# Snoofer Bot: A Biologically Inspired Autonomous Mapping Robot

Aiden Frank

## Project Summary

Mapping a new environment is crucial to operating effectively in it. Some environments, including other planets, are dangerous to humans, and using robots to map them would eliminate the risk to human life. Current space rovers are only partially autonomous; they require instructions about where to go. A fully autonomous rover could operate more efficiently and without radio control. I therefore built Snoofer Bot, a robot which can autonomously map an environment. Its design was inspired by animals which have already evolved to be effective at mapping. The robot has a swiveling sensor, just as an animal moves its head. It has whiskers to feel objects, as do many mammals. And it periodically returns to a “home” location to recenter itself, just as animals do. Snoofer Bot can map dozens of points with high accuracy. It therefore demonstrates technologies which could be used for robotic exploration.

# Introduction

## Project Origin

- I was thinking about rovers on other worlds.
- Radio signals take many minutes to travel between the rover and Earth.
- A rover in an underground area might have no radio connection at all.
- Current rovers rely on human controllers to direct their movements.
- I decided to built a robot which could map an environment autonomously.

## Work by Others

### Simultaneous Localization and Mapping

SLAM “is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce it’s location.”[sic] (1)  Without SLAM, a moving robot would become lost and its maps would become useless.  I therefore developed SLAM capability for Snoofer Bot.

### Active Sensing

“Active sensing” usually refers to one of two types of systems: homeoactive and alloactive.  A homeoactive system changes the state of the environment in order to observe it (as a bat does by sending echolocating pulses), while an alloactive system changes a sensor’s relation with the environment (as does a human looking around a room to observe the entire space). (2)  Active sensing of both types is a powerful tool for collecting data.  Snoofer Bot’s LiDAR distance sensor does both: it sends out light to detect the distance to objects, and swivels to collect more data.  This swiveling configuration was inspired by animals, which similarly move their heads to collect more information.

# Introduction (continued)

## Previous Work

Not Applicable

## Engineering Goal

My overall goal was to construct a robot which could autonomously create a map of an environment.

In order to do this, the robot needed the following capabilities:

- Move around without colliding with objects
- Accurately track its location as it moves
- Determine where objects are relative to self
- Combine relative object location with self location to map object
- Transmit data to a nearby computer

This is a worthwhile goal because building and programming a robot that identify objects and avoid collisions is hard.  Note, for example, the difficulties Tesla and others have with their self-driving cars.

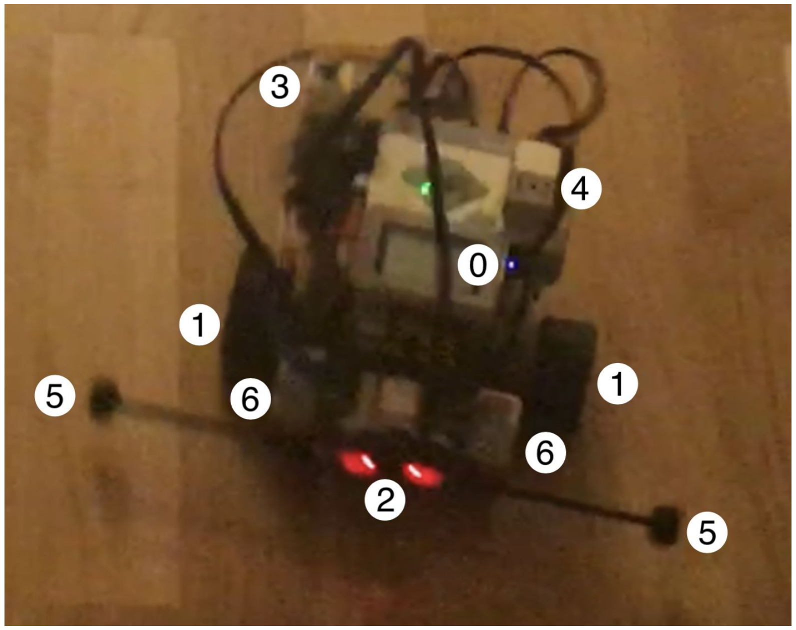
## Methods

### Overview

I developed two versions of the robot: Snoofer Bot 1 and Snoofer Bot 2.  The primary issue with mapping robots is error buildup.  I therefore sought to minimize this.

