Welcome!

Thank you for purchasing our *AZ-Delivery ESP32 Module with adapter plate*. On the following pages, you will be intorduced to how to use and set-up this handy device.

Have fun!

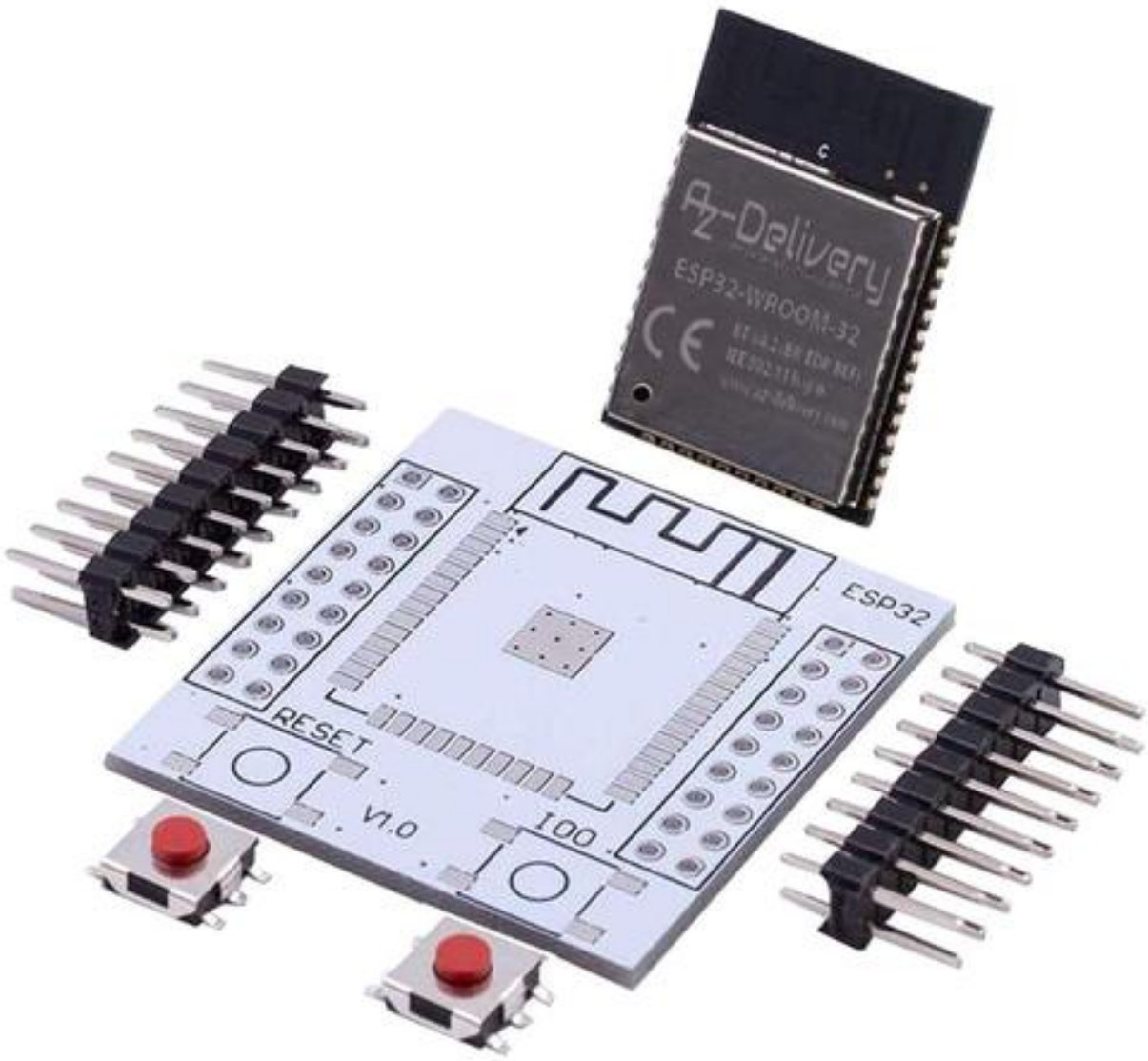


Table of Contents

Introduction

The ESP32 is a System on chip (SoC) designed by Espressifs systems that can be used to add both Wi-Fi and Bluetooth functionality to projects. It is ideal for the development of mobile devices, IoT applications and wearable electronics due to its wireless capabilities and smart power management among other features. Since the ESP32 SoC comes in the form of a surface mount chip or module, it is not easy to use for prototyping in its barebones state. That is why it is advisable to use an ESP32 adapter plate (breakout board) when developing projects.

To access all the features on the ESP32, one must have access to all its pins. The ESP32 adapter board make this possible by breaking out all needed pins that developers can use to easily interface the ESP32 with other devices and components. The adapter plate provides all pins assignment that makes prototyping easy.

Specifications

- 2.54mm (0.1in) pitch
- Easy to solder
- PCB thickness: 1.6mm
- Reset push button switch
- GPIO0 push button switch (enters programming mode)
- Dimensions: 36x33x15mm (1.4x1.2x0.6in)

ESP32 Features

- The smallest 802.11b/g/n WiFi BT SoC Module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed
- Overview built-in 520KB SRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- SMD-38 package for easy welding
- Support for OpenOCD debugging interface
- Deep sleep current as low as 6.5μA
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- General AT commands can be used quickly

For a detailed specifications and features of ESP32 module chip, download the datasheet on the following link.

How to solder the ESP32 to the adapter board

The package consists of:

1x ESP32 chip module

1x Adapter Board

2x Pin headers (2x9 pins)

2x Push button switch

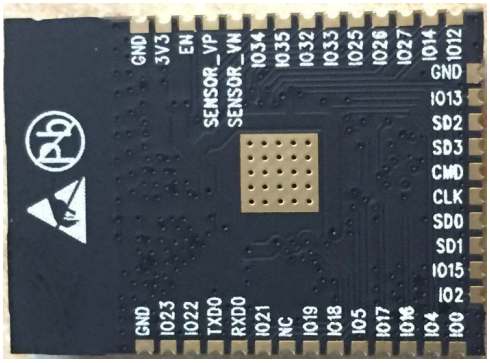
NOTE: The ESP32 chip module is an SMD component and is not suitable for soldering beginners and that one should have SMD soldering experience.

