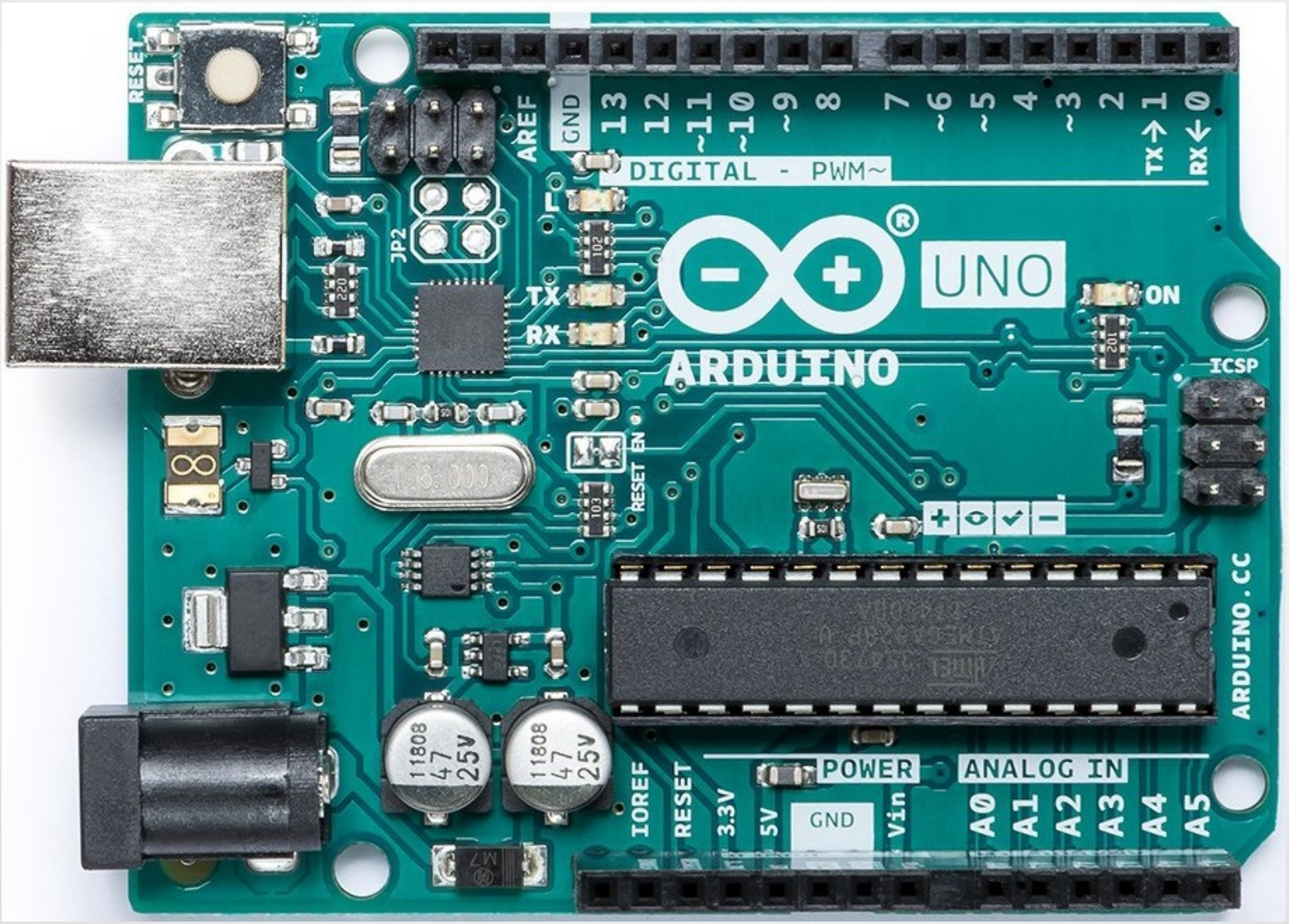
