DRIVES IN AUTOMATION

Control speed of single-phase induction motor

Group 10
Pham Thanh Quang (10221073)
Le Quoc Dinh (10221009)
Tran Hong Vu (10221049)
Pham Nam Nhat (10221036)
Hsu Yu Yuan (10221051)

Supervisor: Bui Minh Duong

Contents

Chapter 1

Introduction

This project involves using an AC fan motor to create a handheld vacuum cleaner. A fan is a device with rotating blades that generates airflow, typically used for ventilation and cooling. The AC motor in this application converts electrical energy into rotational mechanical energy. AC motors are advantageous because they provide reliable performance and can deliver powerful suction for effective cleaning. Users can adjust the vacuum cleaner's suction level, enhancing convenience and energy efficiency. Moreover, the relatively high power-to-weight ratio of AC motors makes them ideal for compact designs, contributing to the portability and usability of the handheld vacuum cleaner.

This experiment investigates the characteristics of a single-phase induction motor and its application as a vacuum cleaner.

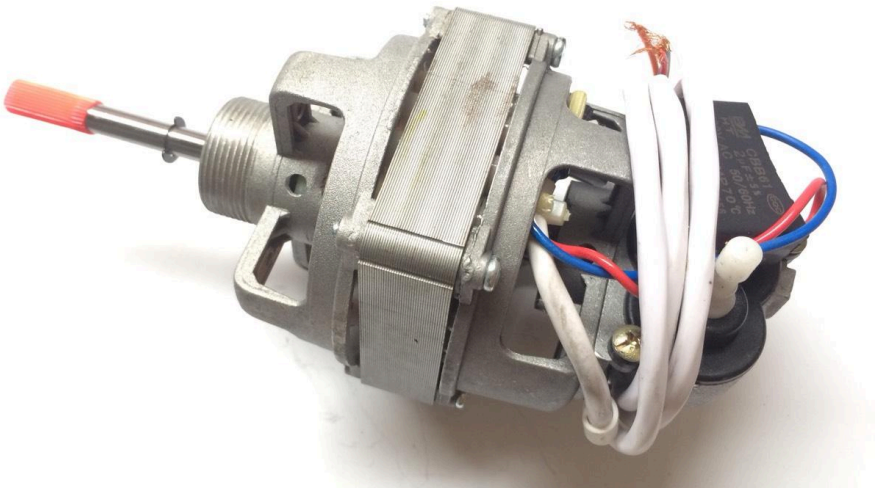


Figure 1: AC fan motor

Chapter 2

Theoretical Background

2.1 Operating Principles of Motor

An AC induction motor operates on the principle of electromagnetic induction, utilizing alternating current to induce a rotating magnetic field in the stator windings. When AC voltage is applied to the stator windings, it generates a magnetic field that rotates at the supply frequency (e.g., 50 or 60 Hz). This rotating magnetic field interacts with the conductive rotor bars or windings, inducing currents in them through electromagnetic induction. These induced currents in the rotor then interact with the stator's magnetic field, producing torque that causes the rotor to rotate. The rotating rotor, in turn, drives the mechanical load coupled to the motor, such as pumps, compressors, or conveyor belts.

AC induction motors are robust and widely used due to their simple design, reliability, and ability to provide consistent torque across a range of speeds without the need of external excitation. They are key components in industrial and commercial applications where continuous and efficient operation is essential.

The power value is obtained by using the formula:

$$P = V.I.cos(\phi)$$

The torque value is obtained by using the formula:

$$T = \frac{P}{\omega}$$

The angular speed of a single-phase induction motor is:

$$\omega = \frac{2\pi n}{60}$$

2.2 Characteristics of a Single-Phase Induction Motor

2.2.1 Torque-Speed Characteristic

- The induced torque of the motor is zero at synchronous speed.
- The torque-speed curve is nearly linear between no load and full load. In this range, the rotor resistance is much larger than the rotor reactance, so the rotor current, the rotor magnetic field, and the induced torque increase linearly with increasing slip.
- There is a maximum possible torque that cannot be exceeded. This torque, called the pullout torque or breakdown torque, is 2 to 3 times the rated full-load torque of the motor.
- The starting torque on the motor is slightly larger than its full-load torque, so this motor will start carrying any load that it can supply at full power.
- The torque on the motor for a given slip varies as the square of the applied voltage.
- If the rotor of the induction motor is driven faster than synchronous speed, the machine becomes a generator, converting mechanical power to electric power.

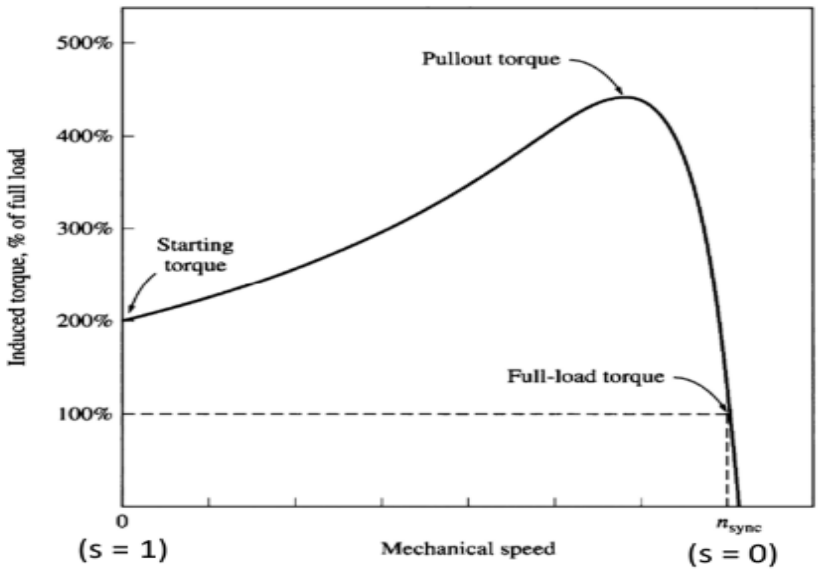


Figure 2.1: A typical induction motor torque-speed characteristic curve.

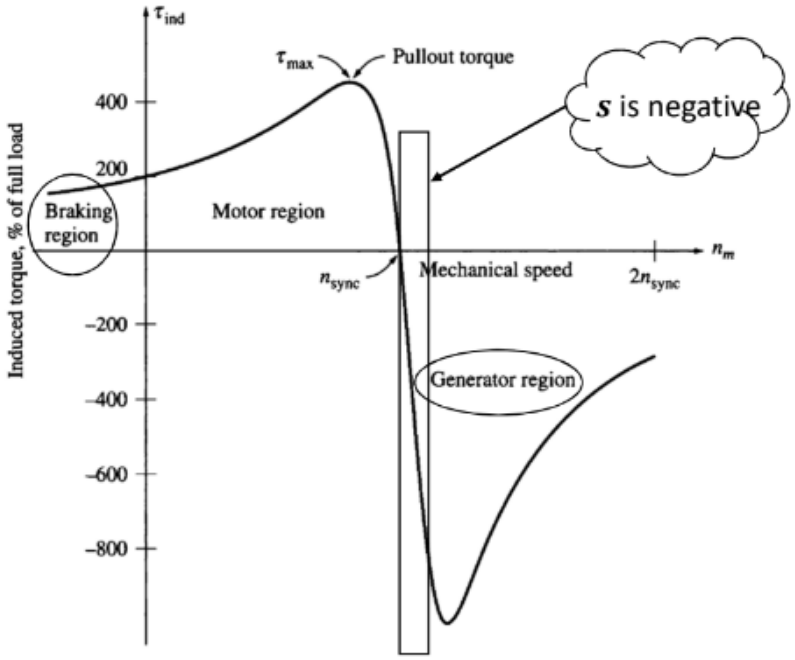


Figure 2.2: Induction motor torque-speed characteristic curve, showing the extended operating ranges (braking region and generator region)

2.2.2 Capacitor-Start Characteristic

- The capacitor-start induction motor develops a very high starting torque about 3 to 4.5 times of the full-load torque.
- The value of the starting capacitor must be large and the starting winding resistance low to obtain a starting torque.
- The capacitor-start induction motor requires the starting capacitor of high VAR rating, thus, the electrostatic capacitors of the order of 250 F are used.
- The starting capacitor is a short-time rated capacitor.
- The capacitor-start induction motors possess good starting and running characteristics.
- For a capacitor-start induction motor, the starting current is small and the starting torque is high. Therefore, starting winding of a capacitor-start induction

motor heats up less quickly and is well suited to the application where the starting period is either small or prolonged.

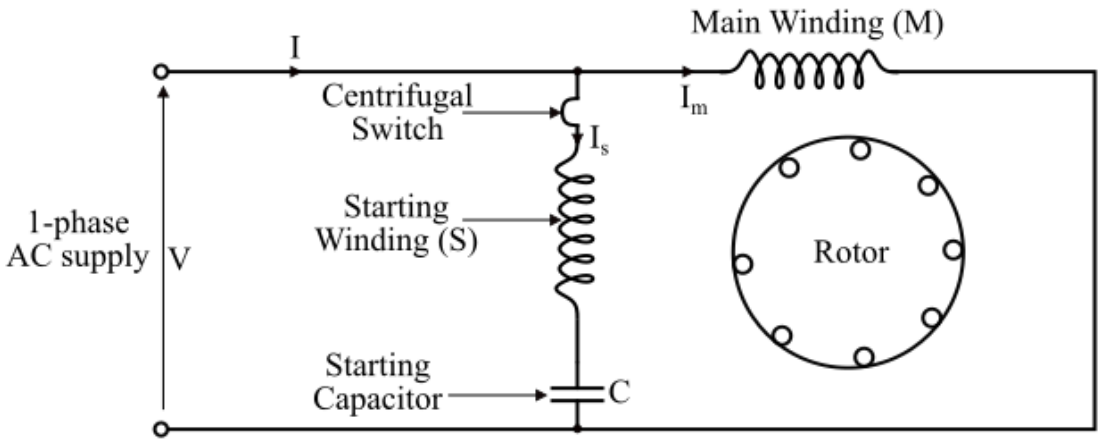


Figure 2.3: A circuit diagram of capacitor-start induction motor

2.3 Dimmer

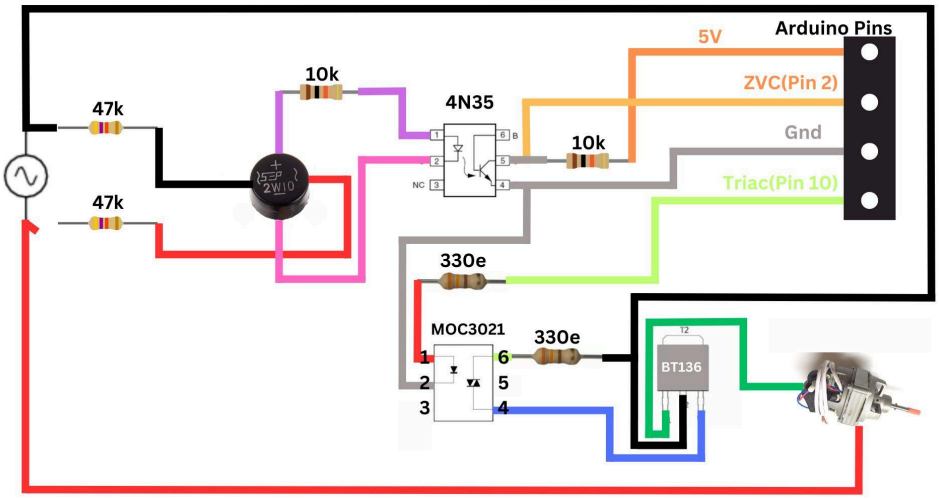


Figure 2.4: Circuit diagram of dimmer

2.3.1. Zero-Crossing Detector

The AC supply we get in our household is 220V-AC RMS, 50 HZ. This AC alternating signal goes through the rectifier to create a new sine wave without the negative side. In the first half of every cycle, it flows in one direction reaching a peak voltage, and then decreases down to zero and repeats. To control the AC Fan's speed, the peak voltage of each half-cycle needs to be chopped and controlled. For this, we need to detect the zero point from which the signal is to be controlled/Chopped. This point on the voltage curve where the voltage changes direction is called zero voltage crossing.

