Deliverable 5

Design 2

Team 2

MCU TNC Design

Kaleb Leon – C00094357
Kobe Keopraseuth – C00092349
David Cain – C00043561

November 2nd, 2020

Appendix F

A. Testing Process

We designed our testing tree by back tracking our way through our current development process to decide which subsystems and components we need to be testing. We plan to have our Preliminary Comprehensive Modular Build and Testing Plan (PCMBTP) thoroughly assess our design’s functionality. The design is statistically feasible as stated in previous appendices, so now we must test that physical and digital feasibility. This is done surgically by looking at every component and testing them then moving up to the subsystem made of these components and testing them together. Then, we finally reach the point where all the subsystem come together to test the main system as a whole. If all subsystems and components work separately and then as a subassembly to combine to a full system, the design is then proven to pass, and work as expected.

The first subassembly that needs to be tested are the components of the physical hardware system and circuits. This includes the micro controller itself and the two external circuits. According to our FMEA, if our micro controller does not function then the code has nothing to run on making the project fail before it has even begun. In terms of the micro controller we will need to test anything, and everything being used to accomplish our design. The main portions we use in terms of signal transmission and reception are mostly the cable connections between the radio to TNC and PC to TNC. These transmission lines include: the USB cable which handles serial communication between TNC to PC, the 2.5 mm Audio jack that handles analog signals between the radio to TNC and vice versa, and RS-232 connecter that is a backup digital communication line. Testing the effects of the micro controller signal processing is also very important due to the design needing to meet a certain amount of specifications. We have to test to make sure all of our power consumption, latency, and voltages are under spec for them to work with our design and have it function properly. In addition to that, we need the I/O pins on the micro controller to be fully functioning so that they can communicate with the external circuits. These external circuits are also a main portion of the hardware that we need to test to assure our input signals are how we need to process them and to change modes. The Audio input circuit needs to be tested down to the component level of the amplifier and the filter to assure we are getting the correct audio signal in and out of our design. The Push To Talk circuit is used mainly to switch modes so that our TNC knows when we are transmitting or receiving. This also needs to be tested down to the component level so that we know if will function as needed to allow our design to perform its tasks.

The second subassembly that needs to be tested are the components of the digital software that controls the hardware and data processing. This subassembly is broken down into three major subsystems which all need to be tested down to the component level. They are as follows: Kiss Packet Handling, AX.25 Protocol Formatting, and Frequency Shifted Audio Tone Handling. The Kiss Packet Handling controls how we handle inputs and outputs at the Data Link layer. This is critical for PC to TNC communication. We must take into account and test multiple components of this subsystem. The serial communication line needs to be tested and functional to make sure we can receive and transmit data across the serial bus between PC and TNC. This component needs to also be assessed on the latency of that communication line to meet our specs. In addition, the packet construction following the KISS protocol is a very important process to KISS transmission to PC. Lastly, we need to test the data extraction from this packet to make sure we are able to extract the correct data. Similarly, for the next subsystem of AX.25 Handling we also need to check whether we are extracting the correct data and formatting correctly following the protocol or when we send it off to the DAC the signal will be incorrect when being received by other radios. Lastly, that is where the last subsystem comes into testing. The Frequency Shifted Audio Tone Handling is a critical and complex section of our software, so it needs to be extensively tested. It needs to be able to handle ADC and DAC control which each entail of their own components. If these signals come in incorrectly or out incorrectly then our design is failed and not useful as a product.

It is through this methodology that our team has designed a testing tree with the described components, subsystems, and subassemblies for testing. As testing progresses, we will add more components we overlooked such that we make sure everything is functional in our design. When navigating the tree we will also write up test reports so that all our tests are well documented and noted fixes to our faults. This will ensure confidence in our design.

B. Testing Tree

As shown in our testing tree below, our design system is split into two Subsystems: software and hardware. The Software Subsystem is more complex than the Hardware Subsystem, since most of our project is mainly software based and the hardware's purpose is only to support receiving and transmitting signals from the TNC. In the Software Subassembly, it is broken into three branches that have many supporting leaves that represent different software processes as opposed to the Hardware Subassembly leaves that represent physical component testing. For each subassembly are different tests that we plan to run for each process and physical components. For example, under the hardware subsystem, under each circuit subassembly we will test and ensure each component's nominal value and desired signal output. Components such as resistors will have to be measured to ensure our circuits' outputs. The signal outputted from the physical amplifier circuit will have to be compared to the signal output from spice software. As for the software subsystem, we will validate that each process or subassembly outputs the correct bitstream and frequencies. The bitstreams outputted from the microcontroller have to be outputted in specific sequence. The microcontroller also must output specific frequencies and must be measured to ensure that there are not many errors from the output. Latency will also be tested for serial communication to ensure optimal performance.

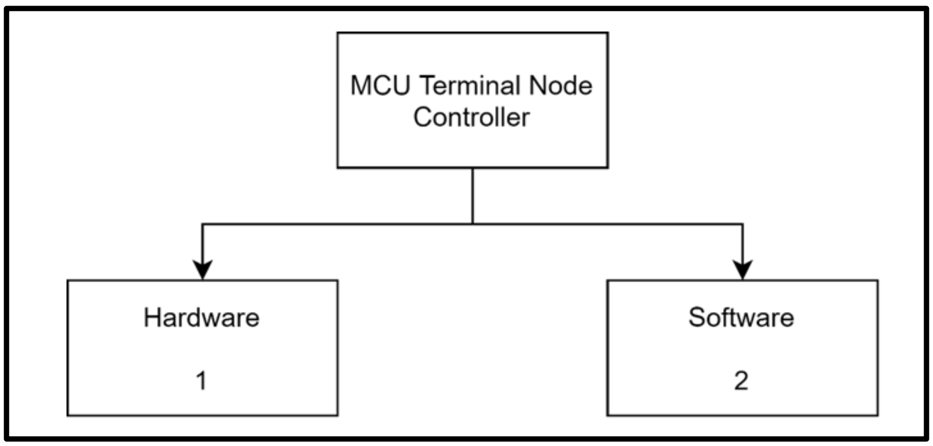


Figure F-1. High Level Comprehensive Testing Tree

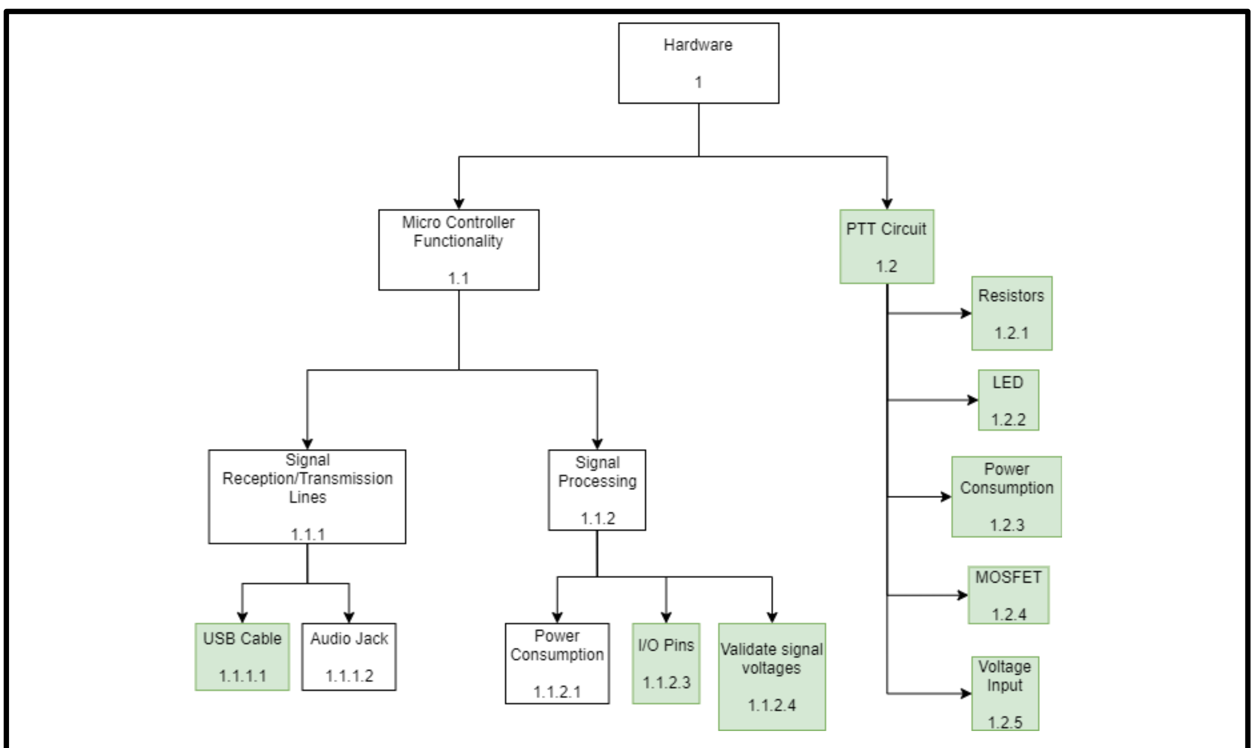


Figure F-2. Hardware Comprehensive Testing Tree

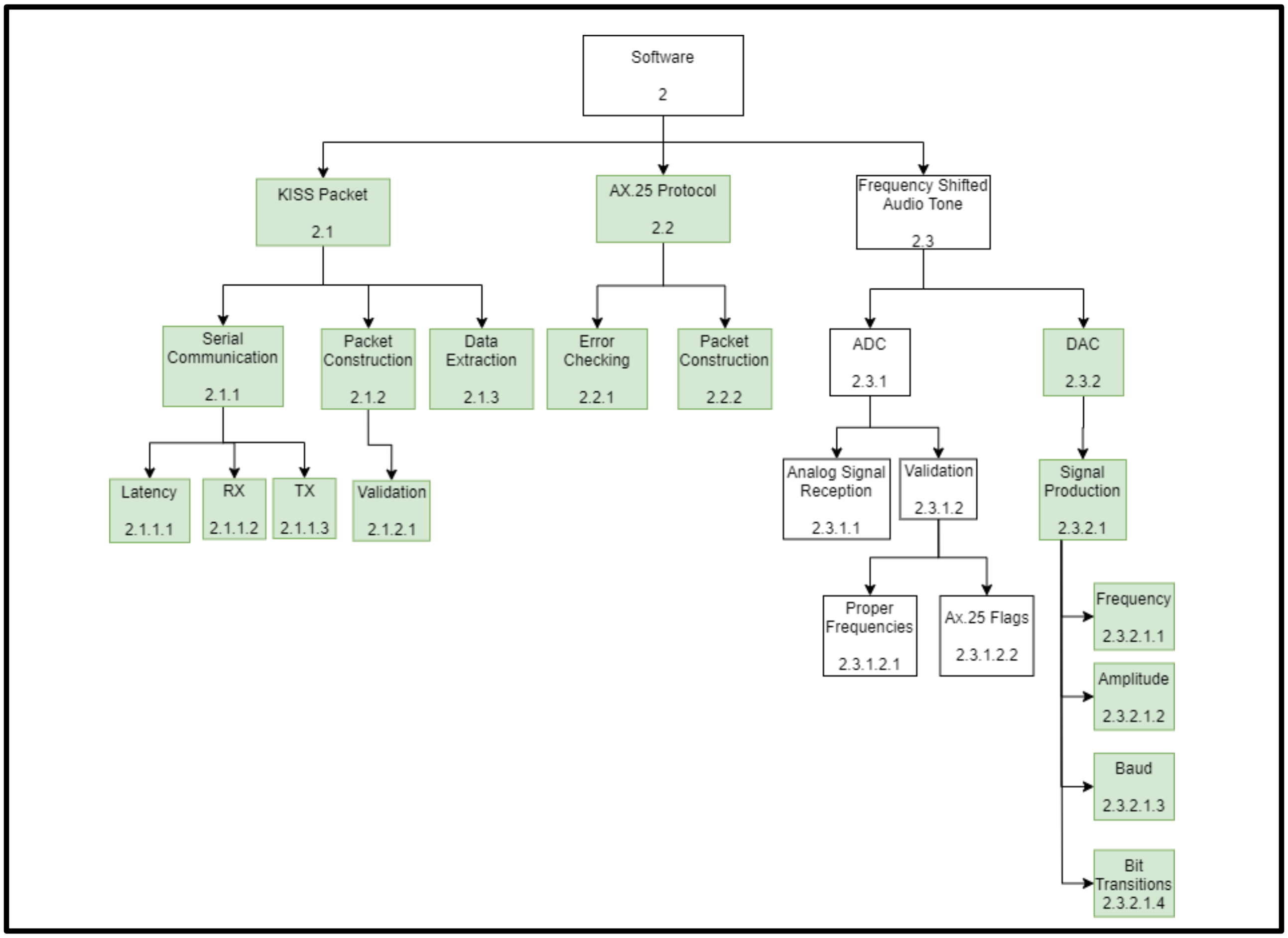


Figure F-3. Software Comprehensive Testing Tree

C. Test Report Template

TNC TESTING FORM(REV1)	
Leaf on the Tree:	
Device Under Test (Testing Tree Number):	
Date:	
Person(s) Conducting Experiment:	
Signature:	
Experiment Purpose:	
Experiment Procedure:	
Equipment Settings/Software Settings (w Revision):	
Testing Diagram/Picture:	
Data Points:	
Pass/Fail:	
Interpreted Notes:	
Recommendations for Modification:	

D. Validate and Verification Plan

1) Hardware

The hardware for this project is split into a couple major components such as: Micro Controller, Circuits, and Connections. The microcontroller is verified as working by testing to see if it will power on and test each pin to make sure it is producing the correct output. The pin testing is done using a known to be working Analog Discovery. Next, the circuits are the PTT circuit and the amplifier circuit. They are tested using the Analog Discovery and simulation. The circuits were each designed and simulated before real life construction to be sure the theory is plausible. The constructed real-life circuit is then tested at each node to make sure we are getting proper input and output voltages/currents. We also test the individual components that make up each circuit by measuring their values and tolerances. Lastly, we can test the interconnections between the micro controller and the circuits to make sure they work in tandem. This is done by analyzing the outputs from the micro controller from the pins connected to the circuit and setting up simple code to view these values in serial to make sure they are accurate. In addition, we test the cable connections between the radio and TNC and PC and TNC. We do this by sending hardcoded packets to make sure they are properly inputted and output through serial and in analog.

