

# Videos

## Sesión 1

- concierto solenoide:  
[https://www.youtube.com/watch?v=g\\_hiz-Kx0kM&list=PL341E603FB13FE2D3&ab\\_channel=reduzent](https://www.youtube.com/watch?v=g_hiz-Kx0kM&list=PL341E603FB13FE2D3&ab_channel=reduzent)
- Processing & Kinect Sensor: Finger Tracking + OSCP5 + Pure Data + Arp2600  
[https://www.youtube.com/watch?v=X5WGyJPS\\_5s](https://www.youtube.com/watch?v=X5WGyJPS_5s)
- Experimento de resonancia acustica:  
[https://www.youtube.com/watch?v=vvJAgUBF4w&ab\\_channel=brusspup](https://www.youtube.com/watch?v=vvJAgUBF4w&ab_channel=brusspup)
- Graffiti Laser: <http://www.graffitiresearchlab.com/blog/projects/laser-tag/>
- Delicate boundaries: <http://csugrue.com/delicateboundaries/>
- Licuadora que funciona con gruñidos:<https://youtu.be/6DDkwdPaYmk>
- Almacenador de gritos: <https://youtu.be/Ta7rN5TeKzw>
- Drawdio: <https://www.youtube.com/watch?v=Ein9asQgfB8> y  
<https://www.youtube.com/watch?v=HYg8iycYZNs&t=2s>
- sensor pez: <https://twitter.com/unloquer/status/1099095464291508224>
- planta feliz: <https://www.youtube.com/shorts/YAsNlonSBac>
- juego de cubos en una caja :<https://www.youtube.com/shorts/YZ7aAG4DWZk>
- pelotica:<https://www.youtube.com/shorts/9eLff37aS4U>
- pez automata:<https://youtu.be/HH9IDf5W-gU>
- concierto de 8bits: <https://youtu.be/nE3JVpOwWuU>
- Lista de reproduccion:  
<https://www.youtube.com/playlist?list=PL36dJutVa9QZj95urcHqORXI1mH-VnsEN>
- galería de algunos sensores:<https://twitter.com/jero98772/status/1383031921484165120>
- multimedia de jugetes: [https://wiki.unloquer.org/personas/jero98772/taller\\_explora/multimedia](https://wiki.unloquer.org/personas/jero98772/taller_explora/multimedia)
- Taller de aire y vestuario: <https://www.flickr.com/photos/37012247@N06/49257858968/>

## Sesión 2

### Para generar ideas:

Sensores de calidad del aire y wearables: <https://youtu.be/8ZFxrDkVOFk>

## Sesión 3

### Para crear los conceptos:

Comprender el mundo a través de los datos:



### ¿Para que sirven los datos?

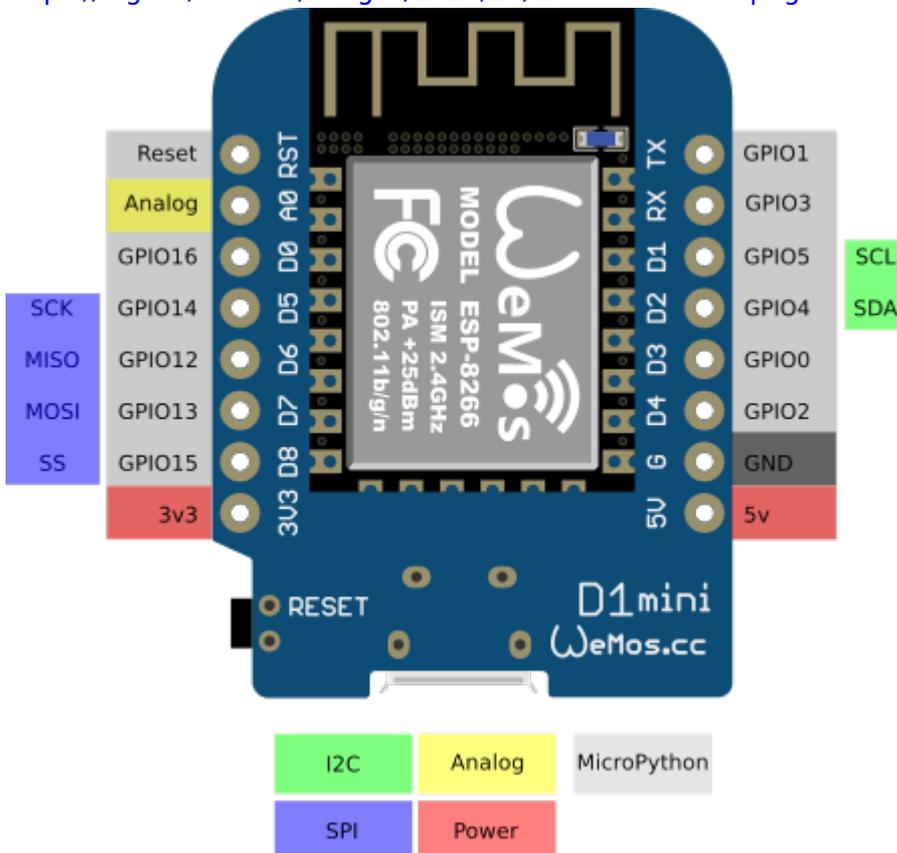
<https://www.behance.net/gallery/82992729/Dear-Data-Postcard>

