Program your robot…

Wirelessly!

Jeremy Cole
jeremy@jcole.us
Technical Mentor
FTC 11574

The problem…


Micro USB!
Noooooooo!

Programming via USB…

Programming is slow!

- Get up from comfy chair and walk to sharp-edged robot
- Dig into robot, disconnect, and un-mount Robot Controller phone
- Connect phone to computer and press a button in Android Studio
- Disconnect phone and walk back to robot
- Re-mount, re-connect phone, and walk away
- Press “go” on Driver Station phone to test
- Bandage hands, clean up blood
- Repeat!

Testing is dangerous!

- The micro USB connectors on cables are easily bent!
- The phone's micro USB **port** is easily damaged – possibly ruining the phone! (Or at least requiring a visit to iFixit…)
- Static discharge (zaaaaaap!) can damage the phone and/or other electronics – walking back and forth on the floor mats in dry climates (such as in Reno!) generates a lot of static!

The solution…

Program wirelessly! – But how?

~ Part 1 ~

Connect the computer to the robot over WiFi

WiFi Direct networking

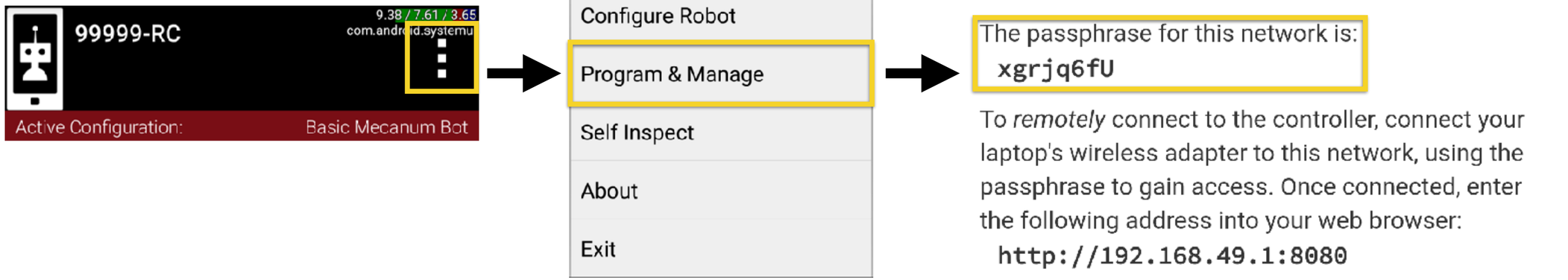
- The Robot Controller (RC) and Driver Station (DS) connect to each other using “WiFi Direct” (a phone-to-phone private network)

- So, neither phone is typically on your home/school/etc. WiFi
  - And this is required by the FTC rules, anyway!

WiFi Direct networking

- With WiFi Direct:
  - The RC “creates” a network named like “DIRECT-xy-1234-RC”
  - The DS “pairs” with the RC to join the WiFi Direct network
  - The RC phone always has IP address 192.168.49.1

