Tyler Morris

Professor Lewicki

CIST 1451

December 11, 2018

Amateur Radio

Amateur Radio or ham radio as it is often called is a two-way radio service that is licensed by the Federal Communications Commission (FCC), at least in the United States. Other countries have their own licensing authorities, but all amateur radio operators must be licensed in order to transmit on any amateur frequency. According to the FCC, "The amateur and amateur-satellite services are for qualified persons of any age who are interested in radio technique solely with a personal aim and without pecuniary interest." Twenty-seven frequency spectrum allocations exist internationally for amateur radio operators to use. Amateur radio operators are also allowed to transmit, "some 1,300 digital, analog, pulse, and spread-spectrum emission types…," according to the FCC. Amateur operators are also given the ability to design, create, modify, repair, and experiment with radio equipment within the designated frequency spectrum allowed by the class of license the user holds.

Currently there are three license classes issued by the Federal Communications Commission: Technician, General, and Extra. Each class of license is earned through examination of a potential operator's skill and knowledge in operating an amateur radio station. The lowest level is the technician class license and the highest level is the extra class license (FCC). The exam questions come from a pool of questions that have been made public, according to the FCC. The technician class exam consists of thirty-five multiple choice questions regarding rules, regulations, and the safety of operating an amateur radio station. The general

class license exam also has thirty-five questions and is more difficult as it includes more radio and electronics theory. The extra class license exam, being the highest class of license available from the Federal Communications Commission consists of fifty multiple choice questions that contain a lot of radio theory, electronics theory, safe operating practices, more rules, regulations, and limits.

Amateur radio is also used as an emergency communications service during times of disaster (ARRL). Typically, amateurs are used during emergencies that cause a loss of power and damage to infrastructure-dependent communications methods (i.e. Cell Phones). Amateur radio operators are often involved in local, state, and national communications organizations. One local organization is the McKean County Emergency Communications Team, which is part of the McKean County Emergency Management Agency. At the national level, amateur operators can work through the Radio Amateur Civil Emergency Service which is coordinated by the Federal Emergency Management Agency (FEMA) and the Amateur Radio Emergency Service (ARES), which is coordinated by the American Radio Relay League (ARRL). Some amateur radio operators may also be involved in Skywarn which is coordinated by the National Weather Service. Amateurs involved in Skywarn provide weather information to the National Weather Service for analysis and dissemination to the public (ARRL).

A repeater is a radio that listens on one frequency and simultaneously retransmits that signal on another frequency (Butler). Typically, repeaters are used on Very High Frequency (VHF) and Ultra High Frequency (UHF) radio frequencies. The reason for this is that VHF and UHF radio waves do not travel very far. A well-placed repeater can easily cover hundreds of miles. Repeaters generally produce higher power than a typical mobile or handheld radio and usually use very efficient high gain antenna systems (Butler). Basically, this means that a user's

weak, low powered signal is being retransmitted with higher power output into a better antenna and in turn this produces a greater coverage area.

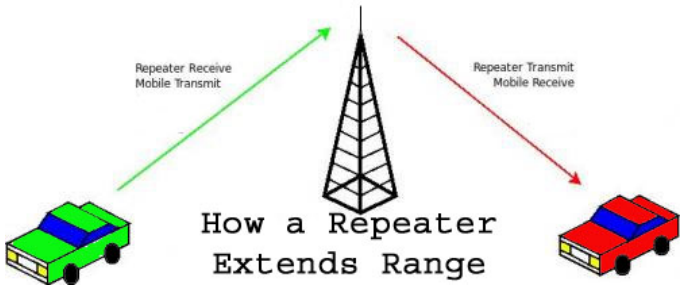


Figure 1: How a Repeater Works

One of the more interesting components of a repeater is the duplexer. This allows the repeater station to use a single antenna for both transmit and receive. According to Donald Butler, duplexers often look like large tall cans. A duplexer is a device that isolates the transmit and receive frequencies so that the repeater does not hear itself. A duplexer also has a very narrow band pass, which allows the repeater to hear stations on its receive frequency and to reject other signals on nearby frequencies (Butler). This helps a repeater to reject signals from nearby repeaters or transmitters and to reject radio frequency (RF) interference (Butler). A repeater can be setup without duplexers, however it will require a very large spacing between the transmit & receive antennas and a larger frequency offset.

Frequency offsets are used due to repeaters transmitting on one frequency and receiving on another. The actual amount of offset between the two frequencies varies by the band. For instance, the popular VHF band two meters (144-148Mhz) typically has an offset of 600 kilohertz, according to Donald Butler. The popular UHF band, 70 centimeters (420-450Mhz in the USA) typically has an offset of 5 megahertz (Butler). These offsets typically increase as the frequency increases; this is to avoid having the transmitter and receiver interfere with one another.

One way that repeaters avoid interfering with each other is through the use of Continuous Tone Coded Squelch System (CTCSS) tones or PL tones. According to Butler, PL is an acronym meaning "private line" and is Motorola's proprietary name for CTCSS. These tones allow a repeater to reject all signals without the proper CTCSS tone encoded in the radios transmission to the repeater. This is helpful in repeater-congested areas where many repeaters' frequencies may be close or the same (Butler). Essentially, if a repeater is using CTCSS tones, it will not hear radio stations that are not sending that tone.

Marconi first communicated across the Atlantic Ocean in 1901, but prior to his experiments, James Clerk Maxwell theorized electromagnetism in 1873. Marconi's first communication across the Atlantic used high power and giant antennas. Due to the high power, interference was a problem leading the United States Congress to approve the Radio Act of 1912. This piece of legislation required amateur radio operators to be licensed by the Federal Communications Commission and limited these amateurs to a wavelength of 200 meters or frequencies below 1,500 kilohertz (ARRL).

