DriMix 4ics

Drink Mixer for Alcoholics

Robert Hartwell

University of Florida
Department of Electrical and Computer Engineering
EEL 5666C – IMDL – Initial Report
Instructors: A. Antonio Arroyo, Eric M. Schwartz
TAs: Joshua Weaver, Ryan Stevens, Tim Martin

Table of Contents

Table of Figures

Abstract

It is often hard to remember even a fraction of the combination of ingredients in the thousands of drink combinations that exist. Not to mention that after you've had a couple of drinks, the only thing you'll remember is to drink the drink, much less mix it. Why not have someone or something do all the work for you? The DriMix 4ic will take a drink order, mix it, then bring it right to your hands.

Executive Summary

The DriMix 4ics uses two types of special sensors to control an autonomous robot. Using a cell phone and an android app that can read QR code, drinks can be chosen from a menu. A webcam is used to identify colors on “control cards,” these cards allow for general controls of the robot. The cards allow for placing an order, changing the mode of operations, or issuing an emergency stop command.

Once the DriMix 4ics receives both an order (QR code) and a command to take that order (green card), the robot will take note of its current position. It will then head to the drink mixing station, activate the correct servos to mix the order, then return the order to the place where it was originally received.

After the order is complete the robot goes back to its normal mode of operations. The normal mode of operations is determined by what mode it is in. There are two modes, a party mode and a solo mode. The party mode cycles up and down the bar, potentially bringing drinks to multiple different locations. In the solo mode, the robot will only bring drinks to one location.

Introduction

The goal of this robot was to combine two things, creating an autonomous robot (the objective of this class) and making a drink mixing robot (something I had wanted to do anyways). The class requirements aspect came into play by making the project mobile.

The main objective of the project was to create a mobile robot that could goto a drink station, mix some drinks, then return to a user with the mixed drink.

This paper will cover: the overall system interaction of the DriMix 4ics, the movement and actuation of the robot, the sensors and their testing conditions, the power distribution system, the behavior of the robot, the experimental changes, and the lessons learned.

Systems of the DriMix 4ics

Integrated System

The DriMix 4ics uses the Epiphany DIY board. This board is based on the ATXMega64A1 Microcontroller. The platform consists of several sensors, servos, DC motors, wireless communication, batteries, and a drink mixing station. The block diagram shown in Figure-1 describes the basic interactions of the platform.

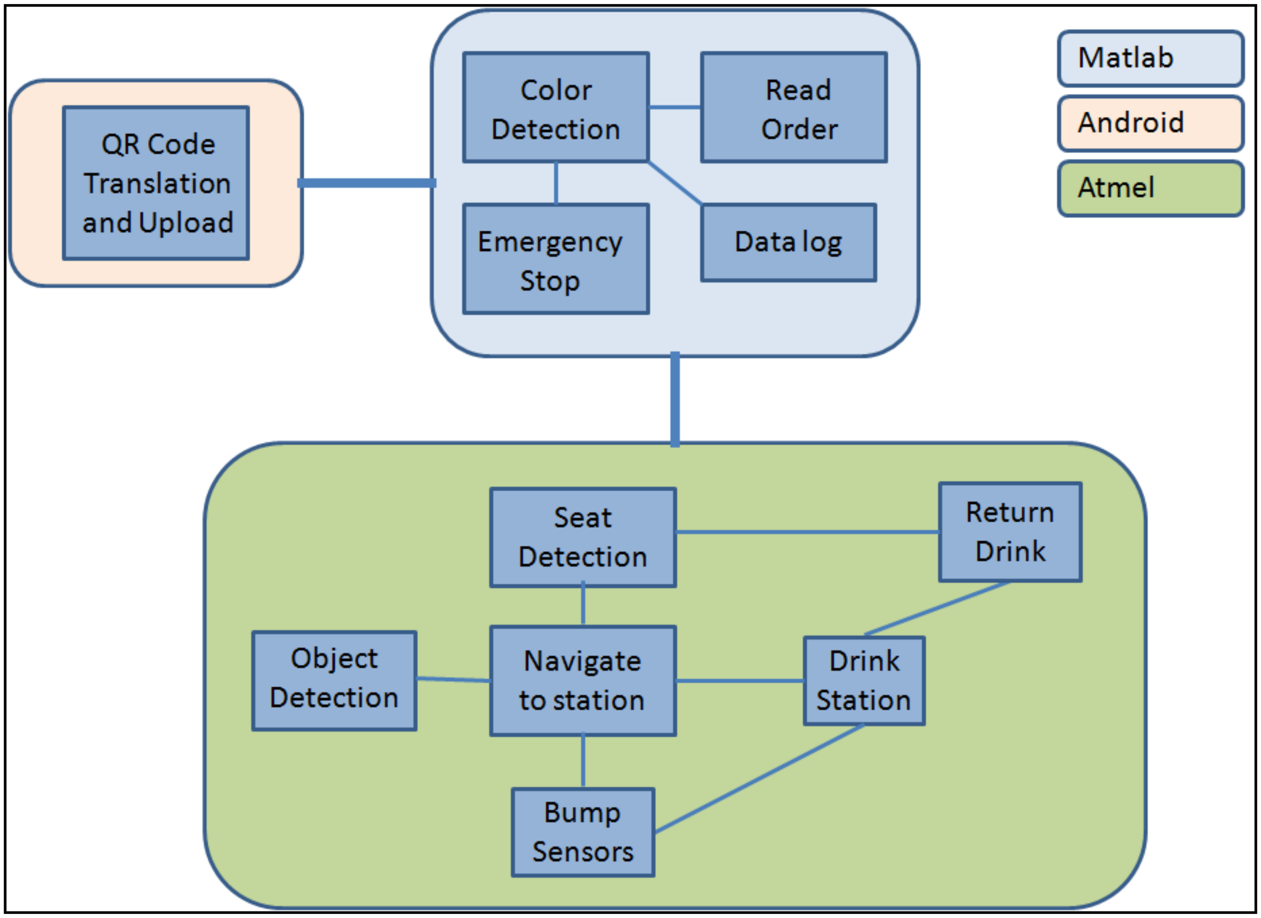


Figure 1: DriMix 4ics System Block Diagram

Mobile Platform – The Robot

The mobile platform is composed of several layers. Each layer hosts a different array of sensors, motors, or actuators. The top two layers holds the adjustable servo array. The layer beneath that hosts the IR rangers. Attached to this layer is the channel for the sliding drink return cart. The layer under this holds the microcontroller and the bump switches. The very bottom of the robot hosts the battery packs, switches, DC motors, and the IR array. The overall setup can be seen in Figure-2.

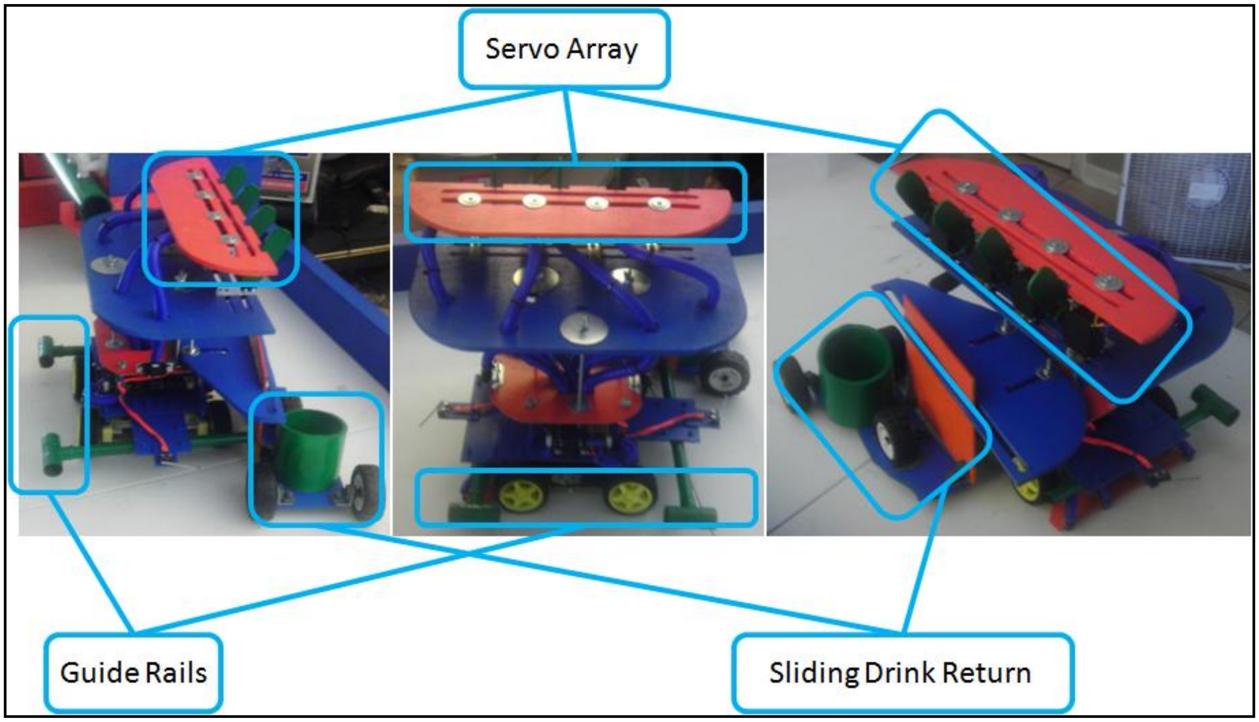


Figure 2: Mobile Platform Principle Components

Stationary Platform – The Drink Station

The stationary drink mixing station hosts plastic containers that hold various mixable drinks. On top of the levers of each of the spigots are lever guides that ensure the cams from the servo motors do not slip off while pressing down. The down spouts from the spigots are channeled into a larger collector that is angled and open at one end. This allows all of the liquid to be collected in the cup on the return cart. This set up can be seen in Figure-3.

