BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

EEE 316: Power Electronics Laboratory

Project Report

Stepper Motor Control using Microcontroller

Course Teachers:

Munia Ferdoushi

Lecturer, Department of EEE, BUET

Satyaki Banik

Adjunct Lecturer, Department of EEE, BUET

Date of Submission: 23/2/2022

Submitted By:

Group No: 6
EEE 17 (A2)

Members:
Shoaib Ahmed – 1706046
Saleh Ahmed Khan – 1706053
Sadat Tahmeed Azad – 1706064
Md. Mahadi Hasan – 1706065

Introduction:

Stepper motor is a digital electromechanical device, which can rotate in discrete steps as opposed to continuous rotation provided by other motors. This characteristic of a stepper motor of providing fixed angle rotation without a closed loop feedback makes it very useful in factory or other fine-tuned applications where precise movement is required. In this project, we will attempt to make a stepper motor driver, which will allow the user to drive the stepper motor to achieve the user's required function.

Objective:

To implement a stepper motor driving circuit and generate necessary gate pulses to drive the stepper motor, in any direction, speed, stepping modes according to the input provided by the user.

Simulink Model:

A custom MATLAB Function is used to the pulses after a variable time delay.

Physical Circuit:

An Arduino Microcontroller with adjustable delay is used to generate pulses.

Simulink Diagram:

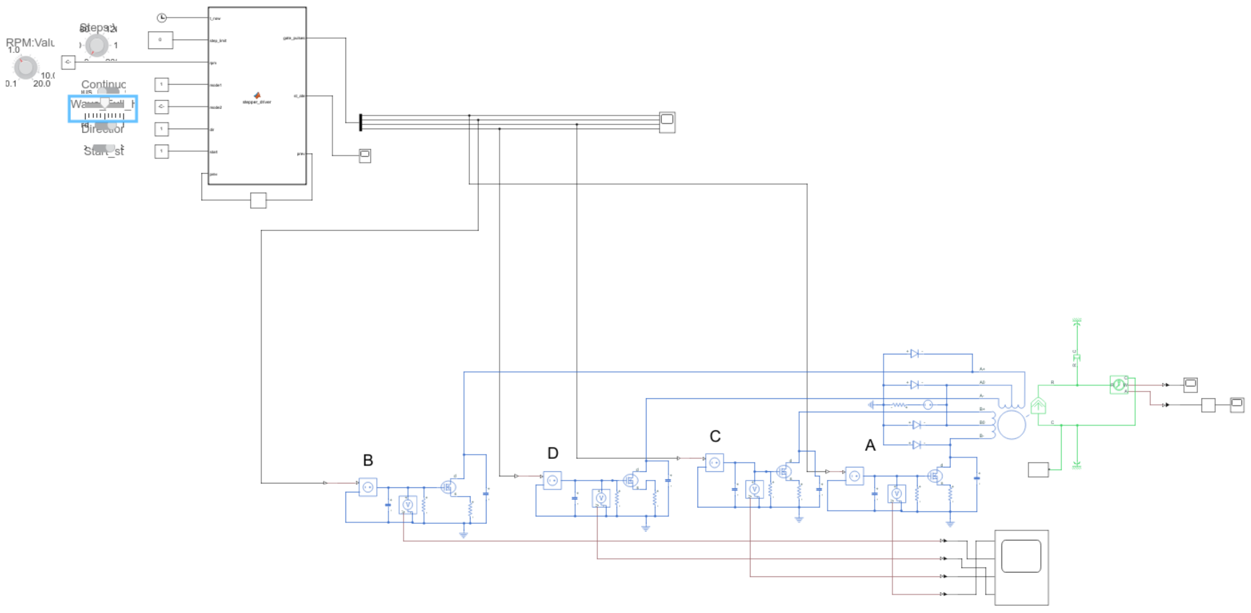


Figure: Simulink Model of the Entire System

The entire circuit consists of 3 parts:

1. **Electrical Power Circuit:** (In Blue - Simscape Electrical)

Consists of a 12V power source, a unipolar stepper motor, 4 n-MOSFETs to drive the stepper motors, freewheeling diodes for absorbing inductive kick, additional capacitance and resistances to introduce non ideality into the system and allow Simulink to run smoothly. Without them, Simulink has difficulty simulating all calculations, as out circuit has both mechanical and electrical parts with abrupt changes given by the user.

