Custom USB Device for PTT Control

Dylon Mutz, KK6OTK
dylon.mutz@gmail.com

Introduction

What was the goal?

- Wanted to try out APRS (Automatic Packet Reporting System)
  - Developed by Bob Bruninga, WB4APR
  - Packets sent on the VHF and HF bands
  - Used to transmit position reports, weather reports, and other kinds of data
  - Every-day positional reports, Public Service, and Emergency communications

- Experiment with digital modes

- Also wanted to use with other computer-related software and control
  - GNUradio
  - Xastir
  - Direwolf

Background

What were the options?

- TNC (terminal node controller)
- USB PC linker Adapter (~$73)
- SignalLink USB ($130)
- Mini-DIN 6-pin Packet cable ($50-$60)
  - Some of these cables do not provide PTT functionality
- Rigblaster Plug & Play ($120)

What option was chosen?

- All of the above were good options
- Personally liked the Rigblaster Plug & Play for being small and simple
- But after doing some research, I felt like I could make this myself
- Also more fun to learn and build something

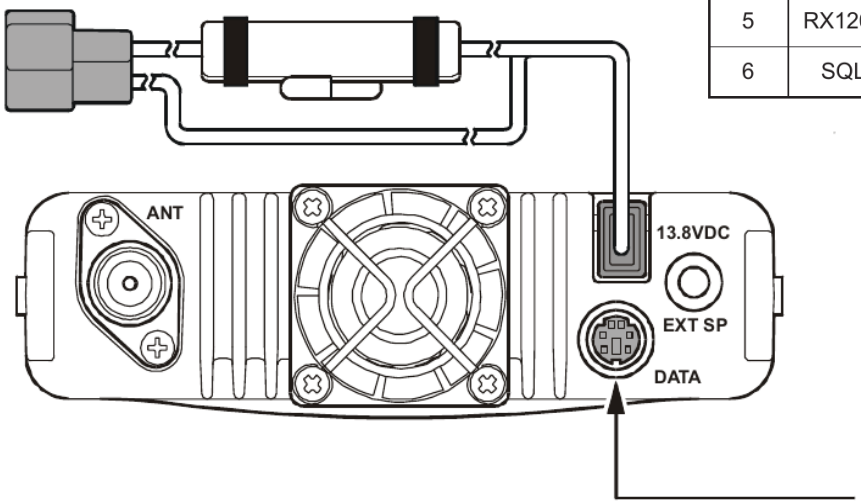
Operation

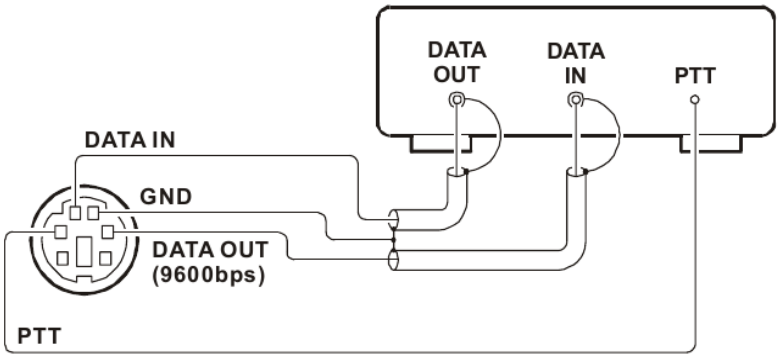
Yaesu FT-8900R

- Data I/O pins can be connected to computer's sound card through audio jacks
- Activating the PTT requires pulling the pin to ground

DATA Jack Pin Out

Pin	Label	Note	CT-39A Wire Color
1	PKD (DATA IN)	Packet Data Input *Impedance*: 10 kΩ, *Maximum Input Level*: 40 mV p-p for 1200 bps 2.0 Vp-p for 9600 bps	Brown
2	GND	Signal Ground	Red
3	PTT	Gound to Transmit	Orange
4	RX9600	9600 bps Packet Data Output *Impedance*: 10 kΩ, *Maximum Output*: 500 mV p-p	Yellow
5	RX1200	1200 bps Packet Data Output *Impedance*: 10 kΩ, *Maximum Output*: 300 mV p-p	Green
6	SQL	Squelch Control *Squelch Open*: +5 V, *Squelch Close*: 0 V	Blue





Operation

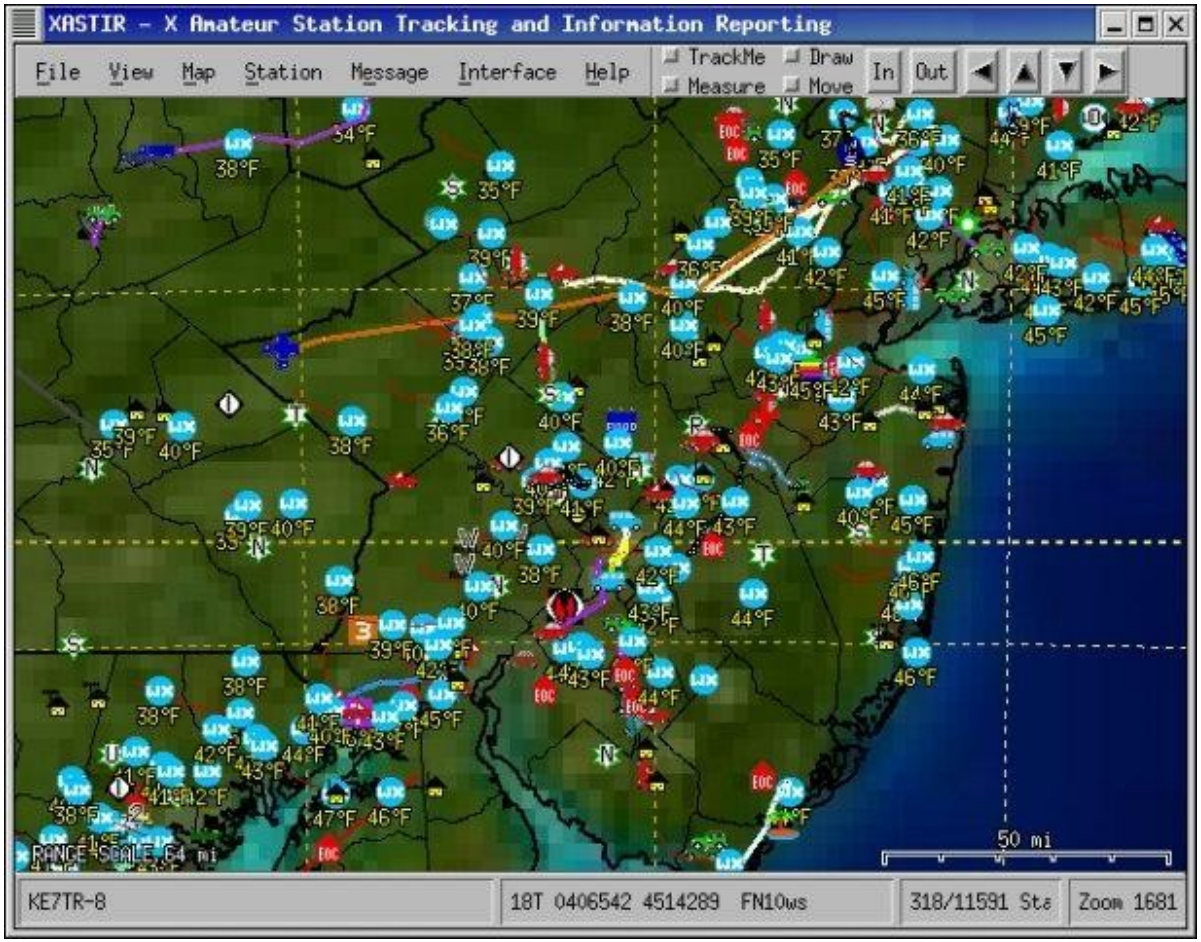
Direwolf

- First of two main programs I want to use for APRS
- Direwolf acts as a software TNC using soundcard and serial port
- Sends/receives packets through mic and speaker ports
- Activates transmitter through serial port (or USB to RS232 adapter)

Operation

Xastir

- Second of two main programs I want to use for APRS
- Xastir provides object mapping and interfaces with Direwolf to send/receive messages

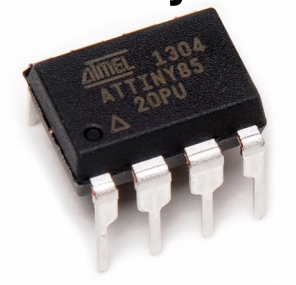


USB Interface

USB interface

- Problem: Need an IC that can send/receive USB packets and has a controllable output pin
- Solution: AVR ATtiny85 + V-USB library
  - ATtiny85 is an 8-pin programmable microcontroller (µC) with 6 I/O pins
  - V-USB is a software library that allows AVR chips to be used as a USB device

