ECOLE NATIONALE SUPERIEURE
DE MECANIQUE ET DES MICROTECHNIQUES

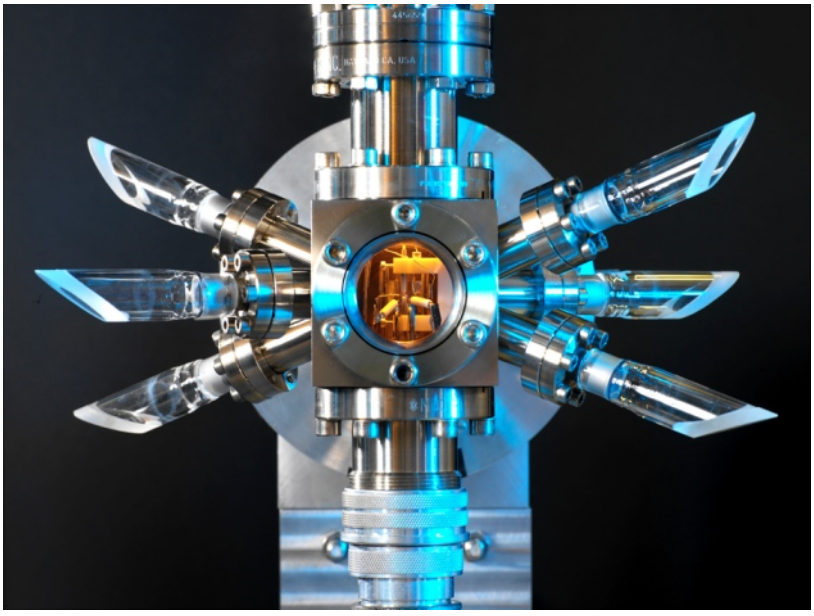
Internship Report

Jun-Aug 2024

SUPERRADIANT LAB

OHMS group

Automatic control of MOT oven



Written by :
Junaid RAMZAN

Submitted to :
Dr. Marion DELEHAYE

Summary

1 Control of Stepper motor with Arduino

1.1 Introduction

Magneto-optical trap is a technique used to cool and confine neutral atoms using a combination of magnetic fields and laser light. An important part of the apparatus is the oven. The oven generates a vapour of ytterbium atoms and a collimated beam is produced. As of now the atoms going out from the oven is controlled by a manual opening and closing of a door mounted on a shaft[1]. This initial set up works fine but we would like to use an automatic and remote control of the oven door.

In nutshell, the problem at hand is to remotely control the opening and closing of the oven door of Magneto-Optical Trap with the help of Arduino board and L298N motor driver.

1.2 Hardware Components

In order to control the stepper motor, we used Arduino Uno3 board, L298N dual DC driver and 28-BYJ48 uni-polar motor. These components were used for testing before mounting the the setup onto the bipolar motor used in the MOT. Since, we had only access to a uni-polar motor, the first step was to convert this uni-polar motor into a bipolar one. The uni-polar motor operates with one winding with a center tap per phase. Each section of the winding is switched on for each direction of the magnetic field. Each winding is made relatively simple with the commutation circuit, this is done since the arrangement has a magnetic pole which can be reversed without switching the direction of the current. In most cases, given a phase, the common center tap for each winding is the following; three leads per phase and six leads for a regular two-phase stepper motor but both these phases are often joined internally, this makes the stepper motor only have five leads as seen in fig 1.1

With bipolar stepper motors, there is only a single winding per phase. The driving circuit needs to be more complicated to reverse the magnetic pole, this is done to reverse the current in the winding. Unlike the unipolar stepper motor, the bipolar stepper motor has two leads per phase, neither of which are common. Our bipolar motor has two independent windings and a wire is connected to each end of the winding, so you get two wires per winding as seen in fig 1.2

To convert our uni-polar motor into a bipolar one we opened the motor and scratched the internal connection connected with the red wire. This meant the center tap was gone

