Arduino IDE Compatible Microcontrollers

Written by Mark Webster, Ousema Zayati, and ??

Summary
Learn to use the Arduino IDE to program other compatible microcontrollers.

Introduction

The Arduino IDE is a popular, well supported, and easy to use interface for programming microcontrollers. Since the Arduino Uno board came out in 2005, microcontrollers have advanced significantly. There are smaller, lower power devices like the ATtiny85. There are more powerful, faster devices like the ESP32. Hobbyists wanted a way to use the familiar and easy Arduino IDE to work with these new devices. Well, there is a way; the Arduino IDE Board Manager. It allows other devices, called cores, to be programmed with the same IDE.

There are so many boards out there in the world, a complete list isn't practical. Wikipedia lists some of them:
https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems

This workshop only looks at a couple of the popular microcontroller boards. Three general types of boards are used: one with no USB, one which requires separate software, and one which is more powerful and complex than an Arduino Uno.

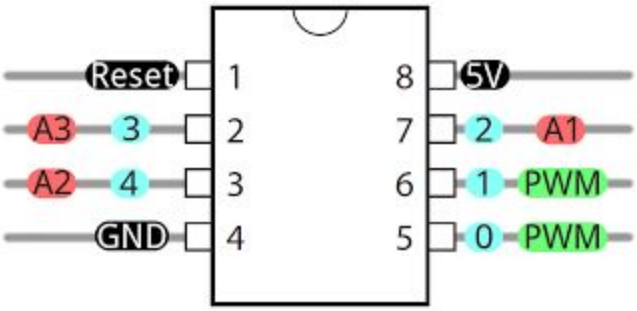
A more comprehensive list, plus the JSON links for the boards, look at the Github page:
https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls

ATtiny

The ATtiny, also called TinyAVR are chips made by ATMEL who also make the microcontroller used on the Arduino Uno (the ATmega328). ATtiny chips typically have 4,8, or 20 pins. They do not have USB connectors so they must be programmed with another Arduino as an ISP or with a FTDI breakout.

An ATtiny is very small, inexpensive and low power. But that minimal footprint has a price-- it has no hardware UART for serial, fewer analog inputs, fewer digital pins, fewer PWM pins,fewer interrupts, and less memory.

For simple Arduino sketches, it can be much lower cost and lower power drain.
Here is the popular ATtiny85

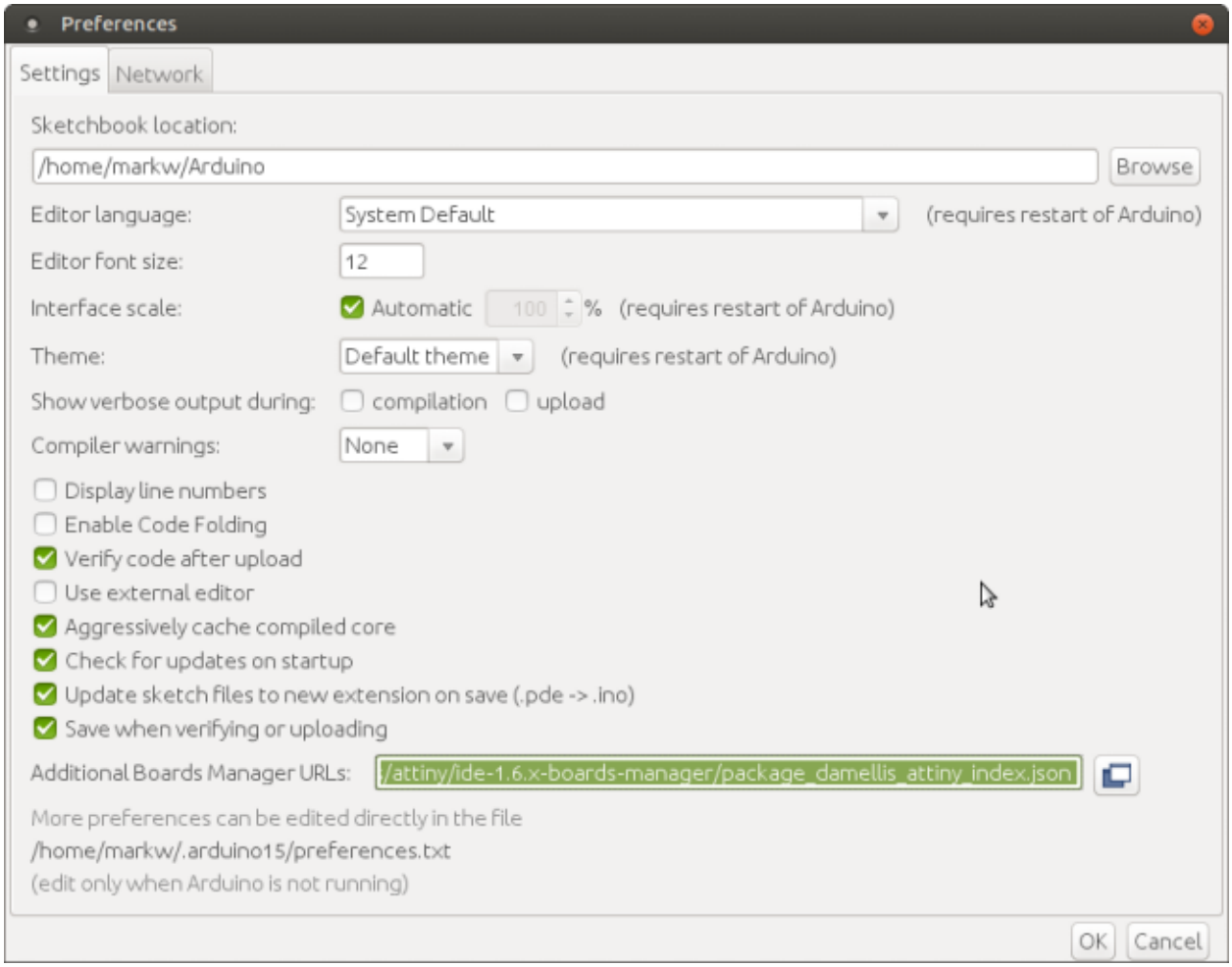


Notice it is a 5V chip, has three analog input pins (A1-A3) and two digital pins which are also PWM (pulse width modulated). A lot can be done with those limitations.

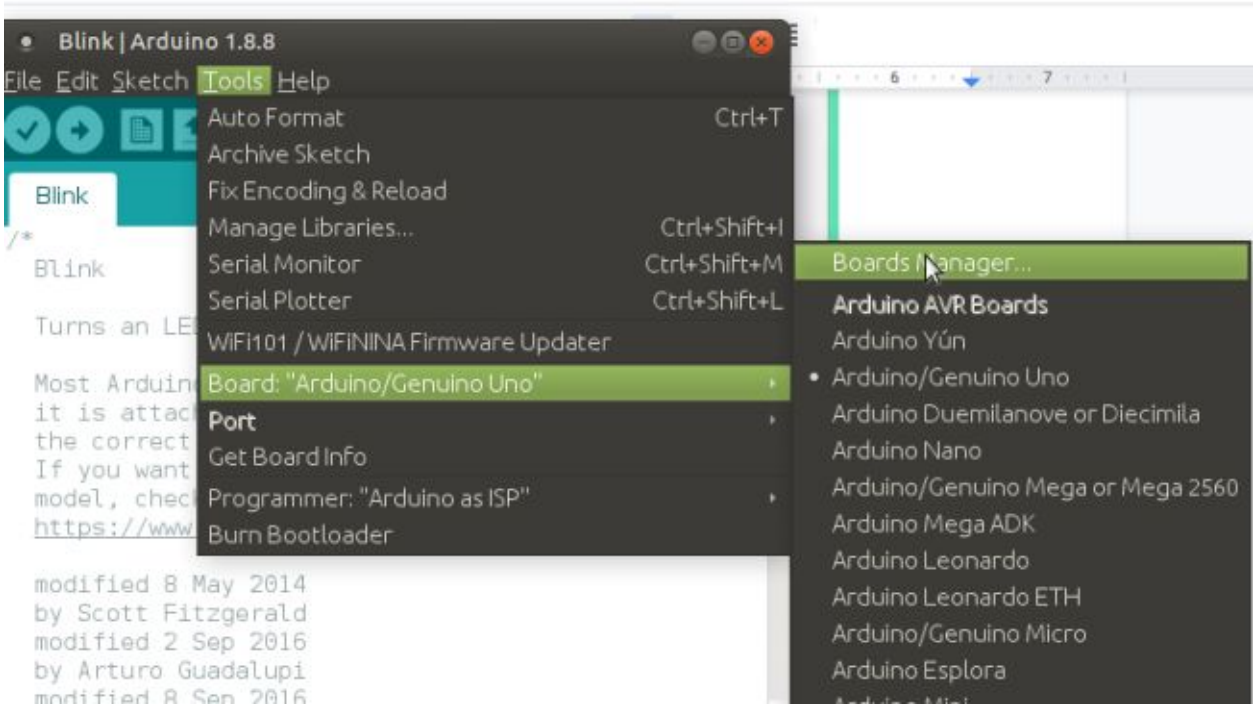
The ATtiny85 has 8KB ISP flash memory, 512B EEPROM, 512-Byte SRA. This is ¼ the memory of an Arduino Uno, but many sketches take less than 32K.

