HANDS ON

A project completed as part of the course:
ECEN 489
Special Topics in Sensing, Acquisition and Innovation Laboratory

By the team:
Brian Bass, Michael Bass, Yayun Lau, Shuli Li, Zachary Partal,
Narayanan Rengaswamy, Dipanjan Saha, Desmond Uzor

During the semester:
Fall 2013

Advised and mentored by:
Dr. Gregory Huff
Dr. Jean – Francois Chamberland
Department of Electrical and Computer Engineering
Texas A&M University, College Station

TABLE OF CONTENTS

INTRODUCTION

Bi-manual coordination serves an important purpose in everyday life, and in most activities coordination is easily achieved, but there is a varying degree of success when performing certain actions independently with each hand. Some researchers attribute this to the intense inter-hemispherical coupling of the left and right hemisphere of the brain.

The HANDS ON project tests the ability of the user to control the independent lateral motion of each hand while moving the other hand at the same time. One hand is moved vertically up and down with the palm parallel to the ground while the other hand is moved away then towards the body with this palm approximately at an angle of 90 degrees to the floor. Both hands move simultaneously, and the user is given a score judging how steady and precise the hand movement is. This can be used as a competitive game with the scores being stored in a database, and a comparison is made of the minimum, maximum, mean and median of the scores of various majors, genders, age groups and "dominant" hands (right-handed / left-handed).

HARDWARE

The hardware for the project is unique. Early in the project it was decided that an IMU would be used as the primary sensing device. An IMU alone though would not be sufficient. Supporting hardware would have to be developed in order for this project to be a success. This supporting hardware includes a custom PCB, wireless Bluetooth communication, a lithium-ion battery as a power source, and a 3D printed container to hold everything together.

The Inertial Measurement Unit, or IMU, is a collection of sensors that provides acceleration, orientation, and rotation data through a serial interface. Specifically the 9DOF Razor IMU was used. This chip is comprised of the following components:

1. ITG-3200 (triple-axis gyro)
2. ADXL345 (triple-axis accelerometer)
3. HMC58833L (triple-axis magnetometer)
4. ATMEGA 328 Microprocessor

Due to the sensitive nature of the sensors, accurate calibration is necessary for valid readings. Calibration is done through the use of onboard software. By following the calibration procedure the inherent error in the hardware can be determined and offset values can then be generated.

Wireless communication was provided by an inexpensive serial Bluetooth chip. The Bluetooth device was configured to a baud rate of 57600 which allows for the TX and RX pins of the IMU and Bluetooth devices to be connected directly together. Power was provided by a small Lithium-Ion battery. The battery connects directly into the IMU. The IMU's voltage regulator is capable of providing enough current to power its own circuitry as well as the Bluetooth device.

The PCB was designed to connect all components together. First a schematic was designed that connected all the components together. Next a layout was designed that placed all the components and traces on a model of the physical board to be printed. Once the schematic and layout were designed it was then milled. The schematic for the PCB can be seen below in Fig. 1. The PCB was designed using the Eagle software and milled in the Huff Research Group lab.

The 3D printed container was designed to hold all the components in place and provide a place for a strap that is used to attach the device to a user's hand. The container was designed using the OpenSCAD modeling software and printed using a Makerbot in the Huff Research Group lab. The PCB was attached to the 3D printout with printed pegs to ensure it did not wobble from the user's hand during trials, and secured to these pegs with super glue.

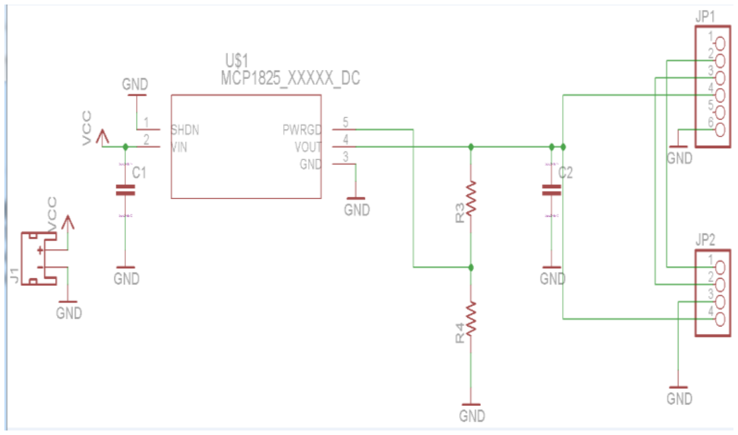

Figure 1. PCB layout.

