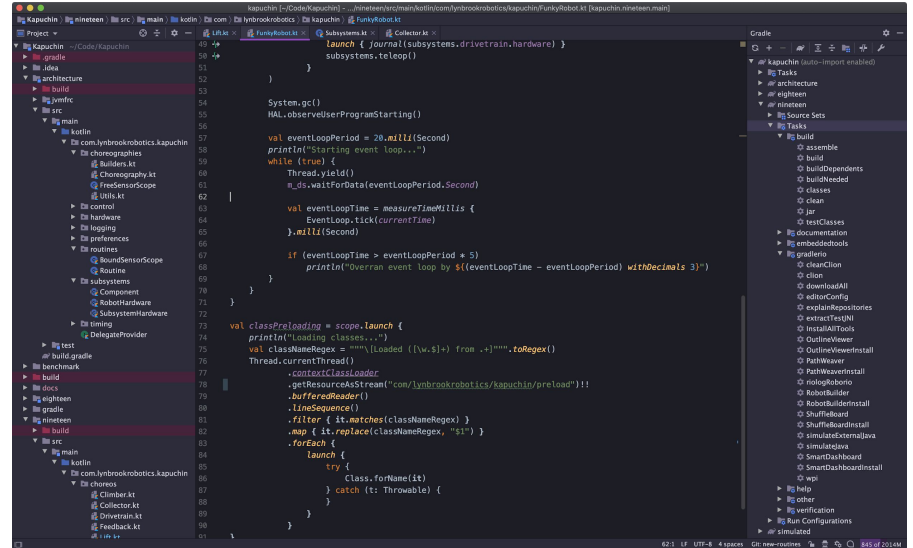


# Software Workshops

Week 1 - 10/11/19

# What do we do?

- Code in Kotlin
- Sensors
- Controlling motors and pneumatics
- Control theory
- Computer vision
- Microcontrollers



```
Project - Kapuchin - /Code/Kapuchin
  - gradle
  - build
  - jvmfrc
  - src
    - main
      - kotlin
        - com.lynbrookrobotics.kapuchin
          - choreographies
          - Builders.kt
          - Choreography.kt
          - FireSensorScope
          - IUI.kt
          - control
          - hardware
          - logging
          - preferences
          - routines
          - SoundSensorScope
          - Routine
          - subsystems
            - Component
            - RoboHardware
            - SubsystemHardware
          - timing
          - DelegateProvider
    - test
  - build.gradle
  - benchmark
  - build
  - docs
  - eighteen
  - gradle
  - nineteen
  - build
  - src
    - main
      - kotlin
        - com.lynbrookrobotics.kapuchin
          - choreos
          - Chamber.kt
          - Collector.kt
          - Drivetrain.kt
          - Feedback.kt
          - ...

Kapuchin (Code/Kapuchin) - .../nineteen/src/main/kotlin/com/lynbrookrobotics/kapuchin/FunkyRobot.kt [Kapuchin/nineteen/main]
  - Subsystem.kt
  - Collector.kt
  - ...
  - launch { journal([subsystems.drivetrain.hardware]
    subsystems.telop())
  }
  System.gc()
  HAL.observeUserProgramStarting()
  val eventLoopPeriod = 20.millis(Second)
  println("Starting event loop...")
  while (true) {
    Thread.yield()
    m_cs.waitForData(eventLoopPeriod.Second)
    val eventLoopTime = measureTimeMillis {
      EventLoop.tick(currentTime)
    }.millis(Second)
    if (eventLoopTime > eventLoopPeriod * 5)
      println("Overran event loop by ${((eventLoopTime - eventLoopPeriod) withDecimals 3)}")
  }
}

val classPreloading = scope.launch {
  println("Loading classes...")
  val classNameRegex = """\((v.[0-9]+) from .+""".toRegex()
  Thread.currentThread()
    .contextClassLoader
    .getResourceAsStream("com/lynbrookrobotics/kapuchin/preload")!!
    .bufferedReader()
    .lineSequence()
    .filter { it.matches(classNameRegex) }
    .map { it.replace(classNameRegex, "$1") }
    .forEach {
      launch {
        try {
          Class.forName(it)
        } catch (t: Throwable) {
        }
      }
    }
  }
}
```

