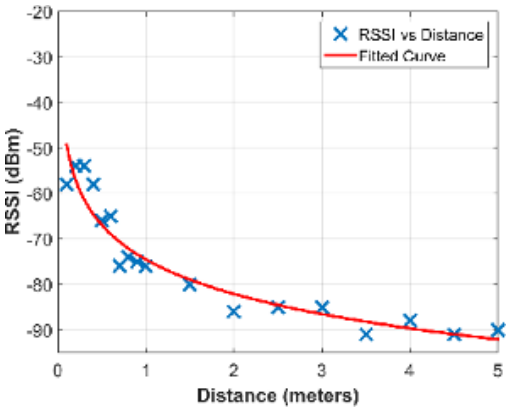
Lab 2 (Part 2)

Finding Hidden Spying Cameras with RSSI

By the end of this lab, you will collect RSSI measurements from your Raspberry Pi to predict the location of the hidden camera.

Background



The hidden camera could send packets to a server whenever a motion is detected. We can capture these packets and localize hidden cameras by analyzing the packet signal strength. Each packet has a **RSSI value** field that represents the received signal strengths. The further away from the hidden camera, the weaker the signal, the lower the RSSI value.

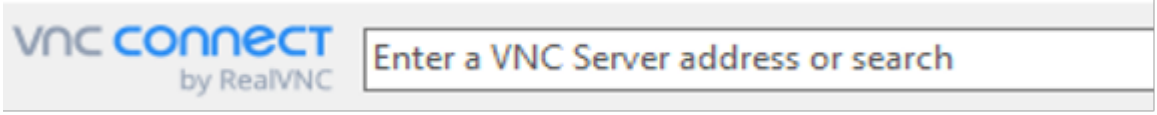
In this lab, you will learn:
(1) how to capture WiFi packets from a wireless device (e.g. a spy camera) using Wireshark and Python.
(2) how to identify and extract RSSI from captured packets.
(3) how RSSI values correlate with distance and other dynamics of the environment.
(4) how to localize a hidden camera by analyzing RSSI trends.

Components Required

- Raspberry Pi + WiFi dongle
- **Power Bank**
- Desktop Computer

Setting up the Raspberry Pi:

1. Make sure TP-Link network adapter is attached to your rpi such as the picture below. You don't need the camera for this lab.
2. **Power your Raspberry Pi with the battery bank.**
3. **Connect your RPi to Illinois_Guest network.**
4. Connect to your Raspberry Pi remotely:
    a. Use the VNC installed on the lab machines.
    b. Put the IP address on LED matrix into the search bar



c. and hit enter (you may need credentials you set in Prelab 1)

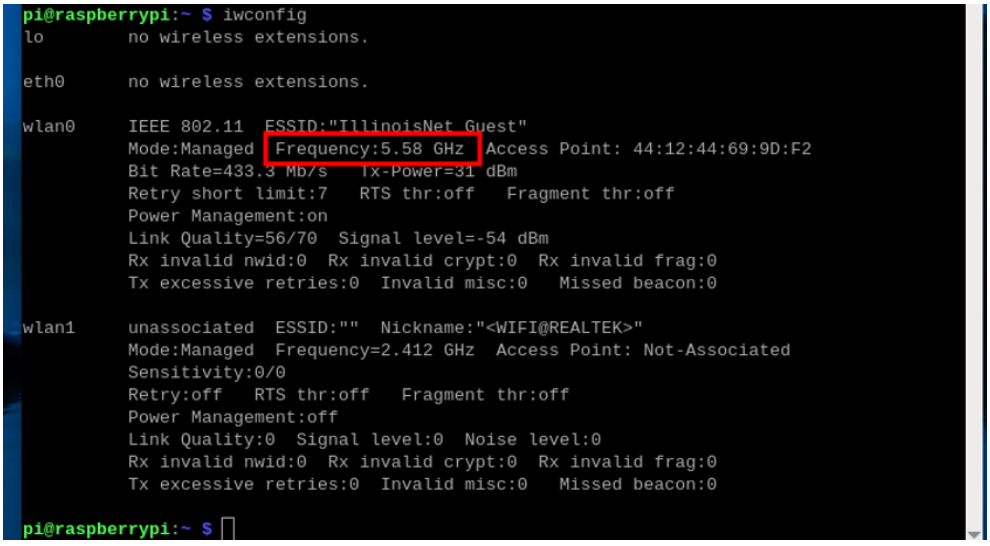
d. If you have any trouble connecting to WiFi or connecting to VNC Viewer, **call a TA for help immediately.**

