OEE MONITORING SYSTEM

OEE is calculated by multiplying the three OEE factors: Availability, Performance, and Quality.

In our project, we are using the ESP32 development board to get data from the IR sensor, and Push-button. Then the ESP32 is processing the data and sending them to Cayenne Dashboard.

Explanation of the code:

```
1  #include <CayenneMQTTESP32.h>
2
3  #define CAYENNE_PRINT Serial
4  #define current_sensor_pin  32
5  #define ir_sensor_pin       34
6  #define push_button_pin     19
7
```

In line 1, we are including Cayenne MQTT library for ESP32. In line 3 we are using the Serial monitor for debugging purpose. The rest of the lines just defines the ESP32 pin numbers for the current sensor, ir sensor and push button.

```
8   // WiFi network info.
9   char ssid[] = "GanKaiSheng";
10  char wifiPassword[] = "0142788175";
11
```

In line 9 and 10 we are saving the WiFi SSID and password.

```
12  // Cayenne authentication info. This should be obtained from the Cayenne
13  char username[] = "6ef0b0b0-cde3-11eb-883c-638d8ce4c23d";
14  char password[] = "6f16917365696511cd2fab20811d78bbb7f41927";
15  char clientID[] = "8af6d3a0-d40c-11eb-b767-3f1a8f1211ba";
```

In line 13 to 15, we are giving the ESP32 our cayenne dashboard credentials to communicate with our cayenne dashboard.

```
17 int idea_cycle_time = 50; //seconds, which means after every 50 seconds there will be a
18 uint64_t planned_production_time = 36000; //seconds which is equal to 10 Hours
19
20 uint64_t  cst, irt, pbt, lrt, onTime = 0;
21 int totalCount = 0, defectCount = 0;
22
23 float current_sensor_offset = 2500.0; //offset
24 float current_sensor_sensitivity = 100.0; // use 100 for 20A Module and 66 for 30A Modu
25 float current_sensor_adcvalue = 0.0;
26 float current_sensor_voltage = 0.0;
27 float current_sensor_current = 0.0;
28 float offCurrent = 0.5;
29
30 bool lastState, currState;
31
32 float performance = 0;
33 float availability = 0;
34 float quality = 0;
35 float oee = 0;
```

In line 17, we are giving the ideal cycle time from our estimated unit pass. In line 18 we are giving planned production time. These were taken from the project description.

1- IR SENSOR(Digital Infrared Sensor (SN-E18-B03N1)) = it's tfunction to dedicate the object pass through the sensor on the conveyor so in this sensor we can calculate the Performance part

For example we estimate 70 units pass though in one hour and the sensor dedicate 64 units so we can calculate the Performance.

Performance = (Ideal Cycle Time × Total Count) / Run Time

(how many objects pass though the sensor)

In line 20 we are taking some variable to store the time to take sensor data periodically.
From line 21 to 35 we have some variables for storing various data

```
37 void setup() {
38   Serial.begin(9600);
39   Cayenne.begin(username, password, clientID, ssid, wifiPassword);
40
41   pinMode(current_sensor_pin, INPUT);
42   pinMode(ir_sensor_pin, INPUT);
43   pinMode(push_button_pin, INPUT_PULLUP);
44
45   cst = millis();
46   irt = millis();
47   pbt = millis();
48   lrt = millis();
49 }
50
```

In the setup function, we are setting up the serial monitor to debug and see values there.
In line 39, we are beginning to communicate with the cayenne dashboard. In line 41-43, we are setting up input pins for the sensors. In line 45-48 we are storing the initial time for the periodic operation of sensors.

```
51 void loop() {
52   Cayenne.loop();
53
```

In loop function line 52 is for continuing the communication with the dashboard.

```
54   if (millis() - cst > 1000) {
55     cst = millis();
56     Serial.print("Current value: ");
57     Serial.println(get_current());
58     if (get_current() > offCurrent) {
59       onTime += 1;
60     }
61   }
```

In line 54 to 61, we are calculating current by the sensor, after every 1000 milliseconds.
If the current sensor is getting some value of current, we will add 1 second with the onTime (total runtime).

```
63   if (millis() - irt > 1000) {
64     irt = millis();
65     get_ir();
66   }
67
```

In line 63 to 65, we are checking with the IR sensor every 1000 milliseconds if there is a new product or not.

```
68   if (digitalRead(push_button_pin) == LOW) {
69     if (millis() - pbt > 500) {
70       pbt = millis();
71       defectCount++;
72       Serial.print("Bad count: ");
73       Serial.println((int)defectCount);
74     }
75   }
```

Here we are checking if the Pushbutton is pressed or not. If pressed, the defect count will increase by 1.

```
77    performance = (1.0 * idea_cycle_time * totalCount ) / onTime;
78    availability = (1.0 * onTime) / planned_production_time;
79    quality = 1.0 - (1.0 * defectCount / totalCount);
```

