Personal Portfolio - Nicholas Gurnard Mechanical Engineering Student

Nicholas (Nick) Gurnard
Undergraduate Engineering Student at UC, Irvine – Interested in Robotics and Controls
Phone: +1 949 257 8760
Email: nk.gurnard@gmail.com || LinkedIn: Nicholas Gurnard

***Purpose*—The purpose of this portfolio is to give any readers information on my professional endeavours and to get a feel for my personality. This portfolio is no longer updated and is designed to feature projects, accomplishments, and experiences I had in my undergrad. This information is not to be distributed or replicated without my consent. If there are any questions or concerns, do not hesitate to contact me.**

CONTENTS

I. INTRODUCTION

Hello! My name is Nicholas Gurnard, but most people call me Nick. I am a mechanical engineering student interested in the robotics industry. Robotics is a field with a wide range of subcategories, and I am still discovering which one is the best fit for me. As of now, my focus lies primarily in learning the brains behind intelligent robots, such as planning tasks and controls & manipulation. For my future career, I see myself working on human interactive robots, autonomous vehicles, or industrial automation. Currently, I am preparing to pursue a graduate degree in robotics to sharpen my skills before industry. Feel free to reach out to me!

II. ROBOTIC ARM SIMULATIONS IN MATHEMATICA
JANUARY 2021 - MARCH 2021

To gain more experience programming robots, I learned Mathematica to simulate robotic arms. Specifically, I was interested in the forward and inverse kinematics of 6 degree-of-freedom (DOF) and 3 DOF robots. Using forward and inverse kinematics, I learned how robotic arms are oriented in a 3D environment starting from either the ground/reference frame or from the end-effector.

For my first simulation, I focused on the Adept 800 SCARA robot. Since forward kinematics are generally easier equations to solve, I started with using all of the SCARA robot's joint angles to position an end-effector in space. First, I assembled all of the joint angles and known parameters into a Denavit-Hartenberg table using Figure 2.

Joint i	Θ_i	d_i	$\alpha_{i,i+1}$	$a_{i,i+1}$
1	Θ_1	342	0	325
2	Θ_2	45	0	275
3	Θ_3	d3	-	-

Table I
DENAVIT-HARTENBERG TABLE FOR THE ADEPT 800 SCARA ROBOT.

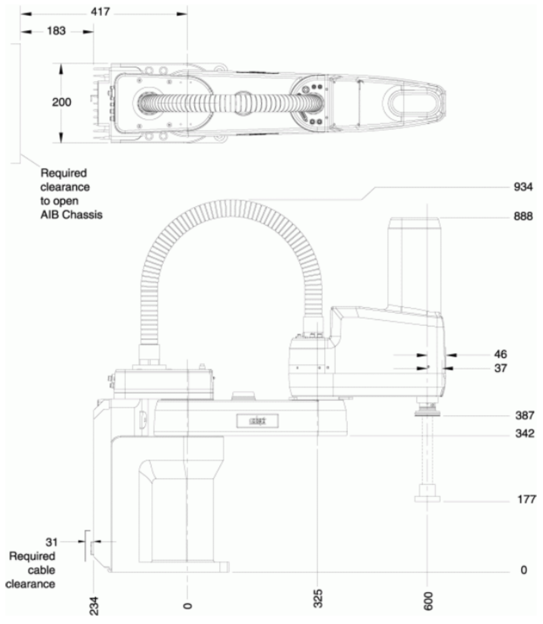


Figure 1. Adept 800 SCARA robot dimensions (in mm) used to assemble the Denavit-Hartenberg table.

From the Denavit-Hartenberg table, I employed the forward kinematics equations which consist of screw displacement matrices. Screw displacement matrices are homogeneous transformation matrices that allow for both rotational and translational movement to be accounted for simultaneously in a 3D environment. A homogeneous transformation matrix has a 3x3 rotation matrix in the upper left corner of a 4x4 matrix, and the rest of the entries are populated with 0 except for the bottom right entry which is populated with 1. The forward kinematics evaluation is as follows:

$$K(\Theta_1, \Theta_2, \Theta_3, d_3) =$$

$$[\mathrm{Z}(\Theta_1, 342)][X(\Theta_0, 325)][Z(\Theta_2, 45)][X(\Theta_0, 275)][Z(\Theta_3, d3)]$$

The Z and X functions represent the Z axis screw displacement and X axis screw displacement matrices, while K is the forward kinematics function. The forward kinematics equations allow for precise calculation of where an end-effector is in space if you know the robot's joint angles. Forward kinematics equations, linear algebra, and some Mathematica programming magic (AKA graphics commands that take a long time to master) eventually led to a successful simulation! The robot is easily able to move between end-effector key frames and actuate its linear actuator on the end. A video of the animation can be found by clicking HERE.

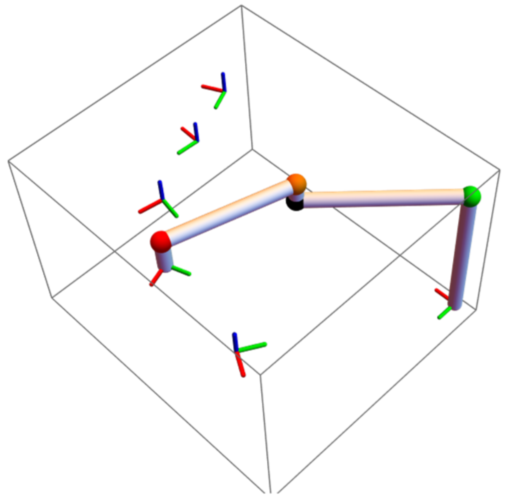


Figure 2. Adept 800 SCARA robot simulation. Full animation found by clicking HERE.

Many 6 DOF robots are not complete without a 3 DOF wrist as the end-effector, so I naturally employed the same techniques to simulate a robotic wrist. Again, a Denavit-Hartenberg table was assembled to employ the same forward kinematics analysis. After completion of the robotic wrist analysis, I can then couple the SCARA robot (or similar) and a wrist to finally simulate a 6 DOF robotic arm.

Joint i	Θ_i	d$_i$	$\alpha_{i,i+1}$	a$_{i,i+1}$
4	Θ_4	d4	$\pi/2$	0
5	Θ_5	0	$-\pi/2$	0
6	Θ_6	0	-	-

Table II
DENAVIT-HARTENBERG TABLE FOR A ROBOTIC WRIST. ASSUMING NO OFFSET, MEANING THE WRIST IS STANDALONE AND NOT ON A ROBOTIC ARM.

Since the keyframes (AKA the end-effector frames) generated by the forward kinematics equations all share the same origin for a robotic wrist, I offset them by multiplying by the coordinates of each keyframe by the Z screw displacement matrix at a specified radius. This allowed the keyframes to be easily visible on the surface of a sphere without mathematically impacting the result. Note that the robotic wrist uses the same forward kinematics equations outlined before. The final simulation can be found by clicking HERE. The animation shows a 3R wrist (a wrist consisting of 3 revolute joints or a) moving between the offset keyframes. The wrist is animated as a polygon.

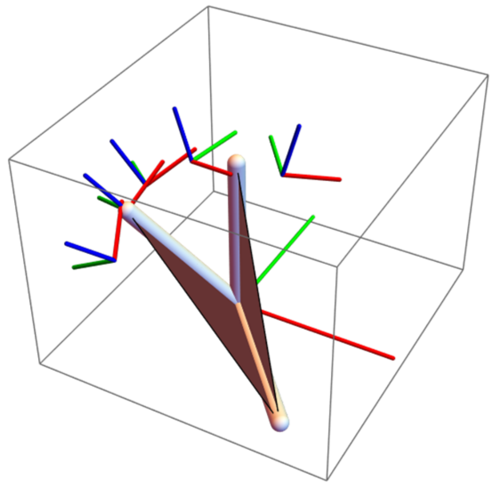


Figure 3. Robotic wrist simulation (3 DOF). Full animation found by clicking HERE.

After completion of the forward kinematics of the SCARA robot and the 3R wrist, I then learned how to calculate the inverse kinematics for each. With inverse kinematics, the goal is to calculate the joint angles needed to position an end-effector at a known location. The math here is pretty complex and lengthy, so I am just going to ask you, the reader, to trust me that I successfully completed this! The result is exactly the same as the figures above. Except this time, I find the joint angles from the end-effector location (inverse kinematics) instead of using the joint angles to find the end-effector location (forward kinematics). If you would like to know how I completed this task, I would be happy to hear from you! I will walk you through the steps on how it was accomplished and will even show my Mathematica code.