The gate pulses generated by a custom MATLAB function is used input and the gate pulses are visualized in the appropriate scopes.

2. **Mechanical System:** (In Green - Simscape Mechanical)

The mechanical shaft of the motor is attached to a damper to smooth out the sudden and oscillating motion of the stepper motor, then it is attached to a motion sensor with output the speed and angle of the motor.

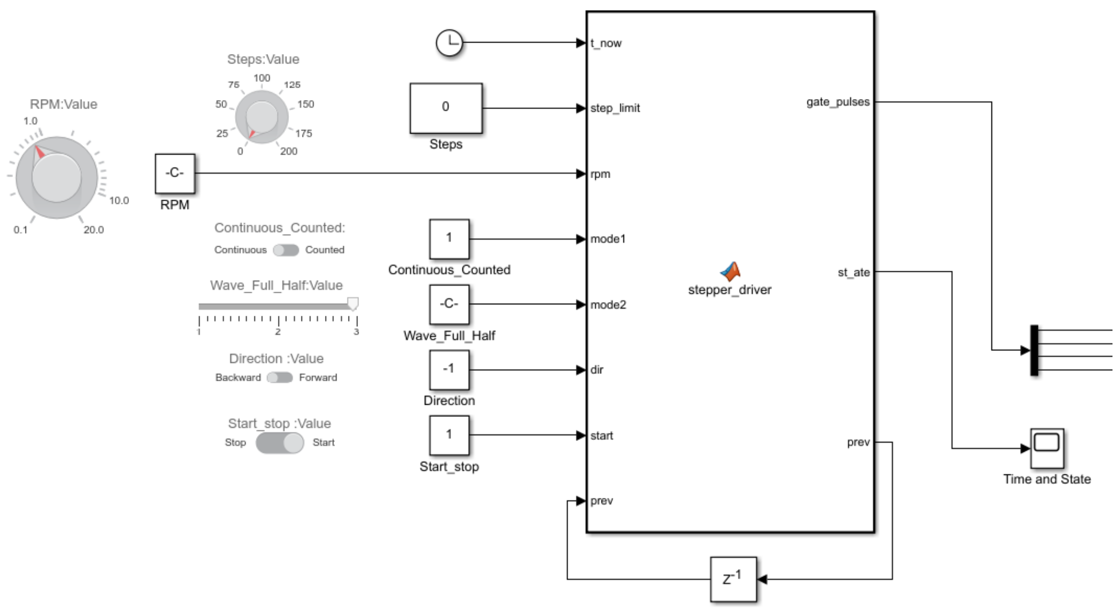


Figure: MATLAB Function and User Inputs

3. **MATLAB Function and User inputs:** (Code in explanation attached in appendix)

Appropriate gate pulses are being generated from the function by taking inputs from the user using various knobs. The function allows the user to set the continuous/discrete stepping move of the driver, as well as the stepping method (wave/full/half), and when to start or stop the motor. It also lets the user specify the speed of the motor and the discrete steps to be taken.

The function also takes the current simulation time as input and the previous state of the motor, which is taken as a feedback.

Demonstration:

An overview of every possible input and subsequent behavior of the gate pulses and the angle of the motor is shown below:

Continuous Stepping Mode (Forward):

