Satellite Telecommunication and Construction of a CubeSat

Arts Louis, Coolen Finn, Small Sander, Tjepkema Jonas, Verges Florian

Supervised by
Chad Ellington

June 29th, 2021

Introduction

This project aimed to replicate and test the main components of a weather satellite communication channel. This aim boiled down to two main steps.

The first part of this project was building a weather satellite monitoring station, which allowed us to receive live black and white images of most of Europe and Northern Africa from National Oceanic and Atmospheric Administration (NOAA) weather satellites. Weather satellites play an essential role in our understanding and prediction of the earth's environment. We use them to monitor fires, storms, cloud patterns, and other meteorological data, Volcanic eruptions, dust clouds, and much more. The American NOAA satellites we focused on in this project are low-orbit polar satellites, meaning they fly from pole to pole, relatively close to the earth's surface. This allows them to image every location on earth twice per day with excellent resolution. The two main components of our weather monitoring station were a self-made antenna, specially designed to receive a specific wavelength of radio signals from NOAA satellites and software-defined radio. The ladder is a device that allows us to demodulate and further process the signals we receive from our antenna using specialized software on a personal computer. Many different antenna designs enable us to receive images from NOAA satellites. The first antenna build followed a v-dipole design. The second version was a much more complicated omnidirectional double cross antenna. For this design, the direction of the antenna during signal receival did not influence the overall strength and resolution of the incoming signals, resulting in much clearer images.

The radio signal is originally received on the pc as a modulated audio signal consisting of repetitive clicks and beeps. Some audio processing and specialized software are later needed to convert such an audio signal into a black and white image of Europe.

The second goal of this project was to build our miniature satellite that transmits real-time data such as live images, temperature, pressure, location, light intensity, satellite orientation, and acceleration. This data was then uploaded live to a web browser with interactive UI to allow easy access. The general design of our miniature satellite was inspired by the CubeSat, a widely used type of small satellite with standardized units of volume and mass. Engineering students widely use these satellites for scientific studies and other educational purposes. The main structure of our satellite was designed using fusion-360 and 3D-printed with Polylactic acid. The monitoring components include a BMP-280 temperature and pressure sensor, GY-302 photodetector, GY-521 gyroscope, Raspberry-pi camera, and a GT-U7 GPS receiver. The satellite comes equipped with a power supply, including a battery and two photovoltaic panels, which can be folded open and closed on command to theoretically point in the sun's direction when orbiting the earth. This folding out of

the photovoltaic panels was made possible by a 3D-printed gearless mechanism powered by a 28BYJ-48 step motor. To allow us to send commands to the satellite and receive data from its many monitoring components, the satellite is also equipped with a 433 MHz radio transmitter and receiver. All components were finally connected to a Raspberry-pi 3 to serve as the central “brain” of the satellite.

The intention of our satellite is, of course, not to send it into low-earth orbit. However, this is a sort of “proof of concept” of a miniature satellite. In this way, we can better understand the considerations and limitations that need to be taken into account when designing a small satellite. Although the monitoring data that was uploaded to our web server is received with the factory-made 433 MHz antenna, the previously made monitoring station was also tested to receive data from our satellite.

The Physics of radio telecommunication:

Information in a satellite communication channel is transmitted and received in the form of electromagnetic radiation. The information-carrying signal, called the modulation signal, is sent through a transmitter on the transmitting end of the communication channel. This device transmits a much higher frequency radio wave than the modulation signal itself (see figure 1). The actual radio waves transmitted are called carrier waves. These are simply sinusoidal electromagnetic waves that have been modulated such that they carry the information encoded in the modulation signal. The receiving end of the communication channel then extracts the modulation signal from the carrier wave, gaining access to the original information encoded by the transmitter. This process is called demodulation. Modulation can be done in many different ways. For example, the amplitude and frequency of the carrier wave can be modulated to encode the modulation signal onto it.

Amplitude Modulation (AM)

When the modulation signal is positive, the carrier wave’s amplitude is maximized. When the modulation signal is negative, the carrier wave’s amplitude is minimized. Thus, the amplitude of the carrier wave alternatingly increases and decreases with the same frequency as the modulation signal, while its frequency does not change.

Frequency Modulation (FM)

Similar to amplitude modulation, when the modulation signal is positive, the carrier wave’s frequency is maximized. Conversely, when the modulation signal is negative, the carrier wave’s frequency is minimized. Thus, the carrier wave frequency alternatingly increases and decreases with the same frequency as the modulation signal, while its amplitude does not change. Figure 1 visually shows how these types of modulation change the carrier wave.

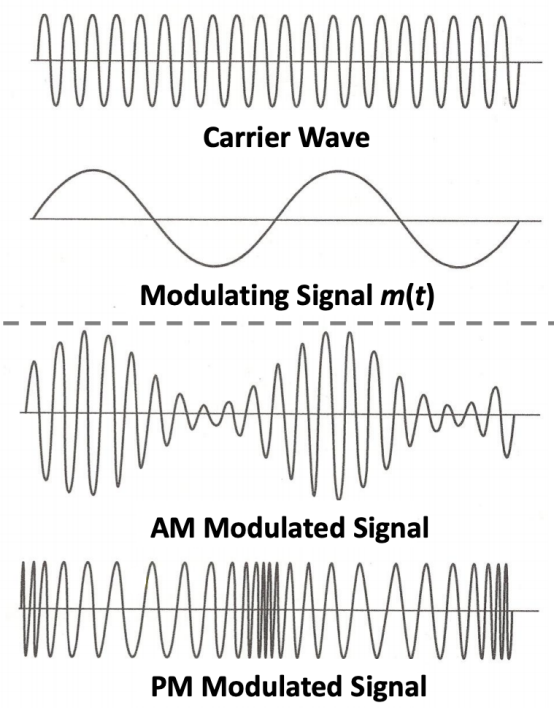


Figure 1: Different modulation types and how they change the carrier wave.

This is the essence of radio communication, but it still does not explain precisely how an electronic signal (essentially a string of 1's and 0's) is converted into an electromagnetic wave by the transmitter and converted back into an electronic signal by the receiver.

Let us start with the transmitting end of the communication channel. We assume the simplest example of radio transmittance: a sinusoidal signal transmitted by a dipole antenna. Before transmittance, the sinusoidal signal exists as a time-varying electric current in the form of an alternating current. The dipole consists of 2 metal rods of specific lengths. When this alternating voltage is applied to the dipole, it will alternatively charge one end of the dipole positive and the other end negative. In other words, the polarity across the dipole is constantly reversing with the frequency and amplitude of the AC signal. This alternating potential over the dipole, in turn, generates an alternating electric field with the same frequency. Of course, the electrons in the circuit constantly move from one end of the dipole to the other. Ampère's law tells us that moving electric charges induce a magnetic field. Since the direction of the current over the dipole is constantly alternating, the dipole will generate an alternating magnetic field in addition to the electric field.

The combination of the induced alternating magnetic and electric fields is what we call electromagnetic radiation. This radiation does not require a medium to travel through since changing electric fields produce magnetic fields and changing

magnetic fields produce electric fields. In essence, the electric and magnetic fields propel each other forwards, so there is no need for any medium to travel through. If the electromagnetic wave moves in the x-direction and the electric field points in the y-direction, then the magnetic field will always point in the z-direction. Figure 2 shows a visual representation of the magnetic and electric components of such an electromagnetic wave. This also indicates the perpendicularity of the electric and magnetic components of the radio wave.

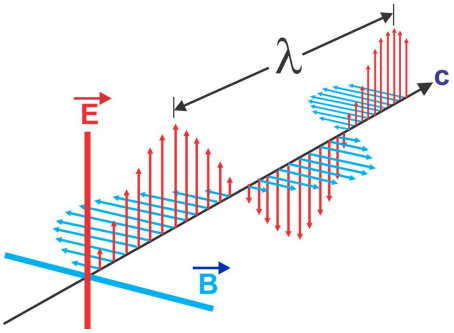


Figure 2: Schematic representation of an electromagnetic wave.

As stated before, the length of each half of the dipole needs to be very specific. The length of the entire dipole will always be exactly one-half of the wavelength of the electromagnetic wave produced by it (see figure 3). This is because the polarity across the dipole as the whole takes exactly one time the period of the AC signal to go from a maximum to a minimum and back to a maximum. Thus, the period (inverse of frequency) of the generated electric field is the same as the AC signal. Since we can approximate the flow of electrons in the dipole to be at the speed of light, we can simply calculate the wavelength of the desired carrier wave. The dipole length should thus be one half of this wavelength since the electrons need to flow across the entire dipole two times in one period. The same principle applies to the induced magnetic field.

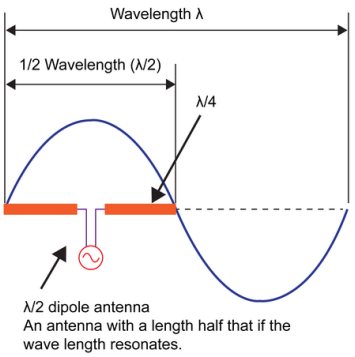


Figure 3: The dipole's dimensions are made to match the electromagnetic wave.

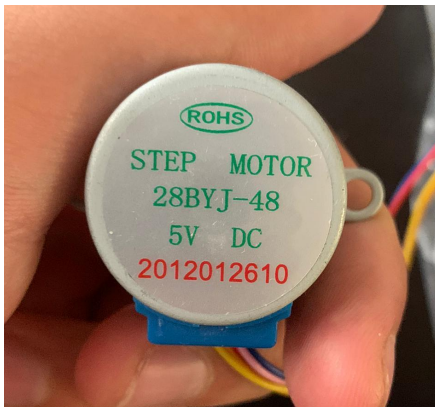
The receiving end of the communication channel work in the opposite way. The same dipole used for transmittance can be used for receiving a radio signal. When

the carrier wave reaches the dipole, the dipole will experience an alternating electric field strength. Analogous to how an alternating potential difference across the dipole will induce an alternating electric field, an alternating electric field strength experienced by the dipole will induce an alternating potential difference across the dipole. This causes a slight alternating current across the dipole circuit. This alternating current thus carries the information of the carrier wave, and the information is successfully transmitted from one place to the other.

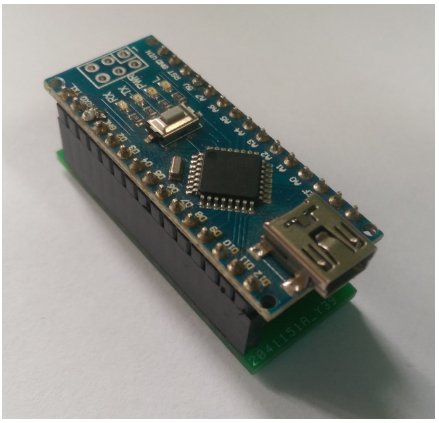
Materials

CubeSat Design

- One 3D printer: Creality Ender 3
- PLA (polylactic acid) plastic filament
- CAD software
- Programming software (Arduino IDE, Python libraries, etc...)
- Electronic components:
    - 1x Raspberry Pi 3 Model B: a microcomputer the size of a credit card that allows for general computing. Has a series of pins that allow it to communicate with external devices like sensors, cameras and antennae
    - 1x Raspberry Pi camera: a small near IR camera made to run on the Raspberry Pi
    - 1x 28BYJ-48 stepper motor and associated ESC

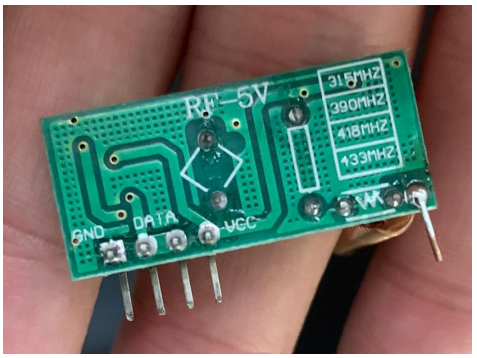


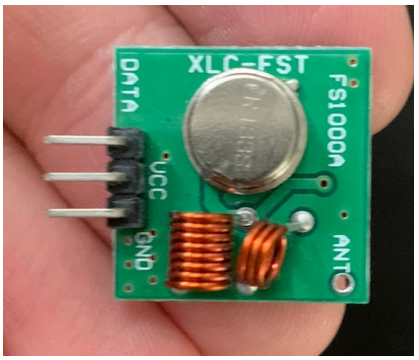
    - 1x Arduino Nano: a microprocessor, allowing the automation of tasks with programmation. Like the Raspberry pi, the arduino has a series of pins that allow the control or use of outside modules. The Arduino Nano board is specifically made to be small and compact



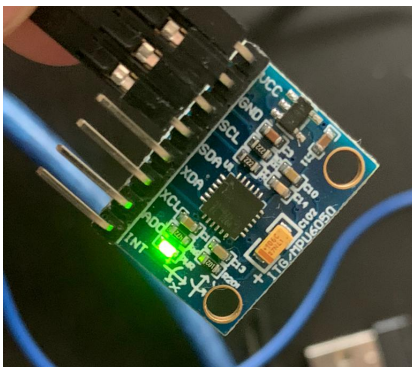
- 1x GT-U7 GPS receiver: a module capable of receiving input from many satellites. The output data includes latitude, longitude and elevation, plus all kinds of extra information about the satellites and geo-localisation.

- 1x FS1000A pair: a radio wave transmitter/receiver pair (433MHz) that allows the transfer of digital data through amplitude modulations over distances up to 500m in optimal circumstances (clear line of sight with minimal radio interference).