2) Software

The software for this project is also split into many major components dedicated to different functions and protocols of our system such as: Kiss Packet Interpretation, Ax.25 Protocol Formatting, DAC, and ADC. The breakdown of how these sections function and work with one another are described in Appendix E Code Breakdown. To validate and test each of these sub processes, debug statements are spread thoroughly throughout the code and comments left to what each function does. At each step in the process we analyze a hardcoded input and validate the produced output. Each stage of the software work in tandem with each other so once each section is able to produce our desired output they are tested together in sections (two at a time). Once all would work in groups with each other and outputs measured correctly whether through serial output or analog measurement then we proceed to testing all together and send a packet through its full course. We send a hardcoded bitstream to create a FSK sine wave and see if we get the correct output KISS packet in serial on the other side. In reverse, we send a KISS packet generated by our Mentor's software and see if it generates an accurate FSK sine wave up to spec and containing the correct data.

E. Workmanship on Design

Plan before going into experiment:

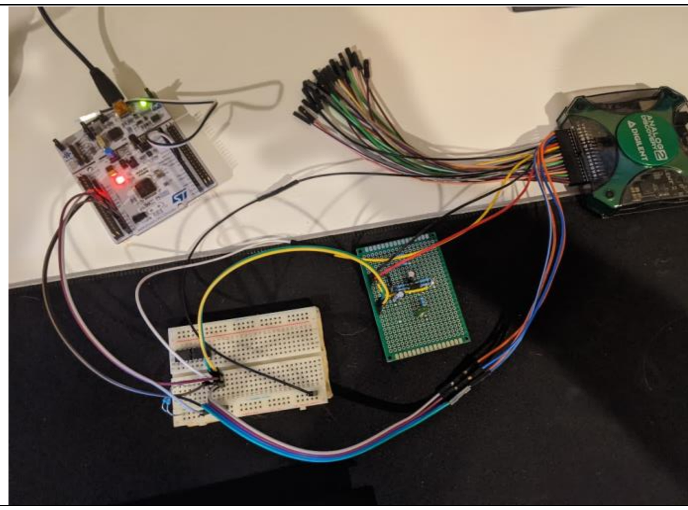
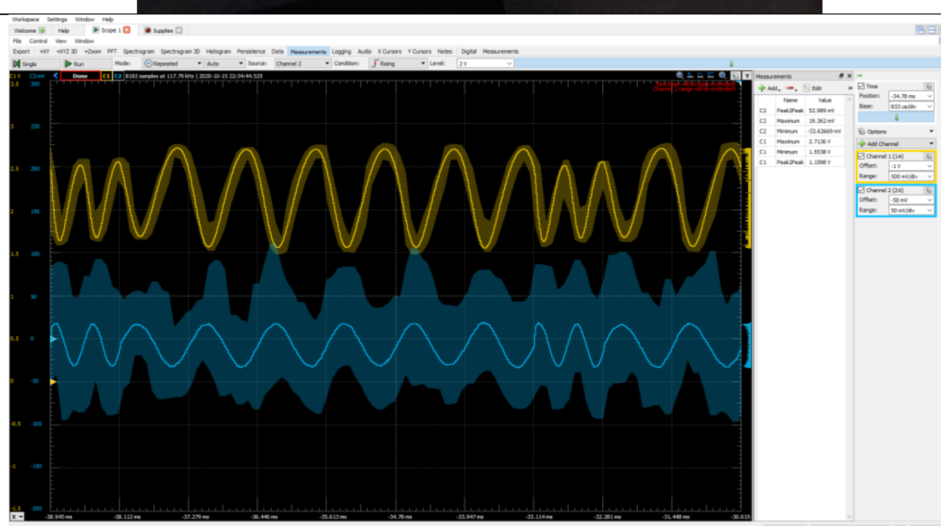
1. What is needed to be done
2. The code or circuit being worked on
3. The code functionality, problem, circuit, and output needed to be tested or created
4. The testing procedure and data collection method
5. Expected results
6. If results not met in one sitting setup a time to revisit

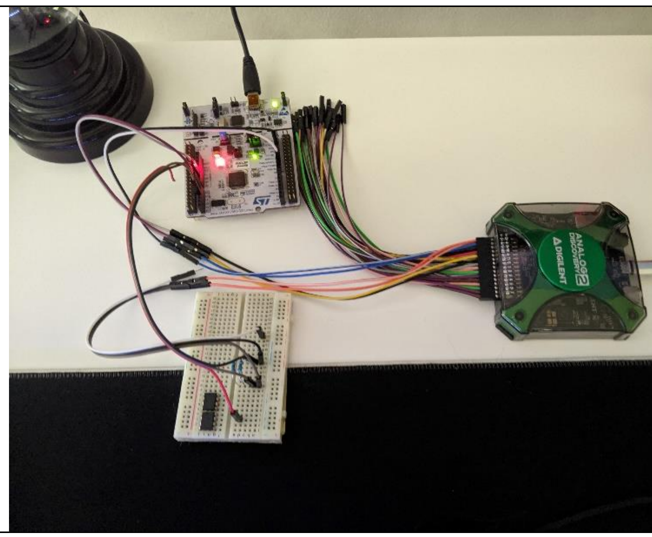
Per experiment we would perform the steps above. Each of us would take action in handling different portions of the design but we follow this same path no matter what task. We code and test things as we go to make sure we are producing accurate results. It is similar to how an AGILE workflow works. We work and test at the same time to make sure we are accurate with each step of the process. This style of work keeps the project going smoothly and meeting small milestones with each experiment completed. Once a main section of the design is accomplished, it is then documented, and all our experiment notes come together to form the paper.

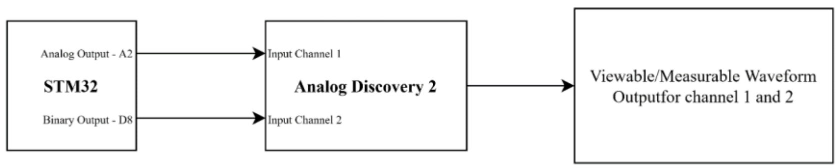
F. Final Presentation Demo

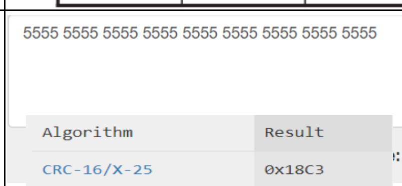
The final presentation demo will consist of two TNC systems communicating with one another. The first system will consist of our TNC, a radio, and one of our laptops. The second system will consist of an off the shelf TNC used in CAPE currently as the satellite TNC, a similar radio, and another PC for sending KISS packets. Each PC/radio combo will have their own callsign as well. These two systems should be able to send packets between one another. For example, our laptop will use our mentor's software to generate a KISS packet and convert it to AX.25 and then to an analog signal. This packet will then be sent over the air for the other radio to receive and break down for the other PC to read. In addition, we will show it in the opposite direction as well. The CAPE TNC system will send a signal over the air and we will receive it and break it down for all PC. This will show off our packet formatting system and our ability to communicate with an already manufactured and in use TNC setup.

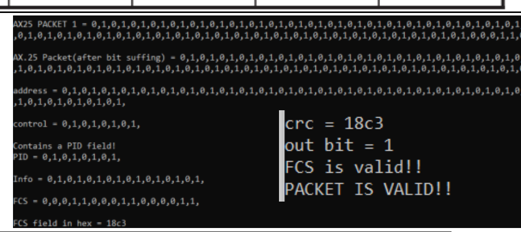
Completed Test Reports

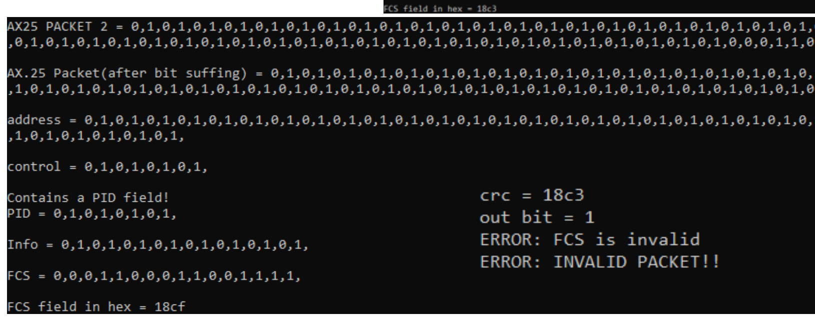
TNC Testing Form (REV1)	
Leaf on the Tree	Amplifier
Device Under Test (Testing Tree Number):	1.3.1.1
Date:	10/15/2020
Person(s) Conducting Experiment:	David Cain, Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure the amplifier output can trigger the external interrupt on the STM32 for reading incoming AFSK waveform frequency components.
Experiment Procedure:	Using waveform generator from Analog Discovery 2, input a 50mV ptp AFSK waveform to amplifier circuit, checking readings reported by microcontroller on serial port.
Equipment Settings / Software Settings (w Revision):	Micro controller is set to receiving mode Using WaveForms software(version 3.12.2), output generated AFSK signal
Testing Diagram / Picture:	
Data Points:	

TNC Testing Form (REV1)	
Leaf on the Tree	Amplitude
Device Under Test (Testing Tree Number):	2.3.1.2.2.1
Date:	10/4/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to measure waveform output voltage. Part of our specifications it to be capable of sinking 400mV(ptp) into 1k.
Experiment Procedure:	To verify amplitude, the analog output will be connected to a 1k load and measured. In this case, 2 x 2k resistors will be wired in parallel on a bread board, creating 1k of resistance across the terminals.
Equipment Settings / Software Settings (w Revision):	The Digilent will be set to record the maximum value of the waveform measured. For general insight, the RMS and minimum were also recorded
Testing Diagram / Picture:	
Data Points:	Maximum: 399.19 mV RMS: 137.11 mV Minimum: -1.43 mV
Pass / Fail:	Pass
Interpreted Notes:	The waveform satisfies the 400mV requirement. Potentially some feedback could be used to tune the output during runtime, but this is not necessarily required.
Recommendations for Modifications:	None, currently

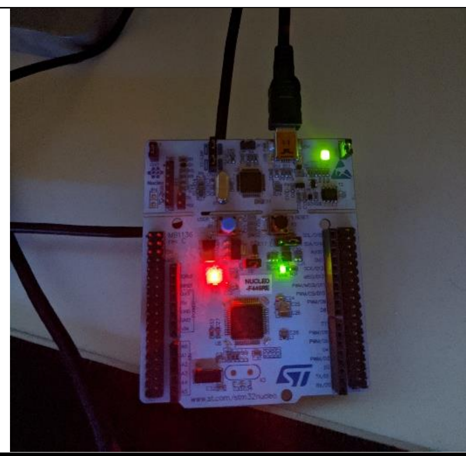
TNC Testing Form (REV1)	
Leaf on the Tree	Baud
Device Under Test (Testing Tree Number):	2.3.1.2.3.1
Date:	10/4/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to measure and ensure the number of signaling events per second (or baud rate) is correctly established as 1200Hz
Experiment Procedure:	To verify the baud rate, a diagnostic signal will be enabled in software to output the current transmission bit value represented in binary. This binary wave form can easily have baud rate measured.
Equipment Settings / Software Settings (w Revision):	Analog Discovery 2 input channel 1 and 2 will be connected to the STM32 output pins D8(PA9) and A2(PA4)
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	Waveform is sustaining a baud rate of 1200Hz. This was tested with multiple wave forms but easily viewed with alternating bit pattern.
Recommendations for Modifications:	None

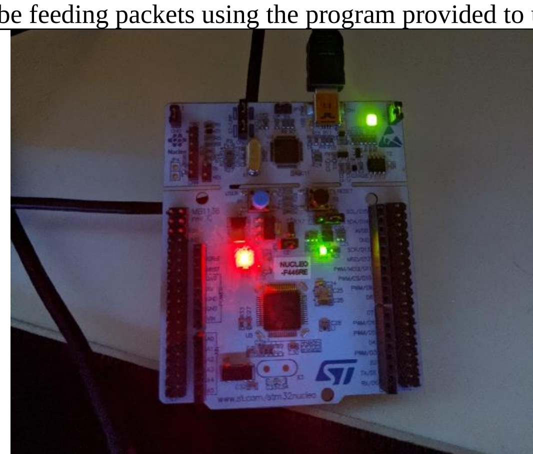
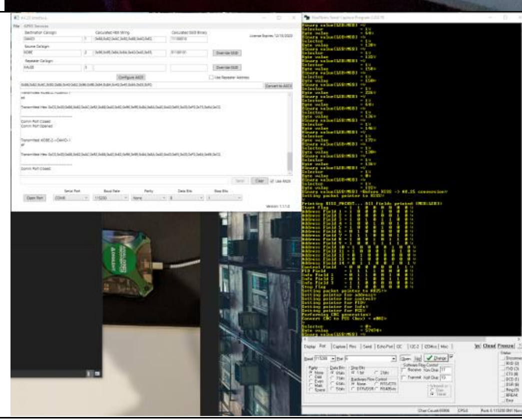
<table>
<tr><td colspan="2">TNC Testing Form (REV1)</td></tr>
<tr><td>Leaf on the Tree</td><td>Error Checking</td></tr>
<tr><td>Device Under Test (Testing Tree Number):</td><td>2.2.1</td></tr>
<tr><td>Date:</td><td>10/10/2020</td></tr>
<tr><td>Person(s) Conducting Experiment:</td><td>Kobe Keopraseuth</td></tr>
<tr><td>Signature:</td><td></td></tr>
<tr><td>Experiment Purpose:</td><td>The purpose of this experiment is to verify an inputted packet has the correct FCS field, by performing a crc check on the other given subfields (Address, Control, PID, Info) as shown in the testing diagram.</td></tr>
<tr><td>Experiment Procedure:</td><td>To verify that it correctly verifies the FCS field, I made 2 testing array inputs, with a size greater than 120, which is the minimum. One array contains a correct FCS field and the other does not. Both arrays contain an input of 0x555555555555555555555555555555555555 (36 fives), excluding the FCS field. According the crc calculator, this input should have a crc output of 0x18c3. To verify that the generated crc is valid with the given input array, an online crc calculator was used.</td></tr>
<tr><td>Equipment Settings / Software Settings (w Revision):</td><td>Code will be implemented in Code Blocks IDE and it print out the input array, array after bit stuffing, the subfields obtained from the input array, FCS field in hexadecimal, crc calculation, and the result of whether the FCS field is valid or not.</td></tr>
<tr><td>Testing Diagram / Picture:</td><td>
<table>
<tr><td>Flag</td><td>Address</td><td>Control</td><td>Info</td><td>FCS</td><td>Flag</td></tr>
<tr><td>01111110</td><td>112/224 Bits</td><td>8/16 Bits</td><td>N*8 Bits</td><td>16 Bits</td><td>01111110</td></tr>
</table>
Figure 3.1a. U and S frame construction.
<table>
<tr><td>Flag</td><td>Address</td><td>Control</td><td>PID</td><td>Info</td><td>FCS</td><td>Flag</td></tr>
<tr><td>01111110</td><td>112/224 Bits</td><td>8/16 Bits</td><td>8 Bits</td><td>N*8 Bits</td><td>16 Bits</td><td>01111110</td></tr>
</table>
</td></tr>
<tr><td>Data Points:</td><td>