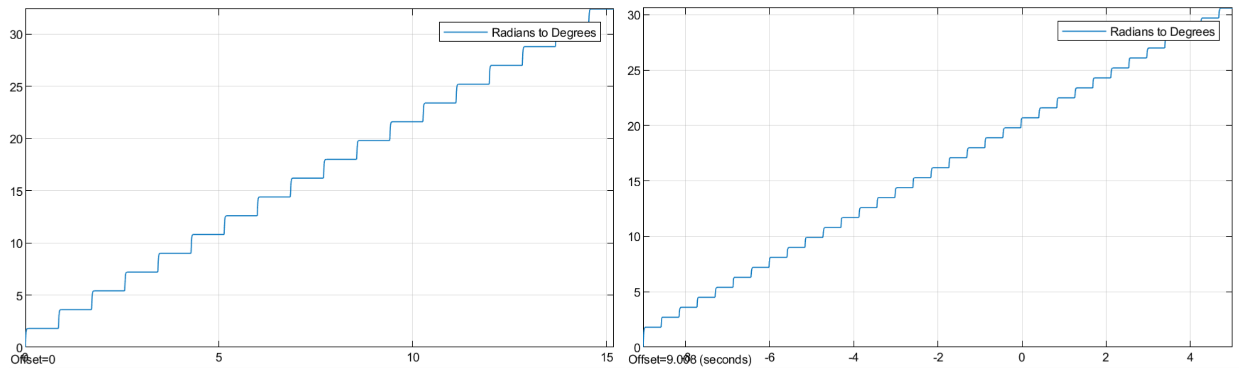


Figure: Continuous, Wave/Full Step Drive (Left), Half Step Drive (Right)

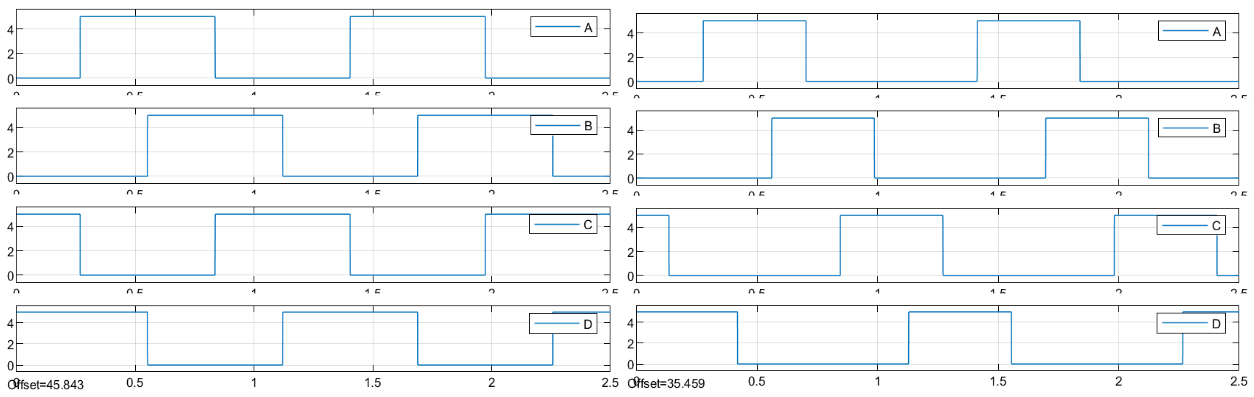


Figure: Continuous, Full Step Drive (Left), Half Step Drive (Right)

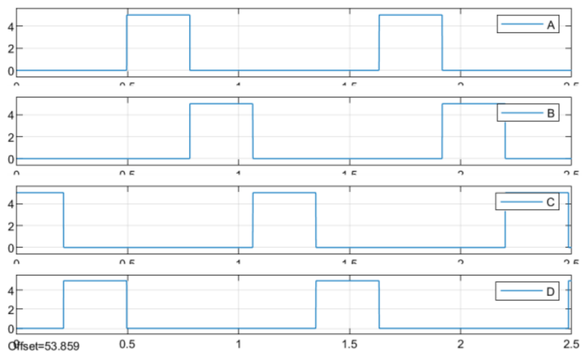
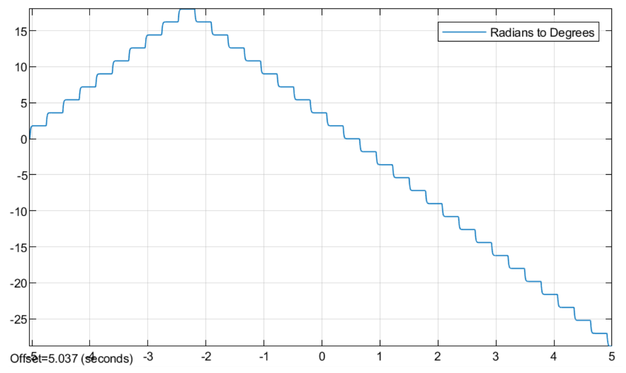


Figure: Continuous, Wave Drive

For Wave/Full Drive, we can see the motor is continuously stepping in 1.8-degree steps, whereas in half stepping mode, the angle is 0.9 degrees. Looking at the gate pulses, we can also see that they're following the sequence of their respective mode.

