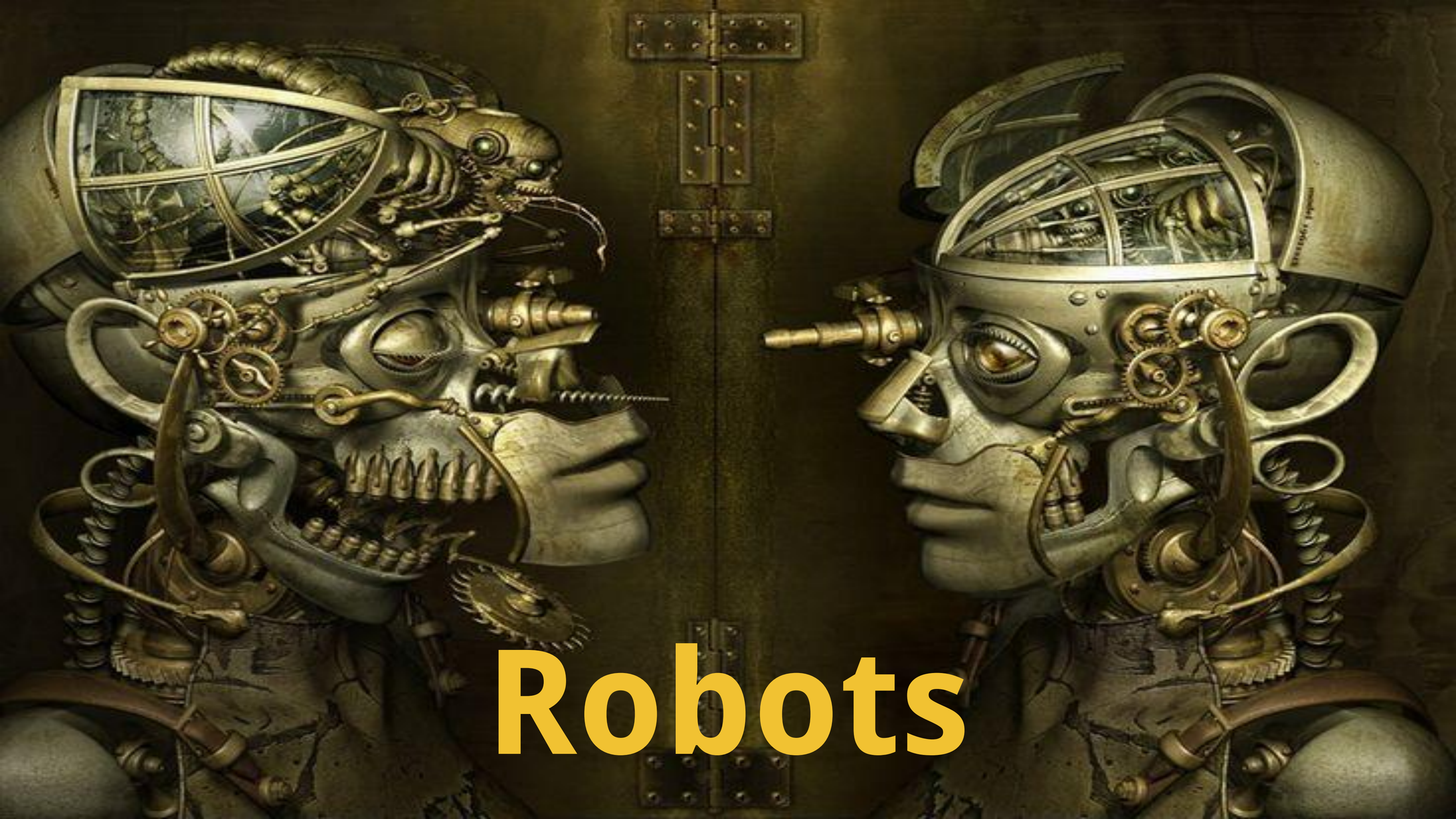
RoboHawk Programing Lab

Day One



Robots

We write code that spins motors

```
motor.set(0.75)
```

Our 2022 competition bot had 13 motors

- SHOOTING
  - Intake wheel
  - Indexer wheel
  - Launch wheel
- CLIMBING
  - Extension
  - Rotation

- DRIVING
  - Front left (x2)
  - Front right (x2)
  - Back left (x2)
  - Back right (x2)

The Romi only has two motors

- Left wheel
- Right wheel

“Just spinning motors” is a bit harder than it sounds

We have to answer questions like …

- How fast should the wheels spin? For how long?
- What does each of the controller buttons do?
- Is there a maximum speed/distance we should spin?