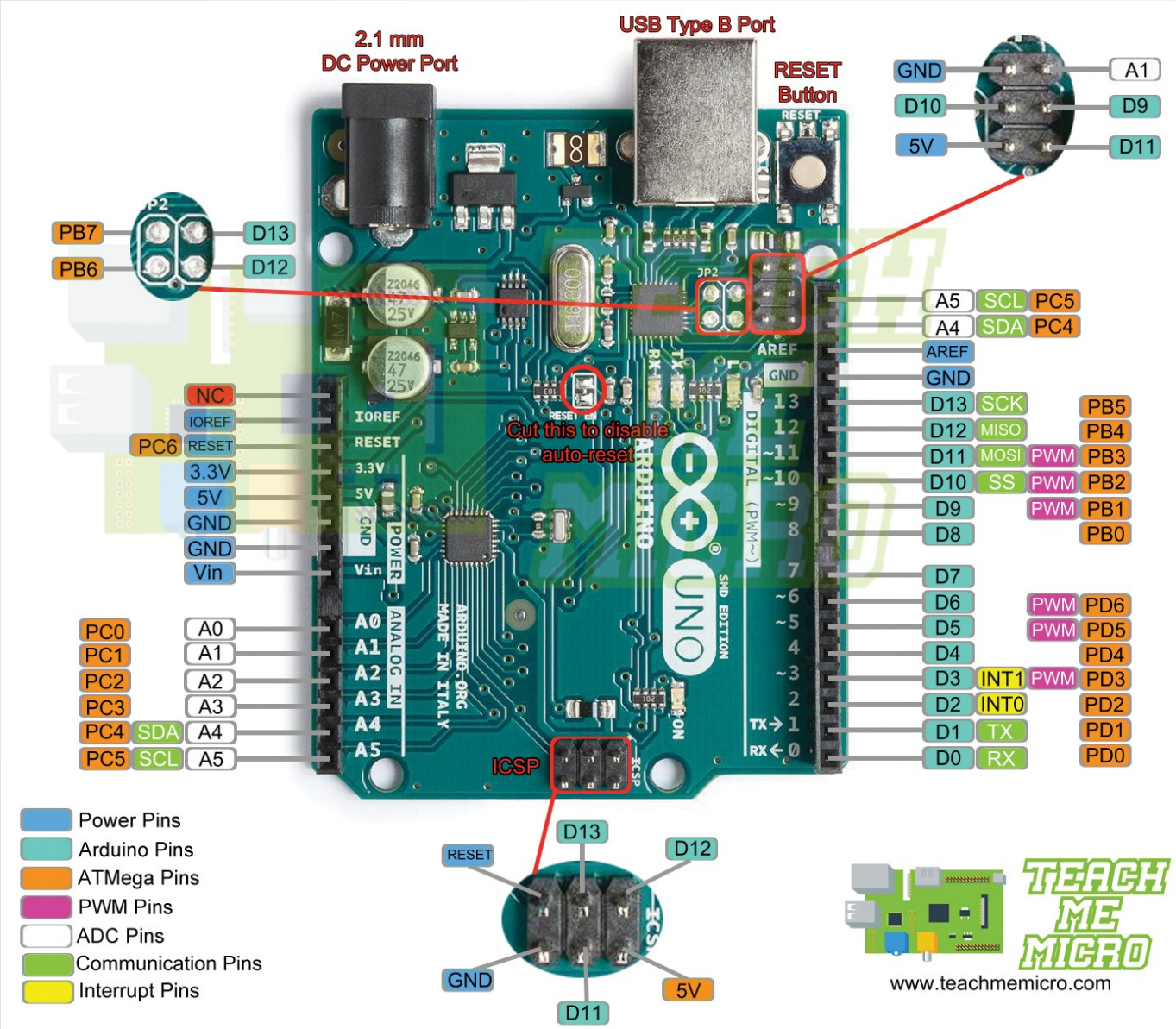
Matriz de LEDs 8x8