For a realistic and fun project, I used forward and inverse kinematics used to fully simulate the real Mars Rover. The fundamental principles of both forward and inverse kinematics were largely the same as before, except now there were minor differences that required mathematical "trickery". Extensive simplification and equation manipulation were required in order to solve a system of 4 equations and 4 unknowns. The following image shows what the Mars Rover looked like:

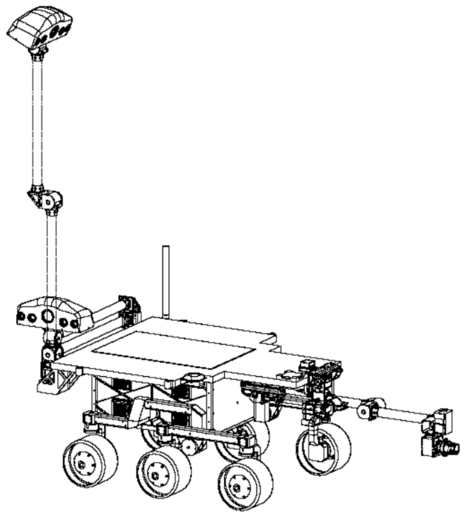


Figure 4. Mars Rover simulation (6 DOF).

The following Denavit-Hartenberg table was used to solve the forward and inverse kinematics (also show the dimensions of the robot):

Joint i	Θ_i	d$_i$	$\alpha_{i,i+1}$	a$_{i,i+1}$
1	Θ_1	12.8	$\pi/2$	0
2	Θ_2	0	0	65.5
3	Θ_3	0	0	65.5
4	Θ_4	0	-	-

Table III
DENAVIT-HARTENBERG TABLE FOR THE MARS ROVER

Once the forward and inverse kinematics were calculated (again, using the same principles as above), an animation was created and it looks incredible!

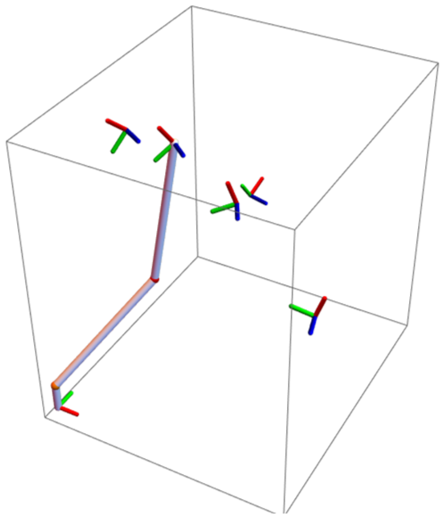


Figure 5. Mars Rover simulation (6 DOF). Full animation found by clicking HERE.

Finally, the last simulation I did was of a platform robot (AKA a parallel robot). These robots are different from serial robots (i.e. robotic arms) because they all must interact synchronously when touching the platform. Serial robots usually consist of several links connected via revolute or prismatic joints with one end connected to ground (fixed frame) and one end being free (moving frame). On the other hand, a parallel robot has 3 fixed frames and 3 moving frames, which are all correlated via a platform. As a result, the forward and inverse kinematics are slightly different in their difficulty of calculating. The inverse kinematics of a platform robot is easier than the forward kinematics because the location of the platform is known, and the only thing that must be calculated is the extension of the rods interfacing with the platform. On the other hand, the forward kinematics becomes much more complicated because the extension/retraction of one of the arms interfacing with the platform creates and effect on the other two arms. In essence, they are not entirely independent. However, the problem is still solvable! The fundamental principles are still very similar to serial robots with some minor differences. The inverse kinematics are solved via constraint equations that compute the leg lengths of each of the 3 legs using the location of the base pivots and the moving platform pivots. The forward kinematics is more complicated to solve and the math is too lengthy to put into this document. However, it is well documented online by solving the Heuristic Resultant and the constraint equations.

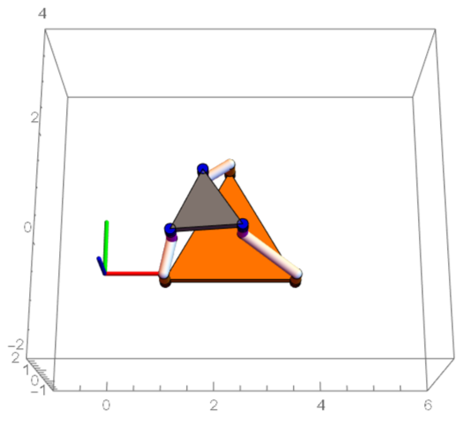


Figure 6. Platform robot simulation found by clicking HERE. Both forward and inverse kinematics were solved for this robot.

The trajectory of each of the joints of the parallel robot was mapped out in an effort to solve the forward kinematics. Each of the joints of the parallel robot drew a perfect circle, which is represented in blue in the following figure:

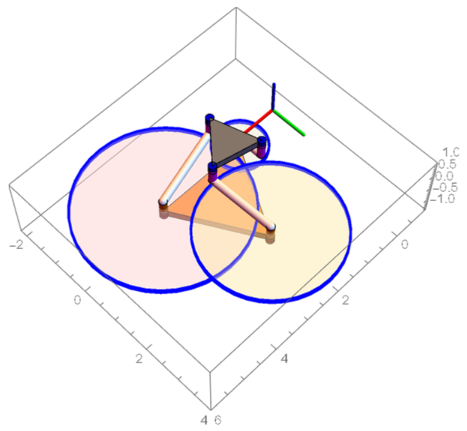


Figure 7. Platform robot joint trajectories found when solving forward kinematics.

III. Autonomous Lane Following Robot January 2021 - March 2021

A video presentation for this project was created and can be found by CLICKING HERE. It packages all of the following information into a convenient crash course!

A. *Problem*

Motor vehicle crashes are a leading cause of death in the U.S. that only grows in scale as more vehicles are continually introduced to already congested roads and distracted drivers. Consequently, autonomous transportation is a solution that will bring a drastically higher level of safety and consistency to public roads. There exists a need for a small-scale driving robot that allows for rapid testing of new control algorithms in real-world traffic settings. This design will be responsive to traffic signals and therefore can later be scaled to traditional vehicle-sized operation.

B. *Our Solution*

Our team (Team Tater Bots), consisting of Patrick Kelley, Luis Ramirez, and Selvin Garcia, present a low-cost, small-scale, autonomous vehicle that allows for rapid prototyping in real-world traffic situations. Our approach includes a device with the capabilities of 1) sensing lines in a path to maintain a singular lane even in curvy roads and 2) reading colored signals in its path to incorporate additional commands such as speed control and traffic signals. With the incorporation of IR sensors, we are able to detect the outer limits of our desired path and correct ourselves appropriately. Turning of our mobile robot is achieved with the slowing of the wheels of one side of the robot. Encoders attached to the motors allow for feedback of the wheel speed. Using the encoder signal, we can slow one side of the robot (2 wheels on one side) such that we achieve a skid-steered mobile robot design. In order to implement traffic commands to our autonomous device, we are incorporating a color sensor in order to send signals to the robot. By presenting different colored symbols in the path of the robot, we are able to slow down, stop, or speed up the overall speed of our car.

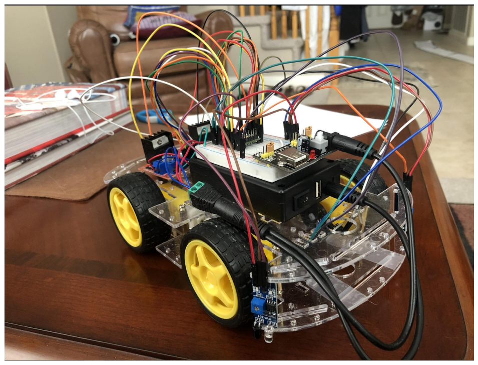


Figure 8. Final Autonomous Car build.

The robot uses a color sensor to detect colors that relay interrupts to the microcontroller that tell the robot to speed up, slow down, stop, etc (think traffic lights and stop signs). IR sensors allow for the robot to sense where it is in the lane by detecting a signal change between a dark street (asphalt) and white painted lane lines. Using feedback from the wheel encoders that provide wheel speed, the IR sensors, and the color sensor, the robot uses a PID controller to steer the robot back into the center of the lane.

