In and Outs of Programming for FRC

Daniel Volya

1 Introduction

Lets assume that you, or your team, has gone through: 1) the robot desinging phase, perhaps with the help of Computer Aided Design (CAD) tools such as SolidWorks; 2) the manufactoring and building phase; and 3) the electrical equipment phase. And now you want to see the robot follow commands and to be controlled and guided. This is where programming comes in, and the primary focus of this document. I will guide you on the installation and preperation steps for FRC, provide examples and exersices for basic functionality, and potentially introduce slightly advanced topics that are important in programming and controlling your robot.

2 Backbones

First, however, I'd like to reveal some important tools and notions that will provide help in preparing and programming for FRC.

2.1 GitHub

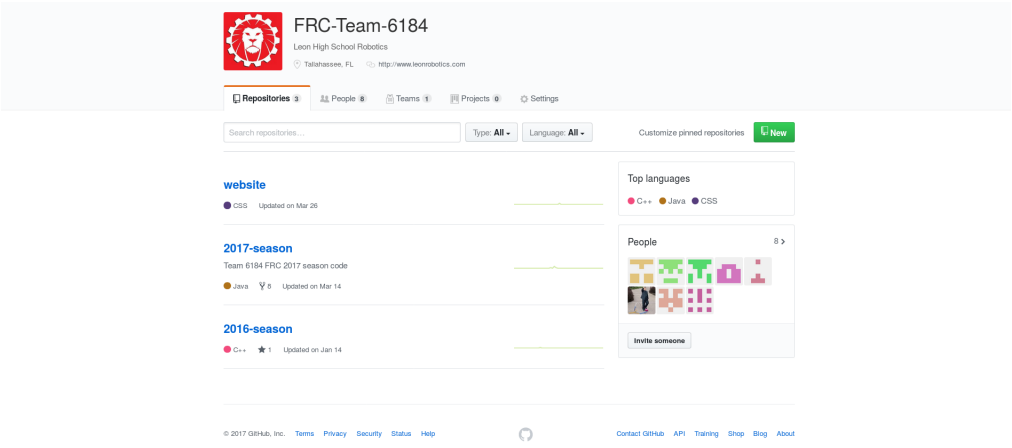

Figure 1: Leon Robotics GitHub page

GitHub is based on a popular tool known as Git, which was first written by the inventor of Linux (Linus Torvalds). Git, along with GitHub provides a way to share and version control software. This allows for various team members to stay up to date with code, and not have to rely on one team member to store all the code on their computer. Of course, as with all tools, GitHub must be used properly to ensure effectiveness. I would highly recommend learning how to use GitHub, Git, and version control in general – this is an important skill. Moreover, many teams in FRC have their own GitHub page, so if you ever need some inspiration or help, I would recommend searching for other teams' GitHub pages.

2.2 Essential Websites

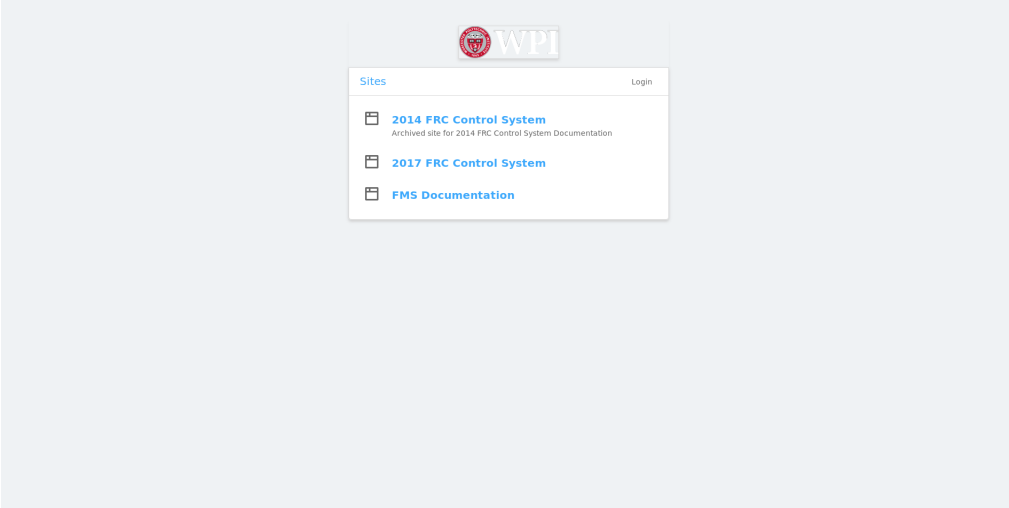

Figure 2: https://wpilib.screenstepslive.com/ – Worcester Polytechnic Institute (WPI) is a major contributer to the software components in FRC. They have made excellent progress in providing installation and basic tutorials for new software that was introduced a few years ago. If you ever run into trouble of setting things up, this is the place to checkout. They also provide the application programming interface (API) documentation that is used in programming robots for FRC – which you will visit quite often.

