JR

A Roving Metal Detector

IMDL Final Report

Justin Cobb
TA's: Scott Jantz,
Aamir Qaiyumi, Patrick O'Malley
April 21, 1999

Table of Contents

Abstract

JR is a roving metal detecting robot. It uses sonar, IR, and CDS cells to determine where to move around, and a metal detector to find objects to dig up. JR uses a conveyor belt with scoops attached to unearth the objects found.

Executive Summary

The purpose of JR was to create a robot that would detect metal objects and dig them up. This has been done before in IMDL, but JR was to be a more robust platform. The initial goals for this robot were for it to be able to dig in sand as well as moderately packed dirt.

There are four types of sensors used on JR: sonar, IR, CDS, and a metal detector. Sonar and IR are used for obstacle avoidance. The CDS cells are used to navigate JR towards a beacon. The metal detector performs the most important function of the robot, locating the objects that are to be dug.

JR uses tracks from a remote controlled toy. The rest of the platform was built from scratch. A platform was made of plexiglas to mount most of the electronics. A frame was constructed on this platform, and the conveyor belt was attached to this. The conveyor belt stretched from the front of the metal detector to the rear of the robot.

The controller for JR is a Motorola 68HC11 EVBU board with the Mekatronix ME11 attached. The 40kHz outputs from the ME11 board are used to control the IR and sonar emitters. The ME11 also controls the drive motors and the servo that raises and lowers the conveyor belt. Analog inputs on the 68HC11 interpret signals from the CDS cells, IR, sonar, and metal detector. The microprocessor also uses the port D outputs to control the drive motor for the conveyor belt.

JR navigates using its sensors and uses the metal detector successfully. The only drawback is that the robot cannot actually dig up dirt. The scoops are stitched on with wire, but only bend when the conveyor belt is lowered into dirt.

Introduction

JR is a metal detecting robot. It uses Infrared (IR) and sonar for collision avoidance. CDS cells are used to follow a light source, and a metal detector finds objects to be dug.

The original purpose of JR was to be set down on the beach so it could roam around, using the metal detector, and dig up any objects it saw. This would cut down on human time used doing this very activity. It was quickly discovered that the most difficult part of designing a robot to do this is dealing with the harsh environment. For this reason, JR is only intended to dig in sand, nothing that is well packed. Also, there is a cover for the electronics compartment to protect it from falling sand.

This report includes information about the integrated system, mobile platform, actuation, sensors, behaviors, and experimental layout and results of JR. These sections describe the various parts that went into the final design of this metal detecting robot. All documentation, code, and parts lists are at the end of the report.

Integrated System

JR uses the Motorola 68HC11 EVBU board with a Mekatronix ME11 used for more memory, 40kHz output ports, and on board motor drivers. JR was programmed using ICC11 v5 for Windows.

Theory of Operation

JR uses a moderately basic program to control operation. When first turned on, the system monitors the IR receivers and the metal detector to set thresholds. After the thresholds are set, the robot begins moving forward. While roving, all four sensor-systems are being used. IR and sonar are used for collision avoidance and CDS cells are used to move towards a beacon. IR is only used as a last ditch effort to avoid running into objects because the IR emitted by the sun is usually too high to rely on IR normal collision avoidance. The program polls the metal detector for valid readings. The order of importance for the sensors is:

1. Metal Detector
2. Sonar (Collision Avoidance)
3. IR (Collision Avoidance)
4. CDS cells (Navigation Toward a Beacon)

The flowchart for JR can be seen in Figure 1.