Any computer can join WiFi Direct



This works, but…

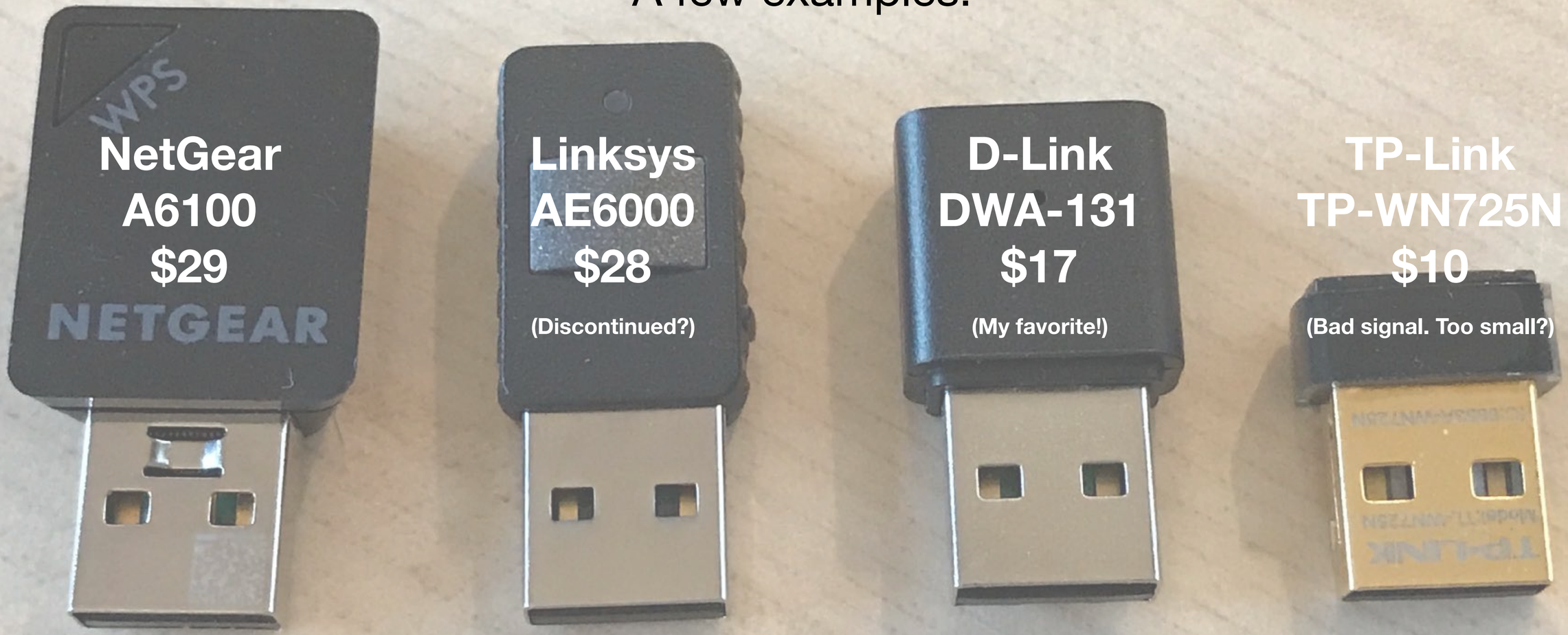
Once you've joined the RC network, your computer is off the Internet!

No Internet? Yes, problems.

- No Github pulls/pushes
- No Google Docs/Drive/Sheets
- No Google Search or StackExchange for Java or FTC reference
- No chat or videoconference with your remote teammates
- No slacking off online (okay, maybe *that* is not a problem!)

~ *Solution* ~

Add a USB WiFi adapter to your PC, and you can
join two WiFi network simultaneously.
A few examples:



The “adb” command

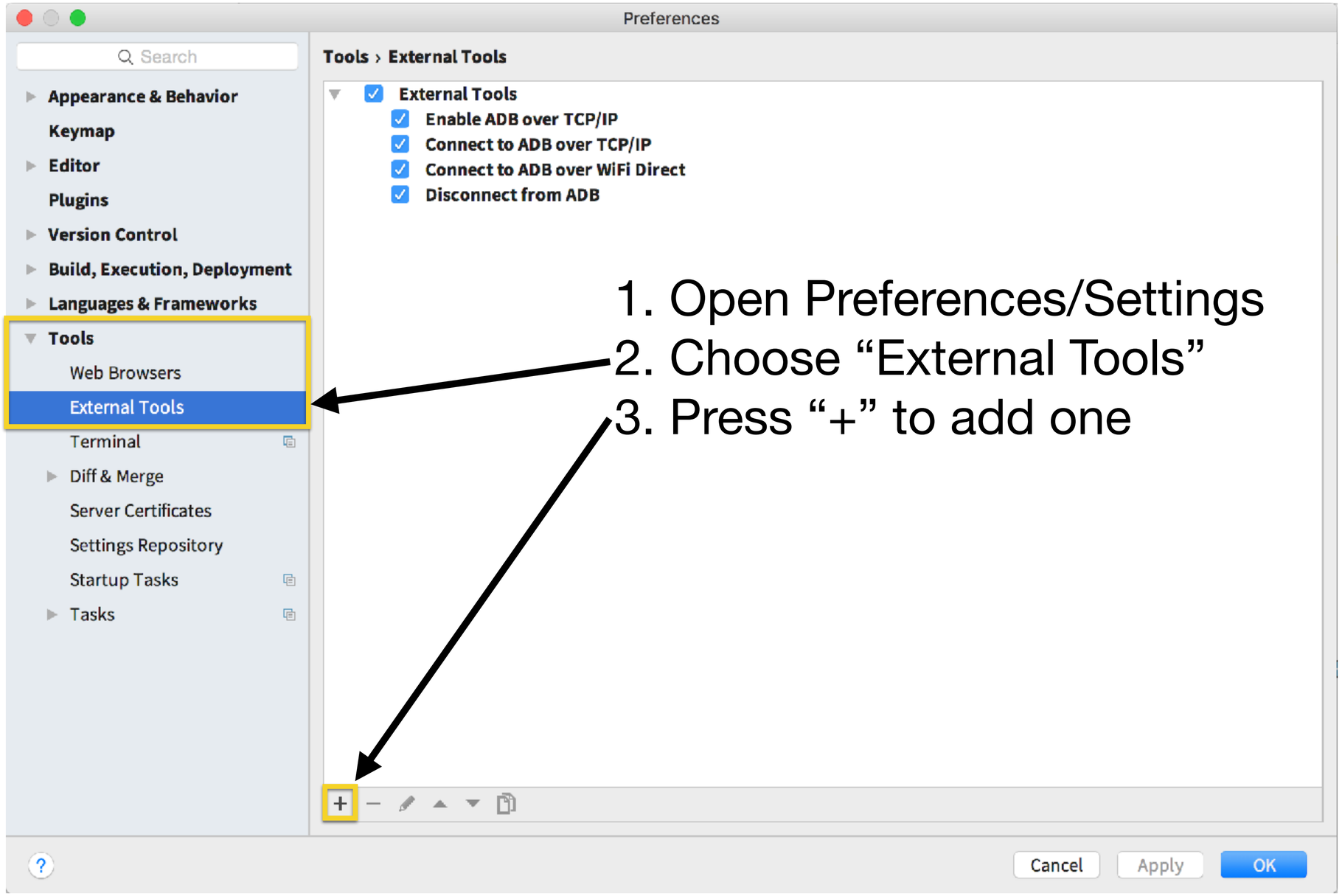
- The `adb` command (ADB = Android Debug Bridge) is used by Android Studio to interact with a USB-connected phone
- It’s possible to tell ADB to connect to the phone over a WiFi network (using TCP/IP) instead of USB:
  - `adb tcpip <port>`
  - `adb connect <ip>:<port>`