To add support for ATtiny boards use this JSON url in the Preferences dialog as additional boards. The link is found on the Github page listed in the introduction above.

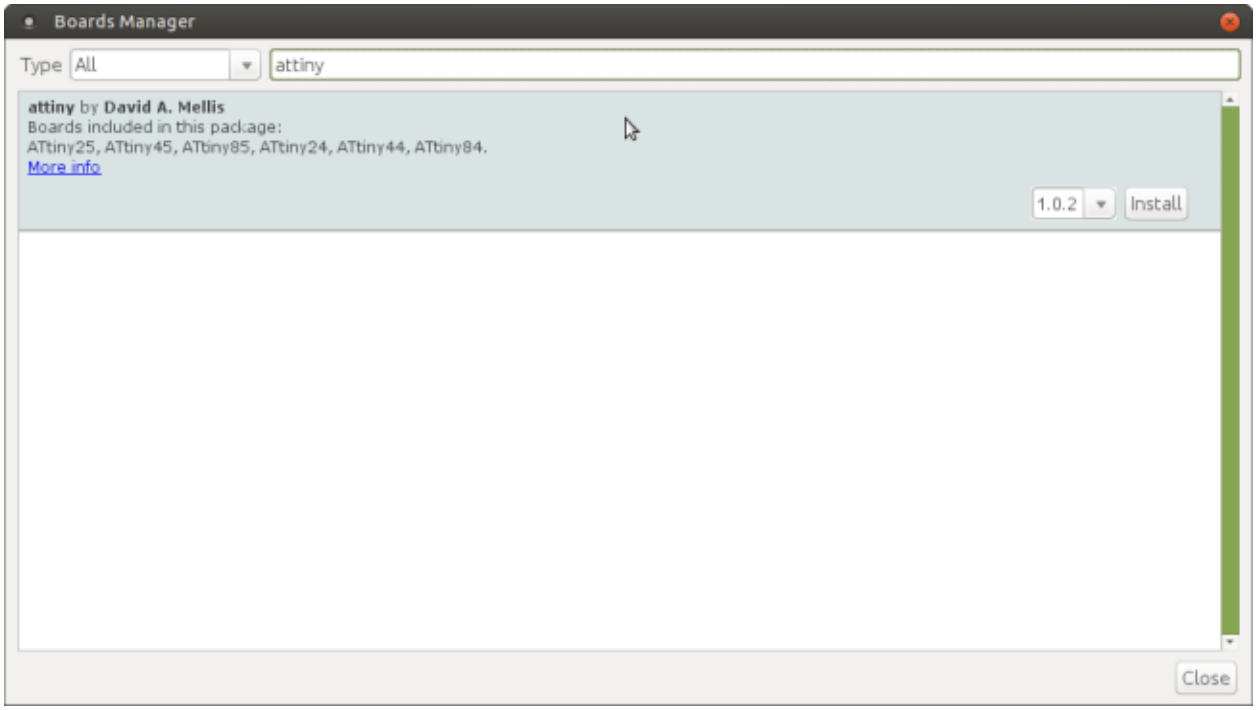
https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json



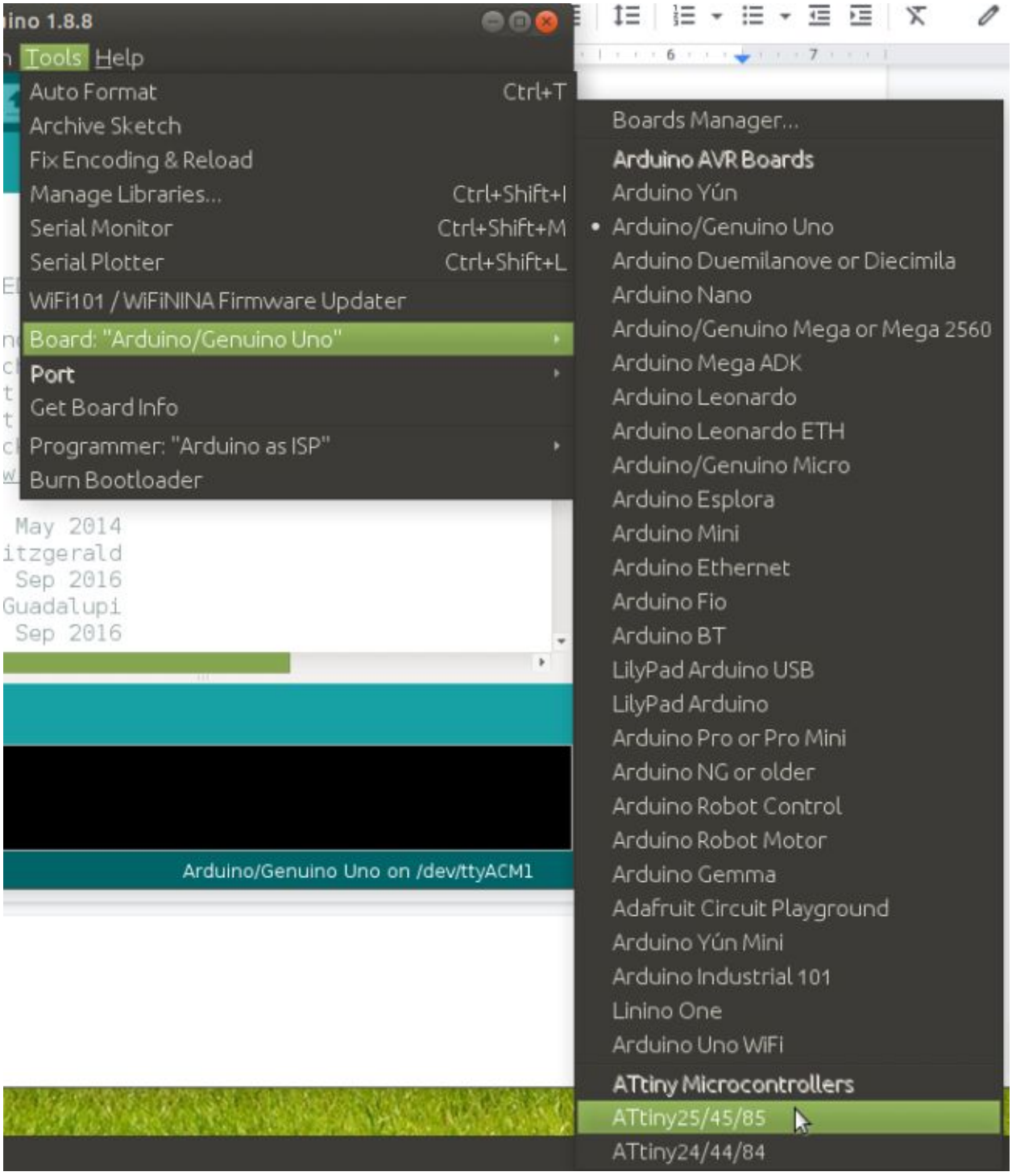
Once the board manager URL is included, go to the Tools ->Boards menu and select the Board Manager menu item at the top.



