Knitting Machine Monitoring system

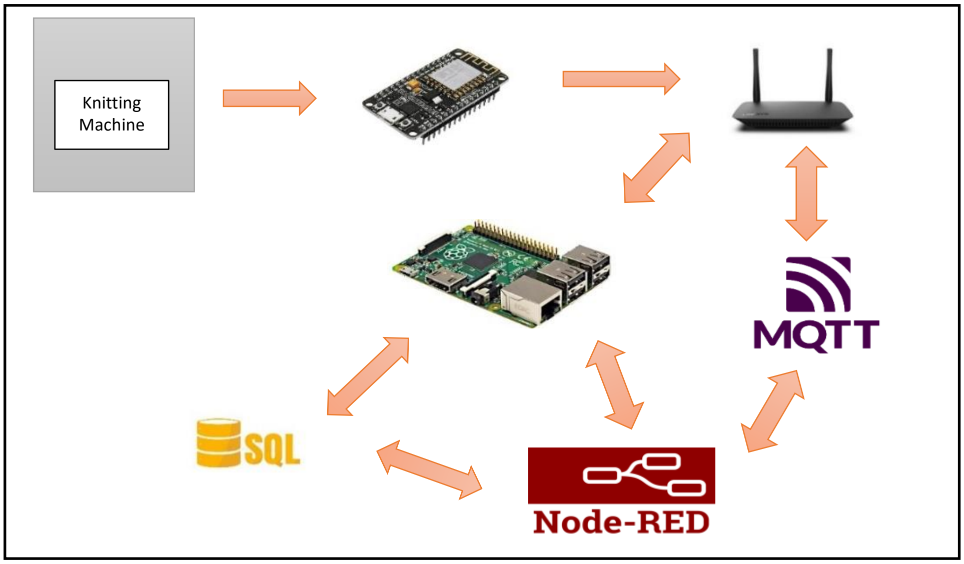
Abstract

Production monitoring is an important task for proper production planning. For this task the information that a system of this nature can possibly gather assumes significant importance. Among all the information available during production, the detection of faults assumes a crucial role since it directly affects the quality and productivity. This report presents a system which was developed with the purpose of performing the analysis in real time of the knitting process, supplying the parameters of major concern for production.

Introduction

Design a system to get knitting machine real time condition and store those data on database. This system has LIVE dashboard where we can check the machine running condition. The system can instantly detect machine conditions and immediately trigger sound alarm to alert technicians. The manual breakdown report system is to be automated and viewed by this system. We used mobile app to collect breakdown.

Conceptual Design Description.

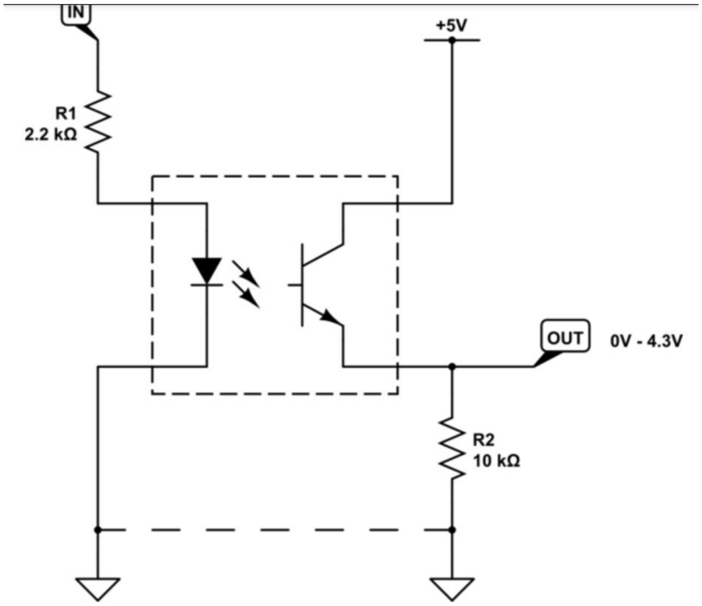


We have attached the module to get the data from the knitting machine. That data will be sent to the MQTT broker thoughts wifi network. Then node red got that data and process. Node red flow run on raspberry pi.

