Dr. Contreras-Vidal

Cullen College of Engineering

University of Houston

4800 Calhoun Rd.

Houston, TX 77004

December 5, 2016

Dear Dr. Contreras-Vidal,

Attached is the Follow-Me Bot Progress Final Report for the 2016 fall semester. This report includes an Abstract, Purpose and Background, Problem, Need and Significance, Overview Diagram, Goal Analysis, Engineering Specifications and Constraints, Budget, Results, Recommendations and a Conclusion.

Quyen Le, Jhangir Awan, Muhammad Balagamwala, and Shriya Bhatnagar have successfully designed the final robot and it is capable of following a person successfully while carrying 20 lbs.

Sincerely,

Shriya Bhatnagar

Jhangir Awan

Quyen Le

Muhammad Balagamwala

Enclosed

Follow-Me Bot

Team 1: Fall 2016 Final Report

Shriya Bhatnagar

Jhangir Awan

Quyen Le

Muhammad Balagamwala

Sponsored by: UH – ECE Department

# Abstract

Automated shopping carts are the closest solution available to the seniors and the handicapped community for simply moving a few items from one place to another. It can already be seen that the cart is only available at grocery stores, and cannot be used at home. Many of these people need the same assistance of carrying and moving items in the comfort of their homes, offices or any indoor facility. Anything similar an automated shopping cart is not a practical solution, due to size, amount of power and manual operation. We present a unique solution, by designing a robot that serves the purpose of carrying small items while also following the targeted user, autonomously. The Follow-me Robot is able to accurately know the location of the user, by combining the information sent by the X-Box Kinect and the IR transmitter/receiver. Thus allowing the robot to successfully follow a target user while simultaneously matching their speed. The Follow-me Robot is able to fulfil its purpose by following the targeted user, autonomously. However, for a better result, the overall design and components used should be re-evaluated to provide more fluid and smooth movements by the robot and for faster processing time. Some recommended improvements for this project are: adding a voice activation feature, and voice commands, incorporating manual operation, and introducing wireless communication and control for the user.

# Table of Contents

Purpose…………………………………………………………………………………….1

Background…………………………………………………………………….………….1

Problem, Need, and Significance………………………………………………………….2

Overview Diagram………………………………………………………………………...3

Statement of Goals………………………………………………………………………...4

Specifications……………………………………………………………………………...4

Constraints………………………………………………………………………………...5

Engineering Standards…………………………………………………………………….5

Design and Methodology………………………………………………………………….5

Results……………………………………………………………………………………10

Conclusion……………………………………………………………………………….10

Recommendations………………………………………………………………………..10

Budget…………………………………………………………………………………....11

Source Code……………………………………………………………………………...12

# List of Figures

1 – Overview Diagram Front……………………………………………………………...3

2 – Overview Diagram Back………………………………………………………………3

3 – Goal Analysis………………………………………………………………………….4

4 – First Layer……………………………………………………………………………..7

5 – Electronics……………………………………………………………………………..7

6 – Kinect LED test………………………………………………………………………..8

7 – Ultrasonic Sensors…………………………………………………………………….8

8 – Kinect Interface………..………………………………………………………………9

9 – Final Robot……………………………………………………………………………9

10 – Budget………………………………………………………………………………11

# Purpose

The purpose of this project is to create a robot that can assist its users to transport lightweight items from A to point B, but directly following behind them. It will eliminate the burden of carrying items for the user.

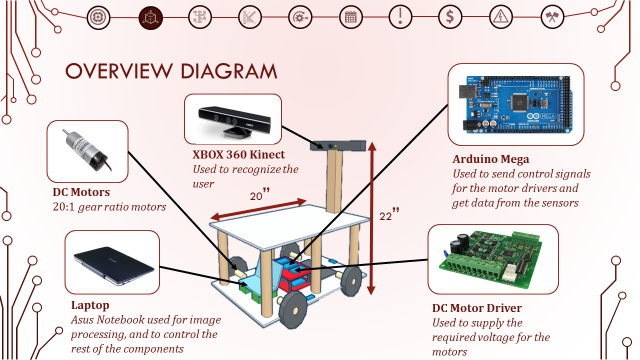
# Background

The follow-me bot is an autonomous robot, with two wooden platforms. The bottom level will be where the motors for the four mecanum wheels (2 on either side of the platform), along with the rest of the electronics, such as the ultra-sonic sensor, the IR sensor, microcontrollers and a laptop. The motors were selected to be powerful enough to go at least 3 mph at full power; this is so that they can keep up with an individual, walking at average human speed. The top level of the robot will be for the user to place their lightweight items, as well as where the X-Box Kinect will be mounted. It will be mounted on an arm, which will be average waist level in height. The X-Box Kinect in this project is being used for image processing, in real time. It will also be used for user-recognition, which will allow the robot to have a target user and avoid any other people that may disrupt the process. Due to the specifications on the X-Box Kinect, a computer will be processing the information gained through the X-Box Kinect, and then send commands out the microcontroller, which will be controller the behavior of the motors, allowing for the appropriate movement of the robot. Along with the information gathered from the X-Box Kinect, information from the ultra-sonic sensors that will be placed in front of the robot, and on either side, will be sent to the microcontroller. The sensors’ purpose is to the help avoid obstacles that cannot be seen by the Kinect (on ground level), as well as avoid running in walls and/or pillars near itself. An IR sensor will help in following the correct user. Putting all this together, the robot will be following the user by using the X-Box Kinect’s user-recognition and the IR sensor, and keeping a distance of approximately 1 meter, as well as processing its surrounding areas to avoid any stationary or moving obstacles in its path.

