SoftRF

An IoT bridge between popular radio-based general aviation proximity awareness system and Wi-Fi consumer devices.

Revision: 1.0

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Table of contents

(Click on a link to proceed directly to this section)

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

List of designs by 2015

- ADS-B - upcoming world-wide aviation industry standard. Operates at specific frequency 1090 MHz dedicated to aviation ;
- FLARM – most popular collision avoidance system for gliders and some powered GA aircrafts. It utilizes ISM 868/915 MHz RF band ;
- Argus – an open source DIY project. ISM 2.4 GHz Wi-Fi band is in use ;
- OGN Tracker – open project for monitoring of glider's traffic. Operates at ISM 868 MHz band ;
- PilotAware – recent high range open hardware / closed source proximity awareness project. Operating band is ISM 868 MHz.

Concept

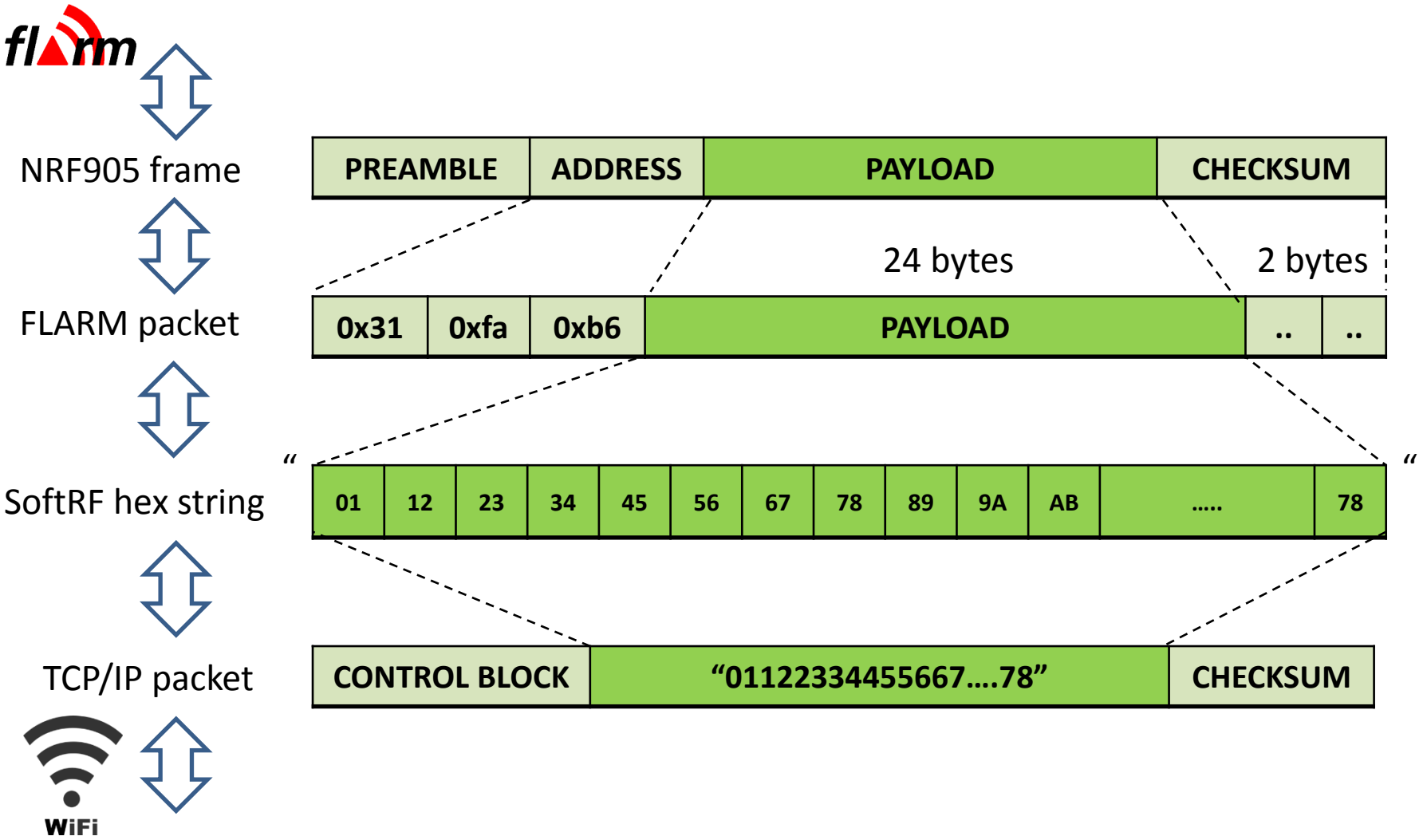
Idea is to:

1) Receive data packets on RF ISM 868/915 band from nearby FLARM transmitters with minimal hardware ;
2) Take raw (encrypted) payload from each packet and convert it into plain-text representation (hexadecimal) ;
3) Forward the text string to a nearby mobile device by Wi-Fi (TCP/IP) for further processing ;

and vice versa:

1) Receive an appropriate hexadecimal text string from mobile device by Wi-Fi ;
2) Convert it into RF raw data packet ;
3) Transmit the packet on 868/915 MHz for a nearby FLARM device(s).

Data flow



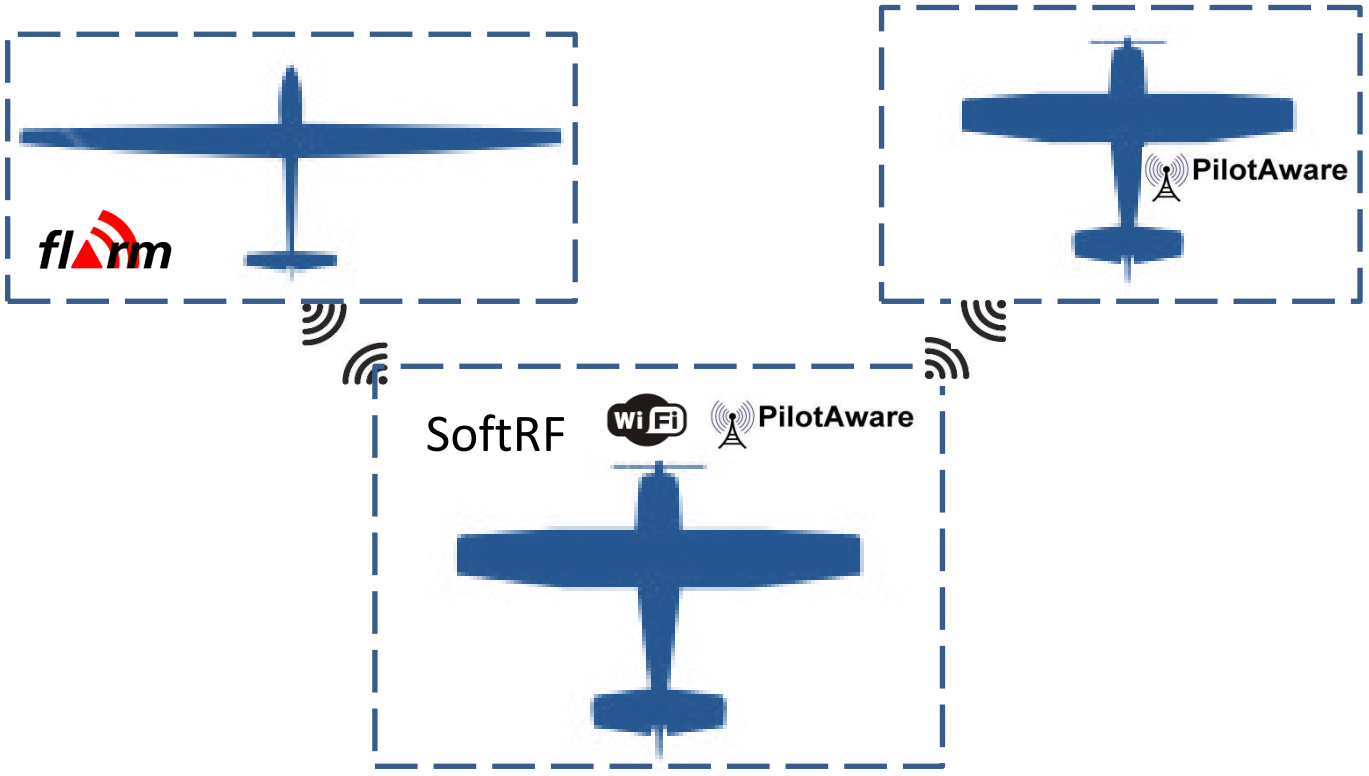
Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Areas of application

- A two-way bridge between FLARM and other traffic awareness designs ;
- Alternative to Raspberry Pi based Open Glider Network (OGN) receiver ;
- Wireless adapter for a smartphone or tablet to receive (transmit) FLARM traffic information ;
- Lightweight traffic awareness transceiver to carry onboard an UAV.

