**Daryn Huang**

**University of Waterloo**

**Mechanical Engineering**

**Undergraduate**

**Candidate**

d42huang@uwaterloo.ca 20481948

**Erwin Fochtberger**

**University of Waterloo**

**Mechanical Engineering**

**Undergraduate**

**Candidate**

[efochtbe@uwaterloo.ca](mailto:efochtbe@uwaterloo.ca) 20462440

**Gabe Richard**

**University of Waterloo**

**Mechanical Engineering**

**Undergraduate**

**Candidate**

[g7richar@uwaterloo.ca](mailto:g7richar@uwaterloo.ca) 20470415

**Ivan Beentjes**

**University of Waterloo**

**Mechanical Engineering**

**Undergraduate**

**Candidate**

ibeentje@uwaterloo.ca

20441321

**Prof. Carol Hulls**

**Department of Mechanical and Mechatronics Engineering**

**University of Waterloo**

**Design and Production of a Guide Robot**

**Mar 29, 2013**

**Department of Mechanical and Mechatronics Engineering**

**Course No. GENE 121**

**University of Waterloo**

**200 University Avenue West**

**Waterloo, Ontario**

**Canada N2L 2G1**

Table of Contents

[List of Figures ii](#_Toc352094335)

[Introduction 1](#_Toc352094336)

[Mechanical Design and Implementation 2](#_Toc352094337)

[Software Design and Implementation 5](#_Toc352094338)

[Conclusions 7](#_Toc352094339)

[Recommendations 8](#_Toc352094340)

[References 8](#_Toc352094341)

[Appendix A 9](#_Toc352094342)

[Appendix B 9](#_Toc352094343)

# List of Figures

[Figure 1.The Guide Robot 4](#_Toc348365483)

[Figure 2. Guide Robot Sensor Positions 5](#_Toc348365484)

[Figure 3. Program Design Flowchart 6](#_Toc348365485)

# Introduction

The feats accomplished through programming are phenomenal. There are programs written for the sole purpose of calculating a simple mathematical operation, to others that form the basis of virtually everything used today.

One of the aspects of programming that was explored over the past few weeks was working on human interaction with robots; specifically, RobotC was used to develop programs for the Lego Mindstorms NXT system.

As part of the GENE 121 programming course students are given a final project to build and program a Lego Mindstorms NXT robot. The proposal presented by Daryn, Erwin, Gabe, and Ivan was an NXT robot system that could guide a visually-impaired person in a direction of their choice.

The robot needed to be controlled through interfacing with the NXT robot brick itself. Additionally, some type of handle was required to acts as a trigger for the robot to move forward. The ability to change its speed was a requirement, as was the incorporation of the primary sensors. These safety sensors would check for curbs, collisions, and objects in front of the user. Auditory prompting was considered essential since the target consumer would be a visually impaired person. For conserving power, the system needed to automatically shut off after a short period of inactivity.

The opportunity to design, and program, a robot to assist humans was regarded by the team as critical to their foundation as future Engineers. The proposed design would also hopefully allow others to have better insight as to what was possible with Lego Mindstorms.

# Mechanical Design and Implementation

The guide robot was designed to be held like a walking cane. A user could then walk forward with the robot in front sensing for danger. Unfortunately, the cables to connect the necessary parts were not long enough to make a full sized prototype so the robot arm was shrunk down to meet the cable length constraints. See Figure 1.

