University of Florida

The 10 Stooges:

The Intimate Life of a Colony…

Dr. A. Arroyo
EEL 5666
IMDL

Jose Garcia-Feliu

Table of Contents

Abstract

The objective of our design is to make a colony of robots that will simulate many behaviors. We intend to show how multiple robots can accomplish tasks that single robots cannot. Some behaviors that we will show are swarm, robot communication, industrial optimization, self-adaptation, basic learning, etc.

Introduction

We will go through all of the steps required to design, assemble, program and activate these robots. We decided to make ten robots to demonstrate the versatility and variety of behaviors that each can accomplish. Each robot will be able to communicate with each other and decide to follow the desired behavior or to ignore it, if something else has higher priority. We chose to design and program these robots the same, so that a multitude of robots could be built, if desired, and added to the field without any further modifications. The advantage is a more flexible assembly of the robots. However, the disadvantage that this format brings is that because they have the same behaviors, very complex combinations can be achieved. Below, we will describe all the systems and behaviors encountered and expected from these robots. We will also write a program, which will be used for the 1 queen, 9 workers, 1 soldier, and 1 probe. We will make the program very flexible and adaptable to each robot. This also facilitates the writing of functions since most robots will share them. Furthermore, we will explain the relations between their behaviors and how the combination of sensors affects this complex set of actions.

Executive Summary

This project was very successful. We accomplished the goals of making robots that would have multiple behaviors and also to show them. At the beginning of the presentation, we also configured the robots to perform following and swarm behavior only if told to do so. During start up, the robot calibrates and performs self-diagnostics; it is during this stage that we push the bumpers to tell it to go to follow mode. The leader is also chosen in this manner and so are the followers. We lined them up and after the leader was ready, the line of ten robots moved on. While the leader performed collision avoidance, the group followed. After some time, the group stopped and a signal propagated across the group to let them know to get ready for swarming. The robots wandered until they saw signs of another. Each is designed to communicate or perform collision avoidance at the sight of another robot. If communication were chosen, the robots would try to align so that the IR data communication would begin. We had a bit of a problem getting the robots to line up correctly. This resulted from the robots using a combination of IR and bumpers to find the other. Because the IR saturates at close range, the alignment was almost left up to the bumpers being pushed. This made it very hard to communicate. During the process, the robots would take readings from their environment and check their battery status. Based on these and any communicated behaviors, the robot will choose its mood. The possible moods are happy, angry, frightened, and sleepy. Because so many behaviors and choices were left to the robot to make, the actions of any robots could not be predicted. This type of random programming

is very useful in creating life-like communities of robots. One of our best findings were the designed functions to deal with multi-robot projects. Of course, these techniques can be applied to any single-robot program. Our first function lets the robot know which one of four different types of robots it is. The second initial function performs a self-diagnostic of any sensors it may have. The third function performs a calibration of lower and upper limits of the sensors found. The other aspect of this latter function tells the robot to continually calibrate the robot's sensors for as long as it is running. This result in a robot that adapts and better learns about its sensors the more it runs. Here is where experience pays off. The last aspect of our project deals with having an input to the swarm. We used a queen robot to input desired behaviors to the group. Unfortunately, the robot has the last word whether to change to the new behavior or to ignore it and continue using its original one. We see this handy specially when the batteries are low and the robot is told to become happy, where battery usage is greatly consumed. In this case, the robot will actually decide that the suggested behavior does not provide the robot and advantage and therefore it is ignore. The last aspect of this project involves the basic design of a basic operating system, which could be adapted to any robot to perform basic self-diagnostic and self-calibration to learn more about the robot it is in and to use such sensors for the proper mobility of the robot.

