SDR Satellite Communications with GOES

Background

The GOES satellites are a series of American weather satellites used by the National Oceanographic and Atmospheric Administration (NOAA) to make weather observations from space. Observation data from GOES satellites is crucial to the National Weather Service's (NWS) ability to forecast. There are always two GOES satellites in active use, one above the east coast, and one above the west coast. Older GOES satellites are moved into parking orbits, where they are kept in reserve in case the active GOES satellites malfunction. The satellites are in geostationary orbits, meaning relative to us ground observers, the satellite does not appear to move.

***Figure 1**: GOES-R (16) Mission Patch*

On board the satellite exists a suite of hardware designed to observe the Earth, the space environment, and the Sun. The Advanced Baseline Imager (ABI) is of interest, which makes observations of the Earth's radiance along 16 spectral bands with resolutions up to 0.5 km. The instrument is flexible, and it is able to image the full disk of the Earth while also producing more frequent and higher resolution of two "Mesoscale" regions selected by the NOAA. These mesoscale regions can change from day to day in order to track hurricanes, snowstorms, or other significant events, but by default they focus on regions of continued interest such as New England.

The NWS and NOAA also use the GOES satellites as a key part of an information dissemination service known as the Emergency Managers Weather Information Network (EMWIN). The NOAA and NWS upload maps and text documents with forecasts and predictions to GOES satellites in order to be shared with amateur receivers across the nation. In recent GOES satellites (GOES-16 or later), the EMWIN data transmissions have been combined with the High Rate Information Transmission (HRIT) service, which automatically sends lower-resolution observational data from GOES satellites to amateur receivers. This HRIT/EMWIN transmission will be the focus of this project.

The Signal

The HRIT/EMWIN signal has properties listed in table 1.

Table 1: HRIT/EMWIN Signal Specifications	
Center Frequency	1694.1 MHz
Bandwidth	1.2 MHz
Polarization	Linear
Modulation	QPSK
Data Rate	400 Kbps
Symbol Rate	927000 Symbols/second

Materials

1.7 GHz Dish Antenna:

Note: The manufacturer I bought mine from no longer is no longer selling them. There aren't many other affordable options for 1.7 GHz dishes. A forum user had success with a modified 2.4 GHz (Wi-Fi) dish.

You will also need a way to mount the antenna firmly

2.4 GHz Dish Forum Post: https://www.sdrplay.com/community/viewtopic.php?f=5&t=3262

Wi-Fi Dish: https://www.provantage.com/premiertek-ant-grid-24dbi~7PREK014.htm

This antenna will require an N-type to SMA connector

Nooelec SAWbird+ GOES filter:

Note: I got the one without a case, but now they sell them with a case too. The case is not necessary, heat has never been an issue with the filter.

With Case: https://www.nooelec.com/store/sdr/sdr-addons/sawbird-plus-goes-302.html

Without Case: https://www.nooelec.com/store/sawbird-plus-goes.html

A USB to micro USB cable will be needed to power the SAWbird+ filter

SDR Device of choice:

Note: I used the Airspy Mini for this project, but the RTL-SDR should work just as well

Airspy: https://airspy.com/airspy-mini/

RTL-SDR: https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/

Raspberry Pi and Peripherals:

Note: Does NOT work with the Raspberry Pi 4, but is tried and true with Raspberry Pi 3B+

Raspberry Pi 3B+: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

A 16 GB or larger microSD card for storage

Approved power supply and outdoors extension cable

A keyboard, mouse, HDMI cable, and display will be needed temporarily for Raspberry Pi setup

Miscellaneous:

A waterproof box was used to house the electronics near the antenna. Keeping the receiver as close to the antenna as possible did a lot to prevent noise from being introduced.

SMA connectors and short, shielded RF wires are necessary to connect the whole setup.

All software necessary for this project is free!

Receiving Hardware

To properly receive the signal, the dish antenna you use must be aimed directly at the target satellite. Even a few degrees of error can make a huge difference in signal quality. I recommend mounting the dish securely somewhere high with nothing obstructing its line of sight. The direction in which you should aim the dish can be calculated at the following website:

https://www.dishpointer.com/

I recommend mounting the dish vaguely in the calculated direction, then wiring up the whole receiving assembly into your SDR device and plugging it in to a laptop with SDR# installed. Then you can find the signal at 1694.1 MHz and fine-tune the dish's direction to maximize signal strength. The dish may move slightly over time, so having access to the antenna so that you are able to adjust it will prove to be very useful.

The dynamics of the seasons may sometimes cause the sun to appear behind the satellite, which can disrupt reception temporarily.

***Figure 2**: Suggested Receiver Station Setup with a 1.7 GHz grid dish antenna mounted on a PVC pipe. Electronics are housed in a strapped-down waterproof box with conduits for the RF cable and extension cord.*

Now that the antenna is set up, the next step is to prepare the RF line in to the SDR. If you want the setup to remain long-term, I recommend utilizing a waterproof box with conduits for wires to come in and out of. The filter should be implemented as immediately after the dish as possible. If you can, the filter should output right into the SDR device with minimal intermediate connecting. It's important to note that the filter has a labelled input and output and it cannot be safely used in reverse. The bias-tee is powered by micro USB, which will eventually be connected to the Pi. Make sure the device is on with the plastic jumper securely over the extended pins. The SDR device will also be connected to the Pi, but first, some Pi setup is required.

