AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery ESP32 Lolin32*. On the following pages, you will be introduced to how to use and set up this handy device.

Have fun!

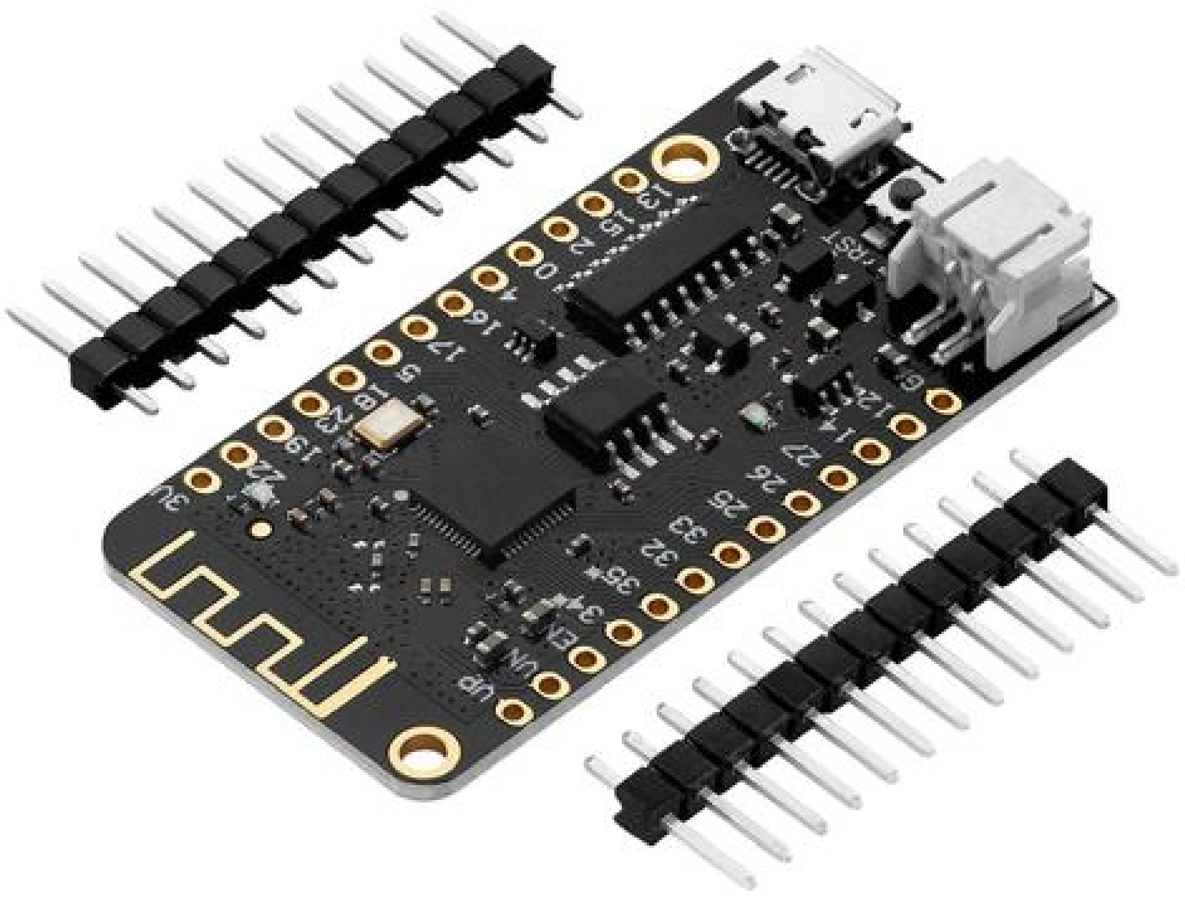


Table of Contents

Introduction

Lolin32 is the ESP32 development board. Like other ESP32 boards, it also has a 32-bit dual-core microcontroller which operates at 240MHz, has 4MB of flash storage. It can be programmed in a range of languages, such as microPython, LUA, mod. C/C++ in Arduino IDE, just like any other microcontroller. It can be powered via lithium polymer batteries with a 500mA max charging current or via micro USB. The firmware that comes with this board accepts AT commands by default. It is equipped with WiFi and Bluetooth. In addition to that, it includes an FTDI FT231x, which converts USB to serial and enables a computer to program and communicate with the microcontroller.

This module does not use the WROOM32 module with built-in flash memory but has the ESP32 chip, the 4MByte flash memory, and the antenna built directly on the board. Thus the width could be reduced by one grid dimension. So this module can be used directly on a simple breadboard.