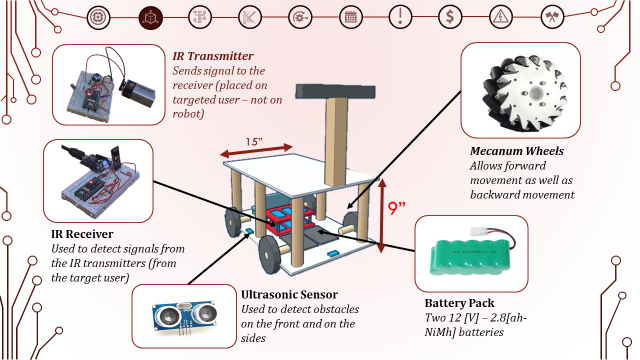
# Problem, Need and Significance

The general population wastes countless amounts of energy carrying lightweights back and forth on a daily basis. Instead of wasting all that energy, it can be harvested in a way, so that the users can use that energy in a more productive manner and avoid carrying lightweight items. With an autonomous robot that can withstand a substantial amount of weight, and also be able to follow a target user, it will eliminate the need to carry small items. Its versatility is also what the general population needs. It can be potentially used for any field and for anyone. It will be a significant change in everyone’s everyday routine, as it provides the ability to alleviate the burden of carrying lightweight items from point A to point B.

# Overview Diagram



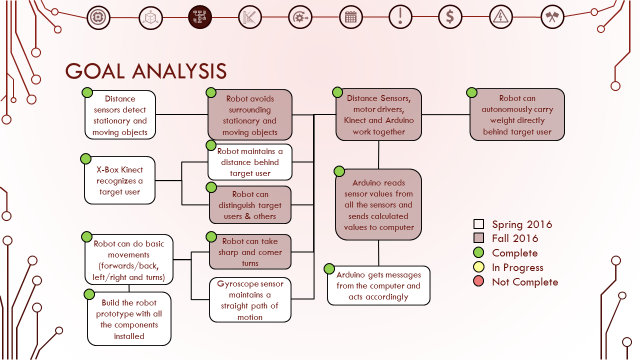
**Figure 1 – Overview Diagram Front View**

****

**Figure 2 – Overview Diagram Back View**

## Statement of Goals

Our goals for the project are shown below. It is easier to view the goals for the entire year and the accomplishments of them.



**Figure 3 – Goal Analysis**

The final goal of this project was to produce a robot that can carry weight and autonomously follow a user. Some of the important goals that led to the completion of our final goals were: Robot distinguishes between the target user and other. Robot can do basic movements, Distance sensors, motors drivers, Kinect, and Arduino work together.

# Specifications

The specifications for the robot are as follows: The robot must be able to carry 20 lbs, it must also maintain a speed of 3 mph while carrying the payload. In specific it should match the speed of the user. It has the ability to recognize as well as avoid objects around it. Battery will last one hour and it cannot travel on stairs, inclined paths, or elevators.

# Constraints

The constraints of the robot are as follows: The robot can only be used indoors due to the Kinect malfunctioning under sunlight. Cannot function efficiently or carpeted floor. Cannot go on stairs or elevators and cannot open any manual doors.

# Engineering Standards

As far as the engineering standards go for the follow-me robot, the only standard followed regarded the lithium battery used to power the robot. Lithium batteries have high energy in smaller packages, this is why they must be handled carefully when charging and transporting them.

# Design and Methodology

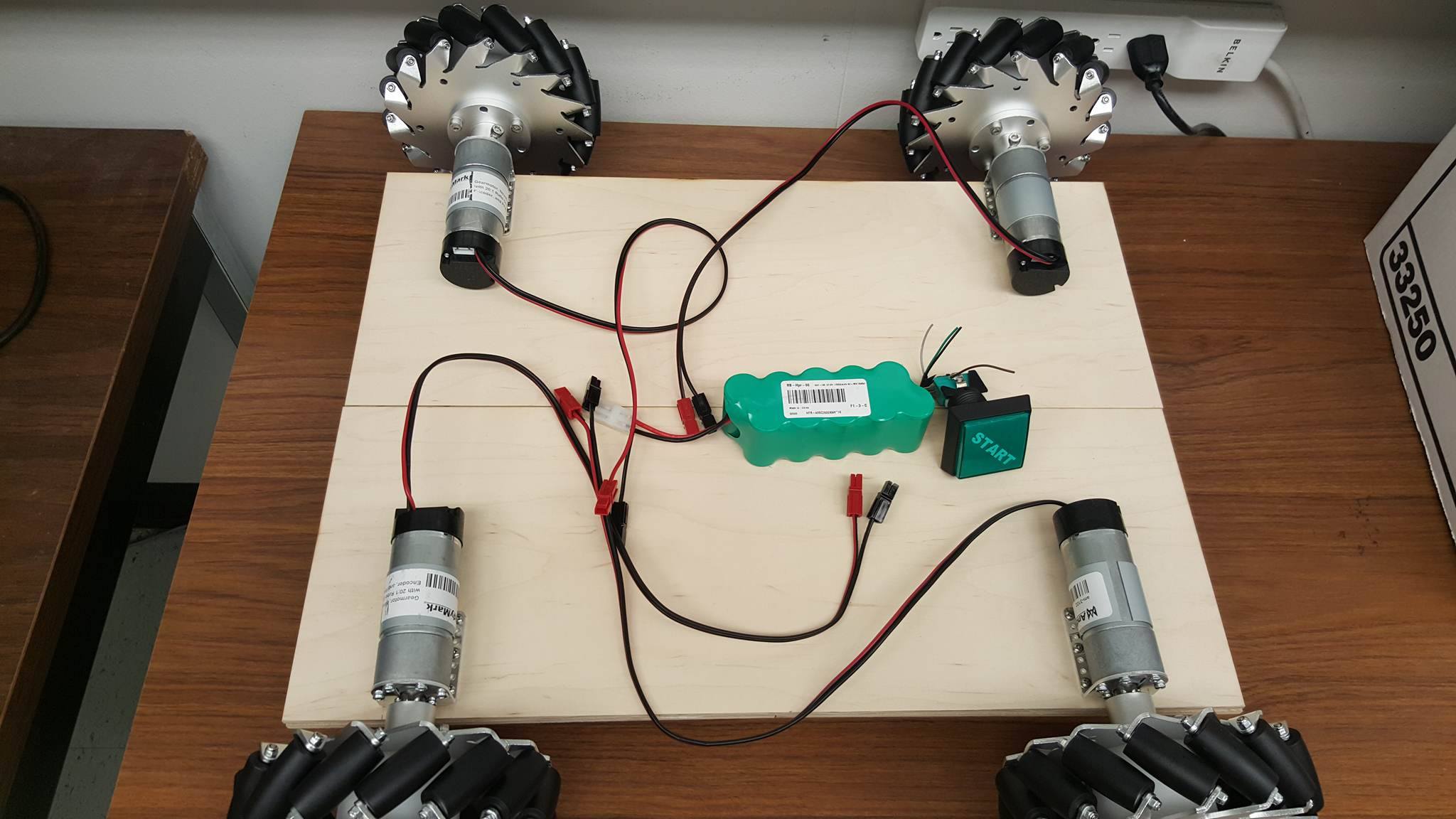
Follow-me Robot is wooden device, with the following dimensions: 15 [in] by 20 [in] by 24 [in]. The two levels are the same size, 15 [in] by 20 [in]. Each level was constructed by laser cutting two pieces in the size of 15 [in] by 10 [in], and two pieces in the size of 7.5 [in] by 20 [in], then the four pieces were glued together by placing the two 7.5 [in] by 20 [in] vertical on top of the two 15 [in] by 10 [in] pieces (like a plus sign). This was repeated again to create the second level.

