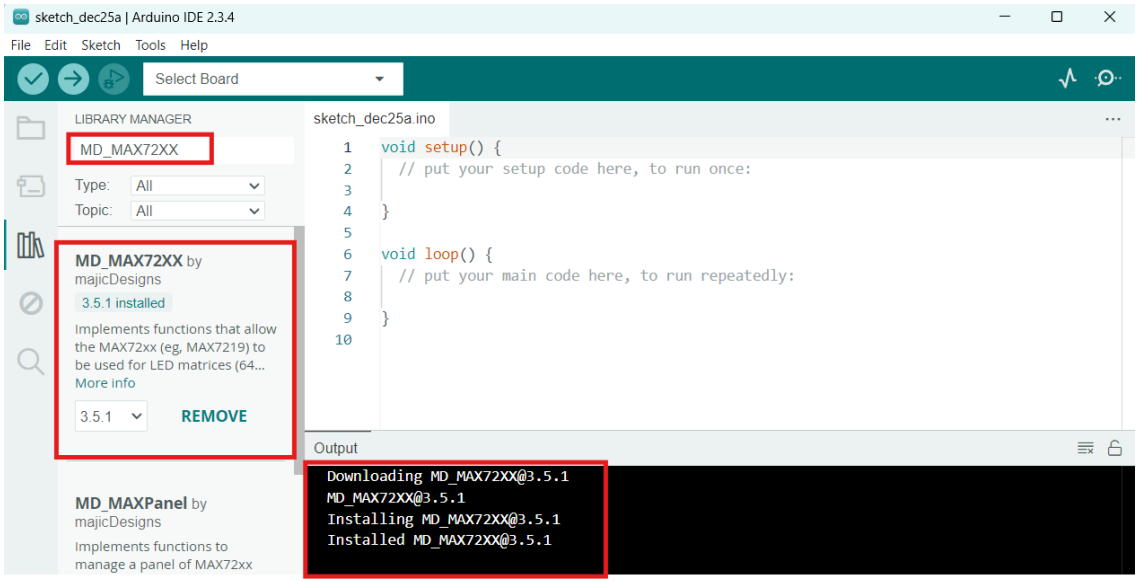
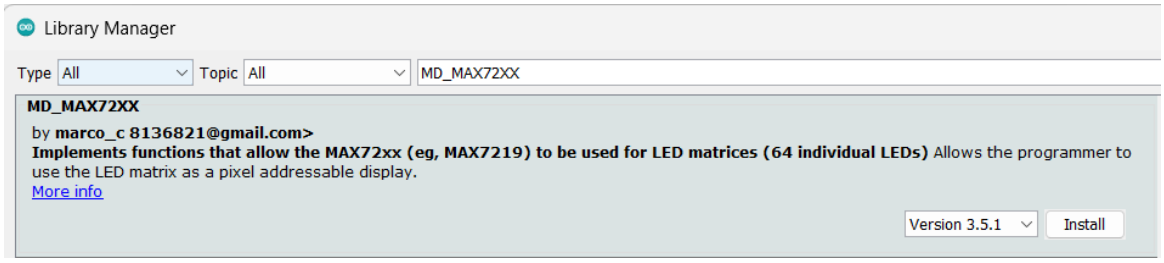
Getting Start ESP32: ESP32 GPIO + ESP32 Interface

Mission 7/12 – ESP32 + 8x8 Dot-matrix LED with MAX7219

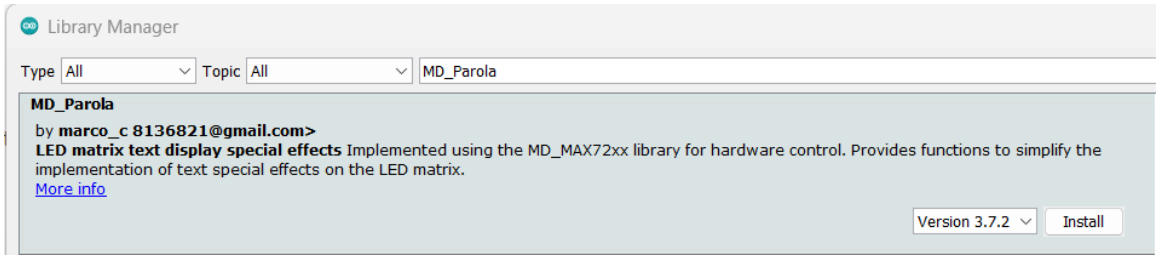
1. Wokwi Simulation https://wokwi.com/projects/388902495924727809
2. Read https://microcontrollerslab.com/led-dot-matrix-display-esp32-max7219/
3. Read http://www.learnerswings.com/2014/09/beautiful-running-arrow-demonstration.html
4. Add Library: Sketch → Include Library → Manage
5. Filter with "MD_MAX72XX", Select MD_MAX72XX by marco_c.. Version 3.5.1





