Date: 4/20/2009
Student Name: Amer Qouneh

TAs : Mike Pridgen
Thomas Vermeer

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

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

Gossima

Final Report

Table of Contents

Abstract

"Gossima" was one of the original names given for the current game of ping-pong. Gossima continues to live on, and takes on the personality of an autonomous ping-pong ball-collecting robot. It certainly is not fun to interrupt a game when balls run out. Gossima is designed to help players play continuously without having to interrupt the game to collect the balls. Equipped with a CMU camera, ultrasonic sensor, infrared sensors, and servos, Gossima will be able to search for, retrieve, and shoot ping-pong balls while avoiding obstacles. Some difficulties were faced when approaching a ball, as tracking the color at close distances was not always successful.

Executive Summary

Gossima's task is to search for, find, track, capture, and shoot ping-pong balls. It possesses behaviors that allow it to carry out its tasks. The search behavior allows it to find orange ping-pong balls, the acquiring behavior allows capturing and shooting a ball. The last behavior allows avoiding obstacles.

Gossima depends on a CMU camera to find an orange-colored ball. The camera allows the robot to track the blob of color as it moves and approaches its target. To that end, it also uses an ultrasonic sensor that provides range information for faster detection of balls and objects. The CMU camera provides relatively accurate color and position information for the robot but that information is somewhat slow. The ultrasonic sensor provides faster position information but somewhat inaccurate because of its wide beam. The ultrasonic sensor first quickly detects the presence and direction of an object, and then the camera is employed to determine if that object is an orange ball or just an obstacle. Combining the strengths of both sensors allows faster initial detection of balls.

Infrared sensors along with the ultrasonic sensor provide the robot with obstacle avoidance capability. Two infrared sensors placed on the front sides of the robot detect the presence of any close objects. Based on this information, the robot turns to avoid the obstacle. Bump switches also help in avoiding obstacles that bump the robot from the back when backing up.

Gossima can capture a ball by lowering a cylinder in front of the ball and a scoop door pushes the ball inside, thereby securing it. Once a ball is captured, the cylinder is raised vertically and the ball falls into a shooting position. The shooter is quite simple consisting of a small flexible piece of plastic that is pulled back by a small servo just like a slingshot. Upon the release of the hammer, the plastic piece hits the ball and shoots out of the robot body.

Introduction

Interrupting a game of ping-pong to collect the balls is not fun. A robot that can collect the balls and relieve the players of the chore would be fun and amusing to watch. The robot is able to avoid obstacles, find the balls and retrieve them. This robot is named Gossima after the old name of the current ping-pong game.

To find ping-pong balls, a robot must be able to see the ball and distinguish it from the other objects. Gossima uses a CMU camera for eyes and it determines if the objects are balls or not. This type of camera has been successfully used in robotics and is simple to use. However, I think it is a bit slow for tracking colors while a robot is moving.

The robot has infrared sensors that help it to avoid obstacles. An ultrasonic sensor takes a dual role of obstacle avoidance and ball detection. With theses capabilities, a robot is able to navigate its way and find a ball to capture it.

Gossima is built out of a simple wooden platform with the different parts attached to its surface. It employs a cylinder to capture balls and a small slingshot-like mechanism to shoot balls.

Integrated System

Gossima relies on its electronic and mechanical capabilities to perform its duties. An Atmel microcontroller will orchestrate all of its functions by using its CMU camera, bump switches, IR, and ultrasonic sensors. Mechanically, the robot will control a set of 6 servos to move itself forward, backward, and to make turns. In addition, the robot will control a cylinder that captures a ball and loads it to a shooter. A battery of suitable power will be used and all of the above components will reside on a wooden platform.

An Atmel Atmega128 microcontroller board with I/O ports and pins will act as the controller for the robot. I programmed the microcontroller using the language C that will make it much easier than writing code in assembly language. I used the PVR board, which is a microcontroller board that is based on the ATMEL Atmega128 microcontroller. The board is equipped with an excellent selection of peripherals that will help in controlling the robot. I used the serial port (RS 232) to communicate with CMU camera, the six servo ports to control the servos (six of them), and the Analog to Digital ports (A/D) to read the values returned by the ultrasonic sensor, and the two infrared sensors.

Three bump switches (tactile) placed on the backside of the robot will detect obstacles on impact. The switches are connected to I/O ports through pull-up resistors. These will give the robot sensory information when it is backing up and hitting an obstacle. Two IR sensors placed in the front will provide Gossima with the ability to avoid obstacles once they are about six inches from the robot. The robot will turn to the opposite direction if an obstacle is detected.

A CMU camera is placed at the front of the robot. It will allow the robot to find the ping-pong balls that are distinguished by color (orange). The camera is controlled by a servo that will allow it to tilt in the vertical plane to search for balls from near to far. Using the camera, the robot detects the approximate location of the ball using an XY coordinate system. Based on the coordinates, the robot determines in what direction to move.

An ultrasonic sensor in the front will provide the robot with the ability to act as a ranger for detecting the balls, thereby helping the camera in its search effort. I found that the CMU camera is relatively slow in detecting the presence of balls, however, the ultrasonic sensor was much faster in determining the general direction of the ball. The ultrasonic sensor is also used in obstacle avoidance for the front when it is not searching or tracking a ball.

A battery pack of six NiMh rechargeable batteries act as the source of power for the robot. I found that they were sufficient for my purposes and lasted about a half hour of running and testing.

I used six servos for carrying out the mobility and capturing functions. Two of them were micro servos that needed to be light. Three servos were hacked to rotate continuously, two of them for the wheels, and one for the shooting mechanism.

