FINAL REPORT:

ROGER RODRIGUEZ

ROBOT'S NAME: PACKY

COURSE - EEL5666

IMDL FALL 2011

INSTRUCTORS:

DR. ARROYO

DR. SCHWARTZ

TAs:

RYAN STEVENS

TIM MARTIN

Contents

1. ABSTRACT

The idea for the project is to design a robot capable of locating objects in its environment and distinguish them based on a special color (as detected by an IP Camera), after that the robot must approach said target and pick it up using a gripping mechanism, once that task is complete, the robot shall proceed in packaging said object, so it can finally store it or place it in a designated storage area.

The Robot uses a replicate system based on a novel gripping mechanism developed at the university of Chicago and Cornell University, to complete gripping tasks.

2. EXECUTIVE SUMMARY

Packy is a mobile autonomous robot designed to locate an object of a certain color, it accomplishes this task by relying in a vision system comprised of a wireless camera, by means of an app running on an android operating system powered phone, and remote processing by means of a wirelessly connected computer, once the object is located the robot will approach said target and proceed to obtain an object of a certain size but of indeterminate shape by means of a “universal” gripper system, based on a novel research carried out by scientists at the university of Chicago and Cornell university, this “universal” gripper is capable to conform to different shape objects and pick the object up. Once the object has been picked up the robot will place it in a plastic bag located in the back of its structure. All the behaviors required for execution of the tasks are processed locally in the robot by means of a microcontroller board with servo control and dc motor control capabilities.

3. INTRODUCTION

This robot explores several aspects of autonomous agents, and artificial vision and micro-controlled systems. It was designed to locate objects of a selected color, pick them up using a universal gripper, and places them in a plastic package. The gripper is based on the interlocking property of granular materials when subject to vacuum, called jamming transition. (The gripper is based on a novel idea by researchers of the University of Chicago and Cornell University.).

4. MAIN BODY

4.1 INTEGRATED SYSTEM

The platform for this robot is a controller board developed by POLOLU called “ORANGUTAN SVP 1284P” it is an AVR ATMEGA1824P microcontroller based board.

Basic components:

- A servo controller capable of controlling up to 8 servos
- And dual dc motor drives.
- As well as an LCD display.
- And user configurable pushbuttons.
- A small piezo speaker.
- On board ISP programmer
- JTAG connector.

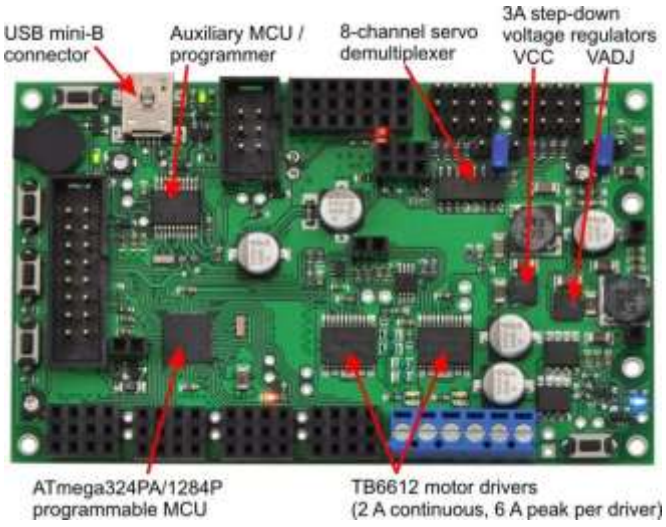
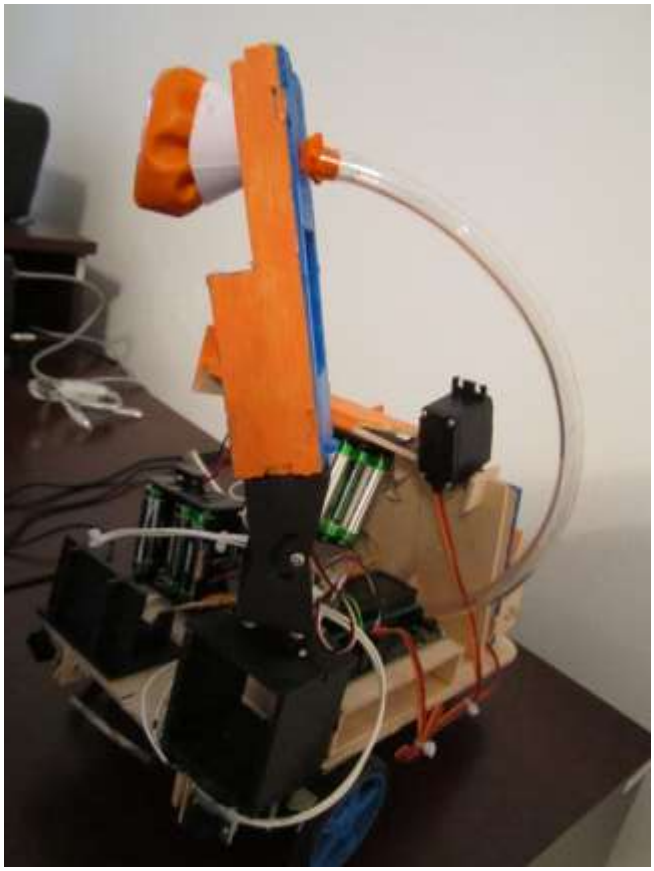


