Dancing Roomba Swarm

Design Document

Team 02

Client: Tyagi Akhilesh

Adviser: Tyagi Akhilesh

Team Members: Marcella Anderson (Software Engineer/Reports),
Joshua Arment (Software Testing/Meeting Scribe),
Adam Brandt (Software Testing),
Greyson Jones (Hardware Testing),
Noah Kiel (Chief Software Engineer),
Devon Kooker (Task Coordinator/Debugging),
Hunter May (Client Interaction/Hardware Testing)

sdmay22-02@iastate.edu

https://sdmay22-02.sd.ece.iastate.edu/

Executive Summary

Development Standards and Practices Used

- IEEE 802.11 - Wireless Networking
    - Allows easy connection between devices
- IEEE 754 - Floating point arithmetic specifications
    - Floating point allows for more precise measurements
- IEEE 1588 - Precision Time Protocol
    - Synchronize clocks across Roombas
- IEEE 1801 - Unified Power Format
    - Track power consumption of the Roomba, to maintain an acceptable charge life.

Summary of Requirements

- Roombas must be able to exhibit swarm-like behavior
- Follower Roombas must follow behind a lead Roomba at a 70 cm specified distance and 30 angle within 10% error
- The follower Roombas should not receive any controls and should rely only on their own sensor data
- The leader Roomba will receive movement directions from a base computer
- Components must be able to be powered by Roomba battery
- Components purchased for the Roomba will cost no more than $500

Applicable Courses from Iowa State University Curriculum

- CprE 288 - Embedded Systems I: Introduction
- CprE 488 - Embedded Systems Design
- ComS 309 - Software Development Practices
- ComS 327 - Advanced Programming for C/C++
- Math 166 - Calculus II

Skills Acquired throughout the Development of this Project

- Swarm algorithms and their implementations
- Determining which parts/devices to purchase
- Digital to physical conversion
- LiDAR programming experience

Table Of Contents

1 Team

1.1 Team Members

Marcella Anderson, Joshua Arment, Adam Brandt, Greyson Jones, Noah Kiel, Devon Kooker, Hunter May

1.2 Required Skill Sets for Project

- C programming experience
- Multiple Sensor Knowledge
- Software/Hardware Experience
- Digital-physical conversion skills
- Low-level Networking

1.3 Skill Sets Covered by Team

- C programming experience - Hunter, Joshua, Devon, Noah, Adam, Greyson, Marcella
- Sensor Knowledge - Hunter, Greyson, Adam, Devon, Marcella
- Software/Hardware Experience - Hunter, Joshua, Devon, Noah, Adam Greyson, Marcella

1.4 Project Management

We used the Scrum project management style. It required a lot of communication, but proved effective for our team.

The team was assigned the following roles to effectively manage the project:

- Joshua Arment - Meeting Scribe, Testing Lead
- Hunter May - Client Interaction, Hardware Tester
- Devon Kooker - Sensor Coordinator
- Adam Brandt - Software Engineer/Tester
- Greyson Jones - Digital Conversion Overseer
- Marcella Anderson - Report Manager
- Noah Kiel - Chief Software Engineer

2 Introduction

2.1 Problem Statement

Implement a design so that a collection of Roombas will follow a lead Roomba that moves via song analysis based on certain specifications.

2.2 Requirements & Constraints

2.2.1 Functional requirements

- Roombas must be able to exhibit swarm-like behavior, follower Roombas will be 30 degrees offset from lead Roomba
- Follower Roombas must follow behind a lead Roomba at a distance of 70 cm
- Specified distance is within a 5cm error and angle is within 5 degrees.
- The follower Roombas should not receive any controls and should rely only on their own sensor data
- The leader Roomba will move based on analyzing song data
- Components must be able to be powered by Roomba battery

2.2.2 Economical Requirements

- Components purchased for the Roomba will cost no more than $500

2.3 Engineering Standards

- IEEE 802.11 - Wireless Networking
    - Allows easy connection between devices
- IEEE 754 - Floating point arithmetic specifications
    - Floating point allows for more precise measurements
- IEEE 1588 - Precision Time Protocol
    - Synchronize clocks across Roombas
- IEEE 1801 - Unified Power Format
    - Track power consumption of the Roomba, to maintain an acceptable charge life.

2.4 Intended Users and Uses

2.4.1 Users

- Iowa State University Computer Engineering 288 Students and Faculty
- Fire Rescue Teams
- Defense Systems
- Self Driving Cars

2.4.1 Use Cases

- Create a swarm of n Roombas.
- Control the lead Roomba, and the swarm follows.
- Play music for the lead Roomba and it moves the swarm, making the swarm “dance”.

3 Project Outline

3.1 Project Management/Tracking Procedures

We managed our project with Trello, which is a web-based project management application that focuses on continuous small changes. Trello works by visually organizing tasks in columns according to their stage in the development process. Trello allows for the backlog of tasks to be constantly changing as well as provides openness about the progress of the project. A combination of a Trello board and various GitLab features helped track progress and manage tasks.