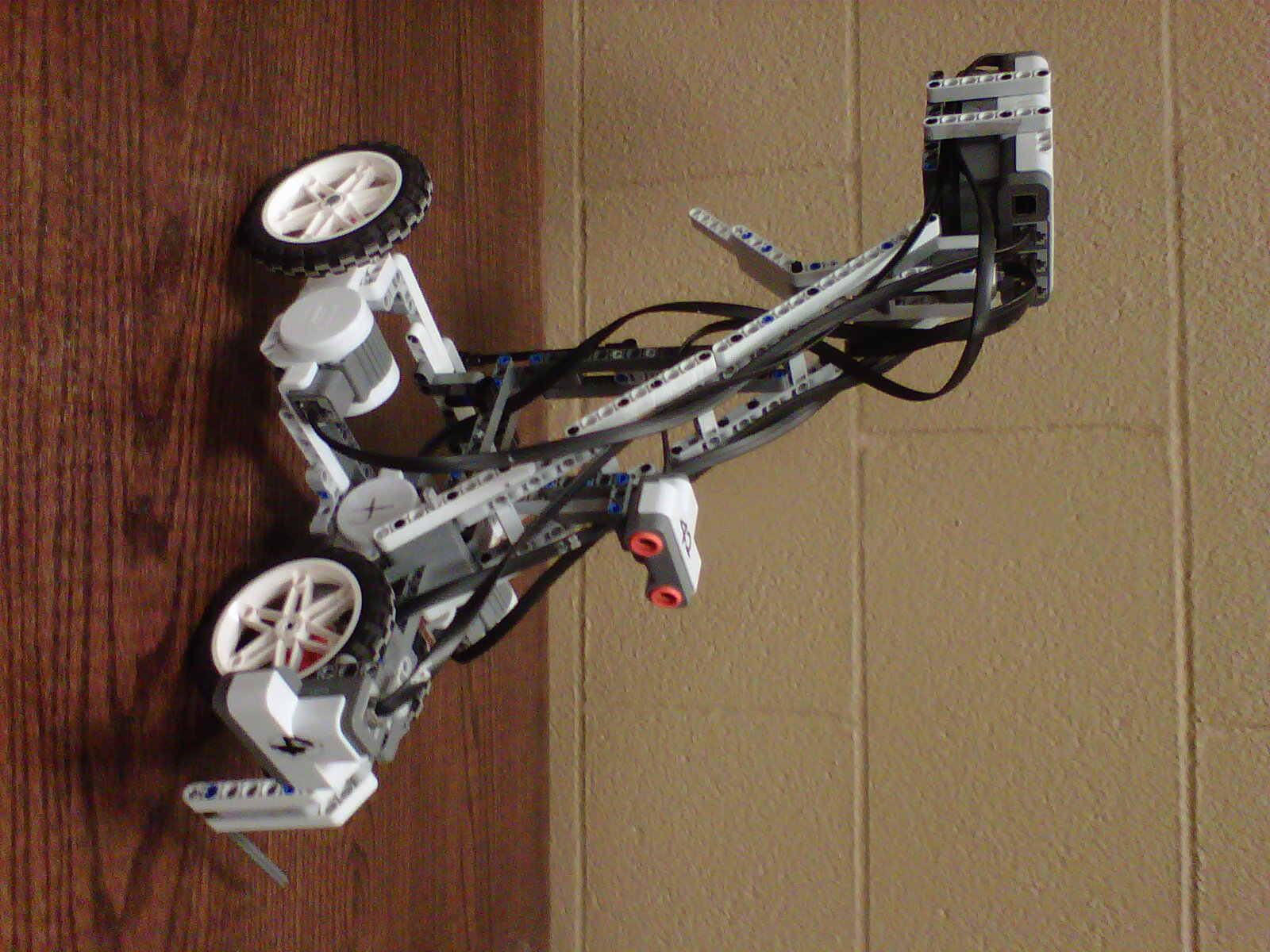


Figure 1. The Guide Robot.

Three motors were assembled in a triangular pattern creating a stable base, while providing additional power to propel the robot. Larger wheels were installed to increase undercarriage clearance, ensuring that small obstacles could be easily driven over; However extra gears to speed up the motors were not included due to the accuracy, and range of the distance sensors.

Two distance measuring sonar sensors were mounted on the walker’s chassis. See Figure 2. One was mounted at the front of the robot pointing down at the ground to sense for curbs and pot-holes. The second sonar sensor was attached twenty centimeters above the ground looking forward. This second sensor measures the distance between large obstacles and the robot.