For soldering the ESP32 to the board following tools are needed:

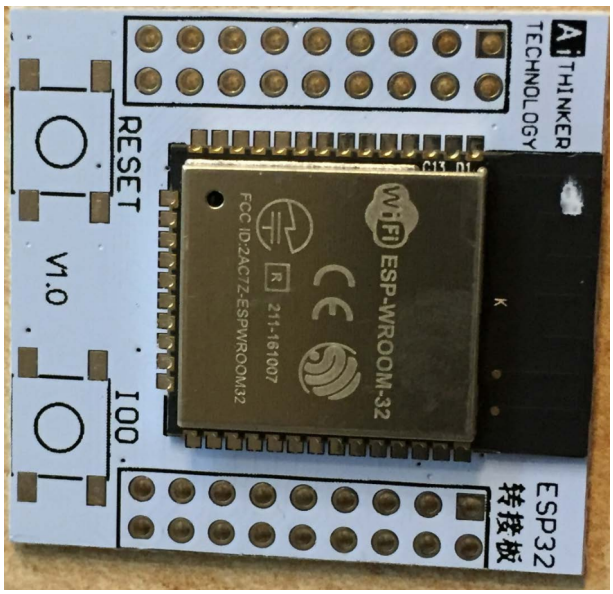
1. Soldering iron with fine tip
2. Soldering tin for electronics (flux inside)
3. Desoldering braid (in case it is needed)

Soldering the board

After everything has been prepared, take the ESP32 out of the packaging



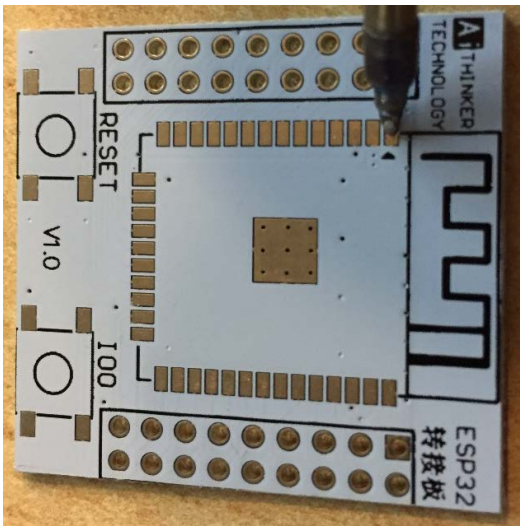
Place it on the circuit board for checking:

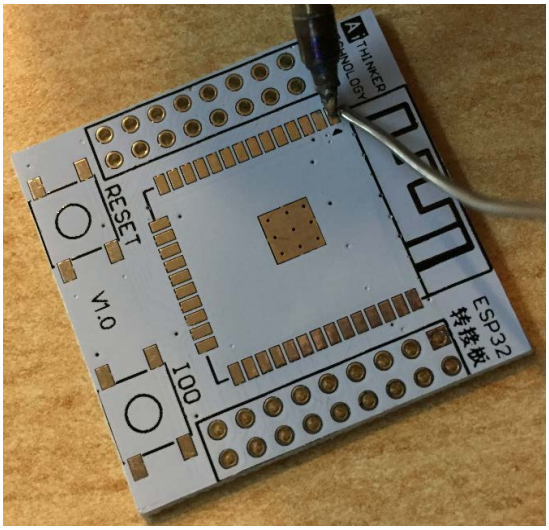


As can be seen in the picture, the contacts should fit perfectly on top of each other.

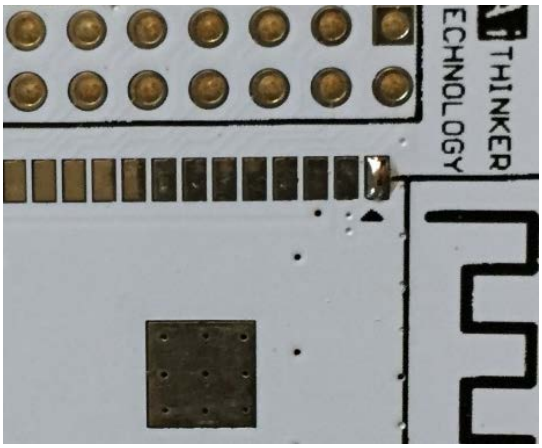
Now take the ESP32 down again and heat up the soldering iron. Depending on the solder used, the temperature should be between 300-330°C (lead-containing solder) or 350-370°C (lead-free solder). If the soldering iron is too hot, the flux can evaporate too quickly and the component (ESP32 chip) can become too hot and get damaged from heat. Therefore, even at low and correct temperatures, it is important to solder quickly.

When soldering iron reaches the working temperature, tin one of the 16 soldering pads on the board by first heating the pad a little and then adding some solder.

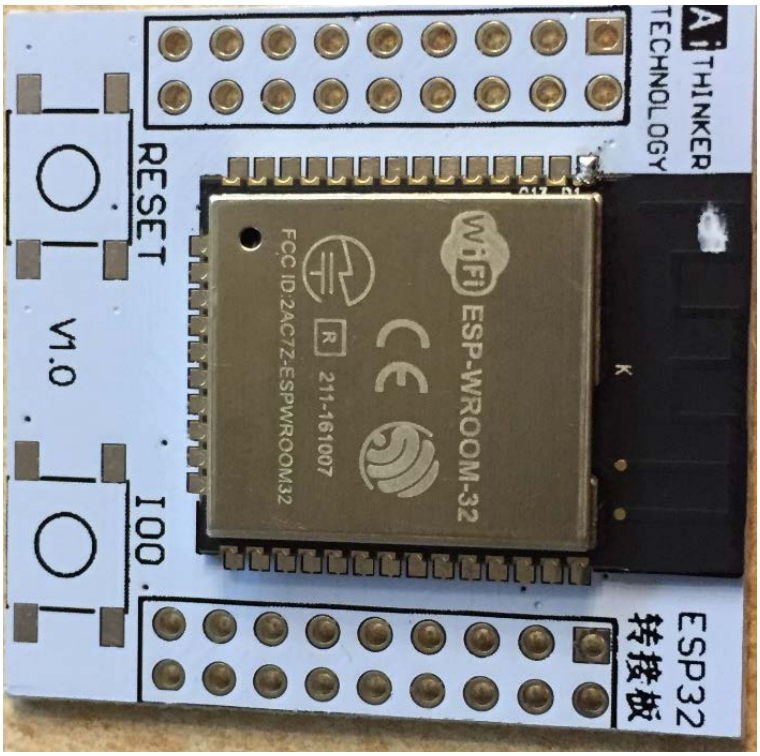




The result should look like the one on the following image:

