ENSEIRB-MATMECA

CONCEPTION D'OBJETS CONNECTES PAR PROTOTYPAGE RAPIDE

Patrice KADIONIK
kadionik.enseirb-matmeca.fr

TABLE DES MATIERES

1. BUT DES TRAVAUX PRATIQUES

Ces Travaux Pratiques ont pour but de présenter la conception d'objets connectés par prototypage rapide.

Comme nous avons pu le voir en cours, le logiciel libre et le matériel libre sont les composants de base essentiels dans cette conception.

Nous n'allons pas réaliser la conception matérielle d'un objet connecté, nous partirons donc de matériels conçus pour le prototypage rapide et nous nous attacherons à la conception logicielle.

Les objets connectés contiennent généralement un système d'exploitation. On retrouve dans l'immense majorité des cas un socle Linux embarqué. Il est possible d'utiliser ce socle avec un langage de programmation comme C ou Python mais on peut aussi trouver au-dessus une machine virtuelle comme Java ou même Android.

Linux est donc incontournable dans ce domaine. Il est aussi possible d'avoir un système d'exploitation Temps Réel comme FreeRTOS (logiciel libre) très largement employé dans la conception d'objets connectés voire même du pur langage C embarqué (*bare metal*).

Les deux premiers TP se feront autour d'une carte Raspberry Pi complétée d'une carte fille ou « *HAT* » incluant différents capteurs : le *Sense HAT*. Celui-ci contient des capteurs de pression, température, humidité relative et de mesure d'accélération ainsi qu'un *joystick* et un gyroscope. Nous nous limiterons aux 2 premiers capteurs de la liste.
Le troisième TP se fera autour d'une carte Raspberry Pi Pico W incluant différents capteurs.

Un premier TP est consacré à la mise en œuvre de Python et du *Sense HAT* sur carte Raspberry Pi pour un contrôle local puis pour un contrôle distant par HTTP.
Un deuxième TP est consacré à la mise en œuvre du protocole MQTT devenu standard de fait dans l'Internet des objets.
Enfin, un troisième TP est consacré à la mis en œuvre de l'environnement MicroPython sur une carte Raspberry Pi Pico W avec le langage MicroPython pour un contrôle local puis pour un contrôle distant par MQTT. Un *dashboard* sera créé pour représenter l'évolution au cours du temps de la température et de la pression de son objet connecté (*time serie*).

Mots clés : objet connecté, Raspberry Pi, Sense HAT, Raspberry Pi Pico W, Maker Pi Pico, Raspberry Pi OS, langage Python, environnement MicroPython, langage MicroPython, MQTT, *broker* Adafruit

2. TP 1 : OBJET CONNECTE A BASE DE CARTE RASPBERRY PI

2.1. Carte cible Raspberry Pi

La carte Raspberry Pi ou RPi est une carte bon marché largement utilisée pour le DIY (*Do It Yourself*) afin de développer de petits systèmes embarqués ou des objets connectés. Le modèle mis en œuvre dans ces TP est la carte Raspberry Pi 3B+.

Notre carte RPi possède ainsi les éléments suivants :

- Un SoC (*System on Chip*) Broadcom BCM2837 avec un processeur quadricœur ARM Cortex-A53 à 1,2 GHz.
- 1 Go de RAM.
- 4 ports USB.
- Sortie vidéo HDMI.
- Sortie audio HDMI et Jack 3,5 mm.
- Support microSD.
- Ethernet 10/100 Mb/s, Wifi 802.11n et Bluetooth 4.1.
- 7 E/S GPIO, 1 UART, 1 bus I2C et 1 bus SPI.

L'image suivante présente la carte cible RPi 3B+ :

