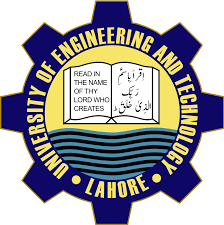
**MCT-338 Embedded System II**



**Project Report:**

**Validation and Verification Of BLDC Motor’s RPM**

**Submitted By:**

**Burhan Riaz 2022\_MC\_13**

**Rohma Masood 2022\_MC\_31**

**Ahsan Abdullah 2022\_MC\_61**

**Submitted To:**

**Sir Shujaat Ali**

**Department of Mechatronics and Control Engineering**

**University of Engineering and Technology Lahore**

**Table of Contents**

* **Methods to control BLDC motor’s RPM**
* **Reasons to choose pwm method**
* **Introduction to project**
* **Objectives of Project**
* **Description of Components**
  + BLDC
  + LCD
  + Current Sensor
  + Hall Effect Sensor
  + 12V Adapter
  + ESC
  + Fuse
  + Potentiometer
  + Tiva
* **System Workflow**
* **Pinout Configuration of TIVA**
* **Circuit Diagram**
* **Proteus Schematic Diagram**
* **PCB layout**
* **Graphs**
* **Project Pictures**
* **Dynamic Motor Dashboard Using Serial Communication**
* Purpose
* Code
* Code explanation
* Workflow Execution
* Flowchart
* **Bill of Materials**
* **Challenges Faced**
* **Conclusion**

* **Methods to control bldc motor’s rpm:**

Controlling the RPM of a BLDC motor is essential in many applications, and it can be achieved through different methods, each with its own advantages and limitations.

1. **Using PWM (Pulse Width Modulation):**  
   PWM is a method where the motor speed is controlled by varying the duty cycle of a periodic digital signal. By adjusting the "on" time of the signal, the voltage and current delivered to the motor are effectively regulated, controlling its RPM. This method is efficient, precise, and widely used due to its ability to provide smooth and stable speed control.
2. **Using Interrupts:**  
   This approach uses hardware or software interrupts to respond to events like Hall effect sensor signals, enabling real-time speed adjustments. It is well-suited for closed-loop control systems but can be complex to implement, particularly when handling high-speed or multi-tasking scenarios.
3. **Using Timer Delay:**  
   Timer delay involves controlling the timing of signals sent to the motor phases. This method is relatively simple but less efficient and precise than PWM. It may struggle to maintain consistent performance under varying load or speed requirements.

* **Reasons to choose pwm method:**

PWM is the preferred method for controlling BLDC motor RPM because it offers unmatched energy efficiency, precision, and adaptability compared to interrupts and timer delay. Here are some reasons explaining why the PWM method is advantageous:

* **Energy Efficiency:**

PWM minimizes power loss by switching the motor on and off rather than dissipating excess power as heat, making it more efficient than timer delay or continuous control.

* **Smooth Speed Control:**

The fine granularity of PWM duty cycle adjustments allows for smoother and more precise speed control compared to the discrete nature of interrupts or timer delays.

* **Scalability:**

PWM is highly adaptable to a wide range of motors and applications without significant hardware or software modifications.

* **Hardware Support:**

Most microcontrollers offer dedicated PWM modules, simplifying implementation and reducing computational overhead compared to interrupt-driven systems.

* **Noise Reduction:**

PWM produces less electrical noise compared to timer delays and ensures a stable motor operation, crucial for sensitive applications.

* **INTRODUCTION to project:**

The aim of this project is to devise an embedded system that facilitates real-time supervision and control of a BLDC motor, ensuring its operational efficacy and safety. To drive the motor and regulate its current flow, an ESC is employed, while a fuse is incorporated to safeguard the system against voltage spikes. A BLDC rotor equipped with a magnet is used in conjunction with a Hall effect sensor for precise speed measurement. Additionally, a current sensor monitors the current drawn by the motor in real time, ensuring the motor operates within its safe range. The system uses Pulse Width Modulation (PWM) for precise control of the BLDC motor's RPM, allowing efficient and smooth adjustments to the motor speed by varying the duty cycle of the applied signal. This approach minimizes energy losses while enhancing the motor's performance and responsiveness.

The real-time RPM and current values are displayed on an LCD, enabling the operator to monitor the system's performance. Two potentiometers are integrated: one for RPM control, utilizing PWM for fine-tuning motor speed, and the other for adjusting LCD brightness, enhancing user customization. This design ensures the motor's operational efficiency, protection against electrical faults, and comfortable user control, making it ideal for applications requiring accurate and reliable motor performance with instantaneous responses to various operational demands.

* **Objectives of project:**
* To devise a BLDC motor which can work efficiently within a real-time monitoring and control system which also enhances safety.
* To control the current of the motor through an electric speed controller (ESC) and manage the voltage levels of the system using a fuse.
* To detect the speed of the motor using hall effect sensor and to assess the consumption of current through the use of a current sensor.
* The goal is to ensure the RPM and the current intake are visible digitally at an LCD level in order to provide the user with instantaneous feedback concerning the performance.
* To achieve simple switchable control through the use of potentiometers for the motor speed and the motor’s LCD backlight adjustment.
* **Components Used:**

1. **BLDC (Brushless DC motor) :**

* A BLDC motor is a type of electric motor that operates without brushes, offering higher efficiency and reliability.
* It uses electronic controllers to drive the motor, eliminating mechanical wear and reducing maintenance.
* BLDC motors are known for their smooth operation, high torque-to-weight ratio, and precise speed control.
* Commonly used in embedded systems for applications like robotics, automotive, and cooling systems.
* Their compact size, durability, and energy efficiency make them ideal for modern embedded designs.
* **Specifications:**
* **Operating Voltage**: 6V to 48V.
* **Rated Power**: 250W to 1000W.
* **Speed (RPM)**: 1000 to 20,000 RPM.
* **Current Rating**: 5A to 50A.
* **Torque:** 0.1Nm to 5Nm

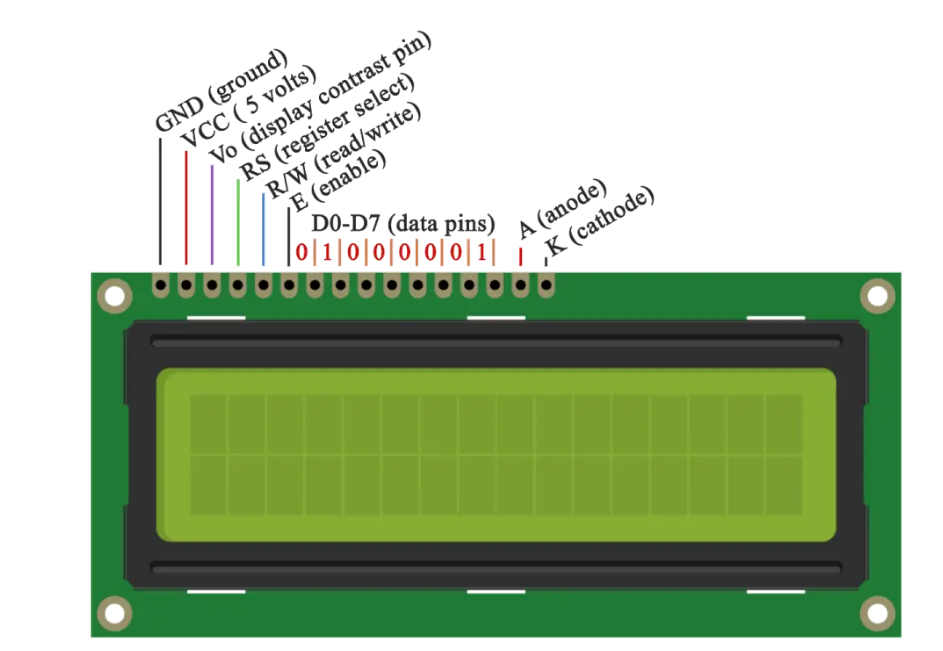


**Figure 1**

**[[1]](#footnote-1)BLDC**

1. **LCD (Liquid Crystal Display) :**

* An LCD is a flat-panel display technology that uses liquid crystals to produce images or text.
* It is widely used in embedded systems for user interfaces, offering clear and energy-efficient visual outputs.
* LCDs come in various types, such as character, graphic, and touchscreen displays, depending on the application.
* Known for low power consumption, compact size, and excellent readability under proper lighting.
* Commonly applied in devices like control panels, digital meters and gadgets.
* **Specifications:**
* **Operating Voltage:** 4.7V to 5.3V
* **Display Type:** 16 characters x 2 line
* **Character Size:** 5x8 dot matrix
* **Dimensions:** 80mm x 36mm x 12mm

