Prototype Satellite Image Receiver

Jake Horder

October 29, 2021

Abstract

A home-made satellite receiver station has been designed and constructed for use with meteorological satellites operating at 137.1 MHz**. An image decoding script was written which takes an audio recording of the satellite signal and produces an image of the east coast of Australia. This receiver station could be improved in many ways, or could be extended to receive other types of radio frequency communications.**

1 Introduction

This paper describes the design and construction of a prototype satellite image receiver station, and demonstrates its capabilities by presenting the results of a recent receiving session. In the following sections, the antenna design and electronic hardware set up is presented, then the data acquisition and processing stages are outlined. The performance of the prototype is discussed in the context of project goals, before further work is suggested for the second iteration of the receiver station.

The enabling technology used in this project is software defined radio (SDR). At a conceptual level, SDR devices allow radio signals to be sampled and digitised so that digital signal processing can be offloaded to a separate downstream computer. These devices are small and cheap, and when combined with a microcomputer such as a Raspberry Pi, they provide a means to create a satellite receiver station for less than $100.

The primary satellite that this prototype aims to receive from is the NOAA 19 weather satellite, operated by the National Oceanic and Atmospheric Administration in the USA. This satellite has a downlink frequency of 137.1 MHz, which sits in the small window of the electromagnetic spectrum that has been reserved for meteorological communications by many governments around the world. The data transmitted by NOAA 19 is modulated, but it is not encrypted, meaning anyone who can receive the signal can decode the images contained within, using the freely available knowledge of the information transmission protocol being employed.

This prototype has been intentionally constrained by a small number of design goals which aim to define the scope of the project. Further to being able to receive and decode images, the prototype has been designed to be lean, portable, automated, and sturdy enough for field deployment. These goals have largely been met, although there are many ways this design could be improved in the second prototype.

2 Receiver hardware

The satellite receiver station, shown in Figure 1, featured several major elements. To maximise the signal reception, a quadrifilar helix (QFH) antenna was constructed from basic components. A 3 m length of RG173 coaxial cable connected the QFH antenna to an RTL-SDR Blog V3 software defined radio device, which in turn was connected to a Raspberry Pi Model 1B+ powered by a 5 V/2 A battery pack through a USB to micro-USB lead. The QFH was taped to the top of a tent pole, which was then secured to the ground with rope and steel tent pegs. The Raspberry Pi was placed inside a plastic soap box, but there was no larger container for the electronics as a whole.

References [1] and [2] were particularly helpful for the design and construction of the QFH antenna. In particular, while Hollander [3] describes this type of antenna as being characteristically complex and difficult to implement, Portune [2] recounts the success of many home-made attempts, and suggests to his audience that the QFH construction does not need to be as precise as many people believe.

Figure 1: Satellite receiver station set up. The tent pole is secured with rope and pegs. The QFH antenna (right inset) is made with aluminium wire and a cardboard postal tube. The hardware (left inset) features the RTL-SDR, a Raspberry Pi encased in a plastic soap box, and a battery pack.

Ruperto [1] presents a measurement guide based on the analytic and experimental work of Kilgus [4]. The only variable in the measurement guide is the wavelength of the carrier. For NOAA 19, with a carrier frequency of 137.1 MHz, the carrier wavelength is $\lambda = 2.19$ m. The QFH design then amounts to the calculation of the loop dimensions, which are shown in Table 1. When cutting out wire segments, the actual length of each piece is twice the 'length' stated in the table.

	Height	**Diameter**	**Length**
Big loop	0.569	0.379	1.225
Small loop	0.521	0.341	1.112

Table 1: QFH element dimensions (all in metres).

A local wire supplier was found that conveniently listed a number of different materials, at varying wire diameters, with the option to cut to a specified length. Copper has a large conductivity, which is desirable for an antenna, however aluminium is much cheaper, with only a small reduction in conductivity. A 10 m length of 5 mm diameter aluminium wire was chosen for the antenna helix. This diameter was selected with the idea that a greater current capacity would improve reception. Whether or not this was the case, the thickness produced a wire stiffness that was low enough to allow shaping by hand, but high enough that no extra support for the helix was required once the shaped was made.

