

EE3080 Design And Innovation Project Report

[Energy-Efficient Robots for Motion and Task Cooperation](https://ntulearn.ntu.edu.sg/webapps/portal/frameset.jsp?url=%2Fwebapps%2Fblackboard%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D_26488_1%26url%3D)

Mohamed Hafizuddin Bin Abu Samah  
U1320945B  
Supervisor: Asst Prof Hu Gouqiang

School of Electrical and Electronic Engineering  
Academic Year 2014/2015  
Semester 1

Table of Contents

**Table of Contents2**

**1.0 Introduction3**

1.1 Background3

1.2 Hardware3

**2.0 Integrating the Robot6**

2.1 Hardware Integration and Design6

2.2 Software Implementation6

**3.0 Features and Algorithm7**

3.1 Obstacle Avoidance7

3.2 Robot Field of View7

3.3 Avoidance Flowchart Algorithm8

**4.0 Improvement and Future Development9**

4.1 Current Improvements9

4.1.1 Difficulties faced9

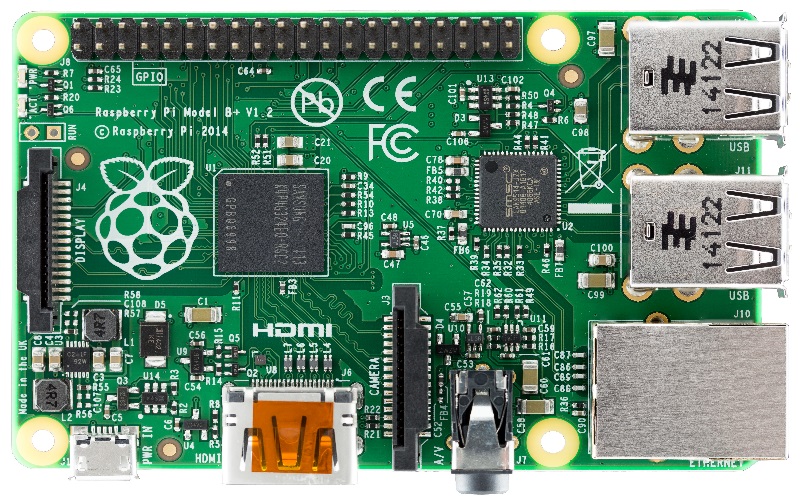
4.2 Future Improvements9

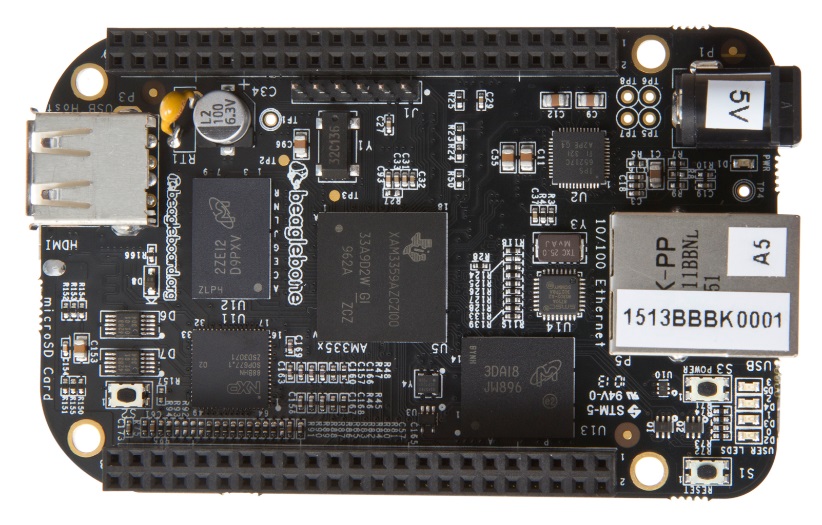
4.3 Contribution to project9

**References and Appendix10**

**1.0 Introduction**1.1 Background  
There are many different robots with different purpose on the market. However there are still lacking of autonomous robots as they are complex and hard to implement. In this project we were tasked to design and implement an energy efficient for motion and task efficient. This robot has to be fully autonomous and able to do various task such as obstacle avoidance.

