

Date: 12/08/09
Student Name: Fernando N. Coviello
TAs : Mike Pridgen
Thomas Vermeer
Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

Final Report

“Metallocalizer”

University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

Table of Contents

Abstract.....	3
Introduction.....	3
Integrated System.....	3
Mobile Platform.....	4
Actuation.....	7
Sensors.....	7
Behaviors.....	9
Experimental Layout and Results.....	10
Conclusion.....	10
Documentation.....	10
Appendix A.....	11
Appendix B.....	15

Abstract

Metallocalizer is an autonomous construction site cleanup robot. It searches for waste metals and debris and sweeps it up. It uses sharp IR sensors to perform obstacle avoidance, and bump switches to detect crashes. The special sensor is a metal detector, which guides the robot where the waste needed to be picked up is, while it performs obstacle avoidance returning LCD feedback.

Introduction

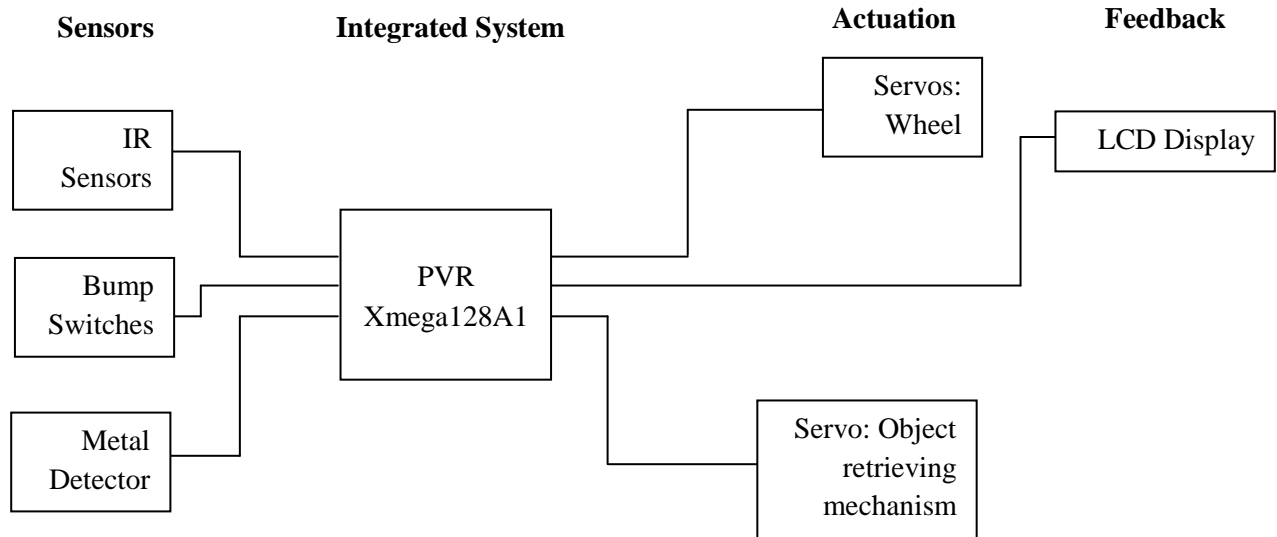
Metallocalizer is a simple obstacle-avoidance robot with an additional system to detect and pick up small metal objects. The area of application of this robot is the construction environment. The idea came up to me after working a couple months in the construction industry, and I noticed the most important task was to keep a clean working area. A clean working area is safe to workers and makes the job a lot easier; however, doing this cleanup while working is a really annoying task that nobody wants to do. So, I thought it would be good to have a machine do this work for me. Also, it would increase the productivity of the workers since they will not be losing time cleaning up.

Integrated System

The main board will be a PVR Xmega128A1 that will control the behavior of the robot as well as the power management of the sensors, servos, and LCD display. The board is powered by eight 1.3V rechargeable batteries, and programmed using AVR studio, and interfaced onboard with the computer by the AVR ISP mkII In-System Programmer.

