Ruben Konings (student);Thom Peters (student)

Project S3 – Rev up your robot coding skills

Plan of approach

# Version revision

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Autor | Note |
| 0.1 | 28-8-2023 | Ruben and Thom | Lay-out and info |
| 0.2 | 6-9-2023 | Ruben and Thom | Full version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of contents

[1. Version revision 1](#_Toc144891532)

[3. Problem 3](#_Toc144891533)

[4. To do plan 4](#_Toc144891534)

[5. Planning 7](#_Toc144891535)

[6. Requirements 8](#_Toc144891536)

# Problem

Improving a Delta Gripper Robot: Fixing Bugs and Creating a User-Friendly Interface

In this project, we have received a Delta Gripper Robot without a working operating system. Our goal is to identify and resolve any issues in the existing system, as well as develop a user-friendly interface for controlling the robot. This involves debugging the legacy system and designing a simple interface to make it easier to operate the Delta Gripper Robot. The project aims to enhance the robot's functionality and user experience.

# To do plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What | How | When | Deadline | Who |
| Research | In-depth research on the functioning and applications of the delta picker based on prior student projects and industry case studies. | Week 35: Gather initial research materials.  Week 36: Analyse gathered data and identify key insights. | Week 37: Complete the research phase. | Ruben & Thom |
| * Documentation | Gather academic papers, technical documentation, and user manuals related to delta grippers. |
| * Contact project teams | Contact previous project teams and experts in the field for insights. |
| * Experiments | Conduct experiments if necessary to understand the delta gripper's capabilities.  And see how other robot’s work. |
| * Qube operating system | Meet with the one of the developers of both the cube and the delta robot to get more insite and a understanding for both of them. |  |  |  |
| Jogging system | Develop a robust control system that allows the delta gripper to move in any direction with an accuracy of 5 mm if the hardware is capable. | Week 37: Initial code development and testing.  Week 38: Iterative improvements and fine-tuning. | Week 39: Achieve seamless multidirectional movement. | Ruben & Thom |
| * Test code | Write and test code for delta gripper movement, focusing on directional accuracy and control responsiveness. |
| * Position | Implement feedback mechanisms to ensure the tool position accuracy. |
| Coordinate system | Create a comprehensive coordinate system that accurately determines the robot's position and destination within its workspace. And move to these position smoothly. | Week 40: Begin coordinate system design.  Week 41-42: Algorithm development and testing. | Week 43: Full implementation of the coordinate system. |  |
| * Framework | Design a coordinate framework that integrates with the robot's hardware and software. |
| * Algorithms | Develop algorithms for real-time position tracking and mapping. |
| GUI (Graphical User Interface) | Develop an intuitive and user-friendly GUI to control the robot's actions and monitor its status. | Week 44: GUI design and initial coding.  Week 45-46: Integration and testing. | Week 47: Fully functional GUI. |  |
| * Design | Design a user interface with clear visuals, controls, and feedback elements. |
| * Integrate | Integrate the GUI with the robot's control software. |
| Programs | Create a suite of programs that enable the robot to execute complex tasks, including moving to multiple positions without interruption. | Week 48: Program architecture design.  Week 49-50: Coding and initial testing. | Week 51: Reliable program suite. |  |
| * Routines | Develop software routines that coordinate movements and task sequences. |
| * Error handling | Implement error handling and recovery mechanisms. |
| Tool Integration | Develop a modular system that allows the robot to adapt to various tools, considering attributes like length and input requirements. | Week 52: Tool interface design.  Week 1-2 (Next Year): Software integration and tool testing. | Week 3 (Next Year): Complete tool integration. |  |
| * Tool interfaces | Design interchangeable tool interfaces that can be quickly attached to the gripper. |
| * Calibrate | Implement software features to recognize and calibrate with different tools. |
| Pendant | if we have time left to make a tablet into a “pendent” by remote controlling the delta robots | To be determined |  |  |

# Planning

The planning can be found in the folder: \Project S3\Documentation\Planning S3.

