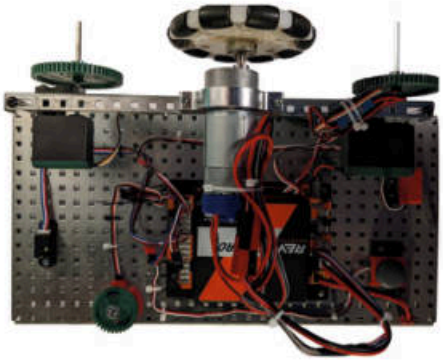
CONTROL SYSTEMS

ASSIGNMENT #2

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

REQUIRED ELEMENTS



TESTBOARD

ANDROID PHONE

BATTERY

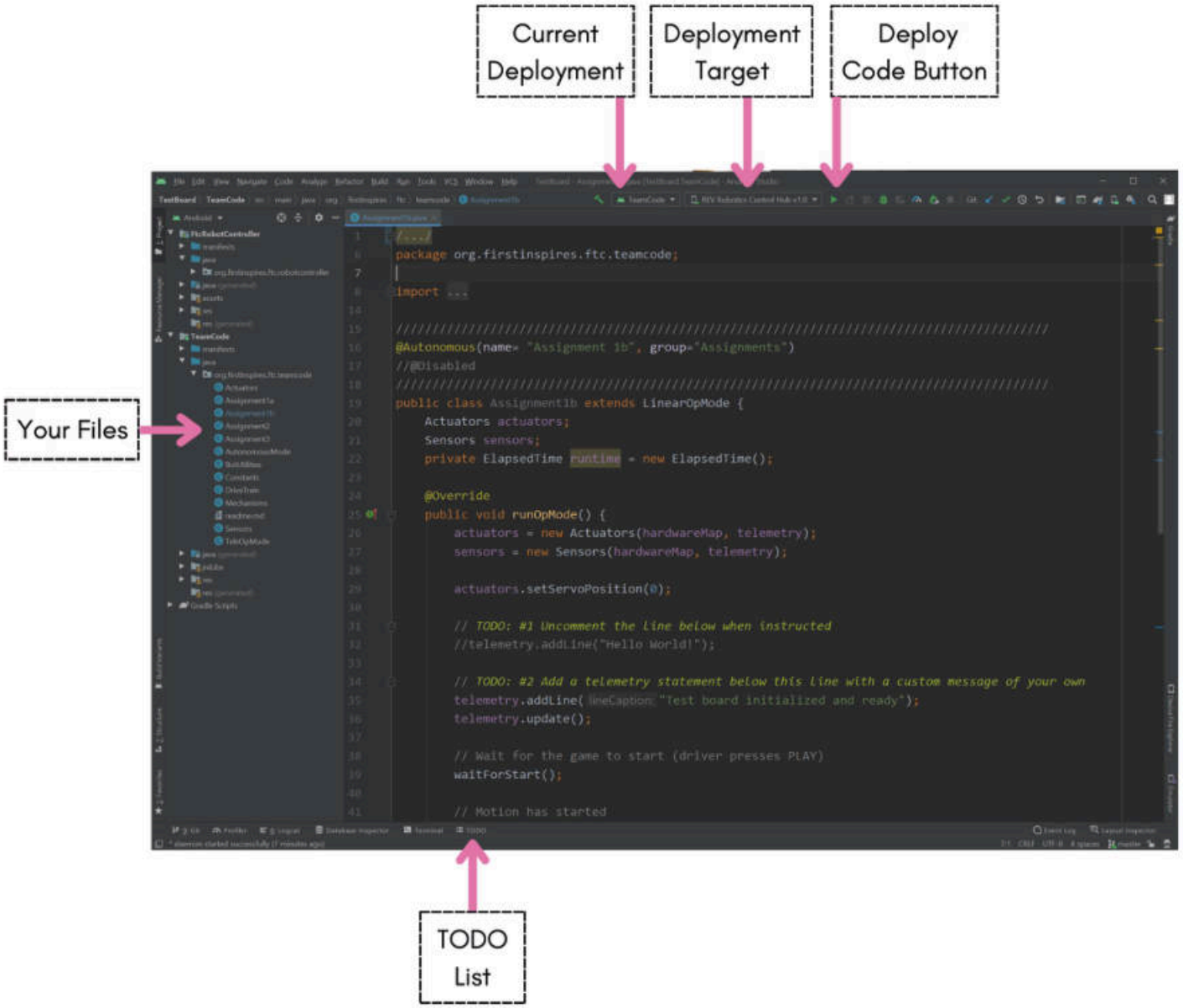
COMPUTER

ANDROID STUDIO

The code that is running on the Control Hub was written in a program called Android Studio, using the Java programming language. In this assignment, you will be making small modifications to this code and beginning to write your own programs to run on the testBoard.

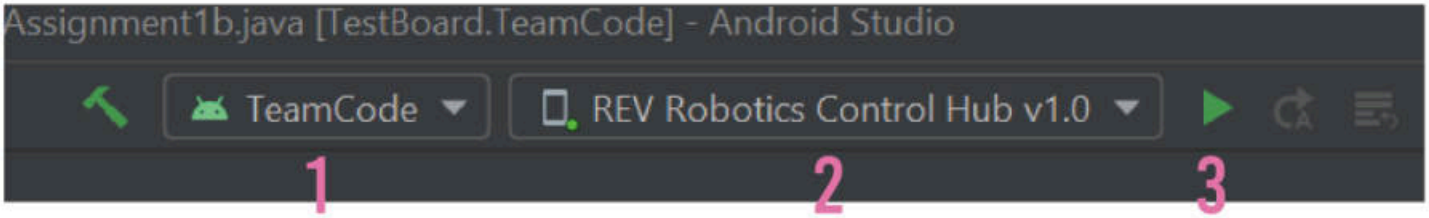
THE INTERFACE

Android Studio is an Integrated Development Environment (IDE) that we use for writing code that can be deployed to a Control Hub. The IDE has many useful features. The screen below shows what you should see on your computers when you are first starting out.



THE INTERFACE

In order to deploy your code, we need to verify some of the settings in your interface. At the top of the window, you should see something that looks very similar to what is shown below.



1. Current Deployment

This dropdown represents what specific package of code will be sent to the Control Hub. For these assignments, this should always read 'TeamCode'

2. Deployment Target

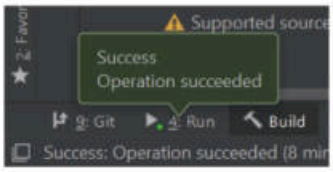
This dropdown represents where this specific package of code will be sent. For these assignments, this should always read 'REV Robotics Control Hub'.

- Occasionally, typically after disconnecting/reconnecting with the computer, this setting will be lost. Hovering over the dropdown while the device is physically connected to the computer will get it to repopulate the field.

3. Run Code Button

This is the button you will utilize to send your code to the Control Hub.

