# Final Report

*Date:12/8/10*
*Student Name:* David Register
*Robot Name:* BevBot
*TAs :* Mike Pridgen
Tim Martin
Ryan Stevens
Thomas Vermeer
Devin Hughes

*Instructors:* Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

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

**Table of Contents**

**Abstract**

BevBot is a beverage retrieval robot. When activated by the user it will search for a canned beverage and safely navigate to a beverage's location. As it does so it will avoid any obstacle in its path. It will then secure the beverage and carry it back to the user's location.

**Executive Summary**

BevBot's function it to retrieve a beverage for a user. To do this the robot utilizes a mobile platform, actuating arms, a CMUcam1 camera, two long range infrared sensors, and other necessary components to perform each behavior necessary to perform it's task. Once activated and all initialization is done BevBot, which is propelled by a pair of hacked high-torque servos, uses its camera to find the beverage's location, the location of which is further indicated by illuminated LEDs. While attempting to find the beverage the robot utilizes the infrared sensors to avoid any obstacles in its path. Once the camera spots the beverage's location, Bevbot uses the camera to guide it to the location and stops when the infrared sensors indicate it is in position. It then uses another pair of servos to actuate a pair of arms to secure the beverage. BevBot then maneuvers away from that location with the beverage and navigates to the user's location, further indicated by a LASER pointer, in the same manner as it navigated to the beverage. Once the camera and infrared sensors indicate BevBot has arrived, the robot will stop to allow the user to take possession of the beverage.

**Introduction**

The robot I am building is a beverage retrieval robot, or BevBot. Upon activation the robot will navigate to the beverage to be served, which will be in aluminum cans for the purpose of this project. When BevBot arrives at the beverage it will secure it and carry it back to the location of the user. Once there it will stop to allow the user to receive the beverage. All of the functions required to complete the task will require various actuators, sensors, controllers, and other hardware and software. The sections to follow will elaborate on all of these requirements and the progress that has been made to implement them.

**Integrated System**

BevBot can be broken down into several integrated component systems: a mobile platform, a beverage securing mechanism, sensors, a microprocessor board, a human-robot interface, and a DC power supply. Each component system provides necessary functionality to perform a portion of the beverage retrieval task. The human-robot interface is the means by which the user turns on the robot and data being used by the robot is displayed. This interface consists of a toggle switch and an LCD display. The

16 x 2 display is made by Xiamen Amotec Display Co. and is module number ADM160SK-NSW-FBS/3.3V. The mobile platform houses all the other components and provides a means of transporting the robot between the location of the user and the beverage being retrieved. The sensors, which include a CMU camera, infrared sensors, and bump sensors, enable the robot to enable it to receive input from its surroundings, such as the location of obstacles or the colors of the beverage containers in the bin. The function of the beverage securing mechanism is self-explanatory. All of these components are controlled by a Pridgen Vermeer XMega128 microprocessor board (PVR board) and powered by a DC power supply housing six nickel metal hydride batteries. A picture of the robot is in Appendix A.

**Mobile Platform**

A vital element to the performance of BevBot's task is mobility. The robot needs to be capable of traveling from the location the beverage is ordered to the bin containing the beverages and returning to the location of the order. This functionality is provided by a wheeled platform driven by hacked servos.

The platform consists of a base, top, support columns, drink holder, two actuating arms, component mounts, and a caster. The base and top of the robot are in the shape of circles with two opposite ends cut off. It is 12 inches across at its longest point and 10 inches across lengthwise. The base of the robot has the servo mounts, infrared (IR) sensors mounts, drink holder, PVR board, caster, and battery pack holder mounted to it. The top is positioned three inches above the base. It has the LCD display, on/off toggle switch and actuating arms mounted to it. The top is supported by four support columns. The support columns have ¾ inch tabs that go through holes in the top of the platform. This allows the top to be secured without epoxy, screws, etc. so the top may be removed to allow access to the components contained in the robot. The front drink holder is mounted to the bottom of the base on the front end of the robot. It provides a surface protruding from underneath the robot for the beverage to rest on during transport. Mounted opposite to it on the back end of the robot is a caster with a 2 inch diameter wheel. This caster provides freedom of motion and its size, mass, and location enable it to act as a counterweight for the robot when carrying a beverage. The servo mounts are mounted to the bottom of the base on either side and towards the front of the robot. They provide a surface to mount the servos used to drive the robot via 4 screws and nuts each.

