Rolls-Royce

GETTING STARTED GUIDE

WITH

RACE YOUR CODE VIRTUAL

FormulaPi.com

The below is insight from the people who create and build the robots you'll be racing on how the challenge works and the code you will receive.

THE CHALLENGE

Thanks for taking part in the Rolls-Royce Race Your Code virtual racing. The racing is being hosted by Formula Pi in Cambridge, in the UK.

In this competition, you will edit and write code which will be used to control virtual robots. These robotswill compete against each other in 4 virtual races and one real world race.

To do this, you take pre-programmed robot code which can navigate the racetrack on it's own, improve the code and submit it to the Formula Pi server to race against other competitors on virtual tracks. This could be tricky as your only sensor input is from a camera mounted at the front of the robot, but improvements can be as simple as changing the lane on the track, or as complicated as major changes, such as a complete code rewrite.

The race dates are as follows:

16th January 2019	Virtual Test round 1
23rd January 2019	Virtual Race round 1
20th February 2019	Virtual Race round 2
27th February 2019	Virtual Race round 3
1st March 2019	Real world race on Formula Pi track (Race round 5)

Watch the Live stream

All races are live streamed on YouTube so you can watch the virtual racing live, or watch it back at any time after the event.

There are 4 virtual tracks and one real world track. These are in order:

Silverstone, UK
Cota, USA
Hockenheim, Germany
Marina Bay, Singapore

The real track is the Formula Pi circuit in Cambridge, UK.

Code can be improved in various ways. You could for example vary the lane choice the robot tries to navigate around, but be careful – the robots move in unexpected ways!

STEPS FOR COMPETITORS

The laptop you are using has been specially configured to simplify the process of editing, testing, uploading code, and viewing the live stream.

You will need to connect the laptop to a Wi-Fi connection in order to upload your code and to view the stream. We recommend using the guest Wi-Fi or an external Internet connection.

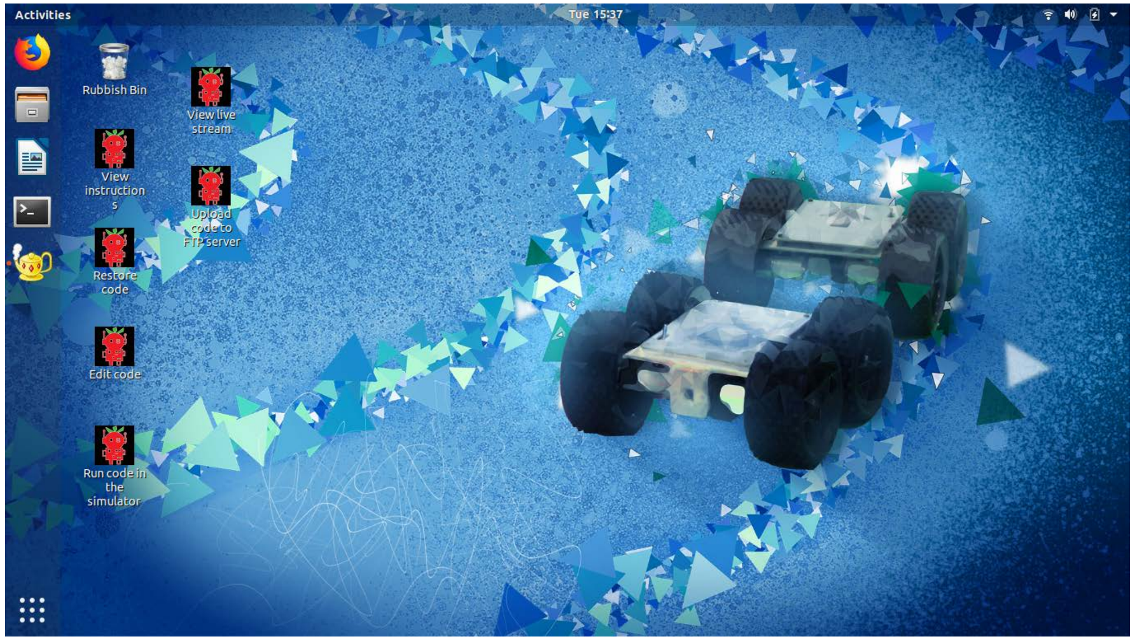
Use the links on the desktop

The laptop desktop has several shortcuts which make it easy to edit your code, test it works in the simulator, and upload it to the server ready for each race.

Desktop icons

From the links you will be able to view and edit the code, get updated instructions, run the simulator, upload your code for the competition and watch the live stream.

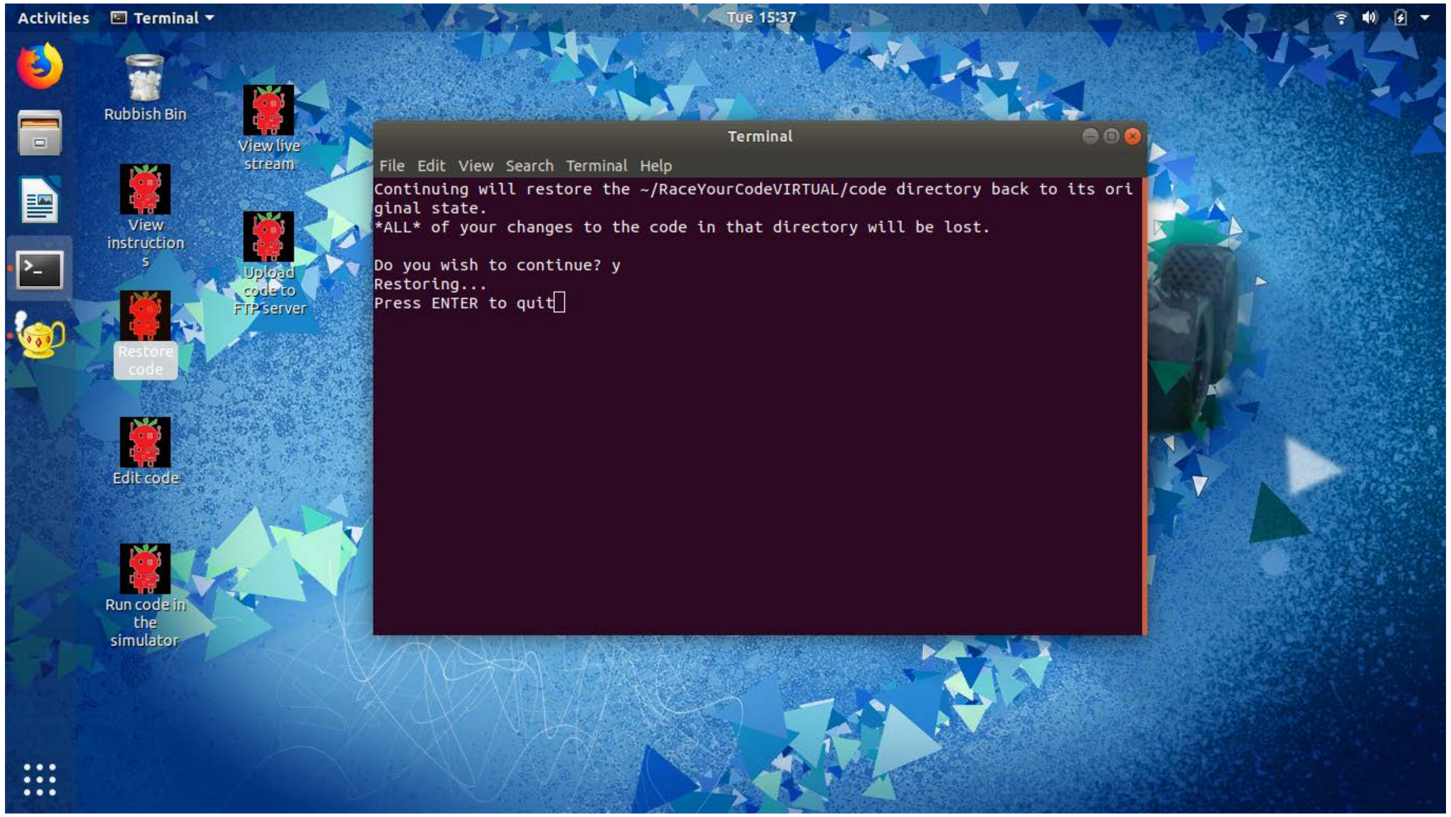
The link that opens this instruction document will attempt to update every time you click on the link to view them, so you should always have the latest information.



