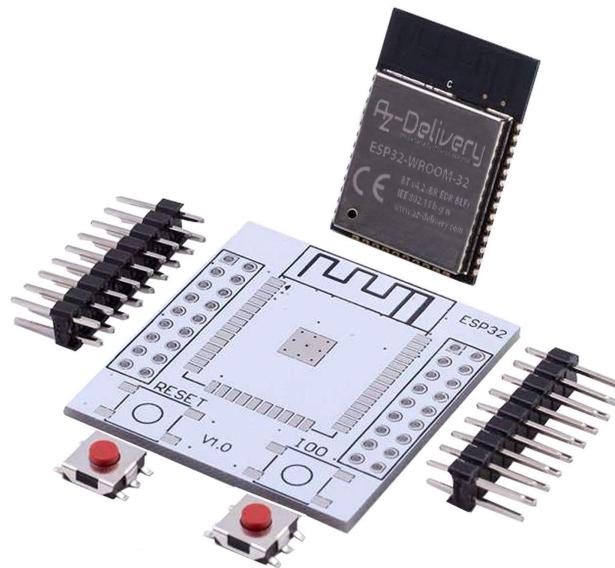


## Bienvenido!

Y muchas gracias por comprar nuestro **AZ-Delivery ESP-32** con tarjeta adaptadora. En las páginas siguientes, le guiaremos para realizar la soldadura del chip ESP32 en la tarjeta adaptadora.

Le deseamos mucha diversión!



Es importante mencionar que aquí se maneja un componente SMD y no es adecuado para principiantes de soldadura. Se debe tener experiencia de soldadura SMD!

Antes de que podamos comenzar a soldar, primero debemos verificar el alcance del suministro:

1x ESP32

1x Adapter Board

2x cabezales de pines (2x9 pines)

2x Microtaster

Si todo ha sido entregado, preparamos nuestro material. Necesitará:

Soldador con punta fina

Soldadura para electrónica (fundente en el núcleo)

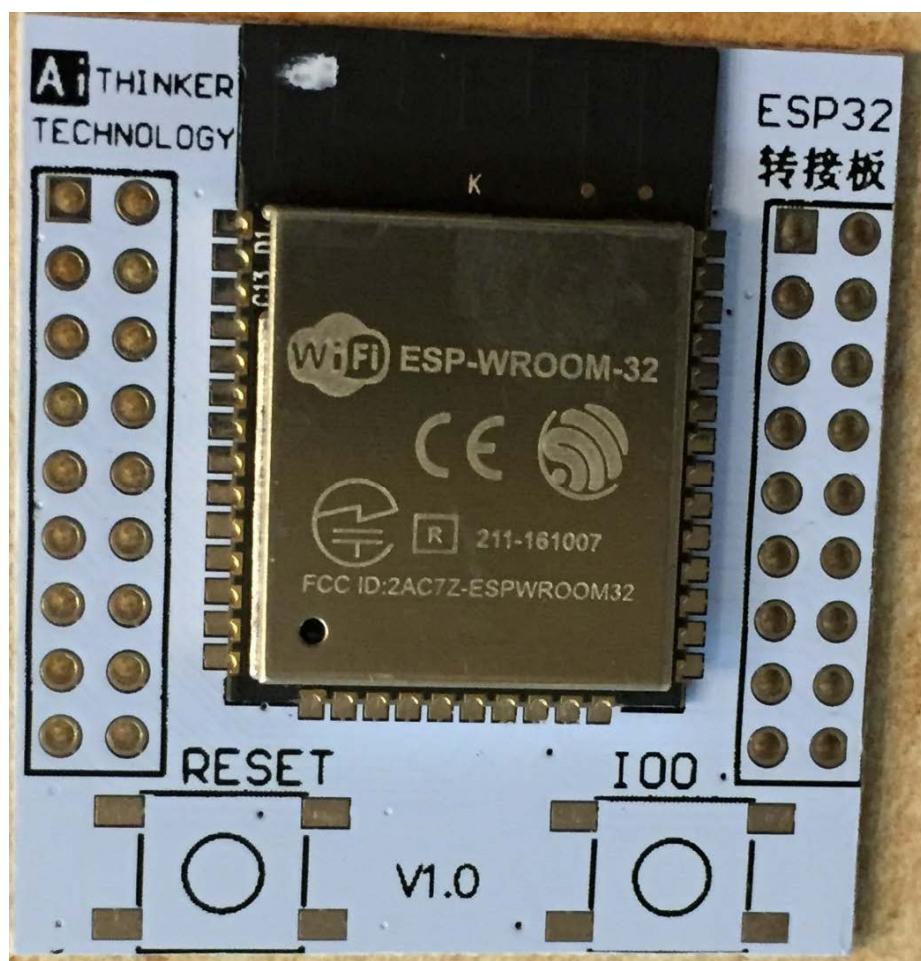
Si es posible desoldar el cable

## Soldar el tablero

Después de que todo esté preparado, sacamos el ESP32 del paquete



y lo colocamos en el tablero para verificarlo:

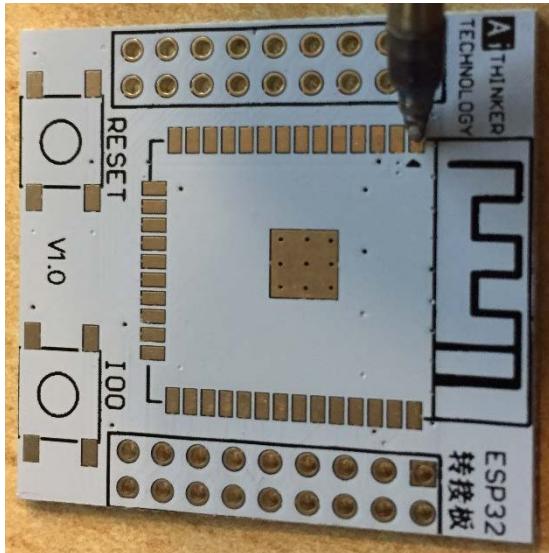


Los contactos deben ser los mismos como se ven en la imagen, encajando con precisión cuando uno se encuentra sobre el otro.

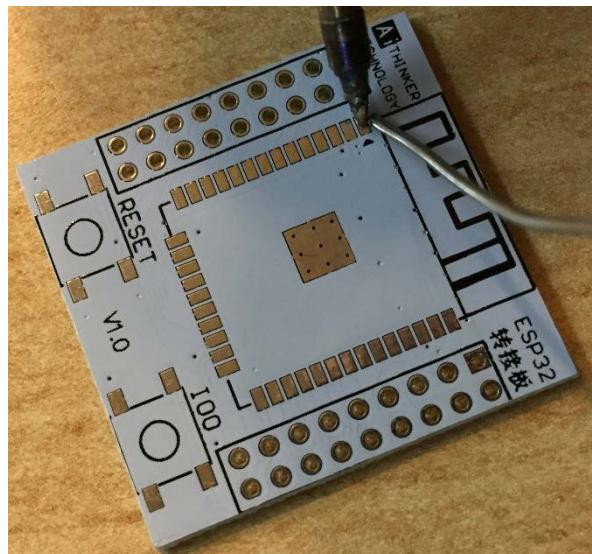
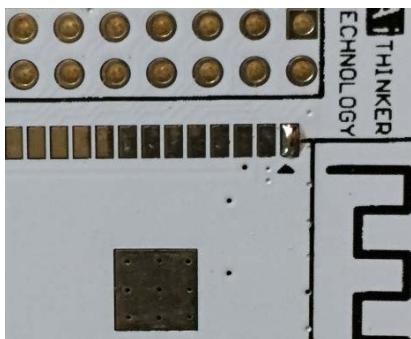
Por lo tanto, tenemos el tablero correcto con el chip correcto.