Integrated System

Once all of the hardware is tested and put in the correct place, we will use the program to control the behaviors of the robots. The

board is in charge of managing, 1 sonar for the queen, 1 sonar detector for the workers and soldiers, 1 sonar beacon for the beacon, 4 contact sensors, 4 IR sets for detection and communication, a status LED segment display, a battery-checker circuit, and a buzzer. Each behavior will be programmed separately until it works properly. Since many behaviors are incorporated, we will resolve any problem they may have with other sensors and their possible new behavior. Refining the behavior interaction can be done as the robots achieve a working format. The behavior interaction can be seen by the following flow-chart in Figure 6, in the appendix section at the end of the report. Here we see how all behaviors are related to each other and when does a particular behavior is allowed and when it is not. This combination of behaviors and reactions will permit each robot to perform the objectives described in the introduction. One of the most difficult and challenging problems will be with the sonar. Because of its long-range capabilities, incorporating the sonar to the design will increase the chances of, not only error, but also of flexibility requirements. Because each robot will have 14 sensors and 2 motors, coordinating all of them will be a real challenge. For this reason, we chose the TJ Pro board as our desired processor board. This board offers plenty of flexibility, and possesses many of the needed connections and pins as part of its layout. Being the purpose of this project for the robots to interact with each other, it is a requirement that the robots be agile enough but also well equipped with many sensors. The TJ body offers a simple structure with many levels for different sensors. Proper integration of all components, including correct behaviors, is vital to our success in this project.

Mobile Platform

The body of our robot is the familiar Talrik Junior or TJ for short. A picture of the TJ can be seen in Figure 7 in the appendix section. The TJ body offers the strength of a hard and simple frame, and a horizontal circular cap, which will later help to support the carrying base of the soldier robot. But most important of all, it offers savings on weight. We intend the robots to be agile and fast. The structure has enough space for two servos, which will drive the robot using 3-inch wheels. It also has a surrounding flexible bumper used to detect physical contact with the field's wall, obstacles or the other robots. Inside the TJ structure, we can also see the processor board, which operates all of the sensors and gives the robot its commands and behaviors. Because all of the robot's wiring and devices are hidden on the inside, we were able to spray paint the frames and make the robots presentable. Also, because the purpose of the game is for spectators to constantly watch them, the more uniform and smooth looking appearance makes them, the friendlier and pleasing to the eye. We cut out the layout of the TJ the first day in lab. Each sheet of wood has enough pieces for two TJs, so we got 5. We used an upside down sander to smooth out the corners and sides of the newly cut parts. We also used super glue to hold the pieces together. After a minute of drying, we applied more glue to the joints to strengthen them. Once the body was glued, we took the robots outside and sprayed them. We applied 5 coats of primer, 2 coats of chrome and 2 coats of a sealant. We also learned that the metallic spray can easily be damaged with fingerprints, so the coat of sealant is very important to preserve this look. One tip to remember when spray painting is flip the can upside

down and spray until no paint comes out. This prevents the clogging of the can. Figure 7 in the Appendix shows the final robot fleet.

Actuation

The robots use two servos for mobility. They also used a round piece of plastic to minimize the weight and to serve as the third point of support. A picture of the servos can be found in Figure 8a, under the appendix section. The board, using a fairly simple program, will control these devices. The program will tell the servos what speed to rotate by sending it a number. We will use the command *"motorp(a, b);"* where 'a' is the number for the left or right servo (0 or 1). In turn, 'b' will be the relative speed of the motor. We hacked these servos to make them work as motors by physically breaking off the connection between the final gear and the position potentiometer. Each servo includes a controller that tells the current position and the desired one. The farther the degree difference the faster it will rotate. Therefore, the variable 'b' used previously is used to tell the servo how fast to go. Just like the TJ body, these servos are also lightweight and very efficient. The servo motion will help the robot in the multi-behavior pattern by turning the entire robot in the needed direction. Mounted onto the servos are two big 3-inch wheels. These will help the robot have a smooth ride and provide enough underside space for unexpected obstacles. Because of its lightweight characteristic, each robot will be able to smoothly drive over objects without too much trouble. Another resulting aspect of the servos is that they are not exactly calibrated to zero as their stopping motion. Therefore, we had to make a calibration system so different clones of

the robot can use the same program without the need of further changes. We approached this problem by first finding out the exact value for which each robot would be at zero. To our surprise, all six servos had about the same value. We then proceeded to make global variables at the beginning of the program where these values could be entered. By adding these values to the desired values, we accomplished to send the servo controller boards the correct values. Most important of all, these variables can be changed in case the robot re-calibrates or if another robot having different values is used.

