LEGO EV3 Prosthetic Hand

Group 8-5

Yaseen Mohamed, Sara Alrifai,

Jack Doehler, Evan Janakievski

MTE 100 and MTE 121

December 5, 2023

Table of Contents

Summary

With prosthetics being a common application of robotics, the team decided to design, build, and test a prototype prosthetic hand. The majority of the robot was constructed with LEGO components, including LEGO EV3 motors, sensors, and an onboard EV3 computer, while a 3D printed hand grip was attached to the base for a user to use to lift and operate the robot. The goal of this robot was to have fingers that were able to open and close around objects and aspects of the local environment. Sensors would be used to autonomously detect, grab, and pick up objects of medium size and weight. However, the autonomy could be overridden so that the user could control the fingers with manual input.

The team sought to make the design lightweight, compact, and easy to operate for a potential user, so the design of the robot went through several iterations before completion. The first concepts for the design included, notably, five motorized and flexible fingers, as well as a wrist that would have the ability to rotate. These aspects were not possible to build given time and materials and would have also made the robot too heavy and bulky for general use, so the design was reduced to having only three fingers and giving the role of rotation to the user's arm. The team had also initially planned to secure the robot to the wearer's forearm, but this would have been difficult to do safely while also keeping the robot from shifting its position during operation, so a handle was attached instead so that it can be held.

The working final product used an ultrasonic sensor to detect the distance of an object from the palm of the robot's hand, which would trigger the fingers to open if that object was in close proximity. The palm of the hand was a large plate that, when pushed, would activate a touch sensor that was used to close the fingers around any object that pressed down on the plate. The fingers would then grip an object such that it could be lifted into the air without falling. The user would be able to request the fingers to release the object via a button press, and the robot would do so if an accelerometer sensor, which was placed on the hand, detected minimal movement of the robot as a whole. The robot was also able to be switched to a mode of manual control, where the fingers could be opened and closed via separate button presses. The robot was also programmed to gesture a type of wave when a specific button was pressed. Two shutdown procedures were also implemented, one of which was an emergency shutdown that could be activated at any time that would throw the fingers fully open; this was intended only to be activated if the robot was causing damage or harm. The other shutdown procedure would slowly close the fingers and could only be activated if the robot was not actively moving the fingers or gripping an object.

The robot was able to complete all tasks assigned to it while maintaining ease of use and a compact and lightweight form. Its functionality was tested by having it pick up several objects, specifically two types of water bottles which were the main subjects, which it did with relative ease. The project was completed on time, although there were some delays with aspects of it that required more time to be allocated to building it towards the end of the project period. Future iterations of this design could include stronger, more flexible fingers, but these could have been accomplished given more time and higher quality materials. The code running the robot could also be optimized to run more quickly, smoothly, and perhaps more autonomously in order to make the robot more efficient and easier to use.

Acknowledgements

The accelerometer header file provided on the MTE 121 LEARN page was used in the main code to configure and run the accelerometer sensor used with the robot.

List of Figures

1.0 Introduction

Robotics is becoming more and more applicable in the field of prosthetics to assist those with limited motor control or a physical disability. Restoring the function of a human hand, for example, that a person may not have been born with, lost through an accident, or had to have amputated would greatly increase the ease with which one would be able to complete daily tasks. About 2 million people in the United States are living with limb loss or limb difference [1]. While this total does not represent those with upper body limb difference, it speaks to the immense quantity of people looking for technology to help them in their day-to-day life. With this being an issue that spans across the globe, designing a prosthetic hand replacement that is easy to use and inexpensive could make the technology accessible to more people. Robotics researchers at Johns Hopkins University created the Modular Prosthetic Limb which could accomplish almost any task a human hand could [2]. This limb is controlled with the use of electrodes that are placed on muscles and detect muscle activity and nerve signals and translate these readings into actions. Decoding these signals has taken extensive time and research, but progress has been made in recent years.

With limited time and materials, building the simplest, but most effective prosthetic prototype using primarily LEGO components was the goal of the project in this report. Reducing the potential size of the prosthetic resulted in the use of 3 fingers, each driven by a separate LEGO EV3 motor, as the motors were large and made it difficult to keep the design compact. Because the robot required the manual input of pressing buttons to activate certain parts of the robot, the design of this prosthetic varied from outside research due to the simplicity of the motors used and the scope of the MTE 100/121 courses. Due to the weight of the robot and the difficulty of attaching it securely to the user's arm, the design required the user's hand to hold onto the robot so that the functionality of it could be tested easily. Making the robot able to lift objects of different shapes and sizes using only sensor input was the central goal of the project. The sensors included were an ultrasonic sensor, touch sensor, and accelerometer. Using these sensors, objects were able to be picked up and held onto autonomously and were released at the push of a button under the condition that the robot was not shaking or moving.

2.0 Scope

Replicating the function of a human hand is a challenging problem. The goal of this project was to create a prototype design for a prosthetic hand that could, in the future, assist those with limited motor control or those without a hand. The robot was intended to have the ability to detect objects directly in front of it and grab and hold on to those that are medium-sized. Autonomous and ease of use by the user was the main goal, although some of the robot's functionality still relies on manual input from button presses.

The sensors on the robot are programmed to assist the user in grabbing and holding onto objects autonomously. An ultrasonic sensor was used and placed beside the palm of the hand to detect the distance of objects from it by constantly returning the distance of an object in its path, and the EV3 brick reads and compares this value with a distance set by the user in the code to determine if the object is “close” enough. The distance that the EV3 brick is waiting to read in this case is 20 centimeters or less. An accelerometer was attached to the hand to detect its movement by measuring the acceleration in three dimensions, x, y, and z, and, by extension, the tilt of the hand. A touch sensor was placed on the robot to detect when the hand comes into contact with an object and is triggered when a plate on the palm of the hand is depressed. The EV3 brick was used to take in button inputs from the user and complete specific actions when the other sensors cannot be used.

The robot will interact with its environment in many ways. The sensor inputs allow the EV3 brick to analyze its environment and determine which actions to complete in various scenarios. In addition to the sensors, the robot is equipped with motors that allow it to interact with objects and the environment around it, as grabbing and lifting objects is one of the main functions of the robot. The motors on the robot consist of two large motors for each of two large fingers and one medium motor, which controls a smaller finger, or thumb. The motors are programmed to open the fingers fully when the ultrasonic sensor measures an object in front of it at a distance of 20 centimeters or less and close them 3 seconds after opening unless an object remains within this distance. The motors are programmed to have the fingers closed by default, so they are activated to close the fingers after the completion of each action or gesture. The motors are also able to be controlled manually with the use of buttons on the EV3 to open and close fingers at the user’s will without sensor input.

The use of a “mode” variable in the robot’s software allowed the robot to record which it is in and return to its default state after completing a set of tasks in a specific mode. The different modes that the robot could be set to were searching for objects, gripping an object,

manual control, gesturing a wave, and shutdown initiation. There were two different shutdowns, both triggered by button press, that could be initiated, and each affected how the robot was powered off. The regular shutdown closes the fingers slowly and is only able to be activated when the robot is not currently completing a task like opening or closing the fingers, gripping, or gesturing. The emergency shutdown, however, can be initiated at any time and throws open the fingers in case there is a need for the robot to release anything immediately, such as if the fingers are causing damage or harm to another person.

Throughout the design process, many changes were made to the original plan. The main adjustment was altering the way the robot was attached to the user. Instead of clamping the robot to their forearm, which would have replicated a prosthetic more closely and was one of the original goals of the design, a handle was attached to the base of the hand that required the user to hold on to lift the robot. This change was made primarily because it would have been difficult to find a solution to securely mount the robot on the user's forearm given the time and materials allotted, and the hand-grip design was safer and a more practical way to test the functionality of the hand.

In addition, having five articulated fingers attached to cables and using a motor to control each was the original idea for the structure of the hand. However, using five motors would have added a lot of weight and would have made the design very bulky, which would have been counterproductive to having a slim and lightweight design. A revolving wrist was another initial idea that the team had, but this could have caused entanglement issues to the wiring running from the hand to the EV3 brick, so the idea was not pursued; also, the user is able to rotate the hand manually, so the wrist rotation would have been redundant anyway. A semi-opposable thumb was another initial idea but was too complicated to design given within the project deadline and was instead adjusted to be on one axle to keep the robot compact, functioning similarly to the other fingers. The use of an accelerometer replaced the original plan of using a gyro to measure the movement of the robot, but it only measured rotation in one axis, which was not sufficient. Lastly, using a colour sensor to detect the colour of objects was another idea, but there was no practical use for it that the team could decide on, so it was not included.

3.0 Constraints and Criteria

3.1 Constraints

While designing this prosthetic, some constraints were kept in mind. The ability to pick up medium-sized and medium-weight objects was critical. The original plan was for the hand to be able to pick up all types of objects, but during the construction of the fingers, it was determined that small objects, such as an orange or small computer mouse, were not able to be picked up by the fingers. Also, if an object was too large, the fingers were not able to wrap around it and have enough grip to lift it. The weight and material of the objects also determined what could be picked up. The goal was to be able to lift a 5-pound object, however smooth objects were difficult to grip in many cases regardless of weight and if an object was too heavy there was not enough force applied by the motors to the object to hold it, or enough friction from the fingertips to keep it from slipping. So, due to the lack of strength of the LEGO EV3 motors, the ability of the robot to pick up objects was limited to medium-sized objects with a weight of 5 pounds or less.

Another limitation of the design was the size and weight of the robot itself. In order to function as a proper prosthetic, the robot would have to be lightweight so that any potential adult user had the strength to pick it up with ease and would not tire of wearing it. It would also have to be similar in size and shape to a human hand because it would have to interact with objects and an environment that a regular human hand would also have to interact with.

3.2 Criteria

Soft requirements for the robot consisted of different tasks that it needed to complete, as well as its usability and accessibility. The ability to open and close the fingers was central to the success of the design; without this, objects would not be able to be grabbed and picked up. The robot also had to be able to detect, grab and hold onto objects so that it could properly interact with the environment around it the way a human hand would.

Another requirement for the robot was for it to have the ability to autonomously pick up objects as well as manually, so that in any case the user would be able to use the robot as they desired; in other words, every action done by the robot should be intended by the user. The autonomous control would use the sensors on the hand and pre-programmed motions and actions to control finger position movement. But with a button press, the state of the robot would also be switched to a manual control mode where the fingers could be controlled with button presses alone.

Lastly, the robot would not work successfully if it was not easy for the user to control. For example, the user should be able to grab onto the robot and lift it up easily. The screen on the EV3 should also display directions and information about the state of the robot to inform the user about what task is currently being executed. This would allow the user to understand which mode the robot was in and how to properly use the robot. When started up, the robot would also have to initialize the fingers no matter what position they were in previously.

