1 Introduction

1.1 Problem Statement:

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

1.2 Requirements & Constraints:

1.2.1 Functional requirements:

- Roombas must be able to exhibit swarm-like behavior
- Follower Roombas must follow behind a lead Roomba at a 60 cm
- specified distance and 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

1.2.2 Economical Requirements:

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

1.2.3 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.

1.3 Intended Users and Uses:

1.3.1 Users:

- Iowa State University Computer Engineering 288 Students and Faculty

1.3.1 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”.

2 Project Plan

2.1 Project Management/Tracking Procedures

We will manage our project with KanBan which is an agile methodology management system that focuses on continuous small changes. KanBan works by visually organizing tasks in columns according to their stage in the development process. KanBan 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 KanBan board (probably on Trello) and various GitLab features will help track progress and manage tasks.

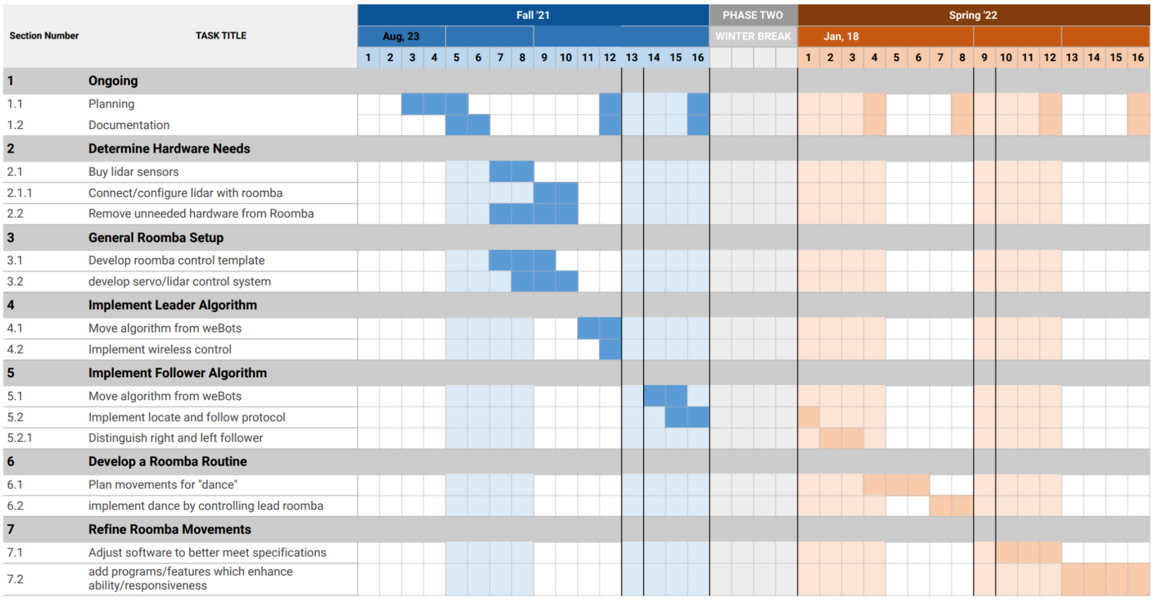
2.2 Task Decomposition

1. Ongoing - Documentation
    a. Weekly team meetings
    b. Project Documentation
2. Determine Hardware Needs
    a. Buy lidar sensors
    b. Connect lidar to Roomba
    c. Remove unneeded hardware from Roomba (IR and Sonic Sensors)
3. General Roomba Setup
    a. Develop template for general roomba control
    b. Develop control systems for lidar and servo
4. Implement leader robot algorithm
    a. Move algorithm from simulated weBots to classroom Roomba
    b. Implement wireless control of leader
5. Implement follower robot algorithm
    a. Move algorithm from simulated weBots to classroom Roomba
    b. Implement locate and follow protocol for roomba
    c. Setup system to distinguish between right and left follower
6. Develop a Routine for Roombas
    a. Plan movements for a “dance” that the Roombas follow
    b. Implement dance by only controlling the lead roomba
7. 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