ATtiny85



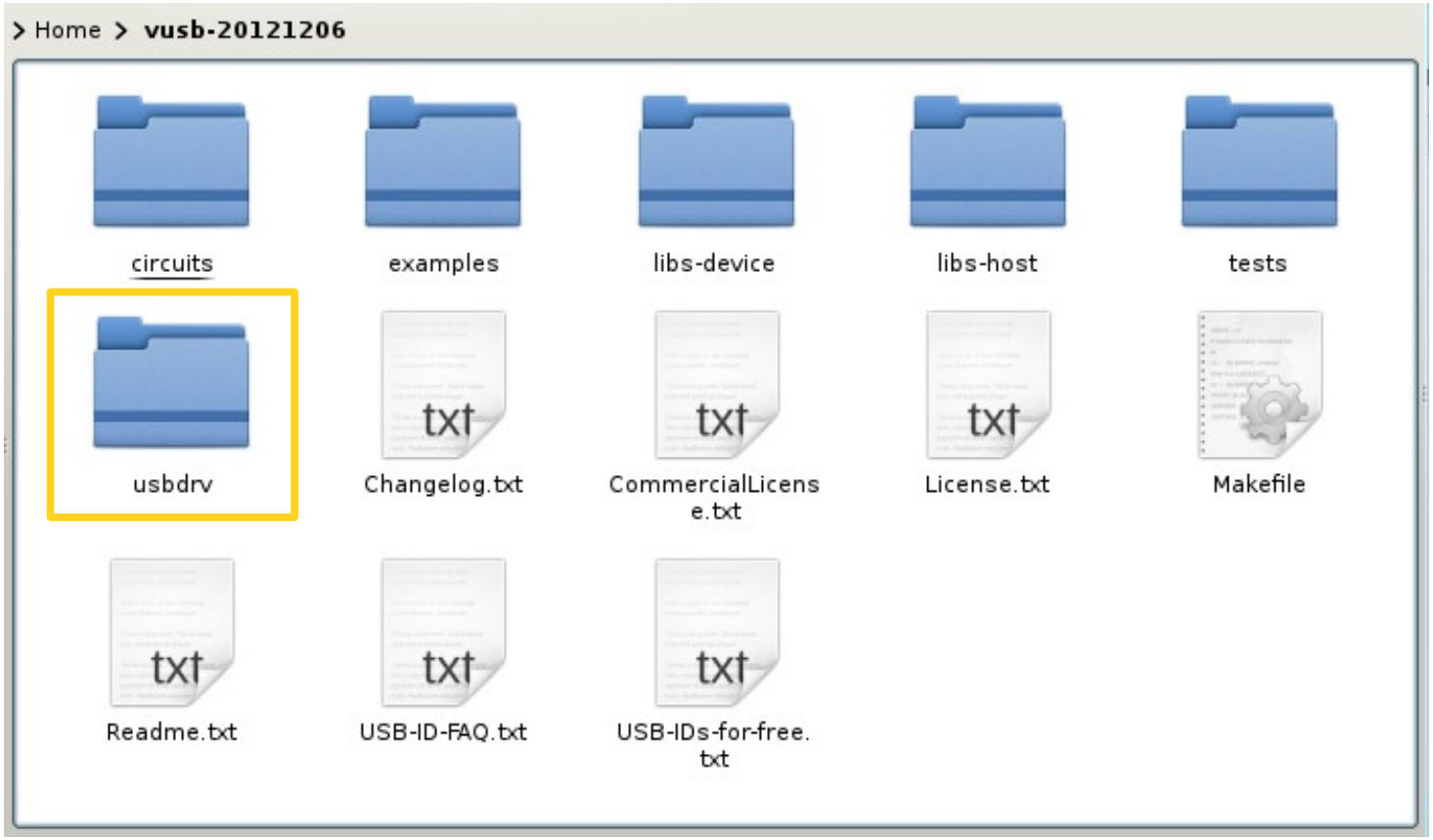
+

V-USB

USB Interface

V-USB Setup

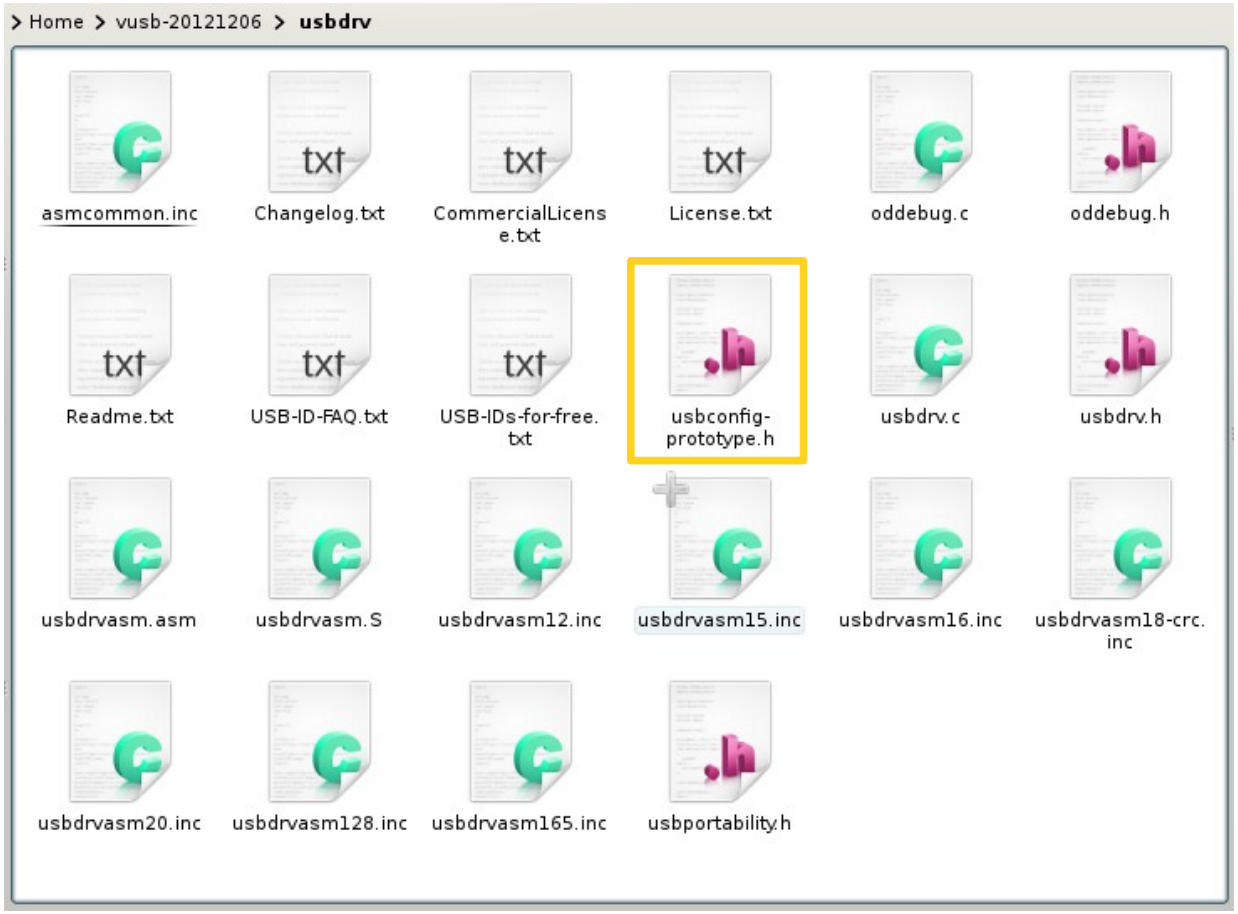
- The latest V-USB package can be downloaded from https://www.obdev.at/vusb/
- Contents include very useful documentation
- Main folder is “usbdrv” which contains the V-USB firmware



USB Interface

V-USB Setup

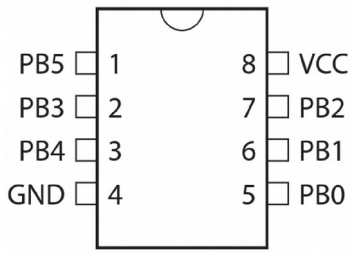
- The “usbconfig-prototype.h” contains settings for USB configuration
  - Only a few need to be set to get up and running



USB Interface

V-USB Setup

- ATtiny85 pinout shows all I/O pins are labeled 'PBx'
  - 'B' is the port name
  - 'x' is the number of the pin



Pinout ATtiny85

usbconf.h

```
24 /* ---------------------------- Hardware Config ---------------------------- */
25
26 #define USB_CFG_IOPORTNAME      B
27 /* This is the port where the USB bus is connected. When you configure it to
28  * "B", the registers PORTB, PINB and DDRB will be used.
29  */
30 #define USB_CFG_DMINUS_BIT      1
31 /* This is the bit number in USB_CFG_IOPORT where the USB D- line is connected.
32  * This may be any bit in the port.
33  */
34 #define USB_CFG_DPLUS_BIT       2
35 /* This is the bit number in USB_CFG_IOPORT where the USB D+ line is connected.
36  * This may be any bit in the port. Please note that D+ must also be connected
37  * to interrupt pin INT0! [You can also use other interrupts, see section
38  * "Optional MCU Description" below, or you can connect D- to the interrupt, as
39  * it is required if you use the USB_COUNT_SOF feature. If you use D- for the
40  * interrupt, the USB interrupt will also be triggered at Start-Of-Frame
41  * markers every millisecond.]
42  */
```

USB Interface

V-USB Setup

- Vendor name and device name can be set
- Many other options can also be configured in this file for more advanced devices

usbconf.h

```
245 #define USB_CFG_VENDOR_NAME     'K', 'K', '6', 'O', 'T', 'K'
246 #define USB_CFG_VENDOR_NAME_LEN 6
247 /* These two values define the vendor name returned by the USB device. The name
248  * must be given as a list of characters under single quotes. The characters
249  * are interpreted as Unicode (UTF-16) entities.
250  * If you don't want a vendor name string, undefine these macros.
251  * ALWAYS define a vendor name containing your Internet domain name if you use
252  * obdev's free shared VID/PID pair. See the file USB-IDs-for-free.txt for
253  * details.
254  */
255 #define USB_CFG_DEVICE_NAME     'U', 'S', 'B', '_', 'P', 'T', 'T'
256 #define USB_CFG_DEVICE_NAME_LEN 7
257 /* Same as above for the device name. If you don't want a device name, undefine
258  * the macros. See the file USB-IDs-for-free.txt before you assign a name if
259  * you use a shared VID/PID.
260  */
```

USB Interface

V-USB Setup

- Main program also has to be written in main.c
- Section includes any initial setup and executes USB operation
- Lines 19 & 20 define commands that will be received via USB
- Line 21 defines an output pin, PB3, that is connected to an LED

main.c

```
19 #define USB_LED_OFF 0
20 #define USB_LED_ON 1
21 #define LED_PIN (1<<PB3)
22
23 //Main program
24 int main() {
25     uchar i;
26
27     DDRB = LED_PIN;                  //Setup LED pin as output
28
29     wdt_enable(WDTO_1S);             //Enable 1s watchdog timer
30
31     usbInit();                       //Initialize V-USB library
32
33     //Enforce re-enumeration
34     //Watchdog automatically resets chip after 1 second if freeze occurs
35     usbDeviceDisconnect();
36     for(i = 0; i<250; i++) {         //Wait 500 ms
37         wdt_reset();                 //Keep the watchdog happy
38         _delay_ms(2);
39     }
40     usbDeviceConnect();
41
42     sei();                           //Enable interrupts after re-enumeration
43
44     while(1) {
45         wdt_reset();                 //Keep the watchdog happy
46         usbPoll();
47     }
48
49     return 0;
50 }
```

USB Interface

V-USB Setup

- This function is automatically called whenever the µC receives a USB command
- The command is passed as data into the function, and a case statement can be used to execute the proper function