4.0 Mechanical Design and Implementation

4.1 Overall Design

The overall design of the robot was aimed to be compact and lightweight. With the materials provided, these goals were difficult to accomplish without some tradeoffs from the original plan and ideas. While the construction of a prosthetic hand remained the central goal of the project, the final design shifted to becoming more like a prototype to test the autonomous controls and ability to pick objects up. The final design used two fingers powered by large EV3 motors and a smaller “thumb” driven by the medium motor. The user in the final design had to hold onto a handle attached to the bottom of the robot during use as clamping the robot to the user’s forearm was not feasible. Figure 1 shows the 3D printed parts that were used to create the handle and a shell for the handle to be contained in. The placement of the LEGO EV3 brick was attached with Velcro to the 3D printed case which allowed the user to see the EV3 computer’s screen and interact with the buttons while also ensuring that there were no long cables that got in the way of operation. While not all the LEGO components could be fit into a compact package, the placement of motors and sensor were integral to ensure it was as compact as possible as seen in Figure 1.

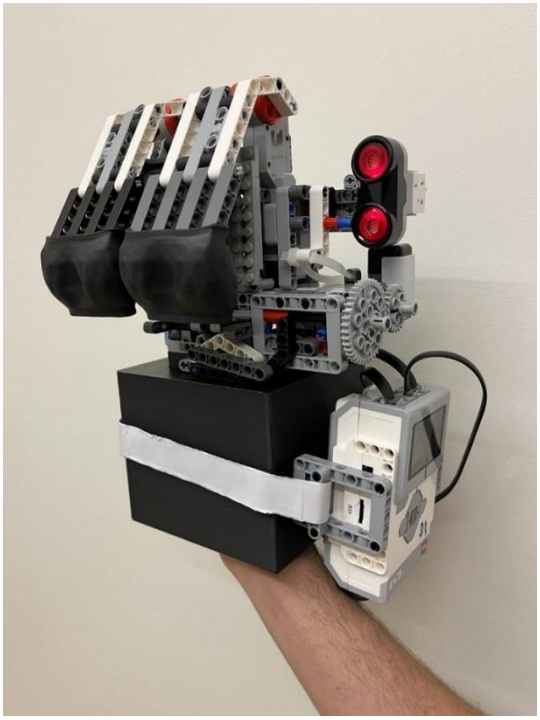

Figure 1: The complete robot in its powered off and stationary form.

4.2 Chassis

The chassis of the robot was constructed using LEGO beams and pins. The focus on the chassis' design was placing the motors and sensors in locations that would be most accessible while keeping the form compact. The connection between the pins and the beams were not strong enough on their own at some locations and required more pieces to hold down. The aim was for the robot to be able to lift an object up to 5 pounds, so each part had to be securely connected. To secure the hand to the 3D print, zip-ties were used as solely LEGO pins were not strong enough to connect the two parts. The anchor points for the zip-ties were the 3D printed handle inside of the shell and the beams at the bottom of the structure. To make sure the weight of the object did not pull apart any connections, zip-ties were also used to secure the chassis in other locations. (Figure 2)

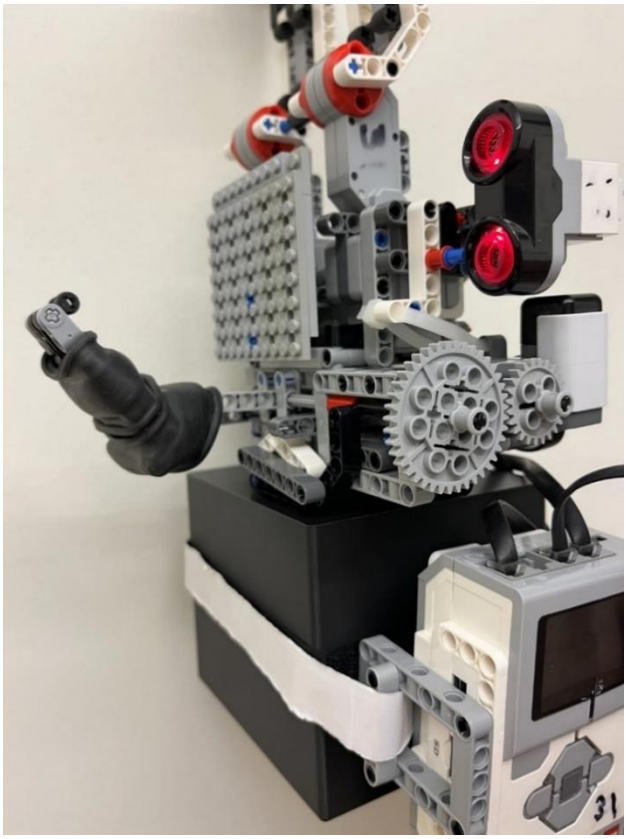

Figure 2: The chassis is built of LEGO beams and pins and allows for each component to branch off and be connected.

4.3 Motors and Fingers

4.3.1 Large Motors

Each motor controlled an individual finger. Two large motors were used for the two main fingers and a medium motor was used for the thumb. The main fingers were attached and driven directly from the motors to help keep the design compact without the use of string and jointed fingers. Beams and pins were used to construct the skeleton of the fingers. The beams that had two bends in the piece provided the best shape to the fingers and helped them have an arched shape. However, with LEGOs not having much grip, LEGO tires were also inverted on top of this skeleton to provide the fingers with a much better grip. The original plan was to make use of small rubber pieces in the kit; however, the tires provide better traction and grip for objects of different materials. The fingers were constructed to have a large surface area to make up for the lack of fingers in the design. The greater surface area provided more friction, which reduced the amount of torque needed from the motor to hold an object. For areas that the tire did not cover, small rubber pieces were added to provide some extra traction. Between the two fingers there were two pins placed to help secure the fingers in place, but also allow them to rotate independently. This assisted in keeping the motors still and locking them down while lifting an object to make sure they did not get pulled apart. (Figure 3)

Figure 3: The two large motors connect to one finger each.

4.3.2 Medium Motor

The thumb was controlled using the medium motor which was placed underneath the large motors. Figure 4 shows how the thumb was not directly driven from the motor and gears and axles were used. Different gear ratios were tested to find which provided the most stability for the thumb, and it was found that a smaller gear attached to the motor, then connected to a larger gear provided the best stability and strength. The torque from the arrangement of gears was greater than if it was in the alternate positions. The smaller gear acts as a lever and allows for more torque while providing a lower speed. The axles for the thumb, however, provided some issues as they are constructed out of plastic, thus they had some give and could over rotate causing the axle to twist and the thumb to not be in the correct position. Constructed out of beams and pins, the thumb had a similar structure to the main fingers. The thumb also had an inverted tire placed on it to provide a better grip while holding objects, similar to the other fingers.

Figure 4: The medium motor rotated the thumb using gears and axles.

4.4 Sensors

4.4.1 Touch

The touch sensor design aimed to be compact while not getting in the way of the motors. It is mounted below the two large motors and the touch portion is facing the back of the hand. This was done because of the method used to detect touch. A touch pad was placed in the palm of the hand, as shown in Figure 5. The touch pad is constructed of flats and pins and was low profile and did not interfere with the fingers. It was mounted on beams that ran to the back of the hand but was allowed to pivot when pressed. This pivot would rotate another beam that would then press the touch sensor. This method allowed for more gentle presses to be sensed due to the rotational torque of the system acting with greater force on the spring-loaded button on the touch sensor. When the original design had the touch pad directly connected to

the touch sensor, it was difficult to press and required a substantial force to be pushed in when pushing lightly. This design also balances the touch pad more successfully than the original methods. Each corner of the pad could be pressed and sensed without breaking the touch pad off.

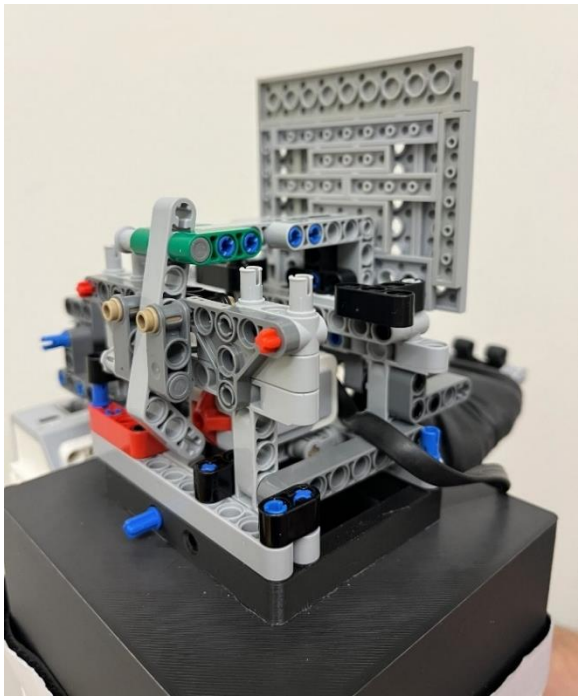

Figure 5: The touch sensor's configuration with a view without the two finger motors, and from an angle showing the back of the hand.

4.4.2 Ultrasonic

The ultrasonic sensor was used to determine when an object was close to the hand. If an object was within 20 centimeters, then the hand would open. To best allow for the distance and location of an object to be measured, the ultrasonic was placed next to the palm. It was attached using multiple LEGO pieces, however, a challenge with LEGO is that it requires multiple pieces to create certain angles. To mount the ultrasonic at an angle, it was placed on a pin that would allow it to swivel with a three tall blue-pin placed next to it limiting the total rotation. Figure 6 shows this pin. To further ensure the ultrasonic stayed at the desired angle, a 2x1 rubber piece was used and press fit into the ultrasonic sensor's pin attachments, not allowing the ultrasonic to rotate back into place.

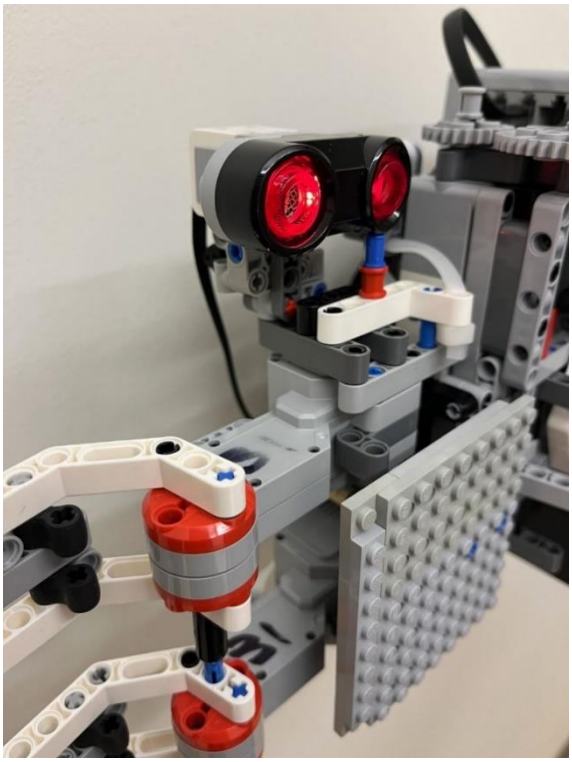

Figure 6: The ultrasonic sensor attached to the chassis and rotated to the center of the palm aiming at a point 20 centimeters away.