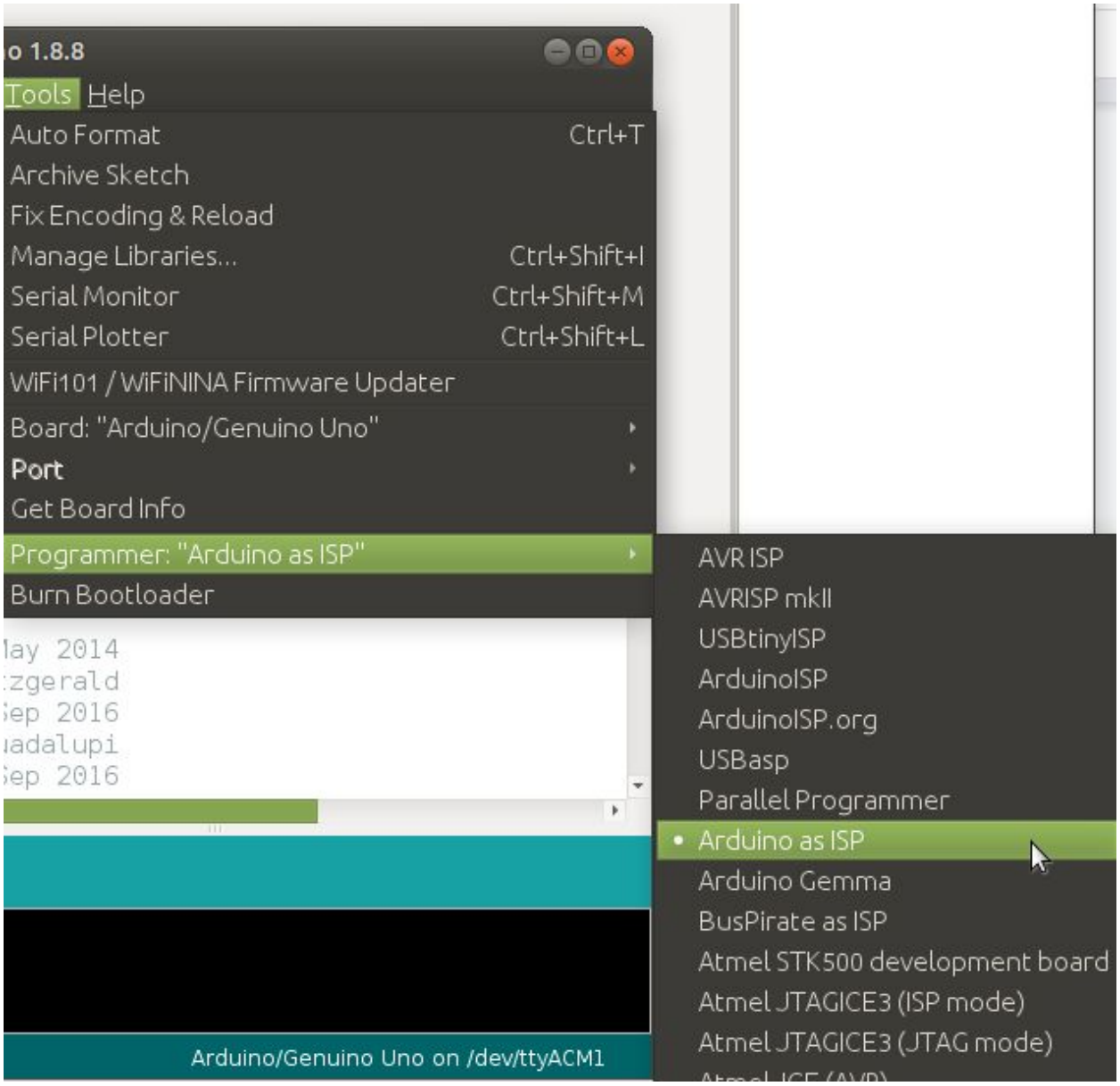
Search in the board manager for ATtiny. Select the board you've added and install it by clicking on the install button.



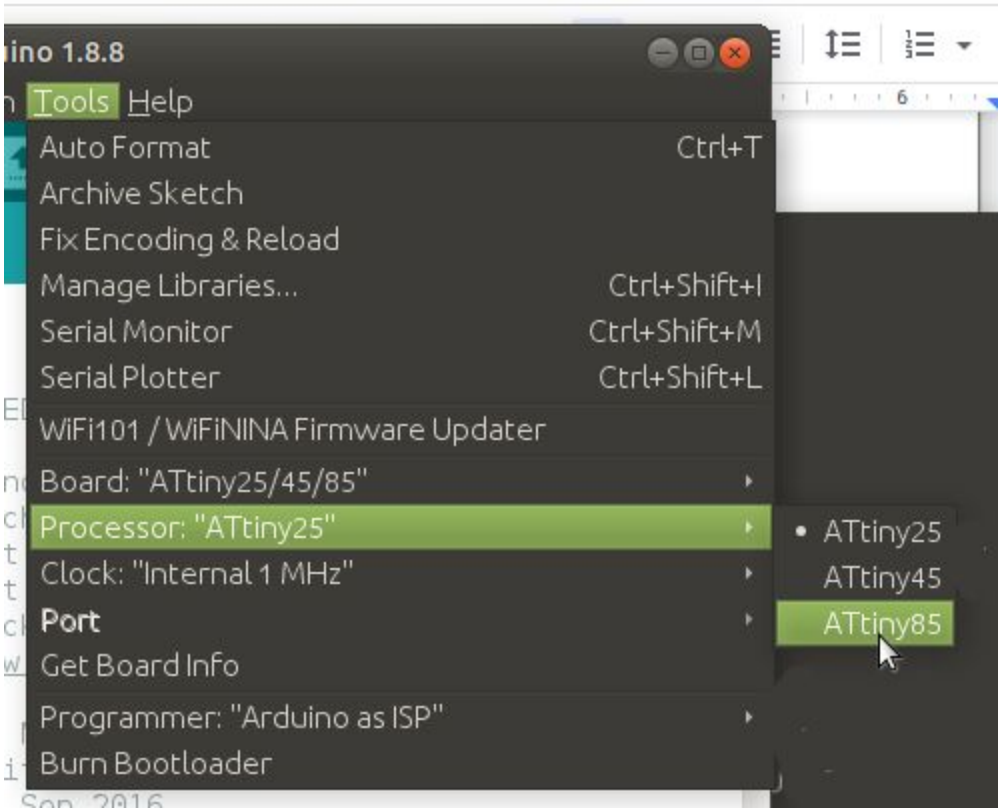
Once the new ATtiny board is installed, on the board manager scroll down to the new section with ATtiny and select that. Then under Processor on the Board menu, pick ATtiny85.



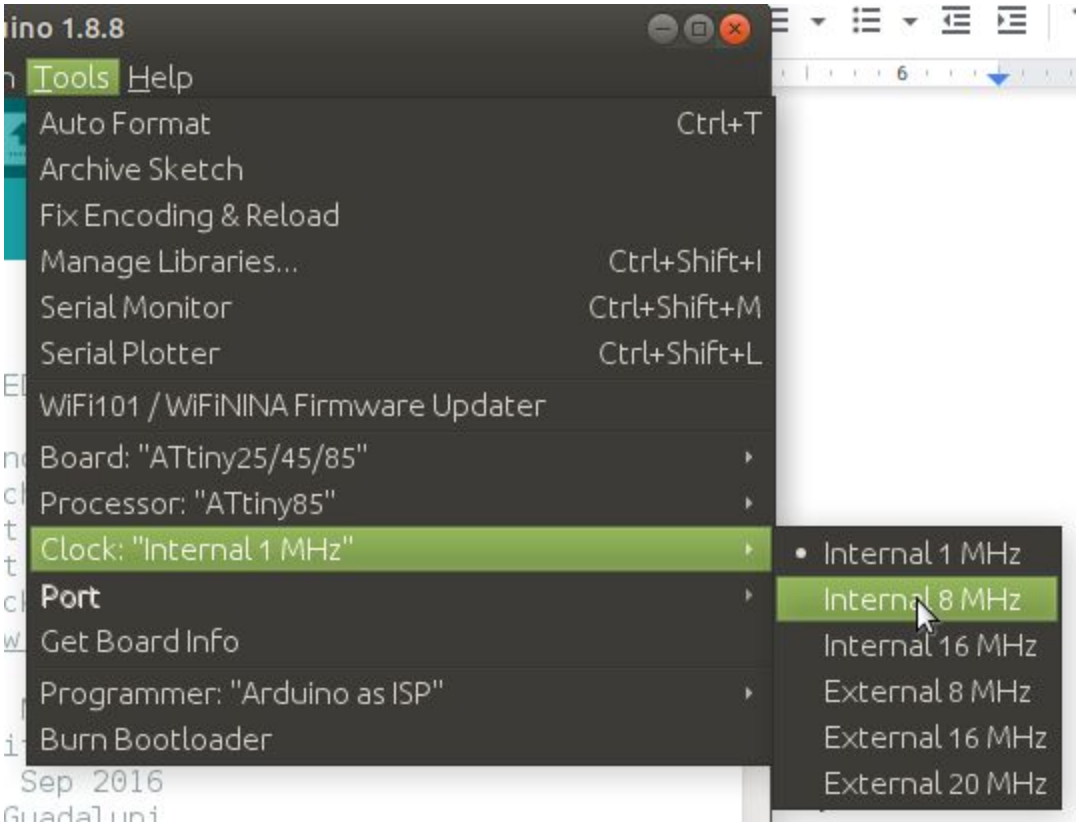
On the Tools menu, select Programmer, and choose “Arduino as ISP”.



Select the processor for the ATtiny85.



By default the ATtiny85 runs at 1 mhz. If this is fast enough, do nothing. Otherwise, select on the Tools->Clock menu select 8 mhz. This will set the chip to 8 mhz.

