Island Escape

Final Project Report

Mechatronics EN.530.421 (1)
Submission Date: May 15, 2012

Team Hot Rod

Robot Name:
Jonathan Taylor Thomas (JTT)

Group Members:
Adam Merritt
Cassie Tarakajian
Kyle Rohrbach
Shing Shin Cheng

Introduction

The main objective of this robotics project was to create a robot that autonomously navigates a course. This course, called “Island Escape,” is a 96” x 144” rectangle which has an 8” “lava” stream running through the middle, separating the start zone from the finish zone. In order to reach the end zone from the start zone, the robot must traverse the lava river without touching it. Therefore the robot had to properly navigate the course by staying within the bounds, not touch the lava, and end in the goal zone. Other objectives were to lift and transport two boxes from the starting side of the lava to the end zone, differentiate between the two colors of the boxes (white and black), and to deliver the white box into the goal first. In order to accomplish our main objective, we drew from Mechatronics class lectures, labs and homeworks, on topics such as line following, state machines, motor control, and Arduino programming. With other sources of creative inspiration, we were able to create a robot that almost successfully completed the course.

There are many different modes of locomotion for a robot to transport itself from place to place. Two such subsets are wheeled and legged locomotion. In legged locomotion, a robot uses (usually an even number) of legs to move around. These legs are designed in the style of human, mammalian, insect, etc. legs. There are also robots designed to slither like a snake, or move like a war tank (wheeled tracked). Controlling legs on a robot is similar to controlling robot arms, except there are a lot of legs that need to be controlled in parallel, which complicates the problem of motion very quickly. In the end, we decided to use wheeled locomotion because it is simple: when using two wheels, there is a simple equation to convert the two motor velocities to robot position.

Given the two velocities of the motors, we can calculate the position and orientation of the robot in the following manner:

$$\dot{g} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{r}{2}\cos(\theta) & \frac{r}{2}\cos(\theta) \\ \frac{r}{2}\sin(\theta) & \frac{r}{2}\sin(\theta) \\ -\frac{r}{L} & \frac{r}{L} \end{pmatrix} \begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix}$$

Where r is the wheel radius and L is the length of the wheel axle. We can use this to calculate the position *g* of the robot, time step by time step, via:

$$g(t + \Delta t) = g(t) + \dot{g}\Delta t$$

Where *t* is the current time and *t* is the time step interval. Given the motor velocities and position at any time, one can calculate the the robot position at the next time step.

It may seem like the easiest thing to do is figure out the exact path we want the robot to follow in terms of a series of *g*’s as a function of *t*; however, this is a complicated problem involving a lot of difficult calculations. It would be possible to do this calculation ahead of time, and program the robot with the series of $\dot{\phi}_1 \dot{\phi}_2$

In the real world, this is not useful due to environmental errors, and therefore we must use a feedback control system. We want to have some sensors on the robot in order to have a frame of reference as to where the robot is at any time. One such solution is to use a technique called line-following. In this scheme, a dark line is placed on the light-colored ground (or the colors vice-versa) as the trajectory that one wants the robot to follow. One then attaches infrared sensors to the bottom of the robot to determine where the robot is with respect to the line.

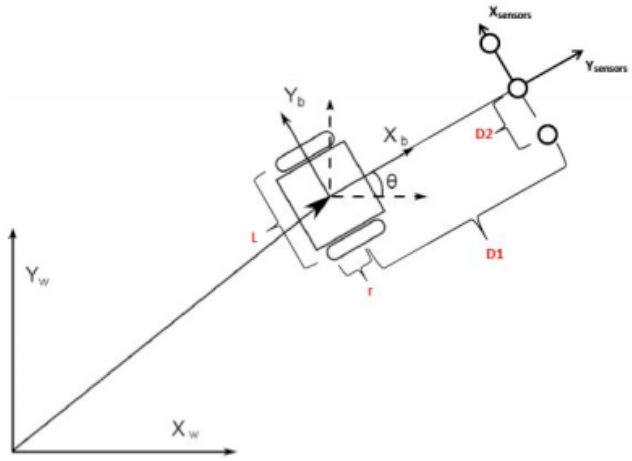

Figure 1: Robot and sensor orientation example, taken from Mechatronics 530.421 Spring 2012 Homework 4: PID Control of a Kinematic Cart by Gregory Chirikjian

