Radio Interface Documentation.

This document describes the protocols used to transmit data between collect device and sensor device.

A collector device is a device which primarily receives data, from one or more sensor devices. It's only transmission is to send back acknowledgement.

A sensor device is a device which primearily function is to collect data (from sensors) and transmit to that data to a collector device.

In the described interface handshaking is required, and any device which does not return handshakes for as described in this spec will be inorged, for this reason, both transmitter and receiver modules are needed on each device

Data transmissions: These are from sensor to collector, and must abide by the following:

Parity bits must be used on every byte transmitted. These are calculated, by creating a bit array from the byte to be transmitted. This bit array is then turned into a 3x3 square, where the top left corner is left empty (0). Then, for every row, if the number each bit is made even. For example, if there is 1 '1' in a row, the parity of that row is a 1, to make the number of 1's even. So that this statement is true:

```
all_bits_in_row + parity_bit modulo 2 MUST EQUAL 0
```

The same is done for each column.

This will leave 6 parity bits, they should be arranged into a byte in the following way:

```
Placeholder0 placeholder0 row1 row2 row3 column1 column2 column3
```

Where row1 is the top row, and column1 is the left column.

Each byte of data must be transmitted followed by it's parity bit, for example

```
byte1, parity_byte1, byte2, parity_byte2, byte3 etc…
```

2 data bytes (with parities) and 1 checksum (with parity) are to be transmitted at a time. The collector will always expect bytes with parities, so these bytes must be filled.

The check sum is calculated as follows

```
checksum = (byte1 + byte2 + secret1 + secret2)/4
```

Where the secrets is the collectors passcodes. This will only be known by authenticated sensors and stops the insertionation of false data. There are 2 secrets as bytes 1 and 2 can be seen as during transmission, and could be used algebraically to calculate a single secret.

A data transmission must start with the collectors id, “:01”

Receivers:

6 bytes after “:01” will be considered purposely sent, and read into variables, all other data will be consider 'noise'

Every second byte read will be considered a parity byte of the byte before it.

The collector will turn all data received into bit arrays, and then test if rows and columns are even, according to added parities. If not, it will look for an intersection, and flip the bit at the intersection. If rows are uneven, but columns are, or visa versa, it will assume there was an error in parity bit transmission, and conutune with the given data.

After parity correction, a collector must recalculate the check sum, based on

```
checksum = (byte1 + byte2 + secret1 + secret2)/4
```

Used the first two bytes it receives and it's own secret. It must then compare this to the 3rd byte it received, if they are the same, accept the data packet. If they are not, it must reject the data packet.

On accepting a data packet:

The collector must broadcast ':02a'

On rejecting a data packet:

The collector must broadcast: ':02r'

This alerts the send to the status of the packet. They may then choose how to respond.

After data transmission:

After transmitting, a sensor should listen for a response. It should listen with a timeout of 10ms before resending a packet. It should do this in a loop for n number of times before giving up, where n is based on how critical the data is.

It should be listening for ':02' followed by either an 'a' or an 'e'

If 'a':

The last data packet was accepted, don't resent.

If 'r'

The last data packet was rejected, resend for n number of times until you give up.

After the letter, the collector node will also send the id number of the sensor it was referring to, encase 2 sensors transmit as similar times. See the id system document.