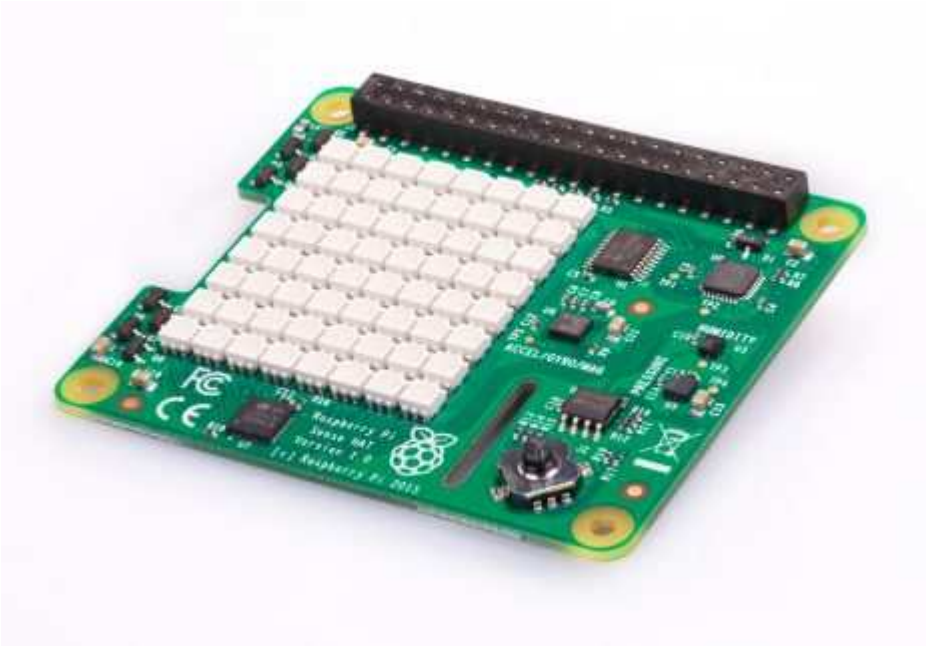
Carte cible RPi 3B+

La carte RPi 3B+ est complétée d'un *HAT*, le *Sense HAT* pour rajouter différents capteurs et un affichage.

La carte *Sense HAT* possède ainsi les éléments suivants :

- 1 gyroscope.
- 1 accéléromètre.
- 1 capteur d'accélération (2/4/8/16 g).
- 1 magnétomètre (4/8/12/16 Gauss).
- 1 baromètre (260 - 1260 hPa).
- 1 capteur de température (0-65°C).
- 1 capteur d'humidité relative (20-80% rH).
- 1 affichage par leds avec une matrice de 8x8.
- 1 *joystick* 5 boutons.

L'image suivante présente la carte *Sense HAT* :



Carte *Sense HAT*

2.2. Environnement de développement

Nous allons utiliser une distribution Linux officielle de la carte cible Raspberry Pi comprenant tous les outils nécessaires pour un développement natif directement sur la carte RPi. Pour cela, la distribution Raspberry Pi OS est utilisée et a été installée et configurée sur la carte microSD de la carte RPi.

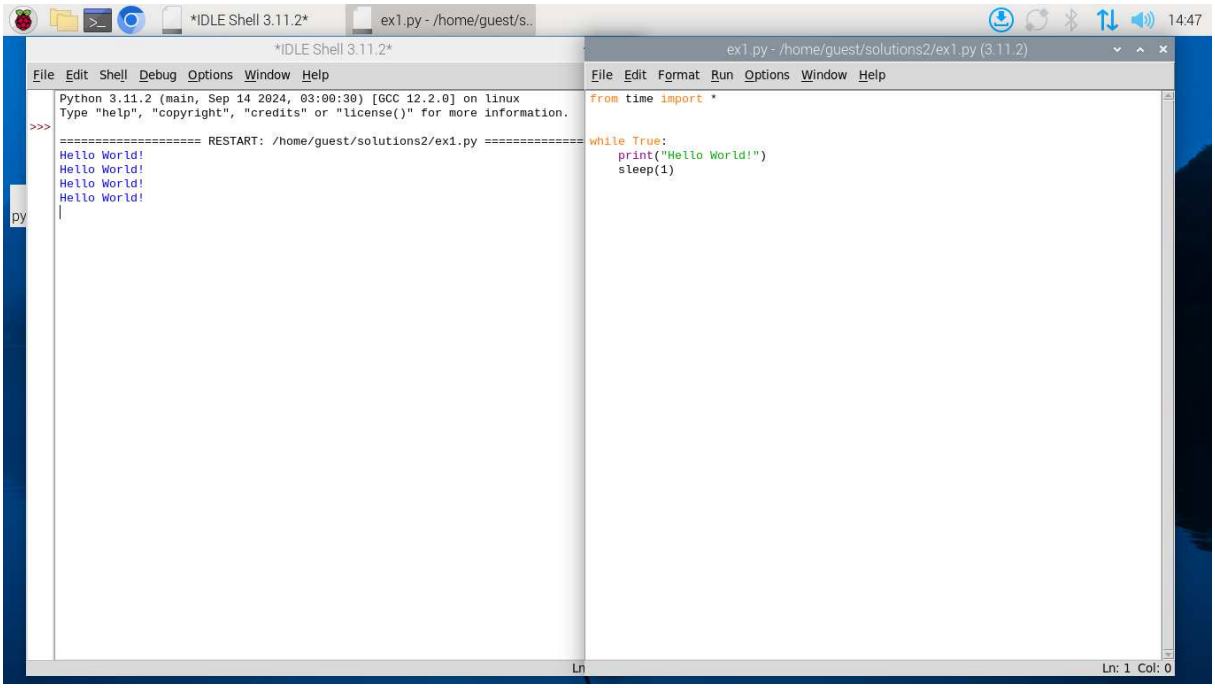
Nous utiliserons le langage C mais aussi le langage Python plus adapté à un contexte de prototypage rapide.