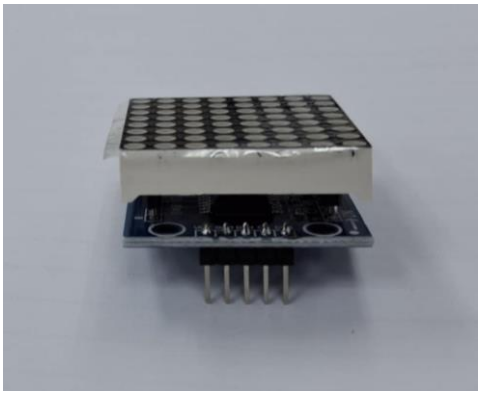
MD_MAX72XX by marco_c813682@gmail.com Version 3.5.1

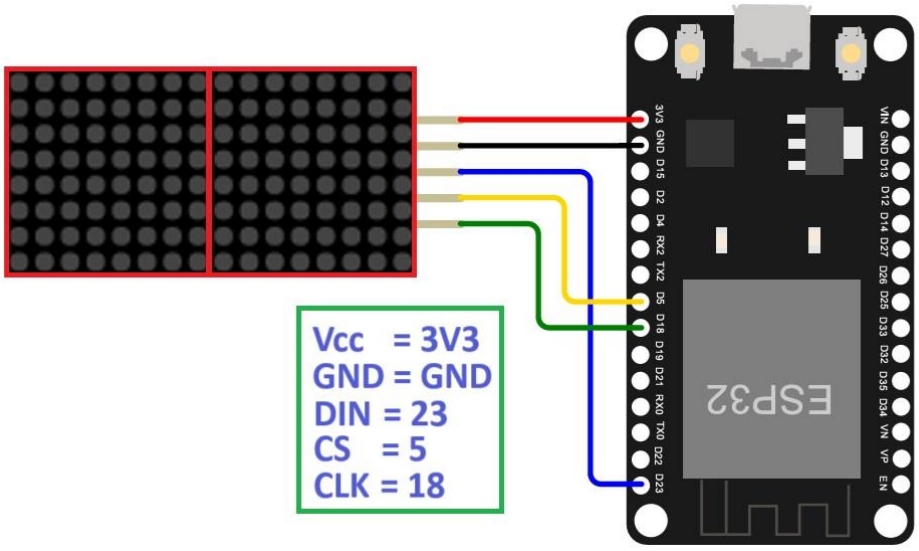
6. Filter with “MD_Parola”, Select MD_Parola by marco_c.. Version 3.7.2