Three IR sensors used for collision avoidance are connected to the A/D port in the main board as well as the bump switches and metal detectors. I connected four bump switches to a single A/D port pin. I was able to achieve this by connecting the switches to a voltage divider circuit, returning different voltages according to the different combinations of switches pressed. Three hacked servos are connected to the PWM port, which was extremely useful because it saved the need for motor drivers. Finally, three LED's are connected to an I/O port. One of these LED's is to indicate that the power is on, and the other two indicate that the metal detectors are working.

The microcontroller is programmed using C language, which made the programming job fairly simpler than other assembly languages. Also, the Xmega A and PVR board manuals are extremely useful and easy to read and understand. However, help from Mike and Thomas was the most useful tool that I had in the process of programming the board.



Mobile Platform

While testing the sensors, servos, and designing the additional system, I used a temporary platform. The temporary platform was similar to the final one, but I used a temporary first because I knew that I was going to make numerous modifications as I moved along, and I did!. Both are made out of wood and have three wheels, two of them powered by two hacked servos. The final platform is a circular wooden platform with a steel cover. It is 12" in diameter and about 2" high from the floor. It has three Sharp IR sensors in the front, attached below the wooden platform. On top of the IR's, there are three bump switches mechanically interconnected by a rubber rod to make a bumper. Approximately in the middle of the platform is the special mechanism. This mechanism uses the broom and dustpan system; it basically consists of a rotating brush with a dustpan on the back. The brush rotates sweeping the floor and the dustpan in the back scoops up what the brush swept. The original idea was to make it pick up metals only, but the original design had many flaws, for example, it would only pick up ferrous metals, also, due to a different mechanism, the platform would have been heavier requiring more powerful servos. Figure 2 (a) shows how the system looks from the bottom, and Figure 2 (b) shows the opening on the right to retrieve picked up objects. The third servo is placed on the top part of the platform with a rubber belt attached that passes through the platform and connects the servo to the rotating brush. This servo is screwed to the platform with two springs to keep the belt under tension (Figure 2 (c)).

The rotating brush was taken off from a vacuum cleaner and linked to a servo by a rubber belt. The cover was a ceiling light that I took apart and drilled some holes on it. Besides that,

everything else was handmade. The most difficult part was the dustpan because I designed it under ideal conditions and reality is definitely not ideal, hence I had to make countless tests and modifications until I could make it work properly.



Figure 1(a). Final Design with cover

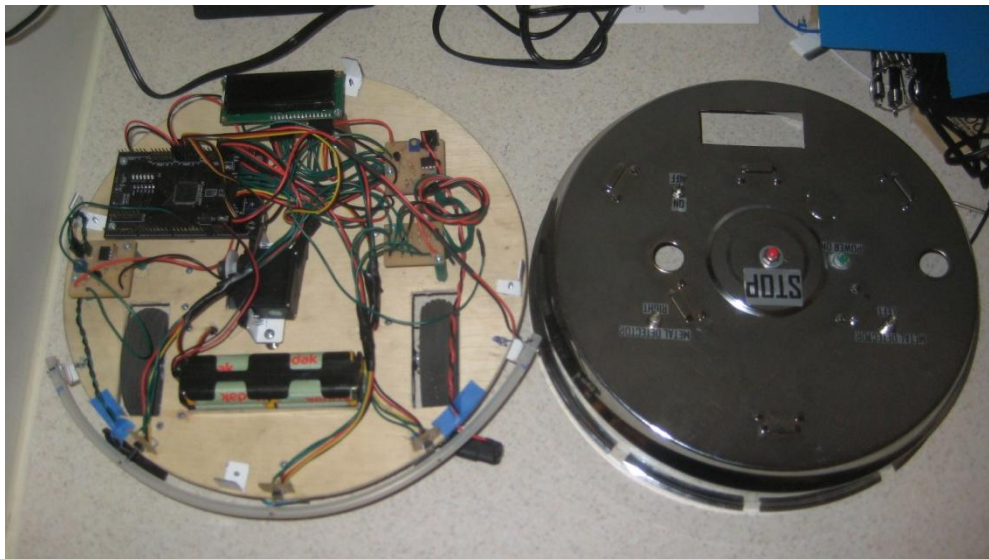


Figure 1(b). Final Design without cover



Figure 2(a)

