Radiator Remote Control Analysis

General

Remote control is a simple 6 button infra red remote control. Each button when pressed issues a repeating identical message all the time the control is held down.

The 6 buttons are

- On/Off – Toggles radiator on/off. On moves into Standby mode
- Temp – Moves into Temp setting. Then use up /down
- Mode – Cycles round Standby, Frost Free, Eco, Comfortable
- Time – Moves into Time setting. Up/Down sets a countdown timer which turns on from Standby, or turns off from the other modes
- Down
- Up

Messages

Messages are 12 bits long

Bits are just encoded as a simple on/off pulse. High frequency modulation is not used.

Each bit is encoded as a pulse 1.7mS long.

A 0 is 1.3mS IR-On, followed by 0.4mS IR-Off

A 1 is 0.45mS IR-On, followed by 1.25mS IR-Off (Probably periods as 0 but rise time affects)

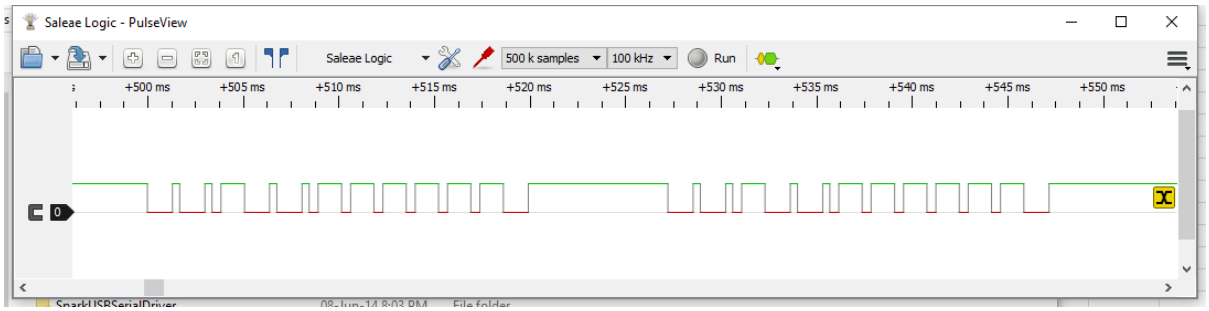
Message takes 12 * 1.7mS to send (20.4mS)

Gap between repeats of same message 6.6mS

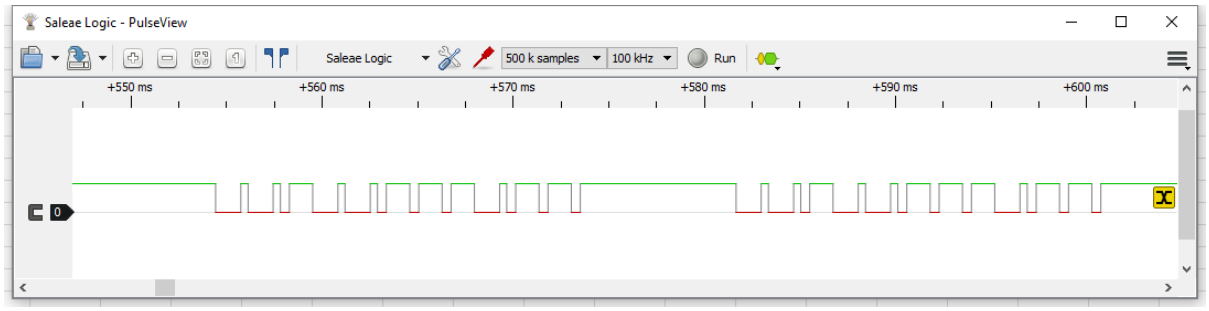
A longer gap would be present when button released (say 500mS). So a sequence of ups for example could be send Up repeats for 250mS, pause 500mS, send Ups for 250mS etc.

