RICH

"Relatively Intelligent Coin Harvester"

EEL5666 – Intelligence Machine Design Lab

Summer 98

By Charles Fedderwitz

Table of Contents

Abstract

The RICH project is an autonomous robot capable of detecting, retrieving and depositing coins of various sizes. Its compact modular design adds to its high autonomous maneuverability and simplicity. It is designed to continuously search and remove coins from a smooth surface with high levels of success.

Executive Summary

The RICH project is an autonomous robot. RICH behaviors include collision avoidance, bump detection, coin calibration, search, detection, retrieval and collection. RICH has a unique body unlike any other robot. The design is compact, simple but functional. Collision avoidance is accomplished by two IR detectors located directly in front for left/right collision avoidance and two to the front sides for maneuvering out of tight areas. Since RICHs maneuverability is restricted forward bumper detection is only required on the front-end. Coin detection is accomplished using a Non-ferrous/ferrous metal detector purchased from Radio Shack. The detector is mounted to the top of the robots control box and the input signal is tapped from the op-amp output from the coil. No variations were required of the metal detection circuitry. After a coin is detected, a bucket located in the rear of the robot accomplishes acquisition. The robot moves forward until the coin detected is pushed against the bucket and the coin is then lifted and deposited in a compartment located in front.

Software capabilities included all behaviors just discussed and a search algorithm. The search algorithm is contained within the collision avoidance process. After a coin is calibrated, if the threshold value is within 10% of the rated value a search of the area is performed. The search entails the robot circling the area back and forth meanwhile monitoring its detection value. If the coin threshold is reached the coin is retrieved otherwise in time the robot will give up and continue.

Introduction

One major limitation was required when first beginning a robotics project over the summer C semester. Unlike other semesters the summer session was substantially shorter. Due to these time constraints the intention was to produce a robot which had some practical capability and also was not mechanically burdened. As a result the RICH project evolved. The RICH project was conceived as a learning tool for children. An exciting way for them to recognize different coin denominations while being awed by RICH itself. The RICH project's capabilities were designed explicitly to locate coins, but are not limited to this task. Practical uses can include locating items of varying metallic masses. Though RICH's retrieval capabilities are limited now, further implementation can be used to find more explicit ways of coin retrieval dependent upon the area being searched. Another possibility could include searching vast areas for lost metallic possessions such as gold necklaces and rings.

The paper is divided into four categories. The Brains, a discussion on the HC11 micro-control unit and its required expansion. The Body is a discussion on specifications and implementation of the design structure. The Looks states all sensor report results and the Attitude states all behaviors assigned.

THE BRAINS

68HC11E9 Micro-controller

The Motorolla MC68HC11E9 is an 8-bit Microcontrol unit. It contains an 8MHz clock with a 2MHz bus. The controller only offers 256 bytes of random-access memory (one reason for the expansion using the ME11 board discussed later). The HC11 also offers several on-chip peripheral capabilities useful to the functionality of RICH. They include an eight-channel analog-to-digital converter with 8-bit resolution, a 16-bit free-running timer system with three input-capture lines, five output-compare lines, and a real-time interrupt function.

ME11 Expansion Board

The Mekatronics ME11 board is built for expansion of the Motorolla HC11 explicitly for enhancing autonomous control. The ME11 memory maps an extra 32kbytes of random-access memory for greater programming capability. It provides 8-signal output ports supplying a 40kHz signal for proper IR response. The ME11 also contains a motor driver chip capable of handling two 12VDC powered motors. The H-bridge circuitry allows the driver to control both direction and speed of the motor.

Other Expansion Required

Although the ME11 board offered a majority of the required implementation, further requirements were needed. The expansion area on the HC11 provided the necessary location. The expansion included a second motor driver chip, servo control, and circuitry for analog inputs, bumpers and contact switches. The overall realization of the expansion requirements for RICH is exhibited in figure 1.

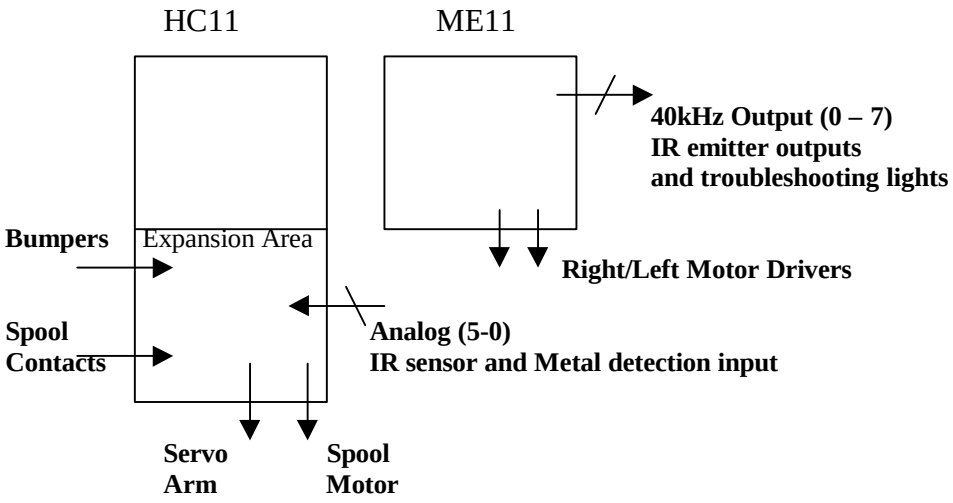


Figure 1

In order to get consistent function from both motors and sensors, compromises had to be made in both programming and implementation. Figure 2 shows the resulting port assignment.

