IOT - Project Report

Giovanni De Lucia - Person Code: 10700658
Lorenzo Battiston - Person Code: 10618906

A.Y. 2022/2023

POLITECNICO
MILANO 1863

Contents

1 Introduction

This project aims to design and implement a lightweight publish-subscribe protocol in TinyOS, drawing inspiration from MQTT. Simulations on a star-shaped network using Cooja will validate its features, while integration with Node-RED and ThingSpeak will facilitate real-time data visualization.

1.1 Interfaces

- **Receive**: this interface is used to receive packet by the other nodes
- **AMSend**: this interface is used to send packet to the other nodes
- **Packet**: this interface is used to extract payload from packets
- **Timer0**: this timer is used to actual sending messages, very similar to the Challenge 3
- **Timer1**: the PANC utilizes a timer to periodically send messages from a message list to their intended destinations
- **Timer2**: this timer handles the connection request to the PAN Coordinator, ensuring retransmission if acknowledgment is not received. Upon successful acknowledgment, it triggers the next timer for subscription
- **Timer3**: this timer sends a subscription request and handles retransmission until a successful acknowledgment is received

- **Random**: this interface is used to generate random topic and payload

1.2 Header File and Definitions

This section provides an overview of the key declarations and parameters used in the project, including message structures, connections and subscriptions states, topic names, payload formats.

```
typedef nx_struct pub_sub_msg {
    nx_uint8_t type;
    nx_uint16_t sender;
    nx_uint8_t topic;
    nx_uint16_t payload;
} pub_sub_msg_t;
```

Message Format definition

```
#define MAX_NODES 20
#define MESSAGE_DELAY 300
#define NODE_DELAY 8000
#define PAN_COORDINATOR_ID 1
#define NUM_TOPIC 3
#define MAX_PAYLOAD 100

enum {
    TEMPERATURE, HUMIDITY, LUMINOSITY
};

enum {
    CONN, CONNACK, SUB, SUBACK, PUB
};
```

Various Parameters used

```
typedef nx_struct client_info {
    nx_uint8_t topic;
    nx_uint8_t is_subscribed;
    nx_uint8_t is_connected;
} client_info_t;

typedef client_info_t client_list_t[MAX_NODES];
```

Client Information Array

- **Message Format**: This is the payload structure that the messages sent by nodes will have. The type filed will define the semantics of the other fields
- **Parameters**: Parameters defined to enable easy expansion and customization of the project
- **Client Information**: The PANC maintains a client list, recording node states, connections, subscriptions, and topic details. Although limited to one topic as per the specification, it can be easily extended by transforming the integer field into an array

1.3 Node Variables

- **locked**: this flags prevents race conditions by handling situations where the send is occupied by another packet
- **connectAcked**: used to handle connection retransmission
- **subscribeAcked**: used to handle subscription retransmission
- **message_list**: dynamic queue used by the PANC to store messages for periodic sequential sending, avoiding concurrency
- **client_list**: stores the updated state of nodes in the network, used by the PANC.

1.4 High Level Code Logic

At a high level, the code follows the following logic:

PANC Initialization:
- PANC boots up and starts the AMControl
- PANC initiate the list of clients
- PANC starts the periodic Timer1, described above

Other Nodes Initialization:
- Nodes boot up and start the AMControl
- Nodes initiate the variables of acks to FALSE
- Nodes starts the periodic Timer2, described above

PANC receives a Conection/Subscription Request:
- Send a connack/suback to the original sender
- If the connack/suback doesn't arrive in time, a retransmission attempt is initiated
- If a connack arrives in time, the node will make a subscription attempt on a **random** topic

PANC receives a Publish Request:
- Send a publish request to each of the client subscribed to the specific topic

1.4.1 Message Received Handling

The Receive.receive event contains a switch on the type of message arrived and will handle the response accordingly as described in the logic above

1.5 Simulation with Cooja

We initially chose TOSSIM for simulation, but it lacked the ability to expose a TCP port, then we switched to using Cooja. In Cooja, we constructed a star-shaped network comprising the PANC at the center and 8 client nodes. All nodes utilized SKY mote hardware with firmware referencing the compiled TelosB binary. The log of the PANC is available in the root of the project in the file log_cooja.txt.

