WEEK 9/20-9/27

AUTONOMOUS

ROBO RAIDERS

SCARSDALE • 12331

Section 1: What is Autonomous?

ROBO RAIDERS
SCARSDALE · 12331

Autonomous Game Phase

- Self-driving
- 30 seconds (need to optimize!)

Video 3:08 - 4:41

ROBO RAIDERS
SCARSDALE · 12331

Section 2: Setup

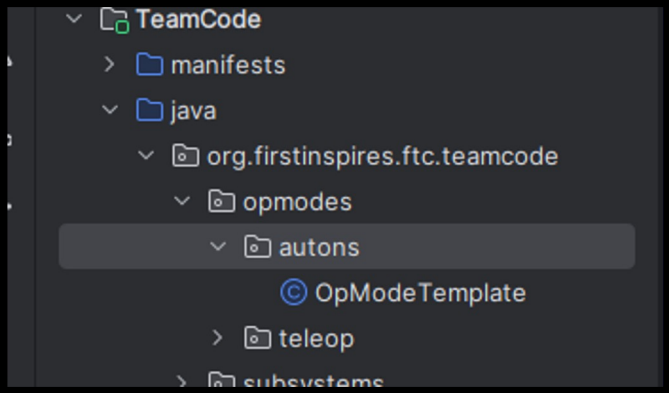
ROBO RAIDERS
SCARSDALE · 12331

GitHub Repo

- Clone the Unit 2 Repo:

```
git clone https://github.com/Scarsdale-Robotics/2024-2025-Training-Program-U2.git
```

- Open it in Android Studio and wait for Gradle to build.
- Create and checkout a new branch called “[your name]-auton-dev”.
- Navigate the file tree to the “autons” folder.



ROBO RAIDERS
SCARSDALE · 12331

New OpMode

- Create a new class called “[Your name]Auto” and copy (or write) the opmode template code into it.

Make sure to rename the class!

StephenAuto.java × OpModeTemplate.java

```
1  package org.firstinspires.ftc.teamcode.opmodes.autons;
2
3  import ...;
7
   Rename usages
8  public class StephenAuto extends LinearOpMode {
9      @Override
10     public void runOpMode() throws InterruptedException {
11         HardwareRobot robot = new HardwareRobot(hardwareMap);
12         DriveSubsystem drive = new DriveSubsystem(
13                 robot.leftFront,
14                 robot.rightFront,
15                 robot.leftBack,
16                 robot.rightBack
17         );
18
19         waitForStart();
20
21
22         // Write your code here
23
24
25     }
26 }
```

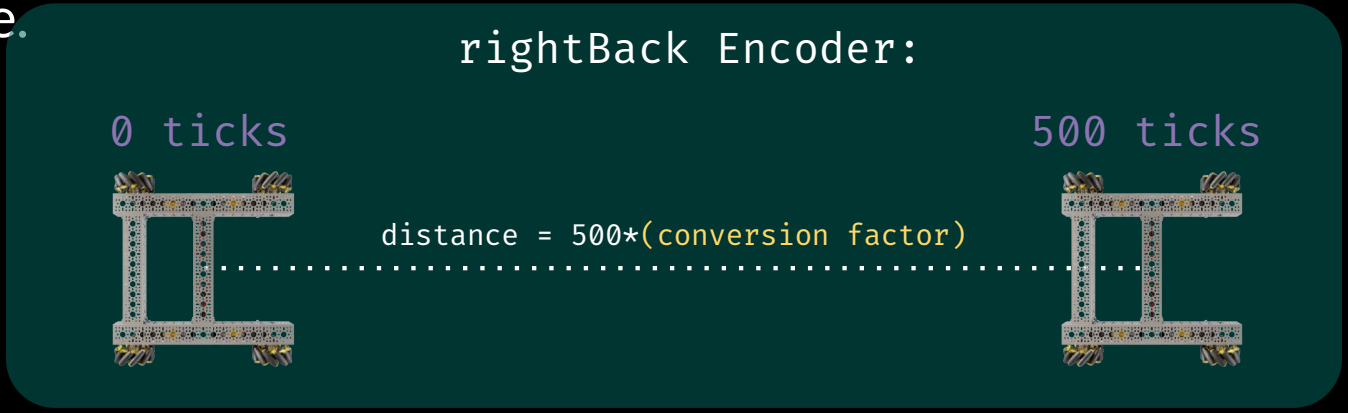
ROBO RAIDERS
SCARSDALE · 12331

Section 3: Sensors

ROBO RAIDERS
SCARSDALE · 12331

Motor Encoders

- To track how far the robot drives, we'll be using **motor encoders**, which measure how much a motor turns in "ticks".
- However, we need to figure out the **conversion** between ticks and distance.



ROBO RAIDERS
SCARSDALE · 12331

Encoder Math

Let's figure out how to convert encoder ticks to distance.

Mecanum Wheel Specs

- Ticks per revolution = 537.7ticks/rev
- Wheel Circumference ≈ 11.87in

→ Distance per revolution = 11.87in/rev

→ Distance per tick = (11.87in/rev) / (537.7ticks/rev) ≈ **0.02208 in/tick**.

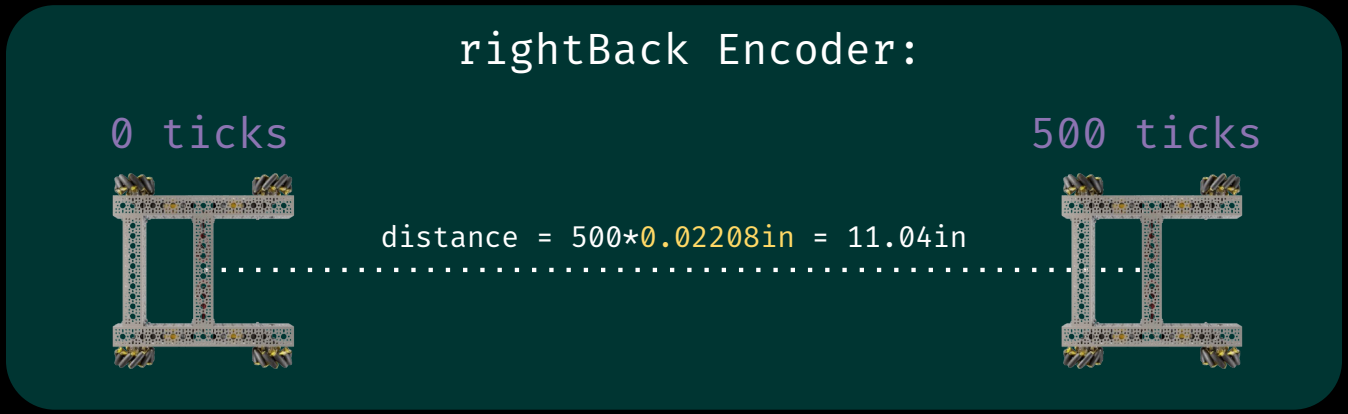
Or, **45.2899 ticks/in**.

Keep in mind that you may need to tweak this value!

ROBO RAIDERS
SCARSDALE · 12331

Encoder Math

So, a distance of 500 ticks is roughly equal to 11.04in!