EDITING CODE

Restore code to the original version

Double click on the "Restore code" icon on the desktop to restore code to the original version. This deletes all changes you have made to the code without backing up! – so only do this if you really want to go back to the original code. We recommend doing this first before doing any code editing, to make sure to clear code any previous teams have written.



```
Continuing will restore the ~/RaceYourCodeVIRTUAL/code directory back to its ori
ginal state.
*ALL* of your changes to the code in that directory will be lost.

Do you wish to continue? y
Restoring...
Press ENTER to quit
```

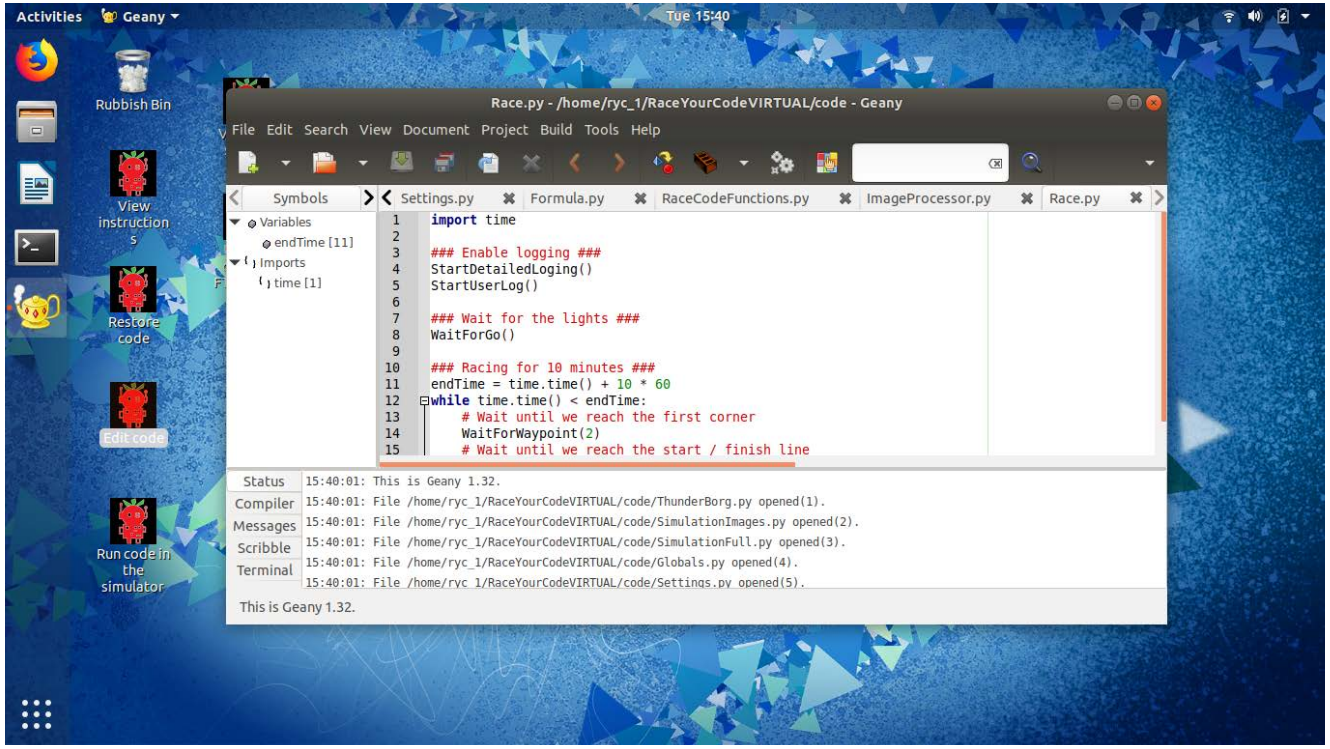
EDITING CODE

Edit code

This link opens the Geany editor with all the relevant python files opened in tabs.

The rightmost tabs have the code you will most likely wish to edit.

There are additional files that are not opened by default. Advanced coders may wish to edit these.



```
import time

### Enable logging ###
StartDetailedLoging()
StartUserLog()

### Wait for the lights ###
WaitForGo()

### Racing for 10 minutes ###
endTime = time.time() + 10 * 60
while time.time() < endTime:
    # Wait until we reach the first corner
    WaitForWaypoint(2)
    # Wait until we reach the start / finish line
```

```
15:40:01: This is Geany 1.32.
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/ThunderBorg.py opened(1).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/SimulationImages.py opened(2).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/SimulationFull.py opened(3).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/Globals.py opened(4).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/Settings.py opened(5).
```

This is Geany 1.32.

Code warning

The code is licensed for your use with a non-open source license, check the LICENSE.txt file for the full license terms.
For more information see https://www.formulapi.com/comment/240#comment-240

THE STANDARD CODE

Our standard code is made up of a number of files most of which is run in both the simulator and on the real robots.

