Team 3176 White Paper

Title: Arcade Drive	Authors: Jacob Aldridge, Sam Cordry, Elijah Furuness, Stuart Goedde, Markale Johnson, Ben McCarty, Anna Nierzwick, Kyle Phillips, Martin Wilson
Subteam: Programming	Subject: Drive System
Season: 2018/19	White Paper Number 1

Table of Contents

Abstract

This paper documents the process and proof for using Arcade Drive for robots. The paper is oriented for the general FIRST community interested in learning the basics of programming an Arcade Drive for movement controls. The goal of the algorithm is to allow the joysticks to direct the robot's movement through the X- and Y- axes, as well as the magnitude.

Definitions

Arcade Drive - Refers to a method of steering via joysticks, in which one controls magnitude and the other direction. Which joystick controls which is up to driver preference, and is a rather controversial subject.

Joystick - Refers to a method of inputting directives to the robot to control its direction or magnitude. It relays this information to the robot through its USB connection to the computer directing the robot.

Process

Research

When we need information about a topic, we first ask mentors or veteran team members for their knowledge to help us. If they don't know, we use the Internet to find resources to help us with what we need to do. We try to find FRC sources when browsing the Internet for help.

Actual Coding

PC Setup

To set up your PC you should have a few applications. You need a development environment such as Eclipse, Microsoft Visual Studio, etc. The language that will be used in this white paper is Java. We also use WPILib which is a digital library that allows us to use FRC code.

Inverse Kinematics

Inverse Kinematics is a style of thinking where you create a table of possible inputs and desired outputs and figure out what code is needed to get from input to output. Usually this involves doing quite a bit of math. A quick way to check the equation is to use Excel (see Excel section).

Coding

Test Driven Development (TDD) is a process of trial and error that can be used with Inverse Kinematics. TDD is just coding something and seeing if it works in practice. In Arcade Drive, you would need to do several steps to get it to work. First, you need to make your robot an instance of the Iterative Robot class (found in WPILib). From there you need to define and instantiate your motors, joysticks, and any other variables you might use. Finally, you code whatever algorithm that you gott from doing Inverse Kinematics.

Issues

There was some problems with the logical thinking, so we had to rethink our algorithm and fix it. We also had code problems so we had to test code frequently and in different parts so if we knew if there were problems in sections of the code.

We had joystick problems because they were returning weird values (see Problems and Solutions). There also seemed to be a problem with the RoboRIO, so we did not have a robot to test the code on during some of offseason.

Arcade Drive Coding

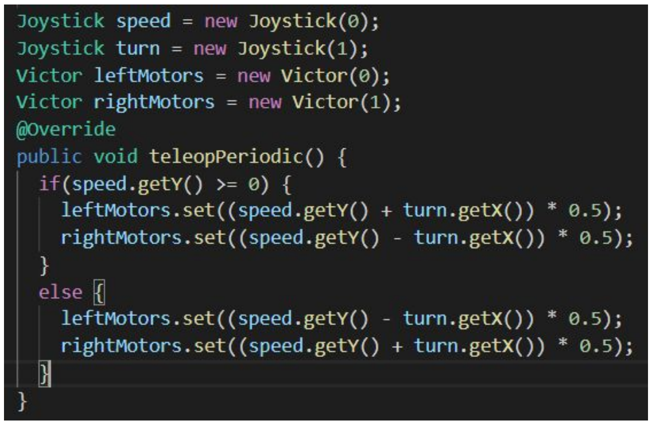
Variables

Victors are the physical motor controllers for a robot. The logical way to name Victors is with camel case. An example of a possible name of a Victor using camel case would be leftVictor. Make sure you always double or even triple check your port numbers.

Joysticks are a way to give instructions to a robot. You should always logically name your Joystick variables with camel case as well. And again you should double check your port numbers to match the USB ports.

Algorithm

Our algorithm is to make the robot drive in Arcade Drive. To get the motors to work properly, you have to make it so that the motors are set for speed plus or minus turn, depending on several factors. In Arcade Drive, you need to set the right motors on the robot to speed plus turn and the left motors to speed minus turn. Also, if the speed is negative, you would need to switch the signs for these equations. If you do not comprehend, look at the code below and apply Inverse Kinematics.

