Date: *4/25/08*

Student Name: Kim Wright

TA : Mike Pridgen

Adam Barnett

Instructors: Dr. A. Antonio Arroyo

Dr. Eric M. Schwartz

University of Florida

Department of Electrical and Computer Engineering

EEL 5666

Intelligent Machines Design Laboratory

RoboPost : Final Design Report

(Version 1.0)

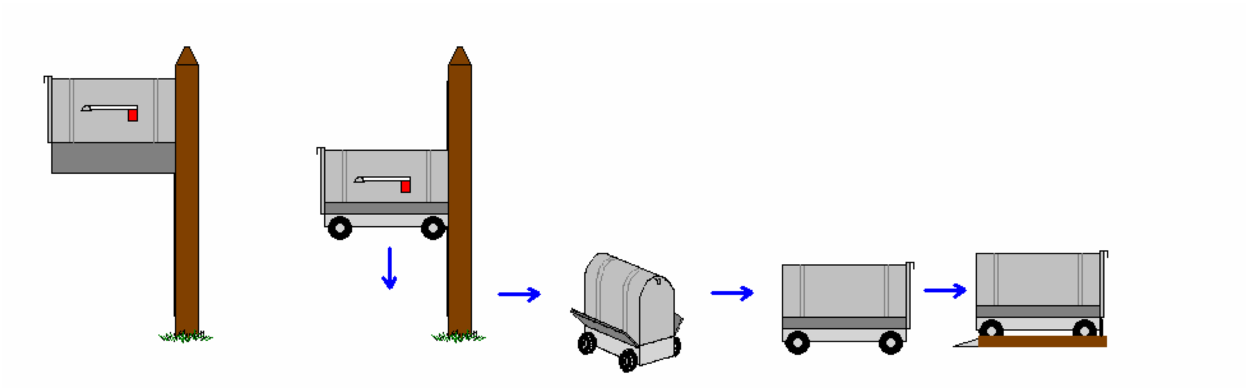
Table of Contents

3. Abstract

Snail mail is finally receiving an upgrade with the development of RoboPost the robotic mailbox. A robot was designed to appear to be a standard mailbox on a post. After the mail was delivered, the robot automatically lowered itself from the post. An invisible fencing system was modified to guide the robot along a predetermined path to a base near the doorstep. Once reaching the base, the mail could easily be retrieved. The original concept required the robot to automatically return to the post at the end of the driveway. A prototype was constructed which displayed the ability to autonomously lower itself from the post after the mail was delivered. The prototype did not include the behavior for returning to the base, but was mechanically capable of achieving this goal.

4. Executive Summary

Snail mail is finally receiving an upgrade, thanks to the development of RoboPost the robotic mailbox. This robot was designed to appear to be a standard mailbox on a post, with one big difference. After the mail has been delivered, the mailbox lowers itself from the post, and drives down a pre-determined path to a base near the homeowner's doorstep. From here, the mail can easily be retrieved, and outgoing mail can be placed in the mailbox. The robot would then return to its post at the end of the driveway.



The RoboPost concept

The RoboPost system required several major components: the mobile robotic platform, a modified invisible fence pet containment system, and a mechanical post. The mechanical post did not require a separate power source, and relied on the mobile robotic platform for power and commands. The invisible fence system consisted of a transmitter base positioned at the house, a buried loop of wire, and a sensor mounted on the robot.

When on the mechanical post, the robot appeared to be a standard robot. This was accomplished by restricting the robotic platform to the same size as the mailbox's footprint, and concealing the drivetrain components with wooden sides while sitting on the post.

A microprocessor onboard the robotic platform provided all of the intelligence for the system. Three bump sensors and the invisible fence dog collar receiver provided information to the microprocessor, which responded to sensor stimuli by issuing commands to its drive system and servos. The microprocessor selected for this task was the Mavric IIB, programmed using C++ code through AVR Studio.

The RoboPost prototype successfully displayed the ability to control the mechanical post to lower itself to the ground under autonomous control. Development is still in progress, and is scheduled to be complete by May 1, 2008. By this date, the prototype should have demonstrated the ability to automatically lower itself from the post after the mail is delivered and follow a per-determined path to its base.

Future plans for RoboPost would include the ability to return to the post, as well as detect the mailbox flag position. Such developments could be accomplished with little mechanical modification to the current prototype.

5. Introduction

The United States Postal Service has been operating since the 18th century, significantly improving communication between citizens. Since then, modern technology has increased productivity and prompted humanity to be much more rushed since the introduction of the postal service. As a result, modern society has begun to avoid traditional postal services, and gravitate towards electronic mail as an alternative way to communicate. Trips to the mailbox can be time-consuming for those with long driveways, and difficult for people with mobility-related disabilities. Most people hesitate to walk outside on cold days, even if it is a short distance. E-mail is a much faster way to communicate, but it has not been able to completely replace the postal services. Packages, official signature forms, and other business materials must be delivered through the postal service, requiring that the recipients participate in this somewhat dated network. Most importantly, a trip to the mailbox requires time: an asset which has become increasingly valuable.

Noting that certain packages and documents cannot be sent in electronic format, it is clear that the postal service still serves a necessary purpose, and cannot yet be eliminated. However, it is equally evident that the general public is not satisfied with the process of using this system. RoboPost the robotic mailbox is being developed to help make using the postal service more convenient for the everyday person. It will look just like a standard mailbox, and will wait on a post at the end of a driveway for the mail. Postal service employees will not even realize it is a robot: they simply place the mail in the mailbox, just as they would any other mailbox. However, after the mail has been delivered RoboPost will lower itself from its post and navigate a predetermined route to its charging base at the homeowner's doorstep. Once RoboPost is on its base, delivered mail can be easily removed. (Please see figure 1.)

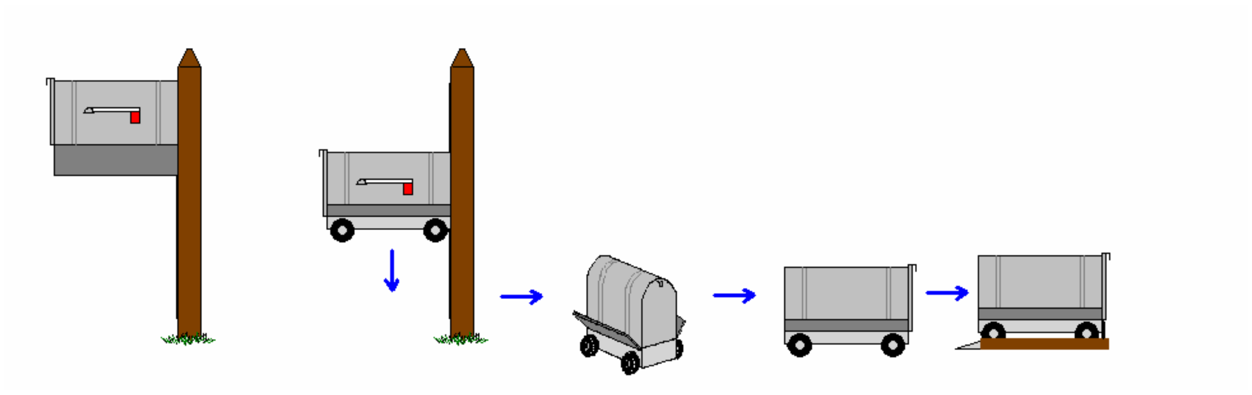


Figure 1. *RoboPost concept*

The original RoboPost concept included plans for the robot to automatically recharge its batteries, and return to its position on the post automatically. These objectives were not met with the current version of RoboPost, but the basic mechanical aspects of the RoboPost design are capable of achieving this behavior.

6. Integrated System

The RoboPost system required several major components: the mobile robotic platform, a modified invisible fence pet containment system, and a mechanical post. The mechanical post did not require a separate power source, and relied on the mobile robotic platform for power and commands. The invisible fence system consisted of a transmitter base positioned at the house, a buried loop of wire, and a sensor mounted on the robot.

A microprocessor onboard the robotic platform provided all of the intelligence for the system. Three bump sensors and the dog collar receiver provided information to the microprocessor, which responded to sensor stimuli by issuing commands to its drive system and servos. The microprocessor selected for this task was the Mavric IIB, programmed using C++ code through AVR Studio.

RoboPost does not require any special human interaction to deliver the mail. It was intended to be a robot in disguise, and perform its task without supervision. Therefore, operating RoboPost is as simple as delivering the mail. Simply walk up to the robot while it is at the top of its post, open the mailbox door, place a letter inside, close the door, and walk away. If you are the homeowner receiving the mail, then all you must do to operate RoboPost is open your front door after the mail has been delivered. The robot will be waiting there with the mail inside.

At this point, the homeowner would also be responsible for returning the robot to its post and manually applying the power to raise the lift platform to the top of the post. This step could be automated given more time, and would certainly be automatic in any commercial products. In fact, the major mechanical challenges for accomplishing this behavior have already been achieved and displayed by the prototype. Regardless of this, RoboPost is currently limited to delivering the mail from the post to a charging base, and requires the user to return it to its starting position.

7. Mobile Platform

7.1 Platform Requirements

The RoboPost system consists of two major components: the mobile robotic mailbox, and the mechanical post. The mechanical post could not operate without the robotic mailbox, and was a purely mechanical device. These components had different objectives, but had to interact to achieve a common goal.

Mobile Robotic Platform:

The mobile robotic platform had several major mechanical requirements. First, the platform had to be capable of driving and steering using two powered wheels. It needed to be stable and capable of supporting a standard mailbox, and also needed to easily permit modifications. In addition, the robot had to be capable of securing itself on the mechanical post.

Many of the requirements for the mobile robotic platform were dictated by the mechanical post mechanism. The post could not lift a heavy load, therefore the robot had to be as light as possible. In addition, the mobile platform needed to appear to be a standard mailbox when docked on the post. To achieve this, the mobile platform's footprint could not be any bigger than a mailbox. Also, the electrical and drive train components had to remain hidden while the robot was docked on the post.

Mechanical Post:

Although it lacked artificial intelligence, the mechanical post was one of the most time-consuming and demanding aspects of the project. The post had to be capable of supporting the robotic platform as a cantilever beam. This was complicated by the requirement to raise and lower the robot in a smooth and controlled fashion. Supporting a standard mailbox on a post is a simple task: just nail it to the side of a post. Supporting RoboPost is much more complicated: the robot cannot be permanently attached, and is heavier than a standard mailbox.

Additional post requirements were dictated by need to plug in to the robotic platform. The post did not have its own battery source or motor control, and therefore had to include reliable electrical contacts to connect to the robotic platform's power system. The robot placed an additional demand on the post by requiring a sensor to detect when the robot had reached the ground. Therefore, a bump switch on the bottom of the lift mechanism also had to connect to the robot when it was connected to the post mechanism.

