Wallmaster

Final Report

Terence Tai
Class: EEL 4665/5666
Instructors: Dr. Arroyo, Dr. Schwartz
TAs: Devin Hughes, Ryan Stevens,
Josh Weaver, Sean Frucht, Tim Martin

Table of Contents

Abstract

Wallmaster is a robot that will build a wall using Jenga blocks. When activated it will obstacle avoid until it finds a line to follow marked by black tape. It attempt to stay on the line by moving straight. If it can it will build a wall along the line, otherwise it will attempt to adjust itself. Once it has built a wall along the line or is unable to adjust itself along the line, it will resume obstacle avoiding and repeat this process when it finds another line.

Executive Summary

Wallmaster's primary goal is build a wall using Jenga blocks. The body of the robot is rectangular to facilitate manipulation and placement of blocks. The platform powered by six AA batteries and controlled by the Pridgen Vermeer Robotics board using the xmega128A1. At the beginning of the semester I intended Wallmaster to not only build a wall, but also be able to retrieve blocks and keep track of wall integrity. These original objectives were overly ambitious and a unrealistic, especially since this was the first time I built a robot from scratch. So instead I revised the objectives to simply obstacle avoid until it finds a black line, follow the line and build a wall using blocks preloaded on the platform, return to obstacle avoiding and repeat operations until it runs out of blocks.

Wallmaster will achieve this by having special mechanism that will actuate and unload blocks to the appropriate level. This special mechanism will use 3 servos, multiple rack and pinion systems, and a nut and bolt system. In addition the special mechanism will have its own set of limit switches, IR sensors, and CDS cells to keep track of block positioning.

For mobility Wallmaster uses two hacked servos to drive the wheels and uses two IR sensors, single sonar sensor, and an array of CDS cells and LEDs for obstacle avoidance and line following. Wallmaster will initially use the two IR sensors and sonar to obstacle avoid while using trying to a line using the CDS cells. Once a line has been found, Wallmaster will move forward a bit on the line and then actuate a block and put it down to the appropriate height. Once Wallmaster has stacked blocks to the maximum allowable height it will move further down the line and repeat the process until it either runs out of blocks or is off the line.

Introduction

Robots to me represent an active implementation of a computer, having a processer/microcontroller manipulate data to drive motors and other forms of actuation, rather than simply storing data to be displayed on screen. Although this is my first time constructing a robot, I wanted a robot that had some mechanical complexity to demonstrate its ability to actively manipulate the environment and to provide a challenge in an area that I have not explored as a computer engineer. I decided to create Wallmaster, a wall building robot.

Wallmaster's primary objective is to build a multi-level wall using blocks in a formation that is commonly seen in most buildings. In order to do so Wallmaster must have a mechanism to store the blocks and move them to the appropriate location. Problems that Wallmaster will solve

include, identifying where to build the wall, manipulating the building blocks to the right orientation and height to be placed on the wall and knowing when it will have to refill on blocks.

Integrated System

The integrated system consists of:

- One PVR xmega 128 microcontroller board, which currently controls and powers all the sensors necessary for obstacle avoidance and actuation
- 6 AA batteries to power they system
- One Sonar Range finder sensor and two IR range finder sensors used to detect obstacles.
- 5 Hi-Tec HS-311 /HS-322HD servos. Two of these servos are hacked for basic platform actuation. Another two are hacked to actuate the block out of the block holding clip and to move the elevator up and down. The third will remain un-hacked to operate the block kick mechanism.
- A line following sensor consisting of three CDS cells and four LEDs detect black tape on a bright sheet of paper. The line following sensor is directly interfaced to the systems ADC channels.
- A beam break sensor consisting of an LED and a CDS cell. Will be interfaced on the ADC on the PVR board.
- One short range IR sensor to detect what level the elevator is at in relation with the wall.
- Three limit switches and one bump switch to determine the position of the actuation system in relation with the platform.

