Polytechnic University of Puerto Rico
San Juan Campus

Electrical and Computer Engineering Faculty

Microprocessor Interfacing Laboratory

Laboratory 6: Stepper Motor

Kelvin Figueroa Figueroa - 108946
Rafael Gonzalez Cartagena - 90352

Professor: Dr. Roman Lopez

May 4, 2022

Contents

List of Tables

List of Figures

1 Objective

This laboratory focuses on turning on and using a stepper motor and an arduino mega-ADK to perform certain functions. One of the functions introduced in this report is to make the motor turn sideways by writing a sequence of characters to the serial monitor.

It is important to mention that the stepper motors have no polarity, in other words they are unipolar, refer to the figure 1. These motors have three coil connections and have a center tap on each coil.

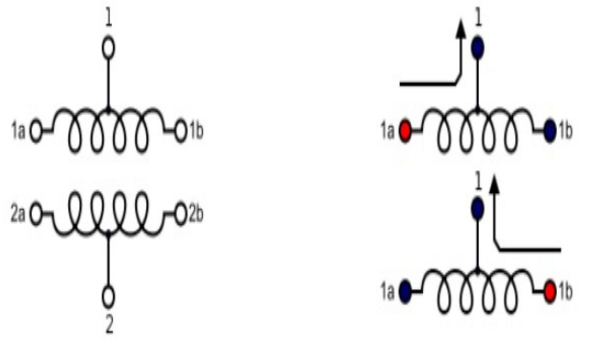


Figure 1: Stepper Motor Unipolar Schematic

Figure 2: Stepper Motor and module

In order to make this laboratory possible, make sure to have the following materials,

1. Arduino Mega ADK
2. Stepper motor and module
3. Wiring Cables

2 Procedures

This lab consists of several parts. The first part is to make the stepper motor rotate clockwise by using the character sequence "CW" and counter clockwise by using the character sequence "CC", refer to sub-section 2.1. The second part consists of making the motor rotate continuously, clockwise "CCW NUMBER" or counter clockwise "CCC NUMBER", where "NUMBER" is an input given by the user, refer to sub-section 2.2. The third part consists of making the motor rotate, right "CSW NUMBER" or left "CSC NUMBER", where "NUMBER" is the number of times to rotate according to the first part of the process, refer to sub-section 2.3. For the last part a "STOP" is implemented so that the stepper stops the rotations instantly and a function "STP NUMBER", where the motor rotates in the desired direction, depending on the number of values entered in the variable "NUMBER", refer to sub-section 2.4.

See table 2 to see the Arduino's Pinout

Table 1: Arduino Pinout

Arduino Mega	**Stepper module**
5V	VCC
GND	GND
Pin 22	IN1
Pin 23	IN2
Pin 24	IN3
Pin 25	IN4

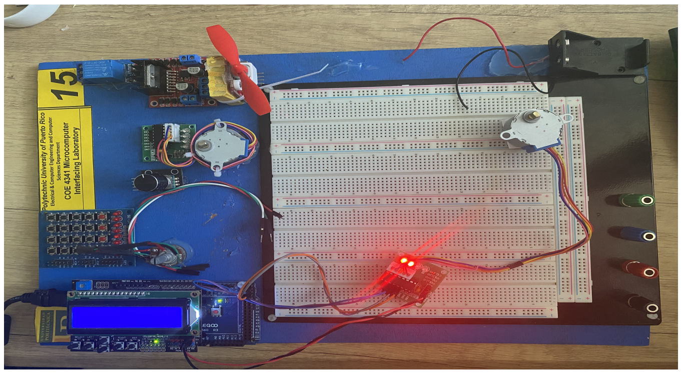


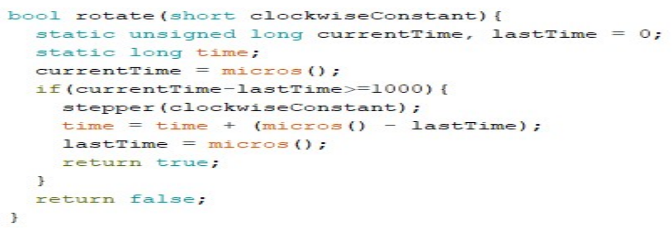
Figure 3: Circuit Design

Table 2: Stepper Functions

Keyboard	**Stepper Function**
CW	Rotate clock wise until an esc key on your keyboard is pressed
CC	Arduino Pin 13
CCW & number of turns	Power Supply VCC
CCC & number of turns	Arduino and Power Supply GND
CSW	Move a number of steps CCW
CSC	Move a number of steps CW
STP, ,No of steps	If a command of STP are typed the microcontroller must asks for the steps number, after that the motor will run the number of steps
STOP	Stop the stepper (a zoon you enter this command)