The feedback from the various sensors that is fed into the PID controller was used to achieve our desired speeds in each independent wheel. Since DC motors do not provide feedback like a stepper motor, a feedback loop with motor encoders is required in order to properly ensure that we are achieving our desired speeds. Each motor incorporates an encoder and a U-shaped IR sensor to read the measured speed of each motor. This reading is then compared with the desired speed and the difference is then implemented into our PID controller as our error. Signals collected by the microcontroller from the IR sensor and color sensor will dictate the desired speed of each motor in order to achieve a desired kinematic response. The calculations associated with our feedback control loop include the calculations for finding the actual speed of our motors and the PID equations for each respective wheel. Calculating the speed of the motor with the encoder sensors used:

$$WheelSpeed =$$

$$1\frac{}{(DurBtwEncoderHoles+Gaps)*(NumEncoderHoles)}$$

With the PID equations taking the form:

$$u(k) = K_p * e(k) + K_i * \int_0^k e(\tau)\,dk + K_d * \frac{d}{dt}e(k)$$

The block diagram for the control of the vehicle is represented as follows:

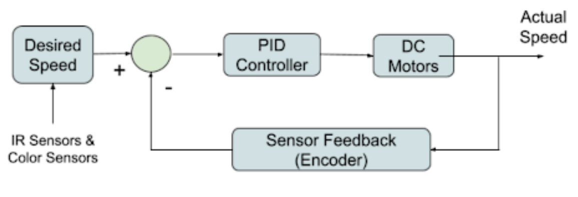


Figure 9. Block diagram for the control of the vehicle.

The gains K_p, K_i, and K_d were found by feeding sets of input and output data through MATLAB's system identification tool, which analyzes the relationship and defines a transfer function to characterize each motor. We used the speed encoder to measure outputs to a step/square wave, ramp, and triangle wave inputs to each motor and set the transfer function parameters to 2 poles, 0 zeros. The following plots are characterization of the front left motor (others are similar):

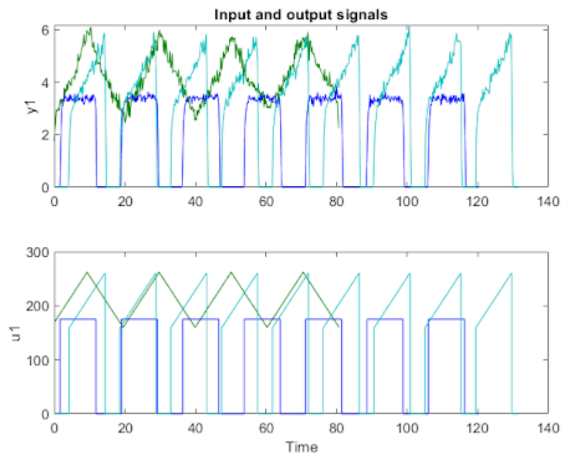


Figure 10. Input signals for a ramp, square, triangle wave and the corresponding wheel speed output.

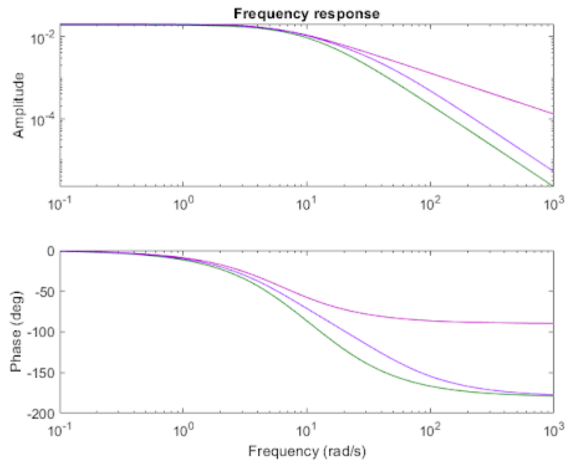


Figure 11. Bode plots for the wheel's response used to characterize the gains for the PID controller.

Finally, we fed the modeled transfer functions into MATLAB's PID tuner application and selected consistent values of T_S and T_R for the 4 motors.

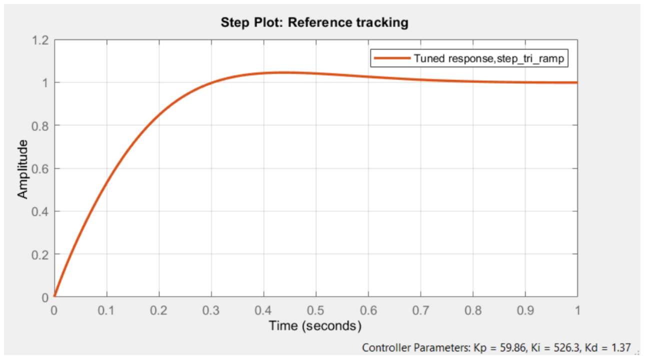


Figure 12. MATLAB's PID tuner application that provided numerical values for each of the gains for the wheels. This graph shows a step response.

PID equations for front left, front right, rear left, and rear right wheels respectively:

$$u(k) = 59.86 * e(k) + 526.3 * \int_0^k e(\tau)\,dk + 1.37 * \frac{d}{dt}e(k)$$

$$u(k) = 29.68 * e(k) + 460.4 * \int_0^k e(\tau)\,dk + .4783 * \frac{d}{dt}e(k)$$

$$u(k) = 28.11 * e(k) + 449.2 * \int_0^k e(\tau)\,dk + .4398 * \frac{d}{dt}e(k)$$

$$u(k) = 23.4 * e(k) + 364.4 * \int_0^k e(\tau)\,dk + .3753 * \frac{d}{dt}e(k)$$

The kinematic model of our robot can be represented by the following diagram. In this diagram, a fixed XY coordinate system is represented by O and a moving coordinate system fixed on the robot is represented by R.

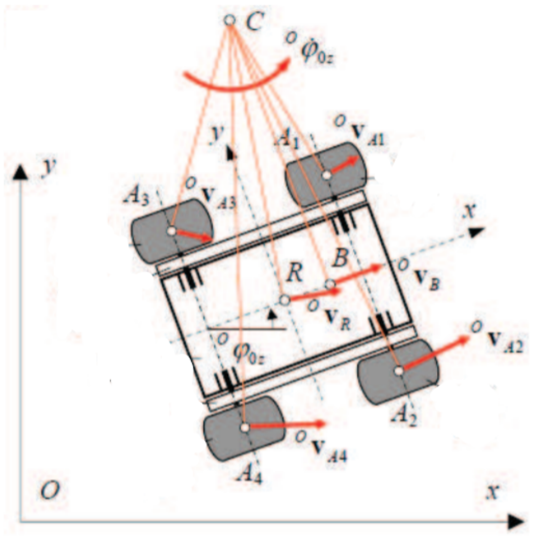


Figure 13. Kinematic model of our robot.

Position and orientation are described as:

$$^{\mathrm{O}}q = [^{\mathrm{O}}x_{\mathrm{R}}, {^{\mathrm{O}}y_{\mathrm{R}}}, {^{\mathrm{O}}\phi_{\mathrm{Oz}}}]^{\mathrm{T}}$$

Where vectors of with respect to the fixed frame O and moving frame R can be written as:

$$^{\mathrm{O}}\dot{q} = [^{\mathrm{O}}\dot{x}_{\mathrm{R}}, {^{\mathrm{O}}\dot{y}_{\mathrm{R}}}, {^{\mathrm{O}}\dot{\phi}_{\mathrm{Oz}}}]^{\mathrm{T}}$$

$$^{\mathrm{R}}\dot{q} = [^{\mathrm{R}}\dot{x}_{\mathrm{R}}{}^{\mathrm{O}}, {^{\mathrm{R}}\dot{y}_{\mathrm{R}}}{}^{\mathrm{O}}, {^{\mathrm{R}}\dot{\phi}_{\mathrm{Oz}}}{}^{\mathrm{O}}]^{\mathrm{T}}$$

The rigid body speed of the robot in the fixed frame O is found by multiplication of the robot Jacobian with the robot velocity vector of the moving frame R. This must satisfy:

$$^{\mathrm{O}}\dot{q} = {^{\mathrm{O}}J^{\mathrm{RR}}}\dot{q}$$

Where the J matrix has the following form:

$$^{\mathrm{O}}J^{\mathrm{R}} = \begin{bmatrix} cos(^{\mathrm{O}}\phi_{\mathrm{Oz}}) & -sin(^{\mathrm{O}}\phi_{\mathrm{Oz}}) & 0 \\ sin(^{\mathrm{O}}\phi_{\mathrm{Oz}}) & cos(^{\mathrm{O}}\phi_{\mathrm{Oz}}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

J is the robot jacobian and can be modeled as a simple rotation matrix since the robot can be modeled as a rigid body. As the robot turns, the Jacobian reflects the above image, where the J is the rotation matrix [1].

