T8 - IoT & Virtuality

T-DEV-811

Smart trash cans

Boostrap 2

{EPITECH.}

IoT

Game 1

Connect arduino to the ESP8266 trough the UART connection.

The ESP is a wifi module that will allow the Arduino to communicate with the internet.

Wikipedia ESP8266
Datasheet ESP8266

The arduino is operating at 5 volts nominal tension and the ESP8266 is operating at a 3 volts nominal tension.
This means that you can neither make them communicate with each other nor power the ESP from the Arduino.

The ESP would be in overvoltage and would certainly be damaged.

> To avoid this, always use the adapter provided in your kit.

Its job is to convert the logic voltage from 5 volts to 3 volts to allow the two components to interface harm-lessly.

The communication speed of the ESP8266 is 115200bps so you will absolutely have to use an arduino mega with several serial links. The arduino uno will make unpredictable errors.
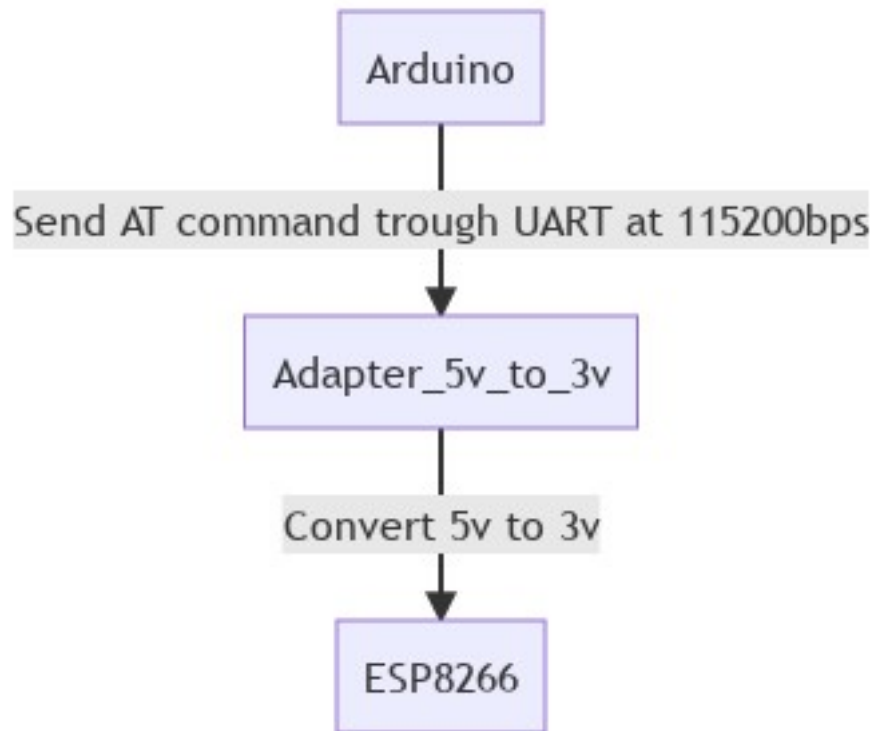
```
graph TD;
    Arduino -- Send AT command trough UART at 115200bps--> Adapter_5v_to_3v;
    Adapter_5v_to_3v -- Convert 5v to 3v--> ESP8266;
```

