Scratch

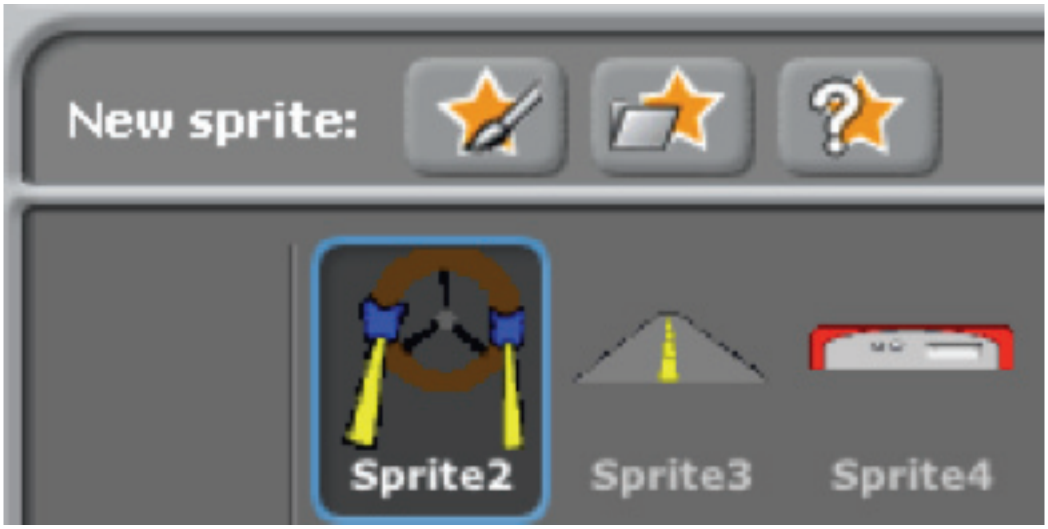
// Overview:

Scratch is a wonderful tool for teaching basic computer programming concepts. With it's drag and drop interface it is intuitive and unintimidating for young children. Now with the Pico Board students can learn how to use sensor input from hardware to control games and animations that they have created in Scratch. This tutorial is designed to get a student started creating a driving simulator that uses the slider and light sensor on the Pico Board to control the game. Once students understand the concept of how to use these sensors they can easily build their own steering wheel using simple electrical hardware as outlined at the end of this tutorial. This tutorial is intended for students who already have at least a little experience programming in Scratch, but have never used hardware with Scratch.

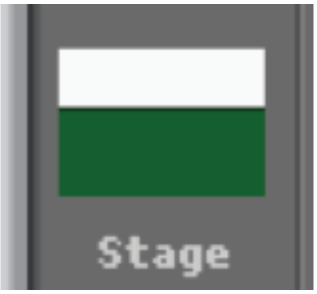
Name:
Date:

// Setup:

1. Make three Sprites, a car, a steering wheel and a road.



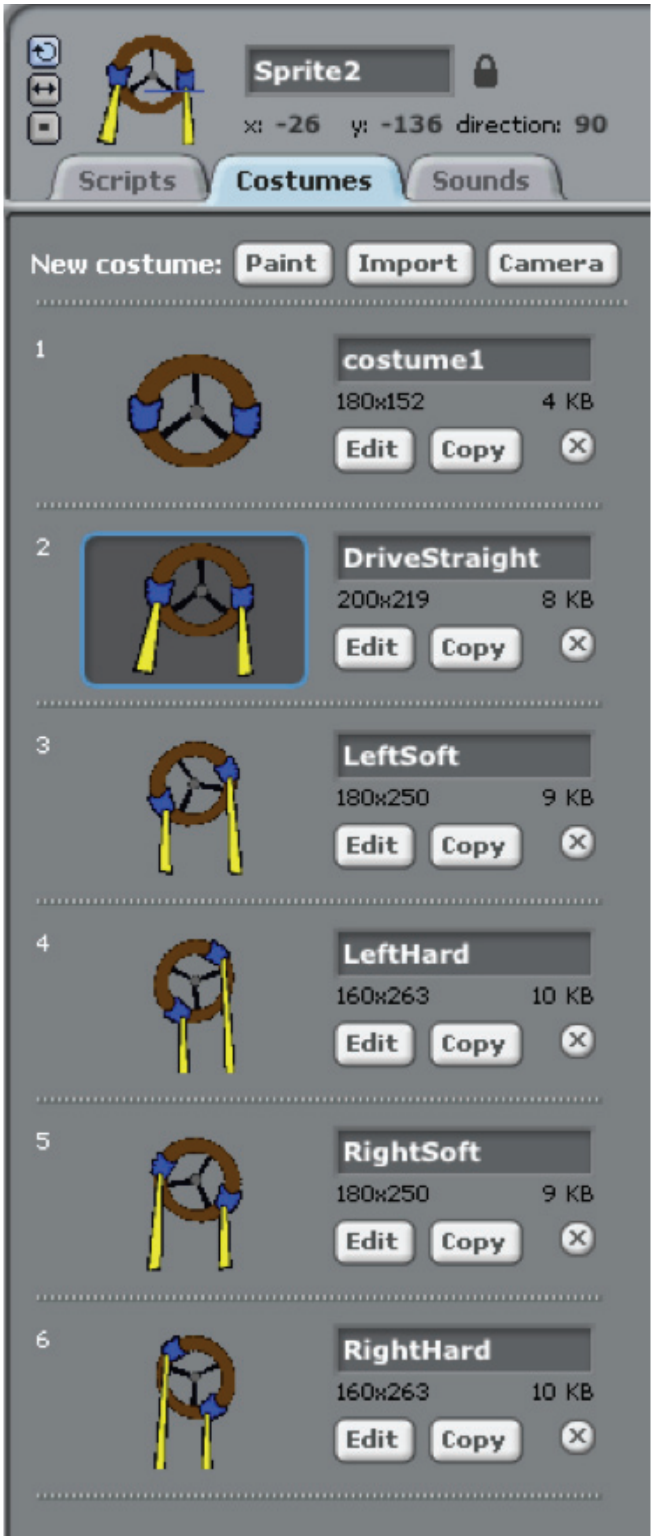
2. Make your stage half sky and half a solid green color.



