Table of Contents

1.INTRODUCTION

In modern industrial environments, the continuous operation of critical assets such as pumps and motors is essential for productivity, safety, and cost-efficiency. Unexpected failures of these machines can lead to unplanned downtime, reduced output, and significant financial losses. To address this, industries are increasingly adopting Predictive Maintenance (PdM) strategies enabled by the Industrial Internet of Things (IIoT). This project aims to implement a predictive maintenance system specifically focused on industrial pumps and motors, leveraging real-time data acquisition, analytics, and cloud connectivity.

Predictive maintenance involves monitoring the health and performance of equipment using various sensors and forecasting potential failures before they occur. Unlike reactive or scheduled maintenance, PdM helps optimize maintenance schedules, reduce operational costs, and extend equipment lifespan. By integrating sensors with embedded systems and wireless communication, IIoT allows for seamless data collection and transmission to cloud-based platforms for analysis using machine learning or threshold-based models.

In this project, temperature, and voltage sensors are used to gather operational data from pumps and motors. The data is processed and analyzed to detect anomalies and trends that indicate possible mechanical issues. An alert system notifies operators in advance, enabling timely interventions and minimizing disruptions. The system can be scaled and adapted for a variety of industrial setups, aligning with Industry 4.0 goals of smart, automated, and connected manufacturing.

This report details the design, implementation, and evaluation of the predictive maintenance system, emphasizing how IIoT technologies can transform traditional maintenance practices into intelligent, data-driven operations.

2.LITERATURE SURVEY

This paper explores the integration of IoT-based sensors with machine learning algorithms to monitor and predict the failure of industrial equipment. Vibration and temperature data were collected from motors and analyzed using Support Vector Machines (SVM) and Random Forest models. The study reported a 92% accuracy in fault prediction and highlighted the importance of real-time cloud connectivity for proactive maintenance.[1]

The authors proposed an end-to-end IIoT system for monitoring rotating equipment like motors and pumps. The system utilized edge devices to pre-process data and send meaningful parameters to a cloud platform. Time-domain and frequency-domain features were extracted for fault classification. The study demonstrated the reduction of unplanned downtime and improved asset utilization.[2]

This work focused specifically on industrial pumps, where vibration sensors were attached to detect bearing faults and cavitation. An Arduino-based system was used for local data acquisition, which was transmitted over Wi-Fi to a cloud dashboard. The system improved detection speed and helped maintenance teams prioritize inspections.[3]

This research paper presents an IoT-based predictive maintenance system specifically designed for industrial motors. It explains how sensors like temperature and current sensors can be integrated with IoT devices to monitor equipment health. The study highlights real-time monitoring and alert systems that help reduce sudden machine failures. It also discusses how cloud platforms can store and process collected data for predictive analysis. The system improves industrial productivity by minimizing downtime. It provides valuable insights for implementing sensor-based IIoT systems. [4]

3.SYSTEM DETAILS

3.1 PROJECT SPECIFICATION

1. Hardware Components:

Component	Specification
Microcontroller	ESP32-WROOM (dual-core, Wi-Fi & Bluetooth enabled)
Display	16x2 LCD (LM044L) with parallel interface
Transformer	Step-down Transformer (220V to 12V AC)
Bridge Rectifier	BR1 – Converts AC to DC for voltage sensing
Optocoupler	PC817 – Used for frequency detection via zero-crossing
Relay Module	12V Relay – Controls the connected AC Load
AC Load	Represented by L1 (e.g., Lamp, Motor, etc.)
Resistors/Capacitors	For signal conditioning and filtering

2. Software Tools:

- Proteus 8 Professional – for circuit simulation
- Arduino IDE – to program ESP32
- Blynk IOT– for IoT-based data logging and alerting

3. Functional Specifications:

- Voltage Measurement:
  AC voltage is stepped down, rectified, and fed into ATMEGA328P for analog reading after conditioning.

- Temperature Measurement:
  DS18B20 is a analog temperature sensor ,which is measure temperature and fed into ATMEGA328P for conditioning.

- Display Output:
  Real-time voltage and frequency are displayed on a 16x2 LCD.

- Load Control:
  A relay is triggered to turn the load ON/OFF based on set voltage/frequency thresholds (e.g., under-voltage or over-frequency conditions).

- IoT Extension (optional):
  The measured parameters can be uploaded to an IoT cloud platform for remote monitoring and alerting.

