Prosper Su
SID: 490447846

SDR 2.0

A simple but functional WSPR reciever

Our goal is to design an improved Software Defined Radio (SDR2.0) circuit behaving as the WSPR receiver that specifically targets the WSPR signal range and decodes it using software. Building a low-cost WSPR receiver can be challenging activity for radio technology enthusiasts who enjoy receiving messages from around the world. The availability of open-source software and affordable hardware options for WSPR receivers makes it accessible to a larger number of people, including students who are interested in learning about radio communication technology. Hence, the entire report focuses on 4 main stages: research, design, testing procedures and performance.

Background

Before moving into the design of the SDR 2.0. The readers are introduced to three main core technologies, SDR, WSPR and ADC implemented for the project.

Software Defined Radio (SDR) has brought about a revolutionary transformation in the field of radio communication. By replacing traditional hardware components with software algorithms, SDR enables remarkable flexibility in radio systems. Through software-based modulation, demodulation, filtering, and signal processing, SDR empowers efficient reconfiguration and adaptability.[1] This breakthrough has resulted in compact hardware designs, cost savings, and enhanced performance in terms of speed and flexibility. SDR has found applications across diverse domains such as telecommunications, defence, amateur radio, and wireless sensor networks. Its impact on these sectors has been truly transformative.

Then, WSPR, introduced in November 2010, was developed by Emeritus Professor Joe Taylor, an astrophysicist at Princeton University. This protocol serves the purpose of testing band propagation through the transmission and reception of low-power signals. Unlike conventional radio transmissions that carry speech or music, WSPR employs specially coded transmissions that reveal the transmitter's source and location.[2] The standard WSPR message format includes the callsign, a 4-character locator, and the transmit power in dBm. Due to its weak nature (typically 5W or less) and low signal-to-noise ratio (as low as -28dB in a 2500Hz bandwidth), it necessitates an amplifier and a well-designed filtering system capable of removing high-frequency noise. The signal also incorporates Forward Error Correction (FEC) to safeguard against bit errors and facilitate message recovery upon correct decoding. WSPR activities take place across 10 different frequency bands, with our focus being on the 7.0386MHz frequency, which is a crucial parameter for designing an appropriate bandpass filter.

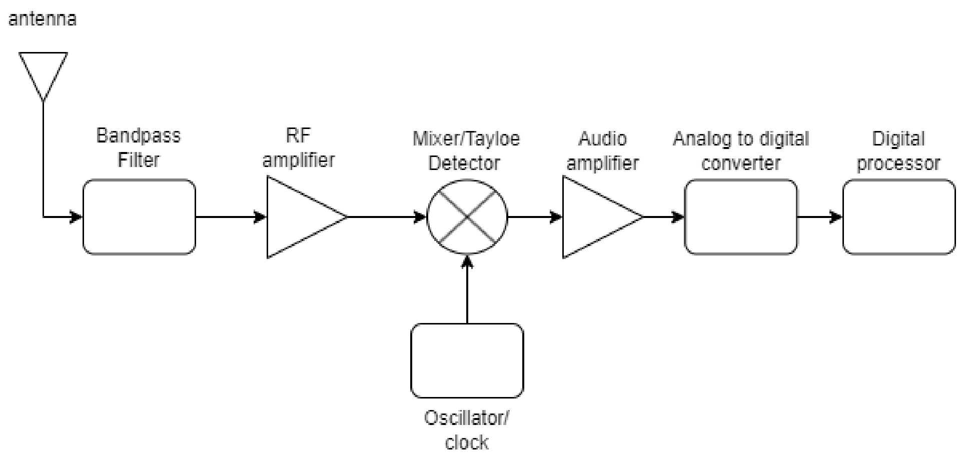


Figure 1.0 SDR 2.0 block diagram

Analog-to-digital converters (ADCs) are essential components found in contemporary electronic systems, facilitating the conversion of analog signals into a digital format. With the increasing need for digital processing and analysis of real-world signals, ADCs serve as vital intermediaries between the analog and digital realms. [3] By capturing and quantizing continuous analog signals, these devices enable the processing, storage, and manipulation of the signals by digital systems. In essence, ADCs bridge the gap between the analog and digital domains, enabling seamless integration and utilization of both types of signals in modern electronic systems.