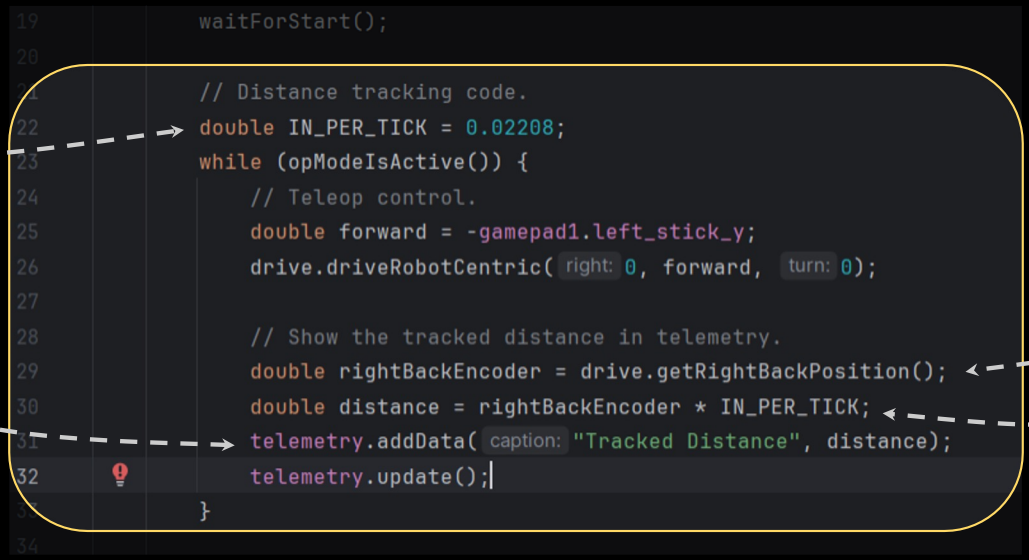
ROBO RAIDERS
SCARSDALE · 12331

Distance Tracking Code

Let’s implement distance tracking! We’ll reuse parts of Teleop code.

Declare the conversion factor outside the loop

Show distance using telemetry



```
19          waitForStart();
20
            // Distance tracking code.
22          double IN_PER_TICK = 0.02208;
23          while (opModeIsActive()) {
24              // Teleop control.
25              double forward = -gamepad1.left_stick_y;
26              drive.driveRobotCentric( right: 0, forward,  turn: 0);
27
28              // Show the tracked distance in telemetry.
29              double rightBackEncoder = drive.getRightBackPosition();
30              double distance = rightBackEncoder * IN_PER_TICK;
31              telemetry.addData( caption: "Tracked Distance", distance);
32              telemetry.update();
            }
34
```

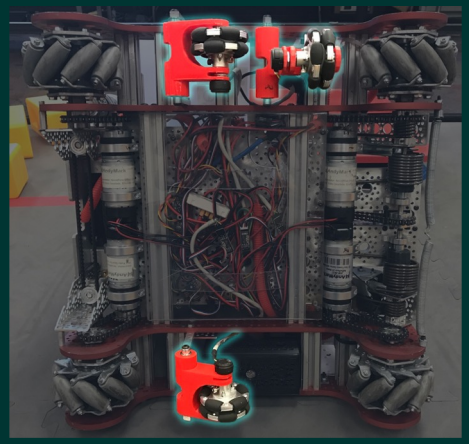
Get encoder position

Calculate distance

ROBO RAIDERS
SCARSDALE · 12331

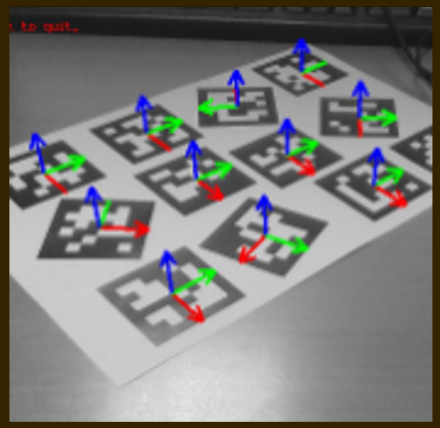
Other Sensors We Have

- Dead wheel odometry



- Super precise
- Easy to code
- May drift over time
- Odometry tutorial

- AprilTag Pose Estimation



- Super accurate
- Harder to code/tune
- Won't always have access to an AprilTag

This is Computer Vision -- not this week!

ROBO RAIDERS
SCARSDALE · 12331

Section 3: Intro To Control Theory

ROBO RAIDERS
SCARSDALE · 12331

What is Control Theory?

- Imagine you are coding a self-driving car to park itself:

- ⭐ **Essential Question**: How do you park accurately and consistently?
  - The field of control theory tries to solve this problem.

ROBO RAIDERS
SCARSDALE · 12331

Section 3.1: Open-Loop Control

ROBO RAIDERS
SCARSDALE · 12331

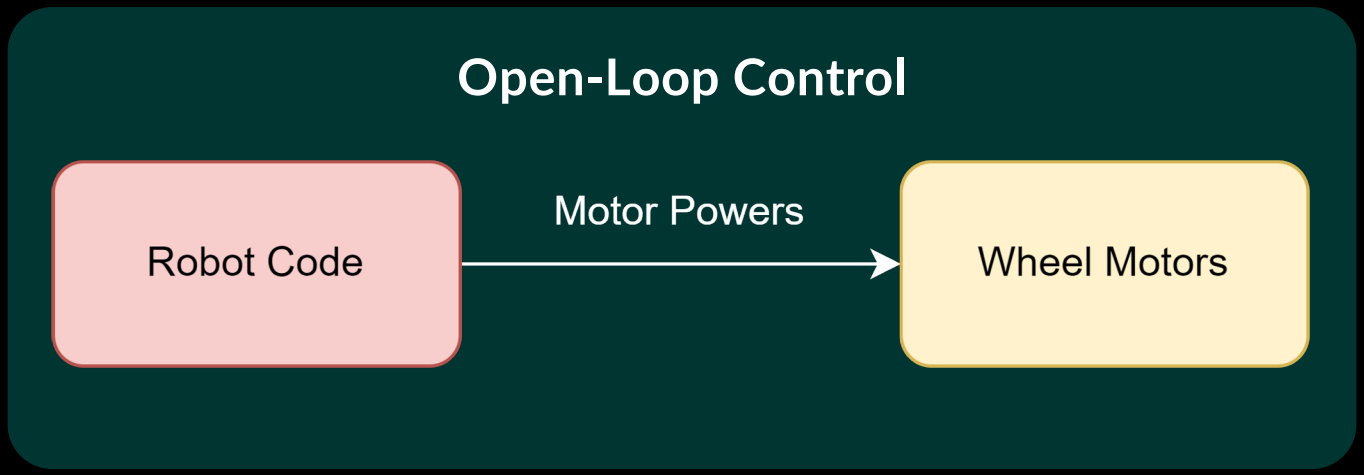
Open-Loop Control

- Assuming we know the travel distance, one solution is to drive at a **certain speed** for a **certain duration**.

ROBO RAIDERS
SCARSDALE · 12331

Open-Loop Control

- This is called “open-loop control” because there’s no sensor feedback loop.



ROBO RAIDERS
SCARSDALE · 12331

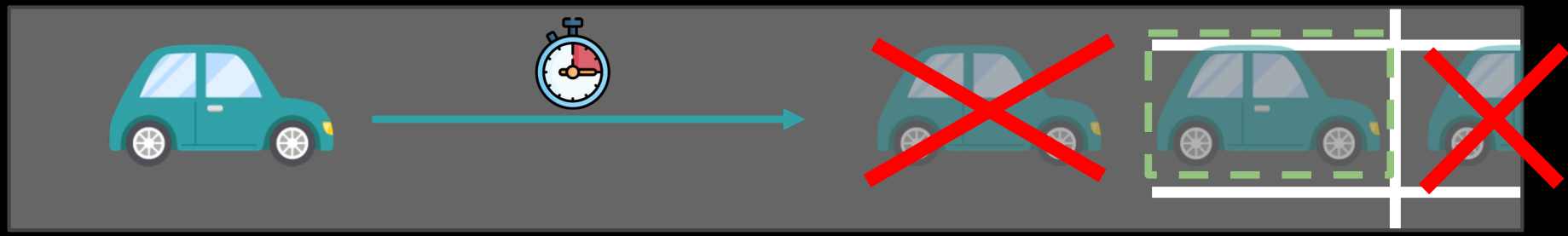
Open-Loop Control

- Let's try implementing this idea on the robot!

