A LoRa KISS TNC for APRS

M.T. Konstapel

2022-05-18

Abstract

The traditional FSK based 1200bd APRS used in conjunction with analogue FM transceivers is still going strong, but APRS over LoRa is gaining in popularity. Unfortunately, most existing modems are stand alone trackers and iGates. These are convenient as they do not need any external hardware, but integrating these modems in existing digipeater-setups is a challenge. That is why I decided to design a LoRa modem with a standardized KISS interface over USB. This modem can, for example, be connected to a computer running APRX on Linux (eg. an existing 2 meter APRS digipeater). Simply attach the modem with *kissattach* to the Linux AX.25 stack and you now have a cross band APRS digipeater without any hassle.

1 The design

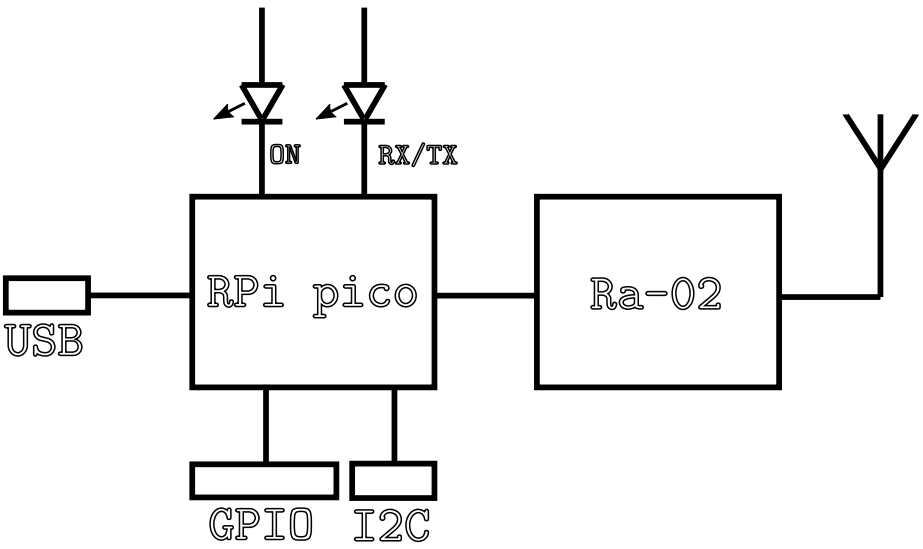


Figure 1: block diagram of the modem

Figure 1 shows the block diagram of the modem. It is basically two blobs attached to each other. The controller is a Raspberry Pi Pico and the radio is a 433MHz Ra-02 module, which is a LoRa module based around an SX1278 chip.

A couple of LEDs provide some status indication, like POWER ON, RX and TX. The I2C-interface as well as six GPIO-pins are available on two headers. These are not used by the default firmware, but could be utilized by altering this firmware yourself.

2 The firmware

Figure 2: block diagram of firmware

2.1 LoRa APRS

A typical LoRa APRS frame is just a string of ASCII characters. And that's a bit of a problem as this is not quite what the original AX.25 standard[1] describes as a valid AX.25 frame. The firmware has to convert the APRS string to a proper AX.25 frame.

2.2 AX.25 APRS

The conversion, of course, goes both ways. From LoRa APRS to AX.25 and from AX.25 to LoRa APRS. Before converting from LoRa APRS to AX.25 some error checking is done as frames could be corrupted in transit.

2.3 KISS

Kiss[2] is stupidly simple: send two bytes (0xC0 0x00), than just send the AX.25 frame and end with one byte: 0xC0. The AX.25 frame has to be escaped, because any 0xC0 value in it is seen as an end of frame.

2.4 USB

The KISS frames are send to the host over the USB interface. The host can simply attach the interface to the AX.25 stack via *kissattach* or attach it directly to APRX via the APRX configuration file.

3 Configuration

The modem can be configured via a serial terminal program like *minicom*. Exit from KISS mode with command *kissparm -p ax0 -x* (assuming the modem is attached to ax0) and kill the kissattach process. Than start minicom (19200bd 8N1). When finished you can go back to KISS mode with command *kiss 1* or you can power cycle the modem.

[1] http://www.ax25.net/AX25.2.2-Jul%2098-2.pdf
[2] https://en.wikipedia.org/wiki/KISS_(TNC)

All commands that alter settings write to RAM. After a reboot, these settings are lost, unless the settings are saved to FLASH.

