Jomo Kenyatta University of Agriculture and Technology

College of Engineering and Technology

School of Mechanical, Materials, and Manufacturing Engineering

Department of Marine Engineering and Maritime Operations

---

# Advance Condition Based Predictive Maintenance of Electric Motors in Ships

**MR-PRO-22-010**

## Final Year Project

**Kibichii Kieran (ENM241-0127/2017)**

**Supervisors**

Dr.Benson Oyunge

Dr.Gerald J. Odhiambo

*A project submitted in Partial Fulfillment of the Requirements for the degree of Bachelor of Science in Marine Engineering of The Jomo Kenyatta University of Agriculture and Technology*

---

# Declaration

I hereby declare that the work contained in this report is original; researched and documented by the undersigned student. It has not been used or presented elsewhere in any form for award of any academic qualification or otherwise. Any material obtained from other parties have been duly acknowledged. I have ensured that no violation of copyright or intellectual property rights have been committed.

1. Kibichii Kieran

   Signature.....Date.....

Approved by supervisors:

1. Dr. Benson Oyunge

   Signature.....Date.....

2. Dr. Gerald J. Odhiambo

   Signature.....Date.....

# Contents

# List of Figures

# Abstract

Maritime shipping is employed to move food, medicines, and far more at the heart of world trade. For growth and development, economical mechanisms of shipping are followed, particularly within the developing world. Machinery and equipment breakdown that occur during marine transportation cause delays, swings in supplies, production, and trigger infinite downstream effects on entire supply chains.

In the shipping industry, planned and reactive maintenance that is primarily practiced requires halting of vessel operations for a number of days despite the fact that no fatigue or machinery and equipment breakdown is observed. Motors are widely utilized in ships auxiliary machinery, the project focuses on creating failure predictions and determining remaining useful life(RUL) for motors aboard ships. Albeit quite fascinating is the role this research plays in the monitoring of equipment on autonomous ships. This is made possible with developments in information and technology. Massive amounts of data is collected and can facilitate condition based monitoring (CBM), conduct analysis on best performance and comprehensively diagnose ship engine room equipment.

Keywords: Efficiency, Condition monitoring, Remaining useful life, Autonomous ships, Performance.

# 1 Introduction

## 1.1 Background

### 1.1.1 Maintenance

As the world's industries push the boundaries of optimization and efficiency, the exponential increase in computational ability and technology, the automation of "higher-level" tasks that require human intellect is now possible. This headway brings unmanned autonomous vessels within the maritime industry closer to mass production. The practicality of autonomous vessels can only be achieved with constant awareness of the performance and operating state of machinery in the engine room. Observations of industry practices display that industry experience in reliability is heavily based on trial-and-error test procedures. Most of the reliability research in industry still focuses on two distinct periods of the product life. The warranty period, where most of the failures are due to product malfunctions or quality related problems, and, wear-out period, where the failures are due excessive wear and use [2].

Using sensors and logging software the condition of equipment is assessed as frequently as needed, enabling efficient analysis of data that facilitates planning of predictive maintenance on-board vessels. The electric motor is the most used device for conversion of energy from electric energy to mechanical energy and is used for electric propulsion, powering thrusters for station keeping, and different on-board equipment on hundreds of ships. Typically, 80-90 percent of the load installations will be electric motors[3]. Smart organizations know they can no longer afford to see maintenance as just an expense. Rather, maintenance must be integrated within the business cycle in order to guarantee predictability, growth and increase the overall quality of operations. Moving from a regime of scheduled rule-based maintenance via on-condition maintenance and ultimately to a data-driven risk-based regime can lead to more accurate and timely maintenance tasks. This smarter view of maintenance allows for achieving many practical advantages

leading to lower costs and increased safety and availability of ship systems.

Making failure predictions and determination of remaining useful life (RUL), realizes significant benefits not limited to: work style reforms, reduction in crew workload in that monitoring is done autonomously, improved safety from preventing accidents before they happen, and ensuring efficient optimal operation. In future more equipment will be added in a modular manner to realize better optimal performance.

With preventive and corrective maintenance still pronounced and used in the marine industry, mechanical systems such as plants, machinery and equipment components are being replaced or overhauled after some interval. Marine mechanical systems' parts could be replaced within the fixed maintenance or scheduled interval when they are still functional thus leading to unnecessary replacement or repair or maintenance costs. Likewise, the plants machinery and equipment system parts may have exhausted their operation lifetime before the maintenance interval reaches. This may lead to the breakdown of the mechanical systems, thus resulting in corrective maintenance. Hence, such traditional maintenance approaches are becoming less effective towards the reliability, safety and maintainability of marine mechanical system.

Detection of operation anomalies is the kind of predictive maintenance that can be carried out even when no data from previous failures in the equipment is available. When available, machine-learning models based on binary classification are used to predict failures in the near future in order to plan repairs or substitution of equipment[3].

There are several parameters to be considered:

- Temperature
- Pressure
- Vibration
- Current

In this study, the following parameters are investigated:

- Temperature
- Current

## 1.2 Problem statement

Out of 880 accidental errors in ship related incidents 62 percent are attributed to human failure; of this, 22 percent are shipboard related operations [4]. Engine room failures are caused by a majority of three ways, either by: natural mechanical failure, electrical failure of components and whole systems, human negligence, and poor competency in engine room procedures, inaccurate diagnosis, and sub-par prevention measures. These accidents are more notably recognized when they result in internationally felt effects such as oil spills damaging large swathes of marine ecosystems or when loss of life of crew members is realized. But with greater significance but less spoken of - the loss of millions in profits in maintenance and shipping costs incurred to the vessel's owner that would otherwise have been used more productively.