A spare cardboard postal tube was used for the central core of the QFH antenna. The tube has a diameter of 60 mm and a length of 660 mm, and is sufficiently stiff for the purpose. Holes were made in the tube using a drill, and the lengths of aluminium wire were threaded through at the bottom and bent up by hand so that the wire ends met at the feed point at the top of the tube. At the feed point, a screw plate was used to secure the opposing ends of each loop of wire. This type of connection provided a solder-free way to adequately connect the core and sheath of the coaxial cable to the aluminium wire loops. The coaxial cable draped down the side of the postal tube, and no balun was implemented. Finally, the completed QFH antenna was affixed to a spare tent pole using many wraps of masking tape.

No collective packaging solution was made for the electronics, as the length of the coaxial cable was enough for all components to be placed on the ground. Additionally, the RTL-SDR and the battery pack each have relatively sturdy housings, and the Raspberry Pi was effectively protected from dirt with the use of the plastic soap container. Satellite recordings were only performed in good weather, so no water-proofing was necessary. Although a Raspberry Pi Model 3+ was available, this model requires a 2.5 A power source, and the only battery pack freely available for this project was limited to 2 A. Arguably these limits could be pushed, but the Raspberry Pi Model 1B+ ended up being sufficient for the major goals of this prototype anyway. The biggest downside is the lack of WiFi connection in the earlier model, which makes data transfer between the Raspberry Pi and the laptop slightly less convenient.

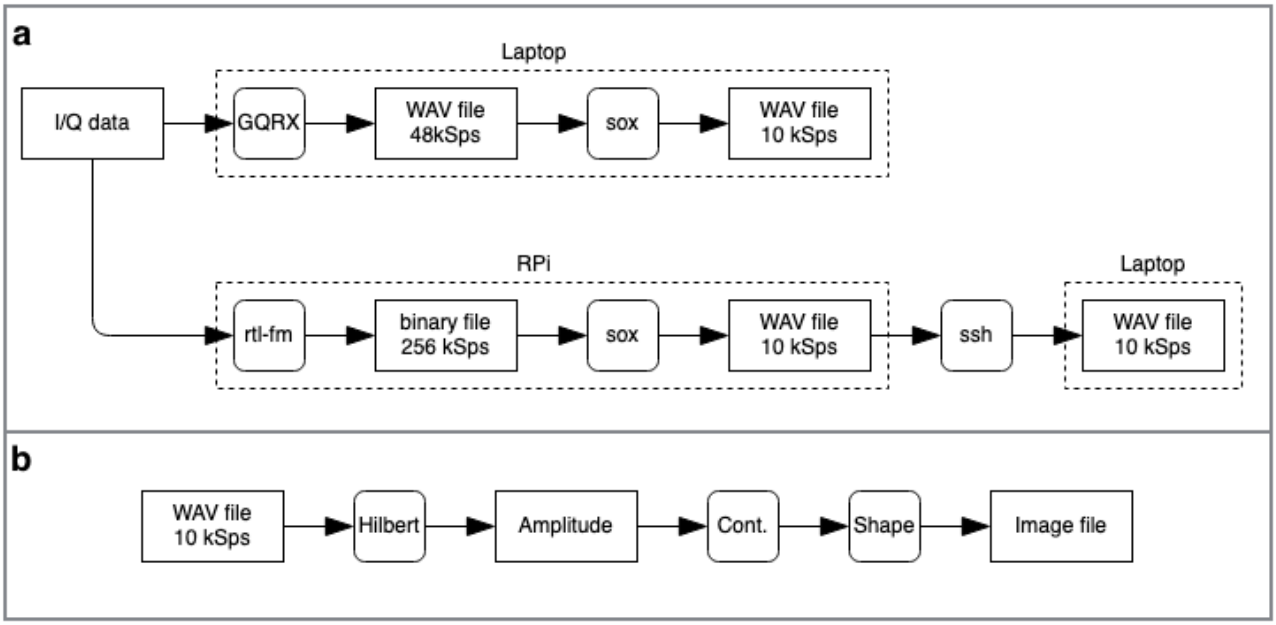

Figure 2: Data flow from the RTL-SDR to the final image. (a) Data acquisition, using just a laptop (top series), or in conjunction with a Raspberry Pi (bottom series). In either case the output is a WAV file. (b) Data processing, performed in a Python script on a laptop.