System commands		
command	**parameter**	**description**
read	ram	read settings from RAM
	flash	read settings from FLASH
save	-	Save RAM settings to FLASH
kiss	1	Switch to KISS mode

LoRa commands		
command	**parameter**	**description**
freq	Herz	Set lora frequency (limited between 420MHz and 450MHz)
spread	value	Set lora spreading factor (can be between 6 and 12)
pre	value	Set lora preamble (can be between 6 and 65535)
rate	value	Set lora coding rate (can be between 5 and 8)
power	dBm	Set lora tx power (can be between 2 and 17)
pa	0/1	Set PA on or off
band	value	Set lora bandwidth (can be between 7800 and 500000)
restart	-	Restart radio (do this after altering LoRa settings)

Stand alone commands (not used in KISS mode)		
command	**parameter**	**description**
mycall	CALL-x	Set call and ssid of modem
servercall	CALL-x	Set call and ssid of server
path 1	PATH	Set first path or leave blank
path 2	PATH	Set second path or leave blank
dest	ID	Set destination

Table 1: request packet

4 Practical notes

4.1 Build the pcb

Building the printed circuit board is straight forward. The design files are made with KiCad 5.1.8 and scaled PDF files of the printed circuit board are available. The pcb has two layers and can be made by yourself or ordered from a pcb manufacturer. When etched yourself, there are no plated through vias. All ground connections, as well as some other connections, should be soldered on both sides of the pcb.

