Polytech Grenoble

Project

RICM4

Wireless deployment for Nucleo

Author:
CHANET Zoran
CHARLOT Servan

Supervisors:
RICHARD Olivier
TORU Sylain

April 8, 2018

Contents

1 Introduction

This project intends to hack an existing technology to push it further, on other platforms. More precisely, the work was about improving or hacking the ESP-link library which provides a Wi-Fi serial bridge for Arduino/AVR microcontrollers to remotely flash (reprogram) an STM-32 microcontroller. This type of microcontroller isn't handled by the ESP-link, even in the latest releases. A serial bridge allows you to connect via Wi-Fi to a microcontroller and act as if you were connected to it via Serial port. May the reader be aware that we didn't fully succeed, but this report intends to explain in a precise but understandable way the steps we realised and what's left for the project to be fully functional.

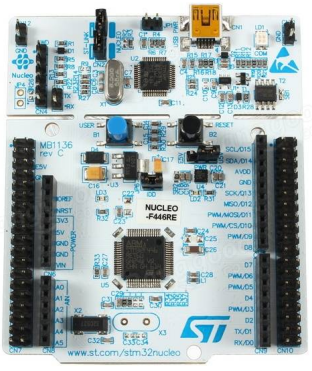
2 Project continuity

To complete this project (and because we didn't know the technologies yet), we had to progress sequentially from the easiest steps to our goal. Every of those steps will be described in this report (mostly in the section 4). The logic of this progression is as follow : first we had to discover the microcontrollers. Thus we experimented how to flash them with simple programs, the easiest way as possible. Then we had to configure the ESP8266EX, which basically means flashing it with a release of the ESP-link [6] (we chose the latest stable release 3.0.14 [2] at the moment). Next we needed to flash each of the microcontrollers via serial port, as the ESP-link [6] will simulate a serial connection. After that comes the use of the ESP-link to remotely flash the Arduino/AVR microcontroller (as intended by the library). Last, but not least, comes the hacking, hence the modification of ESP-link's classic use to remotely flash the STM-32 microcontroller.

3 Technologies and devices

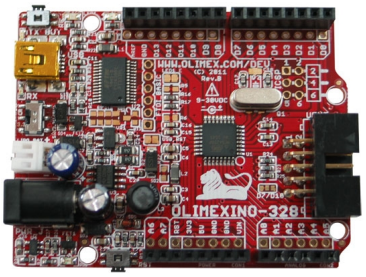
3.1 STM32F446RE Nucleo-64

Also referenced as Nucleo. It is the STM-32 type microcontroller we used for this project. It is a STMicroelectronics product [10].
It is a classic microcontroller, which can be flashed with C++ programs to behave as wanted, and can handle many added modules (sensors, LEDs, screens...).



3.2 Olimexino-328 (Arduino)

Also referenced as Olimexino, Arduino. It is the Arduino/AVR type microcontroller we used for this project. It is an Olimex product [9].
It is a classic microcontroller, which can be flashed with C++ (Arduino) programs to behave as wanted, and can handle many added modules (sensors, LEDs, screens...).
It may also be used as an USB to serial converter (by setting a jumper between `RST` and `GND`), allowing communication between a computer's USB port and a microcontroller's serial port.

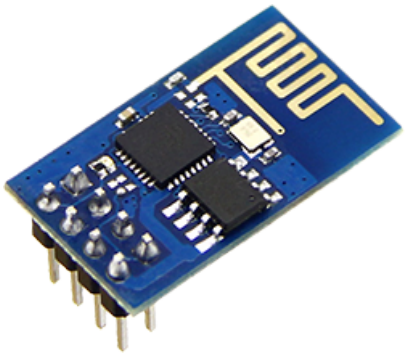


3.3 ESP chipsets

ESP chipsets of every type are Wi-Fi transceivers, which means they can both receive and send data via Wi-Fi. In this project, we implement the ESP-link library [6] on them, which comes with many features including a Wi-Fi network, an associated webpage, the possibility of flashing an attached Arduino/AVR microcontroller...

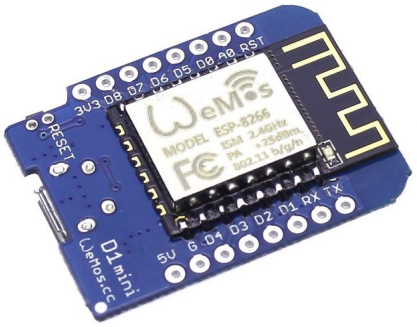
3.3.1 ESP8266EX-01

Also referenced as ESP, ESP8266, ESP8266EX. It is the first ESP type board we used for this project. It is an Espressif product [4].
Difference between type 01 and type 12 mostly comes from the pin mapping.