IMAGE FROM WWW.POLOLU.COM

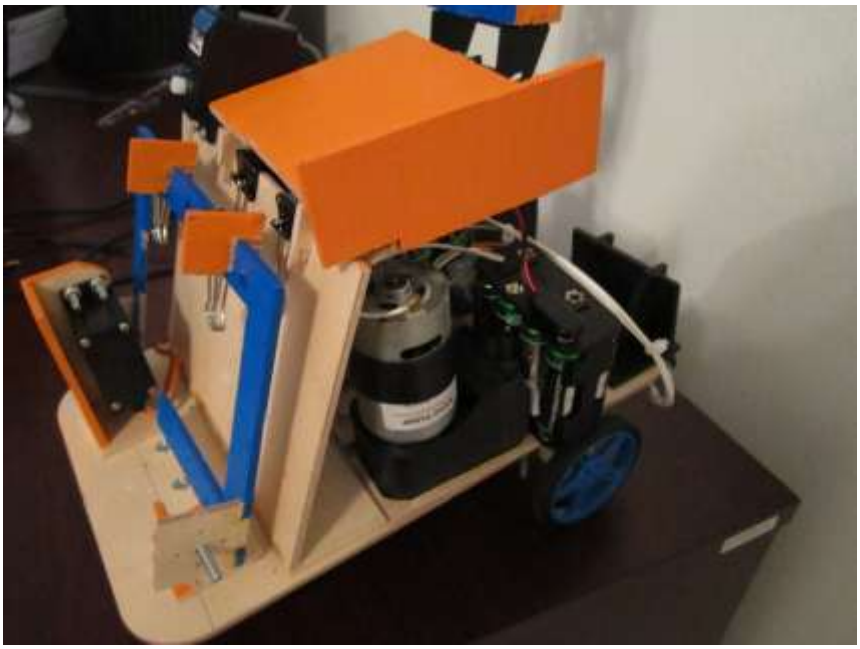
The design of the platform is oriented towards control of medium sized robots, it can be programmed using the AVR studio ide and is compatible with customized libraries created by POLOLU that contain the basic functions to take advantage of the available features of this particular board.

4.2 MOBILE PLATFORM

As a starting point there will be a standard wooden platform with the frame to hold the electronics and the motors, the packaging section of the robot shall be modeled as a sort of chamber with a rotating base and an arm that positions and holds the package in place for the packaging to begin.



It was necessary to design this project in a modular fashion, meaning that the mobile part of the robot will be the base platform that will contain and transport the packaging module plus the gripping arm. So far weight considerations have not been established for each module, but further along the design phase, the system weight will influence the DC motor selection, so the effective weight of the system must be kept at a possible minimum.



