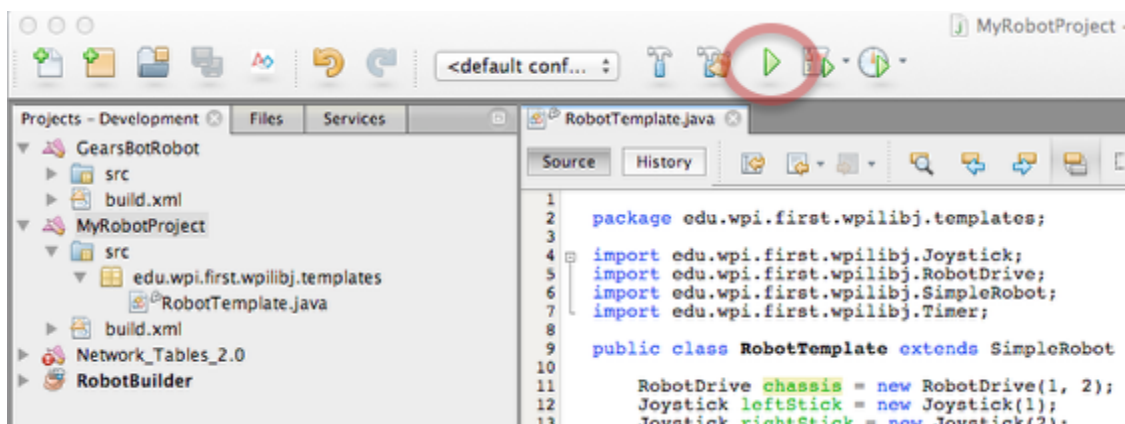


## Running the program on the robot

# Running the program on the robot

Once the program is finished and NetBeans is configured the program can be run very easily. There are a few things you should know about running the program immediately after flashing a new image onto the cRIO that are described here.

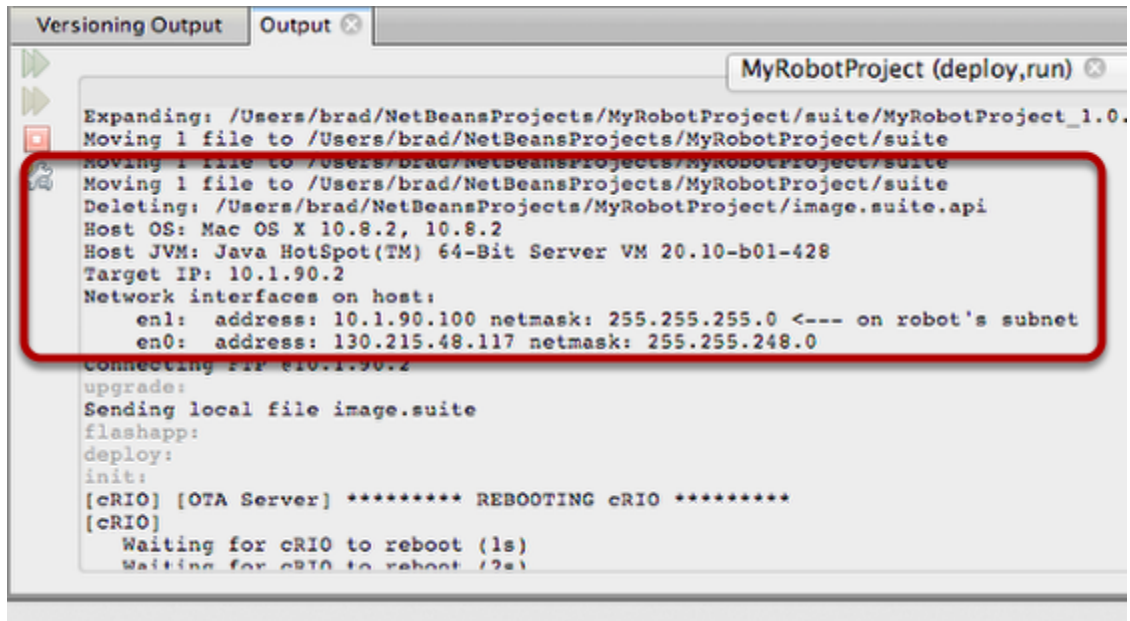
## Running the program



Make sure the program is set as the main program, indicated in bold in the left pane (see instructions in the [previous article](#)). To run the program simply click on the green right-facing triangle. When you do this you'll see messages about the program building in the "Output" window (usually at the bottom of NetBeans). Then the robot reboots, and the program starts. These steps with their associated messages are shown below.

