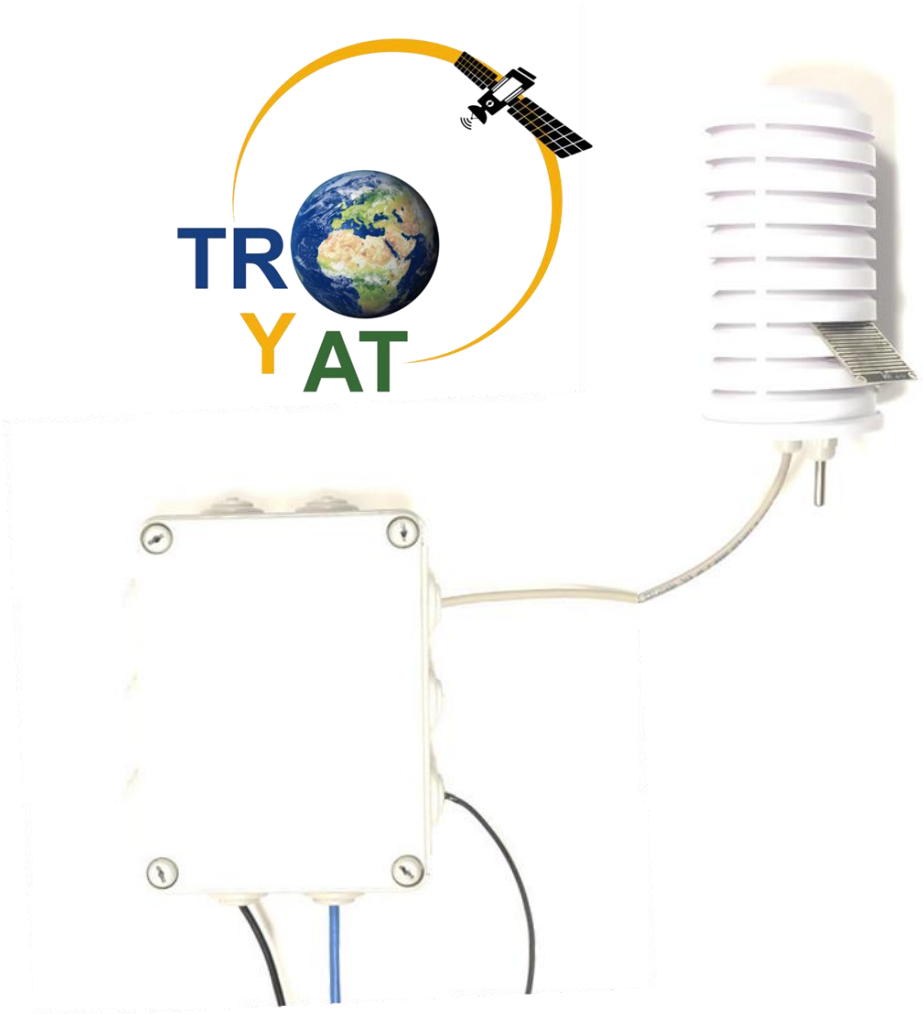
TRACK YOUR ATMOSPHERE

Intellectual Output IO2

STARTER KIT

Erasmus+ Projekt 2017-1-DE02-KA202-004229



STARTER KIT

Manual v. 1.4

Intellectual Output 2

Content

Co-funded by the Erasmus+ Programme of the European Union

The Aim of project.

The main goal of project is to build a simple weather station with a GPS sensor and integrate this information disseminating on a web page.

Our kit is composed by an internal unit (the core of the kit) and an external unit where the weather sensor is mounted, and the GPS antenna is located.

This scheme summarizes the system:

- **E**: External Box with:
  - Temperature Sensor
  - Temperature, Humidity, Pression Sensor
  - Rain Sensor
- **I**: Internal Box with:
  - CPU
  - GPS shield
  - Ethernet Shield
  - Temperature, Humidity, Pression Sensor

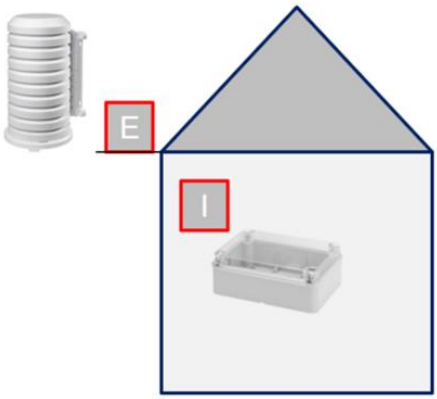
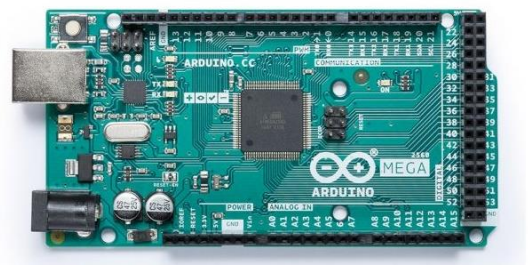


Fig. 18. A simply schematic of the system

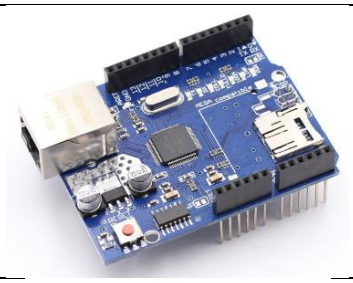
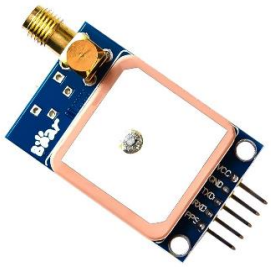
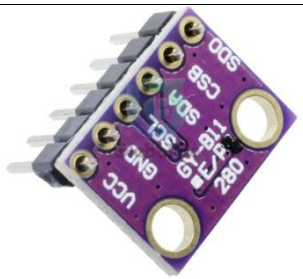
Packing list

Below, in tables 1, the components list required for constructing the Starter Kit.

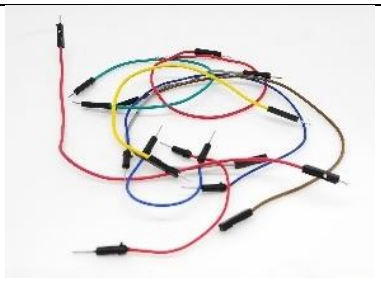
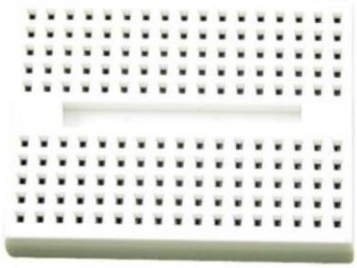
Tab. 1. Components required for the manufacture of the instrument.

Item	Picture	Quantity
ARDUINO MEGA 2560 [A000067]		1

Co-funded by the Erasmus+ Programme of the European Union

Ethernet Shield + Memory Slot per schede di memoria microSD, W5100 Controller per Arduino UNO/Mega		1
GPS Antenna		1
GPS Mini NEO-7M /NEO module		1
GY-BME280 Pressure Sensor		1
DS18B20 Temperature Sensor		1
Resistor **4.7KΩ**		1

Co-funded by the
Erasmus+ Programme
of the European Union

Electrical junction box for Arduino Mega		1
Wires		1
Breadboard mini		2
Enclosure for external sensors		1
Rain sensor		1

Co-funded by the Erasmus+ Programme of the European Union

Starting from scratch!

Installing IDE

Introduction

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. In this session, you will learn how to setup your computer to use Arduino and how to set about the lessons that follow. The Arduino software that you will use to program your Arduino is available for Windows, Mac and Linux. The installation process is different for the three operating Systems and unfortunately a bit tricky.

STEP1:

Go to https://www.arduino.cc/en/Main/Software and find below page.

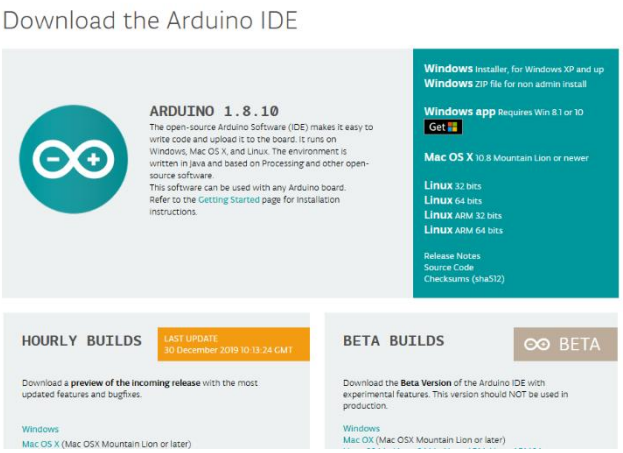


Fig. 1. Screenshot from Arduino's web page

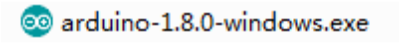
The release available on the website is usually the latest one, and it may be newer than the version shown in the picture.

STEP2:

Download the development software accordingly to the operating system of your computer.

Installing Arduino (Windows)

Install Arduino with the exe. Installation package.



Co-funded by the Erasmus+ Programme of the European Union

Fig. 2. Screenshot to accept the Licence Agreement

Click *I Agree* to see the following interface

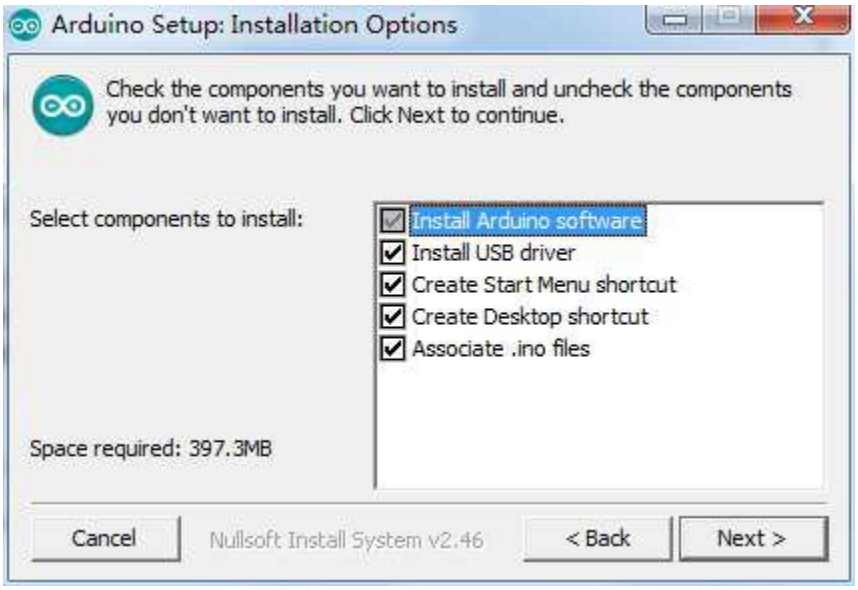


Fig. 3. Screenshot to check the Installation Options

Click *Next*

Co-funded by the Erasmus+ Programme of the European Union

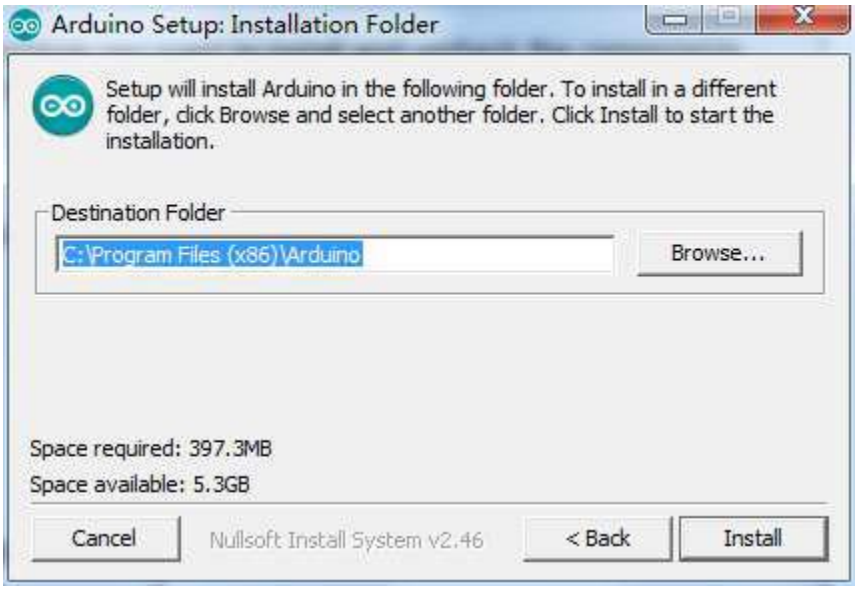


Fig. 4 Screenshot to select the installation folder

You can press Browse… to choose an installation path or directly type the target directory.

