IoT-ALE: Demystifying MCUs with Arduino

Nova (aka Justin) King

SCaLE 17x - March 2019

FIGURE 1. The Sunbeam Radiant Control toaster.

Arduino

- ATMEGA328
- Key parameters [edit]

Parameter	Value
CPU type	8-bit AVR
Performance	20 MIPS at 20 MHz[2]
Flash memory	32 kB
SRAM	2 kB
EEPROM	1 kB
Pin count	28 or 32 pin: PDIP-28, MLF-28, TQFP-32, MLF-32[2]
Maximum operating frequency	20 MHz
Number of touch channels	16
Hardware QTouch Acquisition	No
Maximum I/O pins	23
External interrupts	2
USB Interface	No
USB Speed	–

ESP8266

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz[5]
- Memory:
    - 32 KiB instruction RAM
    - 32 KiB instruction cache RAM
    - 80 KiB user-data RAM
    - 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
    - Integrated TR switch, balun, LNA, power amplifier and matching network
    - WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI
- I²C (software implementation)[6]
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

ESP32

- Processors:
  - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
  - Ultra low power (ULP) co-processor
- Memory: 520 KiB SRAM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE
- Peripheral interfaces:
  - 12-bit SAR ADC up to 18 channels
  - 2 × 8-bit DACs
  - 10 × touch sensors (capacitive sensing GPIOs)
  - Temperature sensor
  - 4 × SPI
  - 2 × I²S interfaces
  - 2 × I²C interfaces
  - 3 × UART
  -

ESP32 Con’t

- SD/SDIO/CE-ATA/MMC/eMMC host controller
- SDIO/SPI slave controller
- Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time Protocol support
- CAN bus 2.0
- Infrared remote controller (TX/RX, up to 8 channels)
- Motor PWM
- LED PWM (up to 16 channels)
- Hall effect sensor
- Ultra low power analog pre-amplifier
-

ESP32 Con’t

- security:
  - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
  - Secure boot
  - Flash encryption
  - 1024-bit OTP, up to 768-bit for customers
  - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)
- Power management:
  - Internal low-dropout regulator
  - Individual power domain for RTC
  - 5uA deep sleep current
  - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

Your devices and networking

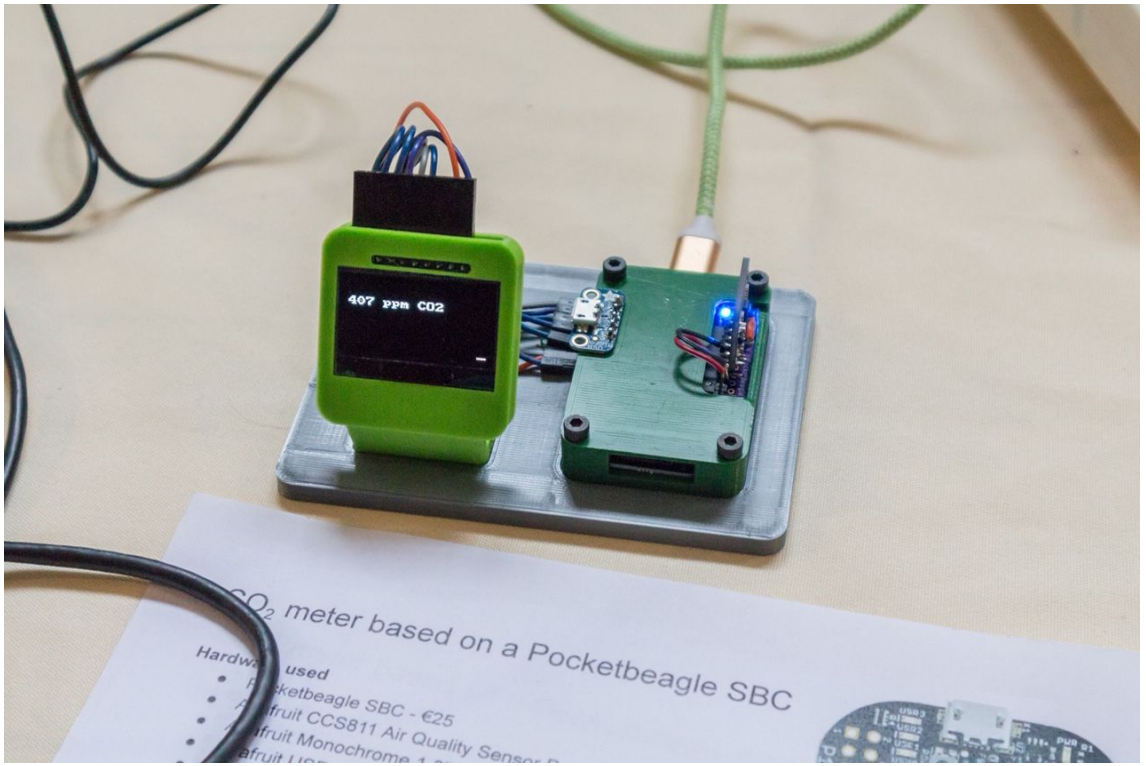
Hybrid solution: Local access + cloud access

Your devices and networking

Cloud only access, no local network

Your devices and networking

No access because you forgot to install the wifi firmware :)