- **Collect data from the Knitting Machine**

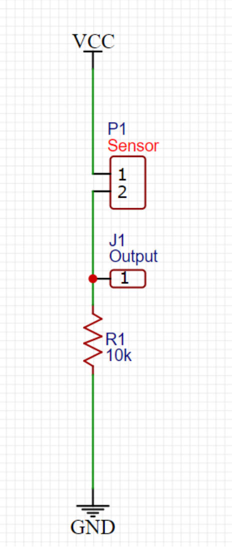
The position of the machine is determined by the indicator light on the machine. The voltage of the Light was 24V and had to be reduce to 3.3V to input the NodeMCU. Potential divider and Zener clamp circuit are the simplest way to obtain the required signal levels of 3.3V. But The safest way (for digital signals only) is with an optocoupler

The optocouplers allow you to completely electrically isolate your own circuit from the industrial controller, which is why I included the dotted connection between the grounds of your controller and your own circuitry. It will work either way, but by omitting that connection you get the ultimate protection from an errant controller.



- **Get Pressure from the knitting machine.**

We used oil pressure sensor to collect pressure of the machine. This pressure sensor work like a switch. When pressure increased desired(6Bar) value, The sensor will be short circuited.

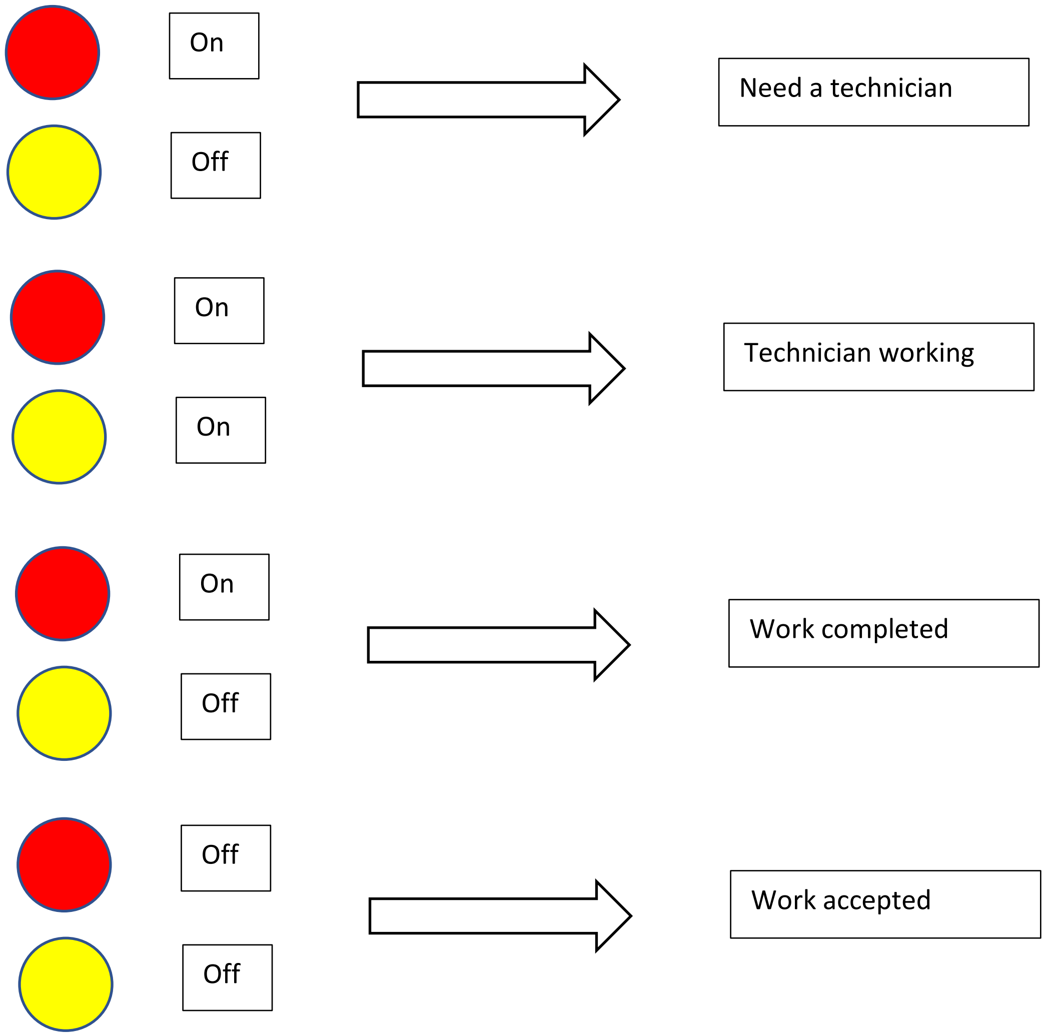


Oil pressure sensor

Circuit digram

- **Breakdown report switch**

There are two switch in knitting machine. Those switches we used to get breakdown.