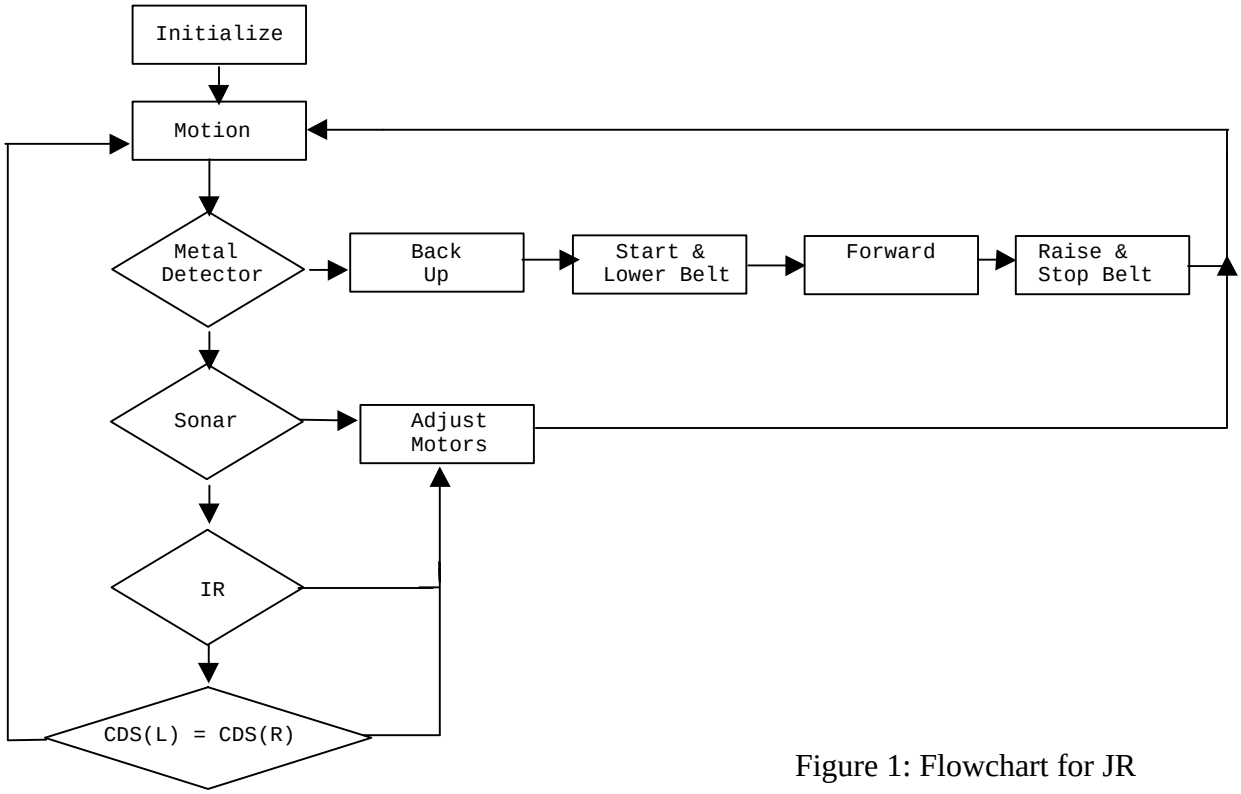

Figure 1: Flowchart for JR

The electrical connections for JR are in Table 1.

IR Emitter (Left)	ME11 pin 7
IR Emitter (Right)	ME11 pin 6
Sonar Emitter (Left)	ME11 pin 1
Sonar Emitter (Right)	ME11 pin 4
IR Receiver (Left)	Analog 0
IR Receiver (Right)	Analog 4
Sonar Receiver	Analog 7
CDS (Left)	Analog 1
CDS (Right)	Analog 2
Metal Detector	Analog 3
Left Motor	ME11 Motor 0
Right Motor	ME11 Motor 1
Belt Motor	Port D 3
Servo	Servo 0

Table 1: Electrical Connections

The only system not implemented on JR for demonstration was the beacon locating function. The code for this function is written and functioning, but since the demo for the robot was to be done outside, and the CDS cells were calibrated for very little light (this was to be done at night, so the sun would not interfere with the light readings) this option was taken out of the code.

Mobile Platform

Because one of the goals for this project was to make a robust robot, there is a lot of detail in the construction of JR. The basic plan for the robot is a platform mounted on top of a set of tracks. Connected to the platform is a frame that supports the conveyor belt system. The conveyor belt spans the length of JR; it digs in front of the metal detector, and dispenses the dirt behind the robot.