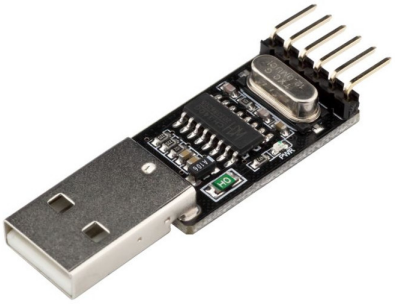
3.3.2 ESP8266EX-12 WEMOS

Also referenced as ESP, ESP8266, ESP8266EX, WeMos. It is the second and last ESP type board we used for this project. It is an Espressif product [4] and WeMos [13].
Difference between type 01 and type 12 mostly comes from the pin mapping.



3.4 USB Serial adapter CH340G

Also referenced as USB to TTL, USB to TTL converter, Serial adapter. It is the converter we used to flash the Olimexino via serial port. It is a RobotDyn product [12].
It allows communication between a computer's USB port and a microcontroller's serial port.



4 How to use the different boards

4.1 Develop programs

4.1.1 Arduino boards

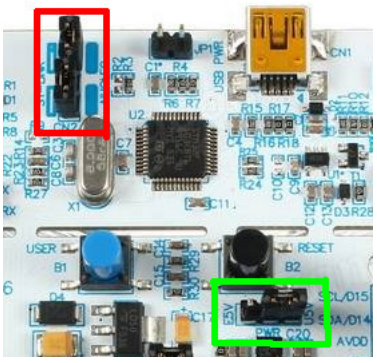
Download the Arduino software on the Arduino website [1]. This is an IDE where you can develop programs for your board in C++ (with additional methods useful for the boards). It produces `.hex` compiled files.

4.1.2 Nucleo boards

You first need to connect on the Mbed website [7]. Then you complete your profile by telling which board model you possess. Then you can access the Mbed Compiler mode where you can start writing your programs in C++. It produces `.bin` compiled files.

4.2 Flash the Nucleo using USB connection

First make sure that on your Nucleo board, you have the two jumpers on the `ST-LINK` `NUCLEO` pins *(in red on the image below)*, and one jumper between `PWR` and `U5V` *(in green on the image below)*.



Plug the Nucleo board to your computer with a USB/MiniUSB cable. It automatically opens the folder of the device. (If not, you can find it in `/media/$USER/` on Linux). Just copy your binary file into it and it will flash the Nucleo with it.

4.3 Flash the Olimexino using USB connection

Plug the Olimexino board to your computer with a USB/MiniUSB cable. The first way to flash it is to do it from the Arduino IDE :

- Launch the Arduino IDE
- Open your program's source code (File → Open...) or an example (File → Examples)
- Select the right board type and USB port (Tools → Board and Tools → Port)
- Upload your program into the board (either via the right arrow in the top left-hand corner or Sketch → Upload)

The other way is to use the program called `avrdude`, with the following command (replacing `m328p` by the code corresponding to your Arduino-like board, `/dev/ttyUSB0` by the port your board is connected to and `your_program.hex` by the path to the `.hex` file of your program) :

```
avrdude -b 57600 -p m328p -c arduino -P /dev/ttyUSB0 \
  -U flash:w:your_program.hex
```

4.4 Flash the ESP with the ESP-link library using USB connection

If your ESP board isn't flashed with the ESP-link library [6] yet, you have to do this step before trying to use the ESP board as a Wi-fi serial bridge.
Download the latest release of the library (we used the 3.0.14 [2]) and download the `Esptool` software [5]. Plug the ESP to your computer with a USB/MicroUSB cable, move to the folder where you unzipped the ESP-link library [6] and execute the following command (replacing this command by the one corresponding to your ESP chipset's flash size in the ESP-Link's documentation [3]) :

```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash \
  -fs 512KB -ff 40m 0x00000 boot_v1.6.bin 0x1000 user1.bin \
  0x7C000 esp_init_data_default.bin 0x7E000 blank.bin
```

Your ESP board is now initialised and you can access its network at the address `http://192.168.4.1/`. If ever the ESP board ain't working anymore, you can erase the library with the following command, then flash it again :

```
esptool.py erase_flash
```

4.5 Flash the Olimexino via TTL Serial Port

You first need to create a serial bridge between the Olimexino and the computer with the Serial adapter.
Hard-wiring :

- `GND` pin of the TTL converter to `GND` pin of the Olimexino.
- `3v3` out pin of the TTL converter to `3v3` in pin of the Olimexino.
- `RX` pin of the TTL converter to `RX (D0)` pin of the Olimexino.
- `TX` pin of the TTL converter to `TX (D1)` pin of the Olimexino.
- `RST` pin of the TTL converter to `RST` pin of the Olimexino.