[http://www.visualcomplexity.com/vc/project\\_details.cfm?id=666&index=27&domain=Music](http://www.visualcomplexity.com/vc/project_details.cfm?id=666&index=27&domain=Music)

## Componentes

### ESP 8266

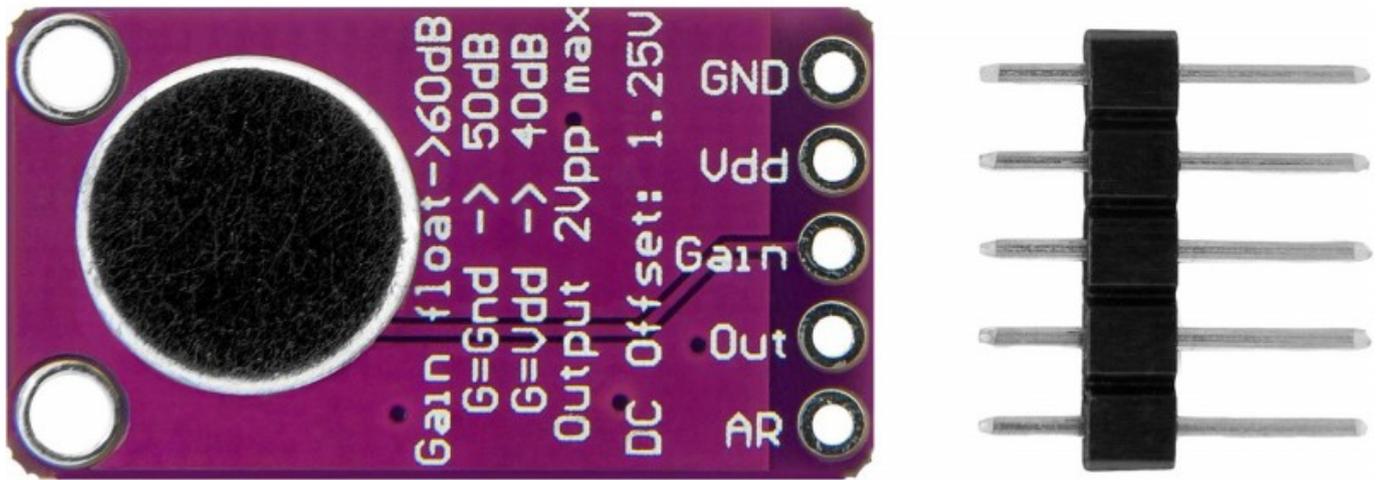
<https://bigl.es/content/images/2018/10/Wemos-D1-Mini.png>



# Micrófono

## Info del Micrófono

[https://maxelectronica.cl/3891-thickbox\\_default/modulo-max9814-sensor-de-sonido-microfono-electret-20-20khz.jpg](https://maxelectronica.cl/3891-thickbox_default/modulo-max9814-sensor-de-sonido-microfono-electret-20-20khz.jpg)



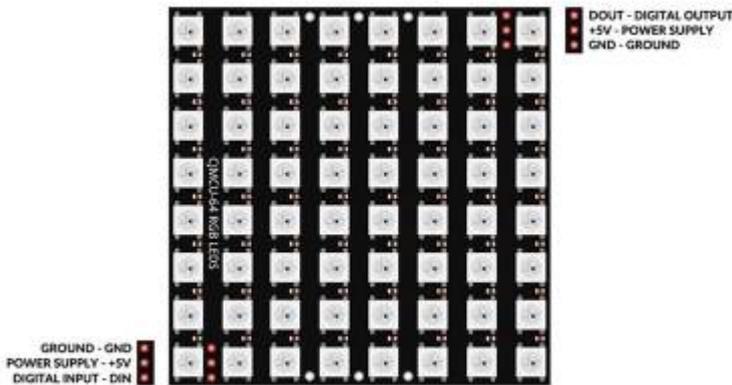
## Amplificador

<https://www.maximintegrated.com/en/products/analog/audio/MAX9814.html>

## Matriz de leds

<https://img.bestdealplus.com/ae04/kf/H0a03f4ded2694b5f8ec2324c68521a6b5.jpg>

U64 LED Matrix Panel module  
Pinout



## Aplicación Android

<https://play.google.com/store/apps/details?id=name.antonsmirnov.android.arduinoroid2&hl=es&gl=CO>

