Internet of Things

Project Report

Enzo BESNAINOU
Marcus LEHEMBRE
Thomas LENFANT
Eloi POYET

M. OSMANI
M. HAMIDI

Links :

- Lab
- Projet - Home assistant
- Youtube

I. Labs (esp-32)

A. Summary

This project implements a simple collect-transmit schema. An ESP32 will be responsible of collecting measures from one or optionally multiple sensors, then send them to a server, in the same local network.

Communication is based on the Light-weight Internet Protocol (LwIP) stack, which is a small independent implementation of the TCP/IP protocol suite that has been initially developed by Adam Dunkels. This library is provided along with the ESP-IDF, so no need to include it.

Source code is provided but there are missing parts that you have to complete. These missing parts are those indicated with a comment of the form `// TODO failwith "Students, this is you job!"` or this instruction, `rc = ENOSYS;`. Note that `ENOSYS` is the *POSIX* error code for "Function not implemented".

Your job then, is to complete the missing parts in order to get the ESP32 collect measures from the `BME280`, format and send them correctly to the server via HTTP/TCP/IPv4/WiFi stack.

- GPIO Ports

SPI	GPIO_NUM	BME	PIN
CLK	18	*SCL*	D18
MOSI	3	*SDA*	RX0
CS	5	*CSB*	D5
MSIO	19	*SDO*	D19

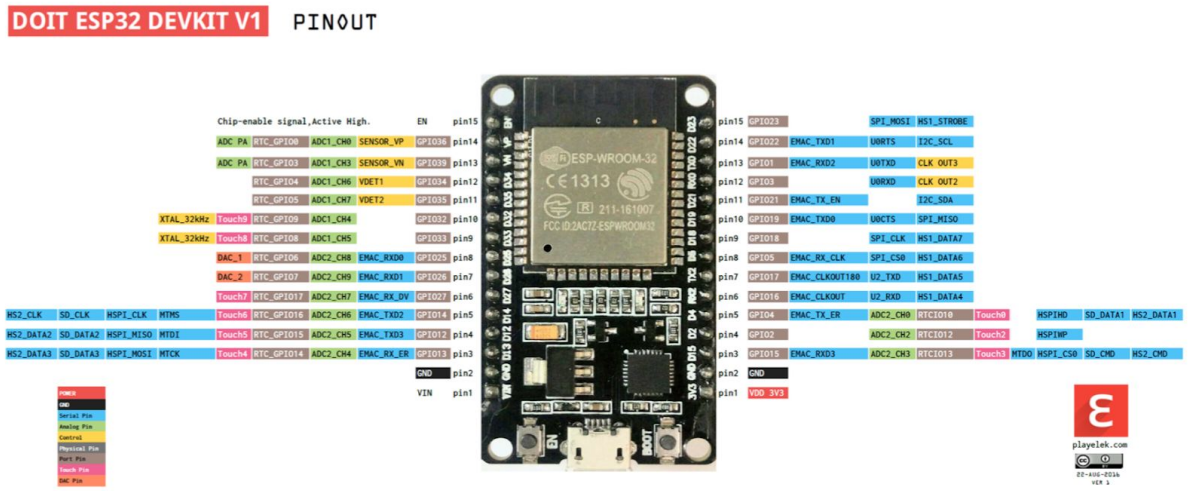
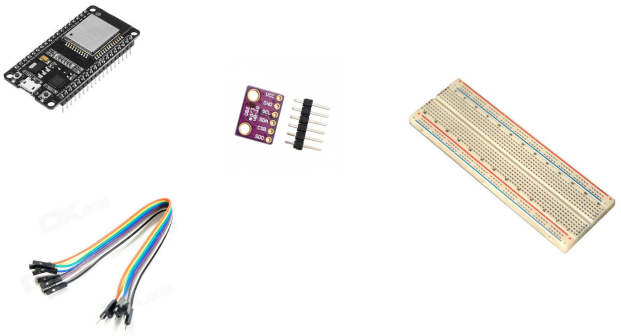


FIGURE 2.2 – ESP32 development board pinout.