The above sensors provide Gossima with the intended behaviors of roaming, searching for balls, shooting, and avoiding obstacles. Once a ball is found, Gossima activates a different behavior, that of acquiring the ball, by engaging an arm mechanism (cylinder). The arm is lowered from the vertical position to the forward position, and captures the ball by scooping it with a servo-operated door. Then Gossima exhibits a behavior that allows it to find a bin of a particular color to shoot at. Figure shows a block diagram of Gossima.

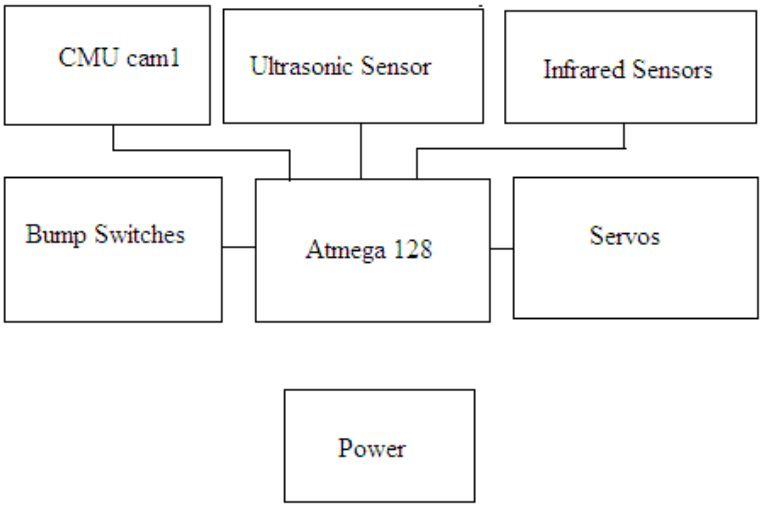

Figure 1

Mobile Platform

A simple wooden platform serves as the body of Gossima. It is rectangular in shape and covers a box that houses the two wheel servos. All the components are placed on the platform except the battery pack, which is placed on a v-shaped piece of wood attached to the box to give a further support for the robot. A caster ball supports the back end of the body. The platform also has a cylinder (capturing arm) that is lowered when needed. The cylinder feeds the shooting mechanism (a flexible piece of plastic). A picture of the robot is provided.

I learned that working with wood is difficult in the long run. It is certainly faster and simpler to use wood for a platform, but as more adjustments are made to the platform, the more difficult it becomes to work with. Wood screws do not stay firm in place, which results in components falling off. A metal frame might be a better choice for a robot. Also, it might be a good idea to build a metal platform that can be used for different projects with little modification.

Actuation

The robot uses six servos for controlling the operation of the robot. Two hacked servos for the wheels, one hacked micro servo for the shooting mechanism, one for the cylinder, one for the scoop door, and one for the camera base. All six servos are attached to the six servo ports on the PVR board. The ports provided the necessary PWM signals to the servos. Controlling the servos was simple. However, two days before demo day, three ports stopped working. All efforts to fix the problem had failed. Thomas Vermeer, one of

the TAs with the help of fellow students Jared Bevis and Joshua Childs, replaced the Atmega chip on the PVR board, but the problem persisted through demo day preventing the robot from performing capturing and shooting the balls.

Using servos as wheels is simple after performing the necessary modifications. Sending a large or a small value can control speed. Sending a positive or a negative value can also control direction of rotation. However, servos are slow as wheels, but acceptable for the purposes I am using them for. The other three servos were not hacked and were used to move within a certain range. One servo controlled the movement of the cylinder from the vertical to the horizontal position. Another servo controlled the movement of the scoop door. This servo was placed on the cylinder itself, which is why I chose it to be a micro servo. The shooter servo was used to rotate whenever the robot wanted to shoot a ball. When it rotates, it pulls a flexible piece of plastic (the handle of a plastic spoon), suggested by Thomas Vermeer, and then releases it to strike a ping-pong ball. The camera servo is used to tilt the camera forward and backward. The range of motion was not large as the camera is not capable of detecting objects at far distances. The servo controlled a metal wire connected to the base of the camera. If pushed forward, the camera tilts down, if pushed back, the camera tilts up.

In general, the servos were surprisingly simple to use and my application did not require large torques or power.

Sensors

Bump Switches

Three bump switches placed on the back are used for obstacle detection. Once a bump switch is pressed, this indicates to the robot that it has bumped into an object. The robot then takes a corrective action and changes its direction randomly. The switches were connected to I/O ports through pull-up resistors. They were small tactile switches intended for surface mounting on a PCB. I found that these are not good as bump switches because they are so fragile. I lost the pins of two of them with normal use. Also, the button itself was not soft enough to detect a light bump on the back of the robot. These tactile switches needed some extra force to make a contact.

Infrared Sensors

Two Sharp IR sensors allowed the robot to detect close obstacles to its sides. Each IR sensor was connected to an A/D port. The microcontroller read the value returned by the sensor and took action. After testing, I determined that for the size of the robot, a distance of about 6 to 7 inches was adequate to make a safe turn. The data sheet indicated a graph that it was not linear; that is the values returned for the distance did not vary linearly with the distance. After testing it myself, I did obtain a similar result. Figure 2 shows the graph for one of the IR sensors.