# What will you learn?

- **Kotlin**
- Various tools
  - IntelliJ, Git, Gradle, Command line
- Electronics
  - Sensors, motor controllers, PWM
- Programming concepts
  - Real-time Control
  - Object oriented programming, Functional programming
  - JVM
- Control Theory

<https://tinyurl.com/846software>

**Start Kotlin tutorials when all setup**

Link to tutorial at the bottom of the setup document

Sensor Input



Calculations



Hardware Output

Consistent periodic updates

Our robot is an example of "real-time" software

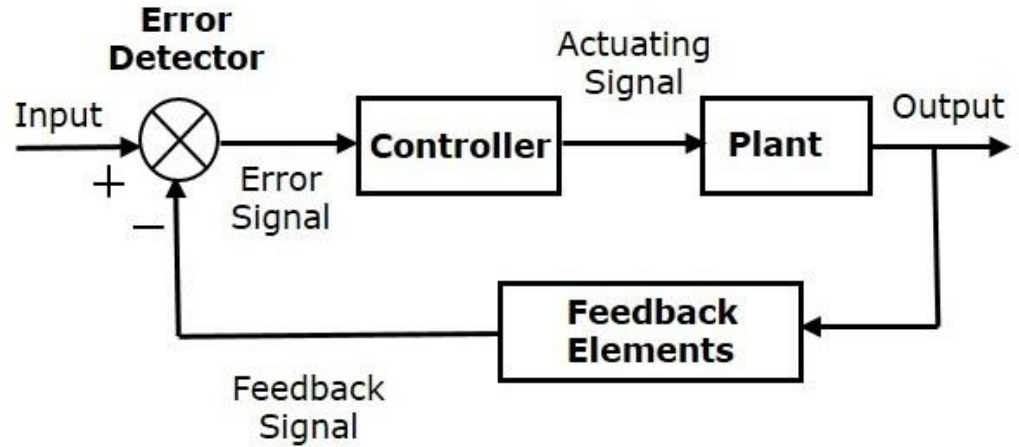
# Sensor Input

- Mechanical
  - Limit switch, hall effect, potentiometer, encoders, gyro
- Driver Input
  - Joystick, Xbox controller, steering wheel
- Camera
  - Limelight
  - Vision system



# Calculations

- Control Theory
  - What do we output to accomplish a goal?



# Hardware Output

- Motors
- Pneumatics
- LEDs





# Kotlin!

If you have a background in...

- Java: <https://tinyurl.com/javakotlin>
- Python: <https://tinyurl.com/pythonkotlin>

# Control Challenges

- [janismac.github.io/ControlChallenges/](https://janismac.github.io/ControlChallenges/)