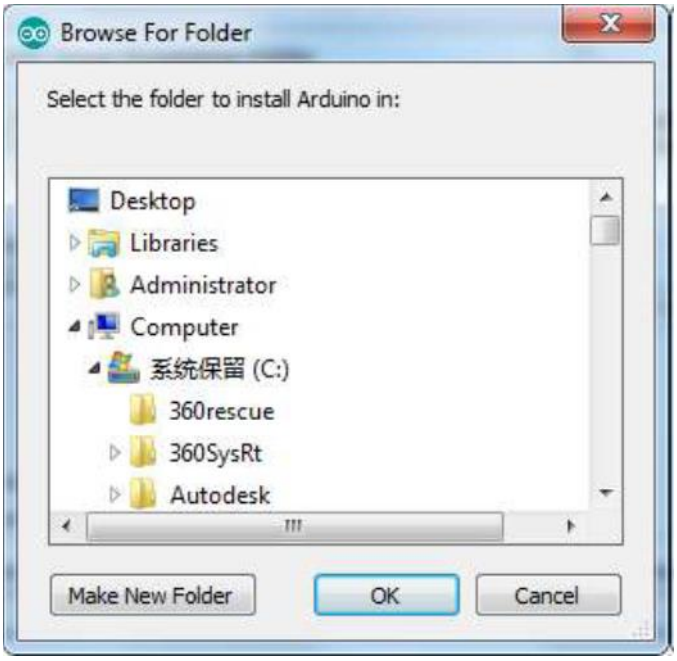


Fig. 5 Screenshot to browse the folder of your device

Click *Install* to launch the installation process

Co-funded by the Erasmus+ Programme of the European Union

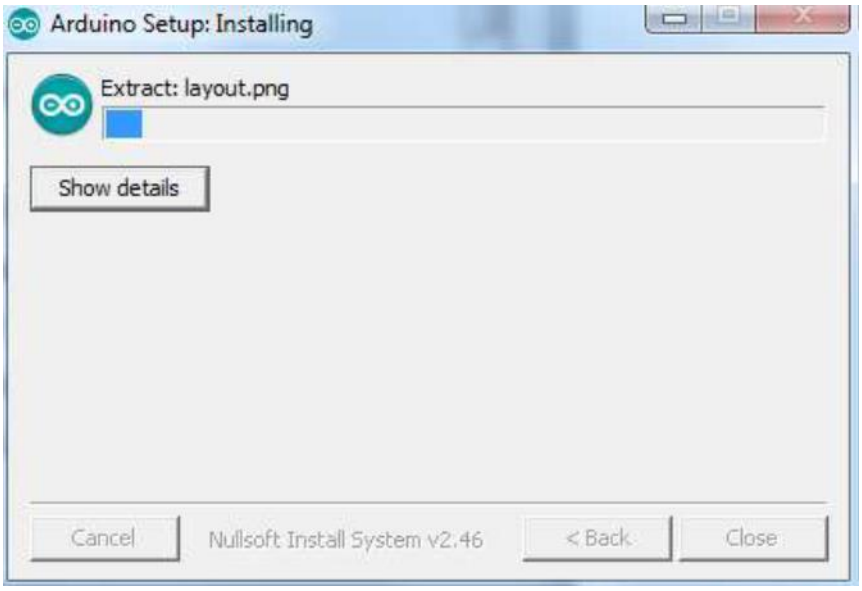


Fig. 6. Screenshot showing the progress bar during the installation

As the following dialog window appears, click *Install* to finish the installation.

Fig. 7. Screenshot showing how to accept the final step of the installation

At the end, the Arduino icon appears on your desktop ready to be clicked.

Co-funded by the Erasmus+ Programme of the European Union

Add Libraries and Open Serial Monitor

Installing Additional Arduino Libraries

Once you are comfortable with the Arduino software and using the built-in functions, you may want to extend the ability of your Arduino with additional libraries.

What are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library allows to control LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you simply need to install them.

How to Install a Library

Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.

Then the library manager will open, and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, then select the version of the library you want. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry - it is normal.

Co-funded by the Erasmus+ Programme of the European Union

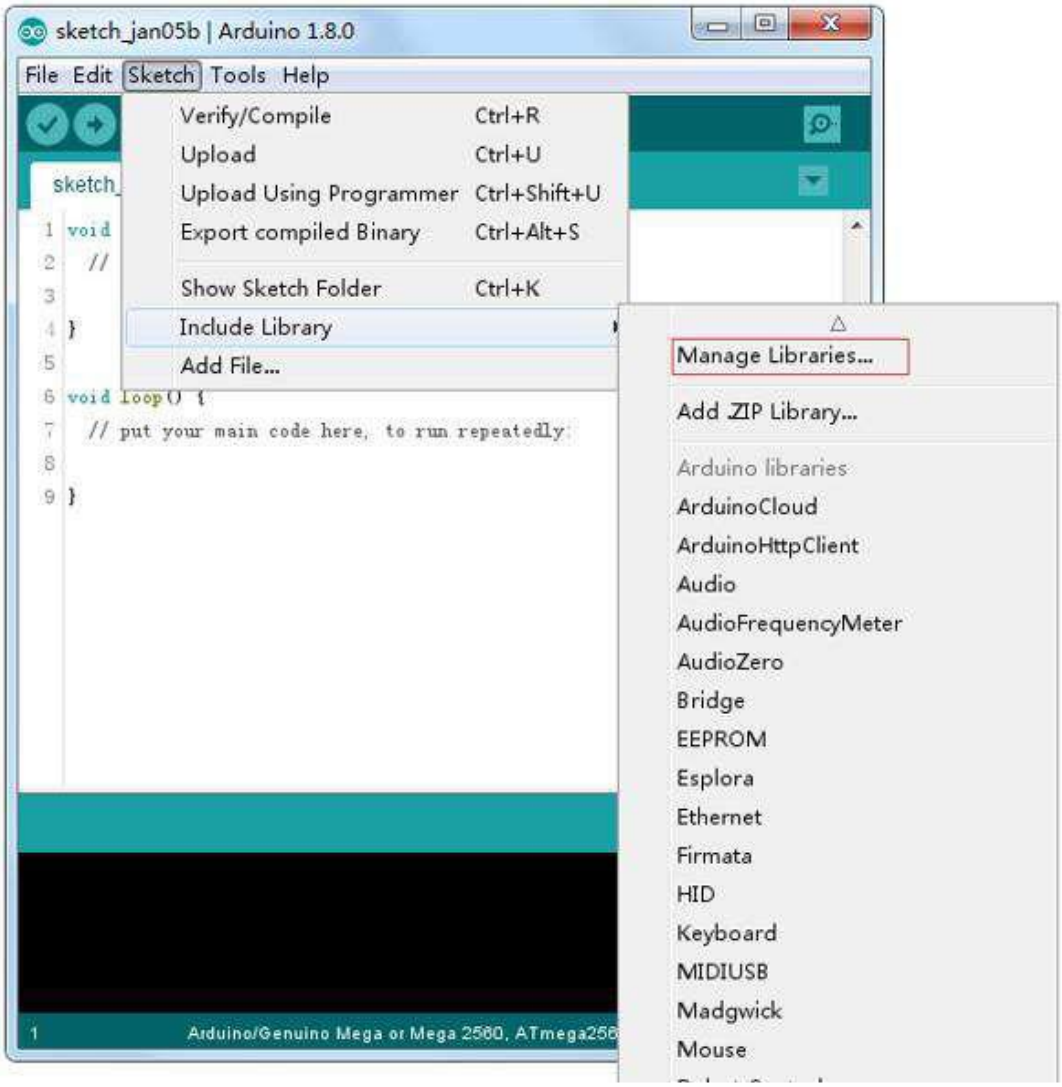


Fig. 8. Screenshot to show how to manage new libraries (1/6)

There are times you have to be patient with it, just as shown in the figure. Please refresh it and wait.

Co-funded by the Erasmus+ Programme of the European Union

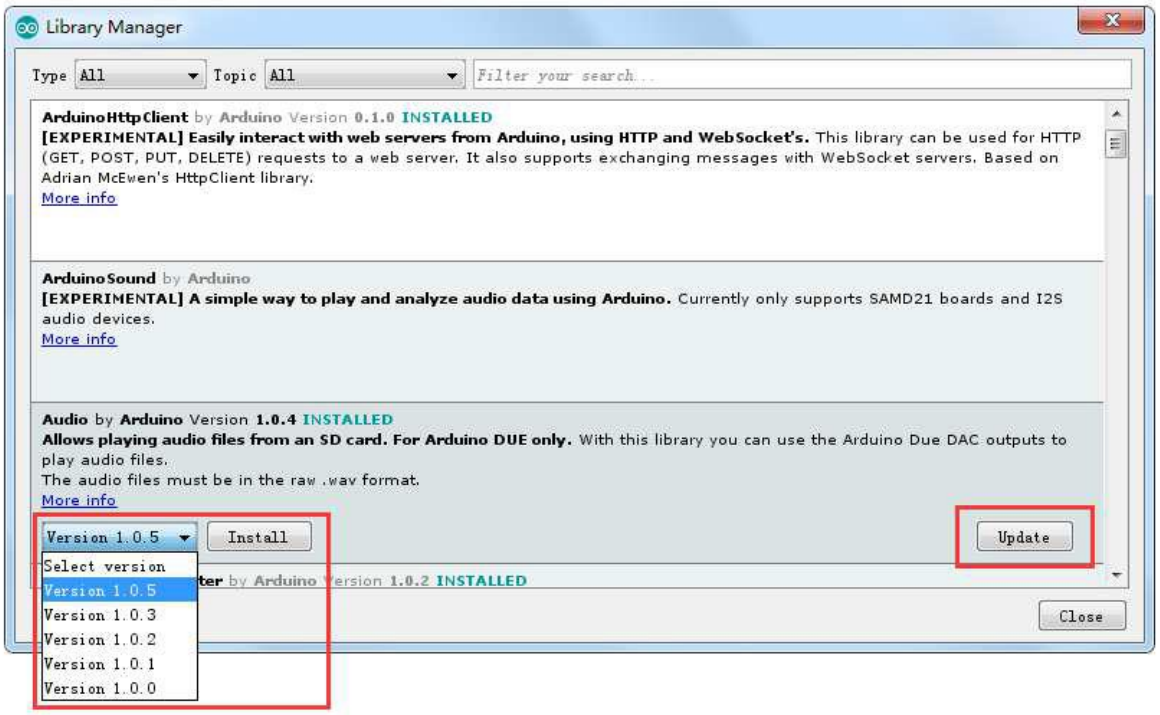


Fig. 9. Screenshot to show how to manage new libraries (2/6)

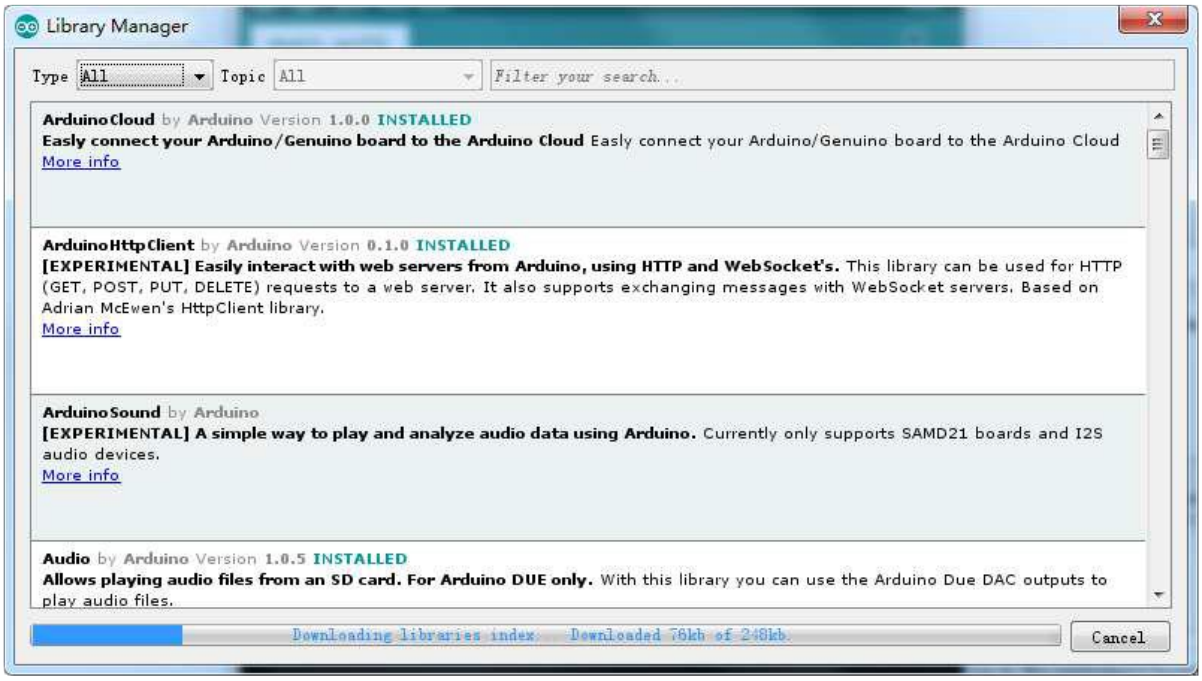


Fig. 10. Screenshot to show how to manage new libraries (3/6)

Co-funded by the
Erasmus+ Programme
of the European Union

Finally, click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. Now, you can close the library manager.