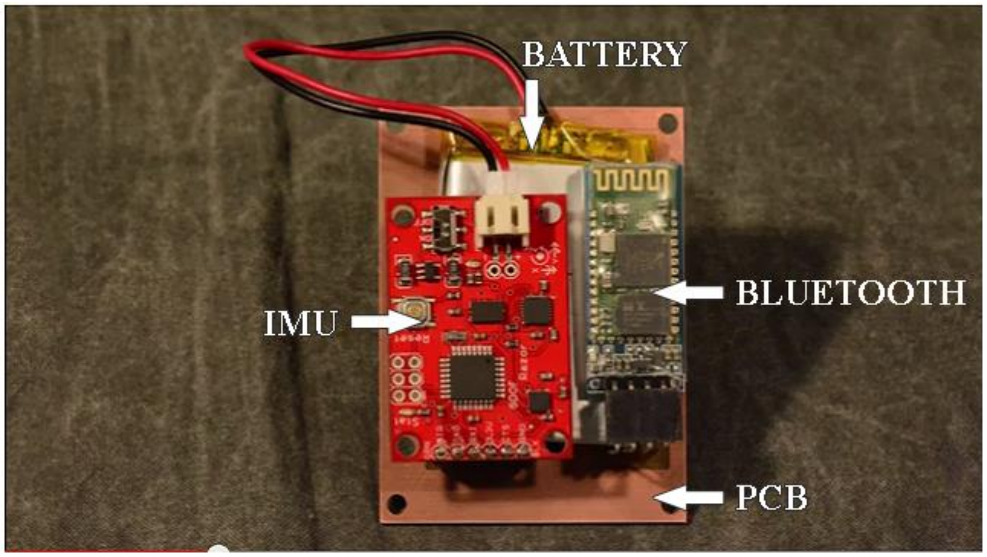

Figure 2: Complete module

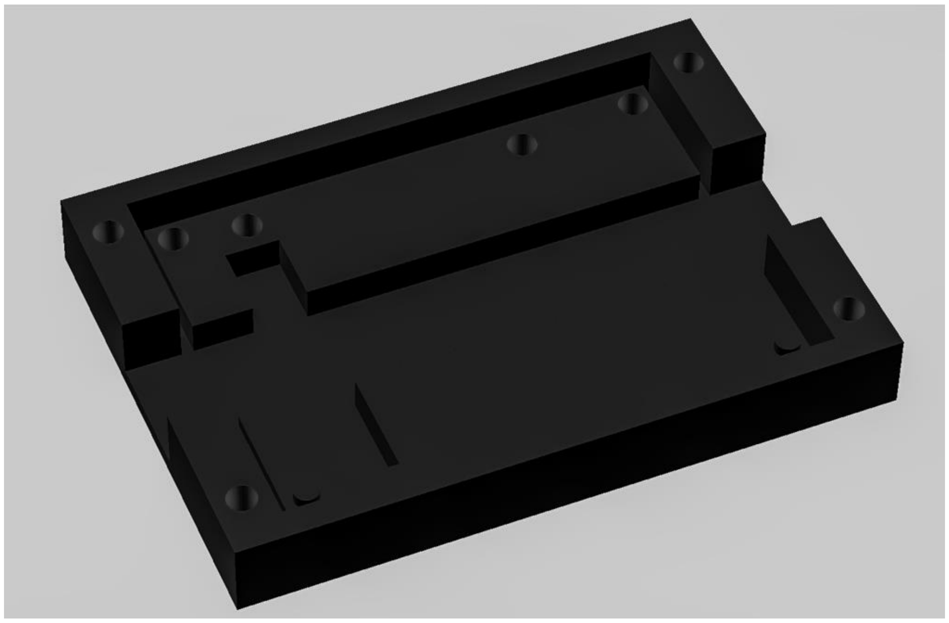
Figure 3. 3D printout. The velcro strap goes across the bottom

A second set of hardware components was used to determine the distance a user's hand has traveled. Two ping sensors were placed in front of each hand. The ping sensors filtered their combined readings to determine the most accurate reading. The ping sensors were also connected to the PC using Bluetooth.

