The Hive

Final Report

Aldo Plaku

Project: The Hive

April18, 2011

EEL 5666 – Intelligent Machines Design Laboratory

Instructors: A. Antonio Arroyo, Eric M. Schwartz

TAs: Devin Hughes, Ryan Stevens, Josh Weaver,

Sean Frucht, Tim Martin

Table of Contents

Abstract 3

Executive Summary 4

Introduction 5

Integrated Systems 6

Mobile Platform 7

Actuation 8

Sensors 9

Behaviors 13

Experimental Layout and Results 14

Conclusion 14

Documentation 16

Abstract

Swarm robotics is the design, coordination and control of a multi-robot system made up of physically simpler robots. It is inspired in part by the social behavior of swarm insects.Simple sensory response agents can then produce complex swarm behaviors.The most important component of a swarm system is the communication between all the agents. Usually, no robot is in charge of the swarm. Instead, the intelligence comes out of the cooperation between the robots. However, for this projectmy computer will act as an overmind to the hive, giving the robots information when needed.

The swarm will consist of several hexapodrobots with various sensors attached to interact with the environment.The hexapod design was chosen as an interesting design that would be relatively simple and provide a good amount of maneuverability.The swarm communication will be accomplished with the Bluetooth protocol. My personal computer will act as a hub and the Bluetooth master and the robots will have class two Bluetooth nodes set in slave mode.Each robot can accomplish tasks individually, but whenever there needs to be communication, the Bluetooth will provide interactivity.

The robots will accomplish obstacle avoidance and the accomplish search and locate. The robots will act as sentries looking for target instances using the sensors. Once an instance is triggered it will talk back to the hub(my laptop) and imitate letting a technician know about an event.

Executive Summary

The goal of this project is to explore the concept of swarm robotics. Agents of the swarm are made that can accomplish individual task, but also capable of becoming part of a larger network of agents. To emulate a swarm of insects, the robots are hexapod walkers that have a set gait using three servos. An Arduino Pro mini board is used to control the robot as well as read and react to various sensors on the robots. The robots will avoid obstacles using whiskers and an IR sensor. They will also have Bluetooth, cds cells, and IR receivers to interact with the environment. The base of the robots is a custom fabricated PCB board.

The robots will run on software that is set up as a hierarchy of modular behaviors. Each behavior will poll the sensors needed and give suggestion to the arbitrator. The arbitrator will then decide on what the robot should do. Some behaviors are more important and are given precedence when certain triggers are present through the sensors.This architecture allows forthe system to be expanded with new behaviors very quickly and easily. This setup is very conducive to a swarm since every behavior is modular.

The swarm uses Bluetooth to communicate to the network. Each agent has a class two Bluetooth module attached. A laptop with a class one module serves as the master node or hive mind of the network. Once the robots have connected to the master node, they have an open serial communication between the two of them. The master node can send commands to each agent, receiver sensor data updates, and relays messages for inter swarm communication.

The two main tasks the swarm is programmed to do is find darkness and find an IR beacon. For both triggers, the robots send back a found report to the master node. From there, the other agent can be told to look for the same trigger. Also, by sending different commands, the robots can be directly controlled. They can be paused for a few seconds and then continue whatever task it was performing. The Bluetooth connection allows the user to have almost complete control over the agents in the swarm.

Introduction

Swarm robotics is a trendy area of robotics nowadays due to the lower price of microcontrollers and sensors. It involves building numerous (usually identical) robots that can work together using a self-organizedsystem of decentralized collective behavior. Current microcontrollers are smaller and cheaper. This allows for the creation of small, cheap robots that can accomplish cooperative tasks efficiently.Innovations in microelectronics make it possible to manufacture such swarm robots in smaller and smaller sizes. However since they are small robots, reaching a common goal is not an easy task due to the constraints of size: small sensor ranges, limited computational power, and imprecise locomotion. The robots are not very technically advanced, however the strength of swarm robotics is in the emergent behavior of the robots working with simple rule set.

The goal of my robots is to explore an area, avoiding obstacle, and act upon a preordained target situation. Practical applications would have the swarm move about a large area patrolling and monitoring conditions autonomously. Using people to monitor a large, tall building would be very time consuming and expensive. The same is true if a single, more advanced robot were to be used. The small robots can patrol multiple areas simultaneously looking for anything out of place. A swarm of robots can reduce response time to any dangerous situation. The robots can then come together to respond to the situation as a collective.

For this project, I created at least twohexapod robots that can sense their environment and communicate with each other. The robots will mimic sentry behavior by searching an area for targets and then communicate to the hive hub once the agent has found it. Other tasks that come up during research that are feasible will also be considered.

This paper will describe the physical platform of the robots, the limited intelligence or behavior given to the robots, and how the swarm agents communicate and interact with each other.