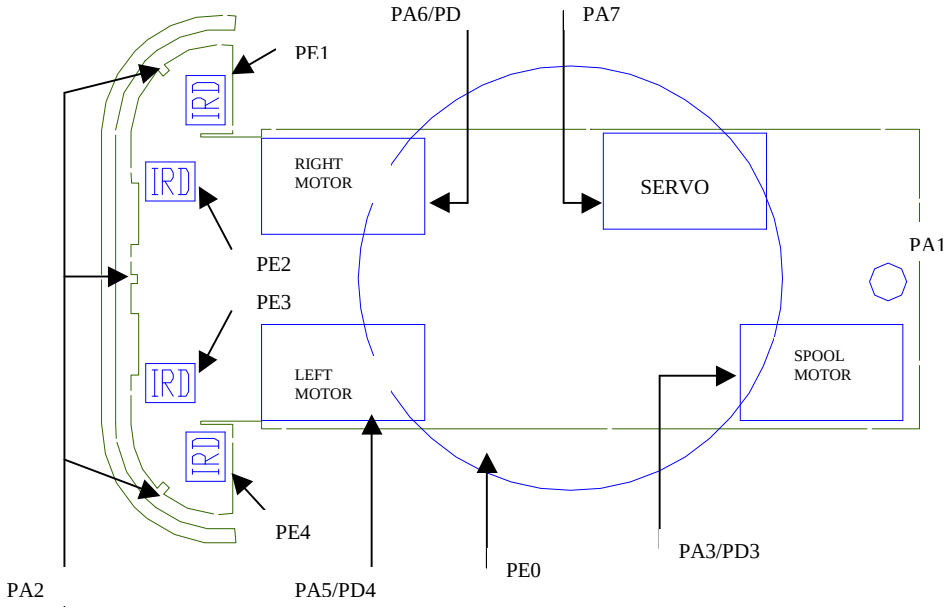


Figure 2: Port assignments of both motors, servo, and sensors

THE BODY

The hardest mechanical capability required of RICH was realizing the acquisition of a body design could be determined. After several different ideas for retrieving a coin were considered, the idea of using a scoop or bucket was decided. This implementation was chosen but required restrictions on the area of acquisition. The area would have to be smooth allowing the bucket and coin to slide freely. Other considerations had to be determined also. Size and weight constraints of the bucket. How the bucket would function to "scoop" the coins? And where would they be deposited?

Bucket

In order to reduce the amount of mechanical devices on the bucket itself, a spring tensioned door was pulled back by a string attached to the spool motor. When the partition was completely open a contact switch would indicate its fully open position. The coin would then be placed in the area between the scoop and the partition and the partition would be released allowing the door to close and pushing the coin into the bucket.

The bucket length is kept small by reducing the area of coin placement. This can be accomplished by allowing the robot to push the coin up to the bucket before acquisition. The body is compact in order to reduce the required bucket arm length. After acquisition of the coin, it is lifted over the body and deposited in front. By reducing the arm length and the area of acquisition, one servo is able to handle the required weight.

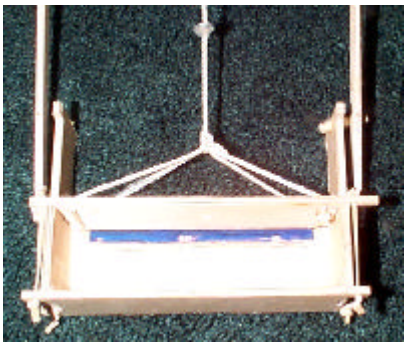


Figure 2a – The bucket in the normally closed position.

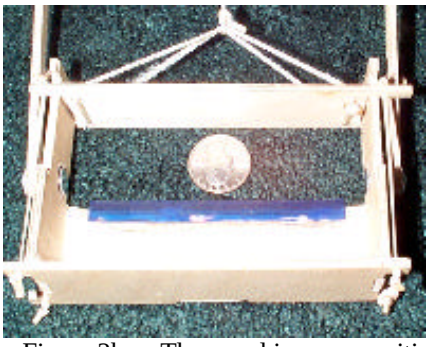


Figure 2b – The spool in open position with spool motor on.

