Lockout Communications Protocol

Contents

Introduction

This document outlines the behavior of the all communications done in the lockout system, which is comprised of the following:

- 915MHz radio communication (SPIRIT1)
- Network communication between the gateways and server (WebSocket)
- UART/Serial communication between the Raspberry Pi and microcontroller on the gateway

Lockout devices and gateways each contain a SPIRIT1 radio transceiver IC (Integrated Circuit). These radios communicate using the 915MHz ISM band. Communication on the 915MHz network is done solely between a device and a gateway.

Gateways contain a Raspberry Pi, a small computer running Linux. This device is connected to one of Rowan's WiFi networks, and communicates with the 915MHz radio module through a separate microcontroller. Gateways communicate with the server using WebSockets. The gateway Pi and microcontroller communicate over a UART (serial) connection. The data sent over UART and WebSockets is very similar to the payload of the radio packets, with an additional destination address field.

SPIRIT1 Radio Transceiver

This radio transceiver implements quite a lot of useful functionality right into the hardware, of which the following is used:

- AES-128 encryption coprocessor
- Packet handling engine
- SPI interface
- RX FIFO buffer
- Automatic CRC handling
- CSMA (carrier-sense multiple access)
- Address-based packet filtering

The SPIRIT1 datasheet is a very useful resource, and should be referred to when reading this document. Sections 7, 8, and 9 are the most relevant.

CSMA is an important feature: before transmitting, the SPIRIT1 will check if the channel it is going to use is already busy. If so, it chooses a different channel and repeats the process. This ensures that if packets are sent simultaneously, they do not interfere. Also if there are other 915MHz transmitters in the area, it will choose channels that do not interfere with those. This makes channel selection automatic, though doing a manual check of the radio usage In the area should still be done.

The automatic packet filtering feature will ensure only packets with the destination address equal to the device's own address are received.

Radio Parameters

These describe the fundamental nature of the communication technology itself, and can be changed in software.

Parameter	**Value**
Base Frequency	915MHz
Channel Space	100kHz
Modulation	FSK
Data Rate	38.4kbps
Frequency Division	20kHz
Bandwidth	100kHz
Power Index	7
Power DBM	11.6
RSSI Threshold	-120
CSMA RSSI Threshold	-90

Radio Packet Structure

Each packet sent between radios uses the following packet structure. Each packet is 30 bytes long. The SPIRIT1's basic packet handling engine is used, which implements this packet structure automatically. The packet handler allow for variable length payloads, but the lockout system uses the same length (30 bytes) for all communication, to reduce software complexity.

Field	**Length**	**Description**
Preamble	4 bytes	'10101010' sequence
Sync	4 bytes	
Length	1 byte	Length of packet
Address	1 byte	Destination address
Payload	19 bytes	Actual data to be sent
CRC	1 byte	For checking packet integrity

Encryption

The radio packets are encrypted using the AES-128 encryption standard. The computation is done by the SPIRIT1 radio modules in batches of 16 bytes at a time. Since the full packet is greater than 16 bytes, the first 16 bytes, and the remaining bytes are encrypted separately.

The 16 byte encryption key is used across all of the devices and gateways, and stored in the program memory of the STM32 microcontrollers. In the source code, the key is stored in the header file aes_key.h. This header file is not present in the version controlled source code on Gitlab.

Radio Payload Structure

Field	**Length**	**Index**	**Description**
Command	1 byte	[0]	
Source Address	1 byte	[1]	Address of sender

Tag	1 byte	[2]	Increments every packet transmitted, for deduplication
Retry	1 byte	[3]	Increments when packet is sent as a retry
Data	16bytes	[4:19]	Argument of command

Websocket/UART Packet Structure

Field	Length	Index	Description
Start Byte	1 byte	---	Only for UART
Command	1 byte	[0]	
Source Address	1 byte	[1]	Address of sender
Destination Address	1 byte	[2]	Address of receiver
Tag	1 byte	[3]	Increments every packet transmitted, for deduplication
Retry	1 byte	[4]	Increments when packet is sent as a retry
RSSI	1 byte	[5]	RSSI of gateway received packets
Data	16bytes	[6:21]	Card ID, device unique ID, etc.
End Byte	1 byte	---	Only for UART

