Treasure Hunt - RF edition

Project for Advanced Topics in Computer Networks

Team B

Dallas Dimitris 2412

Maroulis Manos 2465

----

1. Introduction

1.1 Project presentation

Our components for this project are an autonomous ground vehicle and two Software Defined Radios (SDR). The purpose of this project is to create an application on the vehicle that will try to find the Transmitter in the room, based on the Received Signal Strength Indication (RSSI), and then present a spectrum heatmap based on the recorded data.

1.2 Equipment

2x ADALM-PLUTO SDR

2x Raspberry Pi

Unmanned ground Rover

RP-LiDAR A1-M8

Handmade directional antenna

2. Tools

2.1 ROS

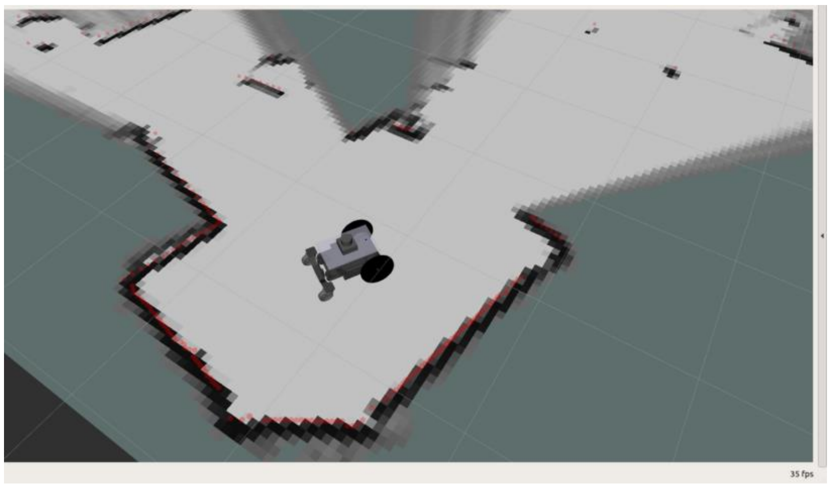
2.1.1 What is ROS

Robot Operating System (ROS) is an open-source meta-operating system. It provides the same services as a normal operating system does, including hardware control, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. In this project we use ROS to navigate a robot autonomously.

2.1.2 Mapping procedure

In order to create a map, we need a camera or a range finder sensor. In this project we use RP-LiDAR A1-M8, a 360^{o} laser sensor which identifies objects in the environment.

We use a SLAM (Simultaneous localization and mapping) algorithm in order to create a map from the sensor data and to position our robot on the map.



2.1.3 Coordinates and goal

In the map created in section 2.1.2 we have a set of cartesian coordinates (x, y) and angular coordinates (w, z). When we specify a goal on the map, the navigation procedure tries to move the robot to those coordinates with some error tolerance. If the goal coordinates happen to overlap with an object, we must specify a recovery behavior for the robot so as not to stuck in a place.

2.1.4 Recovery behavior

Recovery behavior is a sequence of moves the robot has to perform in order to unstuck and be able to move to a normal goal again.

2.2 ADALM-PLUTO SDR

2.2.1 What is SDR and RSSI

A radio that uses software to perform signal-processing tasks that were traditionally performed by hardware.

Received signal strength indicator (RSSI) is a measurement of the power present in a received radio signal.

The PlutoSDR is using by default the AD9363 RF chip with RF coverage from 325 MHz to 3.8 GHz. By manually choosing to use the built-in AD9364 RF chip, we are able to tune up to 6 GHz and down to 70 MHz, a range that better serves our experiments.

2.2.2 Python APIs for libiio

The library libiio is used to interface to the Linux Industrial Input/Output (IIO) Subsystem. The module pyadi-iio, provides device-specific APIs built on top of the current libiio python bindings and is used mainly on the transmitting SDR with Python3 only. We also use the Radio System Design Ltd python package because it is compatible with the robot Python2 environment. These interfaces try to match the driver naming as much as possible without the need to understand the complexities of libiio and IIO.

3. Implementation

3.1 Transmitter SDR

3.1.1 TX properties

We conducted experiments by transmitting on different central frequencies and we ended up using the 5000Mhz (5GHz). We are setting the bandwidth to 5MHz and the TX hardware Gain to –5 to achieve a transmission more efficient for our purpose.