4. Power Supply:

- AC Input: 220V, stepped down to 12V

- DC Supply: Derived from rectified AC for circuit powering and ESP32 operation

Communication Infrastructure:

In a predictive maintenance system for industrial pumps and motors, the communication infrastructure forms the backbone of data acquisition, transmission, processing, and alert generation. It enables seamless interaction between physical equipment, sensors, microcontrollers, cloud platforms, and user interfaces. This infrastructure combines both hardware and software components to ensure real-time and reliable data flow.

1. Sensor Layer (Data Acquisition)

- Sensors such as vibration sensors (e.g., ADXL345), temperature sensors (e.g., DHT22), and current sensors (e.g., ACS712) are installed on pumps and motors.
- These sensors continuously monitor machine health and convert physical parameters into electrical signals.
- The ESP32 microcontroller reads these sensor values using its ADC and I2C/SPI interfaces.

2. Edge Layer (Processing & Local Control)

- The ESP32 acts as an edge device, locally processing the data (e.g., averaging, filtering, threshold checking).
- Basic anomaly detection or trend identification can be implemented at this layer to reduce unnecessary data transmission.
- The ESP32 also controls a relay module to take immediate action (e.g., shut down equipment) if a critical fault is detected.

3. Network Layer (Wireless Communication)

- The ESP32 has built-in Wi-Fi capabilities to connect to a local network (router or hotspot).
- Sensor data is transmitted wirelessly from the ESP32 to a cloud server or IoT platform (e.g., Firebase, ThingsBoard, Blynk, or ThingSpeak).

4. Cloud Layer (Data Storage and Analytics)

- The data received from edge devices is stored and analyzed on a cloud-based IoT platform.
- The cloud infrastructure supports data logging, historical analysis, visualization through dashboards, and machine learning-based predictive modeling (if implemented).
- Alerts and maintenance schedules can be generated based on cloud-side analysis.

5. Application Layer (User Interaction)

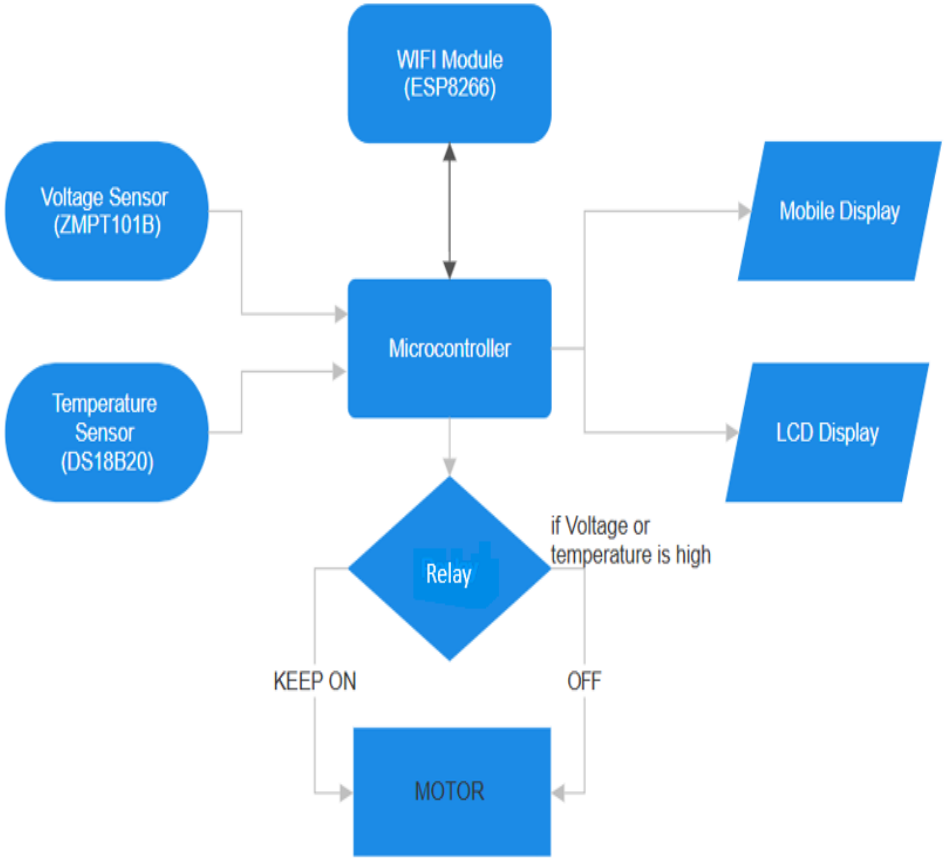
- Users (maintenance engineers or supervisors) can access the cloud platform through a web or mobile app interface.
- The interface displays real-time data, system health status, alerts, and historical trends.
- SMS, email, or push notifications are triggered when abnormal conditions are detected.