Labs

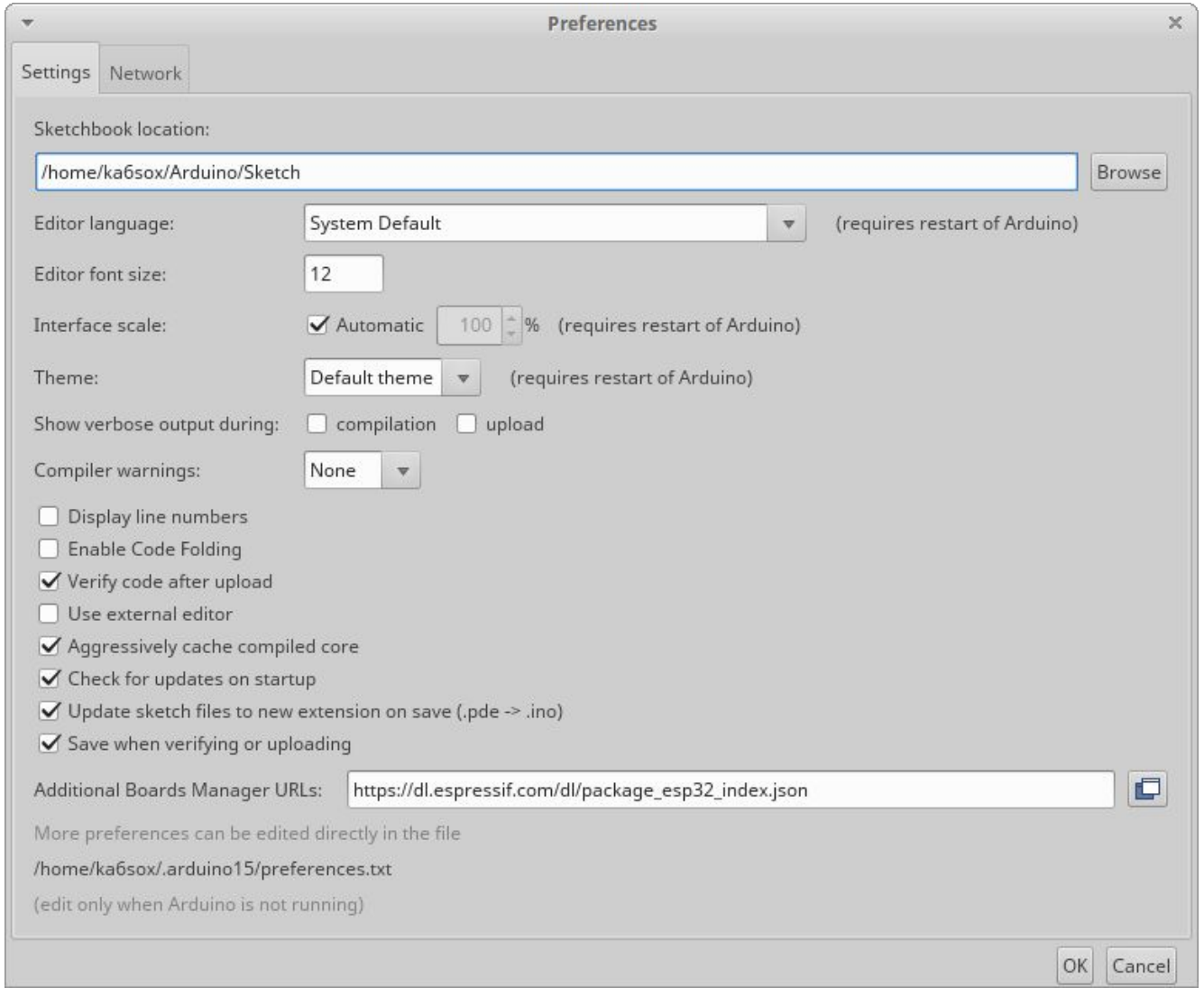
- IDE/Board Setup
  - Install Python if needed
  - Install Arduino IDE
  - Install ESP32 board interface
- Blinky
  - Open and upload to board
- WiFi
  - Open from examples menu
  - Upload
- Sensors
  - Install library from library manager
  - Open example
  - Modify example to work with the current board

IDE Setup

- https://www.arduino.cc/en/Guide/Linux
    - `sudo chmod 666 /dev/ttyUSB0` if it won't upload
- https://www.arduino.cc/en/Guide/Windows
- https://www.arduino.cc/en/Guide/MacOSX