Integrated Systems

The agents' hardware is designed to be simple and easy to implement. Following the bug theme of the robots, obstacle avoidance is done using bumper switches that look like antennas. Whenever an antenna hits an object, a switch is triggered and the robot knows that it should not move in that direction.The agents were designed to have as few complex inputs as possible. The obstacle avoidance is all done through digital input responses from sensors. The bump whiskers are just switches to the agents. The SHARP IR sensor are digital as well; they only know if there is something in front of the agents or not.

The other sensors, beacon detection and CDS, use analog inputs and check against thresholds.

The software runs as a hierarchy in response to the sensor data. The sensors are polled in the main infinite loop and then software then arbitrated behavior based on the importance of the behavior. For example, the obstacle avoidance is the most dominant behavior; it has precedence over all other behaviors. Therefore, the first thing checked are the obstacle sensors. If they are tripped, then the robot reacts immediately and ignores all other sensors.

The movement behavior runs at a global level, that is, almost asynchronously to the other behaviors. However, the other behaviors dictate the direction of the motion and sometimes the number of steps. The figure below explains the architecture.

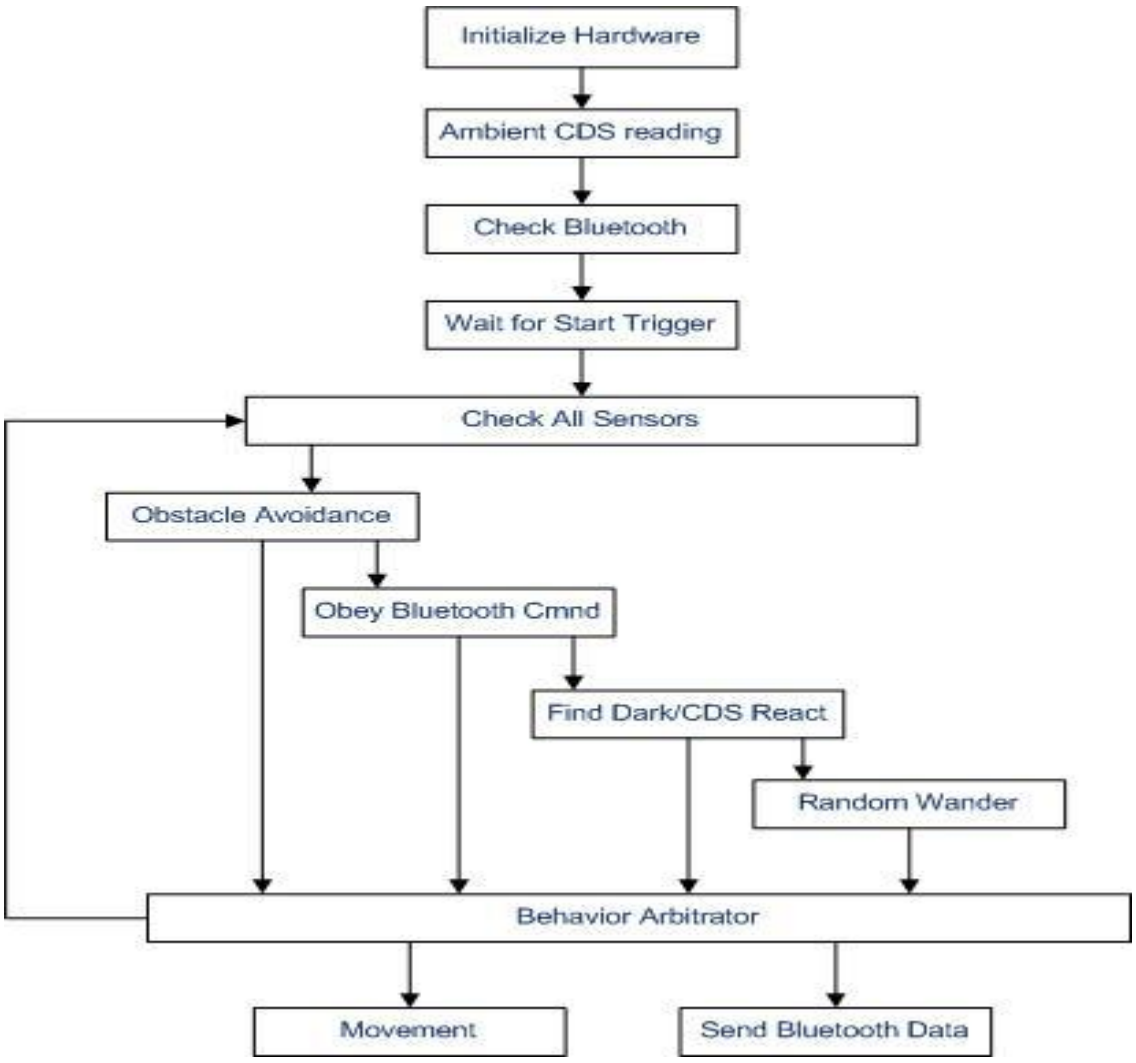

Figure 1. Software Flow.