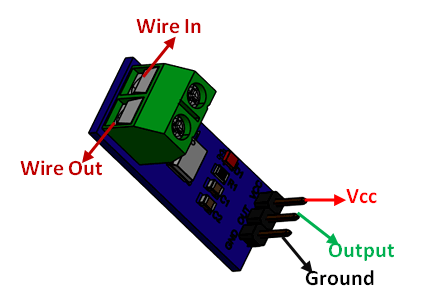


**Figure 2**

**LCD [[2]](#footnote-2)**

1. **Current Sensor:**

* A current sensor measures the flow of electrical current in a circuit and provides accurate, real-time data.
* It ensures safe and efficient operation of embedded systems by monitoring and controlling current levels.
* Operates using various principles like the Hall effect, shunt resistors, or fluxgate technology.
* Common applications include power management, motor control, and energy monitoring systems.
* Essential for enhancing system performance, safety, and energy efficiency.
* **Specifications:**
* **Operating Voltage:** 4.5V to 5.5V
* **Current Sensing Range:** ±5A, ±20A, or ±30A (depending on model variant)
* **Output Voltage:** 0V to 5V (proportional to current)
* **Accuracy:** ±1.5% of full-scale reading
* **Response Time:** 5µs
* **Operating Temperature Range:** -40°C to 85°C

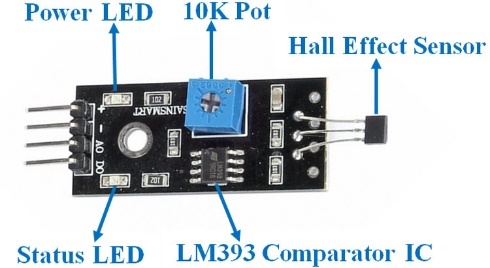


**Figure 3**

**Current Sensor[[3]](#footnote-3)**

1. **Hall Effect Sensor module:**

* A Hall Effect sensor module detects the presence of a magnetic field and converts it into a corresponding electrical signal.
* It operates based on the Hall Effect principle, where a voltage is generated perpendicular to the current flow in a conductor under a magnetic field.
* Widely used for measuring parameters like speed, position, and current in embedded systems and motor control applications.
* The module typically includes a Hall sensor, signal conditioning circuit, and output interface for analog or digital signals.
* It is reliable, contactless, and capable of operating in harsh environments, making it ideal for precision sensing tasks.
* **Specifications:**
* **Operating Voltage:** 4.5V to 24V
* **Output Current:** 20mA (max)
* **Response Time:** <5µs
* **Magnetic Sensitivity:** 20mT to 90mT (typical detection range)
* **Operating Temperature:** -40°C to 125°C
* **Frequency Range:** Up to 100kHz



**Figure 4**

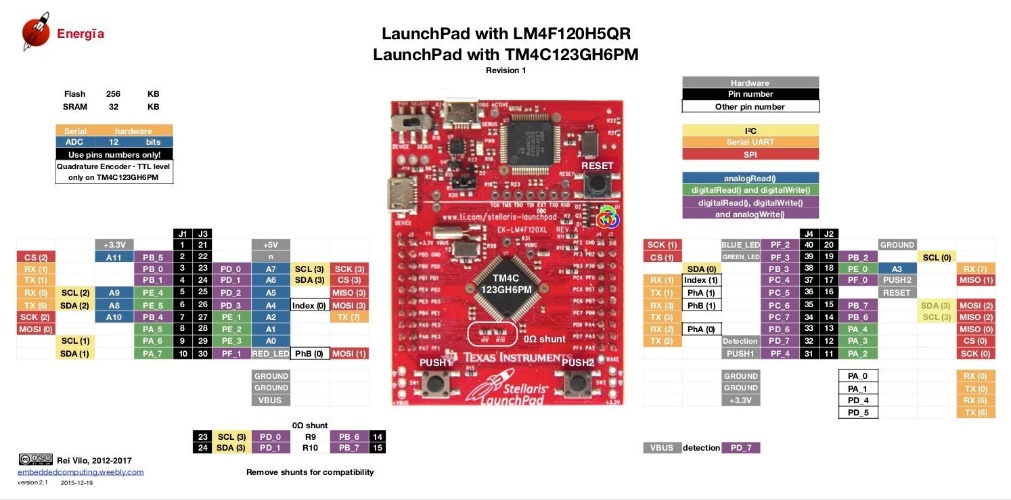
**Hall Effect Sensor[[4]](#footnote-4)**

1. **TIVA (TM4C123GH6PM):-**

The TM4C123GH6PM microcontroller is part of the Tiva™ C Series line from Texas Instruments, designed specifically for robust 32-bit ARM Cortex-M4F performance. This microcontroller is widely used in embedded systems for applications ranging from automotive and industrial to consumer electronics due to its powerful core, versatile peripherals, and compatibility with real-time operating systems.

* **Specifications:**

|  |  |
| --- | --- |
| GPIO pins | 43 |
| Frequency | 80MHz |
| Memory | * 256 KB of flash memory * 32 KB of RAM |
| Timers | * Six 32-bit timers (up to twelve 16-bit) * Two wide 32-bit timers (up to four 16-bit) * 12 PWM outputs * Two watchdog timers |
| ADCs | Two 12-bit ADCs |



**Figure 5**

**TM4C123GH6PM [[5]](#footnote-5)**

1. **12V Adapter:**

* A 12V adapter is a power supply unit that converts AC (alternating current) from a wall outlet into a stable 12V DC (direct current) output.
* It provides the necessary voltage for powering various embedded systems and electronic devices.
* Widely used in applications requiring a 12V power source, such as sensors, microcontrollers, and motors.
* Known for its compact size, reliability, and ability to handle different power requirements.
* Essential for ensuring consistent and safe operation of embedded systems in various projects.
* **Specifications:**
* **Operating Voltage:** 100V to 240V AC (input)
* **Output Voltage:** 12V DC
* **Rated Power:** 10W to 120W (depending on model)
* **Current Rating:** 0.5A to 10A

