Android Studio Programming (Java)

Kickoff 2018 – Rover Ruckus

Agenda

- About Me
- Installation and Downloading
- Java
- Robot Parts – Moving Parts
- Structure of the Code
- Look at some code
- Robot Parts – Sensors
- Tips and Tricks
- Q&A
- Resources

About Me

- Nathan Choi
- Lead Programmer and Scout for Brain Stormz 10298
- 6th season of FIRST (5th of FTC)
- Went to semi finals at World Championship in Houston 2017
- Went to World Championship in Houston 2018 as Dean's List Finalist
- 2018 Regional Control Award Winner

10298

Installation and Downloading

- Follow the instructions:
- FIRST: https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/android-studio-tutorial.pdf
  - Great setup resource
  - Ignore times to complete
- Git: https://github.com/ftctechnh/ftc_app/wiki

10298

Java

- Here are some resources:
- Codeacademy: https://www.codecademy.com/learn/learn-java
- Git: https://github.com/ftctechnh/ftc_app/wiki
- Oracle: https://docs.oracle.com/javase/tutorial/

10298

Robot Parts

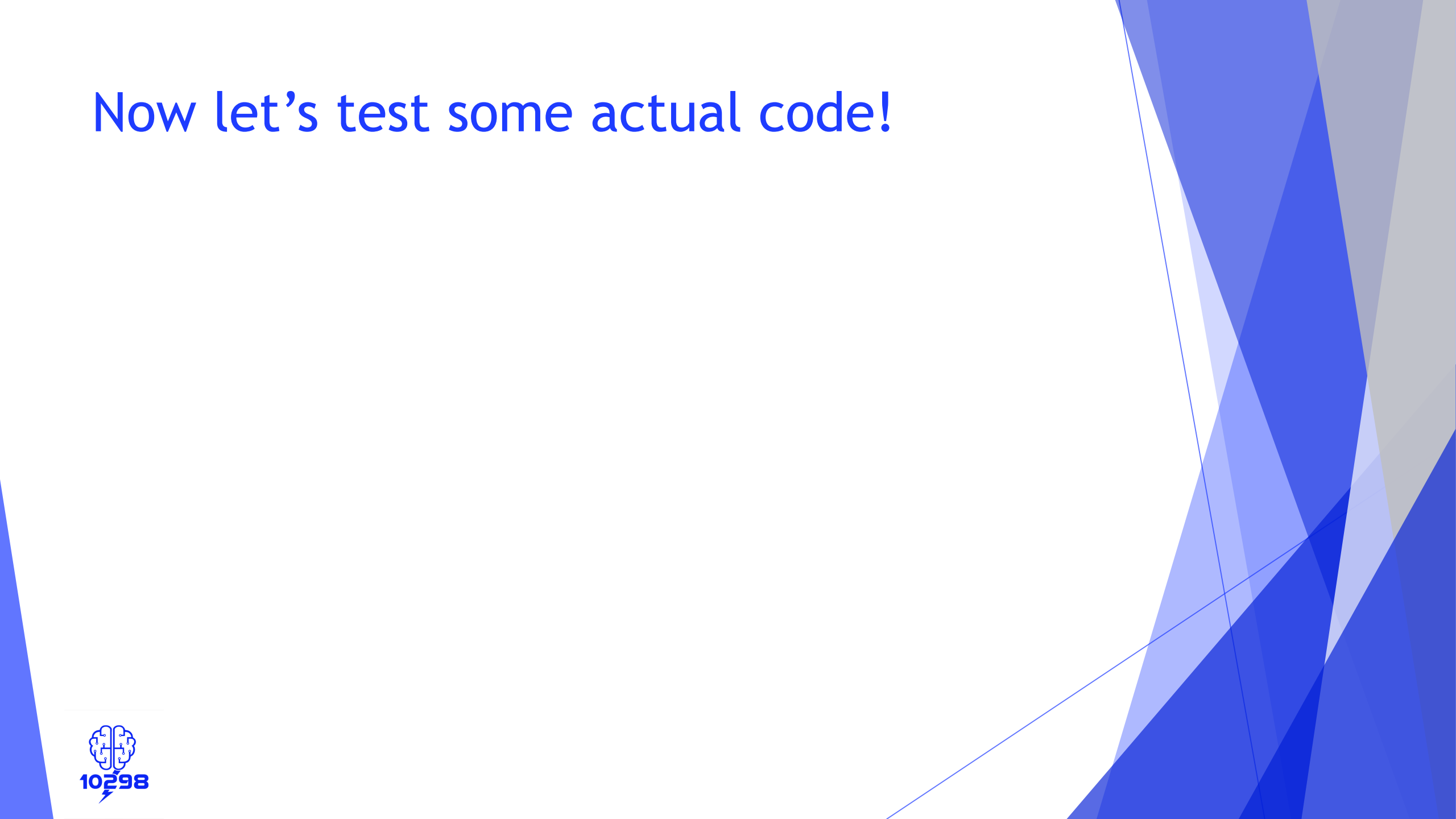
- Motors
  - Have power ranging from -1 to 1
  - Most have encoders
    - Allow for position tracking
- Servos (Standard)
  - Most have 180° of motion
  - Position directly set in code
    - Fixed power
- Servos (Continuous)
  - Same as motor without encoder

