User Guide

Group 5 Team C

February 2017

Cole Zimmerman
Dainius Cerniauskas
Daniel Clemente
Guhan Sundaram
Ivaylo Genev
Kazimieras Onaitis
Tom Bock

The Fellowship of The Ball

Contents

Introduction

The following document will describe how to set up Frodo the robot by firstly going through all the physical setup that needs to be done (eg battery change) as well as the mechanisms of how it operates. It will also outline the software setup including the scripts needed to compile the repository and the calibration needed on the vision interface.

Installation Instructions

Installing The Java Code

There are two ways this guide presents as options to install the source Java code. The first installs the code directly using Git. The second method installs the code using IntelliJ (again through Git).

Direct Install

- Navigate to the directory where you want to store the code
- In a terminal, type "`git clone https://github.com/IVG1995/sdp_2017_fellowship`"
- In a terminal, type "`cd sdp_2017_fellowship`" to enter the newly created directory

Install Using IntelliJ

This install method provides the easiest way to modify the code as it will allow a user to easily open up the entire project using the IntelliJ editor.

- Open IntelliJ IDEA Community Edition (under "Applications → Programming")
- Click "File → New → Project from Version Control → GitHub"
- For "Git Repository URL", enter "`https://github.com/IVG1995/sdp_2017_fellowship.git`" (You may be prompted for a GitHub username and password. If you do not have an account, create one to continue)
- You'll need to do some configuration for your new project now:
  - Click "File → Project Structure"
  - For "Project SDK", click the "New" button, then "JDK", then choose "java-1.8.0"
  - Set the project language level to 1.8
  - Under "Modules", right click the "src" folder and mark it "Sources"
  - Under "Modules" still, click the "Paths" tab and check "Use module compile output path"

  - Under “Libraries”, click the green arrow in the upper-left corner, then “Java”, then click on the “libs” folder and hit OK
  - Apply the changes
- Note: running the code from IntelliJ will cause the code to function incorrectly. For full functionality, run the bash script in the project folder (see “Running the Code...” below)

Installing The Firmware

This section discusses the installation of the firmware necessary for the running of the Arduino.

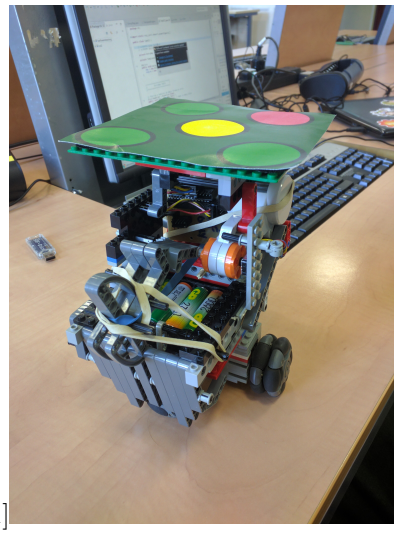
- Navigate to the directory where you want to store the code
- In a terminal, type “`git clone https://github.com/IVG1995/sdp_2017_fellowship_firmware`”
- In a terminal, type “`cd sdp_2017_fellowship_firmware`” to enter the newly created directory
- Fire up the arduino IDE by typing “arduino” in a terminal
- Open the Firmware2.ino project in the newly opened arduino IDE
- Connect the arduino to the computer via the USB cable
- Press “Upload” in the arduino IDE
- If a problem occurs, in the arduino IDE go to:” Tools -¿ Board “ and make sure that Arduino Uno is selected and then in “Tools -¿ Serial Port” is active and points to the name of the USB port (ttyACM0 or ttyACM1 on DICE machines). Unplug and replug the USB cable if it doesn’t show up