Design and Production

The body design was then created using AutoCAD14. Appendix A shows the actual AutoCAD drawings. The drawings were fitted on two 12 x 18 inch templates which were then cut on 1/8 inch plywood sheets using the T-Tech Etching Machine. Making RICH as compact as possible, contributed to a lower amount of materials required in the overall body's construction.

Design flaws

After completion of the body, several adaptations were made to the design specifications. The supports created to house the two front motors were too weak. The front motors tended to bow towards the body causing the wheels to rub. In order to remedy this effect, two rods were mounted between the motors to add further support as shown in figure 3a. Also as the bucket would rotate to meet the front, the bucket would tend to over-rotate preventing the coins from finding their mark. To prevent this a mechanical limiter was molded to the bucket to prevent excessive rotation as shown in figure 3b.

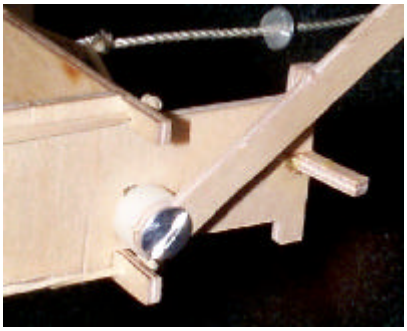


Figure 3a – Mechanical limit

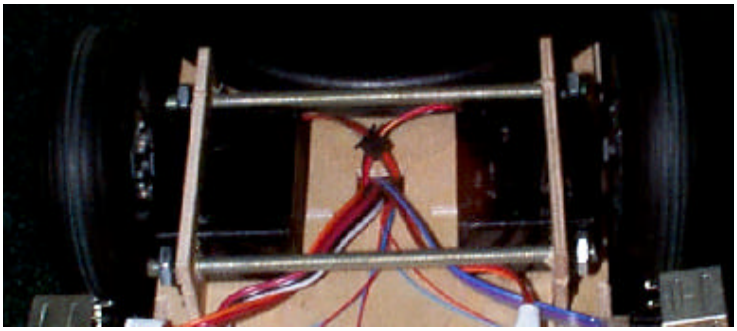


Figure 3b – Support rods for motor mounts

THE LOOKS - Sensor report

Four different sensors are implemented in the design of RICH, IR proximity sensors, a metal detector, bumper switches and a motor limit contact switch.

Wall Proximity Sensors

The wall proximity sensors consist of four sensors mounted in the front of the body. Two sensors are located in the front and one to each side for proper corner response. Two tests were performed on the front IR Sharp sensors. "RICH" the robot was placed at 6inch intervals from the wall from 6 to 36 inches. For each emitter/sensor pair, the signal response received by the EVBU was recorded. The test was performed on a dark rug and the first emitted response was reflected off a white surface. The second test was performed under the same circumstances except the reflective surface was black. The room was fairly lit for both tests. The first test shows the response produces a steady varying output as the robot is removed further from the surface. The second test shows immediate results from 6 to 10 inches but any further away sensor seems non-responsive.

Readings from Left and Right Front IR Sensors Against a White Surface in a Moderately Lit Environment

Dist from wall (in)	***2***	***4***	***6***	***8***	***10***	***12***	***14***	***16***	***18***	***20***	***22***	***24***	***26***	***28***	***30***	***32***	***34***	***36***
Right Analog (0)	131	131	132	131	128	124	120	117	114	114	112	109	107	102	101	99	96	94
	131	131	132	131	128	124	120	117	115	114	112	110	107	103	102	100	97	94
	131	131	132	131	128	123	120	117	114	114	111	109	106	102	102	100	98	96
Left Analog (1)	130	130	126	131	115	110	105	102	100	99	98	96	95	93	93	91	90	88
	130	129	126	120	115	110	106	102	100	99	98	96	95	93	93	92	90	89
	130	129	126	120	115	109	105	102	99	99	98	96	95	93	93	92	91	90

Figure 4a

Readings from Left and Right Front IR Sensors Against a Black Surface in a Moderately Lit Environment

Dist. (in) from wall	***2***	***4***	***6***	***8***	***10***	***12***	***14***	***16***	***18***	***20***	***22***	***24***	***26***	***28***	***30***	***32***	***34***	***36***
Right Analog (0)	129	119	112	105	103	102	101	100	102	102	102	101	101	97	96	96	95	93
	129	122	114	106	103	101	101	100	102	103	102	102	98	97	96	95	92	91
	124	121	117	107	104	102	102	100	102	102	102	101	97	97	96	94	92	91
Left Analog (1)	112	105	98	95	94	93	93	92	94	94	93	93	91	90	90	89	88	87
	112	106	99	96	93	92	92	94	94	93	93	92	91	90	90	89	89	87
	114	105	101	96	93	93	93	92	94	94	93	93	91	91	90	89	88	87

