```
/*
 * Google Assistant Controlled 8X48 LEDScrolling
*Done by Chanukya N
*YouTube:https://youtube.com/c/chanukyaGowda
 */
/************************* Necessary Libraries *********************************/
//#include <MD_MAX72xx.h>
//#include <SPI.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

//#define SCROLL_DELAY 75

/************************* Variables *********************************/
char* str;
String payload;
uint32_t present;
bool first_time;
uint16_t  scrollDelay;  // in milliseconds
#define  CHAR_SPACING  1 // pixels between characters

// Global message buffers shared by Serial and Scrolling functions
```

```
#define BUF_SIZE  75
char curMessage[BUF_SIZE];
char newMessage[BUF_SIZE];
bool newMessageAvailable = false;

ESP8266WiFiMulti WiFiMulti;

// Define the number of devices we have in the chain and the hardware interface
// NOTE: These pin numbers will probably not work with your hardware and may
// need to be adapted
#define  MAX_DEVICES 8

/************************* Matrix Display Pins **********************************/
//#define  CLK_PIN   D5 // or SCK
//#define DATA_PIN  D7 // or MOSI
//#define CS_PIN    D8 // or SS

/************************* Adafruit.io Setup ************************************/
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "chanukya"
#define AIO_KEY         "paste your adafruit key"

/************ Global State (you don't need to change this!) ******************/
```

```
// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// or... use WiFiFlientSecure for SSL

//WiFiClientSecure client;


// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);


/************************************ Feeds ***************************************/


// Setup a feed called 'onoff' for subscribing to changes.

Adafruit_MQTT_Subscribe message = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/feeds/message");


/************************** Sketch Code ************************************/


// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT_connect();


// SPI hardware interface

//MD_MAX72XX  mx = MD_MAX72XX(CS_PIN,MAX_DEVICES);

// Arbitrary pins

//MD_MAX72XX mx = MD_MAX72XX(DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);
```

```
//uint8_t scrollDataSource(uint8_t dev, MD_MAX72XX::transformType_t t)
// Callback function for data that is required for scrolling into the display
/*{
  static char   *p = curMessage;
  static uint8_t  state = 0;
  static uint8_t  curLen, showLen;
  static uint8_t  cBuf[8];
  uint8_t colData;

  // finite state machine to control what we do on the callback
  switch (state)
  {
    case 0: // Load the next character from the font table
      showLen = mx.getChar(*p++, sizeof(cBuf) / sizeof(cBuf[0]), cBuf);
      curLen = 0;
      state++;

      // if we reached end of message, reset the message pointer
      if (*p == '\0')
      {
```

```
    p = curMessage;     // reset the pointer to start of message
    if (newMessageAvailable)  // there is a new message waiting
    {
      strcpy(curMessage, str);  // copy it in
      newMessageAvailable = false;
    }
  }
// !! deliberately fall through to next state to start displaying

case 1: // display the next part of the character
  colData = cBuf[curLen++];
  if (curLen == showLen)
  {
    showLen = CHAR_SPACING;
    curLen = 0;
    state = 2;
  }
  break;

case 2: // display inter-character spacing (blank column)
  colData = 0;
  curLen++;
  if (curLen == showLen)
    state = 0;
  break;
```

```
    default:
      state = 0;
  }

  return (colData);
}

void scrollText(void)
{
  static uint32_t prevTime = 0;

  // Is it time to scroll the text?
  if (millis() - prevTime >= scrollDelay)
  {
    mx.transform(MD_MAX72XX::TSL);  // scroll along - the callback will load all the data
    prevTime = millis();      // starting point for next time
  }
}
void  no_connection(void)
{
  newMessageAvailable = 1;
  strcpy(curMessage, "No Internet! ");
  scrollText();
```

```
}*/
String msg1;
char msg[]="cstech";//Change the text here.
