

MECHTRON 4TB6 - Mechatronics Capstone Projects

**Development Process & Implementation**

Dhruv Aggarwal

Joseph Moolasseril

Gurkarmjit Kooner

Auda Rab

Ahmed Belal

# **Overall Process workflow**

**Process Steps:**

1. Calculate the lengths of joints, angular limits of the joints, motor positions, and motor angles that will provide the correct end-effector (platform) orientations using inverse kinematics equations in Excel.
2. Use the calculated values to design the motor housing and joints for the Stewart Platform in Autodesk Inventor.
3. 3D Print the designed components which include the servo motor housing (the base of the system), the servo joints, and the main platform.
4. Assemble the Stewart platform.
5. Generate software that will take a desired end-effector position and orientation as an input and use inverse kinematics to calculate joint angles and move the servo motors accordingly.
6. Perform unit testing for each servo motor and its joints to ensure that the correct orientation is achieved.
7. Troubleshoot and fix any bugs that are raised during testing.
8. Use X-Sim to extract vehicle orientation information from the racing video game.
9. Feed the extracted X-Sim values to the Arduino microcontroller to set the servo motor angles.
10. Test the system to ensure all parts integrate well.

**Step Detail**

|  |  |  |
| --- | --- | --- |
| **Step** | **Input** | **Output** |
| Calculate the lengths, limits, and positions of joints and motors | Desired limits for the end-effector’s workspace (maximum translational/rotational values) | Measurements for the positions/orientations of the servo motors, as well as elbow joint lengths and platform size/position |
| Motor housing and joint design in Autodesk Inventor | Measurements calculated in the previous step. | 3D CAD models of each major component of the Stewart Platform |
| 3D printing of CAD models | The CAD models created in the previous step. | 3D printed pieces of the Stewart Platform’s major components |
| Assembling the Stewart Platform | 3D printed parts of the Stewart Platform and purchased nuts/bolts/rods | A fully assembled Stewart Platform |
| Software for manipulating the Stewart Platform using inverse kinematics | Measurements calculated in step 1 | Arduino code to manipulate the six servo motor angles |
| Unit testing and troubleshooting | Fully assembled Stewart Platform and the Arduino code to manipulate its position and orientation, as well as several desired end-effector positions for testing | Fully functioning Stewart Platform |
| Use X-Sim to extract information from the video game | A racing video game | Values for the translational and rotational orientations of the car within the racing game |
| Using the Arduino microcontroller to set the servo motor joints to the orientation received from X-Sim | Values for the orientation of the car from the racing game | Voltages to send to the Stewart Platform servo motors |

* What are your acceptance criteria for the each of the outputs of a step?

The acceptance criteria for each step is as follows:

|  |  |
| --- | --- |
| **Step** | **Acceptance Criteria** |
| Calculate the lengths, limits, and positions of joints and motors | Final accepted measurements create a portable and aesthetically pleasing design for a Stewart Platform |
| Motor housing and joint design in Autodesk Inventor | The CAD models match the Excel calculations from step 1 |
| 3D printing of CAD models | The 3D printed components are within tolerance of the desired measurements and CAD models |
| Assembling the Stewart Platform | The assembled system matches the design parameters and measurements gathered in step 1 |
| Software for manipulating the Stewart Platform using inverse kinematics | The servo motors move to the appropriate angle as determined by inverse kinematics calculations |
| Unit testing and troubleshooting | The unit testing will result in a pass or fail depending on whether the actual results matched the theoretical results. |
| Use X-Sim to extract information from the video game | These steps will be completed if the two systems communicate successfully and translate on-screen values into end-effector orientations. |
| Using the Arduino microcontroller to set the servo motor joints to the orientation received from X-Sim |

* Provide some, but limited, detail on how each step should be done?