5. Download lab scripts from Canvas on your Pi.

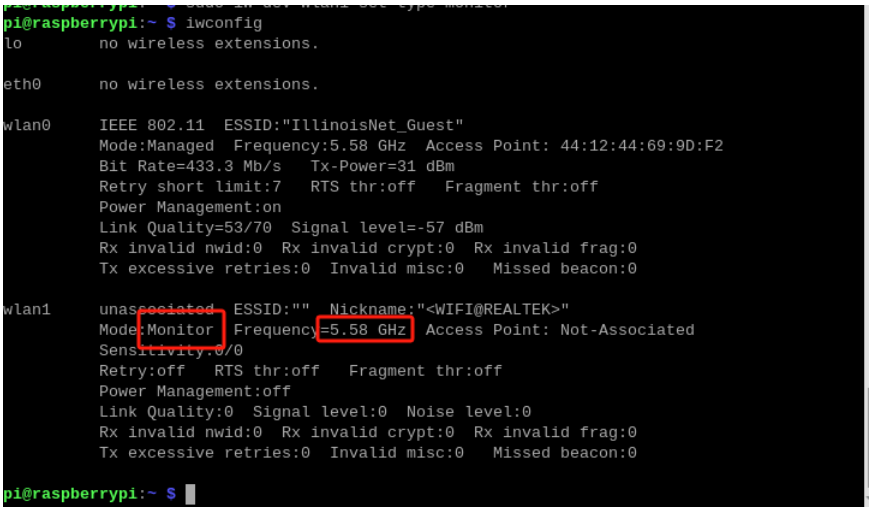
Exercise 1: Configuring the WiFi Dongle to Monitor Mode:

WiFi dongle needs to work in **monitor mode** to capture WiFi packets. Monitor mode allows you to capture 802.11 radio packets in the air.

1. Download monitor_mode.sh from Canvas to your RPI.
2. Run **iwconfig** to obtain the frequency of wlan0.



3. Modify the last two parameters of the last line: “sudo iw wlan1 set freq 5580 80MHz”
   a. **5580:** 5580MHz (5.58GHz). Your RPI is connected to the 5.58GHz channel of the 5GHz WiFi band. **Change this field to the frequency that iwconfig tells you.**
   b. **80MHz:** if wlan0 frequency is in 5GHz, do not modify it. If wlan0 frequency is in 2.4GHz, remove this part. z
4. Run by: **sudo bash monitor_mode.sh**. Rerun the command **sudo bash monitor_mode.sh** if you see any errors.
5. Run **iwconfig** and look for wlan1 to verify if your **WiFi dongle is in monitor mode.**



Double check that **frequency** changes correctly and wlan1 is in monitor mode. If **wlan1 is still in managed mode**, make sure wlan1 is not connected to any WiFi, and run **sudo iw dev wlan1 set type monitor**. Also disconnect wlan1 from any WiFi. If **wlan1's frequency is incorrect,** run **cat monitor_mode.sh** to view the content of monitor_mode.sh, and run the commands line-by-line. If still not working, call a TA.