B. Equipment used

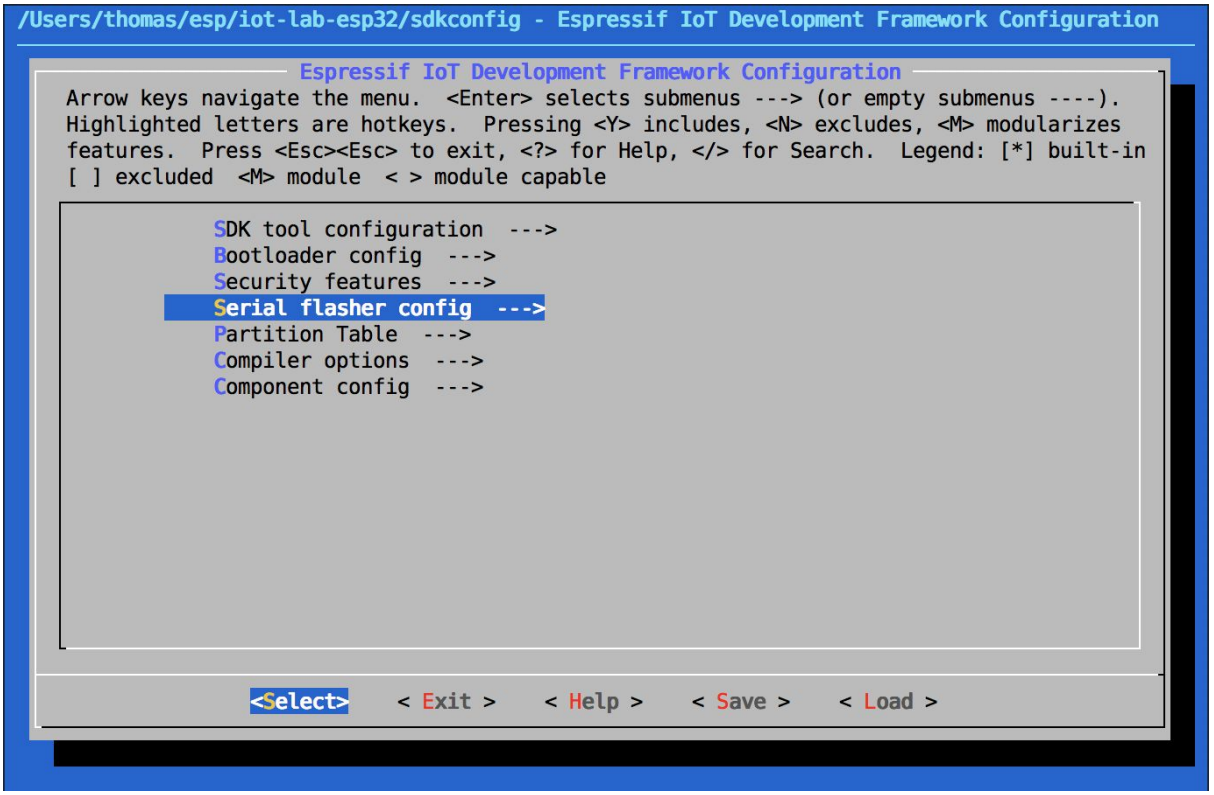


- ESP-32
- BME280 sensor
- Breadboard
- Cables

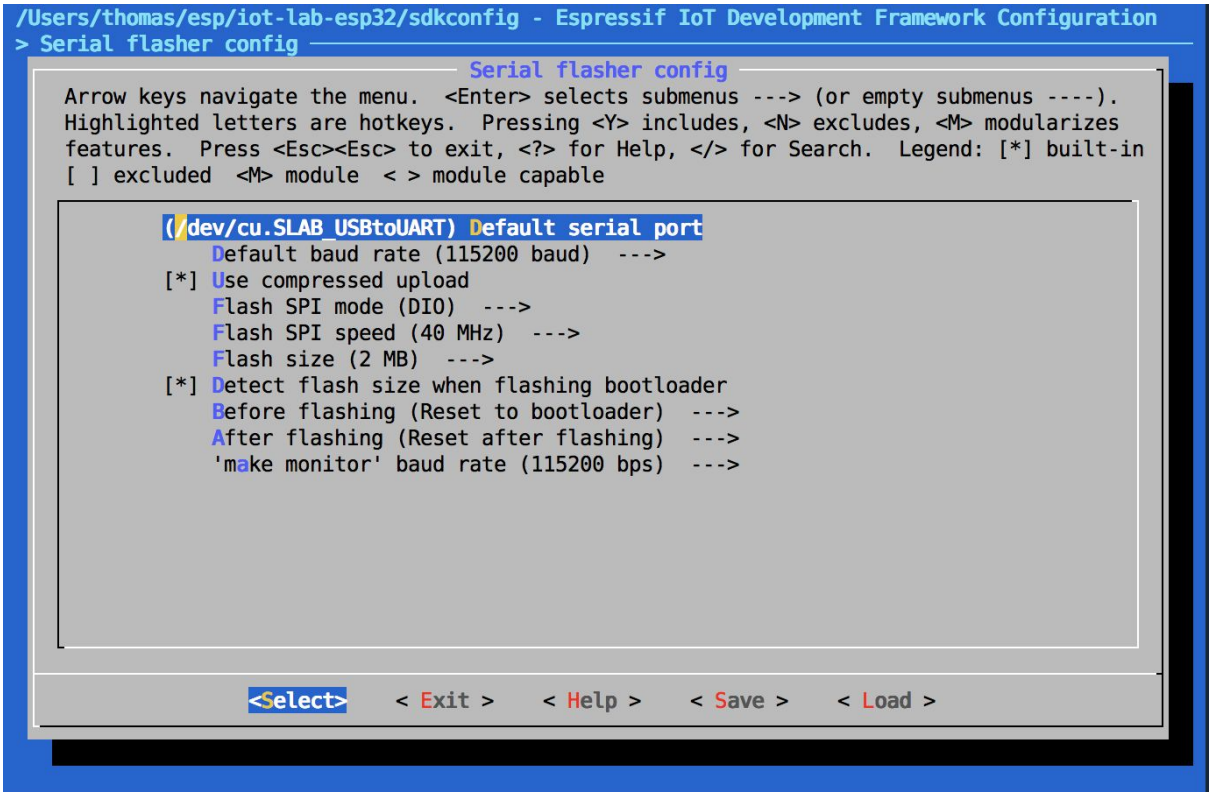
C. How to run it

First we need to set up a proper working environment. We need to have `python 2.7` installed. After that we follow the installation instructions detailed here.

```
$ cd $ESP_HOME/iot-lab-esp32
$ make menuconfig
```



Here we select our serial interface :