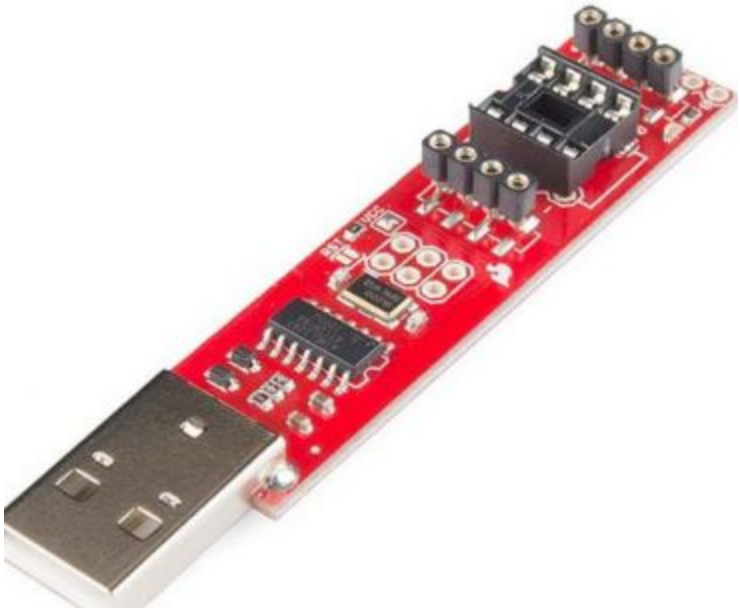


Then on the Tools menu select Burn Bootloader

After connecting wires from the Arduino to ATtiny85, modify the default Blink example to Pin 0 for the LED pin. Compile and upload. Stand back and watch the blinking lights.

With a piece of a protoboard it is easy to make a shield for programming ATtiny chips. (picture)

FTDI board
It is possible to program an ATtiny with an FTDI breakout board. Here is the Tiny AVR USB programmer from Sparkfun.



At the opposite end from the USB, it has a socket to put the ATtiny85. Go to the documents tab in Sparkfun to download the USBTinyISP driver to use the FTDI. Sparkfun also has an ATtiny85 reference card.
https://cdn.sparkfun.com/assets/2/8/b/a/a/Tiny_QuickRef_v2_2.pdf

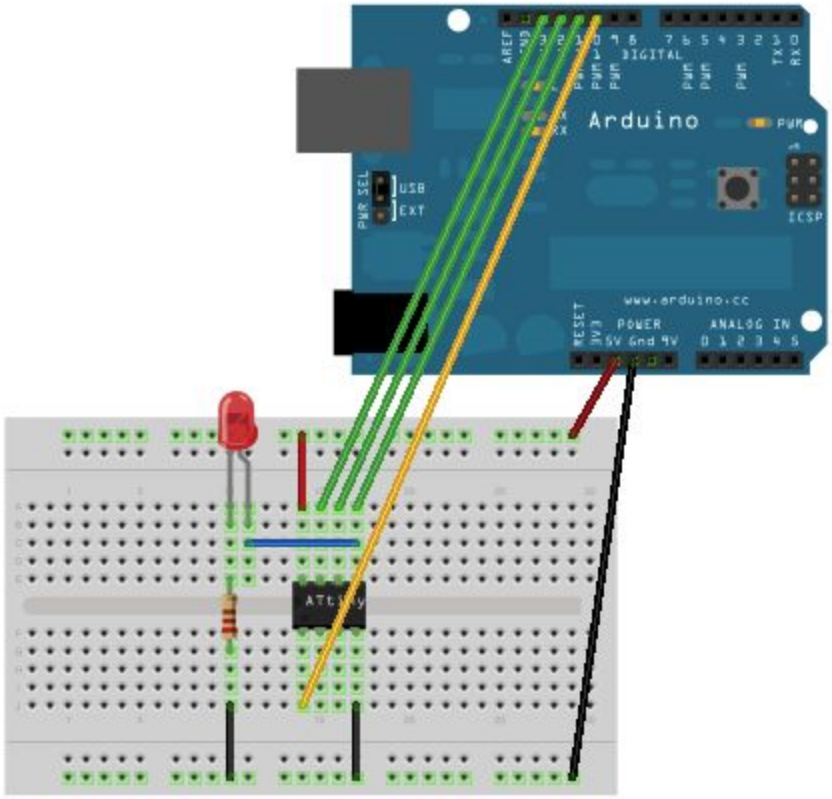
Here is a picture of using ATtiny85 with an Arduino Uno as an ISP to program it. Build this and blink an LED.

Exercise 1: ATtiny85

From the Examples menu, select, compile and upload the ArduinoISP sketch. The configures the Arduino Uno to be an ISP. Choose the Arduino Uno as the board.

Note: The reset button will have to be pressed after the ISP sketch is uploaded.

The wiring diagram is:



Connect the Arduino to the ATtiny as follows:
Arduino +5V ---> ATtiny Pin 8
Arduino Ground ---> ATtiny Pin 4
Arduino Pin 10 ---> ATtiny Pin 1
Arduino Pin 11 ---> ATtiny Pin 5
Arduino Pin 12 ---> ATtiny Pin 6
Arduino Pin 13 ---> ATtiny Pin 7

Then load or type in the ATtiny85 blink program.

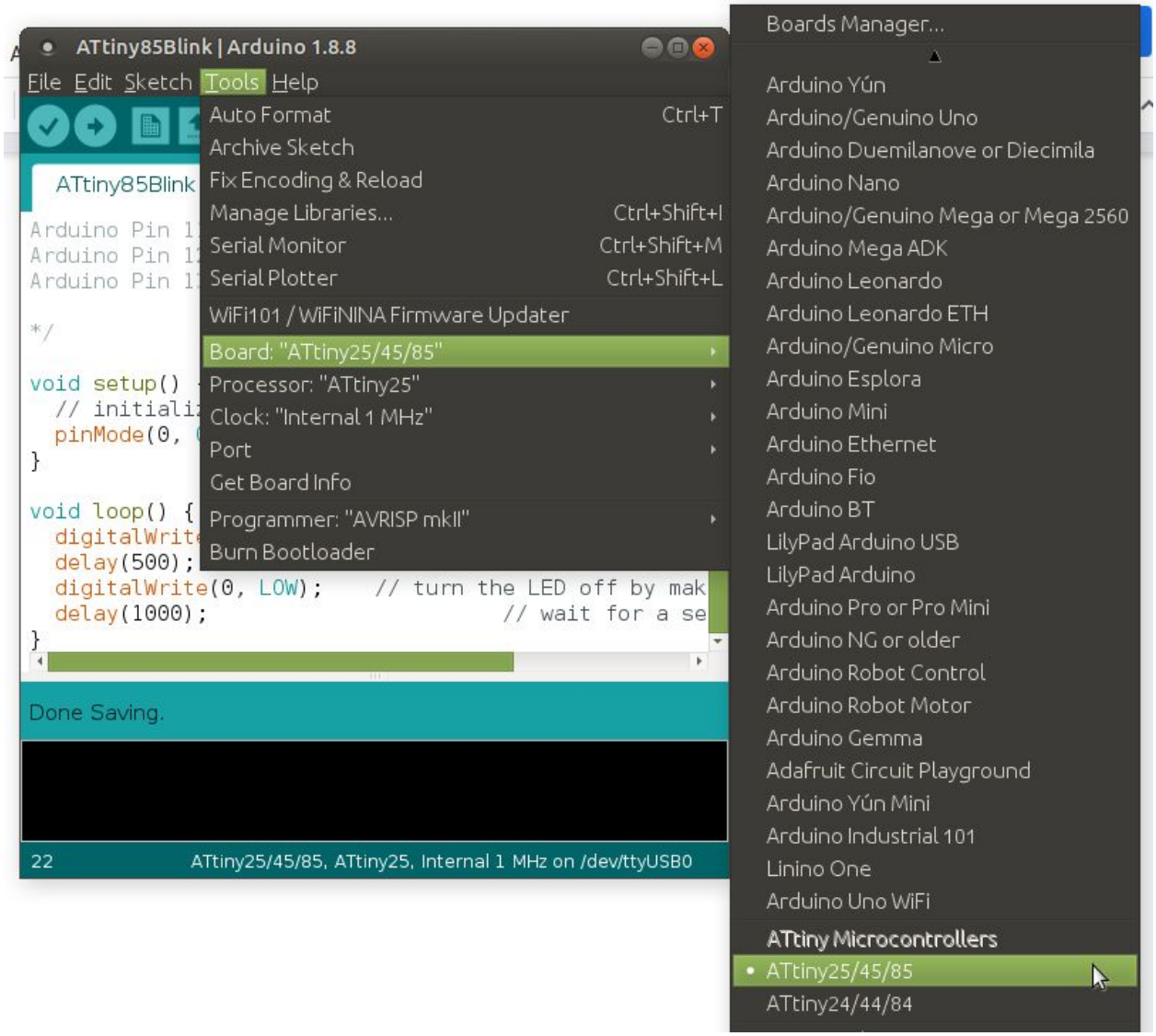
```
/*
 * BLINK for an ATtiny85
 *
```

Connect the Arduino to the ATtiny as follows:
Arduino +5V ---> ATtiny Pin 8
Arduino Ground ---> ATtiny Pin 4
Arduino Pin 10 ---> ATtiny Pin 1
Arduino Pin 11 ---> ATtiny Pin 5
Arduino Pin 12 ---> ATtiny Pin 6
Arduino Pin 13 ---> ATtiny Pin 7