The circuit shown below (Fig. 2.5) is the zero-crossing detector circuit, which is used to get the zero-crossing point. First, the 220V-AC voltage is stepped down by resistors, and it is then fed to a 4N35 optocoupler at pins 1 and 2. The 4N25 optocoupler has an inbuilt LED with pin 1 as an anode and pin 2 as a cathode. So as per the circuit below, when the AC wave goes closer to the zero-crossing point, the inbuilt LED of 4N25 will get turned off, and as a result, the output transistor of 4N25 will also get turned OFF and the output pulse pin will get pulled up to 5V. Similarly, when the signal increases gradually to the peak point, then the LED turns ON and the transistor will also turn ON with the ground pin connected to the output pin, which makes this pin 0V. Using this pulse, the zero-crossing point can be detected using Arduino.

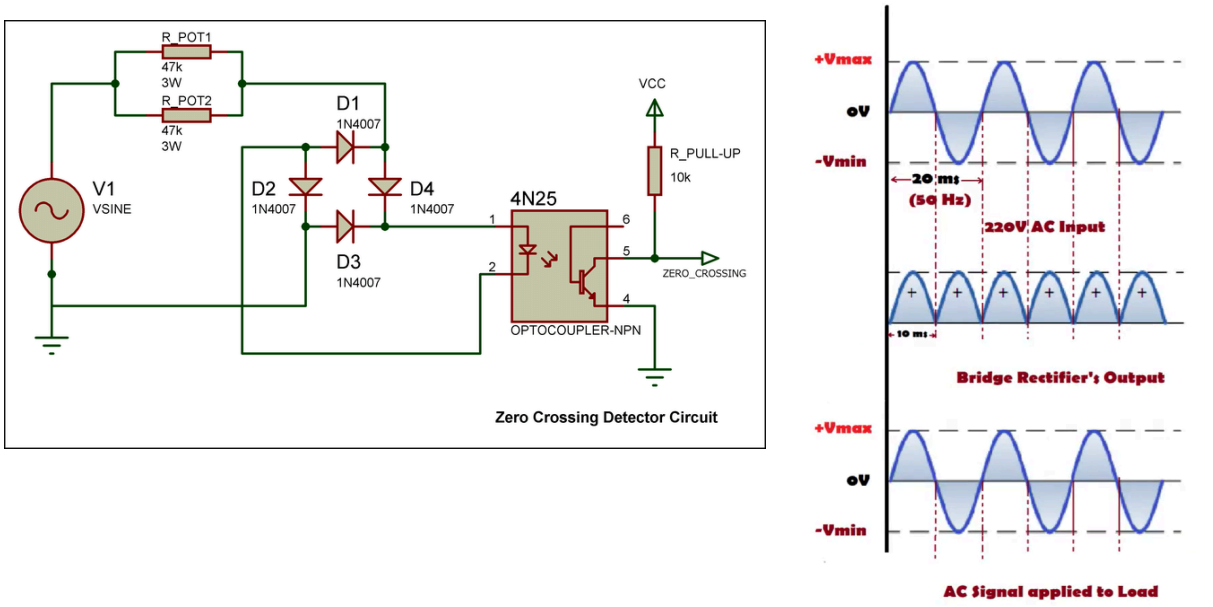


Figure 2.5: Circuit diagram of Zero Crossing Detector and its signal wave

2.3.2 Phase Angle Controlling Circuit

The key components for controlling an AC motor's speed are:

1. **Zero Crossing Detection**: This determines the point in the AC waveform where the voltage crosses the zero line, allowing synchronization of the control signal.
2. **Pulse Width Modulation (PWM)**: The PWM signal controls the duration the power is ON versus OFF, which in turn controls the voltage and speed of the AC motor.
3. **TRIAC**: A TRIAC is a three-terminal AC switch that can be triggered by a low-energy signal at its gate terminal. Unlike an SCR, a TRIAC can control power flow in both directions, making it suitable for AC applications.

The general process is:

1. Detect the zero crossing point in the AC waveform.
2. Generate a PWM signal with a specific duty cycle to control the duration of the power applied.
3. Use the PWM signal to trigger the TRIAC, which in turn controls the AC voltage and speed of the motor.

For example, if the firing angle of the TRIAC is set to 90 degrees, the power output will be halved, causing the connected AC lamp to glow at half intensity.
Additionally, an Optocoupler (or Optoisolator) is used to maintain isolation between the DC control circuit and the AC power circuit, preventing interference and ensuring safety.
In summary, zero crossing detection, PWM generation, TRIAC control, and optocoupler isolation work together to regulate the speed of an AC motor by managing the applied AC voltage.

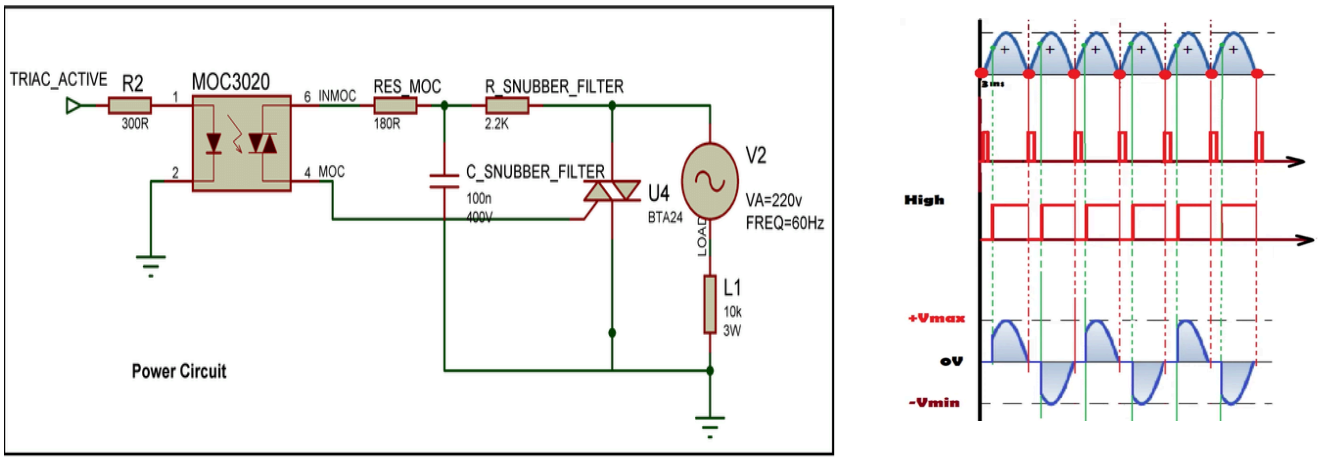
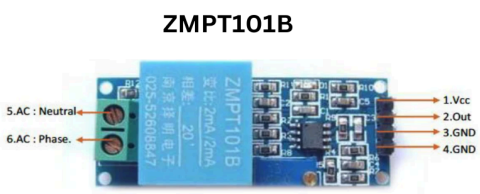


Figure 2.6: Circuit diagram of Phase Angle Controlling Circuit and its signal

2.4 Sensors

The Hall effect current sensor ACS712 is utilized to determine the output current to the motor. A linear Hall sensor circuit with a copper conduction route makes up the apparatus. The inbuilt Hall IC detects the magnetic field created by applying current passing via this copper conduction line and converts it into a proportionate voltage for the output pin, which will be connected to an Arduino analog pin for measurement.

The AC Voltage Sensor ZMPT101B is used to accurately measure AC voltage using the ZMPT101B voltage transformer. This sensor can measure up to 250 VAC and features a potentiometer for fine-tuning the analog output value. It is ideal for applications involving AC voltage measurement with an Arduino or Raspberry Pi.



Pin Name	Description
VCC	The positive terminal of the External voltage source(5~30VDC)
OUT	The Analog pin connected to Analog pin of Arduino
GND	The negative terminal of the External voltage source
Neutral	The wire through voltage has to be measured and is connected here
Phase	

(a) Voltage Sensor

Pin Name	Description
VCC	The positive terminal of the External voltage source(5VDC)
OUT	The Analog pin connected to Analog pin of Arduino
GND	The negative terminal of the External voltage source
Ip-	The wire through current has to be measured and is connected here
Ip+	

(a) Current Sensor

Figure 2.4: Circuit diagram of dimmer

2.5 Tachometer

The GM8905 tachometer is a digital handheld device used primarily for measuring the rotational speeds of various mechanical systems. It operates using an optical sensing method known as photoelectric sensing. This tachometer emits a beam of light towards a reflective surface on the rotating object. As the object spins, it reflects the light back to the tachometer's sensor. The frequency of the reflected light pulses is directly proportional to the rotational speed of the object. The GM8905 tachometer converts these pulses into electrical signals and processes them to display the rotational speed on its digital screen. It typically offers a wide measurement range and high accuracy, making it suitable for use in industrial settings, automotive diagnostics, and other applications where precise measurement of rotational speed is essential. The device is portable, easy to use, and provides real-time measurements, making it a versatile tool for maintenance, troubleshooting, and quality control purposes.

Figure 2.4: Tachometer GM8905

Chapter 3

Design and Implementation

3.1 Methodology

The experiment was conducted indoors in room 272 of the dormitory at the Vietnamese German University in Ben Cat, Binh Duong Province, Vietnam. The testing occurred on July 2, 2024, under normal conditions. Control tests for the sensors were interspersed with each experimental test to account for sensor offset errors and to ensure the temperature remained approximately consistent throughout the experiments. Initially, connect the components according to the provided circuit diagram.

