Final Report

Erik Stegman

Robot: PME
(Personal Mobile Evacuator)

EEL4665
Instructors: Dr. Arroyo, Dr. Schwartz
TAs: Josh Weaver, Ryan Stevens, Tim Martin

Table of Contents

Abstract:

PME is a fully-autonomous, mobile robot designed to transport an empty bottle to the kitchen and dispose of the bottle in the garbage. Its functions include understanding and responding to being summoned, gripping objects, line following, obstacle avoidance, and mechanical lifting. Upon being summoned, PME will locate the summoner by following a line to predetermined destination. The summoner will then load an empty 12 ounce bottle in the gripper of the robot and instruct a command for disposal. PME will then search for a line on the floor which it will follow to the kitchen. Upon detecting the garbage can with reflectivity sensors, the robot will stop, lift and extend its arm, dumping the bottle into the can. After a delay, PME will return to its starting location.

Executive Summary:

The process begins with a pile of small metal pieces of various shapes and sizes, along with a few thousand screws, nuts, and washers. The erector set parts are rigid and fully customizable, allowing for the construction of stable bodies of virtually any shape and size. For this reason they seemed like an ideal candidate for a robot. These parts were used to construct the entire body of PME. They allowed a vision of a functional system to be brought into physical existence, and allowed for a construction process that could be modified along the way and adapted to the needs of the system, without having to redesign the entire frame. Extensions, hinges, and reinforcements were added as needed while maintaining overall design. The final Body uses over 200 parts and 400 screws.

At the front of the body are 2 bump sensors and 3 infrared sensors. One bump sensor, made from a free floating wire within a spring, checks the front and left side, while the other checks the front and right side. The IR sensors point middle, 45 degrees right, and 45 degrees left. They are calibrated for response to detection at approximately 6 inches. Behind these sensors are 2 light sensors facing upward. These are designed to receive a light activation signal.

Moving from front to back, the front axle is arrived at next. A servo motor centered about 5 inches up controls the motion of a bar extending vertically downward to a horizontal axle connecting the front wheels. This allows for a range of steering limited only by the range of the servo. Seated above the steering mechanism is our driver, an erector set construction worker who is always on duty.

Behind the steer servo, in the heart of the robot, is the PVR Xmega board, with all electrical components connected to it. From here, wires extend in all directions. Outside signals are received and processed, and motor control signals are sent out. Above this board is an LCD mounted for real time feedback and debugging purposes.

Below the controller board, mounted about ¼” up from the floor, are the 4 line-following sensors. These are spaced apart by slightly less than the width of a line of electrical tape, allowing 2 sensors to simultaneously detect the line. Behind The controller is another board for the motor drivers. Mounted on the rear of the bot is a final board interfacing the 4 line sensors and the feedback LEDs.

Underneath the motor driver board is the drive motor. The shaft of this motor is rear-facing, extending out to a worm gear which steps down the RPMs for more torque on the rear axle. Above the rear axle is the power station: 2 battery cases holding a combined 8 batteries for a total of 12 volts.

From the top of the robot cab extend 4 bars, a pair on each of 2 axles, which create a dual, 4-bar linkage system. Near the front axle, the linkage arms rest on springs which ease the initial torque demands of the arm. Connected to the rear linkage axle is a gear to interface with the stepper motor. The stepper motor is mounted as close as possible to this rear axle, and provides the actuation to lift the linkage arm from its resting position to its upright position. Stops have been put into place to prevent the arm from moving too far forward.

The linkage arm is used to lift and extend the bottle containing unit at the foremost part of the arm. This unit has a grip arm with a wire extending from the gripper, back through the body. This arrangement allows the lowering of the linkage to contract the gripper. The bottle containment unit consists of this grip and a base platform which is connected to another servo. This servo drops the platform, allowing and contents of the unit to be released into a collection area below.

Total body weight is approximately 8 pounds. The vehicle is 16 inches long with an 18 inch long linkage arm. The body is 6 inches wide, 17 inches tall with the linkage retracted, and approximately 20 inches tall with the linkage fully extended.

Introduction:

To set the mood, imagine yourself sitting on the couch and enjoying a few beverages. Perhaps the Gator game is very exciting, or maybe laziness has just set in. Either way, an autonomous robot could be desirable for a number of functions. Among the more useful are fetching a new beverage or food or answering the door. However, all those pesky empty cans will begin to pile up. An autonomous bot named PME has the capability to dispose of your waste for you, once you have conveniently placed into it a used beverage container. PME won't prevent you from ever having to get up off the couch, but at the least he will allow you to stay seated for a few more hours.

Although PME carries out a highly specialized task, its behaviors to can be adapted to fit many daily needs, depending on the programming. Its initial behavior is one of responding to an operator summoning it for duty. This may be referred to as the clap-on effect. The operator will use a visual method of alerting the robot that its services are in need. This will be a signal with a light. When PME is resting in a dark area, shining a flashlight will alert him. Upon activation, PME will follow a dark line to locate the summoner and then advance that direction. Arrival at the subject will be determined by either a change in light or a change in color of the line.

Once the bot arrives at the operator, it will stop and wait for a continue command. The operator will place a container in the grip arm of the machine and give it a signal to proceed. This signal could be a button on the surface of the bot, a button on a wireless controller, or another visual signal. Once this signal is received, using an array of IR reflective sensors to detect the line, the machine will then exhibit line-following behavior which will take it into the kitchen and up to the edge of the garbage can. While line-following, PME will be awaiting a signal from its line sensors that the target has been reached. Correct identification of the target is vital! The bot will then cease translational movement and begin to raise its mechanical arm. When the arm has been raised to sufficient height, it will extend forward and release the stabilizing platform. In the process, the bottle will be dumped into the can. The arm will then be retracted. The task being complete, PME will then resume line-following behavior as it returns to its resting location. It will then wait until being summoned again.

The following report includes detailed information on the logistics and implementation of the above behaviors. Sensors, mechanics, and programming are covered in detail. All part requirements, parts used, schematics, measurements, and code are provided below

Integrated System:

The most apparent part of PME is its body. Although the compact, circular platforms with two drive wheels and a caster seem reasonable for most applications, the lifting demands of this particular application make this design unreasonable. The design has been modified to only require the lifting of an empty container. The torque demands are much more significant for lifting a full bottle. To provide appropriate stability, PME has been given an elongated, rectangular base. The materials for the base include a variety of shapes and sizes of small, steel construction parts from erector set. These pieces provide a rigid body for the needed stability and are versatile enough for most any application. The lifting mechanism has also been modified. The initial plan was to use a crane style arm with a motor and pulley system to lift the arm and a second system to extend it. The new design involves a 4 bar linkage system. This design reduces hardware requirements by eliminating the need for any pulley systems. Lifting and extending will be performed all in one motion. The torque requirement will also be reduced.

PME will be equipped with several different sensors. The base will be surrounded with bump sensors. The surface of the platform also provides ample mounting space for infrared sensors. A microphone may also been fitted on the surface. The Circuit board will be secured inside the base with wires extending out to sensors and motors. The base also provides ample mounting opportunities for battery packs, an LCD for debugging and feedback purposes, and LED's for behavior indication. A block diagram of the complete system is shown in figure A. The flowchart for the controlling program is in figure B.