int scrollspeed=5;//Set the scroll speed ( lower=faster)

int x;
int y;

//Columns
//int clockPin1 = 2; //Arduino pin connected to Clock Pin 11 of 74HC595
//int latchPin1 = 3; //Arduino pin connected to Latch Pin 12 of 74HC595
//int dataPin1 = 4;  //Arduino pin connected to Data Pin 14 of 74HC595
//
////Rows
//int clockPin2 = 5; //Arduino pin connected to Clock Pin 11 of 74HC595
//int latchPin2 = 6; //Arduino pin connected to Latch Pin 12 of 74HC595
//int dataPin2 = 7;  //Arduino pin connected to Data Pin 14 of 74HC595

//BITMAP
//Bits in this array represents one LED of the matrix
// 8 is # of rows, 6 is # of LED matrices
byte bitmap[8][7];

int numZones = sizeof(bitmap) / 8; // One Zone refers to one 8 x 8 Matrix ( Group of 8 columns)
```

```
int maxZoneIndex = numZones-1;
int numCols = numZones * 8;

//FONT DEFENITION
byte alphabets[][8] = {
 {0,0,0,0,0},//@ as SPACE
 //{8,28,54,99,65},//<<
 {31, 36, 68, 36, 31},//A
 {127, 73, 73, 73, 54},//B
 {62, 65, 65, 65, 34},//C
 {127, 65, 65, 34, 28},//D
 {127, 73, 73, 65, 65},//E
 {127, 72, 72, 72, 64},//F
 {62, 65, 65, 69, 38},//G
 {127, 8, 8, 8, 127},//H
 {0, 65, 127, 65, 0},//I
 {2, 1, 1, 1, 126},//J
 {127, 8, 20, 34, 65},//K
 {127, 1, 1, 1, 1},//L
 {127, 32, 16, 32, 127},//M
 {127, 32, 16, 8, 127},//N
 {62, 65, 65, 65, 62},//O
 {127, 72, 72, 72, 48},//P
 {62, 65, 69, 66, 61},//Q
 {127, 72, 76, 74, 49},//R
```

```
 {50, 73, 73, 73, 38},//S
 {64, 64, 127, 64, 64},//T
 {126, 1, 1, 1, 126},//U
 {124, 2, 1, 2, 124},//V
 {126, 1, 6, 1, 126},//W
 {99, 20, 8, 20, 99},//X
 {96, 16, 15, 16, 96},//Y
 {67, 69, 73, 81, 97},//Z
};
uint8_t latchPin1 = D3;
uint8_t clockPin1 = D2;
uint8_t dataPin1 = D4;
uint8_t latchPin2 = D6;
uint8_t clockPin2 = D5;
uint8_t dataPin2= D7;
void RefreshDisplay()
{
 for (int row = 0; row < 8; row++) {
   int rowbit = 1 << row;
   digitalWrite(latchPin2, LOW);//Hold latchPin LOW for transmitting data
   shiftOut(dataPin2, clockPin2, MSBFIRST, rowbit);   //Transmit data

   //Start sending column bytes
   digitalWrite(latchPin1, LOW);//Hold latchPin LOW for transmitting data
```

```
  //Shift out to each matrix
  for (int zone = maxZoneIndex; zone >= 0; zone--)
  {
    shiftOut(dataPin1, clockPin1, MSBFIRST, bitmap[row][zone]);
  }

  //flip both latches at once to eliminate flicker
  digitalWrite(latchPin1, HIGH);//Return the latch pin 1 high to signal chip
  digitalWrite(latchPin2, HIGH);//Return the latch pin 2 high to signal chip

  //Wait
  delayMicroseconds(300);
 }
}

// Converts row and colum to bitmap bit and turn it off/on
void Plot(int col, int row, bool isOn)
{
 int zone = col / 8;
 int colBitIndex = x % 8;
 byte colBit = 1 << colBitIndex;
 if (isOn)
  bitmap[row][zone] =  bitmap[y][zone] | colBit;
 else
  bitmap[row][zone] =  bitmap[y][zone] & (~colBit);
```

```
}

// Plot each character of the message one column at a time, updated the display, shift bitmap left.