Button Messages

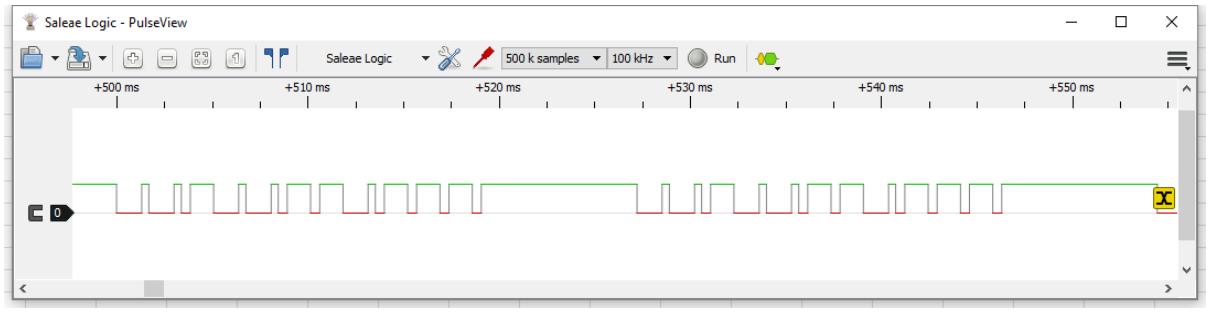
On/Off



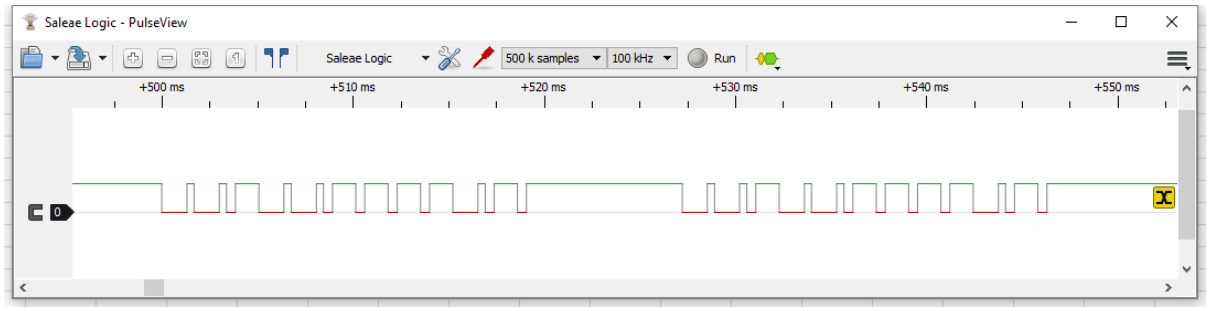
Temp



Mode

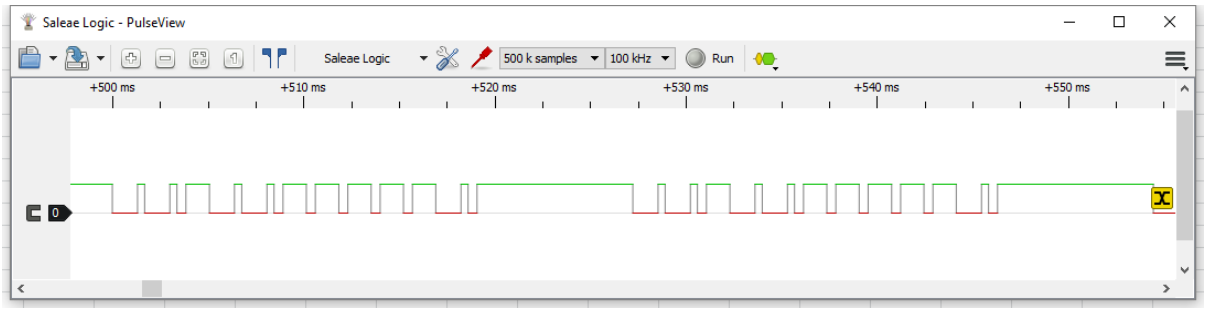


Time



Down

Up



DataBit	0	1	2	3	4	5	6	7	8	9	10	11
On/Off	0	0	1	0	0	1	1	1	1	1	1	0
Temp	0	0	1	0	0	1	1	1	0	1	1	1
Mode	0	0	1	0	0	1	1	0	1	1	1	1
Time	0	0	1	0	0	1	1	1	1	0	1	1
Down	0	0	1	0	0	0	1	1	1	0	0	1
Up	0	0	1	0	0	1	1	1	1	1	0	1

Control Strategies

Main purpose is to provide proper time of day control rather than simplistic countdown which needs to be set up every time. Strategies are complicated by the toggle nature of on/off and the cyclic mode.

