Connected Bus Monitor

For Sustainable Mobility

Dipartimento di Ingegneria informatica
automatica e gestionale Antonio Ruberti

Sapienza
Università di Roma

Francesco Giuseppe Crinò
Constanta Efros

PROTOTYPE

AND

TECHNICAL ASPECT

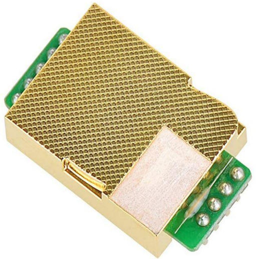
Prototype - Components

STM32
NUCLEO-F446ZE

+

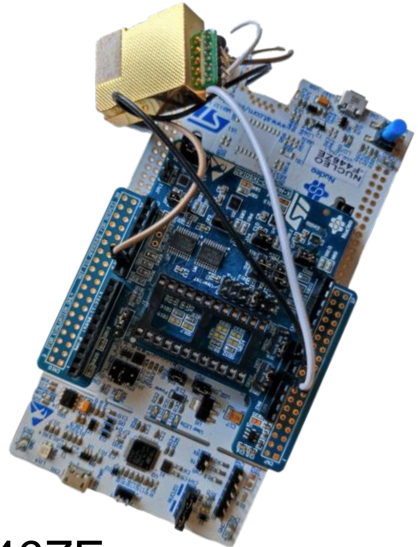
X-NUCLEO-IKS01A2

+

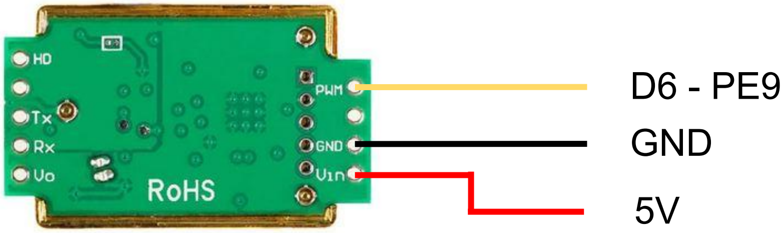


MH-Z19C

=



- The extension board is placed on top of the NUCLEO-F446ZE covering all the Arduino subset of Zio
- MH-Z19C connection to NUCLEO-F446ZE:



D6 - PE9

GND

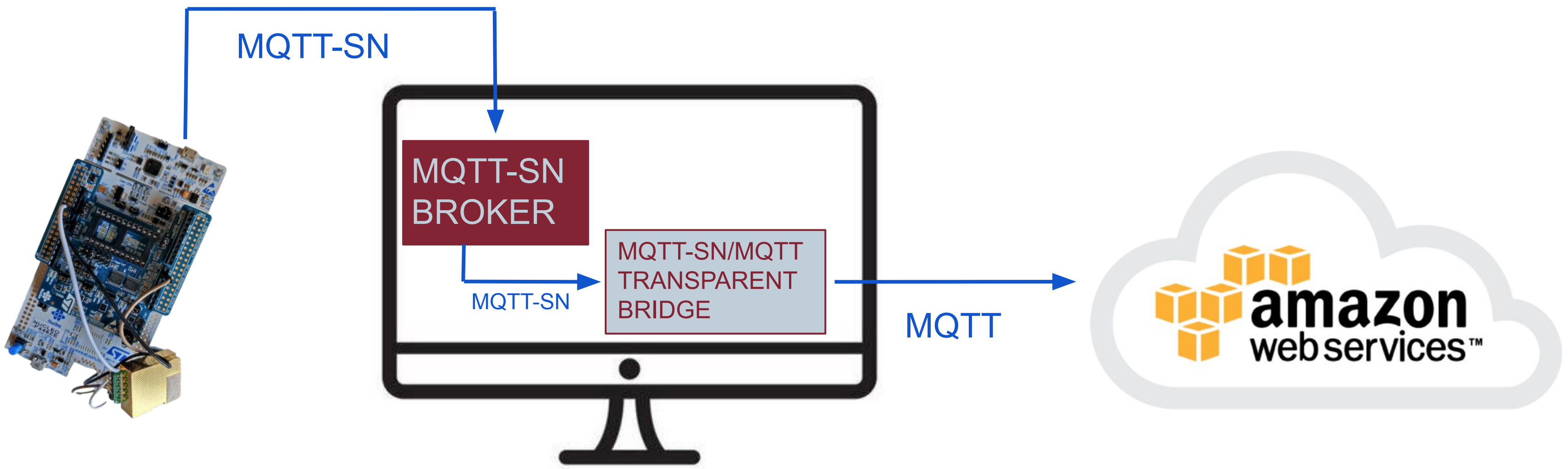
5V

Prototype - Components

For more specific information about the components:

https://github.com/FrancescoCrino/ConnectedBusMonitor/blob/main/Technology.md

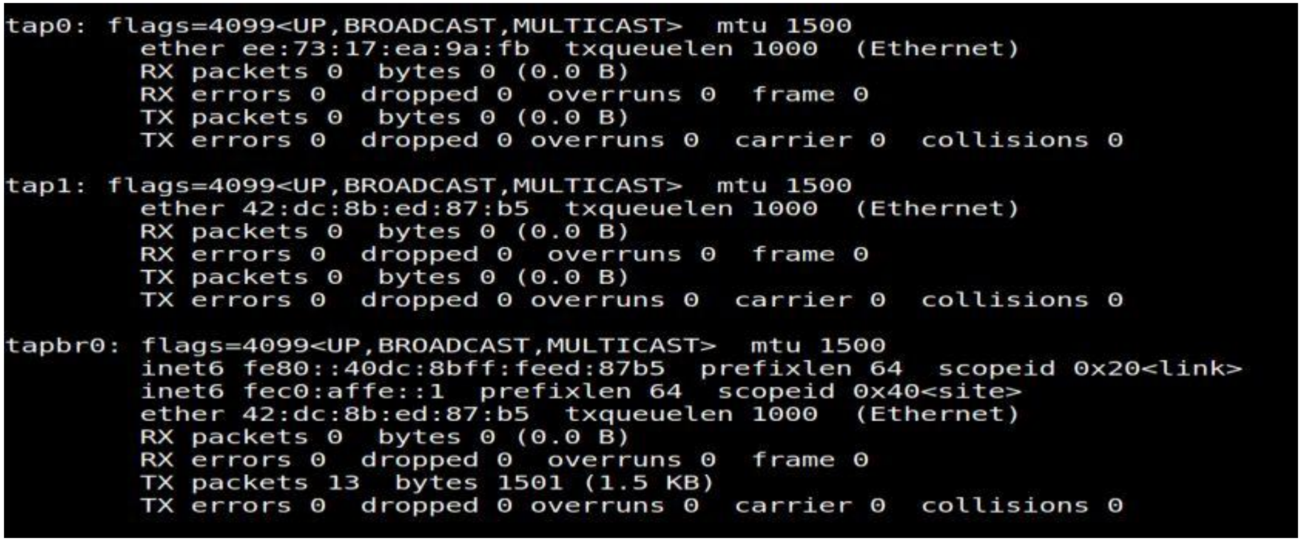
Prototype - Connectivity



- The device is connected to the pc using the IPv6 stack over a serial USB device through the ETHOS
  - It sends the collected data to the MQTT-SN broker deployed on the pc
- The MQTT-SN broker forward the data to the MQTT-SN/MQTT transparent bridge that forward the data to AWS through MQTT
- AWS receives the data and store them into an S3 bucket

Prototype - How it works

1. Creation of virtual networks



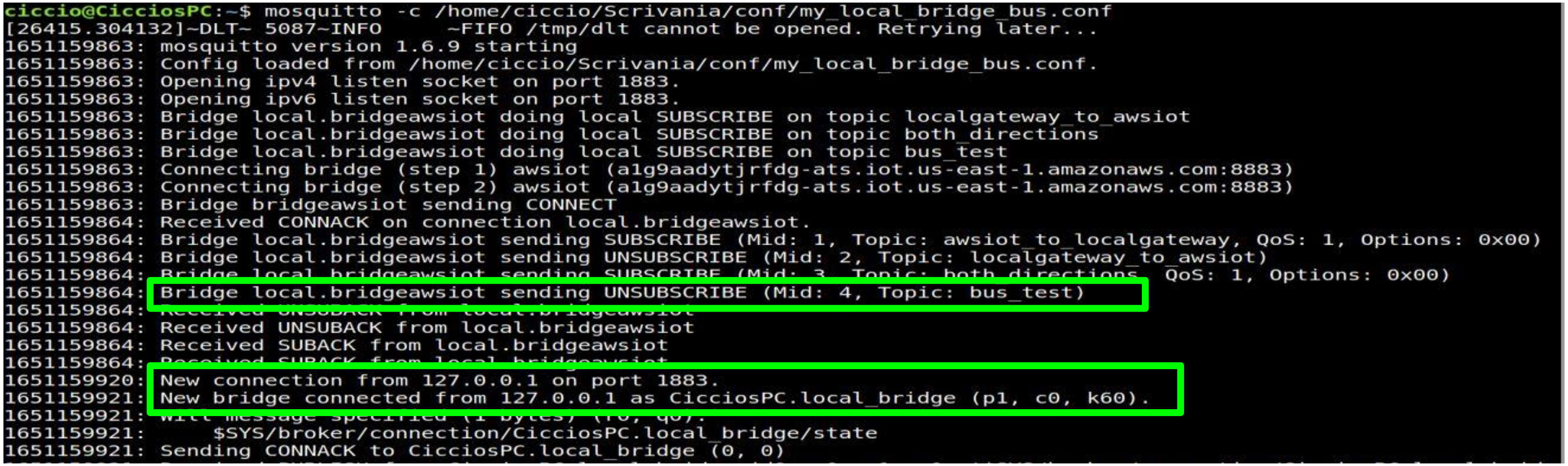
```
tap0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether ee:73:17:ea:9a:fb  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tap1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 42:dc:8b:ed:87:b5  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tapbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet6 fe80::40dc:8bff:feed:87b5  prefixlen 64  scopeid 0x20<link>
        inet6 fec0:affe::1  prefixlen 64  scopeid 0x40<site>
        ether 42:dc:8b:ed:87:b5  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1501 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Prototype - How it works

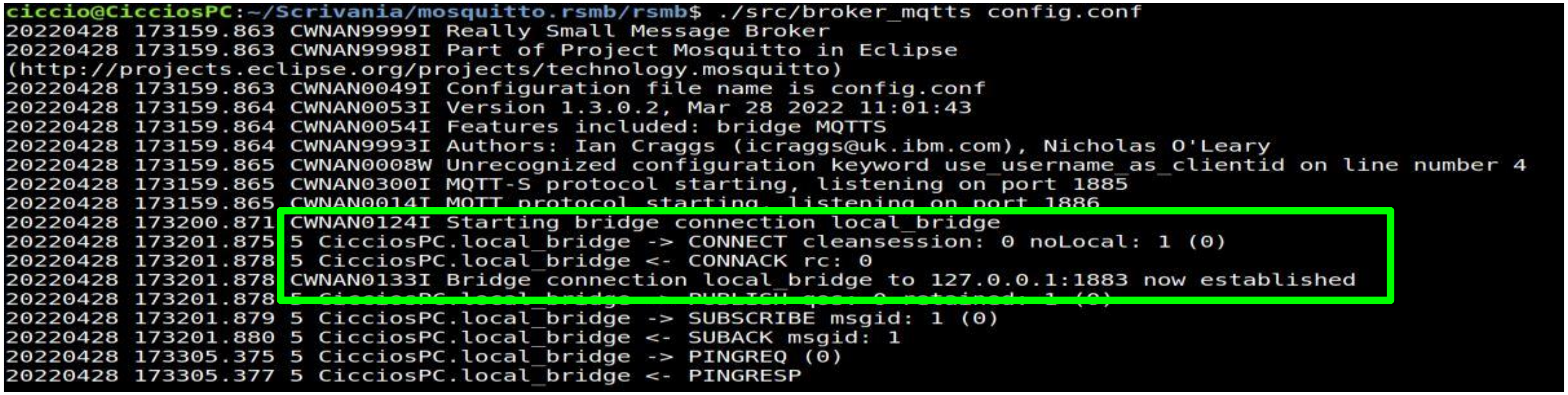
2. Deployment of mosquitto MQTT-SN/MQTT transparent bridge