Figure 3: *A picture of the receiving station that shows the electronics components. Starting from the antenna, the line in uses an N-type to SMA adapter, a 6in shielded SMA cable, a Nooelec SAWbird+ GOES filter (powered by the Raspberry Pi via microUSB wire), another 6in shielded SMA cable, Airspy Mini SDR device, and finally, a Raspberry Pi 3B+.*

Raspberry Pi Setup

Use Etcher on a PC to load Raspbian Jesse onto the microSD card. After booting up for the first time, change the password from the default and configure the Pi with raspi-config. Be sure to enable SSH so the Pi can be accessed remotely.

```
sudo raspi-config
```

To connect the Pi to the campus internet, you will need the MAC address. Use ifconfig.

```
ifconfig
```

Make note of the address labelled “ether” under “wlan0.” This is your MAC address

Eduroam’s Wi-Fi is not usable by Raspberry Pi, since it is a WPA enterprise network. Share your MAC address with ITS (helpdesk@swarthmore.edu) with a quick explanation of the device and your intents, and ask for the password for the SwatDevice network. They will whitelist your device and give you credentials.

In order to connect remotely to the Raspberry Pi, you will need to know its IP address. Since it is not good networking practice to assign a static IP, your Pi’s address may change occasionally. This is not a problem. A quick script will email you the Pi’s IP address whenever it reboots.

In the home directory, create a file named ipmailer.sh

```
sudo nano ipmailer.sh
```

Fill it with the following script:

```
#!/bin/bash
hostname -I | mail -s "RPi 3B+ IP"YOUREMAIL@WEBSITE.COM
```

Make this script executable with

```
sudo chmod +x ipmailer.sh
```

To run the script automatically after boot, run

```
crontab -e
```

Add the following to the bottom of the file:

```
@reboot sleep 60 && sh /home/pi/ipmailer.sh
```

Exit and save, then reboot and check to make sure the IP is delivered to your email within a few minutes.

To log on remotely to the raspberry pi, use PuTTY. Select SSH, use port 22, and type in the IP address which the Pi just emailed to you. Press open, trust the certificate, and log on to your Pi. This will be very important for accessing your Pi while it is receiving GOES data.

Setting Up goestools

Pieter Noordhuis' software "goestools" is perfect for this project. His guide is below, and I will be paraphrasing it here and adding some personal modifications and insights.

https://pietern.github.io/goestools/index.html

First, you will want to install the following dependencies:

```
sudo apt-get install -y \ build-essential \ cmake \ git-core \ libopencv-dev \ zl
ib1g-dev \ librtlsdr-dev \ libairspy-dev \ tmux
```

Next, build and install goestools as per Pieter's guide.

```
git clone --recursive https://github.com/pietern/goestools
cd goestools
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr/local
make
make install
```

goestools uses commands which rely on .conf files to handle the GOES satellite's signal. I made a conf directory to store mine in. My confs are modified from the default ones provided with the installation. They are to be used with an Airspy receiver but can be easily changed for an RTL-SDR.

While in the goestools directory, do the following:

```
mkdir confs

sudo nano goesrecv.conf
```

Fill this file with the contents of **Appendix A**, save and exit.

```
sudo nano goesproc.conf
```

Fill this file with the contents of **Appendix B**, save and exit. Your goestools installation is done.

Receiving and Processing GOES data

To adequately monitor the GOES data stream and its processing, we will make use of tmux, the terminal multiplexer. This software allows one terminal window to contain multiple within itself. Tmux uses a lot of key shortcuts which are based on the combination "ctrl + b" followed by another, separate key, which depends on what action you are trying to accomplish. A handy reference can be found here:

https://tmuxcheatsheet.com/

To start, open a new tmux session named "goes" with

```
tmux new -s goes
```

Then, split the terminal window into three separate panes with the following key commands:

Ctrl + b, "

Ctrl + b, %

There should now be three panes visible within the one terminal window. Using Ctrl + b, followed by the arrow keys, you can switch between the panes.

In one of the bottom panes, run the following command to start processing received GOES data:

```
goesproc -c ~/goestools/confs/goesproc.conf -m packet --subscribe tcp://127.0.0.1:5004
```

Where -c points to the config file, -m refers to "packet" mode (as opposed to reading the data from a previously saved file), and –-subscribe points to the tcp address and port above, which goesrecv outputs from.

After a moment, the pane will say it is waiting for the first packet to arrive.

In the top pane, run the following command to start receiving GOES data:

```
goesrecv -v -i 5 -c ~/goestools/confs/goesrecv.conf
```

Where -v is "verbose," and -i 5 modifies the verbosity to output at every 5 seconds, and -c again points to the config file.