# Homework

- <https://learngitbranching.js.org>
  - Finish the first 4 levels

<https://tinyurl.com/846softwaresurvey>

# Software Workshops

Week 2 - 10/18/19

Sensor Input



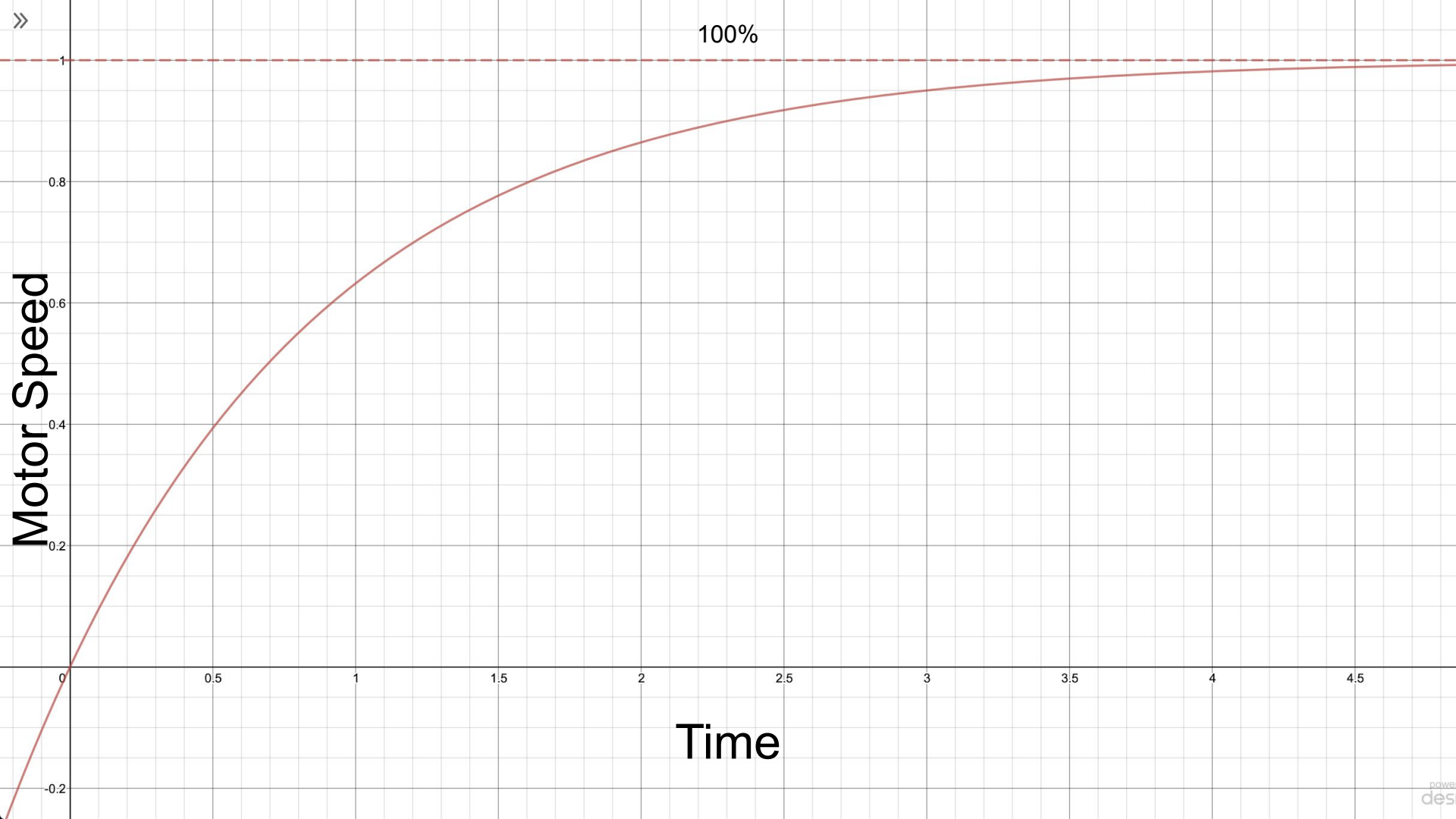
**Calculations**



Hardware  
Output

Consistent periodic updates

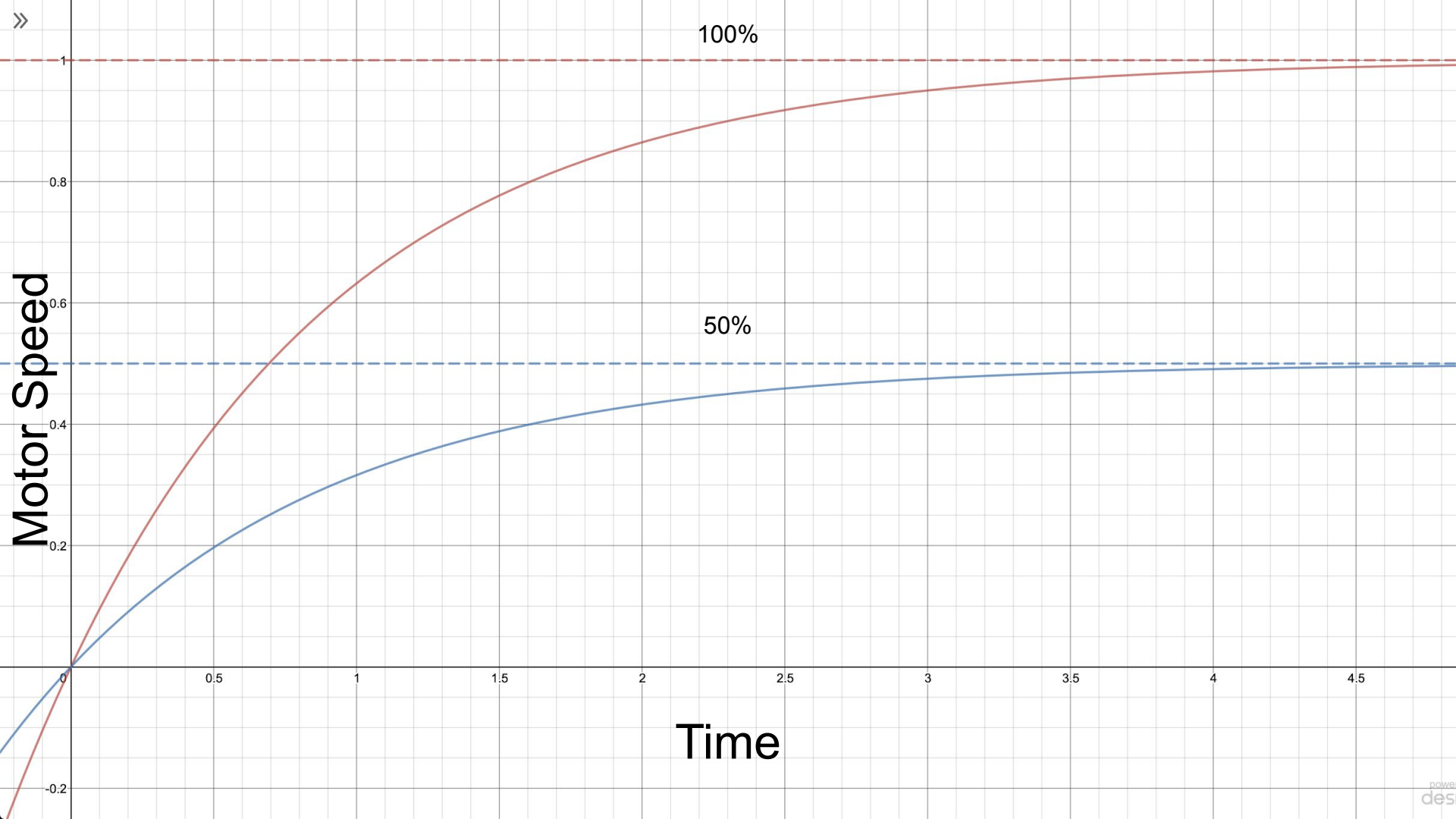
Our robot is an example of  
"real-time" software



Motor Speed

Time

100%



100%

50%

Motor Speed

Time



# Basic Algorithm

If the speed is too slow...

More power

If the speed is too fast...

Slow down

# Basic Algorithm

If the block is too far left...

Move right

If the block is too far right...

Move left

[janismac.github.io/ControlChallenges/](https://janismac.github.io/ControlChallenges/)

# Control Theory

- At least 1 input and output
- Open loop
  - Output calculated using just input
- Closed loop
  - Use feedback
  - Measure the "error" of the output and correct it

# Bang Bang Control

- 2 States
- Most simple algorithm for control
- No tuning
- Examples
  - Thermostat
  - Pump

[janismac.github.io/ControlChallenges/](https://janismac.github.io/ControlChallenges/)

# Proportional Control

- Feedback system
- Error is how far off your block is
  - $\text{Error} = (\text{what you want}) - (\text{what you have})$
- Output is proportional to this error

[janismac.github.io/ControlChallenges/](https://janismac.github.io/ControlChallenges/)



# Proportional + Derivative Control

- Simulating friction
- When the block is going too fast when its approaching the target, we slow it down

[janismac.github.io/ControlChallenges/](https://janismac.github.io/ControlChallenges/)

# Feed Forward

- Sustain a target
- Feed forward is based on prior knowledge, not error

## Bang Bang

- Easy to code
- Fast startup
- Systems with only ON/OFF state

## Proportional + Derivative

- Harder to tune (multiple constants)
- Prevents too much oscillation

# Software Workshops

Week 3 - 11/1/19

# Checklist!

- IntelliJ
- OpenJDK - <https://adoptopenjdk.net>
  - JDK 11
  - Hotspot
  - Check by running "java -version"

```
andrewmin@Andy-Mac  
→ control-workshops-19 git:(week-3) java -version  
openjdk version "11.0.4" 2019-07-16  
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.4+11)  
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.4+11, mixed mode)  
→ control-workshops-19 git:(week-3) █
```

