Assembling your SeaGlass IMSI-catcher detector – a beginner's guide

Linus Nyman
Hanken School of Economics
Helsinki, Finland
linus.nyman@hanken.fi
https://orcid.org/0000-0001-5051-1683

Mikael Laakso
Hanken School of Economics
Helsinki, Finland
mikael.laakso@hanken.fi
https://orcid.org/0000-0003-3951-7990

Abstract
IMSI-catchers, sometimes called cell-site simulators or Stingrays, can eavesdrop on cell phone communications. IMSI-catchers do this by pretending to be a cell tower, thereby tricking nearby cell phones into connecting to them. The SeaGlass IMSI-catcher detector (which we from now on will call "SeaGlass sensor" or just "sensor") gathers data from surrounding cell towers. Thus, the sensor should also pick up nearby IMSI-catchers. Analyzing SeaGlass sensor data for anomalies can enable identifying IMSI-catchers.

The SeaGlass sensor is built using off-the-shelf parts. It is based on a Raspberry Pi and requires no soldering. This guide covers the steps to assemble a SeaGlass sensor, including both hardware and software requirements. The guide also covers the installation of software, on a separate computer, that can be used to analyze data gathered by the SeaGlass sensor.

Background
The SeaGlass project was first introduced in the paper *SeaGlass: Enabling City-Wide IMSI-Catcher Detection* by Peter Ney, Ian Smith, Tadayoshi Kohno and Gabriel Cadamuro, presented at the Privacy Enhancing Technology Symposium 2017 (Ney et al. 2017). More information about the SeaGlass project is available on the SeaGlass website[1] as well as its GitHub page[2]. A presentation of the project, by Ney and Smith, is available on YouTube[3].

This guide is based on the SeaGlass "Sensor Installation" guide[4] by Peter Ney. In this guide, those instructions have been expanded upon with the goal of making assembling a SeaGlass sensor more approachable for the less technically savvy. We cover the necessary steps for setting up a stationary SeaGlass sensor as well as a separate computer for data analysis. This guide does not cover the steps required to deploy a SeaGlass sensor in a vehicle, e.g. issues related to hotspotting or setting up a mobile power supply. We also do not cover data analysis. For more information on those topics, see the SeaGlass project's documentation.

The remainder of the guide is structured as follows. Section 1 discusses the required hardware. Section 2 covers setting up the SeaGlass sensor. Section 3 covers running the sensor and handling the data it generates. Section 4 covers restoring the data. Section 5 covers setting up a separate computer for the analysis and readying your data for analysis.

Acknowledgements
The authors thank Peter Ney for all his help during the writing of this guide.

[1] https://seaglass.cs.washington.edu/
[2] https://github.com/seaglass-project/seaglass
[3] https://www.youtube.com/watch?v=8-G2EJ-1WTg
[4] https://github.com/seaglass-project/seaglass/blob/master/README.md

TABLE OF CONTENTS

1 An overview of the hardware you will need

To assemble a SeaGlass sensor as described in this guide you will need both hardware and software. In this section we cover the hardware requirements.

1.1 A Linux-based computer

The SeaGlass source code is written for Linux-based computers. While it can be any Linux-based computer, this guide is written using a **Raspberry Pi 3**.

We recommend using a more powerful computer than the Raspberry Pi for the analysis of the data gathered by the SeaGlass sensor. The analysis computer does not have to be a Linux-based computer.

1.2 Peripheral devices

The SeaGlass sensor uses a GPS, modem, and antenna to gather data.

A GPS sensor

If you want the SeaGlass sensor to be able to identify its location, you will need a GPS. This guide is written using a **GlobalSat G-STAR IV, Model No BU-353S4**. (However, any GPS that works with gpsd [see section 2.3] should do.)

A GSM Modem

The GSM modem is a crucial part of the SeaGlass sensor, as it is the piece of hardware that acquires information from surrounding cell towers. This guide is written using a **Telit GT864-PY V2**. The Telit was chosen for the breadth of data it can gather.

NOTE: if you use a different modem you will need to modify the SeaGlass source code.

An antenna

Future versions of the SeaGlass sensor will focus more on maximizing the potential of the antenna. However, a 20-30 dollar, or euro, antenna should meet your basic requirements.

1.3 A SIM card and service

You won't be using the SIM card to make any calls. Rather, it is used to get surrounding cell towers to supply the SeaGlass sensor with data. However, for this to be possible it needs to be a functioning SIM card with cell service.

