ESE 519

Kinect Physical Therapy

Final Report

Group 09

Yiran Qin

Bo Yuan

Shalini Chawala

Mingfei Shao

December 13, 2012

Introduction

Physical therapy, from many people's perspective, is a tedious and tough process. This process usually requires considerable amount of human resources: the therapist has to monitors the patient's movement continuously (usually for the entire period of exercise) and provide feedbacks as well as judgments about the exercises he/she has performed. On the other hand, the patients would also suffer since usually their morale will be low because of the injury. We are concerned with these issues so we come out with our Kinect Physical Therapy project.

Our idea is to use Microsoft Kinect, a motion sensing input device by Microsoft for the XBOX 360 gaming console, together with IMU-3000 triple-axis gyroscope, to track a patient's movement through his/her physical therapy process and provide necessary feedbacks. The Kinect is capable for tracking motion of body, head, and limbs. However, motions on small parts such as the rotation of wrist or finger movements cannot be detected by Kinect. Thus, the IMU-3000 gyroscope is critical as a completion for the whole motion capture process.

Our design of the system is based on two approaches: namely, Observation and Guidance. These are the two goals that we are going to incorporate from the entire system design and implementation stages. For observation, we implemented a well-done Java graphic user interface (GUI). The patients would be able to see the examples they should follow on the left, with a real-time representation of the movements of the patients themselves on the right.We also provided some more complicated methods like the overlay mode or the adaptive tracking mode in the GUI.In that way, the patients can easily get known about what he/she is doing. Our system is also capable of guidance: a noticeable judgment panel was placed on the GUI. It will keep on displaying green if the patient is doing right as compared with the sample, and turns into red if he/she did something wrong. In that case, a motor on the special glove will vibrate to alert the patients that he/she should pay more attention on following the sample's movement.

System Overview

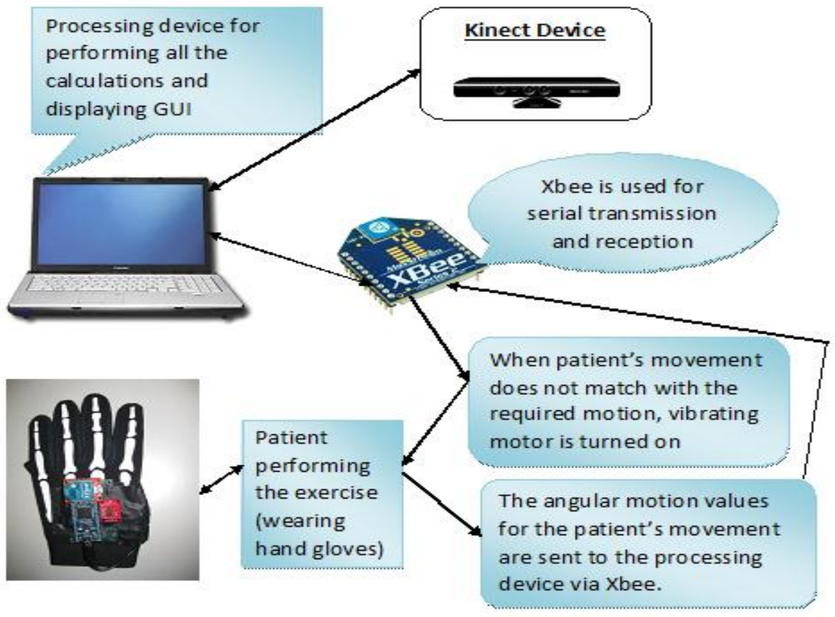

Processing device for performing all the calculations and displaying GUI
Kinect Device
Xbee is used for serial transmission and reception
When patient's movement does not match with the required motion, vibrating motor is turned on
Patient performing the exercise (wearing hand gloves)
The angular motion values for the patient's movement are sent to the processing device via Xbee.

Figure 1 System Architectural Diagram

Figure 1 above describes the overall architecture of the Kinect Physical Therapy system. The whole system consists of a Microsoft Kinect sensor, a mbedNXP LPC1768 microcontroller, an IMU-3000 gyroscope, a vibrating motor, two XBee radio modules and a computer running the Java application. Among them, one XBee radio module was placed on a special glove together with the motor, the IMU gyroscope and the mbed microcontroller. The other XBee module is connected with the computer, as well as the Kinect sensor, via USB ports.