Name:
Date:

// Setup:

3. Make five different costumes for the steering wheel Sprite: One turning left hard, one turning left somewhere in between the first costume and the straight ahead costume, one as the drive straight ahead costume (you already did this in the last step), one turning right hard, and the last costume turning right somewhere in between the turning right hard and the driving straight costume. Hint: Once you have created the first two turn right costumes you can just flip the costume to make your turn left costumes.



Sprite2

x: -26 y: -136 direction: 90

1 costume1 180x152 4 KB

2 DriveStraight 200x219 8 KB

3 LeftSoft 180x250 9 KB

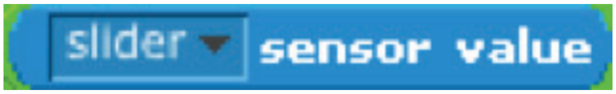
4 LeftHard 160x263 10 KB

5 RightSoft 180x250 9 KB

6 RightHard 160x263 10 KB

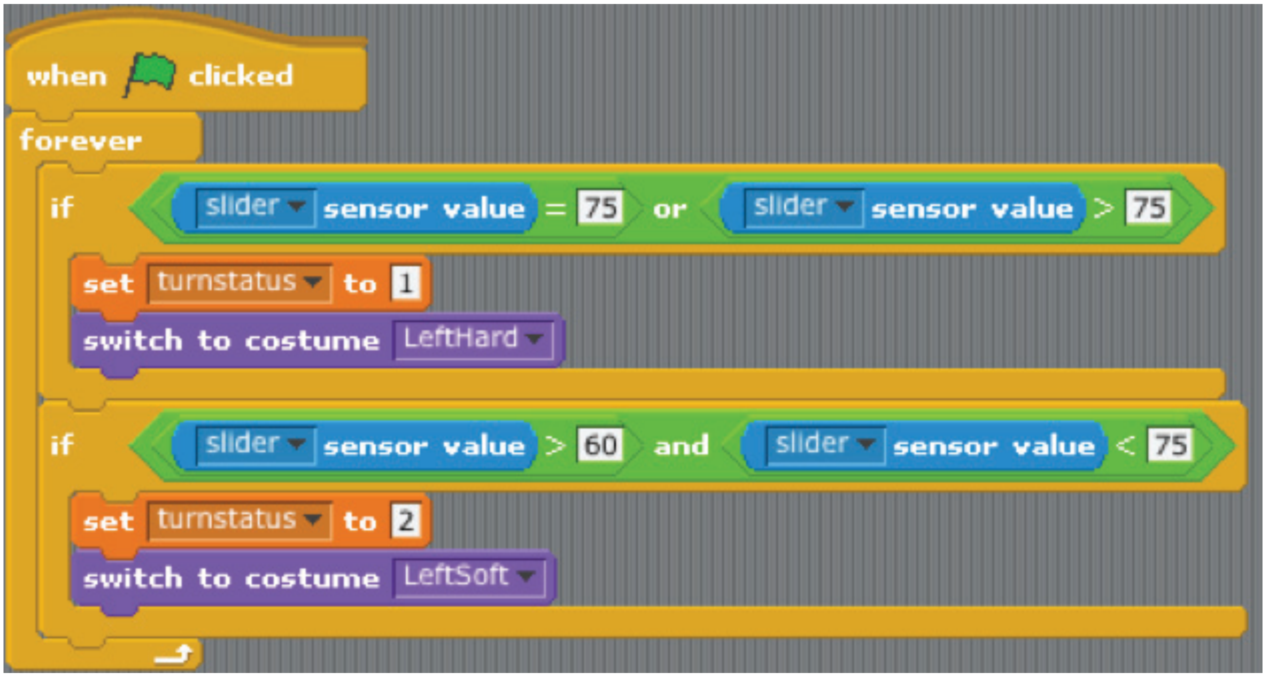
// Setup:

4. Make a variable called “turnStatus” or something that makes sense to you. You will use this variable to control which way and how hard your car is turning. Plug in your Pico Board. Before we get started let’s talk about how the slider on your Pico Board works with Scatch. When the slider is all the way down (closer to the ear symbol) the slider sensor variable will be equal to zero. When the slider is all the way up (closer to the eye symbol) the slider sensor variable will be equal to one hundred. When you slide the slider between these two positions it will be somewhere in between zero and one hundred. Use this block of code to get the slider sensor variable:



```
(slider ▾ sensor value)
```

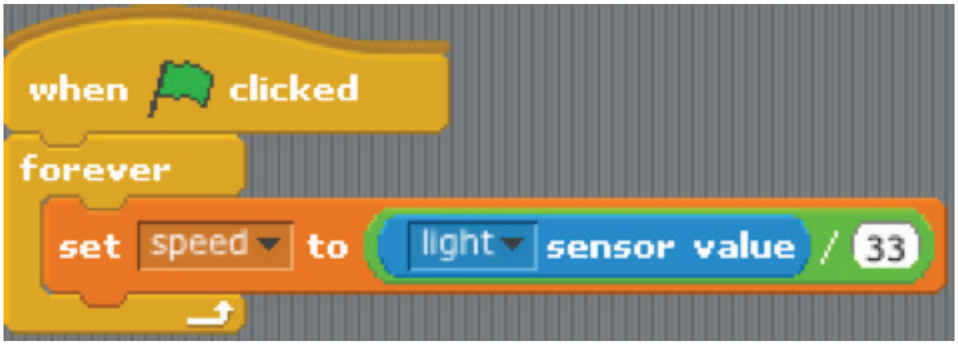
5. Make the steering wheel costume change depending on the position of the slider on the Pico Board. To do this you will create a bunch of if statements that check the position of the slider. The if statement should change the steering wheel costume to match the position of the slider. The if statement should also change the “turnStatus” variable that you will use to control the position of the Road Sprite in the next step. Below is an example of the code I am using to turn left, this code is inside the steering wheel sprite. In order to make your car turn right you will have to add your own if statements.



```
when ⚑ clicked
forever
    if <<(slider ▾ sensor value) = 75> or <(slider ▾ sensor value) > 75>>
        set turnstatus ▾ to 1
        switch to costume LeftHard ▾
    if <<(slider ▾ sensor value) > 60> and <(slider ▾ sensor value) < 75>>
        set turnstatus ▾ to 2
        switch to costume LeftSoft ▾
```