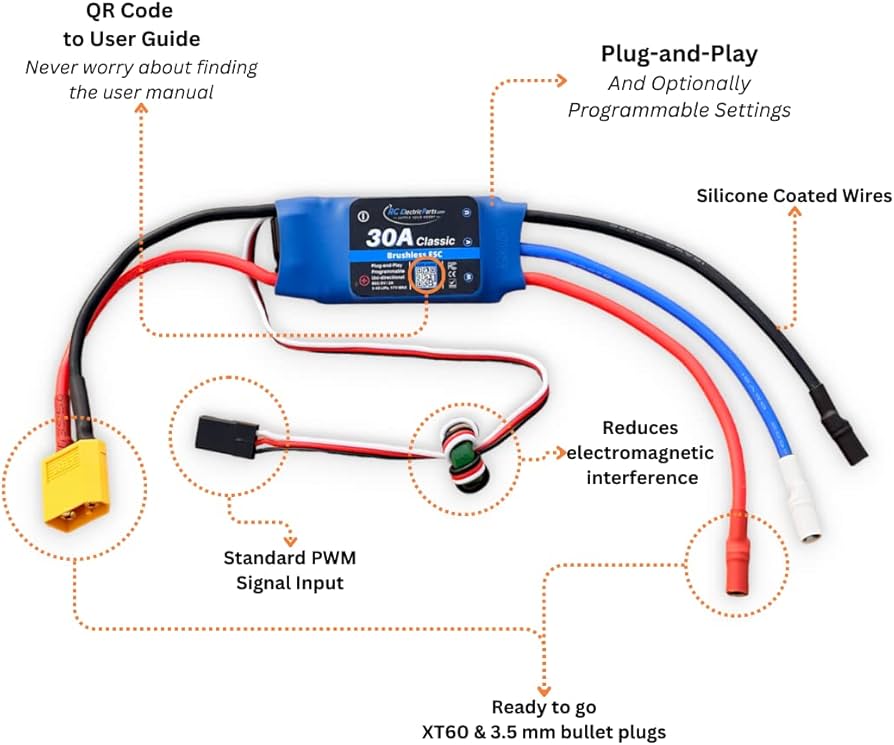


**Figure 6**

**12V Adapter[[6]](#footnote-6)**

1. **ESC (Electronic Speed Control):**

* An ESC is an electronic device used to control and regulate the speed, direction, and torque of an electric motor.
* Provides Pulse Width Modulation (PWM) signals to adjust the motor speed by varying power delivery.
* Commonly used with Brushless DC (BLDC) motors and brushed motors in drones, electric vehicles, and robotics.
* Often controlled by a microcontroller or embedded system to enable precise motor speed regulation and feedback mechanisms.
* Includes protections like overcurrent, overtemperature, and low-voltage cutoffs to prevent damage to the motor and battery.
* **Specifications:**
* **Operating Voltage:** 6V to 48V
* **Rated Power:** 250W to 1000W
* **Current Rating:** 10A to 50A (continuous), up to 70A (peak)
* **Supported Motor Types:** Brushless DC motors (sensorless and sensored)
* **PWM Frequency:** 8kHz to 32kHz
* **Input Signal:** 1ms to 2ms PWM (standard RC signal)
* **BEC Output Voltage:** 5V or 6V (depending on model, for powering peripherals)
* **Efficiency:** >90%
* **Weight:** 20g to 100g (varies with model)



**Figure 7**

**ESC[[7]](#footnote-7)**

1. **Fuse:**

* A fuse is a safety device designed to protect electrical circuits by breaking the connection in the event of excessive current flow.
* Operates by melting a thin wire or element when the current exceeds a predefined threshold, interrupting the circuit to prevent damage.
* Protects components like motors, sensors, and controllers from damage caused by short circuits, overcurrent, or voltage spikes.
* Made from metal alloys with low melting points, housed in glass, ceramic, or plastic casings.
* Specified based on the maximum current the circuit can safely handle; must match the circuit's requirements.
* Ensures the integrity of the embedded system by preventing cascading failures due to electrical faults.

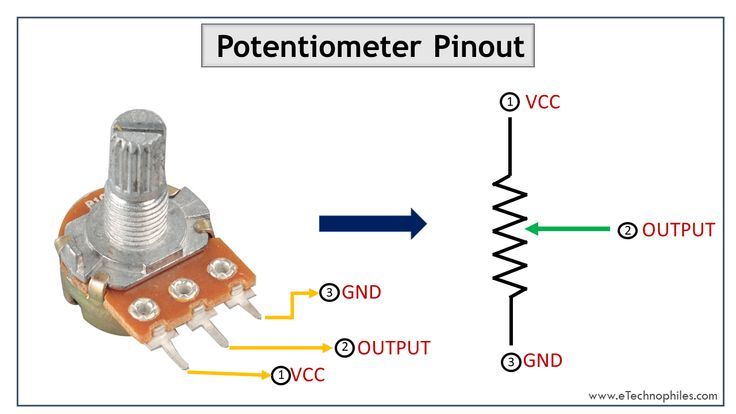


**Figure 8**

**Fuse[[8]](#footnote-8)**

1. **Potentiometer:**

* A potentiometer is a three-terminal variable resistor used to control voltage or current in a circuit.
* Adjusts resistance by sliding a wiper over a resistive track, dividing the input voltage into a variable output voltage.
* Provides an analog voltage signal for microcontrollers to read via ADC (Analog-to-Digital Converters).
* Input device for adjusting parameters (e.g., volume, brightness, or speed).
* **Specifications:**
* **Operating Voltage:** 1V to 25V
* **Rated Power:** 0.25W to 1W
* **Resistance Rating:** 1kΩ to 10kΩ
* **Adjustment Range:** 0Ω to 10kΩ
* **Operating Temperature:** -20°C to 70°C
* **Tolerance:** ±20%
* **Maximum Voltage:** 50V
* **Type:** Rotary
* **Mounting Type:** Through-hole or surface mount

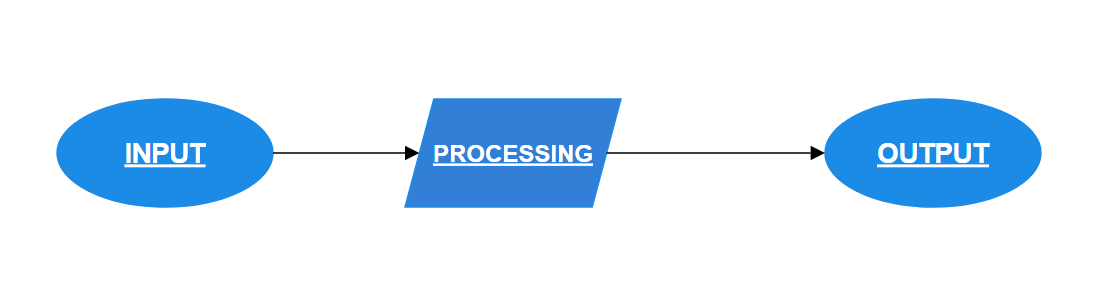


**Figure 9**

**Potentiometer[[9]](#footnote-9)**

* **System workflow:**

This design ensures seamless interaction between hardware components and provides accurate real-time monitoring of the system.

* **Input:**
* Potentiometer adjusts PWM duty cycle to control the RPM of the BLDC motor.
* Current sensor provides real-time current data to monitor the motor's current consumption.
* Hall effect sensor detects magnets on the BLDC rotor to measure RPM.
* **Processing:**
* Timer3B counts edges of the signal from the Hall effect sensor to calculate RPM.
* ADC (Analog-to-Digital Converter) converts the analog signals from the current sensor and potentiometer into digital values for processing.
* Wide Timer3A measures timing intervals to calculate motor performance and other parameters.
* Microcontroller (Tiva C) processes the input signals and adjusts motor control accordingly.
* **Output:**
* LCD displays real-time RPM and current values for user feedback.
* UART (Universal Asynchronous Receiver-Transmitter) sends RPM and current data to a computer or other monitoring system for further analysis and tracking.