Because the IMUs and ping sensors used Bluetooth to connect to the PC, they each used a serial connection. Therefore, the C++ code used a Reader class to connect to the Serial ports and acquire data. The Reader class is capable of reading custom formatted data from both the IMUs and the ping sensors. In addition to this, the Reader class creates its own thread so that it can acquire data independently of the main process.

DATA ACQUISITION

To get anything useful from the hardware, the correct software needs to be put in place to take advantage of the components being used. A scoring system is used that relies on the acceleration and orientation (i.e. roll, pitch and yaw angles) of each hand; this makes it necessary to have accurate measurements available. More importantly, the scoring is done relative to an orthogonal coordinate axis with respect to the earth's floor. The raw outputs of the sensors are proportional to m/s^2 for accelerometer, deg/s for gyro and Tesla for magnetometer if they are calibrated correctly to remove noise and the sensors inherent measurement error offset caused by differences in environmental conditions, amongst others.

Sensor Fusion and Calibration

The I2C protocol is used for communication between the various sensor chips on the IMU. Since each chip has a separate address on the I2C line, it is possible to communicate with all three of the sensors using only two pins. This allows for easy transfer of data using the Bluetooth communication modules. Each sensor is setup to sample data at 50 Hz. The 10 bit accelerometer is set to its full resolution of 4 mg per least significant bit with a range of +/- 2g. The magnetometer is set in continuous sampling mode, and the gyroscope range is set at +/- 2000 ˚/s.

Sensor calibration is very important to IMU performance. The sensors are very touchy and if not calibrated will produce results unusable for our application.

The accelerometer is calibrated by turning each axis to align with the earth's gravitational field. The IMU has a certain range, so by finding the value of gravity (one g), it can determine the offset for each axis from the true value and then subtract it out when outputting data. The magnetometer is calibrated in a similar manner by aligning each axis with the magnetic north pole of the earth's magnetic field. The gyroscope calibration is a noise measurement and is done by leaving the unit still for an amount of time and recording the average value on each sensor. This noise value is then taken out of the gyroscope measurements. All calibration data was used in conjunction with firmware provided by the Razor IMU homepage. The firmware was loaded onto the IMU microprocessor and applied the offset and noise data in its routine to produce accurate measurements from the IMU sensors.

Local to Global Frame Conversion

There is one IMU attached to each hand of the user, and the IMU sensors give readings relative to the "local" reference frame, i.e. the set of axes fixed to the hand. However, the hand is neither at rest nor moving at a uniform velocity and hence constitutes a non-inertial reference frame. For scoring, we need the acceleration components and also the orientation angles relative to the "global" reference frame, which is inertial. The local frame can be related to the global frame by means of a rotation matrix or a Direction Cosine Matrix (DCM) which can be updated using the gyro readings. In addition, we have to take care of a couple of issues – the accelerometer data is noisy; while the gyro data suffers from the phenomenon of 'drift' (i.e. the bias changes slowly with time instead of remaining constant). All these lead to the following algorithm that will fetch the "global" acceleration and orientation for each hand once the three sensors (accelerometer, gyro and magnetometer) are calibrated:

(1) Compute the initial pitch and roll from accelerometer readings by using following equations:

pitch = atan(Ay /Az) * 180) / PI
roll = atan ((Ax / Az) * 180)/ PI
where Ax, Ay, Az represent the acceleration vectors on the x, y , and z axis respectively.

(2) Use the initial pitch and roll along with the magnetometer readings to compute the initial yaw.

