Stepper Motor

Control & Drive

Ayman Miri & Christian Steffen
LYCÉE GUILLAUME KROLL 32, rue Henri Koch

Table of Content

1 Introduction

1.1 Describe the mission

- Stepper motor project : Our task for the 2nd Semester is the controlling of a stepper motor. As we know, controlling a stepper motor needs a driver and a microcontroller, now we can say it's easy until you hear that you are not allowed to use usual Arduino libraries and a driver who has a translation module inside of it and the motor you are going to drive is a bipolar motor. This setup needs a lot of work for the programming part.
- The main task was driving the motor with four buttons and each button has his own task like the first button will move the motor 10 steps forward and the second one will move it just one step. For the other two button it's just the opposite.

1.2 First steps

- We started first with the time planning and dived into the mission. We needed to decide what we are going to use and how big the project will be. We were searching about the materials that we are going to use. Like the type of driver and the microcontroller and afterwards we need to have a look how bipolar motors actually works.
- We decided to use the driver "*L298N*" because it has two full **H bridge circuits**, and it has got no translation module.
- Then the choice of the microcontroller was of course the (Arduino UNO) not because it has more RAM or speed then other, it's just because we had it already and it fitted perfectly in our circuit and abilities.
- After collecting information for the programming part, and the search that we effected on the materials, we created the time planning that you will see coming on the chapter of time planning.
- We found a lot of sources on the internet which explains how to drive the bipolar stepper motor. It was easy once you understood the logic behind it. But here we faced a problem, for us it was a problem, but for our teacher it wasn't, the problem is that when we programmed the microcontroller, we used the function "**delay"** between each step, and we wanted to use an OLED in addition to the project. The OLED wasn't in the task, but we added one just to make it more interactive. And of course, when you use "**delay"**, you block the whole program until this delay is done. So, we decided to use the function "**millis()"**, but also, we faced a problem in controlling the output of Arduino, because "**millis()"** is like a process aside of your main process.
- We also decided to build box to contain the project and it wasn't required too.

2 Material List

Equipment	Type	Nr.
Arduino	Uno (SMD)	1
Stepper Motor (Bipolar)	SM-17HS4023	1
Driver	L298N	1
Power-Supply	MW MB10EU Variant output DC voltage: 3V > 12V Output current : 1000mA	1
Jumper cable	Male/Female , Male/Male	x
Button	Pushbutton	5
Resistance	10k ohm	5
OLED	ADA938 OLED 128x64	1
3D Printer	Artillery X2	
Filament	PETG	

3 Software

Name	Description
IDLE Arduino	Used to write the program
Autodesk Eagle	Used to draw the circuits
Autodesk Fusion 360	Used to draw the 3D design
Ultimaker Cura	Used to generate the (.gcode) extension to use on the 3D printer
Draw.io	Used to draw the flowchart

4 Time planning

- For the time planning it was changed one time, I'm going to show you the first one then the modified one.

Date	Task
28.02.2022	Preparation of materiallist
07.03.2022	Discussion over the materiallist
14.03.2022	Clarify usage of the materials
21.03.2022	Plan the circuit & writing program
28.03.2022	Plan the circuit & writing program
25.04.2022	Testing the circuit
02.05.2022	Writing report
16.05.2022	Insert the circuit in the construction
30.05.2022	Preparation of the report
13.06.2022	Discuss the report
20.06.2022	Discussion over the project & report
27.06.2022	Presentation

Date	Task
28.02.2022	Preparation of materiallist
07.03.2022	Discussion over the materiallist
14.03.2022	Clarify usage of the materials
21.03.2022	Plan the circuit & writing program
28.03.2022	Plan the circuit & writing program
25.04.2022	Writing program
02.05.2022	Writing program and problem solving
16.05.2022	Writing program and problem solving
30.05.2022	Create a 3D box for the circuit
13.06.2022	Modify 3D box for the circuit
20.06.2022	Testing construction & do the report
27.06.2022	Finish the report and presentation

5 Theory

5.1 Driver

- To have a complete control over a motor you must control the speed and rotation direction. This can be achieved by combining these two techniques :
  - **PWM** – to control the speed.
  - **H-Bridge** – to control the rotation direction.

- **Control Speed** : On the Arduino we didn't use the PWM Pins, but we wrote a program to generate digital pulses in controlling the time (frequency). As the digital pulses increase in frequency, the step movement changes into a continuous rotation, with the speed of rotation directly proportional to the frequency of the pulses.

