Design of a Behavior-Based Stair and Platform-Climbing Autonomous Robot

Erek Croteau
Mechanical Engineering Student
University of Massachusetts Lowell
Erek_Croteau@student.uml.edu

Samuel Toups
Computer Science Student
University of Massachusetts Lowell
Samuel_Toups@student.uml.edu

ABSTRACT

In this paper will discuss the design of an interactive behavior-based (BBR) autonomous stair and platform-climbing robot. The design revolves around the challenges faced in robotic elevation, complexity of environmental interactions and the Braitenberg theory. Multiple sensors are to be utilized to allow for logical operations which are decided upon by the robot itself. Here, we will explain what these challenges mean, what makes them difficult and how they were bridged into achievement. A robot in this case was developed successfully using just a CBC, Lego and Vex Mechanical parts, sensors and C-based programming.

Author Keywords

Behavior-Based Autonomous Robotics, Stair/Platform Climbing, Interactive Environment, CBC, Lego

INTRODUCTION

High-rising platforms are a major obstacle for the mobility of robots, however these days they are beginning to become no match to the rising technology and knowledge of robotic sensing input and action output. This project will develop a version of a climbing-type robot, particularly for platforms.

It is to be argued in this study that robots can be built to handle all kinds of obstacles, including the transitioning of various planar or even non-planar elevations. If there are existing variables in its environment that can create logic by stimulating the sensors to decide one action or the other, such as distance of walls and floors or unique color identifications, it can be told to react to them in some way unless an immediate change in them exists. For example, like in the study of author Dario Floreano's team on aerial locomotion in cluttered environments, a perceptual system must be capable of using logical behavior in "detecting obstacles in the robot's surroundings, including the ground, with minimal computation, mass, and energy requirements," as well as have a "flexible and protective framework capable of withstanding collisions and even using collisions to learn about the properties of the surroundings when light is not available" [1].

Based on the Braitenberg theory, it may be claimed that this sort of robot can represent a Vehicle 3 explorer-type robotic system due to its use of multiple sensors that can produce its various actions, however this may not fully be the case for the reason that it takes the sensing and actions of the sensing in sequence rather than all at once [2]. It uses vision to first or last find or place the object, uses ET sensors and top-hats to sense the surroundings of the walls and platform locations or drops, and then the sonar to detect heights of these platforms if going up.

This is an interesting project for the fact that it encompasses numerous sensors and step-by-step interactions between the sensory inputs and the action outputs, and it challenges the users to develop simplified models. The broader impact of this project is that it is that it provides a simplified model of robotic climbing using Lego System parts, sensors, and motors/servos in comparison with the more complex and advanced models illustrated in the real world today. Such a study of behavior-based and developmental robotics on plane-changing would apply in the real world to S. Jain's design team of forming a rescue bot using centroidal position control method for obstacle negotiation, or understand the challenges faced in the simplification of the stair climbing wheelchair, like the one developed by Juanxiu and his team, which already utilizes a closed-loop control system [3,4]. This advancement would even be effective for say, a robot that can clean the stairs, a next level of iRobot's Roomba.

PROJECT DESCRIPTION

Deliverable & Demonstration

The final product to be produced by the project is an environmental interactive robot that can utilize its sensors to determine platform heights and ends and be able to climb them, whether it's going up them or going down. This will include stairwells. It will also detect wall distances, and even pick up objects and move them to the desired locations that are above or below the platforms. The ultimate goal that the robot would demonstrate is its accuracy in the

ability to deliver an object from one level surface to another, overcoming the stair or platform obstacles in its way. This is demonstrated through use of a programmable CBC, allowing for it to be run logically on its own.

By using three ET sensors, the robot would be able to traverse around its environment and sense the obstacles around it. With one sensor up high and another down below in front of it, it would be able to detect a ledge that it could climb. The height determination was designed to be performed by the sonar sensor, at equal multiple small increments. The sensor reading would transpose to how high the lifter motors will move the robot up. Then, separate wheels with a drive motor would move the robot forward (in addition to the raising of each set of lifting legs). With the downward descending of steps, the robot needed to perform this while going backwards (due to the platform wheels). Two separate top hats would be used, front and back, the front one for initially detecting the drop.

