BARR-SYSTEM

An intelligent device that keeps your health on track!

Design of Engineering Systems II

Bangaly Diane, **A**rjun Pawar, **R**ushiraj Parikh, **R**an Wei

Executive Summary

BARR-System is a multi-layered service that helps patients throughout their knee recovery process. A patient is now able to wear a one-size-fits-all device to monitor their performance throughout recovery. The device collects 3D motion data and displays it in real time on a user interface. With the assistance of visual aids and an alert system, the user is able to correct their motion during each exercise. Such recovery assistance helps patients improve their fitness level while also reducing the amount of visits needed to their therapists.

Table of Contents

Introduction

Physical therapy is a preferred post knee injury rehabilitation solution. Repeated therapy visits can be a hassle for patients with limited mobility. It is reported that only 30% of patients who receive outpatient physical therapy services attend all visits that their insurance company authorizes; and only 35% of patients adhere to their plan of care (WebPT, 2018). This inconsistency is caused by cost of travel, patients' sense of posture awareness while performing exercises without supervision and more. A lack of medical knowledge and reliance on medical professionals causes difficulty for patients in ensuring optimal form of exercises or precise monitoring of rehabilitation. The unique solution to this current practice in the realm of physical therapy is BARR-System. The innovative technology is a wearable device used to monitor patient knee form while performing specifically assigned exercises during their rehabilitation period. The smart device is able to track movements around the knee and classify each movement as either a good or bad form displayed on a user interface with green or red colors respectively. BARR-System has been able to create a disruptive innovation by reverse engineering the current health tracking systems in the market for a physical therapy application.

Technical Approach

In order to translate ideas into a final product, a sensor was needed to collect 3D motion data. Once the data was collected on a microcontroller, it needed to be transmitted onto a computer. Whence it can be analyzed, stored and plotted onto an interface.

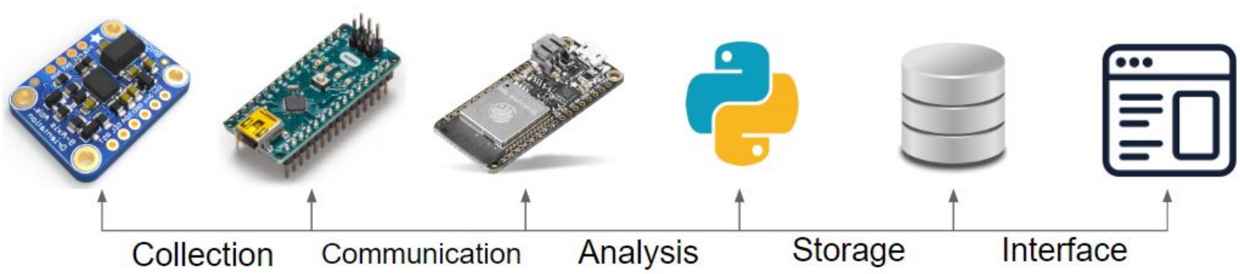


Figure 1: Approach Layout

Hardware Overview

BARR-System can incorporate two versions of a device where the standard edition requires data communication over a wire and the deluxe edition via Wi-Fi. In order to meet software integration deadlines, the standard edition of the device was used for data collection and software testing.

The final hardware layout of the standard device consists of two Inertial Mass Unit (IMU) sensors called BNO055 connected to a TCA multiplexer which is connected to an Arduino Nano. All power and ground wires are mounted on a custom-made breadboard. Figure 11 shows the wire schematic of all components. Each hardware component has its own custom-made casing that slides along the velcro strap.

The IMU sensors record the gyroscope values in X, Y, Z direction. The microcontroller cannot differentiate between the two sensors, so a multiplexer is used to read values from different addresses on an I2C bus. The Arduino Nano reads values from the multiplexer and outputs a string of values from both sensors onto the serial monitor via USB cable.

The final hardware layout for the deluxe edition consists of two BNO055 sensors connected to a TCA multiplexer which is connected to an Arduino Nano which is also connected to an ESP32 featherboard to transmit the data over Wi-Fi. All power and ground wires are mounted on a custom-made breadboard which is connected to a LiPo battery via powerbooster. The powerbooster allows the same 3.7v battery to power the 3.3v featherboard and 5v breadboard which powers the rest of the components.

