# Introduction to robotics

## 6th lab

Remember, when possible, choose the wire color accordingly:

- **BLACK** (or dark colors) for **GND**
- **RED** (or colored) for **POWER (3.3V / 5V / VIN)**
- **Remember** than when you use digitalWrite or analogWrite, you actually send power over the PIN, so you can use the same color as for **POWER**
- **Bright Colored** for read signal
- We know it is not always possible to respect this due to lack of wires, but the first rule is **NOT USE BLACK FOR POWER OR RED FOR GND!**

Now, let's pick it up where we left off…

Pull out your Arduino and breadboard and connect them like in the schematic. This is to "power up" the breadboard so we can easily have access to **5V** and **GND**.

**Attention! Remember how the breadboard works. Use correct wire colors.**

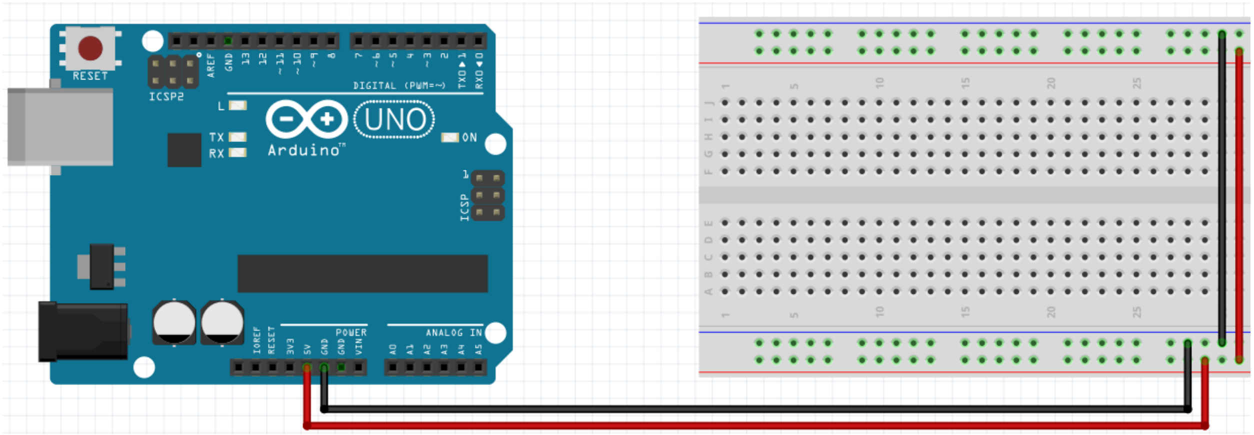


**Fig. 0.1** - Default setup

# 1. MAX7219 Driver

Datasheet: https://www.sparkfun.com/datasheets/Components/General/COM-09622-MAX7219-MAX7221.pdf

The MAX7219/MAX7221 are compact, serial input/out-put common-cathode display drivers that interface microprocessors (µPs) to 7-segment numeric LED displays of up to 8 digits, bar-graph displays, or 64 individual LEDs.

(source: Datasheet)

Basically, it’s a simple and somewhat inexpensive method of controlling 64 LEDs in either matrix or numeric display form. Furthermore they can be chained together to control two or more units for even more LEDs. Overall – they’re a lot of fun and can also be quite useful.

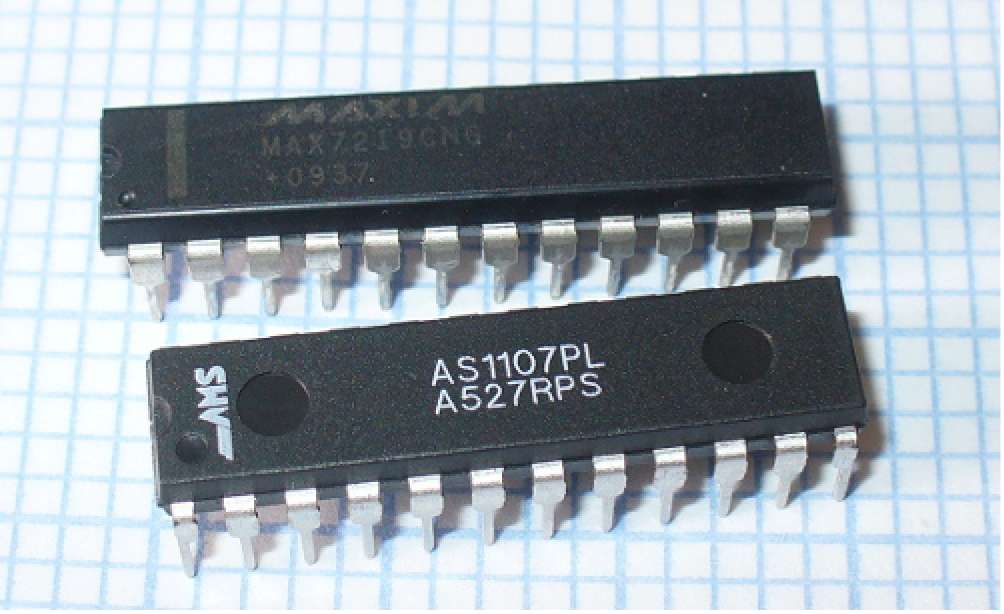


**Fig. 1.1** MAX7219 LED Driver

As mentioned earlier, the MAX7219 can completely control 64 individual LEDs – including maintaining equal brightness, and allowing you to adjust the brightness of the LEDs either with hardware or software (or both). It can refresh the LEDs at around 800 Hz (min 500Hz - max 1300Hz), so no more flickering, uneven LED displays.

