**Technical Building Block**

**Display Unit**

**ID:TBB**

1. **Purpose**

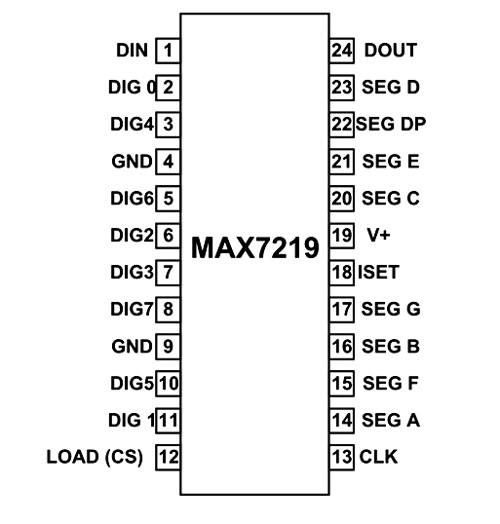
To enable displaying data remotely on multiple displays by using ESP-8266 and max7219 seven-segment display unit.

1. **Part Number, Datasheet and Pinouts:**

Part #: ESP8266, HW-179(max 7219 seven segment display)

Datasheet:

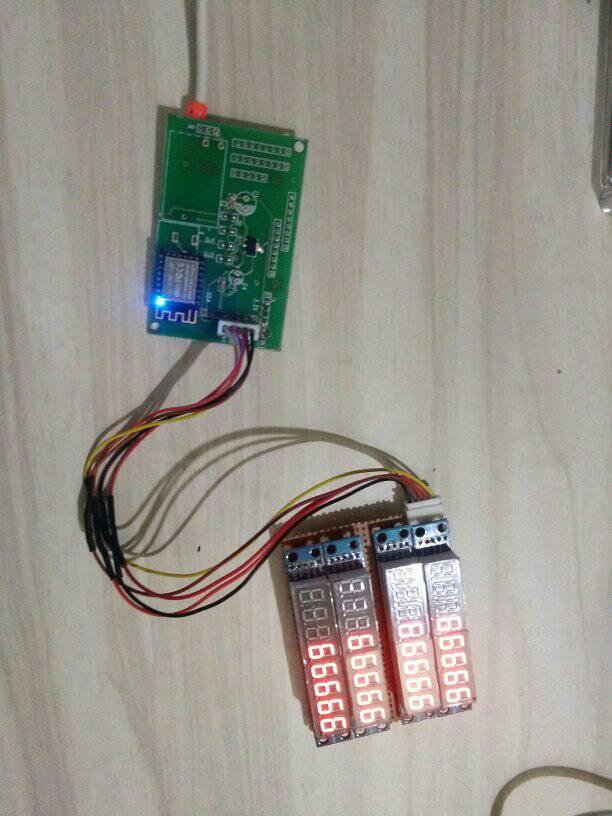
<https://pdf1.alldatasheet.com/datasheet-pdf/view/227793/MAXIM/MAX7219.html?fbclid=IwAR2k2wApzCrapIKfM6NEn8kVh8W-kdUq6iESJqRaJzZSIFIp7fqqAcDCr1s>



1. **Rated Specifications :**

|  |  |
| --- | --- |
| **Variant** | **Operating Value** |
| Operating Voltage Range | 4 V- 5.5V |
| Max Supply Voltage | 6V |
| Max current i/p by segment pins | 100mA |
| Max current i/p through ground pin | 500mA |
| Data-Segment Delay | 2.2mSec |
| Operating Temperature | 0 – 70 ‘C |

1. **Constraints**

Provide source (i.e. Vcc and Gnd) separately (parallelly) to each Display unit, else the intensity will keep on diminishing with every subsequent connection.

1. **Assembly Pic**
2. **Arduino Code:**

/\*  
Basic ESP8266 MQTT example  
  
This sketch demonstrates the capabilities of the pubsub library in combination  
with the ESP8266 board/library.  
  
It connects to an MQTT server then:  
- publishes "hello world" to the topic "outTopic" every two seconds  
- subscribes to the topic "inTopic", printing out any messages  
it receives. NB - it assumes the received payloads are strings not binary  
- If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,  
else switch it off  
  
It will reconnect to the server if the connection is lost using a blocking  
reconnect function. See the 'mqtt\_reconnect\_nonblocking' example for how to  
achieve the same result without blocking the main loop.  
  
