SydeKick

ECE 445 Final Document

Balasabapathi Chandrasekaran (bmchand2), Sam Feizi (feizi2), Rohan Mohapatra (rmohapa2)
Group #64
TA: David Hanley

1. Introduction
    1.1. Purpose

According to the Centers for Disease Control and Prevention (CDC) rates of autism have increased from 1 in 150 newborns being diagnosed with autism in 2000 to 1 in 59 in the year 2018 [1]. Stanford University's Lucile Packard Children's Hospital found that ASD children tend to have short attention spans [4]. Speech Language Pathologist, Beverly Vicker adds on by saying "...students on the autism spectrum appear erratic in their ability to follow directions" [5]. Stanford University's Lucile Packard Children's Hospital did research into how to interact with a child who has autism and found that the following methods were important to better the problem of following directions and maintaining an attention span:

- Interact through physical activity
- Always stay positive (reward system)
- Show your love and interest[4]

Yale University conducted a study in August of 2018 where they witnessed significant improvements in the social skills of children with autism after a month of working with robots[2]. Through our search we are motivated for this project is to build a therapeutic Sydekick for those who have have autism.

Our intention with SydeKick was to create a robot that was a partial humanoid; this means it has a torso, head, and two arm-like appendages that operate on one rotational axis. Sydekick has a set of wheels to govern its translational and rotational motion and make it easier for the child to interact. The robot can play Simon Says which was picked by our group cause we felt that it best captured following directions and keeping attention spans.As discussed throughout the semester, testing our product with an actual child with ASD was not feasible because of the regulations in play. Instead, through our research, as discussed above, we have identified the following : (1)The challenges that children with autism face and (2)The types of activities that can help mitigate some of these challenges. With this information, our team created and verified that we could accomplish these tasks.Through validating that SydeKick could perform these tasks with anyone, our team created a theoretical product that could work with children on the autism spectrum.

    1.2. Functionality

Throughout our project there were three primary engineering objectives that our group worked towards. The first was **gamification,** our goal was the following: Sydekick should be able to reward 3 right answers through our reward system over a time period of 2 minutes through a game of Simon Says with a user (TA/student). The second objective was focused around **proximity**, our goal was the following: Sydekick should be able to stay in front of the test player (TA/student) at all times maintaining a distance of 9 inches, this includes when Sydekick moves its hands up or down. The final objective was centered around **safety**, our goal was: Sydekick will not be able to drive past a certain average PWM (175 - 255); will all only be able to lift arms

from hips to shoulders (0 - 90 degrees). The aforementioned objectives were reached by focusing on three main high-level project functionalities that summarize our **Requirements & Verifications:** (1) Movement, (2) Game Interface and (3) Monitor Features. The Movement functionality describes the movement of the motorized base which included two wheels in the front powered by two DC motors and a castor wheel in the back. It also describes the arms which moved in parallel from the hips to the shoulders which are powered by two Servo motors. The Game Interface functionality describes our game Simon Says which entails the integration the Raspberry Pi 3 Model B with the Force Sensitive Resistors. The Monitor Features describes the two LCD displays that should have been on the robot. The first 7 inch LCD is mounted on the torso and the other is a smaller LCD positioned as a head. The torso LCD is used as the platform for which the user interacts with Sydekick and the head LCD is used for a positive facial expression.