Tracks

The tracks for JR come from a remote controlled bulldozer called "Excavator." The rest of the toy was ripped apart, with the hope that other motors or gears may be used, but that was not possible. To provide enough strength to move the platform, the motors and gears used to drive the Excavator were gutted and two 48 oz. in. hacked servos were epoxied directly to the drive axle for each side (Figure 2)

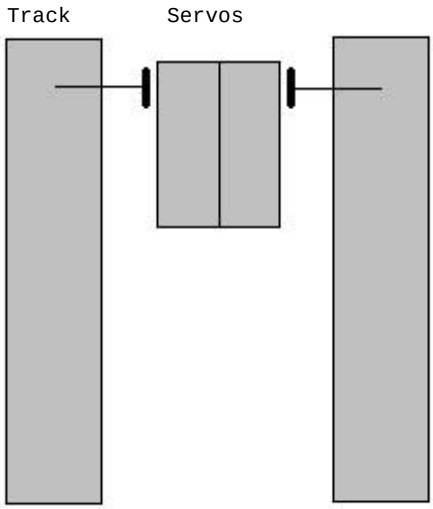

Figure 2: Drive Servos

Platform

The electronics for JR are mounted on a platform that is attached above the tracks. This platform is a 9" by 10" piece of plexiglas, surrounded by 0.75" aluminum angle, for strength. Also mounted on this platform is a frame made of 0.5" aluminum angle, which supports the conveyor belt. Mounted to the front of the frame is an unhacked servo, which raises and lowers the conveyor belt, with the help of two springs, mounted to the back of the frame, to offset the weight of the conveyor belt.

Conveyor Belt

The most difficult aspect of JR was the conveyor belt. This would have to be strong enough to dig, but also light enough so the robot would be able to move at all. The frame of the conveyor belt is constructed of two 2" by 36" by 0.125" pieces of aluminum bar. Pieces of 0.25" all-thread and nuts are used to keep the spacing between the two bars at 4". At the front and rear of the frame, bearings were epoxied to the aluminum for the axles. Bicycle sprockets were attached to the axles. Two lengths of bicycle chain were stretched across the sprockets, one on each side. Attached between the chains was a narrow length of shower curtain. A piece of plastic window molding was cut into 5" pieces. These pieces were stitched to the shower curtain every 9" to act as scoops.

At about the middle of the conveyor belt lengthwise, on the right side of JR, a rectangular hole was cut in the aluminum to house the drive motor. Another sprocket was glued on the servo, to drive the chain. This assembly was mounted in a way so the sprocket would touch both the top and bottom section of chain, to provide maximum force, and reduce skipping of the chain. Also to reduce skipping, a chain tensioner/derailer had to be fashioned. It consisted of a piece of all-thread bent at a 90° angle twice that had a small sprocket attached to it. The all-thread was bolted to the aluminum frame of the conveyor belt. The sprocket was lined up with the chain and a spring was used to keep it tight.

Since the conveyor belt was entirely too large to be lifted by the 100 oz. in. servo attached to it, two springs were added in the rear to offset the weight in the front. These springs were strong enough to keep the belt at almost a horizontal position at equilibrium, while not too tight for the servo to lower it to a digging position.

Attached to the front of the conveyor belt are two pieces of plexiglas, one at each end, bent to 60° angles. The sonar and IR sensors are attached to these, to keep them as close to the front of the robot as possible. Since the conveyor belt is held at a horizontal position while in motion, these sensors 'see' straight forward.

Actuation

The actuation of JR is another major concern. Basic servos and DC motors would not be strong enough to move this robot, due to its weight. For the drive motors, two 48 oz. in. servos were hacked to DC motors. These motors drew a stall current of 0.55 A each. They were attached directly to the motor driver on the ME11 board.

To raise and lower the conveyor belt, a stronger servo was needed. Here, a 120 oz. in. servo was used, connected to an output compare line of the 68HC11 board. To offset some of the weight of the conveyor belt and help the servo, two springs were attached to the back of the belt.

Finally, the strongest actuator on the entire robot had to be the one to drive the conveyor belt. A 306 oz. in. hacked servo was used here. It was attached to a motor driver, made simply of a TIP transistor, connected to the Port D 3 pin. A sprocket is attached to this servo, which turns the chain for the conveyor belt.

Sensors

The sensors used on this robot platform are used for only two purposes: navigation and detecting metal. The sensors used for navigation are infrared (IR), sonar, and CDS cells. The IR and sonar are being used for object avoidance while the CDS cells are used to make the robot move towards some beacon. For detecting metal, obviously a metal detector will be used.

Infrared (IR)

The IR sensors on the robot are being used for object avoidance. A 40 kHz signal is sent to an IR led, this signal is bounced off of anything in front of the module, and then the signal is then observed by the Sharp can. The level of the output increases as the robot moves closer to any objects, as shown in Figure 3.