The servos in use are HiTec HS-645MG Ultra Torque servos. Servos were selected because speed is not a design requirement and for the torque they can provide. This torque is needed to move the relatively large platform and all other components. The HS-645MG provides large amounts of torque for a servo (7.7 kg*cm or 107 oz*in at 4.8 V) and its metal gears provide the ruggedness needed to move the platform and withstand frequent direction changes. Each servo drives a 2.63 inch diameter wheel and tire from Lynxmotion. This wheel was designed to mount to a HiTec servo and has a rubber tire affixed to it to provide adequate traction to move the robot.

**Actuation**

The actuation performed by BevBot is related to robot mobility and beverage container manipulation. The actuation pertaining to mobility was discussed in the previous section. The other actuators on the robot are utilized to secure the beverage. After the robot has located the beverage and positioned itself for retrieval, it actuates a pair of arms that extend from the front of the robot to secure the beverage. The arms simultaneously swing inward and overlap on the opposite side of the beverage from the robot. Further moving these arms force the beverage to slide towards the robot and onto the drink holder. Each arm is moved by a HiTec HS-485HB servo. While this servo is not as powerful and lacks the metal gears of the HS-645MG that propels the entire robot, these servos provide relatively large amounts of torque (4.8 kg*cm or 66.65 oz*in at 4.8 V), have Karbonite gears the manufacture claim are four times stronger than typical white resin gears, and can be purchased at approximately half the price. Each arm is mounted to a servo and has one end anchored by a bolt that extends through the top and bottom levels of the platform. One arm fits more snugly on its bolt and is capable of maintaining the proper level of the arm. The other utilizes a duct tape around the bolt separated from the arm by a spacer to maintain the proper level. This level control is necessary to ensure the arms are on different levels and will not collide when they close.

**Sensors**

BevBot utilizes or will utilize several types of sensors to perform its task. These sensors include a CMU camera, IR sensors, and bump switches.

The special sensor for the robot is a CMU camera, specifically the Boe-Bot CMUcam1 AppMod Vision System from Parallax, Inc. This version of the CMUcam1 is modified for use with the Boe-Bot robot kit by leaving off the Maxim MAX232CPE level shifter chip and nine capacitors needed for the camera to perform RS232 serial communication and it came connected to a printed circuit board (PCB) that would allow the camera to interface with the Boe-Bot and the user to select and identify the current function of the camera. Once physically separated from the PCB and the missing components and a pair of jumpers were installed the camera was capable of operating like the regular version of the CMUcam1. For the BevBot the camera is used to track the location of a color, which is red for this application with the current programming. The camera locates an object of the correct color and provides a data packet that includes an X and Y coordinate for the object's location relative to the camera. The code found in Appendix B polls the camera for the locations of red objects, receives the data packet, pulls out the X and Y coordinate values, and turns the robot so that the red object is centered on the front of the robot. The camera is mounted to the top of the BevBot utilizing the hardware used to connect it to the PCB for the Boe-Bot. The hardware allows the camera to be tilted up and down as needed, and its location on top of the robot keeps it out of the way of other components and allows for an unobstructed view even when carrying a payload.

BevBot utilizes two IR sensor to measure the range from the sensor to an object. The IR sensors used are Sharp long range infrared proximity sensors, part number

GP2Y0A02YK0F. The sensors have a vendor specified range of 20 cm to 150 cm. They utilize a 5 V supply voltage obtained from the ADC port on the PVR board and return a value to the I/O pin ranging from approximately 500-700 if no object is detected to 4095 if an object is at the closest detectable range. This information is used for obstacle avoidance and positioning with the beverage during retrieval. The code currently utilized by BevBot to perform the obstacle avoidance, which is a modified version of the PVR sample code, can be found in Appendix B.