3.2 Task Decomposition

1. General Roomba Setup
    a. Develop template for general Roomba control
    b. Develop control systems for LiDAR and servo
2. Implement leader robot algorithm
    a. Implement wireless control of leader
    b. Implement a way for the leader to receive dancing instructions
3. Implement follower robot algorithm
    a. Interpret LiDAR data
    b. Locate leader Roomba
    c. Implement movement protocol
4. Implement Song Analysis for Leader movements
    a. Interpret songs to find high, mid, and low frequencies

    b. Send character to lead Roomba to have it dance across room
5. Refine Roomba Software and Movements
    a. Adjust software to better comply with specifications by client
    b. Add programs which enhance ability and responsiveness of Roombas

3.3 Project Milestones, Metrics, and Evaluation Criteria

- Roomba control template is coded and configured with the servo and LiDAR sensor
    - LiDAR sensor can be read by Roomba and is accurate within 1 cm
- Lead Roomba can be operated wirelessly
    - The lead Roomba can be controlled by a user without a song
- Followers will follow a leader with less than 5cm deviation from the prescribed following distance (70cm).
    - Members of the swarm will adjust their speed and angle according to where they perceive the leader
- The swarm can reliably move in formation while the follower is 70cm behind the leader
    - The swarm follows the lead Roomba within specifications whether the lead Roomba is controlled manually or via music
- Develop Roomba routine that uses sound/song as input to control the swarm
    - Lead Roomba will follow movement directions based on analysis of a song that the user plays

3.5 Risk Management / Security Concerns

- Hardware Needs
    - Hardware incompatibility of LiDAR with Roomba/Tiva board
    - There are workarounds that may exist to still make the system work.
    - Risk probability: 0.2
- Follower robot algorithm
    - An algorithm is required to process sensor data, and decide how to move the bots.
    - Other objects could cause the follower Roombas to lose track of the leader Roomba
    - There is currently no obstacle detection, so there could be problems in an unknown environment
    - Risk probability:0. 45

- Wireless Communication
    - Unwanted signals could be sent over putty
    - Communication could be interrupted while functioning

4 Design

4.1 Design Context

4.1.1 Broader Context

Our project can help autonomous entities organize their movement and position to achieve a common goal. Many different areas of society have applicable applications that would benefit from the organization of multiple entities to accelerate or optimize a process.

Area	Description	Examples

Public health, safety, and welfare	Fire Rescue Drones	Increased ability to locate fire victims Carry/disperse fire retardants Can function if infrastructure is broken
Global, cultural, and social	National Defense Systems.	Could lead to different types of software in defensive drones
Environmental	Drones which can release fire-fighting chemicals	Increasing ability of drones used to tackle and/or prevent forest fires
Economic	Self Driving Cars	Ease of development of self driving cars could lead to lower costs

4.1.2 User Needs

Fire Rescue: Fire rescue needs a swarm which can work autonomously to search for people because internet and inter-device communication can not be counted on in fire rescue.

Defense Systems: Defense systems need swarms which can work autonomously because then communication between devices cannot be intercepted, interrupted, or interfered.

Self Driving Cars: Self driving vehicles need to work autonomously with each other and know each other's locations in order to prevent vehicle collisions.

4.2 Design Exploration

4.2.1 Design Visual and Description

Fig. 1. Roomba flock-swarm of 3

The diagram above (Fig 1) shows the organization of our Roomba flock-swarm. To allow the LiDAR on the follower Roombas to track the leader Roomba, we used a long pvc pipe as reliable markers for the sensors to identify. The follower Roombas are 70cm behind and 30 degrees offset from the PVC pipe. These distances

and angles are reflective of the space between the center of the LiDAR sensor and the center of the pvc pipe on the leader.

4.2.2 Intended vs. Final Design

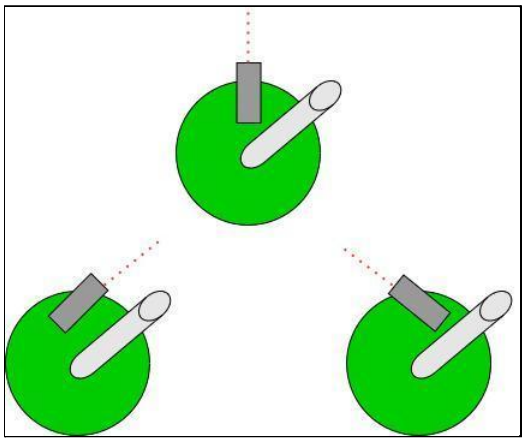


Fig. 2. Intended design plan of Roomba flock-swarm.

Fig 3. Image of implemented Roomba flock-swarm

The diagram above (Fig 2) shows the organization of our Roomba flock-swarm. The dark gray boxes represent our 360 degree LiDAR. To allow the LiDAR to locate other bots we will use long pvc pipes, represented as light gray cylinders above, as reliable markers for the sensors to identify.

In the above images, Fig 2 is the representation of what we believed our project to look like at the end. Fig 3 was taken in the lab with the final changes made to the Roombas. There a two key differences to note:

1. The leader Roomba does not have its own LiDAR system
2. The follower Roombas do not have tracking poles.

The reason that the lead Roomba was excluded from having a sensor was to simplify our design and apply our limited time to the follower algorithms. The followers do not have tracking poles, because we discovered that they would occasionally pick up the wrong pole and start following that one instead.

4.2.3 Areas of Concern

The biggest area of concern was getting the LiDAR to accurately identify and measure the distance and direction of the leader. We used a 360 degree LiDAR which looks around in all directions to follow the leader. However, there were difficulties where occasionally the follower Roombas would be unable to locate the lead Roomba. To help remedy this problem we had the follower Roombas continue to spin in a circle to locate the lead Roomba.

Further Areas of concern include the height of the LiDAR sensors. Because the sensors are at the same height we would occasionally see one follower Roomba begin to follow the other Roomba. This could be solved by staggering the heights of the LiDAR sensors.

4.3 Technology Considerations

The main technical consideration for our project is the sensor we use to determine relative position. The provided hardware includes a directional sonar and IR sensor mounted to a servo to determine distance and direction. The advantages of this system are the already existing software support as well as the price, this system is very inexpensive in comparison to its counterparts, the weaknesses however include a limited view, the sensors limit the sensing distance and accuracy compared to more advanced(expensive) systems. The servo motor also limits the arc of sensing the robot can achieve, limiting the system to only sensing what is directly in front. We considered a LiDAR sensor attached to the 180 degree servo motor included on the roomba, this led to an increase in cost but with better accuracy and distance capabilities, but this still led to the disadvantages a servo comes with, a limited arc for sensing. Our final option and solution was a LiDAR sensor with a servo to allow the LiDAR to sense in all directions. This sensor, although with a high price tag comes the improved sensing distance of a LiDAR sensor along with allowing that sensing range to be in a complete 360 rather than limited to the arc possible by the 180 degree servo. We decided on this sensor because for the robots to not communicate with each other it is vital they understand their surroundings.