If you have any doubt, you can check the Olimexino documentation [8].
Once the wiring is done, you can use `avrdude` to flash the Olimexino with the following command (replacing `m328p` by the code corresponding to your Arduino-like board, `/dev/ttyUSB0` by the port your board is connected to and `your_program.hex` by the path to the `.hex` file of your program) :

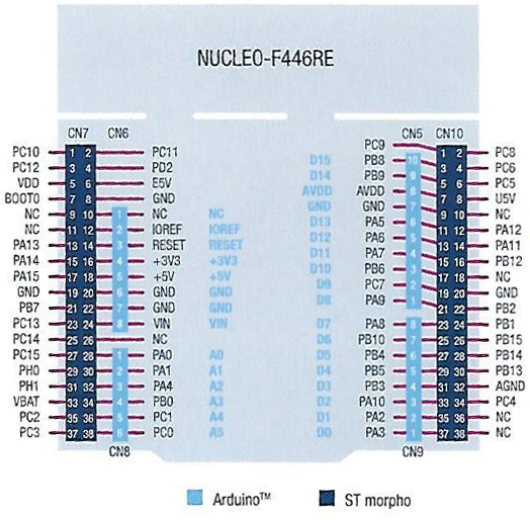
```
avrdude -b 57600 -p m328p -c arduino -P /dev/ttyUSB0 \
  -U flash:w:your_program.hex
```

4.6 Flash the Nucleo via Olimexino Serial port

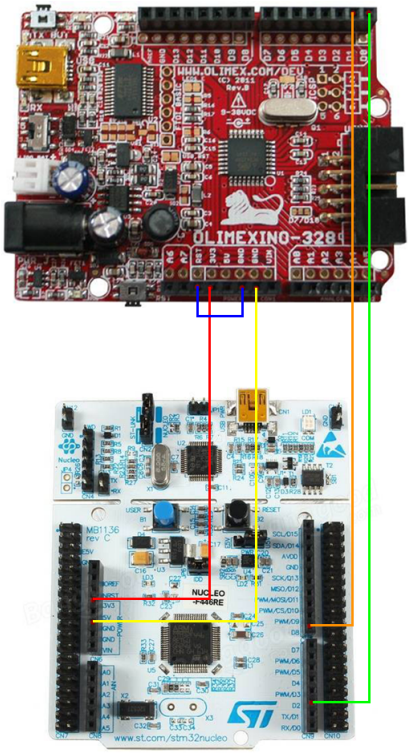
You first need to create a serial bridge between the Nucleo and the computer with the Olimexino.
Hard-wiring :

- Connect `RST` of Olimexino to `GND` of **Olimexino**.
- Connect `3.3v` of Olimexino to `3.3v` of Nucleo.
- Connect `GND` of Olimexino to `GND` of Nucleo.
- Connect `RX (D0)` of Olimexino to `RX (PA10/D2)` of Nucleo.
- Connect `TX (D1)` of Olimexino to `TX (PA9/D8)` of Nucleo.
- Make sure that on the Nucleo you have the two jumpers on the `ST-LINK`. (See 4.2).
- Make sure that on the Nucleo you have a jumper between `E5V` and `PWR` (because when you try to flash it via USB this jumper must be between `PWR` and `U5V` for the Nucleo to execute its program). (See 4.2).

If you have any doubt, you can check the Olimexino documentation [8] or the Nucleo documentation below.



Here is a wiring schematic for this step :



At this point if you power the Olimexino with the USB connection to your computer, the Nucleo will behave normally (executing its program).
Now you want to enter Nucleo's bootloader mode :

- Connect the `BOOT0` of the Nucleo to the `VCC` of the STM32 (set `BOOT0` to high).
- Press the reset button of the Nucleo and wait a little bit
- Then you can flash your Nucleo with a binary file using `stm32flash` [11] with the following command (replacing `your_program.bin` by the path to the `.bin` file of your program and `/dev/ttyUSB0` by the port your board is connected to) :

```
sudo ./stm32flash -w your_program.bin /dev/ttyUSB0
```

- Set `BOOT0` to low (disconnect `BOOT0` of the Nucleo from `VCC`).
- Press the reset button one more time.

The Nucleo now behaves normally and executes your program.

