Examples

WirelessControl

On each loop cycle, a message is transmitted and a message is received from a remote node or a timeout occurs. The message contains a control command. The control command turns the red LED on or off. Both nodes can control each other's red LED through a button press. When the button on one node is pressed or held down, the red LED should light up on the remote node, and vice versa.

WirelessMonitorHub

The hub acts as a simple receiver for a star network. The hub node can receive both broadcast messages and messages directed to its assigned address. The hub node should not be assigned to address 0 (broadcast address). To create an additional network, simply change the hub address.

WirelessMonitorSensor

The sensor acts as a simple transmitter for a star network. The sensor node can transmit any type of message one-way using the broadcast address. To create multiple networks, simply change the address on the hub node and have the nodes in that network transmit to the assigned hub address.

WirelessTest

Each radio will send a message consisting of: 1 byte counter, 5 byte static text. The counter will count from 0 to 9 and will rollover. Each radio will wait in receive mode for approximately one second. Upon receiving data, or timeout of one second, the radio receive function will return. If valid data was received, the radio's receiverOn() method will return the number of bytes that were received. In this example, the data can be monitored on the serial port (please refer to printTxData() and printRxData() functions).

Public Interface

begin(address, channel, power)

Setup the SPI peripheral and I/O, GDO0 interrupt I/O, and initialize the radio session.

Parameters: **address**: uint8_t

Default device address used for hardware message filtering.

channel: channel_t

Default frequency to receive/transmit on.

power: power_t

Default output power level to transmit at.

Returns: Nothing

end()

Close the radio session.

Parameters: None

Returns: Nothing

busy()

Radio busy indicator (transmitter active flag).

Parameters: None

Returns: True if the transmitter is currently in use; false otherwise.

setAddress(address)

Set device address. This address is used for hardware message filtering. If a message is received but does not match the device address and is not a broadcast (message sent to broadcast address of 0x00), the message is automatically discarded; the radio driver is never notified.

Parameters: **address**: uint8_t

The device address of the receiving node.

Returns: Nothing

setChannel(channel)

Set operating frequency.

Parameters: **channel**: channel_t

Frequency to receive/transmit on.

Returns: Nothing

setPower(power)

Set operating transmit output power.

Parameters: **power**: power_t

Output power level to transmit at.

Returns: Nothing

getRssi()

Read the receive signal strength indicator (RSSI) for the last received data stream.

Parameters: None

Returns: RSSI value in absolute dBm increments.

getLqi()

Read the link quality indicator (LQI) for the last received data stream.

Parameters: None

Returns: LQI value.

getCrcBit()

Read the cyclic redundancy check (CRC) bit for the last received data stream.

Parameters: None

Returns: CRC bit value – valid (1) or invalid (0).

transmit(address, dataField, length)

Build a data stream from the data field provided and transmit the resulting message over-the-air to a specified address.

Parameters: **address**: uint8_t

The device address of the receiving node. This address may go to a broadcast address (0x00).

dataField: uint8_t*

Payload for the data stream.

length: uint8_t

Number of bytes in the data field buffer.

Returns: Nothing

receiverOn(dataField, length, timeout)

Turn on the radio receiver and listen until a timeout occurs.

Parameters: **dataField**: uint8_t*

Buffer that stores the data field. This buffer is assumed to be large enough to store the largest expected data field.

length: uint8_t

Size of the data field in bytes.

timeout: uint16_t

Period to listen for (maximum) in milliseconds.

Returns: Number of bytes read from the RX FIFO that were copied into the data field.