1.3. Subsystem Overview

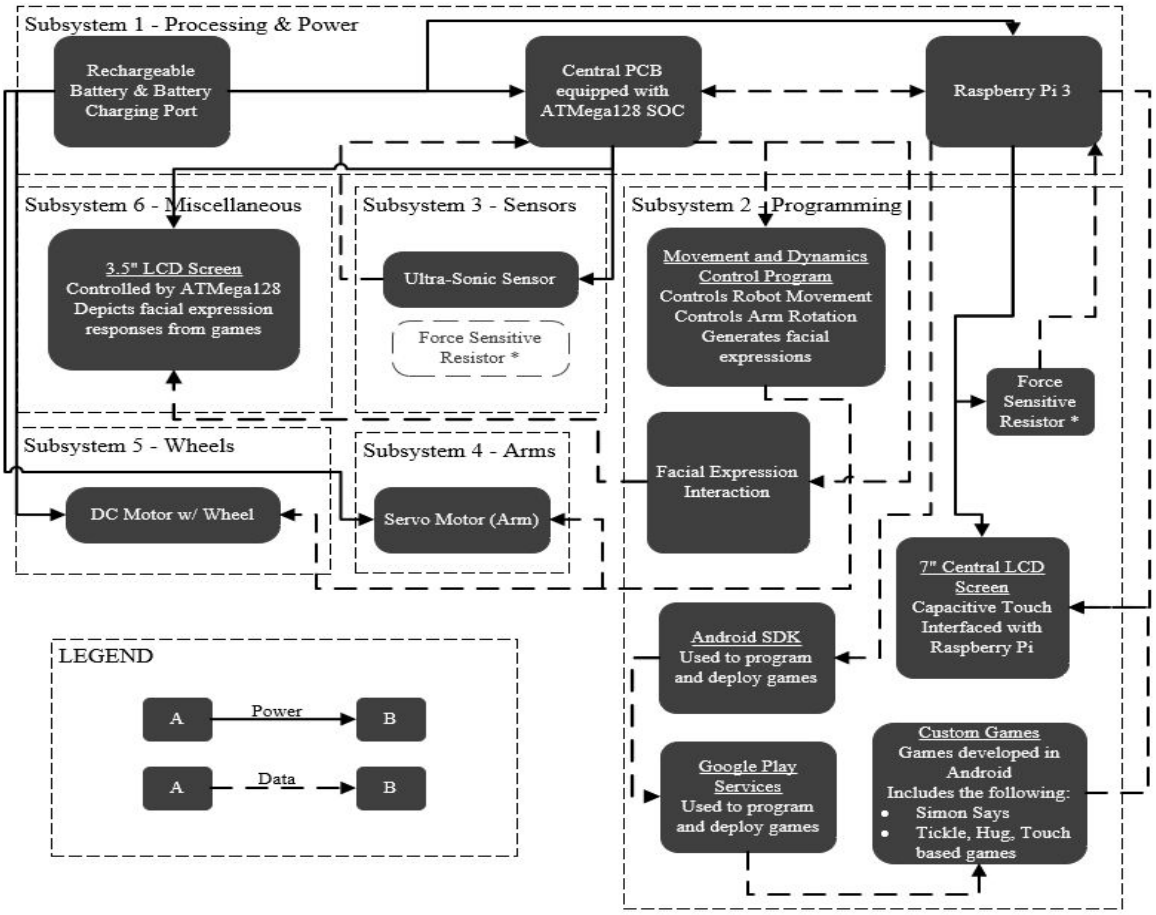

Figure 1: Block Diagram

***** Force sensitive resistor block belongs in the sensors subsystem (Subsystem 3), but was relocated to the programming subsystem (Subsystem 2) to more eloquently display I/O

2. Design

2.1. Design Methodology

2.1.1. Force Sensitive Resistor

Force Sensitive Resistor: Relationship Between Force v. Conduction v. Resistance

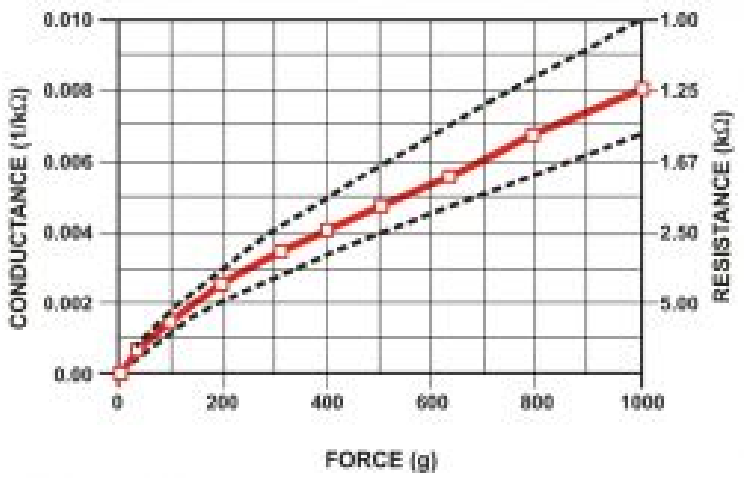

Figure 2: Force Sensitive Resistor Conductance (1/kΩ) vs Force (g)

The force sensitive resistors used for the shoulder sensors in the game demonstrate a positive correlation between the force placed on the sensors and conductance (inverse correlation with resistance)[7]. In order for a child to interact with the robot and play the game effectively, the amount of force needed to trigger a response in the game would also need to be quite small to detect a human touch. Hence, the threshold, as determined from playing the Simon Says game multiple times, was set at 100 grams (g) of force or approximately 0.98 Newtons (N). At this point, the force sensitive resistor that has the pressure applied on it will generate a resistance of just over 5.00 kΩ allowing for an increase in current to the input pins and ultimately generating a digital high on the Raspberry Pi 3 B input pins. Achieving this digital high, since the force sensitive resistors are analog components, required the MCP3008 analog-digital converter with the wiring configuration seen below in Figure 6.