Design Requirements

To design a proper SDR 2.0, we initially conduct the following requirements after researching:

- Low cost
- Isolate the correct range of signal 7.04000MHz - 7.040200MHz
- Signal to Noise ratio (SNR) 65 dB to 80 dB
- Bit error ratio (BER) <2%
- PCB board for SDR 2.0 circuit

Hardware Design

The original SDR 1.0 is modified and added with extra components. The SDR 1.0 provided by the course is only consisted of a RF preamplifier, a mixer, an audio amplifier, whereas the improved version (SDR 2.0 figure 1.0) modifies RF preamplifier and an audio amplifier and adds two new components, a bandpass filter, and an ADC to the design.

Prosper Su
SID: 490447846

Bandpass filter
The design of the Bandpass filter meets several specifications, including a maximum ripple of less than 0.1dB within the WSPR region (7.0400-7.0402 MHz), a centre frequency set at 7.0401 MHz, a minimum of -3dB attenuation 0.25 MHz from the centre frequency (6.7901 MHz – 7.2901 MHz), and a minimum -50dB attenuation at 10 MHz. The input and output impedance are required to be 50 ohms with 1% tolerance capacitor and inductor values. We design a 2nd order Butterworth Bandpass filter with a lower cut-off frequency at 6.9421 MHz and an upper cut-off frequency at 7.1381 MHz, resulting in a centre frequency of 7.0401 MHz that meets the design requirement. We set the input and output impedance to 50 Ohms, allowing a 1% tolerance for the capacitor and inductor values.

RF preamplifier
In SDR 1.0, the preamplifier exhibits a design weakness in power supply rejection, where any change in the supply voltage directly affects the biasing voltage set by the resistor divider. Additionally, as it is a non-inverting amplifier, fluctuations in the signal voltage on the supply line can be fed back to the operational amplifier, leading to undesirable oscillations, commonly known as "motor boating." To mitigate potential noise levels originating from the power supply and enhance the stability of the op-amp and signal output, a decoupling approach is employed to separate the biasing network from the supply.

Mixer
By feeding the appropriate carrier frequency of 7.0401 MHz to the original mixer in SDR 1.0, it successfully down converts the frequency as intended. As a result, we keep the same design. To alter the output of the mixer, pulse modifications are necessary, with the pulse period being 1/fc (where fc represents the carrier frequency). This adjustment is required to eliminate carrier frequency modulation, which ranges from 7040.1 kHz to 7040.104394 kHz, allowing us to extract the WISPR signal in the range of 1.4 kHz to 1.6 kHz. This extraction is achieved by multiplying the signal with a sinusoidal wave having the same frequency as the carrier.

Audio amplifier
The previous audio amplifier in the SDR 1.0 circuit produces an output voltage of only 0.7 V, and its 3dB cut-off frequency stands at 4.58 kHz. Unfortunately, this audio amplifier fails to deliver the required voltage match for our ADC. Hence, we decide to improve the situation by adjusting the resistor values, as per the equation of the active low-pass filter. The output voltage is directly influenced by the resistor ratios, while the 3dB cut-off frequency is determined by a multiple of the resistors. By modifying the resistor values, we can achieve a more suitable output voltage and 3dB cut-off frequency. The resistors used are 50 ohms and 30k ohms based on the equation: $A_v = 1 + \frac{R2}{R1}$, where A is voltage gain in volts and R2 is larger value resistor in ohms and R1 is smaller value resistor in ohms in our systems.

ADC
The current SDR 1.0 design utilizes a USB soundcard for the conversion of analog data to digital. While this implementation suffices, there is room for improvement in terms of downsizing the ADC. Additionally, using a separate sound card would require interfacing through a 3.5mm connector and USB cable, resulting in unnecessary clutter and a clunky setup. To address these concerns, we opt for a solution that offers a streamlined approach. Consequently, we implement the SSM2604 [4] codec chip which effectively resolves all these problems. The selection of the ADC chip is based on its compatibility with the Raspberry Pi, which supports I2C and I2S interfaces through specific pins (SDA4, SCL4, PCM pins). With its two left-right inputs, the ADC chip proves sufficient for our specific use case, while its 90dB signal-to-noise ratio (SNR) is considered good.