```
ciccio@CicciosPC:~$ mosquitto -c /home/ciccio/Scrivania/conf/my_local_bridge_bus.conf
[26415.304132]~DLT~ 5087~INFO     ~FIFO /tmp/dlt cannot be opened. Retrying later...
1651159863: mosquitto version 1.6.9 starting
1651159863: Config loaded from /home/ciccio/Scrivania/conf/my_local_bridge_bus.conf.
1651159863: Opening ipv4 listen socket on port 1883.
1651159863: Opening ipv6 listen socket on port 1883.
1651159863: Bridge local.bridgeawsiot doing local SUBSCRIBE on topic localgateway_to_awsiot
1651159863: Bridge local.bridgeawsiot doing local SUBSCRIBE on topic both_directions
1651159863: Bridge local.bridgeawsiot doing local SUBSCRIBE on topic bus_test
1651159863: Connecting bridge (step 1) awsiot (a1g9aadytjrfdg-ats.iot.us-east-1.amazonaws.com:8883)
1651159863: Connecting bridge (step 2) awsiot (a1g9aadytjrfdg-ats.iot.us-east-1.amazonaws.com:8883)
1651159863: Bridge bridgeawsiot sending CONNECT
1651159864: Received CONNACK on connection local.bridgeawsiot.
1651159864: Bridge local.bridgeawsiot sending SUBSCRIBE (Mid: 1, Topic: awsiot_to_localgateway, QoS: 1, Options: 0x00)
1651159864: Bridge local.bridgeawsiot sending UNSUBSCRIBE (Mid: 2, Topic: localgateway_to_awsiot)
1651159864: Bridge local.bridgeawsiot sending SUBSCRIBE (Mid: 3, Topic: both_directions, QoS: 1, Options: 0x00)
1651159864: Bridge local.bridgeawsiot sending UNSUBSCRIBE (Mid: 4, Topic: bus_test)
1651159864: Received UNSUBACK from local.bridgeawsiot
1651159864: Received UNSUBACK from local.bridgeawsiot
1651159864: Received SUBACK from local.bridgeawsiot
1651159864: Received SUBACK from local.bridgeawsiot
1651159920: New connection from 127.0.0.1 on port 1883.
1651159921: New bridge connected from 127.0.0.1 as CicciosPC.local_bridge (p1, c0, k60).
1651159921: Will message specified (1 bytes) (r0, q0).
1651159921:     $SYS/broker/connection/CicciosPC.local_bridge/state
1651159921: Sending CONNACK to CicciosPC.local_bridge (0, 0)
```

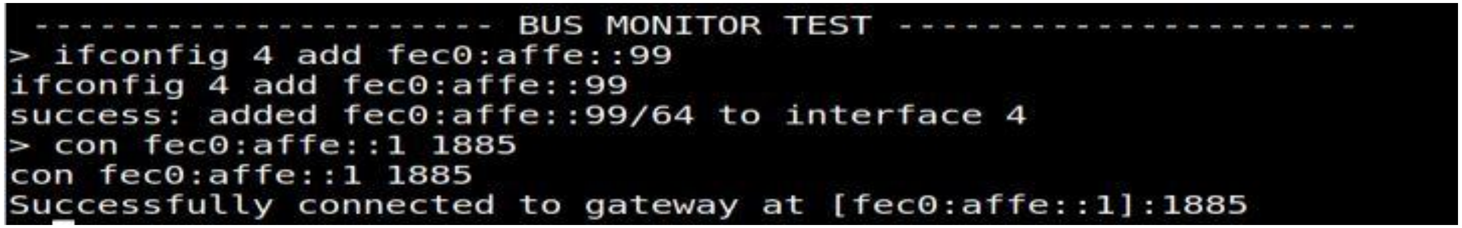
Prototype - How it works

3. Deployment of mosquitto RSMB broker and connection with transparent bridge



```
ciccio@CicciosPC:~/Scrivania/mosquitto.rsmb/rsmb$ ./src/broker_mqtts config.conf
20220428 173159.863 CWNAN9999I Really Small Message Broker
20220428 173159.863 CWNAN9998I Part of Project Mosquitto in Eclipse
(http://projects.eclipse.org/projects/technology.mosquitto)
20220428 173159.863 CWNAN0049I Configuration file name is config.conf
20220428 173159.864 CWNAN0053I Version 1.3.0.2, Mar 28 2022 11:01:43
20220428 173159.864 CWNAN0054I Features included: bridge MQTTS
20220428 173159.864 CWNAN9993I Authors: Ian Craggs (icraggs@uk.ibm.com), Nicholas O'Leary
20220428 173159.865 CWNAN0008W Unrecognized configuration keyword use_username_as_clientid on line number 4
20220428 173159.865 CWNAN0300I MQTT-S protocol starting, listening on port 1885
20220428 173159.865 CWNAN0014I MQTT protocol starting, listening on port 1886
20220428 173200.871 CWNAN0124I Starting bridge connection local_bridge
20220428 173201.875 5 CicciosPC.local_bridge -> CONNECT cleansession: 0 noLocal: 1 (0)
20220428 173201.878 5 CicciosPC.local_bridge <- CONNACK rc: 0
20220428 173201.878 CWNAN0133I Bridge connection local_bridge to 127.0.0.1:1883 now established
20220428 173201.878 5 CicciosPC.local_bridge -> PUBLISH qos: 0 retained: 1 (0)
20220428 173201.879 5 CicciosPC.local_bridge -> SUBSCRIBE msgid: 1 (0)
20220428 173201.880 5 CicciosPC.local_bridge <- SUBACK msgid: 1
20220428 173305.375 5 CicciosPC.local_bridge -> PINGREQ (0)
20220428 173305.377 5 CicciosPC.local_bridge <- PINGRESP
```

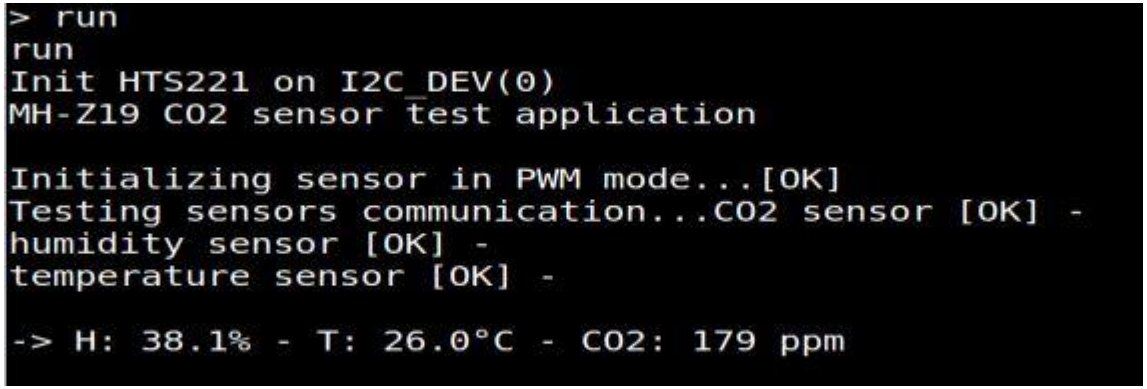
4. Init the prototype