## 2. Controlling the LED matrix display with the MAX7219

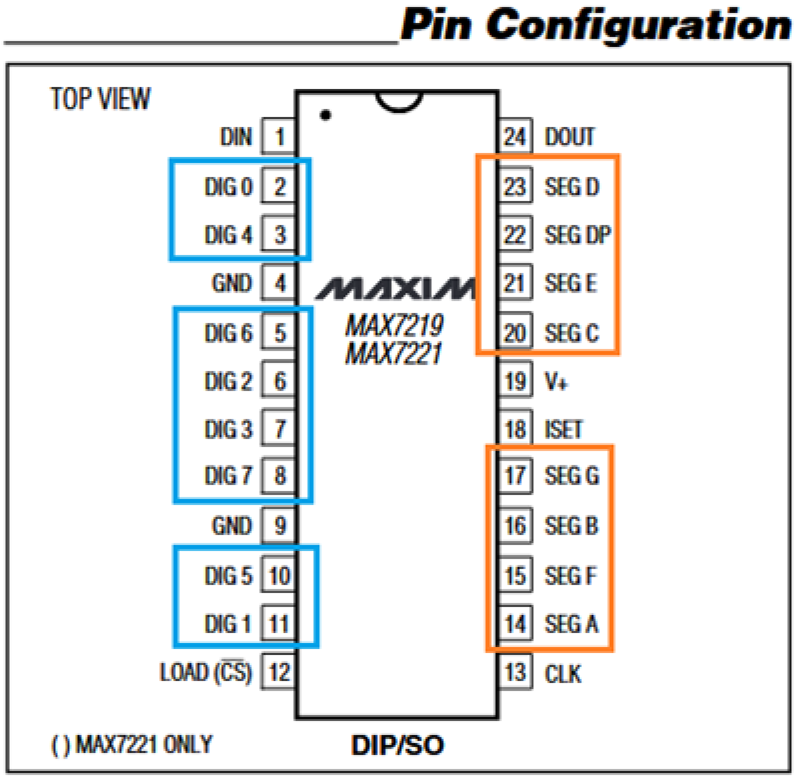
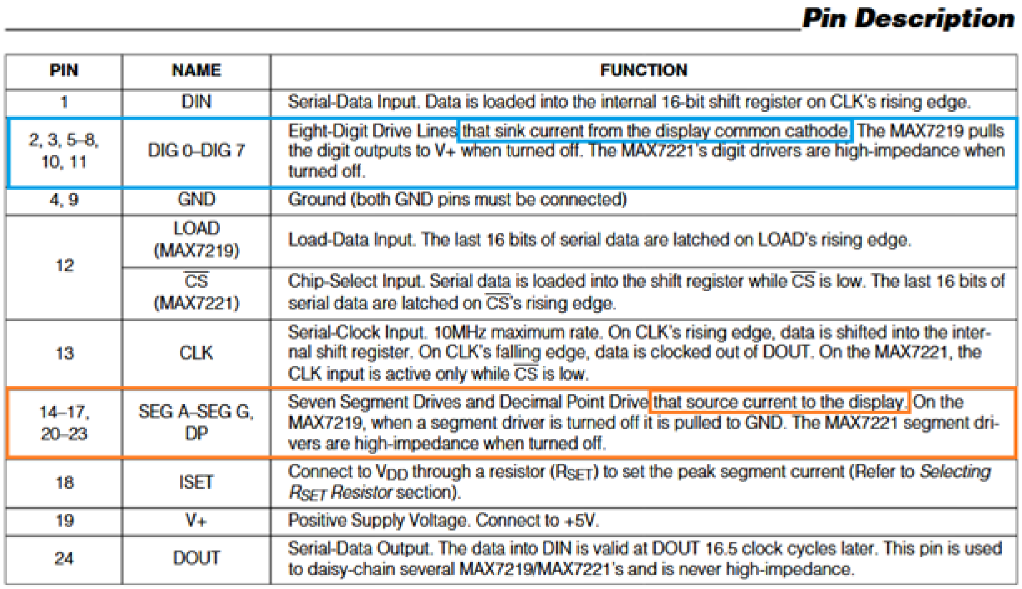


Fig. 2.1 MAX7219 Pin Configuration



Pin Description

| PIN | NAME | FUNCTION |
|---|---|---|
| 1 | DIN | Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge. |
| 2, 3, 5–8, 10, 11 | DIG 0–DIG 7 | Eight-Digit Drive Lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off. |
| 4, 9 | GND | Ground (both GND pins must be connected) |
| 12 | LOAD (MAX7219) | Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge. |
| | $\overline{\text{CS}}$ (MAX7221) | Chip-Select Input. Serial data is loaded into the shift register while $\overline{\text{CS}}$ is low. The last 16 bits of serial data are latched on $\overline{\text{CS}}$'s rising edge. |
| 13 | CLK | Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while $\overline{\text{CS}}$ is low. |
| 14–17, 20–23 | SEG A–SEG G, DP | Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off. |
| 18 | ISET | Connect to $V_{DD}$ through a resistor ($R_{SET}$) to set the peak segment current (Refer to *Selecting $R_{SET}$ Resistor* section). |
| 19 | V+ | Positive Supply Voltage. Connect to +5V. |
| 24 | DOUT | Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance. |

Fig. 2.2 MAX7219 Pin Description

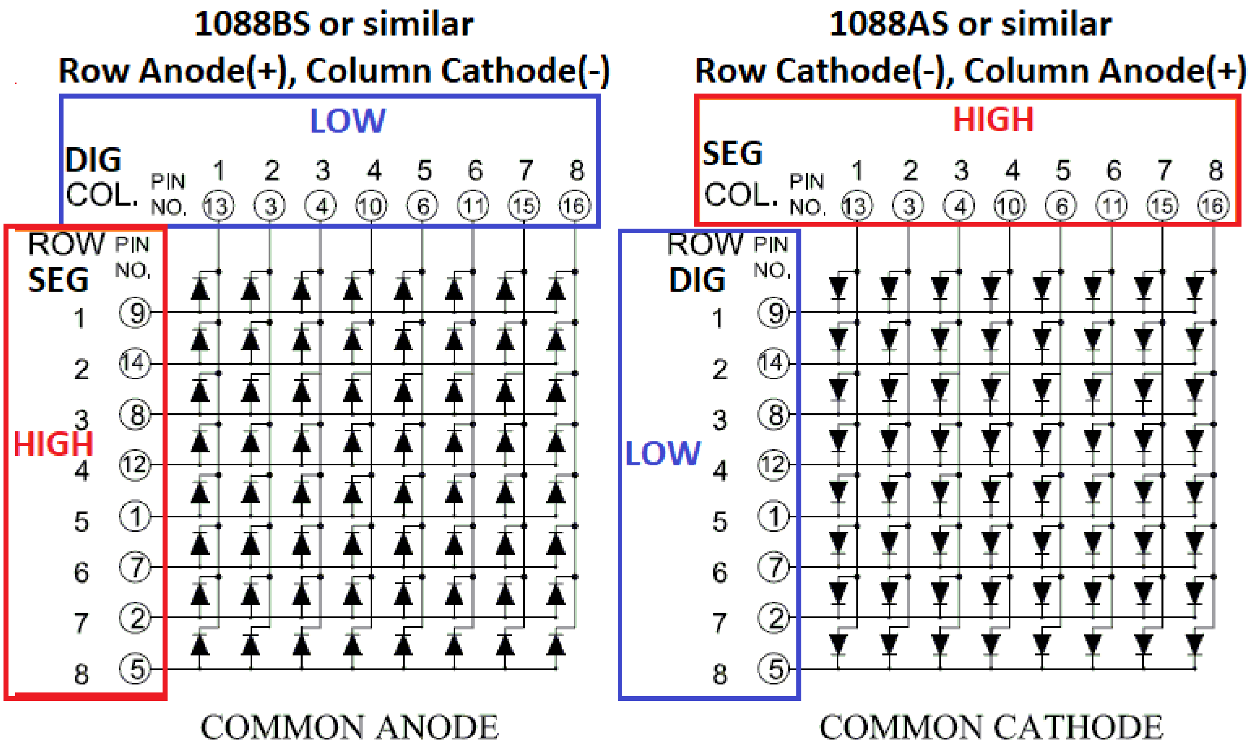
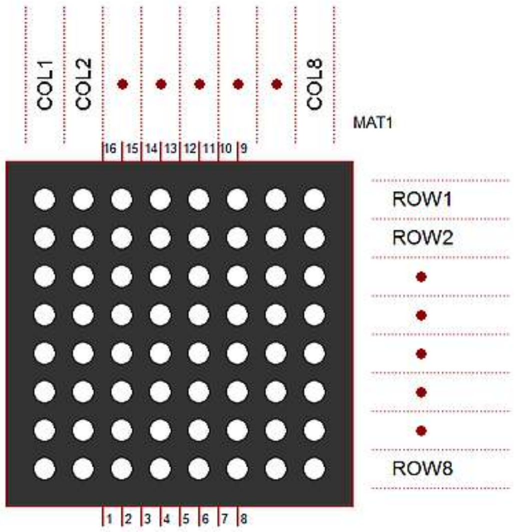


