# SmartGarden RPi Gateway Serial Protocol

## Introduction

SmartGarden RPi Gateway is the “Smart IO” controller based on Arduino. It controls RFM69 RF module as well as basic IO (GPIO and Analog) for RaspberryPi – based SmartGarden v2.0 station. Functionally RPi Gateway is similar to XBee RF modules that combine RF circuitry with a local IO processor and basic GPIO/Analog.

SG RPi Gateway is connected to Raspberry Pi via serial link (using RPi onboard serial interface or USB-Serial converter), and uses packet-based communication protocol. This document describes this protocol.

## SG RPi Gateway Serial Protocol

The serial protocol is a classic message-oriented protocol running over serial channel. Because serial link is inherently unreliable, the protocol includes provisions for error detection and retries.

Conceptually the protocol design is similar to XBee serial communication protocol (in API mode), although it is not compatible with XBee protocol. This is done on purpose, due to potential patent and/or copyright issues with XBee serialization protocol. Use of a custom protocol as opposed to directly supporting XBee protocol allows certain overhead reduction by eliminating elements not applicable to RPi Gateway.

## SG RPiG Serial Protocol Structure

The protocol uses classic frame-based communication with byte stuffing. Each frame consists of:

0x7E

Start Delimiter  
Byte 1

1 byte

FrameID  
Byte 5

1 byte

Checksum  
Byte n+1

Data

Payload  
Byte 6-n

MSB

Frame Length  
Bytes 2-3

LSB

1 byte

Chan/CMD  
Byte 4

Bytes escaped if needed

Each frame starts with the start delimiter byte (0x7E), followed by frame length (2 bytes, MSB first). 4th byte is the control code, which consists of Channel number (upper 4 bits) and Channel Command (lower 4 bits), followed by the frame ID (used to report status and set to 0 if no response is required), and the payload data block (can be empty). Last byte in the frame is the checksum.

When sending or receiving a data frame, specific data values must be escaped (classic byte stuffing). To escape an interfering data byte, escape character (0x7D) is inserted followed with the byte to be escaped XOR’d with 0x20.

**Data bytes that needs to be escaped:**

* 0x7E – Frame Delimiter
* 0x7D – Escape
* 0x11 – XON
* 0x13 – XOFF

**Checksum:**

To test data integrity, a checksum is calculated and verified on non-escaped data.

**To calculate:** Not including frame delimiters and length add all bytes keeping only the lowest 8 bits of the result and subtract from 0xFF.

**To verify:** Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

The same framing structure is used for both sending data & commands to the Gateway module, and for sending data and for receiving data & status from the Gateway module.

This framing and checksum calculation is similar to XBee UART framing in API Mode 2.

## SG RPi GW Channels

SG RPi Gateway serial communication uses the notion of “channels”, which represent logical communication pipe over the basic message framing. Following channels are defined:

|  |  |  |
| --- | --- | --- |
| Channel# | Name | Description |
| 0 | Gateway-Common | SG RPi Gateway itself. Used to get and set common parameters, reset the gateway etc. |
| 1 | GPIO | Digital GPIO channel, used for input/output operations on the gateway IO pins and for pins configuration (for input/output etc) |
| 2 | Analog | Analog GPIO channel, used for input/output of analog data as well as configuring analog pins |
| 3 | RF1 | 1st RF interface. Send/Receive/Configure operations on this RF channel. |
| 4-15 | Not in use | Not in use in v1.0 |

## SG RPi GW Channel Commands

Each channel implements commands appropriate for the channel. By convention three command codes are common for all channels:

Command 0: Get config (MCU->Host)  
Command 1: Set config (MCU->Host)

Command 15: Reply (Gateway->Host)

Actual meaning of config is specific to each channel.

Reply is a common command code used to send responses from Gateway to the Host. First byte of the Reply payload is the status (0 == OK, non-zero means failure code), the rest of the payload may be filled in with requested data (command-specific).

**Gateway-Common Channel Commands**

Gateway-Common (#0) channel in v1.0 implements four commands:

* Get Config (#0)

Get config command responds with a set of key=value pairs reporting gateway hardware (‘gwhw\_v’) and firmware (‘gwfw\_v’) versions. Key=value pairs are separated by semicolon (;).

* Set Config (#1)

Command is accepted, but no specific settings are defined in v1.0.

* Reset (#3)

When receiving this command Gateway will reset itself.

* Reply (#15)

Gateway uses this command to send reply to the Host.

The reply can be either byte of 0/1 (for OK/Fail status), the rest of payload (optional) will have response data. E.g. Reply to Get Config will have a list of current settings in the form of Key=value pairs separated by semicolon.

**RF1 Channel Commands**

RF1 (#3) channel in v1.0 implements 6 commands:

* Get Config (#0) and
* Set Config (#1)

Get Config command responds with a set of key=value pairs reporting RF module settings. Key=value pairs are separated by semicolon (;).

RFChan=nn (radio channel number)

PANID=nn (PAN ID to use)

Addr=nn (Network address of the node)

EncOn=yes/no (indicates whether encryption is On or Off)

EncKey=key (if encryption is enabled, key will represent the encryption key)

Set Config command accepts the same list of Key=value pairs with the same meaning.

* Send (#2)

This command sends data via RF channel. First byte of the payload represents destination network address, or 255 for broadcast. Second byte is the Control byte, with individual flags indicating whether to use ACK or not, number of retries, and whether to notify of successful message delivery (i.e. that ACK is received) or not.

The rest of payload represents data to send.

Send command will handle ACK and retries for unicast transmissions according to flags in the Control byte. For broadcast transmissions no ACK is used.

If ACK was requested and flags indicate that notification to Host is required, after completing the transmission Gateway will send Send-Completed frame.

* Mode (#3)

This command controls RF module mode. Payload is one byte, and can be:

1. Sleep (puts RF module to sleep)
2. Receive (puts RF module into receive mode)
3. Reset (cancels all pending transmissions, discards received data and resets RF module)
4. Reply (#15)

* Received (#4)

Gateway uses this command to report received packets from RF channel.

1st byte of the payload indicates success/failure (should be 0 for successfully received packets), second byte is the SenderID, third byte is the DestinationID (which might be specific address or 255 for broadcast), the rest of payload is the actual received data.

* Reply (#15)

Gateway uses this command to send reply to the Host in response to Host commands.

The first byte in the reply payload can be 0/1 (for OK/Fail status), the rest of payload (optional) will have response data. E.g. Reply to Get Config will have a list of current settings in the form of Key=value pairs separated by semicolon.

For Send command reply will indicate whether the command was accepted. Actual Send command results will be delivered later on, as a Send-Completed command.

* Send-Completed (#14)

Gateway uses this command to send reply to the Host to report completion of the Send command, if requested to do it by flags in the Control byte in the Send command.

Payload of the Send-Completed command are:

Byte 1: Status   
which can be 0-Success, 1-No ACK, 2 – RF unavailable (noise on the line),

3 – cancelled before transmission was completed

Byte 2: Destination address the packet was sent to