[1]Excerpt from Mr.Joshua Ruelle's project report

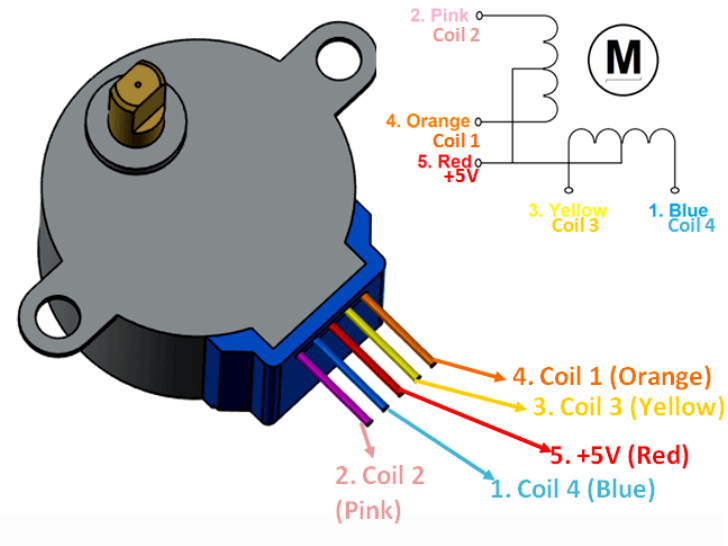


Figure 1.1: Unipolar motor, the color scheme of the cables is consistent with 28-BYJ48 motor

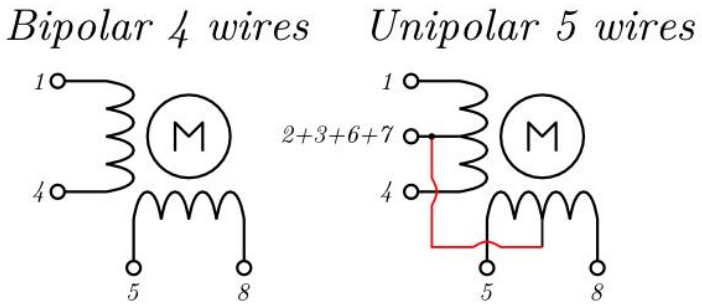


Figure 1.2: Difference between Unipolar and Bipolar motor

and the red wire became redundant, so we cut it off. Having no center tap, the motor was now a 2-phase bipolar motor with same mechanical characteristics as the uni-polar one[2]. It has 4096 steps per rotation and a step angle of 0.087890625°. To check the pair of wires belonging to same pole (winding), we use a multi-meter and measure when the resistance has finite value[3].

To operate this motor with the help of an Arduino, we used an L298N driver. It is dual DC motor driver but one can connect the bipolar motor wires to same and it works fine.

In the L298N motor driver as shown in the fig 1.3, instead of connecting two different motors in the output pins, we connect the pair of wires belonging to same winding to each of the output poles like one pair in OUT1 and OUT2 and another pair in OUT3 and OUT4 and with this fiddling it becomes a bipolar motor driver.

Finally, to give commands to the motor through the driver we use an Arduino Uno-3 board but unfortunately the board was rendered useless when I burnt it, so we went with

[2]To do this conversion, we followed this tutorial but one does not need to do the last step.

[3]For the above scheme of things, pink and orange wires belong to one pole and blue and yellow wires belong to another.

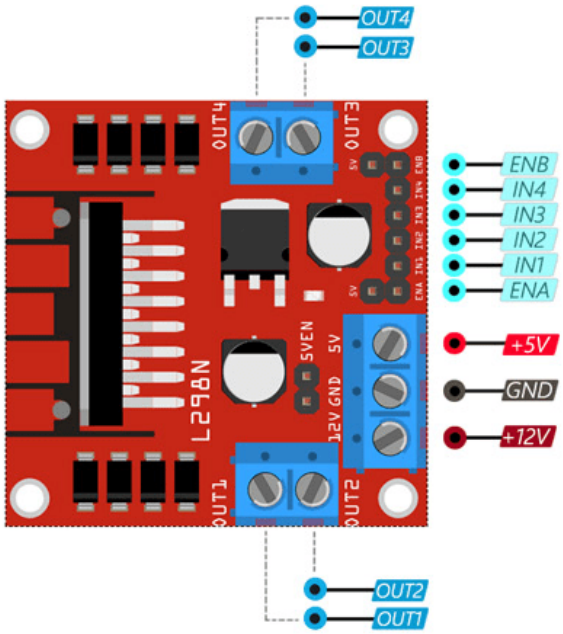


Figure 1.3: L298N motor driver

another board called Digilent Uno32 with same functionality as that of Arduino Uno-3. The board can be controlled through MPIDE rather than the normal Arduino IDE but the code syntax is similar, so the same code can be used to drive both Digilent Uno32 and Arduino Uno-3 to do the same task.