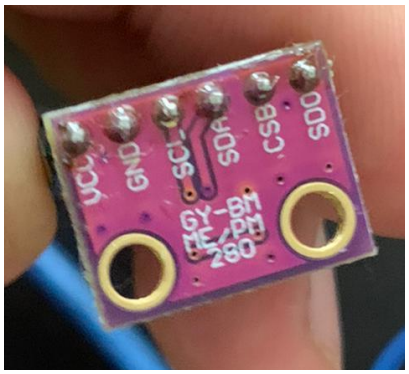




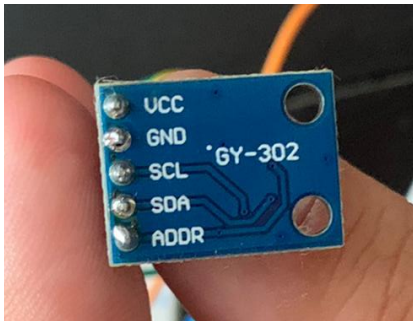
- 1x gy-521 gyroscope and accelerometer module with 6 degrees of freedom



- 1x BMP280: a low energy consumption temperature and pressure sensor



- 1x BH1750: a digital ambient light sensor



Antennae

- Coaxial cable (50 ohms)
- Aluminum or brass rods (approx. 4 mm diameter)
- Wooden poles for structure
- General electrical tools (Coax connectors, electrical tape, chockablock connectors, screws, etc…)
- 1x Software-defined radio (we used the Nooelec RTL-SDR but any brand is acceptable)
- 1x FM bandstop filter
- 1x Personal computer with required software downloaded (see methods)

Methods

CubeSat

Design

1. The first thing to be done was to set some goals and constraints for CubeSat's design and functions. In order to do this, project limitations, available equipment and the official CubeSat standards were considered.
2. After setting the goals and constraints, a CAD model had to be made that could be printed.
3. Once a part had been modeled, it could be sent to the person in charge of the 3D printing, who would check its viability with regards to the printing process. If no major issues were found, the model would be *sliced* so that a file could be made that the 3D printer could use.
4. The printed product would be test-fitted with relevant hardware and checked with the designer to see if the intent had been matched.
5. If there were issues, the part would be modified in CAD. Otherwise, the designer could focus on other aspects of the design.
6. Steps 2 through 5 were repeated until a part was deemed sufficiently functional. The CubeSat as a whole was then assembled and tested according to the goals.

Sensor Data and General Software

The Raspberry Pi in this satellite was made to work with no direct user input like a screen. As such it was necessary to be able to connect to it from distance using SSH. This gave us total control over the computer by the means of the terminal, allowing us to proceed to the next steps.

To receive and then transmit the sensor data, the Raspberry Pi and Arduino have to work together. The two are connected through USB and communicate using a serial protocol. The code on the raspberry pi was written in python, which gives very powerful and easy to implement tools to work with the GPIO pins, and in C++ on Arduino, as it is not possible to use other languages, except C.

All of the sensors as well as the stepper motor module were connected to the Raspberry Pi, and only the RF transmitter was connected to the Arduino. This was done because some powerful radio communication protocols have been written for the Arduino but not for the Raspberry Pi.

All of the sensors happen to use the I2C serial protocol to communicate. It allows for multiple slave-master device connection connections contrarily to other methods that

sensors sometimes use. This made the connection task easier as with I2C it was sufficient to connect all the sensors in parallel to the SDA and SCL pins (fundamental to the I2C protocol) on the raspberry pi. It is important to note that the sensors are 3.3V, and as such are easy to fry if connected to the wrong pins. When connected, each sensor gets a unique ID that can be looked up by using the “i2cdetect” command on the raspberry pi. Knowing these it was then possible to write code that would request the data automatically and save it in memory. That data had then to be sent to the Arduino that would process it and then send it through the RF transmitter using amplitude modulations.

This data was then made to travel to a receiver module connected to another Arduino nano, but this time connected to a server (a laptop or another Raspberry Pi) that would then log the incoming data and upload it to a web page and plot it in multiple fashion depending on actual data. For this Flask and Javascript were used.

Another part of the CubeSat that needed to be programmed was the solar panel system. The motor itself is bipolar and thus requires 4 pins to be attached to the raspberry pi. The code itself is not very complicated and it takes values from the ambient light sensor to open or close the solar panels.

Camera and Image Transmission

In order to receive images from the PiCamera, initially the plan was to convert a jpeg file taken with the camera, convert it into a string, split it up into smaller chunks of data and then send it via the radio transmitter.
However, as any errors in data transmission might result in the inability to decode a decent image on the ground station, we decided to try sending the images via SSTV.

Slow-Scan-Television (SSTV) is a method of transmitting still pictures via radio.
Similar to regular analog television, in SSTV the image is scanned and transmitted line by line from left to right.
In the case of coloured images, each colour component is sent as an individual line.
While many SSTV modes use RGB colour channels, some modes use YUV instead.
In YUV, one channel codes for luminance (Y) while two channels encode chrominance, one Blue-Y, and one Red-Y.
As this colour coding is based on human perception of colours, chrominance can be encoded more efficiently, allowing for a smaller bandwidth.Each line is sent as a tone modulated between 1500 (black) and 2300 Hz (white), depending on the intensity of each channel. Between each line (all colour channels), a 5 millisecond long synchronisation pulse with a frequency of 1200 Hz is inserted. At the start of the transmission, a Header is sent for calibration, and to specify the SSTV mode.
Next to the various images exchanged by amateur ham radio operators, SSTV and similar technology was also utilized to transmit images during some of humankind's

first space missions. Prominent examples include the flight of Yuri Gagarin in 1961, as well as various Apollo missions.

The program “PySSTV”, written by András Veres-Szentkirályi was used to convert a jpeg taken by the PiCamera to an SSTV modulated wav file.
To transmit the wav file, we first tested the software rpitx.
It uses the raspberry pi's GPIO pin 4 as an antenna, which, by attaching a few decimeters of wire, can transmit a fairly strong signal.
While the transmission quality itself was good with this program, the transmission would abruptly end after random time intervals.
Unable to resolve this issue, a program called FM transmitter was used instead.

Decoding of the received audio files was done using MMSSTV and SDR#.
Images decoded from the sound of SDR# were typically warped beyond recognition when played over the speaker. However, using direct input from the sound card, we managed to transmit reasonably sharp images.

While this makeshift antenna was sufficient to demonstrate SSTV use for image transmission, for further testing, or long-term operation of the satellite, a low-pass filter would need to be added, as this setup pollutes the shared airspace with harmonic interference on multiple frequencies. The "chirp" testing function of rpitx showcases this pretty well.

Antennae

Hardware

There are many design options for the antennae designs for picking up NOAA weather satellite signals: v-dipole, double-cross, turnstile, quadrifilar, etc. In this project, we chose to construct a v-dipole and double cross antenna.

The v-dipole antenna is by far the most straightforward design to construct. The design of a v-dipole is essentially the same as a dipole antenna. The only difference here is that the angle between the two dipole arms is 120° instead of 180°. As mentioned in the theory section, the total length of the two dipole arms needs to be one-half of the wavelength of the incoming signal. This means that each rod needs to be approximately one-quarter of the wavelength. NOAA satellites transmit radio signals with a center frequency of about 137 MHz. Therefore, one-quarter of the wavelength of the carrier waves is about 50 cm.

1. First, the 4 mm diameter brass rods were cut to approximately 50 cm. The ends of the 2 brass rods were then twisted to 90°. For the central mount of the antenna, a 2-meter wooden bar was used. These twisted ends were then inserted into a chockablock, which was attached to the end of the wooden bar.

2. One end of a coaxial cable was then stripped away such that the center core can be separated from the metallic shielding.
3. The center core was then attached via the chockablock to one dipole arm, while the metallic shielding was attached to the other arm.
4. On the other end of the coaxial cable, a standardized coax attachment was fitted. This attachment makes it easy to connect the coaxial cable to other hardware.
5. The coax cable was first connected to the FM band-stop filter, which was in turn connected to the software-defined radio (SDR). The SDR has a USB attachment on the other end, allowing us to connect it to a PC easily. Before every field test, a protractor was used to make sure the 120° angle between dipole arms was maintained.

The turnstile antenna is a much more complicated design. In essence, it consists of 4 dipoles in pairs, mounted across from each other and tilted at a 30° angle.

1. To build the mount, a wooden pole was used with two cross supports on one end. Just like the v-dipole, eight dipole arms were cut to approximately 50 cm.
2. To each of the four ends of the cross supports, a chockablock was attached, and two dipole arms were connected. Each dipole was then tilted to 30° with the vertical axis in a counter-clockwise fashion.
3. Instead of using an arbitrary coaxial cable length, this design requires precise coaxial cable lengths. Two coaxial cables were cut to one-fourth the wavelength, and two coaxial cables were cut to one-half the wavelength.
4. Again, the ends of the coaxial cables were stripped. Two opposing dipoles were attached to the quarter wavelength coaxial cables, while the other two dipoles were attached to the half-wavelength coaxial cables. The exact measurements of the coaxial cables were chosen such that the signal from one pair of dipoles is delayed at exactly one-quarter wavelength. This allows all four dipole signals to line up and not create destructive interference.
5. The other ends of the four coaxial cables were connected as shown in figure 4.

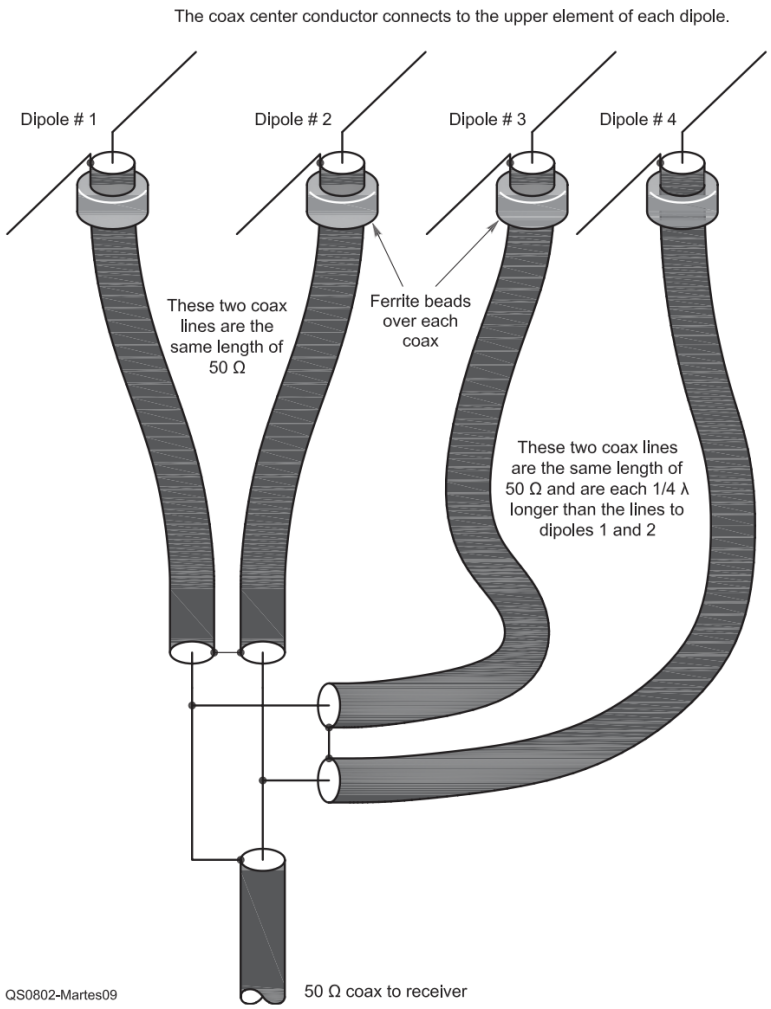


Figure 4: The layout for the coax cables in a double cross antenna.

Analogous to the v-dipole design, the final end of the main coaxial cable was then connected to the band-stop filter, which was then connected to the SDR.

Software

Orbitron is software that was used before field testing to know when exactly there was an NOAA satellite passing over Maastricht. These passes usually take about 10 minutes. Three software were used to convert the AC signal from the antennas to an image; SDR-sharp, Audacity, and WXtoIMG.

1. First, SDR-sharp was used to control the software-defined radio. SDR sharp essentially turns your PC (connected to the SDR) into a radio, allowing us to receive the NOAA signals as audio files.
2. During a satellite pass, the audio signals were recorded and saved as a wav file.
3. Then, using audacity, the sample rate of the recorded audio was changed to 11025 Hz, because the sampling rate from the original SDR-sharp recording was too low to use later.
4. Finally, WXtoIMG was used to convert this processed audio file into a black and white image. WXtoIMG proved to be an instrumental piece of software as it also allowed us to add country borders and false colors to our images.

Field testing was done in open areas with not too much radio interference to make sure we got a long enough satellite pass with the best possible resolution.

Results & Discussion

CubeSat Design

Goals and Constraints

When setting goals for the CubeSat, it was decided that it should be able to carry out three functions that form the basics for many small satellites. It should:

1. Take periodic measurements with an array of sensors and take pictures with a small camera.
2. Transmit the results of the measurements and the camera images.
3. Tilt its solar panels outward.

Then, constraints were put in place. These were guided by the CubeSat organization, NASA's recommendations, and project period limitations. An important constraint was that the main body should fit inside a 10 x 10 x 10 cm cube, which is the standard size for a 1U CubeSat. The limited space that was left was reduced further by the rails that all CubeSats must-have in the corners of the satellite so that it can be placed in launch assemblies (The CubeSat Program et al., 2020). Unfortunately, this is also where the first problem arose: the solar panels were ordered before the project started and were 15 x 13 cm. This meant that they could never neatly fold into the satellite's allotted space. It was then decided that they would simply be attached to the sides of the cube, and any mechanism to fold them out could always be used for smaller ones. The structure that held the components would also need to be entirely 3D-printable. While the use of PLA meant that the satellite would never survive outer space, it was decided that this was not a problem since our goal was to see what was possible and to make a prototype. According to NASA, creating a fully functional CubeSat would take several months at the very least, and launching was not in the cards for this project anyway (NASA, 2017). Due to other limitations of PLA, it was not practical to use gears to move the solar panels, so those would be avoided if possible. CubeSats often do not use motors to fold out their solar panels, if the solar panels even fold out in the first place, but it was decided that the goal would be interesting and fun to work on. The satellite also had to be relatively easy to assemble, since prototypes often need repeated re-assembly. It was further decided that the component mounts need not survive any vibration testing, so they could be rather loose and use materials like tape or zip-ties. Finally, it was also decided that the motor did not need to provide enough torque to lift the solar panels, merely enough to move them. Naturally, the lack of gravity in outer space meant that the mechanism would probably work there if the motor was tested sideways and unhindered by gravity's pull. There are plenty of other restrictions on CubeSat functions and design that we decided to ignore for the purposes of this project.

Basic Structure and Component Mounting

With these goals and constraints, a basic structure was made (see figure 4).

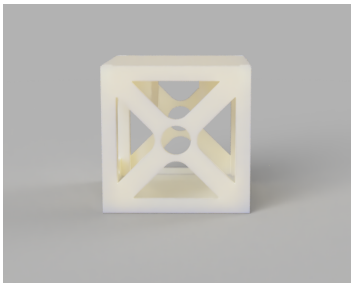


Figure 4: A simple outer shell.

This was a simple shell to encase the components, and it was designed to strike a balance between materials used, structural strength and aesthetics. It was not printed yet, because it was far from ready to be used. The first changes were the separation of most of the outer shell and a plate from the bottom. The rest of the design would be based on placing mounting plates for the electronic components to the bottom plate, and then placing the rest of the cube over that (see figure 5 and video 1 in Appendix [LETTER]).

