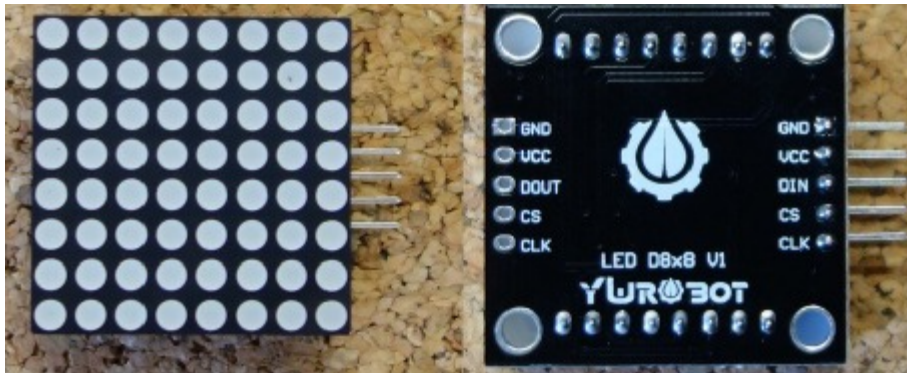


Guide for 8×8 Dot Matrix MAX7219 with Arduino + Pong Game

The dot matrix that we're going to use in this guide is a 8×8 matrix which means that it has 8 columns and 8 rows, so it contains a total of 64 LEDs.

The MAX7219 chip makes it easier to control the dot matrix, by just using 3 digital pins of the Arduino board.

I think the best option is to buy the dot matrix with the MAX7219 chip as a module, it will simplify the wiring. You can [get this module on ebay for less than 2\\$](#).



You can control more than one matrix at a time. For that you just need to connect them to each other, as they have pins in both sides to extend the dot matrix.

Parts required

RELATED CONTENT: [Like ESP8266? Check out Home Automation Using ESP8266](#)

For this guide you'll need:

- 1x 8×8 Dot Matrix with MAX7219 ([view on eBay](#))
- 1x Arduino ([view on eBay](#))
- 1x 1k ohm Potentiometer
- Jumper wires

Pin Wiring

You only need to connect 5 pins from the dot matrix to your Arduino board. The wiring is pretty straightforward:

| Dot matrix pin | Wiring to Arduino Uno |
|----------------|-----------------------|
| GND | GND |
| VCC | 5V |
| DIN | Digital pin |
| CS | Digital pin |
| CLK | Digital pin |

How to control the dot matrix with Arduino

For making it easier to control the dot matrix, you need to download and install in your Arduino IDE the LedControl library. To install the library follow these steps:

1. [Click here to download the LedControl library](#). You should have a .zip folder in your Downloads
2. Unzip the .zip folder and you should get **LedControl-master** folder
3. Rename your folder from **LedControl-master** to **LedControl**
4. Move the **LedControl** folder to your Arduino IDE installation **libraries** folder
5. Finally, re-open your Arduino IDE

Using the LedControl library functions

The easiest way to display something on the dot matrix is by using the functions *setLed()*, *setRow()* or *setColumn()*. These functions allow you to control one single led, one row or one column at a time.

Here's the parameters for each function:

setLed(addr, row, col, state)

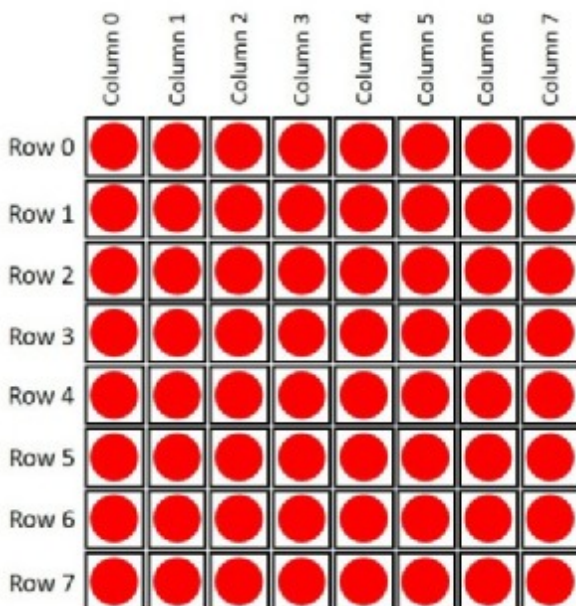
- **addr** is the address of your matrix, for example, if you have just 1 matrix, the int addr will be zero.
- **row** is the row where the led is located
- **col** is the column where the led is located
- **state**
 - It's true or 1 if you want to turn the led on
 - It's false or 0 if you want to switch it off

setRow(addr, row, value)

setCol(addr, column, value)

































































Index

As previously stated, this matrix has 8 columns and 8 rows. Each one is indexed from 0 to 7. Here's a figure for better understanding:



If you want to display something in the matrix, you just need to know if in a determined row or column, the LEDs that are on or off.