4.3 ACTUATION

The platform will be a mobile autonomous agent that will be based in the following the actuators:

- 2X DC MOTORS (GEARED DOWN) INTENDED FOR GENERAL LOCOMOTION OF THE ROBOT AND DISPLACEMENT.

IMAGE FROM SPARKFUN.COM

- 8X SERVO MOTORS (SMALL AND MEDIUM SIZED) INTENDED FOR GENERAL MECHANIC ACTUATION AND TASK COMPLETION.

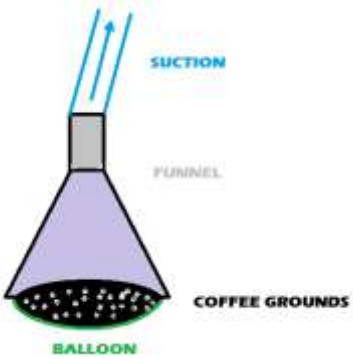
IMAGE FROM SPARKFUN.COM

- SOLENOID, INTENDED FOR OBJECT PUSHING AND POSITIONING.
- SUCTION GRIPPER (A NOVEL IDEA BASED ON A UNIVERSAL GRIPPER DEVELOPED BY THE UNIVERSITY OF CHICAGO) **[1]**
- THE BASIC IDEA OF THE UNIVERSAL GRIPPER IS TO EXPLOIT THE PROPERTIES OF A PHENOMENON CALLED JAMMING TRANSITION, IN WICH GRANULAR PARTICLES CAN CONFORM TO THE SURFACE OF AN OBJECT AND FORM A TIGHT SEAL ONCE THE AIR HAS BEEN TAKEN OUT, THE PARTICLES ARE TIGHTLY INTERLOCKED AND CAN GRIP ANY OBJECT THAT WITHIN THE CONFORM AREA.

- THE SUCTION GRIPPED IS COMPOSED OF SEVERAL PARTS, MAINLY:
  A FUNNEL

  BALLOON FILLED WITH COFFEE GROUNDS

  SUCTION PUMP – 12V 1WATT DC MOTOR POWERED SUCTION PUMP



4.4 SENSORS

Initially the proposed sensors will be:

IR emitters and detectors for package recognition (beacons)

Image from sparkfun.com

Sonar range finders (for obstacle avoidance and proximity detection)

Image from sparkfun.com

Bump switches (obstacle avoidance and package presence)

IP Camera – An alternative camera has been found in the form of an android applications running on top of a Motorola smartphone, this mobile application enable to capture images from the phones camera and send them over a wireless IP network, this is a significant cost saver because it reduces the total cost of the robot using a sensor that is already available and working. Another benefit of this is the reduced energy consumption of the battery, given that the phone is powered by its own internal battery there is no need to draw current from the robots battery, thus increasing available power for the robot.

Images from Motorola.com

Taking into account suggestions and input from the class instructors, it seems to be more reliable to use an on board camera for object detection and navigation, instead of using IR beacons, given that it would become complicated and impractical.

4.5 BEHAVIORS

The initial state of the robot is to seek out for packages and identify them as possible targets, once an object has been located; the robot must proceed to move towards its target, while adjusting its trajectory to compensate for any deviations.

The location of the object is to be performed with the aid of an IP camera, an android smartphone enabled by an internal application to work wirelessly using Wi-Fi.

Once the robot approaches the target it must determine the proximity of said object using and IR sensor and proceed to obtain it by means of a gripping mechanism, once the object has been gripped and confirmed to be in possession of the robot, the packaging process must commence.