Mobile Platform

The body will be composed of the following:

- One 3.7V NiMH battery
- Three small servos controlling the hexapod motion
- An Arduino Pro Mini microcontroller
- Peripheral sensors: bump, IR, hall effect
- Bluetooth slave node: Roving Network RN-42
- A breakout board to allow for easy hook up and act as a base for the body

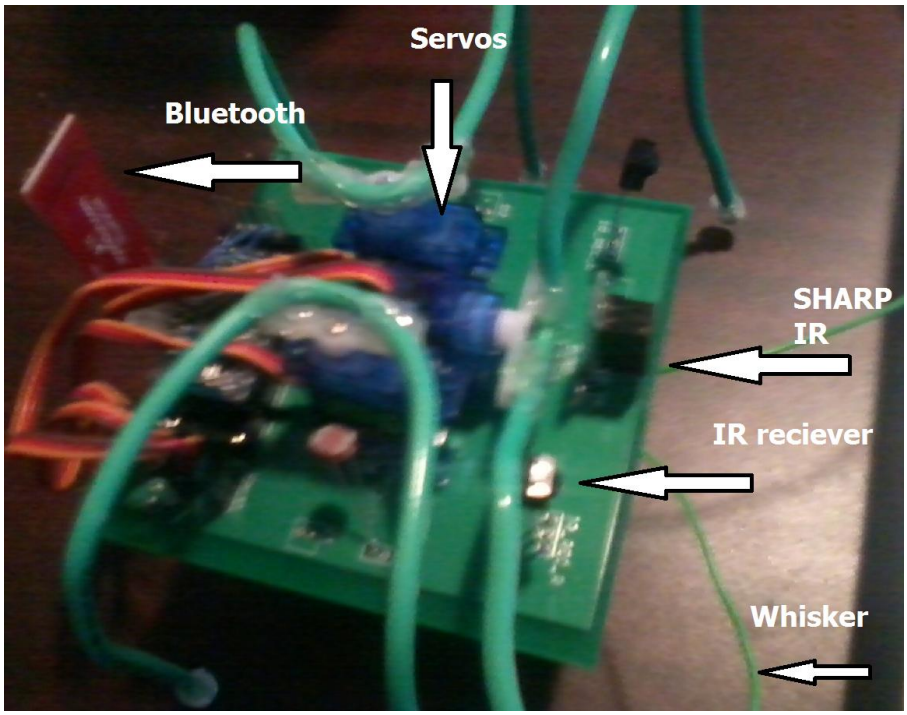

Figure 2. Finished Design

Actuation

The robot will have a very simple style of motion. The main goal of the project is to build an economic swarm of robot agents that don't need to have incredibly complex motion or physically demanding tasks. It needs to be lightweight and easy to remake. The robots' motion will be driven by three servos connected to each "pair" of legs. Two servos control the left and right pair of legs respectively. A middle servo controls the two forward legs. These legs are necessary to shift the weight of the robot from one side to the other. In effect, they act as the lifting motion for each side of the robot; allowing the legs to go up, forward, down, and then back to create forward motion. I am using Hextronik HTX500 5hr micro servo with the following specs:

- Size : 22.9x11.4x22 mm / 0.9x0.45x0.87 in
- Voltage : 3v ~ 6v
- Weight: 5g / 0.18oz

Sensors

- **Whiskers bump sensors**

  The whisker bump sensors are made by driving two of the Arduino pins high using the internal pull up resistors. The whiskers are made of wire that has a section of insulation striped off. When the whisker is hit, the exposed wire comes in contact with a set of headers that are grounded. This will then drive the input pin low. In the code, I look for a low signal each loop of main to see if there are obstacles to the left or right.