THE PROCESS



The first time you deploy your code after turning on the Control Hub, the process may take upwards of two minutes. All subsequent deployments should only take a handful of seconds. However, this does not mean your program is ready to run yet! The program must then be installed and launched on the Control Hub. You will know when this is completed by looking for the success message (see above), the light on the Control Hub returns to solid green, and the phone display has reset back to normal.

ASSIGNMENT #2

- The objective of this assignment is to learn how to utilize sensor input and conditional statements in order to control outputs.

Conditional statements are used in computer science to make decisions.

These decisions are made based on a boolean condition, true or false.

WHAT TODO

Open the file 'Assignment2.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:

```
////////////////////////////////////////////////////////////
@Autonomous(name= "Assignment 2a", group="Assignments")
// TODO #1: Enable the Program
//@Disabled
////////////////////////////////////////////////////////////
```

WHAT ARE VARIABLES?

A variable is a container for some bit of information we want to use in a program. Each variable consists of three main parts:

TYPE IDENTIFIER VALUE

```
double motorPower = 0.75;
```

DATA TYPES

A data type defines what kind of value can be stored in a variable. There are numerous data types that can be used. For our applications, we can largely limit ourselves to three main data types:

DOUBLE	BOOLEAN	INT
decimal values	*true* or *false*	integer values

IDENTIFIERS

An identifier is a **name** given to a variable. We use this name to assign or access the value of the variable within the program.