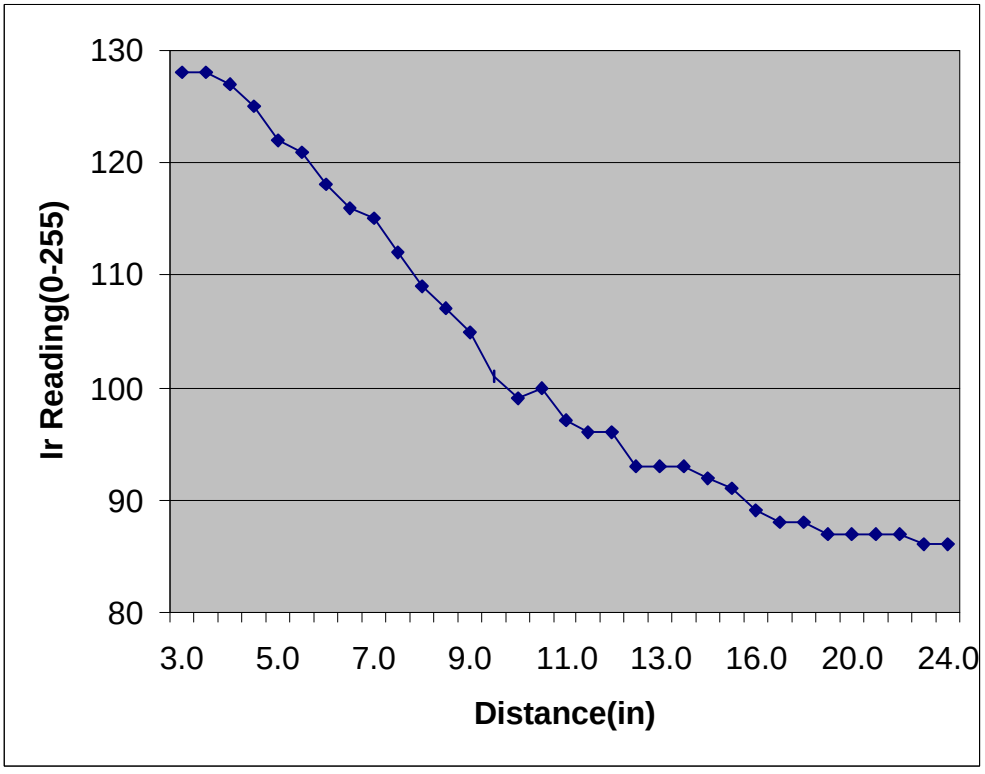

Figure 3

These readings were taken in a dark room, with the signal being reflected off of a white, glossy textbook. Because of the IR emitted by the sun, it is not feasible to use IR outside for any great distances. However, for uses inside, it appears to be a very reliable system. Because of this, IR will only be used as a last ditch measure to avoid obstacles.

Sonar

As with the IR system, the sonar on this robot is being used for collision avoidance. However, the sonar is using time of flight, instead of strength of signal to measure distance. The software the microprocessor uses sends out a 1 µsec pulse to an emitter. The software then polls a receiver until it receives the pulse back, while incrementing a counter. This counter is used to interpret the distance from the object, using the speed of sound and how long it takes to go through the loop. The data observed in the sonar test is in Figure 4.