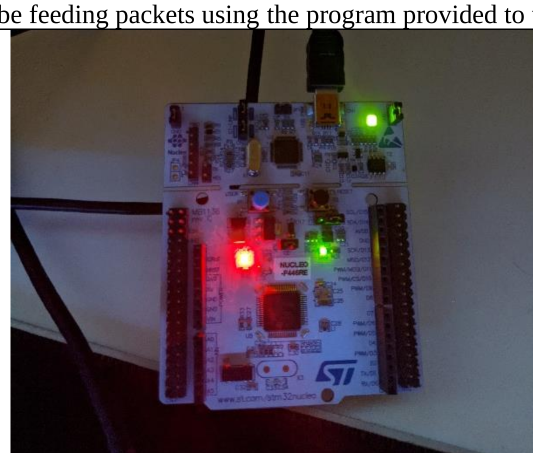
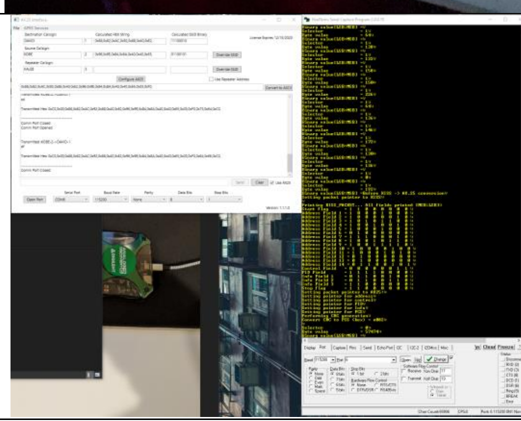


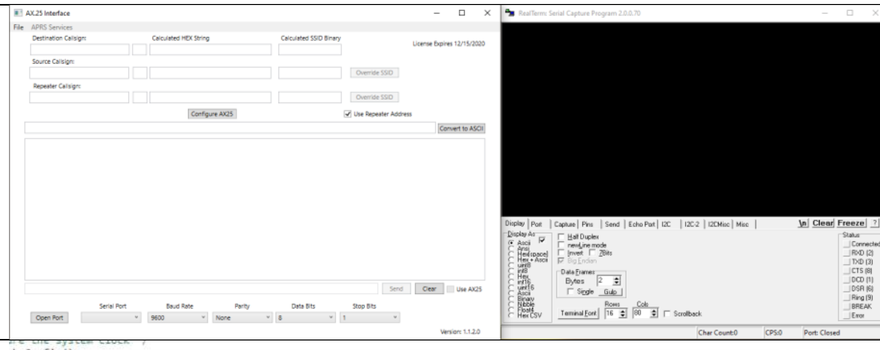
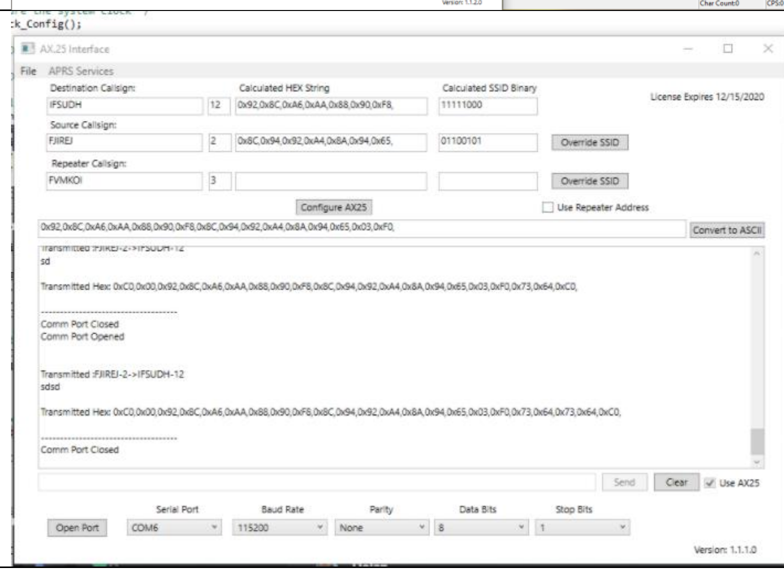


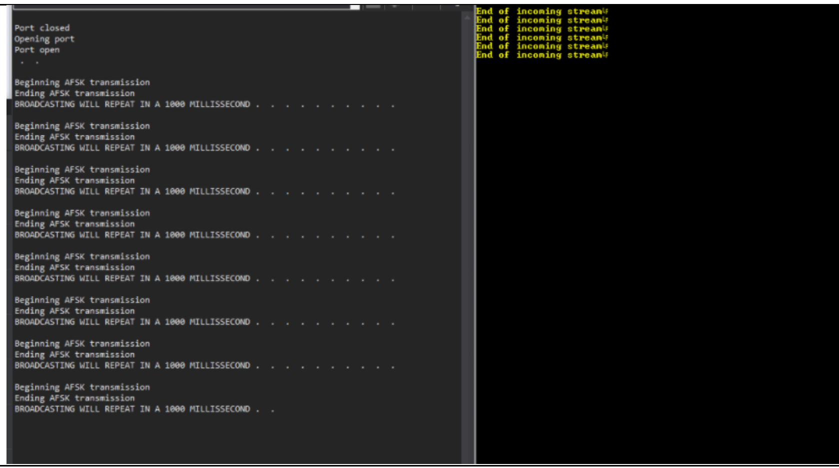
</td></tr>
<tr><td>Pass / Fail:</td><td>PASS</td></tr>
<tr><td>Interpreted Notes:</td><td>Code does verify the FCS field correctly. This was tested with different given FCS fields and different bits for the input array.</td></tr>
<tr><td>Recommendations for Modifications:</td><td>None</td></tr>
</table>

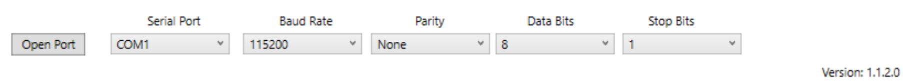
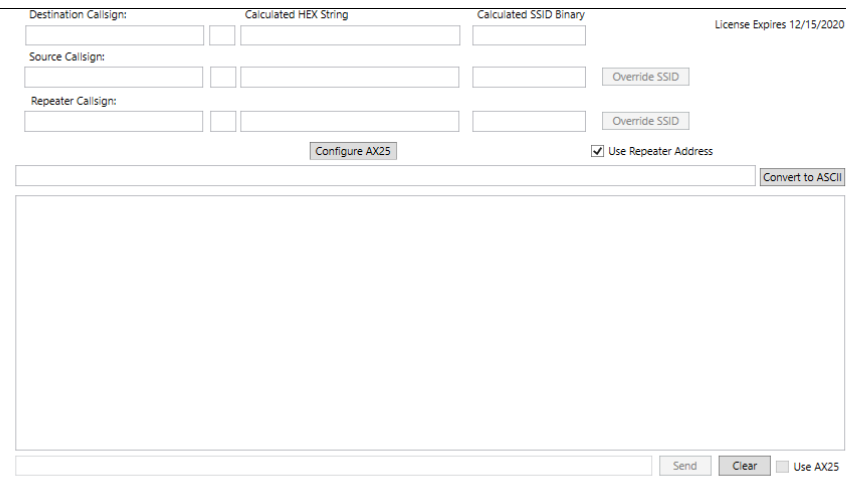
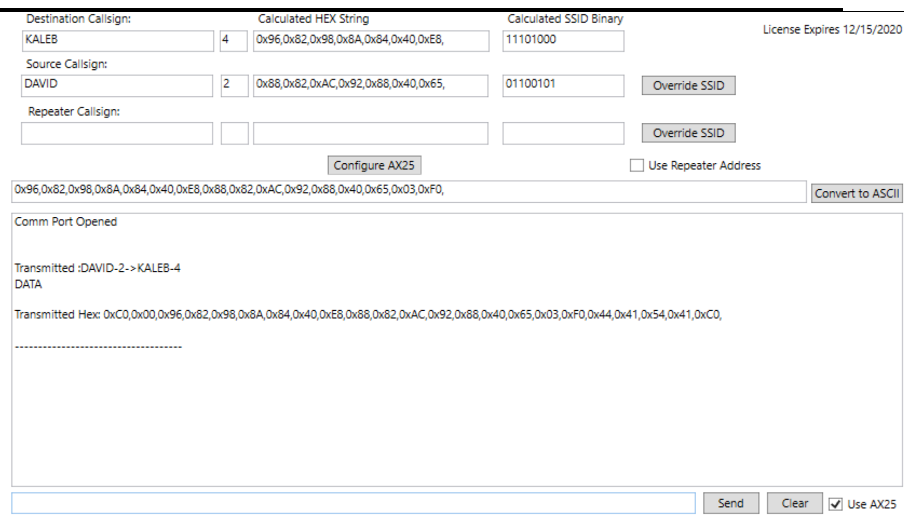
TNC Testing Form (REV1)	
Leaf on the Tree	Latency
Device Under Test (Testing Tree Number):	2.1.1.1
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that the latency of the microcontroller when transmitting data is within an acceptable time. A specified time was not given for the project, but this is an important consideration from a user perspective.
Experiment Procedure:	I will setup a software timer that begins the moment the controller receives a KISS packet over UART. This timer will run until the controller begins to repeatedly output the same message in broadcasting mode.
Equipment Settings / Software Settings (w Revision):	Working entirely within our own software. This is not complete but it is mostly finished.
Testing Diagram / Picture:	
Data Points:	Total transmission time was: 12032 Beginning AFSK transmission Ending AFSK transmission BROADCASTING WILL REPEAT IN A 2000 MILLISSECOND . . . .
Pass / Fail:	Pass
Interpreted Notes:	The device takes ~1.2ms and this is deemed acceptable.
Recommendations for Modifications:	None.

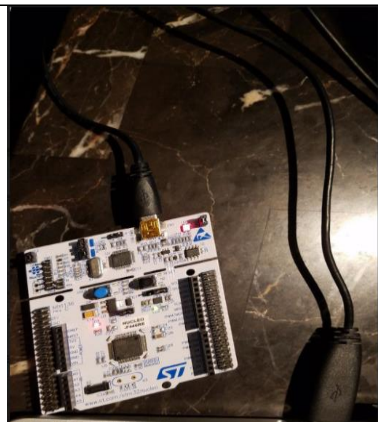
TNC Testing Form (REV1)	
Leaf on the Tree	RX
Device Under Test (Testing Tree Number):	2.1.1.2
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure the microcontroller is receiving data over UART.
Experiment Procedure:	To test the data connection over UART, the microcontroller will be given a packet, and this will then printed over serial again.
Equipment Settings / Software Settings (w Revision):	The microcontroller will be running our most current version of software and I will be viewing the output using a serial monitor program called RealTerm 2.0.0.70. I will be feeding packets using the program provided to us by Rizwan.
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	The data being read over the serial monitor seems acceptable for the given packet.
Recommendations for Modifications:	None.

TNC Testing Form (REV1)	
Leaf on the Tree	RX
Device Under Test (Testing Tree Number):	2.1.1.3
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure the microcontroller is transmitting data over UART properly.
Experiment Procedure:	To test the data connection over UART, the microcontroller will be given a packet, and this will then printed over serial again. Effectively I am testing bother transmitting and receiving in the same step. If any failure occurs I would simplify this test further.
Equipment Settings / Software Settings (w Revision):	The microcontroller will be running our most current version of software and I will be viewing the output using a serial monitor program called RealTerm 2.0.0.70. I will be feeding packets using the program provided to us by Rizwan.
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	The data being transmitted over the serial monitor seems acceptable for the given packet.
Recommendations for Modifications:	None.

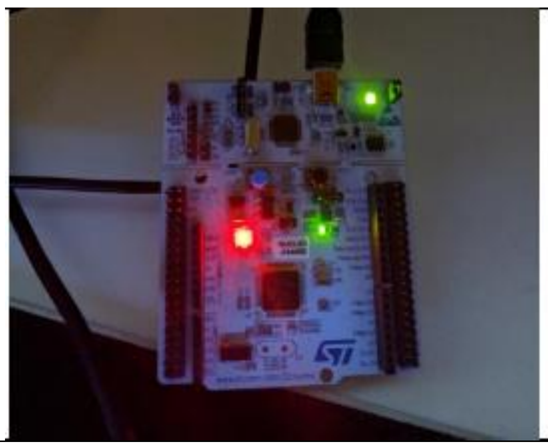
<table>
<tr><td colspan="2">TNC Testing Form (REV1)</td></tr>
<tr><td>Leaf on the Tree</td><td>Serial Communication</td></tr>
<tr><td>Device Under Test (Testing Tree Number):</td><td>2.1.1</td></tr>
<tr><td>Date:</td><td>11/1/20</td></tr>
<tr><td>Person(s) Conducting Experiment:</td><td>Kobe Keopraseuth, Kaleb Leon, David Cain</td></tr>
<tr><td>Signature:</td><td></td></tr>
<tr><td>Experiment Purpose:</td><td>To test whether we can send Kiss packets over serial.</td></tr>
<tr><td>Experiment Procedure:</td><td>Set up mentor’s software to send KISS packet to TNC and then the TNC will send it back over serial and we will see it using realterm</td></tr>
<tr><td>Equipment Settings / Software Settings (w Revision):</td><td>Set to communicate over COM1 and store the KISS packet in a buffer and send that buffer over serial using UART</td></tr>
<tr><td>Testing Diagram / Picture:</td><td>
</td></tr>
<tr><td>Data Points:</td><td>
</td></tr>
</table>

