The Ants: A Community of Microrobots

by

James McLurkin

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1995, in partial fulfillment of the
requirements for the degree of
Bachelors of Science in Electrical Engineering

Abstract

As the field of robotics advances, new areas of research are emerging. Two of these new fields are microrobotics and robotic communities. The goal of my thesis is to explore both of these areas with an example borrowed from nature -- the ant colony.

Ants have evolved into one of the most successful species on earth. Two of the main reasons for this dominance are their small physical size and their community organization. Using real ants as a guide, the robot Ants have been designed with sensors and actuators analogous to their natural counterparts. Their software is written with cooperation in mind, aiming for community behaviors emerging from the interactions of many individuals. Their cubic-inch size produces a robot that is relatively inexpensive and practical to experiment with in a normal-size lab.

Natural ants use a multitude of different foraging techniques, many of which involve synergistic cooperation among several individuals. A synergistic interaction is one that produces a group that is greater than the sum of its parts. In this thesis, I have taken the first steps towards constructing a robotic community. Several autonomous microrobots were built and simple cooperative software was implemented.

Thesis Supervisor: Professor Rodney A. Brooks
Title: Associate Director, MIT Artificial Intelligence Lab

Table of Contents

Acknowledgments

Anita For being the bestest boss in the whole universe!!

Rod For letting me pursue my dreams in his lab, and even talking to me once in a while.

Ann Fraser and the rest of the Biologists at Harvard for providing me with not only assistance and ideas, but with real ants, some of which are even still alive!

Chapter I: Introduction

They are everywhere. You can remember watching them, as a child, fascinated by their constant activity, then deciding they were not that interesting and squashing a few of them. They are ants, small, six-legged arthropods with an amazingly complex society. They are one of the most successful species of animals on the planet. There are approximately 1,000,000 ants for each person on the earth, and if you added up the total weight of ants, it would equal the total weight of humans. Their amazing success can be attributed to two major reasons: their small size allows them to infiltrate almost any environmental niche and their community structure allows them dominate whatever area they enter. Engineers have been drawing inspiration from nature for ages,

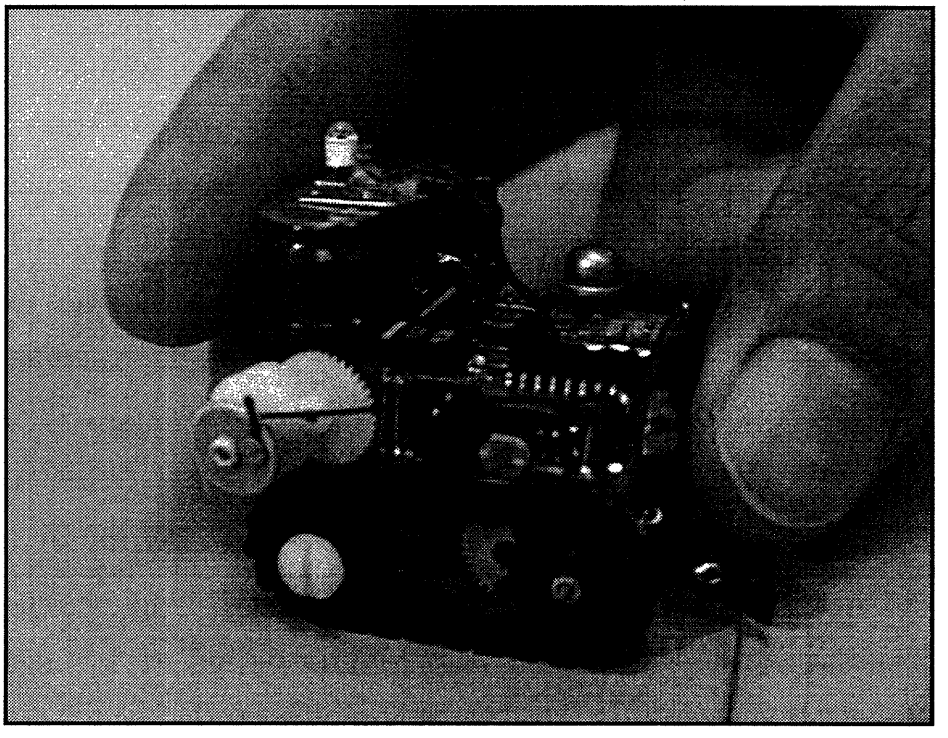


Figure 1: Weighing in at 1.3 oz, Anita is equipped with 2 bump sensors, 4 light sensors, 4 infrared communications receivers, 5 food sensors, a tilt sensor, a battery, a DC-DC converter, a serial port, three motors, and a gripper.

this thesis continues in that tradition; the goal -- to build a microrobotic community based on ants.

Microrobots

Microrobots enjoy many advantages due to their diminutive dimensions. The most obvious is their ability to operate in environments where other robots cannot fit because of their size. There are countless applications for small robots, from spying to inspecting nuclear power plants for coolant leaks in the pipes. The most fascinating potential application so far has been suggested by surgeons, who want to use a microrobot to perform endoscopic surgery. The future robotic surgeon would enter through the rectum and move around the large intestine, observing the colon and even removing polyps.

There are other advantages that come along with small size. Materials and parts get "stronger" as their dimensions get smaller. The mass of a robot is related to the cube of the linear dimensions. For example, a cubic-inch microbot is 1,728 times less massive than a cubic-foot robot. Since force = mass x acceleration, forces acting on the robot also tend to follow the cube root of the linear dimensions because of the mass term. However, the strength of most materials and structural components is usually dependent on the part's cross sectional area, which decreases as the square root of the linear dimensions. Therefore, the forces acting on these parts gets smaller much faster that the strength of the material, resulting in parts that are much stronger relative to the forces applied. The force that electric motors can exert does not fall as fast as the volume of the motor either, so it is possible to achieve insect-like strength with relatively modest power requirements.

Obstacle detection and data transmission ranges decrease as well. If the robots are to communicate with each other, 3 inches to a

microrobot would be equivalent to about 3 feet to a more conventional 1 foot tall robot. It is much easier to design communication and obstacle detection systems that have such small range requirements. The designer of normal size robots is usually struggling to increase the working range of the components. However, with microrobots, the range of transmitters sometimes has to be limited to provide adequate operation. Limiting the range of a device is much easier that extending it, and can produce more reliable transmission or detection.

The robots also become less expensive to build. The most sophisticated microrobots we have made cost about $300 each in parts. At that bargain-basement price it is economicly feasable to build a large number to do experiments in cooperative robotics. In addition, unlike normal sized robots, fifty microrobots will fit in a small lab, so the researcher does not have to worry about the football team practicing next to his or her experiment.

Robotic Communities

The study of collective behaviors and robotic communities are fairly new areas of research in the fields of artificial intelligence and robotics. As robots become more ubiquitous, cooperation among them will become more important in order to perform complex tasks efficiently. Because of their size, community is even more important for microrobots. For many potential applications, one microrobot would take a very long time to get the job done. On the other hand, a whole army of microrobots working together towards one goal would be a formidable work force. A community like this could incorporate synergistic cooperation, which is when the net worth of the group is more than the sum of the parts.

In addition to getting more work done, a community is more reliable than a single robot. If any one robot fails to accomplish their task, there are many others working alongside it. In order to design these future robotic communities, we need to understand how communities work. The main question: What is the underlying structure for a community of simple autonomous agents with no central control? Ants have evolved into what appears to be the best solution to this problem, so they represent the ideal natural model. Any problem is much easier to work through when you are given the solution at the beginning.

"Natural Design"

The design of the robotic Ants was inspired by nature, hence the moniker, "Natural Design". This is an idea with two parts, the first being a design methodology, while the second is to base our expections on observations from nature.

The "hardware" and "software" of natural ants are inseparable, they have evolved together for hundreds of millions of years. In order to build a robot like this, the hardware must be designed with the software in mind, and the software must be written with the hardware in mind. When the hardware is designed for one purpose, it can provide a substantial amount of sensory filtering. For example, the food sensors do nothing else but detect food. Therefore, the software does not need to perform extensive processing on the data from these sensors, either they detect food, or they do not. This is similar to how mant insect senses operate. There is a very select stimulus that excites a purpose-built sensor, eliminating the need for further processing. An example for the opposite case would be to use a camera to find food. The output of the camera has nothing to do with whether or not food is there, it reports the

intensity and luminance of incident light. The software then has to sort through all this data to figure out if there is food present or not.

The second half of the Natural Design idea is that nature is not perfect. When attempting to emulate a natural system, what you expect might not always be what a correctly functioning system will produce. The artificial standards that researchers judge their robots on must be tempered with the ultimate standard, nature. For example, the Ants have no way of ensuring that they drive in a straight line. Real ants do not walk in straight lines, they are constantly bumping in to objects and using information from their sensors to change their course. Communications are not perfect, sensations are not perfect, the environment is not perfect, the list goes on and on. In addition to not expecting the robots to perform perfectly, they are not even programmed to achieve such an unrealistic goal. When a real ant finds food, she then goes back to the nest to report her find to her nestmates. Her way back is far from the straight line that you might assume. She makes errors, gets lost, finds the trail again, turns around, etc. She will eventually get home, but over a very indirect course. Using that as an example, it would be silly to expect a robot to be able to travel directly home after finding food. If nature does not worry about problems like this, perhaps there are design solutions where the engineer need not worry about it either.

The Current State

Currently, we are able to build fairly sophisticated microrobots on the cubic-inch scale. The current design has a microprocessor, 17 sensors, and 3 motors. In order to construct these microrobots, we have developed a manufacturing process which enables us to build integrated machines at the printed circuit board level. This new construction

technique allows us to build a large number of small, simple, and inexpensive robots to explore new ideas in robotic communities. While we are able to build microrobots with some level of proficiency, we have only taken the first steps towards a solution to the larger problem of community. In doing so, we have unearthed more questions than we have answers.

Chapter II: Hardware

The hardware for the robot Ants was designed with their natural counterparts in mind, which is graphically depicted in Table 1. The treads allow the robots to move over many different types of terrain, while the mandibles can grip anything that is about the size and shape of a pea. The sensors give ant-like information to the microprocessor, while the IR emitters emit robotic "scents" for communication.

Main Hardware

Power Supply:

The robots operate from a 2.4 volt Nickel-Cadmium rechargeable battery. There is a MicroLinear ML4861-5 DC-DC converter that produces 5 volts from the 2.4 volt battery voltage. This 5 volt supply is

Table 1: The robot Ants were designed to emulate their natural counterparts as much as possible. This table shows how the functions of the robotic ants compare to the analogous functions of natural ants.