Once the levels had been created, DC motors and Mecanum wheels were mounted on either side of the bottom level. Then to hold all the electronic components, a small two-layer platform was created by laser cutting 3 pieces of wood in the size of 5 [in] by 7 [in]. Each piece was placed on top of another by using four 2 [in] threaded beams placed in each corner for separation. In this 2-layer platform, the very top layer was used for mounting two motor drivers and the Arduino Mega, the middle platform was used for the IR receiver circuit, and the bottom platform was glued to the center of the bottom level of the robot. The space on the bottom was used for the two 12 [V]-2.8[ah-NiMh] rechargeable batteries. Once all the wiring had been connected, the second level for the robot was glued on top by first attaching three 2 [in] by 2 [in] by 12 [in] wooden beams on either side of the bottom level.

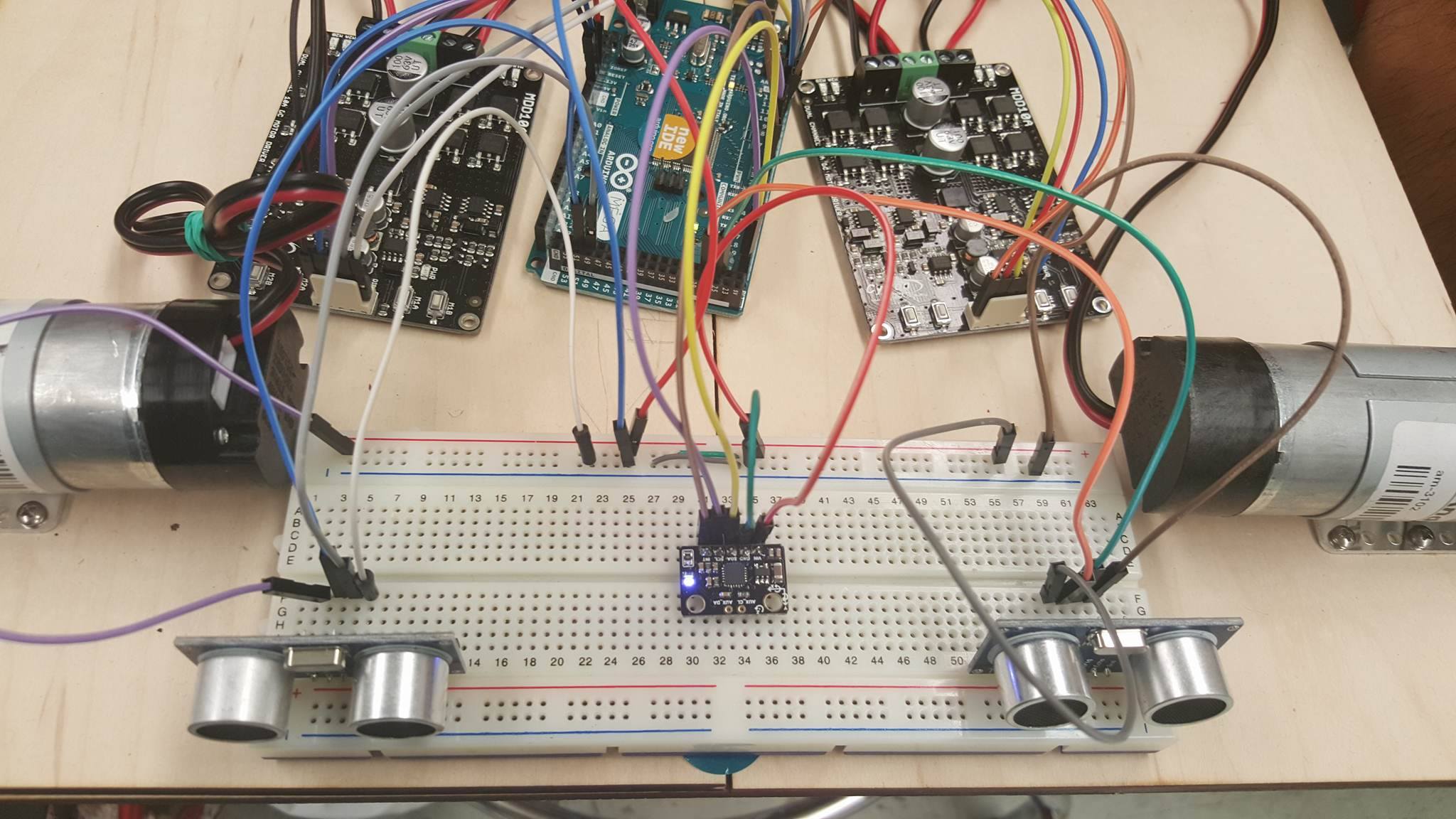
The robot was designed to have the X-Box Kinect at average waist height (about 22 [in] off the ground). In order to get this desired height, an arm-mount was designed. This arm was constructed by using two 2 [in] by 2 [in] by 8 [in], and one 2 [in] by 2 [in] by 6 [in] wooden beam. The two longer beams were then glued to ends of the small beam, leaving a 2 [in] gap in between. This structure was then glued to the back-center of the second level, and then a X-Box Kinect mounting piece was glued on top.