VALUE

A value is what is represented by a variable. A value must always match the data type.

EXAMPLE VARIABLES

```
double servoPosition = 0.5;
boolean defaultState = true;
int encoderTarget = 12500;
```

WHAT TODO

TODO 2. Using Variables 1

The first variable you are going to create will store the value read of the potentiometer. The, you are going to use this variable for a variety of tasks.

This section of code will use the potentiometer value to control the position of the standard servo.

The value coming off the potentiometer ranges from 0-1 (decimal values), meaning it is a double.

Find TODO #2 in your program and add the following code segment.

```
// Code Segment
double potValue = sensors.getPotReading();
actuators.setServoPosition(potValue);
```

Deploy and run your program

QUESTIONS

1. How does the servo correspond to the potentiometer position?
2. Does the servo move without potentiometer motion?

WHAT TODO

TODO 3. Using Variables 2

1. You are going to use the same variable used in the last TODO for this section of code.
    a. **Note**: Variables, once created, do **not** need to be created again. They will exist forever is the section of code you created them in.
2. Double values are used to set the power of the continuous rotation servo. We are going to use the value from the potentiometer to control its speed.
3. Add the following code where indicated for TODO #3:

```
// Code Segment
actuators.setCRServoPower(potValue);
```

Deploy and run your program

QUESTIONS

1. How does the behavior of the CR servo correspond to the potentiometer position?
2. Does the CR Servo move when the potentiometer doesn't?
3. Describe the difference in how the two types of servos work, even when given the same value.

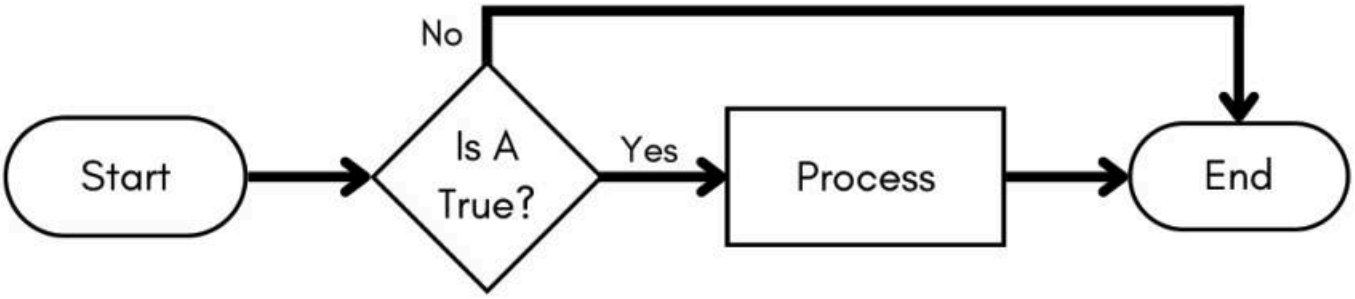
THE IF STATEMENT

Most code runs in a linear manner, top-bottom through sections of code. This is referred to as program flow. There are several types of code that can change this flow, allowing the program to repeat or skip over sections of code.

One type of statement that can change program flow is an if-statement. If-statements perform a certain action if (and only if) a certain condition is met.

```
// Code Example

if (condition A) {
   // Condition is true
   Process;
}
```