For example, if you want to display a happy face, here's what you need to do:

| | Column 0 | Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | |
|-------|---|---|---|---|---|---|---|---|------------|
| Row 0 |  |  |  |  |  |  |  |  | → 00111100 |
| Row 1 |  |  |  |  |  |  |  |  | → 01000010 |
| Row 2 |  |  |  |  |  |  |  |  | → 10100101 |
| Row 3 |  |  |  |  |  |  |  |  | → 10000001 |
| Row 4 |  |  |  |  |  |  |  |  | → 10100101 |
| Row 5 |  |  |  |  |  |  |  |  | → 10011001 |
| Row 6 |  |  |  |  |  |  |  |  | → 01000010 |
| Row 7 |  |  |  |  |  |  |  |  | → 00111100 |

Code

Here's a simple sketch that displays three types of faces: a sad face, a neutral face and a happy face. Upload the following code to your board:

```
/*
  Created by Rui Santos

  All the resources for this project:
  http://randomnerdtutorials.com/
*/

#include "LedControl.h"
#include "binary.h"

/*
  DIN connects to pin 12
  CLK connects to pin 11
  CS connects to pin 10
*/
LedControl lc=LedControl(12,11,10,1);

// delay time between faces
unsigned long delaytime=1000;

// happy face
byte hf[8]=
{B00111100,B01000010,B10100101,B10000001,B10100101,B10011001,B01000010,B00111100};
// neutral face
byte nf[8]={B00111100,
B01000010,B10100101,B10000001,B10111101,B10000001,B01000010,B00111100};
// sad face
byte sf[8]=
```

```

{B00111100,B01000010,B10100101,B10000001,B10011001,B10100101,B01000010,B00111100};

void setup() {
  lc.shutdown(0,false);
  // Set brightness to a medium value
  lc.setIntensity(0,8);
  // Clear the display
  lc.clearDisplay(0);
}

void drawFaces(){
  // Display sad face
  lc.setRow(0,0,sf[0]);
  lc.setRow(0,1,sf[1]);
  lc.setRow(0,2,sf[2]);
  lc.setRow(0,3,sf[3]);
  lc.setRow(0,4,sf[4]);
  lc.setRow(0,5,sf[5]);
  lc.setRow(0,6,sf[6]);
  lc.setRow(0,7,sf[7]);
  delay(delaytime);

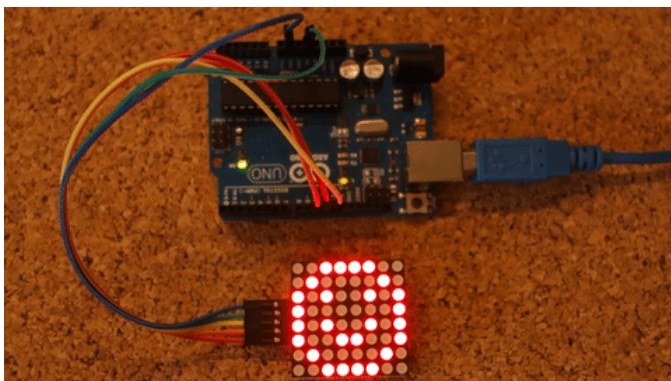
  // Display neutral face
  lc.setRow(0,0,nf[0]);
  lc.setRow(0,1,nf[1]);
  lc.setRow(0,2,nf[2]);
  lc.setRow(0,3,nf[3]);
  lc.setRow(0,4,nf[4]);
  lc.setRow(0,5,nf[5]);
  lc.setRow(0,6,nf[6]);
  lc.setRow(0,7,nf[7]);
  delay(delaytime);

  // Display happy face
  lc.setRow(0,0,hf[0]);
  lc.setRow(0,1,hf[1]);
  lc.setRow(0,2,hf[2]);
  lc.setRow(0,3,hf[3]);
  lc.setRow(0,4,hf[4]);
  lc.setRow(0,5,hf[5]);
  lc.setRow(0,6,hf[6]);
  lc.setRow(0,7,hf[7]);
  delay(delaytime);
}

void loop(){
  drawFaces();
}

