“Machine to Machine Communication”

Special Assignment Report

Submitted in Partial Fulfillment of the
Requirements for completion of

Course on
2ECDE65 INTERNET OF THINGS

By

Jainil Raval (19BEC108)
Mann Raval (19BEC109)

Department of Electronics and Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

November 2022

CERTIFICATE

This is to certify that the Special Assignment Report entitled "Machine to Machine Communication" submitted by Mr. Jainil Raval(19BEC108) and Mr. Mann Raval (19BEC109) towards the partial fulfillment of for completion of Course on 2ECDE65 INTERNET OF THINGS is the record of work carried out by him/her individually.

Date: 11/11/2022
Faculty Coordinator:

Undertaking for Originality of the Work

We, Jainil Raval(19BEC108) and Mann Raval(19BEC109) give undertaking that the Special Assignment Report entitled “Machine to Machine Communication” submitted by us, towards the partial fulfillment of for completion of Course on 2ECOE02 WIRELESS SENSOR NETWORK, is the original work carried out by us and we give assurance that no attempt of plagiarism has been made. we understand that in the event of any similarity found subsequently with any other published work or any report elsewhere; it will result in severe disciplinary action.

Jainil Raval(19BEC108) and Mann Raval(19BEC109)

Signature of the Student

Date: 11/11/2022

Place: Ahmedabad

Abstract

Machine-to-machine (M2M) technologies are becoming more widespread across all industries and regions, and they may be a crucial enabler of services and products serving a wide range of vertical markets such as health-care, transport, utilities, educational research and its development, logistics etc. All of the top electronic communications operators in the world currently have several million M2M users in their mobile networks, making M2M - as a component of the Internet of Things (IoT) - one of the current main drivers behind the growth in mobile customers. The M2M communication paradigm is associated with the interaction of IoT endpoints, which, from the perspective of business processes and end user applications, represent service endpoints that may be readily integrated into growing service-oriented enterprise systems and service delivery platforms.

In order to take advantage of the anticipated growth of the M2M market, regulators, telecoms carriers, and service providers will all need to be quick to adapt. Convergence of domains such as technologies, electronic communications, and intelligence which is enabled by nanoelectronics, context aware technologies, sensors, and cloud computing which will result in the challenge of embedding real world information into networks, services, and applications. This will lead to the development of different services like novel services, new interfaces, innovative products, and new applications.

INDEX

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Introduction

A form of data transmission which involves one or more entities but does not necessary involve any human involvement or intervention known as machine-to-machine (M2M) communication. M2M is also known as Machine Type Communication (MTC). It varies from current communication models in that it could potentially involve a very large number of communicating terminals, different or new market scenarios, lower costs and effort, and little traffic per terminal. In general, M2M communication could be carried over mobile networks like CDMA, GSM-GPRS, and EVDO networks. Mobile networks' primary function in M2M communication is that of a transport network. M2M presents both exceptional prospects and distinct challenges[1][2].

1.2 Importance of the topic

Machine-to-machine technology's basic objective is to gather sensor data and send it via a network. M2M systems frequently use open networks and access techniques like cellular or Ethernet to make them more affordable than SCADA or other remote monitoring solutions. Home appliances and other technologies are given real-time operational control and data transfer capabilities because to the employment of M2M in this embedded system. Robotics, security, traffic control, fleet management, logistics and automotive are other fields where M2M is crucial.

1.3 Objective of the assignment

Other transceiver modules that support wireless communication were substituted in this project. One of our ESP8266-based boards was configured as a WiFi-STA, and the other as a WiFi-AP. The ESP8266 board was utilized in this project in addition to NodeMCU, which can also be used. The goal is to transfer data from one machine to another machine, and for this, we have used buttons as input on one side and LEDs as output on the other side.

1.4 Scope of the assignment

The scope of the project is to create a system with interconnected equipment that can interact. Here, we have two NodeMCU modules, one of which will function as a server and the other as a client.

1.5 Organization of the Rest of the Report

In the chapter 2 we have discussed the previous work which was done for the Machine-to-Machine communication and the various methods they have used for the communication. In the chapter 3 we have discussed about the block diagram and the working of the Machine-to-machine communication which was implemented on NodeMCU. In the chapter 4 the various hardware details about the project like components used and why we used that components and bill of material and in chapter 5 the software details like the software used and the flow chart is discussed. Then further the conclusion and the future scope is discussed in chapter 6.

Chapter 2

Literature review

2.1 Work done in area

The NodeMCU ESP8266 connected to a DHT 11 temperature sensor is represented by Machine A (Client) in the system overview, which features two machines. The client node's job is to read the ambient temperature, determine whether the reading has exceeded the threshold, and, if so, connect with the server[2]. The NodeMCU ESP8266 is also represented by Machine B (Server), and it is linked to a DC relay as depicted. Receiving notifications from the client and choosing whether to turn on or off the actuator are the duties of the server node. TCP/IP is a connection establishment protocol across client and server and is kept active until the application data at each end has finished exchanging. The connection is wireless[3].