```
 --------------------- BUS MONITOR TEST ---------------------
> ifconfig 4 add fec0:affe::99
ifconfig 4 add fec0:affe::99
success: added fec0:affe::99/64 to interface 4
> con fec0:affe::1 1885
con fec0:affe::1 1885
Successfully connected to gateway at [fec0:affe::1]:1885
```

Prototype - How it works

5. Start running the prototype

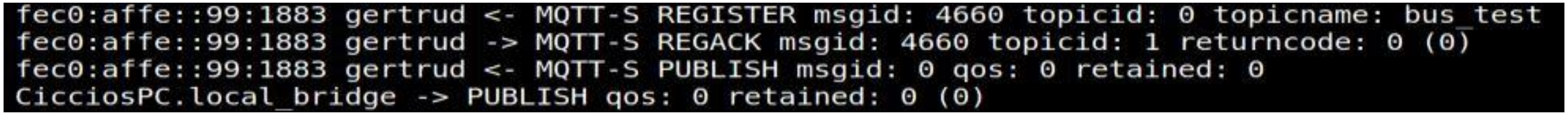


```
> run
run
Init HTS221 on I2C_DEV(0)
MH-Z19 CO2 sensor test application

Initializing sensor in PWM mode...[OK]
Testing sensors communication...CO2 sensor [OK] -
humidity sensor [OK] -
temperature sensor [OK] -

-> H: 38.1% - T: 26.0°C - CO2: 179 ppm
```

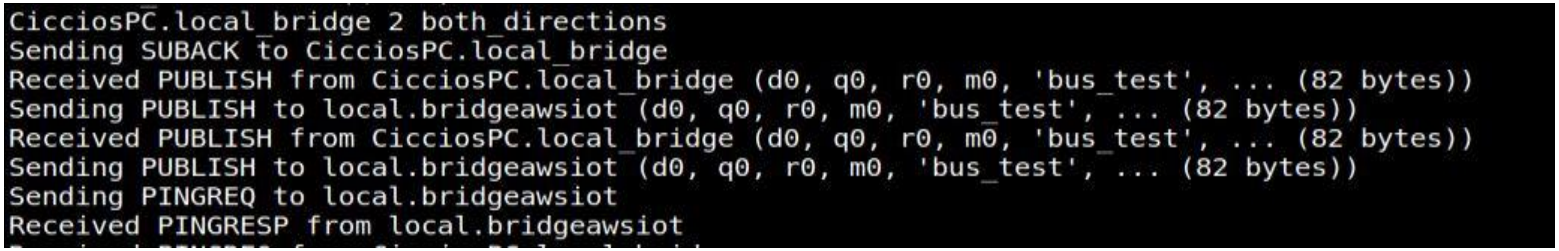
6. RSMB receives pub request and forward the message received to the transparent bridge



```
fec0:affe::99:1883 gertrud <- MQTT-S REGISTER msgid: 4660 topicid: 0 topicname: bus_test
fec0:affe::99:1883 gertrud -> MQTT-S REGACK msgid: 4660 topicid: 1 returncode: 0 (0)
fec0:affe::99:1883 gertrud <- MQTT-S PUBLISH msgid: 0 qos: 0 retained: 0
CicciosPC.local_bridge -> PUBLISH qos: 0 retained: 0 (0)
```

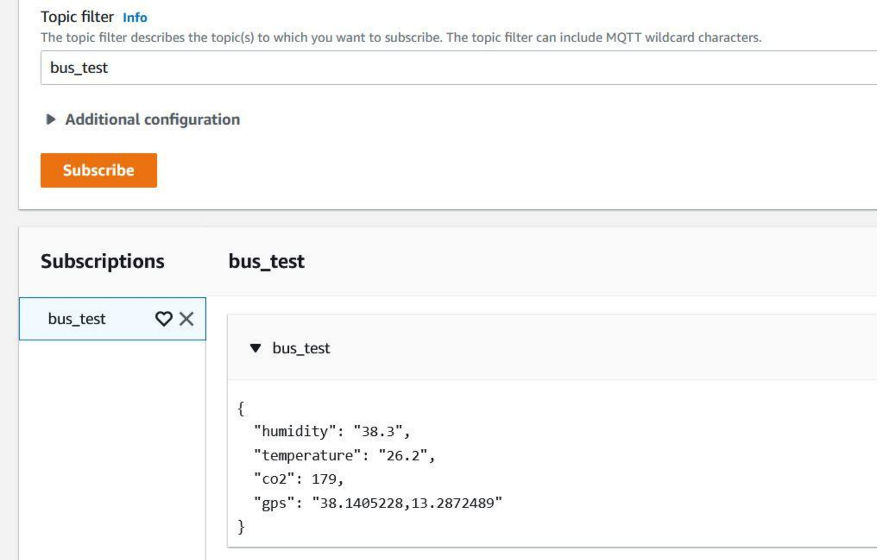
Prototype - How it works

8. Mosquitto transparent bridge receives the pub request from RSMB and publish to AWS with topic bus_test