The workflow of the system starts with the user (that is, the patient). With the gloves on, the user simply moves his/her limbs and body, tries to follow the sample on the GUI. The Kinect tracks the changes in body positions. Through the use of software OpenNI, PrimeSense and NITE, these positions can be converted into coordinates of different points representing the body. Once these points are read by the Java application, it calculates the angles of the shoulders, elbows and wrists. Along with the gyro data, the application can compare these angles with the sample's angles and to decide if the user is in the correct position. The gyro data is collected, processed and sent by the mbed microcontroller. And the result of judgment will be sent back in a similar manner. After receiving the result, the mbed will have to decide whether to turn the vibrating motor on to notify/alert the user.

Hardware Design

Our Kinect Physical Therapy system has the following parts of hardware:

- Microsoft Kinect sensor.
- mbedNXP LPC1768 microcontroller.
- IMU-3000 triple-axis gyroscope.
- XBee radio module.
- ROB-08449 vibration motor.
- Cables and battery pack

An overview of all the components in our system is shown as in **Figure 2**.

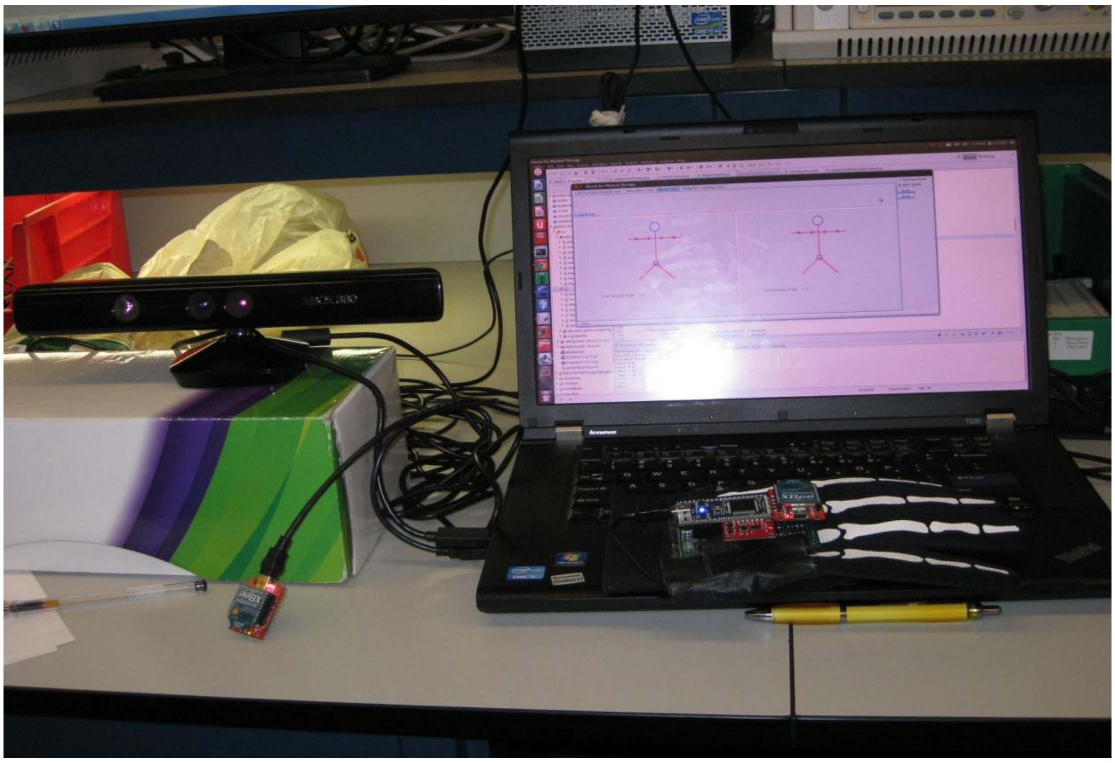


Figure 2 Hardware Overview

1. Microsoft Kinect Sensor

The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone, which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.The depth sensor consists of an infrared laser (IR) projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions (also known as the structural light).

2. mbedNXP LPC1768 Microcontroller

The mbed microcontroller is a small-size, low-power consumption embedded microcontroller. It is based around an NXP microcontroller, which has an ARM Cortex M3 core, running at 96 MHz, with 512 KB flash, 64 KB RAM, as well as several interfaces including Ethernet, USB Device, CAN, SPI, I^2C and other I/O.

We choose to use the mbed microcontroller for this project not only because of its size and power consumption advantages, but also for its ease for programming: the mRK system running on the mbed has provided a number of APIs, which made things even simpler.