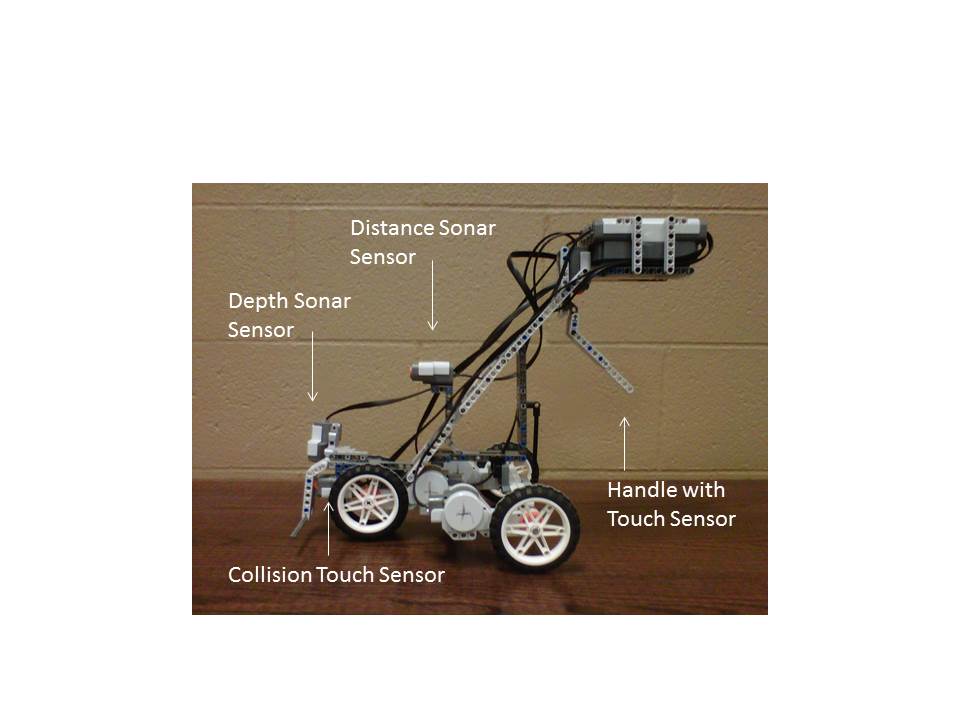


Figure 2. Locations of the robot’s two distance and touch sensors

Additionally, two touch sensors were connected to the robot. One was mounted in the front of the robot to stop the motors in event the distance sensor failed to detect an obstacle in sufficient time to prevent a collision. This sensor added another level of protection for the user. The second touch senor was built into a handle so that the robot only moves forward when the handle is gripped, reducing the possibility of the robot accidentally getting away from a blind user.

Incorporated into the handle is the robots main CPU block. This is so that the user can increase or decrease the speed of the robot by simply pressing a given button with the same hand that holds the walker. Auditory prompts were included so that a visually impaired user could hear what type of obstacle or danger was encountered. No obstacle avoiding steering was included, as this might bring the user into closer proximity with a given danger. Once a user has been alerted to an obstacle, or danger, they simply lift the guide robot to either side, and then start moving forward again. The robot was designed with its center of gravity just in front of the handle to make for an easy lift on the part of the user.

The main chassis was built by Daryn, Erwin, and Ivan. The handle assembly was built by Gabe and Eriwn

# Software Design and Implementation

In order to accomplish the task of creating a robot that can act as a walking cane for a blind person, the right software had to be created. The robot needed to sense for objects ahead of it, while looking for holes in front, and interface with an obstacle sensor in case the forward sonar sensor failed. The robot was to also have a handle that if released, the robot would stop and wait for the orange button to be pressed before proceeding. Speed increase and decrease was to be implemented to take into account that blind people walk at different speeds. The programming of the software was broken down into four parts:

1. Changing the speed of the robot (made by Ivan)
2. Checking for forward distance, any possible holes and a fail -safe sensor in case forward sensing sonar fails (made by Erwin)
3. A function calling upon the different distance checks and implementing the right sounds (made by Daryn)
4. Main program calling upon the above functions at the needed time (made by Gabe)

Changing the speed of the robot was a task that had to be kept separate from checking distances and calling sounds, as it had nothing to do with those functions. Checking the different sensors for depth, forward distance, and collision was kept as a separate task. Keeping the function that called upon the different distance checks and executing the right sounds made the software easier to code and more task specific. The main program was then coded with the functions created from the tasks above inside a do-while loop that would be checking whether the handle sensor was pressed or not to meet the criteria of the robot only functioning if the handle is held.

