PROJECT REPORT

SMART BILLING SYSTEM FOR WATER SU

CONTENTS

Topics

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **IDEATION & PROPOSED SOLUTION**
   - 2.1 Problem Statement Definition
   - 2.2 Empathy Map Canvas
   - 2.3 Ideation & Brainstorming
   - 2.4 Proposed Solution
3. **REQUIREMENT ANALYSIS**
   - 3.1 Functional requirement
   - 3.2 Non-Functional requirements
4. **PROJECT DESIGN**
   - 4.1 Data Flow Diagrams
   - 4.2 Solution & Technical Architecture
   - 4.3 User Stories
5. **CODING & SOLUTIONING**
   - 5.1 Feature 1
   - 5.2 Feature 2
6. **RESULTS**
   - 6.1 Performance Metrics
7. **ADVANTAGES & DISADVANTAGES**
8. **CONCLUSION**
9. **FUTURE SCOPE**
10. **APPENDIX**

1.INTRODUCTION

1.1 Project Overview

A project called the Smart Billing System for Water Suppliers attempts to streamline and automate the billing procedure for water supply businesses. The technology automates metre reading and creates precise invoices based on real-time water consumed. Customers receive billing notifications. The system also has various tools for water suppliers for efficient billing, as well as a customer portal for obtaining billing data. Overall, the system enhances customer satisfaction, increases billing accuracy, and maximises revenue management for water suppliers.

1.2 Purpose

An accurate and effective billing process can be achieved by automating and streamlining it using a smart billing system for water suppliers. The solution removes human metre reading and estimation with the integration of smart metres, resulting in fair and accurate billing based on real-time usage data. It provides clients with convenience by giving them access to their billing information and a variety of payment alternatives. By reducing income leakage and offering analytics and reporting tools for informed decision-making, the system also optimises revenue management. Overall, the goal is to increase billing accuracy, promote customer satisfaction, and improve water supply companies' revenue management.

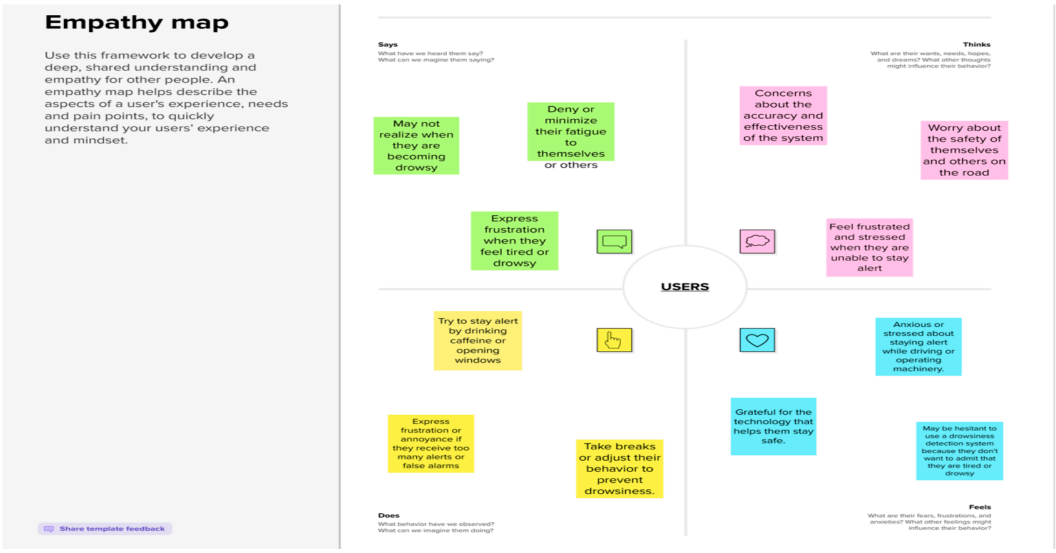
2.IDEATION & PROPOSED SOLUTION

2.1 Problem Statement Definition

In the current scenario, water tanker services are established in cities to supply water to households. To facilitate the payment process, a card system is implemented, where users are issued cards that can be used for payments. A mobile application allows users to top up their cards. Each fill station is equipped with hand-held devices that can read and write data on RFID-based smart cards. These devices also have a Wi-Fi modem to communicate with a central server over the cloud. The data collected is made accessible to users through their respective mobile applications connected to the cloud.