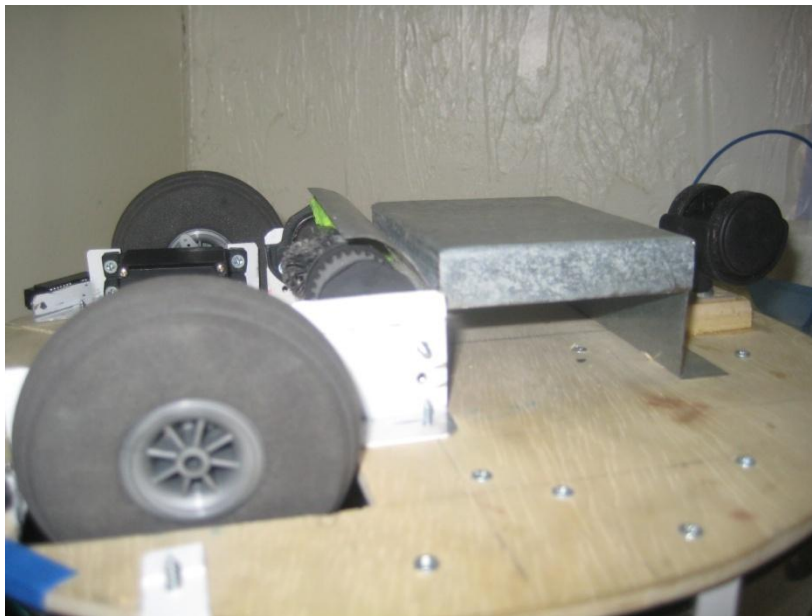


Figure 2(b)

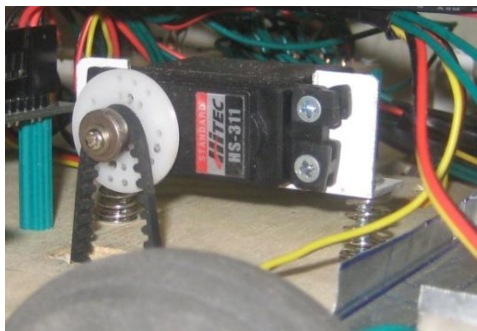


Figure 2(c)

Actuation

The two wheels that moves the robot will be powered by two hacked servos, and a third wheel will be located in the back of the platform for steering and stability. A third servo turns the rotating brush. I chose hacked servos over DC motors because hacked servos do not need motor drivers; they are controlled directly by the PVR board. Also, the implementation of servos is cheaper than motors since they already include gears. For my design, I needed to convert the speed of the motors into torque, which meant build a gearbox to gear down the speed of the motors. It turns out that building two custom made gearboxes costs more than the whole robot itself. In brief, servos may not be as powerful as geared down motors but they are noticeably cheaper.