Regarding choice of network: less advanced IMSI-catchers may only simulate (i.e. work on) one network provider's network, while more advanced IMSI-catchers may simulate several. If you have reason to believe some specific network is more interesting to potential IMSI-catcher users, chose a SIM card by that provider.

1.4 Various power supplies, connectors and adapters

Power supplies

- A power supply for the computer. The Raspberry Pi 3 uses micro USB.
- A power supply for the modem. The Telit modem uses an unusual power cable: what you may remember as a landline phone plug. (Yes, it's weird. No, we don't know why it uses this.) Some further discussion of the requirements and concerns of this power supply, as well as how to create one using off-the-shelf parts, are discussed in **Appendix 2. On powering the Telit modem**.

Connectors

- A cable to connect the SeaGlass sensor computer to a monitor. The Raspberry Pi 3 uses HDMI.
- A cable to connect the modem to the SeaGlass sensor computer. The important term to know when shopping for this cable is that you will be connecting a so-called serial device, or serial port, to a USB. The Telit modem has a serial port that needs to be connected to a USB port on the Raspberry Pi. The cable we use is called a **USB to RS323 Converter**. ("RS323" is the name of the serial standard.)

Adapters

- Regarding the antenna, remember to check whether you need an adapter to connect the antenna to your modem.
- The Raspberry Pi 3 uses HDMI. If your monitor doesn't have an HDMI port, you will need an adapter.

1.5 Miscellaneous

A piece of metal

The antenna will be more effective if you place it on a piece of metal (10×10 inches or so should do the trick). If you don't have anything lying around you can, for instance, use a cover plate. These can be found in pretty much any DIY or hardware store.

2 Setting up your SeaGlass sensor

This part covers plugging in and preparing your peripheral devices, as well as installing the software you will need to run your SeaGlass sensor. This section assumes you are familiar with what a "terminal" is. If you aren't, please start by reading **Appendix 1. Some basics about Linux**. (We also assume you have an Internet connection.)

2.1 Update the operating system

The first thing to do is make sure you are using the latest version of the Raspbian operating system. (Raspbian is the default operating system on a Raspberry Pi.) To do this, open a terminal and type:

```
sudo apt-get update
```

This checks for newer versions of installed software. Then type the following command:

```
sudo apt-get upgrade
```

This updates your system to newer versions of installed software, if available. You need to press Y and Enter to confirm. Depending on when the operating system was last updated, this process might take a while.

2.2 Install the SeaGlass software

The SeaGlass software can be found on a site called GitHub (https://github.com/seaglass-project/seaglass.). We have also uploaded a ZIP file with the files as of 11th July 2018 as an appendix to this guide. As of the time of writing the guide is compatible with both versions, but the GitHub version may change over time, to ensure compatibility with this guide please use the files in the appendix.

To install the SeaGlass software on your Raspberry Pi, follow these steps:

1) Chose either version of the ZIP file: from this guide's Zenodo page or Github
2) Download the file (on Github: click on the green "Clone or download" button, then click on "Download ZIP")
3) When it is done downloading, extract the file. We recommend extracting it to the home folder. To do that, follow these steps:

- right-click on the file
- choose "Extract to..."
- click on the folder icon
- click on "pi" in the left hand "Places" menu
- click on "Open" to confirm
- click on "Extract"

2.3 Install and configure gpsd, reboot your system

The program gpsd is what is called a "deamon" – a program that runs as a background process rather than as something the user is in direct control of. gpsd is used to receive data from the GPS and pass that data on to the applications that will need it. There are three steps to installing and configuring gpsd, followed by a reboot of the system (i.e. turning your Raspberry Pi off and then on again).

2.3.1 Install gpsd

The first step is to install the program gpsd. To do this, open a terminal and type:

```
sudo apt-get install gpsd
```

When prompted, press Y and Enter to confirm the install.

2.3.2 Disable auto connect and hard-code path to the serial device

To prevent interference with the Telit modem, you will need to disable auto connect mode in gpsd. Instead of using auto connect you will hard-code the path to the GPS serial device. To accomplish this, do the following:

1) Make sure the GPS device is plugged in.
2) In the terminal, type:

```
sudo dpkg-reconfigure gpsd
```

You should see the following text:

```
Warning: Stopping gpsd.service, but it can still be
activated by:
  gpsd.socket
Creating/updating gpsd user account…
```

Now you need to manually edit the gpsd configuration file. The file that needs to be changed is called "gpsd". It is located in: /etc/default. To manually edit the file, follow these steps:

1) In the terminal, type:

```
sudo leafpad /etc/default/gpsd
```

This will open the file “gpsd” using a text editor called Leafpad.