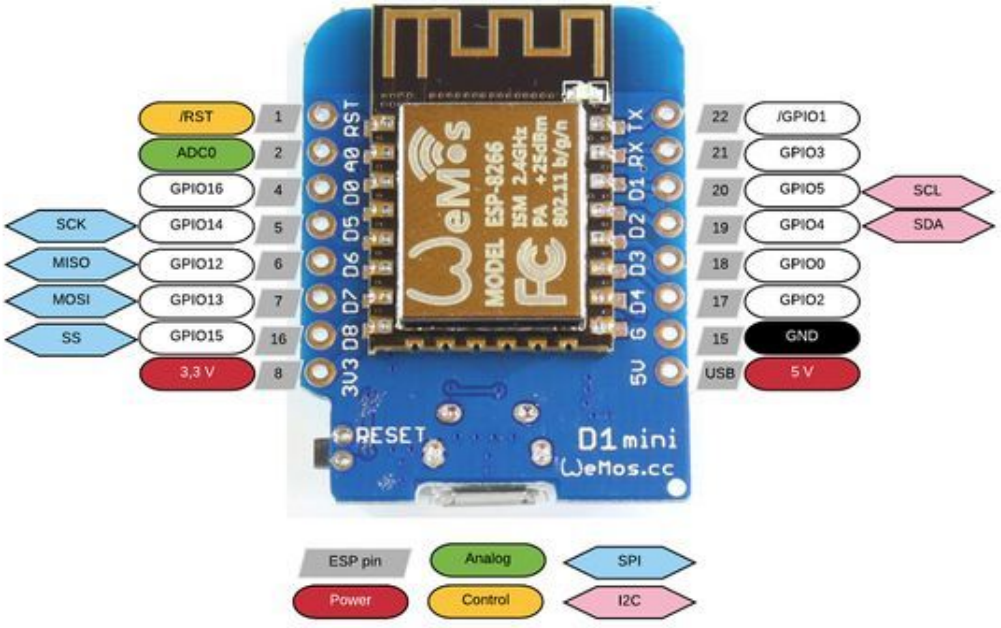
4.7 Flash the Olimexino by Wi-fi with the ESP

Your first need to have your ESP board flashed with the ESP-link library [6]. If it is not the case, refer to the subsection 4.4. Then you need to create a serial bridge between the ESP and the Olimexino. It didn't work for us with the ESP8266EX-01 so we will not detail the hard-wiring for this board here, but you can find it in the ESP-link documentation [3].

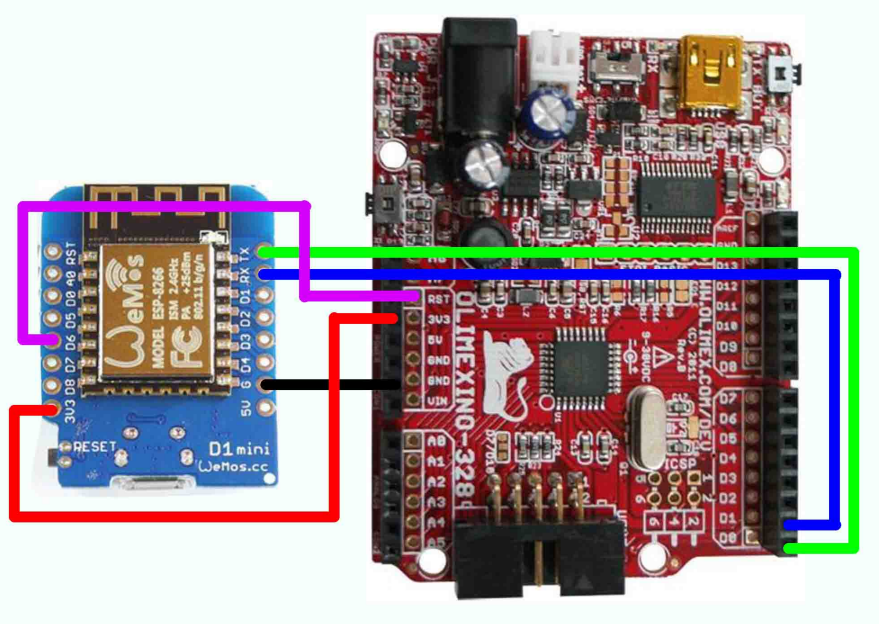
Hard-wiring for the ESP-12:

- `GND` pin of the ESP to `GND` pin of the Olimexino.
- `3v3` out pin of the ESP to `3v3` in pin of the Olimexino.
- `URXD` pin of the ESP to `TX` pin of Olimexino.
- `UTXD` pin of the ESP to `RX` pin of Olimexino.
- `GPIO12` pin of the ESP to `RESET` pin of Olimexino.
- `GPIO13` pin of the ESP to `ISP` pin of LPC/ARM Olimexino (not used with Arduino/AVR).