If it doesn’t work right, we have to consider …

- Are things wired up wrong?
- Is the motor burned out?
- Is it our code?

Safety is critical

- The competition robot can weigh well over 100 lbs
- Full-size electric motors spin at 5000 rpm, driving the robot at high speeds
- Gears can be pinchy

Some notes about programming …

A program is a list of instructions for the computer

- The computer executes them in a specific order
- They are spread out in many files

Visual Studio Code will help us:

- Translate our code into something the computer understands
- Run it and send instructions via wifi to the robot

Enough talk! Let’s play!

Java

Variables

```
double desiredSpeed;
boolean isMoving;
String message;

if (!isMoving) {
     desiredSpeed = 10.3;
}

if (desiredSpeed < 15.4) {
     message = "speed up!";
}
```

Classes

```
public OnOffRobot {

    boolean spinning;
    double currentSpeed;
    double maxSpeed;
    double speedIncrement;
    RobotParts parts;

    public void teleopInit() {
        // can use variables in here
    }

    public void teleopPeriodic() {
        // can use variables in here
    }
}
```

Methods (aka functions aka procedures)

```
public double wrapAngle(double angle) {
    if (angle > 180) {
        angle = angle - 360;
    }
    if (angle < -180) {
        angle = angle + 360;
    }
    return angle;
}

frontLeftWheelAngle = wrapAngle(frontLeftWheelAngle);
frontRightWheelAngle = wrapAngle(frontRightWheelAngle);
backRightWheelAngle = wrapAngle(backRightWheelAngle);
backLeftWheelAngle = wrapAngle(backLeftWheelAngle);
```

Objects

```
PIDController turnPid = new PIDController(1.0, 0, 0);
turnPid.enableContinuousInput(-180, 180);
turnPid.setTolerance(5.0);

CANSparkMax motor = new CANSparkMax(1, kBrushless);
motor.restoreFactoryDefaults();
motor.setIdleMode(IdleMode.kBrake);
motor.setInverted(false);
motor.setOpenLoopRampRate(0.5);
motor.setClosedLoopRampRate(0.5);

System.err.println("foo");
```

Modifiers & Annotations

```java
public static final double WHEEL_DIAMETER_INCHES = 2.75591;

public static void main(String [] args) {
}

@Override
public void teleopPeriodic() {
}
```

Comments

```
public static void main(String [] args) {
    // these are for leaving notes to ourselves
}

/**
 * These are for when
 * we have a lot to say
 */
```

