Real-Time Monitoring of Single Phase Induction Motor Parameters Using ESP32 for Safety Purposes

Submitted by

Rushank Talwar (22BEE1058)

Roshan Mohammed Ali (22BEE1042)

Faculty - Dr. IYSWARYA ANNAPOORANI K

for the course

BEEE312L– AC MACHINES

Fall 2024– 2025

School of Electrical Engineering

VIT®

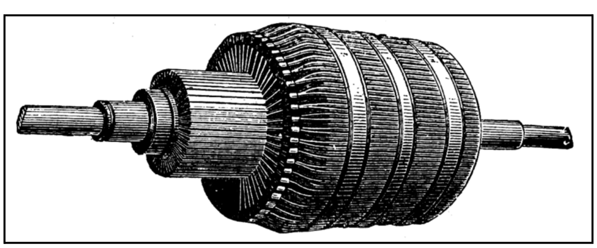
Vellore Institute of Technology

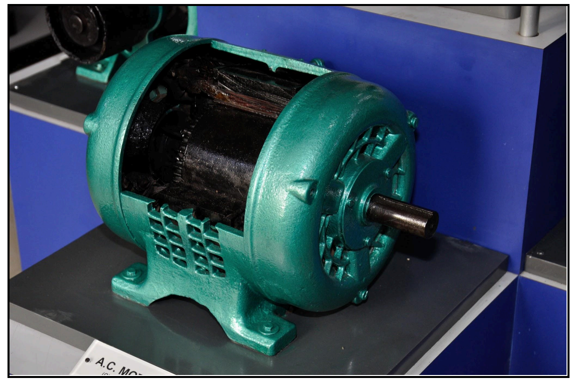
(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Abstract

This project aims to develop a real-time monitoring system for single-phase induction motors, utilizing the ESP32 microcontroller to enhance safety and operational efficiency. Single-phase induction motors are widely used in various applications due to their simplicity and reliability, but they are also prone to faults and operational issues that can lead to inefficiency and safety hazards. By integrating sensors with an ESP32 microcontroller, this project monitors key motor parameters, such as voltage, current, and speed, in real-time. The data is transmitted wirelessly to a remote monitoring platform, enabling timely intervention in case of anomalies. Additionally, the system incorporates threshold-based alerts to notify operators of critical conditions, reducing the risk of motor damage and enhancing overall safety. This solution is designed to be cost-effective, scalable, and suitable for applications requiring real-time monitoring and control, contributing to improved motor longevity, reduced downtime, and enhanced safety in industrial and domestic environments.





Components Required

Materials Required -

- **Sensors :**
    1. **ZMPT101B** - For measuring the AC voltage supplied to the motor.
    2. **ACS712 Current Sensor Module (5A/20A/30A)** - For measuring the current drawn by the motor.
    3. **Inductive Proximity Sensor** - For detecting the rotational speed of the motor by counting the pulses from a metallic mark on the motor shaft.

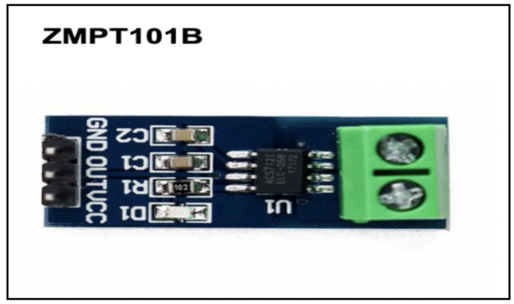
- **Microcontroller :**
    1. **ESP 32 WROOM (Wifi + Bluetooth)** - Used for processing sensor data, handling communication, and controlling the monitoring system.

- **Power Supply**

    1. **5V** for Voltage , Current , Temperature and Microcontroller.
    2. **12 V** for Inductive Proximity sensor.

- **Resistors**
    1. **Voltage divider circuit(**
    2. **27kΩ and 10kΩ)** - For the inductive proximity sensor.

- **OTHERS**
    1. Breadboard
    2. Jumper Wires
    3. PCB (if permanent circuit)
    4. Arduino IDE
    5. Heat Shrink Tubes
    6. Connectors (if PCB is opted)
    7. Soldering setup
    8. LED



Implementation

This project utilizes an ESP32 microcontroller to monitor essential parameters of a single-phase induction motor in real-time, aiming to boost both safety and operational efficiency. The ZMPT101B sensor measures AC voltage, the ACS712 sensor captures motor current, and an inductive proximity sensor detects RPM by counting pulses generated from motor rotation. Each sensor’s data is processed by the ESP32, which continuously checks if any reading reaches or exceeds rated thresholds. If any parameter crosses its limit, an LED connected to the ESP32 blinks, serving as an immediate visual warning. This setup ensures quick intervention, protecting against potential issues like overloads and unsafe operation.