Function	Real Ant	Robot
Locomotion	Legs	Treads
Communication	Secretions, Touch	IR Receivers, IR Beacon, Tag LED
Finding Food	Chemical Senses, Sense Hairs, Eyes	Food Sensors, Bump Sensors
Navigation (Short Range)	Eyes, Antenna, Leg Senses	Bump Sensors, Trail finder
Navigation (Long Range)	Scent Trails, Sun, Eyes	Light Sensors, Compass, Trail finder
Attack	Mandibles, Stinger	Tag LED
Food Transport	Mandibles, Crop	Mandible, Mandible sensors
Taste	Antenna, Taste sensors	Mandible sensors, Food Sensors
Reproduction	Reproductive organs	Not modeled
Nest Building	Mandibles, Legs	Not modeled
Caring for young/queen	Everything	Not modeled

used for the microprocessor and the sensors. The driving motors and the mandible motor run directly off the battery voltage to minimize the load on the DC-DC converter. Careful attention was given to power consumption during the design to minimize the battery requirements. The entire robot prototype pulls 150 mA of current from the 2.4 volt battery when it is not moving and 230 mA when it is. The additional current goes to the two driving motors, which use 40 mA each during normal forward motion. This level of power consumption gives the robots approximately 20 minutes of running time between charges.

Microprocessor:

The Ants are controlled by a Motorola 68HC11E9 microcontroller. This chip has a CPU, 512 bytes of EEPROM (Electrically Erasable Programmable Read Only Memory), 256 bytes of RAM, and many input and output ports all incorporated into a very small package. 512 Bytes of EEPROM is not enough for complex software, so there is also an external XiCor 6875 memory chip that contains 8 kilobytes of EEPROM. This chip uses two of the ports of the 6811 to control it, but it provides two of its own, so there is no net loss of input/output capability. Both of these chips are about one square centimeter in size, which allows them to placed back to back on a very small daughterboard then soldered to the main processor board, as shown in Figure 2. The

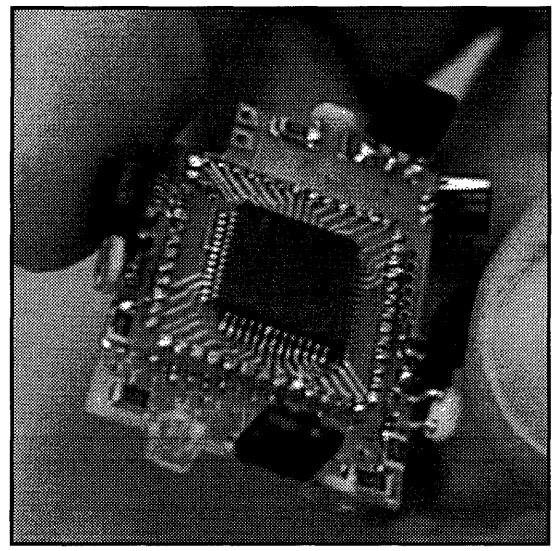


Figure 2: This 1" x 1.25" board contains the microprocessor, the memory, and eight sensors. It is also the backbone for the mechanical structure of the robot.

daughterboard was designed to be pin and footprint compatible with other 6811 processors, so it has the potential of being used in many other designs. The high level of integration of microprocessor, memory, and interface circuitry on these two chips allow a complete robot to be built with a minimal of external electronic components.

The 68HC11E9 is a member of Motorola's highly successful 6811 series of microcontrollers. It contains an 8-bit processor running at 2 MHz. Its instruction set is rich enough to make programming in assembly language a viable option, without having that stone-tool and animal-hide feeling that other processor's instruction sets can give developers. However, where this chip really shines is in its interface and interrupt hardware. There are 38 I/O pins, a serial port, 8 analog-to-digital converter channels, and no less than 21 separate interrupt vectors. All this support circuitry makes controlling hardware external to the chip much easier.

The XiCor 6875's 8K of EEPROM memory must contain all the software that the robots need to operate. The advantages of using this particular chip is that it interfaces directly to the microprocessor, can be reprogrammed without having to remove it from the robot, and replaces the ports it uses for communicating to the 6811. However, 8K is a very limited amount of programming space. Since the Ants have been designed to operate in a community, another way to add complexity to the system is to add more robots, not more code.

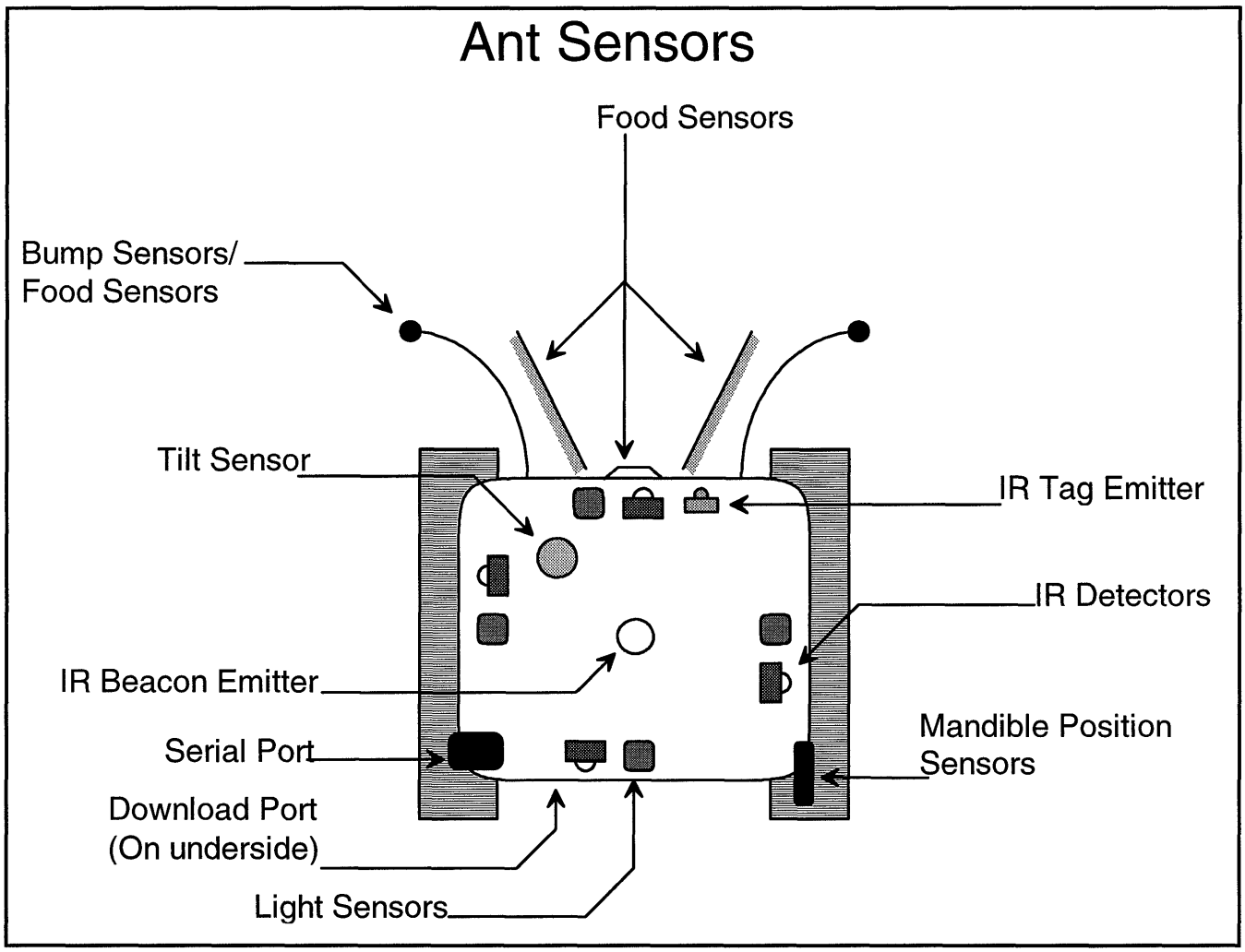


Figure 3: The robot's sensors give it information similar to a real ant, except with much less detail.

Sensors

The Ants have many sensors that provide the processor with information about the world around it. The diagram in Figure 3, along with the two pictures in Figure 5 and Figure 4, show all of the robot's sensors.

Bump Sensors	These let the robot detect obstacles that are in front of itself, walls, other Ants, food, etc.
Light Sensors	These sensors detect ambient light levels. There are four of them mounted in the front, back, and sides, so

the robot can tell which direction the light is strongest or weakest.

IR Beacon — This is an IR (Infrared) transmitter that transmits the "mood" the Ant is in twice a second. The mood is what the robot is doing at the time, looking for food, being angry, carrying food, etc. Other Ants in close proximity, about 6 inches, can receive this signal and know roughly where the transmitting Ant is and what mood she is in.

Tag Emitter — This emits a very low power, short range (about 1 inch) IR signal when the Ant bumps into something. If the object bumped into is another Ant, that Ant will receive the IR signal and know that it has been "tagged". Depending on the program, when an Ant is tagged it could be killed, get scared and run back home, or just be "It".

IR Receivers — These sensors detect the mood and tag signals. Each Ant has four of these mounted on the front, back and sides, so it can get an idea of which direction the signal is coming from.

Food Sensors — The food sensors use conductivity to detect the "food" used by the robots. Each Ant has five food sensors,

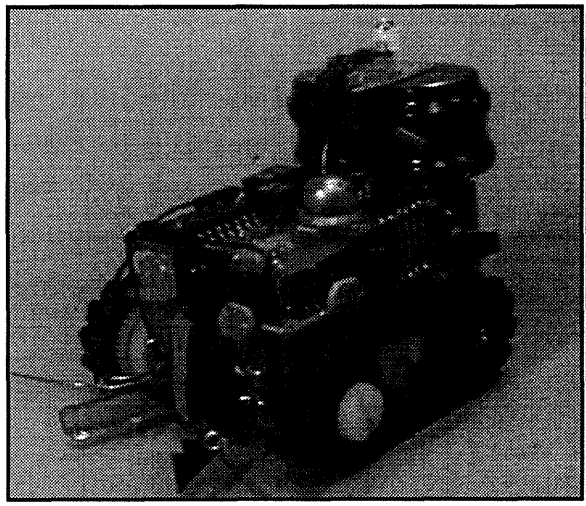


Figure 4: Front view showing mandibles, bump sensors, IR receivers, and light sensors.