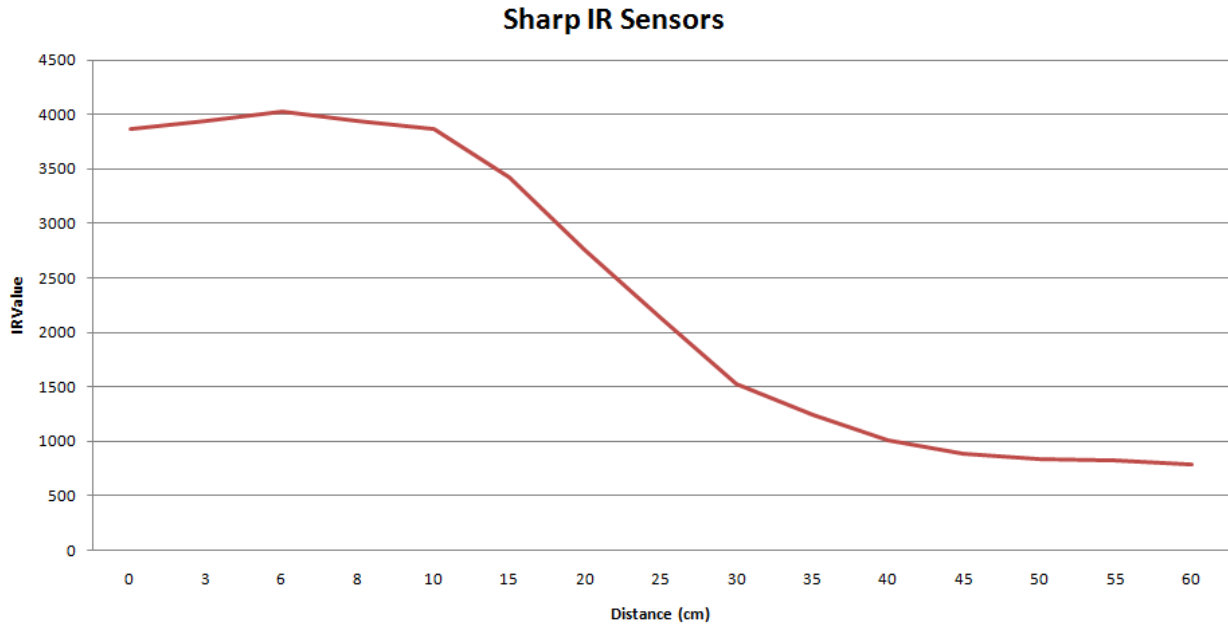
Since there will be traction in only two wheels, steering is performed by turning one of the wheels at a time, and in the case of very sharp turns each wheel will turn in opposite direction. The third wheel is not attached to any motor in order to turn freely in any necessary direction. The servo connected to the brush was originally planned to turn every time the metal detector was triggered, but because the brush rotates slower than the wheels it need to be constantly rotating to make sure everything it sweeps gets on the dustpan.

The bumper is a rubber rod located at the front of the platform, and it is connected to three bump switches. Every time the rubber rod is hit anywhere; at least one switch will be pressed. This will send a voltage to the PVR board letting it know that the robot hit something.

Sensors

Infrared

As stated before, three Sharp IR sensors are used for collision avoidance. IR sensors were chosen over sonar sensors because they are cheaper, less noisy and easier to implement. These sensors are connected directly to the A/D converter port of the PVR board, which also provides power. IR sensors respond to light reflection. Measurements done with a beige object returned the following values read from the PVR board after conversion.



Results showed that about 6cm away, the value is maximum and decays if the object gets closer or further. Since it is a short range IR, it is useful for applications up to 10 cm.

