Faculté des Sciences & Techniques — Université de Limoges

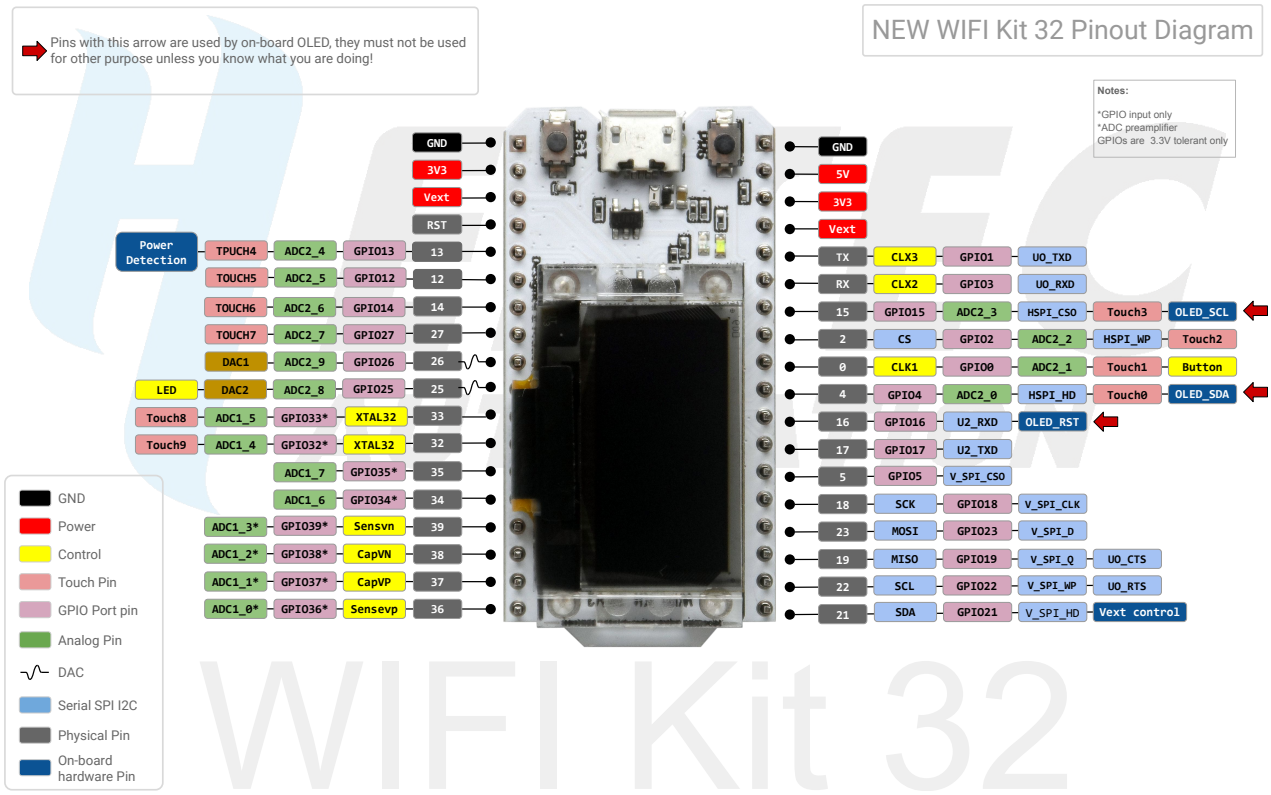
Master 2nde année

TMC

TP n°2

Utilisation de composants avec µPython

Schéma d'interconnexion



https://resource.heltec.cn/download/WiFi_Kit_32/WIFI_Kit_32_pinoutDiagram_V2.pdf

Exploitation du bouton « *PROG* »

```
from machine import Pin
from utime import sleep

button = Pin( , Pin.IN)
led = Pin( , Pin.OUT)

while 1:
    if button.value() == 0:
        print("bouton appuye")
        led.value(1)
    sleep(0.01)
    led.value(0)
```

broche ❶ — *broche* ❷

Identifiez la broche connectée à la led ❷ et au bouton ❶ sur le schéma de d'interconnexion.