Depending on which sensor(s) are over the line, the control system of the robot can determine at any time step whether the robot should keep its same trajectory and not alter the motor velocities because the robot is centered over the line, or change the motor velocities because the robot is moving off the line. The amount we alter the motor velocities is determined by the error: the difference between the desired position of the robot and the current position of the robot with respect to the line. The further the robot is off the line, the larger the error. For example, in a three sensor system as shown in Figure 1, if the middle and left sensor are over the line, the error is smaller than if just the left sensor is over the line.

One highly respected way to control the motor velocities is called the PID (Proportional, Integral, Derivative) control. In this method, the motor velocities are altered each time step with a term $u(t)$:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t)$$

Where $e(t)$ is the error, K_p is the proportional error constant, K_i is the integral gain constant, and K_d is the derivative gain constant. The $K_p e(t)$ term simply adds or subtracts a scaled error to the motor velocities. The integral term keeps track of all past errors, sums them and scales this term with K_i. The derivative term predicts the future error, by subtracting the last error from the current one, and then scales this with K_d. This allows $u(t)$, the adjustment term, to be determined not only by the current error, but by past and future errors as well. This means that the motor velocities are adjusted more "intelligently," and therefore the robot is able to follow the line more closely by not overshooting or undershooting as much.

Another aspect of this robot design is the method to pick up the two boxes. One way to do this is using a robot arm. Robot arms can be made up of multiple bars and linkages, but in our case, one degree of freedom is suitable. Because the boxes have magnets on each face, there are two methods for picking up the boxes: gripping like a human hand to either scoop or grab, or using a magnetic material.

The last aspect in the design of this robot is the method to transport the robot across the lava stream. Since there is a ramp leading up to the lava pit, one method is to have the robot move fast enough to jump the pit. Another safer method is to drop a bridge down across the pit. This bridge can be dropped either over the lava river between the ramps or another section that is at level ground.

With numerous options for our robot design, our creative process for settling on a design is discussed further in the next section.

Materials and Methods

To complete the final challenge course, we focused on accomplishing each smaller goal, including line following, carrying and placing down a bridge, lifting two boxes, traversing the bridge, ending in the goal, and determining the color of the boxes. In this section, we will break down the methods for our attempt to accomplish each smaller goal.

The first step, before any other goals could be accomplished, was creating a robot that could simply follow a line. We decided to use wheeled locomotion in order for the robot to use since it is one of the simplest forms of movement to control. To control the entire robotic system, including the line following and the other above mentioned goals, we used an Arduino Uno. We had two wheels, each controlled by a DC brush motor, and a ball caster in the front, not connected to any motor but used for balance. Since the motors included Hall-effect encoders, it was possible for us to use position, as well as velocity control. However, we decided for simplicity's sake to use only velocity control. In order to control the velocity, we connected each motor power lead to one of the Arduino PWM (pulse-width modulation) pins. The higher the duty cycle of the PWM signal, the greater the velocity of the motor. In order to move the motors in both directions, we had to connect each motor lead to an Arduino PWM pin. Since this involves using four PWM pins, we decided to add multiplexers to our system so that we only needed two PWM pins (and two digital pins) to control the motor velocities. This schematic connecting the multiplexers to the motor drivers can be seen in the Appendix, Figure 3.

Once we were able to control the velocity of the DC motors, we began assembling the line following system. We used an array three Pololu reflectance sensors in order to see the black line on the white background. We also put a reflectance sensor underneath each wheel, to make it easier to make sharp turns. Instead of having to hardcode a sharp turn, we were able to wait until the wheel reflectance went dark to make a perfect 90° turn. To use the data from the reflectance sensors, we implemented PID (proportional, integral, derivative) control. When the robot drifts off the line, there is an error in the position of the robot, and the velocities of the motors are updated at a value proportional to the error. For example, if the robot drifted to the right of the line, then the middle reflectance sensor would no longer see the black line, and the black line would be under the left sensor. The error would be 1 (or -1 if the robot were to the left of the line), and the right motor velocity would be increased and the left motor velocity would be decreased in order to turn slightly right. This causes the robot to go back to being centered on the line.

