ECSE 211 - Design Principles and Methods

SYSTEM DOCUMENT

GROUP 11
John, Durham, Alex, Ian, Ethan

Version 2.00
29th March 2017

Abstract

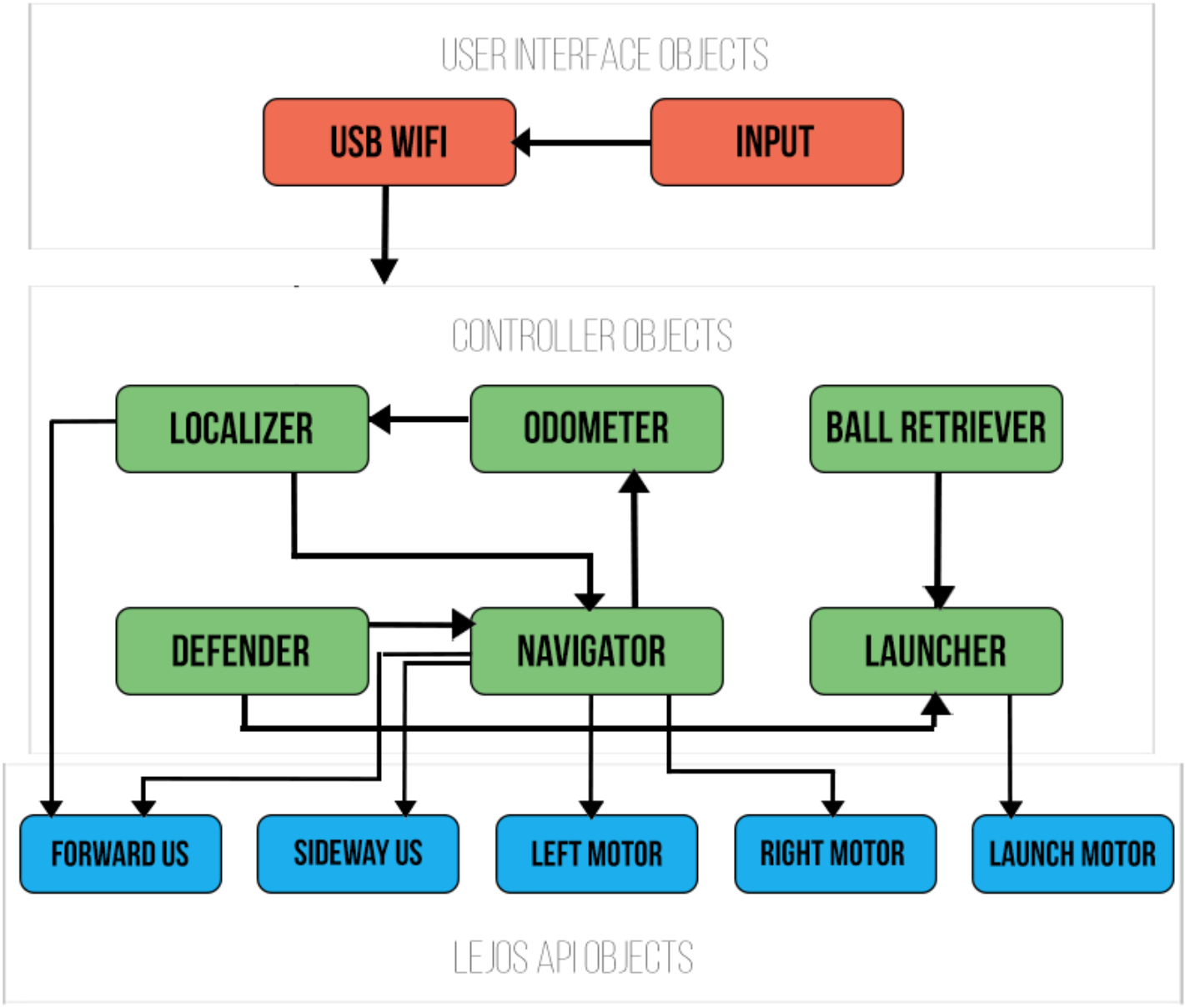
This document is intended to identify and define the solution environment, i.e. the tools and basic components we can use to construct a design, which satisfies our requirements. This document identifies the capabilities of all parts available for the construction of the final product both software and hardware. This document will outline the components necessary to the development of a solution to our design problem.