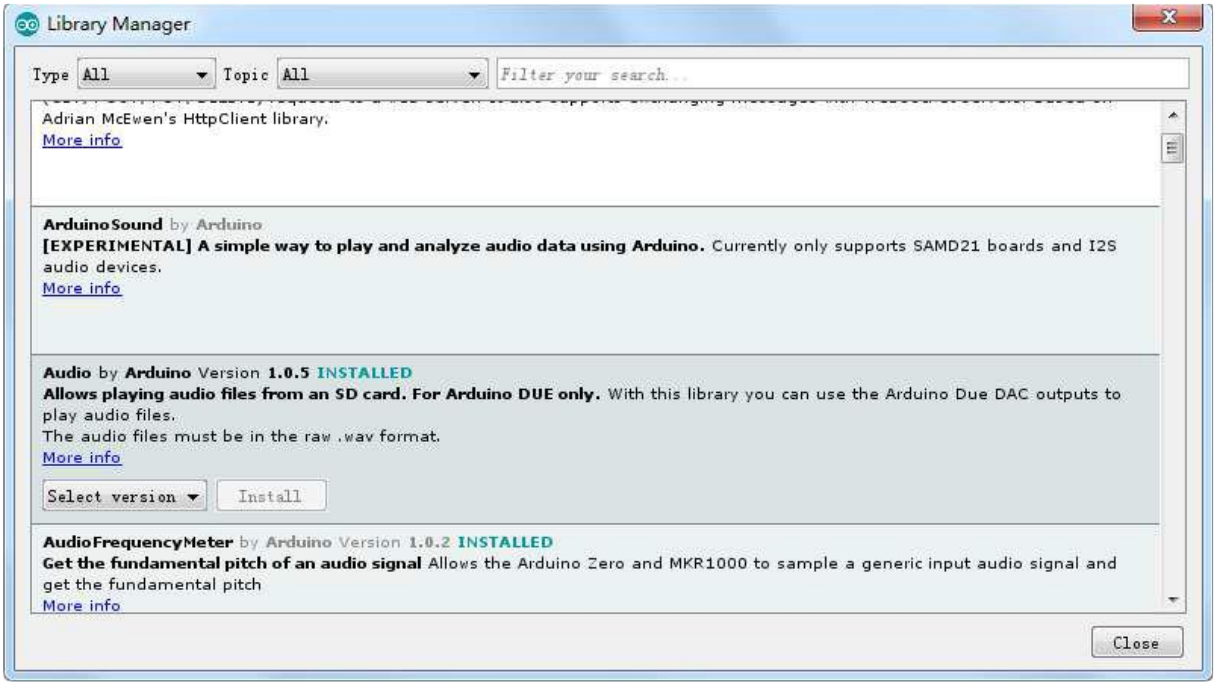


Fig. 11. Screenshot to show how to manage new libraries (4/6)

You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on Github.

Importing a .zip Library

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a *.cpp file, a *.h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is. In the Arduino IDE, navigate to Sketch > Include Library. At the top of the drop-down list, select the option "Add .ZIP Library".

Co-funded by the Erasmus+ Programme of the European Union

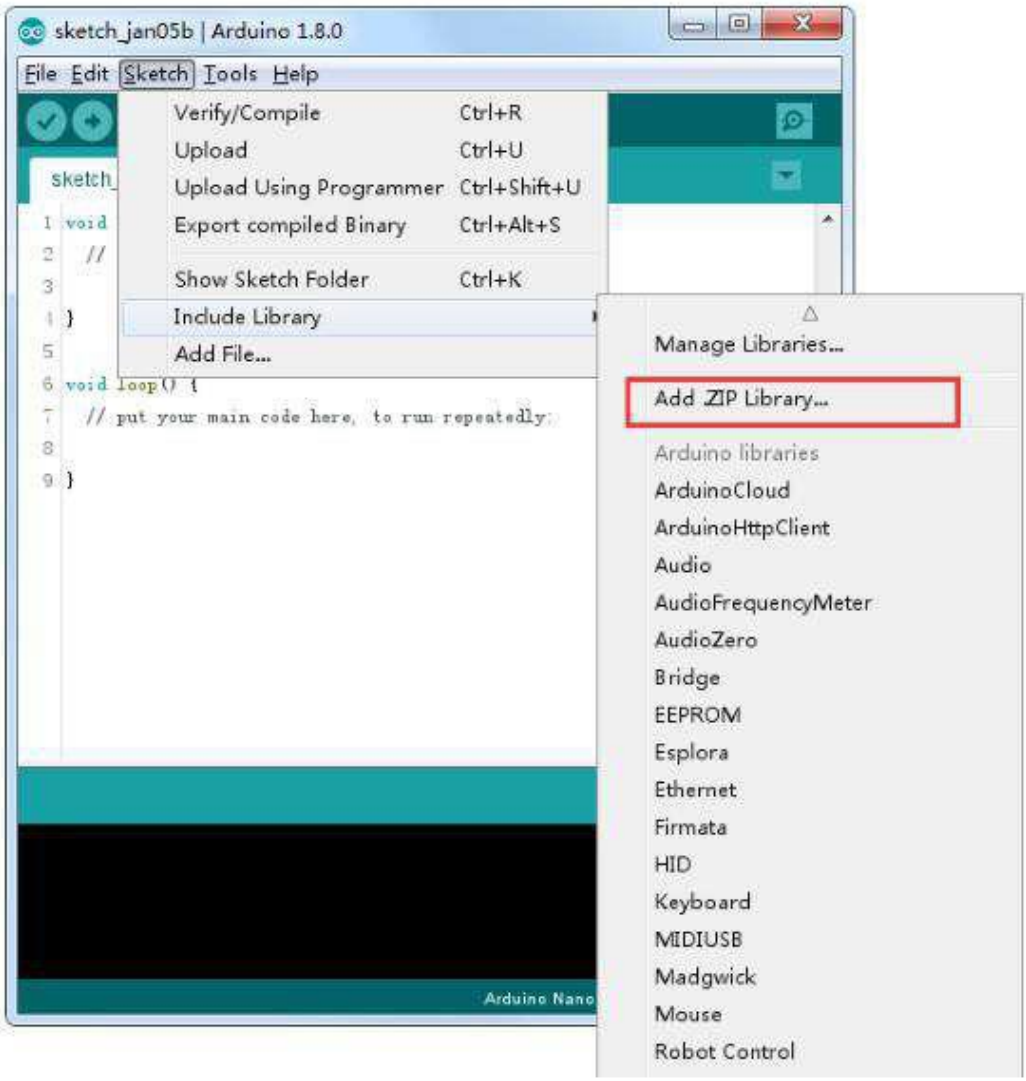


Fig. 12. Screenshot to show how to manage new libraries (5/6)

You will be prompted to select the library you would like to add. Navigate to the *.zip file's location and open it.

Return to the Sketch > Import Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

NB: the library will be available to use in sketches, but examples for the library will not be exposed in the File > Examples until after the IDE has restarted.
Installation under MAC and Linux systems can be handled likewise.
The manual installation we are going to introduce hereafter, as an alternative, is rarely used, hence users with no needs may skip it.

Co-funded by the Erasmus+ Programme of the European Union

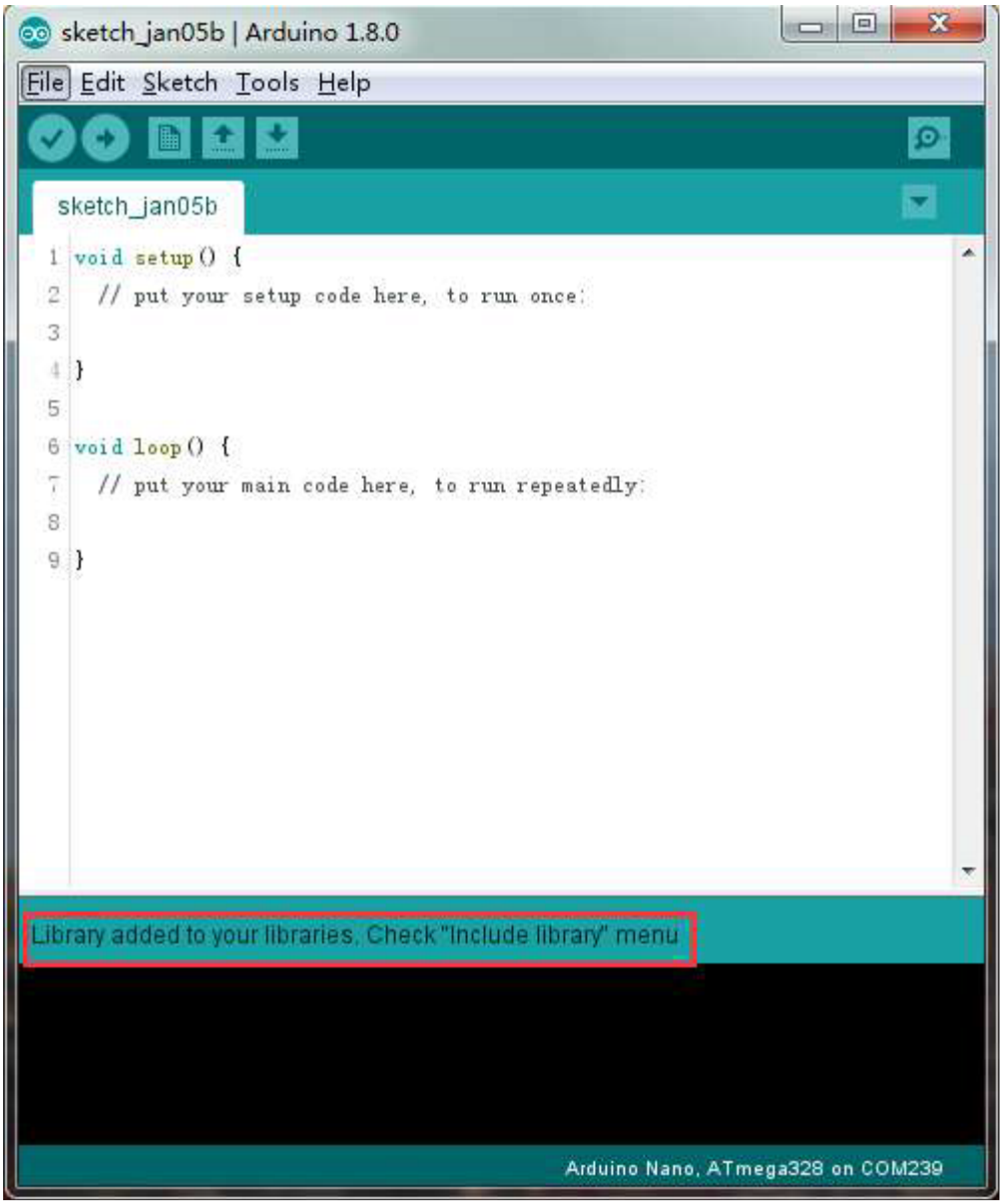


Fig. 13 Screenshot to show how to manage new libraries (6/6)

Manual installation

To install the library, first quit the Arduino application. Then uncompress the ZIP-file containing the library. For example, if you are installing a library called "ArduinoParty", uncompress ArduinoParty.zip. It should contain a folder called ArduinoParty, with files like ArduinoParty.cpp and ArduinoParty.h inside. (If the *.cpp and *.h files are not in the folder, you will need to create one. In this case, you would make a folder called "ArduinoParty" and move into it all the files from the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.) Drag the ArduinoParty folder into this folder (your libraries folder). Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Co-funded by the Erasmus+ Programme of the European Union

There may be more files than just the .cpp and .h files, just make sure they're all there. (The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder.

For example:

Documents\Arduino\libraries\ArduinoParty.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp won't work.)

Restart the Arduino application. Make sure the new library appears in the Sketch ->Import Library menu item of the software. Now you have correctly installed a library.

Arduino Serial Monitor (Windows, Mac, Linux)

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. Since using a terminal is such a main part of working with Arduinos and other microcontrollers, developpers decided to include a serial terminal with the software. Within the Arduino environment, this is called “Serial Monitor”.

Making a Connection

Serial monitor comes with all versions of the Arduino IDE; simply click the Serial Monitor icon to open it

Co-funded by the Erasmus+ Programme of the European Union

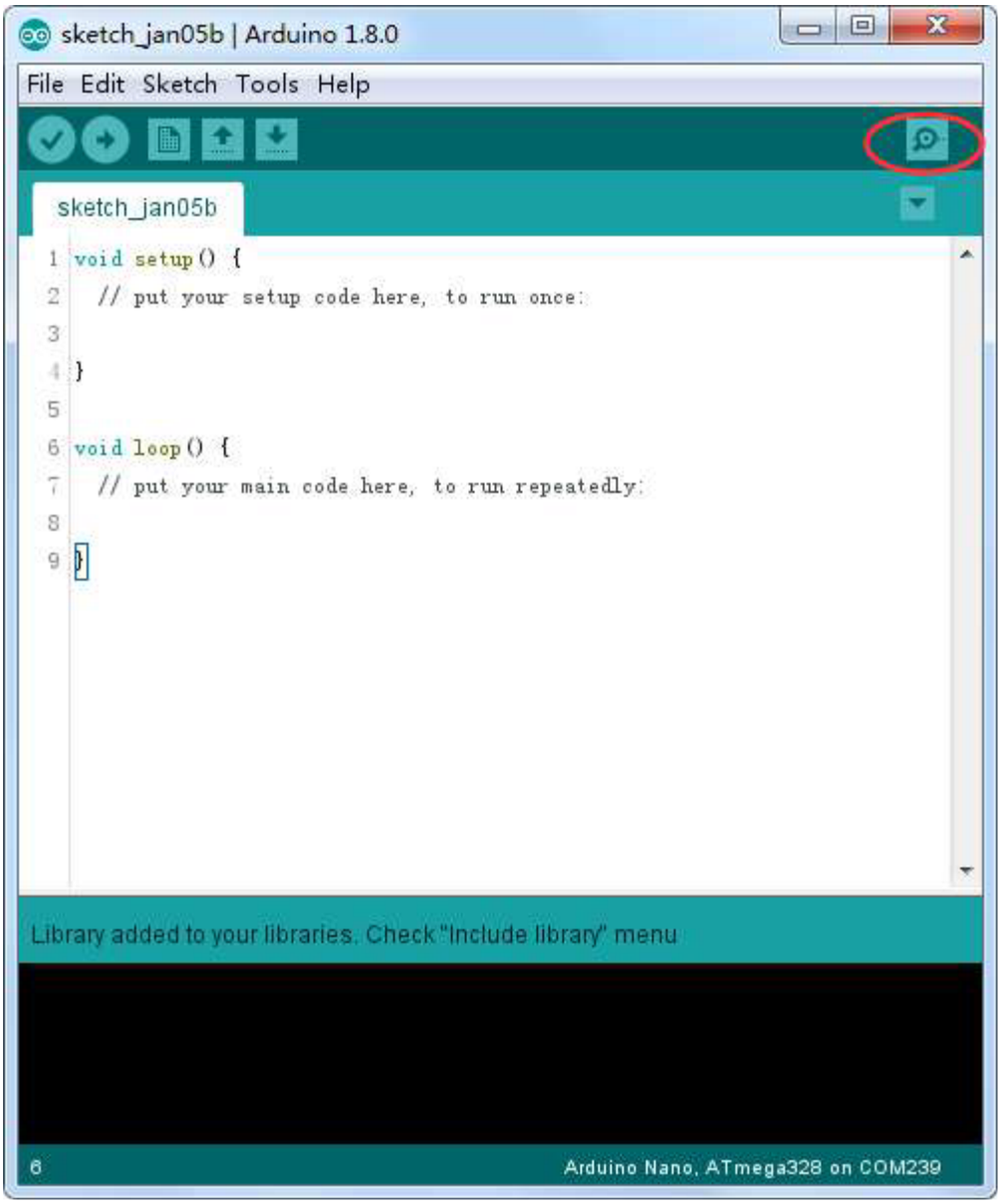


Fig. 14. How to Open a Serial Monitor

Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port and select the suitable port.