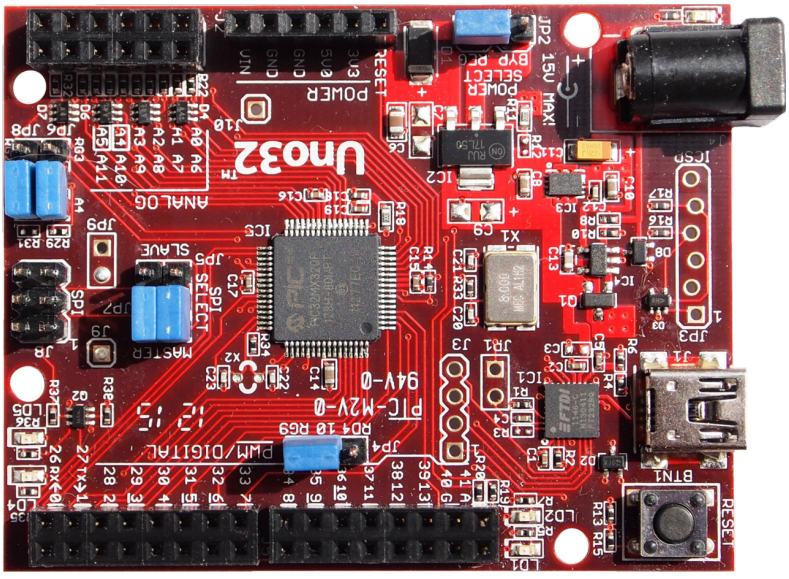


Figure 1.4: Digilent Uno32

1.3 Circuit Design

Fig 1.5 provides the visual representation of the circuit that has been implemented to control the stepper motor. We used a voltage regulator as well, in order to protect the board from burning. In the figure, an Arduino Uno3 board has been used while in practice we used Digilent Uno32[4], as stated earlier both the boards have similar characteristics, so the circuit is applicable for either of them as well as the code. These boards can be used interchangeably. In the fig 1.5, there are resistors present between the ground and input of switch, anything between 1-10 kΩ is fine but in our circuit we used 5 kΩ resistors.