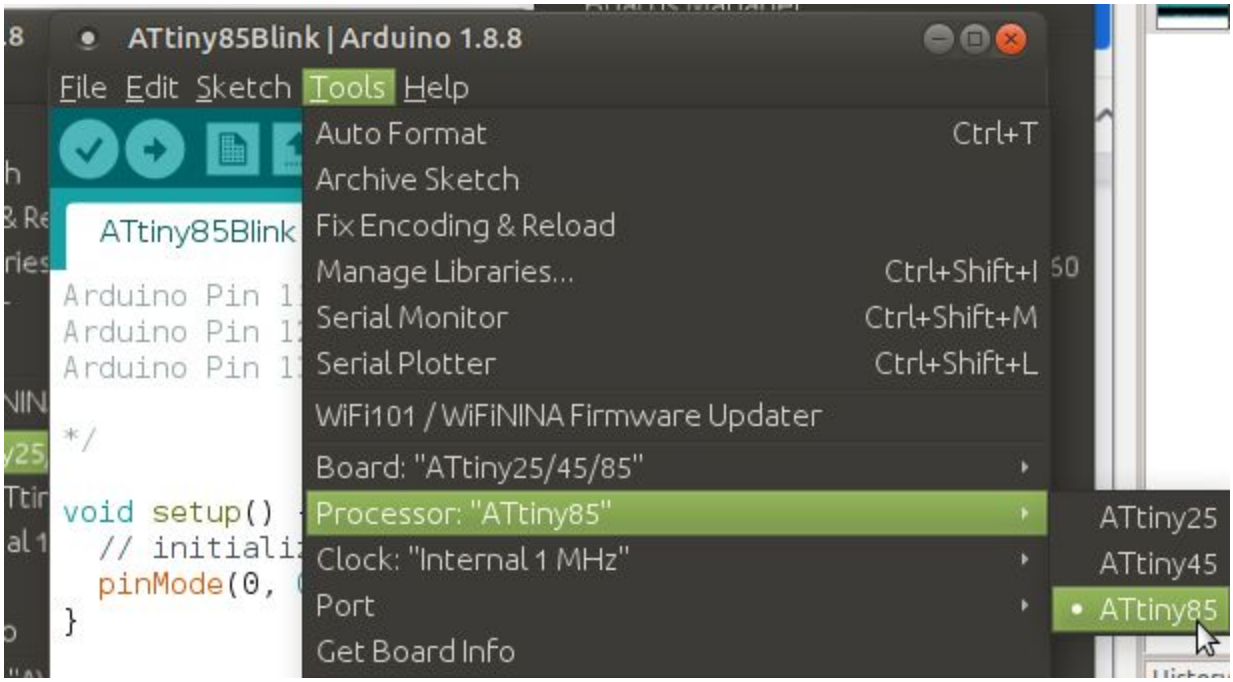
Use ArduinoISP. But previously have uploaded the

```
ArduinoISP sketch.
*/

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(0, OUTPUT);
}

void loop() {
  digitalWrite(0, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                      // wait for a second
  digitalWrite(0, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Modify the blink sketch to use the pin connected to the LED.
The ATTiny85 does not have a pin 13, so use pin 0.

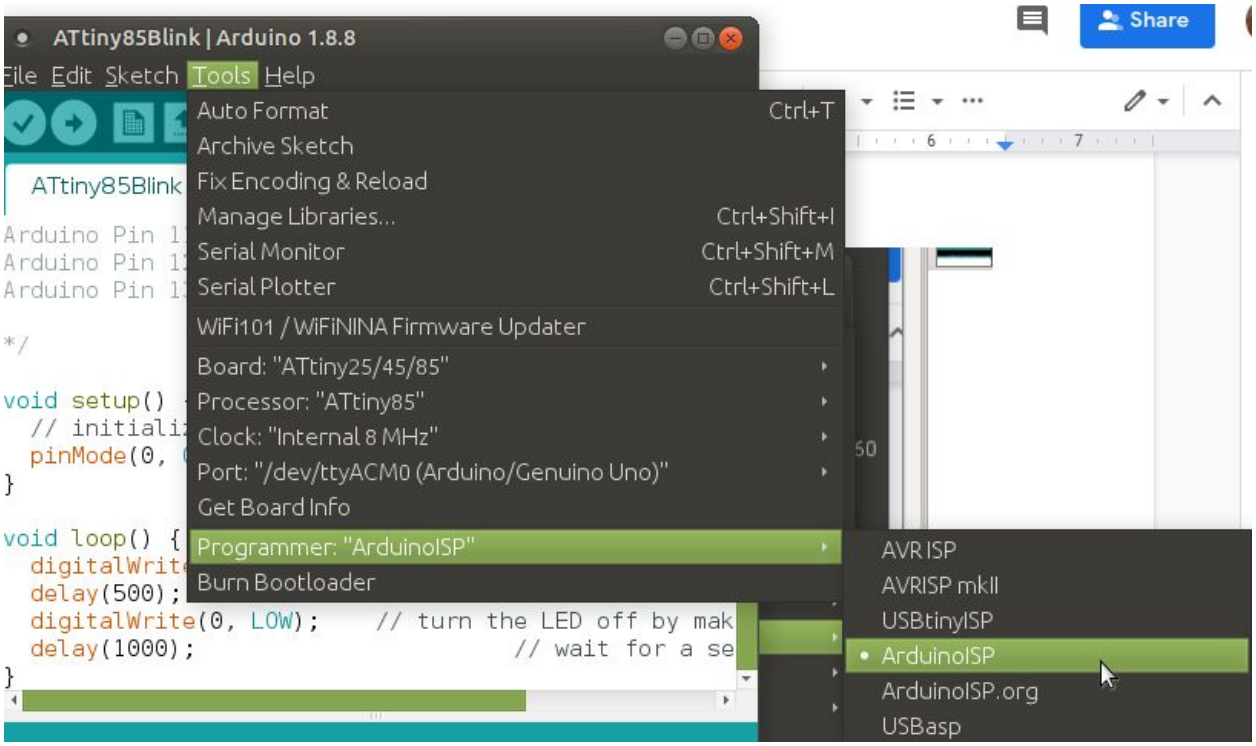


Set the device to ATtiny85.



Be sure to set the port and the clock speed.

Then set the Tools->Programmer to ArduinoISP



To use the FTDI board

Teensy

The Teensy board is more powerful than the ATtiny and more powerful than the Arduino Uno. It has more memory, runs faster, has more IO pins than an Arduino. What's not to like! There are differences, such as it is a 3.3V board which might be incompatible with some Arduino modules without a voltage level converter.

The disadvantage of a Teensy is 3-4 times the cost of an Arduino Uno clone.

The installation is not through the Board Manager, but is using separate software called Teensyduino.

A description of programming a Teensy is at adafruit.com.
https://learn.adafruit.com/usb-snes-gamepad/programming-the-teensy

A link to the Teensyduinoinstallation process is https://www.pjrc.com/teensy/td_download.html

You will need the teensyloader https://www.pjrc.com/teensy/loader.html

On the Tools->Board menu select the Teensy chip you are using.
The USB type is Serial, the CPU speed is 72,120 or 180 mhz depending on the board.

Then you are ready to create a program/sketch in the Arduino IDE and run it.

Exercise 2: Teensy 4.6

Set up a photoresistor and resistor voltage divider. Measure the voltage between the photoresistor and the resistor. Send that voltage to the Arduino Serial Monitor and Serial Plotter.