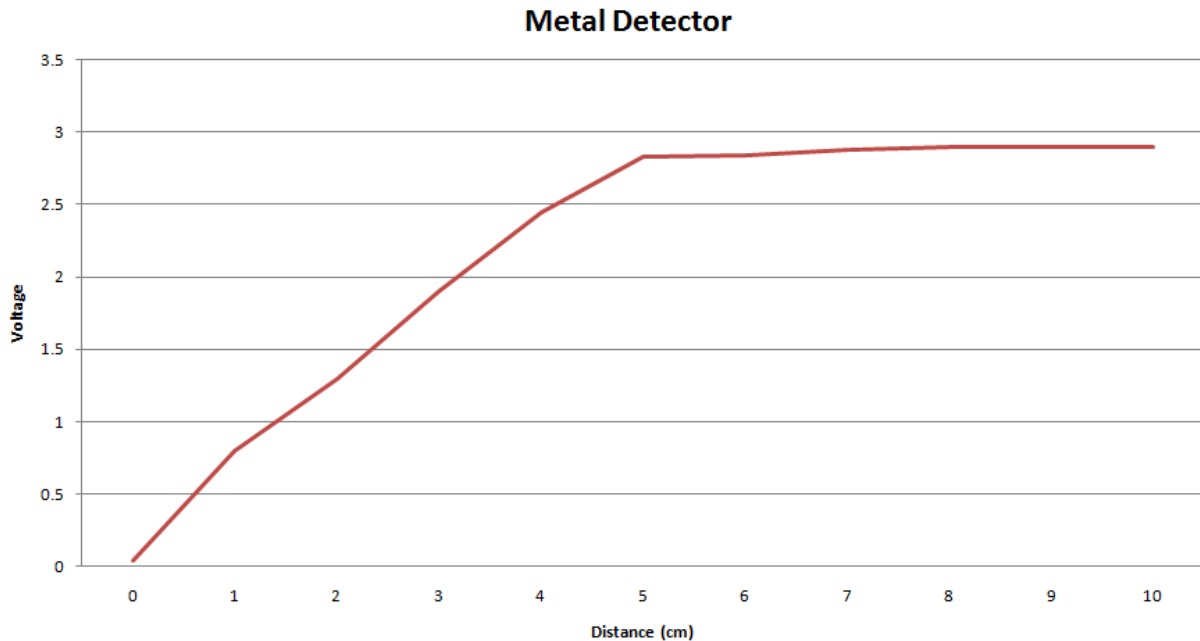
Bump Swithces

Four bump switches were connected in a voltage divider circuit. Three of these switches were connected to the bumper, and the fourth was planned to be a start button, but remained unused. Each switched pressed independently retuned a different value, as well as any possible combination. This is useful because all the switches were connected to a circuit with only one output that was connected to an A/D pin. This made a more efficient use of the board that connecting each switch independently with pull-up resistors to different I/O pins. The values retuned of every switch pushed independently are the following:

Switch	Voltage
S1	0.45V
S2	1.2V
S3	2.3V
S4	3V

Metal Detector

My special sensor is a metal detector, which I designed and built myself (see schematic in Appendix B). It consists of a TDA0161 chip, which a proximity detector, connected to a NPN transistor used as an amplifier. It works with 5V, which is really convenient because that is one of the voltages provided by the PVR board. In idle mode, the detector will return approximately 2.9V, and as it gets closer to metal objects this voltage will decrease until it reaches 0.04V. Experimental results are expressed in the following graph.



Metallocalizer has two of these sensors, one on each side connected to the A/D port. On the cover of the robot there are two LED's that are on when the detectors are working, and turn off if the detectors find something.

Behaviors

The original behavior that I wanted to give the robot was: “When turned on, the robot will give priority to the signal received by the special sensor. If there is no signal, it will start moving randomly avoiding objects until the metal detector is triggered. Once the metal detector is triggered, the robot will be guided to the place where the attraction is strongest. When the metal detector signal returns to idle, the robot will keep on moving randomly”. However, this behavior is becoming hard to implement since the metal detector behavior is not coming up as expected. Therefore, I changed the behavior. Now, when turned on, it takes 5 second to initialize, and then it starts moving randomly avoiding obstacles. While it is moving it displays on the LCD what it is doing. If any of metal detectors is triggered, the robot will turn to where the metal detector is pointing and run over the object to pick it up. If there is no signal from the metal detector it will keep moving randomly looking for more metals.

On top of the cover there are two switches and three LED's. One of the switches is to turn the robot on and off. A red switch in the middle labeled “STOP” is a reset button. A green light next to the reset button indicates that the power is on, and a red LED at each side of the robot indicates that the metal detectors are functioning.

Experimental layout and design

Most of the experimentation has been concerning the special sensor and special mechanism. Also, I have been working with the servos for the first time. First I thought hacking servos meant removing the electronic circuitry inside leaving only the motors and gears. Afterward I learned hacking servos meant removing the stop from the gears and setting the variable resistor at the point where it moves faster.