Figure 4b

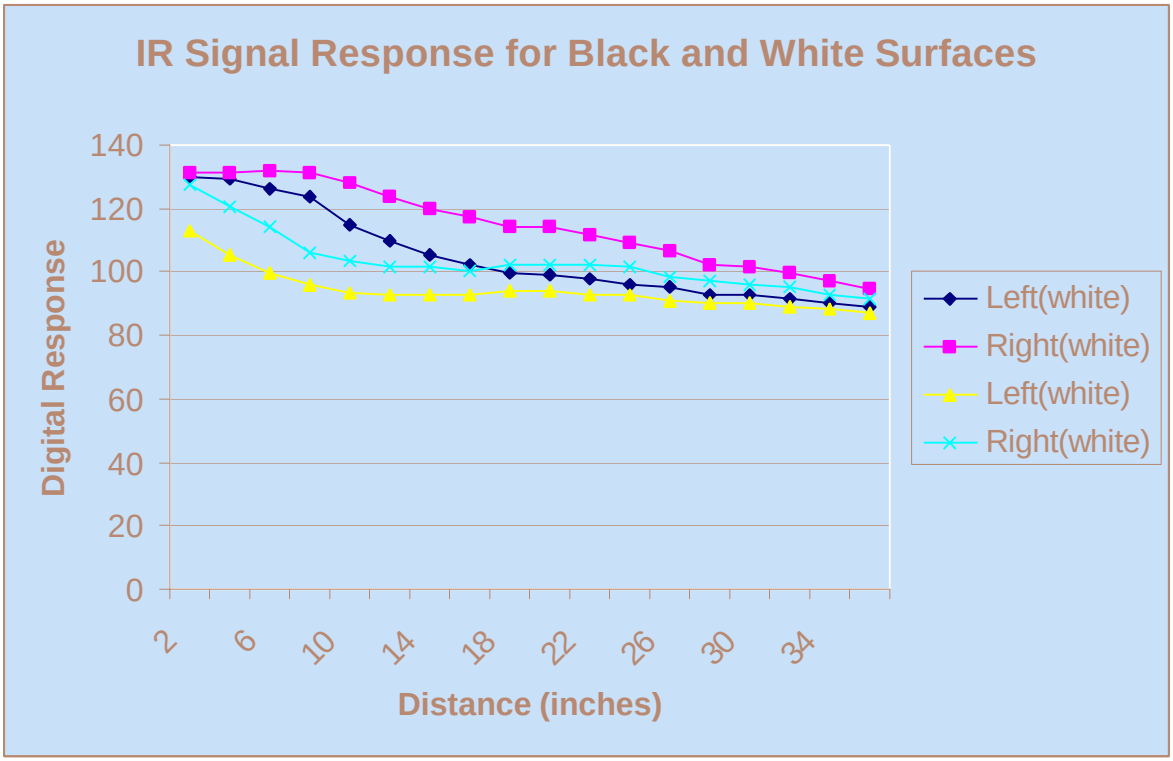


Figure 6

Metal Detection Sensor

The test performed for metal detection is a basis for possible coin recognition capabilities. A quarter was placed at the center of the coil. The three parameters involved in metal detection are volume, tune, and discrimination control. Volume control is self-explanatory. Tuning varies the output gain response received by the coil, and Discrimination control sets the coil sensitivity. The test settings were calibrated on a quarter centered beneath the coil. The Discrimination was set to a minimum sensitivity and the Tune control was set until a signal was audible. Three trials were performed on each coin denomination at varying lengths from center. The results show the quarter as the most responsive while the other three denominations show similar outputs.

Inches from Center		*0.0*	*0.25*	*0.5*	*.75*	*1.0*	*1.25*	*1.5*	*1.75*	*2.0*	*Coin Removed*
Quarter	Trl-1	33	32	28	26	24	20	15	11	8	3
	Trl-2	31	31	31	28	27	25	21	16	12	9
	Trl-3	28	25	24	21	18	15	10	6	3	0
Nickle	Trl-1	11	11	10	10	9	9	9	8	6	6
	Trl-2	12	11	10	9	8	8	6	5	3	2
	Trl-3	9	8	8	6	6	5	4	3	1	0
Penny	Trl-1	18	17	18	16	14	12	11	9	8	6
	Trl-2	15	14	12	11	10	8	7	5	4	2
	Trl-3	14	13	13	12	11	10	8	6	1	0
Dime	Trl-1	16	16	15	15	14	13	12	9	7	3
	Trl-2	11	11	10	9	8	7	5	4	3	0
	Trl-3	12	11	10	10	8	5	4	4	0	0