The connection between the pins should be like this :
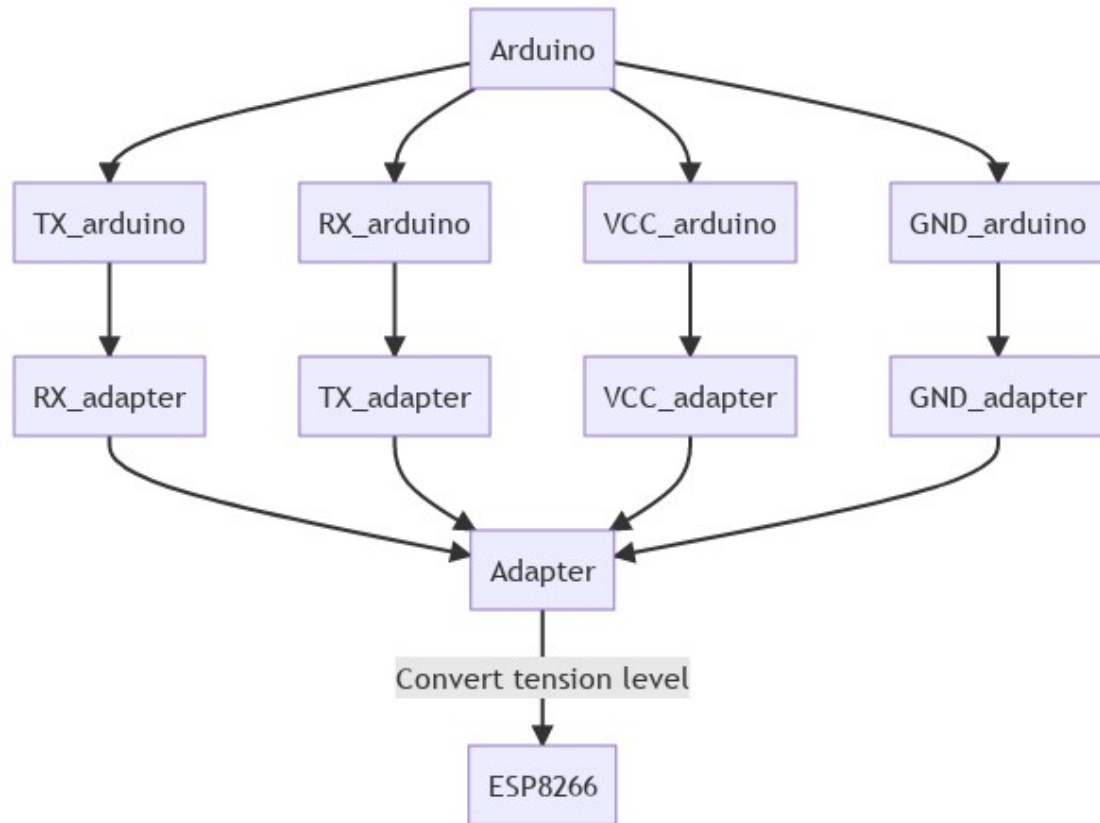
```
graph TD;
    Arduino --> VCC_arduino;
    Arduino --> GND_arduino;
    Arduino --> TX_arduino;
    Arduino --> RX_arduino;
    RX_arduino --> TX_adapter;
    TX_arduino --> RX_adapter;
    VCC_arduino --> VCC_adapter;
    GND_arduino --> GND_adapter;
    RX_adapter --> Adapter;
    TX_adapter --> Adapter;
    VCC_adapter --> Adapter;
    GND_adapter --> Adapter;
    Adapter -- Convert tension level--> ESP8266;
```

- VCC to VCC
- GND to GND
- **RX to TX**
- **TX to RX**

GAME 2

Now that the wiring is done we will send our first AT command.

This is a text protocol that will allow you to give orders to the ESP and communicate information with it. Note that this is not the only way to use ESP. You could program it directly to embed its own code like the arduino.

The default mode is communication via AT command and it is the simplest mode, so we will use this one. You can experiment with the other modes by yourself but be careful to always be able to restore the default firmware so as not to brick the ESP.

AT command on wikipedia
ESP8266 – AT Command Reference

To complete the first task send "AT" from the arduino, to receive the response "OK" from the ESP. This is the test command that will confirm that the ESP is working properly and that the two modules are able to communicate with each other.
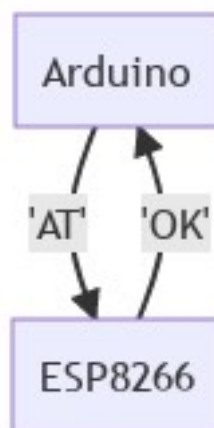
```
graph TD;
    Arduino -- "'AT'" --> ESP8266;
    ESP8266 -- "'OK'" --> Arduino
```
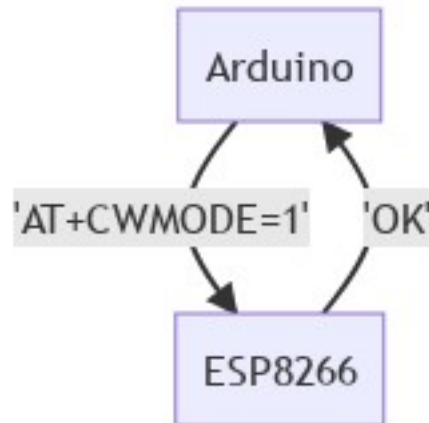
Game 3

