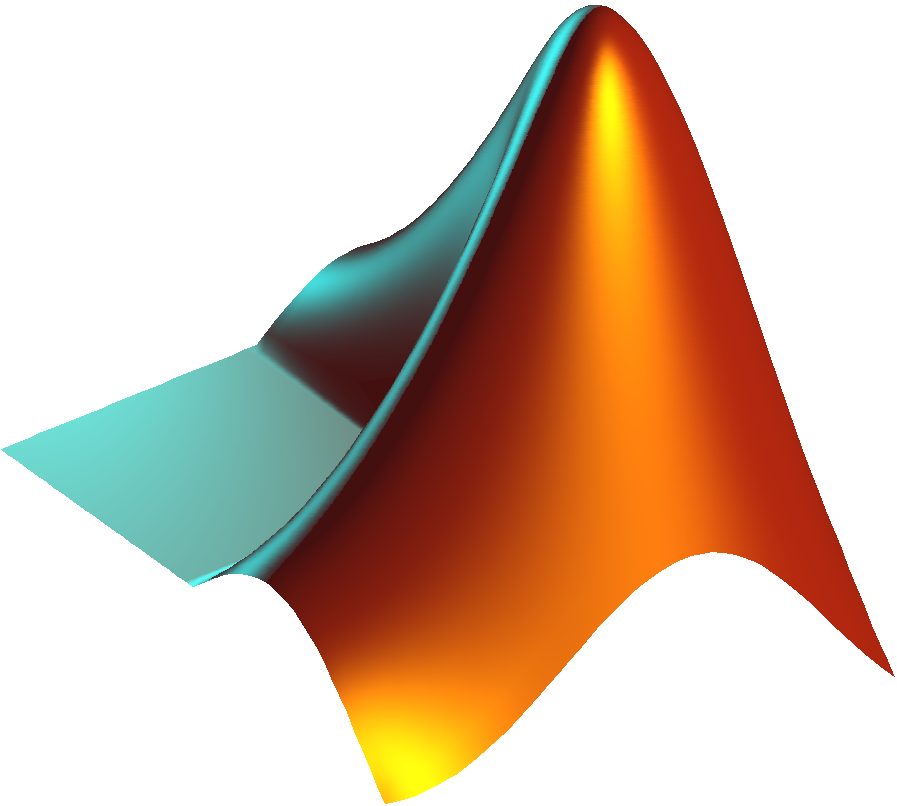
EE407 - COMPUTER SOFTWARE APPLICATIONS

TERM PROJECT

ELEVATOR DESING WITH STEPPER MOTOR

AND CONTROLLING USING MATLAB GUI

KERİM TURAK 170502110111



# INTRODUCTION

The aim of this project is to design an elevator system that works with a stepper motor and to provide floor access control of this elevator with matlab GUI. In addition, it is the examination of this height change in a time / displacement graph with the distance sensor when passing between floors. While the project aims to consolidate our matlab knowledge, it also aims to learn stepper motor control with arduino or any controller.

# PROJECT STEPS

1. Preparation of the elevator mechanism
2. Testing stepper motor controls with arduino
3. Matlab GUI design and matlab serial connection and investigation of communication methods with arduino

