Netaji Subhas University of Technology

IoT workshop

EIECC19

Project Report File

IoT based Notice Board

Submitted to:-
Prof. Dhananjay V. Gadre

Submitted by:-
Deepanshu (2020UEI2810)
Samyak (2020UEI2809)

ABSTRACT

The IOT Notice Board is a revolutionary technology that is transforming the way we communicate in our modern, interconnected world. By harnessing the power of the Internet of Things (IoT), this notice board is able to take text input from a mobile app and display it on a LED matrix display, making it accessible from anywhere in the world.

Traditionally, notice boards have been used in a variety of settings, from schools and universities to offices and public spaces. They serve as an important means of communication, allowing individuals to share important information, announcements, and updates with a large audience. However, these notice boards are typically limited in their reach, as they are only visible to those who are physically present in the same location.

With the advent of the IOT Notice Board, this limitation is eliminated. By using a mobile app to input text, users can communicate with the notice board from anywhere in the world. This means that important information, such as emergency alerts or critical updates, can be quickly and easily shared with a global audience.

The LED matrix display used by the IOT Notice Board is a cutting-edge technology that offers a range of benefits over traditional displays. These displays are energy-efficient, highly visible, and can be programmed to display a variety of information, including text, images, and animations. Additionally, the use of LED technology ensures that the display is long-lasting and requires minimal maintenance.

One of the key advantages of the IOT Notice Board is its ease of use. The mobile app used to input text is simple and intuitive, allowing anyone to quickly and easily share information with the display. Additionally, the display itself can be customized to suit the needs of any organization or individual. This means that the IOT Notice Board can be used in a variety of settings, from schools and universities to hospitals and airports.

In conclusion, the IOT Notice Board is an innovative technology that is transforming the way we communicate in our modern, interconnected world. By allowing individuals to input text from anywhere in the world, this notice board offers a new level of accessibility and convenience. With its energy-efficient LED matrix display and easy-to-use mobile app, the IOT Notice Board is poised to become a ubiquitous presence in public spaces and organizations around the globe.

1 Introduction

1.1 Description

The IOT Notice Board is a state-of-the-art display system that revolutionizes the way we convey information to others. It is an innovative solution that utilizes the power of the Internet of Things (IOT) technology to create a more dynamic and efficient display board. The IOT Notice Board can receive text input from a mobile app, which can be displayed on an LED matrix display in real-time. This makes it possible for the display board to receive inputs from anywhere across the globe, providing a truly global reach for your message.

The IOT Notice Board is a perfect solution for organizations that need to communicate with their employees, customers, or stakeholders in real-time. It can be used in a variety of settings, such as corporate offices, public spaces, educational institutions, or retail outlets. The LED matrix display offers a clear and vibrant display of text messages, making it easy to read from a distance.

One of the significant advantages of the IOT Notice Board is its ease of use. The mobile app is user-friendly and intuitive, allowing anyone to input messages quickly and easily. The display board is also easy to install and set up, making it a hassle-free solution for organizations that need a reliable communication system.

Another significant advantage of the IOT Notice Board is its ability to customize the display content. Organizations can display a wide range of content, including news updates, weather forecasts, promotional messages, and even personalized messages for special occasions. The LED matrix display can also be programmed to display images, animations, or videos, making it an engaging and interactive communication tool.

In conclusion, the IOT Notice Board is a cutting-edge technology that offers a modern and efficient way to convey information. It is an innovative solution that utilizes the power of IOT to create a global communication network. The ease of use and customization options make it a perfect solution for organizations that need to communicate with their audience in real-time. With the IOT Notice Board, organizations can easily and effectively convey their message to a global audience.

1.2 Motivation

In today's fast-paced and highly interconnected world, it has become increasingly important to have access to real-time information from multiple sources. Whether it is for public communication, emergency announcements, or private messaging, there is a growing need for a communication system that is fast, reliable, and accessible from anywhere in the world. This is where the "IOT Notice Board" comes in.

The IOT Notice Board is a cutting-edge technology that leverages the power of the Internet of Things (IoT) to enable real-time communication between users and a display board. With its simple yet powerful design, users can input messages from a mobile app, and the message is instantly displayed on an LED matrix display. This means that the board can be located anywhere in the world, and users can still send messages to it without any limitations.

The IOT Notice Board has numerous potential applications in various industries, including transportation, healthcare, education, and hospitality. In transportation, the board can be used to provide real-time updates on schedules, delays, and cancellations. In healthcare, it can be used to provide critical information to patients and their families, such as appointment reminders and medical updates. In education, it can be used to display class schedules, upcoming events, and important announcements. In hospitality, it can be used to display menu items, promotions, and specials.

Overall, the IOT Notice Board is a powerful tool for real-time communication that has the potential to revolutionize the way we share information in various industries. Its ability to take inputs from anywhere across the globe makes it a highly flexible and accessible communication system that can be tailored to meet the specific needs of any organization.