The weight of the robotic platform causes a large moment about the attachment point on the post, requiring the track mechanism to withstand significant lateral forces while in motion and static. In addition, by supporting the robotic mailbox as a cantilever beam, a moment was also generated about the base of the post. Therefore, the post needed to be able to withstand this moment without tipping over.

Because the robot was intended to stay at the top of the post for an extended period of time, the post mechanism needed to automatically hold the robot one position at any point on the mechanical post's track. This system needed to operate independent of the robot's control system to prevent damage in case of programming error or loss of power.

7.2 Platform Description

<u>Mobile Robotic Platform:</u>

The assembled robotic platform is shown in figure 2. The mailbox was purchased from Lowes, and is a cheap plastic mailbox made by Rubbermaid. It was lighter than the metal versions, and was conveniently also the cheapest. The plastic proved simple to work with, and could easily be drilled or cut.

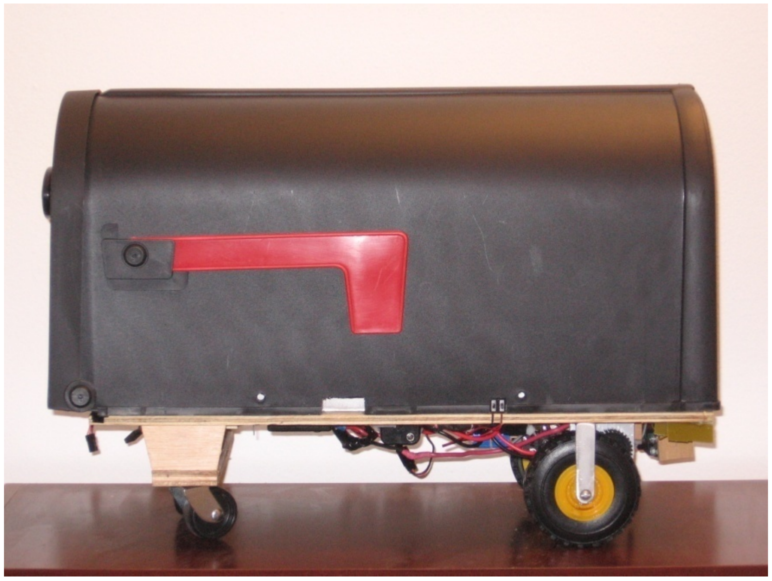


Figure 2. *The assembled mobile robotic platform*

The floor of the mailbox is raised approximately one inch from the bottom of the mailbox sides. Although Rubbermaid likely included this feature to make mounting the mailbox simple, it also turns out to be just enough room for a MAVRIC-IIB, sensors, and related components. Figure 3 shows the robotic platform with the mailbox removed, and the electronics housed underneath the mailbox.

Several materials were considered for constructing the robotic platform: wood, foam and fiberglass composite, and aluminum. Although a foam and fiberglass composite structure would have been the most lightweight option, long curing times and material availability prevented it from being selected for this project. Wood and aluminum were readily available, and were selected as the primary materials. Although not as light as composites, wood and aluminum was simple to work with, and allowed changes to be made easily. For a prototype, this was a critical capability. Changes were made throughout the construction process, as evident by the presence of unoccupied drill holes on the wooden base. Thin Aluminum L-beams provided additional longitudinal rigidity, and also helped to align the mailbox and protect the concealed electronics.

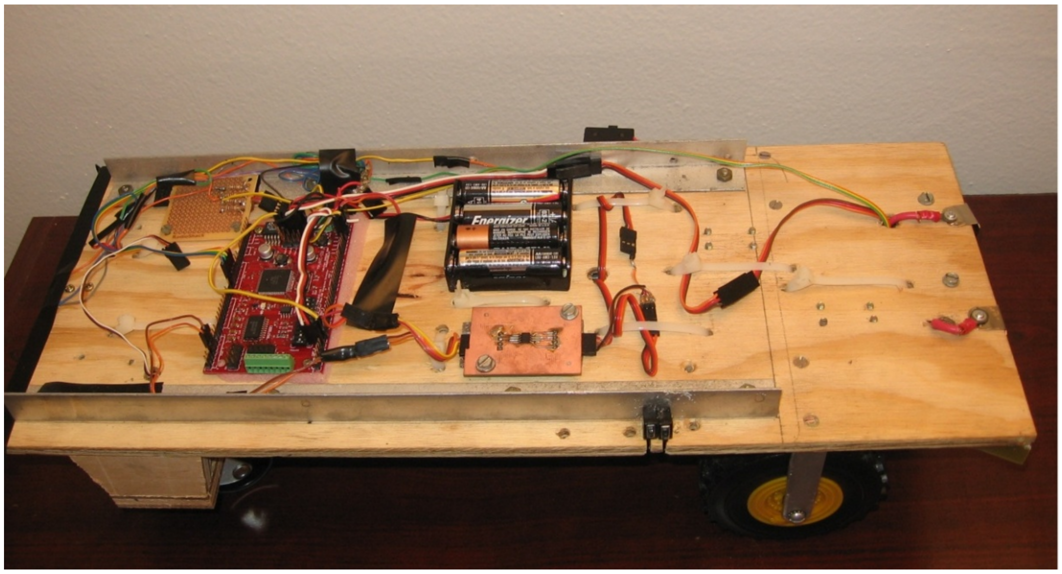


Figure 3. *Robotic platform with mailbox removed*

To enable the robot to appear to be a standard mailbox while on the mechanical post, the robot's wooden platform was cut to the same dimensions as the mailbox's footprint. The drive train components were housed underneath the platform, and were not permitted to protrude outside the footprint of the mailbox. (See figure 4). This simplified the task of hiding the robot's non-mailbox features when on the post.

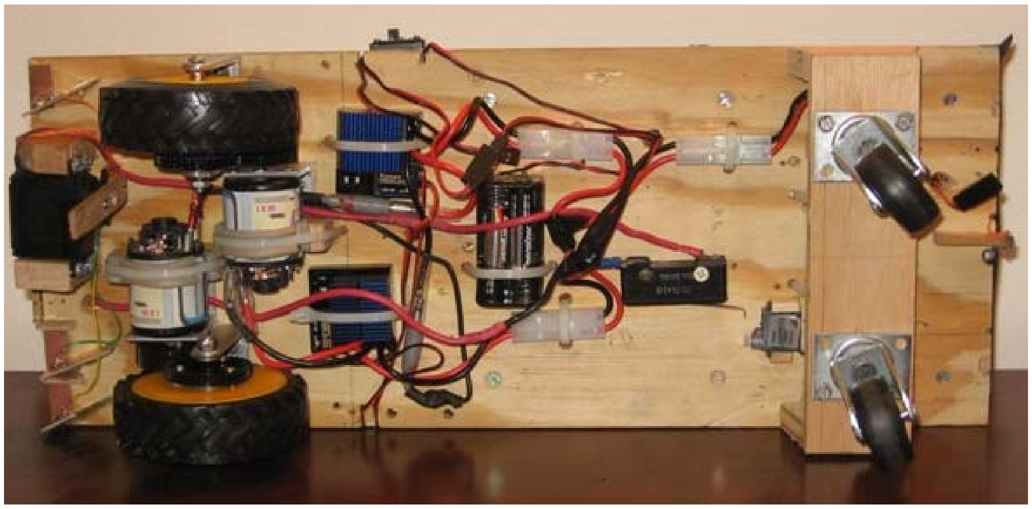


Figure 4. *Underside of the robotic platform*

The robot needed to connect to the mechanical post motor's power connections and bump sensor leads. Two metal contact plates were installed on the back of the robot to provide power to the post motor. (See figure 5). Underneath the robot, to the left and right of the post motor contacts, are three individual copper contacts for the bump sensor. These contacted three wire 'whiskers' on the post, completing the circuit for the bump sensor to sense when the post had lowered the robot to the ground.

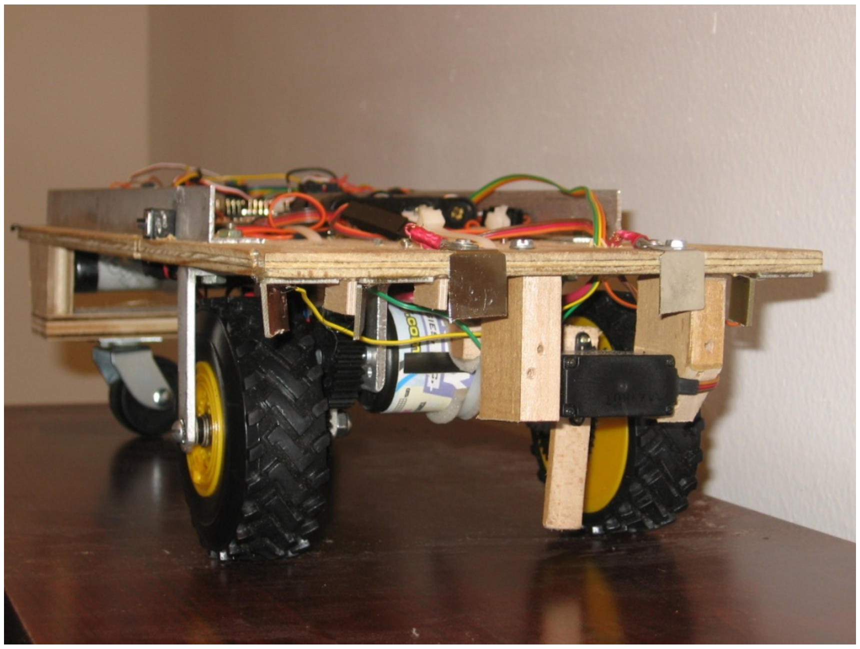


Figure 5. *Metal contacts for post electronics*

<u>Mechanical Post:</u>

Construction of the mechanical post was performed alongside the construction of the robotic platform. The robot relied on the mechanical post, therefore it would not be possible to efficiently program the robot's behaviors without the post. Likewise, the post relied on the robot for power and control.

Because it was meant to stay in one place, lightweight construction was not an important feature for the post, with the exception being the cantilever platform. To transport on demo day and media day, the post would need to be somewhat mobile, and could not be secured staked to the ground. Therefore, a large wooden foot was used as the base for the post, to which upright sections were secured using wood screws and wood glue. Wood was the primary material, with metal components used for the track mechanisms. These metal components were capable of withstanding the moment produced by the cantilever lift platform. The main components of the post included the wooded base, the wooden track support column, the structural wooden walls, the lift platform support structure, the lift platform, the metal track, the moving track mechanism, and the motorized cable and pulley system.