Continuous Stepping Mode (Backward):



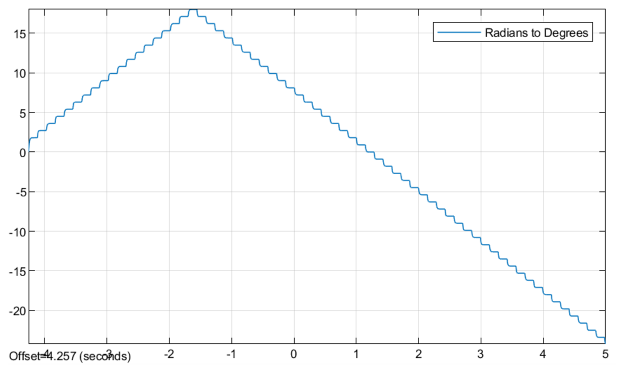
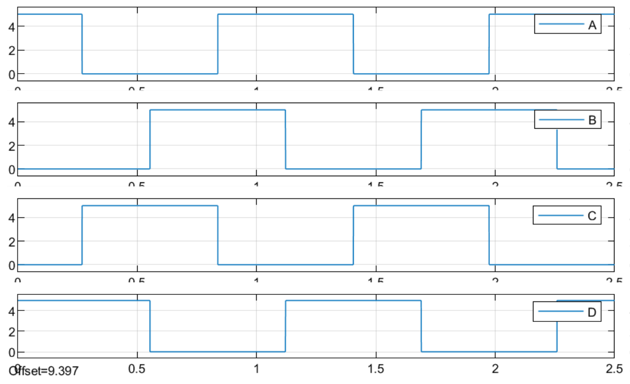


Figure: Continuous, Wave/Full Step Drive (Left), Half Step Drive (Right)



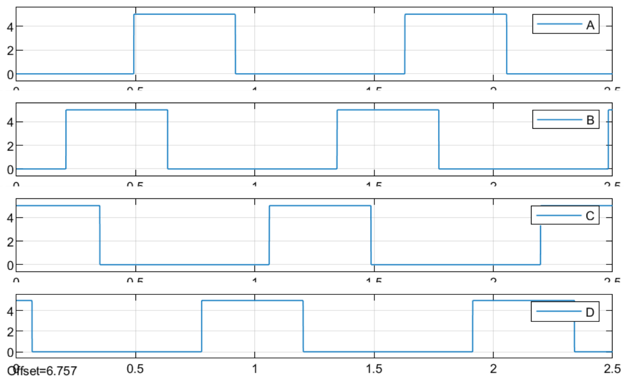


Figure: Continuous, Full Step Drive (Left), Half Step Drive (Right)

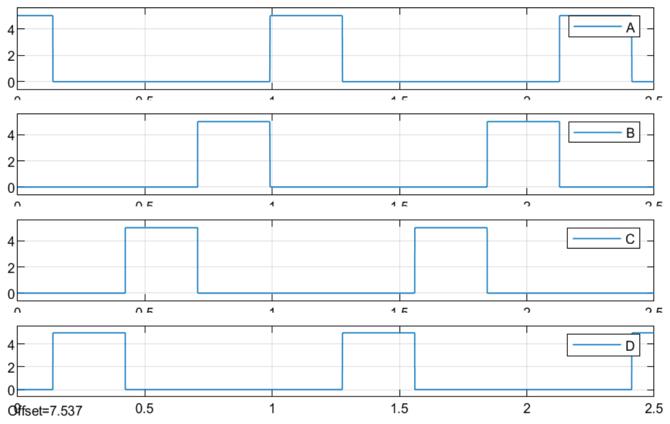
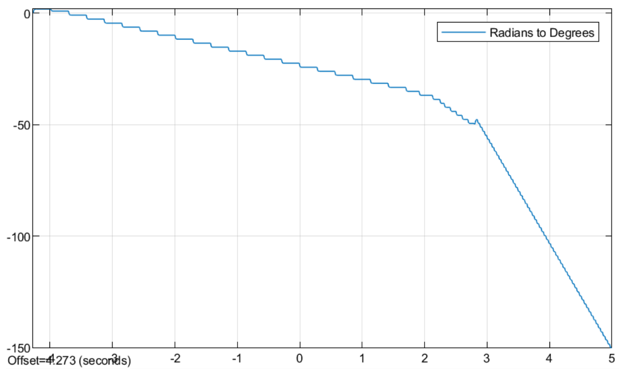


