Universidad de los Andes
Colombia

Levels of Automation Laboratories Application Guide

Giacomo Barbieri

Adriana España

David Sanchez-Londono

Universidad de los Andes, Department of Mechanical Engineering
{g.barbieri, a.espana, d.sanchezl}@uniandes.edu.co

v1.0
July 2022

Contents

1 Preface

1.1 Guidelines and Suggestions

The following are a set of suggestions for instructors who wish to use these laboratories in their own classroom.

- *Materials*: Each student must have access to the materials indicated in the "Materials'' section below. If working remotely, they should have access to the materials with sufficient anticipation to prepare.
- *Preparation*: The kit must work properly for each student before the laboratories. This is especially important if working remotely, as the students themselves must set up the necessary software on their computers.
- *Group work*: These laboratories work best when students can work in groups. It is useful to have one instructor or moderator assigned to each group, with the goal of guiding the activities and providing technical support.

1.2 Materials

The materials that a student needs for these laboratories are now mentioned. Ideally, each student will have all these materials; if working remotely, a set of materials (a ``kit'') can be sent to each student.

- *Controller*: a *NodeMCU* ESP8266 is used in the tutorial and activities here. Any Arduino-compatible single-board microcontroller should work, with minor tweaks to the code.
- *Sensor*: a DHT11 temperature and relative humidity sensor. Any sensor that can read temperature and relative humidity sensor should work, with minor tweaks to the code.
- *Actuator*: a ROB-08449 vibration motor. Any actuator that can be powered by the NodeMCU (an LED, a low-power dc motor, etc.) should work, with minor tweaks to the code.
- *Connections*: all the cables necessary to a) connect the sensor and actuator to the controller, and b) connect the controller to a computer and to a power source. At least two *alligator clip wires*, two *male to female jumper wires*, and three *female to female jumper* wires are required.
- *Computer*: the laboratories require access to a computer that can run the Arduino IDE software. These include Windows, MacOS, and Linux devices.

2 LoA Laboratories: Preparation

2.1 Software and Hardware Configuration

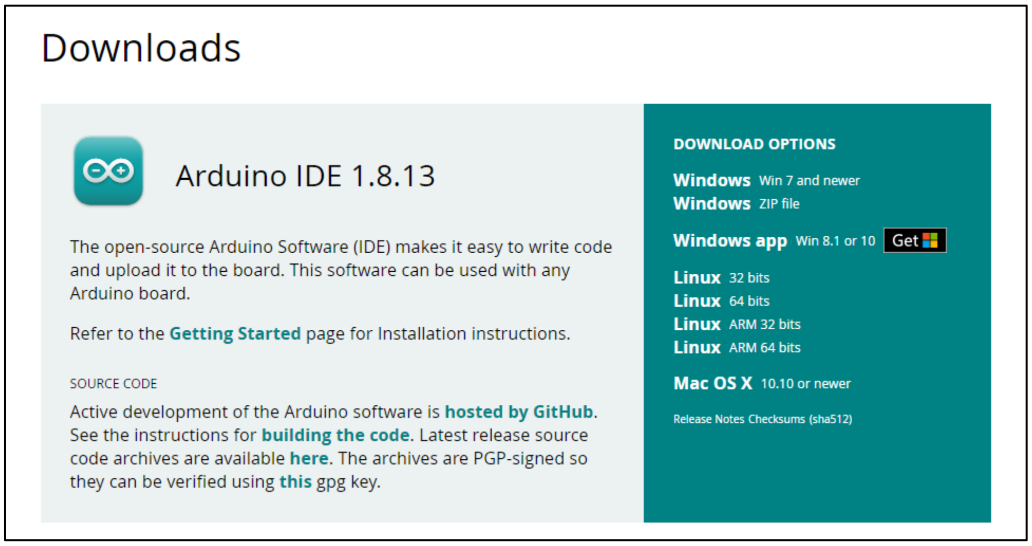
To complete the Levels of automation laboratories in this document, you must first configure the NodeMCU you were given. This configuration has a *hardware* and a *software* component. Hardware refers to the **physical components** of a computer or a system. Software refers to the programs, information, procedures or guidelines involved in a system: everything that is **intangible** in a system is software.

2.1.1 Software Configuration

Download and install the Arduino IDE (Integrated Development Environment):

https://www.arduino.cc/en/software

You will choose one of the download files depending on your Operating System:



Downloads

Arduino IDE 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

SOURCE CODE

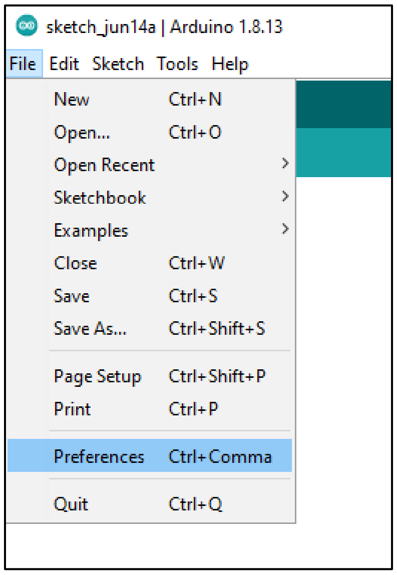
Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer