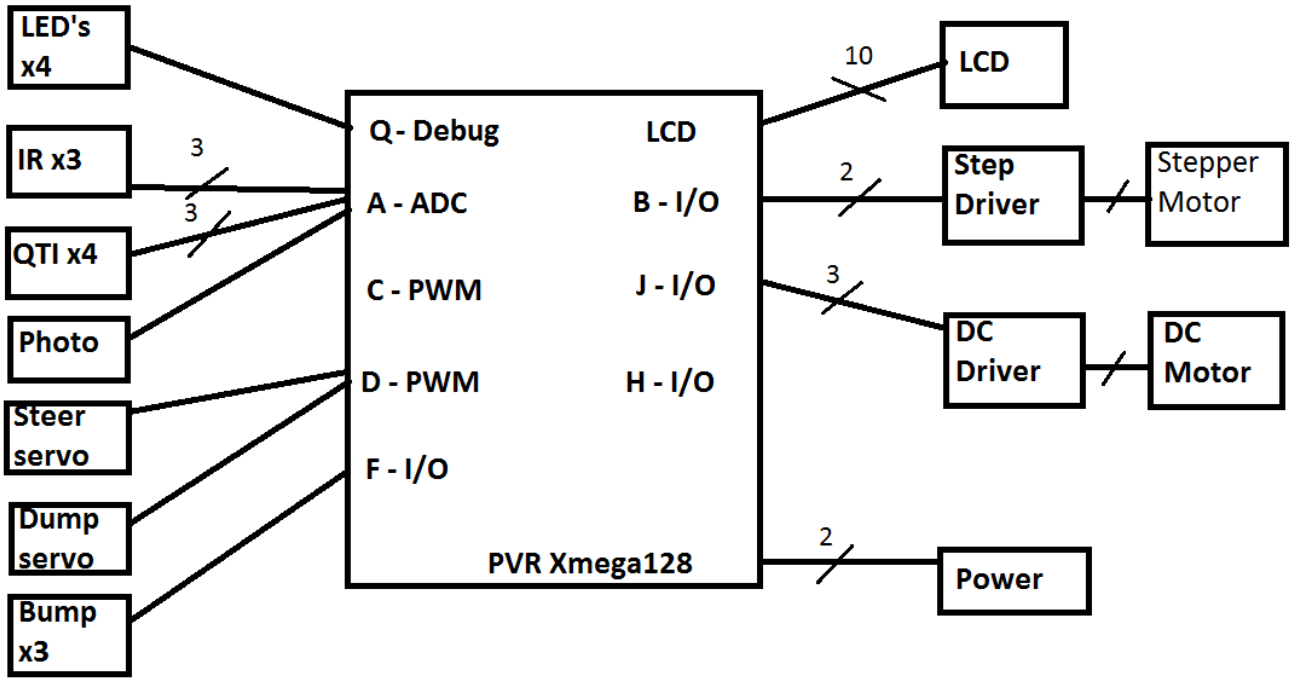

Figure A: System Block Diagram

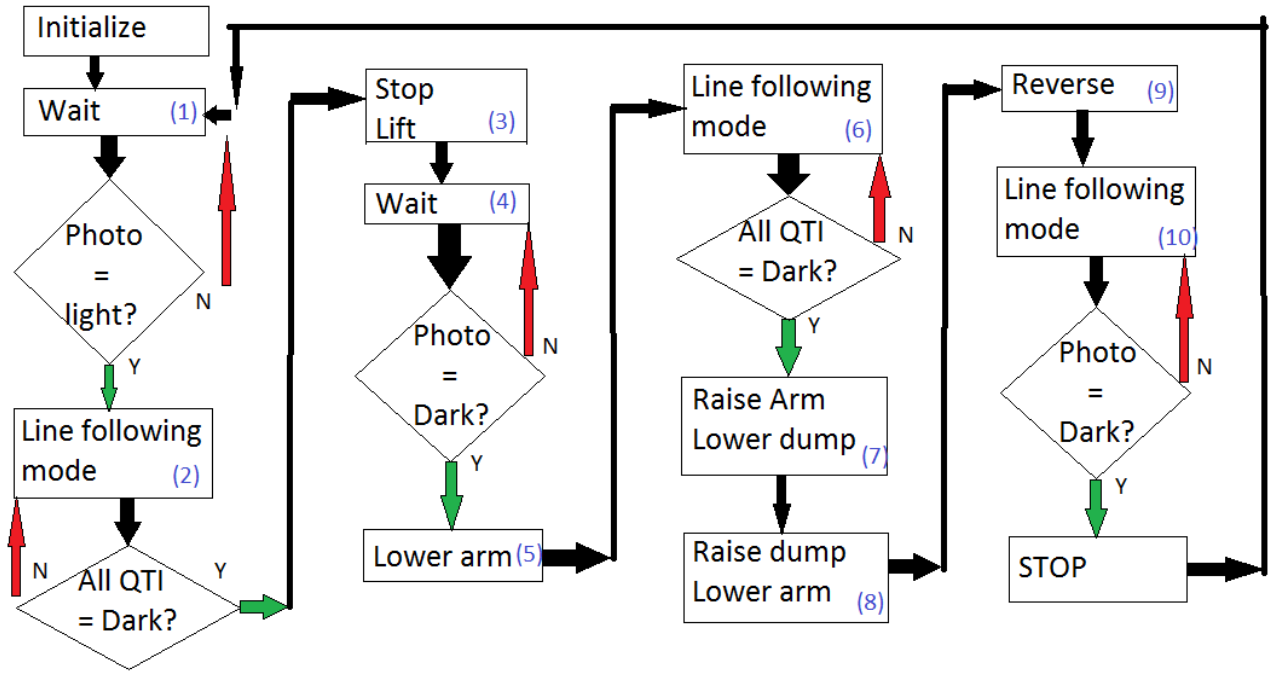

Figure B: System Flowchart

Mobile Platform:

The drive train consists of a two-wheel system driven by one motors. The front wheels are independently connected to a servo motor for steering. The rear wheels are being used for propulsion. Wheels for this vehicle are erector set pieces(figure C). The platform is approximately 14 inches long and 6 inches wide. These dimensions allow for adequate spacing between all sensors and electronic and moving pieces. Rising vertically from the platform are four legs of steel construction totaling between three and four inches in width and 7 inches in length. These legs attach to an elevated 4 bar linkage system. It will rise to a maximum height of 20 inches. It has been set back 6 inches from the nose of the platform for clearance while lifting. The linkage system raises the arm about 4 inches and extends it forward 4 inches past the nose of the platform. This allows enough clearance for dumping. It also ends in a grip arm. Rather than connecting the gripper to a servo with a high torque requirement, a cable has been run through the robot that contracts when the arm is lowered and loosens when it is raised. The bottle will be secured during transportation and released for dumping. The small platform directly underneath the grip arm is used for resting the bottle. This base will by tilted by another servo when the grip is released, allowing the bottle to slide out of the grip. A reversal of the linkage servo retracts the whole arm to a stable position for driving.