Figure 7

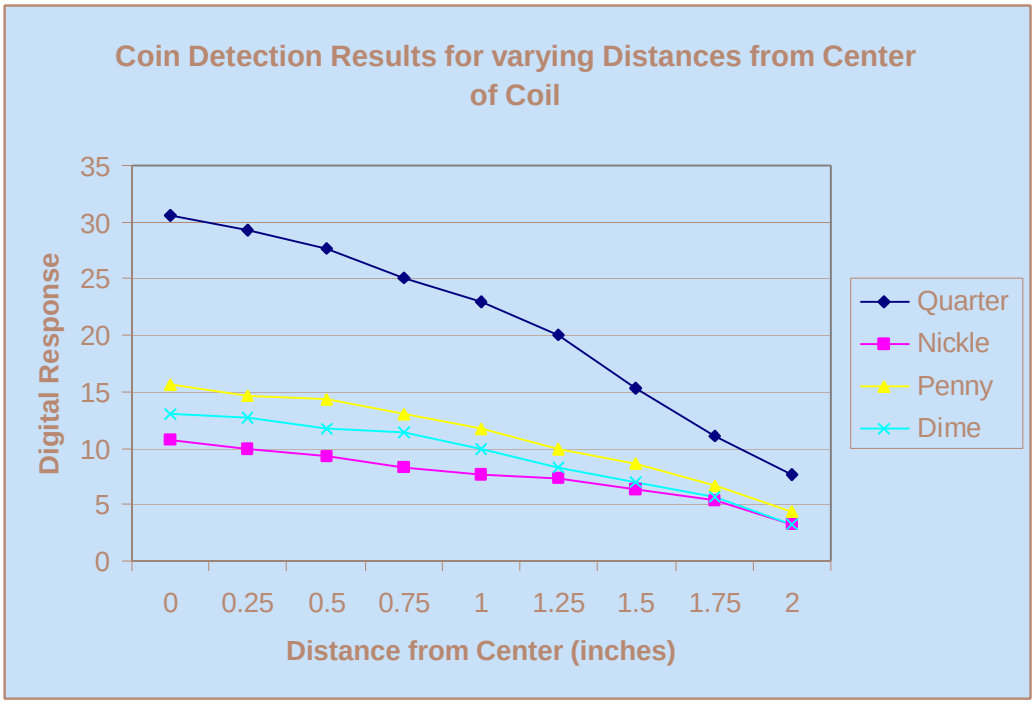


Figure 8

Metal Detection Test Procedure

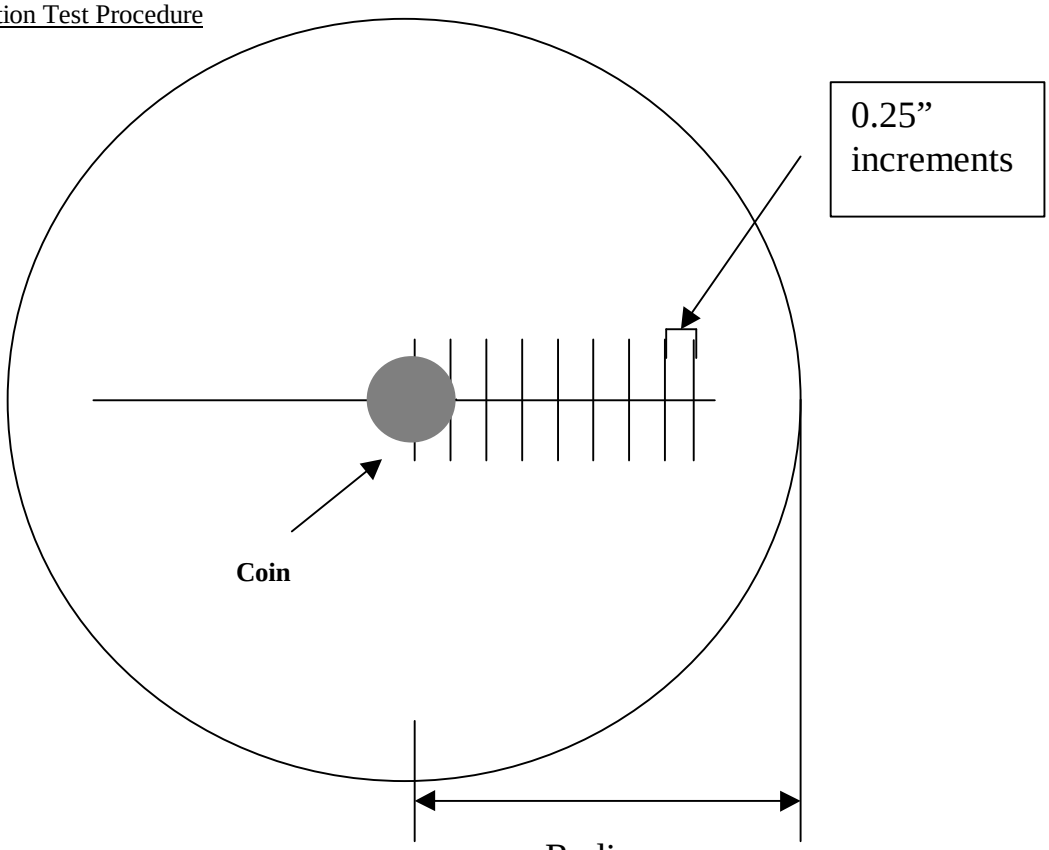


Figure 9

Bumper Switches and Contact Limit Switch

The bumper circuit contains three momentary switches all parallel connected. The basic circuit is shown in figure 10. The contact switch is not unlike the bumper switch. The difference is a metal contact completes the circuit instead of a momentary switch