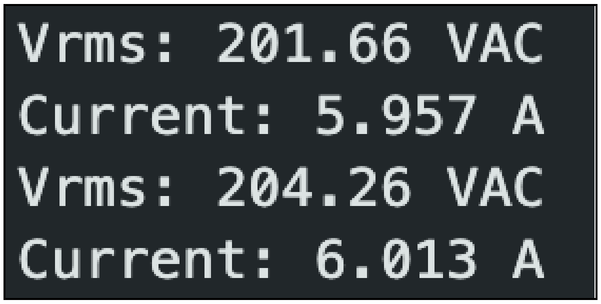
Compiled Code

```
// Voltage sensor variables
int adc_max1 = 2700;                //
Adjusted maximum ADC value for ESP32
int adc_min1 = 1500;                //
Adjusted minimum ADC value for ESP32
float volt_multi = 231;             //
Nominal AC RMS voltage for scaling
float volt_multi_p;                 //
Positive peak voltage multiplier
float volt_multi_n;                 //
Negative peak voltage multiplier
float volt_rms1 = 0;
int Voltage_Sensor_1 = 35;          //
Use GPIO 35 on ESP32 for voltage sensor

// Current sensor variables
int Current_Sensor_Pin = 34;        //
Use GPIO 34 on ESP32 for current sensor
float current = 0;

// LED setup
int LED_Pin = 2;                    //
Use GPIO 2 for LED output (change if
needed)
float voltage_threshold = 240.0;    //
Voltage threshold in VAC
float current_threshold = 6.0;      //
Current threshold in Amps

void setup() {
 Serial.begin(115200);
  // Voltage sensor setup
    pinMode(Voltage_Sensor_1,    INPUT);
// Set voltage sensor pin as input
  volt_multi_p = volt_multi * 1.4142;
// Calculate positive peak multiplier
    volt_multi_n    =    -volt_multi_p;
// Calculate negative peak multiplier

 // Current sensor setup
   pinMode(Current_Sensor_Pin,   INPUT);
// Set current sensor pin as input
  // LED setup
        pinMode(LED_Pin,        OUTPUT);
// Set LED pin as output
       digitalWrite(LED_Pin,       LOW);
// Start with LED off
}

void loop() {
 // Read and calculate RMS voltage
               volt_rms1               =
get_voltage(Voltage_Sensor_1, adc_min1,
adc_max1);
 Serial.print("Vrms: ");
    Serial.print(volt_rms1    -    253);
// Adjust for nominal value
 Serial.println(" VAC");

  // Read and calculate current from
ACS712 sensor
                current                =
get_current(Current_Sensor_Pin);
 Serial.print("Current: ");
   Serial.print(current   +   16,   3);
// Adjusted for offset
 Serial.println(" A");

  //  Check  if  voltage  or  current
threshold is crossed
  if  ((volt_rms1  >=  voltage_threshold)
|| (current >= current_threshold)) {
          digitalWrite(LED_Pin,   HIGH);
// Turn on LED
 } else {
           digitalWrite(LED_Pin,   LOW);
// Turn off LED
 }

 delay(300);  // Delay for stability
}
```

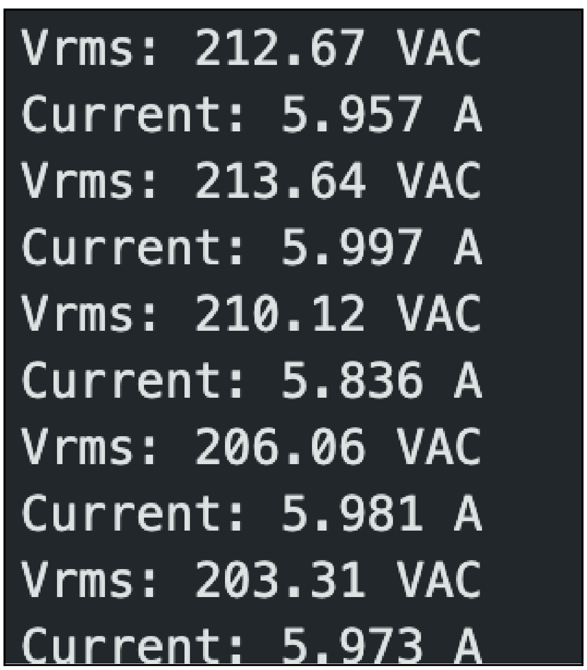
```
// Function to calculate RMS voltage
float get_voltage(int pin, int adc_min, int adc_max) {
 float adc_sample;
 float volt_inst = 0;
 float sum = 0;
 float volt;
 long init_time = millis();
 int N = 0;

 while ((millis() - init_time) < 100) {
   adc_sample = analogRead(pin);
    // Map ADC value from adc_min to adc_max to volt_multi_n and volt_multi_p range
    volt_inst = map(adc_sample, adc_min, adc_max, volt_multi_n * 100, volt_multi_p * 100) / 100.0;
   sum += sq(volt_inst);
   N++;
   delay(1);
 }
 volt = sqrt(sum / N); // Calculate RMS voltage
 return volt;
}

// Function to calculate current based on ACS712 sensor
float get_current(int pin) {
   int adc = analogRead(pin); // Read analog value from current sensor
 float voltage = adc * (3.3 / 4095.0); // Convert ADC value to voltage (ESP32 uses 3.3V reference)

  // Calculate current (assuming 2.5V zero current and 0.1V per amp sensitivity for ACS712-20A model)
 float current = (voltage - 2.5) / 0.1;

  // Apply threshold to ignore small noise
 if (abs(current) < 0.16) {
   current = 0;
 }
 return current;
}
```