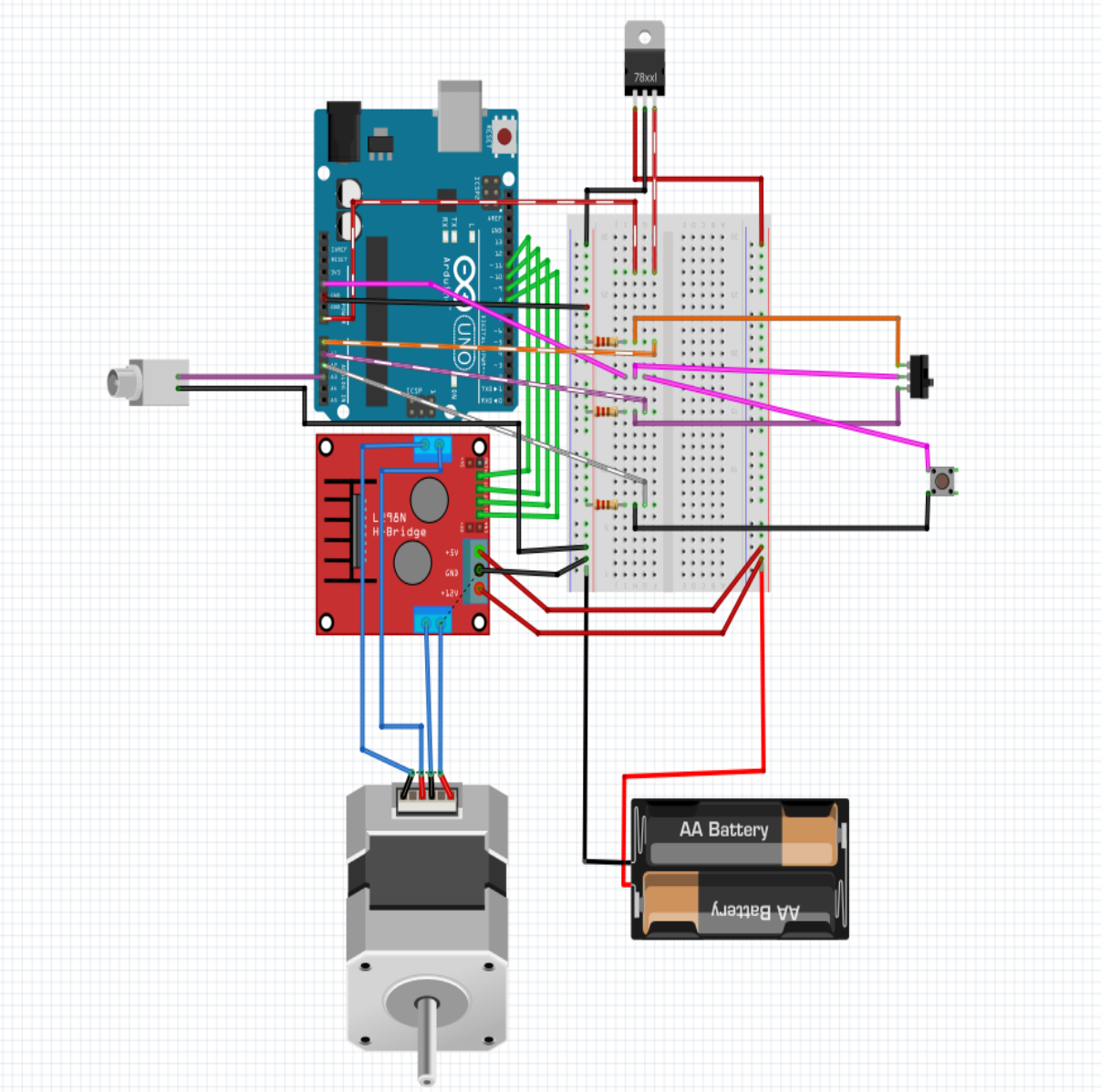


Figure 1.5: Visual representation of the Circuit

[4]Following another mishap, I burnt the L298N driver, after this incident, we went back to using Arduino Board not the Digilent one. This edit overwrites any mention of using Digilent board in the process

The circuit can be assembled as follows:

- Connect the two poles of motor (each with a pair of wires) to the A and B socket of L298N. What is important is that wires belonging to one pole are connected in same socket not the order in which they are connected.
- Connect pins 8,9,10,11 of the Arduino/Digilent board to the E1,M1,E2,M2 pins of the L298N motor driver respectively[5]
- Plug in the negative terminal of the 9V supply (e.g battery) to the negative of the breadboard/circuit board. Connect the ground of L298N and ground of Arduino/Digilent board to this.
- Plug in the positive terminal of the supply into the positive of the breadboard/circuit board.
- Connect the input pin of the voltage regulator to the positive of the battery(through the circuit board) and ground pin to the negative terminal of the supply. Connect the $\texttt{V}_{in}$ pin of the Arduino/Digilent board to the output pin of the voltage regulator.
- For the 3 step toggle switch, wire the middle connection to the `5V` pin of the Arduino/Digilent board.
- Connect one end of the resistor to the ground (through the circuit board) and another end to the breadboard/circuit board. In the same row, connect one end of the switch and then `A0` pin of the Arduino/Digilent board.
- Connect another resistor in the aforementioned manner. In the same row, connect another end of the switch and then `A1` pin of the Arduino/Digilent board.
- To connect the Push button, wire one of its end to the `5V` pin of the Arduino/Digilent board.
- Finally connect one of a resistor to the ground and another end to the breadboard/circuit board. In the same row, wire the second end of the push button and then `A2` pin of the Arduino/Digilent board.
- At last, connect the BNC connector to pin `A3` of the Arduino board and to the ground as well.

Note: The circuit given in fig 1.5 is exactly the one that has been implemented and it has been tested. So, if the figure seems a bit confusing, consult the steps.

[5]In some of the new motor drivers instead of E1,M1,E2,M2, there is IN1,IN2,IN3,IN4. In such a case, the order of connection remains the same, it does not change.