4.4 RP LiDAR

For our final decision on the LiDAR sensor, we chose the RP LiDAR-A1, created by Slamtec. This option included the 360 degree range of motion that we thought necessary with a relatively low cost. A link to the device is included in section 7.2 References.

4.5 Design Plan

We modified the Roombas with a LiDAR sensor by mounting them on the Roombas.. This allowed us to have the Roombas be able to follow the leader Roomba. We then modified the existing code base from the previous team's project to be able to efficiently have the Roombas follow the lead Roomba. In addition, we modified the lead Roomba to be able to analyze music and create a dance routine. The follower Roombas will then follow it in a swarm pattern. We will then follow our testing guidelines to ensure that the Roombas work properly.

4.6 Design Conclusion

For the final iteration of our Roomba swarm, we decided to use a rotating 360 degree LiDAR attached to the top of the Roomba. The reasoning behind this includes the flexibility of seeing in all directions around the Roomba, the low computational overhead and a relatively low cost compared to the other options.

Appendix 2.1 has more concise descriptions of the alternative options that were considered for our project.

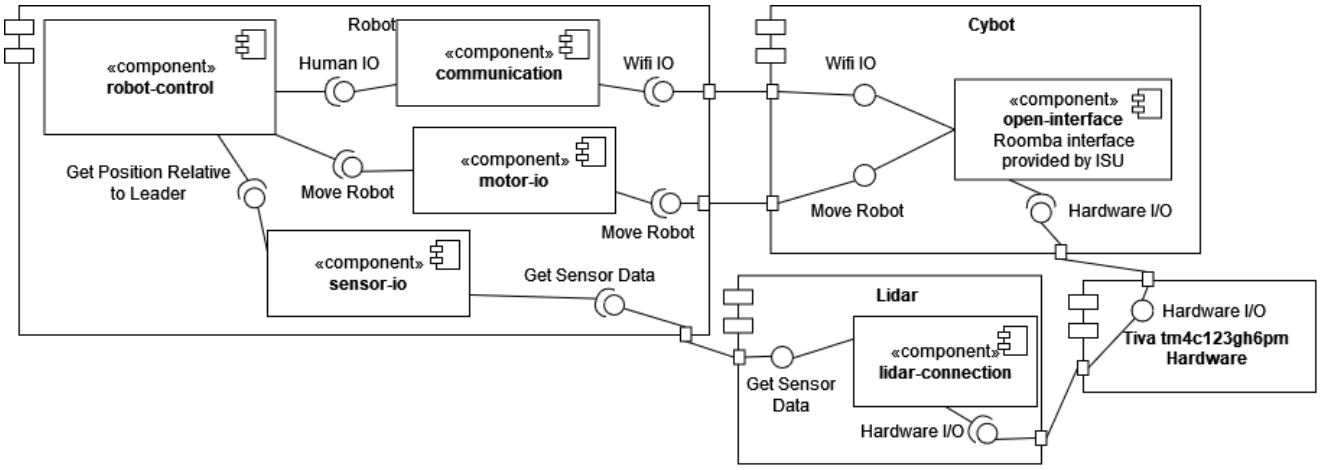


Fig. 4. Design Diagram for Roomba flock-swarm

5 Implementation

5.1 LiDAR

LiDAR implementation required hardware control to send and receive data over UART. We had to implement messaging to start the LiDAR and put it into scan mode, along with additional code to parse the input stream from the LiDAR for data packets. We included many helper functions to easily store and access large amounts of stored data.

5.2 Leader Roomba

The leader robot is controlled over Wi-Fi with keyboard input(8.1.1 for details), this allows the operator to move or test the swarm with the leader robot. It also has a test mode that allows the operator to select one of the preplanned test routines, rather than manual operation

5.2.1 Dancing

We utilized a method of allowing a python program to listen to the audio going through the speakers. This was accomplished by downloading a virtual audio cable and using the program Voice Meter to redirect the audio inputs and outputs. The python program utilizes the Pyaudio library to gather audio samples from the virtual audio cable while using the Numpy library to analyze the FFT(Fast Fourier Transform) of the computer's audio. taking the FFT data we were able to divide the sound into individual frequencies with their respective power values in real-time. We then divide the frequencies into 3 groups; High, Medium, and Low. By taking the average power value of these frequency ranges we are accurately able to determine when a High, Medium, or low frequency hits a peak power level.

The power levels of the averages are then recorded and incremented over the time length of one measure of any song, then sends the frequency range that had the highest recorded average power level of the three frequency ranges to the bot as a single character value of H, M or L. The bot then moves left if an “H” is received, straight if an “M” is received and right if “L” is received.

All of these components combined result in the formation of bots “dancing” to the music. The program works quite quickly with most latency coming from timing the program to the music instead of sending it to

and from the bot. However, this could be later solved in the future by letting the program start a music file on its own or utilizing Spotify's API to start the currently selected song to time things perfectly.

5.3 Follower Implementation

The follower robots needed to do two things to flock in formation. First they needed to locate the leader. Next if the leader was not where they expected (which would imply the follower needed to move) they needed to set their wheels to move in the direction they should go. We describe how these two actions were completed below.

5.3.1 Finding the Leader

The Roomba uses the LiDAR's scanned data to find the pole on the leader robot. The LiDAR data is processed and given to this function as a series of angle-distance pairs. We compare each angle and distance to the last known location of the pole. These comparisons are used to get our angle and distance errors. We compare these errors to an angle looking margin and a distance looking margin. These two margins are shown in green in Fig. 5 below. If the angle and distance errors are within the looking margins, we assume the point we are seeing is the pole. These points which are assumed to be the pole were then used to make motion decisions.