While in PID control the motor velocities are dependent on the current error, they are also dependent on past and future errors. This is where the integral and derivative terms come into play. The integral term sums all of the past errors, and the derivative term predicts future errors. This makes the line following more accurate, because it takes into account the differences in friction between the two motors and other variances.

Once the robot's line following was solidified, we moved onto the design for a method to lift the boxes. After a very complicated initial design, as explained in our design report, we decided to implement a much simpler design. We attached to the front and rear of the robot a steel plate, that would attach to the boxes using the magnets attached to the boxes. In the center of the each plate was also a contact switch, so that the Arduino would know when a box had been attached to the plate. Once the box was attached to the plate, it had to be lifted off the ground so that it would not drag and cause the robot movement to be very difficult. We accomplished this by attaching a one-degree-of-freedom arm to the top of the plate using string. When the arm is pulled back, it also slants the steel plate upwards, causing the block to the lifted off the ground. Once the box is lifted off the ground, the Arduino continuously supplies the PWM signal to the servo controlling the arm, thus holding the plate (and therefore the block) in place. The plate design can be seen in the Appendix, Figures 2a and 2b.

The supply power to our two servo motors (HSR-5498SG and HS-5646WP) was 6.0 V, which means they had a stall torque of 11 kg•cm. Because the blocks weighed 0.5 kg, and our connection to lifting the blocks was a linkage, we needed a maximum of 0.5kg * 7 cm (the distance from the point of rotation, about midway to the edge of the block). = 3.5 kg•cm. Therefore our servos were adequate to lift up the boxes. To ensure that the boxes would initially stick to the steel plate, we "rammed" the robot into the boxes. This means when we were

close to the boxes, we greatly increased the motor velocities and stopped the cart once the contact switch was enabled (i.e. the box was attached to the plate).

The next step was to determine a method to traverse the lava. This could be done in a few ways: try to jump the gap, place a bridge in the gap between the ramps, or place a bridge down over one of the other sections of the lava river. Using a bridge seemed to be the most reasonable and practical method. We decided to use a bridge made of aluminum sheet which was light and yet strong enough to support the robot's weight. We were thinking of using a steel bridge at first because the idea was to carry the bridge by attaching it to magnets placed on an arm. The fact that the design was too complicated and the steel bridge was really heavy made us scrap the idea. In terms of where to place the bridge along the lava, our original design idea was to place it down between the gap of the two ramps, because this meant we could follow the line the whole time and not have to dead reckon for a section of the course to find the other section of the lava to lay down a bridge. However, in testing we realized that the robot did not have enough power to drive up the ramp with the two boxes attached. Therefore we had to drop a bridge at another section of lava. For the sake of speed, we decided to drop it over the section of lava nearest to the goal, to the right of the ramps.

To carry the bridge, we simply used a very strong piece of velcro to attach the bridge to one of the steel plates. Then, to drop the bridge over the lava, we stopped the cart once it hit the cross, and pushed the off the plate by swinging the front servo arm (attached to the front steel plate) so that it forced the velcro apart. Since the velcro was attached in the center of the bridge and the center of the steel plate, and since the arm was attached in the center of the robot, if the robot was lined up with the lava river, the bridge fell straight down, thus making it simple for the robot to cross the lava pit. We also put a strip of electrical tape down the center of the bridge, so that the robot could line follow across the bridge.

We also had to determine the color of each box. On the back steel plate, we attached the ColorPAL color sensor. Since the ColorPAL emits its own light, the block could be attached to the plate when the color was determined. In testing the ColorPAL, we also realized that when initially reading the color of the object in front of the sensor, there is a spike in values. Therefore, we decided to read the value for a few times before using the value to determine the color of the box. We also decided to take the average of a few values so that the reading would be most accurate. The ColorPAL returns the color of an object by an RGB value. Since the black RGB value is "0 0 0" and the white RGB value is "255 255 255", we only needed to use the R value to determine the color of the box. Due to the boxes not being perfectly white or black, we decided that 90 was a good value to choose as the threshold between white and black: if the R value was above 90, the back box was white, and if it was below 90, then the back box was black.