1.4 Code Details

As mentioned earlier, we use an Arduino Board to do the automation of the Oven rotator/door. The Arduino code used can be accessed **here**. The code has been well commented in order to make it easy to understand.

```
// Driving the Oven door bipolar motor

// Motor pins
const int E1 = 8;
const int M1 = 9;
const int E2 = 10;
const int M2 = 11;

// Number of steps per revolution
// The mounted motor has 200 steps per revolution and we do 4 mini-steps in one step_rev

const int step_rev = 50;
const int step45 = 6;  // 43.2 degrees
const int step25 = step_rev /50; // 7.2 degrees
const int stepDelay = 10;  // a delay of 10 ms between the mini-steps

// Switch pins
const int switchPinA0 = A0;   // Toggle switch connection
const int switchPinA1 = A1;  // Another Toggle switch connection
const int reset_pin = A2;   // Press Button Connection
const int controlPinA3 = A3; // TTL logic connection

int curr_pos = 0;  // Track the current step position
```

Figure 1.6: Arduino code to drive the motor, pg1

The snippet of the code provided in fig.1.6 is used to declare various parameters used in the code, the motor pins to drive the motor and the switch pins to control the rotation of the motor have been declared along with the various turn types we want the motor to do.

```
void setup() {
  // Set the pin modes
  pinMode(M1, OUTPUT);
  pinMode(E1, OUTPUT);
  pinMode(M2, OUTPUT);
  pinMode(E2, OUTPUT);

  Serial.begin(9600);

  // Initialize the switch pins as input
  pinMode(switchPinA0, INPUT);
  pinMode(switchPinA1, INPUT);
  pinMode(reset_pin, INPUT);
  pinMode(controlPinA3, INPUT);


}
```

Figure 1.7: Arduino code to drive the motor, pg2

This snippet, i.e fig1.7 sets up the functions of the various pins(already declared above) as input or output and also starts the serial monitor, which is essential for debugging.

```
void loop() {

  // Read the switch states
  bool switchStateA0 = digitalRead(switchPinA0);
  bool switchStateA1 = digitalRead(switchPinA1);
  bool resetButtonState = digitalRead(reset_pin);
  int control_vol = analogRead(controlPinA3);

  // Convert analog reading to voltage
  // Do not use "map" function to convert the voltage in your preferred range
  float vol_converted = control_vol * (5.0 / 1023.0);

  // To Debug
  Serial.print("Switch State A0: ");
  Serial.print(switchStateA0);

  Serial.print(" Switch State A1: ");
  Serial.print(switchStateA1);

  Serial.print(" Reset Button State: ");
  Serial.print(resetButtonState);

  Serial.print(" Control Voltage: ");
  Serial.println(vol_converted);

  Serial.print("Current Position: ");
  Serial.println(curr_pos);

  // Primary logic for switch control
```

Figure 1.8: Arduino code to drive the motor, pg3

After the `setup()` function, which initializes the values, the `loop` function loops consecutively, allowing the control of the Arduino board. Here in fig.1.8, we specify the input pins either as digital (for switches) or analog (for TTL) as well as print the their status on the serial monitor.

```
// Primary logic for switch control
if (resetButtonState == LOW) {
  if (switchStateA0 == HIGH && switchStateA1 == LOW && curr_pos == 0) {
    rotateMotor(step45, true);  // 43.2 degrees clockwise
    curr_pos = 1;  // Update the current position
  } else if (switchStateA1 == HIGH && switchStateA0 == LOW && curr_pos == 1) {
    rotateMotor(step45, false);  // 43.2 degrees counterclockwise
    curr_pos = 0;  // Update the current position
  } else if (switchStateA0 == LOW && switchStateA1 == LOW) {
    if (vol_converted > 2.0 && curr_pos == 0) {
      rotateMotor(step45, true);  // 43.2 degrees clockwise
      curr_pos = 1;  // Update the current position
    } else if (vol_converted <= 2.0 && curr_pos == 1) {
      rotateMotor(step45, false);  // 43.2 degrees counterclockwise
      curr_pos = 0;  // Update the current position
    }
  }
}

// Rotate the motor 7.2 degrees when the reset button is pressed
if (resetButtonState == HIGH) {
  rotateMotor(step25, true);  // Rotate 7.2 degrees
}

delay(100);
}
```