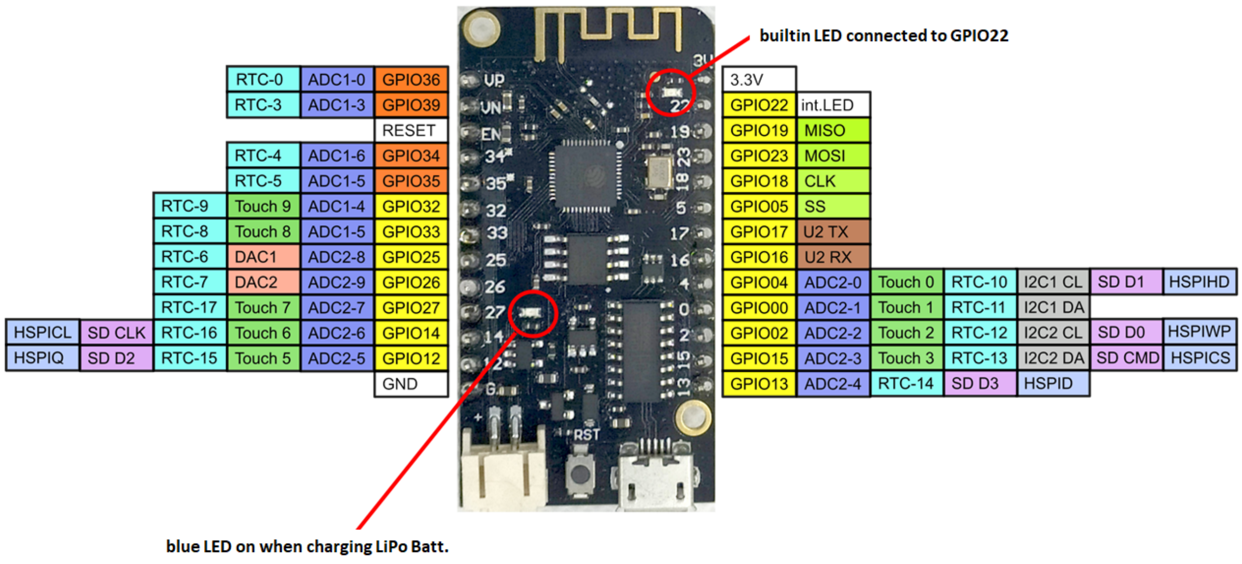
Another advantage of this module is the built-in charge controller for 3.7V LiPo batteries. The module has a JST 2.0 mm socket for connecting a battery. However, the module also has disadvantages. There is no 5V connection and the power supply can only be done via USB or a battery. There are also fewer connections out. There are 19 inputs/outputs and 4 inputs. A LED built on the module can be controlled by GPIO22.

Specifications

Power supply	2.2V to 3.6V
Operating voltage	3.3V
Logic level	3.3V
Mounting holes diameter	2 holes, 3mm diameter
Integrated 802.11 BGN WiFi transceiver	
Integrated dual-mode Bluetooth	
4MB Flash Memory	
Maximum serial baud rate	256000bps
Dimensions	50x25mm (1.97x0.98in)

The pinout

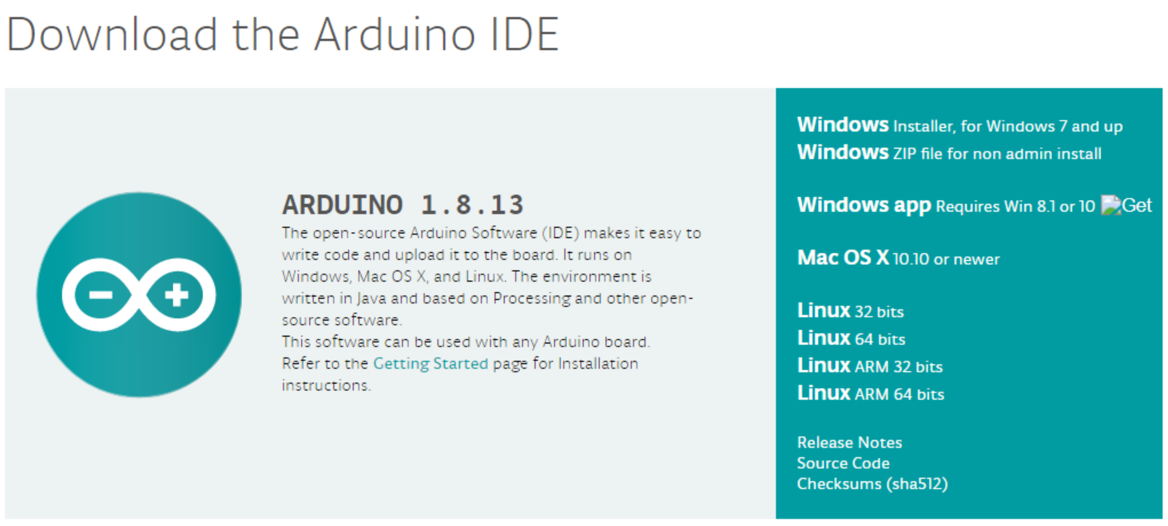
The *ESP32 Lolin32* has 26 pins. The pinout is shown in the following image:



builtin LED connected to GPIO22
RTC-0 ADC1-0 GPIO36
RTC-3 ADC1-3 GPIO39
RESET
RTC-4 ADC1-6 GPIO34
RTC-5 ADC1-5 GPIO35
RTC-9 Touch 9 ADC1-4 GPIO32
RTC-8 Touch 8 ADC1-5 GPIO33
RTC-6 DAC1 ADC2-8 GPIO25
RTC-7 DAC2 ADC2-9 GPIO26
RTC-17 Touch 7 ADC2-7 GPIO27
HSPICL SD CLK RTC-16 Touch 6 ADC2-6 GPIO14
HSPIQ SD D2 RTC-15 Touch 5 ADC2-5 GPIO12
GND
3.3V
GPIO22 int.LED
GPIO19 MISO
GPIO23 MOSI
GPIO18 CLK
GPIO05 SS
GPIO17 U2 TX
GPIO16 U2 RX
GPIO04 ADC2-0 Touch 0 RTC-10 I2C1 CL SD D1 HSPIHD
GPIO00 ADC2-1 Touch 1 RTC-11 I2C1 DA
GPIO02 ADC2-2 Touch 2 RTC-12 I2C2 CL SD D0 HSPIWP
GPIO15 ADC2-3 Touch 3 RTC-13 I2C2 DA SD CMD HSPICS
GPIO13 ADC2-4 RTC-14 SD D3 HSPID
blue LED on when charging LiPo Batt.

