

# Wio Terminal<sub>D51R</sub>

## Manual de Usuario





## Cuidados y advertencias

Antes de empezar a trabajar con la Wio Terminal, debe de tomar en cuenta diversos puntos que evitarán que el dispositivo se dañe:

- Mantén tu área de trabajo limpia
- Evita tener vasos o bebidas que se puedan derramar, si no se puede evitar, asegurar que estén tapados
- Usa fuentes de alimentación estables y seguras, verifica que los cables se encuentren en buen estado
- Si hay cerca mascotas y/o niños cerca, asegurar que no estén cerca del área de trabajo
- Asegúrate que la superficie de trabajo no sea metálica, puede producir cortos que dañen el equipo

Las siguientes son prácticas que pueden dañar de forma irreversible el Wio Terminal

- Unir los cables de pines o algún conductor, por ejemplo, aquellos pines marcados como 5V, 3.3V, GND o VIN. Esto puede dañar el microcontrolador o el conversor USB-Serie y dejar a el Wio Terminal inservible
  - Para ello revisa siempre las conexiones hechas antes de energizar. De ser posible, usar un multímetro para medir la conductividad entre la tierra y otros pines para verificar que no existan cortos.
- Sobre corriente, si piensas conectar el Wio Terminal a cualquier fuente de alimentación con un adaptador de corriente del celular, puedes dañar el microcontrolador.
  - Para ello revisa el voltaje de la fuente de alimentación, en el caso del Wio Terminal, el voltaje recomendado son 5V y los amperes recomendados son 1000 mA o 1 A; los cuales son equivalentes a 5W. Dichos datos los puedes obtener en el manual del cargador.

# Visión del Wio Terminal

## Características Destacables

Los puntos destacables de la Wio Terminal:

- **Diseño altamente integrado**
  - MCU (MicroController Unit), LCD, WIFI, BT, IMU (Inertial Measurement Unit), Micrófono, Altavoz, Tarjeta Micro SD, Sensor de Luz, Interruptor de 5 posiciones (Mini Joystick), Emisor de Infrarrojos (IR 940nm), Cripto-autenticación Listo
- Desarrollado por Microchip ATSAMD51P19
  - Núcleo ARM Cortex-M4F funcionando a **120 MHz** (hasta 200 MHz)
  - Flash externo de **4 MB**, RAM de **192 KB**
- Compatibilidad Integral de Protocolos
  - SPI(Serial Peripheral Interface), I2C(Inter-Integrated Circuit), I2S(Integrated Interchip Sound), ADC(Advanced Direct Connect), DAC(Discretionary Access Control), PWM(Pulse-width modulation), UART(Universal Asynchronous Receiver/Transmitter - Serial)
- Potente Conectividad Inalámbrica (Soportado solo por Arduino)
  - Desarrollado por **Realtek RTL8720DN**
  - Wi-Fi de doble banda de 2,4 GHz/5 GHz (802.11 a/b/g/n)
  - Bluetooth de Baja Energía 5.0
- **Soporte USB OTG**
  - Puerto USB
  - Cliente USB
- Ecosistema Grove (de seeed studio)
- Soporte de Software
  - Arduino
  - MicroPython
  - ArduPy
  - AT Firmware

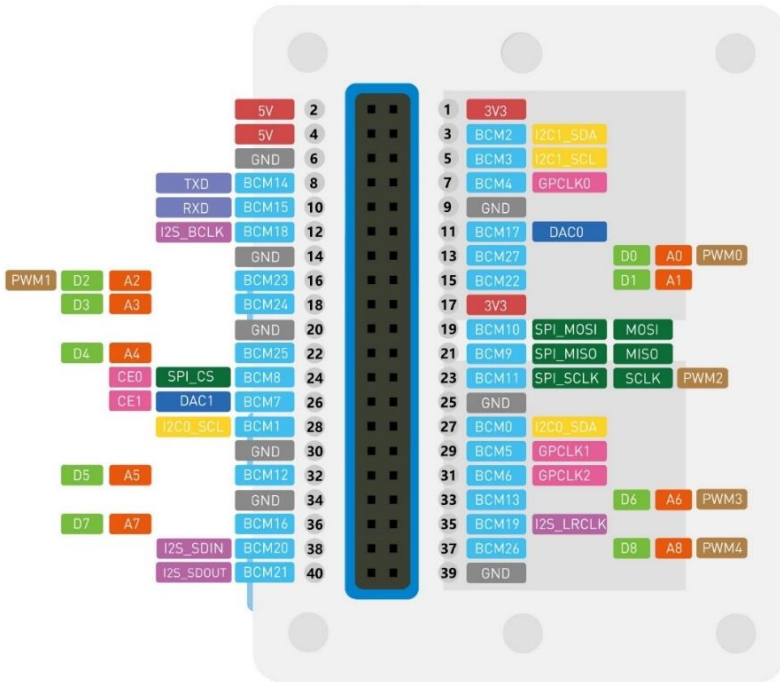
## Especificaciones

Chip Principal	Número de pieza del fabricante	ATSAMD51P19
	Procesador Central	ARM® Cortex®-M4F
	Velocidad de la CPU	120 MHz (hasta 200 MHz)
	Tamaño de la memoria del programa	512KB
	Memoria Externa	4MB
	Memoria RAM	192KB
	Temperatura de Funcionamiento	-40°C ~ 85°C (TA)
Pantalla LCD	Resolución	320 x 240
	Tamaño de Pantalla	2.4 in. (6.096 cm.)
	Driver IC	ILI9341
Conectividad Inalámbrica	Número de pieza del fabricante	RTL8720DN
	KM4 CPU	ARM® Cortex®-M4F
	KM0 CPU	ARM® Cortex®-M0
	Wi-Fi	802.11 a/b/g/n 1x1, 2.4GHz & 5GHz
	Bluetooth	Soporte BLE5.0
	Motor de Hardware	AES/DES/SHA
Módulos Incorporados	Acelerómetro	LIS3DHTR
	Micrófono	1.0V-10V -42dB
	Altavoz	≥78dB @10cm 4000Hz
	Sensor de Luz	400-1050nm
	Emisor de Infrarrojos	940nm
Interfaz	Ranura para tarjeta microSD	Máximo 16 GB
	GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General)	40 Pines (Compatible con Raspberry Pi)
	Grove	2 (Multifunción)
	FPC	20 Pines

	USB Tipo C	Alimentación y USB-OTG
	Interruptor de 5 posiciones (Mini Joystick)	
	Switch de Encendido/Reseteo	
	Botones definidos por el Usuario * 3	
Caja	Dimensiones	72mm*57mm*12mm
	Materiales	ABS + PC

## Vistazo Interno a la Wio Terminal

El Wio Terminal cuenta con 40 pines, y es compatible compartiendo compatibilidad con la Raspberry Pi que también cuenta con 40 pines.



**4 pines de alimentación:** 2 pines de 3.3V y 2 pines de 5V

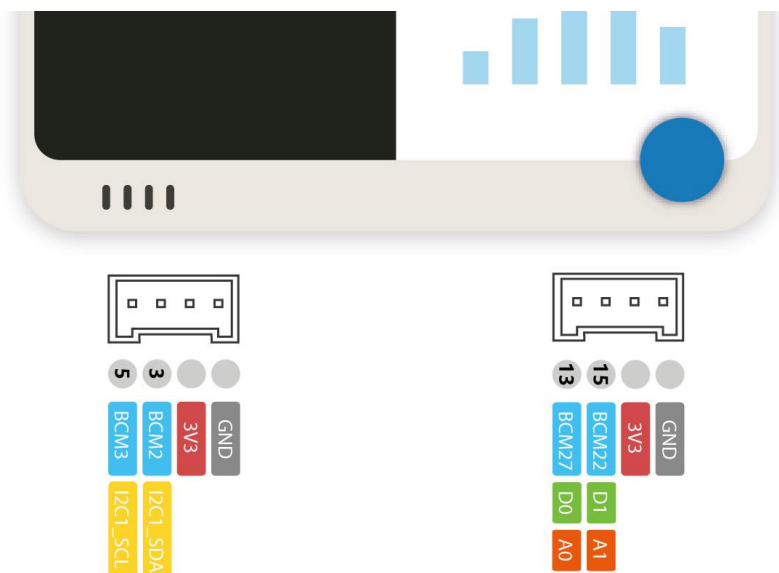
**8 pines de tierra**

**28 pines BCM#(Broadcom - Semejante al GPIO):** # indica el número de pin el cual puede ser utilizado en el software.

- **I2C#\_\*\*\*:** I2C es un protocolo donde los datos se transfieren poco a poco a lo largo de un solo cable. # puede ser 0 o 1.
- **2 pines I2C#\_SDA:** El SDA es la línea para que el maestro y el esclavo envíen y reciban datos.
- **2 pines I2C#\_SCL:** El SCL es la línea que lleva la señal del reloj.

- **I2S\_\*\*\*\***: I2S es un estándar que permite usar pines para conectar dispositivos de audio digitales.
- Pines **I2S\_SDIN** o **I2S\_SDOUT**: Línea de datos, impulsada por uno de los esclavos o maestros dependiendo de la dirección de datos. Puede haber varias líneas de datos en diferentes direcciones.
- Pin **I2S\_BCLK**: Reloj de bits. Esta es una división fija del MCLK y es impulsada por el maestro.
- Pin **I2S\_LRCLK**: Reloj de palabras (o selección de palabras). Esto es impulsado por el maestro.
- **SPI\_\*\*\***: La interfaz periférica en serie (SPI) es una de las interfaces más utilizadas entre el microcontrolador y los circuitos integrados periféricos, como sensores, ADC, DAC, registros de desplazamiento, SRAM y otros.
- Pin **SPI\_SCLK**: Reloj.
- Pin **SPI\_CS**: Chip seleccionado.
- Pin **SPI\_MOSI**: Salida principal, entrada de subnodo.
- Pin **SPI\_MISO**: Entrada principal, salida de subnodo.
- 3 Pines **GPCLK#**: Reloj de propósito general. Los pines de reloj de uso general se pueden configurar para generar una frecuencia fija sin ningún control de software continuo. # puede ser del 0 al 2.
- 2 Pines **DAC#**: Convertidor de Digital a analógico. # puede ser 0 o 1.
- 2 Pines **CE#**: Permite la comunicación con un dispositivo SPI, donde se requiere encender el pin de Chip Select correspondiente al chip con estos pines. # puede ser 0 o 1.
- 9 pines digitales **D#**: Pines digitales de entrada y salida. # puede ser del 0 al 8.
- 9 pines analógicos **A#**: Pines digitales de entrada. # puede ser del 0 al 8.
- 5 pines **PMW#**: PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. # puede ser del 0 al 4.





Puertos compatibles con accesorios Grove de Seedstudio

**2 pines de alimentación de 3.3V.**

**2 pines de tierra.**

**4 pines BCM#(Broadcom - Semejante al GPIO):** # indica el número de pin el cual puede ser utilizado en el software.

- **I2C1\_\*\*\*:** I2C es un protocolo donde los datos se transfieren poco a poco a lo largo de un solo cable. # puede ser 0 o 1.
- **1 pin I2C1\_SDA:** El SDA es la línea para que el maestro y el esclavo envíen y reciban datos.
- **1 pin I2C1\_SLC:** El SDA es la línea que lleva la señal del reloj.
- **2 pines digitales D#:** Pines digitales de entrada y salida. # puede ser del 0 o 1.
- **2 pines analógicos A#:** Pines digitales de entrada. # puede ser del 0 o 1.

## Dentro de la caja

Dentro de la caja del Wio Terminal (D51R) tenemos los siguientes elementos:

- 1 Wio Terminal (D51R)
- 1 cable USB-A a USB-C de 6.5 in. (16 cm.)
- 1 guía rápida del Wio Terminal (D51R) en Inglés, Japonés y Alemán
- 1 repuesto del mini joystick
- 1 juego de calcomanías

## Conceptos y términos

Con el fin de evitar conflictos y confusiones, definiremos algunas cosas:

- Wio Terminal: Dispositivo desarrollado por Seeedstudio, donde se estarán ejecutando las actividades y prácticas que el usuario desee desarrollar.
- Arduino IDE (Software): Se refiere al programa desarrollado por una organización sin fines de lucro, el cual funciona como el entorno de programación para crear aplicaciones para las placas Arduino y otras que sean de Hardware libre.
- Arduino (Lenguaje): Es el lenguaje basado en C con el cual, junto al IDE Arduino podremos desarrollar distintos proyectos.
- Micro Python: Es el resultado de la implementación del lenguaje Python 3, optimizado para la ejecución en microcontroladores, como el que contiene el Wio Terminal.
- Ardupy: Es la combinación de Arduino y Micro Python desarrollado por Seeed, para quienes deseen realizar proyectos complejos, la solución fue la combinación de ambos lenguajes.

## Antes de empezar

Lo que se requiere para trabajar es lo siguiente:

### Hardware

- 1 Wio Terminal (D51R)
- 1 cable USB-A a USB-C
- 1 equipo de computo

*Nota: En la elaboración de este manual estaremos trabajando sobre el Sistema Operativo Windows 10*

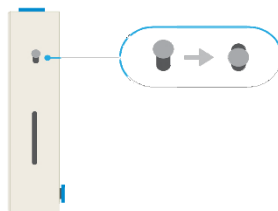
Conecta la Wio Terminal a la computadora mediante el cable USB. El LED azul de la parte inferior debe encender.

### Lo básico antes de continuar

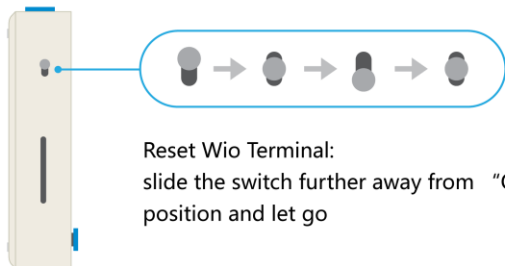
Hay que tomar en cuenta que aquí ya no hay marcha atrás, pero antes de continuar a pulsar código, debemos de saber cómo funcionará el Wio Terminal.

#### Para encender

La posición del Switch que se encuentra del lado izquierda debe de estar en el centro, de esta forma podremos compilar lo que se ejecuta desde Arduino, y ejecutar lo que tenga cargado desde Arduino o Ardupy.



#### Para reiniciar

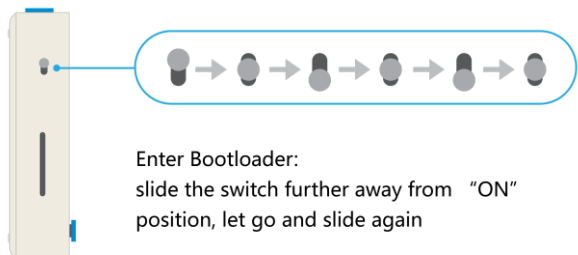


Reset Wio Terminal:  
slide the switch further away from "ON"  
position and let go

Vamos a la posición de Encendido (en caso de no estarlo) y damos un paso para abajo, y soltamos

## Para acceder al Bootloader

Esto es útil cuando Wio Terminal falla, cuando el USB serial no aparece en Arduino IDE u otras ocasiones.



Para entrar al bootloader nos dirigimos a la posición de encendido y después, de forma rápida bajamos dos veces, tomando en cuenta que el dial

regresará a su posición de encendido.

El indicador de que el modo bootloader está en funcionamiento, es cuando el LED azul esté parpadeando, pero más lento y con atenuación.

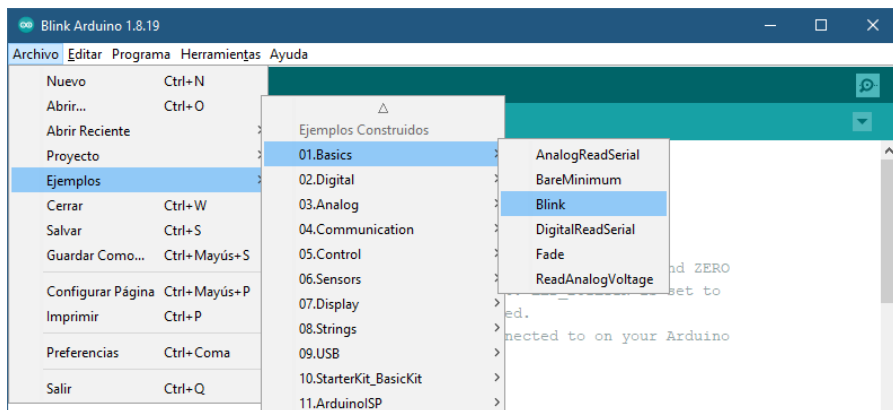
## Software

### Iniciando en Arduino

Paso 1. Necesitas instalar el software Arduino del siguiente enlace:

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

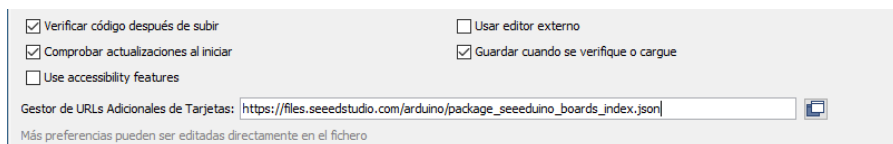
Paso 2. Abrir el archivo de ejemplo “Blink” yendo a *Archivo -> Ejemplos -> 01.Basics -> Blink*



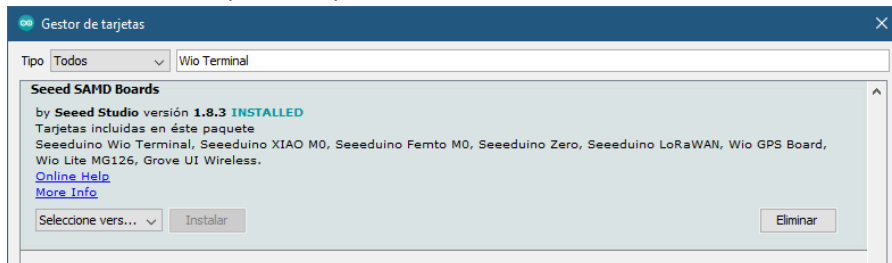
Paso 3. Añadir a el Wio Terminal a la biblioteca de tarjetas

1. Nos dirigimos a *Archivo > Preferencias* o usamos el comando *Ctrl + J* y copiamos el siguiente link en **Gestor de URLs Adicionales de Tarjetas** y aceptamos los cambios

[https://files.seeedstudio.com/arduino/package\\_seeeduino\\_boards\\_index.json](https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json)



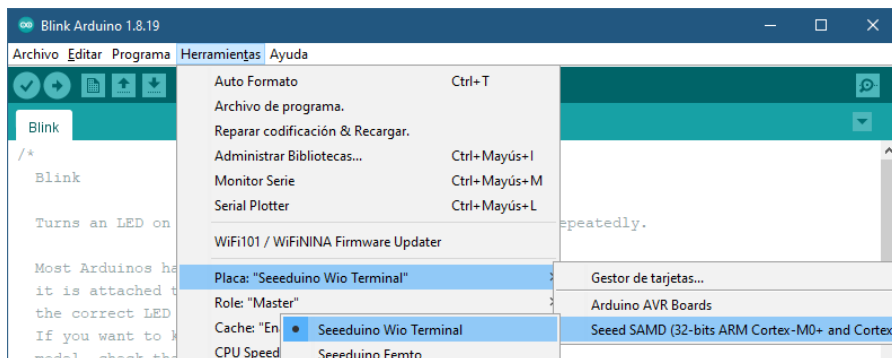
2. Después vamos a *Herramientas > Placa: “tarjeta seleccionada” > Gestor de tarjetas* y buscamos **Wio Terminal** en el gestor de tarjetas. Instalamos la que lleva por nombre **Seeed SAMD Boards**



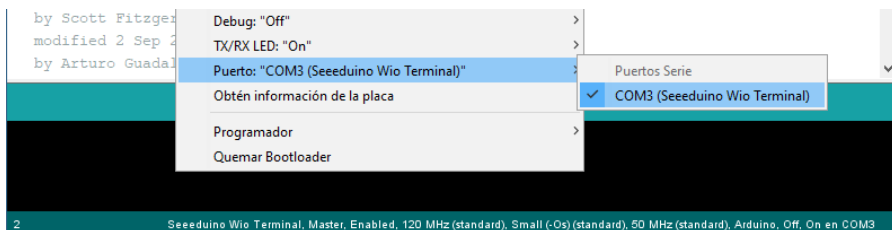
3. Reinicia el programa Arduino

Paso 4. Selecciona la tarjeta y el puerto

Primero nos dirigimos a *Placa: “tarjeta seleccionada” > Seeed SAMD [...]* y en el menú desplegado, seleccionamos **Seeeduino Wio Terminal**.



Después conectamos el Wio Terminal a nuestro equipo. Y en el apartado Herramientas, vamos a Puerto y seleccionamos el que diga **Seeeduino Wio Terminal**.



Al terminar guardamos el programa y veremos que el Wio Terminal es funcional cuando parpadea el LED en la parte inferior.

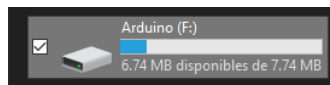




## Iniciando en ArduPy

Paso 1. Conecta el Wio Terminal a tu equipo de cómputo. Después entra al modo bootloader, que se puede ver en el apartado [Para acceder al Bootloader](#).