2.1.2. DC Motor Power and Duty Cycle

The power necessary to move the robot back when the Ultrasonic sensor triggers a distance below 9 inches is discussed below:

$$\left(\frac{300\ revolution}{1\ minute}\right)\left(\frac{2.56\ inches}{1\ revolution}\right)\left(\frac{0.0254\ meters}{1\ inches}\right)\left(\frac{1\ minute}{60\ seconds}\right) = 0.32512\ meters/second$$

The reported velocity above is the maximum velocity at which Sydekick will be able to travel backwards. In our arduino code, we check Ultrasonic readings every second so we want to be able to correct the 9 inch threshold over the course of 1 second. Therefore the average velocity is calculated below:

$$\left(\frac{9\ inches}{1\ second}\right)\left(\frac{0.0254\ m}{1\ inch}\right) = 0.2286\ meters/second$$

$$\frac{0.2286}{0.32512} = 0.703125 \approx 70\%$$

The Pulse Width Modulation is maximized at 255. Taking 70% of 255 we arrive at 178.5 PWM. When we tested this PWM we realized that this was not enough to move the robot which is because we did not factor in the weight of the robot when calculating optimal PWM. We started from 178.5 PWM and started testing higher values until we arrived at 255 being the optimal PWM for DC_Motor_1 and 225 or 255 being the optimal PWM for DC_Motor_2. This variation in PWM's for identical DC motors is discussed further in the **Uncertainties** section of the paper. In this section we discuss potential reasons for identical mechanical components functioning differently as well as solutions that might mitigate these problems.

2.2. Subsystem Diagrams & Schematics

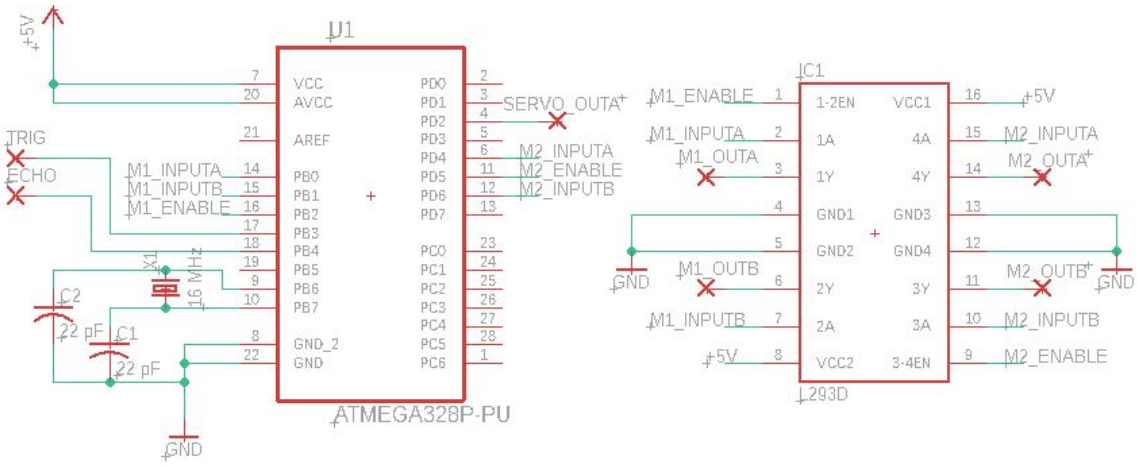

Figure 3: ATMega328P Schematic

Figure 3 shows the schematic used for much of the hardware portion of "Sydekick". The components involved in this schematic are the circuitry necessary for the ATMega to power up and run, the motor driver (L293D), and wiring circuitry for all the other off board components. This schematic was the hardware component of the ultrasonic sensor, both DC motors, both servo motors, and powerlines needed to power the complete design.

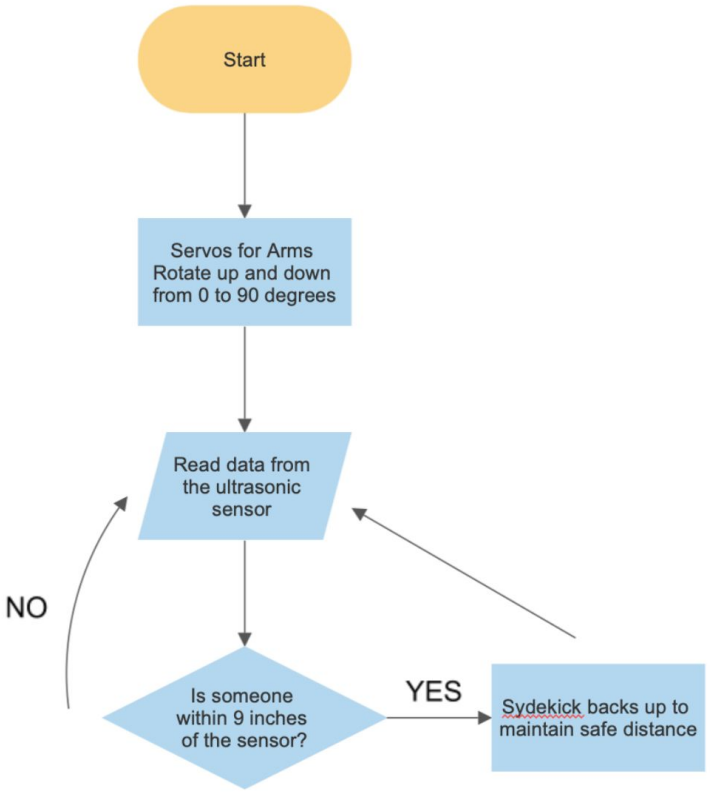

Figure 4: ATMega328P Software Flowchart

Figure 4 represents the ATMega328P software flowchart. On startup of the robot the arms swing up to 90 degrees to show that the robot has limited mobility from 0 degrees to 90 degrees. From then on, the Ultrasonic sensor is taking in readings every second to see if there is an object closer than 9 inches. If there is an object then Sydekick drives back till it hits the threshold of at least 9 inches. If the sensor does not get triggered then it stays in one place.