- **Rotation Direction** : we just reversed how the Pins were activated on the coils of the motor that connected to two H-Bridge circuits.

- **How H-Bridge circuit works** : The spinning direction of any DC motor can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-bridge. Closing two specific switches at a time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

  The following images show the working of the **H-bridge circuit**.

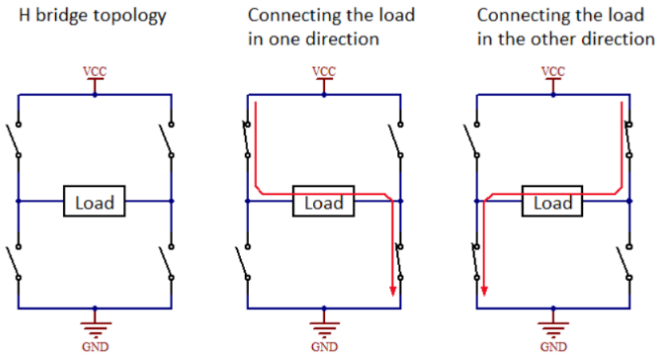

Figure 1 : H-Bridge circuit direction change

- Here is also an image of the L298N Describe his inputs and outputs connections.

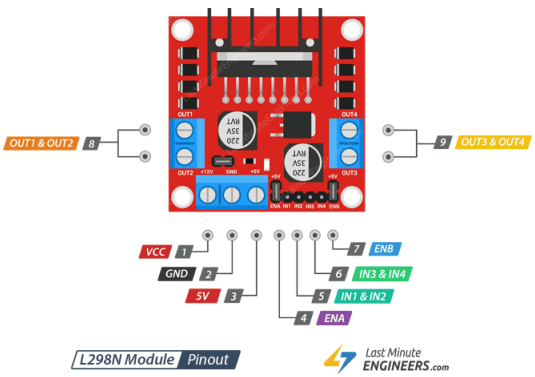

Figure 2 : L298N Driver

5.2 Stepper Motor

5.2.1 What is a Stepper motor?

- A stepper motor is a brushless, synchronous electric motor which converts digital pulses into mechanical shaft rotation. It's normal shaft motion consists of discrete angular movements of essentially uniform magnitude when driven from sequentially switched DC power supply.

5.2.2 How does a stepper motor work?

- Every revolution of the stepper motor is divided into a discrete number of steps, in many cases 200 steps (200 steps = 360° rotation), and the motor must be sent a separate pulse for each step. The stepper motor can only take one step at a time and each step is the same size.

- Since each pulse causes the motor to rotate a precise angle, typically 1.8°, the motor's position can be controlled without any feedback mechanism. As the digital pulses increase the frequency, the step movement changes into continuous rotation, with the speed of rotation directly proportional to the frequency of the pulses.

5.2.3 Types of Stepper motor

- There are two types of stepper motors: unipolar and bipolar. In our project we used a bipolar one. As we know the bipolar stepper requires a lot of software and hardware.
- We start with the hardware since we know it's a little bit confusing for some of people :
  - In this motor we got 4 wires connected to two coils, first two of them go to first coil and call (A- & A+), and the other two go to second coil and call (B- & B+).
  - We connect these 4 wires to the driver l298n how has two built-in H-Bridge circuits, each coil connected to an H-Bridge circuit. See in **Figure 3**.

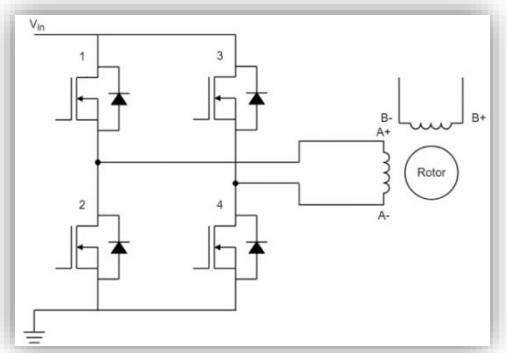

Figure 3: example of coil connected to H-Bridge circuit.

