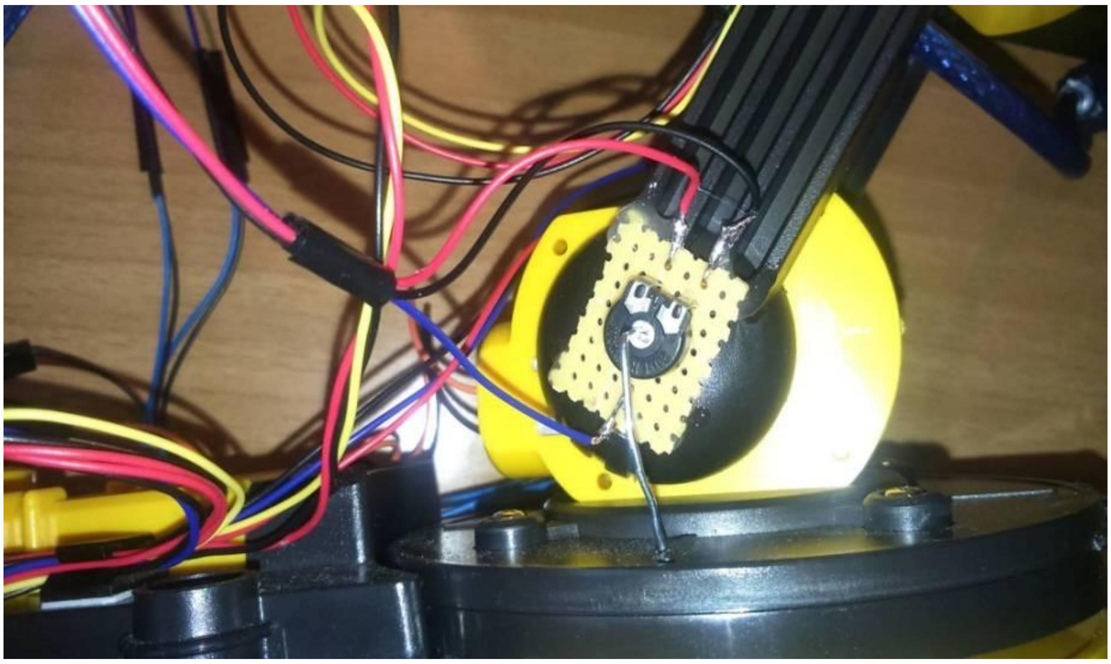


Image13.Base with potentiometer

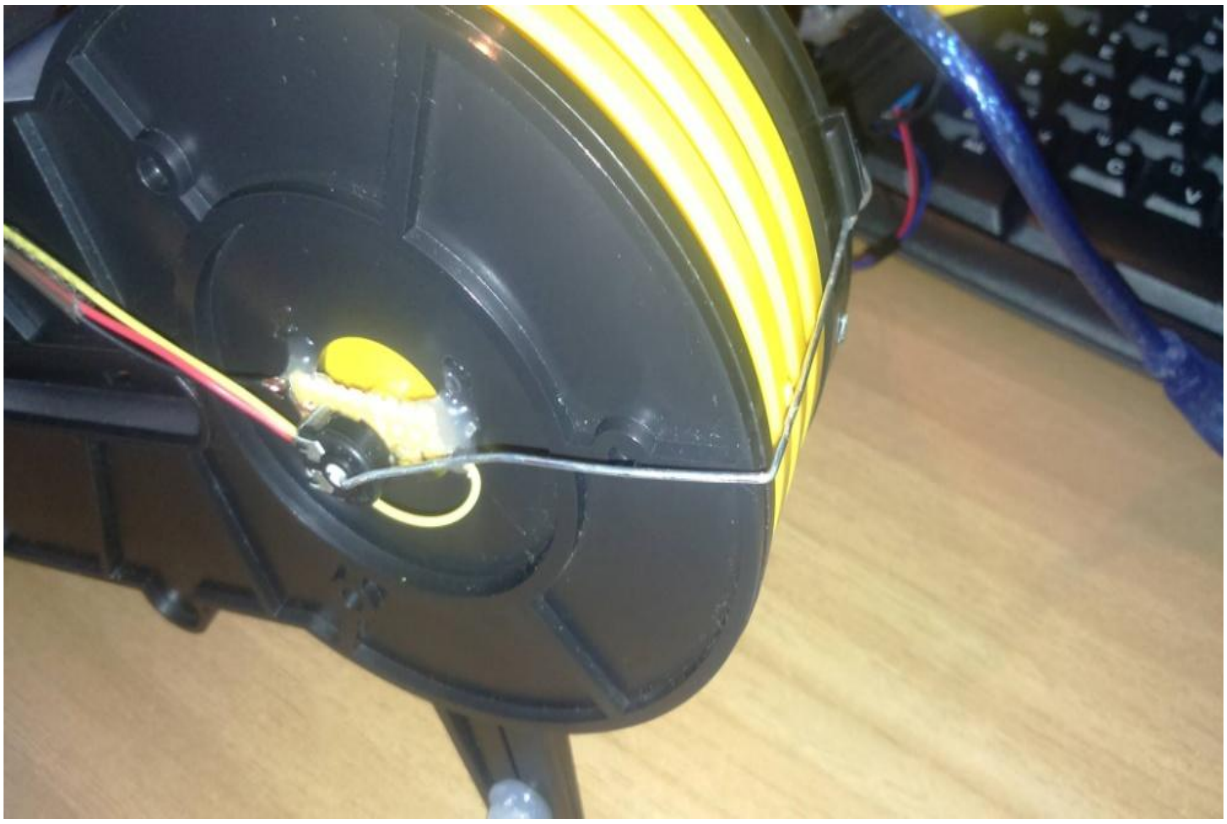


Image 14.Base rotation with potentiometer

The reason we used potentiometers, it's because we need a closed-loop feedback system in order to read through Arduino, the values of angles created everytime it moves.