// Setup:

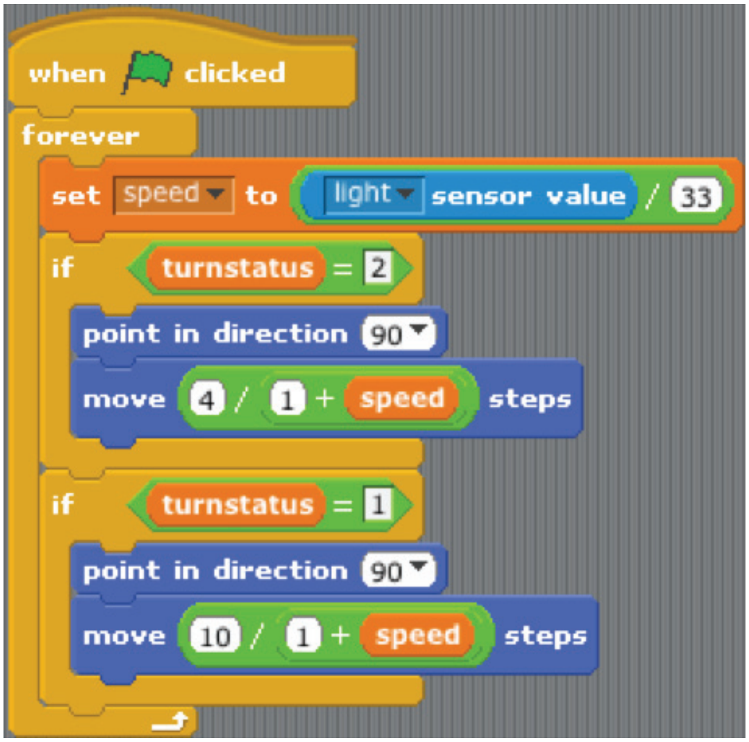
6. Now let’s go ahead and use the light sensor to get a “speed” variable. To do this you will make a variable named “speed” and use the light sensor value block of code to give the “speed” variable a value. In my case I don’t want the “speed” variable to go above three. Because I know that the sensor value I get from the light sensor will go up to 100 I divided it by 33. This is because 33 goes into 100 about 3 times. In my code the higher the variable speed is the slower the car will travel.



```
when ⚑ clicked
forever
    set speed ▾ to (light ▾ sensor value / 33)
```

7. By now your steering wheel Sprite turns, and the light sensor controls the speed variable, but the car doesn’t move around on the road. There are a couple different ways you can do that. One of the easier ways is to move the road while keeping the car and steering wheel in the same position. In order to do this you will need to use the “turnStatus” variable to set the heading of the road and the speed at which the road moves side to side. The block of code below is inside my road sprite and there are two different things going on.

The upper if statement code runs if the turnStatus variable is equal to 2 which means the car is making a slight left turn. If the car is making a slight left turn the code sets the direction of the road to the right so that when the road moves in that direction the car appears to turn left. After that the road Sprite moves a certain number of steps depending on the “speed” variable. The other if statement section of code just below that does the same thing, except that because it is a harder left turn the road moves even more.

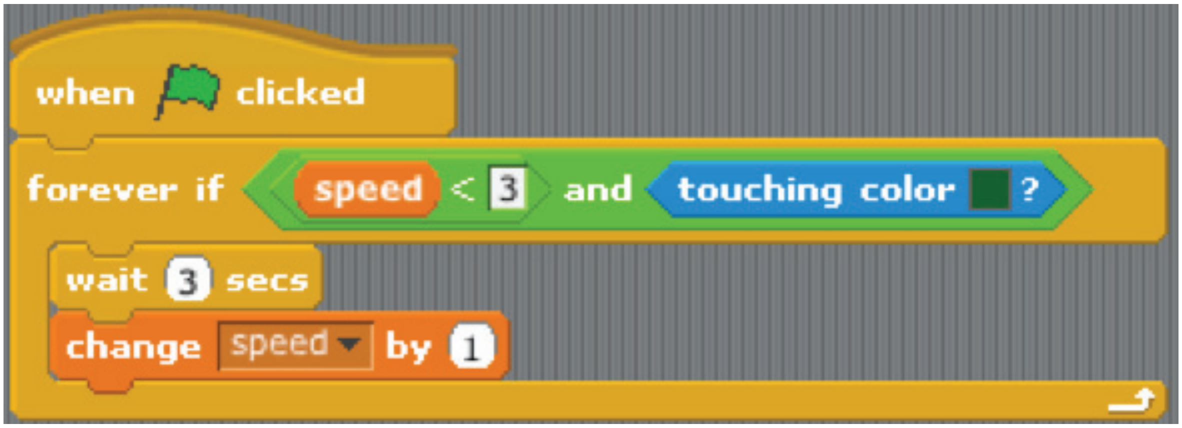


```
when ⚑ clicked
forever
    set speed ▾ to (light ▾ sensor value / 33)
    if turnstatus = 2
        point in direction 90 ▾
        move (4 / (1 + speed)) steps
    if turnstatus = 1
        point in direction 90 ▾
        move (10 / (1 + speed)) steps
```

Name:
Date:

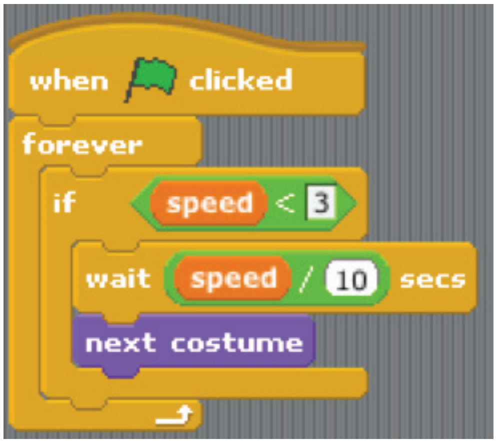
// Setup:

8. Now you have to make it so the car slows down when it touches the grass on the side of the road. To do this put the following code anywhere in one of your sprites.