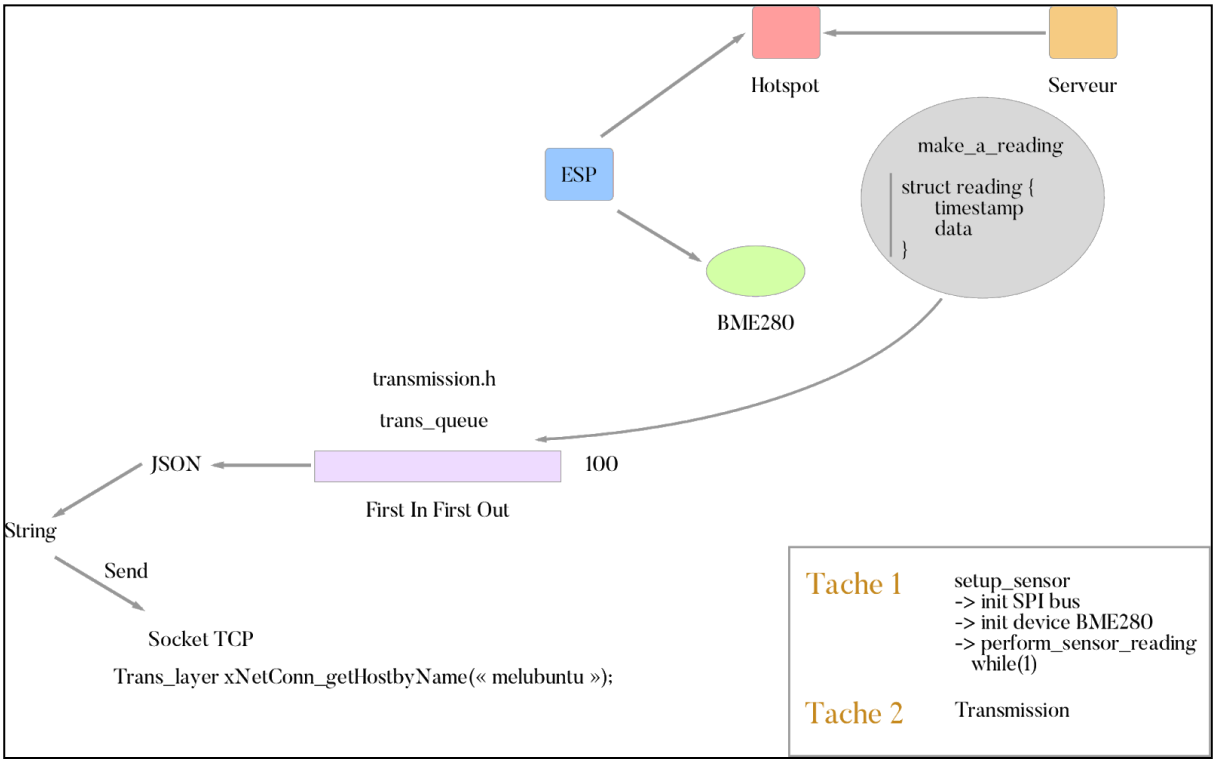


Then :

```
$ make flash monitor
```

The code compiled and the ESP is now up.

D. What we have made



We used the source code gracefully provided by M. Hamidi and we tried to complete it.
In summary, the ESP sends readings from the sensor to a remote server over TCP connection.

II. Personal Project (Personal assistant)

A. Summary

Our choice of connected object was towards the configuration of a personal assistant like Google Home, Alexa, Siri ...

To carry out this project we started from the open-source Jarvis tool which has all the necessary bases for the realization of our personal assistant. It is very customizable, so we can add many connected features ourselves.
The basic functionality of Jarvis is very limited: say hello, goodbye and repeat a sentence.

All the following features will be used in the video demo of our project.

B. Equipment used

- Raspberry Pi 2 on Raspbian (Jessie Full)
- Pi Model B+ Case (Black)
- Edimax EW-7611ULB WLAN/Bluetooth 150Mbit/s Network adapter
- VirtualBox on ElementaryOS (close to Raspbian commands)

We have no microphone and speaker compatible with Jarvis and Raspberry available so we gonna use a Virtual Machine for the presentation video but the whole project was develop and stable on raspberry 2.

C. How to run it

With a terminal or SSH, we need to execute the following command for the installation of Jarvis:

- Install git

```
sudo apt-get install -y git
```

- Clone our repo project : https://github.com/enzobes/iot-project

```
git clone https://github.com/enzobes/iot-project.git
```

- Go into iot-project folder and execute jarvis.sh

```
cd iot-project && ./jarvis.sh
```

- Install missing dependency if you need and follow installation step for configuring Jarvis speaker, microphone and local variables (username, jarvis name…)

Video installation:

- http://showterm.io/01d29a3a181f1f5345e14
- Reboot your system
- Configure speaker and mic in Settings→ Audio and follow the steps

D. What we have made

Voice recognition:

For voice recognition we used the microsoft one:

https://azure.microsoft.com/en-us/try/cognitive-services/

So we asked for an API key for "Speech"> "Bing Speech API"
Key 1: 565855a21fd5474dba477c3acc18f7b9
Key 2: eac2560a1fb2410890108dc8a149f7f2

Then just enter one of the two keys in Jarvis configuration

- Base Commands

```
*AIDE*==jv_display_commands
*BONJOUR*|*SALUT*==say "Bonjour $username"
*COMMENT*APPELLE*==say "Je m'appelle $trigger"
*MERCI*==say "De rien"
*AU REVOIR*|*BYE*==say "Au revoir $username"; jv_exit
ANNULE*|TERMINE*==bypass=false; say "Ok"
ENCORE*==jv_repeat_last_command
*TEST*==say "Ca fonctionne!"
*VERSION*==say "Je suis en version $jv_version"
*REPETE (*) ET (*)==say "(1) (2)"
*CA VA*==say "Très bien et toi ça va?"
>*OUI*==say "ravi de l'entendre"
>*NON*|*PAS*==say "j'en suis navré"
*SENS*VIE*==say "42"
```