The use of bump sensors is an area for future work for the robot. Bumps sensors will be used to to detect when it has collided with an object that was not detected by the IR sensors so it may navigate away from it. Each bump sensor has four pins that may be connected to. On BevBot two pins diagonally across from each other will be used. The connection between these pins is normally open and closes when the bump sensor is pressed. One pin will be connected directly to ground, and the other will be connected to an I/O pin on Port J of the PVR board configured to use an internal pull-up resistor. A total of six bump sensors are intended to be used, three of which are installed on the sides and rear of the bottom level of the platform. The other three will be mounted on each side and on the rear of the top level of the platform.

**Behaviors**

Several behaviors are exercised by BevBot to perform its task. First the robot avoids colliding with obstacles as it searches for the beverage, which is located more easily by locating a pair of illuminated red LEDs directly above it. Once the robot sees the color red it will steer towards it until it is positioned in front of the beverage. Once there the robot secures the beverage. Then it navigates back to the user's location with the beverage and stops upon arrival.

The obstacle avoidance is currently provided by the IR sensors and the code in Appendix B. Action is taken when the IR sensor inputs a value greater than 2500, which corresponds to an object being within approximately 19 inches. The robot turns left if only the right IR sensor detects an object and turns right if only the left IR sensor detects an object. If both detect an object the robot turns left or right randomly. The robot will also randomly turn left or right after traveling a short distance until it locates a red object.

Once a red object is seen, the obstacle avoidance and search behaviors are temporarily disabled. If the object is located outside an allowable range to the left, which is indicted by an X value greater than 48, the robot turns to the left. The same action is taken for an object too far to the right, which has an X value less than 32. When the red object is centered in the allowable range the robot travels straight ahead. This continues until the IR sensors detect an object deliberately located behind the beverage. By detecting the object behind the beverage and the color red in the center, the robot knows it is positioned to secure the beverage.

Once the BevBot is properly positioned the actuating arms secure the beverage as described earlier. After this is completed the robot backs up, rotates 180 degrees and

begins to search again in the same manner as before. This search is aided by a LASER pointer held by the user. Once found the robot navigates to the user the same way it navigated to the beverage. When the robot gets close enough, indicated by both IR sensors reading greater than or equal to 3100, the robot stops so the beverage may be taken by the user.

**Experimental Layout and Results**

The layout of the robot can be seen in the photo in Appendix A. Few adjustments were made to the robot to arrive at this configuration. The first modification was to remove the PVR board, rotate it 180 degrees, and remount it. This was necessary so the wires from the IR sensors could reach the appropriate pins on the board. The other modification was to deviate from the original conceptual design and mount the servos for the actuating arms to the top level rather than the bottom level of the platform. This prevented the other components mounted on the bottom level from interfering with the mounting location of the servos, and it allowed the arms to swing at a lower level than had the servos been mounted to the bottom level.

A significant amount of experimentation was performed to confirm the operation of various components and to develop the performance of the various behaviors of the robot. First it was necessary to determine the values being provided by the IR sensors to write the code for obstacle avoidance. The PVR board was programmed to display the value from each sensor on the LCD display. Using the displayed values the specifics of operation were determined and incorporated into the code. At this point the robot demonstrated successful obstacle avoidance, but the arms had not been installed. The protruding arms in front of the robot required and adjustment to the avoidance code to turn sooner and prevent the arms from hitting objects.

Next the camera's functionality and its utilization had to be checked. Functionality was demonstrated by fabricating a connector for the camera to connect it to a computer and then showing the output of the camera in hyperlink. Next communication with the robot was demonstrated by outputting the data packet from the camera to the LCD display. From there the code to track red to direct the robot to the beverage was developed and functionality verified through trials.