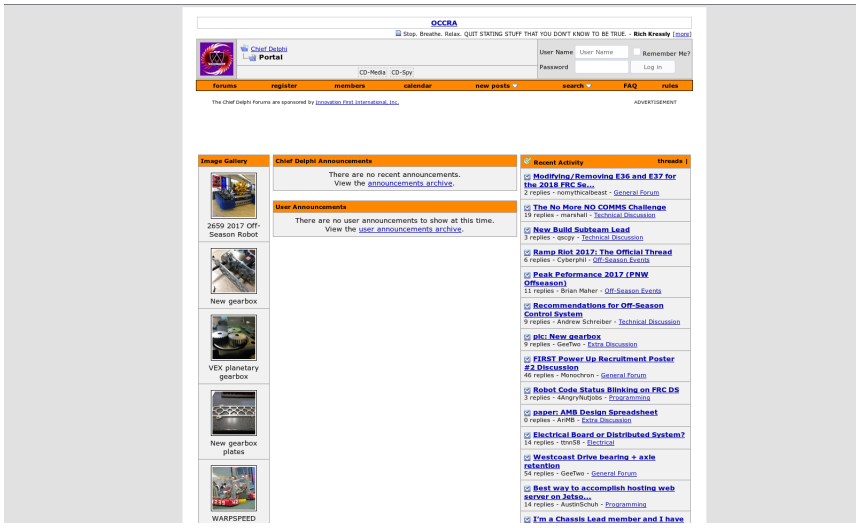

Figure 3: https://www.chiefdelphi.com/ – This is a discussion forum dedicated to FRC. You can find information and seek help from other team members and mentors about a number of topics including programming.

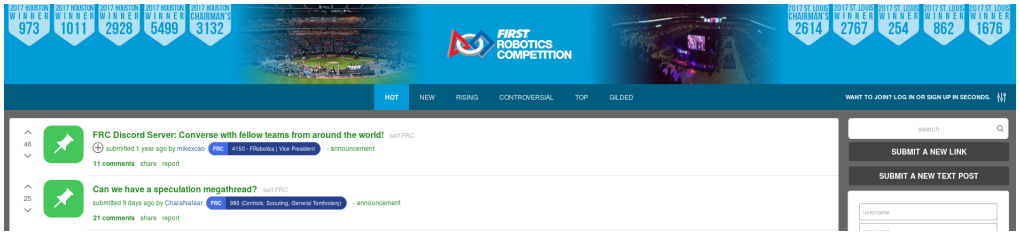

Figure 4: https://www.reddit.com/r/FRC/ – Similar to Chief Delphi, the FRC subreddit is filled by other team members and mentors. I have found rather specific solutions to software problems here. Moreover, this is a convienent place to see people's opnions and thoughts about certains elements, designs, and games.

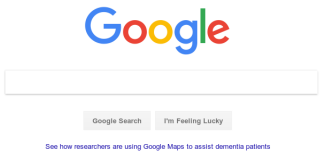

Figure 5: https://www.google.com – This goes without saying, but Google is useful in quickly finding solutions. Topics such as C++, Java, Eclipse, OS, NI, Robotics, and FRC, can be quickly found. If you're concerned about privacy, use something else like duckduckgo.com

2.3 Mentors

If you ever need help or advise, do not hesistate to seek help from mentors. We are thrilled to help students in learning and innovating in the challenging field of robotics (we also like learning cool things too.) On this topic, I am a University Student who is studying Computer Science and is a member of a team that builds autonomous vehicles/robots (https://mil.ufl.edu/). Although I may not be an expert in all the areas of robotics, I can certaintly provide direction, guidances, and solutions, and I do like robots – so please feel free to contact me.

2.4 Philosphy

Perhaps the most important part in robotics, and in other fields in general, is having the correct philosphy. I am not suggesting that there is one correct philosphy to follow, but I think it's important to mention praticular traits that I have found to be important. 1) be proactive – this is a rather abstract trait, but satisfying it is essential for success. This ranges from self-motivation, teamwork, seeking guidance, finding solutions, learning, etc. 2) Don't let failure stop you – there are many ways to think of this, but engineers and designers face this every day. Use failure to your advantage – learn from it, and progress.

3 Programming

Before going into the details of setting up for FRC, let us go over some general programming concepts. It is highly recommend to go over these topics in further detail.