Tips: Choose the same COM port as you have in Device Manager.

Co-funded by the Erasmus+ Programme of the European Union

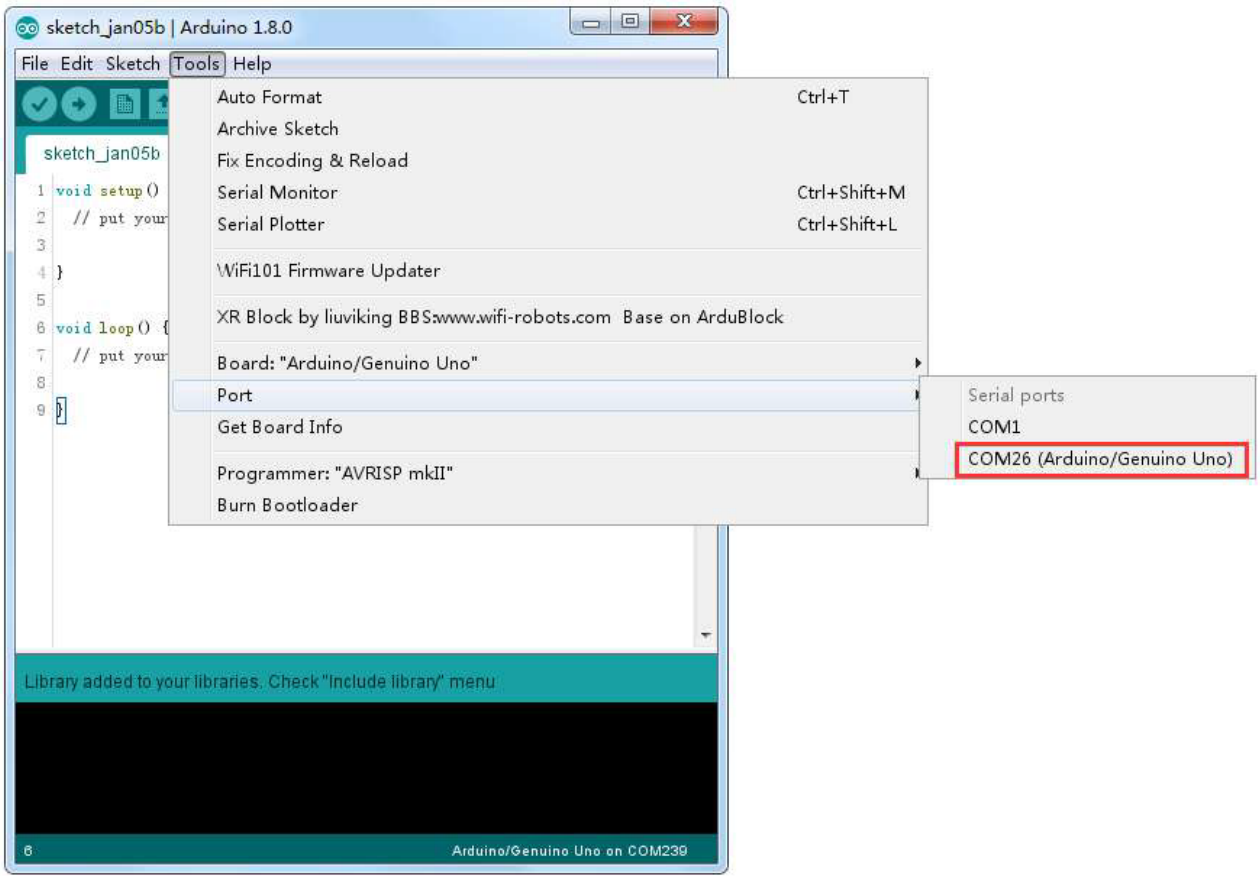


Fig. 15 How to select the right Com port

Once, the Port has been properly selected, you should see something like this:

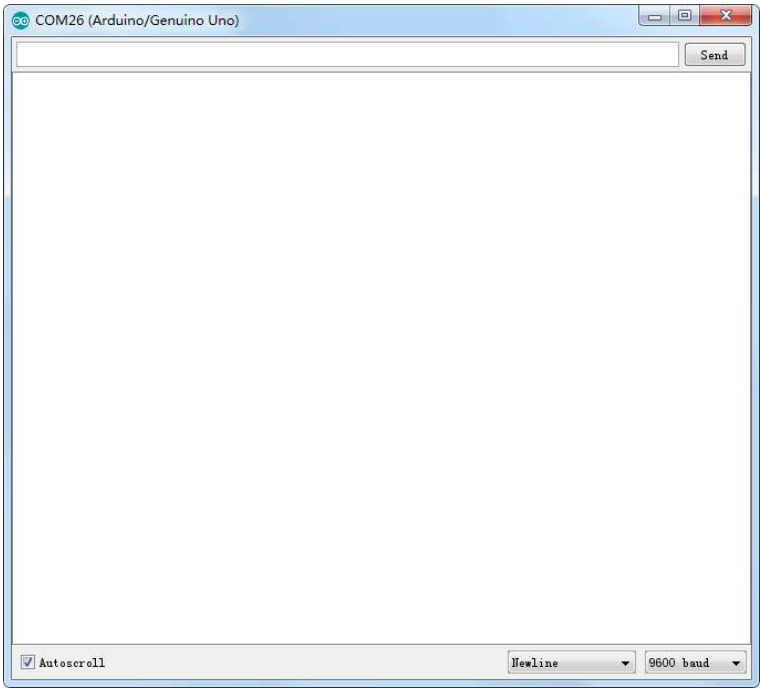


Fig. 16. Screenshot to show how the serial monitor looks

Co-funded by the Erasmus+ Programme of the European Union

Settings

The Serial Monitor has limited settings, but enough to fulfil most of your serial communication needs. The first setting you have to change is the baud rate. Click on the baud rate drop-down menu to select the correct one. (9600 baud)

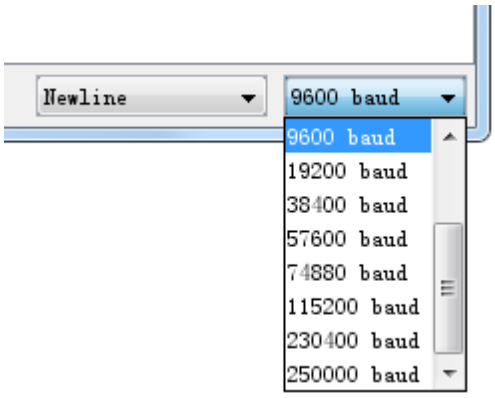


Fig. 17. Select the right serial baud rate set in your code

Pros

The Serial Monitor is a quick and easy way to establish a serial connection with your Arduino. If you're already working in the Arduino IDE, there's really no need to open up a separate terminal to display data.

Cons

The lack of settings leaves much to be desired in the Serial Monitor, and, for advanced serial communications, it may not do the trick.

Which libraries we need

For a working project we need some libraries that will be include in the code.

The libraries to find and install are:

- TIME
- STREAMING
- ONEWIRE
- DALLASTEMPERATURE
- TINYGPS
- ETHERNET
- BME280

Co-funded by the Erasmus+ Programme of the European Union

Internal Box

Let's start working seriously

Boards Assembly

First, we have to get to know the electronic boards we have to work with.

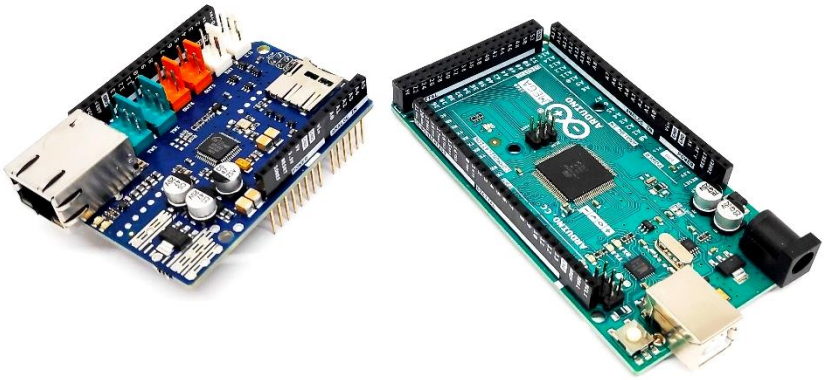


Fig. 19. Left, Ethernet Shield v2. Right, Arduino Mega 2560.

We begin by assembling these two boards. Through the strip line on the borders, we can easily connect wires and sensors to this kit.

Fig. 20. The Arduino Mega with Ethernet Shield plugged

Co-funded by the
Erasmus+ Programme
of the European Union

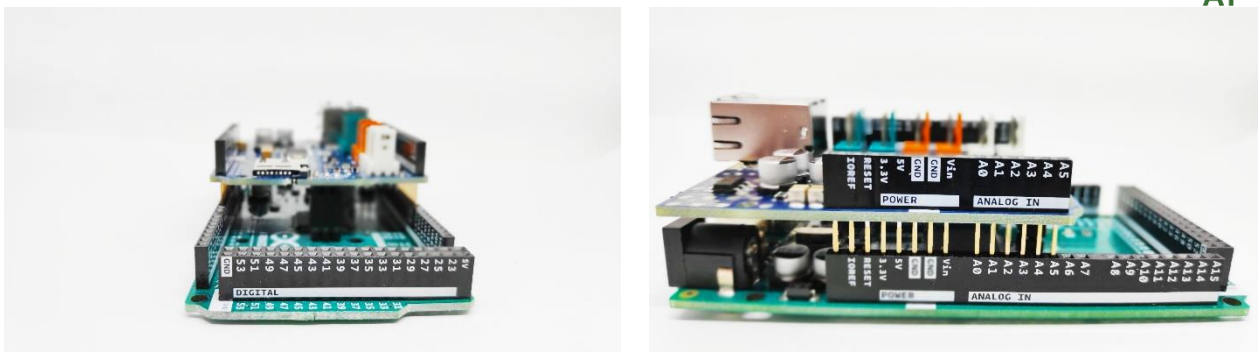


Fig. 21 Different point of view for Arduino Mega and Ethernet Shield (1/2)

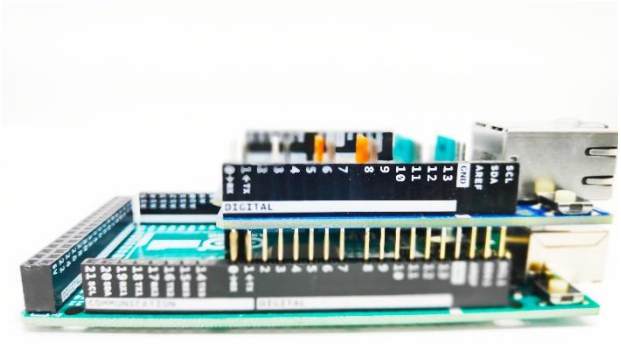


Fig. 22 Different point of view for Arduino Mega and Ethernet Shield (2/2)

As a battleship game, we have to connect the sensor's pins avoiding errors for a successful project.

Co-funded by the
Erasmus+ Programme
of the European Union

Fig. 23 A possible way of implementation

External Box

The External Box is composed of a cheap TFA Weather Sensor Housing capable to protect from rainfall and direct sunrays, which are deadly for electronics components.

Co-funded by the Erasmus+ Programme of the European Union

Fig. 24. Picture of the external box mounting on pole

On the bottom of this enclosure, we can use two cable glands to install the external temperature sensor (DS18B20) and for the connection cable.