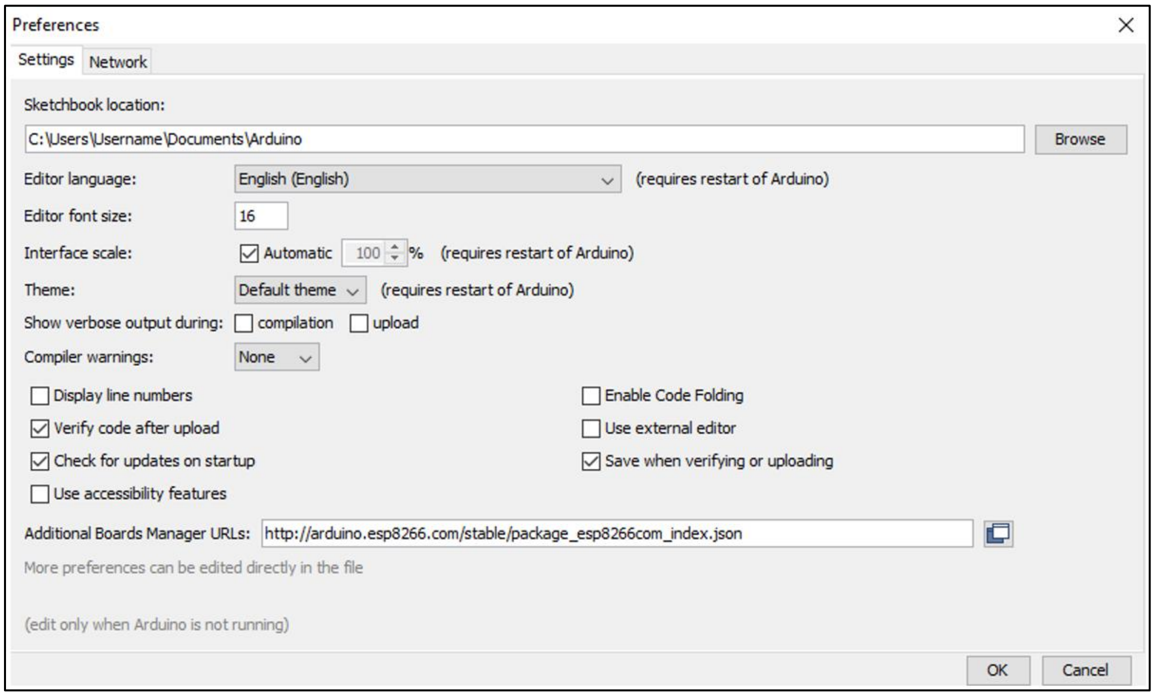
Release Notes Checksums (sha512)

Once Arduino is installed, open it. Then go to **Files -> Preferences**.



A new window will open. Find the "Additional Boards Manager URLs" item, and paste the following URL to this empty cell:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Sketchbook location: C:\Users\Username\Documents\Arduino

Editor language: English (English) (requires restart of Arduino)

Editor font size: 16

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

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

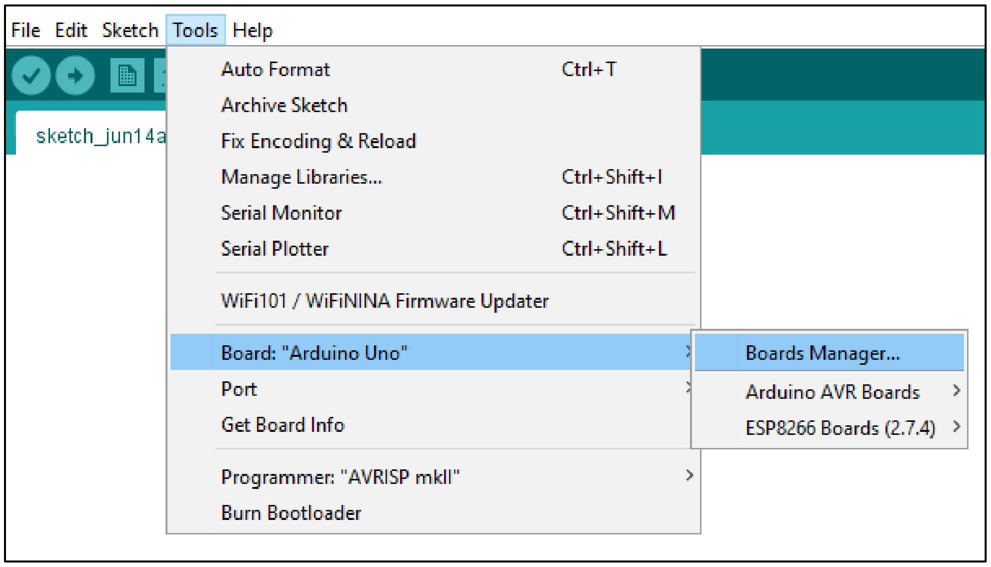
[ ] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