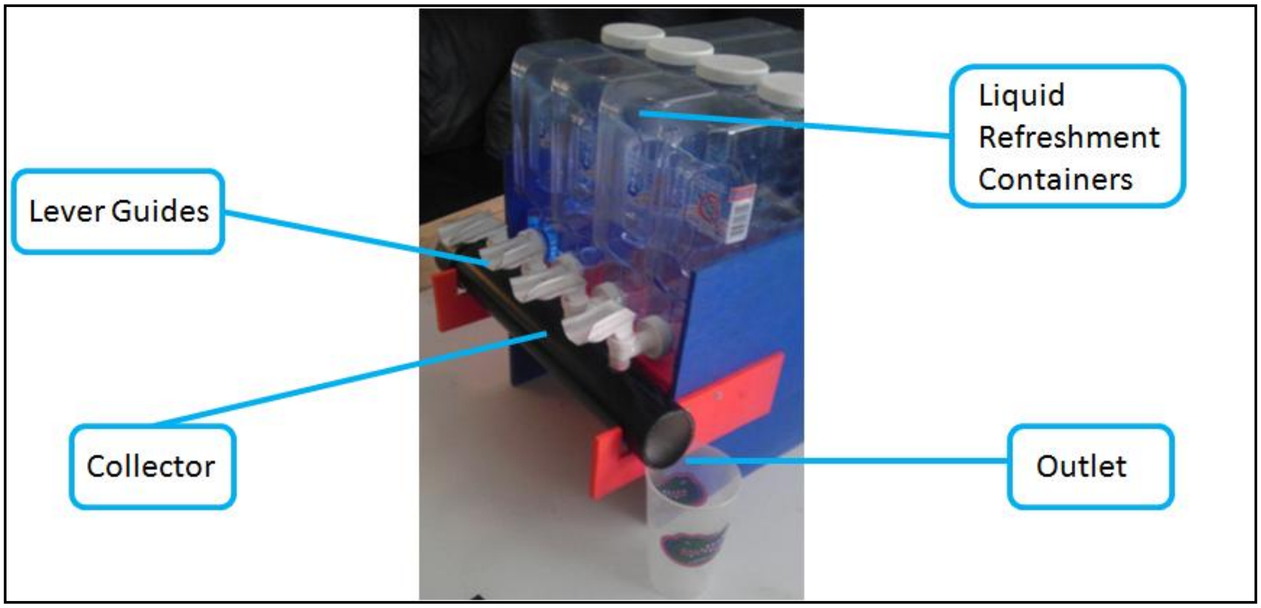
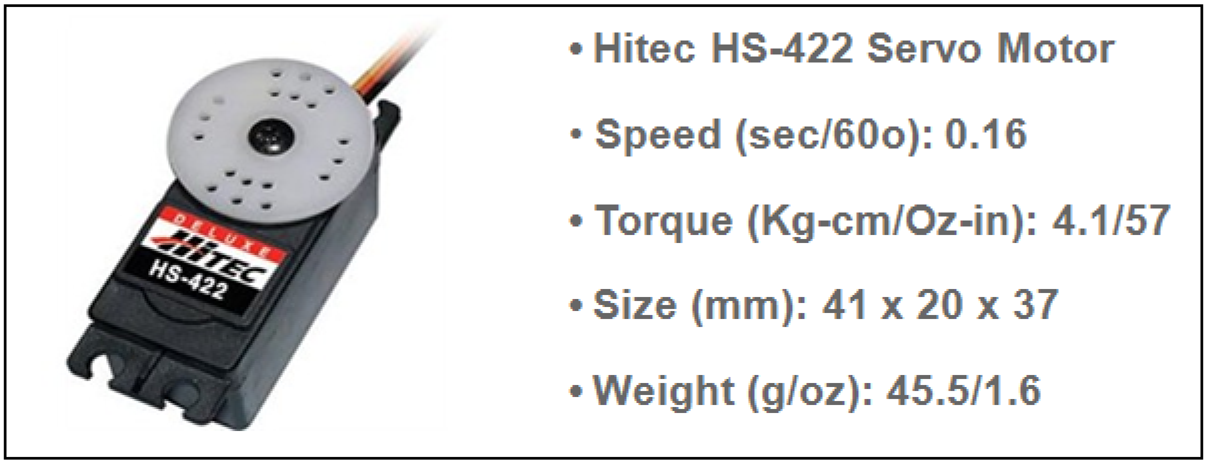


Figure 3: Mixing Station Principle Components

Movement, Actuation, and Drink Return

Actuation



- Hitec HS-422 Servo Motor
- Speed (sec/60o): 0.16
- Torque (Kg-cm/Oz-in): 4.1/57
- Size (mm): 41 x 20 x 37
- Weight (g/oz): 45.5/1.6

Figure 4: Hitec Servo Information

Four Hitec servos were used in an adjustable servo array. The operating conditions and specifications fo the servos can be seen in Figure-4. The servos had cam lobes attached. The cam lobes were profiled in such a way as to allow for easy lever actuation on the drink station. The arrangement of the servos as well as the cam lobes can be seen in Figure-5.

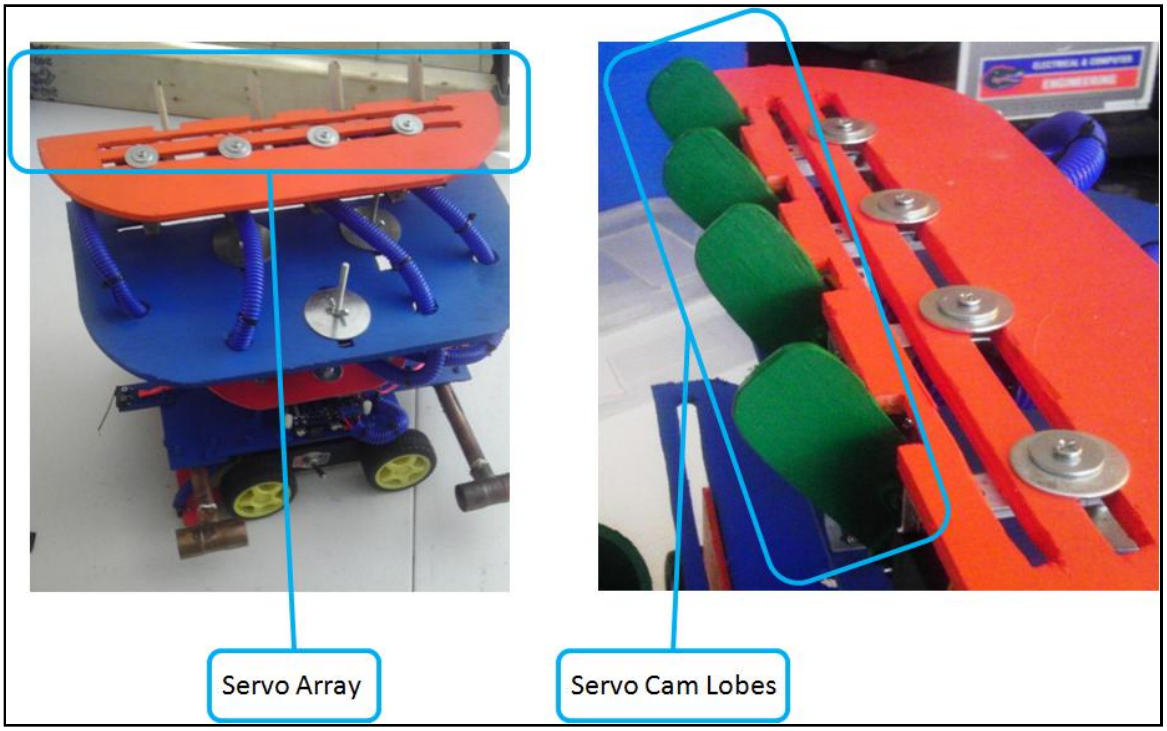
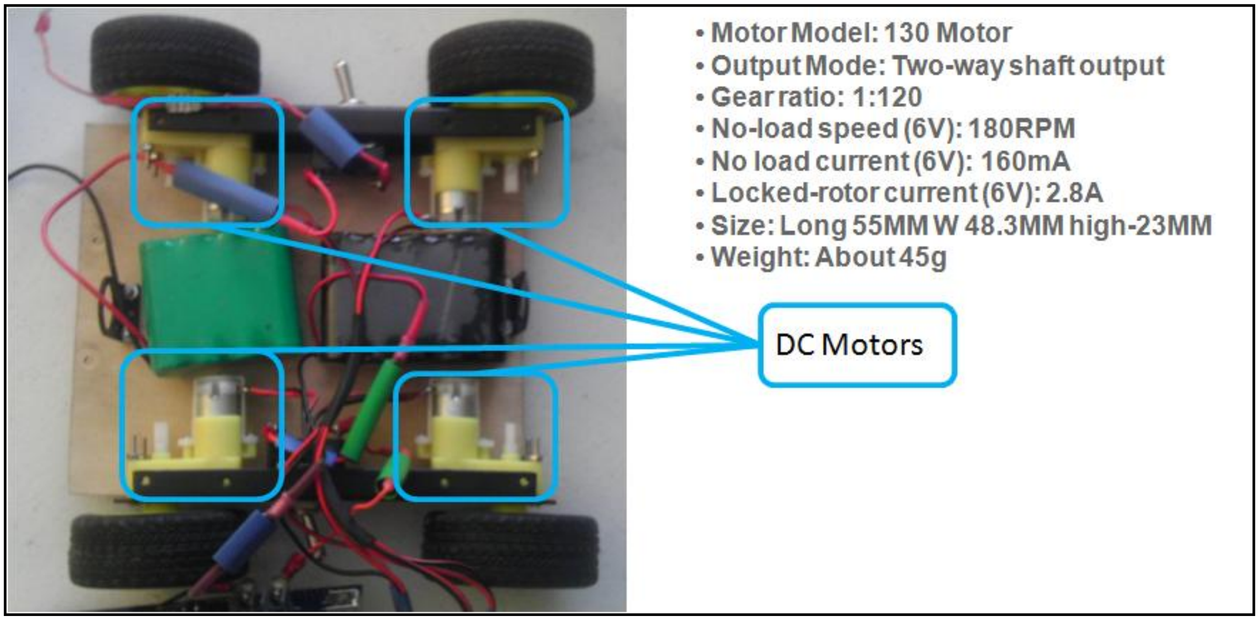


Figure 5: Servo Array and Servo Cam Lobes

Movement



- Motor Model: 130 Motor
- Output Mode: Two-way shaft output
- Gear ratio: 1:120
- No-load speed (6V): 180RPM
- No load current (6V): 160mA
- Locked-rotor current (6V): 2.8A
- Size: Long 55MM W 48.3MM high-23MM
- Weight: About 45g

Figure 6: DC Motors

The movement of the DriMix 4ics was controlled by 4 DC motors, movement was also guided by two guide bars that could slide over a larger guide bar running the length of the bar. The mounting of the DC motors can be seen in Figure-6. The 4 DC motors easily had the power to move the platform. However after completely constructed the plastic hubs of the motors began to show signs of fatigue. The overall platform apparently weighed a bit more than the rated shaft strength.