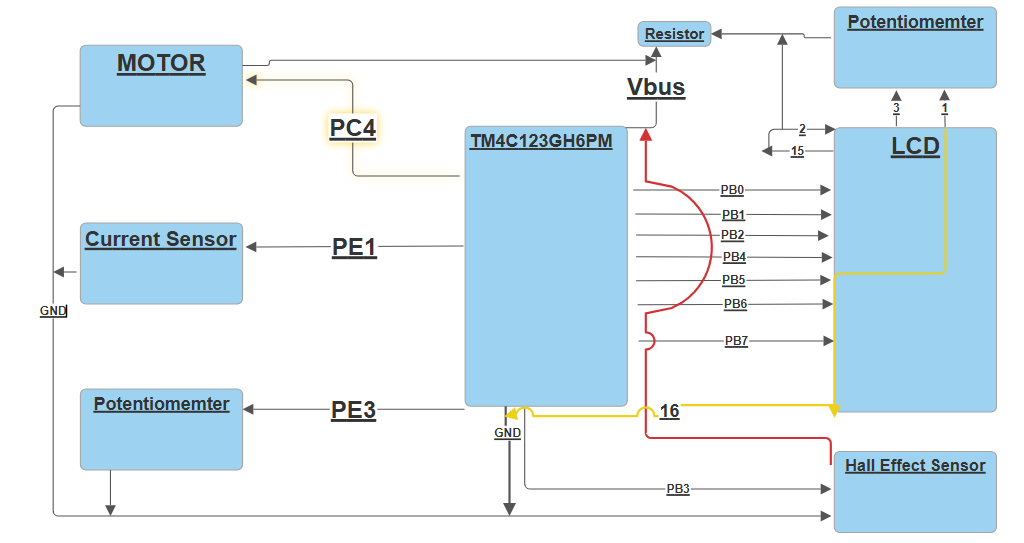
**Figure 10**

**Workflow Diagram**

* **Pinout of Components for Connection:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **TIVA Pin** | **Component** | **Component Pin** |
| **1** | V Bus | Positive of all components | - |
| **2** | GND | negative of all components | - |
| **3** | PB0 | LCD | Rs |
| **4** | PB1 | LCD | R/w |
| **5** | PB2 | LCD | Enable |
| **6** | PB4 | LCD | D4 |
| **7** | PB5 | LCD | D5 |
| **8** | PB6 | LCD | D6 |
| **9** | PB7 | LCD | D7 |
| **10** | PC4 | BLDC | Control pin |
| **11** | PE1 | Current Sensor | Control pin |
| **12** | PE3 | Potentiometer | Control pin |
| **13** | GND | LCD | Vo(potentiometer) |

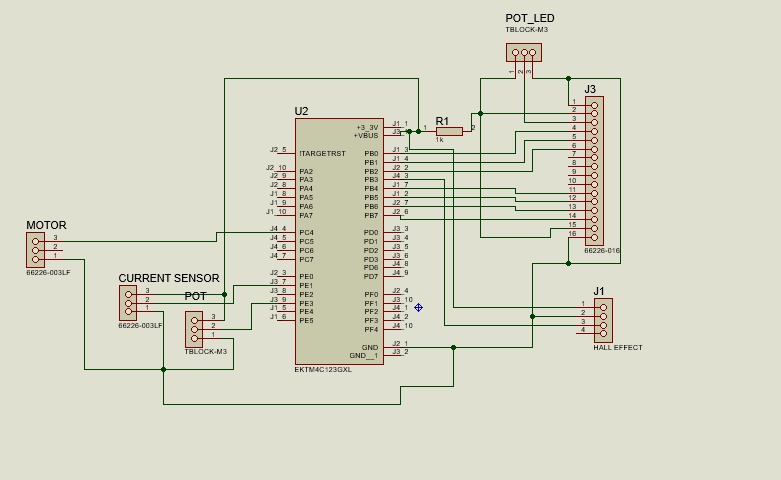
* **Circuit Diagram:**

****

**Figure 11**

**Circuit Diagram**

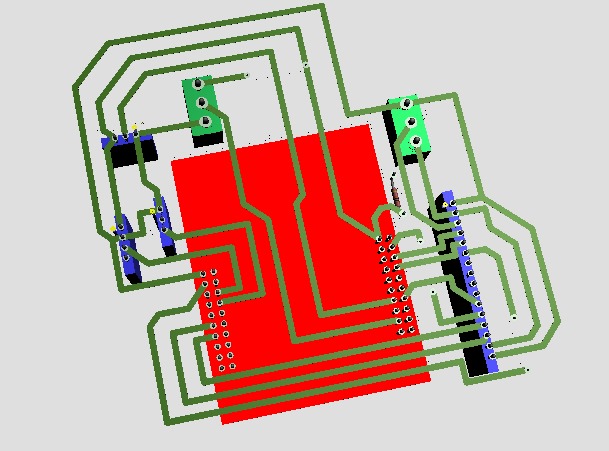
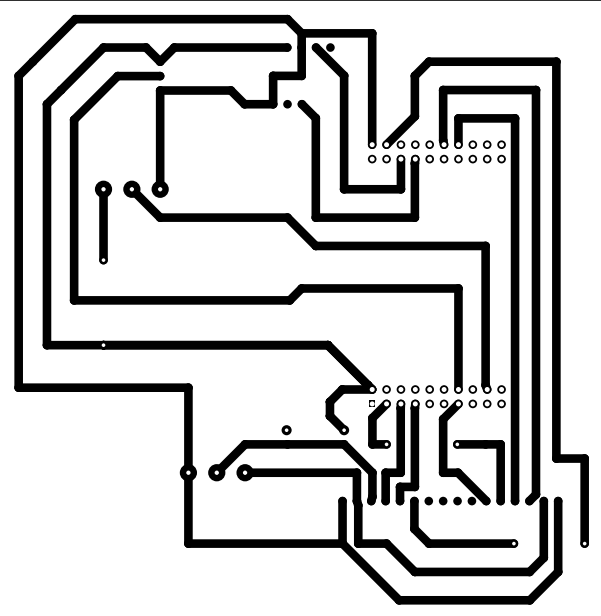
* **Proteus Schematic Diagram:**