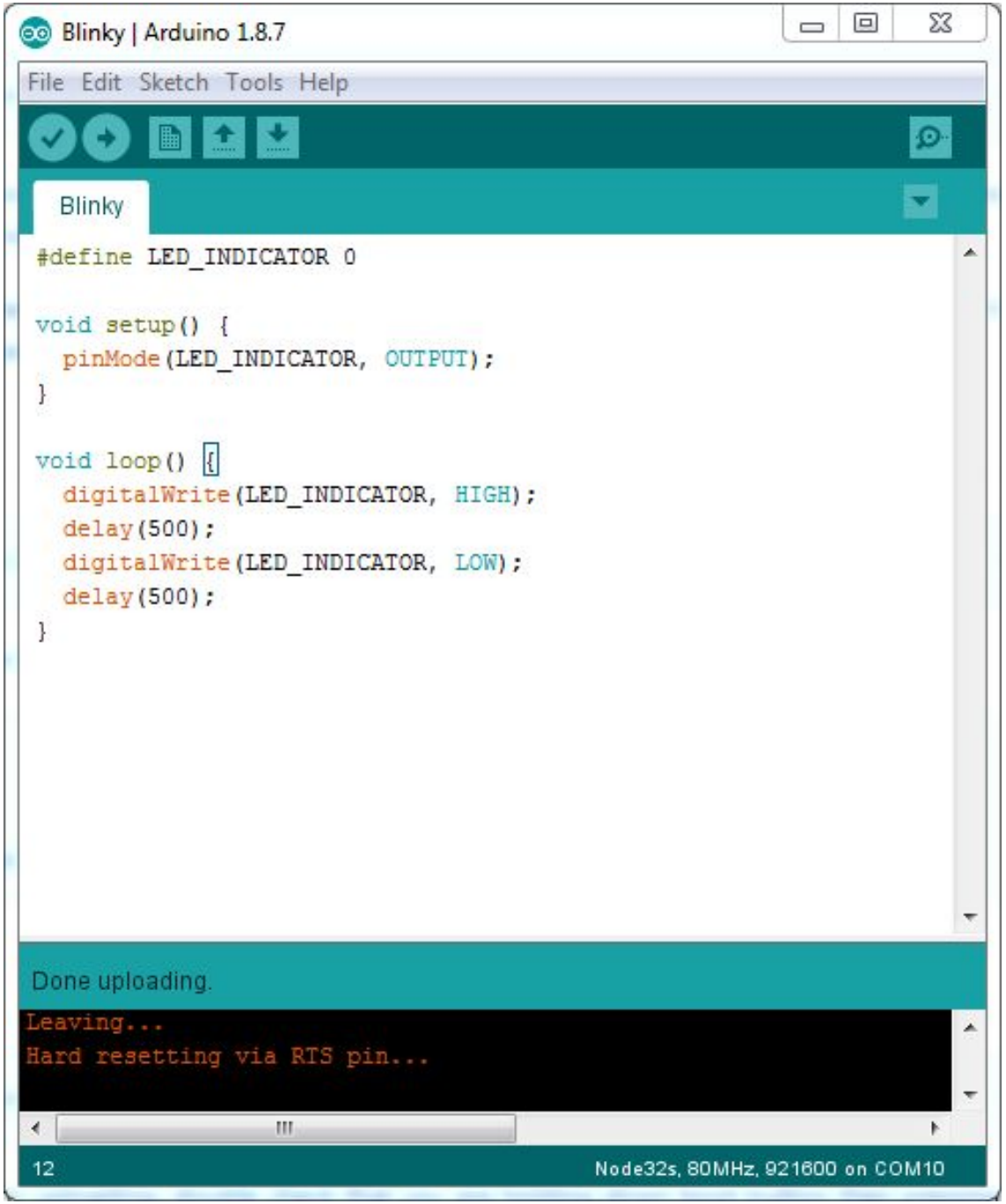
Setting up the ESP32 board drivers

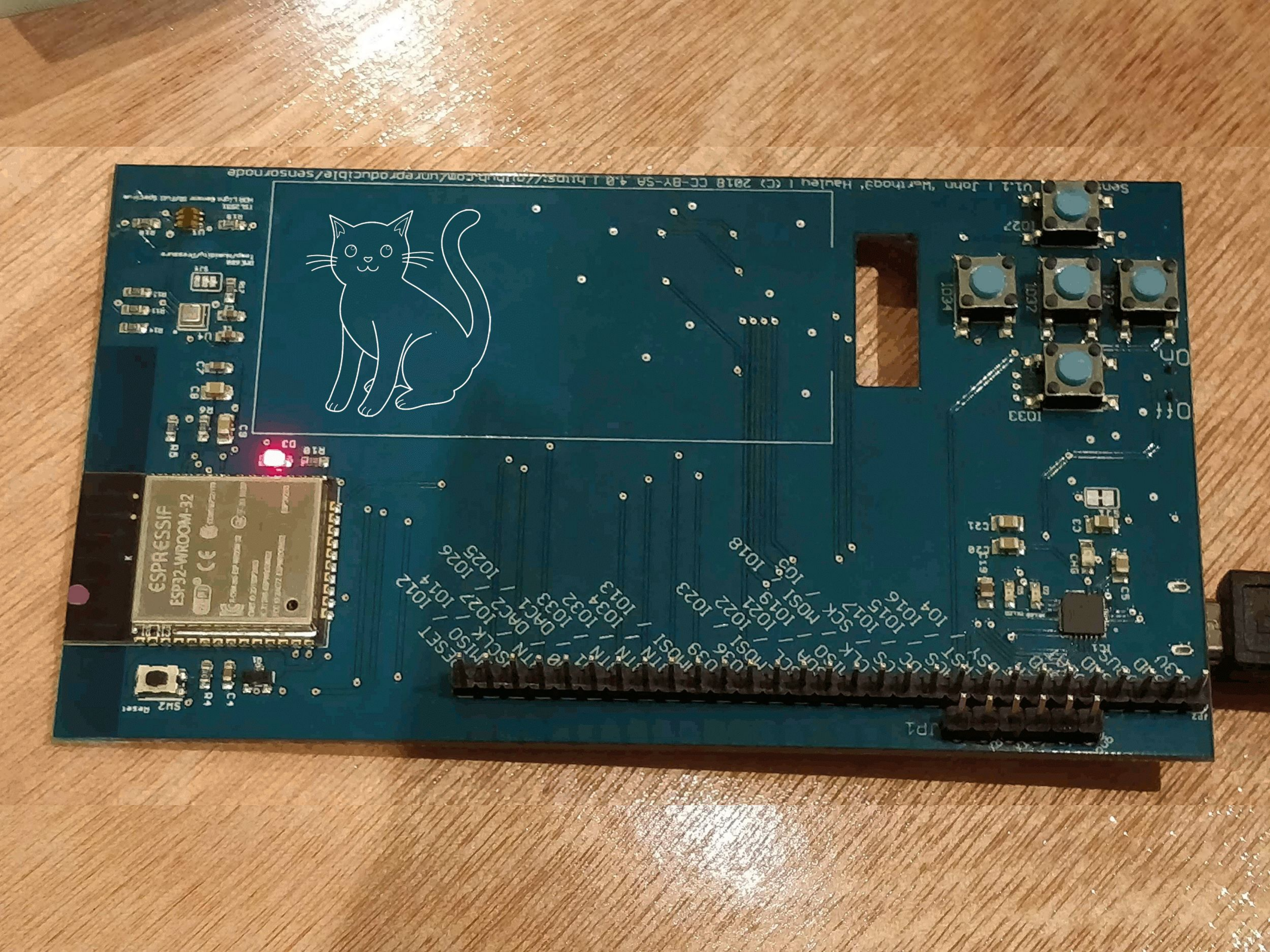
(https://dl.espressif.com/dl/package_esp32_index.json)