3 Data acquisition

The primary goal of the antenna and electronic hardware described above is to receive the raw radio frequency (RF) signal from the electromagnetic field, then demodulate this information to produce an audio file, as shown in Figure 2a. This audio file is then transferred to a laptop for processing, as described in the next section. The receiving process leans heavily on the RTL-SDR device, while the demodulation is performed on the Raspberry Pi.

The data flow between satellite and receiver proceeds as follows. Image data from a camera aboard the satellite is serialised into a stream of 8-bit values, such that two lines of the image are processed per second. This image data is amplitude-modulated into a 2400 Hz carrier signal, and the resulting signal is then frequency-modulated into a 137.1 MHz carrier signal and transmitted from the satellite in the form of right hand circularly polarised light. This analog protocol is known as Automatic Picture Transmission (APT) and will likely be phased out in favour of the digital Low Rate Picture Transmission (LRPT) protocol employed by satellites like Russia's Meteor M2.

At the receiving antenna, a voltage is induced by the incident signal and feeds into the RTL-SDR device. This device uses an R820T2 tuner chip to set a local oscillator (LO) to be mixed with the RF carrier, and an RTL2832U ADC to sample the resulting baseband signal. The LO mixing occurs in quadrature, with the RF signal split across two channels, one of which is mixed with a referent 'in-phase' signal, and the other mixed with a signal shifted by 90°, producing a 'quadrature-phase' or simply 'quadrature' signal. The in-phase and quadrature signals are sampled simultaneously, producing a stream of pairs of numbers referred to as I/Q data.

The data type output by the RTL-SDR depends on the software used to interface with the device. The simplest method is to use the command line application `rtl-sdr`, which outputs raw I/Q data as 8-bit unsigned integer pairs. The command line application `rtl-fm` is similar, but allows for frequency demodulation, and outputs binary data in signed 16-bit integer pairs. The use of a graphical interface can be helpful when debugging on a laptop. `GQRX` is one such application, which by default outputs audio files, but can be set to output raw I/Q data as 32-bit floats.

If the RTL-SDR is being used directly with the laptop, `GQRX` is recommended for obtaining audio recordings of the frequency-demodulated signal. If the receiver is deployed in the field and set up to record automatically using the Raspberry Pi,

then additional `bash` scripts are required. For this prototype, the `rc.local` file in the `etc` folder was appended to included a `bash` command that calls upon the script `rec.sh` located in the home directory. The `rc.local` file was used because it is run automatically as part of the boot up process of the Raspberry Pi.

The `rec.sh` script has two main sections. First, output file names for the audio recording are created by appending an integer to a generic string. The integer is read from a text file, and once the file names are defined, the integer is incremented by 1 and is overwritten to the same text file. The purpose of this approach is to prevent audio files being overwritten every time the Raspberry Pi is turned on, which would be the case if an audio file output name were hard coded in this script.

The second main part of the `rec.sh` script employs the `rtl-fm` command line application to sample and demodulate the I/Q data, and then the `sox` command line application to reformat the output of `rtl-fm` into a WAV file at a sample rate of 10 kHz. This sample rate is chosen because the audio file has most of its important content centred about the AM carrier frequency of 2400 Hz, and 10 kHz provides enough bandwidth to include the sidebands while also leaving the WAV file size as small as possible. These scripts are available to view and download from the `github` link shown in the Appendix.

A typical recording session would involve first consulting the website n2yo.com for upcoming passes of NOAA 19 overhead. This website lists predicted passes up to about a week in advance, with rise time, set time, and elevation included. The elevation at the mid point is the biggest factor determining whether the pass will be worth attempting to record, since a low elevation corresponds to a further distance between receiving station and satellite, and hence poorer signal strength. Experience showed that an elevation greater than 60° from the horizon yielded decent recordings, with greater than 85° being optimal but rare. On the evening of a scheduled pass, the receiving station would be set up, and when the satellite was expected to first appear above the horizon, the battery pack was switched on by hand, which boots up the Raspberry Pi and automatically initiates the recording. Once the satellite sets, after about 15 minutes, the battery pack is switched off and the Raspberry Pi is taken inside to be connected to an ethernet cable for internet access. Finally, the audio recording is transferred to a laptop using a secure shell terminal.

