University of Florida
Department of Electrical and Computer Engineering
EEL5666L
Intelligent Machine Design Laboratory

2-Inch Worm

Final Technical Report

Chris Conger
August 6, 2004

TAs: William Dubel, Max Koessick
Instructors: Dr. A. Arroyo, Dr. E. Schwartz

Table of Contents

Abstract:

2-Inch Worm is a robot designed to scale trees and telephone poles. The robot design is inspired by the movement of centipedes, and achieves motion by utilizing multiple sets of legs that move independently to slowly crawl its way upwards. While climbing, 2-Inch Worm will also monitor its height, and after climbing as high as possible and returning down it will report how high it was able to climb.

Executive Summary

2-Inch Worm is an autonomous climbing robot designed to scale trees, poles, and other cylindrical surfaces. The robot is required to monitor its environment as best as possible, and be aware when it is no longer safe to climb. Upon completion of a climb, the robot will report the maximum height obtained by using a downward-facing sonar device.

Movement is achieved by using 3 identical segments, each involving a servo, gear, and rack, as well as another servo and matching pincer mechanism. The pincer is used to grip the trunk, and the rack and gear mechanism is used to drive the pincers up and down the body of the robot, as well as hoist the robot up the tree. A sonar device mounted on the bottom of the robot monitors the height, and IR sensors located on the head facing up and on the belly facing the trunk monitor the robot's alignment and watch for approaching obstacles.

The mechanical aspect of this robot is much more complex than the electrical or software components, and as such the successful completion of a functional body took the majority of time spent. The functionality of the sensors is elementary, and is easily incorporated into the design after the mechanical design functions successfully.

While the completed robot climbs successfully, its stability on the trunk is very fragile, and must be monitored closely to avoid an emotionally and monetarily devastating tumble. The successful scaling of a post demonstrates the proof-of-concept for this climber design.

2 Introduction

Mobile robots come in many varieties relative to the environment in which they are to operate. From airborne, to watercraft, to ground-based, each environment allows for many different methods of mobility. Ground-based robots employ methods of motion such as wheels, tracks, legs, even pistons/jumping mechanisms. However, there is one class of robots that isn't exactly ground-based, yet cannot be considered airborne either. Robots that are intended to climb vertical surfaces present unique challenges to the mechanics of movement, and to maintaining stability.

The problem of constructing a robot for vertical, flight-less mobility is challenging because a surface on which to move cannot be taken for granted. Ground-based robots are stable due to the pull of gravity (as long as they are constructed sensibly), whereas gravity creates a very annoying problem for climbing robots. In addition to the constant pull of gravity, applying lateral force on the surface being climbed will have the effect of pushing the robot backwards. The need to maintain a constant, strong grip on the vertical surface limits maneuverability and presents a challenging mechanical problem. Lastly, the environment in which climbing robots operate leaves a small margin of survival in the face of a slip or other failure.

The purpose of this project is to construct a robot which can climb real trees of varying sizes and types. In order to accomplish this goal, the robot will have to provide the following abilities; (1) be able to move up, as well as *down* the tree, (2) detect approaching limbs, knots, and forks in the tree, (3) recognize when it cannot successfully climb any higher on the tree, *and not try to*, (4) be able to adjust the grip for a range of

trunk diameters, (5) keep track of its current height, as well as the maximum height obtained. Since the robot should pass branches if possible, the robot will not be able to encircle the tree and thus must be able to cling on to the side.

Presented in this paper is *2-Inch Worm*, a robot designed to climb trees. Soon a conceptual description of the robot and how it will accomplish the specified requirements will be given. Following the conceptual walk-through, a technical description of all body parts, motors, sensors, and circuits is provided. Lastly, the operation and performance of the completed robot will be discussed, along with concluding remarks about the project and suggestions for future work or improvements.

3 Integrated System

2-Inch Worm is a robot, designed with the shape and movement of a centipede in mind. The robot will be constructed of multiple *segments*, each one an exact replica of the others. Weight is distributed as evenly as possible, and an attempt is made to keep the weight minimal. The frame is constructed out of ¼"-thick project wood (it only makes sense to have a tree climb a tree), and to minimize weight the segments are kept as small as permitted by required motor, battery, and circuit-size. Each segment consists of a small platform that can slide parallel to the body, and each segment operates one pair of aluminum pincers that all work together to attach and move the robot. The microcontroller board and power circuitry is mounted on the *back* of 2-Inch Worm, with a sonar sensor on the *tail* facing downward to monitor its height.

2-Inch Worm is named such because of the range of motion available to each segment, resulting in the robot moving 2 inches at a time. The pincers are controlled by two servo motors; one motor is to drive the mounted, hinged pinchers up and down a track, and the other controls the opening and closing of the pincers. Since every body segment is nearly identical, each segment simply needs to be able to lift its own weight; slightly more power is used as a safety factor. Body segments are fixed together using metal rods that run the entire length of the body, which also provide a 'track' for the pincers to slide up and down on.

Movement is achieved in a two-phase fashion. Legs are all adjusted one pair at a time during the first phase, and then the robot is driven up a small amount by combining all segment motors for the second phase. Since the movement is not continuous, keeping the climbing speed from being painfully slow is quite a challenge. The robot also detects forks in the tree as well as impassable knots and branches. Upon reaching a barrier of some sort, the robot reverses and proceeds to climb back to its original height.

To minimize the size of any individual segment and keep them all consistent, only the main microcontroller board is mounted on the head. Small, individual circuits for receiving signals from the "brain" and controlling the motors are mounted on each segment. All segments are connected by a chain of wires and headers. Programming of the robot is very simple relative to the mechanics of the design. The behavior is comprised mainly of a repeating forward or reverse movement routine, the sonar monitoring, and the reaction to IR sensors (which just triggers the reversal of movement).

The robot is very mechanically complex; as such, the body parts are all designed using AutoCAD and significant care is taken to obtain high-quality, custom parts. The most challenging of requirements is also the most fundamental, to attach to and climb up and down a tree. Obstacle detection and height monitoring are trivial compared to the challenge of defying gravity. Using the right balance of a weight/power tradeoff of actuators and a creative mechanical design, the 2-Inch Worm attains each of the goals set forth and looks suave while doing it.

4 Mobile Platform

This section will describe and discuss the robot's body design. The unique operating environment of 2-Inch Worm leads to an uncommon body shape, and each of the features of the body are designed as such for a specific purpose. First the design process is described with an emphasis on addressing each of the inherent challenges of the robot, followed by a discussion of the actual implementation of the body frame.