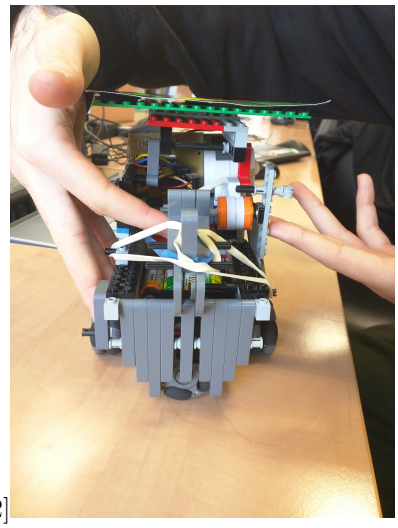
Physical Setup

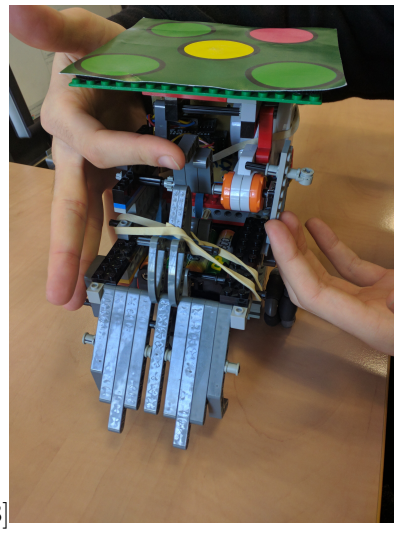
This section deals with the physical aspects of the robot setup

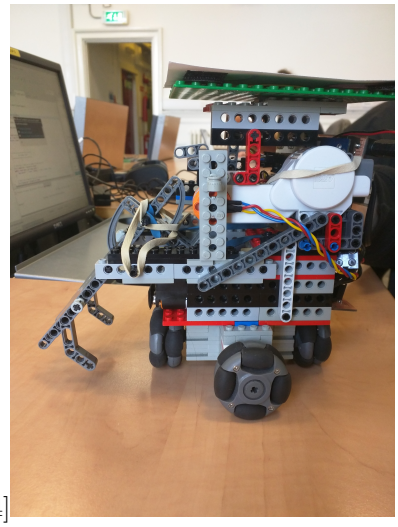
Battery Change

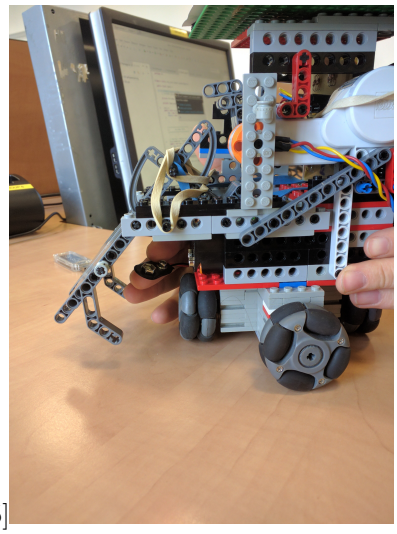
This part first requires that you have a loaded set of batteries to replace the current ones.

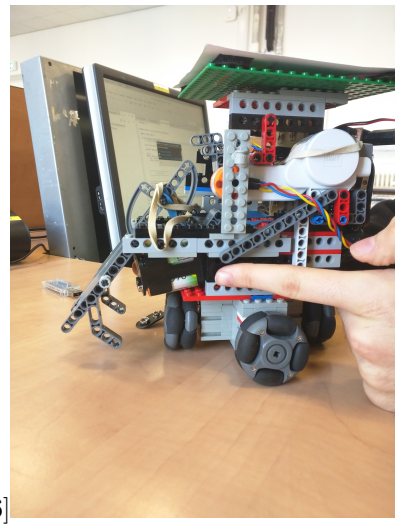
[1] 

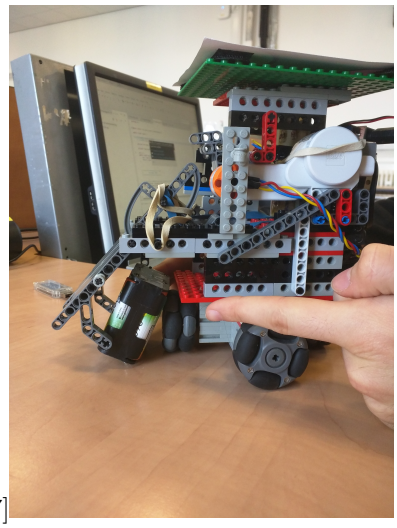
[2] 

[3] 

[4] 

[5] 

[6] 

[7] 

Turning The Arduino On

Here is discussed the remainder of the setup needed to get the arduino ready to be used.