Figure: Continuous, Wave Drive

In case of backward rotation, we can see the motor can easily switch from forward to backward rotation without any issues. The respective gate pulses also switch their sequence to match that of backwards rotation. Same as forward rotation, the motor takes 0.9 degrees step in half step mode and in 0.9 degrees otherwise.

Continuous Stepping Mode (Variable Speed):



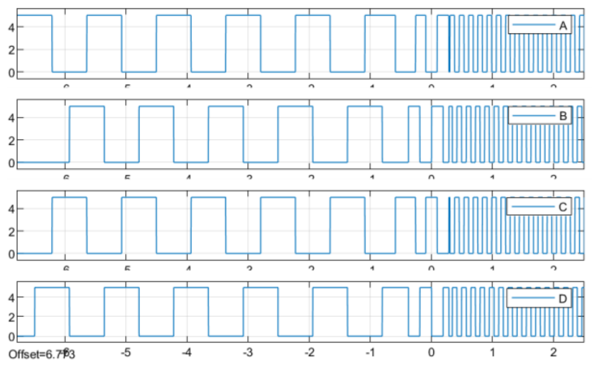
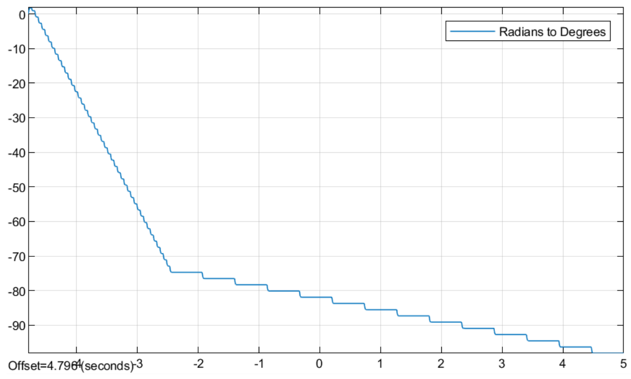


Figure: Continuous, Wave Drive, Speed Increase, Motor Angle (Left), Gate Pulses (Right)



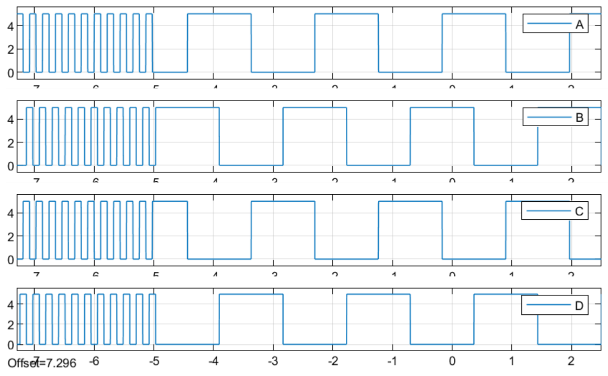


Figure: Continuous, Wave Drive, Speed Decrease, Motor Angle (Left), Gate Pulses (Right)

As the speed knob is increases, we can see the motor movement becomes much smoother, as well the increased slop indicating the increase in speed, also, we can notice the frequency of the gate pulses increase at higher speed.

Consequently, as the speed in lowered, the motor's discrete stepping can once again be recognized, also the gate pulses' frequency gets lowered.