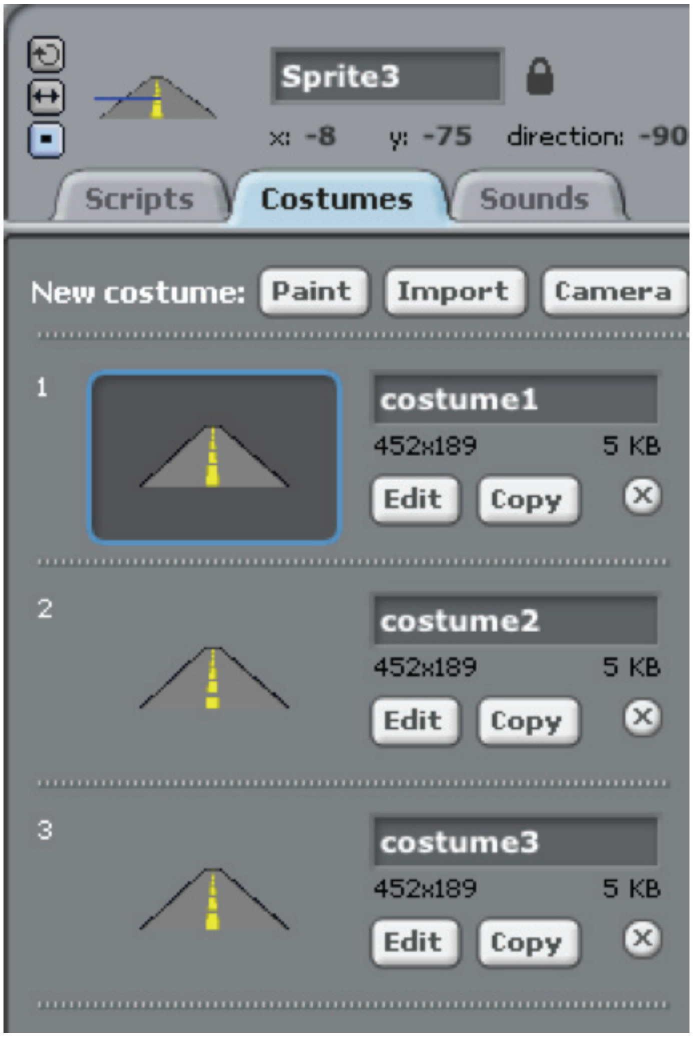


```
when 🟩 clicked
forever if < (speed) < 3 > and < touching color 🟩 ? >
    wait 3 secs
    change speed by 1
```

9. Lastly you need to make the road under the car switch between at least three different images so that it looks like the road is moving. To do this create a second and third costume in your road sprite with the yellow line in the road moved down a little and put the code below in your road sprite.



```
when 🟩 clicked
forever
    if < (speed) < 3 >
        wait (speed) / 10 secs
        next costume
```



Sprite3

x: -8 y: -75 direction: -90

1 costume1 452x189 5 KB

2 costume2 452x189 5 KB

3 costume3 452x189 5 KB

Name:
Date:

// Setup:

By now you should have a project that allows you to drive your car on a straight road using your Pico Board. Want to make it even better? Try thesse challenges…

10. Change the light sensor input for your speed variable to the microphone so that the louder you yell, the faster you go!

11. Add trees or bushes or houses or all three that get bigger and move down the screen on the side of the road so it looks like you are passing them. (you will have to make them move towards the edge of the screen too.)

12. Make them appear in random places (but not on the road, and always start really small on the horizon)

13. Make it so if you hit one of the trees or bushes you lose the game.

14. Make a “Gas” variable that starts out full but decreases depending on your speed and even more when you accelerate.

15. Make gas bonuses that you can collect which refill your gas tank/variable.

16. (Once you have steps # 7 #8 and #8. #10 should be pretty easy, as would adding many other things to the game such as other cars, billboards, etc.)

17. Now for the really tough part... can you make the road turn?

Name:
Date:

// Setup:

Make your own steering wheel:

One of the great things about the PicoBoard is that you can attach your own sensors to the A, B, C, and D ports. Now that you have a Scratch project that simulates driving how about putting together your own steering wheel?

Materials:

A large potentiometer (fancy word for dial), a battery case (with switch on case) with batteries, a cardboard tube, a sheet of thick cardboard (or something to create a steering wheel out of), wire, solder, glue and a PicoBoard.