While it is impractical to try and eliminate accidents in the engine room, this design proposal seeks to provide a solution to improving efficiency and mitigating downtime by implementing strategies to reduce human error through the automation of engine room condition monitoring [5].

## 1.3 Objectives

### 1.3.1 Main Objective

To monitor the health of ship motors thus improving reliability and preventing downtime in ships.

### 1.3.2 Specific Objectives

1. To develop a predictive maintenance algorithm for electric motors in ships.

2. To be able to determine the remaining useful life of electric motors.

## 1.4 Justification of the study

Electric motors serve as a critical component for any facility. However, electric motors can be prone to a number of issues that lead to motor faults and failures. Failures disrupt business operations, decrease productivity, and adversely impact a company's bottom line i.e company's income after all expenses are deducted.

Motor inspection processes have shifted from manual scrutiny to semi-automated and fully-automated inspection. This will replace the time-consuming task of manual review, significantly increasing productivity while preventing missed inspection as well as errors. Traditionally, maintenance involves routine inspection and repair done manually. This cannot completely prevent the risk of machine downtime and will also result in the unnecessary early replacement of usable parts [6].

The purpose of this project is to alert about problems occurring in the motor and trying to mitigate the risk of unexpected failure. A well-planned predictive maintenance is the key to long life operation of motors. In ships, unexpected failure causes downtime which deeply eats into profits. The traditional approach is to repair and replace equipment after a period of time but this cannot prevent downtime due to malfunctions, which put a halt to operations and incur massive losses. Advanced monitoring is implemented on parts that are about to break down and can be discovered in advance to accurately determine the time for repair and risk of unexpected shutdown can be prevented. [6]

Although reactive and preventive maintenance will always have a part in operations, predictive maintenance is the next big step forward in the evolution of asset management. In

fact, the ability to connect assets and feed information into a central system gives organizations the power to turn data into powerful insights and automatically take corrective, preventive or predictive action.

# 2 Literature Review

## 2.1 Operation, Subsystems and Parameters

Every day we rely on a wide range of machines, but every machine eventually breaks down unless it's being maintained.

The control, monitoring and maintenance of ship equipment is fundamental to efficient engine room performance . Sensors and actuators play an important role in operation of various machines such as pumps, generators, among others. Therefore, they must always be in proper working condition. To guarantee this, machines are constantly monitored and types of maintenance of their components are performed. Corrective maintenance is performed in the case of a critical failure in the equipment and causes an unplanned downtime of the production line. Scheduled maintenance is performed periodically and equipment is checked and replaced, if necessary, in order to avoid unplanned downtime. Industry has begun to perform condition-based maintenance where predictive equipment status is used to plan a maintenance [7].

Condition monitoring is a type of maintenance inspection where an operational asset is monitored and the data obtained is analysed to detect signs of degradation, diagnose the cause of faults, and predict for how long it can be safely or economically run. There are five general categories of Condition Monitoring techniques—vibration monitoring and analysis; visual inspection and non-destructive testing; performance monitoring and analysis; analysis of wear particles in lubricants and of contaminants in process fluids; and electrical plant testing. Condition monitoring needs a good quality data such as that obtained by carefully run tests. However, much useful information can often be obtained from a plant's permanent instrumentation once repeatability is established[8]. The basic principle of condition monitoring is to indicate the occurrence of deterioration by taking physical measurements at regular intervals. Condition-based monitoring is used to inspect and replace the system according to the observed deterioration level.

Predictive maintenance is referred to as the use of data, machine learning techniques, and statistical algorithms to predict the most likely failure outcome of systems [**?**]. The machinery data collected by monitors say sensors (wired or wireless), is analysed in order to provide a consistent forecast on when a given component or machinery should be replaced or maintained thus optimizing on maintenance cost and downtime. Predictive maintenance is a branch of condition-based monitoring. Condition based monitoring (CBM) is described as a maintenance practice that tolerates and detects the failure of a system during operation through continuous monitoring of the system during operation[9].

Marine vehicle managers and marine technical experts claim that installation of more sensors and cables for monitoring purposes, will only have an impact on maintenance of marine systems only if the maintenance crew apply, hand-in-hand, other maintenance aspects like visual inspections, vibration measurements, regular oil analysis and enriching the maintenance information base with performance parameters [9].

Remaining Useful Life (RUL) is the time remaining for a component to perform its functional capabilities before failure. The concept of RUL is used to predict the life-span of components (of a service system) with the purpose of minimizing catastrophic failure events in both manufacturing and service sectors. This project involves acquiring real data from a normal working pump at its different stages in life. This will be done using multiple sensors attached to the pump at different times under different working conditions. Over time, the installed sensors will generate more and more data which can be used to improve the initial models and make near-perfect failure predictions.

Currently, the industry is majorly relying on sensors for condition monitoring which has facilitated decision making under time constraints. The time between the point where a potential failure occurs and the point where it deteriorates into a functional failure can be seen as an opportunity window during which decision-making algorithms can recommend actions with the aim to eliminate the anticipated functional failure or mitigate its effect. The system can record and monitor pressure and temperature conditions of an industrial motor and transmit the data through a wireless network to a data logging

centre. The current prototype was developed using open-source software and hardware and can successfully identify abnormal motor conditions from sensor input values that exceed predefined set-points.

The widespread use of sensors in industry allows for data acquisition, which, combined with advanced methods of analysis, can significantly improve and optimize production. Therefore, the data can be used not only to monitor the current state of the process and devices, but also to predict this failure state. The application of predictive methods to sensory data representing production process, including machine condition, allows for early identification and prediction of the faulty or hazardous process state or machine break down.