ESP8266

An older version of the ESP microcontroller is the ESP8266 which has become quite inexpensive, $3 from Amazon as a NodeMCU unit.

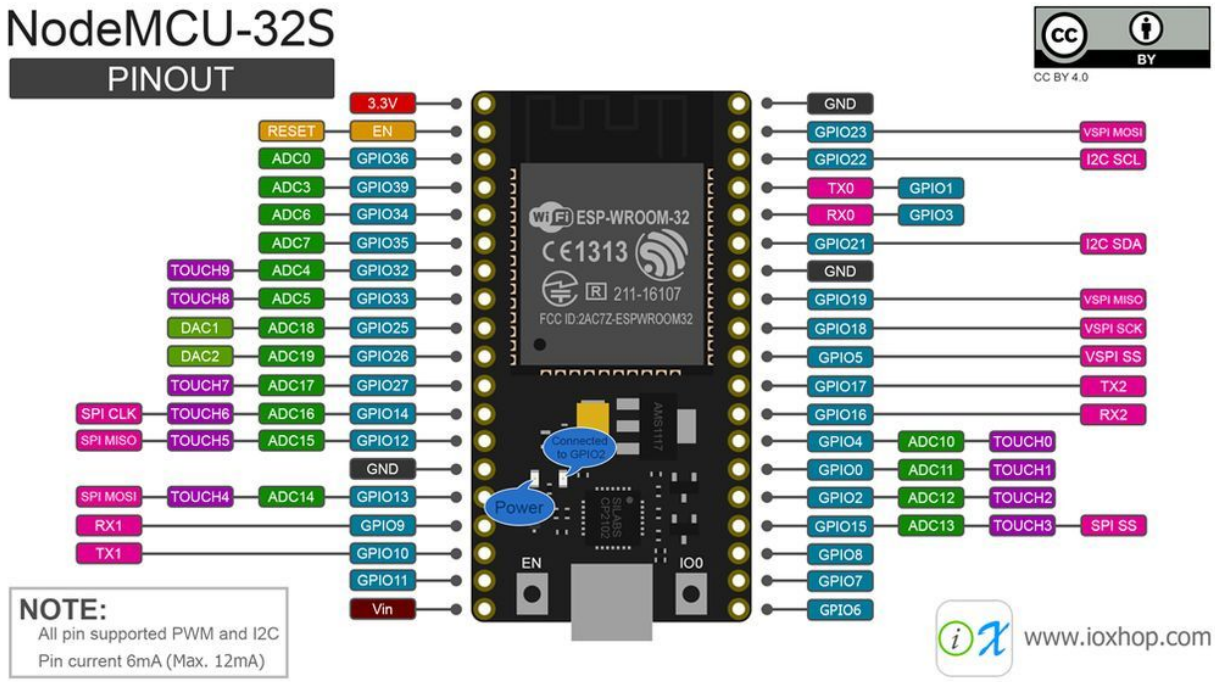
The ESP8266 has onboard Wifi so it is less expensive than an Arduino + Wifi module.
The ESP8266 has a many online tutorials and can be programmed in a manner similar to the newer ESP32.

ESP32

The ESP8266 was and still is a very popular microcontroller which has WiFi onboard. The successor to the 8266 is the ESP32. Much of the approach to the ESP32 also applies to the 8266 chip.

The ESP32 is dramatically more powerful than an Arduino Uno at the same cost as an Arduino Uno clone. Its main features are onboard Wifi and bluetooth.

Each manufacturer has slightly different pinouts. Below is a common pattern. Often the pinouts are listed on a piece of paper that ships with the ESP32.

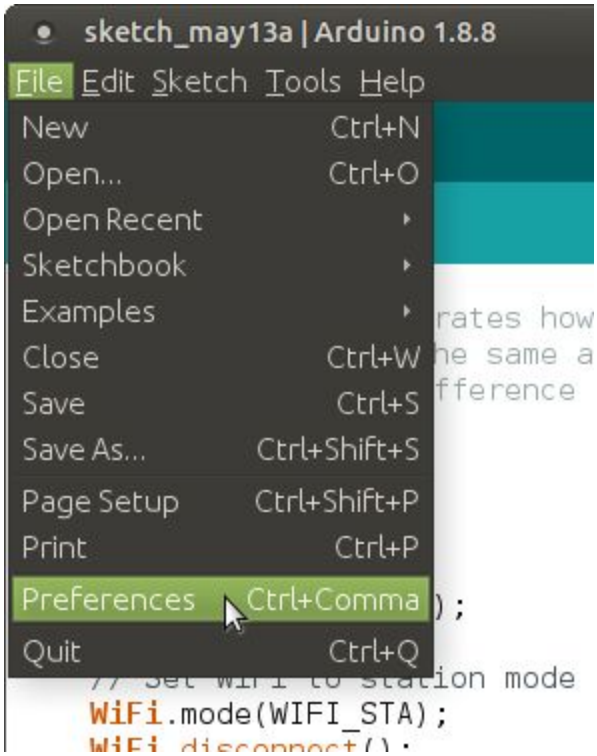


The ESP32 has 4 megabytes of memory, the Arduino Uno only has 32 kilobyles.
The ESP32 boards are smaller than an Uno, and each core of the dual core chip runs about 60 times faster than the Uno single core. Clock speed is higher and power consumption is lower than an Uno. The ESP32 has built-in wifi and bluetooth, as well as a built-in hall sensor, temperature sensor, and touch sensor. It has I2C and SPI built in, as well as UART and ethernet. The ESP32 has two DAC pins as well as ADC while the Uno only has ADC for input.

The ESP32 is a microcontroller but several different manufacturers have made a board containing the ESP32. The Shanghai company Espressif Systems makes the chips, the most popular is the ESP32-1WROOM-32 board. For example, Adafruit has a HUZZAH32-32 ESP32 Feather board. Sparkfun has a Sparkfun ESP32 Thing board. Pycom makes a board running micropython. Wikipedia has a page listing most of the ESP32 boards. Unfortunately, many of the boards require a different board library when using the Arduino IDE. The most common library is the generic github /espressif/arduino-32 library