4.4.3 Accelerometer

The accelerometer was used to measure the movement of the hand in all three dimensional axes and determine if an object being held should be released. The accelerometer was mounted on the back of the hand using LEGO pieces. Figure 7 shows the placement of this sensor which keeps it out of the way of the other features of the hand. The orientation of the sensor, while mounted, affected how the values of each axis would be read for the program. The z-axis was facing down and the x-axis was facing forward when the hand was facing forward.

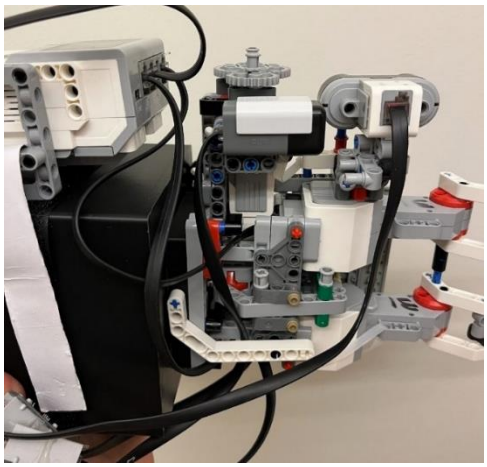

Figure 7: The accelerometer sensor mounted on the back of the hand using LEGO pins.

4.5 3D Printed Parts

To make the handle, three parts were individually 3D printed using a Prusa MK4 printer and were glued together to make the assembly. The original design plan was to use M3 nuts and bolts to attach the parts, but they were not able to be attained as planned. The printed parts were designed in SolidWorks to be user friendly and make the interaction with the robot seamless. Shown in Figure 8, the handle was convex which allowed the user to grip it easily, keeping comfort in mind.

Figure 8: The handle connected to the shell to allow the user to hold the robot.

4.6 Overall Assembly

The assembly of the robot used LEGO pins and axles to connect each part, but also some zip ties due to the weight of the assembly. The 3D printed parts were connected using LEGO parts that fit into a 3D printed collar for the handle (shown in Figure 9). This allowed minimal materials that were not LEGO to be used. The whole chassis of the hand weighed more than what could be supported by the LEGO pins. The friction in some of the connectors was not enough to support the motors and sensors. To secure the chassis down, it was zip tied to the 3D printed handle. Components were also zip tied to other parts to make sure none of the pins failed. The two large motors were secured down as well as the ultrasonic sensor. The EV3 brick was attached to the outside shell of the 3D printed handle. Velcro was used to allow for the brick

to be removed easily; it was looped through LEGO pieces on the side of the brick that then attached to pieces of Velcro on the 3D print as seen in Figure 1. This placement of the brick allowed for easy access with the left hand to the buttons needed to activate certain functions.

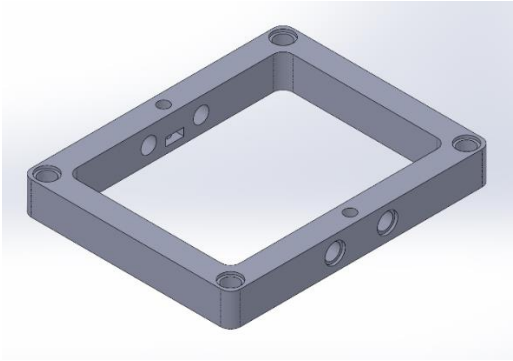

Figure 9: The piece that was connected to the handle's shell to connect with the LEGO parts.

4.7 Trade-Offs

Many tradeoffs to the physical design as the robot was constructed. One of these was the decision to switch to the user's need to hold the robot. To replicate a prosthetic, the original design used 3D printed pieces to clamp to the user's forearm seen in Figure 10. The EV3 brick was also planned to be attached to the top piece allowing the user to have access to the buttons if needed. Elastic straps were to be wrapped around the arm and then buttoned down on the other side of where they were attached.

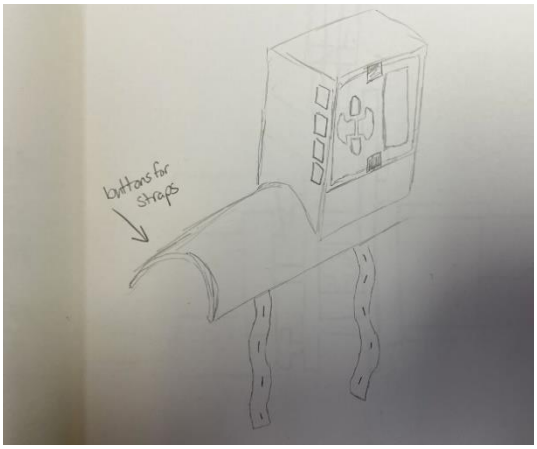

Figure 10: The sketch for the top of the arm clamp that would also hold the EV3 brick.

For the bottom piece, the original plan was to have the ultrasonic sensor mounted under the forearm as seen in Figure 11. This design was adjusted as the method of how the user would use the robot had changed. With the user holding the robot, and the new placement of the thumb, it was no longer feasible to mount the ultrasonic underneath the forearm. Both design changes resulted in a more compact design and ease of grabbing the robot to startup.

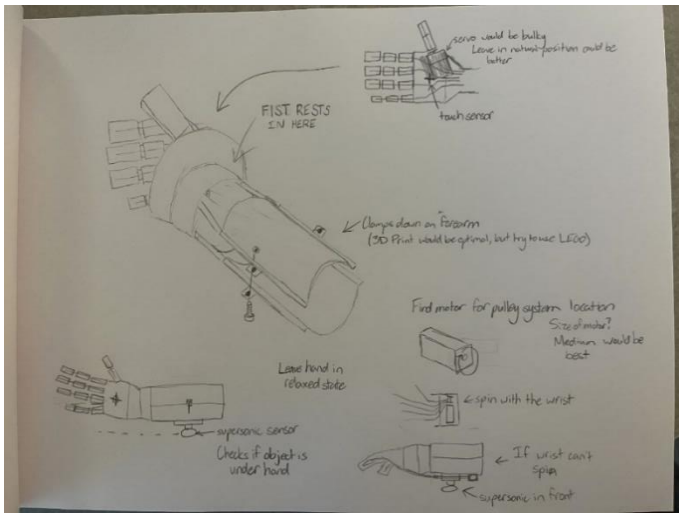

Figure 11: A sketch showing the original placement of the ultrasonic sensor. Mounted underneath the original thought of a clamp for the forearm.

Another trade-off included the change from five fingers to three. The original design included five fingers using string to control the joints. However, the quantity of motors would result in a large and bulky design. Having individual joint control for each finger was beyond the scope for the timeline of the project. The thumb was also planned to be mounted on a servo to allow it to wrap around objects better. The original design for the wrist was to have it swivel and be able to turn to an angle the user wanted. With the robot clamped to the forearm, there would be limited rotation of the robot the user would be able to do without this built in. These ideas can be seen in Figure 11.

5.0 Software Design and Implementation

5.1 Description of Main Code and Functions

See Appendix A for the complete main code for the robot, including all functions.

5.1.1 Main

When the program is initially run, all four sensors are configured and the screen displays a message saying “PRESS ENTER TO START”. Then, the program does nothing until the enter button on the EV3 brick is pressed and subsequently released, after which the screen will display a new message: “BOOTING UP”. Messages like these corresponding to the mode that the robot is in are used throughout to inform the user what the robot is doing; this was also helpful for informing the team during testing and debugging. The “initializeFinger” function is called for all three finger motors and several variables are created. A variable for the “mode” is created and set to 1 and is referenced both in main and in several functions. The value of this variable determines the state of the hand and which processes it should run as the program progresses. A boolean created called “closed” is used to store the state of the fingers as being open (false) or closed (true) and is initially set to false (due to the fingers being opened during initialization); it is also referenced both in main and in several functions.

The first loop of the function repeats while the mode variable is equal to 1. First, the EV3 screen displays “Checking for object to grab…”, after which the program will loop the “checkForObj” function until it returns true (indicating that an object is within range of grabbing) or any button on the EV3 brick is pressed. Following this, a series of conditions are checked. Firstly, if the up or down buttons have been pressed, then the mode is set to 0, the value that begins the shutdown procedure. If, instead, the left button has been pressed, the mode is set to 3 and the “control” function is called. If, instead, the right button has been pressed, the mode is set to 4 and the “wave” function is called. Otherwise, if the fingers are open and the mode is still equal to 1 (which it will be in any case), then each finger is closed and the mode is set to 2.

The next loop repeats while the mode is set to 2. In this loop, the gripObject function is called. When this is exited (which will only happen due to a button press), if the “up” button was pressed on the EV3 brick, the mode is set to 0. If the “enter” button was instead pressed, the program waits for it to be released and then calls the “isMoving” function. If it returns false (indicating that the hand is not moving), then all three fingers are opened and the mode is set to 1. If the function instead returns true, then the screen displays the text: “Cannot release while hand is moving”.

If the mode is set to 0 at the end of either of these loops, then the program will exit out of the first loop and proceed to execute one of two shutdowns. The program is designed to exit out of this loop if the “up” button on the EV3 brick is pressed at any point, and if the “down” button is pressed while the hand is not performing any actions such as gripping or gesturing. At this point, if the “down” button has been pressed, then the program waits for it to be released, then the EV3 screen displays “SHUTTING DOWN…” and closes all fingers slowly. Otherwise, the EV3 screen displays “EMERGENCY SHUTDOWN INITIATED” and opens all three fingers quickly. After either of these actions, the program waits for 1 second before shutting down. The code was written by Jack and Evan. A flowchart for the code is shown in Figure 12.

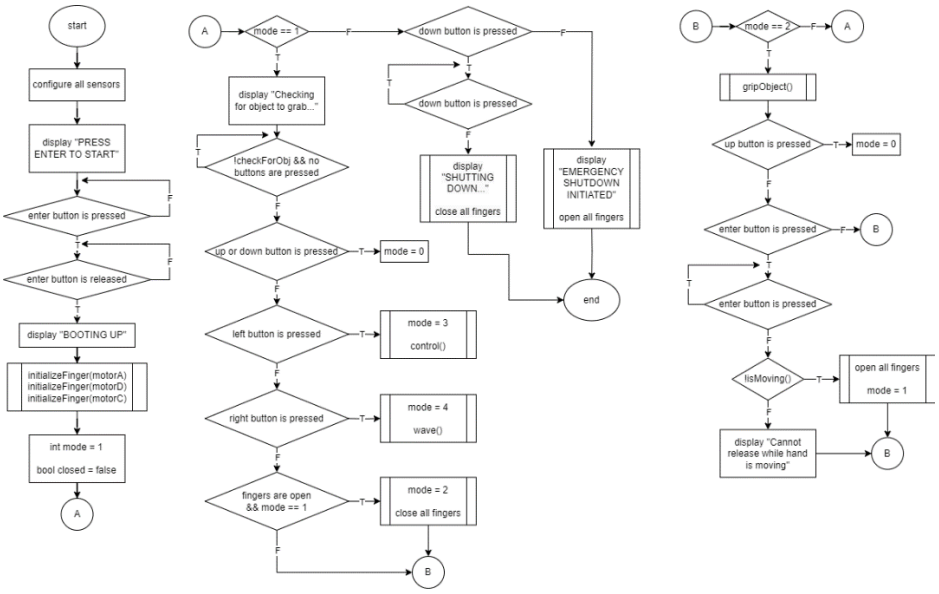

Figure 12: main task flowchart.

