Final Deliverables

Coding and Solutioning

Date	**18 May 2023**
Team ID	**NM2023TMID01199**
Project Name	**Smart Billing system for water suppliers**

Coding and Solutioning:

It refers to the process of creating a solution to a problem or task using programming languages and tools. This involves writing codes, testing and debugging, and optimizing the solution for performance and efficiency.

Parameters:

The important parameters of the code are as follows. Such as

- No. of Functional features included in the solution
- Code-layout, Readability and Reusability
- Utilization of algorithms
- Debugging and Traceability
- Exception Handling

Let us see, how these parameters are implemented in my project.

1.No. of Functional features included in the solution:

The number of functional features included in a solution for a smart billing system for water suppliers can vary depending on the specific requirements and complexity of the system. However, here are some common four Functional features that I have developed for this project is listed below.

- Accessing the IBM platform.
- Wifi and mqtt connection.
- Calculating Payload (motorbill) by using random function generator.
- Publishing these payload to Node-RED Platform (web UI).

2. Code-layout, Readability and Reusability:

When developing a smart billing system for water suppliers, it is essential to consider code layout, readability, and reusability. These factors contribute to the overall quality and maintainability of the codebase.

Code- Layout:

Code layouting refers to the organization and formatting of code in a consistent and visually pleasing manner. Proper code layouting improves readability, maintainability, and collaboration among developers.

Readability:

It refers to Write code that is easy to understand and follow. Use meaningful variable and function names that reflect their purpose and intention. Write concise and self-explanatory comments. Explain the purpose of the code, its expected behavior, and any potential caveats or limitations.

Reusability:

Aim to write modular code that can be easily reused in different parts of the system or in future projects. Implement a clear and consistent API for interacting with your smart billing system. This allows other developers to easily integrate or extend its functionality.

So, with reference to these criteria, I have developed my project with ease of code-layouting, read as well as reuse ability. The Layout of this system is

- Accessing the IBM Platform by providing the credentials

```
//-------credentials of IBM Accounts------

#define ORG "4566ei"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"
```

- Customize these credentials

```
//-------- Customise the above values --------
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command
type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

- Setting up the system

```
void setup(){
//statement
}
```

- Mqtt connection

```
void mqttconnect(){
//statement
}
```

- Wifi connection

```
void wificonnect(){
//statement
}
```

- Managing the device

```
void initManagedDevice(){
//statement
}
```

- Callback function

```
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength){
//statement
}
```

- Receiving information from node-red

```
for (int i = 0; i < payloadLength; i++) {

      data3 += (char)payload[i];
}
```

- Calculating the payload (Motorbill) using random function

```
motorbill=random(60,200);

motorbill=motorbill*5;
delay(1000);
PublishData(motorbill);
```

- Publishing the payload to Web UI

```
void PublishData(float motorbill){
//statement
}
```

- Generating the local time

```
void printLocalTime(){
//statement
}
```

These are the outline of my source code. So, it should be readable and the functions are used for reusability.

3.Utilization of algorithm:

Utilizing resources efficiently is crucial in a smart billing system for water suppliers to ensure optimal performance and cost-effectiveness. The major utilizations are like

- Data storage - recent events in IBM
- Network communication- mqtt() and wifi()
- Connecting hardware with software – Internet of things (IOT)
- User Interface – Node-RED (Web UI)

By focusing on efficient resource utilization, a smart billing system for water suppliers can maximize performance, minimize costs, and ensure sustainable operation. Regular monitoring, analysis, and optimization are key to maintaining an optimized system throughout its lifecycle. Implement monitoring tools to track resource usage and identify bottlenecks or areas for optimization. Continuously analyze resource utilization data and optimize system components based on the findings. Employ predictive analytics or machine learning algorithms to forecast resource requirements and optimize resource allocation.

4. Debugging and Traceability:

Debugging and traceability are crucial aspects of a smart billing system for water suppliers. They help in identifying and resolving issues, tracking system behaviour, and ensuring accurate billing processes. the major debugging properties are

- Logging and Error Handling
- Debugging Tools and Techniques
- Error Reporting and Monitoring
- Traceability and Audit Trail
- Test and Debug in Realistic Environments

By incorporating effective debugging and traceability practices in your smart billing system, you can enhance its reliability, ensure accurate billing, and efficiently resolve issues that may arise.

Here, for debugging parameters I use the GDB Debugger website.

https://docs.wokwi.com/gdb-debugging

5.Exception handling:

Exception handling is an important aspect of any software system, including a smart billing system for water suppliers. It helps ensure that your application can gracefully handle unexpected or exceptional situations that may arise during runtime. Here in my project I use different types of handling an exception using if-else conditional statement. such as

a) Check for publishing a data:

```
// To check if it is published or not
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the
cloud then it will print publish ok in Serial monitor or else it will print
publish failed
  } else {
    Serial.println("Publish failed");
  }
```

b) Check for subscription:

```
void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}
```

c) While client is not connected:

```
if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
```

The exception handling should be a part of my system's overall error handling and recovery strategy. It is crucial to handle exceptions in a way that provides meaningful feedback to users and administrators, ensures the integrity of the billing process, and maintains the stability and availability of the system.

CODING:

By considering and using all these above parameters, the source code of our project is implemented as

```
/*
-----------SMART BILLING SYSTEM FOR WATER SUPPLIERS-------------
*/

//Adding the requied libraries
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "time.h"

// Defining the Relay pin
```

```
#define RELAY_PIN 5 // ESP32 pin GIOP5 connected to the IN pin of relay

//Assigning the variables
float time1=0;
float motorbill;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "4566ei"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;
float h, t;
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 0;
const int   daylightOffset_sec = 3600;

//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command
type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id



//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient);



// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.