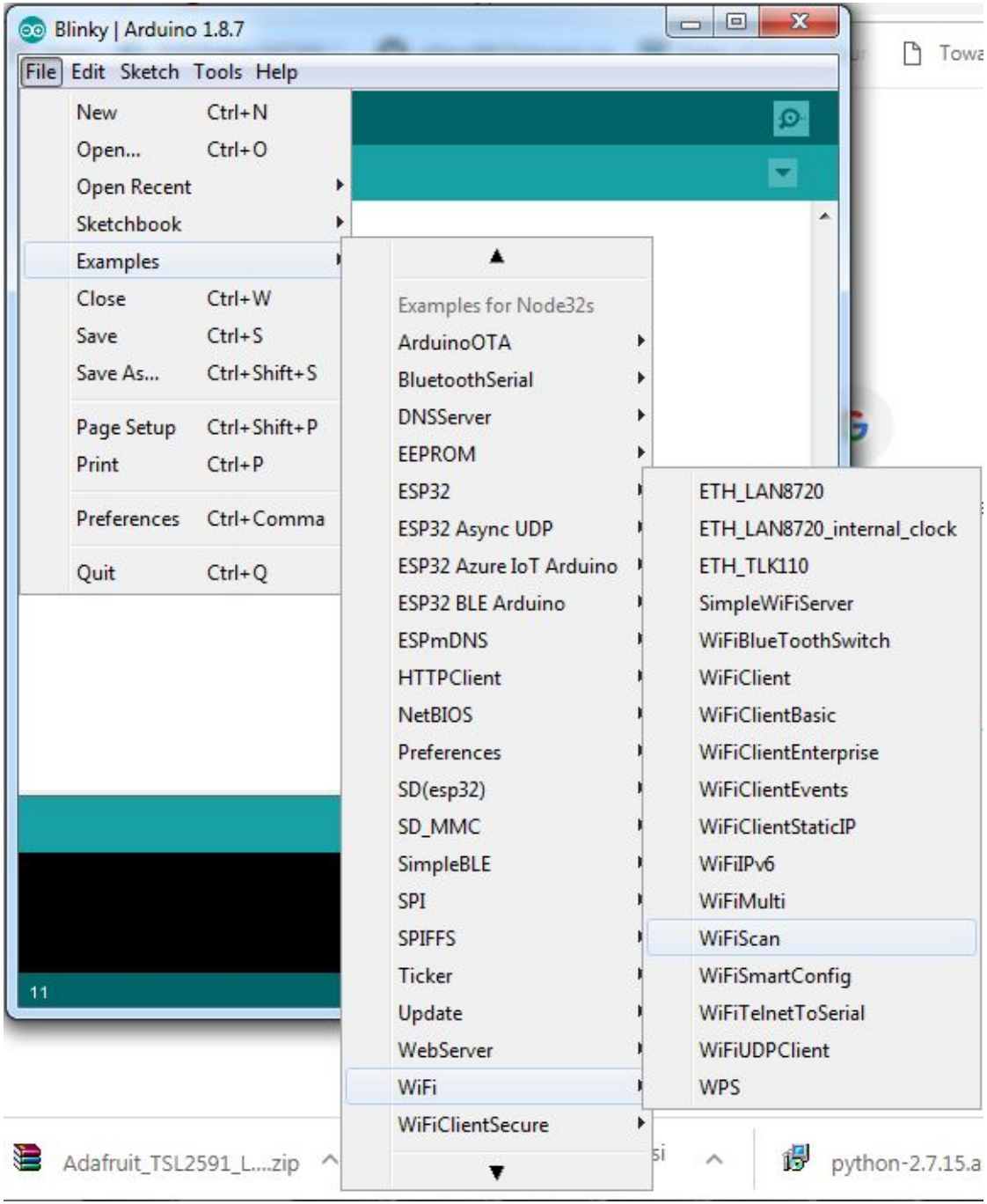
Blinky

- File > Open
- Open the Blinky file in the Blinky folder
- Upload the program to the board
- `sudo chmod 666 /dev/ttyUSB0`
  if it won't upload