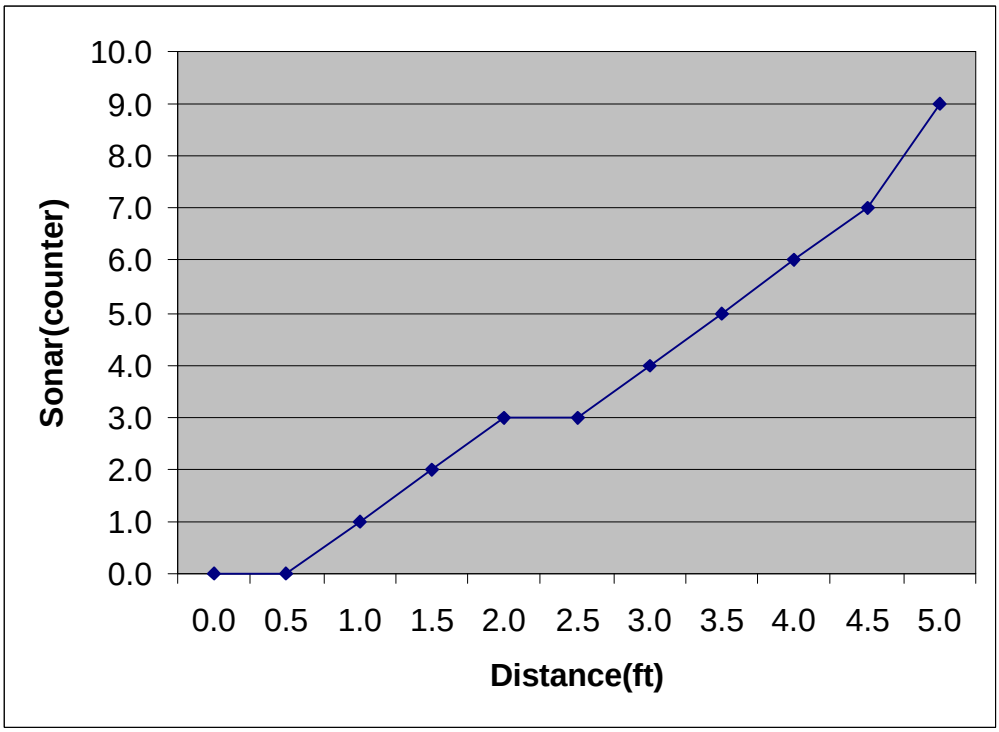

Figure 4

The data shown ignores noise that was seen. There were many readings of zero and 1000. These readings will have to be ignored, and any other readings taken will need to be averaged to have any data that is usable. Also, this data was taken using Interactive C. By the time the final project was done, JR was programmed with ICC11. The values of the counter were different, but the same accuracy of the data was observed. The valid values of the counter are now between 5 and 70. Because of noise, any readings below 5 and above 100 are ignored.

The circuit used for the Sonar was borrowed from William O'Connor's report for Ranos. The only difference is the orientation of the transformer for the emitter is reversed. (Figure 5)

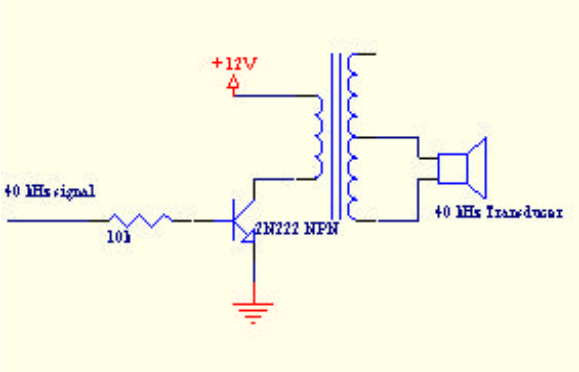

Figure 5

The receiver circuit is exactly the same one used in the Ranos project. (Figure 6)