Now we'll go through the steps for the wiring between the Robotic Arm and Arduino Uno:

Potentiometers:

Positive end to 5V
Negative end to GND
The output of every potentiometer P1-P4 to analog of the Arduino board A1-A4 respectively

The motor's cables (of robotic arm) to motor-shield blue terminals M1-M4 respectively.

The two blue middle terminals remaining to GND.

Of course don't forget the usb connection for Arduino board and a DC power supply(5-20V) for motor-shield.

Below we see a sample showcase.
(All are correct besides the potentiometers which are on the robotic arm and couldn't show it properly here.)

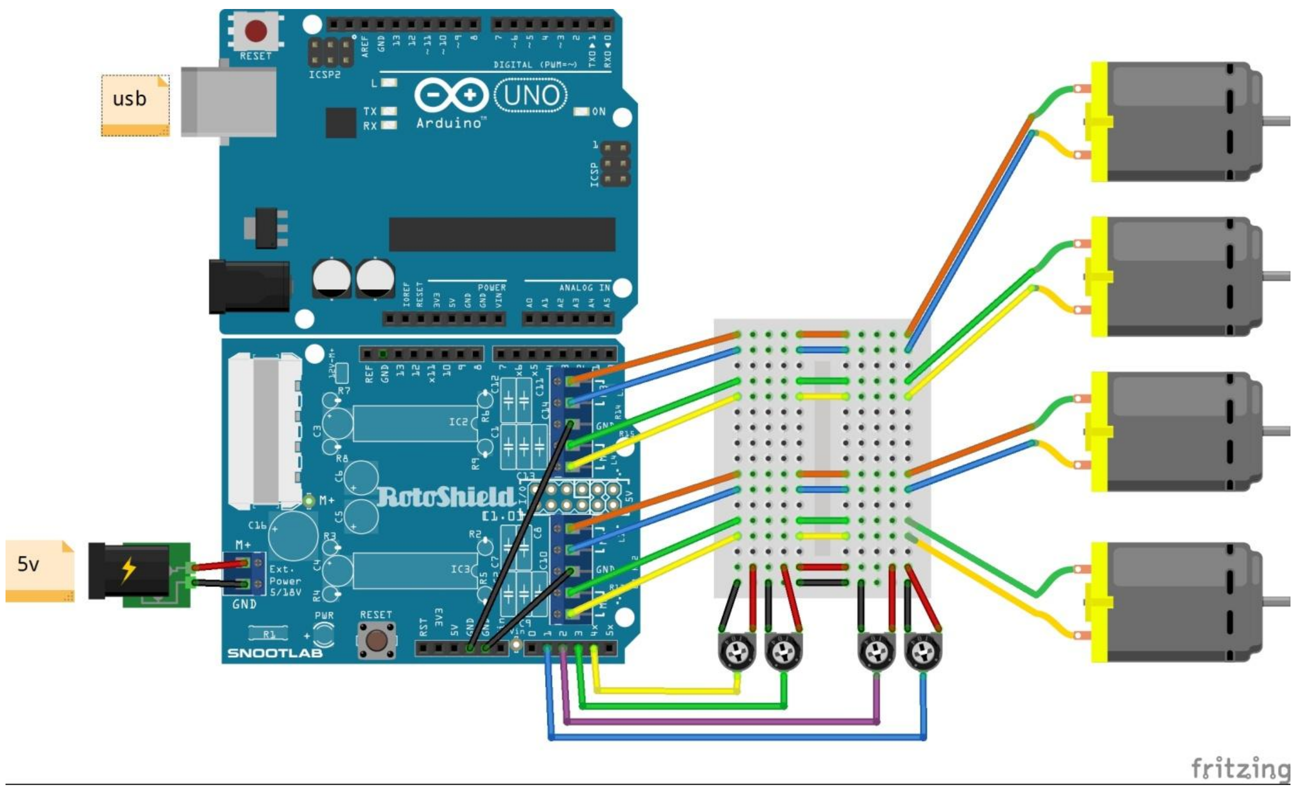


Image 15, Wiring

In the images below we take a close look at our example from different perspectives.