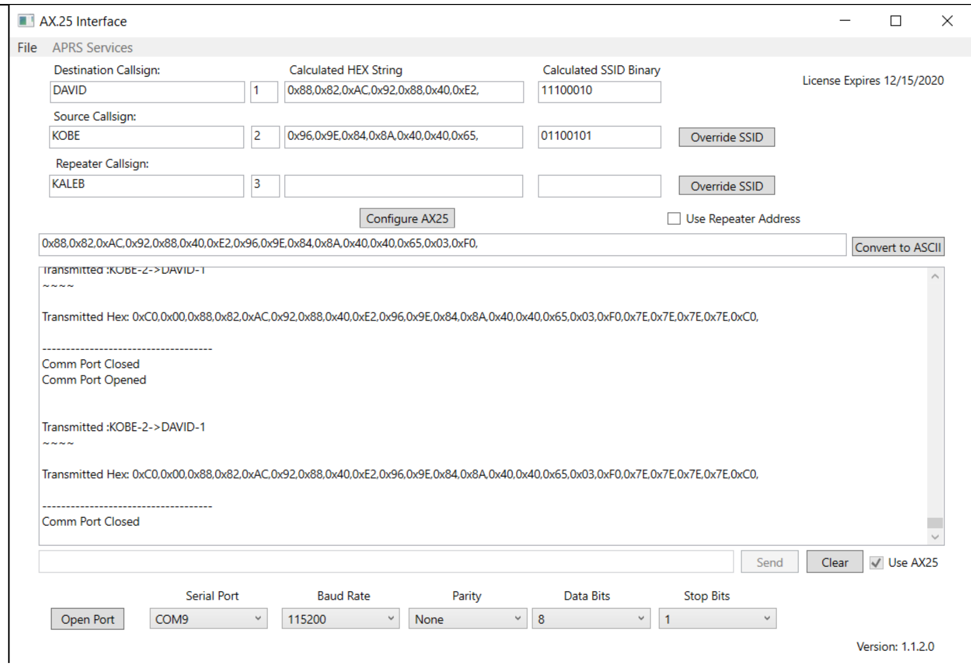
	
Pass / Fail:	PASS
Interpreted Notes:	Communicates as expected over UART serial.
Recommendations for Modifications:	N/A

TNC Testing Form (REV1)	
Leaf on the Tree	Packet Construction
Device Under Test (Testing Tree Number):	2.1.2
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	Testing our mentor's software to make sure it produces the correct KISS packets
Experiment Procedure:	Entering callsigns and data so that it can be made into a packet and then checking that packet to make sure it has the right contents
Equipment Settings / Software Settings (w Revision):	
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	PASS
Interpreted Notes:	The software produces the desired valid Kiss packet for testing on our software.
Recommendations for Modifications:	N/A

TNC Testing Form (REV1)	
Leaf on the Tree	Data Extraction
Device Under Test (Testing Tree Number):	2.1.3.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that we were able to extract the received KISS packet over UART and translate that packet into a binary bit stream.
Experiment Procedure:	We will send a packet from our KISS packet generator software and output the binary conversion (done by our microcontroller) through UART which will be displayed on a serial monitor.
Equipment Settings / Software Settings (w Revision):	We will be using Rizwan's given software to generate and send the KISS packet and visual studio's serial monitor to output the data extraction done by our microcontroller.
Testing Diagram / Picture:	 **Setup** Transmitted Hex: 0xC0,0x00,0x88,0x82,0xAC,0x92,0x88,0x40,0xE2,0x96,0x9E,0x84,0x8A,0x40,0x40,0x65,0x03,0xF0,0x7E,0x7E,0x7E,0x7E,0xC0, **input packet**

Data Points:	Start flag = 1 1 0 0 0 0 0 0 Address Field 1 = 1 0 0 0 1 0 0 0 Address Field 2 = 1 0 0 0 0 0 1 0 Address Field 3 = 1 0 1 0 1 1 0 0 Address Field 4 = 1 0 0 1 0 0 1 0 Address Field 5 = 1 0 0 0 1 0 0 0 Address Field 6 = 0 1 0 0 0 0 0 0 Address Field 7 = 1 1 1 0 0 0 1 0 Address Field 8 = 1 0 0 1 0 1 1 0 Address Field 9 = 1 0 0 1 1 1 1 0 Address Field 10 = 1 0 0 0 0 1 0 0 Address Field 11 = 1 0 0 0 1 0 1 0 Address Field 12 = 0 1 0 0 0 0 0 0 Address Field 13 = 0 1 0 0 0 0 0 0 Address Field 14 = 0 1 1 0 0 1 0 1 Control Field = 0 0 0 0 0 0 1 1 PID Field = 1 1 1 1 0 0 0 0 Info Field 1 = 0 1 1 1 1 1 1 0 Info Field 2 = 0 1 1 1 1 1 1 0 Info Field 3 = 0 1 1 1 1 1 1 0 Info Field 4 = 0 1 1 1 1 1 1 0 Stop flag = 1 1 0 0 0 0 0 0
Pass / Fail:	Pass
Interpreted Notes:	The binary data being displayed on the serial monitor correctly corresponds to the hexadecimal values sent from Rizwan’s software. The second hex value was not displayed, because it is not needed.
Recommendations for Modifications:	None

TNC Testing Form (REV1)	
Leaf on the Tree	AX.25 Protocol
Device Under Test (Testing Tree Number):	2.2
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment of this experiment is to verify that our microcontroller can take in a KISS packet and format the AX.25 Packet Correctly.
Experiment Procedure:	Take in a KISS packet from computer and display the fields of the AX.25 packet. We will also show the calculated crc to show that the KISS packet for properly extracted.
Equipment Settings / Software Settings (w Revision):	Use Rizwan’s software to send a KISS packet and display the AX.25 packet on serial monitor.
Testing Diagram / Picture:	
Data Points:	

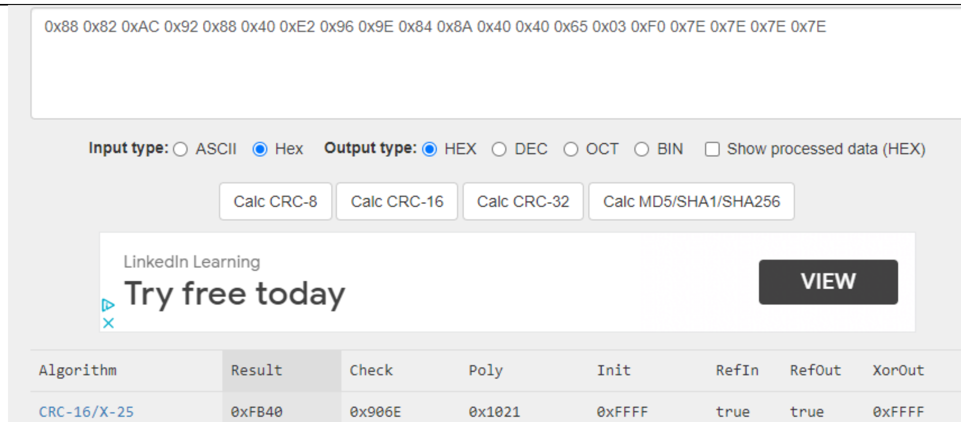


Rizwan’s software for transmitting KISS packets

```
Start flag       = 1  1  0  0  0  0  0  0
Address Field 1 = 1  0  0  0  1  0  0  0
Address Field 2 = 1  0  0  0  0  0  1  0
Address Field 3 = 1  0  1  0  1  1  0  0
Address Field 4 = 1  0  0  1  0  0  1  0
Address Field 5 = 1  0  0  0  1  0  0  0
Address Field 6 = 0  1  0  0  0  0  0  0
Address Field 7 = 1  1  1  0  0  0  1  0
Address Field 8 = 1  0  0  1  0  1  1  0
Address Field 9 = 1  0  0  1  1  1  1  0
Address Field 10 = 1  0  0  0  0  1  0  0
Address Field 11 = 1  0  0  0  1  0  1  0
Address Field 12 = 0  1  0  0  0  0  0  0
Address Field 13 = 0  1  0  0  0  0  0  0
Address Field 14 = 0  1  1  0  0  1  0  1
Control Field    = 0  0  0  0  0  0  1  1
PID Field        = 1  1  1  1  0  0  0  0
Info Field 1     = 0  1  1  1  1  1  1  0
Info Field 2     = 0  1  1  1  1  1  1  0
Info Field 3     = 0  1  1  1  1  1  1  0
Info Field 4     = 0  1  1  1  1  1  1  0
Stop flag        = 1  1  0  0  0  0  0  0
```

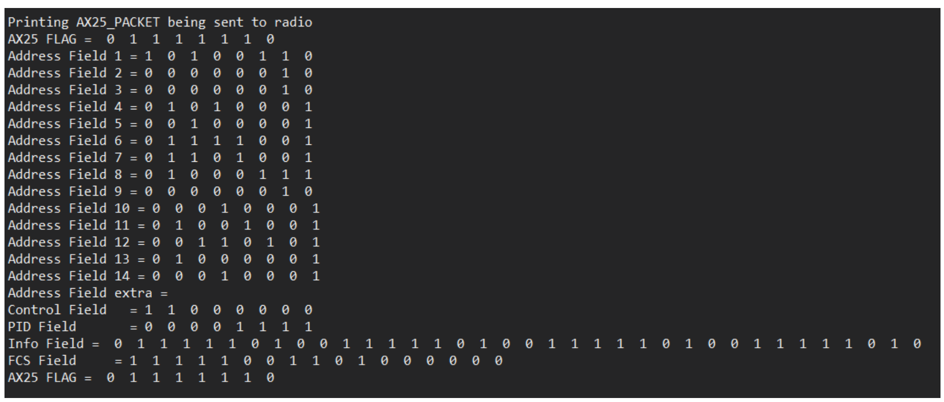
Bitstream output of received KISS packet

<table>
<tr>
<td></td>
<td>



<u>Online crc calculation for the KISS packet</u>

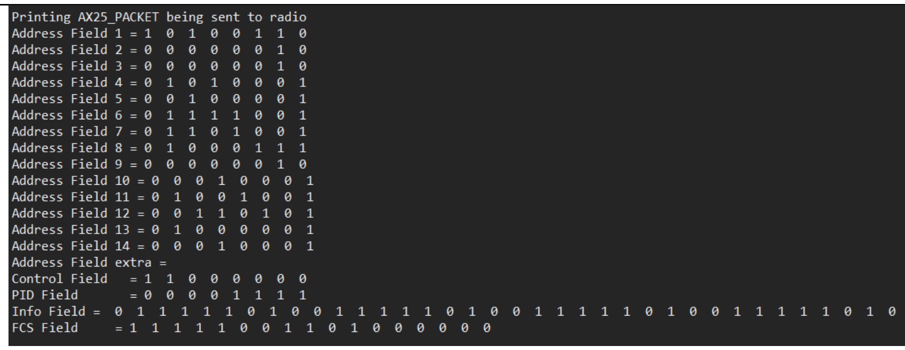
<u>Microcontroller's crc calculation for KISS packet</u>



<u>AX.25's bit sequence when sent to radio</u>

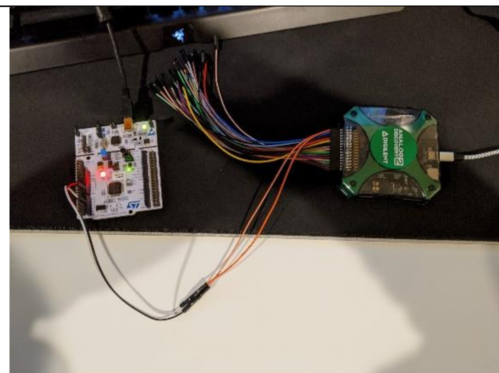
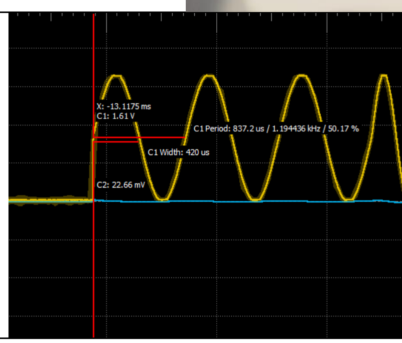
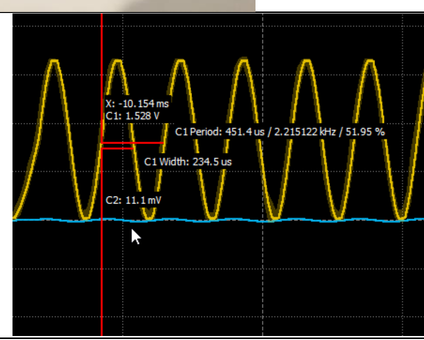
</td>
</tr>
<tr>
<td>Pass / Fail:</td>
<td>Pass</td>
</tr>
<tr>
<td>Interpreted Notes:</td>
<td>Our microcontroller can properly extract a KISS packet and format it into AX.25, along with correctly calculating the crc for the FCS field. It is also able to place the bits to be sent to over radio in the correct order.</td>
</tr>
<tr>
<td>Recommendations for Modifications:</td>
<td>None</td>
</tr>
</table>

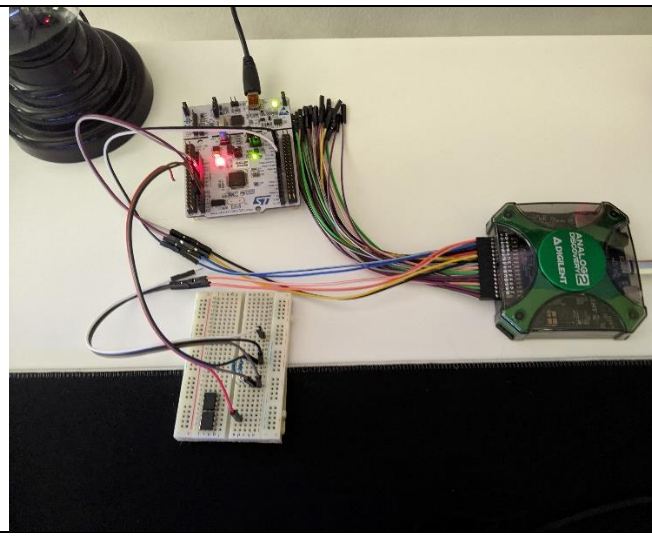
<table>
<tr><td colspan="2">TNC Testing Form (REV1)</td></tr>
<tr><td>Leaf on the Tree</td><td>Validation</td></tr>
<tr><td>Device Under Test (Testing Tree Number):</td><td>2.2.2.1.1</td></tr>
<tr><td>Date:</td><td>11/1/20</td></tr>
<tr><td>Person(s) Conducting Experiment:</td><td>Kobe Keopraseuth</td></tr>
<tr><td>Signature:</td><td></td></tr>
<tr><td>Experiment Purpose:</td><td>The purpose of this experiment is to validate that the microcontroller will output the AX.25 (excluding the flags) in correct order.</td></tr>
<tr><td>Experiment Procedure:</td><td>I will display how the AX.25 packet will be sent to the radio, using a serial monitor.</td></tr>
<tr><td>Equipment Settings / Software Settings (w Revision):</td><td>We will be using Rizwan’s software to send a KISS packet over UART, and we will be using visual studio’s serial monitor to display the AX.25 packet’s bit sequence.</td></tr>
<tr><td>Testing Diagram / Picture:</td><td></td></tr>
<tr><td>Data Points:</td><td><pre>
Start flag   = 1 1 0 0 0 0 0 0
Address Field 1 = 1 0 0 0 1 0 0 0
Address Field 2 = 1 0 0 0 0 0 1 0
Address Field 3 = 1 0 1 0 1 1 0 0
Address Field 4 = 1 0 0 1 0 0 1 0
Address Field 5 = 1 0 0 0 1 0 0 0
Address Field 6 = 0 1 0 0 0 0 0 0
Address Field 7 = 1 1 1 0 0 0 1 0
Address Field 8 = 1 0 0 1 0 1 1 0
Address Field 9 = 1 0 0 1 1 1 1 0
Address Field 10 = 1 0 0 0 0 1 0 0
Address Field 11 = 1 0 0 0 1 0 1 0
Address Field 12 = 0 1 0 0 0 0 0 0
Address Field 13 = 0 1 0 0 0 0 0 0
Address Field 14 = 0 1 1 0 0 1 0 1
Control Field   = 0 0 0 0 0 0 1 1
PID Field   = 1 1 1 1 0 0 0 0
Info Field 1   = 0 1 1 1 1 1 1 0
Info Field 2   = 0 1 1 1 1 1 1 0
Info Field 3   = 0 1 1 1 1 1 1 0
Info Field 4   = 0 1 1 1 1 1 1 0
Stop flag   = 1 1 0 0 0 0 0 0
</pre>
<u>Bitnary Bit stream of packet</u></td></tr>
</table>

