SHINERBOTS: A ROBOTIC SWARM NAVIGATION PLATFORM INSPIRED BY THE GOLDEN SHINER FISH

BY

ENYU LUO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Assistant Professor Grace Xingxin Gao

Abstract

This thesis describes a swarm robots navigation algorithm inspired by the golden shiner fish. The goal of this algorithm is to have the swarm of robots navigate to their destination with minimal requirements on each robot. There is little sensing complexity required. Each robot only needs to sense information at its current location without the need to understand the whole environment. In addition, it only needs to sense its neighbors in a limited radius. Furthermore, each robot requires minimal computation as it operates on basic rules on what it senses, instead of performing localization and path planning like traditional robotics.

The algorithm proposed in this thesis is based on simple rules such as modulating speed according to the environment, as well as staying close to neighboring robots. Simulation results show that the swarm of robots converge to the desired region with this algorithm.

A swarm robotic platform was designed and built to experiment with this algorithm. It is named Shinerbots due to the inspiration taken from the golden shiners. Each robot is 40.6 mm in diameter, actuated by vibration, senses ambient light using a photo diode, and senses neighbors using infrared. Scalable ways of charging and control of the swarm are achieved by a charging plate, as well as relayed system messages.

A total of 85 robots were built. Experiments on the swarm behavior were conducted on a 4 ft by 4 ft platform, using an overhead digital projector to create the light environment that the Shinerbots are supposed to navigate in. The Shinerbots were able to swarm to a target region with minimal sensing and control.

Acknowledgments

I would like to express gratitude to my advisor, Assistant Professor Grace XingXin Gao, for the support of my study and research projects. I would like to thank Xinhui Fang and Kwok Bun Cheng for soldering and assembling robots for this project, as well as Alexandra Bacula and Katie Caroll for calibrating and conducting experiments with the Shinerbots.

Contents

1. Introduction

Traditional robotic platforms rely on a sophisticated self-contained system to perform navigation [24]. The robot would have to determine its current location and destination, and perform path planning to traverse to the desired destination. This requires a wide range of sensors to provide awareness of its surroundings, which leads to dependence on significant processing power to digest the information and make decisions in a centralized system. There is a shift in paradigm in the past decade towards distributed systems [20], [7]. Instead of a single robot, a swarm of robots cooperate to achieve the same navigation goal. This reduces costs of robots as the complexity of sensors and processing is lowered on each robot. It also increases fault tolerances as the robot swarm basically becomes a system of many redundant sensors and will not be affected by a single point of failure as a single robot may experience.

Different types of distributed swarm navigation algorithms have been proposed. There are algorithms that maintain different types of formation, while avoiding obstacles and moving to a goal [2]. Other algorithms utilize artificial potential functions [14] and panel methods [17], as well as incorporating sliding-mode control to potential fields [9]. Stream functions [27], a potential field-based method, were also proposed. Some of the algorithms were designed for specific purposes like cooperative surveillance using micro-aerial vehicles [22], as well as transporting objects in a cooperative manner [1]. Most existing swarm robot systems still utilize complex sensing and computation resources on each robot. Robots were assumed to be able to measure environmental parameters like gradients, and also to communicate with all other robots. Sections 1.1 and 1.2 describe the inspiration and model of an algorithm that reduces requirements of individual robots of a swarm even further.

1.1 Golden Shiner Fish

The golden shiner is a type of fish that likes to be in dark areas. People originally thought that the fishes are able to sense light gradients, which tells them where to swim towards the dark areas, but Berdahl et al. [3] disproved this. A single fish alone in the tank is not able to swim towards the dark area. Instead, it swims in a random manner, modulating its speed according to the light intensity of its current location. It swims fast in bright areas, and slow in dark areas. As more fishes are added to the tank, they are able to school together and end up in the dark region.

1.2 Swarm Algorithm

This thesis proposes and implements a swarm navigation algorithm inspired by the behavior of the golden shiner. This algorithm is based on two aspects which govern the movement of each Shinerbot.

One is an environmental factor. Ambient light was used in the case of this thesis. Each robot modulates its own speed according to light intensity. The second aspect is the social factor, where robots want to stay with neighbors, mimicking schools of fishes. The generalized algorithm is shown below. More details to the algorithm will be explained in chapter 3.

$$\dot{x}_i(t) = \ env\big(x_i(t)\big) \cdot soc(x_i(t), \{x_j(t)\}, j \in \mathcal{N}_i(t))$$

There are several benefits to this distributed algorithm. Minimal sensing is required as a robot only needs to sense its current location and not the whole environment. Robots do not need to communicate and exchange intelligible information about the environment or directions. It is simple and places a very small computational burden on each robot.

1.3 Shinerbots

An 85-robot swarm platform is designed and built. These are ground-based robots that move via vibration. Each robot has an ambient light sensing photodiode and neighbor sensing capability using infrared signals. An RGB LED meant as a general purpose indicator is added as well. The robot is only 40.6 mm in diameter so as to require less workspace for the whole 85-robot swarm. Scalability issues are also considered in the design, which allows for mass simultaneous charging of the robots, and relayed system messages to remotely turn on and off the robots.

1.4 Summary of Contributions and Thesis Outline

The following are the contributions of this thesis:

- We propose a simple, distributed, and compelling algorithm inspired by the golden shiner fish. The algorithm requires only minimal sensing, no communications between robots, and low computational complexity.
- We designed and built an 85-robot swarm called Shinerbots. Each robot costs USD$20, with the size of 40.6 mm. Each robot is able to sense both distance and orientation of its neighbors in an inexpensive way. Scalability features are implemented with a swarm messaging system, as well as a charging plate for simultaneous mass charging.
- We conducted experiments to verify that the Shinerbots are able to navigate to a target region with minimal sensing and control using the algorithm proposed.

This thesis will provide a quick overview on current swarm robots in chapter 2, and then elaborate on the golden shiner swarm algorithm in chapter 3. The hardware and firmware design of the Shinerbots will be described in chapter 4. The experimental results will be presented in chapter 5. Chapter 6 concludes the thesis and outlines future work.

2. Existing Swarm Robotic Platforms

There are existing swarm robot platforms such as MarXbot, eSwarbot, Pi-swarm, e-Puck, Khepera, Jasmine, Kilobot, and Droplet. Their sizes and costs are summarized in Table 1 and photos are in figures 1-8.

Table 1 Size and Cost of Existing Swarm Robotic Platforms

Robot	Size	Cost (USD$)
MarXbot [4]	17cm diameter	
eSwarmbot [5]	12.6cm diameter	
Pi-swarm [12]	9cm diameter	Part cost of $100
e-Puck [1], [25]	7cm diameter	
Khepera [18]	7cm diameter	Retail price of $272
Jasmine [13]	2.7cm cube	Part cost of $112
Kilobot [21]	33mm diameter	Retail price of $113
Droplet [8]		Part cost of $50 without shell

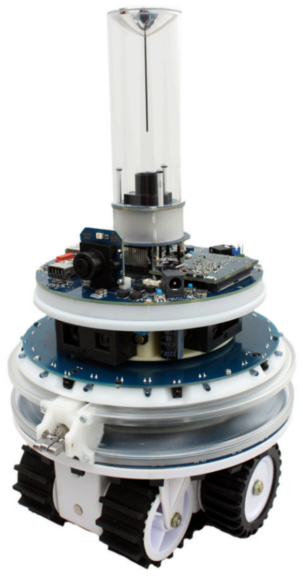


Figure 1: MarXbot

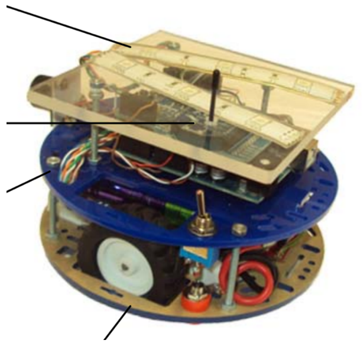


Figure 2: eSwarmbot

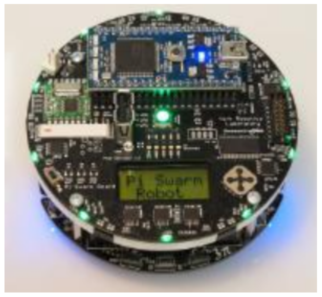


Figure 3: Pi-Swarm

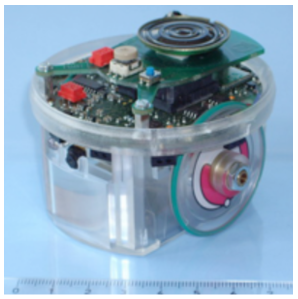


Figure 4: e-Puck

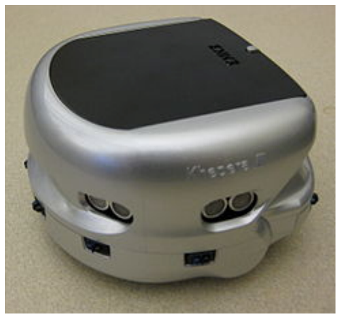


Figure 5: Khepera

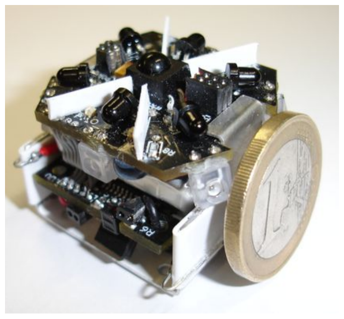


Figure 6: Jasmine

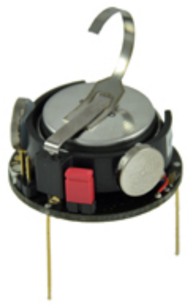


Figure 7: Kilobot

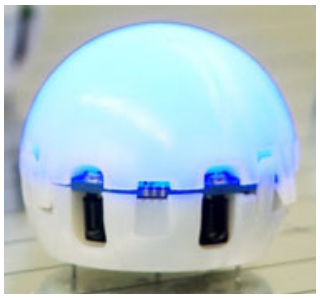


Figure 8: Droplet

The MarXbot[4] is a sophisticated track and wheeled robot consisting of 6D IMU, multiple infrared sensors, Lidar, 2 cameras, Bluetooth, Wifi, and an ARM based processor running Linux. The eSwarmbot [5] is a wheeled robot with quadrature encoders and ultrasonic rangefinder, running on an Arduino microcontroller. The Pi-swarm [12] is another wheeled robot running on ARM Cortex M3 based microcontroller. It has multiple LEDs, multiple infrared sensors, 9D IMU, LCD display, switches, and a radio frequency (RF) transceiver. E-puck [1], [25] has 8 infrared sensors, 2 stepper motors for the wheels, a color camera, a speaker and 3 microphones. Khepera [18] is a wheeled robot with 8 infrared sensors. Both Kilobot [21] and Droplet [8] are smaller robots that move using vibration motors, and cost less compared to the other larger wheeled robots. The Kilobot has 1 infrared transmitter and 1 infrared receiver to provide distance measurement to neighbors without any bearing. The Droplet has 6 pairs of infrared receivers and transmitters spread around its perimeter to sense distance and bearing to neighbors.