Communication Technologies Used

Layer	Technology Used	Description
Sensor to ESP32	I2C, ADC, GPIO	For acquiring data from sensors
ESP32 to Cloud	Wi-Fi (802.11 b/g/n)	For transmitting data to cloud platform
Cloud	Blynk IOT	For storing, analyzing, and visualizing
User Interface	Mobile App	For user access and monitoring

This infrastructure ensures a robust, scalable, and real-time communication system suitable for smart industrial maintenance.

3.2 BLOCK DIAGRAM



3.3 DESCRIPTION OF BLOCKS

1. **Voltage Sensor (ZMPT101B):**
   This sensor is used to measure the AC voltage supplied to the motor. It provides an analog voltage signal proportional to the actual AC voltage, which is fed into the microcontroller for monitoring and comparison against safe operational thresholds.

2. **Temperature Sensor (DS18B20):**
   The DS18B20 is a digital temperature sensor that monitors the operating temperature of the motor. It communicates with the microcontroller via the 1-Wire protocol and helps detect overheating conditions that may indicate motor strain or impending failure.

3. **Microcontroller:**
   The central control unit of the system. It reads data from the voltage and temperature sensors, processes it, and makes decisions based on predefined thresholds. If any parameter exceeds the safe limit, the microcontroller triggers the relay to turn off the motor and sends data to the Wi-Fi module for remote monitoring**.**

4. **Wi-Fi Module (ESP8266):**
   The ESP8266 module provides wireless connectivity to send sensor data and system status to a mobile or cloud platform. It allows real-time monitoring through a mobile application or web dashboard, enabling remote diagnostics and alerts.

5. **LCD Display:**
   A 16x2 character LCD display connected to the microcontroller is used to show real-time readings of voltage and temperature locally. It serves as a user-friendly interface for on-site personnel.

6. **Mobile Display (App):**
   The mobile interface receives data from the ESP8266 over Wi-Fi and displays it to the user in a remote location. This interface can also be configured to send alerts when critical conditions are detected (e.g., overheating or overvoltage).

7. **Relay Module:**
   Acts as an electronic switch controlled by the microcontroller. It is used to cut off or maintain the power supply to the motor based on sensor readings. If voltage or temperature exceeds the safe limit, the relay disconnects the motor to prevent damage.

8. **Motor:**
   The motor is the load being monitored and controlled in this system. It receives power through the relay and operates normally until an unsafe condition is detected. If any anomaly is observed, the system turns off the motor automatically to avoid failure

