Communication LoRa

Utilisation du composant sx1276/sx1278

Vous récupérerez et installerez la bibliothèque :

`https://raw.githubusercontent.com/readcoil/heltec-lora-micropython/main/sx127x.py`

Vous essaierez le programme suivant :

```
from machine import Pin, SoftSPI
from time import sleep
from sx127x import SX127x
from oled import *

device_config = {
    'miso':19,
    'mosi':27,
    'ss':18,
    'sck':5,
    'dio_0':26,
    'reset':14,
    'led':25,
}

lora_parameters = {
    'frequency': 868E6,
    'tx_power_level': 2,
    'signal_bandwidth': 125E3,
    'spreading_factor': 11,
    'coding_rate': 5,
    'preamble_length': 8,
    'implicit_header': True,
    'sync_word': 0x12,
    'enable_CRC': True,
    'invert_IQ': False,
}
def send(lora,oled):
    counter = 0
    screen = ["LoRa Sender"," ", " ", " "]

    while True:
        payload = 'Hello ({0})'.format(counter)
        screen[0] = "Sending packet:"
        screen[1] = "{}".format(payload)
        screen[2] = "BW:{} SF:{}".format(int(lora_parameters['signal_bandwidth']),
lora_parameters['spreading_factor'])
        screen[3] = "Power:{}".format(lora_parameters['tx_power_level'])
        lora.println(payload)
        write_screen(oled,screen)
        counter += 1
        sleep(2)

device_spi = SoftSPI(baudrate = 10000000,
        polarity = 0, phase = 0, bits = 8, firstbit = SoftSPI.MSB,
        sck = Pin(device_config['sck'], Pin.OUT, Pin.PULL_DOWN),
        mosi = Pin(device_config['mosi'], Pin.OUT, Pin.PULL_UP),
        miso = Pin(device_config['miso'], Pin.IN, Pin.PULL_UP))

lora = SX127x(device_spi, pins=device_config, parameters=lora_parameters)
oled,screen = init_oled()
send(lora,oled)
```

Questions

1. Avec quel type de bus est relié le composant LoRa ?

Communication dans les deux sens : émission et réception

Vous récupérerez le git `https://github.com/readcoil/heltec-lora-micropython`

2. Ajouter la fonction de réception au code fourni en début de fiche :

```
def receive(lora,oled,screen):
    print("LoRa Receiver")
    while True:
        if lora.received_packet():
            payload = lora.read_payload()
            print(payload)
            screen[0] = "pkt: {}".format(payload)
            write_screen(oled, screen)
```

3. En étudiant la bibliothèque `sx127x.py`, comment pouvez vous obtenir le RSSI mesuré lors de la réception d'un paquet ?

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

4. Pouvez vous faire un chat pour échanger avec des identifiants ?

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

Format des messages : intégrité et authentification

Voici un programme réalisant un HMAC :

```
import uhashlib
import ubinascii

SHA1 = uhashlib.sha1


print("==========================================")
print("SHA1 test: ", ubinascii.hexlify(SHA1(b'hello world').digest()))
# should be 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed


# HMAC implementation, as hashlib/hmac wouldn't fit
# From https://en.wikipedia.org/wiki/Hash-based_message_authentication_code
def HMAC(k, m):
    SHA1_BLOCK_SIZE = 64
    KEY_BLOCK = k + (b'\0' * (SHA1_BLOCK_SIZE - len(k)))
    KEY_INNER = bytes((x ^ 0x36) for x in KEY_BLOCK)
    KEY_OUTER = bytes((x ^ 0x5C) for x in KEY_BLOCK)
    inner_message = KEY_INNER + m
    outer_message = KEY_OUTER + SHA1(inner_message).digest()
    return SHA1(outer_message)


KEY = b'abcd'
MESSAGE = b'efgh'
print("==========================================")
print("HMAC test: ", ubinascii.hexlify(HMAC(KEY, MESSAGE).digest()))
# should be e5dbcf9263188f9fce90df572afeb39b66b27198
```

Questions

5. Ajoutez le à votre programme d'échange pour réaliser intégrité et authentification.

Infrastructure de capteurs : broker MQTT & transmission LoRa

Questions

6. Réalisez une passerelle MQTT/LoRa.