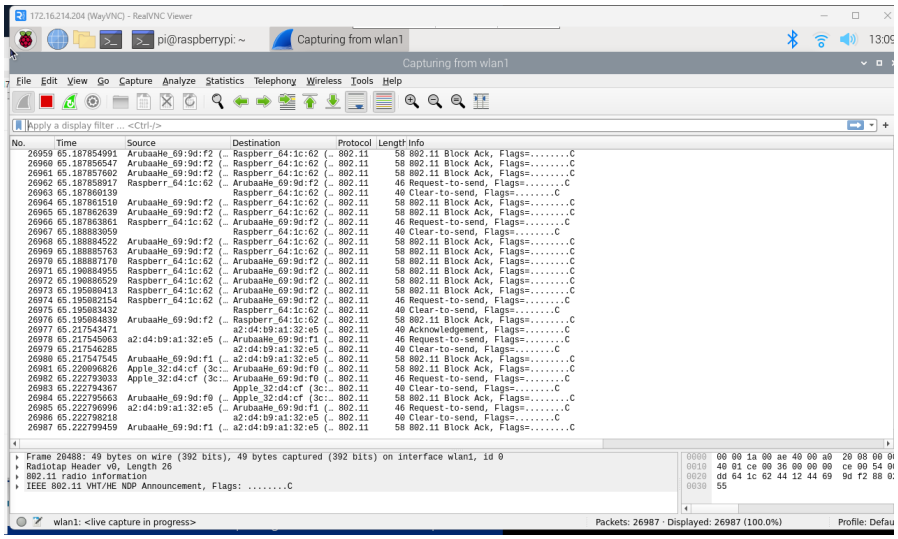
6. Remember to run **monitor_mode.sh** every time your RPI restarts and verify by **iwconfig**.

From now on, the RPI's built-in WiFi card (**wlan0**) will be in default **managed mode** and connected to WiFi for regular internet connection. The WiFi Dongle (**wlan1**) operates in **monitor mode** for 802.11 packet sniffing.

Exercise 2: Capturing 802.11 packets in Wireshark

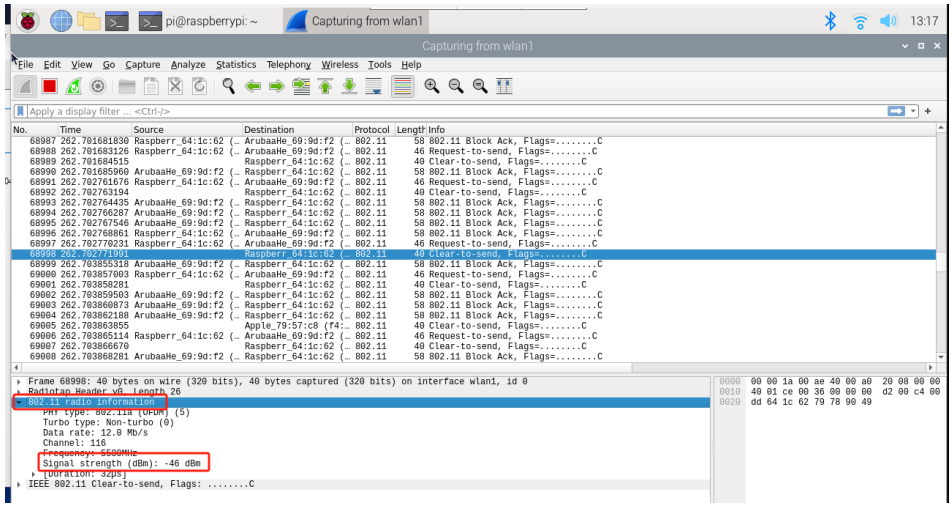
You should have installed Wireshark on RPI in the prelab.

1. Open Wireshark on RPI by: **sudo wireshark**. Select the **wlan1** interface.
2. You would see lots of 802.11 packets. The wireshark should look like this:



These packets are WiFi protocol messages (CSMA/CA).