Ahora tomamos el ESP32 nuevamente y calentamos nuestro soldador. Dependiendo de la soldadura utilizada, la temperatura debe estar entre 300 - 330°C (soldadura que contiene plomo) o 350 - 370°C (soldadura sin plomo). Si el soldador está demasiado caliente, el fundente puede evaporarse demasiado rápido y el componente (chip ESP32) podría sobrecalentarse, lo que provocaría su daño irreparable. Incluso a una temperatura baja y correcta, se aconseja soldar de manera rápida!

Cuando nuestro soldador alcanza la temperatura de trabajo adecuada, estañamos una de las 16 almohadillas de soldadura en el tablero, calentando primero ligeramente la almohadilla y luego añadimos un poco de soldadura.



El resultado debería verse así:

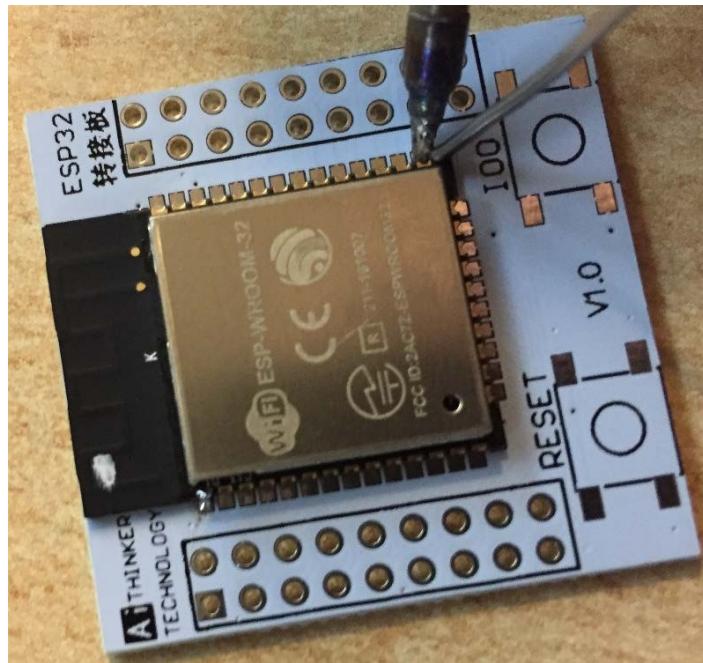


Luego ponemos el chip ESP32 en el tablero y de nuevo calentamos nuestra almohadilla en lata brevemente. Con una ligera presión sobre la parte superior del chip, arreglamos la posición y dejamos que el ESP32 repose sobre el tablero.



Ahora todavía se puede hacer un pequeño ajuste de posición, calentando la soldadura, pero tenga cuidado: no la caliente por mucho tiempo, de lo contrario, pueden producirse daños irreparables por el sobrecalentamiento y hay peligro de formación de puente a través de la soldadura.

Si el chip está conectado a su ubicación prevista, lo solucionamos soldando el contacto opuesto.

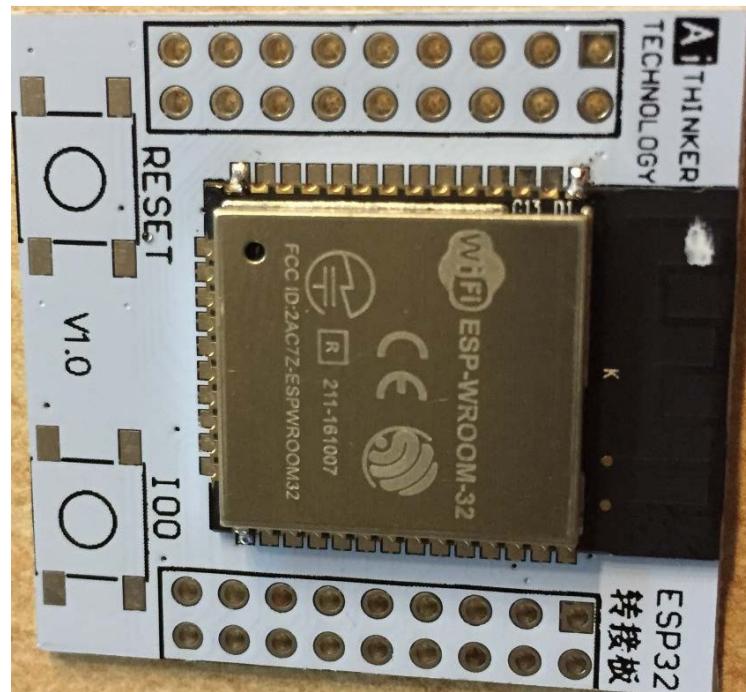


Primero, el contacto y la almohadilla son calentados al mismo tiempo y poco después se suministra algo de soldadura. Tenga cuidado de no suministrar demasiada soldadura. De lo contrario, pueden producirse puentes de soldadura. Si esto sucede, el puente de soldadura se puede eliminar con un cable de desoldadura. Simplemente coloque el cable de desoldadura en el puente de soldadura y caliéntelo con el soldador. El cable desoldador absorbe la soldadura. Repita varias veces si es necesario.

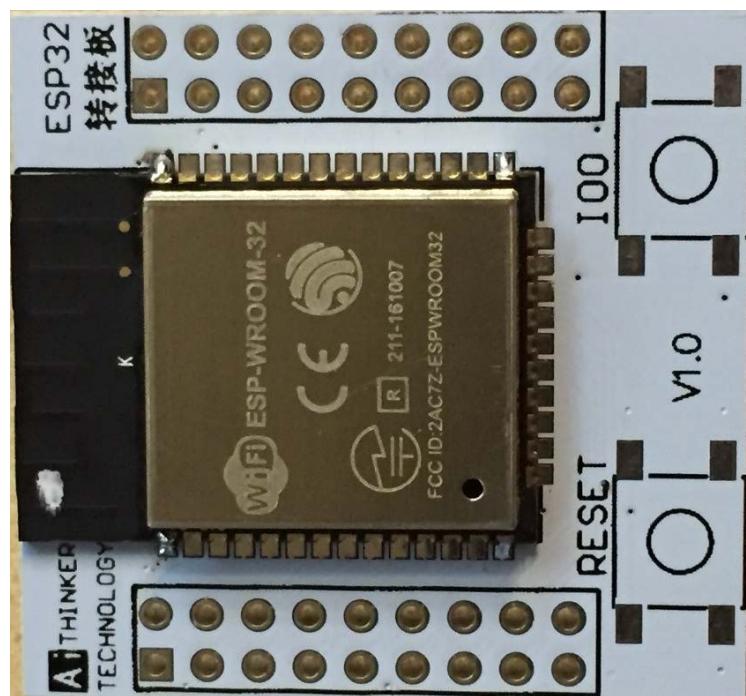
Así es como se ven ahora las 2 juntas soldadas:



Ahora soldamos los otros 2 contactos opuestos firmemente:

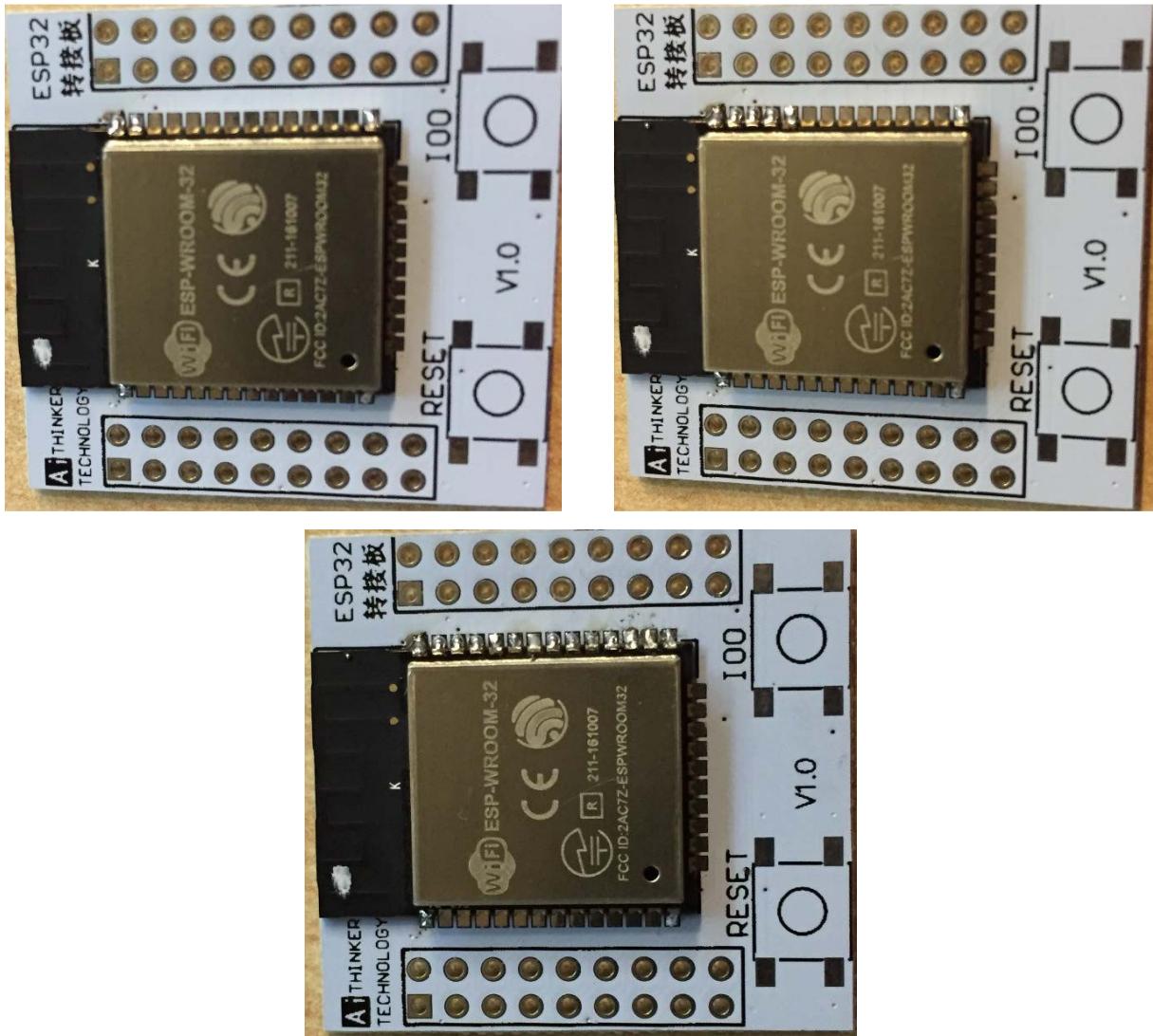


En primer lugar, debe calentar el contacto y luego agregar soldadura.



Los cuatro contactos ahora han sido soldados, lo que significa que el chip ESP32 ahora está fijo y ya no se puede deslizar.

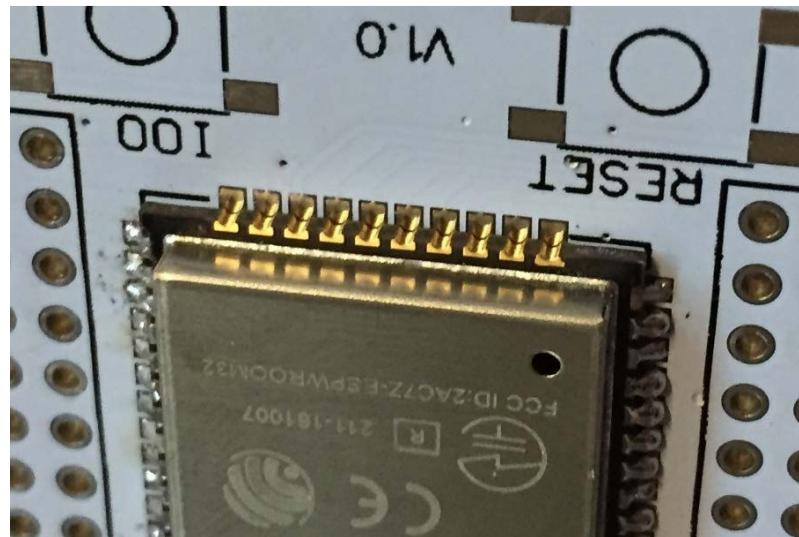
Ahora, un contacto después del otro puede ser soldado en un lado:



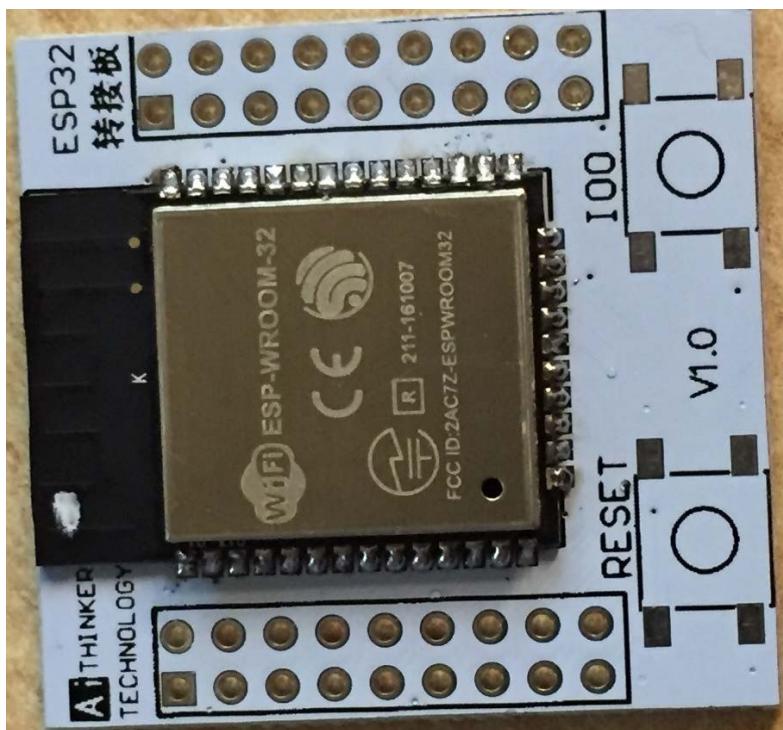
Si un lado está terminado, continuamos con el otro lado:



Finalmente, en el ESP32, el 3er lado está soldado contacto por contacto:

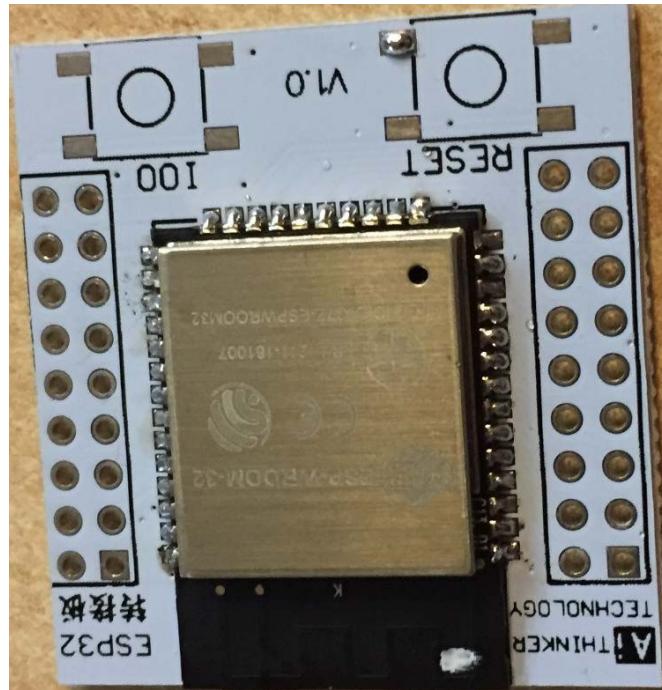


Ahora todos los contactos están soldados:

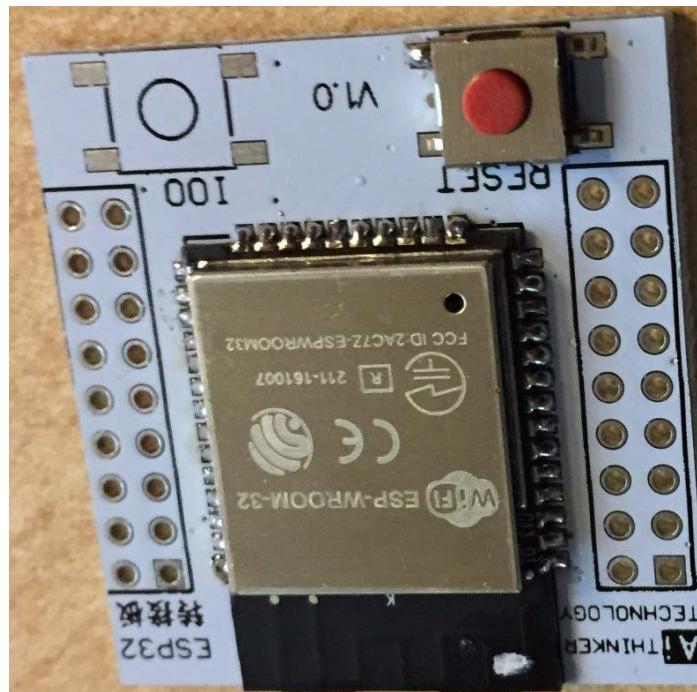


Si todos los contactos del chip ESP32 están soldados, soldamos los 2 botones SMD en el tablero.

Nuevamente, aquí también estañamos una de las almohadillas en el tablero:



Por último, configuramos el botón en el punto enlatado y lo arreglamos.

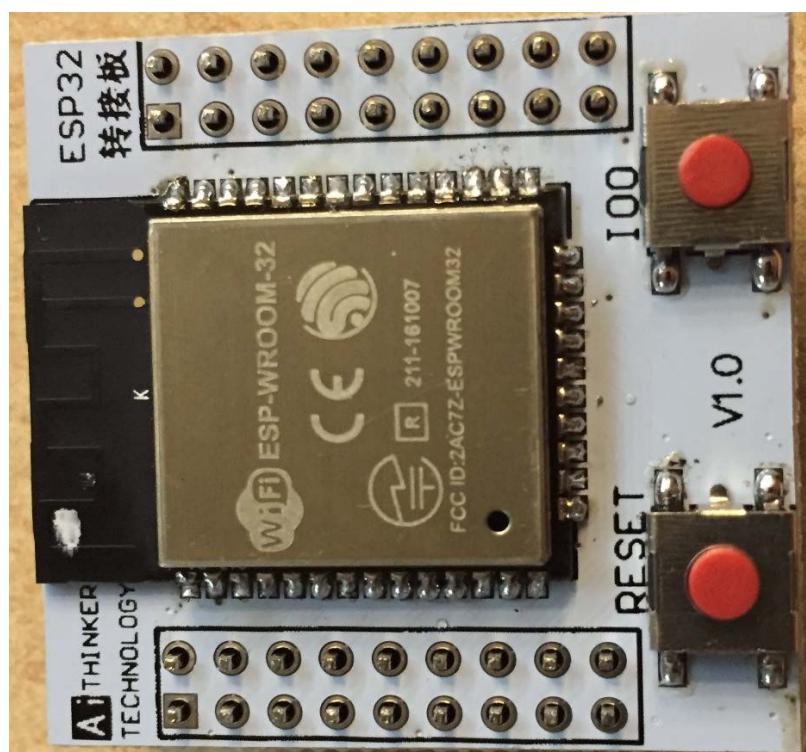


Ahora tiene la posibilidad nuevamente de hacer un pequeño ajuste de posición, y nuevamente es recomendable no calentarlo por mucho tiempo. De lo contrario, el botón mecánico también se puede dañar.

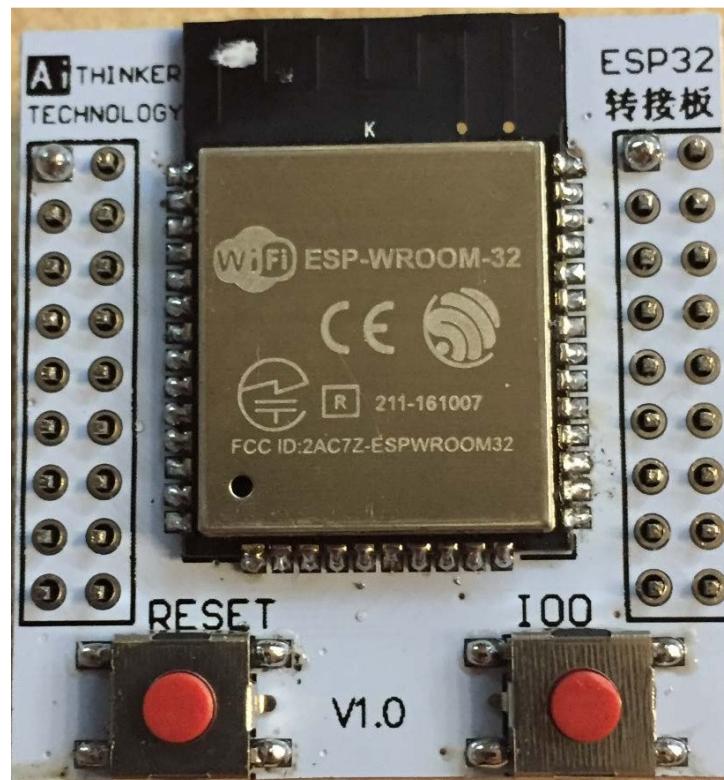
Después, los 4 contactos del botón deben ser soldados. Siga los mismos pasos para el 2do botón.