Figure 1.9: Arduino code to drive the motor, pg4

The code provided in the fig 1.9 contains the main logic for motor rotation. If the press button is pressed, it will only rotate by a predefined small angle (7.2 degrees in our case) and if it not pressed, depending on the state of the toggle switch, the motor will rotate 45^o clockwise or anticlockwise[6]. If the toggle switch is off and the TTL logic in ON, the motor rotates clockwise or anticlockwise by same amount depending on the input voltage(greater or lesser than 2 V).

```
// rotates the motor according to the boolean value of clockwise variable
// "How to rotate" is conveyed by "stepMotor" function
// rotates till n steps
void rotateMotor(int steps, bool clockwise) {
  for (int i = 0; i < steps; i++) {
    stepMotor(clockwise);
  }
}

// Tells the motor on how to use the mini-steps in order
// informs the motor to either move in clockwise or anti-clockwise direction
void stepMotor(bool clockwise) {
  if (clockwise) {
    step1();
    delay(stepDelay); // Adjust the delay to control the speed
    step2();
    delay(stepDelay);
    step3();
    delay(stepDelay);
    step4();
    delay(stepDelay);
  } else {
    step4();
    delay(stepDelay);
    step3();
    delay(stepDelay);
    step2();
    delay(stepDelay);
    step1();
    delay(stepDelay);
  }
}
```

Figure 1.10: Arduino code to drive the motor, pg5

The code provided in the fig 1.10 contains the functions neccessary to rotate the motor by a desired angle. The `rotatemotor` function rotates the motor for n given steps depending on the boolean value of the variable `clockwise` according to another function `stepMotor`. This `stepMotor` function drives the motor either clockwise or anti-clockwise.

The code provided in the fig 1.11 gives the driving scheme of the bipolar motor. There are other driving schemes as well but this one has an advantage that it draws very small current, so prevents the equipment from over heating.

[6]In the motor mounted on the oven the actual rotation is 43.2^o, it is because of the resolution of the motor and how we drive the motor in the code

```
// Steps to drive the motor
// each one of them is an individual step, what we have here cal
// It provides the logic on how to supply output to pins so they
void step1() {
  digitalWrite(E1, HIGH);
  digitalWrite(M1, HIGH);
  digitalWrite(E2, LOW);
  digitalWrite(M2, LOW);

}

void step2() {
  digitalWrite(E1, LOW);
  digitalWrite(M1, LOW);
  digitalWrite(E2, HIGH);
  digitalWrite(M2, HIGH);
}

void step3() {
  digitalWrite(E1, HIGH);
  digitalWrite(M1, LOW);
  digitalWrite(E2, LOW);
  digitalWrite(M2, LOW);
}

void step4() {
  digitalWrite(E1, LOW);
  digitalWrite(M1, HIGH);
  digitalWrite(E2, HIGH);
  digitalWrite(M2, LOW);
}
```

Figure 1.11: Arduino code to drive the motor, pg6

1.5 Oven motor specifications and connections

The oven door is controlled by $BRM-275-03 == 670002-03$ bipolar stepper motor. It has an angular resolution of 1.8^o, thus 200 steps per revolution[7]. It can go upto a top speed of 20 rpm. Since, it is a bipolar motor along with the choice of our driving scheme, the polarity of the wiring is important. If the wiring is wrong the motor won't rotate, it will have a vibratory motion.

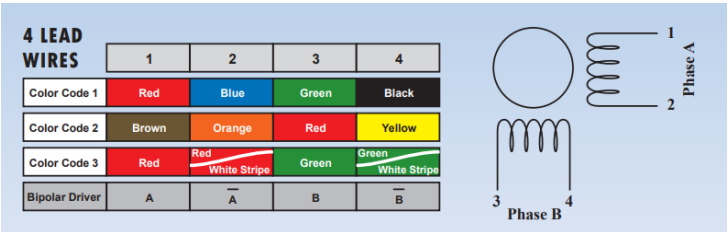


Figure 1.12: Color code of the bipolar motor wiring

[7]Since the documentation on this motor was not extensive, we confirmed this manually with various steps,also Mr. Joshua Ruelle had given this value in his report

The motor mounted on the oven uses the `color code 1` as given in fig 1.12 and our motor driver has two poles A and B. One of the poles has white (+) and blue (-) wires connected and another one has red (+) and orange (-) wires which have been connected to the motor respecting the polarity in following sequence from driver to motor.

- $White --- > Red$
- $Blue --- > Blue$
- $Red --- > Green$
- $Orange --- > Black$

1.6 Conclusion

Our main goal was the automatic control of the oven door opening and closing with the help of Arduino board, which we succeeded in achieving. We used an L298N dual dc motor driver for our purpose. The motor has manual switches to control the movement as well as the TTL logic input. It rotates 43.2^o clockwise and anticlockwise, its motion can also be controlled with the help of a push button, which moves 7.2^o with each press. If the input to the TTL logic is greater than 2 V, the motor moves clockwise and if is lower than 2 V, it moves anticlockwise. Thus, the oven door can now be operated either automatically with the help of various buttons or remotely through the TTL logic input.