The functions pertaining to the changing of speed incorporated a for-loop so that the change in speed would be gradual as opposed to instantaneous. This would lower the instantaneous acceleration of the motors. The lower the instantaneous acceleration, the more comfortable the change of speed to the user will be. It is important to note that this robot is merely a prototype and was shrunk down in size due to a lack of longer wires connecting the sensor to the main brick.

Initially, the sound files pertaining to the different tasks were to be created so that the instruction would be clear and concise to the user. Unfortunately the sound files that were created were not able to be used successfully by the robot and therefore the sounds used in this program were selected from a bank of default sounds located on the Lego Mindstorms website [2]. The sounds to tell the user to press the orange button, or to hold the handle were supposed to be inside while loops that would execute the sounds until the right button/sensor would be activated. Unfortunately, the software was not reading the button/sensor while playing the sound, therefore it stayed in the while loop forever. In order to combat this problem, it was decided that the sound should be executed only once and then have the necessary while loops that would be waiting for the user to press the necessary button or handle.

The collision function was at first part of the main code. When looking at the grand scheme of the program, it was decided that the jobs of the collision sensor and the forward sensing sonar were the same and so the collision sensor was included into the function pertaining to sensing an obstacle forward and executing the right sound.

The following flowchart is the basis on which the whole program was coded.

# 

# Conclusions

The Guide Robot was designed to detect obstacles and warn a visually impaired person of potential dangers. It fulfills this role well, as it is capable of marking objects as far as 250 cm away and also detects holes or steps in front of it and emits a warning sound. The Robot also features a front bumper for circumstances where the sonar sensor would not detect an object in the front due to its low profile, or if the obstacle were to move in from the side. High wheels are essential for a good ground clearance. The high pivot point was introduced to make lifting the Robot as easy as possible. Lifting the Robot is required to choose the direction of forward movement. No automatic steering, only warning sounds were designed so that the user has full freedom to choose the direction. Although steering was omitted the user can select the desired speed by simply pushing the select buttons on the handle. Another important feature is when the user releases the handle the motors automatically stop. Also the Robot automatically shuts down after a given time. Altogether the Robot appears to be a decent guider detecting almost any obstacle with precision while still giving the user complete control and freedom.

# Recommendations

This Guide Robot Prototype has great potential for extension. One of the most obvious improvements would be the ability to select an optimum height for the handle such that it adapts to the users walking style. The handle itself could also be designed to fit more comfortably into a user’s hand. Future designs could also introduce alert options for deaf users. Also helpful, would be a strong enough color sensor to distinguish between red and green lights, in addition to improved depth sensors which could check for holes or gaps on any ground surface.

# References

[1]Gaddis, Tony, et al. *Starting Out With C++: Early Objects - 7th Ed.* Boston, Pearson Education Inc, 2011.

[2] *Sound Effects.* LEGO.com Mindstorms Funzone, 21 Mar. 2013. <http://mindstorms.lego.com/en-gb/funzone/downloads/default.aspx>.

# Appendix A

Full program code

void speedCon(int & initpow, int button, int count);

void incSpeed(int & initpow, int count);

void decSpeed(int & initpow, int count);

void sound(bool distCheck, bool depthCheck, int & iniPow, int & count);

bool distCheck();

bool depthCheck();

task main()

{

SensorType[S1] = sensorTouch; //handle sensor

SensorType[S2] = sensorTouch; //collision sensor

SensorType[S3] = sensorSONAR; //forward check sensor

SensorType[S4] = sensorSONAR; //depth check sensor

int iniPow = 0, button, count = 0;

PlaySoundFile("MS\_sfx\_note\_01.rso");

while (nNxtButtonPressed != 3);

while (nNxtButtonPressed == 3);

wait1Msec(2000);

// wait for orange button to be pressed

PlaySoundFile("MSVoiceTouchMyS.rso");

while (!SensorValue[S1]);

