**ET 488: Robotics and Automation**

Explain the project, the components, the procedure, the program, the hurdles, etc.

To design our robotic arm, we first started out with the parts. We used an Arduino for the microcontroller and two hs-785hb servos for the main arms for rotation. Then for pickup the L12 is used for the linear actuator and another smaller servo was used to open and close the gripper. Starting, we assembled the arm and designed a wooden board. Then to begin the programming, we first found the zero points for the two main servos. Once set to their zero point, the servos can be mapped to degree values so that they can be controlled within a 360-degree range instead of the 600 to 2400 microseconds. With full control of the servos now, we then were able to apply inverse kinematics to the arm to work off of giving it coordinate values. Now that the arm is able to move to coordinate points, the restrictions were then put in place so that the arm would not collapse on itself or go beyond the range of our 2-quadrant board. Up to this point to send the arm x and y values we had to reupload to the Arduino. To fix this we added user input through the serial monitor so that the user can continuously enter the next coordinates for the arm without having to reupload to the board. As for the linear actuator, we treated it much like a servo motor in that we would send it a certain microsecond value to extend and retract the actuator. The gripper was also similar in that we fed it a degree value being either 0 or 180 to open and close the claw. Then the commands were placed in function for ease of use such as inverse kinematics, zero position, open and close claw, and extend and retract for the linear actuator. These functions were then called in the void loop section of the code to continuously run. There were some challenges for the project. One instance was that we had trouble with the zero points resetting themselves if values were not inputted correctly to the servos. To overcome this, we set the restrictions for the arms to further prevent the issue. Another problem was on the hardware side. The gears on the servos did not line up and fit exactly so that for each of the servos there was a great amount of play with movement. While the arm would go to the correct value the hardware would slightly shift over the position by a quarter inch in the y direction. As well the arm itself had a slight lean to it which we were able to fix with a better mounting setup. Overall the arm was a success in that once the code is uploaded the arm will go the zero position, wait for the pickup coordinates and the drop-off coordinates, and constantly ask for the next set of values through the serial monitor.

Please fill out this table which will help me assess your group members’ contributions/grades in this project

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Explain the contribution | % of Contribution /100% | Additional Notes |
| Your name:  Jordan Melancon | Coding the servos and mapping the zero points. Coding the inverse kinematic function. Building the board for the robotic arm as well as the pickup objects. | 25% |  |
| Partner Name:  Cleland Montecillo | Coding and mapping the servos. Coding the linear actuator and gripper functions. Coding the restrictions for the arm. Testing and debugging the arm. | 25% |  |
| Partner Name:  Jeff Parrozzo | Coding the linear actuator and the gripper as well as assisting in programming the forward and inverse kinematics. | 25% |  |
| Partner Name:  Hayley Gilson | Setup the inverse kinematic equations for the arm | 25% |  |