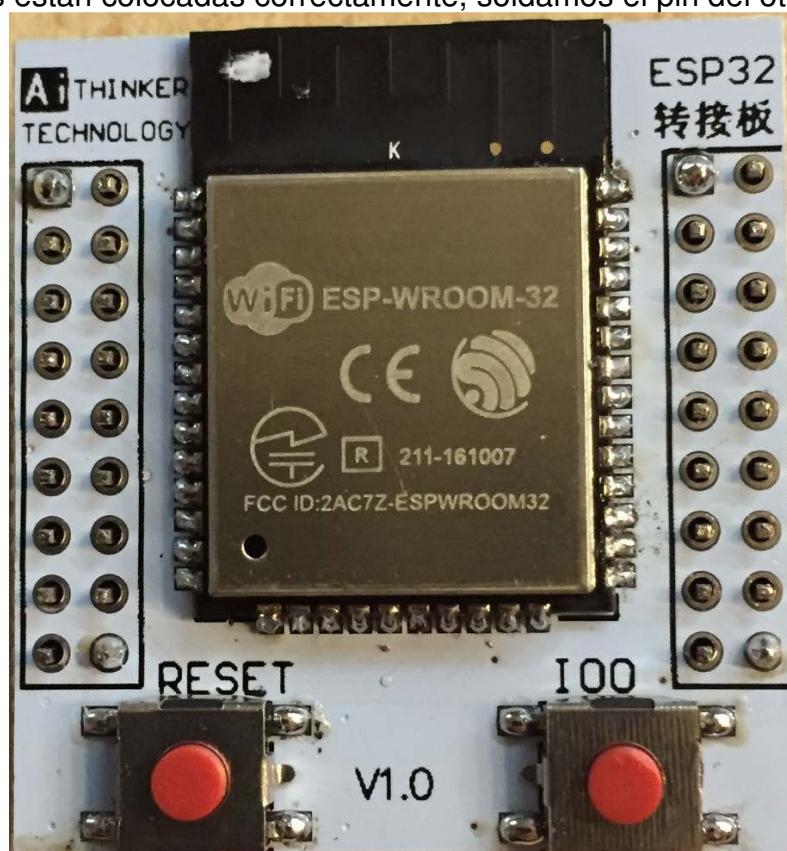
Ahora que hemos soldado todos los contactos SMD, podemos soldar los cabezales de los pines en ambos lados. Para hacer esto, tomamos los cabezales de los pines y los insertamos en el tablero:



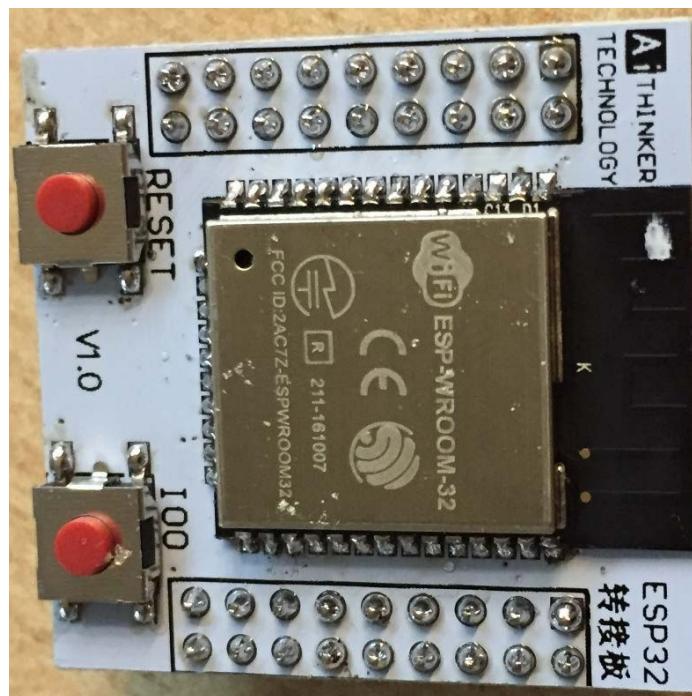
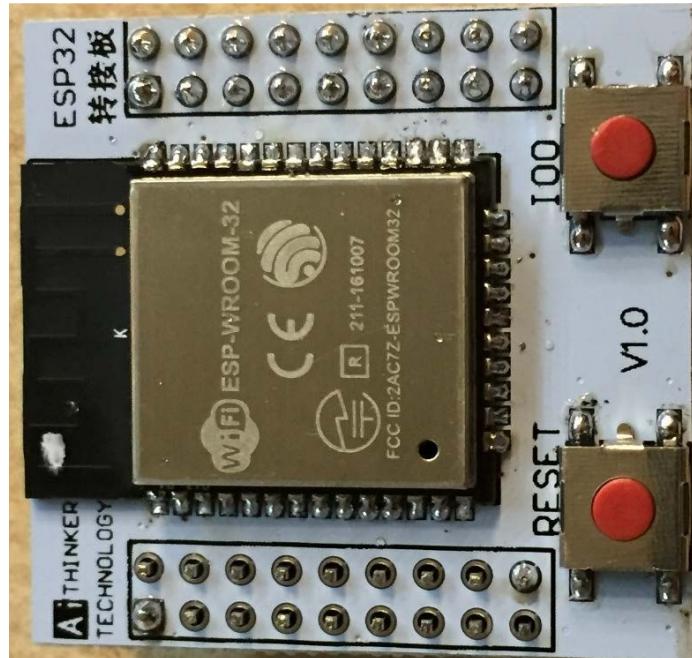
Aquí también soldamos solo un pin en cada lado:



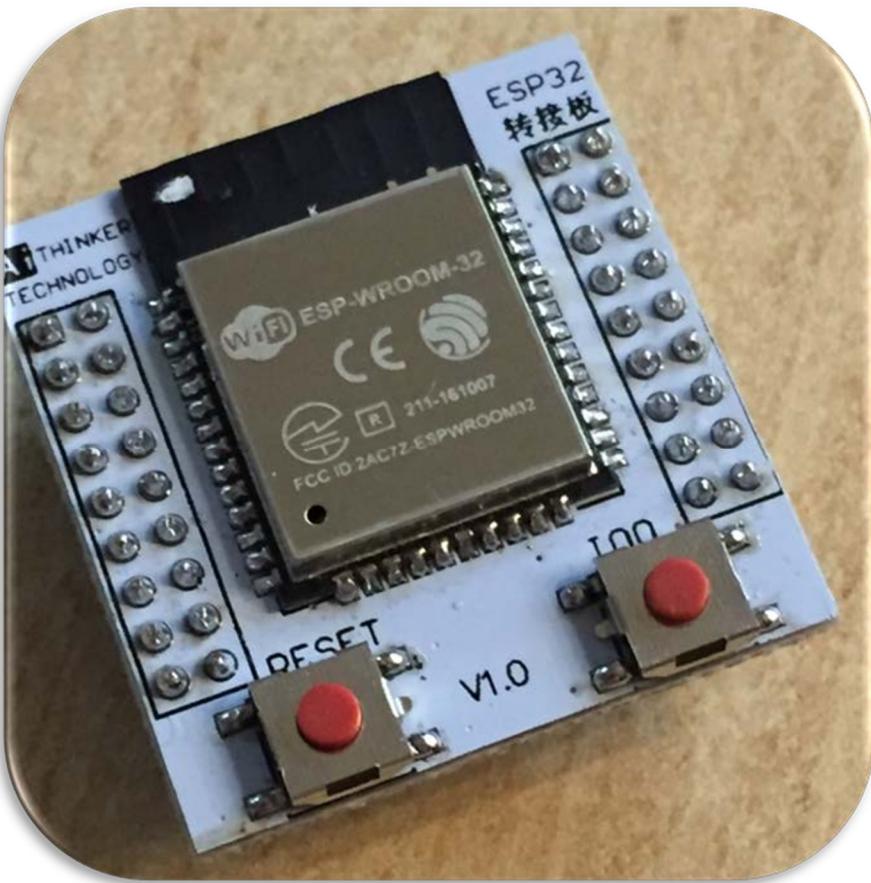
Si es necesario, todavía se puede ajustar y alinear el encabezado del pin.  
Si ambas tiras están colocadas correctamente, soldamos el pin del otro lado:



Soldamos los pines en cada lado:



Ahora hemos terminado con la soldadura:



**¡Lo ha logrado! Ahora es el momento de programar y para saber cómo funciona, puede obtener más información en las siguientes páginas.**

## Programación del ESP32

El ESP32 tiene un mejor rendimiento en comparación con su hermano pequeño (ESP8266) y, además del WLAN contiene Bluetooth en la placa.

### Preparación del Software:

El Arduino Software que vemos en este paso como Installed, si todavía falta que usted lo instale, puede descargarlo en <https://www.arduino.cc/en/Main/Software#> e instalarlo en su PC.

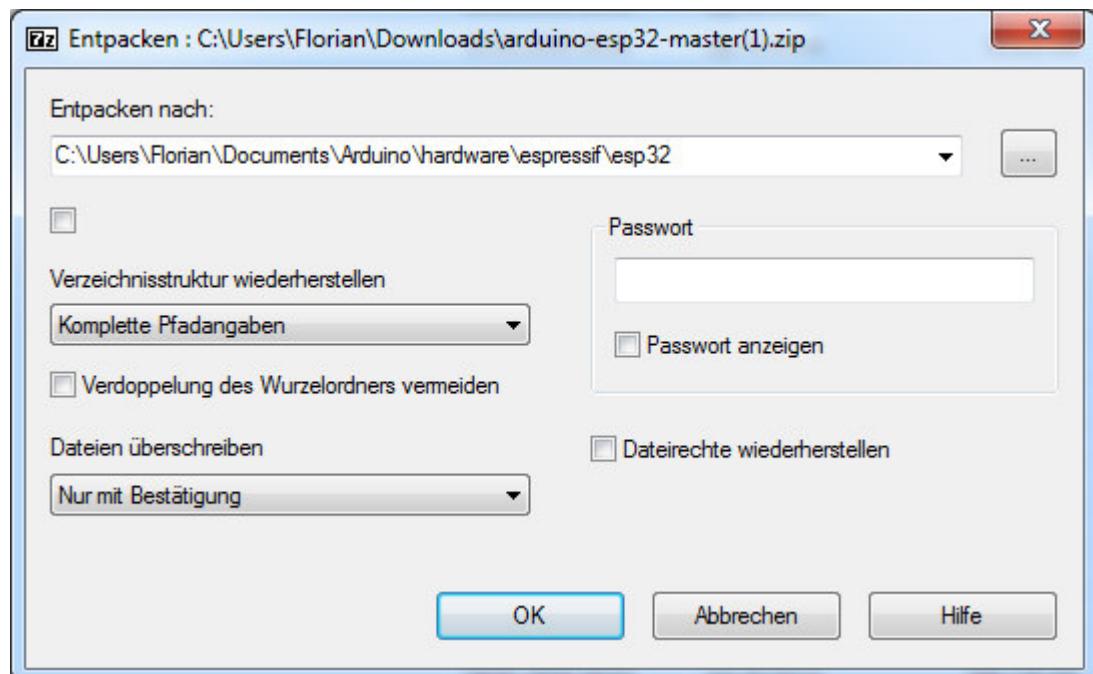
Además, los controladores para el FT232RL ya deberían haber sido instalados, de lo contrario, puede ver el libro electrónico correspondiente.

Una vez que se han cumplido todos los requisitos básicos, todavía tenemos que descargar manualmente los paquetes necesarios para el ESP32 e integrarlos en el Arduino Software. Aquí se pueden descargar de GIT los datos actuales:

<https://github.com/espressif/arduino-esp32/archive/master.zip>

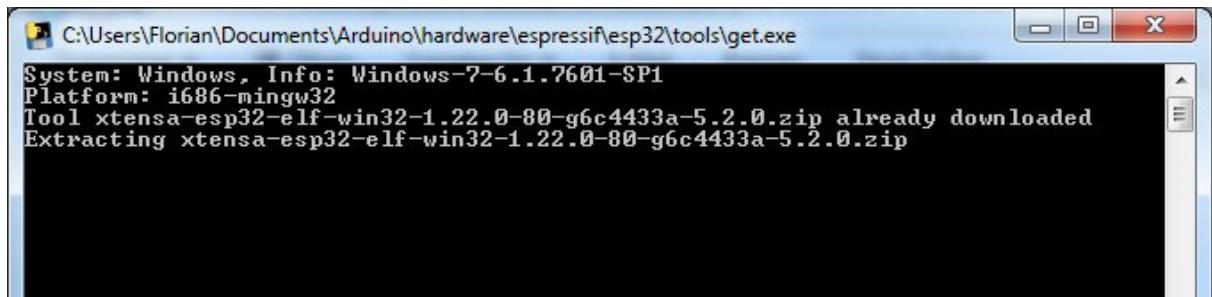
Descomprima este archivo comprimido (con 7zip) en la carpeta: [Mi directorio de usuario (C:\ Usuario \ Florian \] \ Mis documentos \ Arduino \ hardware \ espressif \ esp32

Nota: si estas carpetas no existen, simplemente créelas nuevamente.