Fig. 25. External Temperature sensor DS18B20 and cable to connect to internal box

External Sensor BME280

If you haven't already installed the BME280's library, follow this step.

To monitor the external environmental parameters, we will use the BME280 device. The BME280 is an absolute barometric pressure sensor, which is especially feasible for mobile applications. The

Co-funded by the
Erasmus+ Programme
of the European Union

BME280 is based on Bosch's proven piezo-resistive pressure sensor technology featuring high accuracy and linearity as well as long-term stability and high EMC robustness. Numerous device operation options guarantee for highest flexibility. The device is optimized in terms of power consumption, resolution and filter performance. [for its datasheet: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280]

You can easily wire this breakout to any microcontroller, we'll be using an Arduino one. For different microcontrollers, as long as you have 4 available pins, it is possible to 'bit-bang SPI' or you can use two I2C pins, but usually those pins are fixed in hardware. Just check out the library, and then port the code.

I2C Wiring

Use this wiring if you want to connect via I2C interface

- Connect Vin to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect GND to common power/data ground
- Connect the SCK pin to the I2C clock SCL pin on your Arduino. On an UNO this is also known as A5, on a Mega it is also known as digital 21.
- Connect the SDI pin to the I2C data SDA pin on your Arduino. On an UNO this is also known as A4, on a Mega it is also known as digital 20

Download Adafruit_BME280 library

To start reading sensor data, you need to install the Adafruit_BME280 library (code on our github repository). It is available from the Arduino library manager so we recommend using that.

Open up the library manager from the IDE...

Co-funded by the Erasmus+ Programme of the European Union

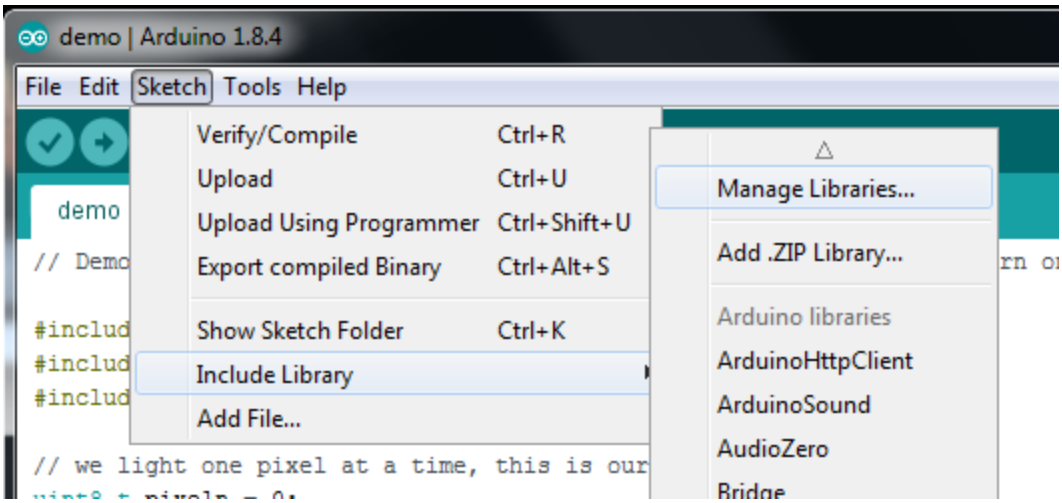


Fig. 26. Screenshot to show how to install the new library (1/3)

Then, type in **adafruit bme280** to locate the library. Click **Install**

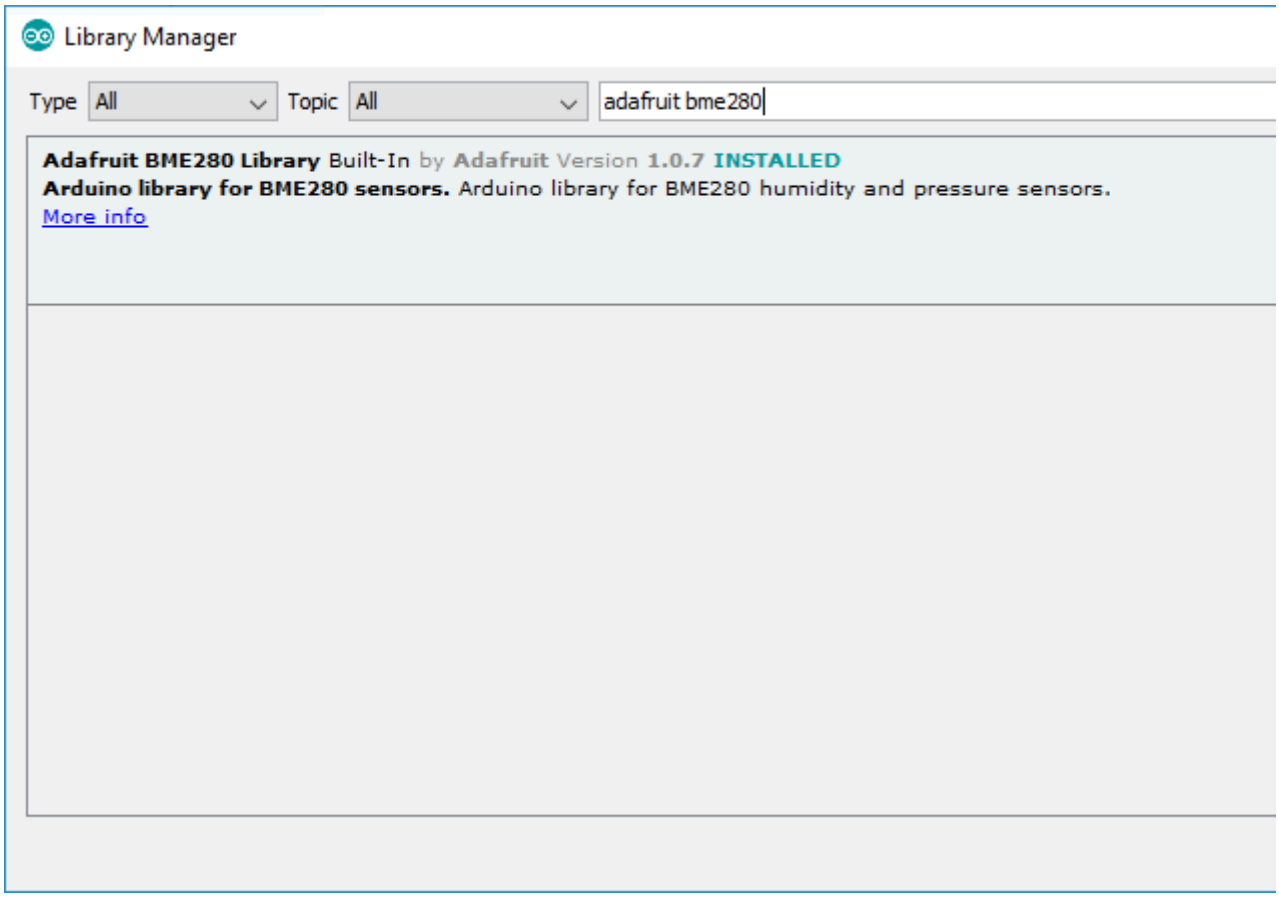


Fig. 27. Screenshot to show how to install the new library (2/3)

Co-funded by the Erasmus+ Programme of the European Union

You'll also need to install the **Adafruit Unified Sensor** library

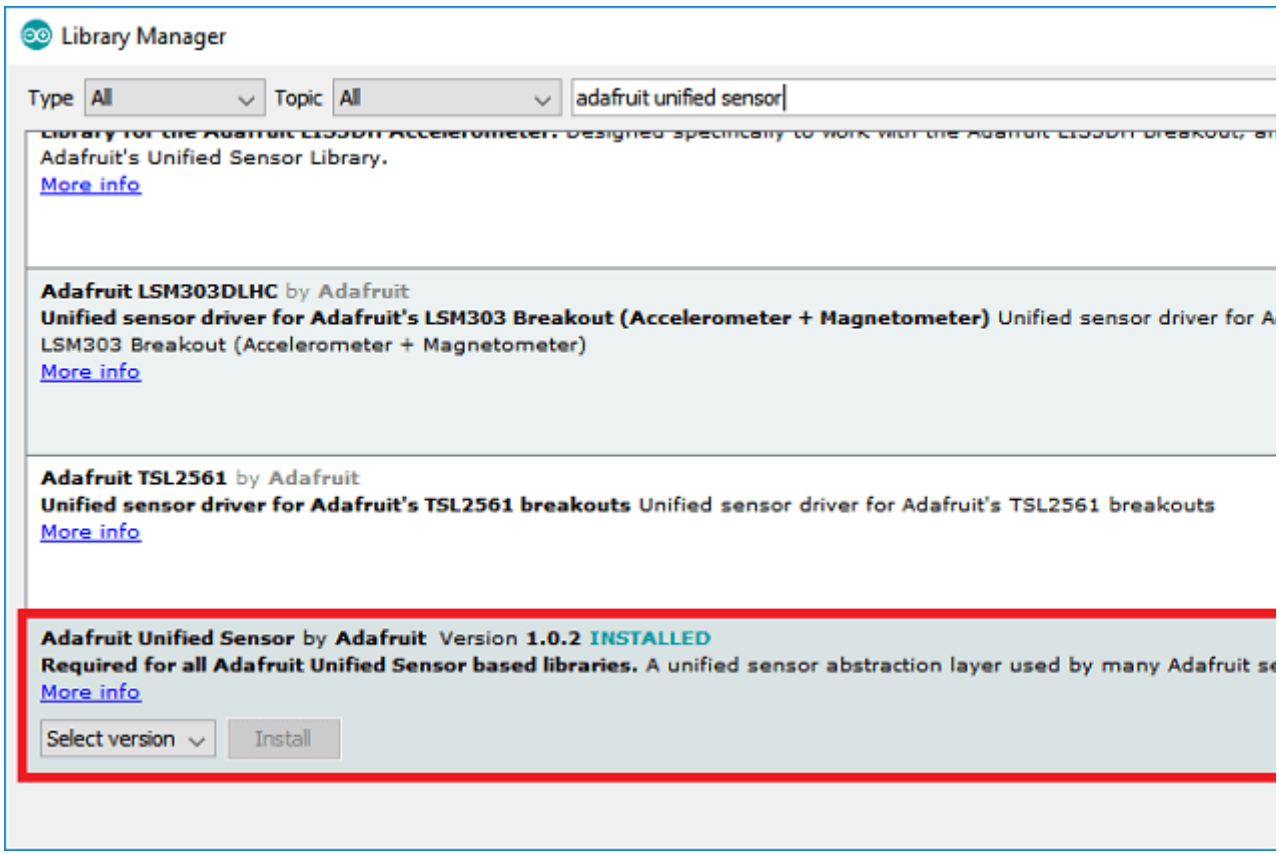


Fig. 28. Screenshot to show how to install the new library (3/3)

Load a demo to practice

Open up **File->Examples->Adafruit_BME280->bme280test** and upload to your Arduino wired up to the sensor

Co-funded by the Erasmus+ Programme of the European Union

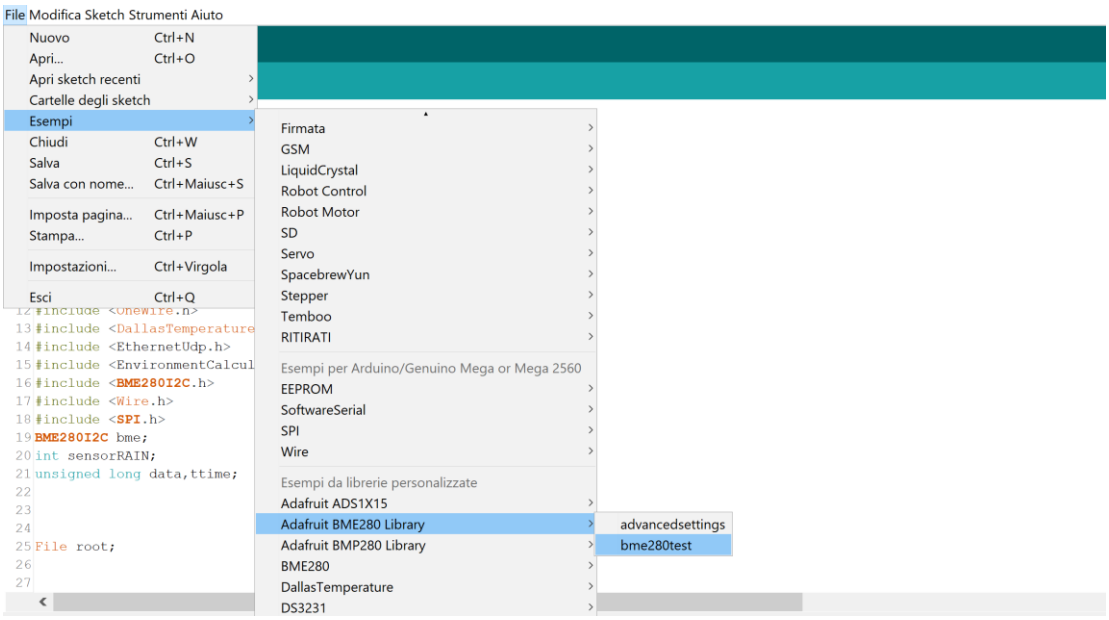


Fig. 29. Screenshot to show how find an example code

Depending on whether you are using I2C or SPI, accordingly change the pin names and comment or uncomment the following lines.