There are more detailed explanations of the scripts and their parts in the Guides folder provided with the code. You should be given access details to all of the code for experimenting and development as part of your registration

Important note

Most of the code is common with our Formula Pi Monster series code. As such, there is a lot of useful information about the race code which may prove useful here:

https://www.formulapi.com/race-code

and also some insight into how our logic works here:

https://www.formulapi.com/tags/image-processing

SimulationFull.py /SimulationImages.py

These are explained in more detail later on, but these are scripts to help with development of the code without the robots, allowing you to run the bulk of the code using the provided simulator to generate camera data and track position.

Settings.py / Globals.py

These scripts hold settings and defaults respectively for the standard processing code. Most of the settings can be changed on the fly by overwriting Settings.py while the scripts are in control, but not all of them. The values in both of these files are shared between all of the scripts.

ImageProcessor.py

This is where the bulk of the code lives. This file is responsible for both the camera processing and the calculations for the motor output speeds. The code in this file is heavily math and OpenCV based, but has been kept fairly simple. If you wish to alter the image processing behaviour you will likely want to modify this script.

Race.py

This script runs alongside the image processing and is mostly intended for determining target track positions and other race type decisions. There are a number of examples provided for how this thread could be used, but it can access and alter most values in the image processing code as well.

RaceCodeFunctions.py

This script provides the simplified race API used in our Race.py examples. They provide functions such as logging and feedback from the processing code. A full list of the functions can be found here: https://www.formulapi.com/race-code/functions

ThunderBorg.py

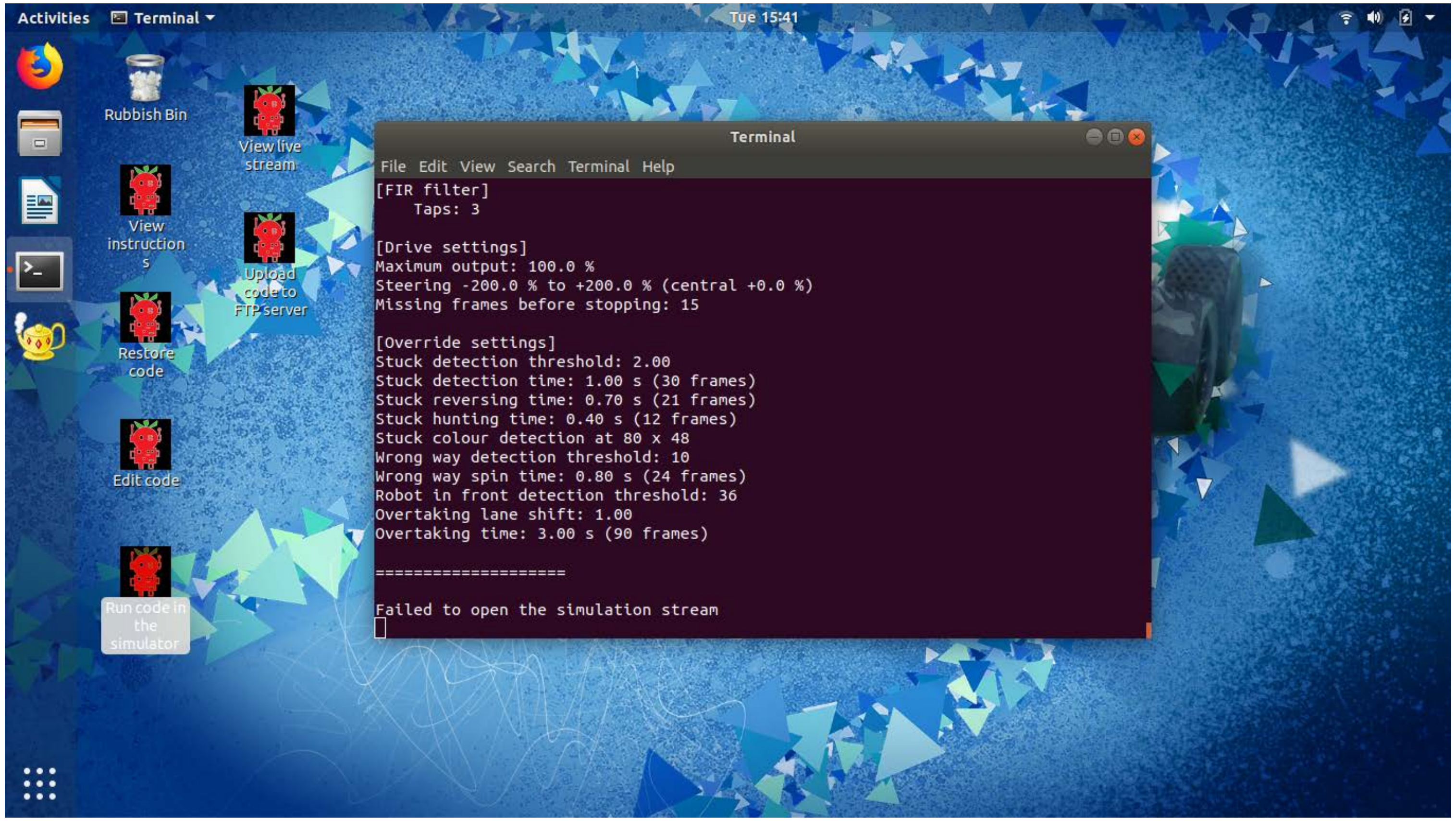
This script provides a simple API to control the on-board ThunderBorg motor controller without the need to write I2C messages by hand.

SIMULATING

Testing the code in the simulator

Run code in the simulator icon

This icon will start the simulator. The terminal window will state a failure. This is not a concern – it just shows that the simulation stream has not started.



```
[FIR filter]
    Taps: 3

[Drive settings]
Maximum output: 100.0 %
Steering -200.0 % to +200.0 % (central +0.0 %)
Missing frames before stopping: 15

[Override settings]
Stuck detection threshold: 2.00
Stuck detection time: 1.00 s (30 frames)
Stuck reversing time: 0.70 s (21 frames)
Stuck hunting time: 0.40 s (12 frames)
Stuck colour detection at 80 x 48
Wrong way detection threshold: 10
Wrong way spin time: 0.80 s (24 frames)
Robot in front detection threshold: 36
Overtaking lane shift: 1.00
Overtaking time: 3.00 s (90 frames)

===================

Failed to open the simulation stream
```

Start the simulation stream

The simulator shows the image the robot sees at the top left, the overview of the track at the bottom right and a top view of the robot displaying the both the current control outputs and the physics model output.