(3) Using a "yaw -> pitch -> roll" sequence, initialize the DCM.

Every time a new set of readings arrives for each hand, perform the steps (4) through (8).

(4) Filter the accelerometer data.

(5) Using the filtered accelerometer and magnetometer data, obtain the drift-corrected gyro data.

(6) Use the drift-corrected gyro data to update the DCM.

(7) Extract the new roll, pitch and yaw angles from the updated DCM.

(8) Use filtered accelerometer data and updated DCM to find the components of acceleration as seen

from the inertial “global” frame.

Kalman Filter: The gyroscope has a drift and sometimes the values returned are completely wrong. The accelerometer, on the other hand, returns a true value when the acceleration is progressive but it suffers from vibrations. Usually a mathematical filter is used to merge the two values, in order to have a correct value -the Kalman filter. This is the best filter we can use, even from a theoretical point of view, since it is the one that minimizes the errors from the true signal value. The filter needs to be able to calculate the coefficients of the matrices, the process-based error, measurement error, etc.

Direction Cosine Matrix: It is important to realize the IMU will provide data in a moving frame, but what we really need are accelerations in a base, or global frame of reference. Through linear algebra, transformation via rotation matrices will resolve this issue.

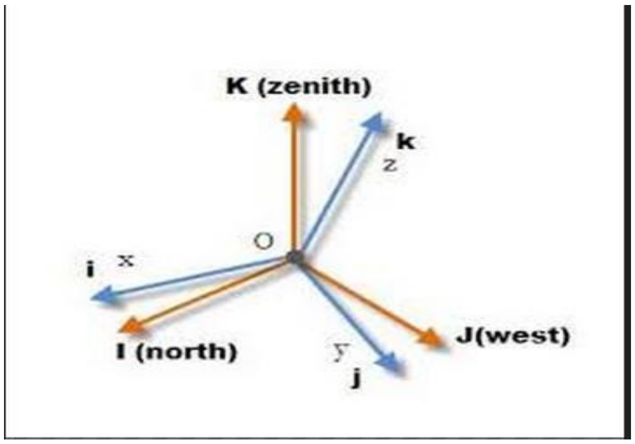

Figure 4: Body and Global Frame Vectors

The global coordinates can be represented by the vector I, J, K while the body coordinates can be represented by the vectors I, j, k. A complete set of the body coordinates is represented by the matrix form:

$$[i\,,\,j\,,k]=\begin{bmatrix} I.i & I.j & I.k \\ J.i & J.j & J.k \\ K.i & K.j & K.k \end{bmatrix}=DCM$$

Converting between reference frames just involves multiplication by the DCM to get the associated reference frame. The DCM above is multiplied with the local frame to get the respective global coordinates. The inverse could also be performed for a global to local conversion.

The gyros are used as the primary source of orientation information. Integrating the differential kinematic equation that relates the time rate of change in the orientation of the IMU to its rotation rate, the present orientation can be found. Recognizing that numerical errors, gyro drift, and gyro offset will gradually accumulate errors in the DCM elements, we use reference vectors to detect the errors and a proportional plus integral negative feedback controller between the detected errors and the gyro inputs to dissipate the errors faster than they can build up. The magnetometer is used to detect yaw error, accelerometers are used to detect pitch and roll errors.

The process is shown schematically in the Figure below:

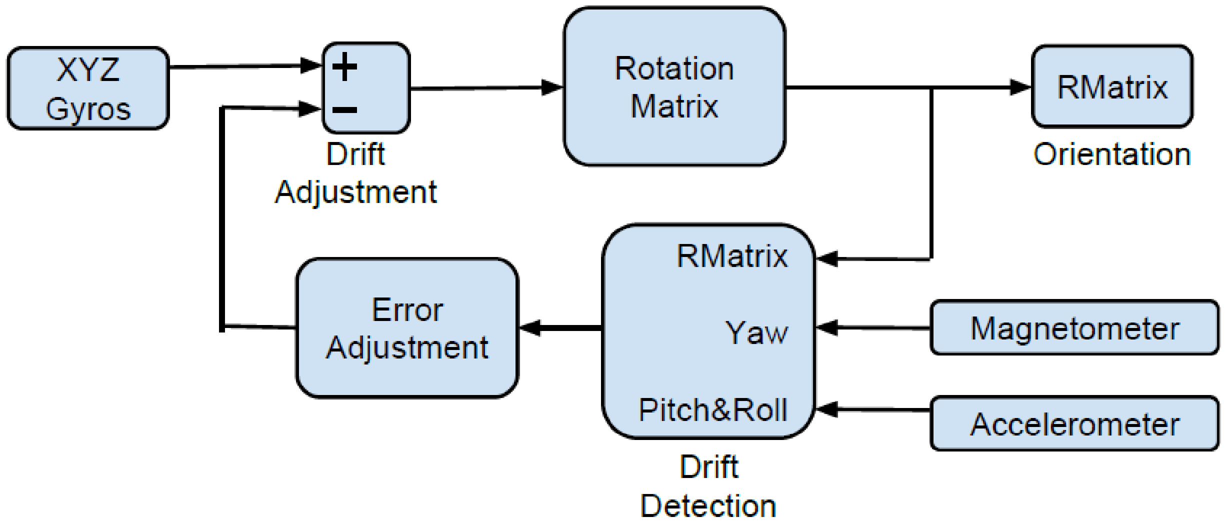

Figure 5. Schematic of Rotation Matrix Calculation

Scoring

The scoring algorithm is done with a simple skee-ball type system, broken down into two parts. The aim of the game is to move one's hands in a straight line parallel or perpendicular with the ground, while keeping the plane of one's hands parallel or perpendicular to the ground. As such, one is scored based on the direction of their acceleration and the angle at which their hand is tilted. The angle is taken as the angle between the normal vector of the IMU and the axis of motion. The frame is chosen so that the Z axis is the direction of motion for each hand.

Each user gets a certain score for every reading that is brought in. As part of the processing from raw sensor values to global frame values, three acceleration components and angle from the Z axis are computed. Simple geometry is then performed with the acceleration components to determine the direction of movement. If movement is directed purely along the Z axis, one gets full points for that reading. However, if they are moving at more than a ten degree angle off axis, they lose percentage points. Further points are lost beyond 20, 45, 60 degrees, etc… Additionally, double the noise floor of the accelerometers is taken as zero acceleration. Zero acceleration receives zero points to prevent users from just leaving their hands still.

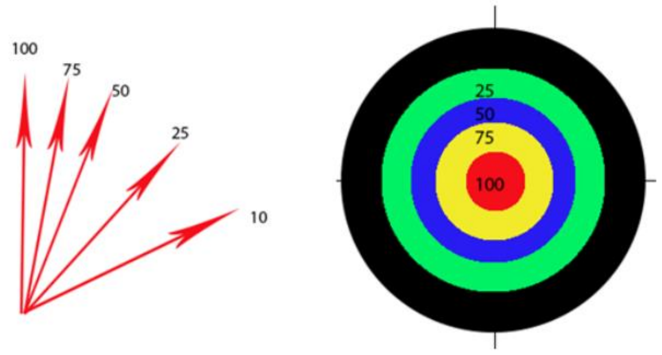

Figure 6. Depiction of Scoring Strategy, Picture on the left is the normal vector of the IMU in the vertical hand motion, and picture on the right is looking down the axis at the acceleration vector

Similar percentage values are used for the angle from the Z axis of each reading, so there is an acceleration component and angle component to each score for each reading. These components are multiplied to get the total score for that reading. Scores are also multiplied by the time interval between samples to normalize for the amount of samples taken over a whole session. There are separate readings from each hand, so each hand gets a separate score for each reading. Since the goal of the project is to test some form of bilateral coordination, we multiplied the scores of each hand so that keeping one hand still to try to perform better with the other hand would result in a total score of zero.