This system provides a convenient way for users to make payments for water tanker services and enables efficient data management and communication between the fill stations, central server, and users. The integration of RFID technology and cloud connectivity ensures secure and real-time transactions, enhancing the overall efficiency and transparency of the water tanker service.

2.2 Empathy map canvas



2.3 Ideation and Brainstorming

2.4 Proposed Solution

The proposed solution for the water billing system includes integrating smart meters to capture real-time water consumption data, implementing a centralized billing system for accurate calculations, automating the billing process, and enabling customer notifications and communication. Multiple payment options will be provided, and a customer portal will be developed for easy access to billing information. Analytics and reporting features will support decision-making, and security measures will be implemented to protect customer data. The solution will be implemented with proper training, ongoing support, and maintenance to ensure a streamlined and efficient water billing system.

3. REQUIREMENT ANALYSIS

3.1 Functional Requirements

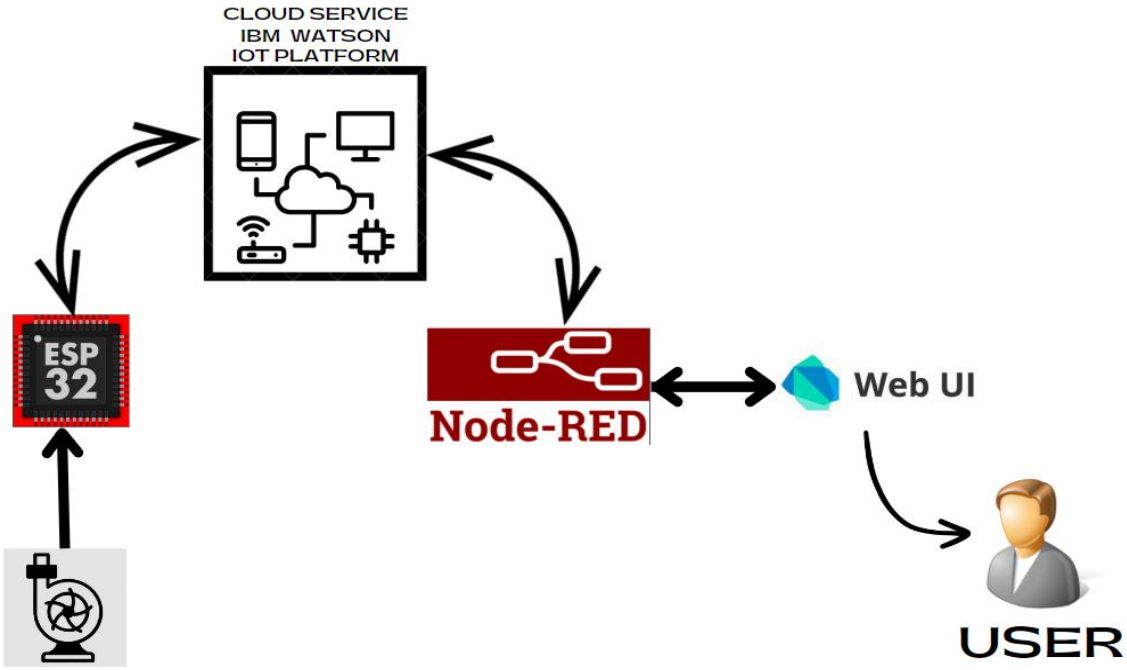
The development of a customer portal for accessing billing information and reporting capabilities, ensuring data security and privacy, integrating with existing systems, ensuring scalability and performance are just a few of the functional requirements for a smart billing system for water suppliers. Other requirements include integrating smart metres for real-time data capture, automating the billing process, sending notifications and reminders to customers, supporting multiple payment options, and supporting multiple payment methods. The overarching goal of these standards is to improve the water billing system's accuracy, consumer convenience, and overall effectiveness.

3.2 Non-function Requirements

For a smart billing system for water suppliers, non-functional requirements include ensuring reliability and availability, high system performance, scalability to handle increasing volumes, robust security measures, user-friendly interfaces, seamless integration with existing systems, compliance with regulations, effective system monitoring and logging, easy system maintenance and upgrades, and thorough support and documentation. The system's dependability, performance, security, usability, and compliance with industry standards and regulations are all improved by these requirements taken as a whole.