On the top right of the simulator, there is a button to start the video streaming. Click on this and wait for a few seconds. The green light should be visible on the robot outline (the top view of the robot)

Start the light sequence

Click on the "Green" button under the start settings. The lights should be visible in the robot view window. Then click the "Red" button. On clicking the "Green" for the second time, the robots should drive off.

Please note the light sequence needs to be Green-Red-Green

The robot should stop automatically after 10 minutes as per a timeout in the code.

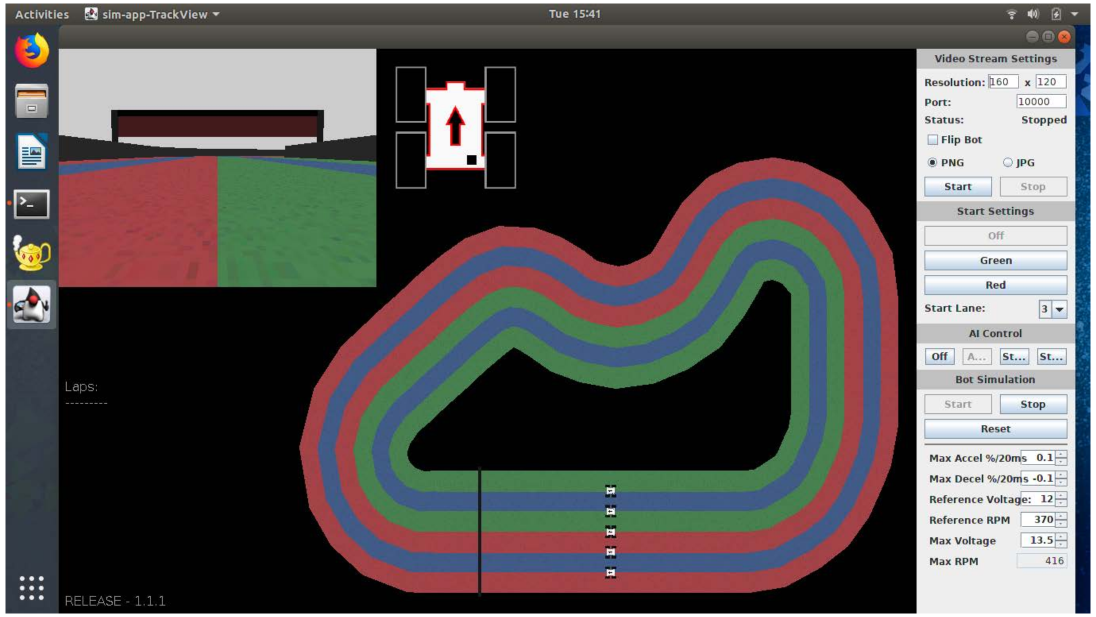
Reset the robot positions

To reset the robots back to the start position within the 10 minutes, click on the "Reset" button under Bot simulation.

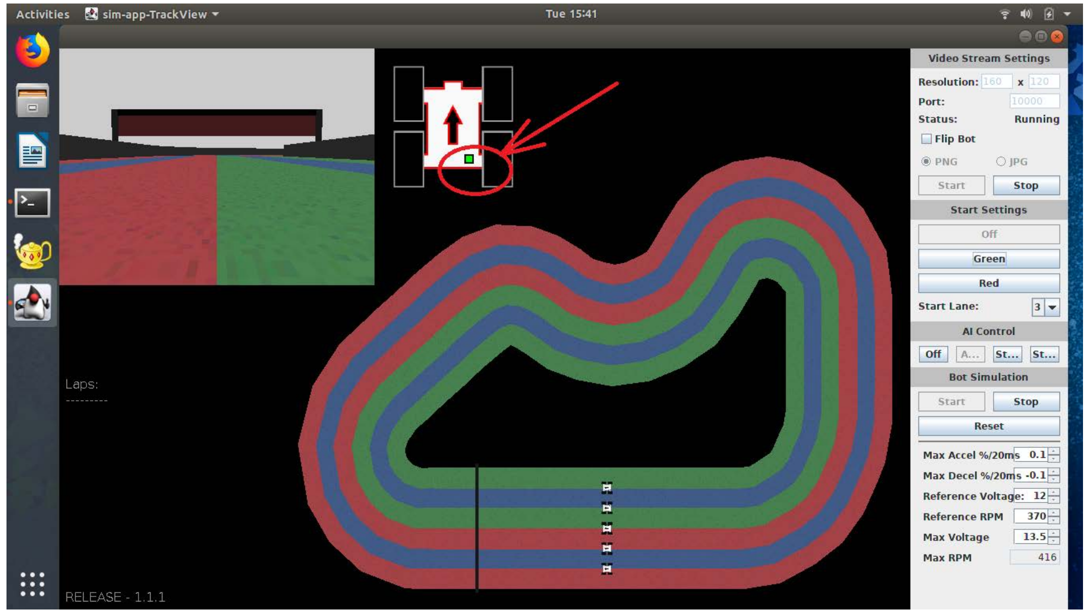
Close the simulator and re-open each time you want to make changes to code, or to re-run the light start sequence.

Note

You will need to close and restart the simulator after the 10 minutes has elapsed to run again.



Resolution: 160 x 120
Port: 10000
Status: Stopped
PNG [✓] JPG
Start Lane: 3
Max Accel %/20ms 0.1
Max Decel %/20ms -0.1
Reference Voltage: 12
Reference RPM 370
Max Voltage 13.5
Max RPM 416