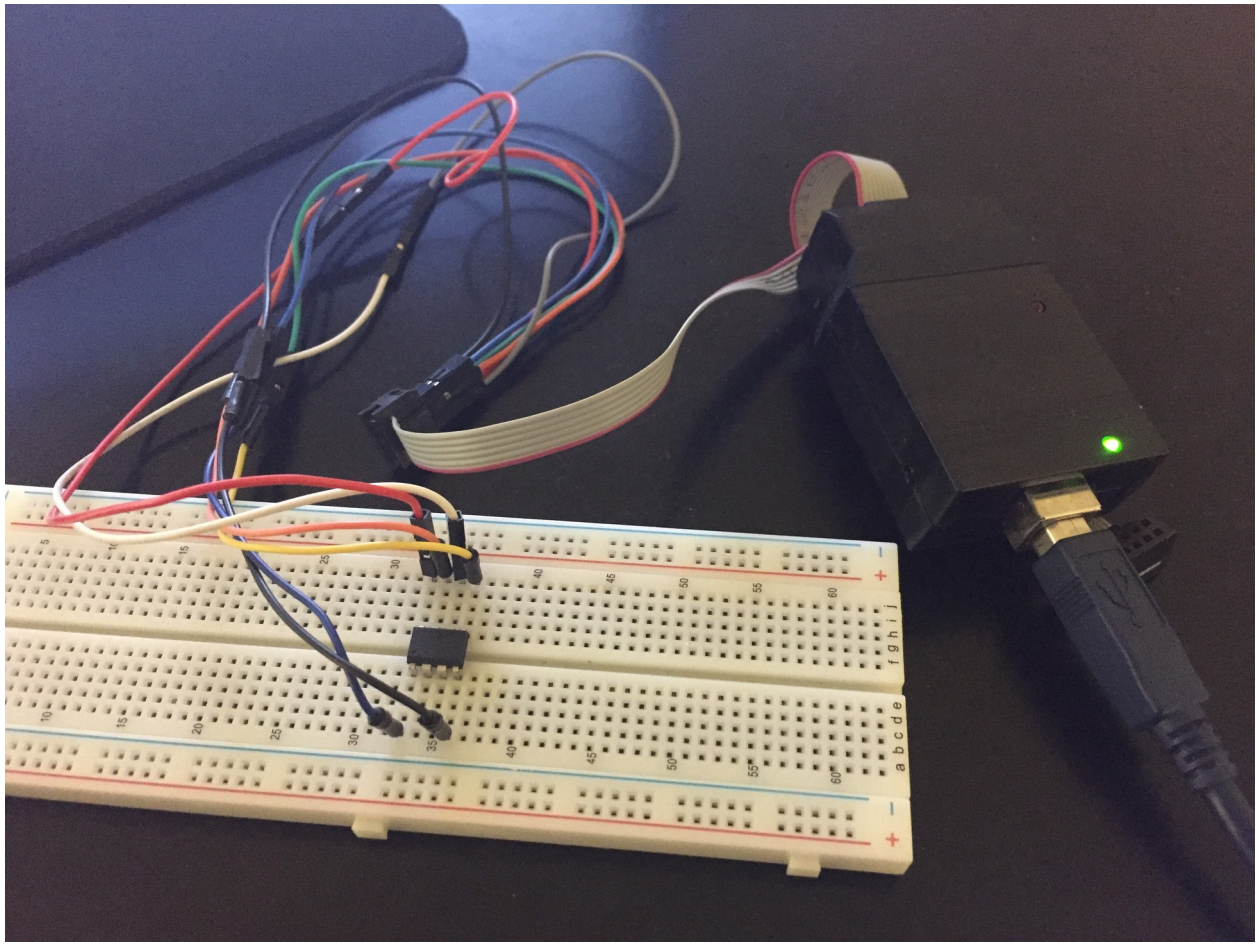
main.c

```
57 // this gets called when custom control message is received
58 USB_PUBLIC uchar usbFunctionSetup(uchar data[8]) {
59     usbRequest_t *rq = (void *)data; // cast data to correct type
60
61     switch(rq->bRequest) { // custom command is in the bRequest field
62     case USB_LED_ON:
63         PORTB |= LED_PIN; // turn LED on
64         return 0;
65     case USB_LED_OFF:
66         PORTB &= ~LED_PIN; // turn LED off
67         return 0;
68     }
69
70     return 1; // should not get here
71 }
```

USB Interface

Programming the µC

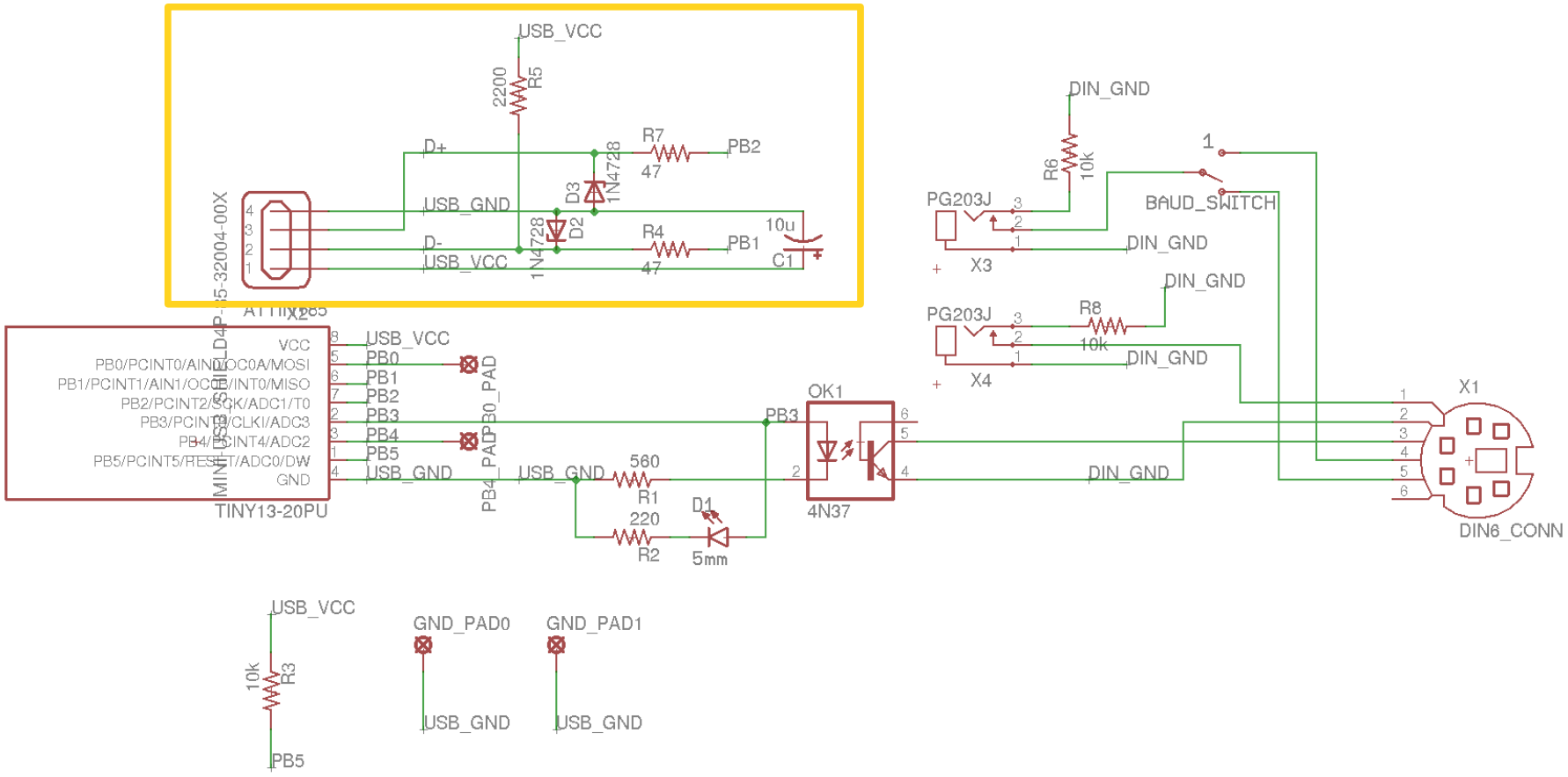
- With main.c and usbconfig.h set up, the µC can be programmed
- USBtinyISP, an AVR programmer, was needed to program the chip
- After programming, the rest of the circuit can be designed around it



Circuit Design

USB Connector

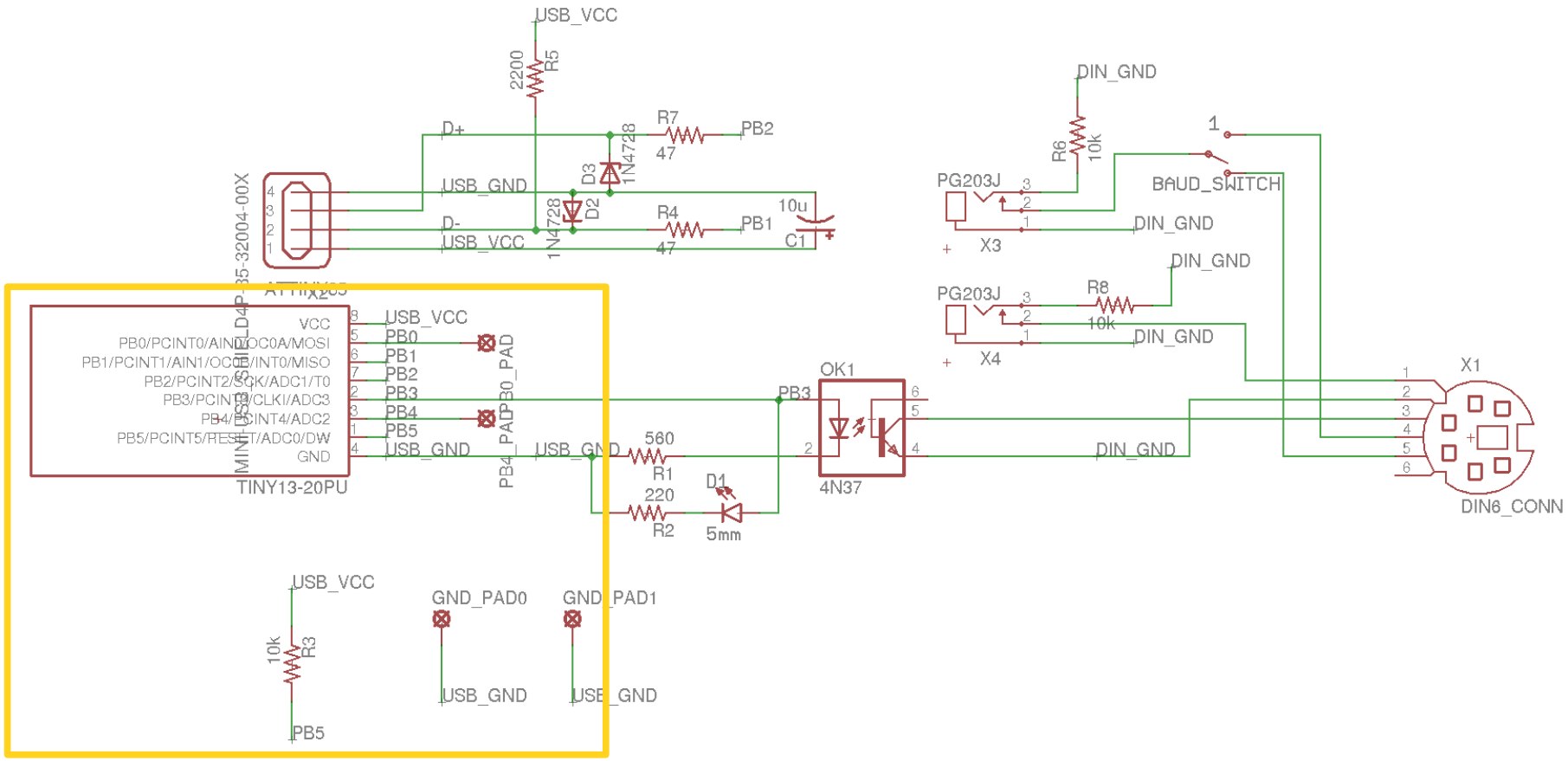
- Standard USB supports 5V power and ground and 3.3V data lines
- Flyback voltage may develop when unplugging USB cables
  - Min 1uF to max 10uF capacitor across Vbus/GND to prevent damage



Circuit Design

ATtiny85 Microcontroller (µC)