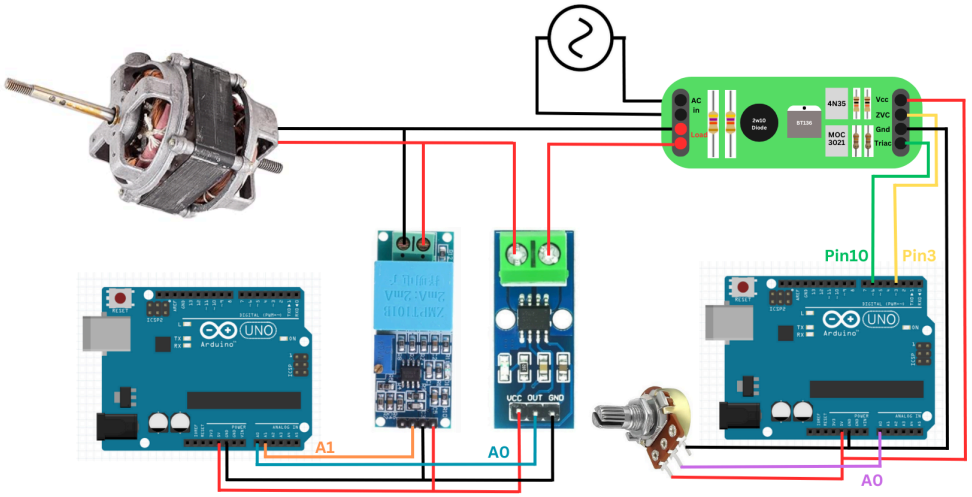


Figure 3.1: Overall configuration of the circuit

The above circuit diagram includes two Arduinos. The first Arduino controls the AC fan motor through the dimmer circuit by adjusting the speed using a potentiometer. The other Arduino is used for collecting current and voltage data, which is connected to the main circuit.

3.2 Implementation

3.2.1 Arduino Code for Motor Controller

Flowchart:

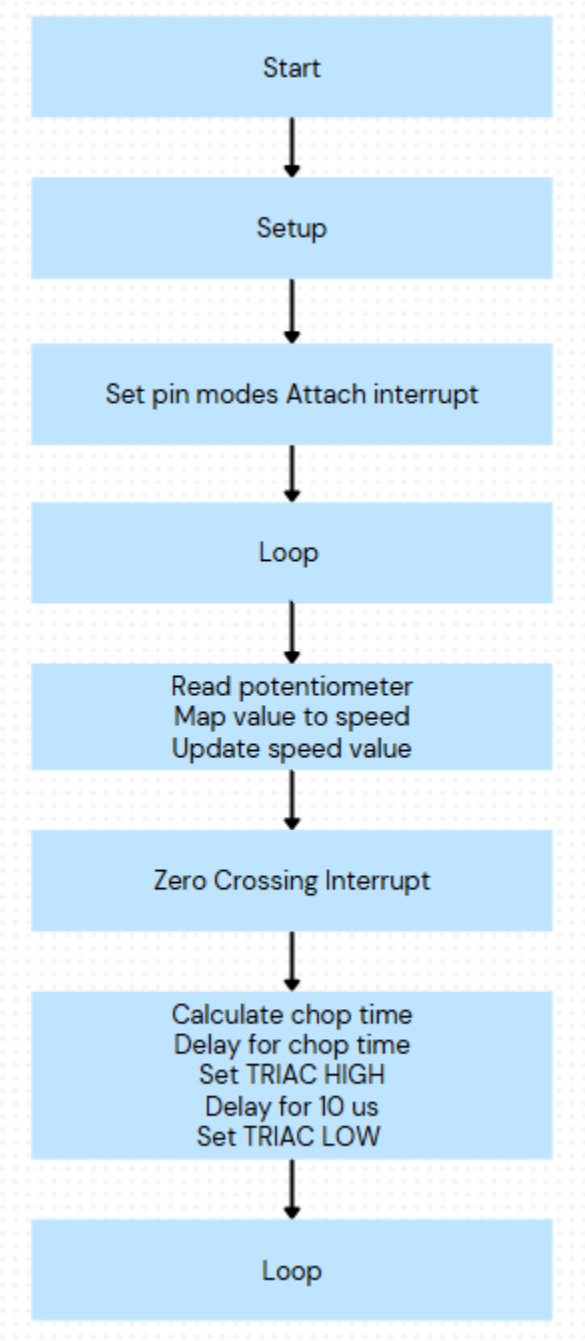


Figure 3.2: The flowchart of the motor controller code

Declaring Variables

```
#define TRIAC 6
#define ZVC 3
int speed_val = 0;
```

- **TRIAC**: Defines a constant TRIAC set to pin 6. This pin will be used to control a TRIAC, which is an electronic component used to control power.
- **ZVC**: Defines a constant ZVC set to pin 3. This pin will be used to control a ZVC, which is an electronic component used to control power.
- **speed_val**: A variable to store the speed_value, initially set to 0.

Setup Function

```
void setup()
{
   pinMode(A0, INPUT);
   pinMode(TRIAC, OUTPUT);
   pinMode(ZVC, INPUT_PULLUP);
   attachInterrupt(digitalPinToInterrupt(ZVC),zero_crossing,FALLING)
}
```

- **void setup()**: This function runs once when the Arduino starts. It sets up the initial conditions.
- **pinMode(A0, INPUT)**: Configures pin A0 (Analog) as an input pin.
- **pinMode(TRIAC, OUTPUT)**: Configures pin 6 (TRIAC) as an output pin.
- **pinMode(ZVC, INPUT_PUllUP)**: Configures pin 3 (ZVC) as an input_pullup pin.
- **attachInterrupt(digitalPinToInterrupt(ZVC), zero_crossing, FALLING)**:
    - **digitalPinToInterrupt(ZVC)**: Converts digital pin 3 to its corresponding interrupt number.
    - **zero_crossing**: The name of the function to call when the interrupt occurs.
    - **FALLING**: Specifies that the interrupt should trigger when the pin ZVC (or 3) goes from high to low.

Zero-Crossing Interrupt Service Routine (ISR)

```
void zero_crossing()
{
```

```
int chop_time = (200 * speed_val);
    delayMicroseconds(chop_time);
    digitalWrite(TRIAC, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIAC, LOW);
}
```

This function is called every time a zero-crossing event is detected on pin 3.

- **int chop_time = (200 * speed_val)**: Calculates the chop time based on the speed_val. The value 200 is a scaling factor to convert speed_val to a time delay in microseconds.
- **delayMicroseconds(chop_time)**: Delays execution for chop_time microseconds.
- **digitalWrite(TRIAC, HIGH)**: Sets the TRIAC pin (pin 6) to HIGH, turning on the TRIAC.
- **delayMicroseconds(10)**: Keeps the TRIAC on for 10 microseconds.
- **digitalWrite(TRIAC, LOW)**: Sets the TRIAC pin to LOW, turning off the TRIAC.

Main Loop

```
void loop()
{
int pot = analogRead(A0);
int data1 = map(pot, 0, 1023, 1, 45);
speed_val = data1;
}
```

- **void loop()**: This function runs repeatedly after the setup() function. It is the main body of the program.
- **int pot = analogRead(A0)**: Reads the analog value from pin A0 (connected to a potentiometer) and stores it in pot. The value ranges from 0 to 1023.
- **int data1 = map(pot, 0, 1023, 1, 45)**: Maps the value of pot from the range 0 to 1023 to a new range of 1 to 45. The map function scales the input value to the desired output range.
- **speed_val = data1**: Updates speed_val with the mapped value, which determines the delay for the TRIAC firing.

3.2.2 Arduino Code for Data Collector

Flowchart:

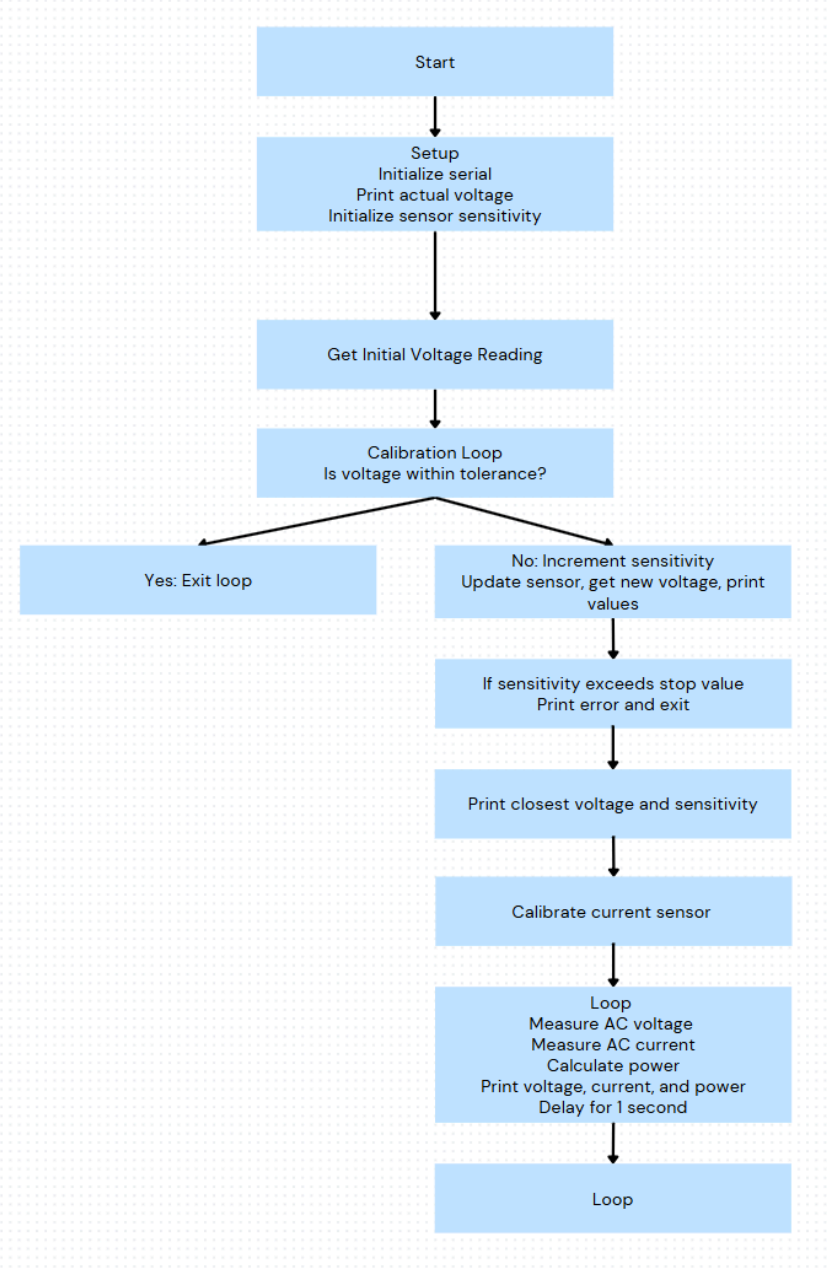


Figure 3.3: The flowchart of the data collector code

Include Libraries

```
#include <ZMPT101B.h>
#include "ACS712.h"
```

These lines include the libraries required to interface with the **ZMPT101B** voltage sensor and the **ACS712** current sensor.