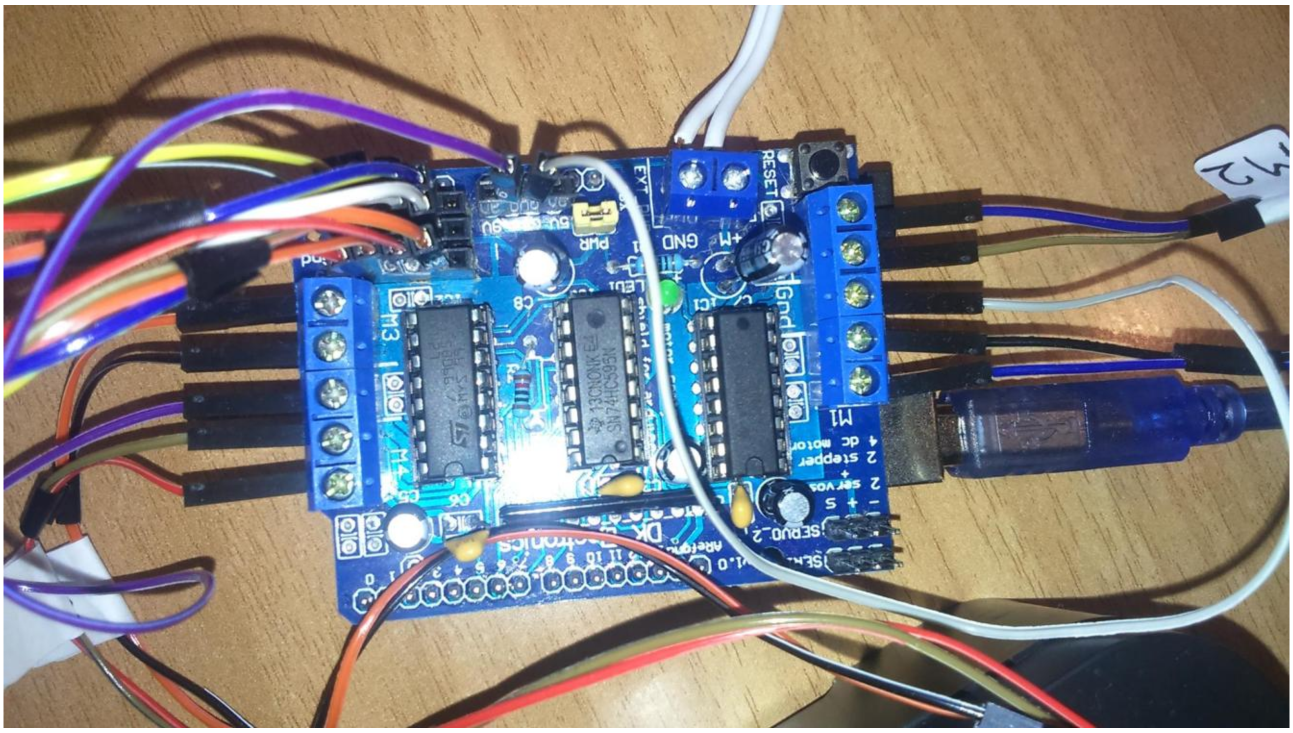


Image16, Top View of Arduino Board with the Motor-Shield

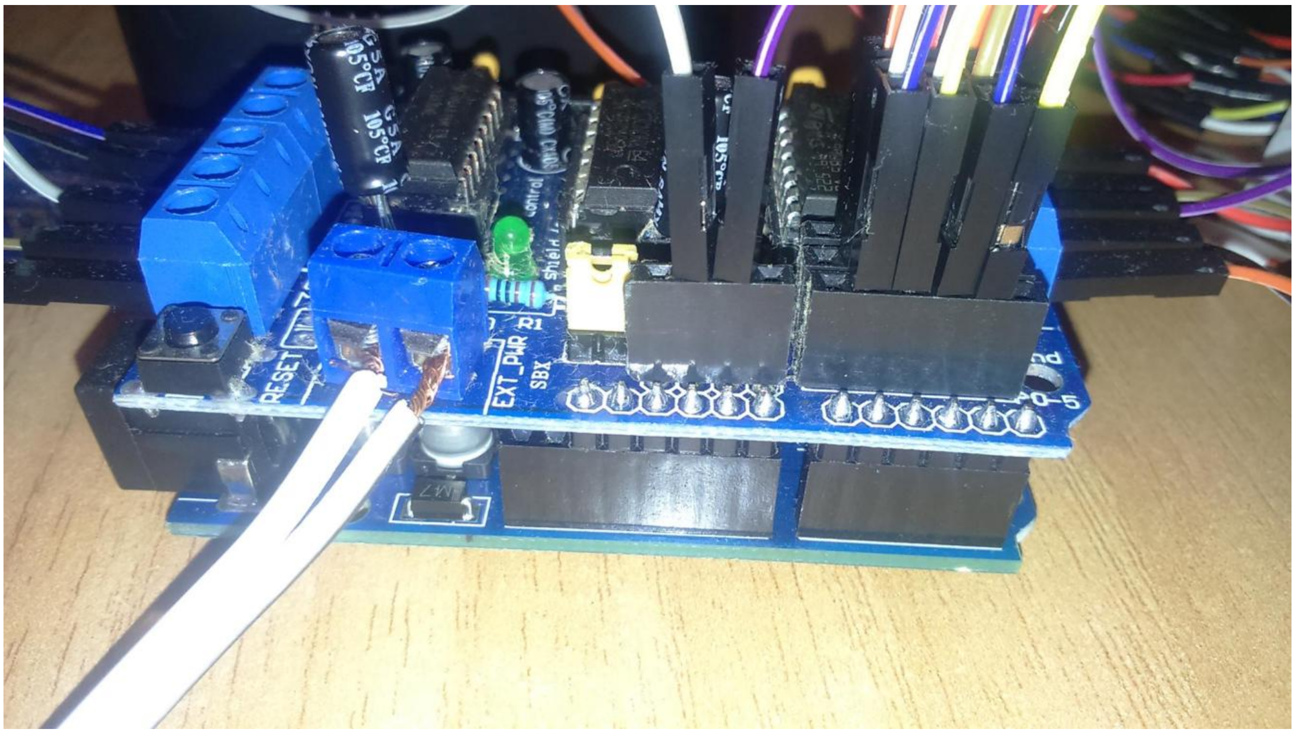


Image 17, Side View of Arduino Board with the Motor-Shield

Software

The programs we'll use:

1) Arduino IDE

2) Matlab

We can download them from their official sites.

https://www.arduino.cc/en/Main/Software

https://www.mathworks.com/academia/student_version.html

Once the installation is complete it's time to download some necessary libraries.

1) Arduino Library for controlling the Motors through Motor-Shield:

   https://github.com/adafruit/Adafruit-Motor-Shield-library

2) Matlab Library to connect Matlab with Arduino

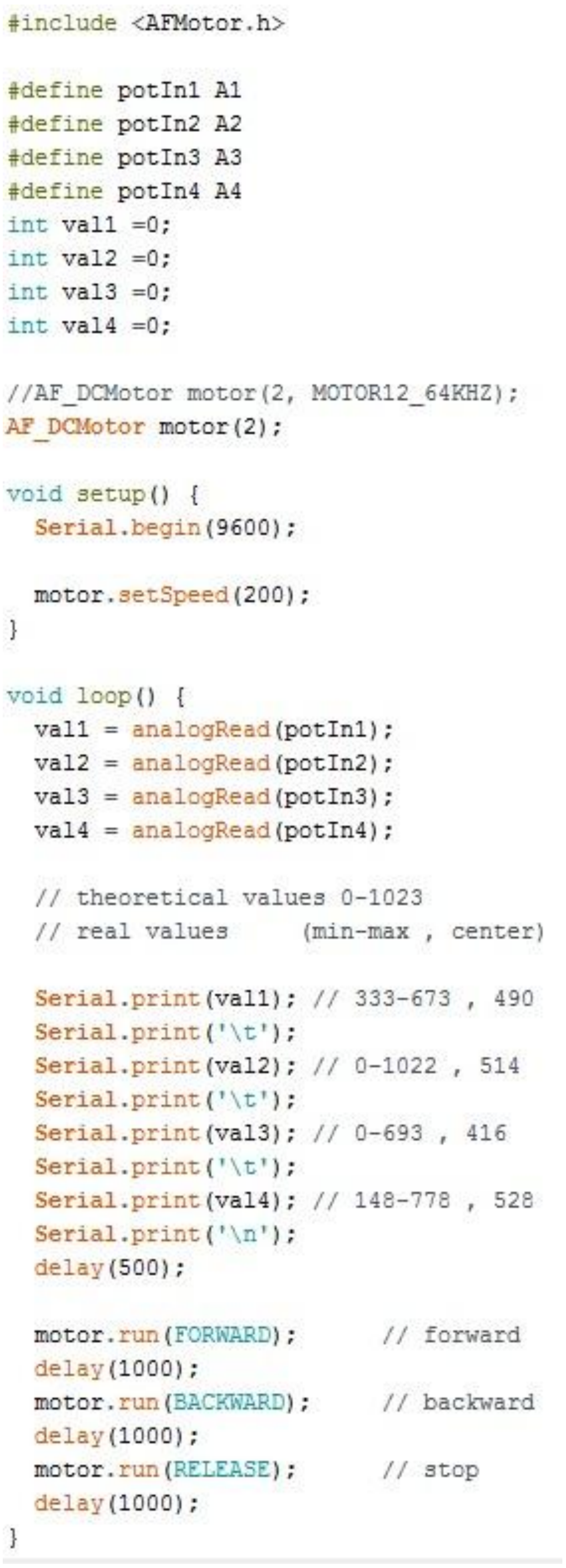
   http://www.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino

In order to “install” the Arduino library we unzip the folder we downloaded and rename it as AFMotor. Then we move our AFMotor folder to libraries folder of Arduino (example: C:\Program Files\Arduino\libraries).

Now for the Matlab, we open the Matlab, and we click the button **Add-Ons**, from the drop down menu we choose: **Get Hardware Support Packages** and search for **32374-legacy-matlab-and-simulink-support-for-arduino**.
Once we find it we click **add**. (A mathrowks account is needed in order to download the library!)
Afterwards, from the Matlab interface we choose the: **Set Path**, in the pop-up which opens we choose: **Add with Subfolders** and we enter the path which contains our library (example: C:\Users\mypc\Documents\MATLAB).