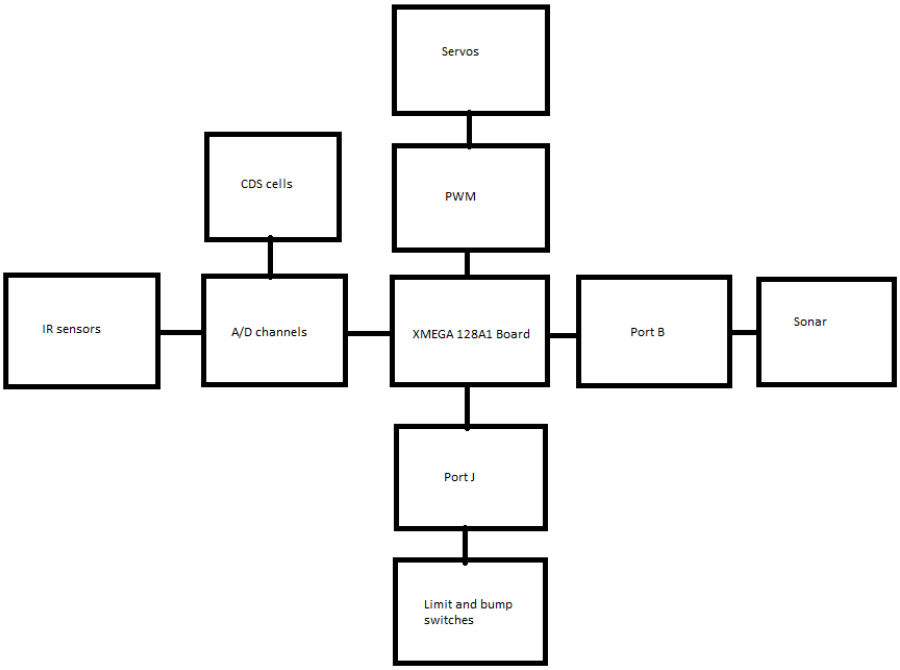


Figure 1: Integrated system diagram

Mobile Platform

The mobile platform I am currently using for Wallmaster is an 8 ½ by 14 inch rectangle. The platform is single level with housing for the board, battery and drive motors and 8 screw holes for mounting wheeled castors. A rectangular shape was chosen such that Wallmaster can better line up with the wall building area and to facilitate in moving blocks around on the platform. The platform is currently using 4 wheels, two powered by servos and two others attached to castors. The wheels and castors are placed at the front and rear ends of the platform respectively to promote movement in a straight line.

In addition the platform has a detachable elevator and block storage compartments. The block storage compartment has one servo mount to operate its rack and pinion system and 4 holes that mount directly to the screws of one of the castors. The elevator compartment is composed of two dowel rods and a servo mount on the main platform, and a single servo mount on the elevator block. Both compartments have grooves that each holds a single rack to be manipulated by servos.