- You should have a blue USB stick marked "SRF-Stick". This is how the code running on the computer communicates with the robot, itself. Ensure this is plugged into the beige cable hanging off the right side of the DICE machine you're using

- Attach the plate to the flat red Lego piece on top of the robot. Each robot plate consists of 5 spots:
  - In the center, a blue or yellow spot (denoting your team)
  - Outer spots: three are green and one is pink (or vice versa) (which distinguishes your robot from your teammate robot)

  Position the plate so the different-colored outer spot is at the back-left of the robot

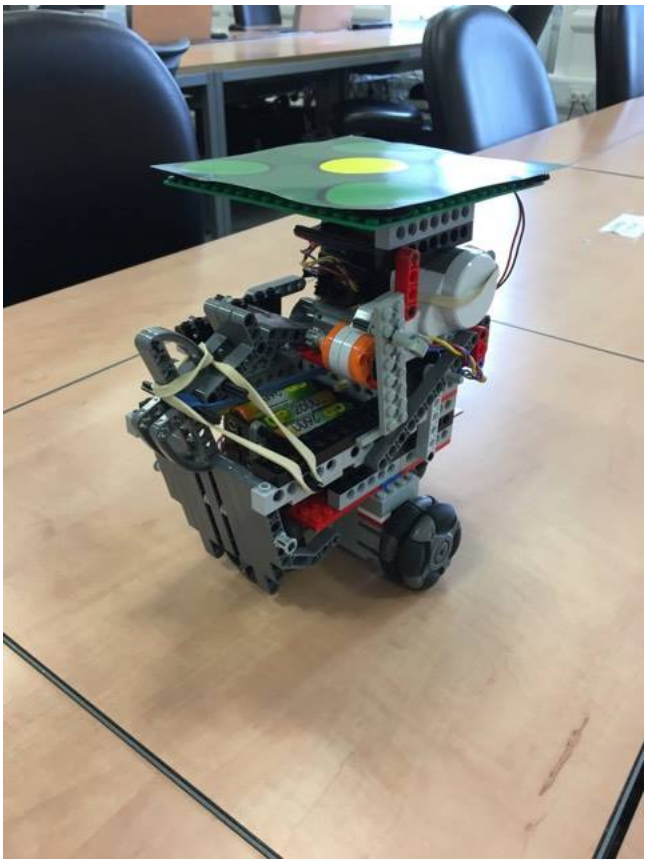


Figure 1: This is what the robot should look like after all the setup

Running The Code

In the project folder, open a terminal. Type "bash start.sh" to run the code. You should see a window pop up with a number of tabs, from which you'll choose your strategy, calibrate the vision, and more.

Communication Setup

To ensure that the robot and the Java code are communicating effectively, the Java program repeatedly sends a number of "ping" messages to the robot. Once the robot responds with a "pang", the program knows the robot is receiving its commands. You can see this process in the "Console" tab: once the "pang" response is received, you'll see a message along the lines of "Breaking news: a robot responding with pang has been found on the port ...". Make sure that you see this message before moving on with the rest of the setup.

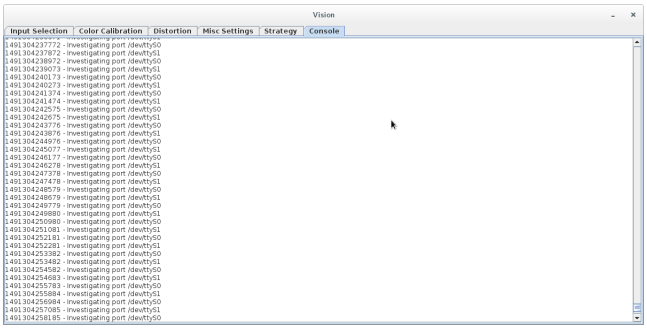


Figure 2: This is what the SDP Console should look like

Vision Setup

Our vision system uses background subtraction, which takes pictures of the empty pitch and subtracts their average from subsequent vision data to reduce noise. Because of this, it is crucial that when you start the vision feed (from the "Input Selection" tab), the pitch must be completely clear. If this isn't done, you may experience some unpredictable behavior running the robot.

- If you accidentally start the feed with anything on the pitch, you can always just remove everything, then hit "Restart Training" in the "Misc Settings" tab.

