Feb 8, 2013

Team 19: Documentation of Risks

Laboratory 1: Risk Reduction Techniques

Morgan, Laura
Miaw, Jireh
Hauser, Steven
Dworak, Catherine
Bertoglio, David

Work Product

This document discusses potential risks of this project in detail. For each risk listed in the pre-lab document, the problem is defined, the risk is explained, and how it was resolved or could potentially be resolved. A list of additional risks is also provided.

Document Revision Information

Feb 8, 2013 - Created

Approval Sheet

All group members whose names are listed below approve of the document and contributed fairly.

Member Names

Pledge

On my honor, as a student, I have neither given nor received unauthorized aid on this assignment.

Names

Table of Contents

Description of Listed Risks

Controlling the movement of the robot

One goal of this project is to get the robot to move. This includes moving forward, backward, and turning. The user must be able to make the robot move using controls on a GUI. This interface must be user-friendly – simple for the user to control, intuitive, and precise. The problem is how to make it have all the functionality required, display it in an easily understandable format, and allow the user to control it. This is a risk because users are unreliable and hardware malfunctions.

We created a prototype that moves the robot statically (forward, backward, and to both sides) using the differential pilot class. It did not have an interface to allow the user to control it, it was programmed to perform certain movements. This addressed the very low level task of getting the robot to move, but has not addressed the task of allowing the user to control it. We will address this problem throughout by first, finding a way to allow the movement to be controlled dynamically rather than statically, and second, design an intuitive interface for user control.

Communicating between the robot computer and the base computer

Data must be shared between the robot and base computers in order to make the robot move and store sensor data. User controls from the interface must be communicated with the robot to make it move, and sensor data must be transmitted back to the base computer to control dynamically solve problems or change settings. The problem is figuring out how to set up communication channel, what data to send back and forth, and what to do with it (how to use it to solve problems in the field). The risk in this is what to do if the communication channel fails.

We created a prototype that controlled the movement of the robot using USB. We programmed the controls and uploaded it to the brick, using USB, to run it. Later, we will figure out how to handle this communication using Bluetooth, rather than USB. Additionally, we have not yet dealt with handling loss of communication.

Robot computer capabilities

The NXT does not have the typical computing power of a modern-day computer, including the computer with the software that will be controlling the robot. Also, the 32-bit architecture of the NXT needs to interface properly with a possibly 64-bit controller. The NXT will likely become slowed by significantly large calculations as well. These are especially costly as the NXT needs to be constantly monitoring its environment with sensors and reporting or reacting to the changes in real-time.

To address these issues, the NXT should forward costly operations to the computer-based software, and rely on the computer-based software for operations that are not necessary for reactions or simple calculations. The software designed should also be a uniform 32-bit architecture to avoid complications between the NXT and the computer-based software.

Data transmission speed and latency

The transmission speed will be an issue when the robot needs to react quickly to its environment. If the transmission speed is quite low and the robot is waiting for signals from the computer-based software, the robot could not behave in time to the stimulus. There is also a risk when the robot becomes disconnected from the computer-based software, and will have to make some decisions on its own.

To handle these risks, small calculations, especially reactions to basic stimuli, should be handled on-board whenever possible. Additionally, there should be a backup system in place on the on-board software on how to react when it becomes disconnected. The on-board software should attempt to reconnect while simultaneously having a simple on-board backup mechanism to solve basic issues that it would normally solve with the computer-based software.

Additional Risks:

Meeting attendance and scheduling conflicts

When the entire group cannot meet at the same time, it will be difficult to synergize our efforts and avoid redundancy. Those who miss the meeting will also possibly miss out on work assigned to them or tasks completed at the meetings. To fix this, the group should schedule meetings far in advance and make sure to find a time that is hopefully accessible for everyone, or at least the maximum number of people.

Time constraints

The later the group attempts to complete the project for the week, the risk of not being able to complete the project in time increases. Additionally, time constraints make it more difficult to meet, as it will be less likely to find a time that fits everyone's schedule at a late notice. To address this risk, the group should attempt to start the work as early as possible.

Physical damage

There is risk in the operation of the robot in regards to its physical structure. High-velocity collisions may cause pieces of the robot to fall off or break. Additionally, simple stress on the structure of the robot due to it's motors can lead to breakage without a collision if it's poorly designed. Throwaway-prototype testing can address this risk, as well as having a good software implementation to detect collisions or minimize their effect with the robot's behavior.

Dealing with terminating errors while robot is running

There is a risk with software errors disrupting the ability of the robot. The robot could enter an "infinite loop" with improper code, or encounter an error handling a certain stimulus. Such errors can be dealt with by reporting them to the computer-based software, which should implement methods to solve or respond to errors that occur while the robot is running.