4.PROJECT DESIGN

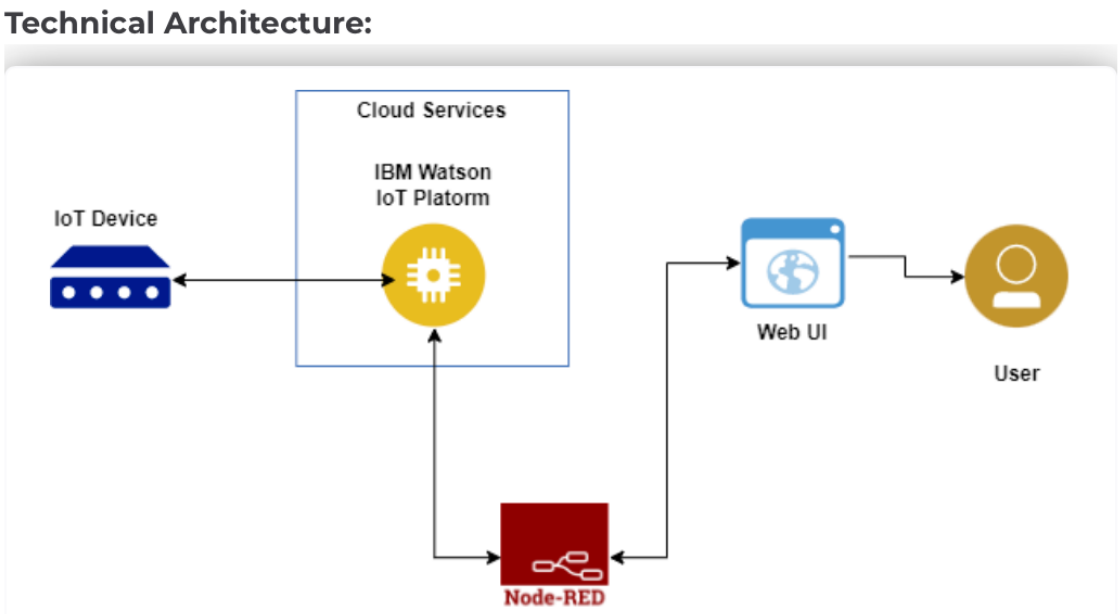
4.1 Data flow diagram



4.2 Solution & Technical Architecture

The solution for a Smart Billing System for Water Suppliers includes integrating smart meters to capture real-time consumption data, developing a centralized billing system, automating billing processes, providing multiple payment options and billing notifications, creating a customer portal for accessing billing information, incorporating analytics and reporting features, ensuring data security and privacy, integrating with existing systems, optimizing scalability and performance, and providing implementation support. This solution aims to streamline billing processes, improve accuracy, enhance customer experience, optimize revenue management, and enable data-driven decision-making for water suppliers.

Architecture:



4.3 User Stories

As a water supplier, I can now create invoices automatically and efficiently with an intelligent invoicing system that has transformed our invoicing process. Previously, our team had to manually collect mileage, calculate invoices and send them to customers, which was a time-consuming and error-prone task. However, with the introduction of a smart billing system, these challenges have become a thing of the past.

Now, when we integrate smart meters into the billing system, real-time water consumption data is automatically collected, so there is no need to read the meters manually. This information is seamlessly transferred to a centralized billing system, where advanced algorithms calculate invoices based on consumer prices, rates and potential discounts. Automating the billing process has greatly improved our efficiency and accuracy. We no longer have to spend hours manually calculating invoices and dealing with possible errors. Instead, the system generates invoices quickly and ensures accurate and transparent invoices for customers.

In addition, the smart billing system has significantly reduced the time and effort required to manage billing functions. Our team can now focus on other important tasks, such as customer service and answering questions, instead of being burdened with manual billing tasks. Customers also benefited from the smart billing system. They receive invoices quickly and access them through a user-friendly customer portal or mobile app. Customers have instant access to their billing history, consumption

trends and payment information, so customers can better understand their water usage habits and make informed decisions about conservation resources.

The automated billing process has increased transparency and trust between us and our customers. They no longer have to worry about inaccurate bills or discrepancies because the system calculates bills based on real-time consumption data. This transparency has increased customer satisfaction and confidence in our billing practices.

