Laboratory: Sensor Networking

For this session we will be using the lopy development board. Besides the board you will need the Atom IDE and the pymark package. For this session we will be focusing in point-to-point connectivity between your sensors using various technologies.

1. Wifi

Wifi is a general purpose wireless technology. Typically used with the IP protocol, it is the standard local wireless protocol used in mobile phones and laptops.

1.1 Scan for available networks

Let us start by scanning for available WiFi networks.

```
from network import WLAN
import socket
import machine
import time

wlan = WLAN(mode=WLAN.STA)

nets = wlan.scan()
for net in nets:
    print(net.ssid)
```

What networks did you find? Did you find the “BEST-IoT-Gateway” network?

1.2 Connect to the BEST network

Using the WiFi interface, connect to the access point named “BEST-IoT-Gateway” (password: “BEST-password”).

```
from network import WLAN
import socket
import machine
import time

wlan = WLAN(mode=WLAN.STA)

nets = wlan.scan()
for net in nets:
        if net.ssid == 'BEST-IoT-Gateway':
            print('Connecting to Wifi...')
            wlan.connect(net.ssid, auth=(net.sec, "BEST-password"), timeout=5000)
            print('Connected!')
            break
```

1.3 Sending UDP messages

You can send messages using any IP based protocol. For example using the UDP protocol.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
s.settimeout(1.0)
while True:
        message = input("Enter your message: ")
        s.sendto(message, socket.getaddrinfo('192.168.4.1', 5555)[0][-1])
```

1.4 Receiving UDP messages

Extend the previous example to receive response messages.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
s.settimeout(1.0)
while True:
   message = input("Enter your message: ")
   s.sendto(message, socket.getaddrinfo('192.168.4.1', 5555)[0][-1])
   try:
      msg = s.recv(1500)
          print(msg)
   except Exception as ex:
      print(ex)
```

What response did you receive?

1.5 Set up your own WiFi network

The LoPy can work as a wireless access point, the initialization needs an extra step to setup the network parameters.

```
wlan = WLAN()
    wlan.init(mode=WLAN.AP, ssid='NetworkName', auth=(WLAN.WPA2, 'password'),
    channel=6)
```

Try to connect using your phone, or a second LoPy

2. LoRa

LoRa is a Low Power Wireless communication protocol, used for low power devices such as sensors that rely on small batteries that need to last for years.

2.1 Send a message using LoRa

Here is an example, for sending a message using LoRa:

```
from network import LoRa
import socket

lora = LoRa(mode=LoRa.LORA, frequency=868000000)
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setblocking(True)
s.send('Hello')
```

This message is visible to anyone listening on the correct frequency. Send us a message with your name.

2.2 Find out what the maximum size for message in LoRa

Messages in LoRa have a maximum size limit, find out what it is.

2.3 Receiving messages

There is sensor broadcasting at frequency 863000000, try to listen to its messages. Here is an example:

```
from network import LoRa
import socket
import machine
import time

lora = LoRa(mode=LoRa.LORA, frequency=863000000)

s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setblocking(False)

while True:
    data = s.recv(64)
    if data:
        print(data)
    else:
        print('.')
        time.sleep(5)
```

Move around to test its range. You can try to use different parameters to improve range.

3. Bluetooth

Bluetooth Low Energy is an extension to the regular Bluetooth protocol for low energy applications. Devices exposes service with various attributes that can be readable (sensors) or writable (actuators).

3.1 Find Bluetooth devices

BLE devices send periodic advertisements. Capture advertisements to find out available devices.

```
from network import Bluetooth
import time
import binascii

bt = Bluetooth()
bt.init()

try:
    bt.start_scan(10)
    print("Scanning...", end='')
except:
    pass

advertisements = []

while bt.isscanning():
    print(".", end='')
    adv = bt.get_adv()
    if adv is not None:
        advertisements.append(adv)
    time.sleep(1)
print('done')

for adv in advertisements:
    name = bt.resolve_adv_data(adv.data, Bluetooth.ADV_NAME_CMPL)
    if name:
        print(name)
    else:
            print(binascii.hexlify(adv.mac))
```

3.2 Find available characteristics

Extending the previous example you can find characteristics published by each device

```
for adv in advertisements:
   name = bt.resolve_adv_data(adv.data, Bluetooth.ADV_NAME_CMPL)
   if name:
      print(name)
   else:
      print(binascii.hexlify(adv.mac))

   try:
      conn = bt.connect(adv.mac)
      for service in conn.services():
         print('Reading characteristics from service = {:02x}'.format(service.uuid()))

         for char in service.characteristics():
            if(char.properties() & Bluetooth.PROP_READ):
               try:
                  print('- characteristic {:02x}'.format(char.uuid()), end='')
                  print(' = {}'.format(char.read()))
               except Exception as ex:
                  print(" error reading characteristic", ex)
            conn.disconnect()
         except Exception as ex:
            print('Error reading from device', ex)
         time.sleep(1)
```

There should be a sensor called "BEST-bt", what characteristics does it publish?

3.3 Publish a sensor using bluetooth

Advertise your own sensor using bluetooth. Try to use your code from the previous lab. Here is an example for a time counter:

```
bt = Bluetooth()
bt.set_advertisement(name='MySensor')
bt.advertise(True)

# 0x1805 is a time service
value = 1
time_srv = bt.service(0x1805)
```

```
char = time_srv.characteristic(uuid=0x2a2b, value="{}".format(value))
time_srv.start()

while True:
   time.sleep(1)
   value += 1
   char.value(value)
```

References

- https://docs.pycom.io/
- The bluetooth services specification https://www.bluetooth.com/specifications/gatt/services

Recovery

If you need to recover the system, because the LoPy is left in an unrecoverable state. You can press the reset button in the board.

If you need to perform a full reset of the contents of the board, you can boot in safe mode and clear out all data. Using a wire connect G20 to the 3.3v pin, and then power the board. The micropython prompt will appear, but main.py will not be executed, you can then alter the contents of the flash storage e.g. to clear the flash memory:

```
>>> import os
>>> os.mkfs('/flash')
```

The full recovery docs are available at
https://docs.pycom.io/chapter/toolsandfeatures/bootmodes.html