POLITECNICO
MILANO 1863

IoT Project Report

LORAWAN SENSOR NETWORK

Mohammadreza Bakhshizadeh Mohajer 913233 - 10647452 | August 2020

Introduction

In this project we are using TinyOS as operating system to create a LoraWAN like sensor network. Cooja is used as simulator, Node-red is used for connecting the project to the internet, and finally all the data is transmitted through MQTT for storage and further processing to Thingspeak cloud.

The procedure is as follows.

1. Sensors start acquiring and sending data to the Root node through the Gateways by setting the destination address of the packets to the address of the Root node.
2. Gateways on reception read the header of the packets and find the destination address and forward the packets to the Root.
3. Received packets in the Root are first acknowledged and then they are sent to Node-red through a TCP socket.
4. Finally, Node-red send the data to be saved in Thingspeak cloud through MQTT.

Implementation

The code is divided into three parts, each for a role in the network. The first code is for the sensors under the name, Nodes. In the code, after booting the devices we try to turn on the radio and in case of success we start a periodic timer that whenever it fires, packet is created and filled for sending. In our project the timer of each individual node is chosen randomly in the range of 15 to 20 seconds using the Random interface. In the creating process of a packet we set three identifiers for each of them. First one is the source address which is the Node ID of the node who sends the data. Then we set the address of the Root as the destination address. Finally, we set a packet ID to be able to recognize duplicates and implement our acknowledgement system. In the end the payload is filled by the data which in our case are randomly generated using the Random interface. When a packet is sent a one-second periodic acknowledgement timer is started and the data sender timer is turned off. In case of the acknowledgement timer expires, and the packet was not acknowledged, we send the exact packet again, until we receive the acknowledgement.

Successfully sent packets will arrive to the Gateway nodes which have a different source code. Gateways on reception start reading the packet header and payload. Information such as source and destination addresses, packet ID, and data in the payload are extracted. Then messages are recreated exactly as they were and are forwarded according to the address in the received packet.

Finally, packets are received by the Root. In the Root we first extract all the information in the packet and check if it is a duplicate or not by using the source address and the packet ID of the received packet. If both the source address and the packet ID are the same, we ignore the message, otherwise we extract and print the data and send back an acknowledgement. The acknowledge packet in the header has the Root Node ID as its source address, the Node ID of the mote who has sent the packet as the destination address, and the packet ID of the received message. The payload of the ack packet, is set to zero however in case we need to send data back to the source node, we can use the payload of the ack packet.

The acknowledgement packet travels the same way through the Gateway with the same process mentioned above, it is forwarded to the source node that it was sent from by extracting the addresses. In the source mote upon reception of a packet, the destination address is extracted and checked to see if the message is for this node. If the answer is positive, the rest of the information of the packet header is extracted and the packet ID is checked to see if the packet is a duplicate or not. If this test result in positive, the packet ID is checked to find out if this is the acknowledgment of the packet that was formerly sent. By passing the last test, the sender timer is reactivated, and the retransmission timer is stopped.

Going back to the Root node, the received data are displayed in Cooja and using a TCP socket they are sent to Node-red. The first step in Node-red is to clean and extract the data using a splitter and modifying the messages. Then the template for the MQTT message is created and completed by placing the variables in it. In the end we use an execute node to run the command in terminal to use MQTT for sending the message to Thingspeak cloud.

Please note that in order to see the full debug in Cooja, the commented printf functions must be uncommented. However due to the bogus functionality of printf in Cooja which adds unwanted characters (such as wired characters that appear between each data line) to the messages, please run the whole program as it is to see the full performance without errors, then modify and uncomment the printfs.

The log file of Cooja plus implementation model of Cooja (.csc file), and Node-red model code are all included in the project file. The link to the Thingspeak channel is the following : https://thingspeak.com/channels/1118622

Attention, please use the provided csc file for Cooja simulation to avoid errors for the scenario of this project. For other scenarios the Root Node ID must be set first.