On the front edge of the platform are 3 infrared sensors and 2 bump sensors. The IR sensors are facing forward-left, middle, and forward-right. The bump sensors are on the left and right corners. Another bump sensor is also mounted in the back for obstacle detection while backing up. Underneath the bot is an array of QTI IR sensors facing the ground. This is positioned less than 1/2 inch from the floor. The last sensor is the pair of Photoresistors mounted upward facing on the front of the bot. These will be used to detect when a shady area has been reached. The platform is shown in figures D and E.

Figure C

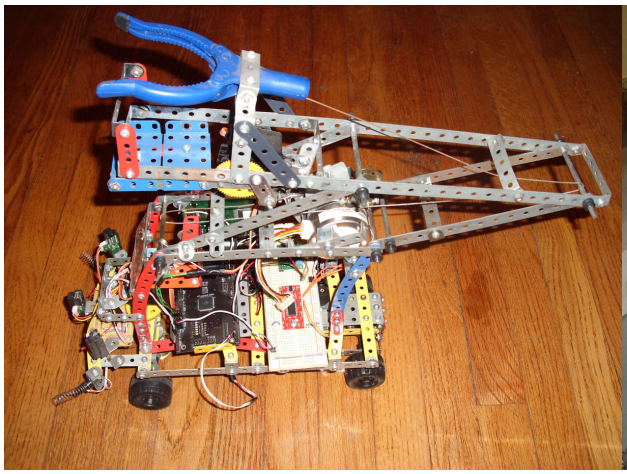

Figure D

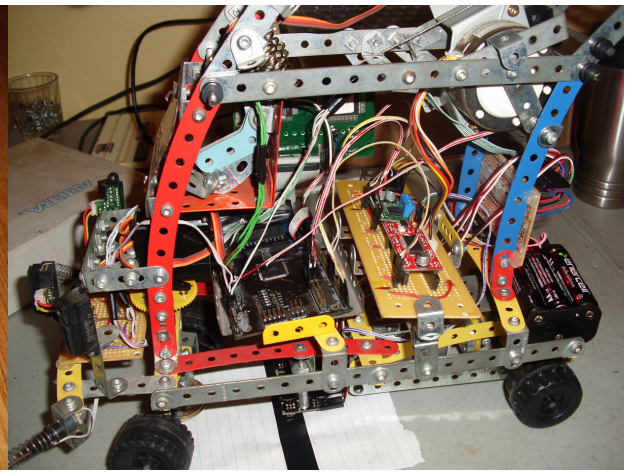

Figure E

Actuation:

PME is a 4-wheeled robot. The front wheels are used for steering by a separate servo (TowerPro MG995). This motor was rated at 13kg/cm but tests have shown this to be an overestimate. The design of the steering servo allows for a turning range limited only by the servo. The servo is shown in figure F.

Figure F

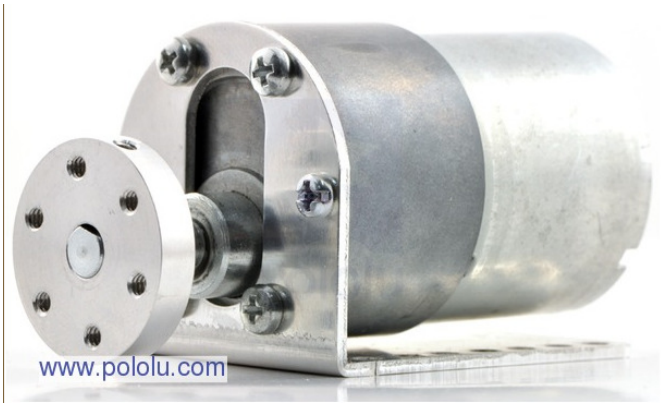

Figure G

The rear wheels are operated jointly by one dc motor (figure G). A gear-down system has been installed, consisting of a worm gear connected to the shaft of the motor connected to a 30 tooth gear on the drive axle. Due to the weight of the body, the first two motors used were insufficient to produce motion. The body also had to be reinforced multiple times around the drive train to prevent the gears from slipping past each other. The current motor is a 300mA free-run (5A stall) gearmotor from Pololu with a 19:1 gear ratio resulting in 500rpms. This was the fastest motor in this family. Unfortunately the geardown reduces the speed of the robot greatly. PME won't win any races, but that just gives him more time to avoid obstacles and detect destinations. The motor operates on a maximum of 12v which will be supplied from the alkaline batteries. Graphical data was unavailable for this motor. The motor is being driven by a 5A max Pololu motor driver carrier MC33926(figure H). The schematic for connections between the driver and the driver board is shown in figure I.

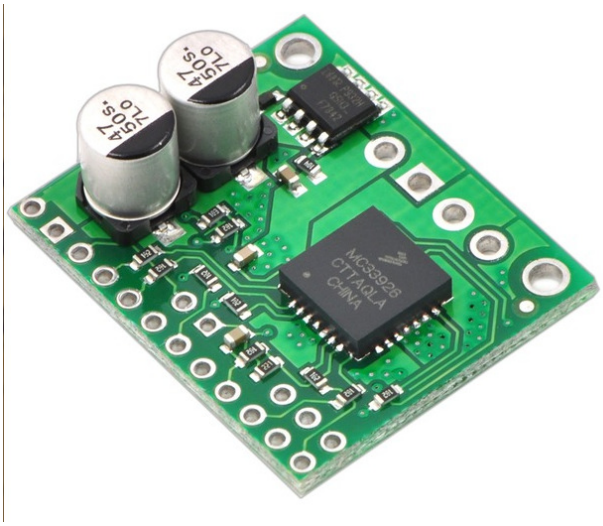

Figure H

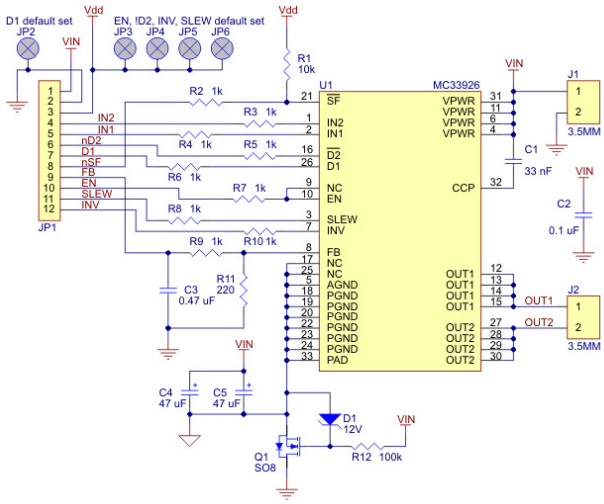

Figure I

One stepper motor (Minebea PM55L-048) in the linkage system will be moving and lifting a load greater than one pound. The motor takes 384 microsteps to complete a revolution. The linkage arm moves approximately 90 degrees, or 96 steps. The stepper and its torque output are shown in figures J and K respectively.