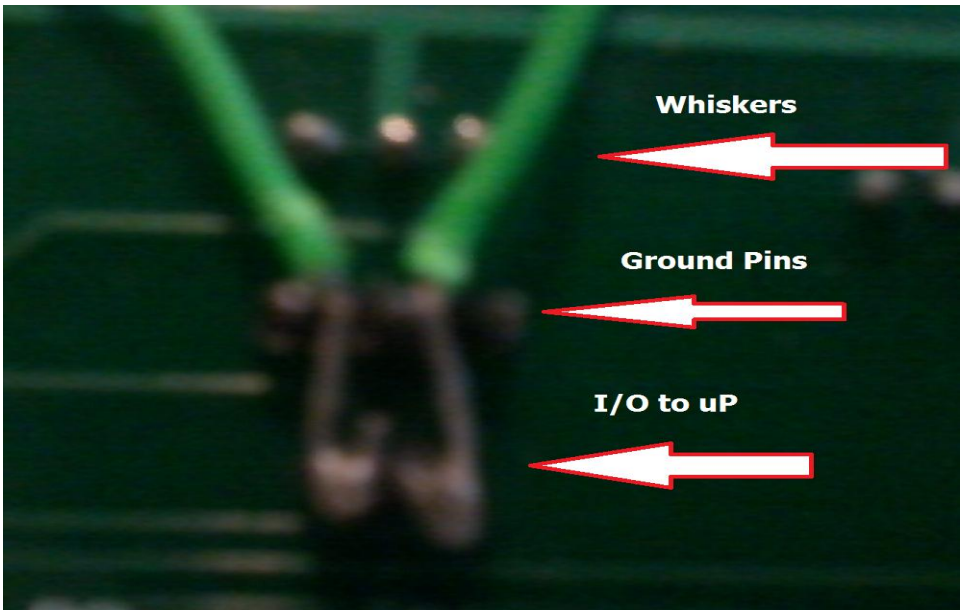

Figure 3. Bump Sensors/

- **IR ProximitySharp Digital Distance Sensor**

  Obstacles that are directly ahead of the robots will be avoided using the Sharp digital IR sensors. The sensors are very small, cheap, and easy to use. The chip with the carrier board from pololu only requires three pins to function. In addition to the power and ground pins, it only has a digital output pin. The sensor produces a high signal when there is nothing in front of it. When an object is detected, the sensors output a low signal. This

signal is tied to the microcontroller. Once read, it will force the robot to react and avoid the obstacle.

The digital behavior of the sensor makes interfacing very simple due to the lack of analog to digital conversion that is required by most IR sensors. However, the simplicity also limits the amount of information that the sensor can provide about the environment. The sensor cannotreveal anything about the distance of obstacle.It can only determine if there is an object within itsten centimeter range.

After some testing, it is clear that the sensors are very responsive. The range is almost always at about eight cm or slightly less. It does not seem that ambient light will be an issue with these IR sensors. The sensors use triangulation to determine if there is an object, so IR intensity does not come into effect.

- **IR Beacon**
  - Custom made beacon with two IR transmitters to mimic a target.
  - Two IR sensors will then be placed at the front of the robot so that they can hone in on a beacon.
  - The robots will have two 36 kHzIR sensors to detect the beacon. They will use VISHAY electronics TSOP34338 IR Led receiver. It produces a digital signal if a 36 kHz IR transmission is detected.

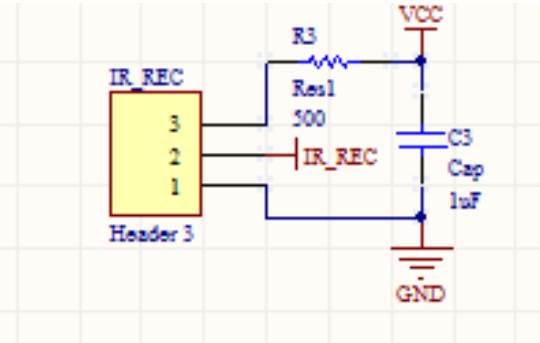

Figure4: 36 kHz IRBeaconReceiver

  - A 555 timer is used to create a 36 kHz square wave then this is output to an IR led. The pulsed IR light can then be sensed by the IR sensors on the robot. The schematic for this circuit can be seen in illustration 3. Pin 12 will determine if the

beacon is on or not. The beacon should not go on until the other robots will need to converge to one location.