A future improvement could be to mount 4 ball casters at the corners of the chassis. This would greatly reduce the amount of strain on the 4 hubs.

Drink Return

Due to the size and locations of the drink mixing station, the ends of the bar, and the robot itself; the drink return car needed to be moveable to compensate for clearance issues at either end of the bar. This was achieved by creating an adjustable platform with a guide channel cut into it. The drink cart had a rigid guide wall that slide into the guide channel attached to the robot. This allowed for linear motion on 1 axis. This can be seen in Figure-7.

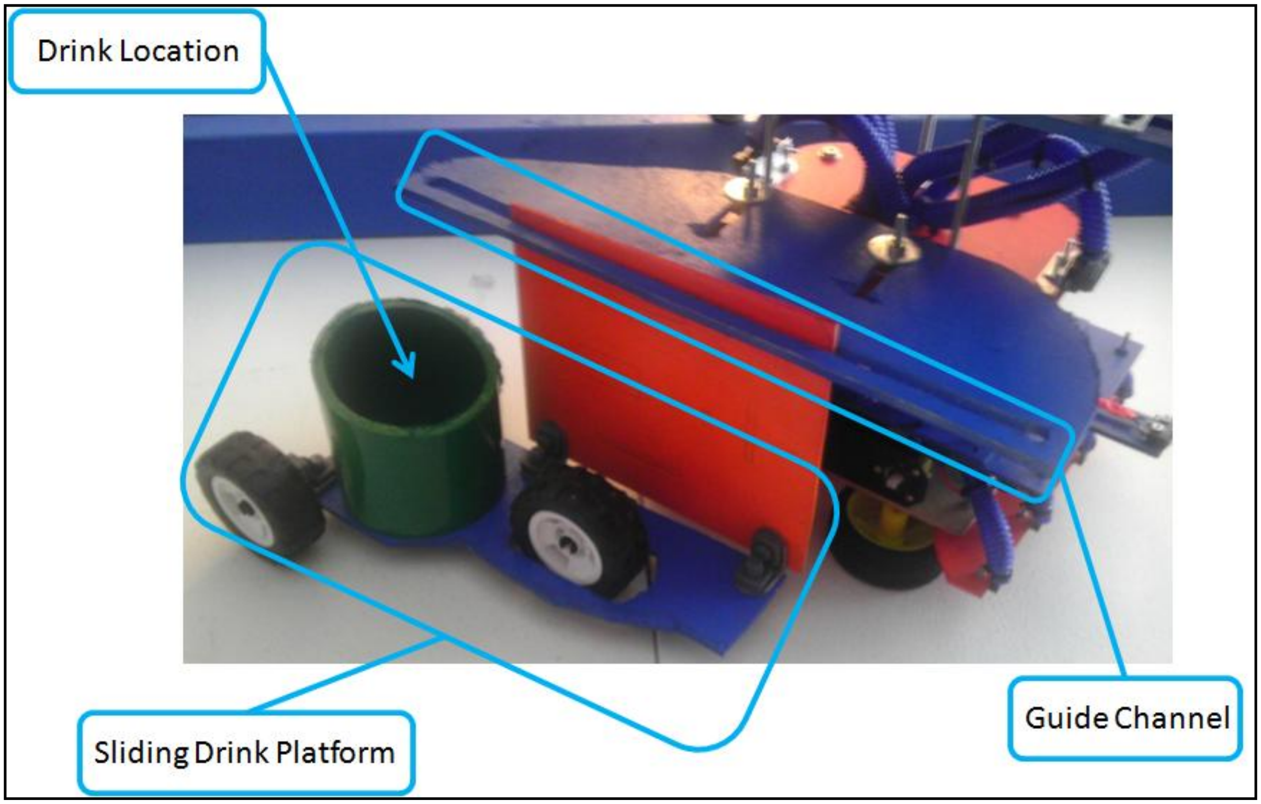
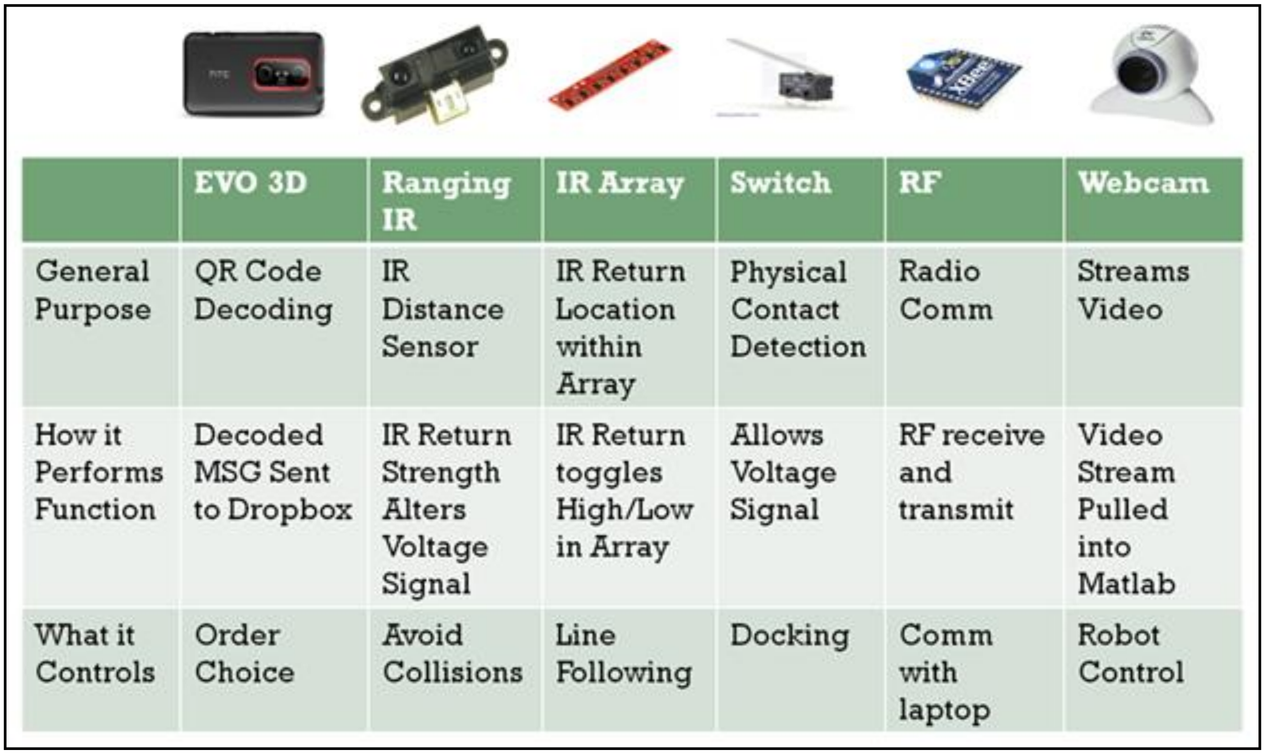


Figure 7: Drink Return Cart

Sensors

This section will outline the multiple sensors used on the DriMix 4ics. The section will also describe the testing conditions used to typify the sensors. The table found in Figure-8 describes the basic functionality of each of the sensors.

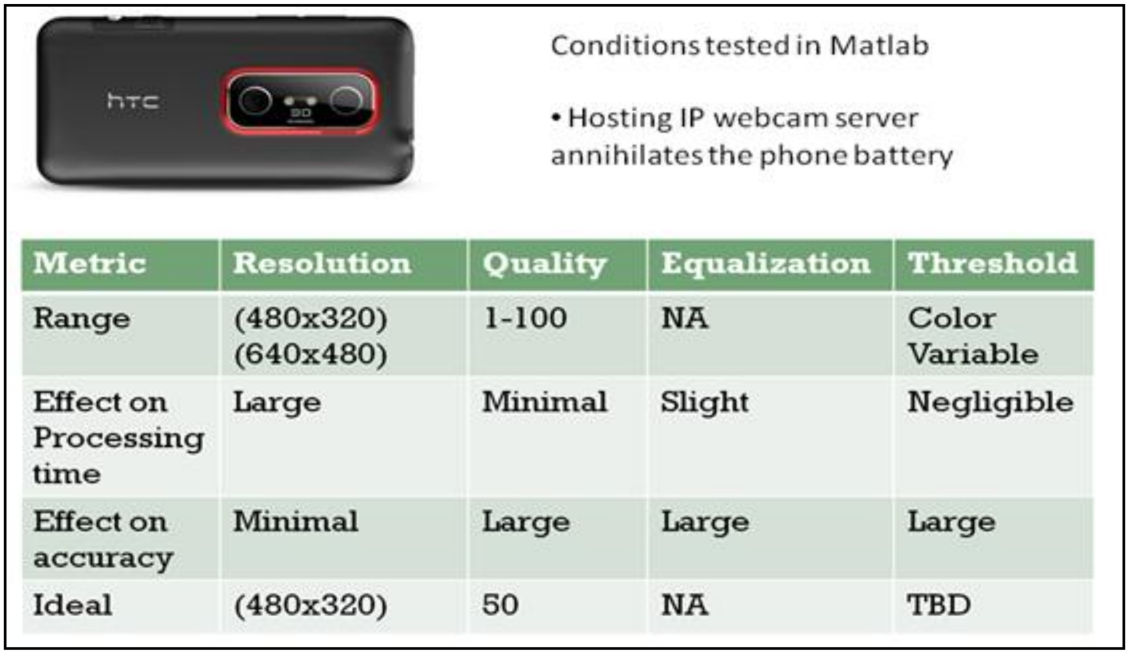


	EVO 3D	Ranging IR	IR Array	Switch	RF	Webcam
General Purpose	QR Code Decoding	IR Distance Sensor	IR Return Location within Array	Physical Contact Detection	Radio Comm	Streams Video
How it Performs Function	Decoded MSG Sent to Dropbox	IR Return Strength Alters Voltage Signal	IR Return toggles High/Low in Array	Allows Voltage Signal	RF receive and transmit	Video Stream Pulled into Matlab
What it Controls	Order Choice	Avoid Collisions	Line Following	Docking	Comm with laptop	Robot Control

Figure 8: Sensor Overview

Evo 3D, Android Phone

The EVO 3D runs an app available in the Android Marketplace called QR Droid. This allows the phone to act as a QR Decoder. After decoding the scanned menu QR graphic, the decoded message is sent to a Dropbox server as a .txt file. Using hooks in Matlab allows for the polling of the file in Dropbox. Then based on the content of the file, Matlab will process the order. Some of the initial conditions that the EVO was tested under can be seen in Figure-9.