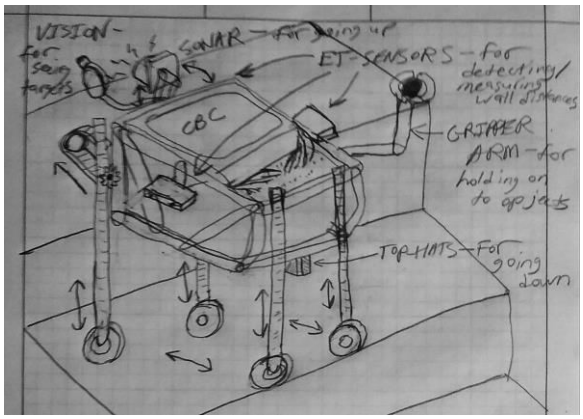

Figure 1. Initial Brainstormed Sketch of CBC Platform/Stair-Climbing Bot

How the robot was to perform its duty is shown by the step-by-step procedure in Figure 2.

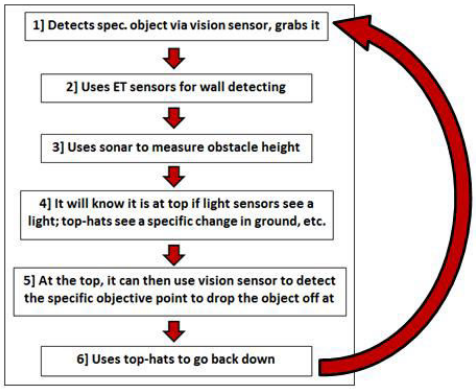

Figure 2. Step-By-Step System

The three deliverables that were involved for this project include:

Fri, April 1st (1st Checkpoint): Rough Design of Robotic Structure & Wall Detecting / Rough Platform Sensing Code

Fri, April 15th (2nd Checkpoint): Final Design of Robotic Structure & Platform Climbing Code

Fri, April 28th (Final Project Presentations): Full Code – addition of object detection, gripping, and delivering

Sensor-Action Background

Many of its sensors would create a specific reaction depending on not only what its thinking its next step is, but also what and how it's sensing based on the widespread environmental constraints. The robot will eventually get to the point where it not only senses the environmental structure, but becomes used to it. Human interaction is key mainly for allowing it to do its job and provide its needs.

Environment & Equipment Needed

The main environment that this project required was a simple stair well. Sensors that were mainly involved include an ET sensor, sonar sensor, top-hat sensor, and initially required 4 Hobbico Command CS-60 sport servos to use as motors and possibly up to 6 Futaba S3003 and Hobbico Command CS-61 standard servos at most. A Botball CBC v2 robotic controller was utilized as the main robotic brain, compatible with KISS-C interface software. Since the CBC has a limit of ports, the challenge was to drop these numbers of servos and motors down.

Structure

Lego was the prime tool to use for its wide variety of complex mechanics, such as different gear types and varying linkage approaches. If used properly, they are also relatively strong and simple to balance the center of masses of the system in using them. The Lego materials were adequate to the needs of construction if connected squarely or with enough hot glue. Problems we had with construction and a propensity to pop off under stress were related to the size of the platform constraints. Some things were going together at not quite right angles (because the base wasn't quite conducive to fit everything on a squared Lego beam grid).

The lifting legs would be designed to only be separate between front and back, since this was needed in order for the robot to retract between one set and then the other as it moves onto the platform.

Design Constraints

Constraints of this design include that the robot had to be able to lift itself up smoothly and level with the ground, be able to be compact enough to fit along the steps of the stairs, and also have an ability to maneuver around its environment regardless of there being platforms or no platforms to climb.

Evaluation of Results

Success was measured by accuracy between detection of platform heights, platform drops, and wall distances to its action of replicating the height amount, keeping away from the wall, and amount it will move past the platform edge.

The next measure of success was then the ability to be able to achieve a smooth transition to the next platform without falling or stumbling. A third one was the ability of accurately finding its targets (object and location to place it) and grabbing them successfully. A fourth would also be the measure of the part quantity and complex mechanics used.

ANALYSIS

Link to video of the final robot prototype climbing up the stairs: https://www.youtube.com/watch?v=I3JRNzRt1xw

Motor & Lifting Operation

For the beginning stage, the stair and platform climbing bot was essentially drafted to include working drive motors, lifter motors, and sonar sensor. The mechanisms had to first be proven that they would work well, including that the motors would lift the robot, and a starting code could work the sonar sensor.

