System Document

Project: LIBERTY

Task:

Document Version Number: 2.0
Date: 15/10/2017
Author: Andi-Camille Bakti, Bill Zhang, Edward Son, Gabriel Negash, Claire LIU
Edit History: https://github.com/Gabetn/DPM_01_Project_Documentation

TABLE OF CONTENTS

2. SYSTEM MODEL

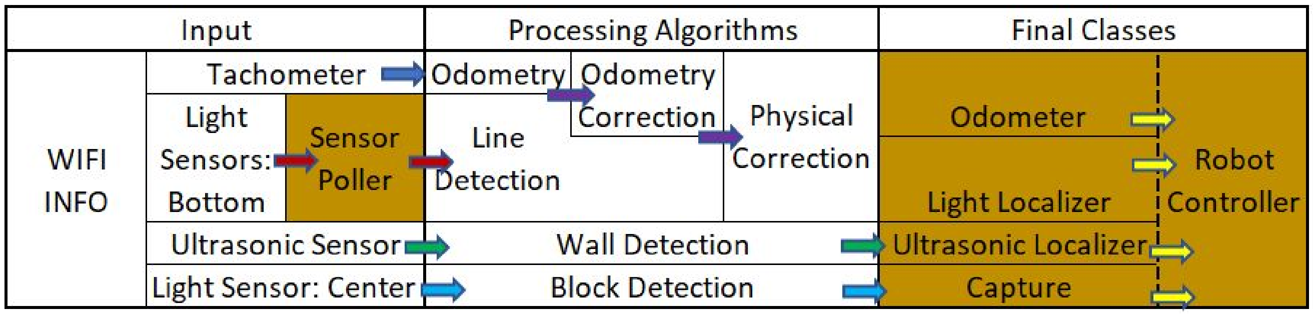

Figure 1: System Block Diagram

In order to minimize unnecessary system complexity and minimize component dependencies/concurrency a streamlined system model is required. This is visualized in Figure 1, and by aiming for as little dependency and concurrency as possible the system is less prone to failure as well as more manageable to develop. As our team is a team of 5 we have a total of 54 fewer total hours available to our cumulative budget, requiring as simplistic a model as feasible in order to compete with teams of 6. In order to simplify software design the hardware design must be developed in such a manner as to allow for a capturing algorithm which only makes use of 1 sensor, a light sensor. This is elaborated in further detail in sections 3 & 4.

3. HARDWARE AVAILABLE AND CAPABILITIES

(In the sense of the design problem you have, there are three hardware definitions. First, the Lego components and their mechanical capabilities. The sets limit the structures you can build. What are these limitations? Second, there are the electromechanical limitations. Third there are the electronic/processor constraints (e.g. – how fast can you execute code?). All these constraints need to be identified and listed.)

See reference at ***CON - GEN; 3.0***

4. SOFTWARE AVAILABLE AND CAPABILITIES

The primary tool in the software development is Eclipse IDE with LeJOS component installed. Hence, it is mandatory to write the system in Java since LeJOS is only compatible with

Java. There are no alternative tool within our scope, thus there is no comparison to determine its advantage or disadvantages. With the absence of comparing advantage, Eclipse IDE is a very well developed environment with debugger and JavaDoc auto-generation in terms of its absolute advantage, and it is most widely used for Java development and more popular than other IDEs.

Integration is a very vital part of the software system when all the subsystems are ready. There are various code-sharing softwares that offers this kind of functionality. Slack offers file sharing features and it is used for our internal team communication. Thus, the advantage is that Slack can integrate team-wide communications and code-sharing into one system and it is completely private only to team members. However, its disadvantage is that there is no version control feature in Slack. Google Doc is another file-sharing system we are using. It is used for storing and edit the project documentation. It has version control feature as an advantage, and it can integrate documentation and code-sharing into one system. However, Google Doc is not designed for code sharing. A lot of features for code-sharing are missing. For example, it has no branch features. GitHub is the best option for this because it has a version control system and it is solely designed for software development. It can offer everything Google Doc and Slack can offer with additional features for software development. Also, GitHub is very popular among the programming community, thus there is no training cost for the software team to adopt to it.