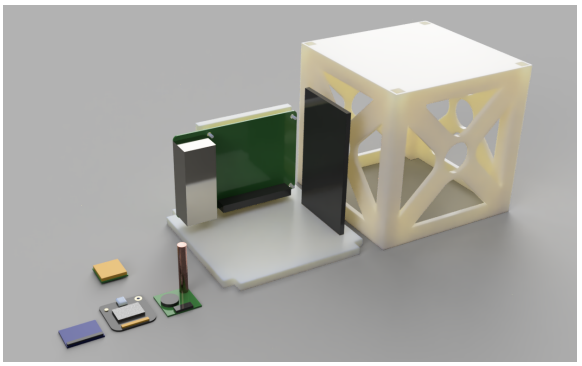


Figure 5: The general idea for component mounting in an early stage. The electronic components on the left would be placed in slots on the black part that protrudes from the bottom plate.

After adding models of electronic components and vertical mounting plates for them, the basics were ready and a test print was made. The first prints failed, but eventually a bottom plate and mounting plates for the Raspberry Pi and the sensors were made and assembled. The design was considered to be viable, and the focus shifted to the solar panels and how to make them fold in and out.

The First Mechanism for Solar Panels

Due to the limitation that gears should be avoided if at all possible, finding a way to transfer mechanical motion was more difficult than anticipated. However, after

discussing a basic design, hinges were made in the CAD model that would form the foundation of this solution. In CAD, a similar design was implemented and tested using the joints system that Fusion 360 has (see figure 6 and 7 and video 2 in the Appendix).

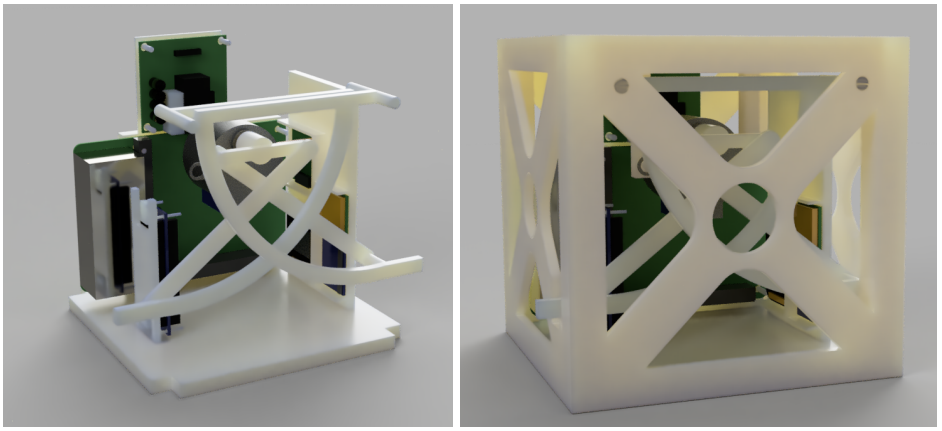


Figure 6 and 7: the mechanism without and with the outer shell.

It soon became clear that despite some limitations such as not being able to make the solar panels tilt up 90°, the mechanism would in fact be able to turn the rotation of one motor into the movement that the two solar panels needed. Printing it, however, was a complete failure. The PLA simply was not up to the task. Had it been sheet metal or aluminium, the solution would have been fantastic. However, a new mechanism had to be devised.

The Second Mechanism for Solar Panels

In order to make a design that would function with PLA, some of the constraints that the first mechanism satisfied had to be abandoned. Without those constraints, mechanism two was born. Unlike its predecessor, the mechanism did not fit inside the CubeSat, was not reversible and required many other components to be moved. However, it made up for these shortcomings by actually working. The new mechanism used a pulley to pull string into the satellite. The string was connected to the solar panels, and it pulled them into a position that was parallel to the bottom plate with little effort. The model can be seen in figure 8 and the printed and assembled CubeSat can be seen in figure 9.

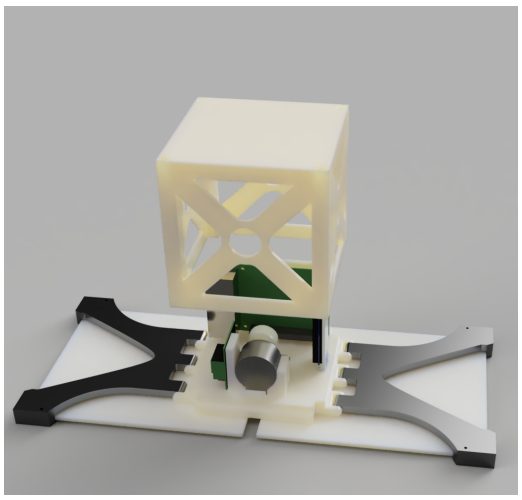
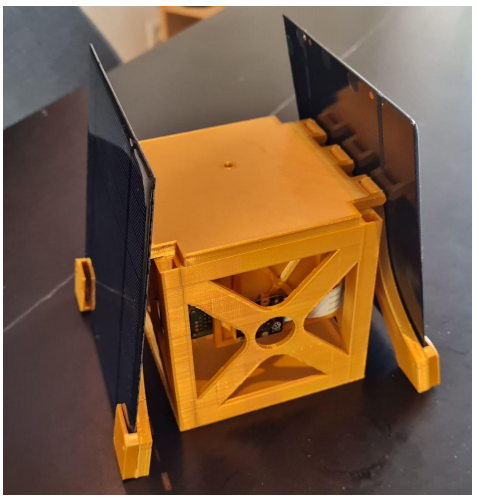



Figure 8 and 9: the second mechanism as a model and assembled after being printed.

Evaluation of the Design

The design is merely a very early prototype and study object, used to learn how spacecraft are designed. Thus, it need not be ready for space. That is a good thing, because it certainly isn't. At this moment, it would: not fit in a launch capsule, fall apart inside the rocket, not have much control over its solar panels once in outer space, be rendered useless by solar radiation and not even be able to transmit a signal to Earth if none of the aforementioned was the case due to the transmitter's range of approximately 500 meters. However, we have a cuboid object that can contain electronic components and that almost fits within our constraints, so further engineering and iterations could at least make a neat toy.

CubeSat Sensors and Software

The coding of the sensor data extraction was for the most parts simple, as the sensors output by themselves a value that is directly understandable. The two exceptions were the GPS module, which returned lines and lines of specially formatted data (see appendix for an example) and the Accelerometer/Gyroscope, which returned values in angular velocity and linear acceleration that needed to be processed to return axial inclination values.

The GPS issue was solved when it was discovered that the formatting was following the NMEA rules for encoding and that most of the text output was of no use for us. It is also probably worthy to note here that even though it might sounds strange to have a GPS module on a satellite prototype, they have actually been used before to track spacecrafts (ESA, 1997) and satellite (Knight, 2001) and as such could be a useful tool for certain applications for an experimental satellite.

The sending of the data through the wireless modules was a little bit more difficult. Those little devices have a low bandwidth and are generally used in the same

context as the little RF remotes for cars or garage doors. As such they have a very limited bandwidth and cannot send huge amounts of data (reason we tried switching to SSTV as a mode of image transmission). We also used a premade RF Arduino library from SurfShark and this, even though it saves a lot of time as we don't need to write our own protocols, allows us limited control over the sent packets. As such when the packets were too big the arduino would just ignore certain parts of the message and sometimes the message itself. These are all programming mistakes on our part and had we more time it would not have been very difficult to fix, but our quick fix was to simply slow the speed of transmission as well as the size of the packets. As of now the data is transmitted every second and cut into three smaller blocks.

The sensors themselves were easy to quite easy to connect into the circuit visible below. It shows the whole wiring of the satellite and sensors with the exception of the solar panels and the batteries. An issue that was encountered however was the uniqueness of I2C addresses in the computer memory. Each sensor has one, set by the manufacturer, and it has to be unique for the computer to recognize the individual devices. Generally they can be changed by powering certain pins of the modules in question, but when they are from cheap sources and that the data sheet is difficult to find or non-existent then it complicates the task. In this case one module had the same address than another, but by changing it to a replacement we had fixed the issue very easily.

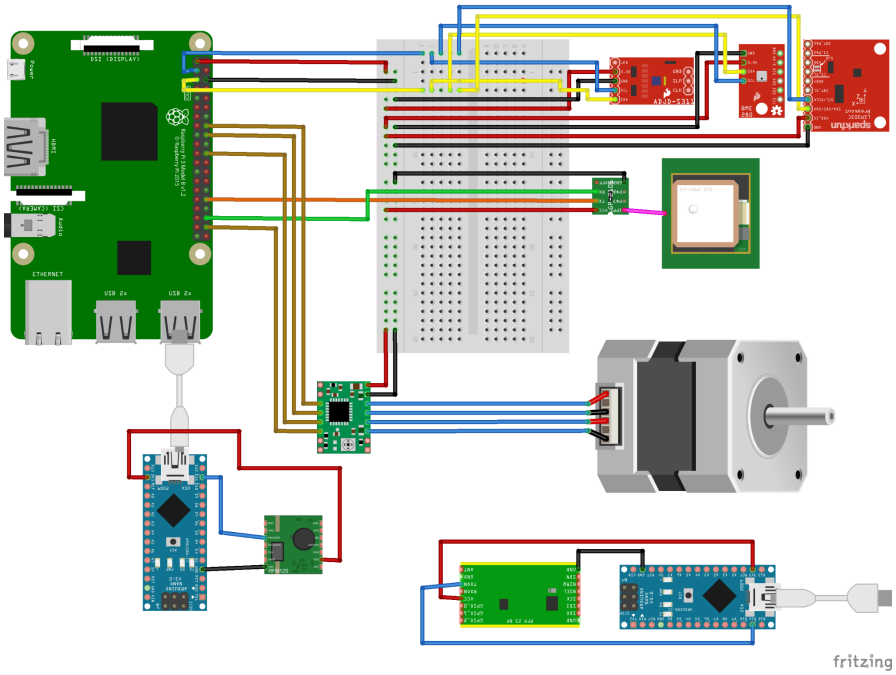


Figure 10: Schematic of the wiring for the CubeSat components.

On the receiving end of the project, we built a nice web server system that collects the data once it arrives through the Arduino nano and logs it into a SQLite database. From there there are a few very easy scripts that take the data and publish it on a

small website using Flask and Javascript to have beautiful plots and data representation. It's very basic but it's also very easy to expand it much further with only a little bit of effort and time. An idea that we had was also to create an interactive model of the satellite that follows it's rotation live thanks to the accelerometer data. It was started but we did not manage to finish that due to complications between the sending of data from the pins to the webpage without going through a database.

All the code running in the Raspberry Pi, arduinos and web server can be found in the following Github repository:
https://github.com/Jonastjep/TinyProjects/tree/main/Satellite_Project

Antennae

Notable passes

Below are some of the results received from NOAA 15, NOAA 18, NOAA19, and NOAA20 passes captured during the project's duration.

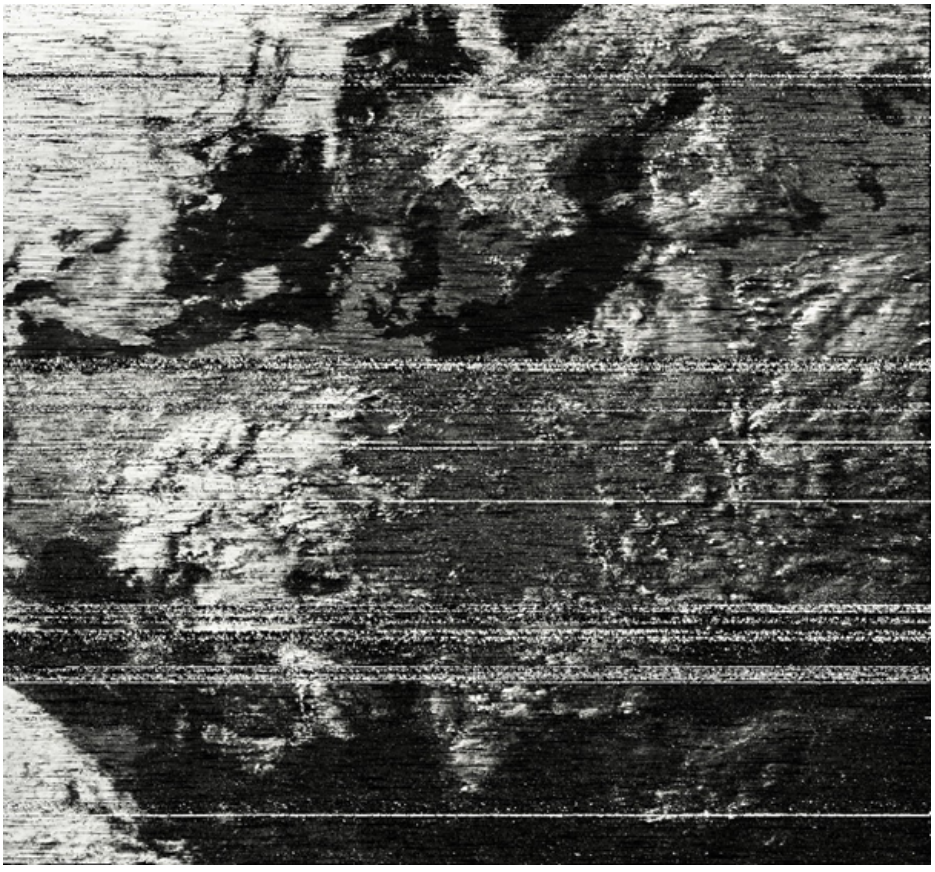


Figure 11: First successful pass – Captured by V-Dipole antenna - NOAA18

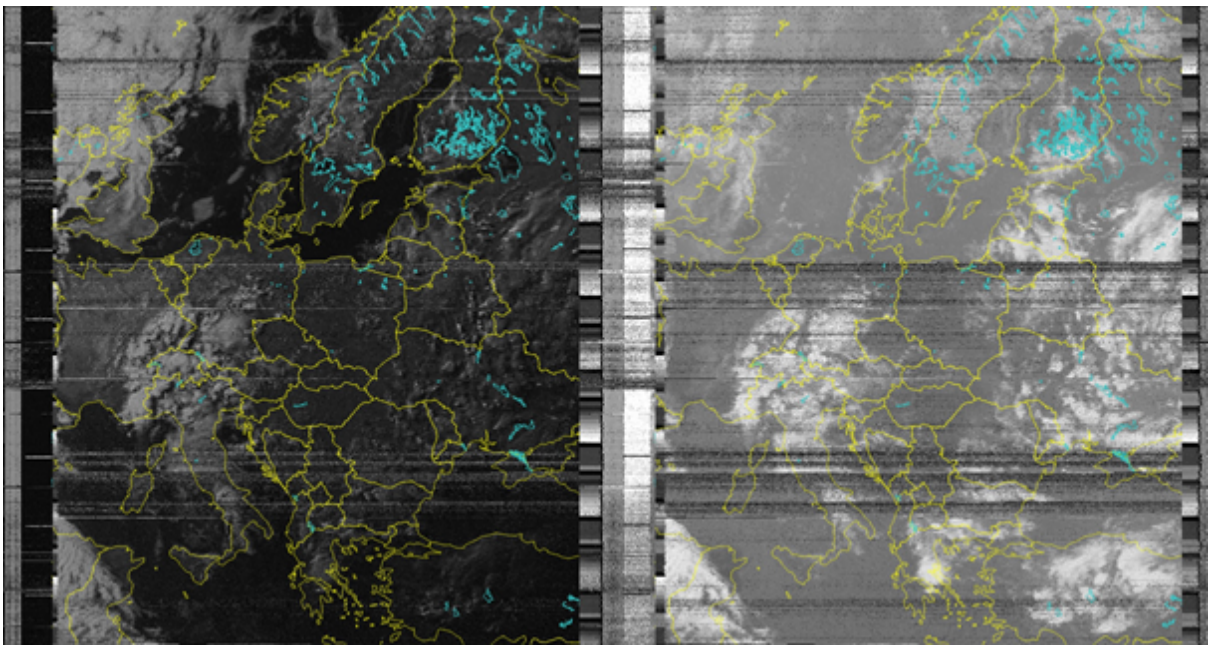


Figure 12: Best pass captured by V-Dipole antenna with border and lake outlines present – Captured by V-Dipole antenna - NOAA18