## Entorno para desktop

Windows: <https://www.arduino.cc/en/software>

## Github y Código

Repositorios de github: <https://github.com/unloquer/ETSesnor>

Repositorio del código sesión 2: intensidad

<https://github.com/unloquer/ETSesnor/blob/main/src/src.ino>

Descargar repositorio código para programar una "imagen" en la matriz

<https://github.com/unloquer/ETSesnor>

ejemplo.ino

```
#include <FastLED.h>
#define LED_PIN D3
#define LED_TYPE WS2812B
#define COLOR_ORDER GRB
#define amarillo CRGB::Yellow
#define negro CRGB::Black
#define rojo CRGB::Red
#define azul CRGB::Blue
#define maplv1 0x00FF00
#define maplv2 0x00AA00
#define maplv3 0xFFFF00
#define maplv4 0xFFE994
#define maplv5 0xFFAA00
#define maplv6 0xEC9BA4
#define maplv7 0xE1AA00
#define maplv8 0xFF00FF
#define maplv9 0x00DAFE
#define maplv10 0x0181FE

const uint8_t matrixWidth = 8;
const uint8_t matrixHeight = 8;
#define NUM_LEDS (matrixWidth * matrixHeight)

int BRIGHTNESS = 60;
CRGB leds[matrixWidth * matrixHeight];

int loop_cnt = 0;
const int sampleWindow = 50; // Sample window width in mS (50 mS =
20Hz)
unsigned int sample;

void setup() {
  Serial.begin(115200);
  LEDs.addLeds<LED_TYPE,LED_PIN,COLOR_ORDER>(leds,NUM_LEDS);
  FastLED.setBrightness(BRIGHTNESS);
}

#define ESCENAS 1
CRGB matrix[ESCENAS][8][8] = {
  {
    {azul, azul, azul, azul, azul, azul, azul, azul},
    {azul, azul, azul, azul, azul, azul, azul, azul},
  }
}
```

```

    {azul, azul, azul, azul, azul, azul, azul, azul},
    {azul, azul, azul, azul, azul, azul, azul, azul},
  },
};

void loop() {
  for(int i = 0; i < matrixHeight; i++) {
    for(int j = 0; j < matrixWidth; j++) {
      leds[i*matrixWidth + j] = matrix[loop_cnt%ESCENAS][i][j];
    }
  }
  unsigned long startMillis = millis(); // Start of sample window
  unsigned int peakToPeak = 0;

  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;

  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(0);
    if (sample < 1024) {
      if (sample > signalMax)
      {
        signalMax = sample;
      }
      else if (sample < signalMin)
      {
        signalMin = sample;
      }
    }
  }
  peakToPeak = signalMax - signalMin;
  int changeBrightness = map(peakToPeak, 20, 500, 0, 100);
  FastLED.setBrightness(changeBrightness);
  FastLED.show();
  Serial.println(peakToPeak);
  loop_cnt++;
  FastLED.clear();
}

```

```

<file c++ matrixlib.ino> #include <Arduino.h> #include <FastLED.h> #include <algorithm>
#define LED_PIN D1 #define LED_TYPE WS2812B #define COLOR_ORDER GRB

```

```

/* https://github.com/FastLED/ https://github.com/gmoehrke/FastFX

```

```

https://www.reddit.com/r/FastLED/wiki/index/user\_examples

```

```

https://macetech.github.io/FastLED-XY-Map-Generator/ */

```

```

#define amarillo CRGB::Yellow #define negro CRGB::Black #define rojo CRGB::Red #define azul