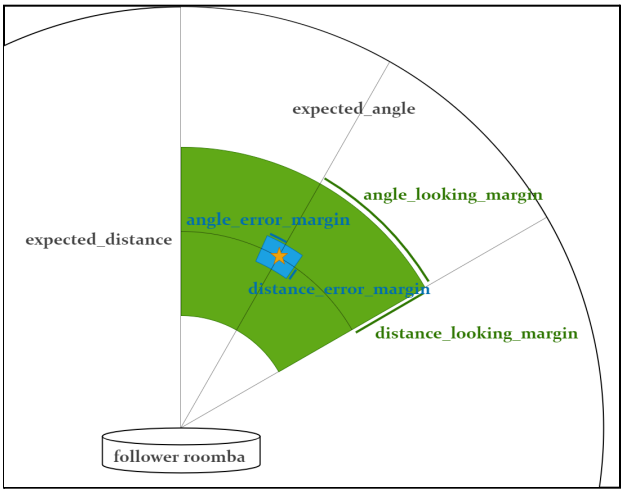


Fig. 5. Margins for a left follower Roomba if the last known location of the pole was the expected location of the pole.

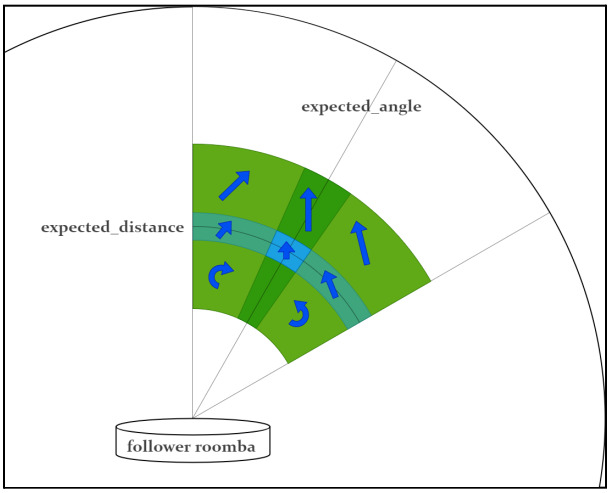


Fig. 6. shows the movement of a left follower Roomba depending on where the pole was located.

5.3.2 Determining Movement

We determine movement using the angle and distance which we have located the pole at (Section 6.3.1).

We first look at the distance. We compare the given distance to the expected distance. If the given distance is within the distance error margin of the expected distance we set the speed of the wheels to the speed of the leaders wheels. If the given distance is more than the distance error margin away from the expected distance we set the speed faster than the leader's speed. If neither of these cases occur the follower is too close to the leader so it sets both wheel speeds to 0.

Next we look at the angle. We compare the given angle with the expected angle. If the given angle is within the expected angle margin we do nothing. This ensures the follower's wheels will have matching speeds set by the distance comparison described above. The left to right motions of the Roomba depend on whether the Roomba itself is meant to be on the left or right side of the leader. To explain how the given angle affects movement we will assume the follower is on the left as this is what is shown in Fig. 6. For this Roomba if the given angle is on the left we will reduce the speed of the right wheel so it turns right. The reverse is true if the given angle is to the right of the angle error margin.

6 Testing

6.1 System Testing

Things which need to be included in system testing:

- Leader system testing
    - Robot follows desired path
- Follower system testing
    - LiDAR detects leader
    - Robot follows leader at the expected distance
    - Left and right robot follow the leader at the correct angle from the leader and each other

Using the Roombas there are two major systems to test the leader system and the follower system. Both systems require the robot's movement, sensor, and LiDAR systems to detect the appropriate tracking pole attached to the leader. These interfaces and specific subsystems allow both robot systems to have the basic functionality required of any robot that would belong in this swarm, to move and avoid obstacles. The leader system would require specific testing to verify any pre-programmed paths can be followed. The follower system would require additional tests that allow it to properly follow the leader. This includes the following leader interface verifying distance and angle.

6.2 Regression Testing

We ensured new things don't break old things after any major changes by putting the Roomba on the ground and running it through various preconfigured tests to check if it moves and responds as expected. Checking various subsystems frequently will allow us to detect where in our implementation of the leader and follower robots we disrupt our systems if such an event occurs. The system we expect to build is very dependent on a variety of subsystems operating correctly and in unison. Updates to the system must be met with testing to ensure that all the dependent systems are kept in working order so new capabilities don't disrupt the old or the tasks previously accomplishable. The main focus of our regression testing will be maintaining that the robot subsystems are kept in working order.

6.3 Acceptance Testing

Test against requirements or other client set constraints

- Build tests to show simple desired flock movement

- Roombas follow the given movement routine
- Build advanced tests showing the flock avoiding obstacles

We will use a video to show the client, as well as to analyze position, angle, etc. We will use this video and the specifications of the movement routine to see if our project meets the criteria above.

6.4 Results

- Results will be in the form of visual verification by the team
- Results of tests will be recorded on video or documented with a written report
- Analyze video for robot meeting positional functional requirements

Our requirements talk about the movement of the Roombas in relation to their environment and each other. If the Roombas move in the way we expect them to in a testing environment then they are meeting the requirements. Therefore watching the Roomba's movements will give us our results.

7 Closing Material

7.1 Conclusion

Our goal for this project was to create a swarm of Roombas which perform a dance routine. To do this we attached a LiDAR to the robots, as well as a pole for detection. We were able to put the Roombas into a flock-like configuration for the swarm to follow the leaders around. During our first semester we were held back in our work by the LiDAR sensor. This roadblock was overcome and has paved the way for a successful second semester. In future iterations it is most important to have the sensors bought early on in the process. In the final month of the project, we were able to get the follower Roombas to find the leader and follow it around. On another device, music was being played for the Roombas to dance to. The dance moves were generated on the device from music being played and interpreted into simple movements. The leader took input over the network of dance moves to do, making the entire flock dance around the room.