3. IMU-3000 Triple-axis Gyroscope

The IMU-3000 triple-axis gyroscope has an embedded 3-axis gyroscope and Digital Motion Processor (DMP) hardware accelerator engine with a secondary I2C-master port that interfaces to third party digital accelerometers.In this project, we use this gyroscope to detect the angular movement of human hand. We set the angular displacement to be 0 when the gyroscope is placed face-up horizontally. As the hand rotating, the x- and y-components of the gyroscope change. Thus, we can derive the angle of displacement of the user’s hand.

4. XBeeRadio Module

XBee 802.15.4 is an initial point-to-point (PTP), point-to-multipoint (PTM) radio running the IEEE 802.15.4 protocol.It is extremely easy to use, just need to read/write as what you would do with serial connection, and it is also reliable. In the project, one XBee module is connected to the mbed and the other to the PC.

5. ROB-08449 Vibration Motor

The ROB-08449 vibration motor is a coin (pancake) style motor with a 2-3.6V operating range.The vibration motor is connected to the GPIO pin of the mbed microcontroller. The microcontroller can toggle the motor on and off by setting the voltage level of that pin to be high or low respectively.

A comprehensive circuit and pin configuration diagram is attached in **Figure 3** below, pin numbers on the mbed have also been denoted.

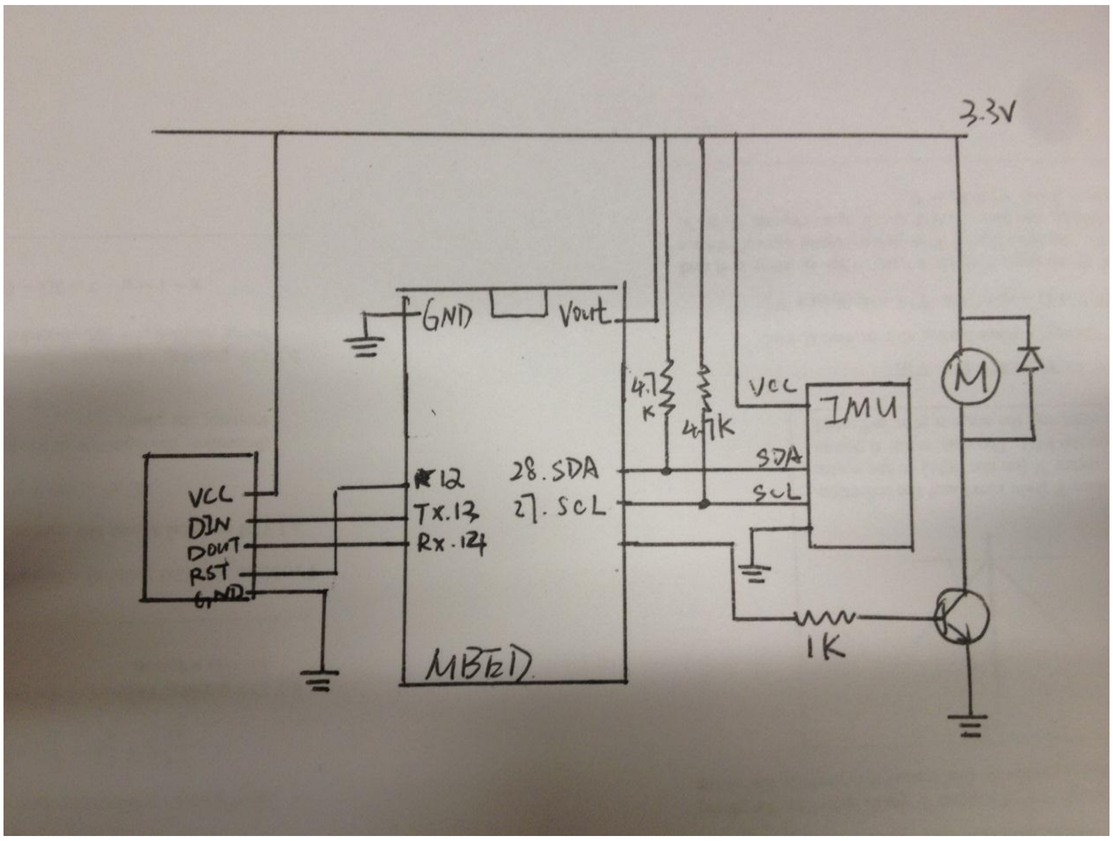


Figure 3 Circuit Configuration and Pin Connection of the Project

Software Design

A list of our software envirnmonet for developing and executing our project:

- OpenNI (unstable build for Ubuntu 12.04 x86 v1.5.4.0 @ http://openni.org/Downloads/OpenNIModules.aspx)

- SensorKinect/PrimeSense (v5.1.2.1 @ https://github.com/avin2/SensorKinect)
- NITE (unstable build for Ubuntu 12.04 x86 v1.5.2.21 @ http://openni.org/Downloads/OpenNIModules.aspx)
- Ubuntu 11.10 (Oneiric Ocelot, desktop version, x86 @ http://releases.ubuntu.com/11.10/)
- OpenJDK 6 (build b27 @ http://download.java.net/openjdk/jdk6/)
- Eclipse IDE (Juno, 4.2 SR1, Linux 32 bit @ http://www.eclipse.org/downloads/?osType=linux)

Other libraries needed to get the project running include:

- RXTX2.1 (RXTXcommon.jar @ http://rxtx.qbang.org/wiki/index.php/Download)
- Java3D (v1.5.2 @ http://java3d.java.net/binary-builds.html)

1. Kinect Set-up:

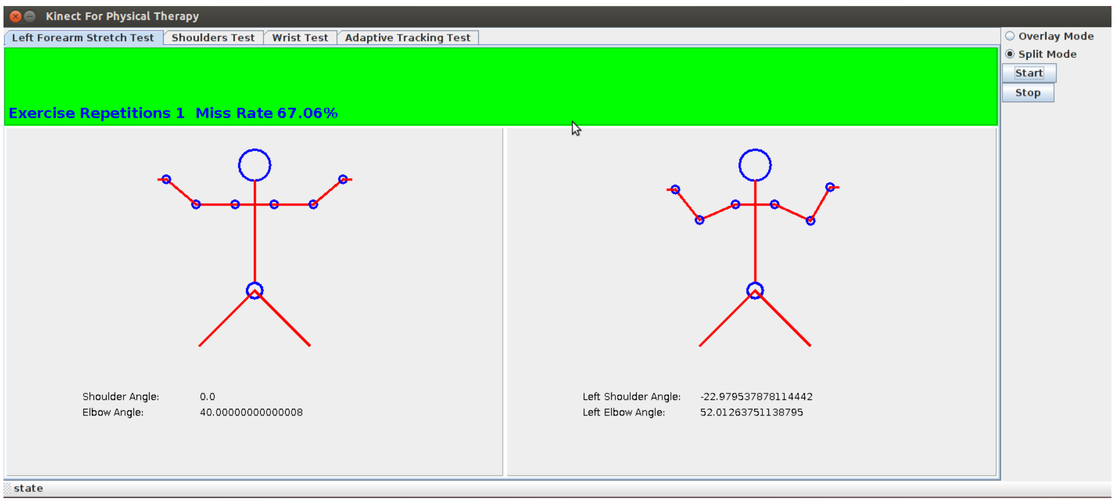
To get the Kinect working, we must first install everything it needs correctly. In order to do that, the following three things must be configured properly: OpenNI, PrimeSense (for driving the Kincet), and NITE (for enabling skeleton tracking). We found this instruction to be particular useful @ (http://mitchtech.net/ubuntu-kinect-openni-primesense/). NITE in this bundle is capable for tracking the human skeleton and generating the points' coordinates for joints. A better thing is, the OpenNI has provided us with a set of sophistic Java API that allows us to initialize, calibrate and read data out from Kinect easily in Java.

2. Programming on mbed

As compared to other embedded microcontrollers, the mbed is friendlier to programmer and easier to program with. The mRK system running on the mbed allows us to specify not only the execution time of each task, but also the priority for each task. For this project, our program on the mbed side is simple: it will first initialize timers and GPIO pins, and then it goes into an infinite loop, each with 0.5 seconds execution time. In each iteration, the mbed will do the following things sequentially: to read the data from the IMU-3000 gyroscope, to calculate the angle from the data, to write the value of angle into XBee, to read the result out from XBee, and to drive the vibration motor if necessary.

3. Java GUI

We choose to implement our main program based on Java, primarily because of its great cross-platform ability. Thus, we would have to worry little about the complier if we try to move our application on to some embedded platform like BeagleBoard. What's more, design and implementing a sophisticated GUI on Java is easier than using other languages. **Figure 4** below shows the GUI of our application.