La distribution Raspberry Pi OS propose un environnement graphique de bureau comme le montre la figure suivante :

Bureau de Raspberry Pi OS

De nombreuses bibliothèques Python permettent de développer simplement sur la carte RPi. Il existe par exemple la bibliothèque `SenseHat` pour piloter très simplement la carte *Sense HAT*.

Si l'on clique sur l'icône IDLE Python 3 (icône sur le bureau), on a alors accès à un IDE (*Integrated Design Entry*) de développement Python. On peut éditer un fichier Python (menu *File*), exécuter un fichier Python (menu *Run* ou touche F5), relancer l'interpréteur Python (CTRL F6)…



IDE IDLE 3 (pour Python 3)

L'IDE IDLE 3 est simple et intuitif. Le langage Python est par ailleurs déjà étudié en classes préparatoires aux grandes écoles…

Dès lors, nous allons développer des programmes Python avec l'IDE IDLE 3.

Pour avoir accès à la bibliothèque `SenseHat`, le fichier source Python aura la structure suivante :

```
from sense_hat import SenseHat

sense = SenseHat()
. . .
sense.show_message("Hello world!", back_colour=[0, 0, 255])
print(("Hello world!")
. . .
print("TEMP=%02.1f" % temperature)
. . .
```

L'API de la bibliothèque `SenseHat` est décrite dans les annexes 1 et 2.

On peut alors avoir accès à l'afficheur matriciel à leds et aux capteurs de la carte *Sense HAT*…

2.3. EX 1 : application Hello World

Nous allons écrire en langage Python la célèbre application « *Hello World!* ».

- Se créer un répertoire de travail à son nom et s'y placer :
```
host% cd
host% mkdir mon_nom
host% cd mon_nom
```

- Ecrire en langage Python le programme `tp1.py` d'affichage à l'écran (fonction Python `print()`) et sur l'afficheur matriciel à leds de *Hello World!* ». Tester.

2.4. EX 2 : capteurs de température et de pression

- Ecrire en langage Python le programme `tp2.py` d'affichage à l'écran de la pression et de la température de la carte *Sense HAT*. Afficher sur l'afficheur matriciel à leds la température. On affichera les valeurs avec un « chiffre après la virgule » (à 10^{-1} près). Tester.

2.5. EX 3 : *threads* et classes Python

Python implémente l'API *thread* de façon native et on a accès à l'API via la directive :
`import threading`

- Ecrire en langage Python le programme `tp3.py` de création de 2 *threads*. On définira une classe de *thread* `task_affiche` qui dans sa méthode `run` affiche à l'écran toutes les 2 secondes le nom de l'objet créé à partir de cette classe. Au bout de 10 secondes, les 2 *threads* seront arrêtés. Tester.

La structure générale de la classe `task_affiche` et donc du programme `tp3.py` est :

```
from sense_hat import SenseHat
from time import *
import threading

sense = SenseHat()

class task_affiche(threading.Thread):
    def __init__(self, nom = ''):
        threading.Thread.__init__(self)
        self.nom = nom
        self.Terminated = False

    def run(self):
        while not self.Terminated:
            . . .

    def stop(self):
        self.Terminated = True

t1 = task_affiche("Task t1")
t1.start()
```

2.6. EX 4 : *threads* et capteurs

- Ecrire en langage Python le programme `tp4.py` de création de 2 *threads*. La classe de *thread* `task_pression` pour le capteur de pression et la classe de *thread* `task_temp` pour le capteur de température seront créées. La pression sera affichée à l'écran toutes les 2 secondes tandis que la température le sera toutes les secondes.

2.7. EX 5 : protocole HTTP. Miniserveur Web

Python implémente l'API *sockets* de façon native et on a accès à l'API via la directive :
`import socket`

La structure générale d'un miniserveur Web correspond à celle d'un serveur TCP qui accepte une connexion TCP entrante et qui renvoie alors une réponse HTTP forgée manuellement. La réponse se compose d'une entête HTTP classique puis d'un saut de ligne et enfin des données codées en HTML correspondant à la page d'accueil du miniserveur Web.