**Fig. 2.3 8x8 LED Matrix configuration (common anode vs common cathode)**

**From the description it is clear that:**

1. **DIG** pins are used for the cathodes of the LED matrix
2. **SEG** pins are used for the anodes of the LED matrix
3. Careful, first check which type of matrix you have



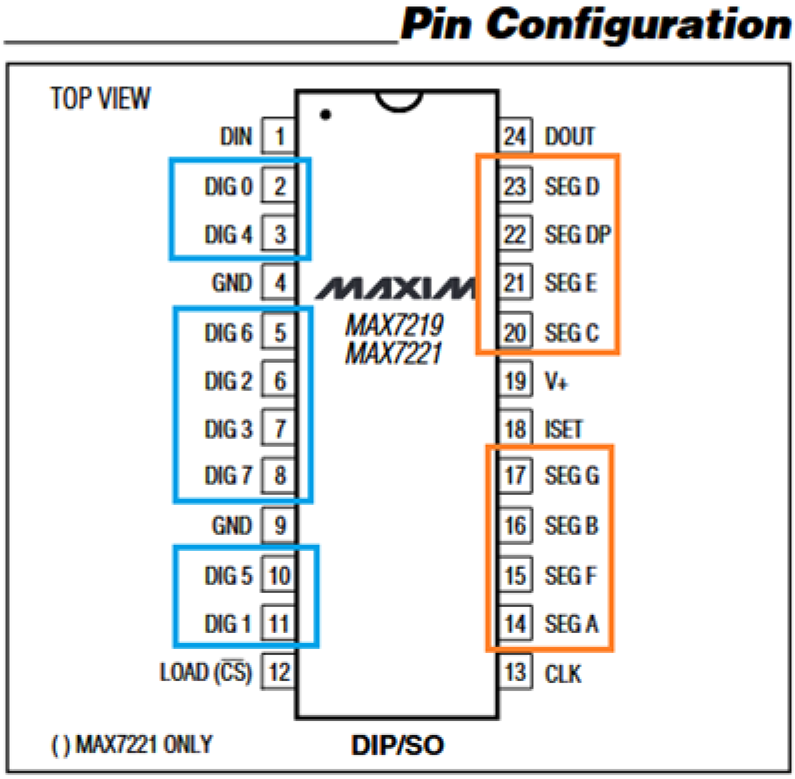
**!important Check with a multimeter to see what matrix type you have. DO NOT RELY ON THE WRITTEN NUMBER.**

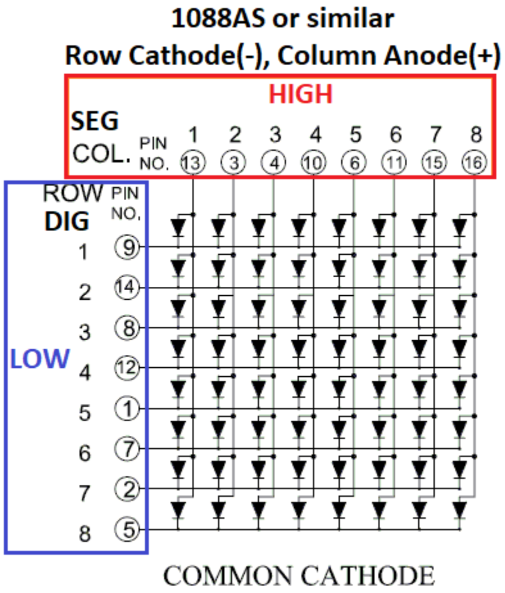
**Fig 2.3 Matrix structure & Pin Numbering**