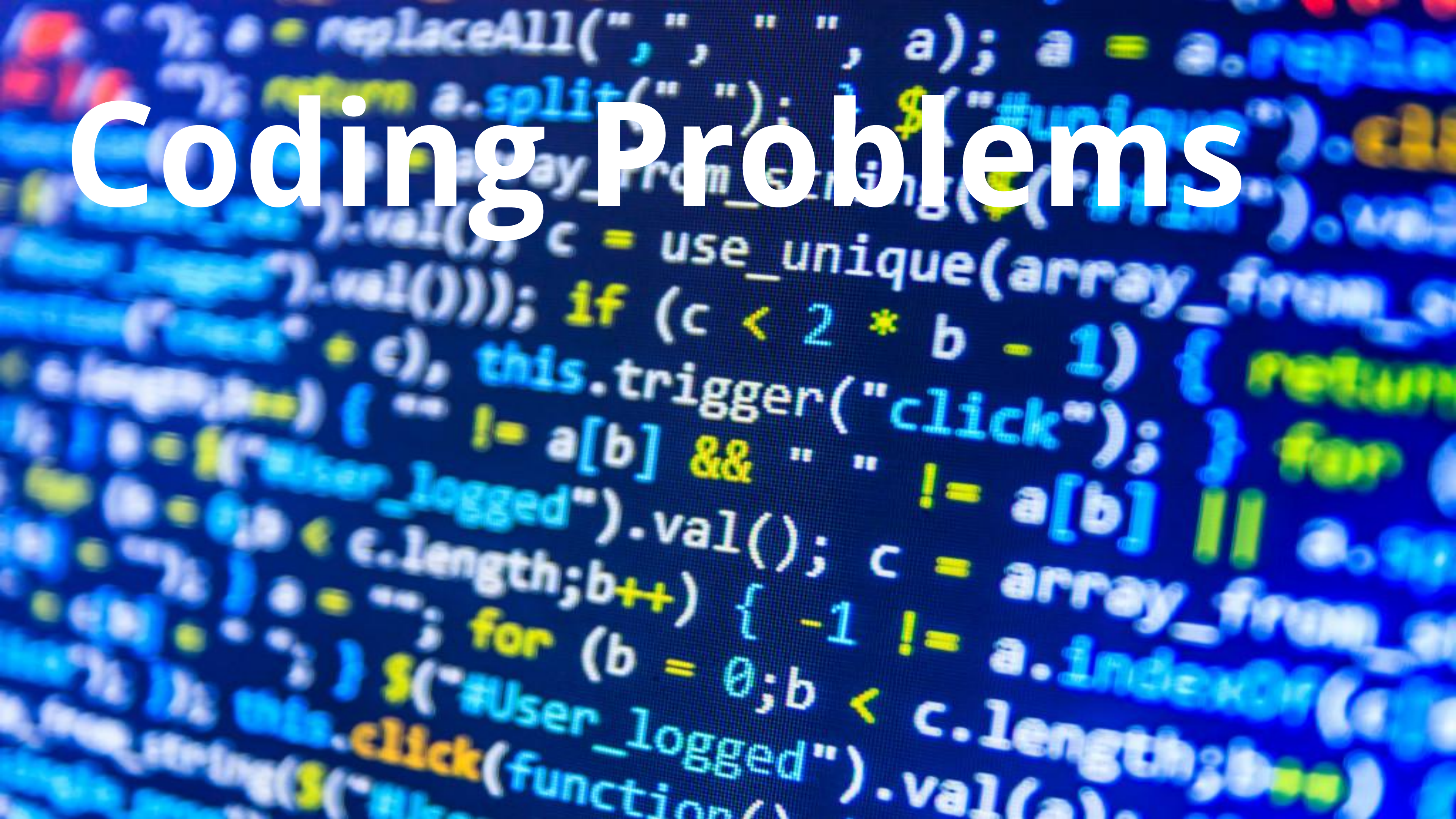
Code files

```
README.md               <-- this has useful stuff in it
src/main/java/frc/robot
     Main.java          <-- this is where it all begins
     RobotParts.java    <-- we will read this together
     EmptyRobot.java    <-- this is what you will work with
     examples/          <-- these are for help if we get stuck
     DualModeDrivingRobot.java
     ManualPositionControlledRobot.java
     OnOffRobot.java
     PIDPositionControlledRobot.java
     ... and so on
```

The “main loop”

- `Main.java` selects which robot class to run
    - `robotInit()` is called once when the robot starts up
    - `robotPeriodic()` is called 50x per second as long as the robot is running
- We will mainly be in "teleop" mode:
    - `teleopInit()` is called once when teleop mode starts
    - `teleopPeriodic()` is called 50x per second while in teleop mode
- You must supply a motor speed in the periodic method (the robot will not “remember”)

Coding Problems



Spin a wheel

- Press a button and the motor spins; release it and it stops
- Variations
  - One button goes forward, one button goes backwards
  - Use a trigger instead of a button for variable speed

Implement a tank drive

- Left joystick spins left wheel forward/backward
- Right joystick spins right wheel forward/backward
- Robot is stopped if you aren't pressing anything
- Variations
  - Square the input values to enable finer control