Figure J

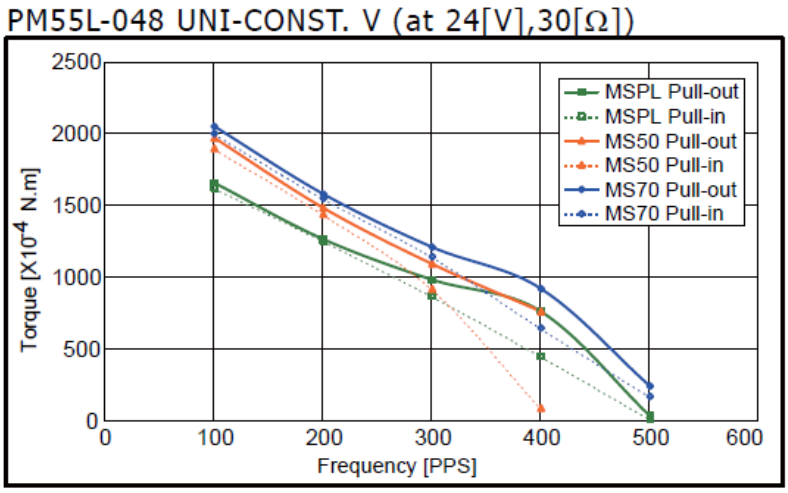

Figure K

The stepper is mounted within the linkage arm. This area of the platform also had to be reinforced to prevent gear slippage. Torque calculations have only been approximated and direct experimentation was necessary to verify operability, but PME has passed all tests. The stepper motor is being driven by an Easy stepper motor driver from sparkfun. The driver is shown in figure L. Figure M shows the schematic for connections from the driver chip to the driver board. Figure N shows a graph of the function of microstepping as related to full steps. Once fully extended, the stabilizing platform under the bottle will be rotated by a servo. This action will dump any contents of the grip arm. The robot will then back up, turn to the side, relocate the line, and continue following it. Figure O depicts a block diagram/schematic of the motor and driver connections.

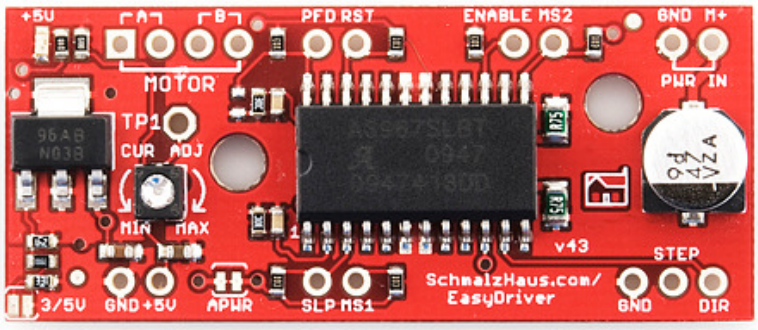

Figure L

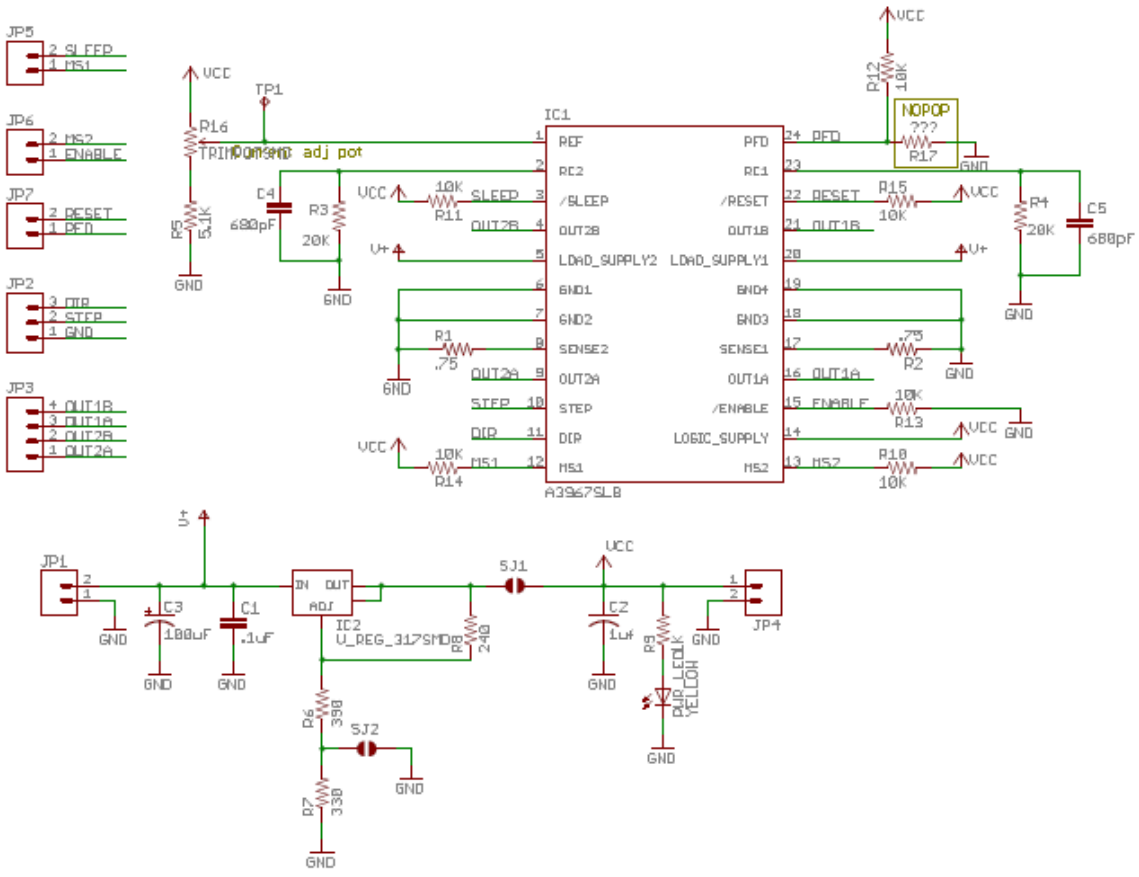

Figure M

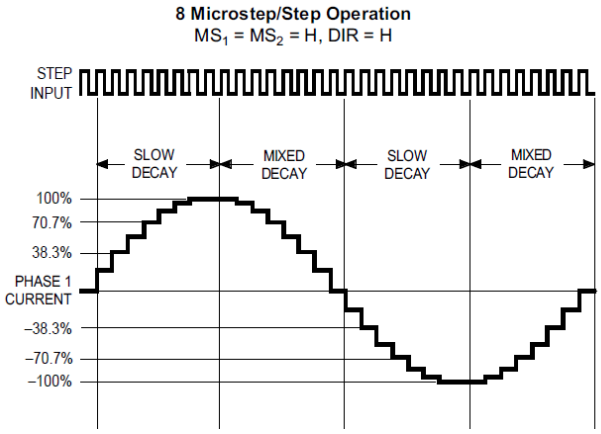

Figure N

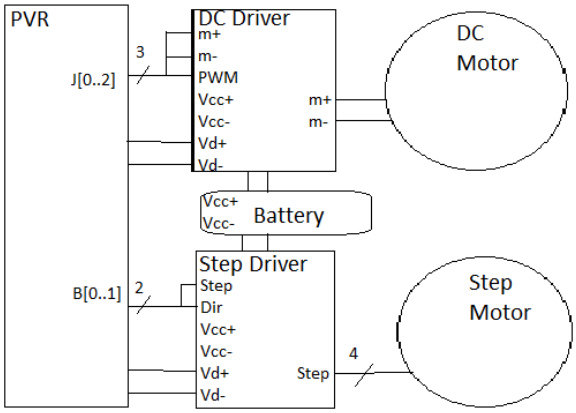

Figure O

Sensors:

PME is equipped with several different styles of sensors. All sensors are interfaced with the Xmega 128 PVR board from Pridgen/Vermeer Robotics(Figure P).