Travail :
Écrivez un programme qui envoie un message « *bouton appuyé* » par MQTT sur le topic "ESP32/bouton" lorsque l'on appuie sur le bouton.

Exploitation de l'écran OLED

Vous récupérerez la bibliothèque pilotant l'écran OLED de type SSD1306 :

xterm

```
$ wget https://raw.githubusercontent.com/micropython
/micropython-lib/master/micropython/drivers/display/ssd1306/ssd1306.py
```

Vous testerez le programme suivant en vérifiant le brochage de l'écran : RESET, bus I2C :

```
import ssd1306
from machine import Pin, SoftI2C as I2C
import time
import network

wlan = network.WLAN(network.STA_IF) # create station interface
wlan.active(True)       # activate the interface
wlan.isconnected()      # check if the station is connected to an AP
time.sleep_ms(500)
if not wlan.isconnected():
  print('connecting to network...')
  wlan.connect('IoT', '12344321') # connect to an AP
  time.sleep_ms(500)
  while not wlan.isconnected():
    pass
print('network config:', wlan.ifconfig())

# Heltec LoRa 32 with OLED Display
oled_width = 128
oled_height = 64
# OLED reset pin
i2c_rst = Pin(16, Pin.OUT)
# Initialize the OLED display
i2c_rst.value(0)
time.sleep_ms(5)
i2c_rst.value(1) # must be held high after initialization
# Setup the I2C lines
i2c_scl = Pin(15, Pin.OUT, Pin.PULL_UP)
i2c_sda = Pin(4, Pin.OUT, Pin.PULL_UP)
# Create the bus object
i2c = I2C(scl=i2c_scl, sda=i2c_sda)
# Create the display object
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
oled.fill(0)
oled.text(wlan.ifconfig()[0], 0, 0)
oled.text('INSA CVL', 0, 25)
oled.text('RnF', 0, 55)

#oled.line(0, 0, 50, 25, 1) # dessine un trait
oled.show() # force l'affichage des modifications
```

Travail :
Écrivez un programme qui affiche sur cet écran OLED un message reçu par MQTT sur le topic "ESP32/OLED".

Exploitation du « *transceiver* » LoRa

Vous récupérerez la bibliothèque disponible à `https://github.com/martynwheeler/u-lora`.

xterm
```
$ wget https://raw.githubusercontent.com/martynwheeler/u-lora/main/ulora.py
```

Travail :

a. Vous comparerez les fonctionnalités de cette bibliothèque avec celle-ci :
`https://github.com/fantasticdonkey/uLoRa`

b. Vous regarderez la configuration LoRa que l'on peut sélectionner sur le composant, en termes de « *bandwidth* », « code rate », « *spreading factor* » et puissance d'émission.

```
class ModemConfig():
    #< Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Default medium range
    Bw125Cr45Sf128 = (0x72, 0x74, 0x04)
    #< Bw = 500 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Fast+short range
    Bw500Cr45Sf128 = (0x92, 0x74, 0x04)
    #< Bw = 31.25 kHz, Cr = 4/8, Sf = 512chips/symbol, CRC on. Slow+long range
    Bw31_25Cr48Sf512 = (0x48, 0x94, 0x04)
    #/< Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, low data rate, CRC on.
Slow+long range
    Bw125Cr48Sf4096 = (0x78, 0xc4, 0x0c)
    #< Bw = 125 kHz, Cr = 4/5, Sf = 2048chips/symbol, CRC on. Slow+long range
    Bw125Cr45Sf2048 = (0x72, 0xb4, 0x04)
```

c. Vous modifierez la bibliothèque en l'adaptant à la connexion du composant LoRa de votre ESP32 : brochages et bus SPI

```
class SPIConfig():
    # spi pin defs for various boards (channel, sck, mosi, miso)
    rp2_0 = (0, 6, 7, 4)
    rp2_1 = (1, 10, 11, 8)
    esp8286_1 = (1, 14, 13, 12)
    esp32_1 = (1, 14, 13, 12)
    esp32_2 = (2, 18, 23, 19)
    heltec = (2,5,27,19)
```
ligne à ajouter