PCB board
We have chosen to utilize a 4-layer PCB for the implementation of our SDR 2.0. This decision is based on the recognition of the circuit's high complexity, primarily due to the significant number of components present on the same board. The coexistence of these components generates potential radiation and interface issues stemming from the currents flowing through each element. By employing a 4-layer PCB, we can achieve better isolation and reduction of these effects. Furthermore, the dimensions of the PCB are specifically designed to align with the mounting holes and pin headers of the Raspberry Pi Model 4.

While building up the PCB board, we find that our ADC requires a stable digital voltage supply, which must be more stable than the analog voltage supply. To address this, we implement a voltage division technique to obtain a 3.3V supply from the 5V pin, which powers all the other components, including the op-amp, Tayloe mixer, and clock generator. We make efforts to exclusively reserve the 3.3V pin for the ADC. However, the clock generator utilizes both the voltage-divided 5V pin and the 3.3V pin, as it requires two 3.3V supplies (as indicated in the schematic). We believe that with only two components using the 3.3V pin and implementing filtering at the digital voltage supply pin (DVDD), it would be sufficient for the ADC's requirements. Moreover, the 5V pin can deliver more power to other components compared to the 3.3V pin, making it unsuitable to rely solely on the 3.3V line.

Software Design
The main two files used to run the WSPR program are parec.c (recording function) and wsprd.c (decoding function).

parec.c modification
We are making changes to parec.c in accordance with the requirements of wsprd.c. In wsprd.c, a recording of 114 seconds is needed, which means we must modify MXRECTIME to be greater than 114 to enable the recording

for the specified duration. Therefore, we change it to 130. The sample rate is being adjusted to 12000 as specified in wsprd.c. In wsprd.c, npoints is set to 12000 multiplied by 114, so we are adjusting parec.c accordingly to match the desired sample rate. The format is set to PA_SAMPLE_S16LE, which is referenced from parecfile.c. This selection ensures the appropriate sample format for the audio recording in parec.c.

wsprd.c modification

Then, we declare the function "parec" near the top of the file, which is responsible for recording the audio. Next, we make a small modification to the "readwavfile" function. Since no command line argument is provided to the "wav" file, the "ptr_to_infile" will be NULL. Therefore, we add an "if" statement to check if we are reading from a file or recording using "parec.c". If it is NULL, we call the "parec" function with "buf2" and "npoints" as arguments. "Npoints" indicates the number of samples we want to record, while "buf2" is an array that stores each sample, with each sample represented as a 2-byte (short int) value. (Both "buf2" and "npoints" remain unmodified.) In the "main" function, we want to prevent the "usage ()" and "return 1" functions from executing, as they would stop our code since no command line argument indicating a file is provided. Therefore, we comment them out since our input is not from the command line but from "parec.c". Additionally, we assign "ptr_to_file" as "input from parec.c". Although this is just a random string, it ensures that the subsequent code using "strstr()" does not result in a segmentation fault while iterating through a NULL pointer. The following code section involves the use of "strstr()". The first two "if" and "else-if" conditions are false since our "ptr_to_infile" is "input from parec.c". The third "else-if" condition on lines 819 is our modification. We call "readwavfile()" with inputs similar to the code section within the "if" statement (lines 798-808). However, this time we set "ptr_to_infile" to NULL to call "parec.c" (referring to modification number 2). The remaining code is the same as the code section within the "if" statement. Additionally, we set "ptr_to_infile_suffix" to ".wav" to avoid a segmentation fault from the subsequent code. These are all the modifications we have made. Now, we can compile and run the code.

Testing methods and results

To test the performance of hardware design and software design, various software is implemented.

LTspice (hardware design)

Developed by Analog Devices, LTspice is a highly popular software used for simulating and analysing electronic circuits. It offers a robust and user-friendly environment that empowers engineers and students to design and evaluate analog and mixed-signal circuits. [5] Whether working on simple or intricate circuit configurations, LTspice provides the necessary tools and capabilities for efficient circuit design and testing. The most of hardware design part is simulated via the software. The final schematic in LTspice is shown, each component is connected in the following orders, a bandpass filter, a RF preamplifier, a mixer, and an audio amplifier.
*Note: An ADC isn't simulating due to its functionality. (Converting analog signal to digital)