Arduino Code

Now we are ready to code! Open the Arduino IDE and write this example below!



```
#include <AFMotor.h>

#define potIn1 A1
#define potIn2 A2
#define potIn3 A3
#define potIn4 A4
int val1 =0;
int val2 =0;
int val3 =0;
int val4 =0;

//AF_DCMotor motor(2, MOTOR12_64KHZ);
AF_DCMotor motor(2);

void setup() {
  Serial.begin(9600);

  motor.setSpeed(200);
}

void loop() {
  val1 = analogRead(potIn1);
  val2 = analogRead(potIn2);
  val3 = analogRead(potIn3);
  val4 = analogRead(potIn4);

  // theoretical values 0-1023
  // real values     (min-max , center)

  Serial.print(val1); // 333-673 , 490
  Serial.print('\t');
  Serial.print(val2); // 0-1022 , 514
  Serial.print('\t');
  Serial.print(val3); // 0-693 , 416
  Serial.print('\t');
  Serial.print(val4); // 148-778 , 528
  Serial.print('\n');
  delay(500);

  motor.run(FORWARD);      // forward
  delay(1000);
  motor.run(BACKWARD);     // backward
  delay(1000);
  motor.run(RELEASE);      // stop
  delay(1000);
}
```

Image 18, Arduino Program

In this program we read our values from potentiometers and move a joint accordingly.

A brief description of what happens in the code:

Firstly, we include the library (AFMotor, we downloaded it earlier in this guide):
#include <AFMotor.h>

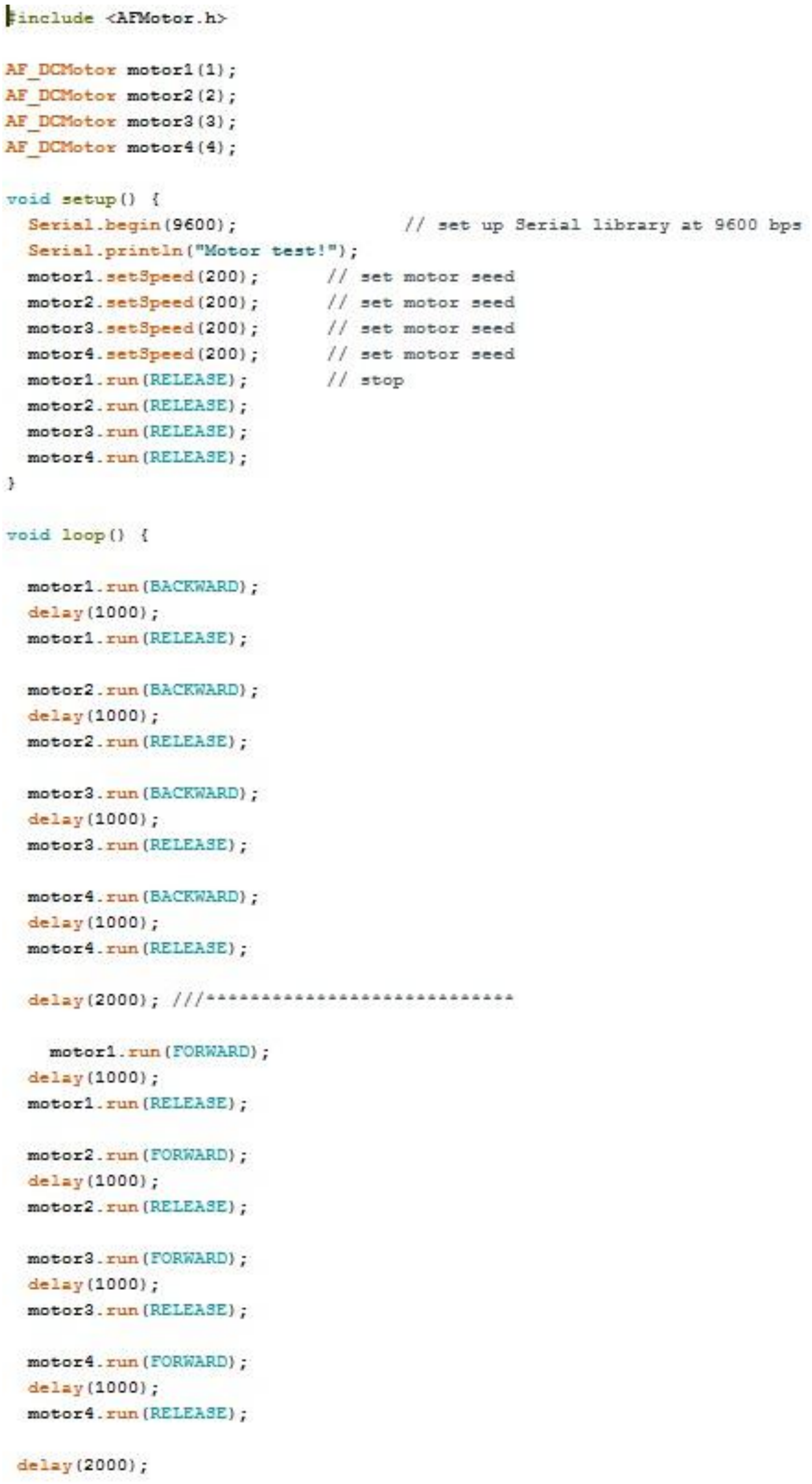
Now we define (for the microprocessor) the variables potIn1,2,3,4 which corresponds to analog inputs of Arduino board and declare with AF_DCMotor() the motor we would like to operate.