1.3 Applications

An IoT Based Notice Board can be used for a variety of applications, including:

1. **Public Information Display:** The IoT Notice Board can be used to display public information such as bus schedules, flight times, and weather updates. This information can be updated in real-time and displayed on the board for easy viewing by the public.

2. **Advertising:** The IoT Notice Board can be used to display advertisements in public places such as malls, airports, and train stations. Advertisers can update the content remotely and display targeted ads to specific audiences.

3. **Emergency Alerts:** The IoT Notice Board can be used to display emergency alerts such as weather warnings, evacuation notices, and other critical information. This can help to keep people safe and informed during emergencies.

4. **Education:** The IoT Notice Board can be used in classrooms to display important announcements, homework assignments, and other educational content. Teachers can update the content remotely and ensure that all students have access to the same information.

5. **Event Information:** The IoT Notice Board can be used at events to display schedules, directions, and other important information for attendees. This can help to reduce confusion and ensure that attendees have a smooth and enjoyable experience.

   Overall, the IoT Notice Board is a versatile device that can be used in a variety of settings to display important information. Its ability to take inputs from anywhere across the globe makes it an ideal solution for organizations and businesses looking for a flexible and cost-effective way to communicate with their audience.

2 Components and Description

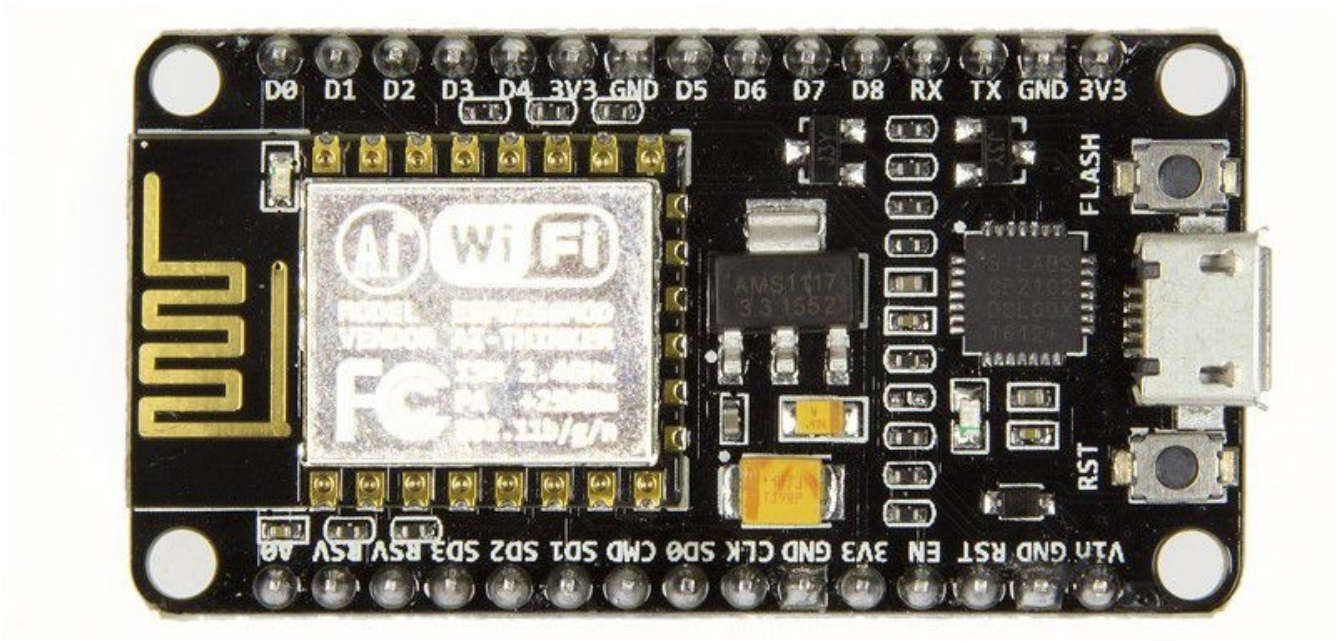
2.1 ESP 8266

The ESP8266 is a highly popular and cost-effective Wi-Fi module that has gained popularity among developers and hobbyists alike. It is a low-power system-on-a-chip (SoC) with integrated Wi-Fi capabilities, making it an excellent choice for Internet of Things (IoT) projects.

The ESP8266 offers a range of features that make it stand out from other Wi-Fi modules, including a built-in TCP/IP protocol stack, support for 802.11 b/g/n Wi-Fi standards, and a range of GPIO pins for connecting sensors, actuators, and other peripherals. It also features a variety of low-power modes, making it ideal for battery-powered applications.

Developers can program the ESP8266 using a variety of programming languages, including Lua, Python, and C++, using various software development kits (SDKs) and integrated development environments (IDEs). Additionally, the ESP8266 supports over-the-air (OTA) firmware updates, enabling developers to update their code remotely without the need for physical access to the device.