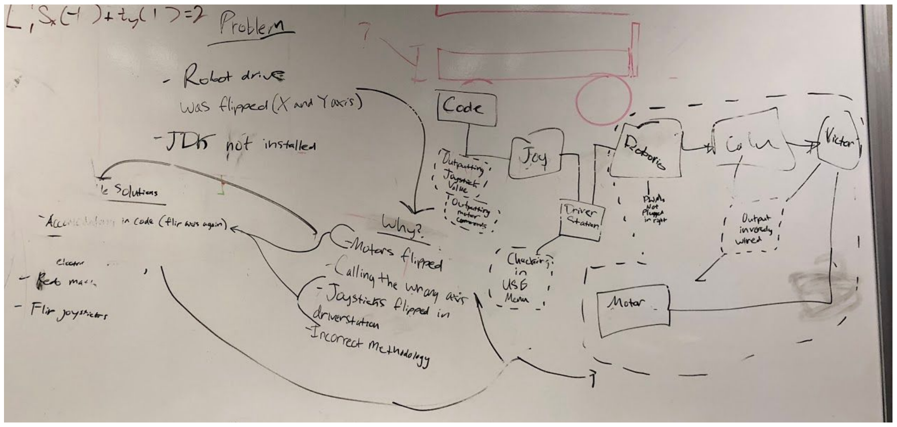


```java
Joystick speed = new Joystick(0);
Joystick turn = new Joystick(1);
Victor leftMotors = new Victor(0);
Victor rightMotors = new Victor(1);
@Override
public void teleopPeriodic() {
  if(speed.getY() >= 0) {
    leftMotors.set((speed.getY() + turn.getX()) * 0.5);
    rightMotors.set((speed.getY() - turn.getX()) * 0.5);
  }
  else {
    leftMotors.set((speed.getY() - turn.getX()) * 0.5);
    rightMotors.set((speed.getY() + turn.getX()) * 0.5);
  }
}
```

Testing

It was difficult to test when we didn't have access to a working robot, but when we did we had a substantial issues. We learned from each test and tried to figure out what was wrong with our code. We isolated sections at a time to by testing out either forward, backwards, and turning. When we were testing such functions, we took mental notes of what was going well and what was going wrong.

Problems and Solutions



One of the problems we came across when programming Arcade Drive was the fact that the code was reading the Y-Axis when we were calling the X-Axis and vice versa. In our Arcade Drive, velocity should the Y-Axis and speed should by the X-Axis. As you can see, this wildly changed what the robot did. The best solution that we came up with is to reverse what axes we call, so that it reverses again. Though this does not logically make sense, it does in practice.

Another problem we ran into was that the Y-Axis was being inverted. This means that the joystick, when being pushed forward, returns a value of -1. When being pulled backward, it returns 1. We believe that the joysticks could have been flight joysticks, which would explain this anomaly. This could also be explained by a mechanical issue. Our solution to this problem was to multiply every time that we wanted to call the Y-Axis by -1.

Efficient Problem Solving

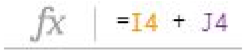
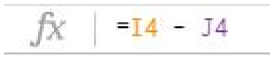
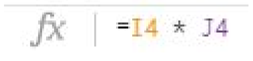
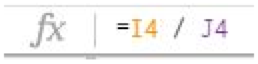
To problem solve efficiently, we recommend using the following tips:

- Consider everything that could be creating problems so you don't get stuck on something that isn't causing a problem
- Do not fixate on one thing that you believe is creating the issue; try to be open minded
- Narrow down your potential problem makers to a few components of the robot before going in depth

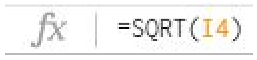
Excel is also a very handy tool when checking equations. You can make a table to quickly check equations by inputting desired inputs into the equation.

Excel

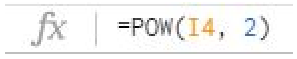
When doing math in Excel, you always have to put a "=". To get a value from a cell, use the column letter and row number, like "A5". To do basic operations, use the symbols common for those operations, examples are below.

- Add: 
  =I4 + J4
- Subtract: 
  =I4 - J4
- Multiply: 
  =I4 * J4
- Divide: 
  =I4 / J4

To take the square root of the value of a cell, use the syntax `SQRT(cell#)`. The example below would return the square root of cell I4.

- Square Root: 
  =SQRT(I4)

To raise the value of a cell to a power, use the syntax `POW(cell#, power)`. The example below would return I4 squared.

- Exponents: 
  =POW(I4, 2)

To get the row number of the cell you are working on, use the syntax ROW(cell#). ROW(D14) would return 14.

To round a number, use the syntax `ROUND(cell#, digits after decimal)`. `ROUND(C6, 3)` would return cell C6 rounded to 3 decimal places.

To select multiple cells, use the syntax `cell1:cell2`. You can use this to select cells or a column. You can also use this to select two corners and Excel will find the cells in that box.

To take the sum of a group of cells, use the syntax `SUM(cell1:cell2)`.
To find the average of a group of cells, use the syntax `AVERAGE(cell1:cell2)`.
(Note: both `SUM()` and `AVERAGE()` must have multiple cells specified.)

To quickly copy an equation across multiple cells, click on the cell that holds the equation you wish to copy. Then a blue square will appear in the lower right corner of the cell. Drag this around; the equation will copy to any box within the box you created.

A detail to keep in mind is that Excel will automatically change any cell names specified in the equation. This means that any cell that you want to be a constant in the equation must be typed in by hand.

You may start getting really complicated with equations in Excel. Luckily, Excel supports the use of parentheses just as you would use them in a math equation. Make sure to use these to your advantage.