In setup() we enable the serial screen and setup the rotation speed(0-250) via PWM.

In loop() we read these analog values (which comes from the potentiometers), assign them to val1,2,3,4 and print them in the serial screen.

Now we use the methods from our library (motor.run() with arguments such as FOWARD,BACKWARD,RELEASE) in order for our desired motor to move.

Another program is the OWI_motor_test which moves all the joints in order to check that the whole Robotic Arm functions as expected.



```
#include <AFMotor.h>

AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

void setup() {
  Serial.begin(9600);               // set up Serial library at 9600 bps
  Serial.println("Motor test!");
  motor1.setSpeed(200);      // set motor seed
  motor2.setSpeed(200);      // set motor seed
  motor3.setSpeed(200);      // set motor seed
  motor4.setSpeed(200);      // set motor seed
  motor1.run(RELEASE);       // stop
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

void loop() {

  motor1.run(BACKWARD);
  delay(1000);
  motor1.run(RELEASE);

  motor2.run(BACKWARD);
  delay(1000);
  motor2.run(RELEASE);

  motor3.run(BACKWARD);
  delay(1000);
  motor3.run(RELEASE);

  motor4.run(BACKWARD);
  delay(1000);
  motor4.run(RELEASE);

  delay(2000); ///****************************

    motor1.run(FORWARD);
  delay(1000);
  motor1.run(RELEASE);

  motor2.run(FORWARD);
  delay(1000);
  motor2.run(RELEASE);

  motor3.run(FORWARD);
  delay(1000);
  motor3.run(RELEASE);

  motor4.run(FORWARD);
  delay(1000);
  motor4.run(RELEASE);

 delay(2000);
```

Image 19, Arduino Program OWI_motor_test

MATLAB CODE

For matlab we'll need some extra steps in order to run our program.

After we installed the library (mentioned earlier, see page 12) we open Arduino IDE and upload the file **motor_v1.pde** which can be found in the folder pde contained in the MATLAB directory.

Now we close the Arduino IDE (we don't need it) and open Matlab...we are ready to code!

We add the necessary command **a=arduino('port')** where 'port' is the COM port in which our arduino board is connected(example 'COM3').

In addition, the command a.delete stops the connection.

In case the connection doesn't stop we use the command:
Delete(instrfind(‘Type’,’serial’));

It's crucial that we turn off the connection otherwise we can’t execute another program we might want.

Basic commands we use:

- motorSpeed(a,2,200); to setup the speed (object,motor,pmw)
- motorRun(a,2,forward);
- motorRun(a,2,backward);
- motorRun(a,2,releasee);

Now that the connection is done, we'll make a simple GUI which will move a joint (forward,backward and stops it).
 **F=forward, B=backward, S=stop