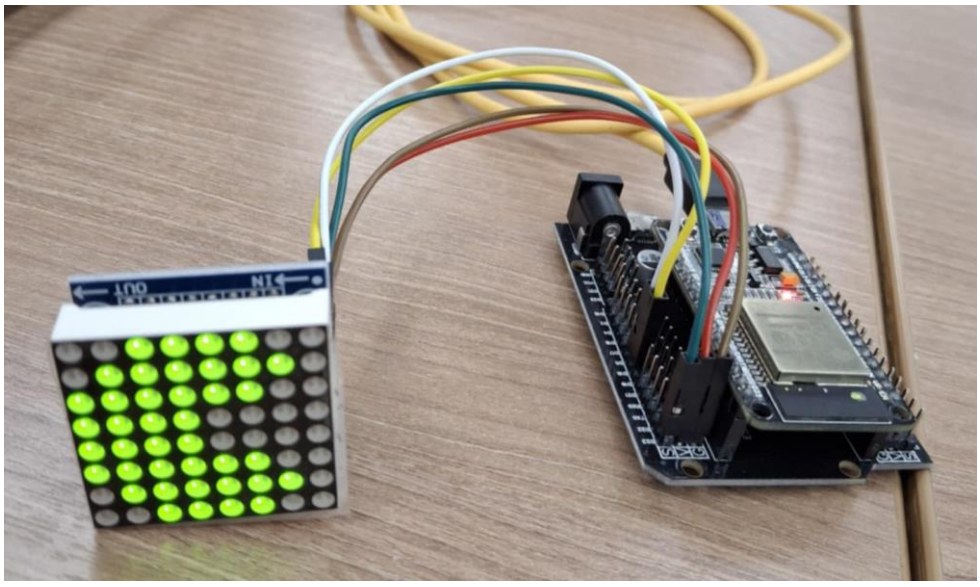


MD_Parola by marco_c813682@gmail.com Version 3.7.2

7. ต่อวงจร







8. Test Code “Test0601-Hello MAX7219 + 8x8Dot Matrix LED”

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#define MAX_DEVICES 1
#define CS_PIN 5 // DIO=23, CLK=18

// Uncomment according to your hardware type
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
// #define HARDWARE_TYPE MD_MAX72XX::GENERIC_HW

MD_Parola Display = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

void setup() {
  Display.begin();
  Display.setIntensity(6);
  Display.displayClear();
}

void loop() {
  Display.setTextAlignment(PA_CENTER);
  Display.setInvert(true);
  Display.print("W");  delay(1000);

  Display.setInvert(false);
  Display.print("W");  delay(1000);
  Display.print("I");  delay(1000);
  Display.print("C");  delay(1000);
  Display.print("H");  delay(1000);
  Display.print("A");  delay(1000);
  Display.print("I");  delay(1000);
}
```

MAX_DEVICES 1
1 คือจำนวนโมดูลที่นำมาต่อกัน

9. Test Code “Test0602-Moving Text”

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#define MAX_DEVICES 1
#define CS_PIN 5 // DIO=23, CLK=18

// Uncomment according to your hardware type
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
// #define HARDWARE_TYPE MD_MAX72XX::GENERIC_HW

MD_Parola Display = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

void setup() {
  Display.begin();
  Display.setIntensity(6);
  Display.displayClear();
}

void loop() {
  if (Display.displayAnimate())
    Display.displayText("Hello Test.1234", PA_CENTER, 250, 1000, PA_SCROLL_LEFT, PA_SCROLL_LEFT);

}
```

10. Test Code “Test0603-Spacial Character”

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#define Pause_Time  1000
#define Shift_Time  100
#define MAX_DEVICES 1
#define CS_PIN 5 // DIO=23, CLK=18

// Uncomment according to your hardware type
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
// #define HARDWARE_TYPE MD_MAX72XX::GENERIC_HW

MD_Parola P = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

#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)

const uint8_t degC[] = { 6, 3, 3, 56, 68, 68, 68 }; // Deg C
const uint8_t degF[] = { 6, 3, 3, 124, 20, 20, 4 }; // Deg F
const uint8_t waveSine[] = { 8, 1, 14, 112, 128, 128, 112, 14, 1 }; // Sine wave
const uint8_t waveSqar[] = { 8, 1, 1, 255, 128, 128, 128, 255, 1 }; // Square wave
const uint8_t waveTrng[] = { 10, 2, 4, 8, 16, 32, 64, 32, 16, 8, 4 }; // Triangle wave

// Global variables
typedef struct {
  uint8_t spacing;  // character spacing
  const char *msg;  // message to display
} msgDef_t;

msgDef_t  M[] = {
  { 1, "User char" },
  { 0, "~~~~" },
  { 1, "24$" },
  { 0, "++++" },
  { 1, "40&" },
  { 0, "^^^^" }
};

#define MAX_STRINGS  (sizeof(M)/sizeof(M[0]))

void setup() {
  Serial.begin(115200);
  PRINTS("\n[Parola User Char Demo]");
  P.begin();
  P.setIntensity(6);
  P.displayClear();

  P.addChar('$', degC);
  P.addChar('&', degF);
  P.addChar('~', waveSine);
  P.addChar('+', waveSqar);
  P.addChar('^', waveTrng);

  P.setCharSpacing(M[0].spacing);
  P.displayText(M[0].msg, PA_CENTER, Shift_Time, Pause_Time, PA_SCROLL_LEFT, PA_SCROLL_LEFT);