5.1.2 Finger Initialization

A function to initialize an individual finger was created in order to zero the motor encoder of that finger when it is fully opened. The only parameter of this function is the motor name itself (“motorA”, for example), so the function is called three times at the beginning of the main code, once for each finger motor. The motor speed is set to a small negative value if the motor name called in the parameter is motorA or motorD – either of the large fingers – in order to open it slowly, whereas if motorC is called for the thumb, the motor speed is set to a small positive value due to the gear system controlling the thumb reversing its open/close direction. Following this, the function is programmed to wait for a fraction of a second while the finger begins to move before calling the stopFinger function to detect when it stops. After the finger comes to rest at a fully open position, the motor encoder for that finger’s motor is set to zero and the

function is exited with no return value. It is necessary to zero the motor encoders of each finger so that their position at any point during the program is numerically equal between separate trials of the code running. Other functions, such as openFinger, opens a finger until the motor encoder value of motor corresponding to that motor reaches 0, which, if the finger was not zeroed to a particular position prior to the openFinger function being called, could result in the motor encoder equalling zero at an unknown position and this could result in the function not opening the finger properly. A flowchart for the code is shown in Figure 13. This function was written by Jack.

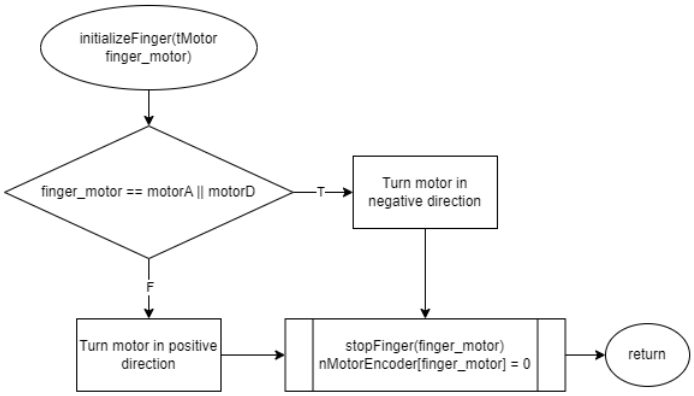

Figure 13: initializeFinger flowchart.

5.1.3 Close Finger

The function to close an individual finger works similarly to the initializeFinger function, only the motor speeds are reversed and variable. The parameters of the function include the motor name and an integer input for the motor speed. If the motor name passed is motorA or motorD, the respective motor is set to motor speed set in the function parameter; if motorC is called, the motor speed is set to the same value but negative in direction. Alternatively, the motor name in the function parameter can be called as “ALL”, which will simultaneously set the motor speed of all three motors equal to the motor speed set in the function parameter, although that thumb motor speed is set to 1.5 times this speed to allow it to close under the larger fingers. Calling all of the fingers to open at once is used several times in the code, so adding this part of the function eliminates the need to call the function three times for each finger individually. After this, the stopFinger function is called either for a single finger if only a single motor was called in the function parameter, or for all three motors individually if that parameter was “ALL”. The function then exits with no return value. A flowchart for the code is shown in Figure 14.

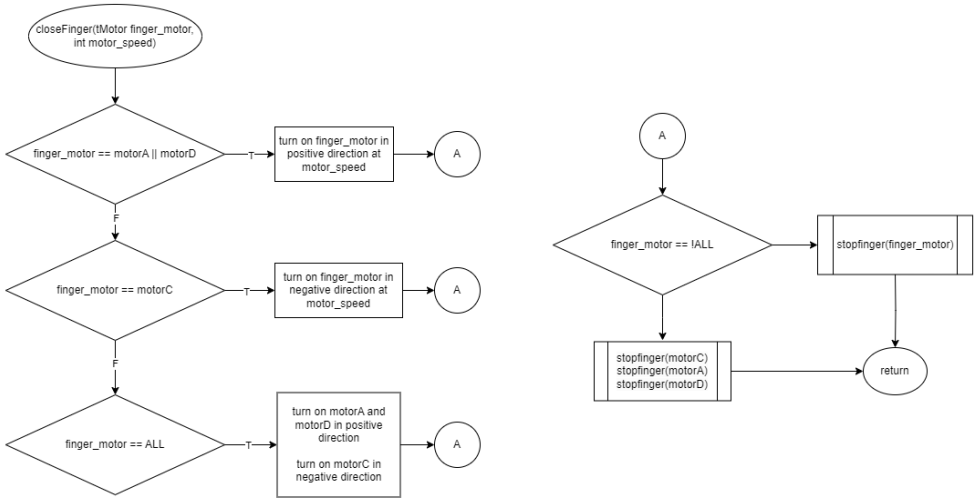

Figure 14: closeFinger flowchart.

5.1.4 Open Finger

The openFinger function is used to open the fingers fully and passes in values for a motor name and an integer value for speed. This function works similarly to the “closeFinger” function. If the motor name called is motorA or motorD, that motor is set to the negative motor speed set in the function parameter and then a loop is called that does nothing while the encoder of that motor is greater than 0 and no button on the EV3 brick is pressed (although the scenario of a button being pressed is only expected to occur if the robot needs to be shut down before this loop is exited). After this loop is exited, the motor power is set to zero. Alternatively if the motor name called is motorC, the same process occurs, but the motor speed is positive. If, instead, the motor name in the function parameter is called as “ALL”, then the two large fingers are set to a negative speed, followed by motorC, which is set to a positive speed after a short delay to ensure the fingers don’t catch on each other while opening. Then, for each motor, the function does nothing while that motor has an encoder value greater than zero and no button on the EV3 brick is pressed, after which it sets that motor’s power to 0. Once this process has been completed for all three motors, the function exits with no return value. A flowchart for the code is shown in Figure 15. This function was written by Evan and Jack.

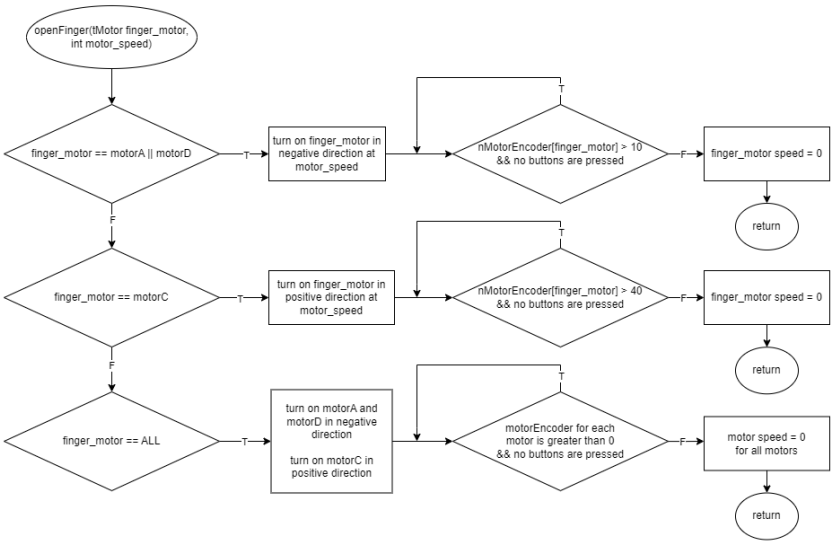

Figure 15: openFinger flowchart.

5.1.5 Stop Finger

The function to stop a finger is used to turn off a finger's motors when it can no longer physically open or close any further. The only parameter for the function is the motor name that that function will act on. At the beginning of the function, variables are created for the initial position, new position, and change in position of the encoder for the motor called. The first two variables are initially set to zero as the motor encoder value has not yet been read, whereas the variable for change in position is set to a relatively large number, as the function will exit if this variable reaches zero. A loop is created that repeats while the change_in_pos variable is greater than zero, or a button on the EV3 brick is pressed (although the scenario of a button being pressed is only expected to occur if the robot needs to be shut down before this loop is exited). A timer is reset at the beginning of this loop and the init_pos variable is set to equal the current encoder value of the motor. Then, another checker for an emergency button press is inserted, followed by a nested while loop that counts for a fraction of a second while doing nothing. After this short time interval, the encoder value of the motor encoder is stored again in the new_pos variable and the absolute difference of new_pos and init_pos is stored in change_in_pos. If, by this point, the value of change_in_pos is 0, the motor power for the motor is set to 0 and the function exits with no return value, otherwise the process repeats. A flowchart for the code is shown in Figure 16. This function was written by Jack.

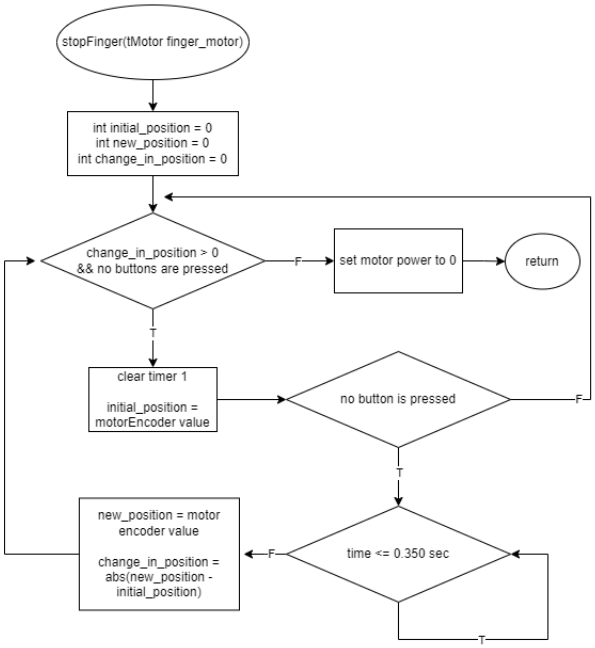

Figure 16: stopFinger flowchart.

5.1.6 Gripping

The grip function has no parameters and loops continuously until either the “up” or “enter” button is pressed on the EV3 brick. At the beginning of this loop, a variable for the initial position is created and set equal to the motor encoder value of motorA and a condition is created called “gripped” and is set to false. Then, a timer is reset before motorA and motorD are set to a positive speed and motorC set to a negative speed. Another loop following this repeats while the “gripped” boolean is false. In this loop, the motor encoder value for motorA is stored in “init_pos” and the timer is cleared again. The function does nothing while the time is less than 0.1 seconds and then compares the value of “init_pos” with the current value of the motor encoder. If the two values are equal, then the “gripped” boolean is set to true and the screen displays the text “GRIPPED”. After this, the current while loop is exited and the motor speed for each finger is set to 0. If the “up” or “enter” button on the EV3 is pressed, the function exits with no return value. A flowchart for the code is shown in Figure 17. This function was written by Evan.