A Full schematic

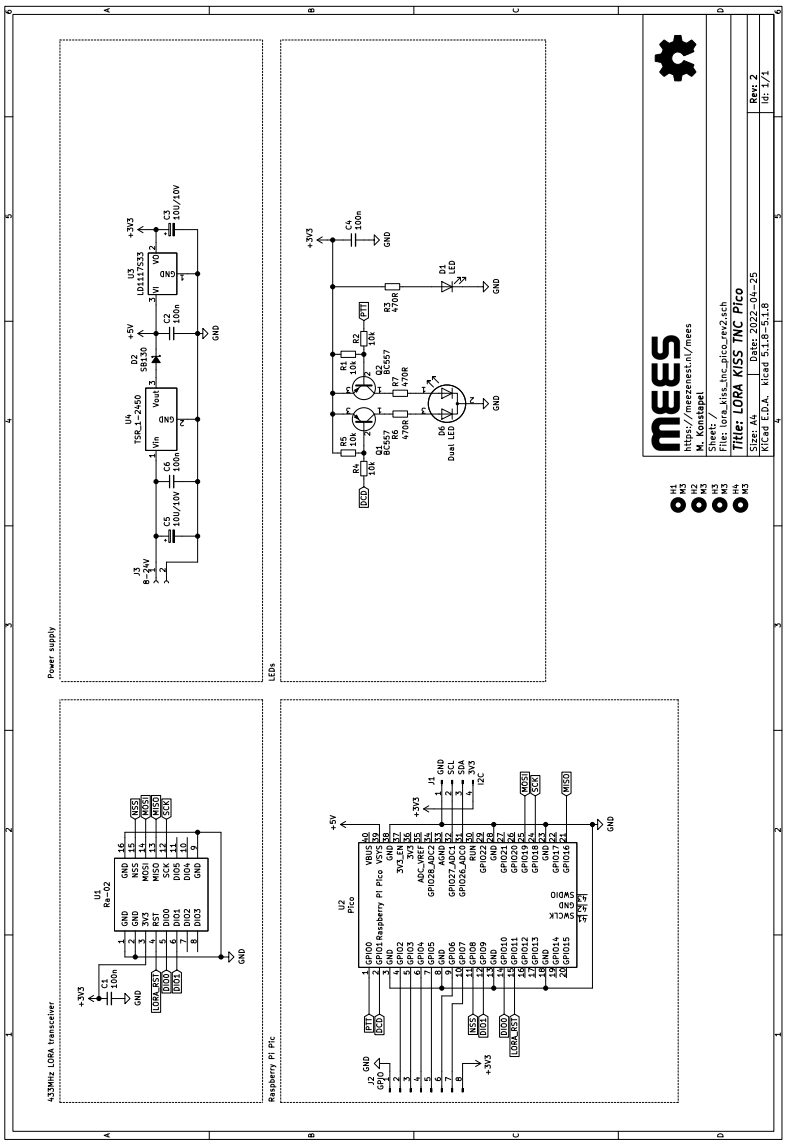


Figure 3: full schematic of the modem

B Component placement

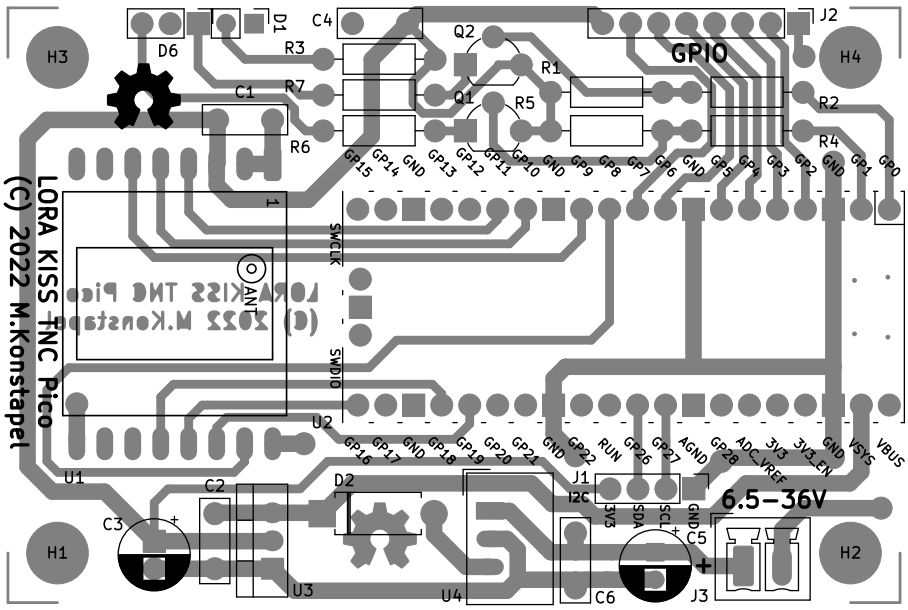


Figure 4: component placement

C Component placement 3D view

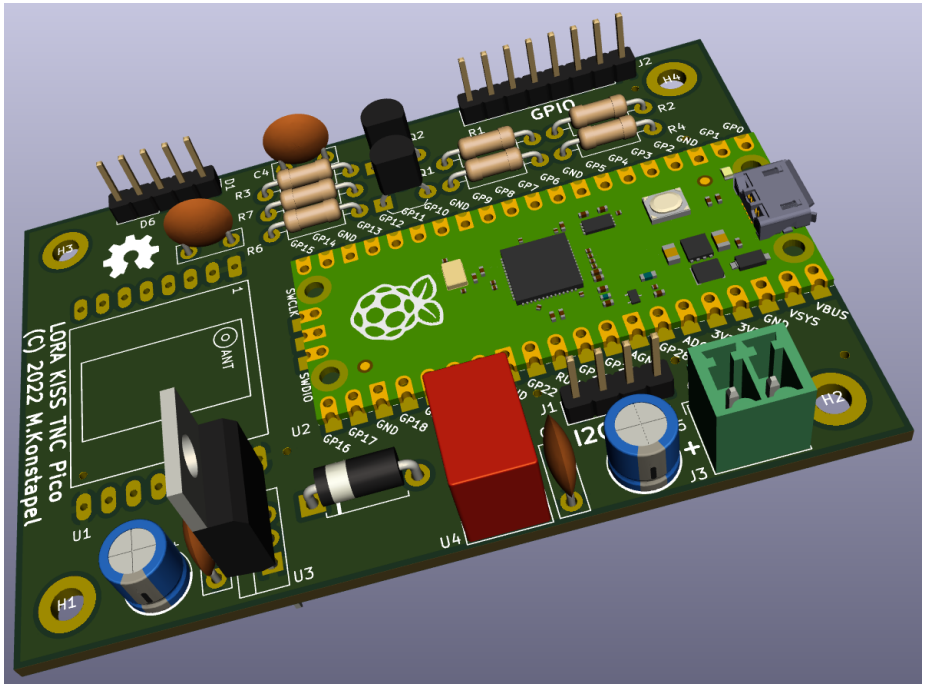


Figure 5: 3d view

D Bill of material

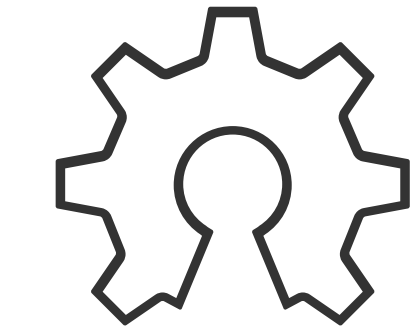
Quantity	Reference	Value	Manufacturer	Ordering #	Conrad #
4	C1 C2 C4 C6	100n	QPL	100n/50V 5mm	
2	C3 C5	10U/10V	QPL	10U/25V 2.5mm	
1	D1	LED	QPL	Pin header 1x2 2.54mm	
1	D2	SB130	Vishay	SB130	164828 - 62
1	D6	Dual LED	QPL	Pin header 1x3 2.54mm	
4	H1 H2 H3 H4	M3	Not a placeable part	-	
1	J1	I2C	QPL	Pin header 1x4 2.54mm	
1	J2	GPIO	QPL	Pin header 1x8 2.54mm	
1	J3	8-24V	Degson	DG381-3.5-02P-14-00AH-1	1327243 - 62
2	Q1 Q2	BC557	QPL	BC557	
4	R1 R2 R4 R5	10k	QPL	10k 1% 0.25W	
3	R3 R6 R7	470R	QPL	470R 1% 0.25W	
1	U1	Ra-02	Thinker	Ra-02	
1	U2	Pico	Raspberry Pi	Pico	
1	U3	LD1117S33	STMicroelectronics	LD1117V33	147028 - 62
1	U4	TSR_1-2450	TracoPower	TSR 1-2450	156673 - 62

Figure 6: bill of material

E Open source hardware and software

All the design files are available on my website: https://www.meezenest.nl/mees

This document is published under the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license.



open source hardware

Free as in Freedom