3.1.2 Transmitting Data

We are transmitting using the DMA mode, so that we can send data via the TX buffer. Data to be transmitted can be as simple as 10000 numbers, but our approach was to create some IQ data resulting to a complex sinusoid. Then the task is as simple as repeatedly writing the IQ data in the TX buffer.

3.2 Receiver SDR

3.2.1 RX properties

Trying to get optimal conditions for our experiment, the SDR receiving signal has to be set at the same central frequency as the transmission, meaning 5000MHz (5GHz). Bandwidth is also the same interval as the transmitting signal. Additionally, TX state is configured "OFF" to avoid self-reception.

3.2.2 Stabilizing Measurements

Continuously measuring RSSI may be chaotic because it is highly unstable. We developed a function to get over this obstacle:

1) Take a number of measurements over a double nested loop and store them in a list
2) For each outer loop, clear the bias of the measurements by rejecting values over and under two thresholds
3) For the same loop, compute the mean RSSI of the cleared list and store it in a list

4) Clear again the list of means from bias and return their mean.

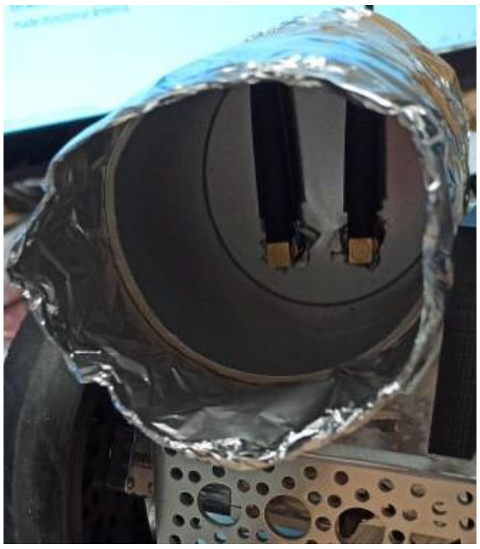
In that way, measuring the RSSI becomes much more stable giving only a 0.5 class deviation for a fixed point.

3.3 Algorithm

3.3.1 Directional Antenna

Robot navigation is driven by the SDR RSSI measurements

Having the SDR antenna exposed made it difficult to distinguish a difference between the measurements around the room. We wanted to determine which way the robot should consider forward. Our solution was a handmade directional antenna with an aluminum–plastic structure.



3.3.2 Algorithm walkthrough

First, we start the SDR transmitter and receiver in the specified frequency described above.

1) The robot performs a 360^o turn around itself taking measurements every 30^o degrees, keeps the best measurement and pose (x,y and w,z)
2) if the RSSI measurement is below a threshold then the goal is reached.
3) measures the probability of having an obstacle some meters or centimeters ahead of these coordinates. If the probability is low it drives to the new specified goal and repeats (a.).
4) Frequently the map is not clear due to sensor errors or harsh environments and the vehicle's goal overlaps an object. We prepared a

recovery behavior for our robot so as to unstuck it from these situations. See the section below.

3.3.3 Custom recovery

The algorithm has a trigger function in order to recover the robot from difficult situations or when it is unsure about its goal.

1) The function commands the robot to do a 360^o circle around itself and clear the obstacles around it in a parameterized radius.
2) Continue driving in a 360^o circle and re-calculate probabilities of having obstacles in front of you on the map, pick the lowest value and drive forward
3) Return to your normal procedure

3.4 Spectrum Heatmap

3.4.1 Dataset

Creating a heatmap without having a complete 2-dimensional matrix for map intensity required a complicated approach using a scatter plot.

For creating the scatter data set we simply create some extra, “virtual” points where the RSSI was supposedly measured. Those points are spaced apart by a random distance, proportional to a desirable scattering factor, but stay near their corresponding origin point. Basically, the more the RSSI measured in a certain “true” point, the more the “virtual” points created around it.