Le canevas en langage Python du miniserveur Web est donc le suivant (les **???** sont à remplacer par les bonnes valeurs…) :

```
from sense_hat import SenseHat
from time import *
import threading
import socket

sense = SenseHat()

socket = socket.socket(socket.AF_???, socket.SOCK_???)

socket.bind(('', ???))

socket.listen(1)

while True:
    client, address = socket.accept()

    request = client.recv(1024)

    client.send(str.encode("HTTP/1.1 ???\n"))
    client.send(str.encode("Content-Type: ???\n"))

    client.send(str.encode("\n"))

    client.send(str.encode("<CENTER><H1>Welcome to the rpi-python Web
Server</H1></CENTER>"))
    client.send(str.encode("Hello World!"))

    client.close()
```

- Ecrire en langage Python le programme `tp5.py` de création d'un miniserveur Web en écoute sur le port 8080. Tester avec le navigateur Web.

2.8. EX 6 : protocole HTTP. Miniserveur Web et capteurs

- Ecrire en langage Python le programme `tp6.py` de création d'un miniserveur Web en écoute sur le port 8080 qui renvoie la température et la pression. Tester avec le navigateur Web. On pourra rajouter la balise HTML suivante pour forcer le rafraîchissement de la page Web toutes les secondes :
  `<meta http-equiv="Refresh" content="1">`

2.9. EX 7 : miniprojet

- Ecrire en langage Python le programme `miniprojet.py` de synthèse de TP précédents.
  - On créera une classe de *thread* `task_sensor` d'acquisition des capteurs de température et de pression.
  - On créera une classe de *thread* `task_led` d'affichage de la température courante sur l'afficheur matriciel à leds.
  - On créera une classe de *thread* `task_www` d'implémentation d'un miniserveur Web qui renvoie la température courante et la pression courante. On pourra rajouter la balise HTML suivante pour forcer le rafraîchissement de la page Web toutes les secondes :
    `<meta http-equiv="Refresh" content="1">`

Une variable globale (à plusieurs *threads*) se déclare en langage Python comme suit :

```
temp = 0

class task_t1(threading.Thread):
      def __init__(self, nom = ''):
            threading.Thread.__init__(self)
            . . .
      def run(self):
            global temp
            while not self.Terminated:
                  . . .
                  temp = lecture_temp()
                  . . .
      . . .

class task_t2(threading.Thread):
      def __init__(self, nom = ''):
            threading.Thread.__init__(self)
            . . .
      def run(self):
            global temp
            while not self.Terminated:
                  while (True):
                        client, address = self.socket.accept()
                        . . .
                        client.send(str.encode("TEMP=%02.1f °C" % temp))
                        . . .
      . . .
. . .
```

- Tester avec le navigateur Web.

3. TP 2 : MISE EN ŒUVRE DU PROTOCOLE MQTT

Ce TP est présenté dans un autre document…

4. TP 3 : OBJECT CONNECTÉ À BASE DE CARTE RASPBERRY PI PICO W

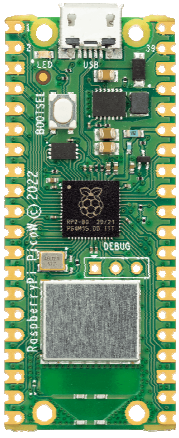
4.1. Carte cible Raspberry Pi Pico W

La carte RPi-Pico W est une carte permettant de concevoir des objets connectés bon marché et qui est utilisée pour le DIY (*Do It Yourself*) afin de développer de petits systèmes embarqués ou des objets connectés.

La carte RPi-Pico W possède ainsi les éléments suivants :