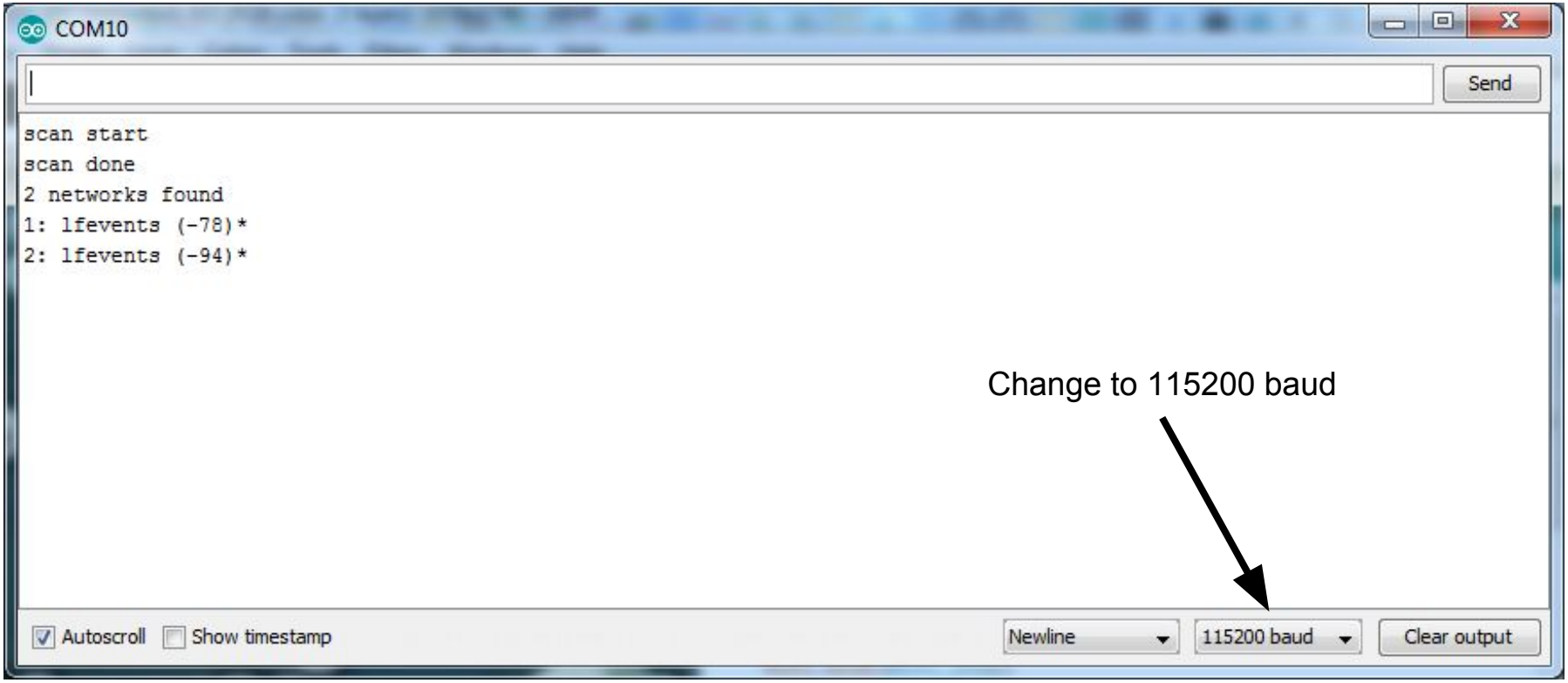

Blinky | Arduino 1.8.7
File Edit Sketch Tools Help
Blinky
#define LED_INDICATOR 0
void setup() {
pinMode(LED_INDICATOR, OUTPUT);
}
void loop() {
digitalWrite(LED_INDICATOR, HIGH);
delay(500);
digitalWrite(LED_INDICATOR, LOW);
delay(500);
}
Done uploading.
Leaving...
Hard resetting via RTS pin...
12
Node32s, 80MHz, 921600 on COM10