4.1 Platform Design

The robot is intended to 'cling' onto the side of a tree, as opposed to wrap around it or encircle the trunk. The challenges associated with such an approach include things such as a need to achieve sufficient lateral force against the tree to maintain grip, keep weight as well as power consumption to a minimum, as well as provide sufficient room for movement while minimizing the size and distance from the tree. Every surface of the body is utilized, making the handling of the robot awkward. The platform has two basic parts, the main body and small, movable segments; all parts are made of ¼"-inch thick pine board.

The main body is basically a long, flat piece of wood constructed to house all circuitry and batteries on one side, and to facilitate the motors and actuation on the other side. Since the farther the robot sits away from the tree, the more force it will exert on the point of grip, this flat design makes optimum use of a single plane for everything. Figure 3.1 below shows the design of the main body segment.
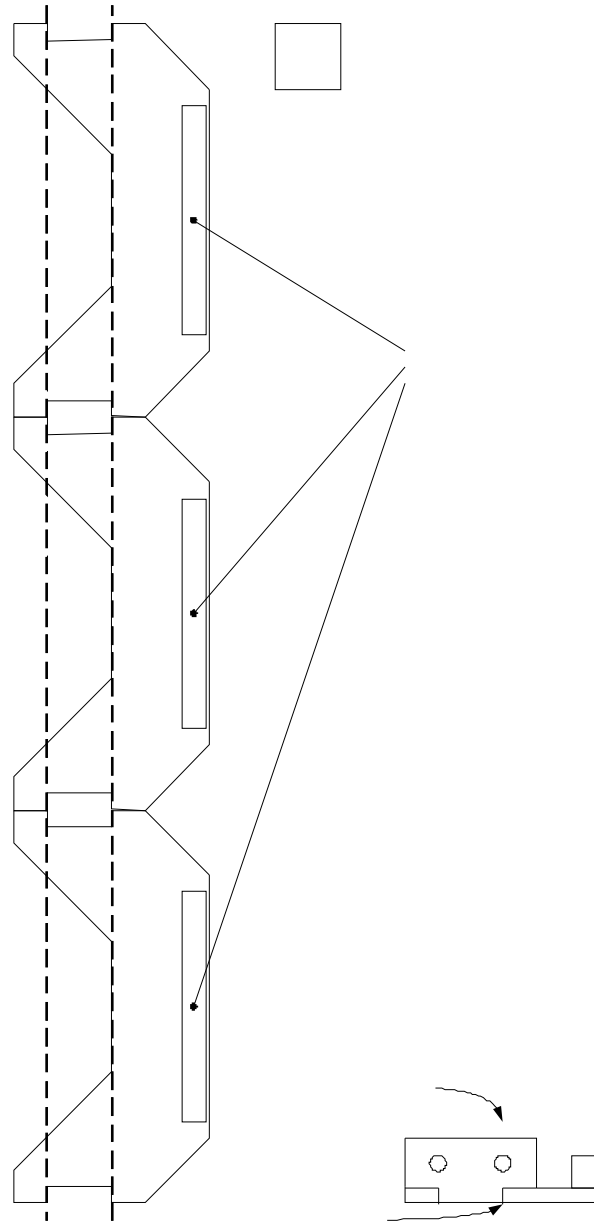

Figure 3.1 - Main body diagram

The dotted lines seen along the length of the body represent where the two steel rods will be running. On the right side of the body, each of the three segments has an elevated block so that a rack (as in rack and pinion) may be mounted at the right height. To minimize weight, it can be seen that sections of each segment which did not need to have something attached were cut out leaving the three trapezoid-shaped gaps on the left side. For further weight reduction, many small holes were cut into every surface where possible; care was taken to not cut out too much so that the strength of the chassis was jeopardized.

In addition to the main body section discussed above, there are also 3 small platforms that slide up and down the rods, also providing a mobile platform for the motors. Figure 3.2 shows the design of one of these platforms, though all three are identical.
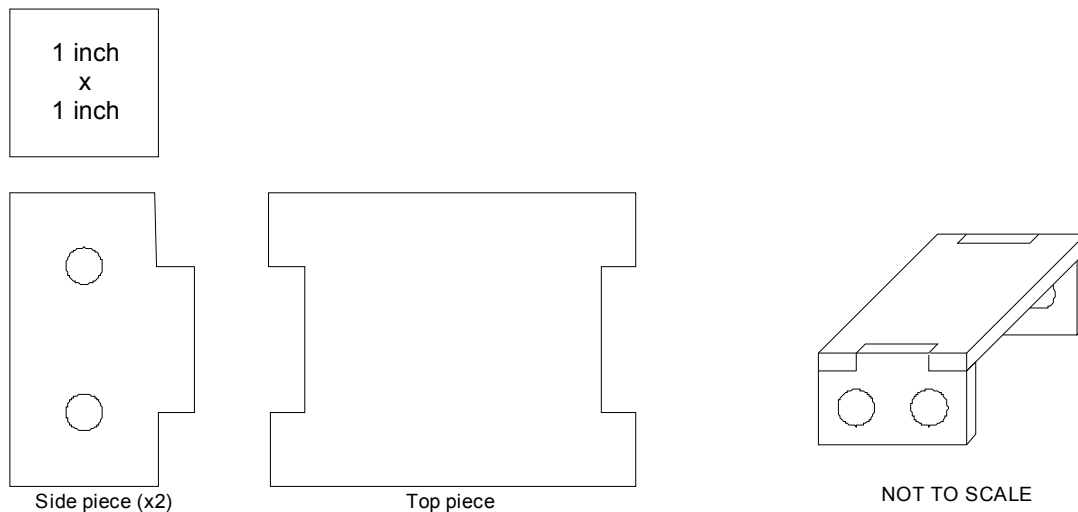

Figure 3.2 - One of three small platforms

Each of the small segments glides along the two metal rods. The rods provide a track for linear movement, as well as providing stability to keep the motor and gear which will go on it in line with the rack (and unable to pull away from it).

4.2 Platform Implementation

The platform was cut out of ¼"-thick pine wood. Wood glue was used to attach the rack blocks and support plates (also made out of the same pine) for the metal rods, and provided a more-than-sufficiently strong bond to create a very rigid chassis. The racks themselves started as a 2'-long piece, out of which three 3" lengths were cut. To attach the racks to the elevated rack blocks, a mixed epoxy proved to create a very strong bond and keeps the racks permanently in place very nicely. The bare body piece is shown below in Figure 3.3, followed by a picture of the fully assembled robot in Figure 3.4, including the visible circuitry mounted on the back of the robot.