ROBO RAIDERS
SCARSDALE · 12331

Open-Loop Control Is Not Great

- Open-loop control is often inconsistent.
  - Varying motor speeds, external interference, low voltage, etc.



How do we fix this?

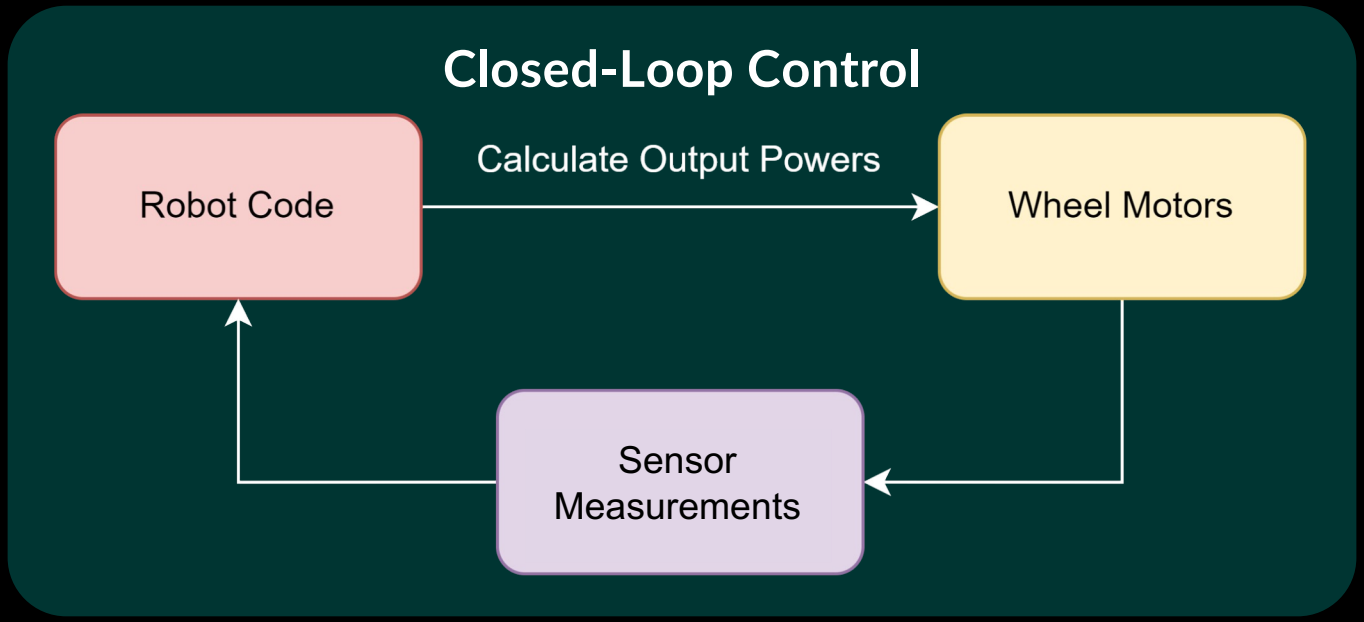
ROBO RAIDERS
SCARSDALE · 12331

Section 3.2: Closed-Loop Control

ROBO RAIDERS
SCARSDALE · 12331

Closed-Loop Control

- What if we use sensor feedback to correct the robot as it drives?



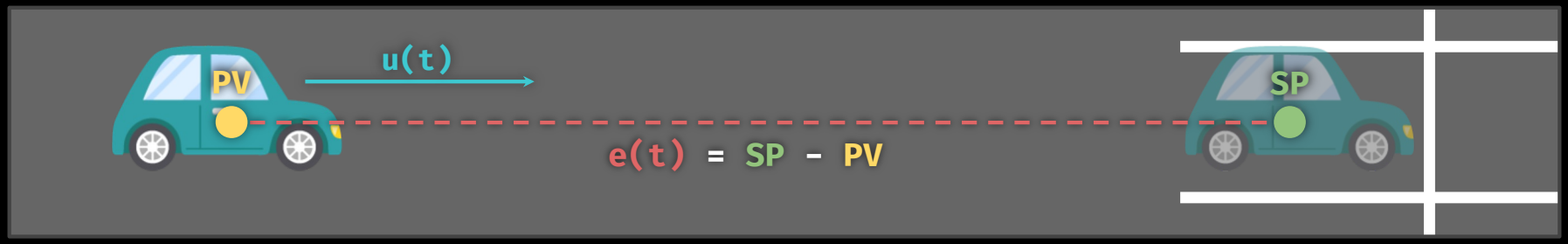
ROBO RAIDERS
SCARSDALE · 12331

Closed-Loop Control

Let's first define some terms

- **Setpoint** or **SP** = target position
- **Process Value** or **PV** = car position
- **Error** or **e(t)** = **SP - PV** = required displacement
- **Control output** or **u(t)** = drive power

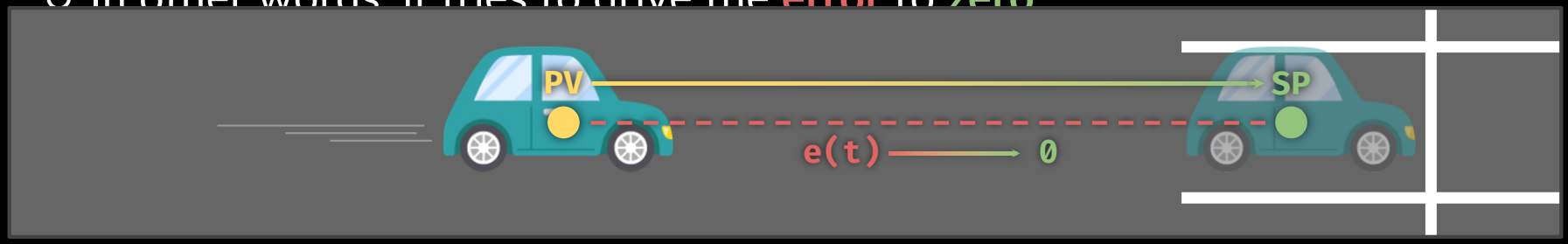
(t) means "at the current time"!



ROBO RAIDERS
SCARSDALE · 12331

Closed-Loop Controllers

- A **closed-loop controller** tries to drive the **process value** to the **setpoint**.
  - In other words, it tries to drive the **error** to **zero**.



Let's see some examples of these controllers!