Contents

1 EDIT HISTORY

- February 19th 2017:
  - **Ian Smith:** Initial set up of document sections (including table of contents)
- February 20th 2017:
  - **Alex Lam:** First draft of section 10
  - **John Wu:** First draft of section 2 (with Ethan), 4, 6, 7 (including 7.1, 7.2 and 7.3) and 8 (including 8.1, 8.2, 8.3, 8.4, 8.5 and 8.6)
  - **Ethan Lague:** First draft of section 2 (with John) and 3 (including 3.1, 3.2 and 3.3)
  - **Ian Smith:** First draft of section 5 (including 5.1 and 5.2)
  - **Durham Abric:** Reference in section 9
- February 22nd 2017:
  - **Alex Lam:** General edits to formatting and changes to sections 5.1, 5.2 and 10
- March 29th 2017:
  - **Ethan Lague:** Added to section 3.1 and 3.2, changed 8.1, 8.2, 8.5, 8.6.

2 SYSTEM MODEL



3 HARDWARE AVAILABLE AND CAPABILITIES

3 Lego EV3 "Mindstorm" kits:
These kits include various structural pieces, connection pieces, bars, bar caps, wheels, gears, tires, tracks, 12 large motors, 3 medium motors, 3 ultrasonic sensors, 3 rechargeable batteries, 3 color sensors, 3 gyroscopes, 6 touch sensors, 3 EV3 bricks, 3 USB cables, numerous RJ12 cables, 3 chargers, and various other specialized Lego pieces.

3.1 Mechanical Limitations

The limit of the amount of pieces that are available will not be an issue since there are more than enough of every piece. The mechanical limitations will come from the fact that given the nature of Lego pieces, it will be difficult to build certain designs. The pieces are connected by pieces that fit inside of the holes and all of the holes are separated by a fixed length. This makes the design process more difficult since there are only a couple of ways to connect each piece to one another. Another mechanical limitation comes from the fact that Lego is made of plastic and therefore will bend under certain pressures which could affect the calculations for certain aspects of the robot such as the ball launcher. A third mechanical limitation will come from the wheels of the robot. It is inevitable that there will be a certain amount of slip between the wheels and the floor surface which will decrease the accuracy of our odometer.

3.2 Electromechanical Limitations

The motors are limited in the amount of torque they can produce. The motors we used have a maximum running torque of 20 Ncm, and a stall torque of 40 Ncm. The motor also has a top rotational speed of 160-170 rpm however this number is dependent on battery life. This can affect our calculations for the ball launcher since it will be powered by motors and an inconsistent rpm can cause inconsistencies in the accuracy of the ball thrower. We will not have this issue with the motors used for driving the robot since we will never need to hit speeds this high. Because of the inconsistencies, we must be conscious of the amount of power the battery will use throughout the round so that it is able to fire the ball with as high an rpm as possible. The motors also have an error of one degree in the tachometer which can be accumulated to accrue a large error when the robot is traveling far distances. This is why odometry correction is mandatory. The EV3 ultrasonic sensors are accurate with an error of 1 cm with a range of 1 to 250 cm. These limitations should not be a concern given what we are asking our ultrasonic sensors to do. The limitation on the color sensor is that it samples at a rate of 1 KHz. This shouldn't be an issue but must be kept in mind when determining the speed of the robot. Another limitation lies within the nature of the color sensor where depending on the exterior lighting, the values might be inconsistent. We solved this issue by implementing a correction for when the light sensor misses a line on the field. The RJI2 cables will also limit our robot design since we must leave enough room in front of the ports so that the cables can be plugged in.

3.3 Electronic Limitations

The EV3 brick runs an ARM9 chip at 300MHz which should be sufficient for our program. We will try to keep the number of threads to a minimum to ensure that the EV3 will be able to run the program with no issues.

4 SOFTWARE AVAILABLE AND CAPABILITIES

Java: See Constraints Document, Section 4.

LeJOS: See Constraints Document, Section 4.

Eclipse: A commonly used Java IDE which has a plugin for the LeJOS API. This plugin also comes with an interface for easy control of communication with the EV3 brick. However, Eclipse often runs very slow and will sometimes crash a lot.

IntelliJ: A newer and increasingly more popular Java IDE which runs faster than Eclipse, with much more intelligent syntax correction. However, it does not have the LeJOS plugin and jars need to be imported manually. Communication with EV3 brick needs to be done through SSH as well.

Github: A web based version control system which uses git. The ability to work on different branches allows users to collaborate and work simultaneously on the same project. Conflicts may occur if two users work on the same lines of code, in which case higher level of experience in git is needed to solve efficiently.