On/off button

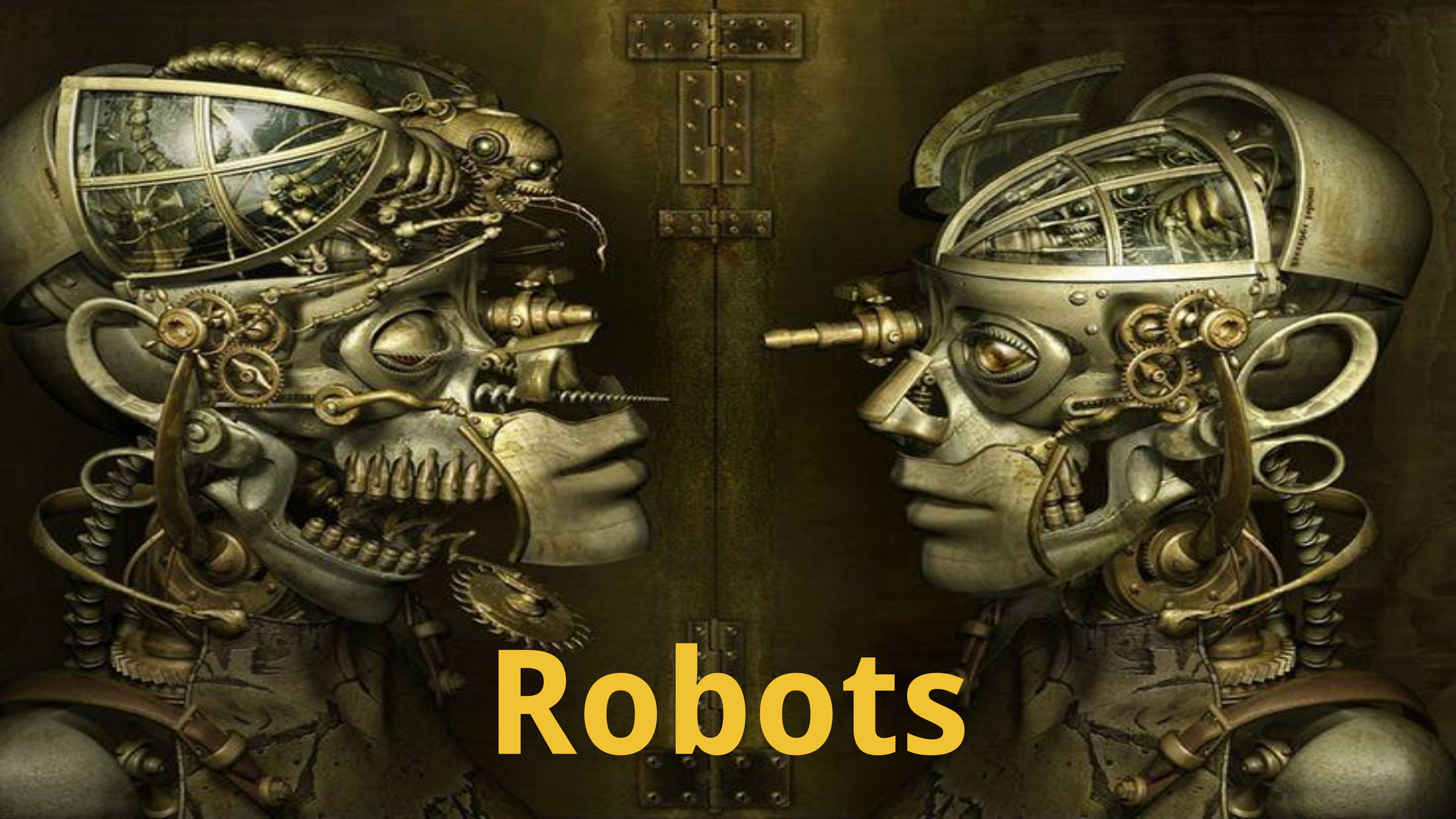
- One button starts the wheel spinning, another one stops it
- Variations
  - Use the same button for on and off
  - Use other buttons to increase/decrease the speed

Hints

```
// spin a motor
parts.leftMotor.set(...)

// read the button state right now
boolean foo = controller.getAButton()

// read the trigger value right now
double foo = controller.getLeftTriggerAxis()

// print something out in the VS Code window
System.err.println("the value of foo is "+foo);

// show something in the simulator window
SmartDashboard.putNumber("Foo", 13.6);
SmartDashboard.putBoolean("Foo", true);
```

RoboHawk Programing Lab

Day Two



Robots

We write code that *reads sensors* and spins motors

```
gyro.getAngle()
```

Our 2022 competition bot had multiple sensors

- System clock
- Wheel encoders (13)
- Limit switches
  - Intake chute
  - Climbing arm

- Controller
  - 4 joystick axes
  - 2 joystick buttons
  - 2 trigger axes
  - 4 letter buttons
  - 2 bumper buttons
  - Back/start
  - POV (aka D-Pad)