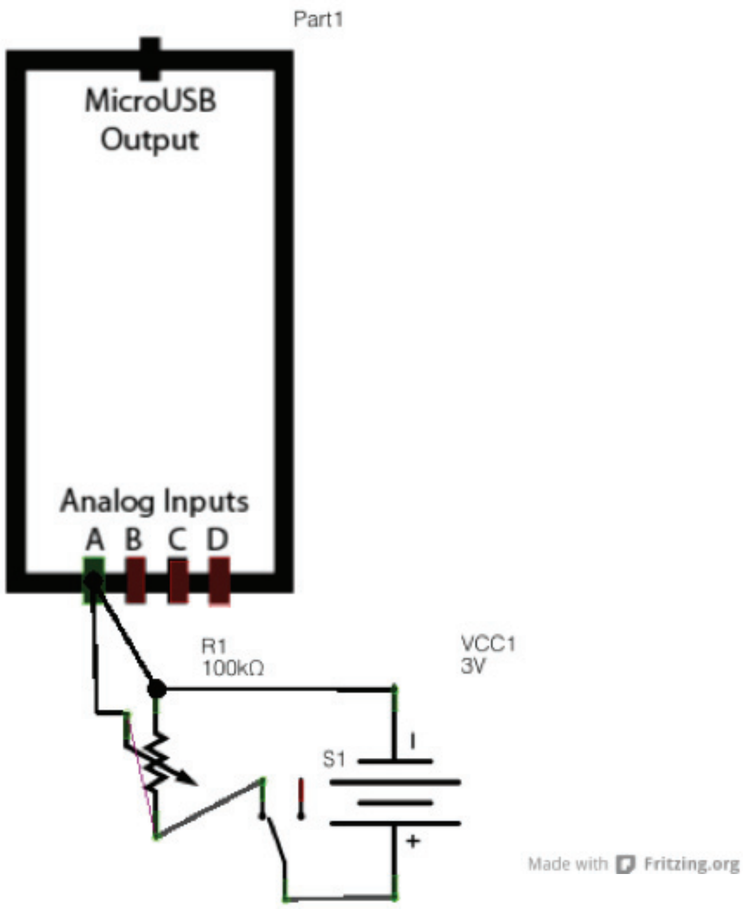
Tools:

Scissors and a soldering iron. (if you want to draw on your steering wheel you will need markers.)
To make things easier you may also want a breadboard.

Standards:

To be filled out

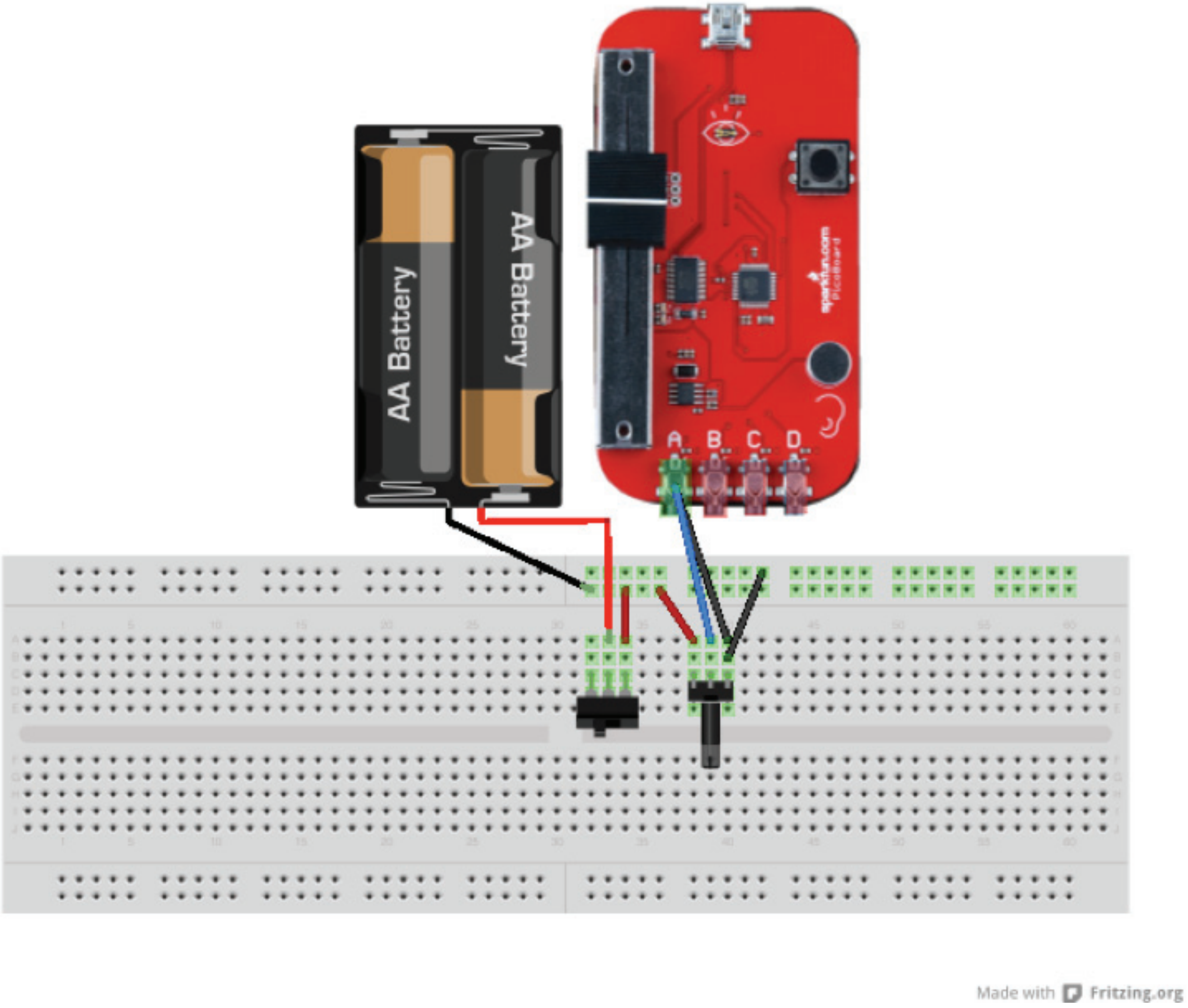
Schematic:



Name:
Date:

// Setup:

This project is fairly simple so I hooked everything up all at once on the breadboard. (This version of the breadboard shows a switch added on to the circuit, but often battery packs come with switches already on them.)