2) In the gpsd file, find where it says: ‘Use USB hotplugging to add new USB devices automatically to the daemon USBAUTO=”true”’. In that sentence, change the word “true” to instead read “false”.

3) Find where it says “# Other options you want to pass to gpsd”. Modify that section to read:

```
# Other options you want to pass to gpsd
GPSD_OPTIONS="/dev/ttyUSB0"
```

After making these edits, in Leafpad, select File and then Save. You can now close Leafpad.

2.3.3 Add your user to the dialout group

To make sure you (i.e. your user account on Raspbian) can connect to the GPS serial device, you need to add your user to the dialout group. To do this, in the terminal, type:

```
sudo usermod -a -G dialout <your-username>
```

Where you should exchange <your-username> for your username. The default username is “pi” (and, FYI, the default password is “raspberry”). So, if you haven’t changed your username, then you should type:

```
sudo usermod –a –G dialout pi
```

(You will see a long list of options. You can ignore these.)

2.3.4 Re-login to (or reboot) your computer

For the group change to take effect, you will need to re-login to, or reboot, your computer. You can either do this through the top left raspberry icon, or type (in the terminal):

```
sudo reboot now
```

This will turn the Raspberry Pi first off, then on again.

2.4 Install MongoDB-server

SeaGlass automatically writes the data it gathers from the cellular scan and GPS to a MongoDB database. To install MongoDB, type (in the terminal):

```
sudo apt-get install mongodb-server
```

Press Y, and then Enter to confirm.

2.5 Install the pip tool, Python 3, and the packages pySerial and pyMongo

The next step is to install python (version 3) and some so-called “packages.” Packages are files containing code for the computer to run. The packages you will need are pySerial and pyMongo. We suggest using the “pip” tool to install them, and the installation process described below uses pip.

2.5.1 Install the pip tool

Some versions of Raspbian (which, as you may remember, is Raspberry Pi's default operating system) come with pip already installed. However, following the steps to install it, even if you already have it installed, is completely harmless.

To install pip, open a terminal and type:

```
sudo apt-get install python3-pip
```

If you already have it installed, you will see some text that includes the following: "`python3-pip is already the newest version`". If you do not have the most recent version, or you have no previous version installed, press Y to confirm the installation.

2.5.2 Install pySerial

When you have pip installed it is time to install pySerial. To do this, type:

```
sudo pip3 install pyserial
```

Note: lowercase "p" in pip3.

2.5.3 Install pyMongo

As of this writing, SeaGlass doesn't work with more recent versions of pyMongo than version 3.5.1. To install a compatible version of pyMongo, open a terminal and type:

```
sudo pip3 install 'pyMongo<3.6.0'
```

Note: lowercase "p" in pip3.

2.6 Prepare the SIM card and modem

The modem requires a SIM card to work. Additionally, SIM cards by default commonly require a PIN number to unlock them.

2.6.1 Removing the PIN requirement from a SIM card

You need to disable the PIN requirement for the modem to be able to access the SIM card. You can ask for the PIN requirement to be disabled when buying the SIM card. Alternatively, you can use a cellphone to disable it yourself. To do this, first insert the SIM card into a cellphone. Then look for the option to disable the PIN. This is most likely located under "settings" and "security".

2.6.2 Insert the SIM card into the modem

Here we assume that you are using the same Telit modem as in the original SeaGlass. Slide off the small cover on the top of the modem. It has an arrow on it indicating which way to slide it. Removing the cover will expose a small compartment for the SIM card. The compartment can be pulled slightly, and can then be folded out.
SIM cards come in varying sizes, most commonly mini, micro, and nano. When purchasing a SIM card it will usually come in credit-card size (called "full-SIM"), allowing you to choose what size SIM card to "push out" of the credit card.

The Telit uses a mini-SIM card (also called standard SIM, regular SIM, or 2FF). This is the largest of the available sizes (apart from the credit-card sized option). If your SIM card is smaller than mini-SIM, you will need to get an adapter for it. A SIM card adapter is basically

just a piece of plastic that holds the SIM card in place. Such adapters are sold pretty much anywhere phones or electronics are sold.