The aim of this project is to present a data-driven method analysing sensory data to predict machine failure or identify its deteriorating condition, so that users can plan maintenance work and avoid unplanned downtime. This method was prepared based on real-life data with the assumption that the created solution will be used in industry as a result of the implemented project.

Predictive maintenance is based on the prognostic and health management technology, which supposes that the remaining useful life of equipment can be predicted. However, due to uncertainty of prognostics there could be wrong decisions regarding the remaining time to failure.

Currently, the most promising strategy of maintenance for various technical systems and production lines is the predictive maintenance (PM), which can be applied to any system if there is a deteriorating physical parameter like vibration, pressure, voltage, or current that can be measured[6].

Motors are the backbone of industry; they start degrading due to different reasons such as long period of operation, variations of power supply, or harsh environment; which gradually lead to permanent damage.Thus, the monitoring of their condition is of prime importance for sustaining the operation and maintainig efficiency. Consequently, it be-

comes crucial to monitor the operation continuously [3].

Industry reliability surveys suggest that AC motor failures may be divided into five categories, including (IEEE, 1997):

- bearing: 44%
- stator winding: 26%
- rotor: 3%
- shaft: 5%
- others: 22%.

Even though the replacement of defective bearings is the cheapest to fix among the causes of failure, it is the most difficult one to detect. Motors that are in continuous use cannot be stopped for analysis.

## 2.2 Sensors

### 2.2.1 Temperature Sensor

A temperature sensor is a device, typically, a thermocouple or resistance temperature detector, that provides temperature measurement in a readable form through an electrical signal. It measures the degree of hotness or coolness of an object. There are many types of temperature sensors, but, the most common way to categorise them is based upon the mode of connection which includes, contact and non-contact temperature sensors[10].

Contact sensors include thermocouples and thermistors because they are in direct contact with the object they are to measure. Whereas, the non-contact temperature sensors measure the thermal radiation released by the heat source. Such temperature meters are often used in hazardous environments like nuclear power plants or thermal power plants.

In this project the contact type of temperature sensor was selected due to the nature of data to be collected from the bearings.

### 2.2.2 Thermistor

A thermistor is a temperature sensitive resistor. They are often used as a temperature sensor. The term thermistor is a contraction of the words "thermal" and "resistor". All resistors have some dependency on temperature, which is described by their temperature coefficient. In most cases, the temperature coefficient is minimized, but in the case of thermistors a high temperature coefficient is achieved. Unlike most other resistors, thermistors usually have negative temperature coefficients (NTC) which means the resistance decreases as the temperature increase. Thermal resistors with a positive temperature coefficient are called PTC thermistors (Positive Temperature Coefficient)[11].

The DS18B20 is a small temperature sensor with a built in 12bit ADC. It can be easily connected to an Arduino digital input. The sensor communicates over a one-wire bus and requires little in the way of additional components [11].

## 2.3 Current Sensor

A current sensor is a device that detects and converts current to an easily measured output voltage, which is proportional to the current through the measured path.

When a current flows through a wire or in a circuit, voltage drop occurs. Also, a magnetic field is generated surrounding the current carrying conductor. Both of these phenomena are made use of in the design of current sensors. Thus, there are two types of current sensing: direct and indirect. Direct sensing is based on Ohm's law, while indirect sensing is based on Faraday's and Ampere's law.

Direct Sensing involves measuring the voltage drop associated with the current passing through passive electrical components. Indirect Sensing involves measurement of the

magnetic field surrounding a conductor through which current passes.

### 2.3.1 Pressure Transducer

Pressure transducers can come in a number of shapes and sizes, but the majority of transducers have a cylinder-shaped centre which houses the diaphragm and the measurement pressure chamber, a pressure port at one end which is typically a threaded, bolted, barbed fitted, or open, and on the other end a location for signal transmission[13].

# 3 Methodology

## 3.1 Data Collection

Temperature and current data was collected from a motor in a series of data points indexed in time order with temperature and current measurements taken at regular intervals.

### 3.1.1 Temperature sensor

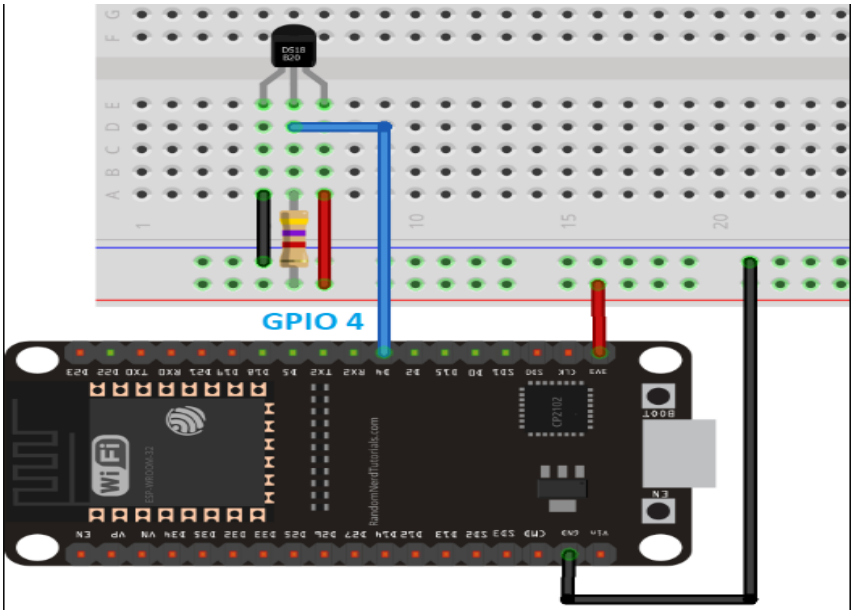


Figure 3.1: ds18b20 pin out

To initiate a temperature measurement and conversion, a read and convert command is issued. Following conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory, to be able to issue this command an external power supply is used to specify read time slots”

```
void loop() {
   sensors.requestTemperatures();
   float temperatureC = sensors.getTempCByIndex(0);
   float temperatureF = sensors.getTempFByIndex(0);
   Serial.print(temperatureC);
   Serial.print(temperatureF);
   Serial.println("F");
   delay(5000);
}
```

