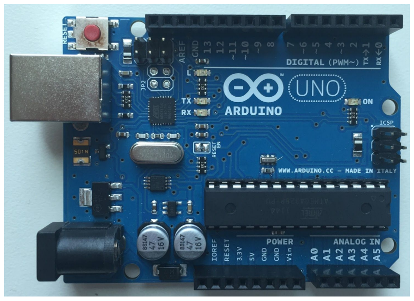
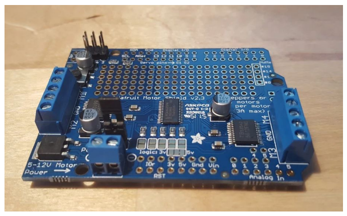
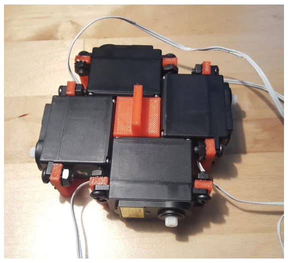
Rudimentary Swarm Robotics
Josiah Hamid-Khani, Thomas Keller, Matthew Sims, & Isaac Swift
Episcopal School of Dallas, josiahhk@gmail

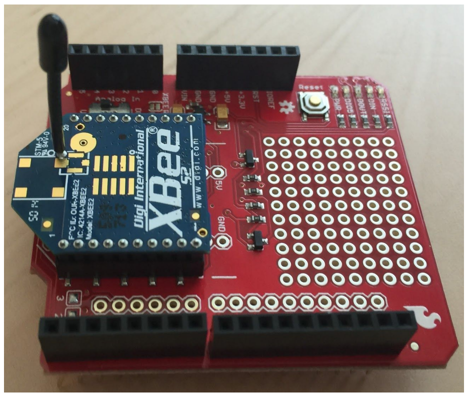
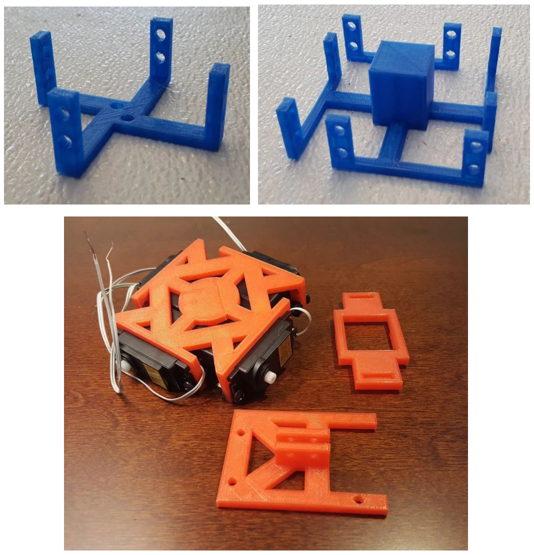
Rudimentary Swarm Robotics

Project Description

The concept of swarm robotics is dividing up a complex task among many simple, small robots working in tandem instead of creating a single unit of immense processing power. This year, the Episcopal School of Dallas's Honors Advanced Computer Science course has created a project dedicated to exploring robot swarm technology. There are many different phases of the project: researching all the components and learning how they work, designing and building the chassis, connecting an Arduino to working sensors, programming the robots for movement, to communicate with one another, and then utilizing these aspects to allow the robots to coordinate with each other and work together. From the outset of the year, our goal was to deliver "three identical, programmable robots with coordinated motion, environment sensing capabilities, and communication hardware."