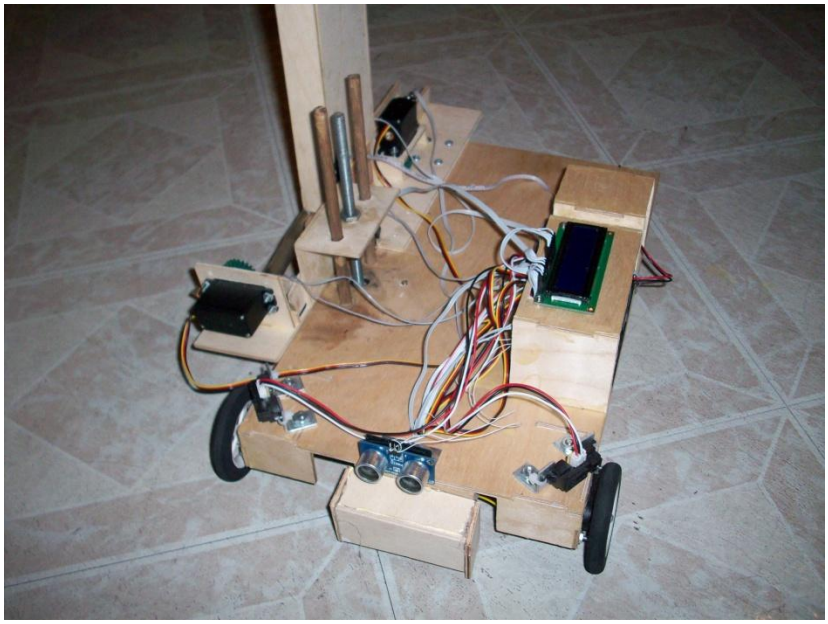


Figure 2: The Wallmaster Platform

Actuation

Wallmaster moves around using 2 two hacked Hi-Tec HS-311 servos mounted on the bottom of the platform attached to 2.5 inch diameter wheels in the front and has two caster wheels in the back for stability. The servos have an operating speed of 60 degrees per .19 seconds (which is estimated to be about 52 revolutions per minute). The servos are directly controlled using the PWM module on the Xmega 128.

Special Design

The special design that I have designed for Wallmaster is the wall building mechanism. Inspiration behind this mechanism is based on how to simplify the process of wall construction. Initially I thought of using a mechanical arm as a means of picking up and placing blocks, but using arm might prove difficult in terms of figuring out if the block is picked up and placed in the correct orientation. With this special design I hope to simplify the operation of orienting and placing blocks.

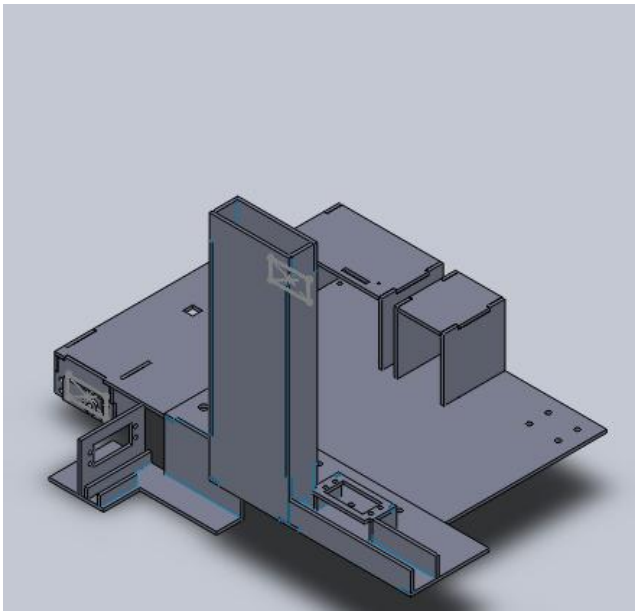


Figure 2: Solidworks of platform

The special design consists of a clip that stores stack of Jenga blocks 15 blocks high. The clip has a small hole on the bottom allows lowest block to move freely. On one side of the hole there is a rack and pinion system that pushes a block out of the hole onto an elevator platform. By pushing the block out of the clip the block arrives on the elevator platform already properly oriented and ready to be placed. The elevator will move the block up and down and push the block off to the correct position.

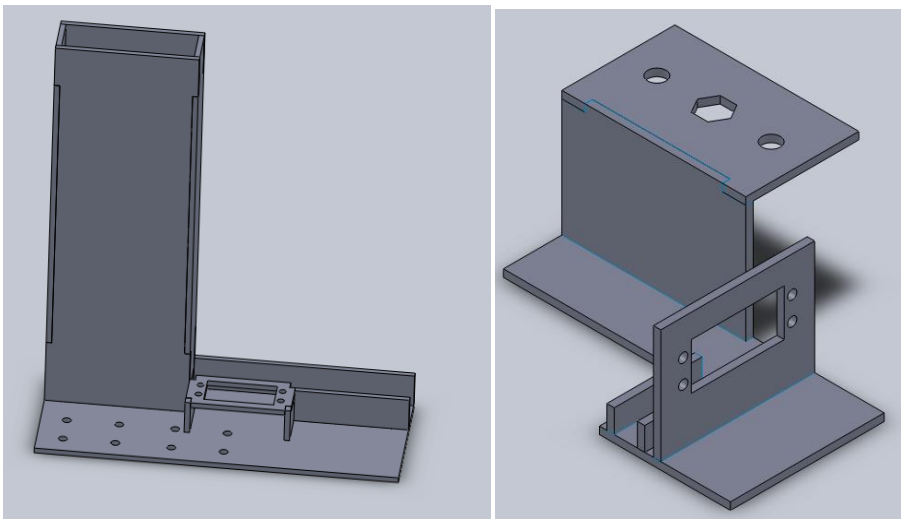


Figure 3: Block container mechanism on the left, elevator system on the right

The rack and pinion is constructed using a VEX robotics gear set that I have purchased from the Robotic Marketplace. The rack is controlled by a pinion directly attached to a servo. The servo rotates, pushing the rack forwards or backwards, allowing it to push a block out of the clip.