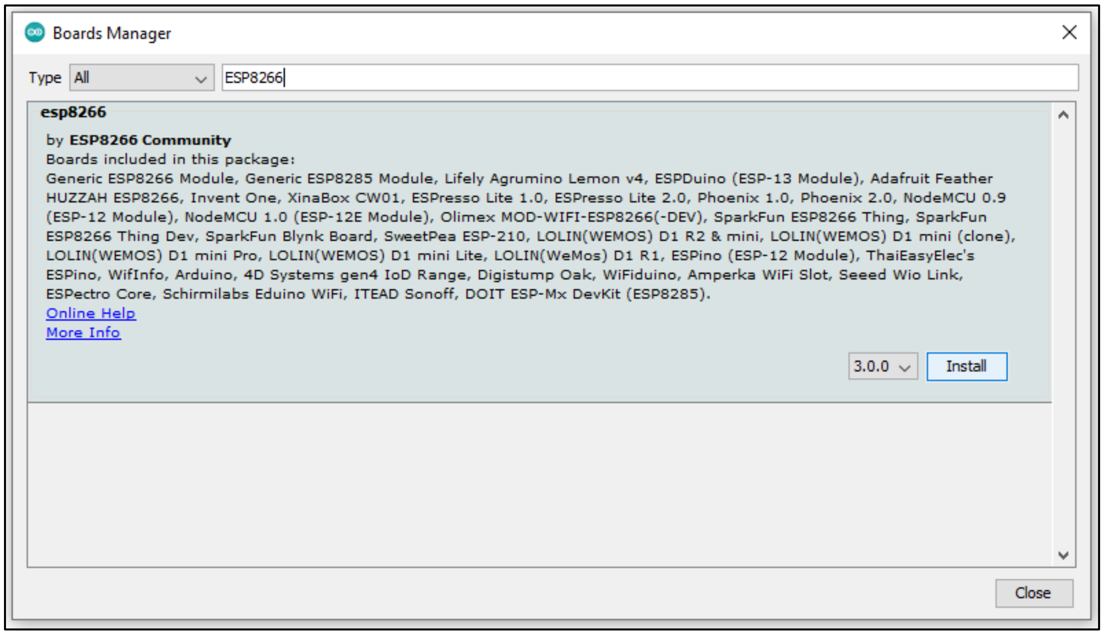
More preferences can be edited directly in the file

(edit only when Arduino is not running)

Click OK, and open **Tools -> Board -> Boards Manager…**

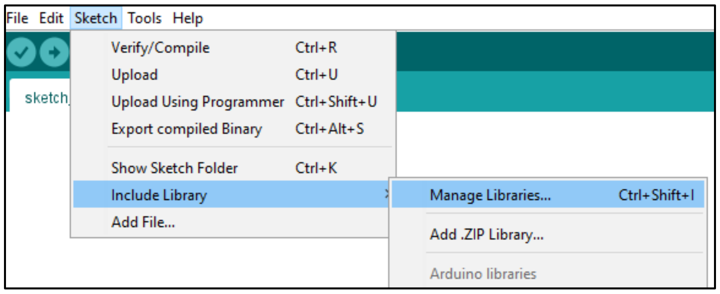


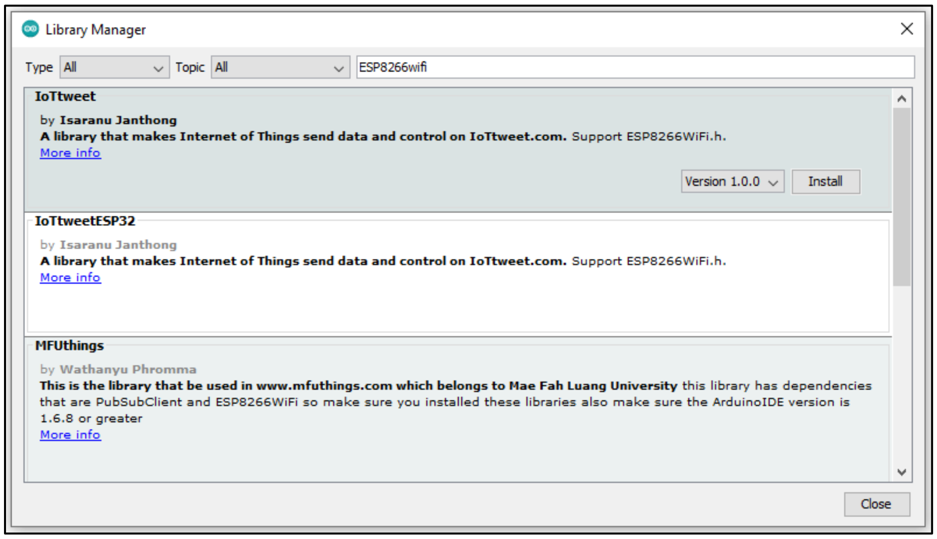
Type the keyword **ESP8266** in the search box, then install the ESP8266 community package by clicking on the **Install** button in the lower right corner.



esp8266
by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, Lifely Agrumino Lemon v4, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini (clone), LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, LOLIN(WeMos) D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
Online Help
More Info

Now open **Sketch -> Include Library -> Add .ZIP Library...** Type the keyword **ESP8266wifi** in the search box, then install the ESP8266 community package.