The big issue was and continues to be the integration of all the behaviors into one comprehensive program. All the behaviors worked properly when performed separately, but when obstacle avoidance and searching were added to the program to track color, secure the beverage, move away, and track color again the robot would move only a couple feet before becoming confused and ceasing to perform any of the behaviors. Thus currently the obstacle avoidance and searching elements to the code as seen in Appendix B have been disabled to permit the other behaviors to be demonstrated.

**Conclusion**

BevBot largely has all the capabilities to perform its intended task, but some future work would need to be performed to perform all aspects of its task in one run. Physically BevBot is sturdily constructed with minimal future work. This future work includes attaching the remaining three bump switches and providing adequate pathways for and securing the wiring. The majority of the work needed to enhance the capabilities of the robot is with the software. Many of the individual robot behaviors have been successfully created, but they must be integrated into a single program in such a way as to not interfere with each other.

Few problems were experienced with the design of the robot. The biggest problem by far was with the CMU camera. Significant time was lost attempting to obtain the camera due to the unavailability of the CMUcam1 and having to resort to a lesser substitute. Given the time lost in obtaining the camera, repopulating the missing components, and making the camera operational, a better approach would have probably been to use the CMUcam2 or CMUcam3. Not only would time have been saved, but with the extra expenses required to make the Boe-Bot camera operation those options potentially would not have been significantly more expensive if they were more expensive at all. Another issue was that adequate tolerances for pieces fitting together were not made. Holes were made too small for tabs, and the rounded edges cut by the T-Tech rather than square edges did not fit properly at first. Some hand chiseling was required to enable a proper fit. Finally, the PVR board had to be turned to allow the wires from the IR sensors to reach the correct pins.

**Documentation**

Vendor documentation for components can be found online at the following locations:

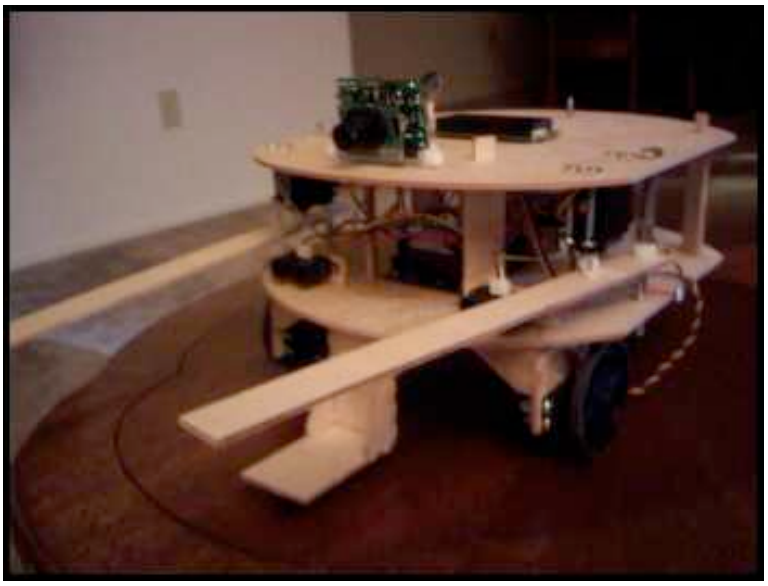
IR sensors: http://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

LCD display: http://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

CMU camera: http://www.parallax.com/Portals/0/Downloads/docs/prod/robo/CMUcam%20Manual.PDF

**Appendices**

Appendix A: BevBot Photo



Appendix B: Current code programmed into robot.

The entirety of the code used by BevBot, including header and source files, is too long to be included in this report. It can be found at the following web page: http://plaza.ufl.edu/dregister/Index.htm. The following is the main program:

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

void main(void)
{xmegaInit();                                   //setup XMega
delayInit();                                    //setup delay functions
ServoCInit();                                   //setup PORTC Servos
ServoDInit();                                   //setup PORTD Servos
ADCAInit();                                  //setup PORTA analong readings
lcdInit();                                      //setup LCD on PORTK

lcdString("BevBot");              //display "BevBot" on top line (Line 0) of LCD
lcdGoto(1,0);                  //move LCD cursor to the second line (Line 1) of LCD
//lcdString("Board Demo");                 //display "Board Demo" on second line