# Requirements

|  |  |
| --- | --- |
| Requirement | MoSCoW |
| 1. Introduction: |  |
| Develop an operating system (OS) tailored specifically for the control and operation of a delta gripper, a specialized robotic end-effector used for precise manipulation and handling tasks in various industries. |  |
|  |  |
| 2. System Architecture: |  |
| Design a modular and scalable architecture that allows for efficient communication between the gripper hardware, sensors, actuators, and external interfaces. The architecture should facilitate real-time control and ensure safety during operation. |  |
|  |  |
| 3. Real-time Control: |  |
| The OS should provide real-time control capabilities to ensure accurate and responsive control of the delta gripper's movements. It should support precise motion planning, trajectory generation, and feedback control loops to maintain desired positions and forces. |  |
|  |  |
| 4. Motion Planning and Kinematics: |  |
| Implement algorithms for forward and inverse kinematics to translate desired end-effector positions and orientations into appropriate joint angles. The OS should support various trajectory planning methods, including point-to-point, spline-based, and Cartesian path planning. |  |
|  |  |
| 5. Sensor Integration: |  |
| Integrate various sensors such as force/torque sensors, vision systems, and proximity sensors to enable the gripper to sense its environment and respond intelligently. Sensor data should be processed and utilized for adaptive gripping and collision avoidance. |  |
|  |  |
| 6. Actuator Control: |  |
| Develop control interfaces to regulate the gripper's actuators, allowing precise and coordinated motion of the gripper's arms. Implement control strategies for actuator synchronization and optimization to prevent jerky or imprecise movements. |  |
|  |  |
| 7. Communication Protocols: |  |
| Support communication protocols for seamless integration with external devices and systems. Implement protocols such as Ethernet, USB, and wireless communication to facilitate remote control, monitoring, and data exchange. |  |
|  |  |
| 8. User Interface: |  |
| Design a user-friendly interface that allows operators to interact with the gripper system intuitively. The interface should provide control over motion, gripping force, and operating modes, as well as visual feedback from sensors and cameras. |  |
|  |  |
| 9. Safety Features: |  |
| Implement safety mechanisms to prevent collisions, limit joint angles within safe ranges, and respond to emergency stop signals. Incorporate software interlocks to prevent unintended actions and ensure safe operation in dynamic environments. |  |
|  |  |
| 10. Error Handling and Diagnostics: |  |
| Develop a comprehensive error handling and diagnostic system that identifies issues, logs errors, and provides meaningful feedback to users. This system should aid in troubleshooting and maintenance of the delta gripper. |  |
|  |  |
| 11. Open Architecture: |  |
| Create an open architecture that supports third-party extensions and plugins for customizing the OS to specific applications and industries. Encourage a collaborative ecosystem for continuous improvement and innovation. |  |
|  |  |
| 12. Documentation: |  |
| Provide thorough documentation including installation instructions, API references, tutorials, and examples to guide users in setting up, configuring, and programming the delta gripper OS effectively. |  |
|  |  |
| 13. Testing and Validation: |  |
| Conduct rigorous testing to validate the performance, stability, and safety of the OS. Include simulation environments for testing various scenarios before deployment on the physical gripper. |  |
|  |  |
| 14. Performance Optimization: |  |
| Optimize the OS for efficient resource utilization, minimal latency, and maximum responsiveness. Implement strategies to reduce computational overhead and ensure smooth operation even during high-load scenarios. |  |
|  |  |
| 15. Compatibility and Portability: |  |
| Ensure the OS is compatible with a range of hardware configurations and operating environments. Strive for portability to allow the OS to run on different platforms with minimal modifications. |  |
|  |  |
| 16. Version Control and Updates: |  |
| Incorporate version control mechanisms and a seamless update process to introduce new features, bug fixes, and improvements over time. Provide a mechanism for users to update their gripper systems easily. |  |
|  |  |
| 17. Security: |  |
| Implement security measures to prevent unauthorized access and potential cyber threats. Secure communication channels, user authentication, and data encryption should be part of the OS design. |  |
|  |  |
| 18. Maintenance and Support: |  |
| Offer ongoing maintenance and technical support to address user inquiries, provide assistance, and ensure the continued functionality of the delta gripper OS. |  |

By meeting these requirements, the Delta Gripper Operating System project aims to provide a robust, adaptable, and user-friendly software platform for efficiently controlling and managing delta gripper operations across a wide range of industries and applications.