Setup ESP in "station mode" to connect to Wifi network

The ESP can operate in 3 modes (see the doc for more information), the mode that interests us is the station mode which allows you to connect to a Wifi AP.
Note that if you can use your phone's 4g in access point mode to deploy a Wifi network.
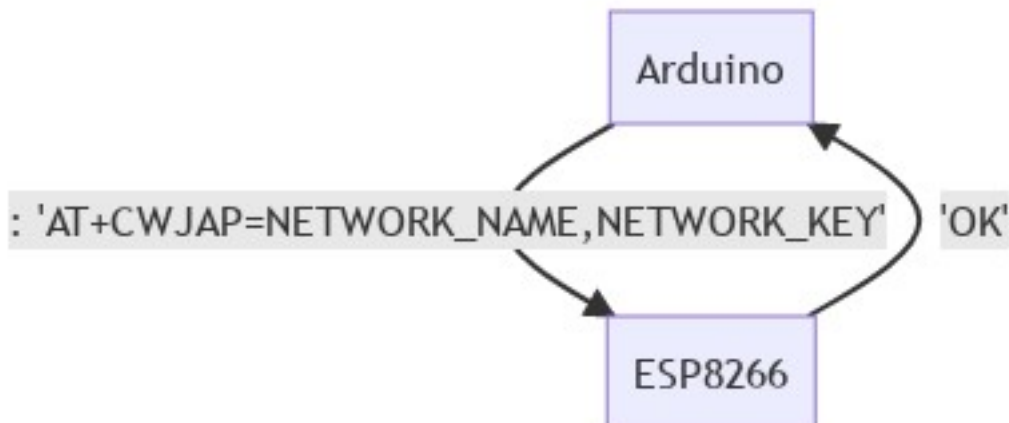
Configuring ESP in station mode :

```
graph TD;
    Arduino -- "'AT+CWMODE=1'" --> ESP8266;
    ESP8266 -- "'OK'" --> Arduino
```

Connecting ESP to the network :


```
graph TD;
    Arduino -- : 'AT+CWJAP=NETWORK_NAME,NETWORK_KEY' --> ESP8266;
    ESP8266 -- "'OK'" --> Arduino
```
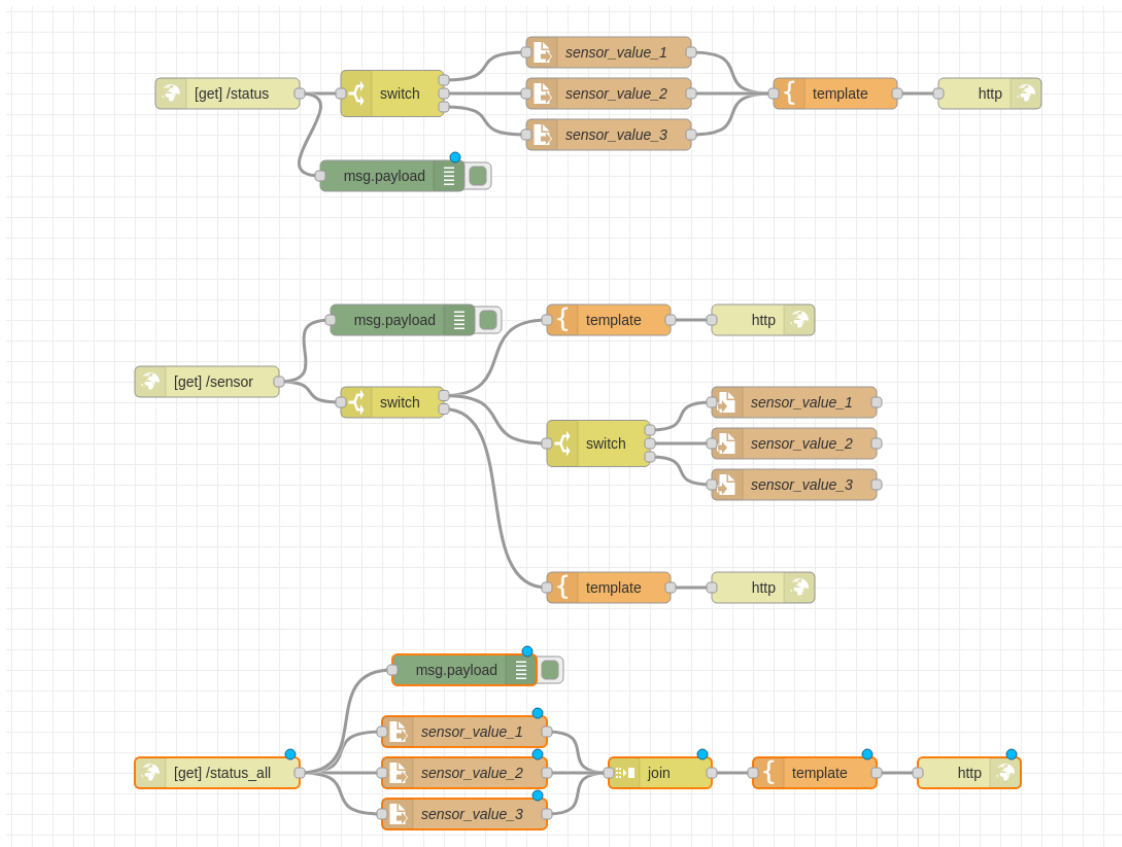
Of course you must replace "NETWORK_NAME" and "NETWORK_KEY" with your AP information (don't forget the char " as string delimiter)
Your system is now connected to the internet.

