SYSTEM DOCUMENT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ICARUS

Version: 2.0

November 27th, 2017

Developed by Team 08

**Project**: ECSE-211 Design Project: Game of Capture the Flag

**Task**: The goal of this project is to construct an autonomous machine that can play a one-on-one version of the game Capture the Flag, performing tasks such as navigation, localization etc.

**Document Version Number**: 2.0

**Date**: November 27th, 2017

**Author**: Ezz Aboulezz

**Edit History**:

|  |  |
| --- | --- |
| Date | Purpose |
| Oct 16 | Added hardware available and capabilities |
| Oct 17 | Added software available and capabilities, tools, re-usability |
| Oct 17 | Added software compatibility and methodologies |
| Oct 17 | Edited hardware available and capabilities |
| Oct 18 | Added structures and completed document |
| Oct 25 | Fixed author to one person |
| Nov 27 | Final edit and re-structure |

1. [TABLE OF CONTENTS](#_top)

[SYSTEM MODEL](#SYSTEM)

[2. HARDWARE AVAILABLE AND CAPABILITIES........................................... 3](#HARDWARE)

[3. SOFTWARE AVAILABLE AND CAPABILITIES............................................. 4](#SOFTWARE)

[4. COMPATIBILITY.................................................................................................5](#COMPATIBILITY)

[5. RE-USABILITY.................................................................................................... 6](#REUSABILITY)

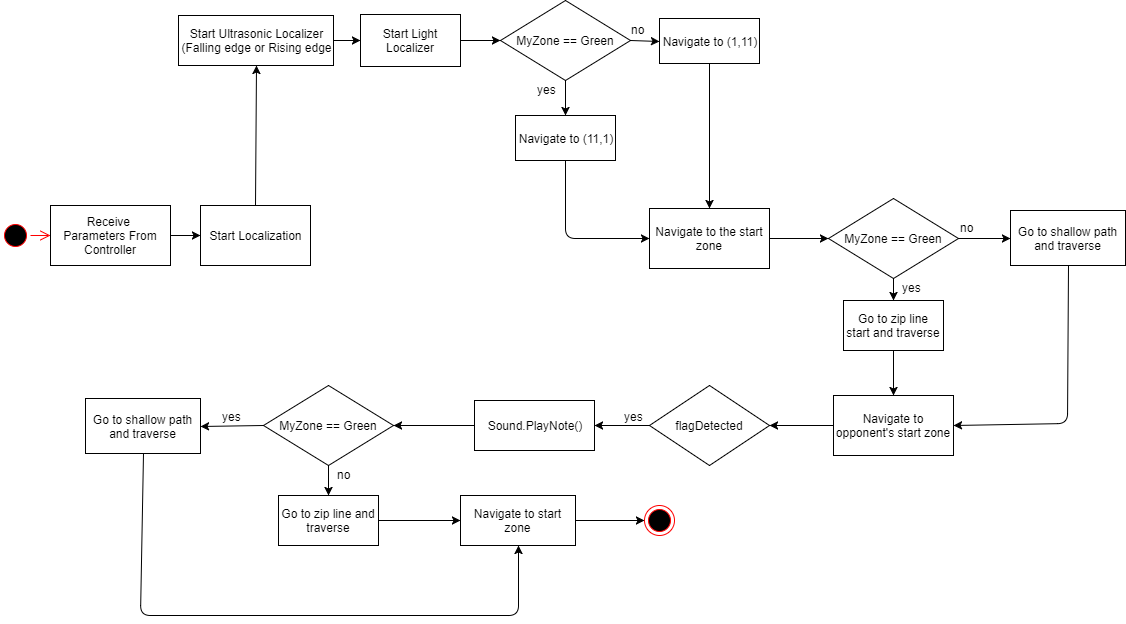
[6. STRUCTURES...................................................................................................... 7](#STRUCTURES)

[7. METHODOLOGIES..............................................................................................7](#METHODOLOGIES)

[8. TOOLS................................................................................................................... 8](#TOOLS)