To install the ESP8266 board, (using Arduino 1.6.4+):  
- Add the following 3rd party board manager under "File -> Preferences -> Additional Boards Manager URLs":  
[http://arduino.esp8266.com/stable/package\_esp8266com\_index.json](https://l.facebook.com/l.php?u=http%3A%2F%2Farduino.esp8266.com%2Fstable%2Fpackage_esp8266com_index.json%3Ffbclid%3DIwAR1aS9LT0Kj-8jZ7frSWGW2g1bUS730XT9BoaHGx1vbhoXhU-f08xDS-MtU&h=AT38GvdcGnSkcqTKOYkcUo33Zcwh0hFjJWcxaeTr8BsdnJtKM7L1ds9pbglJwpZL2u3Som1J_xyVr5fPRBMlJFrEzuhufC0qvr96LxjITaMpQ9HZMWH-kILm7BiGey76J423Ow)  
- Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"  
- Select your ESP8266 in "Tools -> Board"  
  
\*/  
#include "LedControl.h"  
LedControl lc=LedControl(13,14,4,4);  
  
#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
  
// Update these with values suitable for your network.  
unsigned long delaytime=250;  
const char\* ssid = "iot-coppercloud";  
const char\* password = "coppercloud";  
const char\* mqtt\_server = "[3.214.158.175](https://l.facebook.com/l.php?u=http%3A%2F%2F3.214.158.175%2F%3Ffbclid%3DIwAR1qliFi0pqt_jr0Sp0Q2WWAYyxeQUjQsr-tAXK4Cvi2znoTzsGOqr925LI&h=AT38GvdcGnSkcqTKOYkcUo33Zcwh0hFjJWcxaeTr8BsdnJtKM7L1ds9pbglJwpZL2u3Som1J_xyVr5fPRBMlJFrEzuhufC0qvr96LxjITaMpQ9HZMWH-kILm7BiGey76J423Ow)";  
  
WiFiClient espClient;  
PubSubClient client(espClient);  
long lastMsg = 0;  
char msg[50];  
int value = 0;  
  
void setup\_wifi() {  
  
delay(10);  
// We start by connecting to a WiFi network  
Serial.println();  
Serial.print("Connecting to ");  
Serial.println(ssid);  
  
WiFi.begin(ssid, password);  
  
while (WiFi.status() != WL\_CONNECTED) {  
delay(500);  
Serial.print(".");  
}  
  
randomSeed(micros());  
  
Serial.println("");  
Serial.println("WiFi connected");  
Serial.println("IP address: ");  
Serial.println(WiFi.localIP());  
}  
  
void callback(char\* topic, byte\* payload, unsigned int length) {  
Serial.print("Message arrived [");  
Serial.print(topic);  
Serial.print("] ");  
for (int i = 0; i < length; i++) {  
Serial.print((byte)payload[0]);  
}  
Serial.println();  
  
// Switch on the LED if an 1 was received as first character  
for(int i=0;i<8;i++) {  
lc.setDigit(0,i,(byte)payload[0],false);  
lc.setDigit(1,i,(byte)payload[0],false);  
lc.setDigit(2,i,(byte)payload[0],false);  
lc.setDigit(3,i,(byte)payload[0],false);  
delay(delaytime);  
}  
lc.clearDisplay(0);  
lc.clearDisplay(1);  
lc.clearDisplay(2);  
lc.clearDisplay(3);  
//delay(delaytime);  
  
}  
  
void reconnect() {  
// Loop until we're reconnected  
while (!client.connected()) {  
Serial.print("Attempting MQTT connection...");  
// Create a random client ID  
String clientId = "ESP8266Client-";  
clientId += String(random(0xffff), HEX);  
// Attempt to connect  
if (client.connect(clientId.c\_str())) {  
Serial.println("connected");  
// Once connected, publish an announcement...  
// client.publish("ledcontrol", "hello world");  
// ... and resubscribe  
client.subscribe("ledcontrol");  
} else {  
Serial.print("failed, rc=");  
Serial.print(client.state());  
Serial.println(" try again in 5 seconds");  
// Wait 5 seconds before retrying  
delay(5000);  
}  
}  
}  
  
void setup() {  
pinMode(BUILTIN\_LED, OUTPUT); // Initialize the BUILTIN\_LED pin as an output  
Serial.begin(115200);  
setup\_wifi();  
client.setServer(mqtt\_server, 1883);  
client.setCallback(callback);  
lc.shutdown(0,false);  
lc.shutdown(1,false);  
lc.shutdown(2,false);  
lc.shutdown(3,false);  
/\* Set the brightness to a medium values \*/  
lc.setIntensity(0,8);  
lc.setIntensity(1,8);  
lc.setIntensity(2,8);  
lc.setIntensity(3,8);  
/\* and clear the display \*/  
lc.clearDisplay(0);  
lc.clearDisplay(1);  
lc.clearDisplay(2);  
lc.clearDisplay(3);  
}  
  
void loop() {  
  
if (!client.connected()) {  
reconnect();  
}  
client.loop();  
  
long now = millis();  
if (now - lastMsg > 2000) {  
lastMsg = now;  
++value;  
//snprintf (msg, 50, "hello world #%ld", value);  
Serial.print("Publish message: ");  
Serial.println(msg);  
//client.publish("outTopic", msg);  
}  
}

1. **Libraries**

LedControl.h

PubSubClient.h

ESP8266WiFi.h

1. **Other Helpful URLs**

<https://wayoda.github.io/LedControl/pages/software?fbclid=IwAR0_JLH9yfhzYdfGUNMEqHRk3QoJ_SN0POeRBGnFZ4s_laMSHAfY-piVO-M>

1. **Additional Notes**

Use set.digit() function in LedControl library in order to send data to the Display.