University of Florida
Intelligent Machine Design Lab (EEL5666)
PVC - Final Report

Student Name: Phan Vu

Date: 4-21- 09

Robot Name: Phantastic Vudoo Contraption (PVC)

Instructor Names: Dr. Arroyo and Dr. Schwartz

T.A. Names: Mike Pridgen and Thomas Vermeer

Table of Contents

Abstract ……...3

Introduction……..……3

Integrated System…….……....4

Mobile Platform……..……5

Actuation……..6

Sensors…….....7

Behaviors…….9

Conclusion…….11

Appendix A: PVC Behavior Code…….12

Appendix B: CMUCam1 Code……..21

Abstract

The Phantastic Vudoo Contraption (PVC) is the 1st generation autonomous robot ever to be made by made me. PVC task is to perform a search for an object based on its color by communicating with a CMUcam1 camera. After finding the object, PVC will then grab the object with its mechanical arm and drop it off at a designated location. Additionally, PVC uses three infrared sensors to avoid any incoming obstacles during the searching process.

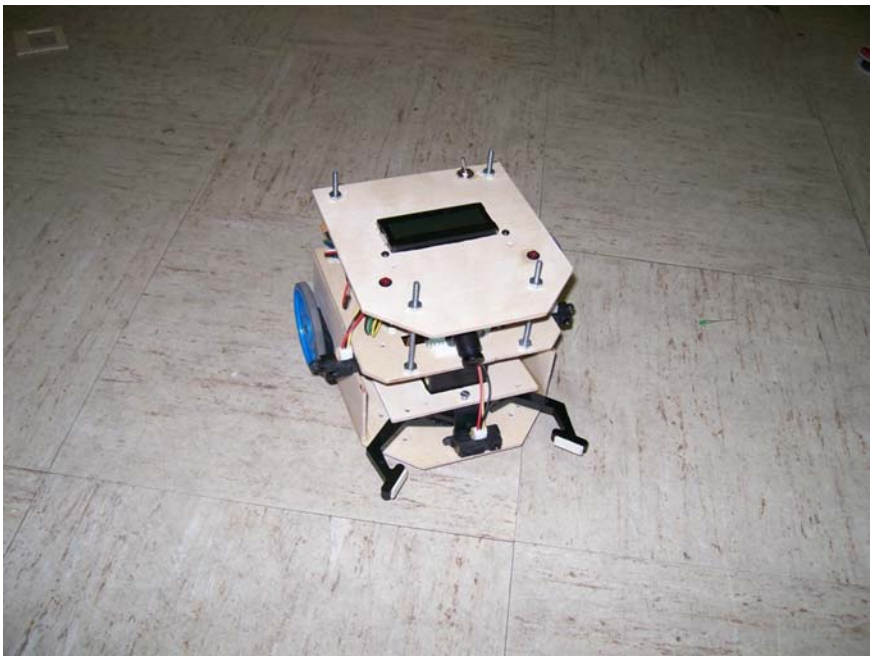

Figure 1: Phantastic Vudoo Contraption (PVC)

Introduction

At the beginning of this semester, I had a difficult time with developing a proposal for a robot that I would be building for this semester. I wanted to do something that could be useful and fun at the same time, so I've decided to build a robot that would behaves as retrieval bot. Unfortunately, a retrieval robot does not have that significant awesome factor I was looking for. After many thoughts, I've decided to make a retrieval robot that could also play a sport. And so, Phantastic Vudoo Contraption (PVC) was born.

The purpose of this report is to provide a detailed description on PVC. In short, PVC behaves as a handball player. PVC task is to search for a ball within an arena, while avoiding obstacles. Once the ball is detected, PVC will then pick up the ball and search for the opposing team goal. After doing so, PVC will place the ball in the team goal. To perform the indicated task, PVC will use a CMUcam1 sensor and three Sharp IR sensors

Integrated System

At the core of PVC's integrated system is the Pridgen-Vermeer Robotics (PVR) ATmega 128 processor board. The board was purchased from Thomas Vermeer and Mike Pridgen. It has six 8-bits I/P Ports, four analog to digital inputs, six pulse width modulator, two RS232 terminals. The board has a clock speed of 16Mhz. PVC uses the processor board to interface with three servos, three IR sensors, two LEDs, one LCD display, and a CMUcam1 camera.