Overall, the ESP8266 is a versatile and affordable Wi-Fi module that offers developers and hobbyists an easy way to add Wi-Fi connectivity to their IoT projects. Its ease of use and vast community support have made it a go-to choice for many IoT enthusiasts.



ESP 8266

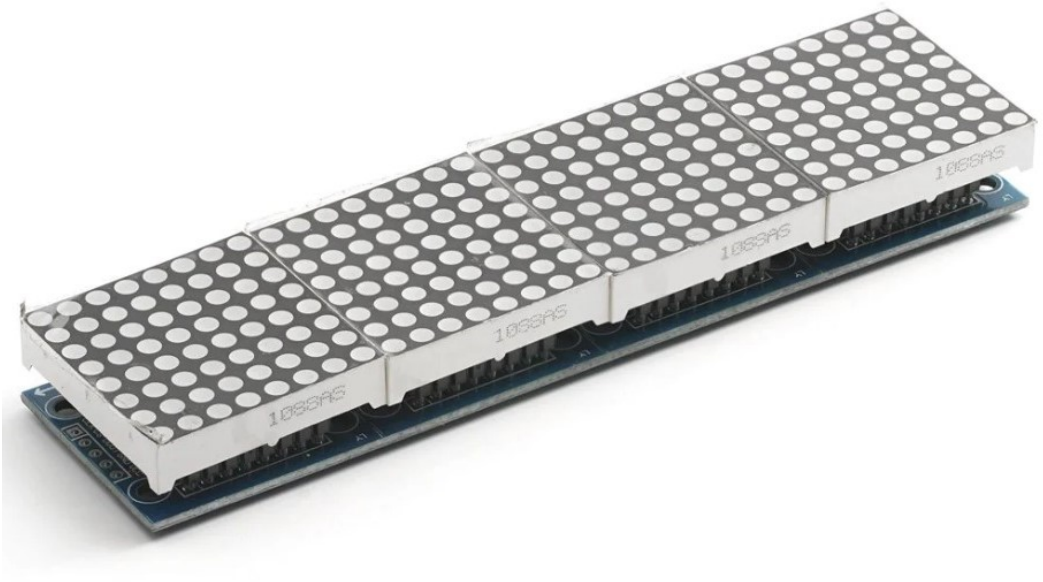
2.2 MAX7219 LED Matrix Display

The MAX7219 LED Matrix Display is a versatile and widely used integrated circuit (IC) driver that is commonly used to control LED matrices of various sizes. It is designed to interface with microcontrollers and provides an easy way to control multiple LEDs with minimal hardware requirements.

The MAX7219 driver IC can drive up to 64 individual LEDs or eight 7-segment LED displays, making it an excellent choice for a variety of applications. It features a serial interface that allows for easy communication with a microcontroller, and the IC is also designed to provide efficient power consumption, making it ideal for battery-powered devices.

One of the key advantages of the MAX7219 is its ability to cascade multiple ICs together, allowing for the control of large LED matrix displays. Each IC can control up to 64 LEDs or eight 7-segment LED displays, and up to eight ICs can be connected in series to control up to 512 LEDs or 64 7-segment displays.

The MAX7219 is a versatile and reliable driver IC that has been widely adopted in the industry due to its ease of use, low power consumption, and the ability to control large LED matrix displays. It is an excellent choice for a variety of applications, including scoreboards, digital clocks, and other display projects that require multiple LEDs to be controlled.



MAX7219 LED Display

2.3 5V Step- Up Power Module with Charging

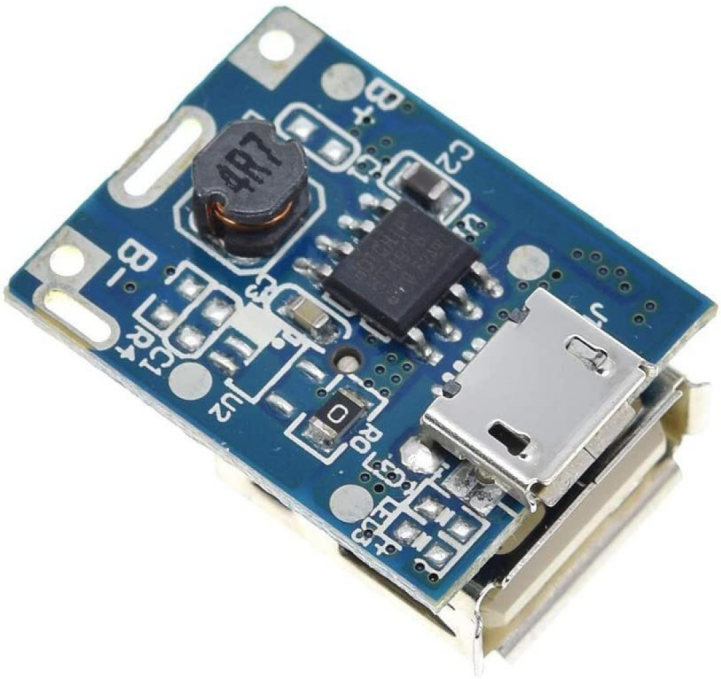
The 5V Step Up Power Module with Charging is a small and compact electronic device that is designed to convert low-voltage DC power sources into higher-voltage outputs. It is commonly used in a variety of electronics projects, including portable devices such as smartphones, tablets, and other battery-powered devices.