The elevator system consists of two dowel rods, a long threaded screw and two servos. The dowel rods are directly attached to the platform serving as rail guides for the elevator shaft. The elevator platform has three holes. Two holes are used to constrain the elevator dowel rods, forcing up and down movement while the third hole has nut glued/wedged in that will screw into the long threaded screw. The threaded screw is attached to the servo, which rotates the screw and moves the elevator platform up and down. A second servo is attached to the elevator platform and moves the block off the platform onto the appropriate place on the wall.

The block building mechanism in the final product worked consistently, but albeit rather slowly, especially when moving the blocks up and down using the elevator. This is primarily due to the high thread count of the screw, and the high gear down ratio of the servo. I intended to install a faster servo, but due to time constraints had to stick with the standard hi-tec servos.

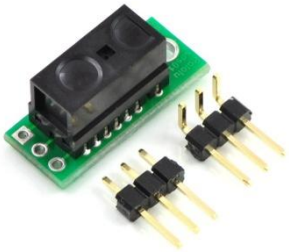
Sensors

- Parallax PING Sonar Range Finder:

Wallmaster currently uses a single Parallax PING sonar sensor to detect obstacles in front of it. The Parallax PING is a digital sonar sensor which takes in a 5 microsecond pulse and transmits back a pulse whose length is based off of the delay between the obstacle and the sensor. This operation is executed through Port B on the PVR board.

- Sharp GP2Y0A21YK0F IR Analog Distance Sensor:

Wallmaster uses two Sharp IR sensors to detect obstacles on the sides to determine when the robot should turn. The IR sensors output a specific voltage based on the current distance which is read by the ADC on the PVR board. The sensors have a much shorter range than the sonar range finder and its maximum range seems to fluctuate based off of lighting conditions. It could also be due to the fact that I am using 3.3V to power the IR sensors when they are clearly designed for 5V operation.



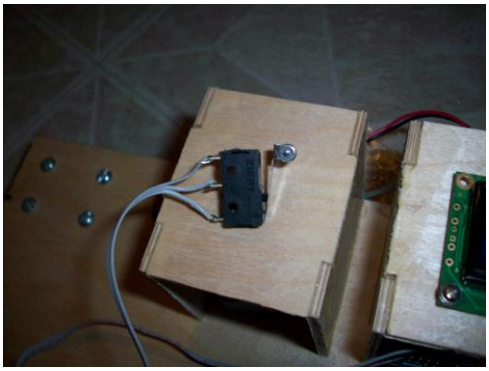
- Sharp GP2Y0D805Z0F Digital Distance Sensor:

Wallmaster uses a single Sharp Digital Distance IR sensor to detect whether it has seen a block underneath the elevator. While I have this sensor connected to the A/D port on the PVR board, the sensor does not vary based off of current distance. Instead the A/D value saturates at 4095 and then dips to about 1000 when it sees the object in range.

- Assorted CDS cells:

Wallmaster uses 4 CDS cells. Three of these cells are used to operate the line following sensor. Depending on lighting conditions the sensor will saturate the A/D at 4095 when it sees a white and dips by about 200 when it sees a black line. The last remaining CDS cell is used as a "break beam" to check whether there are blocks in the clip.

