Our software control system allows the user to control the platform movement. The modeling system enables the user to simulate the platform movement in a simulated environment and quickly switch to controlling the physical platform when the simulation is complete.

Platform Features

The platform had two main features we needed to design our software around.

1) The platform can accommodate a variability of motor modules. The platform could be constructed with between 1 and 4 motor modules. This required us to develop software that could facilitate movement with a variety of controllable wheels.

2) The platform also has a hardware limitation when controlling the wheels. The 4 motor modules are not controlled completely independent of one another. They are combined into bays of two modules each. The linear motion and rotation direction is shared through the bay.

Overall System

Our software control system allows the user to control the platform movement. The modeling system enables the user to simulate the platform movement in a simulated environment and quickly switch to controlling the physical platform when the simulation is complete.

Design

Our design focused on three areas.

1) Onboard microcontroller software

2) Off-platform control interface

3) The communication protocol

We chose to limit the amount of processing on the microcontroller and rely instead on the connected computer to perform any complex operations. This required us to create a robust communications protocol between the control interface and the onboard system.

Usability

Our control interface allows the user to manipulate the power going to each motor via keyboard input or a mouse driven UI. A command-line interface is also available.

Software Control System

Process

Our selection of the spiral methodology required us to repeatedly reevaluate our work based on the risks and unknowns of the project. This process was selected because our initial lack of experience with the subject domain made us feel that detailed planning too far in advance would not allow us to adapt quickly as we learned.

Lessons Learned

1) Research alternate ways of doing things rather than focusing on one that may not end up being available.

2) When selecting a process, plan in detail how it will be implemented.

Future Work

1) Adding a sensor of some kind and enabling simple automated navigation based on sensor input.

2) Creating a simulated environment modeled after a real-world environment for the simulated robot to navigate.

Modeling System

The primary features of the Microsoft Robotics Development Studio (MRDS) model are:

1) A full-scale 3D model of the existing robot

2) Programmable simulation software mimicking RP10 functionality

The model was produced using SolidWorks (version number...9? fix later) and Blender. The individual parts of the robot were modeled and assembled using Solidworks, and Blender was used to export the proper file type needed by MRDS.

The simulation software was written in C# following the MRDS service architecture. The MRDS services call commands in the control interface.

Robotics Software System: Sahil Verma, Karl Orosz, Adam Nabinger, Paul Berens, Joseph Wertz

Robotics Modeling: Jeff Kinner, Aaron Sevedge, Kyle Swift, William Nowak

Project Sponsor: Wayne Walter

Project Coach: James Vallino

Project

The robotics platform used in this project was created by a previous senior project consisting of Mechanical Engineering and Electrical Engineering students. Our team has two main goals to develop the platform further.

• The first is to create a software system that will be used to control the platform’s movement. This will consist of software onboard the platform and a control interface on a connected computer.

• The second is to create a simulation model of the platform in Microsoft Robotics Developer Studio (MRDS) which will allow users to preview movement control in a simulated environment. This system will also allow the user to quickly switch between controlling the simulated platform and the real-world platform.

Our design focused on three areas.

1) Onboard microcontroller software

2) Off-platform control interface

3) The communication protocol

We chose to limit the amount of processing on the microcontroller and rely instead on the connected computer to perform any complex operations. This required us to create a robust communications protocol between the control interface and the onboard system.

Usability

Our control interface allows the user to manipulate the power going to each motor via keyboard input or a mouse driven UI. A command-line interface is also available.

Software Control System

Process

Our selection of the spiral methodology required us to repeatedly reevaluate our work based on the risks and unknowns of the project. This process was selected because our initial lack of experience with the subject domain made us feel that detailed planning too far in advance would not allow us to adapt quickly as we learned.

Lessons Learned

1) Research alternate ways of doing things rather than focusing on one that may not end up being available.

2) When selecting a process, plan in detail how it will be implemented.

Future Work

1) Adding a sensor of some kind and enabling simple automated navigation based on sensor input.

2) Creating a simulated environment modeled after a real-world environment for the simulated robot to navigate.

Modeling System

The primary features of the Microsoft Robotics Development Studio (MRDS) model are:

1) A full-scale 3D model of the existing robot

2) Programmable simulation software mimicking RP10 functionality

The model was produced using SolidWorks (version number...9? fix later) and Blender. The individual parts of the robot were modeled and assembled using Solidworks, and Blender was used to export the proper file type needed by MRDS.

The simulation software was written in C# following the MRDS service architecture. The MRDS services call commands in the control interface.