Exercise Repetitions 1 Miss Rate 67.06%

```
Shoulder Angle:     0.0
Elbow Angle:        40.00000000000008
```

```
Left Shoulder Angle:     -22.979537878114442
Left Elbow Angle:        52.01263751138795
```

Figure 4 The GUI of the Application

When designing the GUI, we use a tabbed panel as the foundation. An option bar allows user to changing between the overlay mode and normal mode, as well as the start/stop button, was placed on the right-most side. Area above the tabbed panel has been divided into three different regions: a long region on top, called judgmentpanel is in charge of displaying results. It will turn to green if the user performs correctly and to noticeable red to warn the user. It can also show how many repetitions for an exercise have been done so far and the "miss rate" for this set of exercises. Below that, the whole area has been divided by half. The left hand side is so-called **"sample view".** It shows the real-time positions of the sample according to the pre-defined data by the therapist. The right hand side is the **"player view",** which is the real-time representation of the user who is captured by the Kinect. The skeleton is draw from the points generated by NITE.

For both panels, a series of important data, particularly the angles the system is tracking, will be displayed under the skeleton models.

Currently there are four exercise sets in our application: a **forearm exercise** which requires the user to move their elbow joints; a **shoulder exercise** which requires the user to swing their arm along with the shoulder; a **wrist exercise** which requires the user to rotate their wrists slowly; and an **adaptive tracking test** which can adjust the speed of the sample in order to try to match the speed of the user.

If the "overlay mode" is chosen, the sample view on the left hand side will disappear and be merged onto the user view on the right. The real-time RGB image of the user will replace the skeleton representation in the user view. On top of that, the sample skeleton will be "overlaid". In this mode, the sample skeleton will also provide a representation of the "error tolerance margin", denoted as yellow polygon areas on the sample so that user can have a better conception on what he/she should do to keep the movement correct.

4. Java Application

The entire Java application contains three packages: org.OpenNI, edu.seas.upenn.rendering, and edu.seas.upenn.ese519.

- **org.OpenNI**

  This package contains all necessary classes for the OpenNI API that can control and play with the Kinect sensor.

- **edu.seas.upenn.rendering**

  This package is responsible for rendering the 3D model from the data generated by the Kinect sensor. The 3D rendering model can only be found under the "Adaptive Tracking Test" tab with the normal mode selected. This package requires Java3D library to execute.

- **edu.seas.upenn.ese519**

  This is the place that the majority of our packages go into. It is not only the backbone of our application, but also the front end.

  The class "Constants.java" contains some pre-defined constants we used throughout the entire application. These constants including window widths and heights, tab numbers, update rate, tolerance and number of repetitions.

The class “DataGenerator.java” is in charge of initializing, configuring and calibrating the Kinect sensor though OpenNI. It is also responsible for generating all points’ data and the RGB images from the Kinect and supplies these data to other parts of the program.

The class “SerialComm.java” is responsible for reading/writing data from/to a serial connection, which would be used by the XBee module.

The class “PlayerPanel3D.java” is responsible for rendering the 3D skeleton model, update it and place it onto a panel.

The class “JudgementPanel.java” is the red/green display panel on the top. The judge is made basically based on the angles of the user at that time as compared with the angles of the sample at the same time. It has other methods for satisfying other modes such as the adaptive mode or the overlay mode. This panel is critical because it is responsible also for adjusting the speed of the sample in the adaptive mode and send out the result. For doing adaptive tracking, we implemented a PID feedback controller to adjust the speed of the movement. The performance of the PID controller is noticeable while still requires some further work since the movement of user is non-linear.

The class “NewJPanel.java”is the setup panel used in adaptive tracking. For the adaptive mode, the user needs first to specify the movement he/she wants to practice with by defining the min/max value of angles he needs. After that, a sample will be built based on these data.

The class “OverlayPanel.java”is in charge of displaying the overlay mode.This class reads out the RGB image data from the DataGenerator class, combines them with the sample skeleton and displays them out on a panel.

The class “MainFrame.java” is the backbone frame class of the whole application. It contains a tabbed panel, on which all other panels will be placed. To add new tab, just simply modify the MainFrame.java, set all things up the same way as pervious tabs and it should work.

The class “SampleView.java” and “PlayerView.java” are the abstract classes that define some common fields/methods in each of the sample models and user models.

The four classes "ShoulderTestSample.java", "WristTestSample.java", "AdaptiveTestSample.java", and "SimDraw.java" are the four actual classes that contain the four different exercise samples. Each sample can be uniquely defined by different shoulder, elbow and wrist angles with update rates. Further implementation could include a setup page that allows user or therapist to configure new exercises easily by themselves.