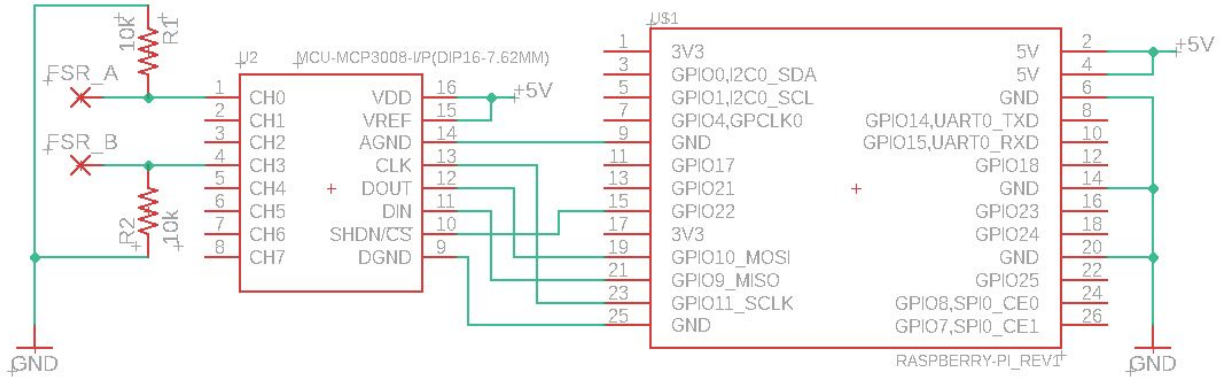

Figure 5: Raspberry Pi 3 B Schematic

Figure 5 represents the schematic for the Raspberry Pi 3 B and the game module of “Sydekick”. The Raspberry Pi is interfaced with the MCP3008 analog-digital converter which takes in the change in current from the force sensitive resistors and converts the analog value to a digital high or low based on a predetermined threshold, noted in section 2.1.1.