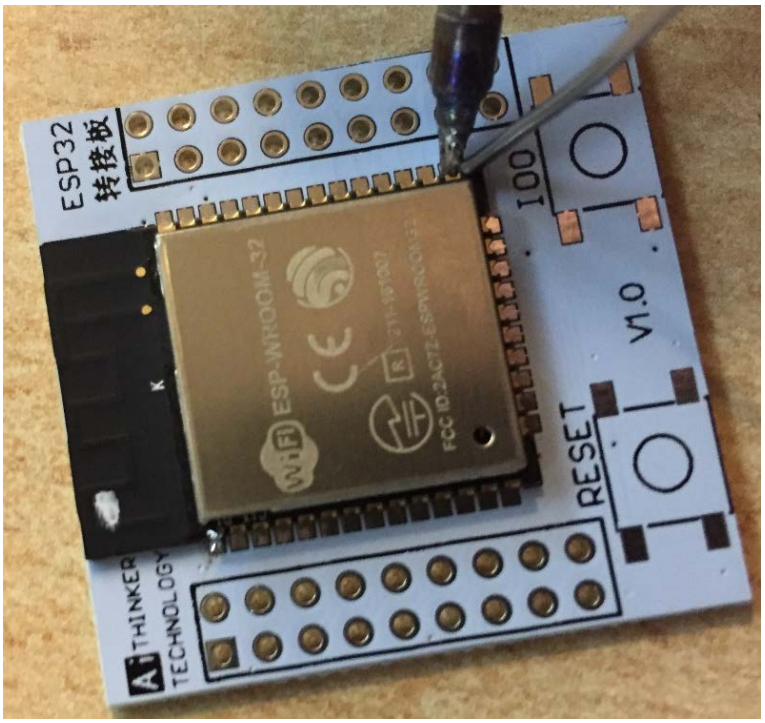


Then put the ESP32 chip on the circuit board and briefly heat tinned pad again. With light pressure from above on the chip, fix the position and let the ESP32 rest on the board.



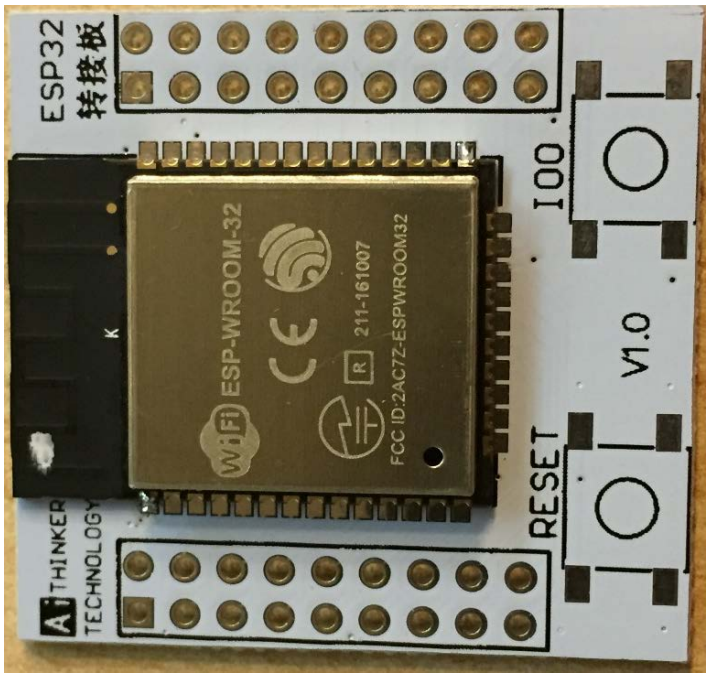
Now a small change in position can be made by heating the solder joint carefully, but do not heat too long to avoid risk of bridging other pins or damage the chip with overheating.

If the chip is attached to its intended place, fix it by soldering the opposite contact.

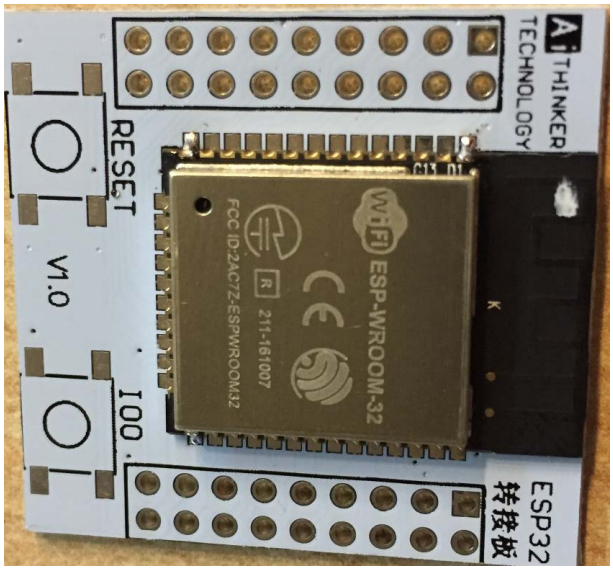


First the contact and the pad are heated at the same time and a little solder is added shortly afterwards. Please be careful not to add too much solder. Solder bridges can then arise. If this happens, the solder bridge can be removed again with a desoldering braid. Simply place the desoldering braid on the solder bridge and heat it with the soldering iron. The desoldering wire soaks up the solder. Repeat several times if necessary.

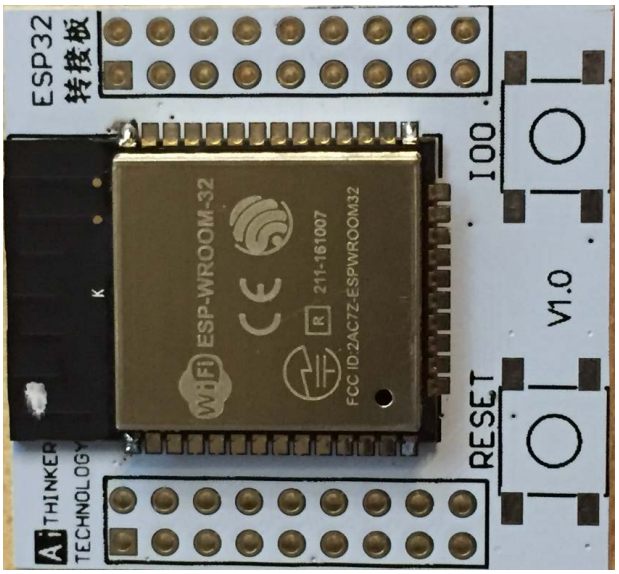
The solder joints should look as on the following image:



Now, solder the two other opposite contacts:

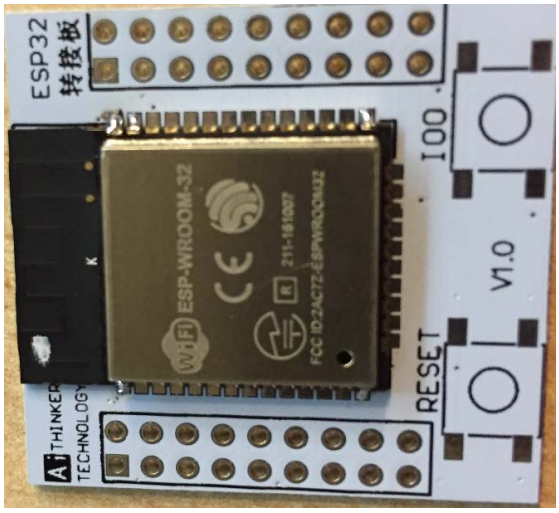


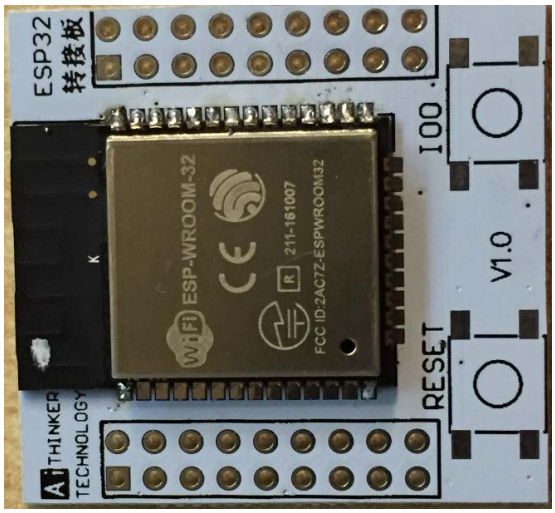
First heat the contact and then apply solder.