Knowing the operating torque to be about 2.6 lbs-in (42 oz-in, or 3.06 kg-cm) per motor, it was determined that including at least two separate motors would be able to lift a ~2-3 lb. robot with 4 lifting legs, all with wheels so that it can maneuver no matter if the robot is on the ground or elevated above it. These motors were tested, and they proved to work smoothly thus far on the back lifters. Figures 2-5 show the 1st stage of the robot design as is initially built with the four operating lifters (two attached) and operating sonar sensor.

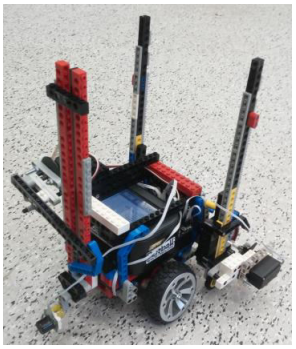

Figure 3. 1st Stage Stair Climber Bot Design

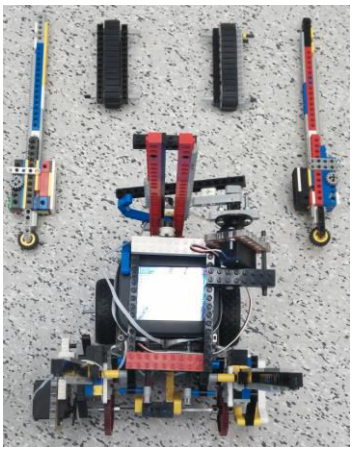

Figure 4. 1st Stage Stair Climber Bot Design with Full Components (Top View)

Sonar Sensing

Coding was developed in order to increment the sonar by a half-inch per reading for ten readings and create an action for it. Since the servo moves in rotational motion, a slider design was made so that it would move in linear motion. As for the incrimination, an initial value was used as the increment to measure the placements of the sonar linearly relative to the equal rotational increments. These increments were then calculated as sonar displacements from the linear measurements and plotted on a graph relative to actual sonar position, as shown by Figure 6. This was used to determine the incremental values needed for each step in order to achieve an equal linear displacement of ½ inch between each other.

For example, if the starting position is 320 (which in this case it is), we know from initial observation that it takes a value of 69 to move the sonar ½ inch up. Therefore, to determine the next incremental value, you would look at the plot at the point that matches with an actual sonar position of 320 + 69 = 389, and find that it is at right about 82, and so on. Once all the values are obtained, these are plugged into the code as a separate if-statement function.

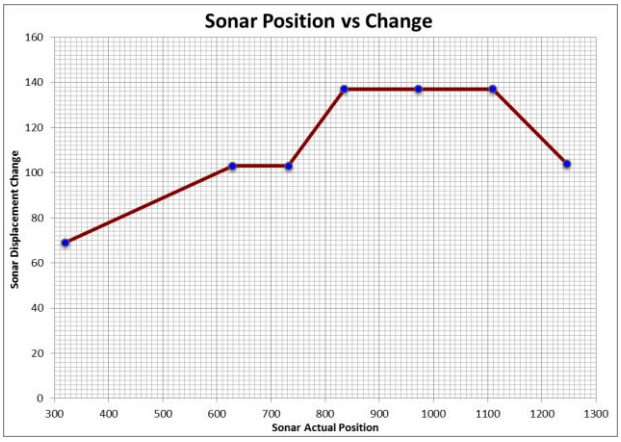

Figure 5. Graph Showing the Sonar Actual Position Relative to Servo vs the Value of Amount of Displacement of that Servo

With incremental values determined and implemented in the code, sonar data could then be transferred to the lifting motors, moving them at incremental values depending on the range between ground level and maximum achievable height. This maximum range value was determined to be roughly a distance of 1250, with a starting height value of 500. This meant that the increments would be set to be a size of about 50 for 12 increments.

2nd Phase Designing

The second stage design involved using four motors to lift the robot instead of just two. This is due to the fact that having one motor on one side and another on the other side turning the large lifting axles from a particular corner had proven to be an insufficient and difficult task when lifting the entire weight at the same time, even though they are rated to produce just enough torque to do so. This is especially difficult if the lifting legs also aren't stable enough. In addition, the four motors used were also able to be ported on just two motor inputs, two each, using specially designed connectors that put them in parallel (same voltage, divided current).