🔴 On
🟡 Off
→ Need a technician

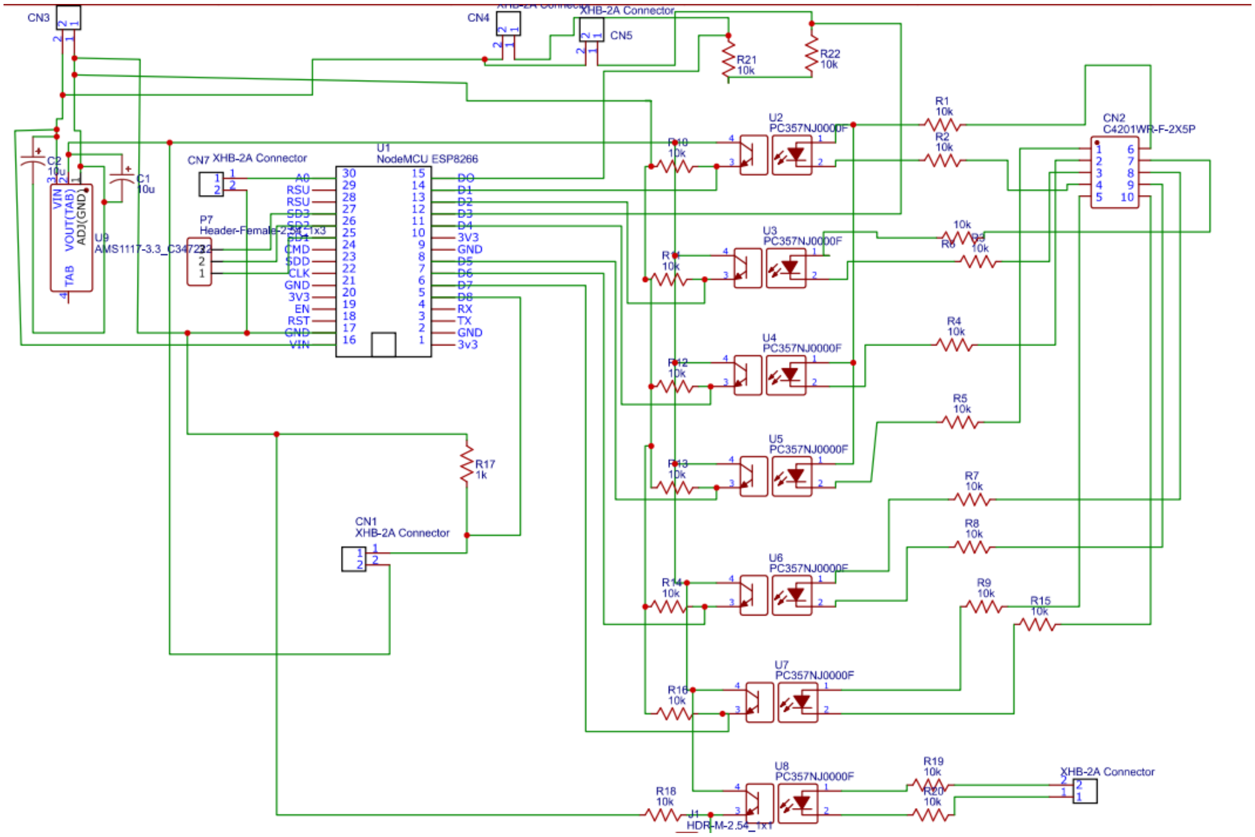
🔴 On
🟡 On
→ Technician working

🔴 On
🟡 Off
→ Work completed

🔴 Off
🟡 Off
→ Work accepted

- **Main Circuit**

There are 6 Indicator lights on knitting machine. We have to design circuit to input this signal to Node-MCU.



Circuit Diagram

CN1-

CN2- Indicator Light Voltage Input scout.

