INTRODUCTION

Purpose
This report documents the development of the Wright State University High Altitude Balloon Software Defined Radio. The project fulfills the requirements for successful completion of EE 482: Senior Design and a Bachelor of Science Degree in Electrical Engineering at Wright State University.

Problem
Our team chose to design a software defined radio to include on the WSU high altitude balloon. Currently, the balloon is using a hardware-based amateur radio communication system which is not configurable to multiple communication protocols. Any significant modifications to the current system require major hardware upgrades which are costly and time consuming. A software defined radio is easily configurable and allows users to change communication protocols through simple programming techniques. This project will allow the communication system on the balloon to be quickly and efficiently customized to each individual launch needs.

Scope
This report documents the details of the project with complete specifications, the design and development of the project with all calculations, all experimental setups and data obtained, results of the project and conclusion.

BACKGROUND
Research
Due to limited experience with software defined radios, the group conducted preliminary research on the current state of the field. The group decided that the GNU Radio toolkit paired with a Universal Software Radio Peripheral (USRP) was the most appealing option to implement our radio considering budget and time constraints.

In order to become proficient in GNU Radio, the team decided to first achieve a digital video broadcast (DVB) using the USRP hosted by a laptop computer. After this task was accomplished, the team would go on to develop an embedded device that could achieve the same result but also meet FAA weight requirements for balloon flight.

The team successfully achieved DVB across two laptop computers using a combination of gstreamer and GNU Radio. The GNU Radio code was a slightly modified version of Alexandru Csete's DVB example[1].

TEST AND EVALUATION
Apparatus
To implement SDR on the high altitude balloon our group acquired a Universal Software Radio Peripheral (USRP), a WBX daughter-board, a low-noise RF amplifier, and a computer-on-module (COM) manufactured by Texas Instruments known as the Beagleboard. Signal processing was accomplished through the GNU Radio toolkit.

The USRP hardware integrates a field programmable gate array (FPGA) with ADC/DAC converters and other components on a single board to form a device well suited for software radio implementation. The RF front-end of the SDR system is provided by a WBX daughter board which connects directly to the USRP. The Beagleboard hosts the USRP by providing a platform on which the software algorithms can execute and process the signal.

USRP-Beagleboard Combo
In addition to the design of the radio, a launch payload package was designed to prepare for flight. This package included the radio, GPS tracking devices, battery, and power bus.

**Procedure**

The group first tested the device's ability to transmit using simple sinusoidal waveforms. Then an attempt was made to emulate the DVB experiment. Before testing could begin, the different hardware architecture of the Beagleboard required some adjustments. First, a lighter operating system, Angstrom Linux, was installed on the board to increase performance. The distribution was chosen due to its speed and its support of GNU Radio software packages. However, it was found that the USRP sink block used in the original experiment was not supported by Angstrom. Another USRP sink block, provided by the Ettus Universal Hardware Driver (UHD) was supported by the distribution, so the code was modified to use this block instead.

**FINDINGS**

**Data**

The device successfully transmitted sinusoidal waveforms, but upon execution of the DVB transmitter, video could not be transmitted from the embedded device.

The amplifier added a significant amount of transmission distance to the device. The original DVB experiment was tested across the research center (no line of sight) and video was received successfully.

**Interpretation**

The FFT plot, provided by the receiving computer, showed evidence of wide-band transmission, so the failure of DVB transmission is thought to be an encoding error. Hardware limitations of the Beagleboard may be contributing to the problem as well. Further testing and research is required to identify the transmission issue.

**CONCLUSION**

**Assessment**

The WSU High Altitude Balloon team is now equipped with an embedded device which has successfully achieved basic transmission. Despite the inability to broadcast digital video at this time, the device still has many possible uses such as tracking with packet radio (APRS). The system could be redesigned for such functions by simply modifying the code.

**Recommendations**

It is suggested that the WSU HIBAL team further research the benefits of the 28 pin expansion port included on the Beagleboard. If it is possible to transfer data faster from the OMAP processor to the USRP through this port then perhaps the transmission issues could be resolved.

**REFERENCES**

GNU Radio Transmitter Python Code

#!/usr/bin/env python
from gnuradio import blks2
from gnuradio import eng_notation
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.gr import firdes
from grc_gnuradio import blks2 as grc_blks2
from optparse import OptionParser
from gnuradio import uhd

class gmsk_tx(gr.top_block):
    def __init__(self):
        gr.top_block.__init__(self, "Video Transmitter (GMSK)")

        self.signal = signal = 32000
        self.samp_rate = samp_rate = 1000000
        self.rfgain = rfgain = 50
        self.freq = freq = 1000000000

        self.blks2_gmsk_mod_0 = blks2.gmsk_mod(
            samples_per_symbol=2,
            bt=0.35,
            verbose=False,
            log=False,
        )

        self.blks2_packet_encoder_0 = grc_blks2.packet_mod_b(grc_blks2.packet_encoder(
            samples_per_symbol=2,
            bits_per_symbol=1,
            access_code="",
            pad_for_usrp=False,
        ),
        payload_length=4000,
    )

        self.gr_file_source_0 = gr.file_source(gr.sizeof_char*1, "/home/root/pythoncode/txfifo.ts", True)
        self.gr_multiply_const_vxx_0 = gr.multiply_const_vcc((signal, ))

        self.usrp_sink = uhd.single_usrp_sink(
            args="serial=46d0ac22",
            type=uhd.io_type_t.COMPLEX_FLOAT32,
            num_channels=1
        )

        self.usrp_sink.set_samp_rate(samp_rate)
        self.usrp_sink.set_center_freq(freq)
        self.usrp_sink.set_gain(rfgain)

        self.connect((self.gr_file_source_0, 0), (self.blks2_packet_encoder_0, 0))
        self.connect((self.blks2_packet_encoder_0, 0), (self.blks2_gmsk_mod_0, 0))
self.connect((self.blks2_gmsk_mod_0, 0), (self.gr_multiply_const_vxx_0, 0))
self.connect((self.gr_multiply_const_vxx_0, 0), (self.usrp_sink, 0))

def set_signal(self, signal):
    self.signal = signal
    self.gr_multiply_const_vxx_0.set_k((self.signal, ))

def set_samp_rate(self, samp_rate):
    self.samp_rate = samp_rate

def set_rfgain(self, rfgain):
    self.rfgain = rfgain
    self.usrp_sink.set_gain(self.rfgain)

def set_freq(self, freq):
    self.freq = freq
    self.usrp_sink.set_frequency(self.freq)

if __name__ == '__main__':
    parser = OptionParser(option_class=eng_option, usage="%prog: [options]"
    (options, args) = parser.parse_args()
    tb = gmsk_tx()
    tb.start()
    raw_input('Press Enter to quit: ')
    tb.stop()