7.2 References

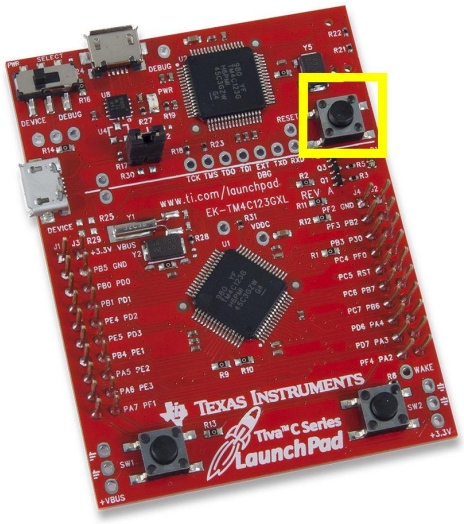
Hauert, S. et al, 2021. Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/6095129>.

Virágh, C. et al, 2021. Flocking algorithm for autonomous flying robots. [online] Arxiv.org. Available at: <https://arxiv.org/ftp/arxiv/papers/1310/1310.3601.pdf>.

Slamtec RP LiDAR Available at: https://www.slamtec.com/en/Lidar/A1

Appendix I: Operational Manual

AI.1 Putty Control

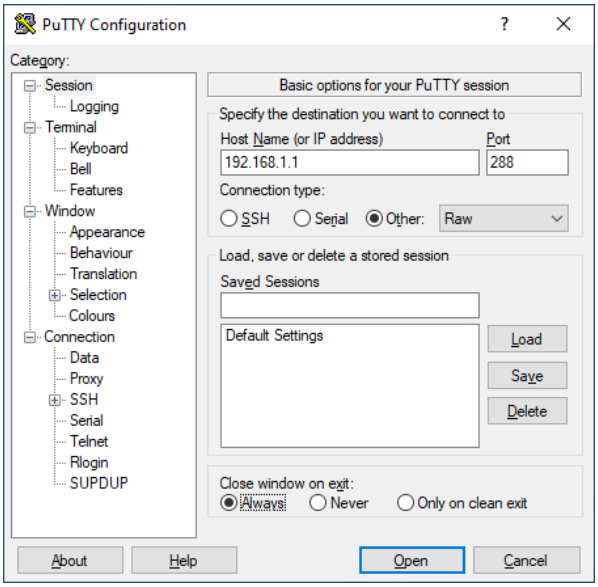


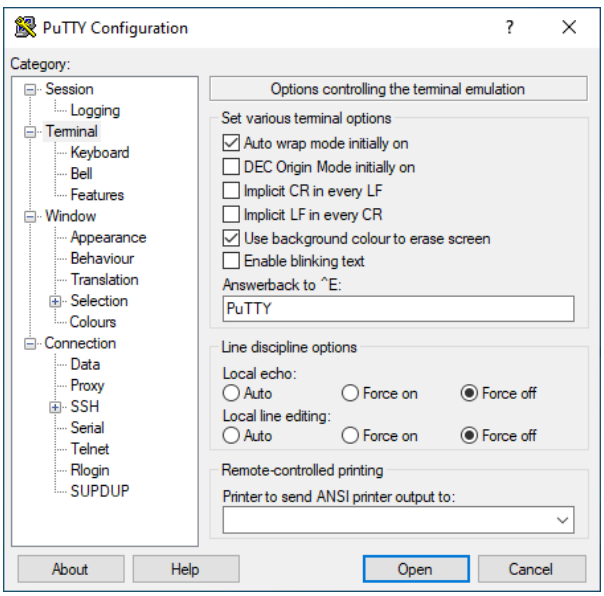
The leader and followers are set up with working code, all you need to control the swarm is a simple Wi-Fi connection and Putty. If you would like to reflash the software or make modifications skip to the end of the Operational Manual for instructions on setting up the development environment. To run the current software, first setup the leader and followers in the proper flock formation, then hit the Tiva board's reset button to get the programs started; it is indicated on the image to the right.

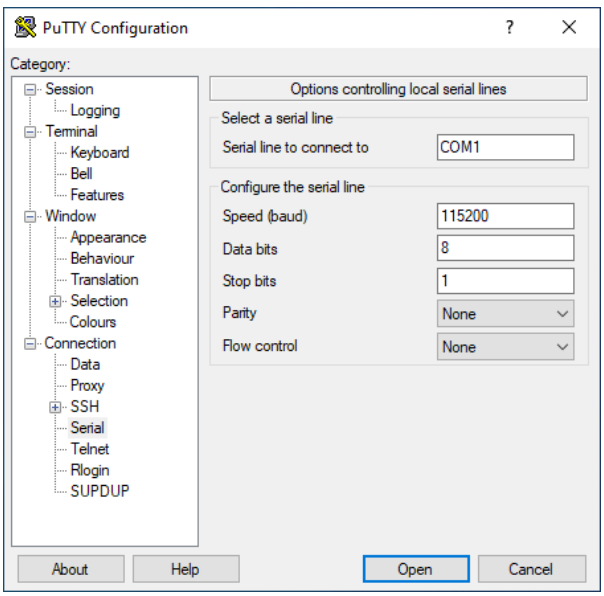
To control the leader with putty first connect to the Wi-Fi hotspot that corresponds with the robot (currently cybot 10) the password is "cpre288psk" if prompted. Then setup putty with settings below to connect and start sending commands.