Bridge

When an aircraft has both SoftRF and other design onboard – it can interchange of positional data between the systems by Wi-Fi.

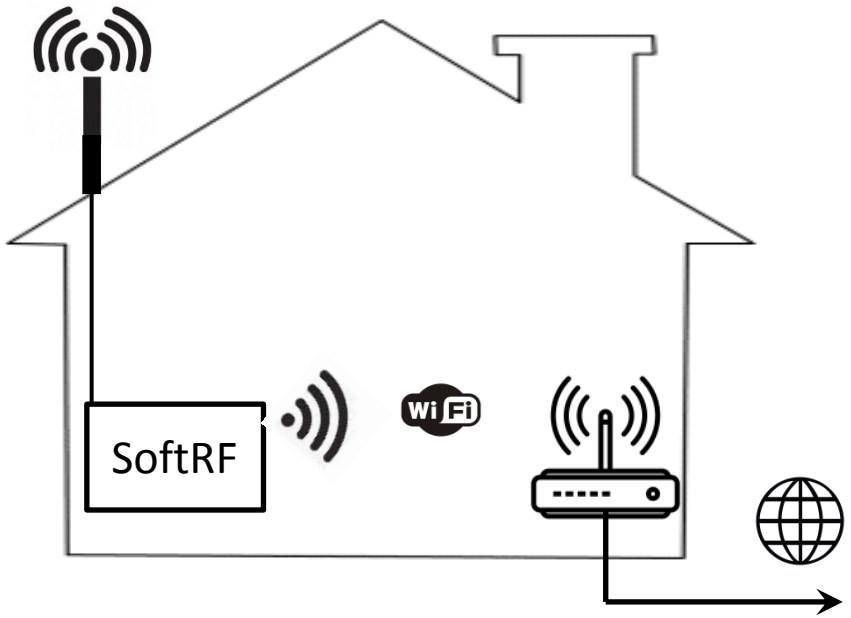


Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

OGN receiver

Generic OGN receiver does capture of radio signals from nearby flying FLARM traffic by USB DVB-T card. Upon completion of processing it sends APRS formatted messages to an OGN server by internet for visualization.

SoftRF is also capable to act as an OGN receiver. It delivers raw (encrypted) FLARM data to an upper-layer server by built-in Wi-Fi adapter through a nearby Wi-Fi hotspot.



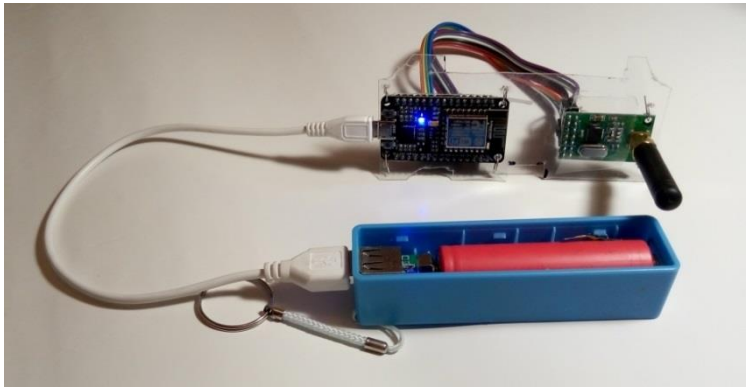
Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

OGN receiver (cont.)

Typical OGN receiver is built from:

- specific model of USB DVB-T adapter ;
- Raspberry Pi or Pi2 microcomputer ;
- flash memory card ;
- optional Wi-Fi adapter.