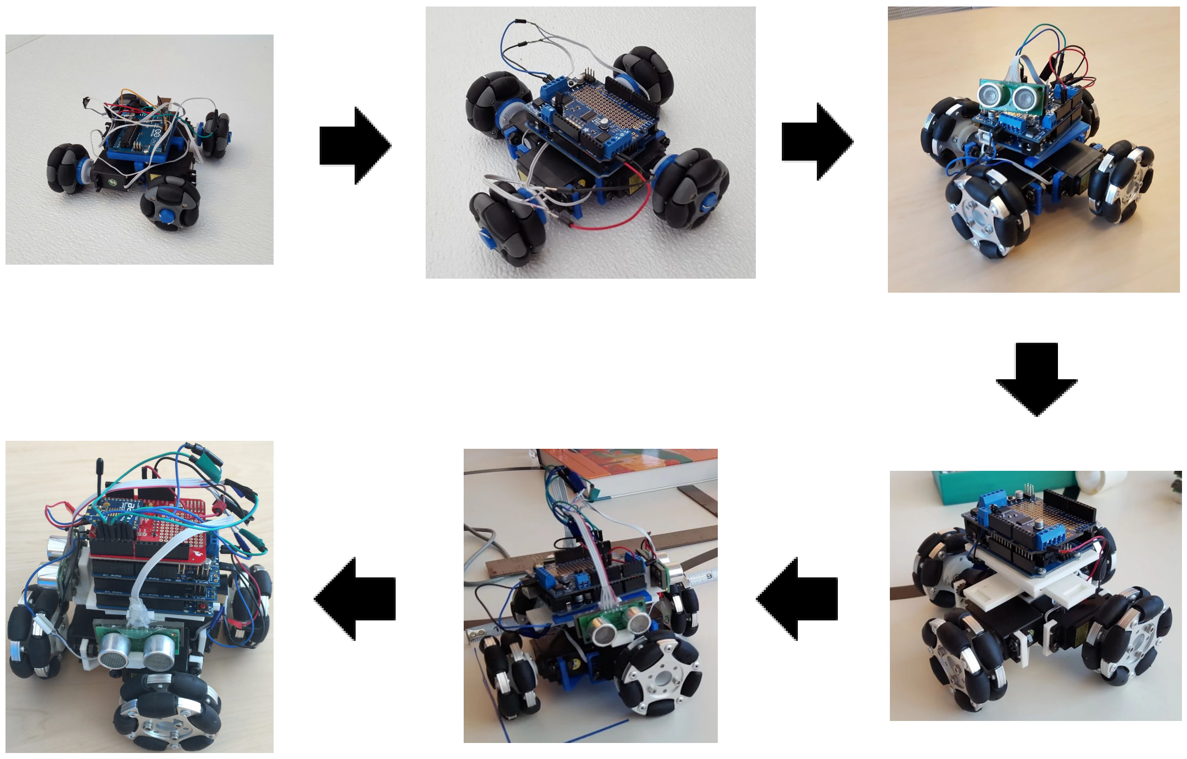
Hardware

Arduino platform For the main circuit board and platform for our robots, we went with the Arduino Uno for its small size yet robust capabilities.	
Movement For the movement mechanism we chose to use omni wheels (wheels that have smaller, free-spinning wheels lining the outer edge) to allow free movement in every direction of the x-y plane without having to turn.. They were powered by spare Botball motors, and Adafruit Motor Shields v2.3 were used to handle all four motors for each robot.	 

Sensors To understand an environment, we needed sensors which could tell how far away an obstacle is. To do this we used the HC-SR04 Ultrasonic Sensor, as it was the most consistent and accurate of all the methods we tested. We positioned one on each side of the robot.	
Communication The robots use the Series 2 Xbee to communicate and coordinate their actions. A SparkFun XBee Shield was used to attach it to the Arduino platform.	
Chassis Given the wide variety of components used in the robots, we decided to model a custom chassis in Autodesk Maya, and 3D print it. The chassis went through many design changes, and the final design is expounded upon in the Final Design section.	