void XProcess()

{

  //Serial.println(msg1);

  msg1.toUpperCase();

 for (int charIndex=0; charIndex < (sizeof(msg1)-1); charIndex++)

 {

  int alphabetIndex = msg1[charIndex] - '@';

  if (alphabetIndex < 0) alphabetIndex=0;


  //Draw one character of the message

  // Each character is 5 columns wide, loop two more times to create 2 pixel space betwen characters

  for (int col = 0; col < 7; col++)

  {

    for (int row = 0; row < 8; row++)

    {

      // Set the pixel to what the alphabet say for columns 0 thru 4, but always leave columns 5 and 6 blank.

      bool isOn = 0;

      if (col<5) isOn = bitRead( alphabets[alphabetIndex][col], 7-row ) == 1;

      Plot( numCols-1, row, isOn); //Draw on the rightmost column, the shift loop below will scroll it leftward.

    }

    for (int refreshCount=0; refreshCount < scrollspeed; refreshCount++)
```

```
      RefreshDisplay();
    //Shift the bitmap one column to left
    for (int row=0; row<8; row++)
    {
      for (int zone=0; zone < numZones; zone++)
      {
        //This right shift would show as a left scroll on display because leftmost column is represented by least significant bit of the byte.
        bitmap[row][zone] = bitmap[row][zone] >> 1;
        // Shift over lowest bit from the next zone as highest bit of this zone.
        if (zone < maxZoneIndex) bitWrite(bitmap[row][zone], 7, bitRead(bitmap[row][zone+1],0));
      }
    }
  }
 }
}
void setup()
{
 // mx.begin();
 // mx.setShiftDataInCallback(scrollDataSource);

 // scrollDelay = SCROLL_DELAY;
 // strcpy(curMessage, "Hello! ");
  //newMessage[0] = '\0';

  Serial.begin(115200);
```

```
 // Serial.print("\n[MD_MAX72XX Message Display]\nType a message for the scrolling display\nEnd message line with a newline");


  Serial.begin(115200);

  // Serial.setDebugOutput(true);


  Serial.println();

  Serial.println();

  Serial.println();


/*  for (uint8_t t = 4; t > 0; t--) {

    Serial.printf("[SETUP] WAIT %d...\n", t);

    Serial.flush();

    delay(1000);

  }

*/

  WiFi.mode(WIFI_STA);

  WiFiMulti.addAP("your hotspot name", "password");

  Serial.println("Connecting");

  newMessageAvailable = 1;

  present = millis();

  first_time = 1;

    // Setup MQTT subscription for onoff feed.

  mqtt.subscribe(&message);

 str = "   Ask Google assistant to change the msg!!!   ";

 pinMode(latchPin1, OUTPUT);
```

```
 pinMode(clockPin1, OUTPUT);
 pinMode(dataPin1, OUTPUT);

 pinMode(latchPin2, OUTPUT);
 pinMode(clockPin2, OUTPUT);
 pinMode(dataPin2, OUTPUT);

 //Clear bitmap
 for (int row = 0; row < 8; row++) {
   for (int zone = 0; zone <= maxZoneIndex; zone++) {
     bitmap[row][zone] = 0;

}
 }
}

void loop()
{
  while (WiFiMulti.run() != WL_CONNECTED) {
    Serial.println("WiFi not connected!");
    delay(1000);
  }
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).  See the MQTT_connect
  // function definition further below.
```

```
 MQTT_connect();

 // this is our 'wait for incoming subscription packets' busy subloop
 // try to spend your time here



 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription(1))) {
   if (subscription == &message) {
     payload ="";
     Serial.print(F("Got: "));
     Serial.println((char *)message.lastread);
     str = (char*)message.lastread;
     msg1=str;
     //Serial.println(msg1);
     payload = (String) str;
     payload += "       ";
     str = &payload[0];
     newMessageAvailable = 1;
   }
 }
XProcess();

//  scrollText();
```

```
}
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```