Output

```
Vrms: 208.85 VAC
Current: 5.981 A
Vrms: 204.75 VAC
Current: 5.965 A
Vrms: 211.72 VAC
Current: 5.989 A
Vrms: 215.79 VAC
Current: 5.916 A
Vrms: 216.01 VAC
Current: 6.013 A
Vrms: 211.99 VAC
Current: 5.916 A
Vrms: 207.87 VAC
Current: 5.925 A
Vrms: 209.56 VAC
Current: 5.965 A
Vrms: 214.01 VAC
Current: 5.852 A
Vrms: 210.49 VAC
Current: 5.957 A
Vrms: 206.53 VAC
Current: 5.957 A
Vrms: 197.69 VAC
Current: 5.973 A
Vrms: 194.50 VAC
Current: 5.916 A
Vrms: 213.49 VAC
Current: 5.908 A
Vrms: 208.94 VAC
Current: 5.941 A
Vrms: 208.13 VAC
Current: 5.949 A
Vrms: 208.99 VAC
Current: 5.908 A
Vrms: 218.30 VAC
Current: 5.916 A
```

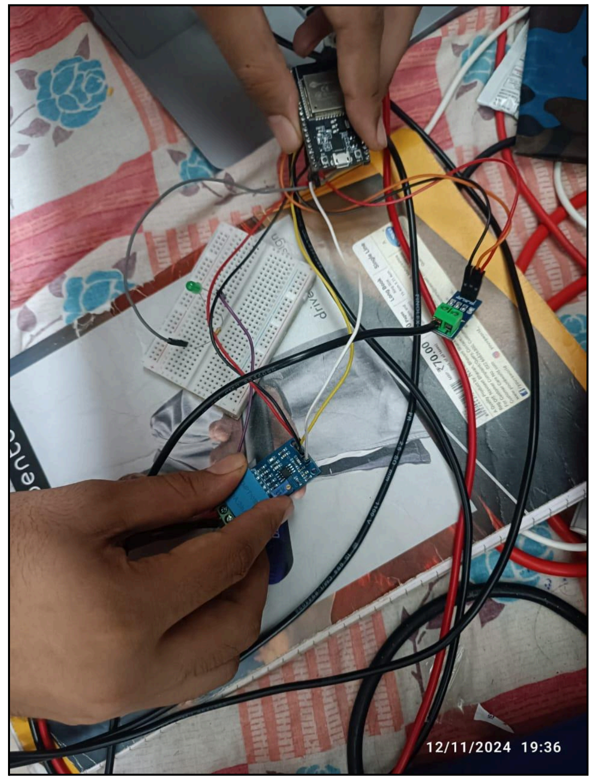


```
Vrms: 201.66 VAC
Current: 5.957 A
Vrms: 204.26 VAC
Current: 6.013 A
```

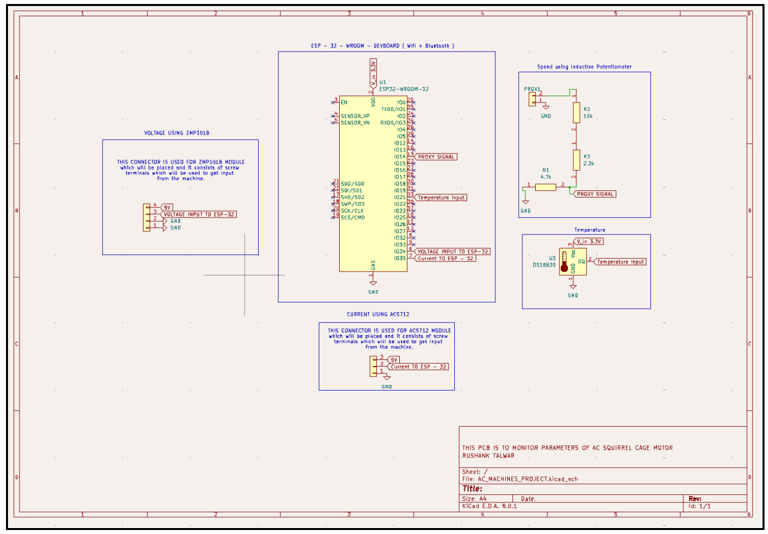


```
Vrms: 212.67 VAC
Current: 5.957 A
Vrms: 213.64 VAC
Current: 5.997 A
Vrms: 210.12 VAC
Current: 5.836 A
Vrms: 206.06 VAC
Current: 5.981 A
Vrms: 203.31 VAC
Current: 5.973 A
```