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Roomba control template is coded and configured with the servo and lidar sensor
    - Roomba can set servo position within 2 degrees of error
    - Lidar sensor can be read by Roomba and is accurate within 1 inch
    - Combo servo and lidar sensor can identify a post representing a roomba
- Lead roomba can be operated wirelessly
    - Lead Roomba will follow the preset dance routine
    - The lead roomba can be controlled by a user
- Followers will follow a leader with less than 15% deviation from the prescribed following distance.
    - Members of the swarm can observe the distance between them and their leader
    - Members of the swarm can observe the angle between them and their leader
    - Members of the swarm can adjust accordingly to follow a leader at an acceptable distance
    - Members of the swarm can adjust accordingly to follow a leader at an acceptable angle
- The swarm can reliably move in formation without falling more than 7 inches out of place.
    - A swarm member can distinguish other members between environment objects
    - A swarm member can observe distances between one another
    - A swarm member can adjust to fit formation based on angle and distance from one another
- Develop Roomba routine that uses sound/song as input to control the swarm
    - The swarm leader will listen/play/know the song and move accordingly
    - The swarm followers will follow the lead roomba without knowledge of the sound

2.4 Project Timeline/Schedule



Section Number	TASK TITLE
1	**Ongoing**
1.1	Planning
1.2	Documentation
2	**Determine Hardware Needs**
2.1	Buy lidar sensors
2.1.1	Connect/configure lidar with roomba
2.2	Remove unneeded hardware from Roomba
3	**General Roomba Setup**
3.1	Develop roomba control template
3.2	develop servo/lidar control system
4	**Implement Leader Algorithm**
4.1	Move algorithm from weBots
4.2	Implement wireless control
5	**Implement Follower Algorithm**
5.1	Move algorithm from weBots
5.2	Implement locate and follow protocol
5.2.1	Distinguish right and left follower
6	**Develop a Roomba Routine**
6.1	Plan movements for "dance"
6.2	implement dance by controlling lead roomba
7	**Refine Roomba Movements**
7.1	Adjust software to better meet specifications
7.2	add programs/features which enhance ability/responsiveness

The Gantt chart starts at the beginning of the 1st semester, but our work is scheduled to start on week 3, that is around the time we got our group assignments and started discussing the project. Determining Hardware Needs starts on week 7 of semester 1, this stage will help us gather the equipment needed as well as remove excess, after this stage we will have a roomba with all the equipment needed to complete the project. Also starting on week 7 of semester 1 is General Roomba Setup. This stage will configure the roombas with base code that will assist the algorithm implementation later on, this stage will also develop a control system for the servo and lidar to scan the area. The Implement Leader Algorithm stage involves moving the previously built algorithm from weBots to the physical roomba and setting up wireless control, this stage begins on week 11 of semester 1 and ends right before fall break. Implement Follower Algorithm starts right after fall break on week 14 and goes until after winter break to the 3rd week of semester 2. This stage will implement the follower algorithm from weBots along with protocols that allow them to join the swarm. After this task we will have a basic swarm built. On week 4 of semester 2 the Develop a Roomba Routine stage begins and will take until spring break to build and test a dance controlled by the lead roomba. The last stage starts after spring break. The Refine Roomba Movements stage works to improve the functionality of the Roomba to have more accurate control, as well as expand beyond that of the original definitions, allowing us to use client input to build outside the original scope.

2.5 Risks And Risk Management/Mitigation

- Determine Hardware Needs
    - Hardware incompatibility
    - The hardware we decided are needed are modeled from the weBots project, and there may not be a real world parallel for the part.
    - There are workarounds that may exist to still make the system work.
    - Risk probability: 0.2
- Implement follower robot algorithm
    - Algorithm Efficiency
    - An algorithm is required to process sensor data, and decide how to move the bots.