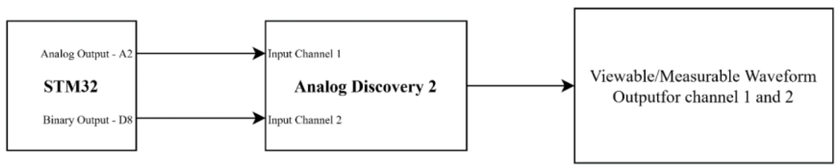
	 Printing AX25_PACKET being sent to radio Address Field 1 = 1 0 1 0 0 1 1 0 Address Field 2 = 0 0 0 0 0 0 1 0 Address Field 3 = 0 0 0 0 0 0 1 0 Address Field 4 = 0 1 0 1 0 0 0 1 Address Field 5 = 0 0 1 0 0 0 0 1 Address Field 6 = 0 1 1 1 1 0 0 1 Address Field 7 = 0 1 1 0 1 0 0 1 Address Field 8 = 0 1 0 0 0 1 1 1 Address Field 9 = 0 0 0 0 0 0 1 0 Address Field 10 = 0 0 0 1 0 0 0 1 Address Field 11 = 0 1 0 0 1 0 0 1 Address Field 12 = 0 0 1 1 0 1 0 1 Address Field 13 = 0 1 0 0 0 0 0 1 Address Field 14 = 0 0 0 1 0 0 0 1 Address Field extra = Control Field = 1 1 0 0 0 0 0 0 PID Field = 0 0 0 0 1 1 1 1 Info Field = 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 FCS Field = 1 1 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 **Binary Bitstream of how AX.25 packet will be sent to radio**
Pass / Fail:	Pass
Interpreted Notes:	As shown on the serial monitor, the AX.25's bits are in the correct order. FCS field is sent MSB first and other fields are sent LSB first. After 5 contiguous ones then a bit stuffed zero is added after.
Recommendations for Modifications:	None

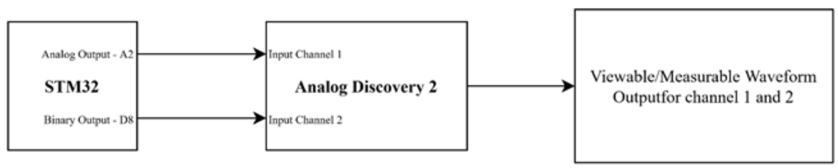
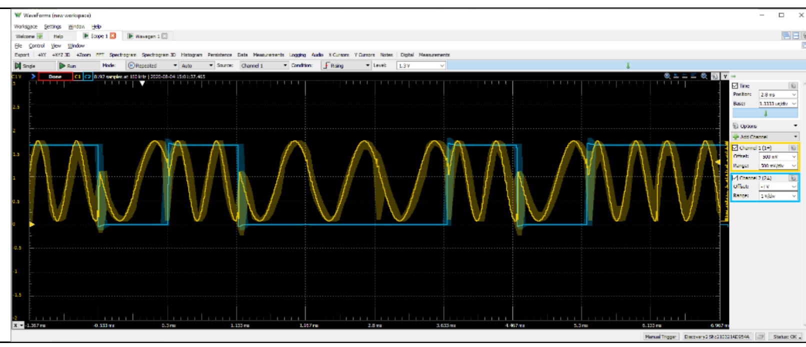
TNC Testing Form (REV1)	
Leaf on the Tree	Packet Construction
Device Under Test (Testing Tree Number):	2.2.2.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to validate that the microcontroller will output the AX.25 (excluding the flags) in correct order.
Experiment Procedure:	I will display how the AX.25 packet will be sent to the radio, using a serial monitor.
Equipment Settings / Software Settings (w Revision):	We will be using Rizwan’s software to send a KISS packet over UART, and we will be using visual studio’s serial monitor to display the AX.25 packet’s bit sequence.
Testing Diagram / Picture:	
Data Points:	Start flag = 1 1 0 0 0 0 0 0 Address Field 1 = 1 0 0 0 1 0 0 0 Address Field 2 = 1 0 0 0 0 0 1 0 Address Field 3 = 1 0 1 0 1 1 0 0 Address Field 4 = 1 0 0 1 0 0 1 0 Address Field 5 = 1 0 0 0 1 0 0 0 Address Field 6 = 0 1 0 0 0 0 0 0 Address Field 7 = 1 1 1 0 0 0 1 0 Address Field 8 = 1 0 0 1 0 1 1 0 Address Field 9 = 1 0 0 1 1 1 1 0 Address Field 10 = 1 0 0 0 0 1 0 0 Address Field 11 = 1 0 0 0 1 0 1 0 Address Field 12 = 0 1 0 0 0 0 0 0 Address Field 13 = 0 1 0 0 0 0 0 0 Address Field 14 = 0 1 1 0 0 1 0 1 Control Field = 0 0 0 0 0 0 1 1 PID Field = 1 1 1 1 0 0 0 0 Info Field 1 = 0 1 1 1 1 1 1 0 Info Field 2 = 0 1 1 1 1 1 1 0 Info Field 3 = 0 1 1 1 1 1 1 0 Info Field 4 = 0 1 1 1 1 1 1 0 Stop flag = 1 1 0 0 0 0 0 0 **Bitnary Bit stream of packet**

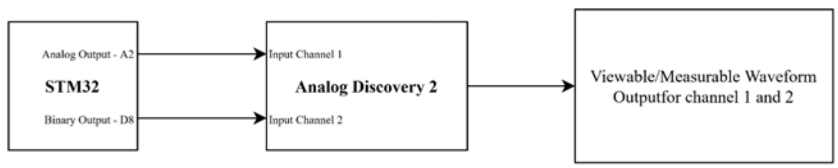
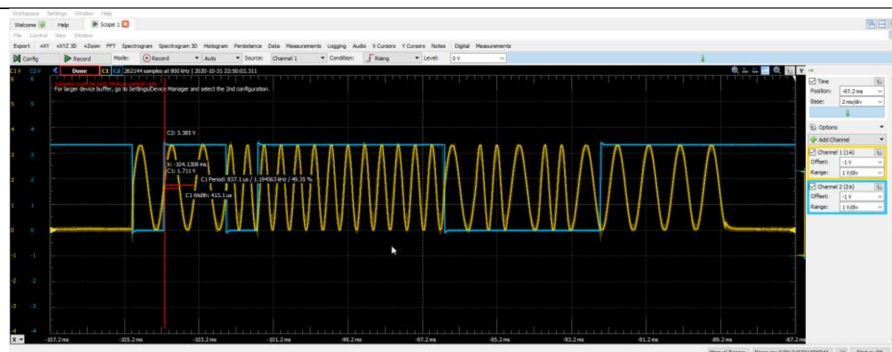
	Printing AX25_PACKET being sent to radio Address Field 1 = 1 0 1 0 0 1 1 0 Address Field 2 = 0 0 0 0 0 0 1 0 Address Field 3 = 0 0 0 0 0 0 1 0 Address Field 4 = 0 1 0 1 0 0 0 1 Address Field 5 = 0 0 1 0 0 0 0 1 Address Field 6 = 0 1 1 1 1 0 0 1 Address Field 7 = 0 1 1 0 1 0 0 1 Address Field 8 = 0 1 0 0 0 1 1 1 Address Field 9 = 0 0 0 0 0 0 1 0 Address Field 10 = 0 0 0 1 0 0 0 1 Address Field 11 = 0 1 0 0 1 0 0 1 Address Field 12 = 0 0 1 1 0 1 0 1 Address Field 13 = 0 1 0 0 0 0 0 1 Address Field 14 = 0 0 0 1 0 0 0 1 Address Field extra = Control Field = 1 1 0 0 0 0 0 0 PID Field = 0 0 0 0 1 1 1 1 Info Field = 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 FCS Field = 1 1 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 **Binary Bitstream of how AX.25 packet will be sent to radio**
Pass / Fail:	Pass
Interpreted Notes:	As shown on the serial monitor, the AX.25’s bits are in the correct order. FCS field is sent MSB first and other fields are sent LSB first. After 5 contiguous ones then a bit stuffed zero is added after.
Recommendations for Modifications:	None

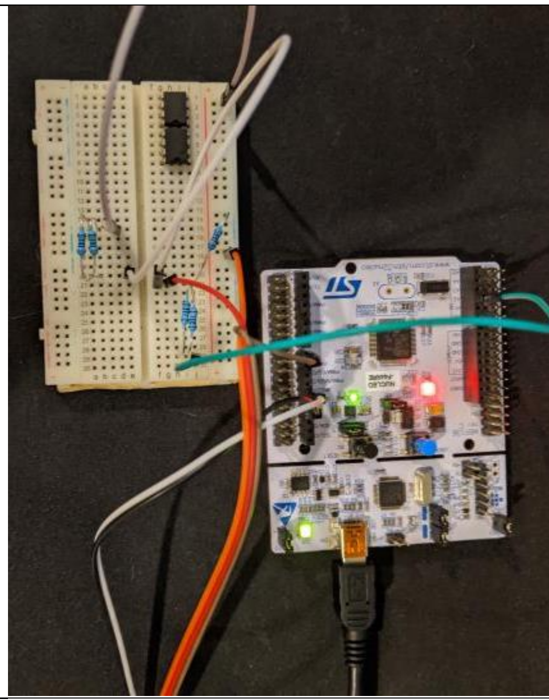
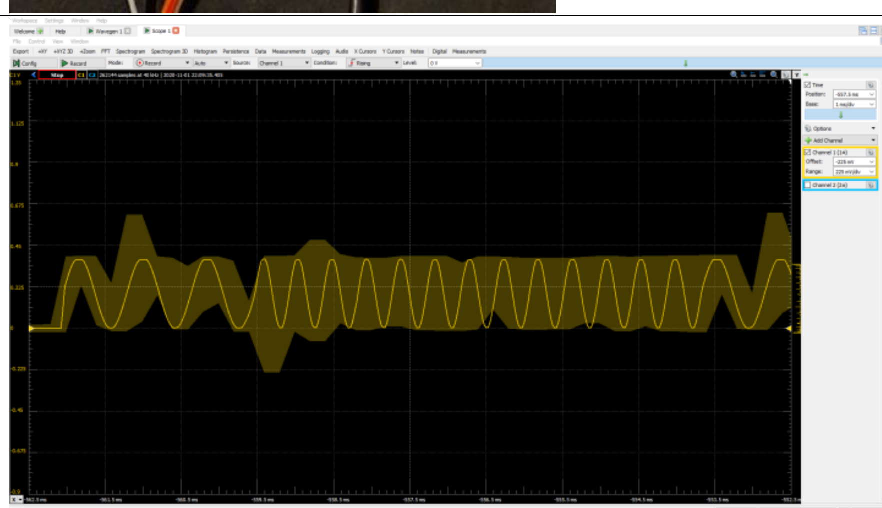
TNC Testing Form (REV1)	
Leaf on the Tree	Frequency
Device Under Test (Testing Tree Number):	2.3.2.1.1
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	Ensure that the output frequencies of the microcontroller are 1200 and 2200 when expected.
Experiment Procedure:	Force the TNC into a debugging broadcast mode, then use the Digilent discovery 2 to measure the waveform frequency at many points.
Equipment Settings / Software Settings (w Revision):	The Digilent will be set to record the frequency of the waveform measured. For general insight, the RMS and minimum were also recorded
Testing Diagram / Picture:	
Data Points:	 
Pass / Fail:	Pass
Interpreted Notes:	As shown in the datapoints, the output waveforms are 1200Hz and 2200Hz as expected.
Recommendations for Modifications:	None.