<https://tinyurl.com/846week>

3

# Software Workshops

Week 4 - 11/8/19



# Checklist!

- Install IntelliJ community
- OpenJDK - <https://adoptopenjdk.net>
  - JDK 11
  - Hotspot
  - Check by running "java -version"
- Use label maker by the teachers desk to put your name on your charger!

```
→ control-workshops-19 git:(week-3) java -version
openjdk version "11.0.4" 2019-07-16
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.4+11)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.4+11, mixed mod
```

# <https://tinyurl.com/846wk4>

- Windows:
  - Open file explorer and find the downloaded .zip file
  - Click "Extract All" on the top bar
- Open IntelliJ
  - Click "Open"
  - Find the control-workshops-19 folder you just downloaded
  - Click "Import Gradle Project" on the bottom right popup
    - If you don't see this, you may have opened the wrong folder

# Challenge #1

- Make a function that moves the lift to a certain position
- Parameters: the target position to go to (Length)
- Use proportional control only
- Find base code in Routines.kt
- Your kP (proportional gain) should be in Percent / Length
  - E.g. 50.Percent / 3.Inch
- Uncomment line 24 in FunkyRobot.kt

<http://janismac.github.io/ControlChallenges/>

# Challenge #2

- Modify challenge #1
- Make the routine exit once the lift is close enough to the target
- To make a routine finish, return null from the controller
- Parameters: the target position to go to (Length), the tolerance (Length)

# Challenge #3

- Modify challenge #2
- Add derivative control!