****

**Figure 1**

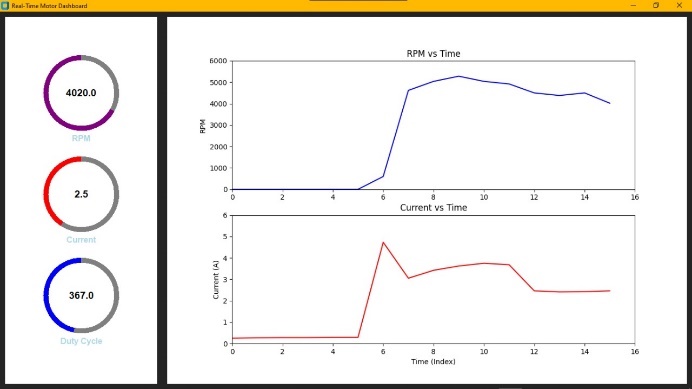
**Proteus Schematic Diagram**

* **PCB layout:**

****

**Figure 13 Figure 14**

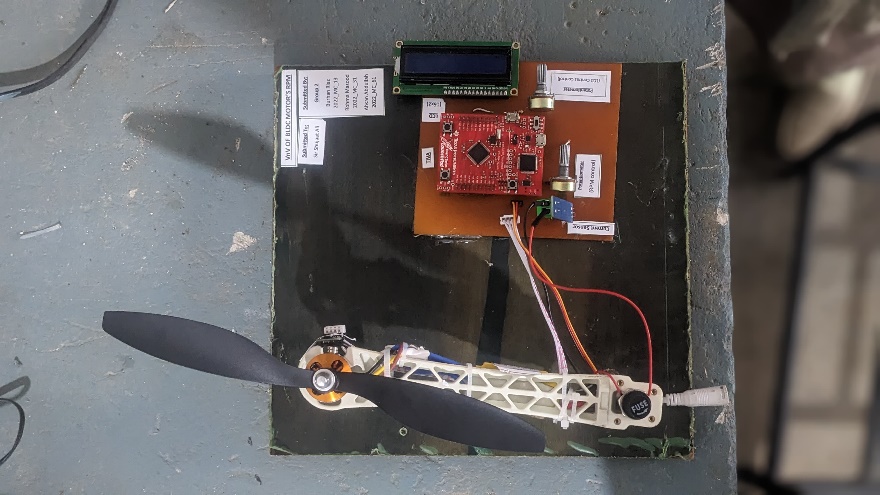
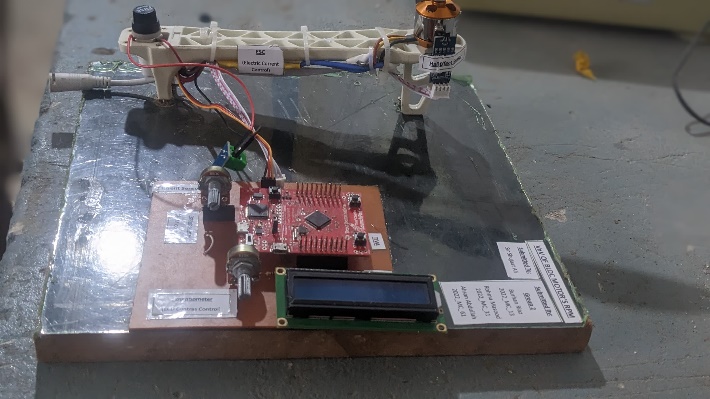
* **Graph:**

****

**Figure 15**

**Graph (RPM and Current vs Time)**

* **Project pictures:**

** **

**Figure 16 Figure 17**

* **Dynamic Motor Dashboard Using Serial Communication:**
* **Purpose:**

To monitor and visualize motor performance in real-time, enabling efficient analysis and logging of key metrics like RPM, current, and duty cycle.

* **Code:**

import customtkinter as ctk

import serial

import threading

from matplotlib.figure import Figure

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from collections import deque

import csv

import os

# Initialize serial communication

ser = serial.Serial('COM28', 9600, timeout=1)

# Global variables

rpm, current, duty = 0.0, 0.0, 0.0

time\_data = deque(maxlen=100)

rpm\_data = deque(maxlen=100)

current\_data = deque(maxlen=100)

# CSV file setup

csv\_file = "motor\_data.csv"

csv\_columns = ["Index", "RPM", "Current (A)", "Duty Cycle (%)"]

# Create or clear the CSV file

if not os.path.exists(csv\_file):

with open(csv\_file, mode='w', newline='') as file:

writer = csv.DictWriter(file, fieldnames=csv\_columns)

writer.writeheader()

# Function to log data to CSV

def log\_to\_csv(index, rpm, current, duty):

"""

Log data to the CSV file.

"""

with open(csv\_file, mode='a', newline='') as file:

writer = csv.DictWriter(file, fieldnames=csv\_columns)

writer.writerow({"Index": index, "RPM": rpm, "Current (A)": current, "Duty Cycle (%)": duty})

# Function to create a modern circular progress bar

def create\_animated\_progress(canvas, x, y, radius, thickness, start\_angle, color, label\_text, theme="dark"):

"""

Create a circular progress bar.

"""

# Background circle

canvas.create\_oval(

x - radius, y - radius, x + radius, y + radius, outline="gray", width=thickness

)

# Dynamic arc

arc = canvas.create\_arc(

x - radius, y - radius, x + radius, y + radius,

start=start\_angle, extent=0, outline=color, width=thickness, style="arc"

)

# Label for displaying the current value

label\_color = "black" if theme == "light" else "white"

label = canvas.create\_text(

x, y, text="0", fill=label\_color, font=("Arial", 14, "bold")

)

# Add a title below the circle

canvas.create\_text(

x, y + radius + 20, text=label\_text, fill="lightblue", font=("Arial", 12, "bold")

)

return arc, label

# Function to update the circular progress bar

def update\_animated\_progress(canvas, bar, percentage, value):

"""

Update the circular progress bar and label dynamically.

"""

arc, label = bar

extent = percentage \* 360

canvas.itemconfig(arc, extent=extent)

canvas.itemconfig(label, text=f"{value:.1f}")

# Initialize customtkinter

ctk.set\_appearance\_mode("dark")

ctk.set\_default\_color\_theme("blue")

# Create the main application window

root = ctk.CTk()

root.title("Real-Time Motor Dashboard")

root.geometry("1400x800")

# Create a canvas for the circular progress bars

rpm\_canvas = ctk.CTkCanvas(root, width=300, height=800, bg="white", highlightthickness=0)

rpm\_canvas.pack(side="left", padx=10, pady=10)

# Create smaller circular progress bars (vertical layout)

rpm\_bar = create\_animated\_progress(rpm\_canvas, 150, 150, 70, 10, 90, "purple", "RPM", theme="light")

current\_bar = create\_animated\_progress(rpm\_canvas, 150, 350, 70, 10, 90, "red", "Current", theme="light")

duty\_bar = create\_animated\_progress(rpm\_canvas, 150, 550, 70, 10, 90, "blue", "Duty Cycle", theme="light")

# Create Matplotlib Figure and Axes

fig = Figure(figsize=(10, 6), dpi=100)

ax1 = fig.add\_subplot(211) # RPM vs Time

ax2 = fig.add\_subplot(212) # Current vs Time

# Set titles and labels

ax1.set\_title("RPM vs Time")

ax1.set\_ylabel("RPM")

ax2.set\_title("Current vs Time")

ax2.set\_xlabel("Time (Index)")

ax2.set\_ylabel("Current (A)")

# Initialize lines for real-time plotting

rpm\_line, = ax1.plot([], [], color='blue', linewidth=1.5)

current\_line, = ax2.plot([], [], color='red', linewidth=1.5)

# Embed Matplotlib Figure in Tkinter

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side="right", fill="both", expand=True, padx=10, pady=10)

# Function to update graphs

def update\_graph\_data(index):

time\_data.append(index)

rpm\_data.append(rpm)

current\_data.append(current)

# Update plots

rpm\_line.set\_data(time\_data, rpm\_data)

current\_line.set\_data(time\_data, current\_data)

# Adjust axis limits

ax1.set\_xlim(max(0, len(time\_data) - 100), len(time\_data))

ax2.set\_xlim(max(0, len(time\_data) - 100), len(time\_data))

ax1.set\_ylim(0, 6000)

ax2.set\_ylim(0, 6)

canvas.draw()

# Function to read data from the serial port

def read\_serial\_data():

global rpm, current, duty

index = 0

while True:

if ser.in\_waiting > 0:

line = ser.readline().decode().strip()

try:

parts = line.split(", ")

rpm = float(parts[0].split(": ")[1])

current = float(parts[1].split(": ")[1].replace("A", ""))

duty = float(parts[2].split(": ")[1])

# Adjust duty cycle scaling (250-500 range)

update\_animated\_progress(rpm\_canvas, rpm\_bar, rpm / 6000, rpm)

update\_animated\_progress(rpm\_canvas, current\_bar, current / 6, current)

update\_animated\_progress(rpm\_canvas, duty\_bar, (duty - 250) / 250, duty)

update\_graph\_data(index)

# Log data to CSV and print in terminal

log\_to\_csv(index, rpm, current, duty)

print(f"Index: {index}, RPM: {rpm}, Current: {current}, Duty: {duty}")

index += 1

except Exception as e:

print(f"Error parsing data: {e}")

# Start serial reading thread

serial\_thread = threading.Thread(target=read\_serial\_data, daemon=True)

serial\_thread.start()

# Run the Tkinter main loop

root.mainloop()

* **Code Explanation:**

1. **Key Steps:**

* **Initialization:**
* **Serial Communication:**

Initializes a connection on COM28 with a 9600 baud rate to receive data from the motor's microcontroller.