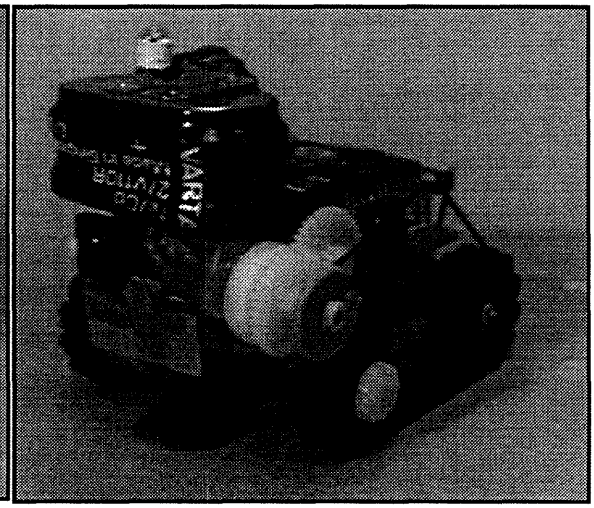


Figure 5: Back view showing mandible motor and IR beacon emitter.

one on each jaw of the mandible, one in the middle of the mandibles, and one built into the circuitry of each bump sensor.

Tilt Sensor — The tilt sensor is a mercury switch that will let the robot know when it is on a level surface and when it is not.

Mandible Position Sensors — These are simple contacts attached to the mandible motor pulley that indicate the position of the mandibles; open, closed, or indeterminate.

Recharge Connector — This consists of a small feeler on the bottom of the robot and the bump sensors on the front. It will let the Ant know when she has bumped into a recharge port so she can charge up her batteries. This will allow the Ants to operate for long periods of time without constant human intervention for battery changes or charges.

There are two sensors that are still on the drawing board, but deserve mention here:

Compass — This is a micro-compass that will let the Ant know what direction it is facing.

Trail marker — This is a small pen with disappearing ink mounted on the back of the robot. The Ant will be able to raise and lower the pen to leave ink trails. The ink vanishes when it evaporates, so the trails will disappear in time. This marker will allow for a more accurate simulation of real ants, but is too far from being implemented to think about in the behavior algorithms now.

This complement of sensors will give the robotic Ants access to similar information that a real ant has, but with much less detail. This information will allow the robot to perform many of the important functions of a real ant. Reproduction could be simulated by having robots act like pupae, then after a certain amount of time, emerge and function as members of the community. I have chosen not to model reproduction in my current system because I am more interested in the foraging and the external interactions between Ants. Adding reproduction would also add many more levels of complexity to an already complex system. (The first law of robotics: Keep it simple, stupid.)

While reading the detailed sensor descriptions below, refer to the pictures in Figure 3, Figure 4 and Figure 5. Technical readers may also wish to refer to the schematic in Appendix A.

Bump Sensors

There are two bump sensors on each Ant. They are similar in function to the whiskers on mammals. These are very simple sensors, made with just two wires, as shown in Figure 6. There is a long, flexible wire that goes through a small loop made out of stiffer wire, like a thread going through the eye of a needle. Undisturbed, the long wire, which is attached to an analog-to-

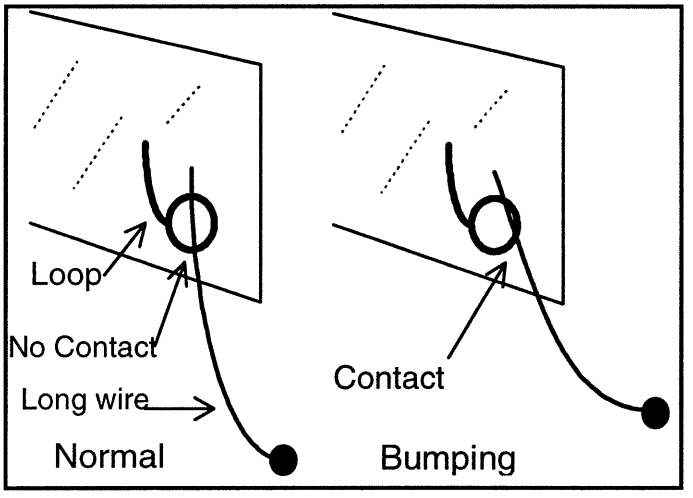


Figure 6: The long wire does not touch the side of the loop during normal operation. However, when the wire bumps something, it flexes and contacts the loop, which signals the processor.

digital input port of the 6811, does not touch the loop. The loop is connected to the 2.4 volt battery with a 100 Ω current-limiting resistor. If the robot runs into something, the long wire flexes and makes contact with one of the sides of the loop. Since the loop is connected to 2.4 volts, this same voltage appears at the analog-to-digital input port, so the processor can tell that there is a solid obstacle to the right or left of the front of the robot. These sensors are easy to build and are very robust, only requiring calibration after a major impact to the robot, such as dropping it on the floor!

Food Sensors

There are five food sensors, two on each "jaw" of the mandibles, one in the middle of the mandibles, and the other two serving double-duty as bump sensors. They all work by conductivity. The ground from the robot's battery is connected to the surface the Ants are running on by a wiper at the rear of the robot. The surface is covered with aluminum foil, and the food particles are balled-up pieces of brass foil. Therefore, the floor is electricly grounded, and since the food is resting on the floor, it is grounded as well. The two bump sensors are already hooked up to an analog-to-digital input port of the 6811, so all the processor has to do is check to see if they are grounded, which means that the sensor has touched a piece of food. The food sensors in the mandibles are pulled up to 5 volts by a 4.7 kΩ resistor and connected to three general-purpose digital input pins. They work the same way, when they touch a piece of food, they are shorted to ground and the processor can detect the presence of food.

The two food sensors in the left and right jaws of the mandibles can also detect whether or not the robot has a good grip on a food particle once she has attempted to pick it up. If the processor can pass a

current from the left mandible, through the food, to the right mandible, that is a sign of a good grip. If the current does not flow, or the food is still touching the ground, the software will re-open the mandibles and try to pick the food up again.

Light Sensors

Each Ant has four light sensing photoresistors, one mounted on each side of the robot. The circuitry that interfaces the sensor to the processor is very simple. The photoresistor is in series with a 6.8 kΩ resistor, which creates a voltage divider between the 5 volt supply and ground. As the resistance of the photoresistor changes, the voltage at the node between the photoresistor and the 6.8 kΩ resistor changes too. This node is connected to one of the analog-to-digital (A/D) converter ports of the 6811. Accuracy is limited to 5-6 bits because of power supply fluctuations, but that is enough to get rough light measurements. Even if the power supply voltage could be held to tighter tolerances, the sensors manufacturing tolerances are not strict enough to warrant precise measurements, and the software does not require such precision anyway. Because manufacturing tolerances on these parts was very bad, four photoresistors had to be matched to each other before they were soldered down to the robot. This matching ensured that all four sides would respond to light similarly.

Tilt Sensor

There is one tilt sensor on each Ant. It is simply a small mercury switch that closes a contact when the robot is level. When the switch is open, a 4.7 kΩ pull-up resistor raises the voltage at the input to the 6811 to 5 volts. The switch is checked 50 times a second and the data is time averaged to account for vibrations when the robot is moving.

Battery Voltage Sensor

The positive terminal of the 2.4 volt battery is connected to one of the A/D ports of the 6811. This allows the processor to monitor the battery voltage, which gives an indication to the amount of battery power remaining. The software can then adjust the robot's activities accordingly.

IR Receivers

This is the most complex group of sensors on the Ants, which is understandable because it is responsible for the communications between the robots. In order to make the explanation simpler, the following text describes the process behind just one receiver, since all four of them operate in the same manner. The IR receiver has a small circuit built-in to detect a 38 kHz pulse burst that lasts for 600 μs. The circuit pulls its output low for about 800 μs when it detects such a burst. The processor is interrupted by an internal timer every 800 μs, regardless of what the receiver is doing. During this interrupt (the IR

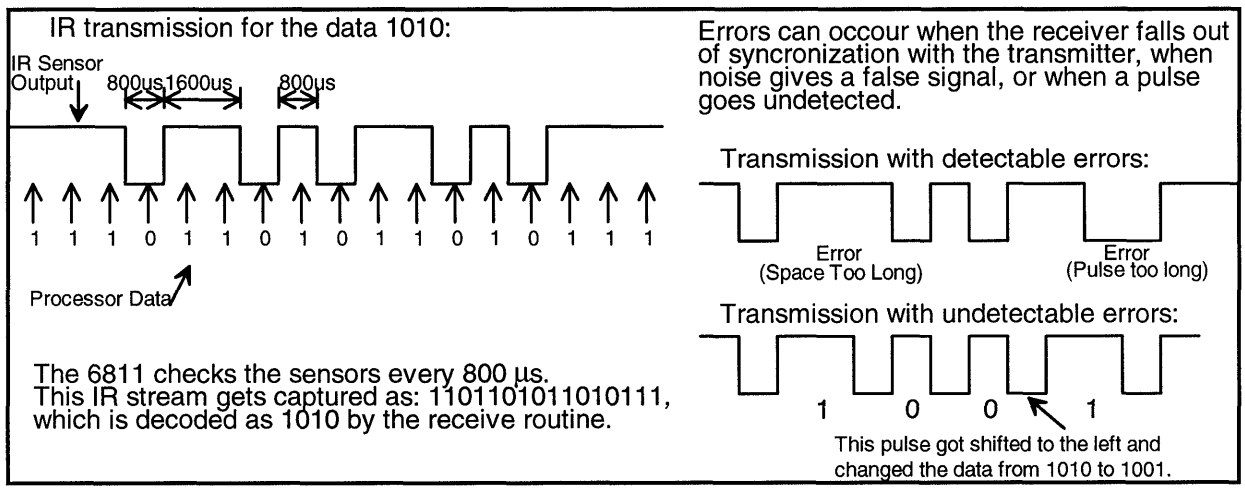


Figure 7: The IR transmission protocol can transmit a 4-bit message over a range of six inches. Each Ant has receivers mounted on all four sides, which will enable the receiving robot to determine the direction of the transmitting robot. This limited bandwidth is adequate, as the robots will not have much to say. The communications will consist only of their present mood: hungry, tired, angry, etc.

interrupt) the 6811 reads the receiver output. This bit of data is shifted into a 16-bit IR register, so the software has a record of the last 16 bits received by each sensor. When the last four bits of the 16-bit data are "0111", that means that the IR detector received a pulse 2400 μs ago and nothing since. This could signal the end of a nibble of transmitted data. When this happens, the program attempts to convert the stored 16-bit data into a 4-bit nibble. This process is graphically illustrated in Figure 7.