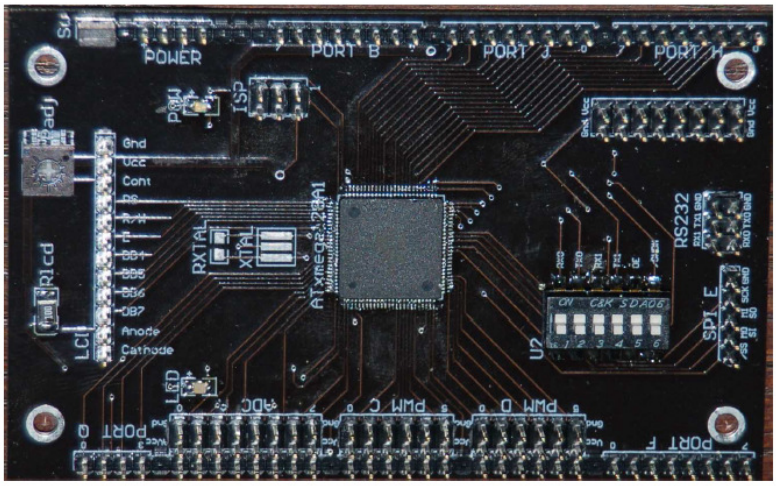

Figure P

The first sensor needed in order of function is one which will receive an activation signal. PME will be waiting in a dark area. Once a flashlight has targeted the Photoresistors mounted on the surface, it will begin line following mode. Two SEN-09085 photoresistors (figure Q) are wired in parallel, resulting in a received signal only when both resistors receive light. This will prevent false data from triggering behavior. A series of data points of photoresistor response was plotted in figure R. This data was taken in a well lit room as an obstruction was placed between the receiver and the light at varying distances. The distance is measured in inches.

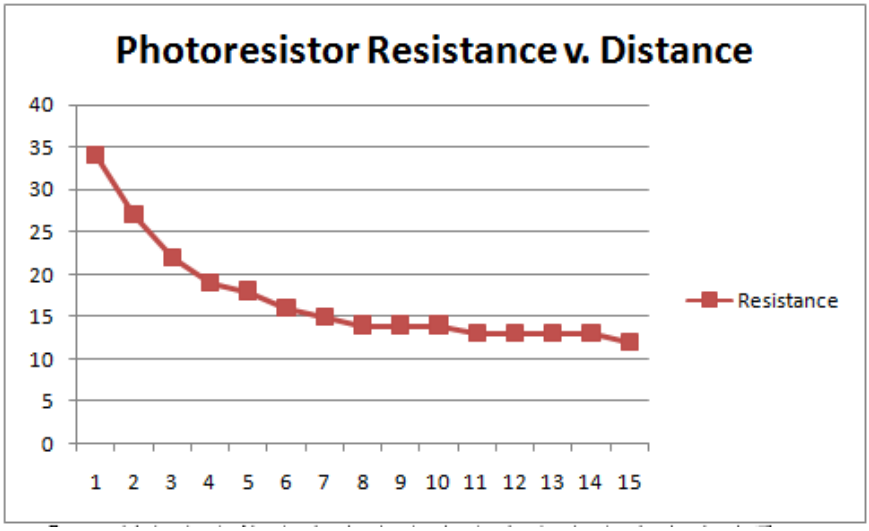

Figure Q **Figure R**

Upon receiving this light signal, the bot will begin line following mode until it reaches the first of its predetermined locations. Line following is achieved via an array of QTI IR reflective sensors mounted on the bottom of the platform. The sensors emit an infrared light and then measure the reflected light in an infrared transistor. A white or reflective surface will return a low voltage while a black or non-reflective surface will return a high voltage. A line of black masking tape is placed on sheets of white poster board in order to control the environment. The array of QTI sensors will then be able to determine which sensor is over top of the black line. Steering can then be determined accordingly. A picture of the line-following kit and a schematic of the sensor are shown in figures S and T respectively. These sensors return a value of approximately 4.3v when over a black surface, and 0.5 volts when over a white surface. To be effective, the sensors need to be less than ½" from the surface, or else all sensors will return non-reflective values.

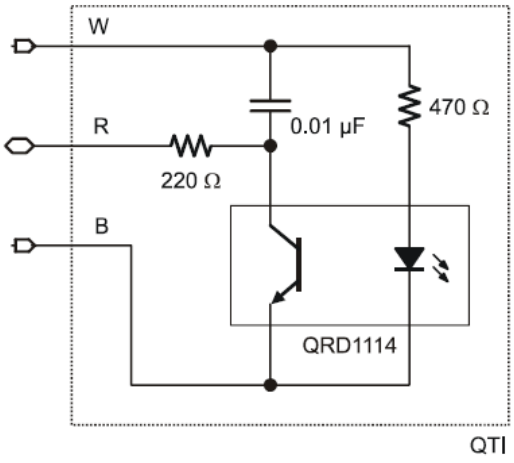

Figure S **Figure T**

Once it reaches its first destination, it will pause and wait for another signal. Another activation signal, pressing one of the bump sensors as a button, will cause the bot to go into line-following and obstacle avoidance mode.

Obstacle avoidance will be achieved through a combination of close-range infrared and bump detectors. Proximity warnings will alert the bot, but since the current task is not to deviate from the line, avoidance is challenging. The bot must be able to relocate the line after avoidance. The physical design of the machine also makes avoidance tedious. This should be accomplished with as little change in speed and direction as possible. Avoidance testing while not line-following has been successful. The bot can steer away from objects detected by infrared. It is also programmed to stop, back up, and continue in a new direction if a bump sensor is activated. Sharp 2D120X infrared sensors(figure U) have been placed on the front middle, front left, and front right of the platform. The sensitivity of the sensor has been tested under typical lighting conditions. The output voltage was measured as an obstruction was placed at varying distances from the sensor. The data is plotted in figure V, with the distance measured in inches. Bump sensors were constructed by mounting outward-facing springs on the platform and placing a wire within the spring. Contact between the spring and wire completes a circuit, through a pull-down resistor, and sends a value of high to the PVR board, indicating a bump.