Conditions tested in Matlab

- Hosting IP webcam server annihilates the phone battery

Metric	Resolution	Quality	Equalization	Threshold
Range	(480x320) (640x480)	1-100	NA	Color Variable
Effect on Processing time	Large	Minimal	Slight	Negligible
Effect on accuracy	Minimal	Large	Large	Large
Ideal	(480x320)	50	NA	TBD

Figure 9: EVO 3D Testing Characteristics

Gigaware Webcam

The Gigaware 2.0MP webcam captures video at a maximum resolution of 2-megapixels (1600x1200).

Figure 10: Gigaware Webcam

The webcam was run under a variety of conditions. The main emphasis was to attain quick processing speed while maintaining accuracy within Matlab. These conditions can actually be seen under the EVO 3D sensor section, as the EVO was initially going to be used as an IP Webcam. However those same testing conditions remained identical for the webcam. The webcam used can be seen in Figure-10.

Matlab was used to pull the webcam's video information. The video was split into its RGB components, then multiple filters and morphological operations were run on each of the layers. Even with multiple calculation per layer, the output was in near real time at almost 30 frames per second. The output of a three color image can be seen in Figure-11.

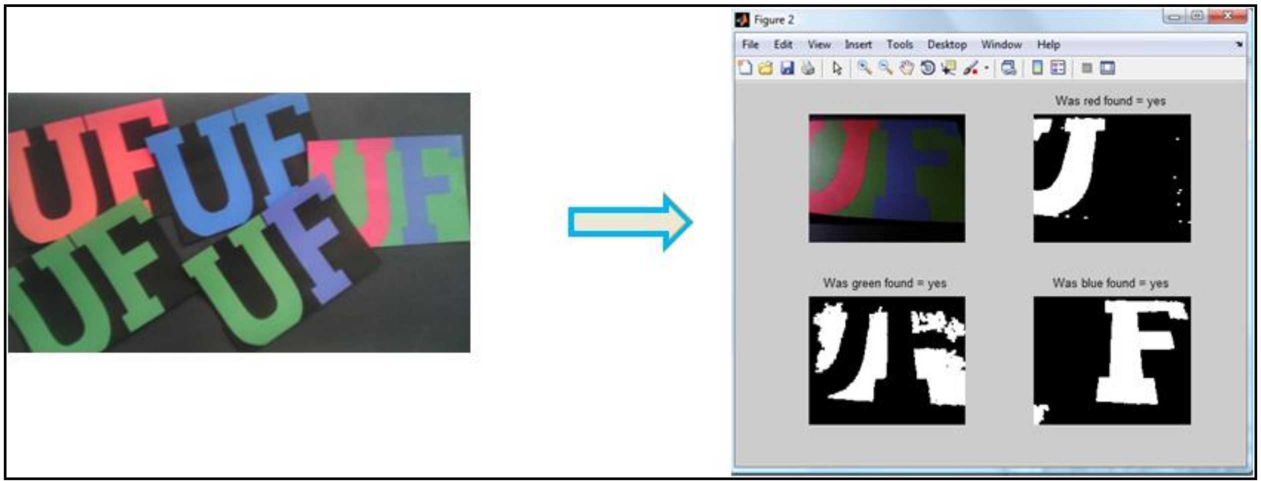
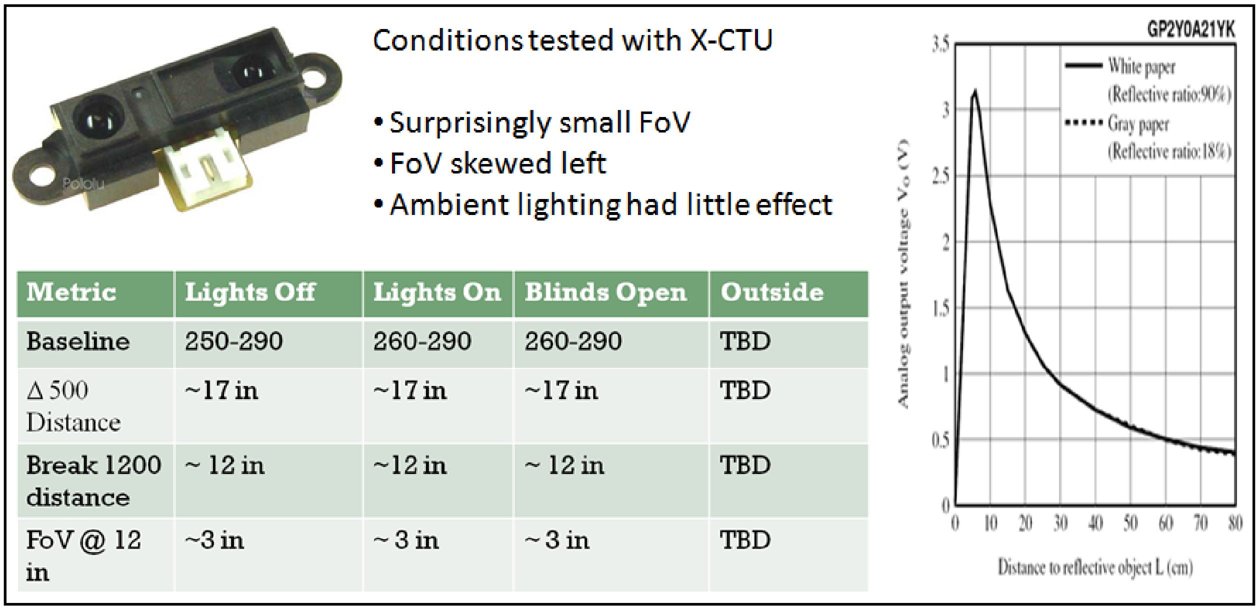


Figure 11: Gigaware Testing Characteristics in Matlab

Sharp Analog Distance Sensor 10-80cm

The Sharp IR range finder will be used for object avoidance within the navigation algorithm. The testing conditions of the IR Ranger were focused on finding an acceptable range at which an object could be detected under most lighting conditions yet still be a safe distance from the mobile platform. The testing conditions of the IR Range finders can be seen in Figure-12.



Conditions tested with X-CTU

- Surprisingly small FoV
- FoV skewed left
- Ambient lighting had little effect

Metric	Lights Off	Lights On	Blinds Open	Outside
Baseline	250-290	260-290	260-290	TBD
Δ 500 Distance	~17 in	~17 in	~17 in	TBD
Break 1200 distance	~ 12 in	~12 in	~ 12 in	TBD
FoV @ 12 in	~3 in	~ 3 in	~ 3 in	TBD

Figure 12: IR Ranger Testing Characteristics

The IR range finders can be seen on either end of the robot platform in Figure-13. They are set to activate over a set threshold value which can be altered according to ambient conditions or wanted activation range. Typically this range was set to be about a foot.

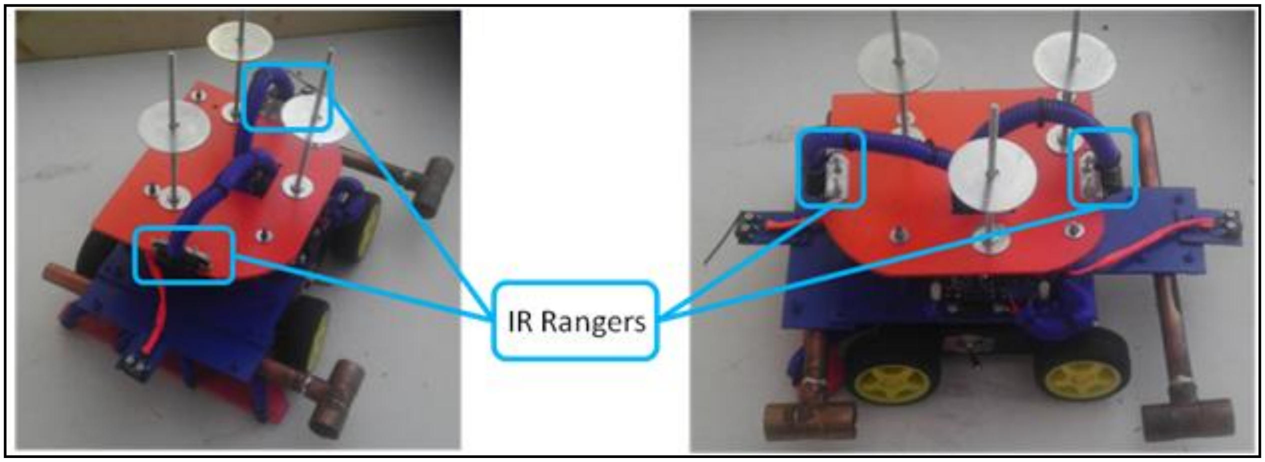
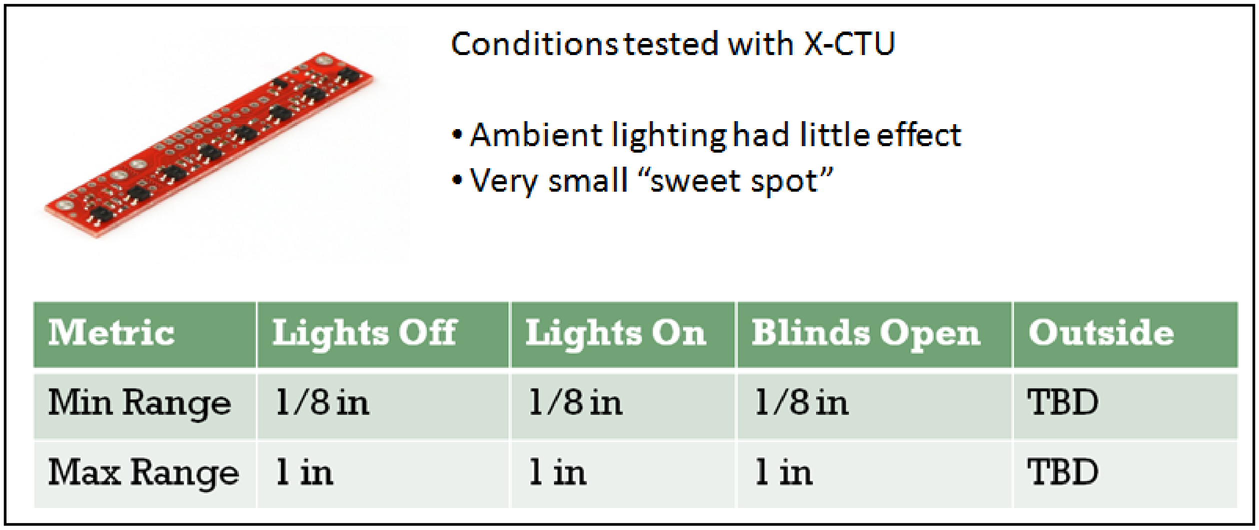