ROBO RAIDERS
SCARSDALE · 12331

Section 3.2.1: Bang-Bang Controller

ROBO RAIDERS
SCARSDALE · 12331

The Bang-Bang Controller

The car must follow these rules:

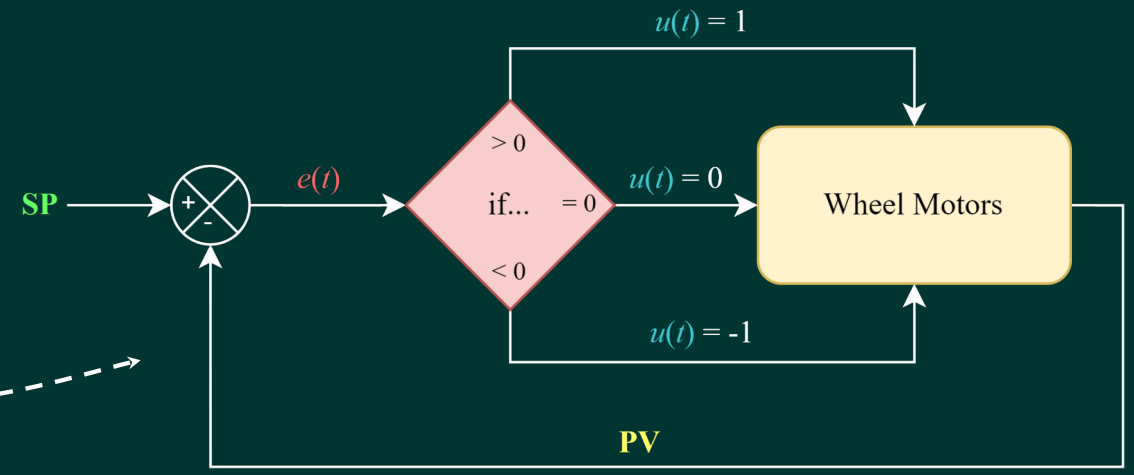
- If error is **positive**, drive forward.
- If error is **negative**, drive in reverse.
- If error is **zero**, stop.

These all say the same thing!

Control Law

$$u(t) = \begin{cases} 1, & e(t) > 0 \\ 0, & e(t) = 0 \\ -1, & e(t) < 0 \end{cases}$$

Flowchart



ROBO RAIDERS
SCARSDALE · 12331

The Bang-Bang Controller

- Let's implement it in code!

Get error

Control law conditions

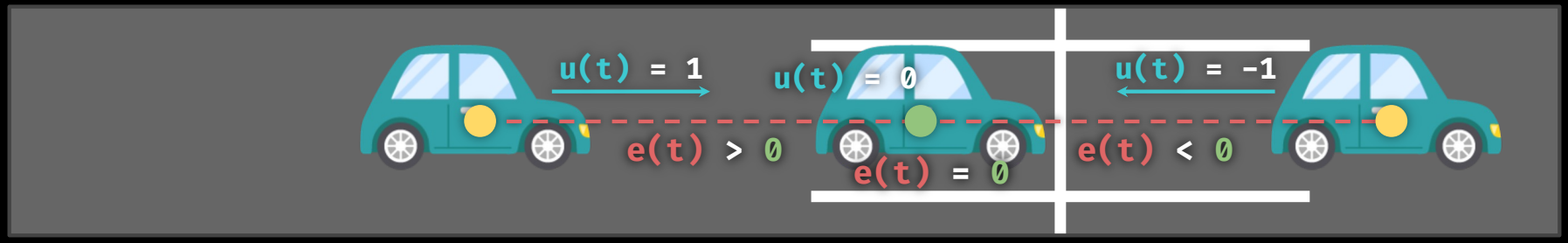
Telemetry is always nice!

```
waitForStart();

// Bang-Bang Controller.
double setpoint = 1000;
while (opModeIsActive()) {
    // Calculate error.
    double encoderPosition = drive.getRightBackPosition();
    double error = setpoint - encoderPosition;

    // Control law.
    if (error > 0) {
        drive.driveRobotCentric( right: 0,  forward: 1,  turn: 0);
    }
    if (error == 0) {
        drive.driveRobotCentric( right: 0,  forward: 0,  turn: 0);
    }
    if (error < 0) {
        drive.driveRobotCentric( right: 0,  forward: -1,  turn: 0);
    }

    // Show the encoder position in telemetry.
    telemetry.addData( caption: "Encoder Position", encoderPosition);
    telemetry.update();
}
```

ROBO RAIDERS
SCARSDALE · 12331

The Bang-Bang Controller

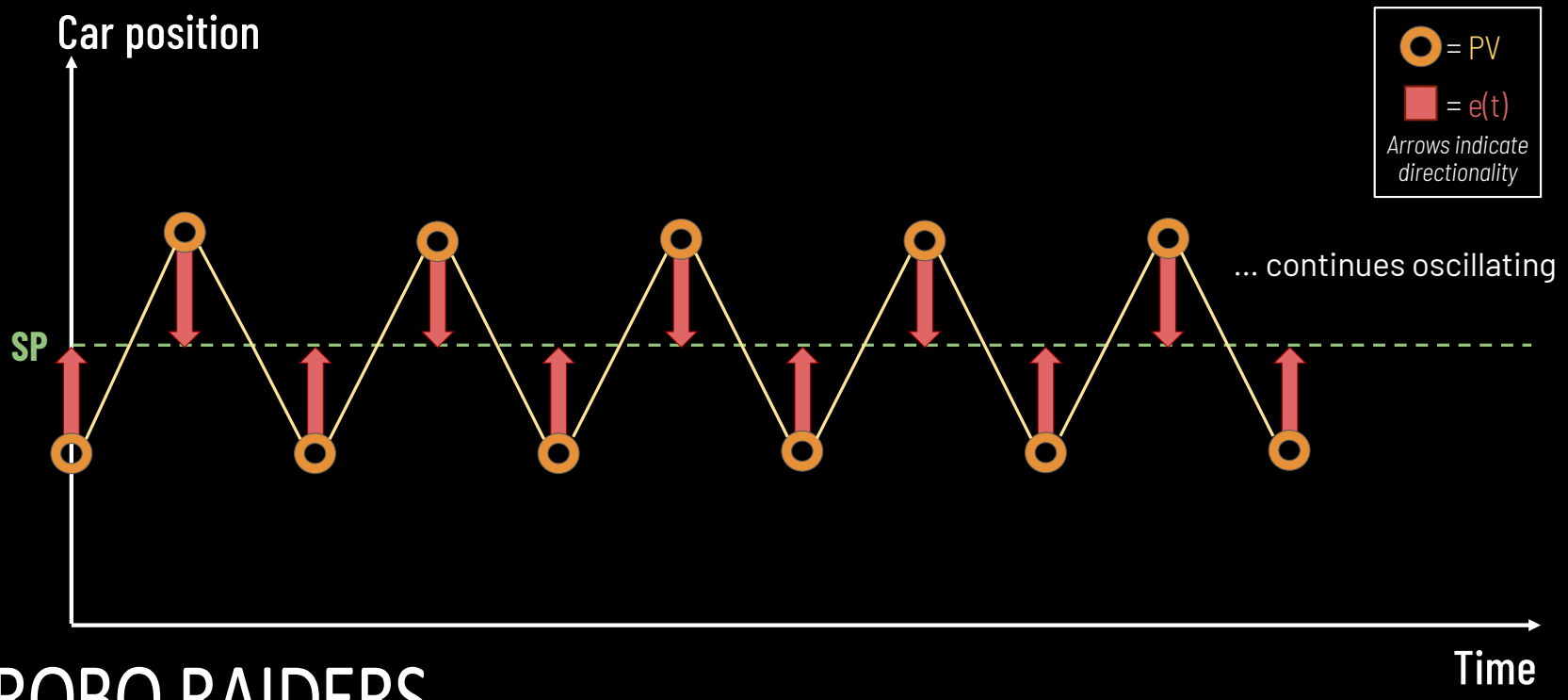
How does this perform?



Oscillations!