One of the last objectives was dealing with the sections in which the robot could not line follow. These sections were when the robot had to drop the bridge, return to the line to pick up the boxes, and then back to the bridge to cross it. This involved using trial and error to time how long the robot had to move in a certain direction, the angle of the turns, the curvature of a path, or the straightness of a path. For the dead reckoning, it would have been useful to have implemented position control to spend less time on the trial and error portion, but unfortunately we did not have the time to implement position control.

Our overall system schematic, our mechanical drawings, and photos of the robot are included in the appendix.

Discussion

On the robot was attached front and rear metal plates used to pick up the blocks. The arms that attached to the servo motors were made of “Erector Set” plates. These arms were then tied to the top of the metal plates to allow for bending of the plates. By doing so, after a cube would stick to the plate, the servos would tilt the plates upwards, causing the blocks to be raised off the ground. This was necessary since the team found out dragging the blocks along the ground caused too much friction for the DC motors to overcome.

Since each arm pulled a plate back towards the robot body to lift the cube, a low amount of torque was required. Due to previous designs requiring more torque, the team bought two high torque motors rated at 17 kg•cm. During wiring the night before the competition, the servos were incorrectly installed causing them to fail. The servo motors found to replace the high torque motors were rated at 13 kg•cm which was still more than suitable for the robot. These servo motors covered angles up to 140° (20-160). After several trials, it was determined that the arms needed to be rotated back 70° to allow for ample bending of the plate.

The bridge was made out of thin aluminum sheet metal, with a white paper wrapped around it and a black line drawn in the middle to allow the robot to do line following. It was also bent on two of its ends to make sure it did not touch the lava when placed on the course. We attached the bridge to the front plate using a small piece of velcro. Then, to drop the bridge, the arm attached to the front servo was rotated forward, causing it to push the bridge off. If the robot was not lined up with the lava river, the bridge did not squarely fall into place. Small adjustments to the code allowed the team to dead reckon further from the lava, giving the robot additional time to line up with the course.

To begin the course, we first calibrated the lreflectance sensors before placing the robot in the green start zone. After dead reckoning in a straight line for about two seconds, the robot would follow the line, soon coming to a right hand turn. During turns, the robot would first recognize the turn when two or all three of the front sensors were above a line. Then the robot would then slow down and wait until the corresponding wheel sensor was above the line. With the DC motors wheelbase inline with the turn, the wheels would rotate in opposite directions to each other to make a perfect 90° turn. The robot would continue to turn until the front middle sensor was above the new line. The robot employed the use of dead reckoning by turning to the right after two seconds of driving past the crossroad.

Due to the nature of dead reckoning, the team had to adjust the turn angle and speed. Once figured out via trial and error, the robot was then followed the line until it found the edge of the lava. The robot stopped, and then the front arm pushed the bridge off the front plate. After doing so, the robot would back up slightly and turn 180° to continue line following. Once it reached the end of the line, it made a slight change in angle to the right and reach the line that would lead it to the other cube on the left side of the course. Several adjustments again were made, and eventually it was able to line follow and speed up after it passed the crossroad.

After picking up the left side cube, the robot would back up and perform a 180° turn to continue to the other cube. As it got close enough to the cube, it backed up and turned 180° and backed into the cube so that the back plate picked it up. With two cubes on it, the robot then went forward for two seconds, made a right diagonal turn, went forward for a second, made another right diagonal turn and then went straight. It would follow the line to the lava, to cross the bridge it had laid down earlier. In our most successful attempt, the robot reached the bridge with two cubes in tow. The bridge, however, was not properly lined up, resulting in the robot being unable to drive over it.

After concluding our first trial, the team was concerned the robot would be unable to pick up both cubes and get them into the goal zone. To ensure the team got enough points, the robot was then programmed to drop the bridge and drive over it into the final location, unfortunately without either of the boxes.

We had been working on the robot for many weeks, but still felt it needed to ramp up efforts towards the end. Our most significant set back around midnight the day of the competition results in both of the servo motors failing. After disassembling and replacing the motors, the robot was then plagued by multiple circuit board and wiring problems causing many hours of debugging to occur.