Lifting the robot was accomplished by using a pulley system powered by a car window motor. The motor included a gear head with a worm screw, which prevented the mechanism from moving when the motor was not powered. Worm gears are commonly used for passively securing a cable system, and are used in boat lift assemblies in setup similar to the RoboPost mechanism. The motor is powered by the robot, and has a spool of wire attach to its shaft. The wire used was hobby-grade wire, intended for control-line model aircraft. Such wire is strong, flexible, and does not stretch much under load. These properties made it ideal for the RoboPost cable. The wire was wrapped through a pulley system to reduce the tension in the wire, and lower the demand placed on the motor. Assuming a total robot and lift platform weight of 15 lbs, the tension in the wire was reduced to 2.5 lbs. (See figure 6). Although less demand was placed on the motor at a given instant, the overall work still required to lift the robot remained the same. This enabled the motor to lift the robot without requiring excessive current. Although less demand was placed on the motor at a given instant, the overall work still required to lift the robot remained the same. The cable system reduced the tension in the wire; however more time was required to lift the robot. A slow lifting action was acceptable, given that the robot was not required to complete its task particularly quickly.

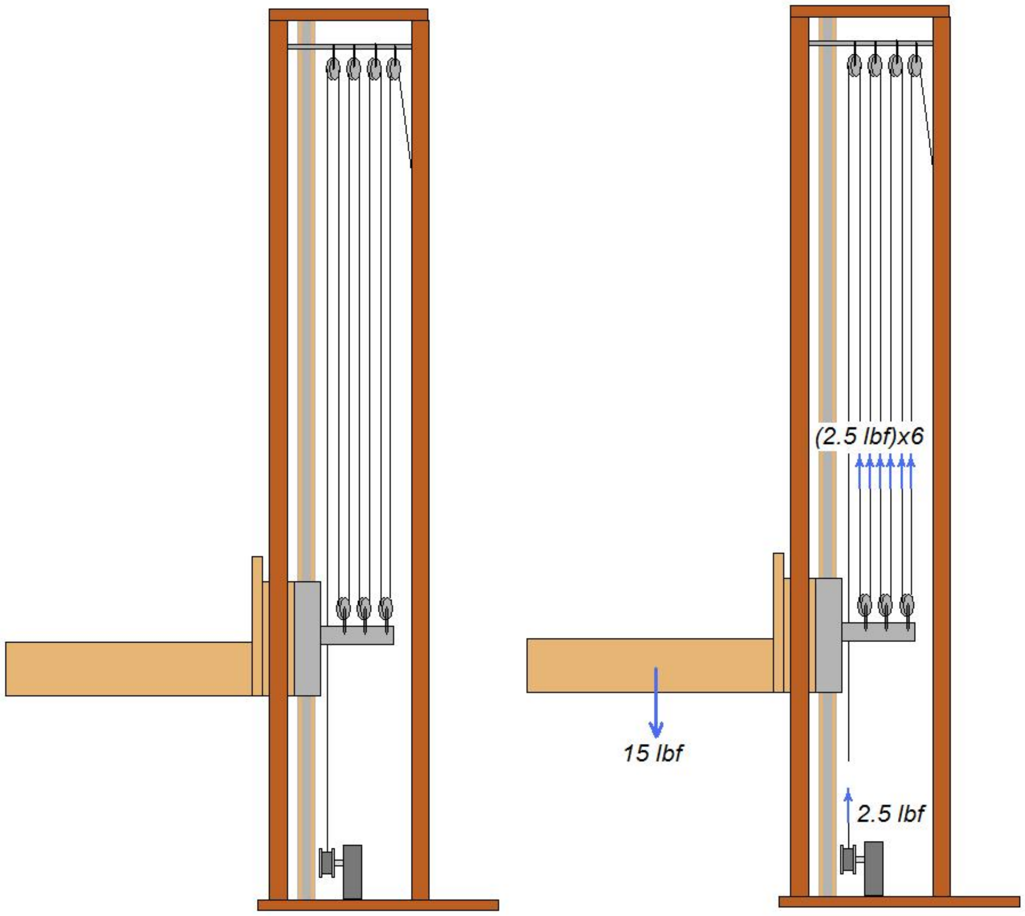


Figure 6. *Diagram of cable system, with force balance in the vertical direction showing tension in the wire.*

The post mechanism could not be fastened to the ground, but it needed to withstand the moment produced by the cantilever lift platform. Although lightweight construction was not of hug importance, the post mechanism had to be light enough to transport up stairs and in a small car. Bricks were available in the robotics lab, and the post was designed to use this resource. The footprint of the base was extended to allow room for the bricks to be stacked, and these provided the force necessary to counteract the moment produced by the lifting platform. (See figure 7).

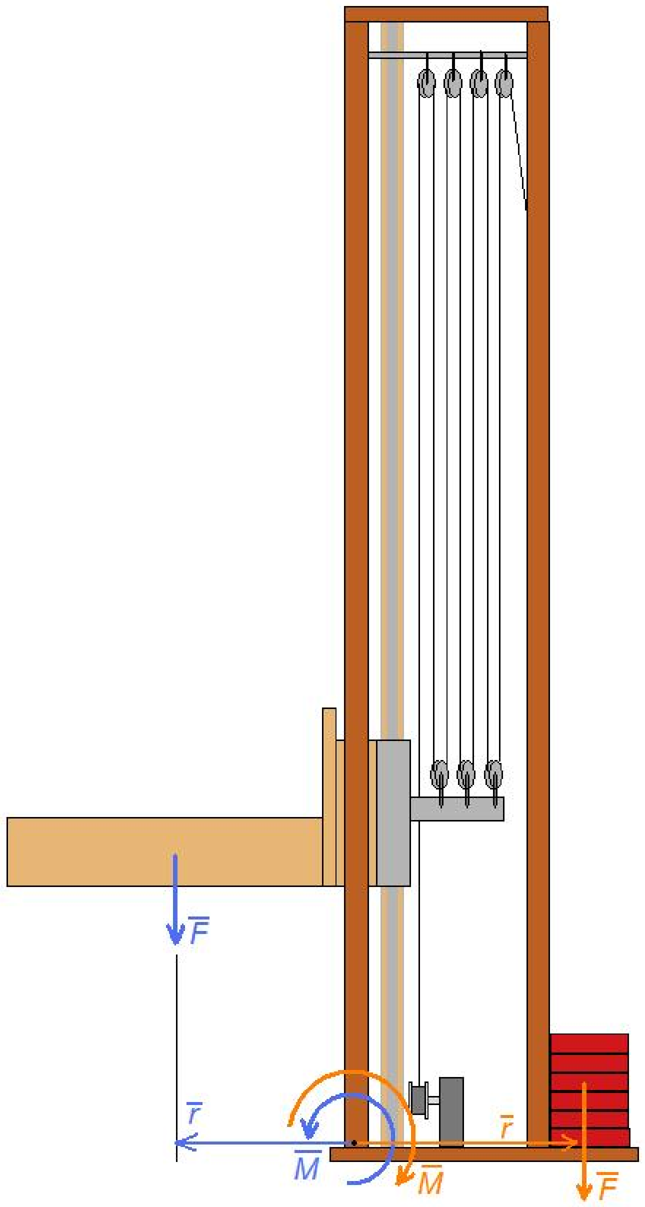


Figure 7. *Post mechanism showing moment balance.*

8 Actuation

8.1 Servo Motors

Switch servo:

The robot was required to use one battery source to drive the main motors along with the post-climbing motor. To activate the post-climbing motor, a switch was installed in the circuit which powered the post-climbing motor. The switch was operated by a servo driven by the Mavric. By using a servo, the 7.2 Volt post-climbing circuit was electrically isolated from the main board.

[Note: a relay was considered for use as opposed to a servo. However, the original design required the current to be able to flow both directions. This, along with the high power requirements, meant that a more expensive relay would be needed. A servo and switch were already available, and were selected for convenience.]

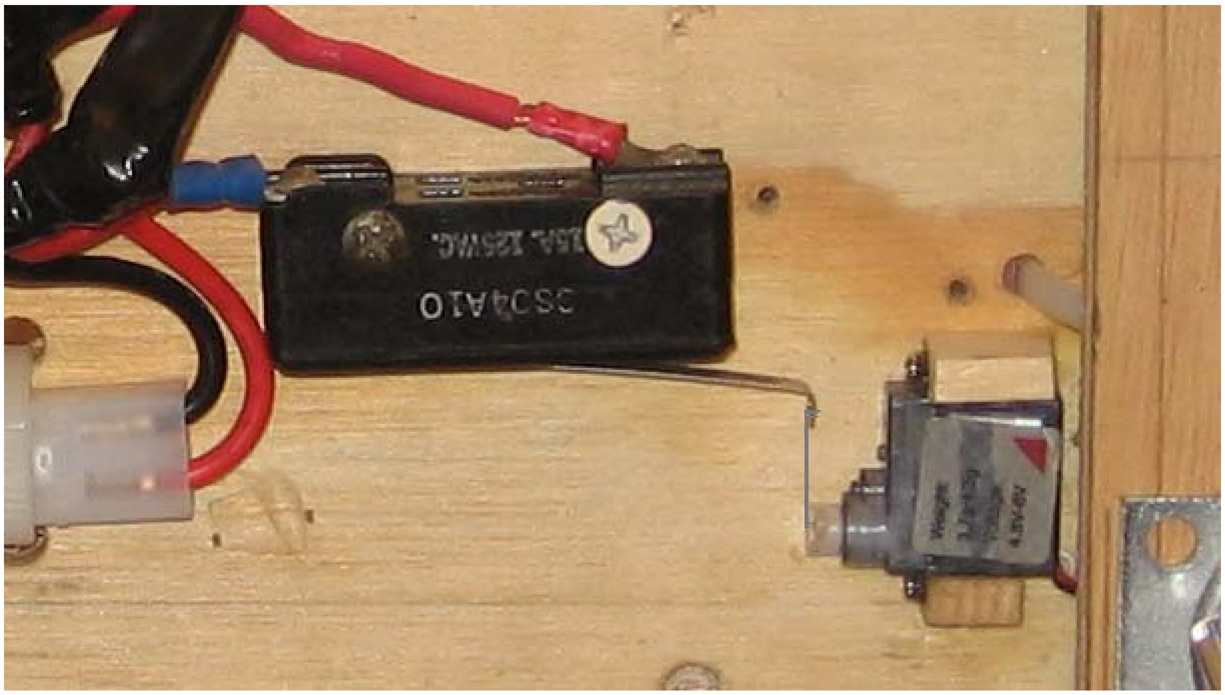


***Figure 8**. The servo and switch mechanism*

The servo was controlled using a PWM signal on port B of the Mavric board. The code to drive the servo was derived from code by Mike Pridgen, and adapted to output the signal on port B. Detailed code is available in the appendix.

On media day, this servo failed to operate. During testing, the switch the servo operated was manually pressed to activate the pole-climbing motor when the Mavric was switched off. In retrospect, this action probably damaged the servo gears. During transport between the lab in Benton Hall and the New Engineering building, the servo arm was likely bumped, leading to the disappointing failure. Although the servo was a convenient choice, this failure proved that it was not the most robust choice. A relay would not be as likely to fail, and would have provided the isolation from the high-power circuit. Therefore, a relay is advised for switching isolated DC circuits whenever possible.