[9. GLOSSARY OF TERMS...................................................................................... 9](#GLOSSARY)

**SYSTEM MODEL**



1. **HARDWARE AVAILABLE AND CAPABILITIES**

[***\*See Constraints Document, Section 3.0 Hardware Constraints, for further details on the constraints imposed by the hardware equipment.***](Constraints%20Document.docx)

There are three available Lego mindstorms kits, since the team is composed of three groups, each includes various lego pieces for the mechanical design along with electromechanical sensors and motors. Each kit is provided with an EV3 brick, which is a general purpose controller that is comprised of four input ports and four output ports. The input port can connect to the sensors. The output port can connect to the motors. A kit is composed of an ultrasonic sensor, a color sensor, a touch sensor and a gyroscopic sensor, as well as one small motor, two NXT motors and two EV3 large motors. A USB cable is also provided for uploading code to the brick. The kit comes with a micro SD card and WI-FI capability.

Some of the components have the following limitations:

* The motors have limited torque based on the power supplied from the controller. To accommodate the motors, a strong design must be built and weight distribution is crucial to avoid high wheel slipage. A lightweight build will have less issues related to the motors. Also, the accuracy of the motors’ tachometer can vary since there are deformations on the toothed wheels that develop over time.
* The processor is a Texas Instruments Sitara AM1808 microcontroller, which is based on the ARM 926EJ-S core running at 300 Mhz. It is fast enough to execute the Java code that drives each component.
* Color Sensors: They can be used in two different modes; color mode and light intensity mode. They have a sample rate of 1 kHz/sec. Color mode can recognize 8 different colors. The sensor must be set up at a right angle and must be close to the examining surface.
* Ultrasonic Sensors: These sensors measure the distance from an obstacle by generating waves and based on the time it takes to receive their reflection the sensor can calculate that distance. The detecting range of the sensor lies in between 1 cm and 254 cm and the sensor can detect the distance within 1 centimeter of its correct value.

In addition, the overall performance of the system will degrade as the voltage of the battery drops. Lower battery voltage makes the sensors unreliable and renders the robot somewhat unusable. Furthermore, the brick has 16MB of Flash memory and 64Mb RAM. The 2.0 USB can communicate with a PC at speeds up to 480 Mbit/sec.

1. **SOFTWARE AVAILABLE AND CAPABILITIES**

The software for the robot is developed using the leJOS EV3 virtual machine. It offers all the classes from the EV3 API (available on: <http://www.lejos.org/ev3/docs/>) and includes a full Java runtime system. Here is a summary of what this package provides:

* Object-oriented language: The use of an object-oriented language such as Java makes it very easy to separate the code in different classes and keep a clean and logical structure.
* Preemptive threads: The robot must be capable to perform several tasks at the same time and thus have multiple processes run in parallel (motors’ action, sensors’ data collection…). The use of threads is therefore mandatory.
* Exceptions: The use of exceptions is very effective when testing and analyzing the code. We know exactly what happens and it make it easier for debugging.
* Classic Java data types and libraries: The leJOS virtual machine is an additional feature, which does not detract from the normal development in Java.

The software is therefore built to fit the Mindstorms bricks and provides everything needed for the correct development of our robot.

Other software availbale to use:

* The LEGO Digital Designer provides a fast and accurate way of designing the robot and removes the inconvience of having to physically dismantle a design if the requirements are altered. However, components not present in the LEGO Mindstorm kit cannot be represented using this with this designer hence they would have to be built and tested physically.
* Github, a version control tool, will be used to keep track of changes to the code. This tool gives the team access to the latest version of the code at any time. This project’s code can be found on: https://github.com/AntonValk/Robot-Competition-Capture-the-Flag. This is a public repository available to all team members.

1. **COMPATIBILITY**

Apart from the issues discussed in the Compatibility Component Reuse in the Requirements Document, there are few compatibility points to be mentioned with regards to the hardware. The robot is mainly made from Lego bricks, which can be easily joined together. These Lego bricks are structured so as to hold the brick as well as the motors and sensors together. The Lego skeleton of the robot is compatible with the brick, the sensors and the motors since the brick is relatively small and the motors can be easily adjusted under the brick. Each of the robot’s motors and sensors is connected with the EV3 brick by a wire in a specific port so that the brick can easily process the data from the sensors and control the motors.