The first few robots described are big, based on wheels, and most of them have sophisticated sensors and processing power, which tend to be expensive. While working on swarm robots, larger robot size will lead to needing an even larger workspace for the robots. As for the swarm algorithm that will be presented in the next chapter, sophisticated sensors and processors are not required. Those expensive robots with advanced features will only drive the total cost of the whole swarm research high. The Kilobot is an innovative design which uses vibration to move, and is only 33 mm in diameter. This allows for a manageable workspace size to operate the whole swarm. The Kilobot can only sense the distance to neighbors, but not their orientation. This may work well for some other swarm navigation algorithms, but not the golden shiner algorithm to be presented in this thesis. Chapter 4 will detail the design of Shinerbots built on top of the idea of Kilobots, allowing both distance and orientation of neighbors to be sensed, plus other minor improvements.

3. Shinerbot Swarm Navigation Algorithm and Simulation

This chapter describes a simple mathematical model which was inspired by the behavior of the golden shiner fishes, and proceeds to demonstrate it in simulation.

3.1 Modeling of the Golden Shiners

Each robot is modeled as a two-dimensional mobile autonomous agent, where $x_i(t) \in \Re^2$ is the location of agent i in the two-dimensional plane at time t. We model the movement of the robot by two factors. The first is an environment factor $env()$ which gives a positive scalar value. This modulates the robot's speed in relation to the type of environment stimulus that we want the swarm robots to navigate to. The second factor is a social factor $soc()$. This function gives a 2D vector of the direction in which the robots go. In this model, we want the robots to go towards their neighbors. Thus, the social factor depends on the location of robots $x_j(t) \in \mathcal{N}_i(t)$, where $\mathcal{N}_i(t)$ defines the set of robots within the neighborhood at time t. In this case, robots within the neighborhood correspond to robots whose locations are within the sensing range of robot i. The generalized model is as follows:

$$\dot{x}_i(t) = env\big(x_i(t)\big) \cdot soc(x_i(t), \{x_j(t)\}, j \in \mathcal{N}_i(t))$$

From this equation, we see that the decision-making of each robot is broken up into two simple rules that obtain inputs from an environment sensor, as well as a neighbor sensor. The environment factor enables convergence to an optimal point, while the social factor enables collective intelligence and expediting of convergence.

Developing the model further to represent the golden shiners, we take the environment stimulus as light intensity. Recall from section 1.1 that the golden shiner fishes loves to be in the dark. They swim slowly in the dark, and faster in brighter places. Let $f(x)$ represent the light intensity of location x, and let $g()$ represent the light sensitivity of the robot. Thus, the environment factor $env(x_i(t))$ becomes $g(f(x_i(t)))$. This function modulates the robot's speed according to light intensity. This corresponds to the observations made by Berdahl et al. [3]. Here, we assume that each robot has the same sensitivity to light $g()$, where $g()$ is a strictly increasing, bounded function such that g(0) = 0.

Next, we look at the social factor. The golden shiners always school together. Thus, we model the social factor as a desire to move towards its neighbors. Mathematically, we represent the vector of the robot's motion to point towards the centroid of its neighbors (mean location) within the robot's sensing range.

In this application, there is no localization system. No robot knows where it is in a global coordinate system, nor can it communicate its location to neighbors. The vector that represents the direction towards the centroid is solely based on the relative position of each neighbor to the robot itself, where the relative position is defined as $x_j(t)$ - $x_i(t)$.

Putting it together, the golden shiner swarm algorithm is as follows:

$$\dot{x}_i(t) = g\left(f\left(x_i(t)\right)\right) \sum_{j \in \mathcal{N}_i(t)} (x_j(t) - x_i(t)), i \in [n] \tag{3.1}$$

where $[n]$ denotes the set $\{1, 2, \ldots, n\}$ of all robots.

3.2 Modeling with Added Randomness

In our experiments with the Shinerbots, there are two scenarios where the robots do not all converge to the dark region. The first scenario is where some Shinerbots are isolated. This is the case when a single Shinerbot does not have any neighbors within its sensing range. According to the social factor as defined in equation 3.1, the isolated robot does not move until another robot comes into its neighborhood. There are algorithms presented in [19] and [15] that preclude the isolated agents problem by maintaining connectivity to neighbors; however, these algorithms increase computational complexity.

The second scenario is where a sub-group of robots remain stuck in a region for a long time. This happens when the net gradient of light intensity in the local neighborhood is zero. In this case, all robots will want to move towards the same centroid over and over again, resulting in a zero-net movement in that whole sub-group.

To overcome these two scenarios, a stochastic term is added to equation 3.1 as follows:

$$\dot{x}_i(t) = g\left(f\left(x_i(t)\right)\right) (\sum_{j \in \mathcal{N}_i(t)} (x_j(t) - x_i(t)) + Z_i(t)), i \in [n]$$

where $Z_i(t)$ is an additive two-dimensional white Gaussian noise. This added random noise removes the undesired equilibria. It prevents isolated robots from staying still, increasing its chances of joining a sub-group of robots. The added random movement also prevents sub-groups from getting stuck in a neighborhood with zero net gradient.

3.3 Simulation of the Model

A simulation was done to illustrate the behavior of the whole swarm according to the algorithm as described above. In figure 9, the red circle represents the point of zero light intensity, which denotes the desired destination for the robots. The robots are randomly placed to the right of the destination initially (time t = 0). The green circle represents how far the whole swarm of robots is to its destination. The center of the green circle is the location of the red circle, with the radius corresponding to the distance from the centroid of the whole swarm to the destination. At t = 15 s, the robots are observed to have split into sub-groups. This makes sense as robots will only try to move to the centroid of their local neighborhood, without knowledge of the global centroid. At t = 80 s, the sub-groups have moved closer towards the destination, joining other sub-groups into one big group. Finally, even isolated robots managed to join the group at t = 180 s. The behavior at t > 30 s is made possible due to the added randomness. At t = 296 s, the green circle finally closed and all robots have arrived at the destination.

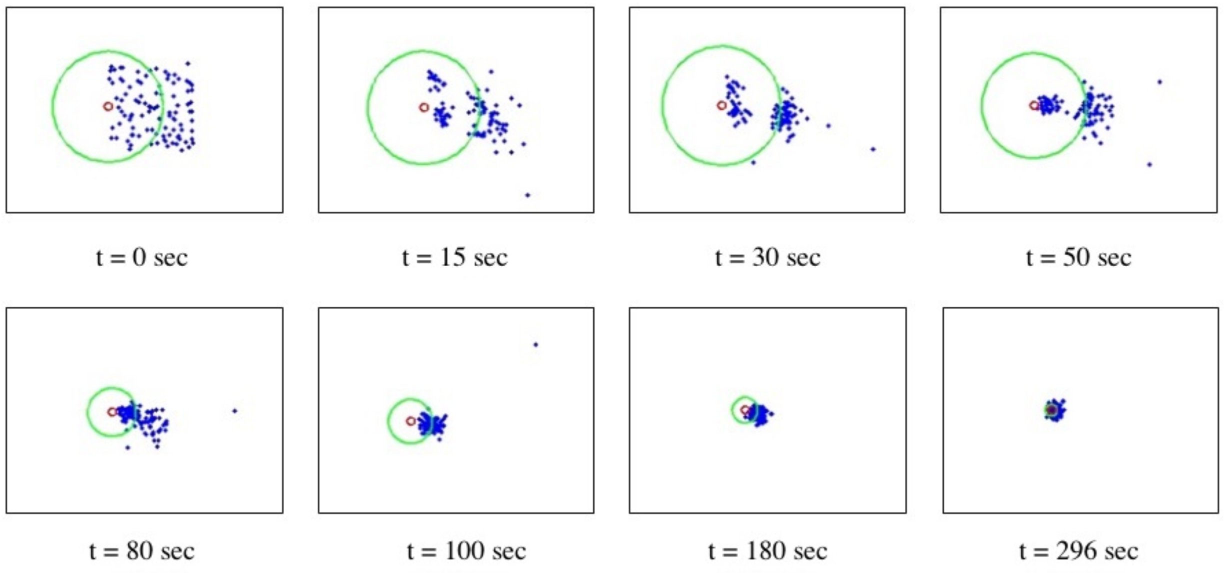


Figure 9: Simulation Results

4. Design and Implementation of the Shinerbot Platform

A robotic swarm system is designed to perform the swarm algorithm as described in the above section. Each robot needs to fulfill simple requirements for this algorithm, namely mobility, neighbor sensing, and light sensing. Neighbor sensing requires the distance to and orientation of each neighbor in a limited sensing radius. No robot need sense other robots far away. Light sensing only requires the measurement of light intensity of its current location, and not light gradients of its surroundings.

Eighty-five robots are built to experiment with this swarm algorithm. On top of the swarm algorithm requirements, additional scalability requirements will be needed to help manage the 85-robot swarm. Cost per robot is a factor to be reduced as it will impact total cost of the system when it is scaled up to 100 robots. Size of a single robot matters as well, since it impacts the space needed to run experiments on an 85-robot swarm. Other factors include management procedures like charging, and commanding of the robots. These requirements will be addressed later in the chapter.

Figure 10 is a block diagram of the various sub-modules in a single Shinerbot. The microcontroller that controls the robot is a MSP430G2553 from Texas Instruments. Each of the other sub-modules will be explained in the following sections of this chapter.

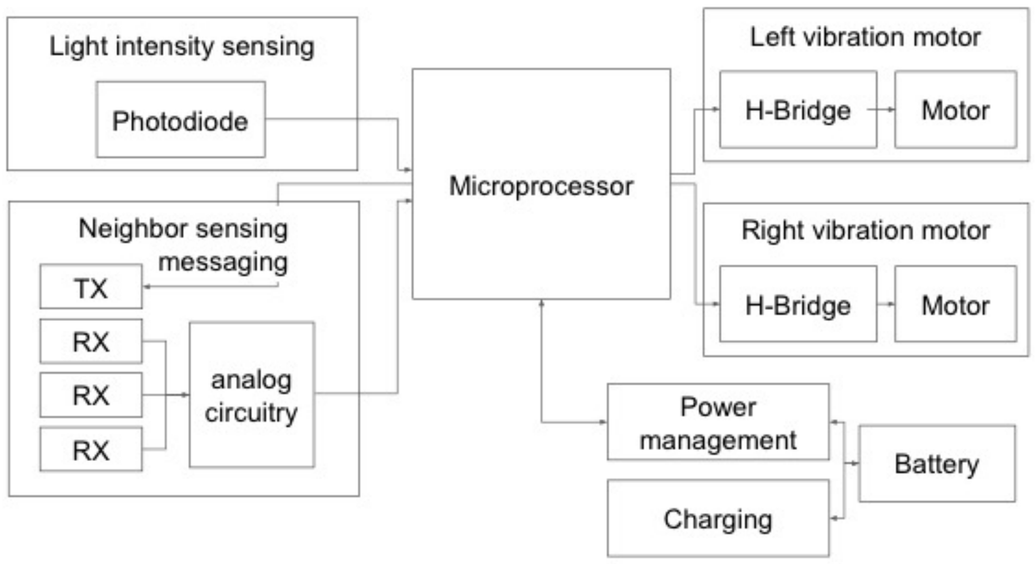


Figure 10: Hardware Block Diagram of a Single Shinerbot

4.1 Mobility

A common means of mobility is to use motors and wheels. While it is possible to make custom miniature motors and wheels, they are prohibitive in cost when making only 100 robots. Off-the-shelf motors and wheels tend to be big (> 3 cm diameter) and cost at least $5. That forces the size of the robot to be at least 10 cm in diameter. To bring the size and cost down, Shinerbots were designed using legs, and move by vibration [26]. There are three legs, which are basically rigid long pins. Vibration is generated by two eccentric rotating mass (ERM) motors placed on each side of the robot. These motors are common in many electronic devices like cell phones, making them cheaply available in bulk.

