SKYTRAXX FANET+ Module

Juergen Eckert

October 2, 2017

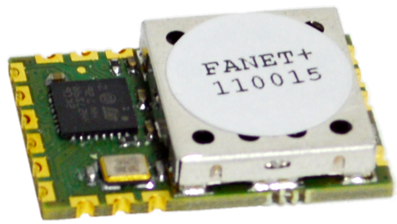

Figure 1: FANET+ Module

Key Product Features

- Physical (PHY) layer (LoRa Modem: SX1272):
  - Up to +20 dBm (100 mW) constant RF output
  - 157 dB maximum link budget
  - 868 MHz and 915 MHz ISM-band compatible
- Medium Access Control (MAC) layer:
  - CSMA/CA
  - Ensures 1 % duty cycle
  - Multi-hop and acknowledgment handling
  - Receiving and transmitting queues
- Application (APP) layer (airborne units only):
  - Tracking broadcast
  - FLARM beacon
- Low RX current of 12 mA
- Small form factor of 15×23×4 mm

1 General Description

Flying Adhoc Network (FANET) is an open communication standard for multi-hop information exchange. It is specially designed to suite the needs of flight instruments in an infrastructure-less 3D environment. Its modular Open Systems Interconnection (OSI) model based approach allows for future extensions as well as the utilization for other related applications like weather stations. LoRa is used for maximizing coverage while keeping the transmission power levels within the allowed ISM band limitations.

Despite of the OSI reference FANET is reduced to three layers: PHY, MAC, and APP. The here described module, depicted in Figure 1, fully handles the former two layers. Optionally, the module can handle the tracking broadcasts which is part of the APP layer.

A brief description of the protocol as well as a reference implementation can be found on GitHub[1]. The remainder of this article purely focuses on the module and its interaction with a host system.

[1] https://github.com/3s1d/fanet-stm32

2 Ordering Options

Skytraxx features two software variants. For exclusive ground usage the open source version will be delivered. For airborne usage FANET gets extended by FLARM. It is than called FANET+. In order to transmit FLARM frames one must use the integrated tracking service. Skytraxx will provide free updates—including FLARM—on an at least annual basis.

3 Characteristics

The sticker on top of the module (see fig. 1) depicts its address in hexadecimal. It is stored in flash memory. The former two digits represent the manufacturer identification number. The remaining four digits are for user identification.

Nota bene: FLARM equipped modules can currently only be supplied using the manufacturer ID: 11_{16}

3.1 Physical Characteristics

Figure 2 shows the pin layout of the module. The antenna connection is optimized for 50 Ω impedance. A high signal during power-on or reset on the pin `BOOT` invokes the STM32 build-in bootloader. The open source module can than be programmed using the STM bootloader protocol. For the airborne module this scheme will fail due to its read-out protection. `RESET` (active low) performs a hard reset. Pins `RXD` and `TXD` are the logic level UART pins for receiving and transmitting, respectively. `DNC` (do not connect) provides pins for future use like USB and I2C.

The recommended footprint is depicted in Figure 3. A suitable Eagle library can be found in the Git repository[2].

3.2 Electrical Characteristics

Tables 1 to 3 show the electrical characteristics. `RESET` and `BOOT` feature an internal 10 kΩ pull-up and pull-down resistor, respectively. Please note that the `PPS` pin is 5V tolerant, however the UART pins are not.

Description	Conditions	Typ.	Unit
Supply Current idle	-	1.2	mA
Supply Current RX	-	12	mA
Supply Current TX	+20 dBm	125	mA
	+10 dBm	25	mA

Table 1: Power Consumption

[2] https://github.com/3s1d/fanet-stm32/blob/master/module/fanet.lbr

Description	Min	Typ.	Max	Unit
Supply Voltage (V_{DD})	1.8	3.3	3.6	V
Temperature	-10	-	55	°C

Table 2: Operating Range

Symbol	Condition	Min	Max	Unit
V_{IL}		-	$0.3 \cdot V_{DD}$	V
V_{IH}		$0.7 \cdot V_{DD}$	-	V
V_{OL}	$\lvert I_O \rvert \leq 8mA$	-	0.4	V
V_{OH}	$\lvert I_O \rvert \leq 8mA$	$V_{DD} - 0.4$	-	V

Table 3: I/O Characteristics

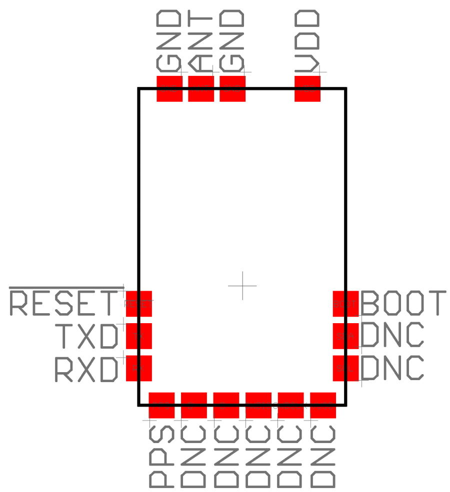

Figure 2: Pin Layout (top view)