Travis CI: An automated build system that is integrated into github. It follows a build file to clean, compile, and run any tests every time a commit is made onto the master branch to ensure the software is working at all times. Needs constant monitoring to ensure build script works accordingly.

LaTeX: A document preparation system to allows users to make clear and concise reports without having to worry about formatting. Usage of the tool requires knowledge of the LaTeX language and syntax.

Overleaf: A web based editing system for documents that use LaTeX. Allows users to collaborate on the same document and also includes a version control system. Documents can also be cloned as a git repository allowing edits to be made on a local development environment.

Gantt Project: Basic project management software that allows detailed Gantt Charts to be created. Unlike other modern applications, it is not a cloud based software making it hard to collaborate and share amongst team members.

Dropbox: A cloud storage platform that can be utilized to store all documentation and other relevant files.

Google Calendar: A Google application which allows the ability to create shared calendars to keep track of meetings and team member schedules.

Slack: A new open source communication and project management platform that allows custom integrations that range from multiple areas. Current integrations that are being used are:

- Busy Bot - Used to assign tasks and keep track of tasks that need to be finished.
- Google Calendar - Sends daily notifications for all events in shared Google calendar.
- Github - Sends notifications every time there is activity in the Github repository.
- Travis CI - Send notifications of every build result.
- Hour Stack - Allows team members to clock their hours simply by starting a timer when they start working and stopping it once they are done.
- Documents (Custom) - Custom made integration that prints a list of the links to all our LaTeX documents.

5 COMPATIBILITY

5.1 Software

The programming of the robot will be done in Java. The EV3 programming brick's default setup does not support Java so alternative firmware is provided. The Mindstorm environment run on top of Linux. The micro SD card contains a different version of Linux that has the LeJOS EV3 environment as a replacement user interface. Using the LeJOS EV3 environment in Eclipse, the robot is programmed using Java.
The programs developed in the weekly labs will be integrated into the robot in different ways. The code for the Wall Follower will not be completely compatible with the rules of the competition nor will it be efficient enough to be able to properly perform in the competition. Thus, the wall follower code will need to be heavily edited and improved before it can be integrated into the current project. The wall follower code will indeed need to be modified to account for the new design of the robot as it was noticed in past labs that larger robots had more trouble turning corners. The code from the odometry lab will be used to allow the robot to keep track of its position on the field as it performed remarkably well. The code developed for the Navigation lab will be used but adjusted significantly due to the nature of the competition being more complex than the one week lab. The code for localization used in lab 4 will be completely reworked due it not being compatible with the added ultrasonic noise mentioned in the Requirements Document under "2.5 Operating Environment - 2.5.4 Ultrasonic noise". The ball launcher code used in lab 5 will need to be adjusted to be compatible with the rules of the competition.

5.2 Hardware

Robot consists of parts only found in the EV3 Mindstorm kits each member has. Consequently, the different hardware components are highly compatible with each other. There are,however, certain constraints, such as the way the Lego pieces attach to each other, that limit the compatibility of pieces. Due to the new requirements of this project, most of the robot designs from previous labs will need to be modified before they can be integrated to the current project. As of Wednesday February 22nd compatibility issues with

the launching arm have been identified. All launching arm designs from previous projects were not designed to accommodate a standard issue table tennis ball and must be modified in order to do so.

6 REUSABILITY

See Requirements Document, Section 3.

7 STRUCTURES

7.1 Mechanical Structures

Wheels: Two of main wheels are to be used for control and navigation with a small back wheel for balance and stability, and a small front wheel from preventing the robot from singing forward when launching the ball. Keeping consistent from the labs, a simple design like this will keep everything simple if built properly. Limited wheels will also reduce the error margin for friction and slipping.

Launcher: The design of the launcher will be kept as basic as possible. It will reduce resources and allow more time for stabilizing and perfecting a simple design. A tower like structure is built up from the EV3 brick and two motors are attached horizontally on top. An arm is extended down the back of the robot with a claw-like structure at the bottom where the ball will rest. See Hardware Document for a more specific description.

7.2 Electrical Structures

Ultrasonic Sensors: The design will consist of two ultrasonic sensors placed statically. One front facing under the EV3, and one facing to the left. See Hardware Document for a more specific description.

Motors: The design will consist of four large EV3 motors. Two of the motors will be attached to the two main wheels to move the robot. The remaining two motors will be used for the luncher. See Hardware Document for a more specific description.

7.3 Software Structures

Refer to System Document, Section 2 for software architecture model.