Finally, the classes "PlayerPanel.java" and "WristPlayerPanel.java" are responsible for drawing the real-time skeleton of the user during the exercise. Generally speaking, they are reading the angular and coordinate data from the DataGenerator and plot the points and lines out according to these data.Other data such as the length of limbs or the width of shoulders are defined in the class "Constants.java".

5. Issues with the BeagleBoard

Originally for this project we were supposed to use BeagleBoard single board computer (we tried BeagleBoard-xMRev.B for this project) instead of traditional PC for the platform of the main application. However, after several attempts with no luck, and after thoroughly searching the Internet for answers while also asking from other professionals, we decided to stay with the PC. Detailed reasons will be discussed below.

Initially we have installed a desktop version of Ubuntu 11.10 (Oneiric Ocelot) onto the BeagleBoard. After we installed OpenNI and PrimeSense, the Kinect simply doesn't work. Then we changed to a much light-weighted version of Ubuntu, the command-line only server version of Ubuntu 10.04 (Lucid Lynx). We then installed the unstable build OpenNI v1.5.4.0 for ARM platform, the PrimeSense v5.1.2.1 from avin2. We have also tried an unofficial NITE build for ARM platform (which is found at http://www.hirotakaster.com/download/nite-bin-linux-arm-v1.5.0.1.tar.bz2). When running the example, the complier compiles but the application complains about "depth buffer corrupted". We tried to reduce the frame rate, hoping this will alleviate the stress on the BeagleBoard. But the attempt fails because it seems that NITE will only support 30 FPS frame rate. Since the fact that NITE is actually a closed-source software, we have no option but to go back to PC.

How to Get the System Working

1) Fetch all the source code of this project from Github @ https://github.com/ESE519/Kinect-Physical-Therapy/

2) Open Eclipse IDE, then choose “File”–“Import”–“Existing Projects into Workspace”, specifies the location of the source code and continues.

3) Make sure your RXTX and Java3D libraries were installed properly on the computer. For Ubuntu users, the RXTXcomm.jar is located at /usr/share/java/RXTXcomm.jar. For installing and configuring the Java3D library, please refer to the following link (which is useful): http://www.cs.utexas.edu/~scottm/cs324e/handouts/setUpJava3dEclipse.htm

4) Right click on the project just imported into Eclipse, choose “Build Path”–“Configure Build Path”, and add the RXTX and Java3D libraries into the build path.

5) Connect the Kinect sensor and XBee module to the computer, Execute the MainFrame.java as Java Application and there you are!

6) After wearing the hand gloves provided, the user can start performing the exercises.

Testing and Evaluation

Overall the accuracy of our system was decent given the hardware limitations.For all exercises, the Kinect sensor can successfully identify the user performing the exercises and capture his/her movements. If the user moves his/her arms, the skeleton representation will change accordingly in real-time.

The judgment of the system is also good. We have set a pre-defined error margin of ±10 degrees, which can reduce the misjudgment caused by the delays between reading the sample’s data and the user’s.

To compensate for the lack of Kinect’s ability of tracking the movement of human’s hands and wrists, we used IMU-3000 gyroscope. For the result, the gyroscope performs well: it is sensitive enough for tracking angular changes in real-time. For the radio thing, the throughput of the XBee modules, though not so large, is enough for this project. But for more complicated applications, a more sophistic wireless module would be more suitable. There is a delay of about 0.1 seconds between the result is received by the XBee module and the vibration of the motor. This delay, although insignificant, should be minimized afterwards.

On the other hand, the mbed microcontroller has revealed some incapability. On occasional circumstances, the mbed microcontroller freezes after some stages. In these cases, we have no other options but to reboot it. We believe this annoying mistake could be caused by the overflow of the input/output buffer of the serial communication to XBee module or because of the tasks were not scheduled properly. Nonetheless, after a thorough examination, we believe these limitations will likely be overcome in the future.

Lessons Learnt

During this intensive but full of fun project, we have learned a great deal. We have learned about how to get the Kinect sensor working on platforms other than XBOX360. Not only were we able to apply our general knowledge of microcontroller, embedded programming and wireless radios, but we were also able to apply our techniques to resolve problems every time when they occurred. Every time we ran into a problem, we spent time debugging, thinking, and researching.We have built this entire system from scratch. Overall, we enjoyed having a truly learning experience in embedded systems.

Video, Snapshots and Source Codes

All can be found under the Github of our project:

https://github.com/ESE519/Kinect-Physical-Therapy/

The demo video can also be watched online at:

http://youtu.be/m0eB3rK-nmk