If you have any doubt, you can check the Olimexino documentation [8] or the ESP-12 WeMos documentation below.



Here is a wiring schematic for this step :



Then you need to connect to the ESP Wi-fi network at the address `http://192.168.4.1/`, and use the following command to remotely flash the Olimexino (replacing `m328p` by the code corresponding to your Arduino-like board and `your_program.hex` by the path to the `.hex` file of your program) :

```
avrdude -b 57600 -p m328p -c arduino -P net:192.168.4.1:23 \
  -U flash:w:your_program.hex
```

It is supposed to be possible with the `avrflash` software available with the ESP-link library [6] but it did not work for us with this one.

5 Difficulties and known issues

5.1 Difficulties encountered

The first (maybe the main) difficulty we encountered was our ignorance of the technologies. We had to discover not only the ESP-Link [6] library, but also the microcontrollers, and how to get them to work. In addition, we had to change several units because the first we were given weren't working properly, which had us spending time over mistakes we could neither detect nor understand.

Also, the lack of documentation on the ESP-Link [6] library's functions and architecture, and the `stm32flash` sources [11] (the software we used to flash the Nucleo) gave us a hard time understanding how to and what to modify in order to implement another programmer on the ESP in order to flash a STM-32 instead of an Arduino/AVR microcontroller. The programmer is the part of the code that flashes the microcontroller, and is then dedicated to a certain microcontroller type. We could have avoided this by adding the STM-32 programmer into `avrdude` software instead (as this software can be used to remotely flash a microcontroller via the ESP thanks to the ESP-Link [6] library), but it seemed more logical to focus on the ESP-Link library [6].

5.2 Known issues with the boards

The remote flashing of our Arduino-like microcontroller (the Olimexino) via the ESP-Link [6] ain't working properly, even though it is one of the first purposes of the library. That may come from an incompatibility between the Olimexino and the `stk500` protocol, which is the one useb by both the ESP-Link [6] and `avrdude` to synchronise with the microcontroller. This could have been possibly avoided with the use of an other Arduino-like microcontroller (e.g. the Arduino Uno or any real Arduino [1] product). We couldn't try this solution, so that issue isn't confirmed.

6 Possible improvements

An improvement we couldn't carry out was to extend the Arduino [1] software in order to detect and use ESP-Link's [6] networks, and use them to remotely flash a microcontroller directly from the software interface.
Another one would be to crop the ESP-Link [6] library in order to extract only the files and features that are useful for this specific project (thus getting rid of the web page etc.), to offer a lighter library. Also, doing this we wouldn't forget to document every part of the extracted library, to ease its use and future improvements.
A last thing we could consider is the security factor. For now everybody is able to see the ESP network, connect to it, and execute the flashing command. It could be interesting to restrict the access to the network to an administrator.

References

[1] Arduino website. https://www.arduino.cc/. Accessed: 2018-04-08.

[2] Esp-link 3.0.14 release. https://github.com/jeelabs/esp-link/releases/tag/V3.0.14. Accessed: 2018-04-08.

[3] Esp-link flashing documentation. https://github.com/jeelabs/esp-link/blob/master/FLASHING.md. Accessed: 2018-04-08.

[4] Esp8266ex specification. https://www.espressif.com/en/products/hardware/esp8266ex/overview. Accessed: 2018-04-08.

[5] Esptool software. https://github.com/espressif/esptool. Accessed: 2018-04-08.

[6] Jeelabs esp-link library. https://github.com/jeelabs/esp-link. Accessed: 2018-04-08.

[7] Mbed website. https://www.mbed.com/. Accessed: 2018-04-08.

[8] Olimexino-328 documentation. https://www.olimex.com/Products/Duino/AVR/OLIMEXINO-328/resources/OLIMEXINO-328-schematic.pdf. Accessed: 2018-04-08.

[9] Olimexino-328 specification. https://www.olimex.com/Products/Duino/AVR/OLIMEXINO-328/open-source-hardware. Accessed: 2018-04-08.

[10] Stm32f446re nucleo-64 specification. http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f446re.html. Accessed: 2018-04-08.

[11] Stm32flash software. http://stm32flash.sourceforge.net/. Accessed: 2018-04-08.

[12] Usb serial adapter ch340g specification. http://robotdyn.com/usb-serial-adapter-ch340g-5v-3-3v.html. Accessed: 2018-04-08.

[13] Wemos website. https://www.wemos.cc/. Accessed: 2018-04-08.