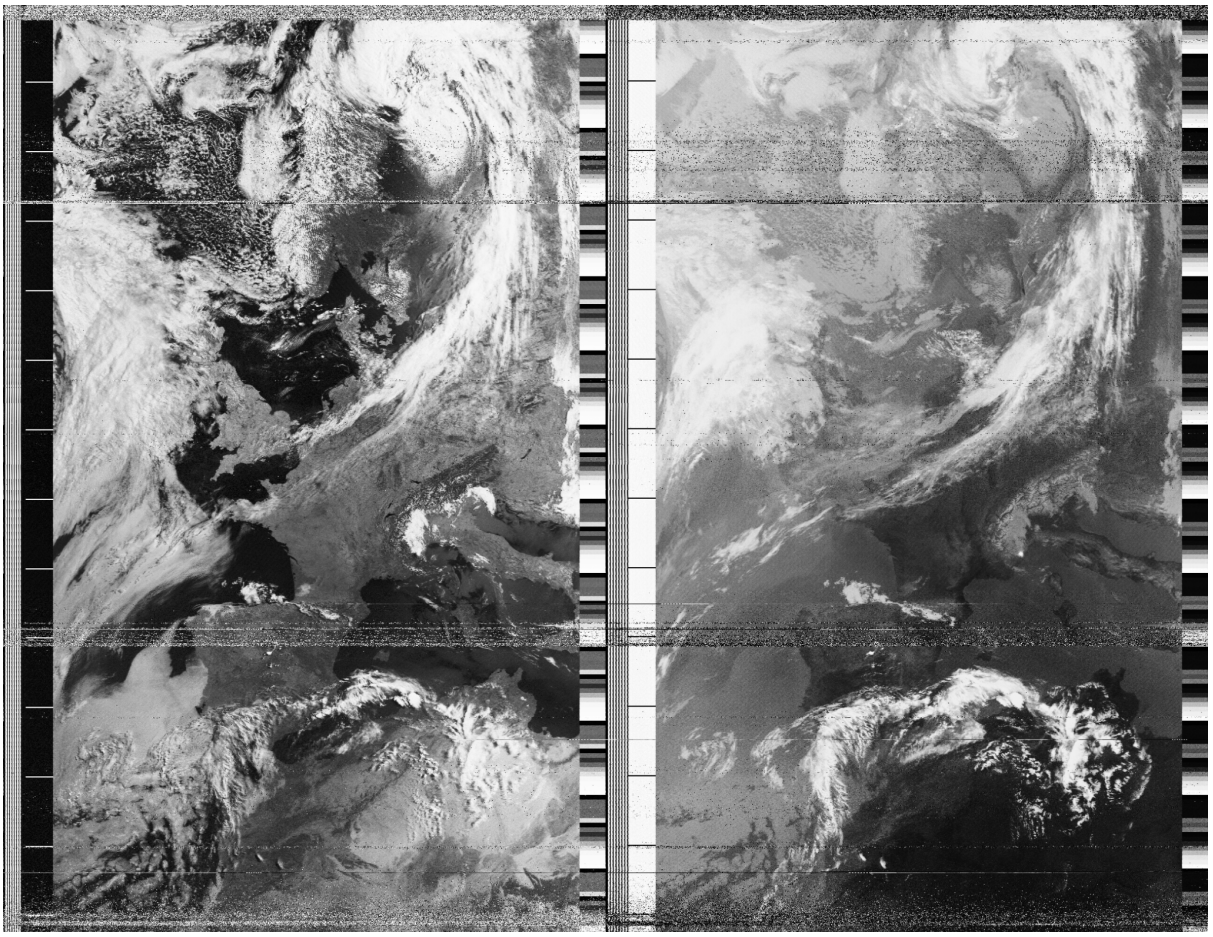


Figure 13: Best long pass – Captured by Double Cross antenna - NOAA19

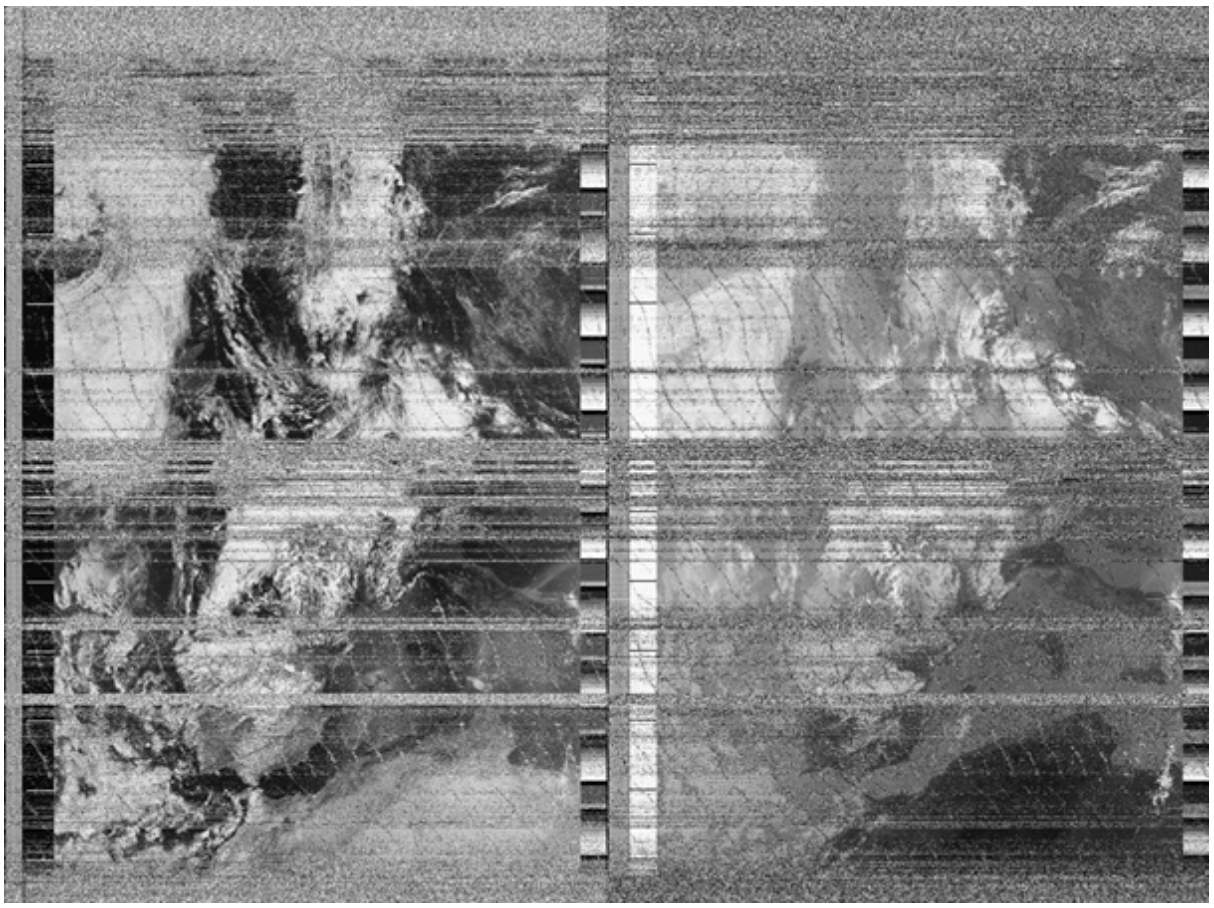


Figure 14: Pass captured on MSP building roof with radio interference visible – Captured by Double Cross antenna - NOAA19

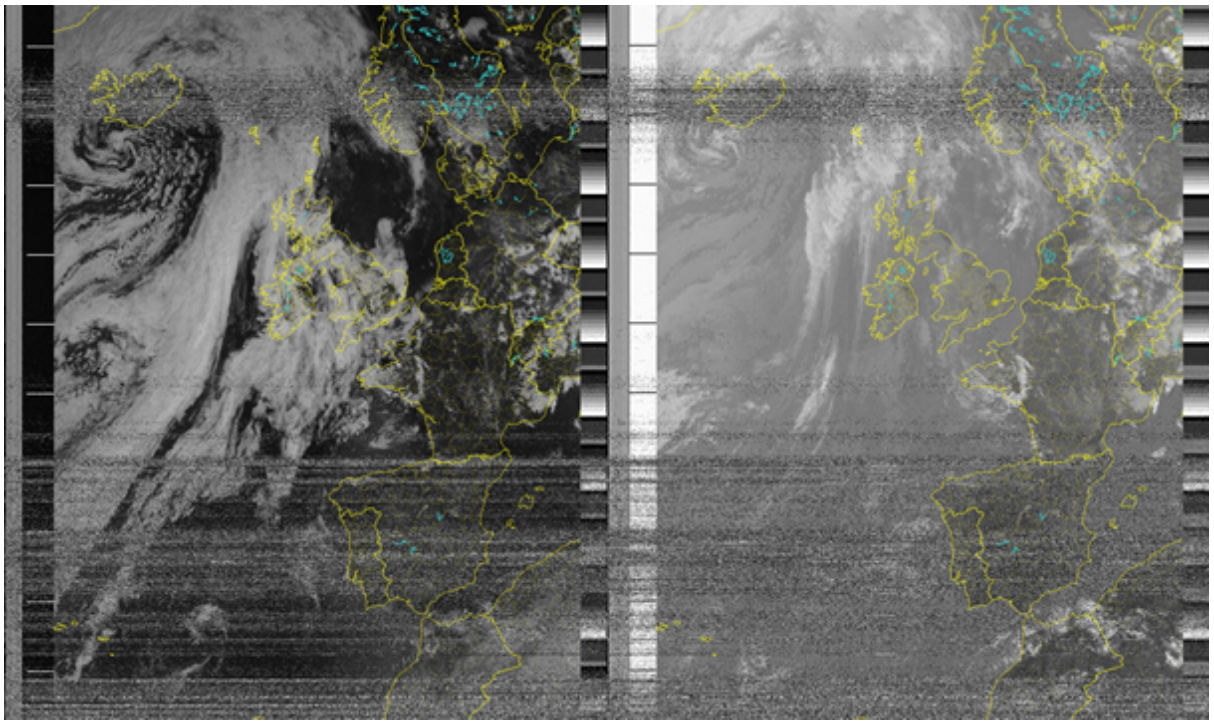


Figure 15: First pass captured by Double Cross antenna – Captured by Double Cross antenna - NOAA18

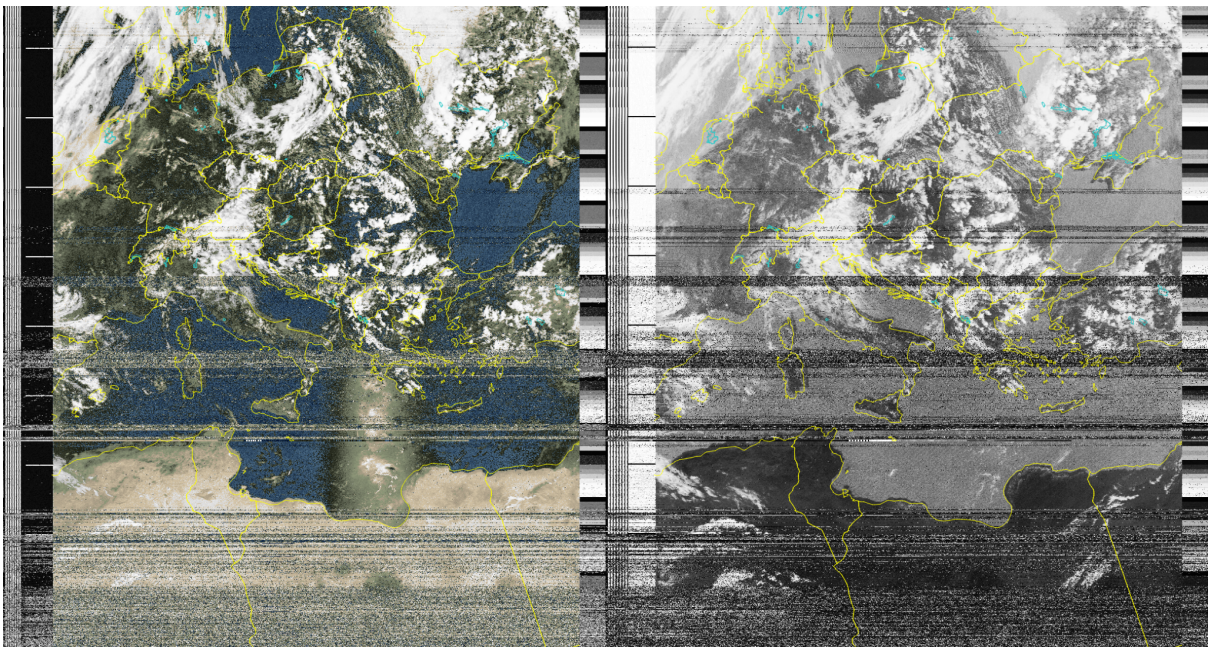


Figure 16: False-colour pass - Captured by Double Cross antenna - NOAA18

Radio interference

Something that was struggled with somewhat during a couple of passes was the presence of radio interference. Even though the antennae were constructed to receive a very specific frequency, that being 137.1-137.9 Mhz, they still got a lot of interference at times. This could be noticed with certain examples, such as there often being a slight background noise which is likely also to be in part attributed to the sun, which is a big source of radio interference at daytime, when all passes were recorded. Another source of interference which was picked up during one pass was Maastricht Aachen Airport. During this pass at a certain point voices could be heard during the latter parts of the pass, which is plausible as air traffic control is allowed to use a wide range of frequencies, including the one the NOAA satellites operate in.