3.1 A Feel For Programming

Let's remember some basic algebra. Suppose Alice, who has a basket with two apples, adds another apple to her basket. How would you express this mathematically?

$$x = 2 + 1$$

What about in a program?

```
int x = 2 + 1;
```

The 'int' indicates that the number x is an integer (such as: -7321, -3, 0, 2, 3, 5, 9999). The semicolon ';' tells the computer that this is the end of your deceleration/assignment.

We can take this a little further:

```
int x = 2;
x = x + 1;
```

What do you think the value of x is at the end? If you say 3, you are correct. We can analogously follow Alice's scenerio. Alice started with 2 apples in her basket (int x = 2;) and then added another apple (x = x + 1;). Often, when it comes to adding 1 to an integer, programmers shortern 'x = x + 1;' to 'x++;'. These mean the same thing.

3.2 Functions

Just like in math, we can define a function such: $y = 2x + 3$. This is a line with slope 2 and y-ntercept of 3. We can also define functions in programming:

```
float y (double x)
{
        return 2*x + 3;
}
```

Here, we define a function named 'y' that takes in a value 'x' and returns $y = 2x + 3$. Notice the 'float' primative, this tells the computer that the function 'y' will return a decimal-like number typically with 7 digits of accuracy (such as: 0.3, 1.5, 3e-7, 300.123). The primative type 'double' is a higher precision of 'float', meaning we can have more decimal places represented. The computer, typically, can automatically convert types to fit it's needs. Such as, in this case, the double is being converted to a float when we return $2x + 3$

Here are some examples of functions:

```
// Find volume of a right cylinder
double cylinder_volume (double radius, double height)
{
        return 3.1415 * radius * radius * height;
}

// Find the sum
int sum (int n)
{
        if (n == 0)
                return 0;
```

```
        else
        {
                return n + sum (n - 1);
        }
}
```

The last function finds the sum ($\sum_{k=1}^{n} k = 1 + 2 + 3 + ... + n - 1 + n$) by calling itself. This is known as recursion, which is useful for solving some problems, but is rarely used.

3.3 Classes

Probably the most the important part in understanding programming for FRC, is knowing how to use Classes. Classes are essentially abstractions of something we wish to represent, such as car, a human, a motor, or a camera. Classes are a branch of what is called Object Orientated Programming, where we represent things as abstract "objects", which makes the life of a programmer easier.

Let us look at some FRC Java code from 2017:

```
Talon mR = new Talon(0), mL = new Talon(1), shoot = new Talon(3), climb = new Talon(4);
```

In FRC, the most common motor controllers are: Jaguars, Victors, and Talons. The motor controller accepta pulse-width modulated (PWM) signal, which then controls the speed of a motor.

In our code, we use a class called 'Talon', which was programmed by WPI to provide an easy way to control our Talon Motor Controllers. Here, we defined a "motor right" (mR), and provided an argument of which port the motor connected to on the roboRIO (which is 0 here.) Simiarly, we defined the shooter motor (shoot) on port 3. We can then later set the speed of a given motor such as:

```
// Set the right motor to spin wheel backwards at 80%
mR.setSpeed(-0.8);
// Set the shooter motor to spin forwards as 50%
shoot.setSpeed(0.5);
```

3.3.1 Timer Class

Useful to store time.

```
Timer my_timer = new Timer();
shoot.setSpeed(0.5); // Spinner shooter motor
my_timer.delay(7.5); // Wait 7.5 seconds
shoot.setSpeed(0); // Turn off shooter motor
my_timer.reset(); // Set the timer back to 0
```

3.3.2 Joystick Class

Used to get input from a joystick connected to the computer.

```
Joystick stick = new Joystick(0); // set to ID 0
double y = stick.getY(); // how much is the joystick tilted in the y direction
if (stick.getRawButton(11)) { // check if a button is pressed
        climb.setSpeed(1); // spin the climber motor
}
```

We can use the DriverStation App to see a visual representation of whether there is a joystick connected (and to what ID it is associated with), along with what the axis are and what buttons are pressed.

3.4 Exercises

1. Write a function called 'fire' that will spin the shooter for 5 seconds.
2. Change the "motor right" (mR) object (from the Talon example) to listen to PWM port 7
3. Make a new Talon motor object called 'arm' assigned to PWM port 5
4. Make the new 'arm' motor spin at one-tength speed of the current y position of the joystick
5. Assuming a drivetrain configuration, with 2 regular wheels being connected to a motor on both sides, how would you make the robot rotate? What values should you put into mR.setSpeed() and mL.setSpeed()?