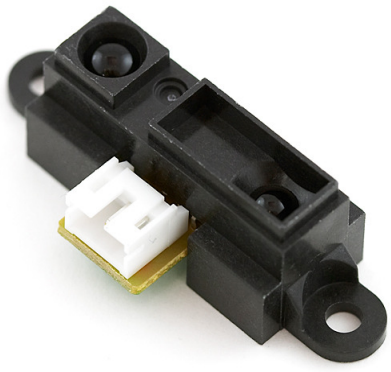

Figure U

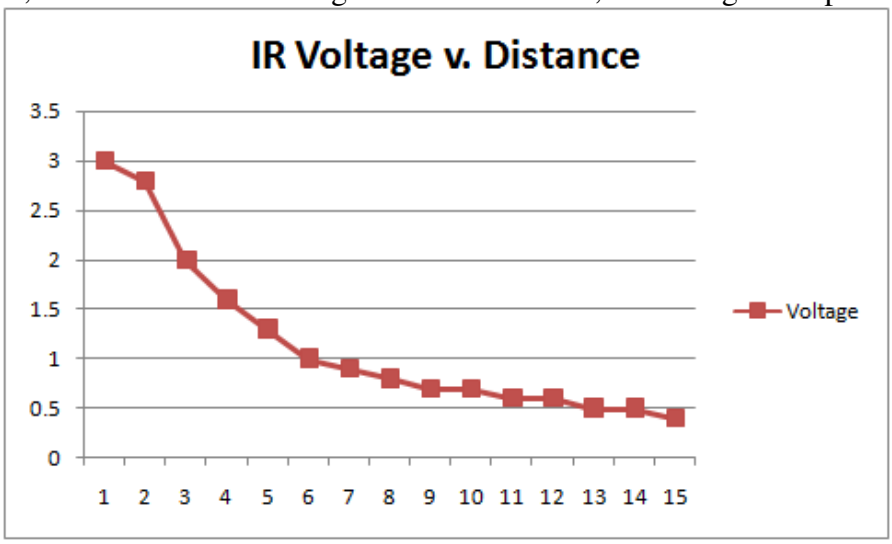

Figure V

To signal the machine that it has arrived at the designated disposal destination, a similar scheme will be employed as described in the initial line-following phase. When all QTI sensors detect a non-reflective service, the machine will know its destination has been reached. The motion stops and at this point no further sensors are required. The movement of the mechanical arm has been set by predetermined motor parameters. Once the cargo is disposed of, line-following will resume as the bot returns home. The bot will detect its home again by light recognition. When it finds itself in the shade, it will loop to the beginning of the program and wait.

Behaviors:

The behaviors of PME include understanding and responding to being summoned, gripping objects, line following, obstacle avoidance, and mechanical lifting. Initiating tasks will be signaled by sensor input. The bot will execute an endless loop waiting for a light sensory input. Once this input is detected, the bot will commence locating mode. Line sensors will provide feedback on the location of a person who will be advanced upon. The bot will know to stop when it has detected a certain surface.

Object gripping was to be performed by connecting a servo motor to the wire controlling a gripper arm. Turning the motor would tighten or loosen the grip. This design has been modified to eliminate the need for this servo. Instead, the wire from the gripper is attached to the platform in such a way that the wire becomes tightened when the arm is down and loosened when the line is up. As a bonus, this tension in the wire also assists in arm raising, reducing the torque requirements at the rotation point. Line following is made possible by photo sensors mounted on the bottom of the machine. A feedback system has been implemented so that the machine may modify its direction of travel while moving, corresponding to the direction of the line it is tracking. A flowchart for line-following behavior is shown in figure W. The abbreviations in the flowchart are as follows: NR=near right sensor, NL=near left sensor, FR=far right, FL=far left.

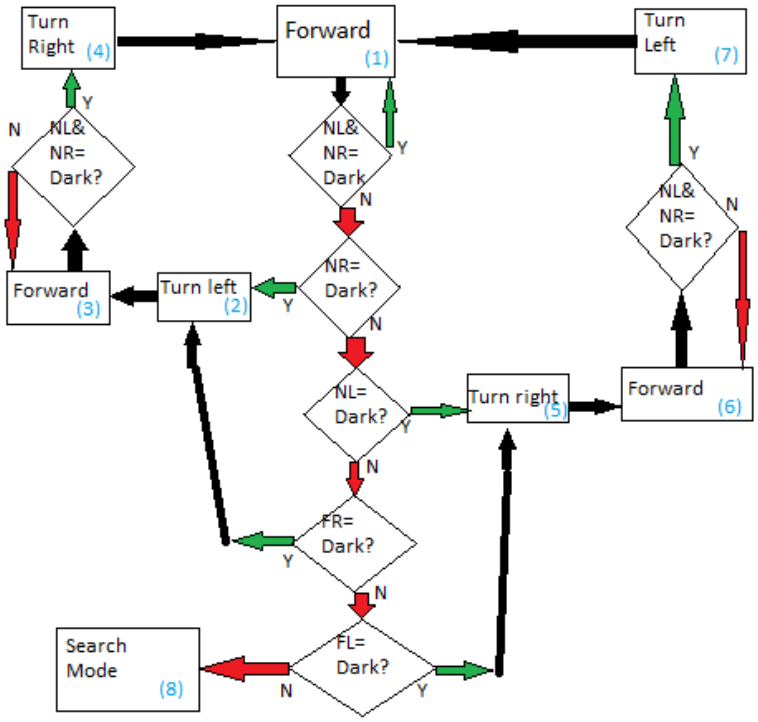

Figure W

Obstacle avoidance is very basic since the focus of the motion is line following. Infrared and bump sensors are used in parallel to notify the bot of any impending obstacles. Course will then be corrected accordingly. A simple back-up and turn will be executed if any bumps are detected. A turn while continuing forward will be executed when any objects are detected by the infrared. Relocation of the line, and determination of proper direction after avoidance, is challenging and has not yet been implemented. Perhaps avoidance will include wall following behavior as the bot navigates its way around an object until it relocates the line. A flowchart describing obstacle preliminary avoidance behavior is depicted in figure X.

Lastly, lifting will be achieved by a 4 bar linkage arm with a single stepper motor. This single motor accomplishes both horizontal and vertical movement. When fully extended, the support platform under the bottle and grip arm will be lowered by an additional servo, thereby dropping the contents of the claw in the container underneath. After a timed delay, the platform will revert to its upright position and the linkage arm will be lowered. The bot will then back up and turn and relocate the line in order to reinstate its line-following behavior. Lift and dump behavior is described in the flowchart in figure Y.