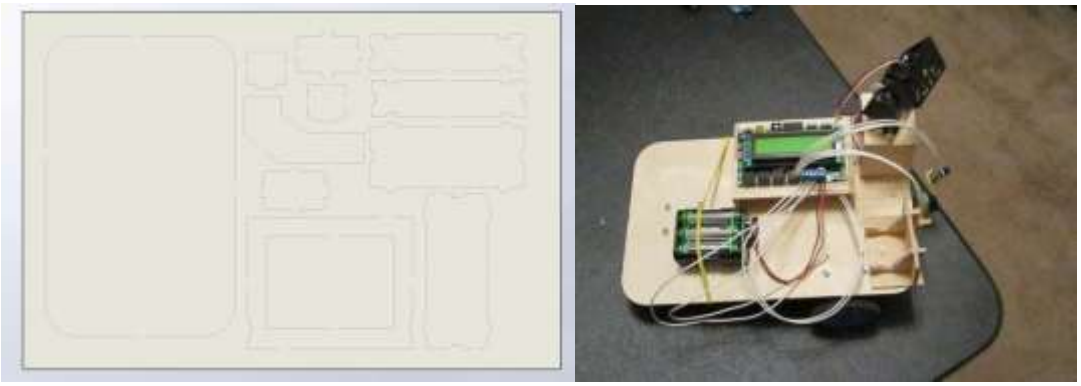
The lack of mechanical details for this stage of the actuation, make it difficult to describe in detail the procedure to be followed once the robot has acquired the object, further along this section will be updated to reflect the designed mechanisms.

As a final task the robot must again seek a suitable container in which to place the packaged object, once detected the robot must approach said container and deposit the packaged object inside of the container.

Once the task is complete the robot will start over and continue the search for remaining objects. In order to preserve battery, the robot might possibly abide by a predefined timer, so that the behavior cycle won't continue indefinitely.

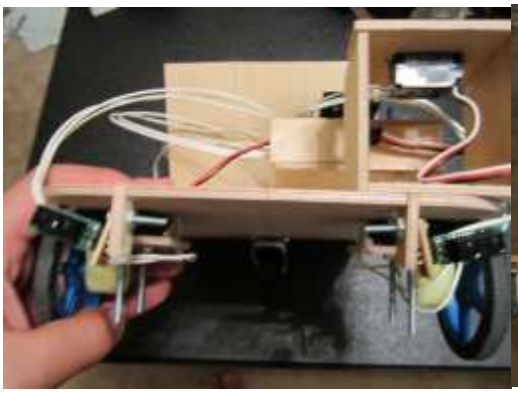
4.6 EXPERIMENTAL LAYOUT AND RESULTS

The platform has been built using the t-tech machine available at the IMDL lab, and half of the of electronic systems have been mounted, the image shows the designed parts and the cut out and assembled platform

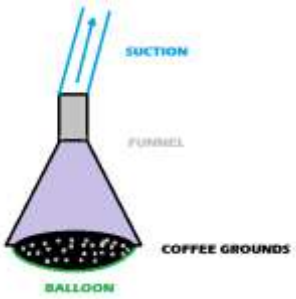


The platform consists mainly of:

- Wooden Base
- Camera mounting support
- Gripper arm and gripping mechanism (see below)
- Two geared DC motors and a Caster wheel
- Package system (to be implemented)



Certain experiments have been carried out with the gripper, and have yield positive results, it has been observed that for the gripper to function appropriately, the gripper must approach and closely attach to the object in order for the seal to form properly.



The size of the balloon matters as well, the granular material needs a certain amount of a space in order to re arrange itself during the conformation process. If the balloon is filled with an excessive amount (in practical terms over half of the balloon's volume) of coffee grounds, the gripper system won't work mainly because it will not conform properly.

MATLAB EXPERIMENTAL RESULTS FOR THE IP CAMERA USING A BASIC COLOR DETECTION ALGORITHM (COLOR DETECTION GREEN)

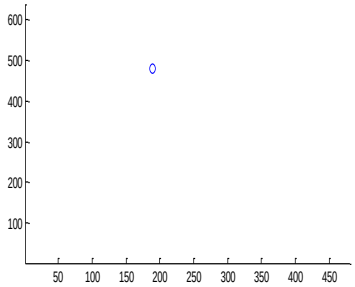
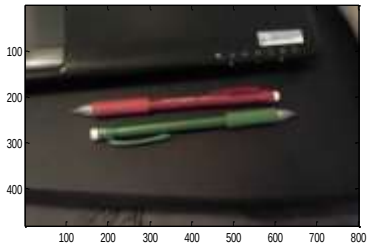
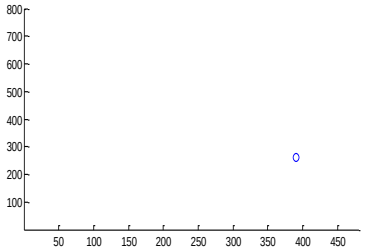
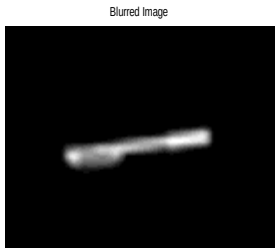