Figure 3.3 - Main chassis
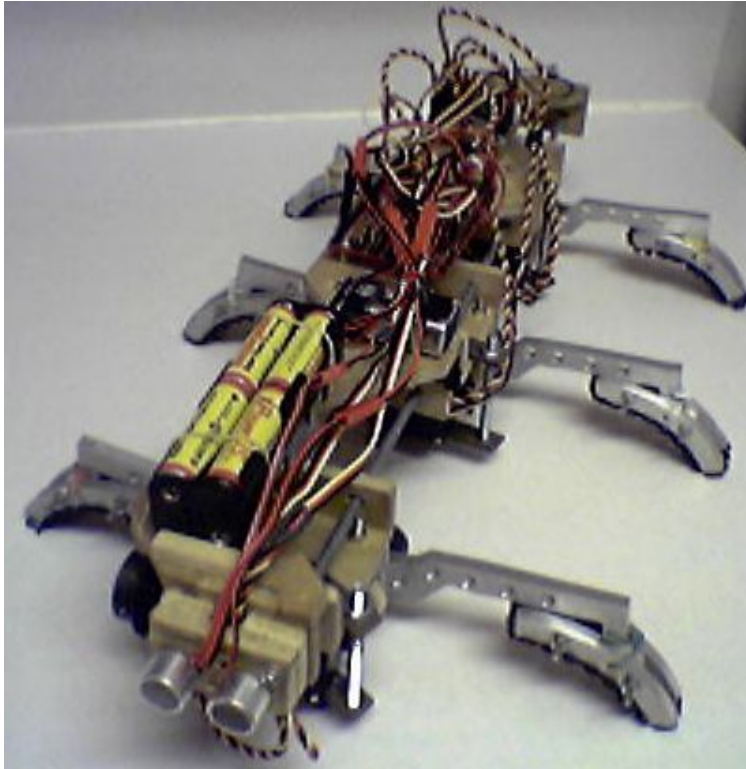
Figure 3.4 - Fully assembled platform

One big problem with this body design is in fact the rigidity of the whole body. Ideally, each segment would be hinged together so that the body could flex, thus adapting to small curves. However, it was soon discovered that the only way to have the segments hinged was to ensure the hinges were powered or controlled with motors. If not, imagine for example what would happen when the top segment opened its pincers; with no grip on the tree, and a free-moving hinge, the segment would fall backwards quickly and the robot would be seriously jeopardized. The rigid connection with other segments is required in order to provide support to a segment which is adjusting its grip. This does impose a requirement to ensure the robot is traveling parallel to the tree at all times since it cannot flex, but this is an easier (or at least more practical) task to achieve than powered, controlled hinges between segments.

5 Actuation

By far the most complex part of this project was the design, and furthermore the implementation of the mechanisms for movement. The actuation design of 2-Inch Worm is a completely original idea, not inspired or adopted from another robot. Movement is achieved by the cooperation of two kinds of motors, one that drives the robot up and down the tree, and the other that tightens or loosens the grip on the tree. Presented in this section is a description of each individual section, followed by a discussion of the complete motor assembly.

5.1 Segment Movement

The three small platforms that slide up and down the rods are driven by a high-torque servo motor capable of 133 oz-in of torque. The most critical measurement for the placement of all other parts is the size of the gear used to push up and down the rack. Considering that the servo is capable of approximately 180˚ rotation, and 2 inches is the desired range of movement, the required gear diameter is easily calculated with the equation:

```
2 inches = π x (gear diameter) / 2
```

```
gear diameter = 1.27 inches
```

The required value falls very close to the common diameter of 1 ¼" inches, so that size was chosen. With this known, the height of the metal rods from the main body piece could be determined. The servo motor is laid flat on the platform, with the shaft pointing just over the edge. Figure 4.1 shows the servo and gear assembly for lateral movement; there are two motors shown, the bottom one with the large gear facing out is the segment

motor. Observing the rod holes at the bottom, one can see how the platform sits right in line with the rack.
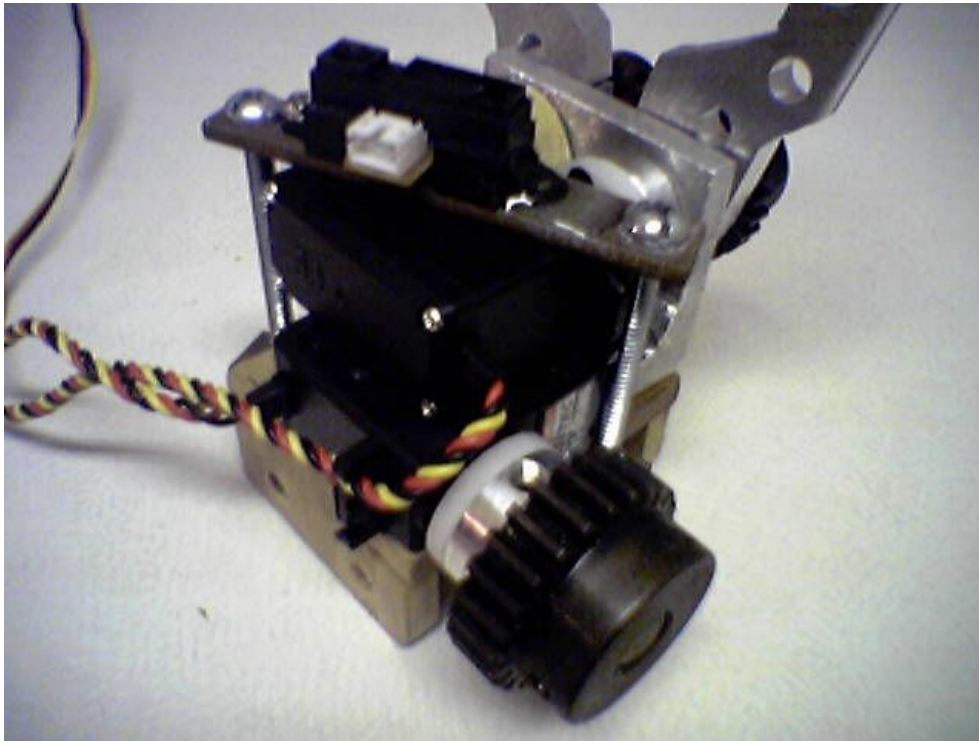
Figure 4.1 - Segment motor and gear assembly

Friction did not turn out to be a problem with the platform sliding up and down the rods. Considering the motor torque (133 oz-in), and the gear radius of 5/8", the force in pounds can be calculated by:

```
(133 oz in)*(1 lb/16 oz)/(5/8 in) = 13.3 lb!!
```

Obviously there is sufficient power in any one segment, let alone all 3 together to hoist the body of the robot. Also, the requirements for motion only demand two possible motor positions, either all the way clock-wise, or all the way counter-clockwise. Since the servo is driven by PWM, programming the servo control was extremely easy. A pair of functions named 'segment()' and 'all_segments()' are used, though they simply set the timer to certain values for each of the channels depending on the chosen segment and

position. Since there is no way to know when the motor completes a movement, each function also has a delay associated with it that simply waits a specified amount of time before executing any further instructions to give the motor time to complete. The tuning of these exact delays came from actual experimentation during design, and were picked to provide the smoothest motion possible.