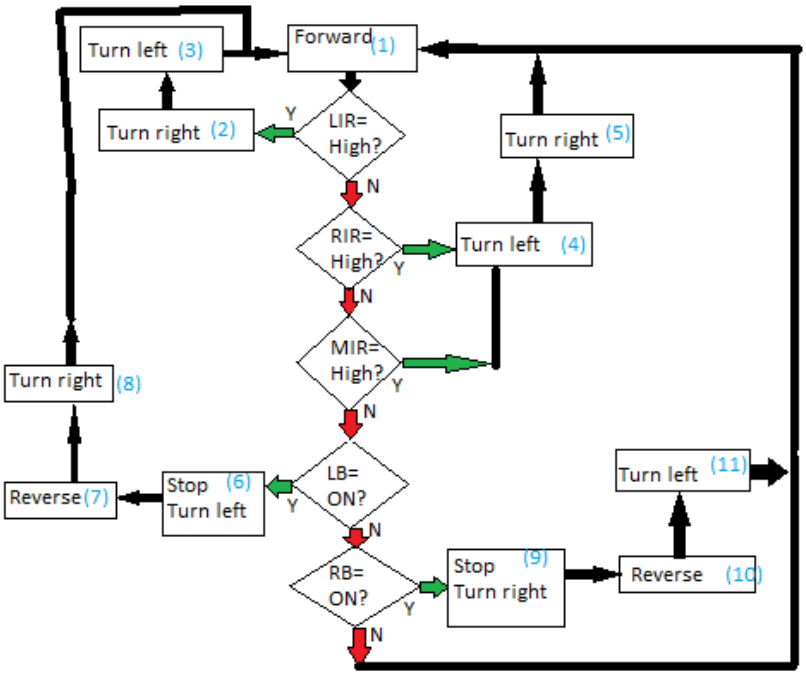

Figure Y

Experimental layout

Experimentation was necessary for the electrical, mechanical, and coding aspects of this project. The first experimenting encountered was the debugging of code. The code was written to accomplish several tasks. The first is to recognize its input signals. These signals include activation and all sensor data. When data is written to a port, the code then needs to properly recognize and process that data. This is the behavior step. All code was tested to ensure that, based on every conceivable combination of input parameters, the program responds optimally by sending the proper signals to the outputs, the final task. Data sent to the outputs needs to be in a form recognizable by the receiver of the signal. Experimentation here includes checking proper timing.

Once code was functioning as desired, the sensors and motors were attached and tested. When the sensors were connected, a variety of inputs was given to measure the relative response from the sensor. This information was used to calibrate the code. Motors needed to be checked for proper speed and power delivery.

When all inputs and outputs were properly functioning, the last stage of experimenting was real-world testing. The electric components were installed on the body and the code was run to see how well the sensors and motors correspond to projected results. For the most part, this was extensive calibration as it became apparent the most efficient way of capturing data. For line following mode, experimentation was based on steady observation. Once improper function was observed, the cause of that error had to be determined based solely on the physical response of the robot. This was tedious because the thought process leading to the bad decision is not observable.

Conclusion

PME strings together a series of relatively simple individual tasks while capturing input data with several relatively simple methods. However, the integration of these methods yields the accomplishment of a much more demanding task.

To date, the physical work on this machine is complete. The most pressing obstacle is obtaining usable feedback from the line-following sensors. Body construction is complete with sensors mounted and a controller board was purchased fully assembled. The body is fully assembled, all torque requirements have been verified, all motors, gear boxes, drivers, and all other necessary equipment has been mounted. Behavior algorithms have been considered at length and all code is now complete. Some behavioral modification may still take place if consistent behavioral errors are identified.

Some progress was limited by space. The line course designed for this machine was a circle approximately 10 feet in diameter. The bot needs at least another foot outside the turns in order to not to hit walls. This much space turned out to be elusive, and the quantity of testing was limited by inadequate space. Small scale behaviors on course sections were used to simulate integrated behavior. But the pieces don't always integrate flawlessly. Several other limitations were based on power needs. Initially, only 9 volts were being used. But the DC motor driver drops to half power and half speed when the voltage falls under 8v. This meant that the batteries had to be changed frequently. The drive motor was considerably harder to instantiate then was predicted. A motor was tested when the first skeleton of the frame was completed and functioned properly. However, when body construction was complete, the torque output was no longer sufficient to move the bot. Two other motors were installed before an adequate one was found. Each motor replacement came with significant body modifications. Time lost here could have been used for behavioral modification.

The stepper motor was very difficult to integrate. The first design was not sturdy enough and allowed the motor to push away from the other gear and slip past it. The motor was then more firmly locked into place and stopped working entirely. I thought this indicated insufficient torque. It turns out that it was not locked to firmly into place and the shaft was not able to overcome the torque requirement that the assembly placed on it. Finally a middle ground was found that prevented the gears from slip while still maintaining motor actuation.

Line-following proved very difficult to achieve success consistently. In the end, relatively few lines of code were needed to implement this behavior. More elaborate code was attempted but ended up causing more problems than it fixed. It was discovered that its actually easier to change the course to meet the current behaviors of the vehicle than to change the behaviors to meet the course. By softening some sharp turns and understanding how the bot seemed to react in certain parts of the course, the course was modified until completion was successful.

In retrospect, many things could have been designed differently. Firstly, Erector Set parts may not have been the best choice for this job. They look pleasing and create a rigid frame, but that frame adds significant weight which demands a stronger motor. The parts are infinitely versatile when interacting with other erector set parts. The problem is that non-erector set parts are incredibly difficult to interface. The body was assembled with the thought that it would be easy to mount anything anywhere. However, this turned out not to be the case. A more thoroughly thought out initial design would have saved significant heartache. The system also needs more supply power. The alkaline batteries should be replaced with rechargeables capable of delivering significant current. The motor driver boards should also be redesigned to give 12 volts to the dc motor driver and up to 28 volts to the stepper motor driver. With more voltage, it may have been able to lift a heavier load.

It would also be nice to be able to move the line-following sensors farther toward the front of the bot. The current design places them behind the steering axle. There is no room on the body to move

these sensors where they will not interfere with steering. Perhaps the steering axle could be moved farther back to allow clearance while turning. Although the bot still successfully follows lines, it would be smoother and more visually appealing if the sensors were moved forward.

Appendix

Hardware + Dcumentation:

Motor drivers: MC33926 Pololu 5A dc driver
http://www.pololu.com/file/download/MC33926.pdf?file_id=0J233
http://www.pololu.com/catalog/product/1212

Easy stepper motor driver
http://www.sparkfun.com/datasheets/Robotics/A3967.pdf
http://schmalzhaus.com/EasyDriver/

Infrared sensors: Sharp 2D120X - F – 04
http://www.sparkfun.com/datasheets/Sensors/Infrared/GP2D120XJ00F_SS.pdf

Photoresistor: SEN-09088
http://www.sparkfun.com/datasheets/Sensors/Imaging/SEN-09088-datasheet.pdf

QTI sensor: Parallax 555-27401: QTI IR reflectivity sensor line following kit
http://www.parallax.com/Portals/0/Downloads/docs/prod/robo/28108-QTILineFollower-v2.0.pdf

Servo Motors: TowerPro MG995
http://www.google.com/products/catalog?q=towerpro+mg995&hl=en&client=firefox-a&hs=tLF&rls=org.mozilla:en-US:official&prmd=ivnsfd&biw=1280&bih=607&bav=on.2,or.r_gc.r_pw.&um=1&ie=UTF-8&cid=17970671049710857146&sa=X&ei=lh-qTeyaB8LKgQei-tjzBQ&ved=0CFMQ8wIwAA#

Stepper motor: Minebea PM55L-048 750mA microstepper
http://www.eminebea.com/content/html/en/motor_list/pm_motor/pdf/pm55l048.pdf

DC motor: Pololu 29:1 metal gearmotor, 3A, 15v, 500RPM
http://www.pololu.com/catalog/product/1102

Frame: Erector Set

Board: PVR Xmega128
http://www.atmel.com/dyn/resources/prod_documents/doc8067.pdf