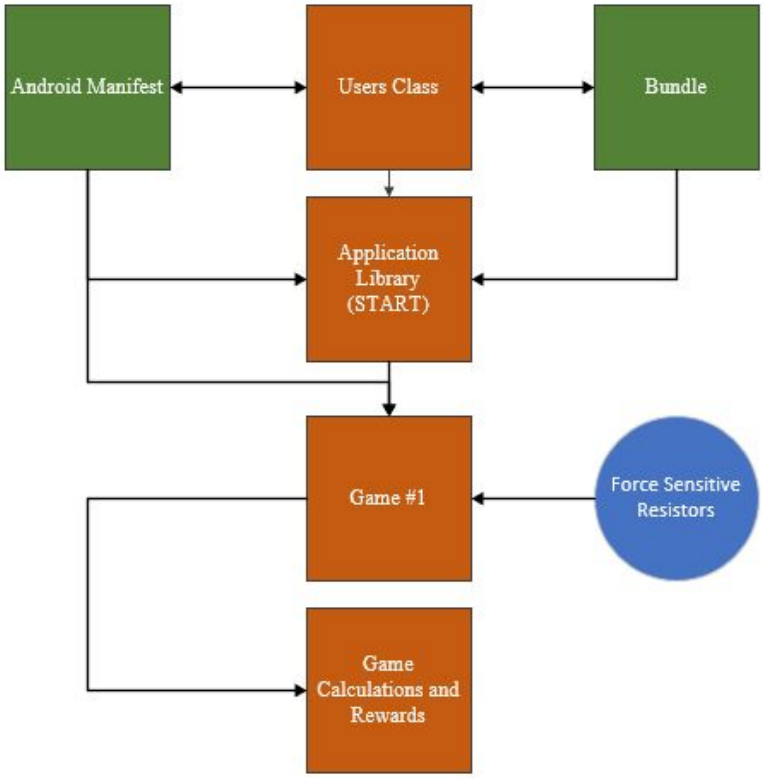

Figure 6: Raspberry Pi 3 B Software Diagram

Figure 6 is the diagram of the software present on the Raspberry Pi B. The Raspberry Pi was loaded with the Android Things operating system and consequently an Android application that contains the game Simon Says within. Upon boot-up, the Raspberry Pi displays the operating system name and then proceeds to enter the ‘Application Library’, which includes a set of games for the user to choose between (for this proof-of-concept design, only one game is present; Simon Says). The first time the software boots-up however, the game will ask the user to enter their name and age to set up the ‘Users Class’ for that particular user. Once Simon Says has been chosen as the game to play, the display gives a Simon Says command and with the use of an on-screen button and the two force sensitive resistors will determine if the command given was followed. If it has been followed, the game counts this as a correct mark, and if not it does not. In the case the game does not determine a correct response, a dialog box emerges allowing the user to try again or to generate a new command. This game can be played for however long the user wants, but upon pressing ‘End Game’, the game will enter the ‘Game Calculations and Rewards’ portion of the diagram and display the performance of the user on the LCD screen and reward the user with an emotionally positive response (i.e. “Good Job!”).

3. Cost & Schedule

3.1. Cost

Product	Cost/Unit	Quantity	Total Cost
Elegoo Uno R3 Arduino	$11.86	1	$11.86
Raspberry Pi 3 Model B	$34.99	1	$34.99
Raspberry Pi 7 Inch Capacitive Touch IPS Display	$61.09	1	$61.09
L293D Motor Drivers Controller	$7.99	1	$7.99
uxcell DC 6V 300RPM Motor	$18.84	2	37.68
16x2 LCD Display Module	$5.99	1	5.99
MCP3008 ADC	$12.50	1	12.5
Ultrasonic Module HC-SR04	$9.59	1	9.59
Force Sensing Resistor	$6.95	1	6.95
Servo Motors	$15	2	$30.00
		Product Tota	**$218.64**
Labor			
$35/hr * 2.5 *(50hrs hardware + 90hrs software)		**Labor Total:**	**$12,250.00**
		Grand Total:	**$12,468.64**

Table 1: Cost Breakdown

3.2. Schedule

Distribution of Work

Date	Bala Chandrasekaran	Sam Feizi	Rohan Mohapatra
2/18	Design Document	Design Document	Design Document
2/25	Gather Parts	Gather Parts	Gather Parts
3/4	Begin on physical design	Begin on Raspberry Pi	Begin on physical design
3/11	Complete physical design	Integration with pressure sensors	Complete physical design
3/18	Implement ultrasonic + data	Create userclass for game	Implement ultrasonic + data
3/25	Work on ultrasonic + wheels	Work on game	Work on ultrasonic + wheels
4/1	Complete arduino code	Complete game	Complete arduino code
4/8	Put together components	Put together components	Put together components
4/15	Bug fix	Bug fix	Bug fix
4/22	Done	Done	Done

Table 2: Timeline and Distribution of Work

4. Requirements & Verification

Subsystem 1 - Programming
ATMega328P

Requirements	Verification	Verification Status (Y or N)
1. PCB should have seamless tracing with components and limit internal interference 2. Microcontroller should be able to	1. No open or short circuits on the PCB that cause the robot to not run altogether 2. Raspberry Pi boots up within 1	1. Y 2. Y 3. Y 4. Y

program the Raspberry Pi auxiliary device at a baud rate of 9600 bps(1) 3. Microcontroller should share a power line with the Raspberry Pi, therefore powering on when the Raspberry Pi does as well.(1) 4. Microcontroller should be able to receive input from motion-based sensors and adjust distance and speed by transmitting to the servo arms and DC motor wheels(1) 5. Utilize on-board memory for sensor data storage and code scripting(1) 6. UART, SPI, and I2C data lines run with a latency of less than 50 ms(1)	minute of robot power-on 3. DC Motor and Servos are functional when microcontroller is powered. 4. Motion sensors respond to changes in proximity and location of the user. 5. Data from the storage and running of code can be checked from a console log of the microcontroller 6. Latency values can be displayed and measured through a console log	5. Y 6. Y

Raspberry Pi 3 Model B

Requirements	Verification	Verification Status (Y or N)
1. Receive power from the central PCB of the robot and be able to also power on the central LCD screen(1) 2. Compile and execute programs related to the game specified by the components of the Programming sub-module (Android SDK and Google Play Services)(1) 3. Be able to compute game progress and reward system(1)	1. When the robot is powered on the central LCD screen will be powered on 2. Game library will instantiate displaying available game to play. Moreover, a game can actually be played and interacted with 3. Reward system and game progress will be displayed on the central LCD screen as an application in the application library	1. Y 2. Y 3. Y

Rechargeable Battery

Requirements	Verification	Verification Status (Y or N)
1. Supply +5V +/- .25V to the PCB as well as all components connected to the battery as specified by the block diagram(1) 2. The robot is able to power on and everything using power is active and working(1) 3. The battery can be recharged	1. Checking of amperage and voltage along all power lines of the robot using an oscilloscope to ensure power and lack of interference 2. Power on the robot, verify the robot can move, arms can move and game is functional 3. Recharge the battery and see a charge level on the central LCD screen be in charging mode	1. Y 2. Y 3. Y