CN3- Input 5V

CN4- Switch 1

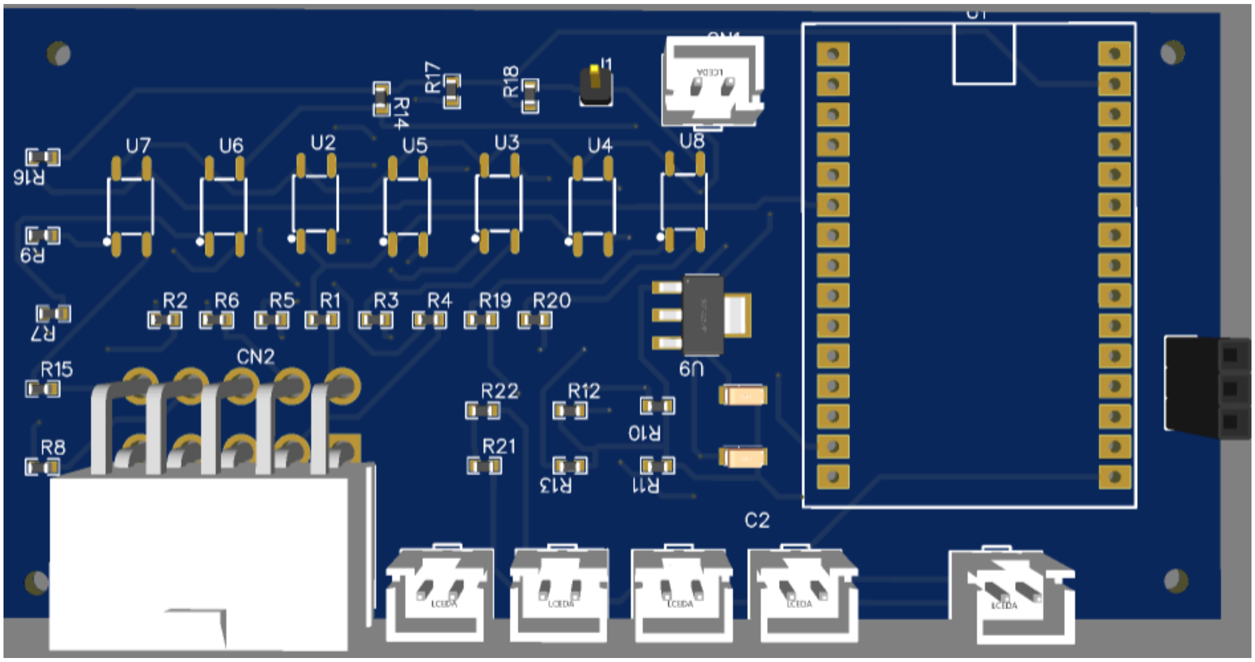
CN5- Switch 2

CN6- Extra Input(24V)

CN7- Pressure sensor Input

P7- Extra Input(3.3V)

The circuit is operated by 5V. Knitting machine have 24V outlet. This voltage should be reduced to 5V using a buck converter.



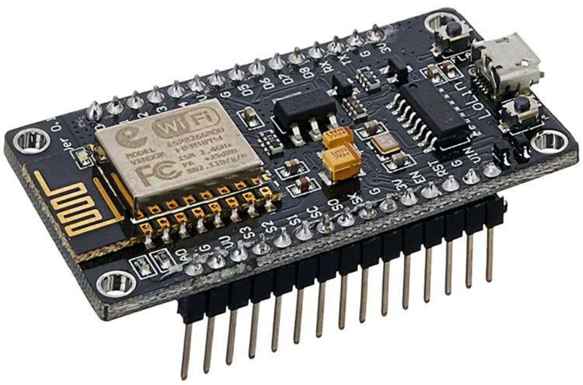
Main PCB

After that The NodeMCU inputs these digital levels and publish it to MQTT broker when the State of machine changes. The NodeMCU is Programed by Arduino.

Node MCU

Today, IOT applications are on the rise, and connecting objects are getting more and more important. There are several ways to connect objects such as Wi-Fi protocol.