### 3.1.2 Current sensor

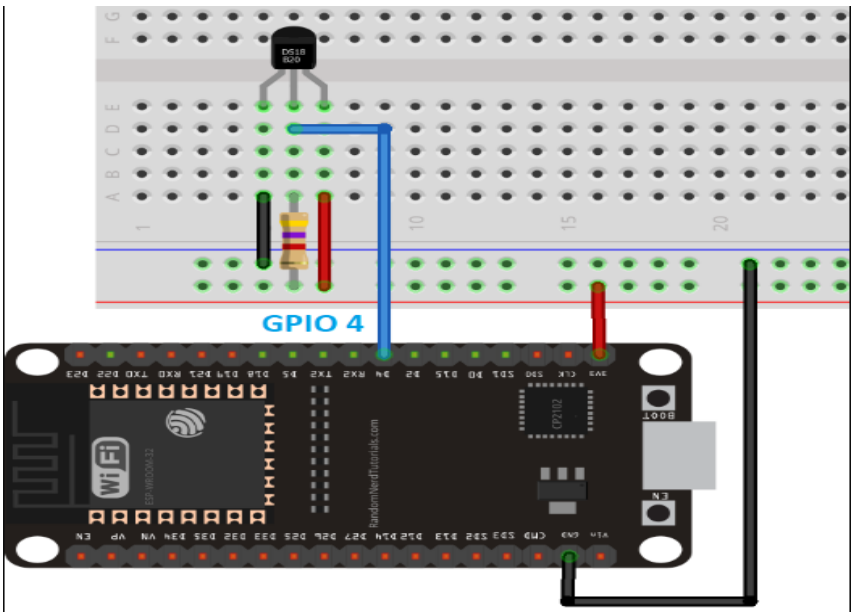


Figure 3.2: current sensor pin out

Current flows through the onboard hall sensor circuit in its IC The hall effect sensor detects the incoming current through its magnetic field generation Once detected, the hall effect sensor generates a voltage proportional to its magnetic field that's then used

to measure the amount of current

```
// The on-board ADC is 10-bits
// Different power supply will lead to different reference sources
// example: 2\^{}10 = 1024 -> 5V / 1024 \~{}= 4.88mV
//         unitValue= 5.0 / 1024.0*1000 ;
float unitValue= RefVal / 1024.0*1000 ;
float voltage = unitValue * sensorValue;

//When no load,Vref=initialValue
SERIAL.print(``initialValue: '');
SERIAL.print(voltage);
SERIAL.println(``mV'');

// Calculate the corresponding current
float current = (voltage - Vref) * sensitivity;

// Print display voltage (mV)
// This voltage is the pin voltage corresponding to the current
/*
voltage = unitValue * sensorValue-Vref;
SERIAL.print(voltage);
```

### 3.1.3 Omnimonitor

The device is responsible for keeping track of and managing data from different sources as well as managing read and write signals from the sensor to the remotely accessible server Both ds18b20 and DHT22 provide the output of temperature and humidity in the complex digital output format which can not be directly read with GPIO pins without writing any technique which can read these output signals. These sensors provide data

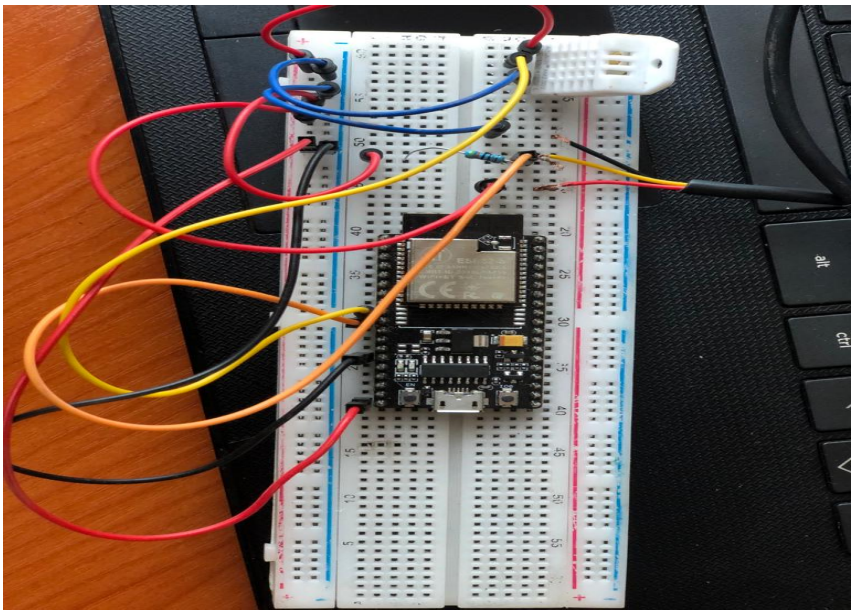

Figure 3.3: combined sensor setup

through a single wire two-way communication protocol. A single process communication consists of 40 bits.To get values of temperature and humidity we call a number of created functions.

```
\#if defined(ESP32)
\#include <WiFiMulti.h>
WiFiMulti wifiMulti;
\#define DEVICE ``ESP32''
\#elif defined(ESP8266)
\#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
\#define DEVICE ``ESP8266''
\#endif


//\#define DHTTYPE DHT11 // DHT 11
//\#define DHTTYPE DHT21 // DHT 21 (AM2301)
```

```
\#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

// ds18b20 sensor
\#define ONE\_WIRE\_BUS 26
OneWire oneWire(ONE\_WIRE\_BUS);
DallasTemperature sensors(\&oneWire);

uint8\_t DHTPin = 25;
DHT dht(DHTPin, DHTTYPE);

float temperature\_Celsius; // ambient
float humidity;
float temperature\_Celcius2; // contact
```