10298

Structure of the Code

- Hardware
  - Tells the bot what parts it has
    - Needs to correspond with configuration
- TeleOp
  - Driver-Controlled Period code
  - Loop based
- Autonomous
  - Autonomous Period code
  - Linear based

Now let’s test some actual code!



Robot Parts

- Sensors
  - Gyro
    - Senses amount of turns (driving, arms, etc.)
  - Color
    - Senses color and light
  - Touch
    - Senses touch
  - Distance
    - Senses distance (somewhat accurate)
  - Camera
    - Used with Vuforia

Tips and tricks

- Go slow
- Math is fine but trial and error works too
- Check battery when testing
- If you don’t have a field connect with a team with one
- Expect variance in field, parts
- Have someone else help troubleshoot your code
- Don’t let programming scare you

10298

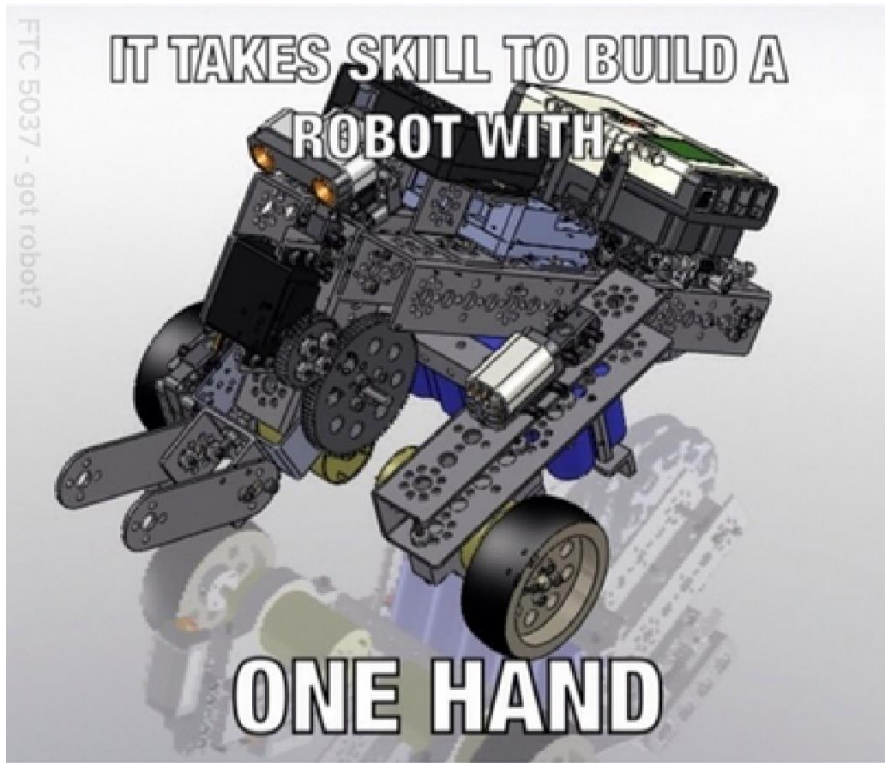
Q&A

Resources

- FIRST: https://www.firstinspires.org/resource-library/ftc/technology-information-and-resources
- Codeacademy: https://www.codecademy.com/learn/learn-java
- Git: https://github.com/ftctechnh/ftc_app/wiki
- Swerve Robotics: https://www.youtube.com/watch?v=7CDK5-m-vQw&index=1&list=PLEuGrYl8iBm7wW9gyxpLDhBJAOWDZid1P
- Brain Stormz Git: https://github.com/FTCTeam10298
- Brain Stormz email: ftc10298@gmail.com
- Brain Stormz website: https://ftc10298.wordpress.com/

10298

Thanks for coming!



