Designing a simple semantic sensor using Arduino based boards for smart home applications

Nathan Ramoly
Amel Bouzeghoub

nramoly@ilyeum.com
amel.bouzeghoub@telecom-sudparis.eu

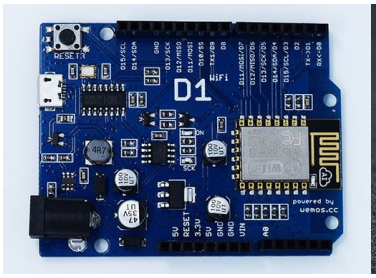
Website:
nara.wp.tem-tsp.eu

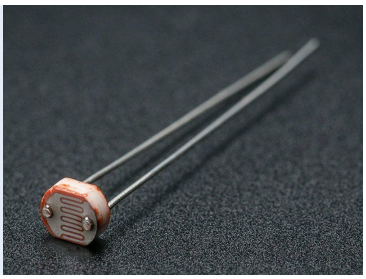
Objective

The aim of this lab is to introduce you to the programmation on Arduino like boards in order to create a sensor for a ‘semantic’ smart home. Through this lab, you’ll see how to set up your board, how to connect it to basic sensors, how to communicate through WiFi, and you’ll create a small application using a context ontology.

Material

- WeMos D1 board
- Solderless breadboard
- Cables
- Resistances
- Various sensors
- USB connector
- Local WiFi network





Documentation

- Java specification: https://docs.oracle.com/javase/7/docs/api/overview-summary.html
- Jena documentation: https://jena.apache.org/documentation/ontology/
- Arduino reference: https://www.arduino.cc/en/Reference/HomePage
- Examples and montages: https://www.arduino.cc/en/Tutorial/BuiltInExamples
- Wemos: https://www.wemos.cc/
- ESP8266 (WiFi) forum: http://www.esp8266.com/

Source

These tutorials were used for preparing this lab:

- http://www.tudovemdachina.com/fr/placa-de-desenvolvimento-wemos-d1-a-evolucao-do-arduino-uno-agora-com-wifi/
- https://www.youtube.com/watch?v=IQVKGAU8jcA
- http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/?ALLSTEPS
- http://www.esp8266learning.com/wemos-webserver-example.php
- https://github.com/wemos/D1_mini_Examples
- https://learn.adafruit.com/photocells/using-a-photocell