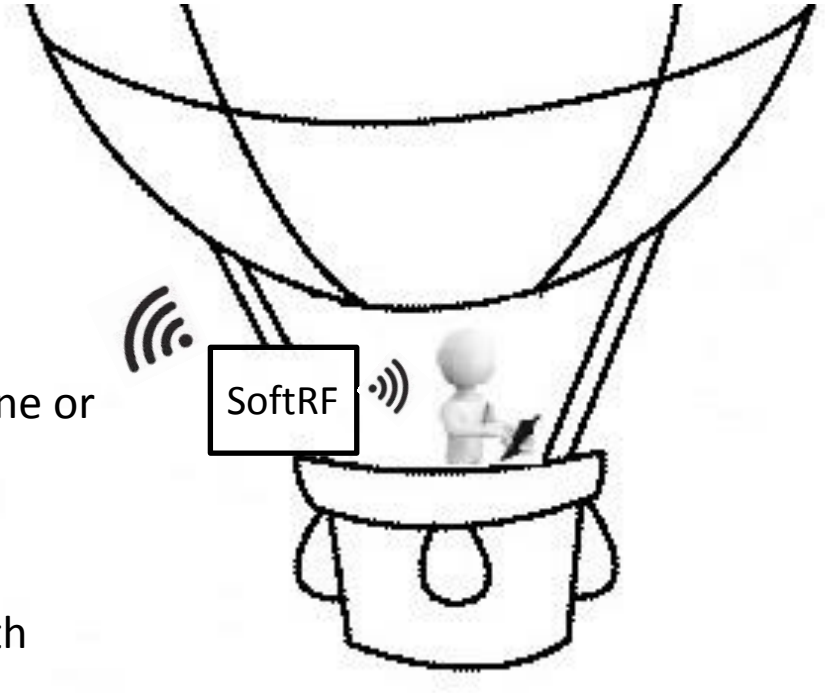
and costs around 50+ € (excluding antenna).



SoftRF is also easy to build from parts. Firmware can be updated "Over-The-Air". SoftRF based OGN receiver has very similar functionality to the Raspberry Pi 's one but 2-3 times less expensive.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Wireless adapter



It this case a mobile device such as smartphone or tablet is connected to the SoftRF by Wi-Fi.

SoftRF relays raw packets between a FLARM equipped aircraft and the mobile device, forth and back.

Processing and visualization of the FLARM data on the mobile device is performed entirely by software (an Android or/and iOS app).

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

UAV transceiver

SoftRF may complement UAV's built-in autopilot to add proximity awareness functionality.

One known good example of such autopilot is very popular ArduPilot open source platform. Most recent version does already have preliminary support for ADS-B.

To reduce radio interference on 2.4 GHz band with other control interfaces it makes sense to disable SoftRF's Wi-Fi and communicate with autopilot only by wires.

My testbed for this winter season: SoftRF prototype been mounted underneath 350-class quadcopter is shown on this picture.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

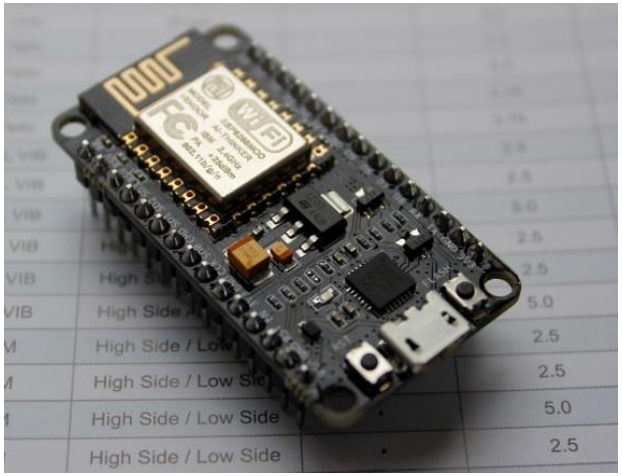
Hardware

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

NRF905 module

- This module is signal-level compatible with FLARM due to that they both use the same radio silicon - Nordic Semiconductor's NRF905 ;
- It can receive and transmit on 868 MHz (Europe) and 915 MHz (North America) bands ;
- Maximum transmit power is 10 dBm (10 mW) which is compliant with both ETSI and FCC limitations.

NodeMCU v1.0 IoT module



- NodeMCU is recent and popular open source "*Internet of Things*" (IoT) platform ;
- Key hardware features are:
  1. built-in IEEE 802.11 b/g/n Wi-Fi ;
  2. 4 Mbytes of flash memory ;
  3. 13 active GPIO pins.
- Key advantages over competitors are:
  1. wide open software support, to be specific: "*Arduino ESP8266 IDE*";
  2. low cost.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Power and connectors

Power bank provides electricity to NodeMCU module through its micro-USB port.
Capacity of the bank is not a critical factor – power consumption by the SoftRF is about 0.7 Watt.

10 jumper wires are connecting NodeMCU to NRF905 module.