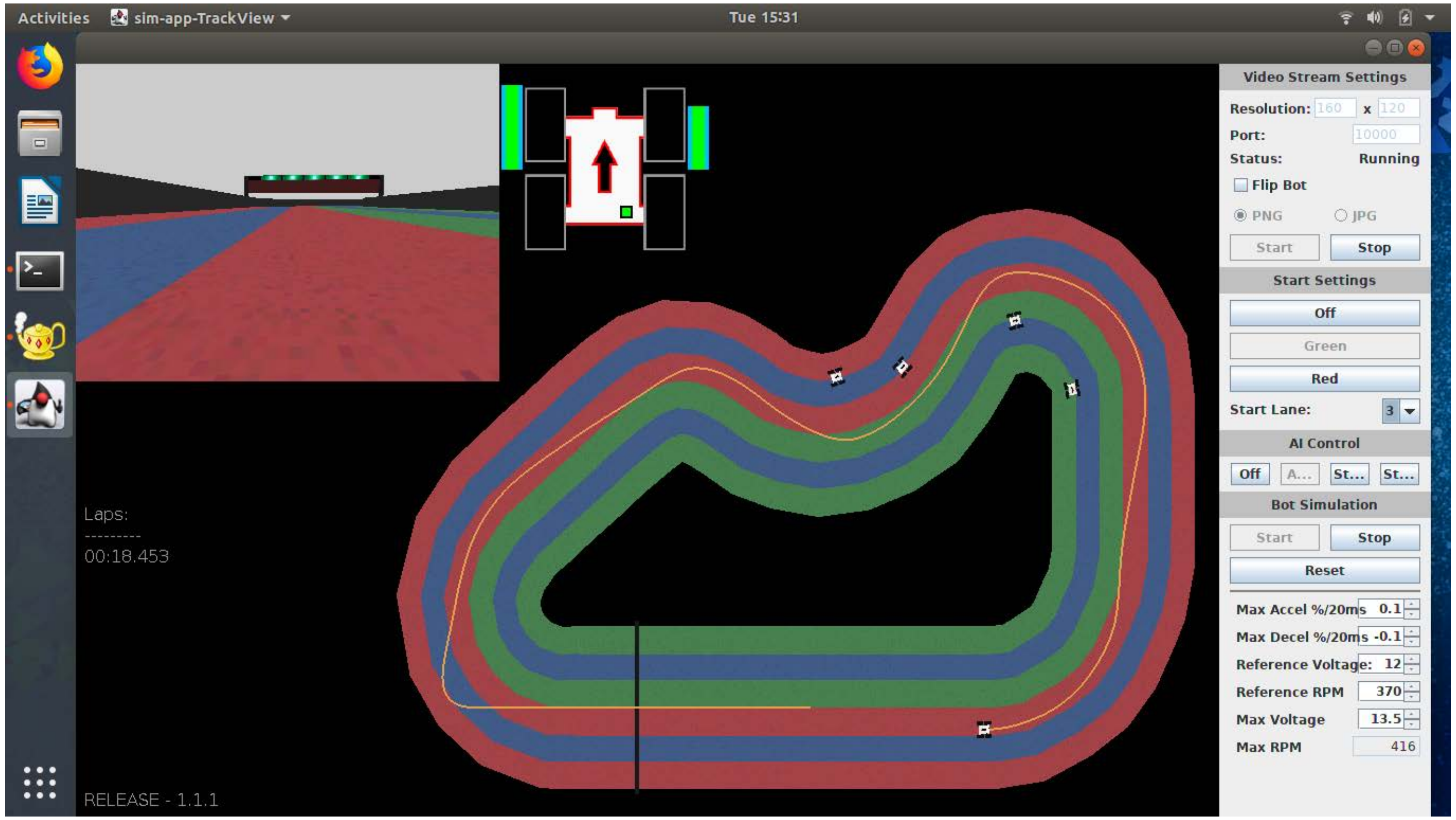
Resolution: 160 x 120
Port: 10000
Status: Running
PNG [✓] JPG
Start Lane: 3
Max Accel %/20ms 0.1
Max Decel %/20ms -0.1
Reference Voltage: 12
Reference RPM 370
Max Voltage 13.5
Max RPM 416

Simulator vs real world track

The simulator does have some limitations compared to the real robots:

Everything is evenly lit, there are no shadows or bright spots
Outside of the track is purely grey, in reality there are a lot of things in the background
The start lights do not project light like the real ones
There is no collision detection, the robots can run straight though the track walls
The robots are only roughly modelled, they do not behave exactly like the real robots

The final race will be on the real world track. It is possible to write code that works well in the simulator but not in real life. Please consider the real world aspects when designing your code.



Laps:

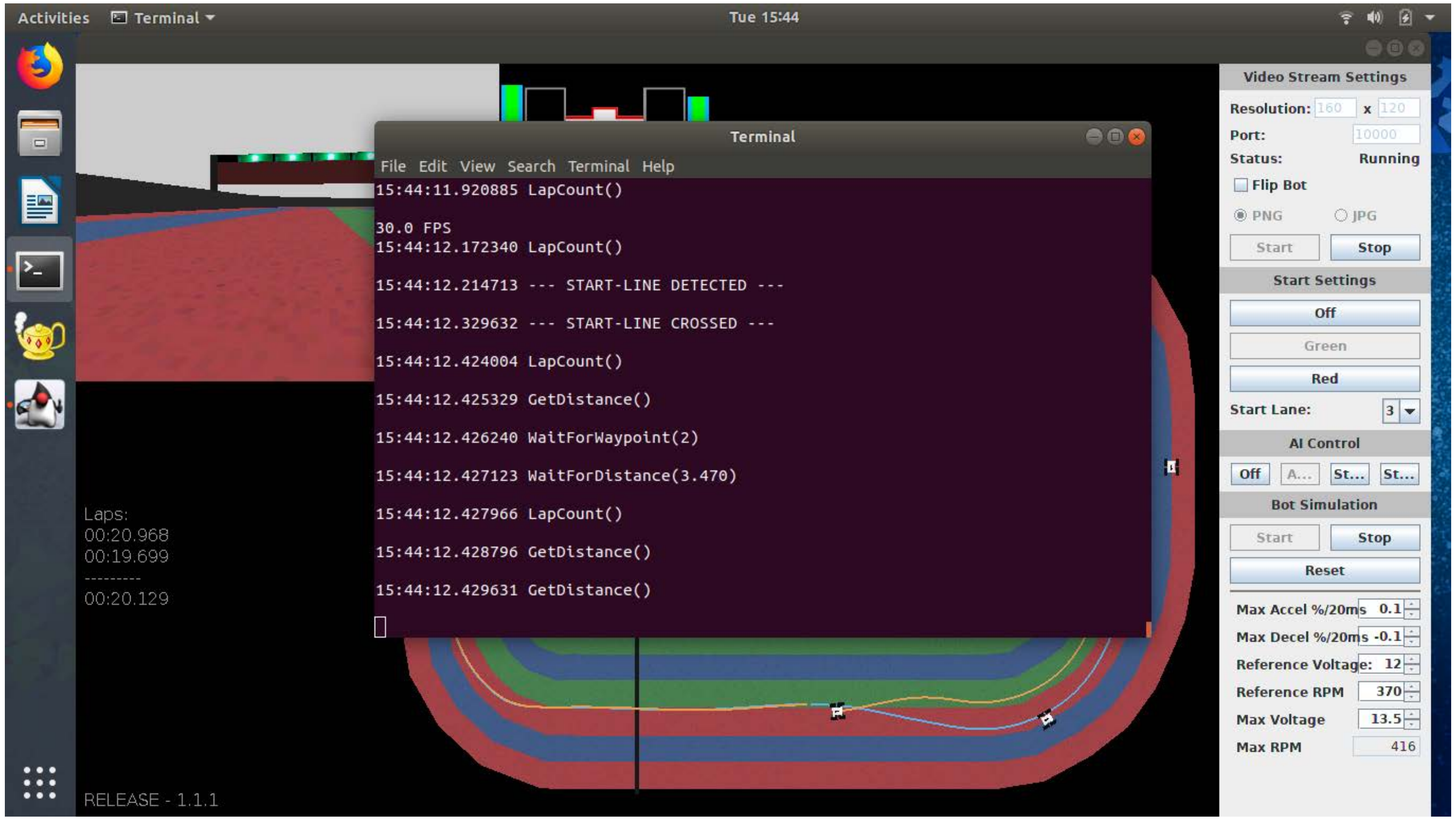
00:18.453

RELEASE - 1.1.1

Resolution: 160 x 120
Port: 10000
Status: Running
Flip Bot
[✓ PNG] JPG
Start Lane: 3
Max Accel %/20ms 0.1
Max Decel %/20ms -0.1
Reference Voltage: 12
Reference RPM 370
Max Voltage 13.5
Max RPM 416

Logging

The terminal displays log information which is very useful for debugging. It displays information such as when the start line is detected, or the overtaking code has been executed. It also displays the current frame rate which should not deviate too far from 30fps.



```
15:44:11.920885 LapCount()

30.0 FPS
15:44:12.172340 LapCount()

15:44:12.214713 --- START-LINE DETECTED ---

15:44:12.329632 --- START-LINE CROSSED ---

15:44:12.424004 LapCount()

15:44:12.425329 GetDistance()

15:44:12.426240 WaitForWaypoint(2)

15:44:12.427123 WaitForDistance(3.470)

15:44:12.427966 LapCount()

15:44:12.428796 GetDistance()

15:44:12.429631 GetDistance()
```

```
Laps:
00:20.968
00:19.699
---------
00:20.129

RELEASE - 1.1.1
```

Video Stream Settings
Resolution: 160 x 120
Port: 10000
Status: Running
Flip Bot
PNG [✓] JPG

Start Settings
Start Lane: 3

Max Accel %/20ms 0.1
Max Decel %/20ms -0.1
Reference Voltage: 12
Reference RPM 370
Max Voltage 13.5
Max RPM 416

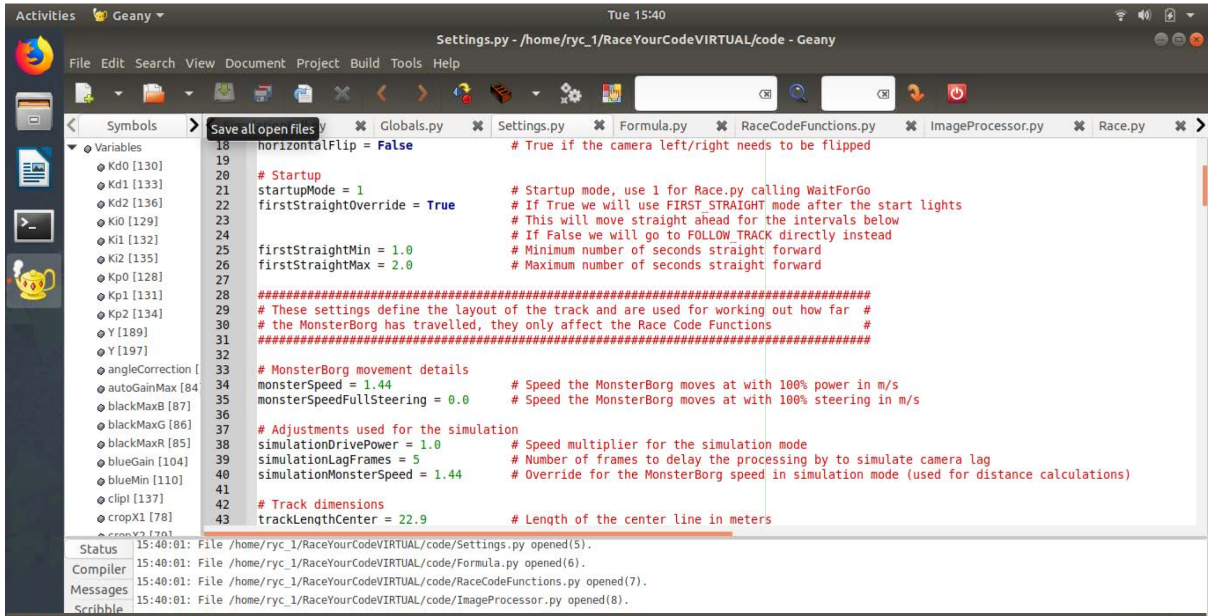
Your first code edit

One of the first things you may wish to do is to correct a bug in the code. Open the settings.py file from the editor. Change the value of the firstStraightMin to 1.0 and the value of firstStraightMax to 2.0

This change should prevent the situation where the robot drives off straight ahead after the start lights and crashes over the barrier as pictured below.

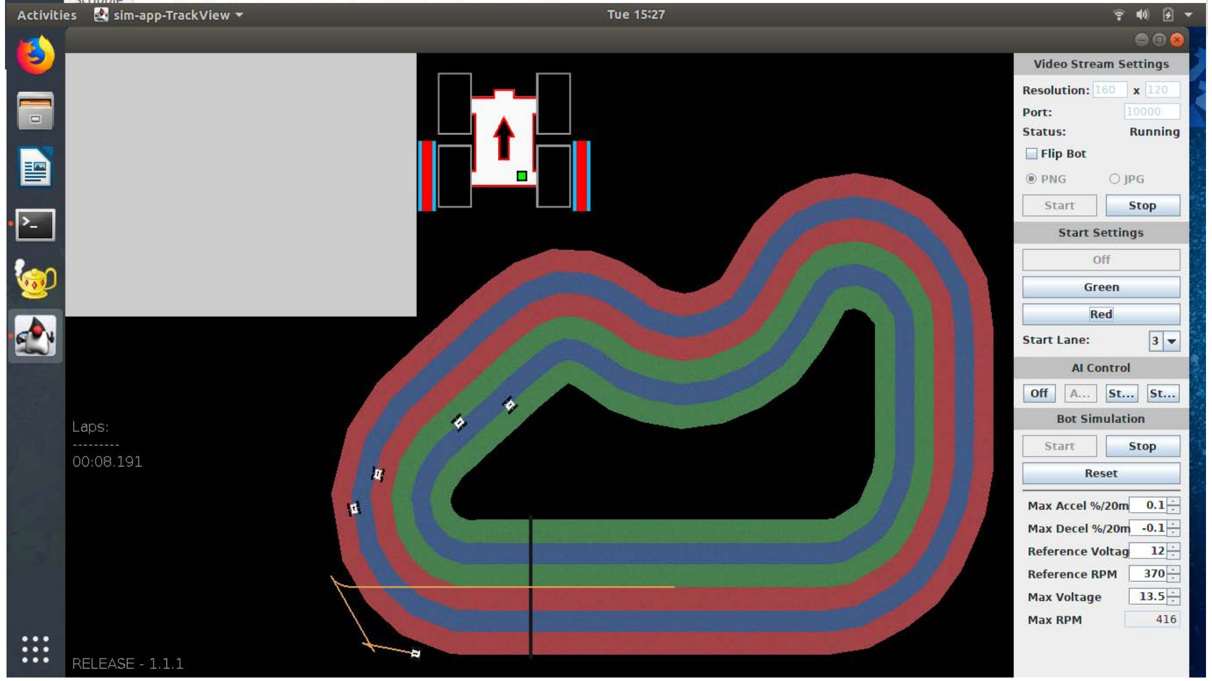
The simulator does not have collision detection, however there is in the virtual racing, meaning collisions will slow you down significantly.