# 3. Matrix to Driver Connections Table:

## 3.1 1088AS or similar (Row Cathode-, Column Anode+)

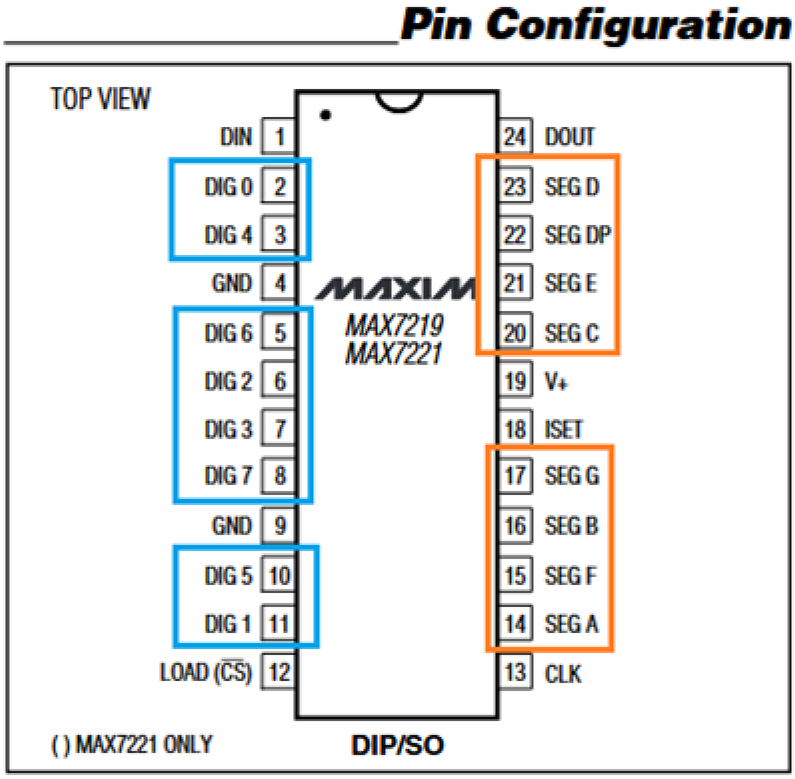
| 1088AS Matrix (Row Cathode-, Column Anode+) | | MAX7219 Driver | |
|---|---|---|---|
| Row / Column | Matrix Pin | DIG / SEG Number | Pin Number |
| Row 5 | 1 | DIG4 | 3 |
| Row 7 | 2 | DIG6 | 5 |
| Col 2 | 3 | SEG A | 14 |
| Col 3 | 4 | SEG B | 16 |
| Row 8 | 5 | DIG7 | 8 |
| Col 5 | 6 | SEG D | 23 |
| Row 6 | 7 | DIG5 | 10 |
| Row 3 | 8 | DIG2 | 6 |
| Row 1 | 9 | DIG0 | 2 |
| Col 4 | 10 | SEG C | 20 |
| Col 6 | 11 | SEG E | 21 |
| Row 4 | 12 | DIG3 | 7 |
| Col 1 | 13 | SEG DP | 22 |
| Row 2 | 14 | DIG1 | 11 |
| Col 7 | 15 | SEG F | 15 |
| Col 8 | 16 | SEG G | 17 |

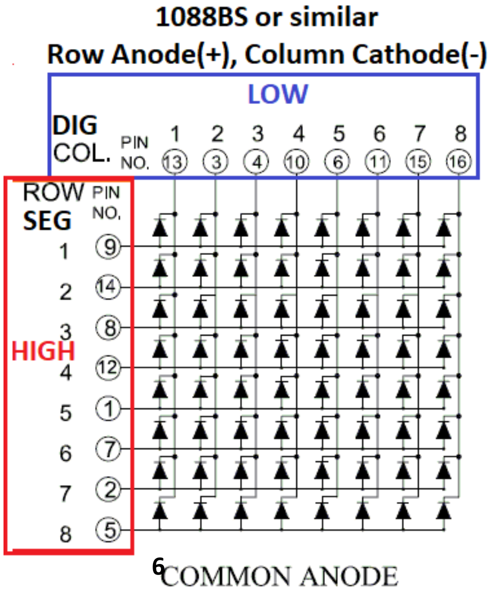




### 3.2 1088BS or similar (Row Anode+, Column Cathode-)

| 1088BS Matrix (Row Anode+, Column Cathode-) | | MAX7219 Driver | |
|---|---|---|---|
| Row / Column | Matrix Pin | DIG Number | Pin Number |
| Col 5 | 1 | SEG D | 23 |
| Col 7 | 2 | SEG F | 15 |
| Row 2 | 3 | DIG1 | 11 |
| Row 3 | 4 | DIG2 | 6 |
| Col 8 | 5 | SEG G | 17 |
| Row 5 | 6 | DIG4 | 3 |
| Col 6 | 7 | SEG E | 21 |
| Col 3 | 8 | SEG B | 16 |
| Col 1 | 9 | SEG DP | 22 |
| Row 4 | 10 | DIG3 | 7 |
| Row 6 | 11 | DIG5 | 10 |
| Col 4 | 12 | SEG C | 20 |
| Row 1 | 13 | DIG0 | 2 |
| Col 2 | 14 | SEG A | 14 |
| Row 7 | 15 | DIG6 | 5 |
| Row 8 | 16 | DIG7 | 8 |