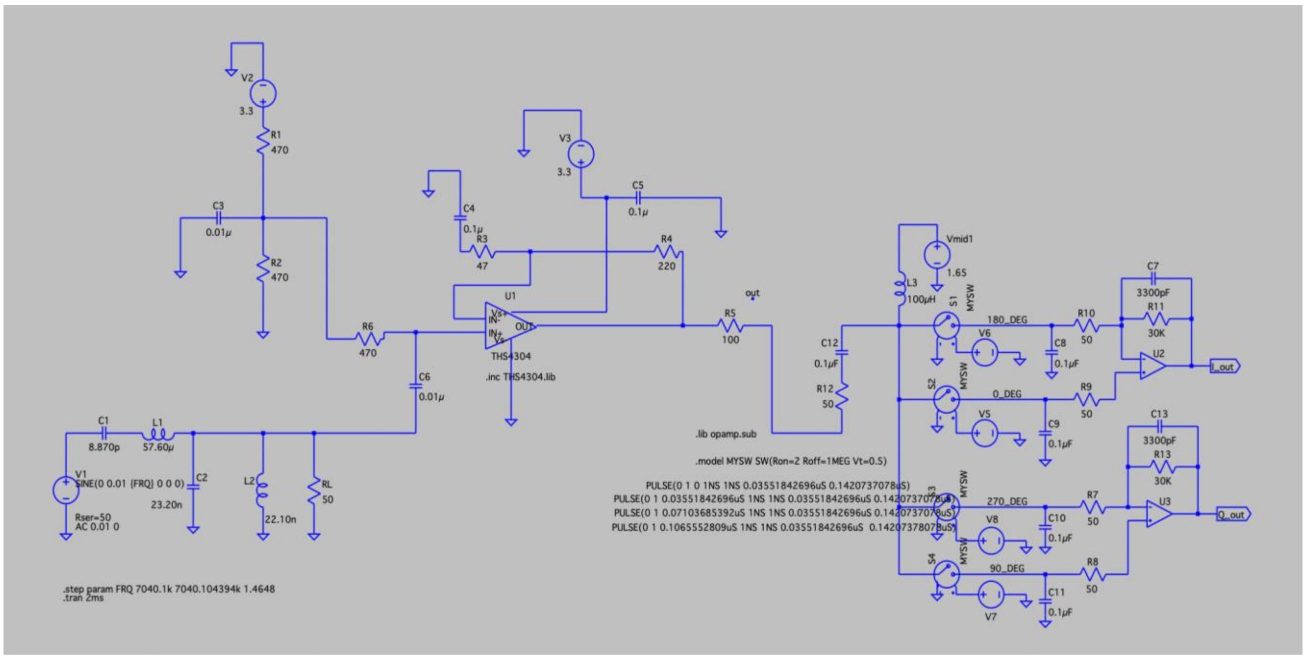


Figure 2.0 SDR 2.0 schematic in LTspice

Prosper Su
SID: 490447846

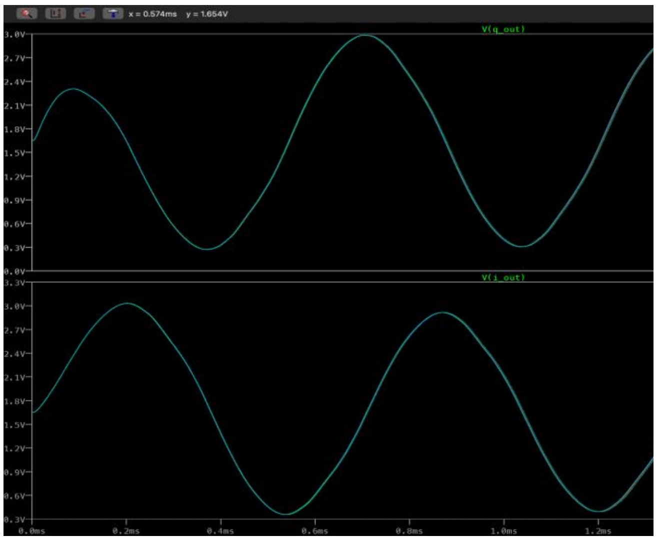


Figure 2.1 The magnitude of SDR 2.0 output in time domain

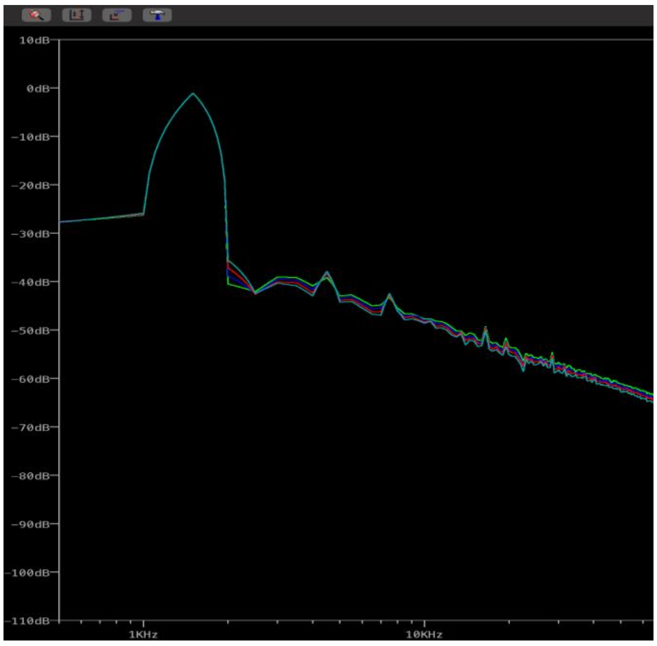


Figure 2.2 The magnitude of SDR 2.0 output in frequency domain.

The outputs obtained from LTspice simulation are 0.957 root-mean-square voltage ($V_{rms} = {V_C}/{\sqrt{2}}$, Vc is approximately 1.35, figure 2.1) within the frequency range of 1.4 kHz to 1.6 kHz (blue tip of figure 2.2).

KiCAD (hardware design)

KiCAD is an EDA software suite that is both free and open source, primarily used for designing PCBs. With its comprehensive