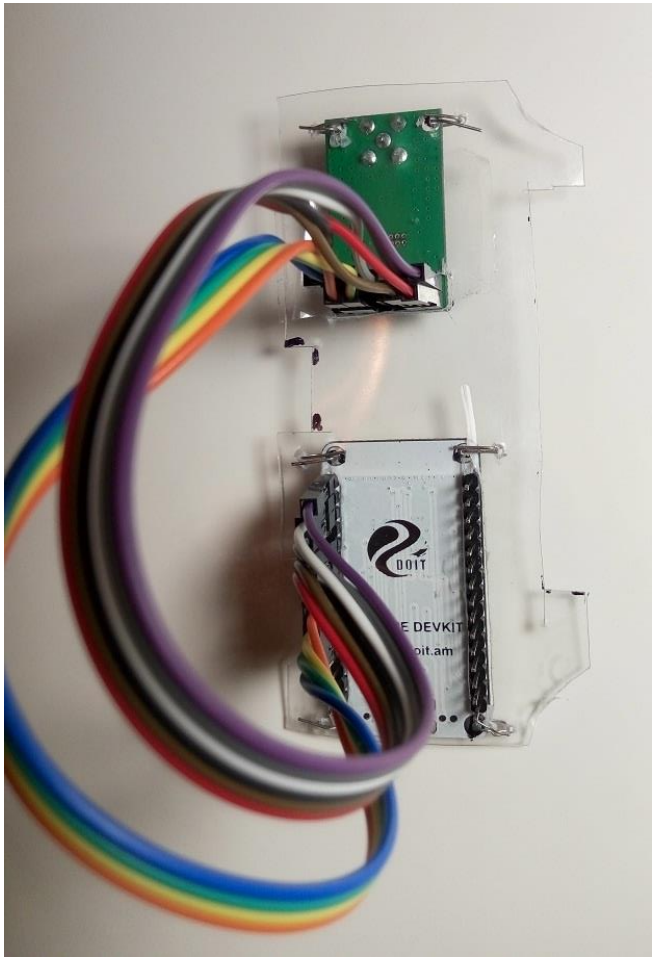
Bill of materials

Item №	Item description	QTY.	Est. price per unit, €	Source(s)
1	NRF905 module (PTR8000/8000+)	1	4-5	A, B, E
2	NodeMCU v1.0	1	4-7	A, E
3	Power bank (5 Volts) and USB<->microUSB cable	1	5-10	ANY
4	40 pcs. female DuPont jumper wires	1	1-2	A, B, E
Total		4	14-24	

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

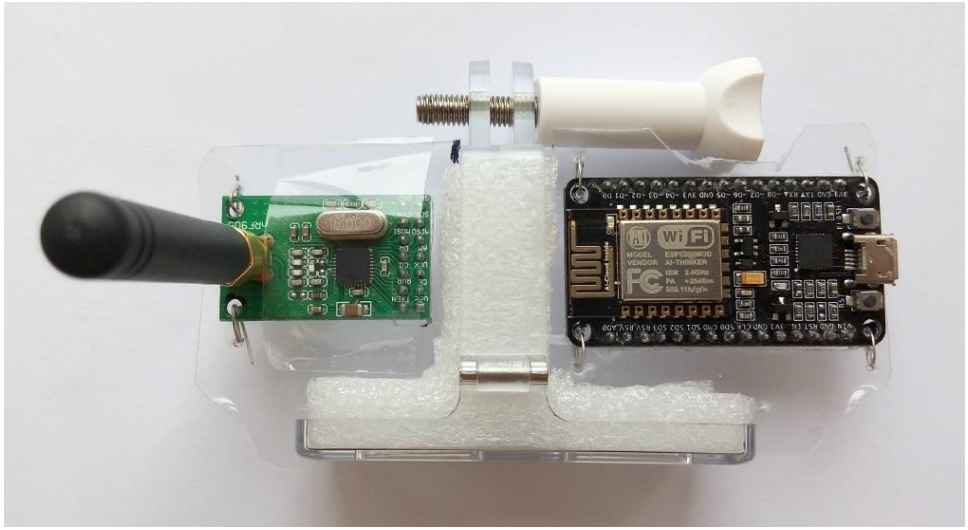
Wiring

Wire Num.	NodeMCU pin	NRF905 module
1	D0	TXEN
2	D1	DR
3	D2	PWR
4	D4	CE
5	D5	SCK
6	D6	MISO
7	D7	MOSI
8	D8	CSN
9	3V3	VCC
10	GND	GND



Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Prototype



My proto is built around a GoPro action camera's clip.

This allows me to mount the prototype

- inside a sailplane's cockpit ;
- onto a car's dashboard ;
- underneath a quadcopter.