Once the robot had been constructed, the robot is programmed using Visual Studios, where the codes are written in C-Sharp (C#). The objective was to make the robot autonomous, and only follow the target user while avoiding stationary and moving objects. While working through the deliverables, it became clear that the X-Box Kinect could not be used alone to pinpoint the target user. Thus, an IR transmitter/receiver was incorporated to the project. Once the robot was able to follow the user, ultra-sonic sensors were added to detect stationary and moving objects that could not be picked up the X-Box Kinect.

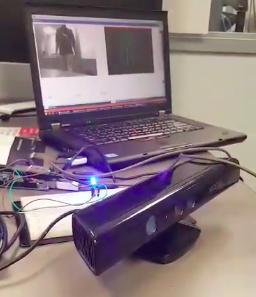
Below we have attached several images which denote the design and build process of the robot. At the end of this document the main loop source code for the robot is also shown.



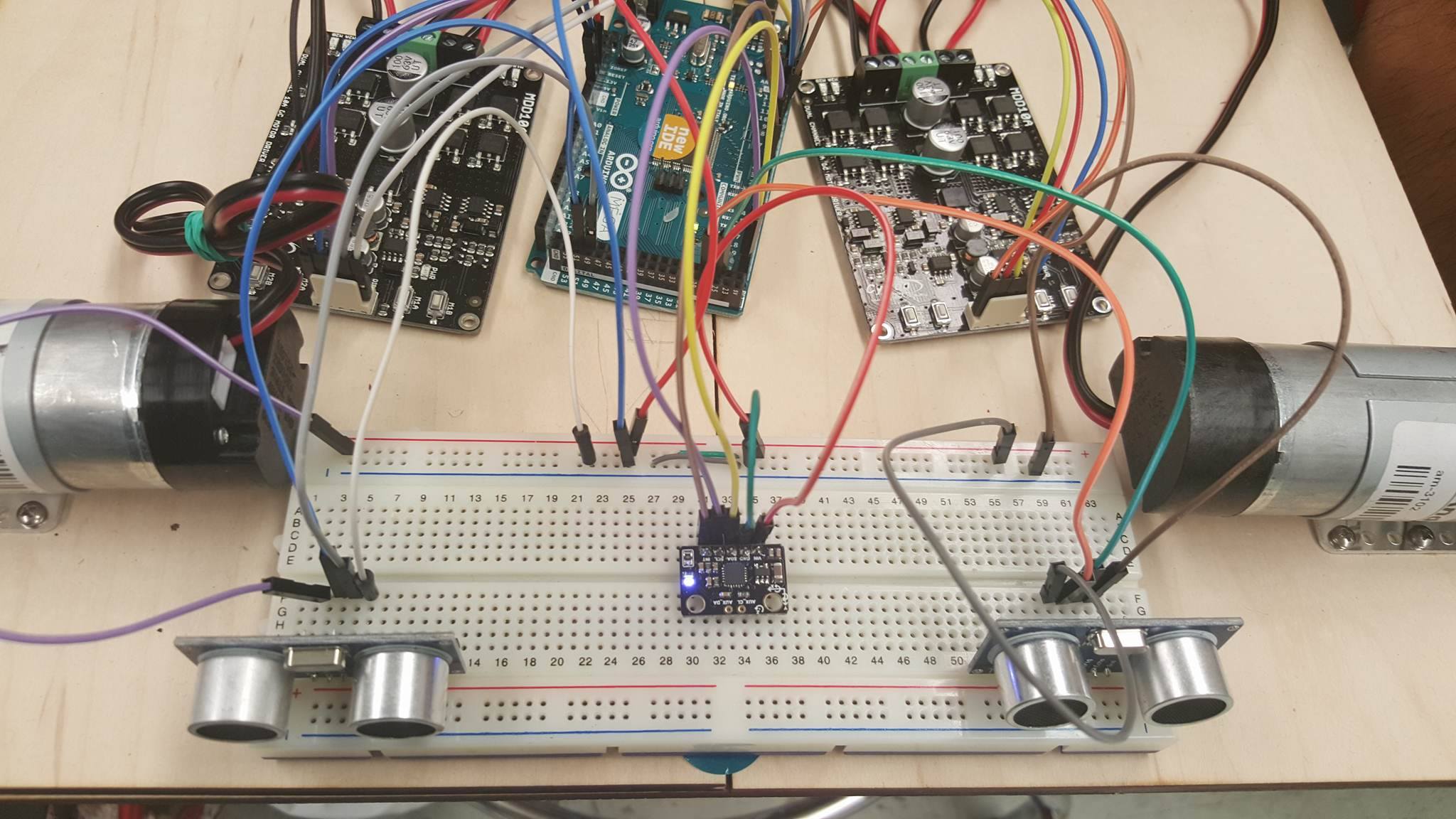
**Figure 4 – Beginning of first layer**

****

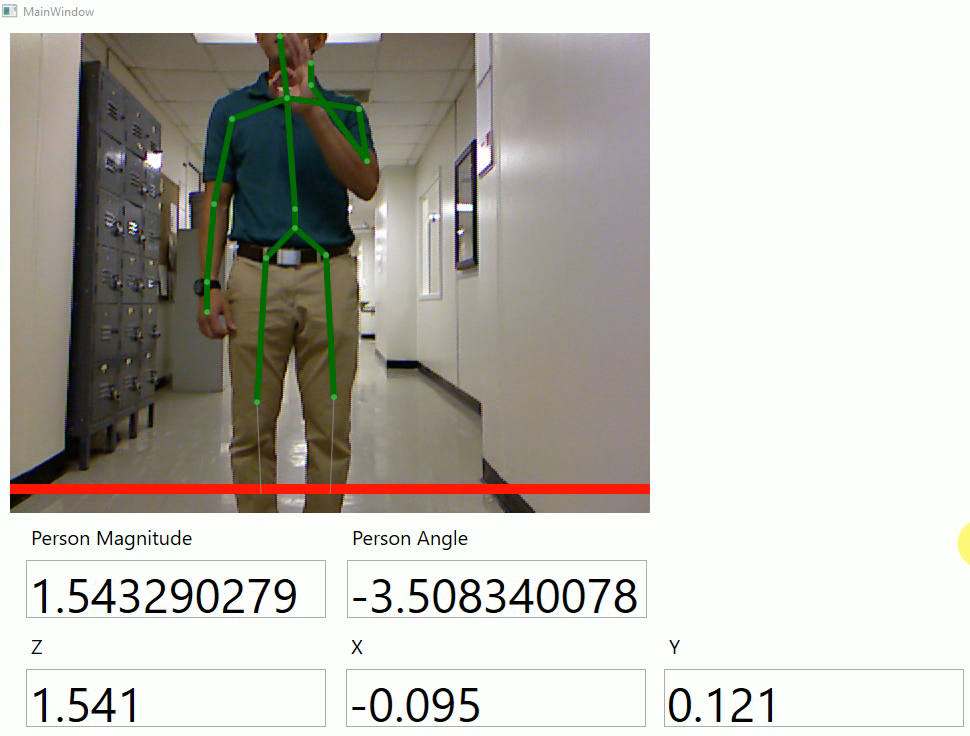
**Figure 5 – Electronics installed**

****

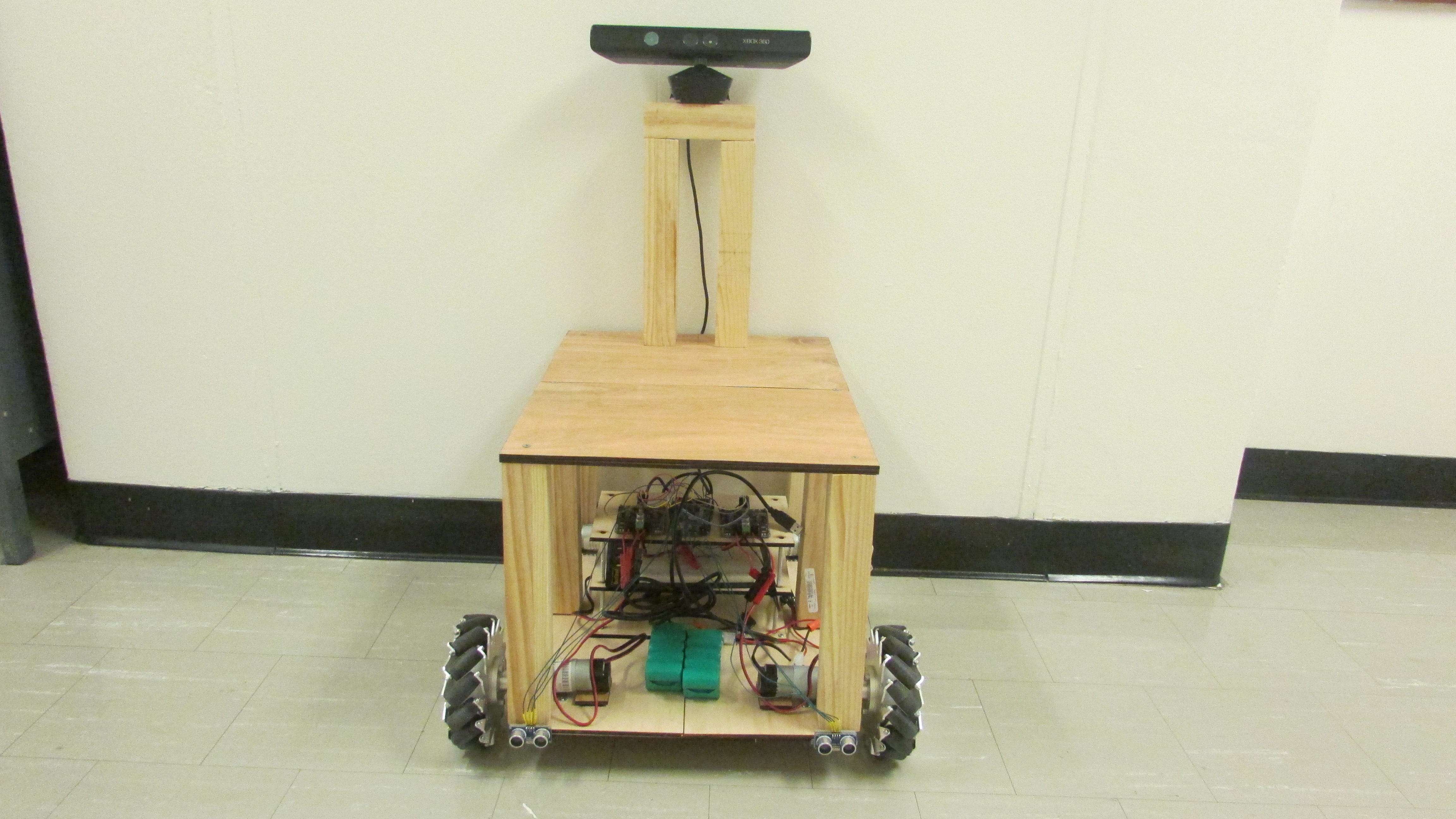
**Figure 6 – Testing Kinect coordinates with LED**