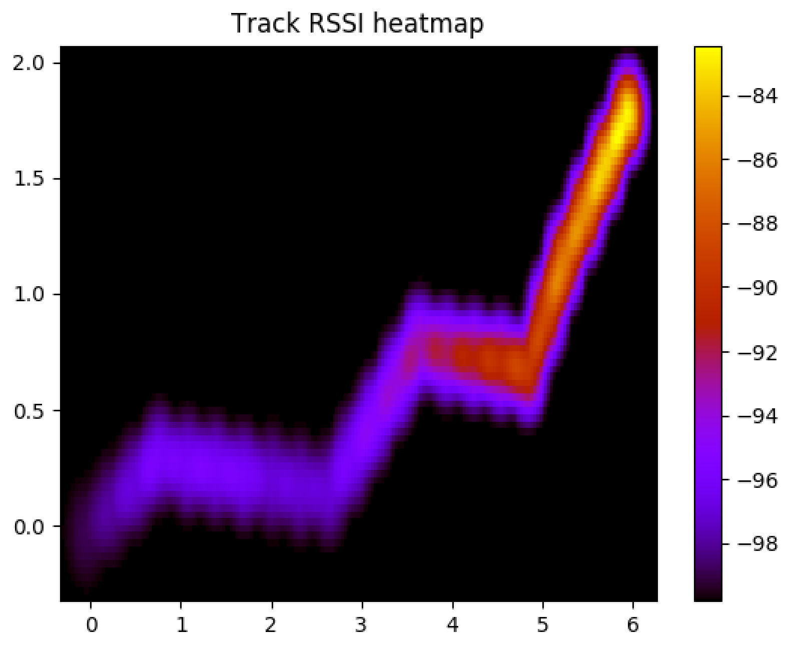
We also took care of the case that there are some “gaps” where the robot did not take a measurement. We simply fill in the gaps, if needed, with equally spaced points and predict the RSSI that would have been recorded.

3.4.2 Vehicle Route Heatmap

This whole procedure described above is just to be used by the heatmap algorithm which utilizes the quartic Kernel Density Estimation (KDE) function. So basically, we have a higher color intensity value wherever the point density is higher, which translates to higher RSSI value.

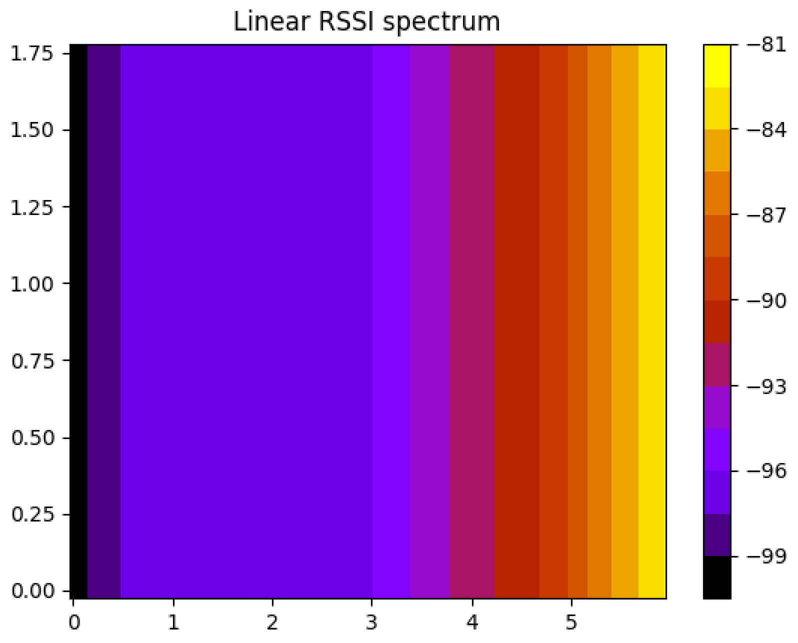
The algorithm also calculates dynamically the radius of each point and the grid size based on distance travelled by the robot. Finally, it dynamically regulates the color bar scale from the KDE values to the RSSI scale.

In the example below, the robot starts at (0,0) with an initial RSSI equal to -99 and reaches the goal which is near (6,2) with an RSSI equal to –82. In the black area the RSSI is not necessarily below -99 but it is unknown to the robot.



3.4.3 Linear Spectrum

Linear spectrum is just a simple way to visualize the RSSI measurements as if the robot was moving linearly. For this task, all we do is creating an all-ones matrix, multiplying it with the 1-D array of RSSI measurements and the drawing filled contours.


Linear RSSI spectrum