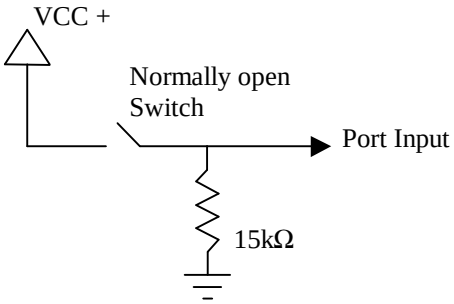


Figure 10 – Basic contact circuit configuration.

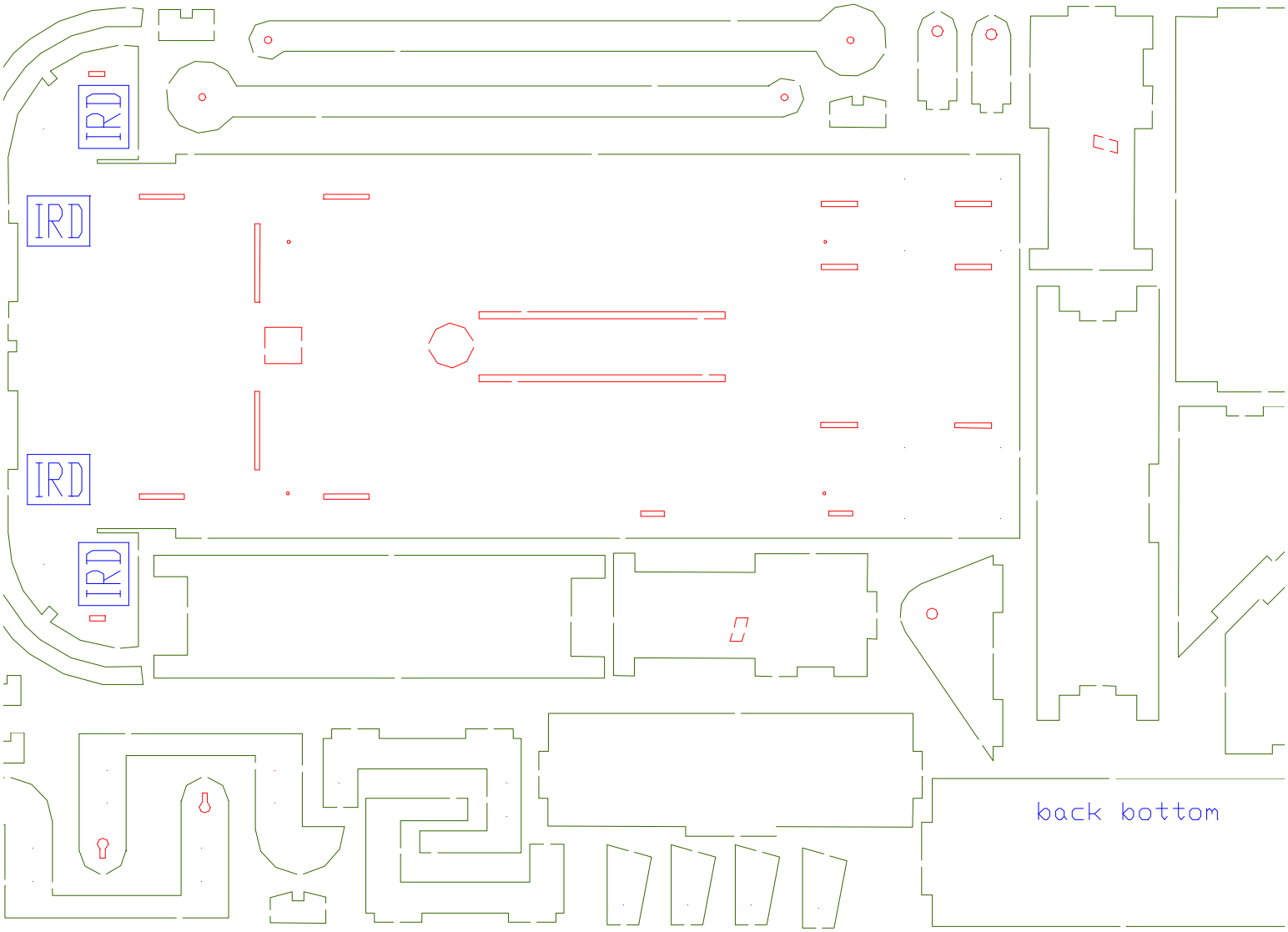
THE ATTITUDE - Behaviors

Responsive behavior is characterized by the algorithms in C. The file Rich.c found in appendix B is the responsive autonomous program, which is compiled in Interactive C. Several libraries were required to implement autonomous behavior. File lib_rw11.c contains basic ME11 board functions, which includes motor function. In order to implement the third spool motor this file was altered to include motor initialization and function to a third motor. Also, the servo library files servo.icb and servo.c were required in order to control servo function on Porta7. Programming restrictions are required when controlling the servo and a motor at the same time. Because motor function requires TOC1 to stay zero and servo function requires this value to change, motor function is limited when the servo is running. When running, motors can only be run at 100% capability. In order to return the motors to