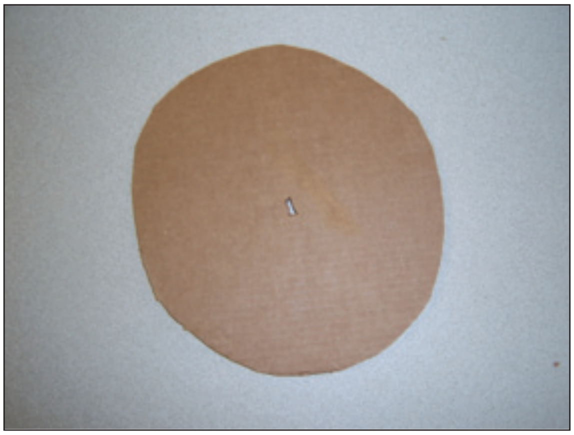
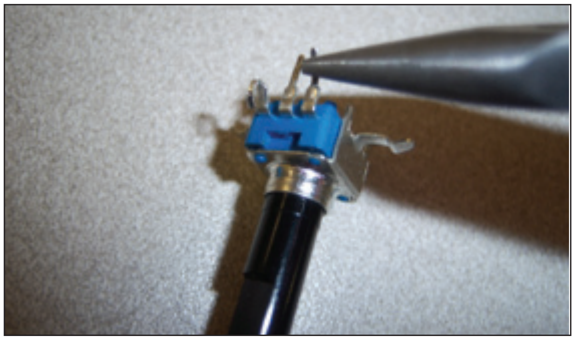
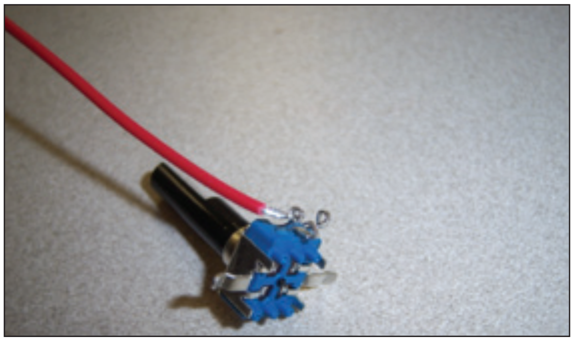


The battery has an on/off switch attached to it’s power line. The battery supplies power to the dial and the ground and communication line (blue wire) plug into the PicoBoard. The PicoBoard comes with alligator clips that attach to the communication line and the ground. The other side of those alligator clips is a single cord that plugs into any of the Analog Inputs on the PicoBoard, providing both the voltage of the dial and a ground signal.

Name:

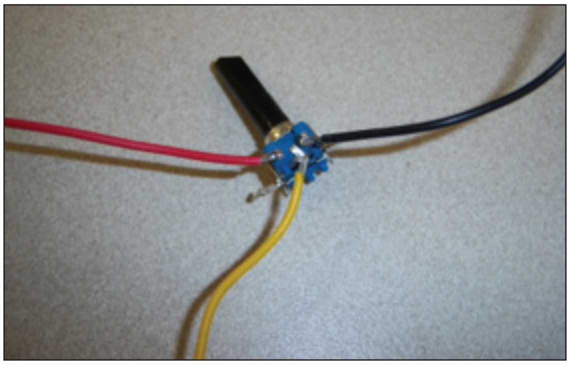
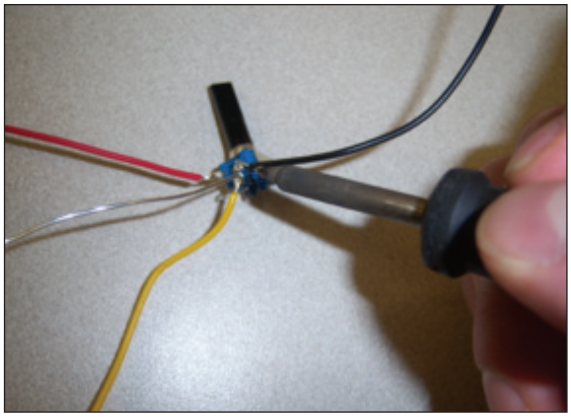
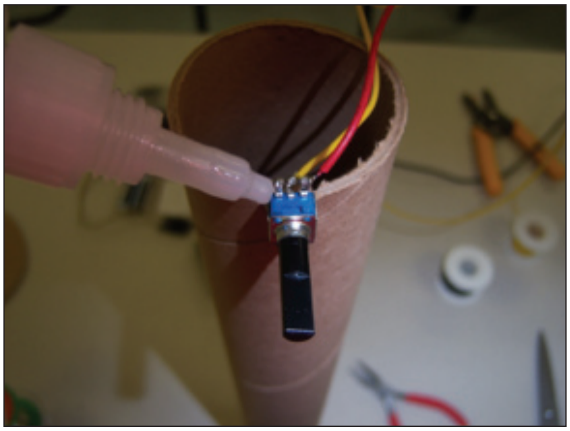
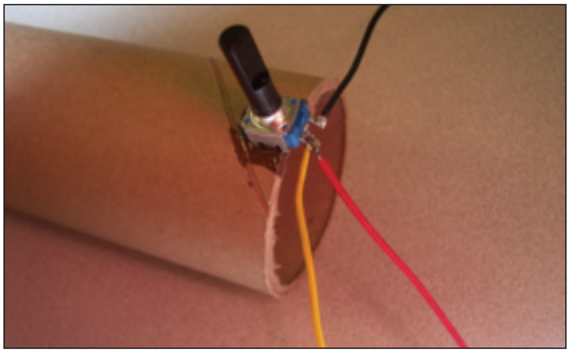
Date:

// Setup:

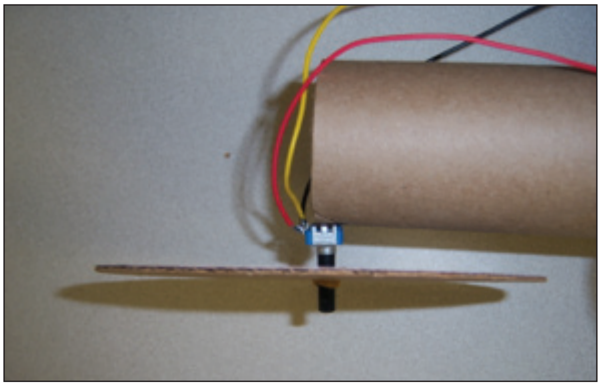
	1. First cut a steering wheel shape out of your flat piece of cardboard. In order to get stiffness that will last a while you may want to glue two or three pieces of cardboard together, or use a different material entirely. You may want to decorate your steering wheel.
	2. Cut a hole smaller than the size of the end of your dial in the middle of your steering wheel. Put your steering wheel aside now.
 	3. Solder a wire about three feet long (this size may vary depending on the size of your cardboard tube, I got away with one a half feet) to each of the three leads, or connections, on the dial. You may need to bend the leads on the dial to make this easier. To bend the leads use a pair of pliers to bend them into circles and then twist the wire around the lead.