5.2.4 Stepper motor driving techniques

- There are 4 different driving techniques for a stepper motor :

1. Wave mode : only one phase at a time is energized.

2. Full-step mode : two phases are always energized at the same time.
3. Half-step mode : is a combination of wave and full-step modes.
4. Microstepping mode : can be seen as a further enhancement of half-step mode because it allows to reduce even further the step size and to have a constant torque output.

- For us we used the **full-step mode**.

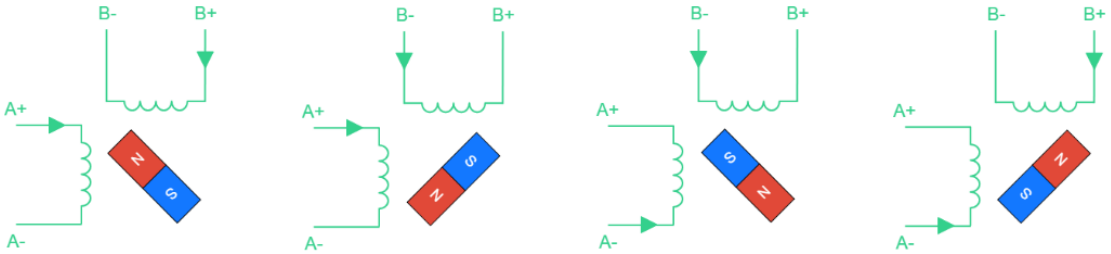

Figure 4: Full-step mode alimentation

- Here also you have the truth table that we used in our code to represent the situation in this image.

		1	2	3	4
	A	HIGH	LOW	LOW	HIGH
	$\overline{A}$	LOW	HIGH	HIGH	LOW
	B	HIGH	HIGH	LOW	LOW
	$\overline{B}$	LOW	LOW	HIGH	HIGH

Figure 5: Truth/False-table of the full-step method

- To understand more about these four different modes, we invite you to have a look on this website in the section of stepper motor driving techniques, here.

6 Schematic

6.1 General schematic

- Picture of the plan. The plan will be printed and provided with the appendices .

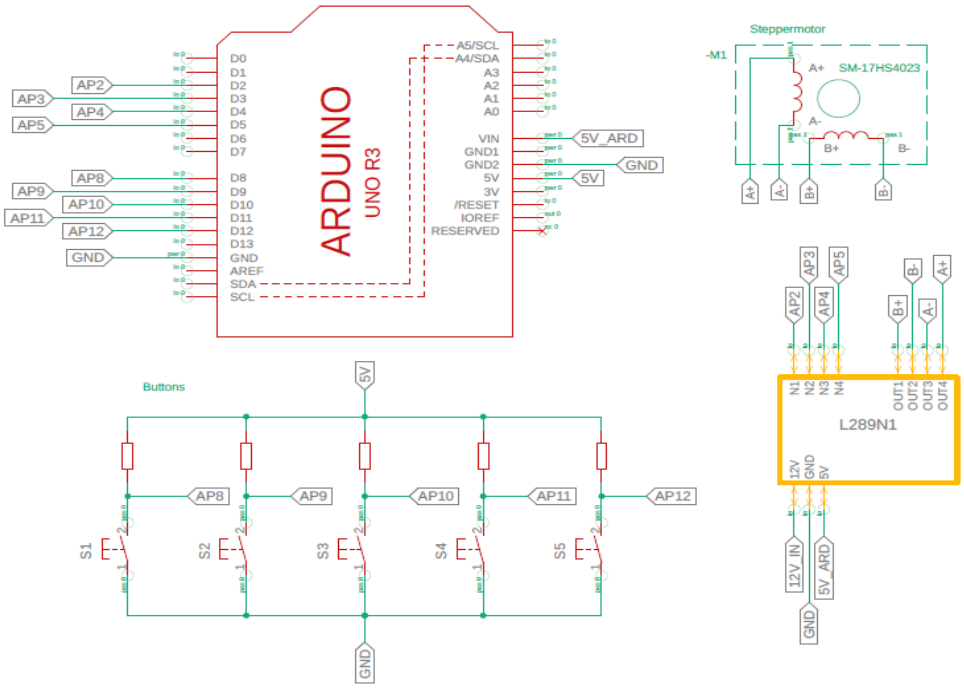

Figure 6 : Circuit schematic

6.1.1 Arduino UNO SMD connecting plan

- Explanation of the connected Pins :

Pins name	Implementation
AP 2/3/4/5	Stepper Motor coils
AP 8/9/10/11/12	Pushbuttons
SDA/SCL	OLED 128x64 (SDA -> Data & SCL -> CLK)
5V_ARD	5V Provided From the regulator in the driver L298N
5V	(Internal 5V Arduino) OLED / pushbuttons circuit
GND	Ground between driver/pushbuttons circuit/ Arduino (Ground common).