The data format calls for one start pulse, then four more pulses separated by spaces. An 800 μs space signifies a zero, while a 1600 μs space is a one. A detectable error occurs if there are any spaces longer that 1600 μs or any two pulses right next to each other. The conversion process will stop if the routine finds any invalid data, which provides some error detection. If the conversion is successful, the 4-bit data is stored in one of four receive data registers and the receive data timer for that register is started. Having four registers per side allows the robot to store received data for four different robots. Each of the four registers had a data timer associated with it. When the timer times out, the data register is cleared. This gives the higher-level software up to date data all the time, but does not require action as soon as the data is decoded, which makes the programmer's life easier. The received data represents the moods of surrounding Ants.

Under ideal conditions, each Ant should be able to receive and decode data from four robots on each side, for a total of sixteen robots around it simultaneously. In the real world, there are many sources for possible errors; IR noise in the environment, power supply glitches, robots moving out of range, or framing errors, just to name a few. When there are many Ants in close proximity to each other, there is also the problem of more than one transmitting at the same time. The

transmitters have a range of only about 6 inches, which was chosen to limit the amount of extra IR noise in the environment. With all this going on, the robots manage to receive and decode correct data about 50% of the time, which sounds much worse than it actually is. Each robot transmits their mood twice a second, so even if the robots miss half of the transmissions, they still get a mood from the surrounding robots about once a second, which is enough for the software work with. Future software will incorporate more sophisticated timing and decoding routines, which should improve communications dramatically.

Mandible Position Sensors

There are two mandible position sensors. One responds when the mandibles are open, the other when they are closed. If neither sensor is active, the mandibles are in an undefined state somewhere in between. Both sensors are simple contact switches that get grounded by a contact on the mandible motor pulley. By bending the wire that makes the contact, the position of the mandibles when they are open and closed can be adjusted.

Motors

Driving Motors

There are two driving motors on an Ant, one for each side of the robot. The processor controls the motors with Motorola 1710 motor controller chips and uses a pulse width modulation (PWM) control routine to produce four different speeds: real slow, slow, fast and turbo.

Each motor drives a small gearbox, which turns the wheel. Figure 8 shows a picture of an Ant gearbox. The components of the drive system are an excellent example of one of the biggest problems associated with microrobotics: finding parts. The motors are vibration motors from the silent alarms of broken beepers. The gears come from Futaba servo motors, the wheels and treads come from Stock Drive Products, the axles come from Small Parts Inc. and the plastic retainers are made by hand. In order to build microrobots, one has to find sources of all these small parts, which can be literally anywhere: Motorola catalogs, fabric outlets, or even toy stores! (My personal favorite.)

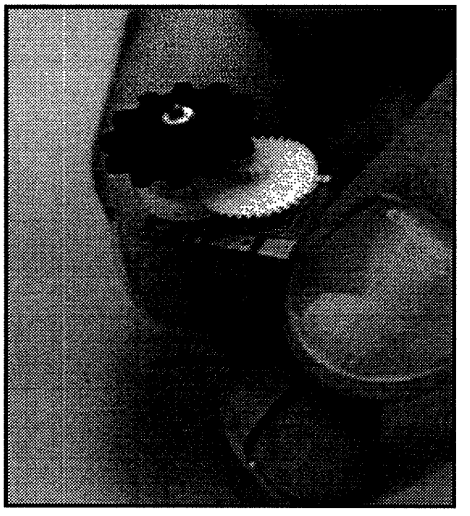
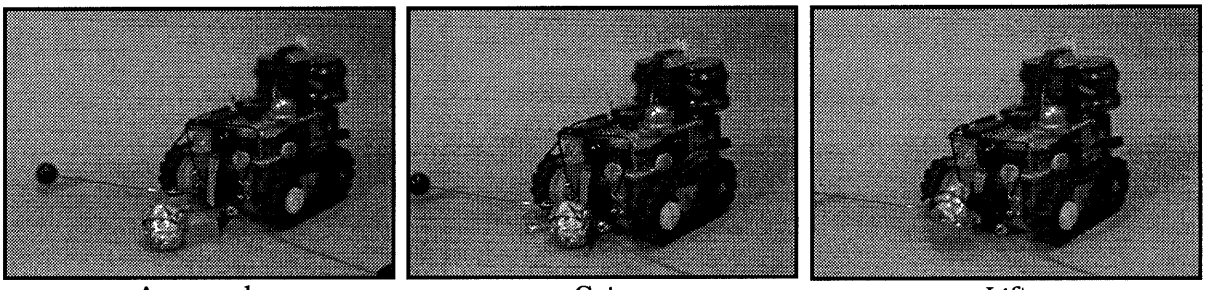


Figure 8: The gearboxes are made by soldering small axles into printed circuit boards. Each Ant has three gearboxes, two for driving motors, and one to actuate the mandibles.

The gear box construction is where using the printed circuit boards for structural elements really pays off. The gears need to be within 0.002" of their correct spacing in order to mesh properly. The tolerances on the circuit boards are within 0.001", so all the builder has to do to ensure proper mesh is to make sure the axles are mounted perpendicularly to the PC board.

The motors produce a considerable amount of power for their size. The stall torque with a 59:1 ratio gearbox is about 1.75 oz-in. Since the wheels have a 1/4 inch radius, the robot produces almost ten ounces of pulling power (2 motors x 0.4 oz-in / 0.5 in), which is enough to propel it over the roughest of terrain, even carpet. If the treads could get a grip on a wall, the robots could crawl right up it, as they only weigh 1.3 oz. The



Approach Grip Lift

Figure 9: The mandibles in action.

gear ratio does not sacrifice forward speed, either. The robot cruises along at 0.5 foot/sec. If it was human size, that would be 25 mph!

Mandible Motor

The mandibles allow the robots to manipulate their environment, allowing them to pick up and carry objects the size of a pea. Figure 9 illustrates the series of motions required for the robot to pick up some "Ant Food", which is really just a balled-up piece of brass foil wrapped around a small ball bearing. The ability to modify the environment opens up many interesting research possibilities.

There are three main parts of the mandible: the flexible frame, the motor, and the string that connects the two. The motor, gearbox, and motor controller are the same as in the driving motors, except the components are mounted at the rear of the robot underneath the battery. The connecting string is a thin piece of thread from a fabric store downtown. (The parts hunt continues.)

The most important part of the mandible is the frame, a flexible piece of printed circuit board material. The patterns for the mandible sensors are etched onto the material. The flexible PCB material is then cut and folded to produce two jaws and a central trunk, which can be seen in Figure 10. The thread attaches at the base of the jaws and is wrapped around a pulley attached to the mandible motor. When the pulley turns, the thread pulls on the jaws, causing them to bend towards each other. When they can no longer move, either from contacting another object or each other, the main trunk flexes up, lifting whatever the mandibles are holding. The spring constants of the jaws and trunk are determined by the shape of the material cut out of the flexible printed circuit material, and are chosen to ensure the mandibles grip before lifting. The end result is a grab-and-lift motion from only one motor, which makes the system smaller, simpler to design, and easier to build. The motor has approximately five ounces of pulling power, which is more than enough to lift anything the mandibles can grip.

The food is designed to prevent it from bouncing away when the robot contacts it. It is made out of balled-up 0.001" thick brass foil with a small ball bearing placed off-center inside the foil ball. Brass was

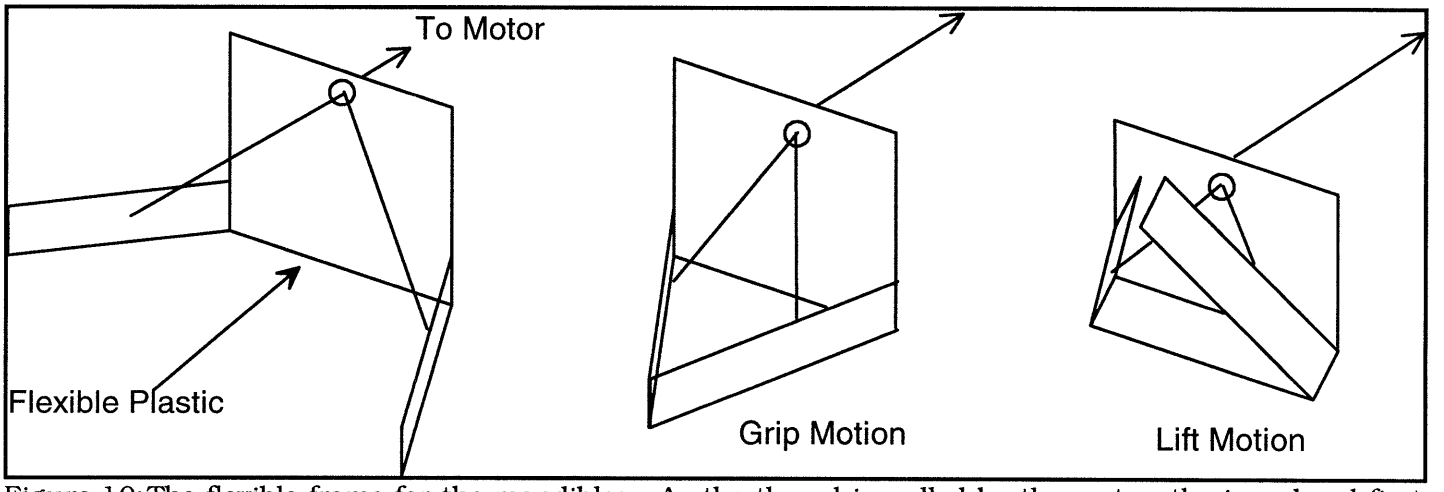


Figure 10: The flexible frame for the mandibles. As the thread is pulled by the motor, the jaws bend first, gripping the object. The stiffer trunk flexes when the jaws are fully closed and lifts the object. This design produces a grab-and-lift motion with just one actuator.

chosen because of it's point conductivity characteristics. The points of contact between the playing surface and the foil and the food sensor and the foil are very small, and brass foil worked better that aluminum foil and was lighter that a solid steel ball. The off-center ball bearing inside the food gives it a "weebelo effect"; the food always wants to orient itself so that the ball bearing is as close to the ground as possible. This makes it harder to roll, and less likely to escape the jaws of an approaching Ant.

