SMALL _ARTIFACTS LAB;

Arduino + ESP32 Instruction

Install Arduino IDE

The Arduino IDE is the primary tool we will use in the class for programming the ESP32 based microcontroller, though there are a lot of other options, such as Visual Studio Code. We will also use the Arduino programming language, which is essentially a wrapper of C/C++. You can also use MicroPython, a subset of Python that designed for to run on a microcontroller. However, we will not cover the grammar of MicroPython, and it is up to you to figure this out.

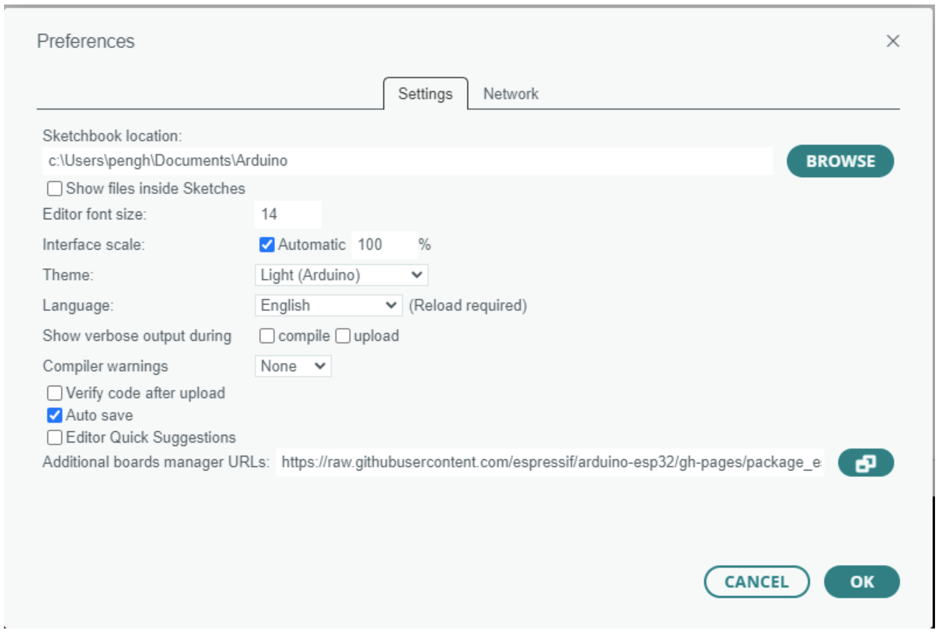
Depending on your operating system, the installation may vary. Contact TA if you encounter any issues and we can figure it out together. Below we will use Windows system as an exampler, with side notes for Mac users.

Installation steps

1. Install Arduino IDE. We will use the Arduino IDE 2.0.0 version: here is the link for Windows or MacOS.

2. Install ESP32 support. Go to File - Preference, as in Figure 1. In the box Additional boards manger URLs, paste `https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json`

Click OK.



Sketchbook location: c:\Users\pengh\Documents\Arduino
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☐ compile ☐ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_e