Different Antenna

Due to the construction of two different types of antennae, the V-Dipole antenna and the Double Cross antenna, a comparison can be made between the two. As expected the Double Cross produced the better results, as it has a better field of reception and is omnidirectional. Along with this it had better dynamics because as a result of it being omnidirectional it only needed to be held upright, as opposed to the V-Dipole which had to be constantly held directionally.

Fake Colour

Because NOAA satellites only have a visual and an infrared camera on them, colour is something that has to be added in later over the picture if wanted. The problem with this is that it is artificially done by the software used to decode the images. This means that it is prone to errors when images are not perfectly clear. Along with that the sun's reflection on the ocean and thick cloud cover can also be interpreted wrong by the software. In most cases the software attributed the colour of a landmass to the sun's reflection. Because of this the Fake Colour did not grant satisfactory results and thus was not used in most of the images.

Conclusion

CubeSat Design

The design is nowhere near ready to be launched into low Earth orbit, as would be expected given that there was a time limit of three weeks and the team only having access to basic consumer-grade materials and tools. However, it forms a basis for further engineering and the process by which it was created was able to get surprising results in the given amount of time. In the end, some constraints that were set had to be lifted to make the goals achievable. As a product, it is a minor success at best. As a learning process, it was a great success.

Sensors and Coding

Overall the software part of the satellite was a success. Most of the functionalities that we had imagined at the beginning have been implemented, even if sometimes not in the most optimal way. The satellite is capable of extracting it's sensor data and sending it to the ground station, which in turn logs them into a database and a web server. A few issues remain, namely the speed at which the transmissions take place and a few minor transmission errors, but seeing that it's mostly a bandwidth and library issue and that satellites don't need to send this type of data much more than every second we find that this is not a huge flaw, at least not up until now.

Antennae

The initial aims set out for the satellite part were only half achieved. Initial objectives were to receive and decode both images from the NOAA satellites and Meteor M2 images. Although objectives for the NOAA passes were achieved, there was no success in receiving meteor M2 images. Although the Double Cross antenna was more than capable of receiving the Meteor M2 signals, decoding proved to be a challenge that was not able to be overcome. This issue was to be found somewhere in the software but in the 3-week timeframe there was insufficient time to work out the software flaw.

Due to the construction of two different antennas, the V-Dipole antenna and the Double Cross antenna, it was possible to compare the differences between both antenna types. Not only was the quality on the Double Cross antenna noticeably better than the V-Dipole, but it was also easier to operate since the Double Cross was omnidirectional, only needing to be held upright for good reception. Possible sources of interference in our images might have been nearby radio towers. Additionally, some satellite passes were relatively close to the horizon, causing the satellites to pass behind nearby buildings for brief moments. These temporary signal losses can be seen on some images as grey bars across the image. Future research

should be done using more complex antenna designs and field testing should be done in areas with a much lower horizon and less radio-interference.

References

ESA. (1997). Satellite Navigation Using GPS . ESA Bulletin Nr. 90. Retrieved from https://www.esa.int/esapub/bulletin/bullet90/b90mur.htm

NASA. (2017, October). CubeSat 101: Basic Concepts and Processes for First-Time CubeSatDevelopers. https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf

Knight, W. (2001). Satellites use a GPS system from above. NewScientist

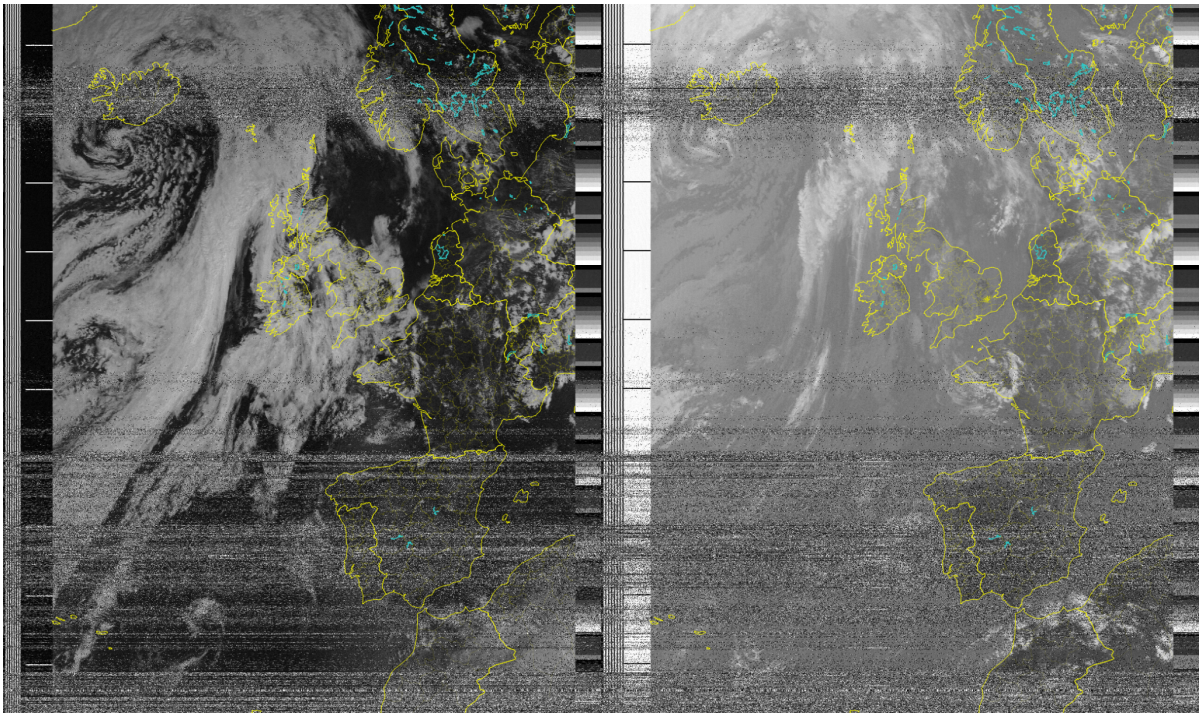
The CubeSat Program. (2020, July). CubeSat Design Specification (1U-12U) Rev. 14. https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/5f24997b6deea10cc52bb016/1596234122437/CDS+REV14+2020-07-31+DRAFT.pdf

Appendix

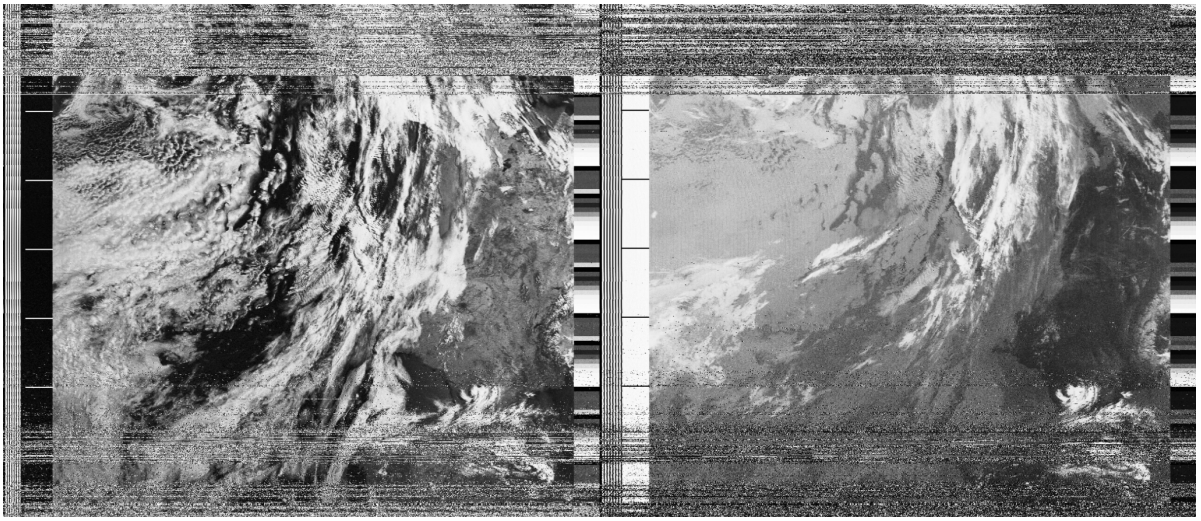
CubeSat Design Videos

Assembly 1:
https://drive.google.com/file/d/1uJAJRbLI2cILMGYJQJVDIKoFNy-CQO8L/view?usp=sharing
Mechanism 1:
https://drive.google.com/file/d/1Uf_3mDEs564jpHimSnsUVNWaT16dC3eS/view?usp=sharing