    - The algorithm may not be efficient enough to calculate a follower bot's next move, causing it to get further and further away from the leader.
    - Risk probability:0. 45

2.6 Personnel Effort Requirements

Hours	Task
60	Ongoing - Documentation
15	Determine Hardware Needs
30	General Roomba Setup
30	Implement Leader robot algorithm
40	Implement follower robot algorithm
60	Develop Roomba Routines
Remaining Time	Refine Roomba Software and Movements
235 +	Total:

2.7 Other Resource Requirements

- At least 3 roombas outfitted with the technology used in CPR E 288
- Wifi chips to operate these remotely
- The LiDAR sensors that were used in the simulation

3 Design

3.1 Design Context

3.1.1 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 Defence 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

3.1.2 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.

Defence Systems: Defence 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 others locations in order to prevent vehicle collisions.

3.1.3 Prior Work/Solutions

We inherited our project from a senior design group last year, they started this project by building a virtual simulation of the Robot flock. The previous group implemented a design to allow the bots to organize into a flock with each other using only the sensors on the bots. The simulation code contains most of the logic for our flock to work, it lacks the design to be easily modified to our physical application but is structured and documented well which will make it simple to rewrite for our design.

Previous work has been done on flocking in robotics. Prior research has gone as far as fixed wing flocking examples(CITATION). Some of these robots can communicate amongst themselves, but they do offer insight issues others have run into before when implementing flocking. There are also examples of algorithms which have been developed for flocking behaviour between autonomous robots (CITATION). This previous work has given the field a base of understanding which we can pull from in our project.

https://arxiv.org/ftp/arxiv/papers/1310/1310.3601.pdf - Previous autonomous robot flocking research/algorithm

https://ieeexplore.ieee.org/document/6095129 - fixed wing flocking example

3.1.4 Technical Complexity

Our project includes different subsystems, including: LIDAR controls, Servo Control systems, Cliff and Edge detection systems, Wall/Bump Systems and potentially more, if we, as a group, decide that we need different equipment to complete our task. Implementation of these systems systems will be a challenge of its own, as there is no guarantee that they will interact well with each other.

After that we will test the logic of the previous team on the physical application, and evaluate if it will fulfill our needs. Our current plan is to design a robot template that can be implemented for any type of physical application allowing the algorithms we create to be reused for other platforms. We will start by moving their C code into C++, as an Object Oriented design for the template will create a better structure for our project.

3.2 Design Exploration

3.2.1 Design Decisions

So far we have made two major decisions: Convert the previous year's C code into C++ and Exchange the previous year's Direction LIDAR/servo with an Omnidirectional LIDAR. We are not completely sure on how the Dance implementation will work. Due to that we are holding off on making any large decisions regarding that aspect of the project.

3.2.2 Ideation

In regards to the Omnidirectional LIDAR, we went through a couple of different design options. Including:
1. Keeping the previous project's directional LIDAR

2. Using the on board IR and Sonic Sensors

3. WIFI strength triangulation

4. Cameras and vision processing

We chose the OmniDirectional LIDAR as it seems to provide the best aspects of most of the previous design considerations. As the LIDAR does both the Sonic and IR tasks in one sensor. And eliminates the need for a rotating servo that the directional LIDAR needs. The WIFI option would not be accurate enough for our needs, and the cameras would need more environment setup then the other options.