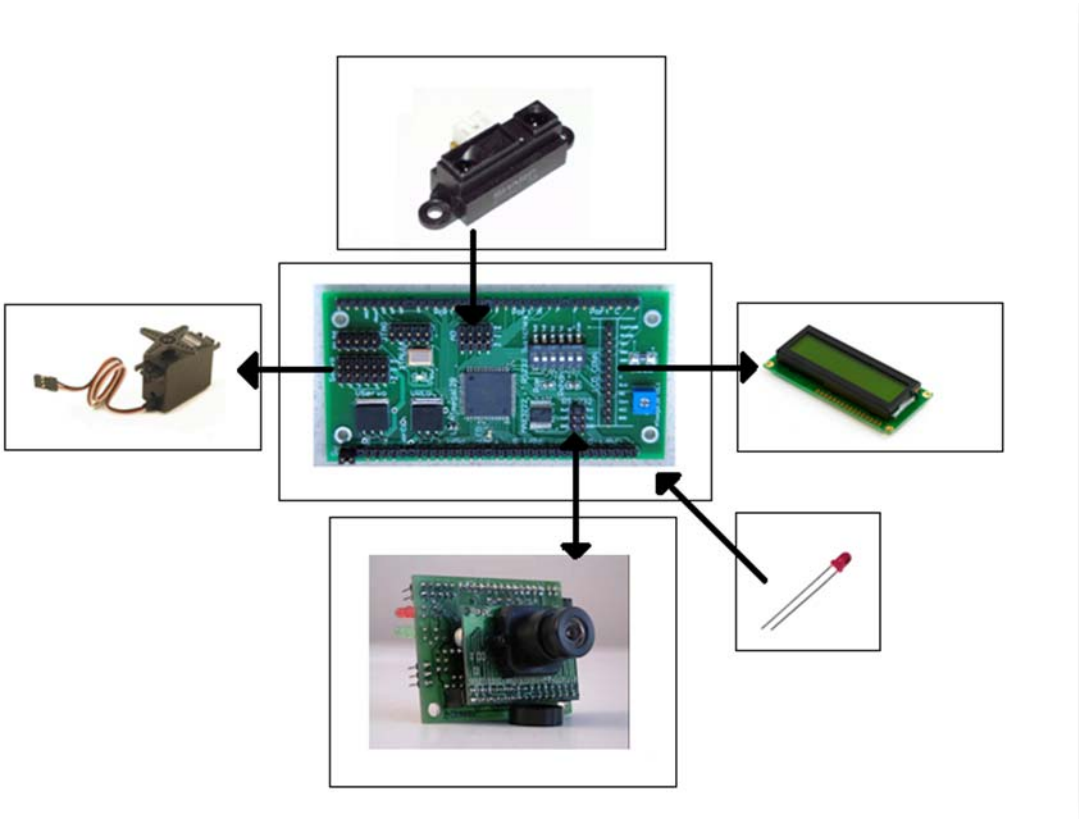

Figure 2: PVC integrated system. The camera seen is a CMUCam1. The servos are continuous parallax servos with a torque rating of 3.4 kg*cm. The LCD is a monochrome type that prints black character on green background. The infrared sensors are Sharp 2Y0A21 infrared sensor, which are capable of detecting an object between 2 inches to 14 inches.

Mobile Platform

My goal for designing PVC's mobile platform is to not limit PVC to a specific task (being a handball player). If possible, I would also like to add different modes of operations on PVC that will utilizes the abilities of CMUcam1 camera for different tasks, such delivering items, going to a specific point that is indicated by a laser, or following an object. As a result, I've decided to design PVC's mobile body to that of a small compact vehicle with differential steering so that it can maneuver around object easier. Additionally, carrying PVC around would also be more of a convenient. Adding a new component to PVC is easy since PVC can be completely dissembled in few minutes by unscrewing the fasteners attached PVC body. The material used to construct PVC's frame is wood.