* **Global Variables:**

rpm, current, and duty store real-time motor data. deque structures are used for graph data storage (up to 100 points).

* **CSV Setup:**

Prepares a CSV file (motor\_data.csv) to log motor data.

1. **GUI interface:**

A custom tkinter GUI is developed with:

* **Circular Progress Bars:**

Representing RPM, Current, and Duty Cycle values.

* **Matplotlib Graphs:**

 Dynamically plot RPM vs Time and Current vs Time.

1. **Real-Time data handling:**

* A background thread continuously reads data from the serial port.
* The data is parsed, scaled, and used to:
* Update progress bars.
* Update graph data and refresh plots.
* Log data to the CSV file for analysis.
* Print parsed data to the terminal for debugging.

1. **Dynamic Updates:**

* Progress bars and graphs are updated in real-time with new data.
* The CSV file stores every data point for offline review.

1. **Execution:**

* The GUI runs in the main thread, while serial data handling is performed in a separate thread to keep the interface responsive.
* **Workflow Execution:**

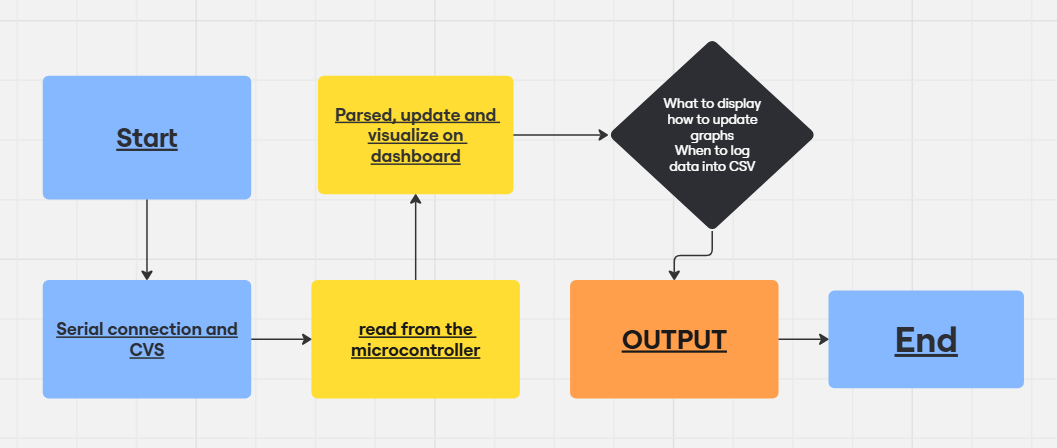
Serial communication begins, and the dashboard waits for incoming data from the motor system.

As data arrives:

* RPM, current, and duty cycle values are extracted.
* Circular progress bars and graphs are updated in real-time.
* Data is logged into a CSV file and printed to the terminal for debugging.

The dashboard continues to display live metrics and update the time-series plots dynamically.

* **Flowchart:**

****

**Flowchart**

**Figure 18**

* **MOTOR RPM’S MAIN CODE:**

**Purpose:**

This code predicts the RPM and current of a motor based on a given duty cycle percentage using a Random Forest Regressor. It also evaluates the model's performance using Mean Squared Error.

**Code:**

* import pandas as pd
* from sklearn.ensemble import RandomForestRegressor
* from sklearn.model\_selection import train\_test\_split
* from sklearn.metrics import mean\_squared\_error
* import numpy as np
* df = 'E:\SOFTWARE PROJECTS\VISUAL STUDIO\python\motor\_data.csv' # Replace with your file path
* data = pd.read\_csv(df)
* data
* Index RPM Current (A) Duty Cycle (%)
* 0 0 0.0 0.23 250.0
* 1 1 0.0 0.23 250.0
* 2 2 0.0 0.24 250.0
* 3 3 0.0 0.24 250.0
* 4 4 0.0 0.24 250.0
* ... ... ... ... ...
* 1739 61 4140.0 3.75 381.0
* 1740 62 4140.0 3.77 381.0
* 1741 63 4200.0 3.74 380.0
* 1742 64 3900.0 3.71 381.0
* 1743 65 4200.0 3.73 381.0
* 1744 rows × 4 columns
* data = data.drop(columns=["Index"])
* data
* Predicted RPM: 4656.00
* Predicted Current (A): 3.48
* C:\Users\TOSHIBA PC\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11\_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
* warnings.warn(
* RandomForestRegressor?i
* RandomForestRegressor(random\_state=42)
* Mean Squared Error: 221455.43613698837
* Index RPM Current (A) Duty Cycle (%)
* 0 0 0.0 0.23 250.0
* 1 1 0.0 0.23 250.0
* 2 2 0.0 0.24 250.0
* 3 3 0.0 0.24 250.0
* 4 4 0.0 0.24 250.0
* ... ... ... ... ...
* 1739 61 4140.0 3.75 381.0
* 1740 62 4140.0 3.77 381.0
* 1741 63 4200.0 3.74 380.0
* 1742 64 3900.0 3.71 381.0
* 1743 65 4200.0 3.73 381.0
* 1744 rows × 4 columns
* RPM Current (A) Duty Cycle (%)
* 0 0.0 0.23 250.0
* 1 0.0 0.23 250.0
* 2 0.0 0.24 250.0
* 3 0.0 0.24 250.0
* 4 0.0 0.24 250.0
* ... ... ... ...
* 1739 4140.0 3.75 381.0
* 1740 4140.0 3.77 381.0
* 1741 4200.0 3.74 380.0
* 1742 3900.0 3.71 381.0
* 1743 4200.0 3.73 381.0
* 1744 rows × 3 columns
* X = data[["Duty Cycle (%)"]]
* y = data[["RPM", "Current (A)"]]
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
* model = RandomForestRegressor(n\_estimators=100, random\_state=42)
* model.fit(X\_train, y\_train)
* y\_pred = model.predict(X\_test)
* #model.score(y\_test, y\_pred)
* # Calculate and display model performance
* mse = mean\_squared\_error(y\_test, y\_pred)
* print(f"Mean Squared Error: {mse}")
* duty\_cycle\_input = float(input("Enter the Duty Cycle (%): "))
* input\_data = np.array([[duty\_cycle\_input]])
* prediction = model.predict(input\_data)
* rpm, current = prediction[0]
* print(f"Predicted RPM: {rpm:.2f}")
* print(f"Predicted Current (A): {current:.2f}")

**Code Explanation:**

**1. Importing Required Libraries**

The code imports essential libraries:

* pandas for handling data in tabular form.
* sklearn for machine learning tasks like model building and evaluation.
* numpy for numerical computations.

**2. Loading the Dataset**

The dataset is loaded using pd.read\_csv() from a CSV file (motor\_data.csv) that contains motor-related parameters:

* **Columns**:
  + Index (dropped later as it's unnecessary)
  + RPM, Current (A), and Duty Cycle (%).