- Startup procedure for standard edition

1. Plug in USB cable and run from user interface

- Startup procedure for deluxe edition

1. Plug in power cable for featherboard
2. Search and connect to Wi-Fi
3. Plug in power cable for breadboard that will power all other components

NOTE: It is vital to start up Wifi shield before everything else because it needs certain amount of current (jolt) just to start up which is not achievable after connecting everything.

Knee Angle

Movement can be defined in the the 3 planes shown in Figure 2. The primary knee angle is defined as the angle between femur (thigh) and tibia (shin) in the sagittal plane. However, knee movement does occur in the transverse plane as well. We name the knee angle with rotation axis perpendicular to the sagittal plane angle Z, and rotation in the frontal and lateral directions angle X and Y.

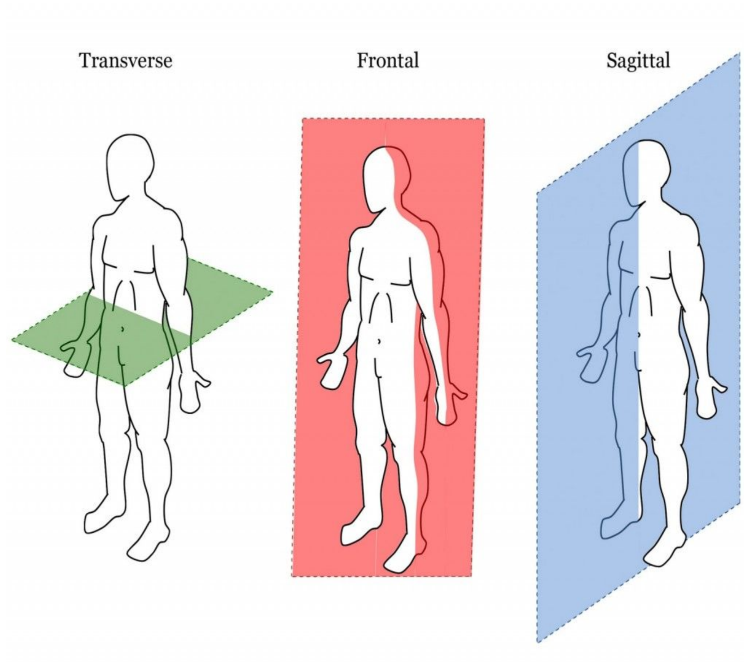


Figure 2: Human Motion Definition

The angle's mentioned above can be measured by 2 IMU sensor mounted on to the side of one's leg, with one IMU sensor worn above the knee joint and the other below. One has to make sure all 3 axis of the 2 IMU sensors line up. The change of knee angle is equivalent to the difference between the 2 IMU sensor measurements. For consistency we define knee angle change ɑ (degree/second) as gyro1 - gyro2, where gyro1 is the gyroscope measurement by the thigh sensor, and gyro 2 is the gyroscope measurement by the shin sensor.

Since gyroscope only measures change of angle, in order to obtain measurement of knee angles, we use trapezoidal rule to integrate the angle's change with respect to time. Real-time integration is done using the following formula:

$$\theta_{t+1} = \alpha_{t+1} * \Delta t + \theta_t$$

where θ is the cumulative change of knee angle, and ɑ is the instantaneous change of knee angle.

For proper functioning, the user is required to initialize the device from a standing position, at which we consider the knee angle as 0 degree. With this, the cumulative change of knee angle is equivalent to the absolute knee angle measurement.

Analysis Method

Overview

The goal of the analysis module is to classify good squatting technique from bad squatting technique. The standard for a good squat is to not have knees going over the toes in the forward-backward direction. One also doesn't want knees to push out in the lateral direction too much, which also causes misalignment. A common mistake due to muscle weakness and instability is knees propensity to cave inwards. This creates a position called valgus. Valgus can cause severe damage to the knee joint, and over time the pain spreads to the hips and lower back. We focus on creating measures for the knee overshoot in the frontal and lateral direction and the severity of valgus. A demonstration of valgus is shown in Figure 3.