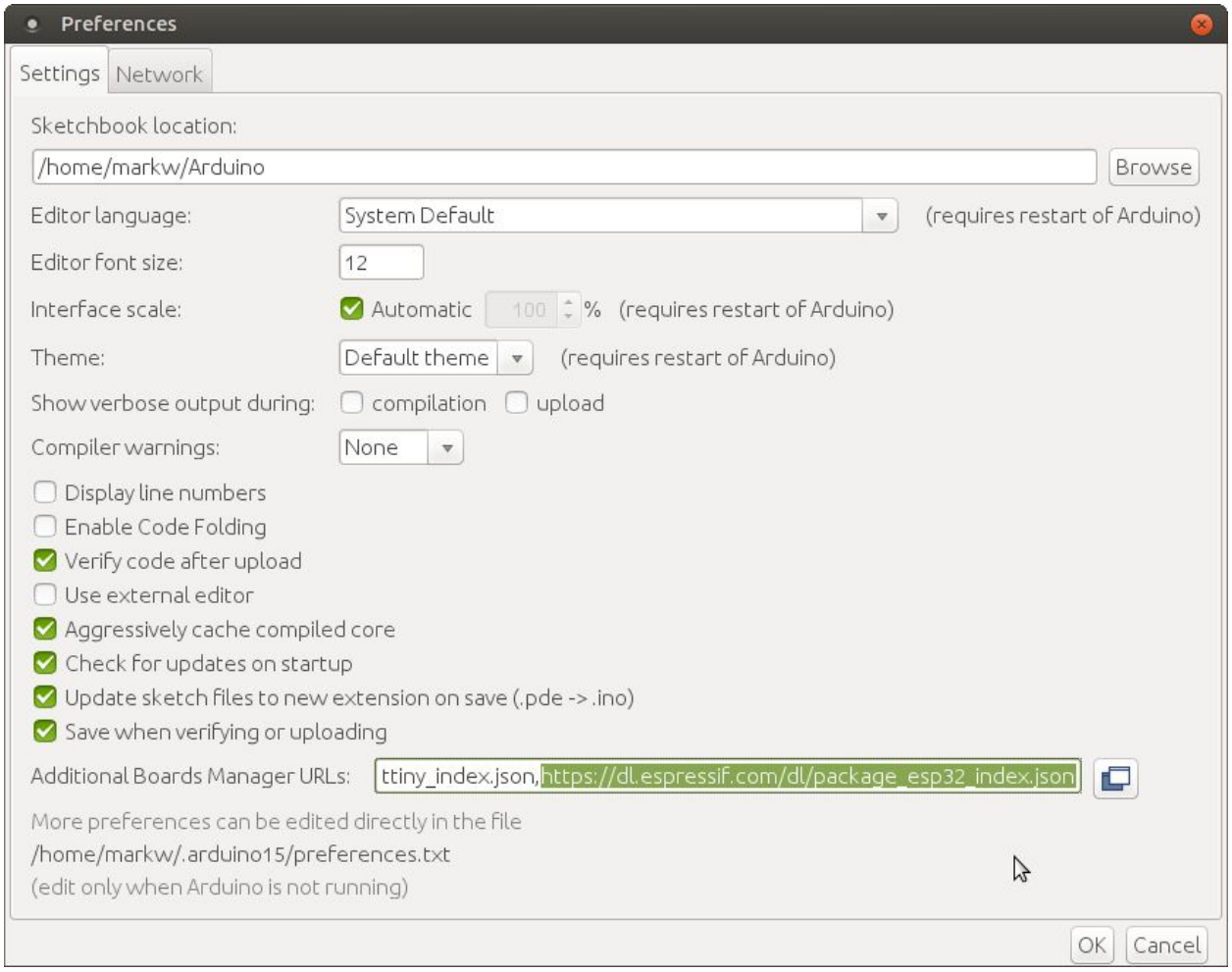
Before using the ESP32 one must install the board using the IDE board library manager. For screen shots of the board manager and Preferences dialog see the description for the ATtiny.

Start Arduino and open Preferences dialog.



In the input box for additional boards manager URLs enter:

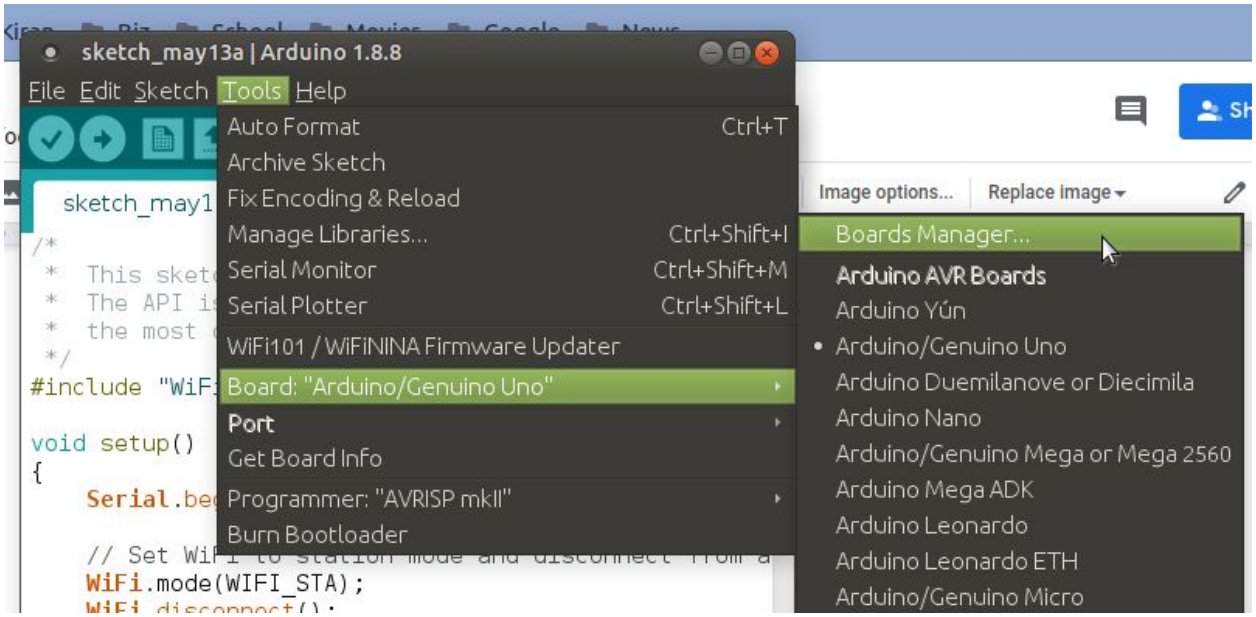
https://dl.espressif.com/dl/package_esp32_index.json



A JSON url is a formatted data structure commonly used in web programming. Multiple board JSON urls can be entered separated by a comma.

Sometimes the Arduino IDE must be restarted to load the new board manager.

After entering the JSON url, open Boards Manager from Tools -> Board menu.



In the board manager, search for ESP32 install the expressif esp32 platform

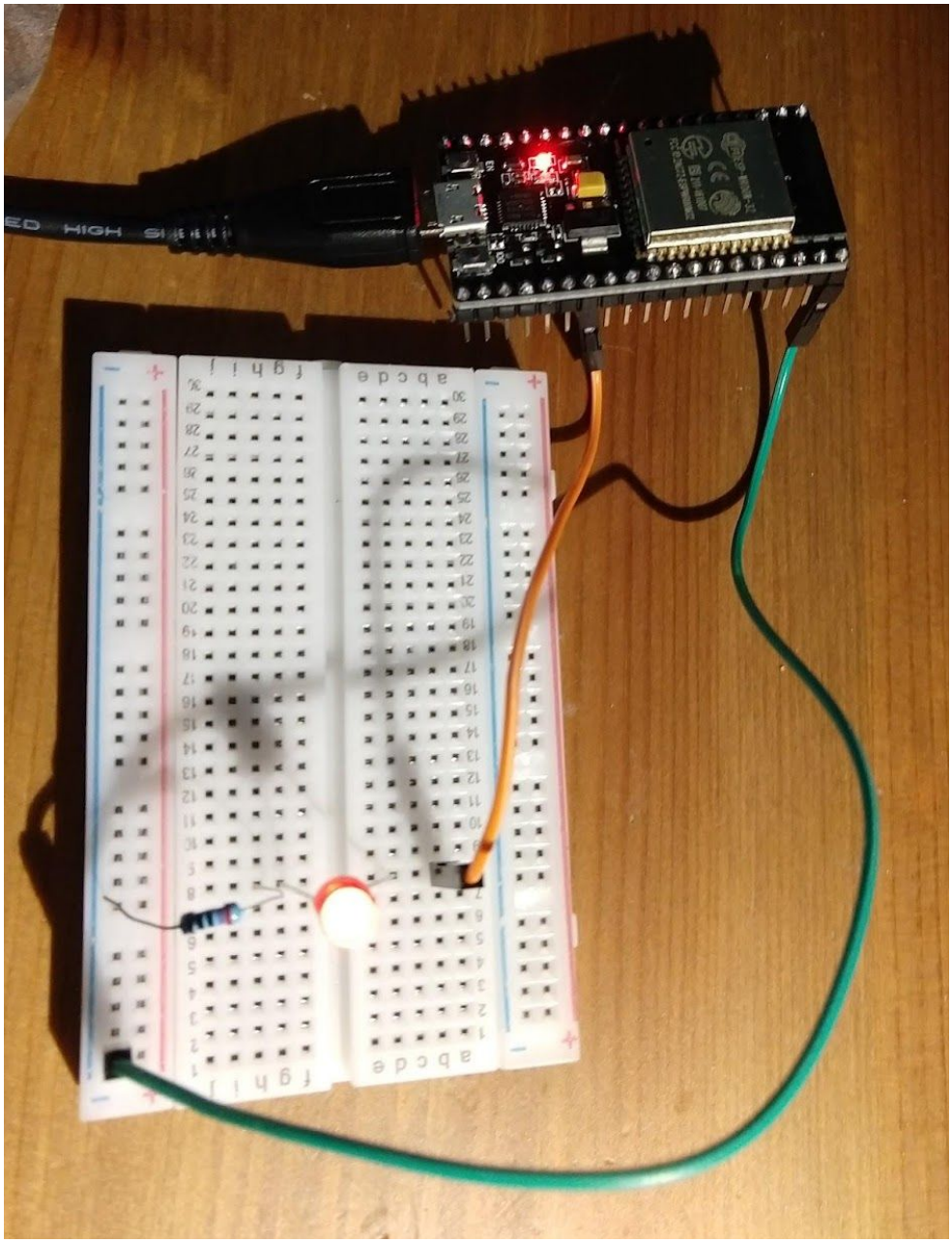
(and don't forget to select your ESP32 board from Tools -> Board menu after installation). Usually it is the ESP32 Dev board.

Note: the ESP32 development environment uses Python 2.7 language so it must be installed. Also to use the Serial.print() Arduino method modules such as pyserial for Python 2.7 will need to be installed.

Exercise 3: ESP32 Blink

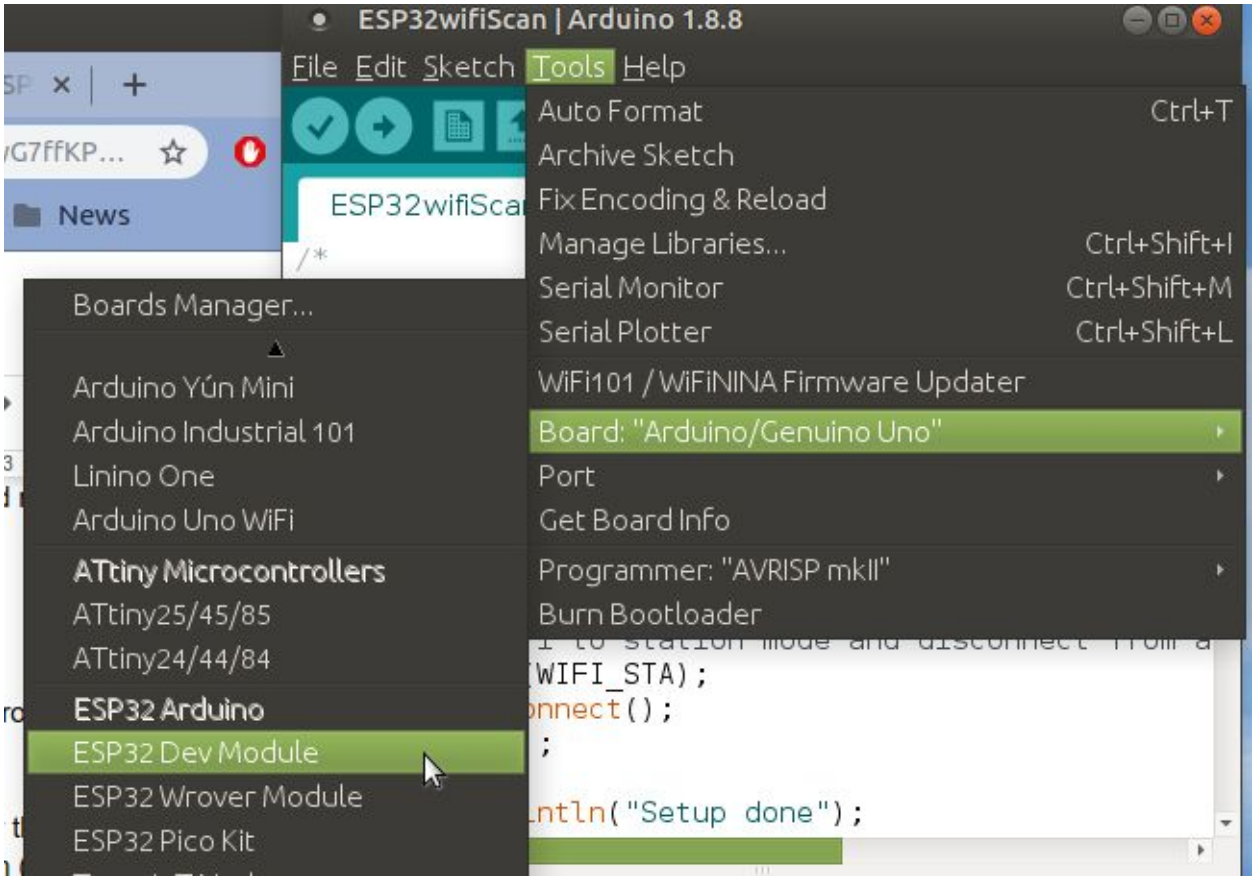
Hardware: Connect an ESP32 via USB micro serial cable. Breadboard, LED, 330 ohm resistor, M-F jumper wires.

Connect a M-F jumper from gpio pin to the LED positive. Pinouts vary by manufacturer but can often be found underneath the ESP32 board. Connect another jumper from the ground pin (far from the micro usb connector).



Open the Arduino IDE and save a sketch as ESP32blink (or any other name). Make sure the board has been installed from the Board manager.

Select the ESP Dev board from the Tools->Board menu

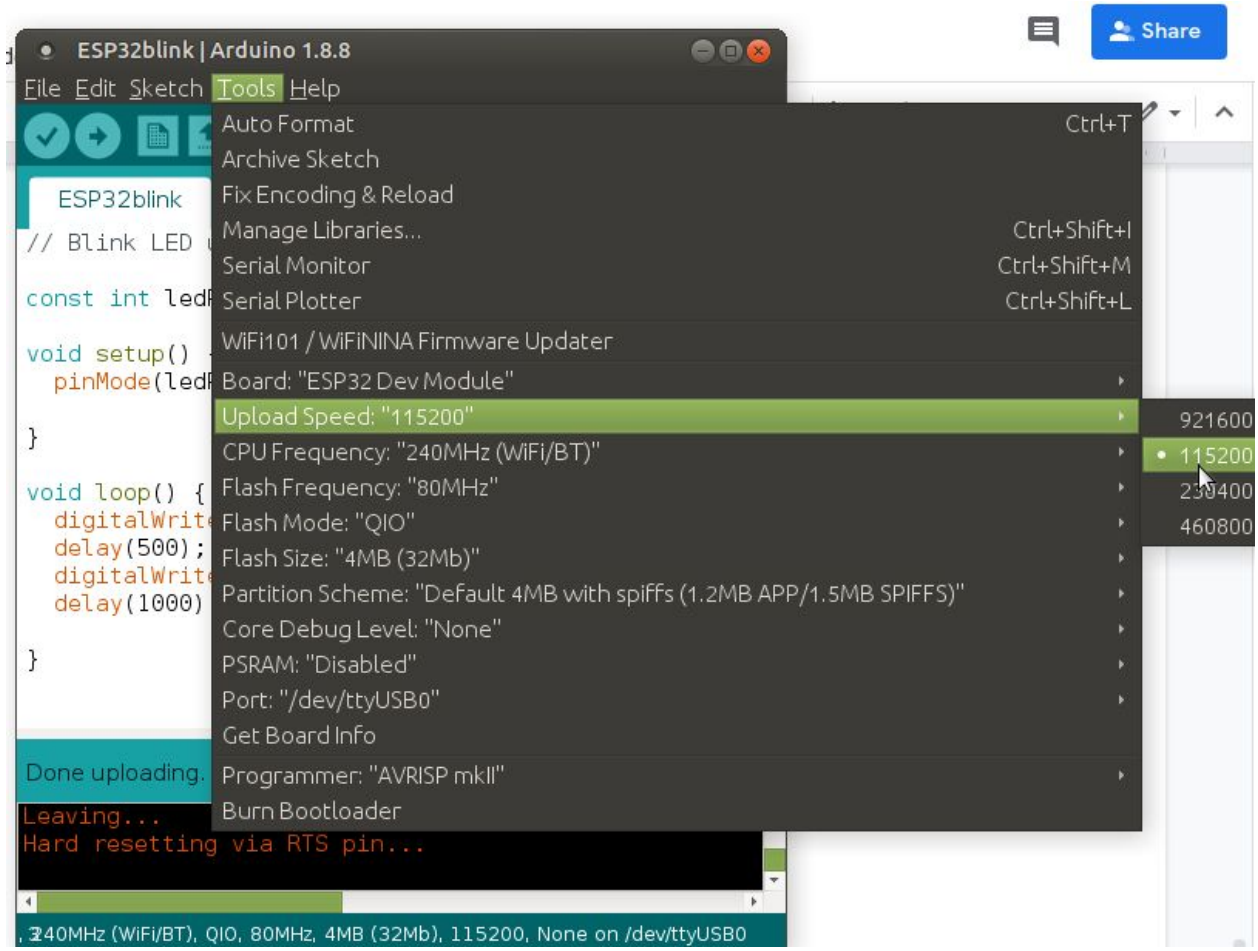


Enter the standard blink code