The following is the electrical circuit schematic for our robot:

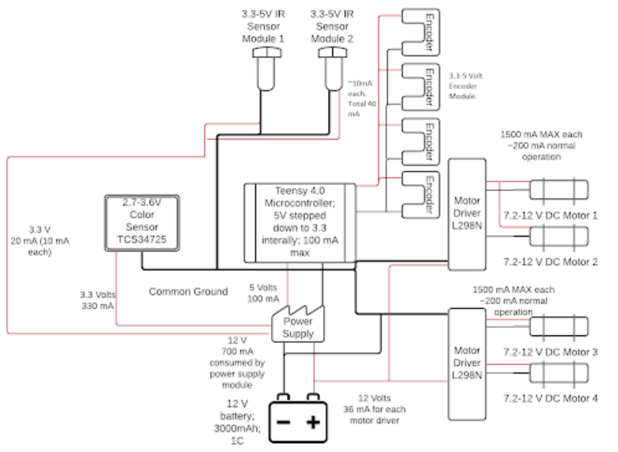

3.3-5V IR Sensor Module 1
3.3-5V IR Sensor Module 2
~10mA each. Total 40 mA
Encoder
Encoder
Encoder
Encoder
3.3-5 Volt Encoder Module.
1500 mA MAX each ~200 mA normal operation
Motor Driver L298N
7.2-12 V DC Motor 1
7.2-12 V DC Motor 2
2.7-3.6V Color Sensor TCS34725
Teensy 4.0 Microcontroller; 5V stepped down to 3.3 interally; 100 mA max
3.3 V 20 mA (10 mA each)
3.3 Volts 330 mA
Common Ground
5 Volts 100 mA
Power Supply
1500 mA MAX each ~200 mA normal operation
Motor Driver L298N
7.2-12 V DC Motor 3
7.2-12 V DC Motor 4
12 V 700 mA consumed by power supply module
12 V battery; 3000mAh; 1C
12 Volts 36 mA for each motor driver

Figure 14. Diagram showing how each electronic component will be connected. A bigger image can be found by clicking HERE.

Due to the number of pins on the Teensy 4.0 (40), each component has a pin that it can connect to. This selection of microcontroller allows for the connection between all IR sensors, color sensor, and motor drivers. The following illustrates how data will be transferred between components:

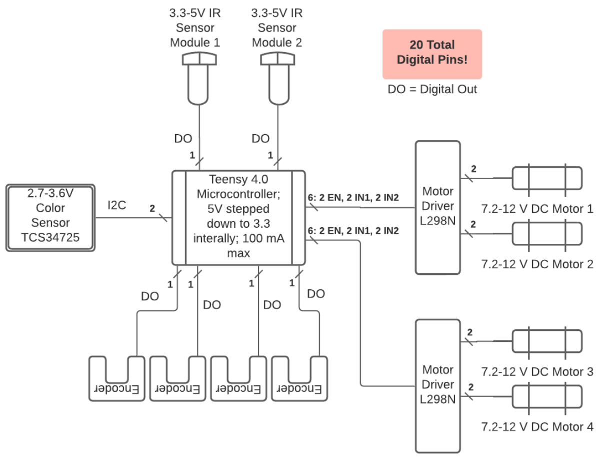


Figure 15. Logic Schematic. A bigger image can be found by clicking HERE.

A finite state machine (FSM) was created in order to program the logic of the robot operation onto the Teensy 4.0 microcontroller accurately. For cleanliness of this document, please navigate to THIS LINK in order to view the finite state machine. It is large and dense, and so it would be best to view independent of this document. The white states represent lane following and the colored states are interruptions representative of traffic sign detection. Inputs are omitted in this version for clarity. The input and flag values are included in this version of the FSM. Inputs are measured by sensors and flags are generated by the microcontroller. To improve diagram readability, the colored states from the previous slide are combined in the 'SC' state, which represents all possible color-detection interruptions and consequent actions taken. The dashed box is a close-up diagram of the state 'SC,' which acts as a black box in the global FSM.

This project was conducted during the peak of the COVID-19 pandemic which caused some logistical issues in the construction of our robot. Despite the added difficulties caused by the pandemic during testing, component integration, and the design process, this robot was an overall success!

IV. Activity Prediction
Winter 2020 - Spring 2020

Smart phones and smart watches are wielded by millions of people in societies all over the world. These gadgets contain powerful data collection hardware inside of them including acceleration sensors, cameras, location sensors, compasses, fingerprint sensors, and even heart rate monitoring sensors. The collection of these various sensor readings can give insight and research opportunities to many fields of study, including healthcare and exercise science, and much more. In this project, the problem of human activity prediction is outlined and analyzed. Using only simple features extracted from the time series data of smartphone/smartwatch accelerometers and gyroscopes, a classification model was made to identify what the subject was physically doing, such as walking, typing, eating, etc. Then, the variable that matters most when attempting to create a predictive model is identified.

This project analyzes sensor readings from the WISDM smartphone and smartwatch activity and biometrics dataset [2]. The WISDM dataset includes data collected from 51 different subjects numbered 1600 to 1650, each of which performing activities such as walking, typing, clapping, eating soup, eating pasta, etc. Each user was instructed to perform each activity from a list of 18 different activities, listed in Table IV, for a total of 3 minutes. Each of the 51 subjects was equipped with a smartphone and smartwatch, which held accelerometers and gyroscopes inside of them that collected acceleration data in each of the coordinate directions x, y, z. The subjects were given a LG G watch as their smartwatch and one of three types of smartphones: Google Nexus 5, Google Nexus 5x, or Samsung Galaxy S5.

The WISDM dataset provides 15,630,426 raw accelerometer and gyroscope readings for each of the x, y, and z axes. For each reading there is an associated time stamp, activity code, and subject identifier. Each reading was sampled at 20Hz for both the smartwatch and the smartphones. There are effectively 4 types of readings available: phone accelerometer, phone gyroscope, watch accelerometer, and watch gyroscope readings (pa, pg, wa, wg).

The data provided gave no indication as to which orientation each device was tested in. The directions x, y, and z could have been any direction in the 3D space the device occupies. In order to identify how the device was oriented relative to the ground, Figure 16 was constructed to show the acceleration for each of the three axes.

Walking	A
Jogging	B
Walking on Stairs	C
Sitting	D
Standing	E
Typing	F
Brushing Teeth	G
Eating Soup	H
Eating Chips	I
Eating Pasta	J
Drinking from Cup	K
Eating Sandwich	L
Kicking Ball	M
Playing Catch	O
Dribbling (Basketball)	P
Writing	Q
Clapping	R
Folding Clothes	S

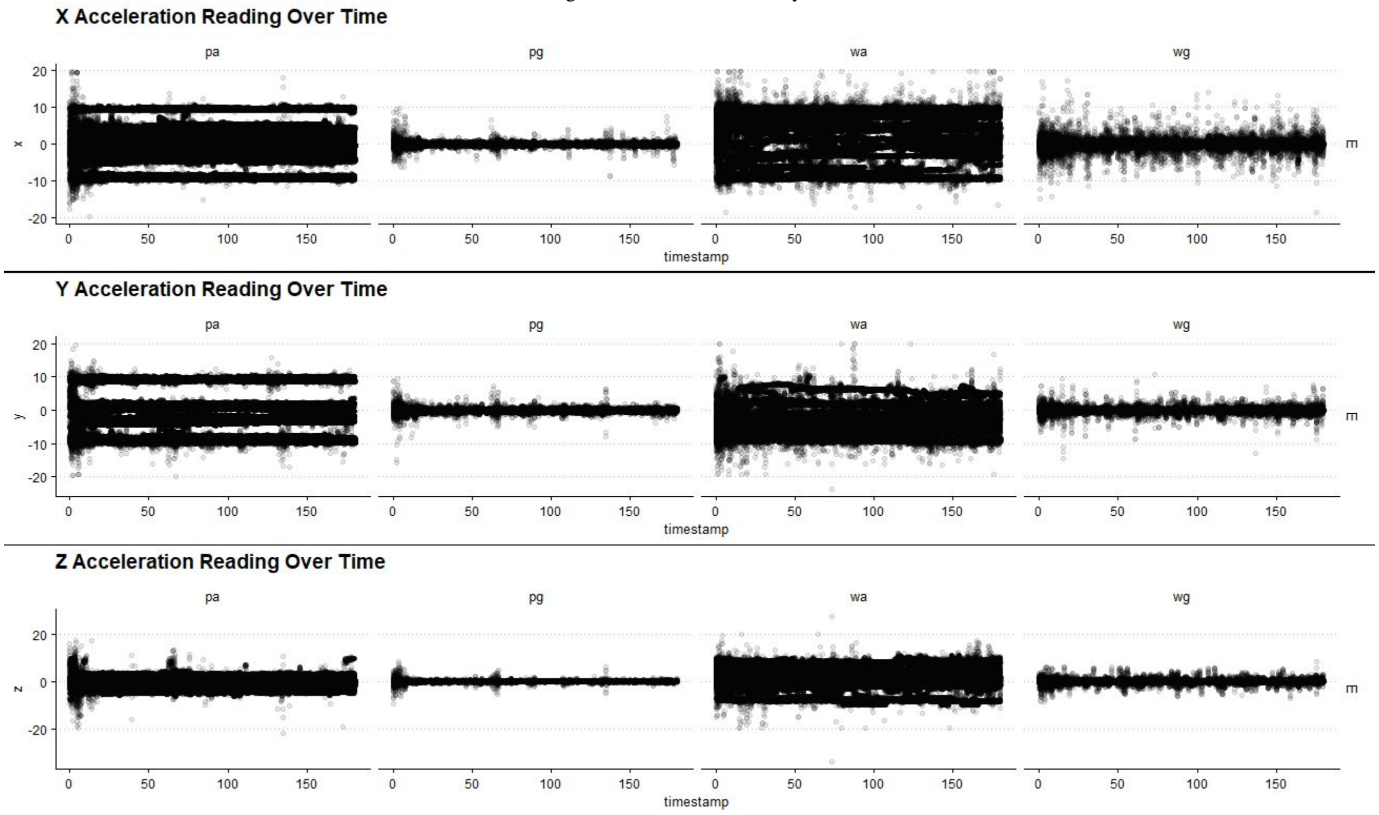
Table IV
List of activities and corresponding code.