Putty Settings:

- **Session**
  - Connection Type: **Raw**
  - Host Name: **192.168.1.1**
  - Port: **288**
  - Close window on exit: **Always**
- **Terminal**
  - Force local echo: **off**
  - Force local line editing: **off**
- **Connection -> Serial**
  - Speed (baud): **115200**
  - Data bits: **8**
  - Stop bits: **1**
  - Parity: **None**
  - Flow control: **None**







Once connected with putty the following commands can be used to control the robot.

q : Veer Left

w : Go Forward

e : Veer Right

a : opt Left Turn

s : Go Backward

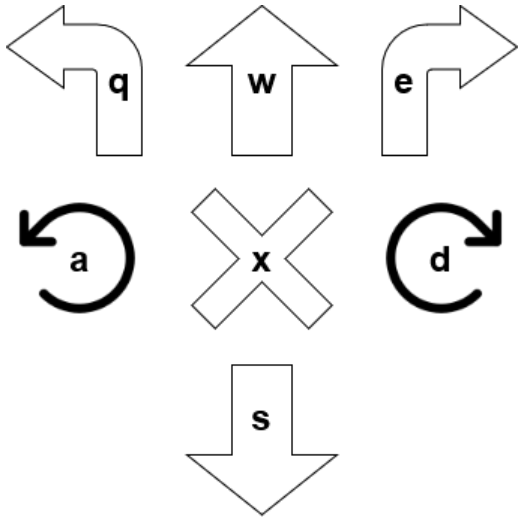
d : opt Right Turn

g : Full Throttle Forward

x or ‘space‘ : Stop Moving

f : Stop Moving and Halt Robot

t : Test Mode (if enabled)



In test mode the robot waits for a decimal input to select the desired test. Input the desired test number then any non number character to select that test.

AI.2 Dance Dance Roomba

In order to run FFT_Analyzer.py you will need to follow a few steps:

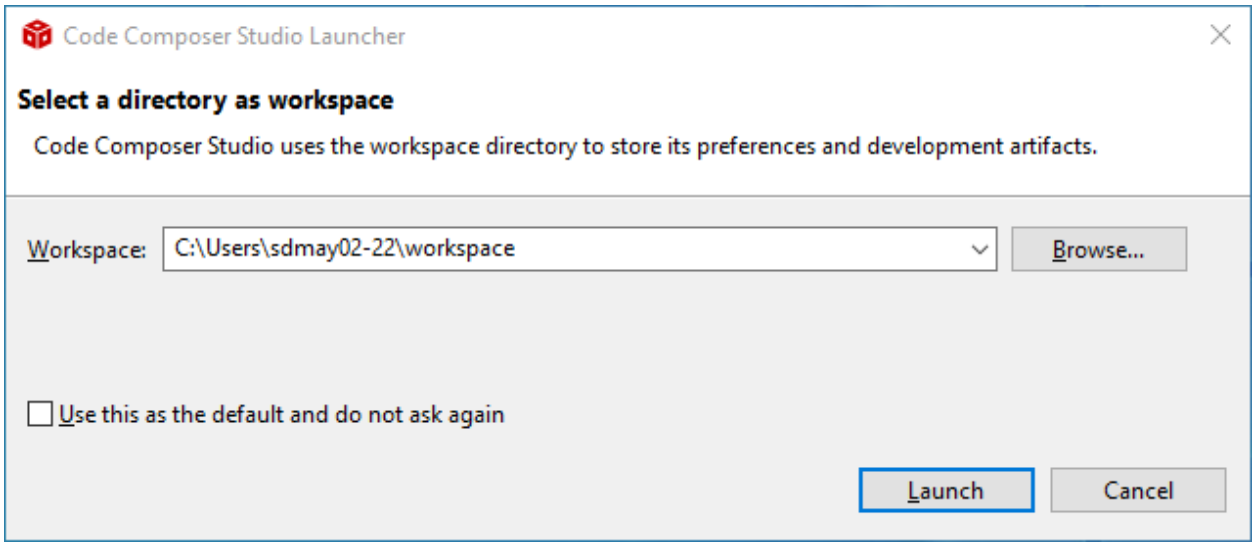
1. First, make sure you have python 3.7 installed on your computer
    a. Specifically between 3.7.3 - 3.7.9 (3.7.10 will not work with the libraries you will need)
2. Install Virtual Audio Cable for free from this site: https://vb-audio.com/Cable/
3. Install Voice Meter Banana for free from this site: https://vb-audio.com/Voicemeeter/banana.htm
4. Open a terminal in the location you wish to save FFT_Analyzer.py and run the following commands:
    a. pip install pyaudio
    b. pip install numpy
    c. pip install matplotlib
    d. pip install math
5. Next, you will need to change the audio input and output for your computer from pc settings:
    a. Output will be set to “CABLE input(VB-Audio Virtual Cable)
    b. Input will be set to “CABLE output(VB-Audio Virtual Cable)
6. Finally, you will need to change the audio input and output on Voice Meter Banana:
    a. Open Voice meter and let the sound engine start (ignore any error messages that pop up)
    b. Set “Hardware Input 1” by clicking on the 1 icon and choose CABLE output(VB-Audio Virtual Cable)
    c. Set “Hardware Out” by clicking on the A1 icon and choose “WDM: * *name of computer speakers** ”
7. Now open the same terminal as before and run “python FTT_Analyzer.py”
8. Play music from any online or offline source after seeing the words “connected” print in terminal
9. Completed Dance routine steps

(if you do not see the printout “connected” then the computer has not successfully connected to the leader romba)