6.1.2 Buttons connecting plan

- Explanation of the connected Pins :

Pins name	Implementation
5V	From Arduino connecting directly to the resistances then from the resistances to the Arduino Pins (Pullup)
GND	Ground common. Short Arduino Pins to ground to detect changes
AP#	Inputs Arduino

6.1.3 L298N connecting plan

- Explanation of the connected Pins :

Pins name	Implementation
12V_IN	External supply
5V_ARD	Supply to the Arduino
GND	External supply (Ground common)
AP#	Arduino outputs Pins
A+	Motor coil A+
A-	Motor coil A-
B+	Motor coil B+
B-	Motor coil B-

6.1.4 OLED display connecting plan

- Explanation of the connected Pins :

Pins name	Implementation
5V	Power the display from Arduino
GND	Ground common
SCL	I2c from Arduino
SDA	I2c from Arduino

6.1.5 Stepper motor connecting plan

- Explanation of the connected Pins :

Pins name	Implementation
A-	Motor coil A- connected to output 2 (driver)
A+	Motor coil A+ connected to output 1 (driver)
B+	Motor coil B+ connected to output 3 (driver)
B-	Motor coil B- connected to output 4 (driver)

6.2 Wiring

- Here in this Diagram, you see a general view of the wiring circuit.

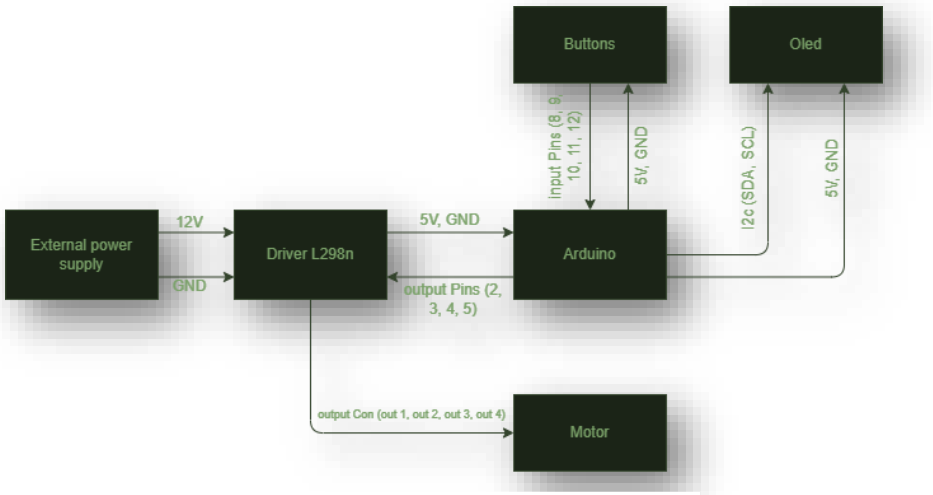

Figure 6 : Wiring circuit (simple)

7 Program :

7.1 Diagram of general function

- Here is a diagram showing how the program works.