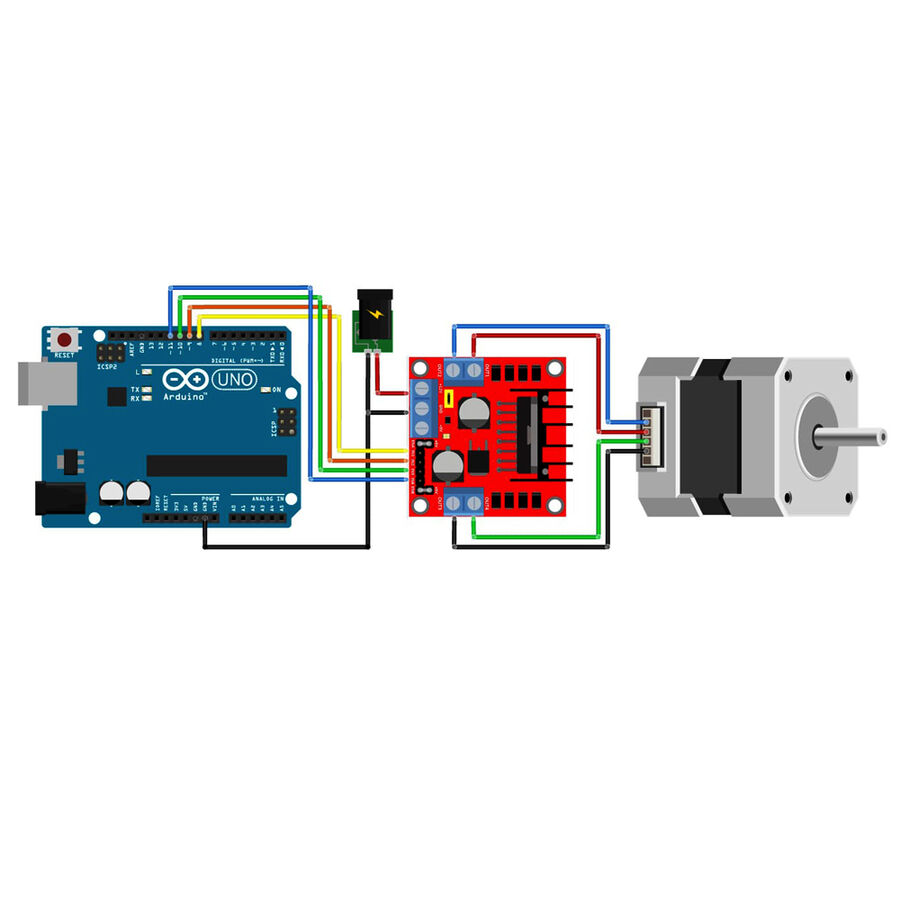


Figure 1.1

First of all, we need a motor driver to be able to start the motor. In fact, we can drive the motor with BJT or mosfets in circuits built with bare components. But here I used the ULN2003 motor diverter. The motor I use is 28BYJ-48 Unipolar Stepper. Figure 1.1 is a representation of this circuit. But in Figure 1.2 you will see the exact one-to-one schematic of this circuit.

Motor drivers actually help us in controlling the current going to the motor with voltage. Because most of the time, when we connect the motor to a source without using a motor driver, it is observed that the voltage is suddenly zero, either the motor runs at a constant speed and its speed cannot be changed, or the motor breaks down. After connecting our motor driver, we need to make the arduino connections. But first, let's give some information about the circuit elements. The 28BYJ-48 is a 5-wire unipolar stepper motor that moves 32 steps per rotation internally but has a gearing system that moves the shaft by a factor of 64. The result is a motor that spins at 2048 steps per rotation. It should be noted that some of these motors may have a different gearing system so the number of steps per rotation of your motor may not be the same. The 28BYJ-48 runs on 5 volts. The motor is commonly packaged with a tiny driver board based around the ULN2003 Darlington transistor array.  The board has a connector that mates perfectly with the motor wires so it is very easy to use. There are also connections for four 5-volt digital inputs as well as power supply connections.

On the subject of power supplies one very important thing to note is that you should NEVER use the 5-volt power from your Arduino to power this (or any) stepper motor no matter how tempting it is. Even though the 28BYJ-48 doesn’t draw much current it will induce electrical “noise” onto its power supply lines and this could damage your Arduino.  Always use a separate power supply to power your stepper motors!

We will hookup our motor, driver, and Arduino as follows:

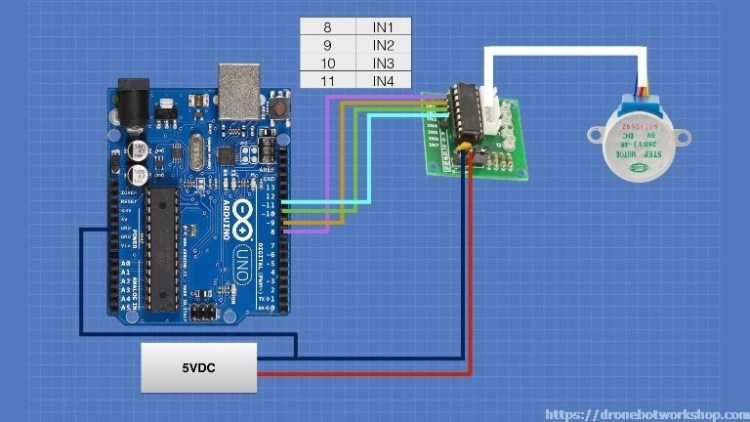


Figure 1.2

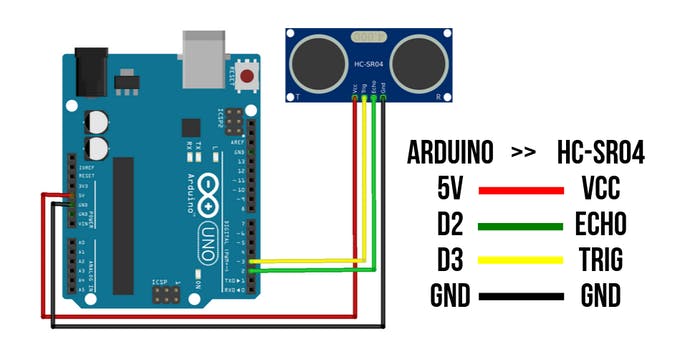
Arduino provides a library for us to use stepper motors without going into the basics and it is very simple to setup.Certain pins are assigned for the distance sensor, but there is no need to use a ready pin for this. The distance sensor is based on the logic of measuring distance with sound waves. It finds the distance by calculating how long after the wave it sends from a channel comes back.