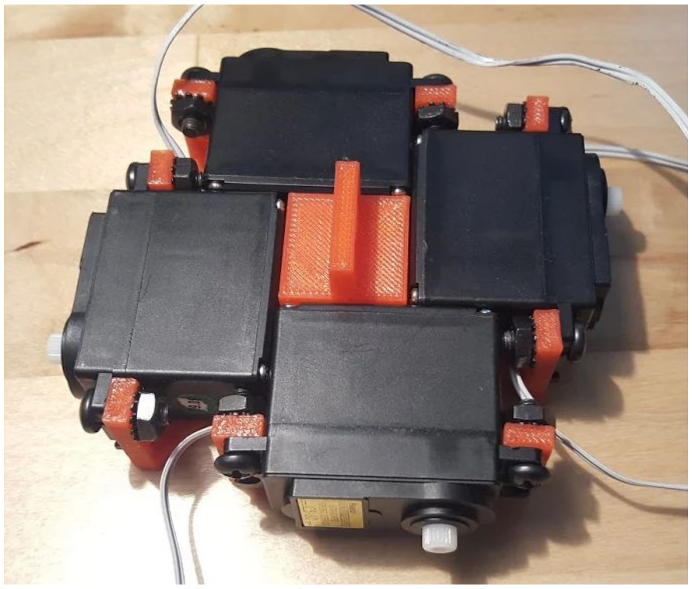
Design & Prototyping Process

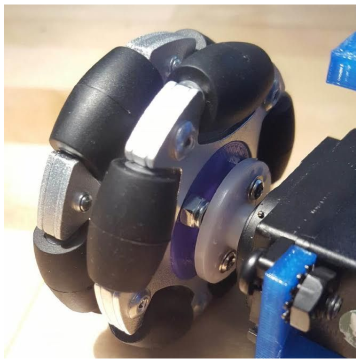
We went through many iterations before arriving with the final product, as displayed in the diagram below. Our initial prototype had an extremely unstable platform, jerry rigged wiring, and unideal wheels in comparison to the robot's weight and size. But through testing and redesign lead to a final product with economically positioned hardware built on a sturdy base, laced with clean and effective wiring, and on wheels of appropriate size for the rest of the robot.



Final Design

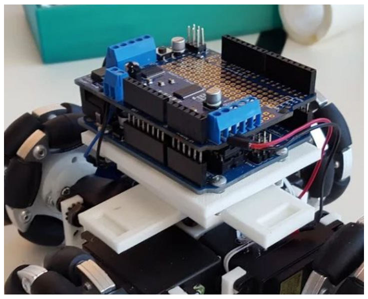
Starting at the base, the omni wheels are designed in a spiral-esque fashion in order to utilise their strength of each wheel having two directions of motion while at the same time maximising stability by positioning them on the corners. Since the motors are very large in comparison with the size of the robot, the bottom chassis allows for the motors to be as close as possible by actually touching and supporting each other, while still offering support from the 3D printed structure itself and the ability to connect further modules on top.



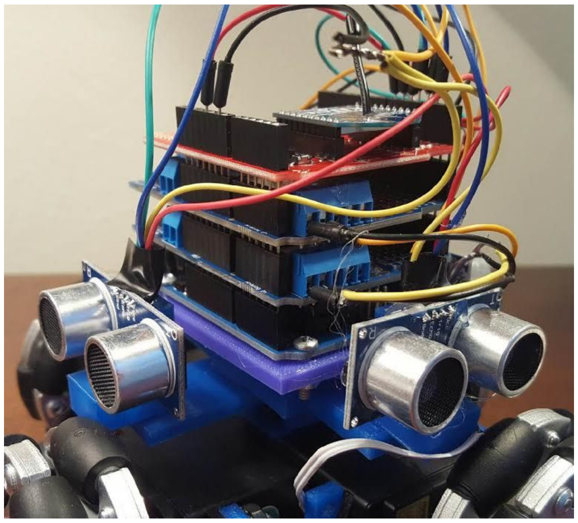


One small design aspect worth mentioning is the 3D printed structure connecting the omniwheels to the motor horns. This design went through many iterations from sticking all the way through the center of the wheel to just being a small disk. The disk allows for the connector to fit in the outer indent of the wheel and have the bolts that hold the wheel together also hold the disk in place. Then, a protrusion from the disk is screwed directly into the motor horn. With the motor horn connected, a hole through the wheel and disk allows for the horn to easily be screwed into the motor and securely attached.