TNC Testing Form (REV1)	
Leaf on the Tree	Amplitude
Device Under Test (Testing Tree Number):	2.3.2.1.2.1
Date:	10/4/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to measure waveform output voltage. Part of our specifications it to be capable of sinking 400mV(ptp) into 1k.
Experiment Procedure:	To verify amplitude, the analog output will be connected to a 1k load and measured. In this case, 2 x 2k resistors will be wired in parallel on a bread board, creating 1k of resistance across the terminals.
Equipment Settings / Software Settings (w Revision):	The Digilent will be set to record the maximum value of the waveform measured. For general insight, the RMS and minimum were also recorded
Testing Diagram / Picture:	
Data Points:	Maximum: 399.19 mV RMS: 137.11 mV Minimum: -1.43 mV
Pass / Fail:	Pass
Interpreted Notes:	The waveform satisfies the 400mV requirement. Potentially some feedback could be used to tune the output during runtime, but this is not necessarily required.
Recommendations for Modifications:	None, currently

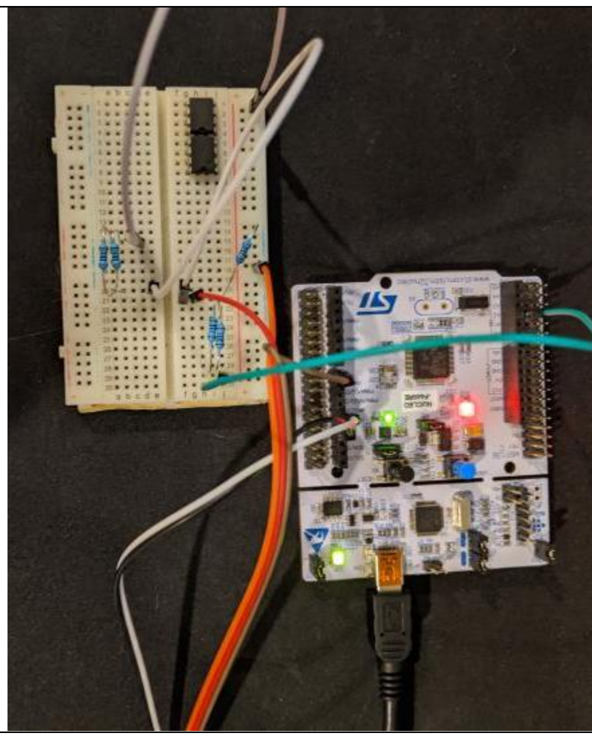
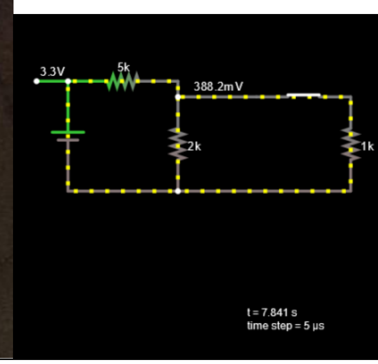
TNC Testing Form (REV1)	
Leaf on the Tree	Baud
Device Under Test (Testing Tree Number):	2.3.2.1.3.1
Date:	10/4/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to measure and ensure the number of signaling events per second (or baud rate) is correctly established as 1200Hz
Experiment Procedure:	To verify the baud rate, a diagnostic signal will be enabled in software to output the current transmission bit value represented in binary. This binary wave form can easily have baud rate measured.
Equipment Settings / Software Settings (w Revision):	Analog Discovery 2 input channel 1 and 2 will be connected to the STM32 output pins D8(PA9) and A2(PA4)
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	Waveform is sustaining a baud rate of 1200Hz. This was tested with multiple wave forms but easily viewed with alternating bit pattern.
Recommendations for Modifications:	None

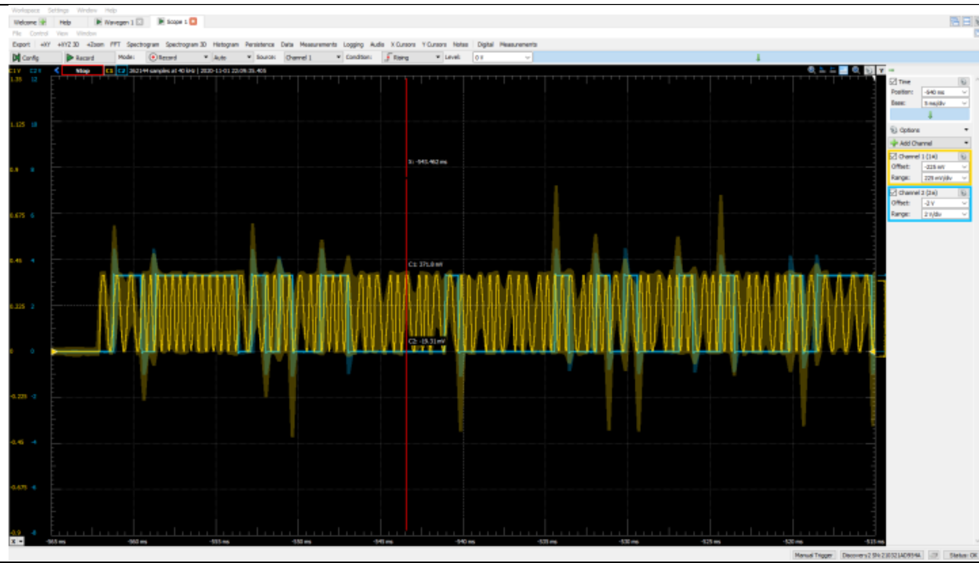
TNC Testing Form (REV1)	
Leaf on the Tree	Bit Transitions
Device Under Test (Testing Tree Number):	2.3.2.1.4
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that the waveforms of our output do not suffer due to bit transitions.
Experiment Procedure:	Force the TNC into a debugging broadcast mode, then use the Digilent discovery 2 to measure the waveform frequency at many points.
Equipment Settings / Software Settings (w Revision):	The Digilent will be set to record the waveform and an optical inspection will be used.
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Fail
Interpreted Notes:	There is an obvious phase shift when the bits are transitioning.
Recommendations for Modifications:	Correct code that generates the output to calculate the next expected starting point of each bit.

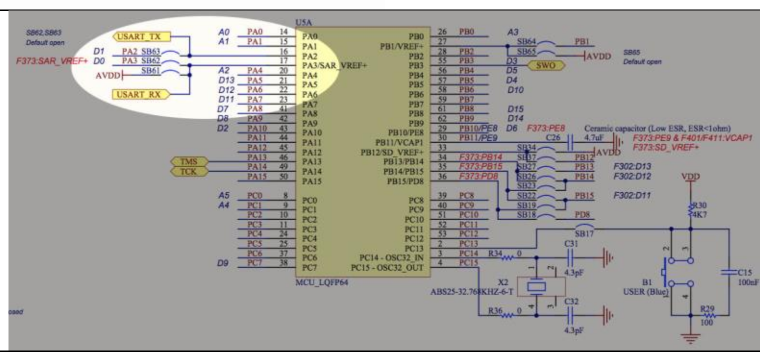
TNC Testing Form (REV1)	
Leaf on the Tree	Bit Transitions
Device Under Test (Testing Tree Number):	2.3.2.1.4
Date:	10/31/2020
Person(s) Conducting Experiment:	David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that the waveforms of our output do not suffer due to bit transitions.
Experiment Procedure:	Force the TNC into a debugging broadcast mode, then use the Digilent discovery 2 to measure the waveform frequency at many points.
Equipment Settings / Software Settings (w Revision):	The Digilent will be set to record the waveform and an optical inspection will be used.
Testing Diagram / Picture:	
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	The waveform is very continuous. You will notice the change in frequency when the digital value is 0. This is due to the NRZI encoding scheme, but the phase is continuous.
Recommendations for Modifications:	None.

TNC Testing Form (REV1)	
Leaf on the Tree	Signal Production
Device Under Test (Testing Tree Number):	2.3.2.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	To show we can produce a basic signal following the coded frequency, amplitude, and baud rate.
Experiment Procedure:	Hard coded parameters for our signal, then produced by our Nucleo. It will be read and judged using the analog discovery.
Equipment Settings / Software Settings (w Revision):	Frequency shift between 1200Hz and 2200Hz Amplitude of 400mV peak to peak Baud rate of 2200
Testing Diagram / Picture:	
Data Points:	

Pass / Fail:	Pass
Interpreted Notes:	Produces a valid signal
Recommendations for Modifications:	N/A

TNC Testing Form (REV1)	
Leaf on the Tree	DAC
Device Under Test (Testing Tree Number):	2.3.2
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	Test the entire DAC subsystem with an actual AX.25 packet actually used in our project.
Experiment Procedure:	Feed a known accurate AX.25 packet then use code to covert that into an analog output signal.
Equipment Settings / Software Settings (w Revision):	Current code base for DAC Sinewave.
Testing Diagram / Picture:	 

Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	Meets the accuracy we required of the output analog signal.
Recommendations for Modifications:	N/A

TNC Testing Form (REV1)	
Leaf on the Tree	1.1.1.1
Device Under Test (Testing Tree Number):	USB Cable
Date:	10/31/2020
Person(s) Conducting Experiment:	Kaleb Leon
Signature:	Kaleb L
Experiment Purpose:	Functionality of USB-B mini works in communicating with the PC over serial.
Experiment Procedure:	Plug Nucleo into PC via the USB-B mini cable and run some code to do serial communication and show we can upload code via this cable as well.
Equipment Settings / Software Settings (w Revision):	Nucleo setup on table connected to USB port on PC using the USB-B mini cable Software Settings: *HARDWARE INIT FILE* (code below), *UART CONFIG* (code below), *MAIN CODE* (code below)
Testing Diagram / Picture:	

HARDWARE INIT FILE

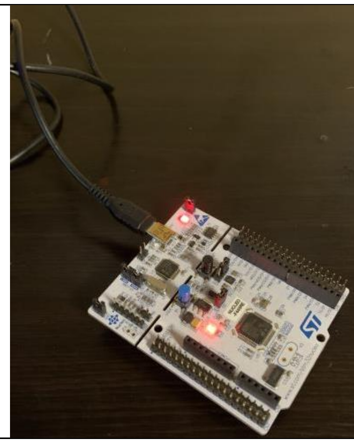
```
void
configure_system_clock(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;

    __PWR_CLK_ENABLE();

    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = 6;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 16;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
    RCC_OscInitStruct.PLL.PLLQ = 7;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_SYSCLK|RCC_CLOCKTYPE_PCLK1;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2);
}
```

UART CONFIG

```
UART_HandleTypeDef huart2;
...
void MX_USART2_UART_Init(void)
{
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    HAL_UART_Init(&huart2);
}
```

MAIN CODE

```
int main(int argc, char* argv[])
{
  char *msg = "Hello Nucleo Fun!\n\r";

  HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), 0xFFFF);
  while(1);
}
```

<table>
<tr><td></td><td></td></tr>
<tr><td>Data Points:</td><td>Serial Output

Type the escape character followed by C to get back,
or followed by ? to see other options.

Hello Nucleo Fun!

Uploading Code

Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a08060.srec
File : ST-LINK_GDB_server_a08060.srec
Size : 20172 Bytes
Address : 0x08000000

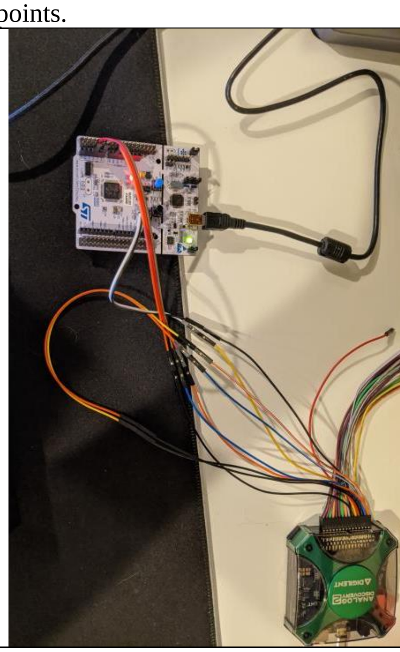
Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 1]
Download in Progress:

File download complete
Time elapsed during download operation: 00:00:00.708

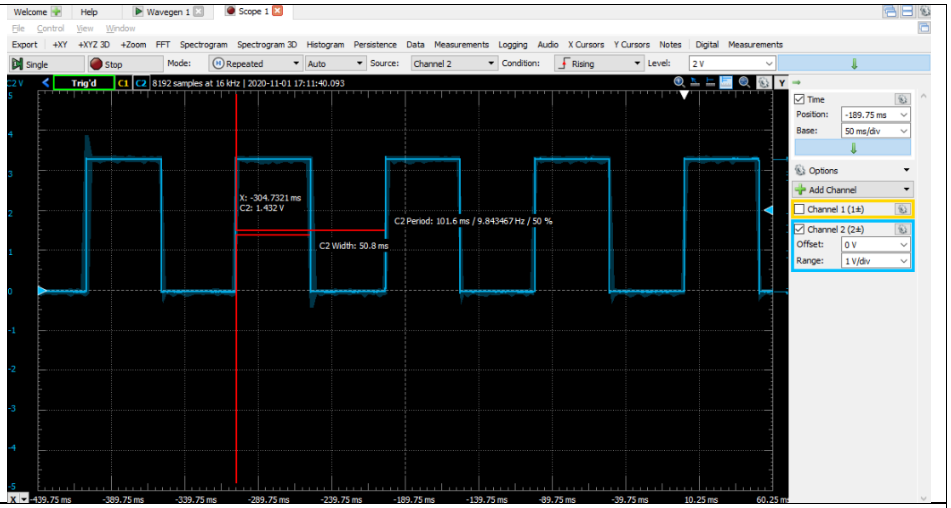
Verifying ...

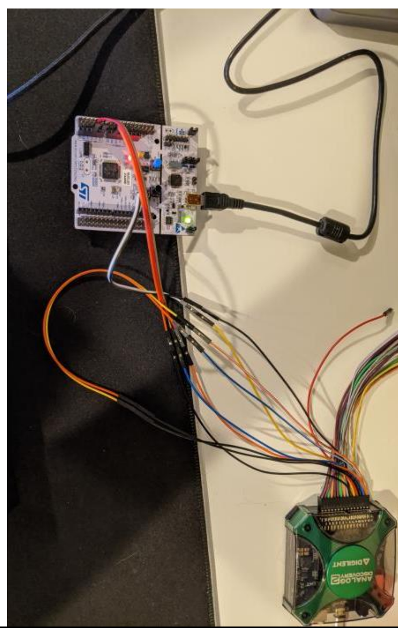
Download verified successfully

Debugger connection lost.
Shutting down...</td></tr>
<tr><td>Pass / Fail:</td><td>PASS</td></tr>
<tr><td>Interpreted Notes:</td><td>The Nucleo seems to be communicated fine with the PC over USB-B mini</td></tr>
<tr><td>Recommendations for Modifications:</td><td>N/A</td></tr>
</table>

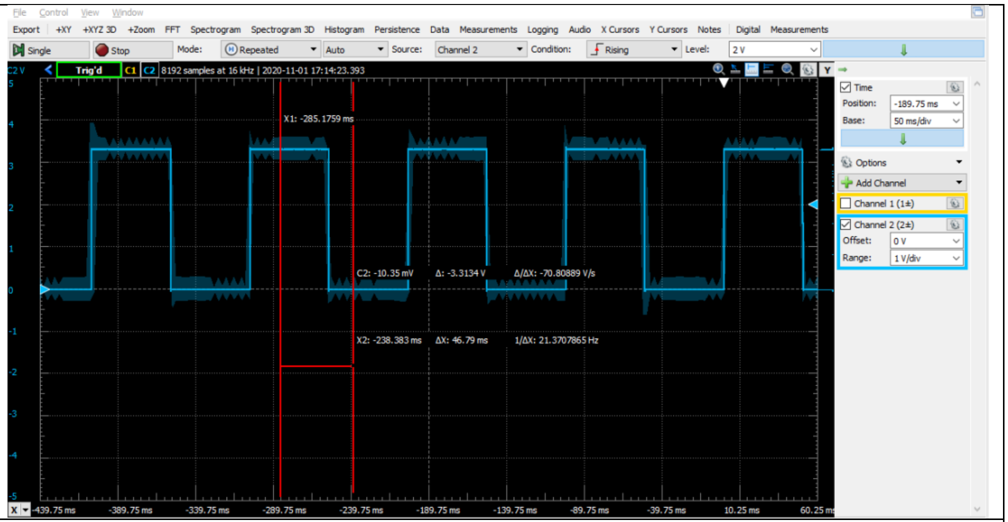
TNC Testing Form (REV1)	
Leaf on the Tree	I/O Pins
Device Under Test (Testing Tree Number):	1.1.2.3
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to test to make sure the I/O pins on the microcontroller are functioning to our standard. Working meaning able to read input and output accurately.
Experiment Procedure:	We will send a square wave analog signal from our microcontroller by waiting a set number of milliseconds and changing the GPIO pin to output that 1 or 0. This is done on each pin we use in the system. This shows we can output and receive signals correctly on the pins.
Equipment Settings / Software Settings (w Revision):	We use the nucleo board with the code shown below to produce the signal. Then we read the signal using our analog discovery shown on channel 2 in the data points.
Testing Diagram / Picture:	