The American Radio Relay League was founded by Hiram Percy Maxim in 1914. Maxim found that sending messages over the radio could be done more reliably if there were relay stations. This led Maxim to create the American Radio Relay League which was an organization of amateur radio operators who acted as these relay stations. Amateurs began testing the transmission and reception of signals across the Atlantic in 1921. According to the American Radio Relay League, "…by July 1960 the first two-way contact via the Moon took place on 1296 Mhz." Today, amateur radio operators use various modes of communication bouncing signals off the moon, the ground, and the ionosphere. Amateur radio operators exist in almost every country on Earth and range in age from ten years old to more than one hundred years old (ARRL).

Modern amateur radio includes more ways to communicate than just voice or Morse Code. Amateur radio operators use a variety of different modes including Digital Smart Technology Amateur Radio (DSTAR) and Digital Mobile Radio (DMR). DSTAR was designed for amateur radio by the Japanese Radio League (DSTAR Info). DMR on the other hand was designed for commercial use and later adapted by amateur radio operators to work as an amateur radio system. These different modes can be used to send voice, text messages, images, files, and all kinds of other data types.

DSTAR or Digital Smart Technology Amateur Radio, is a mode that allows digital voice, text messages, files, pictures, and GPS location data to be transmitted and received. DSTAR also allows repeaters to link to other repeaters through the internet. Repeaters may also be connected together in what are called “reflectors” which are basically like conference servers (DSTAR 101). Repeaters are linked by the user’s transceiver by sending an eight-character code that tells the repeater what to do (DSTAR 101). For instance, to link to the local DSTAR repeater who’s callsign or license number is “KC3ESS”, the following code would be used without quotes, “KC3ESSBL”. The callsign and the module, which in this case is “B”, is used along with the “L” for “link”. The module indicates the frequency of the repeater where “B” is 70cm or around 440 megahertz and “C” is 2m or around 144 megahertz. There are also other commands which are always in the eighth character location such as the unlink command which is the letter “U”, the echo command which is the letter “E”, and the information query command which is the letter “I”. Since callsigns vary in length from three-character special event stations to six-character station callsigns, all commands are padded with spaces if there are not enough characters to reach that eighth position.

Digital Smart Technology Amateur Radio also allows slow speed data to be sent over the radio. This data can include GPS location data, images, and even files. Some DSTAR radios, such as those from Icom, a manufacturer of radios, can be interfaced with Icom's Android and iOS app called, "RS-MS1A." This app allows full control of the transceiver as well as the ability to send photographs and files through the radio. According to Icom America, a software called DRATS was designed for amateur radio first responders within the Amateur Radio Emergency Service (ARES) and the Radio Amateur Civil Emergency Service (RACES) and can be used with any DSTAR radio. The software includes the ability to chat via instant message, send structured emergency forms, send GPS position reports, and transfer files with error detection over amateur radio (Icom America).

DRATS can also be used to send email messages to regular email users, provided the emails are not business related and there is no money involved. Amateur radio is not for profit and to use it to make money is illegal. DRATS users can send data through radio waves to another DRATS station called an "internet gateway" and from there the message or email can be sent to any regular email user (Icom America). According to Icom America's DRATS brochure, this process can also be reversed allowing regular emails to be sent to an amateur radio station over radio.

Digital Mobile Radio or DMR for short, is another digital communications mode that is similar to DSTAR. What makes DMR different according to Bentvision LLC, is that it uses Time Division Multiple Access or TDMA to divide a single frequency into two "timeslots." These two timeslots can support two completely different conversations simultaneously on the same frequency. Instead of using a callsign to identify the radio, like DSTAR uses, DMR utilizes a unique radio identification number. This radio ID number uniquely identifies your radio on the

DMR network (Bentvision LLC). Radio ID numbers are very similar to a computer's IP address or a person's telephone number.

Much like DSTAR, DMR uses the internet to connect repeaters together around the world. Digital Mobile Radio utilizes what are called "talkgroups" to organize or group unique radio identification numbers together. This allows a DMR repeater to connect to a talkgroup and hear all of the stations that are connected as well as transmit to all of the connected stations, very similar to a DSTAR reflector (Bentvision LLC). Talkgroups are also identified by a unique number which is used to connect a repeater to them. Talkgroups can be static or dynamic meaning a repeater can have a static or permanently connected talkgroup or the repeaters may allow dynamic talkgroups in which end users are able to connect to various talkgroups as they please (Bentvision LLC). There are a couple of different DMR networks, one of which is the Brandmeister network (Bentvision LLC).

BrandMeister is an operating software for DMR master servers or repeaters. This software allows these servers to communicate with each other amongst a worldwide infrastructure network consisting of amateur radio digital voice systems (BrandMeister). According to BrandMeister, their network allows the following features: two-way text messaging, sending position reports, making private calls to other amateur radio operators, making worldwide group calls to other amateur radio operators, and roaming between repeaters.

For my capstone project, I wanted to put together a digital multimode amateur radio repeater that would switch between DSTAR and DMR depending on the signal it receives. I wanted to use a Linux distribution designed for making the Raspberry Pi handle this task. The software is called Pi-Star, which has basically become the standard software for MMDVM style radio interface boards. The plan was to add some extra features including remote control through

the ircDDBRemote app and scheduling of automatic linking on DSTAR. The repeater would also include a Nextion brand human machine interface touch screen display. This would allow the displaying of callsigns, names, and locations heard by the repeater as well as a control interface to shut down or reboot the Raspberry Pi. Lastly the project was to be enclosed in a 3D printed case that I would design.

I wanted to do this as my capstone project because it was a way to combine my love for ham radio with my passion for computers. It was also a way to experiment with a repeater on a small scale so that I could help other amateur radio operators in the surrounding area with converting their standard repeaters into digital multimode repeaters. As you will see below, I learned quite a bit more about how these digital repeaters work and how I can better implement the technology on a larger scale within the community.

The first thing I had to do was decide on the hardware and software I would use for this project. I considered building a full-size repeater, but due to cost and demonstration limitations, I decided to use a cheaper Chinese clone of an MMDVM_HS_HAT_DUPLEX, dubbed a JumboSpot. Basically, this is a full repeater on a single printed circuit board. It utilizes two radio transceiver chips and a microcontroller. One of these transceiver chips transmits while the other receives a signal. This repeater board has a maximum output power of ten milliwatts, compared to a repeater's typically much higher output power. My concern in purchasing this board was that due to the antenna spacing I would potentially have issues with the transmitter overloading the receiver, however after I received the board and got it up and running, that doesn't appear to be a problem. It also helps that the repeater's transmit and receive frequencies are offset by five megahertz.