- Assorted Bump and Limit switches:



Wallmaster uses 3 limit switches and one bump switch to assist in the actuation of the wall building mechanism. Two of limit switches are used to determine whether the elevator has reached the top or bottom of the platform. The third limit switch is used determine whether the rack is in the default position for the block storage mechanism. The bump switch is used to check whether the block has fully actuated on to the elevator. The value of the switches are directly sent to port J on the PVR board

Behaviors

Initially the behavior planned for the robot included was to detect blocks, retrieve blocks, and build a wall with it and check the wall integrity, but due to the time constraints given by this semester I believe it was necessary to revise the behavior of the robot. The goal of building a wall remains the same but the revised behavior is now:

- Avoid obstacles until Wallmaster encounters the wall construction area marked by the black tape.
- If Wallmaster finds a marked construction it starts building a wall along the black tape until there is no more tape or Wallmaster runs out of blocks
- If done building the wall obstacle avoid again until it finds another marked spot to build a wall.
- If Wallmaster runs out of blocks it will stop operation until someone reloads it with blocks and resets the system.

Experimental Layout and Results

Experiments that I have performed so are basic sensor tests for the sonar and infrared sensors and basic obstacle avoidance. Experimental layout for the sensor test is to place a ruler next to the sensor and start measuring distances based off of microsecond delay from the sonar and the ADC readings from the infrared sensor. The results of the sonar sensor test proved to be quite accurate and precise. The infrared sensors on the other hand had limited range of about 8 inches and not very accurate at longer range. Differences in lighting conditions reduced this range, which was especially present during media day when Wallmaster started turning due to higher A/D readings when it should have been moving straight.

The experimental layout for obstacle avoidance was to simply allow Wallmaster to navigate the obstacles on the floor. For the most part Wallmaster was successful at avoiding obstacles, but I have run into a few problems. Wallmaster was programmed to turn away from obstacles when it detected an object too close to the infrared sensor and to stop and turn around when it encountered an object too close to the sonar. The current platform design has the servos placed at the very front of the platform, which made the back of the platform have a wide turning angle and caused it to occasionally bump into things when reversing.

The line following sensor was tested using a white tri-fold board and electrical tape. Wallmaster was successfully able to follow the line, but I have experienced issues with it following the line straight. While this was not an issue during Demo and Media day, it did hinder Wallmaster's ability properly build a wall.

The wall building mechanism was initially tested by making a block stack of 3 on a piece of wood. The piece of wood is necessary since the elevator cannot go all the way down due the digital IR sensor being in the way. This would cause the block to not orient properly as it was kicked due to elevator not being totally level with the floor. Initially the wall building mechanism was inconsistent, with the blocks often falling out before reaching its destination. I managed to rectify this by adding an extra metal guide to the block kicking mechanism.

Conclusion

By the time demo and media day came, I was successfully able to build a wall with Wallmaster, though I still had minor issues that hindered performance. One of the biggest issues was having Wallmaster approaching the black line straight. Wallmaster would often attempt to follow a line at an angle partially due to the servo potentiometers moving away from the proper zero spot, and accidentally imposing an obstacle on Wallmaster before it detects the black line during testing. Apart from this issue, as long as Wallmaster was able to move straight on to the black line it was quite successful at building a wall.

If I were to start over, I would have better planned out my proposal and parts I needed earlier in the semester, as this would have reduced the need of stress of trying to find parts which are not normally available locally, such as a rack and pinion system. In addition I would also have opted for a smaller design with smaller blocks, since this current design is a little cumbersome to carry. Finally, I would have used ball casters instead of wheel casters, which might partially resolve the issue having Wallmaster moving in a straight line.

If I had additional time, I would attempt to create a more accurate line following sensor by using extra CDS cells, digital IR sensors or by simply purchasing an actual line following sensor module. In addition I would speed up the elevator system by using a faster servo or using a thicker threaded screw.

Overall, IMDL was quite a challenging course, where I not only had to apply some knowledge from previous classes, such as microprocessors and intro to C programming, but also learn new software that I had absolutely no experience using, such as SolidWorks. In addition this class has taught me to be more proactive when acquiring parts for the robot since most of

these parts will not be available locally. With this in mind IMDL encourages students to have a clear plan in terms of robot design, parts and materials needed to put it together, and necessary software in order to succeed.