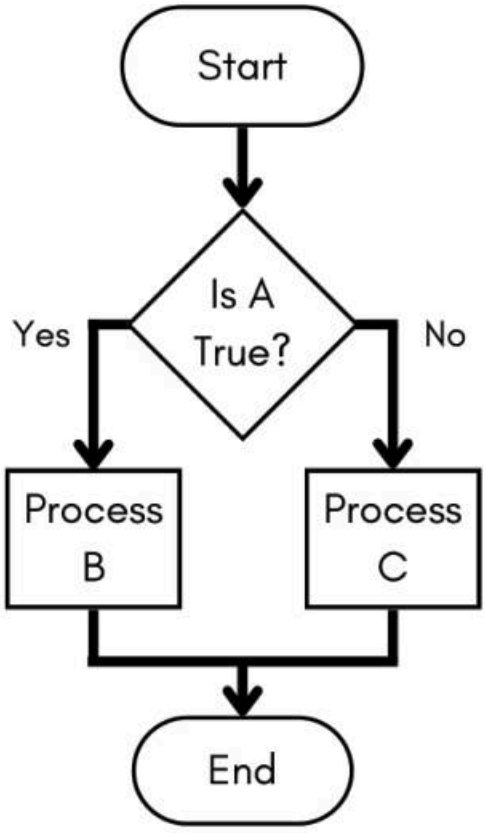


Sometimes, we want a program to do a specific action is something is true and something else if it is false.

The if-else statement is a branching statement that chooses between two possible actions.

```
// Code Example

if (condition A) {
   // Condition is true
   action B;
} else {
   // Condition is false
   action C;
}
```



WHAT TODO

TODO 4. Button Telemetry

1. The next change you are going to make to your program is to write a section of code that will display telemetry statements to the phone based on the status of the limit switch.
2. Remember, sensors such as the limit switch return a value that is either true or false.
    a. True - the lever is pressed
    b. False - the lever is not pressed
    c. This is a boolean value
3. Type in the following code statements into your programs at the location of TODO #4.
    a. **Some parts of this code have been obscured.** You will have to use the programming knowledge you have obtained to write the missing pieces.

```
// Code Segment

if (sensors.getLimitSwitchState()) {
    telemetry.addLine("You pushed the button!");
} else {
    [redacted]("Hello darkness my old friend.");
}
```

Deploy and run your program

The top message should appear when you press the limit switch. If that does not work, check your code and try again.

QUESTIONS

1. Do you think you are limited to the number of actions that can be performed inside an if-statement?

WHAT TODO

TODO 5. Variables and If-else

1. This section of your code will be written in several steps, adding on more features as you go.
    a. This is a good programming practice. Adding things in small steps allows you to verify individual elements are working the way they are intended and can make finding errors easier.
2. Write and if-statement to control the motor based on the status of the pushButton. The speed of the motor's rotation should come from the value read off of the potentiometer.
    a. **The code below has sections obscured.** Use your programming knowledge to fill in the blanks.

```
// Code Segment

if (          .                          ) {
    actuators.setMotorPower(          );
}
```

Deploy and run your program

QUESTIONS

1. What happens when the potentiometer is turned to a value above zero and you press the pushButton?
2. What happens when you release the pushButton?
3. Can the motor be stopped without turning the potentiometer all the way down?

WHAT TODO

TODO 5. (continued)

1. With the basic if-statement, you used the pushButton to start the motor spinning. When you release the pushButton, the motor keeps on spinning! This is because there is nowhere in the program where you tell the motor to turn ***off***.
2. We are going to use and if-else-statement to program the motor not to spin unless the pushButton is pressed.
   a. Remember the purpose of an 'else' is to give a behavior to do whenever the specified condition is not met.
3. Using the code segment below, add an else-statement onto the end of your if-statement to stop the motor from running when the pushButton isn't pressed.
   a. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment

if (          .                          ) {
    actuators.setMotorPower(          );
} else {
    