Víctor . Álvarez 201911

Matriz de LEDs 8x8 – pinout Arduino



E/S

Dixitais:

- Pins 0 a 13
  0 a 5V, 20 mA
  Low: 0 a 2V
  High: 3 a 5V

Entradas Analóxicas

- Pins A0 – A5
  0 a 5V, 20 mA prec 1024
  0 a 3V, 50 mA prec 1024

Saídas PWM

- ~ Pins 3, 5, 6, 9, 10, 11
  0 a 5 V, 20 mA prec 256

Comunic. Serie TX/RX

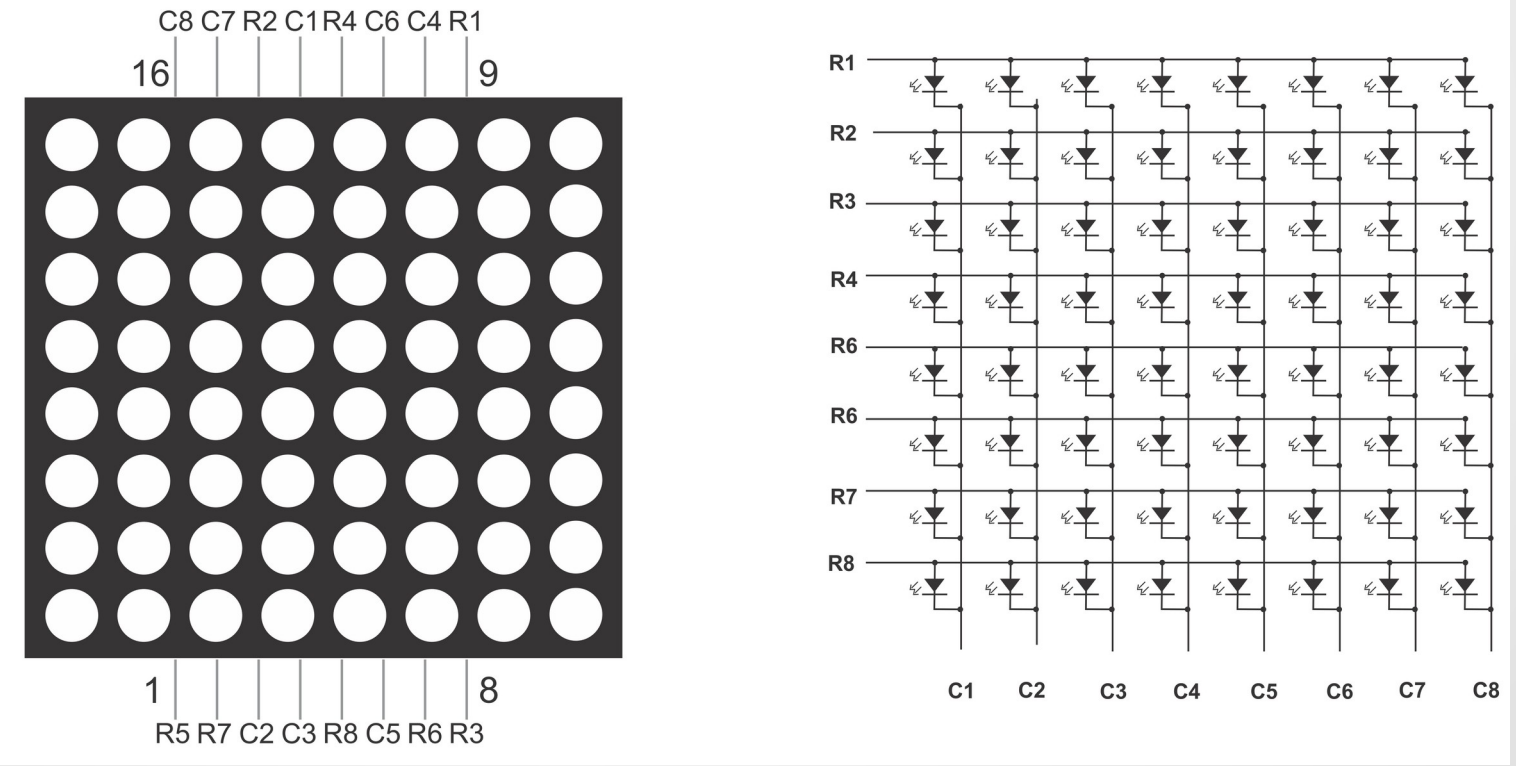
- Pins 0 e 1

ICSP

- 6 pins para comunicarse directamente co proc. Atmega328
- 6 pins para programar o USB

Matriz de LEDs 8x8

- A matriz de LEDs 8x8 é un array bidimensional de LEDs, que se iluminan ao activar a columna e fila correspondentes.
- A imaxe inferior mostra os pins e conexións internas do 1088AS.