See reference at ***Software - Design and CON - GEN; section 4.0*** to see the software constraints.

5. COMPATIBILITY

The design of the system should fit with the Lejos EV3 kit, which is composed of lego components, a brick, motors and sensors. There are but four ports for motors, and four ports for sensors. Thus, we are limited to four motors and four sensors. Wire management for these components will be important to handle, since loose wires may interfere with components such as the sensors. The compatible software with the brick is the Lejos project, which uses Java, can be developed and run in Eclipse. Eclipse has all the necessary plugins for Lejos sensors and motors. There may be compatibility issues with third party software, thus the latter will not be used. However, there will need to be a connection through wifi to a server, which will distribute the necessary data for the final demo. Thus, the system is wifi compatible. The ultrasonic localization developed in the previous labs is compatible with the current system, as it is one of the only ways to localize. There is no need to develop another solution as the aforementioned one works well, and is compatible. The odometer class is also compatible with the system to be designed, since it will be using a similar navigation system. Light localization is also compatible with the current system, however that would restrict the number of light sensors for localization to one. Navigation in the previous labs assumed minimal distance travel, which means the system would be able to travel diagonally. Depending on our decision for correction, this may or may not be compatible with the current solution, as a cartesian way of travelling may be desired. The zipline traversal subsystem is new, and compatibility needs to take into account the available motors. The component hooking onto the zipline will need to somehow turn to get the whole system across said zipline. Thus, it needs to fit on to the coloured end of the EV3 or NXT motors. A 3D printed piece could be compatible, as there is flexibility in the creating of said piece. It would be easy to make it such that it can be screwed into the turning component of the motor.

Refer to ***REQ - GEN*** which discusses the compatibility of each subsystem and the research required.

6. REUSABILITY

A large portion of the system is inherited from the previous lab period. During the 5-week lab period, multiple programs that can be used for sub-systems were developed. Navigation system is inherited from the navigation class written during the lab period. The same applied to Ultrasonic Localizer, Odometer and robotControl with minor changes. Besides the sub-system reusage, the overall architecture and software design are also inherited from the lab development. For example, the design of having a controller class to arrange all the concurrent and sequential processes are used in all previous labs.

See reference for details in ***Software*** - ***Design*** and ***Software - Architecture.***

7. STRUCTURES

The mechanical structures used were made from Lejos MindStorm kits (as imposed by the constraints see document ***CON - GEN; 3.0***).

See reference for details at ***Hardware - Design*** about the structures and the reasons behind their design.

The design of the software structure was based on a MVC pattern. Details and rationale can be seen in the document: ***Software - Design.***

8. METHODOLOGIES

(Approaches being taken in all parts of the design and, for software, the basic algorithms to be used. These are really lists of the possible candidate solutions for parts of the problem. They come out of the Ideas Generation phase and will allow a critical analysis before the final design is performed.)

We explore different approaches during the design of the different components. Regarding subsystems that we have already addressed in previous labs, they require less work since we had already produced a possible solution. However, subsystems such as the double sensor light localization, zipline mounting and dismounting, and the capturing system have to go through a design process.

The team's main concern during the initial analysis was to focus on simplicity over speed and precision. This decision arose from the analysis of the tools and hardware at our disposal (see ***CON - GEN; 3.0***) that were not produced for precision. Most notably the sensors produced considerable noise and the system was heavily reliant on the battery voltage level. Hence the decisions to try and increase our precision using two bottom light sensors and a cartesian displacement rather than polar (see ***Software - Design*** for the detailed rationale). To tackle the issue of precision during mounting and capturing we implemented a hardware solution using a funnel to provide a correction (see ***Hardware - Design***).

9. TOOLS

At our disposal, we are limited to the tools specified in the ***CON - GEN*** document however permission needs to be given by the client for any external hardware or software use.

See reference for more details at ***CON - GEN; 3 and 4*** for Hardware and Software Constraints respectively.