Once uploaded to your Arduino, open up the serial console at 9600 baud speed to see data being printed out

Temperature is calculated in Celsius degrees (°C), you may want to convert this to Farenheit (F) by using the classic F = C * 9/5 + 32 equation.

Pressure is returned in the SI units of **Pascals**. 100 Pascals = 1 hPa = 1 millibar. Sometimes barometric pressure is given in millibar or inches of mercury (inHg). For future reference 1 pascal =0.000295333727 inHg, or 1 inHg = 3386.39 Pascal. So, if you take the value in Pascal of say 100734 and divide by 3389.39 you will get 29.72 inHg.

You can also calculate Altitude. **However, you can only really do a good accurate job of calculating altitude if you know the hPa pressure at sea level for your location and day!** The sensor is quite precise but if you do not have the data updated for the current day then it can be difficult to attain more accurate than 10 meters.

Co-funded by the Erasmus+ Programme of the European Union

Step by step

You can start out by creating a BME280 object with either software SPI (where all four pins can be any I/O) using

1. **Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);**

Alternatively, you can use hardware SPI. With hardware SPI you must use the hardware SPI pins for your Arduino - and each Arduino type has different pins! Check the SPI reference to see what pins to use.
In this case, you can use any CS pins, but the other three pins are fixed

1. **Adafruit_BME280 bme(BME_CS); // hardware SPI**

or I2C using the default I2C bus, no pins are assigned

1. **Adafruit_BME280 bme; // I2C**

Once started, you can initialize the sensor with

1. **if (!bme.begin()) {**
2. **Serial.println("Could not find a valid BME280 sensor, check wiring!");**
3. **while (1);**
4. **}**

begin() will return True if the sensor was found, and False if not. If you get a False value back, check your wiring!

Reading humidity, temperature and pressure is an easy task, just call:

1. **bme.readTemperature()**
2. **bme.readPressure()**
3. **bme.readHumidity()**

Temperature value is always a floating point one, in Centigrade. Pressure is a 32 bit integer value with the pressure in Pascals. You may need to convert to a different value to match it with your weather report. Humidity is in % Relative Humidity

Co-funded by the Erasmus+ Programme of the European Union

It is also possible to turn the BME280 into an altimeter. If you know the pressure at sea level, the library can convert the current barometric pressure into altitude

1. **bme.readAltitude(seaLevelPressure)**

External Rain Sensor

The rain sensor is used to detect rain water and it can detect beyond what a humidity sensor does. The following article explains how to properly use the FC-37 rain sensor module with the Arduino.

The FC-37 rain sensor (or other versions like YL-83) is composed of two pieces (see in the figure below): the electronic board (on the left) and the collector board (on the right) that collects the water drops, as you can see in the following figure:

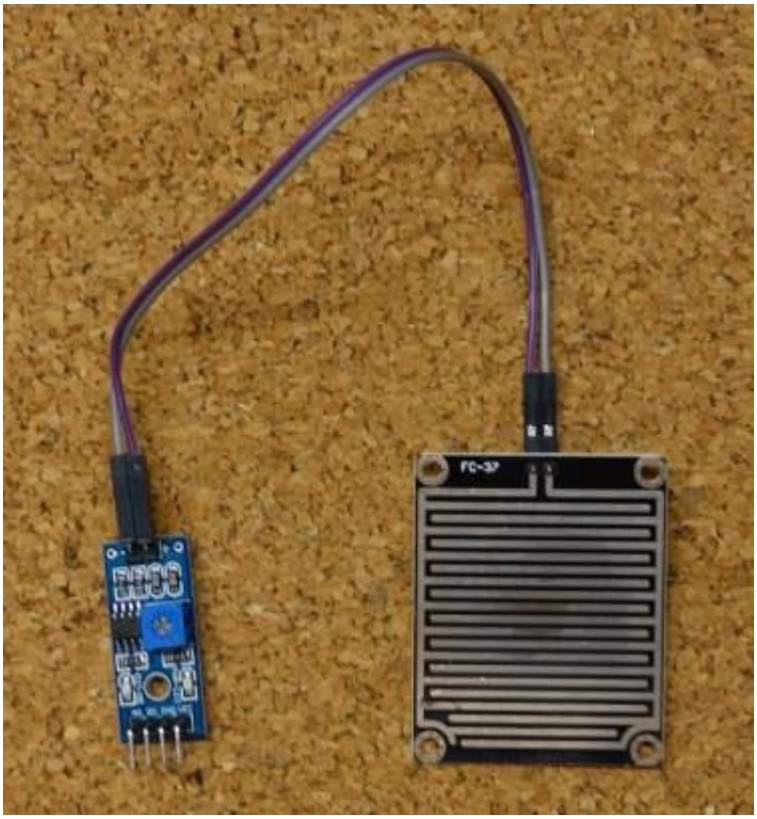


Fig. 30. Rain Sensor

Co-funded by the
Erasmus+ Programme
of the European Union

The rain sensor has a built-in potentiometer for sensitivity adjustment of the digital output (D0). It also has a power LED that lights up when the sensor is turned on and a digital output LED.

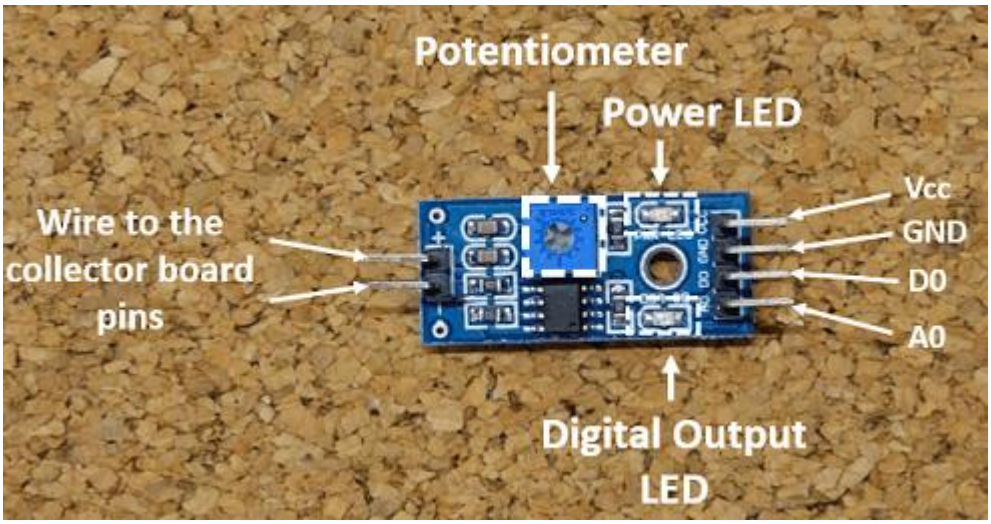


Fig. 31. Detail of rain sensor

How does it work?

Basically, the resistance of the collector board varies accordingly to the amount of water on its surface.

When the board is:

- **Wet:** the resistance increases, and the output voltage decreases
- **Dry:** the resistance decreases, and the output voltage increases

Co-funded by the Erasmus+ Programme of the European Union

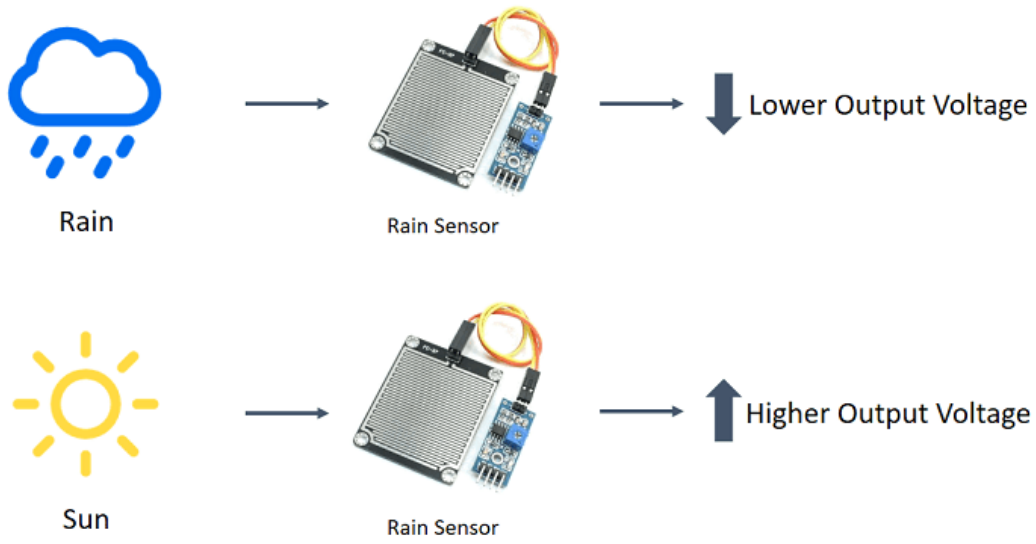


Fig. 32. Easy scheme to understand the operating mode.

With these simple schematics is possible to connect the rain sensor to “Arduino Uno” for a quick test.

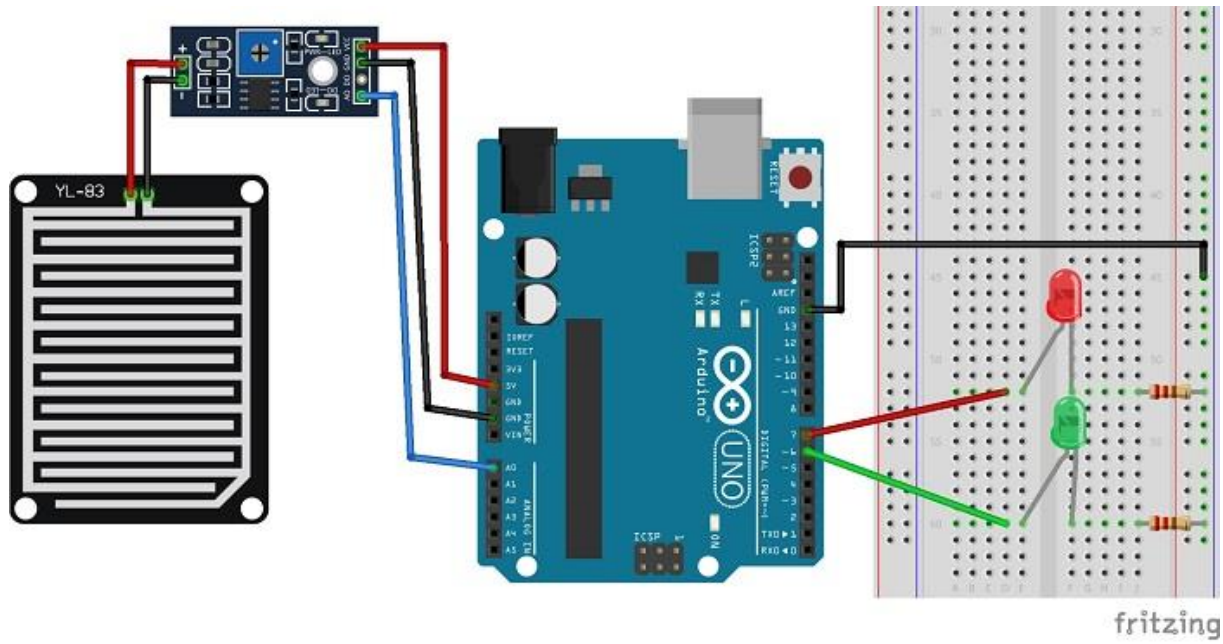


Fig. 33. Connection Diagram for Rain Sensor

Co-funded by the
Erasmus+ Programme
of the European Union

External Temperature Sensor: DS18B20

Just for outdoor use the DS18B20 temperature sensor has to be mounted outside the external box.

DS18B20 is 1-Wire digital temperature sensor from Maxim IC. Reports degrees in Celsius with 9 to 12-bit precision, from -55 to 125 (+/-0.5). Each sensor has a unique 64-Bit Serial number etched into it - allows for a huge number of sensors to be used on one data bus.

Its main features are:

- Unique 1-Wire® interface requires only one port pin for communication
- Each device has a unique 64-bit serial code stored in an onboard ROM
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line.
- Power supply range is 3.0V to 5.5V
- Measures temperatures from –55°C to +125°C (–67°F to +257°F)±0.5°C accuracy from –10°C to +85°C
- Thermometer resolution is user-selectable from 9 to 12 bits
- Converts temperature to 12-bit digital word in 750ms (max.)
- User-definable nonvolatile (NV) alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

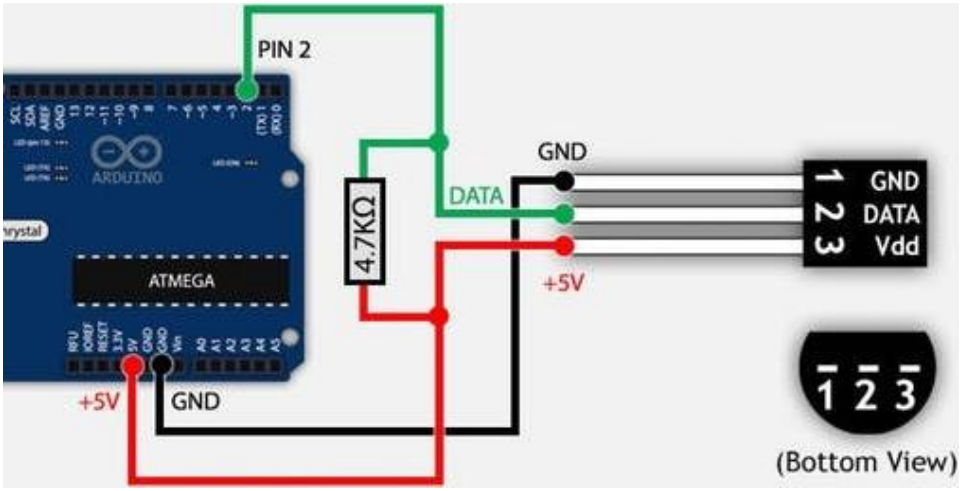


Fig. 34. Connection diagram for temperature sensor DS18B20

Co-funded by the Erasmus+ Programme of the European Union

For our application, we prefer to use a waterproof version of DS18B20 suitable for an installation outside close to the external box.