Brake Servo:

When the robot was sitting on the lift platform, a brake was applied by a servo to prevent the robot from rolling off. In addition, the brake held the robot tightly against the electrical contacts on the post mechanism. Very little force was required to prevent the robot from rolling; therefore a wooden arm directly attached to a servo was sufficient for this task. The wooden arm slid behind a bar on the base, securing the robot.

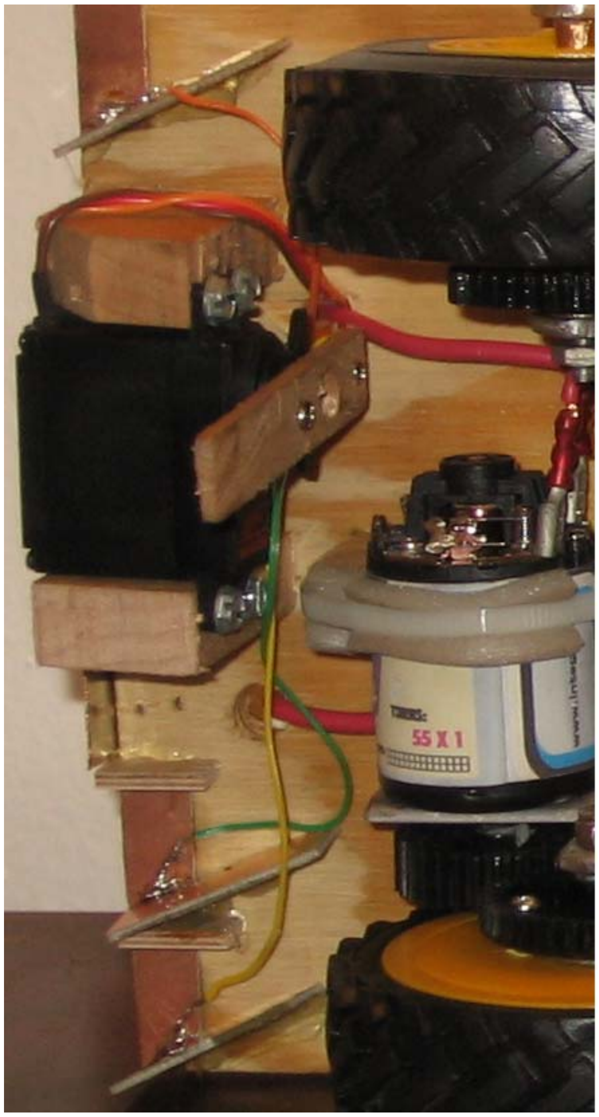


Figure 9. *The brake servo*

By orienting the servo perpendicular to the direction the robot would roll, the force applied to the brake arm was not aligned with the direction of servo arm travel. This meant that the force of the robot attempting to roll would not be able to rotate the brake arm. This system was simple, and reliable.

8.2 Post climbing motor

The motor selected to power the robot up the post was a car window motor, operated at 7.2 V DC. From everyday experience, these motors were known to produce plenty of power for raising and lowering loads in a controlled fashion. In addition, the built-in gear head included a worm gear which would hold the load in position when the motor was switched off.

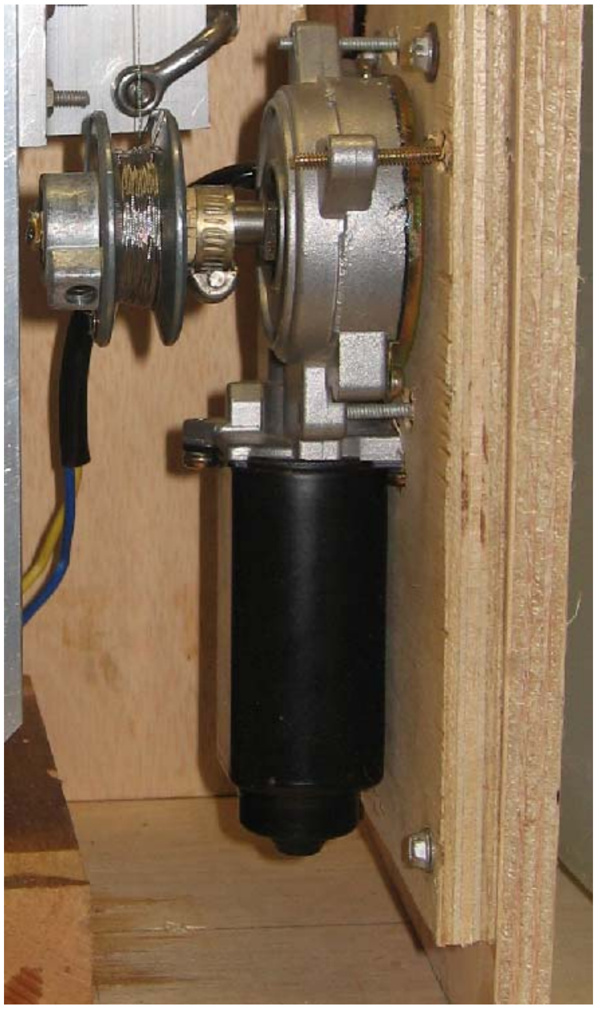


Figure 10. *The post-climbing motor*

This motor was purchased at Skycraft Parts in Orlando, and therefore did not come with any documentation. However, the motor was tested in Skycraft and the lab to draw 3 A of current when operating under load. Although it would normally use 12 V in a car, the motor functioned properly at lower voltages.

8.3 Drive train components

The drive train for RoboPost was originally intended to power the robot over difficult terrain. The original concept called for two powerful Integy 55t R/C rock climbing motors to drive and steer the robot, while two large castor wheels in the front provided additional support. However, the large castor wheels were unreliable and were replaced with smaller castor wheels. These wheels were not well-suited for off-road use, and limited the robot to brick or concrete surfaces. Therefore, the Integy motors were excessive for the final design. Each motor was controlled by a standard R/C speed controller, the Tazer 55t. To protect the Mavric from the high-power drive system, an opto-isolator from Analog Devices was used between the speed controllers and the Mavric board.

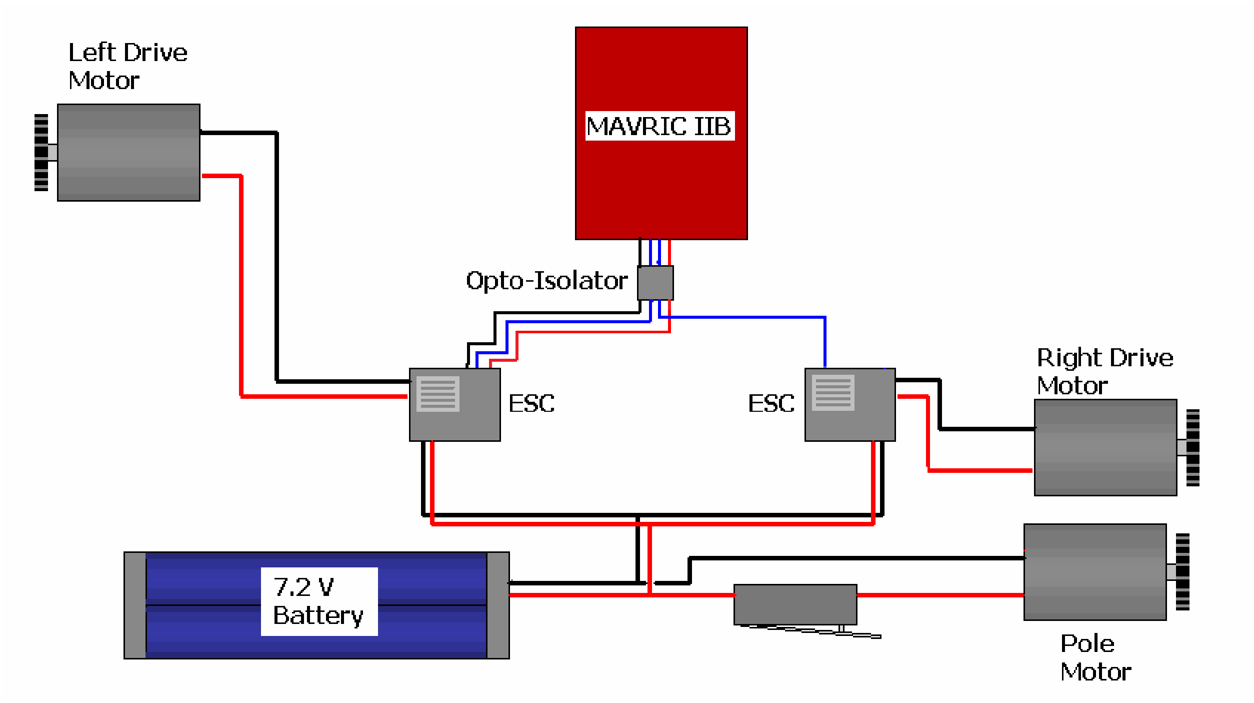


Figure 11. *The 7.2 V drive train and pole-climbing power system*

8.3.1 Integy 55t motors

After some research, the Integy 55t was selected for the drive motors. These motors are favored by the R/C rock climbing community, and can provide high torque for off-road environments. Both motors were purchased at Colonial Photo and Hobby for about $35 total. Each motor was attached to a wheel using a set of strong plastic gears found at Skycraft.

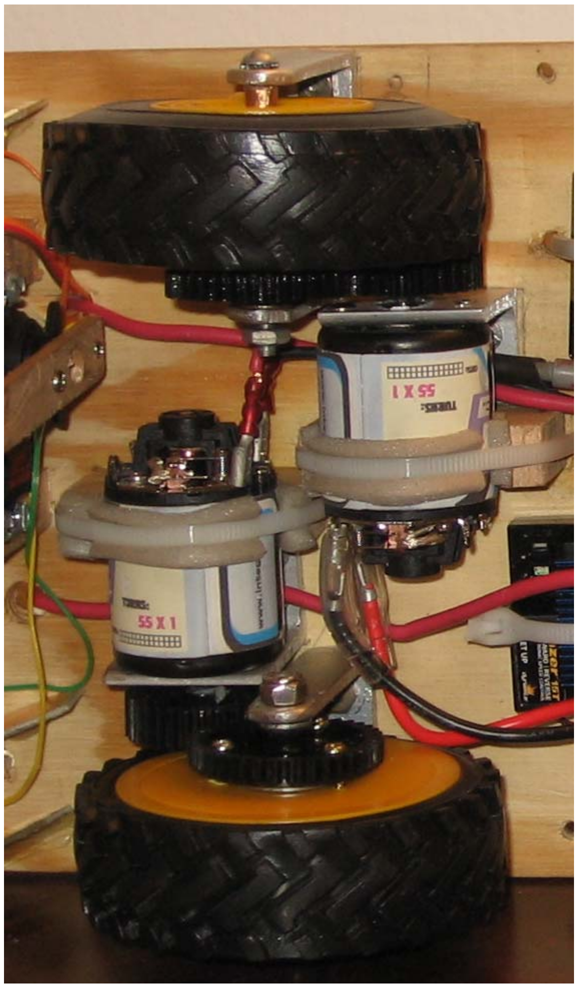


Figure 12. *The Integy 55t motors attached to the rear drive wheels*

These motors were powerful, reliable, and simple to work with. Although they were overpowered for the final design, they were very capable and proved to be a good choice for the robotic platform.