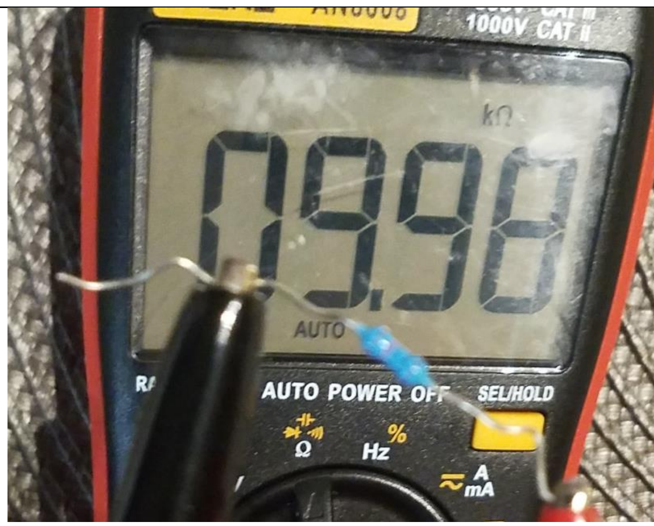
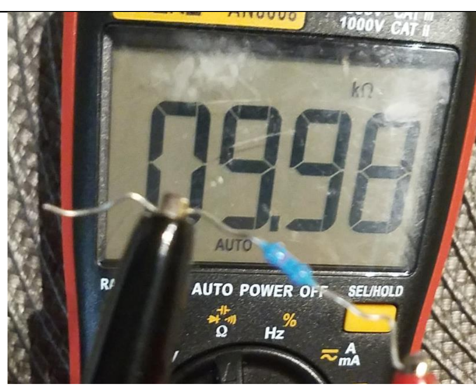
```c
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
  uint32_t captureVal;
  uint32_t time = HAL_GetTick();
  uint32_t millis_wait = 50;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

      if(HAL_GetTick() - time > millis_wait){
          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_0);
          time = HAL_GetTick();
      }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output vo
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTA
  /** Initializes the RCC Oscillators according to th
  * in the RCC_OscInitTypeDef structure.
  */
```

Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	The signal being displayed is correct because the delay between transitions is the same as defined in the code.
Recommendations for Modifications:	None

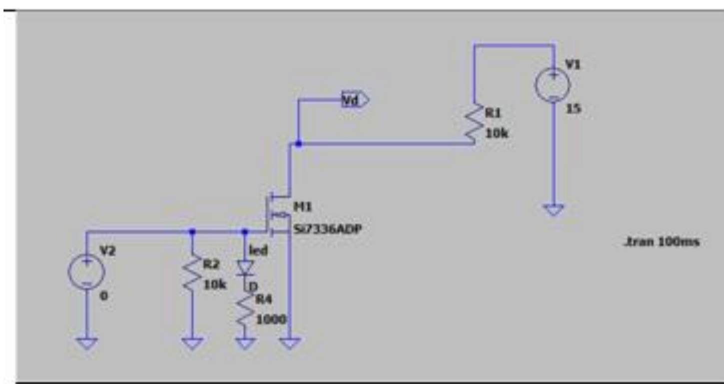
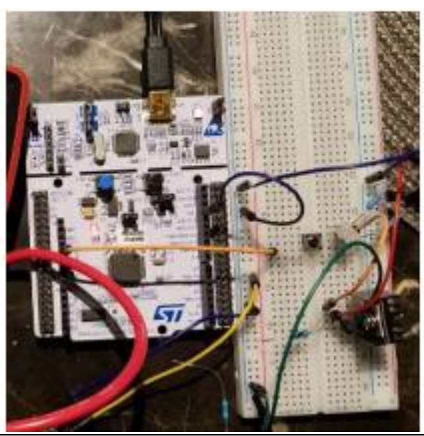
TNC Testing Form (REV1)	
Leaf on the Tree	Validate Signal Voltages
Device Under Test (Testing Tree Number):	1.1.2.4
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth, Kaleb Leon, David Cain
Signature:	
Experiment Purpose:	The purpose of this experiment is to test to make sure the voltages are accurate for our analog signal on the microcontroller.
Experiment Procedure:	We will send a square wave analog signal from the Nucleo, that is generated by waiting a number of milliseconds and change the GPIO pin to output a 0 or 1. We set the amplitude to a specific value to be measured in our scope analog discovery.
Equipment Settings / Software Settings (w Revision):	We use the nucleo board with the code shown below to produce the signal. Then we read the signal using our analog discovery shown on channel 2 in the data points.
Testing Diagram / Picture:	

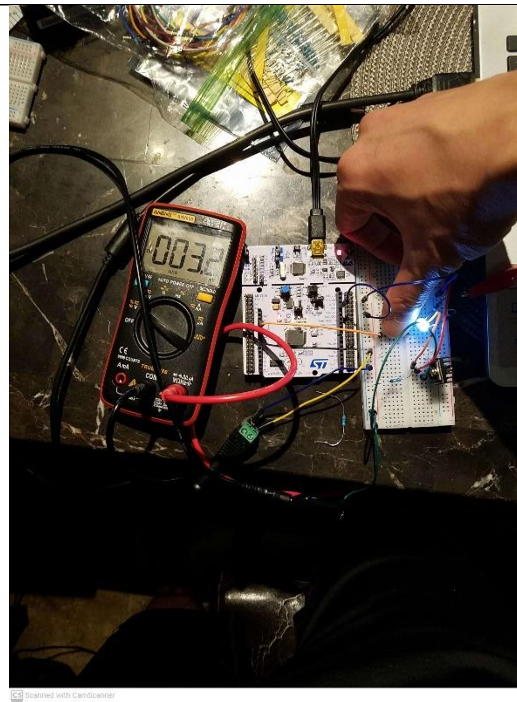
```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
  uint32_t captureVal;
  uint32_t time = HAL_GetTick();
  uint32_t millis_wait = 50;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

      if(HAL_GetTick() - time > millis_wait){
          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_0);
          time = HAL_GetTick();
      }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output vol
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAG
  /** Initializes the RCC Oscillators according to the
```

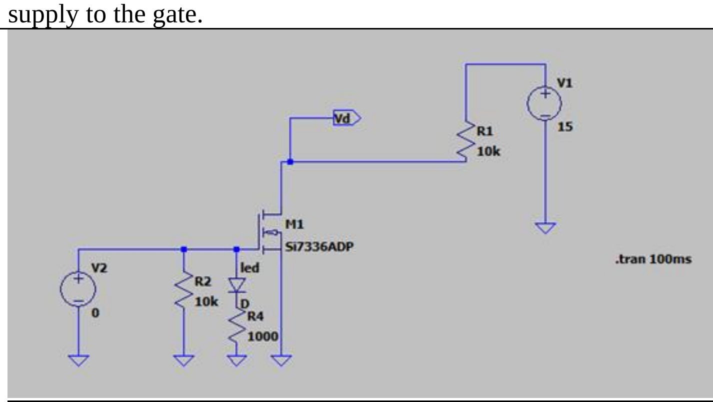
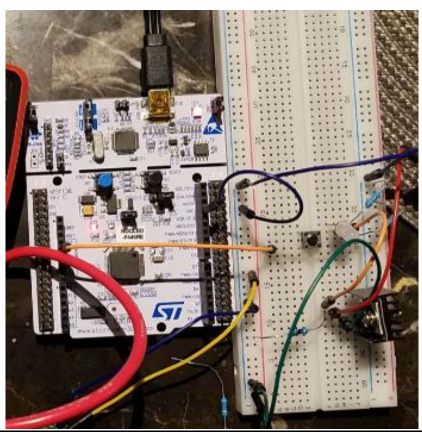
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	The signal being displayed is correct because the amplitude of the wave meets the input set in the Nucleo code.
Recommendations for Modifications:	N/A

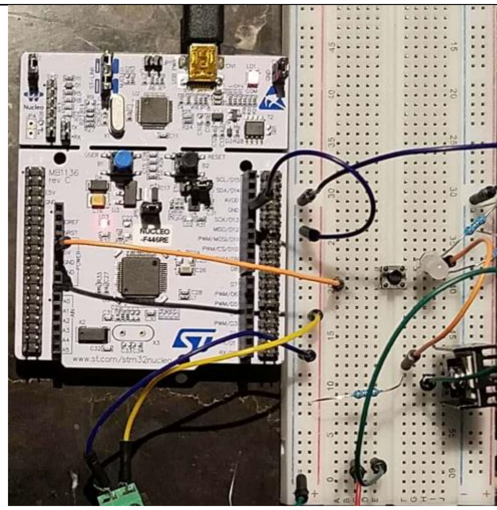
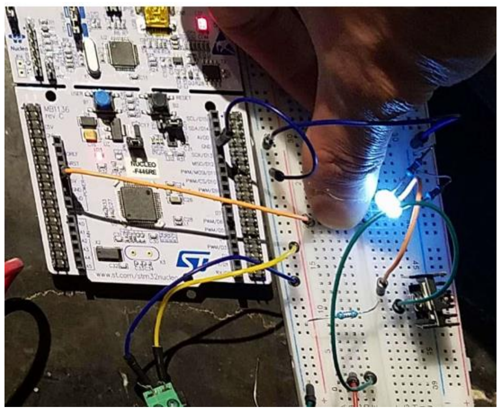
TNC Testing Form (REV1)	
Leaf on the Tree	Resistors
Device Under Test (Testing Tree Number):	1.2.1.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that the resistors used for the PTT circuit are within the proper tolerance, which is in between 5 percent over or under their nominal value.
Experiment Procedure:	Use a voltmeter to measure actual resistance of the 10k and 1k ohm resistors connected to the MOSFET's gate. The 10k passes if the measured value is in between 10.5k – 9.5k ohms. The 1k passes if the measure value is in between 1.5k - .95k ohms.
Equipment Settings / Software Settings (w Revision):	Use an Aneng multimeter to measure each resistor's resistance.
Testing Diagram / Picture:	
Data Points:	 **10k ohm**

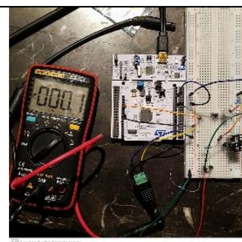
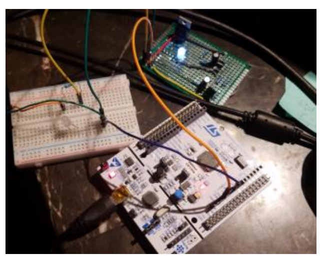
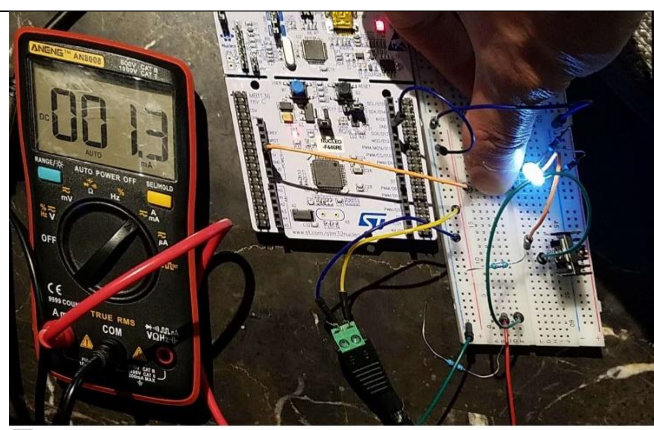
	 **<u>1k ohm</u>**
Pass / Fail:	Pass
Interpreted Notes:	As can be seen both resistors pass since they are with in the desired range.
Recommendations for Modifications:	None

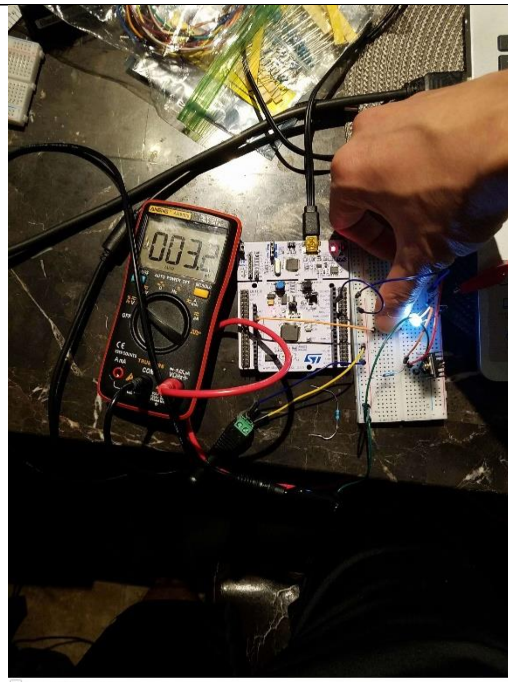
TNC Testing Form (REV1)	
Leaf on the Tree	PTT Circuit
Device Under Test (Testing Tree Number):	1.2.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to verify that the PTT circuit can pull 15V ,going into the drain, to 0 V when the it is turned on.
Experiment Procedure:	We will implement the circuit shown below and input 15 V with a pull-up resistor, to act as the radio's 15 V. Then we will use a tactile switch to switch the PTT circuit on and measure the voltage across the drain to source to see.
Equipment Settings / Software Settings (w Revision):	We use a breadboard to hook up the circuit shown below and a dc power supply for the 15 V. We used LTspice for designing the circuit. We use 3.3V reference to supply to the gate. Also, we will use a voltmeter to measure the drain to source voltage.
Testing Diagram / Picture:	 **Circuit** 