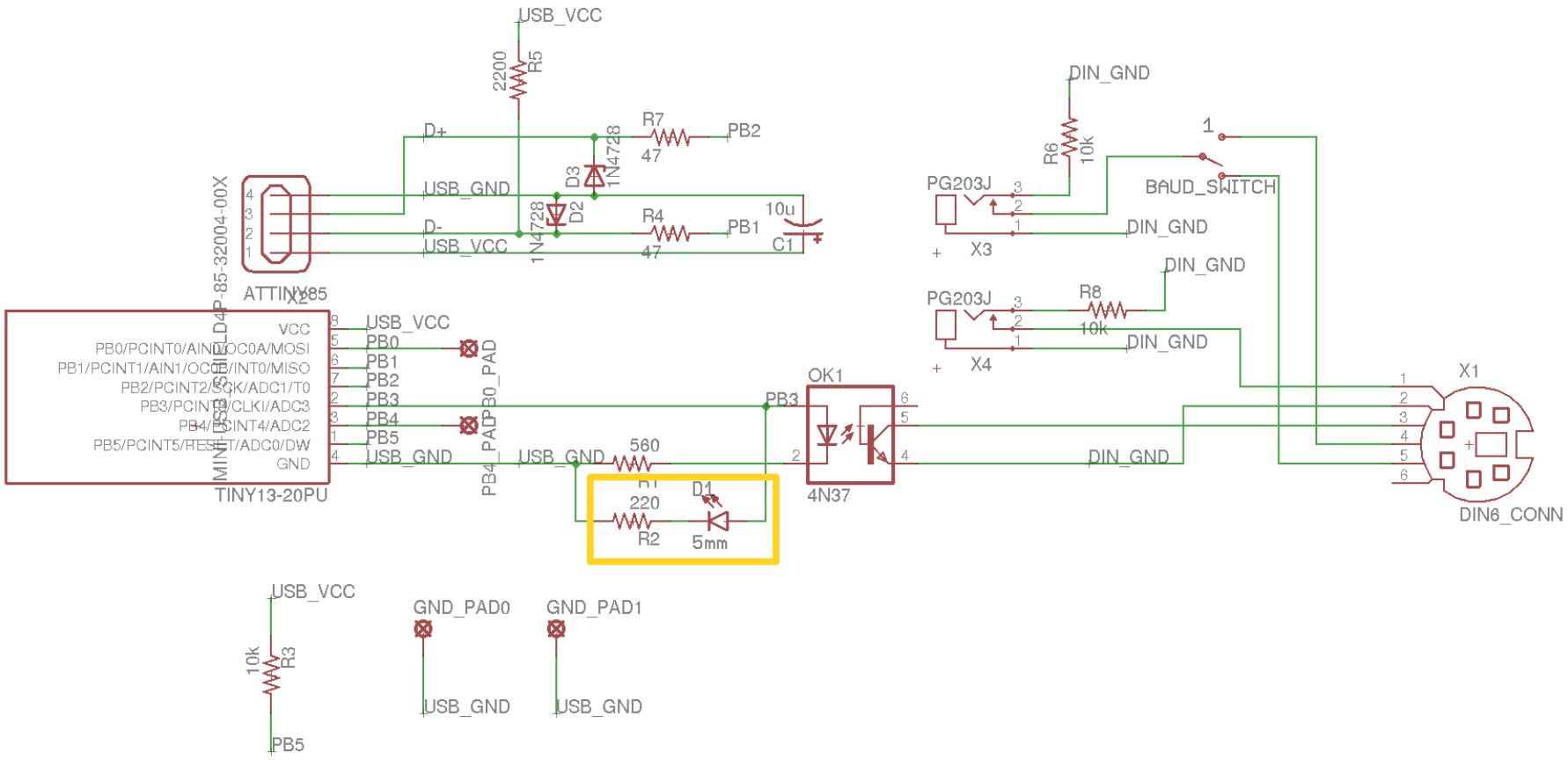
- Communicates with computer through USB
- Interprets commands and controls output pins
- Powered by 5V USB Vbus
- 3.6V zener diodes to regulate µC 5V I/O pins for data line use



Circuit Design

Visual LED

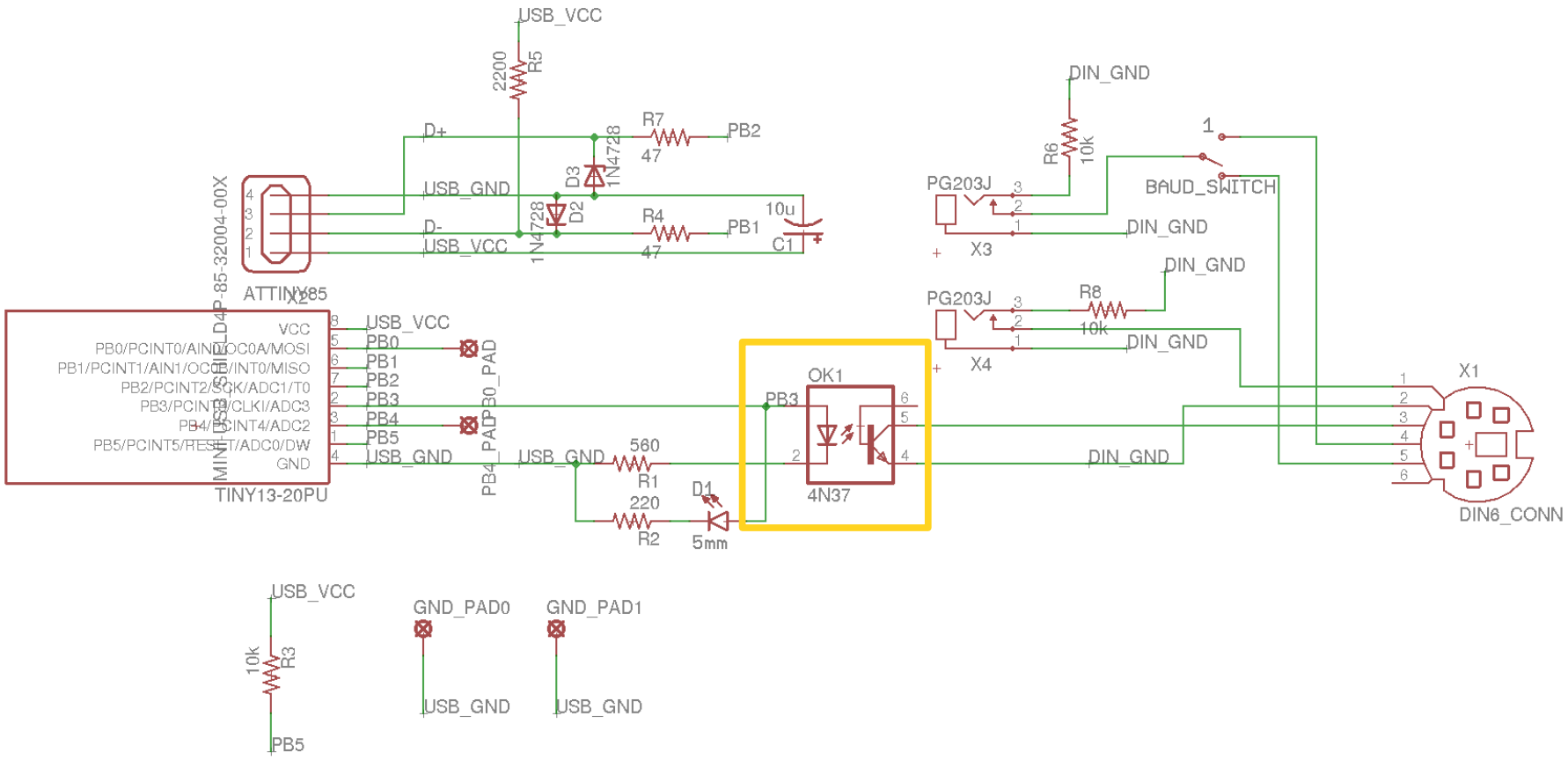
- µC turns the LED on via an output pin
- Just a way to let operator know when the radio is transmitting
- Same output pin also connected to optoisolator (next slide)



Circuit Design

Optoisolator

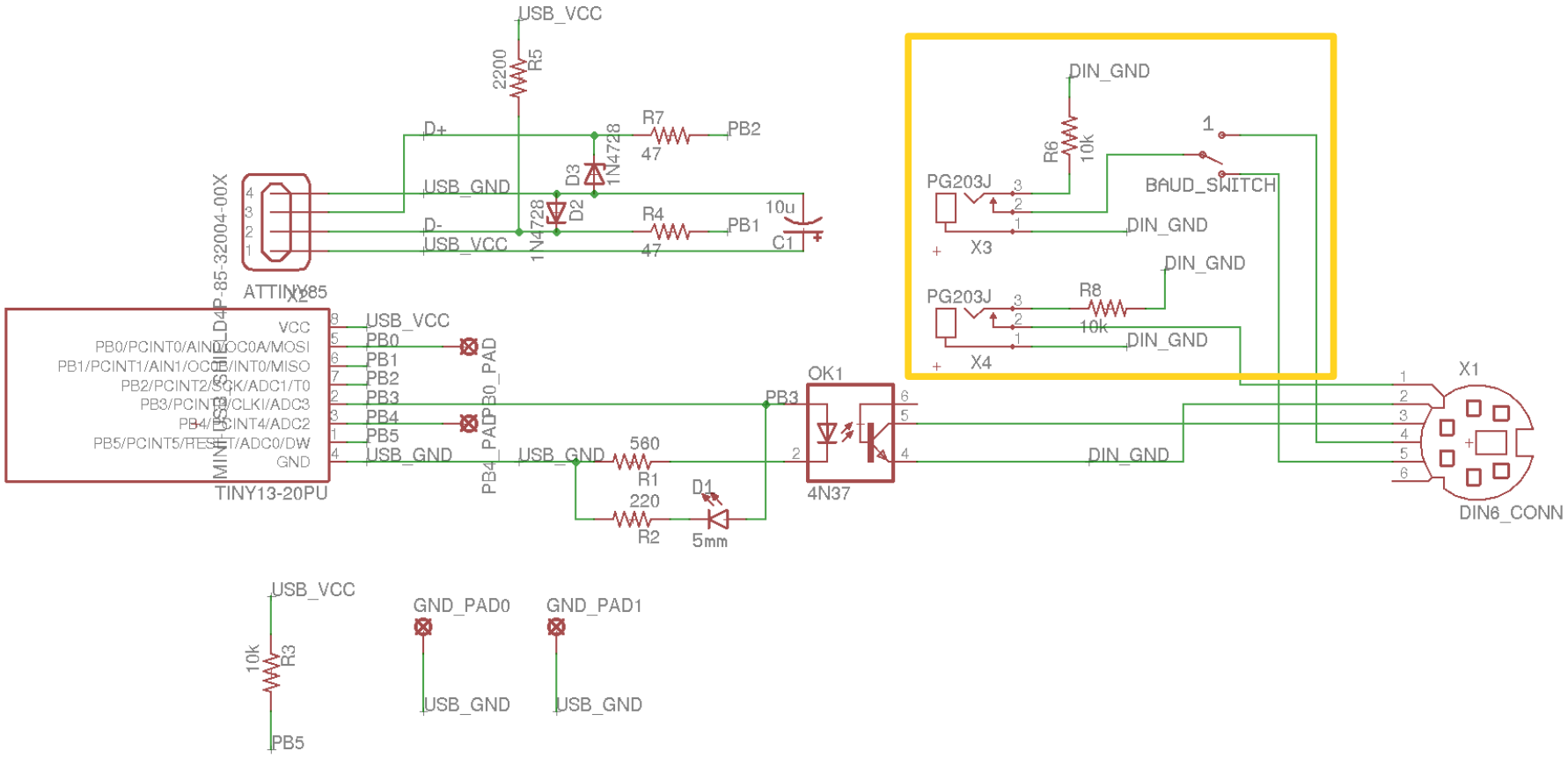
- Used to isolate computer USB from radio
- Transfers signal optically through LED and phototransistor
- µC turns the device on/off via the same output pin as the visual LED
- Two separate grounds (USB_GND & DIN_GND)



