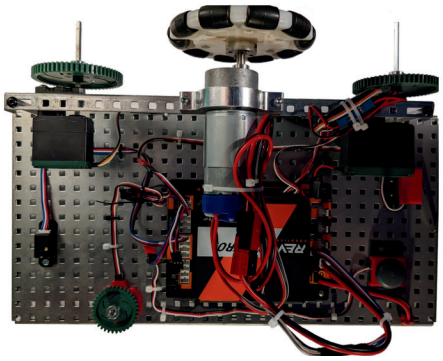
CONTROL SYSTEMS

ASSIGNMENT #3

An introduction to the Rev Control Hub, the Driver Station App, and the testBoard

REQUIRED ELEMENTS



TESTBOARD

ANDROID PHONE

BATTERY

COMPUTER

GAMEPAD

USB ADAPTER CABLE

ASSIGNMENT #3

- The objective of this assignment is to learn how to use an attached gamepad to control various outputs, similar to how one might control an actual robot.

WHAT TODO

Open the file 'Assignment3.java' in Android Studio.

TODO 1. Enable the Program

- The first change you are going to make to your program is to **comment out** a section of code near the top of the program.
- Each program can be enabled/disabled with a simple command. This will hide/show the program on the phone's lists as desired, allowing you to only see the programs you want to run at any given time.
- Remember, that to **comment out** something, add **//** to the beginning of the line.
- Your program should look like the following:

```
//////////////////////////////////////////////////
@TeleOp(name="Assignment 3", group="actuators")
// TODO #1: Enable the program
//@Disabled
//////////////////////////////////////////////////
```

USING A CONTROLLER

Know your controller type

The code you write in the following sections will be dependent on the type of controller you have.

triangle

circle

cross

square

The most notable difference in compatible controller types will be the right thumb buttons. Both ABXY and PlayStation style setups work and can be referenced directly in code, no conversion needed.

Refer to the left for PS4 button names.


```
// Code Segment

// PS4 Style
gamepad1.cross;

// ABXY Style
gamepad1.b;
```

This assignment will utilize the PlayStation layout

WHAT TODO

TODO 2. Button control

1. All of the buttons on the controller are capable of sending signals to the Driver Station App, allowing it to control the various outputs of the Control Hub. Every action must be manually defined by the programmer.
2. The right-thumb buttons (triangle, circle, cross, square) are some of the many digital inputs available on the controller.
    a. Each digital input returns a value of true or false when called
    b. The first part of the call is 'gamepad1'
        i. This is where the input comes from
        ii. Up to 2 gamepads can be used
    c. The second half is the button you want to read
3. This call will return a boolean value (true or false) when invoked
    a. This makes it convenient and easy to use in conditional statements!
4. Add the following code at TODO 2 in your code to send telemetry to the Drivers Station when the triangle button is pressed

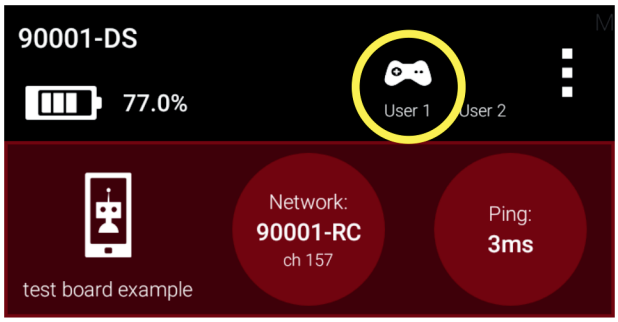
```
// Sample gamepad call
gamepad1.triangle;
```

```
// Code Segment
if (gamepad1.triangle) {
    telemetry.addLine("The triangle has been pushed!");
} else {
    telemetry.addLine("No triangle.");
}
```

Deploy your program

CONNECTING THE GAMEPAD

1. Plug in the gamepad to the phone using the USB adapter cable
2. Once they are connected, press the Options and Cross buttons simultaneously to enable it in the Drivers Station App
3. A symbol of a gamepad will appear once the gamepad is successfully connected.



Run your program

Note: **Assignment 3 is a TeleOp program**. It will be found under the righthand arrow.

TODO 3. Button Types

The gamepad has multiple different ways of communicating an operator's desire to control an output. There are both digital and analog inputs available on the gamepad. This section will look at a comparison between how those two work.

1. Most of the buttons on the gamepad are digital (remember: true or false). The next button we are going to use is part of the dpad.
2. Add the following code to your program.

```
// Code Segment
if (gamepad1.dpad_up) {
    actuators.setServoPosition(0.5);
}
```

Deploy and run your program

QUESTIONS

1. What happens when you press the button?
2. What happens when you release the button and press it again?

TODO 3. Button Types, cont.