Datasheet Matriz LEDs 8x8 1088AS

Víctor . Álvarez 201911

Matriz de LEDs 8x8

- Imos facer un script que mostre sucesivamente na matriz de LEDs a mensaxe:

  ‘OLA ARDUINO’

- Para isto primeiro temos que cablear correctamente a matriz. Como ten 16 pins, teremos que empregar todos os dixitais de Arduino (menos TX e RX), así como os analóxicos.
- Para poder averiguar rapidamente o patrón de bits de cada letra, podemos facer uso das seguintes páxinas:

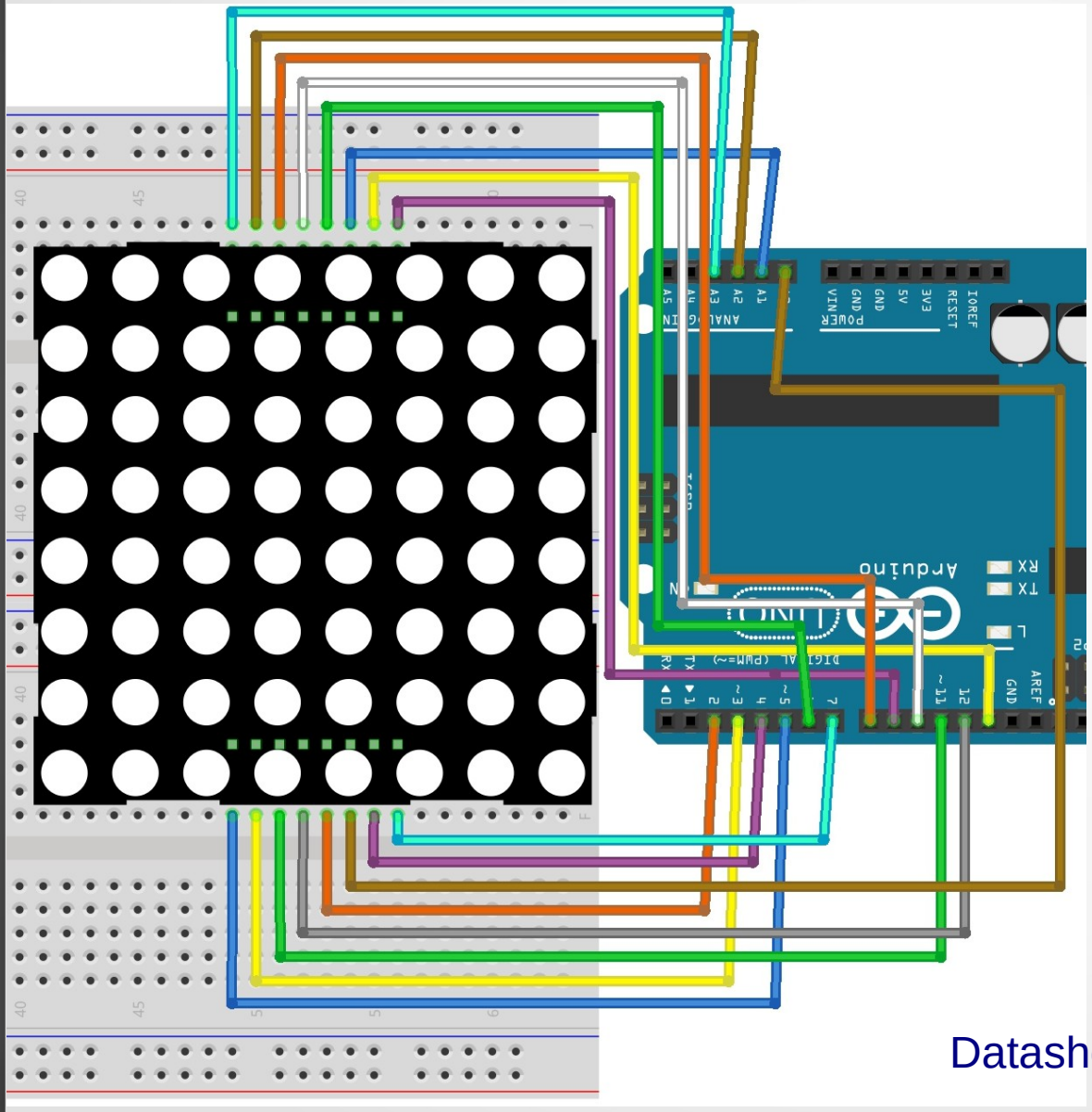
  http://robojax.com/learn/arduino/8x8LED/