4.SYSTEM DESIGN

4.1 DESIGN

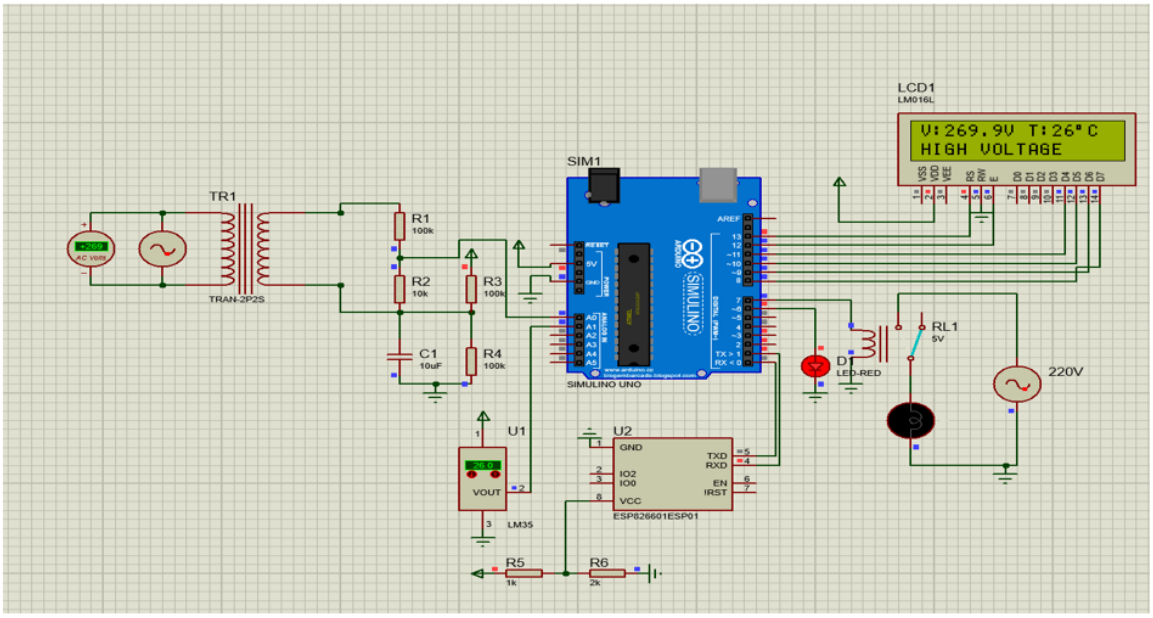


Fig 4.1

4.2 INTERFACING DIAGRAM & EXPLAINATION

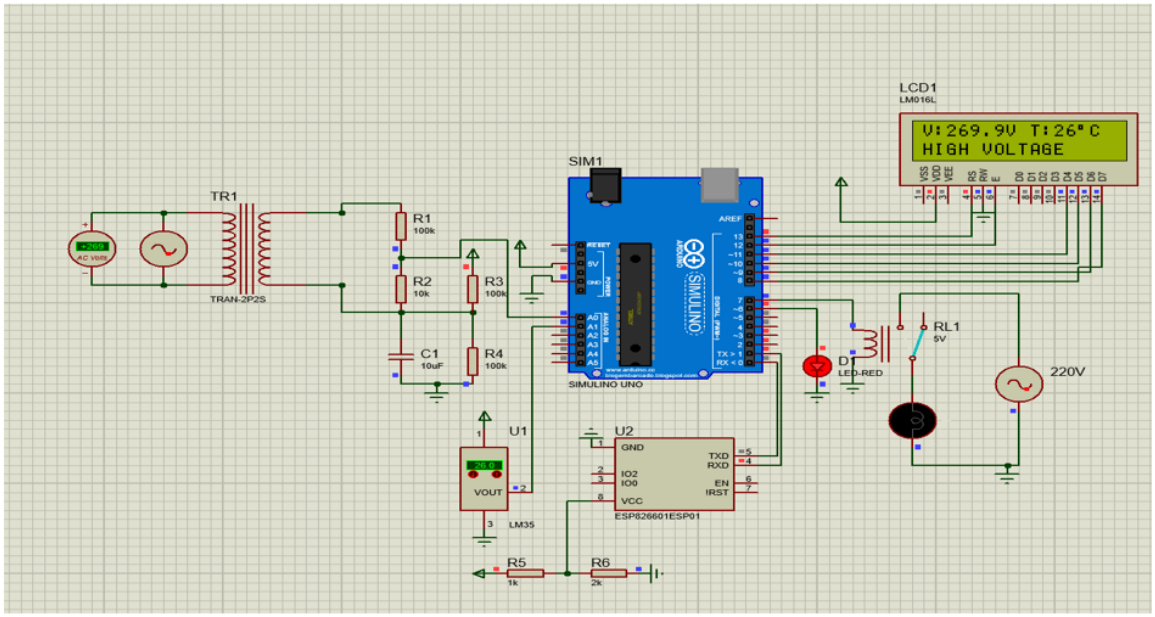


Fig 4.2

Component List:

1. Microcontroller

- Arduino UNO (SIMULINO UNO)
  Central controller to read sensors, control outputs, and communicate with the ESP8266 and LCD.

2. Sensors

- Voltage Sensor Circuit (ZMPT101B Equivalent)
  - Transformer (TR1)
  - Resistors: R1 (100kΩ), R2 (10kΩ), R3 (100kΩ), R4 (100kΩ)
  - Capacitor: C1 (10µF)
    Used to scale and condition the AC voltage for analog input.
- **Temperature Sensor**
  - LM35 (U1): Analog temperature sensor
  - Resistors: R5 (1kΩ) and R6 (unspecified, likely pull-down or current-limiting)

3. Communication Module

- ESP8266-01 (U2)
  Used for wireless data communication (Wi-Fi) to transmit sensor data to cloud or mobile display.
  - Pins: TX, RX, VCC, GND, CH_PD (EN), GPIO0, GPIO2, RST

4. Output Devices

- Relay Module (RL1, 5V)
  Controls AC load (motor or any 220V device) based on sensor readings.
- Red LED (D1)
  Indicates fault/high voltage/temperature condition.

5. Display

- 16x2 LCD (LCD1, LM016L)
  Displays real-time sensor readings (voltage, temperature) and system status (e.g., “HIGH VOLTAGE”).

6. Power Supply & AC Source

- AC Voltage Source (left side)
  Simulated input voltage to be monitored.
- 220V AC Load (right side)
  Simulated motor or industrial equipment powered via relay.