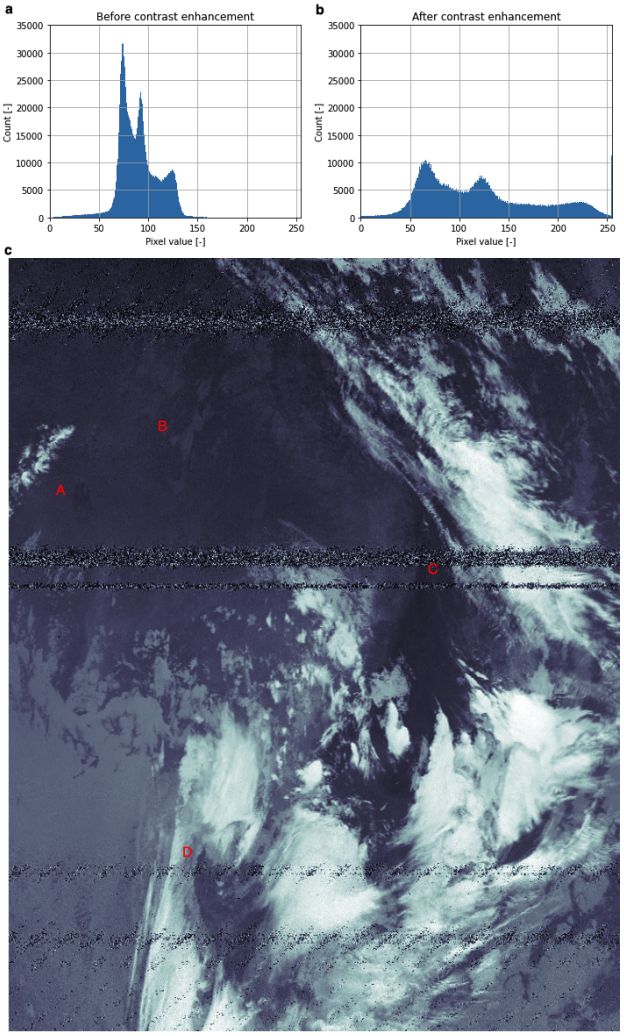

Figure 3: Contrast enhancement of the infrared image channel. (a) Histogram of IR channel data after Hilbert transform, showing an information cluster between 50 and 150. (b) Histogram after data values have been redistributed over the 8-bit pixel value range. (c) The IR channel following contrast enhancement, with notable features labelled: [A] Lake Eyre, [B] Lake Yamma Yamma, [C] Byron Bay, [D] North East Tasmania.

4 Data processing

Once a successful recording has been performed, using either a laptop directly or using a Raspberry Pi, the key data object that is handled at the processing stage is an audio file in WAV format, with duration 10 minutes. Rather than the full 15 min-

utes, this shorter duration is selected because the beginning and end of the satellite pass has poor reception due to the proximity of the satellite to the horizon.

The goal of the data processing stage is to decode the information contained in the audio file and produce an image file representing the viewpoint of the orbiting satellite. A single Python script was written to perform all the necessary steps to achieve this goal. A block diagram of the processing is shown in Figure 2b. Although the telemetry tags in the data offer a degree of structure, which could be exploited to automate the image processing steps, it was found that variable reception between recordings introduced too much difficulty, and so the script requires manual intervention at several places.

Once the WAV file is loaded, it is first trimmed in order to reject particularly noisy sections that usually feature at the beginning and end of the recording. Next, the amplitude envelope of the audio file is extracted using the Hilbert transform approach. Finally, the 1D amplitude timeseries is reshaped into a 2D array representing the twin-channel image. The width of the image is determined using the fact that the APT protocol transmits a single line every 0.5 s, together with the known frequency of the audio file obtained by the WAV import function. At this stage the single 2D image object contains the optical channel, the infra red channel, and two sets of telemetry tags.

In order to produce a cleaner final image, the twin-channel image is sliced to produce two new variables, one 2D array containing only the optical channel, and another 2D array containing only the infra red channel. For a given WAV file, this slicing process requires trial and error to select the correct subset of the twin-channel image.

The contrast of each channel is enhanced using visual inspection of the histogram of pixel values in the 2D array. The region of the histogram containing most of the relevant information is identified by eye, then the data are scaled to spread this information over a broader section of the 8-bit range. For example, a channel with pixel values in the range 0 to 255 might have most of the key image information clustered in a hump between, say, 100 to 150, as determined by visual inspection of the histogram. The contrast is enhanced by setting the lower bound of the data to 100 and the upper bound of the data to 150, then mapping this 100-150 range into the new 0-255 range. Figure 3 shows the contrast enhancement process, and the resulting image on the IR channel.