Circuit Design

Signal I/O

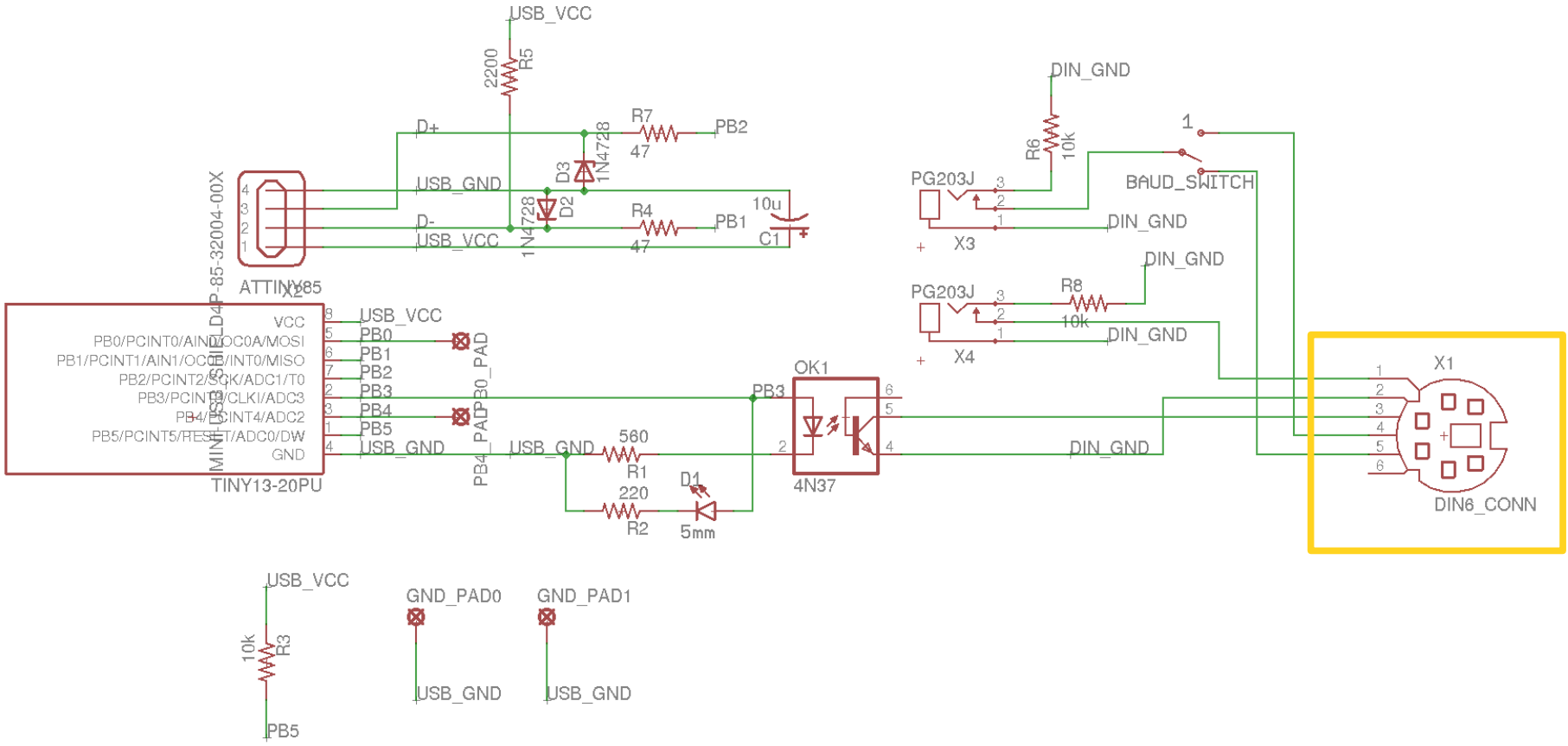
- Audio jacks used for Tx/Rx signals
- Slide switch on Rx signal selects 1200 or 9600bps
- Connected to ground through 10kΩ resistor when unplugged



Circuit Design

Mini-DIN 6-pin Connector

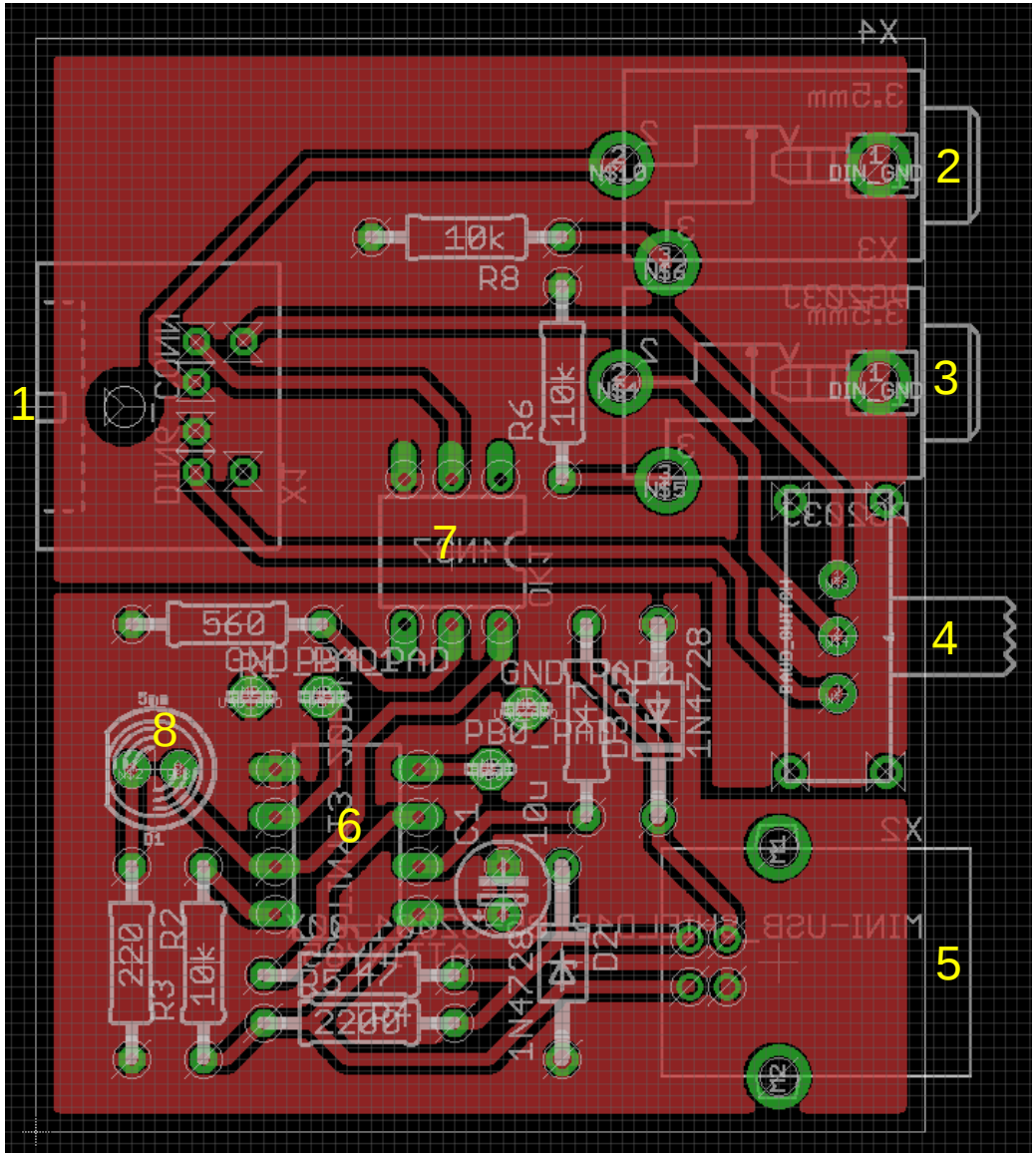
- Connects everything between the circuit and the radio
  - PTT pin
  - Ground
  - Tx line
  - 1200/9600bps Rx lines



Circuit Design

Layout

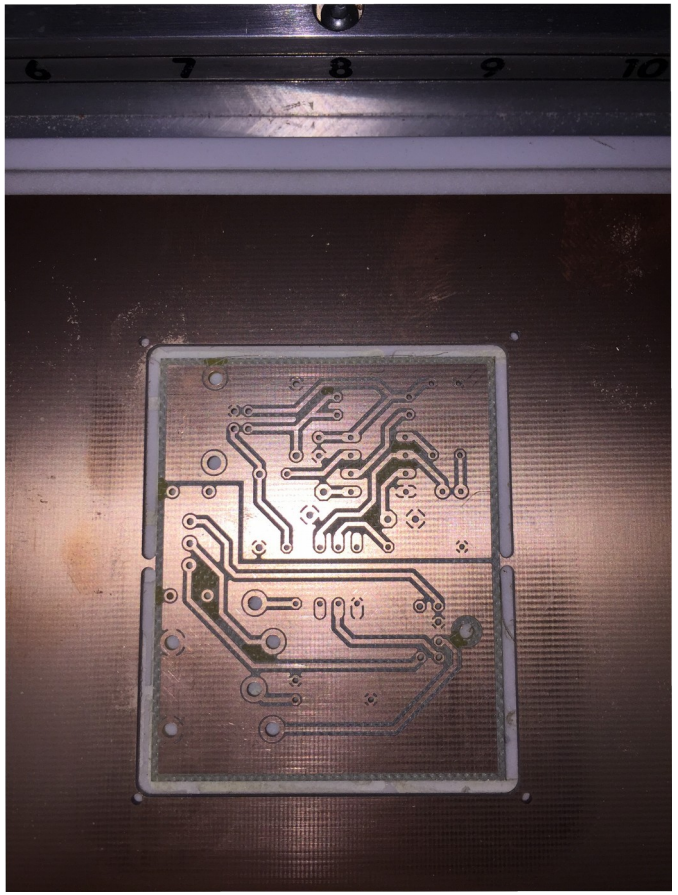
- Layout (and schematic) done in EAGLE PCB Software
- Single-layer PCB
- Two ground planes for isolation
- Component Location
  - 1 – Mini-DIN 6-pin Connector
  - 2 – Audio Input
  - 3 – Audio Output
  - 4 – 9600/12000 Baud Slide Switch
  - 5 – USB Connector
  - 6 – ATtiny85 µC
  - 7 – Optoisolator
  - 8 – Visual LED