7. Miscellaneous

- SIM1 (Simulation Serial Interface)
  Used for simulation testing in Proteus.
- Wires & Connectors
  For interconnecting all components.
- Ground and VCC Rails

Working:

This project aims to monitor voltage and temperature of an industrial motor in real-time and automatically control it using a relay. It also enables remote monitoring via Wi-Fi (ESP8266) and local display using an LCD.

1. Power Supply and Sensor Input

- The AC voltage source is stepped down using a transformer (TR1).
- The stepped-down voltage is conditioned using a resistor divider (R1–R4) and a capacitor (C1) to provide a safe analog voltage to the Arduino UNO.
- LM35 Temperature Sensor (U1) gives an analog voltage output proportional to the surrounding temperature. It is connected to one of the Arduino's analog input pins.

2. Signal Processing (Arduino UNO)

- The Arduino UNO reads:
    - The scaled analog voltage from the voltage sensor circuit.
    - The temperature value from the LM35 sensor.
- It converts analog signals to digital values using its ADC and calculates:
    - Actual AC voltage.
    - Temperature in Celsius.

3. Decision Making

- The microcontroller compares the sensor values with predefined thresholds.
    - Example:
        - Voltage threshold: >250V
        - Temperature threshold: >35°C
- If either value exceeds the threshold:
    - The relay is turned OFF, cutting power to the motor.
    - LED (D1) glows to indicate a fault.

- LCD displays: "HIGH VOLTAGE" or "HIGH TEMP" warning.

- If values are within safe range:
  - Relay stays ON, and the motor continues operation.
  - LCD shows normal readings.

4. Display and Communication

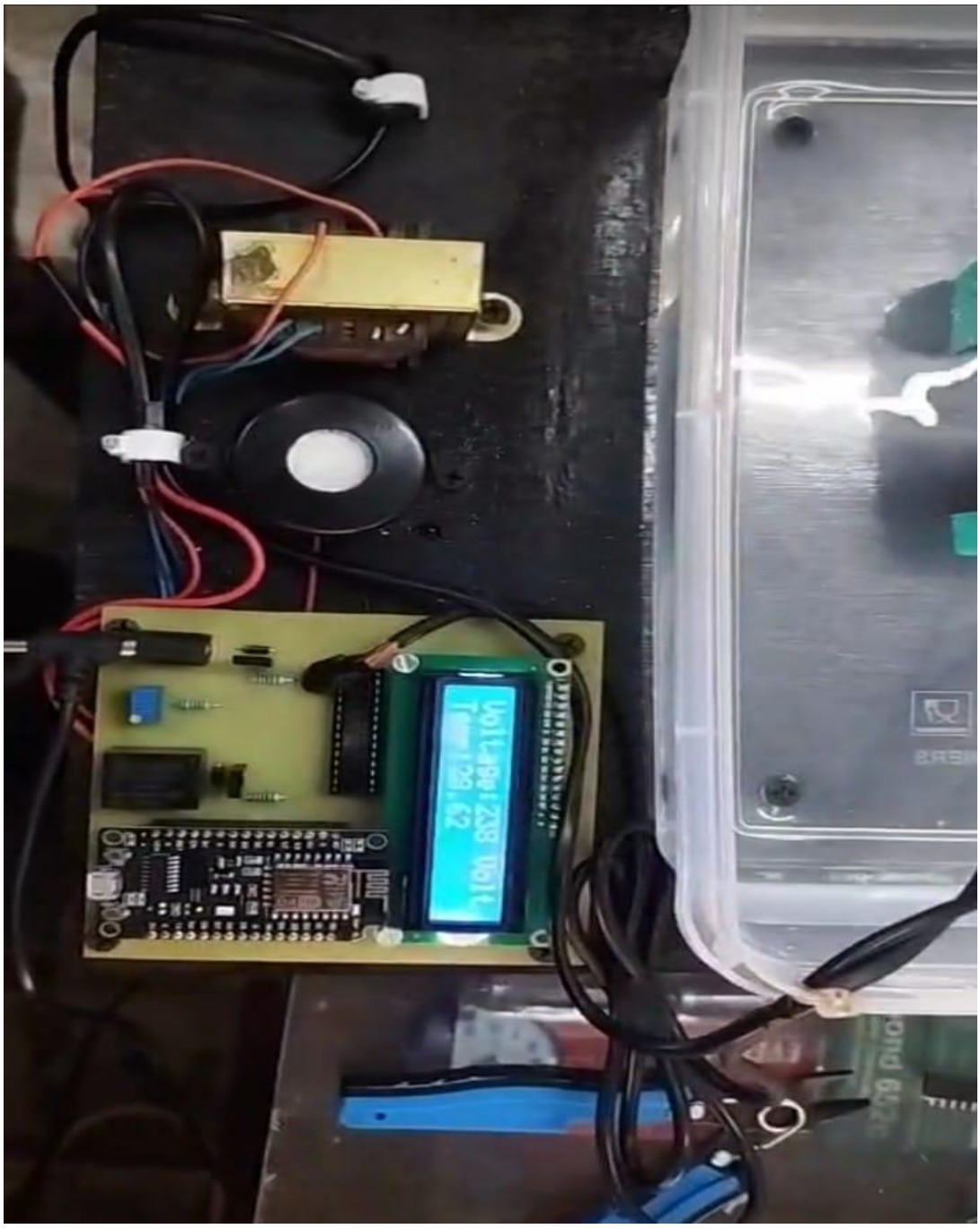
- The 16x2 LCD displays:
  - Real-time voltage (e.g., V: 269.9V)
  - Real-time temperature (e.g., T: 26°C)
  - System status message (e.g., HIGH VOLTAGE)
- The ESP8266 Wi-Fi Module (U2):
  - Sends voltage and temperature data to a cloud platform or mobile app.
  - Enables remote monitoring and alerts in real-time using Wi-Fi.

