Software Design Document

**I-Design Overview**

The goal of the project is the following: Construct an autonomous robot to play a one-on-one game that is a cross between soccer and basketball. The robot must be able to play either forward or defense and be capable of navigating the field without hitting obstacles. Instructions are received via Wi-Fi prior to the game.

Our program starts executing in the EntryPoint class. After the wifi instructions are received, our robot will locate itself using our ultra-sonic localizer and the light localizer. From there, Entrypoint will instruct the robot on whether to attack or defend. This class calls on the Odometer, to keep track of the robot’s position, the Odometry Correction, and our threaded Navigation, before finally shooting. Since many variables have to be shared by many of our classes, we created a class called GlobalVariables to keep track of our values.

Our navigation module, responsible for getting the robot from a current location to a desired location, will naturally incorporate the required obstacle avoidance. This will be done by incorporating a Pathfinder as well as a Pilot which will be threaded in order to be interrupted upon detection of an obstacle. We will use stacks to layer the different coordinate sets that the robot should move to in the correct order, to go around the obstacles and finally reach a shooting position. Moreover, the robot will only move in the North, East, West, and South directions to facilitate the Odometry and mostly Odometry Correction classes.

Furthermore, we have developed interfaces for all our classes. This provides our team with great flexibility regarding development, by enabling substitutability between different implementations of certain classes. Our code repository is hosted on GitHub at <https://github.com/alisharif2/dpm-project> .

**II-Progress Log**

# February 16th 2017

* Project created on Github at <https://github.com/alisharif2/dpm-project>

# February 18th 2017

* Updated README file with guidelines for contributions, pull requests, and project description. Contributions and releases must be approved by the whole team before being made. We will use pull requests to merge branches into master branch.

# February 21st 2017

* New R&D branch created. This will be used to develop code that has not yet been tested or that is experimental.
* New version contains EntryPoint class from which code will start executing. This is the starting point of our code and calls on all our other modules.

# February 23rd 2017

* Updated README file with new classes and modules to be implemented
* Displayed modules and classes are now EntryPoint, GolbalDefinitions, Navigator, Odometer, OdometerCorrection, WifiReader and BallLauncher

# February 28th 2017

* Created classes to match description in README
* Reused classes from previous labs for Localization and Navigation
* Updated README again with better description of modules.

# March 10th 2017

* Created Interfaces for Odometer, Pilot, Pathfinder, OdometerCorrection, BallLauncher
* These interfaces will serve as guides to code the classes as well as make our design very flexible
* Updated GlobalDefinitions with Odometry constants

# March 11th 2017

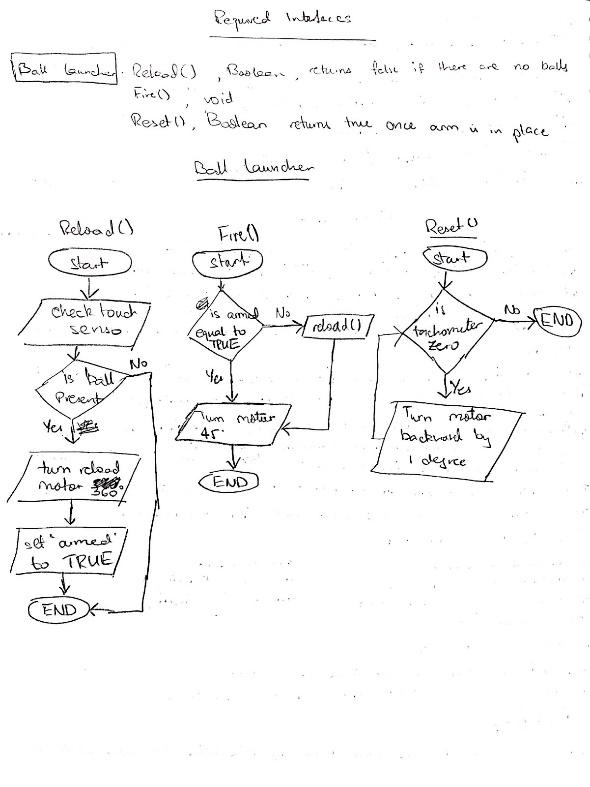
* Added motor and sensor initialization
* Replaced magic numbers with global definitions
* Sensor interfaces and implementation added -> still need to make into thread
* Changes made to EntryPoint with display implemented

# March 15th 2017

* Created Implementations of Pilot and Pathfinder modules
* Updated global constants

**III-Modules and classes**

* EntryPoint - Serves as the starting point of the program.
* GlobalDefinitions - Contains static final variables that will be accessible anywhere in the program. Constants like wheel radius, wheel base, light sensor offset, etc.
* Navigation - Provides high level way to move the robot around the field. It will provide coordinate and heading based movement. Will be created as a thread.
* Odometer - Provides internal tracking of the robot's position and heading using tachometers.
* OdomterCorrection - Complements the Odometer module by correcting it's position and heading using information from the light sensors.
* WifiConnection - Listens on WiFi to receive instructions for the competition.
* BallLauncer - Uses an unregulated motor to launch a ball. The launch structure is to be decided.



* USLocalizer - Orients the robot to zero initial heading according to its position in a corner.
* LightLocalizer - Calculates the robot's actual position using the black gridlines on the floor.

**IV- The Math behind our classes**

1. Odometry

The motors on the EV3 are equipped with tachometers. Tachometers are essentially counters which count the number of rotations made by each motor. With this information, we can compute an arc length of the robot’s displacement since its last position, knowing the circumference and spacing of the wheels. We then use the sinus and cosinus operators,respectively, to update the displacement along the x axis, the y axis, as well as the new heading of the robot. This class is threaded and these calculations therefore happen continuously and over short time intervals which keep the odometer precise.