  Serial.begin(115200);
  pinMode(RELAY_PIN, OUTPUT);
  delay(10);
  Serial.println();
```

```
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  wificonnect();
  mqttconnect();
}

// the loop function runs over and over again forever
void loop() {

  if (!client.loop()) {
    mqttconnect();
  }
}

void PublishData(float motorbill) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"motorbill\":";
  payload += motorbill;
  payload += "}";


  Serial.print("Sending payload: ");
  Serial.println(payload);

  // To check if it is published or not
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud
then it will print publish ok in Serial monitor or else it will print publish
failed
  } else {
    Serial.println("Publish failed");
  }

}

void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
```

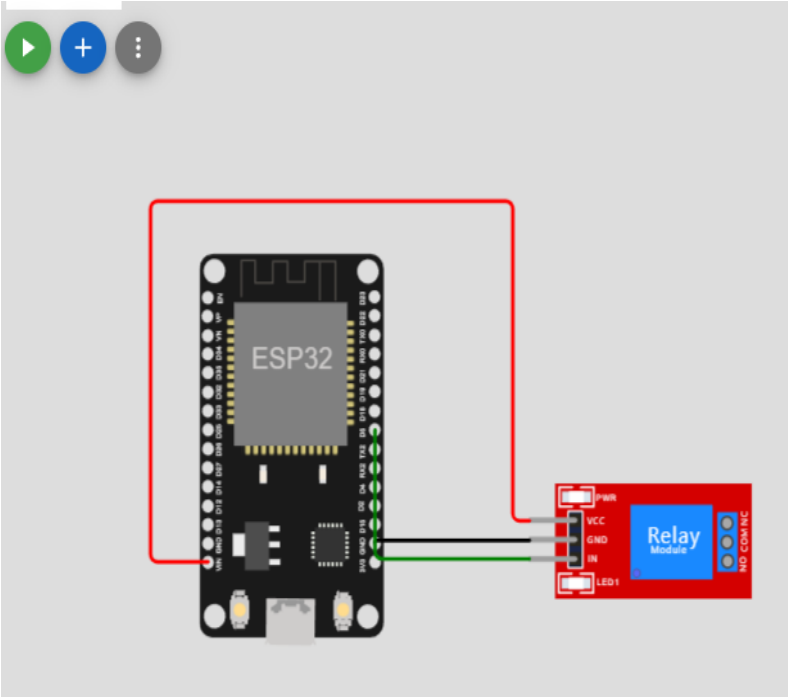
```
    }

     initManagedDevice();
     Serial.println();
  }
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish
the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);

  for (int i = 0; i < payloadLength; i++) {
    data3 += (char)payload[i];
  }

  Serial.println("data: "+ data3);

  // when motor is switched ON
```

```
  if(data3=="on")
  {
Serial.println(data3);
digitalWrite(RELAY_PIN, HIGH);
PublishData(0);
Serial.println("The time at which the motor is switched on:");
printLocalTime();

time1+=1;

  }

// when motor is switched OFF
  else if(data3=="off")
  {

Serial.println(data3);
digitalWrite(RELAY_PIN, LOW);
motorbill=random(60,200);
motorbill=motorbill*5;
delay(1000);
PublishData(motorbill);
Serial.println("The time at which the motor is switched off:");
printLocalTime();
time1=0;

  }

data3="";

}

//Function for print the time
void printLocalTime(){
  struct tm* timeinfo;
  time_t now;
  time(&now);

  timeinfo = localtime(&now);
  Serial.print(timeinfo,"%H:%M:%S");
  Serial.println();
}
```

This is the Major source code of our project “Smart billing system for water suppliers” developed in Wokwi Stimulator.

CONNECTION:

The Connections that we made for this project are shown below.



These connections are made up of by using ESP32 (Micro controller) and Relay. Here, the relay is acts as a switch for motor (ON & OFF).

- Ground of relay is connected to ground of ESP32
- Vcc of relay is connected to vin of ESP32 and
- IN of relay is connected to ESP32 D5 pin.

Reference link:

For your reference, I attached the link here
https://wokwi.com/projects/364076746866810881

This is all about Coding and solutioning phase of our Project. Thus, We have successfully completed this major part here, which is a development phase of our project “Smart billing system for water suppliers”.