Following contrast enhancement, the data from the two channels are plotted and saved as PNG images. This completes the data processing procedure. Although the script requires manual intervention at several steps, the method is consistently applicable for any NOAA 19 recording.

5 Discussion

The design goals at the outset of this project stipulated that the satellite image receiver station should be lean, portable, automated, and sufficiently sturdy to allow field deployment. The first two goals have certainly been met, and the automation goals has also largely been successful. Elements of the receiver station are not satisfactorily damage proof, meaning the prototype has fallen short of meeting the sturdiness design goal. However, the receiver station does function well when handled with care.

The prototype is lean in the sense that only the bare minimum materials have been used, and there is no element of the system that could be omitted without severely reducing the functionality of the receiver station. It might be said that a dipole antenna is actually the leanest option, though the QFH proved to be far less complicated to construct and handle than first anticipated. Further, the omnidirectional reception profile of the QFH design leads to a significant increase in image height when compared to the dipole antenna.

Portability was selected as a design goal with the forethought that optimal reception would be achieved by deploying the system in open fields or similar environments away from built up dwellings and infrastructure. Actually, all satellite recordings performed when testing this prototype occurred at a residential property in a suburban environment, but still yielded decent results. The final support structure for the antenna was still designed with field deployment in mind, and this too played back into the requirement for the system to be lean, so aiming for portability was still influential on the final form of the prototype.

The crux of the project was the automation. Ultimately, an undiagnosed noise issue prevented a demonstration of the QFH successfully working

with the RTL-SDR and Raspberry Pi to record a NOAA 19 pass. This is despite a lengthy attempt to troubleshoot, making use of `GQRX` for sanity checks, and working with commercial FM radio to run test recordings. Nevertheless, the core hardware and software components necessary for the system were identified, and in theory the design presented above is capable of automatically capturing information from satellites.

Significant time was spent investigating the appropriate software applications to achieve lean automation. In part this extended to the choice of operating system to use on the Raspberry Pi. The Model 1B+ is around 10 years old, and since its purchase there have been many SDR software packages and programs developed, mostly with GUI features that demand quite a bit from the CPU, or can only run on recent versions of Linux distributions. Often a bulky OS combined with memory intensive software made using the Raspberry Pi 1B+ extremely slow. The Model 3 could have been used, but would have required buying or otherwise obtaining a more powerful battery pack, as explained elsewhere in the report.

The turning point was the discovery of command line applications like `rtl-sdr` and `rtl-fm`, which have been developed specifically for bare bones control of SDR devices. Using these applications eliminated the need for any kind of desktop capability on the Raspberry Pi, meaning a basic 'server' style OS could be used. Additionally, the data type at each step became more transparent, which then allowed the image decoding script to be optimised.

Sturdiness of the prototype receiver station has remained the lowest priority of all the design goals. The fact that the QFH antenna is sticky taped to the tent pole, and the electronic hardware is exposed resting on the ground would seem to limit the situations in which the system can be deployed. However, for the ultimate goal of obtaining satellite imagery, this aspect is not project-critical. For example, the amateur radio community strongly recommends against handling antennas in stormy or even rainy weather, because of the risk of lightning strikes. So no recordings were attempted in these conditions anyway. On the other hand, particularly windy conditions did cause the cancellation of a number of satellite recordings. There was a risk that the QFH antenna would be blown off the tent pole, and either the aluminium wire, the postal tube core, or the coaxial cable and feed point connection could be damaged as a result of a fall.

6 Further work

There are many immediate steps that could be taken to improve the prototype described above. Alternatively, the core aspects of this project could be used as a springboard into the development of prototype receivers for a whole range of radio communication devices. This section aims to guide researchers who wish to pursue iterations of the prototype satellite image receiver.

Firstly, an upgrade to the materials used for the antenna would provide increased durability, which would enable frequent field deployment. The central core of the QFH antenna could be replaced with PVC pipe of at least 5 cm diameter. It would be simple to drill holes in the PVC for the aluminium wire, and the increased stiffness and weatherability is at a cost of only a small increase in antenna mass.