their normal function, TOC1 must be zeroed this is accomplished by performing a "poke" function on addresses 0x1016 and 0x1017 and clearing these locations.

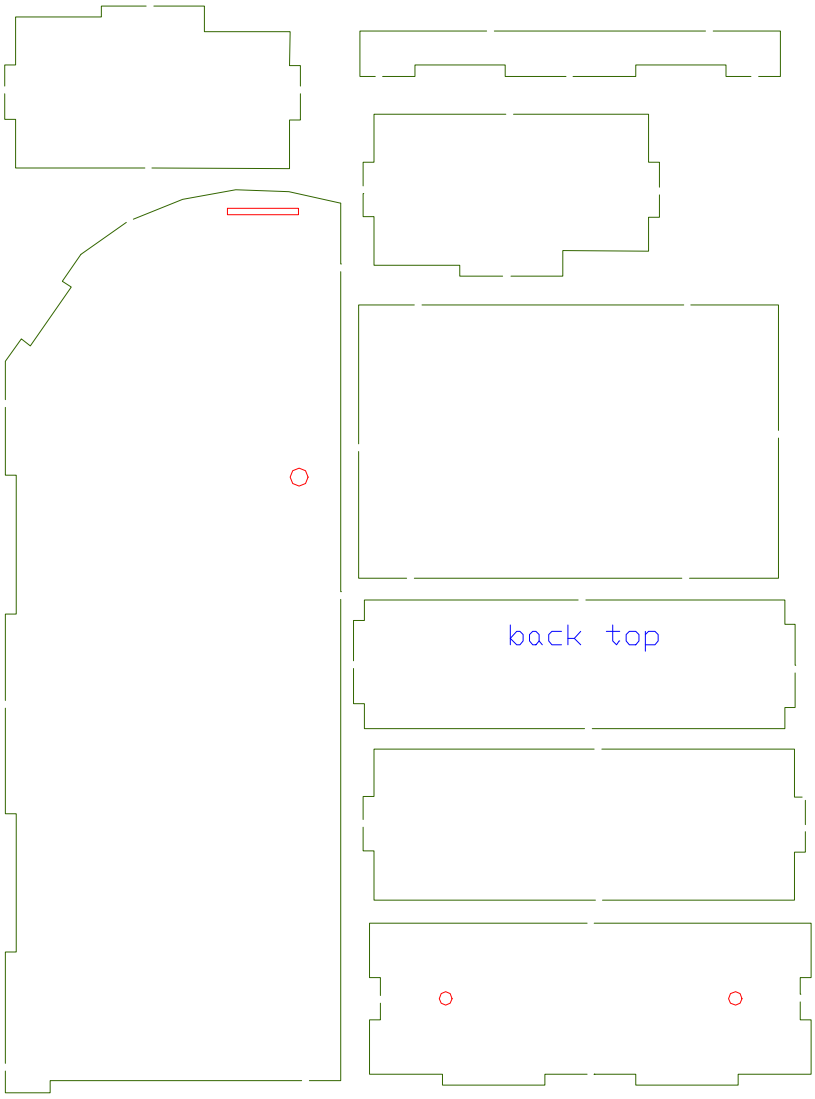
CONCLUSIONS

The robot accomplished all functions it was intended to do. There were only minor problems in structural design, which included a mechanical limit and motor support. Otherwise the design was successful. The bucket though capable of retrieving a coin with legitimate success does seem to function haphazardly. Future works could include a stronger bucket design with greater support. The programming could be critiqued similarly. Although all required functions were there, further refinement would greater enhance its performance. The collision avoidance though highly successful showed some monotonous movement that could have used some further intention. These comments

could be based on all the programming algorithms. Overall the largest reason for not refining these further were simply based on time constraints of the summer semester.