  https://xantorohara.github.io/led-matrix-editor/

- Teremos que escribir repetidamente cada patrón de LEDs das letras, a fin de que sexan lexibles un tempo suficiente.

Víctor . Álvarez 201911

Matriz de LEDs 8x8

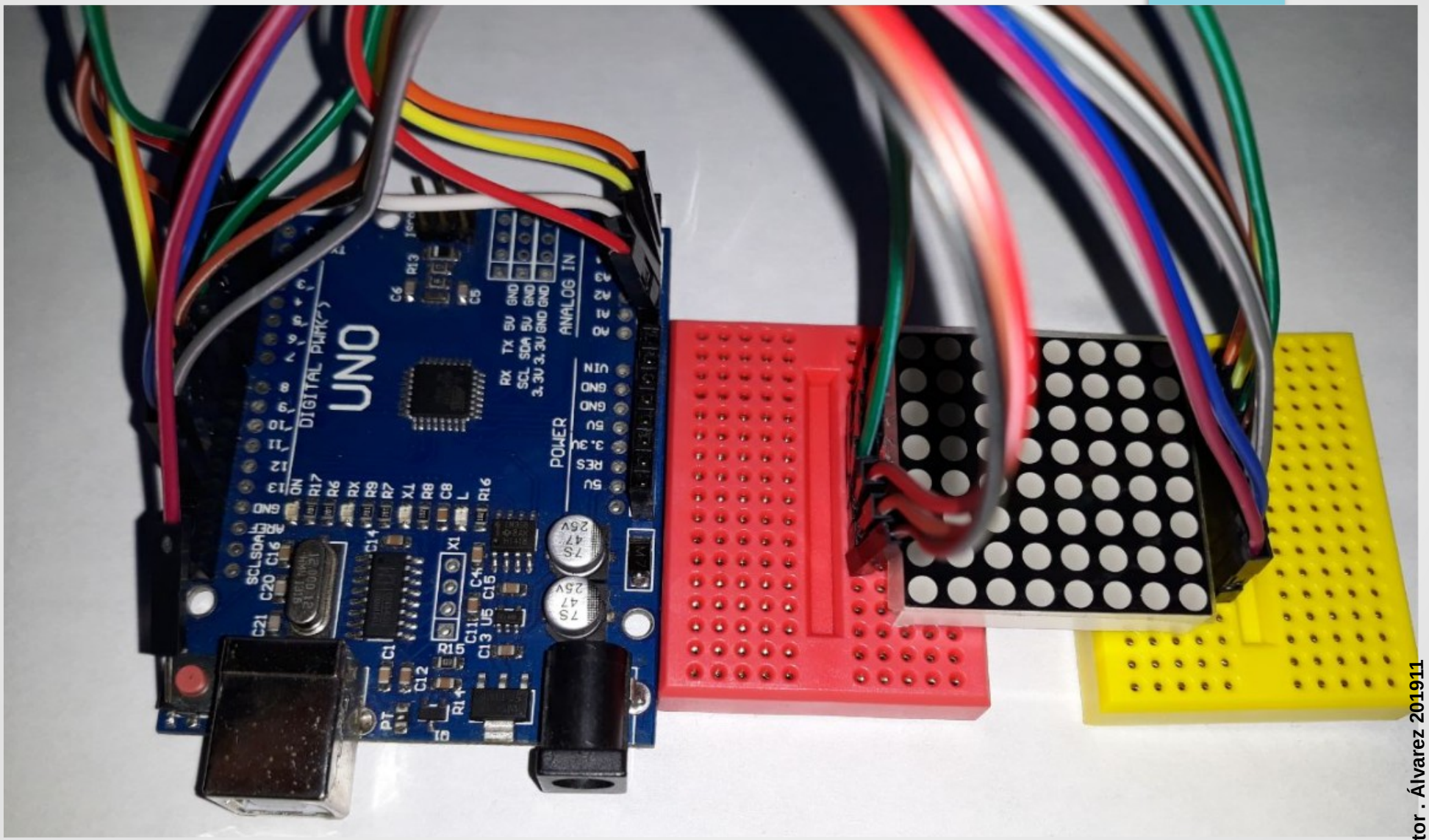


- Non existe este elemento emulado en TinkerCAD.
- Déixase a figura adxunta como referencia para o cableado correspondente ao script seguinte.

Datasheet Matriz LEDs 8x8 1088AS

Víctor . Álvarez 201911

Matriz de LEDs 8x8



Víctor . Álvarez 201911

Matriz de LEDs 8x8

matriz.8x8.leds

```
/*
 * Escreb a frase 'Ola Arduino'
 * na matriz de LEDs 8x8, como
 * exemplo de uso.
 */

//Definimos pins filas
//da matriz
#define F1 2
#define F2 3
#define F3 4
#define F4 5
#define F5 6
#define F6 7
#define F7 8
#define F8 9

//Definimos pins cols
//da matriz
#define C1 10
#define C2 11
#define C3 12
#define C4 13
#define C5 A0
#define C6 A1
#define C7 A2
#define C8 A3

//Arrays para percorrer as filas e columnos da matriz
const byte filas[] = {F1, F2, F3, F4, F5, F6, F7, F8};
const byte cols[] = {C1, C2, C3, C4, C5, C6, C7, C8};

//Patrons de bits de cada letra ou simbolo a representar
byte TODOS[] = {B11111111,B11111111,B11111111,B11111111,B111
byte NING[] = {B00000000,B00000000,B00000000,B00000000,B0000
byte COR1[] = {B00000000,B01100110,B11111111,B11111111,B1111
byte COR2[] = {B00000000,B01100110,B10011001,B10000001,B1000
byte A[] = {B00000000,B00111100,B01100110,B01100110,B0111111
byte B[] = {B01111000,B01001000,B01001000,B01110000,B0100100
byte C[] = {B00000000,B00011110,B00100000,B01000000,B0100000
byte D[] = {B00000000,B00111000,B00100100,B00100010,B0010001
byte E[] = {B00000000,B00111100,B00100000,B00111000,B0010000
```