Sensor Input

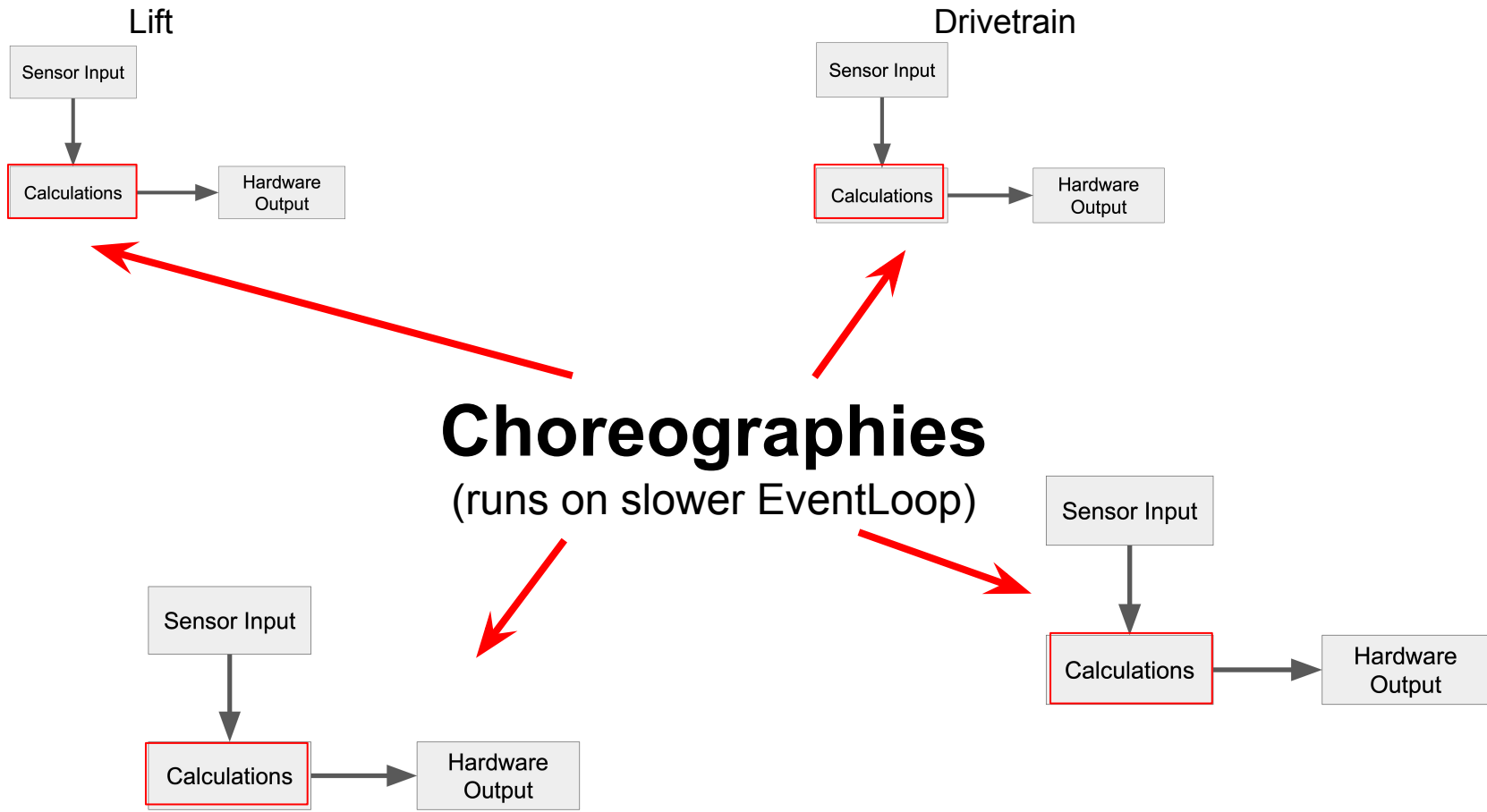


Calculations



Hardware Output

- Each subsystem runs this on a very fast loop



# Choreographies

(runs on slower EventLoop)

# Routines

- Write the calculations for the fast loop
- Sensor input —> **Calculation** —> Hardware Output (only to 1 subsystem!!)



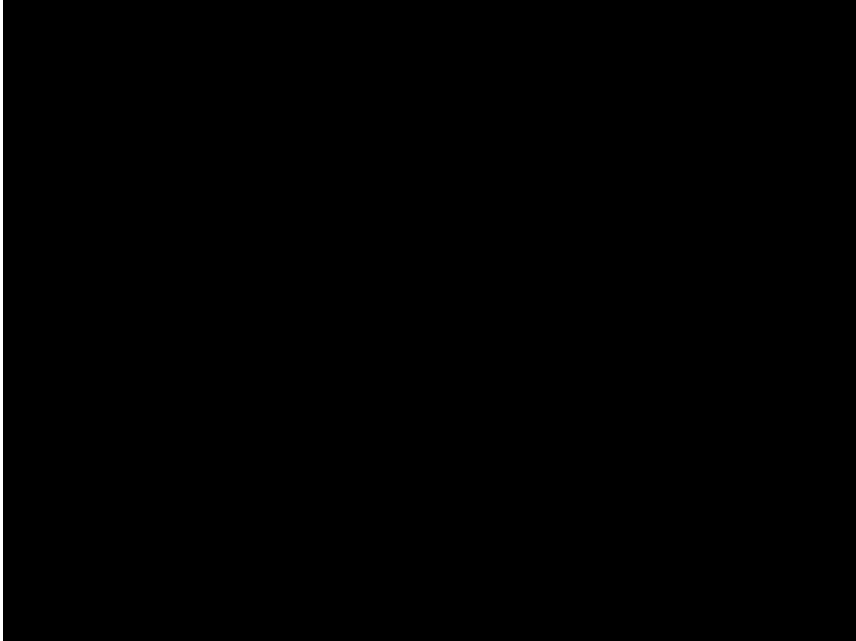
# Choreographies

- Coordinate different subsystems (routines) together
- Run routines sequentially or concurrently