```
CicciosPC.local_bridge 2 both_directions
Sending SUBACK to CicciosPC.local_bridge
Received PUBLISH from CicciosPC.local_bridge (d0, q0, r0, m0, 'bus_test', ... (82 bytes))
Sending PUBLISH to local.bridgeawsiot (d0, q0, r0, m0, 'bus_test', ... (82 bytes))
Received PUBLISH from CicciosPC.local_bridge (d0, q0, r0, m0, 'bus_test', ... (82 bytes))
Sending PUBLISH to local.bridgeawsiot (d0, q0, r0, m0, 'bus_test', ... (82 bytes))
Sending PINGREQ to local.bridgeawsiot
Received PINGRESP from local.bridgeawsiot
```

9. We can see the message arriving to AWS



Topic filter Info

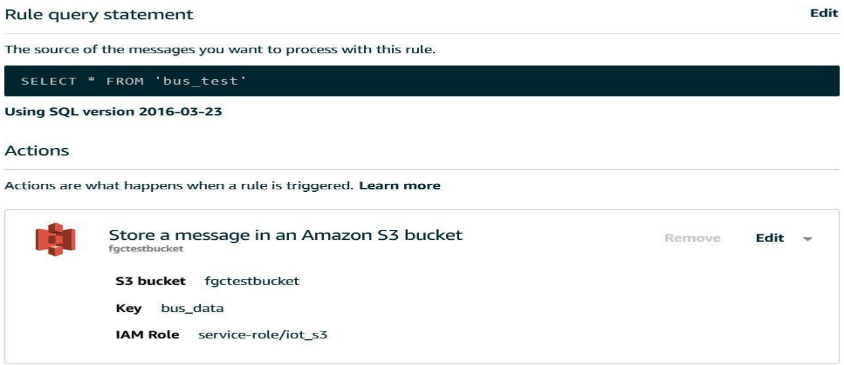
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

bus_test

```
{
  "humidity": "38.3",
  "temperature": "26.2",
  "co2": 179,
  "gps": "38.1405228,13.2872489"
}
```

Prototype - How it works

10. AWS store the incoming message in a S3 bucket following that rule



Rule query statement

The source of the messages you want to process with this rule.

```
SELECT * FROM 'bus_test'
```

Using SQL version 2016-03-23

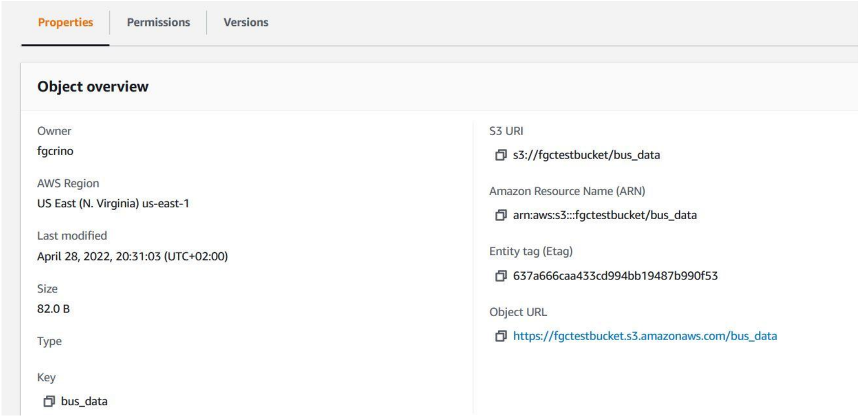
Actions

Actions are what happens when a rule is triggered. **Learn more**

Store a message in an Amazon S3 bucket
fgctestbucket

S3 bucket fgctestbucket
Key bus_data
IAM Role service-role/iot_s3

11. We can find our data stored into the S3 bucket



Object overview

Owner
fgcrino

AWS Region
US East (N. Virginia) us-east-1

Last modified
April 28, 2022, 20:31:03 (UTC+02:00)

Size
82.0 B

Type

Key
bus_data

S3 URI
s3://fgctestbucket/bus_data

Amazon Resource Name (ARN)
arn:aws:s3:::fgctestbucket/bus_data

Entity tag (Etag)
637a666caa433cd994bb19487b990f53

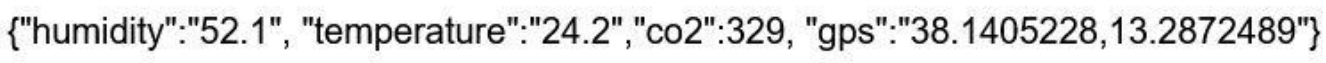
Object URL
https://fgctestbucket.s3.amazonaws.com/bus_data

Prototype - LoRa emulation

To test the LoRa communication we have started from the lorawan example provided by RIOT-OS:

https://github.com/RIOT-OS/RIOT/tree/master/examples/lorawan

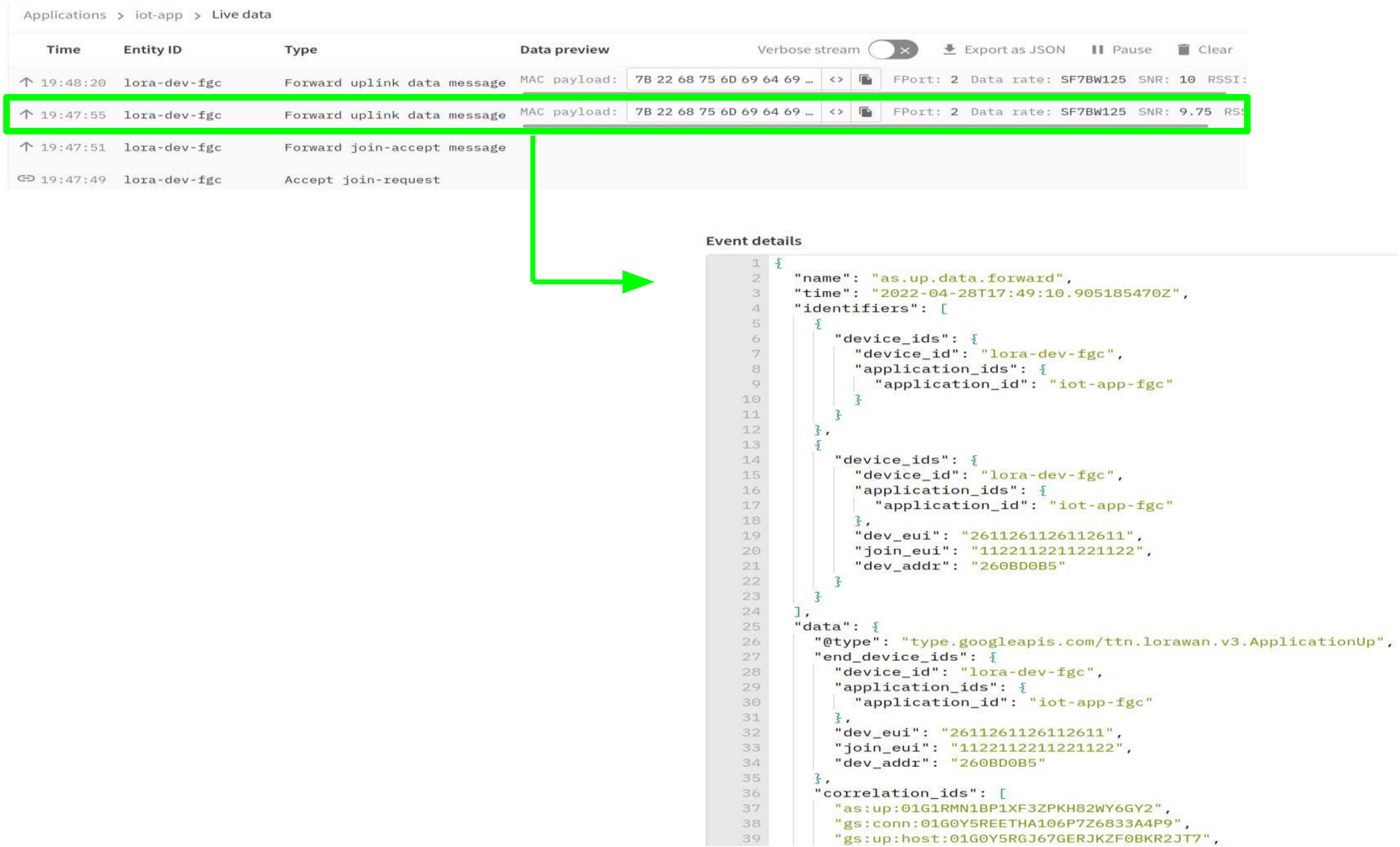
Starting from the example we generate a payload of the form