In the first mini-challenge, the robot did a fantastic job in line following. It successfully completed the course in the first trial and was tweaked over the course of the class period to improve robot speed. We also only employed one sensor on one side of the robot when we did the first mini challenge. Since we would need to make turns in both directions in the final competition, we attached one more reflectance sensor to the other wheel. For the second mini-challenge two plates were placed back to back, of which the front plate had three rectangular holes in it and the back plate had three small pieces of metal glued to it. The back plate could be pulled forward to attract the cube and pulled backward to disengage the cube. There was a huge amount of friction involved and the design was not the easiest to construct. The design was built with the understanding that the cubes would have to be dropped; later it was found this would not be the case. After much thought, the team decided to just glue the two plates together, which would be used to pick up the cube and bring it to the destination. Due to time limitations, the team only had a day to test prior to the second mini challenge. At this point an Arduino Mega was being used to take advantage of the additional PWM pins, but was determined to be faulty the day before the challenge. In addition, the night beforehand the servo motor was found to be faulty. After much time troubleshooting the robot, we were only able to find the cube and ram into it.

After the second challenge, the cube picking up mechanism was redesigned for simplicity. The modifications performed are outlined above. The team experimented with several bridge lowering mechanisms, including carrying the bridge on top of the robot and lowering it using the rotational movement of an arm, and also placing the bridge in front of the robot and lowering the bridge similar to a traditional drawbridge.

We believe a few design ideas added to the strengths of the robot. The two reflectance sensors right next to the wheels of the robot that enabled the robot to turn very precisely at junctions and crossroads. This is because we did not have to dead reckon and time the turns using trial and error; we could wait until the wheel sensor was directly above the line and then make a perfect 90° turn. It was also simple for the robot to lift up both cubes, due to its servo motor-plate bending concept. With our design we needed minimal torque to lift the box. Lastly, the robot's bridge lowering mechanism was simple to implement and very reliable. If the robot were properly lined up with the lava river, the bridge would reliably fall straight.

There were also a few weaknesses in our design. Because the line following reflectance sensors were placed as far apart as they were (with about 1.5 cm between them), we were not able to use the all white state as the state that the robot had reached the end of a line. This is because the line could be directly in between two of the sensors while still properly following the line. The spacing of our reflectance sensors also lead to the line following not being as tight as it could be: the robot would zigzag more than desired on the line. This also made it difficult to line up the robot properly with the lava river to drop the bridge, because sometimes the robot would be correcting itself when it reached the river and not be perfectly lined up with the cross.

If given more time to complete the robot, there are a few steps we would take to perfect our design. First of all, we would place the reflectance sensors closer together, as well as use an array of five sensors instead of three. We would also implement position control to allow for more precise movements of the robot. Lastly, we would clean up both the mechanical and electrical designs, by removing the mess of wires and using sturdier materials to hold the robot together (e.g. use epoxy and hot glue instead of tape). The overall messiness of the robot is due to last minute changes in the robot's mechanical design, but with three more weeks we could cement our ideas and therefore clean up the mess.

Conclusion

The goal of our Mechatronics project was to create a robot that successfully and autonomously completes a course involving a series of challenges. In the end, we were able to almost successfully complete the course. In one run, the robot dropped the bridge, traversed the lava river and reached the end goal. In another run, the

robot was able to drop the bridge, lift and transport both boxes to the bridge, but did not successfully cross the bridge. Due to the number of problems we faced the night before the challenge (breaking the servo motors, shorted electrical connections), we did not have enough time to test on the final challenge course. We believe, however, with a short amount of time more, we would have been able to complete the challenge.

With a little more time, I think we would have been able to better tweak the dead reckoning sections of completing the course. I also think we would have been able to drop the bridge more straight by approaching the lava river dead on. Lastly, we would have been able to fully implement the color sensor and determine which color each box was. We were able to complete every aspect of the challenge separately, just not in one run.

We believe the most useful aspect of this course was the experience of combining an electrical, mechanical, and programming system together. In combining these systems one learns how to debug each separately and together. This is an incredibly useful skill in that most of us up to this point have only had experience debugging each system separately. Another useful skill of combining these systems is learning how to efficiently and properly interface them. For example, it is important at the top of one's Arduino code to define all the pins and which hardware device is using them, in order to make assembling the hardware-programming interface as simple as possible.