//wait for handle to be held

//initilizing motor power to 50

do

{

speedCon(iniPow, nNxtButtonPressed, count);

if (nNxtButtonPressed == 1 || nNxtButtonPressed == 2)

{

button = nNxtButtonPressed;

while (nNxtButtonPressed != -1);

speedCon(iniPow,button,count);

}

sound(distCheck(), depthCheck(), iniPow, count);

if (!SensorValue[S1])

{

motor[motorA] = 0;

motor[motorB] = 0;

motor[motorC] = 0;

count = 0;

iniPow = 0;

time10[1] = 0;

PlaySoundFile("MSVoiceTouchMyS.rso");

while (!SensorValue[S1])

{

if (time10[1] > 6000)

StopAllTasks();

}

}

}

while (SensorValue[S1]);

}

void speedCon(int & initpow, int button, int & count)

{

if(initpow==0 && count == 0)

{ //setting the default speed

count++;

for(int i=0; i<5; i++)

{

incSpeed(initpow,count);

}

}

else if(button==1)

{

incSpeed(initpow,count);

}

else if(button==2)

{

decSpeed(initpow,count);

}

else

{

motor[motorA] = -initpow;

motor[motorB] = initpow;

motor[motorC] = -initpow;

}

// setting the new speed so it can be used in the while statement

}

//increase speed function

void incSpeed(int & initpow, int & count)

{

for(int i=0; i<=9; i++)

{

if (distCheck() || depthCheck())

i = 10;

time1[0] = 0;

++initpow;

motor[motorA] = -initpow;

motor[motorB] = initpow;

motor[motorC] = -initpow;

while(time1[0] < 50 && !distCheck() && !depthCheck() && SensorValue[S1]);

}

sound(distCheck(), depthCheck(), initpow, count);

}

//decrease speed function

void decSpeed(int & initpow, int & count)

{

for(int i=9; i>=0; i--)

{

if (distCheck() || depthCheck())

i = -1;

time1[0] = 0;

--initpow;

motor[motorA] = -initpow;

motor[motorB] = initpow;

motor[motorC] = -initpow;

while(time1[0] < 50 && !distCheck() && !depthCheck() && SensorValue[S1]);

}

sound(distCheck(), depthCheck(), initpow, count);

}

//Sound function that check for depth and distance and executes the necessary procedures

void sound(bool distCheck, bool depthCheck, int & iniPow, int & count)

{

if (distCheck|| depthCheck)

{

count = 0;

iniPow = 0;

motor[motorA] = 0;

motor[motorB] = 0;

motor[motorC] = 0;

if (distCheck)

PlaySoundFile("MSVoiceMoveAway.rso");

else if(depthCheck)

PlaySoundFile("MS\_voice\_Danger.rso");

wait1Msec(3000);

PlaySoundFile("MS\_sfx\_note\_01.rso"); //waiting for orange button to be pressed and released once again

while (nNxtButtonPressed != 3);

while (nNxtButtonPressed == 3);

}

}

// SONAR SENSOR FUNCTIONS

bool distCheck()

{

if(SensorValue[S3] < 30 || SensorValue[S2])

return true;

else

return false;

}

bool depthCheck()

{

if(SensorValue[S4] > 15)

return true;

else

return false;

}

# Appendix B

The projects proposal

Our team of four has decided to build a robot that can guide a blind person forward in a direction of their choice. The Lego Mindstorm block will be mounted next to a hand-height handle. Once the robot has been turned on, it will require the handle to be gripped for it to proceed forward. This action will be accomplished by a touch sensor acting as the trigger. The base of the robot will be a triangle with a motor mounted under each corner to propel it forward. To increase user confidence and safety, the speed of the robot’s forward guiding motors will have the option of being adjusted. Two sonar sensors will monitor the distance of curbs and walls while alerting the user to these obstacles through auditory prompting. A second touch sensor mounted in the front of the robot will act a failsafe for proximity alerts. Additional features, such as turning itself off to save power, after a set period of disuse, and checking that all initial safety related conditions are satisfied will be incorporated into the robots software.