Before proceeding with the parts, first the following function were added,

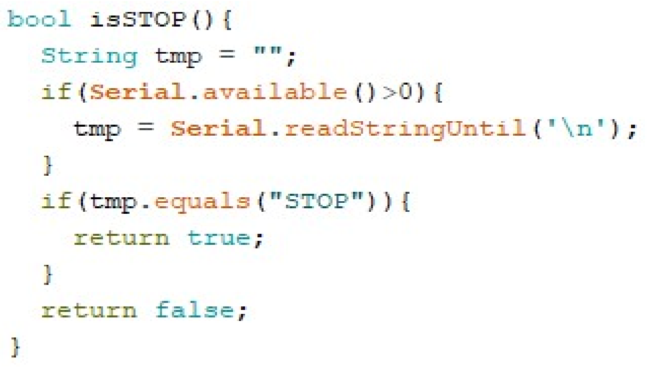
1. Rotate, This function enables the rotation of the stepper motor for one step in a particular direction. This fuction has a parameter "clockwiseConstant, where when is "-1", the motor rotates in clockwise direction.



```
bool rotate(short clockwiseConstant){
  static unsigned long currentTime, lastTime = 0;
  static long time;
  currentTime = micros();
  if(currentTime-lastTime>=1000){
    stepper(clockwiseConstant);
    time = time + (micros() - lastTime);
    lastTime = micros();
    return true;
  }
  return false;
}
```

Figure 4: Rotate function for Stepper motor

2. isSTOP, This functin checks if "STOP" was written on the serial monitor and if the condition is met, the function returns TRUE.



```
bool isSTOP(){
  String tmp = "";
  if(Serial.available()>0){
    tmp = Serial.readStringUntil('\n');
  }
  if(tmp.equals("STOP")){
    return true;
  }
  return false;
}
```

Figure 5: isStop function for Stepper motor

2.1 CW and CC functions

In this part the first two functions were implemented, which make the stepper rotate to the right using a "CW" command or to the left using a "CC" command.

To implement these functions, first the serial input had to be read with the internal function "Serial.readStringUntil(ENTER)", from which the input value is compared with the command to be used.

1. The first IF statement is implemented, in which if the user enters "CW", there will be a "do while" statement that calls the function "rotate(-1)" making the to motor rotate clockwise.

2. Similar to the first implementation, an ELSE IF statement is created, in which if the user enters "CC", there will be a "do while" statement that calls the function "rotate(1)" making the motor to rotate counter-clockwise.

2.2 CCW and CCC functions

In this part we are trying to implement the functions "CCW NUMBER" and "CCC NUMBER", where the code waits for an input with an integer value, this being a value of rotations for the stepper motor.

1. An ELSE IF is used to implement the function "CCW NUMBER", where the input given by the user is read and if the condition is met two for loops are created to perform the right rotation of the stepper motor.
2. Another ELSE IF is created to implement the function "CCC NUMBER", where the input given by the user is read and if the condition is met two for loops are created to perform the left rotation of the stepper motor.

2.3 CSW and CSC functions

This part implements the functions "CSC NUMBER" and "CSW NUMBER", where "CSC" moves a certain number of steps in "CW" and "CSW" moves a certain number of steps in "CCW" (continuous).

1. It is used an ELSE IF as a continuation of the previous functions, where the condition is met when the input is "CSC", the stepper motor should rotate for a definite number of steps clockwise.
2. It is used an ELSE IF as a continuation of the previous functions, where the condition is met when the input is "CSW", the stepper motor should rotate for a definite number of steps counter clockwise.

2.4 STOP and STP functions

For this last part, it is intended to stop the stepper motor with the entered command "STOP" and it is also intended to implement the command

"STP NUMBER" so that the motor rotates the number of numbers entered (NUMBER).

1. An ELSE IF statement is used for the "STP NUMBER" function, where if the condition is met, a "For loop" is used to make the motor rotate in any direction.
2. For the "STOP", the last ELSE statement is used, where only a "goto exit" is written to stop the stepper at the time of input.

It should be noted that the descriptions of all the steps provided in this section were obtained with the laboratory's documentation [1].

3 Results

This section documents the various results by undertaking the already described procedure associated with *Laboratory 6: Stepper Motor*.

3.1 The Developed C Embedded Program

The following **Stepper Motor Program** was developed for the realization of the already described experiment.