```

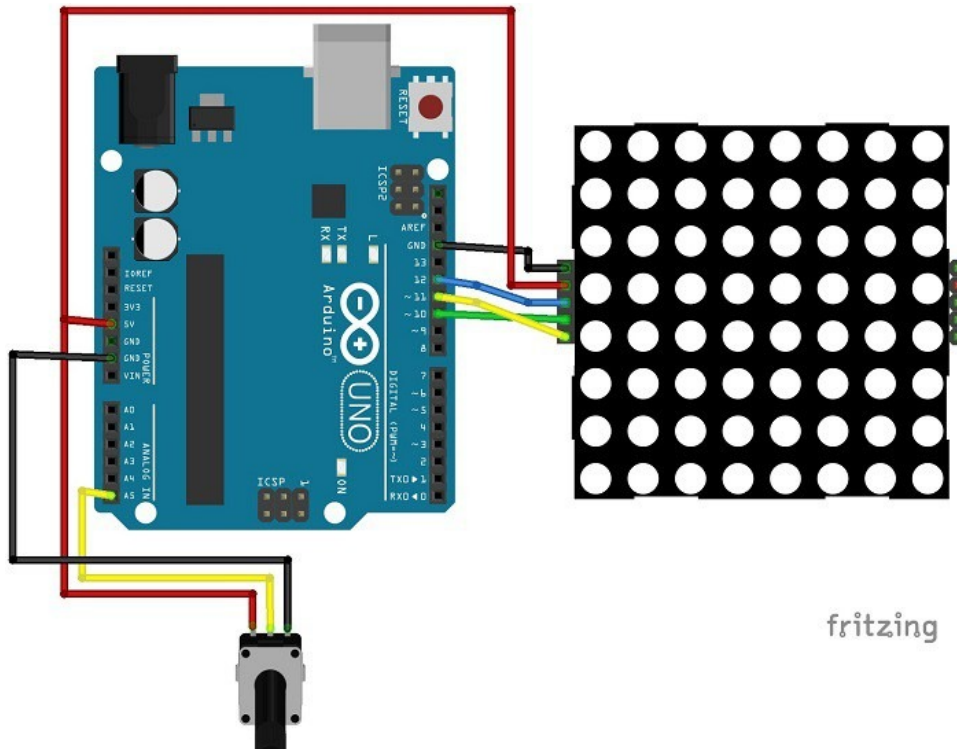
In the end, you'll have something like this:



Pong Game

The pong game that you're about to try was created by [Alessandro Pasotti](#).

For the pong game, you just need to add a 1k ohm potentiometer to the previous schematic. Assemble the new circuit as shown below:



Code

Then, upload the following code to your Arduino board:

```
/*
 *   Play pong on an 8x8 matrix - project from itopen.it
 */

#include "LedControl.h"
#include "Timer.h"

#define POTPIN A5 // Potentiometer
#define PADSIZ 3
#define BALL_DELAY 200
#define GAME_DELAY 10
#define BOUNCE_VERTICAL 1
#define BOUNCE_HORIZONTAL -1
#define NEW_GAME_ANIMATION_SPEED 50
#define HIT_NONE 0
#define HIT_CENTER 1
#define HIT_LEFT 2
#define HIT_RIGHT 3

// #define DEBUG 1

byte sad[] = {
  B00000000,
  B01000100,
  B00010000,
```

```

B00010000,
B00000000,
B00111000,
B01000100,
B00000000
};

byte smile[] = {
B00000000,
B01000100,
B00010000,
B00010000,
B00010000,
B01000100,
B00111000,
B00000000
};

Timer timer;

LedControl lc = LedControl(12,11,10,1);

byte direction; // Wind rose, 0 is north
int xball;
int yball;
int yball_prev;
byte xpad;
int ball_timer;

void setSprite(byte *sprite){
    for(int r = 0; r < 8; r++){
        lc.setRow(0, r, sprite[r]);
    }
}

void newGame() {
    lc.clearDisplay(0);
    // initial position
    xball = random(1, 7);
    yball = 1;
    direction = random(3, 6); // Go south
    for(int r = 0; r < 8; r++){
        for(int c = 0; c < 8; c++){
            lc.setLed(0, r, c, HIGH);
            delay(NEW_GAME_ANIMATION_SPEED);
        }
    }
    setSprite(smile);
    delay(1500);
    lc.clearDisplay(0);
}

void setPad() {
    xpad = map(analogRead(POTPIN), 0, 1020, 8 - PADSIZ, 0);
}

void debug(const char* desc){
#ifdef DEBUG
    Serial.print(desc);
    Serial.print(" XY: ");
    Serial.print(xball);
    Serial.print(", ");
    Serial.print(yball);
    Serial.print(" XPAD: ");
    Serial.print(xpad);
    Serial.print(" DIR: ");