3. Click any 802.11 packet. Expand the **802.11 radio information** tab on the bottom. You will see **Signal strength (dbm).**



Checkpoint 1 (3 points):

Filter 802.11 packets sent from your RPI wlan0. Run **ifconfig** to obtain MAC address of **wlan0**. Use Wireshark filter to filter packets with source address equals to wlan0 MAC address. Hint: this cheatsheet. Demonstrate 802.11 packets originated from your wlan0 in Wireshark. Show the RSSI value of an 802.11 packet. If no packet packets from your wlan0, try to use the browser to create some traffic.

Exercise 3: RSSI Collection

The course staff has set up three hidden cameras around the lab. The MAC address of each device is printed on it. Each RPI is constantly broadcasting packets every 0.2 – 0.25 seconds. You will collect RSSI measurements from one of the hidden cameras. The hidden cameras are on **Illinois_Guest WiFi**, and its frequency is printed on the SenseHat.

Remember to use a power bank to power your PI. This is important because you will be moving your Pi around the room.

Follow steps below to capture 802.11 packets and collect RSSI measurements.

1. Pick one of the three hidden cameras and write down its MAC address and frequency.
2. **On your battery-powered Pi,** modify the last line of the **monitor_mode.sh** script to "**sudo iw wlan1 set freq [freq] [width]"** to switch **wlan1** interface to **the frequency shown on your assigned spy camera in the room**. If frequency is in 5GHz range, width = **80MHz**; if frequency is in 2.4GHz range, ignore this field. Again, run **iwconfig** to verify if the wlan1 frequency is correct and if wlan1 is in monitor mode.

3. Download collect_rssi.py from Canvas. This script uses Scapy to capture packets from your Pi's wlan1 interface. The **AsyncSniffer** object constantly monitors for packets received by the wlan1 interface, and the **captured_packet_callback** function runs whenever a new packet is captured.
4. Implement captured_packet_callback function. This function should:
    a. Filter 802.11 packets **originating** from your assigned spy camera (by filtering packets' **source MAC address**). **Important:** make sure to understand which is the source address which is the destination address.
    b. Write **RSSI values and timestamps** of each filter packet to a CSV file.
5. **Leave RPI still at your desk.** Run collect_rssi.py using sudo: "**sudo python3 collect_rssi.py"** to collect RSSI measurements.
6. Check if the code generates a CSV file with **(timestamps, RSSI values)**.
7. **Plot a RSSI vs Time graph**, with RSSI in y-axis and Time in x-axis. Set the x and y labels and the title. **Save this plot for a checkpoint.**
   Note: Follow this tutorial for reading a CSV file, and this tutorial for line plot. Ask a TA if you get confused.

Checkpoint 2 (4 points):

Manipulate the environment around the RPI while collecting RSSI values. You may try to block the WiFi dongle with your hand or your phone. Run **collect_rssi.py** again to collect RSSI measurements while you interfere with the environment. **Plot another RSSI vs Time graph, illustrating the interference. Call TAs to demonstrate the RSSI vs Time graphs. Be ready to** explain the observed RSSI readings in terms of interference.

Exercise 4: RSSI Localization

A general characteristic of RSSI is: **RSSI increases as you move closer to the transmitter**. You can leverage this to get a rough estimate of the hidden camera's location.

In this experiment, you will move around the room and observe the change in RSSI value. Specifically, your team will take the RPi and walk around the room. You will observe that the RSSI value is higher when you walk closer to the camera.

1. Setup: one person walks around the room with RPi + power bank, one person runs the collect_rssi.py script remotely.
2. **Run collect_rssi.py script**. Make sure the script has started collecting RSSI data before walking.
3. **Walk around the lab** and finish a full circle in 30 seconds. **Walk pass the hidden camera** that you are sniffing**. Always point the Wi-Fi dongle towards the spying camera.**
4. **Rename the RSSI measurements CSV file.** You will use them in the post lab.

Checkpoint 3 (3 points):

Plot RSSI vs. Time graph. Can you observe a trend? Does RSSI increase as you move closer to the hidden camera? Cal TAs to demonstrate the graph.