AI.3 Developer Environment Setup

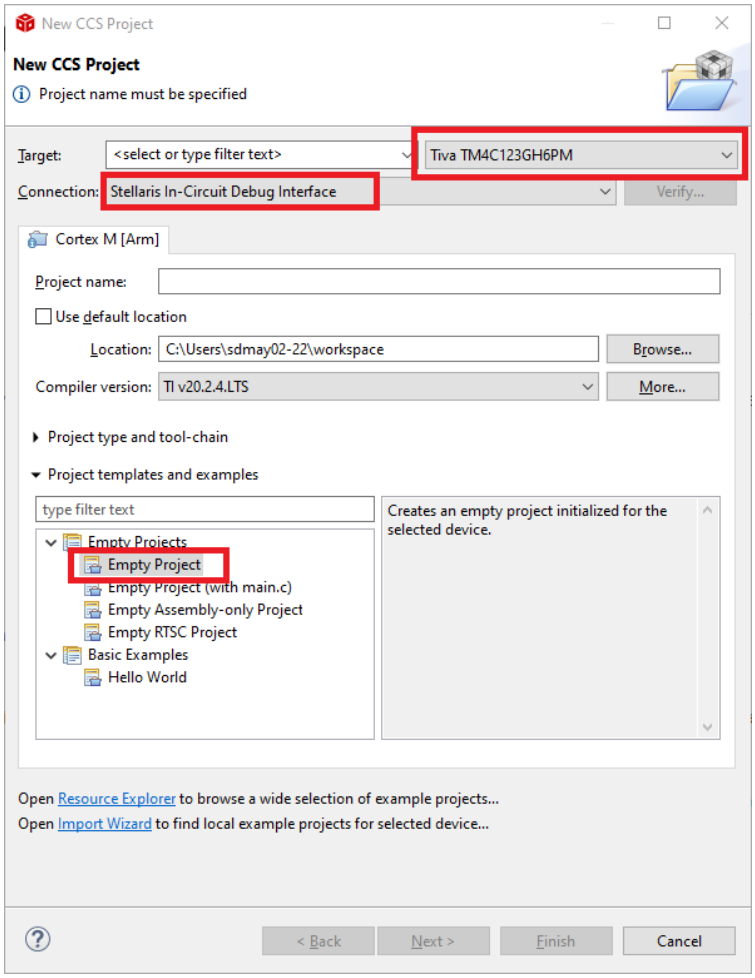
These setup instructions are for setting up our environment in the ISU Roomba lab. If you are not using a computer in this lab, you will need to install Code Composer Studio and download the latest "TivaWare™ for C Series" from Texas Instruments website.

Open Code Composer Studio, select a directory as your workspace.



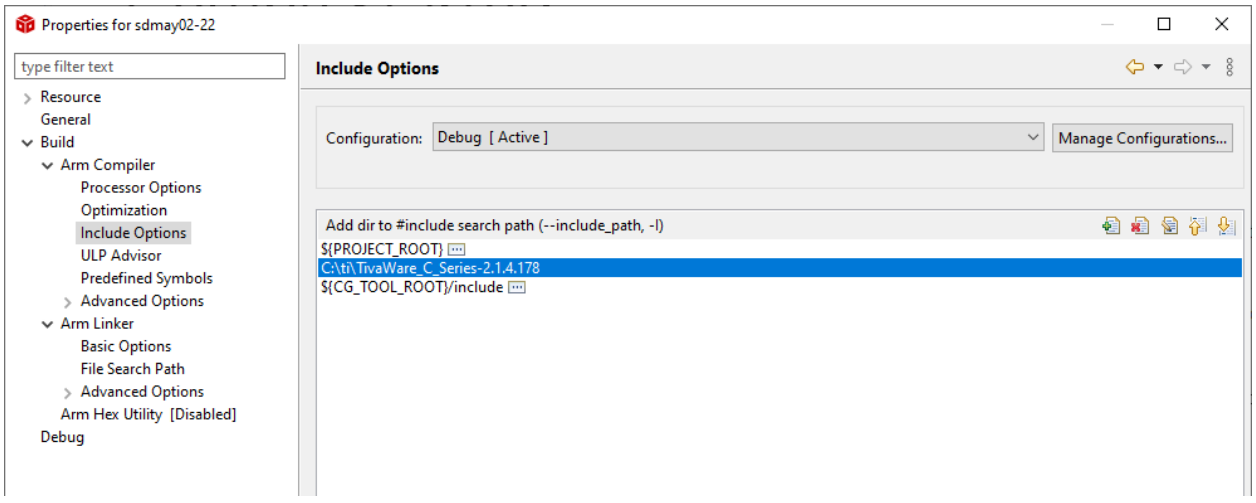
Select File -> New -> CCS Project

Select the following settings, and enter a project name, then click finish to create the project.