ESPRESSIF
ESP32-WROOM-32
IO25
IO26
IO18
IO5
IO12
IO14
IO23
IO16
IO4
JP1

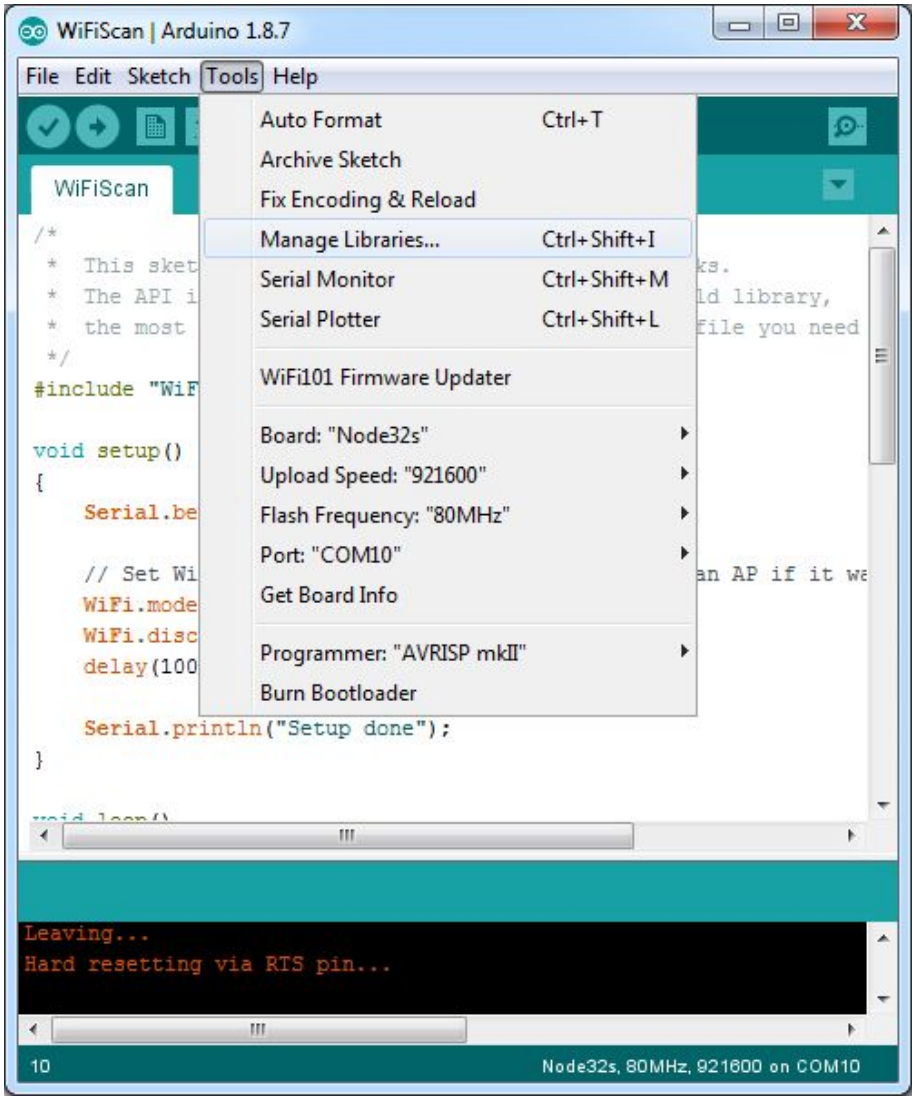
Wifi Scan


Blinky | Arduino 1.8.7
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Examples for Node32s
ArduinoOTA
BluetoothSerial
DNSServer
EEPROM
ESP32
ESP32 Async UDP
ESP32 Azure IoT Arduino
ESP32 BLE Arduino
ESPmDNS
HTTPClient
NetBIOS
Preferences
SD(esp32)
SD_MMC
SimpleBLE
SPI
SPIFFS
Ticker
Update
WebServer
WiFi
WiFiClientSecure
ETH_LAN8720
ETH_LAN8720_internal_clock
ETH_TLK110
SimpleWiFiServer
WiFiBlueToothSwitch
WiFiClient
WiFiClientBasic
WiFiClientEnterprise
WiFiClientEvents
WiFiClientStaticIP
WiFiIPv6
WiFiMulti
WiFiScan
WiFiSmartConfig
WiFiTelnetToSerial
WiFiUDPClient
WPS
11
Adafruit_TSL2591_L....zip
python-2.7.15.a

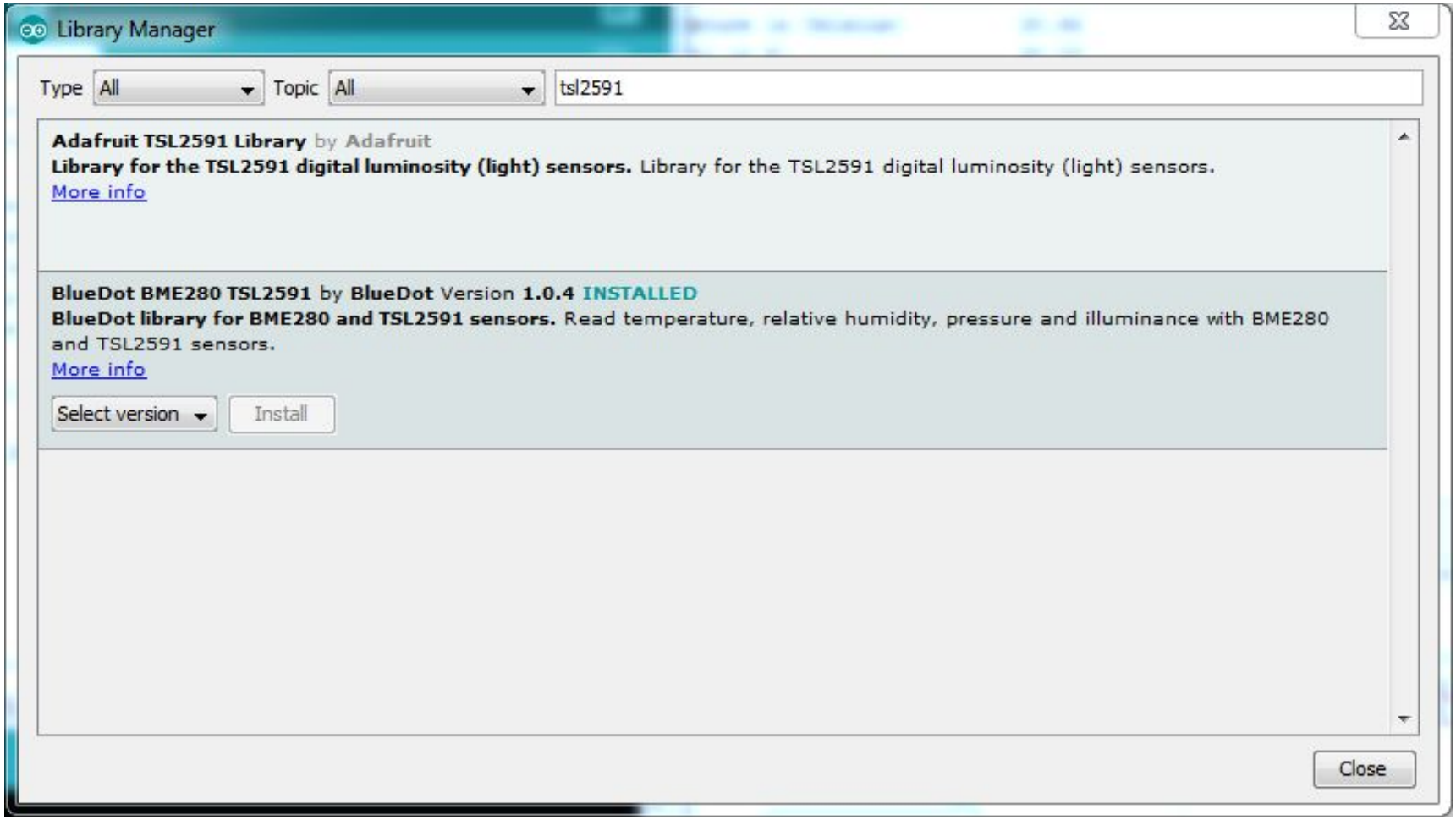
Wifi Monitor (Tools > Serial Monitor)