### Snoofer Bot 1

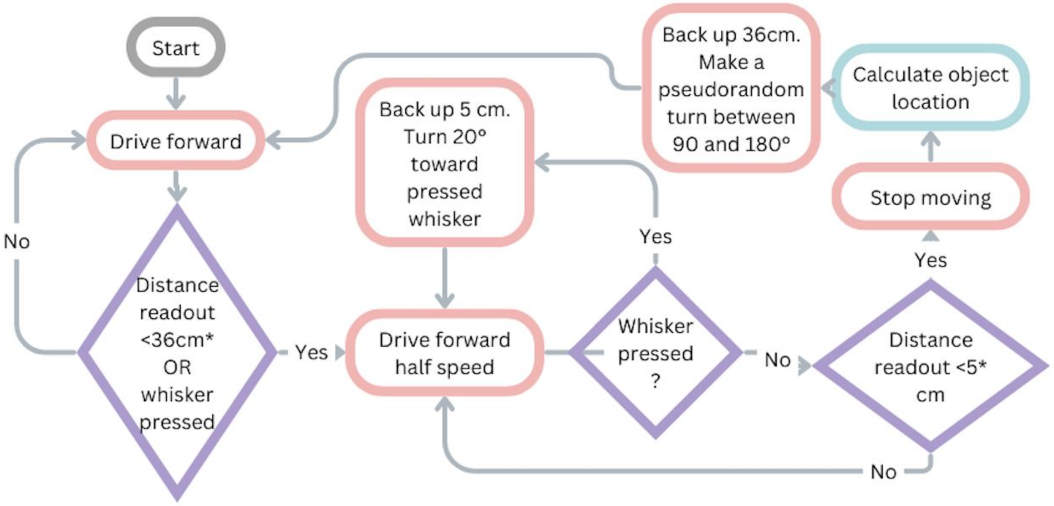
#### Hardware



Snoofer Bot 1 was made entirely of Lego components.
(0) Lego MindStorms EV3: The central computer.
(1) Two powered drive wheels: Move and rotate the robot.
(2) Ultrasonic Sensor: Measures distance to objects.
(3) Sensor motor: Swivels the ultrasonic sensor back and forth
(4) Gyroscope: Tracks the robot's rotation.
(5) Whisker: Is pushed back when it strikes an object and presses the touch sensor (6).  Used to feel for the position of objects.
(6) Touch Sensor: Detects when whisker is pushed back.

#### Software

Programming language: MicroPython
Software Flow Chart (simplified):


Start
Drive forward
Distance readout <36cm* OR whisker pressed
No
Yes
Drive forward half speed
Whisker pressed ?
Yes
Back up 5 cm. Turn 20° toward pressed whisker
No
Distance readout <5* cm
No
Yes
Stop moving
Calculate object location
Back up 36cm. Make a pseudorandom turn between 90 and 180°

Flow Chart Key
Gray: Internal operation
Purple:  Condition
Red:  Action
Blue:  Multistep process:

Calculate object location:  The robot determines the angle of the distance sensor based on the position of the sensor motor, and uses that with the distance  to calculate the relative position of the object, then adds to that its own location.

*For Snoofer Bot 2, these values were 30 and 7 cm.

## Methods (continued)

**Calibration**

In order to determine the angle of the sensor based on the position of the sensor motor (see Hardware above, (3) and (4)) I:

- Recorded a set of motor positions and the corresponding sensor angles
- Calculated a best-fit cosine curve for the data (crank mechanisms follow a cosine curve)
- Built that cosine function into the robot's code to track the position of the sensor

**Experiment**

The experiment was designed to test the robot's mapping ability. For the experiment, I constructed a roughly 1x2 meter arena with a 10cm high perimeter wall.