Stepper Motor Program

```
const int IN1 = 22;                                                 // Pin IN1 of Stepper Motor.
const int IN2 = 23;                                                 // Pin IN2 of Stepper Motor.
const int IN3 = 24;                                                 // Pin IN3 of Stepper Motor.
const int IN4 = 25;                                                 // Pin IN4 of Stepper Motor.
const int MAXSTEPS = 4096;                                          // Maximum value of steps for one (1) Stepper Motor's rotation.

```

```
String command = "";                                                    // Variable to hold a string which will represent a command in order to control the motor.
short iStep = 0;                                                         // Variable to hold the actual number of steps that the mottor has done for a particular moment in time.

// 1 -> counterclockwise and -1 -> clockwise

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  Serial.begin(9600);

  Serial.println("STEPPER ROTATION PROGRAM");
  Serial.println("------------------------");
  Serial.println("This program uses the following series of commands to control a STEPPER MOTOR:");
  Serial.println("1. CW: Specifies the motor to rotate in clocwise fashion indefinitely.\nUSAGE EXAMPLE: CW\n");
  Serial.println("2. CC: Specifies the motor to rotate in counter-clocwise fashion indefinitely.\nUSAGE EXAMPLE: CC\n");
  Serial.println("3. STP: Specifies the motor to rotate for a n number of steps. Furthermore these steps can be either positive or negative.\nUSAGE EXAMPLE: STP 1000\n");
  Serial.println("4. CCC: Sets the motor to rotate clockwise for a n number of times.\nUSAGE EXAMPLE: CCC 10\n");
  Serial.println("5. CCW: Sets the motor to rotate counter-clockwise for a n number of times.\nUSAGE EXAMPLE: CCW 20\n");
  Serial.println("6. CSC: Sets the motor to rotate clockwise for a n amount of steps.\nUSAGE EXAMPLE: CSC 2048\n");
  Serial.println("7. CSW: Sets the motor to rotate counter-clockwise for a n amount of steps.\nUSAGE
      EXAMPLE: CSW 10000\n");
  Serial.println("8. STOP: Interrupts the program.\nUSAGE
      EXAMPLE: STOP\n");
}

void loop() {
  if(Serial.available()>0){
      // Condition to check if the is input in the Serial.
    terminalMode();
        // Function to control the motor given the input.
  }
}

void terminalMode(){
  command = Serial.readStringUntil('\n');
      // Assign the contents of the Serial input to varible
      "command".
  Serial.flush();

  if(command.equals("CW")){
      // If command = "CW", then the stepper motor will rotate
      in clockwise direction until another key(s) is pressed.
    do{
      rotate(-1);
          // Function to rotate Stepper Motor.
    }while(Serial.available()<=0);
  }

  else if(command.equals("CC")){
      // If command = "CC", then the stepper motor will rotate
      in counterclockwise direction until another key(s) is
      pressed.
    do{
      rotate(1);
    }while(Serial.available()<=0);
  }

  else if(command.substring(0,3).equals("STP")){
      // If command = "STP", then the stepper motor will rotate
      for a definite number of steps in any direction.
    long int realSteps = abs(command.substring(3).toInt());
      // Example Usage: STP 4096 should rotate motor 360
      degrees one time clockwise. If value is negative,
    short k = command.substring(3).toInt() / realSteps;
      // the opposite is true.
    for(int i = 0; i < realSteps;){
      // Anyhow: realSteps is the magnitude of the # of
      steps, and k will decide the direction of rotation.
     if(rotate(-1*k)){
      i++;
     }
     if(isSTOP()){
      goto exit;
     }
    }
   }
   else if(command.substring(0,3).equals("CCC")){
     // If command = "CCC", then the stepper motor will rotate
     for a definite number of times clockwise.
    long int realRots = abs(command.substring(3).toInt());
      // Example Usage: CCC 3 will make the motor rotate 3
      times in clockwise direction.
    for(long int i = 0; i < realRots; i++){
      // realRots is the magnitude of the # of rotations. As
      with realSteps, realRots is an absolute value
     for(int j = 0; j < MAXSTEPS;){
       // in order to deal with negative (erronous) values.
      if(rotate(-1)){
       j++;
      }
      if(isSTOP()){
       goto exit;
      }
     }
    }
   }
   else if(command.substring(0,3).equals("CCW")){
     // If command = "CCW", then the stepper motor will rotate
     for a definite number of times counter-clockwise.
    long int realRots = abs(command.substring(3).toInt());
```

```
  for(long int i = 0; i < realRots; i++){
    for(int j = 0; j < MAXSTEPS;){
      if(rotate(1)){
        j++;
      }
      if(isSTOP()){
        goto exit;
      }
    }
  }
}
else if(command.substring(0,3).equals("CSC")){
    // If command = "CSC", then the stepper motor should rotate for a definite number of steps clockwise.
  long int realSteps = abs(command.substring(3).toInt());
  for (long int i = 0; i < realSteps;){
    if(rotate(-1)){
      i++;
    }
    if(isSTOP()){
      goto exit;
    }
  }
}
else if(command.substring(0,3).equals("CSW")){
    // If command = "CSW", then the motor should rotate for a definite number of steps counter-clockwise.
  long int realSteps = abs(command.substring(3).toInt());
  for (long int i = 0; i <realSteps;){
    if(rotate(1)){
      i++;
    }
    if(isSTOP()){
      goto exit;
    }
  }
}
else if(command.equals("STOP")){
  goto exit;
}
```

```

  exit:
  Serial.flush();
}

bool isSTOP(){
    // Function to check if "STOP" was wriiten on the Serial
     port.
  String tmp = "";
      // tmp variable for string of Serial port input
  if(Serial.available()>0){
      // If condition to check wether there is information on
       Serial port, if so read such information until '\n' is
       found.
    tmp = Serial.readStringUntil('\n');
  }
  if(tmp.equals("STOP")){
      // If "STOP" is what was written on the Serial port,
       return true.
    return true;
  }
  return false;
      // Otherwise, resturn false.
}
bool rotate(short clockwiseConstant){
    // Function to enable the rotation of the stepper motor for
     one step in a particular direction.
  static unsigned long currentTime, lastTime = 0;
      // If parameter clockwiseConstant is '-1', the motor will
       rotate in clockwise direction; the opposite
  static long time;
      // is true.
  currentTime = micros();
      // It requires timming variables (currentTime, lastTime,
       time) in order to rotate.
  if(currentTime-lastTime>=1000){
      // It returns true if the motor is able to rotate.
    stepper(clockwiseConstant);
    time = time + (micros() - lastTime);
    lastTime = micros();
    return true;
  }
  return false;
}

void stepper(short clockwiseConstant){
    // Function physically control the steps (rotation) of the
    stepper motor.
  iStep = iStep + clockwiseConstant;
  if(iStep < 0 && clockwiseConstant == -1){
    iStep = 7;
  }
  if(iStep > 7 && clockwiseConstant == 1){
    iStep = 0;
  }
  switch(iStep){
    case 0:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    break;
    case 1:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, HIGH);
    break;
    case 2:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    break;
    case 3:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    break;
    case 4:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    break;
    case 5:
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    break;
    case 6:
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    break;
    case 7:
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    break;
    default:
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    break;
  }
}
```