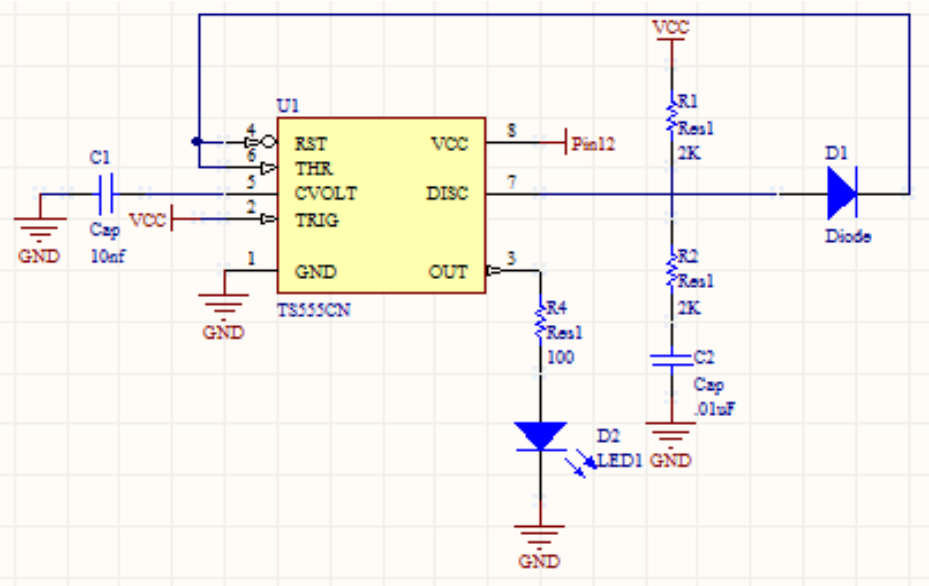

Figure 5: IR Beacon Transmitter

- **Class two Bluetooth module – Roving Networks RN-42**

A small, cheap Bluetooth module allows for inter robot communication using my PC as a hub.The module that will be used is the Roving Networks RN42. It is a module designed to replace serial cables. It is a class twoBluetooth module that can transfer data up to about 15meters.Since they will all be class two, the modules will only be able to communicate to the class one module in my laptop.Therefore, the laptop will act as a hub, transferring data between the different agents in addition to sending commands if necessary.

Setting up the RN-42 chip is very easy. It emulates UART transmission so all the Arduino board needs are the RX and TX pins. It is defaulted to 1 start bit, 8 data bits, and 1 stop bit, and transmit rate of 115.2kbps.

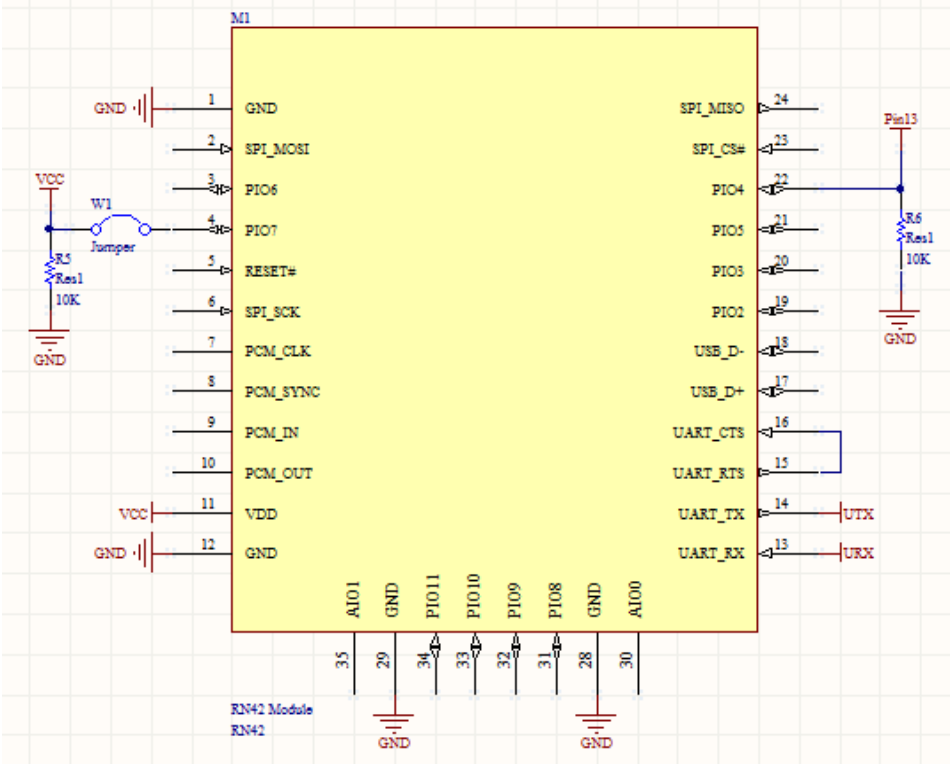

Figure 6: RN-42 Module for Bluetooth Communication

- **CDS cell**

A CDS cellsis a light detectors. As the light intensity shining on it varies, the resistance of the cell changes linearly. It is set up as a simple voltage divider with R1 equal to 1KΩ and input into ananalog pin of the microprocessor. The software is set up so that when the processor starts up, it takes an ambient light value. It is then constantly looping through and reading the current light intensity. If the light intensity is less than the original ambient value, then the microprocessor reacts accordingly. It takes the ambient value, halves it and looks to see if the current value is less than the ambient value minus one half of the ambient. This is to give it a decent sized resolution.

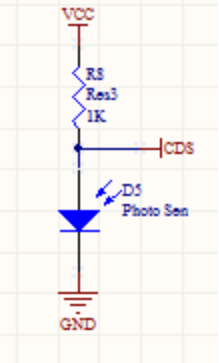

Figure 7. Cds Cell voltage divider.

Behaviors

The most basic behavior the robots will have is to explore the area looking to stumble upon any targets in the field. The robots use the proximity IR sensors and the whisker bump sensors to see if anything is in front of it. The robot can then sense if the obstacle is directly in front of it or if it needs to simply move to the left or right slightly.