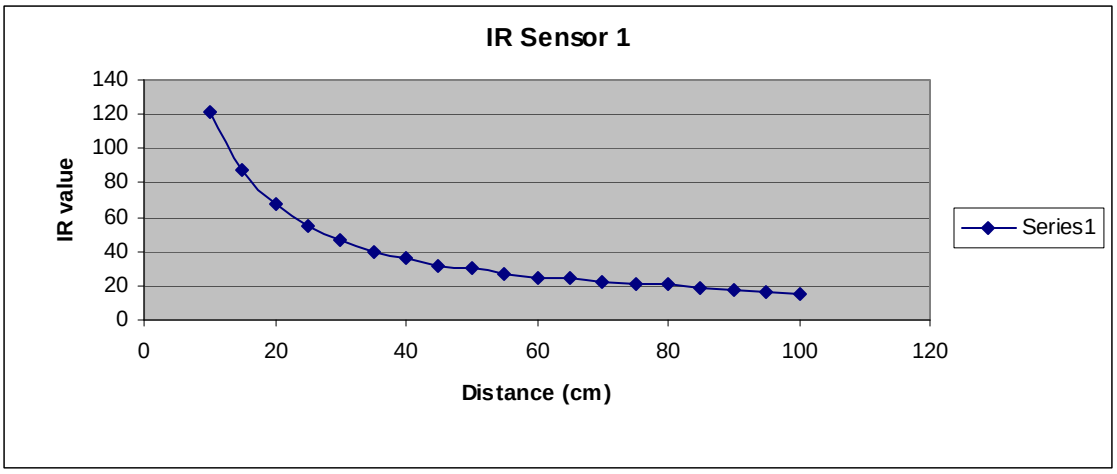

Figure 2

Ultrasonic Sensor

One ultrasonic sensor is used as simple radar for ranging. In one role, it is used in obstacle avoidance for the front of the robot. In a second role, it is used as ranges for detecting ping-pong balls. I found that using the ultrasonic sensor for detecting a ball is very useful as it allows faster detection than the CMU camera. I took advantage of this property and searched for objects using the ultrasonic sensor. After finding one, the camera is used to decide whether it is an orange ball or not. The sensor had to be tested for different positions. I finally placed it at the bottom close to the ground and pointing slightly upward away from the ground. Another good place would have been at the top looking slightly downward. I was able to detect a ball at maximum distance of about 1 m when placed close to the ground. Figure 3 shows a graph of the returned values.

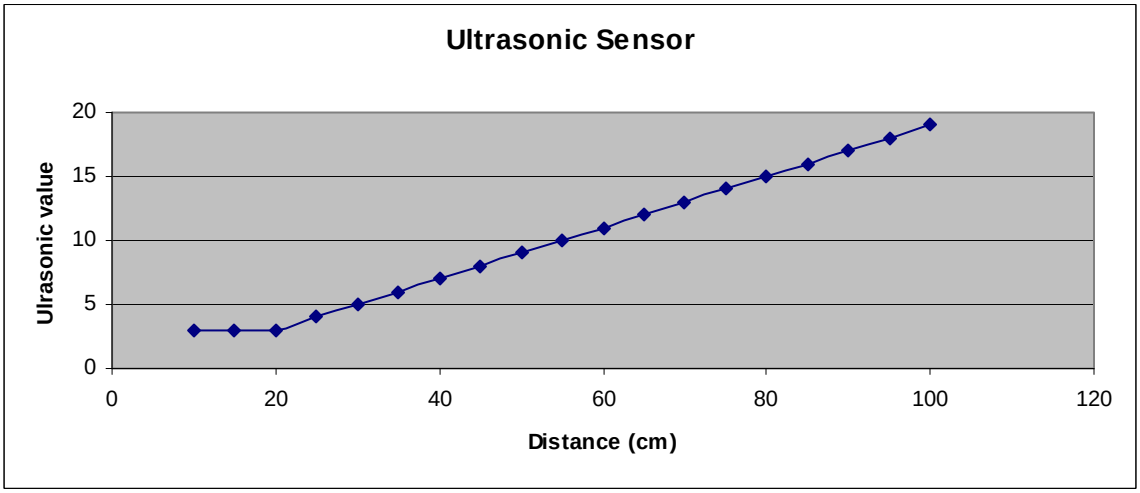

Figure 3

CMU Camera

The CMU camera is used to detect a blob of color and return its pixel position through a serial connection. The microcontroller board communicated with the camera through its serial RS232 port. The microcontroller had to be configured to use the capability, which was not difficult as there are many sources online about using serial ports. Use of the camera is not difficult but did give me some hard time in the beginning. I decided to use the poll mode, which means that the camera will send a packet of data whenever the microcontroller asks for one. The other mode is a continuous mode, which sends packets continuously. The camera can track a given color by specifying the RGB value for that color. Equivalently, the camera can also acquire the color values by using the command TW (Track Window) while placing the object in front of the camera. I used both methods and found that telling the camera what color to look for was slightly better than asking it to determine the color of the object.

The downside for using a CMU camera was that it required a little more time to return data since it processes the image and then sends the data using the serial connection, which is slow by nature. In my robot, there was a problem when tracking an orange ball. Whenever the robot detected a ball, it would approach but would lose the ball when at close distances because the ball would fall out of the screen. Fast movement of the robot would cause this to happen.

In addition, I found that the CMU camera did not handle the orange color and size of the ping-pong ball well. Since lighting is detrimental for the success of tracking, I found that even by placing a ball in front of the camera while stationary, the camera did not even recognize the orange ball as orange. The values returned had a low confidence value. As explained in the camera documentation, a confidence value of greater than 50 means that there is a high probability that it found the intended color. My camera did not return such high confidence values. The values returned ranged from 5 to upper 40. I did use the confidence value to determine whether that was an orange color or not. In my code I used a testing sequence that tested the confidence value. If it were greater than 50, then it is orange. But in practice, the robot was not able to determine it was seeing orange because the confidence value was low. Lowering the acceptance value would cause the robot to assume other colors and objects as orange. I think that this is the reason my robot was not very successful at tracking the balls.