2.7 Ensure the correct GPS and modem path names

Your Raspberry Pi will assign path names to the GPS and modem when you plug them in. For the SeaGlass sensor source code to work, the path name to the modem needs to be correct. More specifically, the GPS needs to be given the path name `ttyUSB0` and the modem needs to be given the path name `ttyUSB1`. You can get these path names to be assigned correctly simply by plugging in the GPS and modem in in the right order. When plugging in your GPS and modem, the Raspberry Pi will assign the path name ttyUSB0 to the one you plug in first, and ttyUSB1 to the one you plug in second.

Note: To ensure correct path names, follow these steps every time you reboot the raspberry.

To ensure the correct path names, follow these steps:

1) First, unplug your GPS and modem. (You don't need to unplug your keyboard or mouse.)
2) Reboot your Raspberry Pi, e.g. using the terminal command:

```
sudo reboot now
```

3) Once the Raspberry Pi is up and running again, plug in the GPS. Then type:

```
ls /dev/ttyUSB*
```

This should result in the text: `/dev/ttyUSB0`. If you get a different path name, make sure you don't have any peripheral USB devices plugged in (apart from a mouse and keyboard) and then redo from step 2.
4) Leave the GPS plugged in. Now plug in your modem. If you want, you can once again run:

```
ls /dev/ttyUSB*
```

Now you should see both `/dev/ttyUSB0` and `/dev/ttyUSB1`. As long as you plugged the modem in after the GPS, the modem should now have the correct path name (i.e. `/dev/ttyUSB1`).

That's it – you're ready to fire up your sensor!

Note on hotplugging: Correct path name assignments could be accomplished through something called "hotplugging". A discussion of how this is done is beyond the scope of this paper, but guides and information can be found online. However, the GPS and modem used in this guide use the same drivers, which is likely to make hotplugging difficult if not impossible. Consider using a different GPS if you want to implement hotplugging.

3 Adjusting scan intervals, running the sensor, handling data

This part first covers how you can adjust the scan interval, meaning how often the sensor runs a scan. Then it covers the necessary steps to run the sensor, copy or move the data generated, as well as delete the data on the sensor machine to free up space.

3.1 Adjusting the scan interval

You can decide how long you want the sensor to wait in-between scans. The default setting, in the code downloaded from GitHub, is 1 second. Meaning, after completing a scan the sensor will wait one second and then start a new scan.

To adjust the scan interval, you will need to make a small change to the sensor's source code:

1) In the seaglass-master folder there is a file called "survey.py". Open this file.
2) In the file (on row 14) you should see the following text:

```
SCAN_PAUSE = 1 #    How long of a pause between scans (in
sec)
```

The number after "`SCAN_PAUSE =`" is the number of seconds the sensor will wait between scans. So changing this number to 10 would make the sensor wait 10 seconds between scans.
3) Change the number to whatever interval you want.

3.2 Switch to the seaglass-master folder

The script that runs the SeaGlass sensor is located in a folder called "seaglass-master". To be able to run that script, and thus run the sensor, you need to be in the seaglass-master folder. To switch to the seaglass-master folder, type (in the terminal):

```
cd seaglass-master
```

This change will show in the terminal through the added text "~/seaglass-master". By way of example, if you are using the Raspberry Pi's default settings, your terminal should now read:

```
pi@raspberrypi:~/seaglass-master $
```

3.3 Starting the SeaGlass sensor

The command to run the SeaGlass sensor is entered in the terminal, and from within the seaglass-master folder (see the previous step for instructions on how to change folders). In the terminal, make sure you are in the seaglass-master folder and then type:

```
./survey.py /dev/ttyUSB1
```

This command will start the sensor. After entering this command, you should see the following text:

```
<Date> <Time> ##########################
<Date> <Time> Beginning Cellular Survey.
<Date> <Time> ##########################
```

The SeaGlass sensor will now start scanning. The first scan may take a while to complete. (Don't panic.) If you get the error message "Modem time went over" you can try running the command (`./survey.py /dev/ttyUSB1`) again.

Note: You can scroll through previously entered commands using the arrow up/down keys. Press the arrow up -key once to bring up the previously entered command.