After the time specified (5 seconds), this pane will show the status of the GOES signal received.

```
2019-12-05T02:20:12Z [monitor] gain: 53.72, freq:  3422.2, omega: 3.236, vit(avg):  296, rs(sum):  836, packets: 575, drops:   0
2019-12-05T02:20:22Z [monitor] gain: 53.74, freq:  3439.4, omega: 3.236, vit(avg):  295, rs(sum):  790, packets: 566, drops:   0
2019-12-05T02:20:32Z [monitor] gain: 53.72, freq:  3419.4, omega: 3.236, vit(avg):  292, rs(sum):  692, packets: 566, drops:   0
2019-12-05T02:20:42Z [monitor] gain: 53.69, freq:  3422.2, omega: 3.236, vit(avg):  290, rs(sum):  749, packets: 565, drops:   0
2019-12-05T02:20:52Z [monitor] gain: 53.78, freq:  3388.9, omega: 3.236, vit(avg):  289, rs(sum):  710, packets: 566, drops:   0
2019-12-05T02:21:02Z [monitor] gain: 53.73, freq:  3386.5, omega: 3.236, vit(avg):  296, rs(sum):  700, packets: 566, drops:   0
2019-12-05T02:21:12Z [monitor] gain: 53.76, freq:  3420.5, omega: 3.236, vit(avg):  292, rs(sum):  716, packets: 566, drops:   0
2019-12-05T02:21:22Z [monitor] gain: 53.69, freq:  3429.7, omega: 3.236, vit(avg):  292, rs(sum):  779, packets: 565, drops:   0
2019-12-05T02:21:32Z [monitor] gain: 53.73, freq:  3456.9, omega: 3.236, vit(avg):  288, rs(sum):  771, packets: 566, drops:   0
2019-12-05T02:21:42Z [monitor] gain: 53.80, freq:  3442.9, omega: 3.236, vit(avg):  292, rs(sum):  744, packets: 566, drops:   0
2019-12-05T02:21:52Z [monitor] gain: 53.70, freq:  3454.6, omega: 3.236, vit(avg):  296, rs(sum):  801, packets: 566, drops:   0
2019-12-05T02:22:02Z [monitor] gain: 53.73, freq:  3441.1, omega: 3.236, vit(avg):  294, rs(sum):  739, packets: 565, drops:   0
2019-12-05T02:22:12Z [monitor] gain: 53.79, freq:  3460.2, omega: 3.236, vit(avg):  294, rs(sum):  781, packets: 566, drops:   0
2019-12-05T02:22:22Z [monitor] gain: 53.78, freq:  3463.4, omega: 3.236, vit(avg):  292, rs(sum):  739, packets: 567, drops:   0
2019-12-05T02:22:32Z [monitor] gain: 53.76, freq:  3443.9, omega: 3.236, vit(avg):  298, rs(sum):  840, packets: 565, drops:   0
2019-12-05T02:22:42Z [monitor] gain: 53.78, freq:  3450.3, omega: 3.236, vit(avg):  301, rs(sum):  817, packets: 565, drops:   0
2019-12-05T02:22:52Z [monitor] gain: 53.78, freq:  3429.3, omega: 3.236, vit(avg):  304, rs(sum):  797, packets: 567, drops:   0
2019-12-05T02:23:02Z [monitor] gain: 53.73, freq:  3451.4, omega: 3.236, vit(avg):  301, rs(sum):  820, packets: 566, drops:   0
2019-12-05T02:23:12Z [monitor] gain: 53.66, freq:  3439.4, omega: 3.236, vit(avg):  303, rs(sum):  790, packets: 566, drops:   0
2019-12-05T02:23:22Z [monitor] gain: 53.71, freq:  3451.5, omega: 3.236, vit(avg):  298, rs(sum):  838, packets: 566, drops:   0
2019-12-05T02:23:32Z [monitor] gain: 53.68, freq:  3446.3, omega: 3.236, vit(avg):  294, rs(sum):  774, packets: 565, drops:   0
2019-12-05T02:23:42Z [monitor] gain: 53.68, freq:  3468.2, omega: 3.236, vit(avg):  297, rs(sum):  783, packets: 566, drops:   0
```

Figure 4: *goesrecv output for successful reception*

The frequency output shows the difference between what the SDR finds the signal's center frequency to be compared to the actual, expected center frequency of the signal.

The Vit(avg) output is a measure of error in the signal, as is rs(sum). These stand for Viterbi and Reed-Solomon. They are methods of error detection and correction. Lower is better!

Packets and drops compare the number of data packets correctly assembled and sent to goesproc versus the number deemed erroneous and dropped. A perfect reception, as shown in the image above, would have all packets sent and 0 dropped.

The goesproc pane will also output information on occasion. It will give an error message when a packet is dropped, and it will also alert you to the creation of any new processed data files and tell you where they are stored.

Accessing Processed Data:

You just saw goesproc output a file: how do you access it? If you've logged onto your Pi remotely, it can be difficult. The software Rclone will help us send our data directly to Google Drive (or your preferred storage service).

Install rclone with the provided script:

```
curl https://rclone.org/install.sh | sudo bash
```

I will not go into details regarding setting up rclone, but a very good tutorial that is specific to your cloud storage of choice can be found at the rclone website.

https://rclone.org/install/

Whatever you use, make sure it has a lot of space. Within a month, I already have 100 GBs of image data. I recommend taking advantage of your .edu email's unlimited google drive storage.