IMAGE: UPPER LEFT = SNAPSHOT, UPPER RIGHT = COLOR THRESHOLD, LOWER LEFT = FILTERED IMAGE, LOWER RIGHT = CENTROID PLOT







THE MATLAB CODE ALGORITHM OVERVIEW:

- TAKE A SNAPSHOT IMAGE
- PERFORM RGB TO HSV COLORSPACE CONVERSION

- HUE LEVEL IMAGE THRESHOLD (COLOR)
- LOW PASS FILTER (ATTENUATE IMAGE NOISE)
- CALCULATE CENTROID
- SERIAL COMMUNICATION (MOTION COMMAND)

MATLAB CODE FOR COLOR DETECTION

```
clear all
clc

img =
imread('http://192.168.1.137:8080/shot.jpg')
;

[w,h,d] = size(img);
output = zeros(w,h);
output_hsv = zeros(w,h);

H = fspecial('disk',10);              %
blurring filter

for i = 1:20,

    img =
imread('http://192.168.1.137:8080/shot.jpg')
;

    hsv_image = rgb2hsv(img);

    for j = 1:w,
        for k = 1:h,
            if (hsv_image(j,k,1)*255 > 64 &&
hsv_image(j,k,1)*255 < 101 ...
                && hsv_image(j,k,2)*255 > 50
&& hsv_image(j,k,3)*255 > 50 ...
                && hsv_image(j,k,3)*255 <
170),

                output_hsv(j,k) = 1;
            else
                output_hsv(j,k) = 0;
            end
        end
    end

   %output_hsv =
imresize(imresize(output_hsv,1/8),8);

   blurred =
imfilter(output_hsv,H,'replicate');
   blurred =
imresize(imresize(blurred,1/16),16);
   STATS = regionprops(ceil(output_hsv),
'Centroid');

   figure(1)
   subplot(221);imshow(output_hsv)
   subplot(222);image(img)

subplot(223);Scatter(STATS.Centroid(1),STATS
.Centroid(2));axis([1 w 1 h])
   subplot(224);imshow(blurred);
title('Blurred Image');

   pause(0.1);

end

close
```

OBSTACLE AVOIDANCE

THE STANDARD OBSTACLE AVOIDANCE RELIES ON A SMALL PROGRAM WRITTEN IN C USING AVR STUDIO 4 AND THE FUNCTION LIBRARIES AND SAMPLE CODE PROVIDED BY POLOLU, FOR CONTROL OF THE ORANGUTAN ROBOT CONTROLLER BOARD.

BASICALLY THE CODE CONTINUOUSLY POLLS THE FRONT IR SENSORS AND BASED ON THE INPUT STATE IT REACTS, BY BACKING UP A CERTAIN AMOUNT OF TIME AND ROTATES EITHER LEFT OR RIGHT DEPENDING OF THE ACTIVATED SENSOR.

THE CODE IS BASED ON ONE OF THE EXAMPLES PROVIDED BY POLOLU IN THEIR LIBRARIES.

```
#include <pololu/orangutan.h>