Fabrication

PCB Fabrication

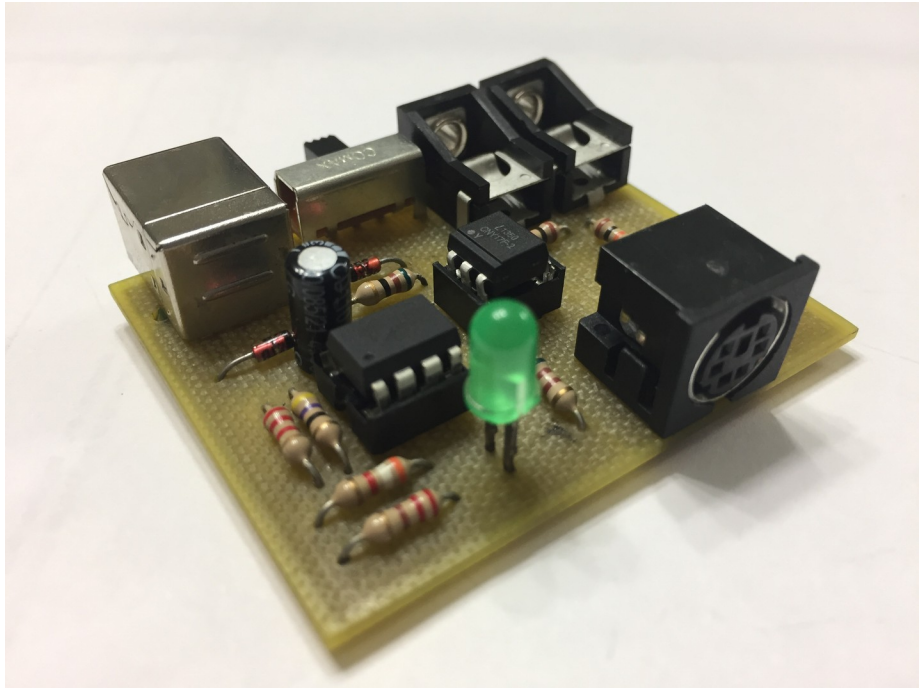
- LPKF PCB milling machine in Fablab
- Approx 45 minutes fabrication time

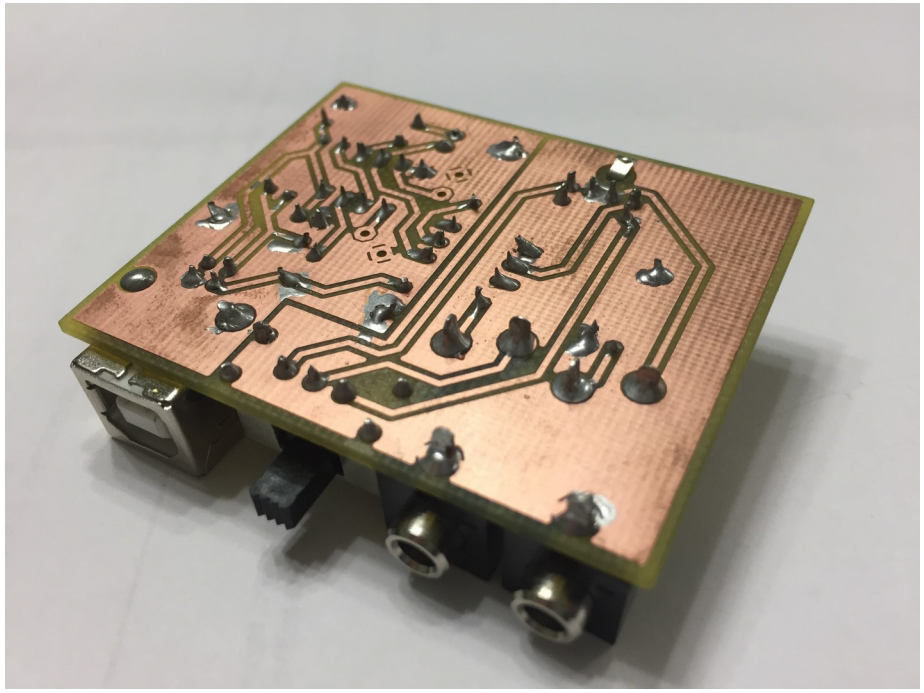


Fabrication

Board Assembly

- Board with soldered components

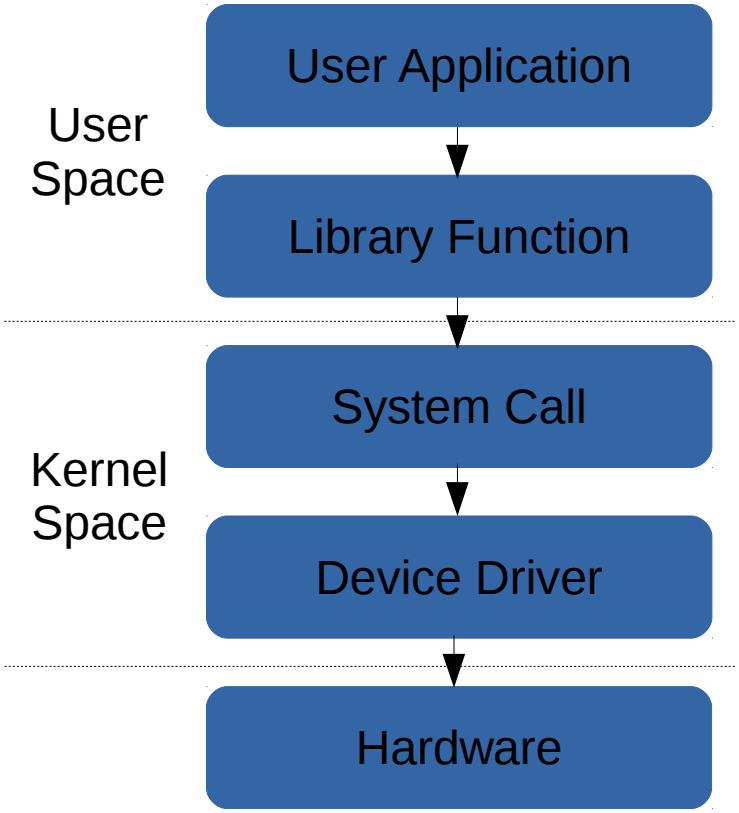




Driver Software

Software Hierarchy

- *The following applies to Linux systems
- Direwolf (user space program) needs to control the USB device
- Device driver (kernel space program) needed to allow user space program to interact with the actual hardware device



Driver Software

ptt_driver.c

- ptt_driver.c is created as the device driver
- One important structure and several important functions are defined here
  - ptt_table is a list of all devices that will work with this driver
    - Device is identified by the same unique number that was programmed into the microcontroller
  - ptt_probe registers the device with the kernel when it plugged in
    - The “class” parameter being passed in contains the name for the device file, “ptt%d”
  - ptt_disconnect will de-register the device when it is unplugged

```
106 /* Table of devices that work with this driver */
107 static struct usb_device_id ptt_table[] =
108 {
109   { USB_DEVICE(0x16c0, 0x05dc) },
110   {} /* Terminating entry */
111 };
```

```
77 static int ptt_probe(struct usb_interface *interface, const struct usb_device_id *id)
78 {
79   int retval;
80
81   if(DEBUG) printk("Function: ptt_probe");
82
83   device = interface_to_usbdev(interface);
84
85   class.name = "usb/ptt%d";
86   class.fops = &fops;
87   if ((retval = usb_register_dev(interface, &class)) < 0)
88   {
89     /* Something prevented us from registering this driver */
90     printk("Not able to get a minor for this device.");
91   }
92   else
93   {
94     if(DEBUG) printk(KERN_INFO "Minor obtained: %d\n", interface->minor);
95   }
96
97   return retval;
98 }
99
100 static void ptt_disconnect(struct usb_interface *interface)
101 {
102   if(DEBUG) printk("Function: ptt_disconnect\n");
103   usb_deregister_dev(interface, &class);
104 }
```

Driver Software

ptt_driver.c

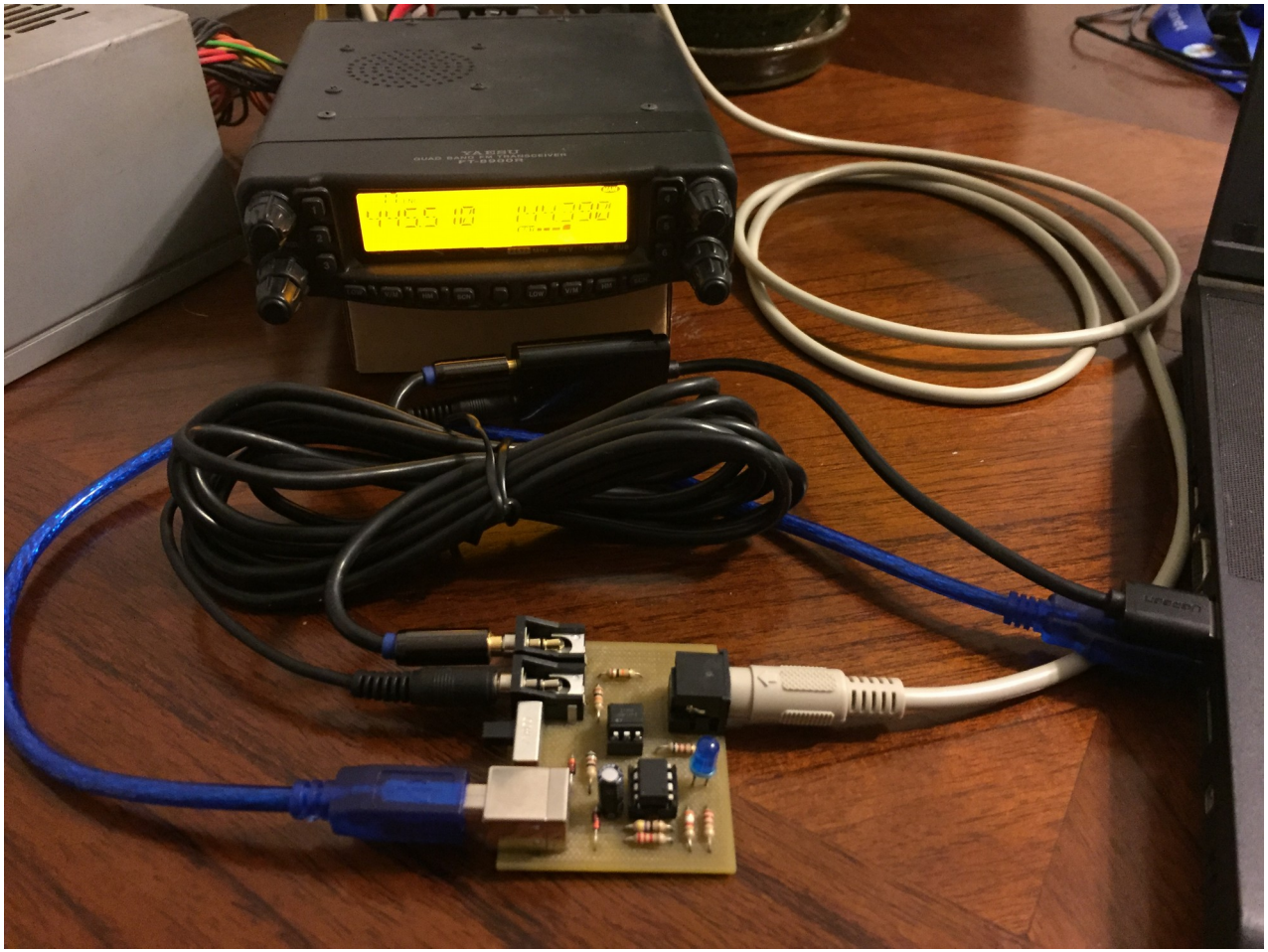
- When Direwolf sends a command to the device, it uses a system call function, called “ioctl”, or I/O Control
- One of the parameters of ioctl specifies the device file that the command is for (“ptt%d”)
- This command is then caught by the driver and interpreted
- The driver then uses a special function called “usb_control_msg” to send the proper command to the actual hardware device

```
 8 #define USB_LED_OFF 0x00
 9 #define USB_LED_ON 0x01
10
11 static long my_ioctl(struct file *f, unsigned int cmd, unsigned long __user stuff)
12 {
13   int retval = 0;
14   char buffer[8];
15   int transfer;
16
17   memset(&buffer, 0, sizeof(buffer));
18   memset(&transfer, 0, sizeof(transfer));
19
20   switch (cmd)
21   {
22     case TIOCMGET:
23       transfer = 0;
24       if (copy_to_user((unsigned long*)stuff,&transfer,4))
25       {
26         return -EFAULT;
27       }
28       break;
29     case TIOCMSET:
30       //These two options are used by Direwolf software to turn on the transitter
31       if ((*(unsigned long*)stuff & 0x00000000FFFFFFFF) != 0)
32       {
33         retval = usb_control_msg(device,usb_sndctrlpipe(device,0),USB_LED_ON,USB_TYPE_VENDOR | USB_RECIP_DEVICE,0,0,&buffer,sizeof(buffer),5000);
34       }
35       else if((*(unsigned long*)stuff & 0x00000000FFFFFFFF) == 0)
36       {
37         retval = usb_control_msg(device,usb_sndctrlpipe(device,0),USB_LED_OFF,USB_TYPE_VENDOR | USB_RECIP_DEVICE,0,0,&buffer,sizeof(buffer),5000);
38       }
39       break;
40     default:
41       retval = usb_control_msg(device,usb_sndctrlpipe(device,0),USB_LED_OFF,USB_TYPE_VENDOR | USB_RECIP_DEVICE,0,0,&buffer,sizeof(buffer),5000);
42   }
43
44   if(retval)
45   {
46     return retval;
47   }
48
49   return 0;
50 }
```