Outline

I - Setting up

II - Sensor montage

III - WiFi communication with Java

IV - Semantic analysis and action

V - Further work

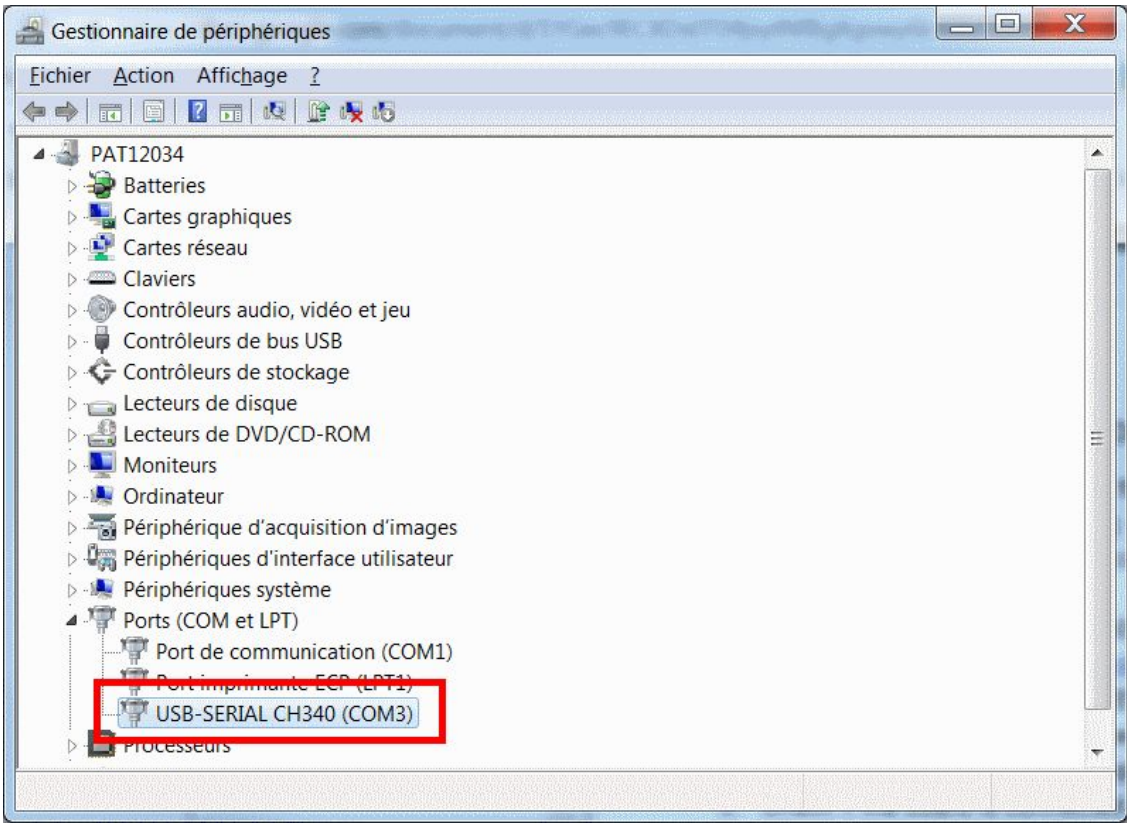
I - Setting up

Objective: Prepare the development environment and ensure everything is ready for the next steps.

Connecting the board to your computer

First of all, it is important to ensure the WeMos board is recognized by your computer. This implies installing driver and checking connectivities. By default, Windows does not have the required drivers, thus it is unable to recognise the board and you can't interact with it.

1. Download and install the CH340G USB interface driver: http://www.arduined.eu/tag/windows-7/ (Or: http://sparks.gogo.co.nz/ch340.html)
    a. Unzip and run CH341SER->SETUP.exe (Windows)
    b. Click on install and close the window
2. Connect your board to the computer through USB
3. Check if the board is connected and installed in your Device Manager panel

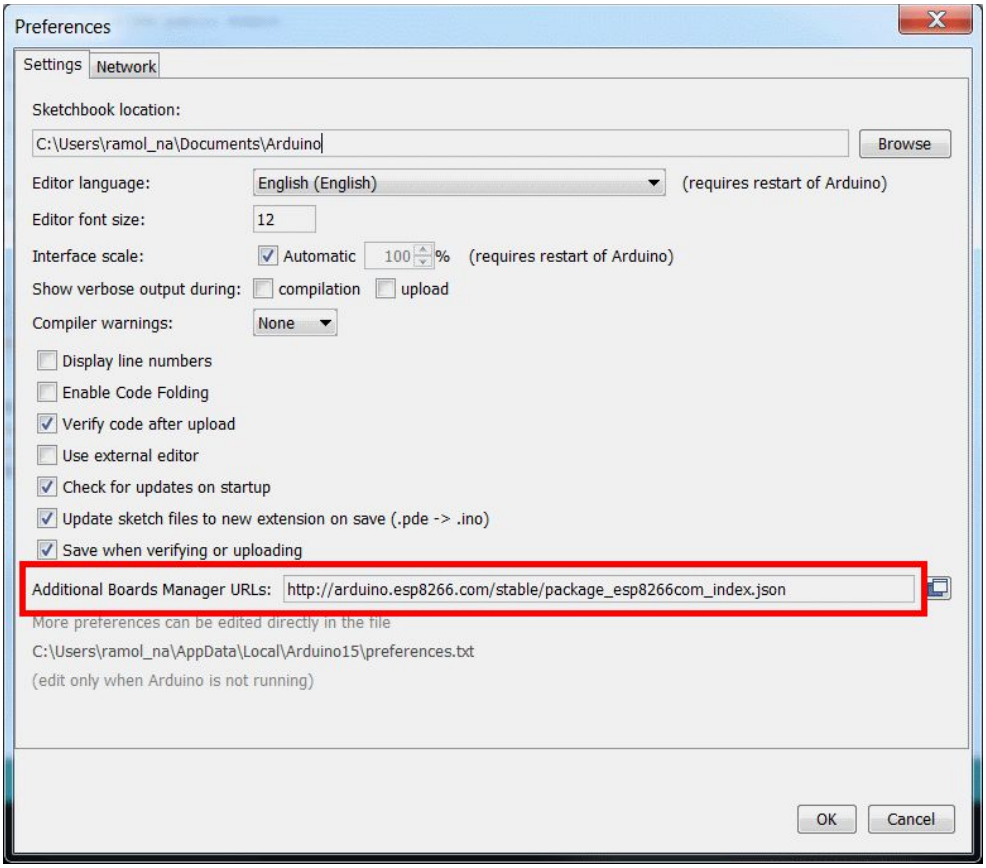


Example on Windows

Setting up the IDE

Once the board is successfully recognized by the computer, we can download and parameter the software. To design and upload programs on the WeMos board, we use the Arduino's IDE. It allows to code programs, called sketches, compile them and send them to the board through the serial bus. Arduino relies on a C like language. Be aware that as we use a third party board, all libraries are not functioning, nevertheless the board and WiFi chipsets are popular and a lot of documentation is available.