ROBO RAIDERS
SCARSDALE · 12331

The Bang-Bang Controller



ROBO RAIDERS
SCARSDALE · 12331

credit

Section 3.2.2: P Controller

ROBO RAIDERS
SCARSDALE · 12331

The P Controller

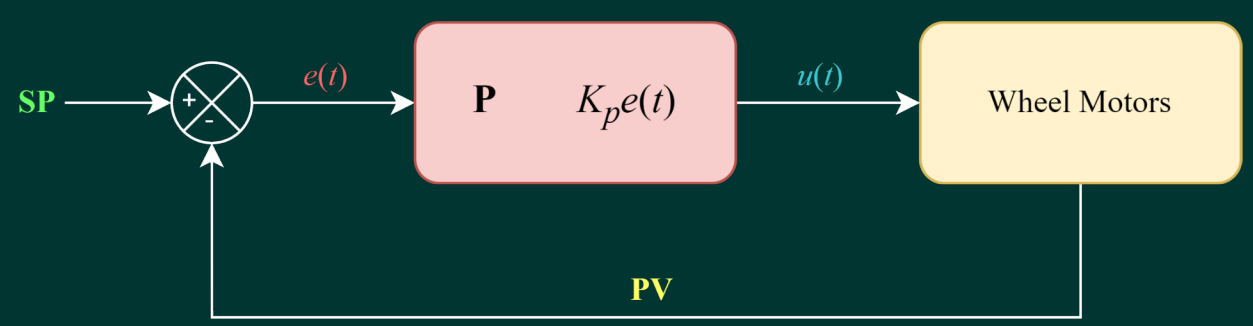
- Normally, you would **slow down** as you approach the space.
- Let's make the power output **proportional** to error (hence the **P**).

Control Law

$$u(t) = K_p e(t)$$

K_p is a constant

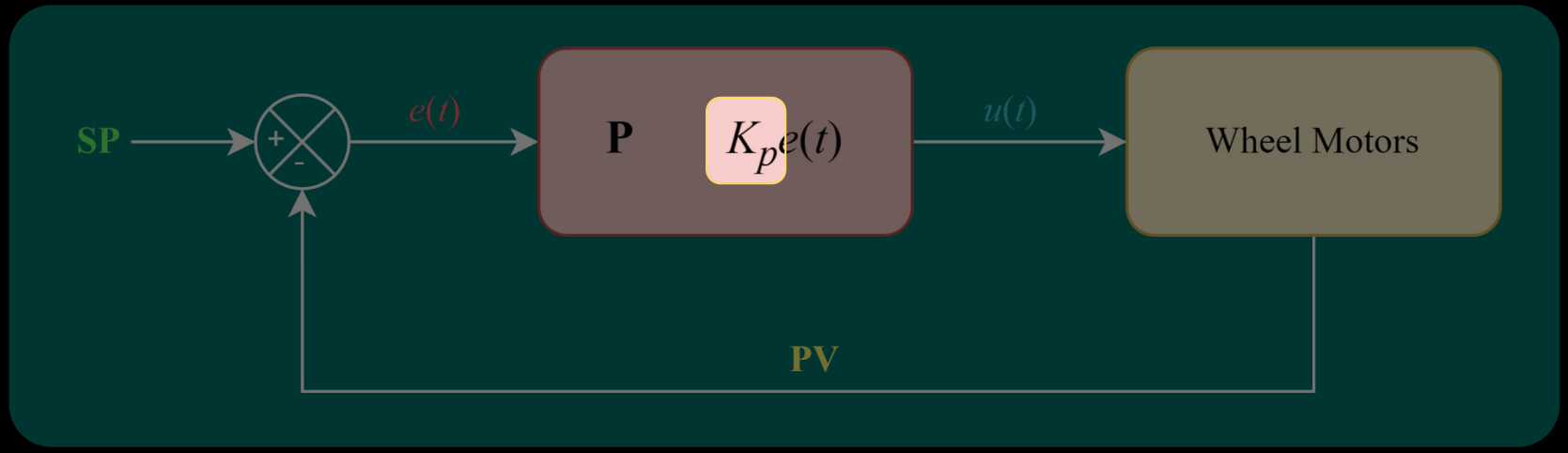
Flowchart



ROBO RAIDERS
SCARSDALE · 12331

The P Controller

- K_p is a term you need to **tune**, or adjust through trial and error.
  - Low K_p → sluggish movement
  - High K_p → bang-bang



ROBO RAIDERS
SCARSDALE · 12331

The P Controller

- Let's implement code for it!

Get error

Multiply by K_P

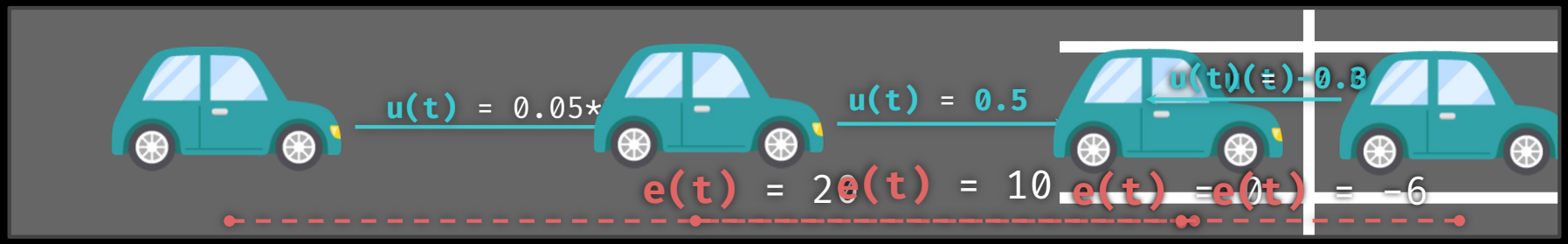
Telemetry!

```
        waitForStart();

        // P Controller.
        double setpoint = 1000;
        double kP = 0.01;
        while (opModeIsActive()) {
            // Calculate error.
            double encoderPosition = drive.getRightBackPosition();
            double error = setpoint - encoderPosition;

            // Control law.
            double u_t = kP * error;
            drive.driveRobotCentric( right: 0, u_t,  turn: 0);

            // Show info in telemetry.
            telemetry.addData( caption: "Encoder Position", encoderPosition);
            telemetry.addData( caption: "u_t", u_t);
            telemetry.update();
        }
```