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 **1.8.13.**



Download the Arduino IDE

ARDUINO 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows 7 and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10 Get

Mac OS X 10.10 or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

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`.

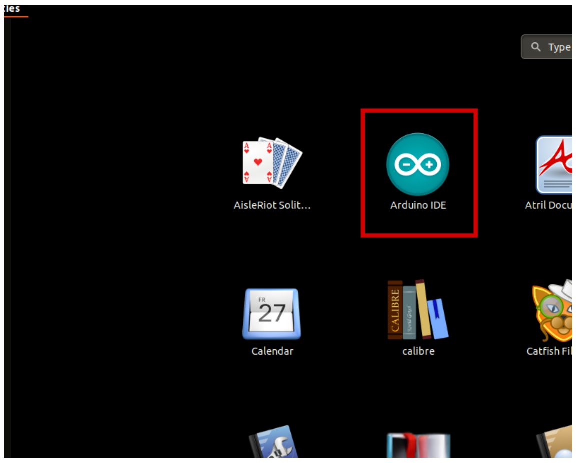
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
sh arduino-linux-setup.sh user_name
```

user_name - is the name of a superuser in 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.

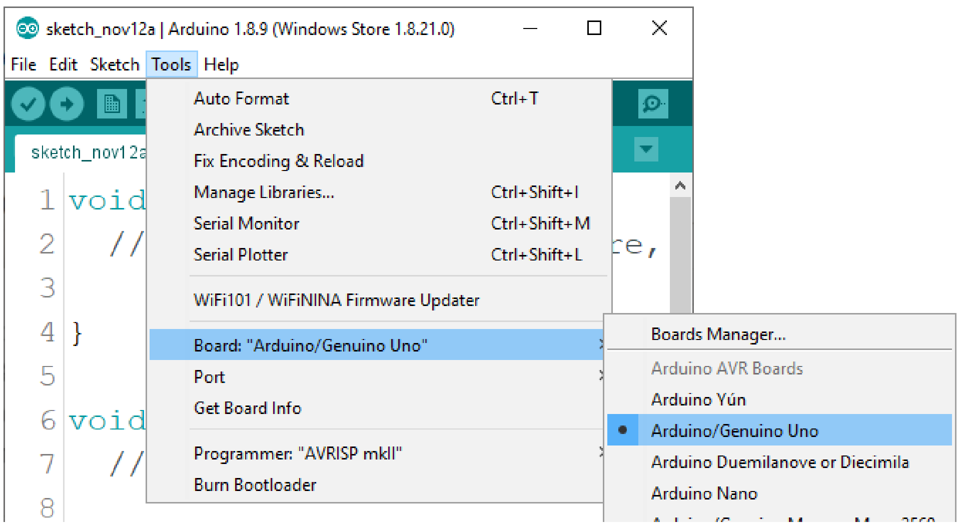


Almost all operating systems come with a text editor preinstalled (for example, `Windows` comes with `Notepad`, `Linux Ubuntu` comes with `Gedit`, `Linux Raspbian` comes with `Leafpad`, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Atmega328p board. Open freshly installed Arduino IDE, and go to:

Tools > Board > {your board name here}

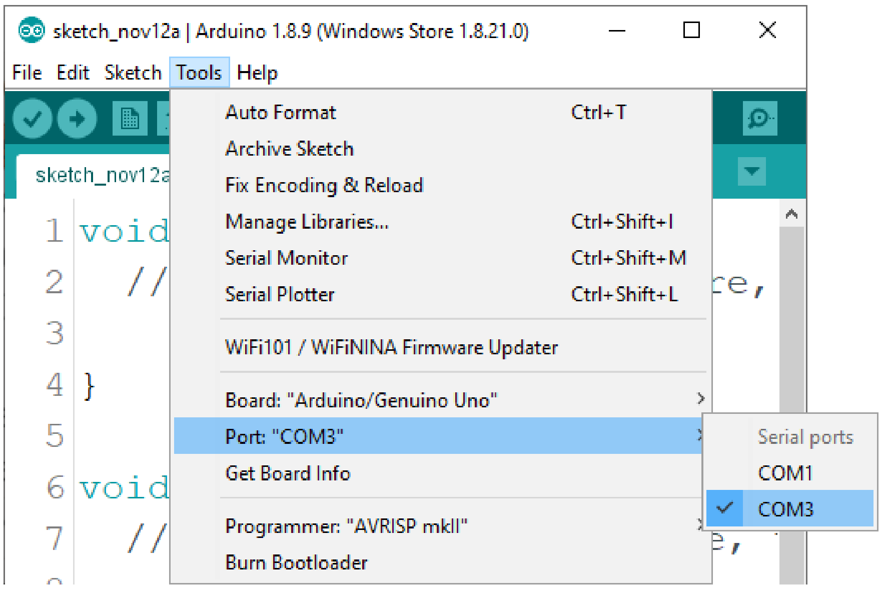
{your board name here} should be the `Arduino/Genuino Uno`, as it can be seen on the following image:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
● Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano

The port to which the Atmega328p board is connected has to be selected. Go to: *Tools > Port > {port name goes here}*