Also, the Lego components are compatible with the software assuming we have defined the same port in our code as the actual port the components are plugged in. In order to get access to the data from the sensor, the software system creates an instance to that sensor with respect to the specific port that it is connected with. The software will not receive sensor readings if it gets information from the wrong port.

The Requirements Document discusses the compatibility issues with respect to third party products. The software is mainly constructed using Java multithreading and concurrency, as well as Java data types and libraries. The LeJOS EV3 virtual machine is also evolved in the software system, where multiple LeJOS hardware and robot libraries are imported. It is fully functional with Java Eclipse environment without compatibility issue. For the purposes of this project the required running environment is Java 1.7 as leJOS has been optimized for Java Runtime Environment (JRE) 1.7. Also, in order to run the software, the brick needs to have the right IP address; otherwise the software will not detect the brick.

Finally, the code for connecting the robot to the competition server is provided (WiFi connection class). The robot will need to receive the parameters for the run that will define all the relevant coordinates such as the zipline position, starting corner etc..

1. **REUSABILITY**

The labs are considered to be our R&D phase and the code from these labs can’t be re-used as they are; they must be modified to be compatible with the most recent robot design. The labs were completed with a single Lego kit whereas three kits are available for the project. This implies that there might be more motors and sensors connected to the brick. Furthermore, the code must be compatible with the Wi-Fi code for connection to the competition server.

The labs entailed the following tasks:

* Lab 1: Navigation around a sequence of wooden blocks, making up a wall containing gaps and corners, without touching or deviating too far from the wall.
* Lab 2: Determining the accuracy of the implemented odometry system, and implement a correction using a light sensor.
* Lab 3: Driving the robot to a position on the field while avoiding obstacles, by use of the odometer and an ultrasonic sensor.
* Lab 4: Localization and usage of the ultrasonic sensor and light sensor to accurately navigate the robot to a known initial position and orientation on the field.
* Lab 5: Usage of the newly acquired pulley and third motor to develop a method that allows the robot to mount, traverse and dismount a zipline.

The following are the classes developed throughout the labs:

* Bang-Bang: controls the distance of the robot from the wall and applies a constant correction to the motors' speed.
* P-type: controls the distance of the robot from the wall and applies a correction to the motors' speed depending on the distance.
* Ultrasonic Poller and Controller: facilitates the data collection for the Ultrasonic sensor and transmits the distance measured to the EV3 controller.
* Odometer: computes the displacement position and angle using tachoCounts. This allows the robot to precisely determine its position and direction.
* Navigation: uses Odometer to know where the robot is and computes the direction and the distance required to reach the target.
* UltrasonicLocalizer: uses the US sensor to get the distance from the wall as the robot rotates. This is used to know the zero degrees direction and compute the displacement angle the robot has to make to be aligned with zero degrees.
* LightLocalizer: rotates and touches the four axes (y-, x+, y+, x-) to understand the robot's precise starting position and know where the origin point is. Then, uses the Navigation class to travel to the origin.

1. **STRUCTURES**

Many mechanical and electrical structures used in previous labs will be used to build the robot. Each of them will allow us to easily implement our software, get efficient data from the sensors and ease the brick's mobility.

First of all, wheels will have to be placed under the brick and each of them will be connected to a motor so that the software will control the speed of the robot. Only two wheels will be needed since the EV3 brick contains only 4 output ports meaning that a maximum of 4 motors can be used for the robot. In order to keep the robot balanced, a rolling metal sphere will also be installed under the brick. We will also make sure to place the ultrasonic sensor in front of the robot and connect it to a motor to allow the software to change its orientation. Depending on the situation (Wall Following or UltrasonicLocalizer), the ultrasonic sensor's heading will be oriented to fit the mode. Furthermore, the light sensor will have to be placed at a compatible distance from the brick to adequately get measurements from the ground for the LightLocalizer. Additionally, a pulley will have to be placed on top of the brick and be connected to a motor. By doing this, the robot will be able to mount and traverse the zip line.