In terms of designing each separate system, there was a lot of knowledge to gain. Arduinos are used in the design of many robotics projects, as well as many other projects. PID control is used in many different robotics systems as robot navigation. DC brush motors and servo motors are the most common motors used in robotics. In the end, it was good experience to build a relatively simple robot, since all of the components are used in more complicated robots, as well as other electronic and mechanical systems.

Before starting any of the challenges, our robot design was fairly complicated. As seen from the design report, we had a mechanism for dropping the boxes. One big lesson we learned is that simple, elegant solutions are always better. The more difficult the design, the more that can go wrong. Once we stripped down our design to the most basic components that could still complete the course, only then were we successful.

Appendix

CAD Drawings

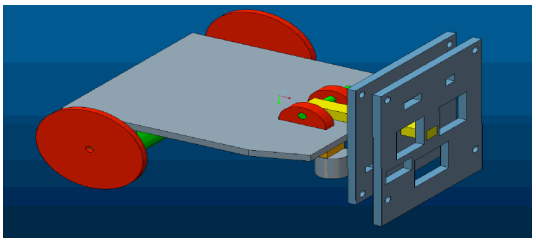

Figure 1: The initial design of the robot used to compete in the first mini challenge.

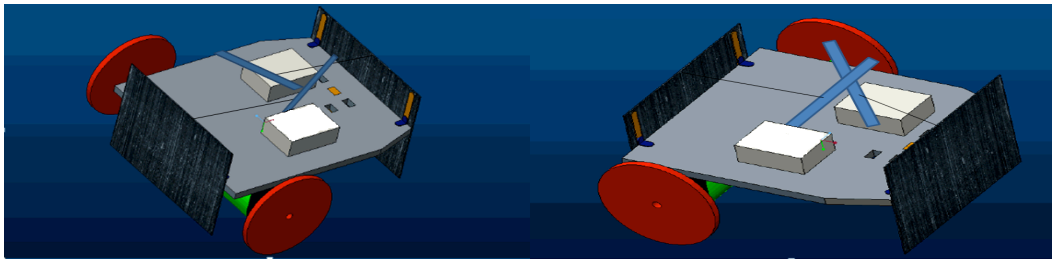

Figure 2(a) Figure 2(b)

Figure 2a and 2b show the design of the robot performed in the final competition. The first picture shows the rear view of the robot while the second shows its front view. These two sketches show the most important parts of the robot, such as the two wheels, a ball caster in the front (seen clearly in Figure 1), the motors (green cylinders underneath the platform), two servo motors (white rectangular boxes) and the rotating arms. The three reflectance sensors near the ball caster, the two reflectance sensors by each wheel, a color sensor and a contact switch attached to each of the plates are not shown.