The IoT paradigm accepts several forms of communication. For instance, WSNs, M2M communications, and Cyber Physical Systems (CPS) all fall under the umbrella of IoT because they all share a number of criteria and features. Since M2M enables autonomous data sharing among devices without human involvement, it stands out. However, the design of the system is severely constrained by the limitations imposed by these devices, particularly for capillary M2M systems, especially in terms of energy and radio usage efficiency, which directly affects the Internet-enabled communication stack[4][5].

Chapter 3

Block Diagram and Working

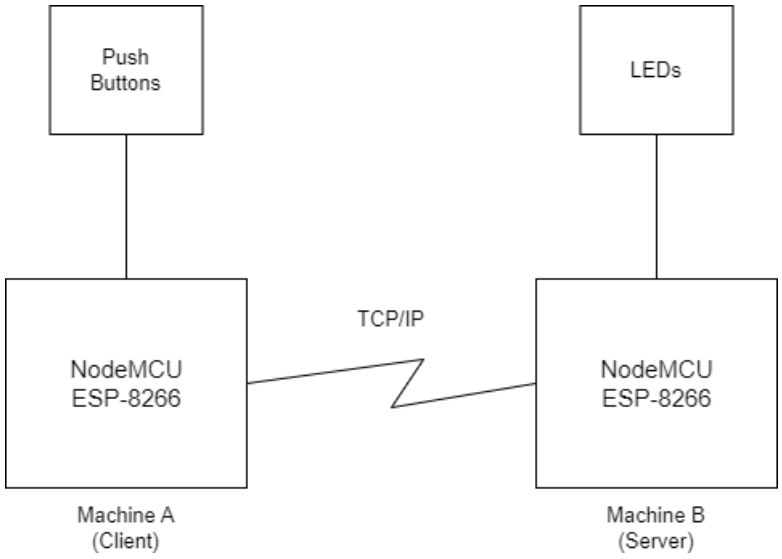

Fig 3.1 Block Diagram

Working

The NodeMCU ESP8266 is represented as Machine A (Client) in Fig. 3.1's system overview, which is made up of two machines. The client node's job is to recognize when a push button is pressed, process the information, and send the information to the server. The NodeMCU ESP8266 is also represented by Machine B (Server), which is linked to a string of LEDs. Receiving notifications from the client and choosing whether to turn on or off the LED are the duties of the server node. TCP/IP (Transfer Control Protocol/Internet Protocol) is a session establishment protocol between client and server and is kept active until the application data at each end has finished exchanging. The connection is wireless.

Chapter 4

Hardware Design and software

4.1 Components Used

In this project the components used were the NodeMCU (ESP-8266), LEDs, Push Buttons, resistors, Bread board and Jumper wires.

4.2 Reason for selecting the components

In the machine-to-machine communication we have taken the two NodeMCU which will communicate with each other. We have chosen NodeMCU for the data transfer using the Wi-Fi. We have used push button as switch LEDs as indicator and resistor to limit the current.

4.3 Bill of Material

Sr No.	Components name	Price
1	NodeMCU (ESP-8266) x 2	395
2	LEDs x 4	20
3	Push Button x 4	20
4	Breadboard	100
5	Resistors 1k ohm x 8	10