```

```
CRGB::Blue #define morado CRGB::Purple #define naranja CRGB::OrangeRed #define verde
CRGB::Green #define aguamarina CRGB::Cyan #define rosado CRGB::Fuchsia #define verdedos
CRGB::LimeGreen #define raro CRGB::DarkOrchid #define rarodos CRGB::DeepPink #define maplv1
0x00FF00 #define maplv2 0x00AA00 #define maplv3 0xFFFF00 #define maplv4 0xFFE994 #define
maplv5 0xFFAA00 #define maplv6 0xEC9BA4 #define maplv7 0xE1AA00 #define maplv8 0xFF00FF
#define maplv9 0x00DAFE #define maplv10 0x0181FE
```

```
const uint8_t matrixWidth = 8; const uint8_t matrixHeight = 8; #define NUM_LEDS (matrixWidth *
matrixHeight)
```

```
int BRIGHTNESS = 50; CRGB leds[matrixWidth * matrixHeight];
```

```
int animator = 0; const int sampleWindow = 50; Sample window width in mS (50 mS = 20Hz)
unsigned int sample; int inputMin = -2; int inputMax = 110; int rangeMin = 20; int rangeMiddleMin =
21; int rangeMiddleMax = 60; int rangeMax = 61; CRGBPalette16 currentPalette = OceanColors_p;
CRGBPalette16 targetPalette = OceanColors_p; TBlendType currentBlending = LINEARBLEND;
NOBLEND or LINEARBLEND
```

```
class Matrix {
```

```
CRGB *leds = NULL;
uint8_t numLeds = 0;
uint8_t currBrightness = 0;
```

```
public:
```

```
Matrix(CRGB *initLeds, uint8_t initNum) {
    leds = initLeds;
    numLeds = initNum;
}
```

```
// fill all matrix with same color
void fill(CRGB color) {
    for (int i = 0; i < 64; i++) {
        leds[i] = color;
    }
};
// fill a binary shape with same color
// {
// B00000000,
// B10101010,
// B00000000,
// B10101010,
// B00000000,
// B10101010,
// B00000000,
// B10101010
// };
void fill(CRGB color, byte *shape) {
    for (int i = 0; i < matrixHeight; i++) {
        for (int j = 0; j < matrixWidth; j++) {
```

```

        if (shape[i] & 1 << j) { // if bitwise AND resolves to
            leds[i * matrixHeight + j] = color; // send 1
        }
    }
};
// fill color shape
void fill(CRGB color_shape[][8]) {
    for (int i = 0; i < matrixHeight; i++) {
        for (int j = 0; j < matrixWidth; j++) {
            leds[i * matrixHeight + j] = color_shape[i][j]; // send 1
        }
    }
};
// fill row with color
void fill_y(CRGB color, int row) {
    for (int j = 0; j < matrixWidth; j++) {
        leds[row * matrixHeight + j] = color; // send 1
    }
};
// fille column with color
void fill_x(CRGB color, int column) {
    for (int i = 0; i < matrixHeight; i++) {
        leds[i * matrixHeight + column] = color; // send 1
    }
};
// of array in matrix
int sound_scale(); // return the actual level of sound intensity
void color_scale(int sl); // return color for 10 levels of sound intensity

```

```

};

Matrix *mym;

```

```

void setup() {

```

```

    Serial.begin(115200);
    LEDS.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS);
    mym = new Matrix(leds, 64);
    FastLED.setBrightness(BRIGHTNESS);

```

```

}

```

```

byte sshape[8] = {B00000000, B10101010, B00000000, B10101010,

```

```

                B00000000, B10101010, B00000000, B10101010};

```

```

CRGB matrix[8][8] = {

```

```

    {maplv2, maplv2, negro, negro, negro, negro, negro, negro},
    {maplv3, maplv3, maplv3, negro, negro, negro, negro, negro},