ROBO RAIDERS
SCARSDALE · 12331

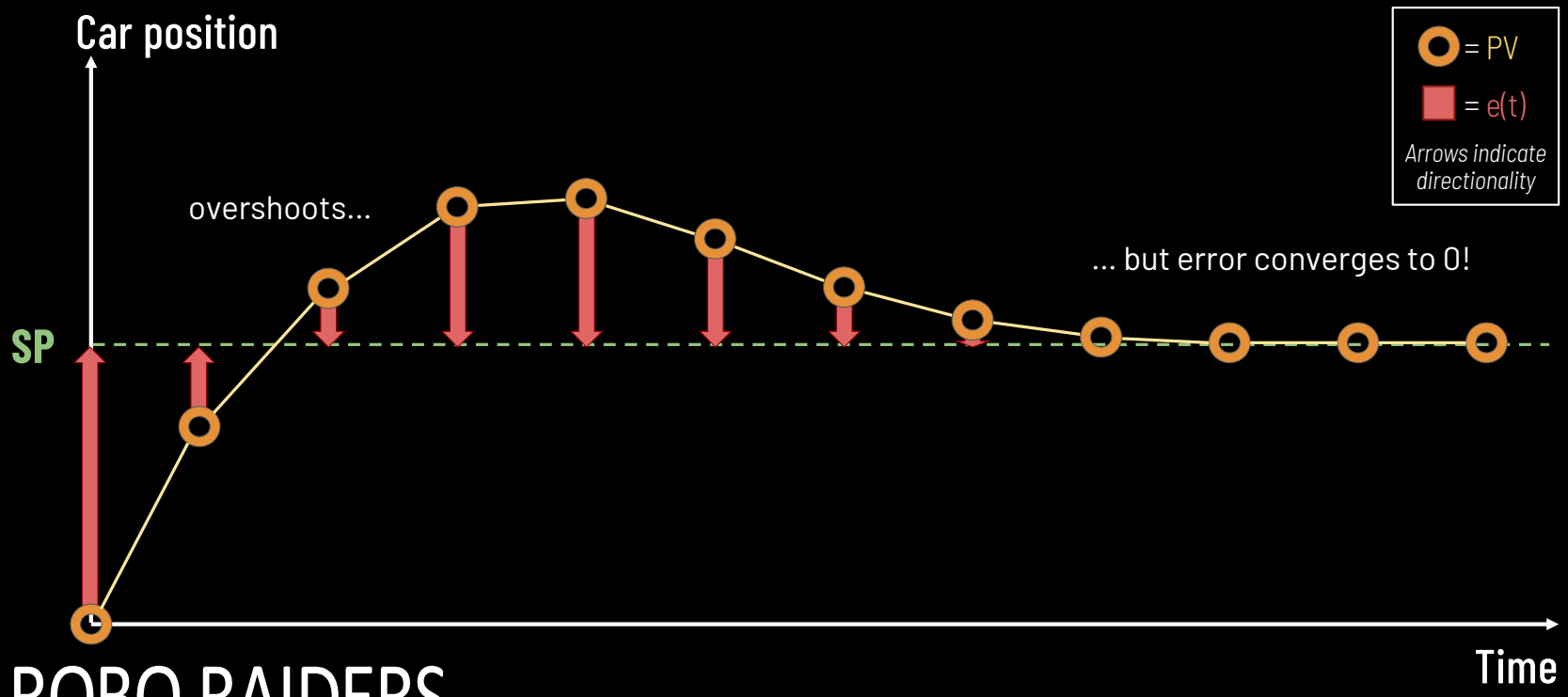
The P Controller

- How does it perform? (Assume Kp = 0.05)



ROBO RAIDERS
SCARSDALE · 12331

The P Controller



ROBO RAIDERS
SCARSDALE · 12331

credit

Section 3.2.3: PD Controller

ROBO RAIDERS
SCARSDALE · 12331

The PD Controller

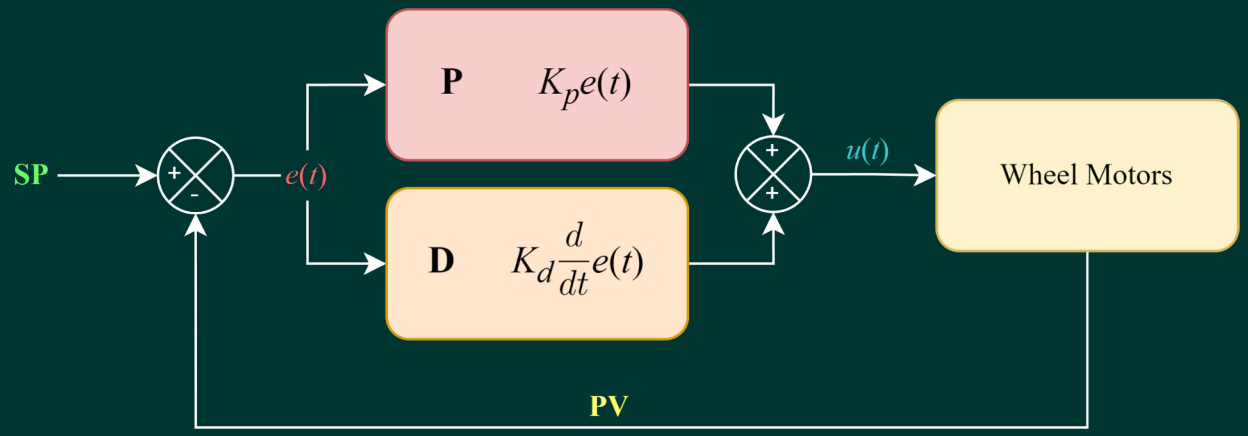
- Normally, you would gradually **press the brakes** to not overshoot the space.
- We can model this behavior by decreasing the power based on the robot's velocity, or the Derivative of error.

Control Law

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t)$$

K_p and K_d are constants

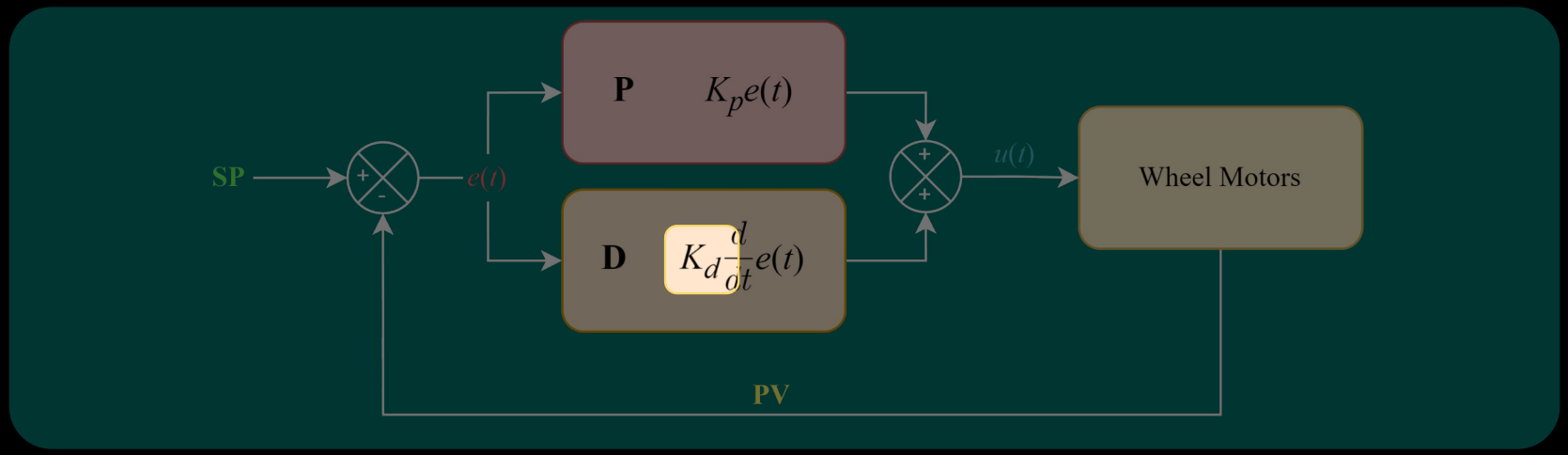
Flowchart



ROBO RAIDERS
SCARSDALE · 12331

The PD Controller

- You will also need to tune K_d.
  - Low K_d → still overshooting
  - High K_d → stuttering or jittering



ROBO RAIDERS
SCARSDALE · 12331

The PD Controller

- How can we calculate the derivative?