5.2 Pincer Mechanism

The most difficult mechanism in the entire project was the design of the pincers. There are many small details that required significant brainstorming to decide on, such as the optimum length and shape of the pincer arms, how to attach all of the components in a secure fashion, as well as just the general exact dimensions of all the parts so that they would all line up. A custom pincer design was chosen over a purchased pincer to be able to minimize the required space, and fit the mechanism cleanly onto the already space-limited platform. All components of the pincer mechanism were constructed out of ¼"-thick aluminum, with the exception of the shafts which are simply 3/8" wooden dowels.

In order to power the pincers with only one motor, the two arms were fixed to parallel shafts and interlocked with gears so that rotating one shaft would force the other to rotate as well. Using an even number of gears results in the desired opposite direction of rotation so that a squeezing motion is achieved. The motor to power the pincers is laid on top of the motor driving the segment, and a custom mounting plate is used to support the shafts at the correct height to line up with the pincer servo. The same high torque servo as used for the segment is also used for each pincer motor, meaning that all 6 motors are identical. The figure below shows the pincers as seen from a front-view:
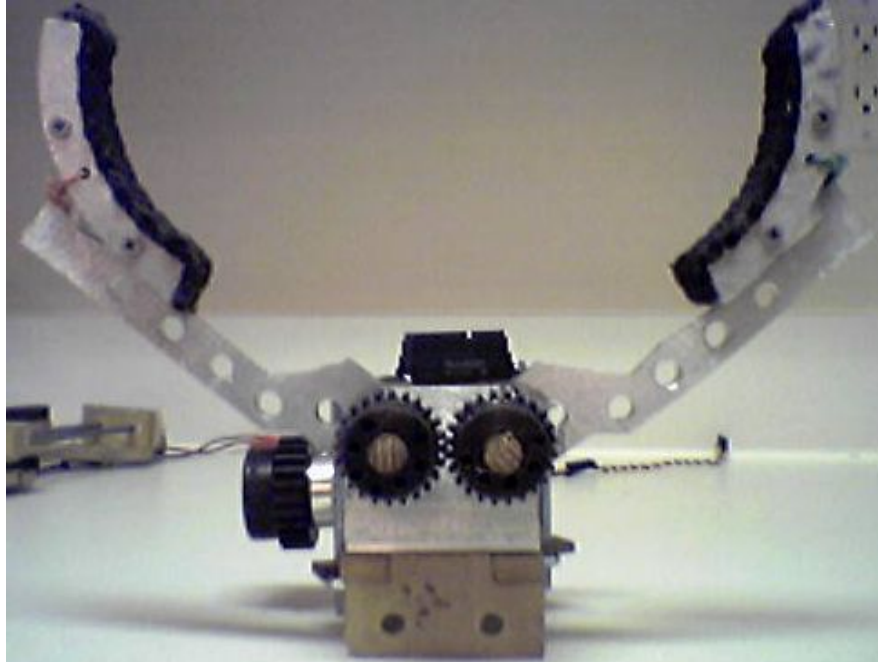
Figure 4.2 - Pincer mechanism

Similar to the segment motors, the pincers are always at one of two positions, *open* or *closed*. Similar simple functions to those used for the segment motors were created for the pincer motors as well. The last aspect of the pincer design which is very important is the use of the "feet", which can be seen as the aluminum arcs at the ends of the pincer arms. The design decided on was optimized for scaling a rounded trunk, and the metal feet themselves are padded with a rubber lining to help provide friction at the point of contact. The feet are tightened around the end of the pincer arms, holding them in place at any desired angle within the feet's limited range of motion. With more time and money, other variations of feet could be designed and substituted due to the modular design of the pincers, such as feet with sharp teeth to dig into the surface being climbed, or spring-supported feet that automatically adjust to the shape of the trunk being scaled.

5.3 Complete Motor-Pincer-Segment Assembly

The complete segment assembly is simply the combination of both mechanisms previously described. The motors are stacked one on top of the other, and held in place using two small steel plates and bolts on either side to tighten it down and hold the motors securely. Cardboard shims were placed between the motors and mounting plates as necessary to adjust the positions of various parts so they would line up, though this was done minimally. Each segment is fairly heavy, due in large part to the weight of the gears as well as motors and metal pincer arms and feet. A close-up of the completed assembly is shown in Figure 4.3, including the sandwich plates used to secure the motors:

Figure 4.3 - Complete segment assembly

6 Sensors

To support the required sensory capabilities of 2-Inch Worm, the designer has chosen the classic IR sensors, as well as a sonar device. Each sensor type plays a specific role in the operation of the robot, highlighted in this section.

6.1 IR Sensors

The infrared sensors selected for 2-Inch Worm are Sharp's GP2D12. This sensor is very small (approx. 1½" × ½"), and provides object detection up to a theoretical limit of 80 cm away. However since 2-Inch Worm will be operating outdoors during the day, the detection capabilities of any IR device is likely to be diminished somewhat due to ambient sunlight. The Sharp GP2D12 interfaces very conveniently with the MAVRIC-II board which controls the robot, making this product a wise choice for IR sensing. The IR sensors will be used to monitor the distance of the robot body from the tree trunk, as well as (hopefully) assist in detecting approaching branches and other obstacles.

6.1.1 IR – Physical Setup

The GP2D12 sensor is interfaced with only 3 wires; (1) power, (2) ground, and (3) signal. The power and ground wires connect directly to the auxiliary power supply of the MAVRIC-II board, and the signal line is fed directly to an ADC input channel on the board. A total of 3 IR sensors are present in the system, providing distance monitoring from various points on the robot.

Two of the sensors are mounted on the "bottom" (the side facing the tree, not the ground) of the robot, one at the head and one at the tail. These two sensors monitor the distance of the robot body from the tree trunk, and provide the ability to recognize if the robot is not moving in a stable path along the side of the tree. In addition to ensuring that the readings each fall within a certain acceptable range, the two readings will also be compared with one another. If for example, the head sensor reads a significantly larger

distance than the tail sensor, it would indicate that the robot is not moving in a stable path along the tree and will likely lose its grip if it continues to climb.

The third IR sensor is mounted on the nose of the robot facing upwards, and is intended to assist in detection of approaching branches or other protrusions. This sensor is complimented by bump sensors and whiskers (discussed later) as it is believed that the sunlight combined with the thin-ness of many branches may cause this sensor to be ineffective.