**3. Data Preprocessing**

The Index column is dropped, leaving only relevant features (RPM, Current (A), and Duty Cycle (%)) for analysis and modeling.

**4. Splitting Data into Features and Targets**

* **Features (X)**: Duty Cycle (%)
* **Targets (y)**: RPM and Current (A)  
  This separation helps in training the machine learning model to predict target values based on the input feature.

**5. Train-Test Split**

The dataset is divided into training (80%) and testing (20%) sets using train\_test\_split().  
This ensures the model is trained on one portion of the data and tested on unseen data.

**6. Building and Training the Model**

A RandomForestRegressor is initialized with 100 estimators and a fixed random seed for reproducibility.  
The model is trained on the X\_train and y\_train datasets using model.fit().

**7. Making Predictions and Evaluating Performance**

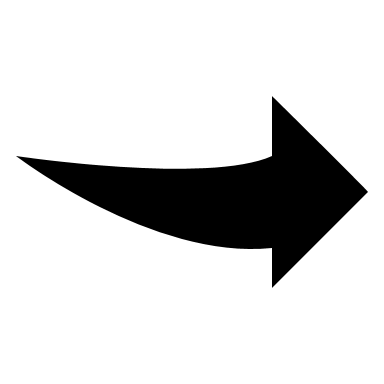
* The trained model predicts the RPM and Current (A) values for the test set (X\_test).
* Performance is measured using **Mean Squared Error (MSE)**, which calculates the average squared differences between predicted and actual values.

**8. Taking User Input and Making New Predictions**

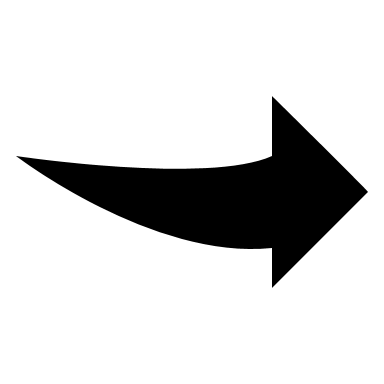
* The user provides a Duty Cycle (%) value via input.
* The model predicts the corresponding **RPM** and **Current (A)** based on this input using model.predict().
* The predictions are displayed with two decimal places.

**Purpose Summary**

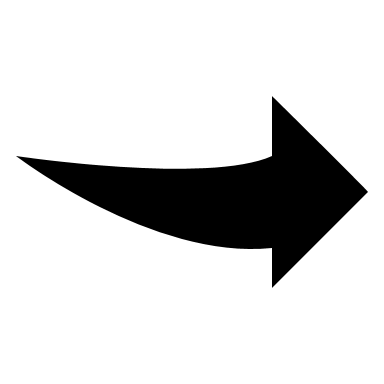
* Train a Random Forest model to predict motor performance metrics (RPM and Current) based on the Duty Cycle.
* Allow users to input new Duty Cycle values to generate real-time predictions.
* **Flowchart:**