8.3.2 Tazer 15T electronic speed control

Control of the Integy motors was dependent upon two electronic speed controllers purchased at Hobby Town USA in Gainesville. Because these components often cost more than $50 per piece, cost was a primary factor. In addition, each controller needed to be capable of handling a brushed Integy motor with 55 turns, and drive the motor in forward or reverse.

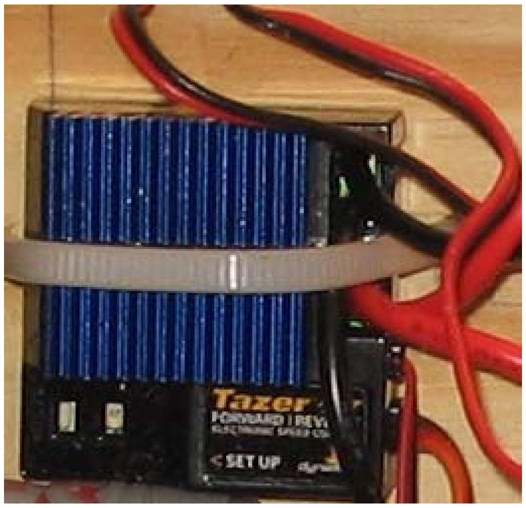


Figure 13. *One of the Tazer 15T speed controllers*

The Tazer 15T was a low-cost solution capable of meeting all the requirements dictated by the motor. The Tazer 15T speed controller cost $30, while the next cheapest option would have cost over $45. Because two systems were necessary, the Tazer was selected. These speed controllers included documentation, however specifications were not trusted. (The documentation claimed that the speed controlled could handle over 700 Amps.)

Because they were designed for radio control hobby use, each speed controller had a cable which was intended to supply 5V to power a radio receiver. This cable also had a signal cable which accepted a PWM signal from the receiver, the same type of signal used to drive a servo. The controller translated the signal, and provided power from the battery pack to the motor proportionally.

8.3.3 Analog Devices iCoupler Opto-isolator

Protecting the Mavric from the drive train circuit was achieved using a two channel opto-isolator from Analog Devices. (Part ADUM1200CRZ). Analog Devices provided two of these as free samples, and even paid for rapid shipping.

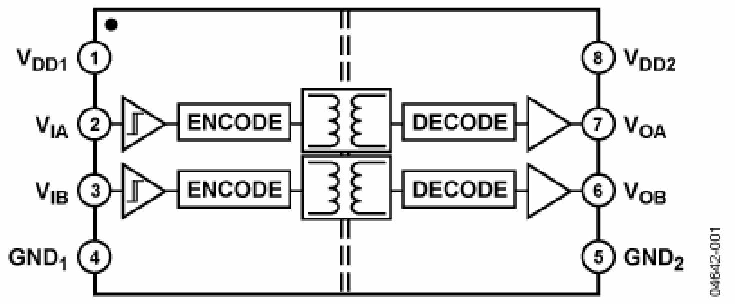


Figure 13. *Opto-isolator schematic. Picture from analog devices datasheet, available at: http://www.analog.com/en/prod/0,2877,ADuM1200,00.html*

The opto-isolator required 5V power and ground to be supplied on both sides from the isolated circuits. The left side of the isolator (the signal encode side) was powered by the Mavric. The right side of the isolator (the signal decode side) was powered by one of the electronic speed controllers. The signal sources were connected directly to the PWM output from Port E on the Mavric, and the signal decode connections provided the PWM signals to the Tazer 15T speed controllers. The datasheet for the opto-isolator can be found in the appendix.

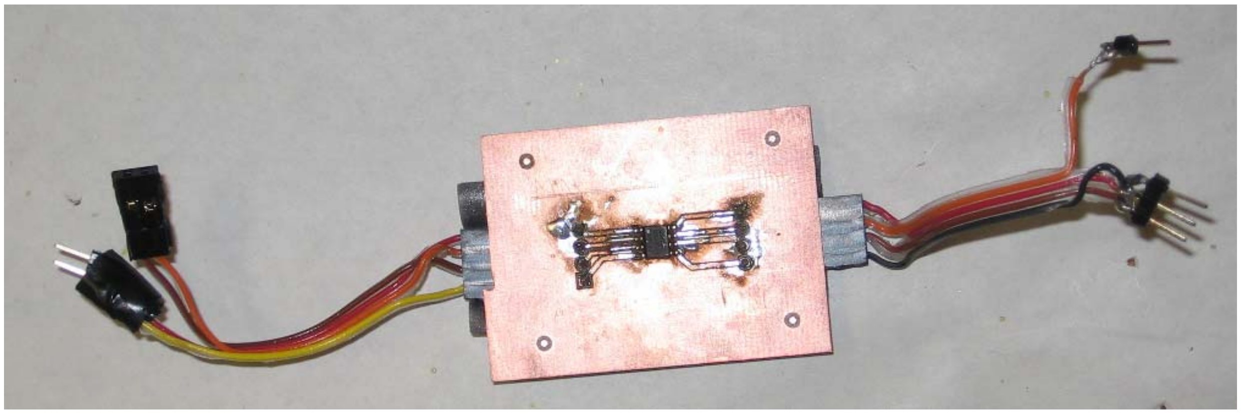


Figure 14. *The opto-isolator circuit with cables attached.*

The opto-isolator concept was simple to understand, but provided some challenges to implement. Analog devices did not provide a footprint for the device, and also only had surface-mount isolators available. This meant that a custom footprint had to be designed in order to create a board on the T-tech machine. Adam Barnett had plenty of experience using the T-tech machine, but had never before created a custom footprint. He provided a great deal of help, and tackled the task of learning to create a custom footprint. With his help and Mike Pridgen's surface-mount soldering skills, the opto-isolator was connected to a circuit board. Custom cables were built to connect the isolator to the Mavric and the two Tazer 15T controllers.

9. Sensors

The original RoboPost concept called for sonar, bump sensors, and Hall Effect sensors to follow buried magnets. However, the idea of using Hall Effect sensors to follow a 'bread crumb' trail of magnets was eliminated due to concern uncertainty in the Hall Effect sensors' abilities to detect the magnets. An invisible fence system was selected instead, because it included a sensor which was observed to easily detect a boundary.

Due to time constraints, sonar was not implemented on RoboPost. Because the robot was following a pre-determined path, obstacle avoidance was not a primary concern. Such a behavior would have been an extra feature, and was not necessary for the robot to accomplish its task.

9.1 Invisible Fence System

The dog fence system consists of a buried wire, a transmitter mounted to the house, and a small receiver originally made to be placed on a dog collar. The transmitter emits a radio signal which is broadcast through the buried wire which acts as a large loop antenna. When the receiver is near the wire, it detects the strength of the radio signal. A beep is emitted when the receiver is in the 'warning zone.' When the receiver is moved closer to the wire, it enters the 'shock zone' and delivers an electric shock. In this way, the receiver can be used to detect the buried wire, and determine proximity based on the beeps and shocks delivered by the receiver.

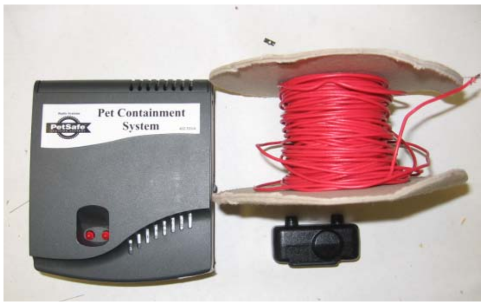


Figure 15. *The invisible fence components*

There were two possible layouts considered which would permit RoboPost to use the invisible fencing system. (See Figure 16.) In Figure 16, two layouts (A and B) are shown. The black box indicated the transmitter position, and the red lines are the buried wire. The post at the end of the driveway is shown at the bottom of each layout, and the charging base is near the transmitter at the top. In layout A, the robot follows the path by positioning itself relative to one side of the loop. It would attempt to stay at the boundary between the warning and shock zones, and continuously alternate between these zones. In layout B, the robot would attempt to stay between both sides of the wire loop, and avoid the shock zone.

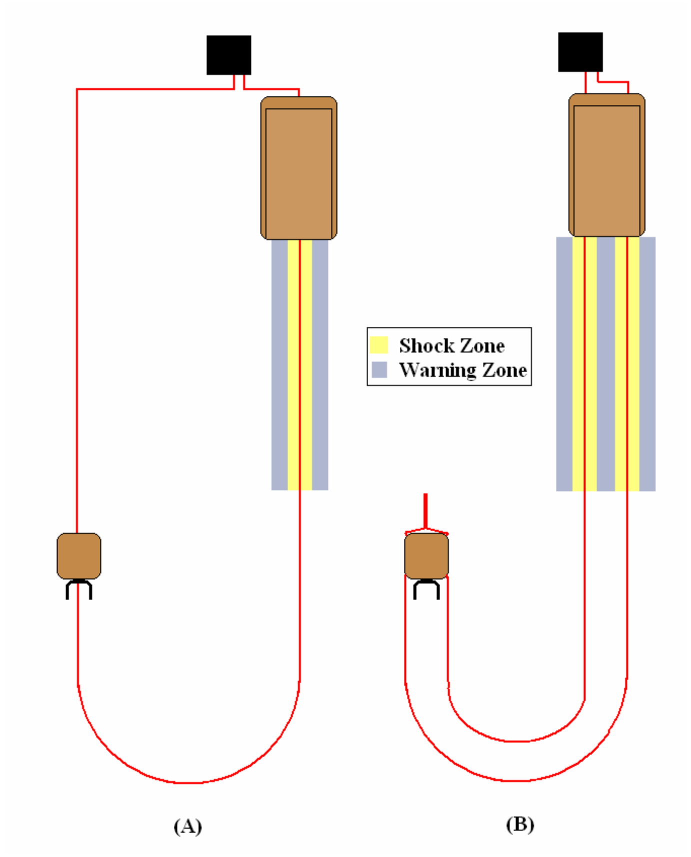


Figure 16. *Possible layouts for invisible fence system*

The layout shown in (A) was selected, with one modification: the robot was unable to detect the warning zone, and would instead use a 'wall-follower' technique to follow the shock zone. Two radio receivers would have prevented the need for this behavior, but was prohibitively expensive.