The Romi has four built-in sensors

- Clock
- Left wheel encoder
- Right wheel encoder
- Gyro

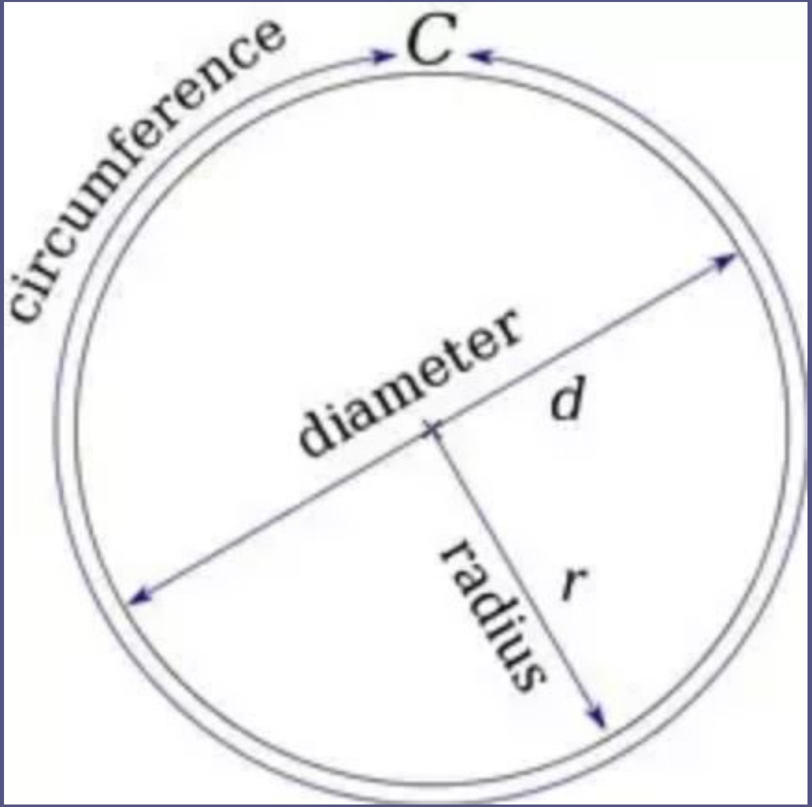
Clock

Measures time in seconds since the robot was turned on.

Useful for timed loops (e.g. "do this for five seconds")

Very important in autonomous mode.

Encoders



Measures distance wheel has travelled in "pulses".

Useful for moving to a specific distance, or spinning at a specific rate.

- 1,440 pulses per revolution
- 2.75591 in. wheel diameter
- (π x d) / ppr inches per pulse

Gyro

Measures heading of robot in degrees.

Must be calibrated each time the robot is turned on.

Useful for turning to a specific heading, or rotating at a specific rate.

Java

API (1/2)

```java
// time in seconds since the robot started
double time = Timer.getFPGATimestamp();

// distance travelled by left and right wheels
// (note that they are different!)
double leftInches = parts.leftEncoder.getDistance();
double rightInches = parts.rightEncoder.getDistance();

// current heading of the robot in degrees (-180 to 180)
double heading = parts.getAngle();
```

API (2/2)

```
// buttons
boolean b = controller.get{xxx}Button();
boolean b = controller.get{xxx}ButtonPressed();
boolean b = controller.get{xxx}Bumper();
boolean b = controller.get{xxx}BumperPressed();


// axes
double d = controller.get{Left/Right}{X/Y}();
double d = controller.get{Left/Right}TriggerAxis();
```