- Update the system

For our first feature we have configured jarvis to be able to update the system.

```
*MISE A JOUR*==say "Mise à jour en cours..." && sudo apt-get update && $
>*OUI*==say "Très, bien" && sudo apt-get -y upgrade && say "Mise à jour
installer !"
>*NON*== "Ok, je ne télécharge pas les MAJ"
```

- List of items

For the creation of a list by Jarvis and to lighten the file of commands it is possible to make calls to functions written in the file my-function.sh

my-function.sh

```
#!/usr/bin/env bash
# Optionally declare here custom functions/variables to use in commands
and hooks
read_file()
{
nl ~/liste.txt
}
```

This function will allow jarvis to display the list as a list by reading the text file line by line. Then in the command that will ask Jarvis to display the list we will call this function.

jarvis-commands.sh

```
AJOUTE (*) A LA LISTE*==echo "(1)" >> ~/liste.tx4t && say "J'ai ajouté
(1) à la liste"
*SUPPRIME*LA LISTE*|*VIDE*LA*LISTE*==say "Voulez-vous vraiment vider la
liste ?"
>*OUI*==rm ~/liste.txt && say "J'ai bien supprimé la liste"
>*NON*==say "Très bien je n'y touche pas"
*AFFICHE LA LISTE*|*LIS LA LISTE*|*QUOI*DANS*LA*LISTE*== say "Voici
votre liste:" && read_file
*ENVOI*LA*LISTE*MAIL*==mpack -s "Votre liste Jarvis" /home/pi/liste.txt
enzobes@gmail.com && say "Je vous transfère la liste par mail"
```

We can also ask Jarvis to send us the list on an email address thanks to the package ssmtp (https://doc.ubuntu-fr.org/ssmtp) which will allow us to send mail to an external SMTP server.

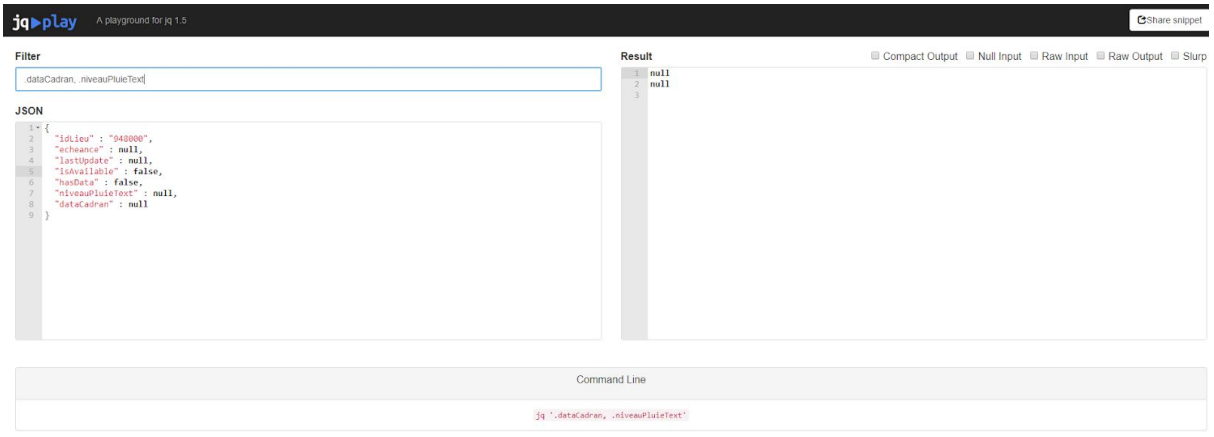
- Will it rain in one hour ?