USER INTERFACE

To put the project together into a complete working product, Qt creator was used to develop a graphical interface. Qt is a framework used to create applications using C++. This project uses Qt to give an effective GUI for the experiment so that the user can initially give some of is demographic information, then view his performance real-time, and finally, see an analysis of his ability to coordinate his hands with respect to others based on major, gender, age group and dominant hand. The user can also see how well each of his hands performed in each of the two runs and also overall. The flow of the GUI is as follows:

When the application is run, it connects to the PostgreSQL database for data storage and analysis. Once that is successful, it connects to the four Bluetooth devices (one each for the two IMUs and one each for the two ping sensors) through their respective COM ports (since this application has been developed for the Windows platform). Once these stages are crossed successfully, the screen for entering demographic information is displayed. Once data is entered and "Ok" is clicked, some basic data validation schemes are run. Data has to be re-entered if found invalid. After that, the first "graphing" screen is ready. This will be for the experiment where right hand moves in the vertical direction and the left hand in the horizontal (front-back). Once the user ready, the "Start" button triggers the application to receive data from the IMUs and Ping Sensors and also plot them. There are 3 plots for each hand. The top most linear plot shows the time series acceleration over the course of the experiment that is not on the perfect axis (X and Y axis in our coordinate frame). Ideally, one must see a straight line with no perturbations. The circular plot below this gives an idea about the magnitude of error the hand is committing at the current moment in time. Larger errors give a larger plotted circle. Both these plots are of the acceleration data in the X & Y axes because the hand is always supposed to move in the Z axis. The bottom most linear plot is to visualize the position of the hand during the motion. For each cycle, the dot in the graph will oscillate between left and right to give the feel of hand motion. The ping sensor data is used to plot this. A 20 second timer counts down and on completion, displays the score for this run. After running the application with the right hand moving up and down, the hand motions are switched so the left hand movement is vertical and the right is horizontal.

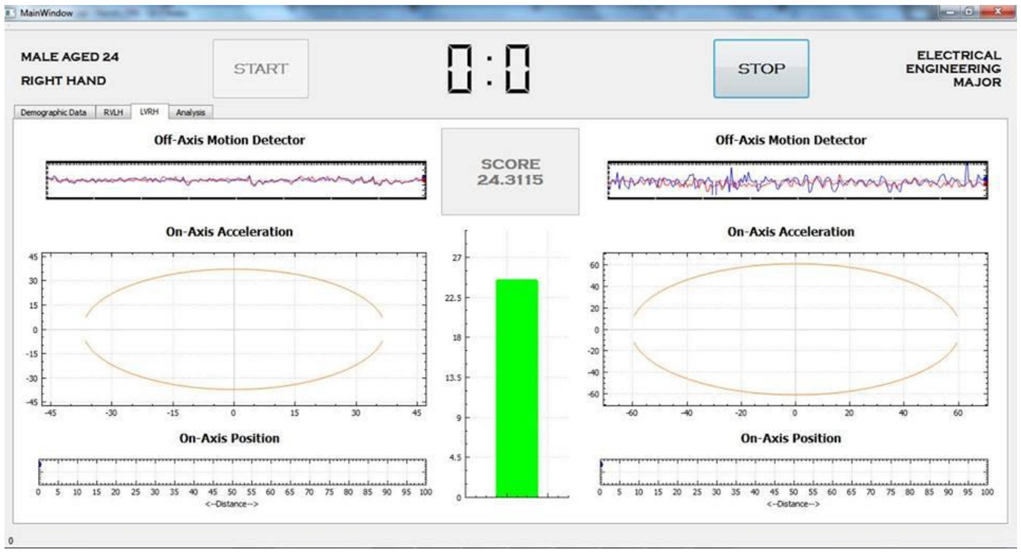

Figure 7. Sample graphical depiction during a user trial

The components shown include:

1. Off-axis motion detector to shows activity when the users hand is not aligned properly.
2. Off- Axis Acceleration which expands to show the magnitude of the hand acceleration, a more instantaneous feedback measure than the time-series data
3. Score graph to show the users current score
4. On-axis Position to show the current position of the hands.