1.2 Hardware  
As we are building our robots from scratch, we will require to have basic components as stated below:  
  
CPU x1  
Depth Sensor x1  
Motor Controller x1  
Wheels x4  
Chassis x1  
Battery x1





BeagleBone Black

Raspberry Pi

For our processors we were found 2 CPU that are favorable for our project these are BeagleBone Black and Raspberry Pi. After making a large comparison with each other, our group decided to go for the Raspberry Pi as the cost price is lower and the latest model B+ is much faster. I have decided not to put the power consumption used as it is a hard gauge to compare, it depends on what you plugged in and what applications are running.

|  |  |  |
| --- | --- | --- |
|  | Raspberry Pi | BeagleBone Black |
| Cost Price: | $45 (exc SD card) | $67 |
| USB Socket: | 4 | 1 |
| Processor: | ARM1176JZF-S | AM3358BZCZ100 Cortex A8 ARM |
| Community Support: | Wide | Small |



Microsoft Kinect

ASUS Xtion

For our main sensors we unanimously chose ASUS Xtion as the size is compact and only requires USB connection for data transfer and power supply unlike the Microsoft Kinect, it requires ACDC power supply.

|  |  |  |
| --- | --- | --- |
| **Device** | **Pros** | **Cons** |
| Microsoft Kinect | High quality of device drivers Stable work with various hardware models Has motor that can be controlled remotely by [iPi Recorder](http://wiki.ipisoft.com/IPi_Recorder) application: this makes device positioning more convenient | Bigger size (12" x 3" x 2.5" against 7" x 2" x 1.5") Higher weight (3.0 lb agains 0.5 lb) Require ACDC power supply Higher interference with another Kinect sensor in ["Dual depth sensor" configuration](http://wiki.ipisoft.com/User_Guide_for_Dual_Depth_Sensor_Configuration) Lower RGB image quality in comparison with MS Kinect |
| ASUS Xtion | More compact ( 7" x 2" x 1.5" against 12" x 3" x 2.5") Lighter weight (0.5 lb agains 3.0 lb) Does not require power supply except USB Lower interference with another ASUS Xtion / PrimeSense Carmine sensor in ["Dual depth sensor" configuration](http://wiki.ipisoft.com/User_Guide_for_Dual_Depth_Sensor_Configuration) Better RGB image quality | Less popular device Lower drivers quality Does not work with some USB controllers (especially USB 3.0) No motor, allow only manual positioning |

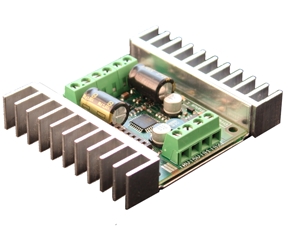
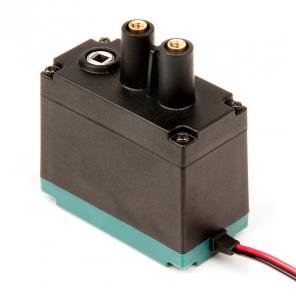


Omni Wheel

Normal Wheel

Our group has decided to use Omni Wheel as it has greater mobility then compared to using regular wheel. Using Omni wheel, our robot was able to do turns at much greater speed then compared to a robot using normal wheel as this was able to move in 360degrees of a direction.

Other Components:



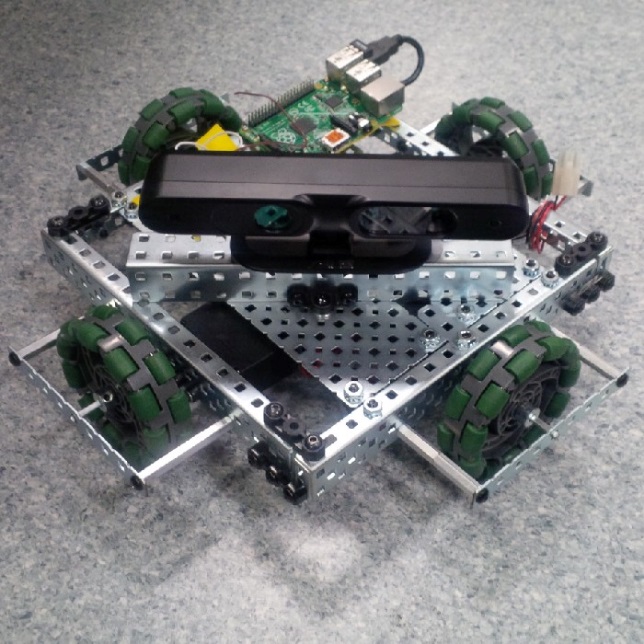
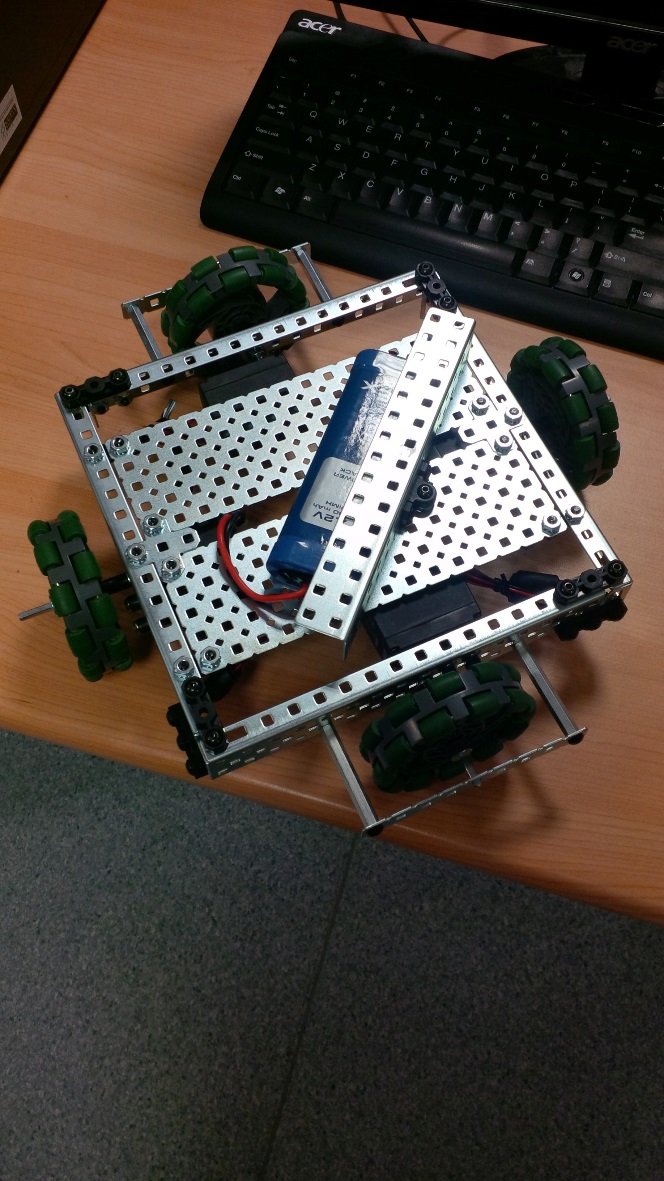
Sabertooth 2x12 Motor Controller

Vex 7.2V Nimh 3000mAh Rechargeable Battery

Vex 2 Wire Motor 393

All these components are selected unanimously as most of the product comes from the same supplier, therefore cheaper shipping costs and also most of these components are generic and has little or no difference with any other models available.

**2.0 Integrating the Robot**2.1 Hardware Integration and Design

****

Complete Integration

Assembling the parts

After we have received all the parts ordered, we quickly managed to integrate all the parts into working manner and was able to test out if all are able to connect and work. Our team have decided the design as such as it will improve the mobility and accuracy of the robot.   
  
2.2 Software Implementation  
Initially we tried to see if we are able to move the robot using Python, and we did manage to make it move. However, we realized if we are using Python we will not be able to use the camera as Python is a scripting language. Therefore we decided to move to C as it enabled to control the hardware output of the Raspberry PI GPIO pins as well as the camera.

During the software programming, I realized that the only way to control the motor controller was to send analog value, however the Raspberry PI can only give out digital values of 0v and 3.3v. To solve this, we did some research and found out the only way was to use Pulse Width Modulation. This works in a way that it keep sending out 3.3v still but it is in a certain frequency that it was able to get values from 0v to 3.3v.

Besides that, I also faced a problem compiling the program for the camera as I have not used command line complier GCC/G++ in Linux environment before for weeks until I realized that in order to compile the file I have to manually insert the libraries of the camera with the command line of it. In our case was “sudo gcc filename.c –lOpenNI” by inserting “-lOpenNI” it will add the libraries from OpenNI where I call them and use their functions.

**3.0 Features and Algorithms**3.1 Obstacle Avoidance  
For a robot to be a robot it has to have obstacle avoidance, as show below is a snippet of a video clip we recorded testing our avoidance algorithm.

****

From the snapshot above, you can see that it is moving towards the red pail, but as it come closer, it turns to the right side and move. Our algorithm for avoidance is simple but effective.