An H-bridge circuit is used as an output driver to each motor. Pulse width modulation (PWM) is used to vary the power supplied to the motor. Forwards and backwards movement are made by driving both motors in the same direction. The robot rotates to the left or right when both motors are driven in opposing directions.

The motion of the robot can only be controlled consistently with a specific amount of power supplied to the motors. This is because the rotation of the eccentric mass in the motor has to be matched to the frequency at which the robot vibrates vertically to allow for the correct interaction of forces by the eccentric mass, and friction between the legs and the floor. As a result, the speed of the robot's movement is fixed. Duty cycling is done to vary the speeds of the robot to perform the swarm algorithm.

4.2 Neighbor Sensing

The Shinerbots uses infrared (IR) to sense its neighbors. Each robot transmits infrared signal from the bottom-middle down into the ground as shown in figure 11. The infrared then bounces off the ground towards any neighboring robots in all directions. This is due to the beam pattern of the IR transmitter. A neighboring robot, with receivers placed underneath, receives the IR signal and is able to estimate distance based on a received signal strength indicator (RSSI). To estimate the orientation of the neighbor, each robot has three receivers placed 120 degrees apart, on its perimeter as shown in figure 12. This allows three distances to be obtained, which are then used to compute the orientation.

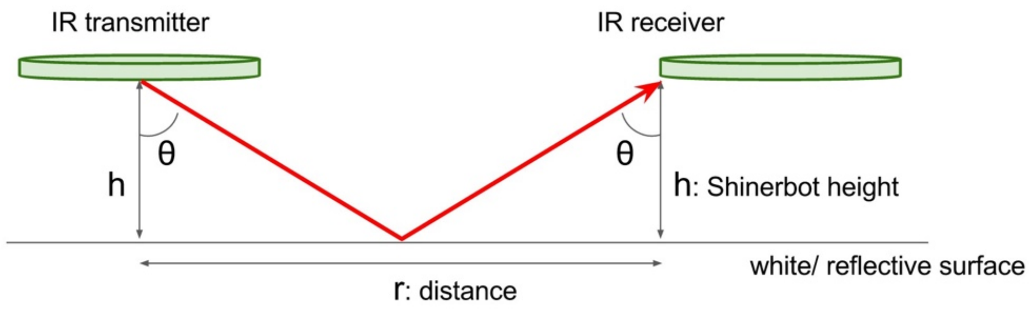


Figure 11: Geometry of the IR Signal Between Two Neighboring Shinerbots

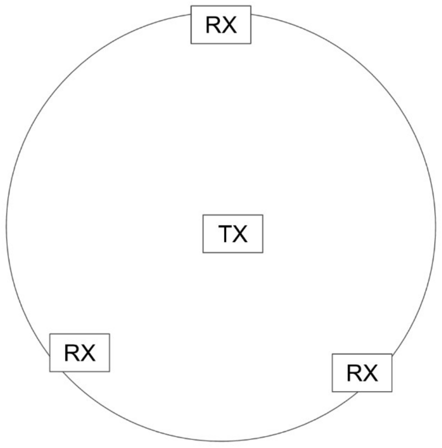


Figure 12: Placement of 3 Infrared (IR) Receivers and 1 IR Tranmitter on the Underside of the Shinerbot

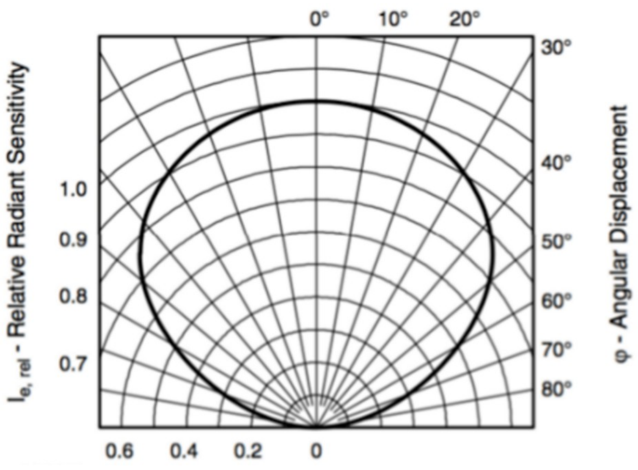


Figure 13: IR Beam Pattern (Semiconductors, 2011) [23]

A factor of 1/r is used to estimate the signal strength at distance r between neighbors due to the spread of the signal in the 2D plane. A factor due to the vertical dimension is considered as well. Both receivers and transmitters have the same beam pattern as shown in figure 13. A cosine function is used to model the signal strength exiting the transmitter, as well as entering the receiver. Thus, cosine squared becomes the factor due to the vertical dimension. An addition factor k lumps various factors such as the transmit power, receiver gain, and analog-to-digital converter (ADC) quantization range. The RSSI is then modeled with the following equation:

$$RSSI = \frac{k}{r} \cdot (\cos\theta)^2 = \frac{k}{r} \cdot \frac{4h^2}{4h^2 + r^2}$$

where θ is the angle between the signal path and the vertical, and h is the height of the Shinerbot. The theoretical RSSI is compared against the measured RSSI across distances (figure 14) and is found to be inaccurate due to manufacturing variances, as well as multipath effects when robots are very close together, which was not modeled.

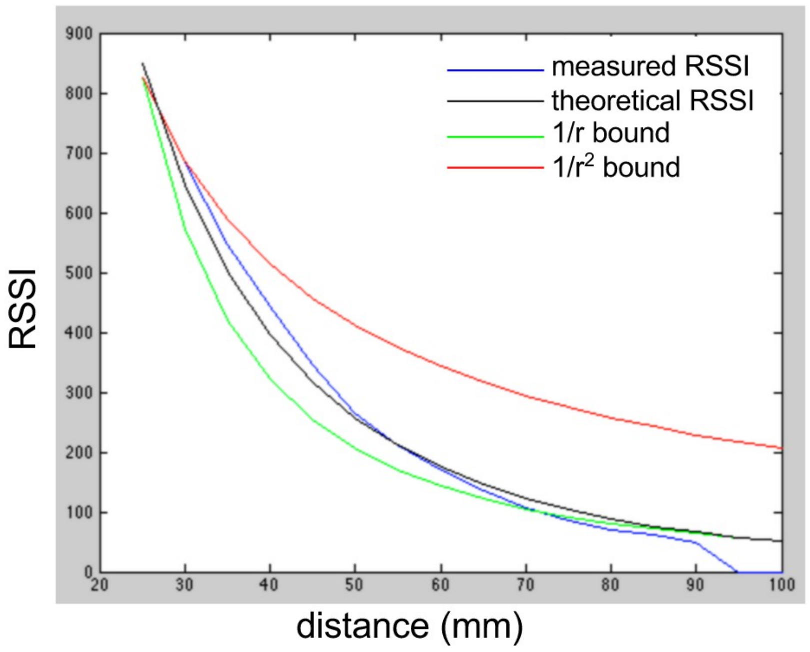


Figure 14: Measured and Theoretical RSSI against Distance

To solve this problem, a calibration-based method is used instead to estimate a more accurate distance from RSSI. A calibration sequence is run after a robot has finished assembly. This consists of moving the robot from a transmitting robot in steps of 5 mm for all three of the IR receivers. Linear interpolation is then used to compute the distance for RSSI values in steps of 64 ADC units. The distance values are then stored as a lookup table in flash memory. Figure 15 shows the measured RSSI values during the calibration sequence, compared to the interpolated RSSI values of estimated distances. The reason for storing distances in 64 ADC units is for efficient lookup at each instance of estimating neighbor distance.

Bit shifting of 6 bits and masking is done to obtain the two nearest distances in the table, and a linear interpolation is used to estimate the actual distance using the lowest 6 bits of the ADC value.

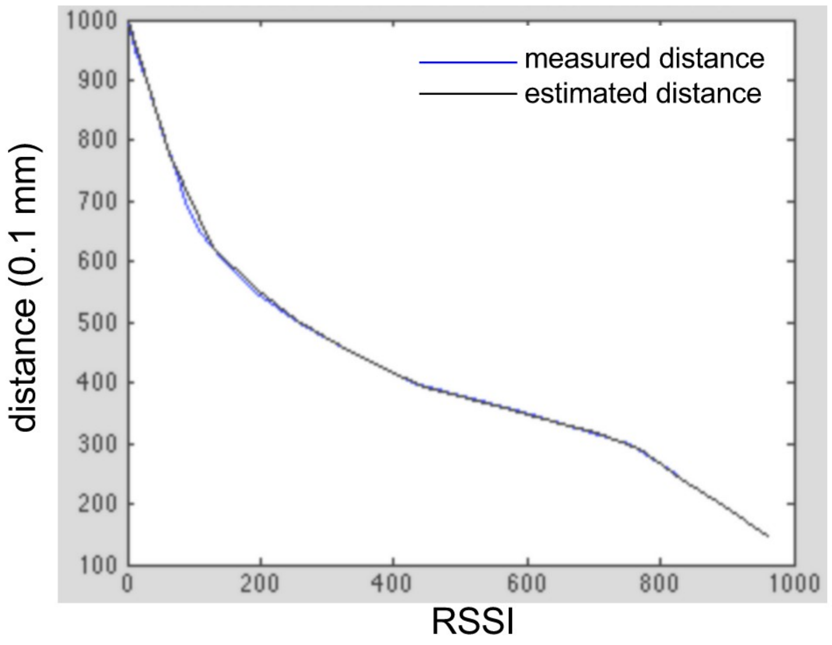


Figure 15: Distance Estimation from RSSI Measurements Using Interpolation

Once three distances are obtained from the three receivers, trilateration is performed to compute the relative position of the neighboring transmitting robot. This is given by the equations below:

$$x = \frac{r_A^2 - r_B^2}{2l_3}$$

$$y = \frac{r_C^2 - r_D^2}{2(l_1 + l_2)} + \frac{l_1 - l_2}{2}$$

$$r_D^2 = \frac{r_A^2 + r_B^2 - \frac{l_3^2}{2}}{2}$$

where variables are defined as in figure 16.