NodeMCU is an open source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, and etc



- **NodeMCU programing code**

- **Add library to arduino**

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

- **Add WiFi user name and password**

```
const char* ssid = "Dialog 4G 086";
const char* password = "9e36e3d1";
```

- **MQTT broker and published topic**

```
const char* mqtt_server = "test.mosquitto.org";
const char* outTopic = "M1883";
```

- **Connect to the WiFi**

```
WiFiClient espClient;
PubSubClient client(espClient);
int value = 0;

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the
voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage
HIGH
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(outTopic, "hello world");
      // ... and resubscribe
      client.subscribe(inTopic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

- **White Blinking Light recognize**

```
  delay(1000);
  new2=digitalRead(pin7);
  delay(1000);
  new3=digitalRead(pin7);
```

- **Top Blinking Light recognize**

```
  new4=digitalRead(pin9);
  delay(300);
  new5=digitalRead(pin9);
```

- **logic for knitting machine running condition**

```
if((ab=="10")&&(bc=="00")){
    logic="1";
  }
  else if((ab=="11")&&(bc="10")){
    logic="2";
  }
  else if((ab=="10")&&(bc=="11")){
    logic="3";

  }
  else if((ab=="00")&&(bc=="10")){
    logic="4";
  }
  else if((ab=="00")&&(bc=="00")){
  if(digitalRead(pin1)==1){
    logic="6";
  }
  else if((new1!=new2)&&(new2!=new3)){
    logic="5";
  }
  else if((digitalRead(pin2)==1)&&(digitalRead(pin6)==1)){
    logic="7";
  }
  else if(digitalRead(pin2)==1){
     logic="8";
  }
  else if((new4!=new5)&&(pin15==1)){
    logic="9";
  }
  else if((digitalRead(pin9)==1)&&(pin15==1)){
    logic="10";
  }
```