## 4. Connecting the driver to Arduino

**Note**: if the connection doesn't work, try a 100k resistor instead of 10k.

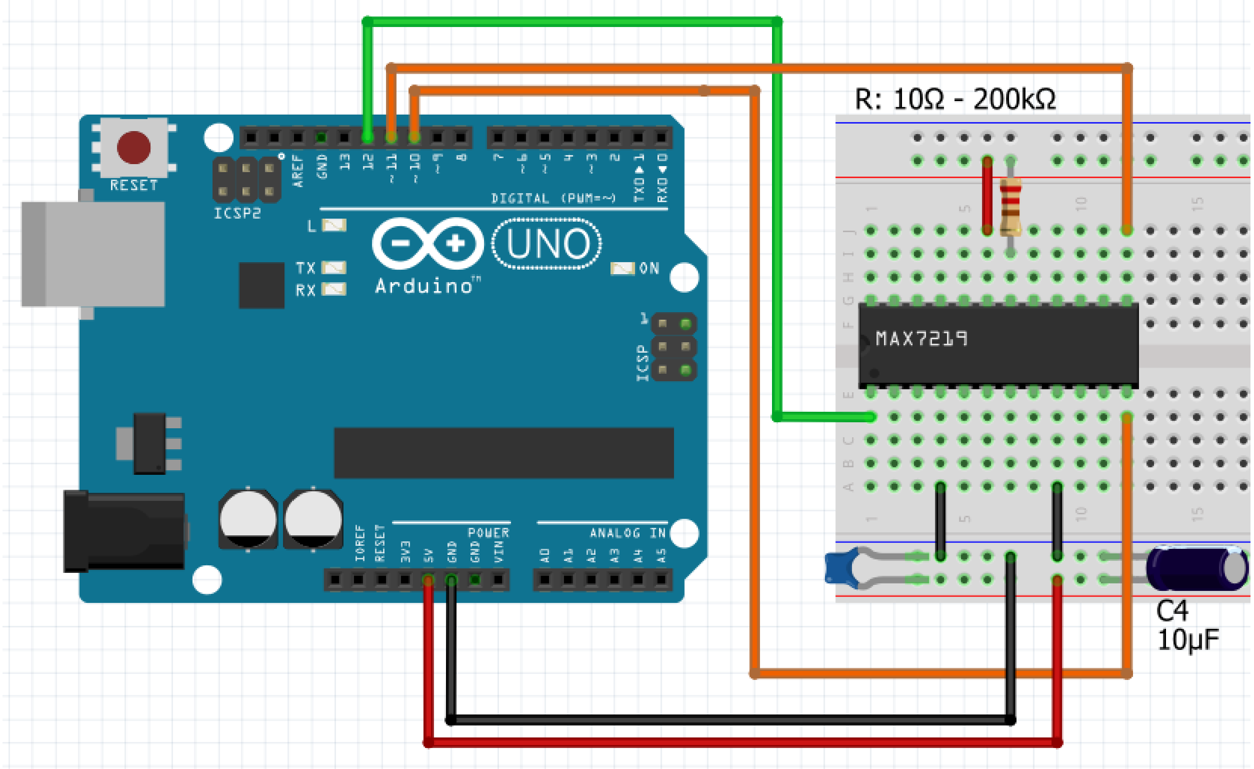


**Fig 4.1 MAX7219 Arduino Connections**

To minimize power-supply ripple due to the peak digit driver currents, connect a 10µF electrolytic and a 0.1µF ceramic capacitor between V+ and GND as close to the device as possible. The MAX7219/MAX7221 should be placed in close proximity to the LED display, and connections should be kept as short as possible to minimize the effects of wiring inductance and electromagnetic interference. Also, both GND pins must be connected to ground.

**!important If connected wrong, the electrolytic capacitor can blow up (remember the first lab, that's what we blew up).**

## Connection list:

| Max7219 Driver Pins | Arduino Pins |
|---|---|
| 4 (GND) | GND |
| 9 (GND) | GND |
| 18 (ISET) | 5V, **through a 10k or 100k+ resistor** |
| 19 (V+) | 5V |
| 1 (DIN) | 12 |
| 12 (LOAD/CS) | 10 |
| 13 (CLK) | 11 |

As you can see in the schematic, there are also 2 capacitors that are connected in parallel to the + and - of our circuit.

- 1 electrolytic capacitor of 10 μF
- 1 ceramic capacitor of 104 pF

**If you change the pinout order, remember to change it in the code as well.**

**The schematic is or 1088AS, pay attention to what type of matrix you hav**

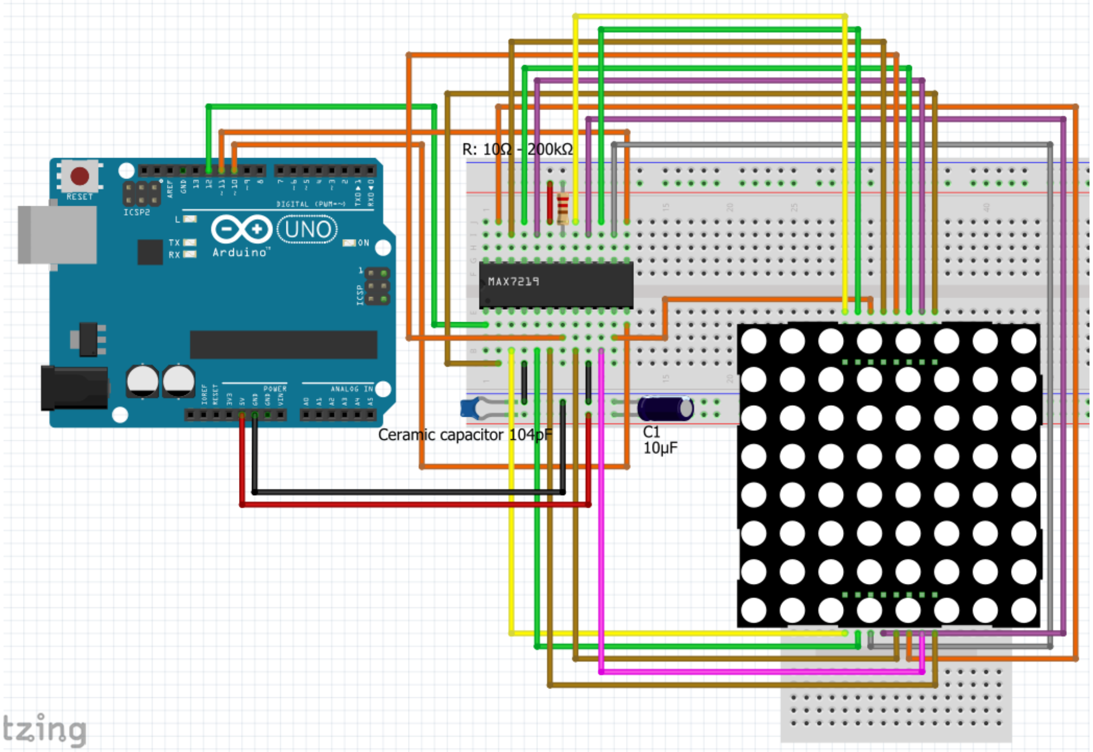


Fig 4.1 1088 AS Matrix Connections

# 5. Code Examples

## 5.1 Turning the each led on and off

We will use the **LedControl library** in our code.