File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
sketch

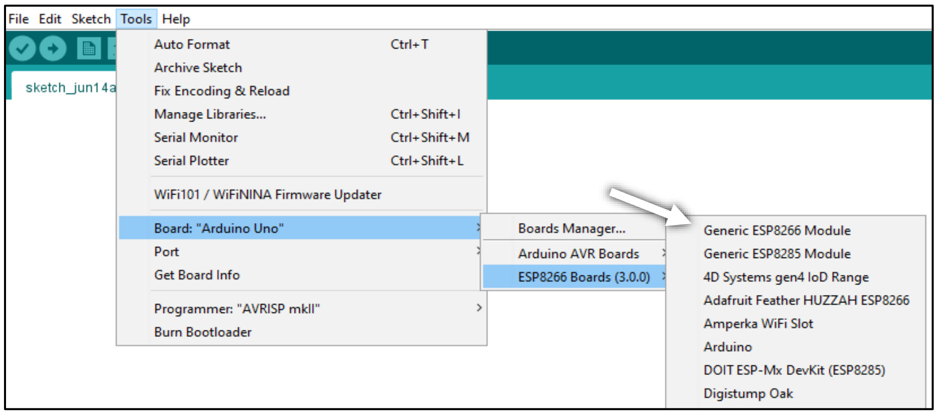


IoTtweet
by **Isaranu Janthong**
A library that makes Internet of Things send data and control on IoTtweet.com. Support ESP8266WiFi.h.
More info

IoTtweetESP32
by **Isaranu Janthong**
A library that makes Internet of Things send data and control on IoTtweet.com. Support ESP8266WiFi.h.
More info

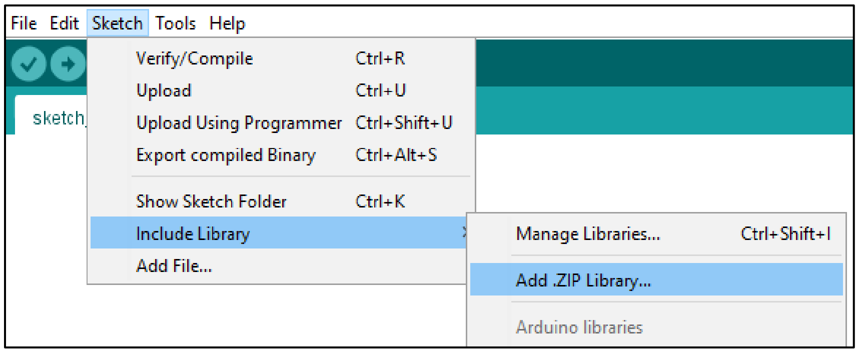
MFUthings
by **Wathanyu Phromma**
This is the library that be used in www.mfuthings.com which belongs to Mae Fah Luang University this library has dependencies that are PubSubClient and ESP8266WiFi so make sure you installed these libraries also make sure the ArduinoIDE version is 1.6.8 or greater
More info

After installing the libraries, click on **Tools -> Board -> ESP8266 Boards -> Generic ESP8266 Module**. This will set the NodeMCU as the device you are currently working with.


File Edit Sketch Tools Help
sketch_jun14a
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 Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP8266 Boards (3.0.0)
Generic ESP8266 Module
Generic ESP8285 Module
4D Systems gen4 IoD Range
Adafruit Feather HUZZAH ESP8266
Amperka WiFi Slot
Arduino
DOIT ESP-Mx DevKit (ESP8285)
Digistump Oak

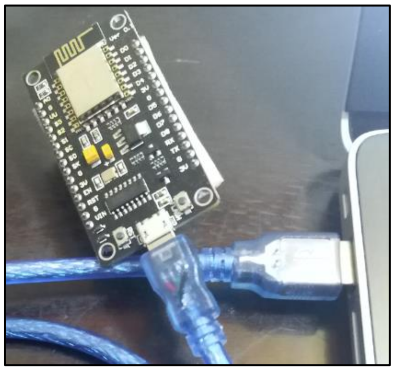
The last step will be to install two libraries manually. First, download the *Adafruit_Sensor-master.zip* and *DHT_sensor_library.zip* files from the **LoA Laboratories Github page**[1], in the **libraries** directory. Then, click on **Sketch -> Include Library -> Add .ZIP Library...** and select the two zip files.

[1] https://github.com/d-sanchezl/levels-of-automation-laboratories

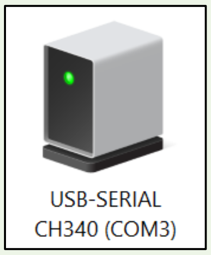

File Edit Sketch Tools Help
sketch_
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries

2.1.2 Hardware Configuration

After the software is installed, connect the NodeMCU controller to your computer using a Micro USB to USB cable.



Windows users: you will now verify if this hardware was correctly identified by your computer. For Windows computers, open the Control Panel -> Hardware and Sound -> Device and Printers. You should see an object called "**USB-SERIAL CH340 (COM3)**".



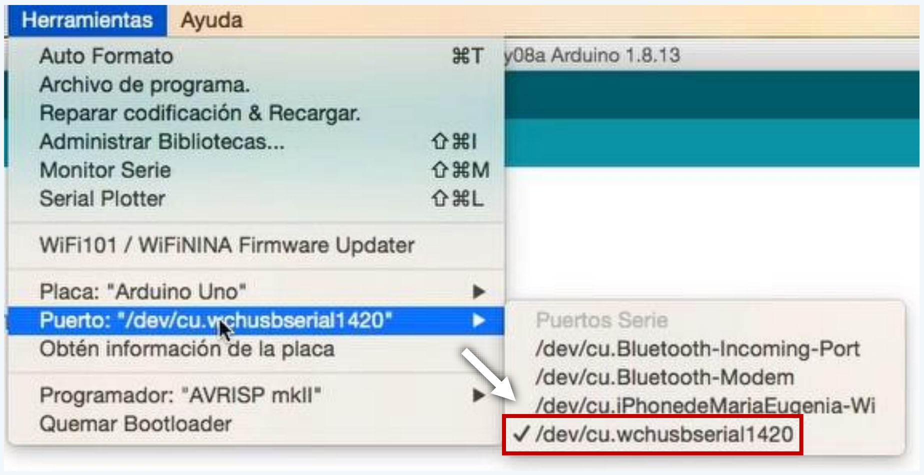
The COM may have a different number (COM2, COM7...). However, if the object does not have the **CH340** label, you must manually install the proper driver. Go to the following webpage and download the *Windows CH340 Driver*, then install it:

https://sparks.gogo.co.nz/ch340.html

Mac users: you will now verify if this hardware was correctly identified by your computer. In the Arduino IDE, open **Tools -> Port**, and verify that one of the ports includes the name:

.wchusbserial

An example of a proper port might be "/dev/cu.wchusbserial1420":


Herramientas
Ayuda
Auto Formato ⌘T
Archivo de programa.
Reparar codificación & Recargar.
Administrar Bibliotecas... ⇧⌘I
Monitor Serie ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 / WiFiNINA Firmware Updater
Placa: "Arduino Uno"
Puerto: "/dev/cu.wchusbserial1420"
Obtén información de la placa
Programador: "AVRISP mkII"
Quemar Bootloader
y08a Arduino 1.8.13
Puertos Serie
/dev/cu.Bluetooth-Incoming-Port
/dev/cu.Bluetooth-Modem
/dev/cu.iPhonedeMariaEugenia-Wi
✓ /dev/cu.wchusbserial1420

If such a port does not appear, you must manually install the proper driver. Go to the following webpage and download the *V1.5 CH340 MacOS Driver Pkg*, then install it:

https://sparks.gogo.co.nz/ch340.html

3 Fully Manual Control Laboratory

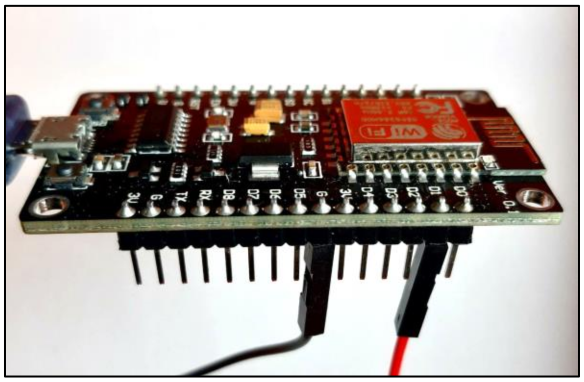
On this Laboratory, you will learn how to manually control an actuator **locally** and **remotely**. You will activate or deactivate the actuator depending on the readings from a sensor – in a sense, you as an operator will provide **intelligence** to the system.

3.1 Local Manual Control

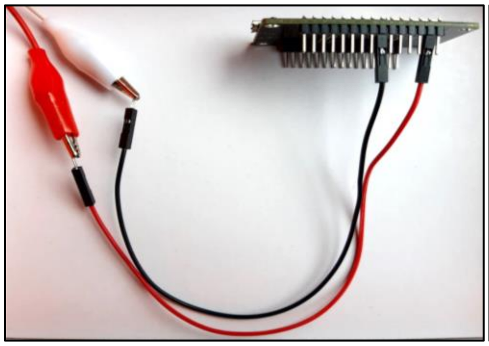
You will first connect both the *actuator* (a ROB-08449 vibration motor) and the *sensor* (a DHT11 temperature and relative humidity sensor) to the *controller* (a NodeMCU ESP8266). Then, you will be able to manually control the actuator, along reading sensor values.

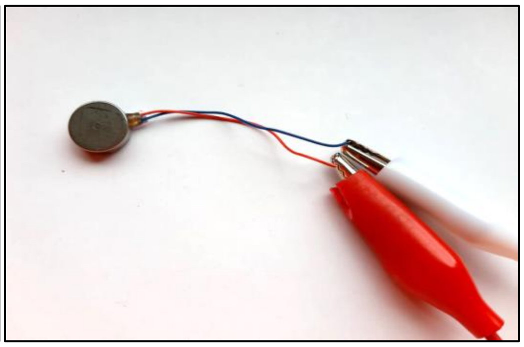
3.1.1 Hardware Configuration

First, connect a (preferably red) jumper cable to the **D1** connector of the NodeMCU, then connect a (preferably black) jumper to a **G** connector.

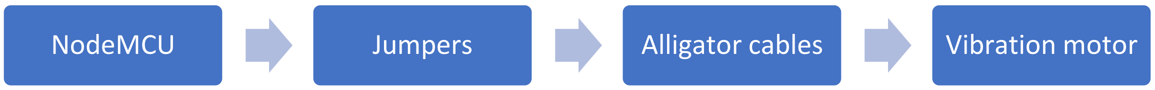


Now, use the alligator clip wires to connect the ROB-08449 vibration motor to the jumpers.