The power module has two primary functions: it steps up the input voltage to a higher output voltage, and it includes a charging circuit to charge a connected battery. It operates by taking in a low voltage DC input, typically 3.7V or 5V, and boosts it to a higher voltage output of up to 5V or higher. This higher voltage output can then be used to power other electronic devices or charge a connected battery.

The 5V Step Up Power Module with Charging includes various safety features, such as over-voltage and over-current protection, which ensures that the device and any connected batteries are protected from damage due to excessive voltage or current. It is also compact and easy to install, making it a popular choice for hobbyists and electronics enthusiasts.

Overall, the 5V Step Up Power Module with Charging is a versatile and reliable device that provides a convenient solution for powering or charging electronic devices using low-voltage DC sources.

Its compact size and safety features make it an excellent choice for a wide range of applications, including portable devices and other battery-powered projects.



5V Step- Up Power Module with Charging

2.4 Lithium Polymer Battery

The 1000 mAh Lithium Polymer battery is a rechargeable battery that is commonly used in small portable electronic devices, such as Bluetooth headphones, wireless speakers, and remote-controlled toys. It is a lightweight and compact battery that offers a high energy density, making it an ideal choice for devices that require a long battery life.

Lithium Polymer batteries are known for their high energy density, which means that they can store a large amount of energy in a small package. The 1000 mAh Lithium Polymer battery is no exception, offering a high capacity for its small size. This makes it an excellent choice for devices that require a high amount of power in a small space.

The 1000 mAh Lithium Polymer battery also features a low self-discharge rate, which means that it can retain its charge for extended periods without being used. Additionally, it is a rechargeable battery, which means that it can be used repeatedly and recharged using a compatible charger.

Safety is also a significant consideration when using Lithium Polymer batteries. The 1000 mAh Lithium Polymer battery includes various safety features, such as over-voltage and over-current protection, which ensure that the battery is protected from damage due to excessive voltage or current. It is also designed to prevent short-circuits, which can be a safety hazard.

Overall, the 1000 mAh Lithium Polymer battery is a reliable and high-performance battery that is ideal for a range of small portable electronic devices. Its high energy density, low self-discharge rate, and safety features make it an excellent choice for powering devices that require a long battery life in a small package.