1. Download and install the Arduino's IDE: https://www.arduino.cc/en/Main/Software
2. The board is not know by default from the IDE, we need to add it.
    a. Go in File->Preferences
    b. Fill "Aditionnal Boards Manager" with: http://arduino.esp8266.com/stable/package_esp8266com_index.json



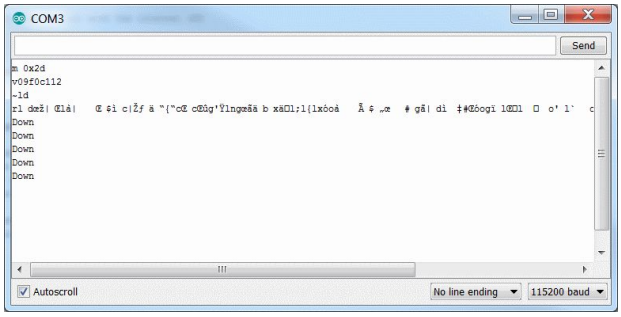
Preferences window

3. Install the board driver
    a. Go to Tools->Board->Boards Manager
    b. Find "esp8266 by ESP8266 Community" and install it. By doing so, you install the drivers for multiple boards using the esp8266 WiFi chipset, including the WeMos D1

4. You can now prepare your IDE for the WeMos board
    a. Go to Tools->Board and select our board “WeMos D1 (retired)”
    b. Go to Tools->Port and select the port of the board (see Device Manager panel)
    c. Go to Tools->Upload Speed and select 115200 (baud)
5. To check if everything is alright, we will run an example
    a. Go to File->Examples->ESP8266->blink
    b. Compile and run
    c. If you see a blue LED blinking, congratulation, the program was successfully uploaded and is now running on the board. :)

Comments

- The black console under the text editor in the IDE only provides log for the compiling and uploading. In order to have feedback from the board, go to Tools->Serial Monitor. It opens a window that displays log sent from the board.



Serial monitor window

- In order to send log, the instruction “Serial.begin(115200);” must be present in the “setup()” function. Please note that the parameter match the upload speed previously selected. You can then use “Serial.print”, “Serial.println”, “Serial.printf”, etc…

- Many examples are available, feel free to play with them to familiarize with the board and the IDE. Be aware that standard example may not work as they are designed for official Arduino board (you can nevertheless check them out !), use the ESP8266 family samples for running examples.

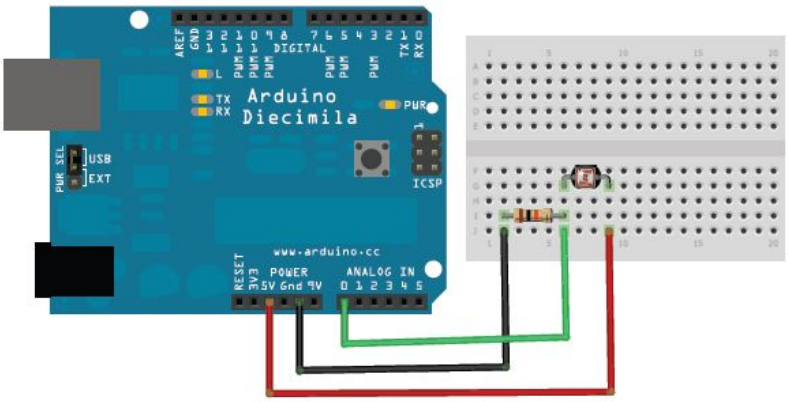
II - Sensor montage

Objective: Link the board to provided sensors using a solderless breadboard, cables and resistances and print the obtained data.

Montage

Build the circuit based on this scheme. Before proceeding, please note:

- WeMos D1 is different from the Arduino Uno depicted in the figure, but is yet similar. Check the indication on the board.
- This circuit uses the 5V source.
- “Gnd” stands for ground.
- On the WeMos, the the analog input starts from the “A0” label.
- Resistance selection:
    - https://en.wikipedia.org/wiki/Electronic_color_code#Resistor_color-coding
    - The sensitivity of the sensor depends on the resistance you put
- Feel free to check: https://learn.adafruit.com/photocells/using-a-photocell



Coding

We will now read the data provided by the light sensor. This is done by reading the voltage on the input port you.