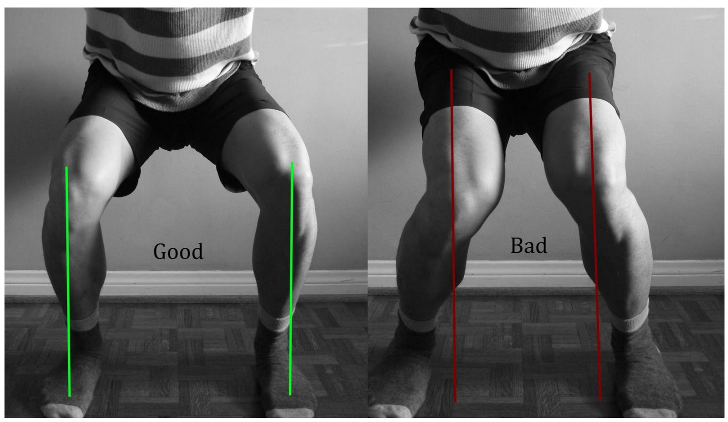


Figure 3: Valgus Knee Position

Data Preprocessing

It takes roughly 3 to 5 seconds for the computer to connect to arduino. We ask the user to not move during the connection and after the exercise. Roughly 20 data points are collected from the IMU sensors every second. These points form a complete trajectory of the angular change in knee. We want to extract the movement features mentioned above from these signals, and let the computer learn to distinguish good data from the bad data.

We take a 3 moving average of the data. The time-series plot from one set of squats of one participant is shown in figure 4. The data shows clear periodicity, which is caused by repetition of the squatting motion. Ideally, after each squat, the y value should go back to 0. However, due to the sensor noise, data tend to shift away from 0. Other factors such as speed of motion may also contribute inaccurate data, however, this is not investigated in this study.