For each axis, x, y, and z, it is clear from the figure that the orientation of each subjects' phone and watch are not consistent. Upright orientation occurs when the acceleration matches that of the earth's gravitational constant of magnitude 9.8 m/s2. This could be due to the fact that each phone has the sensors mounted differently, or upon manufacturing the orientation of the sensor is not consistent. Additionally, every subject could have wielded the devices at different angles when performing activities. Whatever the causation may be, it is clear that orientation cannot be taken into consideration without extensive data cleaning. Therefore, it is assumed that the device orientation is not a variable in the prediction of activities. If data were to be collected real time from millions of different people, the orientation would also not be consistent, so it is important to create a model that ignores that fact.

After some subject analysis, it was discovered that every subject failed to perform every activity. The subject that failed were subjects 1607, 1609, 1616, 1642, and 1643. These users were not included in the training of the models.

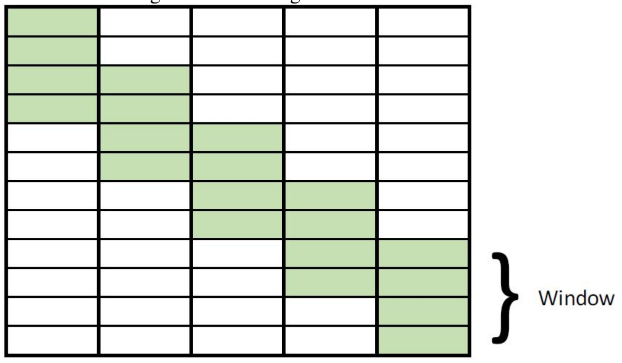
The raw data from any sensor/device combination is extremely noisy, so feature extraction is necessary to create an accurate prediction model. Additionally, building a model to predict the activities with only the instantaneous accelerometer and gyroscope readings, or the raw data, is extremely inaccurate. Out of the 3 minutes each subject performed the activity, a total of 2 minutes was extracted from the activity. Because of uncertainty as to how the data was collected, the start and the end of each activity was removed in case the subject had recordings made while setting up and shutting down their devices, which then amounted to the 2 minutes specified. Thus, 19 different types of features were extracted from the raw data for each of the 4 effective types of readings (pa, pg, wa, and wg respectively). For the extraction of these

Figure 16. Orientation analysis.



features, a window of a size 10 seconds is considered. A 10 second window was chosen because 10 seconds allowed for an adequate number of cycles of the activity being analyzed. The window progressively rolls across the 2-minute span of the activity for each subject with an overlap of 50%. For example, if the first window covers the first 10 seconds of the 2-minute span, that would be rows 1-200. The following window would be rows 100-300 and the window would keep rolling until it reaches the end of the 2-minute span. A visual of how the window operation works is Illustrated in Figure 17.

Figure 17. Rolling window visualization.



The types of features extracted are listed below, with the number of features generated in braces:

- Mean {3} – The average sensor reading over the window. One for each axis.
- Standard Deviation {3} – The standard deviation of the sensor readings over the window. One for each axis.
- Variance {3} – The variance of the sensor readings over the window. One for each axis.
- Time Between Peaks {3} – The time between the peaks of the sensor readings over the window. One for each axis.
- Average Resultant Acceleration {1} - The average resultant value of the sensor readings over the window. Found by taking square root of the sum of each instantaneous reading for each axis squared. Then, averaging those values over the window.
- Maximum {3} – The maximum sensor reading over the window. One for each axis.
- Minimum {3} – The minimum sensor reading over the window. One for each axis.

The maximum, minimum, and time between peaks features are extremely sensitive to outliers. To correct for the outliers, the window was smoothed using a moving average. A small sub-window of size 10 readings, or 0.5 seconds, was used as the smoothing parameter that operates as another rolling window. The sub-window was intentionally designed to be very small to ensure that the data was not being smoothed too much. This way, important peaks are still kept with relatively the same amplitude, and the signals between activities are still distinct. The maximum, minimum, and time between peaks features were then calculated from this slightly smoothed signal.

Figure 18. Moving average visual to demonstrate close fit. Red Line: Smoothed using moving average with sub-window of size 0.5 sec. Black Line: original signal.

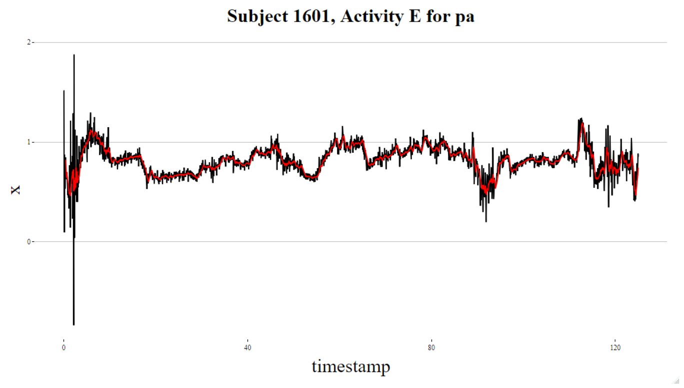
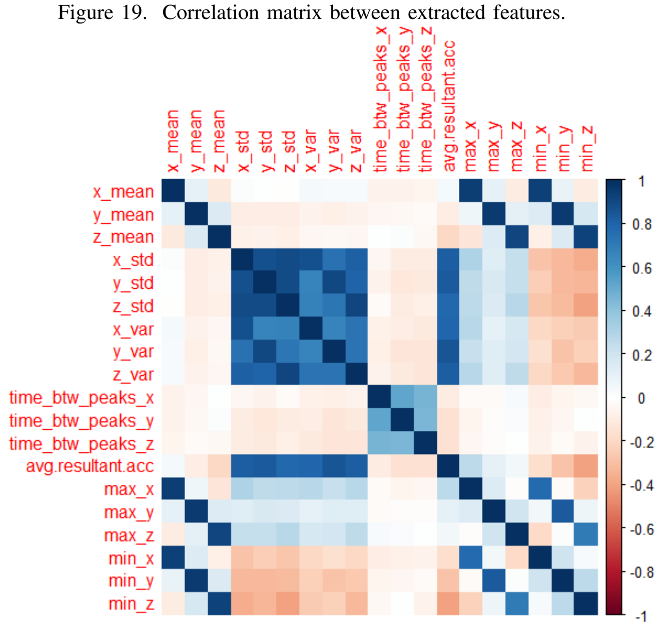


Figure 19. Correlation matrix between extracted features.



It is important to note that, in general, smoothing signals is poor practice because it makes the various signals for each activity too similar to one another and confuses the predictive model. However, since the smoothing was not too extreme, the model performed roughly 3% better on average when the data was smoothed.

In addition to the 19 extracted features, the activity label and the subject numeric identifier (1600-1650) were also included as variables for the model for a total of 21 variables. Once the features were extracted, a correlation matrix was created to determine which variables were highly correlated, depicted in Figure 19. Note that the highly correlated features have a correlation coefficient close to magnitude 1.