Overall, implementing a smart billing system has revolutionized our billing operations, streamlining processes, improving accuracy and customer satisfaction. We have experienced significant time savings, improved efficiency and improved customer relations, allowing us to focus on providing quality water service while ensuring fair and accurate billing to all customers.

5. CODING AND SOLUTIONING

CODE EXPLANATION - NODE RED

The code provided appears to be a configuration file for a Node-RED flow. Here are the features identified in the code:

- **IBM Watson IoT Platform Integration:** The code includes an IBM Watson IoT node (ibmiot in) that is used to connect to the IBM Watson IoT Platform. It enables communication with IoT devices and the exchange of events and commands.
- **Debug Node:** There is a debug node that allows you to inspect the message payload during runtime for debugging purposes. It can be used to view and analyze the data flowing through the flow.
- **Function Node:** A function node is used to define custom JavaScript code logic. In this case, it receives a message payload and performs a conditional check. Depending on the value of the payload, it sets a corresponding message payload for the output.
- **UI Text Node:** The ui_text node is used to display text on the Node-RED Dashboard. It receives the message payload from the previous function node and displays it as dynamic text on the dashboard.
- **Node-RED Dashboard:** The code includes configuration for the Node-RED Dashboard. It defines the appearance and settings of the dashboard, such as the theme, site name, and layout.

  These features collectively form a flow that integrates with the IBM Watson IoT Platform, processes data using custom JavaScript logic, and presents the results on a Node-RED Dashboard.

CODE EXPLANATION – WOKWI

The provided code appears to be an example implementation using the ESP32 microcontroller and the Wokwi platform to create a smart billing system for water suppliers. Here's a brief explanation of the code:

1. Libraries: The code includes the required libraries, such as "WiFi.h" for WiFi connectivity and "PubSubClient.h" for MQTT communication.

2. Global Variables: The code defines global variables for various parameters, including WiFi credentials, IBM Watson IoT platform credentials (ORG, DEVICE_TYPE, DEVICE_ID, TOKEN), and MQTT topics.

3. Setup Function: The setup function initializes the serial communication, sets the relay pin as an output, configures the time using NTP (Network Time Protocol), establishes a WiFi connection, and connects to the MQTT broker.

4. Loop Function: The loop function is where the main logic of the code resides. It continuously checks if the MQTT client is connected and reconnects if necessary. It also contains a placeholder code for controlling the relay (relay pin high/low) and generating a random motor bill. The PublishData function is called to send the motor bill data to the IBM Watson IoT platform.

5. Callback Function: The callback function is invoked when a message is received on the subscribed topic. It parses the payload (message data) and performs actions based on the received command. In this case, if the command is "on," the relay pin is set high, and the motor bill is set to 0. If the command is "off," the relay pin is set low, and a random motor bill is generated and sent to the IBM Watson IoT platform.

6. Helper Functions: The code includes helper functions such as wificonnect for establishing a WiFi connection, mqttconnect for connecting to the MQTT broker, and printLocalTime for printing the current local time.

It's important to note that this code is a simplified example and may require modifications and additional functionality to fit your specific requirements for a smart billing system for water suppliers.

6.Results

6.1 Performance Metrics

1. Billing accuracy: Measure the accuracy of billing calculations and ensure that the generated bills match the actual water consumption recorded by smart meters.

2. Invoice Cycle Time: Track the time required to generate invoices from the data collection stage to the final invoice result. Try to minimize cycle time to ensure efficient billing. 3. System Availability: Monitor the uptime of the system and ensure that it is always available for users. Reduce downtime for maintenance and upgrades.

4. Response Time: Measures the response time of the system when creating invoices, processing payments and obtaining customer information. Aim for fast and responsive performance to improve your user experience.

5. Scalability: Assess the system's ability to handle increasing numbers of clients and increasing data volumes without compromising performance. Watch the system scale as the user base expands.

6. Data processing speed: Measures the speed at which the system processes and analyzes consumption data, generates invoices and performs other important tasks. Optimizing data processing for efficient and timely operation.

7. Customer Satisfaction: Collect customer feedback and satisfaction ratings to gauge their overall experience with the smart billing system. Regularly assess customer satisfaction to identify areas for improvement. 8. Payment Processing Success Rate: Track the success rate of payment transactions and ensure that payments are processed accurately and efficiently without mistakes or errors.

9. System Security: Assess the effectiveness of security measures to protect customer data, prevent unauthorized access and ensure compliance with data protection regulations.