Define Constants for Voltage Measurement

```
#define ACTUAL_VOLTAGE 220.0f // Change this based on actual voltage
#define START_VALUE 0.0f #define STOP_VALUE 1000.0f
#define STEP_VALUE 0.25f #define TOLLERANCE 1.0f
#define MAX_TOLLERANCE_VOLTAGE (ACTUAL_VOLTAGE + TOLLERANCE)
#define MIN_TOLLERANCE_VOLTAGE (ACTUAL_VOLTAGE - TOLLERANCE)
```

These define constants used in the voltage measurement and calibration process:

- **ACTUAL_VOLTAGE**: Expected voltage value in the system (e.g. 220V).
- **START_VALUE**: Initial sensitivity value for the voltage sensor calibration.
- **STOP_VALUE**: Maximum sensitivity value for the calibration process.
- **STEP_VALUE**: Increment step for adjusting the sensitivity during calibration.
- **TOLLERANCE**: Acceptable tolerance range for the voltage readings.
- **MAX_TOLLERANCE_VOLTAGE** and **MIN_TOLLERANCE_VOLTAGE**: Calculate the upper and lower bounds of acceptable voltage readings.

ZMPT101B Sensor Setup

```
ZMPT101B voltageSensor(A0, 50.0);
```

This initializes a ZMPT101B object with the analog pin A0 and a default sensitivity value of 50.0.

ACS712 Sensor Setup

```
ACS712 currentSensor(A1, 5.0, 1023, 100); // Adjust parameters
according to your ACS712 version
```

This initializes an ACS712 object with the following parameters:

- **A1**: Analog pin for current measurement.

- **5.0**: Reference voltage (from Microcontroller).
- **1023**: ADC resolution.
- **100**: ACS712 sensor sensitivity.

Setup Function

```
void setup() {
Serial.begin(115200);// Voltage sensor calibration
Serial.print("The Actual Voltage: ");
Serial.println(ACTUAL_VOLTAGE);
float senstivityValue = START_VALUE;
voltageSensor.setSensitivity(senstivityValue); float voltageNow
voltageSensor.getRmsVoltage();
Serial.println("Start calculate");
while (voltageNow > MAX_TOLLERANCE_VOLTAGE || voltageNow <
MIN_TOLLERANCE_VOLTAGE) {
if (senstivityValue < STOP_VALUE) {
senstivityValue += STEP_VALUE;
voltageSensor.setSensitivity(senstivityValue);
voltageNow = voltageSensor.getRmsVoltage();
Serial.print(senstivityValue);
Serial.print("=>");
Serial.println(voltageNow); }
else {
Serial.println("Unfortunately the sensitivity value cannot be
determined");
return; } }
Serial.print("Closest voltage within tolerance: ");
Serial.println(voltageNow); Serial.print("Sensitivity Value: ");
Serial.println(senstivityValue, 10);// Current sensor calibration
current
Sensor.autoMidPoint();
Serial.print("MidPoint: ");
Serial.print(currentSensor.getMidPoint());
Serial.print(". Noise mV: ");
Serial.println(currentSensor.getNoisemV()); }
```

Explanation:

1. **Initialize Serial Communication**:

```
 Serial.begin(115200);
```

2. **Print the Expected Voltage**:

```
Serial.print("The Actual Voltage: "); Serial.println(ACTUAL_VOLTAGE);
```

3. **Voltage Sensor Calibration**:

    - Start with START_VALUE for sensitivity.
    - Adjust sensitivity in steps until the measured voltage is within the tolerance range.
    - Print each sensitivity adjustment and the resulting voltage:

```
float senstivityValue = START_VALUE;
voltageSensor.setSensitivity(senstivityValue);
float voltageNow = voltageSensor.getRmsVoltage();
Serial.println("Start calculate");
while (voltageNow > MAX_TOLLERANCE_VOLTAGE || voltageNow <
MIN_TOLLERANCE_VOLTAGE) {
if (senstivityValue < STOP_VALUE) {
senstivityValue += STEP_VALUE;
voltageSensor.setSensitivity(senstivityValue);
voltageNow = voltageSensor.getRmsVoltage();
Serial.print(senstivityValue);
Serial.print(" => ");
Serial.println(voltageNow);
}
else {
Serial.println("Unfortunately the sensitivity value cannot be
determined");
return;
}
}
```

```
Serial.print("Closest voltage within tolerance: ");
Serial.println(voltageNow); Serial.print("Sensitivity Value:");
Serial.println(senstivityValue, 10);
```

4. **Current Sensor Calibration**:

    - Automatically find the midpoint and print it along with noise in mV:

```
currentSensor.autoMidPoint();
Serial.print("MidPoint: ");
Serial.print(currentSensor.getMidPoint());
Serial.print(". Noise mV: ");
Serial.println(currentSensor.getNoisemV());
```

Loop Function

```
void loop() {
// Measure AC voltage
float voltageNow = voltageSensor.getRmsVoltage();
// Measure AC current
float currentNow = currentSensor.mA_AC_sampling() / 1000.0; //Convert
from mA to A
// Calculate the power
float power = voltageNow * currentNow;
// Print voltage, current, and power to serial
Serial.print("Voltage (V): ");
Serial.print(voltageNow);
Serial.print(", Current (A): ");
Serial.print(currentNow);
Serial.print(", Power (W): ");
Serial.println(power);

delay(1000);
}
```

Explanation:

1. **Measure AC Voltage**:

```
 float voltageNow = voltageSensor.getRmsVoltage();
```

2. **Measure AC Current**:

Convert the measured current from mA to A:

```
float currentNow = currentSensor.mA_AC_sampling() / 1000.0;
```

3. **Calculate Power**:

```
float power = voltageNow * currentNow;
```

4. **Print Measurements**:

```
Serial.print("Voltage (V): ");
Serial.print(voltageNow);
Serial.print(", Current (A): ");
Serial.print(currentNow);
Serial.print(", Power (W): ");
Serial.println(power);
```

5. **Delay**:

Introduce a delay of 1 second between measurements:

```
delay(1000);
```

This loop continuously measures and prints the voltage, current, and power at 1-second intervals.

3.2.3 Python program to plot and expand data to .xlsx file

Flowchart:

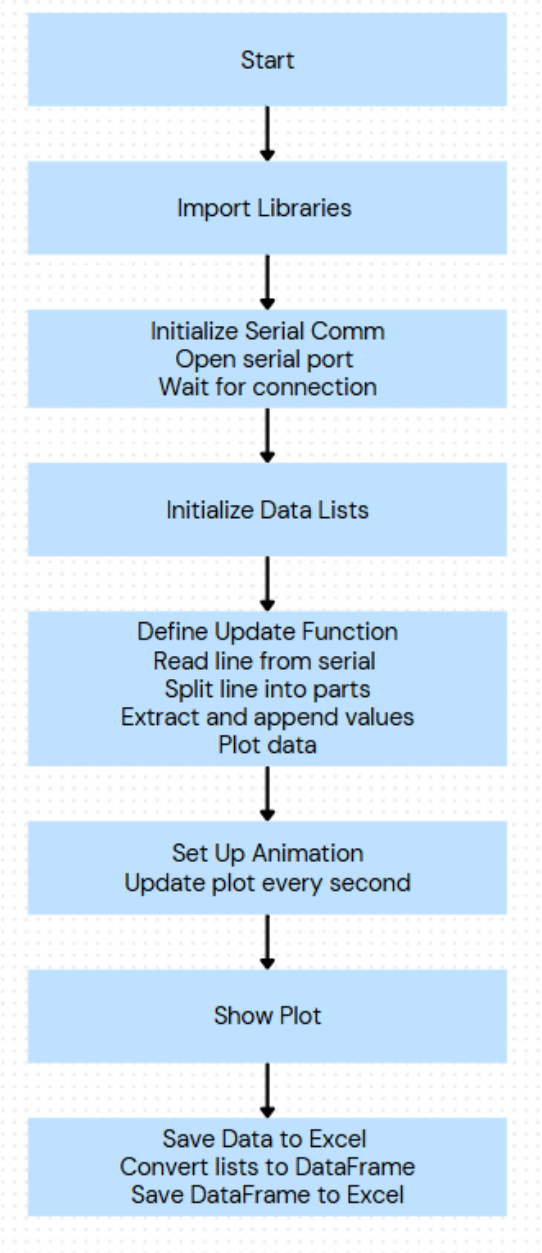


Figure 3.4: The flowchart of the python plot and expand data program

Importing Required Libraries

```
import serial
import time
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

- **serial**: Used for serial communication to read data from a COM port.
- **time**: Provides time-related functions, here used for delays.
- **pandas**: A powerful data manipulation and analysis library.
- **matplotlib.pyplot**: A plotting library for creating visualizations.
- **matplotlib.animation.FuncAnimation**: Used for creating animated plots.

Setting Up Serial Communication

```
ser = serial.Serial('COM9', 115200)
time.sleep(2)
```

- **serial.Serial('COM9', 115200)**: Opens a serial connection on COM port 9 with a baud rate of 115200.
- **time.sleep(2)**: Waits for 2 seconds to ensure the serial connection is established.

Initializing Data Lists

```
voltage_data = []
current_data = []
power_data = []
```

These lists will store the voltage, current, and power data read from the serial port.

Update Function

```
def update(frame):
line = ser.readline().decode('utf-8').strip()
if line:
try:
parts = line.split(", ")
voltage = float(parts[0].split(": ")[1])
current = float(parts[1].split(": ")[1])
```

```
power = float(parts[2].split(": ")[1])
voltage_data.append(voltage)
current_data.append(current)
power_data.append(power)

plt.cla()
plt.subplot(3, 1, 1)
plt.plot(voltage_data, label='AC Voltage (V)')
plt.xlabel('Time (s)')
plt.ylabel('AC Voltage (V)')
plt.legend(loc='upper right')
plt.subplot(3, 1, 2)
plt.plot(current_data, label='AC Current (A)')
plt.xlabel('Time (s)')
plt.ylabel('AC Current (A)')
plt.legend(loc='upper right')

plt.subplot(3, 1, 3)
plt.plot(power_data, label='Power (W)')
plt.xlabel('Time (s)')
plt.ylabel('Power (W)')
plt.legend(loc='upper right')

plt.tight_layout()
except (ValueError, IndexError):
Pass
```

Explanation:

1. **Reading Serial Data**:

```
line = ser.readline().decode('utf-8').strip()
```

   - Reads a line from the serial port, decodes it to a UTF-8 string, and strips any leading/trailing whitespace.

2. **Parsing the Data**:

```
if line: try: parts = line.split(", ")
voltage = float(parts[0].split(": ")[1])
current = float(parts[1].split(": ")[1])
power = float(parts[2].split(": ")[1])
```

- Splits the line by commas, then further splits each part to extract the numeric values for voltage, current, and power.

3. **Appending Data to Lists**:

```
voltage_data.append(voltage)
current_data.append(current)
power_data.append(power)
```

4. **Clearing the Current Axes**:

```
plt.cla()
```

5. **Plotting Voltage Data**:

```
plt.subplot(3, 1, 1)
plt.plot(voltage_data, label='AC Voltage (V)')
plt.xlabel('Time (s)')
plt.ylabel('AC Voltage (V)')
plt.legend(loc='upper right')
```

6. **Plotting Current Data**:

```
plt.subplot(3, 1, 2)
plt.plot(current_data, label='AC Current (A)')
plt.xlabel('Time (s)')
plt.ylabel('AC Current (A)')
plt.legend(loc='upper right')
```

7. **Plotting Power Data**:

```
plt.subplot(3, 1, 3)
```

```
plt.plot(power_data, label='Power (W)')
plt.xlabel('Time (s)')
plt.ylabel('Power (W)')
plt.legend(loc='upper right')
```

8. **Adjusting Layout**:

```
plt.tight_layout()
```

9. **Exception Handling**:

```
except (ValueError, IndexError): pass
```

   - Catches and ignores ValueError or IndexError exceptions that might occur if the data is not formatted as expected.

Creating the Animation:

```
ani = FuncAnimation(plt.gcf(), update, interval=1000,
cache_frame_data=False)
```

- **plt.gcf()**: Gets the current figure.
- **update**: The function to call at each frame of the animation.
- **interval=1000**: The interval between frames in milliseconds (1 second).
- **cache_frame_data=False**: Disables caching of frame data.

Displaying the Plot:

```
plt.show()
```

- Displays the plot window.

Saving Data to Excel:

```
df = pd.DataFrame({'Voltage (V)': voltage_data, 'Current (A)':
current_data, 'Power (W)': power_data})
df.to_excel('voltage_current_power_data_.xlsx', index=False)
```

- **Creating DataFrame:**

```
df = pd.DataFrame({'Voltage (V)': voltage_data, 'Current (A)':
current_data, 'Power (W)': power_data})
```

  - Creates a pandas DataFrame from the collected data.
- **Saving to Excel:**