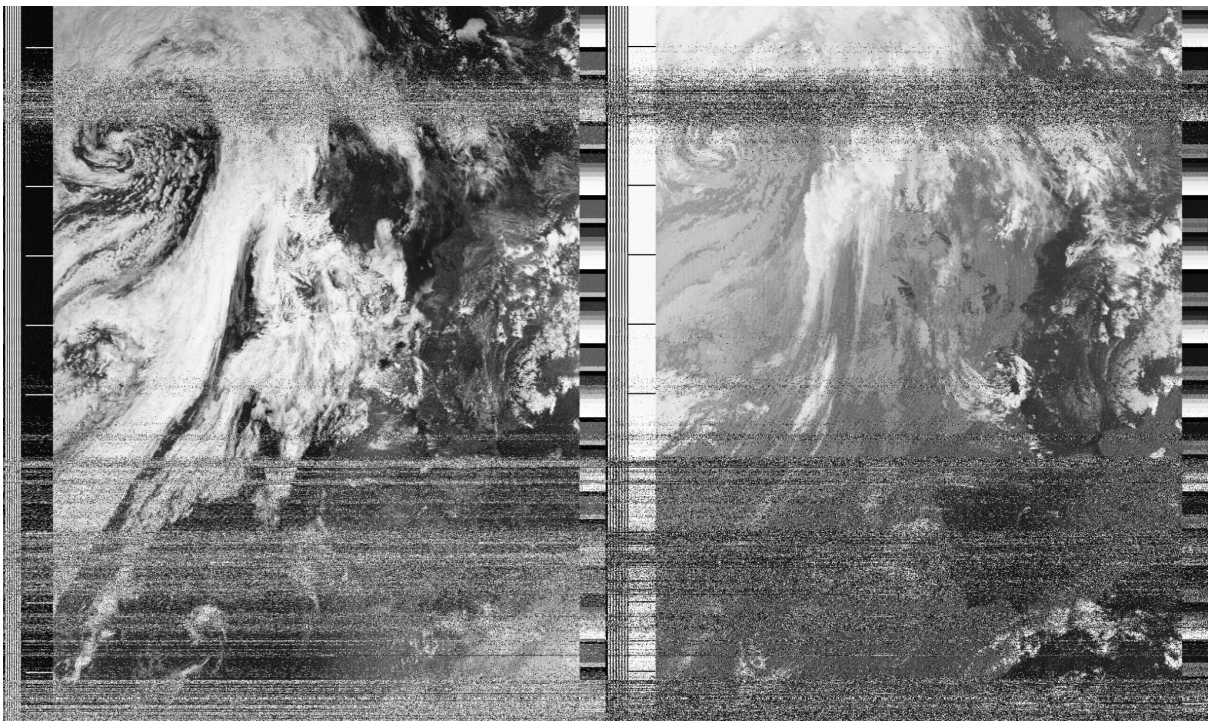
Satellite pass images



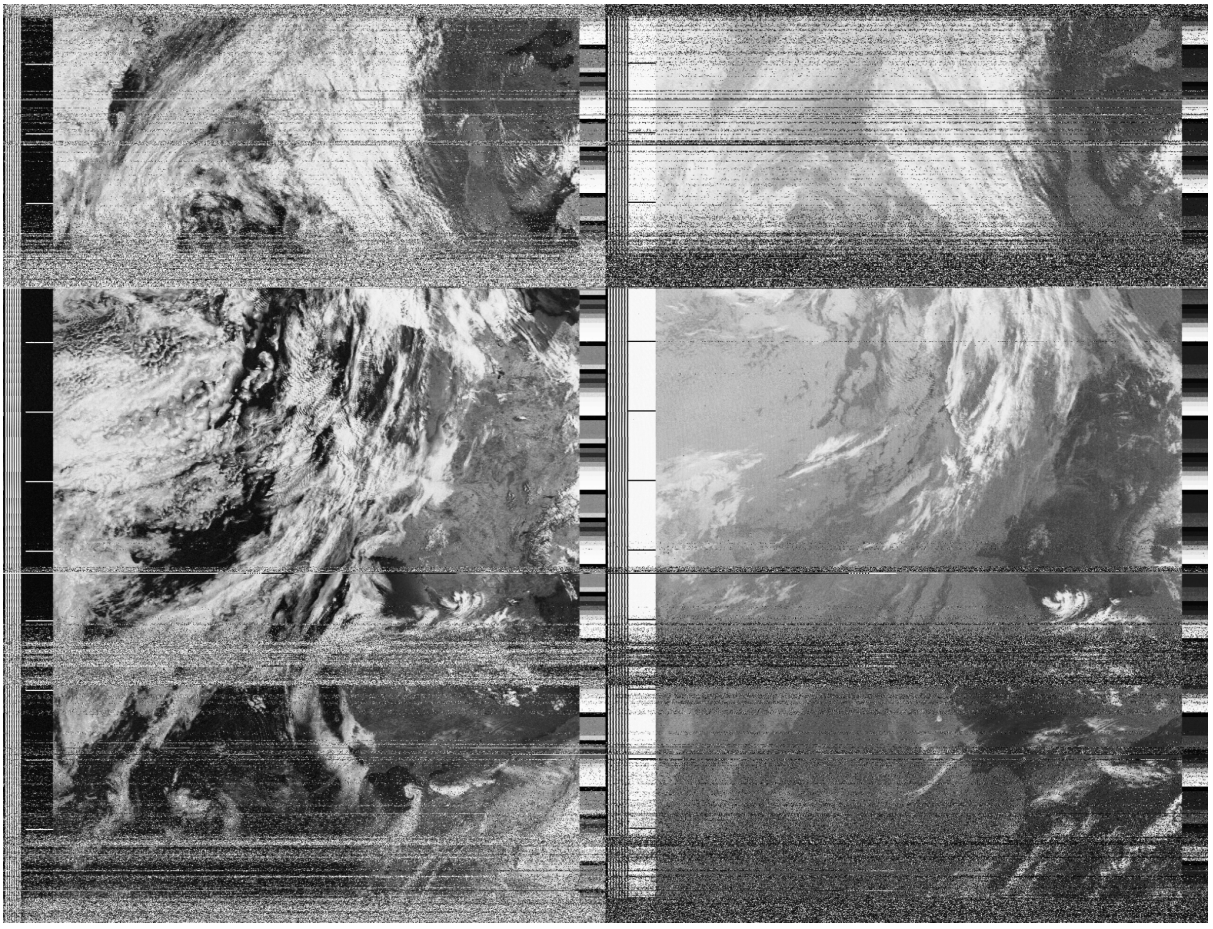
NOAA18



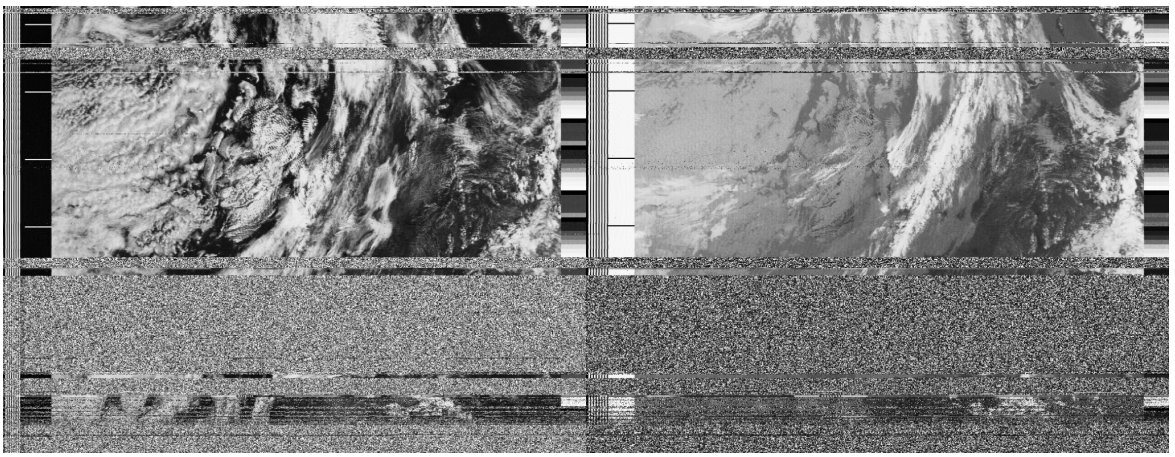
NOAA19



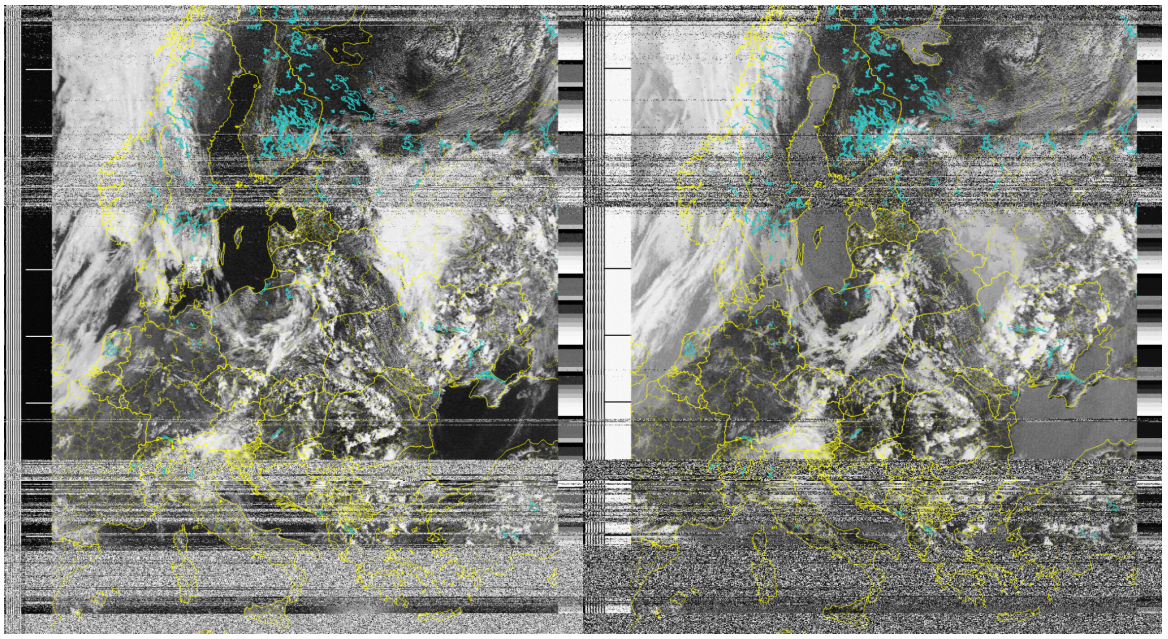
NOAA18



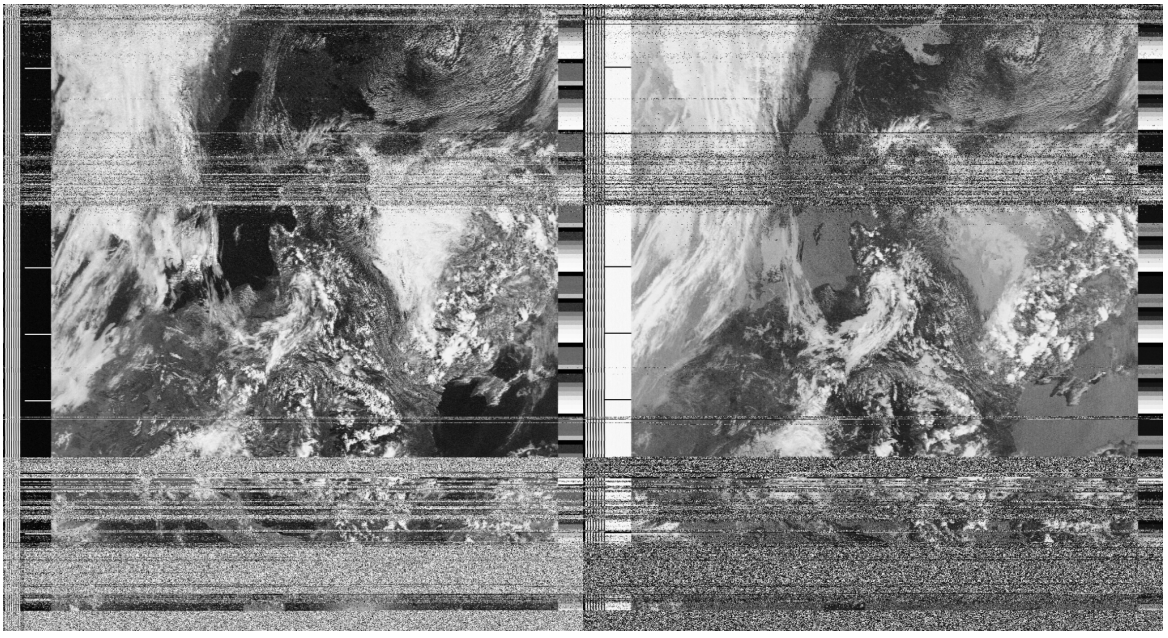
NOAA15



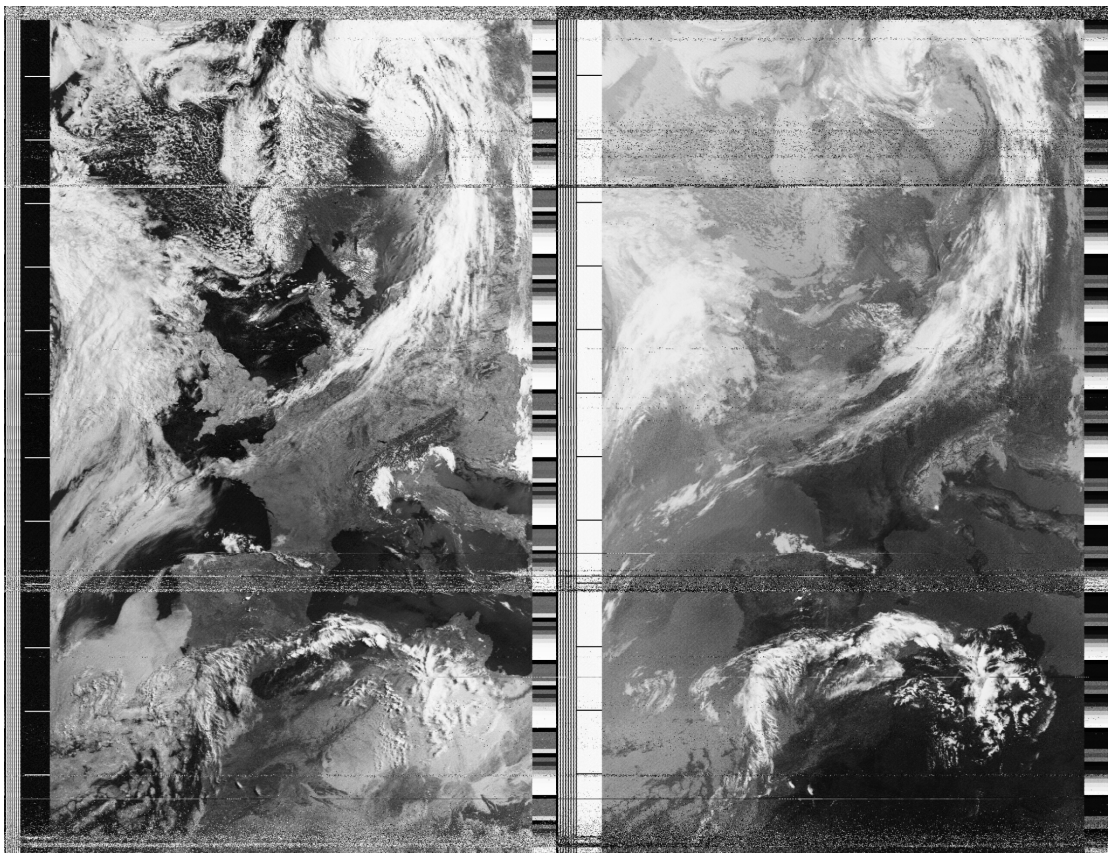
NOAA18



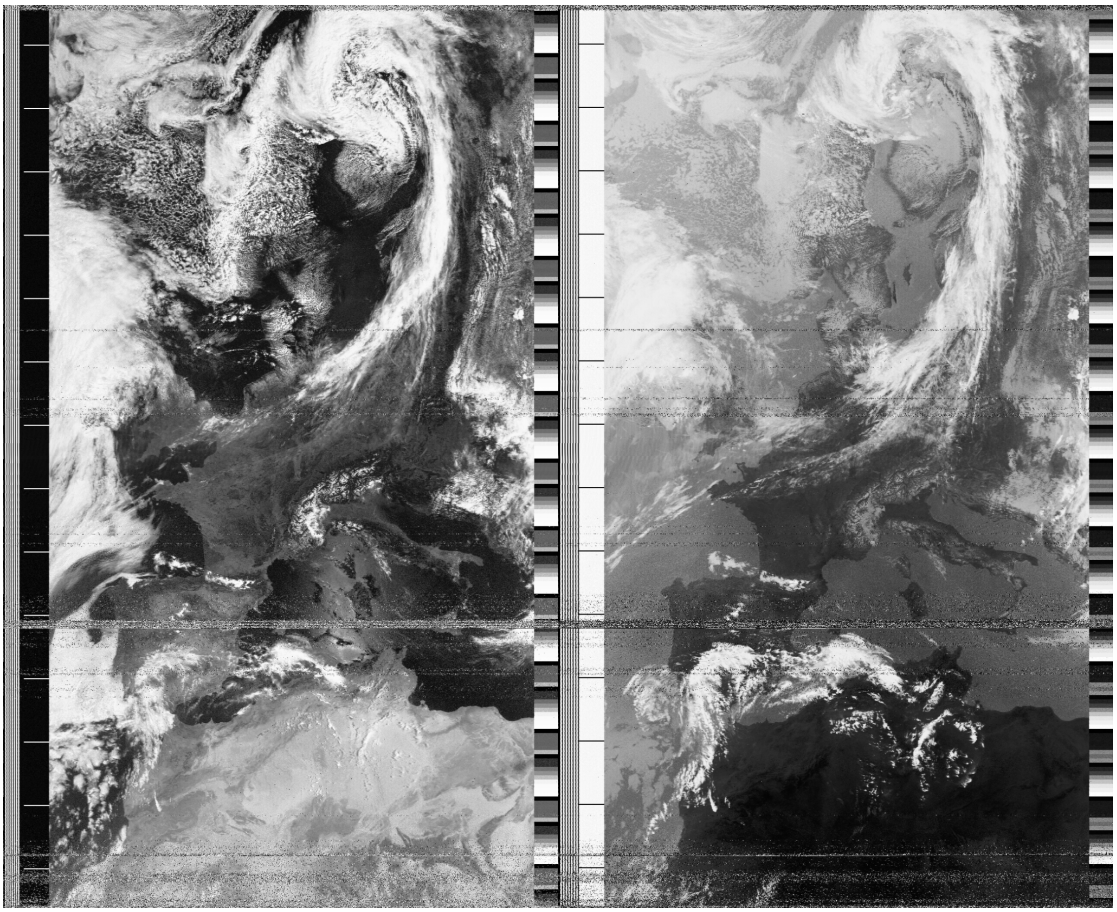
NOAA18



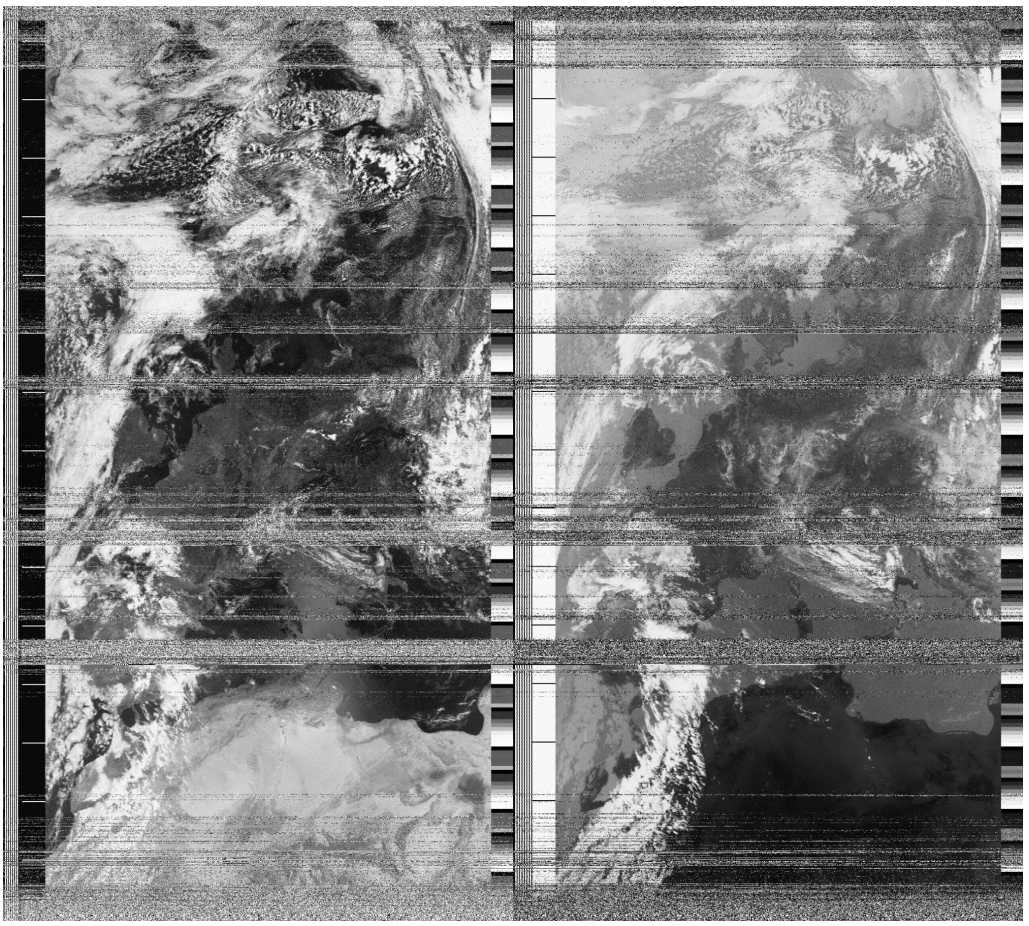
NOAA18



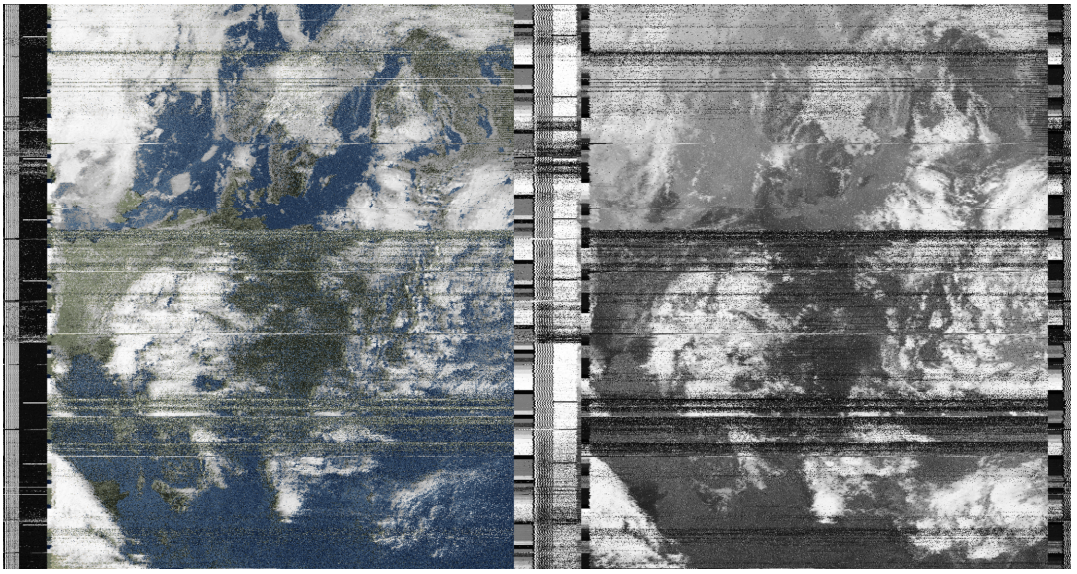
NOAA19



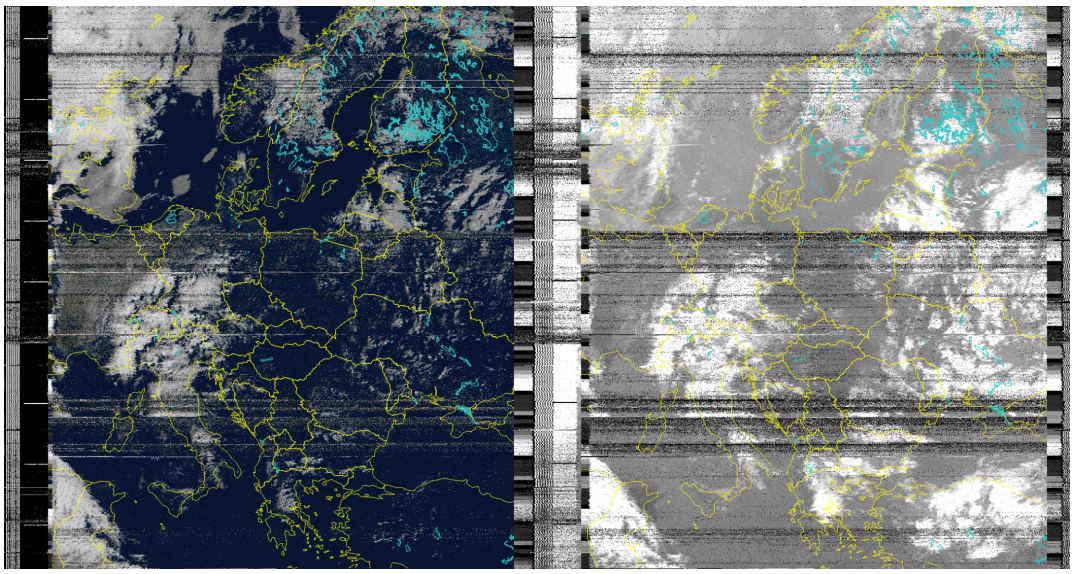
NOAA18



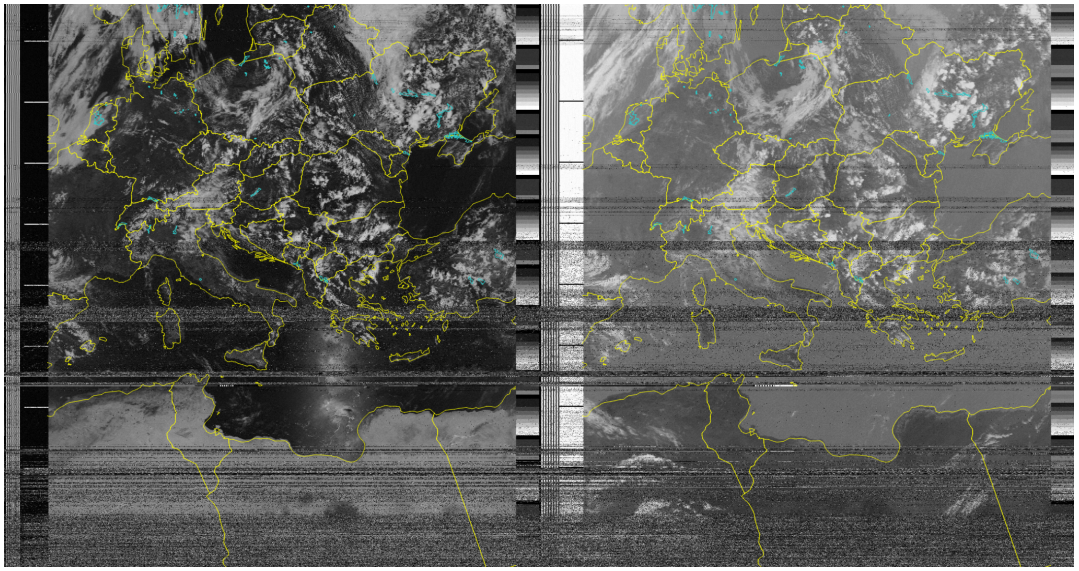
NOAA18



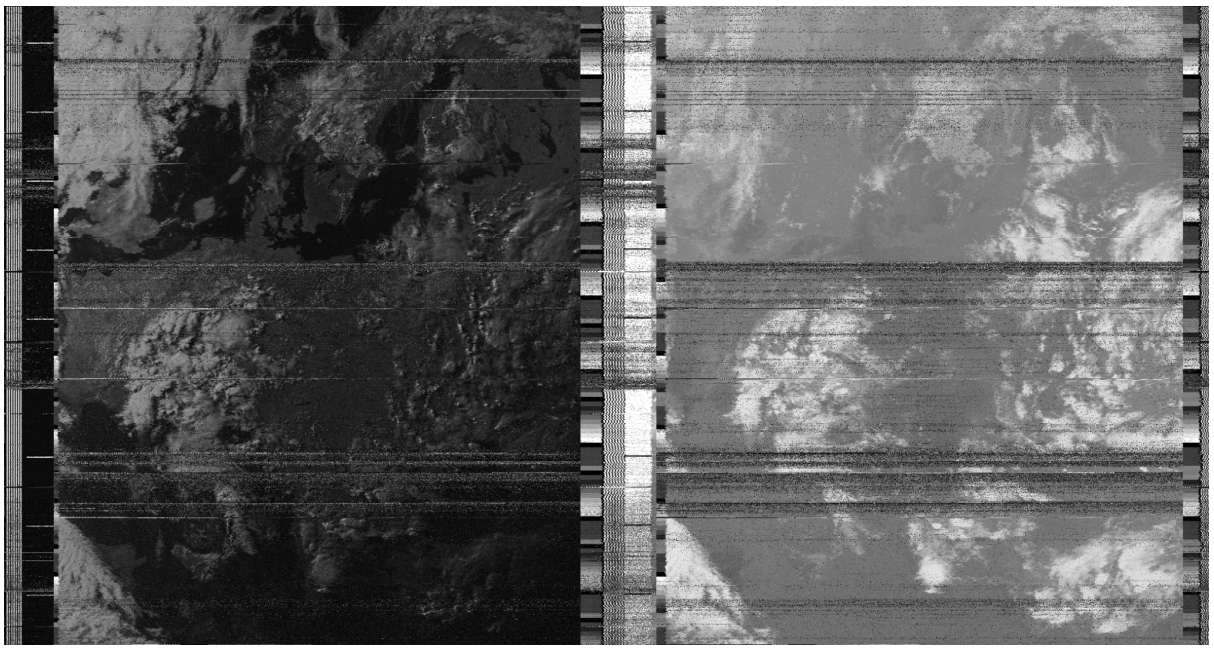
NOAA19



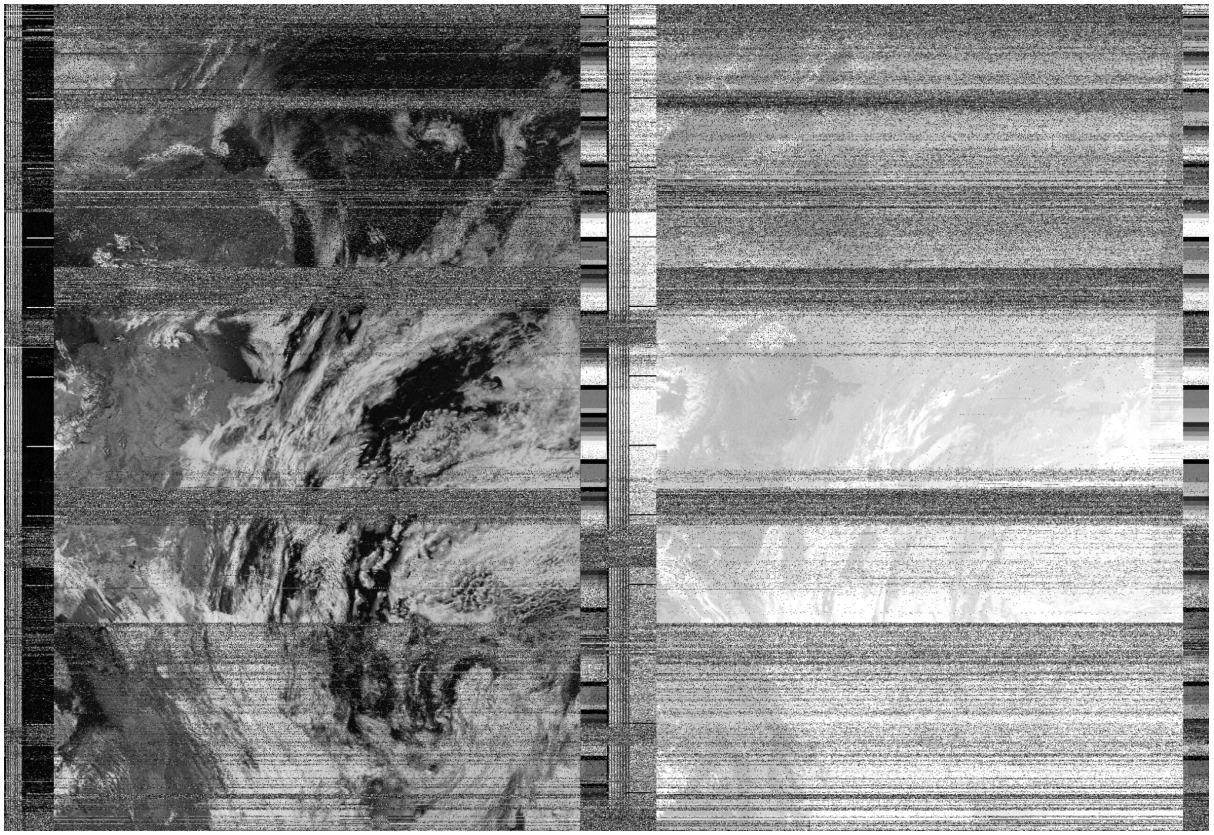
NOAA19



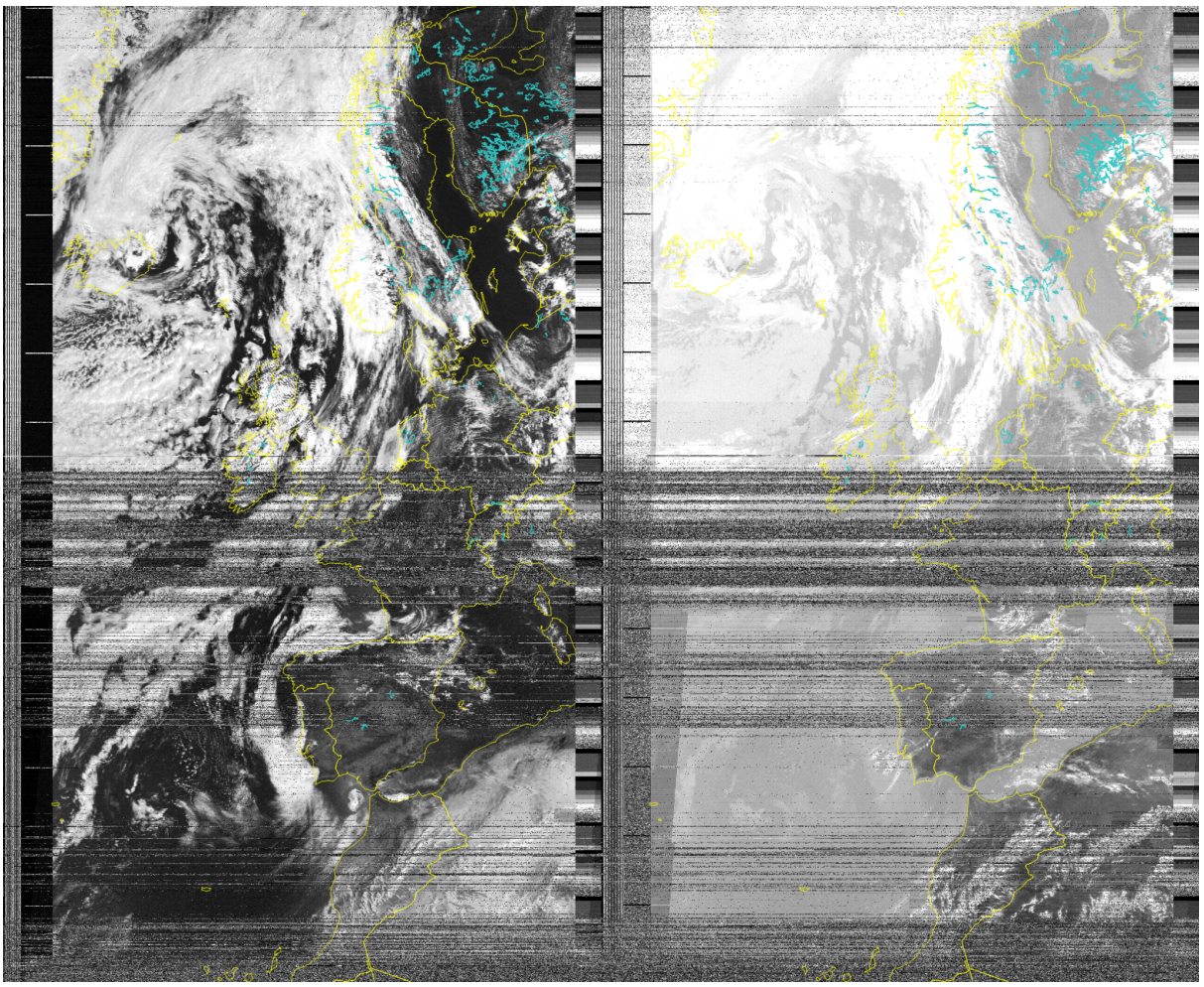
NOAA18



NOAA19



NOAA19



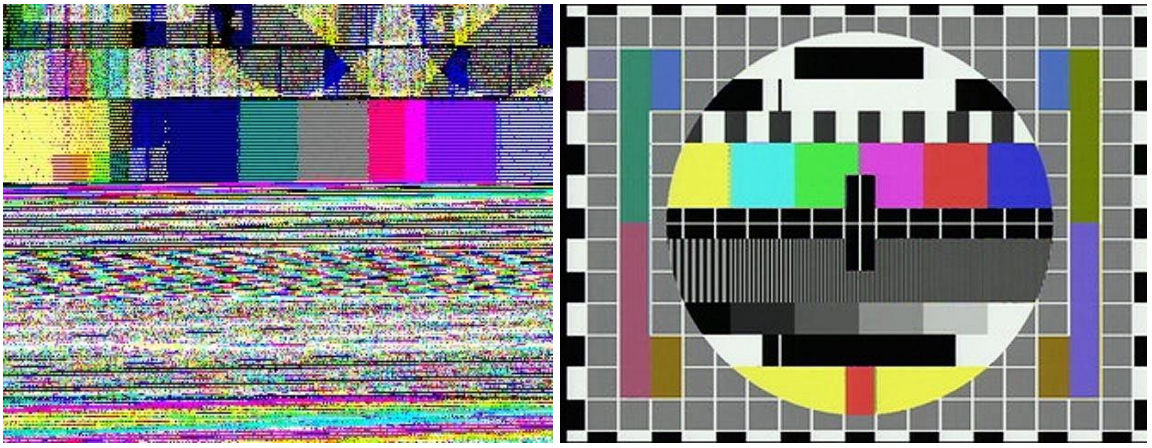
NOAA18

Example of the GPS module output

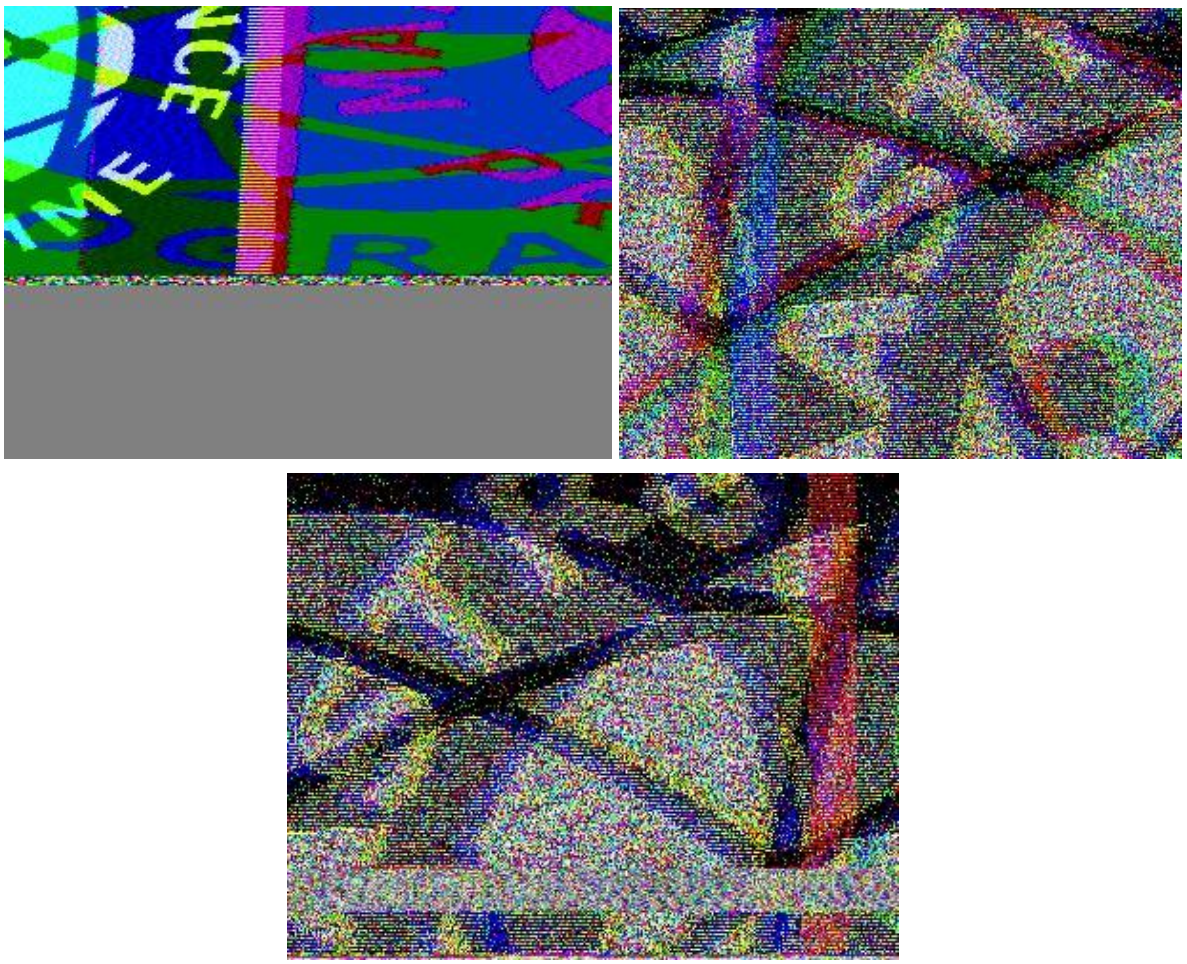