6.1.2 IR – Software Implementation

The software that interprets and controls the IR detection is very simple and straight-forward. The ultimate purpose of the IR detection is to ensure that it is safe to continue to climb. Since motion of the robot is not continuous, the software takes advantage of that fact by using the period in between movements to take sensor readings. Receiving IR data is done with simple conditionals that compare the IR readings to various thresholds. Depending on the outcome of these comparisons, the robot decides if it is safe to continue climbing or if it should begin descending.

The actual C functions used to control the ADC and take readings from various channels are provided from the MAVRIC-II board vendor (BDMicro). These procedures include ADC initialization and configuration, and channel reading. As mentioned above, the sensors are checked each time the robot moves to a new position. By using simple if-statements the actions of the robot are controlled accordingly.

6.2 Sonar Sensor

The sonar sensor used by 2-Inch Worm is the Devantech SRF08 Ultrasonic Range Finder. This device emits high-frequency audio signals and measures the first 17 echoes received, with distances up to approximately 6 meters. The sonar sensor is to be used solely to measure the height of the robot at any given time.

6.2.1 Sonar – Physical Setup

The sonar device is located on the tail of the robot, facing straight down. Since the robot will (ideally) always be oriented in approximately the same direction, the sensor may be directly attached to the frame of the robot. Interfacing the sonar device to the MAVRIC-II board is extremely simple, with only 4 lines that may be connected directly to the board without any additional circuitry. The device operates with the I2C bus protocol, and thus only requires a power and ground line, plus the two I2C signals (clock and data). Power and ground may be drawn from the auxiliary power supplies of the board, and the two I2C lines have dedicated pins on port B of the ATmega128. The diagram below illustrates how the SRF08 is wired to the MAVRIC-II, taken from www.bdmicro.com:
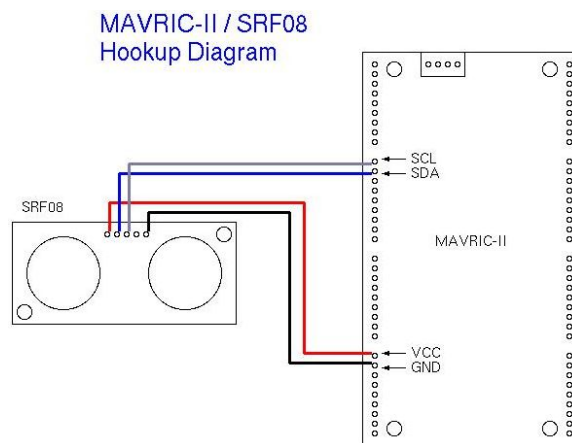

Figure 5.1 - SRF08 interface to the MAVRIC-II board

6.2.2 Sonar – Software Implementation

Like the GP2D12 IR sensor software, basic C functions were obtained from BDMicro. While these functions include basic operations such as various I2C bus controls and SRF08 pinging and reading, a higher-level function has been written that makes use of these basics to provide a single function call to return the robot height in feet. The device itself stores 17 range readings for each ping, so all of these values must be read, and the range corresponding to the ground determined from all of these readings. Since the possibility exists that at certain points a tree trunk may lean, a small degree of error will have to be tolerated as the sonar may be pointed at an angle to the ground.

The highest-level function that returns the robot height in feet is composed of three basic parts. First, the sonar simply emits a single ping and pauses for 100 ms. After these 100 ms elapse, the second phase reads each of the 17 ranges from the SRF08 device memory and stores it in an array. Finally, this array is searched to determine the correct range to ground. The search method is simply by selecting the last echo received, which would correspond to the ground.

The device can return range readings in inches, centimeters, or microseconds, so the inch-configuration was chosen to make an easy conversion to feet before returning the single value. Like the IR sensors, the sonar will take readings in between each movement. Once the robot has decided through the other sensors that it should begin descending, the sonar will take one last reading and record this as the maximum height obtained. The body of the robot is 1½' long, so this is added to the final reading as the official height reached.

7 Behaviors

The behaviors of 2-Inch Worm are relatively simple, consisting of a climbing sequence, and reaction to periodic sensor measurements. The main challenge of this project was getting a robot to climb something period, so to help make that goal more feasible the behavior was chosen to be very simple on purpose. Essentially 2-Inch Worm will start wherever it is placed on a tree, then begin to climb up. As soon as it is deemed unsafe to continue climbing up, for example if a branch or bend in the tree is approaching or if the robot is no longer parallel with the trunk for whatever reason, the robot will stop and measure how high it is. Then it will reverse direction and climb back down the trunk to its original height, and display how high it climbed.

Let us first consider the sequence to climb up. Recall that each segment has an independently operated pair of pincers for grip, and 2" of linear movement. From the starting stationary position, all three pairs of pincers are *closed*, and all three segments are *down*. From this position, the bottom segment goes first an opens its pincers, moves its segment to the *up* position, then closes it pincers on the trunk again. Next the middle segment repeats this pattern, followed by the top segment. Notice that each time any one segment lets go of its grip, the other two segments are maintaining their grip so that the body is supported. Once all three segments are in the *up* position and all pincers are again *closed*, the final movement is for all three segment motors to move to the *down* position. Since the pincers are still closed around the trunk, the result is to drive the large body piece upwards. From here, the process may be repeated. In order to climb down, the sequence is very similar. The process starts with all segments in the up position, and the segments are individually adjusted to the down position. Then the body is moved

down 2 inches by leaving the pincers closed and driving the segments to the up position at the same time.

Using the functions defined in the appendix, the following pseudo-code demonstrates how the climbing (upwards, in this example) is actually implemented in the robot's software:

```
ALL_PINCERS(CLOSED);        // just to be sure they're all closed
ALL_SEGMENTS(DOWN);

PINCER(LOWER,OPEN);
SEGMENT(LOWER,UP);
PINCER(LOWER,CLOSED);
PINCER(MIDDLE,OPEN);
SEGMENT(MIDDLE,UP);
PINCER(MIDDLE,CLOSED);
PINCER(UPPER,OPEN);
SEGMENT(UPPER,UP);
PINCER(UPPER,CLOSED);
```