Figure 1.3

After connecting the distance sensor in a similar way to figure 3, we place the distance sensor under the elevator car and attach a roller to the end of the stepper motor and place it at the top of the elevator shaft. Then after the connections and schematic are established, we can move on to the matlab GUI design.

 Figure 1.4

As can be seen in Figure 4, the elevator installation has taken place. There are 2 suitable ways to use the arduinon via matlab, but it can be used in other ways. The first is to establish serial communication between matlab and arduino. The second is the use of matlab, where it short-circuits the arduino's processor and replaces itself with the arduino's processor. I used the first way here.

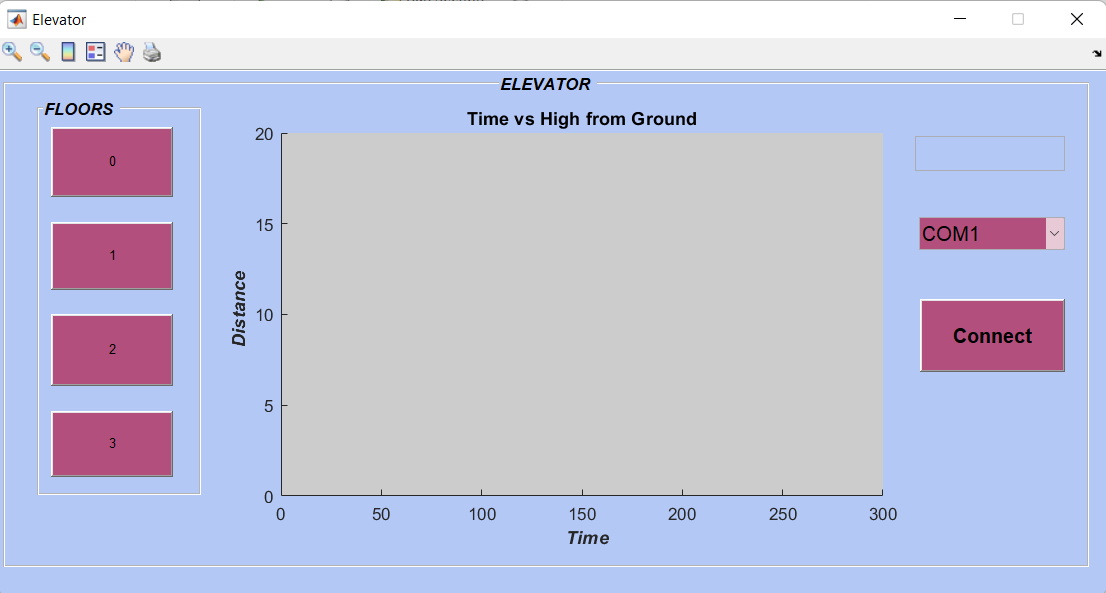


Figure 1.5

My Matlab GUI interface is as in figure 5. The buttons on the left control the transitions to the floors. For the system to work, the com port with the arduino must be selected from the left side and the connect button must be pressed. The distance graph starts to be drawn simultaneously.

**MATLAB CODE:**

function varargout = Elevator(varargin)

% ELEVATOR MATLAB code for Elevator.fig

% ELEVATOR, by itself, creates a new ELEVATOR or raises the existing

% singleton\*.

%

% H = ELEVATOR returns the handle to a new ELEVATOR or the handle to

% the existing singleton\*.

%

% ELEVATOR('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in ELEVATOR.M with the given input arguments.

%

% ELEVATOR('Property','Value',...) creates a new ELEVATOR or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before Elevator\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to Elevator\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Elevator

% Last Modified by GUIDE v2.5 02-Nov-2021 15:25:54

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Elevator\_OpeningFcn, ...