Now, if we want to automatically remove data files from our Raspberry Pi onto our Google Drive, we will once again edit the crontab.

```
crontab -e
```

At the bottom, add another 3 lines as follows:

```
*/5 * * * * rclone move –retries 6 –retries-sleep 60 ~/goestools/goes16 YOUR_DRIV
E_NAME:goes16


*/5 * * * * rclone move –retries 6 –retries-sleep 60 ~/goestools/goes17 YOUR_DRIV
E_NAME:goes17


*/5 * * * * rclone move –retries 6 –retries-sleep 60 ~/goestools/himawari YOUR_DR
IVE_NAME:himawari


*/5 * * * * rclone move –retries 6 –retries-sleep 60 ~/goestools/nws YOUR_DRIVE_N
AME:nws
```

This will remove every file in the goes16, goes17, himawari, and nws folder and send it to the cloud storage drive. This is good for managing the Pi's space, since a microSD card can fill up rather quickly without regularly emptying it. It is important that you list the commands for satellites you are not explicitly receiving, because GOES-16, GOES-17, and Japan's Himawari 8 occasionally broadcast data between each other, so you may receive images taken by GOES-17 via GOES-16.

The numbers and asterisks on the left side signify the frequency with which the commands execute. The code above is set to run every 5 minutes, but you can change it as you see fit. The crontab guru is a handy tool to understand cron schedule expressions.

https://crontab.guru/

Save, and exit the crontab.

As an aside, something I found useful is that you can use the following command to determine the amount of data you have sent to your rclone drive.

```
rclone size "YOUR_DRIVE_NAME:"
```

Understanding the Data

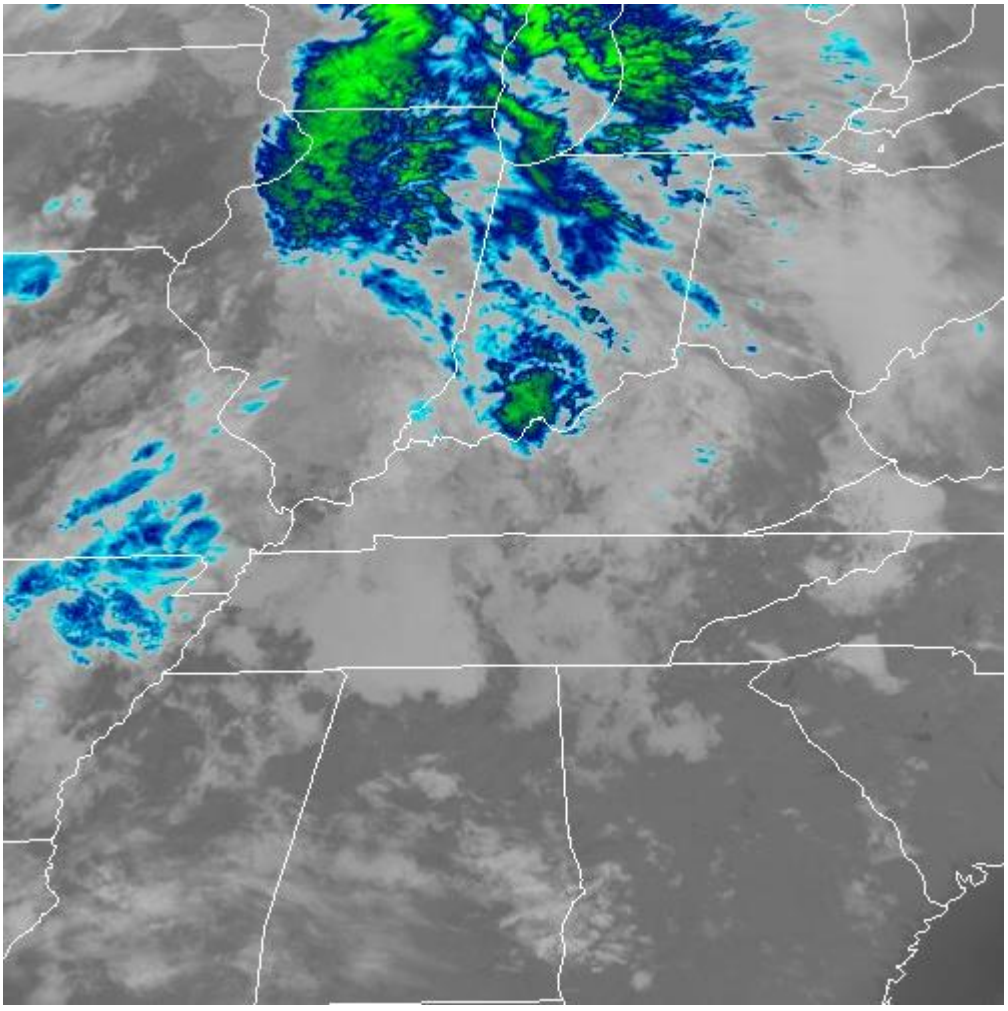

Figure 5: *A typical GOES-16 image of the Midwest using of ABI channel 13 (long-wave IR), enhanced with colors to show the concentration of a certain type of cloud.*