Finally, our software structure will mainly contain threads. This method will optimize the robot's ability to perform many tasks at the same time. It will be able to continuously receive and analyze data while moving around the platform and execute what it has to do.

1. **METHODOLOGIES**

The basic design approach is to identify the goals/requirements, generate ideas and evaluations, design prototypes (both hardware and software), test prototypes, and finally choose the solution based on the results of the tests. If the prototypes fail the test, we go to the iteration of idea generation, and repeat the process of designing and testing the next prototype.

The idea for the software algorithm is based on the Java multi-threading feature. All the necessary threads for the design are running after the main method receives input, which includes the button press, the data from the sensors and remote inputs from user’s computer. Typical threads include the odometer, the display, the navigation thread, the localization thread and the zip-line thread. The system starts or stops the corresponding thread based on the inputs. The algorithm for each thread is shown below:

* Odometer: starts at the beginning of the program, calculates the changes of the robots’ position based on the motor speed.
* OdometerDisplay: starts at the beginning of the program, consistently gets the odometer readings and displays them on the screen.
* Localization: starts when user inputs are loaded. It localizes the robot using the US sensor in either RisingEdge mode or FallingEdge mode and then sets the odometer readings based on the result. Then it navigates to the origin based on the results from LightLocalization. This thread stops when the robot orients itself to the expected point.
* Navigation: starts after Localization stops. It calculates the path that the robot is expected to travel and navigates the robot along that path. This thread stops if the robot completes the path.
* Zip-line: started when the robot reaches mounting point of the zip-line.
* Flag search: starts when the robot reaches the flag search area and then tell the robot to look for a certain color that represents the flag.

1. **TOOLS**

[***Please refer to section 2 on Hardware Capabilities and section 3 on Software Capabilities for the relevant tools information.***](#HARDWARE)

For any project to be successful, there needs to be good communication between members. Slack will provide the team with proffessional means of communication while Facebook group chat will provide the group with a faster and more efficient means of delivering messages.

To organize all the documents related to this project, they were all written on DropBox to be shared and viewed by all members. All the documents can be found in a folder on the website. DropBox has a version control system hence all document history is available for the group members. Furthermore, the clients specified that all versions of the documents are to be submitted in the Dropbox folder. This includes the Requirements Document, the Constraints Document, the Capabilities Document, the System Document, the Software Design Document, the Hardware Design Document, the Testing Document, the Budget spreadsheet and the Gantt chart. All these documents are on DropBox, along with the Meetings Document and several diagrams and pictures.

1. **GLOSSARY OF TERMS**

Eclipse: an integrated development environment (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming).

Dropbox**:** a service that offers cloud file storage and synchronization.

GitHub**:** a web-based [Git](https://en.wikipedia.org/wiki/Git) [repository](https://en.wikipedia.org/wiki/Repository_(version_control)) [hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service).

EV3 brick: Lego Mindstorms EV3 is the third-generation robotics kit in [Lego](https://en.wikipedia.org/wiki/Lego)'s Mindstorms line.

Java: a computer programming language that that is concurrent, class-based, object oriented and designed to have as few implementation and dependencies as possible.

LeJOS: leJOS is a [firmware](https://en.wikipedia.org/wiki/Firmware) replacement for [Lego Mindstorms](https://en.wikipedia.org/wiki/Lego_Mindstorms) programmable bricks. It includes a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine), which allows Lego Mindstorms robots to be programmed Java.

Mindstorms: The Lego Mindstorms series of kits contain software and hardware to create customizable, programmable robots. They include an intelligent brick computer that controls the system, a set of modular sensors and motors, and [Lego](https://en.wikipedia.org/wiki/Lego) parts from the [Technic](https://en.wikipedia.org/wiki/Lego_Technic) line to create the mechanical systems.