****

**Figure 7 – Ultrasonic sensors installed**

****

**Figure 8 – Computer interface showing what the Kinect sees**

****

**Figure 9 – Final Robot**

# Results

The Follow-me Robot is able to accurately know the location of the user, by combining the information sent by the X-Box Kinect and the IR transmitter/receiver. Thus allowing the robot to successfully follow a target user while simultaneously matching their speed. There are few concerns that surface when the robot is used in a busy environment, especially when the user walks faster than the specified speed. Some of these concerns consist of: the robot suddenly not detecting the user through the X-Box Kinect or it considers inanimate objects as the user and also, the ultra-sonic sensor sometimes has false readings.

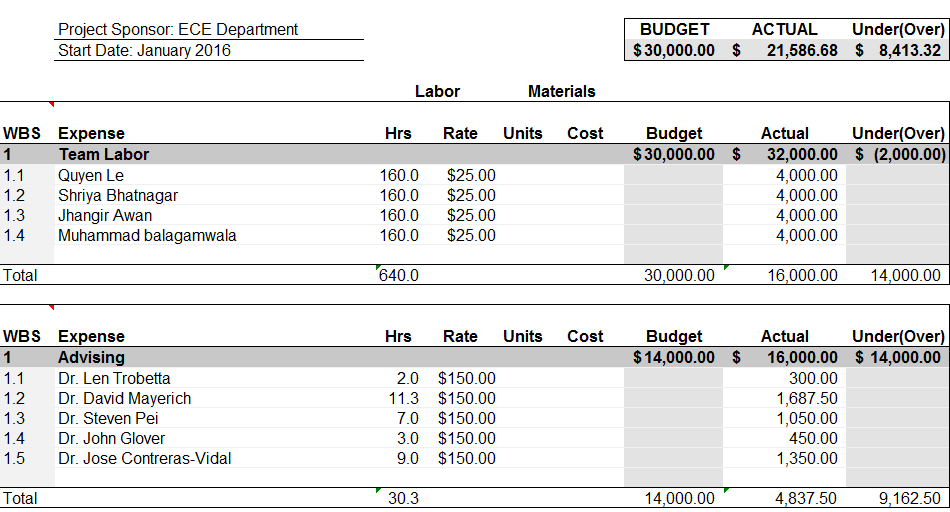
# Conclusion

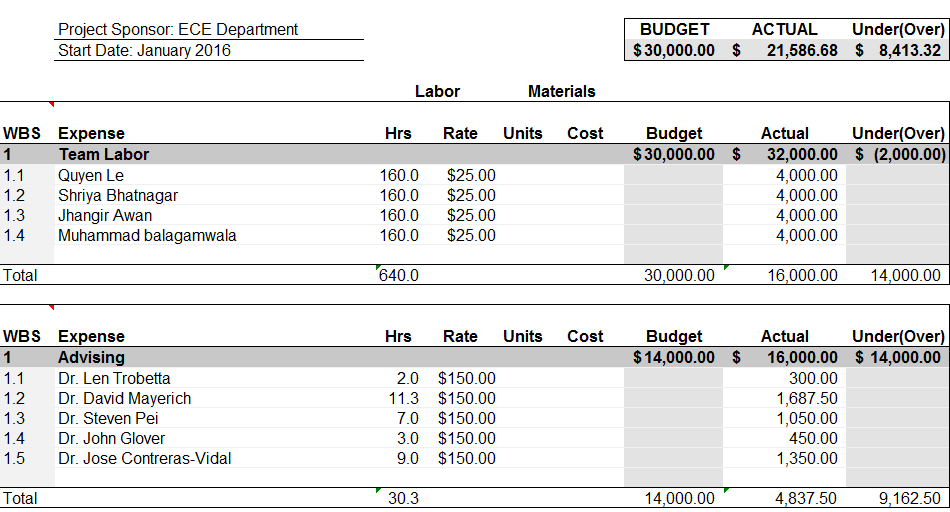
The Follow-me Robot is able to fulfil its purpose by following the targeted user, autonomously. However, for a better result, the overall design and components used should be re-evaluated to provide more fluid and smooth movements by the robot and for faster processing time. Some recommended improvements for this project are: adding a voice activation feature, and voice commands, incorporating manual operation, and introducing wireless communication and control for the user.