The data will be categorized into a few folders that will be in the ~/goestools directory or one level below your rclone drive in cloud storage. The goes16 and goes17 directories will contain reduced resolution images from their respective satellites. They will be further divided within their directories into folders called “fd” for full disk image, “m1” and “m2” for mesoscale regions 1 and 2. These mesoscale regions may change over time as the satellite is controlled by the NOAA. Within the fd, m1, and m2 folders, there is another layer of folders labelled with channel numbers, and an extra folder “fc” for false color. The channel numbers correspond to the wavelength channels available to the Advanced Baseline Imager (ABI). The false color images are generated from observing certain wavelength bands and applying a false color mapping. All 16 ABI bands and their specifications can be viewed in the table in **Appendix C**, which was compiled on the GOES-16 Wikipedia page from ABI factsheets published by the NOAA. For further reading, links to these factsheets are in the “sources” column of the table.

Images in the NWS folder are EMWIN data, created by the NWS and NOAA to be disseminated as a public service. EMWIN data can include images and text documents containing forecasts and disaster warnings.

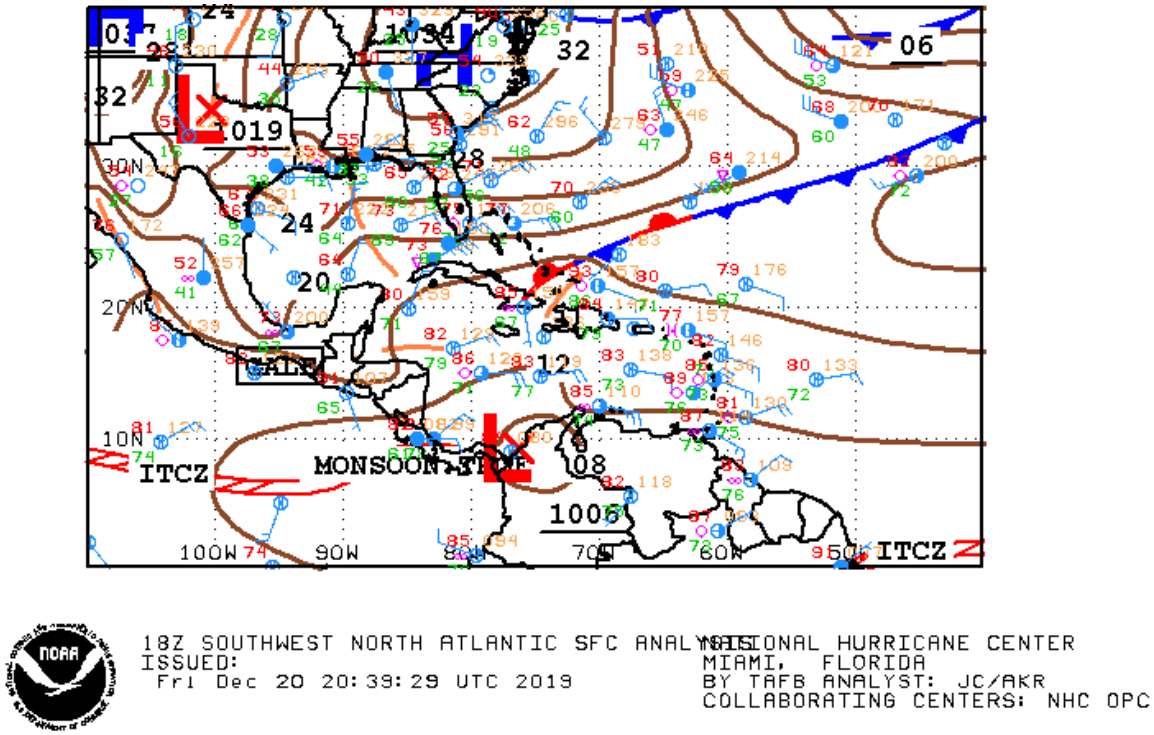

Figure 6: *National Hurricane Center Forecast for Dec 20, 2019*

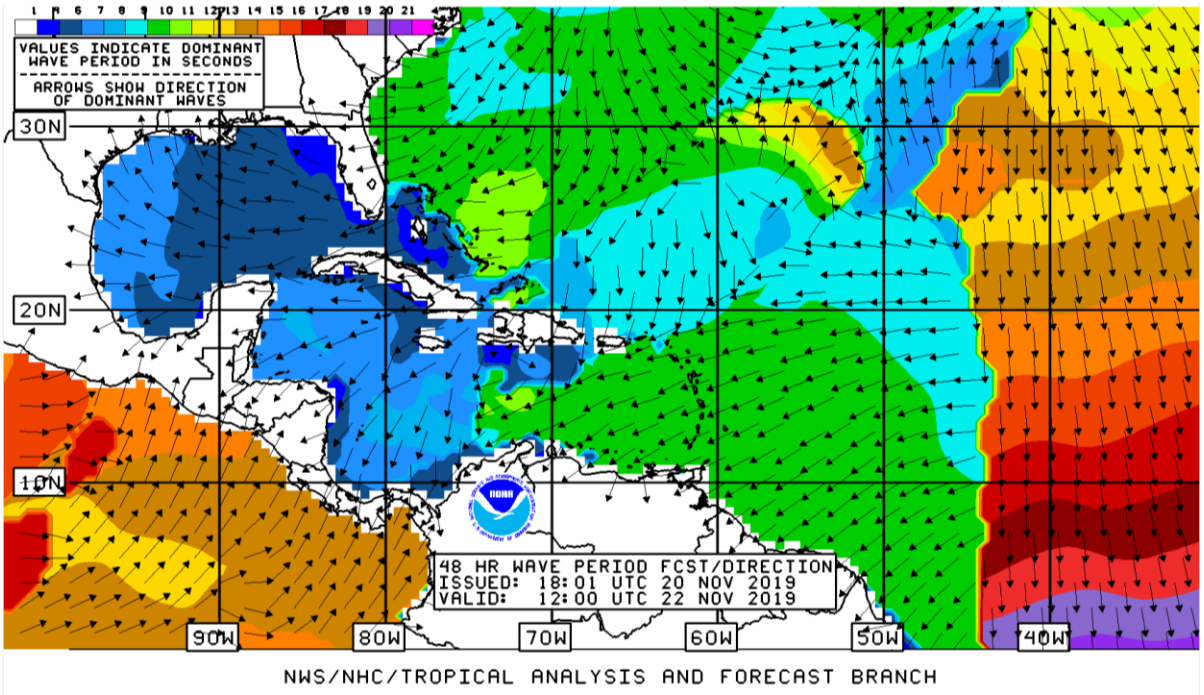

Figure 7: *Wave Period and Direction Forecast for Nov. 20-22 2019*

Appendix A: goesrecv.conf (for Airspy)

```
[demodulator]
## Use LRIT mode for GOES-15.
# mode = "lrit"
## Use HRIT mode for GOES-16 or later.
 mode = "hrit"
source = "airspy"

# The section below configures the sample source to use.
#
# You can leave them commented out to use the default values for the
# demodulator mode you choose ("lrit" or "hrit"). To use and configure
# any of them, uncomment the section below, and change the demodulator
# source field to match the source you want to use.
#

# [airspy]
# frequency = 1694100000
##
## By default, goesrecv will use the lowest sample rate available.
## This is 2.5 MSPS for the R2 and 3.0 MSPS for the Mini.
## Because different Airspy models support different sample rates,
## it is recommended to leave the "sample_rate" field commented,
## so that it works for either model.
##
# sample_rate = 3000000
# gain = 18
# bias_tee = false

# [rtlsdr]
# frequency = 1694100000
# sample_rate = 2400000
# gain = 30
# bias_tee = false
# device_index = 0

# [nanomsg]
# sample_rate = 2400000
# connect = "tcp://1.2.3.4:5005"
# receive_buffer = 2097152

[costas]
max_deviation = 200e3

[clock_recovery.sample_publisher]
bind = "tcp://0.0.0.0:5002"
send_buffer = 2097152

[quantization.soft_bit_publisher]
bind = "tcp://0.0.0.0:5001"
send_buffer = 1048576
```

```
[decoder.packet_publisher]
bind = "tcp://0.0.0.0:5004"
send_buffer = 1048576

# The demodulator stats publisher sends a JSON object that describes
# the state of the demodulator (gain, frequency correction, samples
# per symbol), for every block of samples.
[demodulator.stats_publisher]
bind = "tcp://0.0.0.0:6001"

# The decoder stats publisher sends a JSON object for every packet it
# decodes (Viterbi corrections, Reed-Solomon corrections, etc.).
[decoder.stats_publisher]
bind = "tcp://0.0.0.0:6002"

# The monitor can log aggregated stats (counters, gauges, and
# histograms) to a statsd daemon. Because this uses UDP, you can keep
# this enabled even if you haven't setup a statsd daemon yet.
[monitor]
statsd_address = "udp4://localhost:8125"
```