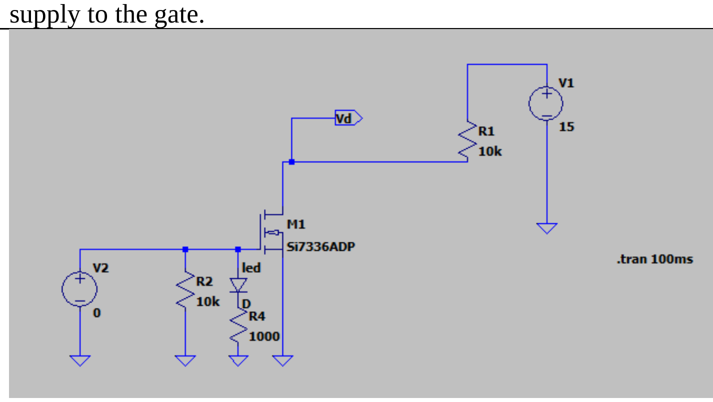
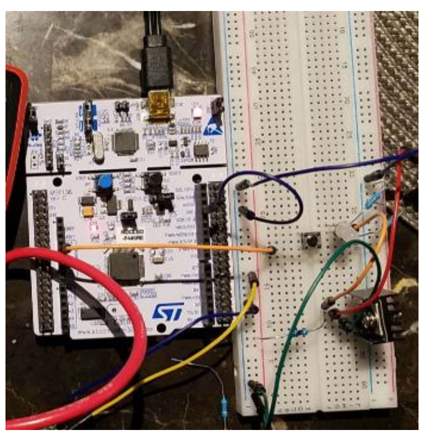
Data Points:	
Pass / Fail:	Pass
Interpreted Notes:	As shown, when a high signal is inputted into the circuit, then it is able to decrease the 15V at the drain down to 3.2 mV which very close to 0 V
Recommendations for Modifications:	None

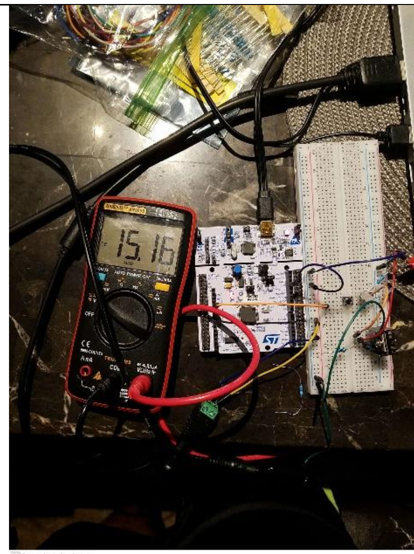
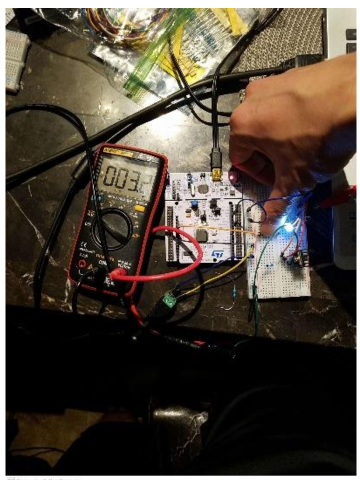
TNC Testing Form (REV1)	
Leaf on the Tree	1.2.2.1
Device Under Test (Testing Tree Number):	LED
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to verify the LED, associated with our PTT circuit is turned on, when the MOSFET switches on.
Experiment Procedure:	We will implement the circuit shown below and input 15 V with a pull-up resistor, to act as the radio's 15 V. Then we will use a tactile switch to switch the MOSFET on and off, which should also turn the LED on and off respectively.
Equipment Settings / Software Settings (w Revision):	We use a breadboard to hook up the circuit shown below and a dc power supply for the 15 V. We used LTspice for designing the circuit. We use 3.3V reference to supply to the gate.
Testing Diagram / Picture:	 **Circuit** 

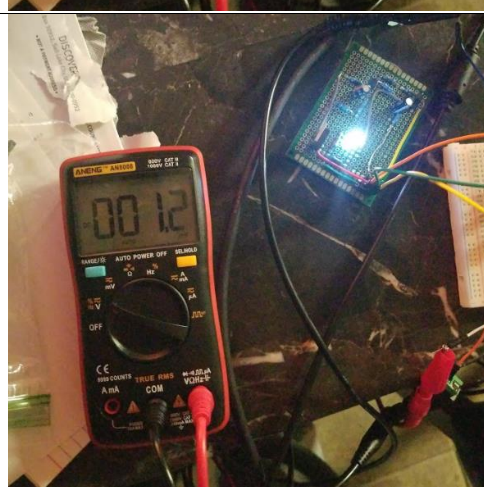
Data Points:	 **<u>LED off</u>**  **<u>LED on</u>**
Pass / Fail:	Pass
Interpreted Notes:	When the MOSFET has 3.3 V inputted into the gate, then the LED turns on. When the MOSFET has 0 V inputted into the gate, then the LED turns off.
Recommendations for Modifications:	None

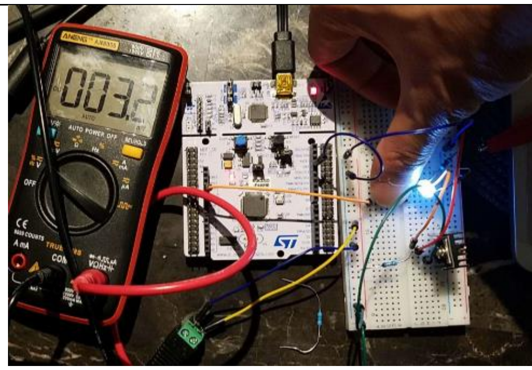
TNC Testing Form (REV1)	
Leaf on the Tree	Power Consumption
Device Under Test (Testing Tree Number):	1.2.3.1
Date:	1/11/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment was to measure the power consumed by the PTT circuit when turned on. Also, to see how stable the MOSFET's internal temperature is.
Experiment Procedure:	I used the soldered circuit on a perf board with 3.3V supplied to gate from my microcontroller and 15 V DC supply with a 10k ohm pull-up resistor connected to drain. Used a multimeter to measure current and voltage across drain to source.
Equipment Settings / Software Settings (w Revision):	Used a stm32 for the 3.3V DC supply and an external 15 V supply. Used a multimeter to measure current and voltage across drain to source.
Testing Diagram / Picture:	 
Data Points:	

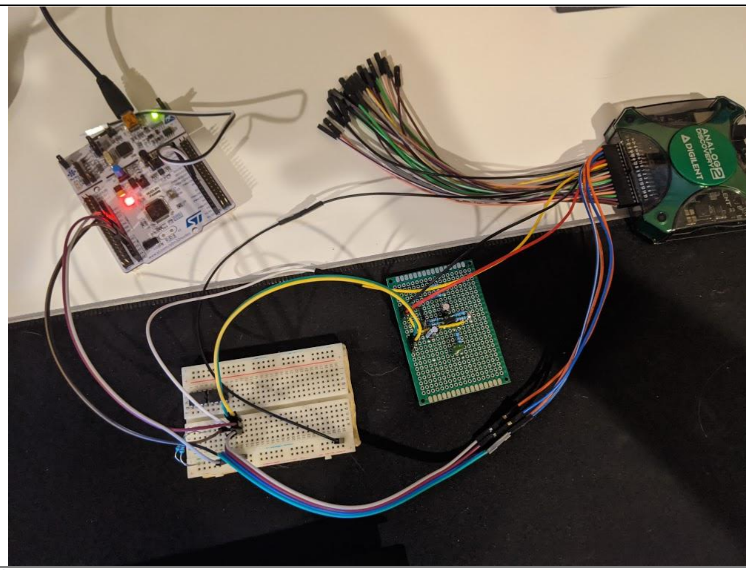
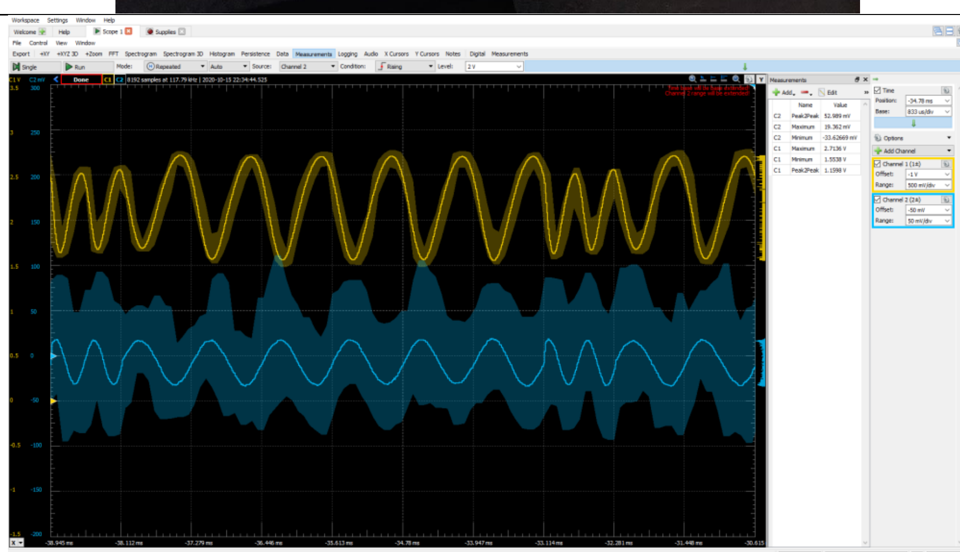
	
Pass / Fail:	Pass
Interpreted Notes:	The calculated Power consumption was 4.16 uW, which barely consumes any power. After leaving the MOSFET on for some time, it not necessary to measure the temperature considering that the MOSFET really did not heat up.
Recommendations for Modifications:	None

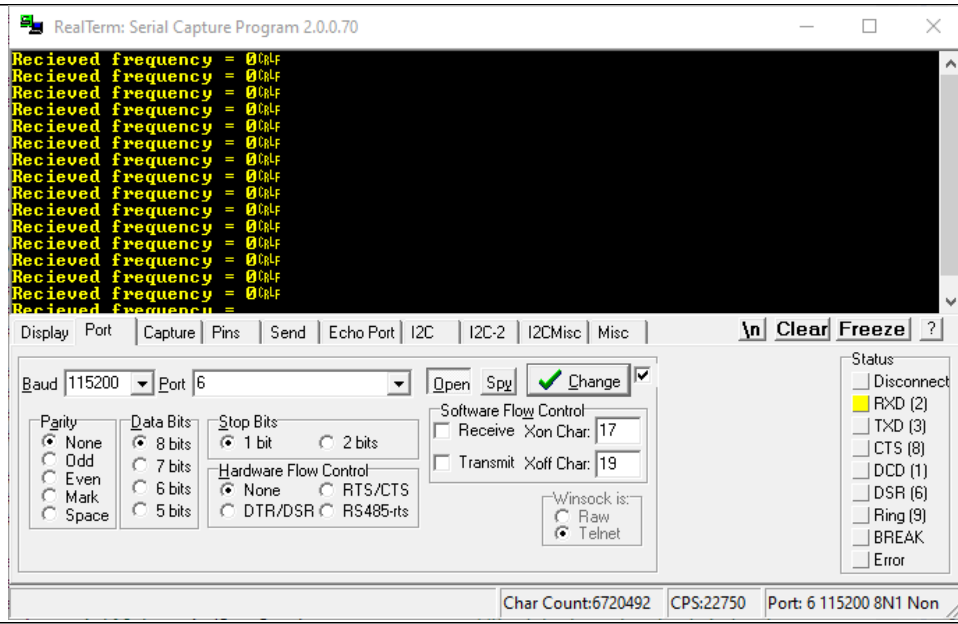
TNC Testing Form (REV1)	
Leaf on the Tree	MOSFET
Device Under Test (Testing Tree Number):	1.2.4
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure that the MOSFET is operating correctly. We are using the MOSFET as a switch to pull 15V, coming from the radio, to ground. This will tell the radio that the TNC is transmitting and the radio can stop transmitting.
Experiment Procedure:	We will implement the circuit shown below and input 15 V with a pull-up resistor, to act as the radio's 15 V. Then we will use a tactile switch to switch the MOSFET on and off. A voltmeter will be used to make sure our drain to source voltage becomes very small when the MOSFET has high signal input at the gate.
Equipment Settings / Software Settings (w Revision):	We use a breadboard to hook up the circuit shown below and a dc power supply for the 15 V. We used LTspice for designing the circuit. We use 3.3V reference to supply to the gate.
Testing Diagram / Picture:	 **Circuit** 

Data Points:	 **MOSFET Off**  **MOSFET on**
Pass / Fail:	Pass
Interpreted Notes:	As can be seen when a low signal is inputted into the gate, then the MOSFET’s drain to source is 15 V. As can be seen when a high signal is inputted into the gate, then the MOSFET’s drain to source is 3.2 mV, which should signal the radio to stop transmitting.
Recommendations for Modifications:	None

TNC Testing Form (REV1)	
Leaf on the Tree	Voltage Input
Device Under Test (Testing Tree Number):	1.2.5.1
Date:	11/1/20
Person(s) Conducting Experiment:	Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to validate that the MOSFET can operate correctly when inputted 15 V and as low as 5 V.
Experiment Procedure:	Input 5 V and 15 V in series, with a pull-up resistor, into the drain of the MOSFET and see if the voltage from drain to source drops to a very small voltage.
Equipment Settings / Software Settings (w Revision):	Use our microcontroller to input a high signal into the gate and 5V into the drain. Also input 15 V into the drain using an external voltage supply.
Testing Diagram / Picture:	
Data Points:	 **<u>5 V input</u>**

	 **<u>15 V input</u>**
Pass / Fail:	Pass
Interpreted Notes:	The MOSFET was successfully able to drop the input voltages very close to 0 V across the drain and souce.
Recommendations for Modifications:	None

TNC Testing Form (REV1)	
Leaf on the Tree	Amplifier
Device Under Test (Testing Tree Number):	1.3.1.1
Date:	10/15/2020
Person(s) Conducting Experiment:	David Cain, Kobe Keopraseuth
Signature:	
Experiment Purpose:	The purpose of this experiment is to ensure the amplifier output can trigger the external interrupt on the STM32 for reading incoming AFSK waveform frequency components.
Experiment Procedure:	Using waveform generator from Analog Discovery 2, input a 50mV ptp AFSK waveform to amplifier circuit, checking readings reported by microcontroller on serial port.
Equipment Settings / Software Settings (w Revision):	Micro controller is set to receiving mode Using WaveForms software(version 3.12.2), output generated AFSK signal
Testing Diagram / Picture:	
Data Points:	

	
Pass / Fail:	Fail
Interpreted Notes:	Frequency output remains zero due to gain not being high enough to trigger interrupt.
Recommendations for Modifications:	Modify amplifier circuit to increase gain.