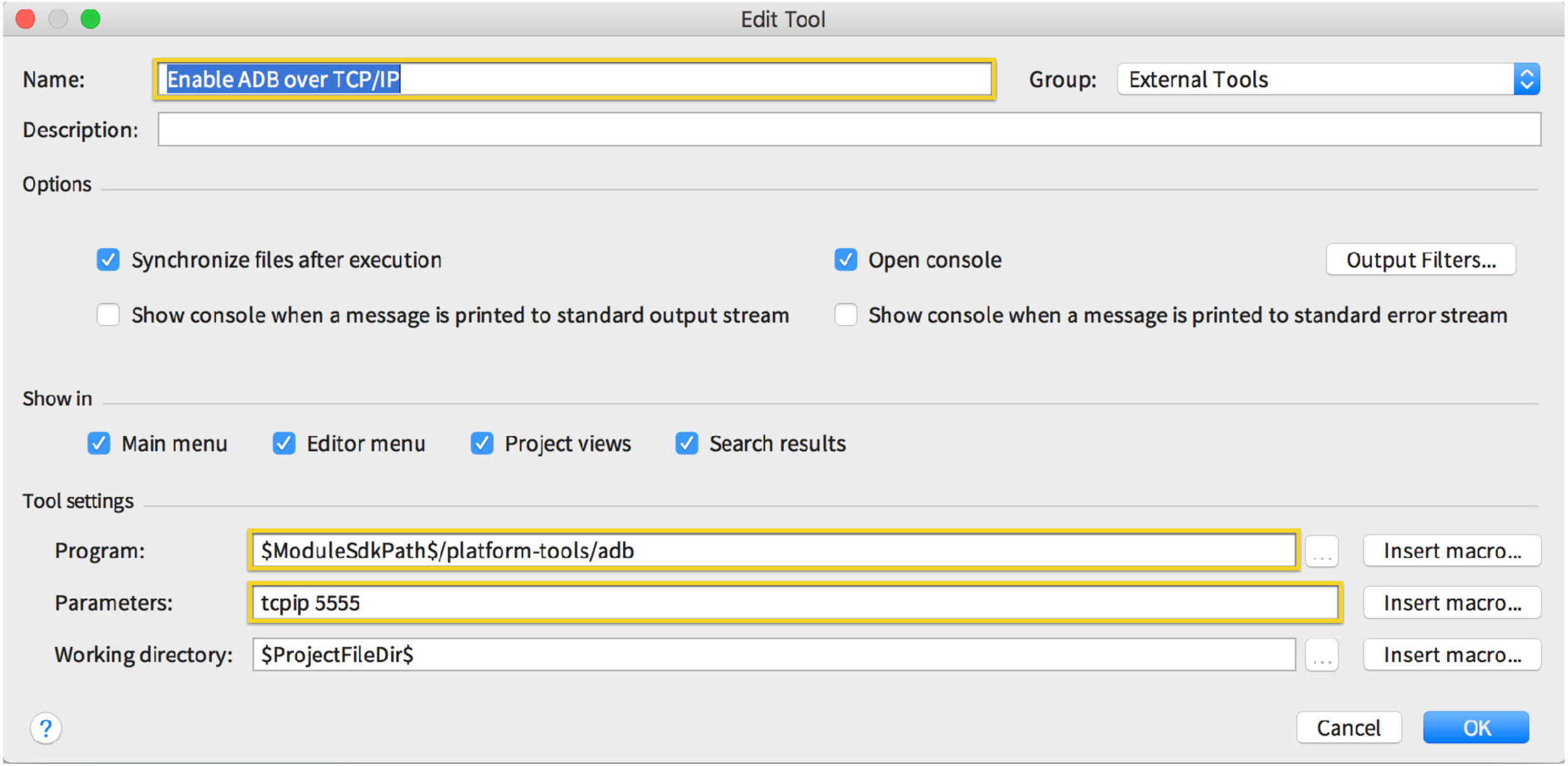
~ Part 2 ~

Make it as *easy* as possible to use

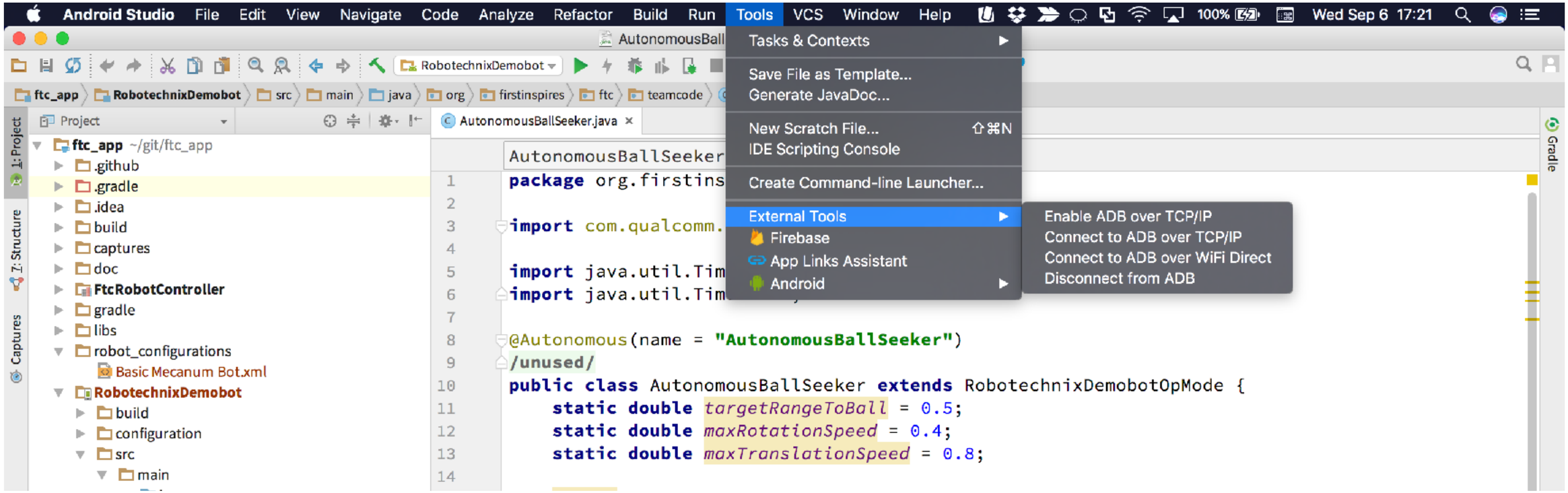
Android Studio “External Tools”

