IoT Lab

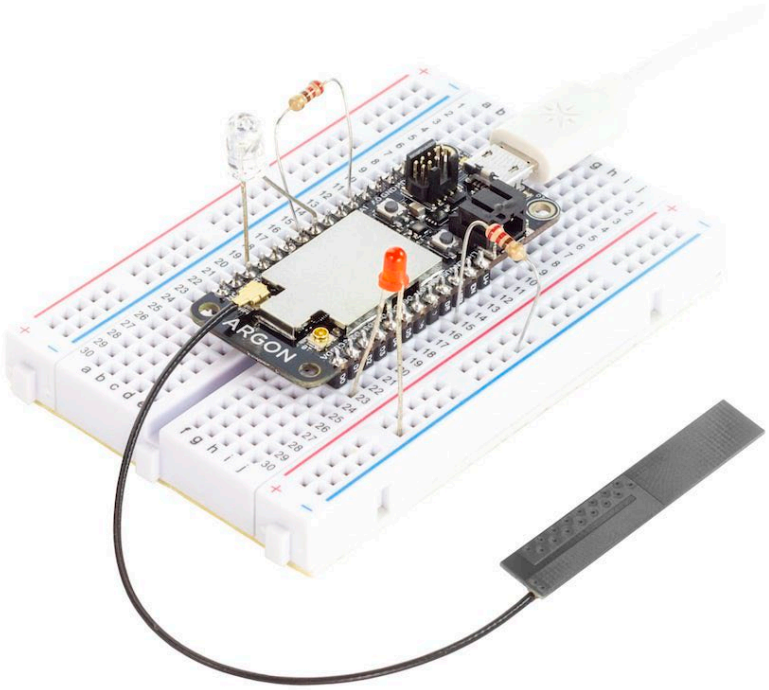
Laboratorio 3 ETSE

Table of Contents

PARTICLE ARGON: WI-FI + BLUETOOTH

The Particle Argon is a powerful Wi-Fi development kit that you can use on Wi-Fi networks.

Equipped with the Nordic nRF52840 and Espressif ESP32 processors, the Argon has built-in battery charging circuitry which makes it easier to connect a Li-Po battery and 20 mixed signal GPIOs to interface with sensors, actuators, and other electronics.



Main processor:

Nordic Semiconductor nRF52840 SoC

- ARM Cortex-M4F 32-bit processor @ 64MHz
- 1MB flash, 256KB RAM
- Bluetooth LE (BLE) central and peripheral support
- 20 mixed signal GPIO (6 x Analog, 8 x PWM), UART, I2C, SPI

- Supports DSP instructions, HW accelerated Floating Point Unit (FPU) calculations
- ARM TrustZone CryptoCell-310 Cryptographic and security module
- Up to +8 dBm TX power (down to -20 dBm in 4 dB steps)
- NFC-A radio

Argon Wi-Fi network coprocessor:

Espressif ESP32-D0WD 2.4 GHz Wi-Fi coprocessor

- On-board 4MB flash for the ESP32
- 802.11 b/g/n support
- 802.11 n (2.4 GHz), up to 150 Mbps

Argon general specifications:

- On-board additional 4MB SPI flash
- Micro USB 2.0 full speed (12 Mbps)
- Integrated Li-Po charging and battery connector
- JTAG (SWD) Connector
- RGB status LED
- Reset and Mode buttons
- On-board 2.4GHz PCB antenna for Bluetooth (does not support Wi-Fi)
- Two U.FL connectors for external antennas (one for Bluetooth, another for Wi-Fi)
- Meets the Feather specification in dimensions and pinout
- FCC, CE and IC certified
- RoHS compliant (lead-free)

ARGON FIRST SETUP

There are different ways for the first setup of the Argon. For this seminary we have chosen to do it with the software “particle-cli”. Install instructions may be found at:

https://docs.particle.io/tutorials/developer-tools/cli/

Using this software, the Argon can be configured through the USB. It is required to be registered at www.particle.io . Once you have an account connect the Argon to the computer with USB cable an follow these instructions:

Argon Setup

1. You must have the Particle CLI (version 1.47.0 or newer) installed. Once the installation is finished follow the instruction and write `particle setup` in the Windows CMD.
   Use `particle update-cli` to upgrade if necessary.
2. Open up your computer's Terminal (Mac) or Command Line (Windows).
3. Download the following from the Argon NCP release site: argon-ncp-firmware-0.0.5-ota.bin (https://github.com/particle-iot/argon-ncp-firmware/releases/download/v0.0.5/argon-ncp-firmware-0.0.5-ota.bin) and save it within an accessible directory. Use `cd` in your Terminal to navigate to the directory that contains this file.
4. Attach the Wi-Fi antenna to your Argon. Make sure you connect it to the correct connector, there are three U.FL connectors: WiFi, BT, and NFC.
5. Remove the Argon from the anti-static foam before powering the device.
6. Plug the Argon into a USB port on your computer (make sure that you have already connected the Wi-Fi antenna). It should begin blinking blue.
7. Put the Argon in DFU mode (blinking yellow) by holding down MODE/SETUP. Tap RESET and continue to hold down MODE. The status LED will blink magenta (red and blue at the same time), then yellow. Release when it is blinking yellow. Once you've updated your device (below), you can just run the following command in the Particle CLI to bring your device into DFU mode:

   ```
   particle usb dfu
   ```

8. Update the device by running the following two commands. If the device goes out of blinking yellow after the first command, put it back into DFU mode (see above).

   ```
   particle update
   particle flash --usb tinker
   ```

9. When the command reports **Flash success!**, reset the Argon. It should go back into listening mode (blinking dark blue). If it doesn't, do it manually by pressing the MODE/SETUP buttom for a couple of seconds.
10. Flash NCP firmware to the device. Note that this must be done in listening mode (blinking dark blue). It cannot be done in DFU mode:

    ```
    particle flash --serial argon-ncp-firmware-0.0.5-ota.bin
    ```

11. Verify that the updates worked by running the following command (to do this the device must be in listening mode, that is blinking dark blue):

    ```
    particle serial identify
    ```

    which should return something like this:

    ```
    Your device id is e00fce681fffffffffc08949b
    Your system firmware version is 1.5.0
    ```

12. Save the device ID, you'll need it later.
13. Set your Wi-Fi credentials. This must be done in listening mode.

```
particle serial wifi
```

14. Login into your Particle account with:

```
particle login
```

15. After setting, you Argon should go through the normal sequence of blinking green, blinking cyan (light blue), fast blinking cyan, and breathing cyan.
16. Claim the device to your account. This can only be done if it's breathing cyan.
    Replace `e00fce681ffffffffc08949b` with the device ID you got earlier from particle serial identify.

```
particle device add e00fce681ffffffffc08949b
```

17. Rename the device: For the sake of clarity, use the number assigned to your group (`X`) and name the device as `argon-X`

```
particle device rename e00fce681ffffffffc08949b argon-X
```

18. Ensure that your setup flag is marked as done:

```
particle usb setup-done
```

19. You have successfully set up your Argon!

HARDWARE –SOFTWARE

Once the Argon is setup, we can program it through the web IDE at particle.io . To test it, and to get a basic knowledge of how it works, we will go through the examples at:

https://docs.particle.io/tutorials/hardware-projects/hardware-examples/

Network Time Protocol

The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. In operation since before 1985, NTP is one of the oldest Internet protocols in current use.

In the context of the IoT, it is vital that all the nodes in the Network have the same time reference. For that, a common NTP server is commonly used. In our case, we will get the time reference from the University of Santiago de Compostela server (hora.usc.es).

An easy approach to implement this service in our projects is to use libraries developed by the community. For instance, ntp-time (in small letters) is widely applied in the Particle community.

To include a library, we must go to the "Libraries" section, at the left panel in the web IDE, search the desired library and include it to our project. An skeleton for ntp-time would be for example:

```
#include <ntp-time.h>

static char timeServer[] = "hora.usc.es";

NtpTime ntptime=NtpTime(10,timeServer);

void setup(){

        ntptime.start();

        Time.setTime(ntptime.now());

        Serial.begin(9600);

}

void loop(){

Serial.println(Time.year());

delay(2500);

}
```

This setup would allow us to use Time library functions synchronized with the configured server.

Once the code is ready, upload it on the device using the `Flash` option on the Web IDE. Note that the code is uploaded through the WiFi connection, not through the USB cable.

Exercise: *Check Time library at Particle support webpage and complete the previous code skeleton to print through the serial connection current date and time every 5 seconds.*

Since the Argon IDE does not have an integrated serial monitor as in the case of Arduino, you need to open it manually. If you are using Windows, you can do it by opening a putty serial session. Be careful to use the correct name of the serial port (for example, COM3). You can check the name on the device manager of your computer.

SERVER SIDE

Once the data are captured and processed with Argon, the goal is to send them to the cloud using the WIFI network deployed at the laboratory. If you have not already done so, connect now to the Bigdata WiFi:

SSID: Bigdata

password: x5vJ*4CaBCkJ

Data management is going to be handled by a Fiware arquitecture. We will begin with a CentOS 8 minimal machine, and we will need to install and configure all required software, i.e., MongoDB, Fiware, Cygnus and a mysql server.