```
{"humidity":"52.1", "temperature":"24.2","co2":329, "gps":"38.1405228,13.2872489"}
```

And we send that message to a LoRa device emulated on TheThingsNetwork

Prototype - LoRa emulation



Time	Entity ID	Type	Data preview
↑ 19:48:20	lora-dev-fgc	Forward uplink data message	MAC payload: 7B 22 68 75 6D 69 64 69 … FPort: 2 Data rate: SF7BW125 SNR: 10 RSSI:
↑ 19:47:55	lora-dev-fgc	Forward uplink data message	MAC payload: 7B 22 68 75 6D 69 64 69 … FPort: 2 Data rate: SF7BW125 SNR: 9.75 RS
↑ 19:47:51	lora-dev-fgc	Forward join-accept message	
19:47:49	lora-dev-fgc	Accept join-request	

Event details

```
{
  "name": "as.up.data.forward",
  "time": "2022-04-28T17:49:10.905185470Z",
  "identifiers": [
    {
      "device_ids": {
        "device_id": "lora-dev-fgc",
        "application_ids": {
          "application_id": "iot-app-fgc"
        }
      }
    },
    {
      "device_ids": {
        "device_id": "lora-dev-fgc",
        "application_ids": {
          "application_id": "iot-app-fgc"
        },
        "dev_eui": "2611261126112611",
        "join_eui": "1122112211221122",
        "dev_addr": "260BD0B5"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.ApplicationUp",
    "end_device_ids": {
      "device_id": "lora-dev-fgc",
      "application_ids": {
        "application_id": "iot-app-fgc"
      },
      "dev_eui": "2611261126112611",
      "join_eui": "1122112211221122",
      "dev_addr": "260BD0B5"
    },
    "correlation_ids": [
      "as:up:01G1RMN1BP1XF3ZPKH82WY6GY2",
      "gs:conn:01G0Y5REETHA106P7Z6833A4P9",
      "gs:up:host:01G0Y5RGJ67GERJKZF0BKR2JT7",
```

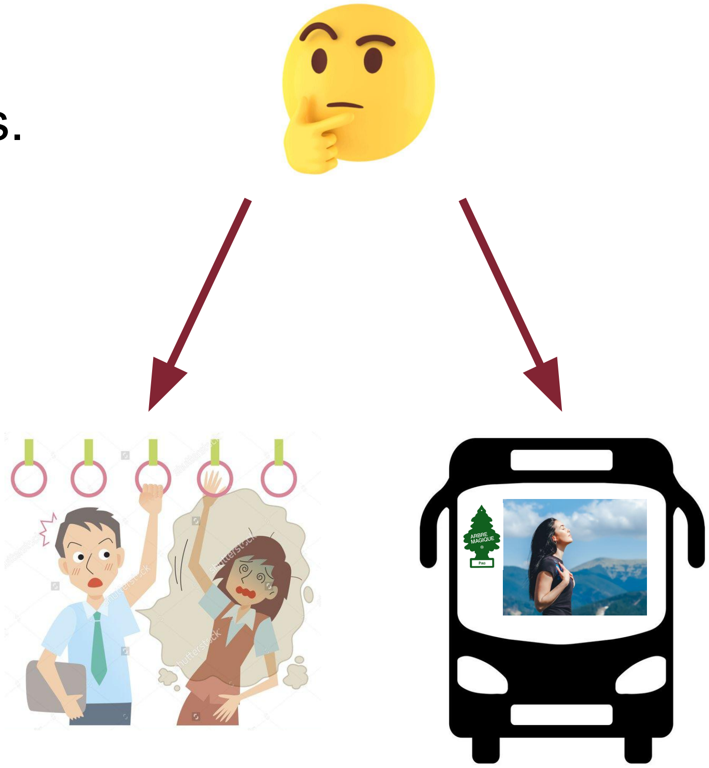