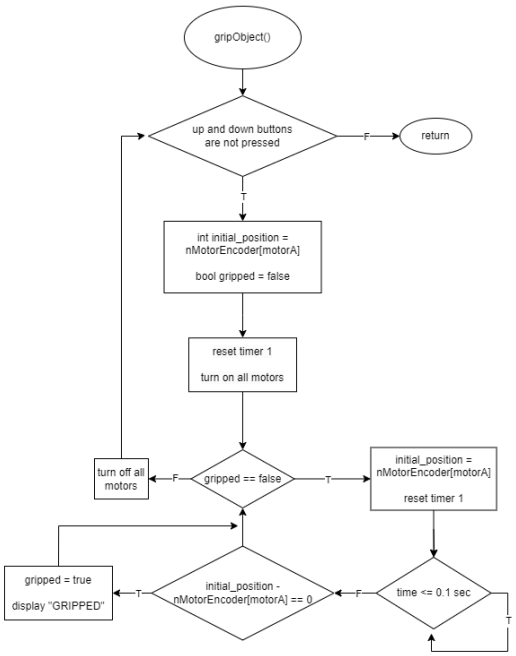

Figure 17: gripObject flowchart.

5.1.7 Checking for Object Distance

The function “checkForObj” utilizes the ultrasonic and touch sensors to return a boolean value for whether an object is “close” and has come into contact with the hand or not. The only parameter for this function is the “closed” boolean from main which is passed by reference in order to allow for the function to change it. Initially, an if statement in the function checks if the ultrasonic sensor reads a value less than or equal to 20, which translates to sensing an object being a distance of 20 cm or less from it. If this condition is true, the openFinger is called to open all three fingers simultaneously and set the closed boolean to false before resetting a timer to 0. Then, a loop runs while the timer is less than 3 seconds or a button on the EV3 brick is pressed (although the scenario of a button being pressed is only expected to occur if the robot needs to be shut down before this loop is exited). This feature is intended to keep the fingers of the robot open for at least 3 seconds after sensing an object within a distance of 20 cm. If at any point during this loop the touch sensor is pressed while the fingers are open or a button on the EV3 brick is pressed, the function will return true. Otherwise, the function will progress forward into another loop. This next loop repeats while the ultrasonic sensor reads an object distance of 20 cm or less and includes several conditional statements that will return either true or false. Firstly, if the touch sensor is pressed and the fingers are not closed, then the function returns

true. Else, if a button on the EV3 brick is pressed and it is the enter button, the function also returns true, otherwise any other button press will return false. After this loop ends, if the fingers are open, then they are all closed simultaneously with the closeFinger function and the closed boolean is set to true. Then, the function exits, returning false. A flowchart for the code is shown in Figure 18. This function was written by Evan and Jack.

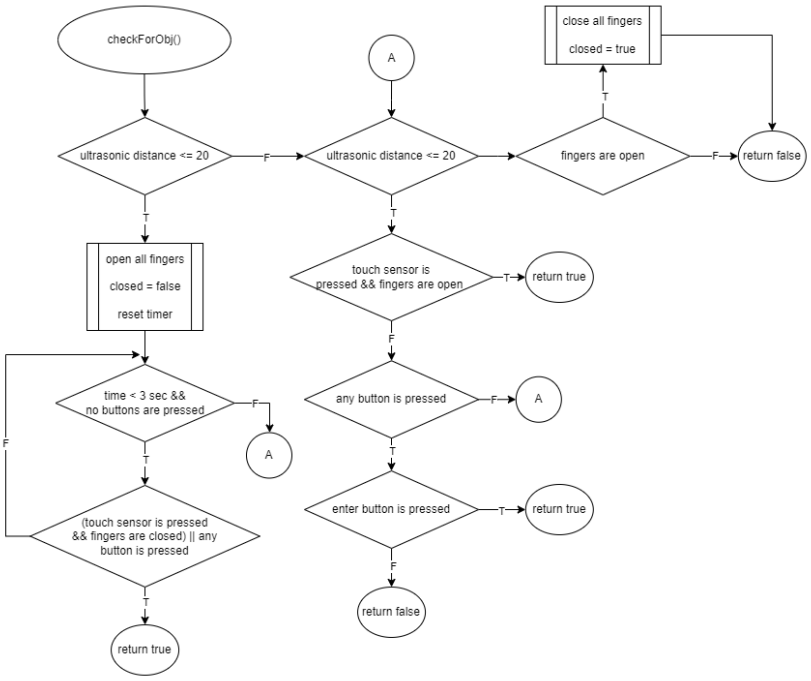

Figure 18: checkForObj flowchart.

5.1.8 Checking for Movement of the Robot

The function “isMoving” utilizes the accelerometer sensor to read the movement of the robot as a whole and return a boolean value depending on if that movement is above (true) or below (false) a defined threshold. The accelerometer sensor is passed as an object in the function’s parameters and is the only parameter for the function. The function first reads the data for the accelerometer and stores the values for the acceleration in each of the x, y, and z axes in the variables x_before, y_before, and z_before respectively. Then, the function waits for a fraction of a second before reading the accelerometer again and storing the new values for the acceleration in each axis in new variables, called x_current, y_current. The function returns true if the difference in accelerometer readings for any axis over the time interval is greater than a value of 30. Otherwise, it returns false. A flowchart for the code is shown in Figure 19. This function was written by Jack and Evan.

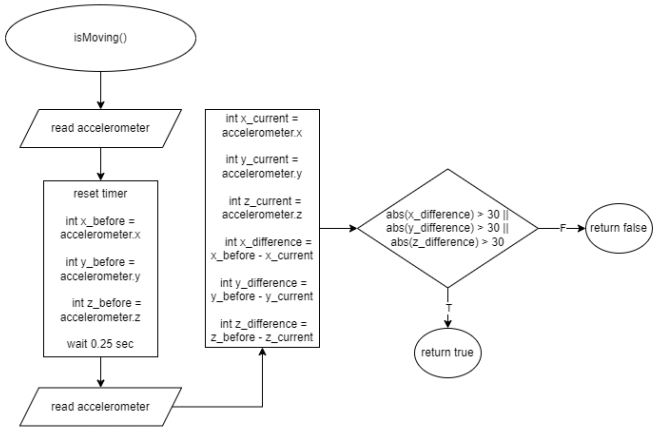

Figure 19: isMoving flowchart.

5.1.9 Manual Control Mode

The “control” function allows for all three fingers to be opened and closed using button presses on the EV3 brick. The “mode” and “closed” variables from the main code are passed by reference in the parameters of this function to allow for them to be changed. When this function is called, the EV3 screen is erased and “MANUAL CONTROL MODE” is displayed. Following this, there is a loop that repeats while the value of “mode” is equal to 3 and contains a series of if/else if conditions. The first condition checks if the “right” button on the EV3 brick is pressed, which, if true, will set the speed of the large fingers to a positive value and the thumb to a negative value, initiating them to close. The function does nothing until the button is released or another button is pressed (that isn’t the “left” button), at which point all three motor speeds will be set to 0. Conversely, if the left button is pressed, the same process occurs, except that the speed of the large fingers is set to a negative value and the thumb a positive value, opening the fingers. If the enter button is pressed at any point, then the “mode” variable is set to 1 and the fingers are first opened fully before closing fully. This is to ensure that they close uniformly and not at the position they were at in manual control to avoid the thumb closing on top of the other fingers. After this, the “closed” variable is set to true. If, instead, neither of these conditions are true and either the “up” or “down” button is pressed, “mode” is set to 0. If “mode” at this point is not 3, the EV3 display is erased, and the function is exited with no return value. A flowchart for the code is shown in Figure 20. This function was written by Yaseen.

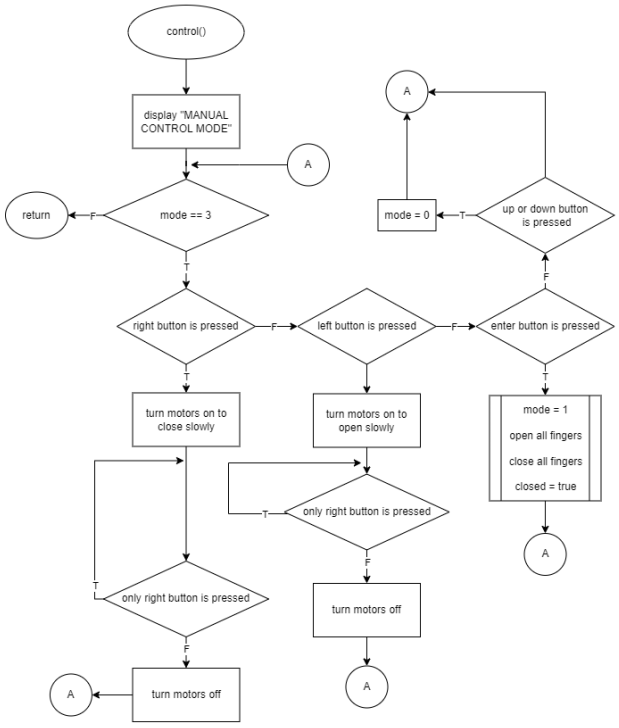

Figure 20: control flowchart.

5.1.10 Wave Gesture

The “wave” function carries out a single action, that is, to simulate a waving hand. Like the “control” function, the “mode” and “closed” variables from the main code are passed by reference in the parameters of this function to allow for them to be changed. When this function is called, the screen on the EV3 brick is erased and displays the text “WAVING”. Then, if “mode” is equal to 4, the function carries out its main task. Firstly, all fingers are opened with the openFinger function before going into a loop that repeats 5 times if the “enter”, “up”, and “down” buttons are not pressed at any point. In this loop, the two large finger motors are turned on until their motor encoders reach a value of 175 or greater and then their motor speed direction is set to 0 and subsequently reversed until their encoder values reach 100 or less, where their motor speed is set to 0, completing a single wave. If this loop is exited by the “up” or “down” buttons being pressed, then “mode” is set to 0. Otherwise, if the loop is completed 5 times, then the fingers are first opened fully before closing fully. This is to ensure that they close uniformly and not at the position they were at following the last wave in order to avoid the thumb closing on top of the other fingers. After this, the “closed” variable is set to true and “mode” is set to 1. Then, the display is erased and the function exits with no return value. A flowchart for the code is shown in Figure 21. This function was written by Yaseen.