Sensors

The robot will be loaded with a number of sensors to help it move smoothly around the field. A sample IR detector is shown in Figure 8b in the appendix section. Along with these sensors, each unit will have touch sensors (bumper), in case the robot makes contact or an unexpected obstacle is present. This is necessary because the robots turn and move at great speed. A sonar emitter and detector is also used to approximate distance from the queen and to approximate the location within a specified area. Another sensor we have added is the battery-checker circuit. It consists of a voltage divider connected from the battery to one of the analogs. All of the robots have this circuit connected across PE1. To aid the spectators' view the status of the game, each robot contains a 7-LED status segment connected to an 8-bit flip-flop chip. The robots will also be equipped with a buzzer, which will aid announce stages to the audience. The same piece will also indicate when one of the robots is low on power.

The sensors used by our robots will enable them to view the world around it in a general but limited way. The sensors will give information with respect to objects in the proximity and moving objects in front of it. Because integrating so many sensors in one robot might overwhelm the processor, we had to rely on better programming and logic to device a program that will read sensors and using data only when it needs it. In this paper, we will see how we adapted each sensor to the robot and how we incorporated them together.

Bumper Sensors

The bumper sensor is a series of switches positioned around a movable bumper frame around the head of the robot. We can see their location in Figure 1. When the robot bumps into an object, the switch closes and the processor can then inspect the analog-to-digital converter PE0. The TJPRO board is setup so that four switches can be individually connected. When PE0 is checked, each switch is differentiated in steps. In our robots, we only used all four switches, one in the rear and three connected in the front. Because these switches are positioned on the top section of the TJ, objects smaller or taller than that height will not make contact and therefore will not be discovered. This is one of the assumptions that obstacles and other robots in the field will be about the same height. Knowing

this, the bumper sensors are used as a last resort in avoiding obstacles in the near proximity. Overall, the bumper will not be used for the main purpose of the game, but becomes important, since during communication, the bumper might give feedback about the other robot and its possible location.

Infrared Sensors

Figure 10: IR sensor hardware

We are using five infrared sensor and emitter pairs on each robot. The location of the IR sensors is drawn in Figure 2. The IR emitters are connected to pre-wired connections of 40KHz signals. These can individually be controlled by writing to the memory-mapped output (0x7000), and sending a value 0xff, where ff can be replaced by the actual combination of sensors desired to be turned on. We have three IR placed in front of the robot below each of the front bumper switches. We also have another positioned in the rear. Each connected to PE2, PE5, PE7 and PE3, respectively. Every detected value has a range from approximately 80 to 130. The A-to-D converters give a digital value to the processor about how far an object would be from the robot. At the beginning of the main program, we call a subroutine called calib_IR(). This routine makes the robot spin slowly and take measurements of the lowest value of the IR. This helps set the lower bound for each IR set. This self-calibration step is needed since each sensor does not necessarily possess the same characteristics as others might. Another important function in our program is the self-diagnostic routine. This particular part of the program aids in

detecting sensors, which are not working, or not working correctly before the robot even begins moving. Again, very important to aid the debugging process and to handle multiple robots.

Sonar Sensors

The final sensor used by the robots is the sonar. Their location can be seen in Figure 3. The sonar is a simple device that transmits ultrasonic waves. When the waves encounter an object, they bounce back and are detected. The sensor can then determine how long it took the wave to travel. Emitting these waves without moving the robot is essential to the accuracy of the sonar. To help avoid false detection, we will give the robot a certain degree of error. The signal processed by the robot will be a number from 0 to 255, just like the IR. But because we added a low pass filter to the output signal, we got a value from 1 to 60. The advantage of this extra step is to convert the value, so that the higher the value of the reading, the closer the sonar waves. The sonar emitter will be connected to the 40KHz pre-wired outputs in the TJPRO board in the queen robot. The emitter will always be turned on and used by the other robots as a way to locate their distance from the queen and to reach certain sectors of the field. The detector will detect depending on the particular behavior. If the behavior demands a high degree of accuracy for position, the robot will use the detector more frequently. The detector will be connected to PE4. The code will constantly check this location for possible readings.

Battery-Status Sensor

Our robots are also equipped with a battery-checker circuit. This will greatly help us in determining robots with low power. When handling 10 mobile robots, 1 stationary robot and 1 probe, each demanding 6 AA batteries, for a grand total of 72 batteries, this circuit can only be but an angel in the nightmare of debugging. We have added behaviors to drive robots away from the queen and to call out the supervisors (us) for help regarding their low battery condition, before they start to malfunction. Another future addition to this project could include self-recharging robots. One even further addition is while a robot recharges, another robot takes over the task, where the previous left off.