It had to be ensured that the center of gravity of the entire structure was towards the direct center as best as possible to give the greatest optimal lifting performance and prevent tilting or collapsing.

Figures 1-3 show the newly improved robot design with operating lifters and sonar sensor.

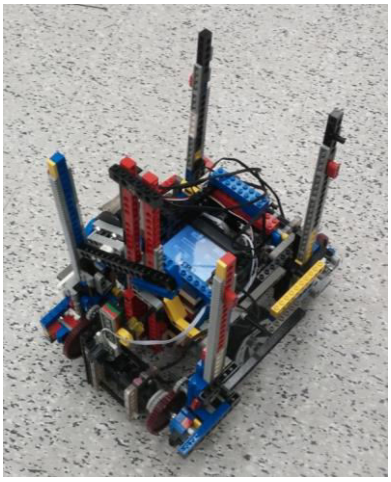

Figure 6. Improved Stair climber Bot Design

Motor Speed

Motor speed was vital in the functionality of the robot. A speed too slow was both not desired and did not fully allow for the robot to make it on the platforms when moving forward onto them. A speed too fast would mean parts would either break apart from each other or cause permanent damage to a specific component (such as a drive gear). Optimal speeds were found to be 50 ticks per second for lifting, 100 for driving onto a platform, and 500 for driving on the ground.

ET & Top Hat Sensor Implementation

A couple of both ET and top hat sensors were further added and integrated into the code to allow the robot to traverse around its environment, detect platforms, sense their proximity in the up-down direction, and go down stairs as well, rather than just up them. Three ETs were used as proposed, two on the sides angled at about 45 degrees, and one directly in the front. At this orientation, these sensors were able to pick up the walls ahead next to the robot, rather than just the walls directly next to it, quite like how peripheral vision works (hence why we have eyes in the front of our head).

With the top hats, values less than 30 showed to occur when it was directly above the dark ground being tested on. With platform drops, they tended to be above this value, since it goes beyond the readable 15 mm detection range. With the front one detecting the drop, it would turn around a complete 180 degrees and use the back one to detect it while it performs the drop sequence (similar to the rising sequence, only backwards procedure). However, this did prove to be inaccurate at times. Since they are more the less color dominant, the ET sensors work perfectly for platforms of different shades. However, in the real world, not all platforms are going to be like this, and since such sensors may initiate the drop sequence if it detected light shades on the ground, ET or sonar sensors may have been a better choice here. Either way, the program does allow for the robot to maneuver down a platform if sensed properly

Coding

As far as the final development of the coding, functions and loops were utilized to ensure that the robot performed specific actions based on what was observed. For example, it would proceed to do a sonar height-detection reading only if it actually sensed the platform via sonar and front ET sensor.

When it came to mechanical slack in the gears during lifting and forward moving as it climbed the platforms, this showed to be a major factor to consider in the code values to develop accurate and smooth movement. Therefore, some values in the lifter position were made to overshoot the normal position values due to this slack. Much of this "slack" however is related to the weight of the robot itself, since it would take more muscle to lift the weight to the desired position. This is also due to the variation between the motor lifting the weight initially, and the motor then raising the lifters once the weight is supported.

DISCUSSION

Developing this robot has found to be a much more challenging feat than previously anticipated. The initial idea

was to have a target object that was to be acquired and taken up a flight of stairs. However, that proved to be too ambitious, so the objective was modified to just have the robot detect stairs which it would ascend and descend. We did not actually explore the ability to locate a staircase from a random point in the room, however, based on prior experiments performed in this laboratory, this should not be difficult to achieve.

Our demonstration had our robot placed on a trajectory toward a set of steps (a standard staircase). It moved forward, and then determined the height of the first step. It then raised itself up. At the time of the demonstration, it usually required a few nudges because it would not be quite close enough to lower the front wheels onto the step. The original plan had been to make something which would fold down to pull the robot forward, but at the time of demonstration, we only had the stationary forward wheel mounts. Another solution would be to have wheel mounts which could extend slightly, pull the robot forward, and then be retracted after use.

In any case, with a nudge forward to place the front wheels onto the step, the robot could retract the front lifters and continue pulling itself onto the step and then retract the rear lifters. At this point, one elevation is finished.

