# Scanner: A letter scanning program for leJOS EV3 Robot

Bishal Regmi

April 2016

## 1 Description of Classes

### 1.1 EV3 Classes

#### 1.1.1 Start.java

This class is the entry point of the program. It contains the 'main' function that initializes the menu. This class initializes the Robot class which handles the robot, its LCD screen to display the menu. This view part of this Main menu consists of two options:

- Calibration Menu
- Scan Menu

#### 1.1.2 Robot.java

This class contains the attributes related to controlling the robot. It is where all the sensors and motors are initialized. When this class is initialized from the Start class, the DifferentialPilot that controls the robot is initialized.

#### 1.1.3 CalibrationMenu.java

This is the menu where you can calibrate the robot to read the ON color, OFF color and the size of the pixel.

#### 1.1.4 ScanMenu.java

This is the menu where you can select between Forward Search or Backward Search on the board. If Backward Search is selected, all the available characters in the rule base are listed on the screen from which you can select the one to search for.

### 1.2 Other Java Classes

#### 1.2.1 Dictionary.java

This is the class where the file to get the rule base from fetched. It then send the InputStream for the file to syntaxReader class to decode the rules. Other than that, this class conducts the bacwardSearch and acts as a controller between forwardSearch and the Menu.

#### 1.2.2 ForwardSearch.java

This is the class which handles the algorith for Forward Search. It has two list passed and failed divides and creates the decision tree.

#### 1.2.3 Letter.java

This is the class to hold all the letters. Basically, this class holds a two dimensional array of the board size as boolean value and has getters and setters to set these values. It also has copyFrom and compareWith function which allow to copy the true values from teh given letter and compare the letter with the given letter respectively.

#### 1.2.4 syntaxReader.java

This is the class that decodes the serialized files and creates a list of rules.

#### 1.2.5 BoardAttributes.java

This is the class that store the dimension of the board.

#### 1.2.6 Menu.java

This class is an interface for all the menus. It has an LCD object and a function to display the cursor and a function to invoke the Menu.

## 2 AI Algorithms

### 2.1 1. Forward Search

This search is implemented in the Class **ForwardSearcher.java** and the method **public void divideTree(int x, int y, boolean pixelSet)** is the AI part of it. This method basically divides the list into two new lists based on the setting of the given pixel. If there are no deductions left at any point, it rolls back one node in the tree and continues scanning for possible candidates skipping the last pixel.

#### 2.1.1 2. Backward Search

This search is implemented in the class **Dictionary.java** and the method **public boolean backWardSearch(int x, int y)** The scan calls this function to check if the scanned row is On or not and based on its return value it scans and compares with the margin value to deduce failure or success.

## 3 Bug Report

- The program takes some time to start the calibration menu as it has to initialize all the utilities for the menu.
- The program throws null-pointer exception sometimes when it starts for no reason. It could be an issue with the robot.
- Other than that, I haven't encountered any bugs yet.

## 4 Features

### 4.1 Implemented

1. The menus are use friendly and easy to navigate.
2. Program shows the deduced values for calibration.

### 4.2 Not implemented

1. None

## 5 Observation Log

### 5.1 Forward Search

Letter **Q** was recognized and was success.
The Deductions were : **C,Q,O**

### 5.2 Backward Search

Letter **T** was success.
Letter **J** was failure.

## 6 Project Log

- April 26th (2 hrs)
  Worked on figuring out how to rotate and travel the robot. Differential Pilot had to be used
- April 27th(2.5 hrs)
  Spent the whole time calibrating the rotation and travel to make the movement precise. Had to scale the size of the wheels and distance between them precisely.
- April 2nd(4 hrs)
  Spend the whole day trying to load the serialized file and split the rules and save them. Create a new class letter with two dimensional array and created a list of those Letter to hold the rules.
- April 3rd (1.5 hr)
  Worked on the backward search as a Java Application. Works perfect.
- April 4th(4 hrs)
  Spent the whole day trying to calibrate the robot. Still can't calibrate precisely.
- April 5th (1.5 hours)
  Figured out that creating margins for values fixes issue. Calibration works.
- April 6th (3 hrs)
  Worked on the forward search as a Java Application. Works perfect.
- April 7th(5 hours)
  The file doesn't load in the robot. Figured out that I had to use getResources() to get the file. Loaded the file searches works. Was having issues with searches but minor debugging fixed issues
- April 8th (6 hours)
  Worked on creating the menus for the robot and testing the robot on various boards.

Total: 29.5 hours