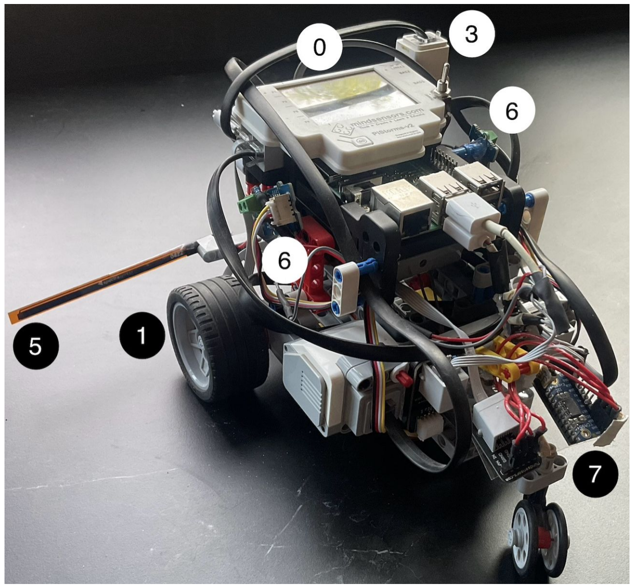
- Snoofer Bot ran its program while inside the arena.
- During the run, the robot was recorded on video.
- It drove around at random and mapped 24 objects (points along the perimeter wall).
- It transmitted their coordinates to a computer, for storage.
- After each run, I went over the video and recorded the coordinates of the actual points along the wall the robot had been looking at. The arena floor is marked with a Cartesian grid, making this straightforward.
- The point coordinates recorded by the robot were compared with the points that it actually encountered.
- Error was calculated as the distance between the two.

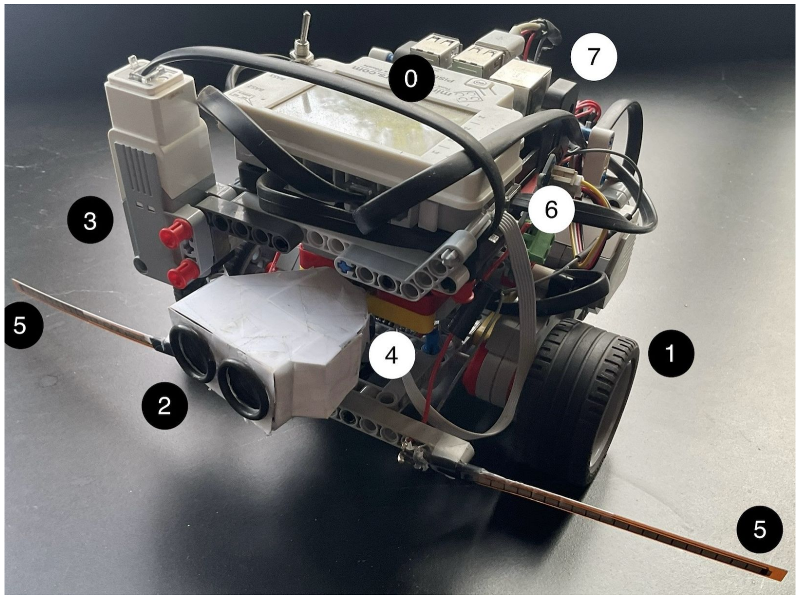
# Methods (continued)

## Snoofer Bot 2

After data collection with Snoofer Bot 1, it became clear that much improvement was needed. I therefore rebuilt the robot into Snoofer Bot 2 and rewrote the software.