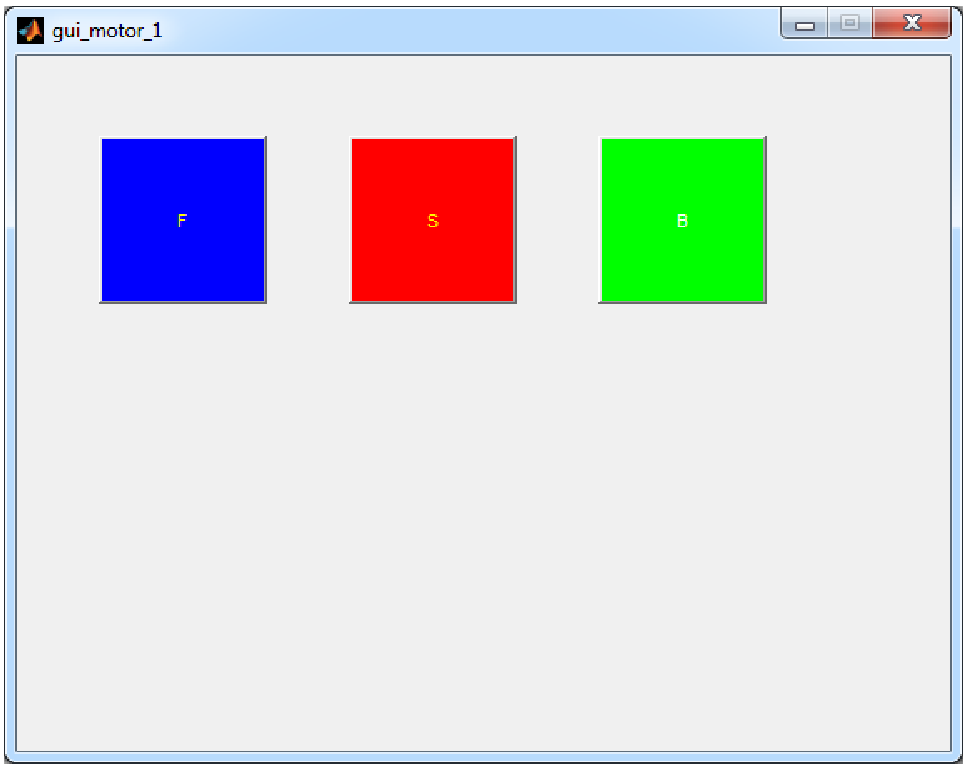


Image 20, Matlab GUI with the buttons

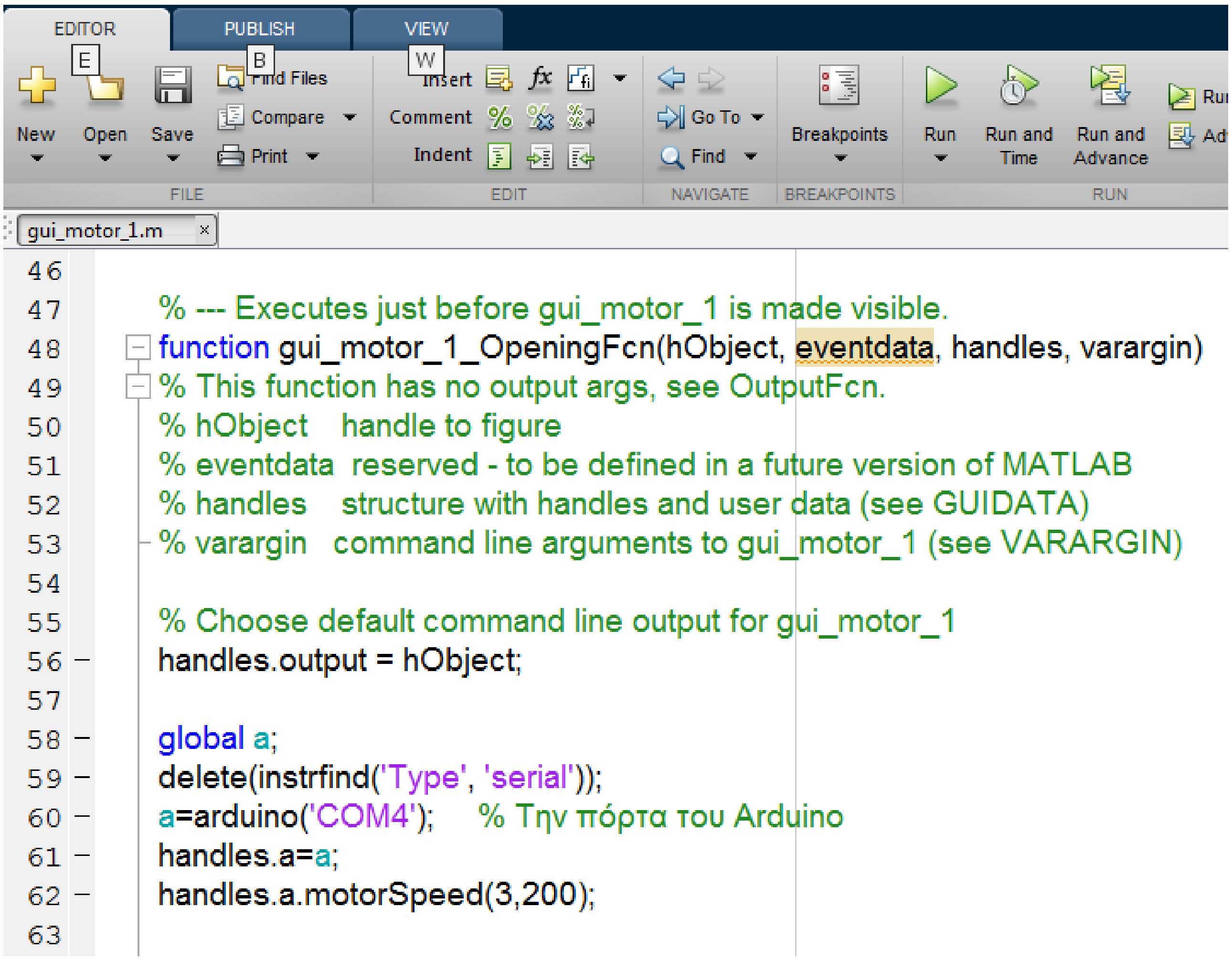
Creating the GUI:
We choose the tab: **New** -> **Graphical User Interface** -> **Blank GUI**

We click **Push Button** and create three buttons.

Double-click in the button and a settings window comes up in which we can set text, colour etc. In the tag field we enter: button_F, button_S, button_B.

As long as we save it as gui_motor_1.fig, an m-file opens (gui_motor_1.m), which is the file that executes our commands if we click in the GUI. With every button we have a callback function is created

Now in the m-file(gui_motor_1.m) we search for our functions(button_F_Callback, button_S_Callback, button_B_Callback) and write the commands shown.