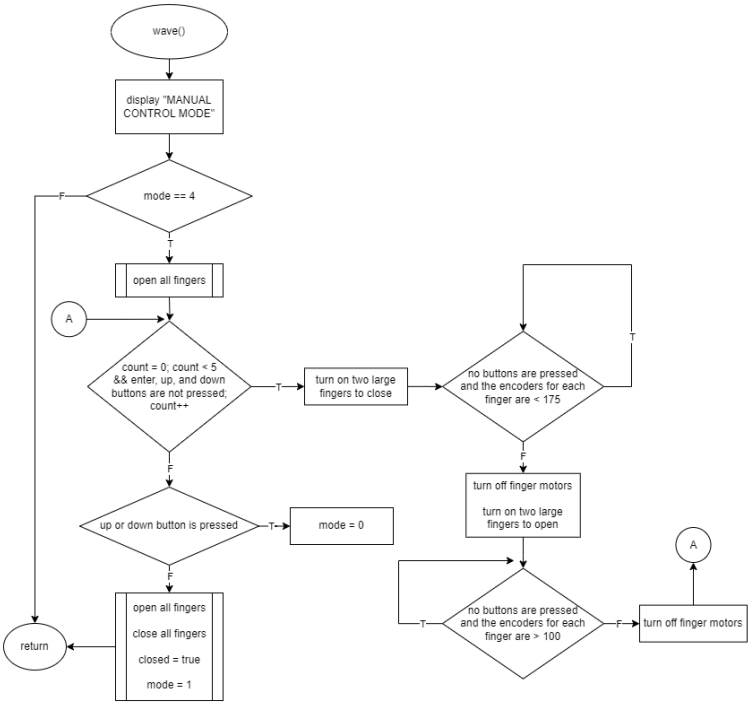

Figure 21: wave flowchart.

5.2 Design Process and Testing Procedure

Early prototypes for the design of the software were created in the form of a flowchart so that the key algorithms and logic structures that the program was going to incorporate could be visualized in a way that would make them easy to replicate in code. Programming began after a few iterations of these flowcharts were made for both the main code as well as each individual function. The team ran software tests on the robot throughout the programming process, especially when major additions or revisions to the code were made.

Initially, the motors had to be tested to make sure they functioned properly and were mechanically sound, which, after some simple tests with turning the motor power on and off for each, the first iteration of the “openFinger” and “closeFinger” functions were written.

Before attaching the ultrasonic sensor to the robot, the team wrote a basic function that opened the fingers when the ultrasonic detected an object at a maximum distance of 10 cm in front of it and closed the fingers if this object moved further away. The team primarily used their hands to test this, although some other shapes and materials were used to experiment with how accurately and quickly the sensor detected objects in front of it. This function (now called “checkForObj”) was then expanded upon to include conditions for the touch sensor being pressed so that if an object was close to the ultrasonic sensor *and* the touch sensor was pressed, the fingers would also close. It was a slight challenge to get this to work smoothly and

consistently, as the ultrasonic sensor would sometimes lose sight of an object in front of it if that object was moving quickly or irregularly. To ensure better results, a timer that would keep the fingers open for 3 seconds after the ultrasonic sensor detected an object directly in front of it was added to the code.

The grip function was created and tested independently, but required fewer iterations. The only testing procedure that was done to validate its functionality was placing an object (such as a water bottle) in the palm of the hand, executing the function, and making sure that the object did not slip out and the fingers retained a firm grasp on it.

Adding a function that used the accelerometer to detect movement of the robot required the most revisions as the sensor was one that the team had not used before. A separate program was made for displaying the readings of the accelerometer to the EV3 screen, which consisted of the acceleration values detected by the sensor in all three-dimensional axes. The team then experimented with holding the accelerometer still and moving it in several directions at different speeds to see how the readings of the sensor would change. This was needed to determine the range of values the accelerometer displayed over a certain time period that would indicate that the sensor is moving a very small amount. After multiple trials, the team decided that if the acceleration read by the sensor changed by a magnitude of over 30 in any of the three axes, then it would not be considered “still” enough if instead it were the entire robot that was moving in such a way. The “isMoving” function was then constructed to return a value of true (meaning that the robot is moving) if the condition for the acceleration changing by more than 30 in any axis was true.

The first prototype for main was created using these functions and allowed the robot to detect, grab, and release an object if the robot was not moving, as these were the primary tasks that the robot was intended to complete. Afterwards, smaller functions were created and integrated into the main code to be tested as a whole. These included the “initializeFinger” and “stopFinger” functions which reduced the lines of code needed for moving the fingers.

The emergency and regular shutdown procedures were added after the team had a fully working version of the main code, but this required a lot of effort as the emergency shutdown was intended to trigger if the “up” button was pressed at any point while the program was running, which was especially time consuming. Any unexpected errors by this point were fixed by using the RobotC compiler to run the code in a stepwise procedure to determine when and where either an additional line of code was needed or an existing one had to be altered in order for the program to function as intended. This was done several times, followed by running the

program normally and testing both shutdown procedures at every step in the robot's operation before the code was considered complete.

The "control" and "wave" functions were added later with the team having extra time to add to the robot's list of features and capabilities. Integrating these functions was simple and didn't require extensive testing. No significant bugs or errors were encountered during the writing and testing of the code.

5.3 Task List

All the robot's functions and capabilities were demonstrated with a list of tasks that was performed once it was turned on. The main tasks intended for the robot to accomplish largely remained the same throughout the design process, mainly because they were set in the criteria of the project. However, some minor tasks were added as the program became more sophisticated with the late addition of the "control" and "wave" functions. The final list of tasks is the following:

1. When the "enter" button is pressed on the EV3 brick, the fingers open fully and then close.
2. The EV3 screen displays "Checking for object to grab…" and waits.
3. Moving an object in range of the ultrasonic sensor causes the fingers to open. Subsequently moving the object out of range causes the fingers to close.
4. While the fingers are open, pressing the touch plate or the "enter" button causes the fingers to close.
5. After the fingers close on the object, the screen displays "GRIPPED" and the fingers remain shut and gripping.
6. If the object is of similar size to the hand and is less than a few pounds, it can be picked up and remain in the grip of the hand.
7. Shaking the robot and pressing the "enter" button displays "Cannot release while hand is moving."
8. Pressing the "enter" button while the robot is relatively still opens the fingers and returns to displaying "Checking for object to grab…" and waiting.
9. Pressing the "left" button displays "Manual Control Mode". Subsequently pressing the "left" button opens the fingers slowly until it is released; pressing the "right" button opens the fingers slowly until it is released. Pressing the "enter" button opens the fingers fully

and then closes them fully and returns to displaying “Checking for object to grab…” and waiting.

10. Pressing the “right” button displays “WAVING” and the two fingers open and close part way five times before opening fully and then closing fully before returning to displaying “Checking for object to grab…” and waiting.
11. Pressing the “up” button on the EV3 brick at any point during tasks 2-10 opens the fingers fully and shuts off the robot.
12. Pressing the “down” button at any point while the fingers aren’t moving or gripping closes the fingers fully and shuts off the robot.

6.0 Verification

To validate the constraints and requirements that were laid out at the beginning of the robot's design process, many objects were used to test the weight and shapes of objects that could be held or picked up. It was found that most objects could be held regardless of shape if they are placed into the hand, but picking up small objects, such as a computer mouse or small orange, was difficult for the bulky fingers. Two water bottles were the main test subjects. The first was a 1 L Nalgene water bottle. The material was a type of plastic that had some stickiness to it and the hand was able to lift this bottle easily and even with it containing lots of water. The other water bottle was a 1 L YETI bottle made of metal. This material was slippery and heavy which made it difficult for the hand to hold onto if the bottle had water in it. To help assist in lifting objects, there were redesigns of the fingers to make them different sizes, but to also add grip to them. Using only the small 2x1 rubber pieces originally, the updated design of using inverted tires improved the grip substantially and helped the hand lift objects such as the water bottles with ease.

A range of distances were tested against the ultrasonic sensor in order for the team to determine the ideal distance from an object to the hand that the fingers should open up in response to. It was found that opening the fingers when an object was 20 centimeters away from the palm was the best distance so that the fingers didn't collide with the object when opening while also not opening too early. However, the ultrasonic itself wasn't always precise, and sometimes couldn't detect an object if it was too small or had an uneven surface. When the fingers were open, the touch sensor was able to recognize the touch of objects no matter where they were pushed against on the touchpad, but challenges came from when the object was too light and when the hand was pushed against it, the object would be pushed instead of pressing in the touch sensor, despite this issue being minimized in the mechanical design of the touch plate system. The manual control provided an alternative solution to some of these challenges, making it more useful for picking up objects in certain scenarios than the autonomous mode. The robot also had the ability to be programmed with different remotes or gestures. The one demonstrated for the purpose of testing was the wave gesture. This was done by pressing one of the buttons on the EV3.

7.0 Project Plan

The project plan was developed to schedule time for testing and fine tuning of the program and construction of the robot. Making substantial progress before each assigned due date for project components was the goal so that the team could receive plenty of feedback from MTE 100 and 121 TAs at each design meeting. Throughout the process, some adjustments were made due to changes to the constraints and criteria. The mechanical side of the project made quick progress, but one step that held back major progress was the creation and attachment of the handle. At the time of the formal presentation for the physical system, the handle had not been printed. Printing the handle at the Rapid Prototyping Center on the Waterloo campus was difficult as printer availability was limited on most days. And once the parts were printed, Rigidware was out of the M3 nuts and bolts needed to connect the printed parts, which kept the handle from being assembled on time. Switching to superglue to connect the parts allowed the pieces to be attached and the entire assembly to be put together. Having the entire assembly streamlined the testing process for code, although many tests were still able to be run before this. At the software meeting, no major code had been written which made it difficult to receive feedback from Praven, the team's assigned TA. However, the flowchart that was made for the entire program allowed the presentation to go smoothly and the core logic behind the code to be discussed. Writing the main task of the robot proved challenging, but through testing each function individually and integrating each one-by-one, it was able to be completed. The team was ultimately delayed in completing some aspects of the project, but because plenty of time for testing and fine tuning was planned in the original schedule, there was still enough time to complete the robot.

Each group member contributed to creating initial designs for the robot. The entire construction of the robot was done by Jack and Evan by first making a prototype chassis, then refining it and making it more structurally sound while adding sensors and motors one-by-one. Yaseen realized there was a need to shape the fingers better so they would have a better grip on objects, so the finger design went through multiple iterations before the final design was made by Jack. The 3D printed parts were designed by Evan using SolidWorks. The program functions were written by those listed in the Software Design section. Yaseen provided some initial ideas for a few of the functions, such as the logic needed for using the accelerometer, that then were written by Jack and Evan. This report was also jointly written by Jack and Evan.

8.0 Conclusion

Vast amounts of research, testing, and design has gone into robotic prosthetics to make them user friendly and compact, while also providing increased functionality. The team's LEGO EV3 prosthetic prototype provided the user with the ability to pick up and grab items without any user input. Keeping hold of a 5-pound item was possible, and the user had the ability to determine when to release an object. The sensors used throughout the project all culminated in making the pickup process autonomous, unless the user wanted manual control over the fingers, though this was also possible. The buttons on the EV3 computer were used to change the mode that the hand was currently in, and the screen provided an accessibility feature of displaying the current mode, so the user knew which task was currently being completed. The ability to pick up medium-sized objects was a success, but difficulty came when the objects were smaller and the fingers were unable to wrap around the object, as just the fingertips made contact.