Game 4

Deploy API

Now to test the communication with the internet we are going to deploy locally an API which will be used for the tests and for the project.
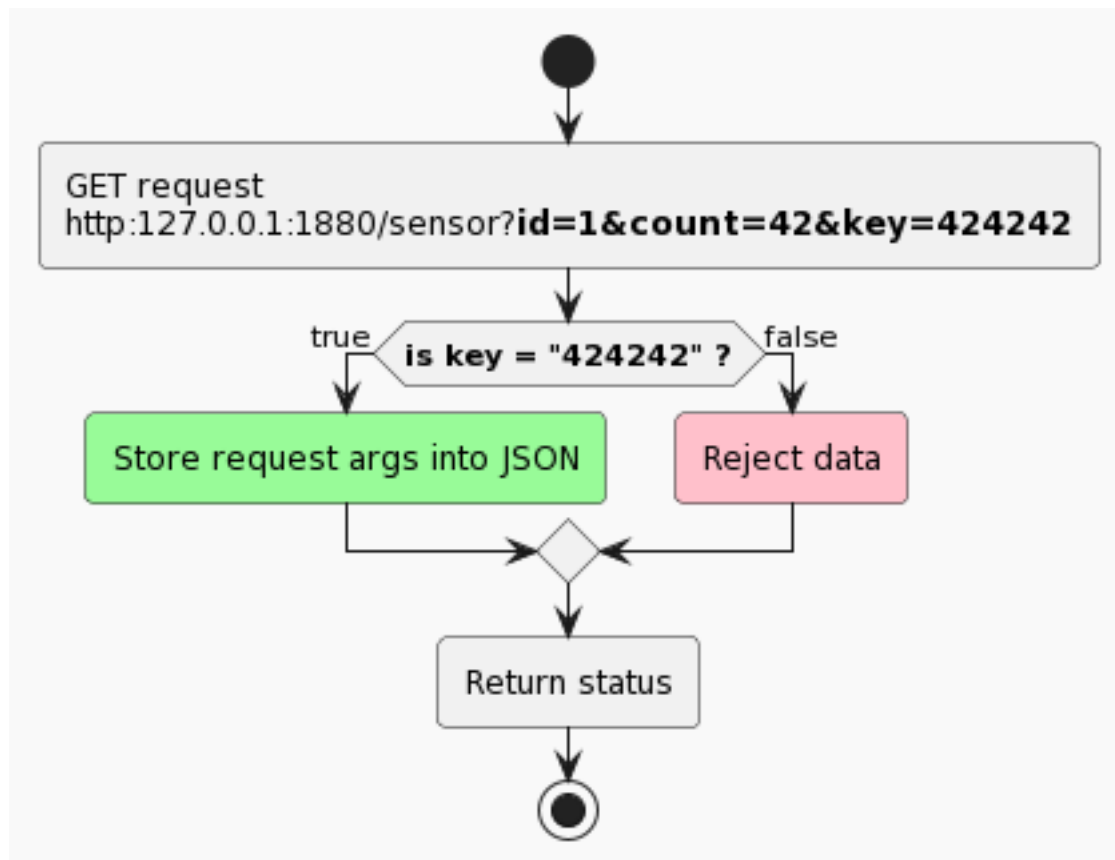
The API was made with Nodered for simplicity.