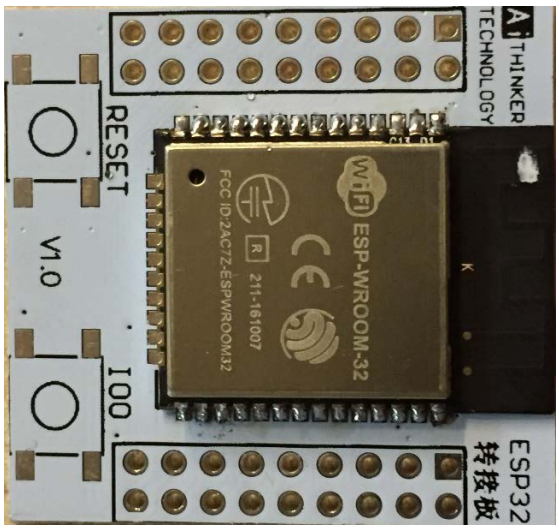
Four contacts are now soldered, the ESP32 chip is now firmly placed and can no longer slip.

Now, solder the contacts each contact on one side:

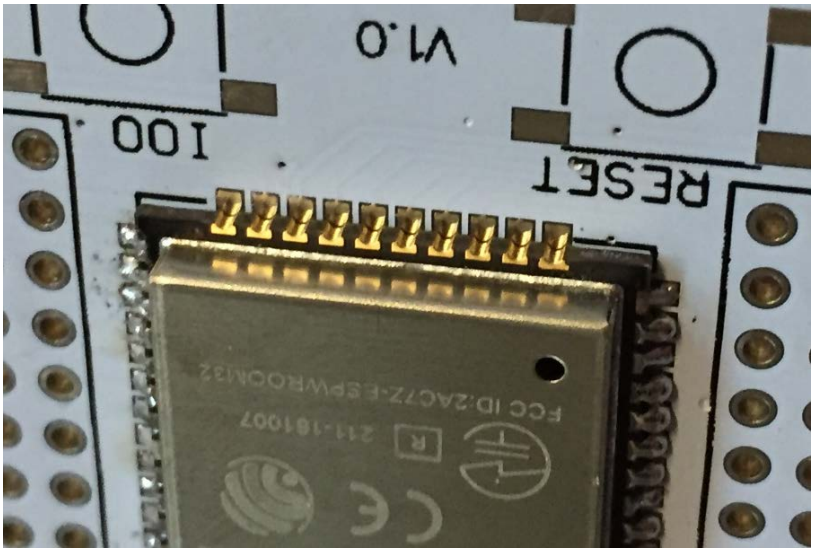




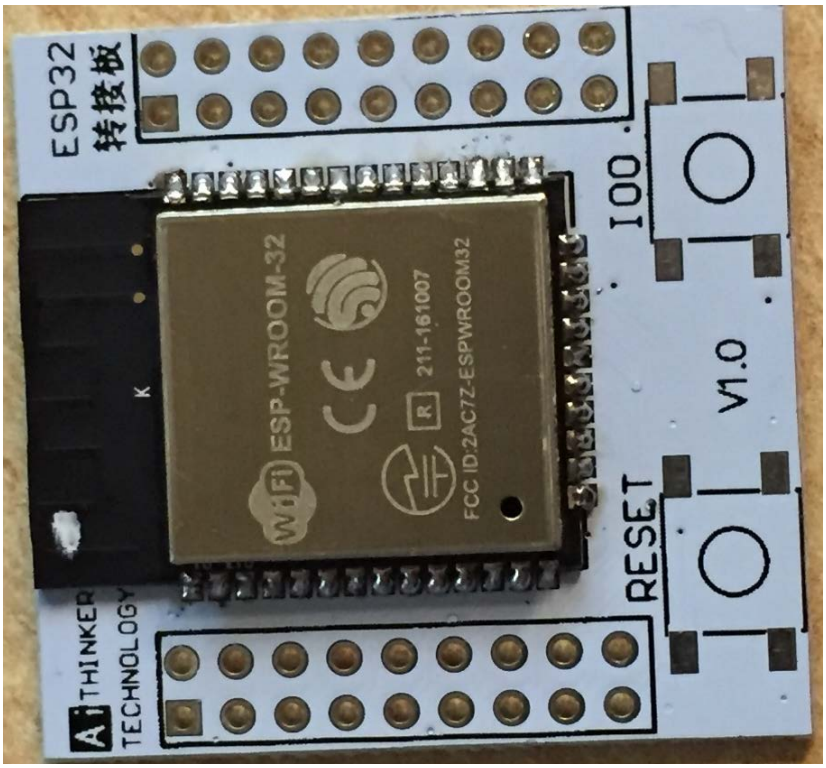
When finished, continue with the other side:



Finally, the third side, solder each contact one by one:

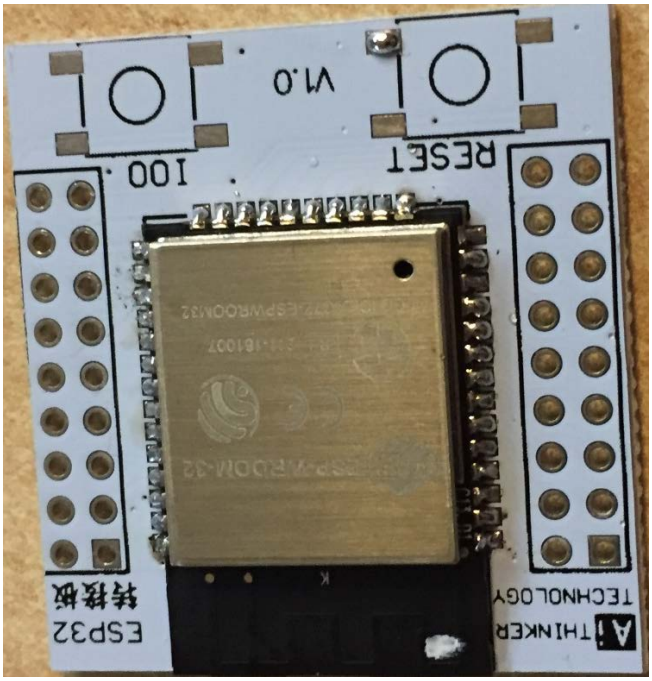


Now, soldering the ESP32 is finished:

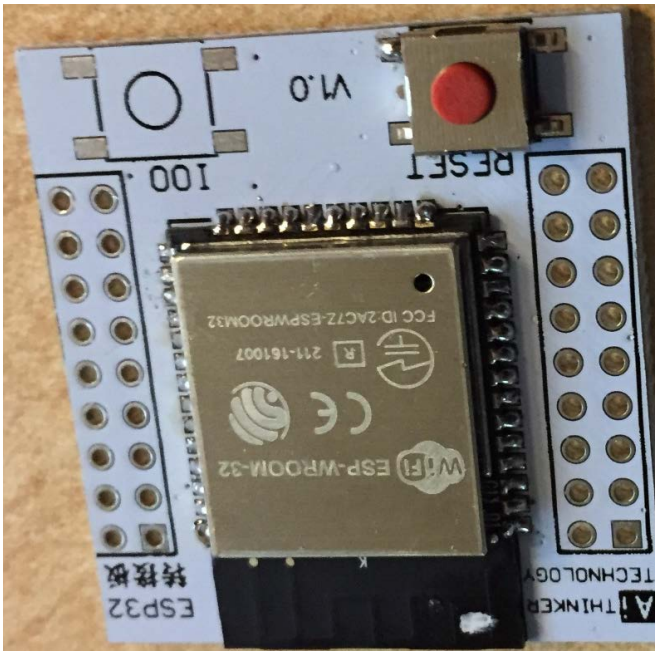


When all contacts of the ESP32 chip are soldered, solder the two SMD buttons to the board.

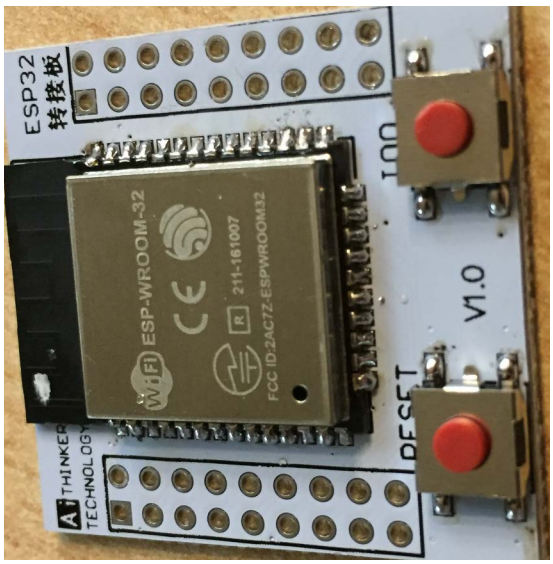
First, tin-coat one of the pads on the board:



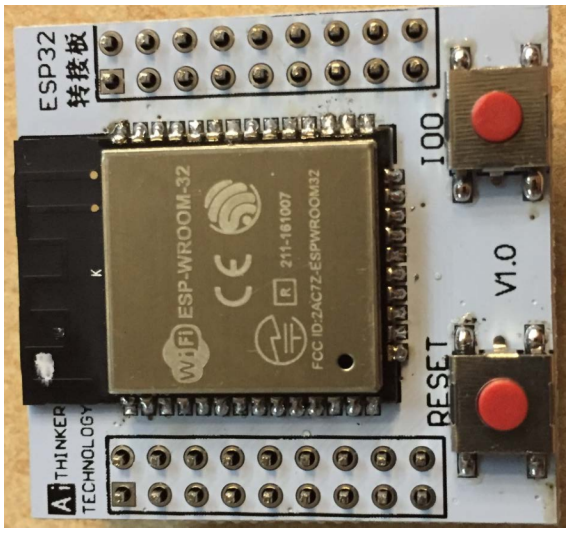
Then put the button on the tinned point and fix it.



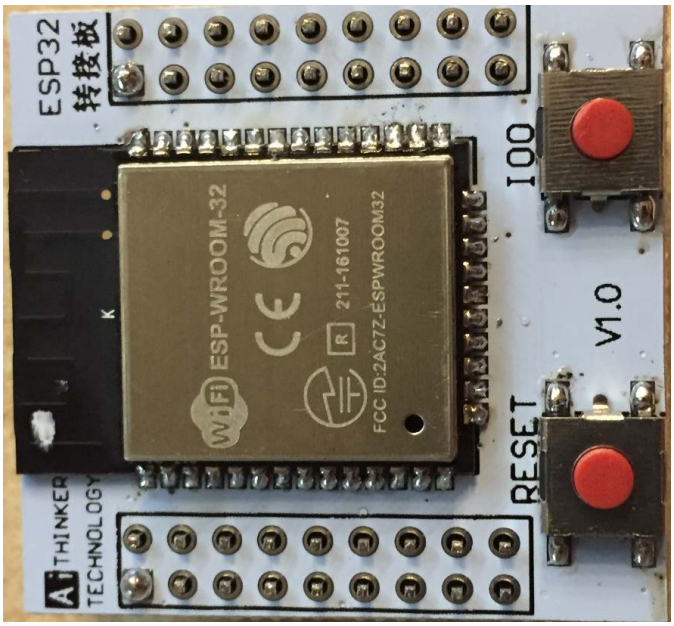
Now a small change in position can be carried out again and again, do not heat too long. The mechanical button can also be damaged. Then all 4 contacts of the button are soldered and all steps are repeated with the 2nd button.