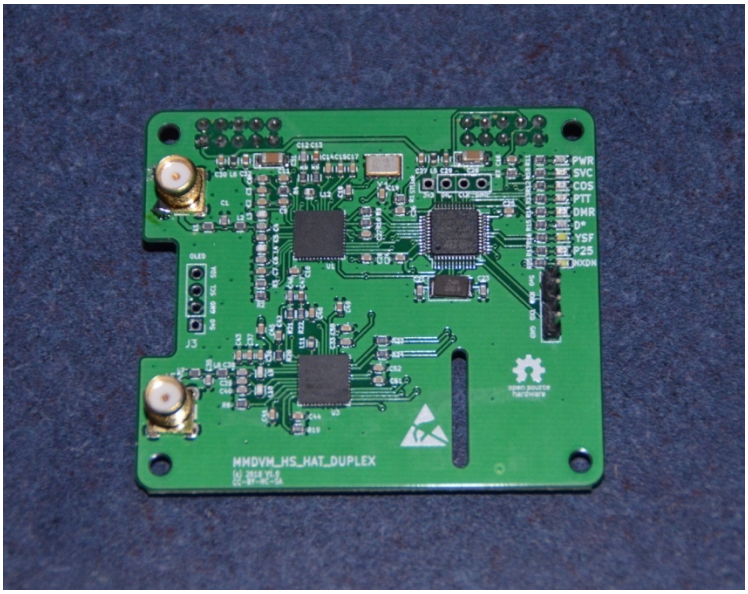


Figure 2: MMDVM_HS_HAT_DUPLEX

For the software, I decided to use the standard Pi-Star software created by Andy Taylor, MW0MWZ, an amateur radio operator in the United Kingdom. Pi-Star is a distribution of Linux based upon Raspbian Linux. It has been heavily modified to include numerous pieces of a software for amateur radio as well as another interesting feature that makes the filesystem read-only to protect the Raspberry Pi's operating system during an unexpected shutdown. The filesystem remains read-only unless you specifically run the command, "rpi-rw" to make it read/write. In my limited experience, this seems to work pretty well; I haven't had any issues with unexpected shutdowns causing data loss. Pi-Star also features an online dashboard that allows you to view all of the activity on the repeater as well as to configure and control the repeater.

My next step was to register my callsign for DSTAR and to get a CCS7 or DMR ID number. Registering my callsign was as easy as visiting the website for the closest DSTAR repeater, which happens to be W3EXW in Pittsburgh, Pennsylvania. After registering with this repeater/gateway, I next had to setup terminal ID's. These terminal ID's identify your radio on

the DSTAR network. In my case I setup two ID's, one for me personally, with a blank ID which would identify my handheld DSTAR radio and another with the ID, "B" which would identify my repeater. The letter "B" corresponds to the module letter of the repeater I am setting up. Since the repeater is on the 70cm or 440-megahertz band, the standard is to use the letter "B" for the module.

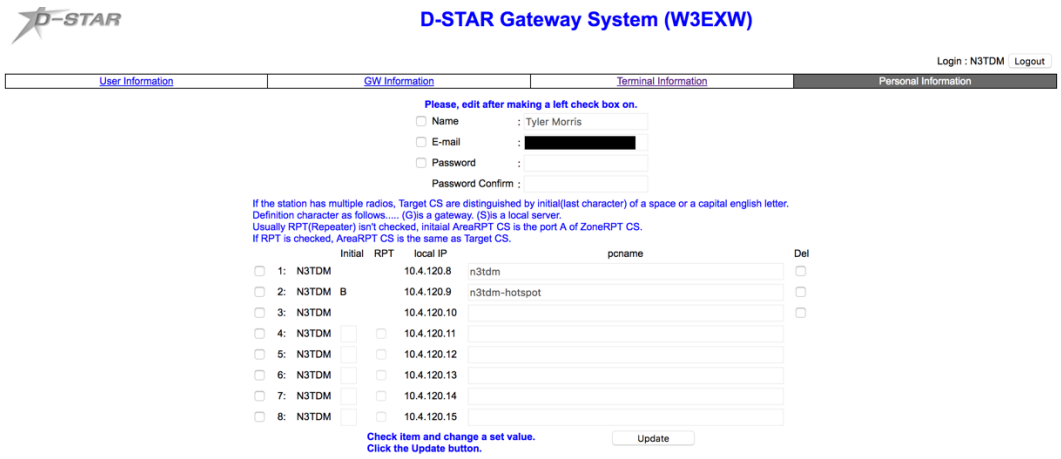


Figure 3: Terminal ID's for DSTAR

Next, I needed a DMR ID or CCS7 Number for the DMR mode to work. This application process is done through the register.ham-digital.org website. DMR routing uses numbers instead of callsigns, so a DMR ID or CCS7 Number is like a phone number. It is used to identify the user on the DMR network. This ID corresponds to the person's callsign and name in the DMR-MARC database. DMR-MARC is the Digital Mobile Radio – Motorola Amateur Radio Club.