Use Photon with IR LED to simulate remote control. Photon can get time easily from Internet. Phone app could be used to set up and provide manual control using Photon cloud integration. If Phone App was used exclusively instead of remote then it would be easier to know what state the radiator was in as all commands pass through Photon.

Assume Off Strategy

If radiator is assumed to be off then it can be turned on by sending an On/Off message followed by a sequence of Mode, Mode, Mode messages.

Similarly if it is assumed to be On then a simple On/Off message would turn it off.

Use timer strategy

Leave radiator permanently in Standby. Turn on by sending a Time, Time sequence (to get it to minutes) followed by up. 1 minute later radiator will turn on.

Similarly if On then send Time, Time, Up and 1 minute later it will turn Off

Other strategies

Explore sending other code sequences not supported by remote buttons to see if there are explicit off and on commands

Find some way of detecting whether radiator is on (detect status light or temp sensor on radiator)

Basic Library

Library supports transmission of button press sequences. A sequence is defined as code words (uint16_t) in a buffer. Code words have 3 types of value

- 0x0000 - 0x1000 defines a button press code (Bit 11 transmitted first)
- 0x1000 – 0x7fff unused
- 0x8000 – 0xfffe delay in units of 25mS (mask of top bit)
- 0xffff end of sequence (must be present)

The library is constructed as a state machine inside an interrupt service routine called at 100 uSec intervals while transmission of a sequence is in progress.

The API into the library consists of 3 calls

- void radtx_setup(int pin, byte repeats, byte invert, int period) Called once to define basic transmission configuration
- boolean radtx_free() Called to see if a transmission is in progress. Should be used to check when it is safe to send another sequence
- void radtx_send(uint16_t *msg) Used to send a message sequence. Automatically activates the service routine to initiate sending
- extern void rad_debug() Used during debug. Returns an int (default 0) radtx_debug which can be set in various routines

Support routines consist of variants of timer handlers to match the sbc board type. Which set is used is determined by #define in RadTx.h

- extern void rad_timer_Setup(void (*isrCallback)(), int period)
- extern void rad_timer_Start()
- extern void rad_timer_Stop()

Note that Particle versions require use of SparkIntervalTimer library to control timer.

Basic Run Time

A simple Particle (Photon) run time app RadTXTest.ino uses the library for testing and as starting point to develop full app.

It defines 10 message sequences of 10 button presses. It sets up a function called sendMsg which may be called via the Particle Cloud with

https://api.particle.io/v1/devices/devicecode/sendMsg?access_token=accesstoken" method="POST"

where there is a named POST variable args containing a (numeric) string value 0 to 9 which selects one of the pre-defined messages sequences.

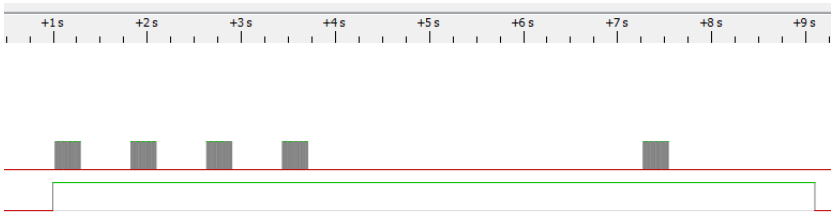
The remote buttons have predefined constants

BTN_ONOFF	0x027e
BTN_TEMP	0x0277
BTN_MODE	0x026f
BTN_TIME	0x027b
BTN_DOWN	0x0239
BTN_UP	0x027d

Example output for a message sequence

BTN_ONOFF,BTN_MODE,BTN_MODE,BTN_MODE,0x8078,BTN_ONOFF,0x8028,0xffff,0,0

Overall Sequence transmit



One Button press repeating 10 times

One button timing