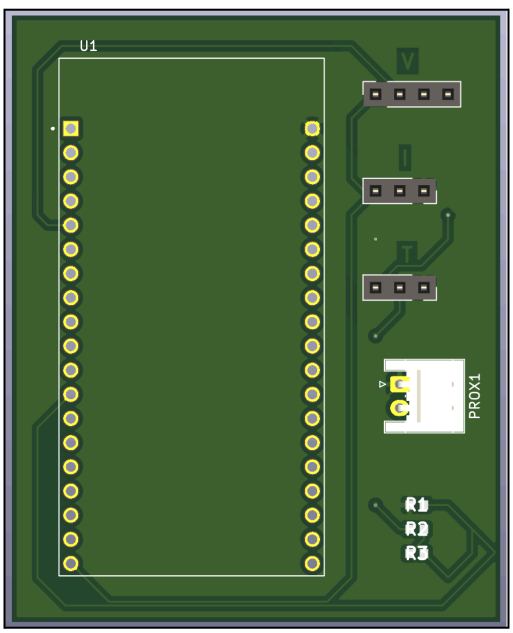
Experimental Setup



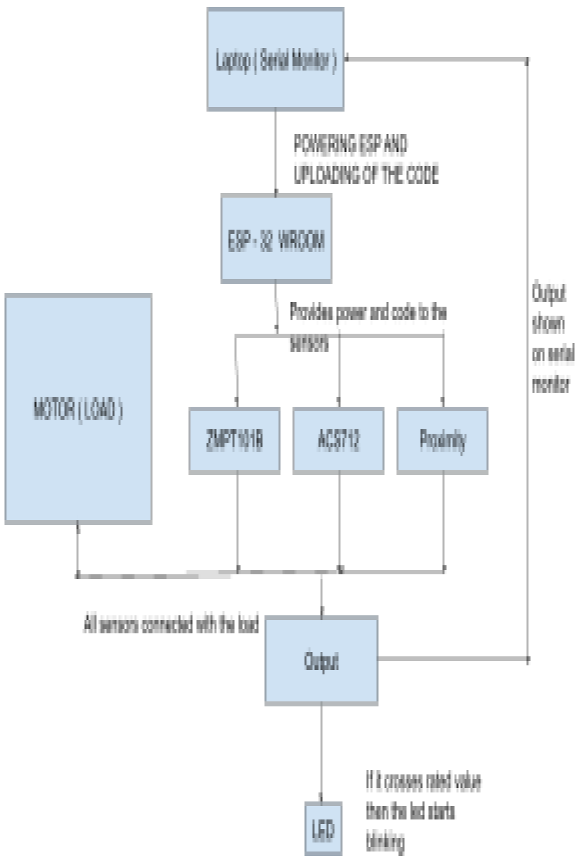
PCB



For a more specialized and professional solution we should try implementing this on a PCB so that the connections are fixed and there are less external factors acting on it like noise , connection issues , etc. This would also be easily scalable as it can be easily placed on every setup and enhance the safety.



Block Diagram



Conclusion

The real-time monitoring system for single-phase induction motors using ESP32 effectively enhances operational safety by promptly detecting abnormal conditions. By integrating ZMPT101B, ACS712, and a proximity sensor, the system accurately tracks voltage, current, and speed parameters. When any parameter exceeds its rated limit, the blinking LED provides an immediate alert, enabling preventive action and minimizing risks of motor failure due to overload or excessive speed. This setup is cost-effective, easily scalable, and suitable for various applications, ensuring continuous monitoring and proactive safety measures that improve motor reliability and extend operational lifespan.

Result

The monitoring system for the single-phase induction motor demonstrated reliable real-time tracking of voltage, current, and speed parameters using ESP32. Each sensor (ZMPT101B for voltage, ACS712 for current, and a proximity sensor for speed) consistently provided accurate readings. During testing, the ESP32 successfully detected when any parameter exceeded its rated threshold, triggering the LED to blink immediately. This visual alert effectively warned of potentially unsafe operating conditions, enabling rapid intervention. The system proved responsive and effective in detecting over-voltage, over-current, and excessive speed, confirming its utility for maintaining motor safety and protecting against damage.