After getting the DMR ID, I was ready to configure Pi-Star. The first thing I had to do was configure the WiFi to work with the University's enterprise network. I followed the instructions I found online here https://gist.github.com/chatchavan/3c58511e3d48f478b0c2. This involved using an ethernet crossover cable between my laptop and the Raspberry Pi, so that I could connect to it and reach the Pi-Star dashboard. Once on the dashboard I went to the

configuration tab, then the expert tab, and then to the SSH access tab and logged into SSH. From here I was able to edit the network interfaces file to include the following for WLAN0:

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
        pre-up wpa_supplicant -B -Dwext -i wlan0 -c/etc/wpa_supplicant/wpa_supplicant.conf
        post-down killall -q wpa_supplicant
```

After editing the network interfaces file, was ready to edit the wpa_supplicant file. This file is where the network information and credentials are stored. The first problem I encountered was that I did not want to store my password to the network in plain text. Luckily the instructions I was following provided information on how to use a hash of the password in the wpa_supplicant file. So, I hashed the password using the following code in SSH:

```
echo -n 'YOUR_PASSWORD' | iconv -t utf16le | openssl md4
```

After copying the hash of my password, I cleared my history in the bash prompt and exited SSH. I then went to the expert WiFi editor on the dashboard and entered the following, putting the password hash after the line “password=hash:”.

Figure 4: Configuring Enterprise WiFi

The next problem I encountered was also related to the WiFi. Pi-Star would only connect to the network occasionally. I determined the underlying cause to be the amount of time it takes to authenticate to WIRELESS-PITTNET. The solution to this problem was to turn off the WLAN interface and turn it back on, after Pi-Star has booted. This was accomplished by creating a bash script with the following line it:

sudo ifdown wlan0 && sudo ifup wlan0

Since I had planned on adding a Nextion touch screen display to the project, I added a button on the control page of the screen to execute the script and reset the WLAN interface. This seemed to solve the problem.

On a related note, if you are just setting up a repeater on a home WiFi network, there is a tool that will create the wpa_supplicant file for you and makes this task super simple. On the pistar.uk website, there is a tool called "WiFi Builder." It gives you an online form to enter your network name and password and then it will download the wpa_supplicant file to your computer. After downloading the file, it is placed into the "boot" partition of the SD card on the Raspberry Pi. After booting the Raspberry Pi, this wpa_supplicant file replaces the current wpa_supplicant file and then Pi-Star reboots and connects to the network specified in this new file.

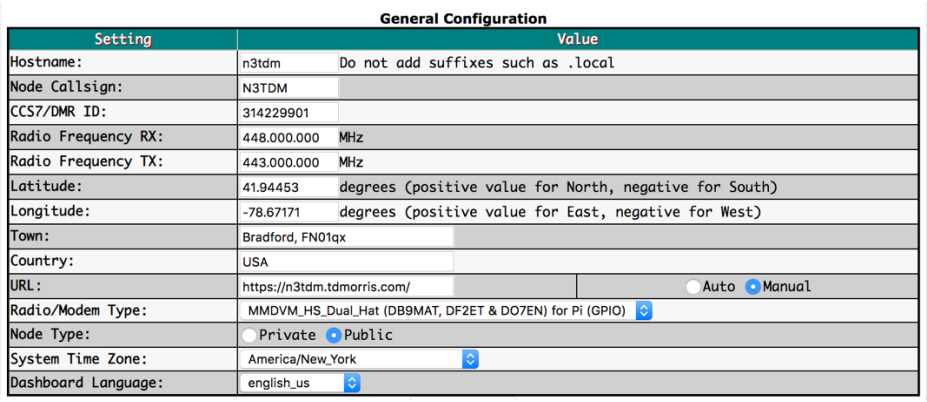
This repeater will work without port forwarding; however, it will not have full functionality in that other repeaters will not be able to route callsigns to it nor will the users be able to connect into my repeater. Pi-Star has the option to forward these ports automatically via Universal Plug N Play (UPNP). I have this turned off on my Pi and on my home network's router, so at home I forwarded the ports automatically. In order to forward ports, you must have a static IP address on the Pi. At home I reserved an IP address for my Raspberry Pi and forwarded the following ports to it, which I found on the DSTAR 101 website.

Port / Port Range	Protocol
20001 - 20007	DPlus - UDP
30001 - 30007	DExtra - UDP
30051 - 30057	DCS - UDP
30061	CCS - UDP
40000 - 40005	G2 - UDP
9007	ircDDB - TCP
8082 forwarded to internal port 80	Dashboard - TCP
10022	ircDDBRemote - UDP