matriz.8x8.leds §

```
//Arrays para percorrer as filas e columnos da matriz
const byte filas[] = {F1, F2, F3, F4, F5, F6, F7, F8};
const byte cols[] = {C1, C2, C3, C4, C5, C6, C7, C8};

//Patrons de bits de cada letra ou simbolo a representar
byte TODOS[] = {B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111,B11111111};
byte NING[] = {B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000};
byte COR1[] = {B00000000,B01100110,B11111111,B11111111,B11111111,B01111110,B00111100,B00011000};
byte COR2[] = {B00000000,B01100110,B10011001,B10000001,B10000001,B01000010,B00100100,B00011000};
byte A[] = {B00000000,B00111100,B01100110,B01100110,B01111110,B01100110,B01100110,B01100110};
byte B[] = {B01111000,B01001000,B01001000,B01110000,B01001000,B01000100,B01000100,B01111100};
byte C[] = {B00000000,B00011110,B00100000,B01000000,B01000000,B01000000,B00100000,B00011110};
byte D[] = {B00000000,B00111000,B00100100,B00100010,B00100010,B00100100,B00111000,B00000000};
byte E[] = {B00000000,B00111100,B00100000,B00111000,B00100000,B00100000,B00111100,B00000000};
byte F[] = {B00000000,B00111100,B00100000,B00111000,B00100000,B00100000,B00100000,B00000000};
byte G[] = {B00000000,B00111110,B00100000,B00100000,B00101110,B00100010,B00111110,B00000000};
byte H[] = {B00000000,B00100100,B00100100,B00111100,B00100100,B00100100,B00100100,B00000000};
byte I[] = {B00000000,B00111000,B00010000,B00010000,B00010000,B00010000,B00111000,B00000000};
byte J[] = {B00000000,B00011100,B00001000,B00001000,B00001000,B00101000,B00111000,B00000000};
byte K[] = {B00000000,B00100100,B00101000,B00110000,B00101000,B00100100,B00100100,B00000000};
byte L[] = {B00000000,B00100000,B00100000,B00100000,B00100000,B00100000,B00111100,B00000000};
byte M[] = {B00000000,B00000000,B01000100,B10101010,B10010010,B10000010,B10000010,B00000000};
byte N[] = {B00000000,B00100010,B00110010,B00101010,B00100110,B00100010,B00000000,B00000000};
byte O[] = {B00000000,B00111100,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000};
byte P[] = {B00000000,B00111000,B00100100,B00100100,B00111000,B00100000,B00100000,B00000000};
byte Q[] = {B00000000,B00111100,B01000010,B01000010,B01000010,B01000110,B00111110,B00000001};
byte R[] = {B00000000,B00111000,B00100100,B00100100,B00111000,B00100100,B00100100,B00000000};
byte S[] = {B00000000,B00111100,B00100000,B00111100,B00000100,B00000100,B00111100,B00000000};
byte T[] = {B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000};