Lithium Polymer Battery

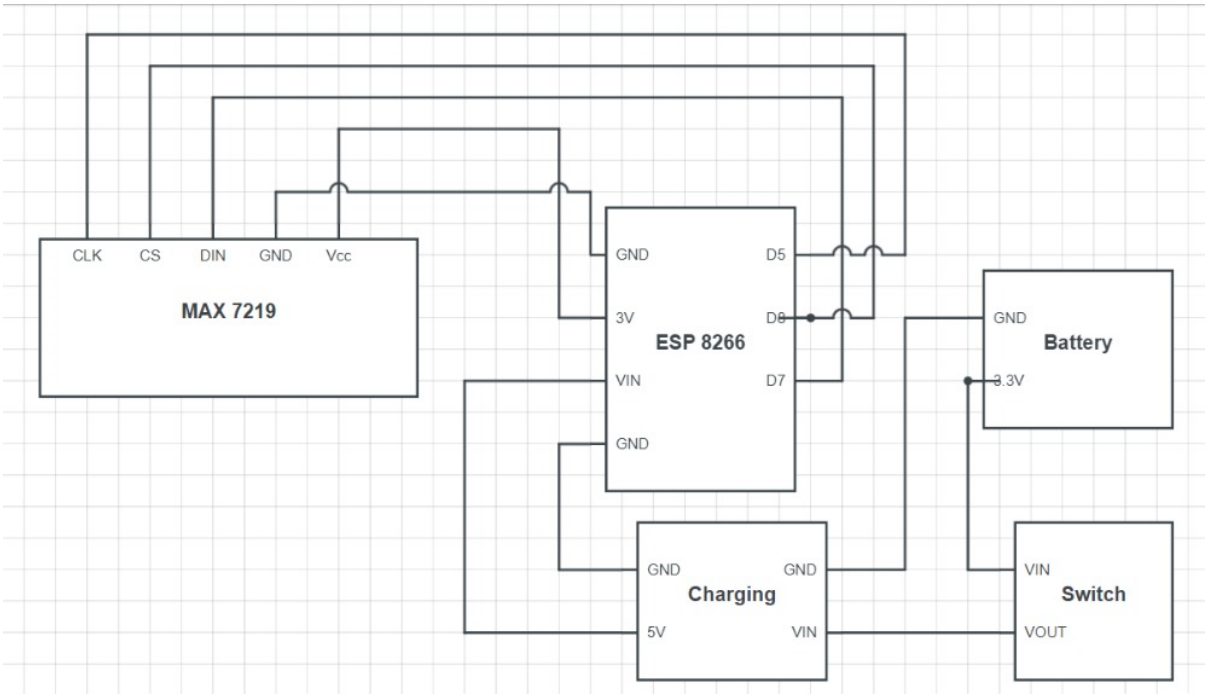
2.5 Jumper Wires

Jumper wires are small insulated wires that are used to connect components on a breadboard or other electronic prototype boards. They are typically made of flexible stranded wire with pins or sockets at each end, allowing them to be easily inserted into the holes of a breadboard or connected to other electronic components.

Jumper wires come in various lengths, colors, and styles, including male-to-male, female-to-female, and male-to-female. They are a fundamental component in electronics prototyping and allow for easy and quick connections between different components.

Jumper Wires

3 Circuit Diagram



Circuit Diagram

4 ESP 8266 Code

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/409cac7c-e943-4e62-b010-1851e4dbb069

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to th

  String data;
  String ip_address;
  int brightness;
  int speed;
  bool color;
  bool direction;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Turn on debug statements to the serial output
```

```
#define  DEBUG  0

#if  DEBUG
#define PRINT(s, x) { Serial.print(F(s)); Serial.print(x); }
#define PRINTS(x) Serial.print(F(x))
#define PRINTX(x) Serial.println(x, HEX)
#else
#define PRINT(s, x)
#define PRINTS(x)
#define PRINTX(x)
#endif

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 8
#define CS_PIN     15 // or SS

// HARDWARE SPI
MD_Parola P = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// WiFi login parameters - network name and password
const char* ssid = "Vineet";
const char* password = "18399770";

// WiFi Server object and param eters
WiFiServer server(80);

// Scrolling parameters
uint8_t frameDelay = 25;  // default frame delay value
textEffect_t  scrollEffect = PA_SCROLL_LEFT;

// Global message buffers shared by Wifi and Scrolling functions
#define BUF_SIZE  512
char curMessage[BUF_SIZE];
char newMessage[BUF_SIZE];
bool newMessageAvailable = false;

const char WebResponse[] = "HTTP/1.1 200 OK\nContent-Type: text/html\n\n";

const char WebPage[] =
"<!DOCTYPE html>" \
"<html>" \
"<head>" \
"<title>MajicDesigns Test Page</title>" \

"<script>" \
"strLine = \"\";" \

"function SendData()" \
"{" \
"  nocache = \"/&nocache=\" + Math.random() * 1000000;" \
"  var request = new XMLHttpRequest();" \
"  strLine = \"&MSG=\" + document.getElementById(\"data_form\").Message.value;" \
"  strLine = strLine + \"/&SD=\" + document.getElementById(\"data_form\").ScrollType.value;"
"  strLine = strLine + \"/&I=\" + document.getElementById(\"data_form\").Invert.value;" \
"  strLine = strLine + \"/&SP=\" + document.getElementById(\"data_form\").Speed.value;" \
"  request.open(\"GET\", strLine + nocache, false);" \
"  request.send(null);" \
"}" \
```

```
"</script>" \
"</head>" \

"<body>" \
"<p><b>Smart Notice Board</b></p>" \

"<form id=\"data_form\" name=\"frmText\">" \
"<label>Message:<br><input type=\"text\" name=\"Message\" maxlength=\"255\"></label>" \
"<br><br>" \
"<input type = \"radio\" name = \"Invert\" value = \"0\" checked> Normal" \
"<input type = \"radio\" name = \"Invert\" value = \"1\"> Inverse" \
"<br>" \
"<input type = \"radio\" name = \"ScrollType\" value = \"L\" checked> Left Scroll" \
"<input type = \"radio\" name = \"ScrollType\" value = \"R\"> Right Scroll" \
"<br><br>" \
"<label>Speed:<br>Fast<input type=\"range\" name=\"Speed\" min=\"10\" max=\"200\">Slow"\
"<br>" \
"</form>" \
"<br>" \
"<input type=\"submit\" value=\"Send Data\" onclick=\"SendData()\">" \
"</body>" \
"</html>";