As the robots move around, they will wander the field and eventually find target events.Once the targetsare found by one of the agents, it will tell the hub it has found the target via Bluetooth. Next, the hub will relay the information to the other robots. Meanwhile, the other robots will react accordingly. If the beacon was found by one robot, they will cease to wander around and start looking for the beacon and start to hone in on the IR beacon.The IR sensors will be set up so that if the IR beacon is directly in front of the robot, both beacons will pick it up. If only one of them picks them up, then the robot has a direction it needs to reorient itself to. It will make slight adjustments until the IR beacon is directly in front of it. If darkness is found by one of the agents, then the other agents will not be involved. The single agent will freeze and tell the hub that darkness was found.The crux of the behavior is the hub created by the Bluetooth. My laptop acts as the hive mind to the agents. It provides them with information gathered by other agents and allows me to control the behavior if needed.

The robots core function is the random wandering. I use the timer to seed the random number generator function inherent to the microprocessor. The robot has a skewed random motion routine. I did not want the robot to be truly random, but have more forward motion than anything

else. The agents move forward for 70% of the time and turn left or right in equal probabilities. Also, the number of steps in each direction is a random number generated in the code. Again, I favor forward motion; they can move anywhere from one to five steps forward and only one to three steps to the left or right.

Experimental Layout and Results

The overall goal of the experimental demo was to show that the swarm could communicate with the master node in my laptop and accomplish the tasks I laid out for them.To start up the demo I plug the batteries into the agents and establish the Bluetooth COM port connection. I used Putty and created two terminal windows linked to each agent. Once that was ready, I initiated the behavior of the robots by touching the front bump sensor. Once they were moving around, I demonstrated and talked about the random wandering routine the agents have programmed into them. Afterwards I showed each of the obstacle avoidance sensors working properly. Finally, I showed the special sensors and the Bluetooth hub I created. I covered the cds cell to create darkness and showed that the robot sent back “Darkness found” to the hub. Next, I put in the IR beacon and showed that the robots recognize it and send back that they have found the target. Finally, I talked about the power of the hive mind interconnecting the robots. With it I can send commands and make the robots do whatever tasks are necessary. I programmed in a dance routine and triggered it using a character sent through the Bluetooth to each agent.

Conclusion

Throughout the semester, for this project, I designed a custom PCB to create a platform for an integrated system of various sensors, servos, a microcontroller, and Bluetooth module. I developed software to incorporate all of these components into an organized hierarchy of behaviors to create sentinel agents that are part of a swarm. The Bluetooth dongles on the robots were set as slave nodes that communicated with the master node in my laptop.

The Bluetooth communication was surprisingly easy once it had been set up. Setting it up gave me some trouble.I had never surface mounted something as small as the Bluetooth dongle. After working on it for a while, I decided it was not worth wasting more time trying to debug further. I

got the blueSmirf modules instead. They were very easy to set up and I had Bluetooth running the day I got them. The Bluetooth dongles I got were just like serial, so to the Arduino, it was just using simple serial communication.

The mechanical actuation was an interesting part of the project. Initially I wanted to do a four legged walker. However, after talking to the TAs and creating various designs I found it was easier to add an extra pair of legs. The biggest challenge in making a four legged walker with only two or three servos was the movement of the waist. There was no robust way to create an upward motion of the legs to get traction. Nonetheless, the robots were very mobile and had a very natural feel when they walked. I feel that the movement seemed very similar to some bug like motion.

If I were to try and improve the project given much more time, I would improve on the PCB design created. Some of the connections I made on the PCB ended up being wrong so those would definitely need to be redone. In addition, I would like to add more headers to the unused microprocessor board. That way I can add functionality and sensors without having to create a totally new board design. I envision that the best way to make a good swarms is to make the agents as modular as possible.

Another weak point in the project was the IR beacon. Originally I wanted to have each agent have individual IR beacons to provide the swarm knowledge of other agent locations. However, the current draw for the beacon was too high to integrate with the other systems. The onboard IR beacons had extremely small range; smaller than the obstacle avoidance.In future projects, I would add an on board transistor circuit to create more current for the IR LEDs and possibly even two power supplies.

Another interesting idea to expand upon this project is to add more nodes for the swarm to connect to in order to establish location. You could use something similar to cell phone towers where the agents would be receiving and transmitting to the towers. The master node could then use all that information to make a decision on where the agent is.This could be used to map an area for example.

Overall, the swarm mechanic is very interesting and gave me a great deal of fun experience in making simple robots that are part of a powerful interconnected network. It was interesting that the agents did not have to be part of the network. If there was no network present, they would still accomplish small tasks on their own.