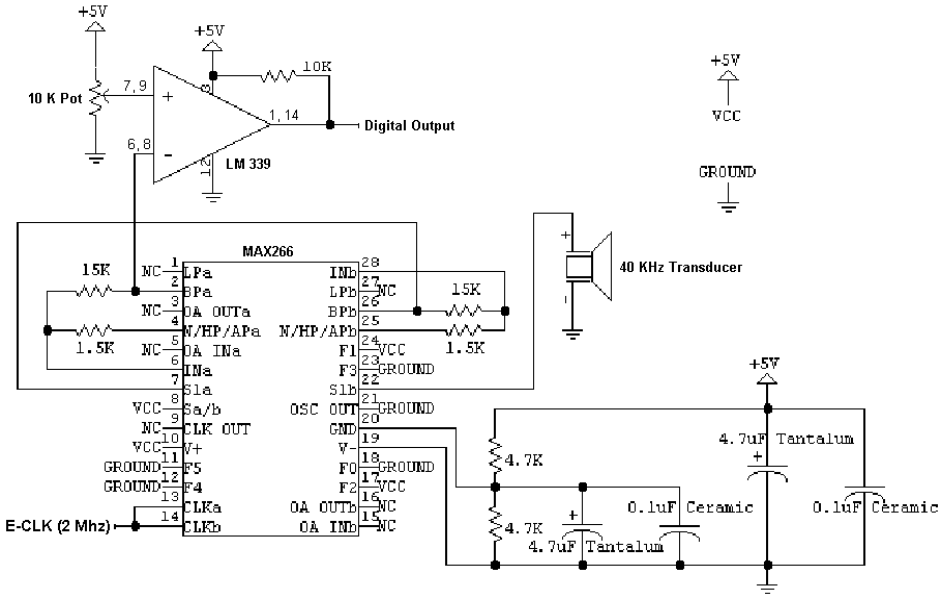

Figure 2: Sonar Receiver Circuit

Figure 6

CDS Cells

CDS cells are used to detect the amount of light observed. As they detect more light, their resistance decreases. For testing, a 60-Watt bulb was placed at varying distances from the cell in a dark room. The data is in Figure 7.

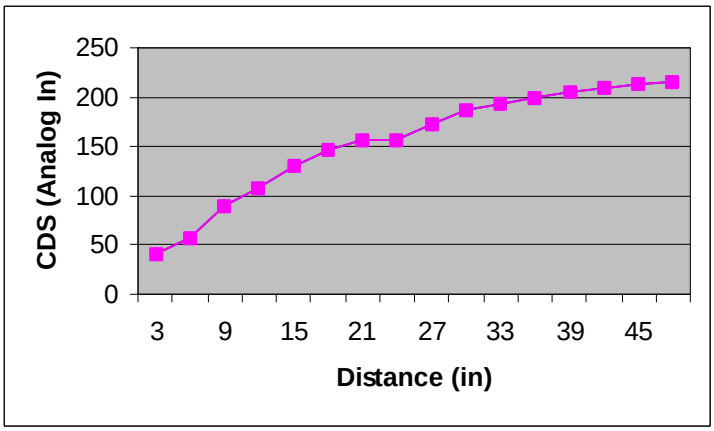

Figure 7

While this data was taken at a relatively close distance, CDS was intended to be used at a long distance. To do this, the resistor that is in series with the detector was increased. This gave a wider range of values and allowed the cell to detect weaker sources of light. Because the demonstration of JR was done during daylight hours, the CDS cells were not implemented.

Metal Detector

The metal detector is the most important sensor in this design. The values sent into the microprocessor were between 1 and 2 volts, for no reading, and full scale, respectively. These values translate to digital values of 50 to 100.

For all tests, the metal detector was giving very accurate and repeatable readings. However, the night before demo day, the metal detector fried. While the problem was researched, no reason was found. Because of the malfunction, the metal detector was bypassed with a switch.

Behaviors

Obstacle Avoidance

JR uses sonar and IR for obstacle avoidance. When one of the transducers receives a reading within the acceptable range, it reverses the opposite motor for a set amount of time. This turns the robot away from an object on one side, and backs it up away from an object in front of it.

Sonar is the main system for obstacle avoidance. However, if there is a malfunction with sonar, IR may see an obstacle before JR runs into it. IR is strictly a backup, though.

Beacon Following

As JR moves, the CDS cells are scanning for the amount of light visible. If there is more light visible by the left cell, JR lists to the left until the light values are equal. The same is true for the right. This behavior allows JR to move toward a beacon placed many yards away from the robot.

Metal Detecting

As JR moves around, the microprocessor is continuously polling the metal detector. If a high reading is ever found, JR switches to digging mode. Otherwise, JR continues obstacle avoidance and beacon following.

Digging

The most difficult behavior for JR is digging. Once an object is detected, JR moves back for a set amount of time. Once there, the conveyor belt starts turning and is lowered into position to dig up dirt. The robot then moves forward for another amount of time. Then the belt is raised and stopped and JR goes back to normal operation.

Experimental Layout and Results

Creating JR was done in three basic parts. Sensors, programming, and mechanical designing was done separately and put together at the end. This was a mistake. Initially, the sonar system was tested and working well, but as soon as it was installed on the platform, many problems arose. First, when the transmitter sends the signal, it is still ringing in the aluminum by the time the receiver is being polled. This gives false readings, and a delay had to be put in the code before it detected a received signal. This is why this sonar configuration is not adequate for close readings.

Another problem with the sonar system is the very small signal sent to the sonar receiver circuit by the receiver. The receiver was not getting any signals because of the length of the conveyor belt. The circuit had to be moved into the frame of the belt, and covered in electrical tape to protect it.