```

```

    Serial.println(direction);
#endif
}

int checkBounce() {
    if(!xball || !yball || xball == 7 || yball == 6){
        int bounce = (yball == 0 || yball == 6) ? BOUNCE_HORIZONTAL :
BOUNCE_VERTICAL;
#ifdef DEBUG
        debug(bounce == BOUNCE_HORIZONTAL ? "HORIZONTAL" : "VERTICAL");
#endif
        return bounce;
    }
    return 0;
}

int getHit() {
    if(yball != 6 || xball < xpad || xball > xpad + PADSIZ){
        return HIT_NONE;
    }
    if(xball == xpad + PADSIZ / 2){
        return HIT_CENTER;
    }
    return xball < xpad + PADSIZ / 2 ? HIT_LEFT : HIT_RIGHT;
}

bool checkLoose() {
    return yball == 6 && getHit() == HIT_NONE;
}

void moveBall() {
    debug("MOVE");
    int bounce = checkBounce();
    if(bounce) {
        switch(direction){
            case 0:
                direction = 4;
                break;
            case 1:
                direction = (bounce == BOUNCE_VERTICAL) ? 7 : 3;
                break;
            case 2:
                direction = 6;
                break;
            case 6:
                direction = 2;
                break;
            case 7:
                direction = (bounce == BOUNCE_VERTICAL) ? 1 : 5;
                break;
            case 5:
                direction = (bounce == BOUNCE_VERTICAL) ? 3 : 7;
                break;
            case 3:
                direction = (bounce == BOUNCE_VERTICAL) ? 5 : 1;
                break;
            case 4:
                direction = 0;
                break;
        }
        debug("->");
    }

    // Check hit: modify direction is left or right
    switch(getHit()){
        case HIT_LEFT:

```

```

        if(direction == 0){
            direction = 7;
        } else if (direction == 1){
            direction = 0;
        }
        break;
    case HIT_RIGHT:
        if(direction == 0){
            direction = 1;
        } else if(direction == 7){
            direction = 0;
        }
        break;
}

// Check orthogonal directions and borders ...
if((direction == 0 && xball == 0) || (direction == 4 && xball == 7)){
    direction++;
}
if(direction == 0 && xball == 7){
    direction = 7;
}
if(direction == 4 && xball == 0){
    direction = 3;
}
if(direction == 2 && yball == 0){
    direction = 3;
}
if(direction == 2 && yball == 6){
    direction = 1;
}
if(direction == 6 && yball == 0){
    direction = 5;
}
if(direction == 6 && yball == 6){
    direction = 7;
}

// "Corner" case
if(xball == 0 && yball == 0){
    direction = 3;
}
if(xball == 0 && yball == 6){
    direction = 1;
}
if(xball == 7 && yball == 6){
    direction = 7;
}
if(xball == 7 && yball == 0){
    direction = 5;
}

yball_prev = yball;
if(2 < direction && direction < 6) {
    yball++;
} else if(direction != 6 && direction != 2) {
    yball--;
}
if(0 < direction && direction < 4) {
    xball++;
} else if(direction != 0 && direction != 4) {
    xball--;
}
xball = max(0, min(7, xball));
yball = max(0, min(6, yball));
debug("AFTER MOVE");

```



```

}

void gameOver() {
    setSprite(sad);
    delay(1500);
    lc.clearDisplay(0);
}

void drawGame() {
    if(yball_prev != yball){
        lc.setRow(0, yball_prev, 0);
    }
    lc.setRow(0, yball, byte(1 << (xball)));
    byte padmap = byte(0xFF >> (8 - PADSIZE) << xpad) ;
#ifdef DEBUG
    //Serial.println(padmap, BIN);
#endif
    lc.setRow(0, 7, padmap);
}

void setup() {
    // The MAX72XX is in power-saving mode on startup,
    // we have to do a wakeup call
    pinMode(POTPIN, INPUT);

    lc.shutdown(0,false);
    // Set the brightness to a medium values
    lc.setIntensity(0, 8);
    // and clear the display
    lc.clearDisplay(0);
    randomSeed(analogRead(0));
#ifdef DEBUG
    Serial.begin(9600);
    Serial.println("Pong");
#endif
    newGame();
    ball_timer = timer.every(BALL_DELAY, moveBall);
}

void loop() {
    timer.update();
    // Move pad
    setPad();
#ifdef DEBUG
    Serial.println(xpad);
#endif
    // Update screen
    drawGame();
    if(checkLoose()) {
        debug("LOOSE");
        gameOver();
        newGame();
    }
    delay(GAME_DELAY);
}

```

[view raw](#) Projects/dot_matrix_pong_game.ino

Demonstration

Here's the final demonstration of me playing the pong game. Have fun!