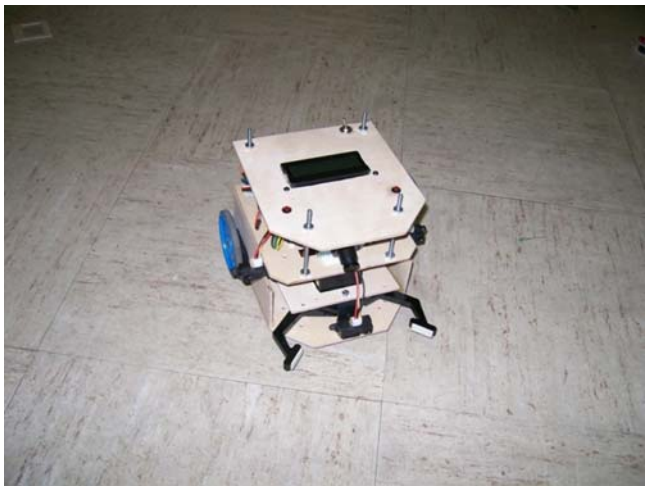

Figure 3: PVC Mobile Platform

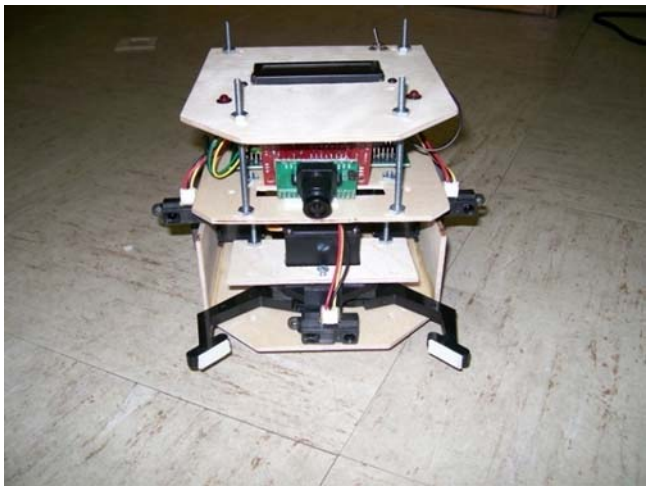

Figure 4: PVC Mobile Platform frontal view

Actuation

While there are many different ways of retrieving a ball, I wanted to use a gripper due to it versatile use. Originally, I wanted to design a gripper using two spur gears and one servo. However, due to a time constraint, I decided to buy a gripper instead. The gripper kit I bought can be seen in the picture below

Figure 5: Gripper kit. (Purchased from Robodyssey.com)

The gripper kit with the servo costs $26 (not including shipping and handling fee). It has an arm length of 4 inches and an arm span of 5.5 inches. The one I received was not machined that well. The left hand did not mate correctly with the right hand due to alignment issue. However, it works well when used with double sided tape.

Sensors

Sharp 2Y0A21 infrared sensors

The Sharp 2Y0A21 infrared sensor is an analog sensor used as a range finder. PVC uses the IR to avoid obstacles within a range of 2inches to 14 inches. Calibrating the three IR sensors mounted on PVC body yields the following result:

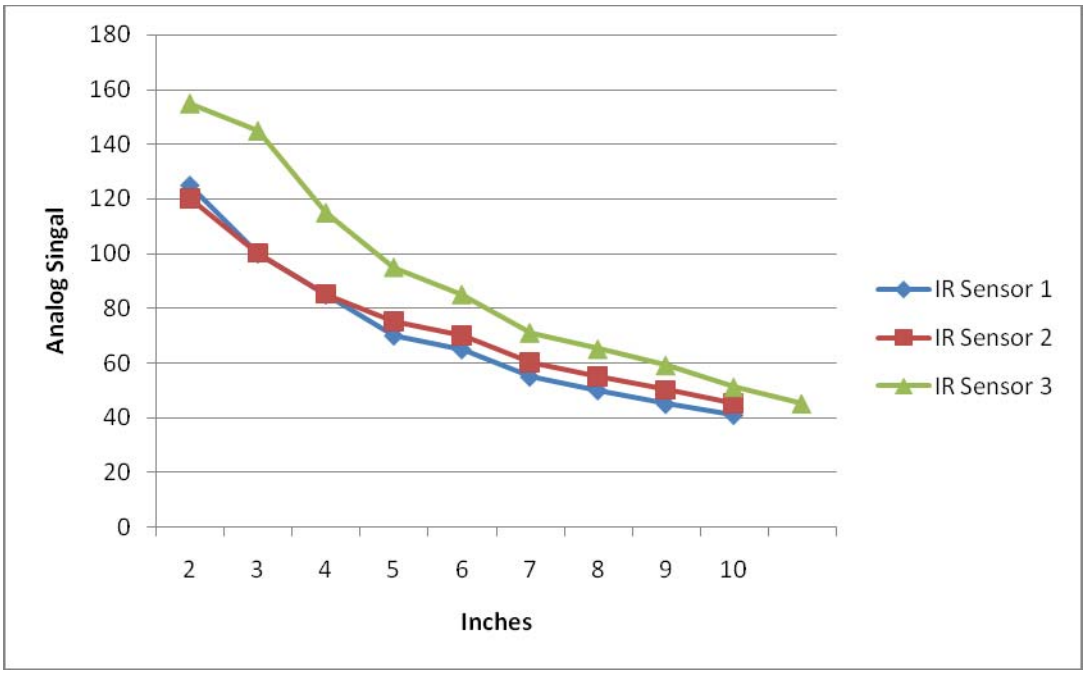

Figure 6: IR sensors reading vs distance. The distance is measured from PVC front bumpter to the object. IR sensor 1 and IR sensor 2 are placed with an offset of two inches away from the front bumper.

Data sheet for the infrared sensor can be found at:

http://www.acroname.com/robotics/parts/gp2y0a21yk_e.pdf

CMUcam1 camera

The CMU cam1 camera has the following functionality:

- Track user defined color blobs at 17 Frames Per Second
- Find the centroid of the blob
- Gather mean color and variance data

- Transfer a real-time binary bitmap of the tracked pixels in an image
- Arbitrary image windowing
- Adjust the camera's image properties
- Dump a raw image
- 80x143 Resolution
- 115,200 / 38,400 / 19,200 / 9600 baud serial communication
- Slave parallel image processing mode off a single camera bus
- Automatically detect a color and drive a servo to track an object upon startup
- Ability to control 1 servo or have 1 digital I/O pin
-

The camera requires at least 200 mA and a voltage between 6 and 7 volts. Calibrating the camera focus can be done through a GUI interface on window. My robot will communicates with the camera through the RS232 port in the PVR board. The camera is going to be used to track a ball and a target positions.

Using the CMUcam1 camera to track an object by its color is difficult. If the surrounding environment contains objects with similar colors, the camera will not be able to determine the correct position. Variations in pixel values will most likely occur due to the variation in lighting in the surrounding environment. To reduce the risk of having the camera pick up an object with similar color, I used bright color object. I also use a calibration routine upon startup to adjust to different lightings condition.

Datasheet for the CMUcam1 and the infrared sensor used can be found at the following website:

http://www.seattlerobotics.com/CMUcamManualv15A.pdf

Behaviors

PVC has two main behaviors; one is to retrieve a ball and to drop it off at a designated location, the other is to perform obstacle avoidance. PVC's method for retrieving an object is to use the track color command with the CMUcam1. By doing so, PVC can determine the middle mass coordinate position of the ball. Once the object position is known, PVC will slowly approach the target until a threshold has been reach for the front IR sensors. When that threshold is reached, PVC will either release the object or grab the object.

PVC's second behavior is to perform obstacle avoidance through the use of Fuzzy Logic. This is my first time learning about Fuzzy Logic. I followed the steps provided by the power point presentation provided by Kevin Harrelson, a previous student of IMDL. First, the outputs of the IR sensors first have to be classified as close, mid, or far, based on the Fuzzfication rule.