and when the Atmega328p 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:

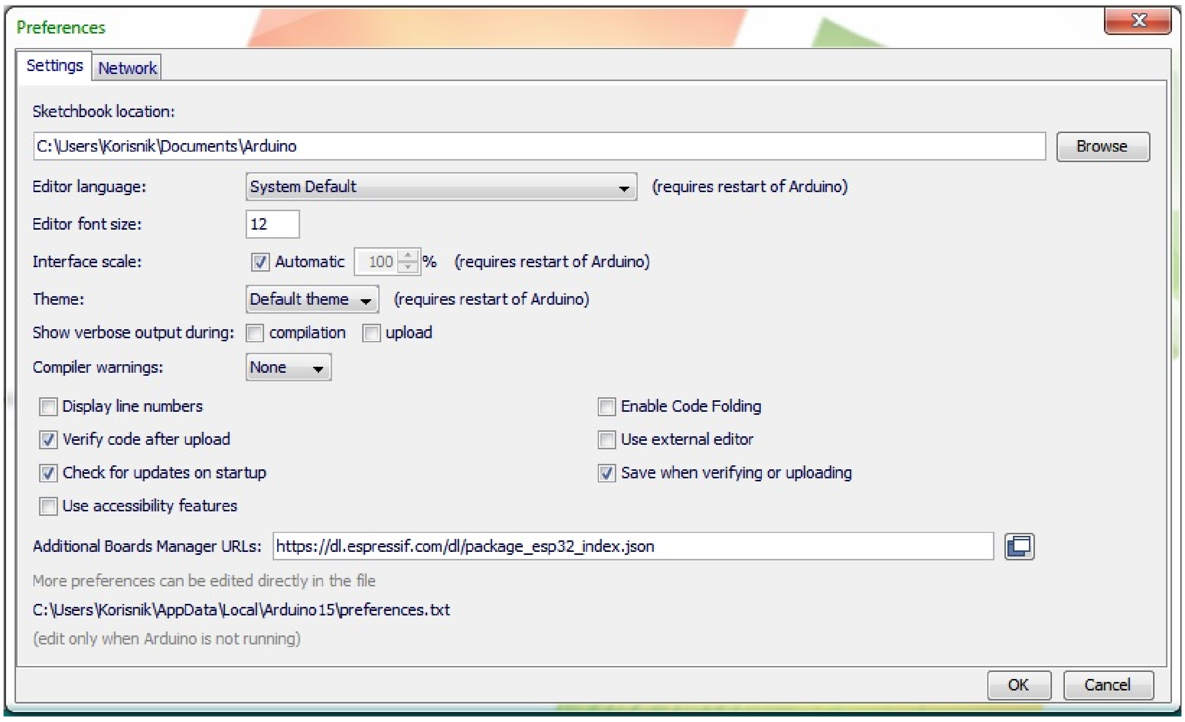


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3

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

Installing the ESP32 Board in Arduino IDE

To install ESP32 add-on in Arduino IDE first open Arduino IDE, than go to *File > Preferences*:



Preferences

Settings | Network

Sketchbook location:
C:\Users\Korisnik\Documents\Arduino — Browse

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers
☐ Enable Code Folding
✓ Verify code after upload
☐ Use external editor
✓ Check for updates on startup
✓ Save when verifying or uploading
☐ Use accessibility features

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

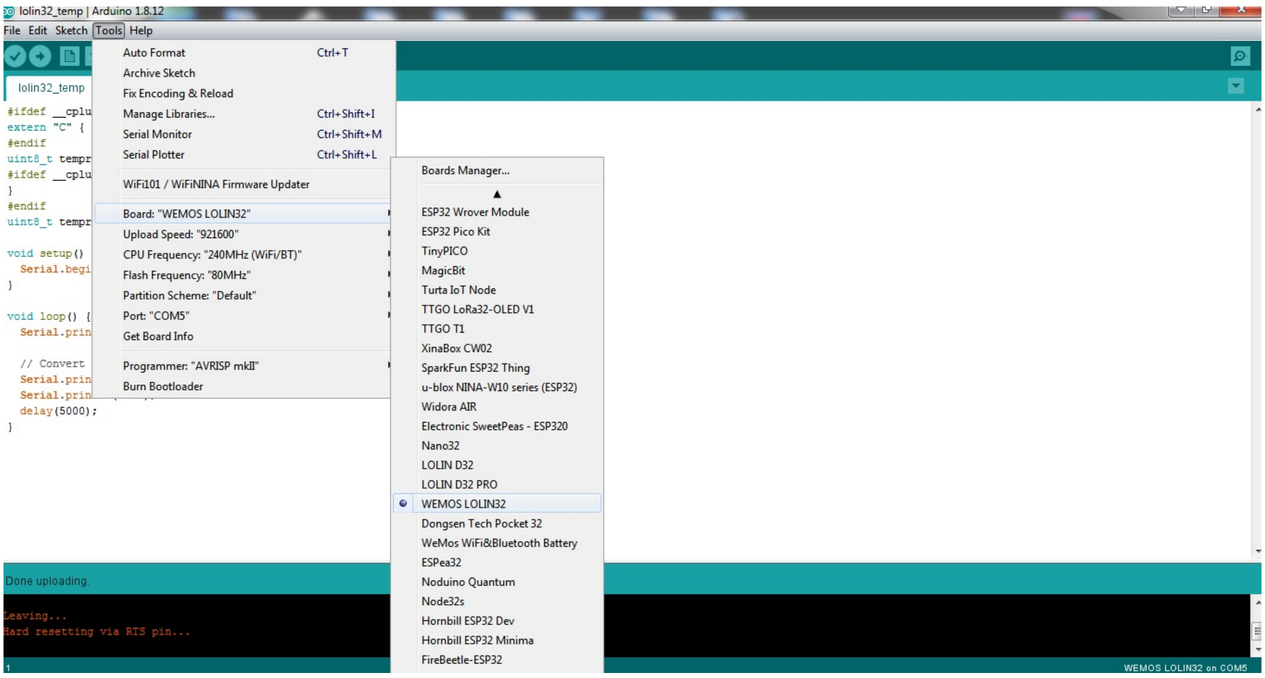
More preferences can be edited directly in the file
C:\Users\Korisnik\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

OK | Cancel

In *Additional Boards Manager URLs* box enter the next URL: *https://dl.expressif.com/dl/package_esp32_index.json*

Than go to *Tools > Board > Boards Manager* and search for ESP32 and press the install button for *esp32 by Espressif Systems*. After a few seconds the ESP32 is installed.

Now, the board can be selected in the boards menu:



```
#ifdef __cplu
extern "C" {
#endif
uint8_t tempr
#ifdef __cplu
}
#endif
uint8_t tempr

void setup()
  Serial.begi
}

void loop() {
  Serial.prin

  // Convert
  Serial.prin
  Serial.prin
  delay(5000);
}
```

Board: "WEMOS LOLIN32"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Partition Scheme: "Default"
Port: "COM5"
Programmer: "AVRISP mkII"

```
Done uploading.

Leaving...
Hard resetting via RTS pin...
```

WEMOS LOLIN32 board is used in this project.

Now, the ESP32 board is ready for code to be uploaded.

Sketch examples

The ESP32 Lolin32 needs a micro USB cable to allow a computer to program and communicate with the microcontroller.

Example with DS18B20

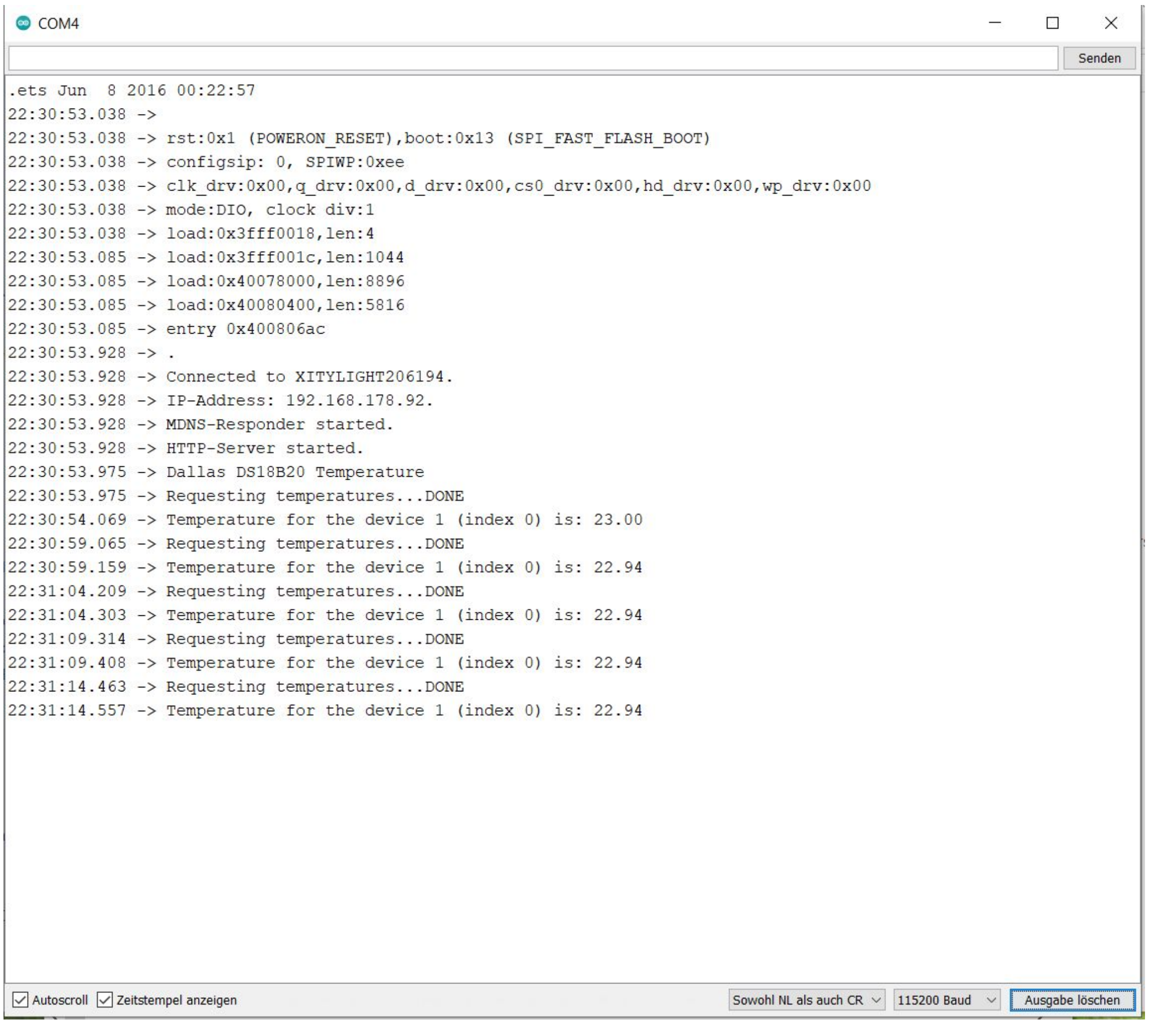
```
// Measure the temperature of your pool or pond with DS18B20
// and report it in your local WiFi