Here we are calculating the Values according to the details

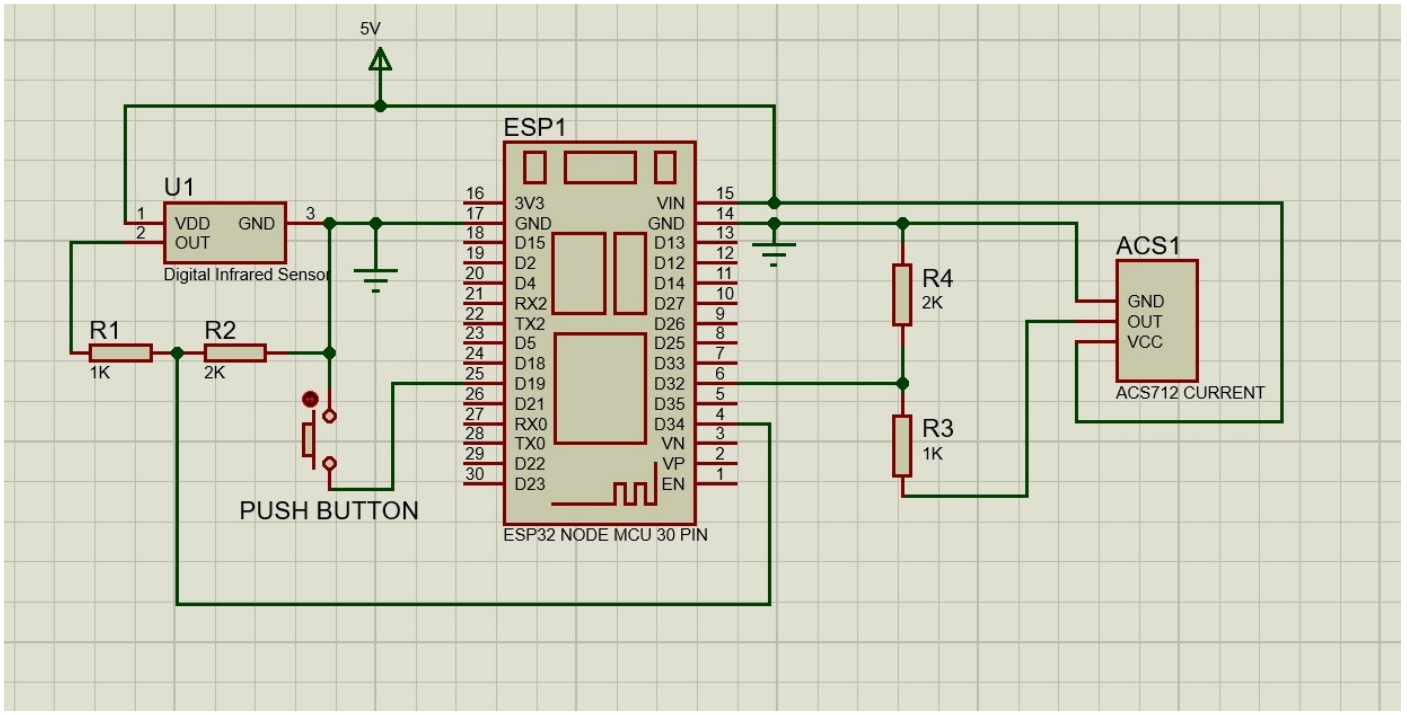
```
81    if (millis() - lrt > 10000) {
82      lrt = millis();
83      oee = performance * availability * quality;
84      Cayenne.virtualWrite(0, performance, "Performance", "null");
85      Cayenne.virtualWrite(1, availability, "Availability", "null");
86      Cayenne.virtualWrite(2, quality, "Quality", "null");
87      Cayenne.virtualWrite(3, oee, "OEE", "null");
88      Serial.println("Data Send to Cayenne Dashboard!!");
89    }
```

Here, after every 10 seconds(10000 milliseconds), we are calculating OEE and sending all data to the cayenne dashboard.

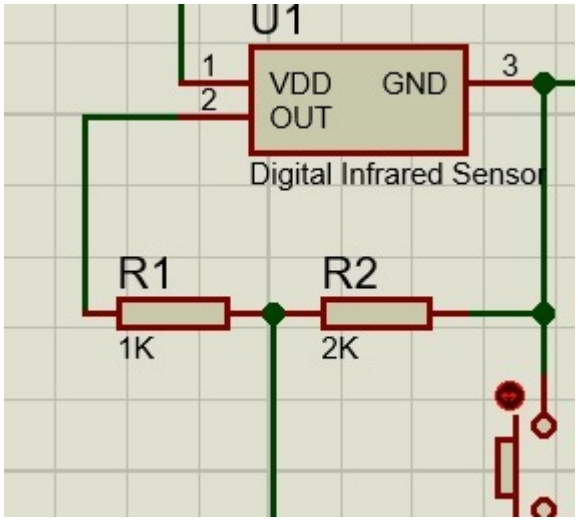
```
92  double get_current() {
93    double average = 0;
94    for (int i = 1 ; i <= 25 ; i++) {
95      current_sensor_adcvalue = analogRead(current_sensor_pin);
96      current_sensor_voltage = ((current_sensor_adcvalue * 5000.0) / 4095.0);
97      current_sensor_current = ((current_sensor_voltage - current_sensor_offset) / current_sensor_se
98      average += current_sensor_current;
99    }
100   return average / 25.0;
101 }
102
103 void get_ir() {
104   currState = digitalRead(ir_sensor_pin);
105   if (lastState != currState && currState == LOW) {
106     totalCount++;
107     Serial.print("Total count: ");
108     Serial.println((int)totalCount);
109   }
110   lastState = currState;
111 }
```

These 2 functions are getting values from the sensors. The current sensor is checking the value of the current. The IR sensor is detecting if there is a product in the line or not. If there is a product it is increasing the total count by 1.

Circuit Diagram:



The output of the IR sensor and Current sensor is going through a voltage divider circuit. The IR and Current sensor works and outputs on logic level 5 volts. But the ESP32 works on logic level 3.3 volts. To make the 5 volts output into 3.3 volts, we are using the voltage divider circuit.



The Current sensor and the IR sensor works at 5 volts. That's why we need to supply 5 volts to the sensors. The outputs of the sensors are connected to the pins of ESP32 like below.

IR sensor -> pin 4 (D34)
Current sensor -> pin 6 (D32)
Push Button -> pin 25 (D19)