Miscellaneous Hardware

Mood Lights

There are three LEDs mounted on the top of the robot: red, yellow, and green. Each LED has three modes: on, off, and flashing. The software changes the state of these indicators depending on what the robot is doing at the time. This enables the researcher to tell at a glance what mood the Ant is in and what she should be doing. The present light pattern is stored in memory and the lights are updated twice a second by the operating system.

Serial Port

Each Ant has two serial ports. There is a small serial connector that is located at the back left side of the robot. This lets the robot run under remote control or transmit data to a computer for debugging. There is also a serial connector on the bottom of the robot, in the form of four plates. These plates are connected to the pins the processor uses for programming the 8K of EEPROM. When the robot is placed on a matching programming base, the plates contact springs on the programming base and the robot can be reprogrammed. This allows for

the quick programming of many robots without having to plug and unplug connectors.

Future Sensors

Compass

The compass will allow the Ants to know which direction they are heading. There are many small hall-effect sensors available on the market, so designing a micro compass is a feasible possibility. There is even a watch that has a digital compass built in, so all the intrepid parts-hunter has to do is buy several of these watches to get a complete set of hall-effect sensors and the associated decoding circuitry, all in a pre-fabricated, miniature package!

Since the Ants will be operating in a controlled environment, it would also be possible to cheat and create an artificial magnetic field many times stronger than the earth's magnetic field. This way, we would not have to worry about detecting the small magnetic field of the earth, and it would save us from the magnetic interference that is present in most labs. Making a strong field that covers a large area is a daunting task, so instead, we would make a grid of magnetic fields. That way the robots would not only get intermittent heading information, they would also have an idea of where they were in the playing field by counting how many grid lines they crossed. This pseudo-compass would be composed of two hall-effect sensors connected to analog-to-digital channels of the 6811. Hall-effect sensors detect the strength of magnetic fields perpendicular to the device, so with one, the robot will be able to detect the sine of its angle relative to the direction of the magnetic field. With another sensor mounted perpendicularly to the first, the sine and cosine

can be determined. With this information, the processor can compute the heading of the robot relative to the applied magnetic field.

Trail Marker

The trail marker is the last major sense that occurs in natural ants that is not emulated in the robots. Real ants are able to leave scent trails to lead their nestmates to food sources and other places of interest. When a forager finds a large food source, she drags the tip of her abdomen on the ground on the way back to the nest, excreting a chemical that other ants can smell. When she gets back to the nest, other workers can follow the trail back to the food. This ability to lay trails and recruit other workers makes ants more effective at efficiently exploiting the resources they need to survive.

This sensor is still in the pre-design phase, but the main idea being considered involves using disappearing ink to mark the trail and line sensors mounted on the underside of the robot to find the trail. Disappearing ink is visible when wet, but disappears as it dries. The marker would be actuated with the mandible motor, so when the mandible was pulled up further than normal, the marker would contact the ground. This would leave a trail of ink wherever the Ant went. If the ground is white and the ink is some dark color, the sensor required to have other Ants find the trail is an optical emitter-reflector mounted on the underside of the robot. A strong reflection indicates that the robot is over the white background. When the reflection is weak, the reflector is directly over a dark trail. With two sensors, one at the front left and one at the front right, the robot can align the trail in between the two, so that both are getting strong reflections. When the robot deviates from the course of the trail, one of the sensors will get blocked, and the robot can change her course to align with the trail again. This scheme also has the

advantage of leaving trails would also be visible to people. However, designing a small, articulated pen will be a serious engineering challenge.

Chapter III: Construction

3-D Circuit Boards

The Ants are built using an innovative three-dimensional printed circuit board construction technique. Miniature circuit boards are designed so that they can be soldered together to make structural as well as electrical connections, as shown in Figure 12. Each Ant requires the twelve circuit boards shown in Figure 11, populated with over 150 components. The boards are designed using Douglas CAD on a Macintosh, which allows incremental design changes to be made easily to existing layouts. When the layout is completed, there are two ways to produce the boards, machining them in the lab or sending them out to be manufactured.

For the prototypes, the boards were made in the lab with a T-Tech computer-controlled PC board milling machine. With careful operation and an abundance of patience, the T-Tech machine is capable of producing small, precise boards quickly. The machine works by milling

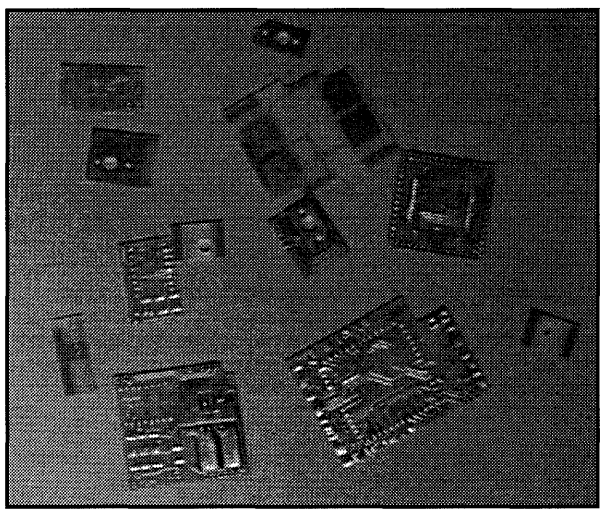


Figure 11: Each Ant is constructed from 12 miniature circuit boards, which make the electrical connections as well as the mechanical structure.

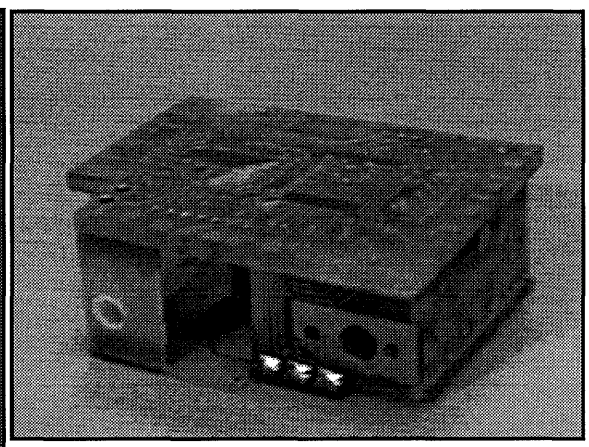


Figure 12: A 3-D circuit board. The large board is the main CPU board. The gearbox boards are the two vertical boards with three large holes drilled through. This integrated manufacturing technique makes the Ants possible.

away material from the surface of copper-clad printed circuit board (PC board) in the negative of the desired pattern, "isolating" traces. Two passes are needed, one rough pass with 0.020" isolations to protect against shorts and a fine 0.010" pass to isolate the areas in fine detail. Spacing tolerances can be held to within 0.001" and 5 mil traces are possible, although we use 10 mil traces and spaces to facilitate ease of manufacture. The machine also drills holes and cuts the board out, producing a complete PC board in the lab in a small fraction of the time and cost required to get the job done commercially. The boards for the last prototype were made in about six hours.

Once the design is finalized, the use of this PC board construction technique presents a unique opportunity for easy production, since the exact same design can be sent to a commercial PC board fabrication company. For the current crop of Ants, the cost of boards was approximately $600, which was enough for 50 robots. This works out to a price per robot of $12, which is by far the cheapest and easiest way to built a large number of small robots.

The Parts Hunt Continues...

Once most of the bugs were worked out of the design, the next six months were spent looking for, ordering, confirming, waiting for, re-ordering, re-confirming, and re-waiting for parts. Most of the time this process goes without many glitches, but sometimes you get truly extraordinary events. For example, the 8k memory chips were ordered in September, but it took the distributor six months to deliver on devices with a 6 week lead time! The pinion gears for the motors are an even an more impressive epic of parts-hunting. Attempts to order them from the manufacturer or find them elsewhere proved miserably unsuccessful, so

an ex-visiting scientist bought $1100 worth of them in Japan and mailed them to us in November. We're still waiting for that package...

Although that package never came, other parts were flowing in from all over the country. Everything but the pinions. After many more calls, we found an executive with enough clout to fax Japan directly and get the parts for us. The price? 25¢ per pinion, so all those months of work and wait amounted to $17.50! After all the ordering and waiting, the grand total came to just over $6000, which is enough parts for almost 30 Ants, but only twenty will be built, giving a cost per ant of about $300.

Ant Day!

The ultimate plan called for 21 robots, which would make the Ant Farm the largest robotic community ever. One person alone cannot build 21 robots, so that spurred the invention of a new holiday, Ant Day. For a free lunch, students were coaxed into helping to build the robots. The first shift started at 8:00 am and went until 12. Then the second shift started, which was planned to end at 5, but went on until 10. That still did not get all the robots built, so we did the same thing again the next weekend. At the end of the second Ant Day, close to 30 people had put in over 200 hours of work. An amazing amount of progress was made, and the only tasks remaining were final assembly and testing.

There were two important lessons learned during these construction marathons. The first verified Flynn's Law of time management. It states: "Anything will take three times as long as you think it will." In empirical form;

$$T_{actual} = 3 \cdot T_{planned}$$

This holds true even if you multiply your original estimates by three before you tell anybody, it still takes three times as long. Initially,

there was only going to be one Ant Day, but in the end it looked like three would have been the correct number, which is exactly what Flynn's law predicted.

The other important lessons were in production management and task scheduling. Most of the production went fine, but on two occasions where the critical path and bottleneck was not spotted in time. This really threw a wrench in the works, but luckily both of the times it happened it was already late in the evening and close to quitting time. These are both very valuable skills and crucial to the successful management of any project.

After all the dust had settled, there were six new robots, Anita, Sandra, Kisha, Tracie, Niqi, and Hope. The robots are named after women, since all worker ants are female. Six is a little shy of the overly-optimistic 21 robot goal, but it is enough to take the first steps towards building a robotic community.