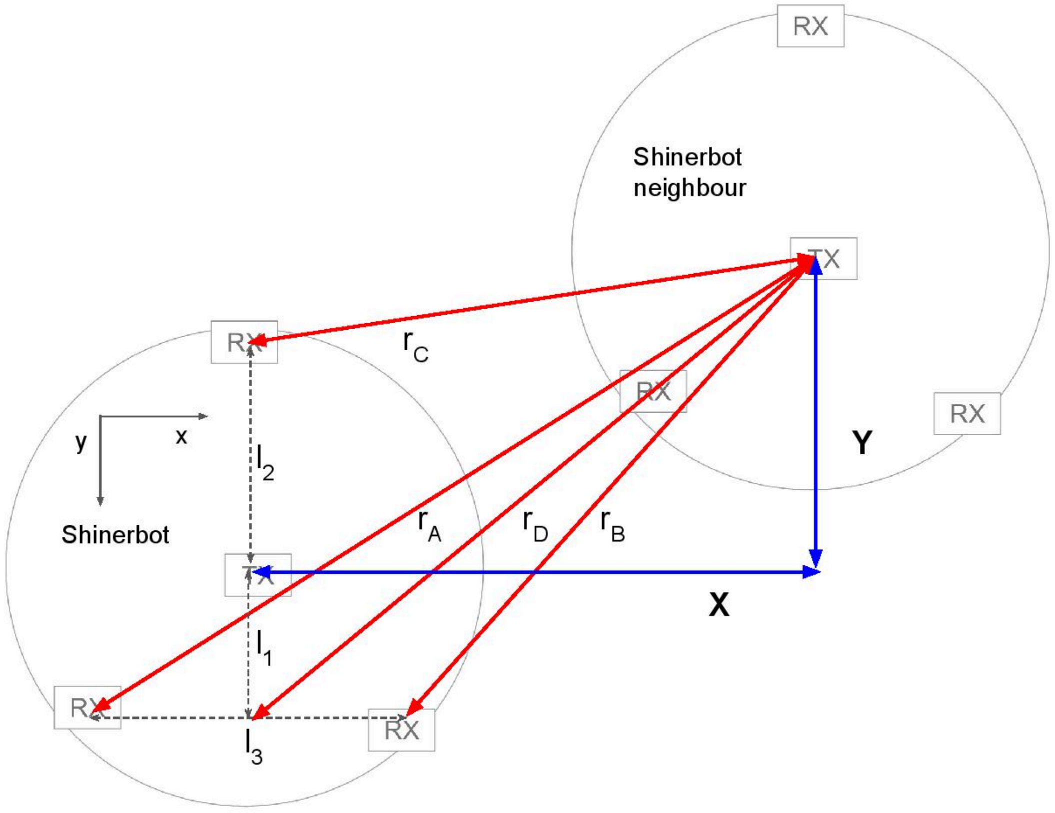


Figure 16: Coordinates of Neighboring Robot Using 3 Distances

For this method of neighbor position estimation to work well, only one robot can transmit at a time so as not to interfere with the RSSI measurements. Thus, a simple communication protocol based on carrier sense multiple access with collision avoidance (CSMA-CA) is implemented to ensure that.

4.3 Communications

Although the golden shiner fish swarm algorithm does not require robots to communicate, a simple communications system is implemented to provide a generic swarm robotics research platform for future algorithms. In addition, the communications system allows for system messages to control the entire swarm, and solves the problem of having multiple robots transmitting at the same time, interfering with the neighbor distance estimations.

The physical layer of this communication system uses the infrared data association (IrDA) standard [10]. Each robot transmits a fixed length packet at regular intervals. The packet consists of the destination ID, RSSI byte, source ID, 5 data bytes, and a CRC. A simplified CSMA-CA algorithm is used to enable multiple robots to transmit in the same area. This simplified version of CSMA-CA is adapted from wireless LANs

[10]. Clear to send (CTS) and ready to send (RTS) are not implemented since all messages act like a regular broadcast for robots to estimate distance.

Each robot transmits a packet at regular intervals of about 200 ms. This allows all its neighbors in the region to detect its presence and compute its relative position. The RSSI byte in the packet structure is basically a fixed value of 0x55. This ensures an alternating sequence of 1 s and 0 s to be received. RSSI is then computed by taking the difference of ADC values from the positive and negative peaks of the pulses. The CRC byte detects if the packet received is valid. If the packet was invalid, the distance calculation of that packet would be ignored.

Although a generic communications protocol is implemented, the data bytes of the packet are unused in the implementation of the golden shiner fish algorithm. This algorithm does not need intelligible information to be communicated among robots. It basically acts only as a ping to enable distance estimations in this case.

4.4 Scalable Charging

Charging every robot with a power cable plugged into each robot is tedious and time consuming. This is not scalable as the swarm gets larger. A charging pad is designed such that robots can be charged simultaneously by placing up to 49 robots on the pad. Each robot charges itself via the legs. The legs are connected to the charge management IC via a full bridge rectifier. As long as 6 V is applied across any of the two legs, the robot will be able to charge. The charging pad is a copper clad FR4 with alternating strips of copper as shown in figure 17. The width and gap of the strips are such that a robot placed at random will always be guaranteed to have two legs on an alternating strip. 6 volts is then applied to the alternating strips of copper. The copper is tin coated to prevent oxidization. Upon placing the robot to charge, the charge management IC status would signal the microcontroller to bring the robot to sleep. This prevents the robot from unwanted vibrations, which will cause unreliable charging. A small red light-emitting diode (LED) turns on when it is charging.

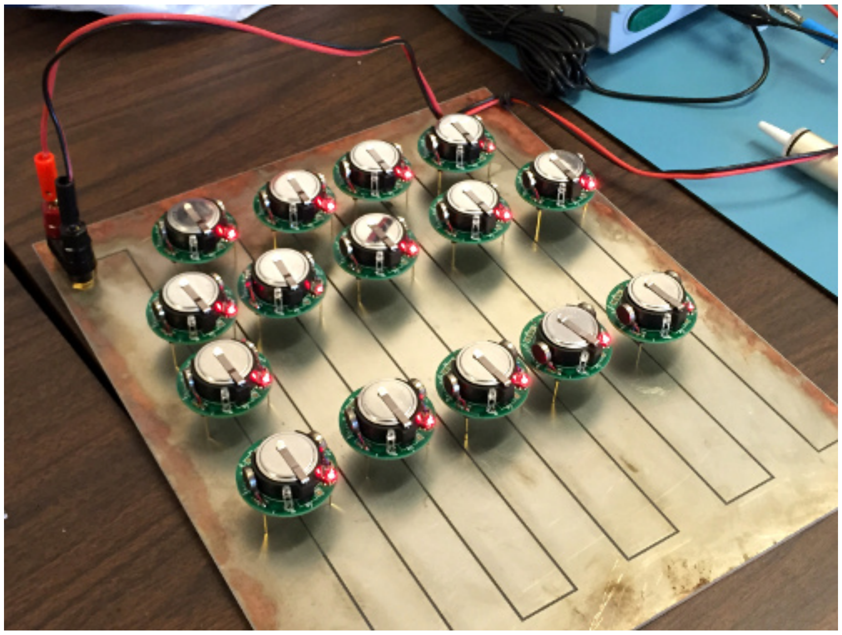


Figure 17: Shinerbots Charging System

4.5 Other Hardware Submodules

A general purpose RGB LED is included in the design. It is driven by PWM signals to each color channel. This is meant to be a general-purpose indicator, and can be programmed to display any color or intensity according to the application, or experiment being run. Light sensing was mentioned as a basic requirement for the golden shiner fish algorithm. This is achieved using a photo diode for ambient light.

4.6 The Physical Printed Circuit Board

The mechanical body of the robot is basically the printed circuit board (PCB) itself. It is a circular board, with 40.6 mm diameter, designed in KiCad. This PCB is a four-layer design with the top layer for digital signals (figure 18), layer 2 for digital ground, layer 3 for analog ground, and the bottom layer for analog signals (figure 19). The analog and digital signals were separated in such a way because the analog traces have to span the entire board. This is due to the IR receivers needing to be 120 degrees apart, near the perimeter of the board. The ground planes in the middle layers improve signal integrity by shielding signals from crosstalk between the digital and analog signals. The analog and digital grounds are connected at one point near the ground of the microcontroller ADC.

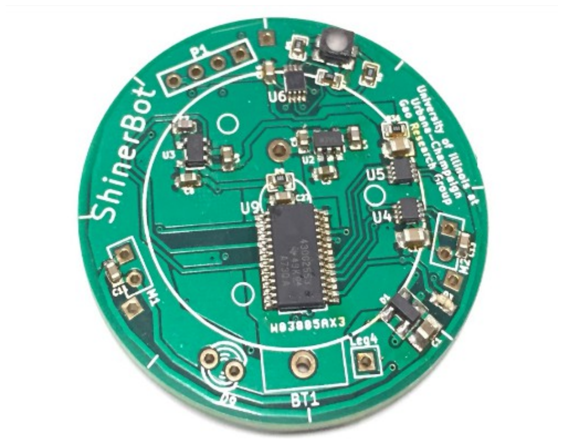


Figure 18: Shinerbot PCB Top

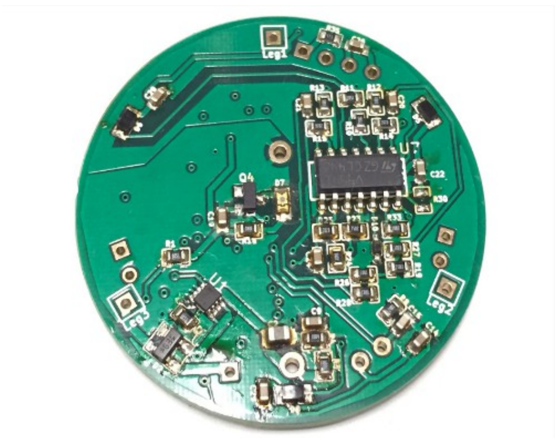


Figure 19: Shinerbot PCB Bottom

4.7 Firmware

The firmware is written in a layered manner as shown in figure 20. MSP430 at the lowest level is basically a header file providing low-level access to registers in the microcontroller. The I/O Drivers layer provides an abstraction to low level tasks. The Shinerbot System layer handles the main operations of the robot. The Application layer consists of code that governs the high-level behavior of the robot.

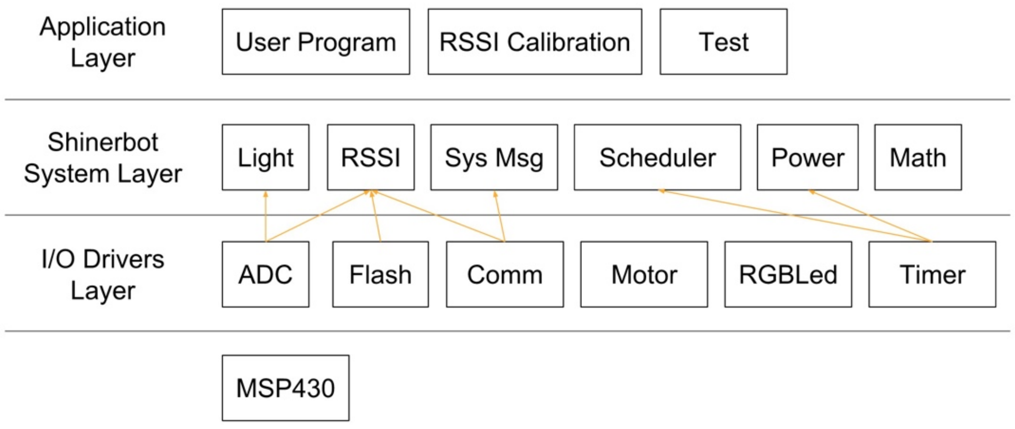


Figure 20: Shinerbot Firmware Block Diagram

4.7.1 I/O Drivers Layer

This layer provides low-level interfaces to the hardware of the robot. ADC driver provides functions to control the ADC peripheral on the microcontroller. These include initialization, triggering of single or multi-channel conversions, and call-backs for completion. Flash provides functions to erase and write to specific banks of flash memory. Comm provides functions to start a packet transmission, and to read packets that have been received. Comm implements a buffer to store incoming packets to be read. Motor provides an abstraction to the PWM peripherals and controls output power to each motor. It provides functions to move forward, backward, left, and right. RGBLed drives the RGB LED according to the values fed to it, in ranges of 0 – 255 for each color channel. Timer provides a 1 ms tick for the entire system.