}
```

You have already written these lines of code.

Deploy and run your program

QUESTIONS

1. What happens when the potentiometer is turned to a value above zero and you press the pushButton?
2. What happens when you release the pushButton?
3. Can the motor be stopped without turning the potentiometer all the way down?

THE ELSE-IF STATEMENT

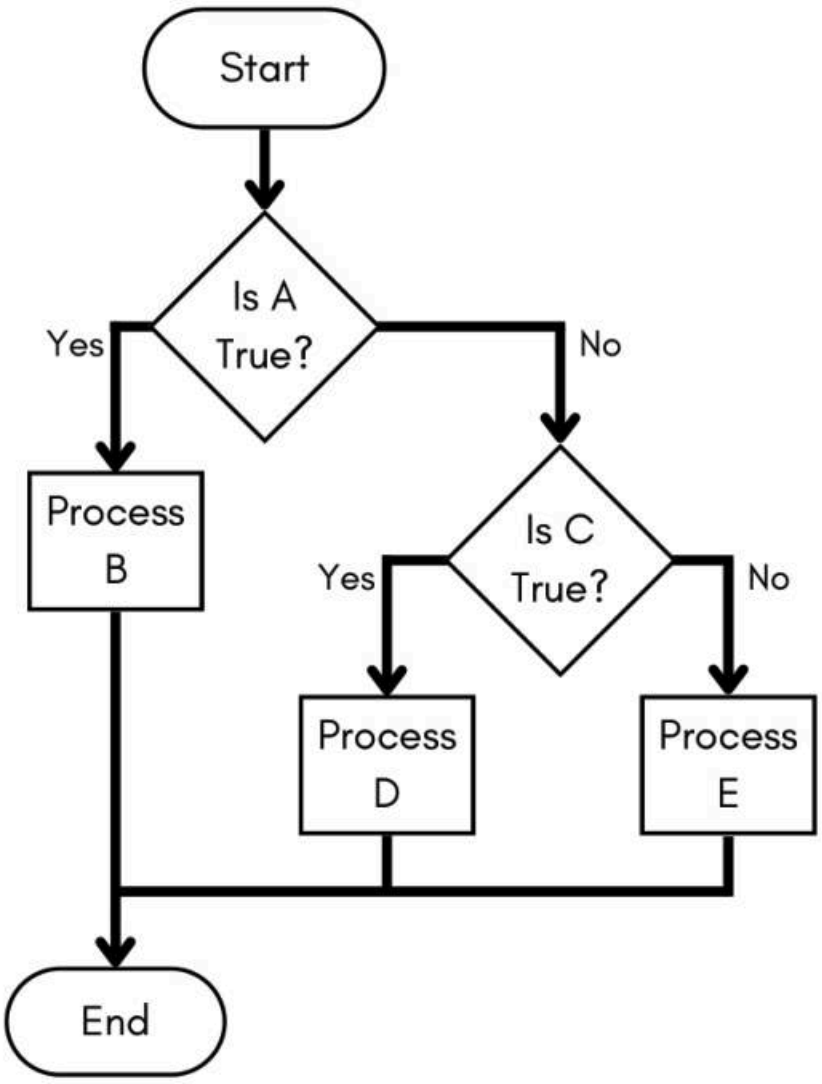
For complex programs, oftentimes, the if-else statement is not enough. We might want a program to exhibit a specific behavior is sometimes occurs, or is something else occurs to do something different, and if neither of those things happen, a default behavior.

To accomplish this, we need to utilize an else-if statement.

When a program sees an if statement with branching else-ifs, it will...

- The program will first check the if-statement
- If it is false, the program will move on to check the else-if statement
- If that is false, the program will default to the else statement

There are no limits to the number of else-ifs that can be added to an if statement.




```
// Code Example

if (condition A) {
   // Condition is true
   action B;
} else if (condition C) {
   // Condition A is false
   action D;
} else {
   // Condition C is false
   action E;
}
```

WHAT TODO

TODO 5. (continued)

1. For the last section of your program, we want to add code that allows us better control over the motor.
2. Add in an else-if statement to your code such that...
    a. When the pushButton is pressed, the motor runs forwards
    b. When the limit switch is pressed, the motor runs backwards
    c. When neither are pressed, the motor does not spin
3. The code segment below is partially obscured. Use your programming knowledge to fill in the missing components.

```
// Code Segment

if ([redacted]) {
    actuators.setMotorPower([redacted]);
} else if ([redacted].getLimitSwitchState()) {
    [redacted].[redacted](-potValue);
} else {
    [redacted]
}
```

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. Press and hold the pushButton. While holding it, press the limit switch. What happened?
2. Press and hold the limit switch. While holding it, press the pushButton. What happened?
3. Why do you think these behaviors did/didn't occur?