Chapter IV: Software

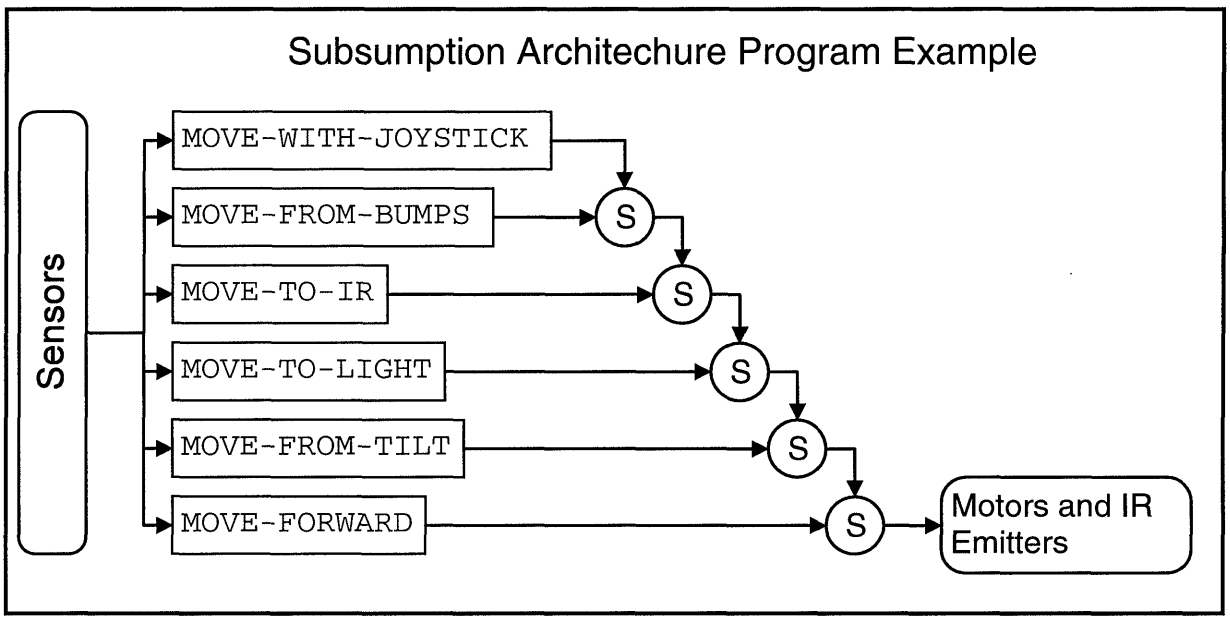


Figure 13: The Ants are programmed with a variant of Subsumption Architecture. The outputs from behaviors higher on the diagram can override, or subsume, the outputs from the lower behaviors.

Subsumption Architecture

The Ants are programmed with a subset of Brooks' Subsumption Architecture implemented in 6811 assembly language. [1] A program for the Ants would consist of a group of behaviors, arranged in a hierarchy as shown in Figure 13. A behavior is a small piece of code that acts like a finite state machine. The behavior monitors the sensors, and outputs a signal whenever a particular input condition is met. For example, the `MOVE-FROM-BUMPS` behavior outputs motor commands when the bump sensors detect an object in the robots path. The commands from this behavior are designed to move the robot away from obstacles. Outputs from behaviors higher on the hierarchy override, or subsume, outputs from less important behaviors. This is an effective method of providing

the robot with a response for every possible sensory input, without having to explicitly program for every condition. Summing the responses of many behaviors is a much easier task that looking at all of the sensory inputs and then trying to decide what to do. As a result, the robots can to exhibit surprisingly complex actions with a very small amount of software.

Behaviors have access to timers and shared memory registers. For example, in the case of MOVE-FROM-BUMPS the output might be: "move backwards-right for 1.5 seconds." Shared memory registers let the behaviors communicate with each other. All the behaviors are cooperatively multitasked, so from the programmer's perspective, they can be thought of as running simultaneously. The behaviors from the example in Figure 13 are basic and are used extensively in many more complex programs, so they warrant further discussion.

MOVE-FORWARD

This behavior simply makes the Ant move forward.

MOVE-FROM-TILT

When the tilt sensor is activated, this routine backs the robot up until it is on level ground again. Then it turns 180° and moves forward a little bit.

MOVE-TO-PHOTOS

This routine compares all the light sensors to their average value and moves the Ant towards the one with the greatest difference, which is the direction with the most light.

MOVE-TO-IR

This routine looks to see if any of the IR data registers contain data. If so, the robot heads in that direction.

MOVE-FROM-BUMPS

This moves the robot away from an activated bump sensor. If the robot has been bumping into several things in the past few seconds, this routine slows the motors down and reads the sensors continuously to make navigation out of tight spaces possible.

MOVE-WITH-JOYSTICK

This behavior monitors the serial port to see if there is a joystick plugged in. This allows the operator to move the robot around and pick up things with the mandibles, which is lots of fun for kids of all ages!

This implementation of subsumption on the Ants is not as flexible as the original, as it only allows for the behaviors to be arranged in a simple hierarchy. To augment the flexibility of the software, each program can have many different hierarchies, or moods. The moods can be changed from any behavior, but the software is more structured if there are special mood changing behaviors that do nothing else but monitor the sensors and the state of the robot and select the appropriate mood. This extra degree of programming freedom allows for fairly complex software to be encoded onto the robots, while still keeping the use of memory to a minimum. The most complex software to date contained nine moods and over 30 behaviors, yet only occupied about 1K of memory space.

Operating System

All these behaviors run on top of the Ant operating system. The operating system consists of four main parts, the subsumption loop, the low-level subroutines, the timer interrupt handler, and the IR interrupt handler. The subsumption loop cooperatively multitasks the behaviors,

which allows the programmer to model them as all processing information simultaneously. Although cooperative multitasking is easy to implement, the operating system is dependent on well-behaved behaviors in order to switch tasks and prevent crashes. The low-level routines provide support for the software, including the interface with the sensors and actuators. The timer interrupt occurs 50 times a second, and runs the PWM routine that controls the driving motors. Every 25 timer interrupts, additional code, called the `slow-timer-routine` is run. This code updates the mood lights and decrements the counters that the operating system uses to time various functions. Every 800 μs the processor checks the IR receivers. This piece of code is called the IR interrupt handler and was discussed in detail in the section on the IR receivers on page 16. The operating system has been heavily optimized for space and currently occupies less than 2K of memory.

Example Antware

The program given in Figure 13 is very simple, and does not demonstrate of all the features of Antware. Tag is a better example, and illustrates the basic process of getting the Ants to do something useful, or at least entertaining. The code fragment below is the section where the behavior hierarchy and moods are defined. The full printout can be found in Appendix B.

```
startup-mood
not-it-mood
  (!8 no-signal)
  (!16 move-forward-slowly)   (!8 flashing-green-light)    (!8 not-it-signal)
  (!16 move-from-it)          (!8 dont-subsume)            (!8 dont-subsume)
  (!16 move-from-tagged-ant)  (!8 dont-subsume)            (!8 dont-subsume)
  (!16 move-from-bumps)       (!8 dont-subsume)            (!8 dont-subsume)
  (!16 did-I-get-tagged?)     (!8 all-lights)              (!8 got-tagged-signal)
  (!16 am-I-tilted?)          (!8 dont-subsume)            (!8 dont-subsume)
  (!16 end)

it-mood
  (!8 it-tag-signal)
  (!16 move-forward-turbo)    (!8 flashing-red-light)      (!8 it-signal)
  (!16 move-to-not-it)        (!8 dont-subsume)            (!8 dont-subsume)
  (!16 did-I-tag-somebody?)   (!8 dont-subsume)            (!8 dont-subsume)
  (!16 move-from-bumps)       (!8 dont-subsume)            (!8 dont-subsume)
  (!16 xmit-tag-if-bumping)   (!8 red-light)               (!8 dont-subsume)
  (!16 am-I-tilted?)          (!8 dont-subsume)            (!8 dont-subsume)
  (!16 end)
```

Tag has two moods which are labeled **it-mood** and **not-it-mood**, which are in bold for readability. The label **startup-mood** tells the operating system which mood to use when the robot is first turned on. The first byte following the mood label is the tag signal. This tells the operating system what IR signal to transmit from the tag emitter when the Ant bumps into something. The **not-it-mood** transmits *no-signal* when the robot bumps an object, while the **it-mood** transmits the *it-tag-signal*. All IR signals are printed here in italics, also for readability.

The list following the tag signal is the structure of the behavior hierarchy. Each entry has three items, the behavior name, the mood light color, and the IR beacon signal. If a particular behavior is active, it can override the mood light color and IR beacon signal of the lower behaviors, or leave them intact with the item `dont-subsume`. For example, the `move-forward-slowly` behavior flashes the green light. But if the robot gets tagged, the `did-I-get-tagged?` behavior overrides the lower behavior's outputs and turns on all the lights. The IR beacon signal works in the same fashion.

Unlike the diagram in Figure 13 on page 30, the behaviors in the real software are arranged with the least important at the top. So for the **not-it-mood**, the robot will `move-forward-slowly`, until an output from one of the more important behaviors subsumes that output. For example, if the robot detects a signal from the Ant that is "It", the `move-from-it` behavior directs the robot to a safer area of the game field. The other behaviors work in the similar manner, doing more or less what their names imply.

There are two mood-altering behaviors, `am-I-tilted?`, and `did-I-get-tagged?`. The first of these, `am-I-tilted?`, monitors the tilt sensor. If a tilt condition is detected, the robot switches moods. This allows the researcher to decide which robot will be "It" and to correct for a unsuccessful tag, which could result in two "It" robots. The second, `did-I-get-tagged?`, monitors the IR receivers for the *tag-signal.* If that is received, then this robot just got tagged by the Ant that is "It". The behavior then becomes active and stuns the robot for several seconds so the tagger can make her getaway.

Although simple, this software environment allows for systems with many inter-robot interactions to be programmed easily and efficiently. Whether or not it will stand up to the challenge of programming a full-blown community remains to be seen.

Chapter V. Community

Simple Communication

The single most important attribute of any community is the ability of its members to communicate with each other. Communication does not need to be as complex as human language. Ants have survived for 150 million years using very simple communication, mostly by scent and touch. In order for the robot Ants to form any type of community, their ability to communicate in real-world situations needed to be tested and debugged with simple multiple robot experiments.

The first of these tests was a game of Follow the Leader. One Ant is the leader, and it transmits *leader-signal* from its IR beacon. Another Ant is the first follower, and it looks for the *leader-signal* on its IR receivers. When the follower receives that signal, she heads in the direction of the leader and transmits the *first-follower-signal* from her IR beacon. A third Ant is the second follower. The software for this robot is the same as for the first follower, except it looks for the *first-follower-signal* and transmits *second-follow-signal.* Ideally, this software would produce a line of robots moving around their environment. In reality, you get a bunch of robots bumping into each other and becoming confused when errors occur in the IR transmissions. This may seem like a failure, but the second half of the Natural Design idea is that nature is not perfect. In this case, real ants do not follow scents perfectly and they bump into each other all the time. Even if these robots could play a perfect game of Follow the Leader, that would not be a predictor on whether or not they would be good ants.