It is imperative to note that this program does not contain any means for data type validation for commands which need a parameter, like: **CCC** or **CCW**! Henceforth, it is believed if, for instance, the command **CCC xgsh** is used, the program will crash. With that being said, various of the resources utilized for the creation of the already described program were found in [1]

3.2 The Videos of the Results

In relation with the previous subsection, the following table, i.e., **Table 2**, illustrates the names of the video files related with the already illustrated *C Embedded* program:

Table 3: Video Names of Laboratory Results

Video Names of Laboratory Results
SM_1

4 Analysis of Results

For the completion of this experiment, the following caveats were found in regard to both the the *Stepper Motor* and *C Embedded Program*:

- In order for the *Stepper Motor* to make one (1) rotation, **4096** steps are needed.
- It was not possible to identify its cause; however, the *Stepper Motor*, after some resets, activates but does not rotate with the usage of the CC command, it instead vibrates. This is not the case for most occasions, but, as stated, it tends to happen from time to time.
- It was found to be possible to implement a program which asks for inputs in the type of **Linux Terminal Commands**. This was achieved with the usage of the String Library.
- A viable usage of **goto** was found; that is, of jumping out of loops without the need of more complex **If Statements** and **breaks** for the interruption of the program whenever the *STOP* command is used. This realization comes from the fact that *C Embedded* does not have **try / catch** functionality and that some of the loops used were **nested**.
- The **Serial Library** does not contain any function related with the deletion of all the text in the Arduino's IDE **Serial Monitor**.

5 Conclusion

In conclusion, it was possible to construct and control a Stepper Motor with the aid of both a *C Embedded* program and an *Arduino Mega2560 ADK Board* for various usages regarding the number of its rotations and the direction of such.

References

The following *References* related with this document are in the IEEE Format, as shown below:

[1] Roman Lopez, Phd., "Laboratory 6: Stepper Motor". Mega. 2022. [PDF]
[2] Arduino, “Language reference,” Arduino Reference - Arduino Reference, 2022. [Online]. Available: https://www.arduino.cc/reference/en/. [Accessed: 30-Apr-2022].