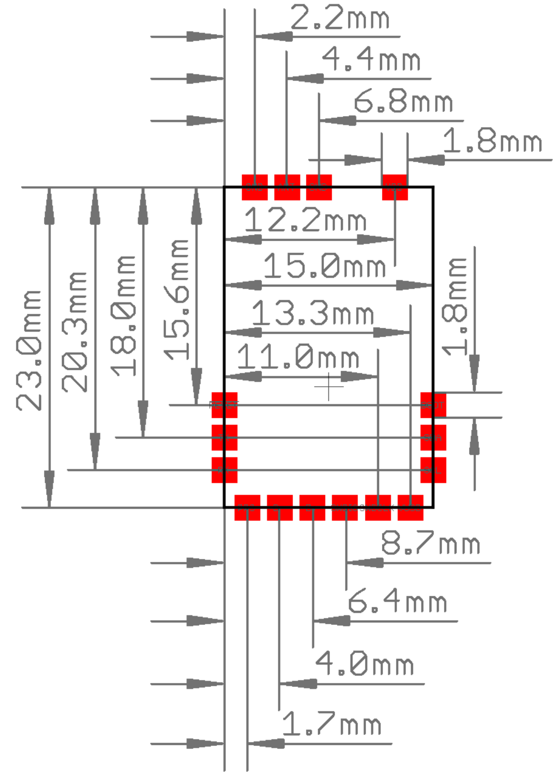

Figure 3: Recommended Footprint (top view)

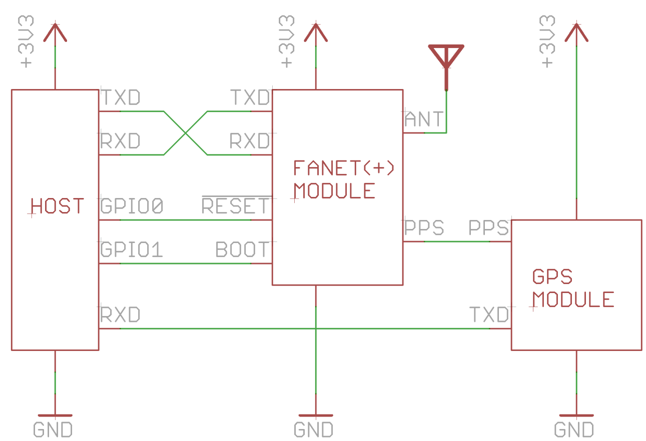

Figure 4: Recommended Connection Diagram

4 Theory of Operation

The module fully handles the PHY and MAC layer. It ensures to comply with the ETSI 1 % duty cycle regulations for the fitting 868 MHz ISM-Band. It features two queuing systems, one for transmitting and one for receiving which enable forwarding and automated acknowledgment generation. A neighbor database is generated to support optimal forwarding strategies. If the state of the instrument is continuously feed into the module it can also automatically generate and broadcast tracking information (APP layer FANET type 1 and FLARM). No packet decoding is implemented on the module. The received header gets decoded and combined with the raw payload gets handed over to the host system for further processing.

4.1 Connection

Figure 4 shows the recommended connection diagram. All the signal levels must be TTL compatible $U \in [0, V_{DD}]$. `PPS` is 5 V tolerant. The absolute minimum is to connect the UART of the FANET module. However, it is highly advised to connect at least the `RESET` pin to a free GPIO pin of the host system in order to be able to always enter a known state. Connecting the PPS signal from the GPS/GNSS module is crucial for the FLARM timings. The modules power consumption may in the future as well benefit from this time synchronization as sleep modes could be entered with a very high precision. For none-airborne modules it is likely that no GPS/GNSS is present, therefore this connection is not mandatory.

4.2 UART Commands

The UART is set to 115200 bits^{-1}, 8 bit data length, no parity bit, and one stop bit (aka. `8N1`). Communication is handled using an ASCII based human readable protocol. In each byte the most significant bit must remain zero.

Each string starting with '#' (23_{16}) and ending with '\n' ($0A_{16}$) is considered to be a valid line and will be evaluated according to the following set of instructions. Generally, communication is based on an request-response scheme were the host application acts as the master.

After the line start indicator the following two chars define what subunit shall be addressed or which subunit is reporting back. The three available types are:

- 'FN' FANET for data exchange
- 'DG' Dongle for module management
- 'FA' FLARM for airborne units only

Always followed by a single subcommand char. In case additional data is supplied a space (20_{16}) must be inserted followed by the actual data values as strings separated by commas ($2C_{16}$). Each line send from the host system will trigger a response from the module.