Circuit Diagrams

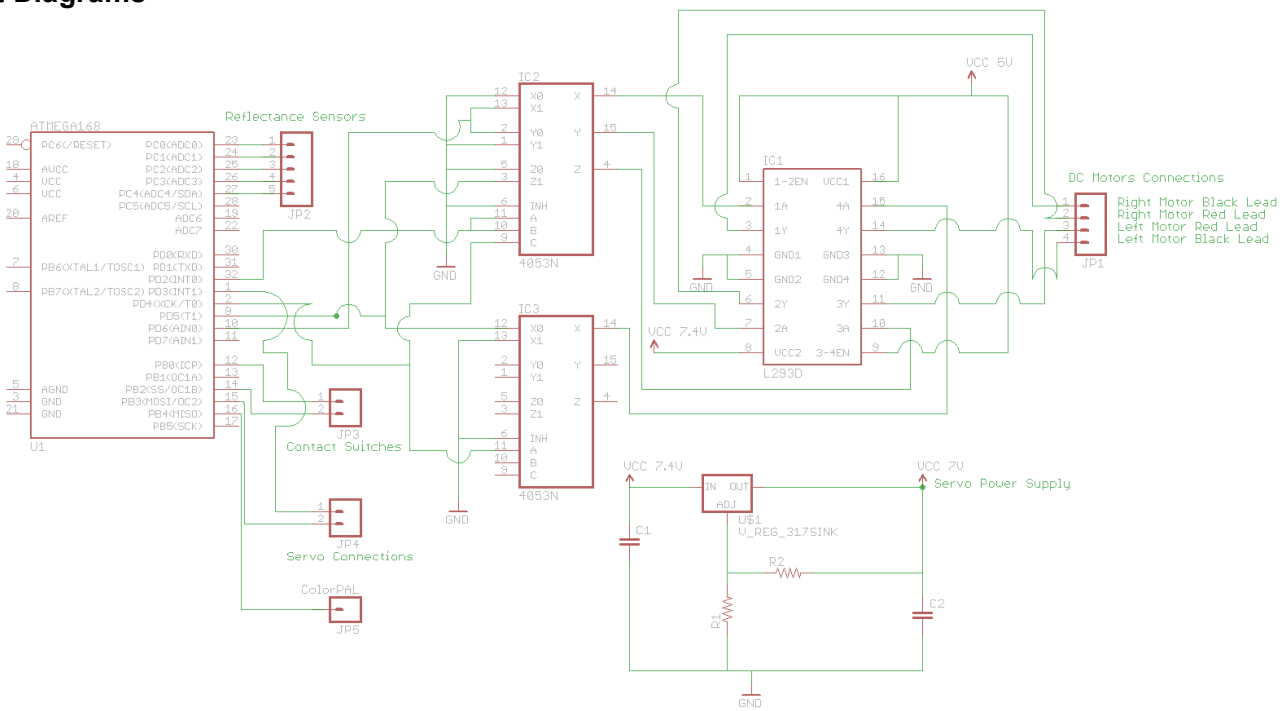

Figure 3: Overall schematic diagram of the robot, ATMEGA168 connections are connections to the Arduino

Photos of the robot

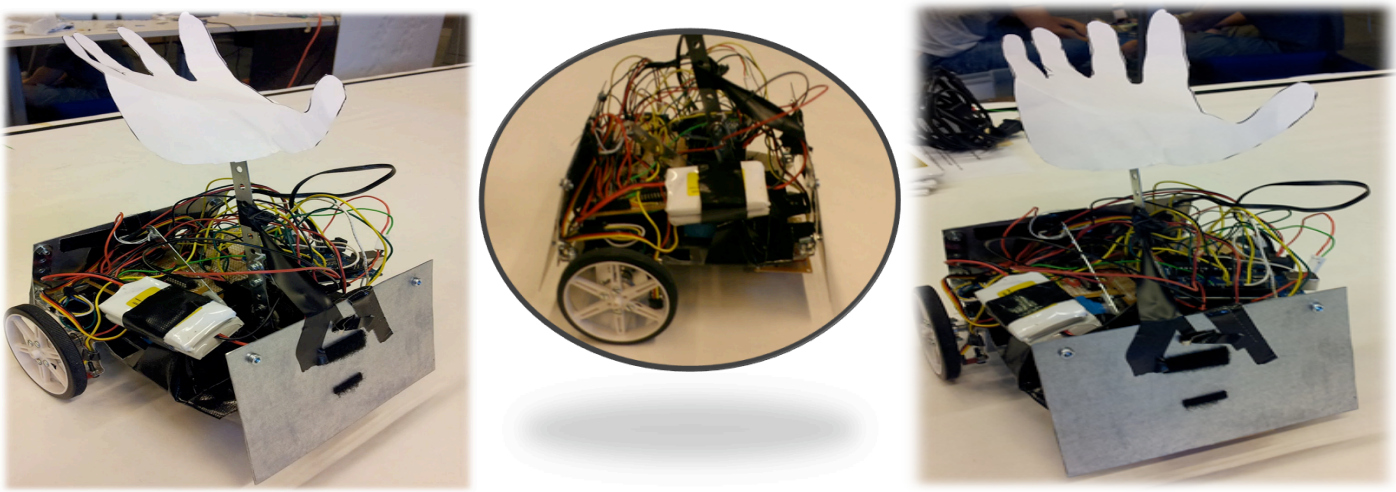

Figure 4a, 4b, and 4c: Front and side views of the robot