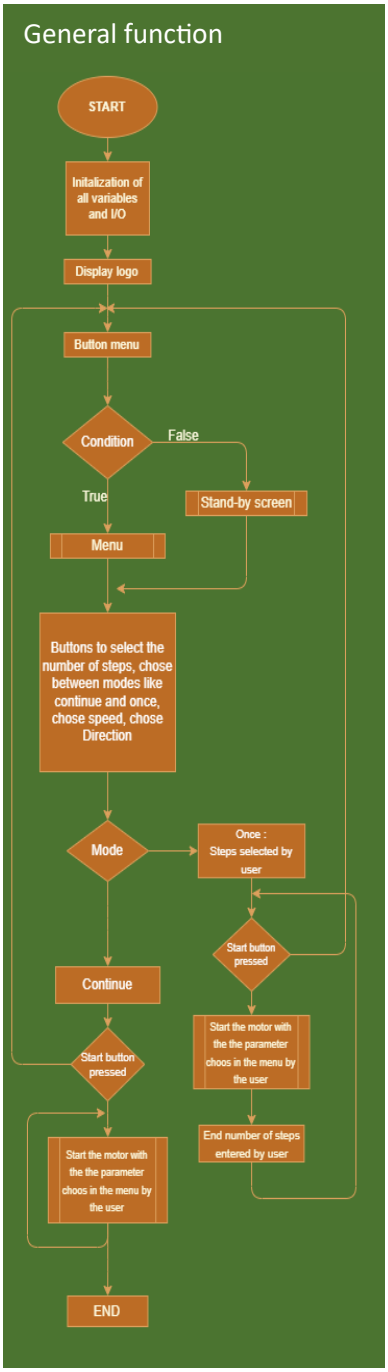

Figure 7 : Main Program

7.2 How it works ?

- First of all, our code has no blocking delay.
- Second thing that the code works with millis() function only.
- When you plug the supply, the Arduino starts the initialization of the I/O pins and variables.
- The OLED starts showing a logo, then with button “menu” you select which menu you want.
- Then with the other buttons you can select the mode, speed, direction, and steps.

7.3 Steeper motor code

- For the stepper we chose to do the program as a state machine, because it is simple and easily readable for anyone.
- To understand more about “state machine” and how it works we strongly recommend you watch this video.
- The diagram that you’ll see now is a general diagram about how the state machine works. In the code you will see conditions and counters also to count the steps and for this we can’t draw a flowchart because it’ll be too big, but don’t worry, our code explains himself, and as we said the code is easily readable.

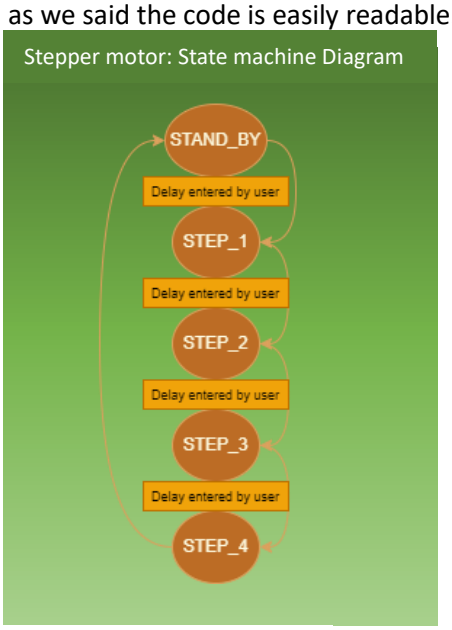

Figure 8 : State machine (simplified)

8 3D Design and Printing :

- In this chapter, there is nothing important to say because you'll have the (.stl) files and you can print it and modify it by yourself.
- The box was okay for our needs, the intention was to get a good looking of the circuit. It wasn't recommended for this project. So, we didn't focus on this, we were just trying out, what's possible for the next time.
- For the dimensions we used to come with the materials and we're going to provide them also to you with the dependencies.

Figure 9 : View of the box

9 Evaluation

In the end we have a great box with buttons and an OLED display, to manage the stepper motor. Our menu is simple structured, so everybody should be able to take control over it.

To built up the circuit was very easy, because we needed just one alimentation for the whole circuit. The code of the program was the hardest exercise we had in this project. We needed more hours and research for it, as we expected. It's also shown under the time planning.
To get the process of driving the motor working with the menu wasn´t so easy to realize. After some tries, we have now the way to manipulate the parameters first, then the display will show us the start button and then we can drive the motor the way we have set.

Additional we also made the case to implement the circuit just in a good simple safe way. The separation of the top and the bottom, makes it very easy and stable.
We had some dimension problems, but because it´s 3D printed, we could adjust very quick some corners and borders. Also, some diameters are more increased than ours shown in the document. The most important parts are fixed with screws. There was also an issue with the buttons because the space between the 3D printed parts and the button circuit board were too hard to push. So, we quickly managed the problem and just put a drop of glue on the 3D printed part to make it more touchable.

We enjoyed our project and put a lot of work in it, to make it as cool as possible without being to complicate. We are satisfied with the result and how it looks.

Appendices

- Here you can find the whole project :
    - https://github.com/Ayman628/Stepper_motor_bipolar
- Datasheet of the stepper motor.
- Datasheet of the driver.
- Schematic printed.
- Program printed.
- Dimensions of materials.