Table 4.1 Bill of Materials

4.4 Schematic of the Hardware

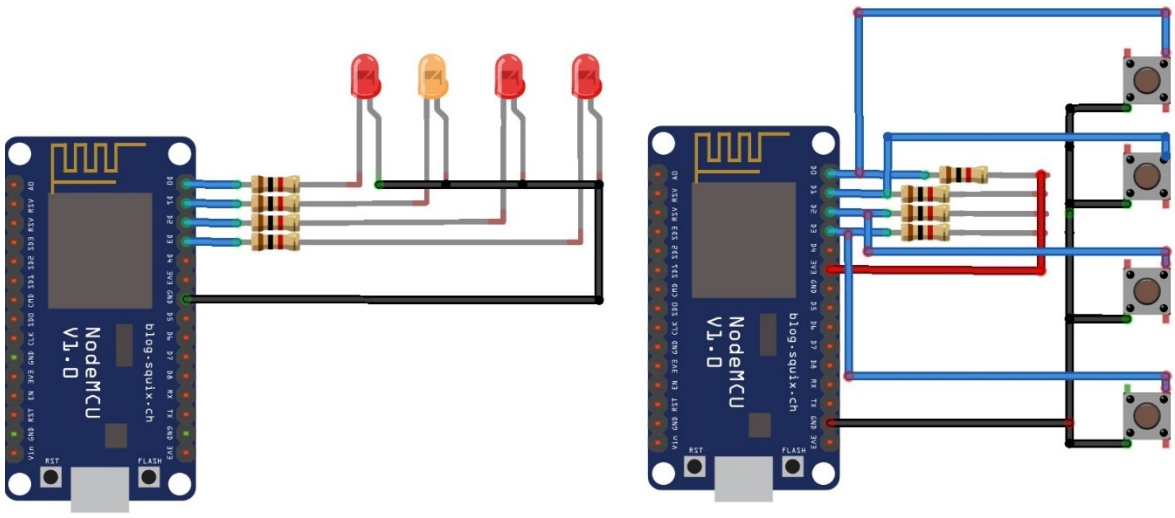

Fig 4.1 Schematic diagram

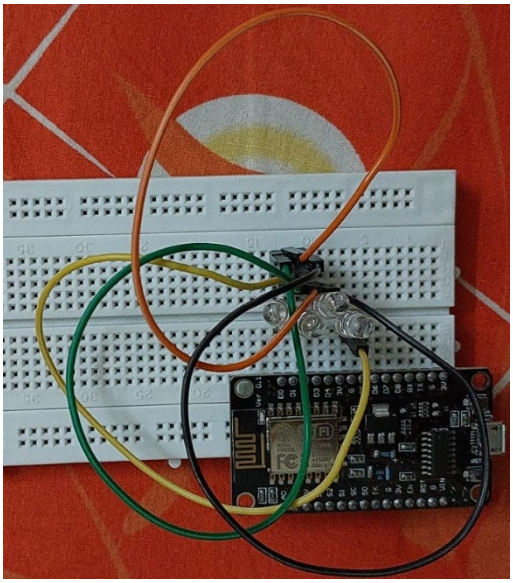

Fig 4.2 Server-side implementation

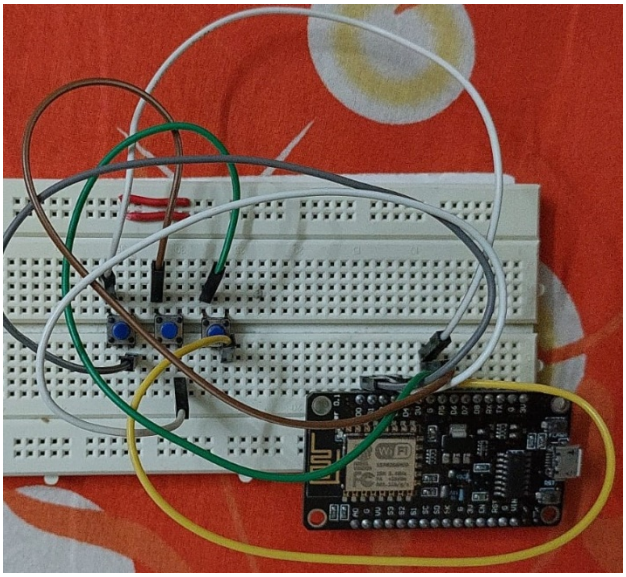

Fig 4.3 Client-side implementation

Software

4.5 Programming tools used

The project includes the NodeMCU which needs to be programmed and for that we have used the Arduino IDE for the programming of both the NodeMCU one from the server side and other is from client side.

Chapter 5

Conclusion and Future Scope

Creating and implementing a low-cost M2M communication system based on two ESP8266 NodeMCU boards was the aim of this endeavor. One node will serve as a Station (Client), which is coupled to a push button that serves as a switch, and the other node will serve as an Access Point (Server). The client should contact the server whenever the push button is pushed. The server will either turn on or off the LED depending on the circumstance. The system showed a maximum communicating distance of 10 meters during distance analysis; at this distance, the Station loses connection with the Access Point. Also, we can add some sensors at the client side and which will sense the data and the processes the data and if found useful it can send it to server and on basis of the data it can turn on or off some actuators.

References

[1] Pethiyagoda, Nadin & Zain, Oshadi & Weerasekara, Rusiru. (2021). Machine to Machine (M2M) Communication Using ESP8266. 10.13140/RG.2.2.35872.71689.

[2] C. Makaya, M. -Y. Lai and F. J. Lin, "Over-the-air remote management and control of IP-based M2M devices," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 172-176, doi: 10.1109/WF-IoT.2015.7389047.

[3] N. V. R. Kumar, B. S. B. Praveen, A. V. S. Reddy and B. B. Sam, "Study on IOT with reference of M2M and WiFi," 2017 International Conference on Information Communication and Embedded Systems (ICICES), 2017, pp. 1-6, doi: 10.1109/ICICES.2017.8070754.