Appendix A – Android Studio Tutorial

Hardware Map

- Tells robot what parts it has
- Also used when naming constants
- Tells what config port to look at
- Sets direction, power
- Sets runmode

```
/* Public OpMode members. */
public DcMotor  leftDrive   = null;
public DcMotor  rightDrive  = null;
public DcMotor  leftArm     = null;
public Servo    leftClaw    = null;
public Servo    rightClaw   = null;

public static final double MID_SERVO       =  0.5 ;
public static final double ARM_UP_POWER    =  0.45 ;
public static final double ARM_DOWN_POWER  = -0.45 ;
```

```
// Define and Initialize Motors
leftDrive  = hwMap.get(DcMotor.class, deviceName: "left_drive");
rightDrive = hwMap.get(DcMotor.class, deviceName: "right_drive");
leftArm    = hwMap.get(DcMotor.class, deviceName: "left_arm");
leftDrive.setDirection(DcMotor.Direction.FORWARD); // Set to REVERSE if using AndyMark motors
rightDrive.setDirection(DcMotor.Direction.REVERSE);// Set to FORWARD if using AndyMark motors

// Set all motors to zero power
leftDrive.setPower(0);
rightDrive.setPower(0);
leftArm.setPower(0);

// Set all motors to run without encoders.
// May want to use RUN_USING_ENCODERS if encoders are installed.
leftDrive.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
rightDrive.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
leftArm.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

// Define and initialize ALL installed servos.
leftClaw  = hwMap.get(Servo.class, deviceName: "left_hand");
rightClaw = hwMap.get(Servo.class, deviceName: "right_hand");
leftClaw.setPosition(MID_SERVO);
rightClaw.setPosition(MID_SERVO);
```

10298

TeleOp Tank

- Uses standard tank drive
  - Left stick controls left wheels etc.
  - Power equals joystick push
- Moves Arm
  - Buttons apply power
- Controls Claw
  - Bumpers change offset
  - Offset is position
- Uses Range.*clip*(x, y, z);
  - Keeps values from going out of range
- Sends Telemetry
  - Displayed on driver phone screen

```
public void loop() {
    double left;
    double right;

    // Run wheels in tank mode (note: The joystick goes negative when pushed forwards,
    left = -gamepad1.left_stick_y;
    right = -gamepad1.right_stick_y;

    robot.leftDrive.setPower(left);
    robot.rightDrive.setPower(right);

    // Use gamepad left & right Bumpers to open and close the claw
    if (gamepad1.right_bumper)
        clawOffset += CLAW_SPEED;
    else if (gamepad1.left_bumper)
        clawOffset -= CLAW_SPEED;

    // Move both servos to new position.  Assume servos are mirror image of each other.
    clawOffset = Range.clip(clawOffset, min: -0.5, max: 0.5);
    robot.leftClaw.setPosition(robot.MID_SERVO + clawOffset);
    robot.rightClaw.setPosition(robot.MID_SERVO - clawOffset);

    // Use gamepad buttons to move the arm up (Y) and down (A)
    if (gamepad1.y)
        robot.leftArm.setPower(robot.ARM_UP_POWER);
    else if (gamepad1.a)
        robot.leftArm.setPower(robot.ARM_DOWN_POWER);
    else
        robot.leftArm.setPower(0.0);

    // Send telemetry message to signify robot running;
    telemetry.addData( caption: "claw",   format: "Offset = %.2f", clawOffset);
    telemetry.addData( caption: "left",   format: "%.2f", left);
    telemetry.addData( caption: "right",  format: "%.2f", right);
}
```