Figure 13: IR Ranger Mounting Locations

QTR-8A Reflectance Sensor Array

The IR reflectance Sensor array will be used for line following. Line following will be used for basic navigation. Testing for the IR Array was much like the testing conditions of the IR Ranger. The focus was to find the ideal range for accurate sensor measurement. The lighting conditions did not have a particularly large effect on the sensors. However there is a very small range for which the sensors are reliable, less than an inch of workable area. The table showing the testing conditions of the IR transmit and receive pairs can be seen in Figure-14.



Conditions tested with X-CTU

- Ambient lighting had little effect
- Very small "sweet spot"

Metric	Lights Off	Lights On	Blinds Open	Outside
Min Range	1/8 in	1/8 in	1/8 in	TBD
Max Range	1 in	1 in	1 in	TBD

Figure 14: IR Array Testing Conditions

A custom array was constructed using individual IR transmit/receive pairs. The individual pairs had the same characteristics as the QTR-8A array, so the above testing data applies to the actual implementation as well. The custom array can be seen in Figure-15.

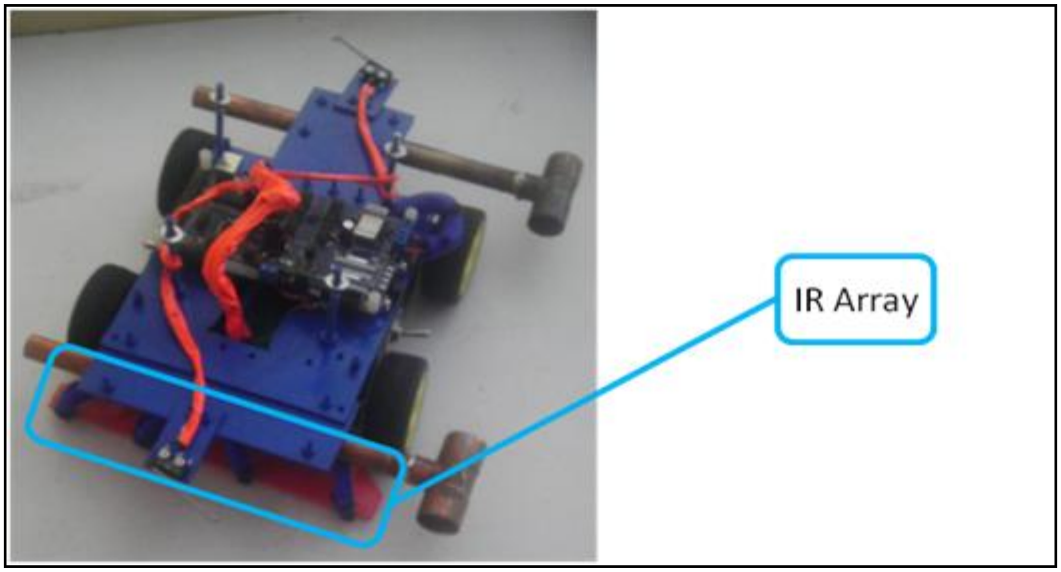


Figure 15: IR Array Mounting Location

Snap-Action Switch with 50mm Lever: 3-Pin, SPDT, 5A

Two snap switches were used for aiding in docking the platform as well as the routine for checking back and forth along the bar for the bar ends. Testing the switch was a simple matter of checking continuity. Information about the switch can be seen in Figure-16.

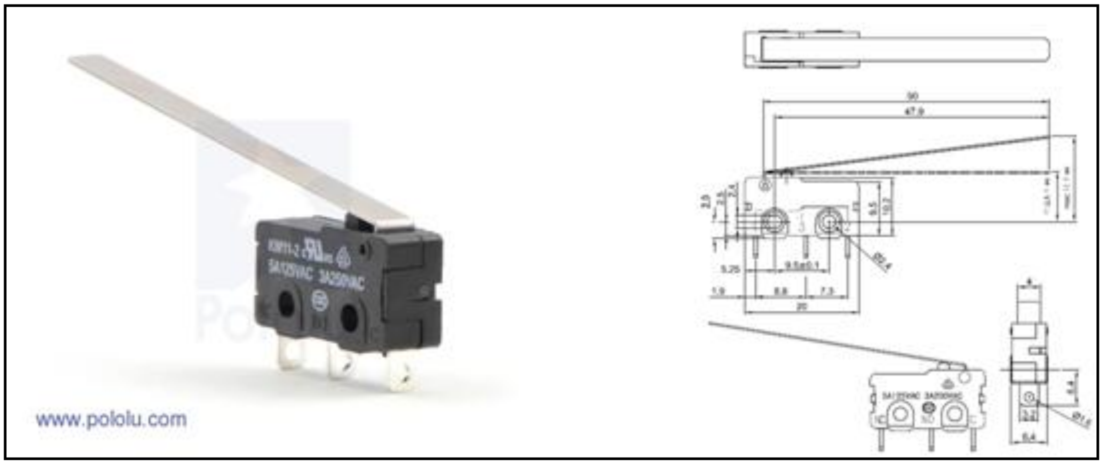


Figure 16: Snap Switches

The switches were added to the microprocessor by toggling some input pins to be Pull-Down resistors. That way when activated the voltage would be read as high (5v). They can be seen labeled in Figure-17.

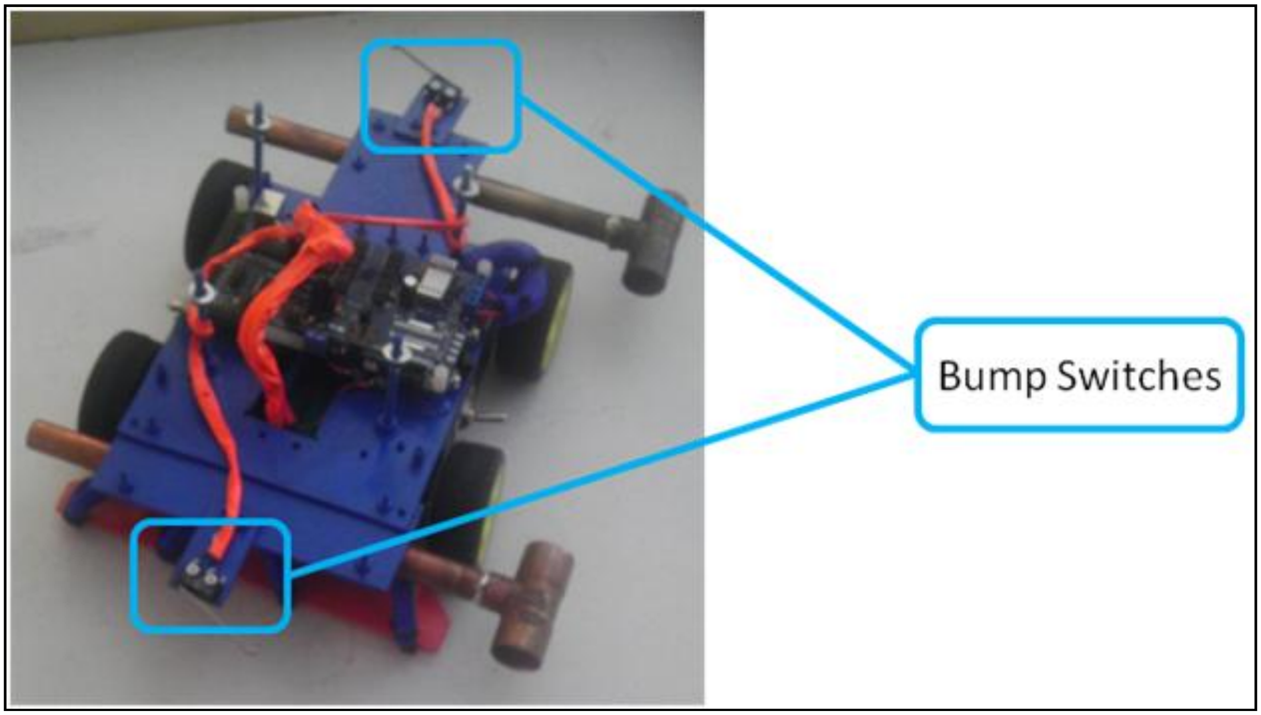


Figure 17: Bump Switch Mount Locations

Power Distribution

Power layout

Power to the entire platform could have easily been supplied by a single battery pack, however two battery packs allowed for very long term testing and tuning without having to recharge. One battery pack supplied power to the Microcontroller main board while another supplied power to the motors. Each was independently controlled by a switch. This allowed for easily turning off the motors when movement was not needed. The layout can be seen in Figure-18.