To access the machine we will use a ssh connection. Depending on you OS this will be done in different ways. Typically, the ssh connection can be executed directly in a command line terminal through the following command:

```
ssh usuario@172.16.240.210 -p 22XX
```

where XX is the number assigned to each group.
If a password is required, use “iot2023”. “usuario” is an user created with administration permissions.
If you are using Windows, use putty to connect to your virtual machine.

Software install

The first recommended steps are to update the machine and to install nano, if not installed already:

yum check-update

sudo yum update

Fiware

The Orion Context Broker is a service that allows us to send and receive big amounts of data through the concept of entities that will have attributes. Entities are an abstraction of the physical nodes or connected devices in the context of IoT. These data are stored in a MongoDB database.

To install Orion Context Broker:

wget https://nexus.lab.fiware.org/repository/el/8/x86_64/release/contextBroker-3.6.0-1.x86_64.rpm

sudo yum localinstall contextBroker-3.6.0-1.x86_64.rpm

MongoDB

MongoDB is a open-source database system NonSQL oriented to documents that, instead of storing data in tables as SQL databases, uses data structures in documents such as JSON, making data integration in some applications faster and easier.

We will install it from the official repositories. For that, we need to create the repository manually. Using nano, we create the file mongodb-org.repo in the /etc/yum.repos.d/ folder with command:

sudo nano /etc/yum.repos.d/mongodb-org.repo

Then, in this file we add:

[mongodb-org-4.4]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.4/x86_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-4.4.asc

To check that the repository was added properly, we can use “yum repolist”, that will show that the repository was included correctly.

Now, we can install it with:

sudo yum install mongodb-org

MySQL

MongoDB does not create an historical record of the gathered data, and every data submission overwrites the previous value. In order to store the historical record, we need to install also a SQL database. In this case we are going to use MySQL, as it is one of the most widely used databases. It is included in the official Centos 8 repositories. Thus, we just need to execute:

sudo yum install mariadb-server

Cygnus

Cygnus is the software responsible of the communication between Orion Context Broker and the MySQL database. It is included in the Fiware repolist, but it is not in the Centos 8 repository yet. However, the one in the Centos 7 repository works fine, and we need to change the repository file. This time, we will download directly the repository file from the Fiware webpage with command:

sudo wget -P /etc/yum.repos.d/
https://nexus.lab.fiware.org/repository/raw/public/repositories/el/7/x86_64/fiware-release.repo

Then we install it with

sudo yum install cygnus-ngsi

Also, as Cygnus needs java, we must install it through:

sudo yum install java-1.8.0-openjdk

Finally, for tcp ports management we will need the firewalld tool. To install it:

sudo yum -y install firewalld

It is also a good idea to start and enable it right away with:

sudo systemctl enable firewalld

sudo systemctl start firewalld

Server configuration

The first step to configure the server can be for instance to open the MongoDB port, where it is going to receive the data. By default, it is going to be the 1026. This can be done with command:

```
sudo firewall-cmd --zone=public --add-port=1026/tcp --permanent
sudo firewall-cmd --reload
```

NOTE: MongoDB uses 1026 port but, due to the framework built for this seminary each group will send the data to port 23XX, and depending on the group (XX) these data are going to be redirected to port 1026 of their machine.

Now we can initiate now MongoDB and Orion Context Broker with commands:

```
sudo systemctl start mongod
sudo systemctl enable mongod

sudo /etc/init.d/contextBroker start
sudo /etc/init.d/contextBroker enable
```

Cygnus configuration

For the configuration of Cygnus you need to copy the files `agent_ngsi_prueba2.conf` `cygnus_instance_prueba2.conf` and `grouping_rules.conf` that you can find in your virtual campus to the path `/usr/cygnus/conf` in your virtual machine. If you are using windows you will need WinSCP or the FileZilla client. If you have permissions' problems when copying the files to `/usr/cygnus/conf` do the following: first copy them into your /home/usuario path and then move them (with sudo) to `/usr/cygnus/conf`.

Once MongoDB and Orion Context Broker are both running, we can test them by creating an entity, updating its data, and making a query to check if the data has been correctly updated (more info here).