10. Maintenance and Support: Monitor system maintenance and support processes, including response time to resolve issues, customer support satisfaction ratings, and adherence to SLAs.

7. ADVANTAGES & DISADVANTAGES

1. Accuracy and Transparency: The system utilizes real-time consumption data from smart meters, resulting in accurate and transparent billing. Customers can trust that their bills reflect their actual water usage.

2. Efficiency and Automation: The system automates billing processes, eliminating the need for manual meter reading and calculation. This improves efficiency, reduces errors, and saves time for both water suppliers and customers.

3. Cost Savings: By automating processes and reducing manual intervention, water suppliers can achieve cost savings in terms of manpower, resources, and operational expenses.

4. Improved Customer Experience: Customers can access their billing information, payment options, and consumption history through a user-friendly portal or mobile application. This enhances their experience, promotes self-service, and enables better control over their water usage.

5. Data Insights: The system provides valuable data insights, such as consumption patterns and revenue trends, enabling water suppliers to make informed decisions, optimize resource allocation, and develop effective water management strategies.

6. Integration and Compatibility: The smart billing system can integrate with existing systems, databases, and payment gateways, ensuring compatibility and seamless data exchange. This simplifies implementation and reduces disruption to existing processes.

Disadvantages of a Smart Billing System for Water Suppliers:

1. Upfront Costs: Implementing a smart billing system requires upfront investment in smart meters, infrastructure, software development, and integration. The initial costs may be significant, especially for small water suppliers with limited budgets.

2. Technical Challenges: Integrating smart meters, developing robust software, and ensuring data security can present technical challenges. Water suppliers may require specialized expertise or external support to overcome these challenges.

3. System Dependency: The smart billing system relies on consistent and reliable communication between smart meters, hand-held devices, and the central server. Any network or connectivity issues could impact the system's functionality.

4. Data Privacy and Security Risks: The system involves the collection and storage of customer consumption data. Protecting this data from unauthorized access, cyber threats, and privacy breaches requires robust security measures and compliance with data protection regulations.

5. Customer Adoption and Education: Customers may require education and support to understand the new billing system, its benefits, and how to effectively use the customer portal or mobile application. Some customers may be resistant to change or face challenges in adapting to the new system.

6. Maintenance and Support: The smart billing system requires ongoing maintenance, upgrades, and technical support to ensure its smooth operation. Water suppliers need to allocate resources and plan for regular system maintenance and troubleshooting.

8.Conclusion

In conclusion, a smart billing system for water suppliers offers several advantages that can greatly benefit both water suppliers and customers. The system's ability to leverage real-time consumption data from smart meters leads to accurate and transparent billing, improving trust and eliminating discrepancies. Automation of billing processes enhances efficiency, saves time, and reduces errors, resulting in cost savings for water suppliers.

Customers benefit from the convenience of accessing billing information, payment options, and consumption history through user-friendly portals or mobile applications. The system provides valuable insights into consumption patterns and revenue trends, empowering water suppliers to make data-driven decisions and optimize resource allocation. Integration with existing systems and compatibility with payment gateways streamline processes and ensure a seamless transition.

However, it's important to consider the potential disadvantages associated with a smart billing system. These include upfront costs, technical challenges, system dependency, data privacy, customer adoption and education, and ongoing maintenance requirements. Addressing these challenges requires careful planning, expertise, and commitment to data security and customer support.

Despite the challenges, the advantages of implementing a smart billing system outweigh the disadvantages. The system promotes accuracy, efficiency, transparency, and improved customer experience. By embracing technology and automation, water suppliers can enhance their billing operations, optimize revenue management, and foster stronger relationships with their customers.

In conclusion, a smart billing system for water suppliers presents an opportunity for streamlined operations, increased efficiency, and enhanced customer satisfaction. With proper planning, implementation, and ongoing support, water suppliers can leverage the benefits of a smart billing system to revolutionize their billing processes and drive long-term success in the water supply industry.

9.Future Scope

The future scope of a smart billing system for water suppliers is promising and offers several avenues for further development and innovation. Here are some potential areas of future growth and expansion:

1. Advanced Analytics and Predictive Modeling: Enhance the analytics capabilities of the smart billing system to provide more advanced insights and predictive modeling. By leveraging machine learning and artificial intelligence, water suppliers can better forecast water demand, identify potential leaks or abnormalities, and optimize resource allocation.