Sensor Experiments

Bumper Experiment

To test the correct working order of the bumper sensors, we wrote a simple program which would call the subroutine bumper(). Table 1 shows the results obtained from this experiment. This routine would check the bumpers for any sign of contact. If any of the switches were detected to be close, then the robot would move in reverse and rotate in the opposite direction. This code checks for any of the bumpers to be closed. If any of them has contact, the routine would play out its instructions. Otherwise, it would do nothing. The bumpers are located at a height of 3.25 inches from the ground. If we assume the

ground to be flat, then any contact between robots or with objects within the playing field should, at worst case, touch at that height.

Analog Value	Front Center	Front Left	Front Right	Rear Center
0	X			
43		X		
79			X	
21				X
126	X		X	
59	X	X		
101		X	X	
110	X		X	X
139	X			X
150		X		X
132			X	X
162	X	X	X	X

Table 1: Bumper results

Infrared Experiments

The infrared sensors were tested similarly. To protect the sensitive IR, we enclosed them underneath the upper body. This would ensure that any bumps would not directly affect the sensors, or the calibration. We wrote a simple program routine for obstacle avoidance. This code is shown in Appendix B. This code shows how as each sensor senses an obstacle, the motors move the robot around it. Below, we show Table 2 of the IR sensors and what the robot does when an obstacle is directly in front. The IR sensors were located slightly below the bumper switches at a height of 3 inches. Anything much higher than 3

inches would not, otherwise, be detected by the sensors. The forward and rear looking IR reacted at a distance of 10 inches. This relatively small distance was intentionally chosen to maximize the space within the playing field, as well as, to give the robots a really closed feel when detecting each other.

Figure 9: Collimated IR sensor

Distance (inches)	Front Center	Front Left	Front Right	Rear Center
0	127	127	127	127
2	117	126	126	126
4	95	109	119	102
6	91	101	108	100
8	89	93	101	98
10	87	89	93	91
12	87	88	91	90
14	87	87	88	87
16	87	87	87	87
18	87	86	86	86
20	86	86	86	86

Table 2: IR sensor table

Sonar experiment

The sonar was used to measure distance. We performed a series of simple tests. The results are shown in Table 3. We put the robot in front of the queen robot shooting at intervals and programmed it to sense sonar measurements. At the end of the measurements, the robot either would light up the segments from 1 to 6 if there was sonar

detected. The first would be lit if the robot detected a signal with a value from 10 to 19, the second if the value was from 20 to 29, etc. The results from the sonar test showed that the sonar would work at a distance of over 3 feet, which was more than enough to give each robot enough capability. Because the sonar is a wave, the calibration routine did not need to adjust the sonar. We positioned the detectors on top of the robot on a tower aiming forward. This will give the robot the maximum range of detection, while easily accomplishing the behavior of approaching the queen. The circuit design for the sonar emitter is shown in Figure 4 in the Appendix and the circuit for the detector in Figure 5.

SENSOR	**EMITTER**	**DISTANCE(ft)**	**RESULT(# LEDs)**
Sonar	off	any	0
Sonar	on	0.5	6
Sonar	on	1.0	5
Sonar	on	1.5	4
Sonar	on	2.0	3
Sonar	on	2.5	2
Sonar	on	3.0	1
Sonar	on	3.5	0
Sonar	on	4.0	0
Sonar	on	4.5	0

Table 3: Sonar sensor table

We can also see in Figure 4, the layout of the schematics used for the sonar emitter.

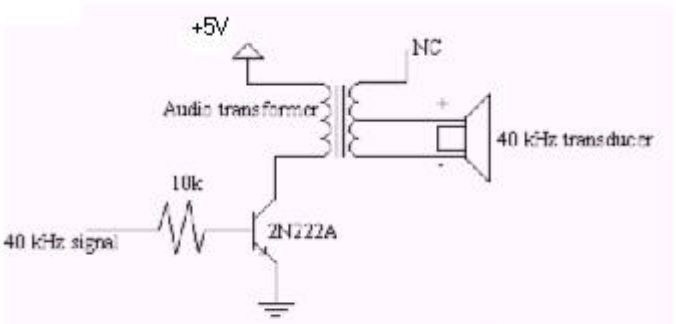

Figure 4: Sonar Transmitter

In Figure 5, we also show the circuit layout of the sonar detector.