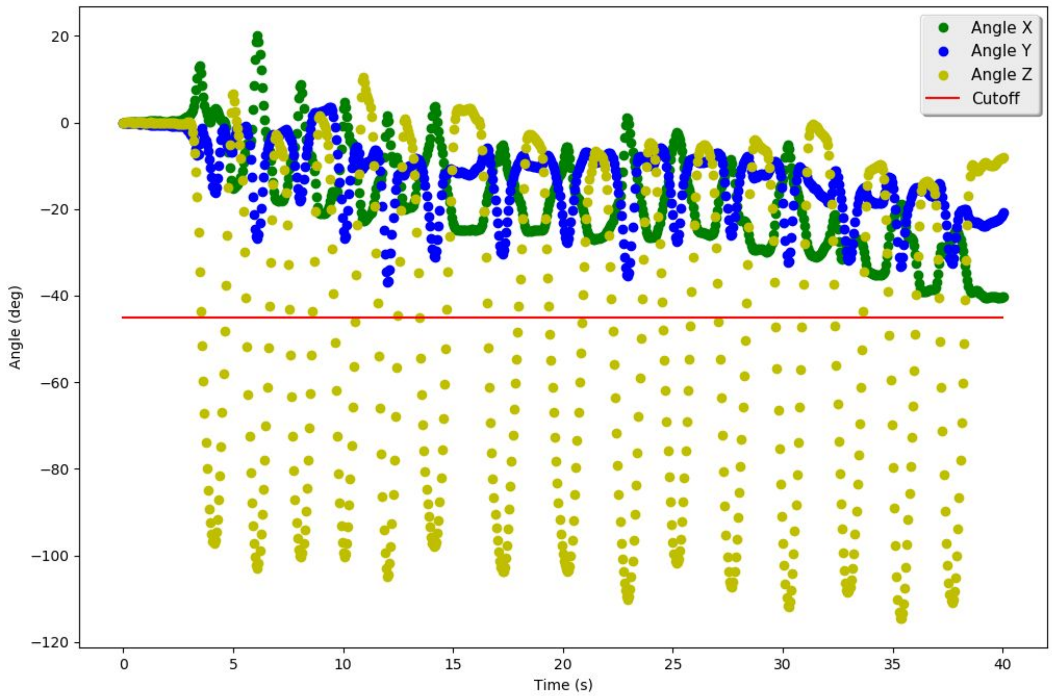


Figure 4: Raw Data Plot

Identify Period

We assume that every user is able to squat more than 45 degrees. Practically, it takes more than 0.3 seconds to rise from 45-degree position, stand up, and drop to another 45-degree position. Therefore, any 2 45-degree points that have more than 0.3 second difference marks a new period. For each period, we fit a line to the first half of the points, and the last half of the points. If the first half has a negative slope, or the second half has a positive slope, the data has a concave up shape; otherwise, the data has a concave down shape. Depending on the shape, we can find the maximum or minimum point, which is close to the absolute maximum knee angle.

For each local extreme value, we go left and right respectively until the absolute value of angular acceleration is less than 15. These 2 points become the starting point and the ending point of each squat. Time difference between the start time and the time of the peak in Z is the time to go down, and the time difference between the peak and the end point is the time to go up. The time between the previous end point and the next start point is the time between repetitions.

Since the device measures the change in angle instead of angle, and we clearly see that data drift away and the drift becomes more significant overtime, a better measure of the knee angle is to subtract the extreme value of each period by the y value of the starting point.

Valgus and Pause Time

The severity of valgus can be described by the amount of movement in the opposite direction to the main path. Since valgus only happens in the lateral direction, we only examine the y angle data.

Take the first repetition of squat for example (Figure 5), the arrow is clearly pointing downward. However, the data curve towards the opposite direction at 3.5 seconds. A clean peak would look like Figure 5 right. We use the peak detection algorithm to find the local maximum points and minimum points (red cross in Figure 5) of each period and remove the extreme point (green cross in Figure 5) that we calculated from above. Any 2 points whose difference in y value is less than 2 degrees we call it a pause point. After removing the pause points, we take the difference between the y values of the remaining local extreme points again and call these values valgus. The final measurement of valgus is the maximum value of all the valgus measurements. The final pause value is the accumulated addition of the pause times.

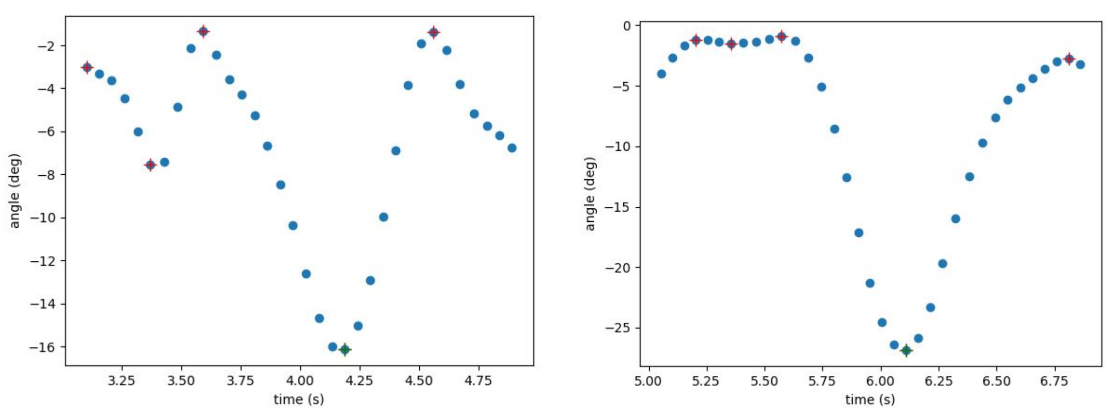


Figure 5: Valgus (left) vs. Pause (right)

Features

The features extracted from each repetition of squat are the following:

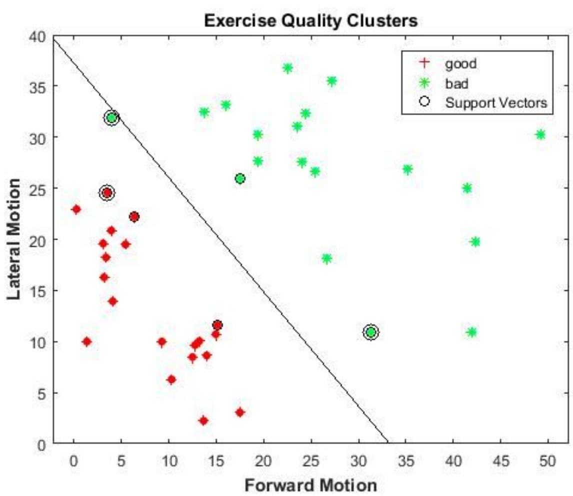
- Maximum angle in X, Y, and Z axis: reading of angle (degrees) in X, Y, and Z subtracted by offset of each repetition
- Time down: time (seconds) to go down

- Time up: time (second) to go up
- Time between: time (second) between repetitions
- Valgus: reading of angle (degree) in the opposite direction of the main movement
- Pause: accumulated time (second) of pause

Support Vector Machine Model

Since all the good data is collected on healthy participants whose squat technique is good but arguably not optimal, and our bad data is collected on healthy participants who try to simulate the common squatting mistakes, we can neither reflect the data feature of the optimal squatting technique nor that of actual recovering knee patients. In this case, Support Vector Machine (SVM) becomes an ideal algorithm for our application because it finds the best decision boundary by finding the line or surface that separates data points from different categories with the maximum margin. And example of SVM is shown in Figure 6.

We collected 70 repetitions of good squats and 25 repetitions of bad squats, out of which we selected 60 good squats and 20 bad squats for model training, and 10 good and 5 bad for model validation. Our prediction accuracy on the validation data came out to be 93 percent, and the confusion matrix is show in Figure 6.



		Actual	
		Good	Bad
Predicted	Good	10	1
	Bad	0	4

Figure 6: SVM Example (left) and Confusion Matrix (right)

Analysis Results

In order for the user to easily understand their physical performance during each exercise session, all measurements and technique classification results were aggregated to 2 final outputs.

Each repetition of squat is given a grade of 1 or 0 for good or bad. If more than 80 percent of the repetitions during the session is good, the final result is good and it is shown in green; otherwise, it is bad and shown in red. To encourage the user to activate range of motion, only the minimum knee angle is displayed in the result page. If the user is able to squat more than 85 degrees, the result is displayed in green, and otherwise red.

Operation Instructions

In order to operate the system, the script titled "gui.py," must be ran in order to display the user interface. Doing so will display the information page of the user interface as shown in Figure 7 below. Here, the user is prompted to enter in their information in the entry fields. After filling in the necessary information, the user should click the 'Submit' button. This button aggregates the user's data over to a SQL database. This feature gives the therapist the ability to go into each one of their patient's registration and view each session for a particular exercise. They are then able to see how each patient is progressing with a particular exercise to ensure a more complete recovery.

Figure 7: Information Page

The next page of the user interface is shown in Figure 8 below. This page deals with data collection during the exercise being performed. Here a GIF of proper motion as well as a brief description are displayed for the user to reference. Once the user clicks 'Start Data Collection' button the sensors on the system are triggered. The user should then wait until the terminal starts to display incremental counts to begin the exercise. After they have finished completing their assigned number of repetitions, the ‘See Results’ button will bring them to the final page.

Figure 8: Exercise Page

The final page of the interface displays the user’s results for that particular session and is displayed below in Figure 9. This displays a new graph each session showing the user’s squat depth, stiffness and instability throughout their exercise. This graph gives the user the ability to see how long they are able to endure in order to maintain good form throughout their session. The user’s minimum squat depth and their form classification for that session are displayed as well. Once finished with their session, the ‘End Program’ button should be clicked to close the interface and BARR-System can be removed.

Figure 9: Results Page

Risks and Testing

Initially, BARR-System utilized a generic sensor called MPU9255 and an Arduino Uno. Testing the sensors on moving mechanical apparatuses like a pendulum and a toy car tracks revealed the high sensitivity and low noise reduction capability of the sensor. Whence, quantifying the problems and establishing required specifications for the hardware led to a justified upgrade of the IMU. The BNO055 IMU sensors originate from Adafruit, a trustworthy company that offers inbuilt kalman filters inside the sensor to reduce external noise and sensitivity. Moreover, quantifying the size of the arduino code helped identify how much memory it requires on the microcontroller. Thence, the Arduino Uno was replaced by its smaller alternative; Arduino Nano.