Prototype Code:

```
//Erik Stegman
//EEL 4665
//PME
//Final demo code

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

#define Left_Bump (PORTF_IN & 0x02)          //Left bumper set to pin 0
#define Right_Bump (PORTF_IN & 0x01)         //Right bumper set to pin 2
#define Back_Bump (PORTF_IN & 0x04)          //Rear bump sensor set to pin 3
#define All_Bump (PORTF_IN & 0x07)
#define     Left_Eye (ADCA2())                   //Left IR set to pin 0
#define     Mid_Eye (ADCA1())                    //Middle IR set to pin 1
#define     Right_Eye (ADCA0())                  //Right IR set to pin 2
#define     Shade (ADCA3())                      //Photoresistors set to pin 3
#define Line_FR (ADCA7())                    //Far right line sensor
#define Line_MR (ADCA6())                    //Middle right line sensor
#define Line_ML (ADCA5())                        //Middle left line sensor
#define Line_FL (ADCA4())                    //Far left line sensor
#define     Drive (PORTJ_OUT)                    //Set drive control to pins 1,0 (+,-)
#define Steer (ServoD0)                      //Set Steer to servoD0
#define Dump (ServoD1)                       //Set dump to servoD1
#define Arm (PORTH_OUT)                      //Pin 0= Step, Pin 1= Direction
#define LED (PORTQ_OUT)                      //Feedback/debug output LEDs
#define Forward (0b10)                       //Pin values for drive forward
#define Reverse (0b01)                       //Pin values for drive reverse
#define Stop (0b00)                          //Pin values to turn off motor
#define Left (40)                            //Servo value for steer left
#define Right (-50)                          //Servo value for steer right
#define Near_Left (20)                       //Servo value for steer left
#define Near_Right (-30)                     //Servo value for steer right
#define Center (-4)                          //Servo value for steer straight
#define On (0x1)
#define Off (0x0)
#define Up (-25)                             //servo value for raise dump
#define Down (-85)                           //servo value for lower dump

//INITIALIZE
void main(void)
{
    xmegaInit();                             //setup XMega
    delayInit();                             //setup delay functions
    ADCAInit();                              //setup PORTA analong readings
    PORTB_DIR |= 0xff;                       //set PORT B to OUT
```

```
	ServoCInit();				//setup PORTC Servos
	ServoDInit();				//setup PORTD Servos
	PORTF_DIR |= 0x00;			//set PORT F to IN
	PORTH_DIR |= 0xff;			//Initialize PORT H for debugging output
	PORTJ_DIR |= 0xff;			//set I/O PORT J
	PORTQ_DIR |= 0x0f;			//set Q (LED) as output for debug

	lcdInit();				//setup LCD on PORTK
	lcdString("Erik Stegman");		//display "Erik Stegman" on top line (Line 0) of LCD
	lcdGoto(1,0);				//move LCD cursor to the second line (Line 1) of LCD
	lcdString("PME");			//display "PME" on second line

while(1)
	{
	Steer(Center);
	Dump(Up);

//wait for summon
	while(Shade < 2500)
	{
		lcdGoto(0,0);
		lcdString("Shade");
		lcdGoto(1,0);
		lcdInt(Shade);
		lcdString("     ");
	}

//begin line follow
//look for black box at summoner
	Drive = Forward;
	delay_ms(1000);
	Follow();
	Drive = Stop;

//lift arm, wait, lower
	Lift();

//begin line follow
//look for black box at garbage
	Drive = Forward;
	delay_ms(1000);
	Follow();
	Drive = Stop;

//lift, dump, lower
	Lift_Drop();
```

```
//find line
        Find_Line();

//begin line follow
//look for black box at home
        Follow();
        Drive = Stop;
        }
}

void Follow(void)               //Line following function
{
        while(Line_FL < 3500 || Line_ML<3500 || Line_FR < 3500 || Line_MR<3500)
        {
        int i;

        if (Line_FL > 3500){LED |=8;}               //LEDs show what sensor the line is under
        else {LED &= 7;}
        if (Line_ML > 3500){LED |=4;}
        else {LED &= 11;}
        if (Line_MR > 3500){LED |=2;}
        else {LED &= 13;}
        if (Line_FR > 3500){LED |=1;}
        else {LED &= 14;}

        Steer(Center);
        if (Line_FL > 3500 && Line_ML<3500)
        {
        for(i=Center; i<Left && Line_ML<3500; i++)
                {                                   //turn left until line is under center
                Steer(i);
                delay_ms(10);
                }

        while(Line_ML<3500){}

        //straighten out
        for(; i>Center; i--){
                Steer(i);
                delay_ms(10); }
                }

        else if (Line_FR > 3500 && Line_MR<3500)
        {
                //steer right
                for(i=Center; i>Right && Line_MR<3500; i--){    //turn right until line is under center
                Steer(i);
```

```
                delay_ms(10);
                }

        while(Line_MR<3500){}

        //straighten out
        for(; i<Center; i++){
                Steer(i);
                delay_ms(10); }
                }
        }
}

void Lift(void)                         //Lift arm function
{
        ServoD1(Up);

//Raise Arm
        int i;
        PORTH_OUT = 2;
        for(i=0; i<330; i++)
                {
                PORTH_OUT += 1;
                delay_ms(20);
                PORTH_OUT -= 1;
                }

//Wait for signal from bump sensor
        while(All_Bump < 1){}

//Lower Arm
        PORTH_OUT = 0;
        for(i=0; i<330; i++)
                {
                PORTH_OUT += 1;
                delay_ms(20);
                PORTH_OUT -= 1;
                }
        delay_ms(2000);
}

void Lift_Drop(void)                    //Lift and dump function
{
//Raise Arm
        int i;
        PORTH_OUT = 2;
        for(i=0; i<330; i++)
```

```
                {
                PORTH_OUT += 1;
                delay_ms(20);
                PORTH_OUT -= 1;
                }
        delay_ms(500);

//Lower Dump
        for(i=Up; i>Down; i--){
                ServoD1(i);
                delay_ms(10); }
        delay_ms(500);

//Raise Dump
        for(i=Down; i<Up; i++){
                ServoD1(i);
                delay_ms(10); }
        delay_ms(500);

//Lower Arm
        PORTH_OUT = 0;
        for(i=0; i<330; i++)
                {
                PORTH_OUT += 1;
                delay_ms(20);
                PORTH_OUT -= 1;
                }
        delay_ms(2000);
}

void Find_Line(void)                    //Function to back up and locate line
{
        int i;

//Steer Right
        for(i=Center; i>Right; i--){
                Steer(i);
                delay_ms(10);
                }

        Drive = Reverse;
        while(Line_FL>3500 || Line_ML>3500 || Line_FR>3500 || Line_MR>3500){}
        while(Line_FL<3500 && Line_ML<3500 && Line_FR<3500 && Line_MR<3500){}
        Drive = Stop;

//straighten out
        for(; i<Center; i++){
```

```
            Steer(i);
            delay_ms(10); }

        Drive = Forward;
        delay_ms(2000);

        while(Line_FL<3500 && Line_ML<3500 && Line_FR<3500 && Line_MR<3500){}
}
```