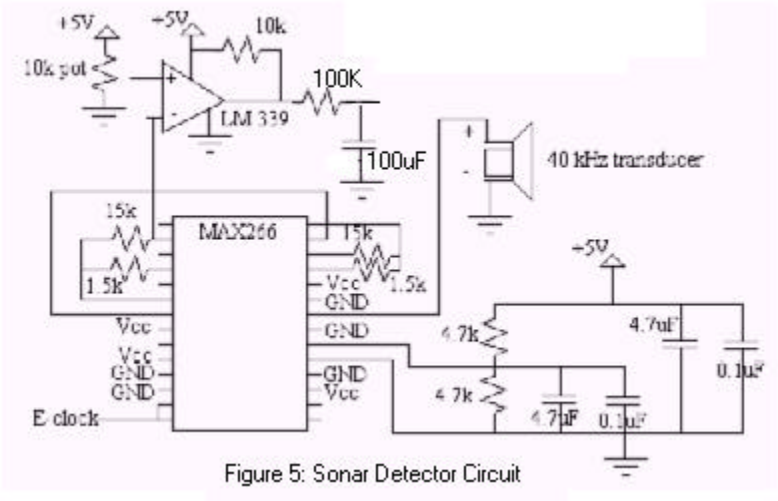

Figure 5: Sonar Detector Circuit

Behaviors

The robots will have quite a few behaviors, which we hope to combine to make more complex behaviors. We have divided these behaviors into two major groups: individual and group behaviors.

Individual Behaviors

Single robots can do individual behaviors, while group behaviors can only be accomplished by two or more robots. The first individual behavior performed by all robots is self-identification. Because all of the robots use the same program, they will need to know which functions and variables to use for their specific purposes. An individual behavior is to avoid physical contact. It accomplishes this by using the four bumper switches. Another basic behavior for the robots will be collision avoidance. The three IR sets located in front of the robot will aid in this task. A very important behavior occurs at the beginning of the game. The self-diagnostics routine makes sure that all components of the robot work properly, before the robot starts its task. In conjunction, the robots rotate at a slow speed and calibrate their IR. This is accomplished by taking readings of the environment and comparing the lowest possible value to a predetermined variable. If the value is lower, then that variable is updated with the new value. In essence, this calibration helps determine the changes in infrared ranges. Another is to navigate from or to a specified sector. This is accomplished by measuring the strength of the wave sent by the sonar from the queen. Another resulting behavior is that when the queen calls out for any of the robots to attend to her, the robots can directly go to her by following the strength and direction of the sonar. At the same time, if an object is found in the

way, like another robot, the program can make the robot go around it and afterwards relocate the queen signal. Another important behavior is the constant look out for low power. If low batteries are detected, the robot will then locate the queen, and go as far as possible from her and inform the supervisors (us) of the battery condition. Another important aspect of the individual behaviors is to communicate data between them. While communication is part of the group behaviors, assimilating the data and performing the desired task is part of the individual behaviors. While performing these behaviors is dependent on the robot, another aspect of our program to give behaviors certain priority, so that the robot can "choose" to ignore the behavior based on, not only priority, but also power level. This means that if the robot's batteries are low, the robot will ignore the tasks that use most power. Along with self-identification, mentioned above, the robots will also be able to identify their own kind. This will show that they are aware of certain sensors and understand their surroundings. Below we show a summary of the behaviors in Table 4.

Group Behaviors

Group behaviors are those that are accomplished by multiple robots. A basic behavior is to follow the leader. This is accomplished by following the rear emitter from the robot ahead. The leader will, instead, follow a remote control device. Another important behavior is the ability to communicate data between robots. Through this behavior, other behaviors can be accomplished. One being, the recruiting of other robots to perform the same behavior. Communication between the robots and the queen is similar. Swarm and