// Server
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
const char* SSID = "  your SSID   ";
const char* PASSWORD = "  your password   ";
WebServer server(80);

//Sensor
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into port 22 on the Lolin32, remember 4k7 pull-up
resistor
#define ONE_WIRE_BUS 22

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);
```

```
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

void setup() {

// Server
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID, PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.printf("Connected to %s.\n", SSID);
  Serial.printf("IP-Address: %s.\n",  WiFi.localIP().toString().c_str());

  if (MDNS.begin("Lolin32")) {
    Serial.println("MDNS-Responder started.");
  }

  server.on("/", handleRoot);

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP-Server started.");
```

```
// Sensor
  Serial.println("Dallas DS18B20 Temperature ");

  // Start up the library
  sensors.begin();
}

void loop() {
// Sensor

  // call sensors.requestTemperatures() to issue a global temperature
  // request to all devices on the bus
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperatures
   Serial.println("DONE");
  // After we got the temperatures, we can print them here.
  // We use the function ByIndex, and as an example get the temperature from
the first sensor only.
  Serial.print("Temperature for the device 1 (index 0) is: ");
  float tempC = sensors.getTempCByIndex(0);
  Serial.println(tempC);
  delay(5000);

// Server
  server.handleClient();
  server.on("/Sensor", HTTP_GET, []() {
    String message = "<!doctype html>\n";
    message += "<html>\n";
    message += "  <head>\n";
    message += "    <title>ESP32 Thermometer / Hygrometer</title>\n";
    message += "  </head>\n";
    message += "  <body>\n";
    message += "    <h1>Temperature: ";
     message += String(sensors.getTempCByIndex(0), 2);
    message += " &#8451;</h1>\n";
    message += "  </body>\n";
```

```
     message += "</html>\n";
    server.send(200, "text/html", message);
  });
}

void handleRoot() {
  server.send(200, "text/plain", "Here is your Lolin32!");
}

void handleNotFound() {
  String message = "path ";
  message += server.uri();
  message += " not found.\n";
  server.send(404, "text/plain", message);
}
```

The output should be as in the following image:

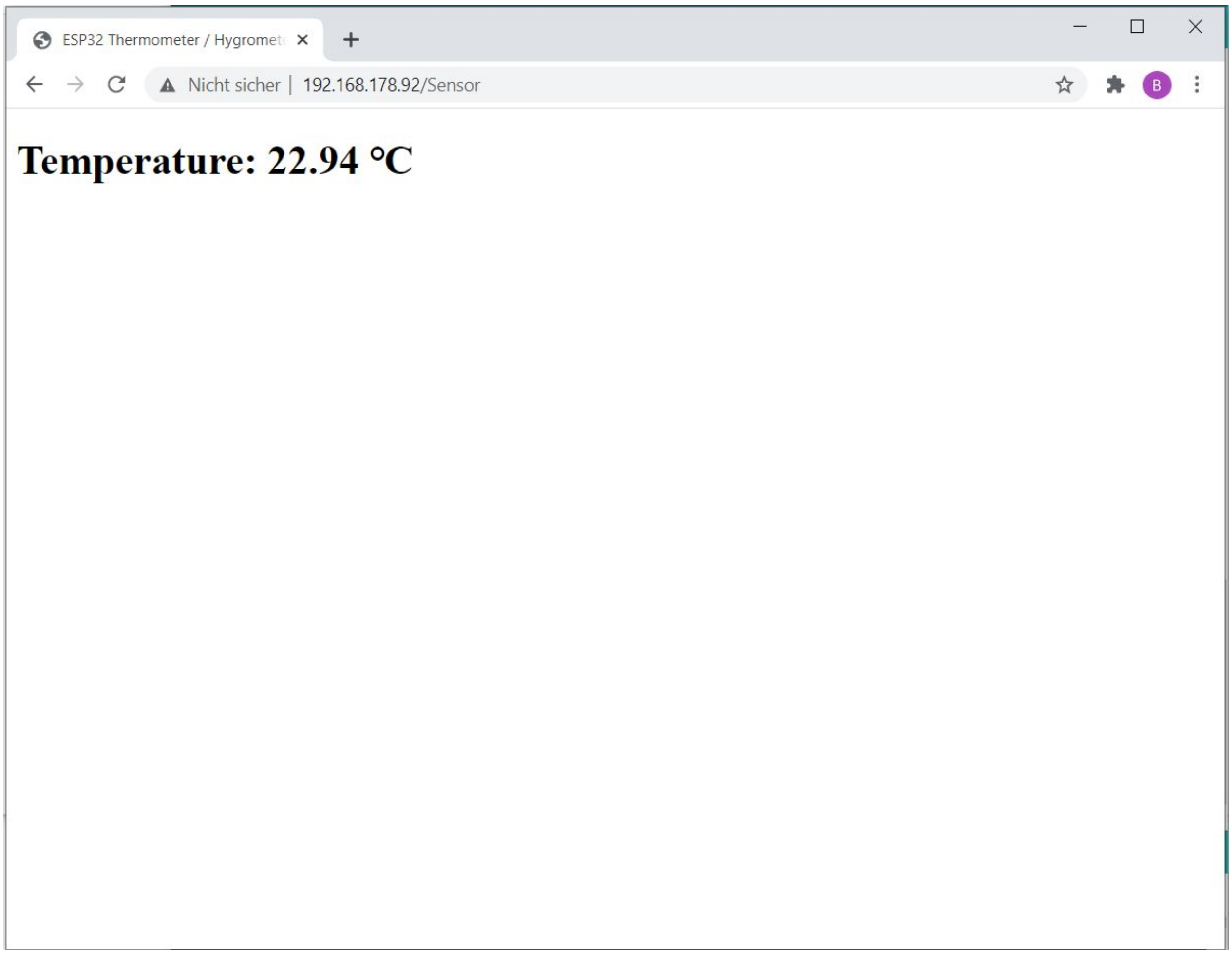