- Who wants to run those `adb` commands manually every time you want to program your robot?!
- The “External Tools” feature allows you to add your own custom commands to the Android Studio menu tree


Preferences
Search
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Web Browsers
External Tools
Terminal
Diff & Merge
Server Certificates
Settings Repository
Startup Tasks
Tasks
Tools › External Tools
External Tools
Enable ADB over TCP/IP
Connect to ADB over TCP/IP
Connect to ADB over WiFi Direct
Disconnect from ADB
1. Open Preferences/Settings
2. Choose “External Tools”
3. Press “+” to add one
Cancel
Apply
OK


Edit Tool
Name:
Enable ADB over TCP/IP
Group:
External Tools
Description:
Options
Synchronize files after execution
Open console
Output Filters...
Show console when a message is printed to standard output stream
Show console when a message is printed to standard error stream
Show in
Main menu
Editor menu
Project views
Search results
Tool settings
Program:
$ModuleSdkPath$/platform-tools/adb
Insert macro...
Parameters:
tcpip 5555
Insert macro...
Working directory:
$ProjectFileDir$
Insert macro...
Cancel
OK

Any commands added show up under:
Tools ➔ External Tools



~ Bonus ~

Real-time debugging With Android logcat

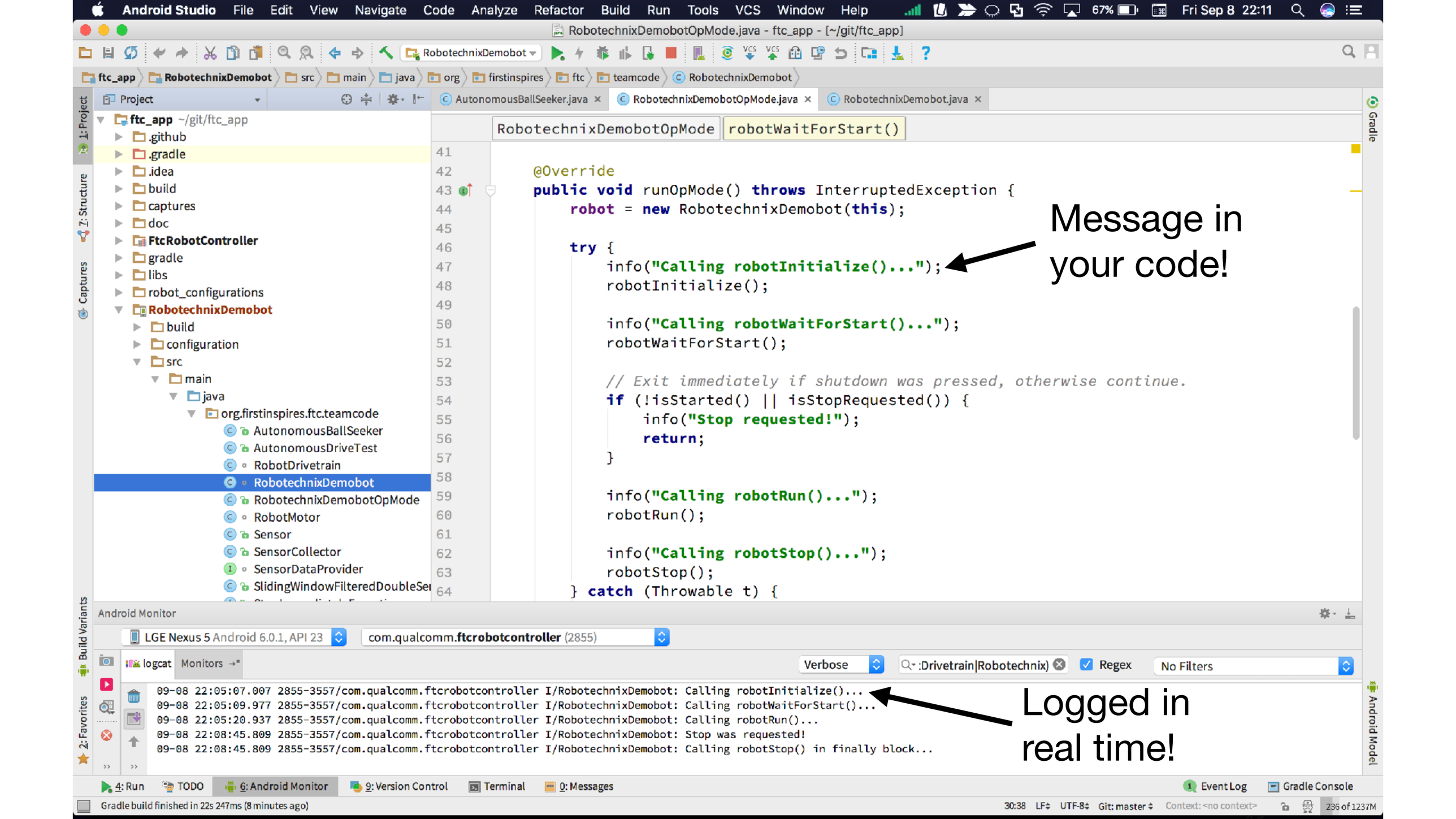
The “Android Monitor” (aka logcat)