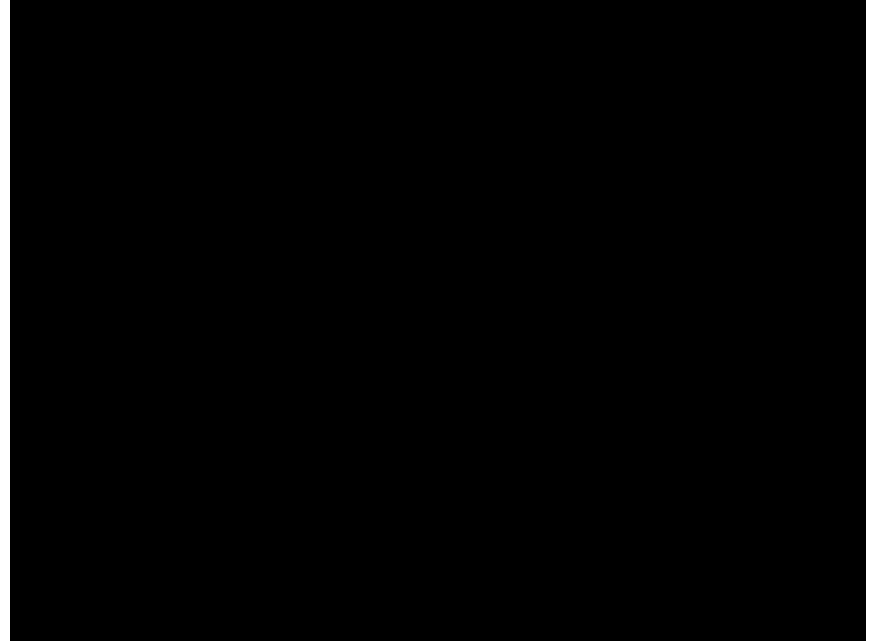
# Challenge #4

- Picking up a hatch panel
- Base code in Choreographies.kt
  
- Comment line 24 in FunkyRobot.kt to disable challenge 1/2/3
- Uncomment lines 27-34 in FunkyRobot.kt
  
- Hint: quickly comment/uncomment multiple lines
  - Highlight the lines you want to comment
    - Mac: command + /
    - Windows: control + /

When the trigger is pressed/held



When the trigger is released



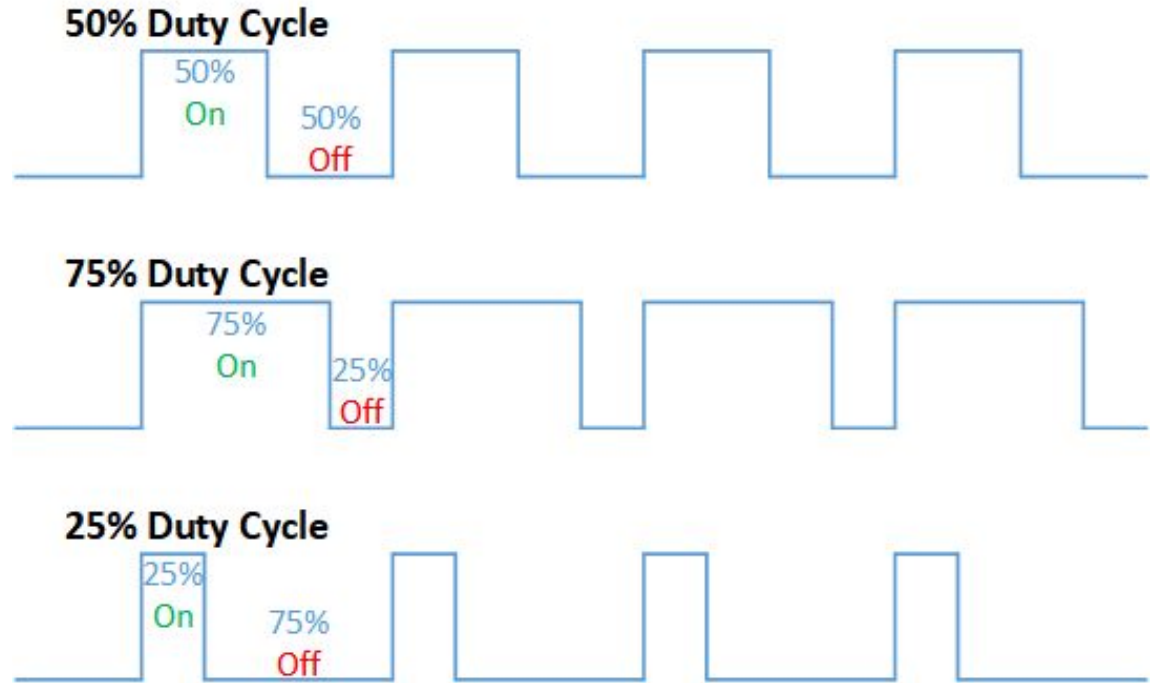
# Software Workshops

Week 5 - 11/15/19

What is PWM?

# PWM (Pulse Width Modulation)

- Control power output
- 0-100% by switching on/off very quickly



What is the CAN bus?

# CAN (Controller Area Network)

- Communicate between different devices
  - Speed controllers, pneumatics, roboRIO
- Send packets of data
- Chain multiple devices together