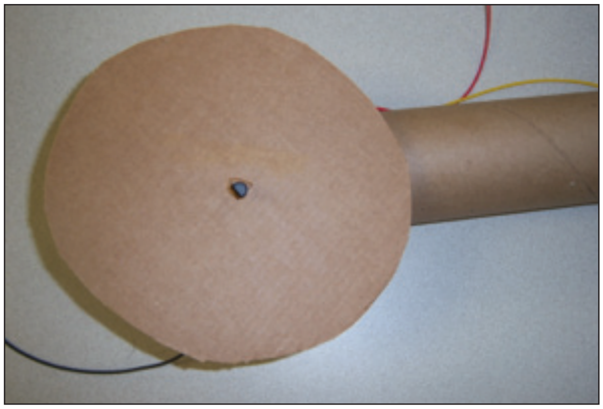
Name:
Date:

// Setup:

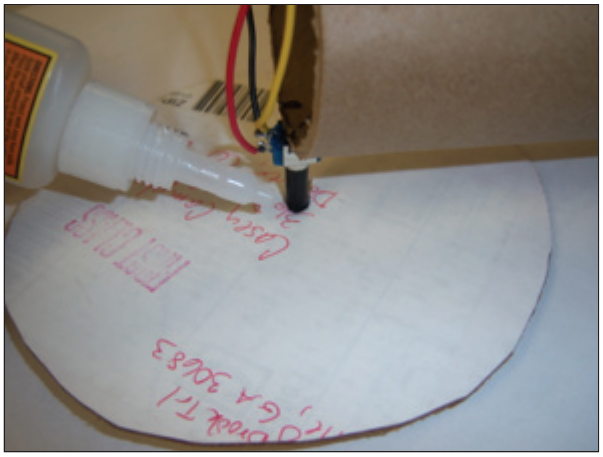
 	Step 3 continued.
 	4. Glue (or attach it in a better way) the base of the dial to the inside of the top of the cardboard tube. The top of the dial should stick up out of the tube. Let the glue dry. Make sure you don't glue your dial to itself so it doesn't glue itself to one position.

// Setup:





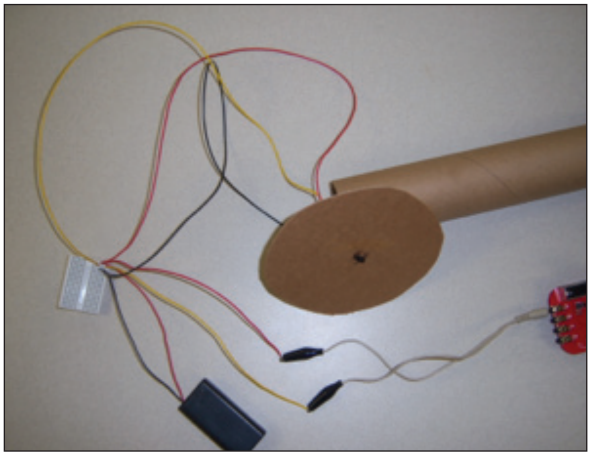
5. Slide the steering wheel onto the end of the dial that is sticking out of the cardboard tube.



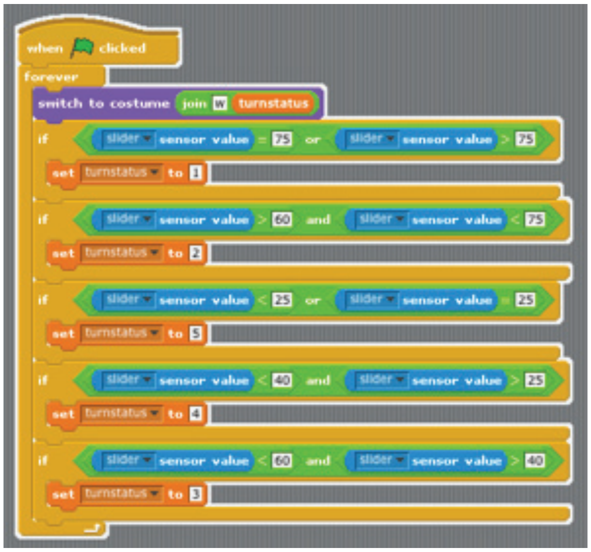
6. Here's the tricky part, glue the steering wheel onto the dial without gluing the dial so it won't turn. You will mess up the dial if you let glue drip down the dial covering where the dial attaches to the base. First apply glue to the steering wheel and dial, then make sure you're only gluing what you want to glue. To do this we suggest turning everything upside down so the steering wheel lies on the table (which has a piece of paper or something on it because you don't want to glue your steering wheel to the table) while it dries.

Name:
Date:

// Setup:

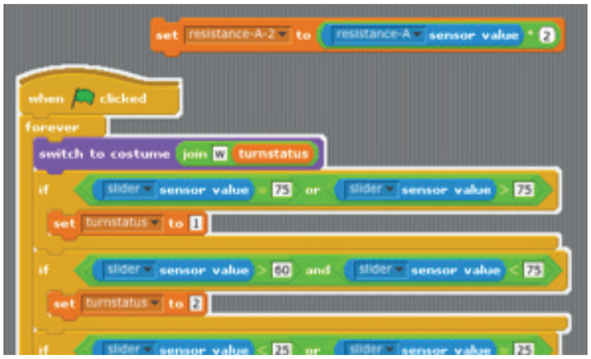


7. Once the glue is dry plug the wires in. Follow the breadboard diagram above and the pictures below to figure out how to hook up the steering wheel.



```
when ⚑ clicked
forever
  switch to costume (join W turnstatus)
  if <(slider sensor value) = 75> or <(slider sensor value) > 75>
    set turnstatus to 1
  if <(slider sensor value) > 60> and <(slider sensor value) < 75>
    set turnstatus to 2
  if <(slider sensor value) < 25> or <(slider sensor value) = 25>
    set turnstatus to 5
  if <(slider sensor value) < 40> and <(slider sensor value) > 25>
    set turnstatus to 4
  if <(slider sensor value) < 60> and <(slider sensor value) > 40>
    set turnstatus to 3
```

Change this...
to this...