Total weight is 41 gram.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Software

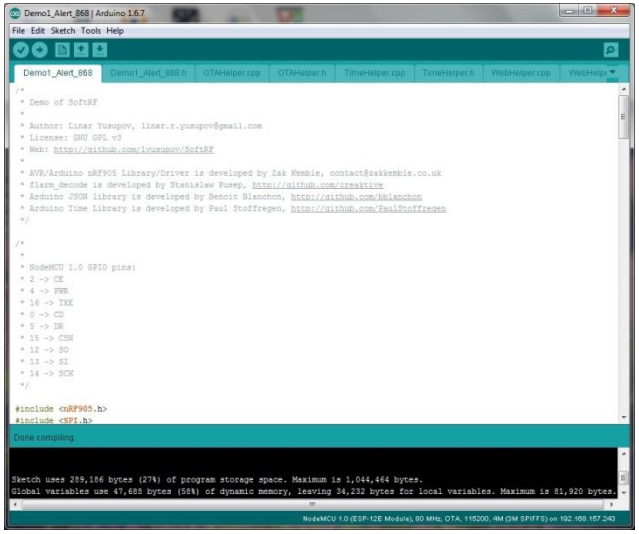
Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Intro

At first I would like to express my appreciation to the following people whose open source projects are in use by my SoftRF demo sketches:

- Arduino ESP8266 IDE from a team led by Ivan Grokhotkov ;
- Arduino nRF905 by Zak Kemble ;
- flarm_decode by Stanislaw Pusep ;
- Arduino JSON by Benoit Blanchon ;
- Arduino Time by Paul Stoffregen ;

Intro (cont.)

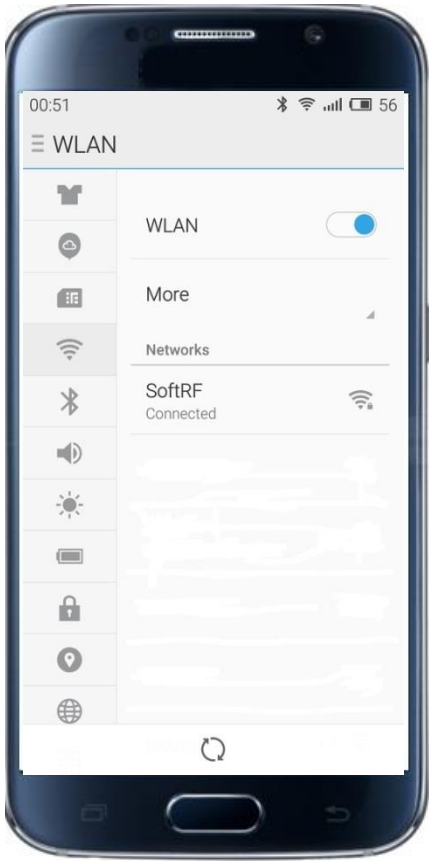


All the following demo examples are to:

- be **built** on a PC together with Arduino IDE (ver. 1.6.7 is recommended) ;
- **run** on the SoftRF hardware specified earlier ;

- and **visualized** on a Wi-Fi connected Android or iOS smartphone/tablet by means of a web browser.

Intro (cont.)

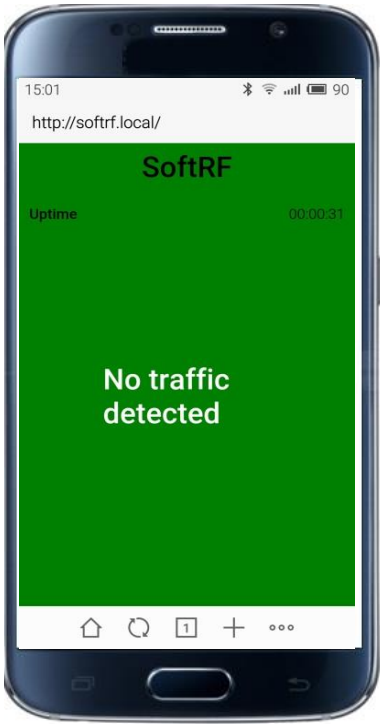


To provide adequate UI and functionality each of SoftRF demo sketches is running auxiliary services in background, such as:

- ❑ Wi-Fi Access point (except Demo4)
  SSID: SoftRF
  Password: 123456123456
- ❑ Web server
  IP: 192.168.4.1
  Hostname: soft.local
  Port: 80
- ❑ Network time sync (NTP) client
- ❑ “Over-the-air” firmware update service

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Demo 1: FLARM detector.



Demo1_Alert_868.ino sketch does implement very basic traffic alert algorithm.

SoftRF is listening to the air for any valid FLARM transmission session.