Discrete Stepping Mode:

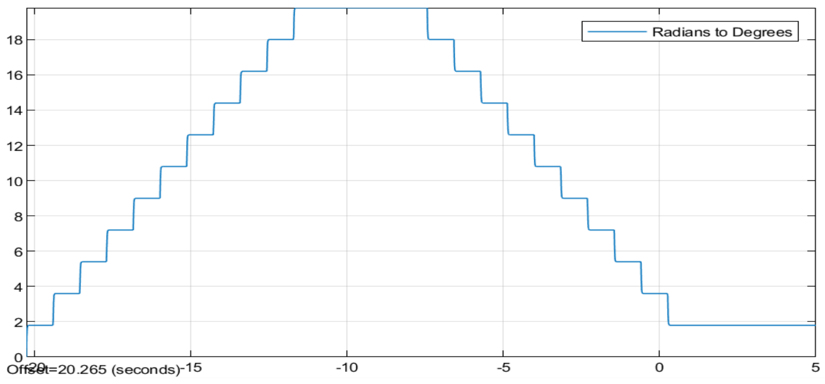


Figure: Discrete Step Mode, Wave Drive (Forward/Backward)

Both forward and backwards rotation in discrete stepping mode is shown here. The inputs are set to wave drive mode and 10 steps, and we can see from the graph the motor has taken exactly 10 steps and 1.8 deg intervals. The initial motor angle in this case was 1.8 degrees.

Hardware Implementation:

For offline Demonstration of this project, we managed to implement a bi-polar stepper motor driving mechanism using H bridges using an Arduino Microcontroller as a gate pulse generator.

Main Components:

- Arduino Uno (Main Controller)
- IRFZ44n MOSFETs (8x)
- Resistors (10k)
- Battery (Power Source)
- NEMA 23 Bipolar Stepper Motor
- 16x2 LCD Display (for visualization)