After setting up the network on the Raspberry Pi, I had to configure the two modes, DSTAR, and DMR. I started by configuring only DSTAR in order to test the device and setup the transmit and receive offset of the MMDVM. On the configuration page, I enabled DSTAR and set the “RF Hangtime” and “Net Hangtime” to ten. RF Hangtime is the amount of time in seconds after the end of the last radio transmission, that the repeater will remain in DSTAR mode. Net Hangtime is essentially the same thing, but it applies to the end of the last transmission coming through the internet from the DSTAR network. After enabling the DSTAR mode, I needed to configure the general repeater information.

I began configuring the repeater information by first changing the hostname from “pistar” to my callsign, “n3tdm”. Then I entered the node callsign as my personal callsign and the CCS7/DMR ID to my personal DMR ID, with a supplemental station ID (SSID) of “01”. This SSID would prevent confusing the DMR network with two radios that have the same SSID. It is a way of uniquely identifying my individual radios, the handheld radio I would be talking on and the repeater. I also chose to make the repeater public instead of private. Setting this to public means that any licensed amateur radio operator can use the repeater. If this setting is set to private, the repeater will only accept your callsign. You can see the information I have entered in Figure 5, below.

Next came the task of finding a set of frequencies that I could use for my repeater. Normally on a full-size repeater, this is done through a repeater coordination organization, that makes sure your repeater and its coverage area will not interfere with any other coordinated repeater. This helps to keep interference to a minimum. To begin, I looked up the allocated frequencies for repeaters in the 70-centimeter (70cm) amateur radio band and found that the repeater sub-band is in the range of 440 to 450 megahertz. Next, I checked some online directories of repeaters and asked another local amateur radio operator named, Ted Leonard, W3VG, about the local 70-cm repeaters. Ted is a local amateur radio operator that has installed, tested, and maintained repeaters for more years than I have been alive. I also considered the fact that this repeater was going to be portable and was going to produce ten milliwatts or less due to losses in the antennas and coax cables. I settled on a repeater transmit frequency of 443 megahertz with a standard five-megahertz positive split, giving me a receive frequency of 448 megahertz.



General Configuration

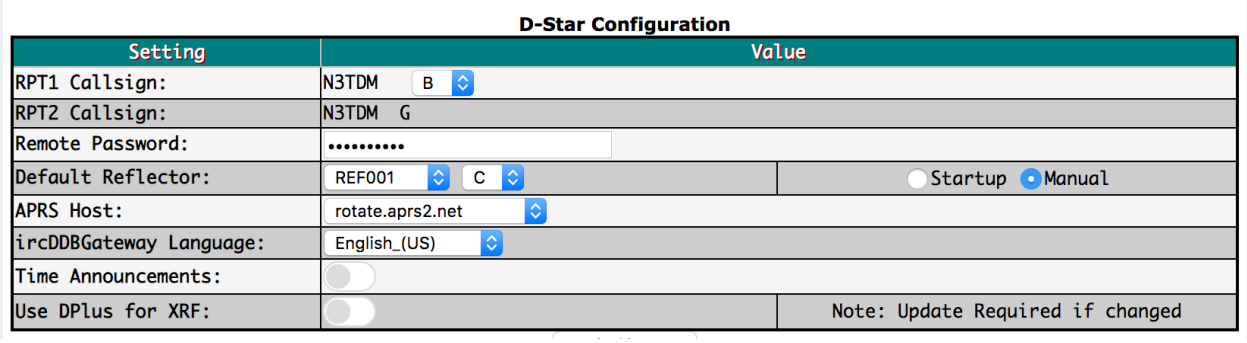
Setting	Value	
Hostname:	n3tdm	Do not add suffixes such as .local
Node Callsign:	N3TDM	
CCS7/DMR ID:	314229901	
Radio Frequency RX:	448.000.000	MHz
Radio Frequency TX:	443.000.000	MHz
Latitude:	41.94453	degrees (positive value for North, negative for South)
Longitude:	-78.67171	degrees (positive value for East, negative for West)
Town:	Bradford, FN01qx	
Country:	USA	
URL:	https://n3tdm.tdmorris.com/	Auto Manual
Radio/Modem Type:	MMDVM_HS_Dual_Hat (DB9MAT, DF2ET & DO7EN) for Pi (GPIO)	
Node Type:	Private Public	
System Time Zone:	America/New_York	
Dashboard Language:	english_us	

Figure 5: General Configuration

After configuring the general parts of the receiver, I moved on to configure DSTAR. In this section I had to set the RPT1 callsign, the remote password for the ircDDBRemote App, the default reflector, and the ircDDBGateway language. The RPT1 callsign must be 8 characters in