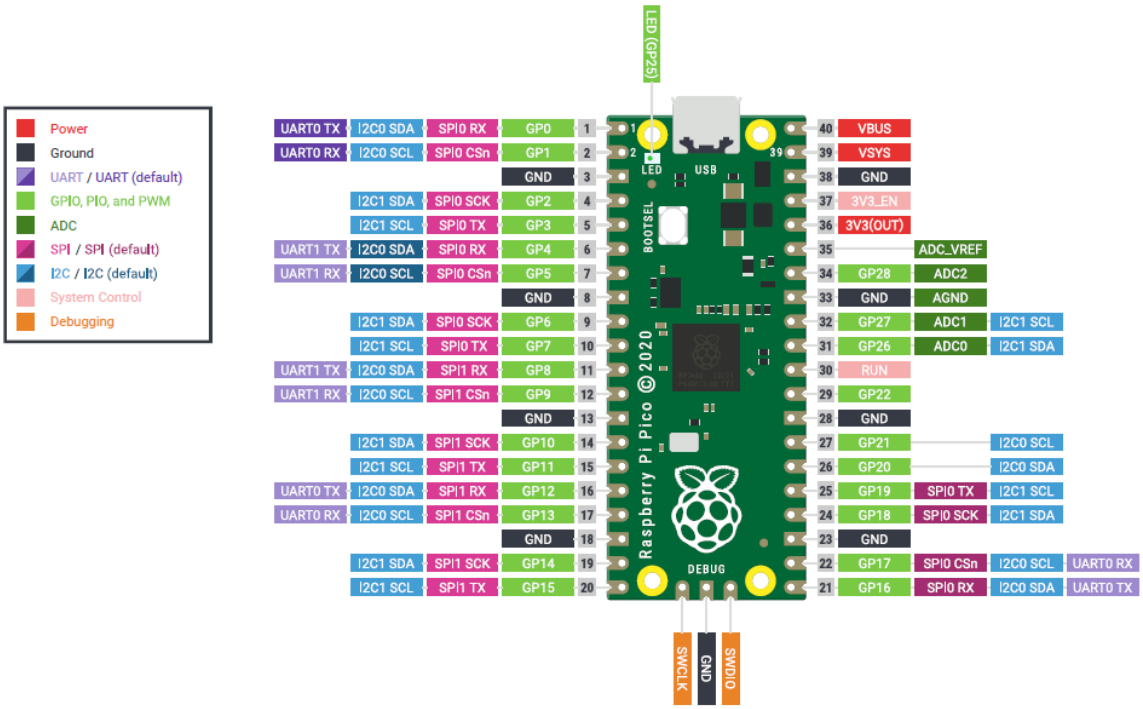
- Processeur ARM Cortex-M0+ *Dual Core* à 133 MHz (SRAM de 264 Ko et Flash de 2 Mo).
- Connectivité Wifi IEEE 802.11b/g/n avec le composant Infineon CYW43439.
- Alimentation 5 V via le connecteur micro USB.
- 26 ports d'entrées/sorties (E/S) comprenant :
    - 3 entrées analogique/numérique (12 bits).
    - 2 ports UART.
    - 2 bus I2C.
    - 16 ports PWM.
- 1 interface SWD de *debug*.
- Horloge RTC intégrée.
- 1 led programmable.
- 1 capteur de température intégré.

L'image suivante présente la carte cible RPi-Pico W :



Carte RPi-Pico W

L'image suivante présente le brochage de la carte cible RPi-Pico W :



Brochage de la carte RPi-Pico W

La carte RPi-Pico W est complétée d'une carte d'accueil, la carte *Maker Pi Pico* pour rajouter différents capteurs et E/S.

La carte *Maker Pi Pico* possède ainsi les éléments suivants :

- Leds sur les E/S GPIO.
- 1 led de couleur connectée sur l'E/S GP28.
- 3 boutons connectés sur les E/S GP20, GP21 et GP22.
- 1 *buzzer* piezo connecté sur l'E/S GP18.
- 1 sortie audio connectée sur les E/S GP18 (L) et GP19 (R).
- 1 *socket* ESP-01.
- 1 *socket* Micro SD sur les E/S GP10 à GP15.

L'image suivante présente la carte *Maker Pi Pico* :

Carte *Maker Pi Pico*

L'ensemble est aussi complété d'un capteur connecté sur le connecteur Grove numéro 0 : capteur BME680.

Le capteur BME680 muni d'une interface I2C propose en fait un ensemble de capteurs :

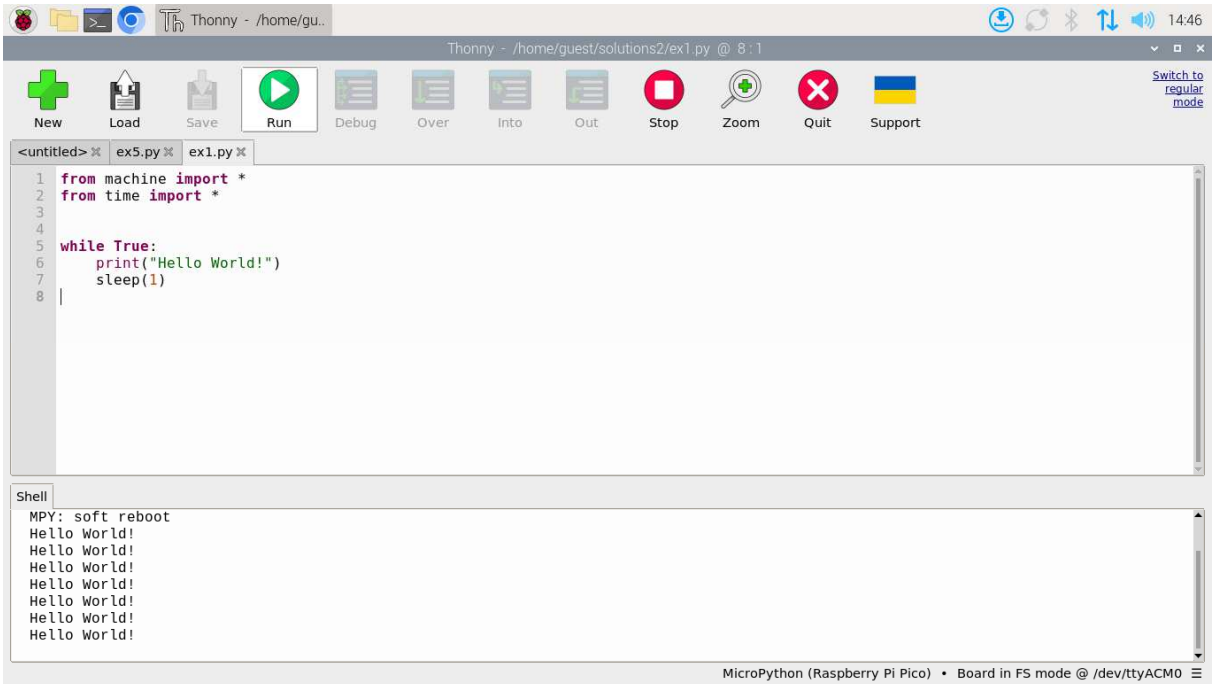
- Température : -40 °C à + 85°C avec une résolution de 0,01 °C.
- Humidité : de 0 à 100 % d'Humidité Relative avec une résolution de 0,008 % HR.
- Pression : de 300 à 1100 hPa avec une résolution de 0,18 Pa.