4.7.2 Shinerbot System Layer

The code in this layer does most of the operations related to making the Shinerbot work. They provide a set of APIs for the Application layer to command the Shinerbot's high-level behavior. The Light module is basically an abstraction to the ADC driver to read values from the photodiode ADC channel. The RSSI module is in charge of estimating a neighbor's relative position. It takes ADC values from the three IR channels, computes the RSSI, estimates distances, and computes the position. Sys Msg is a module that interprets incoming system messages, relays the message, and performs the system command. This will be elaborated in section 4.8. Scheduler is a simple priority-based CPU scheduler. In the Shinerbot, the RSSI module runs as a high priority task, the application layer runs as a lower priority task, and another idle task that brings the CPU to sleep. The Power module is in charge of bringing the robot to sleep and waking it up. More details on power stages will be elaborated in section 4.9. Math is a lightweight math library which includes a random number generator, and a fast integer square root, optimized for the range of values that are being used in Shinerbot.

In some cases, different libraries in the System and Driver layers work together behind the scenes to make the Shinerbot work, without the program in the Application layer needing to do anything. For example, upon reception of the Destination Byte of the incoming packet, the Comms module triggers the RSSI module to start ADC reads of the IR channels. The scheduler then starts the RSSI module on a high priority thread to compute the position of the transmitting neighbor. By the time the whole packet is received, the position information is ready. When the CRC check is good, both the packet data and the position information are pushed into the receive message queue, ready for the user program to retrieve. All this happens behind the scenes of the user program, allowing code in the user program to focus only on the navigation algorithm.

4.7.3 Application Layer

This layer is where the high-level behavior of the robot is programmed. The User Program is where the navigation algorithm is implemented. It only has to call APIs from the System layer to read inputs, make a decision, and command the robot to move in the desired direction. Test is a routine that is run after the robot is soldered and assembled. It calls the System layer APIs in a specific sequence, testing all aspects of the robot such as movement, RGB LED, ambient light sensing, IR transmission and reception.

After testing is done, the program RSSI calibration is run. This program basically runs a sequence of IR receptions with the robot moved over a distance. It then stores RSSI to distance calibration data to flash storage. The robot ID is flashed in at this stage too. Due to memory constraints, only one of the three programs in the Application layer is flashed into the microcontroller at any one time. This is okay as the test and the calibration routine only needs to be run once after the robot is first assembled.

4.8 System Messages

System messages are a means to control the swarm in a scalable manner. Having an on-off switch on each robot would be impractical when operating a hundred-robot swarm. System messages are packets that contain a system command. When a robot receives a command, it relays the system message 3 times, before taking action to that specific command. Due to the relaying of messages, a strong overhead IR transmitter is not required to broadcast the command to all robots. Special robots are programmed with the sole purpose of transmitting system messages. The swarm operator will only need to bring a command robot near the swarm, and all robots in the swarm will receive the same message in a split second. This is illustrated in figure 21. There are 4 system messages, namely Reset, Sleep, Wake, and Run. Reset brings the microcontroller to a full software reset. Sleep brings the robot into deep sleep, practically turning off the robot. Wake wakes the robot into a light sleep stage where its ready to resume the user program. Run resumes the user program.

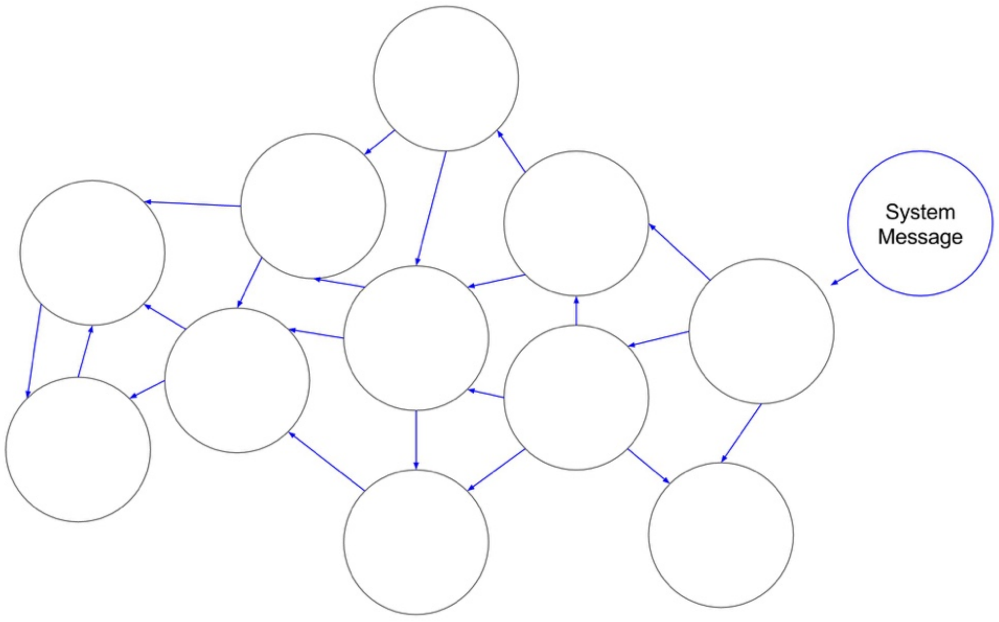


Figure 21: Relaying of System Message

4.9 Power States

There are four power states in the system. State 0 is where every part of the Shinerbot is running, including the user program on the CPU. State 1 is where the CPU sleeps, while all the peripherals are running. This happens when the user program calls the delay function. It puts the user thread to sleep, running the idle thread, which then brings the CPU to sleep. At the specified number of milliseconds, the interrupt handler wakes the user thread up, which brings the CPU back up to run.

Power state 2 is in a semi-sleep state. In addition to the CPU sleep in state 1, peripherals like the motors and LEDs are turned off. Only the analog circuits are left on to enable reception of messages. Power state 3 brings the robot into deep sleep. In addition to turning off the components in states 1 and 2, it turns off the analog circuits, main clocks, and phase lock loops (PLLs). The timer peripheral now runs on a low frequency auxiliary clock.

System messages tell the robot to transition between power states. The Sleep command brings the robot to state 2, relays the message three times, then goes to state 3. The robot will then go back to state 2 every 30 seconds for 400 ms before going back to state 3. This is to allow time to listen to any possible Wake system messages. To Wake the robots up, the Wake system message has to be broadcasted continuously for more than 30 seconds to ensure that the robots have received that message. Upon reception, the robot stays in state 2.

The robots do not instantly wake to full power in state 0 since they are not synced in that deep sleep cycle. Waking to full power instantly would not cause the whole swarm to wake up at the same time. Thus, the Wake command only brings the robot to state 2, in which they standby for the Run command, as they are now continuously waiting for messages. The Run command then brings them out of state 2 back to state 0. The block diagram in figure 22 illustrates the state transitions.

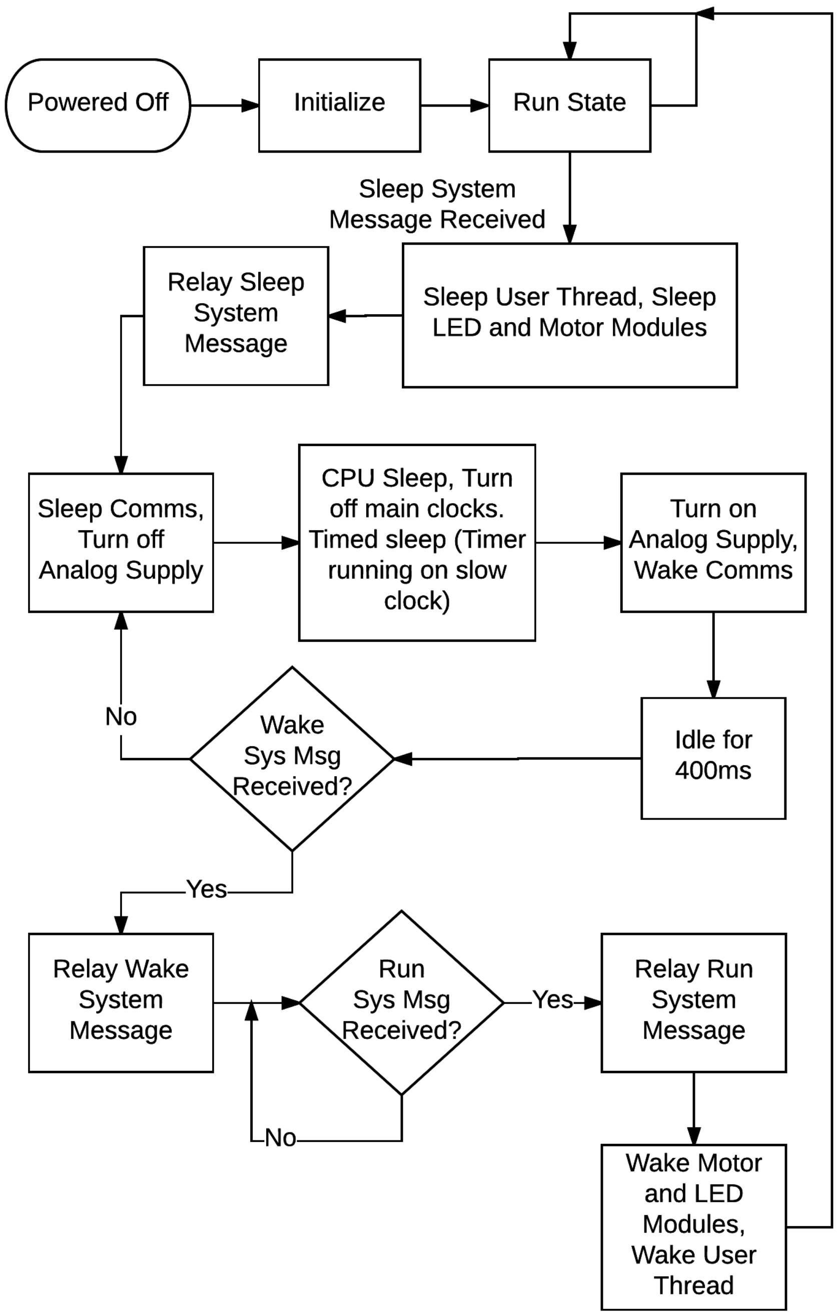

Powered Off
Initialize
Run State
Sleep System Message Received
Sleep User Thread, Sleep LED and Motor Modules
Relay Sleep System Message
Sleep Comms, Turn off Analog Supply
CPU Sleep, Turn off main clocks. Timed sleep (Timer running on slow clock)
Turn on Analog Supply, Wake Comms
Idle for 400ms
Wake Sys Msg Received?
No
Yes
Relay Wake System Message
Run Sys Msg Received?
Yes
No
Relay Run System Message
Wake Motor and LED Modules, Wake User Thread

Figure 22: Power State Transitions in Response to System Messages

5. Experimental Results

The experiments are conducted on a workspace made of a white wooden 4 ft x 4 ft square. The board was placed almost near the ground, and a level was used to ensure the board was horizontal. Since the robots move by vibration, a slight tilt to the workspace can bias the robot's motion. A digital projector is placed about 6 feet above the workspace to project the light pattern to simulate the dark and bright regions. We use a circular gradient light pattern to indicate the desired region where we want the Shinerbots to end up in.

