Explorer Post 1010
Botball Team 16-0161
Greater DC Region

Period 2 Software Design

Introduction:

The code in review is for our link robot who will be driving up the ramp on our side, cleaning one of the solar arrays, coming back down, and then going through the middle to the other team's side and driving up their ramp. The robot's code was written by Elvin Liu and reviewed by Elvin and Aine Kenwood on March, 28th 2016.

Best Practices Checklist:

- [x] Code uses functions to organize it effectively
- [x] Code contains comments that document each function's purpose
- [x] Code contains comments that document each function's arguments
- [ ] Code contains comments that document each functions return values
- [x] Code contains variable names that are clearly labeled to properly convey their use in the program
- [ ] Code does not contain any unnamed numeric constants
- [x] Code is formatted to show flow of control
- [ ] Code does not contain any unused blocks of code

None of the functions implemented return any values, so there is no need for comments to be used to document them. Otherwise each function's intended purpose and parameters have been properly and thoroughly documented. Some of the unnamed constants are used multiple times throughout the program because using named constants would be unnecessarily clunky and difficult to understand. Due to this, we refrained from using named constants for those purposes. Other constants, such as motor and servo ports and open/close values have been defined as constants. There are some sections of commented out code that are not being used currently, however they have been left in the program in case we need to use them later and currently we don't have a better place to store them.

Reliability:

In order to drive up the ramp, we needed to find a way to line-follow, otherwise the bot would drift to the left and ram into the left PVC pipe repeatedly. However, the original design which was to use just one top-hat sensor was a little too slow and left the robot with not enough accumulated speed to be able to drive up the ramp. The original code implementation for this was to attempt to turn in place to find the line, since this is the most efficient possible solution. However, it was too slow so we tried the "arc path" solution which was to arc left on white values and arc right on black values. This, however, was also an ineffective solution because the bot would run into trouble if it ran too far over the tape. To remedy this, we attached another top-hat sensor to the front of the bot, and changed the code accordingly. The new version of the code bounces between the two lines with the two top hats (if the right goes over the tape, then it turns left, and vice-versa). This helps to increase accuracy with the line-following to make sure that we are able to drive up the whole ramp. The new solution also helps maintain speed since it is not stopping in the process. The reliability is good, but the speed should be increased a little bit more because the bot is still having a trouble fully scaling the ramp. We would like to test different speeds in small increasing increments to see if the ability of the robot to drive up the ramp can be improved. We will also be considering the effect a higher speed will have on the line-following accuracy (it may drift too much and mess up the sensors).

Maintainability:

Elvin is the main coder of the team, but our team believes that the entire team should be able to understand and make changes to the code if necessary. To assist in the maintainability of the code, Elvin has left a plethora of comments throughout the code to document different sections of movement for the robot and the purposes of certain lines, as well as TODOs and future goals. Each function's purpose, implementation, and parameters have been thoroughly documented to let team members know how to make changes. Older versions of code have also been document for logging purposes and in case someone wants to access it. To improve the maintainability of the code, using a repository online or having storage somewhere portable would help with documenting versions and logging. We would also like to start making full backups of code, in case a laptop blows up or some other catastrophic event occurs.

Effectiveness:

For starters, the code is able to perform its task relatively effectively. However, it is certainly lacking efficiency implementation wise. For example, there are blocks of code that perform a specific purpose within the main body of code, but it would look nicer and be easier to edit if they were moved to a separate function. For instance, here is the “servo spin” function that is used to clean the solar array:

```
// spin arm to hit the thing
set_servo_position(SPIN, SPINCLOSE); // should open
msleep(500);
set_servo_position(SPIN, SPINOUT); // should close
msleep(500);
msleep(500);

move(500, -500); // turn left to prepare to get off ramp
msleep(600);

move(500, 500); // go back down ramp
msleep(5000);

move(500, -500); // turn left again
msleep(600);
move(500, 500); // drive some forward
msleep(1000);
```

In this, the section of code to make the servo spin is nestled within some blocks of move functions and comments. This makes it fairly difficult to read and change. Below is a better version of this code.

```
// IMPLEMENTATION: there are two servos: one for main claw and one for "spin"
// parameter: PROBABLY NEEDS SOME jk
// SPIN TO WIN FT. NOT RNG
void servoSpin()
{
    set_servo_position(SPIN, SPINCLOSE);
    msleep(500);
    set_servo_position(SPIN, SPINOUT);
    msleep(500);
}
```

…

```
// line follow up the ramp to avoid crashing into the ramp
lineFollow();

// spin arm to hit the thing
servoSpin();
msleep(500);

move(500, -500); // turn left to prepare to get off ramp
msleep(600);

move(500, 500); // go back down ramp
msleep(5000);
```

In this improved version of the code, the previous servo-spin section of code has been moved to its own function, which allows it and the other section of code to be changed far more easily. It also allows for a generally more effective implementation. Superfluous comments have also been cut out because they tend to crowd up the program unnecessarily.