Commands

Name	Byte	Directionality	Description	Data Field
ACK	0xff	bidirectional	Acknowledge last received command	Command, tag and RSSI of last received packet
Ping	0x01	bidirectional	Get an ACK response from from destination	Empty
Request Address	0x02	Device -> server	Device requests a new address	12-byte unique ID of device
Provide Address	0x03	Server -> device	Server provides a new address to a device	12-byte unique ID of device, new address
Access Request	0x10	Device -> server	Access to a machine is requested after ID insert	16-byte card ID number
Access Response	0x11	Server -> device	Server responds to Access Request	Denied: 0x00 Accepted: 0x01 Errors: 0x02+
Machine Control	0x12	Server -> device	Change machine state from server	Disable: 0x00 Enable: 0x01
Machine State	0x13	Device -> server	Device updates server on whether a machine is enabled or disabled	Disabled: 0x00 Keyswitch Enabled: 0x01
User Button	0x14	Device -> server	Notify server that someone pressed the help, broken, or report	Help: 0x00 Report: 0x01 Broken: 0x02

			buttons	
Set Force Disable Status	0x15	Server -> device	Change whether force disable mode is on	Off: 0x00 On: 0x01

Common Addresses

Name	**Byte**
Unassigned address	0xff
Generic Gateway	0x01

Radio Communications Overview

Device to Server Communication

A packet sent from a device to the server will use the generic gateway address as the destination address, and the device's address as the source address. All gateways that receive the packet will relay it to the server. Gateways are configured to filter packets based on the generic gateway address. The gateway microcontroller sends the packet payload along with signal strength information (not yet implemented) to the gateway Pi over UART. The gateway Pi also adds the message payload to a log file. The Pi sends the message over the network (WebSocket) to the server and adds the packet to a temporal cache (not yet implemented). There may be multiple gateways that receive the same packet, in this case each gateway relays the message to the server.

Server to Device Communication

A packet sent from the server uses device's address that the message should be sent to as the destination address, and the generic gateway address as the source address. The server sends the response to all gateways, over multiple individual WebSocket connections. Each gateway logs the message and sends it to the gateway microcontroller over UART, which transmits the packet over the 915MHz network. Each device within range will receive the response, but those whose address does not match will discard the message due to the SPIRIT1 packet filtering functionality.

Acknowledgement

Once a device or the server sends a packet, it expects an ACK to be sent for confirmation. This is necessary even if a response is expected anyway (such as Access Request). The only exception is Request Address, where no ACK is provided aside from the Provide Address.

The data field of the ACK packet contains the command, tag and RSSI of the received packet. The recipient of the ACK verifies that the command and tag match the packet it originally sent.

Retries

When a device of the server does not receive an ACK after a certain amount of time after sending a packet, it will send the original packet again. The retry field of the packet is normally 0, but is incremented every time a retry is sent.

Device Addressing

When a device is turned on, it transmits the Request Address command every 3 seconds, which includes the microcontroller's unique 96-bit ID in the data field.

When the server receives a Request Address command, it creates a new address and correlates it with the 96-bit ID in a database table. The server sends a Provide Address command, still using the Generic Unassigned address as the destination address, and including the new address and the 96-bit ID in the data field of the payload. The 96-bit ID is contained in the first 12 bytes of the payload, and the 13th byte is the new address. Any device with the Generic Unassigned address will receive this message. Devices will check to ensure the 96-bit ID matches their own. If the IDs match, the device writes the new address to EEPROM and exits pairing mode.

If a Request Address command is received by the server containing a 96-bit ID that is already in the database, the server responds with the address that it has already assigned.

Once a device is given an address, it is up to the user to enter the description fields of what this device is into the server.

The user interface of the server allows users to generate a new address for devices that have already been registered.

Packet Deduplication

When the server receives a packet, it will check to see if the tag and source address exactly match the packet that was sent last. If they match, it is assumed it is a duplicate and it will be ignored. This same processes is done when a device receives a packet, but only the tag is checked, since the source address will always be the same (generic gateway address).

Access Response Error Codes

Code	**Error Message**
0x02	Machine not set up in server
0x03	ID number does not exist in server