Appendix B: goesproc.conf

```
# Example goesproc configuration file to process GOES-R series products.
#

# Store all original GOES-16 products.
[[handler]]
type = "image"
origin = "goes16"
directory = "./goes16/{region:short|lower}/{channel:short|lower}/{time:%Y-%m-%d}"
filename = "GOES16_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Store all original GOES-17 products.
[[handler]]
type = "image"
origin = "goes17"
directory = "./goes17/{region:short|lower}/{channel:short|lower}/{time:%Y-%m-%d}"
filename = "GOES17_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
```

```
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# GOES-16 ABI false color.
[[handler]]
type = "image"
origin = "goes16"
regions = [ "fd", "m1", "m2" ]
channels = [ "ch02", "ch13" ]
directory = "./goes16/{region:short|lower}/fc/{time:%Y-%m-%d}"
filename = "GOES16_{region:short}_FC_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  [handler.remap.ch02]
  path = "/usr/local/share/goestools/wxstar/wxstar_goes16_ch02_curve.png"

  [handler.lut]
  path = "/usr/local/share/goestools/wxstar/wxstar_goes16_lut.png"

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# GOES-16 ABI RGB-enhanced
[[handler]]
type = "image"
origin = "goes16"
regions = [ "fd", "m1", "m2" ]
channels = [ "ch07", "ch08", "ch09", "ch13", "ch14", "ch15" ]
directory =
"./goes16/{region:short|lower}/{channel:short|lower}_enhanced/{time:%Y-%m-%d}"
filename = "GOES16_{region:short}_{channel:short}_enhanced_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  ## The following gradients are rough approximations of the
  ## McIDAS RGB enhancements used by NOAA/NESDIS/STAR on their site..
  ##
  ## For more info:
  ##
  ##   https://www.star.nesdis.noaa.gov/GOES/GOES16_FullDisk.php
  ##   http://cimss.ssec.wisc.edu/goes/visit/water_vapor_enhancement.html
```

```
##   http://cimss.ssec.wisc.edu/goes/visit/enhanced_v_enhancements.html

## Shortwave IR (Channel 7)
[handler.gradient.ch07]
points = [
  { units = 400, color = "#000000" },
  { units = 250, color = "#b9b9b9" },
  { units = 249.999, color = "#00ffff" },
  { units = 240, color = "#000080" },
  { units = 230, color = "#00ff00" },
  { units = 220, color = "#ffff00" },
  { units = 210, color = "#ff0000" },
  { units = 200, color = "#000000" },
  { units = 190, color = "#ffffff" }
]

## Water Vapor (Channels 8 and 9)
[handler.gradient.ch08]
points = [
  { units = 276, color = "#000000" },
  { units = 275.9, color = "#ff0000" },
  { units = 258, color = "#ffff00" },
  { units = 250, color = "#000070" },
  { units = 233, color = "#ffffff" },
  { units = 195, color = "#408020" },
  { units = 178, color = "#00ffff" }
]
[handler.gradient.ch09]
points = [
  { units = 276, color = "#000000" },
  { units = 275.9, color = "#ff0000" },
  { units = 258, color = "#ffff00" },
  { units = 250, color = "#000070" },
  { units = 233, color = "#ffffff" },
  { units = 195, color = "#408020" },
  { units = 178, color = "#00ffff" }
]

## Longwave IR (Channels 13, 14, and 15)
[handler.gradient.ch13]
points = [
  { units = 333, color = "#000000" },
  { units = 238, color = "#b9b9b9" },
  { units = 237.999, color = "#00ffff" },
  { units = 228, color = "#000080" },
  { units = 218, color = "#00ff00" },
  { units = 208, color = "#ffff00" },
  { units = 198, color = "#ff0000" },
  { units = 188, color = "#000000" },
  { units = 178, color = "#ffffff" }
]
[handler.gradient.ch14]
```

```
  points = [
    { units = 333, color = "#000000" },
    { units = 238, color = "#b9b9b9" },
    { units = 237.999, color = "#00ffff" },
    { units = 228, color = "#000080" },
    { units = 218, color = "#00ff00" },
    { units = 208, color = "#ffff00" },
    { units = 198, color = "#ff0000" },
    { units = 188, color = "#000000" },
    { units = 178, color = "#ffffff" }
  ]
  [handler.gradient.ch15]
  points = [
    { units = 333, color = "#000000" },
    { units = 238, color = "#b9b9b9" },
    { units = 237.999, color = "#00ffff" },
    { units = 228, color = "#000080" },
    { units = 218, color = "#00ff00" },
    { units = 208, color = "#ffff00" },
    { units = 198, color = "#ff0000" },
    { units = 188, color = "#000000" },
    { units = 178, color = "#ffffff" }
  ]

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# GOES-17 ABI false color.