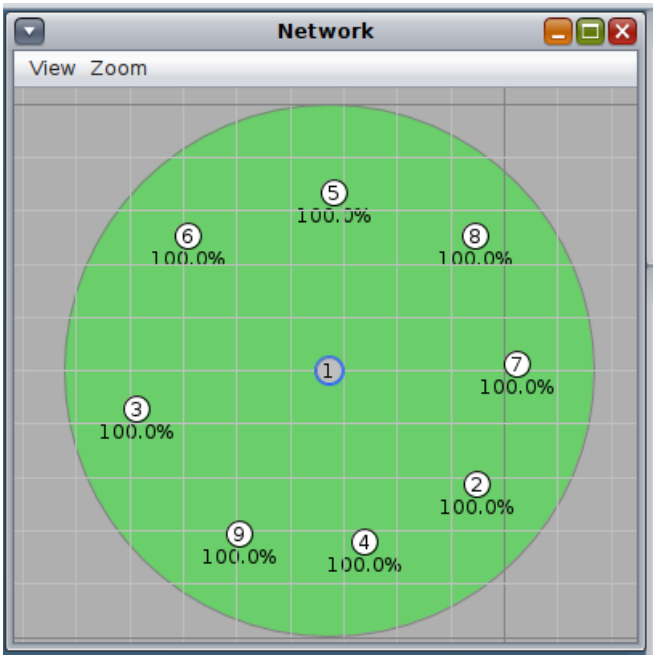

Figure 1: The star-shaped network

1.6 Node-RED

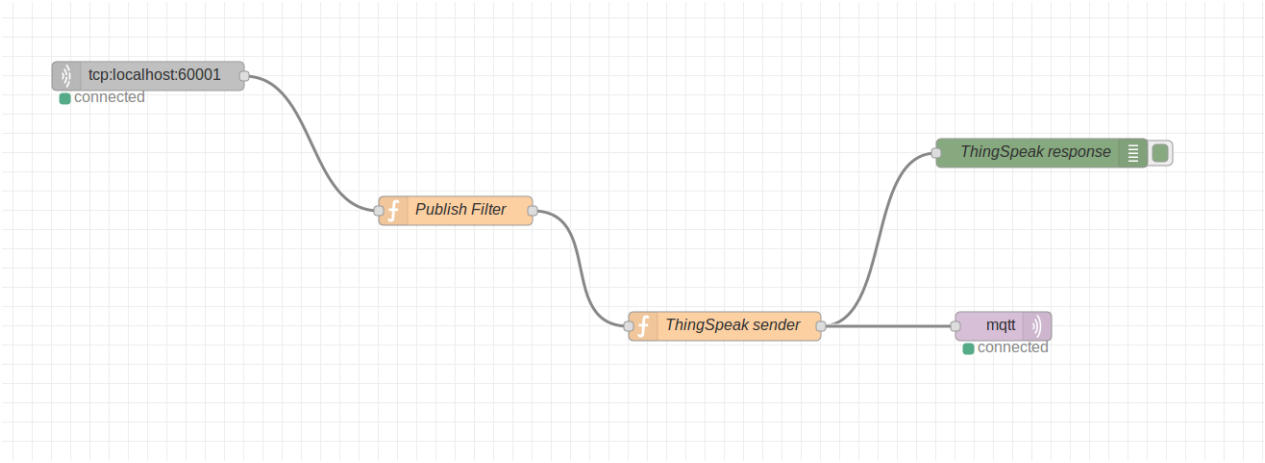

Figure 2: Node-RED flow graph

- TCP input: this node establishes a connection with the TCP server exposed by the PANC in Cooja and receives the transmitted strings
- Publish Filter: this node uses a regular expression (regex) to extract the payload and topic from a publish message
- ThingSpeak sender: this node sets the topic to ensure the publish message payload is sent to the correct topic on the MQTT broker of ThingSpeak
- mqtt: this node is used to send the message to the channel via MQTT

1.7 ThingSpeak

The public channel is available here: `https://thingspeak.com/channels/2183018`

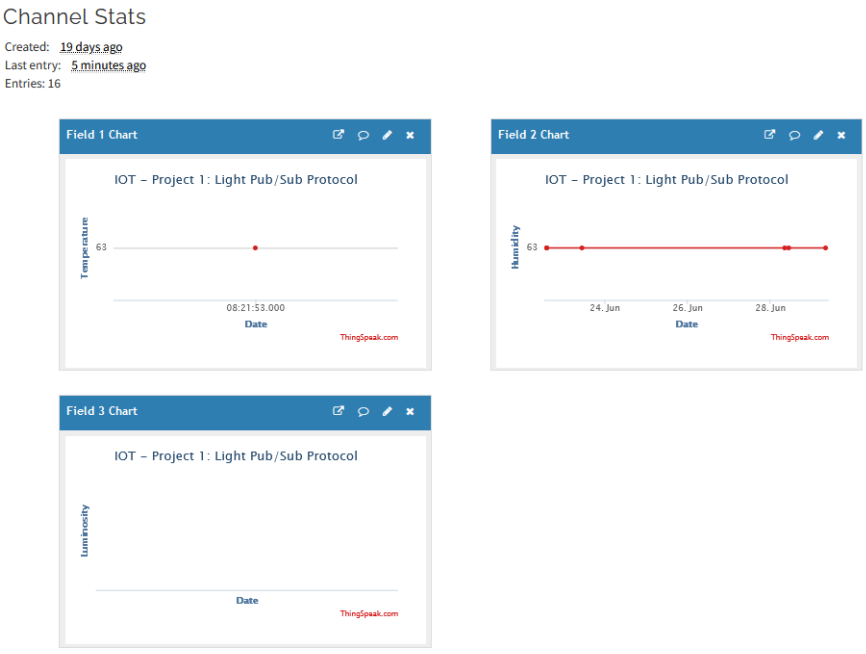

Figure 3: Note: due to a limitation in the VM's Pseudo Random Number Generator, the 'Random' function consistently produces the same number. An alternative approach could be using the last digits of the current time for randomness.