A tx module for the Red Pitaya and the Raspberry Pi (Zero)

Jon Ove Johnsen, LA3JJ
18th of November 2016

1. Introduction

I strongly felt that I wanted to make a printed circuit board that would add some tx ability to the Red Pitaya and also to the Raspberry Pi Zero. The idea started some years ago with Ross (VK1UN etc) who was using Raspberry Pi with the WsprryPi program to generate wspr signals. I know he might say today: What took you so long! My only excuse is that I had to learn to master some techniques before I could do this properly. Mainly two things were important:

- Basic knowledge of a PCB design program (I selected Kicad)
- Basic knowledge of Python programming and making scripts

At age of 73 this takes some time, however it accelerated when I took some of the courses of Peter Dalmaris [1].

Another great inspirational source was the QRP Labs of Hans Summers [2]. I have used his U3 and U3S a lot and decided that the pluggable and switch-able LPF modules was a great idea.

This summer with the following units on my desktop:

- Red Pitaya
- Raspberry Pi Zero
- QRP Labs U3S

I started making sketches of schematics and layout.
I used the footprint of the Pi Zero, and it should also fit nicely on top of the Red Pitaya.

Important in every project is also to have a list of what NOT to do:

- not integrating the rx input circuitry of the Red Pitaya (a future project?)
- not using surface mount components

The the drawing and layout work went smoothly with Kicad and the prints were ordered by sending the Gerber files to OSH Park.

Finally when trying to document this I see that it needs a name. I decided to call it “wiZPit”.

2. Design

The main design steps:

- make a prototype wiZPit-alfa
- redraw the schematic while getting user experience with the xZPit-alfa
- do the PCB layout
- assemble 6 wiZPit-betas
- let friends test the wiZPit-betas
- remake layout, make PCB available to public

How it works: (with reference to the schematic included as separate file attachment)
wiZPit has two active parts, the amplifier and the relay.

The amplifier is the standard amplifier with a BS170 used in the QRP Labs transmitters. It takes its rf signal from the GPIO-04 of the Raspberry Pi's 40 pin connector. When using Red Pitaya the rf signal is supplied through the SMA connector P6. The bias of the transistor is adjusted by VR1. The 3-pin jumper JP1 is there for making bias adjustment fool proof. The procedure is like this:

- with no rf-drive applied, attach a voltmeter between JP1 pin 1 and pin 2
- adjust VR1 until voltmeter shows 0.3 volts
- remove voltmeter and place a shorting plug between JP1 pin 2 and pin 3

This setting will give more than 200mW output on the 40m band. It is possible to get more power out of the BS170 by using higher supply and/or using a transformer output in stead of just an rf choke. I have not decided yet if these possibilities will be included in next generation of the board. The amplifier is followed by a LPF for the band in question and is well documented in the QRP Labs website,

The relay is controlled from signal GPIO-17 of the Raspberry Pi or from P1 nn of the Red Pitaya. A transistor BC547B or similar is driving the coil which is polarized. That had slipped my mind when doing the beta layout so I had to mount the relay on the backside of the board to give the coil the correct polarity. The relay switches the signals between antenna, rx and tx. It shortens the rx input when transmitting.

While testing the units I have found that a LED indicating the relay status is important, and that will surely be included in the next layout. Currently LED and resistor is just directly soldered across the relay protection diode D1.

3. Software

Of course, without proper software the wiZPit is of no use. It may be used with many software programs once they are properly adapted. What was important for me was to have wspr programs to run both from Red Pitaya and the Raspberry Pi Zero at an early stage for the testing and for getting on the air!

Software for the Raspberry Pi Zero was quite easy to adapt. I used the
https://github.com/JamesP6000/WsprryPi

First install that program according to instructions.
You should end up with it installed in a directory called WsprryPi under you home directory.

To run it every 8th minutes starting at 4th minute after the hour I used this cron command:

*3-59/8 * * * * sleep 54 && /home/pi/wsprtx.sh*

To set up and edit a *cron* command you use:

crontab -e

When doing this first time you will be asked which editor you would use, I recommend *nano*.

The *cron* will call the program *wsprtx.sh*

it looks like this:

cd ~/
sudo python relay_on.py
cd WsprryPi
sudo ./wspr --self-calibration LA3JJ JO59 23 7040133
 # change the line above to YOUR call sign, location, power level and frequency.
cd ~/
sudo python relay_off.py

The *wsprtx.sh* command will call the two python commands for relay on and relay off:

```
#!/usr/bin/python
# relay_on.py
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
GPIO.setwarnings(False)
GPIO.setup(17,GPIO.OUT)
print "Relay on tx"
GPIO.output(17,GPIO.HIGH)
```