In previous chapters, we described the golden shiners as wanting to go towards the dark area. In our experiments, this created a scenario where the swarm has two destinations: the dark circular spot that we are projecting, as well as the border of the projection which happens to be interpreted as a dark area by the Shinerbots. The results end up having a sub-group of robots at the desired circular region, and another sub-group of robots staying at the edge of the projection. For the purposes of this thesis, we wanted to show that the algorithm converges to the same region. Thus, to remove the confusion on the Shinerbots, the following experimental results are based on having a bright region as the desired location, using a bright centered circular gradient light pattern. The robots will no longer treat the projection border as an unintended desired destination

5.1 Intermediate Clustering Behavior of the Shinerbots

An intermediate clustering behavior is observed in our experiments. In our experiments, all robot locations are initialized randomly, and evenly spread across the workspace. The robots sometimes cluster into sub-groups upon the start of the experiments. As the clusters slowly move towards the desired region, they begin to converge into a bigger group later on, and finally to one whole swarm at its destination. Figure 23 illustrates such a scenario. In this particular experiment, the swarm is observed to split into two big sub-groups, a few pairs of robots, and some individual robots at 9.5 minutes after initialization. They continue to evolve into three sub-groups at time t = 36 minutes. Finally, the whole swarm converges at time t = 48 minutes.

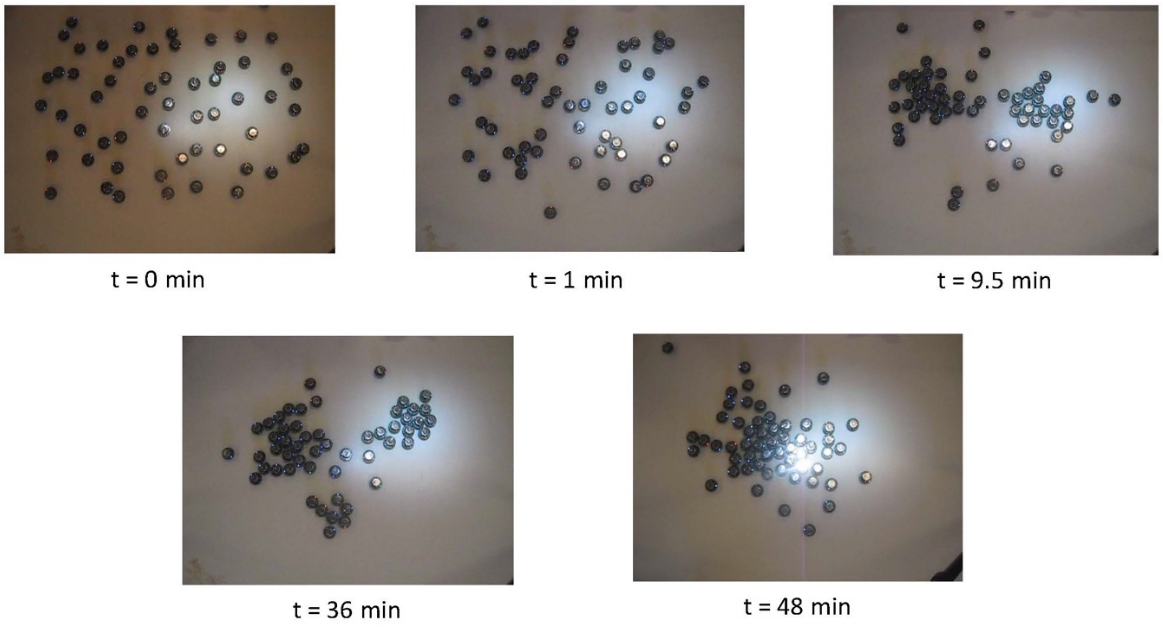


Figure 23: Clustering Behavior of Shinerbots from Initialization to Convergence

5.2 Experiment with Centered Destination Region

This next experiment shown in figure 24 shows the workspace having its desired bright region at the middle of the workspace. This experiment mainly demonstrates the social factor of the golden shiners algorithm at work, where the robots collectively gather together to mimic the schooling behavior of the fishes. There is still some clustering behavior of the swarm in the first 12 minutes. The swarm eventually converged in the middle to the desired region after 25 minutes.

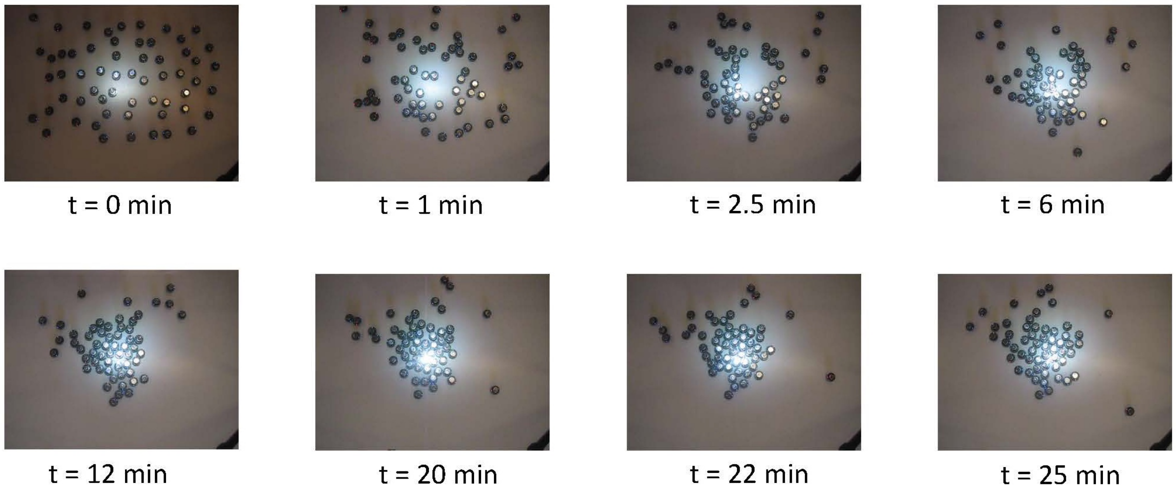


Figure 24: Experiment with Light Source In the Middle

5.3 Experiment with Off-centered Destination Region

The next experiment in figure 25 demonstrates the swarm moving into the desired region. The bright area is placed to the right of the workspace this time, while the robots were distributed towards the left of the workspace. This demonstrates the environmental factor of the algorithm at work. At t = 3 minutes, clustering behavior can be observed again. From t = 4.5 minutes to 14 minutes, the clusters that are close to the bright light source remain in the region and move a little closer to the center of the light pattern. The clusters that are in the dark still moving about randomly, while staying as a sub-group. From t = 20 minutes onwards, we can see that the sub-groups begin to merge to form a bigger group and finally converge to the desired region at t = 42 minutes.

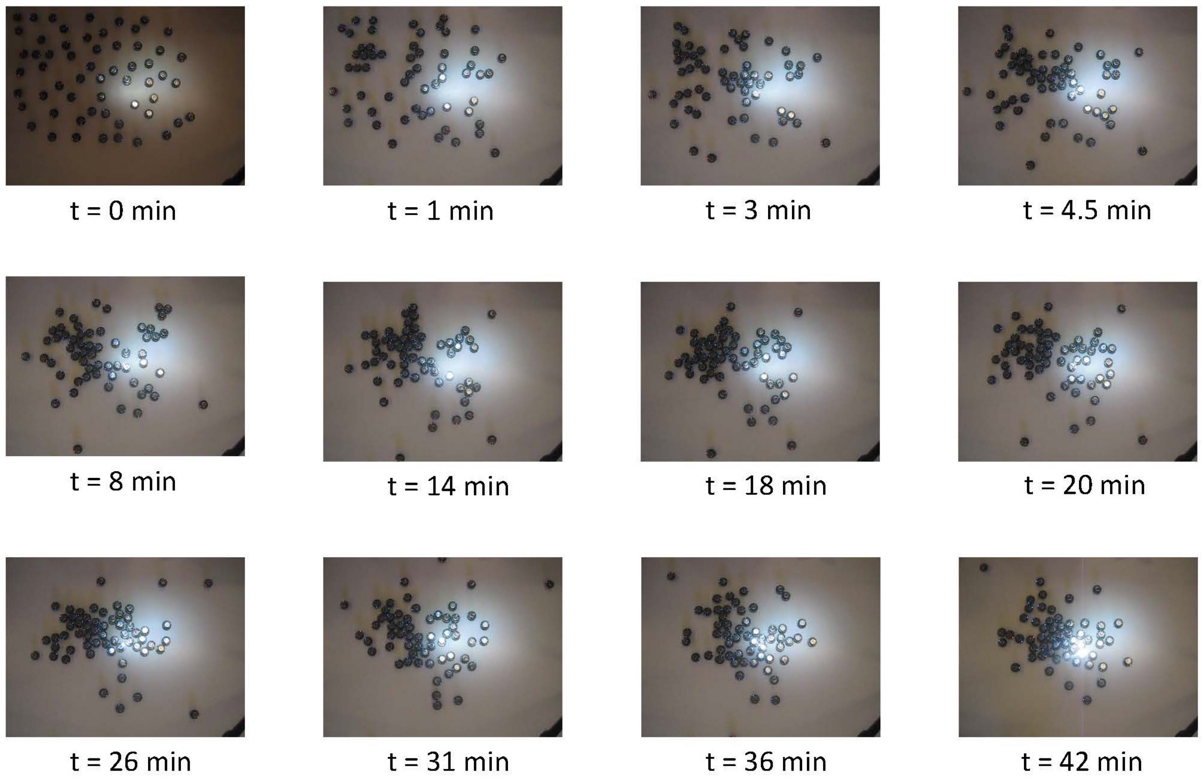


Figure 25: Experiement with Light Source Off-Centered

6. Conclusion

6.1 Summary of Achievements

This thesis presented a swarm navigation algorithm that mimics the behavior of the golden shiner fish. The algorithm is based on simple rules and requires minimal sensors and processing power of each individual robot. A ground based swarm robotic system was designed and built in-house to experiment with algorithm. Eighty-five robots were built, costing USD$20 each, with a size of 40.6 mm diameter. The robots have basic sensing, processing, and mobility requirements for the Golden Shiners algorithm. In addition, a messaging system is designed into the system to be a generic swarm robotics platform for future algorithms. In addition, scalability features were added to help manage and charge a swarm of robots. These are implemented as a charging plate as well as system command messages. Our experiments demonstrated that the Shinerbots collectively navigate to a desired region with the proposed algorithm. The work in this thesis was partially presented in [11], and [16].

6.2 Future Work

Both the swarm algorithm and the robots have possible improvements for future work. The current mathematical model of the algorithm models a single robot as a point mass which can move freely in a two-dimensional plane. More complex models such as a nonholonomic [6] cart can be considered in future work. The current algorithm ignores collisions. The robots will still try to move into the centroid of the neighborhood even though there is no longer any space. This is not a destructive behavior to the Shinerbots platform, but collision avoidance can be incorporated into the algorithm to enable application in bigger robotic systems and even unmanned aerial vehicles (UAVs).