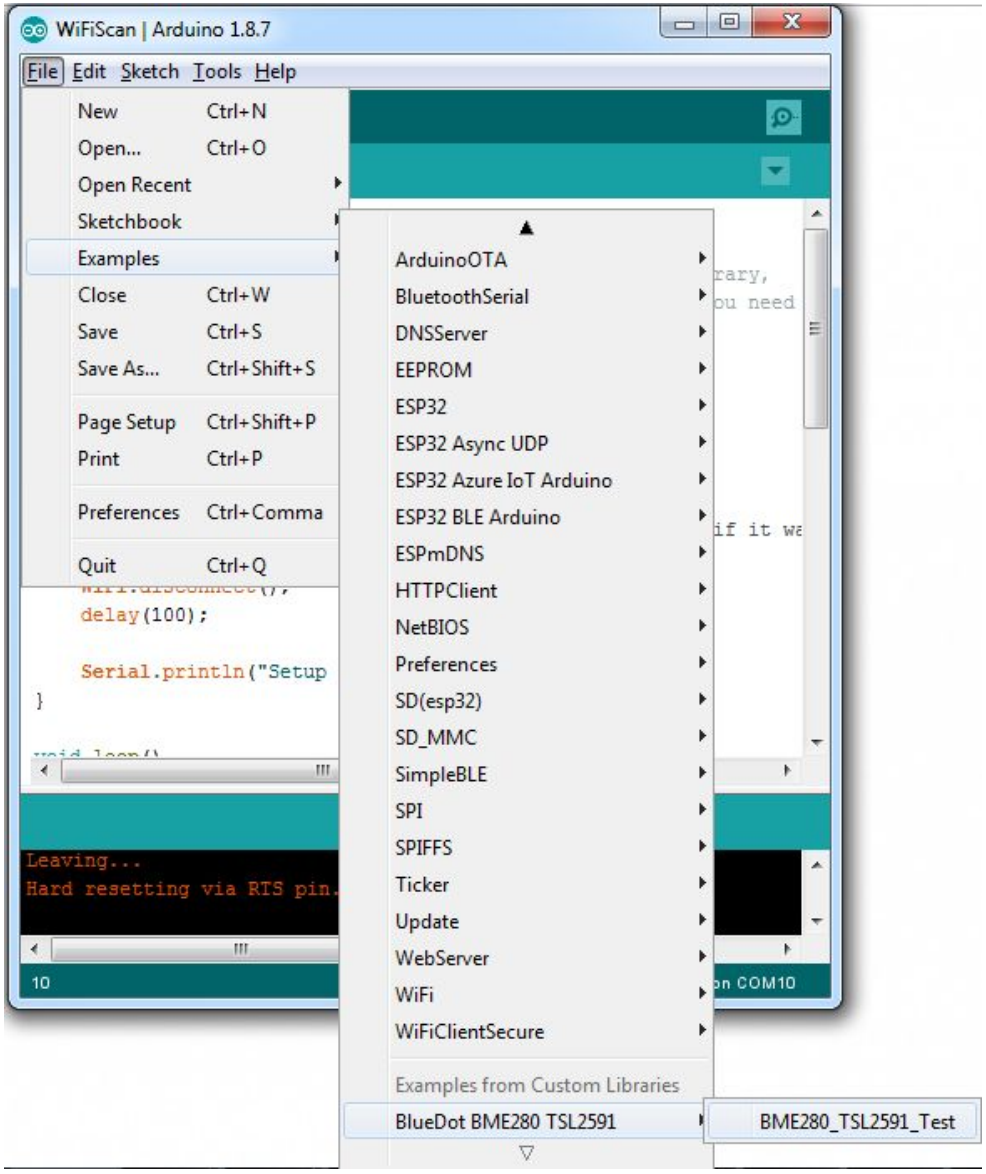
Library Manager (for sensors)