User Interface Objects: The only interaction from the user will be through a user interface screen that is connected to an input object on the layer. This object will send commands via WiFi to a server that will be hosted locally. The WiFi will then initialize the proper controller objects in the middle layer.

Controller Objects: Controller objects will be the bulk of the application. Objects that include a Localizer, Odometer, Ball Retriever, Defender, Navigator, and Launcher will work together to properly control the LEJOS API Objects in the bottom layer.

LEJOS API Objects: These objects each connect to a corresponding electrical hardware component. Each of the sensor objects will return an input from its corresponding hardware component to the controllers whenever they are requested. Each of the motor objects will output a physical movement to the corresponding hardware component when the command is sent from the controllers.

8 METHODOLOGIES

8.1 Navigation

The Robot will always navigate in the middle of the tiles driving parallel to one set of lines. Odometry will be calculated using the tachometer in the same way it was used for the labs. Since the robot has to travel long distances, odometry correction is implemented using two light sensors. When he robot comes across a line perpendicular to its velocity, once the first light sensor hits the line the corresponding wheel stops and waits for the next light sensor to reach the line. the odometry is then corrected to the exact odometry of that line.

8.2 Localization

The vehicle will start by rotating in place and using a falling edge method on the front-facing ultrasonic sensor to find the side wall. Once found, it will continue to rotate along the side walls and use the ultrasonic sensor to keep track of its distance to the wall. It will then take the two minimum values of the ultrasonic sensor, and register them as the x and y coordinates depending on which square it starts in. If the values are not precise enough to produce a quadratic function, the robot will repeat this process. Once it has the x and y values registered, it can find its current theta orientation and then travels to the center of the square it started in to commence navigation. All this will be implemented in the localizer object.

8.3 Obstacle Avoidance

The obstacle avoidance will use both ultrasonic sensors for support. Whenever the robot reaches the middle of a tile, it will stop and evaluate its next move based on the navigation algorithm. It will turn to face its next move and then swivel so the front sensor can check the front tile and the left sensor can check the tile to the left of the robot for obstacles. If an obstacle is detected by the front sensor, the robot will find the tile around it with the next priority and travel there.

8.4 Boundary Avoidance

Boundary avoidance will use a mapping of the field that is initialized accordingly based on user inputs. It will use odometer values to determine its position within the boundaries. If a boundary is approaching, it will implement a wall follower method except it will replace the ultrasonic sensor readings with distances calculated from the odometer. Just like the obstacle avoidance, this will be implemented into the navigator object and interrupt the navigation thread anytime a boundary is detected.

8.5 Retrieving Balls

In order to retrieve balls, it will use the mapping of the field initialized at the beginning of the program to know the exact position of where the balls are located. Upon retrieving this position, it will then use the navigator class to move the robot to the tile next to the desired location to retrieve the ball. It will then lower the throwing arm from its position on top of the robot. Once the arm is down, the robot will travel to the line perpendicular to the ball dispenser and line its wheels up to the line using the color sensors. It will turn 90 degrees away from the dispenser and line up to the next line parallel to the ball dispenser using the same mechanism. It will then back up 12.24 cm where it lift the arm 47 degrees so that the claw lines up perfectly with the ball dispenser where the robot will emit a beep to signal for a ball to be dropped.

8.6 Scoring

In order to score, the robot will begin by using the navigator object to move into a position that is optimal to shoot from. It will line up to the the line perpendicular from the target and then the line parallel to the target to correct its odometry similar to ball retrieval. The ball launching motors will then rotate the arm at a specific rotational velocity and to a specific release angle depending on the square the robot is shooting from which will be determined by the obstacle detection and the offensive line passed through wifi.

9 TOOLS

See System Document, sections 3 and 4 (Hardware/Software available).

10 GLOSSARY OF TERMS

- RJ12 cable: Connector cables used to connect the main EV3 brick to other electrical components.
- Ball launcher: The mechanism used to launch the ball
- Mindstorm EV3 kit: This is the hardware kit containing the majority of the hardware components to be used in this project.
- API: The Application program interface is a set of routines, protocols, and tools for building software applications
- Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network
- Slack integration: an addition to a specific slack channel that allows for additional functionalities by the users of the channel
- Slack channel: A channel in the slack application through which users communicate. Multiple slack channels can be created at once.
- EV3 programming brick: The main part of the mindstorm kit. This is the component of the robot on which the software will run.
- Lever: Part of the robot used to launch the ball. Also referred to as the launching arm.