Subsystem 2 - Programming

Central LCD Screen

Requirements	Verification	Verification Status (Y or N)
1. Display the Application library for the user(2) 2. Touch inputs work through capacitive touch(2) 3. Torso LCD should switch to a“Good Job” or“Try Again” based on how game is going(2)	1. Power on the robot and check the central LCD screen if the application library is there 2. Touch the Apps/play a game/touch anything on the interactive parts of the LCD screens and see if it is working 3. Start up a game of Simon Says and play the game and verify based on the answer if the robot is displaying the correct response.	1. Y 2. Y 3. Y

Android SDK + Google Play Services + Custom Game

Requirements	Verification	Verification Status (Y or N)
1. Android applications are able to be programmed and ran through the Raspberry Pi(3) 2. Game is able to launch and run seamlessly (2) 3. The Simon Says app displays in the application library, functional in terms that the user can play the game and have their progress tracked and the robot displays the correct responses depending on the users input(5)	1. Game selected is successfully loaded and can be played through the central LCD screen 2. In playing a game, the game can run without crashing until completion 3. Start up a game of Simon says and have someone play it for 2 minutes and see if the robot displays the correct response as well as tracks the user’s score	1. Y 2. Y 3. Y

Movement & Dynamics Control Program + Facial Expression Interaction

Requirements	Verifications	Verification Status (Y or N)
1. The robot goes a set speed with marginal error (within 2% of the set speed) (3) 2. Face LCD should be able to display facial expressions on the screen when the game is not being played but robot is on (1)	1. Run code that activates DC motors and notice robot moves at a steady/safe speed without any jerking motions. This can be logged by registering the value of the DC motor on the console of the microcontroller 2. Power on the robot but don’t start a game and check the face LCD	1. Y 2. N; Face LCD was not incorporated due to a lack of time on project. With an extra day to work we would have been able to integrate the LCD with the ATMega328P

Subsystem 3 - Sensors

Left & Right Force Sensitive Resistors + Ultrasonic Sensor

Requirements	Verification	Verification Status (Y or N)

1. Pressure sensor should respond to a force on the right or left shoulder by projecting a“Good Job” onto the LCD on the torso(1) 2. Fail state: after Simon Says gives prompt and no pressure data is read in for a period of 15 seconds LCD will display a“Try Again”(1) 3. Detects a minimum of 9 inches separation from user through proximity sensor and will maintain this minimum distance in front of the user when arms are pointing downwards(2) 4. Detects a minimum of 9 inches separation from user through proximity sensor and will maintain this minimum distance in front of the user when arms are pointing towards user(2) 5. Given a distance of less than 9 inches separation from user, Sydekick will reverse till at least 9 inches away(2)	1. Start up Simon Says game and wait for“Simon Says, touch my shoulder” command. Put pressure on corresponding shoulder and wait for LCD reaction 2. Start up Simon Says game and wait for“Simon Says, touch my shoulder” command. Don’t put pressure and wait for time out and LCD to play“Try again” 3. Sydekick should not move forward based on proximity sensor data which will give distance from user of approx. 9 inches 4. Sydekick will lifts arms up and will move backwards until proximity data once again is at least 9 inches away 5. Place Sydekick less than 9 inches from user and then given proximity sensor data Sydekick will reverse	1. Y 2. Y 3. Y 4. Y 5. Y

Subsystem 4 - Arms

Left & Right Arm Servos

Requirements	Verification	Verification Status (Y or N)
1. Arms should be able to move up from 0 degrees(pointing downwards) to 90 degrees(pointing at user)(1) 2. Arms should always be stopping at 90 degrees based on safety measures our team desires(2)	1. Start up robot and watch arms moves from 0 to 90 degrees and stop at 90 degrees 2. Run code to move from 0 to 180 degrees and note that it stops at 90	1. Y 2. Y

Subsystem 5 - Wheels

Wheels

Requirements	Verification	Verification Status (Y or N)
1. Wheels should be able to move backwards and forwards based on data from proximity sensor in order to maintain distance(3) 1. Wheels should be able to make robot move	1. Run code that shows movement of the robot based on proximity sensor data. Give position of robot in front of user, proximity sensor data should make the robot move forward or backwards or stay still 2. Run code that makes robot move forward, backward and turn	1. Y 2. Y

Subsystem 5 - Misc.

Top LCD Screen + Facial Expression Interaction

Requirements	Verification	Verification Status (Y or N)

1. Receive power from microcontroller and receive data from the 'Facial Expression Interaction' software on the microcontroller(2) 2. Face LCD should be able to display facial expressions on the screen when game is not being played but robot is on(1)	1. Turns on when robot is powered on 2. Displays a particular facial expression depending on game reward system 3. Power on the robot but don't start a game and check the face LCD	1. N; Face LCD was not built into the robot because of lack of time. 2. N; Face LCD was not built into the robot because of lack of time.