If/Else Ladder

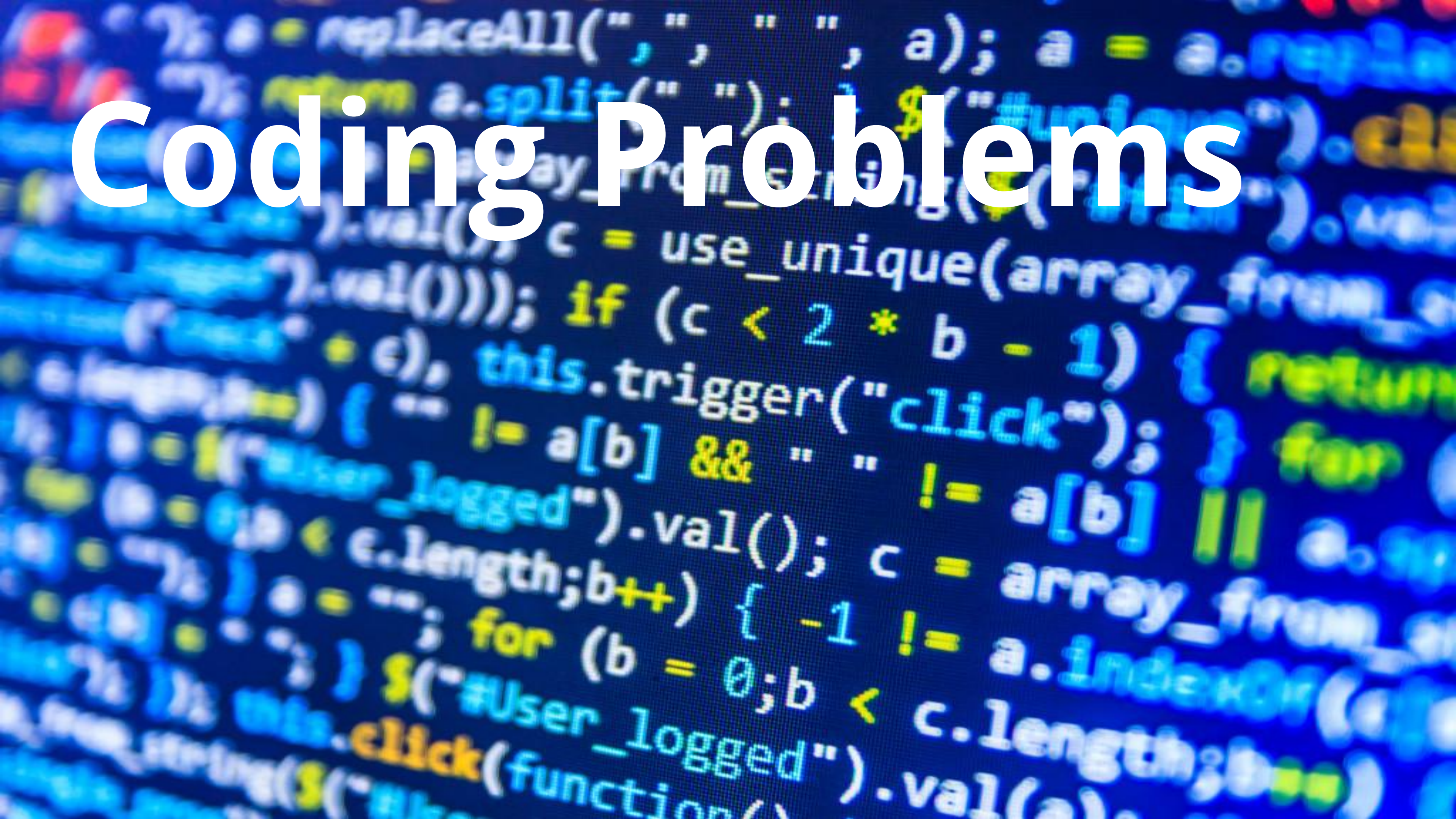
Use this when you want to pick exactly one of a couple of different options.

```
if (x < 10) {
    // do something
} else if (x < 20 {
    // do something else
} else {
    // do something else
}
```

Variables

```
double desiredSpeed;
boolean isMoving;
String message;

if (!isMoving) {
    desiredSpeed = 10.3;
}

if (desiredSpeed < 15.4) {
    message = "speed up!";
}
```

Coding Problems



Timed activity

- When the B button is pressed, drive forward for exactly 3 seconds
- Variations
    - Allow setting max speed from dashboard
    - Allow setting time interval from dashboard

Position control

- When the B button is pressed, drive forward exactly 12 inches
- Variations
  - Allow setting parameters from dashboard
  - Smooth in/out velocity

Heading control

- When the B button is pressed, turn right by 90 degrees
- Variations
    - Allow setting parameters from dashboard
    - Smooth in/out velocity
    - A button turns left by 90 degrees

Autonomous bot

- When autonomous mode is engaged
  - Drive forward 12 inches
  - Turn left 180 degrees
  - Drive forward 12 inches

Pad Drive

- Y button moves forward 12 inches
- A button moves backward 12 inches
- X button turns left 90 degrees
- B button turns right 90 degrees