For more details please see the reference implementation `serial_interface.[h,cpp]`.

4.2.1 Response Lines

The module response format is depicted in Code 1. Unless 'OK' is returned the status data is expected to be a 3-tuple consisting of a general type, followed by a unique status number, and an human understandable string. A list of all possible return codes can be found in the reference implementation.

Code 1: Response Format

```
#[FN,DG,FA]R [OK,ERR,WRN,MSG],num,str\n
```

Other responses are possible in case more data needs to be provided.

4.2.2 State Command (APP)

The command depicted in Code 2 extends the application running on the host system into the module. All the values are given in decimal float fashion. It triggers the transmission of type 1 (tracking) packets on a regular basis. Time, UTC formated in `struct tm` style, and turn rate are optional values. Please note that time information are mandatory for FLARM to operate. Code 3 depicts all possible responses.

Code 2: State (APP layer)

```
#FNS lat(deg),lon(deg),alt(m),
speed(km/h),climb(m/s),heading(deg)
<,year(since 1900),month(0-11),day,
hour,min,sec,turn(deg/s)>\n
```

Code 3: Possible State Responses (APP layer)

```
#FNR OK\n
#FNR ERR,10,no source address\n
#FNR MSG,13,power down\n
```

4.2.3 Config Command (APP)

The command depicted in Code 4 configures the user option for the automated type 1 transmission. Code 5 depicts all possible responses.

Code 4: Config (APP layer)

```
#FNC type(0..7),onlinelogging(0..1)\n
```

Code 5: Possible Config Responses (APP layer)

```
#FNR OK\n
#FNR ERR,12,incompatible type\n
```

4.3 Address Command

The command depicted in Code 6 returns the address for the module in hexadecimal, see Code 7. The value must be globally unique and therefore must not be changed but the user.

Code 6: Address

```
#FNA\n
```

Code 7: Address Response

```
#FNA manufacturer,id\n
```

4.3.1 Transmit Command

The command depicted in Code 8 provides an interface for transmitting general data. Code 9 is the corresponding response. The package payload needs to be encoded on the host side according to the protocol specification. All values will be given in hexadecimal format. The address `00:0000` is considered to be the broadcast address. Each payload byte is formated as 2-byte-chars in hex format. Transmission is done automatically.

Code 8: Transmit

```
#FNT type,dest_manufacturer,dest_id,
forward(0..1),ack_required(0..1),
length,payload(length*2 hex chars)\n
```

Code 9: Possible Transmit Responses

```
#FNR OK\n
#FNR ERR,14,tx buffer full\n
#FNR ERR,10,no source address\n
#FNR MSG,13,power down\n
#FNR ACK,dest_manufacturer,dest_id\n
#FNR NACK,dest_manufacturer,dest_id\n
```

4.3.2 Received Packet

The command depicted in Code 10 will be received from the host system every time the dongle successfully decodes a received FANET packet. All values will be given in hexadecimal format. Each payload byte is formated as 2-byte-chars in hex format.

Code 10: Receive

```
#FNF src_manufacturer,src_id,
broadcast(0..1),signature,type,
length,payload(length*2 hex chars)\n
```

4.3.3 Version Command

The command depicted in Code 11 returns the build version for the module, see Code 12.

Code 11: Version

```
#DGV\n
```

Code 12: Version Response

```
#DGV build-datecode\n
```

4.3.4 Enable Command

The command depicted in Code 13 sets the power mode or returns the current power mode of the module if no additional data is presented, see Code 14. If enabled the receiver chip is turned on.

Code 13: Enable

```
#DGP <powermode(0..1)>\n
```

Code 14: Enable Responses

```
#DGP (0..1)\n
#DGR OK\n
#DGR ERR,70,power switch failed\n
```

4.3.5 Region Command

The command depicted in Code 15 sets the regional characteristics: the frequency and the maximal allowed power level. By setting the output power one must take the antenna gain into account. Please note that the 915 MHz option is currently unavailable. Code 16 depicts all possible responses.

Code 15: Region

```
#DGL freq(868,915),dBm(2..20)\n
```

Code 16: Region Responses

```
#DGR OK\n
#DGR ERR,80,too less parameter\n
#DGR ERR,81,unknown parameter\n
```

4.3.6 Jump to Boot-loader Command

The command depicted in Code 15 enables the firmware to directly jump to its boot-loaders. For the open source version the default STM built-in program is used ('`BLstm`'). For FANET+ devices a boot-loader which supports decryption is used ('`BLxld`'). No return answer is provided upon success. On error Code 18 is returned.

Please note that the recommended procedure for entering the bootload is to use GPIO pins connected to '`RESET`' and '`BOOT`' pins of the module (the later one is only required in case of the open source option).