The robot barely fits onto one stair riser. This caused some trouble as at one point the robot decided it was too close and started trying to move away from the "wall" which caused it to roll off the step. The best success achieved was climbing two steps in a row.

Due to our choice of sensors the robot was unable to detect when it was approaching a drop off and would just try to roll off the edge. We were unable to fix that at the time of the demonstration. Therefore, we were unable to demonstrate descending action.

SUMMARY & CONCLUSION

This study is made to emphasize autonomous sensor-action behavior based robotic design. Since platform climbing is one of the most difficult challenges that are typically faced with everyday robotics, it has been seen as a significant tool of challenge. By utilizing four Hobbico Command CS-60 sport servos to lift the robot under two ports, the robot was able to lift itself up and move onto each elevated platform or step. Though all tasks in this design were not fully achieved in the time given (such as the ability to pick up objects, etc.), this work and the results developed will allow for future reference in developing behavior-based action by autonomous mobile robots.

For future studies of this case, the next phase would be to bring the robot to more of a developmental (epigenetic) style complexity, relating to the field of evolutionary robotics. For future studies in this particular design, using a more improved structure should be taken to consideration, and simplification of mechanics should be re-visited.

ACKNOWLEDGMENTS

Between the team of two in developing this autonomous mobile robotic design, the specific objectives of each of us through the duration of the project can be noted by Table 1. Erek has the specialty with ensuring things work, fulfilling optimal accuracy in actions and the majority of coding, while Samuel is the main mechanical and structural components developer and integrator for the robot. Design brainstorming, constructing and troubleshooting occurs between the both of us combined.

	Erek	Samuel
April 1	Develop starting code for ET and sonar sensors; ensure sensors/motors will work out with the different mechanisms	Work out some complex mechanical parts of the robot; work out modifications to 1st draft code
April 15	Work out/ensure platform/stair-climbing accuracy and exertion of these actions; advance the code	Ensure a good and working final design of the structure and its mechanics; advance the code
April 28	Develop final interactive coding of all sensors and object detection	Develop final interactive coding of all sensors and object detection

Table 1. Deliverables Per Individual

The work described in this paper was conducted as part of a Spring 2016 robotics course, taught in the Computer Science department of the University of Massachusetts Lowell by Prof. Fred Martin.

Special thanks to Professor Martin and the CS department for providing their helpful advice and dedication.

REFERENCES

1. Floreano, Dario, et al. "Aerial Locomotion in Cluttered Environments." *Laboratory of Intelligent Systems*. Proceedings of the 15th International Symposium on Robotics Research, 2011.
2. Braitenberg, Valentino. Vehicles: Experiments in Synthetic Psychology. MIT Press, 1986.
3. Jain, S., et al. "Analytical approach for obstacle negotiating capability of stair climber." *Proceedings of the 2015 Conf. on Advances In Robotics*. ACM, 2015.
4. Liu, Juanxiu, et al. "High-Order Sliding Mode-Based Synchronous Control of a Novel Stair-Climbing Wheelchair Robot." *Journal of Control Science and Engineering* 2015 (2015).

APPENDIX A:

FULL CODE USING KISS & CBC BOTBALL SOFTWARE

```
/*************************************
 Erek Croteau; Samuel Toups
 STAIR AND PLATFORM-CLIMBING BOT
 Test Code Rev. 4 (4/29/16)
*************************************/

/* This program will measure platform
heights using a sonar sensor and
platform drops using top hat sensors,
then react by elevating itself to the
desired height. It will also be able to
traverse itself throughout its
environment using ET sensors, in
addition to the sonar and top hats. */

// PORT IDENTIFY
     // Motors
const int LDMotor = 0; //Motor port 0 =
                       right drive motor
const int RDMotor = 3; //Motor port 3 =
                        left drive motor
const int FEMotor = 1; //Motor port 2 =
                       fnt. elev. motors
const int BEMotor = 2; //Motor port 1 =
                       back elev. Motors
     // Servos
const int SonarServo = 0;
     // Sensors
const int ETR = 4;      //Right ET sensor
const int ETL = 5;      //Left ET sensor
const int ETF = 7;      //Front ET Sensor
const int Sonar = 1;    //Sonar sensor
const int TopHatFront = 6; //Inner
                  (Back) Top-Hat sensor
const int TopHatBack = 0; //Outer
                 (Front) Top-Hat sensor
const int Bumper = 15;  //Bumper sensor

// CONSTANTS
const int BumperOn = 1;
int c = 0;              //turn counter
const int Dir = 1; /* direction setting
                   (default = 1,forward)
                For backwards direction,
                          make it -1 */

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

// FUNCTIONS
//void BotMove(int ? );
//void BumperHit();
int SonarScan(void);
int SonarIncrement(int iVal, int inc);
void BotHeight(int SonarPosition);

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

// MAIN PROGRAM
int main()  {
     int i = 0;
     //comparator variable
set_each_analog_state(1,1,0,0,1,1,1,1);
    //set ports to floating
sleep(0.02);
     //wait for state to change
enable_servos();
set_servo_position(SonarServo,1460);
    //preset sonar servo port
printf("Servo is at position 320 \n");
int largestDistanceIndex;

while(1){   //loop forever

   // MENOUVER AROUND THE FLOOR

   int FS = analog10(ETF);
   int LS = analog10(ETL);
   int RS = analog10(ETR);

   //int ETFreading = analog(ETF);
   int SONARreading = analog(Sonar);
   int FTH = analog10(TopHatFront);
   int BTH = analog10(TopHatBack);

   printf("FS = %d , LS = %d , RS = %d ,
SON = %d\n", FS, LS, RS, SONARreading);

   if(FS <= 800 && LS <= 300 && RS <=
300) {                    //if far
      mav(RDMotor,500);    //go straight
      mav(LDMotor,500); }

   else if(FS <= 800 && LS <= 400 && LS
> 300 && RS <= 300) {  //if left wall
                          somewhat close
      mav(RDMotor,300);  //move slight
                           right
      mav(LDMotor,500); }

   else if(FS <= 800 && LS > 400 && RS
<= 300) {           //if left wall close
      mav(RDMotor,-500); //move sharp
                           right
      mav(LDMotor,500); }

   else if(FS <= 800 && LS <= 300 && RS
<= 400 && RS > 300) { //if right wall
                         somewhat close
      mav(RDMotor,500);  //move slight
                           left
      mav(LDMotor,300); }
```

```
   else if(FS <= 800 && LS <= 300 && RS
> 400) {           //if right wall close
      mav(RDMotor,500);//move sharp left
      mav(LDMotor,-500); }

   else if(FS > 800 && SONARreading > 8)
{ //if front wall close & no platform is
detected
      mav(RDMotor,-250); //move back and
                          turn around 45
                         degrees to left
      mav(LDMotor,-250);
      sleep(1.0);
      mav(RDMotor,500);
      mav(LDMotor,-500);
      sleep(1.0); }

   else if(FS > 800 && SONARreading <=
8) { //if front wall close & platform is
       detected

      //Initiate sonar scan
      sleep(0.8);

      while(FS > 800) {

            mav(RDMotor, 0);
            mav(LDMotor, 0);

            int HeightVal = SonarScan();

// RAISE THE BOT TO THE PROPER HEIGHT
            BotHeight(HeightVal);
// Also Lowers Front End Slightly After
Being Raised

            int dist = HeightVal * 50;

// LOWER PLATFORM ARM
          //set_servo_position(1, 1000);
          //sleep(3.0);

// MOVE BOT SLIGHTLY FORWARD
          //set_servo_position(2, 2047);
            mav(RDMotor, 100);