```
df.to_excel('voltage_current_power_data_.xlsx', index=False)
```

  - Writes the DataFrame to an Excel file named voltage_current_power_data_.xlsx without including the DataFrame index.

Overall, this code reads voltage, current, and power data from a serial port, plots it in real-time, and saves the collected data to an Excel file.

3.3 Model Design

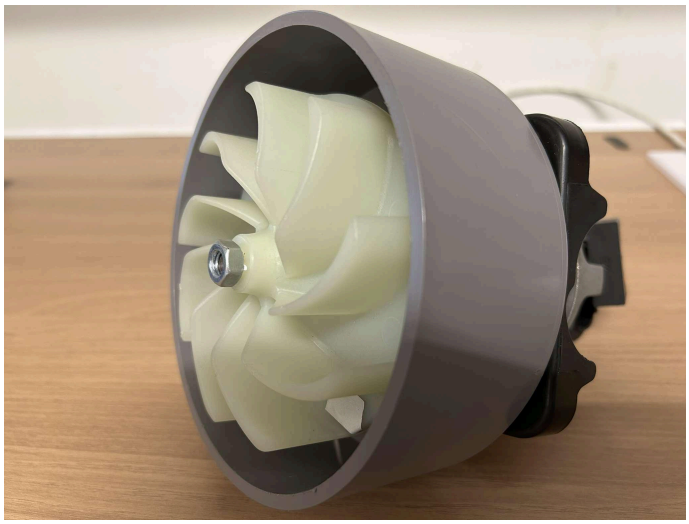


Figure 3.2: 3D-printed vacuum fan

The 3D-printed vacuum fan is printed with an MK8 1.75-0.1 3D extrusion nozzle and uses hard ABS plastic for good heat and force resistance. The fan is connected to the AC fan motor via a 140-gauge plastic seal. The seal is perforated to pass through the motor's connection shaft and direct the airflow out.

Figure 3.3: Overview of vacuum cleaners

The vacuum fan is connected to a 140/90 reduction tube, then to a 90/60 reduction tube, reducing the tube diameter. This increases air flow and vacuum pressure, allowing the vacuum cleaner to pick up heavier materials or vacuum farther. A thin mesh between the tubes prevents objects from being sucked into the vacuum impeller.

Chapter 4

Testing and conclusion

4.1 Table Value and Characteristics Curve

Test case 1: 1410 rpm - 10%

- The Actual Voltage: 220.00
- Closest voltage within tolerance: 219.38
- Sensitivity Value: 537.2500000000
- MidPoint: 510
- Noise mV: 21

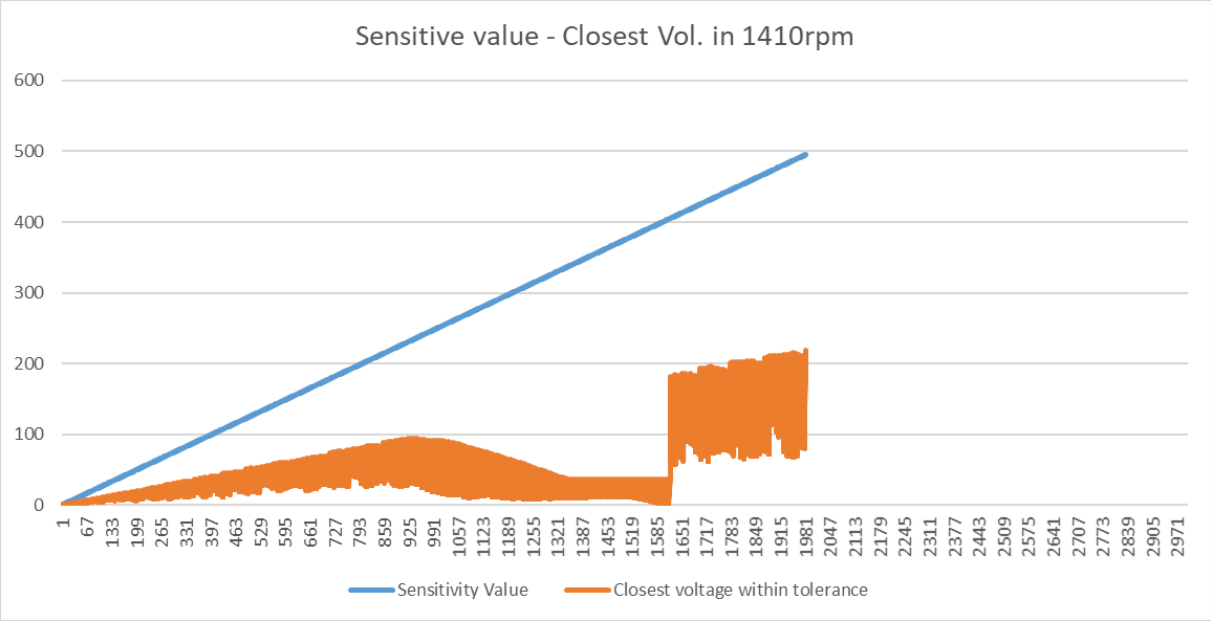


Figure 4.1: Sensitive value - Closet Vol of test case 1 - 10%

Voltage (V)	Current (A)	Power (W)	Torque (N.m)
214.86	0.14	29.51	0.1999593441
208.60	0.13	26.78	0.1814609026
214.94	0.16	35.25	0.2388535032
192.26	0.17	32.77	0.2220490581
197.27	0.17	33.52	0.2271310476
171.00	0.18	30.21	0.2047025342
162.49	0.18	29.78	0.2017888603
134.41	0.19	25.20	0.1707548448
122.60	0.18	22.34	0.1513755251
103.75	0.18	18.29	0.1239327822
159.34	0.12	19.28	0.1306410083
120.30	0.11	13.80	0.0935086055
131.03	0.08	10.65	0.0721642499
176.48	0.08	13.33	0.09032389213
220.43	0.07	16.43	0.1113294484
222.80	0.11	23.92	0.1620815829
211.48	0.14	29.79	0.2018566201
211.25	0.14	30.50	0.2066675701
201.80	0.18	35.48	0.2404119799
188.53	0.17	32.48	0.2200840222
191.29	0.18	34.59	0.2343813525
161.42	0.18	29.73	0.201450061
157.49	0.18	27.72	0.1878303293
107.88	0.16	17.22	0.1166824773
91.79	0.16	14.27	0.09669331888
122.38	0.11	13.94	0.09445724353
208.59	0.06	13.52	0.09161132945
213.12	0.12	26.48	0.1794281068
190.97	0.17	33.30	0.2256403307
162.99	0.18	28.97	0.1963003117

95.69	0.17	16.21	0.1098387315
122.74	0.12	14.77	0.1000813118
192.84	0.07	13.59	0.09208564846
221.90	0.11	24.54	0.1662826941
193.94	0.17	33.56	0.227402087
170.42	0.18	29.96	0.2030085377
122.35	0.19	22.96	0.1555766364
97.30	0.17	16.42	0.1112616886
73.80	0.14	10.36	0.07019921399
138.65	0.08	10.71	0.07257080905
219.22	0.07	14.35	0.09723539775
207.87	0.13	26.45	0.1792248272
199.37	0.18	36.05	0.2442742919
145.71	0.18	26	0.1761756336

Figure 4.2: Table value of test case 1 - 10%

Test case 2: 1425 to 1435 rpm - 50%

- The Actual Voltage: 220.00
- Closest voltage within tolerance: 219.46
- Sensitivity Value: 523.2500000000
- MidPoint: 509
- Noise mV: 21

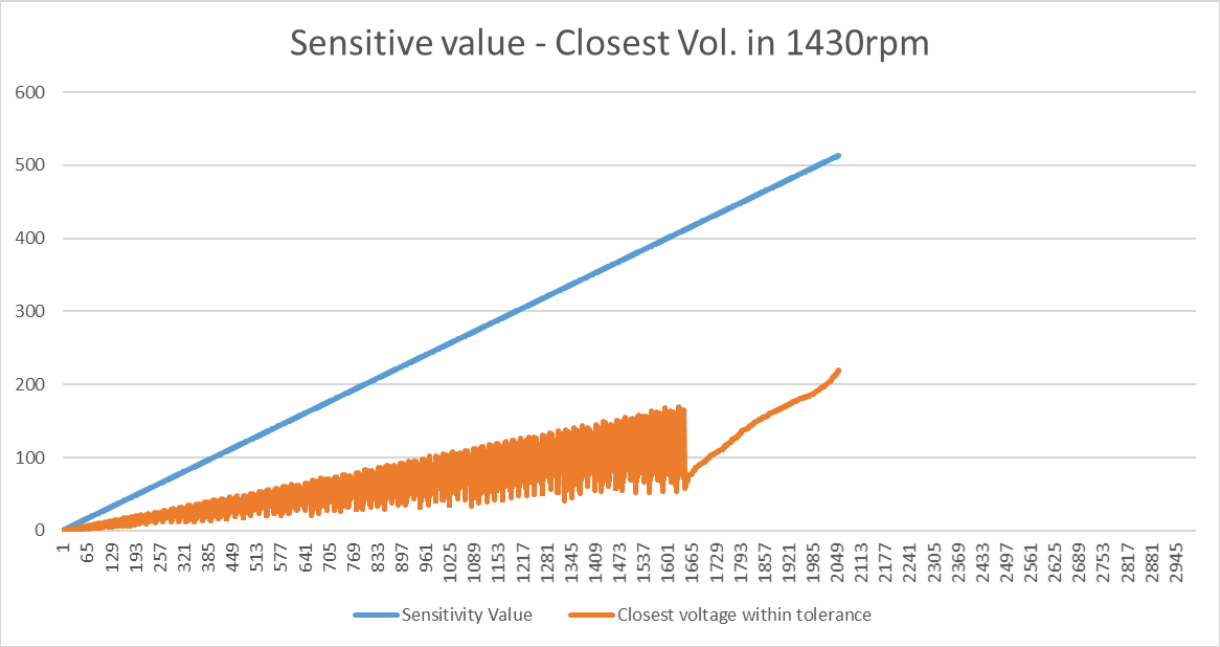


Figure 4.3: Sensitive value - Closet Vol of test case 2 - 50%

Voltage (V)	Current (A)	Power (W)	Torque (N.m)
185.96	0.07	17.76	0.1186584117
145.23	0.20	28.7	0.1917509242
181.11	0.09	16.54	0.1105073271
176.67	0.18	31.46	0.2101910828
148.20	0.11	15.64	0.1044942319
187.43	0.17	31.7	0.2117945749
89.47	0.17	15.22	0.1016881208
214.60	0.17	35.66	0.2382521937
93.20	0.17	15.82	0.1056968509
203.54	0.13	26.84	0.1793238609
84.86	0.19	16.11	0.1076344038
215.25	0.13	28.87	0.1928867311
105.72	0.17	17.96	0.119994655
204.23	0.13	27.42	0.1831989666
90.17	0.16	14.76	0.09861476104
202.88	0.13	26.22	0.1751815064

88.81	0.16	14.62	0.09767939067
199.41	0.15	29.44	0.1966950247
86.61	0.13	11.63	0.07770255222
189.91	0.17	32.38	0.2163378023
86.76	0.16	13.54	0.09046367645
205.50	0.15	31.48	0.2103247071
106.00	0.14	14.62	0.09767939067
196.24	0.12	24.36	0.162754443
94.24	0.18	16.54	0.1105073271
217.83	0.12	27.11	0.1811277894
115.65	0.18	20.39	0.136230012
212.97	0.08	16.52	0.1103737027
120.93	0.19	22.97	0.1534675516
215.28	0.07	15.32	0.1023562425
166.10	0.18	30.45	0.2034430538
151.41	0.09	13.66	0.09126542248
177.46	0.18	31.75	0.2121286357
66.69	0.17	11.61	0.07756892789
213.66	0.13	26.77	0.1788561757
118.06	0.17	20.31	0.1356955147
191.89	0.06	11.79	0.07877154692
149.21	0.18	27.23	0.1819295354
119.54	0.12	14.69	0.09814707585
206.71	0.15	31.14	0.2080530934
93.76	0.17	16.9	0.1129125651
211.45	0.08	17.25	0.115250991
131.33	0.19	24.9	0.1663623001

Figure 4.4: Table value of test case 2 - 50%

Test case 3: 1460 rpm - 100%:

- The Actual Voltage: 220.00
- Closest voltage within tolerance: 219.73
- Sensitivity Value: 527.0000000000
- MidPoint: 509
- Noise mV: 21

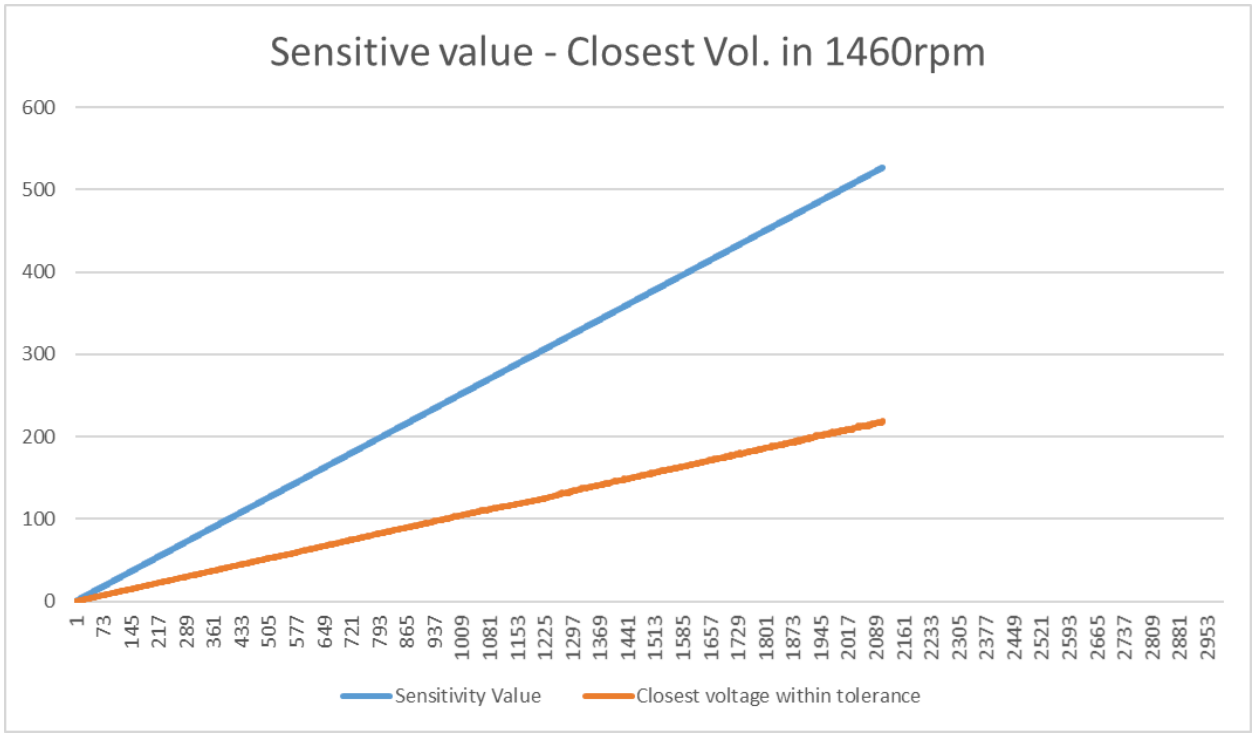


Figure 4.5: Sensitive value - Closet Vol of test case 3 - 100%

Voltage (V)	Current (A)	Power (W)	Torque (N.m)
215.62	0.17	36.42	0.2383299887
218.50	0.16	35.93	0.2351234622
217.52	0.17	36.80	0.2408166827
219.02	0.17	36.47	0.2386571852
217.13	0.18	39.55	0.2588124945
217.95	0.16	34.61	0.2264854725
218.96	0.17	36.40	0.23819911

216.90	0.18	38.83	0.2541008638
216.16	0.17	37.12	0.2429107408
217.72	0.17	36.48	0.2387226246
216.60	0.17	36.99	0.2420600297
219.81	0.17	38.19	0.2499127476
217.40	0.18	38.86	0.2542971817
217.09	0.17	36.72	0.2402931681
218.53	0.17	37.36	0.2444812844
216.39	0.17	36.85	0.2411438792
217.52	0.17	36.47	0.2386571852
218.31	0.18	39.31	0.257241951
216.15	0.17	37.23	0.2436305732
216.84	0.17	37.48	0.2452665561
217.07	0.17	36.21	0.236955763
217.15	0.18	38.18	0.2498473083
216.69	0.17	37.47	0.2452011168
217.19	0.16	35.76	0.2340109938
217.60	0.18	39.17	0.2563258005
218.42	0.17	37.66	0.2464444638
217.57	0.16	35.38	0.2315242998
218.04	0.17	36.89	0.2414056365
218.13	0.17	37.39	0.2446776023
217.09	0.17	37.24	0.2436960126
217.57	0.17	36.31	0.2376101562
217.79	0.16	35.65	0.2332911613
218.39	0.17	36.76	0.2405549254
216.28	0.17	35.72	0.2337492365
217.28	0.17	37.75	0.2470334177
217.00	0.17	36.67	0.2399659716
218.59	0.17	36.77	0.2406203647
217.30	0.17	36.92	0.2416019545
217.67	0.17	36.38	0.2380682314

217.57	0.16	35.69	0.2335529186
216.45	0.17	36.32	0.2376755955
218.10	0.18	38.50	0.2519413664
216.55	0.17	36.08	0.2361050519
217.81	0.17	36.43	0.238395428

Figure 4.6: Table value of test case 3 - 100%

4.2 Summary:

- As we can see from test scenario 1, there is some noise signal present in the figure between the sensor's sensitive value and the nearest voltage within tolerance, making it difficult to interpret. 510 is the biggest midpoint. Thus, with a value of 0.1609265853 N.m.
  -> Most unstable Voltage signal example, the average torque comes in second place; yet, it is still superior to example 2.