```
1-Need an ENG technician
2-Technician Working
3-Work Completed
4-Work Accepted
5-F1 Running
6-Emergency Stop
7-Low Speed Running
8-Garment Knitting
9-Error Stop
10-Force Stop
```

- **Knitting machine conditione published**

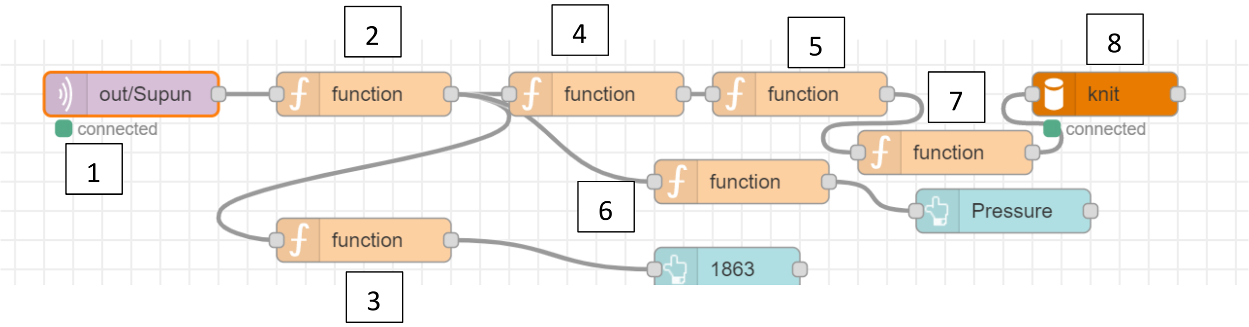
```
  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
    byte arrsize=st.length()+1;
    char msg[arrsize];
    st.toCharArray(msg,arrsize);
    if(logic!=logic1){
      time1=millis();
      Serial.println(msg);
    client.publish(outTopic, msg);
    st="";
    }
```

Node-red

We use node red flow to subscribe MQTT data and process. The node-red Flow was used for following purposes. This is the open source IoT platform.

- To receive MQTT data
- To translating that data into something that can read by human
- To create a dashboard to view real time position of the machine

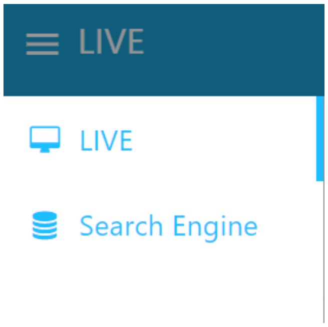
- To insert status of machine to MySql data base



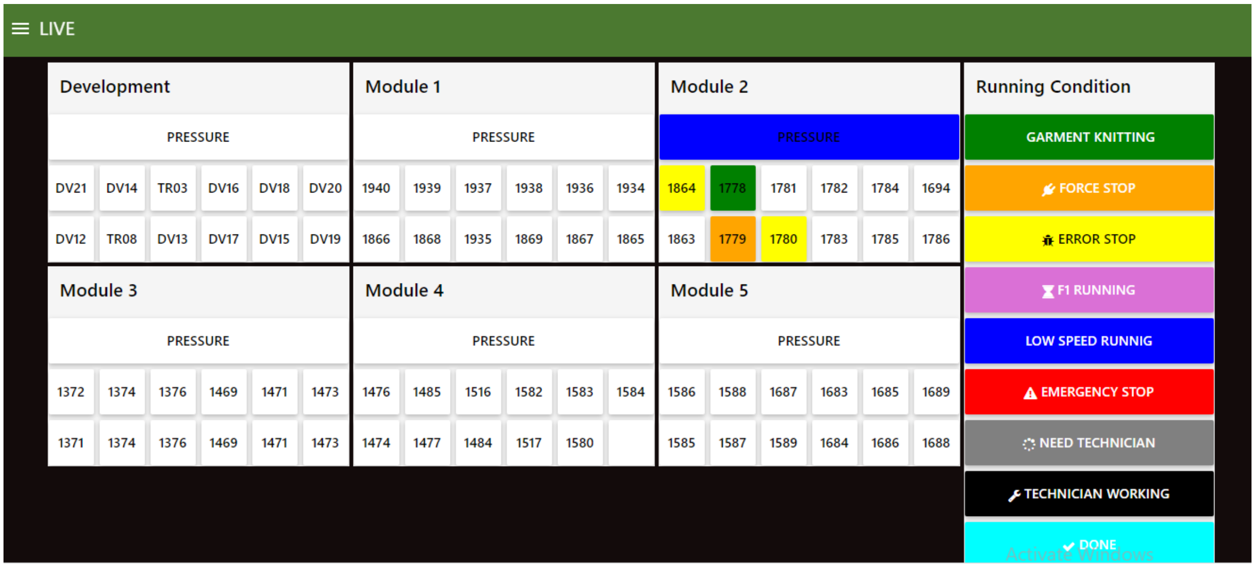
1- Receive MQTT data
   Data-Current_state#Previous_state#pressure+Time
2- Split data and current state send to function 3
3- Decode the Current state and send it to dashboard
4- Split previous_state and Time
5- Decode the previous state
6- Send the current pressure to dashboard
7- Store database query
8- Store machine condition and time on MySQL database

User Interface design

User interface design by using node red dashboard. We can access user interface in any device. Dashboard include two tab that are LIVE and Search Engine.



LIVE tab shows real time machines condition and pressure of each module.



LIVE

Development

PRESSURE

DV21	DV14	TR03	DV16	DV18	DV20
DV12	TR08	DV13	DV17	DV15	DV19

Module 1

PRESSURE