### 3.1.4 MQTT

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol designed for usage in situations where clients require a minimal code footprint and are linked to unreliable networks or networks with restricted capacity. It's mostly utilised for M2M (machine-to-machine) communication and Internet of Things connectivity.

MQTT uses a PUSH/SUBSCRIBE architecture to run on top of TCP/IP. There are two sorts of systems in MQTT architecture: clients and brokers. The server with which the clients communicate is known as a broker. Client messages are received by the broker, who then forwards them to other clients. Clients connect to the broker rather than communicating directly with one another.

MQTT also reduces transmissions by using a well-defined, compact message structure. In comparison to HTTP, each message has a fixed header of only 2 bytes.

### 3.1.5 OTA

Instead of needing the user to connect the ESP32 to a computer via USB to execute the update, OTA programming allows update of new data to the ESP32 via Wi-Fi.

When there is no physical access to the ESP module, the OTA capability comes in handy. It helps to cut down on the time spent on each ESP module during maintenance.

One of the most useful features of OTA is that it allows a single central location to deliver an update to many equipment on the same network ideally all monitored equipment in a ship engine room.

### 3.1.6 Declaration, definition and initialization

Define, declare variables and initialize classes.

```
#define CURRENT_VERSION "1.0.0" // The current version of the firmware
    running
#define DOWNLOAD_URL SERVER_URL // The server url where we will get the
    latest firmware version
#define DHTPIN 2 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define LED 4 // Digital pin connected to the inbuilt LED
WiFiClient espClient; //Initializing the WiFiClient
void setup_wifi();
void reconnect();
PubSubClient client(espClient); // Initializing the PubSubClient
WebServer server(80); // Initializing the WebServer
// functions to take care of OTA firmware update
void handleRoot();
String getDownloadUrl();
bool downloadUpdate(String);
```

```
DHT_Unified dht(DHTPIN, DHTTYPE);
int ledState = LOW; // Set the ledstate to be off at the first instance
const long interval = 1000; // This is the interval we will be using to
    check for a new version
unsigned long previousMillis = 0; // previous timings in milliseconds
unsigned long currentMillis; // current timings in milliseconds
bool success; // download success
```

### 3.1.7 Setup function

This is executed only once in the beginning of the program.

```
void setup() \{
Serial.begin(115200); // begin Serial at 115200 baud rate
setup\_wifi(); // function to setup up wifi
espClient.setServer(mqtt\_server, 1883); //Setup the MQTT server

// Initialize device.
dht.begin();
// Print temperature sensor details.
sensor\_t sensor;
dht.temperature().getSensor( \& sensor);
Serial.println(F(‘));
Serial.println(F(‘‘Temperature Sensor Present’’));

// Print humidity sensor details.dht.humidity().getSensor(\&sensor);
Serial.println(F(‘‘ ));
Serial.println(F(‘‘Humidity Sensor Present’’));
Serial.setDebugOutput(true); // Set debug to true in order to print more
    serial output
```

```
pinMode(LED, OUTPUT); // Initialize the inbuilt led as an output device
delay(3000); // Set a delay of 3 seconds (optional)
String version = String(``<p>Current Version- v'') +
    String(CURRENT\_VERSION) + String(``</p>'');
Serial.println(version);
// Setup Wifi Manager
WiFiManager wm;
WiFiManagerParameter versionText(version.c\_str());
wm.addParameter( \& versionText);
if (!wm.autoConnect()) \{
Serial.println(``failed to connect and hit timeout'');
ESP.restart();
delay(1000);
\}
// Check if we need to download a new version
String downloadUrl = getDownloadUrl();
if (downloadUrl.length() > 0) \{
success = downloadUpdate(downloadUrl);
if (!success) \{
Serial.println(``Error updating device'');
\}
\}
server.on(``/'', handleRoot);
server.begin();
Serial.println(``HTTP server started'');
Serial.print(``IP address: '');
Serial.println(WiFi.localIP());
\}
```

### 3.1.8 Loop function

This section is repeated after a specified interval to read and write parameters specified

```
// Get temperature event and print its value.
sensors_event_t event;
dht.temperature().getEvent( & event);
if (isnan(event.temperature)) {
   Serial.println(F("Error reading temperature!"));
} else {
   Serial.print(F("Temperature: "));
   Serial.print(event.temperature);
   Serial.println(F("C"));
}
// Get humidity event and print its value.
dht.humidity().getEvent( & event);
if (isnan(event.relative_humidity)) {
   Serial.println(F("Error reading humidity!"));
} else {
   Serial.print(F("Humidity: "));
   Serial.print(event.relative_humidity);
   Serial.println(F("%"));
}
currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
   previousMillis = currentMillis;
   ledState = ledState == LOW ? HIGH : LOW;
   digitalWrite(4, ledState);
}
// Just chill
server.handleClient();
```

```
   delay(1000);
   char msg[200];
   if (!espClient.connected()) {
      reconnect();
   }
   StaticJsonBuffer < 300 > JSONbuffer;
   JsonObject & JSONencoder = JSONbuffer.createObject();
   JSONencoder["time"] = millis();
   JSONencoder["temperature"] = event.temperature;
   JSONencoder["humidity"] = event.relative_humidity;
   char JSONmessageBuffer[100];
   JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
   Serial.println(JSONmessageBuffer);
   if (espClient.publish("dht/user_id892", JSONmessageBuffer) == true) {
      Serial.println("Success sending message");
   } else {
      Serial.println("Error sending message");
   }
   espClient.loop();
}
```