|  |  |
| --- | --- |
| **Step** | **Details** |
| Calculate the lengths, limits, and positions of joints and motors | Cells in an Excel spreadsheet will contain input values for the Stewart Platform’s main lengths/angles. These cells will be referenced by other cells which contain inverse kinematics equations which will help determine if the currently entered lengths and angles will provide the desired range of motion for the end-effector. |
| Motor housing and joint design in Autodesk Inventor | The CAD models will be created using Autodesk Inventor based on the values chosen in the Excel equations in the previous step |
| 3D printing of CAD models | The CAD models will be exported and printed using a 3D printer |
| Assembling the Stewart Platform | The Stewart platform will become a complete system when the 3D printed parts along with screws, nuts, ball joints and threaded rods are put together |
| Software for manipulating the Stewart Platform using inverse kinematics | The software for manipulating the Stewart platform will begin by connecting the motors to the appropriate Arduino Pulse-Width Modulation (PWM) pins. The software will take a desired end-effector orientation and use inverse kinematics equations to determine the angles for each servo motor, and will then set each motor to the calculated angle through the PWM pins. |
| Unit testing and troubleshooting | The testing will involve choosing a variety of end-effector positions and determining if the Stewart platform can reliably set itself to those positions. |
| Use X-Sim to extract information from the video game | X-Sim extracts vehicle orientation values from a racing video game which will be sent to the Arduino microcontroller to allow the Stewart platform to mimic the vehicle’s movements. |
| Using the Arduino microcontroller to set the servo motor joints to the orientation received from X-Sim | The Arduino will receive values from X-Sim which will represent the desired orientation. The Arduino’s software will use inverse kinematics to convert this orientation into servo motor angles, and will then move the servo motors to match the orientation of the vehicle in the video game. |

* What tools are you going to use? What versions?

The tools we are going to be using for our capstone includes:

* Arduino (Latest)
* Simulation software (X-Sim Version 3.0.2.9)
* Excel 2016
* Autodesk Inventor 2018
* What information to put in version control?

The information to put in the Git Hub would include the file name and version number. It would follow a format similar to Filename\_VXX.

* Any standards you should follow? (e.g. coding standards)

There are no specific coding standards that will be followed. However, the only rule is to keep terminology, syntax, comments consistent throughout the code and deliverables.

* Who should perform each step?

Everyone in the group is responsible for following the version control scheme for any deliverable.

* How are you using version control?

We have a created Git Repository to keep control of our software

* How are you going to deal with changes to development artefacts?

In order to deal with the changes during the development phase we will document everything that is completed during each step to minimize the risk for losing progress. Also, with the correct documentation changes will be easier to deal with

* What bug tracking/change request tool?

We use GitHub to keep track of changes, different software versions, and bugs.

* How do you document change requests/bugs?

We use GitHub’s documentation functionality to document all the changes made to our project and issues that arise.

* How do you classify changes?
  + Changes are classified as software, hardware, or design changes.
  + Software changes include changes to our scripts or programs.
  + Hardware changes include changes to the hardware used in the project such as motors, microcontrollers, or fasteners.
  + Design changers include changes to the appearance of the device whether they be aesthetic or functional.
* How do you disposition them (decide what to do, verify that they have been

Completed, etc)

* + When ideas for changes are presented, the group meets to discuss the pros and cons of each idea.
  + When an idea is selected, the task is assigned to one or two group member to implement.
  + To verify that the changes have been completed, the change is tested and presented to all group member. We then decide whether the change is satisfactory and reached its intended result.

ARCHIVE

1. Design the Stewart platform using Inverse Kinematics and Autodesk Inventor
   * All the calculations for the input variables such as pitch,yaw, roll, X,Y and Z need be conducted
2. A racing video game will provide input to the simulation software
   * The simulation software will then get the orientation of the platform
3. 3D print the components which involved the base, servo arms and the platform.
   * All the design work need to be completed using Autodesk Inventor
4. Inputs are the desired angles that the joint should reach and its dimensions.
   * The software will run an algorithm to get the angles for all the motors based on the input
5. Assemble all the Stewart Platform parts together.
   * In order to assemble the Stewart platform, the 3D printed components will come together to form the system
6. Generate software that will take desired position as an input and move the motors accordingly
   * In order to generate software, the key component that will be an input will be all the calculations for the input variables X,Y, Z, roll, pitch and yaw.
7. Unit Testing for each servo and joints to get the correct orientation
   * The testing will require the test cases along with the assembled 3D printed components

* Are there any special instructions on settings or how to use the tools?