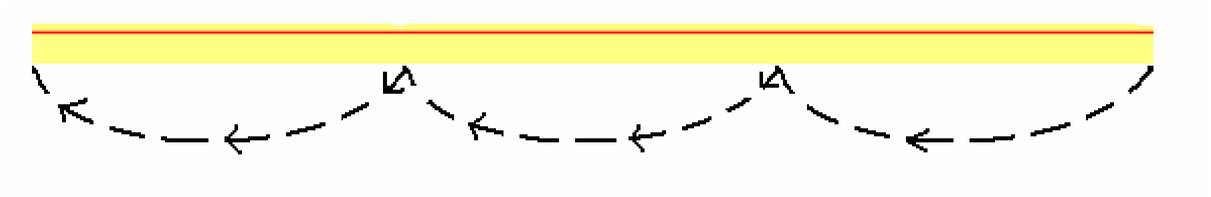


Figure 17. *Wall-following technique*

Detecting the shock from the dog collar sensor was the most challenging part of this program, and required the extensive help of an electrical engineer, Subrat Nayak, to be completed. Subrat is very capable building circuits, and kindly designed a circuit which was able to sense the high-voltage pulses emitted from the dog collar receiver. His circuit supplied a signal to the Mavric when the shocks were received, and he developed an interrupt-based code to interface with the steering functions to guide the robot. This code is provided in the appendix.

9.2 Bump Sensors

Three bump sensors were used by RoboPost: one on the front of the robot to detect the charging base, one in the mailbox door to detect mail delivery, and a third on the post mechanism to detect when the robot had reached the bottom of the post.

Each bump sensor was connected to ground, power, and signal pins on the Mavric. A 10 kΩ resistor was used for each bump switch between the power and the signal pins. The bump switches for the charging base and mailbox door were designed to be normally open when not bumped. Therefore, the Mavric board would detect a high signal when the sensor was not bumped, and a low signal when the sensor was bumped. The pole position bump sensor was normally closed, therefore the board would sense a high signal when the switch was bumped.

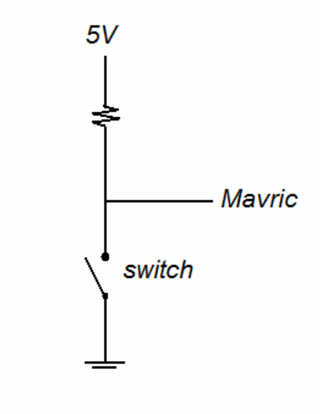


Figure 18. *Bump sensor schematic*

10. Behaviors

RoboPost is a mechanically complicated robot with simple behaviors. Its primary behaviors are:

- Detect mail delivery
- Move down post
- Leave post
- Follow buried cable to base

Originally, the robot was designed to return to its base, and climb the post to wait for the next mail delivery. The mechanical post is capable of lifting the robot, and this behavior could probably be implemented with additional time. The only expected mechanical changes would be the inclusion of 'guide doors' on the entrance to the lifting platform. These doors would help to funnel the robot into the platform, and could easily be attached to the existing mechanism.

<u>Detect Mail Delivery:</u>

The bump sensor inside RoboPost's mailbox door is responsible for detecting mail delivery. The program logic is outlined below:

- While the mailbox door bump switch signal is low, the switch is closed and the mailbox door is closed.
    - Do nothing while the door is closed, just wait for the mail.
- When the mailbox door bump switch signal is high, the mailbox door is open.
    - Enter a delay loop to permit the user enough time to place mail in the box.
- Go to the function for lowering robot from post.

This assumes that the only reason the door is opened is to deliver the mail. Upgrades to this would include a system for sensing the flag position: if the door is opened and the flag is then raised, do not climb down the post. This would require an additional sensor to detect flag position, and a simple 'if' statement added to the program.

<u>Move down post:</u>

After the mail has been delivered, the robot automatically lowers itself from the post. The bump sensor underneath the lifting platform tells the robot when it has reached the bottom of the post. The program logic is as follows:

- Ensure the brake is on
- Turn the switch on to provide power to the post motor
- Wait for post position bump sensor to read 'high' (this bump sensor is normally closed. A 'high' signal to the Mavric indicates the bottom of the post has been reached.)
    - Once the bottom has been reached, turn off the switch that delivers power to the post motor.

<u>Leaving the Post:</u>

After reaching the bottom of the post, the robot drives away from the mechanical post. This is a simple step, with the logic outlined below:

- Turn off the brake
- Apply power to both wheels by calling the 'move both forward' function for a given period of time.
    - When this happens, the robot pushes down the door at the end of the lifting platform. This door becomes a ramp for the robot to drive off the platform.

<u>Follow buried cable to base:</u>

Following the buried cable required the use of an interrupt-driven program developed by Subrat during the building and testing of the dog-collar circuit. The circuit provides a high signal when the shock zone is detected. Because only one sensor is available, the behavior is similar to a wall-following robot. The buried wire must be properly positioned relative to the post and charging station for this to work as planned: The wire must always be to the right of the path the robot is intended to follow. This is because the robot will sense the presence of the wire before reaching it, and will never actually move over the wire. In addition, the wire must be placed in a loop such that the robot is only making right turns. The logic is as follows:

- After clearing the lift platform, make a slight turn to the right.

Begin ‘wall-following’ loop:

- Move forward slowly
    - If a ‘high’ signal is detected soon:
        - Turn left a predetermined amount
        - Go straight for a short amount of time
        - Turn right again
        - Return to outer loop to repeat process
    - If a ‘high’ signal is not detected soon:
        - The wire must be curving to the right. Turn right a little more and continue moving forward.

This continues until the robot runs into the base. (Senses a ‘low’ signal on the front bump sensor.)

Additional behaviors, such as returning to the post, could be programmed easily by basically reversing the commands to the wheels. (This would require the robot to assume the wire is always to the left of its position.) Accomplishing this behavior would also require the need to turn the robot around. The main reasons for not including this behavior are:

- Unusual difficulty in reversing the motor direction. The electronic speed controllers are capable of reversing the motors, and did so during testing with a function generator. However, after repeated efforts, the correct PWM signal for this command was not found. This severely limited RoboPost’s capabilities, forcing it to only drive forward.
- Time to implement the required code.
- Challenge of correctly aligning the robot to enter lifting platform. The platform is wider than the robot, permitting a margin of error; however this task would still require some additional time to complete reliably.

11. Experimental Layout and Results

11. 1 Invisible Fence Testing

A simple test was performed to determine to position of the warning zone and shock zone boundaries relative to the wire. (See figure 19).

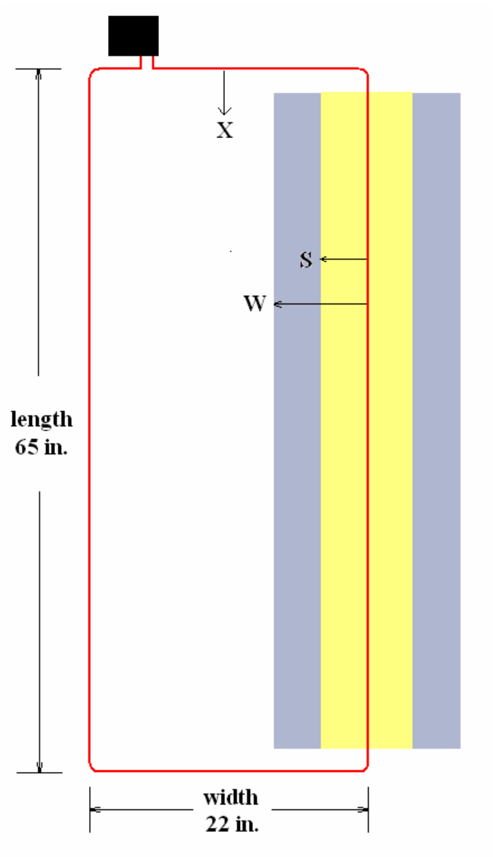


Figure 19. *Fence test schematic*

A length of wire was attached to the transmitter, and a loop was formed. A voltmeter was attached to the shock prongs of the receiver, and the receiver was taped to a cardboard box at the approximate height of its position underneath the robot platform. The transmitter was set to its lowest setting, corresponding to the smallest possible warning zone. The transmitter LED indicators verified the loop was complete and functioning properly. The box with the receiver was slowly moved towards the wire until a warning beep was heard. The receiver distance corresponding to the warning zone (W) was recorded, and the receiver was moved closer to the wire until a shock was indicated by the voltmeter. This distance (S) was also recorded. This procedure was repeated at several stations along the wire, and is summarized in table 1. A picture of the test setup is shown in figure 20.

(in.)	(in.)	(in.)	(in.)
X	**W**	**S**	**difference**
5	8	8	0
10	9.5	9.5	0
15	10.5	10.5	0
20	10.5	10.5	0
25	10.5	10.5	0
30	11.5	11.5	0
35	11.5	10.5	1
40	12	11	1
45	12.5	11.25	1.25
50	12.5	11	1.5
55	11.5	10.25	1.25
60	10.5	9.25	1.25
65	8.5	7.5	1

Table 1. *Warning and Shock zone boundary border locations*

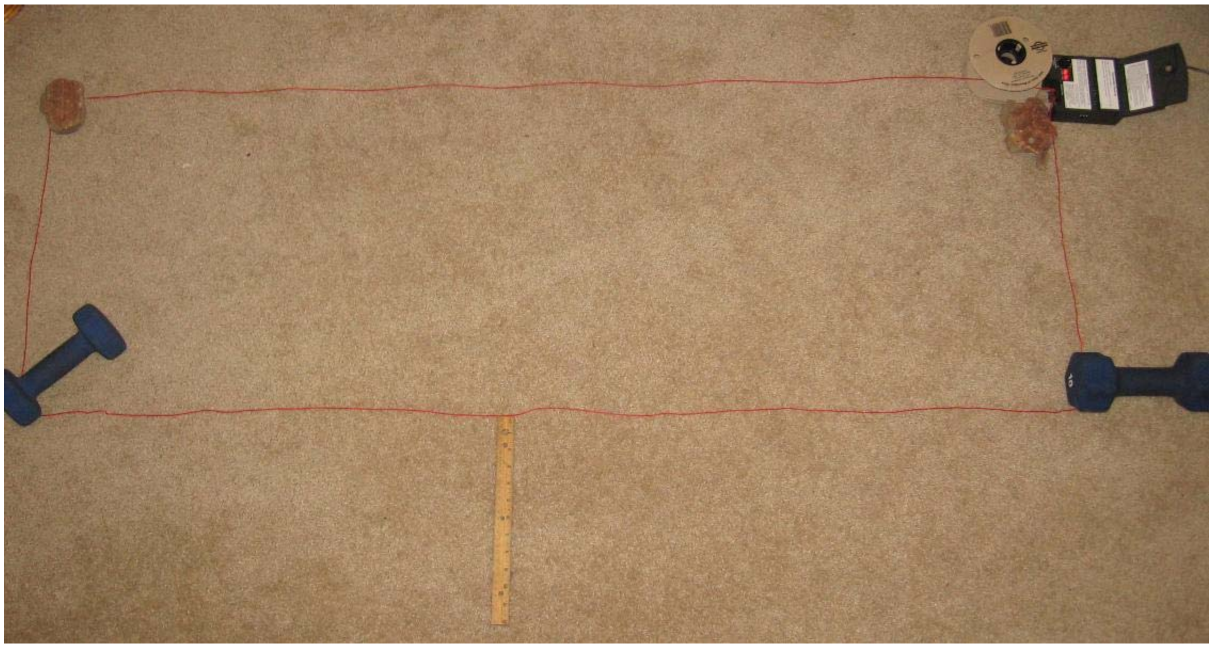


Figure 20. *Invisible fence system test setup*

As seen in table 1, a region of approximately one inch lay between the shock zone and warning zone boundaries. This would provide a sufficiently accurate path for RoboPost to follow. For locations nearest the transmitter, the receiver failed to indicate a warning zone. This tendency disappeared further from the transmitter. These results indicate that a length of wire of at least 40 inches must be left between the transmitter and the path followed by RoboPost to prevent errors.