Thanks to NPM it should be easy to deploy on any laptop.

You will be able to implement your own API in the desired language during the project, **once everything else works correctly** (bonus).

You will find the corresponding flow.json file in this repo.

You will have to import it then click on "deploy" and the API will be normally functional (be careful to disable for firewall if necessary).

The API asks for an id and a key, then it will store all the arguments in a JSON file.
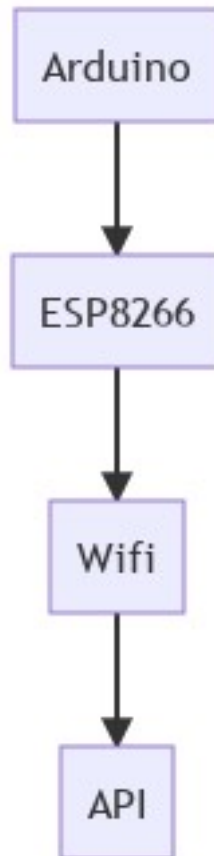
GET request
http:127.0.0.1:1880/sensor?**id=1&count=42&key=424242**

is key = "424242" ?

true

false

Store request args into JSON

Reject data

Return status

Game 5

Send GET request

Our setup should look like this :

```
graph TD;
    Arduino --> ESP8266;
    ESP8266 --> Wifi;
    Wifi --> API;
```

We are now going to send our first HTTP request.

To send our "GET" request we will use AT commands which allow us to send information through a TCP connection.
To establish tcp connection my code look like this :

```
Serial2.println("AT+CIPSTART="TCP","192.168.43.99",1880");
```

And to actually send data, it look like this :

```
String message = "GET /sensor?id=1&count=12&key=424242nn";
Serial2.println(String("AT+CIPSEND=") + String(message.length()));
delay(100);
Serial2.println(message.c_str());
```

You are now able to save data from the Arduino to the cloud and retrieve it from the application of your choice using the API, well done.

If you meet problems giving access from outside to your local API, I advise you to use ngrok.

Virtuality

In this new episode, we'll communicate between the two systems *Unity & Arduino* …
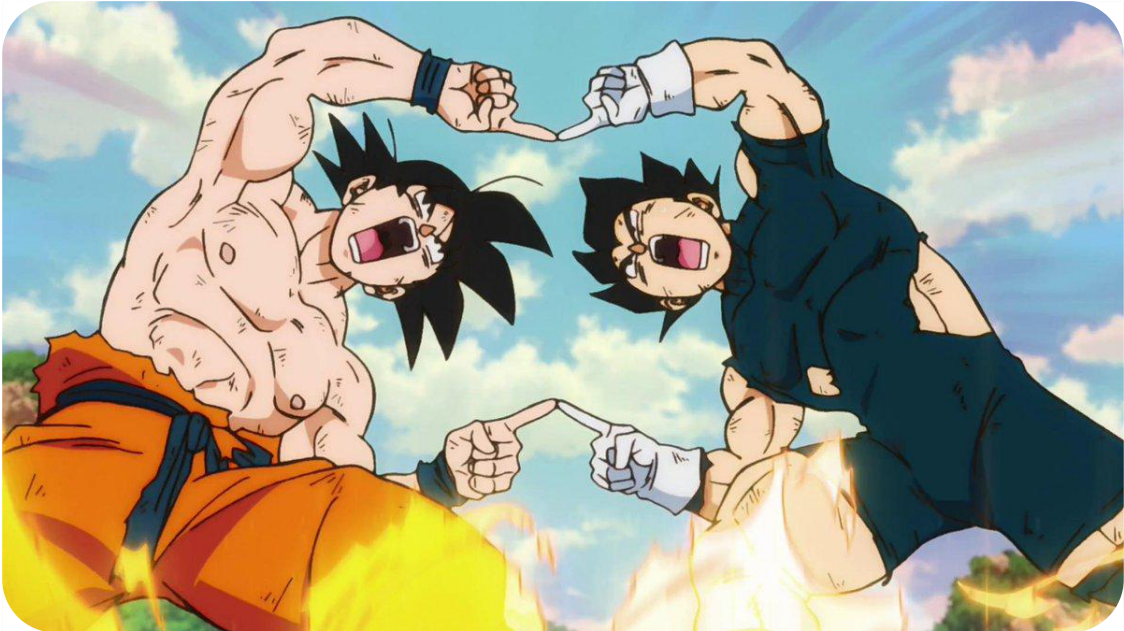
 An episode often begins too often with a summary of the previous one!