```
.ets Jun  8 2016 00:22:57
22:30:53.038 ->
22:30:53.038 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
22:30:53.038 -> configsip: 0, SPIWP:0xee
22:30:53.038 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
22:30:53.038 -> mode:DIO, clock div:1
22:30:53.038 -> load:0x3fff0018,len:4
22:30:53.085 -> load:0x3fff001c,len:1044
22:30:53.085 -> load:0x40078000,len:8896
22:30:53.085 -> load:0x40080400,len:5816
22:30:53.085 -> entry 0x400806ac
22:30:53.928 -> .
22:30:53.928 -> Connected to XITYLIGHT206194.
22:30:53.928 -> IP-Address: 192.168.178.92.
22:30:53.928 -> MDNS-Responder started.
22:30:53.928 -> HTTP-Server started.
22:30:53.975 -> Dallas DS18B20 Temperature
22:30:53.975 -> Requesting temperatures...DONE
22:30:54.069 -> Temperature for the device 1 (index 0) is: 23.00
22:30:59.065 -> Requesting temperatures...DONE
22:30:59.159 -> Temperature for the device 1 (index 0) is: 22.94
22:31:04.209 -> Requesting temperatures...DONE
22:31:04.303 -> Temperature for the device 1 (index 0) is: 22.94
22:31:09.314 -> Requesting temperatures...DONE
22:31:09.408 -> Temperature for the device 1 (index 0) is: 22.94
22:31:14.463 -> Requesting temperatures...DONE
22:31:14.557 -> Temperature for the device 1 (index 0) is: 22.94
```

Note: Set baud rate to 115200.

WiFi works even when the micro USB cable is not connected. Simply enter the IP address in your browser followed by /Sensor.

When not connected to the computer, the Serial Monitor of the Arduino IDE does not get the information.


ESP32 Thermometer / Hygromet
Nicht sicher | 192.168.178.92/Sensor

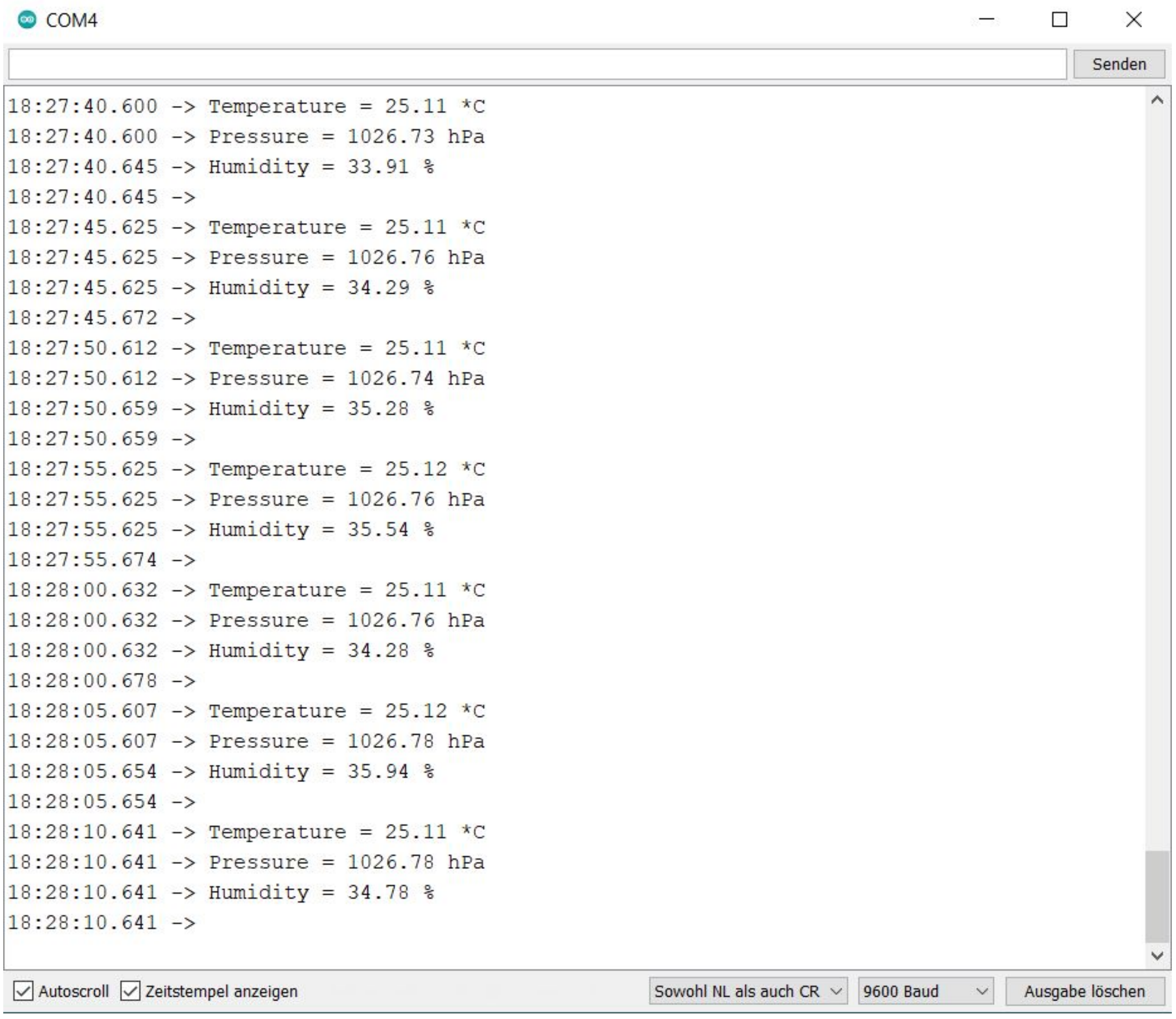
Temperature: 22.94 °C

Example with BME280

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define BME_SCK 18            //use these pins for hardware SPI
#define BME_MISO 19
#define BME_MOSI 23
#define BME_CS 5


Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI


unsigned long delayTime;

void setup() {
    Wire.begin();
    Serial.begin(9600);
    while(!Serial);    // time to get serial running
    Serial.println(F("BME280 test"));
    // default settings
    unsigned status;
```

```
   Wire.begin(26,27);              //add this line for I2C with SDA=26 and
SCL=27
    status = bme.begin();           //default is 0x77; otherwise