11.2 Post Climbing Tests

Immediately after the post mechanism construction was completed, the robot was placed on the lifting platform and the robot's 7.2 V battery pack was manually connected to the post climbing motor. The connection was maintained until the motor had lifted the robot to an acceptable height, at which point the wires were reversed. This connection was held until the post had lowered the robot back to the ground. Video of this test was obtained for evidence.

Surprisingly, the mechanical post did not have any problems lifting the robot, and only faced difficulty when lowering the robot. During this phase, the track mechanism would routinely bind and jerk as the robot was being lowered, causing a rough descent. The solution was to grease the track thoroughly using WD-40 or similar products. This fixed the problem, and the mechanism then functioned as desired.

Once the robot had been properly programmed and the post mechanism was outfitted with a bump sensor, the robot was placed on the lifting platform with the platform at the top of the post. The robot was switched on, placing the post mechanism under the control of the Mavric. The mailbox door was opened, and closed immediately. After a short delay, the robot applied power to the post motor and began lowering itself from the post. Once it reached the bottom, it switched off the post motor power and released its parking brake. It applied power to its wheels to move forward, but did not properly exit the platform. (The setting for the 'move forward' command was not set for a long enough time to clear the platform.) At this point, the robot and post mechanism was moved down to the new engineering building for media day.

12. Conclusion

Although RoboPost was not completed to the original specifications, it still performed some of the original tasks and provided an excellent opportunity to develop robot construction and programming skills. The post mechanism was a complete success, and proved to be a mechanically sound device capable of achieving all of its original tasks. The robot itself was capable of driving and commanding the post mechanism, and at the time of writing this report it only lacks the integration with the dog collar sensor. This sensor has been built, and program code for interpreting the sensor is ready. Integrating this sensor with the robot will permit the wall-following behavior to be achieved, and is scheduled to be completed by the end of April.

This project provided many electrical and mechanical challenges. One of the unforeseen challenges was the construction of the opto-isolator circuit; however the dog shock sensor provided the most difficulty. This challenge would not have been overcome without the help of Subrat Nayak, who single-handedly built a custom circuit for the device. Many thanks are owed to him, along with the professors, teaching assistants, and other students who contributed their time and advice to this project.

Appendix

RoboPost.c (Main Program code):

Portions of the PWM motor code were adapter from code by Mike Pridgen

```
// RoboPost

//

// Kim Wright

// Intelligent Machine Design Lab

// Spring 2008

//


#include<avr/io.h>

#include"adc.h"


#define DIR_L PORTE2

#define DIR_R PORTE5

#define MOTOR_L OCR3A           //pin3 port E

#define MOTOR_R OCR3B           //pin4 port E

#define ServoSwitch OCR1A

#define Brake OCR1B


void init_timer();

void init_bump();

void init_servos();

void move_right_motor(uint16_t);

void move_left_motor(uint16_t);

void move_left_forward(uint16_t);

void move_right_forward(uint16_t);

void move_both_forward(uint16_t);
void turn_right_forward(uint16_t);

void stop_both_motors();

void move_both_back();

void delay(uint16_t);

void wait_for_door();

void go_down_pole();

void SwitchOn();

void SwitchOff();

void BrakeOn();

void BrakeOff();

#define DIR_L PORTE2

#define DIR_R PORTE5

#define MOTOR_L OCR3A              //pin3 port E

#define MOTOR_R OCR3B              //pin4 port E

////////////////////////////////////////////////////////////////////////////////////////

// initialize PWM timer

void init_timer()

{

        //initializes motors, enable OC3A, B, and C

        DDRE = 0xFF       ;        //PORT E out

        //DDR_OC3A = 0b1;

        //DDR_OC3B = 0b1;

        TCCR3A = 0xA8;   //10101000 clear OCA,B,C on campare match- non-inv PWM

        TCCR3B = 0x12;   //00010010 mode  8 - PWM phase/freq correct, clk(io)/8

        ICR3 = 18432;    //PWM = 50hz.  1/(14.7456 MHz/8)*2*18432 = 1/50

        TCNT3 = 0x00;    //

        MOTOR_L = 0;     //not moving

        MOTOR_R = 0;

        //PORTE = 0x0;
```

```
}

//////////////////////////////////////////////////////////////////////////////////////////

// intialize servos timer


void init_servos(void)

{

		DDRB=0xFF;			// PORTB out

		TCCR1A = 0b10100000;   // 0b1010x000 write 1 to the first x to enable output 1A, a 1 to the second to
enable 1B, and a 1 to the third to enable aC

		TCCR1B = 0x12;	//set clock to divide by 8

		ICR1 = 0x8F9C;	//14.7 MHz = 68ns.  *8 (for clock divider) = 544ns.  20ms / 544 ns = 36764 = 0x8F9C

		TCNT1 = 0x00;

		//OCR1X = 0x0AC5;	//1.5ms period (same math as ICR).  X = A, B, or C depending on which PWM output
you want to use.

		ServoSwitch = 0;

		Brake = 0;

}

//////////////////////////////////////////////////////////////////////////////////////////

void init_bump()

{

		// Set pins 1,2,3,4 as input pins on PortA:

//		DDRA = ((DDRA &11100001) | 0b00000000);

		DDRA = 0x00;	// Sets PortA as input

}

//////////////////////////////////////////////////////////////////////////////////////////

void delay(uint16_t dt)	// delay for 10000 clock cycles

{

  long int ms_count = 0;

  while (ms_count < dt*10000)

  {
      ms_count = ms_count + 1;
   }
}
/////////////////////////////////////////////////////////////////////////////////////////
void move_right_motor(uint16_t speed)
{
        int increment_R = 0;
        if(MOTOR_R > speed)
        {
                increment_R = -1;
        }
        else if (MOTOR_R < speed)
        {
                increment_R = 1;
        }
        else
        {
                increment_R = 0;
        }
        while (MOTOR_R != speed) // if right motor does not equal speed, run this loop
        {
                if (MOTOR_R != speed)
                {
                        MOTOR_R =MOTOR_R + increment_R;
                }
                //delayus(50);
                //lcd_cmd(0x01);          // clrscn rtn home
                //lcd_int(MOTOR_L);
        }
```

```
}
/////////////////////////////////////////////////////////////////////////////////////
void move_left_motor(uint16_t speed)
{
        int increment_L = 0;
        if(MOTOR_L > speed)
        {
                increment_L = -1;
        }
        else if (MOTOR_L < speed)
        {
                increment_L = 1;
        }
        else
        {
                increment_L = 0;
        }
        while (MOTOR_L != speed ) // if left motor does not equal speed, run this loop
        {
                if (MOTOR_L != speed)
                {
                        MOTOR_L = MOTOR_L + increment_L;
                }
                //delayus(50);
                //lcd_cmd(0x01);          // clrscn rtn home
                //lcd_int(MOTOR_L);
        }
}
/////////////////////////////////////////////////////////////////////////////////////
```

```
void turn_right_forward(uint16_t time)
{
  int dt = 1;
  long int ms_count = 0;
   while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
            move_left_motor(1200);
            move_right_motor(1300);
  }
  while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
            move_left_motor(1150);
            move_right_motor(1300);
  }
  ms_count=0;
  while (ms_count < time*10000)
  {
    ms_count = ms_count + 1;
            move_left_motor(1100);
            move_right_motor(1300);
  }
           stop_both_motors();
}
//////////////////////////////////////////////////////////////////////////////////////////
void move_left_forward(uint16_t time)
{
  int dt = 1;
  long int ms_count = 0;
  while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
		move_left_motor(1250);
  }
  ms_count=0;
  while (ms_count < time*10000)
  {
    ms_count = ms_count + 1;
		move_left_motor(1320);
  }
	stop_both_motors();
}
//////////////////////////////////////////////////////////////////////////////////////////
void move_right_forward(uint16_t time)
{
  int dt = 1;
  long int ms_count = 0;
  while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
		move_right_motor(1160);
  }
  ms_count=0;
  while (ms_count < time*10000)
  {
    ms_count = ms_count + 1;
		move_right_motor(1260);
  }
            stop_both_motors();
}
/////////////////////////////////////////////////////////////////////////////////////
void move_both_forward(uint16_t time)
{
  int dt = 1;
  long int ms_count = 0;
  while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
             move_right_motor(1180);
             move_left_motor(1180);
  }
  ms_count=0;
  while (ms_count < time*10000)
  {
    ms_count = ms_count + 1;
             move_right_motor(1260);
             move_left_motor(1260);
  }
            stop_both_motors();
}
/////////////////////////////////////////////////////////////////////////////////////
void stop_both_motors()
{
            MOTOR_L = 0;    //not moving
            MOTOR_R = 0;
}
```

```
/////////////////////////////////////////////////////////////////////////////////////
void move_both_back(uint16_t time)
{
  int dt = 1;
  long int ms_count = 0;
  while (ms_count < dt*10000)
  {
    ms_count = ms_count + 1;
            move_right_motor(1300);
            move_left_motor(1300);
  }
  ms_count=0;
  while (ms_count < time*10000)
  {
    ms_count = ms_count + 1;
            move_right_motor(1300);
            move_left_motor(1300);
  }
          stop_both_motors();
}
/////////////////////////////////////////////////////////////////////////////////////
void wait_for_door()
{
          int bump;
          bump = PINA;
          while ((bump & 0x04)==0) // while mailbox door is closed
                    {
                    bump = PINA;
                    if (!(bump & 0x04)==0)
                    break;
                // do nothing
                }
                // if out of loop, door was opened
        //delay(500);     //wait some more for them to close the door
}
//////////////////////////////////////////////////////////////////////////////////////////
void go_down_pole()
{
        int count;
        int bump;
        int pole_bottom;
        pole_bottom = 0;
        bump = PINA;
        SwitchOn(); // turn on pole climbing motor switch
        while (pole_bottom==0) // while bump sensor on base is not bumped
        {
                        bump = PINA;
                        if (!(bump & 0x08)==0)    // if pole bottom bump switch is bumped
                        {
                            pole_bottom = 1; // pole bottom found
                        }
        }
        SwitchOff();   // turn off pole climbing motor switch
//count = 0;
//while (count < 185)
//      {
//              delay(1000);
//              count++;
```

```
//		}
//
//		SwitchOff();		// turn off pole climbing motor switch
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
void SwitchOn()
{
		ServoSwitch = 880;
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
void SwitchOff()
{
		ServoSwitch = 1110;
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
void BrakeOn()
{
		Brake = 1700;
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
void BrakeOff()
{
		Brake = 1300;
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////	MAIN PROGRAM BLOCK	/////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
int main (void)

{

        init_timer();

        init_bump();

        init_servos();

        int bump = 1;

        int found_base = 0;         // base sensing bump sensor

        int door = 0;               // mailbox door bump sensor

        int pole_bottom = 0;

        int count=0;

        SwitchOff();        // pole climbing switch off

        BrakeOn();                  // brake on

        bump = PINA;

        //if ((bump & 0x08)==1)     // if pole bottom sensor is connected

        //{

        //      Brake = 1700;               // Brake set means connections are correct

        //}

        wait_for_door();  // do nothing until someone opens and closes door

while (count < 80)

        {

                delay(1000);

                count++;

        }

        go_down_pole();

        delay(2000);        // wait a moment

        BrakeOff();                 // brake is off

        while (count < 20)

        {

                delay(1000);
```

```
                    count++;
          }
move_both_forward(18);
          stop_both_motors();
          while(1)
          {
                    bump = PINA;
                    if ((bump & 0x02)==0)     // If base bumper is bumped
                              {
                                        found_base = 1;
                                        //SwitchOn();
                              }
                    else
                              {
                                        found_base = 0;
                                        //SwitchOff();
                              }
                    if ((bump & 0x04)==0)     // If mailbox door bump is bumped (door closed)
                              {
                                        door = 1;          // door is closed
                                        //Brake = 1700;  // Brake is on
                                        //ServoSwitch = 1000;               //Switch off
                              }
                    else
                              {
                                        door = 0;          // door is open
                                        //ServoSwitch = 920;       // switch on
                                        //Brake = 1300;            // brake off
                              }
```

```
}

                    if (!(bump & 0x08)==0)      // if pole bottom bump switch is bumped
                              {
                                        pole_bottom = 1; // pole bottom found
          //                            BrakeOff();
          //                            SwitchOff();
                              }
                    else
                              {
                                        pole_bottom = 0; // pole bottom not found
          //                            BrakeOn();
          //                            SwitchOn();
                              }
                    if (found_base == 1)
                              {
                                        turn_right_forward(3);
                              }
                    else
                              {
                                        stop_both_motors();
                              }
                    if (door == 1)
                              {
                    //        ServoSwitch = 1000;
                              }
                    //else
                    //        ServoSwitch = 920;
                    }
return 0;
}
```

```
// end of main

// Right motor:

// 1150 = quite fast forward

// 1250 = fast forward

// 1300 = difficulty starting

// Left motor:

// 1150 = quite fast forward

// 1250 = fast forward

// 1300 = gentle forward
```