The last major problem was the metal detector. Testing it by itself gave great results, but as soon as it was placed below the conveyor belt, it detected the aluminum in it. The normal state of the belt had to be raised to move it away from the metal detector.

Initially, JR was going to be programmed in IC. However, there are problems with using two motors and a servo in IC. For that reason, ICC11 was used. In version 5, however, there are problems with the motor command. There is speed control, but the motors cannot be reversed. The motor routine given is to be used with the Talrik platform, but does not work on the ME11 board. To fix this, Darren Kelley wrote a subroutine that sets a PortD bit high or low, and controls the motor direction. Now to use ICC11 v5 with the ME11 board, the programmer can paste this subroutine at the end of the code and this will fix any problems. This subroutine is located at the end of JR's code.

Conclusions

While the metal detector malfunctioned at the last minute, and the CDS cells were not implemented on Demo day, the robot was successful. The greatest success of JR is the conveyor belt. More hours were spent on perfecting the operation of the belt than any other single part of the platform. All through design, the belt would slip off of the gears. Finally after mounting the sprockets much more accurately, the belt stayed on. Then building the chain tensioner/derailer kept the chain on the drive gears well.

The largest problem with JR is also the conveyor belt. While it functions well, it is not currently strong enough to actually dig. To make it dig, the scoops would have to be attached in a firmer manner.

Future work on this robot would include making it much more robust. The first part of work would be determining what made the metal detector malfunction and fix the problem. The part would be making the scoops firmer so they would actually dig.

Overall, JR performed well. While some of the previous problems and future work would have enhanced its performance, there was a lot accomplished this semester.

Documentation

Many people inside and outside of IMDL helped with this project. Darren Kelley worked with me a lot to develop the sonar code used, as well as the fix for the motor problem with ICC11. Without the sewing help of Corrie Ross, the conveyor belt would not have had a belt in it for demo day. Brent Fox donated all the bicycle parts used in the construction of the conveyor belt (many sprockets, a derailer the was torn apart, and the chains.) Finally, the following people helped out as the semester went along with ideas to fix various problems: Scott Jantz, Aamir Qaiyumi, Patrick O'Malley, Eric Anderson, Megan Grimm, and many others I'm sure I'm forgetting here.

Vendors

All parts not listed here were either obtained from the IMDL lab, common items from Radio Shack, or scrap pieces picked up from Ace Hardware.

The servos were ordered from several different vendors. However, most vendors offered the same servos at similar prices. Also, prices would fluctuate in as small amount of time as a couple of weeks. For this reason, it is best to start pricing servos well before they are needed and keep an eye on the prices.

Electronic Goldmine

Sonar Transducers
Part description: 40 kHz ultrasonic transducers, 15/16" diameter
Part #: G2528
Unit price: $1.50
Website: http://www.goldmine-elec.com

Maxim Integrated Products

Sonar Filter (MAX266)
Part description: switched-capacitor active filters
Part #: MAX266ACPI
Unit price: unknown
Website: http://www.maxim-ic.com

Radio Shack

Part Description: audio output transformer, 1K ohm to 8 ohm
Part #: 273-1380
Unit price: $1.99

Part Description: Metal detector
Part #: 63-3005
Unit price: $49.99

Appendix A
Sonar Test Code

```
/* Justin Cobb ('borrowed' from Megan Grimm) */
/* 3/21/99 */
/* Sonar testing program */

int counter;

void main()
{
        init_serial();
        counter=0;
        while(1)
        {
        counter=0;
                init_serial();
                msleep(1L);
                poke(0x7000,0x00);

        while((analog(3)>200)&&(counter<1000))
        {

                counter=counter+1;

        }
        write_int(counter);
        }
}
```