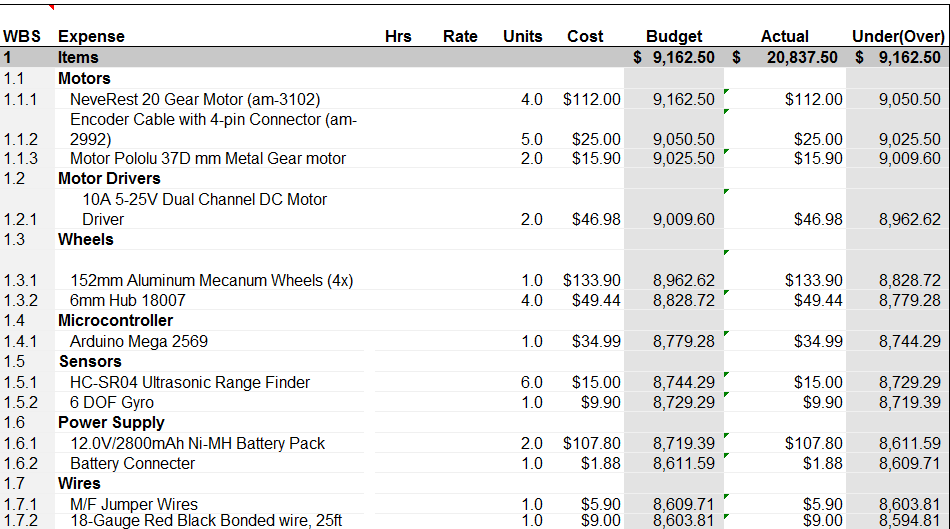
# Recommendations

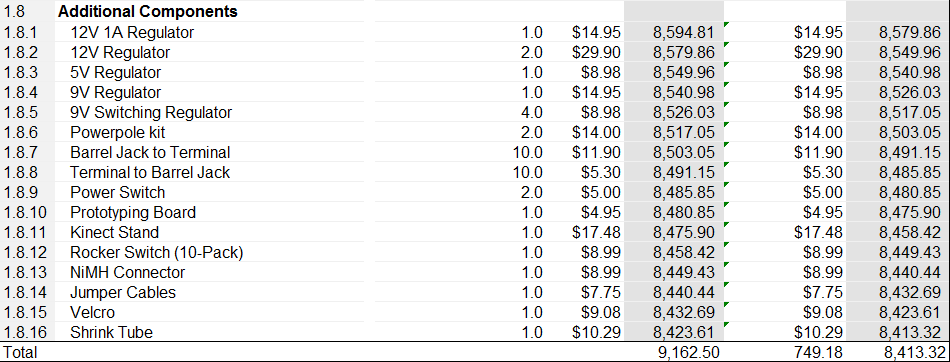
Some of the changes our robot can use are to re-evaluate the overall design and components such that the robot has more fluid and smooth movements. We can also add new features such as voice activation and commands, manual operation, and incorporate wireless communication.

# Budget









**Figure 10 - Budget**

# Source code

Below the main program loop is shown.

using System;

using System.Windows;

using System.Threading;

using Microsoft.Kinect;

using System.Collections;

using System.Collections.Generic;

namespace RobotFollowerWPF2

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public float personZ = 0;

public float personX = 0;

public float personY = 0;

public float needToMoveX = 0;

public float needToMoveZ = 0;

string commandText = "";

public bool currentlyTracking = false;

double personMagnitude = 0;

double personAngle = 0;

bool stop = false;

public Queue<Command> commandQueue = new Queue<Command>();

SerialCommunication arduino = new SerialCommunication();

SkeletonHandler skeletonHandler;

ColorHandler colorHandler;

KinectSensor kinect;

public MainWindow()

{

InitializeComponent();

arduino.PortNameToDetect.Add("Arduino");

new Thread(new ThreadStart(arduino.Connect)).Start();

initialize();

//new Thread(new ThreadStart(checkDistance)).Start();

}

public void UpdateTextBoxes()

{

while (true)

{

this.Dispatcher.BeginInvoke((Action)(delegate ()

{

spineZ.Text = Math.Round(personZ, 3).ToString();

spineX.Text = Math.Round(personX, 3).ToString();

spineY.Text = Math.Round(personY, 3).ToString();

commandTextBox.Text = commandText;

personMagTextBox.Text = personMagnitude.ToString();

personAngleTextBox.Text = personAngle.ToString();

//Thread.Sleep(50);

}));

Thread.Sleep(10);

}

}

public void Controller()

{

// double rotation = 0;

//magnitude 0.35 -> 39.5 inches / 5 seconds -> 2.3 mph

const double CONSTANTSPEED = .5;

int missCounter = 0; //Counts the number of loops that the person has not being tracked

Command lastCommand = new Command(0, 0, 0);

bool leftObstacle = false;

bool rightObstacle = false;

double angle = 0;

double magnitude = 0;

double rotation = 0;

while (true)

{

stop = false;

leftObstacle = false;

rightObstacle = false;

magnitude = 0;

rotation = 0;

angle = 0;

personAngle = Math.Atan2(personX, personZ) \* (180 / Math.PI); //angle of the person in degrees

personMagnitude = Math.Sqrt(Math.Pow(personX, 2) + Math.Pow(personZ, 2)); //distance from robot to person

rotation = personAngle / 90.0;

//double speed = CONSTANTSPEED;

//if (arduino.getLeftSonar() < 20)

//{

// leftObstacle = true;

//}

//if (arduino.getRightSonar() < 20)

//{

// rightObstacle = true;

//}

//Console.WriteLine("Left Sonar: " + arduino.getLeftSonar());

//Console.WriteLine("Right Sonar: " + arduino.getRightSonar());

if (personZ > 0) //check if the person is being tracked

{

missCounter = 0;

//determine magnitude

if (personMagnitude < 1.3)

{

//Back

magnitude = -CONSTANTSPEED;

}

else if (personMagnitude > 1.3 && personMagnitude < 1.75)

{

//Stop

magnitude = 0;

}

else

{

//Forward

magnitude = CONSTANTSPEED;

}

//determine rotation

if (personX < -0.3)

{

//person is on left

rotation = 1;

}

else if (personX > 0.3)

{

//person is on right

rotation = -1;

}

else

{

rotation = 0;

}

//determine angle

if (leftObstacle && !rightObstacle)

{

//obstacle on right

angle = -45;

}

else if (rightObstacle && !leftObstacle)

{

//obstacle on left

angle = 45;

}

else if (rightObstacle && leftObstacle)

{

//obstacle on both sides

stop = true;

}

//if (magnitude < 0)

//{

// rotation = -rotation;

//}

if (stop)

{

magnitude = 0;

}

if (magnitude < 0)

{

angle = 0;

}

if (rotation != 0)

{

magnitude = CONSTANTSPEED;