Key Functions

Function	Component Involved
Voltage Monitoring	Transformer + Voltage Divider + Arduino
Temperature Monitoring	LM35 + Arduino
Fault Detection	Arduino (logic comparison)
Motor Control	Relay
Local Display	16x2 LCD
Wireless Communication	ESP8266

5. HARDWARE & SOFTWARE IMPLEMENTATION

HARDWARE



SOFTWARE

Code For Arduino:

```
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Filters.h>
#define RS              PIN_PB0
#define EN              PIN_PD7
#define D4              PIN_PD6
#define D5              PIN_PD5
#define D6              PIN_PB7
#define D7              PIN_PB6
#define Relay           PIN_PC1
#define Phase_Voltage   PIN_PC0
#define ONE_WIRE_BUS    PIN_PB2
#define Buzzer          PIN_PB1
#define Relay           PIN_PC1

char buf[17];
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);
float testFrequency = 50;
float windowLength = 40.0 / testFrequency;
float intercept = -0.04;
float Slope_Voltage = 1.40;
unsigned long printPeriod          = 1000;
unsigned long previousMillis       = 0;
int Voltage_Analog_Sample          = 0;
```

```
unsigned int Voltage_Digital_Sample = 0;
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);


void setup()
{
  lcd.begin(16, 2);
  Serial.begin(9600);
  sensors.begin();
  pinMode(Buzzer, OUTPUT);
  pinMode(Relay,  OUTPUT);
}


void loop()
{
  RunningStatistics inputStats1;  //Easy life lines, actual calculation of the RMS requires a load of coding
  inputStats1.setWindowSecs(windowLength);
  while(1)
  {
    Voltage_Analog_Sample    = analogRead(Phase_Voltage);
    inputStats1.input(Voltage_Analog_Sample);
    if((unsigned long)(millis() - previousMillis) >= printPeriod)
    {
      previousMillis = millis();
      Voltage_Digital_Sample = intercept + Slope_Voltage * inputStats1.sigma();
      Voltage_Digital_Sample = map(Voltage_Digital_Sample, 310, 430, 230, 240);
```

```
sensors.requestTemperatures();
float temperatureC = sensors.getTempCByIndex(0);
if((Voltage_Digital_Sample >= 200) && (Voltage_Digital_Sample <= 260))
{
 lcd.setCursor(0, 0);
 sprintf(buf, "Voltage:%3d Volt", Voltage_Digital_Sample);
 lcd.print(buf);
 lcd.setCursor(0, 1);
 lcd.print("Temp:");
 lcd.print(temperatureC);
 if(temperatureC >= 35)
 {
  lcd.setCursor(0, 0);
  lcd.print("HIGH Tempature ");
  lcd.setCursor(0, 1);
  lcd.print("tor Off.....");
  digitalWrite(Buzzer, HIGH);
  digitalWrite(Relay,  LOW);
 }
 digitalWrite(Buzzer, LOW);
 digitalWrite(Relay,  HIGH);
}
else if(Voltage_Digital_Sample >= 260)
{
 lcd.setCursor(0, 0);
 lcd.print("HIGH Voltage Mo-");
 lcd.setCursor(0, 1);
```

```
        lcd.print("tor Off.....");
        digitalWrite(Buzzer, HIGH);
        digitalWrite(Relay,  LOW);
      }
      else if(Voltage_Digital_Sample <= 200)
      {
        lcd.setCursor(0, 0);
        lcd.print("LOW Voltage Mo- ");
        lcd.setCursor(0, 1);
        lcd.print("tor Off.....");
        digitalWrite(Buzzer, HIGH);
        digitalWrite(Relay,  LOW);
      }
      sendMultipleData(temperatureC, Voltage_Digital_Sample);
    }
  }
}

void sendMultipleData(float intValue1, int intValue2)
{
  Serial.print("<");
  Serial.print(intValue1);    //Value1
  Serial.print(",");
  Serial.print(intValue2);    //Value2
  Serial.print(",");
  Serial.println(">");
}
```