Una vez hecho eso, puedes comprobar que está listo cuando en el Explorador de archivos veas un dispositivo extraíble de nombre “Arduino”.



Paso 2. Descargue el firmware ArduPy en forma de archivos UF2. Y guarde donde este accesible.

Link de descarga: [https://files.seeedstudio.com/wiki/Wio-Terminal/res/ArduPy\\_wio\\_terminal\\_lastest.uf2](https://files.seeedstudio.com/wiki/Wio-Terminal/res/ArduPy_wio_terminal_lastest.uf2)

Paso 3. Instalacion del firmware ArduPy en el Wio Terminal. Para ello debe de arrastrar el archivo descargado del tipo UF2 hacia la unidad extraíble “Arduino”. Una vez hecho eso, la unidad desaparecerá del explorador. Debe reiniciar el Wio Terminal y el firmware de ArduPy ya estará listo para su funcionamiento.

Paso 4. Ahora, aparecerá una unidad USB llamada ARDUPY en su PC. Abra ARDUPY y verá un archivo python main.py. Abra main.py con su editor favorito, como [Microsoft Visual Studio Code](#), [Atom](#), [Sublime Text](#), etc. Copie el siguiente código y guarde main.py.

```
from machine import Pin, Map
import time
```

```
LED = Pin(Map.LED_BUILTIN, Pin.OUT)
```

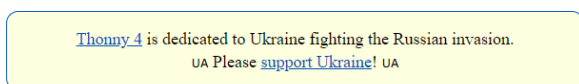
```
while True:
    LED.on()
    time.sleep(1)
    LED.off()
    time.sleep(1)
```

Al guardar el programa, veremos que el Wio Terminal parpadea el LED azul en la parte inferior.

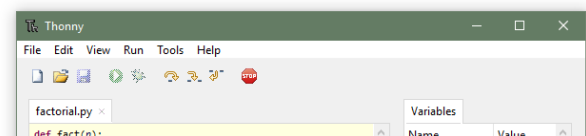
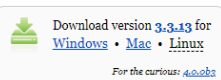


Para seguir trabajando en ArduPy, instalaremos una herramienta que nos será útil para poder hacer el código usando ArduPy (Python) como para poder recibir una respuesta al ‘Monitor Serial’ que envíe el Wio Terminal.

Es por lo que usando el buscador de nuestra preferencia y buscamos “Thonny”, que será el IDE en el que estaremos trabajando. Una vez encontrado y dentro de su página, damos clic al sistema operativo sobre el que se realizará la instalación (En este caso es Windows).

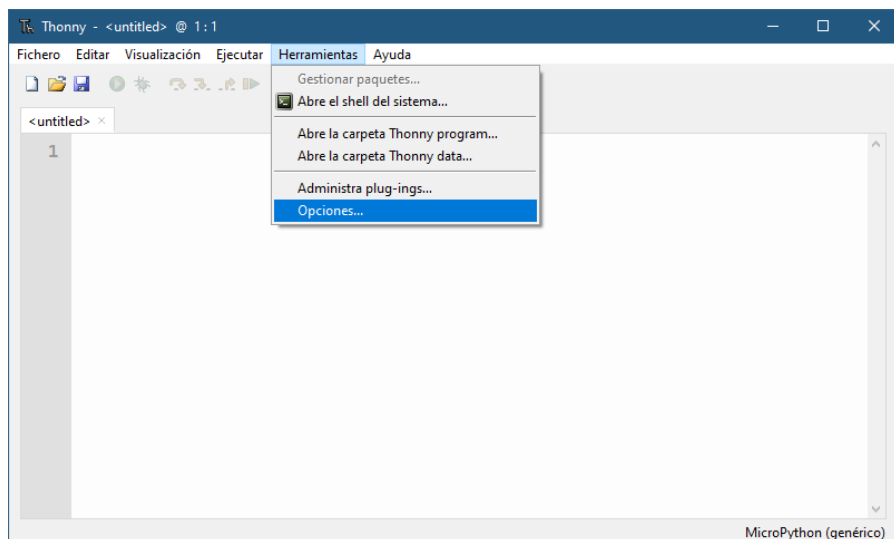


**Thonny**  
Python IDE for beginners

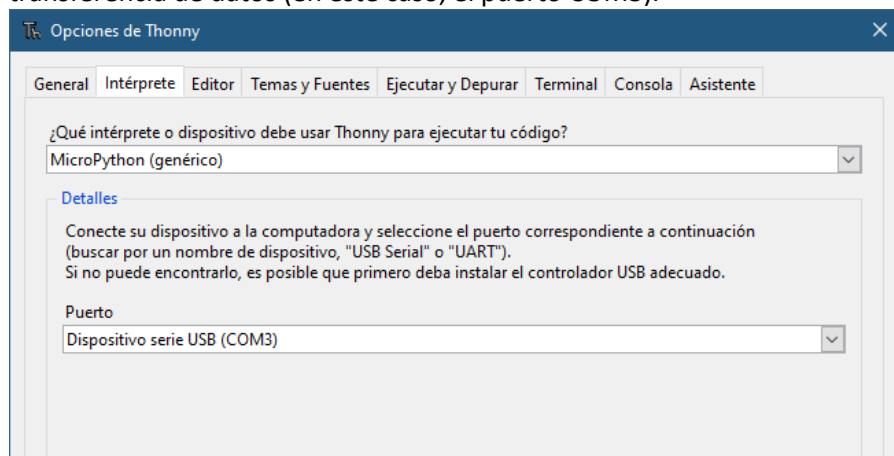


Ejecutamos el instalador en modo de administrador. Seleccionamos la instalación de preferencia entre todos los usuarios o solo para mí. La instalación es sencilla, solo se da clic a “Siguiente” las veces que sea necesario, aceptamos los términos y condiciones, verificamos la ubicación de instalación del archivo y esperamos a que finalice la instalación.

Una vez finalizada, abrimos Thonny y en la barra de herramientas damos clic a Herramientas > Opciones...



Dentro de las opciones de Thonny, seleccionamos la pestaña Interprete y en la opción del interprete que Thonny usará, seleccionamos la opción de MicroPython (genérico). Y en caso de tener conectado el Wio Terminal, podemos seleccionar el puerto serial por el que se realizará la transferencia de datos (en este caso, el puerto COM3).



Guardamos los cambios realizados, y estaremos listos para realizar proyectos desde Thonny. Para la prueba se puede usar el código anterior.

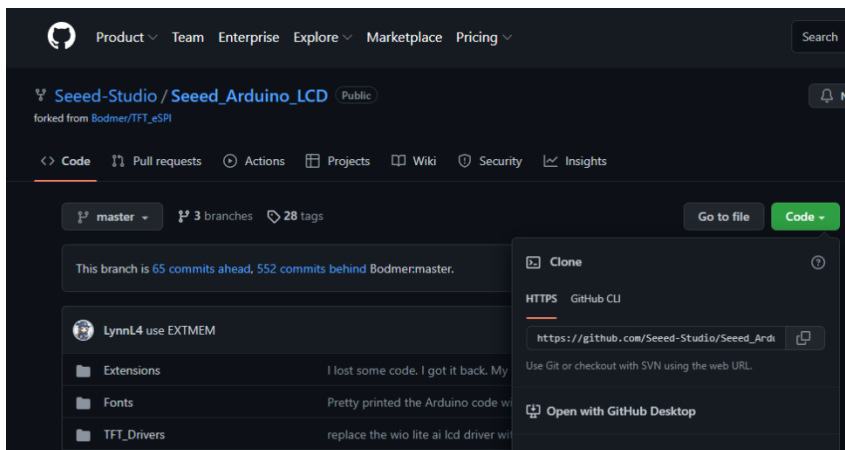
# Funcionamiento de los componentes

## LCD

### Antes de empezar

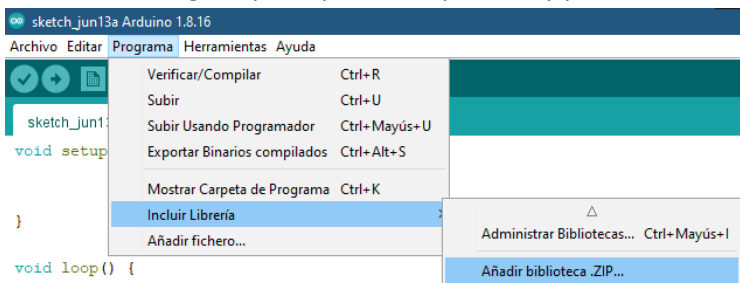
Antes de empezar con el uso de TFT LCD, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías conforme se vayan a ir usando.

Para instalar las librerías, nos dirigimos al siguiente repositorio de GitHub en el siguiente link [https://github.com/Seeed-Studio/Seeed\\_Arduino\\_LCD](https://github.com/Seeed-Studio/Seeed_Arduino_LCD) y descargamos todos los archivos como un ZIP.



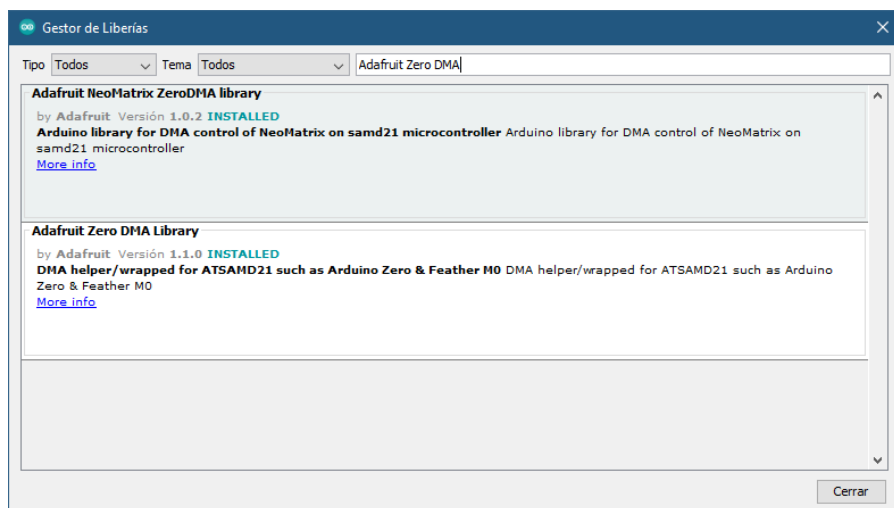
Captura de pantalla del sitio de GitHub para la descarga de la librería para la TFT LCD

Una vez descargada la librería en nuestro equipo, vamos a Arduino -> *Programa* -> *Incluir Librería* -> *Añadir biblioteca .ZIP...* y buscamos el archivo recién descargado para que sea importado y pueda ser usado.



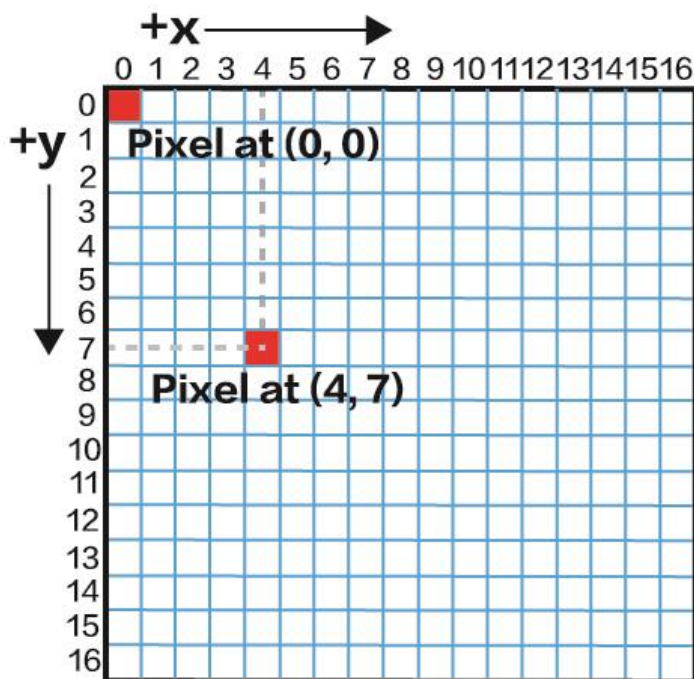
Algunas funciones pueden llegar a requerir una librería extra la cual se descargará desde *Administrar Bibliotecas...* A la cual accedemos desde

Programa -> Incluir Librería -> Administrar Bibliotecas o usando el comando *Ctrl + Mayus + i*. Y una vez en la ventana, buscamos **Adafruit Zero DMA** y la instalamos con los complementos que está pida.



## Primeros usos de la LCD

Lo primero que debemos de saber es que la LCD usa un Sistema de Coordenadas por Píxeles (Pixel Coordinates Systems), donde se puede indicar que pixel queremos que encienda o cualquier otra acción.



Donde dependiendo de la posición de X que se dirige de la parte superior izquierda hacia la derecha y de Y que se dirige de la parte superior hacia abajo, podemos hacer que un pixel se torne de algún color específico u otra cosa.

También debemos tener en cuenta que, al ser una LCD, puede representar diferentes colores, los cuales son los siguientes en base a ejemplos predefinidos:

## Ejemplo de colores de 16 bits

```
#define TFT_BLACK      0x0000    /*  0,   0,   0
*/
#define TFT_NAVY      0x000F    /*  0,   0, 128
*/
#define TFT_DARKGREEN  0x03E0    /*  0, 128,   0
*/
#define TFT_DARKCYAN  0x03EF    /*  0, 128, 128
*/
#define TFT_MAROON     0x7800    /* 128,   0,   0
*/
#define TFT_PURPLE     0x780F    /* 128,   0, 128
*/
#define TFT_OLIVE      0x7BE0    /* 128, 128,   0
*/
#define TFT_LIGHTGREY  0xC618    /* 192, 192, 192
*/
#define TFT_DARKGREY   0x7BEF    /* 128, 128, 128
*/
#define TFT_BLUE       0x001F    /*   0,   0, 255
*/
#define TFT_GREEN      0x07E0    /*   0, 255,   0
*/
#define TFT_CYAN       0x07FF    /*   0, 255, 255
*/
#define TFT_RED        0xF800    /* 255,   0,   0
*/
#define TFT_MAGENTA    0xF81F    /* 255,   0, 255
*/
#define TFT_YELLOW     0xFFE0    /* 255, 255,   0
*/
#define TFT_WHITE      0xFFFF    /* 255, 255, 255
*/
#define TFT_ORANGE     0xFDA0    /* 255, 180,   0
*/
#define TFT_GREENYELLOW 0xB7E0    /* 180, 255,   0
*/
```





Para inicializar la pantalla en Arduino, debemos de insertar el siguiente código.

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    ...
    tft.begin();
    tft.setRotation(r);
    digitalWrite(LCD_BACKLIGHT, HIGH); // Enciende la
    luz de fondo
    ...
}
```

Debemos de considerar lo siguiente:

**r**: Es equivalente a la rotación de la pantalla TFT que tiene valores que van del 0 - 3, dando referencia de en qué esquina comenzará. El valor recomendado es 3.

Encender el fondo de la pantalla en algún color

Debemos insertar el siguiente código

En Arduino:

```
#include"TFT_eSPI.h" // Importamos la librería
TFT_eSPI tft; // Iniciamos la librería como un objeto

void setup() {
    tft.begin(); // Iniciamos el objeto
    tft.setRotation(3); // Establecemos el sentido de
    la rotación

    tft.fillScreen(TFT_RED); // Enciende toda la
    pantalla de color rojo
}

void loop() {

}
```

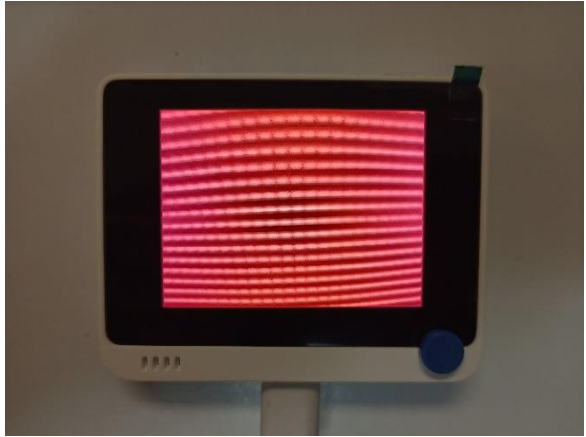
En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
```

Debemos de considerar lo siguiente:

Se puede encender la LCD de algún otro color que seleccione el usuario, solo sustituyendo a `TFT_RED` por el color que se prefiera.



*Ilustración 1: Pantalla encendida en rojo*

Apagar el fondo de la pantalla de la LCD

Usando el primer código donde se inicializaba la pantalla, hemos de modificar la parte donde se enciende la iluminación de fondo que estaba en estado **HIGH** a estado **LOW**

En Arduino:

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    ...
    tft.begin();
    tft.setRotation(r);
    digitalWrite(LCD_BACKLIGHT, LOW); // Apaga la luz
    de fondo
    ...
}
```

## Funciones graficas básicas

### *Dibujar pixeles*

Para dibujar un píxel en la pantalla LCD debemos usar el siguiente comando

```
drawPixel(int32_t x, int32_t y, uint32_t color);
```

Debemos de considerar que 'x' y 'y' son valores enteros, que marcan la posición en coordenadas en la LCD. Y en color debemos de introducir de qué **color** queremos el píxel.

Código de ejemplo:

Para Arduino:

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);

    tft.fillScreen(TFT_WHITE); //usamos un fondo blanco
    tft.drawPixel(4,7,TFT_BLACK); //dibujamos un pixel
    negro en (4,7)
}

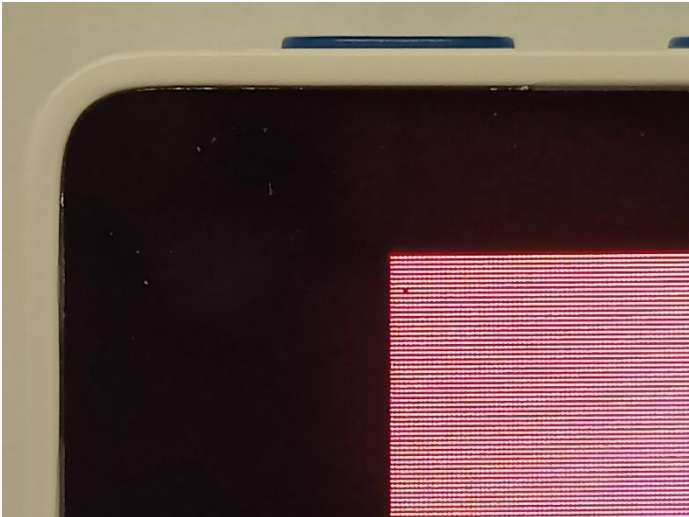
void loop() {}
```

Para ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
lcd.drawPixel(4,7,lcd.color.BLACK) # Dibujamos un pixel
negro en (4,7)
```

Resultado:



*Ilustración 2: Pixel apagado en pantalla roja*

### *Dibujar una línea libre*

Para dibujar una línea en la pantalla LCD debemos usar el siguiente comando

```
drawLine(int32_t x0, int32_t y0, int32_t x1, int32_t y1, uint32_t color);
```

Debemos de considerar que 'x0' y 'y0' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que inicie la línea; que 'x1' y 'y1' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que finalice la línea. Y en color debemos de introducir de qué **color** queremos la línea.

Código de ejemplo:

En Arduino:

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);

    tft.fillScreen(TFT_RED); //Fondo de color rojo
    tft.drawLine(0,0,160,120,TFT_BLACK); //Línea negra
    de (0,0) a (160,120)
}

void loop() {}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
lcd.drawLine(0,0,160,120,lcd.color.BLACK) # Línea negra
de (0,0) a (160,120)
```

Resultado:



*Ilustración 3: Línea recta dibujada*



### *Dibujar líneas horizontales y verticales*

Para dibujar líneas horizontales

```
drawFastHLine(int32_t x, int32_t y, int32_t w, uint32_t  
color); //Horizontal line
```

Para dibujar líneas verticales

```
drawFastVLine(int32_t x, int32_t y, int32_t h, uint32_t  
color); //Vertical line
```

Para ambos casos debemos de tener en cuenta

Debemos de considerar que para ambos casos 'x' y 'y' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que inicie la línea; que 'w' indica lo ancho que queremos que sea la línea Horizontal y que 'h' indica lo alto que queremos que sea la línea Vertical. Y en color debemos de introducir de qué **color** queremos la línea.