[4] Y. Yang, H. Ye, S. Fei, Y. Yang, H. Ye and S. Fei, "Design of communication interface for M2M-based positioning and monitoring system," 2011 International Conference on Electronics, Communications and Control (ICECC), 2011, pp. 2624-2627, doi: 10.1109/ICECC.2011.6067754.

[5] A. Temprilho, L. Nóbrega, P. Pedreiras, P. Gonçalves and S. Silva, "M2M Communication Stack for Intelligent Farming," 2018 Global Internet of Things Summit (GIoTS), 2018, pp. 1-6, doi: 10.1109/GIOTS.2018.8534560.

Appendix

Code:

Client Side code

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#define btn0 16
#define btn1 5
#define btn2 4
#define btn3 0


const char *ssid = "ssid";
const char *password = "password";

WiFiServer server(80);

int sensorValue0 = 0;       //sensor value, I'm usingg 0/1 button state
int sensorValue1 = 0;
int sensorValue2 = 0;
int sensorValue3 = 0;

void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(btn0, INPUT_PULLUP);
  pinMode(btn1, INPUT_PULLUP);
  pinMode(btn2, INPUT_PULLUP);
  pinMode(btn3, INPUT_PULLUP);


  // set the ESP8266 to be a WiFi-client
  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(ssid, password);

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

  }

}

void loop() {
if(digitalRead(btn0) == LOW) sensorValue0 = 1;
if(digitalRead(btn1) == LOW) sensorValue1 = 1;
if(digitalRead(btn2) == LOW) sensorValue2 = 1;
if(digitalRead(btn3) == LOW) sensorValue3 = 1;

if(digitalRead(btn0) == HIGH) sensorValue0 = 0;
if(digitalRead(btn1) == HIGH) sensorValue1 = 0;
if(digitalRead(btn2) == HIGH) sensorValue2 = 0;
if(digitalRead(btn3) == HIGH) sensorValue3 = 0;


  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const char * host = "192.168.4.1";            //default IP address
  const int httpPort = 80;

  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request. Something like /data/?sensor_reading=123
  String url = "/data/";
  url += "?sensor_reading=";
  url +=
"{\"sensor0_reading\":\"sensor0_value\",\"sensor1_reading\":\"sensor1_value\",\"sensor2_reading\":\"sensor2_value\
",\"sensor3_reading\":\"sensor3_value\"}";
```

```
  url.replace("sensor0_value", String(sensorValue0));
  url.replace("sensor1_value", String(sensorValue1));
  url.replace("sensor2_value", String(sensorValue2));
  url.replace("sensor3_value", String(sensorValue3));


  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 500000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }
}
```

Server Side code

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ArduinoJson.h>

#define led0 16             //D0
#define led1 5              //D1
#define led2 4              //D2
#define led3 0              //D3

DynamicJsonBuffer jsonBuffer;

const char *ssid      = "ssid";
const char *password  = "password";
```

```
int sensorValue0 = 0;
int sensorValue1 = 0;
int sensorValue2 = 0;
int sensorValue3 = 0;
String sensor_values;

ESP8266WebServer server(80);

void handleSentVar() {

 if (server.hasArg("sensor_reading"))
 {
  sensor_values = server.arg("sensor_reading");
  Serial.println(sensor_values);
 }
 JsonObject& root = jsonBuffer.parseObject(sensor_values);
{
  sensorValue0         = root["sensor0_reading"].as<int>();
  sensorValue1         = root["sensor1_reading"].as<int>();
  sensorValue2         = root["sensor2_reading"].as<int>();
  sensorValue3         = root["sensor3_reading"].as<int>();

// }

 Serial.println(sensorValue0);
 Serial.println(sensorValue1);
 Serial.println(sensorValue2);
 Serial.println(sensorValue3);

 toggle_leds();

 server.send(200, "text/html", "Data received");
}


void setup() {
 Serial.begin(9600);
 WiFi.softAP(ssid, password);
```

```
  IPAddress myIP = WiFi.softAPIP();

  pinMode(led0, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);

  //toggle_leds();                //turn off all leds as all the sensor values are zero

  server.on("/data/", HTTP_GET, handleSentVar); // when the server receives a request with /data/ in the string then
run the handleSentVar function
  server.begin();
}

void loop() {
  server.handleClient();
}

void toggle_leds()
{
  if (sensorValue0 == 0)  digitalWrite(led0, LOW);
  if (sensorValue1 == 0)  digitalWrite(led1, LOW);
  if (sensorValue2 == 0)  digitalWrite(led2, LOW);
  if (sensorValue3 == 0)  digitalWrite(led3, LOW);

  if (sensorValue0 == 1)  digitalWrite(led0, HIGH);
  if (sensorValue1 == 1)  digitalWrite(led1, HIGH);
  if (sensorValue2 == 1)  digitalWrite(led2, HIGH);
  if (sensorValue3 == 1)  digitalWrite(led3, HIGH);
}
```