Save the file, close and re-open the simulator. Now the robot should not experience this error.



```
horizontalFlip = False             # True if the camera left/right needs to be flipped

# Startup
startupMode = 1                    # Startup mode, use 1 for Race.py calling WaitForGo
firstStraightOverride = True       # If True we will use FIRST_STRAIGHT mode after the start lights
                                   # This will move straight ahead for the intervals below
                                   # If False we will go to FOLLOW_TRACK directly instead
firstStraightMin = 1.0             # Minimum number of seconds straight forward
firstStraightMax = 2.0             # Maximum number of seconds straight forward

###########################################################################################
# These settings define the layout of the track and are used for working out how far  #
# the MonsterBorg has travelled, they only affect the Race Code Functions             #
###########################################################################################

# MonsterBorg movement details
monsterSpeed = 1.44                # Speed the MonsterBorg moves at with 100% power in m/s
monsterSpeedFullSteering = 0.0     # Speed the MonsterBorg moves at with 100% steering in m/s

# Adjustments used for the simulation
simulationDrivePower = 1.0         # Speed multiplier for the simulation mode
simulationLagFrames = 5            # Number of frames to delay the processing by to simulate camera lag
simulationMonsterSpeed = 1.44      # Override for the MonsterBorg speed in simulation mode (used for distance calculations)

# Track dimensions
trackLengthCenter = 22.9           # Length of the center line in meters
```

```
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/Settings.py opened(5).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/Formula.py opened(6).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/RaceCodeFunctions.py opened(7).
15:40:01: File /home/ryc_1/RaceYourCodeVIRTUAL/code/ImageProcessor.py opened(8).
```



Laps:

00:08.191

RELEASE - 1.1.1

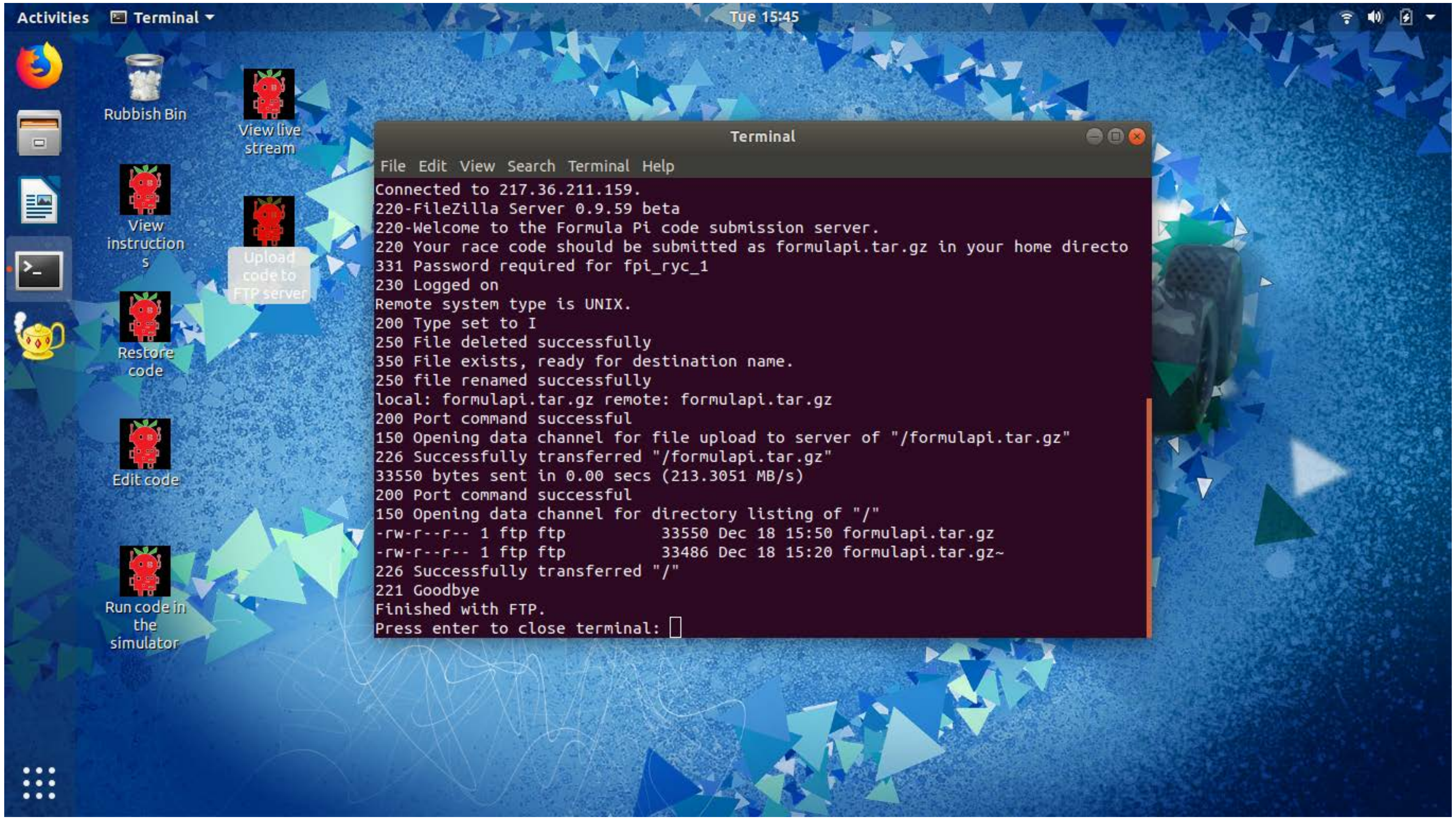
Resolution: 160 x 120
Port: 10000
Status: Running
Start Lane: 3
Max Accel %/20m 0.1
Max Decel %/20m -0.1
Reference Voltag 12
Reference RPM 370
Max Voltage 13.5
Max RPM 416