Entity Creation, Query and Update

Orion Context Broker starts in an empty state, so first of all we need to make it is aware of the existence of certain entities. In particular, we are going to "create" the Mote1 entity which has two attributes ('temperature' and 'timeInstant'). We do this using the POST /v2/entities operation. In Windows we create an script *pruebaCreation.bat that calls jsonpruebaCreation.txt*

pruebaCreation.bat

```
curl 172.16.240.210:2320/v2/entities -s -S --header "Content-Type:
application/json" -d @jsonpruebaCreation.txt
```

jsonpruebaCreation.txt

```
{
  "id": "Mote1",
  "type": "Mote",
  "timeInstant": {
    "value": "2018-01-01T00:00:00Z",
    "type": "time",
    "metadata": {
      "timeInstantUnit": {
            "value": "ISO8601",
            "type": "string"
            }
      }
  },
  "temperature": {
    "value": 27.0,
    "type": "float",
    "metadata": {
      "temperatureUnit": {
        "value": "celsius",
        "type": "string"
      }
```

```
    }
  }
}
```

You can find these files in your virtual campus, as well as those for making an update and a query on the created entity. These files must be modified updating:

- XX: Port number according to group number.
- idname: Name of the device (for example Mote1)
- typename: Type of the device (for example Mote)
- magnitudename: Magnitude measured (for example light intensity)
- magnitudeUnit: Units used (for example volts).

To execute these files it is necessary to install curl. For Linux systems this is usually installed by default. Windows users will need to install it (from https://curl.se/windows/). Of course, you can use any REST client tool instead (e.g. RESTClient). Indeed, in a real case, you will probably interact with the Orion Context Broker using a programming language library implementing the REST client part of your application. However, curl will be the tool used within this lab.

To check that everything is correct, **create an entity** (pruebaCreation.bat if you are using Windows), then make a query (pruebaQuery.bat) to check that the entity has been created and finally **update** the value (pruebaUpdate.bat) followed by another **query**.

Subscriptions

To create, query and update entities are the basic building blocks for *synchronous* context producer and context consumer applications. However, Orion Context Broker has another powerful feature: the ability to subscribe to context information so when "something" happens your application will get an asynchronous notification. This way, you don't need to continuously repeat query requests. The next step will be to **create a subscription** to retrieve info about Mote1 and to tell MongoDB that it must send data to Cygnus. When creating a subscription you must identify the entities and attributes you are subscribing to, the URL where to send notifications and the condition that triggers the notification (in the example below a change in the attribute `timeInstant`). The throttling element is used to specify a minimum inter-notification arrival time. You can also define an expiration date of the subscription.

You can use the files on the virtual campus (pruebaSubscription.bat) after editing accordingly to your requirements (port, ids, types…).

```
curl 172.16.240.210:23XX/v2/subscriptions -s -S --header "Content-Type:
application/json" -d @jsonpruebaSubscription.txt
```

jsonpruebaSubscripion.txt file:

```
{
  "description": "A subscription to get info about Mote1",
  "subject": {
    "entities": [
      {
        "id": "Mote1",
```

```
        "type": "Mote"
      }
    ],
    "condition": {
      "attrs": [
        "timeInstant"
      ]
    }
  },
  "notification": {
    "http": {
      "url": "http://localhost:5050/notify"
    },
    "attrs": [],
    "attrsFormat": "legacy"
  },
  "expires": "2040-01-01T14:00:00.00Z",
  "throttling": 1
}
```

Once the proper behaviour of the system is checked, we must initiate cygnus and mysql to get an historical record of submitted data. For that, type:

```
sudo systemctl start mariadb.service
sudo systemctl enable mariadb.service

sudo service cygnus start
sudo service cygnus enable
```

Be patient, you might have to wait several seconds. It may also happen that the system is limited in RAM, and it is not able to start mariadb. If that happens, stop mongodb, start mariadb and then mongodb again.

Mysql database

We should create a root user in the mysql database with command:

```
mysqladmin -u root password bigdata
```

To check that everything works fine, use the pruebaUpdate.bat file to update the data a few times (remember to increase the default timestamp in the Update command each time). Then access the mysql database to see the historical record. That can be done entering mysql databases with:

```
mysql -u root -p
```

We must type the password (bigdata) and then:

```
Show databases;

Use def_serv;
```

```
Show tables;

Select* from XXXXX
```

where def_serv is the default name of the database set in the configuration files and XXXXX is the table that we want to check.

Particle Argon – Server communication

The purpose of this section is to use the data gathered by the Particle Argon to update the created database. For that, we must POST these values in the appropriate format with the correct timestamp. You can find a program skeleton in file Argon_to_server.ino file, that you must edit according to your requirements. The important changes that you must do are:

- Edit line where the JSON is created to add the measured value instead of the default number.
- Edit the port you must use.
- Edit “idname”
- Edit “magnitudename”