As expected, the variance and the standard deviation features are correlated because the standard deviation is simply the square root of the variance. In addition, a relatively strong correlation exists between the average resultant acceleration and the standard deviation/variance, as well as the minimum/maximum and the mean. In the creation of the model, it is important to recognize what is correlated in order to avoid confusing the model thus decreasing prediction accuracy.

Now, the predictive model is ready to be created. The random forest classification algorithm was the main predictive model used to classify subject activity. A random forest is powerful for this particular data set because there is little need for model interpretability, only high performance. There is no clear indication of clustering, which is the motivation for choosing random forests over a simpler, faster, and more interpretable model such as KNN. Random forests are bagged decision tree models that randomly choose a specified number of m predictors as split candidates from a full set of p predictors. Each split can only use one of the m predictors and a fresh set of m predictors is taken at each split. This leads to decorrelation of the trees within the forest thus leading to lower variance. The predictors in this case will be the extracted features.

For each of the 4 effective reading types (pa, pg, wa, wg), there were 3-4 random forests made. The difference between each of the random forests is outlined as follows:

1) RF1 - Random Forest predicting all activities at once.
2) RF2 - Random Forest predicting only activities where the corresponding smart device is of importance.
    a) Phone activities: walking (A), jogging (B), walking on stairs (C), sitting (D), standing (E)
    b) Watch activities: typing (F), brushing teeth (G), eating soup (H), eating chips (I), eating pasta (J), drinking from a cup (K), eating sandwich (L), kicking ball (M), playing catch (O), dribbling (P), writing (Q), clapping (R), folding clothes (S)
3) RF3 - Random forest predicting all activities at once and eating activities are combined as one category.
    a) Eating activities: H, I, J, L
4) RF4 - Random forest predicting only activities where the corresponding smart device is of importance and eating activities are combined into one category.

In the creating of each random forest, the highly correlated features identified from Figure 19 were removed as variables. The variance was removed instead of the standard deviation because it resulted in marginally less accuracy for each random forest. The minimum, maximum, average resultant acceleration, and mean features were all reincluded in the random forests because the removal of any combination of those features resulted in a decrease in accuracy that was greater than 5%. Additionally, the value of m was tuned to get the maximum possible accuracy.

The features were then split into model training data and model validation (test) data. The first 75% of features for each of the 4 types of readings was used as the model training data, and the remaining 25% as the test data. This allowed for the model to be trained on most users while intentionally leaving out several users to then validate the model.

The performance of each random forest for each reading type is outlined in Table V. Note that RF4 for the phone has no entries because there are no eating activities where the phone was considered of importance.

	Accuracy	Kappa	95% Confidence Interval
RF1 - pa	81.03	0.80	(79.85, 82.17)
RF2 - pa	95.80	0.95	(94.52, 96.84)
RF3 - pa	84.13	0.82	(83.02, 85.19)
RF4 - pa	NA	NA	NA
RF1 - pg	60.04	0.58	(58.58, 61,48)
RF2 - pg	89.65	0.87	(87.82, 91.29)
RF3 - pg	65.89	0.61	(64.47, 67.28)
RF4 - pg	NA	NA	NA
RF1 - wa	83.59	0.83	(82.46, 84.66)
RF2 - wa	84.87	0.84	(83.58, 86.09)
RF3 - wa	86.90	0.85	(85.87, 87.88)
RF4 - wa	89.56	0.88	(88.44, 90.60)
RF1 - wg	68.46	0.67	(67.06, 69.82)
RF2 - wg	70.89	0.68	(69.28, 72.46)
RF3 - wg	74.41	0.71	(73.10, 75.69)
RF4 - wg	79.44	0.76	(77.99, 80.83)

Table V
PERFORMANCE OF EACH RANDOM FOREST MODEL.

For each random forest model, the performance was assessed by prediction accuracy, Cohen's kappa coefficient, and a 95% confidence interval. The accuracy column displays what the percent agreement is between the predictions made from the test data and the actual test data. The kappa coefficient is a number that lies between 0 and 1 that gives insight as to how many of your correct predictions may have resulted by pure chance. A kappa closer to 1 indicates that there is less chance that the prediction accuracy was due to chance. Lastly, 95% confidence interval is the interval in which the model accuracy will fall given a new validation set with 95% confidence.

When comparing the features from the accelerometer readings against the gyroscope readings, the accelerometer outperformed the gyroscope. This is likely due to the fact that gyroscopes and accelerometers are inherently different in how they collect acceleration data. An accelerometer measures the rate of change of velocity an object, while a gyroscope maintains its orientation by allowing the freedom of rotation and then measuring rotational changes in velocity. Therefore the, rate of change of the linear velocity is a more important factor than any rotational movements.

As expected, the phone features performed significantly better when considering only the activities where the phone is of importance (RF2-pa). A maximum classification accuracy of nearly 96% was achieved when using only the phone accelerometer data in the RF2 model. In addition, this accuracy was not by chance since there was a high kappa value of 0.95. The watch features performed slightly better to the phone features when trying to predict all activities at once. However, only marginal improvements were made when considering only activities where the watch is of importance (RF2-wa) against predicting all activities at once (RF1-wa). So, even after filtering out phone activities, the model struggled to reach a result as impressive as RF2-pa. The causation for this absence of drastic improvement is because of how similar the eating activities were to each other.

The differentiation between activities H, I, J, and L, the eating activities, was the most difficult challenge when considering any subset of data with the given features extracted. About a 5% increase in accuracy was observed between models RF1 and RF3 for each of the 4 types of readings when combining all eating activities. Activity K, drinking from a cup, was equally as difficult to distinguish from each of the eating activities. However, it was intentionally left separate because a subject eating does not necessarily imply drinking and vise-versa. If it were to be included in the combination of eating activities, the accuracy would likely increase by a few percent. Table VI shows more performance measures for RF1-pa to demonstrate the difficulty of differentiating between eating activities/drinking.

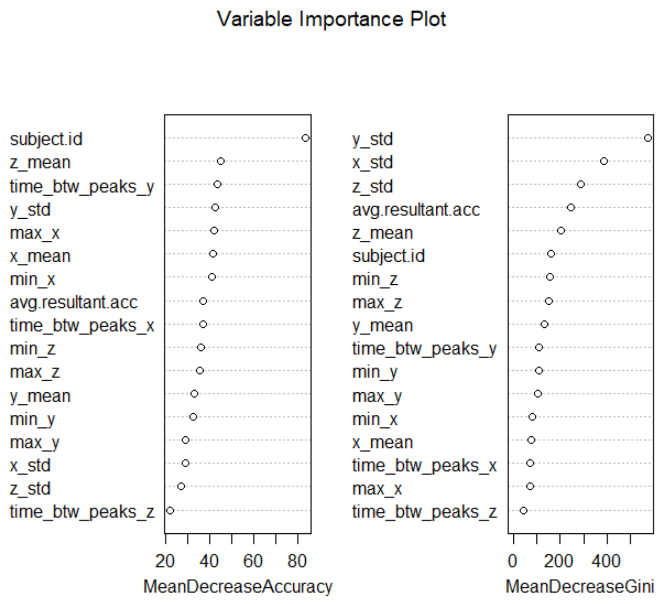
	Precision	Recall	F1
Class: A	0.896	0.888	0.892
Class: B	0.948	0.979	0.963
Class: C	0.792	0.847	0.818
Class: D	0.779	0.801	0.790
Class: E	0.899	0.880	0.889
Class: F	0.876	0.847	0.861
Class: G	0.782	0.843	0.811
Class: H	0.833	0.785	0.809
Class: I	0.717	0.731	0.724
Class: J	0.787	0.613	0.689
Class: K	0.696	0.624	0.658
Class: L	0.752	0.719	0.735
Class: M	0.716	0.806	0.758
Class: O	0.734	0.806	0.768
Class: P	0.860	0.755	0.804
Class: Q	0.861	0.905	0.882
Class: R	0.815	0.889	0.851
Class: S	0.864	0.877	0.871

Table VI
EXTRA PERFORMANCE MEASURES FOR RF1-PA.

For activities H, I, J, K, and L, the recall and F1-score are much lower in comparison to the other activities. The F1-score is the harmonic mean of the recall and the precision, where a higher F1-score indicates better results. Therefore, differentiating between tasks that entirely different is much easier than trying to distinguish eating chips vs. eating pasta.

After each model was created and assessed, a variable importance plot was created to identify the impact each variable had in the training of the model, shown in Figure 20. Each variable importance plot was slightly different when training each forest, however they are largely very similar.

Figure 20. Variable importance plot for RF3-pa.