Dog Collar Sensor Code: (by Subrat Nayak)

```
/*

kimtest1.c

if the input signal on pin INT0 (Port D, Bit 0) is cheked as source of extrenal interrupt

*/

/*

connect the signal pin from black box to INT0 (Port D, Bit 0)

there are 8 pins on the black box

1) and 2) - conenct the shockin output terminals to them - doesnt matter even if u swap 1) and 2) bcose input is ac

3) and 4) - comes from the battery power cable to the sensor - doesnt matter even if u swap 3) and 4)

5) and 6) - be very careful.. connect 5v and gnd from maverick board to this... - if you swap the terminals

you going to burn the circuit used of sensor(black box)... see the label next to pin 5) and 6) on the black box

7) and 8 ) - both are shorted inside - i needed one but put 2 two make it stronger....

any pin can be used as signal and needs to be conencted to (Port E, Bit 2) of mavrick board..

*/

/*--------------------------------------------------------------

-----------------HEADER FILES------------------------------------

---------------------------------------------------------------*/

#include <avr/io.h>

#include <avr/signal.h>

#include <avr/interrupt.h>

/*--------------------------------------------------------------

-------------CONNECTION BETWEEN Analog inputs AND ATMEGA128---------

---------------------------------------------------------------*/

#define Ext_Interrupt_DDR                          DDRD

#define Ext_Interrupt_PORT                 PORTD
```

```
#define Ext_Interrupt_PIN                          PIND

#define Ext_Interrupt_Int0    0

/*---------------------------------------------------------------

-------------CONNECTION BETWEEN LED AND ATMEGA128-----------------

---------------------------------------------------------------*/

#define LED_Output_DDR                             DDRB

#define LED_Output_PORT                    PORTB

/*---------------------------------------------------------------

-------------IMPORTANT/USED BITS OF REGISTER ACSR-----------------

---------------------------------------------------------------*/

#define ACIE_bit  3

#define ACIS1_bit 1

#define ACIS0_bit 0

#define ACO_bit            5

/*---------------------------------------------------------------

-------------IMPORTANT/USED BITS OF REGISTER EICRA-----------------

---------------------------------------------------------------*/

#define ISC01_bit  0

#define ISC00_bit  1

/*---------------------------------------------------------------

-------------IMPORTANT/USED BITS OF REGISTER EIMSK-----------------

---------------------------------------------------------------*/

#define INT0_bit  0

/*---------------------------------------------------------------

-------------IMPORTANT/USED BITS OF REGISTER EIFR-----------------

---------------------------------------------------------------*/

#define INTF0_bit  0

/*------------------------------------------------------------------
```

```
----------FUNCTION PROTOTYPES - its rather written in the sublcd.h-------

---------------------------------------------------------------*/

void Ext_Interrupt_Init_Ports(void);

void LED_Output_Init_Ports(void);

void Ext_Interrupt_Init(void);

void Ext_Interrupt_Delay(void);

/*---------------------------------------------------------------

---------------INTERRUPT FUNCTION FOR Ext_Interrupt on int0--------------

control comes here when we get a rising edge on pin int0 (Port D, Bit 0) ----

---------------------------------------------------------------*/

SIGNAL(SIG_INTERRUPT0)

{

LED_Output_PORT = ((LED_Output_PORT & 0b11111110) | 0b00000001);

// sets the bit 0 of LED_output_PORT to turn on the LED on PortB.Pin0.

// MAKE THE ROBOT MOVE TO LEFT - IF THE WIRE IS ON RIGHT OF ROBOT

//move to left means - make both wheels moving forward --

//but right wheel shud be very fast while left wheel must be very slow

// PUT SMALL DELAY

// PUT DELAY-LEFT -> may need to keep varying it (by trial and error) to get better response

while((Ext_Interrupt_PIN & _BV(Ext_Interrupt_Int0)) >> (Ext_Interrupt_Int0))

        {

        //wait till Int0 remains high

        }

LED_Output_PORT = ((LED_Output_PORT & 0b11111110) | 0b00000000);

// clears the bit 0 of LED_output_PORT to turn off the LED on PortB.Pin0.

// MAKE THE ROBOT MOVE TO RIGHT - IF THE WIRE IS ON RIGHT OF ROBOT

//move to right means - make both wheels moving forward --

//but left wheel shud be very fast while right wheel must be very slow
```

```
// PUT SMALL DELAY

// PUT DELAY-RIGHT -> may need to keep varying it (by trial and error) to get better response

}

//----------------------------------------------------------------

//----------------MAIN FUNCTION-------------------------------------

//----------------------------------------------------------------

int main( void )

{

 Ext_Interrupt_Init_Ports();

 LED_Output_Init_Ports();

 Ext_Interrupt_Init();

 sei();

 //do whatever

 //rest of your code comes here...

 for (;;){}

// ensure that the boundary between the danger zone(danger beep) and safe zone(safe beep or no beep),

// is exactly at middle of you docking station, else it will not enter the docking station...

// disable wheel motor control and inettrupts when the robot is inside the docking station,

// you need a good reliable bump switch for this....

return 0;

}

//---------------------------------------------------------------------

//------------FUNCTION TO INITIALIZE PORT FOR Ext_Interrupt0-----

//-----------------------------------------------------------------------

void Ext_Interrupt_Init_Ports(void)

{

Ext_Interrupt_DDR = ((Ext_Interrupt_DDR & 0b11111110) | 0b00000000);
```

```
//clears bit 0  of the Ext_Interrupt_DDR to make pin 0 as input pin

Ext_Interrupt_PORT = ((Ext_Interrupt_PORT & 0b11111110) | 0b00000001);

// sets the bit 0 enable the pull up resitors for pin 0

}

/*--------------------------------------------------------------

-------------FUNCTION TO INITIALIZE PORT FOR LED OUTPUT------------

----------------------------------------------------------------*/

void LED_Output_Init_Ports(void)

{

LED_Output_DDR = ((LED_Output_DDR & 0b11111110) | 0b00000001);

//sets bit 0 of the LED_output_DDR to make pin 0 as output pin

LED_Output_PORT = ((LED_Output_PORT & 0b11111110) | 0b00000000);

// clears the bit 0 of LED_output_PORT to have it LOW by default.

}

//--------------------------------------------------------------

//-----------------FUNCTION TO INITIALIZE THE Ext_Interrupt0 ------

//---------------------------------------------------------------

void Ext_Interrupt_Init(void)

{

EICRA = ((EICRA & 0b11111100) | 0b00000011); // to sense rising edge on int0

EIMSK = ((EIMSK & 0b11111110) | 0b00000001); // enable ext interrupt on int0

}

/*--------------------------------------------------------------

------------ FUNCTION TO HAVE A DELAY ---------------------------

----------------------------------------------------------------*/

void Ext_Interrupt_Delay(void)

{

  long int ms_count = 0;
  while (ms_count < 200)

  {

    ms_count++;

  }

}
```

Datasheets for the Analog Devices Opto-isolator:

This document is 20 pages long, and may be found for free from analog device's website for the iCoupler technology. The isolator used in this project is the ADUM1200CRZ. The website for the datasheet is:

http://www.analog.com/UploadedFiles/Data_Sheets/ADUM1200_1201.pdf