The pole and guide rope set up is an effective method for achieving an adequate height for the antenna above ground level. If an area allows for decent reception closer to the ground, a tripod system could offer a time saving when setting up and taking down the receiver. In either case, the next generation structure should provide a means for safely securing the electronics, perhaps in a small weather-proof box at the foot or mid-height of the structure. The box would need to allow the coaxial cable to exit cleanly, while switch access for the battery pack could be on the outside or inside.

Although the RTL-SDR does contain a small LNA, image quality may be significantly enhanced by employing a separate LNA in between the antenna and the RTL-SDR. Vendors like GPIO Labs offer a range of LNAs at prices similar to the cost of the RTL-SDR. These amplifiers often have a particular frequency that they amplifier the most, with shoulder frequencies being amplified much less.

A review of the microcomputer capabilities provides an opportunity to improve the workability of the system. The Raspberry Pi Model 3 and Model 4 each have wireless internet connection, and although these models require 25% more current than the Model 1B+, WiFi would allow for

increased automation. For example, a `bash` script could be written to scrape the predicted times for upcoming NOAA 19 passes from the n2yo.com website. The recording script described in Section 3 could then be automatically initiated at the appropriate time. Next, the image decoding script could be run to produce an image file, which could then be automatically uploaded to a personal homepage or Twitter account.

The final antenna design for this prototype worked quite well, but it would be interesting to see just what kinds of materials are sufficient to receive data from satellites. A selection of metals with varying conductivity could be obtained and used to construct several identical QFH antennas. A qualitative comparison of the image quality might yield important information about the cost-benefit tradeoff, which would be useful for the amateur radio community. For example, purchasing aluminium wire along with a specialty LNA could be cheaper but just as effective as purchasing only copper wire for the QFH antenna.

Finally, a second prototype receiver could be produced to target other orbiting satellites, or with slight modifications the receiver could be optimised for radio astronomy applications. A number of geostationary satellites positioned over Australia are constantly available, unlike the intermittent passes of NOAA 19. The geostationary satellites, like South Korea's GEO-KOMPSAT-2B, are roughly forty times further from the surface of the Earth, meaning they have an almost 'full disk' viewing footprint. The greater altitude also means they operate with higher downlink frequencies, typically around 1690 MHz, and require large parabolic or grid dish antennas [5]. Separately, an RTL-SDR device can be used in conjunction with a home-made horn antenna to investigate the distribution of neutral hydrogen in the Milky Way, or to study the rotational dynamics of the galaxy [6].

7 Conclusion

The project outline presented above demonstrates that only a small catalogue of materials are required to receive and decode images from weather satellites. Some materials, such as the RTL-SDR and the Raspberry Pi, have minimal flexibility, while other aspects, like the type of metal used in the antenna, have a wider range in which a designer can explore.

The QFH antenna is a surprisingly simple antenna design to implement, and is recommended pursuing for anyone have second thoughts about its complexity. Similarly, many software packages available on the web for image decoding obscure how simple this process is. The Python script presented here gives the minimal code required to transform an audio recording into an image, with the core functions only amounting to a few key lines of code.

As an entry level project for the field of software defined radio, this prototype design allows for both high level and low level interaction with the RTL-SDR device. The I/Q sampling process is widely applicable in many other areas of radio communication or even general signal reception and analysis. Numerous pathways are present for continuing with this satellite image receiver project.

8 Appendix

Data and scripts used to make images in this report can be accessed at https://github.com/jakehorder/prototype_satellite_image_receiver.

References

[1] E. F. Ruperto, *The W3KH Quadrifilar Helix Antenna*, QST, Aug. 1996, p. 30-34

[2] J. E. Portune, *The Quadrifilar Helix as a 2 Meter Base Station Antenna*, ARRL, 2009

[3] R. W. Hollander, *Resonant quadrifilar helix antenna*, Working Group Satellites, TechNote, 1999, p. 1-17.

[4] C. C. Kilgus, *Resonant quadrifilar helix*, IEEE Trans. AP-17, 1969, p. 349-351.

[5] VKSDR, *Receiving Images from Geostationary Weather Satellite GEO-KOMPSAT-2A*, https://vksdr.com/xrit-rx accessed 27/10/21

[6] West Virginia University , *Digital Signal Processing in Radio Astronomy*, https://wvurail.org/dspira-lessons/categories/astronomy/ accessed 27/10/21