APPENDIX A


back top

APPENDIX B

```
/* GLOBALS */

/*Variables*/
int leftc, rightc, lcd, rcd;
int lefts, rights, lsd, rsd;
int left = 1;
int right = 0;
int coin;
int count;
int position;
int give_up;
int go;
int reference;
int check;
int bumped;
int bumper_mask = 0b00000001;
int bucket_mask = 0b00000010;

void coin_deposit() {
position = 1000;
servo(1000);
servo_on();
while(position < 2600) {
      motor(2,-5.0);
      sleep(0.06);
      stop();
      servo(position);
      position = position + 50;
      }
sleep(1.5);
servo(4500);
sleep(1.5);
servo_off();
poke(0x1016,0x00);
poke(0x1017,0x00);
}

void arm_over(){
position = 4500;
servo(position);
servo_on();
while(position > 1000) {
      position = position - 1500;
      servo(position);
      motor(2,5.0);
      sleep(0.5);
      stop();
      }
servo_off();
poke(0x1016,0x00);
poke(0x1017,0x00);
}

void pick_up(){
servo(1300);
```

```
servo_on();
sleep(0.5);
servo(1250);
sleep(0.5);
motor(2,10.0);
reset_system_time();
while ((seconds() < 2.0) || (peek(0x1000) & bucket_mask) == 0);
servo(950);
sleep(0.5);
servo_off();
poke(0x1016,0x00);
poke(0x1017,0x00);
sleep(0.5);
motor(2,-5.0);
sleep(1.0);
stop();
}
void coin_calibrate(){
  count = 0;
  coin = analog(0);
  while (count < 3){
      poke(0x7000,0x80);
      sleep(0.5);
      poke(0x7000,0x00);
      poke(0x7000,0x10);
      sleep(0.5);
      poke(0x7000,0x00);
      count++;
      }
  count = 0;
  poke(0x7000,0x01);
  while (analog(5) < 130){
      if (coin < analog(0)) {
          coin = analog(0);
          poke(0x7000,0x81);
          sleep(0.5);
          poke(0x7000,0x01);
          }
  }
  poke(0x7000,0x00);
  while (count < 3){
      poke(0x7000,0x10);
      sleep(0.5);
      poke(0x7000,0x00);
      count++;
      }
  count = 0;
}

void coin_find(){
       give_up = 1;
       if ((coin - 5) < analog(0)){
           stop();
             give_up = 0;
       }
       else if ((coin - 15) < analog(0)){
                stop();
```

```
                search();
        }
        if (!give_up){
              while (count < 3){
             poke(0x7000,0x90);
                 sleep(0.5);
             poke(0x7000,0x00);
                 sleep(0.5);
             count++;
            }
              arm_over();
              sleep(1.0);
              motor(left,25.0);
              motor(right,25.0);
              sleep(1.75);
              motor(left,-10.0);
              motor(right,-10.0);
              sleep(0.25);
              stop();
              sleep(1.0);
              pick_up();
              coin_deposit();
      }
      count = 0;
}

void avoid()
{
      /*reference reading of IR sensors*/
      poke(0x7000,0xff);
      wait(100);
      leftc = analog(3);
      rightc = analog(2);
      lefts = analog(5);
      rights = analog(1);
      poke(0x7000,0x00);

  while(1){
      coin_find();
      poke(0x7000,0x0f);
      wait(100);
      lcd = analog(3) - leftc;
      rcd = analog(2) - rightc;
      lsd = analog(5) - lefts;
      rsd = analog(1) - rights;
      poke(0x7000,0x00);
        if (lcd > 14 && rcd > 12 ) {
             rich_back();
                 wait(1500);
             if (lsd > rsd){
                         motor(left,20.0);
                         motor(right,-20.0);
                   wait(500);
             }
             else {
                         motor(left,-20.0);
                         motor(right,20.0);
```

```
                wait(500);
            }
      }
      else if (lcd > 14) {
                motor(left, 20.0);
                motor(right, -20.0);
                wait(1000);
      }
        else if (rcd > 12) {
                motor(left, -20.0);
                motor(right, 20.0);
                wait(1000);
      }
      else {
                motor(left, 30.0);
           motor(right,30.0);
      }
  }
}

void search(){
        reset_system_time();
        go = left;
        reference = analog(0);
        check = reference;
        while (seconds() < 20.0){
          if ((coin - 5) <= analog(0)){
                stop();
                give_up = 0;
                break;
          }
          else if (analog(0) >= (check-1)){
                if (go == left){
                   motor(left, 10.0);
                   motor(right,-10.0);
                }
                else {
                   motor(left, -10.0);
                   motor(right, 10.0);
                }
                wait(500);
          }
          else{
                stop();
                check = analog(0);
                if (go == right)
                    go = left;
                else go = right;
          }
          if (go == left && check != reference){
                while((coin - 10) >= analog(0)){
                    motor(left, 10.0);
                    motor(right, -10.0);
                }
                break;
          }
       }
```

```
}

void rich_back(){
       motor(left, -25.0);
       motor(right, -25.0);
}

void wait (int milli_seconds)
{
  long timer_a;
  timer_a = mseconds() + (long) milli_seconds;
  while (timer_a > mseconds())
        defer();
}

void bump(){
  if ((bumper_mask & peek(0x1000)) == 2)
       rich_back();
}

void main()
{
    coin_calibrate();
    start_process(avoid());
    start_process(bump());
}
```