2. Demand-Side Management: Integrate demand-side management features into the smart billing system, allowing water suppliers to incentivize customers for water conservation efforts. This can involve dynamic pricing structures, personalized recommendations for reducing water consumption, and rewards programs for efficient water use.

3. IoT Integration: Explore the integration of Internet of Things (IoT) technologies to enable a more interconnected water infrastructure. This can involve smart sensors in water distribution networks, real-time monitoring of water quality, and automated alerts for maintenance or repairs.

4. Mobile Payment Innovations: Keep pace with advancements in mobile payment technologies and expand payment options. This could include incorporating digital wallets, contactless payments, and innovative payment methods to provide customers with seamless and convenient ways to pay their water bills.

5. Water Usage Tracking and Notifications: Develop features that allow customers to track their water usage in real-time and receive proactive notifications when consumption patterns deviate from the norm or when leaks are detected. This empowers customers to take immediate action and minimize wastage.

6. Blockchain Technology: Explore the use of blockchain technology for secure and transparent transactions, as well as for maintaining immutable records of water consumption and billing data. Blockchain can enhance data security, privacy, and trust in the billing system.

7. Integration with Smart Home Devices: Integrate the smart billing system with smart home devices and platforms to enable automated water usage control and monitoring. This can include features such as remotely turning off water supply, setting usage limits, and receiving alerts for abnormal usage.

8. Enhanced Customer Engagement: Develop interactive features within the customer portal or mobile application to enhance customer engagement. This could include educational resources on water conservation, community forums for sharing tips and experiences, and gamification elements to encourage sustainable water practices.

9. Expansion to Other Utility Services: Explore the possibility of expanding the smart billing system to encompass other utility services, such as electricity and gas. This would provide customers with a centralized platform for managing and monitoring their overall utility usage and bills.

10. Integration with Smart City Initiatives: Collaborate with smart city initiatives to integrate the smart billing system into broader urban development plans. This can include sharing data and insights with other city systems, such as infrastructure planning, environmental monitoring, and sustainable development initiatives.

10.Appendix

Source code

Code for wowki:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define RELAY_PIN 18 // ESP32 pin GIOP18 connected to the IN pin of relay
#include "time.h"
float time1=0;
float motorbill;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

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

#define ORG "1z668o"//IBM ORGANITION ID
#define DEVICE_TYPE "Relay"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345" //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() {
 // digitalWrite(RELAY_PIN, HIGH);
  //delay(1000);
  //digitalWrite(RELAY_PIN, LOW);
  //delay(1000);
 // motorbill=random(60,200);
  //motorbill=motorbill*5;
  //delay(1000);
  //PublishData(motorbill);
  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);


  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++) {
    //Serial.print((char)payload[i]);
```

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

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

  }

  else if(data3=="off")
  {
  //Serial.print(time1);
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="";



}
void printLocalTime(){
  struct tm* timeinfo;
  time_t now;
  time(&now);

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

  /* Serial.println("Hour: ");
  Serial.println(timeinfo, "%H");
  Serial.print("Hour (12 hour format): ");
  Serial.println(timeinfo, "%I");
  Serial.print("Minute: ");
  Serial.println(timeinfo, "%M");
  Serial.print("Second: ");
```

```
  Serial.println(timeinfo, "%S");*/


  Serial.println();
}
```