## Hardware



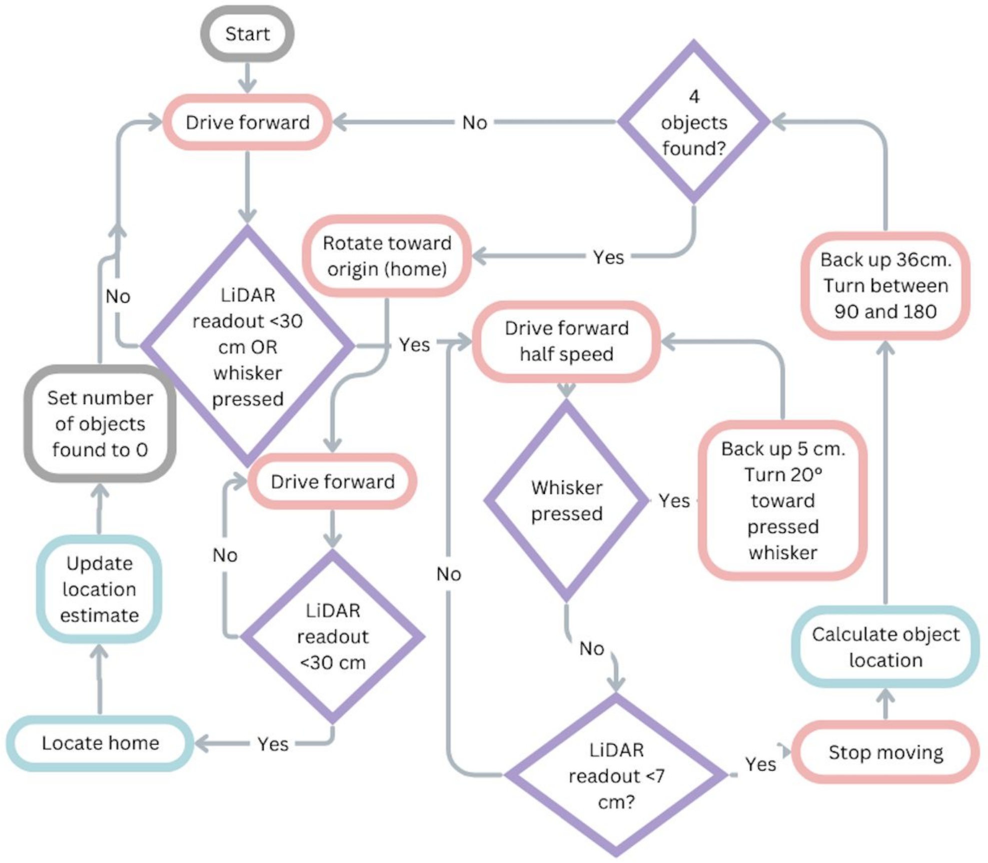


(0) Raspberry Pi with PiStorms-v2: The robot's central computer.
(1) Powered drive wheel: The robot uses two of these to move and rotate itself.
(2) LiDAR Distance Sensor: This sensor informs the robot of how far away objects are. It is mounted on a hinge.
(3) Sensor motor: This motor is connected by a crank to the LiDAR distance sensor (2). It swivels the sensor back and forth.
(4) Gyroscope (mounted inside robot, so not really visible): A gyroscopic sensor that tracks the robot's rotation.
(5) Flexible resistor: A component that changes its electrical resistance when bent. One is mounted on each side of the robot.
(6) Voltage sensor: This Grove sensor measures the voltage of across a voltage divider which includes the the flexible resistor (5).
There are two, one for each resistor.
(7): Sensor wiring: This group of components powers the voltage dividers (6), and converts their data into the I2C format.

# Methods (continued)

## Software

The software was rewritten in standard Python. Many of the changes were related to communicating with the new sensors, including adding calibration methods for the whiskers and gyro sensor. I also wrote the Find Home protocol, which enables the robot to periodically recenter itself by observing an object of known location. This was inspired by the behavior of animals, which often have a home from which they set out and return.



Start
Drive forward
No
4 objects found?
Yes
Rotate toward origin (home)
Back up 36cm. Turn between 90 and 180
No
LiDAR readout <30 cm OR whisker pressed
Yes
Drive forward half speed
Set number of objects found to 0
Drive forward
Whisker pressed
Yes
Back up 5 cm. Turn 20° toward pressed whisker
Update location estimate
No
LiDAR readout <30 cm
No
No
Calculate object location
Locate home
Yes
LiDAR readout <7 cm?
Yes
Stop moving

**Flow Chart Key**

Gray: Internal operation

Purple: Condition

Red: Action (Note that Drive Forward includes a feedback-based heading correction to keep the robot driving straight)

Blue: Multistep process:

Calculate object location: The robot uses the LiDAR distance and absolute angle of the LiDAR sensor to calculate the relative position of the object, then adds to that its own location.

Locate home precisely: The robot sweeps its LiDAR sensor back and forth, and locates the closest point on that sweep. ("Home" is a cylindrical object.)

