Sensors and Actuators

CDA4621 Lab 1

Andres Izurieta, Kiril Lange

February 8, 2016

Contents

[Introduction: 4](#_Toc442619657)

[Tasks 4](#_Toc442619658)

[IR Short Distance Sensor 4](#_Toc442619659)

[Conceptual Solution 4](#_Toc442619660)

[Code Description 4](#_Toc442619661)

[Images 4](#_Toc442619662)

[Video 4](#_Toc442619663)

[Conclusion 4](#_Toc442619664)

[IR Long Distance Sensor 4](#_Toc442619665)

[Conceptual Solution 4](#_Toc442619666)

[Code Description 4](#_Toc442619667)

[Images 4](#_Toc442619668)

[Video 4](#_Toc442619669)

[Conclusion 4](#_Toc442619670)

[Open Loop Control Motor 4](#_Toc442619671)

[Conceptual Solution 4](#_Toc442619672)

[Code Description 4](#_Toc442619673)

[Images 5](#_Toc442619674)

[Video 5](#_Toc442619675)

[Conclusion 5](#_Toc442619676)

[Square Open Loop Movement 5](#_Toc442619677)

[Conceptual Solution 5](#_Toc442619678)

[Code Description 5](#_Toc442619679)

[Images 5](#_Toc442619680)

[Video 5](#_Toc442619681)

[Conclusion 5](#_Toc442619682)

[“8” Open Loop Movement 5](#_Toc442619683)

[Conceptual Solution 5](#_Toc442619684)

[Code Description 5](#_Toc442619685)

[Images 5](#_Toc442619686)

[Video 5](#_Toc442619687)

[Conclusion 5](#_Toc442619688)

[Function Reference 5](#_Toc442619689)

[Sensors 5](#_Toc442619690)

[getLongDistance 5](#_Toc442619691)

[getShortDistance 6](#_Toc442619692)

[IRDistToString 6](#_Toc442619693)

[mergeDistance 7](#_Toc442619694)

[printIRDistance 7](#_Toc442619695)

[updateIRSensors 7](#_Toc442619696)

[Movement 8](#_Toc442619697)

[calibrate 8](#_Toc442619698)

[moveArc 9](#_Toc442619699)

[moveLine 10](#_Toc442619700)

[movePivot 11](#_Toc442619701)

[ServoFwdLookup 12](#_Toc442619702)

[ServoRevLookup 12](#_Toc442619703)

[Miscellaneous 12](#_Toc442619704)

[countDown 12](#_Toc442619705)

# Introduction

For this lab we were instructed to complete a series of tasks relating to gathering information from the IR sensors and moving our robot.

# Tasks

## IR Short & Long Range Distance Sensors

Using the three short rage IR sensors positioned on the left, right and front sides of the robot compute the distance to an object located on a respective side. When computing the center distance use the long range sensor as well.

### Conceptual Solution

Start

While true

Read Buttons

If exit button

Update IR Sensors

Print Distance

Delay .5s

Return

Clear LCD

Center Short Range Sensor

Center Long Range Sensor

### 

Right Sensor

LCD Screen Buttons

LCD Screen

Left Sensor

Arduino

### Code Description

The sensorFunction function performs all the nessasary steps to complete this task.

void sensorFunction(){

This Section of code clears the LCD and prepares it for writing the distance of subsequent sensor readings.

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Right|Centr|Left");

This section continues looping until the Left button is pressed returning it to the main menu.

while (1){

uint8\_t buttons = lcd.readButtons();

if (buttons) {

if (buttons & BUTTON\_LEFT) {

return;

}

}

The updateIRSensors and printIRDistance funtions are called to update and print the distance retrieved from all three sensors. These two functions are described near the end of the report.

updateIRSensors();

printIRDistance();

delay(500);

}

}

### Images

### Video

### Conclusion

Using graphs obtained from the manufacturer’s website, functions that describe voltage vs distance were calculated. The accuracy of these functions depend on ideal conditions, any variations from ideal and the sensor readings returned lose accuracy. There is no solution for the variability in the sensors themselves.

## Open Loop Control Motor

### Conceptual Solution

Forward and Backward Clockwise Circle Counterclockwise Circle

Delay 5s

Start

Return

Return

Start

Delay 5s

Start

Delay 5s

Turn counterclockwise 360 degrees around a 10 inch radius

Turn clockwise 360 degrees around a 10 inch radius

Move Forward 50 inches

Move Reverse 50 inches

Return

Arduino

Left Servo

Right Servo

### Code Description

The forwardReverse50 function acts a wrapper utilizing the moveLine function to travel for a given distance at a given rate. The countDown and moveLine functions are defined neat the end of this report.

void forwardReverse50(){

countDown(5, "F & R 50in");

moveLine(1270, 70);

moveLine(1270, -70);