Código de ejemplo:

En Arduino:

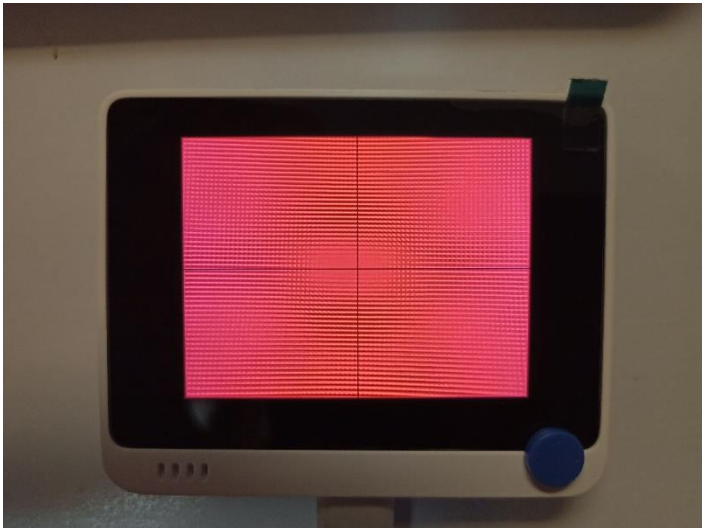
```
#include "TFT_eSPI.h"  
TFT_eSPI tft;  
  
void setup() {  
    tft.begin();  
    tft.setRotation(3);  
  
    tft.fillScreen(TFT_RED); //Fondo rojo  
    tft.drawFastHLine(0,120,320,TFT_BLACK); //Una línea  
horizontal negra que inicia en (0, 120)  
    tft.drawFastVLine(160,0,240,TFT_BLACK); // Una  
línea vertical blanca que inicia en (160, 0)  
}  
  
void loop() {}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD
```

```
lcd = LCD() # Iniciamos la LCD y encendemos el fondo  
lcd.fillScreen(lcd.color.RED) # Enciende toda la  
pantalla de color rojo  
lcd.drawFastHLine(0,120,320,lcd.color.BLACK) # Una  
línea horizontal negra que inicia en (0, 120)  
lcd.drawFastVLine(160,0,240,lcd.color.BLACK) # Una  
línea vertical blanca que inicia en (160, 0)
```

Resultado:



*Ilustración 4: Líneas cruzadas en la pantalla con fondo rojo*

### *Dibujar rectángulos*

Para dibujar un rectángulo, pero solo los contornos debemos de usar el siguiente comando

```
drawRect(int32_t x, int32_t y, int32_t w, int32_t h,  
uint32_t color);
```

Para dibujar un rectángulo, pero solo el relleno debemos usar el siguiente comando

```
fillRect(int32_t x, int32_t y, int32_t w, int32_t h,  
uint32_t color);
```

Debemos de considerar que para ambos casos 'x' y 'y' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que inicie el dibujo o el relleno; que 'w' indica lo ancho que queremos que sea la línea Horizontal y que 'h' indica lo alto que queremos que sea la línea Vertical. Y en color debemos de introducir de qué **color** queremos tanto en el contorno como el relleno.

En Arduino:

```
#include "TFT_eSPI.h"  
TFT_eSPI tft;  
  
void setup() {  
    tft.begin();  
    tft.setRotation(3);  
  
    tft.fillScreen(TFT_DARKGREEN); //Fondo verde oscuro  
    tft.drawRect(110,70,100,100,TFT_CYAN); //El  
    contorno de un rectángulo cyan de 100x100 iniciando de  
    (110, 70)  
    tft.fillRect(111,71,98,98,TFT_MAGENTA); //El  
    relleno del rectángulo magenta, si está en las mismas  
    coordenadas y con el mismo ancho y alto, tapa el  
    contorno  
}  
void loop() {}
```

En ArduPy:

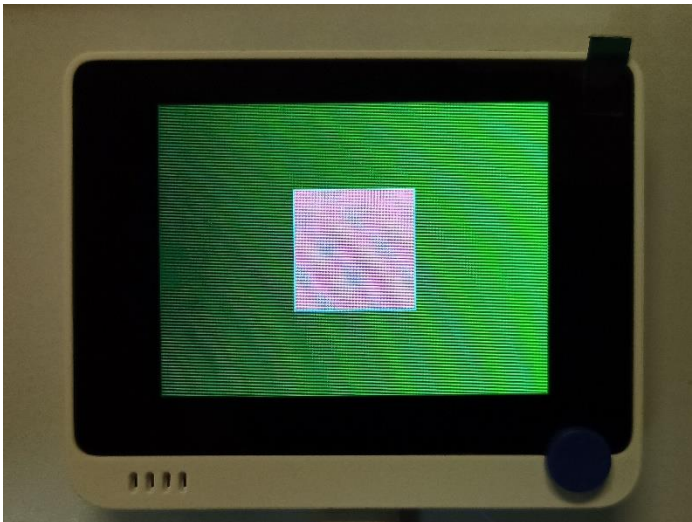
```
from machine import LCD # Importamos la libreria de LCD
```

```
lcd = LCD() # Iniciamos la LCD y encendemos el fondo  
lcd.fillScreen(lcd.color.DARKGREEN) # Enciende toda la  
pantalla de color verde oscuro
```

```
lcd.drawRect(110,70,100,100,lcd.color.CYAN) # El  
contorno de un rectángulo cyan de 100x100 iniciando de  
(110, 70)
```

```
lcd.fillRect(111,71,98,98,lcd.color.MAGENTA) # El  
relleno del rectángulo magenta, si está en las mismas  
coordenadas y con el mismo ancho y alto, tapa el  
contorno
```

Resultado:



*Ilustración 5: Rectángulo de lados iguales con contorno Cian y fondo verde*

## Dibujar círculos

Para dibujar un círculo, pero solo los contornos debemos usar el siguiente comando:

```
drawCircle(int32_t x0, int32_t y0, int32_t r, uint32_t color);
```

Para dibujar un círculo, pero solo el relleno debemos usar el siguiente comando:

```
fillCircle(int32_t x0, int32_t y0, int32_t r, uint32_t color);
```

Debemos de considerar que para ambos casos 'x0' y 'y0' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que inicie el dibujo o el relleno del círculo y será el centro de nuestro círculo; que 'r' indica el radio que queremos. Y en color debemos de introducir de qué color queremos tanto en el contorno como el relleno.

Ejemplo de dibujo de un círculo:

En Arduino:

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);

    tft.fillScreen(TFT_RED); //Fondo rojo
    tft.drawCircle(160,120,50,TFT_BLACK); //Un círculo negro con origen en (160, 120)
}

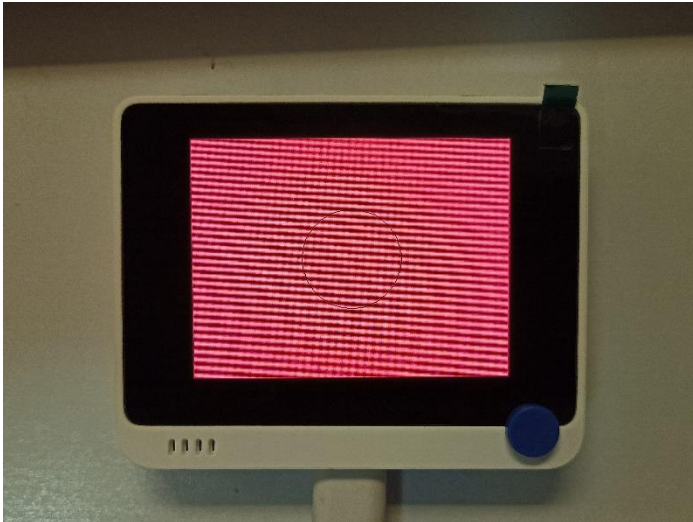
void loop() {}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
lcd.drawCircle(160,120,50,lcd.color.BLACK) # Un círculo
negro con origen en (160, 120)
```

Resultado:



*Ilustración 6: Círculo dibujado en pantalla con fondo rojo*

### *Dibujar elipses*

Para dibujar una elipse, pero solo los contornos debemos de usar el siguiente comando

```
drawEllipse(int16_t x0, int16_t y0, int32_t rx, int32_t ry,
uint16_t color);
```

Para dibujar una elipse, pero solo el relleno debemos de usar el siguiente comando

```
fillEllipse(int16_t x0, int16_t y0, int32_t rx, int32_t ry,
uint16_t color);
```

Debemos de considerar que para ambos casos 'x0' y 'y0' son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que inicie el dibujo o el relleno de la elipse y será el centro de nuestro circulo; que 'rx' indica el radio que queremos sobre el eje X desde el punto de origen; que 'ry' indica el radio que queremos sobre el eje Y desde el punto de origen. Y en color debemos de introducir de qué **color** queremos tanto en el contorno como el relleno.

Ejemplo de dibujo de una elipse:

En Arduino:

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);

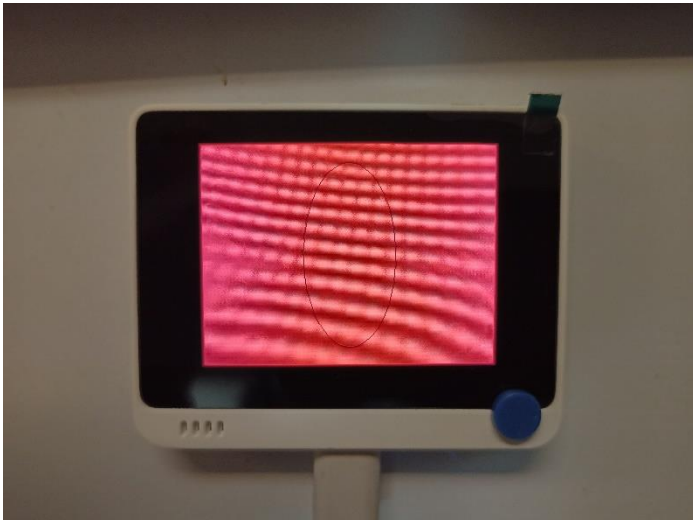
    tft.fillScreen(TFT_RED); //Fondo rojo
    tft.drawEllipse(160,120,50,100,TFT_BLACK);
    // Una elipse negra con origen en (160, 120) con un
    radio horizontal de 50, y un radio vertical de 100
}
void loop() {}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
lcd.drawEllipse(160,120,50,100,lcd.color.BLACK) # Una
elipse negra con origen en (160, 120) con un radio
horizontal de 50, y un radio vertical de 100
```

Resultado:



*Ilustración 7: Elipse dibujada en pantalla con fondo rojo*



### *Dibujar triángulos*

Para dibujar un triángulo, pero solo los contornos debemos de usar el siguiente comando

```
drawTriangle(int32_t x0, int32_t y0, int32_t x1, int32_t y1,  
int32_t x2, int32_t y2, uint32_t color);
```

Para dibujar un triángulo, pero solo el relleno debemos de usar el siguiente comando

```
fillTriangle(int32_t x0, int32_t y0, int32_t x1, int32_t y1,  
int32_t x2, int32_t y2, uint32_t color);
```

Debemos de considerar que para ambos casos las 'x's y las 'y's, independientemente del número del que vengan acompañados son valores enteros, que marcan la posición en coordenadas en la LCD donde queremos que se ubiquen los puntos del triángulo con 'x0' y 'y0' como punto uno, 'x1' con 'y1' como punto dos y 'x2' con 'y2' como punto tres. Y en color debemos de introducir de qué **color** queremos tanto en el contorno como el relleno.

Ejemplo de un triángulo:

En Arduino:

```
#include "TFT_eSPI.h"  
TFT_eSPI tft;  
  
void setup() {  
    tft.begin();  
    tft.setRotation(3);  
  
    tft.fillScreen(TFT_RED); //Fondo rojo  
    tft.drawTriangle(160,70,60,170,260,170,TFT_BLACK);  
    // Un triángulo con puntos en (160, 70), (60, 170)  
    y (260, 170)  
}  
  
void loop() {}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD

lcd = LCD() # Iniciamos la LCD y encendemos el fondo
lcd.fillScreen(lcd.color.RED) # Enciende toda la
pantalla de color rojo
lcd.drawTriangle(160,70,60,170,260,170,lcd.color.BLACK)
# Un triángulo con puntos en (160, 70), (60, 170) y
(260, 170)
```

Resultado:



*Ilustración 8: Triangulo dibujado en pantalla con fondo rojo*

### *Para llenar el fondo*

Para llenar el fondo debemos usar el siguiente comando:

```
fillScreen(uint32_t color);
```

Donde la única consideración que debemos de tener es el color que vamos a usar.

Un ejemplo del uso de este comando es hacer que los colores de la pantalla cambien entre tres colores cada segundo, los cuales son rojo, verde y azul (Colores que componen el famoso RGB).

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);
}

void loop() {
    //En ciclo cambia entre los colores R-G-B
    tft.fillScreen(TFT_RED);
    delay(1000);
    tft.fillScreen(TFT_GREEN);
    delay(1000);
    tft.fillScreen(TFT_BLUE);
    delay(1000);
}
```

En ArduPy:

```
from machine import LCD # Importamos la libreria de LCD
import time

lcd = LCD() # Iniciamos la LCD y encendemos el fondo

while True: # En ciclo cambia entre los colores R-G-B
    lcd.fillScreen(lcd.color.RED) # Enciende toda la
    pantalla de color rojo
    time.sleep(1)
    lcd.fillScreen(lcd.color.GREEN) # Enciende toda la
    pantalla de color verde
    time.sleep(1)
    lcd.fillScreen(lcd.color.BLUE) # Enciende toda la
    pantalla de color azul
    time.sleep(1)
```

Resultado:



*Ilustración 9: Pantalla con fondo rojo*

### *Para escribir en la pantalla*

Para escribir en la pantalla debemos usar el siguiente comando:

```
drawString(const String& string, int32_t poX, int32_t poY);
```

Donde debemos de tener en consideración que en `const String& string` va el mensaje que deseamos escribir entre comillas, que `poX` y `poY` marcan de donde queremos que parta el mensaje

Otro de los comandos útiles al momento de escribir en la pantalla es definir el tamaño del texto en pantalla; el comando usado para el tamaño de la pantalla es:

```
setTextSize(n)
```

Un ejemplo es escribir “Hola Mundo” con fondo rojo en dos tamaños.

```
#include "TFT_eSPI.h"
TFT_eSPI tft;

void setup() {
    tft.begin();
    tft.setRotation(3);

    tft.fillScreen(TFT_RED); //Fondo rojo

    tft.setTextColor(TFT_BLACK); //Define el color del
    texto
    tft.setTextSize(1); // Define tamaño del texto a 1
    tft.drawString("Hola Mundo", 0, 0); // Imprime texto
    desde (0, 0)
    tft.setTextSize(2); // Define tamaño del texto a 2
    tft.drawString("Hola Mundo", 0, 10); // Imprime texto
    desde (0,10)
}

void loop() {}
```

En ArduPy:

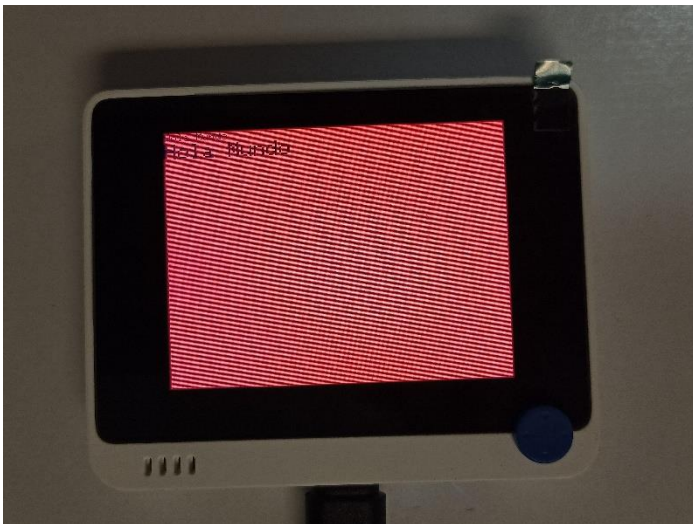
```
from machine import LCD # Importamos la libreria de LCD
import time

lcd = LCD() # Iniciamos la LCD y encendemos el fondo

lcd.fillScreen(lcd.color.RED) # Llenamos el fondo de la
LCD en rojo

lcd.setTextColor(lcd.color.BLACK) # Define el color del
texto
lcd.setTextSize(1) # Define tamaño del texto a 1
lcd.drawString("Hola Mundo",0,0) # Imprime texto desde
(0, 0)
lcd.setTextSize(2) # Define tamaño del texto a 2
lcd.drawString("Hola Mundo",0,10) # Imprime texto desde
(0,10)
```

Resultado:

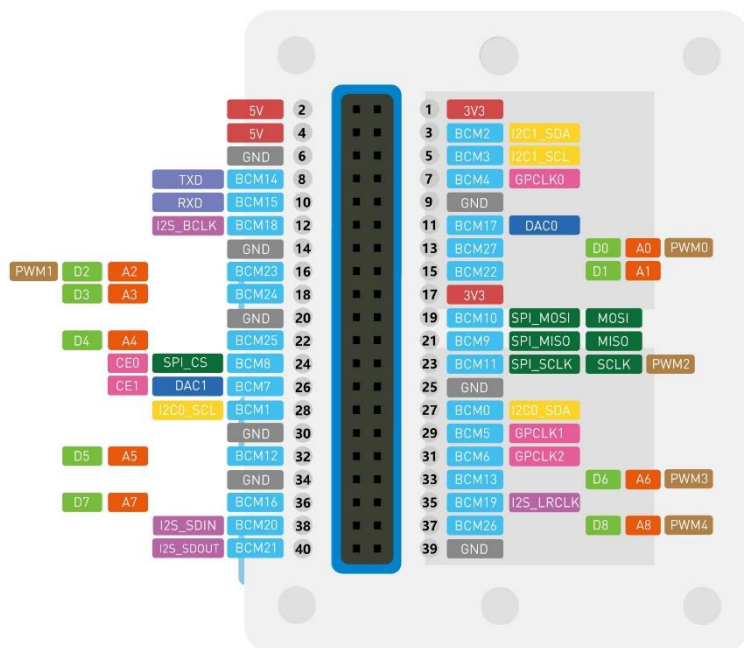


## IO

### Describiendo los Pines

El IO (Input / Output) son parte fundamental del Wio Terminal, debido a que a través de ellos podremos conectar distintos elementos, tanto sensores como actuadores que nos serán fundamentales para la lectura de datos en casos específicos.

Para que sean utilizables, podemos utilizar el nombre de los pines como se indica a continuación, y al ser algunos multifuncionales, podrán ser referenciados de manera diferente.



**4 pines de alimentación:** 2 pines de 3.3V y 2 pines de 5V

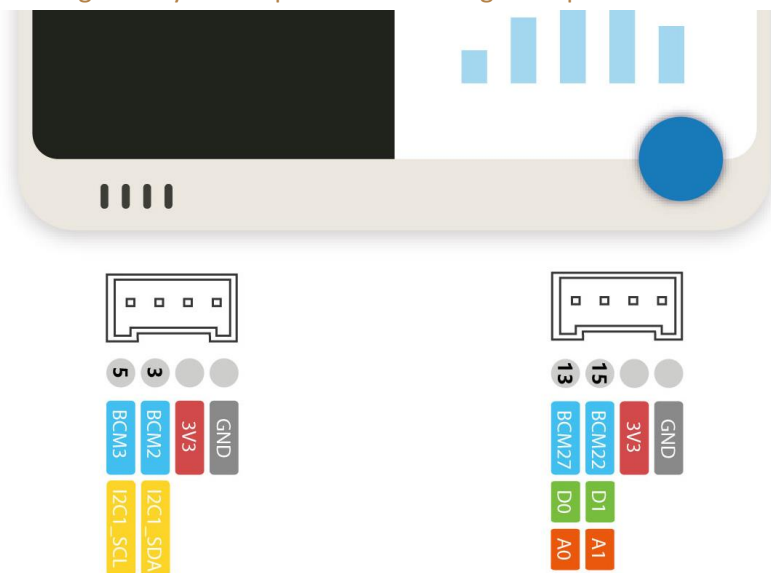
**8 pines de tierra**

**28 pines BCM#(Broadcom - Semejante al GPIO):** # indica el número de pin el cual puede ser utilizado en el software.

- **I2C#\_\*\*\*:** I2C es un protocolo donde los datos se transfieren poco a poco a lo largo de un solo cable. # puede ser 0 o 1.
- 2 pines **I2C#\_SDA:** El SDA es la línea para que el maestro y el esclavo envíen y reciban datos.
- 2 pines **I2C#\_SCL:** El SCL es la línea que lleva la señal del reloj.
- **I2S\_\*\*\*\*:** I2S es un estándar que permite usar pines para conectar dispositivos de audio digitales.
- Pines **I2S\_SDIN** o **I2S\_SDOUT:** Línea de datos, impulsada por uno de los esclavos o maestros dependiendo de la dirección de datos. Puede haber varias líneas de datos en diferentes direcciones.
- Pin **I2S\_BCLK:** Reloj de bits. Esta es una división fija del MCLK y es impulsada por el maestro.
- Pin **I2S\_LRCLK:** Reloj de palabras (o selección de palabras). Esto es impulsado por el maestro.
- **SPI\_\*\*\*:** La interfaz periférica en serie (SPI) es una de las interfaces más utilizadas entre el microcontrolador y los circuitos integrados periféricos, como sensores, ADC, DAC, registros de desplazamiento, SRAM y otros.
- Pin **SPI\_SCLK:** Reloj.
- Pin **SPI\_CS:** Chip seleccionado.
- Pin **SPI\_MOSI:** Salida principal, entrada de subnodo.
- Pin **SPI\_MISO:** Entrada principal, salida de subnodo.
- 3 Pines **GPCLK#:** Reloj de propósito general. Los pines de reloj de uso general se pueden configurar para generar una frecuencia fija sin ningún control de software continuo. # puede ser del 0 al 2.
- 2 Pines **DAC#:** Convertidor de digital a analógico. # puede ser 0 o 1.
- 2 Pines **CE#:** Permite la comunicación con un dispositivo SPI, donde se requiere encender el pin de Chip Select correspondiente al chip con estos pines. # puede ser 0 o 1.
- 9 pines digitales **D#:** Pines digitales de entrada y salida. # puede ser del 0 al 8.
- 9 pines analógicos **A#:** Pines digitales de entrada. # puede ser 0 al 8.



- 5 pines **PMW#**: PWM es una técnica que se usa para transmitir señales analógicas cuya señal portadora será digital. # puede ser del 0 al 4.



Puertos compatibles con accesorios Grove de Seedstudio

**2 pines de alimentación de 3.3V.**

**2 pines de tierra.**

**4 pines BCM#(Broadcom - Semejante al GPIO):** # indica el número de pin el cual puede ser utilizado en el software.

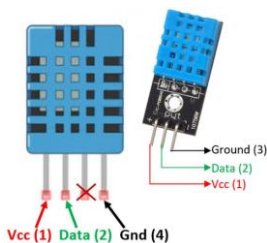
- **I2C1\_\*\*\***: I2C es un protocolo donde los datos se transfieren poco a poco a lo largo de un solo cable. # puede ser 0 o 1.
- 1 pin **I2C1\_SDA**: El SDA es la línea para que el maestro y el esclavo envíen y reciban datos.
- 1 pin **I2C1\_SCL**: El SDA es la línea que lleva la señal del reloj.
- 2 pines digitales **D#**: Pines digitales de entrada y salida. # puede ser del 0 o 1.
- 2 pines analógicos **A#**: Pines digitales de entrada. # puede ser del 0 o 1.

## Usando Pines Digitales

Para hacer el uso de pines digitales del Wio Terminal, debemos de tener en cuenta primero que sensor que usaremos. En el caso de este ejemplo usaremos el sensor de temperatura y humedad:

### Paso 1. Identificar los pines que usaremos

Debe de haber en consideración los pines que utiliza el sensor DHT11, los cuales son uno de alimentación (Uso preferente de 3.3V), uno de Tierra y uno de transmisión de datos, el cual será el que conectará con el pin digital.



### Paso 2. Realizar la conexión

De forma adecuada deberemos de tener lista la conexión para su uso, por lo que de acuerdo con la imagen () usaremos los siguientes pines:

- Alimentación de 3.3V (PIN 1)
- Tierra (PIN 9)
- Transmisión de datos (PIN 13)

Debemos de evitar hacer uso de cables que estén en mal estado.

### Paso 3. Realizar el código

Para Arduino:

Debes de copiar el siguiente código:

```
#include "TFT_eSPI.h"
#include "DHT.h" //Llamamos a la libreria
TFT_eSPI tft;
```

```

DHT dht(D0,DHT11); //Definimos el PIN a usar y el
sensor

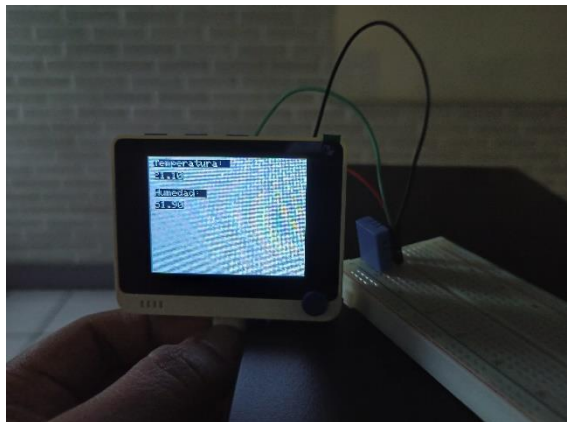
void setup() {
  tft.begin();
  dht.begin(); //Inicializamos el sensor
  tft.setRotation(3);

  tft.fillScreen(TFT_WHITE); //Fondo blanco
}

void loop() {
  float temp = dht.readTemperature(); //Leemos la
temperatura y guardamos el dato en una variable float
  float hume = dht.readHumidity(); //Leemos la humedad
y guardamos el dato en una variable float
  tft.setTextSize(2);
  tft.drawString("Temperatura: ",10,10);
  tft.drawString(String(temp),10,30); //Imprimimos en
el Serial el valor de la temperatura
  tft.drawString("Humedad: ",10,50);
  tft.drawString(String(hume),10,70); //Imprimimos en
el Serial el valor de la humedad
  delay(5000); //Realizamos una pausa de 5000
milisegundos (5 segundos)
}

```

Resultado:



## Ejemplo 2: Encender un LED

Lo primero que debemos de hacer es definir el PIN el cual ocuparemos para que se conecte el LED, en este caso será D0.

Código para Arduino:

```
#define LED1 D0

void setup() {
  pinMode(LED1,OUTPUT);
}

void loop() {
  digitalWrite(LED1,HIGH);
  delay(1000);
  digitalWrite(LED1,LOW);
  delay(1000);
}
```

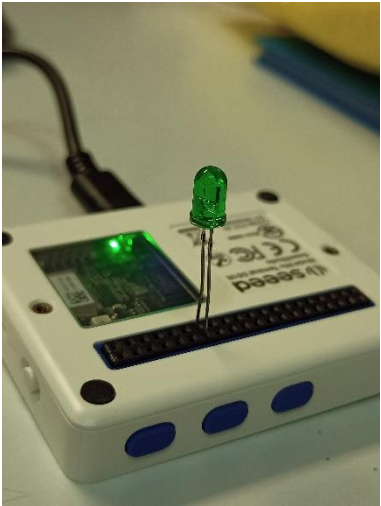
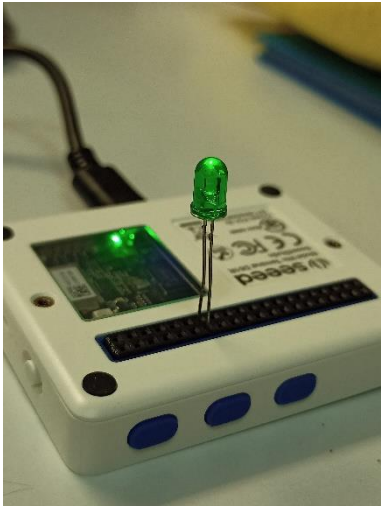
Código para ArduPy:

```
from machine import Pin, Pin # Importamos la libreria
de Pin, Map
import time

LED = Pin(Pin(13), Pin.OUT)

while True:
  LED.on()
  time.sleep(1)
  LED.off()
  time.sleep(1)
```

Resultado:



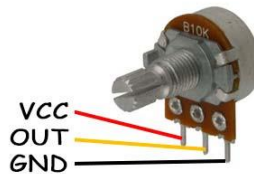
## Usando Pines Analógicos

Para hacer el uso de pines analógicos del Wio Terminal, debemos de tener en cuenta cual será el sensor que usaremos. En este caso de ejemplo usaremos un potenciómetro:

Al no tener una librería que nos permita usarlo, deberemos de solo leer el estado del sensor usando la función.

Paso 1. Identificar los pines que vamos a utilizar

Debe haber en consideración los pines que utiliza el potenciómetro de 10k, los cuales son uno de alimentación(Usa preferente 3.3V), uno de



Tierra y uno de transmisión de datos, el cual será el que se conectará con el pin analógico.

De forma adecuada deberemos de tener lista la conexión para su uso, por lo que de acuerdo con la imagen () usaremos los siguientes pines:

- VCC (PIN 1)
- GND (PIN 9)
- OUT (PIN 13)

Debemos de evitar hacer uso de cables que estén en mal estado.

Paso 2. Realizar el código

Para Arduino:

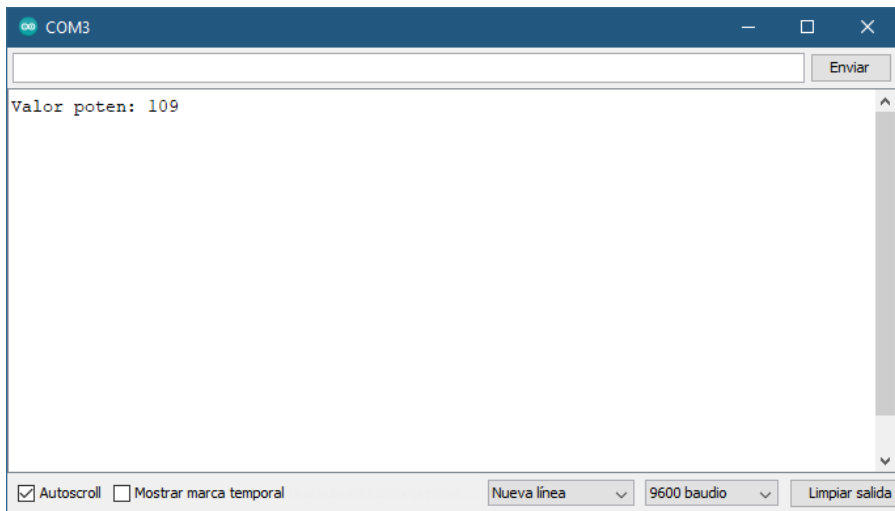
```
void setup() {  
  Serial.begin(9600); // Definimos la velocidad de  
  transmisión de datos  
}
```

```

void loop() {
    int poten = analogRead(A0); // Leemos el valor del
    potenciómetro
    Serial.println("Valor poten: " + String(poten)); //
    Imprimimos el valor del potenciómetro
    delay(5000); // Pausamos por 5000 milisegundos (5
    segundos)
}

```

Resultado:



Para ArduPy:


```

from machine import ADC, Pin # Importamos las librerías
ADC y PIN
import time
adc = ADC(Pin(13)) # Creamos el convertidos ADC en Pin
13
poten = adc.read() # Leemos los valores de ADC, 0 ~
1023
while True:
    print("Valor poten: " + str(poten)) # Imprime el
    valor del potenciómetro
    time.sleep(5) # Pausamos por 5 segundos

```

```
poten = adc.read() # Leemos los valores de ADC, 0 ~ 1023
```

Resultado:



The screenshot shows the Thonny IDE interface. The top menu bar includes 'Fichero', 'Editar', 'Visualización', 'Ejecutar', 'Herramientas', and 'Ayuda'. Below the menu is a toolbar with icons for file operations and execution. The main editor window displays a Python script named 'main.py' with the following code:

```
1 from machine import ADC, Pin # Importamos las librerías ADC y PIN
2 import time
3 adc = ADC(Pin(13)) # Creamos el convertidos ADC en Pin 13
4 poten = adc.read() # Leemos los valores de ADC, 0 ~ 1023
5 while True:
6     print("Valor poten: " + str(poten)) # Imprime el valor del potenciómetro
7     time.sleep(5) # Pausamos por 5 segundos
8     poten = adc.read() # Leemos los valores de ADC, 0 ~ 1023
```

Below the editor is a console window titled 'Consola'. It shows the output of the script:

```
main.py output:
Valor poten: 109
Valor poten: 115
```

The status bar at the bottom right indicates 'CircuitPython (genérico)'.



## Wifi (Compatible solo con Arduino)

Uso por primera vez

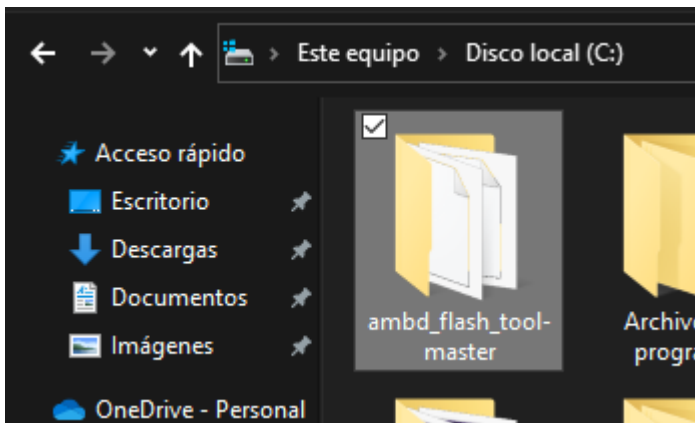
Hay una herramienta llamada *ambd\_flash\_tool*, y cuando ejecuta esta herramienta, primero habilita una conexión en serie de SAMD51(Microprocesador) a RTL8720(Adaptador Wifi y Bluetooth) para que el firmware se instale en el microprocesador.

Esto se hace porque el adaptador Wifi-Bluetooth no puede comunicarse directamente con microprocesador.

Nota: Solo necesita borrar el firmware de fábrica por primera vez. Luego, puede actualizar el nuevo firmware para sobrescribir el firmware existente.

**Paso 1.** Debemos de descargar el archivo para realizar el flasheo del Wio Terminal, eso en el siguiente link: [https://github.com/Seeed-Studio/ambd\\_flash\\_tool](https://github.com/Seeed-Studio/ambd_flash_tool)

**Paso 2.** Una vez descargada la herramienta, la extraemos del ZIP en el que está contenido en el lugar donde deseemos.



**Paso 3.** Una vez extraído el contenido, ejecutamos el procesador de comandos, en este caso, en la barra de dirección escribimos CMD y automáticamente se abrirá en el directorio deseado.

**Paso 4.** Conectamos el Wio Terminal y luego lo encendemos.

**Paso 5.** Ejecutamos el siguiente comando **ambd\_flash\_tool.exe erase** , dicho servirá para que se borre el firmware inicial. **OJO: No cerrar la terminal una vez iniciado el proceso, puede dañar el Wio Terminal de forma grave.**

```
C:\Windows\System32\cmd.exe - ambd_flash_tool.exe erase
Microsoft Windows [Versión 10.0.19044.1865]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\ambd_flash_tool-master>ambd_flash_tool.exe erase
MODE COM3:BAUD=%d PARITY=N DATA=8

Estado para dispositivo COM3:
-----
Baudios:          1200
Paridad:          None
Bits de datos:    8
Bits de paro:     1
```

Nota: Al mismo tiempo, el LCD del Wio Terminal encenderá y mostrará el mensaje “Burn RTL8720 fw”

```
read(addr=0x13400,size=0x200)
[=====] 99% (123/124 pages)checksumBuffer(start_addr=0x13600, size=0x80) = a95f
read(addr=0x13600,size=0x200)
[=====] 100% (124/124 pages)
Verify successful
Done in 3.657 seconds
writeWord(addr=0xe00ed0c,value=0x5fa0004)
wait...
wait...
wait...
Flashing...
All images are sent successfully!
Image tool closed!

Success!

C:\ambd_flash_tool-master>
```

**Paso 6.** Una vez terminado el proceso de eliminar el firmware inicial, ejecutaremos el siguiente comando: **ambd\_flash\_tool.exe flash** dicho servirá para que el módulo Wifi-Bluetooth sea usable.

```
C:\ambd_flash_tool-master>ambd_flash_tool.exe flash
COM3
MODE COM3:BAUD=%d PARITY=N DATA=8

Estado para dispositivo COM3:
-----
Baudios:          1200
Paridad:          None
Bits de datos:    8
Bits de paro:     1
Tiempo de espera: ON
XON / XOFF:       OFF
Protocolo CTS:    OFF
Protocolo DSR:    OFF
Sensibilidad de DSR: OFF
Círculo DTR:      OFF
Círculo RTS:      OFF

[=====] 100% (124/124 pages)
Verify successful
Done in 3.332 seconds
writeWord(addr=0xe00ed0c,value=0x5fa0004)
wait...
wait...
wait...
copy img to workspace...
Flashing...
All images are sent successfully!
Image tool closed!

Success!

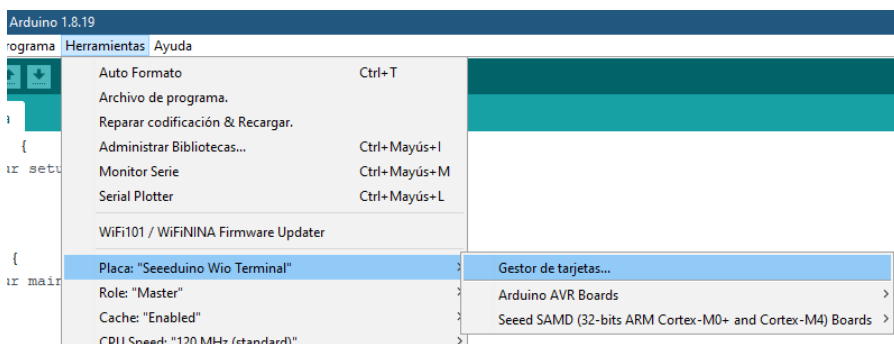
C:\ambd_flash_tool-master>
```

Una vez terminado dicho proceso, habremos terminado el flasheo del Wio Terminal.

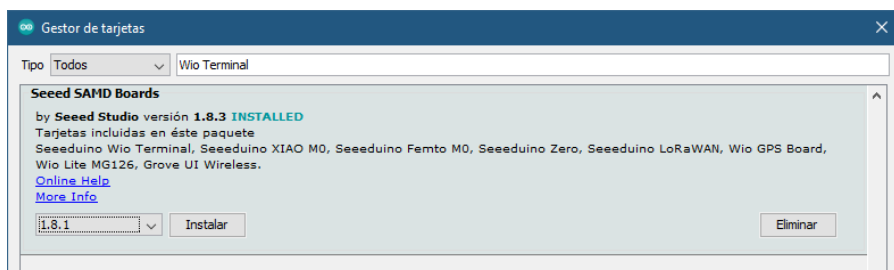
### Antes de empezar

Una vez realizado el flasheo, debemos de tener en cuenta que el modulo Wifi-Bluetooth solo es compatible con Arduino y que se requerirán algunas modificaciones al IDE y librerías para su funcionamiento.

Lo primero que haremos es ir al gestor de tarjetas, en Herramientas > Placa > Gestor de tarjetas...

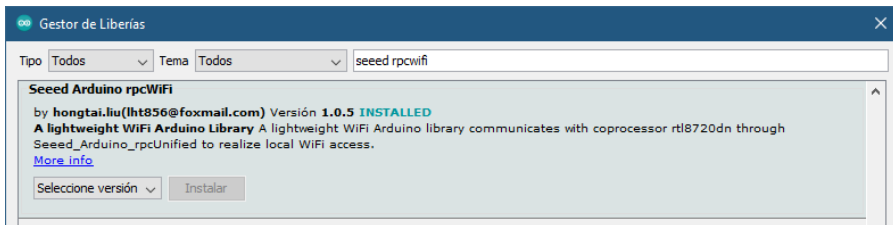


Una vez en el gestor de tarjetas, buscaremos Wio Terminal y tomando en cuenta que se instalo la versión más reciente, en el cuadro de selección de versiones, seleccionamos la versión 1.8.1 y hacemos la instalación de dicha versión.

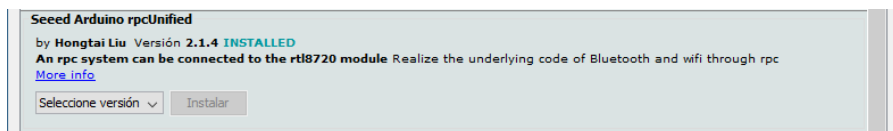


Una vez terminada la instalación de la versión 1.8.1 del Wio Terminal en arduino, instalaremos las librerías.

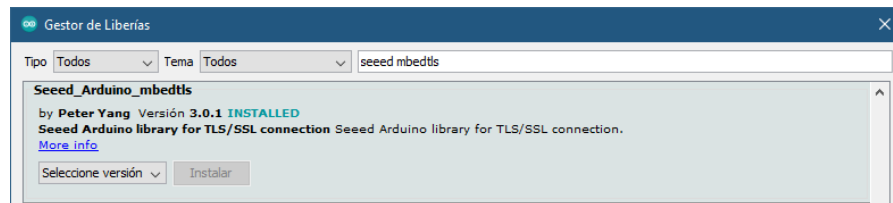
Buscamos “seeed rpcwifi” e instalamos.



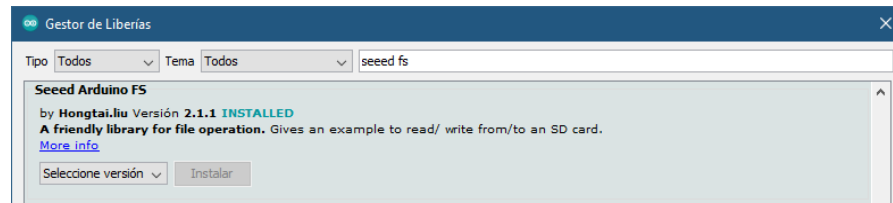
Buscamos “seeed rpcunified” e instalamos.



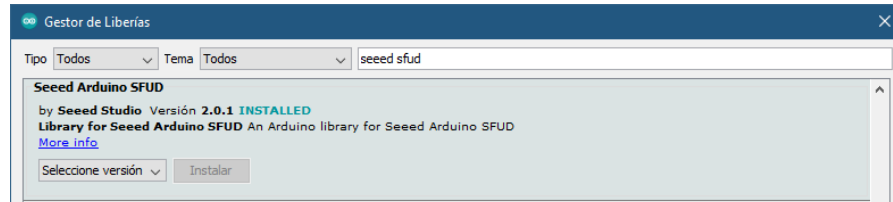
Buscamos “seeed mbedtls” e instalamos.



Buscamos “seeed fs” e instalamos.



Buscamos “seeed sfud” e instalamos.



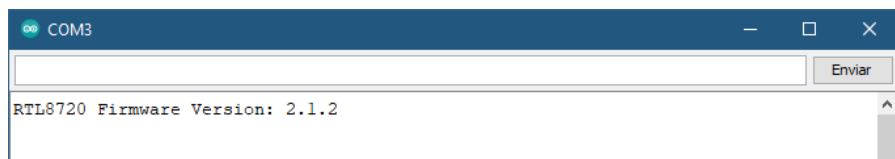
Una vez terminada la instalación de las librerías, copiamos el siguiente código en el IDE de Arduino:

```
#include "rpcWiFi.h" // Importar la librería

void setup() {
    Serial.begin(115200);
    while(!Serial); // Espera a que se abra el Monitor
    Serial
    Serial.printf("RTL8720 Firmware Version: %s",
    rpc_system_version());
}

void loop() {
}
```

Y si el resultado es el siguiente, habremos hecho de forma exitosa la configuración del módulo Wifi-Bluetooth.



Usando el Adaptador Wifi

Escaneando redes Wifi:

```
#include "rpcWiFi.h" // Importamos la librería

void setup() {
    Serial.begin(115200);
    while(!Serial); // Espera al monitor serial abra
    delay(1000);
    WiFi.mode(WIFI_STA); // Establecemos el adaptador
    como Estación
    WiFi.disconnect(); // Desconectamos el adaptador de
    cualquier red en caso de estar conectada
    delay(100);
}
```

```

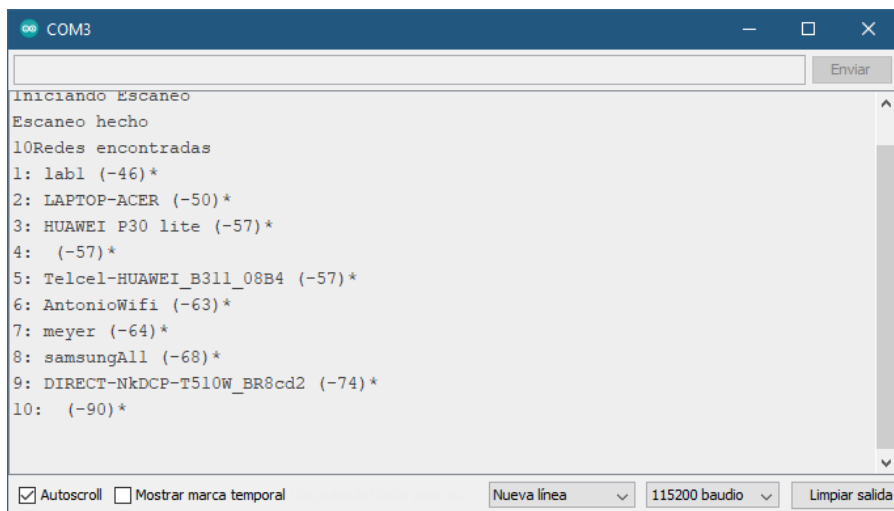
    Serial.println("Configuración hecha");
}

void loop() {
    Serial.println("Iniciando Escaneo");
    int n = WiFi.scanNetworks(); // WiFi.scanNetworks
    //regresa el número de redes encontradas
    Serial.println("Escaneo hecho");
    if (n == 0) {
        Serial.println("No se encontraron redes");
    } else {
        Serial.print(n);
        Serial.println("Redes encontradas");
        for (int i = 0; i < n; ++i) {
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i)); // Imprime el
            // nombre de la red
            Serial.print(" (");
            Serial.print(WiFi.RSSI(i)); // Imprime la
            // potencia de la señal
            Serial.print(")");
            Serial.println((WiFi.encryptionType(i) ==
WIFI_AUTH_OPEN) ? " " : "*"); // Identifica si una red
            // es abierta (" ") o tiene autenticación ("*")
            delay(10);
        }
        Serial.println("");

        delay(5000); // Espera un poco para escanear otra
        // vez
    }
}

```

## Resultado:



The screenshot shows a serial monitor window titled 'COM3'. The text displayed is as follows:

```
Iniciando Escaneo
Escaneo hecho
10Redes encontradas
1: lab1 (-46)*
2: LAPTOP-ACER (-50)*
3: HUAWEI P30 lite (-57)*
4: (-57)*
5: Telcel-HUAWEI_B311_08B4 (-57)*
6: AntonioWifi (-63)*
7: meyer (-64)*
8: samsungAll (-68)*
9: DIRECT-NkDCP-T510W_BR8cd2 (-74)*
10: (-90)*
```

At the bottom of the window, there are control options: ☒ Autoscroll, ☐ Mostrar marca temporal, a 'Nueva línea' dropdown menu, a '115200 baudio' dropdown menu, and a 'Limpiar salida' button.

## Conectarse a una red Wifi:

```
#include "rpcWiFi.h" // Importamos la libreria
```

```
const char* ssid = "redPrueba"; // Escribimos el nombre
de la red a conectarnos
const char* password = "contrasenia"; // Escribimos la
contraseña de la red a conertarnos
```

```
void setup() {
    Serial.begin(115200);
    while(!Serial); // Espera al monitor serial

    WiFi.mode(WIFI_STA); // Establecemos el adaptador
como Estación
    WiFi.disconnect(); // Desconectamos el adaptador de
cualquier red en caso de estar conectada

    Serial.println("Conectando al WiFi..");
    WiFi.begin(ssid, password); // Usa el nombre y
contraseña de red para intentar conectarse
```

```

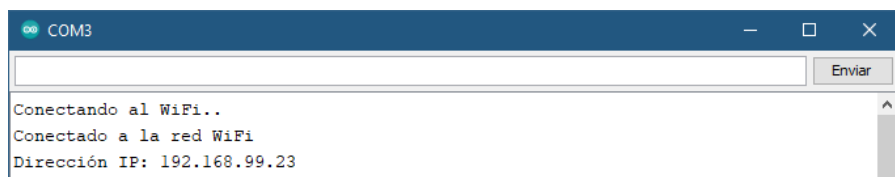
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.println("Conectando al WiFi..");
        WiFi.begin(ssid, password); // Usa el nombre y
        contraseña de red para intentar conectarse
    }
    Serial.println("Conectado a la red WiFi");
    Serial.print("Dirección IP: ");
    Serial.println (WiFi.localIP()); // Imprime la
    dirección IP del Wio Terminal en la red
}

void loop() {

}

```

Resultado:



Crear un servidor Web Local:

```

#include <rpcWiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define LED_BUILTIN 2    // LED de prueba

// Definimos credenciales para el punto de acceso
const char* ssid = "Wio Terminal";
const char* password = "102991299";
int estado = 0;

WiFiServer server(80);

```



```

void setup() {
    pinMode(LED_BUILTIN, OUTPUT);

    Serial.begin(115200);
    while (!Serial); // Wait for Serial to be ready
    delay(1000);
    Serial.println();
    Serial.println("Configurando el punto de acceso...");

    // Puede remover el parametro "password" si se desea
    que la red sea abierta
    WiFi.softAP(ssid, password);
    IPAddress myIP = WiFi.softAPIP();
    /*
        Recuerda la dirección IP, será usada como cliente
        (en el navegador)
    */
    Serial.print("Dirección IP del Punto de Acceso: ");
    Serial.println(myIP);
    server.begin();

    Serial.println("Servidor iniciado");
    digitalWrite(LED_BUILTIN, LOW);
}

void loop() {
    WiFiClient client = server.available(); // Escuchar
    a los clientes entrantes

    if (client) { // Si
    tienes un cliente,
        Serial.println("Nuevo Cliente."); //
    imprimir un mensaje por el puerto serie
        String currentLine = ""; // hacer
    una cadena para contener los datos entrantes del
    cliente

```

```

        while (client.connected()) {           // Bucle
mientras el cliente está conectado
            if (client.available()) {           // Si hay
bytes para leer del cliente,
                digitalWrite(LED_BUILTIN, estado);
                char c = client.read();         // leer un
byte, entonces
                    Serial.write(c);           //
imprimirlo en el monitor serie
                    if (c == '\n') {           // si el
byte es un carácter de nueva línea

                // si la línea actual está en blanco, tiene
dos caracteres de nueva línea seguidos.
                // ese es el final de la solicitud HTTP del
cliente, así que envíe una respuesta:
                    if (currentLine.length() == 0) {
                        // Los encabezados HTTP siempre comienzan
con un código de respuesta (por ejemplo, HTTP/1.1 200
OK)

                        // y un tipo de contenido para que el
cliente sepa lo que viene, luego una línea en blanco:
                        client.println("HTTP/1.1 200 OK");
                        client.println("Content-type:text/html");
                        client.println();

                        // el contenido de la respuesta HTTP sigue
al encabezado:
                        client.print("Pica <a href=\"/H\">aquí</a>
para ENCENDER el LED.<br>");
                        client.print("Pica <a href=\"/L\">aquí</a>
para APAGAR el LED.<br>");

                        // La respuesta HTTP termina con otra línea
en blanco:
                        client.println();

```

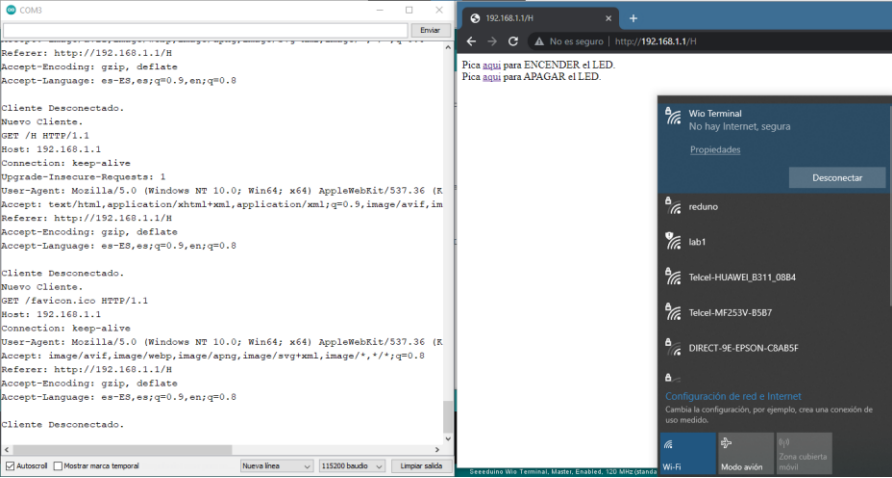
```

        // salir del ciclo while:
        break;
    } else {    // si tiene una línea nueva,
borre la línea actual:
        currentLine = "";
    }
    } else if (c != '\r') { // si tiene algo más
que un carácter de retorno,
        currentLine += c;    // agréguelo al final
de la línea actual
    }

    // Verifique si la solicitud del cliente fue
"GET /H" o "GET /L":
    if (currentLine.endsWith("GET /H")) {
        digitalWrite(LED_BUILTIN, HIGH);
// GET /H enciende el LED
        estado = 1;
    }
    if (currentLine.endsWith("GET /L")) {
        digitalWrite(LED_BUILTIN, LOW);
// GET /L apaga el LED
        estado = 0;
    }
}
}
// Cierre de conexión:
client.stop();
Serial.println("Cliente Desconectado.");
}
}

```

Resultado:



## Bluetooth (Compatible solo con Arduino)

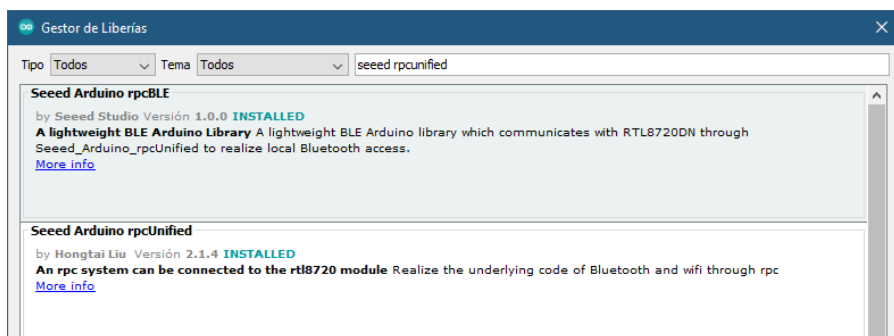
Antes de empezar con el uso del Bluetooth del Wio Terminal, se debe de seguir lo que esta en el apartado Uso por primera vez en el capítulo anterior (Wifi), en caso de ya haber realizado el flasheo y la configuración del IDE Arduino, se puede proceder con lo siguiente.

### Antes de empezar

Antes de empezar con el uso del acelerómetro, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías que pueden ser importadas desde las bibliotecas en el gestor de librerías lo siguiente:

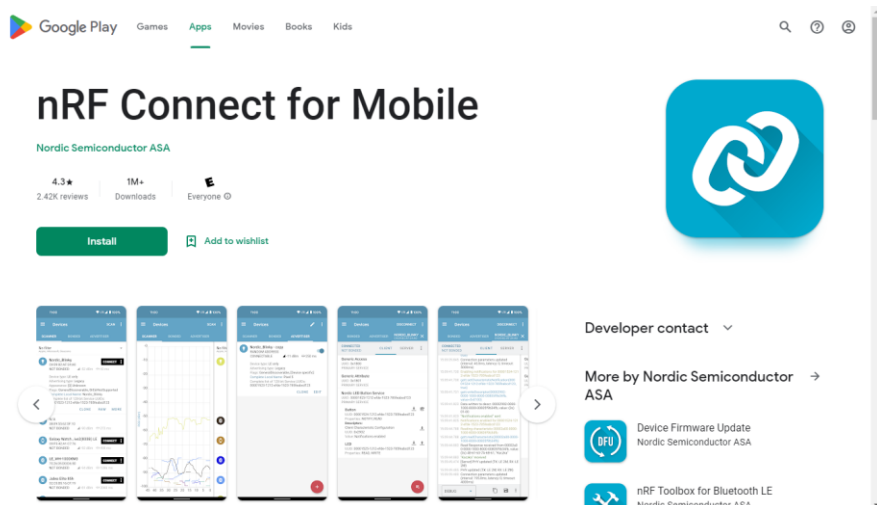
Buscamos “seeed rpcble” e instalamos.

Buscamos “seeed rpcunified” e instalamos.



Y con ello tendríamos lo necesario para trabajar con Bluetooth desde el Wio Terminal por medio de Arduino.

Por cierto, será necesaria una aplicación para el envío de mensajes desde el celular, por lo que usaremos la aplicación “nRF Connect for Mobile” disponible solo para Android en la Google Play Store.



Usando el Adaptador Bluetooth

```
#include "rpcBLEDevice.h"
#include <BLE2902.h>
#include <TFT_eSPI.h> // Biblioteca específica de hardware
#include <SPI.h>
TFT_eSPI tft = TFT_eSPI(); // Invocar biblioteca personalizada
TFT_eSprite spr = TFT_eSprite(&tft); // Sprite
```

```
BLEServer *pServer = NULL;
BLECharacteristic * pTxCharacteristic;
bool deviceConnected = false;
bool oldDeviceConnected = false;
String Value11;
```

```
#define SERVICE_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E" // UUID del servicio UART
```

```

#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-
E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-
E0A9-E50E24DCCA9E"

class MyServerCallbacks: public BLEServerCallbacks {

    // Mensaje en caso de que se conecte de forma
    exitosa
    void onConnect(BLEServer* pServer) {
        deviceConnected = true;
        spr.fillSprite(TFT_BLACK);
        spr.createSprite(240, 100);
        spr.setTextColor(TFT_WHITE, TFT_BLACK);
        spr.drawString("Mensaje: ", 20, 70);
        spr.setTextColor(TFT_GREEN, TFT_BLACK);
        spr.drawString("Estado: Conectado", 10 , 5);
        spr.pushSprite(0, 0);
    };

    // Mensaje en caso de desconexión
    void onDisconnect(BLEServer* pServer) {
        deviceConnected = false;
        Serial.print("123123");
        spr.fillSprite(TFT_BLACK);
        spr.createSprite(240, 100);
        spr.setTextColor(TFT_WHITE, TFT_BLACK);
        spr.drawString("Mensaje: ", 20, 70);
        spr.setTextColor(TFT_RED, TFT_BLACK);
        spr.drawString("Estado: desconectado", 10 , 5);
        spr.pushSprite(0, 0);
    }
};

class MyCallbacks: public BLECharacteristicCallbacks {
    // Imprime el mensaje que se envia desde el celular

```

```

        void onWrite(BLECharacteristic *pCharacteristic) {
            std::string rxValue = pCharacteristic-
>getValue();

            if (rxValue.length() > 0) {
                spr.fillSprite(TFT_BLACK);
                spr.setTextColor(TFT_WHITE, TFT_BLACK);
                for (int i = 0; i < rxValue.length(); i++) {
                    // Serial.print(rxValue[i]);
                    spr.drawString((String)rxValue[i], 10 + i *
15, 0);
                    spr.pushSprite(10, 100);
                }
            }
        };

void setup() {
    tft.begin();
    tft.init();
    tft.setRotation(3);
    tft.fillScreen(TFT_BLACK);
    spr.setTextSize(2);

    BLEDevice::init("Wio Terminal"); //Define el nombre
del dispositivo

    // Crear el servidor BLE
    pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());
    BLEService *pService = pServer-
>createService(SERVICE_UUID);

    // Crear una característica BLE
    pTxCharacteristic = pService->createCharacteristic(
        CHARACTERISTIC_UUID_TX,

```



```

BLECharacteristic::PROPERTY_NOTIFY |
BLECharacteristic::PROPERTY_READ
    );
    pTxCharacteristic->
>setAccessPermissions(GATT_PERM_READ);
    pTxCharacteristic->addDescriptor(new BLE2902());

    BLECharacteristic * pRxCharacteristic = pService->
>createCharacteristic(
        CHARACTERISTIC_UUID_RX,
        BLECharacteristic::PROPERTY_WRITE

    );

    pRxCharacteristic->
>setAccessPermissions(GATT_PERM_READ |
GATT_PERM_WRITE);

    pRxCharacteristic->setCallbacks(new MyCallbacks());

    // Iniciar el servicio
    pService->start();

    // Empezar a anunciar
    pServer->getAdvertising()->start();
    spr.fillSprite(TFT_BLACK);
    spr.createSprite(240, 100);
    spr.setTextColor(TFT_WHITE, TFT_BLACK);
    spr.drawString("Estado: Desconectado", 10 , 5);
    spr.drawString("Mensaje: ", 20, 70);
    spr.pushSprite(0, 0);
}

void loop() {
    // Desconectando
    if (!deviceConnected && oldDeviceConnected) {

```

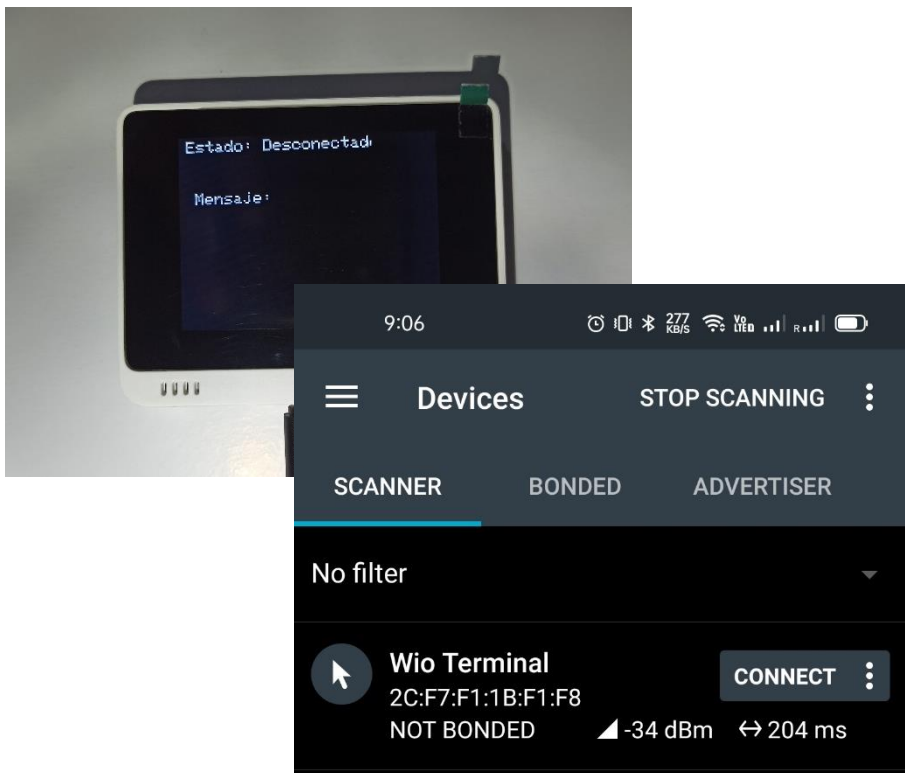
```

    delay(500); // dale a la pila bluetooth la
    oportunidad de preparar las cosas
    pServer->startAdvertising(); // reiniciar la
    publicidad
    oldDeviceConnected = deviceConnected;
}
// Conectando
if (deviceConnected && !oldDeviceConnected) {
    // hacer cosas aquí en la conexión
    oldDeviceConnected = deviceConnected;
}
}