### 3.1.9 Handle server requests

Ensures data is read only when it can be written to the server.

```
   void handleRoot() {
      server.send(200, "text/plain", "v" + String(CURRENT_VERSION));
   }
```

### 3.1.10 Download links

```
String getDownloadUrl() {
   HTTPClient http;
   String downloadUrl;
   Serial.print("[HTTP] begin\n");
   String url = DOWNLOAD_URL;
   http.begin(url);
   Serial.print("[HTTP] GET\n");
   // start connection and send HTTP header
   int httpCode = http.GET();
   // httpCode will be negative on error
   if (httpCode > 0) {
      // HTTP header has been send and Server response header has been
          handled
      Serial.printf("[HTTP] GET code: %d\n", httpCode);
      // file found at server
      if (httpCode == HTTP_CODE_OK) {
         String payload = http.getString();
         Serial.println(payload);
         downloadUrl = payload;
      } else {
         Serial.println("Device is up to date!");
      }
   } else {
      Serial.printf("[HTTP] GET failed, error: %s\n",
          http.errorToString(httpCode).c_str());
   }
   http.end();
   Serial.println(downloadUrl);
   return downloa
```

### 3.1.11 Download binary firmware

This is used to

```
/*
Download binary image and use Update library to update the device.
*/
/*
Download binary image and use Update library to update the device.
*/
bool downloadUpdate(String url) \{
HTTPClient http;
Serial.print(‘‘[HTTP] Download begin\textbackslash{}n’’);
http.begin(url);
Serial.print(‘‘[HTTP] GET\textbackslash{}n’’);
// start connection and send HTTP header
int httpCode = http.GET();
if (httpCode > 0) \{
// HTTP header has been send and Server response header has been handled
Serial.printf(‘‘[HTTP] GET code: \%d\textbackslash{}n’’, httpCode);
// file found at server
if (httpCode == HTTP\_CODE\_OK) \{
int contentLength = http.getSize();
Serial.println(‘‘contentLength : ’’ + String(contentLength));
if (contentLength > 0) \{
bool canBegin = Update.begin(contentLength);
if (canBegin) \{
WiFiClient stream = http.getStream();
Serial.println(‘‘Begin OTA. This may take 25 mins to complete. Things might
```

```
    be quite for a while.. Patience!'');
size\_t written = Update.writeStream(stream);
if (written == contentLength) \{
Serial.println(``Written : '' + String(written) + `` successfully'');
\} else \{
Serial.println(``Written only : '' + String(written) + ``/'' +
    String(contentLength) + ``. Retry?'');
\}
if (Update.end()) \{
Serial.println(``OTA done!'');
if (Update.isFinished()) \{
Serial.println(``Update successfully completed. Rebooting.'');
ESP.restart();
return true;
\} else \{
Serial.println(``Update not finished? Something went wrong!'');
return false;
\}
\} else \{
Serial.println(``Error Occurred. Error \#: '' + String(Update.getError()));
return false;
\}
\} else \{
Serial.println(``Not enough space to begin OTA'');
client.flush();
return false;
\}
\} else \{
Serial.println(``There was no content in the response'');
client.flush();
```

```
return false;
\}
\} else \{
return false;
\}
\} else \{
return false;
\}
\}
```

### 3.1.12 Setup wifi function

```
void setup_wifi() {
   // Connecting to a WiFi network
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }
   Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
}
```

### 3.1.13 Reconnect function

```
void reconnect() {
   // Loop until we're reconnected
   Serial.println("In reconnect");
```

```
    while (!espClient.connected()) {
      Serial.print("Attempting MQTT connection");
      // Attempt to connect
      if (espClient.connect("SES_DHT", MQTT_USER, MQTT_PASS)) {
        Serial.println("connected");
      } else {
        Serial.print("failed, rc=");
        Serial.print(espClient.state());
        Serial.println(" try again in 5 seconds");
        delay(5000);
      }
    }
}
```

## 3.2 Data Analytics Process Steps

The motor sensors output digital and analogue electrical signals. This is parsed to literal temperature and current values i.e degrees and amperes respectively. This is automated by use of a C bash script that then data is written into a CSV file that is now the accessible database for the algorithm fetch requests.

The data is prepared for analysis by wrangling to remove unwanted and redundant values. The stage is now set to visualize the data to be able to observe meaningful patterns that can provide useful insights to the motors condition. Ultimately this will also provide the logical descriptive equations on which predictive algorithms will learn from.

To prove meaningful results tests were developed to check if the output is in line with the expected results

## 3.3 Data Analysis

The data collected from the sensors will be analyzed using custom software created using vim. Graphs will be generated to compare the performance of each component.

```
DallasTemperature::request_t DallasTemperature::requestTemperatures() {
  DallasTemperature::request_t req = {};
  req.result = true;

  _wire->reset();
  _wire->skip();
  _wire->write(STARTCONVO, parasite);

  // ASYNC mode?
  req.timestamp = millis();
  if (!waitForConversion)
  return req;
  blockTillConversionComplete(bitResolution, req.timestamp);
  return req;
}
```

### 3.3.1 Descriptive analysis

Using exploratory data analysis prior and current running status is determined

### 3.3.2 Predictive analysis

Achieved by building predictive models. The required replacement parts as well as time required for maintenance can be determined.