}

The CWCircle20in and CCWCircle20in functions utilize the moveArc function. By specifying the radius, theta, x & y quadrant and rate the robot will move in a circular pattern. The moveArc function is defined at the end of this report.

void CWCircle20in(){

countDown(5, "CW Circle 20in");

moveArc(254, 360, 1, -1, 70);

}

void CCWCircle20in(){

countDown(5, "CCW Circle 20in");

moveArc(254, 360, 1, 1, 70);

}

### Images

### Video

### Conclusion

Major issues were encountered when attempting to control the servos accurately, due to the lack of feedback from the wheels (via an encoder). When the same values such as 1400, 1600 were fed into the servos, the robot could not be guaranteed to go straight. Manual adjustment in the code was necessary to have the robot go as required. This had to also be done for reverse and turning. Another issue came from the fact that battery charge affected the performance immensely. In fact the 5 cell rechargeable pack we were using has a nominal voltage of only 6 volts. NIMH AA batteries are ~1.2v each 5\*1.2 = 6 volts. The japan plug on the Arduino recommends 7v to ensure a steady 5v. A 6 cell AA battery pack or a 2S lipo would provide the necessary voltage ensure consistent results.

## Square Open Loop Movement

### Conceptual Solution

Clockwise Square Counterclockwise square

Pivot 90 degrees CClockwise

Move Forward 20 inches

Pivot 90 degrees CClockwise

Move Forward 20 inches

Pivot 90 degrees CClockwise

Delay 5s

Move forward 20 inches

Start

Return

Move Forward 20 inches

Move Forward 20 inches

Start

Delay 5s

Pivot 90 degrees Clockwise

Left Servo

### 

Move Forward 20 inches

Arduino

Pivot 90 degrees Clockwise

Right Servo

Move Forward 20 inches

Pivot 90 degrees Clockwise

Return

Move forward 20 inches

### Code Description

### Images

### Video

### Conclusion

## “8” Open Loop Movement

### Conceptual Solution

### Code Description

### Images

### Video

### Conclusion

# Function Reference

## Sensors

### getLongDistance

int getLongDistance(int voltage) {

if (voltage < 80 || voltage > 600)

return 0;

//calculate distance in mm from formula

double coefficient = 22392.0;

double power = 1.107;

double distance = coefficient / (pow((double)voltage, power)) \* 10;

if (distance < 0)

return 0;

return (int)distance;

}

### getShortDistance

//Short range sensor distance values in milimeters

int getShortDistance(int voltage) {

if (voltage < 50 || voltage > 600)

return 0;

//calculate distance in mm from formula

double distance = 0.0;

double coefficient = 3858.8;

double power = 1.074;

distance = coefficient / (pow((double)voltage, power)) \* 10;

if (distance < 0)

return 0;

return (int)distance;

}

### IRDistToString

//Converts the IR value to a printable string

void IRDistToString(String &StrDist, int distance){

if (distance < 1000){

StrDist = "0";

if (distance < 100){

StrDist = String(StrDist + "0");

if (distance < 10){

StrDist = String(StrDist + "0");

}

}

}

StrDist = String(StrDist + String(distance));

}

### mergeDistance

//Merges the short and long range distance sensor values

int mergeDistance(int shortDist, int longDist) {

if (longDist >= 300)

return longDist;

if (shortDist <= 200)

return shortDist;

if (longDist < 300 && longDist > 200 && shortDist < 300 && shortDist > 200){

double shortFactor = ((double)shortDist - 200.0) / 100.0;

double longFactor = ((double)longDist - 200.0) / 100.0;

double mergeFactor = (shortFactor + longFactor) / 2;

double distance = shortDist\*(1 - mergeFactor) + longDist\*mergeFactor;

return (int)distance;

}

else

{

if (longDist < 300 && longDist > 200)

return longDist;

if (shortDist < 300 && shortDist > 200)

return shortDist;

}

}

### printIRDistance

//Outputs the sensor data in a readable format

void printIRDistance() {

lcd.setCursor(0, 1);

String Right, Center, Left;

IRDistToString(Right, rightDistance);

IRDistToString(Center, centerDistance);

IRDistToString(Left, leftDistance);

String printDist = String("0" + Right + "|" + "0" + Center + "|" + Left);

lcd.print(printDist);

}

### updateIRSensors

//Gets fresh data from all istance sensors

void updateIRSensors() {

leftDistance = getShortDistance(analogRead(SLSensor));

rightDistance = getShortDistance(analogRead(SRSensor));

int tempShort = getShortDistance(analogRead(SFSensor));

int tempLong = getLongDistance(analogRead(LFSensor));

centerDistance = mergeDistance(tempShort, tempLong);

}

## Movement

### calibrate

void calibrate(){

int trefreshLCD = true;

int freq = 0;

delay(100);

while (1){

uint8\_t buttons = lcd.readButtons();

if (trefreshLCD){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Dist Cal");

lcd.setCursor(0, 1);

lcd.print(freq);

lcd.setBacklight(RED);

trefreshLCD = false;