Documentation

Bibliography

The Arduino Community
http://www.arduino.cc/

Arduino Playground – Four Legged Walker
http://www.arduino.cc/playground/Main/4LegWalker

Final Arduino Code

```
/*
Aldo Plaku
EEL 5666C
Movement Code with IR recievers and Bluetooth
*/

#include <Servo.h>

//Define the center position of each servo.
#define CntrR 90
#define CntrL 95
#define CntrW100  //or 100 or 115
//Define the the angle of movement per step.
#define MoveR 18
#define MoveL 18
#define MoveW 14
//Define dirint values for readability
#define Frwd 0
#define Rvrs 1
#define Left 2
#define Right 3
#define Halt 4

//Define Servo Entities
Servo LeftS;
Servo RightS;
Servo WaistS;

intincomingByte = 0;  // Char recieved by Bluetooth

intLpos, Rpos, Wpos, dir,cycle, steps;
intrandDir, randSteps;//Long variables for randomness
int buttonState11 = 0;  //Forward Sensor State
int buttonState12 = 0; //Left Whisker State
int buttonState13 = 0; //Right Whisker State


int RIR = A0;    // Set Right IR Rec as A0
int LIR = A1;    // Set Left IR Rec as A1
intRIRval = 0;  // Stores the IR analog values
intLIRval = 0;
```

```
intCDS_val = 0;
intCDS_amb = 0;

int Start = 0;

void setup(){
Serial.begin(115200);    // opens serial port
pinMode(11, INPUT); // Sharp
pinMode(12,INPUT);  // Bump Left
pinMode(13,INPUT);  // Bump Right
pinMode(10,OUTPUT); // Becon
//Attach Servos to pins 9, 8, and 7
LeftS.attach(9);
RightS.attach(8);
WaistS.attach(7);
//Initialize the servos to center
Rpos = CntrR;
Lpos = CntrL;
Wpos = CntrW;
//Initialize variables
dir = Frwd;
cycle=0;
steps= 0;
randDir=Halt;
randSteps=0;
//Initialize Ambient Light
CDS_amb = analogRead(A3);
}

//Function for gait during forward motion
voidgoFrwd(int cycle){
//Waist Legs Movement
if (cycle<12) Wpos=CntrW+MoveW;
else if (cycle<37) Wpos=CntrW-MoveW;
else if (cycle<62) Wpos=CntrW+MoveW;
else if (cycle<87) Wpos=CntrW-MoveW;
elseWpos=CntrW+MoveW;
//Right Legs Movement
if (cycle<25) Rpos=CntrR+MoveR;
else if (cycle<50) Rpos=CntrR-MoveR;
else if (cycle<75) Rpos=CntrR+MoveR;
```

```
elseRpos=CntrR-MoveR;
//Left Legs Movement
if (cycle<25) Lpos=CntrL+MoveL;
else if (cycle<50) Lpos=CntrL-MoveL;
else if (cycle<75) Lpos=CntrL+MoveL;
elseLpos=CntrL-MoveL;
}

//Function for gait during backwards motion
voidgoBack(int cycle){
goFrwd(100-cycle);
  //Motion is same as forward but in reverse
}

voidgoLeft(int cycle){
//Turning Left is going forward with left legs in reverse
goFrwd(cycle);
//Left Legs Movement
if (cycle<25) Lpos=CntrL-MoveL;
else if (cycle<50) Lpos=CntrL+MoveL;
else if (cycle<75) Lpos=CntrL-MoveL;
elseLpos=CntrL+MoveL;
}

voidgoRight(int cycle){
//Turning Right is going forward with right legs in reverse
goFrwd(cycle);
//Right Legs Movement
if (cycle<25) Rpos=CntrR-MoveR;
else if (cycle<50) Rpos=CntrR+MoveR;
else if (cycle<75) Rpos=CntrR-MoveR;
elseRpos=CntrR+MoveR;
}

//Function to stop the robot
voidgoHalt(){
Rpos = CntrR;
Lpos = CntrL;
Wpos = CntrW;
}

//Creates random direction for the robot
```

```
//70%-Forwards
//15%-Left
//15%-Right
voidrandomDir(){
randDir = (int) random(100); //random number between 0 and 100
if(randDir<15){
dir=Right;
  }
else if(randDir>=15 and randDir<30){
dir=Left;
 }
else{
dir=Frwd;
 }
}

//Turns the robot in random directions
//50% Right or Left
voidrandomTurn(){
intrandTurn = (int) random(2); //random number between 0 and 1
if(randTurn==0){
dir=Right;
 }
else{
dir=Left;
 }
}

//Chooses a random amount of steps to walk
//Anywhere from 1 to 5 steps for forward motion
//If turning between 1 and 3 steps
voidrandomSteps(){
if(dir==Frwd){
randSteps = (int) random(1,5);
 }
else{
randSteps = (int) random(1,3);
 }
}

//Wandering function that uses random functions to
//update the direction and number of steps to walk
```

```
void wander(){