Database Storage

Immediately upon the completion of the user trial, the QT code exports a total of nine scores to a PostgreSQL database: the right hand score for the first and second sessions, and total right hand score, the left hand score for first and second sessions, and total left hand score, and the total score for each session and total score overall.

The database is then queried and the mean, median, and maximum scores for the user’s major, gender, hand dominance, and age group are returned. The user will then see how his score compares against all of these groups. Additionally, he can see how the scores of his separate hands compare to determine if he was favoring one hand or the other, or maybe he is just better at the vertical motion than horizontal. All of this is displayed in QT instantly upon completion of the trial.

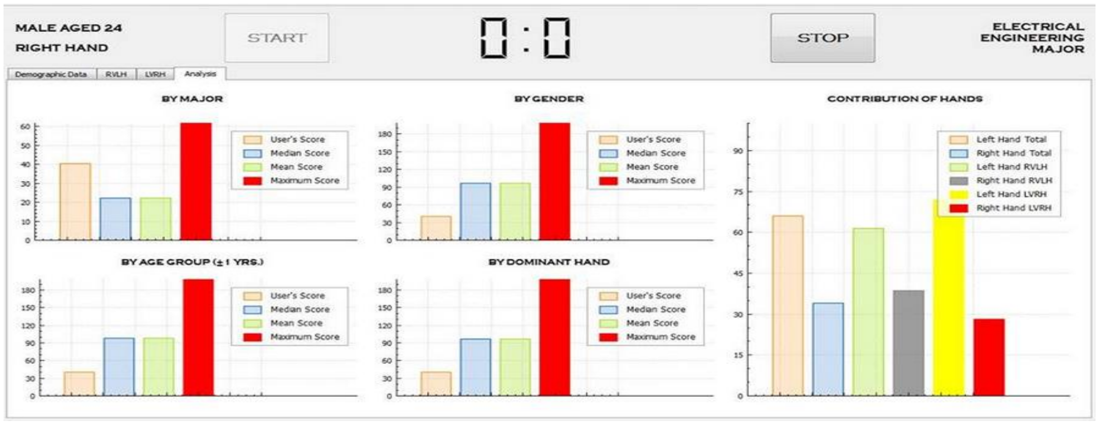

Figure 8: Data Comparison

More specific groups of contestants can also be compared using the search function in C++ on a separate station. Here, one can search for as many scores as they want, searching for any combination of none to all of the demographic categories: major, gender, age, and dominant hand. Minimum, median, maximum, and mean scores are then compared for each of the queried groups, and histograms are plotted (showing mean and standard deviation as well) for right, left, and total scores for each of the queried groups. The data shown in this case is similar to the above with the addition of a histogram and standard deviation calculation in addition to the ability to search for any set of data.

CONCLUSION

In the future, the project could be integrated into a more developed study on hand coordination or could be used for a number of visualization projects involving IMU's using QT. One would really need to develop tests for multiple sets of movements to really determine if someone had good independent coordination rather than just a lot of practice at one particular movement (I think the team got better at these movements over the course of the last couple weeks). Another option would be to have them to move based on external stimuli like flashing lights rather than a controlled continuous motion.

One future improvement would be to make better use of distance measurements in scoring and analysis. This would require more ping sensors to gain accurate data, since each sensor only covers a narrow beam pattern angle and more sensors would allow better coverage even when the user strays from the desired axis. The user could be instructed to move his hand at least 2 feet say on each rotation, and if he does not meet this requirement, he gets no points for that cycle, or diminished points. This would also provide some reward for speed. Additionally, better drift correction for velocity measurements could be done, allowing for velocity vectors to be used in scoring rather than acceleration. This would be desired because the direction of velocity is more intuitive to users than acceleration.

This project incorporated many of the tools we learned in class earlier in the semester. It uses sensors such as accelerometers, gyroscopes, magnetometers, and ping sensors. I2C communication is used on the IMU to coordinate data from all three sensors as well. C++ was used in all programming applications. PostgreSQL databases were used for all data storage. Qt creator was used for visualization. All of these tools were taught in class, and some others like PCB layout, 3D printing file design, and some mathematical tools were learned along the way during project development.