10298

Autonomous Drive By Encoder

- Uses a premade function
  - Math already done
- isBusy()

```
static final double     COUNTS_PER_MOTOR_REV    = 1440 ;    // eg: TETRIX Motor Encoder
static final double     DRIVE_GEAR_REDUCTION    = 2.0 ;     // This is < 1.0 if geared UP
static final double     WHEEL_DIAMETER_INCHES   = 4.0 ;     // For figuring circumference
static final double     COUNTS_PER_INCH         = (COUNTS_PER_MOTOR_REV * DRIVE_GEAR_REDUCTION) /
                                                  (WHEEL_DIAMETER_INCHES * 3.1415);
static final double     DRIVE_SPEED             = 0.6;
static final double     TURN_SPEED              = 0.5;
```

```
// Wait for the game to start (driver presses PLAY)
waitForStart();

// Step through each leg of the path,
// Note: Reverse movement is obtained by setting a negative distance (not speed)
encoderDrive(DRIVE_SPEED,   leftInches: 48,   rightInches: 48,   timeoutS: 5.0);  // S1: Forward 47 Inches with 5 Sec timeout
encoderDrive(TURN_SPEED,    leftInches: 12,   rightInches: -12,  timeoutS: 4.0);  // S2: Turn Right 12 Inches with 4 Sec timeout
encoderDrive(DRIVE_SPEED,  leftInches: -24,  rightInches: -24,  timeoutS: 4.0);  // S3: Reverse 24 Inches with 4 Sec timeout

robot.leftClaw.setPosition(1.0);            // S4: Stop and close the claw.
robot.rightClaw.setPosition(0.0);
sleep( milliseconds: 1000);      // pause for servos to move

telemetry.addData( caption: "Path",  value: "Complete");
telemetry.update();
```

10298

Autonomous Drive by Encoder

```
public void encoderDrive(double speed,
                         double leftInches, double rightInches,
                         double timeoutS) {
    int newLeftTarget;
    int newRightTarget;

    // Ensure that the opmode is still active
    if (opModeIsActive()) {

        // Determine new target position, and pass to motor controller
        newLeftTarget = robot.leftDrive.getCurrentPosition() + (int)(leftInches * COUNTS_PER_INCH);
        newRightTarget = robot.rightDrive.getCurrentPosition() + (int)(rightInches * COUNTS_PER_INCH);
        robot.leftDrive.setTargetPosition(newLeftTarget);
        robot.rightDrive.setTargetPosition(newRightTarget);

        // Turn On RUN_TO_POSITION
        robot.leftDrive.setMode(DcMotor.RunMode.RUN_TO_POSITION);
        robot.rightDrive.setMode(DcMotor.RunMode.RUN_TO_POSITION);

        // reset the timeout time and start motion.
        runtime.reset();
        robot.leftDrive.setPower(Math.abs(speed));
        robot.rightDrive.setPower(Math.abs(speed));
```

10298

Autonomous Drive by Encoder

```
// keep looping while we are still active, and there is time left, and both motors are running.
// Note: We use (isBusy() && isBusy()) in the loop test, which means that when EITHER motor hits
// its target position, the motion will stop.  This is "safer" in the event that the robot will
// always end the motion as soon as possible.
// However, if you require that BOTH motors have finished their moves before the robot continues
// onto the next step, use (isBusy() || isBusy()) in the loop test.
while (opModeIsActive() &&
       (runtime.seconds() < timeoutS) &&
       (robot.leftDrive.isBusy() && robot.rightDrive.isBusy())) {

    // Display it for the driver.
    telemetry.addData( caption: "Path1",    format: "Running to %7d :%7d", newLeftTarget,  newRightTarget);
    telemetry.addData( caption: "Path2",    format: "Running at %7d :%7d",
                                robot.leftDrive.getCurrentPosition(),
                                robot.rightDrive.getCurrentPosition());
    telemetry.update();
}

// Stop all motion;
robot.leftDrive.setPower(0);
robot.rightDrive.setPower(0);

// Turn off RUN_TO_POSITION
robot.leftDrive.setMode(DcMotor.RunMode.RUN_USING_ENCODER);
robot.rightDrive.setMode(DcMotor.RunMode.RUN_USING_ENCODER);

//  sleep(250);   // optional pause after each move
```

Appendix B – Other Information

Fancier stuff

- Vuforia
  - Camera (external or on phone) sees pictures
  - Used to tell location and for scoring
- Sensors
  - Used to sense environment in autonomous mode
- Menu
  - Used to shorten program list
- State Machine
  - Used in TeleOp
  - Changes state variable when condition is met