The feature that was most critical in the performance of every model was the subject identifier, as demonstrated by the left-hand panel of Figure 20. What matters most when analyzing a single activity is the style in which each activity is performed. For example, the features of one user while walking is entirely different than another user while walking despite the fact that they are performing the same exact activity.

Every feature in every model contributed significantly to the overall performance of the random forest models thus signifying the simplistic features that were chosen are appropriate when predicting subject activity.

In conclusion, simple feature extraction of time series accelerometer and gyroscope data is sufficient in the prediction of human activity. To further improve on the results in the future, a support vector machine model can be created since it is known that subject identity is an important variable, thus pointing towards potential clustering. Comparing the random forests with the SVM model could give insight to more phenomena that are currently not apparent due to the low interpretability of a random forest. In addition, knowing the orientation of each device could improve performance tremendously since the features dependent on the axes were of high importance, such as the mean and the maximum/minimum amplitude. Finally, increasing the amount of overlap for the rolling window operation could increase prediction accuracy, but the trade off would be an increase model training time as the amount of feature data being extracted would be substantially increased.

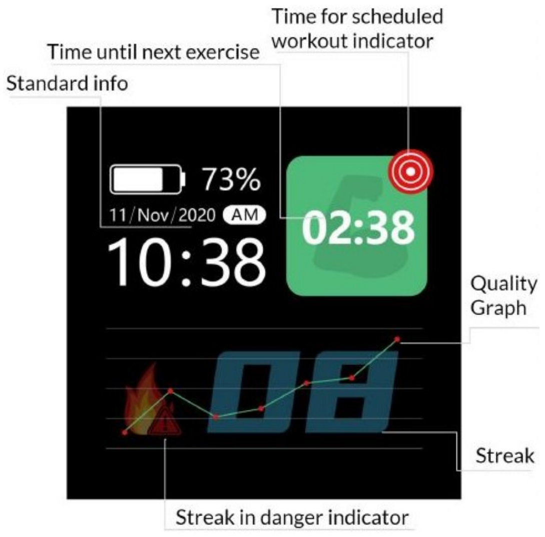
This project taught me what it was like to pursue a project with absolutely no guidance. Independent learning was critical to the project's success. In addition, I learned valuable skills in machine learning, statistics, and data analysis/visualization.

V. Wearable Sensing Rehabilitation Device
September 2020 - December 2020

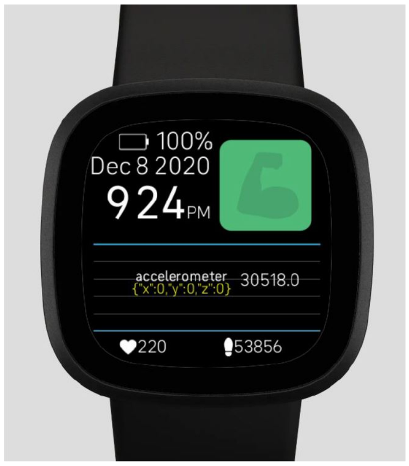
Today, there exists no successful device that allows stroke patients to feel independent and motivated to continue their rehabilitation exercises. Rehabilitation is a long and slow process, often discouraging patients from recovering any lost mobility. BrilliAnt, my newly established wearable sensing device design team, is creating a smartwatch application that will motivate the user via statistical measures, complex movement quality algorithms, and fun and easy to use user interface.

The application's main focus is on the homepage, or clockface, of a Fitbit Versa 3 or Fitbit Sense. On the homepage, users will be presented with several features including a motivational daily exercise streak counter, notifications to perform their next exercise, countdown timer for the next exercise, and movement quality progress graphs.

On the application homepage, the users can then begin a workout via the start button (green button) that will allow them to progress through a series of instructions that will guide them through their exercise. In the process, there are informational guidelines, tips on how to perform the best possible exercise provided by medical professionals, and motivational messages. Once an exercise is selected, a warm up routine ensues followed by the final exercise. Throughout the duration of the exercise, data from the watch's internal sensors such as the accelerometer, gyroscope, orientation sensor, and more will be recorded. The recorded data can then be directly piped to the team's scientifically validated algorithms to assess their exercise quality and overall success. The algorithm's main focus is on a movement quality quantification algorithm named "spectral arclength" or SPARC [3]. SPARC is used to measure the arc length of the Fourier magnitude spectrum within an adaptive frequency range. Essentially, it is used to calculate how smooth an inputted signal is (in our case, the acceleration profile for an arm movement) by considering the presence of high frequencies as an indication of roughness in the signal. Furthermore, SPARC is only valid for discrete movements, but is usable across varying frequency ranges and is independent



(a) Ideal Design

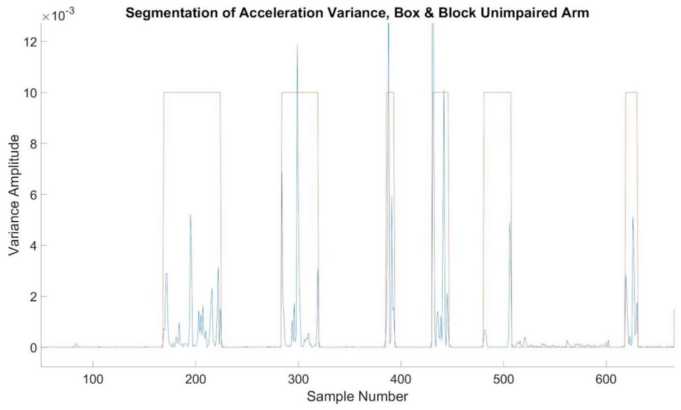


(b) Current Design

Figure 21. Clockface/Homepage – Ideal Design vs. Current Design. The current design is undergoing testing with the accelerometer API and a count based off of a conditional statement. Simulated sensor values read "0" instead of a sensible value.

of movement amplitude and duration.

In order to successfully use SPARC, the acceleration profiles for the gyroscope and accelerometer must be segmented into discrete movements. First, the data is arranged into a set of arrays, one for each sensor. For example, the data for the accelerometer will have values that are in each coordinate direction, X, Y, and Z, with a corresponding time stamp. Data analysis is then performed on the data for each unique exercise in order to apply an appropriate filter and apply signal processing techniques to extract relevant information. The most successful filter to smooth out the signal noise was a 10th order butterworth low pass filter. Once the data is cleaned, it moves through a custom segmentation algorithm that defines where the user's repetitions are located in the time domain. There are 3 main parameters that are optimized for the segmentation: population size/window size, time threshold, and variance magnitude threshold. For each exercise, the ideal parameters are discovered via a gridsearch in MATLAB to allow for the watch to only compute segmentation based off of 3 simple parameter values.



(a) Segmentation Performance

Figure 22. Segmentation Performance (Red) overlaid on top of the movement profile (Blue). Accuracy ranges between 80% and 95% depending on the exercise and the patient's degree of impairment.

Once the movements are segmented, the movement of the patient can then be quantified with the SPARC algorithm. When the workout concludes, the user will receive an exercise breakdown such as quality over time, repetitions per minute, and exercise duration. The user's exercise statistics will then be stored for motivational features on the clockface and, with user approval, for usage by their assigned physician. The motivational process can then be repeated whenever, wherever with the ultimate goal of encouraging consistent workouts and recovering overall motor skills of the arms.

	Box & Block, Impaired	Box & Block, Unimpaired
Population Size	5	2
Amplitude Threshold	.0004	.00037
Time Threshold	5	4
Segments Found	23	44
Actual Segments	25	44
Segment Error	8%	0%
SPARC Found	-8.4	-6.54
Predicted SPARC	-8 to -12	-8 to -12

Table VII
ANALYTICAL SEGMENTATION ALGORITHM PERFORMANCE AND SPARC ALGORITHM PERFORMANCE FOR THE BOX & BLOCK EXERCISES.

This project taught me how to lead a team, apply knowledge from coursework into an unguided design project, and learn JavaScript, CSS, SVG, GitHub from a command line interface, and Fitbit Studio independently.

VI. Applied Medical - Manufacturing Engineering Internship

Summer 2019

Note: In this section there will not be any provided pictures, drawings, graphs, etc. because an NDA was signed and will be respected. In addition, specifics about the products I worked on and the details of company procedures will not be disclosed.

Finishing my sophomore year of college, I decided that pursuing an internship was the best way to discover what subcategory of engineering would be best for me. Thus, a manufacturing role was acquired so that I could gain experience with industrial robots, design work, communicative tasks, programming, and more. In essence, the role was a great introduction to a wide variety of engineering tasks.

I joined the team in the plastics/polymers department which focused on mass producing disposable, cheap, and sanitary medical devices. The goal was to provide affordable medical devices so that people all over the world could have access to top of the line medical products. In this internship, my duties mostly consisted of the design and manufacturing of end-of-arm tooling (EOAT) for robotic arms, fixtures for product assembly, and devices that aid in the manufacturing process.