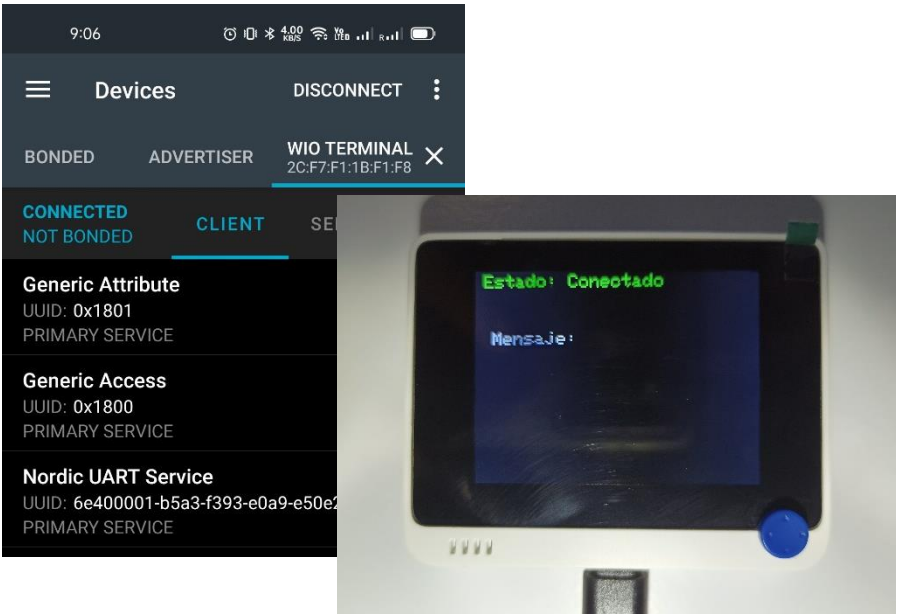
```

Resultado:

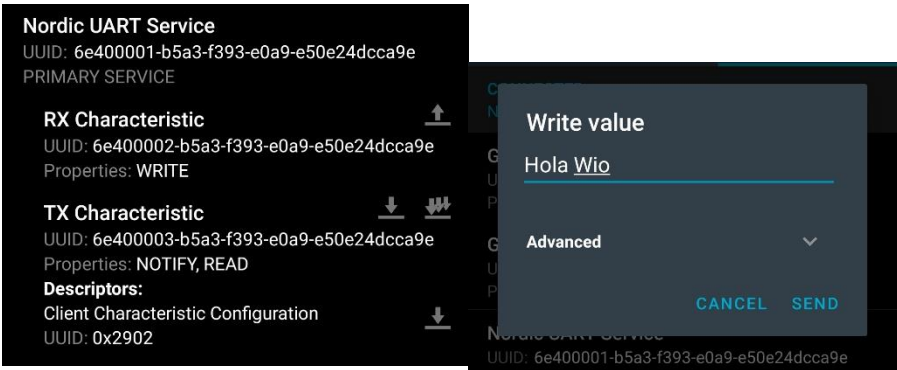
Al conectar el Wio Terminal, abre la conexión Bluetooth., abrimos la aplicación descargada y damos a “CONNECT”.



Una vez conectada, cambiara el mensaje del Wio a Conectado, y nos aparecerán las opciones genéricas y el “Nordic UART Service”.



Desplegamos Nordic UART Service, abrimos RX Characteristic picando a la flecha hacia arriba y le establecemos un valor que será el mensaje para enviar.



Una vez hecho el cambio de valor, será enviado al Wio Terminal.

### Nordic UART Service

UUID: 6e400001-b5a3-f393-e0a9-e50e24dcca9e

PRIMARY SERVICE

### RX Characteristic



UUID: 6e400002-b5a3-f393-e0a9-e50e24dcca9e

Properties: WRITE

Value: Hola Wio



## Acelerómetro

El acelerómetro en el Wio Terminal es un sensor el cual mide la aceleración del dispositivo en 3 ejes: X, Y, Z.

### Antes de empezar

Antes de empezar con el uso del acelerómetro, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías requeridas que se encuentra en [el repositorio de GitHub](https://github.com/Seeed-Studio/Seeed_Arduino_LIS3DHTR). Y descargamos todos los elementos en un ZIP.

Link de descarga: [https://github.com/Seeed-Studio/Seeed\\_Arduino\\_LIS3DHTR/tree/master](https://github.com/Seeed-Studio/Seeed_Arduino_LIS3DHTR/tree/master)

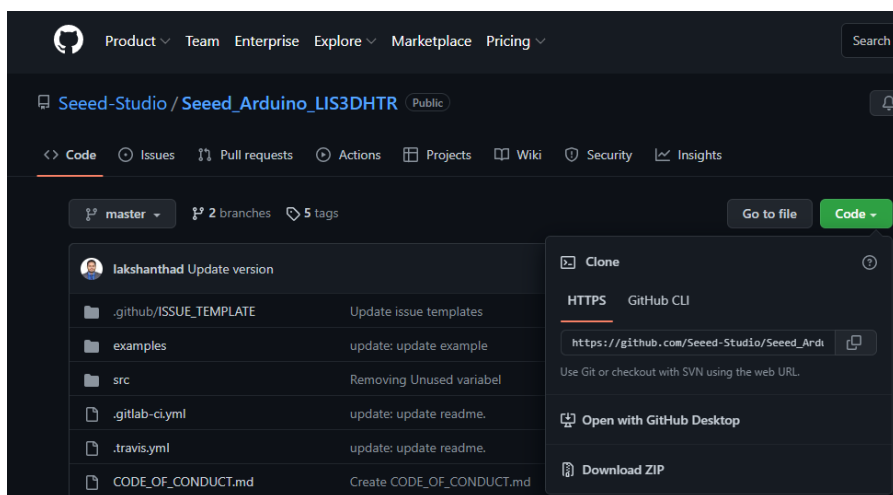


Ilustración 10: Pagina de descarga de librería del acelerómetro en GitHub

Después en Arduino vamos a Programa > Incluir Librería > Añadir biblioteca .ZIP... Y añadimos la librería.

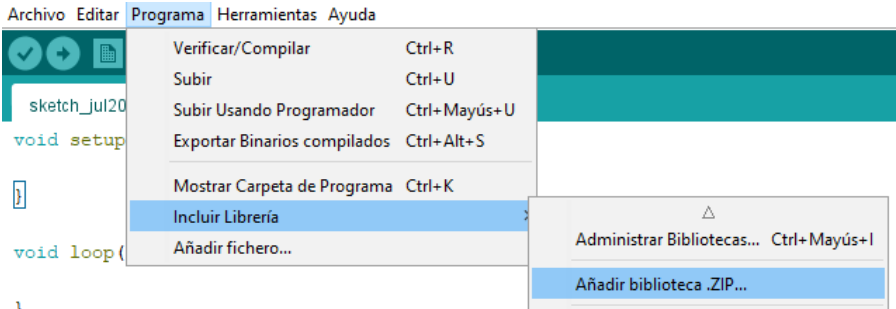


Ilustración 11: Pasos antes de la importación de la librería en Arduino

### Entendiendo al acelerómetro en el Wio Terminal

El acelerómetro tiene 3 ejes, los cuales de acuerdo con la posición del Wio en el espacio, cambiarán.

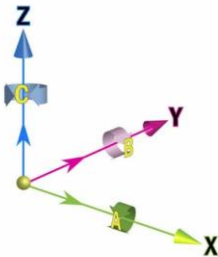


Ilustración 13: Referencia de los ejes XYZ y su rotación



Ilustración 12: Imagen del Wio de acuerdo con la referencia de los ejes

Tomando en cuenta que el Wio Terminal está con la pantalla de frente a quien lo manipula, estos serían los casos posibles:

El frente del eje X son los 3 botones configurables por el usuario, en el estado supuesto por defecto se encuentra apuntando hacia arriba (1.0), si se acuesta el Wio se encuentra en posición neutral (0.0) y si el Wio se pone de cabeza su posición será negativa (-1.0).

El frente del eje Y es el logo de 'seed', en el estado supuesto por defecto se encuentra apuntando hacia la derecha (0.0); cuando éste empieza a

rotar hacia la izquierda hasta llegar el frente a arriba, su posición es 1.0; y cuando éste empieza a rotar hacia la derecha hasta llegar el frente abajo, su posición es -1.0.

El frente del eje Z son los 40 pines, en el estado supuesto por defecto se encuentra apuntando hacia en frente en posición neutral (0.0); si el Wio rota hasta ‘acostarse’ con la pantalla hacia arriba, su valor será negativo (-1.0); y si el Wio rota hasta ‘acostarse’ con la pantalla hacia abajo, su valor será positivo (1.0).

Usando el acelerómetro

En Arduino:

```
#include "LIS3DHTR.h" // Importamos la libreria
LIS3DHTR<TwoWire> lis; // Creamos el objero

void setup() {
  Serial.begin(115200); // Definimos la velocidad de
  transmisión de datos
  lis.begin(Wire1); // Iniciamos al acelerometro

  if (!lis) { // Si no está funcional marcará error
    Serial.println("ERROR");
    while(1);
  }
  lis.setOutputDataRate(LIS3DHTR_DATARATE_25HZ); //Tasa
  de datos de salida
  lis.setFullScaleRange(LIS3DHTR_RANGE_2G); //Confugura
  el rango de escala 2g
}

void loop() {
  /* Se definen variables que almacenen la posición y
  se leen posteriormente los valores */
  float x_values, y_values, z_values;
  x_values = lis.getAccelerationX();
  y_values = lis.getAccelerationY();z
```

```
z_values = lis.getAccelerationZ());

/* Se imprimen los valores de las posiciones */
Serial.print("X: "); Serial.print(x_values);
Serial.print(" Y: "); Serial.print(y_values);
Serial.print(" Z: "); Serial.print(z_values);
Serial.println();
delay(1000); // Se hace una pausa de 1000
milisegundos (1 segundo)
}
```

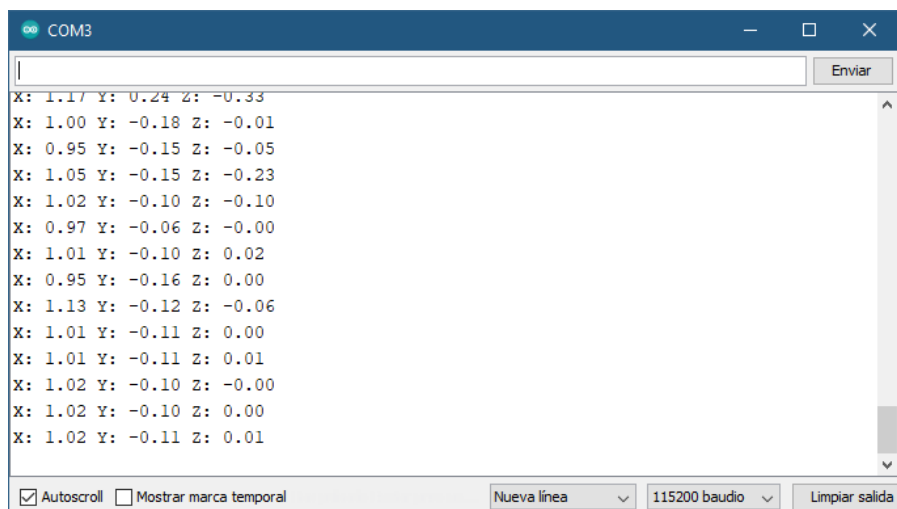


Ilustración 14: Resultado de la posición del Wio en el ejemplo propuesto



## Botones Configurables

Los botones configurables están disponibles en el Wio Terminal, lo cual nos facilita la interacción con elementos, ahorrando cables, botones y conexiones a los pines del Wio Terminal.



*Ilustración 15: Imagen del Wio Terminal visto desde arriba*

El orden de los botones es interesante, lo intuitivo sería que el orden sea de izquierda a derecha, pero no es así; el orden de los botones en este caso va de la derecha a la izquierda, siendo el botón de la izquierda el botón 3 y el botón de la derecha el botón 1.

Al no ser alguna clase de sensor que requiera más componentes de software para ser funcional, su uso puede ser inmediato, sin tener que instalar alguna librería extra.

Se puede usar `BUTTON_1`, `BUTTON_2` y `BUTTON_3` para usar los botones configurables sustituyendo a `"WIO_KEY_"`

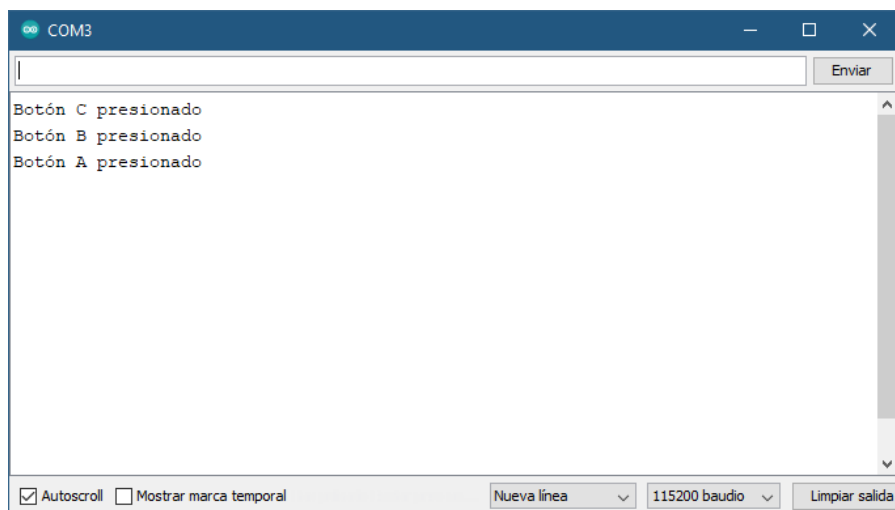
Usando los botones configurables

Para Arduino:

Ejemplo 1: Usar el Serial para proyectar respuestas del botón presionado.

```
void setup() {  
    Serial.begin(115200); // Definimos la velocidad de  
    conexión serial  
    pinMode(WIO_KEY_A, INPUT_PULLUP); // Se declara al  
    PIN del Botón A como entrada con valor verdadero  
    pinMode(WIO_KEY_B, INPUT_PULLUP); // Se declara al  
    PIN del Botón B como entrada con valor verdadero  
    pinMode(WIO_KEY_C, INPUT_PULLUP); // Se declara al  
    PIN del Botón C como entrada con valor verdadero  
}  
void loop() {  
    if (digitalRead(WIO_KEY_A) == LOW) { // Condicionamos  
    el estado del botón A, en caso de ser falso lleva a  
    cabo lo que está dentro  
        Serial.println("Botón A presionado"); // Imprimimos  
    en el serial el mensaje  
    }  
    else if (digitalRead(WIO_KEY_B) == LOW) { //  
    Condicionamos el estado del botón B, en caso de ser  
    falso lleva a cabo lo que está dentro  
        Serial.println("Botón B presionado"); // Imprimimos  
    en el serial el mensaje  
    }  
    else if (digitalRead(WIO_KEY_C) == LOW) { //  
    Condicionamos el estado del botón C, en caso de ser  
    falso lleva a cabo lo que está dentro  
        Serial.println("Botón C presionado"); // Imprimimos  
    en el serial el mensaje  
    }  
    delay(200);  
}
```

Resultado:



*Ilustración 16: Resultado de presionar de orden izquierdo a derecho los botones.*

Ejemplo 2: Usar la pantalla y los colores RGB para mostrar la respuesta del último botón que se presionó.

```
#include"TFT_eSPI.h" // Importamos la libreria del  
LCD_TFT  
TFT_eSPI tft; // Creamos el objeto de la libreria del  
LCD_TFT
```

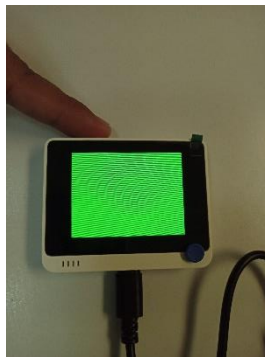
```
void setup() {  
    tft.begin(); // Inicializamos el objeto  
    tft.setRotation(3); // Decimos que rotación será  
    Serial.begin(115200); // Definimos la velocidad de  
    conexión serial  
    pinMode(WIO_KEY_A, INPUT_PULLUP); // Se declara al  
    PIN del Botón A como entrada con valor verdadero  
    pinMode(WIO_KEY_B, INPUT_PULLUP); // Se declara al  
    PIN del Botón B como entrada con valor verdadero  
    pinMode(WIO_KEY_C, INPUT_PULLUP); // Se declara al  
    PIN del Botón C como entrada con valor verdadero  
}
```

```

void loop() {
    if (digitalRead(WIO_KEY_A) == LOW) { // Condicionamos
el estado del botón, en caso de ser falso lleva a cabo
lo que está dentro
        tft.fillScreen(TFT_RED); // Ponemos la pantalla en
rojo
    }
    else if (digitalRead(WIO_KEY_B) == LOW) { //
Condicionamos el estado del botón, en caso de ser falso
lleva a cabo lo que está dentro
        tft.fillScreen(TFT_GREEN); // Ponemos la pantalla
en verde
    }
    else if (digitalRead(WIO_KEY_C) == LOW) { //
Condicionamos el estado del botón, en caso de ser falso
lleva a cabo lo que está dentro
        tft.fillScreen(TFT_BLUE); // Ponemos la pantalla en
azul
    }
    delay(100); // Pausamos 100 milisegundos (0.1
segundos)
}

```

Resultado:



Para ArduPy:

Ejemplo 1: Proyectar respuestas del botón presionado.

```
from machine import Pin, Map # Importamos las librerías
PIN y MAP
import time # Importamos la librería de tiempo

btnA = Pin(Map.WIO_KEY_A, # Definimos el Pin a usar
            Pin.IN, # Definimos que será un pin de entrada
            Pin.PULL_UP) # Definimos el estado en el que
será leído
btnB = Pin(Map.WIO_KEY_B,
            Pin.IN,
            Pin.PULL_UP)
btnC = Pin(Map.WIO_KEY_C,
            Pin.IN,
            Pin.PULL_UP)

while True:
    time.sleep_ms(200) # Establecemos 200 milisegundos
(0.2 segundos) como pausa para la lectura
    if btnA.value() == 0:
        print("Botón A presionado")
    elif btnB.value() == 0:
        print("Botón B presionado")
    elif btnC.value() == 0:
        print("Botón C presionado")
```

## Resultado:



The screenshot shows the Thonny IDE interface. The top menu bar includes 'Fichero', 'Editar', 'Visualización', 'Ejecutar', 'Herramientas', and 'Ayuda'. Below the menu is a toolbar with icons for file operations, running, and debugging. The main editor window displays a file named 'main.py' with the following Python code:

```
13
14 while True:
15     time.sleep_ms(200) # Establecemos 200 milisegundos (0.2 segundos) como pau
16     if btnA.value() == 0:
17         print("Botón A presionado")
18     elif btnB.value() == 0:
19         print("Botón B presionado")
20     elif btnC.value() == 0:
21         print("Botón C presionado")
```

Below the editor is a console window titled 'main.py output:' showing the output of the script:

```
Botón C presionado
Botón B presionado
Botón B presionado
Botón A presionado
```

The status bar at the bottom right indicates 'MicroPython (genérico)'.

Ejemplo 2: Usar la pantalla y los colores RGB para mostrar la respuesta del último botón que se presionó.