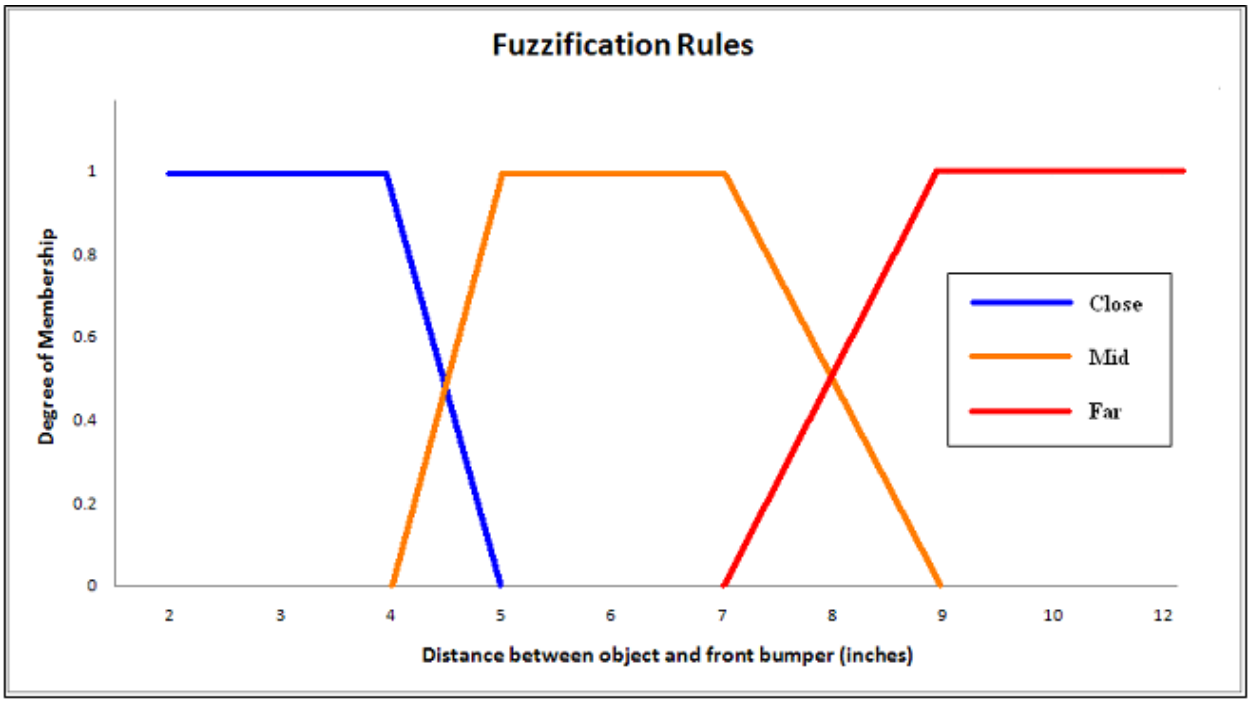

Figure 7: Fuzzfication rules for classifying an object distance.

After doing so, a fuzzy logic table was made to identify the appropriate behavior based on the fuzzy output obtained from the fuzzification rules.

Table 1: Fuzzy Logic table. LF = Left far, LM =Left mid, LC = Left close, RF = Right far, RM = Right mid, RC = Right close, MF = Middle far, MM =Middle mid, MC = Middle close, L = Left turn, R= Right turn, B = Back, F = Forward.

	Right Sensor/Middle Sensor								
Left Sensor	RF/MF	RF/MM	RF/MC	RM/MF	RM/MM	RM/MC	RC/MF	RC/MM	RC/MC
LF	F	L	B	L	L	B	L	L	B
LM	R	R	B	R	R	B	L	L	B
LC	R	R	B	R	R	B	B	B	B

Based on the Fuzzy Logic table, the following logic equations were then derived:

F = LF*RF*MF

B= (RF*MC+RM*MC+RC*MC)*(LF+LM+LC) +LC*(RM*MC+RC*MF+RC*MC)

R = (LM+LC)*(RF*MF+RF*MM+RM*MF+RM*MM)

L = LF*(RF*MM+RM*MF+RM*MM+RC*MF+RC*MM)+(LF+LM)*(RC*MF+RC*MM)

The asterisk, *, represent a fuzzy logic AND, which takes in inputs and output the maximum value. The plus, +, represent a fuzzy logic OR, which takes in the inputs and output the minimum value. For example:

AND (0, 30) = 0

OR (0,100) = 100

After calculating for F, B, R, and L, PVC will then performs the appropriate task based on a winner takes all system. For instance, if F is the largest value, PVC will move forward.

Conclusion

At the end of the project, PVC was only able to track down a ball and pick it up. All in all, I would say the project is a minor success. PVC was able to avoid obstacles and pick up a ball through the use of the camera and the three infrared sensors. Unfortunately, the behavior code I wrote for picking up the ball does not work as well as I wanted. There was an inconsistency in how PVC would approach the ball due to being confuse with similar colors in the surrounding environment. I also found that using fuzzy logic to perform obstacle avoidance with the infrared sensor does not improve the smoothness that much. Due to PVC's traveling speed and the short range on the infrared sensor, the obstacle avoidance behavior had to be abrupt. Nonetheless, I still found it to be a great learning experience.