1. While the code works perfectly as written, we will get better functionality if we add in an else-statement that resets the position of the servo when the button is not pressed.
2. This will be the servo's resting position. Add in the following code to enable this functionality to your program.
3. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment
if (gamepad1.dpad_up) {
    actuators.setServoPosition(0.5);
} else {
    actuators.                    (0);
}
```

} You have already written these lines of code.

Deploy and run your program

QUESTIONS

1. What happens when you press the button?
2. What happens when you release the button and press it again?
3. Are there any other positions you can get the servo to rest in?

TODO 3. Button Types, cont.

1. To add more stopping positions for the servo, we'll need to add in more code programming in those positions.
2. We'll use a different button this time, to demonstrate other digital inputs on the controller
3. To do this, we'll add in an else-if statement. Remember that the last section of an if statement must be the else statement. (See example below)
4. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment
if (gamepad1.dpad_up) {
    actuators.setServoPosition(0.5);
} else if ([redacted].left_bumper) {
    [redacted](1);
} else {
    actuators.[redacted](0);
}
```

You have already written these lines of code.

You have already written these lines of code. *However*, they will need to be moved to the right location.

Deploy and run your program

QUESTIONS

1. What happens when you press the button?
2. What happens when you release the button and press it again?
3. Are there any other positions you can get the servo to rest in?

TODO 3. Button Types, cont.

1. In order to add all possible stopping values for the servo, we'll have to do something more creative.
   a. Remember that some of the buttons on the gamepad behave as analog sensors.
   b. Analog sensors give back a range of values (in this case 0-1), instead of only true or false
   c. Because these values are not true/false, we cannot directly put them in an if statement. Instead, we must create a boolean expression.
   d. These expressions, when evaluated return a response of true/false
      i. For example: Is the value greater than 0? Is the value equal to 0.5?
2. We'll add in an additional else-if statement.
   a. This else-if will come after the one you have already written
   b. Remember that the last section of an if statement must be the else statement. (See example below)
3. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment

if (gamepad1.dpad_up) {
    actuators.setServoPosition(0.5);
} else if (          .left_bumper) {
                                   (1);
}          (gamepad1.left_trigger > 0) {
    actuators.                    (gamepad1.left_trigger);
} else {
    actuators.                    (0);
}
```

You have already written these lines of code.

You have already written these lines of code. *However*, they will need to be moved to the right location.

Deploy and run your program

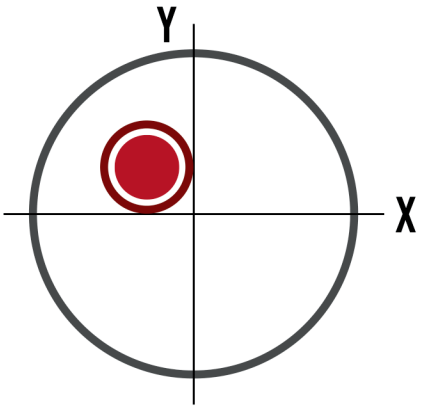
QUESTIONS

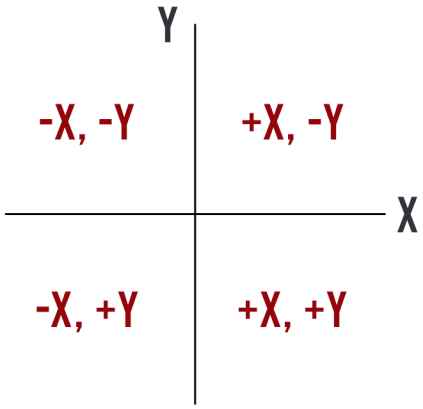
1. What happens when you press the back trigger?
2. Can you stop the position of the servo in any location you want?
3. How can you hold it in one spot?

TODO 4. Joysticks

1. The last major component of a controller you might wish to use are the two thumbsticks (joysticks)
2. Each joystick is capable of returning a precise location, based on its x and y position
    a. The coordinate plane below illustrates the joystick's potential movement. The larger outer circle encompasses the outer limits for the joystick

        When reported back to the Driver Station, X and Y coordinates are assigned positive and negative values as shown below.





3. We would like to use the values given back from the joystick to directly control the output speed of the motor. Since the joysticks' default position is (0,0), we do not need an if statement, and can directly send the value to the motor.
4. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment

          .setMotorPower(gamepad1.left_stick_y);
```

Deploy and run your program

QUESTIONS

1. Carefully move the joystick to the left and right without going up and down. What happens? Why?
2. Carefully move the joystick to up and down without going left and right. What happens? Why?
3. What occurs when you move the joystick diagonally?