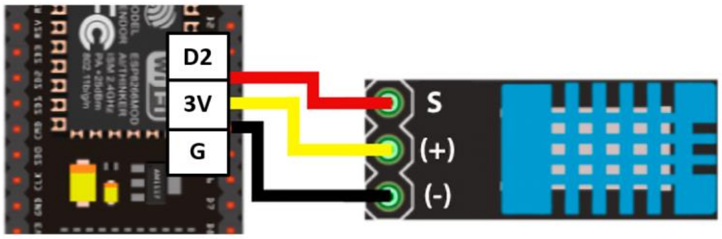


The connection diagram of this setup would look like this:

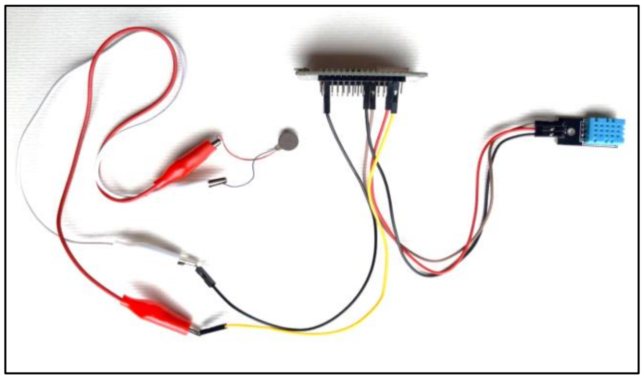


Now, use any three female to female jumpers to connect the DHT11 temperature and relative humidity sensor to the NodeMCU:

- Connect the S terminal of the sensor to the **D2** pin of the controller.
- Connect the positive (+) terminal of the sensor to the **3V** pin of the controller.
- Connect the negative (-) terminal of the sensor to a **G** pin of the controller.



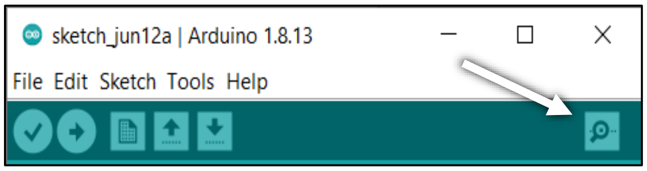
After both the actuator and sensor are connected, the setup should look like this:



Now, connect the NodeMCU to your computer if it is not connected.

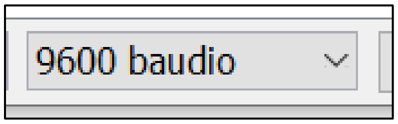
3.1.2 Software Configuration

Open the *Fully-Manual-Local.ino* file from the **LoA Laboratories Github page**, in the **arduino code files** directory. If you get a message asking you to create a new folder, accept and continue. Open the Serial Monitor by clicking on the magnifying glass icon:



If pressing the icon returns an Arduino error, make sure that you have selected the proper COM port through **Tools -> Port**.

After you open the Serial Monitor window, look for the option to change the baud rate in the bottom part of the window. **Select a baud rate of 9600**.

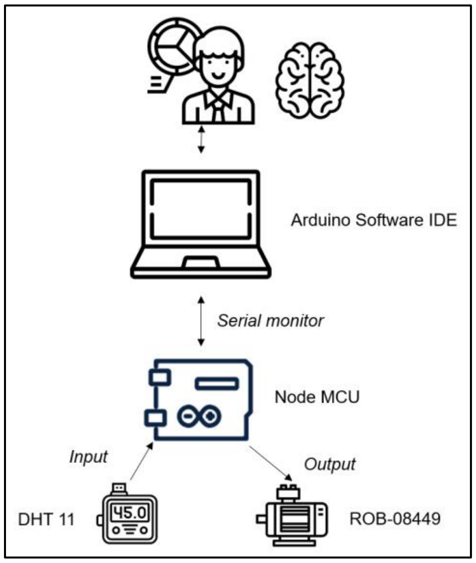

9600 baudio

If everything was setup correctly, you should now be in control of the vibration motor. You may **type 1** in the serial monitor to **activate it**, and **type 2 to deactivate it**.

Notice how this same serial monitor displays sensor values for both **temperature** and **humidity**. You may try covering the sensor with your hands to raise its temperature. Alternatively, try exhaling over it to increase the relative humidity.

3.1.3 Reflection

In this first part of the laboratory, you were able to control an actuator and read data from a sensor through an electronic controller. The controller does not operate the actuator by itself – rather, a human oversees the actuation. The following figure shows the architecture of the system you just worked with – note how the **intelligence** (shown as a brain) lies in the operator. Therefore, this first laboratory reflects Industry 2.0: **electrification** allows for the **fully manual control** of a system.



3.2 Remote Manual Control

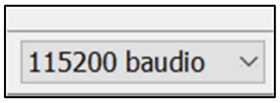
For the second part of this laboratory, you will use a system with the same control capabilities as before: you will be able to manually control the actuator, along reading sensor values. This time however, the controller will not be connected directly to a computer: it will operate independently through a **wireless local area network** (WiFi).

3.2.1 Software Configuration

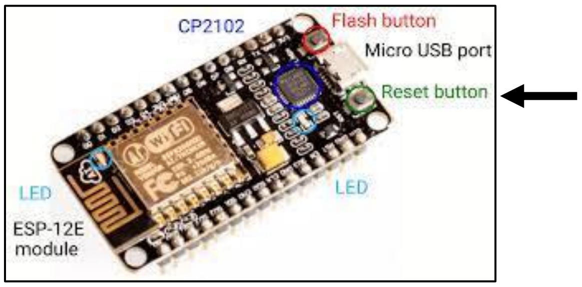
Open the *Fully-Manual-Remote.ino* file from the **LoA Laboratories Github page**, in the **arduino code files** directory. If you get a message asking you to create a new folder, accept and continue. Within the code, enter the SSID (network name) and password for your WiFi network[2].

```
const char* ssid = "network.name";
const char* password = "password1234";
```

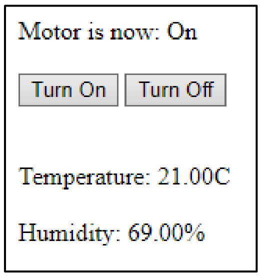
Open the Serial Monitor by clicking on the magnifying glass icon. After you open the Serial Monitor window, look for the option to change the baud rate in the bottom part of the window. **Select a baud rate of 115200**.