            mav(LDMotor, 100);
            sleep(1.0);
            mav(RDMotor, 0);
            mav(LDMotor, 0);

// RAISE FRONT LIFTERS
            mrp(FEMotor,-200,-500-dist);
            sleep(10.0);

// MOVE BOT SLIGHTLY FORWARD
          //set_servo_position(2, 2047);
            mav(RDMotor, 100);
            mav(LDMotor, 100);
            sleep(15.0);
            mav(RDMotor, 0);
            mav(LDMotor, 0);
            //disable_servos();

// RAISE BACK LIFTERS & ARM
            //set_servo_position(1, 0);
            mrp(BEMotor,-200,-750-dist);
            sleep(10.0);

//  MOVE  BOT  FORWARD  UNTIL  WALL  IS
DETECTED
            while(analog10(ETF) <= 400){
          //set_servo_position(2, 2047);
                  mav(RDMotor, 100);
                  mav(LDMotor, 100);
                  }

// RECHECK FOR LEDGE
            FS = analog(ETF);
      }
   }

   else if(FS <= 800 && FTH >= 30) {
//if platform drop detected

//Initiate drop sequence
      mav(RDMotor,-250); //move back and
                          turn around 180
                          degrees to left
      mav(LDMotor,-250);
      sleep(2.0);
      mav(RDMotor,500);
      mav(LDMotor,-500);
      sleep(2.0);

      mav(RDMotor,-250);
      mav(LDMotor,-250);
      sleep(2.0);

      while(BTH >= 30){
         mav(RDMotor,-100);
         mav(LDMotor,-100);
         sleep(3.0);

// LOWER BACK LIFTERS
         mrp(BEMotor, 50, 550 + dist);
         sleep(10.0);

// MOVE BOT SLIGHTLY BACKWARDS
         mav(RDMotor,-100);
         mav(LDMotor,-100);
         sleep(5.0);

// LOWER FRONT LIFTERS
         mrp(FEMotor, 50, 500 + dist);
         sleep(10.0);
```

```
// MOVE BOT SLIGHTLY BACKWARDS
         mav(RDMotor,-100);
         mav(LDMotor,-100);
         sleep(2.0);

// LOWER BOT
            mrp(FEMotor,-50,-600-dist);
            mrp(BEMotor,-50,-700-dist);

// RECHECK FOR LEDGE
            mav(RDMotor,-100);
            mav(LDMotor,-100);
            BTH = analog(TopHatBack);
            sleep(3.0);

            if(BTH < 1000){
               break;
            }
         }

         if(BTH < 30){
            mav(RDMotor,500);
            mav(LDMotor,-500);
            sleep(2.0);
         }
      }
   }
}

/*************************************

int SonarScan(void)
{
// SAMPLE & RETRIEVE SONAR DATA
set_servo_position(SonarServo, 320);
int i = -1;
int increment = 0;
int max = analog(Sonar);
printf("Sonar is at position %d: %d\n",
i, max);
msleep(200);
int reading;
int maxDir = -1;

for (i = 0; i < 12; i++) { //For max of
                           10 positions
 set_servo_position(0, 320 + increment);
 /*perform at varying increments
   starting at 320 that will equal ~1/2
   inch; use SonarIncrement function */

   msleep(50);
   reading = analog(Sonar);
   printf("Sonar  is  at  position  %d:
%d\n", i, reading);

   if(reading > max) {
      max = reading;
      maxDir = i;
   }

   msleep(500);
   increment = SonarIncrement(i,
                              increment);
}

printf("Max  Value  is  %d  at  Position
%d\n", max, maxDir);

//  RESET  SERVO  &  IDENTIFY  LARGEST
DISTANCE VALUE
set_servo_position(0, 1460);
int largestDistanceIndex = maxDir;
return largestDistanceIndex;
}

/*************************************

int SonarIncrement(int iVal, int inc) {
   if (iVal == 0)
     {
      inc = inc + 70;
     }
     else if (iVal == 1)
     {
      inc = inc + 50;
     }
     else if (iVal == 2)
     {
      inc = inc + 60;
     }
     else if (iVal == 3)
     {
      inc = inc + 60;
     }
     else if (iVal == 4 || iVal == 5)
     {
      inc = inc + 100;
     }
     else if (iVal == 6)
     {
      inc = inc + 140;
     }
     else if (iVal == 7)
     {
      inc = inc + 150;
     }
     else if (iVal == 8)
     {
      inc = inc + 130;
     }
     else if (iVal == 9)
     {
      inc = inc + 120;
     }
     else if (iVal == 10 || iVal == 11)
```

```
      {
       inc = inc + 80;
      }
      else
      {
       inc = 0;
      }
      return inc;
}

void BotHeight(int SonarPosition)  {
   // DETERMINE DISTANCE VALUE
   int distance = SonarPosition * 50;

   // MOVE MOTORS TO PROPER HEIGHT
   mrp(FEMotor, 50, 500 + distance);
   mrp(BEMotor, 50, 550 + distance);
   //mrp(FEMotor, 50, 1000);
   //mrp(BEMotor, 50, 1050);

   sleep(25.0);

   // MOVE FRONT DOWN
   mrp(FEMotor, -50, -100);
   sleep(5.0);
}
```