randomDir();
randomSteps();
MotionF(randSteps,0);
}

//Obstacle Ahead function
//Stops for one second=> Reverses for 3 steps=> Turns randmoly
voidObstAhead(){
dir=Halt;
MotionF(1,1);
dir=Rvrs;
MotionF(3,1);
randomTurn();
randomSteps();
MotionF(randSteps,1);
}

//Obstacle to the Left
//Stops for one second=> Reverses for 3 steps=>Turns right for 2 steps
voidObstLeft(){
dir=Halt;
MotionF(1,1);
dir=Rvrs;
MotionF(3,1);
dir=Right;
MotionF(2,1);
}

//Obstacle to the Right
//Stops for one second=> Reverses for 3 steps=> Turns Left for 2 steps
voidObstRight(){
dir=Halt;
MotionF(1,1);
dir=Rvrs;
MotionF(3,1);
dir=Left;
MotionF(2,1);
}

voidreadSensors(){
  //Read Obstacle sensors
```

```
digitalWrite(12,HIGH); //Drive pin 12 high with internal resistor
   buttonState11 = digitalRead(11);
   buttonState12 = digitalRead(12);
   buttonState13 = digitalRead(13);
   //Read Beacon Sensors
RIRval = analogRead(RIR);
LIRval = analogRead(LIR);
   //Read CDS
CDS_val = analogRead(A3);
}

//Arbitrates motion
voidMotionF(int steps, int check){
for(int i=0; i<(steps*100); i++){ //for loop to go x steps
cycle=(cycle+1)%100;          // 100 step cycle
readSensors(); //Update sensor data
//Checks fron IR or bump; break if interrpt is needed.
if((buttonState11==LOW or buttonState12==LOW or buttonState13==LOW)
and check==0) {
break;
   }
if((RIRval<= 900 or LIRval<= 900) and check==0){
break;
   }
  //React to CDS
if(CDS_val> (CDS_amb + (CDS_amb))){
break;
  }
   //Checks Direction and runs movement routines
if(dir==Frwd) goFrwd(cycle);
else if(dir==Rvrs) goBack(cycle);
else if(dir==Left) goLeft(cycle);
else if(dir==Right) goRight(cycle);
elsegoHalt();
  //Write position to Servos
LeftS.write(Lpos);
RightS.write(Rpos);
WaistS.write(Wpos);
delay(10);  //delay so each step is 1 second
  }
}
```

```
void loop(){

// see if there's incoming serial data:
if (Serial.available() > 0) {
// read the oldest byte in the serial buffer:
incomingByte = Serial.read();
if(incomingByte== 'a'){
Serial.println("Looking for beacon");
dir= Halt;
MotionF(1,0);
dir= Right;
MotionF(1,1);
if(RIRval<= 900 or LIRval<= 900){
Serial.println("Beacon Found");
  }
MotionF(1,1);
if(RIRval<= 900 or LIRval<= 900){
Serial.println("Beacon Found");
  }
MotionF(1,1);
if(RIRval<= 900 or LIRval<= 900){
Serial.println("Beacon Found");
  }
MotionF(1,1);
if(RIRval<= 900 or LIRval<= 900){
Serial.println("Beacon Found");
  }
 }
else if(incomingByte== 'd'){
Serial.println("Stop Dance time");
dir=Halt;
MotionF(1,1);
dir=Right;
MotionF(3,1);
dir=Halt;
MotionF(1,1);
dir=Left;
MotionF(3,1);
 }
else if(incomingByte== 'h'){
Serial.println("Pause command recieved");
dir=Halt;
```

```
MotionF(3,1);
 }
 }

if(Start==0){
  buttonState11 = digitalRead(11);
if(buttonState11==LOW){
   Start=1;
  }
delay(1000);
}
else{
readSensors();
 //React to CDS
if(CDS_val> (CDS_amb + (CDS_amb)/4)){
Serial.println("Darkness Found");
while(buttonState11==HIGH){
dir=Halt;
MotionF(1,0);
   }
  }
 //React to Obstacle sensors
if(buttonState11==LOW) { //Read SHARP IR to check ahead
ObstAhead();
 }
else if(buttonState12==LOW) {//Read Right Bump Switch
ObstRight();
 }
else if(buttonState13==LOW) {//Read Left Bump Switch
ObstLeft();
 }
 //React to IR becon
else if((RIRval<= 900) and (LIRval<= 900)){
Serial.println("IR Beacon found ahead");
while(buttonState11==HIGH){
dir=Halt;
MotionF(1,0);
   }
  }
else if(RIRval<= 900){
Serial.println("IR Beacon found");
dir=Halt;
```

```
MotionF(1,0);
dir=Right;
MotionF(1,1);
while(buttonState11==HIGH){
dir=Halt;
MotionF(1,0);
    }
  }
else if(LIRval<= 900){
Serial.println("IR Beacon found");
dir=Halt;
MotionF(1,0);
dir=Left;
MotionF(1,1);
while(buttonState11==HIGH){
dir=Halt;
MotionF(1,0);
    }
  }
else {
wander(); //Robot wanders using random funcions
 }
}
}
```