### 3.3.3 Prescriptive analysis

This will provide insights on how to create desired outcomes i.e reduce load by 5% for increased service life

## 3.4 System Modelling

The predictive maintenance algorithm for motors system will be obtained from governing equations from which a transfer function will be generated from the linearized model. The transfer function will be used to generate a state space model for the system. The observed sources of faults and their relative frequency. Such sources can be the core components of the machine or its various sensors (such accelerometers and flow meters). The process measurements through sensors. The number, type and location of sensors, and their reliability and redundancies all will create both algorithm and comparative model. The sources of faults will translate to observed symptoms. Such cause-effect analysis will require extensive processing of data from the available sensors. Physical knowledge about the system dynamics will result in mathematical modeling of the system and its faults and from the insights of data. Understanding system dynamics will involve detailed knowledge of relationships among various signals from the machinery (such as input-output relationships among the actuators and sensors), the machine operating range, and the nature of the measurements (for example, periodic, constant or stochastic). The ultimate maintenance goal, such as fault recovery or development of a maintenance schedule

## 3.5 Sensors

Sensors will be used to collect data from the system as it runs. These include:

1. Humidity sensor
2. Temperature sensor

3. Flow rate sensor
4. Pressure sensors
5. Voltage sensor
6. Current sensor

   These sensors will be used by the controller to observe system performance and optimize for each parameter.

## 3.6 Process chart

From the generated models on, simulations will be performed using the different controllers and the responses and other metrics will be plotted out for further analysis using NumPy and Pandas python libraries . Values such as rise time, settling time and stochastic response will be observed to determine the system performance.