4.2. Environnement de développement

De nombreuses bibliothèques MicroPython permettent de développer simplement sur la carte RPi-Pico W.

Si l'on clique sur l'icône IDLE Thonny (icône sur le bureau), on a alors accès à un IDE (*Integrated Design Entry*) de développement Python et MicroPython.

Thonny a été configuré pour s'interfacer à la carte RPi-Pico W très simplement et avoir accès à l'interpréteur REPL de MicroPython via la liaison série.



IDE Thonny

L'IDE Thonny est simple et intuitif. La partie supérieure permet d'éditer du code MicroPython sur l'ordinateur. La partie inférieure donne accès à l'interpréteur REPL. L'icône du « triangle vert » permet d'exécuter un programme MicroPython sur la carte cible RPi-Pico W.

Dès lors, nous allons développer des programmes MicroPython avec l'IDE Thonny.

On pourra aussi s'aider des exemples donnés en cours…

4.3. EX 1 : application Hello World

Nous allons écrire en langage MicroPython la célèbre application « *Hello World!* ».

- Se créer un répertoire de travail à son nom et s'y placer :
```
host% cd
host% mkdir mon_nom
host% cd mon_nom
```

- Ecrire en langage Python le programme `tp1.py` d'affichage à l'écran de « *Hello World!* ». Tester.

4.4. EX 2 : application Hello World par led

- Ecrire en langage MicroPython le programme `tp2.py` qui permet de faire clignoter la led connectée sur l'E/S GP10 toutes les secondes. Tester.

4.5. EX 3 : capteurs de température et de pression

- Ecrire le programme `tp3.py` d'affichage à l'écran de la pression et de la température de la carte RPi-Pico W. On affichera les valeurs avec un « chiffre après la virgule » (à 10^{-1} près). Tester.

4.6. EX 4 : Wifi

- Ecrire le programme `tp4.py` qui permet de se connecter au réseau Wifi « SE » avec le mot de passe donné en TP. Quelle est la configuration Internet de la carte RPi-Pico W. Tester.