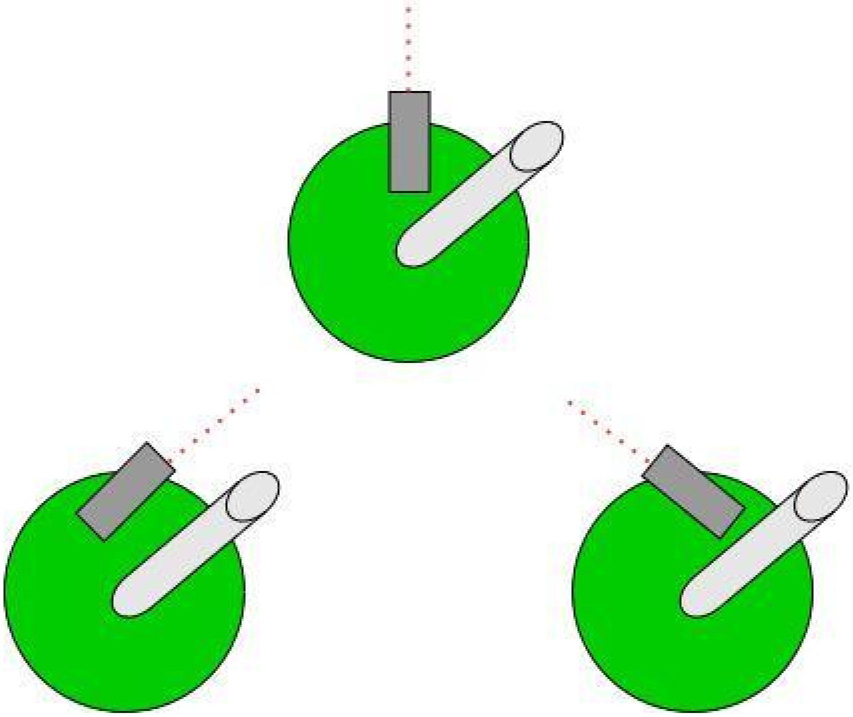
3.2.3 Decision-Making and Trade-Off

The WIFI and Camera with vision processing would not be viable due to the low accuracy those systems would provide. As well as the cameras would need extra environment setup adding to its complexity. Both types of LIDAR provide greater accuracy and scannable area compared to the Sonic and IR sensors which lack the technological advances of LIDAR. Finally we settled on the omni-directional LIDAR due to the scanning area. This will allow the sensor to easily track both the lead bot and obstacles without extra complexity on our part.

	Simplicity	Accuracy	Scan Area	Price	
WEIGHT	1	3	3	2	
directional LIDAR	2	4	3	3	29
IR & Sonic	6	3	2	1	23
WiFi Tri	1	2	4	4	27
Camera & Vision	2	1.8	2.5	3	20.9
Omni-directional LIDAR	3	3.5	6	4	39.5

3.3 Proposed Design

3.3.1 Design Visual and Description



The diagram above shows the organization of our Roomba flock-swarm. The dark grey boxes represent our omni-directional LIDAR. To allow the LIDAR to locate other bots we will use long pvc pipes, represented as light grey cylinders above, as reliable markers for the sensors to identify. Since they will be the same size we can configure the robot to reliably identify that marker and track it for relative position.

3.3.2 Functionality

Determining a follower's next move will be a proportion of the follower's distance to the leader and the relative angle between the follower's heading direction and the leader's detected angular position. If the follower reads the leader as moving outside the desired angular position, the follower will alter its direction to maintain a lock on the leader, constantly modifying the left and right speeds to maintain distance and angle from the leader.

To allow followers adequate headroom to catch up to a maneuvering leader, the leader's maximum allowed speed should be around half to three-quarters that of the iRobot Create's top speed. This will ensure that followers are able to catch up to the leader if it begins a maneuver like a turn. Since the follower on the outside edge of the turn must travel a longer distance to maintain relative position it needs to move faster than the leader to do so. This helps ensure that the follower's movement wouldn't require it to exceed the maximum possible speed.

3.3.3 Areas of Concern and Development

The biggest area of concern is getting the LIDAR to accurately identify and measure the distance and direction of the leader. The inherited project used a direction LIDAR on a servo in which they controlled the angle of the sweeping scan following the leader, our proposed solution will use an omni-directional LIDAR which will look around in all directions to both follow the leader and help avoid obstacles. Since the previous project used a simpler LIDAR solution we will have to greatly modify their code to work with our more elaborate system. It is also new technology for any of our group members to use with a Roomba so it may take more time to properly connect. Early testing and a strong design will help us develop a solution that will be accurate and minimize the changes needed to the inherited code.

NOTE: The following sections will be included in your final design document but do not need to be completed for the current assignment. They are included for your reference. If you have ideas for these sections, they can also be discussed with your TA and/or faculty adviser.

/*

3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

3.5 Design Analysis

- Did your proposed design from 3.3 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate over the design?

3.6 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

*/