Hardware setup:

1. Powerbank connected to Pi4 for power
2. Hack RF connected via the USB interface
3. SD card with rasbian (headerless)
4. Connect momentary switch to GND pin on the Pi4 (E.g pin 6)
5. Connect the other side of the momentary switch to a GPIO pin on the Pi4 (e.g pin 7)

Software setup:

1. Dependencies installed for hackrf
2. **Create the bash script:**

Its up to you if you want to make a directory that stores all of your scripts for this project..

For example you can make a directory called ‘scripts’:

*mkdir scripts*

You can then navigate to that directory by typing:

*cd scripts*

Now that you are in the directory ‘scripts’ you can use ‘nano’ text editor to create the bash script:

*nano call\_hackrf\_gps\_jamming\_script.sh*

1. Add the following lines of code to the script:

|  |
| --- |
| #!/bin/bash  # Change directory to where the HackRF GPS jamming script is located  cd /path/to/hackrf\_gps\_jamming\_script/  # Call the HackRF GPS jamming script  ./hackrf\_gps\_jamming\_script.sh |

Note that you need to change the path: /path/to/hack\_rf\_jamming\_script/ to the actual path where you have placed the python script. Honestly id recommend just putting all of your scripts in this one folder.

1. Now make the script executable:

*chmod +x call\_hackrf\_gps\_jamming\_script.sh*

1. You can test the functionality of the script by typing in the terminal:

*./call\_hackrf\_gps\_jamming\_script.sh*

Create the python script that detects the momentary button press and executes the bash script:

1. Install gpiozero package

*Sudo apt-get install python3-gpiozero*

1. Write the python script that listens on pin 7 for when the momentary switch is pressed.

This program will be called button\_script.py:

|  |
| --- |
| *from gpiozero import Button*  *from subprocess import call*  # Set the GPIO pin number  *button\_pin = 7*  # Create a Button object  *button = Button(button\_pin)*  # Define the function to be executed when the button is pressed  *def button\_pressed():*  *call(["./call\_hackrf\_gps\_jamming\_script.sh"])*  # When the button is pressed, call the button\_pressed() function  *button.when\_pressed = button\_pressed* |

This script will run in the background and detect when the momentary switch has been pressed. When the button is pressed will execute the bash script that you have written.

1. Now lets make the button\_script.py program execute on boot:
2. Create a file called ‘button\_script.service’ in the /etc/system/system directory. Add the following lines of scode to the file:

|  |
| --- |
| *[Unit]*  *Description=Button script*  *[Service]*  *ExecStart=/usr/bin/python3 /path/to/button\_script.py*  *Restart=always*  *User=pi*  *[Install]*  *WantedBy=multi-user.target* |

Replace ‘/path/to/button\_script.py’ with the actual path to where you created your python script.

If you’ve made a username other than ‘pi’ for your RPI4. Make sure you change it to your actual user name.

1. Run the following command to enable the system service:

*sudo systemctl enable button\_script.service*

This will allow the system service to start automatically on boot.

1. Start the system service:

*sudo systemctl start button\_script.service*

This will start the script running immediately.

1. Check the status of the system service

*sudo systemctl status button\_script.service*

Note: You can remote into the pi at any time and run the command at step 8. It will tell you if the service is running or not and what the status is.

You only need to use the command at step 7 if you want to manually start the service yourself. (for what ever reason the service isn’t running.