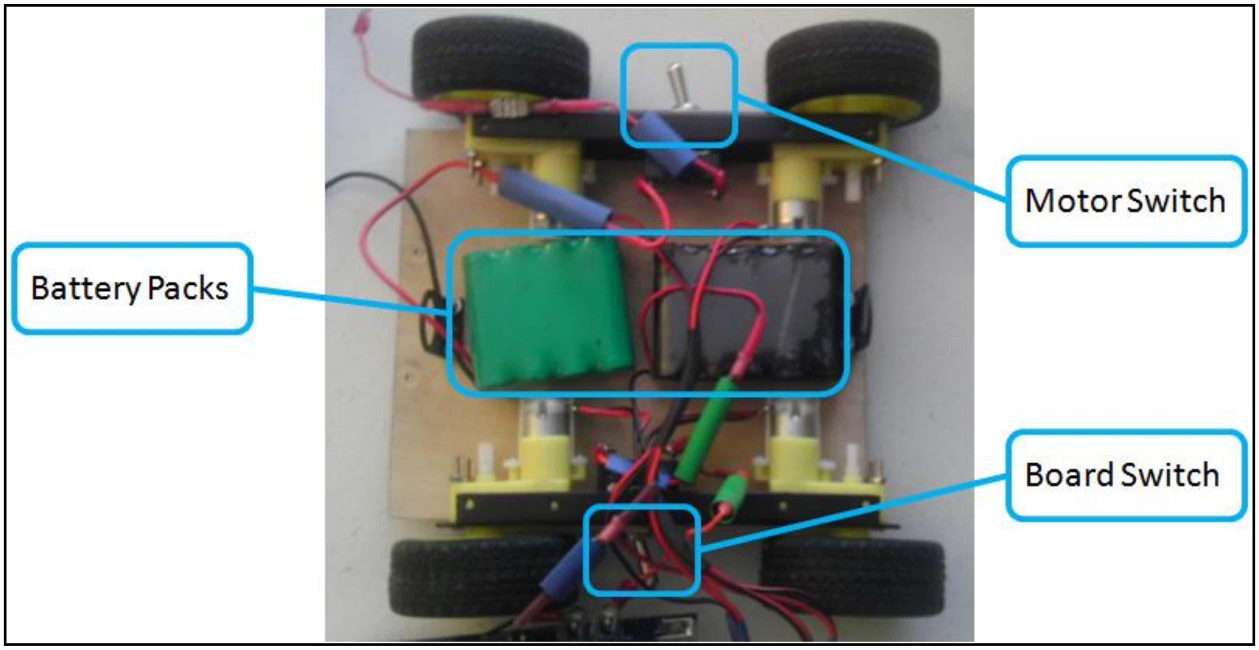


Figure 18: Power Distribution

Behaviors

Navigation

The DriMix 4ics essentially cycles back and forth on the bar top. Different seating locations at the bar are marked by black bars of different lengths. This is how the robot knows where to take a drink.

Docking

Based on the direction of travel and activation of bump sensors, the robot knows which end of the bar it is located. Prior to activation the adjustable servo array is properly aligned with the mixing station. This way when the robot reaches the end of the bar with a drink order, activation of the drink levers will be aligned with the drinks

Mixing

After the process involved in decoding and transmitting the order, and the robot is docked next to the mixing station, the proper servos will be activated. The servos correspond to the drinks in the mixing station. The correct servos activate and the fluids pour into the collector then into the cup.

Object avoidance

There are essentially three forms of object avoidance on the DriMix 4ics. IR rangers, bump sensors, and the emergency stop triggered by the IP cam and “red” card. The IR rangers and red card cause a stop in motion, the bump switches will cause a reverse in motion.

Taking Order

After an order is scanned and sent to Dropbox, it is only after a green order card is scene that the values is scanned in. This is what happens on the matlab side. On the robot side unless the “waitfororder” value is 1 (essentially telling the robot that green has been found), any order value sent will be disregarded.

Experimental Layout and Results

Original sensor platform and layout

The platform used is the DFRobot 4WD Arduino Mobile Platform. The platform was chosen because it is light weight, inexpensive and has a 4 wheel drive system with 4 DC Motors. It was only during the course of the project that the platform was discovered to be inadequate for the required tasks. The DC Motors, wheels, and motor mounts were the only things that were kept from the platform. The base chassis was about 3 inches wider and 2 inches longer than the original DFRobot. Also it ended up being multi-tiered.

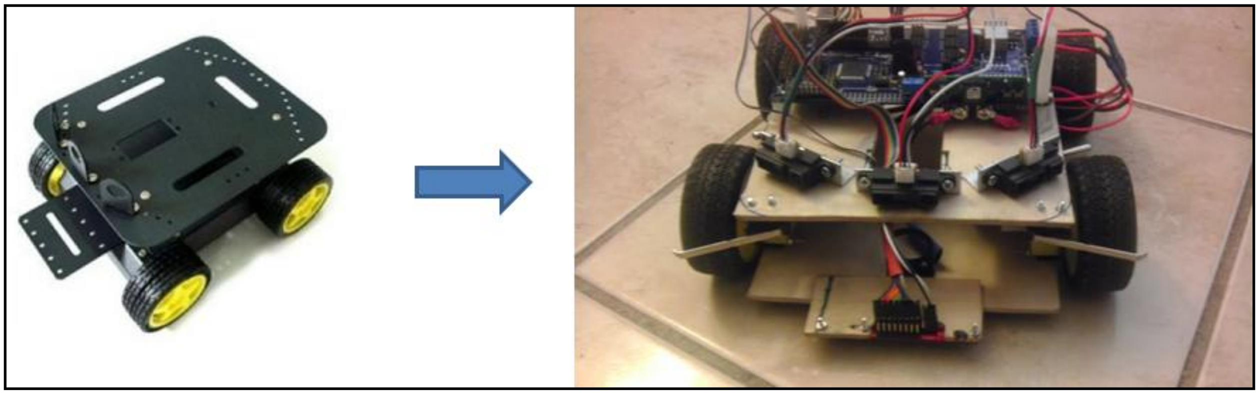


Figure 19: Platform Early Evolution

The original platform ended up have little in common with the finished product. Seen in Figure-19 is the original set up of the DriMix 4ics. In this configuration the DriMix 4ics was a much more mobile platform. The code at the time allowed the platform to “dodge and weave” as well as line follow. However, none of those functions were required on a bar top serving drinks.

Sensor experimentation

Most of the experimental data obtained from the sensors can be seen in the sensors section. This includes ranges tested and the lighting conditions that these measurements were made in.

However, included here is a screen shot showing the evolution of the RGB code. The RGB lighting conditions are critical for accurate detection and classification. Much better results would have been obtained by working in the HSV color domain as it is much more robust to changes in lighting. The code base evolved from simply locating an RGB centroid to separating the channels, multiple filtering steps, then checking required congruent pixel sizes to determine if thresholds were enough to be considered detection. The original RGB centroid finding output can be seen in Figure-20.

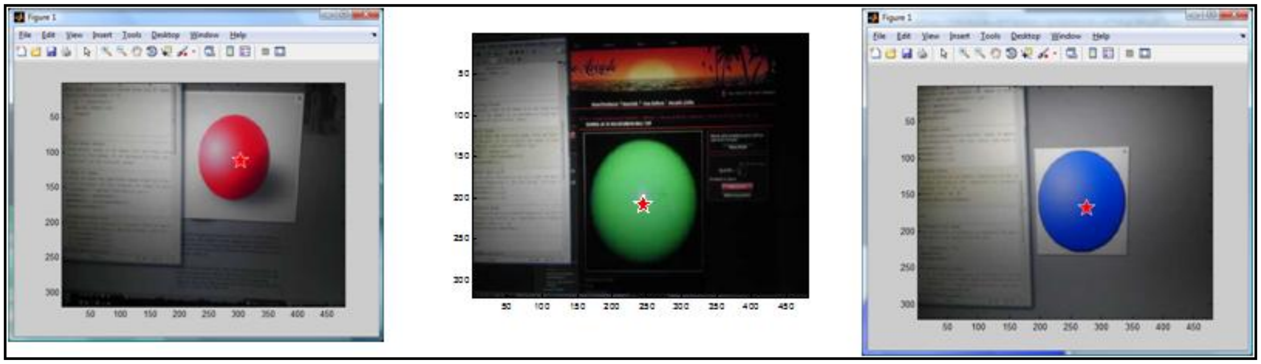


Figure 20: First Implementation of Color Tracking

CLOSING

Lessons Learned

There were multiple lessons learned of the course of building the DriMix 4ics. Some of them will be listed below in no particular order.

- Gainesville humidity will warp thin wood. Mount everything to either sturdy wood or metal.
- Order backups of all sensors. You don't have time to wait for a sensor in the mail.
- Work with HSV, not RGB
- Make sure that sensors and moving part mounts are adjustable. Easy tweaking
- is the key to rapidly fixing problems.

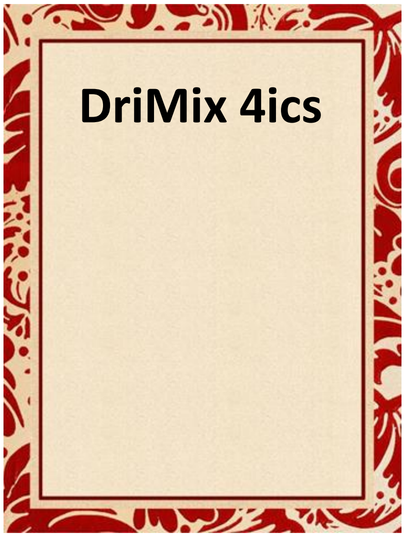
Enhancements

One of the best ways to improve this project for the future would be to create a sturdier base chassis (see comments above about thin wood) and mount motors with metal shafts. I am sure that the code could be improved as well, coding can always be improved. In particular migrate from RGB to HSV color space.

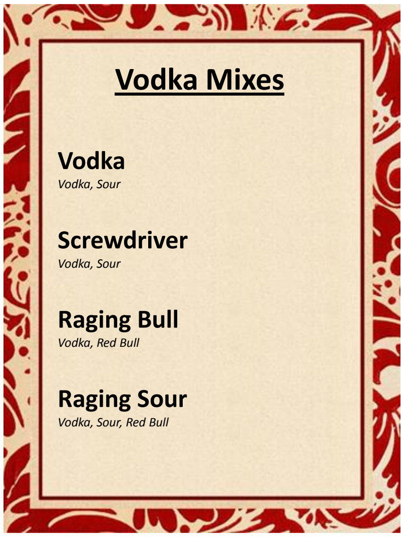
Appendices

Menu

The following shows the menu used and the QR code associated with the menu items.