}

if (buttons) {

if (buttons & BUTTON\_UP) {

trefreshLCD = true;

freq = freq + 5;

}

if (buttons & BUTTON\_DOWN) {

trefreshLCD = true;

freq = freq - 5;

}

if (buttons & BUTTON\_LEFT) {

return;

}

if (buttons & BUTTON\_SELECT) {

trefreshLCD = true;

countDown(3, "Dist Cal");

LServo.writeMicroseconds(1500 + freq);

RServo.writeMicroseconds(1500 - freq);

delay(2000);

LServo.writeMicroseconds(1500);

RServo.writeMicroseconds(1500);

}

}

delay(100);

}

}

### moveArc

//Moves the robot a certain number of degrees around a circle

// x

// 1,1 | 1,-1

// y-----|------

// -1,1 | -1,-1

void moveArc(int radius, double theta, int x, int y, int rate){

//Calculate circle circuference per degree

double outerpd = ((radius + 51) \* 2.0 \* 3.14159) / 360.0;

double innerpd = ((radius - 51) \* 2.0 \* 3.14159) / 360.0;

//calculate inner rate in mm/s

double innerRate = innerpd / outerpd \* rate;

//calculate runtime in ms

double time = outerpd \* theta / rate \* 1000;

//write to servo

//FCCW

if (x > 0 && y > 0){

int outerFreq = ServoFwdLookup(rate);

int innerFreq = ServoFwdLookup((int)innerRate);

LServo.writeMicroseconds(1500 + innerFreq);

RServo.writeMicroseconds(1500 - outerFreq);

}

//FCW

else if (x > 0 && y < 0){

int outerFreq = ServoFwdLookup(rate);

int innerFreq = ServoFwdLookup((int)innerRate);

LServo.writeMicroseconds(1500 + outerFreq);

RServo.writeMicroseconds(1500 - innerFreq);

}

//RCCW

else if (x < 0 && y < 0){

int outerFreq = ServoRevLookup(rate);

int innerFreq = ServoRevLookup((int)innerRate);

LServo.writeMicroseconds(1500 - outerFreq);

RServo.writeMicroseconds(1500 - innerFreq);

}

//RCW

else if (x < 0 && y > 0){

int outerFreq = ServoRevLookup(rate);

int innerFreq = ServoRevLookup((int)innerRate);

LServo.writeMicroseconds(1500 - innerFreq);

RServo.writeMicroseconds(1500 - outerFreq);

}

delay((unsigned long)time);

LServo.writeMicroseconds(1500);

RServo.writeMicroseconds(1500);

}

### moveLine

//Moves the robot a given distance at a given speed

void moveLine(int distance, int rate){

//calculate runtime

unsigned long time = (double)abs(distance) / (double)abs(rate) \* 1000;

//write to servo

if (rate > 0){

int temp = ServoFwdLookup(rate);

LServo.writeMicroseconds(1500 + temp);

RServo.writeMicroseconds(1500 - temp);

}

else if (rate < 0){

int temp = ServoRevLookup(abs(rate));

LServo.writeMicroseconds(1500 - temp);

RServo.writeMicroseconds(1500 + temp);

}

else

return;

delay(time);

LServo.writeMicroseconds(1500);

RServo.writeMicroseconds(1500);

}

### movePivot

void movePivot(int theta, int rate){

//We need to know some details of the robot

//specifically wheelbase and the wheel function

double perDegree = 102.0 \* 3.14159 / 360.0;

//calculate runtime

double time = perDegree \* abs(theta) / rate \* 1000;

//write to servo

//left

if ((int)theta > 0){

int leftFreq = ServoRevLookup(rate);

int rightFreq = ServoFwdLookup(rate);

LServo.writeMicroseconds(1500 - leftFreq);

RServo.writeMicroseconds(1500 - rightFreq);

}

else if ((int)theta < 0){

int leftFreq = ServoFwdLookup(rate);

int rightFreq = ServoRevLookup(rate);

LServo.writeMicroseconds(1500 + leftFreq);

RServo.writeMicroseconds(1500 + rightFreq);

}

else{

return;

}

delay((unsigned long)time);

LServo.writeMicroseconds(1500);

RServo.writeMicroseconds(1500);

//loop until milis have been reached

//write to servo

}

### ServoFwdLookup

int ServoFwdLookup(double rate){

double temp = (rate + 14.107) / 2.2139;

return (int)temp;

}

### ServoRevLookup

int ServoRevLookup(double rate){

double temp = (rate - 1.4333) / 2.1186;

return (int)temp;

}

## Miscellaneous

### countDown

void countDown(int x, String message){

for (int i = x; i > 0; i--){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(message);

lcd.setCursor(8, 1);

lcd.print(i);

delay(1000);

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(message);

lcd.setCursor(4, 1);

lcd.print("Running");

lcd.setBacklight(GREEN);

}