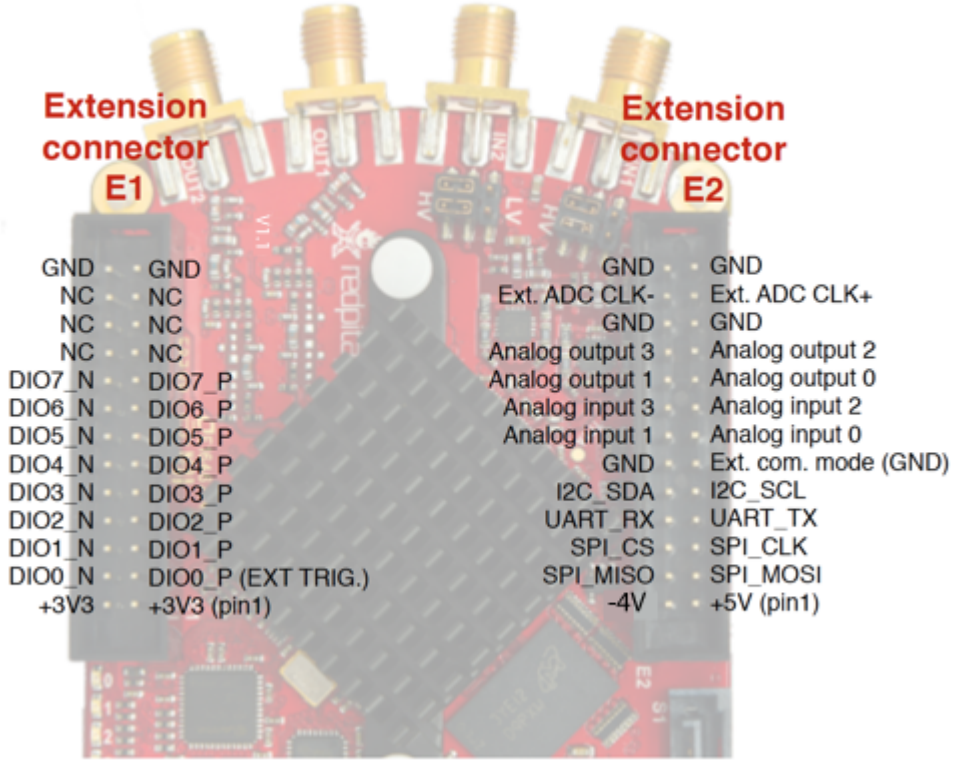
```
#!/usr/bin/python
# relay_off.py
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
# GPIO.cleanup()
GPIO.setwarnings(False)
GPIO.setup(17,GPIO.OUT)
print "Relay on rx"
GPIO.output(17,GPIO.LOW)
```

After having made those 3 files your /home/pi directory should look like this:

```
pi@raspberrypi:~ $ ls
Desktop  Downloads  Pictures  Templates  WsprryPi     relay_off.py  wsprtx.sh
Documents  Music    Public    Videos     python_games  relay_on.py
pi@raspberrypi:~ $
```

When you restart the RasPi it should be transmitting, - and you could look for spots on wsprnet.org.
A receiver may be connected to the P4 SMA-coax connector sharing the antenna with the wiZPit.

With the software for the Red Pitaya Pavel Demin kindly helped with making some code for writing to the extension pins of the Red Pitaya.



Signal to switch the relay is taken from pin DIO0_P of the E1 connector and the 5V power is taken from pin 1 of the E2 connector. In addition I decided to connect GND from both connectors. There is a 4-pin connector for this purpose in the upper left corner of the wiZPit. I used 4 wires from a ribbon cable with female Dupont connectors in both ends for this purpose. If you are using the wiZPit for Red Pitaya only there is no reason for mounting the 40pin connector on the wiZPit board. I assume you have already a correctly installed version of Pavel's multiwspr program that is working and that you know how to control the rx and tx channels. Here is then the instruction for making the software changes to the *multiwspr* program from Pavel:

I've just added a small program called gpio-output.c with the following commit:
https://github.com/pavel-demin/red-pitaya-notes/commit/b0e13be7980fa55929c158489a416880e13c75ec

This program controls the DIO*_P pins of the extension connector E1:
http://wiki.redpitaya.com/index.php?title=Extension_connectors#Extension_connector_E1

I've also added this program to the decode-wspr.sh and transmit-wspr.sh scripts. The pin DIO0_P is switched to high one second before the transmission and it's switched to low one second before the reception.

The program and all the modified files can be downloaded to Red Pitaya with the following commands:

wget https://raw.githubusercontent.com/pavel-demin/red-pitaya-notes/master/projects/sdr_transceiver_wspr/Makefile
wget https://raw.githubusercontent.com/pavel-demin/red-pitaya-notes/master/projects/sdr_transceiver_wspr/gpio-output.c
wget https://raw.githubusercontent.com/pavel-demin/red-pitaya-notes/master/projects/sdr_transceiver_wspr/decode-wspr.sh
wget https://raw.githubusercontent.com/pavel-demin/red-pitaya-notes/master/projects/sdr_transceiver_wspr/transmit-wspr.sh
The program can be compiled with the following command:

make

I got some error messages when running the compilation with the *make* command, and I found that I had to rename some of the files and to run chmod +X on two files after compiling to make it work.

I will make some more SD cards for my hardware and after that will include a more detailed step-by-step instruction.

4. Hardware setup

The signal connectors of the wiZPit are the same as for the Red Pitaya: 50 ohms SMA connectors. You must be careful to use correct connection cables. Please see to it, when ordering cables, that connectors are of the SMA type and NOT the RP-SMA!

On the wiZPit the 3 coax connectors should be marked T – R – A from left to right. The A connector is for the antenna, or via an antenna tuner. It is important that the the antenna has acceptable SWR on the band you intend to transmit on. The antenna should also have a DC return to ground to avoid static charges. The wiZPit has as a small bleeder resistor installed, but it should not be regarded as sufficient protection for statics.

A word of caution regarding the inputs of the Red Pitaya: The wiZPit does not include rx input circuits to adapt and protect!

The R connector is to be attached via an input protection circuit for the red Pitaya input 1 or input 2.. This circuit should at least contain an impedance converter 1:25 or 1:36 ratio (1:5 or 1:6 turns ratio) and a loading resistor less than 2k2 ohms. It is also important to have a diode clamping circuit to limit the voltage from near by transmitters. The wiZPit does not include any of this.

The T connector is to be attached to the Red Pitaya output 1.

The control signal is attached as explained in the previous chapter.

For the Raspberry Pi Zero all connections are done via the 40pin connector. The coax T connector is not used. The R connector may be attached to receiver. The A connector is attached to the antenna adapted to the actual band(s) for the txing.