Documentation

http://www.pololu.com/catalog/product/136 - IR analog sensor picture

http://www.pololu.com/catalog/product/1132 - IR digital sensor picture

http://www.parallax.com/Store/Sensors/ObjectDetection/tabid/176/CategoryID/51/List/0/SortField/0/Level/a/ProductID/92/Default.aspx - Ping Ultrasonic sensor picture

Appendices

Final Code:

```
#include <avr/io.h>
#include "PVR.h"
#include <stdio.h>
#include <util/delay.h>


int dir = 0;
int empty = 0;
char beam[10] = {0};
char level[10] = {0};
void actuateBlock() {
		int complete = 0;
		while (complete != 2) {
			lcdGoto(0,0);
			if(dir == 0) {
				if(PORTJ_IN & 1) {
					ServoD1(-2);
					dir = 1;
					delay_ms(1000);
					complete++;
				}
				else {
					ServoD1(-100);
				}
			}
			else {
				ServoD1(100);
				while(PORTJ_IN & 4);
```

```
                                dir = 0;
                                ServoD1(-2);
                                delay_ms(1000);
                        }
                }
                dir = 0;
}

void putBlockDown() {
        lcdInit();

        while(!(PORTJ_IN & 2)) {
                ServoC1(100);
        }
        ServoC1(-2);
        int breakBeam = 4000;
        while(!(breakBeam < 4000)) {
                breakBeam = ADCA6();
                lcdGoto(0,0);
                lcdString("Break Beam: ");
                sprintf(beam,"%d",breakBeam);
                lcdString(beam);
                ServoC1(-100);
        }
        lcdString(level);
        delay_ms(5000);
        ServoC1(-2);
        kick();
        while(!(PORTJ_IN & 2)) {
                ServoC1(100);
        }
        ServoC1(-2);
        unkick();
        lcdInit();
}

void kick() {
        ServoC2(100);
        delay_ms(1000);
        ServoC2(-70);
        delay_ms(1000);
}

void unkick() {
        ServoC2(-70);
        delay_ms(1000);
```

```
	ServoC2(100);
	delay_ms(1000);
}

void main(void)
{
	xmegaInit();                                    //setup XMega
	delayInit();                                    //setup delay functions
	ServoCInit();                                   //setup PORTC Servos
	ServoDInit();                                   //setup PORTD Servos
	ADCAInit();                                             //setup PORTA
analong readings
	lcdInit();                                              //setup LCD on
PORTK
	int i = 0;
	int j = 0;
	int reverse = 0;
	int digits = 0;
	int base = 1;
	char msdel[10] = {0};
	char C0adc[10] = {0};
	char C1adc[10] = {0};
	char C2adc[10] = {0};
	int ir0 = 0;
	int ir1 = 0;
	int turndir = 0;
	int turndecide = 0;
	int straightcount = 0;
	int mode = 0;
	int C0 = 0;
	int C1 = 0;
	int C2 = 0;
	int calibrate = 1;
	int C0AVG;
	int C1AVG;
	int C2AVG;
	int C0SUM;
	int C1SUM;
	int C2SUM;
	int tries;
	int on_line;
	int lvl2 = 0;
	PORTJ_DIR |= 0x0;
	while(ADCA5() < 2000) {
		C0SUM = 0;
		C1SUM = 0;
```

```
C2SUM = 0;
for(int k = 0; k < 20; k++) {
        C0 = ADCA2()/20;
        C1 = ADCA3()/20;
        C2 = ADCA4()/20;
        C0SUM += C0;
        C1SUM += C1;
        C2SUM += C2;
}
C0 = C0SUM;
C1 = C1SUM;
C2 = C2SUM;
if(calibrate) {
        C0AVG = C0;
        C1AVG = C1;
        C2AVG = C2;
        calibrate = 0;
}

ir0 = ADCA0();
ir1 = ADCA7();
PORTB_DIR |= 0x01;
PORTB_OUT = 1;
_delay_us(5);
PORTB_OUT = 0;
PORTB_DIR &= 0xFE;
i=0;
while(!(PORTB_IN & 1));
while(PORTB_IN & 1) {
        _delay_us(1);
        if(PORTB_IN & 1) {
                i++;
        }
}

if(mode == 0) {
        lcdGoto(0,0);
        lcdString("US Delay:    ");
        lcdGoto(1,0);
        lcdString("0");
        lcdGoto(1,0);
        sprintf(msdel,"%d",i);
        lcdString(msdel);
        base = 1;
        digits = 0;
        while(i/base) {