b'$GPGSV,3,1,09,01,50,134,22,03,80,047,,04,49,183,12,06,14,306,*7E\r\n'
b'$GPGSV,3,2,09,09,19,206,21,19,40,295,,21,31,137,23,22,55,080,*78\r\n'
b'$GPGSV,3,3,09,31,23,072,*46\r\n'
b'$GPGLL,,,,,173155.00,V,N*4E\r\n'
b'$GPRMC,173156.00,V,,,,,,,220621,,,N*7F\r\n'
b'$GPVTG,,,,,,,,,N*30\r\n'
b'$GPGGA,173156.00,,,,,0,03,11.00,,,,,,*62\r\n'
b'$GPGSA,A,1,09,04,21,,,,,,,,,,11.04,11.00,1.00*0B\r\n'
b'$GPGSV,3,1,09,01,50,134,20,03,80,047,,04,50,183,14,06,14,306,*72\r\n'
b'$GPGSV,3,2,09,09,19,206,21,19,40,295,,21,31,137,23,22,55,080,*78\r\n'
b'$GPGSV,3,3,09,31,23,072,*46\r\n'
b'$GPTXT,01,01,02,u-blox ag - www.u-blox.com*50\r\n'
b'$GPTXT,01,01,02,HW UBX-G70xx 00070000 FF7FFFFFo*69\r\n'
b'$GPTXT,01,01,02,ROM CORE 1.00 (59842) Jun 27 2012 17:43:52*59\r\n'
b'$GPTXT,01,01,02,PROTVER 14.00*1E\r\n'
b'$GPTXT,01,01,02,ANTSUPERV=AC SD PDoS SR*20\r\n'
b'$GPTXT,01,01,02,ANTSTATUS=OK*3B\r\n'