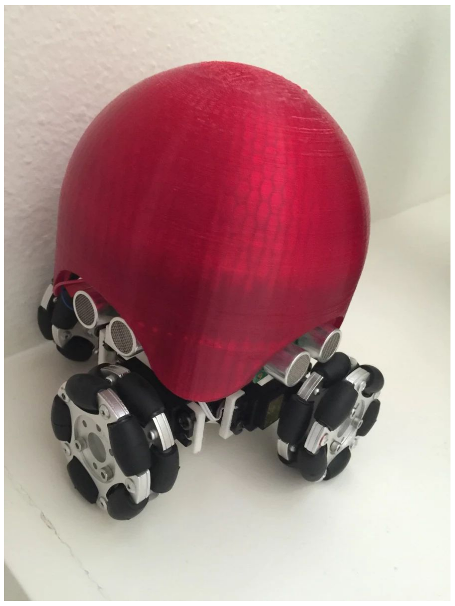
With the base of the structure out of the way, we examined how we wanted to mount sensors and the Arduino and its shields. We decided to manufacture a modular system through which connectors were uniform. This allowed us to initially have the Arduino mounted to the base then, when we were ready, to add the sensor mount in between the two components without reprinting old parts. Although we didn’t need any more sensors or arms for our purposes, the modular system also allows for different or differently mounted sensors or any extensions to be mounted easily for whatever purpose the robot may find.



The sensor module is just above the base chassis and allows for the secure mounting of four HC-SR04 ultrasonic sensors—one on the center of each side. This module is designed to mount the sensors as low to the ground as possible to pick up any low-lying obstacles, while not expanding the radius of the robot. They are mounted just above the wheels through a six-piece 3D printed connector. For extra security, the sensors are hot glued to the 3D printed structure. Above the sensor module is the brain, the Arduino module. The Arduino module consists of a basic 3D printed connector allowing an Arduino to be screwed to the base and as many shields as need to be stacked on top. This module has been generally considered the top of the robot, since its stacking capability is crucial and unknown. That said, we always had plans for some sort of cover to give the robot a sort of personality beyond a circuit board and a bunch of wires.



The “cap,” as we named it, has been debated in its function and aesthetics, but there are several features that it incorporates beyond its original design. The cap includes one mount for a sensor in the back of the robot, where the wire for the battery must be plugged in, since the sensor module mount had interfered with having both the wire plugged in and sensor mounted at the same time. As for personality, we settled on having each robot be a different Pac-Man Ghost (Inky, Blinky, and Clyde). The higher-mounted sensor from the cap serves as eyes and the necessary rolls over the wheels serve as the wavy nature of the bottom of the ghost.



Programming

Movement

After doing a bit of research to learn the basics of coding on Arduinos (while the official Arduino website did have some useful information, we had to resort to many forums and blog posts by Arduino enthusiasts to learn find some crucial information), going through a myriad of hardware revisions, and unintentionally tripping a slew of Arduino errors, we built the fundamental drive code for our robots. We built a drive() method that uses the Adafruit Motor Shield library to move the robot in any vector on the x-y plane using simple trigonometry, based on a direction on the x-axis (left, right, or none), power for the x-axis (0-255), direction on the y-axis (forward, backward, or none), and power on the y axis (0-255). This allows us to direct our robots in precise trajectories in a single line of code.

Sensors

While figuring out how to wire the sensors took some research and testing, the programming aspect of the sensors was fairly simple. We used basic Arduino methods to create a snesnor() method—named so because of a running joke within our team—so that we could read the distance from the nearest obstacle to any of our robot’s four sides on demand.