If I had to do it again, I would definitely spend more time developing PVC mobile platform and actuation system at the beginning of the semester, so that I didn't have to make alterations to the original design throughout the semester. I also had to compensate for some design flaw through software. Additionally, I would build an arena for PVC to operate to reduce the level of interference from the surrounding environment.

Appendix A: PVC Main Software Code

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "sleep.h"
#include "LCD.h"
#include "uart0.h"
#include "cmucam1.h"
#include "pvr_servos.h"

typedef unsigned int u8;
volatile u8 temp;
volatile u8 i = 0;
volatile u8 cmudata[10];
volatile u8 R_object;          //Red value for object to be track
volatile u8 G_object;          //Green value for object to be track
volatile u8 B_object;          //Blue value for object to be track

#define OR( a, b ) ( ((a) > (b)) ? (a) : (b) )   //Fuzzy Logic OR
#define AND( a, b ) ( ((a) < (b)) ? (a) : (b) )  //Fuzzy Logic AND
```

```
void move (float x, float y);
void obstacle_avoid(int RightIR, int LeftIR, int FrontIR);
void FlashLight(int x);

int main(void) {
	//Initialize Sleep (used for delay)
	initSleep();
	//Initialize LCD settings (must initialize sleep prior to doing this)
	lcdInit();
	lcdString("Initialzing Settings");
	FlashLight(5);
	//Initialize Servos
	initServo();
	FlashLight(5);
	//Initialize baud rate setting to 115,200 for a 16Mhz processer speed
	uart0_init (8);
	//Initialize cmucam1 settings refer to source file for more information
	init_camera();
	FlashLight(5);
	//Initialize ADCs
	initADC();
	//Begin tracking ball color
	lcdClear();
	lcdString("Hold target in front");
	lcdGoto(1,0);
```

```
lcdString("of camera");
uartstring("TW\r");
ms_sleep(1000);
lcdClear();
lcdString("Hide Ball now");
ms_sleep(5000);
lcdClear();
lcdString("Searching for ball");

for (;;) {
        uartstring("MM 1\r");
        ms_sleep(1000);
        do {
                uartstring("TC\r");
                displayMpacket(cmudata);
                obstacle_avoid(adcOne(),adcTwo(),adcThree());
        }while( (cmudata[2]==0)||(cmudata[3]==0)||(cmudata[8]==0) );
        lcdClear();
        lcdString("Ball found");
        lcdGoto(1,0);
        lcdString("Centering...");
        do {
                uartstring("TC\r");
                displayMpacket(cmudata);
                while(cmudata[2]<=42){
```

```
                //Turn Left
                move(0,4);
                uartstring("TC\r");
                displayMpacket(cmudata);
        }
        while(cmudata[2]>=48) {
                //Turn Right
                move(-3,0);
                uartstring("TC\r");
                displayMpacket(cmudata);
        }
        move(-2,2);
        lcdClear();
        ms_sleep(500);
}while( (cmudata[2]<42)||(cmudata[2]>48)||(adcThree()<50));
lcdClear();
lcdString("Ball found");
while( (adcThree()>50)&&(adcThree()<=145) ){
        move(-2,2);
        lcdInt(adcThree());
}
while (adcThree()>100);
moveServo3(100);
ms_sleep(1000);
moveServo3(0);
```

```
        }

}//end main


// USART Interrupt routine for storing cmucam1 outputs.
// This code was written by Kyle Tripician
ISR(USART0_RX_vect) {
        temp = UDR0;
        // 0x3A = ':'
        // 0x20 = space
        // 0x09 = tab
        // anything less than 9 = other indicators like end of text, etc...
        // 0xFF = a byte received having a value of 255

        if(temp != 0x3A) {
                if(temp == 0x20 || i==10) {
                        i=0;
                }
                else if(i==0){
                        if(temp == 0xFF){
                                cmudata[i]=temp;
                                i++;
                        }
                }
```

```
                else if(temp !=0x20 && i<10) {
                        cmudata[i]=temp;
                        i++;
                }
        }//if
}//end
void FlashLight(int x){
   DDRB = 0b00000001;
        int i;
        for (i=0;i<x;x--){
                ms_sleep(100);
                PORTB = 0b11111111;
                ms_sleep(100);
                PORTB = 0b00000000;
                ms_sleep(100);
        }
}
void move (float x, float y){
        moveServo2(x);
        moveServo1(y);
}void obstacle_avoid(int RightIR, int LeftIR, int FrontIR)
{
        //Avoid obstacles using fuzzy logic
        int temp1=RightIR;
        int temp2=LeftIR;
```