There are opportunities for further work on the Shinerbots swarm platform as well. First, a more automated way of calibrating the infrared sensors can be designed instead of the manual dragging of individual robots over a distance that is being done now. Second, motion control of the Shinerbots can be improved. A calibration step can be added to ensure each robot moves accurately in the desired direction. Linear resonance actuators(LRA) vibration motors can be used instead of eccentric rotation mass(ERM) to improve efficiency. A mass programming interface with bootloader on each robot should be designed to improve scalability in trying out different experiments and re-flashing firmware. Neighbor sensing hardware has room for improvement as well. Increasing the sensing radius of the robots would help speed up the convergence of the algorithm.

The reasons for going with the vibration method of mobility are size and cost, which are important to scalability in a large swarm. The disadvantage of this mode of mobility is that it is relatively slow. Experiments often have to be conducted over at least an hour. If miniature and cheap commercial-off-the-shelf (COTS) wheeled parts are available, that would help in speeding up experiments.

References

[1] Alkilabi, M. H. M., Narayan, A., and Tuci, E. (2017). Cooperative object transport with a swarm of e-puck robots: robustness and scalability of evolved collective strategies. Swarm Intelligence. https://doi.org/10.1007/s11721-017-0135-8.

[2] Balch, T., and Arkin, R. C. (1998). Behavior-based formation control for multi-robot teams. IEEE Trans. Robot. Autom., 14(6):926–939.

[3] Berdahl, A., Torney, C. J., Ioannou, C. C., Faria, J. J., and Couzin, I. D. (2013). Emergent sensing of complex environments by mobile animal groups. Science, 339(6119):574–576.

[4] Bonani, M., Longchamp, V., Magnenat, S., R´etornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., and Mondada, F. (2010). The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2010), pages 4187–4193. IEEE.

[5] Couceiro, M. S., Figueiredo, C. M., Luz, J. M. A., Ferreira, N. M., and Rocha, R. P. (2011). A low-cost educational platform for swarm robotics. International Journal of Robots, Education and Art, 2(1):1–15.

[6] Desai, J. P., Ostrowski, J., and Kumar, V. (2001). Modeling and control of formations of nonholo-nomic mobile robots. IEEE Trans. Robot. Autom., 17(6):905–908.

[7] Dimakis, A. G., Kar, S., Moura, J. M. F., Rabbat, M. G., and Scaglione, A. (2010). Gossip algorithms for distributed signal processing. Proceedings of the IEEE, 98(11):1847–1864.

[8] Farrow, N., Klingner, J., Reishus, D., and Correll, N. (2014). Miniature six-channel range and bearing system: algorithm, analysis and experimental validation. In Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA2014), pages 6180–6185. IEEE.

[9] Gazi, V. (2005b). Swarm aggregations using artificial potentials and sliding mode control. IEEE Trans. Robot. Autom., 21(6):1208–1214.

[10] IEEE Computer Society LAN MAN Standards Committee (1997). Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE.

[11] Heng, L., and Gao, G. X. (2014). Navigating robot swarms using collective intelligence learned from Golden Shiner fish. In Proc. Collective Intelligence Conference (CI2014).

[12] Hilder, J., Naylor, R., Rizihs, A., Franks, D., and Timmis, J. (2014). The pi swarm: A low-cost platform for swarm robotics research and education, pages 151–162. Springer.

[13] Kernbach,S., Häbe,D., Kernbach,O., Thenius,R., Radspieler, G., Kimura, T., and Schmickl, T. (2013). Adaptive collective decision-making in limited robot swarms without communication. The International Journal of Robotics Research, 32(1):35–55.

[14] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In Proc. IEEE International Conf. Robotics Automation, pages 1398–1404.

[15] Lin, J., Morse, A. S., and Anderson, B. D. O. (2007). The multi-agent rendezvous problem. Part 1: the synchronous case. SIAM J. Control Optim., 46(6):2096–2119.

[16] Luo, E., Fang, X. H., Ng, Y., and Gao, G. X. (2016). Shinerbot: Bio-inspired collective robot swarm navigation platform. In Proc. Institute of Navigation GNSS+ conference (ION GNSS+ 2016).

[17] Merheb, A.-R., Gazi, V., and Sezer-Uzol, N. (2016). Implementation studies of robot swarm navigation using potential functions and panel methods. IEEE Trans. Mechatronics, 21(5):2556–2567.

[18] Mondada, F., Franzi, E., and Guignard, A. (1999). The Development of Khepera. In Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop, HNI-Verlagsschriftenreihe, Heinz Nixdorf Institut, pages 7–14.

[19] Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Trans. Autom. Control, 51(3):401–420.

[20] Olfati-Saber, R., Fax, J. A., and Murray, R. M. (2007). Consensus and cooperation in networked multi- agent systems. Proceedings of the IEEE, 95(1):215–233.

[21] Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In Proceedings of 2012 IEEE International Conference on Robotics and Automation (ICRA-2012), pages 3293–3298. IEEE.

[22] Saska, M., Vonasek, V., Chudoba, J., Thomas, J., Loianno, G., and Kumar, V. (2016). Swarm distri-bution and deployment for cooperative surveillance by micro-aerial vehicles. J. Intelligent Robotic Sys- tems, 84(1):469–492.

[23] Vishay Semiconductors (2011). TEMD7100X01 silicon PIN photodiode spec sheet.

[24] Sturza, M. A. (1988). Navigation System Integrity Monitoring Using Redundant Measurements. Wi-ley Online Library.

[25] Torrellas Socastro, S. (2013). Cognitive stimulation through task-oriented telerobotics. Master's Thesis, Universidad Oberta de Catalunya.

[26] Vartholomeos, P., and Papadopoulos, E. (2006). Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators. In Proceedings of 2006 IEEE International Conference on Robotics and Automation (ICRA- 2006), pages 649–654. IEEE.

[27] Waydo, S., and Murray, M. (2003). Vehicle motion planning using stream functions. In Proc. IEEE Int. Conf. Robot. Autom., pages 2484–2491.

Appendix A: Schematics of a Shinerbot

Figures 26 – 29 show the schematics of a single Shinerbot.