## Running the program on the robot

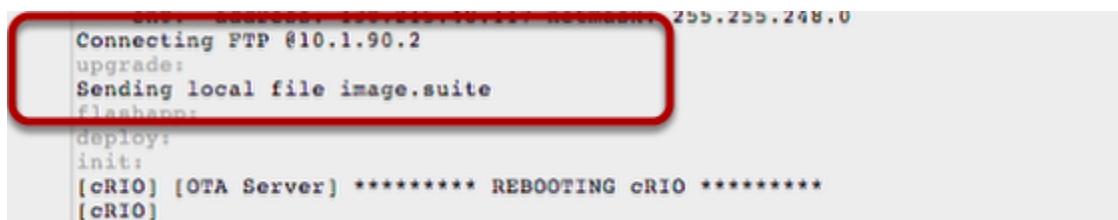
### Getting ready to transfer the program to the robot



```
Versioning Output  Output x  MyRobotProject (deploy,run) x
Expanding: /Users/brad/NetBeansProjects/MyRobotProject/suite/MyRobotProject_1.0.
Moving 1 file to /Users/brad/NetBeansProjects/MyRobotProject/suite
Moving 1 file to /Users/brad/NetBeansProjects/MyRobotProject/suite
Moving 1 file to /Users/brad/NetBeansProjects/MyRobotProject/suite
Deleting: /Users/brad/NetBeansProjects/MyRobotProject/image.suite.api
Host OS: Mac OS X 10.8.2, 10.8.2
Host JVM: Java HotSpot(TM) 64-Bit Server VM 20.10-b01-428
Target IP: 10.1.90.2
Network interfaces on host:
  en1: address: 10.1.90.100 netmask: 255.255.255.0 <--- on robot's subnet
  en0: address: 130.215.48.117 netmask: 255.255.248.0
Connecting FTP @10.1.90.2
upgrade:
Sending local file image.suite
flashapp:
deploy:
init:
[cRIO] [OTA Server] ***** REBOOTING cRIO *****
[cRIO]
Waiting for cRIO to reboot (1s)
Waiting for cRIO to reboot (2s)
```

NetBeans verifies that there is a network interface on the development computer that is on the same subnet as the robot. In this case since we have set NetBeans and the robot to Team 190, then NetBeans notices that the development computer assigned IP address is 10.1.90.100 the same subnet as the robot (10.1.90.2).

### Transferring the program to the robot

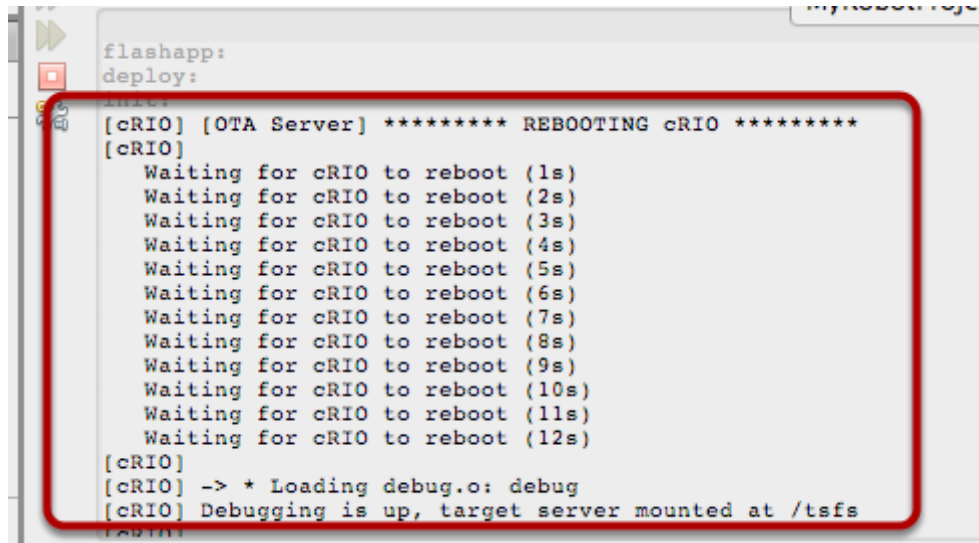


```
en1: address: 10.1.90.100 netmask: 255.255.255.0
en0: address: 130.215.48.117 netmask: 255.255.248.0
Connecting FTP @10.1.90.2
upgrade:
Sending local file image.suite
flashapp:
deploy:
init:
[cRIO] [OTA Server] ***** REBOOTING cRIO *****
[cRIO]
```

