SDR Based Robotic Navigation: An Update

Arjun Vedantham March 13, 2024

Outline

1. SDR Background
2. Previous Presentation Summary
3. RF Theory
4. Implementing DOA with RTL-SDRs
5. ROS2/ROS1 Simulation
6. Evaluation + Path Forward

What are SDRs?

- PySDR: "A radio that uses software to perform signal processing tasks that were traditionally performed in hardware"
- Want to take discrete hardware components -> move implementations into software for modularity + reduced costs
- Off the shelf SDRs can capture wide radio bands!

RTL-SDR (L), PlutoSDR (R)

Navigation as a Problem Space

- Traditionally we use GPS for location-finding (1.17 GHz)
  - Imprecise, prone to interference
  - "Signals of opportunity" (SOP)?
- WiFi SSIDs (2.4/5 GHz)
  - Wigle map
- Cellular/LTE
- Can we capitalize on ambient SOP to aid with robotic navigation?
  - Cool talk on this subject given on Monday: longer-range RFID localization with BladeRF, with a depth camera and manipulator task

Prior Research + Literature Review

- Exploiting LTE Signals for Navigation (2018)
  - Exploits elements of the LTE protocol (primary synchronization signal, cell reference signals)
  - Eventually obtained 8.15 m RMSE
  - Not easily replicable
    - Extremely complicated software stack (also nothing open source?)
    - Custom hardware

Prior Research + Literature Review

- Direction of Arrival Estimation Analysis in GNURadio (2017)
  - Uses two RTL-SDRs with a shared clock
  - "Eigenvector method" - sweep over the range of angles and pick the one that corresponds with the highest signal strength
  - Provides GNURadio flowgraphs, but not easily replicable
    - Uses off the shelf hardware, flowgraphs are generally easy to use
    - Custom code is WORN (write once, read never)

Signal + Direction of Arrival (DOA) Theory

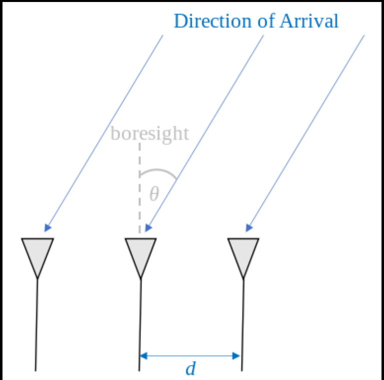
- SDRs capture signals in "IQ" format
- Uses quadrature sampling (two signals that are 90° out of phase)
  - "I": in phase
  - "Q": quadrature
- Stored as a complex number: real component stores in phase component, imaginary component stores the quadrature component
- Allows us to disentangle the carrier signal from the phase modulation

Signal + Direction of Arrival (DOA) Theory

- Main equation at play: $s[n]e^{-2j\pi f_c \Delta t} \rightarrow s[n]e^{-2j\pi dksin(\theta)}$
- Variables:
  - $s[n]$ - discrete (nth) sample of the signal
  - f_c - center frequency (in this case 433 MHz)
  - Δt - the time difference of arrival between when the signal hits the first vs. second antennas: $\Delta t = \frac{dsin(\theta)}{c}$
    - This is why phase coherence is so important
    - Want to represent d in the above definition as a fraction of the signal wavelength
    - d gets scaled by k, the number of antenna elements in the phased array (in this case there's only two, so $k = 2$)
- Euler's identity used to decompose the IQ signal representation

UNIVERSITY OF MARYLAND

Signal + Direction of Arrival (DOA) Theory

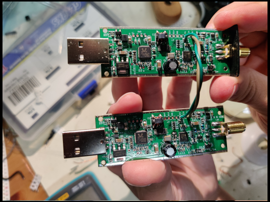


From PySDR

UNIVERSITY OF MARYLAND

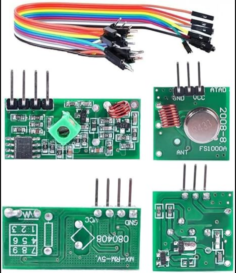
Hardware Implementation

- Bridge the clocks together by jumping the clock and ground pins
- Mounted onto a metal heat sink using a thermal pad
- Change the read-only memory to allow for daisy chained radios on a separate USB line