```
// Blink LED using ESP32

const int ledPin = 4;

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

}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin,LOW);
  delay(1000);

}
```

Make sure the port is selected such as “/dev/ttyUSB0”. Then compile and upload the code. Be patient, it takes longer than a regular Arduino Uno.
If there is a compile error about no “serial” module, then download the python 2.7 serial module, “sudo apt install python-serial”

If the upload fails, then the BOOT button and EN button need to be pressed simultaneously during upload and released.
Depending on the computer, the serial port upload speed may have to be set to 115200 instead of the default.



Example 4: ESP32 Scan Wifi

From the example programs, select the one that lists available WiFi networks. Run it and view the results in the Serial Monitor.
The *.ino file can be downloaded from GitHub

https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/examples/WiFiScan/WiFiScan.ino

In the examples menu, under ESP32 -> Wifi ->Wifiscan load this same sketch.

```
/*
 *  This sketch demonstrates how to scan WiFi networks.
 *  The API is almost the same as with the WiFi Shield library,
 *  the most obvious difference being the different file you need to include:
 */
#include "WiFi.h"
//----------------------------------
void setup()
{
    Serial.begin(115200);

    // Set WiFi to station mode and disconnect from an AP if it was previously connected
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    Serial.println("Setup done");
}
// -------------------------------
void loop()
{
    Serial.println("scan start");

    // WiFi.scanNetworks will return the number of networks found
    int n = WiFi.scanNetworks();
    Serial.println("scan done");
    if (n == 0) {
        Serial.println("no networks found");
    } else {
        Serial.print(n);
        Serial.println(" networks found");
        for (int i = 0; i < n; ++i) {
            // Print SSID and RSSI for each network found
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i));
            Serial.print(" (");
            Serial.print(WiFi.RSSI(i));
            Serial.print(")");
            Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?" ":"*");
```

```
            delay(10);
        }
    }
    Serial.println("");

    // Wait a bit before scanning again
    delay(5000);
}
```

Open the serial monitor or the tools menu. Set the baud rate to 115200 and watch the list of wifi networks repeatedly scroll by.