PORTQ_DIR |= 0x01;                              //set Q0 (LED) as output

int i = 0;
int irleft = ADCA2();
int irright = ADCA0();
int rightforward = 100;
int rightreverse = -100;
```

```
int rightstop = 6;
int leftforward = -100;
int leftreverse = 100;
int leftstop = 6;
int leftarmopen = 100;
int leftarmclosed = -50;
int rightarmopen = -100;
int rightarmclosed = 50;
int x = 0;
int y = 0;
int gotdrink = 0;

PORTE_DIR = PIN3_bm; //Pin 3 of port E is output
PORTE_OUT = PIN3_bm; //Pin 3 of port E is TXO
PORTE.DIRCLR = PIN2_bm; //Pin 2 of port E is RXO
USARTE0_CTRLC = 0x03; // USART Control Register C: ASYNCHRONOUS, no
parity 1 stop bit 8 bit word
USARTE0_BAUDCTRLA = 0x06; // Page 238 of Atmel manual, fbaud=115200 (my
case)=32MHz/(16*(((2^BSCALE) * BSEL)+1))
USARTE0_BAUDCTRLB = 0xC1; // BSEL= 262 and BSCALE= -4 in 2s comp .
USARTE0_CTRLB |= 0x08; // TX0 is on
USARTE0_CTRLB |= 0x10; // RX0 is on
static char *temp;

PORTJ_DIR |= 0x00;
PORTJ_PIN3CTRL |= 0b10011000;
PORTJ_PIN5CTRL |= 0b10011000;
PORTJ_PIN7CTRL |= 0b10011000;


//Program
CMUsend("RS\r"); //Reset the camera
temp=CMUreceive(); //Receive ACK
//        lcdString("ACK receive");
delay_ms (5000); //Long enough to wait for ACK and to get the camera ready
CMUsend("L1 1\r"); //To turn on green light
//        temp=CMUreceive(); //Receive ACK
//        lcdString("Made it here");
//        while(1){}
delay_ms(5); //For some reason that I havent figure out, there must be a delay between
commands
CMUsend("PM 1\r"); //Activate polling mode
temp=CMUreceive(); //Receive ACK
//        lcdString("ACK receive");
//        while(1){}
delay_ms(5);
```

```
while(1)
        {lcdData(0x01);                                                 //clear LCD
//      lcdString("Left IR ");          //display "Left IR " on top line (Line 0) of LCD
        irleft = ADCA2();               //get latest left IR sensor value
//      lcdInt(irleft);                 //display left IR sensor value on top line
//      lcdGoto(1,0);           //move LCD cursor to the second line (Line 1) of LCD
//      lcdString("Right IR "); //display "Right IR " on bottom line (Line 0) of LCD
        irright = ADCA0();                      //get latest right IR sensor value
//      lcdInt(irright);                //display right IR sensor value on bottom line
//      delay_ms(100);                  //delay 100ms

//      lcdData(0x01);                                          //clear LCD
        CMUsend("TC 155 255 0 30 0 30\r");                      //Track color
        temp=CMUreceive();                                      // Receive ACK
        temp=CMUreceive();                                      // Receive M packet
        delay_ms(5);

        lcdString(temp);

        CMUTC(temp, &x, &y);
        lcdGoto(1,0);           //move LCD cursor to the second line (Line 1) of LCD
//      lcdInt(x);
//      lcdString("  ");
//      lcdInt(y);

        delay_ms(5);                            //delay 200ms
if (x != 0)
        {if (irleft >= 3100  && irright >= 3100)        //If the left IR sensor detects an
                                                        object within range...
        {ServoC0(leftstop);
        ServoC4(rightstop);                     //  Stop moving.
        ServoD0(leftarmclosed);
        ServoD3(rightarmclosed);                        //  Close arms.
        gotdrink = 1;
        delay_ms(3000);                                 //delay 3000ms
        ServoC0(leftreverse);
        ServoC4(rightreverse);
        delay_ms(1200);
        ServoC0(leftforward);
        delay_ms(2200);
        }
else
        {if (gotdrink == 0)
        {ServoD0(leftarmopen);
```