Code For ESP826601ESP01

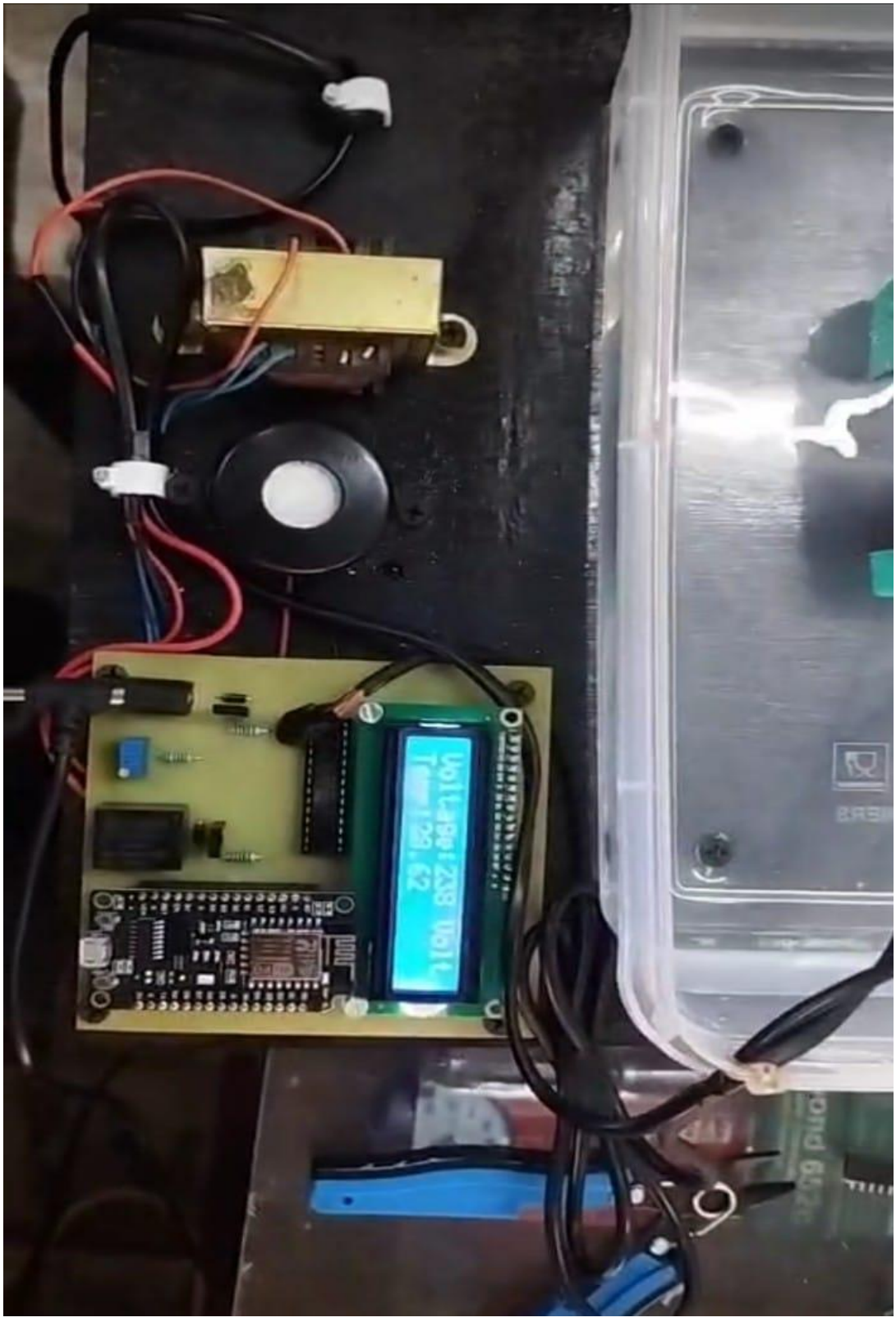
```
#define BLYNK_TEMPLATE_ID "TMPL3ZhYXgUc8"
#define BLYNK_TEMPLATE_NAME "Predictive Maintenance System"
#define BLYNK_AUTH_TOKEN
"gBvS_JuGYh64GQRh3utTTWrFFPLmZ2fP"
String receivedData = "";
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <BlynkSimpleEsp8266.h>
const int flashButtonPin = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(flashButtonPin, INPUT);
  pinMode(D8, OUTPUT);
  pinMode(flashButtonPin, INPUT);
  WiFiManager wifiMn;
  if (!wifiMn.autoConnect("Esp Web Portal"))
  {
    Serial.println("Failed to connect and hit timeout");
    ESP.reset();
    delay(1000);
```

```
  }
  Serial.println("Connected to Wi-Fi!");
  Blynk.config(BLYNK_AUTH_TOKEN);
  Blynk.connect();
}
void loop()
{
  Blynk.run();
  receiveData();
  if (digitalRead(flashButtonPin) == LOW)
  {
    WiFiManager wifiMn;
    wifiMn.resetSettings();
    ESP.reset();
  }
}

void receiveData()
{
  while (Serial.available() > 0)
  {
    char incomingChar = Serial.read();
    if (incomingChar == '<')
    {
```

```
      receivedData = "";
    }
    else if (incomingChar == '>')
    {
      parseData(receivedData);
    }
    else
    {
      receivedData += incomingChar;
    }
  }
}
void parseData(String data)
{
  float intvalue = data.substring(0, data.indexOf(',')).toFloat();
  data.remove(0, data.indexOf(',') + 1); // P Tank
  Blynk.virtualWrite(V0, intvalue);
  int intvalue1 = data.substring(0, data.indexOf(',')).toInt();
  data.remove(0, data.indexOf(',') + 1); // P Tank
  Blynk.virtualWrite(V1, intvalue1);
}
```