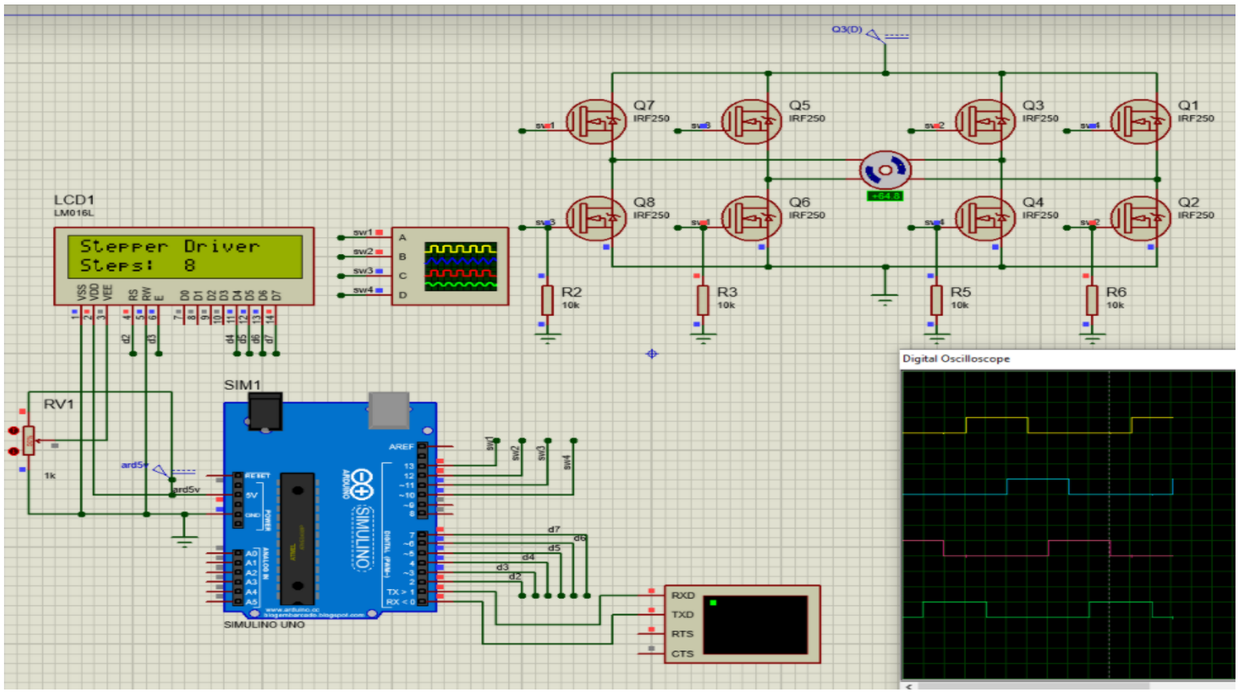


Figure: Circuit Diagram of Physical Circuit

Bi-polar Motors are slightly different than unipolar ones because they have 2 coils and 4 terminals, which needs to be energized using opposite polarities while stepping. This requires the use of H-bridges, which makes this polarity switch possible.

In this circuit, the motor is rotating in half step mode.

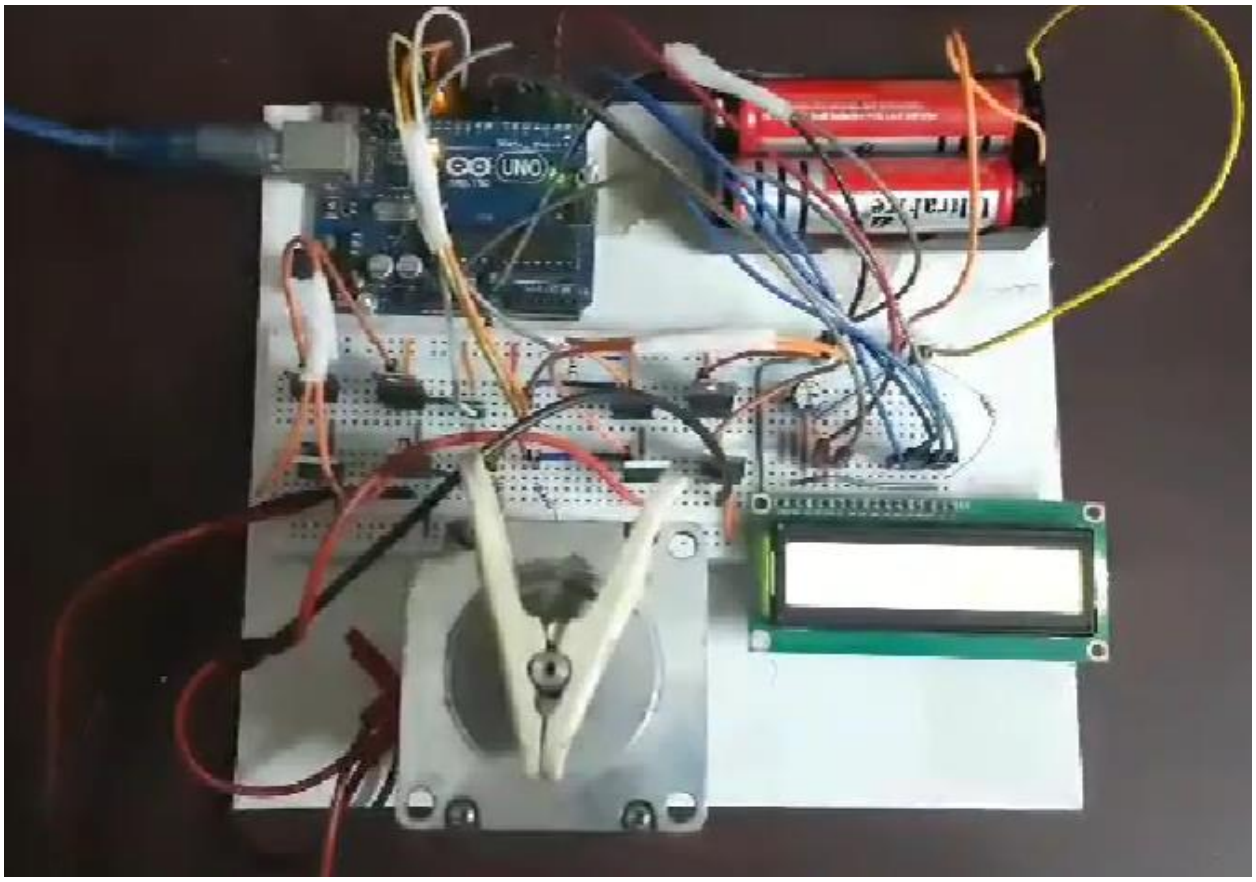


Figure: Hardware Implementation of Stepper Motor Driver

Conclusion:

In fine, the goal of this project was to familiarize ourselves with the implementation of power electronic circuitry and their proper implementation, which has been achieved by successful implementation of this driver, which brings versatility and convenience to a very useful machine. Similar drivers also available in the market, which provide different stepping options as well as micro stepping for more precise operation of the stepper motor.