const char *err2Str(wl_status_t code)
{
  switch (code)
  {
  case WL_IDLE_STATUS:     return("IDLE");            break; // WiFi is in process of changing
  case WL_NO_SSID_AVAIL:   return("NO_SSID_AVAIL");   break; // case configured SSID cannot be
  case WL_CONNECTED:       return("CONNECTED");       break; // successful connection is estab
  case WL_CONNECT_FAILED:  return("CONNECT_FAILED");  break; // password is incorrect
  case WL_DISCONNECTED:    return("CONNECT_FAILED");  break; // module is not configured in st
  default: return("??");
  }
}

uint8_t htoi(char c)
{
  c = toupper(c);
  if ((c >= '0') && (c <= '9')) return(c - '0');
  if ((c >= 'A') && (c <= 'F')) return(c - 'A' + 0xa);
  return(0);
}

void getData(char *szMesg, uint16_t len)
// Message may contain data for:
// New text (/&MSG=)
// Scroll direction (/&SD=)
// Invert (/&I=)
// Speed (/&SP=)
{
  char *pStart, *pEnd;       // pointer to start and end of text

  // check text message
  pStart = strstr(szMesg, "/&MSG=");
  if (pStart != NULL)
  {
    char *psz = newMessage;
```

```
  pStart += 6;  // skip to start of data
  pEnd = strstr(pStart, "/&");

  if (pEnd != NULL)
  {
    while (pStart != pEnd)
    {
      if ((*pStart == '%') && isxdigit(*(pStart + 1)))
      {
        // replace %xx hex code with the ASCII character
        char c = 0;
        pStart++;
        c += (htoi(*pStart++) << 4);
        c += htoi(*pStart++);
        *psz++ = c;
      }
      else
        *psz++ = *pStart++;
    }

    *psz = '\0'; // terminate the string
    newMessageAvailable = (strlen(newMessage) != 0);
    PRINT("\nNew Msg: ", newMessage);
  }
}

// check scroll direction
pStart = strstr(szMesg, "/&SD=");
if (pStart != NULL)
{
  pStart += 5;  // skip to start of data

  PRINT("\nScroll direction: ", *pStart);
  scrollEffect = (*pStart == 'R' ? PA_SCROLL_RIGHT : PA_SCROLL_LEFT);
  P.setTextEffect(scrollEffect, scrollEffect);
  P.displayReset();
}

// check invert
pStart = strstr(szMesg, "/&I=");
if (pStart != NULL)
{
  pStart += 4;  // skip to start of data

  PRINT("\nInvert mode: ", *pStart);
  P.setInvert(*pStart == '1');
}

// check speed
pStart = strstr(szMesg, "/&SP=");
if (pStart != NULL)
{
  pStart += 5;  // skip to start of data

  int16_t speed = atoi(pStart);
  PRINT("\nSpeed: ", P.getSpeed());
  P.setSpeed(speed);
```

```
    frameDelay = speed;
  }
}



void handleWiFi(void)
{
  static enum { S_IDLE, S_WAIT_CONN, S_READ, S_EXTRACT, S_RESPONSE, S_DISCONN } state = S_IDL
  static char szBuf[1024];
  static uint16_t idxBuf = 0;
  static WiFiClient client;
  static uint32_t timeStart;

  switch (state)
  {
  case S_IDLE:   // initialise
    PRINTS("\nS_IDLE");
    idxBuf = 0;
    state = S_WAIT_CONN;
    break;

  case S_WAIT_CONN:   // waiting for connection
  {
    client = server.available();
    if (!client) break;
    if (!client.connected()) break;

#if DEBUG
    char szTxt[20];
    sprintf(szTxt, "%03d:%03d:%03d:%03d", client.remoteIP()[0], client.remoteIP()[1], client.
    PRINT("\nNew client @ ", szTxt);
#endif

    timeStart = millis();
    state = S_READ;
  }
  break;

  case S_READ: // get the first line of data
    PRINTS("\nS_READ ");

    while (client.available())
    {
      char c = client.read();

      if ((c == '\r') || (c == '\n'))
      {
        szBuf[idxBuf] = '\0';
        client.flush();
        PRINT("\nRecv: ", szBuf);
        state = S_EXTRACT;
      }
      else
        szBuf[idxBuf++] = (char)c;
    }
    if (millis() - timeStart > 1000)
    {
```

```
      PRINTS("\nWait timeout");
      state = S_DISCONN;
    }
    break;

  case S_EXTRACT: // extract data
    PRINTS("\nS_EXTRACT");
    // Extract the string from the message if there is one
    getData(szBuf, BUF_SIZE);
    state = S_RESPONSE;
    break;

  case S_RESPONSE: // send the response to the client
    PRINTS("\nS_RESPONSE");
    // Return the response to the client (web page)
    client.print(WebResponse);
    client.print(WebPage);
    state = S_DISCONN;
    break;

  case S_DISCONN: // disconnect client
    PRINTS("\nS_DISCONN");
    client.flush();
    client.stop();
    state = S_IDLE;
    break;

  default:  state = S_IDLE;
  }
}

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is fou
  delay(1500);