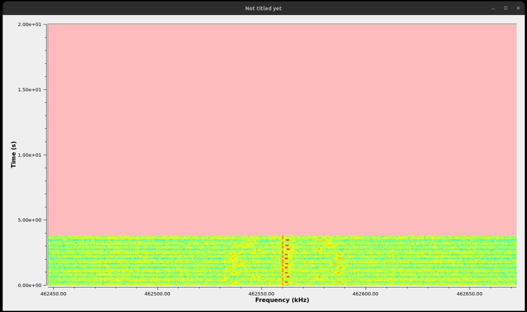


Hardware Implementation

- Off the shelf, 433 MHz transmitters
- Frequency locked - use pulsewidth modulation to encode messages
- Also used walkie talkies (467 MHz, frequency modulated)



Hardware Implementation



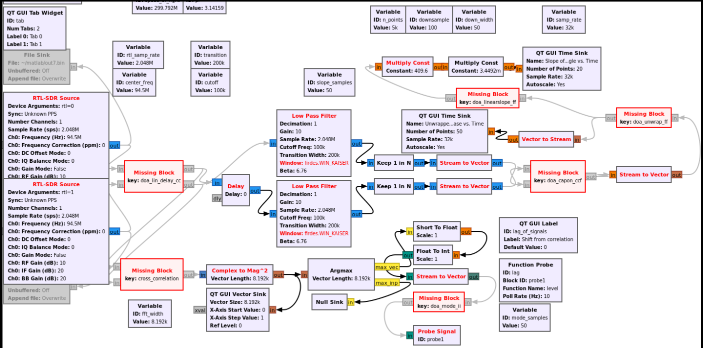
GNURadio

- Framework for dealing with SDR data
- Allows you to string together graphical blocks to form "flowgraphs" that define the data processing pipeline
- Generates Python that is actually executed by the underlying system
  - Code behind graphical blocks is implemented in C++/Python and is dynamically linked at runtime

Software Implementation

- As described in the original talk/slide deck, the original GNURadio flowgraph was heavily reliant on out-of-tree (OOT) modules
  - These are modules outside of GNURadio's standard library of blocks
  - GNURadio provides the runtime services and allows end-users to interact with a block
  - Actual processing happens in the block's C++ or Python implementation
- Building + upgrading GNURadio modules relies on a complex, difficult to use C++ build system and API
  - Example: CMake refused to generate build files because a source file included the SHA-256 hash of a file that had been changed on disk

Software Implementation



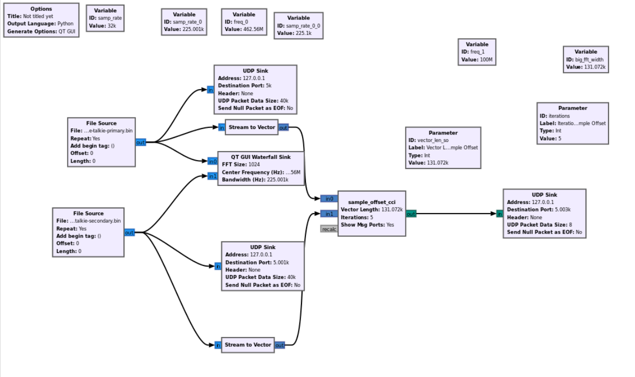
Every block in red is broken!

UNIVERSITY OF MARYLAND

Software Implementation

- Simplify flowgraph
  - Ensure phase coherence - want the only phase difference to be a result of seeing the signal arrive at different times, not due to timing/clock drift issues
    - Sampling using the same clock means phase offset should be fixed from the point of sampling
    - Still have a constant phase difference - need to calculate sampling Δt and use that to align the sample captures
  - Hand off the remaining processing tasks to an easier to use platform