```

```
{maplv4, maplv4, maplv4, maplv4, negro, negro, negro, negro},
{maplv5, maplv5, maplv5, maplv5, maplv5, negro, negro, negro},
{maplv6, maplv6, maplv6, maplv6, maplv6, maplv6, negro, negro},
{maplv7, maplv7, maplv7, maplv7, maplv7, maplv7, maplv7, negro},
{maplv8, maplv8, maplv8, maplv8, maplv8, maplv8, maplv8, maplv8},
{maplv9, maplv9, maplv9, maplv9, maplv9, maplv9, maplv9, maplv9},
```

```
};
```

```
unsigned int sample_sound() {
```

```
    unsigned long startMillis = millis(); // Start of sample window
    unsigned int peakToPeak = 0;
```

```
    unsigned int signalMax = 0;
    unsigned int signalMin = 1024;
```

```
    // collect data for 50 mS
    while (millis() - startMillis < sampleWindow) {
        sample = analogRead(0);
        if (sample < 1024) {
            if (sample > signalMax) {
                signalMax = sample;
            } else if (sample < signalMin) {
                signalMin = sample;
            }
        }
    }
}
```

```
    peakToPeak = signalMax - signalMin;
```

```
    return peakToPeak;
```

```
}
```

```
void loop() {
```

```
    // int sample = sample_sound();
    // int maplv = map(sample, inputMin, inputMax, 0, 10);
```

```
    // EVERY_N_SECONDS(5) {
    //     for (int i = 0; i < 16; i++) {
    //         targetPalette[i] = CHSV(random8(), 255, 255);
    //     }
    // }
```

```
    // mym->fill(azul);
    // FastLED.show();
    // delay(1000);
    // mym->fill(rojo);
    // FastLED.show();
```

```
// delay(1000);  
// mym->fill(maplv5);  
// FastLED.show();  
// delay(1000);  
// mym->fill(maplv6);  
// FastLED.show();  
// delay(1000);  
// mym->fill(maplv8);  
// FastLED.show();  
// FastLED.delay(1000);  
// mym->fill(maplv10);  
// FastLED.show();  
// FastLED.delay(1000);  
// mym->fill(matrix);  
// FastLED.show();  
// FastLED.delay(1000);  
// mym->fill(verde);  
// mym->fill(raro, sshape);  
// FastLED.show();  
// FastLED.delay(1000);  
FastLED.clear();  
mym->fill_x(maplv1, 0);  
mym->fill_y(maplv1, 0);  
FastLED.show();  
FastLED.delay(50);
```

```
FastLED.clear();  
mym->fill_x(maplv2, 1);  
mym->fill_y(maplv2, 1);  
FastLED.show();  
FastLED.delay(50);
```

```
FastLED.clear();  
mym->fill_x(maplv3, 2);  
mym->fill_y(maplv3, 2);  
FastLED.show();  
FastLED.delay(50);
```

```
FastLED.clear();  
mym->fill_x(maplv4, 3);  
mym->fill_y(maplv4, 3);  
FastLED.show();  
FastLED.delay(50);
```

```
FastLED.clear();  
mym->fill_x(maplv5, 4);  
mym->fill_y(maplv5, 4);  
FastLED.show();  
FastLED.delay(50);
```

```
FastLED.clear();
```

```
mym->fill_x(maplv6, 5);
mym->fill_y(maplv6, 5);
FastLED.show();
FastLED.delay(50);
```

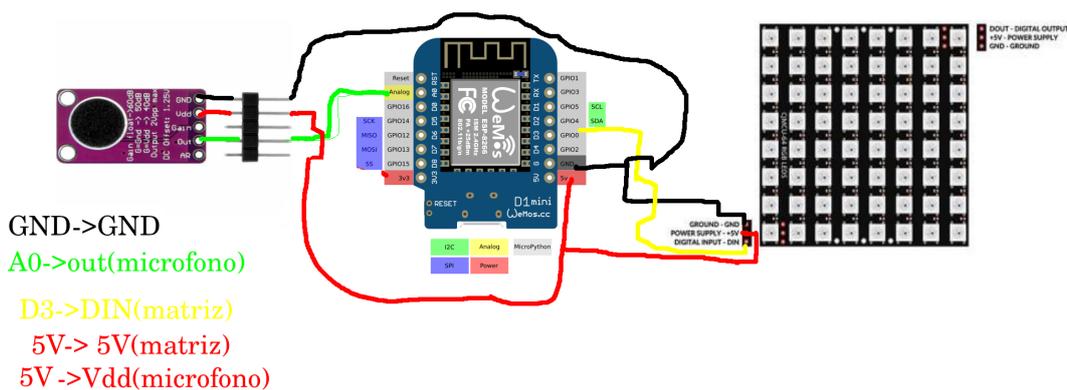
```
FastLED.clear();
mym->fill_x(maplv8, 6);
mym->fill_y(maplv8, 6);
FastLED.show();
FastLED.delay(50);
```

```
FastLED.clear();
mym->fill_x(maplv10, 7);
mym->fill_y(maplv10, 7);
FastLED.show();
FastLED.delay(50);
FastLED.clear();
```

```
// Serial.println(maplv);
FastLED.clear();
```

} </filename>

### Diagrama electronico



Descargar repositorio codigo para programar varios estados en la matriz

# Referentes

- [https://www.reddit.com/r/FastLED/wiki/index/user\\_examples](https://www.reddit.com/r/FastLED/wiki/index/user_examples)

# Contacto

<https://t.me/unloquer>

From:

<https://wiki.unloquer.org/> -

Permanent link:

<https://wiki.unloquer.org/proyectos/talleres/ets/recursos?rev=1638389414>

Last update: **2021/12/01 20:10**