4.7. EX 5 : Broker MQTT

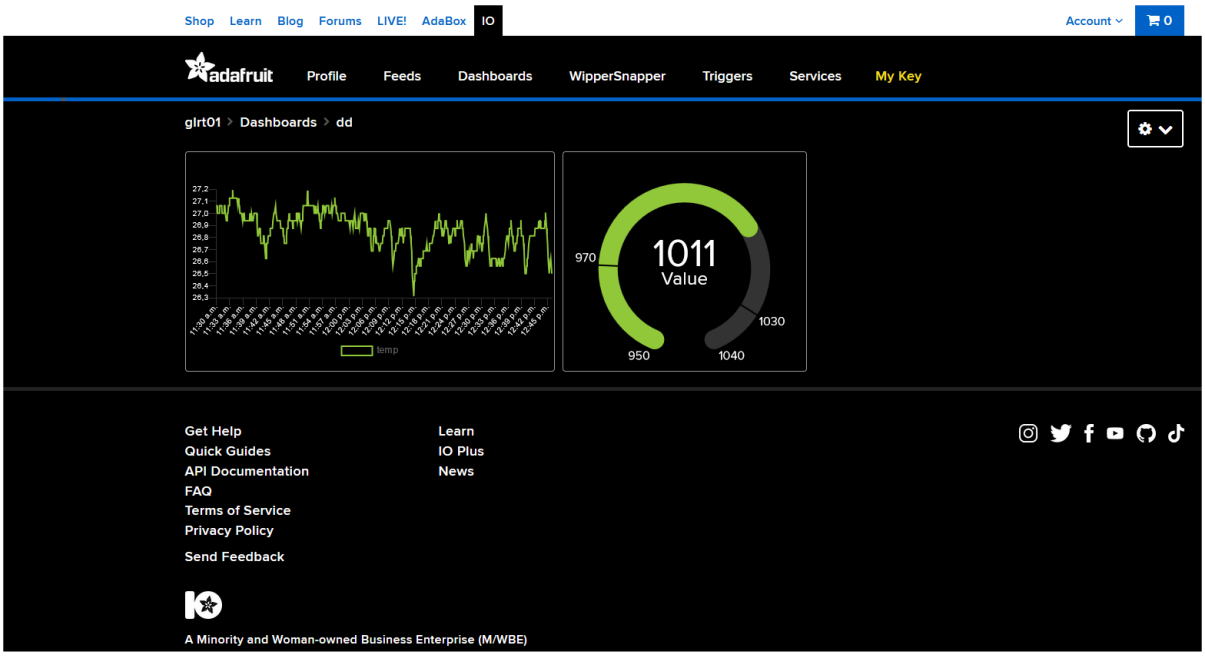
- Ecrire le programme `tp5.py` qui permet d'envoyer au broker MQTT io.adafruit.com la température et la pression de la carte RPi-Pico W. Tester.

On pourra utiliser les informations données en annexe 5 pour la structuration du nom des 2 *topics* MQTT, pour les noms et mots de passe MQTT pour chaque cible RPi-Pico W.
Pour récupérer le mot de passe MQTT, on se connectera sur son compte avec les identifiants donnés en annexe 5 sur le site io.adafruit.com et on pourra le récupérer dans l'onglet « My *Key* » (champ IO_KEY) .

Dans l'onglet « *Feeds* » du site io.adafruit.com, on pourra vérifier la collecte en continu de la température et de la pression.

4.8. EX 6 : dashboard Adafruit

- A partir de l'onglet « *Dashboards* » » du site io.adafruit.com, créer un nouveau dashboard pour représenter graphiquement via des *widgets* la température sous forme d'un graphique et la pression atmosphérique sous forme d'une gauge. Tester.



***Dashboard* io.adafruit.com**

4.9. EX 7 : client MQTT

- Le code Python d'un client MQTT avec la bibliothèque Paho et le suivant :

```
import paho.mqtt.client as mqtt

MQTT_SERVER = "???"     # MQTT server address
MQTT_TOPIC = "???"      # Topic name

def on_connect(client, userdata, flags, rc):
   print("Connection : " + str(rc))
   # Subscribe to the topic
   client.subscribe(MQTT_TOPIC)

# A publish message is received from the server
def on_message(client, userdata, msg):
   print("Sujet : " + msg.topic + " Message : " + str(msg.payload))

client = mqtt.Client()

client.username_pw_set(username="???", password="???")

client.on_connect = on_connect
client.on_message = on_message
client.connect(MQTT_SERVER, 1883, 60)

client.loop_forever()
```

- Modifier le code source pour récupérer par MQTT les informations de pression et de température de sa carte RPi-Pico W.
- Lancer le script Python.
- Observer et analyser les traces MQTT capturées.

5. CONCLUSION

On a pu voir une approche de conception logicielle d'un objet connecté par prototypage rapide basée sur l'usage d'une distribution standard (*Raspberry PI OS*) en utilisant le langage Python mais aussi le langage MicroPython sur microcontrôleur (Raspberry Pi Pico) pour le développement de l'application IoT.

On a pu ensuite étudier et comprendre le protocole MQTT très utilisé dans la conception des objets connectés.

6. REFERENCES

- Carte Raspberry Pi : https://www.raspberrypi.org/
- Carte *Sense HAT* : https://www.raspberrypi.org/learning/addons-guide/sensehat/
- Distribution Linux *Raspberry Pi OS* : https://downloads.raspberrypi.com/raspios_armhf/images/
- *Sense HAT API Reference* : https://pythonhosted.org/sense-hat/api/
- Python et Raspberry Pi : https://www.raspberrypi.org/documentation/usage/python/
- Apprendre Python : http://apprendre-python.com/
- API Python *sockets* : http://apprendre-python.com/page-reseaux-sockets-python-port
- *Threads* Python : https://openclassrooms.com/courses/apprenez-a-programmer-en-python/la-programmation-parallele-avec-threading
- Serveur Web Python : http://apprendre-python.com/page-python-serveur-web-creer-rapidement
- Carte *Raspberry Pi Pico W* : https://www.lextronic.fr/carte-raspberry-pi-pico-w-avec-connecteurs-soudes-64870.html
- Carte *Maker Pi Pico* : https://www.lextronic.fr/maker-pi-pico-base-pour-raspberry-pi-pico-63666.html
- Documentation Maker *Pi Pico. Simplifying Raspberry Pi Pico for Beginner* : https://docs.google.com/document/d/1JoHsZk5IipQPCLXWbZYpDKjGlnkyACOJ1taUrKVsRg8
- Documentation MicroPython : http://docs.micropython.org/en/latest/index.html
- Documentation MicroPython sur la carte *Raspberry Pi Pico* : https://docs.micropython.org/en/latest/rp2/quickref.html