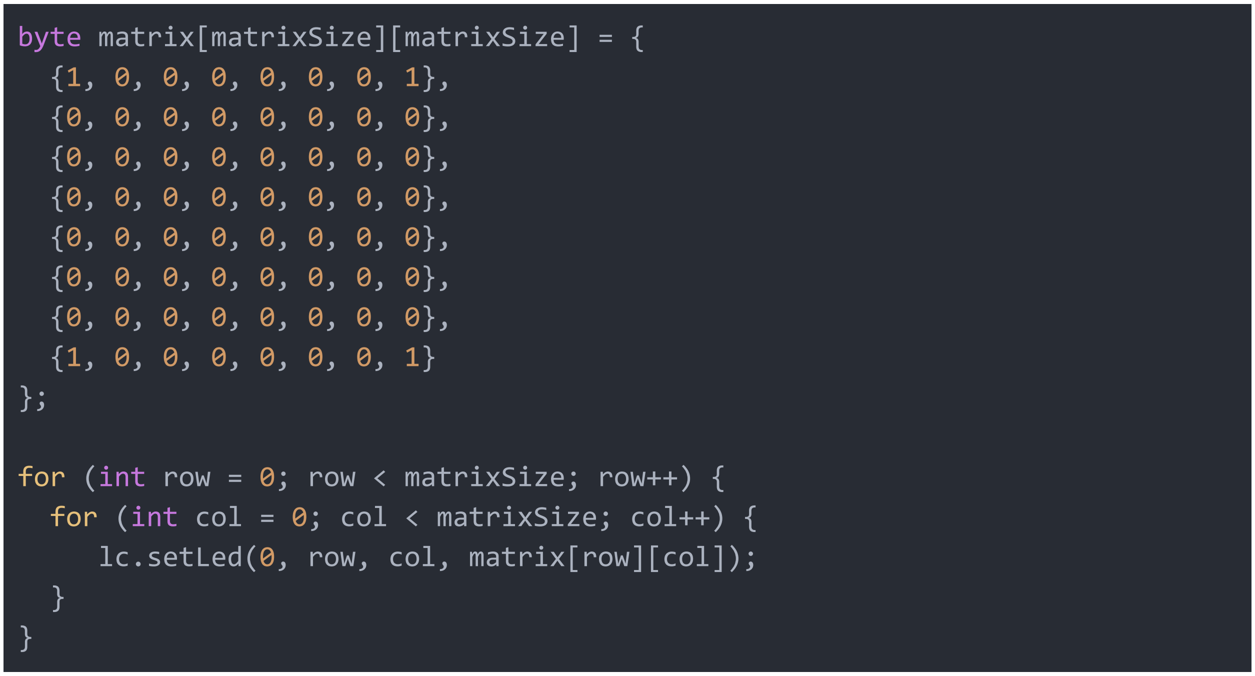
```
#include "LedControl.h" //  need the library
const byte dinPin = 12;
const byte clockPin = 11;
const byte loadPin = 10;
const byte matrixSize = 8;
// pin 12 is connected to the MAX7219 pin 1
// pin 11 is connected to the CLK pin 13
// pin 10 is connected to LOAD pin 12
// 1 as we are only using 1 MAX7219
LedControl lc = LedControl(dinPin, clockPin, loadPin, 1); //DIN, CLK, LOAD, No.
DRIVER
byte matrixBrightness = 2;

void setup() {
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0, false); // turn off power saving, enables display
  lc.setIntensity(0, matrixBrightness); // sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
}

void loop() {
  for (int row = 0; row < matrixSize; row++) {
    for (int col = 0; col < matrixSize; col++) {
      lc.setLed(0, row, col, true); // turns on LED at col, row
      delay(25);
    }
  }
  for (int row = 0; row < matrixSize; row++) {
    for (int col = 0; col < matrixSize; col++) {
      lc.setLed(0, row, col, false); // turns off LED at col, row
      delay(25);
    }
  }
}
```

## 5.2 Matrix representation

**There are 2 ways to represent the matrix.**

### 5.2.1 8x8 byte matrix

**8*8 array  which you can cycle through, setting the value on each LED**



```
byte matrix[matrixSize][matrixSize] = {
  {1, 0, 0, 0, 0, 0, 0, 1},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {1, 0, 0, 0, 0, 0, 0, 1}
};

for (int row = 0; row < matrixSize; row++) {
  for (int col = 0; col < matrixSize; col++) {
     lc.setLed(0, row, col, matrix[row][col]);
  }
}
```

### 5.2.2 8-byte array

**8-byte array of bytes, which you can cycle through, setting the value on each ROW**

```
const byte matrixByte[matrixSize] = {
  B10000001,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B10000001
};

for (int row = 0; row < matrixSize; row++) {
  lc.setRow(0, row, matrixByte[row]);
}
```

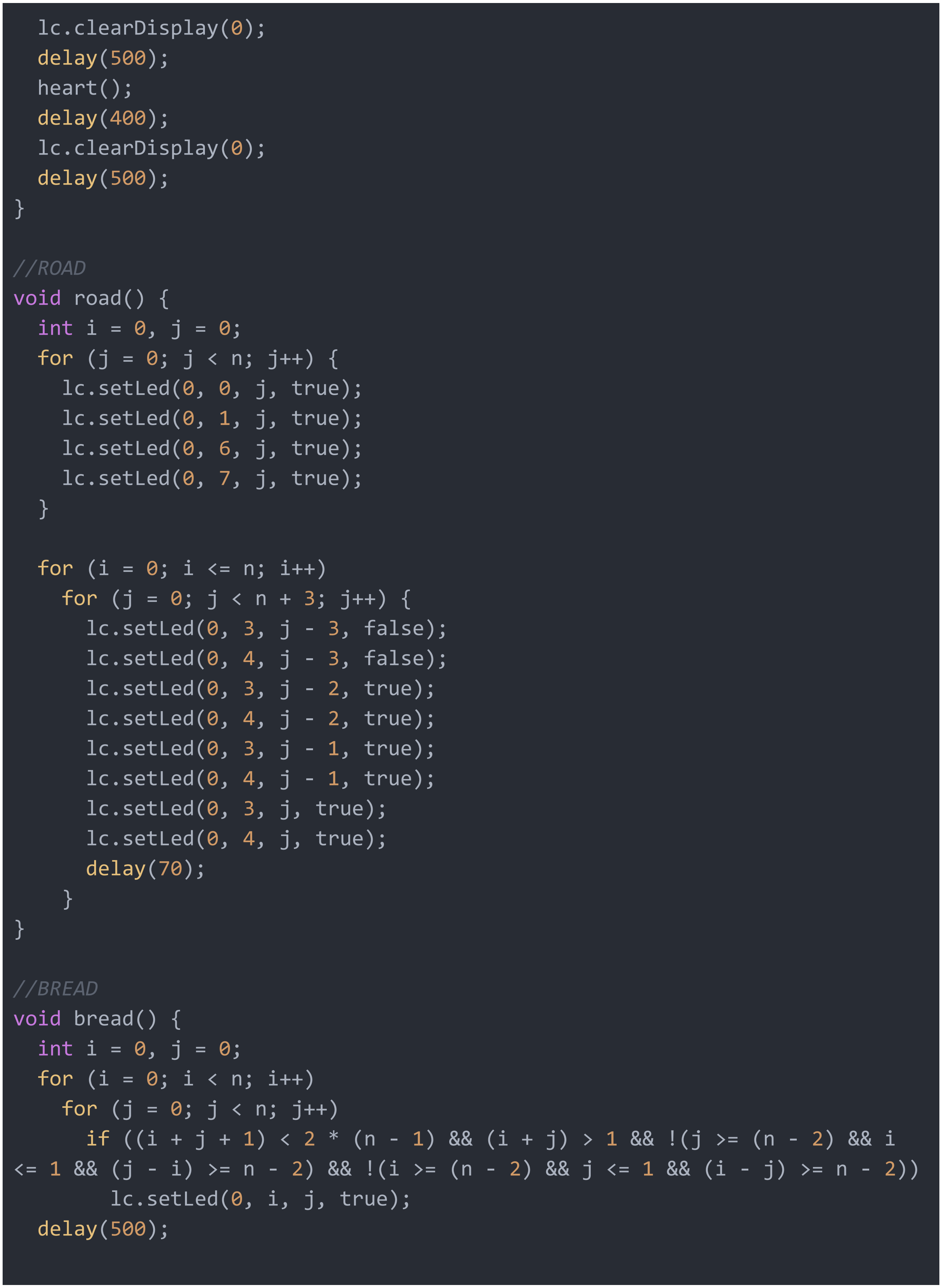
### 5.3 Both representations in practice

```
#include "LedControl.h" //  need the library
const byte dinPin = 12;
const byte clockPin = 11;
const byte loadPin = 10;
const byte matrixSize = 8;
// pin 12 is connected to the MAX7219 pin 1
// pin 11 is connected to the CLK pin 13
// pin 10 is connected to LOAD pin 12
// 1 as we are only using 1 MAX7219
LedControl lc = LedControl(dinPin, clockPin, loadPin, 1); //DIN, CLK, LOAD, No. DRIVER

byte matrixBrightness = 2;


byte matrix[matrixSize][matrixSize] = {
  {1, 0, 0, 0, 0, 0, 0, 1},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {1, 0, 0, 0, 0, 0, 0, 1}
};

byte matrixByte[matrixSize] =
{
  B10000001,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B10000001
};


void setup() {
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0, false); // turn off power saving, enables display
  lc.setIntensity(0, matrixBrightness); // sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
}


void loop() {
  // for byte matrix
  for (int row = 0; row < matrixSize; row++) {
    for (int col = 0; col < matrixSize; col++) {
       lc.setLed(0, row, col, matrix[row][col]);
    }
  }
  // for byte matrix
//  for (int row = 0; row < matrixSize; row++)
//  {
//    lc.setRow(0, row, matrixByte[row]);
//  }
}
```