Code for Node-red

```
[
    {
        "id": "b4cebfd261883356",
        "type": "tab",
        "label": "Flow 7",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "fbbb78df6cc07468",
        "type": "ibmiot out",
        "z": "b4cebfd261883356",
        "authentication": "apiKey",
        "apiKey": "",
        "outputType": "cmd",
        "deviceId": "12345",
        "deviceType": "streetlight",
        "eventCommandType": "test",
        "format": "String",
```

```
    "data": "Data",
    "qos": 0,
    "name": "IBM IoT",
    "service": "registered",
    "x": 1100,
    "y": 160,
    "wires": []
},
{
    "id": "f153cef746832e9c",
    "type": "debug",
    "z": "b4cebfd261883356",
    "name": "debug 12",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 360,
    "y": 300,
    "wires": []
},
```

```
{
    "id": "a10d8fbaaa72435e",
    "type": "ibmiot out",
    "z": "b4cebfd261883356",
    "authentication": "apiKey",
    "apiKey": "1fa68f6f6ed1ebcf",
    "outputType": "cmd",
    "deviceId": "12345",
    "deviceType": "Relay",
    "eventCommandType": "test",
    "format": "String",
    "data": "Data",
    "qos": 0,
    "name": "IBM IoT",
    "service": "registered",
    "x": 580,
    "y": 340,
    "wires": []
},
{
    "id": "4653cc0a16b48f37",
    "type": "ibmiot in",
    "z": "b4cebfd261883356",
    "authentication": "apiKey",
```

```
"apiKey": "1fa68f6f6ed1ebcf",
"inputType": "evt",
"logicalInterface": "",
"ruleId": "",
"deviceId": "12345",
"applicationId": "",
"deviceType": "Relay",
"eventType": "+",
"commandType": "",
"format": "json",
"name": "IBM IoT",
"service": "registered",
"allDevices": "",
"allApplications": "",
"allDeviceTypes": "",
"allLogicalInterfaces": "",
"allEvents": true,
"allCommands": "",
"allFormats": "",
"qos": 0,
"x": 170,
"y": 120,
"wires": [
    [
```

```
            "9d6c07e5afd023a7",
            "208caa2c346590f7"
        ]
    ]
},
{
    "id": "9d6c07e5afd023a7",
    "type": "debug",
    "z": "b4cebfd261883356",
    "name": "debug 13",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 500,
    "y": 140,
    "wires": []
},
{
    "id": "208caa2c346590f7",
```

```
        "type": "function",
        "z": "b4cebfd261883356",
        "name": "motor bill",
        "func":
"msg.payload=msg.payload.motorbill\nif(msg.payload==0){\n\n
msg.payload=\"The motor is switched on!\"+\"->Billing
started\";\n}\nelse{\n\n    msg.payload=\"The motor is switched off!\"+\"-
>Elapsed motor bill is:\"+msg.payload+\"RUPEES\";\n}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 320,
        "y": 200,
        "wires": [
            [
                "6507a4e5c5e4e715"
            ]
        ]
    },
    {
        "id": "6507a4e5c5e4e715",
        "type": "ui_text",
        "z": "b4cebfd261883356",
```

```
    "group": "b7771ab55bf3a9e9",
    "order": 3299,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "Motor Bill",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "className": "",
    "x": 580,
    "y": 240,
    "wires": []
},
{
    "id": "a6c3cc3a889e83fb",
    "type": "ui_switch",
    "z": "b4cebfd261883356",
    "name": "",
    "label": "SWITCH FOR MOTOR",
    "tooltip": "",
    "group": "b7771ab55bf3a9e9",
    "order": 1,
    "width": "15",
    "height": "4",
```

```
    "passthru": true,
    "decouple": "false",
    "topic": "topic",
    "topicType": "msg",
    "style": "",
    "onvalue": "true",
    "onvalueType": "bool",
    "onicon": "",
    "oncolor": "",
    "offvalue": "false",
    "offvalueType": "bool",
    "officon": "",
    "offcolor": "",
    "animate": false,
    "className": "",
    "x": 340,
    "y": 440,
    "wires": [
      [
        "a10d8fbaaa72435e",
        "f153cef746832e9c"
      ]
    ]
  },
```

```
{
    "id": "1fa68f6f6ed1ebcf",
    "type": "ibmiot",
    "name": "apikey",
    "keepalive": "60",
    "serverName": "1z668o.messaging.internetofthings.ibmcloud.com",
    "cleansession": true,
    "appId": "",
    "shared": false
},
{
    "id": "b7771ab55bf3a9e9",
    "type": "ui_group",
    "name": "Motor Switch and Bill",
    "tab": "fac361559fed2381",
    "order": 1,
    "disp": true,
    "width": "22",
    "collapse": false,
    "className": ""
},
{
    "id": "fac361559fed2381",
    "type": "ui_tab",
```

```
    "name": "Smart Water Billing System",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
  }
]
```


GITHUB: https://github.com/naanmudhalvan-SI/IBM--12032-1682491607/tree/main/FINAL%20DELIVERABLES

DEMO LINK: https://drive.google.com/drive/folders/1b_dWQRrQgpyNrzNFsWZYjMTC-a1rU4Ce

WOKWI LINK: https://wokwi.com/projects/364909105770454017