If everything is set up correctly, the scan should (eventually) start working and you will see a running text including the following:

```
<Date> <Time> Beginning Scan: <number of scan>
<Date> <Time> Collecting GPS and modem data
GPS MODE: <number>
GPS TIME: <Date> <Time>
<Date> <Time> Reading modem scan data on arfcn range:
(<number>, <number>)
```

If you get the value "None" for GPS MODE and GPS TIME, then the sensor hasn't received data from the GPS. This is often the case on the first scan. If it happens continuously, you need to start debugging. (One place to start debugging is this guide's section on hard-coding the path to the serial device.)

When you get the sensor up and running it will continue scanning for as long as you let the program continue to run.

3.3.1 A note on file sizes and a bug in MongoDB's 32-bit version

At the time of this writing, Raspbian only exists as a 32-bit version. Leading to our having to use the 32-bit version of MongoDB. (MongoDB is the database program the sensor uses to store data.) There is a bug in the 32-bit version of MongoDB that causes MongoDB to crash if the file sizes become too large (over 2 Gigabytes). To avoid this bug, you need to empty the database periodically.

3.4 Stopping the sensor

To stop the sensor from collecting additional data, press:

```
"Ctrl" + "z"
```

After pressing control + z you should see the following text:

```
^Z
[1]+  Stopped    ./survey.py /dev/ttyUSB1
```

This simply means that the sensor has stopped running. ("survey.py" is the name of the program and /dev/ttyUSB1, you may remember, is the path name to the modem.)

3.5 Generate a copy (called a "dump") of the mongo database

Now it's time to prepare the sensor data to be moved to another machine (or other storage). The first step is to create a copy, or "dump", of the data you have gathered. This is done using the command "mongodump". Note that this dump will be created in whatever directory you are when you run the mongodump command. I.e. if you are in the seaglass-master directory then that is where the dump folder will be created.

The mongodump command is run in the terminal (*not* in a MongoDB shell). Type:

```
mongodump --db SensorDB --collection Scan
```

This command will create a file called "dump" (which will contain the folder "SensorDB").

3.6 Move the file to your analysis machine/storage/etc.

Copy the dump file to your analysis machine, a USB, etc. – wherever you want to store it.

3.7 Delete old data on the Raspberry Pi to free up space

The sensor generates a lot of data. While your Raspberry Pi will eventually run out of space, a bigger issue is that 32-bit MongoDB (as noted previously) will probably crash before your computer's memory is full. Therefore, once you have created a dump of the database and moved it to your analysis machine (or however you have arranged your storage of the data), it is important to delete the data from the Raspberry Pi to free up space.

To delete the data, first connect to the mongo database using a Mongo shell. To do this in the terminal, type:

```
mongo
```

Then, once the Mongo shell has started, type:

```
use SensorDB
```

SensorDB is the name of the database in which your sensor is storing all the information it gathers. This command tells the computer you want to use that database now. Alternatively, if no such database exists, it will create a database with that name. So double-check your spelling before pressing enter. Then type:

```
db.dropDatabase()
```

This will delete the contents of the SensorDB database. (So make sure you have run the mongodump command first!) To test if the deletion worked you can use the "count" command. Type:

```
db.Scan.count()
```

This will return the number of records that are in the database (or, more specifically, in the "Scan" collection of the SensorDB database). This command should now return "`0`", i.e. that the Scan collection is empty. To exit the Mongo shell, type:

```
exit
```

NOTE: database and collection names are case sensitive. So for instance "Scan" and "scan" would be treated as two separate collections. If you're having trouble with any of the above commands, start by checking capitalization.

4 Restoring the SeaGlass sensor's MongoDB database

This section will cover how to restore the MongoDB database that was created when you ran the mongodump command. Follow these steps on whatever machine you will be using for your data analysis. This guide is written for Linux-based computers. (If you are doing these steps on the same Raspberry Pi you are using for your SeaGlass sensor, start at 4.3.)

4.1 Install the SeaGlass software

See section 2.2 of this guide.

4.2 Install MongoDB

To install MongoDB, type (in the terminal):

```
sudo apt-get install mongodb-server
```

4.3 Restore the MongoDB data using mongorestore

First, switch to the folder where you have the dump file. Change folders by typing:

```
cd <name of folder>
```

In the terminal, type mongorestore followed by the name of the file:

```
mongorestore <file name>
```

At this point, you should have the data in the mongo database on the analysis machine. Now you can decide how to proceed with the analysis. One option is to stick with MongoDB and use a program that can handle MongoDB data. Some examples of such software are Robo3T and Studio 3T, which also allow exporting the data in various formats. You may also choose to export the data to PostgreSQL, which we will cover in the following section.