b'$GPTXT,01,01,02,LLC FFFFFFFF-FFFFFFFF-FFFFFFFF-FFFFFFFF-FFFFFFFD*2C\r\n'
b'$GPRMC,174815.00,A,5050.85125,N,00542.20978,E,0.048,,220621,,,A*7B\r\n'
b'$GPVTG,,T,,M,0.048,N,0.089,K,A*2E\r\n'
b'$GPGGA,174815.00,5050.85125,N,00542.20978,E,1,05,3.54,52.7,M,46.4,M,,*6F\r\n'
b'$GPGSA,A,3,01,09,04,21,17,,,,,,,,5.03,3.54,3.57*0E\r\n'
b'$GPGSV,3,1,12,01,42,138,38,03,74,063,17,04,57,180,32,06,20,307,16*74\r\n'
b'$GPGSV,3,2,12,09,26,207,32,12,04,343,28,17,43,251,32,19,41,285,14*7A\r\n'
b'$GPGSV,3,3,12,21,24,140,31,22,48,084,15,25,00,014,,31,24,064,16*7F\r\n'

SSTV Images

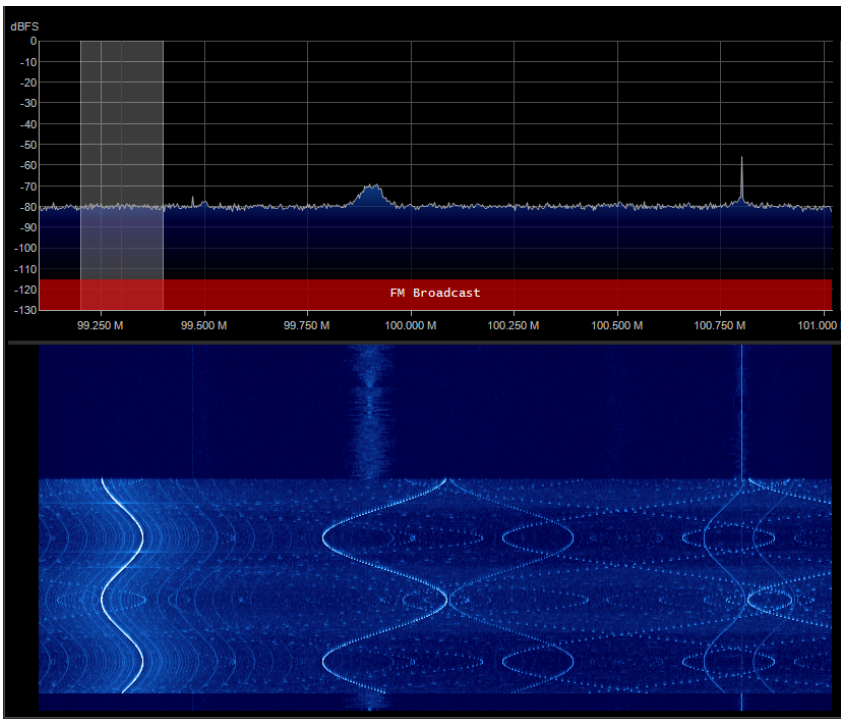
SSTV image decoded from transmission during the 20th anniversary of the ISS' Ham Radio



Left: Decoded Test Image Right: Original Test Image



Various Decoded MSP Test Images with visible but distorted logo



Chirp Signal with visible harmonics