The following list summarizes the layout iterations of the critical components of the device.

1. One MPU9255 sensor with Arduino Uno
2. Two MPU9255 sensor with multiplexer and Arduino Uno
3. Two BNO055 sensors with multiplexer and Arduino Nano
4. Two BNO055 sensors with ESP32 Featherboard
5. Two BNO055 sensors with multiplexer and Arduino Nano with ESP32 featherboard

The iterations personify learning how IMUs work, testing a proof of concept, enhancing the hardware for more accuracy via smaller size, testing reliability of wireless communication via eliminating intermediary components, and delivering a robust product that ensures accurate data with reliable data collection and communication.

The risk commonality vs severity matrix in Figure 10 helps visualize the possible issues that BARR-System might face. Majority of issues root from hardware failure; from the Wi-Fi components of the deluxe edition to be specific.

Possibilities	Insignificant	Minor	Moderate	Major	Critical
Almost certain 80-100%	Varying Squat Time	Tired User			
Likely 60-80%				Featherboard failure	Wi-Fi Disconnection
Possible 40-60%	Abnormal Data Handling	False alarm	Abnormal Data Pattern		Battery Dies

Unlikely 20-40%	Angle offset	Wire disconnection	Logic converter failure	MPU Failure	Sensor failure
Rare 0-20%	Frozen User Interface	Corrupted data	Multiplexer failure	Power booster failure	Arduino Nano failure

Figure 10: Risk Severity Matrix

The arduino code for the device has built in warning system which would stop outputting data and send out an alert identifying which particular sensor has been disconnected or has malfunctioned. Properly soldering the wires can severely reduce the risk of disconnection. Proper charging of the battery can reduce the risk of battery leakage or short circuiting. Powering the Wi-Fi featherboard before anything else would ensure that the proper jumpstart current is delivered for the Wi-Fi module. The high baud rate ensures faster transmission of data and reduces the impact of any corrupted data which is extremely rare. The twisting of the wires and taping all exposed metallic parts reduces electromagnetic interference and reduces risk of short circuiting.

To ensure that the entire system is working properly, multiple users were asked to strap on the standard device, log into and navigate through the user interface, and confirm that the analysis results print updated graphs. If there was a hardware issue, the startup timer would stop. If there was an user interface issue, the GUI would freeze. The analysis module is currently not able to process more complicated motions than what it is currently designed for; however, the reliability of its current application is experimentally proven. By establishing a delay in the device connectivity, we minimize connectivity issues.

Economic Analysis

The standard and deluxe edition of the device requires components that cost $85 and $135 respectively. The most expensive component on each version is the IMU sensor at $32 each. If the product is scaled to put into production, the price can decrease severely for parts not made by adafruit. For example, buying unofficial Arduino Nanos or cheap USB cables in bulk could reduce manufacturing cost. Adafruit is the primary supplier of the multiplexer and the sensor. The quality and reliability of their product justifies the cost but it might be possible to get a price quote for lower estimate of a bulk order.

Majority of components have open source schematics so it would be possible to combine some component into a self made PCB which can be manufactured in China. However, time restrictions and the return on investment analysis revealed that purchasing unofficial Arduino Nano in bulk is a more efficient solution than making custom-made PCB.

Competitive motion tracking products in the market like Xsens utilize sensors that cost at least $430. BARR-System can battle such product by offering a more affordable solution for a specific application.