5. Conclusion

Accomplishments

Our group is very proud of how far we have come with Sydekick. When compared to what we had originally promised at the start of the semester we were able to accomplish almost every requirement that we promised.

In terms of our high-level project functionalities, we were able to successfully accomplish the **Movement** of Sydekick. This involves Sydekick being able to autonomously drive with the two DC motors and the castor wheel. Sydekick was able to drive at a safe speed after testing and also go in straight lines as well as make turns and drive backwards. Additionally, the arms also functioned correctly and aligned with our requirements of being able to swing from 0 degrees to 90 degrees and no further. The second high-level project functionality was the **Game Interface**. A user was able to successfully play a game of Simon Says with Sydekick without any hiccups. This involves all the Force Sensitive Resistors operating properly as well as Android Things functioning accurately on Raspberry Pi. The last high-level project functionality was the LCD monitors. The torso LCD worked as planned and acted as the platform where Users could interact with the robot and play Simon Says. The head LCD was not completely finished but that will be covered in the "Uncertainties " section.

5.1. Uncertainties

There were a few issues that our group ran into through the development of Sydekick.

The first error that we ran into when debugging the drivetrain was the DC motors were not operating correctly when we set them to work at certain PWM's. For example, our group thought that setting both DC motors to a PWM of value of 150 would lead to the wheels spinning. This did not end up being the case; through testing random PWM's our group realized that the PWM's for motors worked when either both motors were set to 255 or one was set to 225 and

the other was set to 255. Any other values than the ones aforementioned would not result in the robot moving.

Through some research of how DC motors worked we realized that DC motors will almost always run at different speeds especially if they are not mechanically connected on an axis. Our drivetrain was created by leveraging a L293D Motor Driver Controller and we connected both DC motors in parallel but detached without using an axis to connect the two motors. From research, it seems like the best solution to fixing the problem where we have unpredictability from DC motors is by adding an extra motor driver controller and operating each DC motor through its own driver as well as making a mechanical connection between the two motors. Another temporary solution is to keep the one motor driver controller design for both DC motors but create some sort of switching arrangement where the driver controls each motor a percentage of the time and ignores the other motor. This is problematic though because each motor would be getting maximum 50% of coverage[6].

The second error that we ran into was with the mechanical design of the base. The back of our robot was controlled by a castor wheel. During the building phase this reduced the mechanical work that we had to do because attaching the castor wheel was a simple addition to the robot. However, during the actual testing phase we realized that having the castor wheel forces Sydekick to give up its sense of direction when driving backwards. If the castor wheel is slightly at an angle than the robot will automatically start biasing at an angle and will not drive backwards straight. This was only observed when the Ultrasonic was triggered for safety reasons and never happened when the robot drove straight because the front of the robot was steered by two wheels. The simple fix to this problem would be to remove the castor wheel and have un-motored wheels in the back. This would allow Sydekick to drive in straight lines easier without having to physically correct th castor wheel.

The third area of concern was the battery distribution amongst the robot. Our original engineering plan was to have an external battery pack that would power the entire robot. This would include all sensors, motors as well as the Raspberry Pi which controls the game. Unfortunately, when we started testing we realized that the battery pack was only capable of powering either the game or the sensors and motors. This was because the battery pack had two ports. The first port had 5V and 2.1A and the second port at 5V and 1A. In order for all sensors and motors to be powered this required the 2.1A source and in order for the game to run properly it required a 2.1A source. Attempting to power both of these modules with the same 2.1A source proved to be not enough. During the demonstration, our group showcased the sensors and motors and then unplugged the battery and then plugged it into the Raspberry Pi to boot up the game. The solution to this was an easy fix because our group could either add another external battery which would have the appropriate amperage or switch battery sources all-together.

The last issue that we ran into was primarily due to time constraints. The original design of the robot incorporated a small LCD where the head of the robot would be to integrate a happy facial

expression. Unfortunately, although we had the resources we ran out of time to finish this portion. If given another day to finish then this portion of the project could definitely have been finished.

5.2. Future Work

If we had more time to work on this project the future work would be divided into the following portions: (1) mechanical build, (2) drivetrain, (3) battery and (4) Simon Says.

Due to the way we timelined the project our group had to build the mechanical portion of the robot on our own instead of utilizing the ECE machine shop. Due to this the robot looked a lot less humanlike then we had imagined. Additionally, we had to saw off large chunks of the robot in order for the DC motor to be able to carry the robot. Given time in the future, our group would love to rebuild the mechanical portion of this robot so that it is lighter and looks more human-like as well as pick out better DC motors that would be able to handle heavier weight so we do not have to sacrifice aesthetics of the robot. Iterating on the drivetrain, our group would like to add two more wheels as mentioned before and remove the castor wheel. This would allow for more control of the direction of the robot.