byte U[] = {B00000000,B01000010,B01000010,B01000010,B01000010,B00100100,B00011000,B00000000};
byte V[] = {B00000000,B00100010,B00100010,B00100010,B00010100,B00010100,B00001000,B00000000};
byte W[] = {B00000000,B10000010,B10010010,B01010100,B01010100,B00101000,B00000000,B00000000};
byte X[] = {B00000000,B01000010,B00100100,B00011000,B00011000,B00100100,B01000010,B00000000};
byte Y[] = {B00000000,B01000100,B00101000,B00010000,B00010000,B00010000,B00010000,B00000000};
byte Z[] = {B00000000,B00111100,B00000100,B00001000,B00010000,B00100000,B00111100,B00000000};

float conta = 0;
int tempo = 0, tempoMensaxe = 700;
int reten = 10;

void setup() {
    Serial.begin(9600);
```

Víctor . Álvarez 201911

Matriz de LEDs 8x8

matriz.8x8.leds §

```
//Arrays para percorrer as filas e columnos d
const byte filas[] = {F1, F2, F3, F4, F5, F6,
const byte cols[] = {C1, C2, C3, C4, C5, C6,

//Patrons de bits de cada letra ou simbolo a
byte TODOS[] = {B11111111,B11111111,B11111111
byte NING[] = {B00000000,B00000000,B00000000,
byte COR1[] = {B00000000,B01100110,B11111111,
byte COR2[] = {B00000000,B01100110,B10011001,
byte A[] = {B00000000,B00111100,B01100110,B01
byte B[] = {B01111000,B01001000,B01001000,B01
byte C[] = {B00000000,B00011110,B00100000,B01
byte D[] = {B00000000,B00111000,B00100100,B00
byte E[] = {B00000000,B00111100,B00100000,B00
byte F[] = {B00000000,B00111100,B00100000,B00
byte G[] = {B00000000,B00111110,B00100000,B00
byte H[] = {B00000000,B00100100,B00100100,B00
byte I[] = {B00000000,B00111000,B00010000,B00
byte J[] = {B00000000,B00011100,B00001000,B00
byte K[] = {B00000000,B00100100,B00101000,B00
byte L[] = {B00000000,B00100000,B00100000,B00
byte M[] = {B00000000,B00000000,B01000100,B10
byte N[] = {B00000000,B00100010,B00110010,B00
byte O[] = {B00000000,B00111100,B01000010,B01
byte P[] = {B00000000,B00111000,B00100100,B00
byte Q[] = {B00000000,B00111100,B01000010,B01
byte R[] = {B00000000,B00111000,B00100100,B00
byte S[] = {B00000000,B00111100,B00100000,B00
byte T[] = {B00000000,B01111100,B00010000,B00
byte U[] = {B00000000,B01000010,B01000010,B01
byte V[] = {B00000000,B00100010,B00100010,B00
byte W[] = {B00000000,B10000010,B10010010,B01
byte X[] = {B00000000,B01000010,B00100100,B00
byte Y[] = {B00000000,B01000100,B00101000,B00
byte Z[] = {B00000000,B00111100,B00000100,B00

float conta = 0;
int tempo = 0, tempoMensaxe = 700;
int reten = 10;

void setup() {
    Serial.begin(9600);
```