EVALUATION

Bus cab air monitoring

Provide to the user an indication about the air quality inside the bus.

We will evaluate the air inside the bus considering the following indicators:

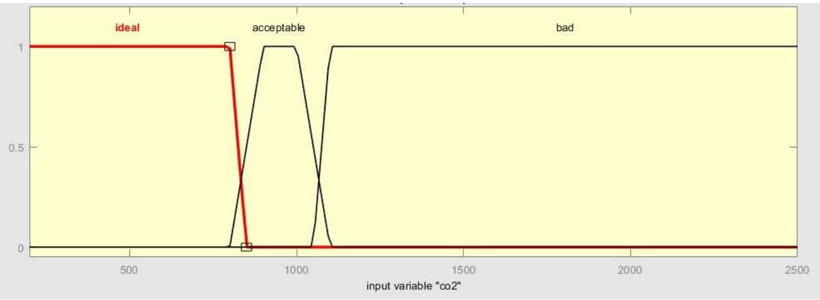
1. CO2 concentration
2. Air temperature
3. Humidity percentage



Bus cab air monitoring

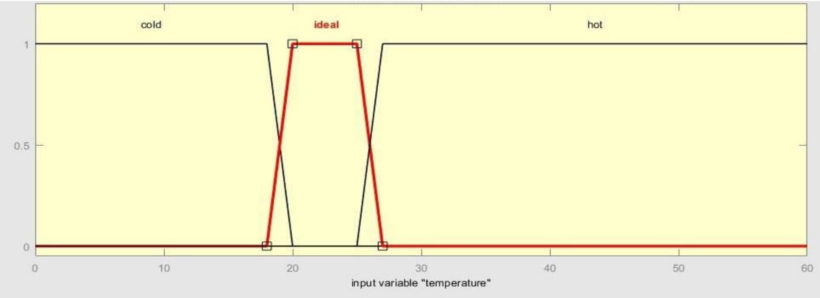
CO2 concentration thresholds [ppm]:

- Ideal: < 800
- acceptable: [800, 1110]
- bad: > 1100



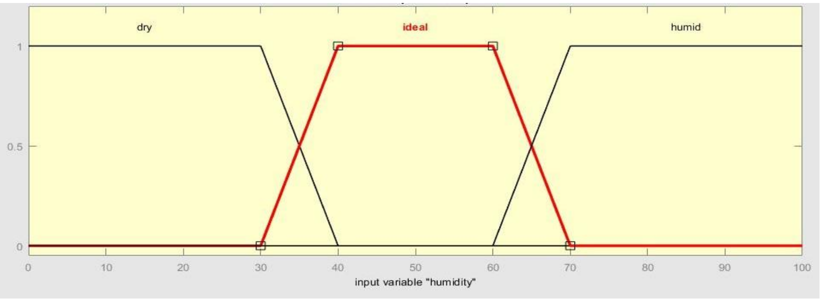
Temperature thresholds [°C]:

- cold: < 20
- ideal: [20, 25]
- hot: > 25



Humidity thresholds [%]:

- Dry: < 40
- Ideal: [40, 60]
- Too humid: > 60



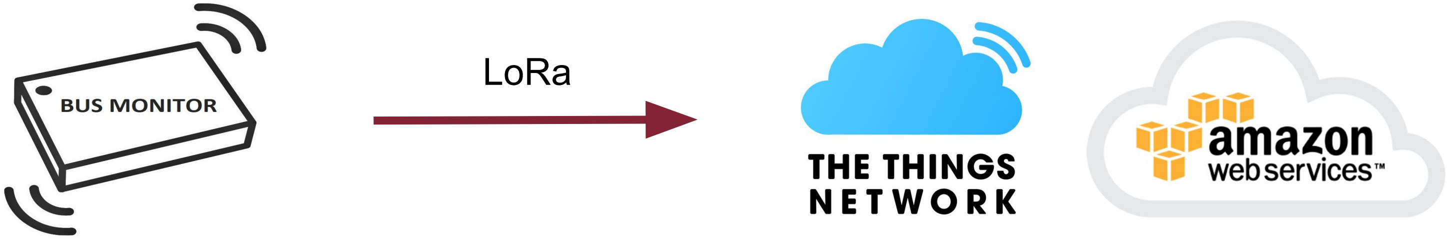
Bus cab air monitoring

We can define 4 levels of air quality inside the bus:

- bad 🥵: all the indicators are not in the ideal range
- ok 😑: only one indicator is in the ideal range
- good 😊: two indicators are in the ideal range
- ideal 😄: all the indicators are in the ideal range

Transmission period

We want to store the data collected by bus monitor devices in a cloud service to make them accessible to the user

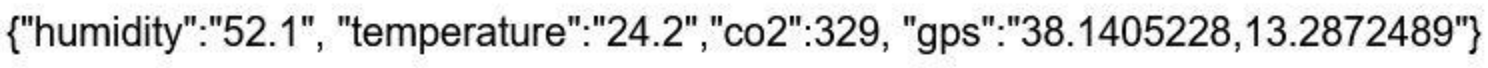


To send data from bus monitors to cloud we will use LoRaWAN and so we need to respect LoRa constraints.

Transmission period

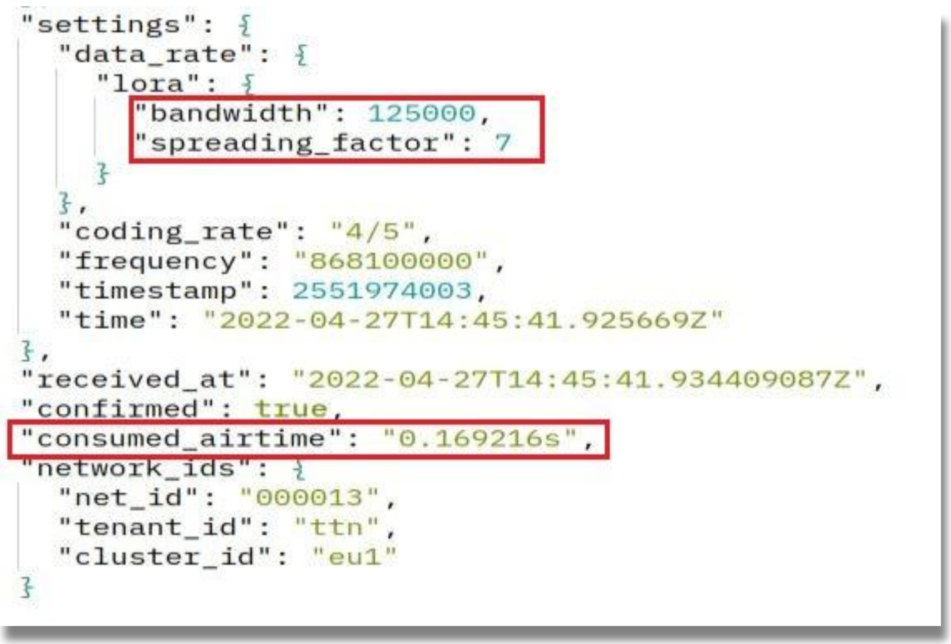
We have sent some LoRa messages to TTN through iot-lab with.

Plain payload of dimension 82 bytes:



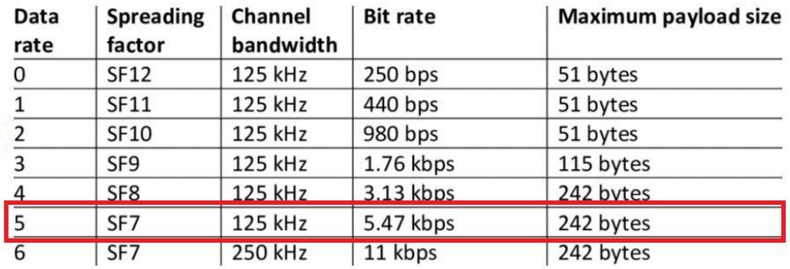
{"humidity":"52.1", "temperature":"24.2","co2":329, "gps":"38.1405228,13.2872489"}

Transmission setting of the LoRa packet received by TTN:



```
"settings": {
  "data_rate": {
    "lora": {
      "bandwidth": 125000,
      "spreading_factor": 7
    }
  },
  "coding_rate": "4/5",
  "frequency": "868100000",
  "timestamp": 2551974003,
  "time": "2022-04-27T14:45:41.925669Z"
},
"received_at": "2022-04-27T14:45:41.934409087Z",
"confirmed": true,
"consumed_airtime": "0.169216s",
"network_ids": {
  "net_id": "000013",
  "tenant_id": "ttn",
  "cluster_id": "eu1"
}
```

Transmission period



Data rate	Spreading factor	Channel bandwidth	Bit rate	Maximum payload size
0	SF12	125 kHz	250 bps	51 bytes
1	SF11	125 kHz	440 bps	51 bytes
2	SF10	125 kHz	980 bps	51 bytes
3	SF9	125 kHz	1.76 kbps	115 bytes
4	SF8	125 kHz	3.13 kbps	242 bytes
5	SF7	125 kHz	5.47 kbps	242 bytes
6	SF7	250 kHz	11 kbps	242 bytes

SF=7
BW=125kHz

→

Bit Rate of
5.47 kbps

Transmission time t_T needed to send a LoRa packet with the previous setting

$$t_T = \frac{PayloadSize_{bit}}{BitRate} = \frac{PayloadSize_{byte} * 8}{BitRate} = \frac{(82 * 8)bit}{5470\ bit/s} \approx 0.120s$$

Since we need to respect a duty cycle of 1% imposed by LoRa:

We can transmit each 12 seconds

Power consumption - stop mode

Normally a city bus can stop at the terminus also for few hours before to restart its service.
During this time we want to keep the system and the sensor on.

- AA rechargeable battery
- STM32 board in stop mode

STM32 nucleo-f401re stop mode electric consumption: 65 µA
Classic AA battery capacity: 2600mAh

An AA battery can power the STM32 nucleo-f401re board on stop mode for:

$$\frac{2600\ mAh}{0.065\ mA} = 40000\ h$$

Using a rechargeable AA battery that recharge itself while receiving power from the bus we will not have power issues for a very long time

FUTURE PLANS

Geolocation feature

X-NUCLEO-GNSS1A1

STM32CUBE (X-NUCLEO-GNSS)

Data acquisition

- Latitude and longitude
- Timestamp/elapsed since last acquired value

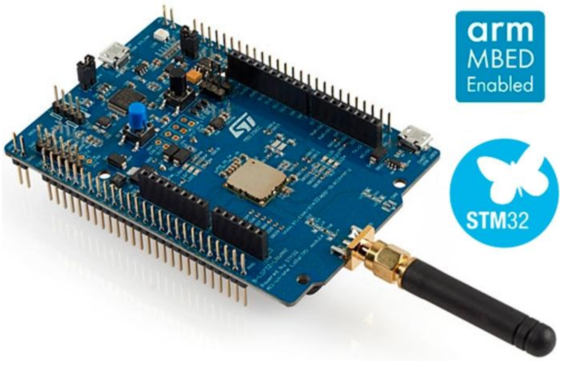
Metrics

- Power consumption
- Payload size (to improve the transmission period)

Connectivity

LoRa Gateway

I-NUCLEO-LRWAN1 or B-L072Z-LRWAN1



Metrics

- Latency
- Throughput

Power source

Our bus monitor devices are static devices installed inside the bus cabin.

A city bus company have to manage thousands of buses

the device needs to be autonomous as possible from the energy point of view

Ideas:

- Connect the device to the grid of the bus (USB port)
- Use a rechargeable battery with a energy source like
    - Small solar panel
    - Dynamo

Power source

Use a rechargeable battery with a solar panel

- expensive
- problems during night or cloudy days
- difficulty in inserting the system in a bus

Use a rechargeable battery with a dynamo

- not expensive
- lots of energy generated from the wheels movement
- difficulty in inserting the system in a bus

Connect the device to the grid of the bus (USB port)

- all next generation bus project include USB port
- simple installation
- reliable

Web dashboard

Provide the indications about the air quality and geolocation of the bus.

- **Client side**
- Angular (OpenStreetMaps, chart libraries)
- **Server side**
- Amazon API Gateway (REST API)
- Amazon DynamoDB (key/value data storage)

Thanks for your attention

https://github.com/FrancescoCrino/ConnectedBusMonitor