115200 baudio

Now, press the *reset button* in the NodeMCU:



An IP address will now appear on the serial monitor. Copy this IP address and paste it into the URL bar of your web browser (Chrome, Firefox, Safari…). If everything was setup correctly, you should now see an interface like this one:


Motor is now: On

Temperature: 21.00C

Humidity: 69.00%

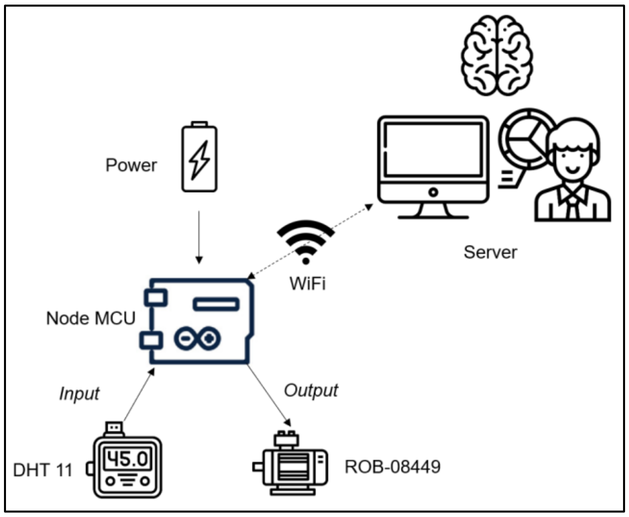
Press the two buttons to turn the vibration motor on and off. Like in the previous exercise, try covering the sensor or exhaling onto it to alter the sensor values (*please refresh your browser to update the displayed sensor value*). Note that the NodeMCU is communicating

[2] Note that the NodeMCU controller only works with 2.4Ghz networks, and is not compatible with 5.0GHz networks. If you are unsure about your network, it is probably 2.4GHz.

to the web browser through the network, not through the USB cable. To test this, **disconnect** the NodeMCU from your computer, and connect it to a **standard USB charger**. Press the reset button on the controller and try reloading the web browser. Both the sensor and actuator should work.

3.2.2 Reflection

In this second part of the laboratory, you were able to control an actuator and read data from a sensor like you did before, but in a remote manner. As the next figure shows, the connection between the operator and the controller was **wireless**, which required the controller to have an **external power source**. A human can now oversee the system from afar; this would allow for a person to oversee multiple devices from a single location. Note that both the local and the remote version of this laboratory operate on an **open loop**: the sensor values are not utilized from the controller to change the actuator condition.



4 Fully Automatic Control Laboratory

Universidad de los Andes
Department of Mechanical Engineering

On this Laboratory, you will setup a controller that will control an actuator **locally** and **remotely** automatically, depending on sensor values. The controller itself will contain the **intelligence** of the system.

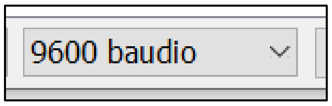
4.1 Local Automatic Control

First, connect both the *actuator* (a ROB-08449 vibration motor) and the *sensor* (a DHT11 temperature and relative humidity sensor) to the *controller* (a NodeMCU ESP8266) like you did in the previous laboratory. Make sure that the NodeMCU is connected to your computer.

4.1.1 Software Configuration

Open the *Fully-Automatic-Local.ino* file from the **LoA Laboratories Github page**, in the **arduino code files** directory. If you get a message asking you to create a new folder, accept and continue. Open the Serial Monitor by clicking on the magnifying glass icon.

After you open the Serial Monitor window, look for the option to change the baud rate in the bottom part of the window. **Select a baud rate of 9600**.

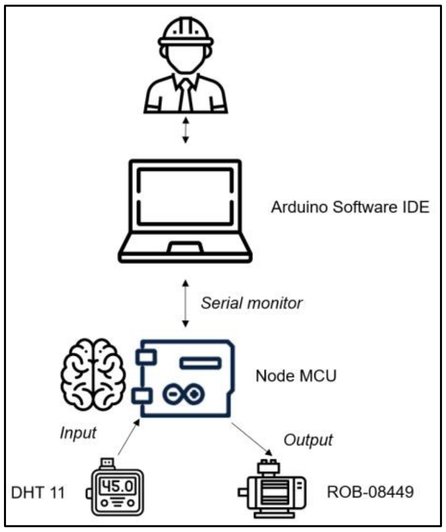


9600 baudio

This time, you will not control the actuator directly. If everything was setup correctly, the vibration motor will turn on automatically if it detects a **relative humidity value over 90%.** You may try covering the sensor with your hands to raise the relative humidity. Alternatively, try exhaling over it. Observe how the actuator turns on and off depending on the humidity value.