7. ANNEXE 1 : MEMENTO PYTHON DU *SENSE HAT*

8. ANNEXE 2 : API PYTHON DU SENSE HAT

9. ANNEXE 3 : CONFIGURATION RESEAU HOTES ET CIBLES RASPBERRY PI

	POSTE PC01	
	NOM	**ADRESSE IP**
HOTE	rodirula	10.7.4.223
CIBLE	rpi001	10.7.2.118

	POSTE PC02	
	NOM	**ADRESSE IP**
HOTE	sarracenia	10.7.4.225
CIBLE	rpi002	10.7.2.119

	POSTE PC03	
	NOM	**ADRESSE IP**
HOTE	utricularia	10.7.4.227
CIBLE	rpi003	10.7.2.120

	POSTE PC04	
	NOM	**ADRESSE IP**
HOTE	ibicella	10.7.2.112
CIBLE	rpi004	10.7.2.121

	POSTE PC05	
	NOM	**ADRESSE IP**
HOTE	nepenthes	10.7.2.114
CIBLE	rpi005	10.7.2.122

	POSTE PC06	
	NOM	**ADRESSE IP**
HOTE	pinguicula	10.7.2.116
CIBLE	rpi006	10.7.2.123

Masque de sous réseau : **255.255.248.0**

Exemple : configuration réseau de la carte cible `rpi001` :

```
RPi3# ifconfig eth0 10.7.2.118 netmask 255.255.248.0
```

10. ANNEXE 4 : CONFIGURATION RESEAU HOTES ET CIBLES RASPBERRY PI PICO W

	POSTE PC01	
	NOM	**ADRESSE IP**
HOTE	rpi001	10.7.2.118
CIBLE	pico01	DHCP Wifi 192.168.5.x

	POSTE PC02	
	NOM	**ADRESSE IP**
HOTE	rpi002	10.7.2.119
CIBLE	pico02	DHCP Wifi 192.168.5.x

	POSTE PC03	
	NOM	**ADRESSE IP**
HOTE	rpi003	10.7.2.120
CIBLE	pico03	DHCP Wifi 192.168.5.x

	POSTE PC04	
	NOM	**ADRESSE IP**
HOTE	rpi004	10.7.2.121
CIBLE	pico04	DHCP Wifi 192.168.5.x

	POSTE PC05	
	NOM	**ADRESSE IP**
HOTE	rpi005	10.7.2.122
CIBLE	pico05	DHCP Wifi 192.168.5.x

	POSTE PC06	
	NOM	**ADRESSE IP**
HOTE	rpi006	10.7.2.123
CIBLE	pico06	DHCP Wifi 192.168.5.x

11. ANNEXE 5 : CONFIGURATION MQTT DES CIBLES RPI-PICO W

	POSTE PC01
Nom	pico01
Nom compte io.adafruit.com	se01
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se01
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se01/feeds/temp
Topic MQTT pression	se01/feeds/pression

	POSTE PC02
Nom	pico02
Nom compte io.adafruit.com	se02
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se02
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se02/feeds/temp
Topic MQTT pression	se02/feeds/pression

	POSTE PC03
Nom	pico03
Nom compte io.adafruit.com	se03
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se03
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se03/feeds/temp
Topic MQTT pression	se03/feeds/pression

	POSTE PC04
Nom	pico04
Nom compte io.adafruit.com	se04
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se04
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se04/feeds/temp
Topic MQTT pression	se04/feeds/pression

	POSTE PC05
Nom	pico05
Nom compte io.adafruit.com	se05
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se05
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se05/feeds/temp
Topic MQTT pression	se05/feeds/pression

	POSTE PC06
Nom	pico06
Nom compte io.adafruit.com	se06
Passwd compte io.adafruit.com	voir en TP
Username MQTT	se06
Passwd MQTT	voir onglet « My Key » sur io.adafruit.com
Topic MQTT température	se06/feeds/temp
Topic MQTT pression	se06/feeds/pression