Post-lab 2 Assignments (Due March 20):

Create a repository on GitLab hosted by the Engineering GitLab. After creating your repository, please add the following NetIDs as members with *Reporter* access:

- elahe
- dwivedi6
- bkao2
- hanbog2
- pbalaji3

Include names and netid for everyone in your group in your submission and description of each assignment in the ReadMe file. Submit the repo link on canvas. **You must include plots in the repo.**
Make sure your repository is set to *private*.

Post Lab Assignment-1 (3 points): Develop a program that outputs timestamps of a sliding window during which you walk pass the hidden camera. This program takes the (timestamp, RSSI values) file and outputs two timestamps. If you walk pass the hidden camera, you would see an increasing and then decreasing trend of RSSI values. This program should identify the peak.

Note: Using the maximum RSSI value is one way, but what if there is a random high-RSSI outlier?

Deliverable:

1. Plot the RSSI vs Time graph, and highlight the time window of the peak.
2. The RSSI measurements CSV file.
3. Code to generate the above RSSI vs Time graph.

Post Lab Assignment-2 (3 points): Implement record_joystick function which write the timestamp and key pressed to a (csv) file whenever you press the Joystick. Redo RSSI Collection above. While walking around the lab, press the joystick to record the true label when you are closest to the hidden camera.

Demonstrate the RSSI vs Time graph and plot the true label. Show that RSSI peaks correspond to the recorded Joystick timestamp.

Post Lab Assignment-3 (4 points): Develop a program to use the Joystick to record your steps and orientation when walking. For example, press the joystick for every step, and press the joystick to record orientation of each turn. This program should output a (timestamp, key pressed) CSV file. While walking, also collect the RSSI values of a camera in the room and save timestamp and RSSI values. Use the timestamps to match the RSSI and joystick data.

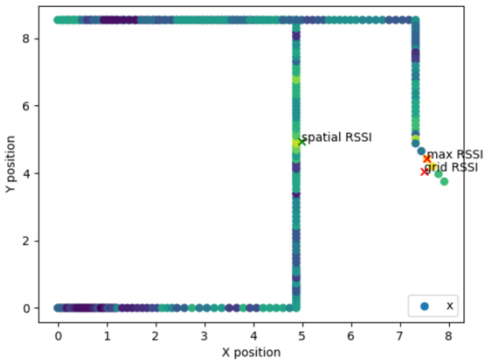
Draw a scatter plot of your walking path with this (timestamp, key pressed) CSV file. You can assume you have fixed step sizes, and your path starts at (0, 0). Make sure your plot has the same turns as your actual walking path. The walking distance between each turn could be rough as long as the overall plot looks legitimate. Finally, color code dots on the scatter plot with the corresponding RSSI values. Your completed plot is a scatter plot of your walking path with RSSI values as color code.

Deliverable:

1. The Joystick (timestamp, key pressed) CSV file.
2. The RSSI (timestamp, RSSI) CSV file.
3. A video of your walking path.
4. A scatter plot of **every step**, with RSSI values as color code.
5. Code to collect and generate the (timestamp, key pressed, RSSI) CSV file and plotting.

Post Lab Assignment-4: Write in a few sentences what you liked about this lab and what you didn't like about this lab.

Extra Credit Assignment (3 points): Instead of plotting each step in Post Lab Assignment 3, plot every RSSI measurements along the walking path by interpolating the location where the RSSI is captured. Your plot would look like the following:



Extra Credit Assignment (3 points): Hidden camera detectors require real-time RSSI detection capabilities. Develop a program that uses the SenseHat as a RSSI meter. This program should change the displayed color, or the number of displayed pixels, to represent the instantaneous (or moving average) RSSI value. Submit a video of your RSSI-meter in action with the source code. The video should demonstrate changes in color and/or number of pixels as you walk closer to the hidden camera.