[[handler]]
type = "image"
origin = "goes17"
regions = [ "fd", "m1", "m2" ]
channels = [ "ch02", "ch13" ]
directory = "./goes17/{region:short|lower}/fc/{time:%Y-%m-%d}"
filename = "GOES17_{region:short}_FC_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  # This reuses the GOES-16 contrast curve assuming it is identical
  [handler.remap.ch02]
  path = "/usr/local/share/goestools/wxstar/wxstar_goes16_ch02_curve.png"

  # This reuses the GOES-16 LUT assuming it is identical
  [handler.lut]
  path = "/usr/local/share/goestools/wxstar/wxstar_goes16_lut.png"

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"
```

```
  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Images relayed from Himawari-8.
[[handler]]
type = "image"
origin = "himawari8"
directory = "./himawari8/{region:short|lower}/{time:%Y-%m-%d}"
filename = "Himawari8_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# NWS text (weather reports).
[[handler]]
type = "text"
origin = "nws"
directory = "./nws/{time:%Y-%m-%d}"
filename = "{time:%Y%m%dT%H%M%SZ}_{awips:nnn}{awips:xxx}"
json = false

# NWS images.
[[handler]]
type = "image"
origin = "nws"
directory = "./nws/{time:%Y-%m-%d}"
filename = "{time:%Y%m%dT%H%M%SZ}_{filename}"
format = "png"
json = false

# Miscellaneous text.
[[handler]]
type = "text"
origin = "other"
directory = "./text/{time:%Y-%m-%d}"
filename = "{time:%Y%m%dT%H%M%SZ}_{filename}"
json = false

# Store relayed GOES-15 full disks
[[handler]]
type = "image"
origin = "goes15"
regions = [ "fd" ]
directory = "./goes15/{region:short|lower}/{time:%Y-%m-%d}"
```

```
filename = "GOES15_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
crop = [ -2374, 2371, -1357, 1347 ]
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Store relayed GOES-15 northern hemisphere region
[[handler]]
type = "image"
origin = "goes15"
regions = [ "nh" ]
directory = "./goes15/{region:short|lower}/{time:%Y-%m-%d}"
filename = "GOES15_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
crop = [ -1864, 1447, -1357, -3 ]
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Store relayed GOES-15 southern hemisphere region
[[handler]]
type = "image"
origin = "goes15"
regions = [ "sh" ]
directory = "./goes15/{region:short|lower}/{time:%Y-%m-%d}"
filename = "GOES15_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
crop = [ -1864, 896, -19, 1043 ]
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Store relayed GOES-15 US region
[[handler]]
type = "image"
origin = "goes15"
```

```
regions = [ "us" ]
directory = "./goes15/{region:short|lower}/{time:%Y-%m-%d}"
filename = "GOES15_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
crop = [ -1312, 1542, -1327, -345 ]
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# Store relayed GOES-15 special regions
# No crop specified because it is expected to move around
[[handler]]
type = "image"
origin = "goes15"
regions = [ "si00", "si01", "si02", "si03", "si04" ]
directory = "./goes15/{region:short|lower}/{time:%Y-%m-%d}"
filename = "GOES15_{region:short}_{channel:short}_{time:%Y%m%dT%H%M%SZ}"
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"

# The following handler takes the same crop from the FD, NH, and US
# products to get more frequent imagery of a smaller area on the
# northern hemisphere. The crop region is a combination of the NH and
# US crop regions.
[[handler]]
type = "image"
origin = "goes15"
regions = [ "fd", "nh", "us" ]
crop = [ -1312, 1447, -1327, -345 ]
directory = "./goes15/combine-north/{time:%Y-%m-%d}"
filename = "GOES15_{channel:short}_{time:%Y%m%dT%H%M%SZ}_{region:short}"
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"
```

```
# The following handler takes the same crop from the FD and SH
# products to get more frequent imagery of a smaller area on the
# southern hemisphere.
[[handler]]
type = "image"
origin = "goes15"
regions = [ "fd", "sh" ]
crop = [ -1864, 896, -19, 1043 ]
directory = "./goes15/combine-south/{time:%Y-%m-%d}"
filename = "GOES15_{channel:short}_{time:%Y%m%dT%H%M%SZ}_{region:short}"
format = "jpg"
json = false

  [[handler.map]]
  path = "/usr/local/share/goestools/ne/ne_50m_admin_0_countries_lakes.json"

  [[handler.map]]
  path =
"/usr/local/share/goestools/ne/ne_50m_admin_1_states_provinces_lakes.json"
```