Figure 1

3. Click Tools - Board: - Boards Manger... (Figure 3).

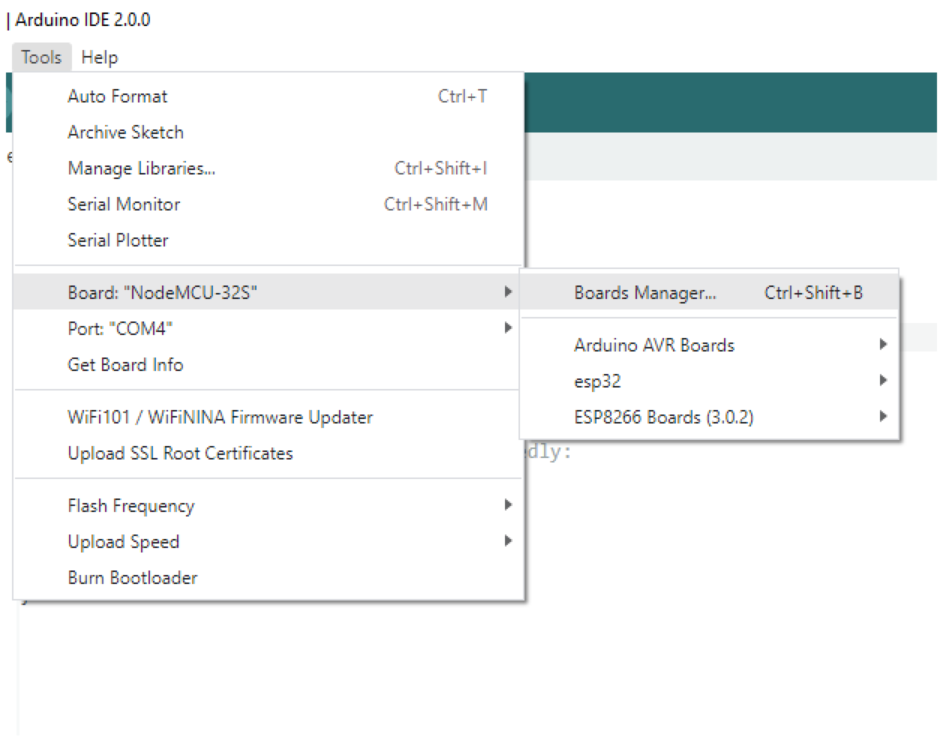
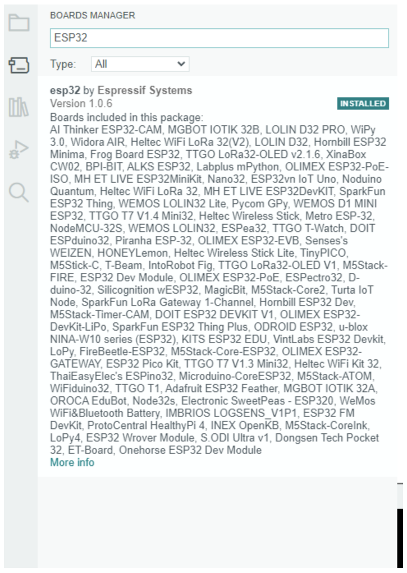