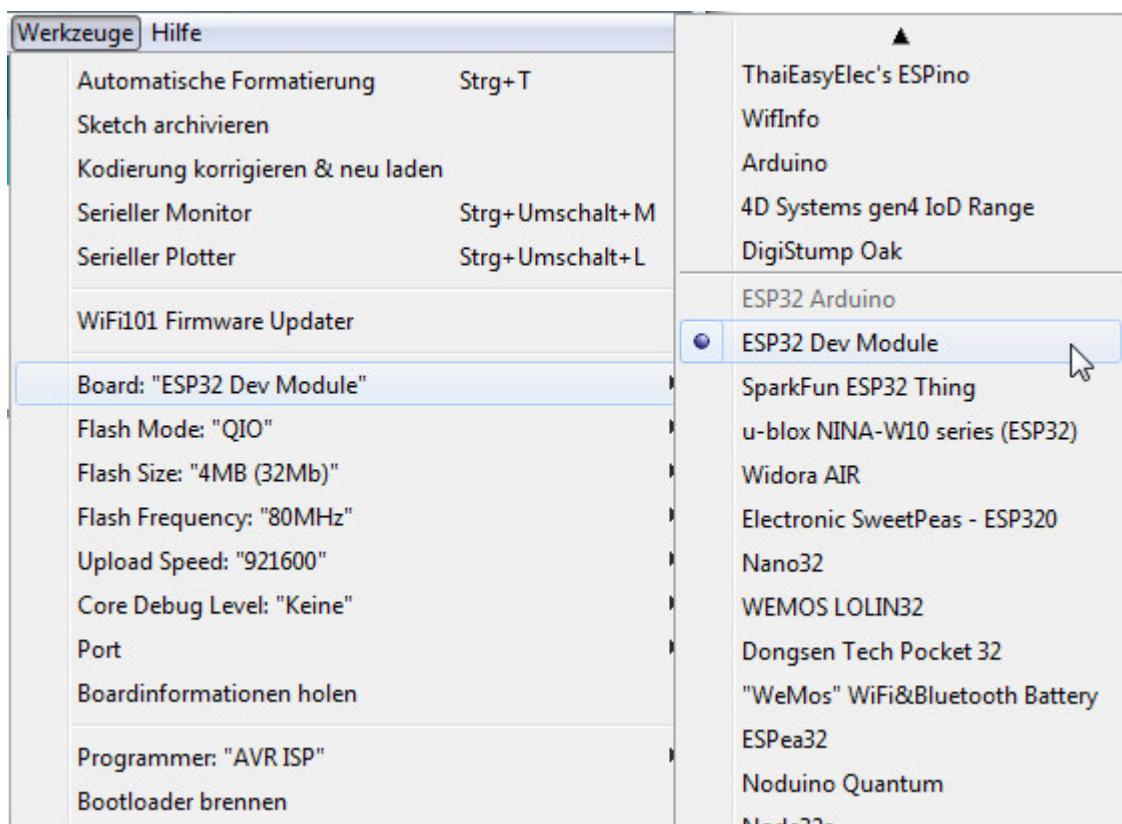
Name	Änderungsdatum	Typ	Größe
arduino-esp32-master	23.01.2018 12:08	Dateiordner	
cores	23.01.2018 12:08	Dateiordner	
docs	23.01.2018 12:08	Dateiordner	
libraries	23.01.2018 12:08	Dateiordner	
package	23.01.2018 12:08	Dateiordner	
tools	28.01.2018 17:38	Dateiordner	
variants	23.01.2018 12:08	Dateiordner	
.gitignore	23.01.2018 12:08	GITIGNORE-Datei	1 KB
.gitmodules	23.01.2018 12:08	GITMODULES-Datei	1 KB
.travis.yml	23.01.2018 12:08	YML-Datei	3 KB
appveyor.yml	23.01.2018 12:08	YML-Datei	1 KB
boards.txt	23.01.2018 12:08	TXT-Datei	51 KB
component.mk	23.01.2018 12:08	MK-Datei	1 KB
Kconfig	23.01.2018 12:08	Datei	3 KB
Makefile.projbuild	23.01.2018 12:08	PROJBUILD-Datei	1 KB
package.json	23.01.2018 12:08	JSON-Datei	1 KB
platform.txt	23.01.2018 12:08	TXT-Datei	9 KB
programmers.txt	23.01.2018 12:08	TXT-Datei	0 KB
README.md	23.01.2018 12:08	MD-Datei	3 KB

En la carpeta tools, hay un "get.exe". Tenemos que ejecutar esto una vez y tener todos los paquetes de software requeridos instalados y descargados.



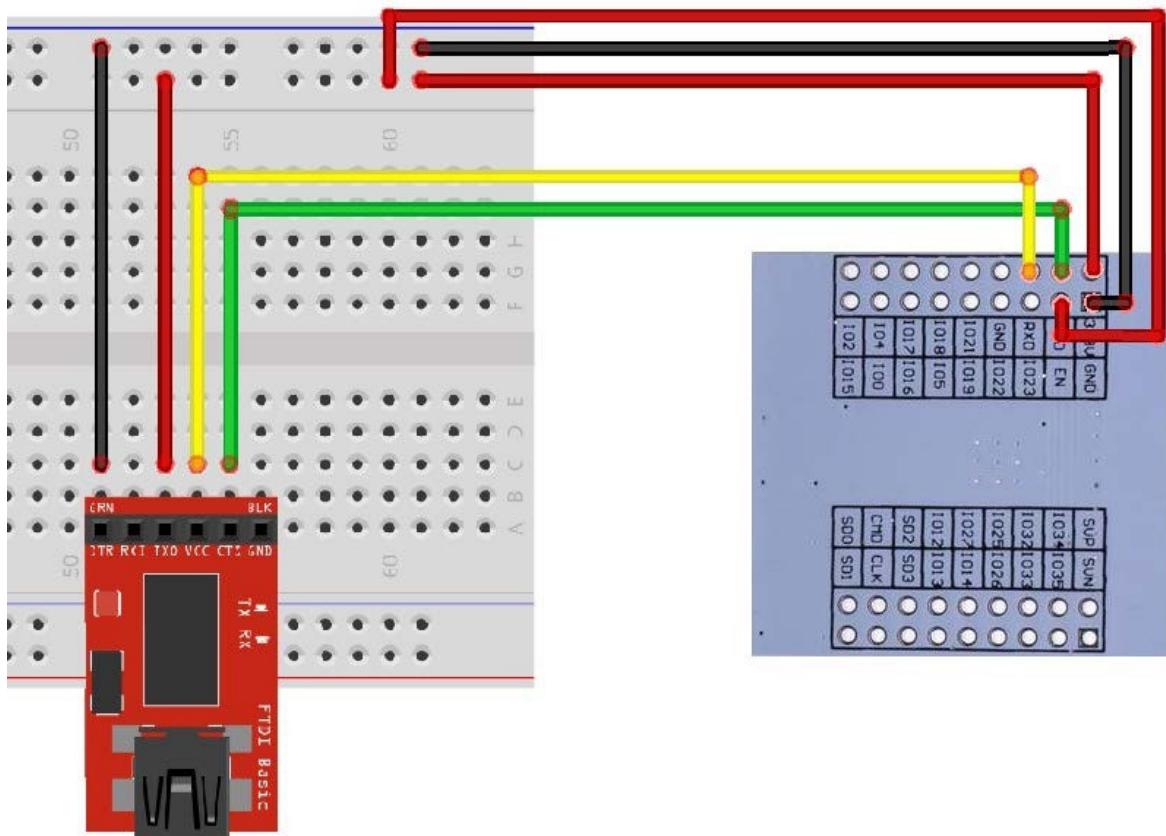
Esto toma un momento. Cuando todo se ha completado, la ventana negra se cierra sola.