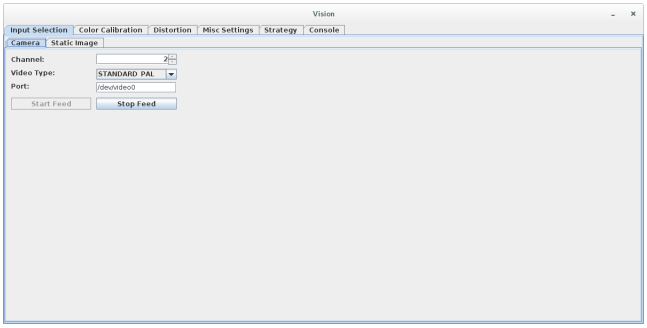


Figure 3: Picture of Vision GUI Here

Now click on the "Color Calibration" tab. Robots are identified by their plates, each of which has a unique array of different-colored dots on it. In order for this to work, you need to identify each color to the vision system. Here's a run-through on calibrating one color:

- Click on "PINK", then click "Calibrate"
- A window will pop up showing the vision feed. It also contains sliders for things like Min/Max Hue/Saturation, etc: you can mess with these if you'd like, but this is not recommended
- Instead, take a look at the feed. Click on any pink spot you see. Upon doing so, all visible pink spots should turn white. If not every pink spot is white, click on various pink spots in different locations to see if they will do better (You might not be able to get a perfect calibration, so just try to get as much pink covered as possible)
- Remember to repeat for all other colors, and the procedure for calibrating the ball is the same

You might notice there are 5 green colors: "GREEN", "`GREEN_0`", "`GREEN_1`", and so on. These have been added to help cope with the vision system's particular finickiness with regards to green colors. You are initially recommended to ignore these extra colors and only use "GREEN", but if you're having some vision issues, do as follows:

- Calibrate "GREEN" normally
- Calibrate "`GREEN_0`" using a green dot from the top-left corner of the feed. Calibrate "`GREEN_1`" using a dot from the top-right, and so on for each corner

This effectively allows the vision system to have a green calibration for each region of the pitch, making it more robust to lighting differences and other potentially disruptive conditions.

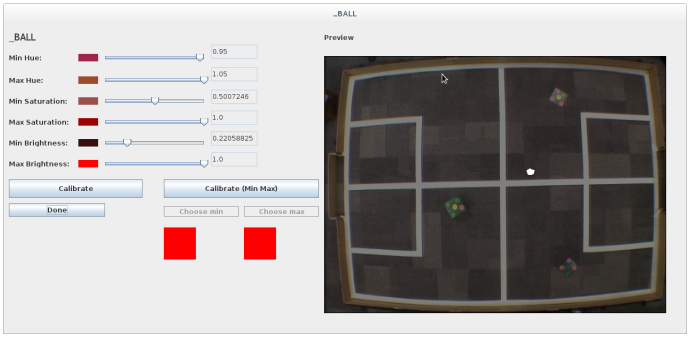


Figure 4: Individual Colour Calibration

Our system allows you to save color calibration settings so you don't have to recalibrate every time you start the system. To do this, go to the "Misc settings" tab and click on "Save settings". The name of the file and where it's stored is up to you.

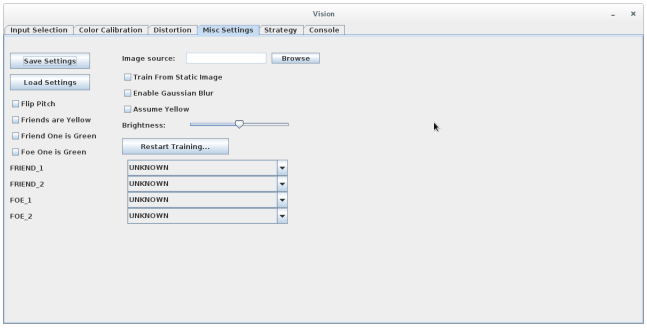


Figure 5: The Misc Settings Tab

Before we move any further with the setup, take a look at the window marked "Robot Preview" in the Color Calibration tab. This displays (more or less) the end result of the vision pipeline, and is your most crucial tool to ensuring everything is running as expected.

- The background should be black, with white lines denoting boundaries and half-field lines. You should be able to see the spots from each robot's plates on it, plus each robot should be delineated by a white rectangle and marked with a name (i.e. "`FRIEND_2`", "`FOE_1`"). The ball should be represented as a red circle, and lastly, if the ball is particularly close to any robot, that robot should also have a red rectangle around it, denoting it as the ball holder. Compare the pitch to the preview to make sure these things are true.

