Names: Tijam Moradi & Oskari Vahala

Instructor: Ethiopia Nigussie

Course: Secure Sensor Network Systems

Project: Arduino-based test beds

Updated: 14-10-2019

1. SSNS Project Work: Introduction

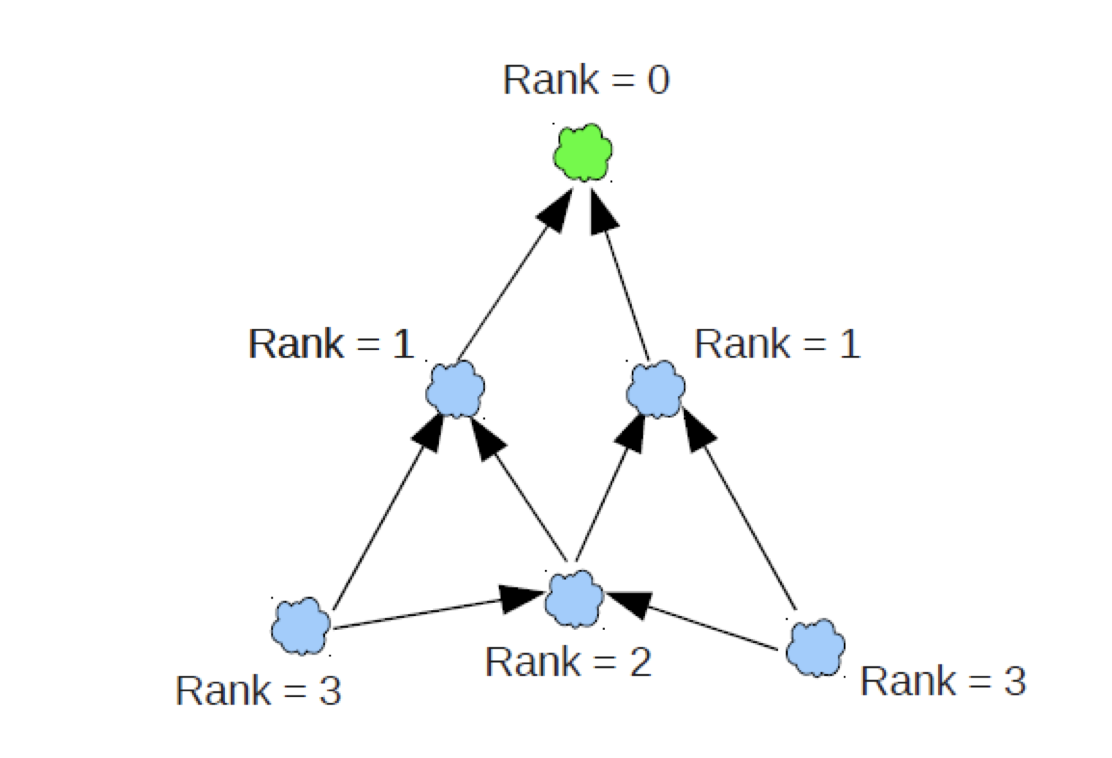
This project has the intention for us to learn about networking technologies right on low level. The project uses DIGI XBee S2D shields for communication. We have four sets of: Arduino, XBee shield, XBee transmitter, Sensor (either proximity or humidity+temperature).

Since the original documentation for the project requirements were ambiguous, we decided to make an implementation as close as possible.

One of the four devices will be solely a **base station**. This base station will receive the data from other devices and forward it to USB serial for remote monitoring. We consider USB Serial is remote enough for this project, since we don’t have Bluetooth or WiFi shields to send data to cloud. The purpose of the project also is not to build something in the cloud but wireless secure networking on ground.

The other four devices will have one of two sensors: temperature & humidity or proximity. Each device will read and process the data from the respective sensors. The processed data will then be sent to the base station for remote monitoring. The base station itself does not make use of a sensor since we have decided that its sole purpose is to gather the data and forward it in good telemetry. If we were to mix the base station with sensor logic there could be problems in future with scalability.

The telemetry from the sensor devices is transferred to the base station with RPL IPv6 via mesh network. In this kind of scenario, we can have the devices more far away from the base station. See the illustration below.



The difference with the illustration and our system is that we have only 4 devices from which one is Rank 0 device, the receiver. Other devices than Rank 0 device are defined to be transmitters. Transmitters ranks are dynamic with RPL stack so they can hop on each other based on signal strength.

1. Network

On this project we used ZigBee network with XBEE S2D units. Network consists of a coordinator and three routers that work also as a sensor node. ZigBee is a low powered personal area network that is designed for low powered IOT devices.

Sensor nodes are working as routers because we would place them on forklifts. It is possible for node to move around the storage unit. There is a possibility for a node to bee too far away from the coordinator that the communication would not be possible without using another node as a relay node. That is the reason why our implementation does not have ‘real’ end nodes. Power management is also harder to implement, because routers are not allowed to sleep or if they sleep, it would have to be synchronized.



Project environment

We used general programming conducts and good guidelines. Our development is based mostly on Git versioning tool and is hosted on GitHub as a collaborative environment. We used both VSCode with Arduino extension and Arduino IDE to program the devices. For RF modules we used XCTU to flash and program their topology. We used several libraries, both internal and external and designed an appropriate folder structure.



Formats & Code logic

Our project has two kinds of IoT devices (based on sensors we got). We have the DHT (humidity and temperature) and Proximity devices. The IoT data is defined to be called as *Telemetry*. Telemetry is constructed from key (string), a value (float) and unit (string). We did this to have a unified format across all devices and protocols.



DHT module works by using an external library on converting and querying info from sensors. It would have been a waste of time and really unnecessary to redevelop the basic tools. We used the Adafruits’ *DHT.h* -library. It has a simple function void DHT::begin that initiates the class, sets up pin modes and debug prints stuff for figuring out if something is wrong. We then gather the data separately as a DHT\_Message that is basically a struct that has humidity and temperature. The information is queried from the sensor through the library with methods dht.readHumidity() and dht.readTemperature().

Upon sending telemetry we obviously separate the temperature and humidity data into their own telemetry units. This is called controlled windowing. Since XBee library internals accept **only** an array of unsigned chars, we mask the data and compress it into the bare essentials of 5 chars.



Implementation

We used

|  | Column Heading | Column Heading | Column Heading |
| --- | --- | --- | --- |
| Row Heading |  |  |  |
| Row Heading |  |  |  |
| Row Heading |  |  |  |

Source: [This source text uses a style named Table Source.]

1. [This note text uses a style named Table Note. Table notes use a lowercase letter instead of Arabic numerals to differentiate them from the notes to body content.]