Appendix C: ABI Specifications

Table 2: Advanced Baseline Imager Channel Specifications

Band	λ (μm)	Central λ (μm)	Pixel spacing (km)	Nickname	Classification	Primary function	Source
1	0.45–0.49	0.47	1	Blue	Visible	Aerosols	[37]
2	0.59–0.69	0.64	0.5	Red	Visible	Clouds	[38]
3	0.846–0.885	0.865	1	Veggie	Near-infrared	Vegetation	[39]
4	1.371–1.386	1.378	2	Cirrus	Near-infrared	Cirrus	[40]

5	1.58–1.64	1.61	1	Snow/Ice	Near-infrared	Snow/ice discrimination, cloud phase	[41]
6	2.225–2.275	2.25	2	Cloud Particle Size	Near-infrared	Cloud particle size, snow cloud phase	[42]
7	3.80–4.00	3.90	2	Shortwave Window	Infrared	Fog, stratus, fire, volcanism	[43]
8	5.77–6.6	6.9	2	Upper-level Tropospheric Water Vapor	Infrared	Various atmospheric features	[44]
9	6.75–7.15	6.95	2	Mid-level Tropospheric Water Vapor	Infrared	Water vapor features	[45]
10	7.24–7.44	7.34	2	Lower-level Tropospheric Water Vapor	Infrared	Water vapor features	[46]
11	8.3–8.7	8.5	2	Cloud-Top Phase	Infrared	Cloud-top phase	[47]
12	9.42–9.8	9.61	2	Ozone	Infrared	Total column ozone	[48]
13	10.1–10.6	10.35	2	Clean Infrared Longwave Window	Infrared	Clouds	[49]
14	10.8–11.6	11.2	2	Infrared Longwave Window	Infrared	Clouds	[50]

15	11.8–12.8	12.3	2	Dirty Infrared Longwave Window	Infrared	Clouds	[51]
16	13.0–13.6	13.3	2	CO_2 Longwave Infrared	Infrared	Air temperature, clouds	[52]