Another reason for not being able to track a ball correctly might be that the size of the ping-pong ball is relatively small. A larger size would certainly give the camera more to work with. The ping-pong ball appears as 1 pixel frequently, but of course much more than that when it is close and it has a lock on it.

The color also might have played a role. The orange color is not as bright and vibrant as a bright green or yellow for example. It might be worthwhile to use larger and brighter balls than a small orange ball.

The camera itself was mounted on a small wood base that was attached to the arm of a servo. The servo would tilt the camera forward for near objects and up for far objects. I did not think that made a lot of difference. Many times I left it in a neutral position and obtained similar results.

Behaviors

The robot has three behaviors: avoiding obstacles, searching for and tracking ping-pong balls, and finally acquiring them and shooting them.

To avoid obstacles the robot depends on two Sharp IR sensors and one ultrasonic sensor. Bump switches detect bumps on the back. Once an obstacle is found, the robot turns to the opposite direction. If an obstacle is detected in front, then the robot makes a random turn.

The search behavior starts with the ultrasonic sensor, as it seemed to detect objects faster than the camera. Once the object is detected, the camera is used to determine if it is a ball or an obstacle. If it were a ball, then the robot starts the track color sequence and approaches the ball by correcting its path. The robot is capable of scanning an angle to see if missed the ball on approach. The fact that the robot would lose the ball at small distances made the task of acquiring the ball very hard. The size of the ball compared to the cylinder is very close and hence the docking mechanism is somewhat more demanding than if the acquiring mechanism was much larger than the ball. I learned that in then design of robots one must give a lot of thought about how to make margins of error more forgiving. If the mechanism that acquires a ball is much larger than the ball, then any errors in the robot docking next to the ball would diminish and the collection of the ball would be much easier.

The collection behavior is simple enough if the previous search was successful. The robot goes into this mode one a ball is right in front of the robot. Then it lowers a cylinder and opens its door. Once the ball is in front of the opening of the cylinder, a door closes down and pushes the ball inside the cylinder. Then the cylinder is raised vertically, which causes the ball to fall in place to be ready to by shot by the flexible plastic shooter.

Experimental Layout and Results

Since my sensors were simple enough a little testing was enough to determine how to use them. Attached above are graphs for the infrared and ultrasonic sensors output. Based on this data and some experiments when the sensors were mounted on the robot, I was able to determine the suitable threshold values to use. The CMU camera testing consisted of determining what values constitute an orange color and the margin of error that existed. I did increase the margin of error for the CMU camera but I can’ t say the results were much better.

Conclusion

Gossima is a robot that can search for, find, capture, load, and retrieve ping-pong balls. It carries out its activities by using a microcontroller board to orchestrate its functions. Gossima is capable of avoiding obstacles, detecting objects and determining if these objects are orange ping-pong balls or just obstacles. Although the robot found it hard to determine if the color in front of it was orange or not especially at very close distances, it was able to find, track and approach a ball. Frequently, it lost tracking when it was very close to the ball. Some of the reasons for that are the speed of approach, size of the ball, and the color of a ball. If the approach was successful, acquiring the ball and shooting it are very simple procedures. Important lessons to be learned are to build the platform early on and preferably out of metal or some sturdy material; wood is not sturdy enough. Also, when designing the robot large margins of error should be employed like using a wide structure to acquire a small ball; that way, the robot has a very good chance in succeeding.

Documentation

References:

CMUcam Vision Board User Manual. Anthony Rowe and Carnegie Mellon University.
http://www.seattlerobotics.com/cmucam.htm

Pridgen Vermeer Robotics ATmega128. Revision 0

Sharp General Purpose Type Distance Measuring Sensors: GP2Y0A21YK

Ultrasonic Sensor LV-MaxSonar®-EZ2

Servos: HS-322HD Standard Deluxe

Thanks to:

Mike Pridgen
Thomas Vermeer
Jared Bevis
Joshua Childs
Joe Bari
John Kurien
Fellow students in IMDL
Kyle Tripician

Appendices

Code is attached in the appendix.