```
        ServoD3(rightarmopen);}                 //  Leave arms open otherwise.
        ServoC0(leftforward);
        ServoC4(rightforward);
        }
/*
        if (y >= 135)
        {ServoC0(leftstop);
        ServoC4(rightstop);
        }
        else
                {ServoC0(leftforward);
                ServoC4(rightforward);
                }
*/
if (x <= 32  && x != 0)         //If both IR sensors detect an object within range...
        {//lcdGoto(1,0);        //move LCD cursor to the second line (Line 1) of LCD
//      lcdString("turn right");
        ServoC4(rightstop);
//      delay_ms(5);
        }

if (x >= 48)                //If the random function (range 0 to 32677) is greater than 16338
        {//lcdGoto(1,0);        //move LCD cursor to the second line (Line 1) of LCD
//      lcdString("turn left");
        ServoC0(leftstop);
//      delay_ms(5);
        }
/*      else                //If the random function (range 0 to 32677) is less than 16338
        {//lcdGoto(1,0);        //move LCD cursor to the second line (Line 1) of LCD
//      lcdString("go straight");                               //delay 200ms
        ServoC0(leftforward);
        ServoC4(rightforward);
        delay_ms(5);
        }*/
}
else
{       if (gotdrink == 0)
                {ServoD0(leftarmopen);
                ServoD3(rightarmopen);}     //  Leave arms open otherwise.

        ServoC0(leftforward);
        ServoC4(rightforward);

        if (irleft >= 3100  && irright >= 3100)         //If the left IR sensor detects an
                                                        object within range...
        {ServoC0(leftstop);
```

```
        ServoC4(rightstop);                 //  Stop moving.
//      ServoD0(leftarmclosed);
//      ServoD3(rightarmclosed);                    //  Close arms.
//      delay_ms(3000);                     //delay 3000ms
                }
/*      else
                {ServoD0(leftarmopen);
                ServoD3(rightarmopen);      //  Leave arms open otherwise.
                }
*/

/*if (irleft >= 2200  && irright >= 2200)   //If both IR sensors detect an object within
                                            range...
        {if (random() > 16338)              //If the random function (range 0 to 32677)
                                            is greater than 16338
        {ServoC0(leftforward);
        ServoC4(rightreverse);                      //  Turn right.
        delay_ms(600);                              //delay 200ms
        }
else    //If the random function (range 0 to 32677) is less than 16338
        {ServoC0(leftreverse);
        ServoC4(rightforward);                      //Turn left
        delay_ms(600);                                      //delay 200ms
        }
}
else                                    //If not on positive half of servo range...
        {ServoC0(leftforward);
        ServoC4(rightforward);
        }


if (irleft >= 2200  && irright <= 2500)   //If the left IR sensor detects an object
                                            within range...
        ServoC4(rightreverse);                      //Reverse right wheel.
else                                                //Otherwise...
        ServoC4(rightforward);                      //Right wheel goes forward.

if (irright >= 2200 && irleft <= 2500)    //If the right IR sensor detects an object
                                            within range...
        ServoC0(leftreverse);               //Reverse left wheel.
else                                                //Otherwise...
        ServoC0(leftforward);               //left wheel goes forward.
*/
/*if (i == 20)            //If both IR sensors detect an object within range...
        {if (random() > 16338)          //If the random function (range 0 to 32677) is
                                        greater than 16338
```

```
                {ServoC0(leftforward);
                ServoC4(rightreverse);                  //  Turn right.
                delay_ms(600);                          //delay 200ms
                }
        else            //If the random function (range 0 to 32677) is less than 16338
                {ServoC0(leftreverse);
                ServoC4(rightforward);                  //Turn left
                delay_ms(600);                                          //delay 200ms
                }
        i = 0;
        }
        i = i + 1;*/
}
free(temp);

        }}
```