Fig. 35. The waterproof version for DS18B20 Temp Sensor

For our Kit, we choose to connect the data pin to digital Input 8 of Arduino Mega.

Co-funded by the Erasmus+ Programme of the European Union

Let's start to install sensors

First of all we start to connect the breadboard with GPS shield inside the Internal Box.

Connecting VCC to 5V and GND to GND of Arduino, the GPS shield will turn on. If an external GPS antenna is connected, or the box is close to a window, the PPS led will start to blink.

If we connect a simple led between the PPS pin of GPS shield and GND, in case of satellites in view, this blink will start to blink. To communicate with Arduino, instead, we have to connect the TX and RX pin of GPS to RX1 and TX1 of Arduino to permit the use of SERIAL1 connectivity.

To connect the External Sensor, we will use the Ethernet cable (see below). Thanks to coloured cable, is very easy to connect the right pins to Arduino.

Of course, as for the GPS shield, we connect the VCC and GND pins to bring power supply to the external box.

For the BME280 Sensor, we connect the SCL and SDA pins to correspondent pins on Arduino (pin 20 and 21). (see below)

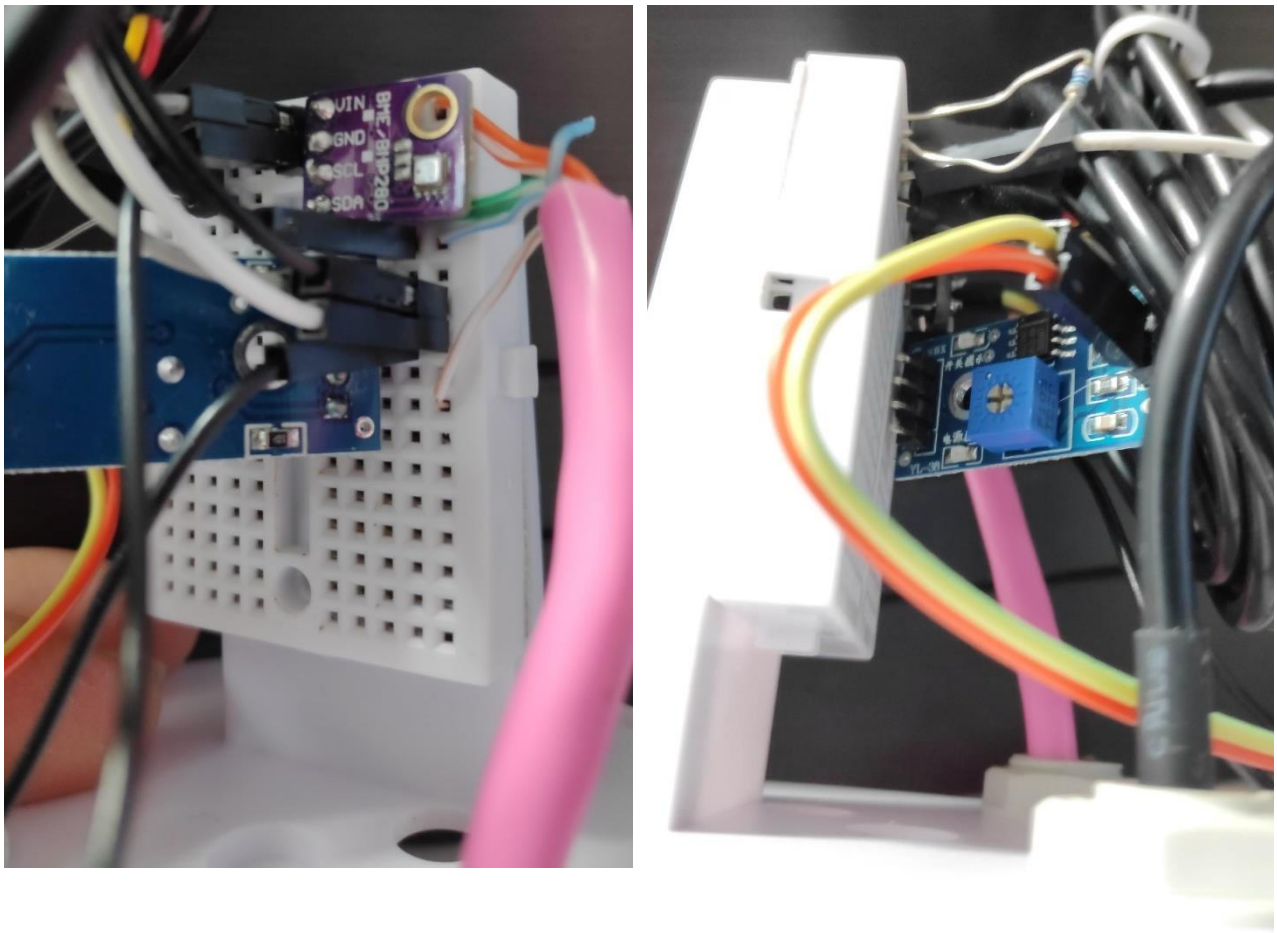


Fig. 36. Pictures for connections inside the External box

Co-funded by the
Erasmus+ Programme
of the European Union

For the Rain Sensor, we connect the Analog output to Analog Input of Arduino (A5) (see below)

Fig. 37. Picture for connections inside the External box

About the DSB18B20, we use another pin of Arduino, the Digital Pin 5, to connect the DATA Pin of the sensor. Warning: Remember to insert a resistor of 4,7kΩ between VCC and DATA pin.

Co-funded by the Erasmus+ Programme of the European Union

The link between Internal and External Box

To connect the external box to the internal one, we prefer to use a simple ethernet cable for these reasons:

1. Cheap and widespread
2. 4 couples of twisted pair cables [colored]

The external box will have two cable glands, one for this ethernet cable and one for the DS18B20 temperature sensor.

To simplify the connections and to avoid mistakes, this guide indicates which colored wires to connect and where.

First of all, we need to cut the ethernet cable in two parts. The shorter one has to be maximum 10 cm long.

This part, in fact, is the cable that will be used inside the internal box.

The rest of the cable will be used instead to connect the external box to the internal one.

Eventually, to cover longer lengths, we suggest using female-female ethernet adapter as shown here:

Fig. 38. Ethernet adapter to extend the cable between the boxes

Co-funded by the Erasmus+ Programme of the European Union

The Code

```
#include <Ethernet.h>
#include <TinyGPS.h>
TinyGPS gps;
#include <SD.h>
#include <Time.h>
#include <TimeLib.h>
#include <EEPROM.h>
#include <Streaming.h>
#include <avr/io.h>
#include <avr/wdt.h>
#define Reset_AVR() wdt_enable(WDTO_30MS); while(1) {}
#include <OneWire.h>
#include <DallasTemperature.h>
#include <EthernetUdp.h>
#include <EnvironmentCalculations.h>
#include <BME280I2C.h>
#include <Wire.h>
#include <SPI.h>
BME280I2C bme;
int sensorRAIN;
unsigned long data,ttime;



File root;



float lat,lon;

String BoolRain;
float temp(NAN), hum(NAN), pres(NAN), dewPoint, Alt, seaLevel;


Sd2Card card;
SdVolume volume;
String readString;
File myFile;
String filename;
const int chipSelect = 4;
char timedata[16];
#define ONE_WIRE_BUS 8
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
int counter;
float temperature = 0;
float humidity = 0;
float Ext_Temp,Int_Temp;
float Pressure = 0;
unsigned long epoch;
int Sync;
String dataString = "";
String LAT,LON;
```


Co-funded by the Erasmus+ Programme of the European Union

```
//################################################ # STATION NAME
################################################
String STATIONNAME = "ITIS LDV";


//################################################ # NETWORK CONFIGURATION -
START ################################################
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 110, 110);
IPAddress gateway(192, 168, 110, 1);
IPAddress subnet(255, 255, 255, 0);
EthernetServer server(80);
//################################################ # NETWORK CONFIGURATION - END
################################################

// NTP Servers:
char timeServer[] = "193.206.115.20"; // timeonewire.nist.gov NTP server
const int timeZone = 0;                // UTC Time
const int NTP_PACKET_SIZE = 48;       // NTP time stamp is in the first
                                       // 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming
                                       //and outgoing packets



void setup() {
   Serial1.begin(9600);          // SERIAL1 to connect gps sensor
   pinMode(13, OUTPUT);
   Serial.begin(115200);
   delay(1000); //Attende 1 secondi
   // Open serial communications and wait for port to open:
   Wire.begin();
   while(!bme.begin())
   {
      Serial.println("**********************************Could not find BME280
sensor!");
      break;
   }
   switch(bme.chipModel())
   {
     case BME280::ChipModel_BME280:
       Serial.println("Found BME280 sensor! Success.");
       break;
     case BME280::ChipModel_BMP280:
       Serial.println("Found BMP280 sensor! No Humidity available.");
       break;
     default:
       Serial.println("Found UNKNOWN sensor! Error!");
   }
  sensors.begin();

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip, gateway, gateway, subnet);
 // server.begin();
   delay(1000);

  Serial.print("My IP Address is: ");
  Serial.println(Ethernet.localIP());

  Serial.print("My Name is: ");
```


Co-funded by the Erasmus+ Programme of the European Union

```
  Serial.println(STATIONNAME);

 // Udp.begin(localPort);
  delay(100);
  Serial.print("\nInitializing SD card...");
  if (!SD.begin(chipSelect)) {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
    digitalWrite(13, HIGH);
    Serial.println("Card failed, or not present");
    // don't do anything more:
    delay(1000);
  }

 delay(2000);
  GPS();



 //#####################################################
 // We prepare the string to print by serial and to save in the file
 //#####################################################

   dataString += String("Sync");
   dataString += "|";
   dataString += String("Time");
   dataString += "|";
   dataString += String("Date");
   dataString += "|";
   dataString += String("Temp");
   dataString += "|";
   dataString +=String("Humidity");
   dataString += "|";
   dataString +=String("DewPoint");
   dataString += "|";
   dataString += String("Ext_Temp");
   dataString += "|";
   dataString += String("Pressure");
   dataString += "|";
   dataString += String("SensRain");
   dataString += "|";
   dataString += String("Rain");
   dataString += "|";
   dataString += String("Lat");
   dataString += "|";
   dataString += String("Lon");
   dataString += "|";
   dataString += String("hdop");
   dataString += "|";
   dataString += String("#Sat");
delay(1000);
digitalWrite(13, LOW);

 //#############################################################
 // if Arduino is not sync by GPS, waiting for a new sync from GPS
 //#############################################################

if (Sync != 0)
{
  delay(2000);
```


Co-funded by the Erasmus+ Programme of the European Union

```
  Serial.println("Waiting for GPS data to open new file");
  GPS();
}
 //##############################################################
 // Create a new file to save data. the file name is automatic generated taking
the epoch time in seconds.
 // In this way, we have a file name made by numbers that increase in time.
 // All this is valid if we have a sync time from GPS.
 //##############################################################

filename=now();
filename=filename.substring(0,8);
filename+=".dat";
 //##############################################################
 // example of file name: "12345678.dat"
 //##############################################################

delay(1000);
Serial.println("");
root = SD.open("/");
Serial.println("done!");

 //##############################################################
 // We open the file ready to write
 //##############################################################

myFile = SD.open(filename, FILE_WRITE);
    if (myFile) {
    Serial.print("Writing to "); Serial.println(filename);
 //##############################################################
 // WRITING DATA IN FILE
 //##############################################################

    myFile.println(dataString);

 //##############################################################
 //##############################################################

 // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(100);
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(100);
    digitalWrite(13, HIGH);// if the file didn't open, print an error:
    Serial.print("error opening file: ");
    Serial.println(filename);
  }



}
unsigned long time;
time_t prevDisplay = 0; //

void loop() {
```


Co-funded by the
Erasmus+ Programme
of the European Union

```
Serial.println(minute());
delay(5000);
if (minute() % 1 == 0)
{
  //  Acquisition Loop 60s  ==> 60000 ms
//#################################################

delay(300000);  // data acquisition every 5 minutes  [5 * 60 * 1000]
//#################################################
String datastring;
String header;
counter++;
if (counter % 5  == 0)
{
    GPS();
}
if (minute() == 0 && hour() == 12)
{
  counter == 0;
  Reset_AVR();
}

sensorRAIN = analogRead(A5);
Serial.print("From Rain Sensor: "); Serial.println(sensorRAIN);

if (sensorRAIN >= 900)
  {
    Serial.println("  (DRY)");
    BoolRain="DRY";
  }
else
  {
    Serial.println("  (WET)");
    BoolRain="WET";
  }

sensors.requestTemperatures(); // Send the command to get temperatures
    /*  Int_Temp = (sensors.getTempCByIndex(0));
    */
Ext_Temp = (sensors.getTempCByIndex(0));
Serial.print("======================");
Serial.println();
Serial.print("From ds18b20 Ext: ");
Serial.print(Ext_Temp);
Serial.println(" *C");
Serial.println();
Serial.print("From ds18b20 Int: ");
Serial.print(Int_Temp);
Serial.println(" *C");

Serial.print("From BME: ");
printBME280Data(&Serial);

header = "\n";
header += String("Sync");
header += "|";
header += String("Time");
header += "|";
header += String("Date");
header += "|";
header += String("Temp");
header += "|";
```