```

```
void goMotors(int motorSpeed, int motorSpeed2) {
    lcd_goto_xy(0, 1);
    print("spd=");
    print_long(motorSpeed);        // print the resulting motor speed
                                        (-255 - 255)
    print("   ");
    set_motors(motorSpeed, motorSpeed2);  // set speeds of motors 1
                                                and 2

    // all LEDs off
    red_led(0);
    green_led(0);
    // turn green LED on when motors are spinning forward
    if (motorSpeed > 0)
      green_led(1);
    // turn red LED on when motors are spinning in reverse
    if (motorSpeed < 0)
      red_led(1);
    delay_ms(100);
}

void haltMotors(void){
       set_motors(0,0);                  // halt speeds of motors 1 and 2
       delay_ms(100);
}

int main()                        // run over and over again
{
  while(1)
  {

    int motorSpeed;

       motorSpeed = 128;

             while(1) {

                    while(is_digital_input_high(IO_D3) &&
is_digital_input_high(IO_D2)){
                           goMotors(motorSpeed,motorSpeed);
                    }
                    haltMotors();

                    while(!is_digital_input_high(IO_D3) &&
!is_digital_input_high(IO_D2)){
                           goMotors(-motorSpeed,-motorSpeed);
                           delay_ms(1000);
                    }
                    haltMotors();

                    if(!is_digital_input_high(IO_D3)){
```

```
                        goMotors(-motorSpeed,-motorSpeed);
                        delay_ms(1000);
                        goMotors(-motorSpeed,motorSpeed);
                        delay_ms(1000);
                    }
                    haltMotors();

                    if(!is_digital_input_high(IO_D2)){
                        goMotors(-motorSpeed,-motorSpeed);
                        delay_ms(1000);
                        goMotors(motorSpeed,-motorSpeed);
                        delay_ms(1000);
                    }
                    haltMotors();
                }
  }
}
```

Behaviors

These are the basic functions that comprise the code

- wait_for_matlab(); - wirelessly send a '?' character to Matlab, and await results for the image processing
- search_ai(); - once results of the image processing are received act accordingly to approach the object, this routine exercises basic motor control
- haltMotors(); - self explanatory
- align(); - once the object is reached align for better arm operation
- pick_and_package(); - moves the arm and applies suction to grip, waits a few seconds and brings the picked object and the arm towards the packaging section to be packaged