GossimaRobot.c

LCD.h
sleep.h
USART.h
PVR_Servos.h
ADC.h
cmuCamera.h
globalDefinitions.h

```
//
//
//                          File Name: GossimaRobot.c
//
//
//
//
//
/*****************************************
*
*                          INCLUDE FILES
*
*****************************************/
#include "LCD.h"
#include "sleep.h"
#include "USART.h"
#include "PVR_Servos.h"
#include "ADC.h"
#include "cmuCamera.h"
#include "globalDefinitions.h"

/*****************************************
*
*                          DEFINES
*
*****************************************/
#define FALSE               0
```

```
#define TRUE            1
#define LEFT            0
#define RIGHT           1
#define FORWARD             0
#define BACKWARD    1
#define CYLINDER    0
#define SCOOP           1

/*****************************************
*
*                                   PROTOTYPES
*
*****************************************/
void init_modules();
void init_camera();
void setServoPositions();
void GossimaCheckObstacles();
void GossimaCheckUltrasonic();

void SendCMUcommand();

void GossimaStop();
void GossimaMove(int direction);   // moves forward or backward
void GossimaTurnRandomly();
void GossimaTurn(int direction);    // turns left or right

void GossimaScanCircle();
void GossimaScanAngle();
void GossimaSearchRandomly();
void GossimaApproachBall();
void GossimaCaptureBall();
void GossimaFindBin();
void GossimaShootBall();


void GossimaSetSpeed(int speed);  // sets the wanted speed
void servoSlowMove(int last_pos, int new_pos, int servoName);          // moves cylinder from last position
to new position

void GossimaLowerScoop();
void GossimaRaiseScoop();
void GossimaLowerCylinder();
void GossimaRaiseCylinder();
void GossimaDockCylinder();


void GossimaTrackColor();
void setCameraTilt(int level);



void test_ultrasonic() {

        //lcdClear();
        //lcdString("A/D Demo");
        //lcdGoto(1,0);
```

```
	//lcdString("Channel 1: ");

	int x= 4000;                                    // take 1000 samples
	while(x>0){
		lcdClear();
		lcdString("Channel 2: ");
		lcdGoto(1,0);                           // start at row 1 column 0 (i.e. second line)
		lcdInt( adcTwo() );                     // display the value returned
		ms_sleep(2000);

		//lcdClear();
		//lcdString("looking left: ");
		//lcdGoto(1,0);
		//lcdInt( adcOne() );
		//ms_sleep(2000);

		x--;
		//ms_sleep(1000);
	}
}

/*******************************************
*
*                       GLOBAL VARIABLES
*
*******************************************/

/*
*
* irdr: IR detector located on the right side of robot looking to the right.
* irdl: IR detector located on the left side of robot looking to the left.
* speedr: speed of the right servo.
* speedl: speed of the left servo.
*
* Left wheel servo:             Connected to servo 1 connector.
*                                       +MAX_SPEED is backward, -MAX_SPEED is
forward.
*
* Right wheel servo:    Connected to servo 2 connector.
*                                       -MAX_SPEED is backward, +MAX_SPEED is
forward.
*
* Camera tilt servo:    Connected to servo 3 connector.
*                                       Range of movement; from -100 (looking straight) to -
40 (looking down)
*
* Cylinder servo  :     Connected to servo 4 connector.
*                                       Range of movement: from -100 (lowered) to 50
(raised)
*
* Scoop servo           :       Connected to servo 5 connector.
*                                       Range of movement: from -120 (lowered) to 50
(raised)
*
*
* Shooter servo         :       Connected to servo 6 connector.
```

```
*                                                   The servo stops by giving it a value of -48

*                                                   It seems that the zero point has shifted a lot.
*
*
*
* Right IR sensor connected to:             ADC channel 0
* Left IR sensor connected to:              ADC channel 1
* Ultrasonic sensor connected to:    ADC channel 2
*
*
*/

int irdr = 0;                                   // value of right IR sensor
int irdl = 0;                                   // value of left IR sensor
int speedr = 0;                                 // speed of right servo
int speedl = 0;                                 // speed of left servo
int current_speed = 100;    // contains the current speed of robot

int sonarRange = 0;                             // value of ultrasonic sensor
int lastSonarRange = 0;             // previous value of ultrasonic sensor

int objectDetected = FALSE;         // true of false
int ballDetected = FALSE;// true of false
int ColorDetected = FALSE;          // true of false
int ballTracked = FALSE; // true of false

int cylinderLastPos = 0;    //contains cylinders last position
int cylinderNewPos = 0;             //contains cylinders new position
int scoopLastPos = 0;               //contains scoop last position
int scoopNewPos = 0;                //contains scoop new position


u8 mmx, mmy, lcx, lcy, rcx, rcy, pix, conf, packetName;


/*********************************************
*
*                                           Main program
*
*********************************************/
int main(void) {

//////////////////   VARIABLE DECLARATIONS   ///////////////////////



//////////////////   INITIALIZATIONS   ////////////////////////////
        init_modules();


/*****************************************************************
*
*                   Temporarily: Left wheel is connected to servo 4
*                                           Right wheel is connected to servo 5
```

```
*               camera is connected to servo 6
*
************************************************************/

        setServoPositions();


GossimaLowerCylinder();
GossimaRaiseScoop();


GossimaLowerScoop();

GossimaRaiseCylinder();

        init_camera();

        ms_sleep(6000);


//////////////////   MAIN LOOP   ////////////////////////


        GossimaScanCircle();


                        GossimaStop();
                        ms_sleep(2000);

        while(1) {
                GossimaCheckObstacles();

                if(objectDetected == TRUE) {

                        //GossimaStop();
                        //ms_sleep(2000);
                        GossimaTrackColor();
                        if(ColorDetected == TRUE) {
                        // then object is an orange ball, so approach ball
                                ballTracked = TRUE;

                                GossimaApproachBall();
                                GossimaCaptureBall();
                                GossimaFindBin();
                                GossimaShootBall();

                        }
                        else {
                        // it's an obstacle, so avoid
                                //GossimaCheckObstacles();
                        }

                }
                else {
```

```
                                        lcdClear();
                                        lcdString("search randomly");
                                        ms_sleep(2000);
                                        lcdClear();

                                GossimaSearchRandomly();
                        }

        }// while loop

        return 0;

}//end main


/*********************************************
*
*               Check if object color is what we are looking for
*
*********************************************/
void GossimaTrackColor() {
        int count = 0;

        SendCMUcommand();

        // display an M packet
        //cmuDisplay_M_packet();

        if(conf < 30) {
                ColorDetected = FALSE;

        }//if
        if(conf > 30) {
                ColorDetected = TRUE;
        }//if

        // scan to the left first, if ball found then approach
        // if ball is not found, then scan to the right
        // if ball is found to the right, then approach
        // else if ball is not found to the right, then search randomly

        for(int j = LEFT; j < RIGHT + 1; j++) {
                while(ColorDetected == FALSE && count < 2) {

                        GossimaTurn(j);
                        ms_sleep(250);
                        GossimaStop();

                        for(int i = 0; i < 2; i++) {