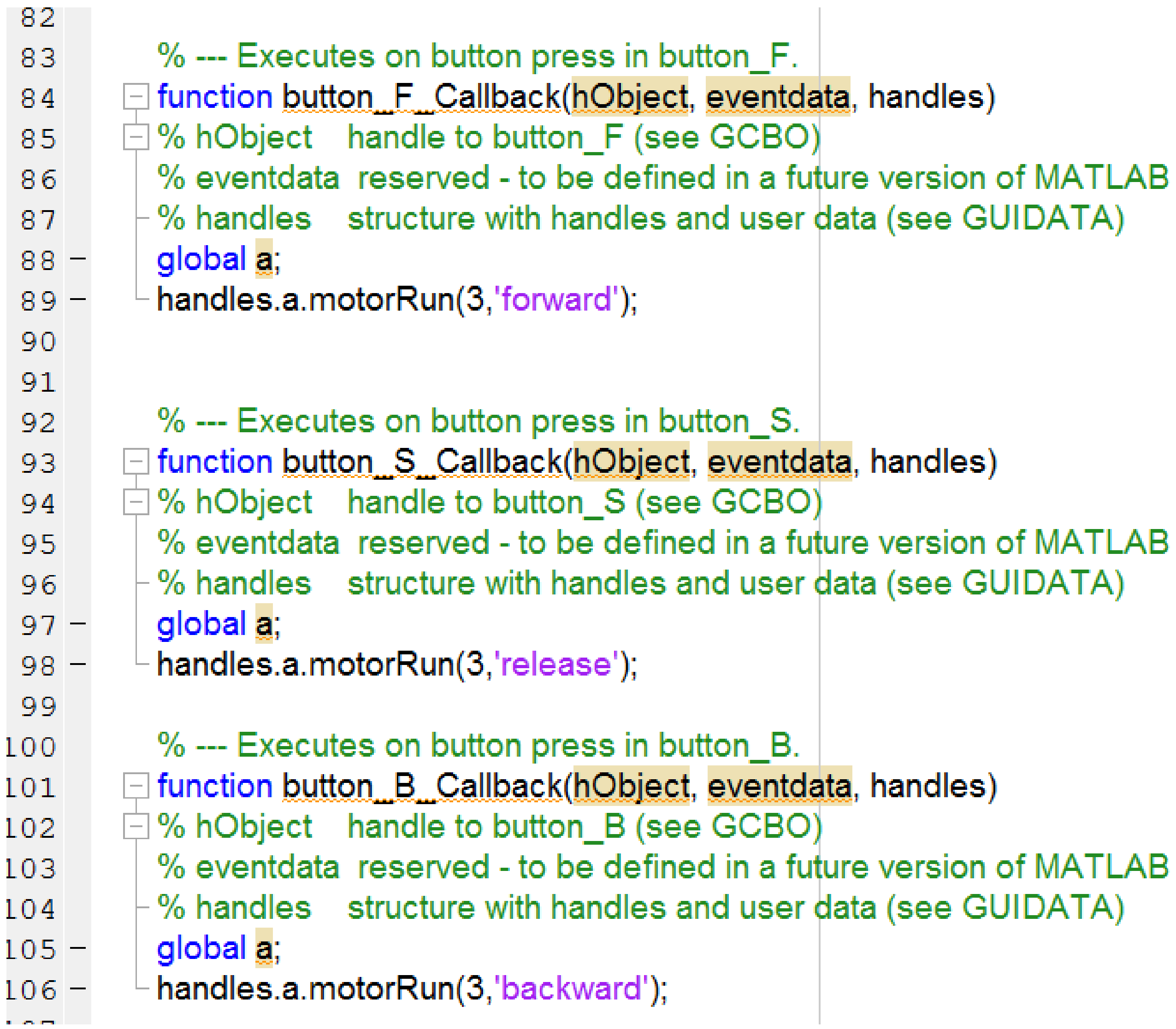


```

% --- Executes just before gui_motor_1 is made visible.
function gui_motor_1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_motor_1 (see VARARGIN)

% Choose default command line output for gui_motor_1
handles.output = hObject;

global a;
delete(instrfind('Type', 'serial'));
a=arduino('COM4');     % Την πόρτα του Arduino
handles.a=a;
handles.a.motorSpeed(3,200);

```

Image 21, Matlab generated Code

Specifically, in function button_F_Callback we assign the forward move.
In function button_S_Callback we assign the stop.
In function button_B_Callback we assign the backward move.

As you see all the functions refer to the 3rd Motor.



```
% --- Executes on button press in button_F.
function button_F_Callback(hObject, eventdata, handles)
% hObject    handle to button_F (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a;
handles.a.motorRun(3,'forward');


% --- Executes on button press in button_S.
function button_S_Callback(hObject, eventdata, handles)
% hObject    handle to button_S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a;
handles.a.motorRun(3,'release');

% --- Executes on button press in button_B.
function button_B_Callback(hObject, eventdata, handles)
% hObject    handle to button_B (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a;
handles.a.motorRun(3,'backward');
```

Image 22, Matlab Functions

BIBLIOGRAPHY

https://github.com/adafruit/Adafruit-Motor-Shield-library

http://www.instructables.com/id/Intro-and-what-youll-need/

https://github.com/KevinGrandon/robots/tree/master/90-owi-robot-arm-hacked

http://www.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino

https://www.youtube.com/watch?v=Lps9gAPHFY8

https://www.youtube.com/watch?v=m8Pq-4ecAwQ