'gui\_OutputFcn', @Elevator\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before Elevator is made visible.

function Elevator\_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 Elevator (see VARARGIN)

% Choose default command line output for Elevator

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes Elevator wait for user response (see UIRESUME)

% uiwait(handles.figure1);

axes(handles.axes1);

title('Time vs High from Ground');

xlabel('\it \bf Time');

ylabel('\it \bf Distance');

xlim([0,300]);

ylim([0,20]);

global x a

x=0;

% --- Outputs from this function are returned to the command line.

function varargout = Elevator\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

global a;

sendData = 1;

fprintf(a,'%i',sendData); %this will send 5 to the arduino

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

global a;

sendData = 2;

fprintf(a,'%i',sendData); %this will send 5 to the arduino

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

global a;

sendData = 3;

fprintf(a,'%i',sendData); %this will send 5 to the arduino

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

global a;

sendData = 4;

fprintf(a,'%i',sendData); %this will send 5 to the arduino

% --- Executes on selection change in popupmenu1.

function popupmenu1\_Callback(hObject, eventdata, handles)

% hObject handle to popupmenu1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array

% contents{get(hObject,'Value')} returns selected item from popupmenu1

% --- Executes during object creation, after setting all properties.

function popupmenu1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to popupmenu1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

delete(instrfindall);

contents = get(handles.popupmenu1,'String');

popupmenu1value = contents{get(handles.popupmenu1,'Value')};

global a;

try

a = serial(popupmenu1value,'BaudRate',9600); % insert your serial

fopen(a);

set(handles.edit1, 'BackgroundColor','green');

set(handles.edit1,'String','Connected');

catch

set(handles.edit1, 'BackgroundColor', 'red');

set(handles.edit1,'String','Error');

end

global a;

global x;

for i = 1:300

val=str2double(fscanf(a));

x = [x val];

plot(handles.axes1,x,"Color","green");

title('Time vs High from Ground');

xlabel('\it \bf Time');

ylabel('\it \bf Distance');

xlim([0,300]);

ylim([0,20]);

drawnow

set(handles.edit1,'String',num2str(val));

pause(1);

end

save('data.mat');

guidata(hObject, handles);

**ARDUINO CODE:**

#include <Stepper.h> // Include the header file

#define STEPS 32// change this to the number of steps on your motor

Stepper stepper(STEPS, 8, 10, 9, 11);// create an instance of the stepper class using the steps and pins

int val = 0;

int flr = 1;

int one = 1024;

int two = 2048;

int three = 4096;

int trigPin = 13;

int echoPin = 12;

int j=0;

long zaman;

long mesafe;

void setup() {

Serial.begin(9600);

stepper.setSpeed(600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin,INPUT);

cli();

/\* Ayarlamaların yapılabilmesi için öncelikle kesmeler durduruldu \*/

/\* Timer1 kesmesi saniyede bir çalışacak şekilde ayarlanacaktır (1 Hz)\*/

TCCR1A = 0;

TCCR1B = 0;

TCNT1 = 0;

OCR1A = 15624;

/\* Bir saniye aralıklar için zaman sayıcısı ayarlandı \*/

TCCR1B |= (1 << WGM12);

/\* Adımlar arasında geçen süre kristal hızının 1024'e bölümü olarak ayarlandı \*/

TCCR1B |= (1 << CS12) | (1 << CS10);

TIMSK1 |= (1 << OCIE1A);

/\* Timer1 kesmesi aktif hale getirildi \*/

sei();

/\* Timer1 kesmesinin çalışabilmesi için tüm kesmeler aktif hale getirildi \*/

}

/\* Arduino otomatik olarak her saniye aşağıdaki fonksiyonu çalıştıracaktır \*/

ISR(TIMER1\_COMPA\_vect){

digitalWrite(trigPin, LOW);

delayMicroseconds(5);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

zaman = pulseIn(echoPin, HIGH);

mesafe= (zaman /29.1)/2;

Serial.println(mesafe);

}

void loop() {

while(Serial.available()>0)

{

val = Serial.parseInt(); // istenilen kat

switch(val){

case 1:

if(flr==1){}

else if(flr==2)

{

stepper.step(one);flr = 1;

}

else if(flr==3)

{

stepper.step(two);flr = 1;

}

else if(flr==4)

{

stepper.step(three);flr = 1;

}

break;

case 2:

if(flr==2){}

else if(flr==1)

{

stepper.step(-one);flr = 2;

}

else if(flr==3)

{

stepper.step(one);flr = 2;

}

else if(flr==4)

{

stepper.step(two);flr = 2;

}

break;

case 3:

if(flr==3){}

else if(flr==1)

{

stepper.step(-two);

flr = 3;

}

else if(flr==2)

{

stepper.step(-one);flr = 3;

}

else if(flr==4)

{

stepper.step(one);flr = 3;

}

break;

case 4:

if(flr==4){}

else if(flr==1)

{

stepper.step(-three);flr = 4;

}

else if(flr==2)

{

stepper.step(-two);flr = 4;

}

else if(flr==3)

{

stepper.step(-one);flr = 4;

}

break;

}

}

delay(100);

}