Conclusion

BARR-System is a device that has been able to replicate innovative motion tracking systems in the current market tailored for a physical therapy application. The device has the capability to recognize motion patterns and categorize them as either good or bad flexion form, accomplishing its initial goal. The device overall has proven a truly remarkable problem solving concept which is open to endless development opportunities for sustainability purposes. BARR-System can be further improved by providing the following additional capabilities:

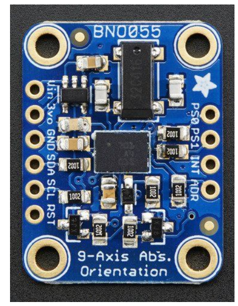
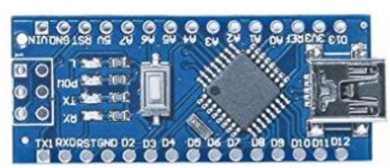
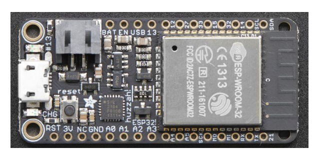
1. Better hardware capabilities
2. Reliable wireless communication
3. Customized user interface for physical therapists
4. Accounting for additional exercises

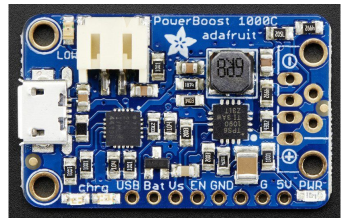
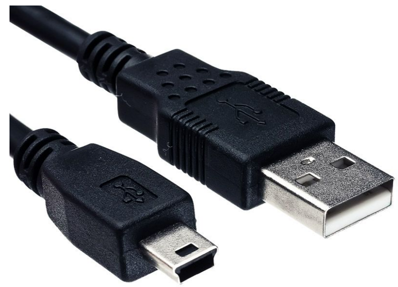
The devices could proves to make a significant difference in the market after the necessary upgrades, but with limitless opportunities in the realm of physical therapy, BARR-System has the potential to be the answer to all gait motions of the body.

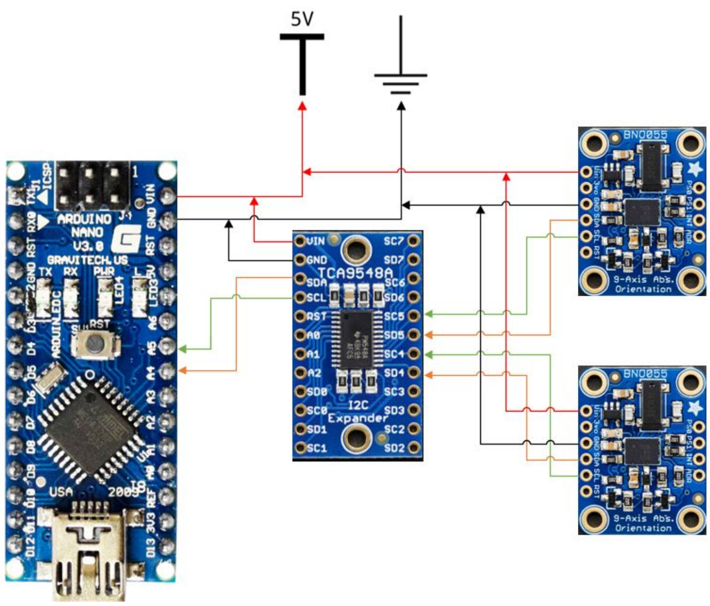
Main Contributions

- **Bangaly Diane:** Database creation, control limits, real time simulation testing
- **Arjun Pawar:** Data reliability testing, user interface creation and integration
- **Rushiraj Parikh:** Hardware selection/purchasing/assembly/testing, custom-made component casings, Arduino reading module, Matlab SVM analysis, Wi-Fi configuration/testing, data collection, flexion angle analysis
- **Ran Wei:** Arduino reading module, database entry and fetch functions, matlab analysis sketch, final analysis module, final component integration

Appendix

Table of Components		
	Item Name/function	Item Image
1	BNO055: IMU sensor records values	
2	TCA9548A multiplexer: allows microcontroller to read from multiple sensors	
3	Arduino Nano: microcontroller	
4	Adafruit ESP32 Feather: Wifi shield	
5	Adafruit 2500mAh lipo battery: power source for deluxe edition	

6	Adafruit powerboost 1000: power splitter for deluxe edition	
7	USB to Mini USB cable	



	5v Power
	Ground
	SCL
	SDA

Figure 11: Wire Schematic