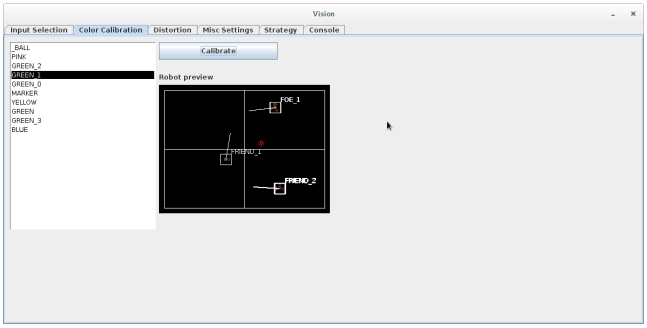


Figure 6: The Colour Calibration Tab with the Robot Preview

Now that you know about the Robot Preview, you need to tell the system which plates correspond to which robots, as well as which team is defending which side. It is very important that our robot is represented as "`FRIEND_2`", and that our ally is shown as "`FRIEND_1`". After following these steps, this should be true:

- Look at the color of your robot's plate's center dot. If it's yellow, tick the boxes "Friends are Yellow" and "Assume Yellow". This lets the system know you're team yellow
- Next, look at the color of your teammate robot's single outer dot (not the color of the other three dots). If it's green, tick the box "Friend One is Green". Else, our robot's single outer dot must be green, so leave it unticked
- If you're playing a 2v2 match, you can ignore the "Friend One is Green" box. If you're playing a 1v1, look at your opponent robot's single outer dot, and if it's green, tick the box

Look in the "Robot Preview" window again. In this window, your half always appears on the left-hand side of the preview. Set your robot down in your half. If, in the preview, your robot appears on the right, tick "Flip pitch".

Look back at the Robot Preview again and ensure everything's in order before continuing.

Strategy Setup

Go to the "Strategy" tab. Two of these text fields are important to you.

'Action:' displays what action the robot is currently performing.
A command box, through which you can tell the robot to perform certain actions.

Frodo can perform a number of actions, some of which are relatively low-level

routines (i.e. “OffensiveKick” and “DefendGoal”). The others effectively act as the Frodo’s brains during the match, dynamically evaluating the situation and deciding which lower-level actions to execute when. Our two “brainy” actions are called “Behave” and “MainOffense”.

- Behave is our defensive action. With this, Frodo similarly to a goalkeeper in many situations, staying in between enemies and our goal, blocking shots, pushing the ball away from our goal, and (when the coast is clear) making offensive kicks himself
- MainOffense is Frodo’s offensive mode. With this, he’ll only defend when it’s absolutely necessary, opting instead to aggressively attack the enemy goal and get in the enemy’s way as much as possible, leaving the bulk of the defending to his teammate

We’ve listed a number of actions (plus brief descriptions and what to type into the command box to execute them) in an appendix below. Use it to try out each of Frodo’s actions, or just type ‘b’ for Behave, or ‘o’ for MainOffense.

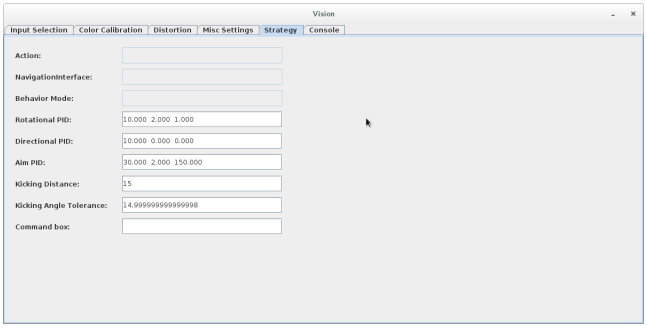


Figure 7: This is the strategy tab in the vision GUI

Structural Information

Kicker

The kicker is a relatively simple design using the tension of several rubber bands to snap forward the kicker and hit the ball. The kicker is moved back into pre-kick position using a motor that pushes the top of the kicker back and increases tension in the bands. This allows the robot to achieve a powerful kick without the need for a lot of power other than the power to return the kicker into rest position.