Now when all SMD contacts are soldered, the pin headers can be soldered too on both sides. To do this, take the pin headers and insert them into the circuit board:

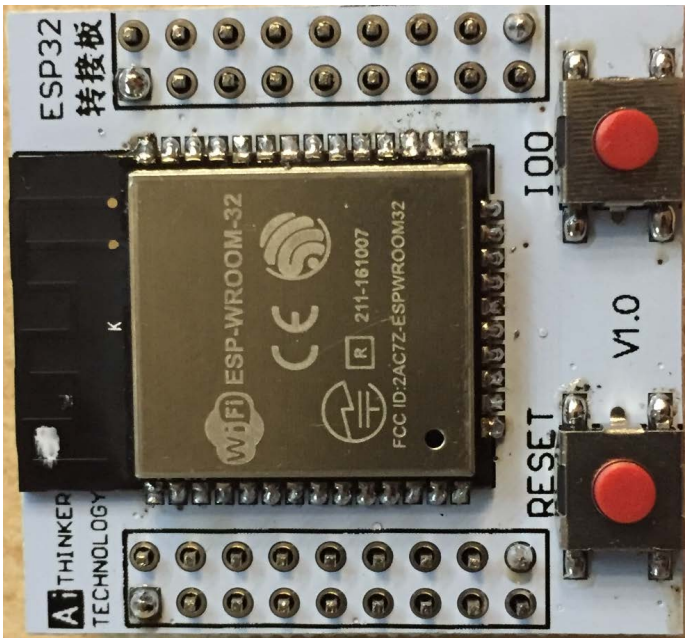


As before, solder one pin on each side:

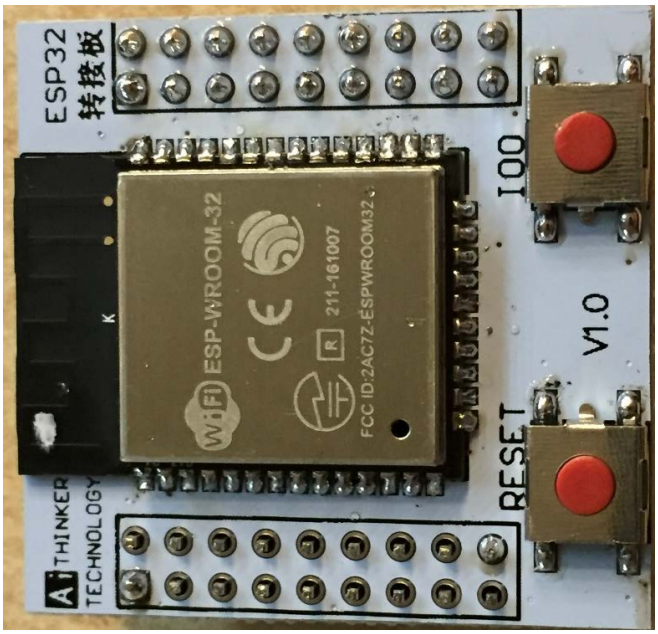


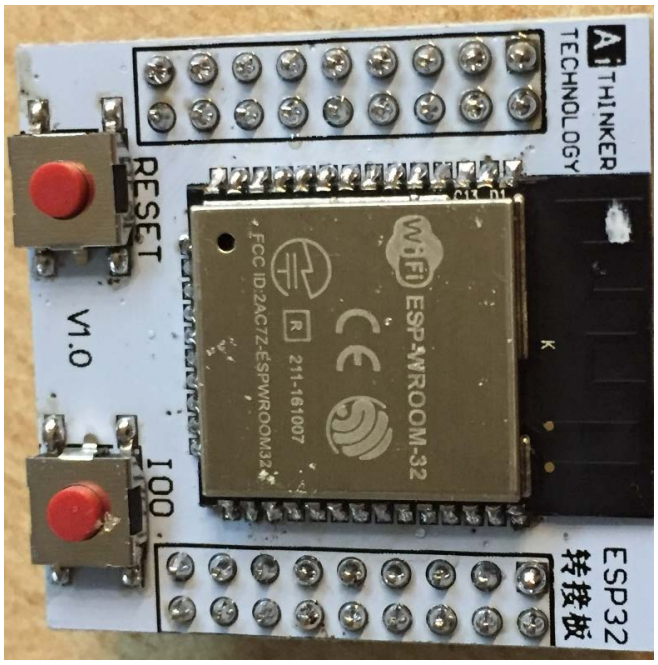
If necessary, the pin bar can now be aligned a little.

If both bars sit correctly, solder the pin on the other side:



Then, solder the pins on each side:

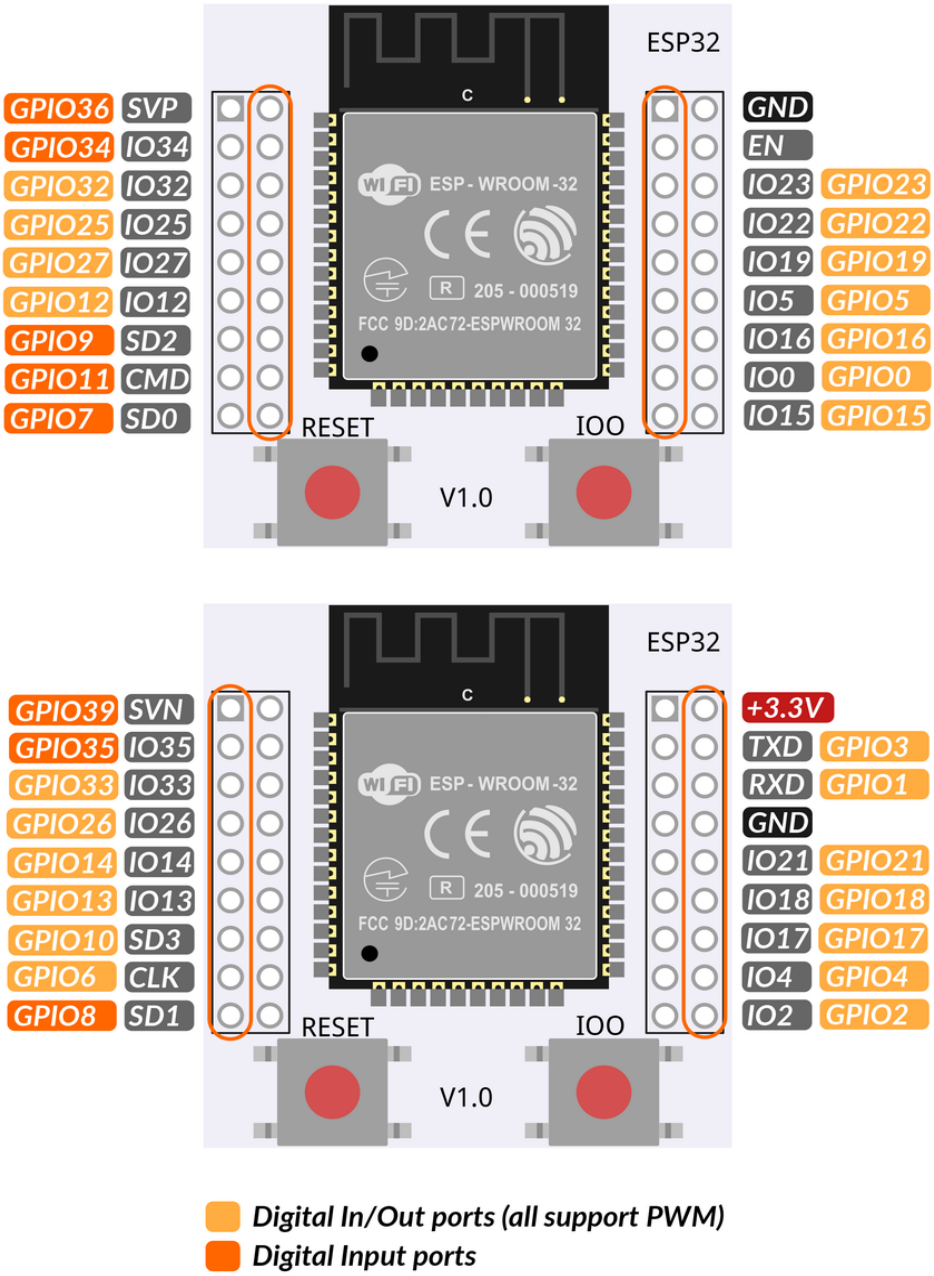




And the soldering is done.