bme.begin(0x76)
    if (!status) {
        Serial.println("Could not find a valid BME280 sensor, check wiring,
address, sensor ID!");
        Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
        Serial.print("        ID of 0xFF probably means a bad address, a BMP
180 or BMP 085\n");
        Serial.print(" ID of 0x56-0x58 represents a BMP 280,\n");
        Serial.print("      ID of 0x60 represents a BME 280.\n");
        Serial.print("      ID of 0x61 represents a BME 680.\n");
        while (1) delay(10);
    }

    Serial.println("-- Default Test --");
    delayTime = 5000;
    Serial.println();
}

void loop() {
    printValues();
    delay(delayTime);
}
```

```
void printValues() {
    Serial.print("Temperature = ");
    Serial.print(bme.readTemperature());
    Serial.println(" *C");
    Serial.print("Pressure = ");
    Serial.print(bme.readPressure() / 100.0F);
    Serial.println(" hPa");
    Serial.print("Humidity = ");
    Serial.print(bme.readHumidity());
      Serial.println(" %");
    Serial.println();
}
```



```
18:27:40.600 -> Temperature = 25.11 *C
18:27:40.600 -> Pressure = 1026.73 hPa
18:27:40.645 -> Humidity = 33.91 %
18:27:40.645 ->
18:27:45.625 -> Temperature = 25.11 *C
18:27:45.625 -> Pressure = 1026.76 hPa
18:27:45.625 -> Humidity = 34.29 %
18:27:45.672 ->
18:27:50.612 -> Temperature = 25.11 *C
18:27:50.612 -> Pressure = 1026.74 hPa
18:27:50.659 -> Humidity = 35.28 %
18:27:50.659 ->
18:27:55.625 -> Temperature = 25.12 *C
18:27:55.625 -> Pressure = 1026.76 hPa
18:27:55.625 -> Humidity = 35.54 %
18:27:55.674 ->
18:28:00.632 -> Temperature = 25.11 *C
18:28:00.632 -> Pressure = 1026.76 hPa
18:28:00.632 -> Humidity = 34.28 %
18:28:00.678 ->
18:28:05.607 -> Temperature = 25.12 *C
18:28:05.607 -> Pressure = 1026.78 hPa
18:28:05.654 -> Humidity = 35.94 %
18:28:05.654 ->
18:28:10.641 -> Temperature = 25.11 *C
18:28:10.641 -> Pressure = 1026.78 hPa
18:28:10.641 -> Humidity = 34.78 %
18:28:10.641 ->
```

Note: This works for I2C on pins 26 and 27 and could also be used (with minor modifications) for SPI. Sketch and Libraries are from Adafruit (US).

Now it is the time to learn and make your own projects. 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.