3.1.1 Robot Field of View

Robot

Vison Range

Top View

Side View

ASUS Xtion vision is between 0.8m and 3.5m with the field of view of 58° H, 45° V, 70° D (Horizontal, Vertical, Diagonal)

3.1.2 Avoidance Flowchart Algorithm

Start

Move Front

Turn Left

Turn Right

If Right = Obstacle

If Left = Obstacle

Halt

If Front = Obstacle

Initialize Camera

True

False

If Front = Obstacle

False True

False

True

True

Turn Left

False

**4.0 Improvement and Future Development**4.1 Current Improvements  
During our recent meetings, as we were initially divided into 2 subgroups, we now have 2 independent robots at our disposal. In order to integrate both robots we need to have some sort of communication between both robots. So current improvements are still in progress, however we now managed to at least link both robots and send information to each other via TCP Socket and having one of the laptop as the central server for both connections to wait for commands.

Both Robots Side By Side

4.1.1 Difficulties Faced  
Due to time constraints, we have to learn a lot more on TCP Socket (Networking Protocol) in a short notice. Our progress in connecting both robot to cooperate in a certain task is still under testing and we are not sure if we can do it on time for the final presentation.   
  
Our camera has the ability to do more than just detecting the distance but as again, due to lack of time and knowledge in the 3D depth camera technology, we weren’t able to utilize fully the technology that are at disposal for us to use.  
  
4.2 Future Improvements  
Considering if we have much more time and funding for this project, future improvements can be further done. Such improvements are Color Detection, Facial Recognition, Environment Mapping, Omni Directional Movements and many more.   
4.3 Contribution to the projectAs in our subgroup I have most relevant skills in robotics and programming, I took up as the programmer for our subgroup which handled all the software algorithms and software-hardware interfacing such as using the motor controller and the camera. It wasn’t an easy task as I am relatively new to the depth camera technology. However I feel that everyone did their part in this project to be successful.

**REFERENCES**

[1] <http://www.asus.com/Multimedia/Xtion_PRO_LIVE/#specifications>

[2] <https://www.dimensionengineering.com/products/sabertooth2x12>

[3] <http://www.vexrobotics.com/276-2177.html>

[4] <http://www.raspberrypi.org/documentation/>

[5] <https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/>

[6] <http://cs.smith.edu/dftwiki/index.php/Tutorial:_Client/Server_on_the_Raspberry_Pi>

[7] <http://cdn.intechopen.com/pdfs-wm/5610.pdf>

[8] <http://www.robot-r-us.com/vmchk/wheel-omni/high-performance-double-plastic-omni-wheel-100mm.html>

**APPENDIX**

if((middlePoint==0)&&(EdgeL==0)&&(EdgeR==0)){  
if(halttime<30){  
softPwmWrite(24,61);  
softPwmWrite(25,61);  
halttime=halttime+1;  
printf("Collision Imminent: NOTHING\n");  
}else{  
halttime=0;  
softPwmWrite(24,0);   
softPwmWrite(25,100);   
printf("STUCKED SO LONG, TURNING LEFT\n");   
delay(270);  
}  
lefttime=0;  
righttime=0;  
}  
else if((EdgeL!=0)&&(EdgeR==0))  
{halttime=0;  
softPwmWrite(24,67);   
softPwmWrite(25,55);   
delay(100);  
printf("Turning Right : Obstacle on the Left\n");  
}else if((EdgeL==0)&&(EdgeR!=0))  
{halttime=0;  
softPwmWrite(24,55);  
softPwmWrite(25,67);  
delay(100);  
printf("Turning Left : Obstacle on the Right\n");  
}else{  
halttime=0;  
lefttime=0;  
righttime=0;  
softPwmWrite(24,76.5);  
softPwmWrite(25,76);  
printf("GOGOGOGO Lets Goooooo!\n");  
}

import RPi.GPIO as GPIO  
import time  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(25,GPIO.OUT)   
GPIO.setup(24,GPIO.OUT)   
p=GPIO.PWM(24,1200)   
q=GPIO.PWM(25,1100)  
p.start(81)  
q.start(80)  
time.sleep(5)  
p.ChangeDutyCycle(81)  
q.ChangeDutyCycle(100)  
time.sleep(2)  
p.ChangeDutyCycle(100)  
q.ChangeDutyCycle(80)  
time.sleep(2)  
p.ChangeDutyCycle(20)  
q.ChangeDutyCycle(80)  
time.sleep(2)  
p.ChangeDutyCycle(81)  
q.ChangeDutyCycle(100)  
time.sleep(2)  
p.ChangeDutyCycle(81)  
q.ChangeDutyCycle(80)  
time.sleep(5)  
GPIO.cleanup()

C++ Code for Avoidance

Python Code for Movement