The central idea for the mechanical design of the robot was to keep it as compact as possible, while providing all the functionality from the sensors. Using a 3D printed handle, the user was able to pick up the robot and carry it around. This was a trade off from a traditional prosthetic because of the weight of the LEGO motors and sensors. The ultrasonic pointed at objects facing the palm of the robotic hand and detected those that were up to 20 centimeters away. The fingers would then open revealing the touch pad. The touch pad allowed for an object to be detected when the palm of the robot was pressed, but with light objects this was sometimes a challenge. The accelerometer measured the movement of the robot, and the software analyzed this data and determined whether an object being held by the robot was allowed to be released. The program used a variable called "mode" throughout the entire program, which was able to determine which tasks were accessible at each instance and which task to complete. The two types of shutdown procedures that were programmed into the code also worked as intended. The emergency shut off can be triggered by button press at any time, and the standard shutdown can be triggered when the robot is not in the middle of completing a task.

The robot accomplished tasks that it was set out to do and the team gained insight into which aspects could be improved upon if it were to be redesigned. Despite difficulties involving deciding how to hold the robot and compacting the design, showing what the sensors and motors could accomplish with limited user input was a significant accomplishment. While the prototype diverges from a typical prosthetic prototype, the proof of concept of what could be

accomplished with more time and sophisticated materials was the main goal of the project and was certainly demonstrated by this robot's design.

9.0 Recommendations

Future iterations could be improved greatly from this first prototype. There are many design changes that could be adjusted as well as streamlining different tasks. One of the main adjustments would be to redesign the fingers. Designing the fingers to have even more traction on them, but also being slimmer would be best. The thumb could also benefit from a redesign to find a method to make it stronger. With it being attached to a plastic axle, there was room for the axle to bend. Reinforcing the thumb would help improve which objects could be gripped. The fingers and the thumb also collided with one another while opening and closing in some instances. Designing them to not overlap as much when closed would allow for more surface area to be in contact with the object being picked up. Designing the robot to be able to pick up smaller objects would also be beneficial as lots of objects were not in the size range of what the robot could pick up with ease. The weight of the robot could also be improved if it was possible to cut down the number of parts used. Most of the weight comes from the motors and sensors, so cutting down weight could be a challenge. The handle that was 3D printed could also benefit from some type of rubber or material that would provide better traction than the PLA. Over time, the user's hand could get sweaty, or the whole assembly could rotate as the handle would slip in the user's hand. Decreasing the weight could also help it possibly be mounted to a user's forearm if the robot was to be rebuilt, removing the need for the user to hold onto it.

For programming, some changes could be to the initialization process or methods to release objects. The initialization process is timely and needed so that the fingers can be zeroed appropriately. However, this initialization process delays the ability for the user to use the robot immediately upon startup, which would be bothersome for the user. Releasing objects also required user input on the EV3; finding a new method could make the autonomous process completely hands-free, which would be ideal. One idea was to have the hand stay still for 5 seconds then release, but that also means if a user was to hold an object by their side, the hand would just drop it after a certain amount of time. Overall, the major adjustments to programming would be to make the robot more autonomous, and less dependent on user input.

References

[1] Johns Hopkins University, "Revolutionizing Prosthetics Research," [Online]. Available: https://www.jhuapl.edu/work/projects/revolutionizing-prosthetics/research. [Accessed 29 November 2023].

[2] Amputee Coalition, "2019 Annual Impact Report," 2019. [Online]. Available: https://www.amputee-coalition.org/about-us/annual-report/2019-annual-report/. [Accessed 29 November 2023].