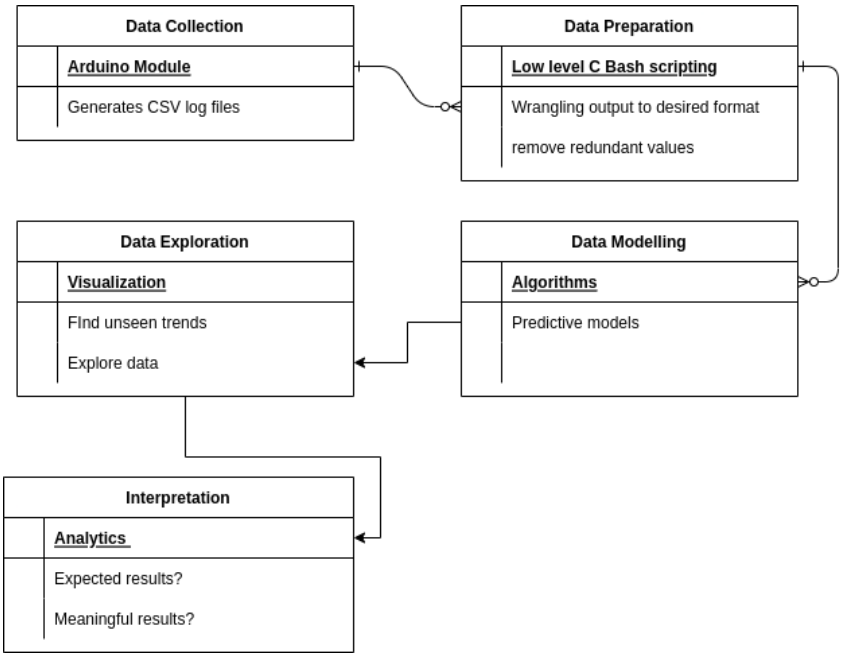


Figure 3.4: Data Analytics

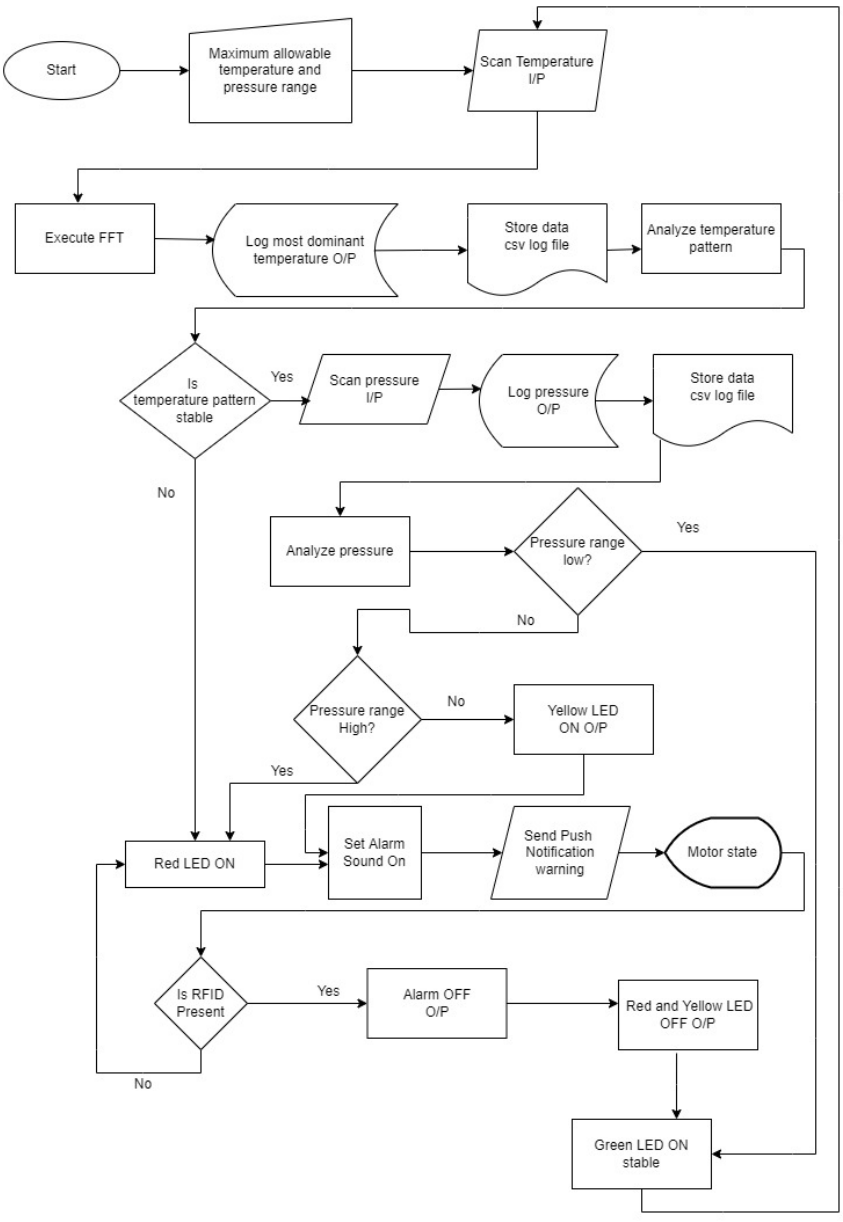


Figure 3.5: Module Process

# 4 Results and Discussions

1. A functional motor health monitoring device will be developed and tested.
2. The predictive maintenance algorithm will be formulated and proved from a selection of diverse methods.
3. The module supporting circuitry will be designed. board.
4. Electric motor system performance and efficiency will be optimized using insights from real-time data collected.

# 5 Project Budget

| **Item List** | | | |
|---|---|---|---|
| ITEM | SPECIFICATION | QUANTITY | PRICE |
| Arduino | Uno Rev3, usb 2.0 Cable Type A/B | 1 | 3000 |
| Thermocouple Amplifier | MAX31855K | 1 | 1600 |
| WiFi Development Board | NodeMCU 32S ESP32/ CH340c | 1 | 1400 |
| Thermocouple K type | Temp range: 0-600 | 1 | 1760 |
| Thermocouple DH22 | 3.3-6v input | 1 | 750 |
| Power Unit | 12v/5W SMPS Module | 1 | 600 |
| Breadboard | MB102 165*55*50mm | 2 | 400 |
| Jumper Wires | 20cm male 20cm female | 80 | 400 |
| **Total** | | | 9910 |

# 6 Work Plan

| WEEK | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Project Refinement | | | | | | | | | | | |
| System Modelling | | | | | | | | | | | |
| Controller Modelling | | | | | | | | | | | |
| Circuit Design | | | | | | | | | | | |
| Preparing Interim Report | | | | | | | | | | | |
| Presentation | | | | | | | | | | | |

Figure 6.1: Workplan table

# References

[1] R. White. Subject guides: Referencing, citing, and structuring bibliographies: Using zotero with LaTeX. [Online]. Available: https://libguides.rhul.ac.uk/referencing/Zoterolatex

[2] M. U. Thomas and S. S. Rao, "Warranty Economic Decision Models: A Summary and Some Suggested Directions for Future Research," *Operations Research*, vol. 47, no. 6, pp. 807–820, Dec. 1999. [Online]. Available: http://pubsonline.informs.org/doi/10.1287/opre.47.6.807

[3] J.-H. Han, D.-J. Choi, S.-K. Hong, and H.-S. Kim, *Motor Fault Diagnosis Using CNN Based Deep Learning Algorithm Considering Motor Rotating Speed*, Apr. 2019, pages: 445.

[4] K. Bratić, I. Pavić, S. Vukša, and L. Stazić, "Review of autonomous and remotely controlled ships in maritime sector," vol. 8, pp. 253–265.

[5] "13 common causes of motor failure." [Online]. Available: https://www.fluke.com/en-gb/learn/blog/motors-drives-pumps-compressors/13-common-causes-of-motor-failure

[6] G. Sampaio, A. Vallim Filho, L. Silva, and L. Silva, "Prediction of Motor Failure Time Using An Artificial Neural Network," *Sensors*, vol. 19, p. 4342, Oct. 2019.

[7] "Predictive Maintenance - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/engineering/predictive-maintenance

[8] S. Lu, P. Zhou, X. Wang, Y. Liu, F. Liu, and J. Zhao, "Condition monitoring and fault diagnosis of motor bearings using undersampled vibration signals from a wireless sensor network," *Journal of Sound and Vibration*, vol. 414, pp. 81–96, Feb. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022460X17307691

[9] D. Kimera and N. Fillemon Nduvu, "Predictive maintenance for ballast pumps on ship repair yards via machine learning," *Transportation Engineering*, vol. 2, p. 100020, Dec. 2020.

[10] "Temperature Sensors: Types, How It Works, & Applications," Jul. 2019. [Online]. Available: https://www.encardio.com/blog/temperature-sensor-probe-types-how-it-works-applications/

[11] Learn | OpenEnergyMonitor. [Online]. Available: https://learn.openenergymonitor.org/electricity-monitoring/temperature/DS18B20-temperature-sensing?redirected=true

[12] "Types of Pressure Sensor - A Guide." [Online]. Available: https://www.thomasnet.com/articles/instruments-controls/pressure-sbratic$_r eview_2 019, ensors/$

[13] "Different Types of Pressure Transducers." [Online]. Available: https://www.omega.com/en-us/resources/pressure-transducers-types