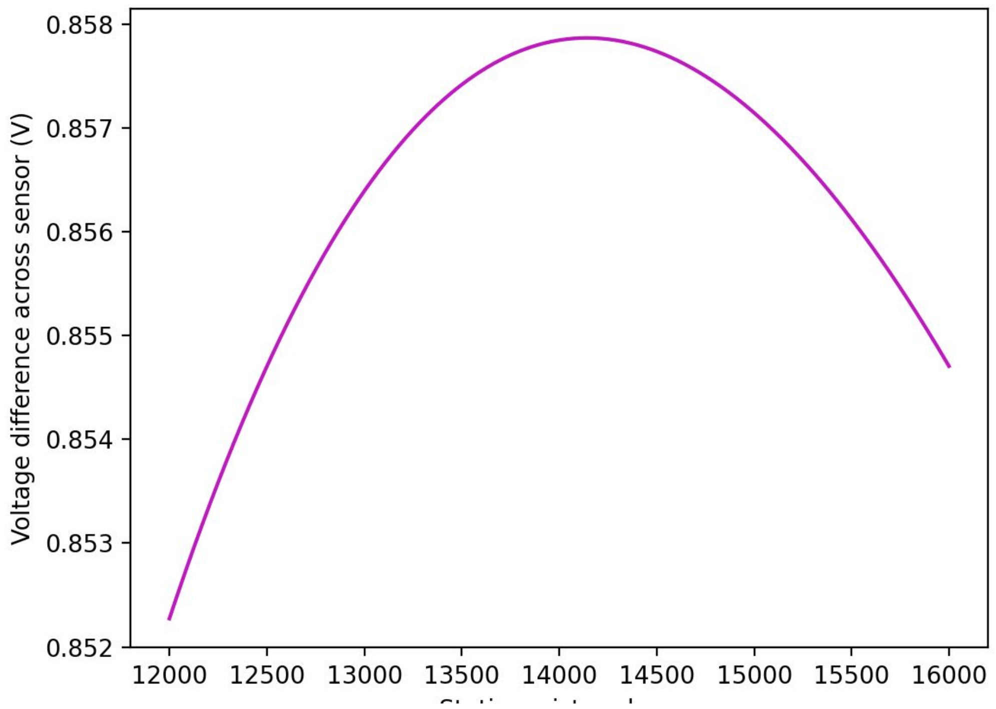
Use location of home to reset own location: The robot looks for the closest point on the home cylinder, and combines that distance with its own absolute angle (which it tracks independently) to reset its location estimate.

# Methods (continued)

**Whisker circuit design**

As described in the Hardware section, Snoofer Bot 2 has two flexible resistors, wired into a voltage divider circuit, with a voltage sensor measuring the voltage across the flexible resistor

The circuit needed a second, static resistor.  I wanted to determine what resistance would maximize the difference in voltage for the volt sensor when the flexible resistor was bent.  I created a graph of static resistance vs. voltage change when flexible resistor is bent, and concluded that the ideal static resistance was 14,150 Ohms.  I therefore chose a 14 kOhm resistor.



**Driving straight**

I found that the Lego motors that the robot uses to move were somewhat unpredictable in what speed they would run at.  I therefore wrote a Python thread which:

- Monitors the gyroscopic sensor
- Uses that to constantly tweak the power to each motor, compensating for any difference in efficiency between them, keeping the robot on an even heading

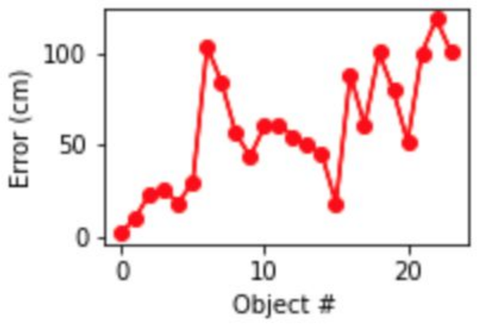
**Experiment**

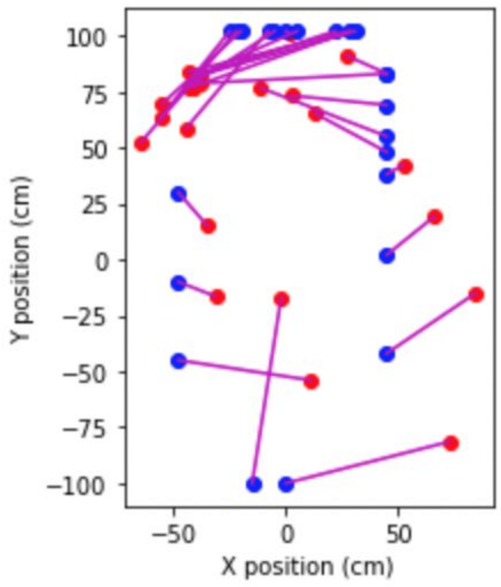
The experiments done on Snoofer Bot 2 were identical to those described under Snoofer Bot 1.