Driver Software

ptt_driver.ko

- ptt_driver.c can then be compiled into an executable file and loaded into the Linux kernel
- The driver file has the “.ko” extension at the end, as do all Linux driver files
- Finally, the hardware device can be plugged into the computer and radio



Integration

Direwolf

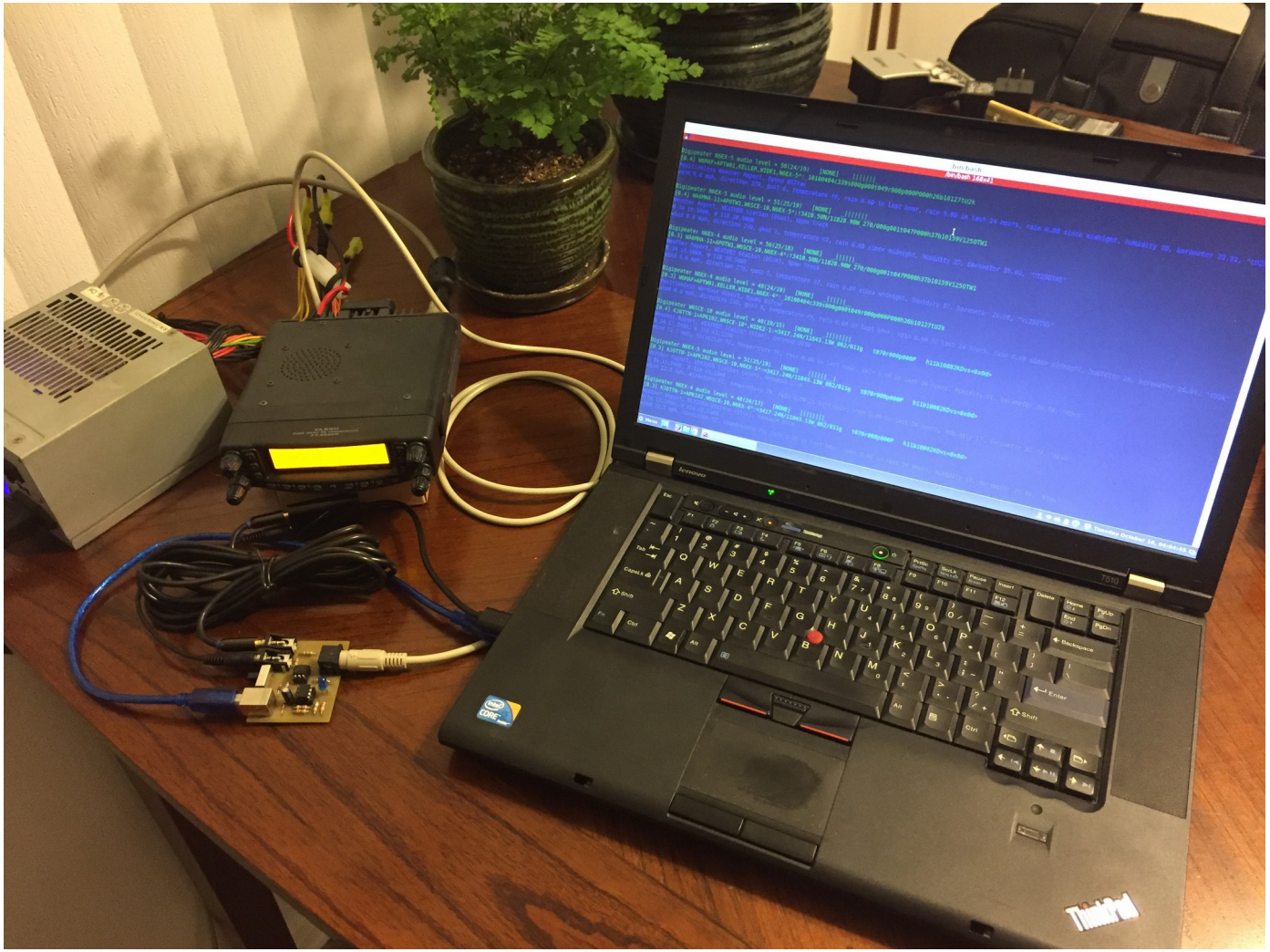
- Now, Direwolf can be configured to use the new device
  - “direwolf.conf” is created automatically by the software in the user's home directory
  - Changed the desired channel PTT property to “PTT /dev/ptt0 RTS”

```
176 # For the PTT command, specify the device and either RTS or DTR.
177 # RTS or DTR may be preceded by "-" to invert the signal.
178 # Both can be used for interfaces that want them driven with opposite polarity.
179 #
180 # COM1 can be used instead of /dev/ttyS0, COM2 for /dev/ttyS1, and so on.
181 #
182
183 #PTT COM1 RTS
184 #PTT COM1 RTS -DTR
185 PTT /dev/ptt1 RTS
```

Integration

Direwolf

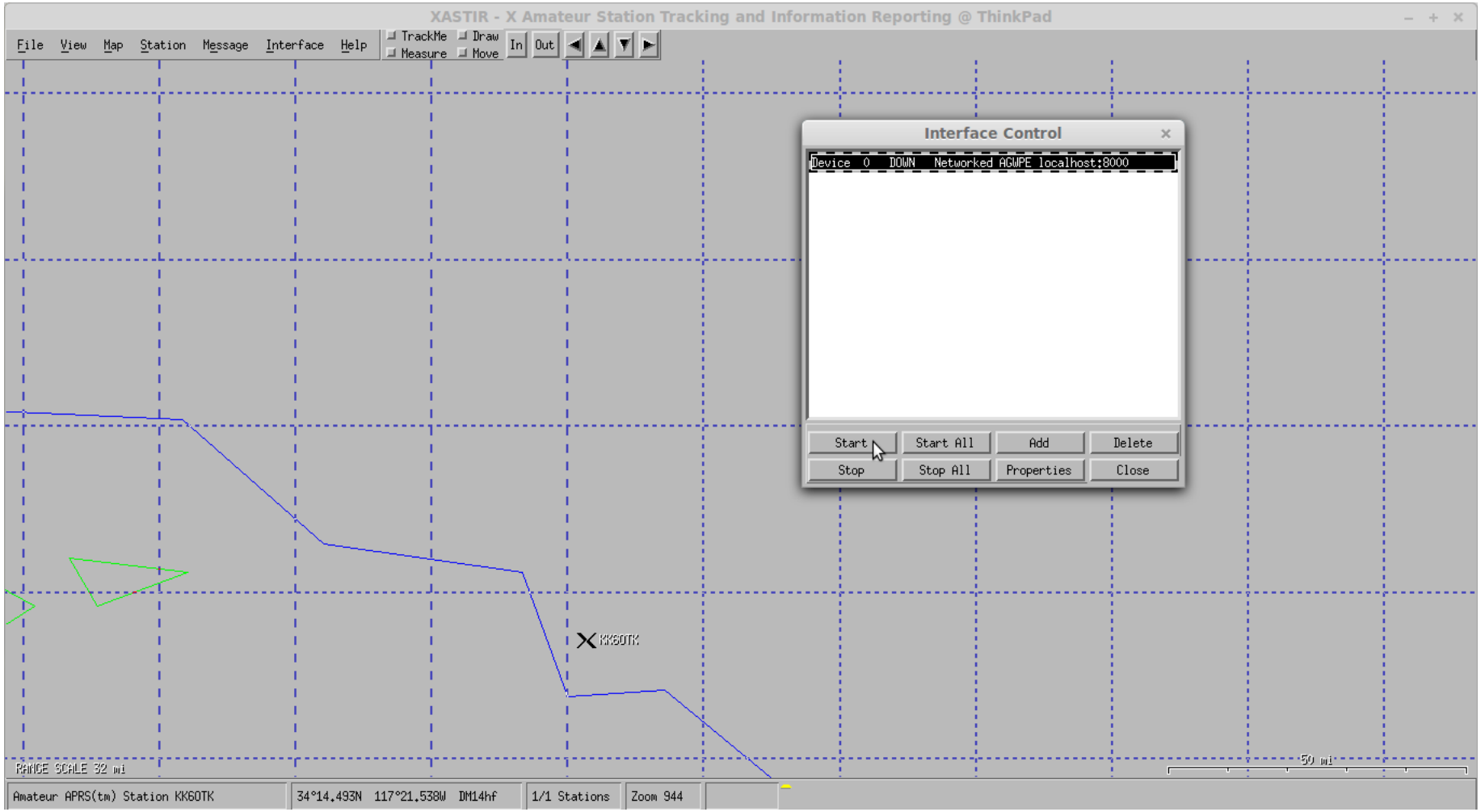
- Open Direwolf and see all the incoming messages