Slope formula

$$\frac{de(t)}{dt} \approx \frac{e(t) - e(t - \Delta t)}{\Delta t}$$

- Intuitive
- Simpler (last error only)
- Prone to high-frequency noise

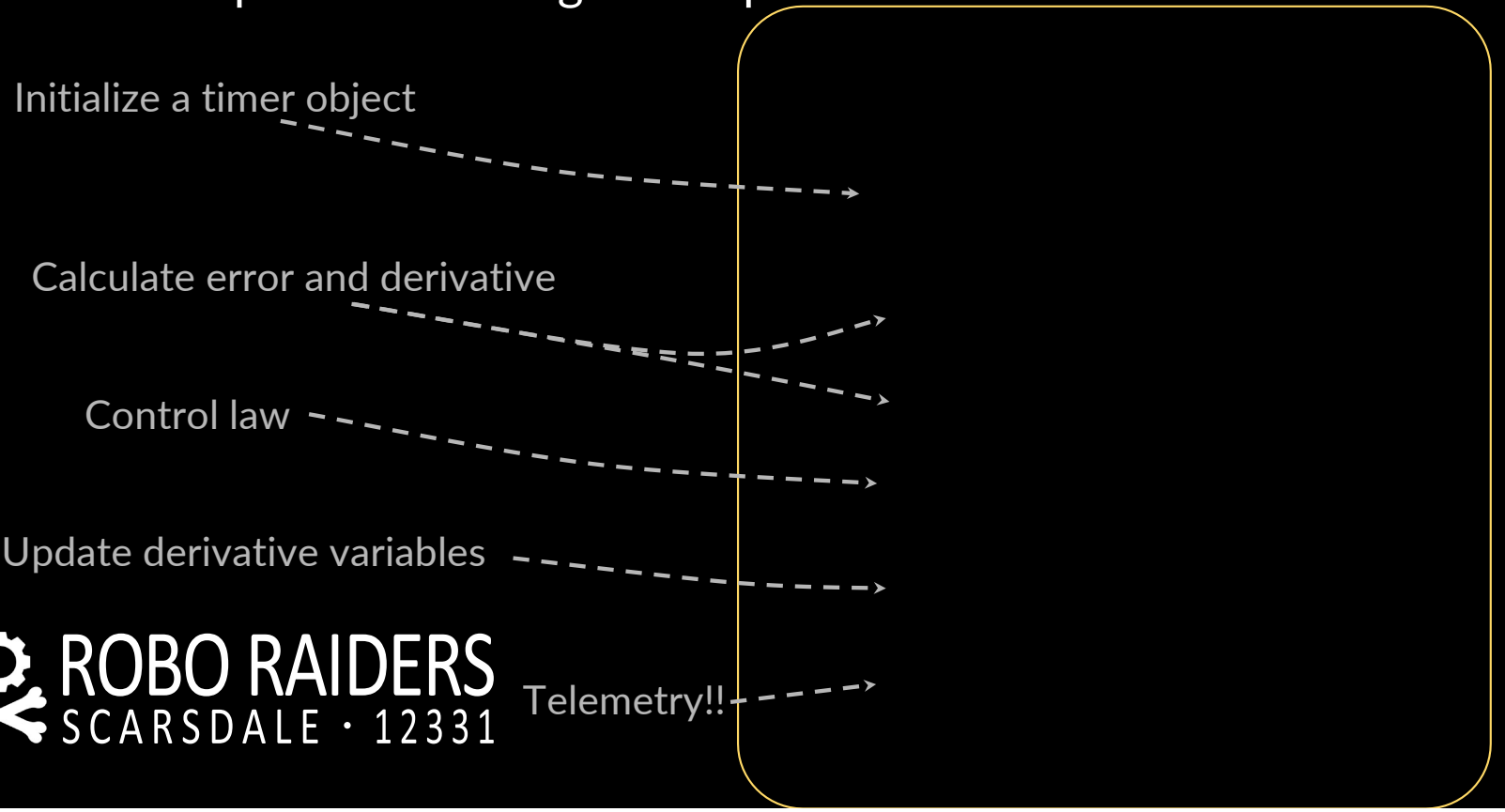
Five-Point Stencil (Wikipedia)

$$\frac{de(t)}{dt} \approx \frac{-e(t) + 8e(t - \Delta t) - 8e(t - 3\Delta t) + e(t - 4\Delta t)}{12\Delta t}$$

- Less intuitive
- More complex (keep track of last four errors and deltaTimes)
- Very robust against noise

The PD Controller

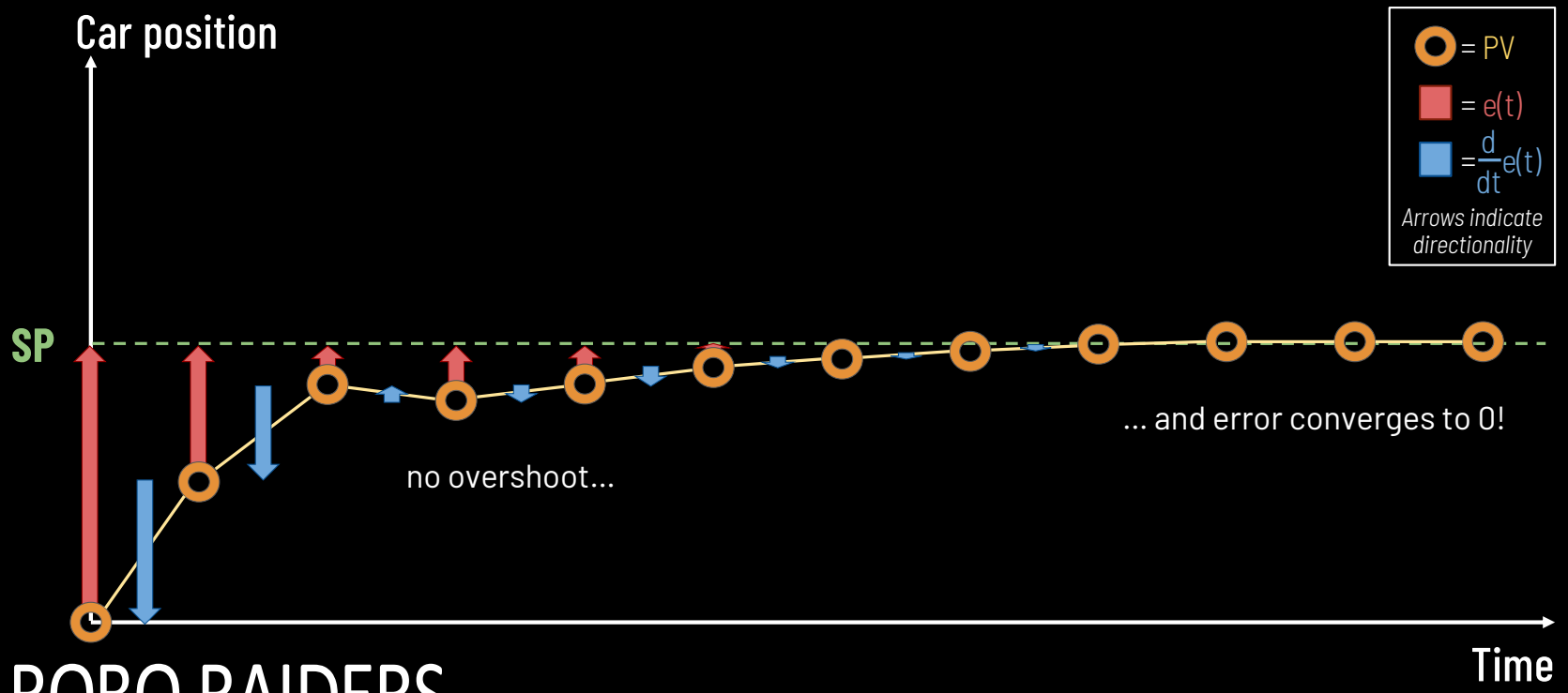
- Let's implement it using the slope formula!