Appendix

Appendix A: Main Robot Code

```
/*********************************************************************
**********
 * Project:      EV3 Prosthetic Hand (LEGO Mindstorms)
 * File:         Prosthetic-Hand-EV3-code.c
 * Team/Group:   Group 8-5
 * Course:       MTE 121 - Digital Computation
 * Version:      v8.0
 * Date:         December 5, 2023
 *
 * Authors:
 *   -Yaseen Mohamed
 *   -Sara Alrifai
 *   -Jack Doehler
 *   -Evan Janakievski
 *
 * Description:
 *   EV3 prosthetic hand control program written in RobotC. Implements:
 *   - Auto-grab mode using Ultrasonic + Touch sensors
 *   - Grip detection via motor encoder stall behavior
 *   - Movement-based release lockout using HiTechnic Accelerometer
 *   - Manual control mode + waving demo mode
 *
 * Hardware/Ports (edit to match your build):
 *   Motors:
 *     - motorA: Main finger 1
 *     - motorD: Main finger 2
 *     - motorC: Thumb
 *   Sensors:
 *     - S1: EV3 Touch sensor
 *     - S2: EV3 Ultrasonic sensor
 *     - S4: HiTechnic Accelerometer (I2C)
 *
 * Notes:
 *   - Encoder thresholds and distance thresholds are tuned for the current
mechanism.
 *   - Emergency shutdown opens the hand at high speed.
 *********************************************************************
**********/

#pragma config(Sensor, S1, HTAC, sensorI2CCustom)
#include "hitechnic-accelerometer.h"

const int ALL = -1;

void stopFinger(tMotor finger_motor);
```

```
bool checkForObj(bool&closed);
bool isMoving(tHTAC & accelerometer);
void gripObject();
void openFinger(tMotor finger_motor, int motor_speed);
void closeFinger(tMotor finger_motor, int motor_speed);
void initializeFinger(tMotor finger_motor);
void control (int&mode, bool&closed);
void wave (int&mode, bool&closed);

/*********************************** MAIN
************************************/
/**
 * task main
 * Purpose:
 *   High-level state machine for the EV3 prosthetic hand demo.
 *   - Waits for user to start
 *   - Auto-detects object proximity and touch input
 *   - Grips object and prevents release while moving (accelerometer check)
 *   - Provides manual control and "wave" modes
 *
 * Key Inputs:
 *   S1: EV3 Touch sensor
 *   S2: EV3 Ultrasonic sensor (distance in cm)
 *   S4: HiTechnic accelerometer (I2C)
 *
 * Key Outputs:
 *   motorA, motorD: main finger motors
 *   motorC: thumb motor
 */
task main()
{
    SensorType[S1] = sensorEV3_Touch;
    SensorType[S2] = sensorEV3_Ultrasonic;

    tHTAC accelerometer;
  initSensor(&accelerometer, S4);

    displayCenteredBigTextLine(6, "PRESS ENTER");
    displayCenteredBigTextLine(8, "TO START");

    // Wait for a clean "Enter" press
    while (!getButtonPress(buttonEnter))
    {}
    while (getButtonPress(buttonEnter))
    {}

    eraseDisplay();
    displayCenteredBigTextLine(6, "BOOTING UP");

    // Zero motors so encoder positions are repeatable
```

```
    initializeFinger(motorA);
    initializeFinger(motorD);
    initializeFinger(motorC);

    int acc_x = 0, acc_y = 0, acc_z = 0;
    int mode = 1;
    bool closed = false;

    while (mode == 1)
    {
        eraseDisplay();
        displayCenteredBigTextLine(5, "Checking for");
        displayCenteredBigTextLine(7, "object to grab...");

        // Auto mode: check for object proximity + touch trigger
        while (!checkForObj(closed) && !getButtonPress(buttonAny))
        {}

        // Exit conditions
        if (getButtonPress(buttonUp) || getButtonPress(buttonDown))
                mode = 0;

        // Manual control mode
        if (getButtonPress(buttonLeft))
        {
            mode = 3;
            control(mode, closed);
        }

        // Wave mode
        if (getButtonPress(buttonRight))
        {
            mode = 4;
            wave(mode, closed);
        }

        // If we found an object and we're in auto mode, close to pick it up
        if (!closed && mode == 1)
        {
            closeFinger(ALL, 20);

            // Capture tilt/pose after grabbing (stored but not used
elsewhere)
            readSensor(&accelerometer);
            acc_x = accelerometer.x;
            acc_y = accelerometer.y;
            acc_z = accelerometer.z;

            mode = 2;
        }
```

```
    while (mode == 2)
    {
        gripObject();

        if (getButtonPress(buttonUp))
            mode = 0;

        // Attempt to release: block if accelerometer indicates movement
        if (getButtonPress(buttonEnter))
        {
            while (getButtonPress(buttonEnter))
            {}

            if (!isMoving(accelerometer))
            {
                openFinger(ALL, 20);
                mode = 1;
            }
            else
            {
                eraseDisplay();
                displayCenteredBigTextLine(4, "Cannot release");
                displayCenteredBigTextLine(6, "while hand");
                displayCenteredBigTextLine(8, "is moving.");
                wait1Msec(750);
            }
        }
    }
}

eraseDisplay();

// Regular shutdown
if (getButtonPress(buttonDown))
{
    while (getButtonPress(buttonDown))
    {}

    displayCenteredBigTextLine(5, "SHUTTING DOWN...");
    closeFinger(ALL, 10);
}
// Emergency shutdown
else
{
    displayCenteredBigTextLine(5, "EMERGENCY SHUTDOWN INITIALIZED");
    openFinger(ALL, 50);
}

wait1Msec(1000);
```

```
}

/*********************************************************************************
**********
************************************ FUNCTIONS
************************************
*********************************************************************************
**********/

/**
 * initializeFinger
 * What:
 *   Moves the specified motor briefly to create a consistent starting pose,
 *   then stops and zeros its encoder.
 *
 * How:
 *   - Applies a short motor pulse (direction depends on motor)
 *   - Calls stopFinger() to halt motion
 *   - Sets nMotorEncoder[...] = 0
 *
 * Why:
 *   Ensures later open/close thresholds use repeatable encoder references.
 *
 * Params:
 *   finger_motor: EV3 motor port (motorA/motorC/motorD)
 */
void initializeFinger(tMotor finger_motor)
{
    if (finger_motor == motorA || finger_motor == motorD)
        motor[finger_motor] = -15;
    else
        motor[finger_motor] = 8;

    wait1Msec(175);

    stopFinger(finger_motor);
    nMotorEncoder[finger_motor] = 0;
}

/**
 * closeFinger
 * What:
 *   Closes one finger motor or all motors (fingers + thumb).
 *
 * How:
 *   - For motorA/motorD: positive closes
 *   - For motorC (thumb): negative closes (opposite direction)
 *   - For ALL: closes A and D, then closes C with a scaled speed
 *
 * Why:
```

```
 *   Performs a controlled closing action and then stops motors to avoid
stalls.
 *
 * Params:
 *   finger_motor: motorA, motorC, motorD, or ALL (-1)
 *   motor_speed: base speed (thumb may be scaled when ALL)
 */
void closeFinger (tMotor finger_motor, int motor_speed)
{
    if (finger_motor == motorA || finger_motor == motorD)
        motor[finger_motor] = motor_speed;

    else if (finger_motor == motorC)
        motor[finger_motor] = -motor_speed;

    else if (finger_motor == ALL)
    {
        motor[motorA] = motor_speed;
        motor[motorD] = motor_speed;
        wait1Msec(250);
        motor[motorC] = -motor_speed*1.5;
    }

    wait1Msec(250);

    if (finger_motor != ALL)
        stopFinger(finger_motor);
    else
    {
        stopFinger(motorC);
        stopFinger(motorA);
        stopFinger(motorD);
    }
}

/**
 * openFinger
 * What:
 *   Opens one motor or all motors until a target encoder position is
reached.
 *
 * How:
 *   - Drives motor(s) in the opening direction
 *   - Uses encoder thresholds to stop at a repeatable "open" position
 *   - Aborts if any button is pressed
 *
 * Why:
 *   Returns fingers/thumb to a known open state without overdriving the
mechanism.
 *
```

```
 * Params:
 *   finger_motor: motorA, motorC, motorD, or ALL (-1)
 *   motor_speed: speed used for opening (thumb may be scaled when ALL)
 */
void openFinger (tMotor finger_motor, int motor_speed)
{
    // NOTE: "(finger_motor == (motorA || motorD))" is preserved as-is to
avoid behavior changes.
    if (finger_motor == (motorA || motorD))
    {
        motor[finger_motor] = -motor_speed;
        while (nMotorEncoder[finger_motor] > 10 &&
!getButtonPress(buttonAny))
        {}
        motor[finger_motor] = 0;
    }
    else if (finger_motor == motorC)
    {
        motor[finger_motor] = motor_speed;
        while (nMotorEncoder[finger_motor] > 40 &&
!getButtonPress(buttonAny))
        {}
        motor[finger_motor] = 0;
    }
    else
    {
        motor[motorA] = -motor_speed;
        motor[motorD] = -motor_speed;
        wait1Msec(350);
        motor[motorC] = motor_speed*2;

        while(nMotorEncoder[motorA] > 10 && !getButtonPress(buttonAny))
        {}
        motor[motorA] = 0;

        while(nMotorEncoder[motorD] > 10 && !getButtonPress(buttonAny))
        {}
        motor[motorD] = 0;

        while(nMotorEncoder[motorC] > 40 && !getButtonPress(buttonAny))
        {}
        motor[motorC] = 0;
    }
}

/**
 * stopFinger
 * What:
 *   Stops a motor once it is no longer changing encoder position (i.e., has
settled).
```

```
 *
 * How:
 *   - Samples encoder position, waits a short time window, samples again
 *   - Repeats until no change is detected (or a button is pressed)
 *   - Sets motor power to 0
 *
 * Why:
 *   Prevents overshoot and ensures the motor actually finishes moving before
actions continue.
 *
 * Params:
 *   finger_motor: motor port to stop
 */
void stopFinger(tMotor finger_motor)
{
    int init_pos = 0;
    int new_pos = 0;
    int change_in_pos = 1000;

    while (change_in_pos > 0 && !getButtonPress(buttonAny))
    {
        clearTimer(T1);
        init_pos = nMotorEncoder[finger_motor];

        if (!getButtonPress(buttonAny))
        {
            while (time1[T1] <= 350)
            {}
            new_pos = nMotorEncoder[finger_motor];
            change_in_pos = abs(new_pos - init_pos);
        }
    }

    motor[finger_motor] = 0;
}

/**
 * gripObject
 * What:
 *   Actively closes the hand until it detects a "grip" condition.
 *
 * How:
 *   - Drives A/D (fingers) and C (thumb) at higher speeds
 *   - Monitors encoder change; if the finger encoder stops changing, assumes
contact/stall = gripped
 *
 * Why:
 *   Produces a simple “close until contact” behavior without force sensors.
 *
 * Notes:
```

```
 *   This approach assumes "no encoder movement" means an object is resisting
motion.
 */
void gripObject()
{
    while (!getButtonPress(buttonUp) && !getButtonPress(buttonEnter))
    {
        int init_pos = 0;
        bool gripped = false;

        init_pos = nMotorEncoder[motorA];
        clearTimer(T1);

        motor[motorA] = motor[motorD] = 50;
        motor[motorC] = -30;

        while (!gripped)
        {
            init_pos = nMotorEncoder[motorA];
            clearTimer(T1);

            while (time1[T1] <= 100)
            {}

            if (init_pos - nMotorEncoder[motorA] == 0)
            {
                gripped = true;
                eraseDisplay();
                displayBigTextLine(6, "GRIPPED");
            }
        }

        motor[motorA] = motor[motorD] = motor[motorC] = 0;
    }
}

/**
 * checkForObj
 * What:
 *   Uses ultrasonic proximity + touch input to decide when the system should
grab.
 *
 * How:
 *   - If an object is within 20 cm, open the hand and wait up to ~3 seconds
for a touch trigger
 *   - If object remains within 20 cm, continue waiting for touch or button
exit
 *   - If object leaves range and hand is open, close the hand back up (idle
posture)
 *
```

```
 * Why:
 *   Enables an “auto grab” flow: open when object is near, grab when touch
is pressed.
 *
 * Params:
 *   closed (in/out): tracks whether the hand is currently closed
 *
 * Returns:
 *   true  -> trigger condition met (ready to grab or user confirms)
 *   false -> no trigger / user exit
 */
bool checkForObj(bool&closed)
{
    if (SensorValue[S2] <= 20)
    {
        openFinger(ALL, 20);
        closed = false;

        clearTimer(T1);

        while (time1[T1] < 3000 && !getButtonPress(buttonAny))
        {
            if ((SensorValue[S1] && !closed) || getButtonPress(buttonAny))
                return true;
        }
    }

    while (SensorValue[S2] <= 20)
    {
        if (SensorValue[S1] && closed == false)
            return true;

        else if (getButtonPress(buttonAny))
        {
            if (getButtonPress(buttonEnter))
                return true;
            else
                return false;
        }
    }

    if (!closed)
    {
        closeFinger(ALL, 20);
        closed = true;
    }

    return false;
}
```

```
/**
 * isMoving
 * What:
 *   Detects whether the hand is moving by comparing accelerometer samples.
 *
 * How:
 *   - Reads accelerometer (x/y/z)
 *   - Waits 250 ms and reads again
 *   - If any axis changes beyond a threshold, considers the hand "moving"
 *
 * Why:
 *   Prevents releasing the grip when the hand is in motion.
 *
 * Params:
 *   accelerometer: reference to the initialized accelerometer sensor object
 *
 * Returns:
 *   true  -> movement detected
 *   false -> hand appears stationary
 */
bool isMoving(tHTAC & accelerometer)
{
    readSensor(&accelerometer);
    clearTimer(T1);

    int x_before = accelerometer.x;
    int y_before = accelerometer.y;
    int z_before = accelerometer.z;

    wait1Msec(250);

    readSensor(&accelerometer);
    int x_current = accelerometer.x;
    int y_current = accelerometer.y;
    int z_current = accelerometer.z;

    int x_difference = x_before - x_current;
    int y_difference = y_before - y_current;
    int z_difference = z_before - z_current;

    if (abs(x_difference) > 30 || abs(y_difference) > 30 || abs(z_difference)
> 30)
        return true;
    else
        return false;
}

/**
 * control
 * What:
```

```
 *   Manual control mode for opening/closing the hand while buttons are held.
 *
 * How:
 *   - Right button: close (A/D positive, C negative)
 *   - Left button: open (A/D negative, C positive)
 *   - Enter: return to auto mode and set a known closed posture
 *   - Up/Down: exit program
 *
 * Why:
 *   Provides direct manual override for testing and demos.
 *
 * Params:
 *   mode (in/out): state machine mode; exits when changed
 *   closed (in/out): updated to keep consistency with auto logic
 */
void control (int&mode, bool&closed)
{
    eraseDisplay();
    displayCenteredBigTextLine(6, "Manual Control Mode");

    while (mode == 3)
    {
        if (getButtonPress(buttonRight))
        {
            motor[motorA] = motor[motorD]=20;
            motor[motorC] = -10;
            while (getButtonPress(buttonRight) &&
!getButtonPress(buttonEnter) && !getButtonPress(buttonDown) &&
!getButtonPress(buttonUp))
            {}
            motor[motorA] = motor[motorD] = motor[motorC] = 0;
        }
        else if (getButtonPress(buttonLeft))
        {
            motor[motorA] = motor[motorD] = -20;
            motor[motorC] = 10;
            while (getButtonPress(buttonLeft) && !getButtonPress(buttonEnter)
&& !getButtonPress(buttonDown) && !getButtonPress(buttonUp))
            {}
            motor[motorA] = motor[motorD] = motor[motorC]=0;
        }
        else if (getButtonPress(buttonEnter))
        {
            mode = 1;
            openFinger(ALL, 20);
            closeFinger(ALL, 20);
            closed = true;
        }
        else if (getButtonPress(buttonUp) || getButtonPress(buttonDown))
            mode = 0;
```

```
    }
    eraseDisplay();
}

/**
 * wave
 * What:
 *   Runs a simple “wave” animation using finger motors A and D.
 *
 * How:
 *   - Opens hand slightly
 *   - Repeats a back-and-forth motion using encoder thresholds
 *   - Returns to auto mode with a closed posture
 *
 * Why:
 *   Provides a demo-friendly gesture mode.
 *
 * Params:
 *   mode (in/out): wave mode runs while mode==4; updates to exit/return to
auto
 *   closed (in/out): updated when returning to auto mode
 */
void wave (int&mode, bool&closed)
{
    eraseDisplay();
    displayCenteredBigTextLine(6, "WAVING");

    if (mode != 0 && mode == 4)
    {
        openFinger(ALL, 15);
        wait1Msec(100);

        for (int count=0; count<5 && !getButtonPress(buttonEnter) &&
!getButtonPress(buttonDown) && !getButtonPress(buttonUp); count++)
        {
            motor[motorA] = motor[motorD] = 25;
            while (nMotorEncoder(motorA) < 175 && nMotorEncoder(motorD) < 175
&& !getButtonPress(buttonEnter) && !getButtonPress(buttonDown) &&
!getButtonPress(buttonUp))
            {}
            motor[motorA] = motor[motorD] = 0;
            wait1Msec(100);

            motor[motorA] = motor[motorD] = -25;
            while (nMotorEncoder(motorA) > 100 && nMotorEncoder(motorD) > 100
&& !getButtonPress(buttonEnter) && !getButtonPress(buttonDown) &&
!getButtonPress(buttonUp))
            {}
            motor[motorA] = motor[motorD] = 0;
            wait1Msec(100);
```

```
        }

        if (getButtonPress(buttonUp) || getButtonPress(buttonDown))
            mode = 0;
        else
        {
            openFinger(ALL, 20);
            closeFinger(ALL, 20);
            closed = true;
            mode = 1;
        }
    }
    eraseDisplay();
}
```