dissipate can also be done through this method. By informing all of the robots to do the same behavior, swarming and dissipating is possible. Using the probe, the soldier robot can mark positions and objects. A future addition to our project would be for the worker robots to find their exact location by using the sonar signals from the probe and the queen. Then passing this information to other robots or even the queen. The queen could even map the boundaries of her kingdom. Another behavior we plan on demonstrating is the sleep routine, where the robots will sleep until the queen advises them on waking up. This could be used as a battery saving option that each robot can implement on its own if no activity is detected for some time. One added bonus we plan on achieving is to place three robots inside the field in different locations. Each one of these robots will represent a machine. Then we will add the rest of the robots along. When one of the three machine robots call a robot for assistance, the wandering robots will approach that machine robot and pick up an imaginary cargo. They will then look for the next machine and drop off the imaginary cargo with it. This cycle will repeat itself. As robots detect their battery to be low, they will abandon their task, hopefully for another robot to take over. Below we show a summary of the behaviors in Table 4.

Individual Behaviors	**Preferred Sensor**
1. Avoid Physical Contact	(Bumpers)
2. Collision Avoidance	(IR)
3. Navigate to sector	(Sonar)
4. Find and go to Queen	(Sonar)

5.	Self-Calibration	(of IR)
6.	Self-diagnostic	(Sonar, IR, Bumpers)
7.	Monitor and alert of Battery	(Battery-Checker Circuit)
8.	Follow behavior	(Flexible program)
9.	Chooses to ignore behaviors	(Flexible program)
10.	Self-identification	(Flexible program)
11.	Identify own kind	(Flexible program)
12.	Decision of priority of behaviors	(Flexible program)

Group	**Behaviors**	**Preferred Sensor**
1.	Communicate between robots	(through IR)
2.	Recruit others for same behavior	(through IR)
3.	Swarm	(Flexible program)
4.	Dissipate	(Flexible program)
5.	Follow the leader	(through IR)
6.	Communicates to queen with protocol	(through IR)
7.	Mark object or position with probe	(through probe)
8.	Sleep until queen advises to wake up	(Sonar)
9.	Simulate Flexible conveyor belt between different “machines”	(Flexible program)

Table 4: Behaviors at a glance

Experimental Layout and Results

We will be testing the MTJPRO11 boards using simple programs to test the servos. A simple board test will also be used to check the correct working condition of all of its elements. We need to perform these tests to assure the working order of the robots, since these boards were assembled in the lab. The next test will check for correct direction of the servo rotation to assure correct response from each servo. The memory test will check for correct space handling from the 32K bytes of RAM in the boards. Along with the servo test, we need to check if the new 3-inch wheels will fit and not interfere with any other component from inside the TJ.

Problems

Below is a table describing the problems we had while completing this project. These are found in Table 5.

- *12 Robots, 66 AA batteries*: Recharge as many as you can, use the fast chargers and unfortunately, buy more batteries.

- *Robots lining up for communication*: Because the robots use both bumpers and IR to detect and line up, the robots had a hard time correctly aligning because at close range, the IR saturates quickly. So when having ten of them the odds get better and many by chance communicated correctly.

Conclusion

The robot construction was a definite success. We built these robots using a modular approach. Each part of its total 7 sections was built identically. This helped us to later test them by simply interchanging them with robots loaded with a testing program. We design the program using the same approach. The software has many modules or subroutines. Each one can easily be tested individually. This format helped us later in the programming by providing an easy to understand and compose a sequence of behaviors. We met our original goal: to study group behavior of multi-robot communities. We also designed a platform that can be copied exactly and replicated infinite times and expected to perform the same way. We also accomplished other goals of building modular hardware identically enough that it can be exchanged between robots. Also to use software to help solve conflicts in differentiating IR from robots or from different sources. And last, to develop a modular software flexible enough that it can adapt to any of the robots. Also, a self-diagnostic program (which detects if any of the components are not working), add different types of robots (to see the behavior change in having different robots).

Documentation

The following is a list of our sources for information, specifications, and design:

Fred Martin, The 6.270 Robot Builder’s Guide, MIT Media Lab, Cambridge, MA, 1992.

Intelligent Machines Design Laboratory Web page:
http://www.mil.ufl.edu/
http://www.mil.ufl.edu/imdl

Mekatronix home page:
http://www.mekatronix.com/

Appendices

Figure 6: Picture of Robots during presentation.

Figure 7: Picture of TJs

Figure 8a: picture of servo

Figure 8b: picture of infrared

Figure 11: Top robot view