Thanks to the service Meteo France it is possible to know if it will rain in the coming hour in a specific department:
http://www.meteofrance.com/previsions-meteo-france/previsions-pluie
More precisely thanks to the JSON code to return on this page:
http://www.meteofrance.com/mf3-rpc-portlet/rest/pluie/'..CityCode..'.json'
For Villejuif: http://www.meteofrance.com/mf3-rpc-portlet/rest/pluie/94800
Who sends us something of this type:

```
 "niveauPluieText" : [ "De12h35 à 13h35 : Pas de précipitations" ],
 "dataCadran" : [ {
    "niveauPluieText" : "Pas de précipitations",
    "niveauPluie" : 1,
    "color" : "ffffff"
  }
...
```

The two pieces of information needed for the weather forecast are `niveauPluieText` and `dataCadran`

Once parsed with jq (https://stedolan.github.io/jq/) it is possible to configure this service to Jarvis to ask if it will soon rain.



jarvis-commands.sh

```
*PLUIT*|*PLUIE*|*PLEUVOIR* == pluie_dans_une_heure
```

my-function.sh

```
jv_pg_pluie_lang () {
    case "$1" in
        no_data) echo "Pas de données disponibles.";;
```

```
        already_raining) echo "Il pleut déjà, ou il va pleuvoir dans les
5 prochaine minutes.";;
        rain_in) echo "Il va probablement pleuvoir dans
approximativement $2 minutes.";;
        no_rain) echo "Pas de pluie prévue dans l'heure.";;
    esac
}

pluie_dans_une_heure()
{
      local sum=0
      local has_unknowns=false
      local has_rain=false
      local iteration=0
      local iter_of_first_rain=0
      local code_insee='940760'

      local infos="$(curl -s
http://www.meteofrance.com/mf3-rpc-portlet/rest/pluie/${code_insee} | jq
'.["dataCadran"][]["niveauPluie"]')"
      jv_debug $infos

      for row in $infos; do
            if [ $row -eq 0 ]
            then
                  has_unknowns=true
            fi

            if [ $row -ge 2 ] && ! $has_rain
            then
                  has_rain=true
                  iter_of_first_rain=$iteration
            fi

            let iteration=iteration+1
            let sum=sum+row
      done


      if [ $sum -eq 0 ]
      then
            say "$(jv_pg_pluie_lang no_data)"
            return 0
      fi
```

```
	if $has_rain && [ $iter_of_first_rain -eq 0 ]
	then
		say "$(jv_pg_pluie_lang already_raining)"
		return 0
	fi

	if $has_rain
	then
		say "$(jv_pg_pluie_lang rain_in $((5*$iter_of_first_rain)))"
		return 0
	fi

	say "$(jv_pg_pluie_lang no_rain)"
	return 0

}
```

- Speed Test :

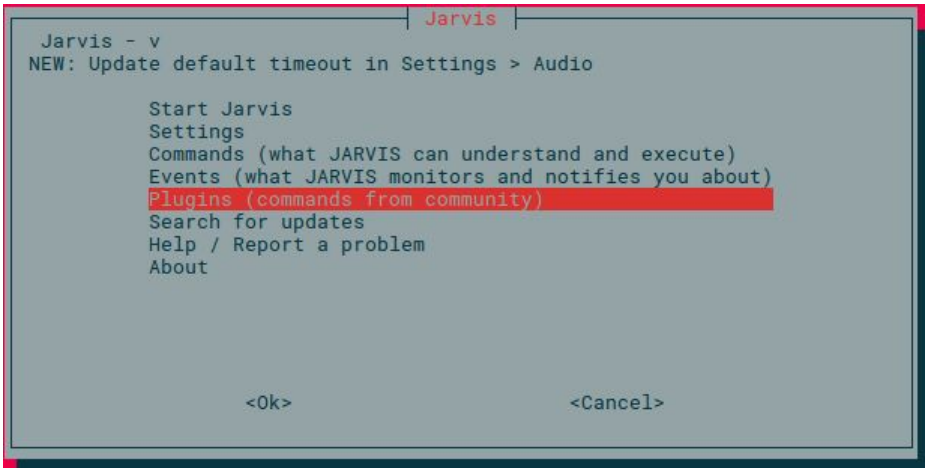
Require speedtest-cli (https://github.com/sivel/speedtest-cli)