Appendix B
JR Main Program

```
/************************Main Program*******
* Justin Cobb
* Revision (Who Knows)
*****************************************/


/******** Includes ********/

#include <tkbase.h>
#include <stdio.h>

/******** Defines ********/
#define Output_Latch *(unsigned char *)(0x7000)

/******* Prototypes ******/

void motor(int, int);
void LeftSonar(void);
void RightSonar(void);
void TurnAway(void);

/********** Variables ***********/
long i, j;
int IrRt_tmp, IrLft_tmp, IrRt, IrLft, RtSpd, LftSpd;
int LeftThresh, RightThresh, counterlft, counterrt, SnrLvl;
int Rt = 0;
int Lft = 1;
int Rt_IR = 0;
int Lft_IR = 4;
int SonarLft = 0x08;
int SonarRt = 0x01;
int Sonar = 7;
int MDetect = 3;
int MetalThresh = 0x40;
int belt = 0x04;
int Servoval = 0;
int ServoDrive = 2300;
int ServoDig = 3600;
int Clear = 0;




/************* Main ***************/

void main(void)
{
       init_analog();
       init_motortk();
       init_clocktk();
       init_servos();
       init_serial();
       DDRD = 0xff;
       CLEAR_BIT(PORTD, belt);
       for(i = 3350; i > 1750; i--)
       servo(0, i);
       motor(0, 0);
       motor(1, 0);
       Output_Latch = 0x00;

 while(1)
 {
       LeftSonar();
       RightSonar();
```

```
        if ((RtSpd == -100) || (LftSpd == -100))
                TurnAway();
        motor(Rt, RtSpd);
        motor(Lft, LftSpd);

        if (analog(MDetect) > 240)
                {
                RtSpd = -100;
                LftSpd = -100;
                motor(Rt, RtSpd);
                motor(Lft, LftSpd);
                for(i = 0; i < 5000; i++)
                        {
                        for(j = 0; j < 1; j++);
                        }
                motor(Rt, 0);
                motor(Lft, 0);
                SET_BIT(PORTD, belt);
                for(i = ServoDrive; i < ServoDig; i++)
                        {
                        servo(Servoval, i);
                        for(j = 0; j < 5; j++);
                        }
                motor(Rt, 25);
                motor(Lft, 25);
                for(i = 0; i < 5000; i++)
                        {
                        for(j = 0; j < 5; j++);
                        }
                motor(Rt, 0);
                motor(Lft, 0);
                for(i = ServoDig; i > ServoDrive; i--)
                        {
                        servo(Servoval, i);
                        for(j = 0; j < 5; j++);
                        }
                CLEAR_BIT(PORTD, belt);
                }
 }
}



/************* End Main ****************/


void LeftSonar(void)
{
        counterlft = 0;
        SnrLvl = 255;
        Output_Latch = SonarLft;
        for(i = 0; i < 100; i++);
        Output_Latch = Clear;

        while((SnrLvl > 125) && (counterlft < 300))
        {       SnrLvl = analog(7);
                counterlft++;
        }
        if ((counterlft < 100) && (counterlft > 5))
                LftSpd = -100;
        else LftSpd = 100;
}


void RightSonar(void)
{
        counterrt = 0;
        SnrLvl = 255;
        Output_Latch = SonarRt;
        for(i = 0; i < 100; i++);
        Output_Latch = Clear;
```

```
        while((SnrLvl > 125) && (counterrt < 300))
        {       SnrLvl = analog(7);
                counterrt++;
        }
        if ((counterrt < 100) && (counterrt > 5))
                RtSpd = -100;
        else RtSpd = 100;
}

void TurnAway(void)
{

        motor(Rt, RtSpd);
        motor(Lft, LftSpd);

        for(i = 0; i < 5000; i++);

}

void motor(int index, int percent)
/*************************** motor ****************************
 * Description
 * Sets the speed and direction for the motors.
 *
 *
 *
 *Usage: to controls motor direction
 ************************************************************************/
{
        if (index == 0)
        {       /* Set the Direction for motor0*/
                if ( percent < 0)
                {       SET_BIT(PORTD, 0x20);         /* SET PD5 to high */
                        printf("you are in the motor function index 0\n");
                        motortk(index, percent);      /* Call motor command*/
                }
                else
                {       CLEAR_BIT(PORTD, 0x20);       /* Set PD5 to low */
                        motortk(index, percent);      /* Call motor command*/
                }

        }
        else if (index == 1)
        {       /* Set the direction of the motor1 */
                printf("you are in the motor function index 1\n");
                if ( percent < 0)
                {       SET_BIT(PORTD, 0x10);         /* Set PD4 to high*/
                        motortk(index, percent);      /* Call motor command*/
                }
                else
                {       CLEAR_BIT(PORTD, 0x10);       /* Set PD4 to low*/
                        motortk(index, percent);      /* Call motor command*/
                }

        }
}
/*************************** motor **** ***************************/
```