DriMix 4ics



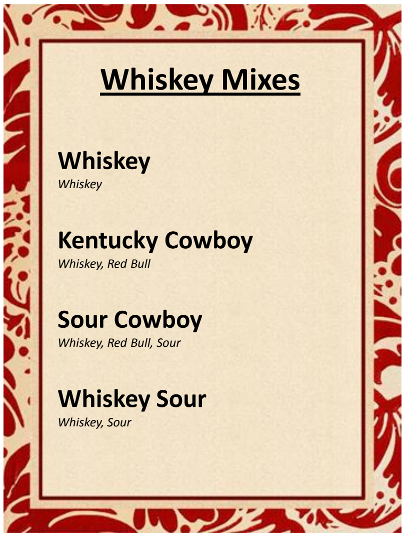
<u>Vodka Mixes</u>

Vodka
Vodka, Sour

Screwdriver
Vodka, Sour

Raging Bull
Vodka, Red Bull

Raging Sour
Vodka, Sour, Red Bull



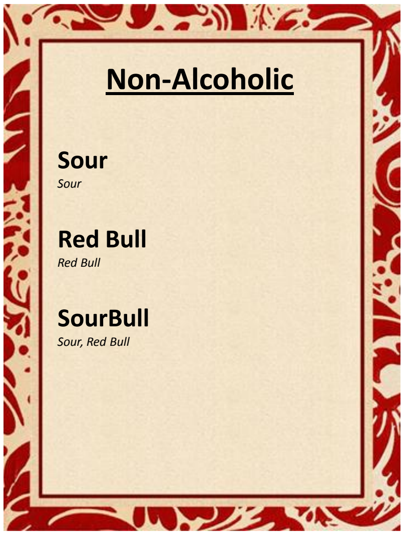
<u>Whiskey Mixes</u>

Whiskey
Whiskey

Kentucky Cowboy
Whiskey, Red Bull

Sour Cowboy
Whiskey, Red Bull, Sour

Whiskey Sour
Whiskey, Sour

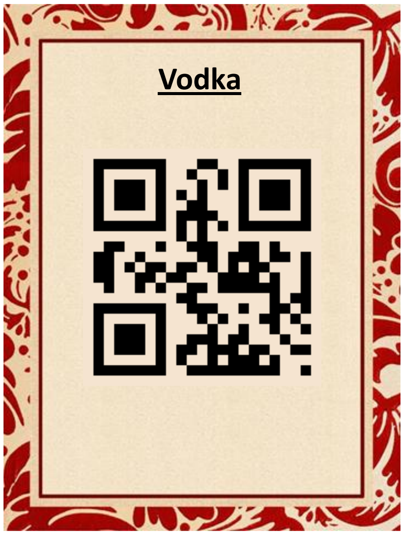


<u>Non-Alcoholic</u>

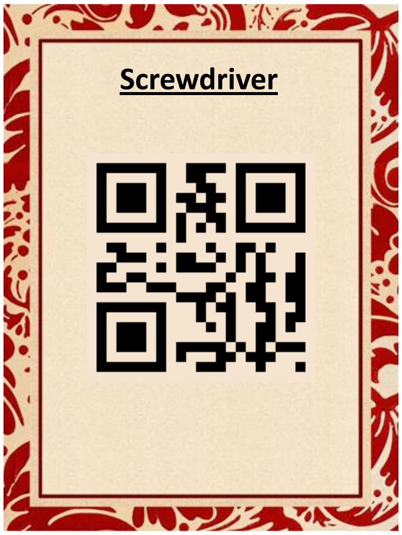
Sour
Sour

Red Bull
Red Bull

SourBull
Sour, Red Bull



<u>Vodka</u>



<u>Screwdriver</u>

Raging Bull

Raging Sour

Whiskey

Kentucky Cowboy

Sour Cowboy

Whiskey Sour

Sour

Red Bull

SourBull

Program Code

Matlab code

Get_colors function

```
function [ rd, gd, bd ] = get_colors( threshold, opening, detect, vid)

ball = getsnapshot(vid);
% ball = imread(url); % this line if using a ipwebcam



r = ball(:,:,1);
g = ball(:,:,2);
b = ball(:,:,3);



justblue = 2*b - g - r;
justgreen = 2*g - b - r;
justred = 2*r - b - g;

colors(:,:,:,1) = justred;
colors(:,:,:,2) = justgreen;
colors(:,:,:,3) = justblue;

% thresholding
bw = colors > threshold;

% remove small pixels
ball1 = bwareaopen(bw, opening);
% Dilate
se1 = strel('square',7);
ball2 = imdilate(ball1,se1);

cc = bwconncomp(ball2(:,:,:,1));
stats = regionprops(cc, 'Area');
% idxr = find([stats.Area] > 80);
% BWR = ismember(labelmatrix(cc), idxr);
arear = max(cat(1, stats.Area));
if arear > detect
    r_detected = 'yes';
    rd = 1;
else r_detected = 'no';
    rd = 0;
end

cc = bwconncomp(ball2(:,:,:,2));
stats = regionprops(cc, 'Area');
% idxg = find([stats.Area] > 80);
% BWG = ismember(labelmatrix(cc), idxg);
areag = max(cat(1, stats.Area));
if areag > detect
    g_detected = 'yes';
    gd = 1;
```

```
else g_detected = 'no';
    gd = 0;
end

cc = bwconncomp(ball2(:,:,:,3));
stats = regionprops(cc, 'Area');
% idxb = find([stats.Area] > 80);
% BWB = ismember(labelmatrix(cc), idxb);
areab = max(cat(1, stats.Area));
if areab > detect
    b_detected = 'yes';
    bd = 1;
else b_detected = 'no';
    bd = 0;
end

strr = sprintf('Was red found = %s',r_detected);
strg = sprintf('Was green found = %s',g_detected);
strb = sprintf('Was blue found = %s',b_detected);


figure(2)

subplot(2,2,1)
imshow(ball);

subplot(2,2,2)
imshow(ball2(:,:,:,1))
title(strr);

subplot(2,2,3)
imshow(ball2(:,:,:,2))
title(strg);

subplot(2,2,4)
imshow(ball2(:,:,:,3))
title(strb);

drawnow;
end
```

Pull_qr values

```
function [order, scanned_value] = pull_qr_excel()

fid = fopen('c:\users\bob\dropbox\myfile.txt');
scanned_value = fscanf(fid, '%s');
fclose(fid);

% drink_tracking = xlsread('c:\users\bob\dropbox\drink_counter.xls');
% headers = {'Vodka', 'Whiskey', 'Sour', 'RedBull'};

% comp = strcmp(old_value,scanned_value);
% if (comp ~= 1);
    switch scanned_value
        case 'vodka'
            order =  1;  %  1 hex, 0b1
            disp('vodka');
            old_value = 'vodka';

%             drink_tracking(1,1) = drink_tracking(1,1) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'screwdriver'
            order = 5; %  5 hex, 0b101
            disp('screwdriver');
            old_value = 'screwdriver';

%             drink_tracking(1,1) = drink_tracking(1,1) + 1;
%             drink_tracking(1,3) = drink_tracking(1,3) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'ragingbull'
            order = 9; % 9 hex, 0b1001
            disp('ragingbull');
            old_value = 'ragingbull';

%             drink_tracking(1,1) = drink_tracking(1,1) + 1;
%             drink_tracking(1,4) = drink_tracking(1,4) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'ragingsour'
            order = 13; % 13 dec, D hex,  0b1101
            disp('ragingsour');
            old_value = 'ragingsour';

%             drink_tracking(1,1) = drink_tracking(1,1) + 1;
%             drink_tracking(1,3) = drink_tracking(1,3) + 1;
%             drink_tracking(1,4) = drink_tracking(1,4) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'whiskey'
            order = 2; % 2 hex, 0b10
            disp('whiskey');
            old_value = 'whiskey';

%             drink_tracking(1,2) = drink_tracking(1,2) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');
```

```
       case 'whiskeysour'
            order = 6; % 0b110
            disp('whiskeysour');
            old_value = 'whiskeysour';

%             drink_tracking(1,2) = drink_tracking(1,2) + 1;
%             drink_tracking(1,3) = drink_tracking(1,3) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'kentuckycowboy'
            order = 10; % A hex, 10 dec,  0b1010
            disp('kentuckycowboy');
            old_value = 'kentuckycodboy';

%             drink_tracking(1,2) = drink_tracking(1,2) + 1;
%             drink_tracking(1,4) = drink_tracking(1,4) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'sourcowboy'
            order = 14; % 14 dec, E hex,  0b1110
            disp('sourcowboy');
            old_value = 'sourcowboy';

%             drink_tracking(1,2) = drink_tracking(1,2) + 1;
%             drink_tracking(1,3) = drink_tracking(1,3) + 1;
%             drink_tracking(1,4) = drink_tracking(1,4) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'sour'
            order = 4; % 4 hex, 0b100
            disp('sour');
            old_value = 'sour';

%             drink_tracking(1,3) = drink_tracking(1,3) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'energydrink'
            order = 8; % 8 hex,  0b1000
            disp('energydrink');
            old_value = 'energydrink';

%             drink_tracking(1,4) = drink_tracking(1,4) + 1;
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%             xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');
```

```
        case 'sourbull'
            order = 12; % 12 dec, 0b1100
            disp('sourbull');
            old_value = 'sourbull';

%              drink_tracking(1,3) = drink_tracking(1,3) + 1;
%              drink_tracking(1,4) = drink_tracking(1,4) + 1;
%              xlswrite('c:\users\bob\dropbox\drink_counter.xls', headers);
%              xlswrite('c:\users\bob\dropbox\drink_counter.xls',
drink_tracking, 'Sheet1', 'A2');

        case 'clear'
            order = 0;
            disp('No order');
            old_value = 'clear';

    end % end switch



end % end function
```