To control the direction change, the sensors are all scanned in between iterations of the above loop. If the sonar reports that the robot did not successfully move upwards (such as the height reading being the same or close to the previously measured height), or if the IR sensors detect that the robot is not parallel to the trunk or there is an object approaching, the robot will complete its last height measurement and then attempt to climb back down. However, if the robot detects that it is not parallel to the trunk, it will simply close all of its pincers and stay put. This is semi-dangerous, as the operator would need to be aware of this and remove the robot before it runs out of power and falls. This is also true of when the robot completes it's climb successfully, as it will simply sit at its original height until it is removed. The pseudo-code below illustrates the sensor scan

algorithm that is completed periodically. The overall goal is to set a variable called

ok_to_continue at either '0' (safe to keep climbing up) or some positive value as an error

code.

```
(global variable 'altitude' holds current height)
ok_to_continue = 0
A = read_sonar();
ir1 = read_IR(top);        // IR distances in cm
ir2 = read_IR(bottom);
ir3 = read_IR(head);

if (A <= altitude) ok_to_continue = 1;
altitude = height;

if (ir1 < ir2 - 1) OR (ir1 > ir2 + 1) ok_to_continue = 2;
if (ir3 <= 10) ok_to_continue = 3;

return ok_to_continue;
```

The final software structure that defines the operation of the robot is as follows. The

pseudo-code above resides in a function named 'sensors()', and the climbing algorithms

for ascending and descending are represented as functions 'climb_up()' and 'climb_down

(),' respectively.

```
place robot on tree;

altitude = read_sonar();
all_clear = 0;

while (all_clear == 0){
     climb_up();     // complete one iteration of climb sequence
     all_clear = sensors();
}
max_height = read_sonar();     // can't go higher, take reading

while (altitude > start_height){
     climb_down();
     altitude = read_sonar();
}
```

```
send_to_LCD(max_height);
```

8 Experimental Layout and Results

Presented in this section is experimental data obtained from testing the operation of the IR and sonar sensors. For each sensor, a description of the experimental setup is offered followed by graphs illustrating the results obtained from these initial sensor tests. Following the sensor experiments is a brief description of the test procedures employed while getting the robot to actually climb.

8.1 IR Sensor

To test the IR sensor, these first tests were run indoors by holding an obstacle at various distances from the sensor, and recording the voltage returned. The initial tests were run indoors so that the board could remain attached to the workstation used to program and debug the robot. The performance of the sensors will have to be also tested outdoors, although these tests have yet to be performed. The purpose of this initial test is to simply verify and validate the functionality of both the device itself and the software functions used to control the microcontroller and sensor. Figure 2 illustrates the voltage vs. distance relationship of the sensor. The points indicate actual measured values, and are accompanied by a blue trend line.
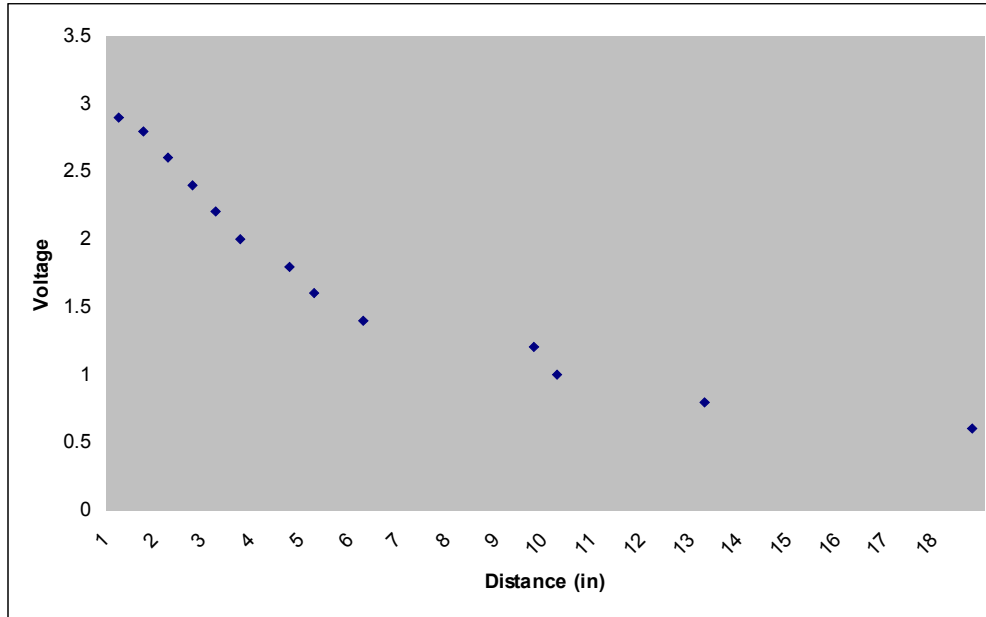
Figure 7.1 - Voltage vs. Distance relationship of the GP2D12

8.2 Sonar Sensor

One test was run for the sonar device, to simply verify operation and to get an idea for the typical error in measurement. For the verification test, the SRF08 device was aimed across an empty room and a target was moved to various positions/distances. The device is very accurate, so it is seen that the range returned matches the actual target distance almost exactly as shown in Figure 3. The average error in readings was approximately 1 to 2 inches. One surprising result from this test was the amount of errant echoes recorded by the device. It was at times difficult to determine which reading corresponded to the target, especially as the target got farther and farther away. The tests only go out to 10 feet, although the sonar should be able to read farther. Empty space limitations kept the tests from being performed on longer target distances.
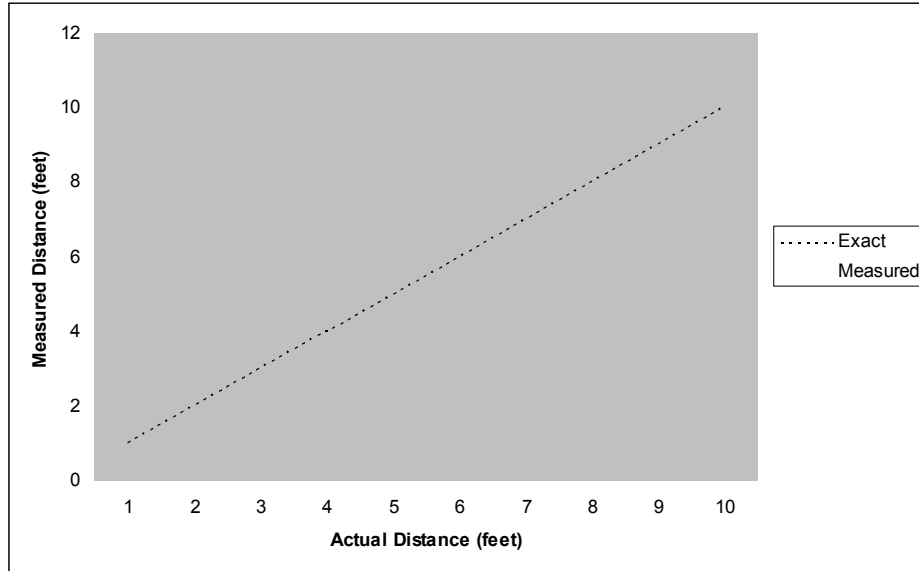
Figure 7.2 - Sonar operation verification and validation

8.3 Climbing Experiments

Experimentation for getting the robot to climb was actually relatively simple. As mentioned in the introduction, the operation of the robot is very binary; either it is working perfectly or it is falling to the ground. Also, the simplicity of its objective makes simply getting it to climb sufficient. Three steps were taken in getting the robot to climb successfully, (1) get the robot to hold on to the trunk by itself, stationary, (2) getting the robot to hold on to the tree and move all segments from *up* to *down* positions and vice versa, and (3) get the robot to complete the segment adjusting sequence. With all three of these operations working, the robot will successfully be able to climb.