### 5.4 Some animations

**(BADLY WRITTEN CODE, DON’T IMITATE IT)**
**Forgot the source, somewhere on the internet. If you find it, please linkit.**

```
#include "LedControl.h" //  need the library
const int dinPin = 12;
const int clockPin = 11;
const int loadPin = 10;
const int rows = 8;
const int cols = 8;
LedControl lc = LedControl(dinPin, clockPin, loadPin, 1); //DIN, CLK, LOAD, No. DRIVER

const int n = 8;

void setup() {
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0, false); // turn off power saving, enables display
  lc.setIntensity(0, 2); // sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
}
void loop() {
  //some animations
  road();
  delay(500);
  lc.clearDisplay(0);
  bread(); //or pizza
  delay(2000);
  lc.clearDisplay(0);
  flower();
  delay(2000);
  lc.clearDisplay(0);
  heart();
  delay(400);
  lc.clearDisplay(0);
  delay(500);
  heart();
  delay(400);
  lc.clearDisplay(0);
  delay(500);
  heart();
  delay(400);
```



```
    lc.clearDisplay(0);
    delay(500);
    heart();
    delay(400);
    lc.clearDisplay(0);
    delay(500);
}

//ROAD
void road() {
  int i = 0, j = 0;
  for (j = 0; j < n; j++) {
    lc.setLed(0, 0, j, true);
    lc.setLed(0, 1, j, true);
    lc.setLed(0, 6, j, true);
    lc.setLed(0, 7, j, true);
  }

  for (i = 0; i <= n; i++)
    for (j = 0; j < n + 3; j++) {
      lc.setLed(0, 3, j - 3, false);
      lc.setLed(0, 4, j - 3, false);
      lc.setLed(0, 3, j - 2, true);
      lc.setLed(0, 4, j - 2, true);
      lc.setLed(0, 3, j - 1, true);
      lc.setLed(0, 4, j - 1, true);
      lc.setLed(0, 3, j, true);
      lc.setLed(0, 4, j, true);
      delay(70);
    }
}

//BREAD
void bread() {
  int i = 0, j = 0;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if ((i + j + 1) < 2 * (n - 1) && (i + j) > 1 && !(j >= (n - 2) && i <= 1 && (j - i) >= n - 2) && !(i >= (n - 2) && j <= 1 && (i - j) >= n - 2))
        lc.setLed(0, i, j, true);
  delay(500);
  for (i = 0; i < n / 2; i++) {
    for (j = 0; j < n / 2; j++) {
      lc.setLed(0, i, j, false);
    }
  }
  delay(500);

  for (i = 0; i < n / 2; i++) {
    for (j = 0; j < n / 2 + 1; j++) {
      lc.setLed(0, i, n - j, false);
    }
  }
  delay(500);

  for (i = 0; i < n / 2 + 1; i++) {
    for (j = 0; j < n / 2 + 1; j++) {
      lc.setLed(0, n - i, n - j, false);
    }
  }
  delay(500);

  for (i = 0; i < n / 2 + 1; i++) {
    for (j = 0; j < n / 2 + 1; j++) {
      lc.setLed(0, n - i, j, false);
    }
  }
}

//FLOWER
void flower() {
  for (int i = 3; i >= 1; i--) {
    for (int j = 3; j >= 1; j--) {
      if ((i + j) != 4) {
        lc.setLed(0, i, j, true);
        delay(150);
        lc.setLed(0, n - i - 1, n - j - 1, true);
        delay(150);
        lc.setLed(0, n - i - 1, j, true);
        delay(150);
        lc.setLed(0, i, n - j - 1, true);
        delay(150);
      }
    }
  }
}

//HEART
void heart() {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n / 2; j++) {
      if (
        (i == 0 && (j == 0 || j == 3)) ||
        (i == 5 &&  j == 0) ||
        (i == 6 && (j == 0 || j == 1)) ||
        (i == 7 && (j == 0 || j == 1 || j == 2))
      ) {
        lc.setLed(0, i, j, false);
        lc.setLed(0, i, n - j - 1, false);
      }

      else {
        lc.setLed(0, i, j, true);
        lc.setLed(0, i, n - j - 1, true);
      }
    }
  }
}
```

