Relatório laboratório de Tópicos Especiais em Redes de Computadores

Equipe: Gabriel Rocha, Davi Kallebe, Linnik Maciel, Daniel, Kalil, Gabriel Toledano

Objetivo do projeto:

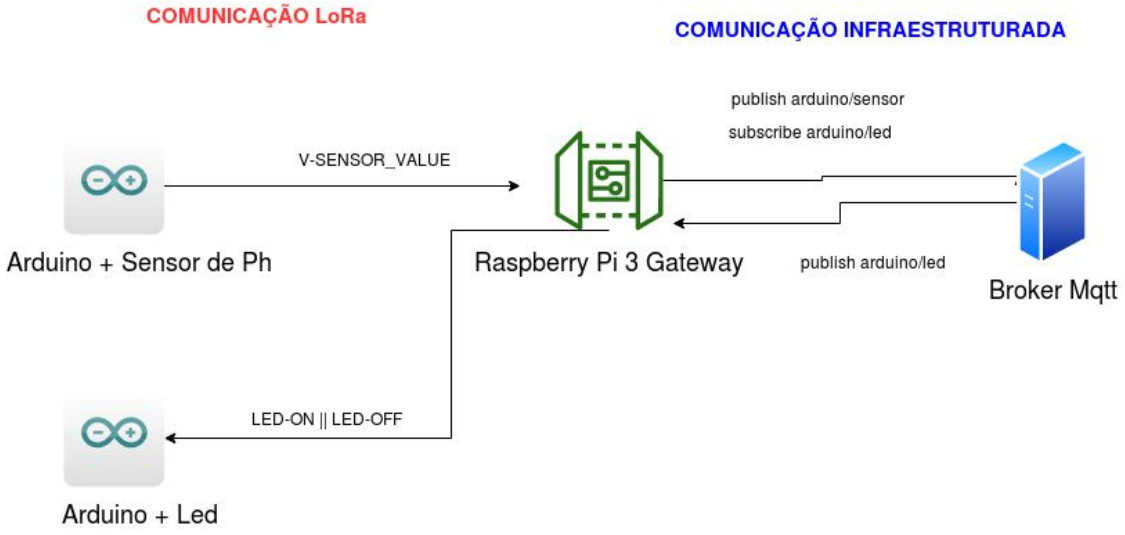
Montar a comunicação central de uma rede voltada para IoT, através da implementação de um nó gateway para fazer a interoperação (e comunicação) entre os nós atuadores que utilizam comunicação LoRa e um servidor Broker Mqtt que utiliza comunicação infraestruturada.

Material

- 2 Arduino equipado com LoRa Shield
- Jumpers
- 1 Led
- Resistores
- 1 Sensor de Ph
- 1 Raspberry pi 3

Execução do projeto:

O arranjo da comunicação dos sistemas envolvidos ficou da seguinte forma como se pode ver na figura abaixo:



No terceiro laboratório, foram utilizadas as bibliotecas: time, sys, os, random para comodidades de programação, a biblioteca pyRadioHeadRF95 para comunicação via LoRa, e a biblioteca paho.mqtt.client para comunicação via mqtt.

Código utilizado:

```
import time, sys, os, random
from datetime import datetime
# Add path to pyRadioHeadRF95 module
sys.path.append(os.path.dirname(__file__) + "/../")

import pyRadioHeadRF95 as radio
import paho.mqtt.client as mqtt

# SetUp Geral

# Configurando RadioHead
rf95 = radio.RF95()
rf95.init()
rf95.setTxPower(20, False)
rf95.setFrequency(915)
rf95.setSpreadingFactor(rf95.SpreadingFactor9)
rf95.setCodingRate4(rf95.CodingRate4_6)
rf95.setSignalBandwidth(500000)

# Metodo para quando o cliente recebe uma resposta CONNACK do servidor MQTT
def on_connect(client, userdata, flags, rc):
        print("Conexao com o Broker estabelida com sucesso com codigo " + str(rc))
        # subscribes to the topic
        client.subscribe("arduino/led")
        print("Gateway escrito no topico: arduino/led")
```

```
# Metodo para lidar quando o servidor publica uma mensagem em um topico que o cliente esta inscrito
def on_message(client, userdata, msg):
        print(msg.topic + " " + str(msg.payload))
        rf95.send(str(msg.payload), len(msg.payload))
        rf95.waitPacketSent()
        print("pacote enviado, dormindo..")
        sys.stdout.flush()

        # na realidade, aqui o comando eh repassado para o arduino
        # chamar funcoes da radiohead para enviar o comando para o arduino

# Configurando o cliente MQTT
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("10.208.3.226") # 1883, 60

# Iniciando a thread que ira escutar e lidar com o canal do MQTT
#client.loop_start()
print('Conexao com o broker estabelecida. Thread do Broker iniciada...Aguardando mensagens do topico')
# client.loop_forever()

bw = 0
print("Rasp Iniciado! em:")
print(datetime.now())
print("-"*50)
sys.stdout.flush()
```

```
msg = 'LED-0'
msg2 = 'LED-1'
#while True:
#       continue

'''while True:
        rf95.send(msg, len(msg))
        rf95.waitPacketSent()
        print("LED-0 enviado, dormindo..")
        time.sleep(5)
        rf95.send(msg2, len(msg2))
        rf95.waitPacketSent()
        print("LED-1 enviado, dormindo..")
        time.sleep(5)'''

#'''
while bw < 10:
        rf95.setSignalBandwidth(500000)
        while True:
                # escutando
                #if rf95.available() or random.randint(0,4) == 1:
                if rf95.available():
                        print("trying listening LoRa")
                        (msg, l) = rf95.recv()
                        if(msg):
                                valor = msg.split('-')
                                valor[1]
                                client.publish("arduino/sensor", valor[1])
```

```
                    # client.publish("arduino/sensor", msg)

                    print('msg:', msg, l)
                    print(valor[1])
                    # print(msg)
                    print('finish\n')
                    sys.stdout.flush()
                #time.sleep(1)
            else:
                print("Waiting mqtt response")
                client.loop_start()
                time.sleep(1)
#'''
```

Considerações sobre a execução do projeto:

Consistência do botão: Ao apertar o botão do broker, nem sempre o LED do Arduino final acende, já que não foi implementado um sistema de entrega seguro. Os motivos para perda de pacotes são o meio, pois a mensagem pode se perder antes de chegar ao host, e a forma como foi desenvolvido o servidor.

Para evitar uma execução simultânea do protocolo mqtt e LoRa, os dois são executados em alternância de forma que é possível que a mensagem chegue num momento em que o servidor não esteja escutando o canal do mqtt, mas sim o do LoRa, e assim sendo perdida.

Criptografia: Apesar da biblioteca RadioHead no Arduino ter funções criptográficas preparadas, a biblioteca utilizada no raspberry, PyHadioReadRF95, não implementou estas funções. Procurando sobre, foi descoberto a biblioteca PyLoRa que possuía estas funções. Foi testado a sua utilização durante um certo tempo, mas apesar de ser teoricamente possível utilizá-la, não se conseguiu executar o código utilizando a mesma. Desta forma, por escopo de tempo, optamos por remover a criptografia da mensagem, pois era o único detalhe no trabalho que não conseguiu-se realizar.

Na execução: “waiting mqtt”: está escutando o canal mqtt, e após um tempo, ele tenta escutar o canal do LoRa.

Arduino/led: LED-X – 1 se for para ligar, 0 se fora para desligar

Msg: mensagem

‘V-XX’ – é o dado captado

‘9’ – perguntar do Gabriel