# Results

## Snoofer Bot 1

Snoofer Bot 1 was able to avoid running into objects, but did not produce an accurate map.
The graph on the right is a map of the arena. Each point that the robot mapped in plotted in blue, and connected by a magenta line to a red point, showing where the robot thought the object was.



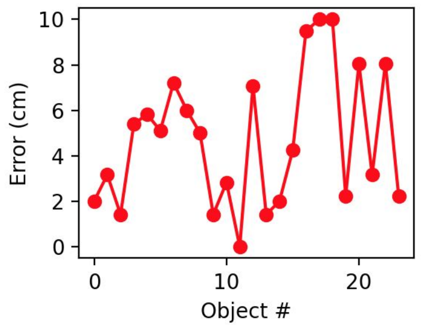


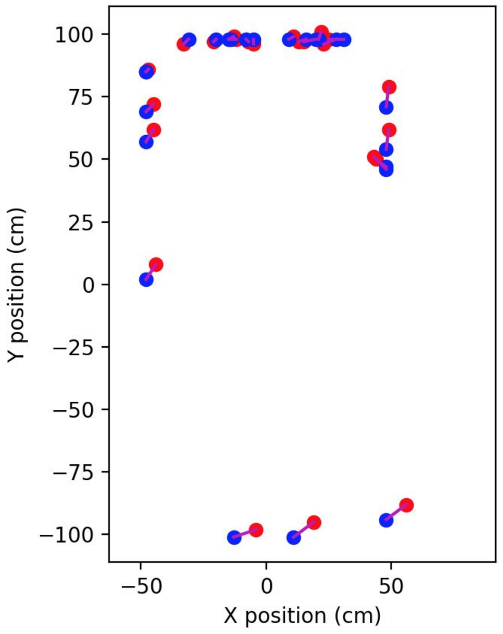
The graph on the left is error (length of magenta line on right-hand graph), plotted against object number.

## Snoofer Bot 2

These are the same graphs for Snoofer Bot 2.

Snoofer Bot 2 was able to map the arena with very little error.

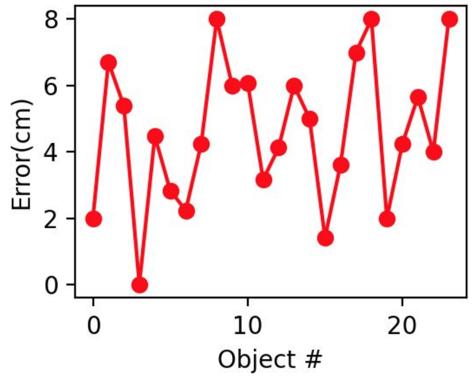




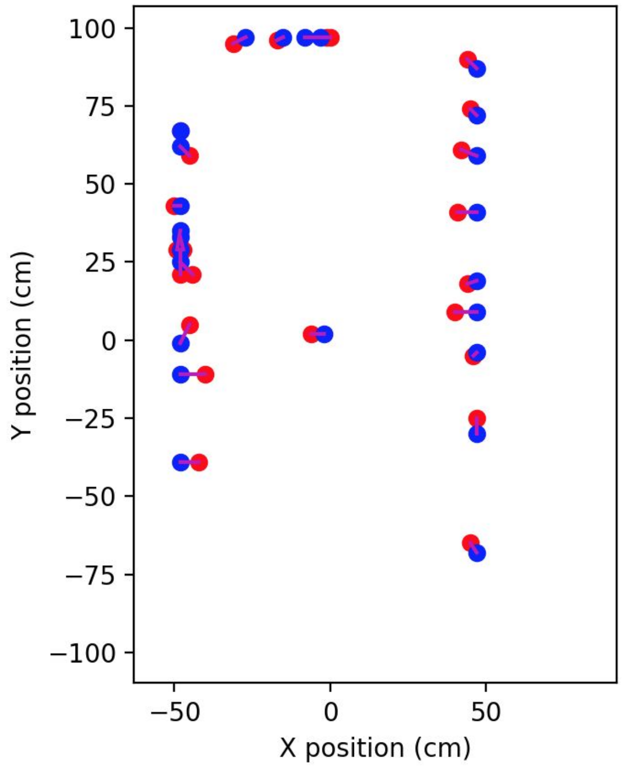
# Results (Continued)