Comment peut-on se rendre compte si cela fonctionne ?

d. Vous testerez l'envoi de message par LoRa et la réception.

```
from time import sleep
from ulora import LoRa, ModemConfig, SPIConfig
import time, ssd1306
from machine import Pin, SoftI2C as I2C

oled_width = 128
oled_height = 64
i2c_rst = Pin(16, Pin.OUT)
i2c_rst.value(0)
time.sleep_ms(5)
i2c_rst.value(1) # must be held high after initialization
i2c_scl = Pin(15, Pin.OUT, Pin.PULL_UP)
i2c_sda = Pin(4, Pin.OUT, Pin.PULL_UP)
i2c = I2C(scl=i2c_scl, sda=i2c_sda)
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

# Lora Parameters
RFM95_RST = 14
RFM95_SPIBUS = SPIConfig.heltec
RFM95_CS = 18
RFM95_INT = 26
RF95_FREQ = 868.0
RF95_POW = 20
CLIENT_ADDRESS = 1
SERVER_ADDRESS = 2
# initialise radio
lora = LoRa(RFM95_SPIBUS, RFM95_INT, CLIENT_ADDRESS, RFM95_CS, reset_pin=RFM95_RST,
freq=RF95_FREQ, tx_power=RF95_POW, acks=True)

# loop and send data
while True:
    lora.send_to_wait("This is a test message", SERVER_ADDRESS)
    oled.fill(0)
    oled.text("sent",0,10)
    oled.show()
    sleep(1)
```

```
from time import sleep
from ulora import LoRa, ModemConfig, SPIConfig
import ssd1306
from machine import Pin, SoftI2C as I2C
import time

# This is our callback function that runs when a message is received
def on_recv(payload):
    print("From:", payload.header_from)
    print("Received:", payload.message)
    print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))
    oled.fill(0)
    oled.text("From:"+str(payload.header_from),0,0)
    oled.text("Received:",0,10)
    oled.text(str(payload.message),0,20)
    oled.text("RSSI: {}".format(payload.rssi),0,40)
    oled.text("SNR: {}".format(payload.snr),0,50)
    oled.show()

# Lora Parameters
RFM95_RST = 14
RFM95_SPIBUS = SPIConfig.heltec
RFM95_CS = 18
RFM95_INT = 26
RF95_FREQ = 868.0
RF95_POW = 20
CLIENT_ADDRESS = 1
SERVER_ADDRESS = 2

# initialise radio
lora = LoRa(RFM95_SPIBUS, RFM95_INT, SERVER_ADDRESS, RFM95_CS, reset_pin=RFM95_RST,
freq=RF95_FREQ, tx_power=RF95_POW, acks=True)

# set callback
lora.on_recv = on_recv

# set to listen continuously
lora.set_mode_rx()

# pour l'oled
# Heltec LoRa 32 with OLED Display
oled_width = 128
oled_height = 64
# OLED reset pin
i2c_rst = Pin(16, Pin.OUT)
# Initialize the OLED display
i2c_rst.value(0)
time.sleep_ms(5)
i2c_rst.value(1) # must be held high after initialization
# Setup the I2C lines
i2c_scl = Pin(15, Pin.OUT, Pin.PULL_UP)
i2c_sda = Pin(4, Pin.OUT, Pin.PULL_UP)
# Create the bus object
i2c = I2C(scl=i2c_scl, sda=i2c_sda)
# Create the display object
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
oled.fill(0)
oled.text("Mode serveur",0,20)
oled.show()

# loop and wait for data
while True:
    sleep(0.1)
```

Les différents fichiers sont disponibles à :

`https://git.unilim.fr/pierre-francois.bonnefoi/Heltec_MicroPython_LoRa`

Questions :

a. Comment pouvez vous éviter les « *collisions* » entre tous les composants LoRa transmettant simultanément dans la salle ?

b. Comment évolue le RSSI si vous touchez l'antenne, la réorientez ou la déplacez ?

c. Pouvez vous faire un chat pour échanger entre deux appareils ?