2 Appendix

The various other electrical components used in the circuit are:
As mentioned earlier, I changed the Digilent Uno32 for Arduino Uno3 afterwards when everything was done. They are practically similar.

1. Arduino Uno R3

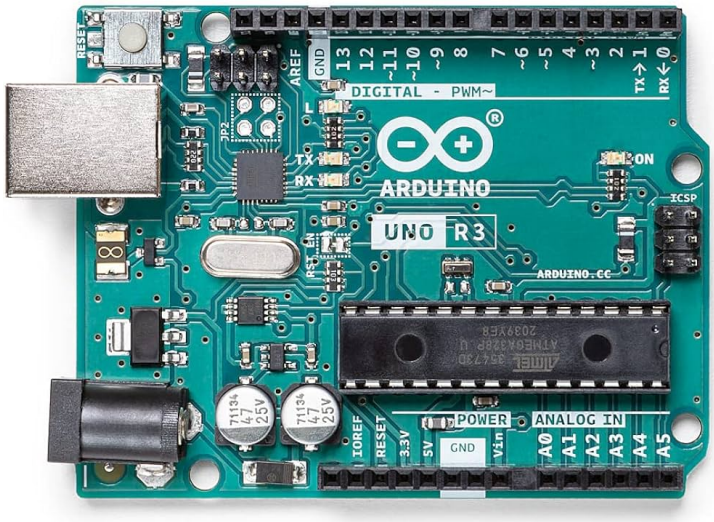


Figure 2.1: Arduino Uno R3 Board

2. Voltage Regulator

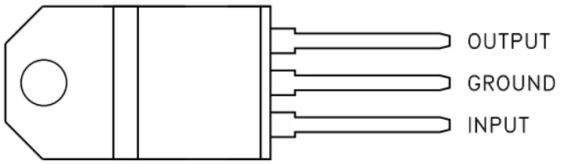


Figure 2.2: Voltage regulator

The fig.2.2 shows a typical voltage regulator, the middle pin is usually the ground pin but it is not always the case.

3. Push Button

Figure 2.3: Push button

4. Toggle Switch

Figure 2.4: Toggle switch 3 pin

5. BNC female connector

Figure 2.5: A female BNC connector

Conversion of unipolar motor into a bipolar one

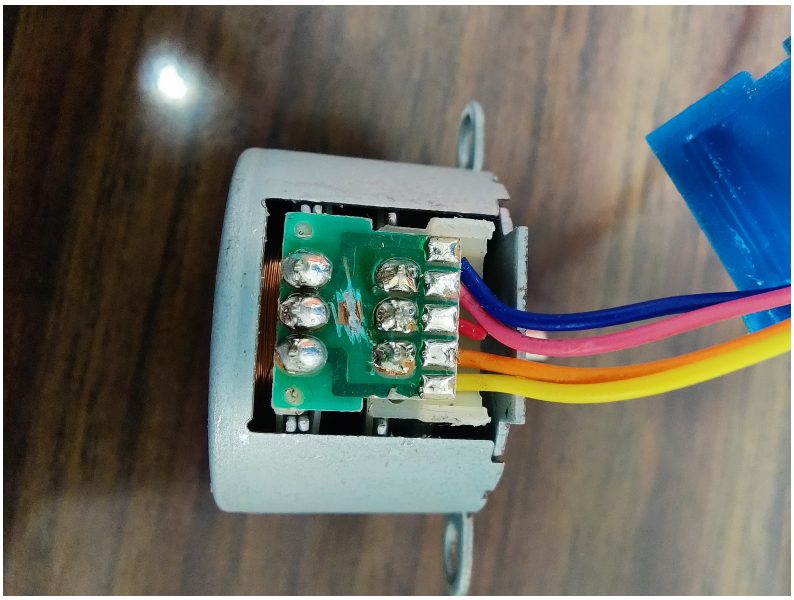


Figure 2.6: Bipolar motor

The fig 1.2 shows the conversion of 28-BYJ48 unipolar motor into a bipolar one. Scratching and removing the copper of the center wire cuts the center tap, effectively turning this motor into a bipolar motor without changing the steps per revolution of the motor.