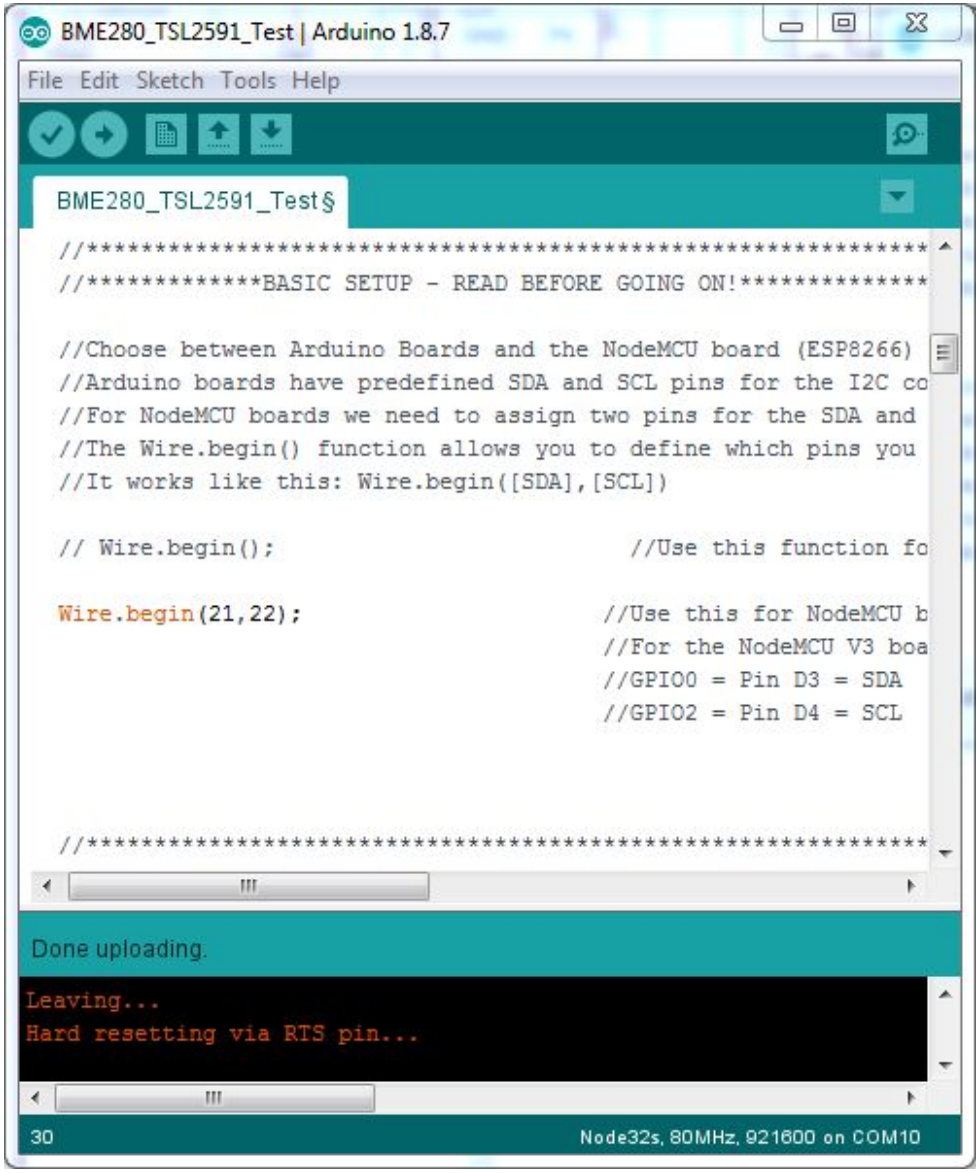
Sensors Library Installation



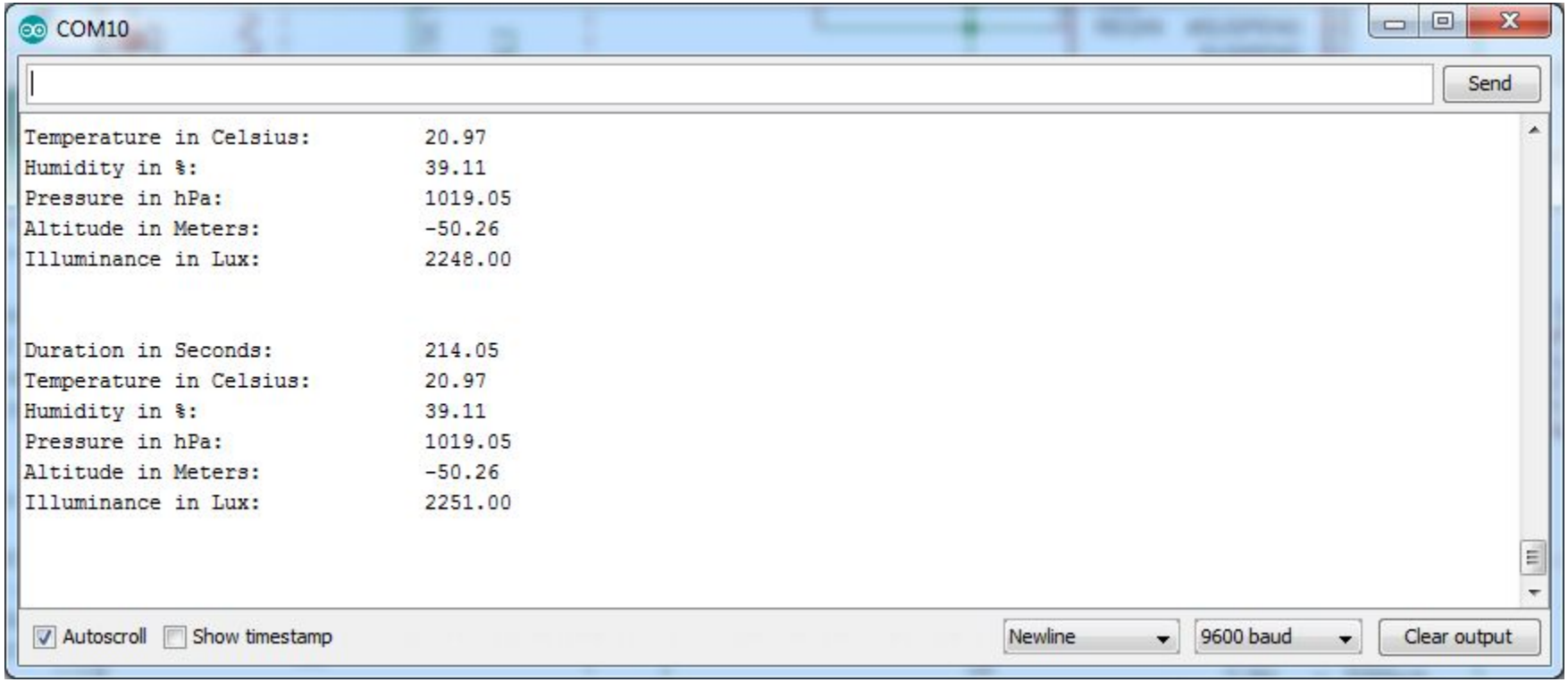
Sensors Library Example



Modifying it to Work



Expected Output



Examples I Used:

github.com/chromenova/sensornodeexamples