Co-funded by the
Erasmus+ Programme
of the European Union

```
header +=String("Humidity");
header += "|";
header +=String("DewPoint");
header += "|";
header += String("Ext_Temp");
header += "|";
header += String("Pressure");
header += "|";
header += String("SensRain");
header += "|";
header += String("Rain");
header += "|";
header += String("Lat");
header += "|";
header += String("Lon");
header += "|";
header += String("hdop");
header += "|";
header += String("#Sat");

dataString = "\n";
dataString += Sync;
dataString += "|";
dataString += String(hour())+":"+String(minute())+":"+String(second());
dataString += "|";
dataString += String(month())+"/"+String(day())+"/"+String(year());
dataString += "|";
dataString += temp;
dataString += "|";
dataString += hum;
dataString += "|";
dataString += dewPoint;
dataString += "|";
dataString += Ext_Temp;
dataString += "|";
dataString += pres;
dataString += "|";
dataString += sensorRAIN;
dataString += "|";
dataString += BoolRain;
dataString += "|";
dataString += LAT;
dataString += "|";
dataString += LON;
dataString += "|";
dataString += gps.hdop();
dataString += "|";
dataString += gps.satellites();
dataString += "||";
Serial.println(header + dataString);

myFile = SD.open(filename, FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to ");Serial.println(filename);
    myFile.println(dataString);
    // close the file:
    myFile.close();
    Serial.println("done.");
    }
    else
    {
      digitalWrite(13, HIGH);
```


Co-funded by the Erasmus+ Programme of the European Union

```
      delay(500);
      digitalWrite(13, LOW);
      delay(100);
      digitalWrite(13, HIGH);
      delay(500);
      digitalWrite(13, LOW);
      delay(100);
      digitalWrite(13, HIGH);// if the file didn't open, print an error:
      Serial.print("error opening file: ");
      Serial.println(filename);
    }
    // listen for incoming clients
    EthernetClient client = server.available();
    if (client)
    {
      Serial.println("new client");
      boolean currentLineIsBlank = true;
      while (client.connected()) {
        if (client.available()) {
        char c = client.read();

          if (c == '\n') {
          // send a standard http response header


            if (readString.length() < 100) {
              //store characters to string
              readString += c;
              //Serial.print(c);
              }

          }

        }

        if (client.available()) {
          char c = client.read();
          //read char by char HTTP request
          if (readString.length() < 100) {
            //store characters to string
            readString += c;
            //Serial.print(c);
          }
        //if HTTP request has ended
          if (c == '\n' && currentLineIsBlank) {
            client.println("HTTP/1.1 200 OK"); //send new page
            client.println("Content-Type: text/html");
            client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          //client.print("<img src='http://www.alfadetectives.it/wp-
content/uploads/2015/12/servizio-gps-tracking.jpg' width='200'>");
          client.println("<center><font size=5
color='blue'>"+STATIONNAME+"</font></center><br>");
          client.println("Station: "+STATIONNAME +"<br>");
          client.print("Date - Time:    ");
          client.print(String(month())+"/"+String(day())+"/"+String(year()));
client.println(" - ");
client.println(String(hour())+":"+String(minute())+":"+String(second()));
          client.println("<br>");
          client.println("<font size='4' color='blue'>External Box</font><br>");
```


Co-funded by the
Erasmus+ Programme
of the European Union

```
          client.print("External Temperature:  "); client.print(Ext_Temp);
client.println(" *C<br>");
          client.print("Rain Sensor:  ");
          if (sensorRAIN >= 900)
          {
            client.println("DRY<br>");
          }
          else
          {
            client.println("WET<br>");
          }
          client.print("Internal Temperature: ");client.print(temp);
client.println(" *C<br>");
          client.print("Humidity:     ");client.print(hum); client.println("
%RH<br>");
          client.print("Pressure:     ");client.print(pres); client.println("
Pa<br>");
          client.print("DewPoint      ");client.print(dewPoint);
client.println(" *C<br>");
          client.print("Equivalent Sea Level Pressure:
");client.print(seaLevel); client.println(" Pa<br>");

          client.print("GPS Location: ");client.print("<a
href='https://www.google.it/maps/place/");client.print(lat,6); client.print("
N,"); client.println(lon,6); client.print("
E");client.print("'>");client.print(lat,6); client.print(" N,");
client.println(lon,6); client.print(" E");client.print("</a><br>");
          client.print("Altitude:
");client.print(gps.f_altitude());client.println(" m<br>");
          client.print("HDOP:
");client.print(gps.hdop());client.println("<br>");
          client.print("File Name:   ");client.println(filename);
          client.print("<br>");
          delay(10);
          //stopping client
          client.stop();
          readString="";
        }
      }
    }
    Serial.println("client disconnected");
  }
}

}

// Print an integer in "00" format (with leading zero),
// followed by a delimiter character to Serial.
// Input value assumed to be between 0 and 99.
void printI00(int val, char delim)
{
    if (val < 10) Serial << '0';
    Serial << _DEC(val);
    if (delim > 0) Serial << delim;
    return;
}



void printBME280Data
(
   Stream* client
```


Co-funded by the
Erasmus+ Programme
of the European Union

```
)
{

   BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
   BME280::PresUnit presUnit(BME280::PresUnit_Pa);

   bme.read(pres, temp, hum, tempUnit, presUnit);

   client->print("Temp: ");
   client->print(temp);
   client->print(" *C");
   client->print("\tHumidity: ");
   client->print(hum);
   client->print(" %RH");
   client->print("\tPressure: ");
   client->print(pres);
   Pressure=(pres); //
   client->print(" Pa");

   EnvironmentCalculations::AltitudeUnit envAltUnit  =
EnvironmentCalculations::AltitudeUnit_Meters;
   EnvironmentCalculations::TempUnit     envTempUnit =
EnvironmentCalculations::TempUnit_Celsius;

   Alt = EnvironmentCalculations::Altitude(pres, envAltUnit);
   dewPoint = EnvironmentCalculations::DewPoint(temp, hum, envTempUnit);
   seaLevel = EnvironmentCalculations::EquivalentSeaLevelPressure(Alt, temp,
pres);
   // seaLevel = EnvironmentCalculations::SealevelAlitude(altitude, temp, pres);
// Deprecated. See EquivalentSeaLevelPressure().

   client->print("\t\tAltitude: ");
   client->print(Alt);
   client->print(" m");
   client->print("\tDew point: ");
   client->print(dewPoint);
   client->print(" *C");
   client->print("\t Equivalent Sea Level Pressure: ");
   client->print(seaLevel);
   client->println(" Pa");
}

//////////////////////////////////////
//////////////////////////////////////

float julianDay()
{
  int a = year()/100;
  int b = a/4;
  int c = 2-a+b;
  long e = 365.25 * (year() + 4716);
  long f = 30.6001 * (month()+1);
  return (float)c + day() + (float)e + (float)f - 1524.5 + julianDayFraction();
}

float julianDayFraction()
{
  return ((float)(hour())/24.0f) + ((float)minute()/1440.0f) +
((float)second()/86400.0f);
}

unsigned long julianDayFractionAsLong(){
```


Co-funded by the Erasmus+ Programme of the European Union

```
  return (unsigned long)((julianDayFraction() + .5) * 10000.0f) % 10000;
}
////////////////////////////////////
////////////////////////////////////

void GPS()
{
  Serial1.end();
  delay(1000);
  int counterGPS = 0;
  Serial1.begin(9600); // connect gps sensor
  Serial.println();
  Serial.println("GPS Connection Opening...");
  delay(1000);
  while(1)
  {
   // Serial.print(".");
    if (Serial1.available())
    {
      // Serial.print("GPS Available");
      if (gps.encode(Serial1.read()))
      {
        gps.f_get_position(&lat,&lon);
        LAT=String(lat,6);
        LON=String(lon,6);
        gps.get_datetime(&data,&ttime);
        gps.hdop();
        unsigned long age;
        int Year;
        byte Month, Day, Hour, Minute, Second;
        gps.crack_datetime(&Year, &Month, &Day, &Hour, &Minute, &Second, NULL,
&age);
        if (age < 500) {
      // set the Time to the latest GPS reading
          setTime(Hour, Minute, Second, Day, Month, Year);
      // adjustTime(offset * SECS_PER_HOUR);
        }

        Sync = 1;

        if (lat!=0.00 && gps.f_altitude() < 50000.00)
        {
          Serial.println("----------------------");
          Serial.print("GPS: ");
      // Latitude
          Serial.print("Lat: ");
          Serial.print(lat,6);
          Serial.print(",");
      // Longitude
          Serial.print("Lon: ");
          Serial.print(lon,6);
          Serial.print(",");
      // Serial.print(gps.altitude());
          Serial.print("Alt: ");
          Serial.print(gps.f_altitude());
          Serial.print(",");
          Serial.print(data);
          Serial.print(",");
          Serial.print(ttime);
          Serial.print(",");
          Serial.print(gps.satellites());
          Serial.print(",");
```


Co-funded by the
Erasmus+ Programme
of the European Union

```
            Serial.println(gps.hdop());
            Serial.println("GPS Connection Closed");
            break;
          }
        }
        else
        counterGPS++;
        if (counterGPS >= 10000)
        {
          Serial.println("GPS Data NOT AVAILABLE");
          Sync = 0;
          Serial.println(" ****************    W A R N I N G - Check GPS
Connection ");
          break;
        }
      }
    }
}
```

Data Dissemination

All data acquired will be publish on a web page.

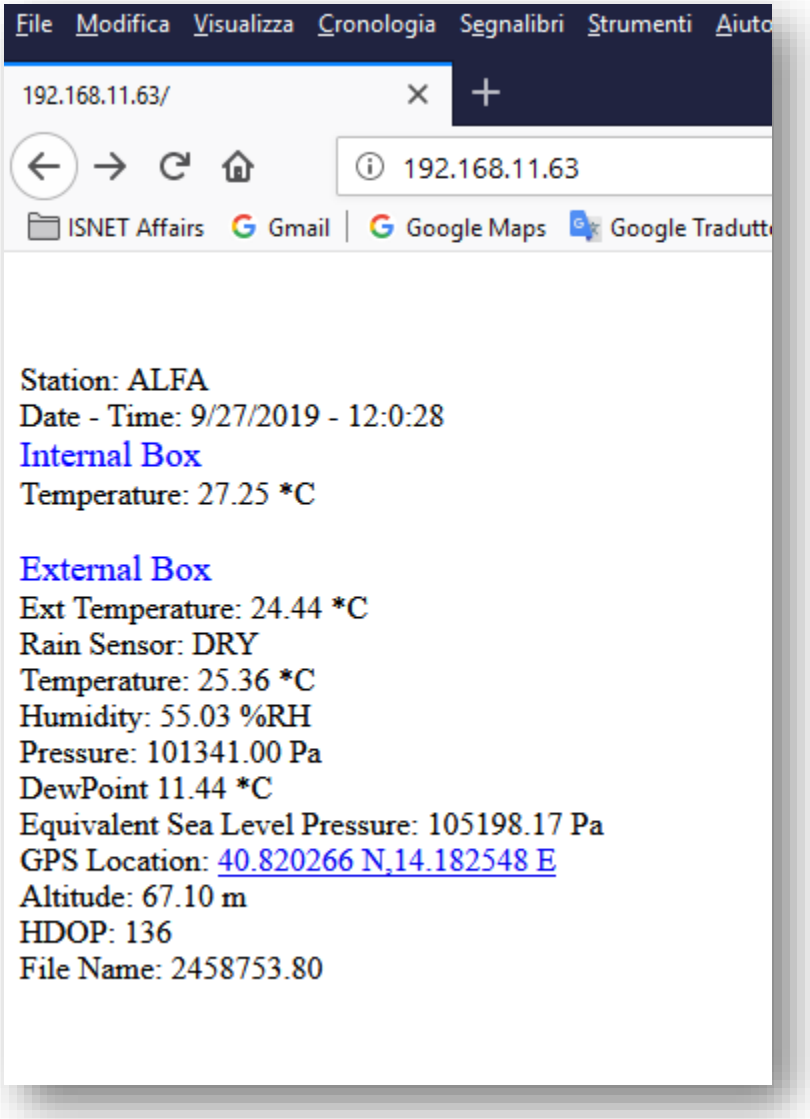
At the link http://192.168.11.65 (ip address used for this example) you'll find:

- Name of the KIT
- Information about date and time (if GPS antenna is connected)
- Temperature outside the box
- Information about rain
- Temperature inside the box
- Humidity
- Pressure
- Equivalent Sea Level Pressure
- GPS Location [Latitude and Longitude]
- Altitude
- HDOP [horizontal error of GPS Location]
- File Name where data are logged

Every 15 seconds, this data is updated and available.

Co-funded by the Erasmus+ Programme of the European Union

and now have fun to invent and expand the project!



Co-funded by the Erasmus+ Programme of the European Union