matriz.8x8.leds §

```
float conta = 0;
int tempo = 0, tempoMensaxe = 700;
int reten = 10;

void setup() {
    Serial.begin(9600);
    //Declaramos como saidas incluso
    //os pins analoxicos
    for (byte i = 2; i <= 13; i++)
        pinMode(i, OUTPUT);
    pinMode(A0, OUTPUT);
    pinMode(A1, OUTPUT);
    pinMode(A2, OUTPUT);
    pinMode(A3, OUTPUT);
}

void loop() {
  //Ponhemos un tempo maximo para a mensaxe
  //que se vai mapear ao numero de letras
  conta++;
  if(conta > tempoMensaxe) {
    conta = 0;
  }
  tempo = map(conta, 0, tempoMensaxe, 0, 17);
  switch(tempo) {
    case  0: setMatriz(O); delay(reten); break;
    case  1: setMatriz(L); delay(reten); break;
    case  2: setMatriz(A); delay(reten); break;
    case  3: setMatriz(NING); delay(reten); break;
    case  4: setMatriz(NING); delay(reten); break;
    case  5: setMatriz(A); delay(reten); break;
    case  6: setMatriz(R); delay(reten); break;
    case  7: setMatriz(D); delay(reten); break;
    case  8: setMatriz(U); delay(reten); break;
    case  9: setMatriz(I); delay(reten); break;
    case 10: setMatriz(N); delay(reten); break;
    case 11: setMatriz(O); delay(reten); break;
    case 12: setMatriz(NING); delay(reten); break;
    case 13: setMatriz(COR1); delay(reten); break;
    case 14: setMatriz(NING); delay(reten); break;
    case 15: setMatriz(COR2); delay(reten); break;
    case 16: setMatriz(NING); delay(reten); break;
```

matriz.8x8.leds §

```
void loop() {
  //Ponhemos un tempo maximo para a mensaxe
  //que se vai mapear ao numero de letras
  conta++;
  if(conta > tempoMensaxe) {
    conta = 0;
  }
  tempo = map(conta, 0, tempoMensaxe, 0, 17);
  switch(tempo) {
    case  0: setMatriz(O); delay(reten); break;
    case  1: setMatriz(L); delay(reten); break;
    case  2: setMatriz(A); delay(reten); break;
    case  3: setMatriz(NING); delay(reten); break;
    case  4: setMatriz(NING); delay(reten); break;
    case  5: setMatriz(A); delay(reten); break;
    case  6: setMatriz(R); delay(reten); break;
    case  7: setMatriz(D); delay(reten); break;
    case  8: setMatriz(U); delay(reten); break;
    case  9: setMatriz(I); delay(reten); break;
    case 10: setMatriz(N); delay(reten); break;
    case 11: setMatriz(O); delay(reten); break;
    case 12: setMatriz(NING); delay(reten); break;
    case 13: setMatriz(COR1); delay(reten); break;
    case 14: setMatriz(NING); delay(reten); break;
    case 15: setMatriz(COR2); delay(reten); break;
    case 16: setMatriz(NING); delay(reten); break;
    case 17: setMatriz(NING); delay(reten); break;
  }
}

void setMatriz(byte texto[]) {
  for(byte i=0; i<8; i++) {
    digitalWrite(filas[i], HIGH);
    for(byte j=0; j<8; j++) {
      digitalWrite(cols[j], (~texto[i]>>j) &  0x01);
      delayMicroseconds(100);
      digitalWrite(cols[j], 1);
    }
    digitalWrite(filas[i], LOW);
  }
}
```

Víctor . Álvarez 201911

Matriz de LEDs 8x8

- Nesta unidade aprendemos a:
  - declarar e usar funcions definidas polo usuario,
  - cablear e usar unha matriz de LEDs 8x8,
  - definir arrays de byte e int,
  - usar e percorrer arrays de byte e int, en combinación cos bucles for(),
  - usar operadores de bit.

Víctor . Álvarez 201911