# 6. Let’s think game-like

## Controlling a point on the matrix with the joystick

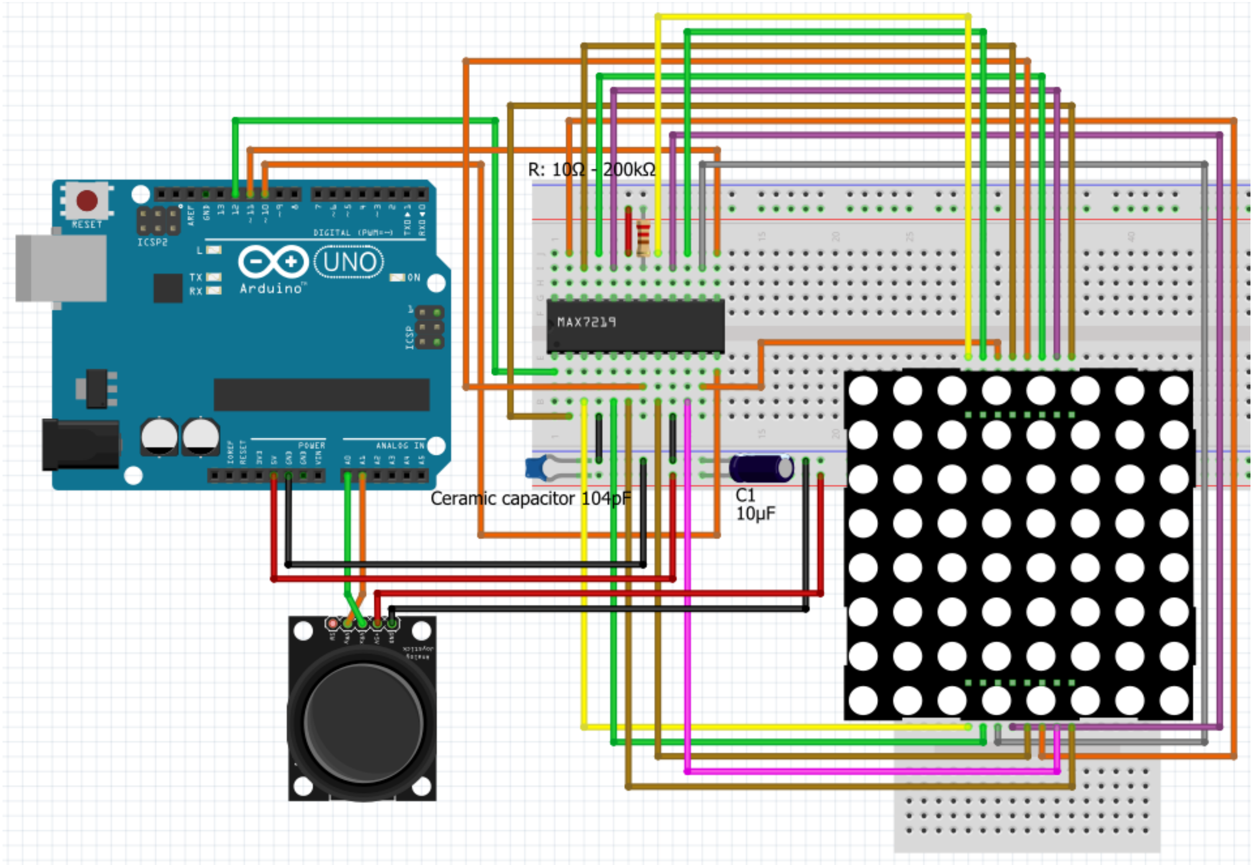


Fig 6.1 Arduino UNO, MAX7219, 1088AS Matrix and Joystick connections

```
#include "LedControl.h" //  need the library
const int dinPin = 12;
const int clockPin = 11;
const int loadPin = 10;

const int xPin = A0;
const int yPin = A1;

LedControl lc = LedControl(dinPin, clockPin, loadPin, 1); // DIN, CLK,
LOAD, No. DRIVER

byte matrixBrightness = 2;

byte xPos = 0;
byte yPos = 0;
byte xLastPos = 0;
byte yLastPos = 0;

const int minThreshold = 200;
const int maxThreshold = 600;

const byte moveInterval = 100;
unsigned long long lastMoved = 0;

const byte matrixSize = 8;
bool matrixChanged = true;

byte matrix[matrixSize][matrixSize] = {
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0}
};

byte matrixByte[matrixSize] = {
  B00000000,
  B01000100,
  B00101000,
  B00010000,
  B00010000,
  B00010000,
  B00000000,
  B00000000
};

void setup() {
  Serial.begin(9600);
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0, false); // turn off power saving, enables display
  lc.setIntensity(0, matrixBrightness); // sets brightness (0~15 possible
values)
  lc.clearDisplay(0);// clear screen
  matrix[xPos][yPos] = 1;
}
void loop() {
//  updateByteMatrix();
  if (millis() - lastMoved > moveInterval) {
  // game logic
    updatePositions();
    lastMoved = millis();
  }
  if (matrixChanged == true) {
    // matrix display logi
    updateMatrix();
    matrixChanged = false;
  }
}
// theoretical example
void generateFood() {
  // lastFoodPos = currentPos;
  // newFoodPos = random(ceva);
  // matrix[lastFoodPos] = 0;
  // matrix[newFoodPos] = 1;
  matrixChanged = true;
}

void updateByteMatrix() {
  for (int row = 0; row < matrixSize; row++) {
    lc.setRow(0, row, matrixByte[row]);
  }
}
void updateMatrix() {
  for (int row = 0; row < matrixSize; row++) {
    for (int col = 0; col < matrixSize; col++) {
      lc.setLed(0, row, col, matrix[row][col]);
    }
  }
}

void updatePositions() {
  int xValue = analogRead(xPin);
  int yValue = analogRead(yPin);

  xLastPos = xPos;
  yLastPos = yPos;
  if (xValue < minThreshold) {
    if (xPos < matrixSize - 1) {
      xPos++;
    }
    else {
      xPos = 0;
    }
  }
  if (xValue > maxThreshold) {
    if (xPos > 0) {
      xPos--;
    }
    else {
      xPos = matrixSize - 1;
    }
  }

  if (yValue > maxThreshold) {
    if (yPos < matrixSize - 1) {
      yPos++;
    }
    else {
      yPos = 0;
    }
  }

  if (yValue < minThreshold) {
    if (yPos > 0) {
      yPos--;
    }
    else {
      yPos = matrixSize - 1;
    }
  }
  if (xPos != xLastPos || yPos != yLastPos) {
    matrixChanged = true;
    matrix[xLastPos][yLastPos] = 0;
    matrix[xPos][yPos] = 1;
  }
}
```

# Resources

1. Matrix and scanning explanation https://www.youtube.com/watch?v=G4lIo-MRSiY
2. Led matrix editor: https://xantorohara.github.io/led-matrix-editor/
3. https://www.youtube.com/watch?v=X9tsfOeYnAU