length, but the Pi-Star software does this for you. So, I entered my callsign, “N3TDM”, and then selected module letter “B” which is the standard module used for the 70cm band. If my repeater were on the two-meter band (144 to 148 megahertz), I would have used module letter “C”. I next set the remote password which would be used with the ircDDBRemote application on my phone or computer. It is important to note that there are some characters that Pi-Star does not work with, for instance the dollar sign ($). When I tried to use the dollar sign in my password, Pi-Star would not let me connect the app to the gateway. Once I removed this character from my password, I had no problem making the connection. The next item is the default reflector. A reflector is similar to a conference server and if it is set as the default and “startup” is marked, the repeater will automatically connect to the reflector on boot. I did not want this feature, as I wanted to have more control over which reflectors I connect to. I set the default reflector to REF001C and checked the “manual” radio button. Setting this to manual means that the repeater must be manually connected to the default reflector and will not connect on boot.

The next setting is the Amateur Packet Reporting System (APRS) host. This is the server that GPS location data will be sent to for both the repeater and any user who transmits their coordinates to the repeater. I chose the “rotate.aprs2.net” server based on the fact that it is a tier two server and will cycle through various aprs2.net core servers increasing reliability that the packets will reach the APRS network. DSTAR uses D-PRS or DSTAR Packet Reporting System which converts the data strings from the GPS in Icom (a DSTAR radio manufacturer) radios into APRS strings, according to Peter Loveall. This location data can be viewed on the aprs.fi website which tracks APRS stations on a map.

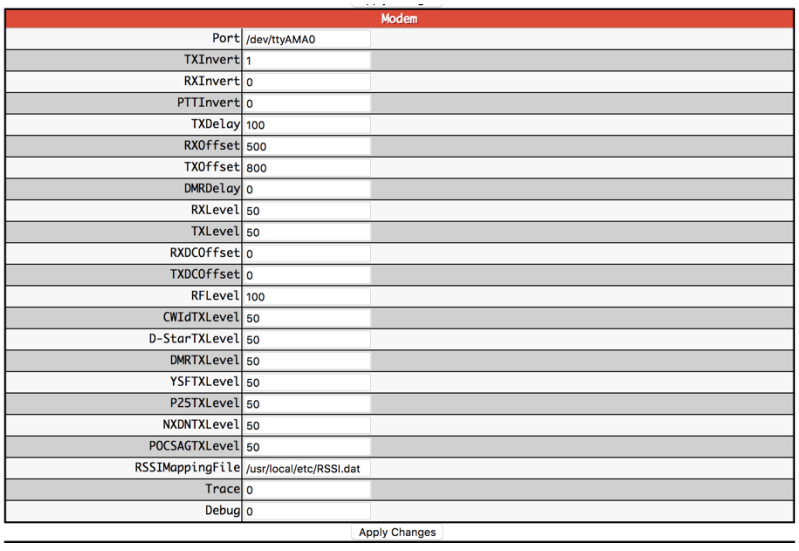


D-Star Configuration

Setting	Value	
RPT1 Callsign:	N3TDM B	
RPT2 Callsign:	N3TDM G	
Remote Password:	••••••••••	
Default Reflector:	REF001 C	○ Startup ◉ Manual
APRS Host:	rotate.aprs2.net	
ircDDBGateway Language:	English_(US)	
Time Announcements:		
Use DPlus for XRF:		Note: Update Required if changed

Figure 6: DSTAR Configuration

The next step is to configure the transmit and receive offset. On the back of the board there is a sticker that tells you an estimated transmit and receive offset. In my case, the sticker listed 500 for the transmit offset and the receive offset. So, I set those in the MMDVMHost expert editor and then I saved it. The next step is to transmit into the pi using DSTAR and see if it receives the signal. It will tell you on the dashboard if it received the signal. Mine did work, but wasn't transmitting any audio out, so that told me to change the transmit invert in the MMDVMHost configuration. After setting the transmit inversion to one for on, I transmitted into the repeater again using an echo test and was able to receive the transmitted audio when the repeater retransmitted my signal.



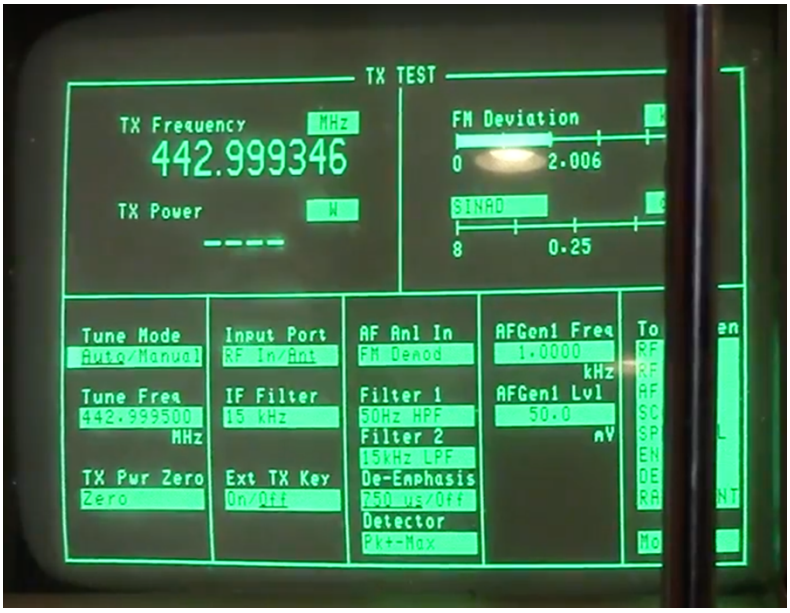
Modem

Port	/dev/ttyAMA0
TXInvert	1
RXInvert	0
PTTInvert	0
TXDelay	100
RXOffset	500
TXOffset	800
DMRDelay	0
RXLevel	50
TXLevel	50
RXDCOffset	0
TXDCOffset	0
RFLevel	100
CWIdTXLevel	50
D-StarTXLevel	50
DMRTXLevel	50
YSFTXLevel	50
P25TXLevel	50
NXDNTXLevel	50
POCSAGTXLevel	50
RSSIMappingFile	/usr/local/etc/RSSI.dat
Trace	0
Debug	0

Apply Changes

Figure 7: MMDVMHost Configuration

To set the transmit and receive offsets to be more accurate, I used a Hewlett Packard service monitor that my friend Ted owns. It reads the frequency that the repeater is transmitting on and tells me what it is, down to one hertz. I set my transmit frequency to 443 megahertz and the repeater was transmitting on 442.999200 megahertz, so the transmit offset on the repeater should be set to 800. In Figure 7, I had tested the repeater's transmit frequency after setting the transmit offset to 200.



Setting the receive offset is a bit more complicated and I set that by watching the BER or Bit Error Rate. The Bit Error Rate should be less than five percent, however even at five percent, audio can drop in and out. I had set my receive offset to 500, based on the sticker on the back of the repeater board. My bit error rate was very low to begin with so I did not adjust the receive offset anymore and left it at 500.