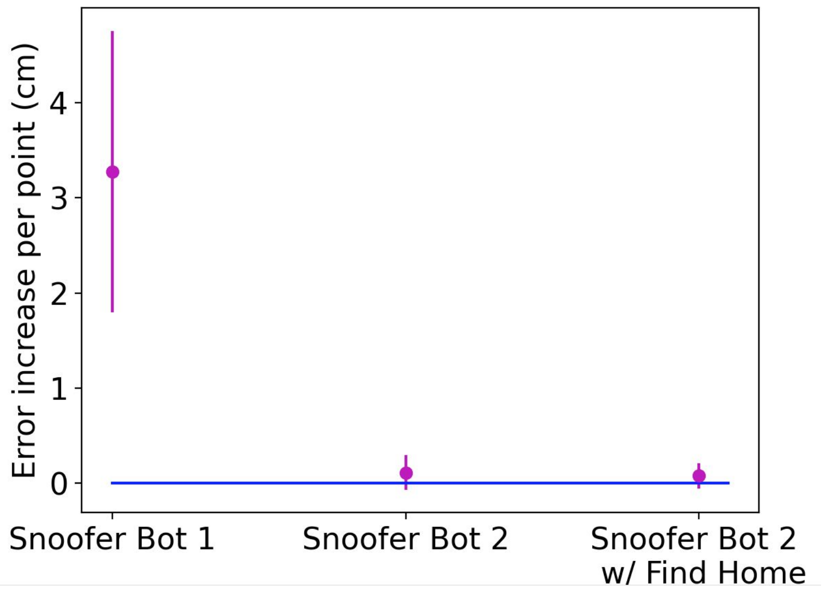
## Snoofer Bot 2 with Find Home

At right are the same graphs as in the previous slide for Snoofer Bot 2 using the Find Home protocol. The point set in the center of the lower right graph occurred when the robot mapped the “home” object.



To compare error buildup, I performed linear fit on the error graphs, and compared the slopes. Below are the slopes for the increase in error per object found for all three tested version of Snoofer Bot. Magenta lines indicate the 95% confidence interval. Snoofer Bot 2 has significantly less error accumulation than Snoofer Bot 1. ($p<<0.05$).





# Discussion

## Snoofer Bot 1

- Snoofer Bot 1 produced a highly inaccurate map, which would have been completely useless for understanding the shape of the arena.
- In one case, error reached 1.2 meters, comparable to the dimensions of the arena.

## Snoofer Bot 2

- Snoofer Bot 2 produced a highly accurate map, which gives a good idea of the shape of the arena.
- The average error increase per object was 0.11 cm.

## Snoofer Bot 2 with Find Home

- Snoofer Bot 2 with Find Home also produced a highly accurate map, which gives a good idea of the shape of the arena.
- The average error increase per object was 0.08 cm.
- This is slightly better than without Find Home, suggesting that the Find Home protocol increased accuracy.

## Conclusions

- Snoofer Bot 1 proved itself to be terrible at mapping
- Snoofer Bot 2 proved itself to be highly effective at mapping
- The Find Home protocol- where the robot tried to correct its estimate of its position- appears to slow down error accumulation

Due to the above, I consider the Snoofer Bot project to be a full success.  Although initial results were disappointing, after a complete redesign Snoofer Bot can produce a readable, usable map.  In addition, the effectiveness of the moving sensor, whiskers, and return-to-home speaks to the efficacy of biomimicry in engineering.

### Future Directions

There are a couple of ways in which Snoofer Bot could be improved.  These include:

- Allowing robot to use any known object, not just the “home” object
- Using a non-random movement pattern to increase mapping efficiency

### Applications

Snoofer Bot demonstrates a viable platform for robotic mapping.  This platform could be extended to almost any application where a robot deals with an unknown environment.  This encompasses everything from warehouse robots, to self-driving cars, to rovers on distance worlds.

## References and Supplemental Information

### References

(1): H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," in *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99-110, June 2006, doi: 10.1109/MRA.2006.1638022

(2): N. Zweifel and M. Hartmann. Defining “active sensing” through an analysis of sensing energetics: homeoactive and alloactive sensing. *Journal of Neurophysiology,* 124(1), pp. 40-48, 2020

### GitHub

The code for Snoofer Bot can be viewed on my GitHub page:
https://github.com/Aiden-Frank/Snoofer_Bot

### Presentation Credits

Except for cited information, the contents of this presentation are entirely my own work. No generative AI or similar program was used in any way.