GNU Radio Going Forward

Reference List:

GNU Radio Blocks:

- GNU Radio Block Wiki Search
- FM Demod
- QPSK Mod and Demod

LimeSDR:

- LimeSDR Blocks Info
- Hardware Documentation
- Updated LimeSuite

Finishing Up From Last Week

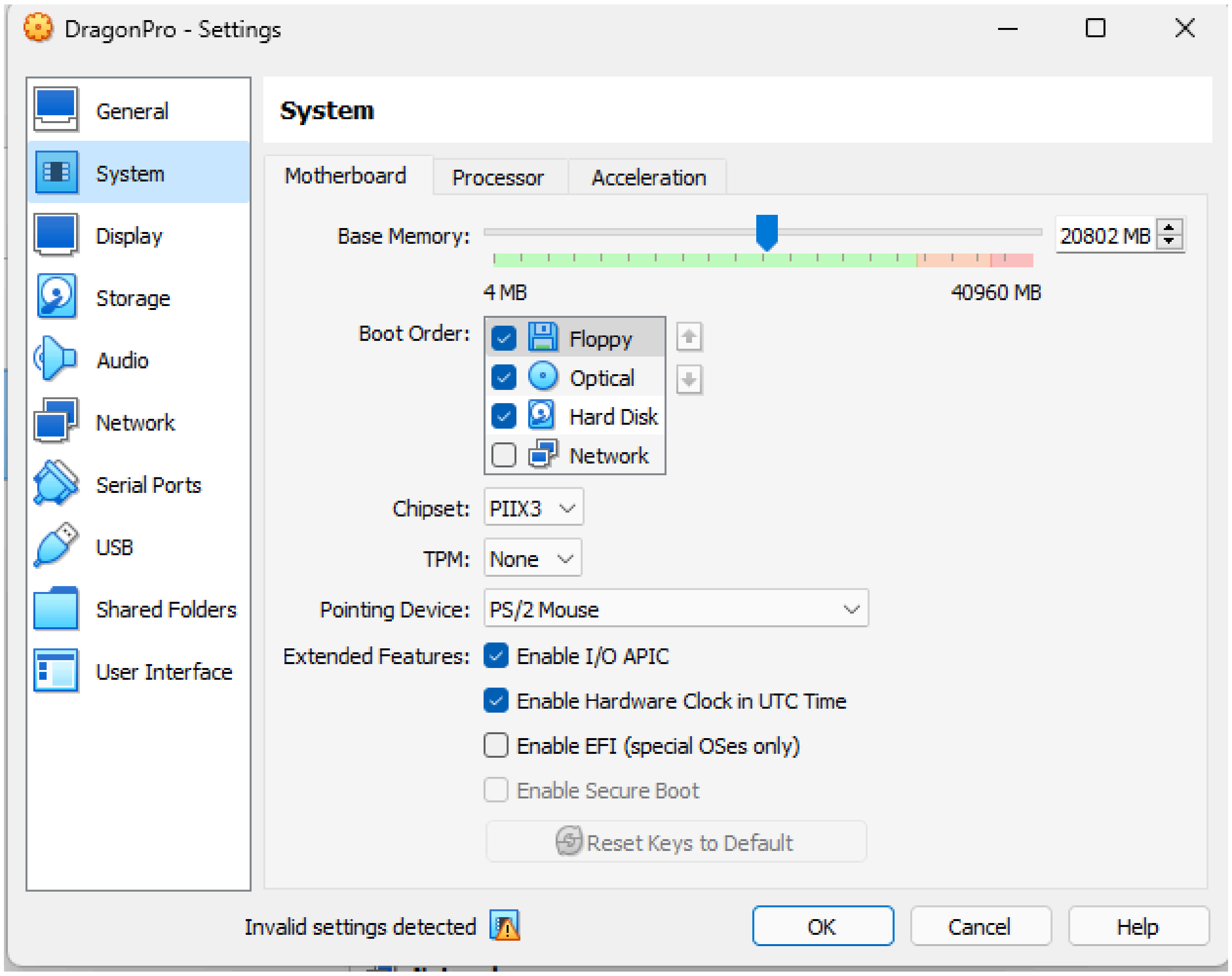
This week we will finish up the last two flowgraphs we worked on last week, go over a few more blocks in a "high level" overview of the QPSK flowgraph, and make sure everyone is able to run their flowgraphs through working hardware.

Deliverables for today:

- Screenshots of AM Receiver
- Demonstration of working receiver either by video or in-class demo
- End of quarter project idea

Latency, GNU Radio Interfacing with LimeSDR

Some of you may have latency or sampling issues when running DragonOS through a virtual machine on your computer. You can try increasing the VM's resources, in some cases this might solve your problem, but the most consistent way is to run your flowgraphs using the LimeSDR on the Raspberry Pi. Follow the steps in last week's quick note about updating LimeSuite and call me over if you are having issues.



Base Memory: 20802 MB (4 MB – 40960 MB)

Boot Order: [✓] Floppy, [✓] Optical, [✓] Hard Disk, [ ] Network

Chipset: PIIX3

TPM: None

Pointing Device: PS/2 Mouse

Extended Features: [✓] Enable I/O APIC, [✓] Enable Hardware Clock in UTC Time, [ ] Enable EFI (special OSes only), [ ] Enable Secure Boot

Invalid settings detected

After powering off your virtual machine, click go to Settings->System and use the motherboard tab to increase the RAM or the processor tab to increase the resources you allocate. This likely will not be as efficient in

Quick Notes on Packets, Payloads, Synchronization

All important block pages:

- Polyphase Clock Sync or Symbol Sync for clock recovery
- Costas Loop for phase, frequency correction
- Linear Equalizer and Adaptive Algorithm
- Constellation Modulator also see constellation object to implement and also constellation decoder and soft decoder

We will quickly go over the QPSK tutorial page on GNU Radio and explain the functions of these blocks. Most projects will implement something that needs one or more of these blocks/concepts

Project Suggestions

Here are some basic requirements for your projects:

1. Make a system that implements both Transmit and Receive. This will be 2 separate flowgraphs.

OR

1. Make a more sophisticated Receive only system that implements more signal processing and/or a more complicated protocol

2. Don't just use GNU Radio examples. They can help or even be the starting point of your project, but you must sufficiently modify and/or "upgrade" them
3. Implement some DSP into your flowgraph, not just "we send a signal, we receive a signal"
4. Thoroughly document the entire process, including failures and/or issues encountered. A great project will be documented in a way that another group could follow their procedure and recreate it.

Tips:

- Building in pieces is your friend. Don't be afraid to implement simple sources and GUI sinks to verify something is working as you think it should before you add on to it.
- Use "Channel Model" blocks to verify end to end transmission before worrying about hardware and power. The channel model can model noise and errors, also using delay blocks can help you test your clock recovery blocks
- For testing and replication, signals can be saved to file sources as "complex" files, and then read from file sources for repeatable testing. This could be useful to anyone making a receive-only or sniffing system, you wouldn't have to create data every time you had to test.
- Audio sinks can be buggy and annoying to work with, especially on the Pi, I highly suggest you use something else in your project.