C Code for the Microcontroller

```
#include <pololu/orangutan.h>
#include <math.h>

// parallel y = 2200

#define y_level 2100
#define y_initial 1150
#define x_level 800
#define x_initial 800

#define motorLeftSlow 35
#define motorRightSlow 34
#define motorLeftFast 76
#define motorRightFast 73


int goMotors(int motorSpeed, int motorSpeed2) {
```

```
       int timerValue = 0;

//     lcd_goto_xy(0, 0);
       timerValue = get_ms() % 512;
//     print("timer =");
//  print_long(timerValue);        // print timer (random value)

//     lcd_goto_xy(0, 1);
//    print("spd=");
//    print_long(motorSpeed);        // print the resulting motor speed (-255 - 255)

       set_motors(motorSpeed, motorSpeed2);  // set speeds of motors 1 and 2

       return(timerValue);
}

void haltMotors(void){
       set_motors(0,0);  // halt speeds of motors 1 and 2
       delay_ms(50);
}

void obstacleAvoidance() {

    int motorSpeed, randomValue;

       motorSpeed = 128;

                     while(is_digital_input_high(IO_D3) && is_digital_input_high(IO_C0)
&& is_digital_input_high(IO_D1)){
                            goMotors(motorSpeed,motorSpeed);
                     }
                     haltMotors();

                     while(!is_digital_input_high(IO_D3) &&
!is_digital_input_high(IO_C0)){
                            goMotors(-motorSpeed,-motorSpeed);
                            delay_ms(1000);
                     }
                     haltMotors();

                     if(!is_digital_input_high(IO_D1)){
                            randomValue = goMotors(-motorSpeed,-motorSpeed);
                            delay_ms(1000);
                     }
                     haltMotors();

                     if(!is_digital_input_high(IO_D3)){
                            randomValue = goMotors(-motorSpeed,-motorSpeed);
                            delay_ms(1000 + randomValue);
                            randomValue = goMotors(-motorSpeed,motorSpeed);
                            delay_ms(1000 + randomValue);
                     }
                     haltMotors();

                     if(!is_digital_input_high(IO_C0)){
                            randomValue = goMotors(-motorSpeed,-motorSpeed);
                            delay_ms(1000 + randomValue);
                            randomValue = goMotors(motorSpeed,-motorSpeed);
                            delay_ms(1000 + randomValue);
                     }
                     haltMotors();

}
```

```
void stallCurrentSensing() {

	int currentValue1 = 0;
	int currentValue2 = 0;

	currentValue1 = analog_read(0);
	currentValue2 = analog_read(1);
	clear();
	print("Stall I1 =");
	print_long(currentValue1);
	lcd_goto_xy(0,1);
	print("Stall I2 =");
	print_long(currentValue2);
	delay_ms(250);

}

void remoteControl(){

}

void servo_init() {

	const unsigned char demuxPins[] = {IO_B3, IO_B4};	// four servos, B3=SA,
B4=SB

	// Start servos, set speeds and initial positon values
	servos_start(demuxPins, sizeof(demuxPins));

	clear();
	print("Servo Zero");

//	set_servo_speed(0, 150);
//	set_servo_speed(1, 150);
//	set_servo_speed(2, 200);
//	set_servo_speed(3, 200);

	delay_ms(500);			// Wait

	// Make all the servos go to a neutral position.
	set_servo_target(0, y_initial);  // Tilt
	delay_ms(2000);			// Wait
	set_servo_target(1, x_initial);  // Pan
	delay_ms(2000);			// Wait
//	set_servo_target(3, 1500); // Clamp
//	delay_ms(1000);			// Wait*/
//	set_servo_target(2, 2300); // Swing
//	delay_ms(1000);			// Wait

	clear();

}


void pump_control(char enable){

	if(enable == 1) {
		set_digital_output(IO_C1,LOW);
		clear();
		print("LOW");
	}
	if(enable == 0) {
```

```
              set_digital_output(IO_C1,HIGH);
              clear();
              print("HIGH");

       }

}

void blink_led() {

       green_led(HIGH);
       delay_ms(75);
       green_led(LOW);
       delay_ms(75);

}

void search_ai(){

       char received;
       print("Searching...");

       while(is_digital_input_high(IO_D2)){

              serial_send(UART0,"?",1);

              serial_receive_blocking(UART0,&received,1,1000);


                     switch (received){
                            case 'w':
                                   goMotors(motorLeftSlow,motorRightSlow);
       // forward
                                   break;
                            case 's':                                   // backward
                                   goMotors(-100,-90);
                                   break;
                            case 'd':                                   // hard turn
                                   goMotors(100,-90);
                                   break;
                            case 'a':
                                   goMotors(-100,90);
                                   break;
                            case 'z':                                   // soft turn
                                   goMotors(-motorLeftSlow,motorRightSlow);
                                   delay_ms(150);
                                   haltMotors();
                                   break;
                            case 'c':
                                   goMotors(motorLeftSlow,-motorRightSlow);
                                   delay_ms(150);
                                   haltMotors();
                                   break;
                            case 'e':
                                   goMotors(motorLeftSlow,-motorRightSlow);
                                   break;
                            case 'x':                                   // halt
                                   haltMotors();
                                   break;
                            default:
                                   blink_led();
                                   break;
                     }
```

```
	}
}

void arm_forward() {

	delay_ms(1500);
	set_servo_target(0,y_initial);	// y
	delay_ms(2500);
	set_servo_target(1,x_level);	// x
	delay_ms(2500);
	set_servo_target(0,y_level);	// y
	delay_ms(4000);
}

void arm_backward() {

	delay_ms(1500);
	set_servo_target(0,y_initial);	// y
	delay_ms(2500);
	set_servo_target(1,2450);  // x
	delay_ms(4000);
	set_servo_target(0,1400);  // y
	delay_ms(2500);

}

void swing() {

	delay_ms(1500);
	set_servo_target(2,2300);  // y
	delay_ms(1500);
	set_servo_target(2,600);   // x
	delay_ms(1500);
}

void clamp(char enable) {

	delay_ms(1500);
	switch (enable)
	{
	case 0:
		set_servo_target(3,1600);  // y
		delay_ms(500);
		break;
	case 1:
		set_servo_target(3,1100);  // x
		delay_ms(500);
		break;
	default:
		clear();
		print("CLAMP ERROR");
		wait_for_button_press(ANY_BUTTON);
		break;
	}
}


void bat_test()
{

	unsigned int batt, bat[3];

	bat[0] = read_battery_millivolts_svp();
```

```
	delay_ms(100);
	bat[1] = read_battery_millivolts_svp();
	delay_ms(100);
	bat[2] = read_battery_millivolts_svp();
	batt = (bat[0] + bat[1] + bat[2])/3;

	print("Battery Voltage");
	lcd_goto_xy(0,1);
	print_long(batt);
	print("mV");

}

void setup() {

	lcd_init_printf();

	serial_set_baud_rate(UART0,9600);

	servo_init();

	set_digital_output(IO_C1,LOW); // HOLD PUMP OFF
	set_digital_input(IO_C0,HIGH_IMPEDANCE);

	clear();
	bat_test();
	delay_ms(2500);
	clear();

}

void pick_and_package(){

		clear();

		print("Arm Forward");

		arm_forward();			// postion arm

		clear();
		print("Pump ON");
		pump_control(0);	// grip

		clear();
		print("Wait");
		delay_ms(5000);			// wait for suction

		clear();
		print("Arm Backward");
		arm_backward();			// bring in

		clear();
		print("Open Clamp");
		clamp(0);				// open bag

		clear();
		print("Pump OFF");
		pump_control(1);	// release

		clear();
		print("Wait");
		delay_ms(2000);			// wait
```

```
              clear();
              print("Close Bag");
              clamp(1);                  // close bag

              clear();
              print("Swing Arm?");
              swing();

}

void wait_for_matlab(){

       char received;
       print("WaitingForMatlab");
       while(serial_receive_blocking(UART0,&received,1,1000)){
              blink_led();
              // do nothing
       }
       clear();
}

void align(){

//     char received;
       goMotors(-motorLeftFast,-motorRightFast);
       delay_ms(300);
       haltMotors();

}

//---------------------------------Main Program----------------------------//

int main()                     // run over and over again
{

       delay_ms(300);
       setup();

       print("Push Switch");
       lcd_goto_xy(0,1);
       print("to end ObsAvoid");


       while(1) {

              //delay_ms(50);

              clear();
              wait_for_matlab();
              search_ai();
              clear();
              haltMotors();

              align();
              pick_and_package();

              blink_led();
              clear();
              print("Done!");
              haltMotors();
              delay_ms(1500);
       }
}
```