$$\frac{de(t)}{dt} \approx \frac{e(t) - e(t - \Delta t)}{\Delta t}$$

ROBO RAIDERS
SCARSDALE · 12331

The PD Controller



ROBO RAIDERS
SCARSDALE · 12331

credit

Section 3.2.4: PID Controller

ROBO RAIDERS
SCARSDALE · 12331

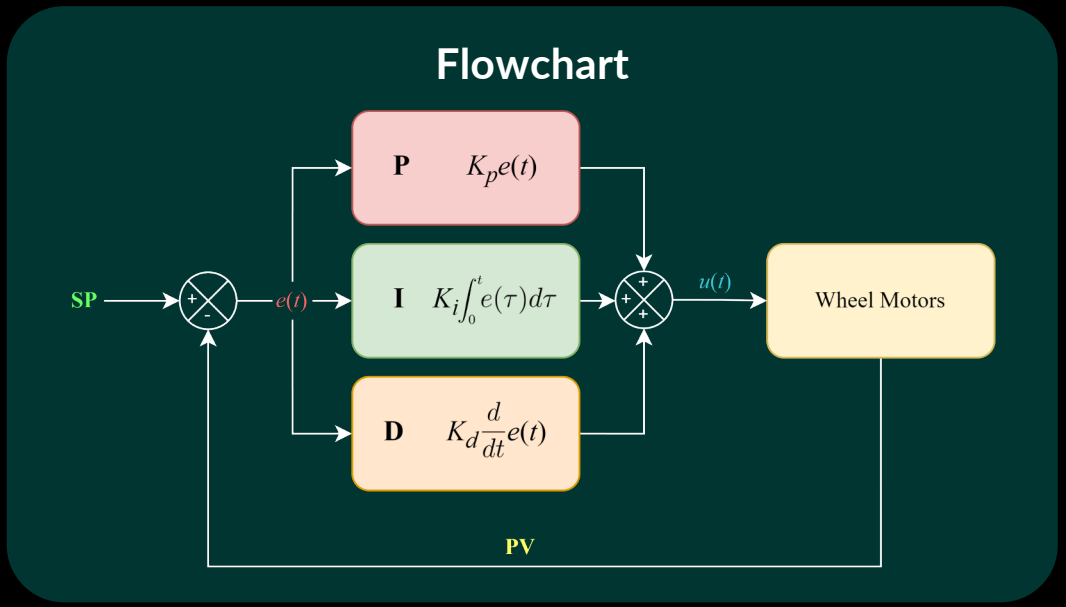
The PID Controller

- *You do **not** need a PID controller for this lab!*
- For systems such as an arm/lift, where the motor has to **counteract gravity or friction**, a PD controller might not output enough force to reach the setpoint.
- An **Integrator** accumulates small errors and nudges the system to the setpoint.

Control Law

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

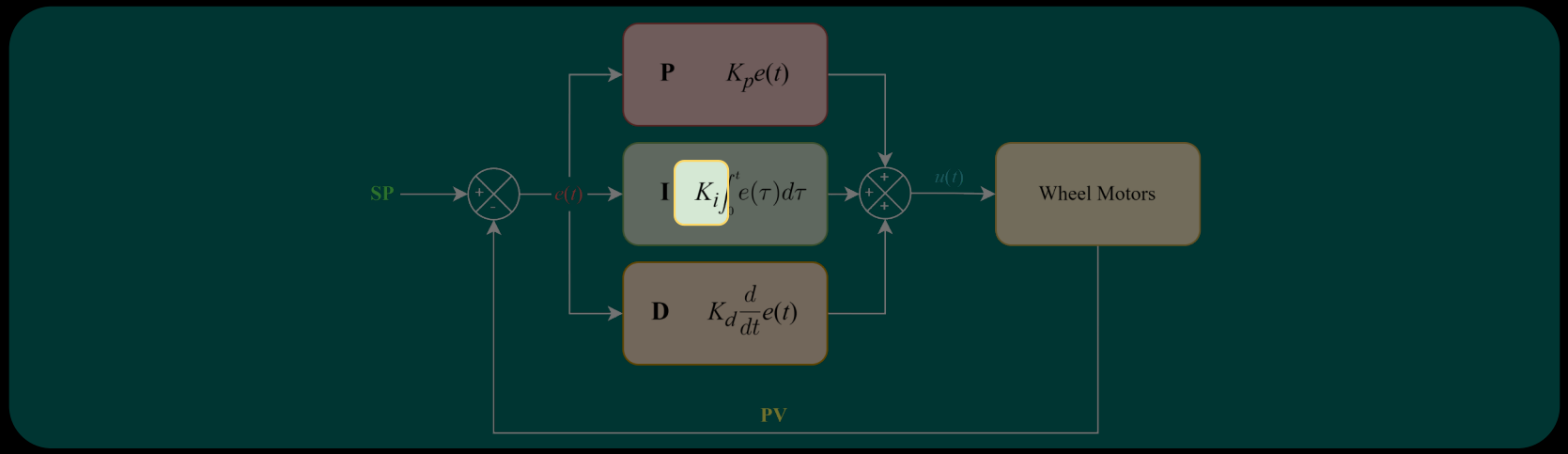
K_p, K_i, and K_d are constants



ROBO RAIDERS
SCARSDALE · 12331

The PID Controller

- Tuning K_i:
  - Low K_i → slow convergence to setpoint
  - High K_i → oscillations that keep getting larger



ROBO RAIDERS
SCARSDALE · 12331

The PID Controller

- Code implementation (Slope formula for derivative)

```java
// PID Controller Using Slope Formula.
double setpoint = 1000;
double kP = 0.01;
double kI = 0;
double kD = 0.001;

// For finding integral.
double integral = 0;

// For finding derivative.
double lastError = 1000;
ElapsedTime runtime = new ElapsedTime( startTime: 0);

while (opModeIsActive()) {
    double deltaTime = runtime.seconds();

    // Calculate error.
    double encoderPosition = drive.getRightBackPosition();
    double error = setpoint - encoderPosition;

    // Update integral.
    integral += deltaTime * error;

    // Calculate derivative.
    double derivative = (error - lastError) / deltaTime;

    // Control law.
    double u_t = kP*error + kI*integral + kD*derivative;
    drive.driveRobotCentric( right: 0, u_t, turn: 0);

    // Update lastError and runtime for next loop.
    lastError = error;
    runtime.reset();

    // Show info in telemetry.
    telemetry.addData( caption: "Encoder Position", encoderPosition);
    telemetry.addData( caption: "P", value: kP * error);
    telemetry.addData( caption: "I", value: kI * integral);
    telemetry.addData( caption: "D", value: kD * derivative);
    telemetry.addData( caption: "u_t", u_t);
    telemetry.update();
}
```

ROBO RAIDERS
SCARSDALE · 12331

Key Takeaways

- **Closed-loop control** is generally preferred over open-loop control.
  - You should still try both for this lab!

For closed-loop control:

- Your objective is to **drive error to zero**.
- You can use **motor encoders** as the sensor, although alternatives exist.
- There are different types of closed-loop controllers, many of which **need manual tuning**, so try to be patient!

☆ **If you ever need help, ask one of us!**

ROBO RAIDERS
SCARSDALE · 12331