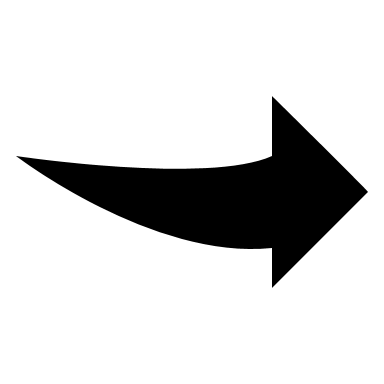
Import Libraries



Load and preprocess dataset

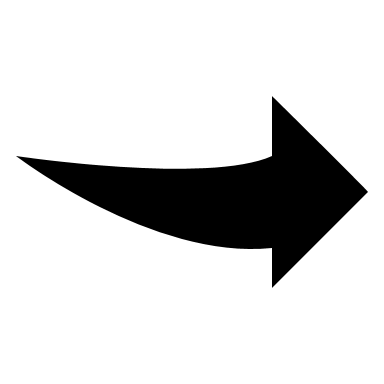


Split dataset

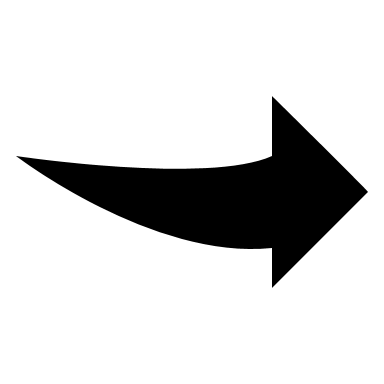


Train dataset

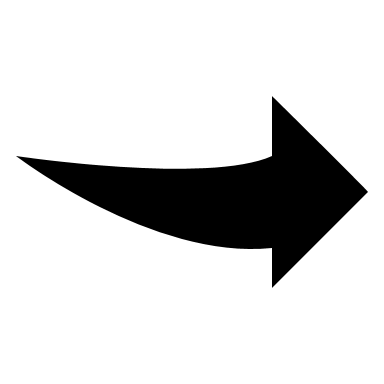
Perform train test split



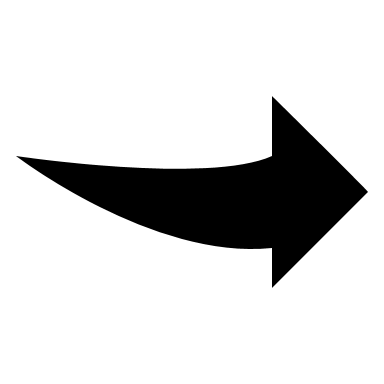
Random Forest Classifier initilize



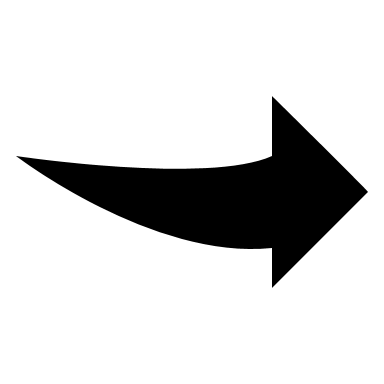
Train the model



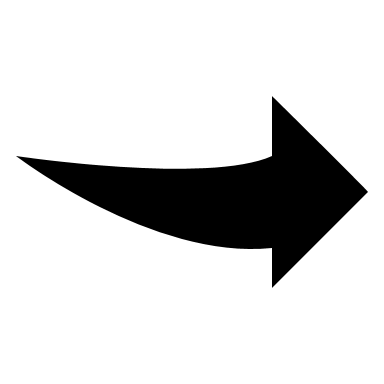
Predictions on dataset



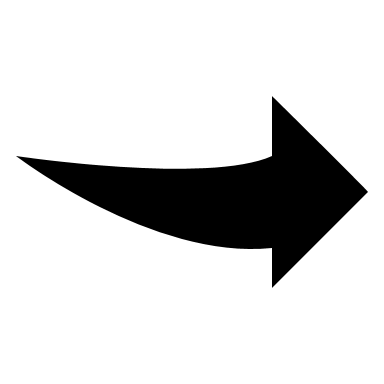
Calculate mean sq error



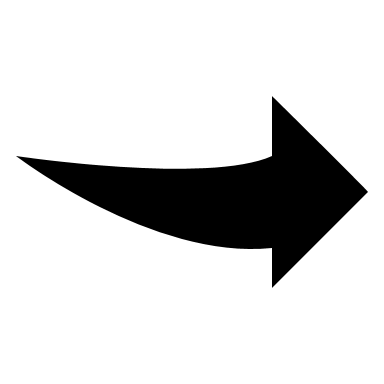
Take user input for duty cycle



Predict RPM and current for input



Display Predictions



* **Bill of Materials:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Component** | **Description** | **Application** | **Quantity** | **Price**  **(unit)**  **(Rs.)** | **Total Price (Rs.)** | **Status** | **Vendor Info** |
| **1** | **LCD**  (16x2) | A flat-panel display that uses liquid crystals to control light for visuals. | **1)**Display screens in smartphones.  **2)**Digital watches and clocks. | **1** | **290** | **290** | **Active** | **Epro[[10]](#footnote-10)** |
| **2** | **BLDC**  (A2212 1400KV) | A motor with no brushes, offering high efficiency, reliability, and precise control. | **1)**Electric vehicle propulsion.  **2)**Fan speed control. | **1** | **1150** | **1150** | **Active** | **Epro[[11]](#footnote-11)** |
| **3** | **Current Sensor**  (ACS712 Range-current) | A device that measures and monitors electric current flow. | **1)**Monitoring power consumption in devices.  **2)**Overcurrent protection in electrical systems. | **1** | **320** | **320** | **Active** | **Epro[[12]](#footnote-12)** |
| **4** | **Hall Effect Sensor**  **module** | A device that detects magnetic fields and converts them into electrical signals. | **1)**Position and speed sensing in motors.  **2)**Proximity and magnetic field detection. | **1** | **130** | **130** | **Active** | **Epro[[13]](#footnote-13)** |
| **5** | **ESC**  (Electric Speed Control) |  |  | **1** | **1350** | **1350** | **Active** | **Epro[[14]](#footnote-14)** |
| **5** | **Magnets** | Objects that produce a magnetic field, attracting certain materials like iron. | **1)**Magnetic sensors | **1** | **174** | **174** | **Active** | **Epro[[15]](#footnote-15)** |
| **6** | **Potentiometer** | A variable resistor used to adjust voltage levels in a circuit. | **1)**Volume control in audio devices.  **2)**Adjustable voltage dividers in circuits. | **2** | **60** | **120** | **Active** | **Epro[[16]](#footnote-16)** |
| **7** | **Fuse** | A safety device that protects circuits by breaking the connection during overloads | **1)**Overcurrent protection in electrical appliances.  **2)**Circuit protection in power supplies. | **1** | **50** | **50** | **Active** | **Epro[[17]](#footnote-17)** |
| **8** | **Tiva**  (TM4C123GH6PM) | A microcontroller series by Texas Instruments designed for high-performance embedded applications. | **1)**Real-time control systems.  **2)**IoT and automation devices. | **1** | Already available | **-** | **Active** | **-** |
| **9** | **Jumper Wires** | Insulated wires used to connect components in a circuit. | Connecting components in electronic projects | **-** | Already available | **-** | **Active** | **-** |
| **10** | **PCB** | A board used to mechanically support and electrically connect electronic components. | **1)**Electronics device assembly.  **2)**Signal routing in circuit boards. | **1** | Already available | **-** | **Active** | **-** |
| **11** | **Frame** | - | **-** | **1** | **750** | **750** | **Active** | **-** |
| **12** | **Wood work** | - | **-** |  | **500** | **500** | **Active** | **-** |
| **13** | **Adapter** | - | **-** | **1** | **620** | **620** | **Active** | **Epro[[18]](#footnote-18)** |
| **14** | **Optocoupler** | A device that transfers electrical signals using light for isolation between circuits. | **1)**signal transmission  **2)**Isolation in power electronics | **1** | **150** | **150** | **Not used** | **Epro[[19]](#footnote-19)** |
| **15** | **Hall effect sensor 44E** | A device that detects magnetic fields and converts them into electrical signals. | **1)**Position and speed sensing in motors.  **2)**Proximity and magnetic field detection. | **1** | **95** | **95** | **Not used** | **Epro[[20]](#footnote-20)** |
| **16** | **TA/DA (Fare)** | - | **-** | **-** | **600** | **600** | **-** | **-** |
| **17** | **Total Price** | **-** | **-** | **-** | **6239** | **6299** | **-** | **-** |

* **Challenges Faced:**
* **CONCLUSION:**

In conclusion, the embedded system project successfully integrates various components such as an ESC, BLDC motor, Hall effect sensor, current sensor, potentiometers, and an LCD to create a comprehensive motor control and monitoring system. The ESC manages the current supplied to the BLDC motor, while a fuse ensures voltage overflow protection, safeguarding the system from potential damage. The Hall effect sensor accurately measures the motor’s RPM, and the current sensor provides real-time current data for effective motor monitoring. The system offers user interaction through potentiometers—one to control the motor’s RPM and the other to adjust the LCD brightness. The LCD serves as an essential output display, providing the operator with real-time feedback on RPM and current levels. This project demonstrates a practical and efficient way to control and monitor motor performance, with real-time data displayed for easy assessment and adjustment. Overall, this embedded system offers valuable insights into motor control, safety mechanisms, and user interface integration, making it suitable for applications in robotics, drones, and other motor-driven systems.

1. <https://epro.pk/product/a2212-1400kv-brushless-dc-motor/> [↑](#footnote-ref-1)
2. <https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/> [↑](#footnote-ref-2)
3. <https://components101.com/sensors/acs712-current-sensor-module> [↑](#footnote-ref-3)
4. <https://components101.com/sensors/hall-effect-sensor-module> [↑](#footnote-ref-4)
5. <https://microcontrollerslab.com/uart-communication-tm4c123-tiva-c-launchpad/> [↑](#footnote-ref-5)
6. <https://www.daraz.pk/products/ac-to-dc-12v-2a-power-supply-power-adapter-for-modem-router-imported-stock-i432752184.html> [↑](#footnote-ref-6)
7. <https://www.amazon.com/RC-Brushless-Electric-Controller-bullet/dp/B071GRSFBD?th=1> [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Fuse_%28electrical%29> [↑](#footnote-ref-8)
9. <https://www.pinterest.com/pin/pinout-of-a-potentiometer--802344489874975605/> [↑](#footnote-ref-9)
10. <https://epro.pk/product/1602-character-lcd-16x2-green-color-in-pakistan/> [↑](#footnote-ref-10)
11. <https://epro.pk/product/a2212-1000kv-brushless-dc-motor/> [↑](#footnote-ref-11)
12. <https://epro.pk/product/acs712-range-current-sensor-module-of-5a-20a-30a-in-pakistan/> [↑](#footnote-ref-12)
13. <https://epro.pk/product/hall-effect-sensor-module-rpm-sensor-speed-sensor/> [↑](#footnote-ref-13)
14. <https://epro.pk/product/electronic-speed-control-esc-30-amp-with-bullet-connector-in-pakistan/> [↑](#footnote-ref-14)
15. <https://www.daraz.pk/products/156mm-original-neodymium-magnet-n52-for-magnetic-experiments-i375708967-s1918513519.html>? [↑](#footnote-ref-15)
16. <https://epro.pk/product/variable-resistor-potentiometer-volume-5k-10k-50k-100k/> [↑](#footnote-ref-16)
17. <https://epro.pk/product/agc-5a-panel-mount-screw-cap-fuse-holder-case-6x30mm-in-pakistan/> [↑](#footnote-ref-17)
18. <https://epro.pk/product/12v-5amp-high-quality-power-supply-adapter/> [↑](#footnote-ref-18)
19. <https://epro.pk/product/a3120-hcpl3120-hcpl-3120-sop-mop-driver-in-pakistan/> [↑](#footnote-ref-19)
20. <https://epro.pk/?s=hall+effect+sensor&post_type=product&product_cat=0> [↑](#footnote-ref-20)