Avoid

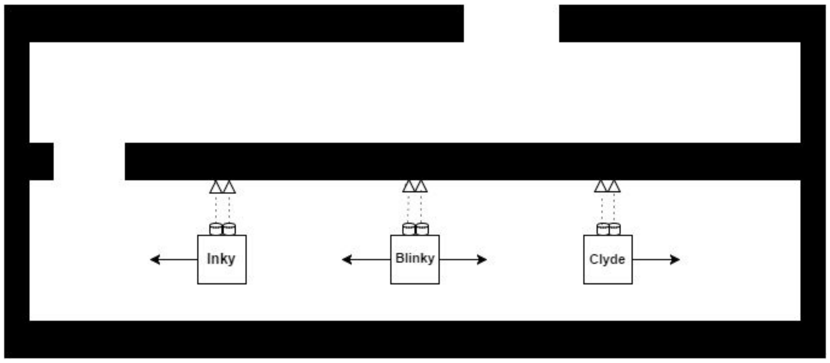
To test these previous methods, and as a proof of concept, we began to develop the avoid() method. It was an algorithm to let our robot traverse forward, despite what obstacles might be in the way (within reason, if there happens to be a large pit or hole, our robot's in trouble). This is the one-robot version of our final plan for our system of robots. Since Arduinos loop through their code indefinitely, the basic concept was to check if there was something in front of the robot, and if not, drive forward. If there is something to the front, it checks if there's an obstacle to the right. If not, drive to the right until there's nothing in the way to the front. If there is something to the right, drive to the left and try the same kind of thing. If all else fails, drive backwards until a sensor opens up. Since the Arduino is constantly looping, it runs through this fast enough to cause seemingly seamless, constant movement.

Getting successful communication

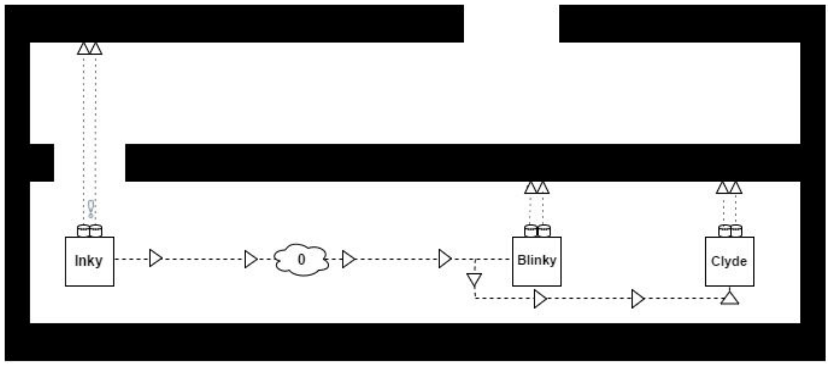
While communication was one of the most frustrating aspects of the code to complete, it was not overly complicated. The difficulty came from the fact that there is no comprehensive "manual" when it comes to coding Arduinos. Instead, we had to scour the internet to find people doing similar projects, and extrapolate how to code Xbees based on how they used them. After calibrating the Xbees in X-CTU, we were then able to get them to send basic data packages to each other on command.

Swarm Logic

Our network of robots can perform the function of the avoid() method much faster than a single robot, and can also outperform a single robot with equivalent to their composite processing power due to their ability to cover more ground than a single unit possibly could. They scan the area in front of them looking for a passage through, with the two on the edge pushing outward, and the middle one roving back and forth, searching the middle area and after that rechecking the area that the outer two had scanned. Once a bot finds an opening, he sends a message to the other two bots (he sends a 0 if it's the left bot that found an opening, a 1 for the middle bot, and a 2 for the right bot). In the example, Inky—the bot on the left—finds an opening and sends a 0 to the other bots.