5 Optional: Install PostgreSQL (and more) to ease the analysis

A database program called PostgreSQL can make analyzing the data easier. In this section we cover the steps to get Postgres up and running and connected to your data. We also cover installing a separate graphical user interface (pgAdmin 3) to make analyzing your data easier.

Note: This guide is written for how to install the software in Linux. The same software is also available for Mac and Windows but the installation procedures and commands will vary.

5.1 Install PostgreSQL, libpq-dev, posgresql-client, and postgresql-client-common

First of all, install Postgres as well as the packages libpq-dev, postgresql-client, and posgresql-client-common. To do this, open a terminal and type the following commands:

```
sudo apt-get install postgresql

sudo apt-get install libpq-dev

sudo apt-get install postgresql-client

sudo apt-get install postgresql-client-common
```

5.2 Install psycopg2

To install psycopg2, type:

```
sudo pip3 install psycopg2
```

5.3 In Postgres, create the database “sensordb” and create a password

First you want to change to a user with more rights. More specifically, you want to change to a user called “postgres”. The user postgres has full access to creating new databases in Postgres. In the terminal, type:

```
sudo su postgres
```

Now, start a Postgres shell. To do this, type (in the terminal):

```
psql
```

Now, create a database called sensordb. To do this, type CREATE DATABASE, followed by the name of the database, followed by a semicolon. As follows:

```
CREATE DATABASE sensordb;
```

Now you need to create a password for the user postgres. This is also done in the Postgres shell. (Which you have previously started when you typed `psql`). In the shell, type:

```
ALTER USER postgres WITH PASSWORD 'new_password';
```

Where you exchange `new_password` for a password of your choosing. You need to include the apostrophes on either side of the password.

5.4 Edit the postgres_config file

The postgres_config file is located in the seaglass-master folder. Which, in turn, is located wherever you extracted the SeaGlass ZIP file to. If you followed this guide, the seaglass-master folder will be located under home/pi. Click on the folder icon in the upper left-hand corner of the screen. That should take you to the folder location home/pi, where you should see the “seaglass-master” folder.

In the seaglass-master folder you should see a file called “postgres_config”. Open this file. When you open the postgres_config file, you should see the following text:

```
database=””
username=””
password=””
port=5432
hostname=”localhost”
```

You will need to edit (i.e. add some text to) the database, username, and password fields. Your edited version should read:

```
database=”sensordb”
username=”postgres”
password=”<Enter your password from step 5.3>”
port=5432
hostname=”localhost”
```

5.5 Convert the MongoDB file to Postgres

If you want to do your analysis in Postgres, you will first need to convert the data from MongoDB to Postgres. There is a program, included in the seaglass-master folder, that will do this conversion for you. This program is called mongo2postgres.py.

To convert the MongoDB data to Postgres, run the mongo2postgres.py command from where the script is located. Which, unless you have altered it, will be in the seaglass-master folder. To switch to the seaglass-master directory, type:

```
cd seaglass-master
```

And then run the conversion program by typing:

```
./mongo2postgres.py
```

At this point, all of the data stored in the Mongo database on the analysis machine should be in the Postgres database, ready to be analyzed.

5.6 Install pgAdmin 3

pgAdmin 3 is a graphical user interface aimed at making it easier to interact with your Postgres database. To install pgAdmin 3, type:

```
sudo apt-get install pgadmin3
```

5.6.1 Configure pgAdmin 3 to connect to your Postgres database

Click on “File”, then “Add server”. This will open a new window. In this window, your settings should be:

```
Name: sensordb
Port: 5432
Host: localhost
MaintenanceDB: sensordb
username: postgres
password: <whatever you decided on in step 5.3>
```

You can query (think: ask to see) the data using pgAdmin. You can enter queries in pgAdmin by clicking on the button that looks like a magnifying glass that says “SQL” (or by pressing Ctrl + E). This will open a new window. You type your queries in the top left part of the window. To execute the query, press F5, or, in the dropdown list, select “Query” and “Execute”.