Figure 2

4. In the Boards Mangager, type ESP32 and click Install.



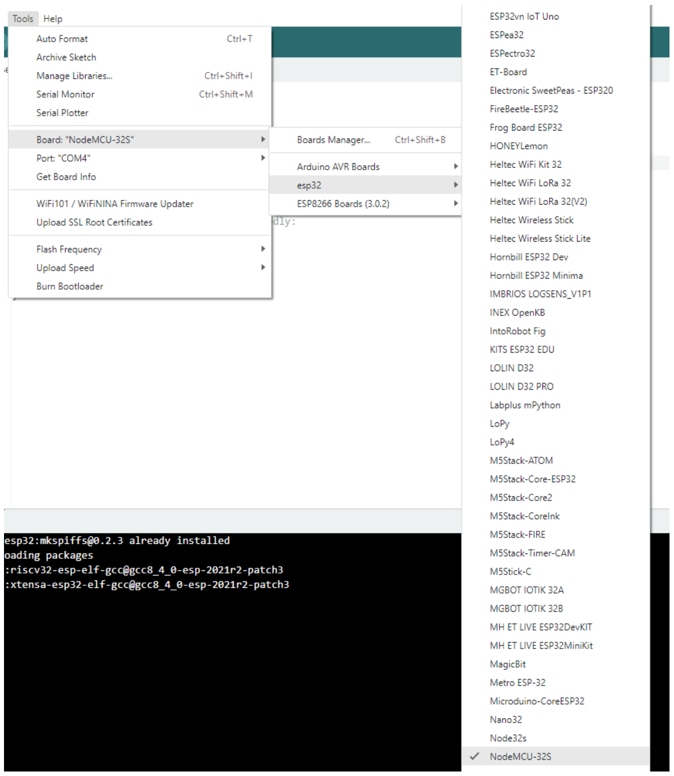
esp32 by Espressif Systems
Version 1.0.6
Boards included in this package:
AI Thinker ESP32-CAM, MGBOT IOTIK 32B, LOLIN D32 PRO, WiPy 3.0, Widora AIR, Heltec WiFi LoRa 32(V2), LOLIN D32, Hornbill ESP32 Minima, Frog Board ESP32, TTGO LoRa32-OLED v2.1.6, XinaBox CW02, BPI-BIT, ALKS ESP32, Labplus mPython, OLIMEX ESP32-PoE-ISO, MH ET LIVE ESP32MiniKit, Nano32, ESP32vn IoT Uno, Noduino Quantum, Heltec WiFi LoRa 32, MH ET LIVE ESP32DevKIT, SparkFun ESP32 Thing, WEMOS LOLIN32 Lite, Pycom GPy, WEMOS D1 MINI ESP32, TTGO T7 V1.4 Mini32, Heltec Wireless Stick, Metro ESP-32, NodeMCU-32S, WEMOS LOLIN32, ESPea32, TTGO T-Watch, DOIT ESPduino32, Piranha ESP-32, OLIMEX ESP32-EVB, Senses's WEIZEN, HONEYLemon, Heltec Wireless Stick Lite, TinyPICO, M5Stick-C, T-Beam, IntoRobot Fig, TTGO LoRa32-OLED V1, M5Stack-FIRE, ESP32 Dev Module, OLIMEX ESP32-PoE, ESPectro32, D-duino-32, Silicognition wESP32, MagicBit, M5Stack-Core2, Turta IoT Node, SparkFun LoRa Gateway 1-Channel, Hornbill ESP32 Dev, M5Stack-Timer-CAM, DOIT ESP32 DEVKIT V1, OLIMEX ESP32-DevKit-LiPo, SparkFun ESP32 Thing Plus, ODROID ESP32, u-blox NINA-W10 series (ESP32), KITS ESP32 EDU, VintLabs ESP32 Devkit, LoPy, FireBeetle-ESP32, M5Stack-Core-ESP32, OLIMEX ESP32-GATEWAY, ESP32 Pico Kit, TTGO T7 V1.3 Mini32, Heltec WiFi Kit 32, ThaiEasyElec's ESPino32, Microduino-CoreESP32, M5Stack-ATOM, WiFiduino32, TTGO T1, Adafruit ESP32 Feather, MGBOT IOTIK 32A, OROCA EduBot, Node32s, Electronic SweetPeas - ESP320, WeMos WiFi&Bluetooth Battery, IMBRIOS LOGSENS_V1P1, ESP32 FM DevKit, ProtoCentral HealthyPi 4, INEX OpenKB, M5Stack-CoreInk, LoPy4, ESP32 Wrover Module, S.ODI Ultra v1, Dongsen Tech Pocket 32, ET-Board, Onehorse ESP32 Dev Module

Figure 3

Verify and Upload your Code

Let’s test if the installation is successful.

1. Plug the ESP32 board to your computer. In the IDE, Select Tools - Board - NodeMCU-32S (Figure4)



```
esp32:mkspiffs@0.2.3 already installed
oading packages
:riscv32-esp-elf-gcc@gcc8_4_0-esp-2021r2-patch3
:xtensa-esp32-elf-gcc@gcc8_4_0-esp-2021r2-patch3
```

Figure 4

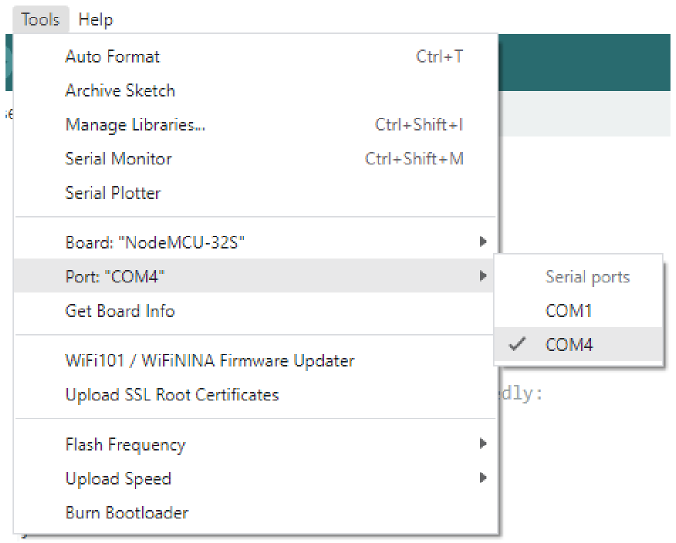
2. Copy and Paste the following code sketch to the canvas.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

3. Select the Port that talks to your ESP32. In my case, it's COM4 (Figure 5). On Mac, it is something similar to '/dev/tty.usbserial-A6004byf 9600'. Yours may be different. If you don't see the correct port in your Arduino IDE, you may need to install the UART driver: CP210x USB to UART Bridge VCP Drivers.