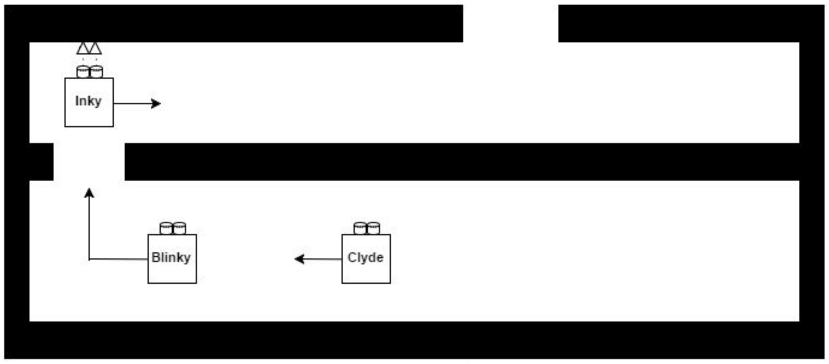


Step 1

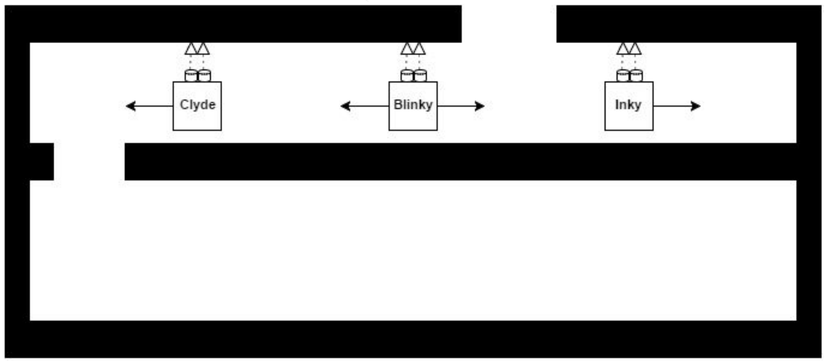


Step 2

Upon receiving the 0, the other two bots immediately set about following inky through the opening. Once they are all through, they line up again and begin to search for another opening. Based on which robot led the way through, they will know what order they are now in, and will modify what values to send out upon finding an opening accordingly.

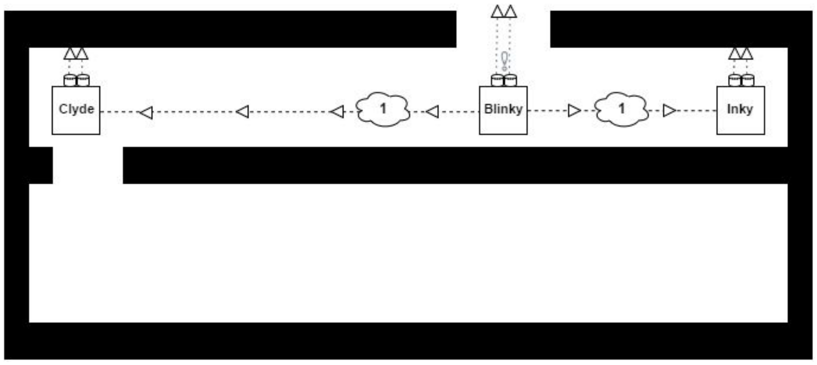


Step 3

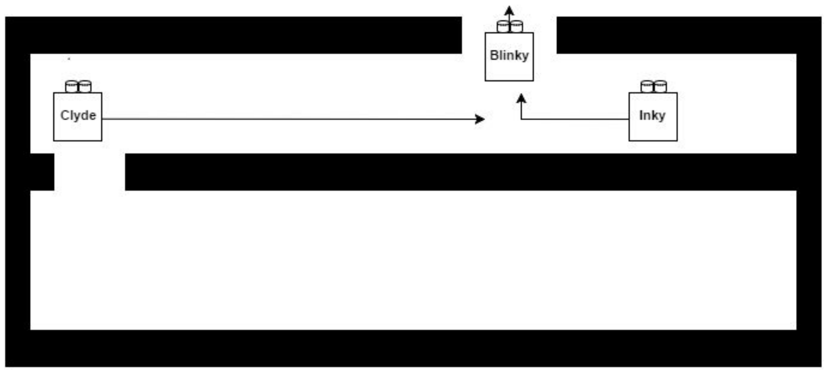


Step 4

Blinky, upon finding an opening, sends a 1 to the other bots (as he is now the middle bot), and the other two robots follow him through—Inky first, as he is on the right, and in the event of the middle bot finding an opening, they are programmed to go middle bot, right bot, then left bot.



Step 5



Step 6

In Summation

Our swarm based traversal system, while rudimentary, demonstrates the basic logic and promise of swarm robotics. A similar system could be used to map out terrain, search for victims in a natural disaster, or scour a minefield. Dividing a difficult task among a group of robots of low to moderate intelligence is much more efficient in its endgoal and cost effective than designing a single unit of immense processing power.