The crucial problem that we did not expect in the project was the power supply for the entire robot. As mentioned in the “Uncertainties” section a quick fix to the lack of battery power would be to introduce a new external battery. The byproduct of this is making the robot heavier because each external battery pack is rather heavy. A more engineering driven solution to this that could be introducing a Li-Poly battery instead of the external mobile battery packs all together. A Li-Poly battery would be able to offer more power, depending on the demand of power, and not be restricted by the current (voltage would still be 5V) limitations of a USB port.

The biggest changes that we would like to work on in the future would be in the Simon Says game. It would be more engaging if we were able to add more prompts so that each game becomes more challenging. Some examples of this include, utilizing the arms for commands and adding for Force Sensing Resistors. This would be an easy add-on but something a little more complicated might be adding a completely new game. This would include research to figure out what other types of games specialize in helping children learn to follow directions as well as maintain attention spans.

5.3. Ethics

As a preface, “SydeKick” had to be built with extra precaution because of the children that theoretically could be using it, in this case that would be children that have ASD. This section will be divided into robotic specific hazards and then hazards related users interfacing with the robot.

During development it was important to take into consideration the overall appearance of the robot and making sure that we took care of our parts. The first being the mechanical build, with

all parts being smoothed and soldered to avoid live wires and splinters. The second concern is the wiring, integrating the perforated breadboard with the ATMega and Raspberry Pi with the humanoid design with heavy wiring that was all soldered and we used shrink wrap to cover the live wires. Our group was extra careful in making the appearance of the product ready in order to make it as safe as possible for the child. That meant as little danger as possible to the child with the mechanical side of Sydekick.To best solve the wiring and soldering problem it was important as a team to make every part modularly perfect. During usage of the robot it was important to take in to account the the arms and motorized base. Both units were tested to ensure they would not harm the user interacting with Sydekick. Our group solved this by using an ultrasonic sensor that Sydekick uses to detect the user and maintain a safe distance. There will be a safety precaution in the arms so that they don't raise to quickly or too high in order to avoid any harm to the user as well. The restriction we decided was to prevent Sydekick from raising above 90 degrees. Additionally, the motorized base was also fine tuned so that it always stays 9 inches away from the user.

An ethical issue that we could see coming into play is whether or not it makes sense to leave a child with autism to play with a robot. Robotics is a very fast-paced and constantly advancing area of study which is not always fully understood. With this being said it creates an ethical puzzle of whether or not it's okay to push the responsibilities of people to nurture kids to a robot. I think this problem is shown in IEEE Code of Ethics (#5) where the goal of IEEE as a whole is to help improve the understanding and capabilities of individuals when it comes to new and emerging technologies[3]. The best way "SydeKick" will avoid this issue is to create an environment where parents need to be around with the child while they are playing and having therapy sessions but allow enough space to where their full attention is not commanded. As an emerging technology continues to grow its important to start giving more responsibility to technology while also not completely unlatching. IEEE Code of Ethics (#9) will be another really important code to keep in mind because safety has to be the paramount priority. Throughout building as well as usage by consumers this project will take every precaution necessary to maintain safety standards. This will include following lab safety trainings as well as taking the best care of our equipment and documenting our processes.

Testing the product will be another ethical issue that we have considered. Getting authorization to work with children with autism is very tricky and the liabilities that surround it make the environment difficult. Because of this, our team found literature that shows in theory playing these kinds of games with children on the spectrum does indeed improve their following direction skills as well as increased attention span.

References:

[1] “Data & Statistics on Autism Spectrum Disorder | CDC .” *Centers for Disease Control and Prevention,* Centers for Disease Control and Prevention, www.cdc.gov/ncbddd/autism/data.html

[2] Weir, William. “Robots Help Children with Autism Improve Social Skills.” *YaleNews*, 22 Aug. 2018, news.yale.edu/2018/08/22/robots-help-children-autism-improve-social-skills.

[3] “IEEE Code of Ethics.” *IEEE - Advancing Technology for H*umanity, www.ieee.org/about/corporate/governance/p7-8.html.

[4] “Default - Stanford Children's Health.” Stanford Children's Health - Lucile Packard Children's Hospital Stanford, www.stanfordchildrens.org/en/topic/default?id=interacting-with-a-child-who-has-autism-spectrum-disorder-160-46.

[5] “Comprehension of the Message: Important Considerations for Following Directions.” IIDC - The Indiana Institute on Disability and Community at Indiana University, www.iidc.indiana.edu/pages/Comprehension-of-the-Message-Important-Considerations-for-Following-Directions.

[6] Gsdaemon, et al. “2 DC Motors Connected in Parallel to H-Bridge Move with Different Speed.” Electrical Engineering Stack Exchange, electronics.stackexchange.com/questions/331263/2-dc-motors-connected-in-parallel-to-h-bridge-move-with-different-speed.

[7] “Force Sensor.” Force Sensor [Robotic & Microcontroller Educational Knowledgepage - Network of Excellence], 27 Dec. 2018, home.roboticlab.eu/en/examples/sensor/force.