toolset for schematic capture, PCB layout, and 3D visualization, KiCAD offers a versatile solution for various electronic design projects. Its capabilities extend from capturing schematics accurately to efficiently laying out PCB designs, all while enabling 3D visualization for a comprehensive understanding of the final product. [6]

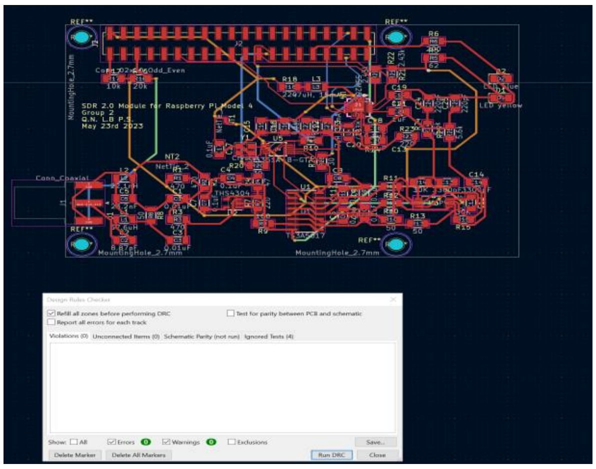
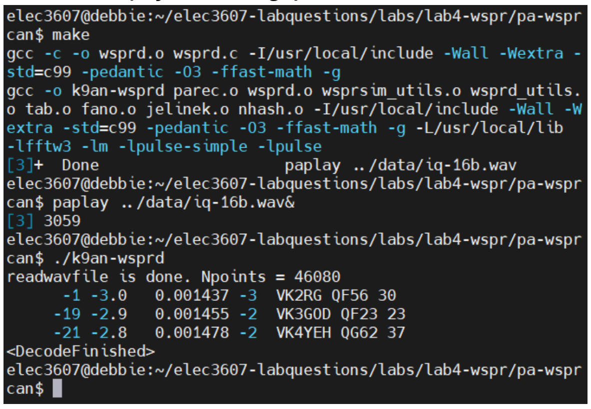


Figure 2.3 PCM schematic design

The four layers utilized in our design are F.Cu (red tracing), In1.Cu (light green), In2.Cu (orange), and B.Cu (blue). Each layer serves its unique functionality, namely signal, ground, power supply, and signal layers. (figure 2.3)