  PRINTS("\n[MD_Parola WiFi Message Display]\nType a message for the scrolling display from y

  P.begin();
  P.setIntensity(15);
  P.displayClear();
  P.displaySuspend(false);

  P.displayScroll(curMessage, PA_LEFT, scrollEffect, frameDelay);

  curMessage[0] = newMessage[0] = '\0';

  // while (WiFi.status() != WL_CONNECTED)
  // {
  //   PRINT("\n", err2Str(WiFi.status()));
  //   sprintf(curMessage, "Connecing...");
  //   delay(500);
  // }
  PRINTS("\nWiFi connected");

  // Start the server
```

```
  server.begin();
  PRINTS("\nServer started");

  // Set up first message as the IP address
  sprintf(curMessage, "IoT Notice Board");
  // sprintf(curMessage, "%03d:%03d:%03d:%03d", WiFi.localIP()[0], WiFi.localIP()[1], WiFi.lo
  PRINT("\nAssigned IP ", curMessage);



  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here

  handleWiFi();

  if (P.displayAnimate())
  {
    if (newMessageAvailable)
    {
      strcpy(curMessage, data.c_str());
      newMessageAvailable = false;
    }
    P.displayReset();
  }
}

void onDataChange()
{
    strcpy(curMessage, data.c_str());
    P.displayReset();
}

void onSpeedChange()
{
    P.setSpeed((15-speed) * 5);
}

void onDirectionChange()
{
    scrollEffect = (direction ? PA_SCROLL_RIGHT : PA_SCROLL_LEFT);
```

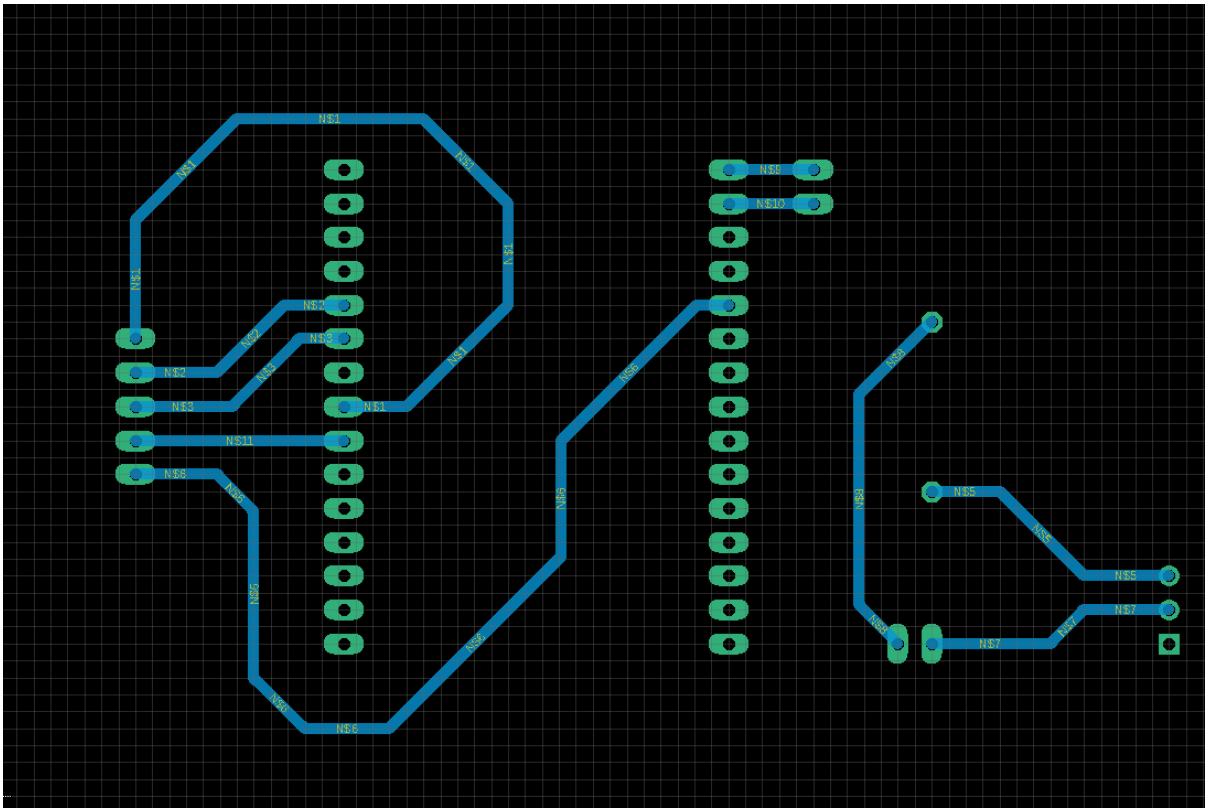
```
    P.setTextEffect(scrollEffect, scrollEffect);
}

