WAVENED WSN FOR SMART HOME EXPERIMENTS

SPECIFICATION

Document information

Title	WaveNed WSN for smart home experiments
Subject	Specification
Current version	0.1
Status	preliminary release
Author	Joost van Velzen
Project id	N/A
Customer	WoTS 2016
Filename	WaveNed WSN for smart home experiments.doc
Document Id	WOTS2016-WSN1
Template	SE_EN_basic.dot
Last updated	06-Jun-16 11:04:00 by Joost van Velzen.
Number of pages	7, including front page and this information page.

Revision history

Date	Version	Status	Author	Description	Distribution
06-06-2016	0.1	Preliminary	JvV	First release	External

CONTENTS

1	introduction	4
2	overview	5
3	interface	6
3.1	Serial port settings	6
3.2	Serial data format.	6

1 INTRODUCTION

Dear hacker, let me introduce myself: I am Joost van Velzen. I run the innovation department at SallandElectronics, which basically means that my job is one long hackathon. I often do large scientific experiments with cutting edge technology together with universities. Last December we did a very exciting experiment with 1000 wireless bracelets. You get to win 16 of these bracelets pre-programmed with custom firmware, specially designed for hacking together a cool smart home solution.

I cannot open up the source of the firmware (at this moment), so I tried to give you the most low level interface I could come up with, which means you get maximum freedom to do whatever you like with these.

Let me be very clear about this: The sensornetwork is experimental hardware and software and quite unique. You can’t buy this: the only way to get these is to win the hackathon…

Good luck!

Joost van Velzen

2 OVERVIEW

The kit is a wireless sensornetwork running the WaveNed fusion MAC. The kit consists of:

- Sniffers (to get the sensor data to a computer)
- Anchors (for indoor localization, point of interest tagging or presence detection)
- Wristbands with button and 3D accelerometer (for wearing)
- Micro usb cables for charging / connecting the sniffer.

The wristband is a slapband type. There is no battery, but instead the devices have a supercap that charges to full capacity in roughly 90 seconds.

Installation

I will not be pre-installing the anchors: you get to choose where to put them and how you want to use them. My recommendation would be to first just have one wristband, one sniffer and two anchors per team to play around with and do some desk testing, but if you keep things to yourself and don't share the hardware during experiments, then you are going to be limiting what you can do with the sensornetwork. So, I highly recommend you share with the other teams for bigger experiments (so you can win their hardware in the end ☺).

Charging, connecting and leds

Please take care not to bend the USB cable when it is plugged into the device: It is an SMT micro-usb port and yes you can break it if you are not careful. A continuous red led indicates an empty supercap, no leds at all indicate an even emptier supercap. A blinking green led indicates charging and when the green led is continuously on then the supercap is full. Blinking red and orange leds are for me to keep an eye on the mac layer and for good looks.

3 INTERFACE

I have kept the interface deliberately simple:

Each wristband will transmit roughly 11 messages per second:

- 10 of these messages contain accelero and button data
- One message also includes RSSI measurements of the other wristbands and the anchors

The sniffer will also add an RSSI measurement of the received message.

3.1 Serial port settings

Set the serialport to 460800, 8N1, no flow control.

In case you are a .Net fan, forget about Microsoft's SerialPort implementation and use the FTDI drivers directly instead.

3.2 Serial data format.

The line starts with LF and ends with CR. Thus CR is actually the start character.

The lines look as follows:

```
Cz>2FF7F00
```

```
Cz:2FF7F00A1B2D3E0F1G2H3I0J1K2L3
```

The difference between the two lines is the length, but it is also identified by the third character: a short line has a greater than sign '>', and a long line has a colon ':'. All other characters are the same and have the same meaning.

Each wristband sends the short line roughly 10 times per second and the long line just once per second.

The first character identifies the wristband. There are four and they are identified by A, B, C or D. The line in the example is sent by wristband C.

Each line has signal strengths that are sent as base64 chars. They are coloured red and green below:

```
Cz:2FF7F00A1B2D3E0F1G2H3I0J1K2L3
```

The green char indicates the signal strength with which the sniffer received the message. Value 0 (base 64 'A') is the strongest and value 63 (base64 '/') is the weakest signal, this is because the value is actually negative and has an offset about 30 that was removed for "compression".

The red chars are the signal strengths that the wristband measured from the anchors and other wristbands. In this example I put in a base64 char that corresponds to the id of the received device. So, the signal strength for wristband A is in position 11. The anchors follow the wristbands and I put E to L in as signal strength to identify them. Now as this is a message from wristband C it is obviously never going to measure any signal from itself and therefore it is omitted in the data. Have a guess what this implies for the formats send by the other three wristbands.

Following the signal strength is a 2bit number (0,1,2,3), this indicates how many seconds ago the signal strength was measured. 0 is this round, 1 is one second ago, 2 is two seconds ago and 3 is "please ignore me" as the data is simply old or even never measured.

The character in position 4 (it is a 2 in this example) indicates the state of the button. The value is 0 if the button is currently pressed. Any higher value indicates that the button is not currently pressed, but it will indicate how many messages ago the button was last pressed. Thus 1 indicates the button was pressed during the last. Two indicates it was two messages ago etc. the value is 3 bits, thus, if the value is 7, then the button was either pressed 7 messages ago or not at all.

The last 6 characters of the short message and characters 5 to 10 of the long message are accelerometer data. The values correspond to X,Y and Z and use two hexadecimal characters per value to create a single two's complement byte value.

```
Cz>2FF7F00
```

So in the example above the values coloured Red-Green-Blue are X-Y-Z in that order and they correspond to -1, 127, 0. To be able to detect the orientation in 3D it is easiest to have a range of 1 G. So 1G corresponds to 127. As we always have 1G around to keep us from being flung off this spinning planet, this means that you can quite easily detect the orientation of the device and you may even be able to pull off some slow motion gesture-based interfacing.

Now this is 2.4GHZ RF technology and at a venue like the WoTS there's going to be a lot of WiFi, Bluetooth etc. sending interfering messages. This isn't a problem, because the WSN was designed to cope with that and the mac can handle a lot of missed messages. However you need to implement some robustness into your application to cope with this: The sniffer may miss a lot of messages, especially if you use just one sniffer.

So that's all the technical details you need to know, as I said it is nicely low level. There is no way to control the leds on the wristband, but if you need lights, then head over to my description of this year's WoTS gadget. You'll find that you have plenty of lights to play with ☺.

Happy hacking and good luck!

Joost van Velzen

P.s. This specification is subject to change without prior notice (you'll probably notice when you start testing).