Follow the Leader exercises the IR beacon transmitter and the IR receivers, but the tag emitter is not used. The best way to test the tag

emitter is to program the robots to play a game of Tag. One Ant would be "It" and the others would be "Not it". The "It" Ant transmits *it-signal* from its IR beacon emitter, while the "Not it" Ants transmit *not-it-signal*. Whenever the Ant that is "It" detects a *not-it-signal*, it heads in that direction. When the "It" robot bumps into anything, it will transmit *tag-signal* through the tag emitter. If the obstacle is another robot, the robot will receive the *tag-signal* in one of its IR receivers, and change its mood to "It", which causes the IR beacon to transmit *it-signal*. When the tagging robot receives *it-signal* in her IR receivers, she will change her mood to "Not it" and run away. The initial tests with this software illuminated the need for more elaborate IR signal decoding routines. With the updated software, the robots were able to tag each other about 50% of the times they bumped into each other. This poor performance can probably be attributed to the type of emitter that was chosen to be the tag emitter. The current tag emitter transmits a narrow beam, which can miss the IR receivers at close range. A wide-beam emitter would fix that problem. Once a robot is tagged, they do a pretty good job of transferring the "itness" from one to the other, which is successful about 75% of the time. Future IR decoding software should increase this percentage.

Future Ant Games

Once the robots are finding and tagging each other with reasonable proficiency, then the next step is for them to interact with each other to reach a common goal, a process commonly known as cooperation. When Tag is extended to include cooperation it becomes a new game, Manhunt. In Manhunt, the robots are divided up into two teams. The goal is to tag all the members of the opposing team. Each time a robot gets tagged, she becomes a member of her tagger's team. Although this software is

not yet implemented, it would allow different strategies and levels of inter-team cooperation to be tested against each other to see which works better. Manhunt could also be extended and used to explore ideas about predator-prey relations, an area with a developed theoretical background.

Taking Manhunt one step further leads you to Capture the Flag, which is also on the software drawing board. The goal in this game is to capture the opposing team's "Flag", which is a piece of ant food. The rules about tagging could be the same as in Manhunt, with tagged Ants switching sides. Traditionally, the rules of Capture the Flag call for a tagged player to be sent to "Jail", but they can be released at any time by being tagged by a member of their own team. Another option is to have the robots become stunned for a short period of time. When the flag is found, the attacking Ants need to get the flag back to their base, while the defending Ants need to try and stop them. This would be a very sophisticated game, with many variations to explore and even more bugs to be worked out!

The Beginnings of an Ant Colony

The ultimate goal is to enable the robots to act like real ants. Of the many things ants do, foraging is arguably the best starting point for a robotic community. First of all, most natural ants forage above ground, so their activities can be readily monitored and recorded. Second, there are many uses for robots that can forage as a community. Right now, the Department of Defense is very interested in small robots that can collect unexploded cluster bomblets. A group of robots that can forage together like Ants might be the way to accomplish this.

The main feature of ant foraging that makes it so successful is that they help each other all the time. When a worker ant finds a new food

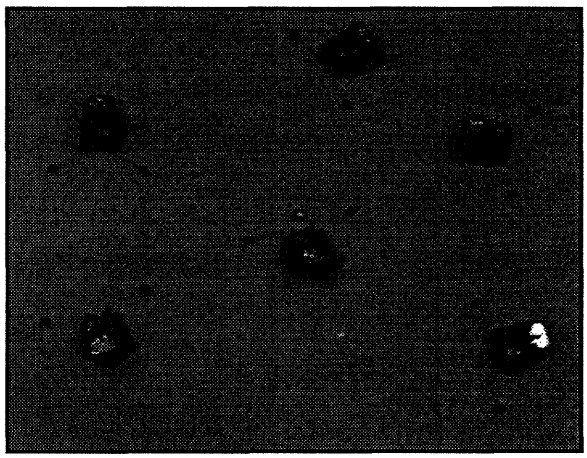


Figure 14: When the Ant in the center touches the food, she will signal to her sisters...

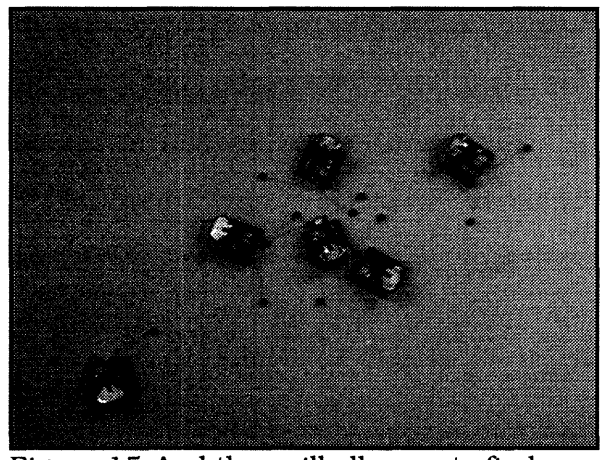


Figure 15: And they will all come to find some food too. Cooperation in action!

source that she cannot carry back to the nest, she will lay a scent trail from the food to the nest, so her nestmates can locate the resource quickly and efficiently.

In order to experiment with foraging software with the robot Ants, a larger ant farm had to be constructed. Previous experiments were conducted in a small 3' x 3' playing area. However, to have six robots moving as fast as the Ants move and not bumping into each other and the walls constantly, a larger 4' x 8' arena was constructed. Natural ants forage over amazingly large areas compared to their body size. An analogous area for cubic-inch robots would take up most of the 9th floor of the Artificial Intelligence Lab!

Cooperation among robot ants is accomplished using the IR transmitters. When a robot finds food, she stops and transmits *I-found-food* through her IR beacon emitter. Other robots in the vicinity that detect this signal head towards her, transmitting *I-see-a-robot-with-food* from their beacon emitters. Any robot that detects this secondary signal heads towards it until they receive the primary signal, then they head towards the first robot. In this manner, many robots can be vectored towards a large food source quickly, as can be seen in Figure 14 and Figure 15.

Ants in nature have evolved a bewildering variety of foraging techniques. The three that have been tried with the robot Ants so far are random foraging, swarm foraging, and radial foraging. The garden variety ants that can be found in most areas usually practice some variant of random foraging. The worker leaves the nest and heads off in some random direction. During the course of the expedition, the worker moves randomly until she finds a food source or gets tired. Random foraging with robots starts with all the Ants on one side of the ant farm and the food randomly distributed in the middle. The robots then move around randomly bouncing off of walls and each other. Whenever they bump into a piece of food, they stop and transmit *I-found-food* from their IR beacons. Experiments were conducted to see whether or not cooperation had an effect on the performance of the group. Performance was measured by how long it took for the robots to find all the particles of food. After several trials, it was clear that randomly bouncing off of the walls and not communicating was just as effective as cooperative communication.

Driver ants in Africa use a very different technique. They live in colonies of up to 20 million workers. When they forage, they leave their nest and head out along trails laid down by scouts. When they get away from the nest, they start to spread out, catching anything and everything in their path. They have been known to kill tethered horses, human infants, and have even been used to execute criminals! The robots are not nearly that fearsome, but they can still try and swarm. They were programmed to head towards light and then heads towards each other. They way, the group would all stay together and head in a similar direction. In reality, they would just bunch up and get in each other's way. This idea still needs more work, it is not very effective in its present state.

Desert ants use yet another kind of foraging pattern. Workers brave 140º mid-day heat to collect other arthropods that have dissected in those temperatures. Each forager picks her own individual direction and then she heads out, taking navigational cues from the sun. However, these ants do not cooperate. If they cannot carry it back by themselves, they just leave it there. On the robots, this type of foraging was implemented with cooperation. Each Ant would pick a direction at random, then start foraging. If they found food, they would turn and head in the opposite direction while transmitting the way they were heading when they found food. Other robots who received this transmission would head in the communicated direction. After several seconds of transmitting the position of food to their nestmates, they would then turn around and look for more food in the same direction. The ant farm was placed near a window in the lab, so that light from the sun would illuminate the playing area. The software assumed the sun to be in the east, and the robots used their four light sensors to navigate around the environment. The ability of the robots to head in the correct direction was surprisingly good, and the IR transmissions proved effective. However, these talents were balanced by their complete inability to stay away from the walls of the ant farm. They would pick a direction to forage in, run into a wall, and refuse to leave! The main cause of this was the lack of traction of the plastic treads, which can be corrected with rubber compounds.

Overall, the robots work reasonably well, but there are still some minor bugs to work out in the hardware. The software is far behind the hardware and has a lot room for improvement.

Chapter VI: The Next Step

Looking back

The integrated electromechanical three-dimensional PC board construction technique has proven to be cheap, easy, and reliable, which makes fabricating a large number of microrobots a realistic option. The hardware works reasonably well, with only a few problems remaining to be solved. The robots run very reliably, even after dropping them! This is very important for a successful community. With robots that break down often, much of your time and energy is spent just keeping a population ready for experimentation.

Solving Problems

There are a few problems with the hardware and several enhancements that would augment the abilities of the robots. First of all, the robots do not have working mandibles yet. There was just not enough time during the term to install them on the new robots. We also need more robots. A population of six robots is a start, but twenty would allow us to explore complex social interactions. More robots might provide a whole different level of community. Instead of having individual Ant interacting with other Ants to form the structure, you might have sub-groups interacting with other sub-groups. A larger colony of robots would also require a larger Ant Farm to operate in. The robots are equipped with tilt sensors and powerful motors, so the terrain need not be a flat rectangle, but can be a varied and interesting landscape.

Without question, the robots need more sensors. Their current long-range navigational capabilities are very limited, as only the light

sensors can receive information from further that 6 inches away from the robot. The compass and trail marker would open up many new navigational possibilities, in addition to making the robots more analogous to their natural counterparts.

The goal of building an artificial ant colony raises an important question. How will you know when you get it right? Maybe just observing behaviors is enough of a test. If they act like ants, then they must be working correctly. However, a more rigorous approach to the definition of community would be desirable. Again, a return to nature might provide us with a good reference from which to judge our efforts. Some criteria, like number of food particles gathered, or trip effectiveness, or colony efficiency, probably are applicable to natural as well as artificial ants, and could let us know how we measure up to the systems we are trying to emulate.