The next step is to enable DMR and configure it. In the MMDVMHost configuration file, I found the section for DMR and enabled it by setting enable to "1". I set the hangtime to 10 seconds and left everything else as default. I made sure to use color code one as well. Color codes are a way of grouping repeaters.

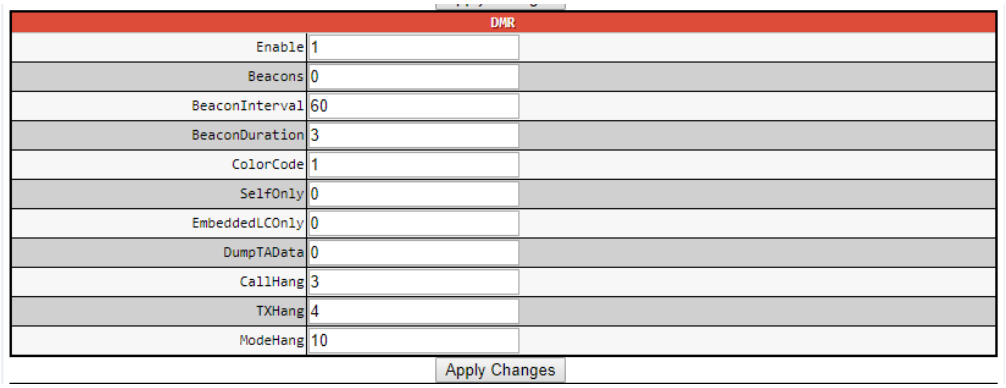


Figure 8: Enabling DMR in MMDVMHost Expert Editor

The next step was to setup the DMR network by first turning it on and then telling my repeater what server to connect to. There are two DMR networks, DMR-MARC and Brandmeister. DMR-MARC is a closed network and is only open to Motorola DMR repeaters, so I chose to use Brandmeister. I went to the brandmeister.network website and found the closest server which is Brandmeister 3108, which is located in the USA. I found the IP address of the server on the same page. I used that IP address as the address and left the ports as the default. The default password is used for most of these servers, which happens to be “passw0rd”. This is also where we turn on slot one and slot two. The two slots are called time slots and are essentially channels on the same frequency. This means that DMR can support two entirely different conversations on the same frequency at the same time.

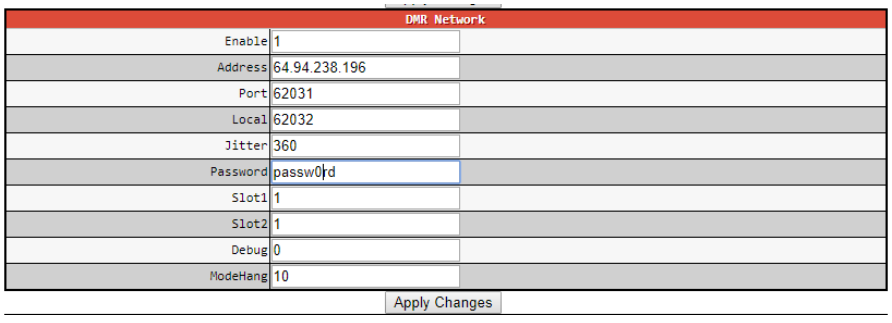


Figure 9: Configuring DMR Network

At this point the repeater is working, but I also wanted to add a Nextion display to this project. A Nextion display is a human machine interface display that processes code sent to it by a driver on the Raspberry Pi. The Nextion Driver from ON7LDS, gave me some problems while trying to install it. There is an automatic installation script, but for some reason the display did not update fast enough with the automatically installed Nextion Driver. I ran the install script and then ran most of the commands in the script manually. I also changed the source code of the Nextion Driver to output the CPU temperature in Fahrenheit instead of Celsius. The Pi-Star software comes with a driver for the Nextion display already installed, but I wanted to add some buttons to my display and in order to process the bash commands from those buttons, you need the Nextion Driver installed and the display needs to be connected to the Raspberry Pi using a USB to TTL Serial adapter. I used a PL2303 USB to TTL adapter.

Figure 10: Nextion Display Front *Figure 11: Nextion Display Back*

After installing the Nextion Driver, I needed to create a Nextion Layout. I modified one from PD0DIB, an amateur radio operator in the Netherlands. I changed his layout to have the transmit and receive frequencies be populated from the MMDVMHost configuration file instead of having them manually typed into the display in static text. I also added a system status page and an information page. On the information page, I had to add the text boxes for CPU load, IP address, transmit and receive frequencies, and CPU temperature. These boxes are targeted by the

Nextion Driver using their object labels. For instance, CPU load is labeled “cpuload” and IP address is labeled “t3”. So, the Nextion Driver sends the information to the display’s objects and shows it in the corresponding text box. I also had to edit the background images used in the Nextion layout because they had some text embedded in the images that I did not want on my layout.

Figure 12: Nextion Layout Main Screen

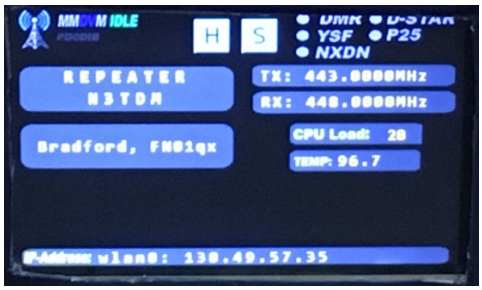


Figure 13: Nextion Layout Status Screen

On the control screen, I added buttons to reboot, shutdown, start and stop MMDVMHost, and reset the WLAN0 interface. The buttons contain code in a touch release event. I found the code on the Nextion Driver GitHub from ON7LDS. In this event the code looks like the following:

```
printh 2A
printh F1
printh <linux command>
printh FF
printh FF
printh FF
```

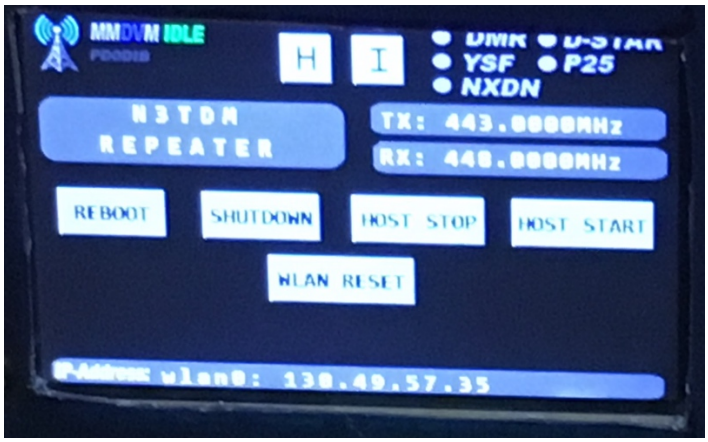


Figure 14: Nextion Layout Control Screen

The layout was created using Nextion Editor that is produced by Itead Nextion. The editor is a graphical interface used to create these layouts. It looks very similar to Visual Studio, but there really is not much code involved. It is more graphical than anything. As I said above, the text boxes have labels and the code in Nextion Driver targets those objects to display the information from MMDVMHost in those objects/text boxes. After creating the layout in Nextion Editor, you can either compile the layout into a *.tft file and put it on an SD card to image the screen, or you can connect the screen to your Windows PC with a USB to TTL Serial adapter and flash the screen directly, which is what I did. The USB to TTL Serial adapter I used was plug and play, but I did need to find the serial port or COM port it was assigned by Windows. I did this in Device Manager and then set the COM port in the Nextion Editor and uploaded the screen layout to the display.

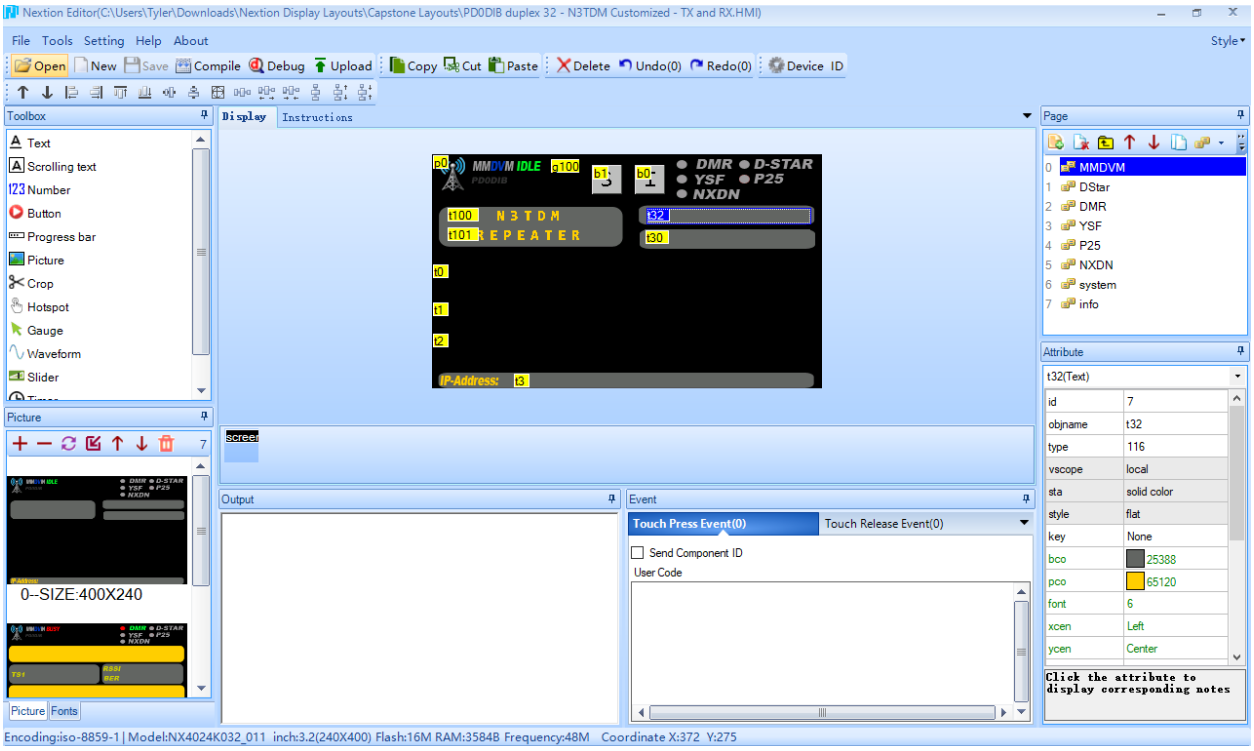


Figure 15: Nextion Editor Showing Main Screen

I also wanted to create a 3D printed case for this project, but was unable to complete it in time. I used tinkerCAD.com to create the 3D rendering of a box with mounting holes for the Raspberry Pi. That was all I could accomplish. I learned that 3D modeling is difficult and that there are a lot of different tools used to create something that seems so simple, like a box.

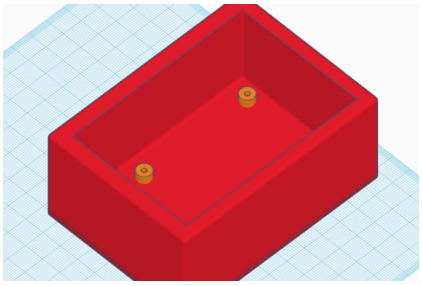


Figure 16: 3D Case Bottom Design

In conclusion, I learned a great deal of useful information from working on this capstone project. Doing this as my capstone project allowed me to learn more about the Raspberry Pi,

Raspbian Linux, and amateur radio digital modes. I learned that DSTAR, which was designed for amateur radio was much simpler to implement than DMR which was designed for commercial use and adapted for use with amateur radio. I also learned more about repeaters and how duplexers work to isolate signals from interfering with the repeater. The duplexers also allow the repeater to use the same transmit and receive antenna. I gained some experience using a service monitor to analyze a radio signal and set the transmit frequency offset. I gained a lot of experience working with the WiFi and the trouble that gave me when trying to connect to the enterprise WiFi on campus. I also gained real world experience with port forwarding on my home router. I also learned how difficult 3D modeling can be when designing a simple box to use as a case for this project. Perhaps the most important thing I learned was about myself and not about computers; I learned that I need to do better time management and not to expect success. While my project was successful in my opinion, I learned not to expect it to be an easy road to the end. I learned that something will inevitably go wrong and I learned how to find answers to solve my own problems.

Overall, I think this project turned out well, but there are a few things I would have done differently. I would have not taken on so much of the project in 16 weeks, as it was a lot of work to do in such little time. I also would have liked to have added more features to Pi-Star, but this project is just the beginning for me as I am positive I will continue working on it and improving it to learn even more to help my fellow amateur radio operators in the community with their digital repeaters. I also would have liked to build a full-size repeater, but due to cost and space, I am happy to have been able to experiment with repeaters on a small scale.

Works Cited

ARRL. *Amateur Radio Emergency Communication*. n.d. 10 November 2018. <http://www.arrl.org/amateur-radio-emergency-communication>.

—. *Ham Radio History*. n.d. 10 November 2018. <http://www.arrl.org/ham-radio-history>.

Bentvision LLC. *DMR Talkgroups*. n.d. 11 November 2018. <http://www.dmrfordummies.com/talkgroups/>.

—. *What is DMR?* n.d. 11 November 2018. <http://www.dmrfordummies.com/library/>.

BrandMeister. *What is BrandMeister*. n.d. 11 November 2018. <https://wiki.brandmeister.network/index.php/What_is_BrandMeister>.

Butler, Donald I. *How to Use Amateur (Ham Radio) Repeaters*. n.d. 11 November 2018. <http://www.hamuniverse.com/repeater.html>.

DSTAR 101. *What is DSTAR?* n.d. 10 November 2018. <http://www.dstar101.com/whatisdstar.htm>.

DSTAR Info. *Frequently Asked Questions*. n.d. 11 November 2018. <http://dstarinfo.com/faq.aspx>.

FCC. *Amateur Radio Service*. 6 July 2017. 10 November 2018. <https://www.fcc.gov/wireless/bureau-divisions/mobility-division/amateur-radio-service>.

Icom America. "dratsbrochure.pdf." 15 May 2009. *Icom America.* 10 November 2018. <http://www.icomamerica.com/en/products/amateur/dstar/dstar/dratsbrochure.pdf>.

Loveall, Peter. *D-PRS*. n.d. 8 December 2018. <http://www.aprs-is.net/dprs.aspx>.

ON7LDS. *Nextion Driver*. January 2018. 8 December 2018. <https://github.com/on7lds/NextionDriver/tree/master/Nextion>.