Game 1

Communicate an *Arduino* with *Unity* via the **serial**. A button connected to the *Arduino* sends a message to *Unity*, which is rewritten in the debug console.

Game 2

Connect an RGB LED to the *Arduino* and create *(or integrate)* a color picker at *Unity*. Change the color of the LED via *Unity* to confirm/deny your actions.
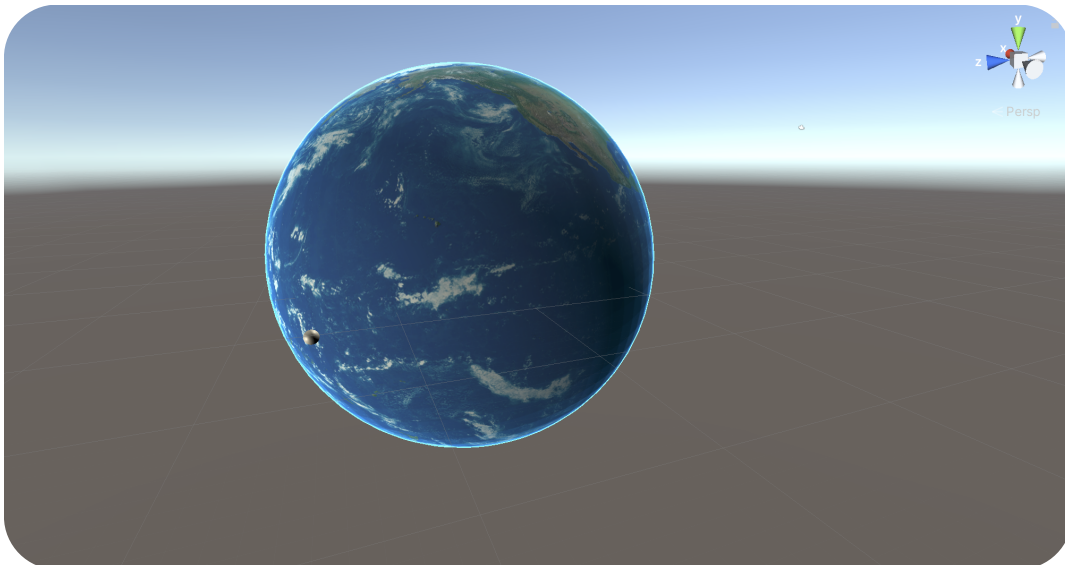
Game 3

Display a world map (2D or even better, globe).
On that map, display a Sphere at the location of the International Space Station *in real time*

You can find RESTful APIs that give you the position of the ISS, for instance on *http://api.open-notify.org/*
You'll also need to find a JSON parser in the asset store

You could check the API every seconds, or by the press of a button.

Game 4

The objective is the same as *Game 2*.

But this time, the arduino can't be connected to Unity via Serial.

You will communicate only via an API.

(You *could* use AirTable and Postman to build a no-code RESTfull API!
A no-code node-red API is also provided on this repo.