This guide does not offer an in-depth tutorial for querying. However, we conclude with a query to help you get started:

```
SELECT *
FROM scan S,
gps_scan GPS,
gsm_scan GS,
gsm_measurement GM,
bcch_measurement BM
```

```
WHERE S.gsm_id = GS.id and
GS.id = GM.gsm_scan_id and
BM.gsm_measurement_id = GM.id and
S.gps_before_id = GPS.id and
GPS.mode = 3 and
BM.cell_id != 0;
```

References

Ney, P., Smith, I., Cadamuro, G., & Kohno, T. (2017). SeaGlass: Enabling City-Wide IMSI-Catcher Detection, *Proceedings on Privacy Enhancing Technologies*, *2017*(3), 39-56. doi: https://doi.org/10.1515/popets-2017-0027

Appendix 1. Some basics about Linux

Linux is an operating system. You don't need to know much about Linux to run a SeaGlass sensor. Two things that will come up often, that a new user might not be familiar with, are the terminal and the "sudo" command.

The terminal

The terminal, or "command-line" is, simply put, a way to interact with the computer using text (rather than, say, a mouse). You will be using the terminal a lot to install programs on your Raspberry Pi.

To open a terminal on a Raspberry Pi, click on the icon that looks like a monitor (black screen with a blue bar on top) with a ">_" written on it.

You can scroll through previously entered terminal commands using the arrow up/down keys. Press the arrow up -key once to bring up the previously entered command.

If you want to read more about the terminal, see e.g. Raspberry Pi's documentation: https://www.raspberrypi.org/documentation/usage/terminal/

The "Sudo" command

Linux doesn't let just anyone install programs; you commonly need to have special permission to do so. This is a security feature – think of it like needing to have a key to get into your house. Gaining permission isn't difficult and requires no extra skills: you just need to tell your computer that you have the necessary permissions to install the program. Telling your computer that you are a "super user" does this. Returning to the locked door example, telling the computer that you are a super user is like having the key to the door.

To tell the computer that you are a super user, all you have to do is start a terminal command with the abbreviation for super user, "sudo".

Note: if you have trouble installing a program, check to see if you remembered to start the install command with "sudo". Forgetting that is a common and easy way to run into problems when installing.

Appendix 2. On powering the Telit modem

The Telit modem, used in the original SeaGlass as well as in this guide, came in a casing that uses an unusual plug for its power supply. It is a plug that looks like a smaller version of an Ethernet plug. Older readers will remember these plugs from their home phones. This section covers how to power your Telit modem if it uses the same kind of plug.

One option is to find a ready-made solution online. These may not be easy (or cheap) to come by. Another option is to build one yourself. This sounds more advanced than it is – all you need to do is switch the plug on a store-bought charger. (No soldering required.)

Regardless of what option you choose, make sure you are supplying your modem with the correct amount of power. Technical information about your modem is available on its specification sheet, which can be found online (try the manufacturer's website). The Telit specification sheet notes a supply voltage range of 3.22-4.5 V DC. (Anecdotal evidence suggests it can handle considerably more voltage. However, given the cost of the modem that can be an expensive thing to test…)

If you choose to build your own power supply you will need a power supply, the correct plug, and some tools.

The Power supply

Perhaps the easiest, or cheapest, way to get the correct amount of power for your modem is to use a "switching power supply". These are inexpensive and quite easy to come by. It is a power supply just like, e.g., you have for your phone. With the exception that it has a dial that you can set to the Volt amount you want. You want a power supply with a 4.5 V setting.

The plug

You will replace the power supply's plug with the one required by the Telit modem. It uses something called a registered jack (RJ) wiring standard. Your RJ plug should have 6 small slots for wires. (RJ11, RJ14, and RJ25 are all the correct size, but I forgot which one of them I used. Sorry.) RJ plugs can be found e.g. online or in specialist electronics stores.

How-to

- Cut off the power supply's plug.
- Strip the insulation from the end of the power supply's wires. (Half an inch or so.) There is a tool for this, called a **wire stripper**, which will make your life easier.
- Twist the wires a bit to keep them together. (Both wires separately, don't connect them.)
- Insert the wires into the new RJ plug.
    - With the prong (the little plastic bit that sticks out) facing towards you, insert the negative wire to the extreme left and the positive wire to the extreme right.
    - (Check the power supply documentation to see which wire is positive and which is negative. If you can't find that information, you can use a **multimeter** to test polarity. There are several beginners' guides online for multimeter use.)
- Now attach the plug to the wire. To do this, you will need a tool called a **crimper**. (Many crimpers come with a wire stripping function as well.) There are crimpers for different kinds of plugs, so make sure you get the right kind, i.e. for RJ11, RJ14, and RJ25 plugs.

Note: If you feel uncertain about these steps, check to see if you have a Makerspace or amateur radio club nearby. Such groups might well be willing to help.