Tools Help

Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Board: "NodeMCU-32S"
Port: "COM4"
Get Board Info
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Flash Frequency
Upload Speed
Burn Bootloader

Serial ports
COM1
✓ COM4

Figure 5

4. Upload the code to ESP32. Press the Upload button in the Arduino IDE. For our ESP board, you need to hold-down the “BOOT” button in your board until you see the code is uploading, like what is showed in Figure 7. The ”BOOT” button is to the right of the micro USB cable of your board. See Figure 6.

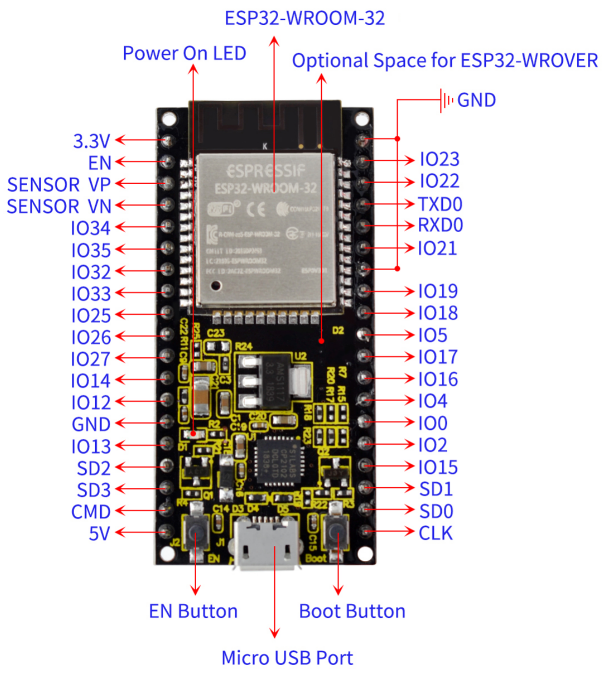
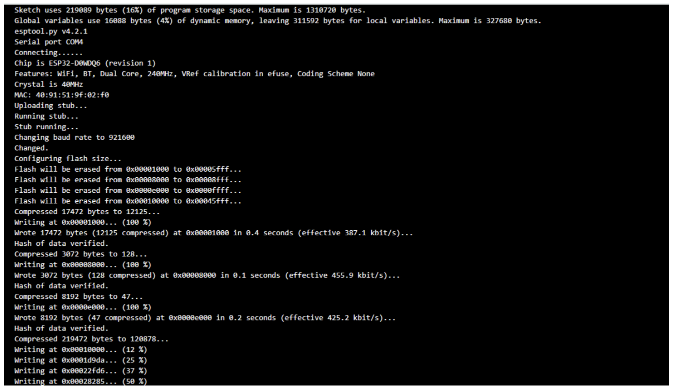


Figure 6



```
Sketch uses 219089 bytes (16%) of program storage space. Maximum is 1310720 bytes.
Global variables use 16088 bytes (4%) of dynamic memory, leaving 311592 bytes for local variables. Maximum is 327680 bytes.
esptool.py v4.2.1
Serial port COM4
Connecting......
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 40:91:51:9f:02:f0
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x00045fff...
Compressed 17472 bytes to 12125...
Writing at 0x00001000... (100 %)
Wrote 17472 bytes (12125 compressed) at 0x00001000 in 0.4 seconds (effective 387.1 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.1 seconds (effective 455.9 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.2 seconds (effective 425.2 kbit/s)...
Hash of data verified.
Compressed 219472 bytes to 120878...
Writing at 0x00010000... (12 %)
Writing at 0x0001d9da... (25 %)
Writing at 0x00022fd6... (37 %)
Writing at 0x00028285... (50 %)
```

Figure 7

5. If everything works out, you should see a blue LED flash at 1Hz on your board.

Last Modified Date: Sep 25th, 2022