Rolls-Royce

RaceYourCode VIRTUAL

Upload the code

You will need to upload the code at least 24 hours before each race. Make sure to re-upload your code even if it hasn't changed since last race. The "Upload code to FTP server" icon will do this for you. Double check to see that the directory listing shows the formulapi.tar.gz file and that the date is correct (our server is GMT). The link creates a backup of the last uploaded code and calls this formulapi.tar.gz~

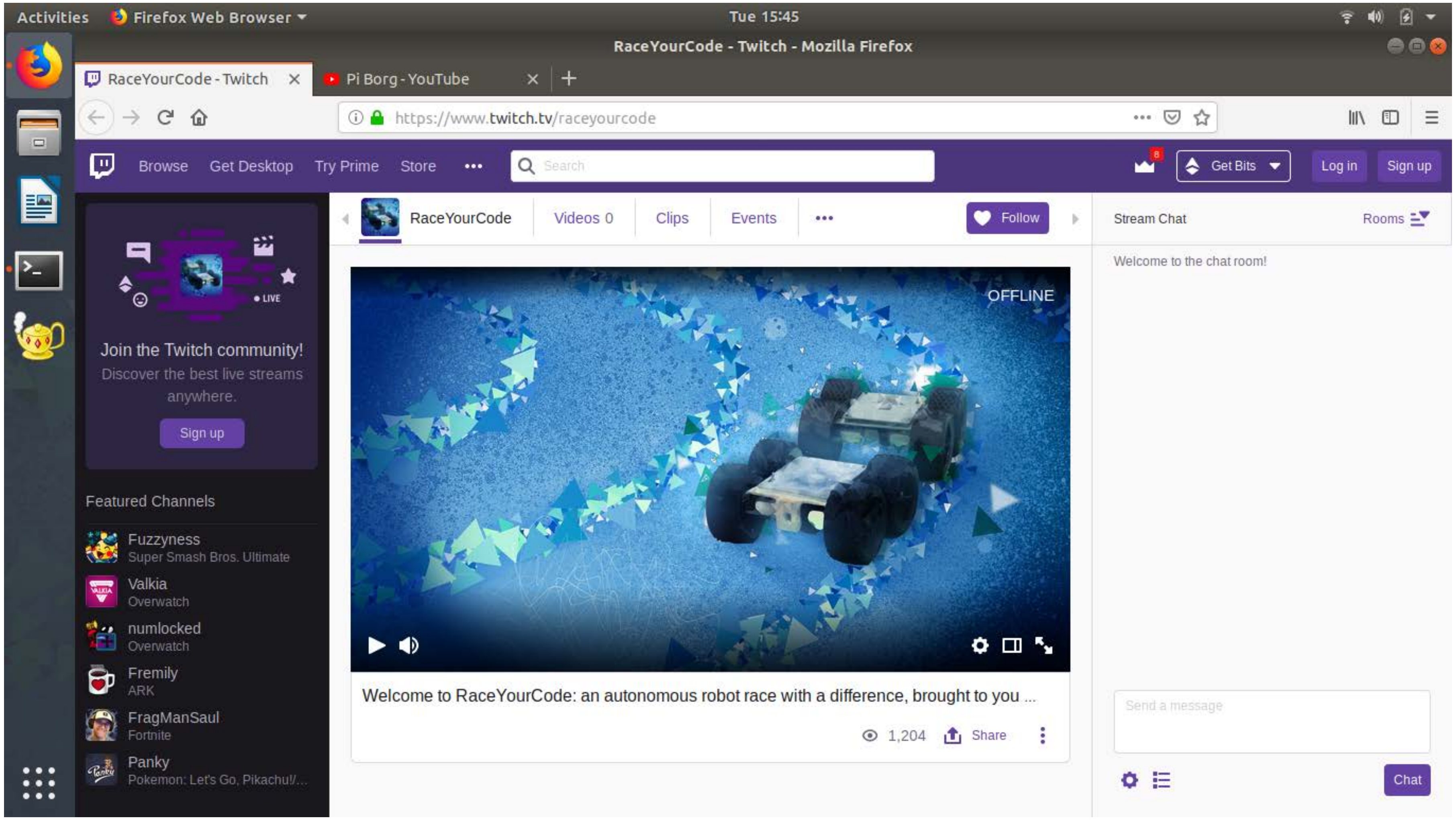


```
Connected to 217.36.211.159.
220-FileZilla Server 0.9.59 beta
220-Welcome to the Formula Pi code submission server.
220 Your race code should be submitted as formulapi.tar.gz in your home directo
331 Password required for fpi_ryc_1
230 Logged on
Remote system type is UNIX.
200 Type set to I
250 File deleted successfully
350 File exists, ready for destination name.
250 file renamed successfully
local: formulapi.tar.gz remote: formulapi.tar.gz
200 Port command successful
150 Opening data channel for file upload to server of "/formulapi.tar.gz"
226 Successfully transferred "/formulapi.tar.gz"
33550 bytes sent in 0.00 secs (213.3051 MB/s)
200 Port command successful
150 Opening data channel for directory listing of "/"
-rw-r--r-- 1 ftp ftp          33550 Dec 18 15:50 formulapi.tar.gz
-rw-r--r-- 1 ftp ftp          33486 Dec 18 15:20 formulapi.tar.gz~
226 Successfully transferred "/"
221 Goodbye
Finished with FTP.
Press enter to close terminal:
```

View the stream

The view live stream icon will open a firefox window connected to twitch.tv/raceyourcode. This is where we intend to stream the racing from. Please be sure to be there on 16th January for the test round 1. If you sign up for a twitch account, you will be able to use the chat room.

If we have problems with the stream we may use the YouTube channel which is open in the firefox browser on the next tab along.



Welcome to RaceYourCode: an autonomous robot race with a difference, brought to you ...

More information and help

Loads more information is available at www.formulapi.com

Click on the "Race Code" link at the top for detail information about the code and how it works. The "Race code processing results" shows a few simple animations which show the difference between offset, angle and curvature calculations

There is a specific RaceYourCode thread on the forum where you can post any questions you have.

The forums and the blog show lots of useful information.

Thanks for competing, and we'd love to hear your feedback!