Integration

Xastir

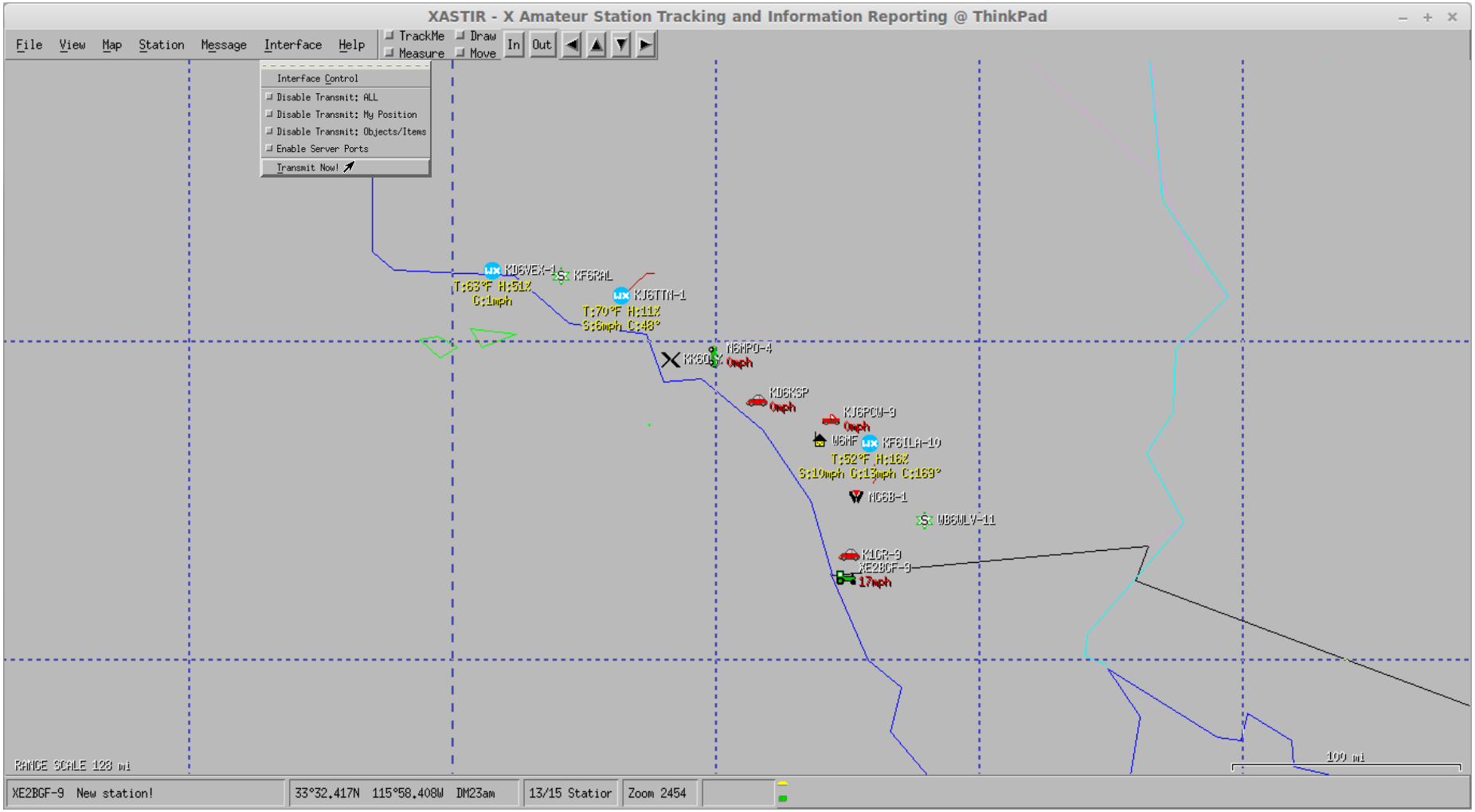
- Open Xastir and connect to Direwolf



Integration

Xastir

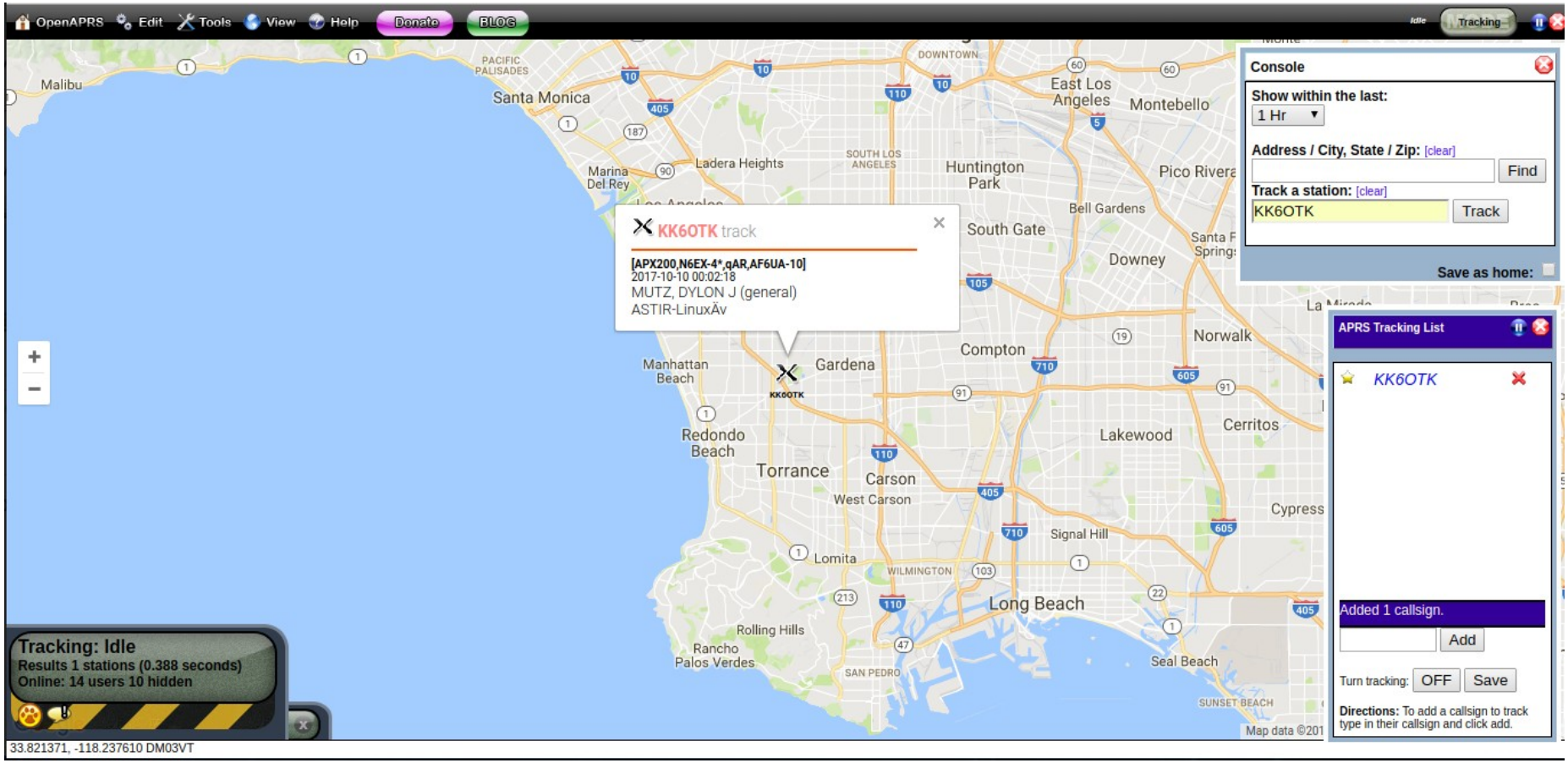
- See object start popping up on the map
- Transmit callsign and location



Integration

OpenAPRS

- Check www.openaprs.net to verify message was heard



Demo

Other Comments

Other Comments

- Developed in Linux, but could be applied to Windows and Mac
- AVR chips + V-USB
  - Can be applied to much more complicated devices/machines
    - Interfaces & adapters
    - Data acquisition
    - Display and LEDs
    - Motor control
  - DIY software as well to control custom hardware
- Slides and all software code, schematic, layout, etc is on my GitHub profile
  - https://github.com/KK6OTK/USB_PTT
- Radio connector and layout could be modified for other radios

Future Improvements

Future Improvements

- Support for HT connections
- Audio isolation between radio and soundcard
- Support for squelch
- Support more software options
- Resistor on optoisolator bias pin
  - Prevent stray RF from being picked up on open wire
- Housing for the PCB
- Installation script for driver
- Microsoft Windows driver

Bill of Materials

Part	Digikey Part #	Quantity	Total Cost
USB Connector	ED2982-ND	1	$0.55
8-pin IC Socket	AE9986-ND	1	$0.18
6-pin IC Socket	A1203467-ND	1	$0.20
ATtiny85	ATTINY85-20PU-ND	1	$1.26
Optoisolator	160-1318-5-ND	1	$0.37
3.6V Zener Diode	1N5226B-TPCT-ND	2	$0.24
6-pin mini-DIN Connector	MD-60S	1	$1.15
SPDT Slide Switch	CKN10393-ND	1	$0.50
3.5mm Mono Jack	CP-3536N-ND	2	$1.32
5mm LED	Spare parts	1	~$0.10
Resistors and capacitors	Spare parts	9	~$0.90
USB, Audio, & DIN Cable	Spare parts	4	~$10.00
PCB	Made in Fablab	1	$0.00
Total			**~$16.77**

References

References

[1] APRS: Automatic Packet Reporting System. Available: http://www.aprs.org/

[2] Sound Card Packet TNC Computer to Radio Interface. Available: http://kb3kai.com/sound-card-tnc

[3] Ham Radio Software on Centos Linux. Available: http://www.trinityos.com/HAM/CentosDigitalModes/hampacketizing-centos.html#6.softtnc

[4] Direwolf Soundcard Packet on Linux, with AX25 and LinPac. Available: https://www.kevinhooke.com/2015/09/12/direwolf-soundcard-packet-on-linux-with-ax25-and-linpac/

[5] Packet Radio: Introduction to Packet Radio. Available: https://www.tapr.org/pr_intro.html

[6] GitHub: wb20sz/direwolf. Available: https://github.com/wb2osz/direwolf

[7] XastirWiki. Available: https://xastir.org/index.php/Main_Page

[8] V-USB with ATtiny45/ATtiny85 without a crystal. Available: http://codeandlife.com/2012/02/22/v-usb-with-attiny45-attiny85-without-a-crystal/

[9] USB in a Nutshell. Available: http://www.beyondlogic.org/usbnutshell/usb2.shtml

[10] An Introduction to Device Drivers in the Linux Kernel. Available: http://opensourceforu.com/2014/10/an-introduction-to-device-drivers-in-the-linux-kernel/

[11] OpenAPRS. Available: http://www.openaprs.net/

[12] Google Maps APRS. Available: https://aprs.fi/