NetBeans then uses FTP to send the program to the robot. You can see the IP address of the robot (10.1.90.2) and it is transferring a file called image.suite, the default name for the program. Then the cRIO is rebooted.

# Running the program on the robot

## Rebooting the cRIO

A screenshot of the NetBeans IDE's Output window. The window title is 'myRobotProject'. The output text is as follows:

```
flashapp:
deploy:
[cRIO] [OTA Server] ***** REBOOTING cRIO *****
[cRIO]
  Waiting for cRIO to reboot (1s)
  Waiting for cRIO to reboot (2s)
  Waiting for cRIO to reboot (3s)
  Waiting for cRIO to reboot (4s)
  Waiting for cRIO to reboot (5s)
  Waiting for cRIO to reboot (6s)
  Waiting for cRIO to reboot (7s)
  Waiting for cRIO to reboot (8s)
  Waiting for cRIO to reboot (9s)
  Waiting for cRIO to reboot (10s)
  Waiting for cRIO to reboot (11s)
  Waiting for cRIO to reboot (12s)
[cRIO]
[cRIO] -> * Loading debug.o: debug
[cRIO] Debugging is up, target server mounted at /tsfs
```

A red rectangular box highlights the section of the output from the '[cRIO] [OTA Server] \*\*\*\*\* REBOOTING cRIO \*\*\*\*\*' header down to the final status messages.

It takes about 12 seconds before the cRIO has rebooted enough to start echoing status message in the NetBeans Output window. You can see the NetBeans plugin counting the time as the robot is rebooting. When it finally starts running again, the counting messages stop and are replaced by status messages.

**Important:** the first time you run a program after reimaging the cRIO you'll see the **counting messages going on forever**. This is because the OTA server (the robot code that handles rebooting the cRIO) hasn't yet started. It gets downloaded with the first program you try to run, but it only starts when the robot reboots. So if you just reimaged the cRIO you must manually reboot the robot manually the first time you run a program to get the OTA server started, then subsequent downloads with that image will work correctly. When you see the "Waiting for cRIO" messages going well past 12 seconds, manually reboot the robot and everything should start working properly after that.

## Robot program is now running

After a number of status messages you should see an announcement from the OTA Server that it is running. You might also see some messages indicating that the "Default robotInit() method is running" and "Default disabled() method is running". This tells you that your program hasn't supplied your own implementation of these methods in the SimpleRobot main class. This is not important and would only be a problem if you thought that your program was overriding those built-in methods.

## Running the program on the robot

You can now test your robot program by enabling the robot in either teleop or autonomous modes using the Driver Station. You might see additional messages in this window if your program throws an exception or if you have some debug printing.

**Caution:** when testing a robot be sure that it is on blocks and the wheels are free to turn. This will prevent the robot from driving away from you in case there are programming errors.

If you've programmed a robot with this sample program it can be tested at this point. Set the robot in Teleop mode and verify that the joysticks control the motors. You can operate the left joystick and verify that pushing forward operates the left side motor(s) in the forward direction. If the left joystick operates the right motors then either the joysticks need to be swapped or the PWM cables going to the speed controllers. Pulling the joysticks back the motors should operate backwards. If the motor directions are reversed look for the `setInvertedMotors()` method on the `RobotDrive` class to invert the direction of one or more motors.

Verify that the autonomous program works by setting the robot into Autonomous mode and the robot should drive forwards for 2 seconds.