void onBrightnessChange()
{
    P.setIntensity(brightness);
}

void onColorChange()
{
    P.setInvert(color);
}
```

5 thingProperties.h Code

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char DEVICE_LOGIN_NAME[] = "6ea2420a-1404-48ca-a1fa-f29dc4af2bd9";

const char SSID[] = SECRET_SSID;     // Network SSID (name)
const char PASS[] = SECRET_OPTIONAL_PASS;     // Network password
const char DEVICE_KEY[] = SECRET_DEVICE_KEY;     // Secret device password

void onDataChange();
void onBrightnessChange();
void onSpeedChange();
void onColorChange();
void onDirectionChange();

String data;
String ip_address;
int brightness;
int speed;
bool color;
bool direction;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(data, READWRITE, ON_CHANGE, onDataChange);
  ArduinoCloud.addProperty(ip_address, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(brightness, READWRITE, ON_CHANGE, onBrightnessChange);
  ArduinoCloud.addProperty(speed, READWRITE, ON_CHANGE, onSpeedChange);
  ArduinoCloud.addProperty(color, READWRITE, ON_CHANGE, onColorChange);
  ArduinoCloud.addProperty(direction, READWRITE, ON_CHANGE, onDirectionChange);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

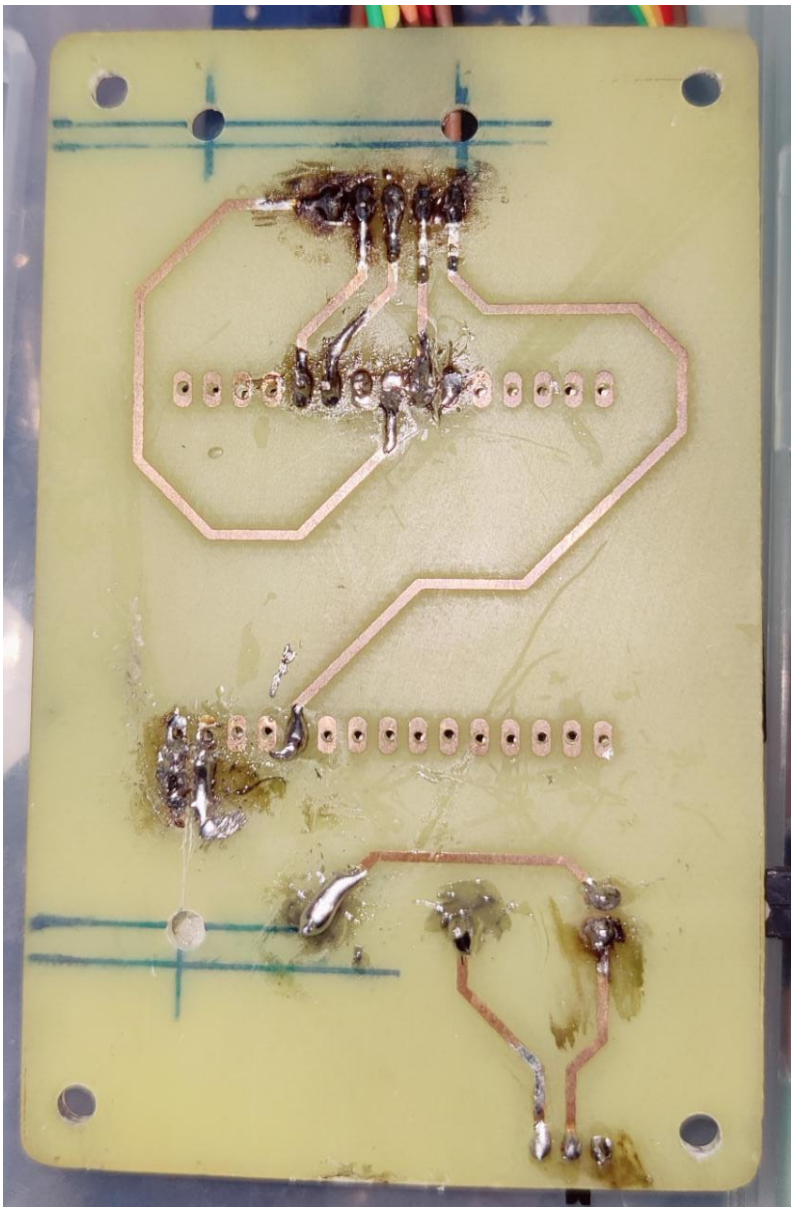
6 Setup and Fabrication

6.1 PCB Fabrication



PCB layout

6.2 Final PCB



PCB

6.3 Final Product

Final Product

7 References

1. https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/
2. https://microcontrollerslab.com/max7219-led-dot-matrix-display-esp8266-nodemcu-tutorial/
3. https://stackoverflow.com
4. https://io.adafruit.com/
5. https://www.arduino.cc/en/software
6. https://www.python.org/doc/