Designing the end-of-arm tooling required innovative design work followed by rigorous testing and methodological verification. In general, other team members would design a medical device and its corresponding mold cavity for a specific surgical need and then pass it on to my team for manufacturing and testing. Given the mold cavity, product specifications such as geometric restrictions, critical part dimensions, surface conditions, array layout, and more had to be taken into consideration. Once all of the specifications were identified and accounted for, I designed an EOAT located on a robotic arm that would remove the devices from the mold cavity without damaging, warping, or contaminating the part, trim any and all excess plastic (runners), and then transport the devices to a storage bin. The process had to be repeatable for autonomous operation and must operate as efficiently as possible. In my brief 2.5 month internship, three successful EOAT devices were completed, all of which are still being used in production, that could automate the injection and insert molding process for newly designed parts created by the team.

Once an EOAT was successfully designed, manufactured, and placed on the assembly line, I worked on part verification and product assembly. In the polymers department, there were injection molded parts and insert molded parts that needed to pass strict specifications to qualify as safe medical devices. Thus, I designed fixtures and devices to aid in product assembly, inspection, and verification for quality control. Here, I created an inspection method using a fixture I designed that brought part failures down from 50% to 12-15%.

Overall, the internship was a great experience in which I learned a lot about design, communication, and manufacturing. However, the medical device industry proved to be too slow and restrictive for my personality. My favorite part of the job was working on the EOAT devices because I had the freedom to design and manufacture as I see fit. Because the tooling was integrated with a robotic arm, I became fascinated in more complex robots and how they operate (i.e the brains, such as the code that controls and manipulates the robotic arms). In the end, the internship gave me insight into what interested me most – robotics.

VII. Payload Delivery Drone

January 2018 - March 2018

A video of this project during our first flight can be found by clicking HERE.

The objectives of this project were to construct a functioning quadcopter for under \$500 within the given time frame (10 weeks) that had an autonomous payload delivery system. The payload system was under separate constraints where the sensors must detect a color, either red or blue, at about a distance of about 150 cm; from there it then must drop one of two balls depending on the color of the target. The team wanted to provide a quadcopter that could be used recreationally and professionally, like for delivery services or military medical assistance on the battlefield.

The following were the design requirements:

- Motor to motor distance no longer than 14 inches
- The only permitted fabrication materials. are woods, plastics, or carbon fiber. No metal frames.
- Structures must be fabricated by scratch.
- Internal combustion engines are prohibited.
- Quadcopter must be able to fly for 5 minutes without recharging the battery at an altitude of 2-6 feet, no higher than 8 feet.
- Propeller guards at least a half inch out from the tip of the propeller must cover at least a quarter of the circumferential distance
- Expenditures must total \$500 or less

For the design, the weight of the quadcopter had to be taken into consideration in order to select the right motors and create the optimal sized frame. The equation for trust is as follows:

$$T = 2(\frac{sg}{16}[\sqrt{(1+\frac{64}{3sg}*\theta)}-1])^2*\rho(\Omega R)^2 A \tag{1}$$

Where s is the distance between motors, g is gravitational acceleration, theta is the pitch, rho is the air density, R is the propeller radius, and A is the quadcopter area.

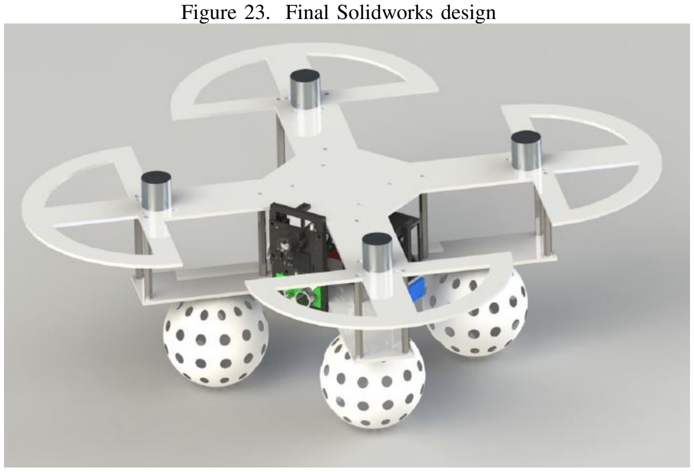
The experimental value of thrust for A Sunny Sky 1500 motor was 500 g at 8 amps for one motor with an 8 inch propellor and 370 g at 5 amps. So at 8 amps, 4 Sunny Sky

1500 motors can lift 2000 grams. The quadcopter holding an Arduino and all essential components was measured to be 1211.2 grams fully assembled, so the thrust to weight ratio is 2000g/1211.2g = 1.651. This means that in order to lift off, 61% throttle must be used. (1/1.651). By using the experimental value of thrust in grams, the thrust for one motor is 4.9N. For four motors, the thrust is 19.6N theoretically.

The battery has a capacity of 3000mAh = 3Ah. So, charge/discharge time = (3Ah)/(32 amps) = .09375 hours = 5.625 minutes flight time while running a consistent 8 amps, which satisfies the design requirement.

Next, the frame was designed and constructed according to specs from the thrust equation and design parameters. The basic form of the quadcopter became two polycarbonate sheets separated by 3 inch standoffs. The 3 inches of space in the body of the quadcopter allow for placement of the all essential components and electrical wiring: 2 boxes to contain the balls, NAZA flight controller, Arduino, both servo motors, the receiver, and the regulator. The Naza, receiver, and regulator all fit in between the ball (payload) housing which hung slightly off of the sides of the quadcopter. In addition, there are two slots bored out of the bottom plate to thread velcro straps through in order to hold the battery below the bottom plate. The propeller guards were originally half-circles with only a 1 inch outer lip, but when assembling the frame, it was decided the guards would structurally benefit from extending that lip down the center as seen Figure 23. In order to hang the Arduino upside-down, electrical wire was threaded through the 4 holes in the Arduino and top plate of the quadcopter. The landing gear became 4 wiffle balls zip tied to the bottom piece of polycarbonate. The PiXY camera and ultrasonic sensor were mounted to a 3D printed bracket. For more stability, the team found it suitable to include most of the weighty components at a level under the motors.

Figure 23. Final Solidworks design



The quadcopter then passed the durability test post construction by surviving a 7 foot drop. Next, a tilt test was performed to ensure the stabilization of each motor was performing properly from the flight controller. Once the tilt test was passed, a test flight sans payload equipment verified the quadcopter flew as expected.

Finally, the payload delivery system was coded in Arduino. The code was basic as it only required color detection from a PiXY camera which then actuated one of two servo-motors depending on the perceived color (red or blue). Once the code was written, the payload equipment was mounted to the frame of the quadcopter and tested during flight.

This was the first successful hands-on project I worked on with a team during my undergraduate education. At this point, I was unfamiliar with coding, manufacturing, testing procedures, and proper communication within a design team. This project exposed me to all of the above, thus justifying its place on my portfolio. In addition, the success of the quadcopter put it at 1st place in the timed flight competition among other UC Irvine students.

VIII. Tactical Shotgun - Solidworks Project

April 2018 - June 2018

Instead of a text-based explanation, I made a fun, brief YouTube video showing my creation found by clicking HERE.

Acknowledgments

I especially thank Professor Donald Dabdub for his inspiration in my life and his influence on my career ambitions. His mentorship and friendliness has changed my perspective on countless subjects. Thanks to my peers Felix Slothower, Jonathon Palafoutas, Matthew Polcyn, Patrick Youssef, Eric Abdulaziz, Abdullah Jawhar, Patrick Kelley, Joseph Parra, Matthew Wong, Kristen Rosenberg, Myriam Khalil, Sara Lippert, and many others in provoking new ideas and thoughtful conversations in my undergradute studies. I would like to thank all of the professors who helped me along the way, including Professor Jabbari and Professor Reinkensmeyer for their willingness to go the extra mile for me. I thank my family for supporting me every step of the way.

References

[1] M. Trojnacki, "Dynamics model of a four-wheeled mobile robot for control applications–a three-case study," in *Intelligent Systems' 2014*. Springer, 2015, pp. 99–116.

[2] G. M. Weiss, K. Yoneda, and T. Hayajneh, "Smartphone and smartwatch-based biometrics using activities of daily living," *IEEE Access*, vol. 7, pp. 133 190–133 202, 2019.

[3] "A Robust and Sensitive Metric for Quantifying Movement Smoothness," accessed: 09-2020. [Online]. Available: doi:10.1109/TBME.2011.2179545