```

```
        digits++;
        base = base*10;
}
for(j = 0; j < (10-digits);j++) {
        lcdString(" ");
}
lcdString("     ");
if(i > 1500) {
        if(reverse) {
                ServoC0(0);
                ServoD0(0);
                _delay_ms(1000);
                reverse = 0;
                turndecide = i%2;
                if(turndecide) {
                        ServoD0(10);
                        ServoC0(10);
                }
                else {
                        ServoD0(-10);
                        ServoC0(-10);
                }
                _delay_ms(2000);
        }
        else {
                ServoD0(-10);
                ServoC0(10);
        }
}
else if(reverse) {
        ServoD0(10);
        ServoC0(-10);

}
else if(reverse == 0){
        reverse = 1;
        ServoC0(0);
        ServoD0(0);
        _delay_ms(1000);

}
if(ir0 >= 2548 && reverse == 0) {
        ServoD0(0);
        ServoC0(12);
        turndir = 1;
}
```

```
		if(ir1 >= 2548 && turndir == 0 && reverse == 0) {
			ServoD0(-12);
			ServoC0(0);
		}
		turndir = 0;
		if(C1 < C1AVG-100 || C0 < C0AVG-100 || C2 < C2AVG-100) {
			mode = 1;
			reverse = 0;
			tries = 6;
		}
	}
	if(mode == 1) {
		lcdGoto(0,0);
		lcdString("C1   C2   C3");
		lcdGoto(1,0);
		sprintf(C0adc,"%d",C0);
		lcdString(C0adc);
		base = 1;
		digits = 0;
		tries = 6;
		while(C0/base) {
			digits++;
			base = base*10;
		}
		for(j = 0; j < (5-digits);j++) {
			lcdString(" ");
		}

		sprintf(C1adc,"%d",C1);
		lcdString(C1adc);
		base = 1;
		digits = 0;
		while(C1/base) {
			digits++;
			base = base*10;
		}
		for(j = 0; j < (5-digits);j++) {
			lcdString(" ");
		}

		sprintf(C2adc,"%d",C2);
		lcdString(C2adc);
		base = 1;
		digits = 0;
		while(C2/base) {
			digits++;
```

```
            base = base*10;
        }
        for(j = 0; j < (5-digits);j++) {
            lcdString(" ");
        }

        if(C1 < C1AVG-100) {
            on_line = 1;
            if(straightcount) {
                ServoC0(0);
                ServoD0(0);
                int limit = 4000;
                while((limit > 1100) && (ADCA5() < 2000)) {
                    actuateBlock();
                    putBlockDown();
                    limit = ADCA6();
                }
            }
            ServoC0(0);
            ServoD0(0);
            _delay_ms(1000);
            ServoC0(8);
            ServoD0(-8);
            _delay_ms(800);
            straightcount++;
        }
        if (C0 < C0AVG-100 && C1 > C1AVG-100) {
            ServoC0(12);
            ServoD0(0);
        }
        if(C2 < C2AVG-100 && C1 > C1AVG-100) {
            ServoD0(-12);
            ServoC0(0);
        }
        if (C0 >= C0AVG-100 && C1 >= C1AVG-100 && C2 >= C2AVG-100)
{
            if(on_line && ADCA6() > 1100 && tries > 0) {
                straightcount = 0;
                ServoC0(-8);
                ServoD0(8);
                _delay_ms(1500);
                turndecide = i%2;
                if(turndecide) {
                    ServoD0(-8);
                    ServoC0(-8);
                }
```

```
                                        else {
                                                ServoD0(8);
                                                ServoC0(8);
                                        }
                                        _delay_ms(500);
                                        tries--;
                                }
                                else {
                                        mode = 0;
                                }
                                on_line = 0;

                        }
                }
        }
        ServoD0(0);
        ServoC0(0);
}
```