                        // for two levels of camera tilt, scan to the left

                                setCameraTilt(i);
                                SendCMUcommand();
                                ms_sleep(250);
                                // display an M packet
                                //cmuDisplay_M_packet();
```

```
                                        if(conf < 20) {
                                                ColorDetected = FALSE;
                                        }//if

                                        if(conf >= 20) {

                                                ColorDetected = TRUE;

                                                break;
                                        }//if
                                }//for

                        count++;

                        }//while

                        count = 0;
                        if(j == 0 && ColorDetected == FALSE) {
                                GossimaTurn(RIGHT);
                                ms_sleep(400);
                                GossimaStop();
                        }

                        if(ColorDetected == TRUE) {
                                break;
                        }

                }//for

}// end

/*********************************************
*
*                       Set camera tilt level
*
*********************************************/
void setCameraTilt(int level) {
        switch(level) {
                        case 0: moveServo6(-60);
                                        break;

                        //case 1: moveServo6(-80);
                        //                      break;

                        case 1: moveServo6(-95);
                                        break;

                        default:  moveServo6(-95);
        }
}


/*********************************************
*
*                       Approach balls
*
```

```
**********************************************/
void GossimaApproachBall() {

		SendCMUcommand();
//		cmuDisplay_M_packet();

//		GossimaCheckUltrasonic();
//		ms_sleep(3);

	while(pix >=1) {

		SendCMUcommand();


		if(conf < 30) {
			GossimaSetSpeed(MED_SPEED);
			GossimaMove(FORWARD);
	}

			if(conf > 30) {
				GossimaSetSpeed(SLO_SPEED);
				GossimaMove(FORWARD);

			if(mmx < 30) {
				GossimaTurn(LEFT);
				ms_sleep(150);
				GossimaMove(FORWARD);
			}
			else if(mmx > 50){
				GossimaTurn(RIGHT);
				ms_sleep(150);
				GossimaMove(FORWARD);
			}
			else {
				GossimaMove(FORWARD);
			}
		}//if
		if(conf > 50 && mmy < 30 && pix > 50){
			while(1) {
				GossimaStop();
				lcdClear();
				lcdString("done!");
				ms_sleep(1000);
				lcdClear();
				ms_sleep(1000);
			}//while
		}//if
		GossimaCheckUltrasonic();
		ms_sleep(3);


	}// while

}// end
```

```
/*********************************************
*
*                   Scan for balls in a small angle
*
*********************************************/
void GossimaScanAngle() {

        // scan an angle for color

////////////////////////////////
        // scan to the left first
        GossimaSetSpeed(SLO_SPEED);
        GossimaTurn(LEFT);

        for(int i = 0; i < 500; i++) {
                GossimaCheckUltrasonic();
                ms_sleep(2);

                if(sonarRange < 20 ) {
                        // then the ball is re-aquired
                        ballTracked = TRUE;
                        //break;
                }
                else {
                        ballTracked = FALSE;
                }
        }// for


        GossimaStop();
        ms_sleep(3000);


        USARTstring("TC\r");
        ms_sleep(100);
        if(cmudat[9] > 50) {
                lcdClear();
                lcdString("angle color");
                ms_sleep(5000);
                lcdClear();
        }


////////////////////////////////
        if(ballTracked == FALSE) {
                // scan to the right if did not find a ball to the left
                GossimaTurn(RIGHT);

                for(int i = 0; i < 500; i++) {
                        GossimaCheckUltrasonic();
                        ms_sleep(2);

                        if(sonarRange < 20 ) {
                                // then the ball is re-aquired
                                ballTracked = TRUE;
```

```
                                        //break;
                                }
                                else {
                                        ballTracked = FALSE;
                                }
                        }// for

                        GossimaStop();
                        USARTstring("TC\r");
                        ms_sleep(100);

                }// if


////////////////////////////
                GossimaCheckUltrasonic();
                ms_sleep(3);

}//end

/*********************************************
*
*                       Search randomly for a ball
*
*********************************************/
void GossimaSearchRandomly() {

}//end

/*********************************************
*
*                       Capture a ball
*
*********************************************/
void GossimaCaptureBall() {
GossimaLowerCylinder();
GossimaRaiseScoop();
GossimaLowerScoop();
GossimaRaiseCylinder();

}


/*********************************************
*
*                       Find the bin
*
*********************************************/
void GossimaFindBin() {

}



/*********************************************
```

```
*
*                       Shoot a ball
*
*********************************************/
void GossimaShootBall() {

}


/*********************************************
*
*                       Make a circle while scanning for balls
*
*********************************************/
void GossimaScanCircle() {
        // scan in place using ultrasonic sensor by making a full turn
        // if detected an object less than 20, then stop

        GossimaTurn(LEFT);

        for(int i = 0; i < 1500; i++) {
                GossimaCheckUltrasonic();
                ms_sleep(2);

                if(sonarRange < 20 ) {
                        // then there is an object in front of robot
                        objectDetected = TRUE;
                        break;
                }
                else {
                        objectDetected = FALSE;
                }
        }//for

}// end

/*********************************************
*
*       Initialize Modules: LCD, USART, ADC, Servos
*
*********************************************/
void init_modules() {
        initSleep();
        lcdInit();
        lcdString("Starting System");
        ms_sleep(2000);
        lcdClear();
        USART_Init();
        //ms_sleep(200);
        initADC();
        //ms_sleep(200);
        initServo();
        ms_sleep(200);
        GossimaSetSpeed(MAX_SPEED);
}
```

```
/*********************************************
*
*                                  Initialize Camera
*
*********************************************/
void init_camera() {

        // Reset the camera with several RS commands
        for(int j=0; j<2; j++) {
                USARTstring("RS \r");
                lcdClear();
                ms_sleep(200);
                lcdGoto(0,0);
                lcdString("Resetting Camera");
                ms_sleep(500);
        }

        // Turn on auto tracking LED
        USARTstring("L1 1\r");
        ms_sleep(5000);

        // To apply a fluroscent band filter and auto lighting adjust
        // when working under fluroscent lighting, use the following command:
        USARTstring("CR 45 7 18 44\r");

        // For normal and incandescent lighting, just use the following command:
        //USARTstring("CR 18 44\r");

        // wait for 10 seconds to adjust to lighting conditions
        for(int j=0; j<10; j++) {
                ms_sleep(500);
        }

        // Turn off auto lighting adjust
        USARTstring("CR 18 40 19 32\r");

        // Turn off auto tracking LED (default mode)
        USARTstring("L1 2\r");

        // Indicate to user to prepare the target by flashing light and sending on LCD
        //FlashLight(5);
        //lcdClear();
        //lcdGoto(0,0);
        //lcdString("Hold target");
        //ms_sleep(3000);

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

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