Applications

Thinking of applications for microrobots, cooperating or operating independently, is not difficult. In fact, it is difficult to think of applications where smaller robots or robots that can work together would not perform well. Cleaning, spying, foraging though pipes, or even performing surgery are all potential applications, and the list goes on as far as the imagination will take it. Maybe twenty years from now, you will have a colony of microrobots living under your refrigerator. When you turn the lights off, they will come out and pick up crumbs. Maybe they will even have little lasers to exterminate at any non-robot life in your kitchen!

Looking Ahead

The ultimate goal is to understand the underlying rules and structure of a community of simple autonomous agents. One of the best ways to do this is to continue to learn from the model, nature. Myrmecology is the branch of entomology involved with the study of ants. I have completed coursework on entomology at Harvard, and plan to continue to expand my knowledge in both these areas with independent reading, and hands-on research with live ants. It is not hard to argue that the behaviors of ants are some of the best examples for a robotic community.

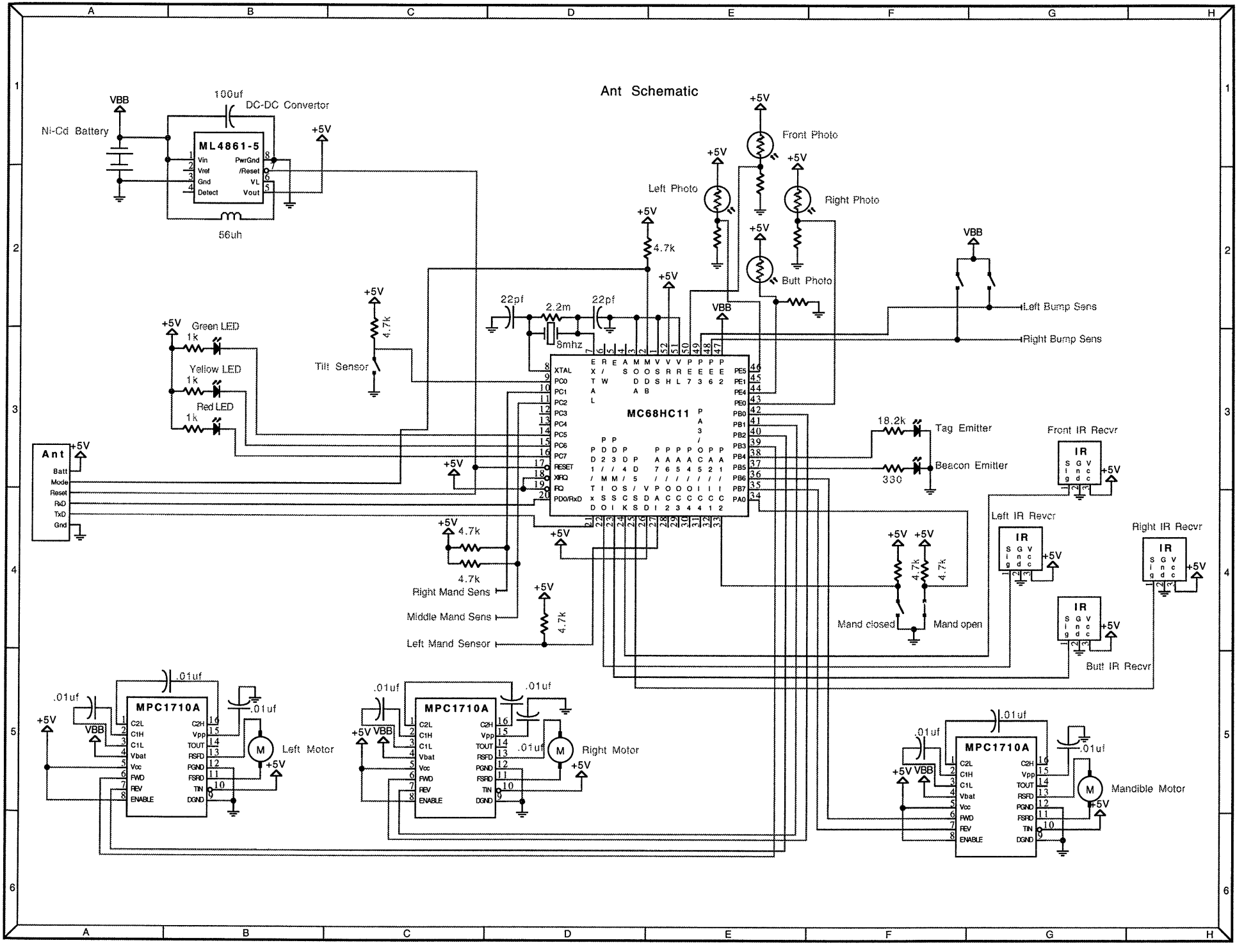
The second goal is to push the limits of micro-robotic systems. The current Ants have 17 sensors, 3 motors, a set of mandibles, and the ability to communicate with other robots, which is an order of magnitude more complicated that any other robot its size. This is the kind of integration that will enable us to get complex, useful behaviors out of microbots. This is also the kind of integration we would like to have on the integrated-circuit level, which will be the next step to making robots smaller.

Using nature as a guide, we hope to combine both of these ideas into a working community of microbots. With a little luck and a lot of hard work, maybe the Massachusetts Institute of Technology can be the home of the world's next guests to the robotic picnic, the Ants.

References

[1] Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, RA-2, pp. 14-23, April, 1986

Appendix A: Schematic


Ant Schematic
VBB
Ni-Cd Battery
100uf
DC-DC Convertor
ML4861-5
Vin
Vref
Gnd
Detect
PwrGnd
/Reset
VL
Vout
56uh
+5V
Front Photo
Left Photo
Right Photo
Butt Photo
4.7k
22pf
2.2m
8mhz
MC68HC11
Left Bump Sens
Right Bump Sens
Green LED
Yellow LED
Red LED
1k
Tilt Sensor
Ant
Batt
Mode
Reset
RxD
TxD
Gnd
18.2k
Tag Emitter
330
Beacon Emitter
Front IR Recvr
Left IR Recvr
Right IR Recvr
Butt IR Recvr
Right Mand Sens
Middle Mand Sens
Left Mand Sensor
Mand closed
Mand open
.01uf
MPC1710A
Left Motor
Right Motor
Mandible Motor

Appendix B: Tag

```
;;; Tag - Ver 3.0
;;; James McLurkin
;;; Copyright, Massachusetts Institute of Technology, 1994

;;; ***History***
;;;
;;; 7/21/94 James: Created
;;;
;;; ***End history***

(def8kant ant

;;;     Variable definitions

  (basic-ant-variables)
  (allocate-data gottaggeddelay 1)
  (allocate-data oldtilt 1)
  (=c itsignal          1)
  (=c notitsignal       3)
  (=c ittagsignal       7)
  (=c gottaggedsignal   5)

;;;     Main Loop

  (basic-ant-startup)
  (subsumption-loop)

;;;     Moods

startupmood
notitmood
  (!8 nosignal)
  (!16 moveforwardslowly)          (!8 flashinggreenlight) (!8 notitsignal)
  (!16 movefromnotit)        (!8 dontsubsume) (!8 dontsubsume)
  (!16 movefromit)           (!8 dontsubsume) (!8 dontsubsume)
  (!16 movefromtaggedant)    (!8 dontsubsume)   (!8 dontsubsume)
  (!16 movefrombumps)        (!8 dontsubsume) (!8 dontsubsume)
  (!16 didIgettagged?)       (!8 (+ redlight greenlight yellowlight)) (!8 gottaggedsignal )
)
  (!16 amItilted?)           (!8 dontsubsume) (!8 dontsubsume)
  (!16 end)

itmood
  (!8 ittagsignal)
  (!16 moveforwardturbo)           (!8 flashingredlight) (!8 itsignal)
  (!16 movetonotit)          (!8 dontsubsume) (!8 dontsubsume)
  (!16 didItagsomebody?)     (!8 dontsubsume) (!8 dontsubsume)
  (!16 movefrombumps)        (!8 dontsubsume) (!8 dontsubsume)
  (!16 xmittagifbumping)     (!8 redlight) (!8 dontsubsume)
  (!16 amItilted?)           (!8 dontsubsume) (!8 dontsubsume)
  (!16 end)

;;;     Tag Behaviors

didItagsomebody?
  (ldaa (! gottaggedsignal))
  (jsr findirsignal)
  (beq ciftinosig)
  (ldd (! notitmood))
  (jsr changemood)
  (jsr clearirregs)
ciftinosig
  (rts)
```

```
didIgettagged?
  (ldaa gottaggeddelay)
  (beq lookfortagsignal)
  (cmpa (! 1))
  (bne sitthereaftertag)
  (ldd (! itmood))
  (jsr changemood)
  (jsr clearirregs)
  (clr gottaggeddelay)
  (rts)
lookfortagsignal
  (ldaa (! ittagsignal))
  (jsr findirsignal)
  (beq didIgettaggeddone)
  (ldaa (! 8))
  (staa gottaggeddelay)
sitthereaftertag
  (ldaa (! mstop))
  (jsr subsumemove)
didIgettaggeddone
  (rts)

movetonotit
  (ldaa (! notitsignal))
  (ldy (! findfront-moveforward))
  (jsr movetoirsignal)
  (rts)

movefromit
  (ldaa (! notitsignal))
  (ldy (! findbutt-moveforward))
  (jsr movetoirsignal)
  (rts)

movefromnotit
  (ldaa (! notitsignal))
  (ldy (! findbutt-moveforward))
  (jsr movetoirsignal)
  (rts)

movefromtaggedant
  (ldaa (! gottaggedsignal))
  (ldy (! findbutt-moveforward))
  (jsr movetoirsignal)
  (rts)

amItilted?
  (ldaa tilt)
  (beq notilttag)
  (ldaa oldtilt)
  (bne notilttag)
  (ldd mood)
  (cpd (! notitmood))
  (beq tiltedandnotit)
  (ldd (! notitmood))
  (bra tiltchangemood)
tiltedandnotit
  (ldd (! itmood))
tiltchangemood
  (jsr changemood)
notilttag
  (ldaa tilt)
```

```
  (staa oldtilt)
  (rts)

;;; Everything Else

800uscounters
  (basic800uscounters)
50hzcounters
  (basic50hzcounters)
2hzcounters
  (!8 gottaggeddelay) (basic2hzcounters)

  (simple-behaviors)
  (simple-serial)
  (basic-ant-routines)
  (interrupt-vectors :start start
                     :rti 50hzinterrupt
                     :serial serial-interrupt
                     :ocmp1 ir-interrupt
                    ))

;(assemble ant t)
(antdload ant)
```