MobaXterm (software design)



```
elec3607@debbie:~/elec3607-labquestions/labs/lab4-wspr/pa-wspr
can$ make
gcc -c -o wsprd.o wsprd.c -I/usr/local/include -Wall -Wextra -
std=c99 -pedantic -O3 -ffast-math -g
gcc -o k9an-wsprd parec.o wsprd.o wsprsim_utils.o wsprd_utils.
o tab.o fano.o jelinek.o nhash.o -I/usr/local/include -Wall -W
extra -std=c99 -pedantic -O3 -ffast-math -g -L/usr/local/lib
-lfftw3 -lm -lpulse-simple -lpulse
[3]+  Done                    paplay ../data/iq-16b.wav
elec3607@debbie:~/elec3607-labquestions/labs/lab4-wspr/pa-wspr
can$ paplay ../data/iq-16b.wav&
[3] 3059
elec3607@debbie:~/elec3607-labquestions/labs/lab4-wspr/pa-wspr
can$ ./k9an-wsprd
readwavfile is done. Npoints = 46080
      -1 -3.0   0.001437 -3  VK2RG QF56 30
     -19 -2.9   0.001455 -2  VK3GOD QF23 23
     -21 -2.8   0.001478 -2  VK4YEH QG62 37
<DecodeFinished>
elec3607@debbie:~/elec3607-labquestions/labs/lab4-wspr/pa-wspr
can$ 
```

Figure 2.4 recording and decoding program in MobaXterm channel.

MobaXterm is a powerful and versatile remote access and terminal emulator software designed specifically for Windows. It combines a wide range of network tools into a single, convenient package, providing a comprehensive solution for remote computing, system administration, and network management tasks. With MobaXterm, we can seamlessly run Linux system (simple, powerful, and compatible to run our SDR 2.0) to design and program our software. The result is shown above (figure 2.4), the codes successfully record the sound and decode it into our pc.

Conclusion

The time taken to complete the entire project is long, however it ends up with satisfactory while learning and having fun at same time. The SDR 2.0 is a successful improvement because it meets most of our requirements, it is low cost because we avoid using inductor and simplify the circuit as small as we can. Then, when we feed the circuit with signals normally obtained from WSPR protocol and it produces the expected output which is between 1.4 kHz to 1.6 kHz. Next, the ADC we implemented guarantees at least 90 dB SNR which performs better than our requirements 65 dB to 80 dB. Lastly, the pcb is built up without any issues after compiling the in-built design rule check application. However. we forget to test out BER with systems. As the result, the future improvements are suggested that BER testing should be conducted. A lower BER indicates a higher level of accuracy and reliability in the data transmission system. For example, a BER of 0.01 means that, on average, one out of every 100 bits transmitted is received incorrectly. Furthermore, more filters should be implemented to obtain more accurate frequency desired.

Contributions

Our group has three members. Each member is assigned to different tasks to complete the SDR 2.0:
My contribution is listing below:

- A mixer design
- An audio amplifier design
- Overall circuit simulation
- PCB schematic design of mixer and audio amplifier

Reference

[1] P. V.-P. Vices, E. Team, and Jacob, “Everything RF,” everything RF, https://www.everythingrf.com/community/what-is-a-software-defined-radio (accessed Jun. 10, 2023).

[2] Andrew, “WSPR - weak signal propagation reporter,” VK3FS, https://3fs.net.au/vhf/wspr/ (accessed Jun. 10, 2023).

[3] W. Storr, “Analogue to digital converter (ADC) basics,” Basic Electronics Tutorials, https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html (accessed Jun. 10, 2023).

[4] “SSM2604,” Datasheet and Product Info | Analog Devices, https://www.analog.com/en/products/ssm2604.html#product-overview (accessed Jun. 10, 2023).

[5] Person, “An introduction to LTSpice,” Key, https://forum.digikey.com/t/an-introduction-to-ltspice/2023 (accessed Jun. 10, 2023).

[6] R. PCB, “What is kicad eda guide,” Printed Circuit Board Manufacturing & PCB Assembly - RayMing, https://www.raypcb.com/kicad-eda-guide/ (accessed Jun. 10, 2023).

[7] “Navigasjon,” NTNU Hjemmeside, https://i.ntnu.no/wiki/-/wiki/English/MobaXterm#:~:text=MobaXterm%20is%20a%20toolbox%20for,in%20a%20more%20simple%20fashion. (accessed Jun. 10, 2023).