```
	// display the command being sent
	lcdClear();
	lcdGoto(0,0);
	lcdString("Sending TW cmnd");
	ms_sleep(4000);

	// issue Track Window command
	USARTstring("TW\r");
	ms_sleep(300);

	lcdClear();
	lcdString("remove");
	ms_sleep(4000);


	// display an M packet
	//cmuDisplay_M_packet();


	lcdClear();
	lcdString("Get ready!");
	ms_sleep(2000);

}

/*********************************************
*
*			Avoid Obstacles using IR sensors
*
*********************************************/
void GossimaCheckObstacles() {

	// Beams of IR sensors cross each other.
	// irdr: IR detector located on the left side of robot looking to the right side
	// irdl: IR detector located on the right side of robot looking to the left side
	// speedr: speed of the right motor
	// speedl: speed of the left motor
	//
	// Left servo: Connected to servo 1 connector
	//                                      +MAX_SPEED is backward, -MAX_SPEED is forward
	//
	// Right servo:     Connected to servo 2 connector
	//                                       +MAX_SPEED is forward, -MAX_SPEED is backward

	// read IR sensors
	irdr = adcZero();
	irdl = adcOne();
	//FlashLight(1);
	// if there is an obstacle on the left side
	// then set variable to turn right
	if(irdl > AVOID_THRESHOLD)
	{
		//speedr = -MAX_SPEED;
		speedr = -current_speed;
		moveServo2(speedr);			// move right servo backward
		ms_sleep(1000);				// can make this random time
```

```
	}
	else
	{
		//speedr = MAX_SPEED;
		speedr = current_speed;
		moveServo2(speedr);		// move right servo forward
	}

	// if there is an obstacle on the right side
	// then set variable to turn left
	if(irdr > AVOID_THRESHOLD)
	{
		//speedl = MAX_SPEED;
		speedl = current_speed;
		moveServo1(speedl);		// move left servo backward
		ms_sleep(1000);
	}
	else
	{
		//speedl = -MAX_SPEED;
		speedl = -current_speed;
		moveServo1(speedl);		// move left servo forward
	}

	// check in front
	if (adcTwo() < 4) {
		GossimaTurnRandomly();
		ms_sleep(500);
	}


	// write code to test if it's stuck!!

}// end

/*********************************************
*
*		Find Range Using Ultrasonic Module
*
*********************************************/
void GossimaCheckUltrasonic() {
	// take the average of 10 readings

	sonarRange = adcTwo();
}// end

/*********************************************
*
*		Sets the wanted speed
*
*********************************************/
void GossimaSetSpeed(int speed) {
	current_speed = speed;
}
```

```
/*********************************************
*
*                   Turn in a random direction
*
*      By Ivan Zapate, 1996
*********************************************/
void GossimaTurnRandomly() {
        unsigned rand;
        rand = TCNT0;
        if(rand & 0x0001) {
                // turn right
                GossimaTurn(RIGHT);
        }
        else {
        // turn left
                GossimaTurn(LEFT);
        }
}

/*********************************************
*
*                   Turn in a given direction
*
*********************************************/
void GossimaTurn(int direction) {
        if(direction == LEFT) {
        // turn left
                //speedr = MAX_SPEED;
                speedr = current_speed;
//              moveServo2(speedr);             // move right servo forward
                moveServo5(speedr);             // move right servo forward


                //speedl = MAX_SPEED;
                speedl = current_speed;
//              moveServo1(speedl);             // move left servo backward
                moveServo4(speedl);             // move left servo forward


        }
        if(direction == RIGHT) {
                // turn right
                //speedr = -MAX_SPEED;
                speedr = -current_speed;
//              moveServo2(speedr);             // move right servo backward
                moveServo5(speedr);             // move left servo forward

                //speedl = -MAX_SPEED;
                speedl = -current_speed;
//              moveServo1(speedl);             // move left servo forward
                moveServo4(speedl);             // move left servo forward


        }
}//end
```

```
/*********************************************
*
*                   Move forward or backward
*
*********************************************/
void GossimaMove(int direction) {
        if(direction == FORWARD) {
                // move forward
                //speedl = -MAX_SPEED;
                speedl = -current_speed;
//              moveServo1(speedl);             // move left servo forward
                moveServo4(speedl);             // move left servo forward
                //speedr = MAX_SPEED;
                speedr = current_speed;
//              moveServo2(speedr);             // move right servo forward
                moveServo5(speedr);             // move left servo forward