Main function

```
clear all
clc

%%%%%%%%%%%%%%%%%%%%%% variables n shit %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
threshold = 40;
opening = 100;
detect = 200;

order = 0;
stopbot = 0;
mode = 0;
waitonorder = 0;

count = 0;

order = uint8(order);
stopbot = uint8(stopbot);
mode = uint8(mode);
waitonorder = uint8(waitonorder);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ////////////////// webcam//////////////////////////////////
vid = videoinput('winvideo', 1, 'YUY2_352x288', 'ReturnedColorSpace','rgb');
start(vid);
preview(vid);
% //////////////////////////////////////////////////////

% //////////// opening the serial port /////////////////
```

```
serialport = serial('com10');
set(serialport,'baudrate', 9600)
set(serialport, 'databits',8);
set(serialport,'stopbits', 1);
set(serialport, 'flowcontrol', 'none');
set(serialport, 'terminator', 'CR');
fopen(serialport)
% //////////////////////////////////////////////////////

fwrite(serialport, [order mode stopbot waitonorder]);

while(1)
count = count + 1;

[rd,gd,bd] = get_colors( threshold, opening, detect, vid);

% /////////////////////// RED /////////////////////////////////////
if ((rd == 1) && (bd == 0) && (gd == 0)) % just red
    disp('stop... hammer time');
    stopbot = 1;
    stopbot = uint8(stopbot);
else
    stopbot = 0;
    stopbot = uint8(stopbot);
end
% ///////////////// END OF RED /////////////////////////



% /////////////////// GREEN /////////////////////////
if((gd == 1)) % just green
waitonorder = 1;
waitonorder = uint8(waitonorder);
% fwrite(serialport, [order mode stopbot waitonorder]);

[order, old_value] = pull_qr_excel;
order = uint8(order);

fwrite(serialport, [order mode stopbot waitonorder]);
else %
waitonorder = 0;
waitonorder = uint8(waitonorder);
end % of iff
% ////////////////////////////////////////////////////////




if (count > 10)
    clc
    count = 0;
end



order_stop_mode_wait = [order stopbot mode waitonorder]
```

```
mode = uint8(mode);
fwrite(serialport, [order mode stopbot waitonorder]);







end
```

Atmel / C code

```c

/**
 * \file
 *
 * \brief Empty user application template
 *
 */

/*
 * Include header files for all drivers that have been imported from
 * AVR Software Framework (ASF).
 */

#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"es
#include "mixing.h"

#define DbLedOn()          (PORTR.OUTCLR = 0x02)
#define DbLedOff()         (PORTR.OUTSET = 0x02)
#define DbLedToggle()      (PORTR.OUTTGL = 0x02)


// Collision IRS
int FRONT, REAR, poop, seat1, seat2, seat3, seat4;

// seat location
int orderlocation = 0;
int seat;


// moving
```

```c
int speed, toggle;
int filled = 0;

// which end
int move_direction, station_bump, end_bump;

// other variables
uint8_t order = 0;
uint8_t amp = 0;
int THRESHOLD = 1700;
int amount, realorder, randomseat, randomorder;
int mode = 0;
int stopbot = 0;
int waitonorder = 0;



// bartender functions
void seat_location();
void which_end();
void moving();

// movement functions
void go_forward(int speed);
void go_back(int speed);

// other functions
void read_sensors();
void avoid_obstacles();
//void fake_delay(uint16_t amount);
void get_order();




//
*****************************************************************************************
******************
// Start Main Function
//
*****************************************************************************************
******************

int main (void)
{
       board_init();
       RTC_DelayInit();
       DbLedOn();
       ATtinyServoInit();
       uartInit(&USARTC0,115200);
       uartInit(&USARTE1,9600);
       ADCsInits();
       sei();

    // making pins 0 and 1 on port d pull down resistors activated inputs
    PORTD.DIRCLR = 0xFF; // sets all of D as input
       PORTD.PIN0CTRL |= 0x10;
       PORTD.PIN1CTRL |= 0x10;
       //PORTB.DIRCLR = 0xFF;
       PORTB.PIN0CTRL |= 0x10;
       PORTB.PIN1CTRL |= 0x10;


       while(1)
       {



              get_order();
              //fprintf(&USB_str,"order = %d mode = %d stopbot = %d waitonorder = %d
\r\n\r\n",order, mode, stopbot, waitonorder);

              //if (mode == 0)
              //{

              seat_location();
              //fprintf(&USB_str,"seat1 =%d seat2 =%d seat3 =%d seat4 =%d
\r\n\r\n",seat1, seat2, seat3, seat4);

              //mixing(15);

              read_sensors();

              avoid_obstacles();

              which_end();
              //fprintf(&USB_str,"station bump = %d end bump = %d move direction
=%d\r\n\r\n",station_bump, end_bump, move_direction);

              moving();

              //fprintf(&USB_str,"order AFTER mixing = %d \r\n\r\n",order);
              //fprintf(&Xbee_str,"orderlocation = %d  seat = %d  filled = %d
\r\n\r\n",orderlocation, seat, filled);
              //}

              //if (mode == 1)
              //{

              //}

               //fprintf(&USB_str,"FRONT = %d    REAR = %d    \r\n\r\n", FRONT, REAR);
               //fprintf(&USB_str,"Seat = %d, orderlocation = %d, move direction = %d,
station_bump = %d, end_bump = %d      \r\n\r\n",seat,orderlocation,
move_direction,station_bump, end_bump);

              //fprintf(&USB_str,"The order is %d\r\n\r\n",order);
              //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",i,analogRead_ADCA(i));

              //fprintf(&Xbee_str,"%c", order);

              //rnd_turn = rand() % (HIGH - LOW + 1) + LOW;
              //randomseat = rand() % 3 + 1;
               //order = rand() % 9 + 1;
       }
       }

       //
******************************************************************************************
**************************************
       //
******************************************************************************************
**************************************
       // Start subfunctions
       //
******************************************************************************************
***************************************
       //
******************************************************************************************
****************************************




       //
******************************************************************************************
***************************************
       // bartending functions
       //
******************************************************************************************
***************************************


       void seat_location ()
       {

       if ((PORTA.IN & 0b11110000) == 0b11110000)
       {
       seat = 1;
              if ((order > 0) && (orderlocation == 0))
              {
              orderlocation = 1;
              }

       }

       else if ((PORTA.IN & 0b01110000) == 0b01110000)
       {
       seat = 2;
              if ((order > 0) && (orderlocation == 0))
              orderlocation = 2;

       }
       else if ((PORTA.IN & 0b00110000) == 0b00110000)
       {
       seat = 3;
              if ((order > 0) && (orderlocation == 0))
              orderlocation = 3;

       }
       else if ((PORTA.IN & 0b00010000) == 0b00010000)
```

```
{
seat = 4;
        if ((order > 0) && (orderlocation == 0))
        orderlocation = 4;

}
else
seat = 0;

}

void which_end ()
{

        if ((PORTB.IN & 0b00000001) == 0b00000001)
        {
                station_bump = 1;
                move_direction = 0;
        }
        else if ((PORTB.IN & 0b00000010) == 0b00000010)
        {
                end_bump = 1;
                move_direction = 1;
        }
        else
        {
                station_bump = 0;
                end_bump = 0;
        }

}


void moving ()
{

        // logic for at the ends
        if ((station_bump == 1) && (order > 0))
        {
                go_forward(0);
                _delay_ms(3000);
                mixing(order);
                filled = 1;
                order = 0;
                station_bump = 0;
            go_back(595);
                _delay_ms(100);
        }

        if (station_bump == 1)
        {
                go_forward(0);
                _delay_ms(1000);
                go_back(595);
                _delay_ms(100);
        }

                if (end_bump == 1)
```

```
              {
                     go_back(0);
                     _delay_ms(1000);
                     go_forward(590);
                     _delay_ms(100);
              }

              // logic for movement
       if (move_direction == 0)
              {
                     go_back(590);
                     _delay_ms(100);

                     if ((filled == 1) && (seat == orderlocation))
                     {
                            go_back(0);
                        _delay_ms(25000);
                        filled = 0;
                        orderlocation = 0;
                     }

                     if ((seat > 0) && (filled == 0))
                     {
                            go_back(0);
                            _delay_ms(1000);
                     }
              }


              if (move_direction == 1)
              {

                     go_forward(595);
                     _delay_ms(100);
                     if ((seat > 0) && (order == 0))
                     {
                            go_forward(0);
                            _delay_ms(1000);
                        go_forward(595);
                        _delay_ms(100);
                     }

              }


       }

       void get_order()
       {
              if (dataInBufE1())
              {

                     fscanf(&Xbee_str,"%c %c %c %c", &order, &mode, &stopbot,
&waitonorder);
                     if (waitonorder == 0)
                            order = 0;

                     //fprintf(&USB_str,"order received = %x \r\n\r\n",order);
              }
       }




       //
*****************************************************************************************
************************************
       // Obstacle detection and avoidance
       //
*****************************************************************************************
**************************************

void go_forward(int speed)
{
       //if (speed > 500)
        //turn on forward greed LED
       // else
        //turn off forward green led

       setMotorDuty(1,speed, MOTOR_DIR_FORWARD_gc);
       setMotorDuty(2,speed, MOTOR_DIR_FORWARD_gc);
       _delay_ms(100);

}

void go_back(int speed)
{
       //if (speed > 500)
       //turn on back green led
       //else
       // turn off back green led


       setMotorDuty(1,speed, MOTOR_DIR_BACKWARD_gc);
       setMotorDuty(2,speed, MOTOR_DIR_BACKWARD_gc);
       _delay_ms(100);

}

void avoid_obstacles()
{

     while (FRONT >= THRESHOLD)
        {
         go_forward(0);
         // turn on red led at front
         _delay_ms(1000);
         read_sensors();

        }

        // turn off red led at front

    while (REAR >= THRESHOLD)
       {
         go_back(0);
         //turn on red led at rear
         _delay_ms(1000);
         read_sensors();
       }

       // turn off red led at rear

       while (stopbot == 1)
       {
              // turn on stop led
       go_back(0);
       go_forward(0);
       get_order();
       }
            // turn off stop led
 }


void read_sensors()
{
    FRONT = analogRead_ADCA(0);
    REAR = analogRead_ADCA(1);
       seat1 = analogRead_ADCA(7);
       seat2 = analogRead_ADCA(6);
       seat3 = analogRead_ADCA(5);
       seat4 = analogRead_ADCA(4);
}
```