The metal detector required a lot of time finding the correct inductor for the job. I tried making a custom made one, but all trials failed. I ended up using two 100 μ H connected in parallel for each sensor.

As expressed before, the dustpan of the special mechanism also took longer as expected. The biggest issues were weak materials that were light but useless, and the speed of brush could not work at the same pace of the wheels leaving most of the objects it was supposed to pick up on the floor.

Conclusion

Overall I am proud of my robot; it had many complications but it finally resulted as I wanted, I was able to keep it in a reasonable budget. Undoubtedly, I would take this class again many times if I could, but, of course, I would do many things different. Even though the class is over, I will keep improving my actual robot, and start working in other projects. I would also like to learn how to design my own board.

Documentation

- [1] Pridgen Vermeer Robotics, "Pridgen Vermeer Robotics Xmega128 Manual"
Available: <http://plaza.ufl.edu/rhaegar/XMega%20Manual.pdf>
- [2] Sharp, "General Purpose Type Distance Measuring Sensors", Available:
<http://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf>
- [3] SGS-Thomson Microelectronics, "TDA0161 Proximity Detectors", Available:
<http://www.datasheetcatalog.org/datasheet/stmicroelectronics/1441.pdf>
- [4] Atmel, "XMEGA A MANUAL", Available:
http://www.atmel.com/dyn/resources/prod_documents/doc8077.pdf

Appendix A

```
#include <avr/io.h>
#include "PVR.h"

/*****
*
*   Wheel Movement Definitions
*
*****/

// Connect left wheel servo to PWMC 0
// Connect right wheel servo to PWMC 1
// Connect brush servo to PWMC2

void MoveForward(void)
{
    ServoC0(-100);           //move left wheel forward
    ServoC1(85);            //move right wheel forward
}

void TurnLeft(void)
{
    ServoC0(100);           //move left wheel backwards
    ServoC1(100);          //move right wheel forward
}

void TurnRight(void)
{
    ServoC0(-100);          //move left wheel forward
    ServoC1(-100);         //move right wheel backwards
}

void MoveBackwards(void)
{
    ServoC0(85);            //move left wheel backwards
    ServoC1(-100);         //move right wheel forward
}

void StopMoving(void)
{
    ServoC0(-22);           //Stop left wheel
    ServoC1(44);           //Stop right wheel
}

/*****
*
*   Metal Detector Commands
*
*****/

void LeftMetalDetector()
{
```

```

    if (ADCA6() < 700)                // Connect left metal detector to ADCA 6
    {
        PORTF_OUT = 0;                // Turn off LED
        TurnLeft();                   // Turn right if metal is detected on the right
        ServoC2(100);                 // Turn on brush servo
    }
    else
    {
        ObstacleAvoidance();          // Keep performing obstacle if no metal found
        PORTF_OUT = 1;                // Turn on LED
    }
}

void RightMetalDetector()            // Connect right metal detector to ADCA 7
{
    if (ADCA7() < 700)
    {
        PORTH_OUT = 0;                // Turn off LED
        TurnRight();                  // Turn right if metal is detected on the right
        ServoC2(100);                 // Turn on brush servo
    }
    else
    {
        PORTH_OUT = 1;                // Keep performing obstacle if no metal found
        ObstacleAvoidance();          // Turn on LED
    }
}

/*****
*
*   Obstacle Avoidance
*
*****/

void ObstacleAvoidance()
{
    // Connect left IR to ADCA0
    // Connect center IR to ADCA1
    // Connect right IR to ADCA2
    // Connect bumper to ADCA4

    int i=3950;

    {
        if (ADCA0() > i)
        {
            lcdGoto(0,0);              // Write to first LCD line
            lcdString("Obstacle Found"); // Display "Obstacle Found" on
                                         LCD bottom line
            lcdGoto(1,0);              // Write to second LCD line
            lcdString("Turning Right  "); // Display "Turning Right" on LCD
                                         top line
            TurnRight();                // Turn Right to avoid obstacle
                                         on the left
        }
    }
}