- In test scenario 2, there is an unstable signal from the start but a linear signal that persists from the second of 16.76 until the finish.
  -> Case with the lowest average torque value (0.1454407687 N.m.), but a more reliable voltage signal

- The third test instance displays the most steady and linear signal among the three, with the best power value at 0.2420317718 N.m.
  -> Most reliable test scenario

4.3 Interactions between component values

Scenario	Speed(rpm)	Voltage(V	Current(mA)	Power(W)	Torque(N.m.e-3)
CASE 1 - 10%	1410	205.586	154	31.566	214
		142.265	172	24.183	164
		174.208	90	15.626	106
		200.870	162	32.568	221
		128.192	158	20.576	139

		174.272	140	23.696	161
		180.368	130	23.284	158
		130.264	130	14.960	101
		184.317	163	29.500	200
CASE 2 - 50%	1430	167.434	130	22.020	147
		157.648	162	25.048	167
		163.650	144	23.046	154
		130.300	154	20.322	136
		163.962	142	22.822	152
		166.186	140	21.130	141
		145.456	148	20.820	139
		152.222	136	20.350	136
		171.390	135	21.075	141
CASE 3 - 100%	1460	217.558	170	37.034	242
		217.538	170	36.688	240
		217.886	172	37.624	246
		217.042	172	37.468	245
		217.140	172	37.358	244
		217.850	168	36.912	242
		217.462	168	36.438	238
		217.626	168	36.486	239
		217.228	173	36.833	241

Figure 4.7: Table value of 3 cases

4.3.1. Interaction between Torque and Speed

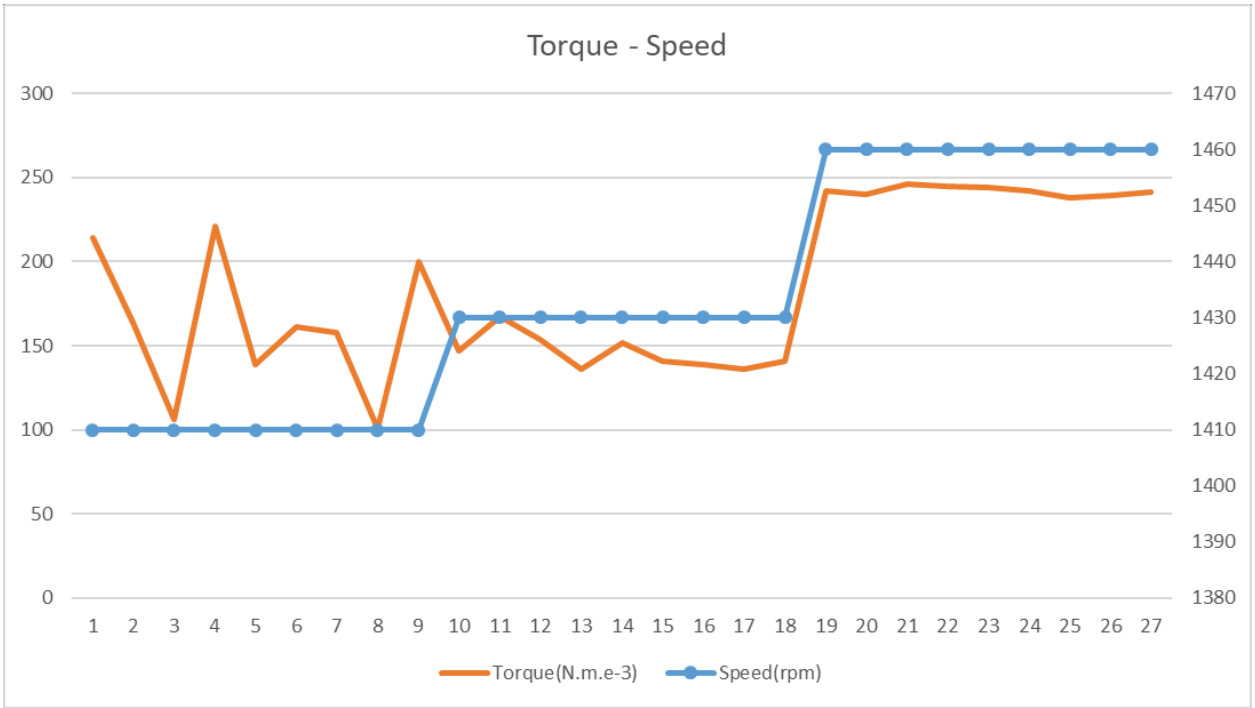


Figure 4.8: Torque - Speed Curve

Torque (N.m.e-3) vs Speed (rpm)

1. **Initial Phase (Data Points 1-10):**
    - Torque: The torque fluctuates significantly, ranging approximately between 100 and 250 N.m.e-3.
    - Speed: The speed remains relatively constant at around 100 rpm during this phase.
2. **Middle Phase (Data Points 11-17):**
    - Torque: Initially, there is a sharp decline in torque after point 10, followed by a period of lower and more stable torque.
    - Speed: The speed is steady at around 1460 rpm.
3. **Final Phase (Data Points 18-27):**
    - Torque: There is a noticeable spike in torque at point 19, which then stabilizes with minor fluctuations around 150 N.m.e-3.
    - Speed: The speed shows a significant increase to around 1460 rpm and remains stable.

Analysis:

1. **Inverse Relationship (Initial Phase):**
    - There seems to be an inverse relationship between torque and speed. As the torque fluctuates, the speed remains constant and low, indicating that variations in torque do not significantly affect the motor speed in this range.
2. **Direct Relationship (Middle Phase):**
    - During the middle phase, torque decreases and stabilizes, while speed increases and then remains constant. This suggests that once the motor reaches a certain speed, the torque requirements stabilize.
3. **Stabilization (Final Phase):**
    - In the final phase, both torque and speed stabilize. The motor reaches a higher speed and maintains a stable torque, indicating efficient operation where the motor can handle the load without significant fluctuations in speed or torque.

Conclusion:

- The graph indicates that the motor experiences different operational behaviors in different phases. Initially, there is a less defined relationship where torque fluctuates significantly with a constant low speed. As the motor reaches higher speeds, the relationship stabilizes, indicating efficient performance with stable torque and speed. This behavior is typical in motors where initial loads cause fluctuations, but as the motor reaches its optimal operational speed, the torque requirements stabilize.

4.3.2 Current vs Speed

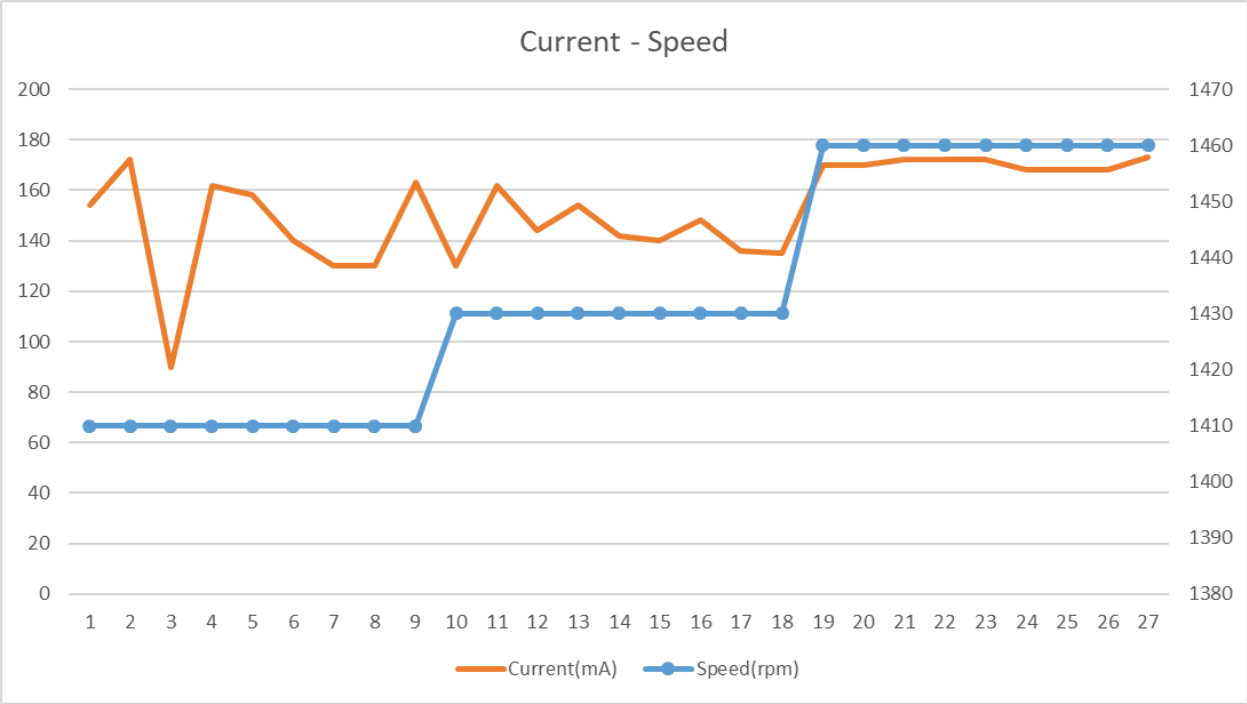


Figure 4.9: Current - Speed Curve

Current (mA) vs Speed (rpm)

1. **Initial Phase (Data Points 1-10):**
    - Current: The current fluctuates significantly, ranging approximately between 80 and 180 mA.
    - Speed: The speed remains constant at around 60 rpm during this phase.
2. **Middle Phase (Data Points 11-17):**
    - Current: Initially, there is a sharp decline in current after point 10, followed by a period of lower and more stable current.
    - Speed: The speed increases and remains steady at around 100 rpm.
3. **Final Phase (Data Points 18-27):**
    - Current: There is a noticeable spike in current at point 19, which then stabilizes with minor fluctuations around 150 mA.
    - Speed: The speed shows a significant increase to around 1460 rpm and remains stable.

Analysis:

1. **Inverse Relationship (Initial Phase):**
    - There is an inverse relationship between current and speed. As the current fluctuates significantly, the speed remains constant and low, indicating that variations in current do not significantly affect the motor speed in this range.
2. **Direct Relationship (Middle Phase):**
    - During the middle phase, current decreases and stabilizes, while speed increases and then remains constant. This suggests that once the motor reaches a certain speed, the current requirements stabilize.
3. **Stabilization (Final Phase):**
    - In the final phase, both current and speed stabilize. The motor reaches a higher speed and maintains a stable current, indicating efficient operation where the motor can handle the load without significant fluctuations in speed or current.

Conclusion:

- The graph indicates that the motor experiences different operational behaviors in different phases. Initially, there is a less defined relationship where current fluctuates significantly with a constant low speed. As the motor reaches higher speeds, the relationship stabilizes, indicating efficient performance with stable current and speed. This behavior is typical in motors where initial loads cause fluctuations, but as the motor reaches its optimal operational speed, the current requirements stabilize.

4.3.3 Current vs Torque

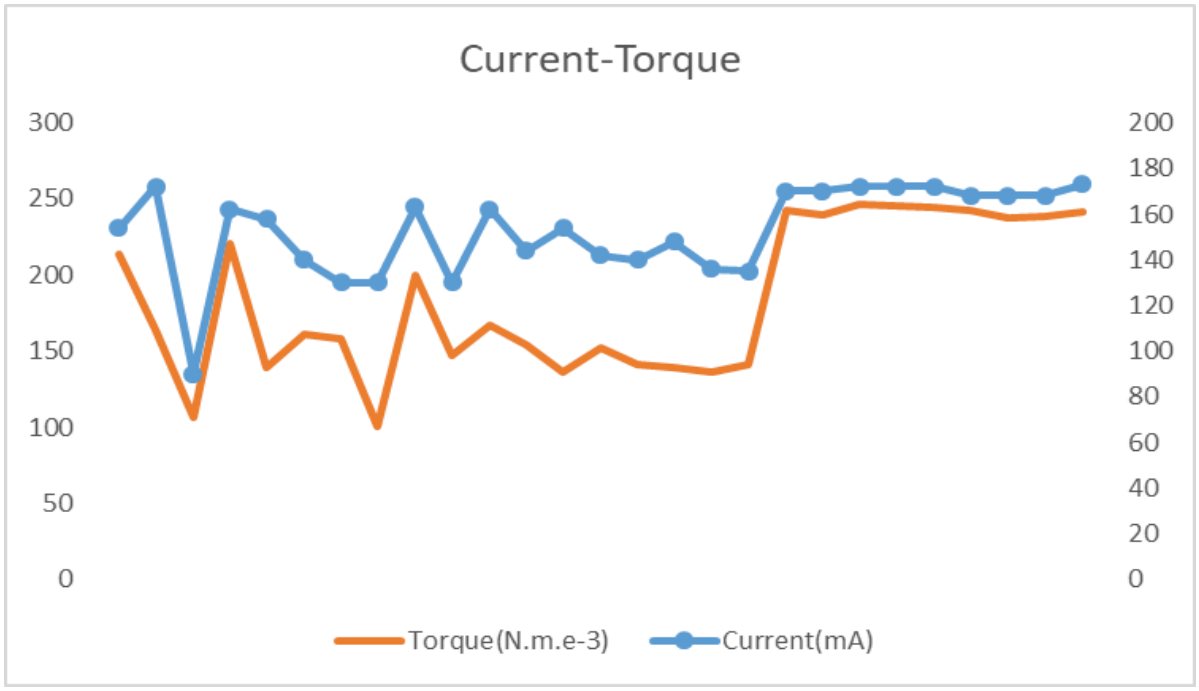


Figure 4.10: Current - Torque Curve

Current (mA) vs Torque (N.m.e-3)

1. Initial Phase (Data Points 1-10):
    - Current: The current fluctuates significantly, ranging from approximately 80 mA to 180 mA.
    - Torque: The torque also fluctuates significantly, ranging from approximately 100 N.m.e-3 to 300 N.m.e-3.
2. Middle Phase (Data Points 11-17):
    - Current: The current shows a slight decrease and then stabilizes around 150 mA.
    - Torque: The torque shows a declining trend and stabilizes around 150 N.m.e-3.
3. Final Phase (Data Points 18-27):
    - Current: There is a slight increase in current, stabilizing around 160-180 mA.

- Torque: The torque experiences a spike at point 19 and then stabilizes around 150 N.m.e-3 with minor fluctuations.

Analysis:

1. Initial Phase:
    - Both current and torque exhibit significant fluctuations, indicating a less stable relationship. High variability in torque is matched by fluctuations in current, suggesting that the motor is experiencing varying loads.
