Device Setup and Function

Introduction

The hardware used in SPRN networks consists of an ESP32 microcontroller, which uses an HC-12 transceiver to communicate with other devices, and a DS1307 RTC module for timestamping. Considering that the RTC module should be powered constantly, it is recommended that this module have a lithium battery at all times, even when the device itself is connected to a rechargeable battery.

Also, considering that this is an IoT device, an implementation would require other devices to be connected to the ESP32, using either UART, I2C or SPI. However, the exact implementation and data/command parsing functions of these modules should be implemented on top of the SPRN framework, since SPRN itself only provides the communication framework for the IoT network.

The root node, i.e. the node that is connected to the Internet, can use any hardware or computer that can run python; with that ranging from a server, a PC, or an SBC like Raspberry Pi.

Setting up a node requires another device that can run python, has a USB port, and has an up-to-date clock. This setup consists of setting the network ID and the pre-shared secret, and updating the time of the RTC module.

Configuration

Configuration of a module is done through UART0, which is connected to a USB-TTL converter on ESP32 development kits, and therefore can be accessed via most PCs and laptops. Configuring an ESP32 would require the device's password, which is either hard-coded during the initial implementation, or saved in the NVS.

Each time that data is read from RX1, an interrupt is called which processes the data. The general structure of a configuration packet is as follows:

Password (varying length): The static password of the module, which is an ASCII string, set during implementation.

EOF (1 byte): One '\0' character, indicating the end of the password.

Secret (30 bytes): The network secret, which is shared throughout the network.

Network ID (4 bytes): The network ID.

Time (4 bytes): The RTC time, used to set the RTC module.

If a packet is deemed valid by the device, the network data is written into memory, and the RTC time is updated. After this, the device restarts.

Device Startup

A device that has started and successfully read the network data, will start sending out SRCH packets, and waits for ADP packages. After checking the received ADP packets, the device will send out a

CHECK packet to the chosen parent. After this, the device will begin the sensor data collection and CHECK transmission cycles.

Packet Reception

The Radio transceiver is connected to the ESP32 via UART2, which is initiated at startup and has its RX interrupt enabled. The built-in RX FIFO of the HC-12 only offers a maximum of 64 bytes for storage, and is therefore not sufficient for packet reception in SPRN. With each incoming byte, the byte is written into a software implemented FIFO in ESP32, and during each clock cycle, the FIFO is given to a parse function to see if it contains a valid packet.

If the FIFO contains a valid packet, the parse function return the valid packet, and deletes it from the FIFO. If the packet is not completely received yet, the parse function will return NULL without changing the FIFO, and if the head of the function does not contain a valid packet, one byte will be removed from the beginning of the FIFO, and the parse function is called again.

Packet Transmission

Each device has a transmission queue, which consists of packets that are supposed to be transmitted. A member of the queue would have the plaintext version of the packet, and the sequence number of its last transmission. These packets are then randomly chosen for transmission.

Each time that the ESP chooses a package to transmit, it is assigned a sequence number, the plaintext is encrypted using the cipher. After the final packet is ready, a timer is set for a random amount of time, and a reception flag is set to false. Once the timer runs out, the timer interrupt checks if the reception flag is still false, i.e. no packet was received during the wait time, the packet is written to TX2. If the flag is set to true, it will be reset, and the timer will be set for a random period once again. Note that with each transmission attempt, a new sequence number is generated, and the packet is encrypted anew.

Once a MSG or CMD packet is sent, the device will wait for an ACK packet, after receiving which, the packet will be removed from the transmission queue. Other packet types are removed from the queue after transmission.

Connection Status Detection

As an IoT device, each node needs to know if its parent or children are still connected, or have gone silent. To do so, a device assigns a timeout counter to each connected node, and a timer for the last transmission to each of its child nodes.

The reception of an ACK packet from a node will reset its timeout counter to zero. If during the transmission, the device encounters a packet that was transmitted once before, i.e. one that has a sequence number assigned to its queue member, it will count it as a timeout. Once the number of timeouts for a node reaches a certain number, the device will assume that the said node has turned off and will remove it from its connected nodes. For a child node, this would mean removing it from the

array of children. For a parent node, this would mean sending the device state machine back to the startup and searching for a new parent.

Each child device also has a last transmission assigned to it, which is set to current time with each transmission received from it. To keep the connection alive, a child device would have to periodically send CHECK packets to its parent.