For the sake of practicality, a 'post' was used during design as opposed to an actual tree (a post can sit indoors right by your workbench). The post is constructed of a 4" PVC pipe, 5' tall and planted in a pot filled with cement. The PVC pipe was then sprayed with

general-purpose spray-on rubber coating to help increase friction over the smooth PVC surface.

 The first two objectives, getting the robot to hang on as well as hoist the chassis up and down, worked flawlessly on the first try. However, the adjustment of the legs at first did not work, as each time a pincer would let go the opening would jar the robot and the other two pincers holding on did not have enough grip to keep the robot perfectly still. The result was that the robot would lose its alignment with the tree, and if it continued to climb it would walk itself off the post. Additionally, once all motors were being used, the power supply began to get extremely hot.

The first remedy for both problems was beefing up the power supply. Initially a single voltage regulator and heat sink powering all 6 motors, this was changed to be two independent voltage regulators powered by separate sets of batteries (for double current delivery). Each regulator powered three motors, and the pincer motors which draw by far the most current were staggered across the two regulators as best as possible. Since each regulator is identical, it does not matter which motors are powered by which regulator. This doubled current delivery, which enabled the power to stay strong during maximum effort, as well as helped cooling by splitting the power as evenly as possible between two heat sinks.

9 Conclusion

The design and construction of 2-Inch Worm was a phenomenal learning experience, and saw for the most part success towards the goals being pursued. While it never was shown

to be capable of handling any arbitrary tree, it certainly did accomplish the difficult task of scaling a vertical surface. Likely the most limiting part of the design is either the delicate balance required to maintain operation, or also the limited range of diameters that 2-Inch Worm can hold on to, no larger than 6". It is very easy for the robot to slip or loose its grip, and such a mistake would spell disaster if not watched closely.

The designer is a computer engineering graduate student, so the success of the body and pincer design itself was a very satisfying accomplishment. Careful planning was required to get the exact dimensions of many of the parts correct so that everything would line up, and the pure design of the platform-rail system and pincer mechanism was a resounding success. Likely the only thing that might be done differently if the project was to be done over, would be to pay more attention to the strength, as well as power consumption of the pincers. Techniques such as gearing down the servo, or adding a return spring to the pincers themselves could help add strength to the robot's grip. The weight of the robot was higher than desired, though aside from using weaker nylon gears (the gears were by far the heaviest part of the robot) it is still unknown how weight could be further reduced as the size is already minimal.

The conceptual design and operation of the robot, in addition to the success as demonstrated in the lab is a perfect proof-of-concept for this robot. For other designers of climbing robots, one general rule of thumb I would recommend in designing their robots is to go with the philosophy of 'overkill' when designing the gripping mechanism. A strong grip is of vital importance to climbing robots, for being able to sustain the hold for

a long period of time, as well as provide stability as the other parts of the robot perform their operations.

The project as a whole was a success and a great experience. I have gained a new hobby as well as a whole new set of skills as a result of this project. The robot performed nearly up to its full expectations, and the idea was clearly shown to be an effective method of scaling vertical surfaces. The limitations of this design are easily overcome with more time and money, and future designers could adopt the lessons learned from this effort to build a new generation of capable and robust tree-climbing robots.

10 References and Acknowledgements

There truly are no literature references to list for this project, as not a single book or document was referred to for ideas or direct material on the mechanical design and implementation.   The following website was used to obtain source code used in programming 2-Inch Worm, also listed below is a previous IMDL project referred to for other source code ideas:

BD Micro	http://www.bdmicro.com/

Jeff Panos	"Gimp"	IMDL, Spring 2004

The author would also like to thank fellow IMDL student *Steve Corbett* for his vital assistance in the creation of any aluminum part of the robot. He cut out the feet, welded the pincer mounting plates, and created set screw holes in the pincers themselves to attach to the shafts. I also made extensive use of his tools to make other parts on my own.

Without his assistance this project would have been an embarrassing failure, so a warm thank you is extended to Steve for his willingness to lend a hand.