2. Middle Phase:
    - The current decreases slightly and stabilizes, while the torque also declines and stabilizes. This phase indicates a more stable relationship, with the motor reaching a consistent operational state.
3. Final Phase:
    - Both current and torque stabilize, with a slight increase in current. This phase indicates that the motor has reached a stable operational state, suggesting efficient performance.

Conclusion:

- Initial Phase: The relationship between current and torque is unstable, with both parameters showing significant fluctuations. This may indicate that the motor is handling varying loads and trying to stabilize.
- Middle Phase: As both current and torque stabilize, the motor reaches a more consistent operational state. This indicates a more direct relationship where a decrease in current corresponds with a decrease in torque.
- Final Phase: The motor operates efficiently with stable current and torque, indicating that it has reached an optimal operational state where it can handle the load with minimal fluctuations.

4.3.4 Conclusion on Effect of Changing Speed on Current and Torque

Based on the analysis of the three provided graphs, we can draw the following conclusions about how changing speed affects current and torque in a motor:

1. **Initial Phase (Low Speed - 1410rpm):**
    - Current and Torque Fluctuations: At lower speeds, both current and torque fluctuate significantly, indicating varying loads and unstable current draw.

This inconsistency suggests inefficiencies in the motor's performance at low speeds.

- Relationship: The relationship between current and torque is less predictable at low speeds due to these fluctuations.

2. **Middle Phase (Moderate Speed -1430rpm):**
    - Decreasing Current with Increasing Speed: As speed increases to a moderate level, the motor's current draw decreases and stabilizes, indicating improved efficiency at higher speeds.
    - Stabilizing Torque: Similarly, the torque also stabilizes at moderate speeds, indicating that the motor can handle the load more consistently without significant variations.
    - Relationship: There is a more direct relationship between current and torque, with both parameters stabilizing as the speed increases.
3. **Final Phase (High Speed -1460rpm):**
    - Stable Current and Torque: At higher speeds, both current and torque stabilize with minimal fluctuations, indicating efficient motor operation with consistent load handling.
    - Efficient Operation: The stable current and torque at high speeds indicate that the motor reaches its optimal operational state, where it can maintain high speed with minimal energy loss and consistent torque.
    - Relationship: A stable and efficient relationship between current and torque is observed, suggesting that the motor operates best at higher speeds.

4.3.5 Conclusion on Theory Torque-Speed Curve

Speed(rpm)	Theory Torque (N.m.e-3)	Real Torque(N.m.e-3)
100	4299.363057	2068.666667
200	2149.681529	1068.888889
400	1074.840764	549.8888889
600	716.5605096	387.6666667
800	537.4203822	250.3333333
1000	429.9363057	204.8888889
1200	358.2802548	190.7777778
1400	307.0973612	183.6666667

1410	304.9193658	162.6666667
1430	300.6547593	145.8888889
1460	294.4769217	241.8888889

Figure 4.11: Data values of theory torque

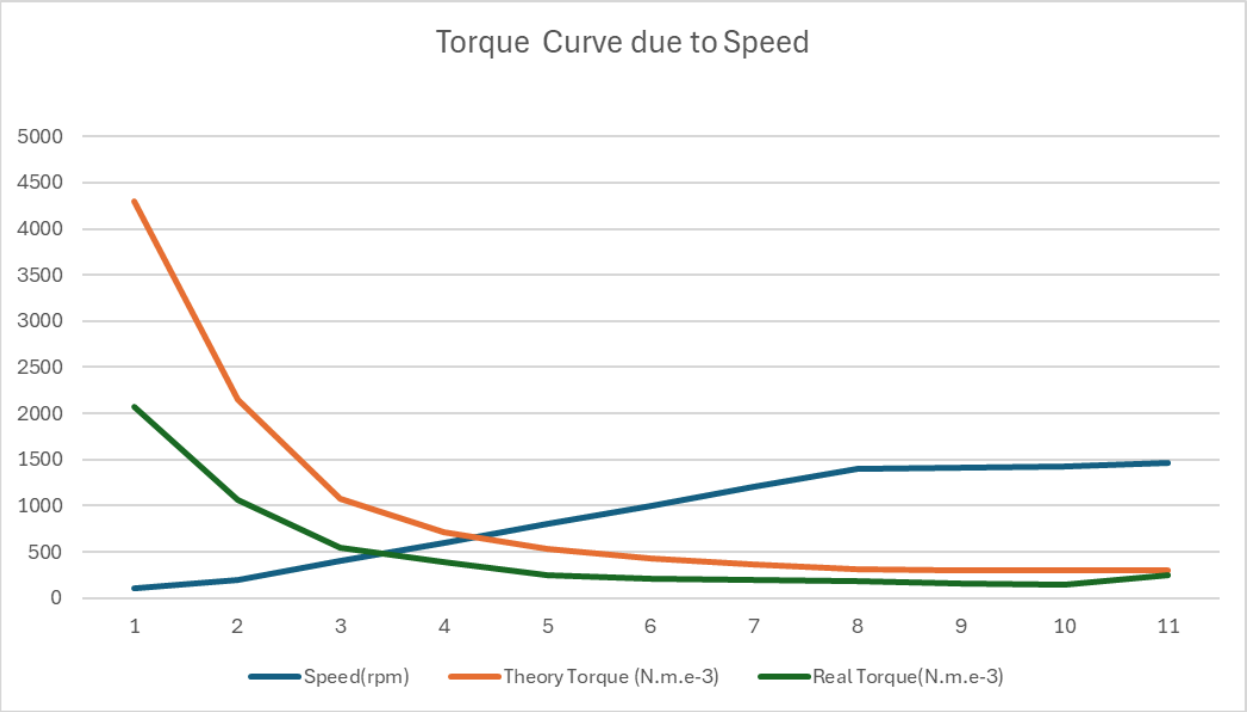


Figure 4.12: Torque-Speed Curve Theory vs Relatime measurements

Data evaluation :

- In general, the theory and real torques are inversely proportional to the speeds (rpm). Statistically, the increase in speeds by time results in the decrease in torques of both the theory and the real once by time.
- The theory torques are considered to be nearly double in value compared to the real torque. However, at the maximum speed of 1460rpm, it is witnessed a nearly similarity in value of torque of both the theory and the real once.

4.4 Overall Conclusion:

- Effect of Speed on Current:
    - At lower speeds, the current fluctuates significantly due to varying loads.
    - As speed increases, the current stabilizes and decreases, indicating improved efficiency.
    - At high speeds, the current remains stable, reflecting optimal motor performance.
- Effect of Speed on Torque:
    - At lower speeds, torque fluctuates, indicating inconsistent load handling.
    - As speed increases, torque stabilizes, suggesting more consistent load handling.
    - At high speeds, torque remains stable, reflecting efficient load handling.

Chapter 5

Experiences and Problems During The Project

1. Hardware & Electronics

- Ensuring proper connections is crucial for the success of your project. Make sure to double-check all hardware and electronics connections.
- Refer to the manufacturer's documentation for the correct wiring diagrams and pinouts.
- Use a multimeter to verify continuity and measure voltages at various points in the circuit.
- Implement basic techniques, such as isolating components, testing them individually, and routing wires neatly.
- Consider enlisting the help of an experienced electronics engineer or technician if you're facing persistent issues.

2. Coding & Data collection

- In my opinion, AC is going to be the hardest part of Motors that we can code for control speed. There are not a lot of libraries available that can support us.
- In terms of Python code, we are already testing all DC and AC measuring sensors. In the DC experiment, we can easily collect data and graph, and in

the AC experiment, it’s worse. We tried to fix the Py environment and PlatformIO, but that’s not working. So we decided to collect raw analog signals from 2 AC sensors and filter them by hand.

3. Problems during project

- Misunderstanding and using the wrong codes involved bad behaviors for the AC motor.
- Measuring wrong input value for Voltage and Current sensor made microcontrollers inside are burned out and broken.
- Over-using or manipulating too much 5V-DC supplied by the Arduino for various devices made the micro-chip of the Arduino burned out and broken.
- Lacking experience in making the Zero-crossing dimmer circuit on Zero-PCB (or prototype board) led to not working circuit.

4. Advantages and Disadvantages

Advantages:

- Reliable torque-speed curves represent characteristics of single-phase induction motor.
- Handheld and modifiable vacuum cleaner.
- Simple circuit and low-cost components.

Disadvantages:

- Not vacuum cleaner exclusive motor so it doesn't have a wide range of speed.
- Couldn't remove any garbage heavier than a piece of thin paper due to low maximum speed.
- Couldn't start from lower rpm because a weak torque couldn't run the motor.
- Low maximum torque and small range in speed control (1400rpm - 1460rpm) could lead to difficulty in analyzing the result data.