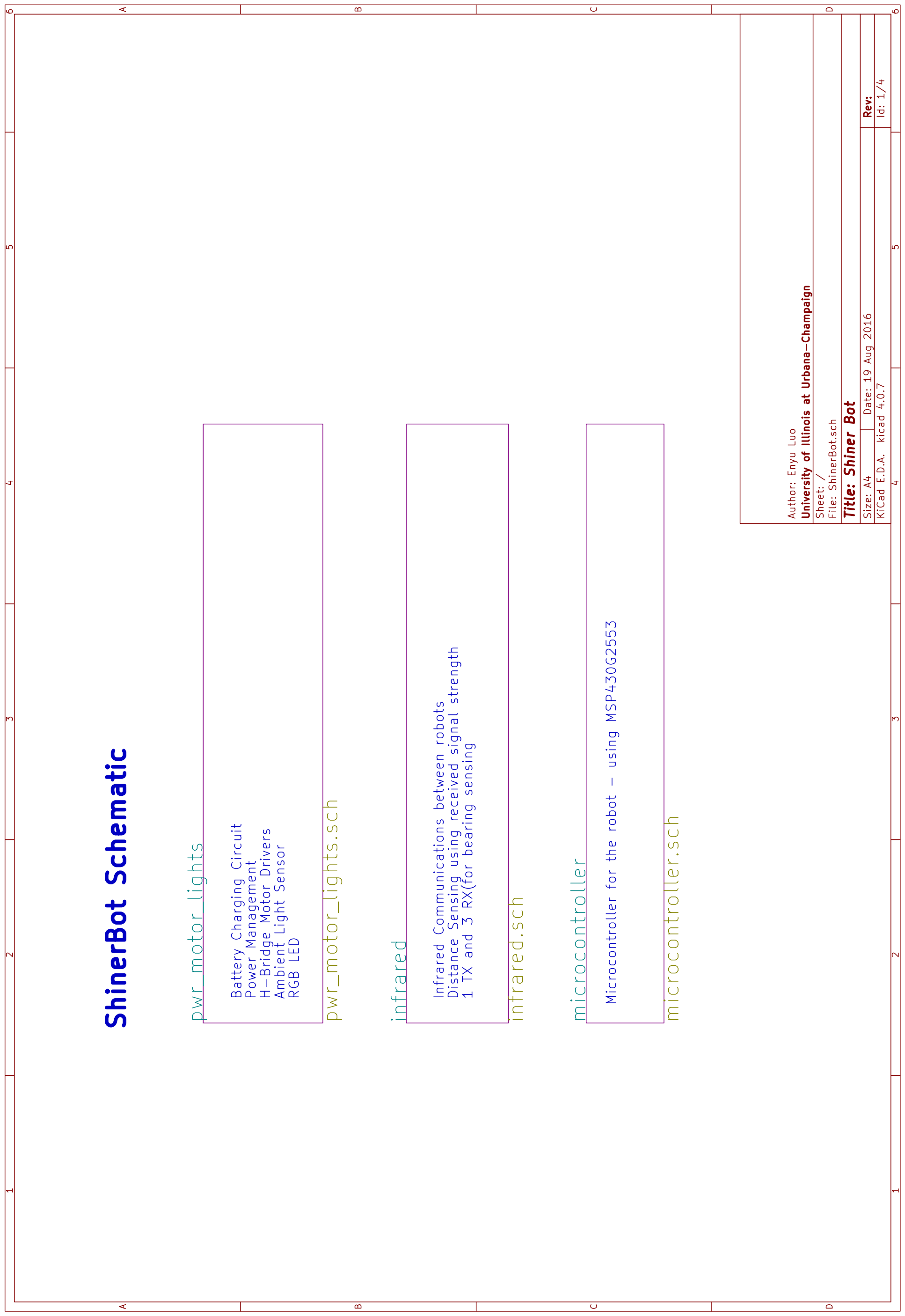


Figure 26: Contents of Shinerbot Schematic

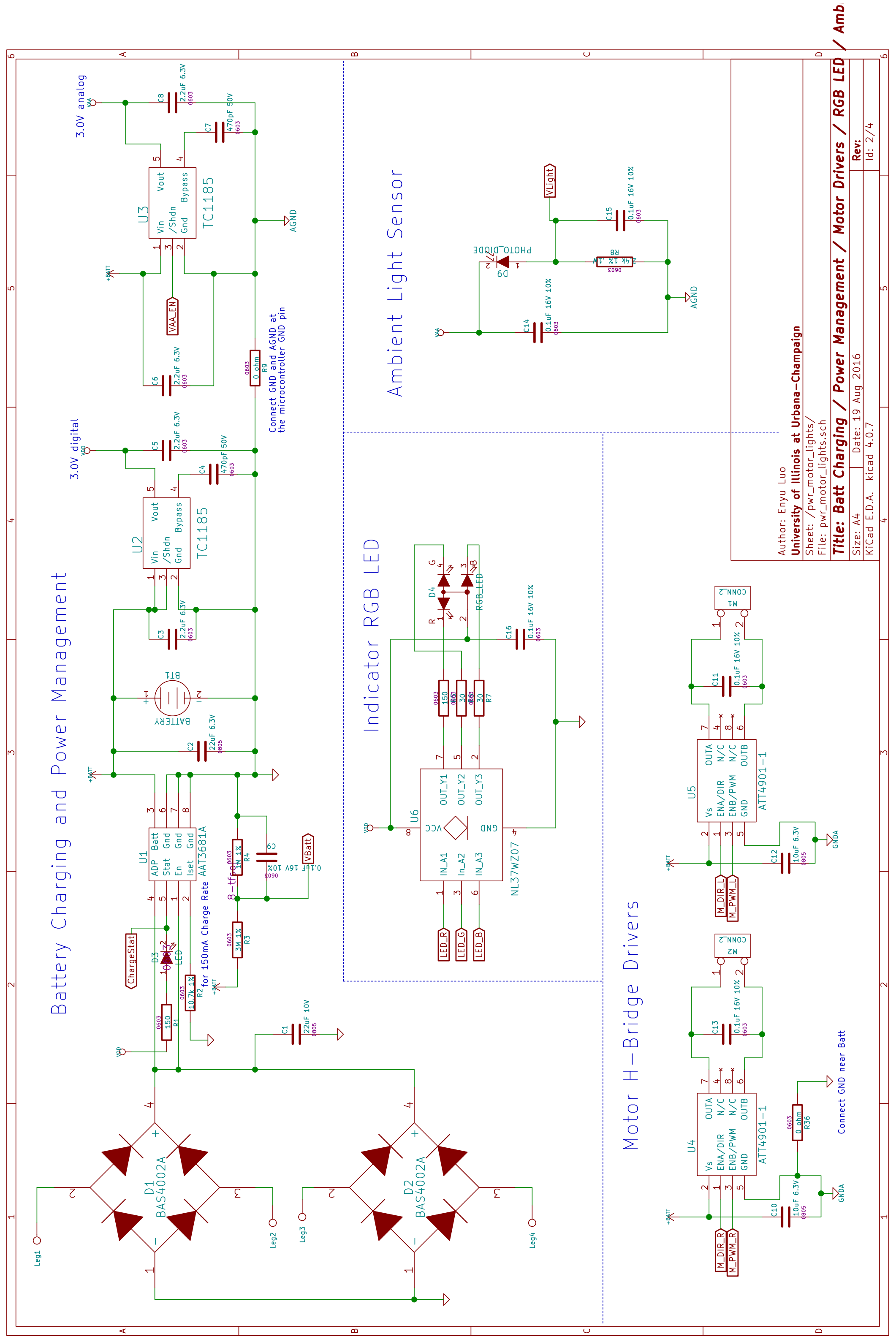


Figure 27: Shinerbot Schematic - Power, Light Sensor, H-Bridge, LED

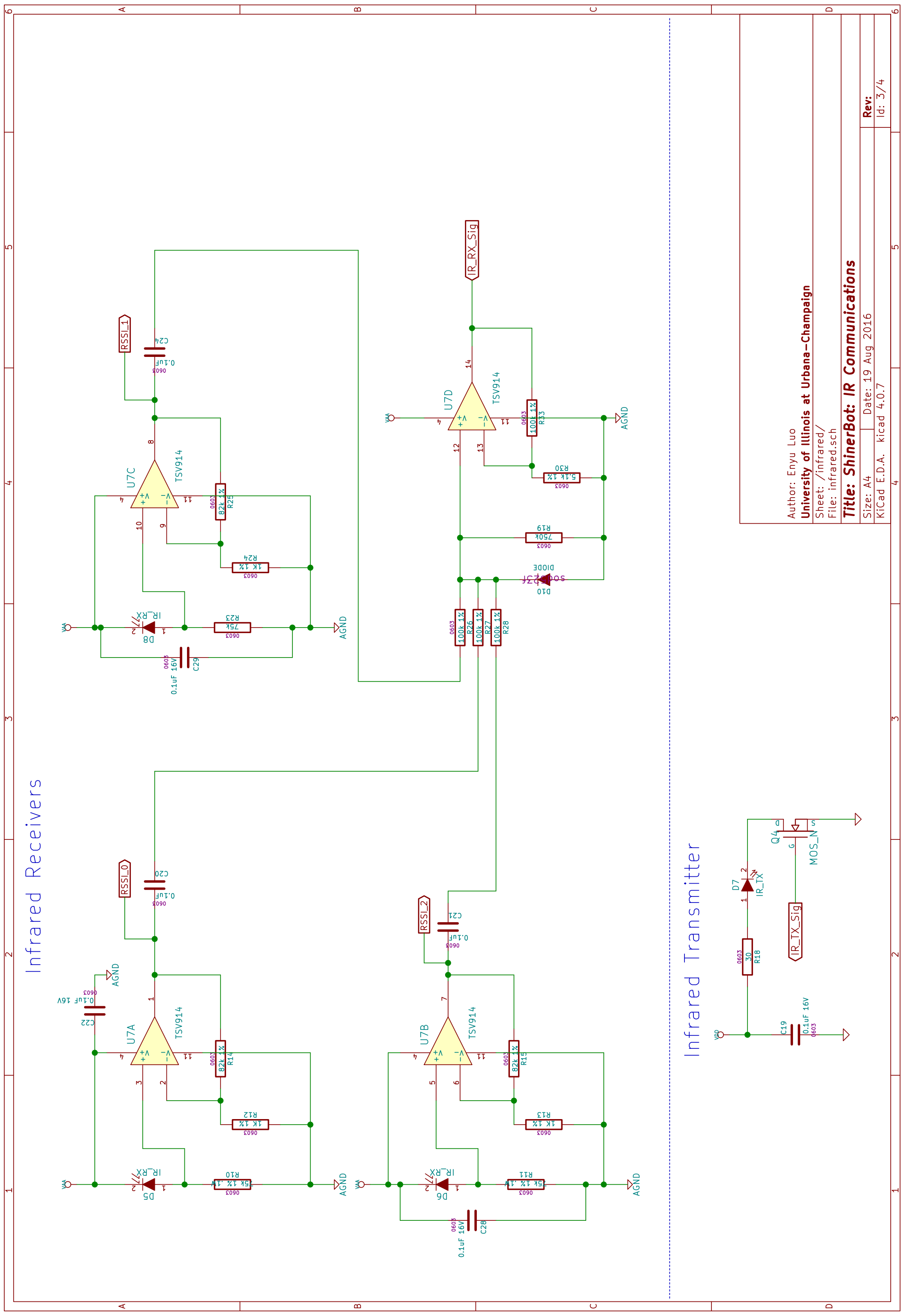
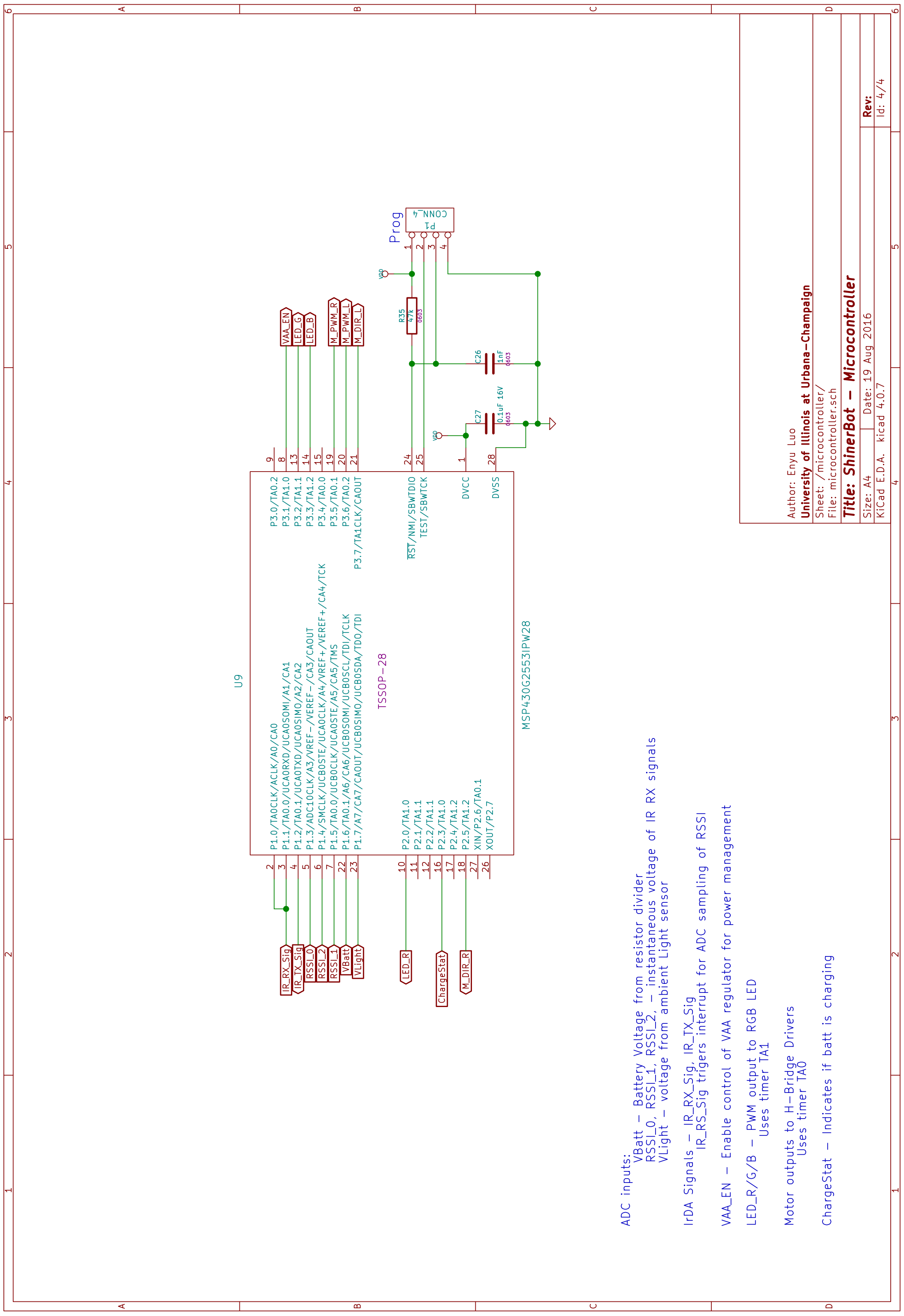


Figure 28: Shinerbot Schematic - Infrared Transmitter and Receivers



ADC inputs:
VBatt – Battery Voltage from resistor divider
RSSI_0, RSSI_1, RSSI_2, – instantaneous voltage of IR RX signals
VLight – voltage from ambient Light sensor
IrDA Signals – IR_RX_Sig, IR_TX_Sig
IR_RS_Sig trigers interrupt for ADC sampling of RSSI
VAA_EN – Enable control of VAA regulator for power management
LED_R/G/B – PWM output to RGB LED
Uses timer TA1
Motor outputs to H–Bridge Drivers
Uses timer TA0
ChargeStat – Indicates if batt is charging
U9
TSSOP-28
MSP430G2553IPW28
Prog
P1
CONN_4
R35
47k
C26
1nF
C27
0.1uF 16V
Author: Enyu Luo
University of Illinois at Urbana-Champaign
Sheet: /microcontroller/
File: microcontroller.sch
Title: ShinerBot – Microcontroller
Size: A4
Date: 19 Aug 2016
Rev:
KiCad E.D.A. kicad 4.0.7
Id: 4/4

Figure 29: Shinerbot Schematic - Microcontroller