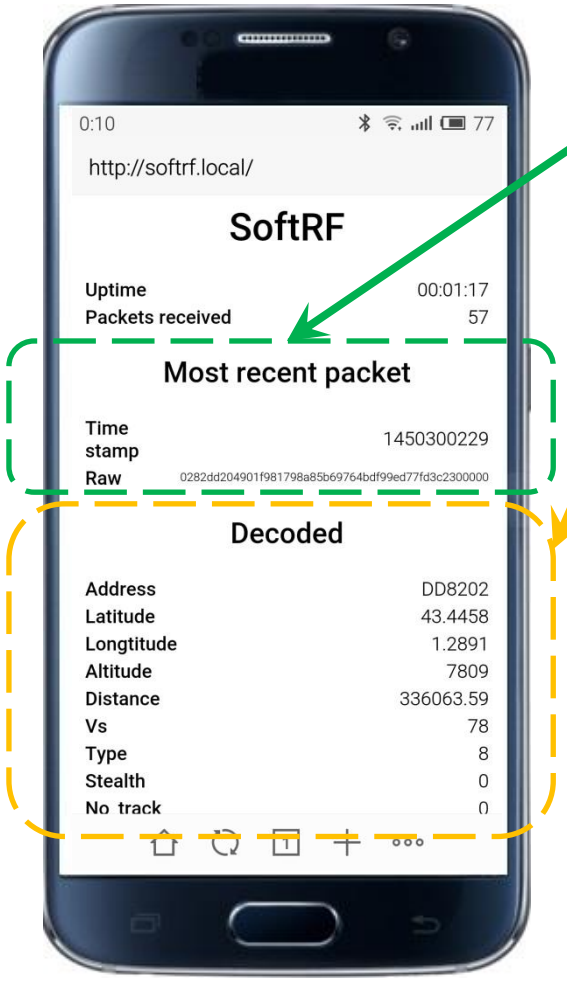
A packet is considered to be "*valid*" if:

1) it has specific values in "address" field ;
2) 24 bytes of payload size ;
3) valid checksum.

Reception of at least one valid FLARM packet will trigger red alarm screen.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Demo 2: Receiver.



Primarily ***Demo2_RX_868.ino*** sketch displays most recently received FLARM packet in plain text hexadecimal format. It also shows time stamp when the packet was captured.

Secondarily the sketch makes an attempt to decode content of the FLARM packet's payload and displays the result.

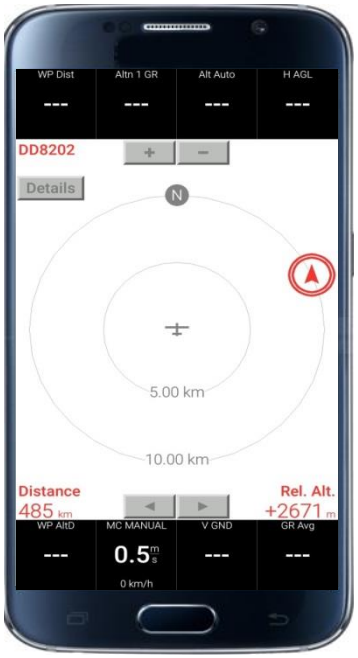
Algorithm of FLARM V4 decryption was revealed by Hiram Yaeger in 2008.
In 2015 unknown author has revealed the algorithm of FLARM V6.
Clayton Smith and Stanislaw Pusep have developed implementation of the decoder for GNU Radio and Linux. Source code of Stanislaw's decoder is in use by this sketch.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Demo 2 (cont.)

In addition to visualization of incoming data in text representation the ***Demo2_RX_868.ino*** sketch does also supply traffic information to external hardware/software clients, namely:

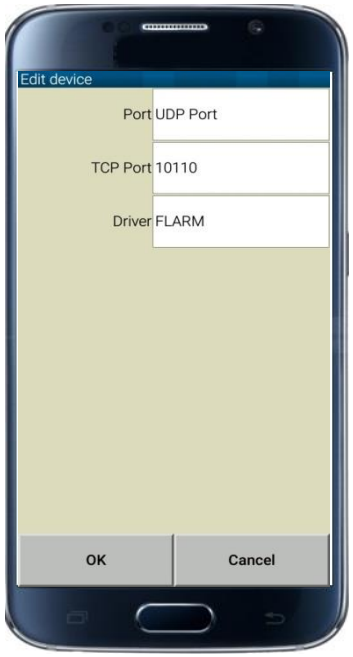
1) Argus ;
2) XCSoar.