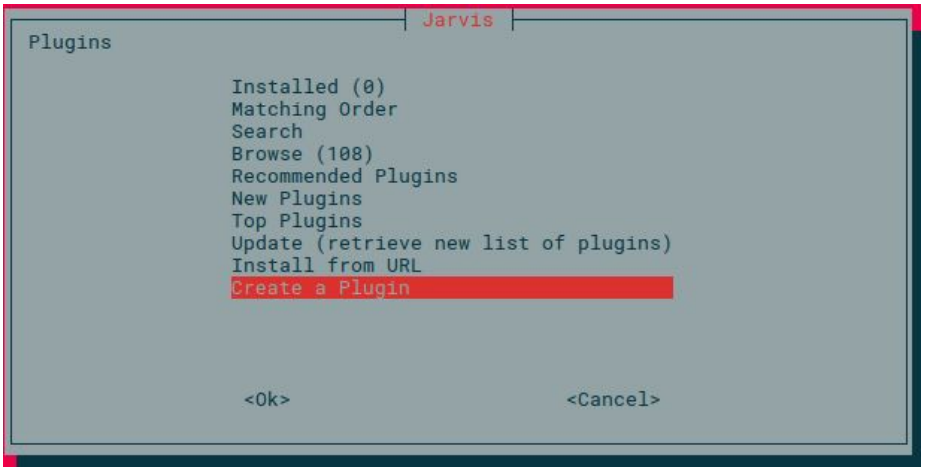
jarvis-commands.sh

```
*DEBIT*INTERNET*|*ADSL*|==say "Je test actuellement votre connexion ..."
&& speedtest-cli
```

- Plugins

Jarvis also offers many plugins developed by the community it is possible to install these plugins through the Jarvis interface:





Or directly from the command line from the Jarvis store (https://openjarvis.com/plugins)

1. **Google Map Traffic**

- Get Google Map Direction API
- Add plugin to Jarvis
  `./jarvis.sh -p https://github.com/QuentinCG/jarvis-google-map-traffic`
- Edit config.sh with API key and custom adresses if you want

```
# To get a Google Map direction API key:
https://developers.google.com/maps/documentation/directions/get-api-key
var_jv_pg_gm_api_key="AIzaSyA7M1uVVdriVFo6Myu4KL-jjiw1WGx2o40"

# Base route
var_jv_pg_gm_from="NICE" # From address (You can enter complete addresses)
var_jv_pg_gm_to="PARIS" # Destination address (You can enter complete addresses)
```

2. **Wikipedia**

Ask Jarvis definitions for words or phrases from the Wikipedia site.

Installation:

```
./jarvis.sh -p https://github.com/Sellig28/jarvis-wikipedia
```

3. **Weather**

Based on weather data from https://www.prevision-meteo.ch and data parsed in JSON, this plugin makes it possible to give more precise indications on the weather than the one developed previously which makes it possible to know only the forecast for the rain in the hour to come

Installation:

```
./jarvis.sh -p https://github.com/Sh1n1x/jarvis-meteo-suisse
```

E. Video link

Demo video:
https://www.youtube.com/watch?v=XRGRwMrI4VQ

F. Ref - Libraries

- https://github.com/Cqoicebordel/jarvis-pluie-a-une-heure/blob/master/functions.sh
- https://openjarvis.com/
- https://github.com/Cqoicebordel/jarvis-pluie-a-une-heure
- https://jqplay.org/
- https://www.raspberrypi.org/

III. Work Distribution

<table>
<tr><th></th><th>Labs</th><th>Jarvis</th></tr>
<tr><td>Marcus LEHEMBRE</td><td>2 first labs contribution, research on documentation and specs of the board (which outputs, GPIO ports for examples), researches on the wifi sync, data sensor lecture</td><td>Contribution on some functionalities like Will it rain in one hour ?, researches on Jarvis for setting up</td></tr>
<tr><td>Thomas LENFANT</td><td>Code for the whole lab</td><td></td></tr>
<tr><td>Eloi POYET</td><td>-Mainly research on the documentation and specifications of the board
-Creation and layout of the report
-Assistance with cabling and some fragments of code</td><td rowspan="2">-Research for a new project as asked using raspberry or arduino for a connected object.
-Buy a Raspberry and the necessary components (toggle wifi, cables, radiators for CPU and RAM)
-Install and setup jarvis and its components</td></tr>
<tr><td>Enzo BESNAINOU</td><td>Buy components:
- Elegoo Kit
- ESP 32
Help to fix bug on blink led and measurement with sensor</td></tr>
</table>