}

Command newCommand = new Command(magnitude, angle, rotation);

commandQueue.Enqueue(newCommand);

lastCommand = newCommand;

////FORWARD

//else if (personMagnitude > 1.75)

//{

// //TURN

// //Command newCommand;

// //if (!(leftObstacle && rightObstacle))

// //{

// //if (leftObstacle)

// //{

// // angle = 0.5;

// //}

// //if (rightObstacle)

// //{

// // angle = -0.5;

// //}

// if (personX < -0.3)

// {

// newCommand = new Command(Command.actions.TURN\_RIGHT, speed, angle, rotation);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// else if (personX > 0.3)

// {

// newCommand = new Command(Command.actions.TURN\_LEFT, speed, angle, rotation);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// //JUST FORWARD

// else

// {

// newCommand = new Command(Command.actions.FORWARD, speed, angle, 0);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// //}

// //else

// //{

// // stop = true;

// //}

// }

//}

//else //if the person is not being tracked

//{

// missCounter++;

// if (missCounter > 30)

// {

// commandQueue.Enqueue(new Command(Command.actions.STOP));

// stop = true;

// }

// else

// {

// commandQueue.Enqueue(lastCommand);

// }

//}

//if (personZ > 0) //check if the person is being tracked

//{

// missCounter = 0;

// //BACK

// if (personMagnitude < 1.3)

// {

// magnitude = -CONSTANTSPEED;

// //Command newCommand = new Command(Command.actions.BACK, speed, 0, 0);

// //commandQueue.Enqueue(newCommand);

// //lastCommand = newCommand;

// }

// //STOP

// else if (personMagnitude > 1.3 && personMagnitude < 1.75)

// {

// //TURN

// Command newCommand;

// if (personX < -0.3)

// {

// magnitude = CONSTANTSPEED \* 2;

// rotation = -1;

// //newCommand = new Command(Command.actions.TURN\_RIGHT, speed \* 2, 0, -1);

// //commandQueue.Enqueue(newCommand);

// //lastCommand = newCommand;

// }

// else if (personX > 0.3)

// {

// magnitude = CONSTANTSPEED \* 2;

// rotation = 1;

// //newCommand = new Command(Command.actions.TURN\_LEFT, speed \* 2, 0, 1);

// //commandQueue.Enqueue(newCommand);

// //lastCommand = newCommand;

// }

// //JUST STOP

// else

// {

// //newCommand = new Command(Command.actions.STOP);

// stop = true;

// //if (lastCommand != newCommand)

// //{

// // commandQueue.Enqueue(newCommand);

// // lastCommand = newCommand;

// //}

// }

// }

// //FORWARD

// else if (personMagnitude > 1.75)

// {

// //TURN

// //Command newCommand;

// //if (!(leftObstacle && rightObstacle))

// //{

// //if (leftObstacle)

// //{

// // angle = 0.5;

// //}

// //if (rightObstacle)

// //{

// // angle = -0.5;

// //}

// if (personX < -0.3)

// {

// newCommand = new Command(Command.actions.TURN\_RIGHT, speed, angle, rotation);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// else if (personX > 0.3)

// {

// newCommand = new Command(Command.actions.TURN\_LEFT, speed, angle, rotation);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// //JUST FORWARD

// else

// {

// newCommand = new Command(Command.actions.FORWARD, speed, angle, 0);

// commandQueue.Enqueue(newCommand);

// lastCommand = newCommand;

// }

// //}

// //else

// //{

// // stop = true;

// //}

//}

//}

//else //if the person is not being tracked

//{

// missCounter++;

// if (missCounter > 30)

// {

// commandQueue.Enqueue(new Command(Command.actions.STOP));

// stop = true;

// }

// else

// {

// commandQueue.Enqueue(lastCommand);

// }

//}

}

else //if the person is not being tracked

{

missCounter++;

if (missCounter > 30)

{

stop = true;

}

else

{

commandQueue.Enqueue(lastCommand);

}

}

Thread.Sleep(200);

}

}

public void CommandExecutor()

{

// Thread.Sleep(500);

while (true)

{

if (stop)

{

commandQueue.Clear();

commandQueue.Enqueue(new Command(0, 0, 0));

}

if (commandQueue.Count > 0)

{

Command currentCommand = commandQueue.Dequeue();

arduino.SendString("MOVE " + currentCommand.angle + " " + currentCommand.magnitude + " " + currentCommand.rotation);

//switch (currentCommand.action)

//{

// case Command.actions.FORWARD:

// arduino.SendString("MOVE 0 " + currentCommand.magnitude + " 0");

// break;

// case Command.actions.BACK:

// arduino.SendString("MOVE 0 -" + currentCommand.magnitude + " 0");

// break;

// case Command.actions.TURN\_LEFT:

// arduino.SendString("MOVE 0 " + currentCommand.magnitude + " " + currentCommand.rotation);

// break;

// case Command.actions.TURN\_RIGHT:

// arduino.SendString("MOVE 0 " + currentCommand.magnitude + " " + currentCommand.rotation);

// break;

// case Command.actions.STOP:

// arduino.SendString("STOP");

// break;

// default:

// arduino.SendString("STOP");

// break;

//}

}

Thread.Sleep(200);

}

}

public void initialize()

{

//find the correct sensor

foreach (var potentialSensor in KinectSensor.KinectSensors)

{

if (potentialSensor.Status == KinectStatus.Connected)

{

this.kinect = potentialSensor;

break;

}

}

if (kinect != null)

{

skeletonHandler = new SkeletonHandler(kinect, this, skeleton\_image);

colorHandler = new ColorHandler(kinect, this, color\_image);

while (!arduino.currentlyConnected)

{

Thread.Sleep(50);

}

new Thread(new ThreadStart(Controller)).Start();

new Thread(new ThreadStart(UpdateTextBoxes)).Start();

new Thread(new ThreadStart(CommandExecutor)).Start();

}

}

}

}