- Create a new sketch that read the sensor value and print it
    - You can use the function “analogRead”, refer to the documentation for more info
- Adapt the code to switch on the embedded LED when it’s dark
    - Refer to the previously seen examples
    - If the LED turn on when you hide the sensor, congratulation, you have successfully set up your light sensor. :)

III - WiFi communication with Java

Objective: Exchange data through WiFi between the board and a Java program running on your computer.

In this section, we will design and test a simple client server application based on example. The application will use TCP/IP protocol. A local WiFi network will be set up for the need of this exercise, feel free to use your own if you prefer. Note that serial communication is possible, but we focus on a wireless scenario which more convenient for a sensor in a smart home.

The client on the board

1. A client example is available, we will use and adapt it
    a. Load the example File->Examples->ESP8266WiFi->WiFiClientBasic
    b. As you can't modify the example, save a copy of it (simply "save as")
    c. Modify the code with the network information provided during the session
    d. Modify the code with the IP address of the server
        i. Use "ipconfig" or "ifconfig" in your computer to obtain it
    e. Adjust/modify the code according to your will

The server on the computer

1. Create a new Java Project
2. Implement a simple TCP server, you can find multiple examples on the web, feel free to use one
    a. The server shall continuously print the received data

Run the application

1. Start the Java server
2. Start the client on the board
3. Adjust the code and debug if necessary
4. If the server print the message from the client, congratulation, you have successfully exchanged data through WiFi from the board to a Java program. :)

Data acquisition

With the communications up and running, we have to send the sensor data to server:

1. Adapt the client sample to send the data obtained from the sensor from section II
2. Adapt the server to interpret, format and print the data received

IV - Semantic analysis and action

Objective: Design a context ontology, integrate sensor data, apply reasoning and act accordingly.

Scenario

In this work, we will implement a very simple scenario of a smart home application of light management.
In this case, the WeMos board measure the light. If you the user is expected to be in a room (in the office in order to work for example) while the there is not light, switch on the board's LED (simulating the light). Oppositely, the LED shall be turned off when the office is supposed empty. We will implement this scenario in this section.

Ontology creation

With Protégé[1], design a context ontology that carries the following information:

- Individuals (example: Pierre, Maeva, Nono…)
- Rooms (example: living room, kitchen)
- Sensors
- Context data (example: luminosity, temperature, location, etc…)
    - Relation towards entities
- User habit and/or agenda (example: working at 5)
- Anything you think is important to store

The structure and the general design remain to your concern. Don't forget that you will use it after, thus, think about your needs and use the scenario to warrant your design choices.

Java integration with Jena

Based on the previous sections, create a program that:

1. Obtain the light sensor data from the board and insert it in the ontology
2. Apply rules to reason on the ontology
    a. Design and insert rules that match the scenario
3. Make the LED on or off according

And test it out. If the system behave and react as expected through your test, congratulation, you now have a simple smart home application running. :)

[1] https://protege.stanford.edu/

V - Further work

Objective: Improve your work on multiple axis and make the application (more) usable.

Before you go…

In the case you finished this lab early, don't worry, this was just the beginning. For now, you just did a very simple scenario with a few case. The next step is to create a real and complete application. Based on what you learnt, you have to slightly improve the system following multiple axis:

- Consider and propose a (or more) complete and realistic scenario (not only one single use case !)
    - Don't limit yourself to the light sensor, but remain in implementable bounds
    - The scenario should concern one board at a time for now
- Improve the context ontology and its rules accordingly
- Explore further and more complex montage
    - More sensors are available
    - Browse examples on the internet
- Add processing inside the board
    - Example: filtering, pattern recognition, data fusion, etc…

In this step, you are autonomous in your directions, choices and implementation. Have fun. ;)

Projects

The work performed in this lab can be used for bigger scale projects. Here are some features that can be addressed as part of project, each of them having its problematics and challenges:

- Freedomotic integration
    - Freedomotic is a framework allowing to create, test and manage smart space
    - Website: http://freedomotic.com/
    - Integrate your WeMos sensor in Freedomotic
- Multiboard systems
    - Gather data from multiple board at a time
    - Use some board as actuators
- Connection to the internet
    - Make the sensor data accessible over the web
    - Be aware of privacy and security issues
- Uncertainty management
    - Some sensor are not accurate, impacting the reasoning
    - Modeling and solving uncertain is a strong issue
- Any awesome idea you may have :)