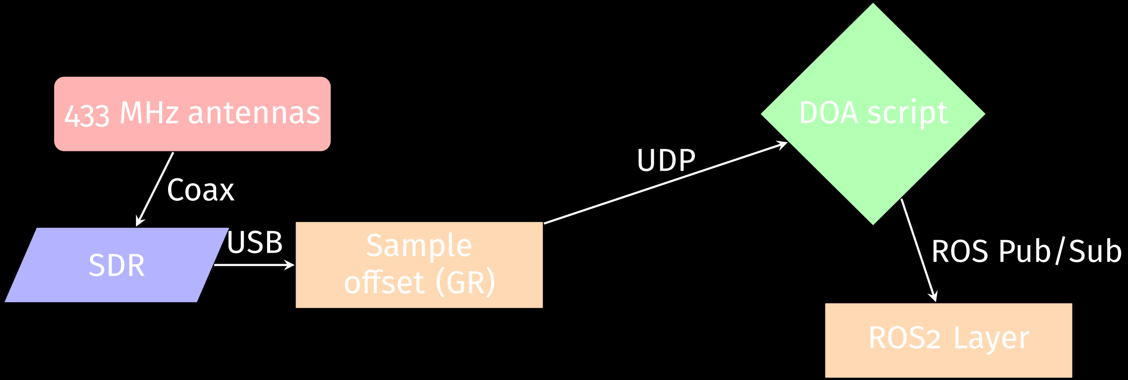
Software Implementation



Software Implementation

- Unpack UDP packets (40000 samples at once, sampling at 225KHz)
- Validate that both captures are the same size (same # of samples)
- Reshape + vertically stack in Numpy (call this matrix S, where the first row contains the IQ samples from the first SDR/antenna)
- For a sample θ_i (sweep from -180° to 180°):
  - Calculate the array factor matrix: $A = e^{-2j*\pi*d*k_i*sin(\theta_i)}$
  - Take the conjugate transpose of the array factor A and multiply against the signal capture matrix: $A^H S$
  - Add to our list of angle candidates, use argmax on the output list to find the angle corresponding to the "loudest" signal

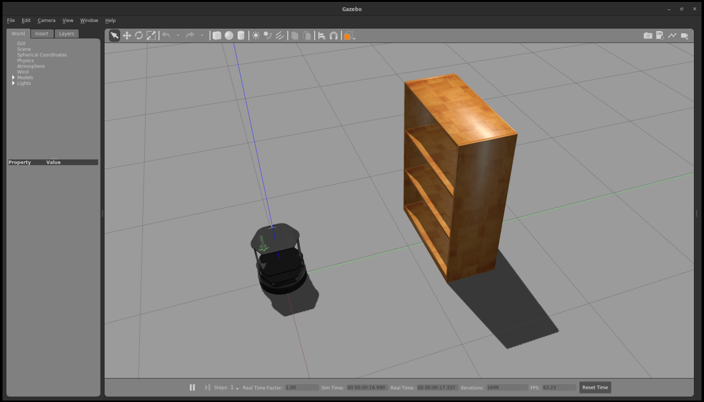
Overall Software Structure



Deployment in ROS2 + ROS1

- Want to use this guidance technique with a Turtlebot 2
- Uses ROS1 - modern systems (including my own) use ROS2!
  - Incorporate DOA analysis into a ROS publisher/subscriber topic
  - Serve the resulting angle calculation through a common topic on the ROS2 daemon
  - Have the ROS1 environment running in a Docker container
    - Use the ROS1 bridge
  - Driver for the Turtlebot
    - Subscribe to the angle topic (just called "topic" in this case)
    - Turn the robot in that direction
- I would demo this, but unfortunately my ROS1 bridge is broken :/

Deployment in ROS2 + ROS1



UNIVERSITY OF MARYLAND

Results to Date

- Unfortunately, not great.
- Not fully certain that samples are phase coherent at this point
- Calculated angle varies wildly, and frequently jumps to/from 180° or -180°, likely due to the angle correlation being so weak that the argmax just returns the default values

Future Work

- Currently working on:
  - Better visualization in RViz/Gazebo
  - Simulated radio data to decouple robotic simulation work from signal processing work
  - Improve angle calculations by reimplementing more of the signal cleanup techniques from the original flowgraph
  - (More) extensively evaluating the signal captures to check for phase coherence, like in the original presentation
- Move to a real robotics platform!

Future Work

- Fix software fragmentation: use a newer robot platform? Removes ROS1 dependency
- CMSC838L: A functional domain specific language for FPGA accelerated signal processing
- Hardware modifications?
  - Real SDR-based phased antenna arrays do not use just two RTL-SDRs - use ones with better clocks, sampling bandwidth
  - Better to have more samples - CPU throughput to handle the samples becomes a problem
- Other DOA algorithms? MUSIC?

References

- Exploiting LTE Signals for Navigation: Theory to Implementation
- Direction of Arrival Analysis on a Mobile Platform
- PySDR