```
int temp3=FrontIR;


//L = Left IR, R = Right IR
int RF=0,RM=0,RC=0,MF=0,MM=0,MC=0,LF=0,LM=0,LC=0;
int Forward=0,Back=0,Left=0,Right=0;


//Fuzzification for right IR sensor
if (temp1>=85) RC =100;
else if ( (temp1>70)&&(temp1<85) ) {
        RC = 100+(0-100)*(85-temp1)/15;
        RM = 100-RC;
}
else if ( (temp1>=55)&&(temp1<=70) ) RM=100;
else if ( (temp1>45)&&(temp1<55) ) {
        RM = 100+(0-100)*(55-temp1)/10;
        RF = 100-RM;
}
else if (temp1<=45) RF =100;



//Fuzzification for Left IR sensor
if (temp2>=85) LC =100;
else if ( (temp2>75)&&(temp2<85) ) {
        LC = 100+(0-100)*(85-temp2)/10;
        LM = 100-LC;
```

```
}
else if ( (temp2>=60)&&(temp2<=75) ) LM=100;
else if ( (temp2>50)&&(temp2<60) ) {
        LM = 100+(0-100)*(60-temp2)/10;
        LF = 100-LM;
}
else if (temp2<=50) LF =100;



//Fuzzification for Middle IR sensor
if (temp3>=115) MC =100;
else if ( (temp3>95)&&(temp3<115) ) {
        MC = 100+(0-100)*(115-temp3)/20;
        MM = 100-MC;
}
else if ( (temp3>=71)&&(temp3<=95) ) MM=100;
else if ( (temp3>59)&&(temp3<71) ) {
        MM = 100+(0-100)*(71-temp3)/12;
        MF = 100-MM;
}
else if (temp3<=59) MF =100;
//Fuzzy Logic
int RFMF=AND(RF,MF),RFMM= AND(RF,MM),RFMC= AND(RF,MC);
int RMMF=AND(RM,MF),RMMM= AND(RM,MM),RMMC= AND(RM,MC);
int RCMF=AND(RC,MF),RCMM= AND(RC,MM),RCMC= AND(RC,MC);
```

```
        //F = LF*RF*MF

        //B = (RF*MC+RM*MC+RC*MC)*(LF+LM+LC) + LC*(RM*MC+RC*MF+RC*MC)

        //R = (LM+LC)*(RF*MF+RF*MM+RM*MF+RM*MM)

        //L =
LF*(RF*MM+RM*MF+RM*MM+RC*MF+RC*MM)+(LF+LM)*(RC*MF+RC*MM)

        Forward = AND(AND(LF,RF),MF);

        Back =
OR(AND(OR(OR(RFMC,RMMC),RCMC),OR(OR(LF,LM),LC)),AND(LC,OR(OR(RMMC,R
CMF),RCMC)));

        Right = AND(OR(LM,LC),OR(OR(OR(RFMF,RFMM),RMMF),RMMM));

        Left =
AND(AND(LF,OR(OR(OR(OR(RFMM,RMMF),RMMM),RCMF),RCMM)),AND(OR(LF,LM)
,OR(RCMF,RCMM)));

//Defuzzification by computing for the weighted average

        //Left Servo (Servo 2) Direction Output Mappings:

        //Back = -100, Forward = 100, Right = 100, Left = -100;

        //Right Servo (Servo 1) Direction Output Mappings:

        //Back = 100, Forward = -100, Right = 100, Left = -100;

        //Employ winner takes all

        if ( (Forward>=Back)&&(Forward>=Right)&&(Forward>=Left) ) move(-6,6);

        if ( (Back>=Forward)&&(Back>=Right)&&(Back>=Left) ) move(6,-6);

        if ( (Left>=Back)&&(Left>=Right)&&(Left>=Forward) ) move(0,5);

        if ( (Right>=Back)&&(Right>=Forward)&&(Right>=Left) ) move(-3,0);

}
```

Appendix B: CMUcam1 Software Code

These codes are from Amer Quouneh, a student of IMDL in Spring 2009. I modified them a little bit, but they are his code.