```
from machine import Pin, Map, LCD # Importamos las
librerías PIN, MAP, LCD
import time # Importamos la librería de tiempo

lcd = LCD() # Iniciamos la LCD y encendemos el fondo

btnA = Pin(Map.WIO_KEY_A, # Definimos el Pin a usar
            Pin.IN, # Definimos que será un pin de entrada
            Pin.PULL_UP) # Definimos el estado en el que
# será leído
btnB = Pin(Map.WIO_KEY_B,
            Pin.IN,
            Pin.PULL_UP)
btnC = Pin(Map.WIO_KEY_C,
            Pin.IN,
```

```
Pin.PULL_UP)
```

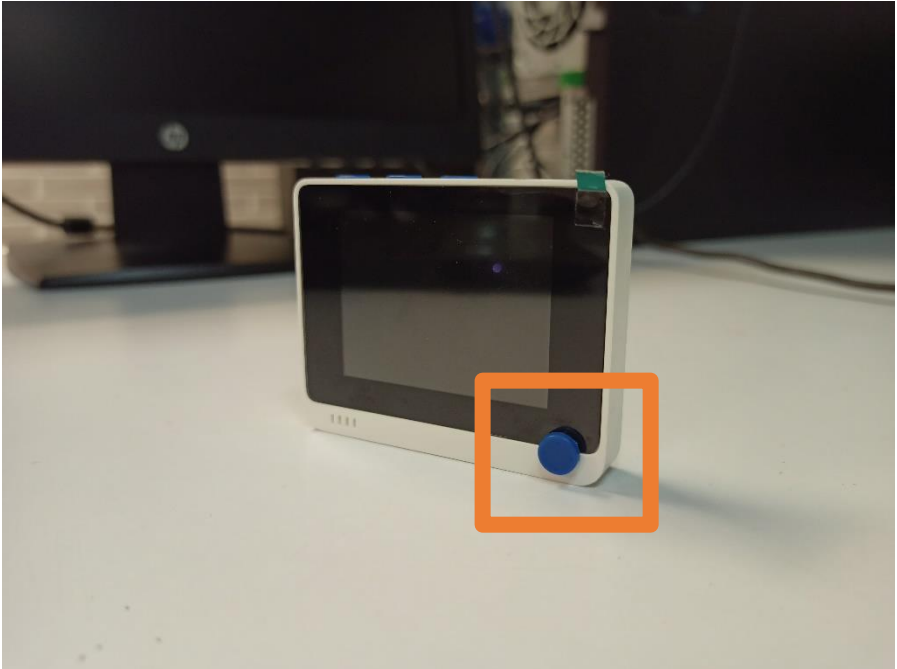
```
while True:  
    time.sleep_ms(200) # Establecemos 200 milisegundos  
    (0.2 segundos) como pausa para la lectura  
    if btnA.value() == 0:  
        lcd.fillScreen(lcd.color.RED)  
    elif btnB.value() == 0:  
        lcd.fillScreen(lcd.color.GREEN)  
    elif btnC.value() == 0:  
        lcd.fillScreen(lcd.color.BLUE)
```

Resultado:



## Switch de 5 posiciones (Mini Joystick)

El Switch de 5 posiciones o también nombrado mini joystick, es uno de los elementos que más destacan en el Wio Terminal y no solo por su funcionalidad, si no por el hecho que se encuentra junto a la pantalla, lo que lo hace un elemento más visible para el usuario.



Dicho Switch tiene 5 posiciones: hacia arriba, hacia abajo, hacia la izquierda, hacia la derecha y hacia adentro (presionado).

WIO\_5S\_UP, WIO\_5S\_DOWN, WIO\_5S\_LEFT, WIO\_5S\_RIGHT y WIO\_5S\_PRESS están definidos para el interruptor de 5 vías de la terminal Wio.

Usando el Switch de 5 posiciones

Para Arduino:

```
void setup() {  
  Serial.begin(115200);  
  pinMode(WIO_5S_UP, INPUT_PULLUP);  
}
```



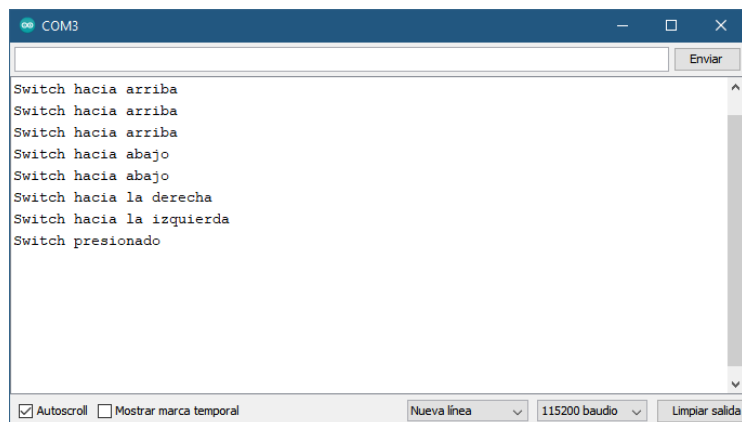
```

pinMode(WIO_5S_DOWN, INPUT_PULLUP);
pinMode(WIO_5S_LEFT, INPUT_PULLUP);
pinMode(WIO_5S_RIGHT, INPUT_PULLUP);
pinMode(WIO_5S_PRESS, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(WIO_5S_UP) == LOW) {
    Serial.println("Switch hacia arriba");
  }
  else if (digitalRead(WIO_5S_DOWN) == LOW) {
    Serial.println("Switch hacia abajo");
  }
  else if (digitalRead(WIO_5S_LEFT) == LOW) {
    Serial.println("Switch hacia la izquierda");
  }
  else if (digitalRead(WIO_5S_RIGHT) == LOW) {
    Serial.println("Switch hacia la derecha");
  }
  else if (digitalRead(WIO_5S_PRESS) == LOW) {
    Serial.println("Switch presionado");
  }
  delay(200);
}

```

Resultado:



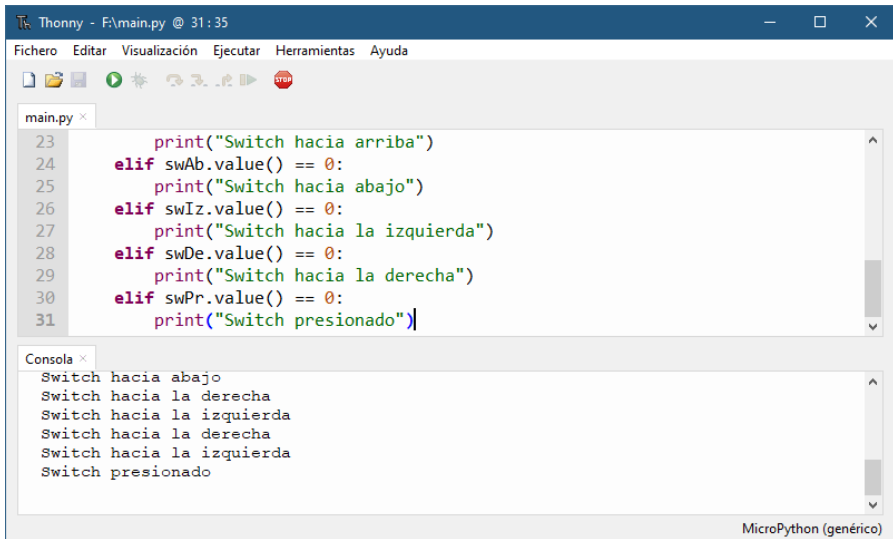
Para ArduPy:

```
from machine import Pin, Map # Importamos las librerías
PIN y MAP
import time # Importamos la librería de tiempo

swAr = Pin(Map.WIO_5S_UP, # Definimos el Pin a usar
hacineod uso de Map
    Pin.IN, # Definimos que será un pin de entrada
    Pin.PULL_UP) # Definimos el estado en el que
será leído
swAb = Pin(Map.WIO_5S_DOWN,
    Pin.IN,
    Pin.PULL_UP)
swIz = Pin(Map.WIO_5S_LEFT,
    Pin.IN,
    Pin.PULL_UP)
swDe = Pin(Map.WIO_5S_RIGHT,
    Pin.IN,
    Pin.PULL_UP)
swPr = Pin(Map.WIO_5S_PRESS,
    Pin.IN,
    Pin.PULL_UP)

while True:
    time.sleep_ms(200) # Establecemos 200 milisegundos
(0.2 segundos) como pausa para la lectura
    if swAr.value() == 0:
        print("Switch hacia arriba")
    elif swAb.value() == 0:
        print("Switch hacia abajo")
    elif swIz.value() == 0:
        print("Switch hacia la izquierda")
    elif swDe.value() == 0:
        print("Switch hacia la derecha")
    elif swPr.value() == 0:
        print("Switch presionado")
```

## Resultado:



The screenshot shows the Thonny IDE interface. The title bar indicates the file is 'F:\main.py' at 31:35. The menu bar includes 'Fichero', 'Editar', 'Visualización', 'Ejecutar', 'Herramientas', and 'Ayuda'. The toolbar contains icons for file operations, running, and debugging. The editor window shows a Python script with the following code:

```
23     print("Switch hacia arriba")
24     elif swAb.value() == 0:
25         print("Switch hacia abajo")
26     elif swIz.value() == 0:
27         print("Switch hacia la izquierda")
28     elif swDe.value() == 0:
29         print("Switch hacia la derecha")
30     elif swPr.value() == 0:
31         print("Switch presionado")
```

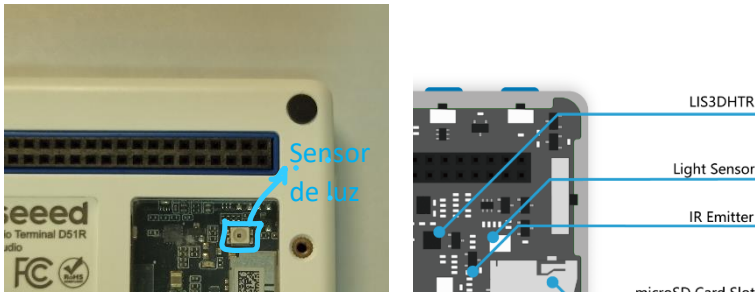
The console window at the bottom shows the output of the script:

```
Switch hacia abajo
Switch hacia la derecha
Switch hacia la izquierda
Switch hacia la derecha
Switch hacia la izquierda
Switch presionado
```

The status bar at the bottom right indicates 'MicroPython (genérico)'.

## Sensor de Luz

El sensor de luz utiliza una interfaz analógica y simplemente puede leer los valores del sensor de luz circundante mediante la lectura de su pin.



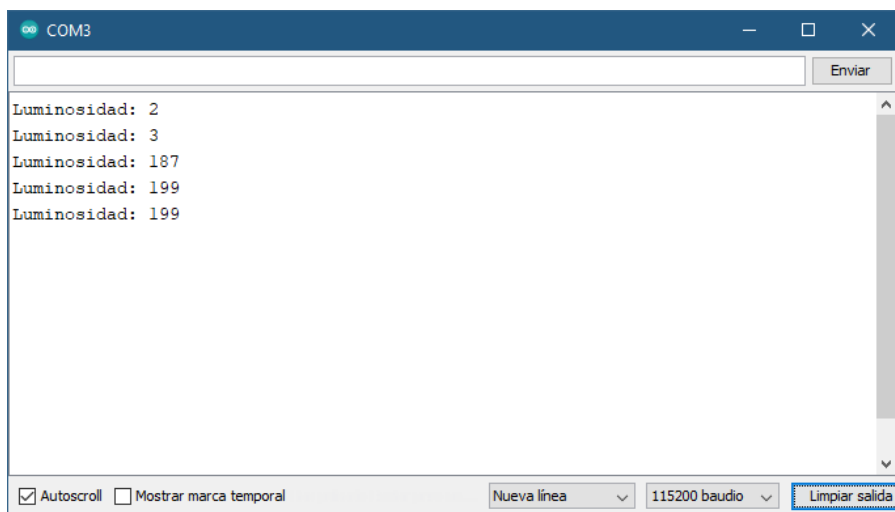
WIO\_LIGHT es el pin para el sensor de luz integrado. El sensor de luz está conectado a A13.

## Usando el Sensor de Luz

Para Arduino:

```
void setup() {  
    pinMode(WIO_LIGHT, INPUT); // Seleccionamos el pin de  
    entrada del sensor de luz  
    Serial.begin(115200); // Establecemos la velocidad de  
    conexión  
}  
  
void loop() {  
    int light = analogRead(WIO_LIGHT); // Leemos el  
    valor del sensor de luz  
    Serial.print("Luminosidad: "); // Imprimimos el  
    mensaje  
    Serial.println(light); // Imprimimos el valor  
    delay(2000); // Pausamos por 2000 milisegundos (2  
    segundos) al programa  
}
```

Resultado:



Para ArduPy:

```
from machine import ADC, Pin, Map # Importamos las
librerías ADC y PIN
import time
adc = ADC(Pin(Map.WIO_LIGHT)) # Creamos el convertidos
ADC en pin del sensor del Wio (Pin)
lumi = adc.read() # Leemos los valores de ADC, 0 ~ 1023
while True:
    print("Luminosidad: " + str(lumi)) # Imprime el
valor del sensor de luz
    time.sleep(2) # Pausamos por 2 segundos
    lumi = adc.read() # Leemos los valores de ADC, 0 ~
1023
```

## Resultado:



The screenshot shows the Thonny IDE interface. The main editor window displays a Python script named `main.py` with the following code:

```
1 from machine import ADC, Pin, Map # Importamos las librerias ADC y PIN
2 import time
3 adc = ADC(Pin(Map.WIO_LIGHT)) # Creamos el convertidos ADC en pin del sensor d
4 lumi = adc.read() # Leemos los valores de ADC, 0 ~ 1023
5 while True:
6     print("Luminosidad: " + str(lumi)) # Imprime el valor del sensor de luz
7     time.sleep(2) # Pausamos por 2 segundos
8     lumi = adc.read() # Leemos los valores de ADC, 0 ~ 1023
```

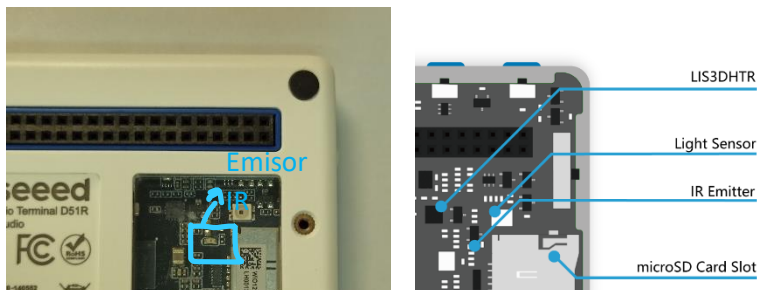
Below the editor, the console window shows the output of the script:

```
Luminosidad: 2
Luminosidad: 2
Luminosidad: 3
Luminosidad: 4
Luminosidad: 132
Luminosidad: 139
```

The status bar at the bottom right indicates "MicroPython (genérico)".

## Emisor de Infrarrojos

El Wio Terminal tiene un emisor de infrarrojos con el cual se pueden controlar distintos elementos que sean compatibles con los mismos, como Televisores, Blurays, Auto Estéreos, Aires acondicionados, etc.

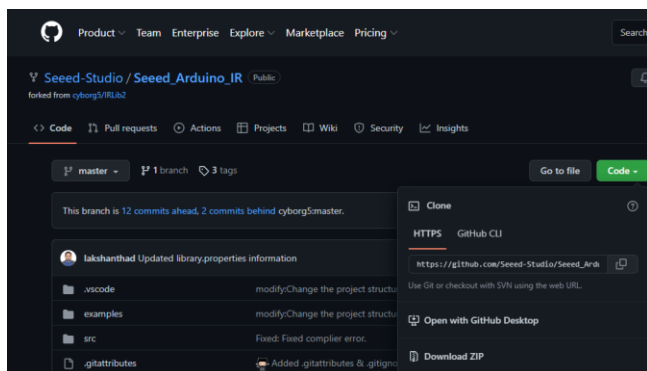


El emisor es un emisor digital infrarrojos, que se puede usar como un control remoto.

### Antes de empezar

Antes de empezar con el uso del acelerómetro, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías requeridas que se encuentra en [el repositorio de GitHub](https://github.com/Seeed-Studio/Seeed_Arduino_IR). Y descargamos todos los elementos en un ZIP.

Link de descarga: [https://github.com/Seeed-Studio/Seeed\\_Arduino\\_IR](https://github.com/Seeed-Studio/Seeed_Arduino_IR)



*Ilustración 17:Pagina de descarga de librería del emisor de infrarrojos en GitHub*

Después en Arduino vamos a Programa > Incluir Librería > Añadir biblioteca .ZIP... E importamos la librería.

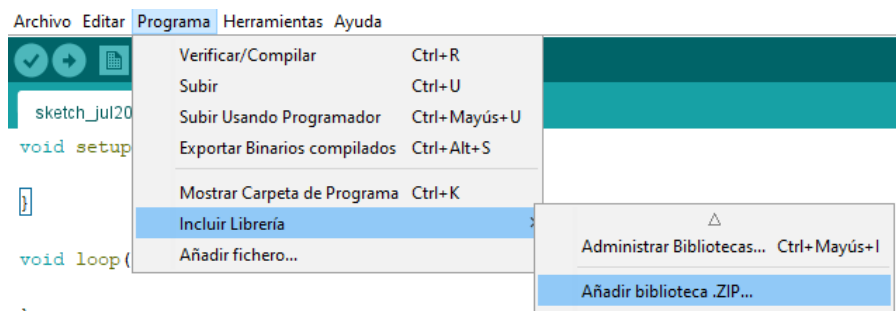


Ilustración 18: Pasos antes de la importación de la librería en Arduino

Usando el emisor de infrarrojos

Ejemplo: Encender un DVD Sony

Para Arduino:

```
#include <IRLibSendBase.h> // Incluimos la librería base
#include <IRLib_P02_Sony.h>
#include <IRLibCombo.h> // Incluimos esta librería
```

```
IRsend mySender;
```

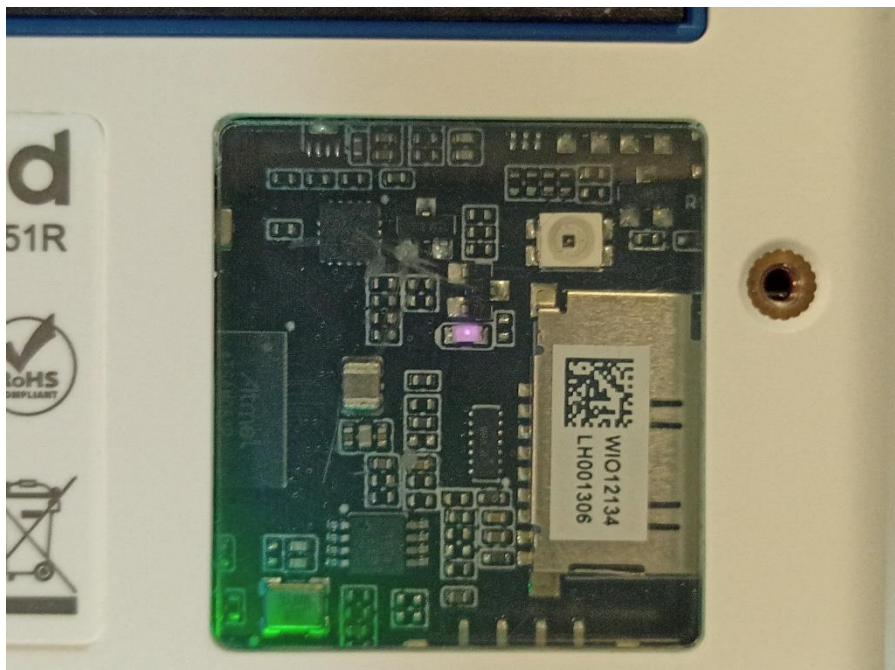
```
void setup() {
    Serial.begin(9600);
    delay(2000); while (!Serial); //Pausamos
    Serial.println(F("Cada que tu presiones el botón
    Enviar, la señal será enviada."));
}
```

```
void loop() {
    if (Serial.read() != -1) {
        mySender.send(SONY,0xa8bca, 20); //Señal de
        encendido Sony DVD A8BCA, de 20 bits
        Serial.println(F("Señal enviada!."));
    }
}
```



}  
}

Resultado:



*Ilustración 19: Emisor IR emitiendo la señal para apagar un DVD Sony*

## Micrófono

El Wio Terminal cuenta con un micrófono integrado, el cual permite la recepción del sonido, el cual es interpretado en formato analógico para el Wio Terminal.



*Ilustración 20: : Ranuras del micrófono/zumbador frente al Wio Terminal*

## Usando el micrófono

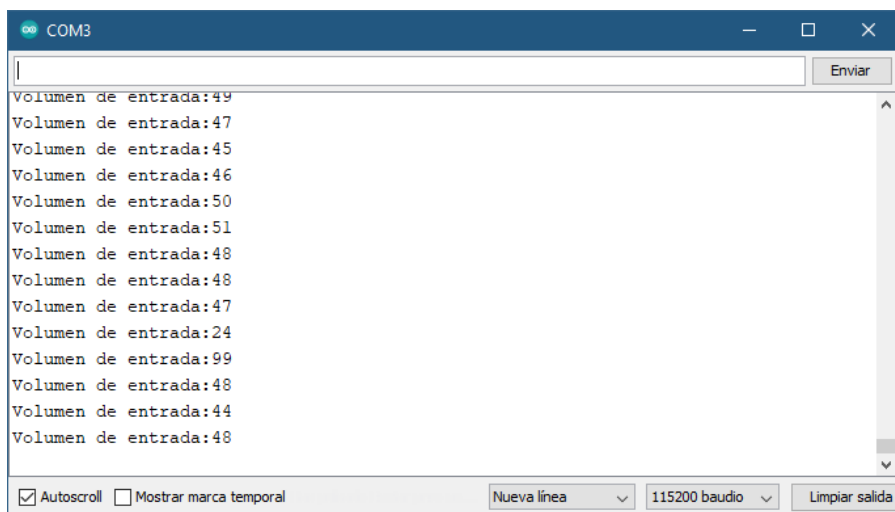
Al no tener que usar alguna clase de librería, podemos usar directamente el pin analógico asignado para el micrófono.

Ejemplo: Leer la entrada de ruido en valores analógicos

En Arduino:

```
void setup() {  
    pinMode(WIO_MIC, INPUT); // Seleccionamos el PIN del  
    Microfono y lo marcamos como dispositivo de entrada  
    Serial.begin(115200); // Marcamos la velocidad de  
    transmisión de datos  
}  
  
void loop() {  
    int val = analogRead(WIO_MIC); // Damos el valor del  
    sensor a una variable  
    Serial.println("Volumen de entrada:" +  
    String((val*100)/1024)); // Imprimimos el valor de la  
    lectura en una escala del 0 al 100  
    delay(1000); // Hacemos una pausa de 1000  
    milisegundos (1 segundo)  
}
```

Resultado:



*Ilustración 21: Monitoreo del volumen de entrada de audio del micrófono*

En ArduPy:

```
from machine import ADC, Pin, Map # Importamos las
librerías ADC y PIN
import time
adc = ADC(Pin(Map.WIO_MIC)) # Creamos el convertidos
ADC en pin del sensor del Wio (Pin)
vol = adc.read() # Leemos los valores de ADC, 0 ~ 1023
while True:
    print("Volumen de entrada: " +
str(((vol*100)/1024))) # Imprime el valor del microfono
    time.sleep(1) # Pausamos por 1 segundos
    vol = adc.read() # Leemos los valores de ADC, 0 ~
1023
```

## Resultado:



The screenshot shows the Thonny IDE interface. The top menu bar includes 'Fichero', 'Editar', 'Visualización', 'Ejecutar', 'Herramientas', and 'Ayuda'. Below the menu is a toolbar with icons for file operations and execution. The main editor window displays a Python script named 'main.py' with the following code:

```
1 from machine import ADC, Pin, Map # Importamos las librerias ADC y PIN
2 import time
3 adc = ADC(Pin(Map.WIO_MIC)) # Creamos el convertidos ADC en pin del sensor del
4 vol = adc.read() # Leemos los valores de ADC, 0 ~ 1023
5 while True:
6     print("Volumen de entrada: " + str(((vol*100)/1024))) # Imprime el valor d
7     time.sleep(1) # Pausamos por 1 segundos
8     vol = adc.read() # Leemos los valores de ADC, 0 ~ 1023
```

Below the editor is a console window titled 'Consola' showing the output of the script:

```
Volumen de entrada: 46.97266
Volumen de entrada: 44.92188
Volumen de entrada: 45.80078
Volumen de entrada: 47.94922
Volumen de entrada: 50.87891
Volumen de entrada: 47.46094
```

The status bar at the bottom right indicates 'MicroPython (genérico)'.

## Zumbador (Buzzer)

El zumbador incorporado de Wio Terminal es un zumbador pasivo, lo que significa que requiere una señal de CA (PWM) para activar y emitir un sonido, por lo que puede generar una salida.



*Ilustración 22: Ranuras del zumbador/micrófono frente al Wio Terminal*

Para generar un sonido de zumbador predeterminado, se recomienda activar el zumbador con un voltaje más bajo.

### Usando el Zumbador

Ejemplo 1: Hacer sonar un *beep* cada segundo de duración de un segundo

Para Arduino:

```
void setup() {  
    pinMode(WIO_BUZZER, OUTPUT); // Habilitamos el  
    zumbador y lo establecemos como salida  
}  
  
void loop() {  
    analogWrite(WIO_BUZZER, 128); // Enviamos un valor  
    que emita el sonido  
    delay(1000); // Pausa de un segundo  
    analogWrite(WIO_BUZZER, 0); // Enviamos un valor para  
    un silencio  
    delay(1000); // Pausa de un segundo  
}
```

Para ArduPy:

```
from machine import DAC, Pin, Map
import time
```

```
buz = DAC(Pin(Map.WIO_BUZZER))
```

```
while True:
    buz.write(127)
    time.sleep(1)
    buz.write(0)
    time.sleep(1)
```

Ejemplo 2: Hacer sonar la melodía “Estrellita, ¿Dónde estás?”

Para Arduino:

```
#define BUZZER_PIN WIO_BUZZER // Definimos un PIN
```

```
int length = 15; // Número de notas
char notes[] = "ccggaagffeeddc "; // Notas
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 }; // Beats
int tempo = 300; // Tempo
```

```
void setup() {
    pinMode(BUZZER_PIN, OUTPUT); // Definimos al Pin
    del zumbador como salida
}
```

```
void loop() {
    for(int i = 0; i < length; i++) { // Hacemos un
        ciclo con la longitud de las notas
        if(notes[i] == ' ') {
            delay(beats[i] * tempo);
        } else {
            playNote(notes[i], beats[i] * tempo); // Se
            envia la nota, el beat y el tempo a la función que
            reproduce una nota
        }
    }
}
```

```

    }
    delay(tempo / 2); // Pausa entre notas
}

}

// Reproduce un tono
void playTone(int tone, int duration) {
    for (long i = 0; i < duration * 1000L; i += tone *
2) {
        digitalWrite(BUZZER_PIN, HIGH);
        delayMicroseconds(tone);
        digitalWrite(BUZZER_PIN, LOW);
        delayMicroseconds(tone);
    }
}

// Reproduce la nota
void playNote(char note, int duration) {
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b',
'C' };
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136,
1014, 956 };

    // Reproduce el tono correspondiente al nombre del
mismo
    for (int i = 0; i < 8; i++) {
        if (names[i] == note) {
            playTone(tones[i], duration);
        }
    }
}

```

Nota: El resultado para ambos casos, es audible

## RTC (Reloj en Tiempo Real o *Real Time Clock*)

Dentro del núcleo SAMD51 del Wio Terminal se tiene la funcionalidad de reloj en tiempo real, lo cual nos ahorra la instalación de un modulo RTC externo a el Wio Terminal. Un reloj de tiempo real (RTC) es un dispositivo electrónico que permite obtener mediciones de tiempo en las unidades temporales que empleamos de forma cotidiana.

### Antes de empezar

Antes de empezar con el uso del acelerómetro, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías requeridas que se encuentra en [el repositorio de GitHub](https://github.com/Seeed-Studio/Seeed_Arduino_RTC). Y descargamos todos los elementos en un ZIP.

Link de descarga: [https://github.com/Seeed-Studio/Seeed\\_Arduino\\_RTC](https://github.com/Seeed-Studio/Seeed_Arduino_RTC)

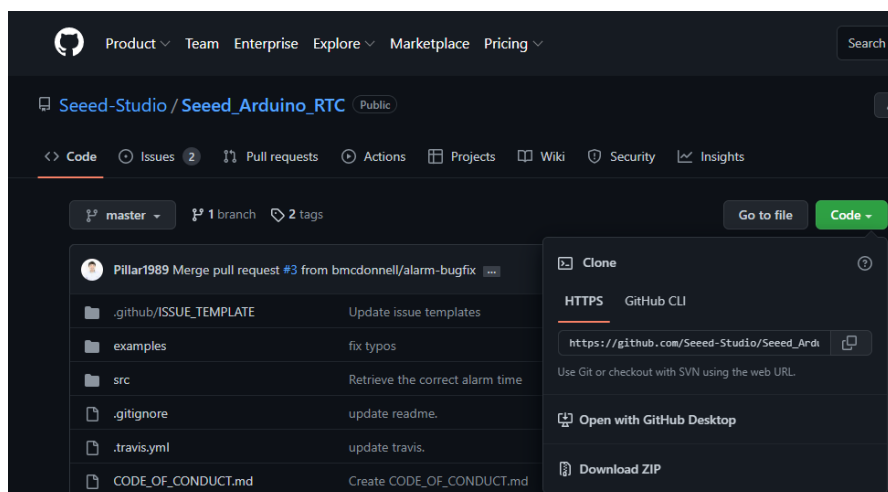


Ilustración 23: Pagina de descarga de librería del RTC en GitHub



Después en Arduino vamos a Programa > Incluir Librería > Añadir biblioteca .ZIP... Y añadimos la librería.

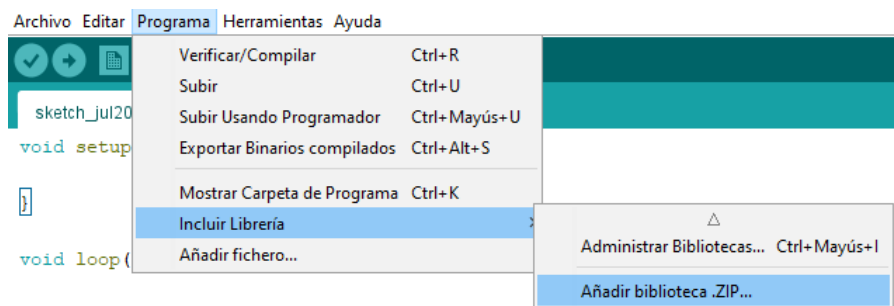


Ilustración 24: Pasos antes de la importación de la librería en Arduino

Usando el RTC

Ejemplo: Ajustar la hora, y establecer una alarma que suene en 15 segundos después del ajuste.

En Arduino

```
#include "RTC_SAMD51.h" // Importamos la librería que interactúa con el Wio
#include "DateTime.h" // Importamos la librería de tiempo
```

```
RTC_SAMD51 rtc; // Creamos el objeto de la librería del Wio
```

```
void setup()
{
```

```
    rtc.begin(); // Iniciamos el objeto del Wio
```

```
    Serial.begin(115200); // Definimos la velocidad de conexión
```

```
    while (!Serial) // Hacemos una espera
    {
        ;
    }
```

```

    DateTime now = DateTime(F(__DATE__), F(__TIME__));
// Importamos la fecha y la hora según el equipo
    Serial.println("¡Fecha ajustada!");
    rtc.adjust(now); // Se realiza el ajuste de la
fecha y la hora

    now = rtc.now(); // Se actualiza

    Serial.print(now.year(), DEC); // Se imprime el año
    Serial.print('/');
    Serial.print(now.month(), DEC); // Se imprime el
mes
    Serial.print('/');
    Serial.print(now.day(), DEC); // Se imprime el día
    Serial.print(" ");
    Serial.print(now.hour(), DEC); // Se imprime la
hora
    Serial.print(':');
    Serial.print(now.minute(), DEC); // Se imprime los
minutos
    Serial.print(':');
    Serial.print(now.second(), DEC); // Se imprimen los
segundos
    Serial.println();

    DateTime alarm = DateTime(now.year(), now.month(),
now.day(), now.hour(), now.minute(), now.second() +
15); // Se establece un objeto alarma a la hora más 15
segundos

    rtc.setAlarm(0,alarm); // Se habilita la alarma con
el objeto anteriormente creado
    rtc.enableAlarm(0, rtc.MATCH_HHMMSS); // Y se
habilita para todos los días

```

```

    rtc.attachInterrupt(alarmMatch); // Devuelve la
    alarma cuando las horas coinciden y llama a una función

}

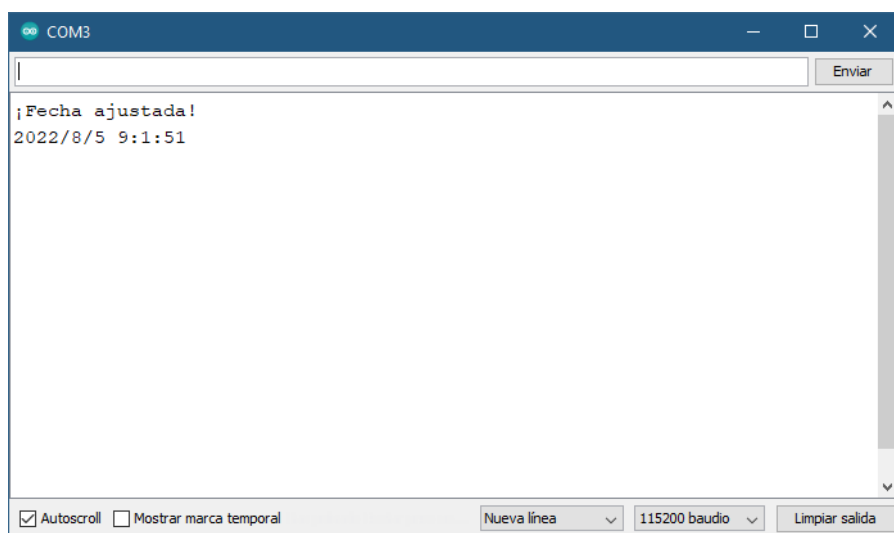
void loop()
{
}

void alarmMatch(uint32_t flag) // En caso de que haya
coincidencia de hora, esta función corre e imprime un
mensaje de la coincidencia, junto a la fecha y hora
{

    Serial.println("Alarma Coincide!");
    DateTime now = rtc.now();
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
}

```

Resultado:



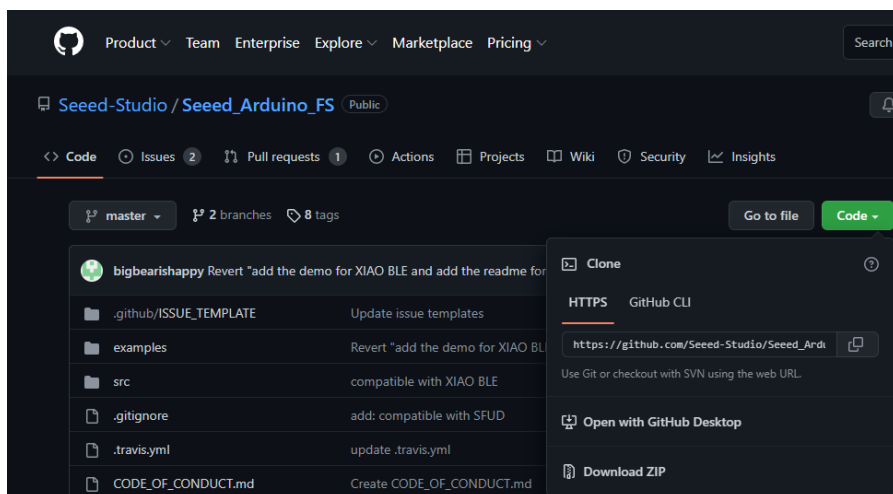
## Sistema de Archivos

Proporciona la funcionalidad básica de archivo que opera con la tarjeta SD, lo que permite leer/escribir en o desde la tarjeta SD mediante la interfaz SPI.

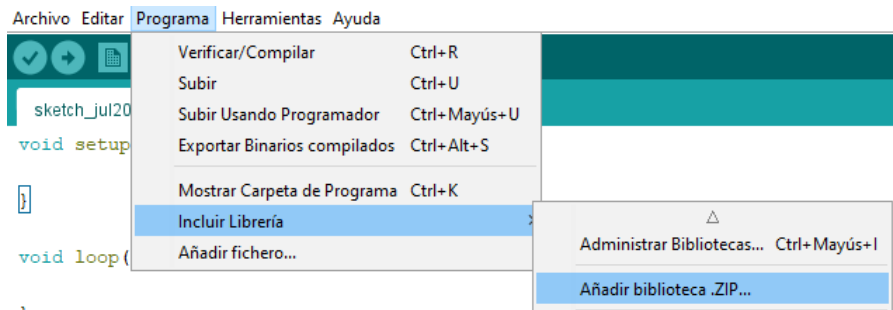
### Antes de empezar

Antes de empezar con el uso del acelerómetro, debemos de tener en cuenta de que no tiene los elementos de precargados en el Software, por lo de deberemos de importar las librerías requeridas que se encuentra en [el repositorio de GitHub](https://github.com/Seeed-Studio/Seeed_Arduino_FS/). Y descargamos todos los elementos en un ZIP.

Link de descarga: [https://github.com/Seeed-Studio/Seeed\\_Arduino\\_FS/](https://github.com/Seeed-Studio/Seeed_Arduino_FS/)



Después en Arduino vamos a Programa > Incluir Librería > Añadir biblioteca .ZIP... Y añadimos la librería.



Usando el sistema de archivos

Ejemplo: Usar el sistema de archivos para crear un archivo de texto que escriba un mensaje y se visualice en el monitor

```
#include <SPI.h>
#include <Seeed_FS.h>
#include "SD/Seeed_SD.h"

File myFile;

void setup() {
  Serial.begin(115200);
  while (!Serial) {
  }
  Serial.print("Inicializando tarjeta SD...");
  if (!SD.begin(SDCARD_SS_PIN, SDCARD_SPI)) {
    Serial.println("Inicialización fallida!");
    while (1);
  }
  Serial.println("Inicialización hecha.");

  // Abrir el archivo. tenga en cuenta que solo se
  // puede abrir un archivo a la vez,
  // así que tienes que cerrar este antes de abrir
  // otro.
  myFile = SD.open("test.txt", FILE_WRITE);

  // si el archivo se abrió bien, se escribirá:
  if (myFile) {
    Serial.print("Escribiendo en test.txt...");
    myFile.println("probando 1, 2, 3.");
    // Cerrar el archivo:
    myFile.close();
    Serial.println("Escritura terminada. Hecho!");
  } else {
    // si el archivo no se abre, imprime un error:
```

```

        Serial.println("Error abriendo test.txt");
    }

    // volver a abrir el archivo para leer:
    myFile = SD.open("test.txt", FILE_READ);
    if (myFile) {
        Serial.println("test.txt:");

Serial.println("*****INICIO*****");
        // leer del archivo hasta que no haya nada más en
        él:
        while (myFile.available()) {
            Serial.write(myFile.read());
        }

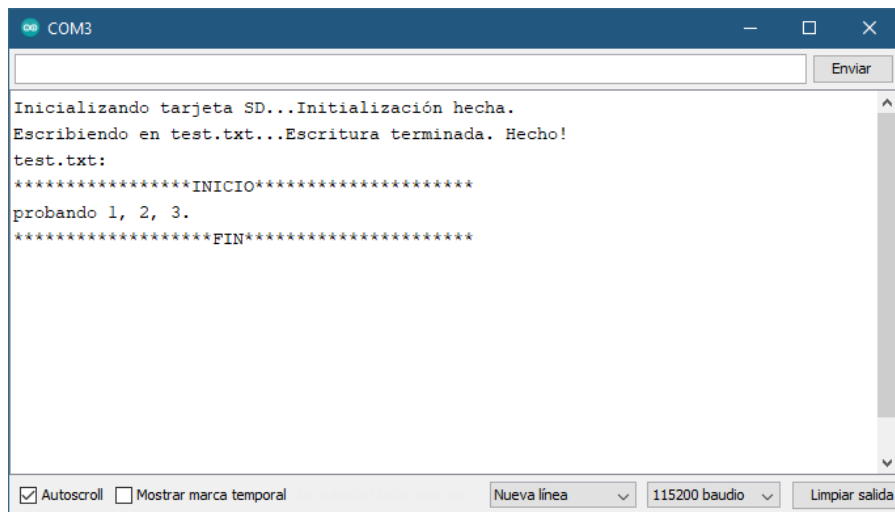
Serial.println("*****FIN*****");
        // cerrar el archivo:
        myFile.close();
    } else {
        // si el archivo no se abre, imprime un error:
        Serial.println("Error abriendo test.txt");
    }
}

void loop() {

}

```

## Resultado:



The screenshot shows a serial monitor window with a dark blue header bar containing the text 'COM3' and standard window control icons. Below the header is a text input field and an 'Enviar' button. The main area displays the following text:

```
Inicializando tarjeta SD...Inicialización hecha.  
Escribiendo en test.txt...Escritura terminada. Hecho!  
test.txt:  
*****INICIO*****  
probando 1, 2, 3.  
*****FIN*****
```

At the bottom of the window, there is a control bar with the following elements from left to right: a checked 'Autoscroll' checkbox, an unchecked 'Mostrar marca temporal' checkbox, a 'Nueva línea' button with a dropdown arrow, a '115200 baudio' button with a dropdown arrow, and a 'Limpiar salida' button.



## Contenido

Cuidados y advertencias .....	3
Visión del Wio Terminal.....	4
Características Destacables .....	4
Especificaciones.....	5
Vistazo Interno a la Wio Terminal .....	7
Dentro de la caja.....	10
Conceptos y términos.....	11
Antes de empezar.....	12
Hardware .....	12
Lo básico antes de continuar .....	12
Para encender.....	12
Para reiniciar.....	12
Para acceder al Bootloader.....	13
Software .....	14
Iniciando en Arduino .....	14
Iniciando en Ardupy.....	17
Funcionamiento de los componentes .....	20
LCD.....	21
Antes de empezar .....	21
Primeros usos de la LCD .....	23
Encender el fondo de la pantalla en algún color .....	26
Apagar el fondo de la pantalla de la LCD.....	28
Funciones graficas básicas.....	29
IO .....	47
Describiendo los Pines.....	47

Usando Pines Digitales.....	50
Usando Pines Analógicos.....	54
Wifi (Compatible solo con Arduino) .....	57
Uso por primera vez .....	57
Antes de empezar .....	59
Usando el Adaptador Wifi .....	61
Bluetooth (Compatible solo con Arduino).....	69
Antes de empezar .....	69
Usando el Adaptador Bluetooth.....	70
Acelerómetro.....	77
Antes de empezar .....	77
Entendiendo al acelerómetro en el Wio Terminal .....	78
Usando el acelerómetro .....	79
Botones Configurables.....	81
Usando los botones configurables .....	82
Switch de 5 posiciones (Mini Joystick).....	88
Usando el Switch de 5 posiciones.....	88
Sensor de Luz.....	92
Usando el Sensor de Luz .....	92
Emisor de Infrarrojos .....	95
Antes de empezar .....	95
Usando el emisor de infrarrojos .....	96
Micrófono .....	98
Usando el micrófono .....	98
Zumbador (Buzzer) .....	101
Usando el Zumbador .....	101

RTC (Reloj en Tiempo Real o <i>Real Time Clock</i> ) .....	104
Antes de empezar .....	104
Usando el RTC.....	105
Sistema de Archivos.....	109
Antes de empezar .....	109
Usando el sistema de archivos .....	110