1940	1939	1937	1938	1936	1934
1866	1868	1935	1869	1867	1865

Module 2

PRESSURE

1864	1778	1781	1782	1784	1694
1863	1779	1780	1783	1785	1786

Module 3

PRESSURE

1372	1374	1376	1469	1471	1473
1371	1374	1376	1469	1471	1473

Module 4

PRESSURE

1476	1485	1516	1582	1583	1584
1474	1477	1484	1517	1580	

Module 5

PRESSURE

1586	1588	1687	1683	1685	1689
1585	1587	1589	1684	1686	1688

Running Condition

GARMENT KNITTING

FORCE STOP

ERROR STOP

F1 RUNNING

LOW SPEED RUNNIG

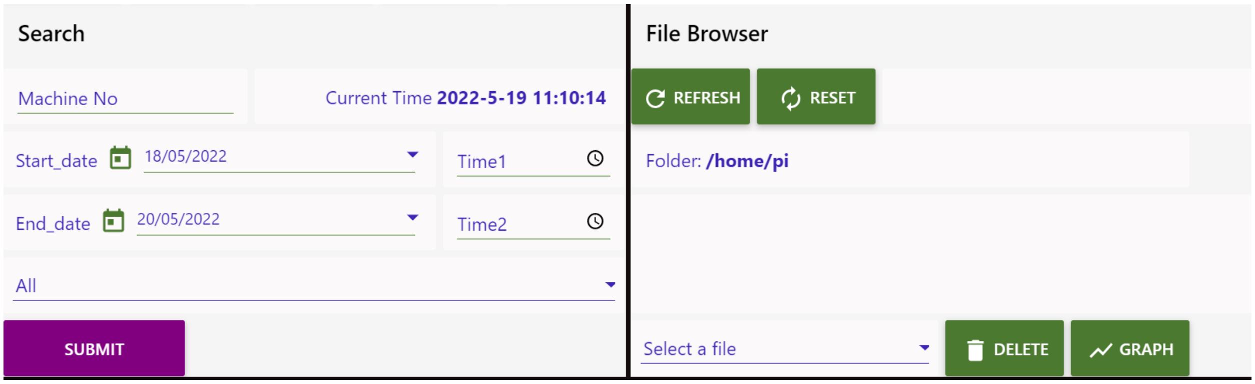
EMERGENCY STOP

NEED TECHNICIAN

TECHNICIAN WORKING

✓ DONE

The search engine tab gives us the option to filter the database and the filtered data can be saved in the excel sheet. That data can be viewed through the graph.



Search

Machine No

Current Time **2022-5-19 11:10:14**

Start_date 18/05/2022

Time1

End_date 20/05/2022

Time2

All

SUBMIT

File Browser

REFRESH

RESET

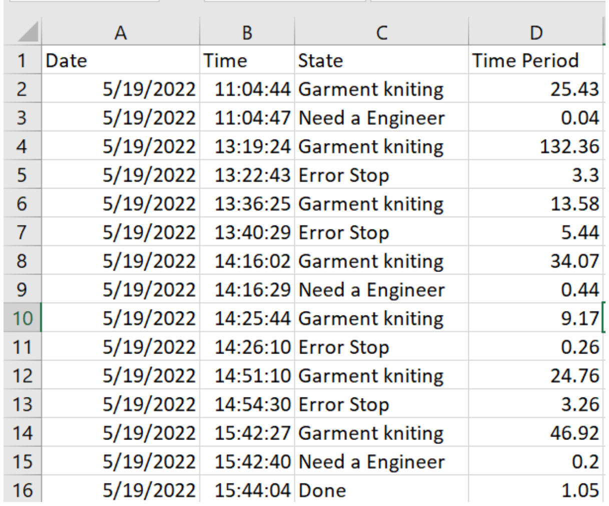
Folder: **/home/pi**

Select a file

DELETE

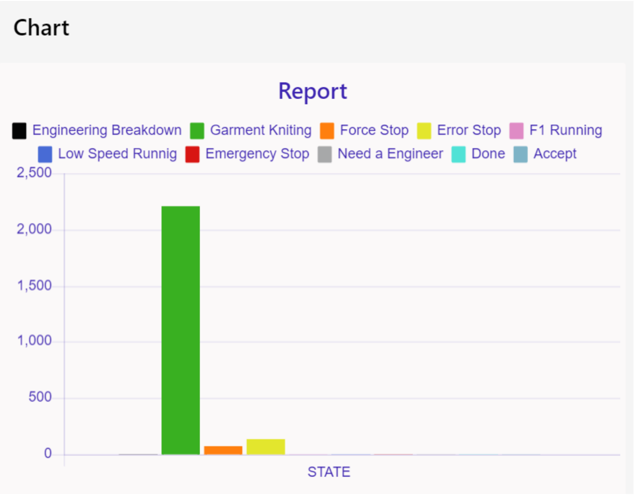
GRAPH

Filtering Search Engine



	A	B	C	D
1	Date	Time	State	Time Period
2	5/19/2022	11:04:44	Garment kniting	25.43
3	5/19/2022	11:04:47	Need a Engineer	0.04
4	5/19/2022	13:19:24	Garment kniting	132.36
5	5/19/2022	13:22:43	Error Stop	3.3
6	5/19/2022	13:36:25	Garment kniting	13.58
7	5/19/2022	13:40:29	Error Stop	5.44
8	5/19/2022	14:16:02	Garment kniting	34.07
9	5/19/2022	14:16:29	Need a Engineer	0.44
10	5/19/2022	14:25:44	Garment kniting	9.17
11	5/19/2022	14:26:10	Error Stop	0.26
12	5/19/2022	14:51:10	Garment kniting	24.76
13	5/19/2022	14:54:30	Error Stop	3.26
14	5/19/2022	15:42:27	Garment kniting	46.92
15	5/19/2022	15:42:40	Need a Engineer	0.2
16	5/19/2022	15:44:04	Done	1.05

Excel Sheet



Graph