The pinout

The module has 36 pins. The pinout is shown in the following image:



How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice. The Arduino IDE version used for this eBook is **1.8.15.**

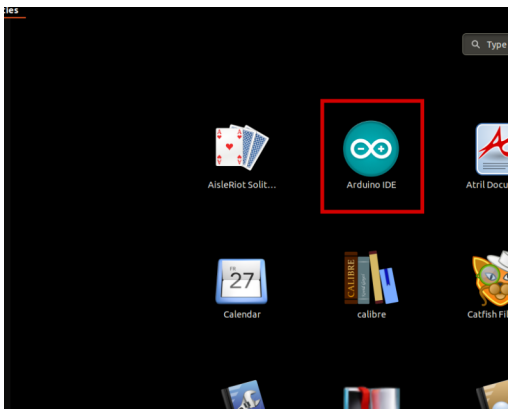
For *Windows* users, double click on the downloaded *.exe* file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

To run the first script in the terminal, open the terminal in the extracted directory and run the following command:
sh arduino-linux-setup.sh user_name
user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

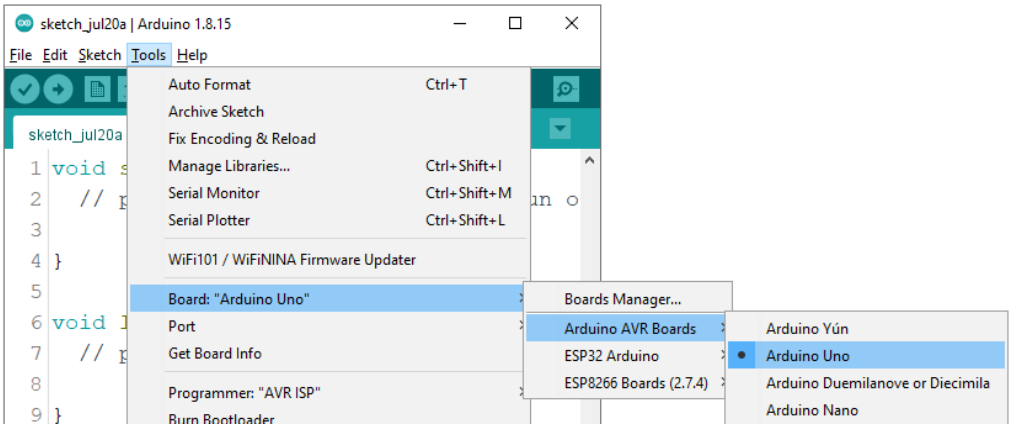
The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



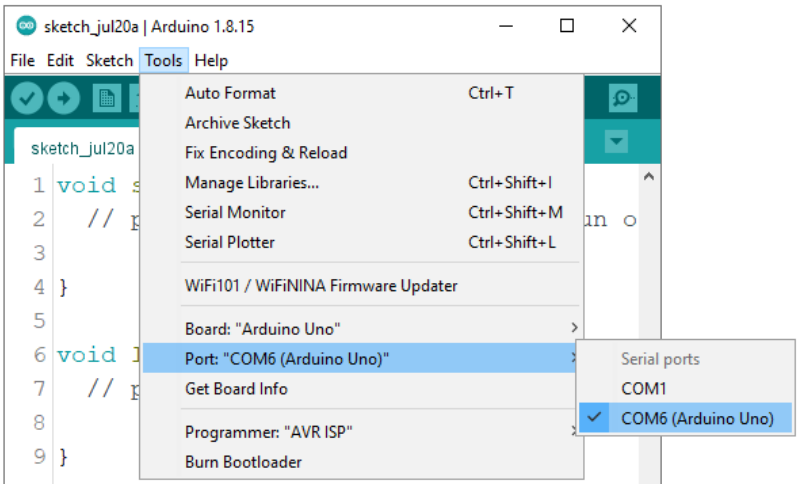
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect a microcontroller board. Open freshly installed Arduino IDE, and go to:
Tools > Board > {your board name here}
{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:



The port to which the microcontroller board is connected has to be selected. Go to: *Tools > Port > {port name goes here}*
and when the microcontroller board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:



For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

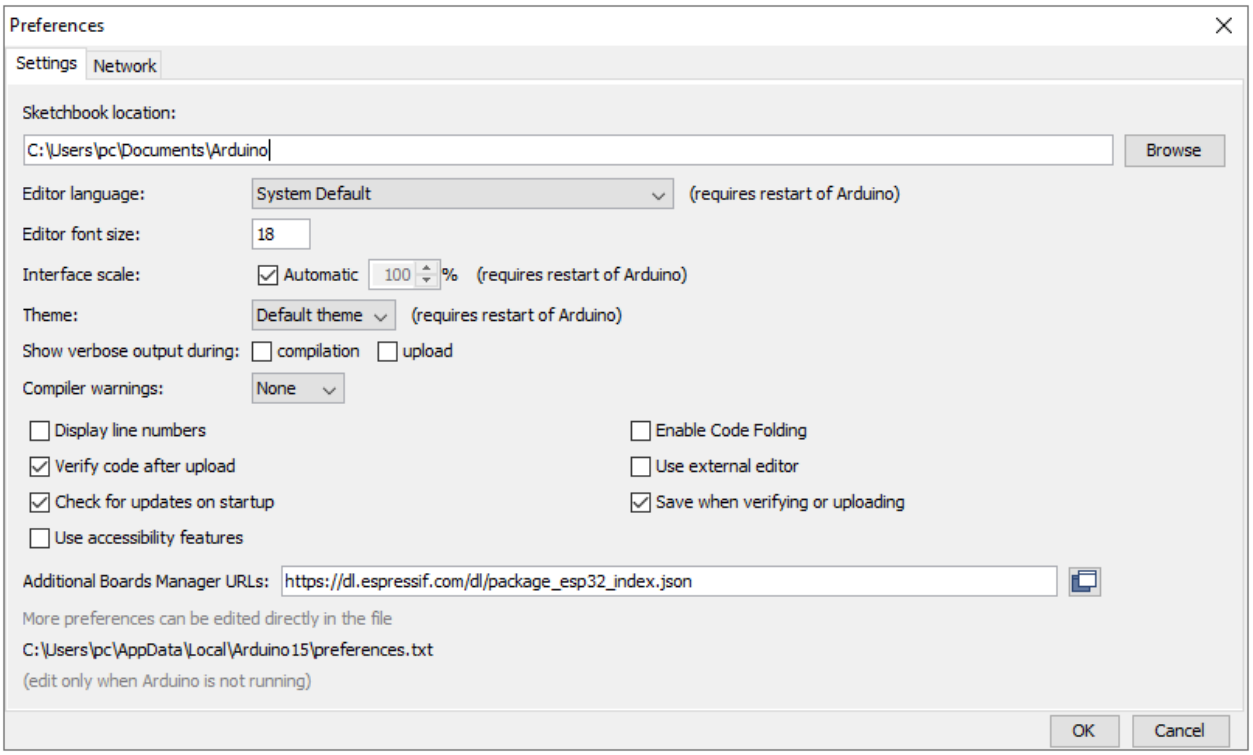
Driver setup

In order to use ESP32 with Arduino IDE, follow few easy steps. Before setting the Arduino IDE, the driver for the USB to Serial adapter (FT232-AZ) has to be installed. There is a support page that contains the drivers for Windows/Mac or Linux and can be chosen depending on which one is used. Drivers can be downloaded from the following link.

ESP32 support installation

Next, to install support for the ESP32 platform, open Arduino IDE and go to: *File* > *Preferences*, and find Additional URLs field.

Then copy the following URL:
https://dl.espressif.com/dl/package_esp32_index.json



Paste this link in the Additional URLs field. If one or more links are inside this field, just add one comma after the last link, paste new link after comma and click the *OK* button.

Open Arduino IDE again and go to:

Tools > Board > Boards Manager

When new window opens, type *esp32* in the search box and install the board called *esp32* made by *Espressif Systems*, as shown on the following image:

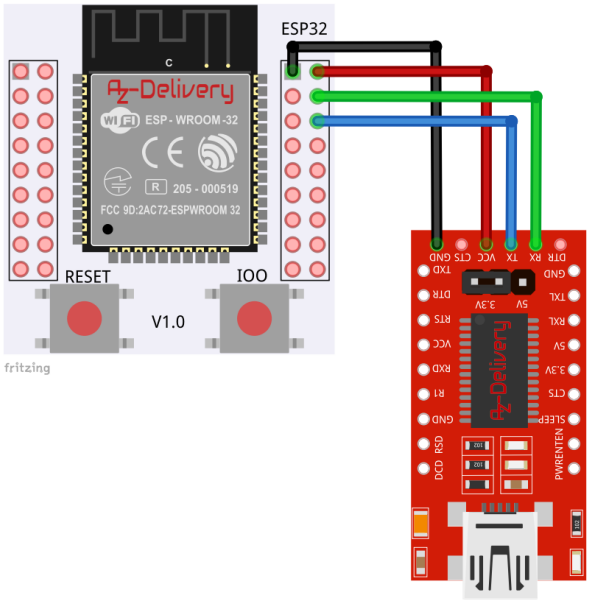
To select ESP32 board, go to:

Tools > Board > ESP32 Arduino > ESP32 Dev Module

To upload the sketch code to the ESP32 board, first select port to which you connected the board. Go to: *Tools > Port > {port name}*

Connecting ESP32 module with FT232-AZ Adapter

Connect the ESP32 with the FT232-AZ Adapter as shown on the following connection diagram:



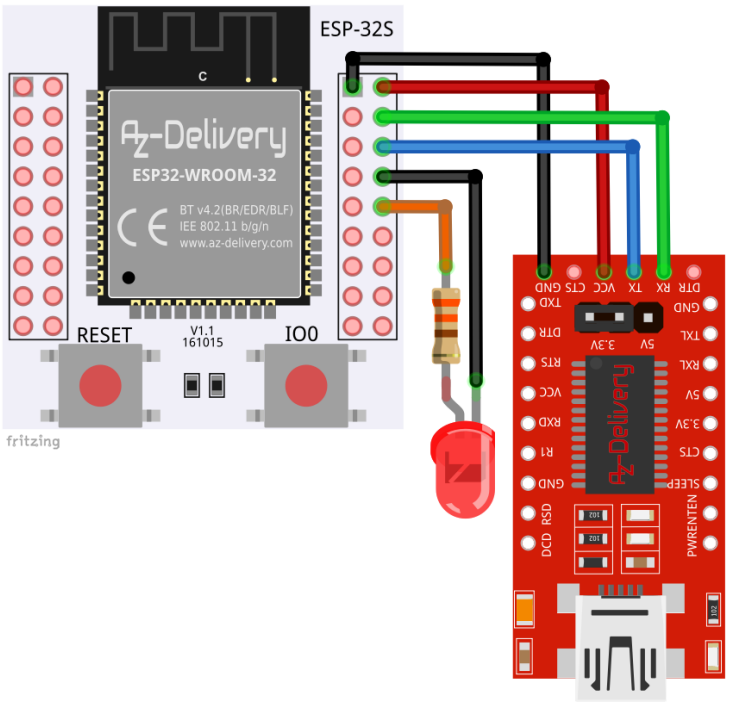
ESP32 Board pins	FT232-AZ pins	Wire color
GND	GND	**Black wire**
3.3V	VCC	**Red wire**
RXD	TX	**Blue wire**
TXD	RX	**Green wire**

NOTE: With the board version 1.1, a jumper has to be connected between GPIO0 and GPIO4.

WARNING: Before connecting the ESP32 and FT232-AZ adapter to the power source (USB), a jumper on the adapter has to set to 3.3V, otherwise the ESP32 may be damaged!

Programming the ESP32

In order to upload the code to the ESP32 and test it, first add two components to the circuit. Connect one LED and one resistor between GPIO21 and GND pins as on the following image:



NOTE: The resistor value is 330Ohms

Next, open the Arduino IDE and go to *File* > *Examples* > *01. Basics* and open the *Blink* sketch example.

Sketch examples

When *Blink* sketch is opened, the following code has to be changed:

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

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

Typical Arduino version *LED_BUILTIN* function does not work. The ESP32 has no built-in LED. To make the external LED flashing, the *LED_BUILTIN* function has to be replaced with the pin number to which the LED is connected. For example, the LED is connected to the GPIO21 which is freely programmable IO pin, so instead of *LED_BUILTIN* function, the number of the GPIO pin is used, as on the following example:

```
digitalWrite(21, HIGH);
```

The LED can be connected to any other IO pin.

The following code is a working sketch without comments:

```
void setup() {
  pinMode(21, OUTPUT);
}

void loop() {
  digitalWrite(21, HIGH);
  delay(1000);
  digitalWrite(21, LOW);
  delay(1000);
}
```

Now is the time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

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