- Shows log messages from Android and any running programs
- Can be used from your code with:

```
import android.util.Log;

Log.i(“SomeCleverLogTag”, “Your message”);
```

- You can easily make a nice wrapper for this e.g. info()


Android Studio File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Fri Sep 8 22:11
RobotechnixDemobotOpMode.java - ftc_app - [~/git/ftc_app]
ftc_app RobotechnixDemobot src main java org firstinspires ftc teamcode RobotechnixDemobot
Project
ftc_app ~/git/ftc_app
.github
.gradle
.idea
build
captures
doc
FtcRobotController
gradle
libs
robot_configurations
RobotechnixDemobot
build
configuration
src
main
java
org.firstinspires.ftc.teamcode
AutonomousBallSeeker
AutonomousDriveTest
RobotDrivetrain
RobotechnixDemobot
RobotechnixDemobotOpMode
RobotMotor
Sensor
SensorCollector
SensorDataProvider
SlidingWindowFilteredDoubleSe
AutonomousBallSeeker.java
RobotechnixDemobotOpMode.java
RobotechnixDemobot.java
RobotechnixDemobotOpMode
robotWaitForStart()
41
42 @Override
43 public void runOpMode() throws InterruptedException {
44 robot = new RobotechnixDemobot(this);
45
46 try {
47 info("Calling robotInitialize()...");
48 robotInitialize();
49
50 info("Calling robotWaitForStart()...");
51 robotWaitForStart();
52
53 // Exit immediately if shutdown was pressed, otherwise continue.
54 if (!isStarted() || isStopRequested()) {
55 info("Stop requested!");
56 return;
57 }
58
59 info("Calling robotRun()...");
60 robotRun();
61
62 info("Calling robotStop()...");
63 robotStop();
64 } catch (Throwable t) {
Message in your code!
Android Monitor
LGE Nexus 5 Android 6.0.1, API 23
com.qualcomm.ftcrobotcontroller (2855)
logcat
Monitors
Verbose
:Drivetrain|Robotechnix)
Regex
No Filters
09-08 22:05:07.007 2855-3557/com.qualcomm.ftcrobotcontroller I/RobotechnixDemobot: Calling robotInitialize()...
09-08 22:05:09.977 2855-3557/com.qualcomm.ftcrobotcontroller I/RobotechnixDemobot: Calling robotWaitForStart()...
09-08 22:05:20.937 2855-3557/com.qualcomm.ftcrobotcontroller I/RobotechnixDemobot: Calling robotRun()...
09-08 22:08:45.809 2855-3557/com.qualcomm.ftcrobotcontroller I/RobotechnixDemobot: Stop was requested!
09-08 22:08:45.809 2855-3557/com.qualcomm.ftcrobotcontroller I/RobotechnixDemobot: Calling robotStop() in finally block...
Logged in real time!
4: Run
TODO
6: Android Monitor
9: Version Control
Terminal
0: Messages
Event Log
Gradle Console
Gradle build finished in 22s 247ms (8 minutes ago)
30:38 LF UTF-8 Git: master Context: <no context>
236 of 1237M

Thank you!

blog.jcole.us/ftc