6.RESULT & EVALUATION



7.ADVANTAGES & APPLICATIONS

Advantages of the Basic Current and Voltage Monitoring System:

1. Real-time Monitoring
   Continuously tracks voltage and temperature to ensure timely detection of faults.

2. Remote Access
   With the ESP8266 module, data can be accessed via mobile or cloud platforms, enabling remote supervision.

3. Preventive Shutdown
   The relay-based motor control automatically turns off the system under abnormal conditions, preventing equipment damage.

4. Cost-Effective
   Utilizes low-cost sensors and microcontrollers, making it suitable for small to medium-scale industries.

5. Data Logging Possibility
   The system can be extended to store historical data for performance analysis and maintenance planning.

6. Improved Equipment Lifespan
   By avoiding over-voltage and overheating conditions, the lifespan of motors and pumps is significantly increased.

7. Scalable Design
   Easy to add more sensors (e.g., vibration, current, humidity) for advanced diagnostics.

Applications:

1. Industrial Motor and Pump Monitoring
   Used in factories to monitor motors in conveyor belts, pumps in fluid handling systems, and compressors.

2. Smart Manufacturing Plants
   Part of Industry 4.0 systems to automate fault detection and reduce manual supervision.

3. HVAC Systems
   Monitors and protects motors in heating, ventilation, and air conditioning equipment.

4. Water Treatment Plants
   Prevents damage to pumps and motors due to voltage surges or overheating in continuous operation environments.

5. Remote Utility Monitoring
   Used by utility companies to monitor motor-based equipment in substations or remote pumping stations.

6. Agricultural Irrigation Systems
   Ensures reliable operation of water pumps used for irrigation by detecting faults early.

8. REFERENCES

[1] "Predictive Maintenance of Industrial Machines using IoT and Machine Learning" – IEEE, 2020 https://circuitdigest.com/

[2] "An IIoT Framework for Predictive Maintenance of Rotating Machinery" – Elsevier, 2021 https://mqtt.org/documentation/

[3] "Condition Monitoring of Industrial Pumps Using Vibration Analysis and IoT" – Springer, 2019 https://www.arduino.cc/reference/en/

[4] "IoT Based Predictive Maintenance System for Industrial Motors.International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), 10(4), 1523-1530." https://wokwi.com/

APENDIX

Component Name	Quantity	Price per Unit (₹)
ESP32 Development Board	1	390
Temperature Sensor (LM35/DHT22)	1	90
16x2 LCD Display	1	350
Buzzer Module	1	50
ATmega328p	1	160
Voltage Sensor	1	90
Pump	1	320
Transformer	1	200
Total	8	1650