5. CONCLUSION

The whole experience was a great engineering exercise, from conception to design and from building to testing, a lot of lessons learned and a lot of room to improve. A few important lessons learned:

- Solid mechanical platform and simple organized software - key to success
- Keep it simple!
- Divide and conquer (break things down in small problems)!
- Listen for input from others, constructive criticism is very helpful to find solution to seemingly "unsolvable" issues.
- Test your mechanical systems thoroughly to avoid surprises on demo day!

A lot of the progress was made by taking into account suggestions from the TAs and Faculty, in order to improve the design and idea. Surely many details can be improved and ideas changed to better adapt the project to a useful prototype.

The design for the robot has been finished; there are some details that could have been ironed out, however the experience was more than worth it.

6. DOCUMENTATION

[1] Eric Brown, Nicholas Rodenberg, John Amend, Annan Mozeika, Erik Steltz, Mitchell R. Zakin

Hod Lipson, and Heinrich M. Jaeger, Universal robotic gripper based on the jamming of granular material.

Internet: http://home.uchicago.edu/~embrown/BRAMSZLJ10_PNAS.pdf, University of Chicago

Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M., Lipson, H., Jaeger, H. (2010) "Universal robotic gripper based on the jamming of granular material," **Proceedings of the National Academy of Sciences (PNAS)**, Vol. 107, no. 44, pp.18809-18814.

POLOLU WEBSITE – ORANGUTAN BOARD - http://www.pololu.com/catalog/category/86

SPARKFUN WEBSITE – Sensor and actuator Images - http://www.sparkfun.com

MOTOROLA WEBSITE - http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/MOTOROLA-TRIUMPH-US-EN

7. APPENDICES

Robot's Website - https://sites.google.com/site/robotswebsite/