IP address/port pairs of these clients are specified in ***Demo2_RX_868.h*** file.

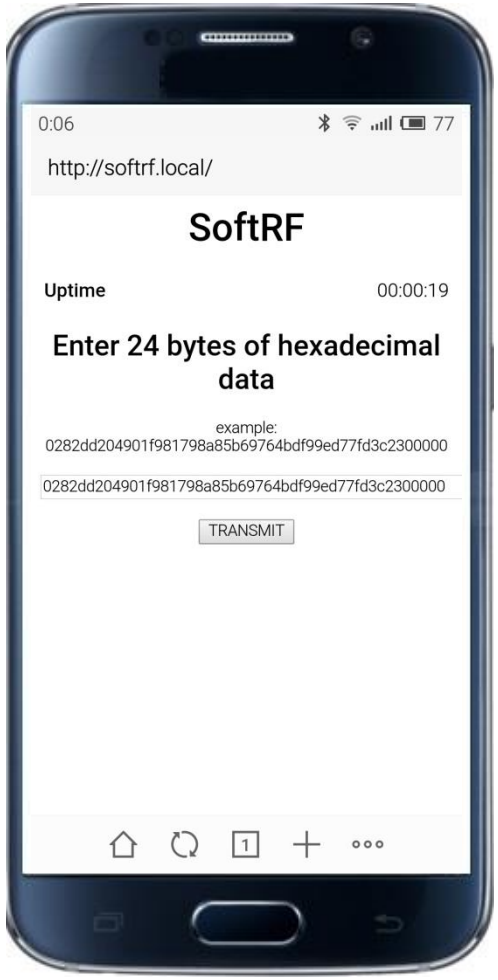
As an example, when an ***XCSoar*** client is configured like this

and "***FLARM View***" window is opened you will be able to see a traffic map



Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Demo 3: Transmitter.



Demo3_TX_868.ino sketch gives an opportunity to emit user-defined data into the radio space on 868.4 MHz.

SoftRF's web page has an input field to fill it with arbitrary hexadecimal data of 24 bytes in size.

In order to submit the data, please, press "TRANSMIT" button.

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

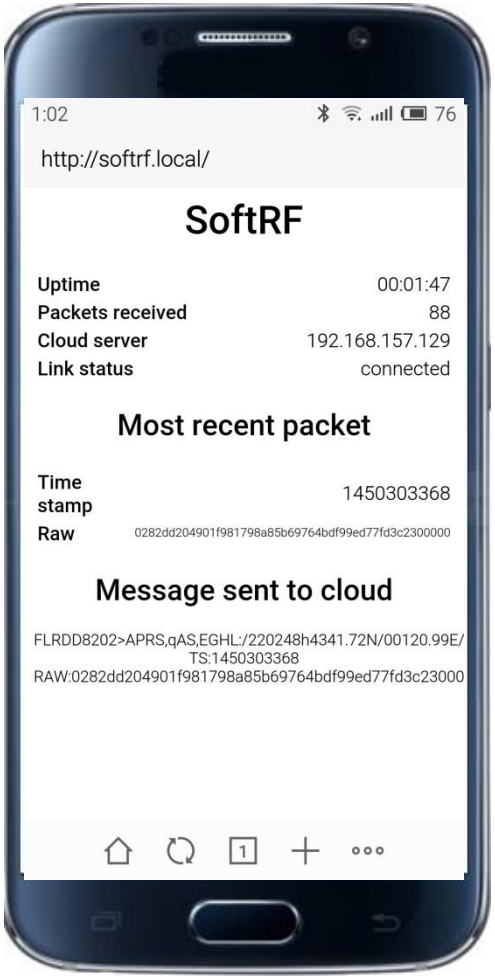
Demo 4: Put into a cloud.

Demo4_Cloud_868.ino sketch is doing simulation of OGN receiver's behavior.

In this case SoftRF is connected to local Wi-Fi network as a client. SSID and password of the network has to be specified in ***Demo4_Cloud_868.h*** file before the sketch is being built:

#define MY_ACCESSPOINT_SSID <Your SSID>
#define MY_ACCESSPOINT_PSK <Your PASSWORD>

This demo open and maintain connection with a cloud server. When a FLARM packet is received it becomes converted into APRS format then sent to the cloud.



Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.

Thank you for your attention!

To visit home page of this project, please, come to:

http://github.com/lyusupov/SoftRF

Linar Yusupov (linar.r.yusupov@gmail.com). December, 2015.