        }
        else if(direction == BACKWARD) {
                // move backward

                //speedl = MAX_SPEED;
                speedl = current_speed;
//              moveServo1(speedl);             // move left servo backward
                moveServo4(speedl);             // move left servo forward

                //speedr = -MAX_SPEED;
                speedr = -current_speed;
//              moveServo2(speedr);             // move right servo backward
                moveServo5(speedr);             // move left servo forward


        }
        else {
                // do nothing
        }

}// end

/*********************************************
*
*                   Stop
*
*********************************************/
void GossimaStop() {

        // Be careful setting current speed to ZERO_SPEED!!!!
        // If you forget to reassign it, the robot will stand still!!!!
        // Use moveServo5(0)       , it's better.
        //current_speed = ZERO_SPEED;
        //speedr = current_speed;
        //moveServo5(speedr);

        moveServo5(0);
```

```
	moveServo4(0);
}

/*********************************************
*
*		Slowly moves servo from last position to new position
*
*
*********************************************/
void servoSlowMove(int last_pos, int new_pos, int servoName) {

	if(servoName == CYLINDER) {

		if(new_pos > last_pos) {
			for(int i = last_pos; i < new_pos + 1; i++) {
				moveServo1(i);
				ms_sleep(30);
			}
		}
		if(new_pos < last_pos) {
			for(int i = last_pos; i > new_pos - 1; i--) {
				moveServo1(i);
				ms_sleep(30);
			}
		}

		cylinderLastPos = new_pos;
	}// if
	if(servoName == SCOOP) {
		if(new_pos > last_pos) {
			for(int i = last_pos; i < new_pos + 1; i++) {
				moveServo2(i);
				ms_sleep(30);
			}
		}

		if(new_pos < last_pos) {
			for(int i = last_pos; i > new_pos - 1; i--) {
				moveServo2(i);
				ms_sleep(30);
			}
		}

		scoopLastPos = new_pos;
	}// if
}//end
/*********************************************
*
*		Lower Scoop servo
*
*********************************************/
void GossimaLowerScoop() {
	servoSlowMove(scoopLastPos, -120, SCOOP);
}
```

```
/*********************************************
*
*                   Raise Scoop servo
*
*********************************************/
void GossimaRaiseScoop() {
        servoSlowMove(scoopLastPos, 70, SCOOP);
}
/*********************************************
*
*                   Lower Cylinder servo
*
*********************************************/
void GossimaLowerCylinder() {

        servoSlowMove(cylinderLastPos, -100, CYLINDER);
}

/*********************************************
*
*                   Raise cylinder servo
*
*********************************************/
void GossimaRaiseCylinder() {
        servoSlowMove(cylinderLastPos, 50, CYLINDER);
}

/*********************************************
*
*                   Dock cylinder servo
*
*********************************************/
void GossimaDockCylinder() {
        servoSlowMove(cylinderLastPos, 110, CYLINDER);
}

/*********************************************
*
*                   set starting servo positions
*
*********************************************/
void setServoPositions() {
        moveServo1(-120);
        moveServo2(70);
        moveServo3(-48);

        moveServo4(0);
        moveServo5(0);
        moveServo6(-90);
}
```

```
/*********************************************
*
*                   Send Commands to CMU Camera
*
*********************************************/
void SendCMUcommand() {
        // Send "Track Color" command
        // Camera will track last color grabbed

        // check if color is in front of camera
        USARTstring("TC\r");
        ms_sleep(100);

        // save camera values
        //packetName = cmudat[1];                   // 'S' = 83, 'M' = 77, 'C' = 67
        mmx = cmudat[2];
        mmy = cmudat[3];
        pix = cmudat[8];
        conf = cmudat[9];

}//SendCMUcommand()



************************************************************
#include "cmuCamera.h"

#include "sleep.h"
#include "USART.h"
#include "LCD.h"



// Display an M packet
void cmuDisplay_M_packet() {

        u8 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];
        //ms_sleep(2000);

/*
        lcdClear();
        lcdGoto(0,0);
        lcdString("Packet Name");
        lcdGoto(1,0);
        if(cmudat[1] == 83) {
```

```
                lcdString("Packet S");
        }
        else if(cmudat[1] == 77) {
                lcdString("Packet M");
        }
        else if(cmudat[1] == 67) {
                lcdString("Packet C");
        }
        else {
                lcdString("Packet ERROR!");
        }
        ms_sleep(3000);

*/
        lcdClear();
        lcdGoto(0,0);
        lcdString("Middle Mass X");
        lcdGoto(1,0);
        lcdInt(mmx);
        ms_sleep(2000);

        lcdClear();
        lcdGoto(0,0);
        lcdString("Middle Mass Y");
        lcdGoto(1,0);
        lcdInt(mmy);
        ms_sleep(2000);
/*
        lcdClear();
        lcdGoto(0,0);
        lcdString("Left Corner X");
        lcdGoto(1,0);
        lcdInt(lcx);
        ms_sleep(3000);

        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(2000);

	lcdClear();
	lcdGoto(0,0);
	lcdString("Confidence");
	lcdGoto(1,0);
	lcdInt(conf);
	ms_sleep(2000);
}



// Display an S packet
void cmuDisplay_S_packet() {

	u8 rMean, gMean, bMean, rDev, gDev, bDev, packetName;

	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];

	ms_sleep(2000);

	// Display the packet vlaues
	lcdClear();
	lcdGoto(0,0);
	lcdString("Packet Name");
	lcdGoto(1,0);
	if(cmudat[1] == 83) {
		lcdString("Packet S");
	}
	else if(cmudat[1] == 77) {
		lcdString("Packet M");
	}
	else {
		lcdString("Packet ERROR!");
	}
	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);
}
```

****