11 Appendices – Source Code

```
MAIN.C -

// Chris Conger
// EEL5666C - Intelligent Machine Design Lab
// Pentapede source code

#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include <inttypes.h>

#include "adc.h"
#include "i2c.h"
#include "srf08.h"
#include "mono.h"

volatile uint16_t ms_count;
volatile uint16_t altitude,start_height;

/*************************** LCD FUNCTIONS ****************************
 */
void init_display(){

}


void send_to_LCD(char dummy[32]){
        int a;
        for(a=0; dummy[a]!='\0'; a++){
                PORTA = dummy[a];                       // send data 1 character at a time
        }
}


void command(){
        PORTC = 0x20;                                                   // port C = 0x10xxxx
        PORTC = 0x00;                                                   // port C = 0x00xxxx
}


void data(){
        PORTC = 0xA0;                                                   // port C = 1x10xxxx
        PORTC = 0x80;                                                   // port C = 1x00xxxx
}


/*************************** MOTOR FUNCTIONS ***************************
 */
 void motor_init(){
        ICR1 = 0x0271; // gives 20 ms wavelength.  (SETS TOP value FOR MODE 8)
        ICR3 = 0x0271; // gives 20 ms wavelength.  (SETS TOP value FOR MODE 8)

        TCCR1A = 0xFC;         // enables OC1A, OC1B, and OC1C for set on upcount
                                                                // and    clear   on
downcount. also, sets part of mode 8.
        TCCR3A = 0xFC;         // enables OC3A, OC3B, and OC3C for set on upcount and
                                                                // clear on downcount.
also, sets part of mode 8.

        TCCR1B = 0x14;         //sets part of mode 8, and sets prescaler to 256.
        TCCR3B = 0x14;         //sets part of mode 8, and sets prescaler to 256.

        TCNT1 = 0x0000;        //sets TCNT to zero just in case it isn't already.
```

```
        TCNT3 = 0x0000;         //sets TCNT to zero just in case it isn't already.

        DDRB = 0xE0;                    //sets ports to outputs.(OC1A ,B, C)
        DDRE = 0x38;                    //sets ports to outputs.(OC3A, B, C)
 }

void all_pincers(int oc, int time){
        if (oc == OPEN){
                MPNCL = POPN;
                MPNCM = POPN;
                MPNCU = POPN;
        } else if (oc == CLOSED){
                MPNCL = PCLS;
                MPNCM = PCLS;
                MPNCU = PCLS;
        }
        sleep_for(time);
}


void pincer(int oc, int pinc, int time){
        if (pinc == LOW){
                if (oc == OPEN) MPNCL = POPN;
                else if (oc == CLOSED) MPNCL = PCLS;
        } else if (pinc == MID){
                if (oc == OPEN) MPNCM = POPN;
                else if (oc == CLOSED) MPNCM = PCLS;
        } else if (pinc == UPR){
                if (oc == OPEN) MPNCU = POPN;
                else if (oc == CLOSED) MPNCU = PCLS;
        }
        sleep_for(time);
}


void all_segments(int dir, int time){
        if (dir == UP){
                MBODL = SUP;
                MBODM = SUP;
                MBODU = SUP;
        } else if (dir == DOWN) {
                MBODL = SDWN;
                MBODM = SDWN;
                MBODU = SDWN;
        }
        sleep_for(time);
}


void segment(int dir, int seg, int time){
        if (dir == UP){
                if (seg == LOW) MBODL = SUP;
                else if (seg == MID)  MBODM = SUP;
                else if (seg == UPR) MBODU = SUP;
        } else if (dir == DOWN) {
                if (seg == LOW) MBODL = SDWN;
                else if (seg == MID) MBODM = SDWN;
                else if (seg == UPR) MBODU = SDWN;
        }
        sleep_for(time);
}

void climb_up(){                                        // each 2-inch step takes ~9 seconds
                all_segments(DOWN,8192);                // drive robot 2 inches up surface

                pincer(OPEN,LOW,512);                           // adjust lower segment
                segment(UP,LOW,2000);
                pincer(CLOSED,LOW,4000);

                pincer(OPEN,MID,512);                           // adjust middle segment
                segment(UP,MID,2000);
```

```
                 pincer(CLOSED,MID,4000);

                 pincer(OPEN,UPR,512);                             // adjust upper segment
                 segment(UP,UPR,2000);
                 pincer(CLOSED,UPR,4000);
}

void climb_down(){                                  // each 2-inch step takes ~9 seconds
                 all_segments(UP,8192);                                      // descend 2 inches

                 pincer(OPEN,UPR,512);                            // adjust upper segment
                 segment(DOWN,UPR,1536);
                 pincer(CLOSED,UPR,1536);

                 pincer(OPEN,MID,512);                            // adjust middle segment
                 segment(DOWN,MID,1536);
                 pincer(CLOSED,MID,1536);

                 pincer(OPEN,LOW,512);                            // adjust lower segment
                 segment(DOWN,LOW,1536);
                 pincer(CLOSED,LOW,1536);
}

/*********************** SENSOR FUNCTIONS ***************************
 */
uint16_t read_IR(int sensor_num){
        uint16_t reading = 0;

        if(sensor_num == IRNOSE){
                PORTF = 0x80;
        // Port F, bit 7 (MSB)
        } else if (sensor_num == IRLFRONT){
                PORTF = 0x40;
        // Port F, bit 6
        } else if (sensor_num == IRRFRONT){
                PORTF = 0x20;
        // Port F, bit 5
        }
        return reading;
}


uint16_t read_sonar(){
        uint16_t reading = 0;
        uint16_t range = 0;
        uint16_t read_array[17];
        int8_t a,b;

        b = srf08_ping(0x00, RANGE_IN);              // ping and pause
        sleep_for(8192);

        for(a=0; a<17; a++){                         // read all data registers
                b = srf08_range(0x70, a, &reading);
                if (b == 0) read_array[a] = reading;
                else  read_array[a] = 0xff;
                b = 0xff;
        }
        range = read_array[0];
        return range;
}


uint16_t ok_to_climb(){
        uint16_t OK = 0;
        uint16_t height,ir_up,ir_top,ir_bottom;

        ir_up = 0;                                            // take all IR readings
        ir_top = 0;
        ir_bottom = 0;
```

```
        height = read_sonar();        // check height, be sure its higher than last
reading
        if(height <= altitude) OK = 1;
        altitude = height;

        return OK;     // return '0' if OK to continue, return 1-3 if problem encountered
}


/********************* MISCELLANEOUS FUNCTIONS ************************
 *
 void init_timer(){                  // taken from example code at www.bdmicro.com
        TIFR |= BV(OCIE0)|BV(TOIE0);
        TIMSK |= BV(OCIE0);
        TIMSK &= ~BV(TOIE0);
        ASSR |= BV(AS0);
        TCNT0 = 0;
        OCR0 = 32;
        TCCR0 = BV(WGM01)|BV(CS00);
        while(ASSR & 0x07);
        TIFR |= BV(OCIE0)|BV(TOIE0);
 }*/


 void sleep_for(uint16_t ms){       // taken from example code at www.bdmicro.com
        /* TCNT0 = 0;
         ms_count = 0;
         while(ms_count != ms);*/
         long int t;
         for(t=0;t<(((long int)ms)*1024);t++)
                 ;
 }


 void setup(){


        //start_height = read_sonar();           // get initial height reading
        //altitude = start_height;

        all_segments(UP,2048);              // initialize with all segments up, pincers
open
        all_pincers(OPEN,15000);

        pincer(CLOSED,LOW,8000);
        pincer(CLOSED,MID,8000);
        pincer(CLOSED,UPR,20000);
 }



 SIGNAL(SIG_OVERFLOW3){
         TCNT3 = 0;
         TCCR3C |= BV(FOC3A);
}


/************************ MAIN FUNCTION **************************
 */
int main(void)
{
        uint8_t READY;
        uint16_t all_clear;
        DDRC = 0xff; DDRA = 0xff; DDRF = 0x00;
        READY = 0;

        sei(); /*init_timer();*/ adc_init(); /*init_display();*/ motor_init();

        TWSR &= ~0x03;                      // set I2C bit rate generator to 100 kb/s
        TWBR = 28;
        TWCR |= BV(TWEN);
```

```
        setup();                                // place robot on tree

        all_clear = 0;
        while(all_clear == 0){                   // climb up, until sensors say to
stop
                climb_up();
                //all_clear = ok_to_climb();
        }

        // RECORD MAXIMUM HEIGHT HERE

        while(altitude > start_height){
                climb_down();
                altitude = read_sonar();
        }

        while(1)                                 // don't ever reach the end of main()
                ;
}
```

ADC.C –   see BDMicro.com
SRF08.C -   see BDMicro.com
I2C.C -   see BDMicro.com