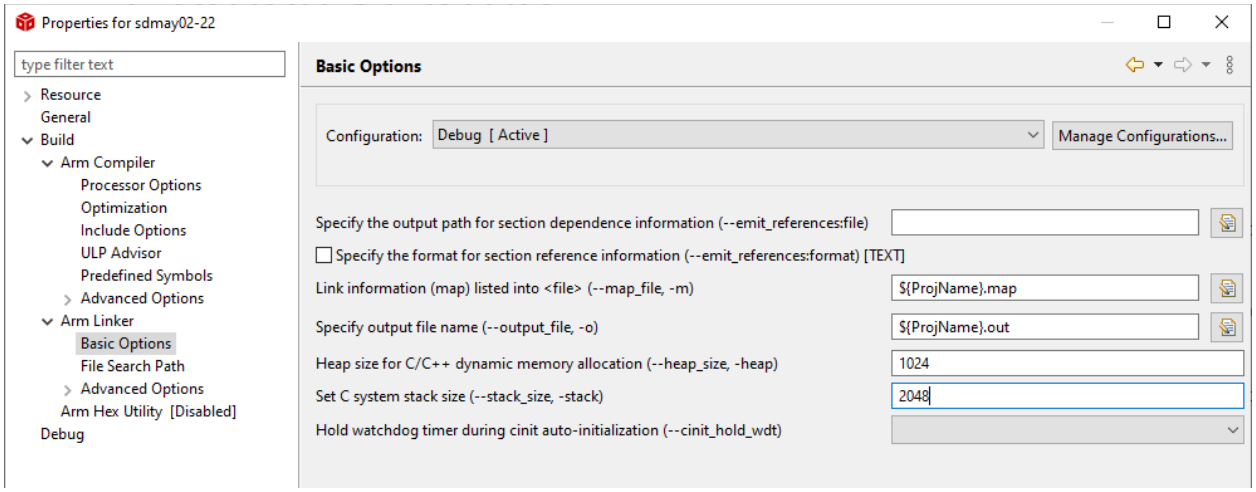


Next right click on the new project and select Properties

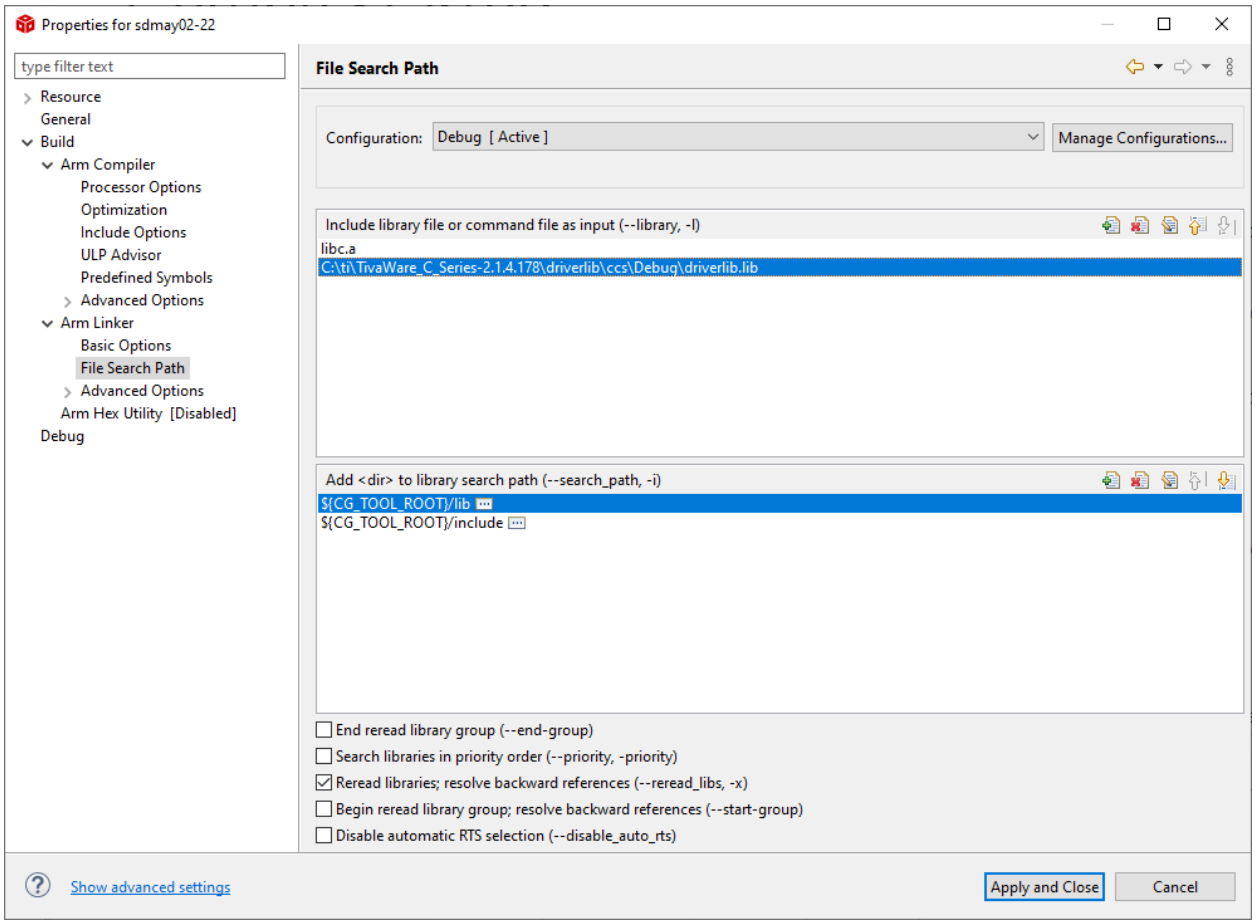
Go to Build -> ARM Compiler -> Include Options, then add “C:\ti\TivaWare_C_Series-2.1.4.178”



Also go to Build -> ARM Linker -> Basic Options, set heap to “1024” and stack to “2048”



While still changing project settings, go to Build -> ARM Linker -> File Search Path and add "C:\ti\TivaWare_C_Series-2.1.4.178\driverlib\ccs\Debug\driverlib.lib"



Then click "Apply and Close" now the project environment is set up. Next copy the "project" directory from the Git repository into your project directory. In the main file you can change which robot program will be run as well as modify which side the follower will be on.

To Debug the software make sure the Roomba is plugged in and click the “Debug” button, if you would like to deploy the code to use away from the computer click the “Flash” button.

Appendix II: Alternate Versions

AII.1 Alternative Designs

- **Keeping the previous project’s directional LiDAR**
  - The directional LiDAR wasn’t included as we needed the flexibility of looking at the full 360 degree area around the Roomba.
- **Using the on board IR and Sonic Sensors**
  - Similar to AII..1, the on board IR and Sonic Sensors were not flexible enough for what we wanted to do.
- **Wi-Fi strength triangulation**
  - While using Wi-Fi to triangulate is possible, the degree of precision that we require, could not be achieved with this method.
- **Cameras and Vision Processing**
  - After heavy consideration for this method, it was decided that the computational overhead of image processing would be too much for the single core processor on the Roomba. With a more powerful processor this would be a more viable option.

AII.2 Alternative Implementation: Wall Detection

Wall detection was designed to help the robot track the walls of the environment by tracking the LiDAR data points trajectory, so it can find the leader even when near a wall. The algorithm was designed and code built, but the hardware we are using doesn’t have enough memory to support storing this data. We modified the logic and algorithm to try and minimize the storage but this still did not fix the issue.