```
#include <avr/io.h>
#include "cmucam1.h"
#include "sleep.h"
// The following codes are from Amer Qouneh.  I just modified them for organizational sake.
//Initalize cmucam1
void init_camera() {
        // Reset the camera with several RS commands
        uartstring("RS \r");
        ms_sleep(200);
        // Turn on auto tracking LED to indicate camera is being initialize
        uartstring("L1 1\r");
        ms_sleep(100);
        uartstring("NF 1\r");
        uartstring("CR 5 255 6 255/r");
        // wait for 10 seconds to adjust to lighting conditions
        for(int j=0; j<10; j++) {
                ms_sleep(500);
        }

        // Set poll mode; 1 packet
        uartstring("PM 1\r");
        ms_sleep(100);
```

```
	// Set raw mode
	uartstring("RM 3\r");
	ms_sleep(100);

	// Turn off auto tracking LED to indicate camera is finish initializing
	uartstring("L1 2\r");
}


void displayMpacket(unsigned int cmudat[]) {
	unsigned int mmx, mmy, lcx, lcy, rcx, rcy, pix, conf, packetName;

	// Display the packet vlaues
	packetName = cmudat[1];		// 'S' = 83, 'M' = 77, 'C' = 67
	mmx = cmudat[2];
	mmy = cmudat[3];
	lcx = cmudat[4];
	lcy = cmudat[5];
	rcx = cmudat[6];
	rcy = cmudat[7];
	pix = cmudat[8];
	conf = cmudat[9];

//	Identify_Packet(packetName);
```

```
        lcdClear();
        lcdString("X_mid:");
        lcdInt(mmx);
        lcdGoto(1,0);
        lcdString("Y_mid: ");
        lcdInt(mmy);
        lcdInt(pix);
/*
        lcdGoto(0,0);
        lcdString("Left Corner X");
        lcdGoto(1,0);
        lcdInt(lcx);
        ms_sleep(3000);
        lcdClear();

        lcdClear();
        lcdGoto(0,0);
        lcdString("Left Corner Y");
        lcdGoto(1,0);
        lcdInt(lcy);
        ms_sleep(3000);

        lcdClear();
        lcdGoto(0,0);
```

```
lcdString("Right Corner X");
lcdGoto(1,0);
lcdInt(rcx);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Right Corner Y");
lcdGoto(1,0);
lcdInt(rcy);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Pixels");
lcdGoto(1,0);
lcdInt(pix);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Confidence");
lcdGoto(1,0);
lcdInt(conf);
ms_sleep(3000);
```

```
*/
}


// Display an S packet
void displaySpacket(unsigned int cmudat[]) {
        unsigned int packetName, rMean, gMean, bMean, rDev, gDev, bDev;
        packetName = cmudat[1];                 // 'S' = 83, 'M' = 77
        rMean = cmudat[2];
        gMean = cmudat[3];
        bMean = cmudat[4];
        rDev = cmudat[5];
        gDev = cmudat[6];
        bDev = cmudat[7];

        // Display the packet vlaues
        Identify_Packet(packetName);
        ms_sleep(3000);

        lcdClear();
        lcdGoto(0,0);
        lcdString("Red Mean");
        lcdGoto(1,0);
        lcdInt(rMean);
        ms_sleep(3000);
```

```
lcdClear();
lcdGoto(0,0);
lcdString("Green Mean");
lcdGoto(1,0);
lcdInt(gMean);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Blue Mean");
lcdGoto(1,0);
lcdInt(bMean);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Red Deviation");
lcdGoto(1,0);
lcdInt(rDev);
ms_sleep(3000);

lcdClear();
lcdGoto(0,0);
lcdString("Green Deviation");
```

```
	lcdGoto(1,0);
	lcdInt(gDev);
	ms_sleep(3000);

	lcdClear();
	lcdGoto(0,0);
	lcdString("Blue Deviation");
	lcdGoto(1,0);
	lcdInt(bDev);
	ms_sleep(3000);
}

// Display Packet Name
void Identify_Packet(unsigned int Packet_Name) {
	lcdClear();
	lcdString("Packet Name: ");
	// 'S' = 83, 'M' = 77, 'C' = 67
	if(Packet_Name == 83) lcdString("S");
	else if(Packet_Name == 77) lcdString("M");
	else if(Packet_Name == 67) lcdString("C");
	else lcdString("Packet ERROR!");
	ms_sleep(3000);
}
```