```

```

        ServoC2(100);          // Turn on brush servo
    }

    else if (ADCA1()>i && ADCA2()<i || ADCA4()>3000 && ADCA4()<i)
    {
        lcdGoto(0,0);          // Write to first LCD line
        lcdString("Obstacle Found"); // Display "Obstacle Found" on
                                   LCD top line
        lcdGoto(1,0);          // Write to second LCD line
        lcdString("Turning Right "); // Display "Turning Right" on
                                   LCD bottom line
        TurnRight();           // Turn Right to avoid obstacle
                                   at front if there is nothing on the
                                   right or if something is hit from
                                   center to left
        ServoC2(100);          // Turn on brush servo
    }

    else if (ADCA2()>i)
    {
        lcdGoto(0,0);          // Write to first LCD line
        lcdString("Obstacle Found"); // Display "Obstacle Found" on
                                   LCD top line
        lcdGoto(1,0);          // Write to second LCD line
        lcdString("Turning Left "); // Display "Turning Left" on
                                   LCD bottom line
        TurnLeft();            // Turn left to avoid obstacle
                                   on the right
        ServoC2(100);          // Turn on brush servo
    }

    else if (ADCA1()>i && ADCA0()<I || ADCA4()<2500)
    {
        lcdGoto(0,0);          // Write to first LCD line
        lcdString("Obstacle Found"); // Display "Obstacle Found" on
                                   LCD top line
        lcdGoto(1,0);          // Write to second LCD line
        lcdString("Turning Left "); // Display "Turning Left" on
                                   LCD bottom line
        TurnLeft();            // Turn left to avoid obstacle
                                   at front if there is nothing on the
                                   left or if something is hit from center
                                   to right
        ServoC2(100);          // Turn on brush servo
    }

    else
    {
        lcdGoto(0,0);          // Write to first LCD line
        lcdString("No Obstacles "); // Display "No Obstacles" on
                                   LCD top line
        lcdGoto(1,0);          // Write to second LCD line
        lcdString("Moving Forward"); // Display "Moving Forward" on
                                   LCD bottom line
    }

```

```

        MoveForward();                // Move forward if there are no
                                        obstacles around
        ServoC2(100);                // Turn on brush servo
    }
}

/*****
*
*   Main Program
*
*****/

void main(void)
{
    xmegaInit();                    //setup XMega
    delayInit();                    //setup delay functions
    ServoCInit();                   //setup PORTC Servos
    ServoDInit();                   //setup PORTD Servos
    ADCAInit();                    //setup PORTA analog readings
    lcdInit();                      //setup LCD on PORTK
    PORTH_DIR |= 0x01;              //set H0 as output
    PORTQ_DIR |= 0x01;              //set Q0 as output
    PORTF_DIR |= 0x01;              //set F0 as output

    // Display initialization sequence on LCD Display

    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Initializing");
    lcdGoto(1,0);
    lcdString("4");
    delay_ms(1000);
    PORTQ_OUT = 1;
    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Initializing");
    lcdGoto(1,0);
    lcdString("3");
    delay_ms(1000);
    PORTQ_OUT = 0;
    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Initializing");
    lcdGoto(1,0);
    lcdString("2");
    delay_ms(1000);
    PORTQ_OUT = 1;
    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Initializing");
    lcdGoto(1,0);
    lcdString("1");
    delay_ms(1000);
}

```

```

PORTQ_OUT = 0;
lcdData(0x01);
lcdGoto(0,0);
lcdString("Initializing");
lcdGoto(1,0);
lcdString("0");
delay_ms(1000);
PORTQ_OUT = 1;

```

```

while (1)
{
    LeftMetalDetector();
    RightMetalDetector();
    ObstacleAvoidance();
}
}

```

Appendix B

Metal Detector