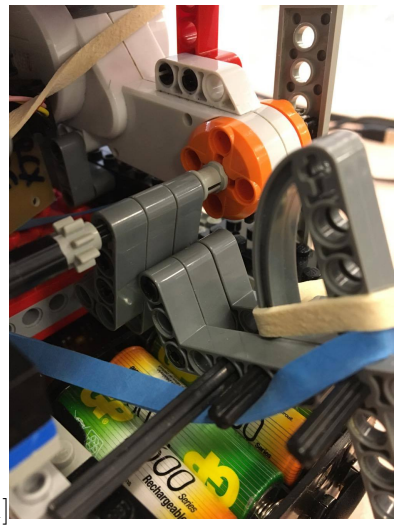
[1]  [2] 

Figure 8: Picture 1 is the kicker in rest position and 2 is the kicker during the kick

Holonmic Motion Control

The robot uses Holonomic Wheels as its motion system. The wheels are mounted on the forward, back, left and right directions of the robot. Using these wheels allows the robot to travel in all directions easily thus eliminating the need for turning to move in certain directions.

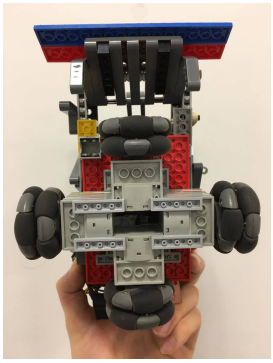


Figure 9: Holonomic Motion Control Wheels

Troubleshooting

The robot won't pang/the SDP console is constantly printing "Investigating port /dev/ttys0"	• Make sure the batteries are charged, the arduino is plugged in and the RF stick is plugged into the computer • If all of this is done and you're still having this problem, try plugging the RF stick into a different USB port • If this also doesn't work, stop the program and run it again, or try running it on a different machine • If you've done all of this and the robot still isn't responding, your RF stick may be broken. You'll have to find a replacement to use the robot
All of the previews are blank	Make sure to start the vision feed (in the "Input Selection" tab)
The vision feed has been started, but I'm not seeing any robots or the ball in the robot preview	Make sure to do the colour calibration; without it, the vision system won't be able see anything
The vision feed has been started and the robots and ball are on the pitch, but nothing is showing up in the preview window	Our vision system uses background subtraction, which takes a picture of the empty pitch and subtracts it from all subsequent vision data to reduce noise. Make sure that when you start the vision feed, all robots (and the ball) are off the pitch; otherwise, whatever's on the pitch will be permanently subtracted

I'm sure I've set up the vision correctly, but (in the Robot Preview) some robots and/or the ball keep getting lost, or they flicker a lot	Try to re-do the color calibration and see if you get better results. Often, calibration settings you made a while ago won't be good now (i.e. the lighting may have changed, or the overhead camera may have been tweaked). Our vision uses a sophisticated neural network aided object recognition system that will fill in a lot of the blanks, but unfortunately there's only so much it can do if conditions are particularly terrible

Appendix A: Command Box Actions

- Behave (type '**b**'): see description in section Strategy Setup
- MainOffense (type '**o**'): see description in section Strategy Setup
- DefendGoal (type '**d**'): Frodo positions himself in between the ball and/or enemies and the friendly goal to block shots
- OffensiveKick (type '**k**'): Frodo lines himself up for a kick toward the enemy goal, approaches the ball and kicks
- GoToSafe (type '**s**'): This action has Frodo run back in front of the friendly goal, and take care not to own-goal in the process
- GoToBall (type '**g**'): Frodo runs to the ball
- Annoy (type '**a**'): Frodo positions himself directly in front of the enemy ball holder, blocking its passes, shots and much of its movement
- BlockPass (type '**p**'): Frodo positions himself in between the two enemies, blocking any potential pass
- WallKick (type '**w**'): A special routine for dealing with situations where the ball is close to the wall (meant primarily to avoid Frodo slamming into walls and getting stuck)

Appendix B: PID Calibration

The PID calibration is not easy and requires background knowledge in PID. As such it is not recommended to change these settings unless absolutely necessary.

Depending on the motors you are using, the constants for the PID controls will change, as different values correspond to different motor powers. You can

change the PID constants by editing either the Rotational PID or the Directional PID. Since the robot uses a holonomic wheel system to move through space, it can turn whilst moving, which calls for tuning the rotation independently of the motion. You can employ any way of calibration you wish for, the system just takes a String of 3 floating point numbers, separated by whitespace characters, so if you want your PID constants to be 30.5, 1.8, and 0.9, you would put "30.5 1.8 0.9" in the text field in the Strategy GUI. Word to the wise: try to tune motion and rotation PID in match-like environment, so that it produces predictable results.