Luego, iniciamos el Arduino Software y vamos a Herramientas>Board y seleccionamos el ESP32 Dev Module.



En el puerto, debe ingresar solo el Com-Port de su Serial Adapter, que puede leerlo en el administrador del dispositivo y, si es necesario, también cambiarlo.

### **Cableado del módulo con el Serial Adapter:**



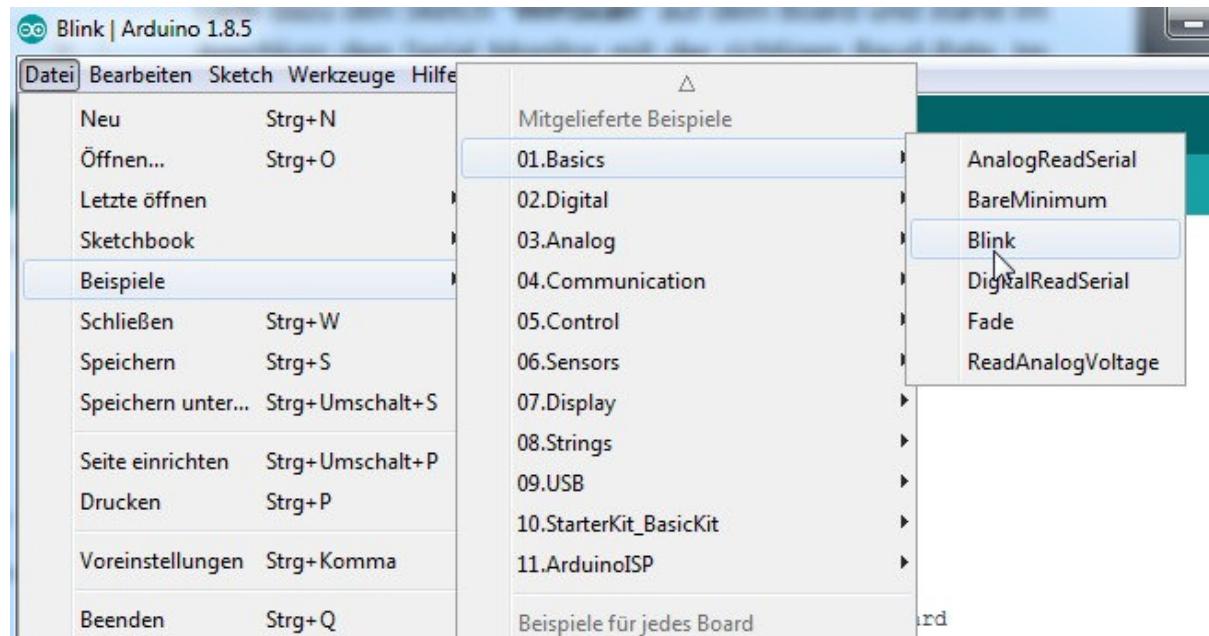
Desafortunadamente, la placa del adaptador no es adecuada para tableros de laboratorio, por lo que debemos conectarnos al Serialadapter con los cables de puente apropiados.

- |  |                |
|--|----------------|
| Conecitar VCC a una fuente de alimentación de 3,3V | cable rojo     |
| Conecitar GND a tierra                             | cable negro    |
| Conecitar EN (habilitación de chip) a 3,3V (High)  | cable rojo     |
| TXD a RX Serial Adapter                            | cable amarillo |
| RXD a TX Serial Adapter                            | cable verde    |

## El Arduino Code

Ahora que el cableado ya está hecho, escribimos nuestro primer código. Dejemos que un LED parpadee.

Para hacer esto, vaya a Archivo > Ejemplos > 01.Basics > Blink.



Necesitamos cambiar el siguiente código:

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(21, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(21, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                // wait for a second
    digitalWrite(21, LOW);       // turn the LED off by making the voltage LOW
    delay(1000);                // wait for a second
}
```

La versión típica de Arduino "LED\_BUILTIN" no funciona.

El ESP32 no tiene un LED incorporado.  
Sin embargo, si nos gustaría tener un LED parpadeando,  
tenemos que reemplazar LED\_BUILTIN con 21.  
El IO21 es un pin libremente programable del ESP32.

Pero también puede conectar un LED a cualquier otro IO

```
(IO15 => digitalWrite(15, HIGH);)
```

Después de que hayamos cambiado el código, hacemos clic en:  y verificamos nuestro programa:

Si todo es correcto y nuestro programa no contiene errores

```
Der Sketch verwendet 158969 Bytes (12%) des Programmspeicherplatzes. Das Maximum sind 1310720 Bytes.  
Globale Variablen verwenden 10968 Bytes (3%) des dynamischen Speichers, 283944 Bytes für lokale Variablen verbleiben. Das Maximum sind 294912 Bytes.
```

podemos subirlo al ESP32. Para esto hacemos clic en: 

Después de un corto tiempo, va a aparecer Connecting:

```
esptool.py v2.1  
Connecting.....
```

Luego presionamos ambos botones en la placa y soltamos Reset, pero IO0 permanece presionado hasta que aparezca Hard Reset; luego soltamos el IO0 y solo se presiona brevemente el Reset. El LED, que estaba conectado entre el pin y la tierra (¡no olvide la resistencia en serie para voltaje de 3.3V!) comienza a parpadear.

```
esptool.py v2.1  
Connecting...  
Chip is ESP32D0WDQ6 (revision 1)  
Uploading stub...  
Running stub...  
Stub running...  
Changing baud rate to 921600  
Changed.  
Configuring flash size...  
Auto-detected Flash size: 4MB  
Compressed 8192 bytes to 47...  
  
Writing at 0x0000e000... (100 %)  
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 4369.0 kbit/s)...  
Hash of data verified.  
Compressed 14128 bytes to 9196...  
  
Writing at 0x00001000... (100 %)  
Wrote 14128 bytes (9196 compressed) at 0x00001000 in 0.1 seconds (effective 1056.3 kbit/s)...  
Hash of data verified.  
Compressed 160112 bytes to 81979...  
  
Writing at 0x00010000... (16 %)  
Writing at 0x00014000... (33 %)  
Writing at 0x00018000... (50 %)  
Writing at 0x0001c000... (66 %)  
Writing at 0x00020000... (83 %)  
Writing at 0x00024000... (100 %)  
Wrote 160112 bytes (81979 compressed) at 0x00010000 in 1.5 seconds (effective 855.6 kbit/s)...  
Hash of data verified.  
Compressed 3072 bytes to 122...  
  
Writing at 0x00008000... (100 %)  
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (effective 1638.4 kbit/s)...  
Hash of data verified.  
  
Leaving...  
Hard resetting...
```

**Lo hizo! Ahora puede programar sus propios proyectos.**

A partir de ahora es tiempo de realizar sus propios proyectos.

Para más hardware, nuestra tienda en línea siempre está a su disposición:

<https://az-delivery.de>

Disfrute!

**Impressum**

<https://az-delivery.de/pages/about-us>