4.1.2 Reflection

In this second part of the laboratory, you observed how the controller decided when to turn on and off the actuator automatically. The state of the actuator depends on the sensor readings: the operator then takes the role of an observer. The following figure shows the architecture of the system you just worked with – note how the **intelligence** (shown as a brain) is not on the operator anymore. Instead, the intelligence lies in the controller, which makes decisions in the system. Therefore, this second laboratory reflects Industry 3.0: **digitalization** allows for the **fully automatic control** of a system.



4.2 Remote Automatic Control

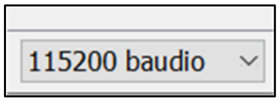
For the second part of this laboratory, you will use a system with the same control capabilities as before: the controller will decide the state of the actuator. This time however, the controller will not be connected directly to a computer: it will operate independently through a **wireless local area network** (WiFi).

4.2.1 Software Configuration

Open the *Fully-Automatic-Remote.ino* file from the **LoA Laboratories Github page**, in the **arduino code files** directory. If you get a message asking you to create a new folder, accept and continue. Within the code, enter the SSID (network name) and password for your WiFi network[3].

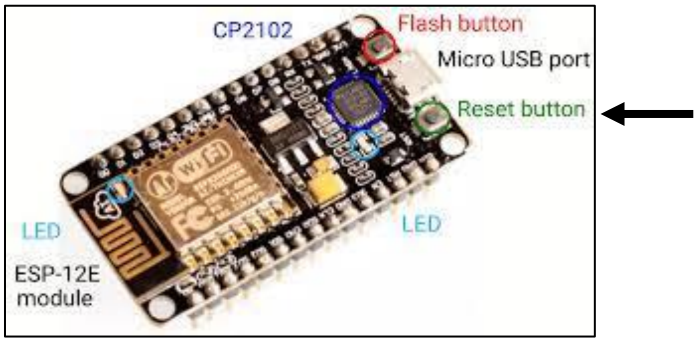
```
const char* ssid = "network.name";
const char* password = "password1234";
```

Open the Serial Monitor by clicking on the magnifying glass icon. After you open the Serial Monitor window, look for the option to change the baud rate in the bottom part of the window. **Select a baud rate of 115200**.

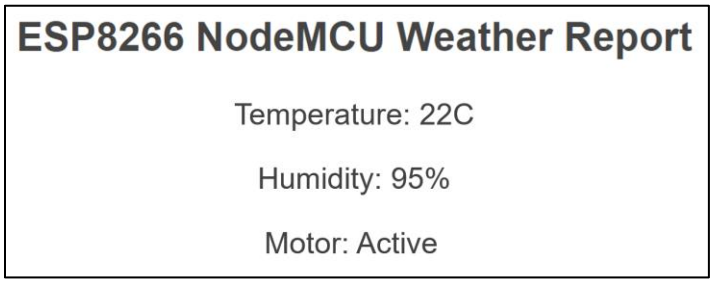

115200 baudio

[3] Note that the NodeMCU controller only works with 2.4Ghz networks, and is not compatible with 5.0GHz networks. If you are unsure about your network, it is probably 2.4GHz.

Now, press the *reset button* in the NodeMCU:



An IP address will now appear on the serial monitor. Copy this IP address and paste it into the URL bar of your web browser (Chrome, Firefox, Safari…). If everything was setup correctly, you should now see an interface like this one:


ESP8266 NodeMCU Weather Report

Temperature: 22C

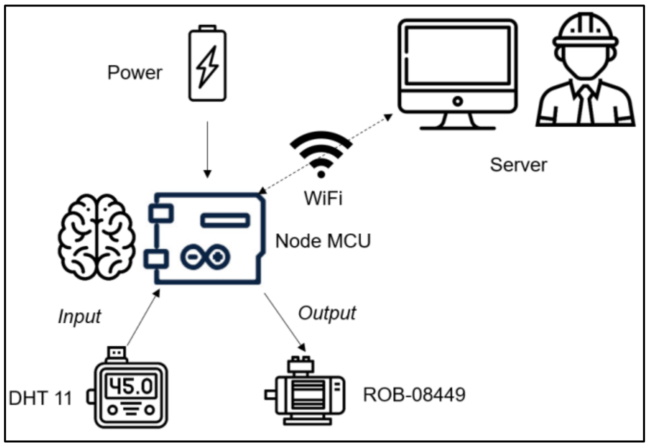
Humidity: 95%

Motor: Active

Since the controller decides when to activate the actuator, this interface only has supervision purposes. Like in the previous exercise, try covering the sensor or exhaling onto it to alter the sensor values (*please refresh your browser to update the displayed sensor value*). To test the remote connectivity, **disconnect** the NodeMCU from your computer, and connect it to a **standard USB charger**. Press the reset button on the controller and try reloading the web browser. Both the sensor and actuator should work.

4.2.2 Reflection

In this second part of the laboratory, you were able to supervise the controller and the sensor like you did before, but in a remote manner. As the next figure shows, the connection between the operator and the controller was **wireless**, which required the controller to have an **external power source**. A human can now oversee the system from afar. Note that both the local and remote version of this Laboratory operate on a **closed loop**: the sensor values are fed back to the controller, that in turn activates the actuator. This part of the first laboratory still reflects Industry 3.0: digitalization allows for the fully automatic control of a system.


Power
Server
WiFi
Node MCU
Input
Output
DHT 11
45.0
ROB-08449