Code 17: Jump

```
#DGJ BL[stm,xld]\n
```

Code 18: Jump Responses

```
#DGR ERR,61,unknown jump point\n
```

4.3.7 Enable Command (FLARM)

The command depicted in Code 19 sets the power mode for the FLARM submodule or returns its current power mode if no additional data is presented, see Code 20. If enabled the receiver chip is turned on.

Code 19: Enable (FLARM)

```
#FAP <powermode(0..1)>\n
```

Code 20: Enable Responses (FLARM)

```
#FAP (0..1)\n
#FAP OK\n
```

4.3.8 Expiration Command (FLARM)

The command depicted in Code 21 returns the expiration date of FLARM submodule in Code 22. The user must perform an update prior to that date to keep the FLRAM submodule functional. The date is formated in `struct tm` style. If FLARM is used past this date an error message will be received once, see Code 23.

Code 21: Expiration (FLARM)

```
#FAX\n
```

Code 22: Expiration Responses (FLARM)

```
#FAX year(since 1900),month(0-11),day\n
```

Code 23: Expired (FLARM)

```
#FAR ERR,91,FLARM expired\n
```

4.4 Example

Upon power-on or reset the module enters the idle mode. The receiver is in sleep mode and needs to be enabled for FANET and FLARM separately. Code 24 shows an exemplary boot-up sequence. `->` depicts bytes that get received by the module. `<-` depicts bytes that get transmitted by the module. `%` indicates a comment.

Code 24: Example Data Flow

```
% FANET+ custom bootloader (xmodem)
% wait 10sec or terminate with \n
<- C
-> \n
```

```
% In case of an hardware fault
% an error will occure
<- #FNR MSG,1,initialized\n

% Check for correct version
% if not perform an update
-> #DGV\n
<- #DGV build-201709261354\n

% Get module addr
-> #FNA\n
<- #FNA 11,003F\n

% Check FLARM expiration
% Expires on 31.January 2018
-> #FAX\n
<- #FAX 118,0,31

% Configure APP
% PG, online tracking
-> #FNC 1,1\n
<- #FNC OK\n

% Enable receiver
-> #DGP 1\n
<- #DGR OK\n
-> #FAP 1\n
<- #FAP OK\n

%%%%

% Update state in a loop
% once a second
-> #FNS 45.1234,10.5678,500,37,-1.5,45,
   117,9,30,12,35,2\n
<- #FNS OK\n

% Module received a packet
<- #FNF 11,2E,1,0,1,B,
   7963469EC507369100002\n
```

4.5 FLARM (airborne modules only)

All airborne FANET modules manufactured by SKYTRAXX get equipped with FLARM. The following limitations must be followed:

- FLARM is in beacon/passive mode only; no FLARM packet will get received.
- Aircraft types are limited to paraglider (1) and hangglider (2), otherwise FLARM will get disabled.
- Annual (free) updates are required.
- Custom boot-loader to upload the firmware, see section 5.

The FLARM and the FANET address will be kept in sync with each other. The most significant byte of the FLARM address correspond to the manufacturer ID of FANET. The middle and least significant bytes of the FLARM address correspond to the user ID of FANET.

FLARM requires the PPS pulse to arrive within ± 5 ms of the true second start. The status command must be received within the next 300 ms.

5 Firmware Update

To determine whether an update must be performed the version of the current module must be check using the command explained in section 4.3.3.

5.1 Ground/Base Module

The build version of the available binary file can be checked by performing a global search for the regular expression 'build-' (`'strings file.bin | grep build-'`). If both versions differ the new firmware can be flashed using the well known STM32 boot-loader protocol. The boot-loader can be entered using the `RESET` and `BOOT` pin (recommended) or by utilizing the command introduced in section 4.3.6.

5.2 Airborne Module

The version of the firmware file (`fanet.xlb`) can be found at position 24. It is for formated as a string. The upload is done using the custom boot-loader. It supports the xmodem protocol for 128 B and 1 kB chunks. It is entered automatically upon reset or by utilizing the command introduced in section 4.3.6.

An example upload using python with the pyserial and xmodem libraries is depicted in Code 25.

Code 25: Example Firmware Upload

```
import serial
from xmodem import XMODEM

ser = serial.Serial(port=<port>,
        baudrate=115200)
#assuming reset is connected to RTS
ser.setRTS(True)
time.sleep(0.1)
ser.setRTS(False)

def getc(size, timeout=1):
    return ser.read(size) or None
def putc(data, timeout=1):
    return ser.write(data)
print('Uploading...')
modem = XMODEM(getc, putc)
fwstream = open('fanet.xlb', 'rb')
if not modem.send(fwstream):
    print('failed')
else:
    print('OK')

fwstream.close()
ser.close()
```