APPENDIX B: ROBOT DESIGN SEQUENCE PHOTOS

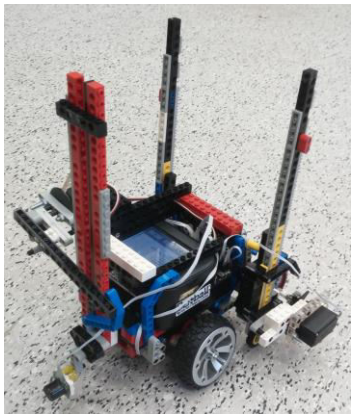

Figure B1. 1st Stage Stair Climber Bot Design

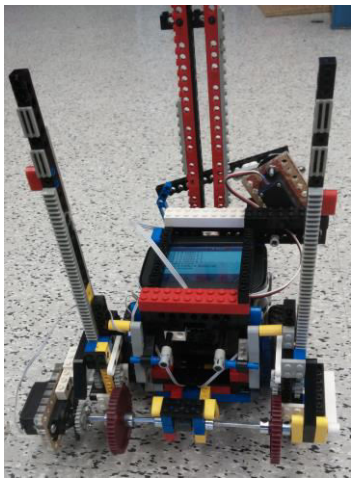

Figure B2. 1st Stage Stair Climber Bot Design (Front View)

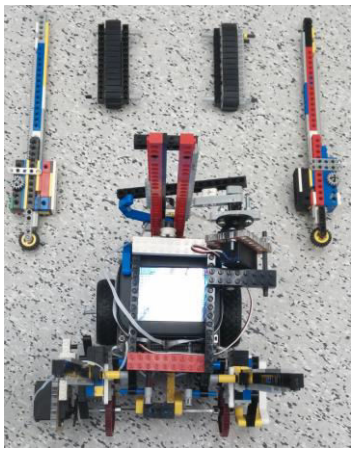

Figure B3. 1st Stage Stair Climber Bot Design with Full Components (Top View)

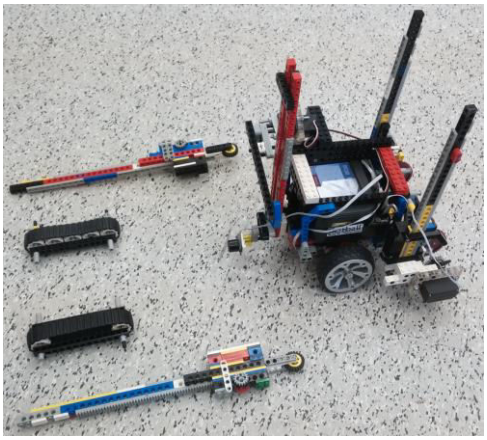

Figure B4. 1st Stage Stair Climber Bot Design with Full Components (Side View)

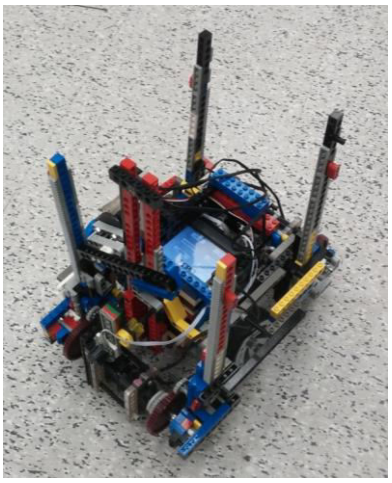

Figure B5. Improved Stair Climber Bot Design

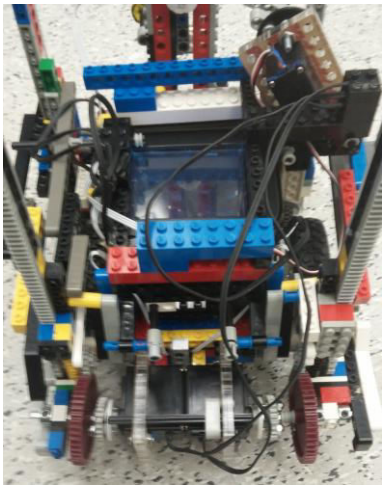

Figure B6. Improved Stair Climber Bot Design (Back View)