}

void loop(void) {
  static uint8_t  n = 1;
  if (P.displayAnimate()) {
    P.setTextBuffer(M[n].msg);
    P.setCharSpacing(M[n].spacing);
    P.displayReset();
    n = (n + 1) % MAX_STRINGS;
  }
}
```

11. Test Code “Test0604-Pac Man”

```
// Use the MD_MAX72XX library to display a Pacman animation
#include <MD_MAX72xx.h>
#include <SPI.h>

#define HARDWARE_TYPE MD_MAX72XX::PAROLA_HW
#define MAX_DEVICES 1
#define DATA_PIN  23  // or MOSI
#define CLK_PIN   18  // or SCK
#define CS_PIN    5  // or SS

MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);       // SPI hardware interface
//MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES); // Arbitrary pins
// ================================================
// Constant parameters
#define ANIMATION_DELAY 75  // milliseconds
#define MAX_FRAMES     4   // number of animation frames
// ============== General Variables ===============
const uint8_t pacman[MAX_FRAMES][18] =  { // ghost pursued by a pacman
  { 0xfe, 0x73, 0xfb, 0x7f, 0xf3, 0x7b, 0xfe, 0x00, 0x00, 0x00, 0x3c, 0x7e, 0x7e, 0xff, 0xe7, 0xc3, 0x81, 0x00 },
  { 0xfe, 0x7b, 0xf3, 0x7f, 0xfb, 0x73, 0xfe, 0x00, 0x00, 0x00, 0x3c, 0x7e, 0xff, 0xff, 0xe7, 0xe7, 0x42, 0x00 },
  { 0xfe, 0x73, 0xfb, 0x7f, 0xf3, 0x7b, 0xfe, 0x00, 0x00, 0x00, 0x3c, 0x7e, 0xff, 0xff, 0xff, 0xe7, 0x66, 0x24 },
  { 0xfe, 0x7b, 0xf3, 0x7f, 0xf3, 0x7b, 0xfe, 0x00, 0x00, 0x00, 0x3c, 0x7e, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x3c },
};

const uint8_t DATA_WIDTH = (sizeof(pacman[0]) / sizeof(pacman[0][0]));
uint32_t prevTimeAnim = 0;  // remember the millis() value in animations
int16_t idx;              // display index (column)
uint8_t frame;            // current animation frame
uint8_t deltaFrame;       // the animation frame offset for the next frame
// ========== Control routines ===========
void resetMatrix(void) {
  mx.control(MD_MAX72XX::INTENSITY, MAX_INTENSITY / 2);
  mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
  mx.clear();
}

void setup() {
  mx.begin();
  resetMatrix();
  prevTimeAnim = millis();
}

void loop(void) {
  static boolean bInit = true;  // initialise the animation

  // Is it time to animate?
  if (millis() - prevTimeAnim < ANIMATION_DELAY)
    return;
  prevTimeAnim = millis();     // starting point for next time

  mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::OFF);

  // Initialize
  if (bInit) {
    mx.clear();
    idx = -DATA_WIDTH;
    frame = 0;
    deltaFrame = 1;
    bInit = false;

    // Lay out the dots
    for (uint8_t i = 0; i < MAX_DEVICES; i++)   {
      mx.setPoint(3, (i * COL_SIZE) + 3, true);
      mx.setPoint(4, (i * COL_SIZE) + 3, true);
      mx.setPoint(3, (i * COL_SIZE) + 4, true);
      mx.setPoint(4, (i * COL_SIZE) + 4, true);
    }
  }

  // clear old graphic
  for (uint8_t i = 0; i < DATA_WIDTH; i++)
    mx.setColumn(idx - DATA_WIDTH + i, 0);
  // move reference column and draw new graphic
  idx++;
  for (uint8_t i = 0; i < DATA_WIDTH; i++)
    mx.setColumn(idx - DATA_WIDTH + i, pacman[frame][i]);

  // advance the animation frame
  frame += deltaFrame;
  if (frame == 0 || frame == MAX_FRAMES - 1)
    deltaFrame = -deltaFrame;

  // check if we are completed and set initialise for next time around
  bInit = (idx == mx.getColumnCount() + DATA_WIDTH);
  mx.control(MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
  return;
}
```