Appendix:

MATLAB Code and Explanation:

This is the MATLAB function which we are using to generate the necessary gate pulses.

The inputs are current time, steps, rpm, modes, direction, start/stop and [previous state of the motor.

We can notice a feedback, which is 7x1 array called “prev”, which stores some necessary information from the previous run of the function and carries them to the next one.

```
t_prev=prev(1);
state=prev(2);
mode1_prev=prev(3);
mode2_prev=prev(4);
dir_prev=prev(5);
start_prev=prev(6);
steps=prev(7);
%Transition
if start~=start_prev || mode2~=mode2_prev || dir~=dir_prev %||
mode1~=mode1_prev
    %Transition took place
    start_prev=start;
    mode1_prev=mode1;
    mode2_prev=mode2;
    dir_prev=dir;
    %RESET Steps
    steps=0;
end
```

Here we are retrieving the info from the matrix. The first one is the time when a last time was taken. By comparing it with the current time, the function will know when to take the next step. The other 6 inputs are necessary because we want to reset the motor step count and other parameters (inside the transition block) when the mode is changed while running the motor or the motor is stopped/started again.

```
step_limit = round(step_limit);
% mode1 = 2;  % 1=continuous, 2=counted steps
mode2 = round(mode2);  % 1=wave drive, 2=full step, 3=half step
T = 60/rpm;
step_size=T/(200 + 200*(mode2==3));
% dir = 1;  % -1 = backward, 1 = forward
```

Here the step size is being calculated from mode2. Notice that for half step mode (i.e. mode2=3) the step size will be half.

```
st_mat= [1 0 0 0;
         1 1 0 0;
         0 1 0 0;
         0 1 1 0;
         0 0 1 0;
         0 0 1 1;
         0 0 0 1;
         1 0 0 1;];
```

This state matrix has 8 rows and contains all the necessary variations of the gate pulses that will be required to drive our motor in the desired mode. The four columns in each row denote the pulse in each of the four MOSFET gates i.e. A, B, C and D, during one step duration. For example, if we want to run it in wave drive mode, we will need rows 1, 3, 5, 7; for full step mode, we will need row 2, 4, 6, 8; and for half step mode, we will need all 8 rows.

```
if (t_now-t_prev)>=step_size %Counting Time
    t_prev=t_prev+step_size;

    if start==1 && (steps<step_limit || mode1 == 1) %Running Motor
        steps=steps+1;

        %Wave Drive
        if mode2==1 %continuous
            if mod(state,2)==0
                state=mod(state+dir,8);
            else
                state=mod(state + dir*2,8);
            end
        end

        %Full Step
        if mode2==2 %continuous
            if mod(state,2)==1
                state=state+dir;
            else
                state=mod(state + 2*dir,8);
            end

            if state==0
                state=8;
            end
        end

        %Half step
        if mode2==3 %continuous
            state = mod(state+dir,8);

            if state==0
                state=8;
            end
        end
    end
end
```

Here, when the difference between current time and last step time is equal to that of a step size, the function takes the next step, and advances or goes back one row in the necessary sequence. For example, if the motor is running backwards in full step mode and the last gate pulse sequence was [A B C D]=[0 1 1 0] (row 4), then in the next step the sequence will be [A B C D]=[1 1 0 0] (row 2)

```
gate_pulses=5*st_mat(state, :);
st_ate=state;
prev=[t_prev; state; mode1_prev; mode2_prev; dir_prev; start_prev; steps];
```

Finally, we are multiplying the voltages by 5 (which is required to drive the MOSFETs) and sending the necessary info to the feedback array.