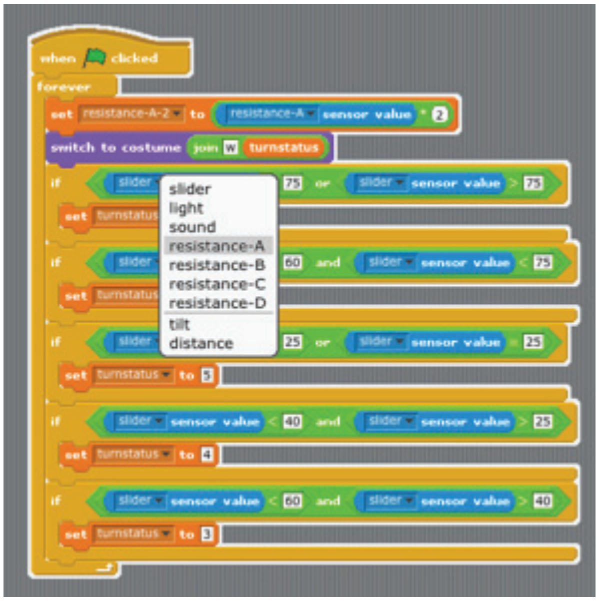
```
set resistance-A-2 to ((resistance-A sensor value) * 2)

when ⚑ clicked
forever
  switch to costume (join W turnstatus)
  if <(slider sensor value) = 75> or <(slider sensor value) > 75>
    set turnstatus to 1
  if <(slider sensor value) > 60> and <(slider sensor value) < 75>
    set turnstatus to 2
  if <(slider sensor value) < 25> or <(slider sensor value) = 25>
```

8. Change the code on your steering wheel so it uses the “resistance-A” sensor instead of the slider to control the car. To do this your will need to do more than just change the sensor your code uses from “slider” to “resistance-A”. This is because my steering wheel is hooked up to 2 AA batteries which supply 3.3V instead of the 5V the Picoboard expects.

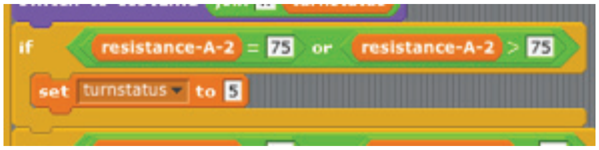
One way around this is to create a new variable (I called mind “resistance-A-2”) to store a number that is your “resistance-A” sensor multiplied by two. This way you know that your variable will get all the way up to 100 when the sensor goes up to 50. Another way around this would be to use a 5V power supply. There is a 5V power supply right on the Picoboard. Below is a picture of the connections you would need to make to replace the battery pack, if you want to make these permanent connections consider soldering some female headers onto your Picoboard.
(Just remember that you don’t need to multiply the “resistance-A” value by two if you use a 5V power source.

// Setup:

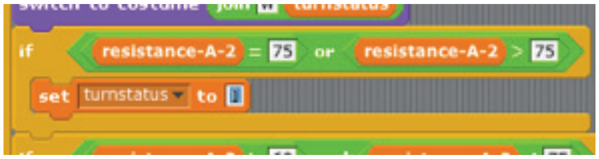


```
when clicked
forever
  set resistance-A-2 to resistance-A sensor value * 2
  switch to costume join w turnstatus
  if slider 75 or slider sensor value > 75
    set turnstatus
  if slider 60 and slider sensor value < 75
    set turnstatus
  if slider 25 or slider sensor value = 25
    set turnstatus to 5
  if slider sensor value < 40 and slider sensor value > 25
    set turnstatus to 4
  if slider sensor value < 60 and slider sensor value > 40
    set turnstatus to 3
```

9. Then you have to replace your “slider” variable with the “resistance-A-2” variable (or whatever you called your variable).



```
if resistance-A-2 = 75 or resistance-A-2 > 75
  set turnstatus to 5
```



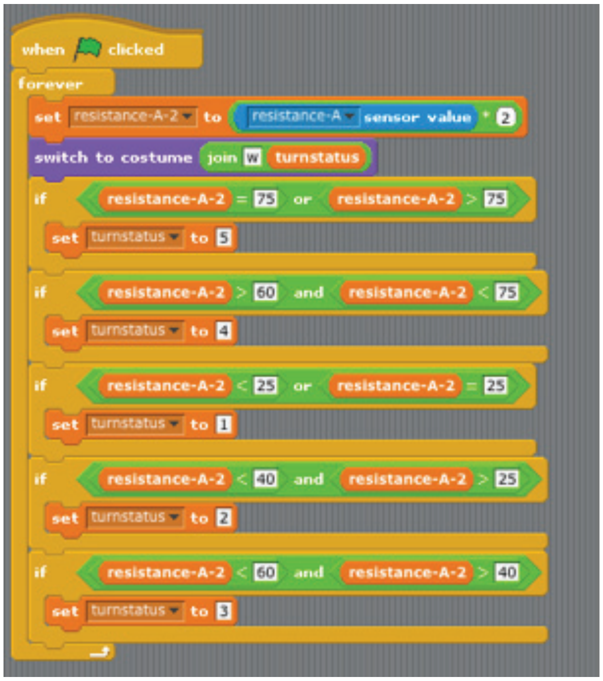
```
if resistance-A-2 = 75 or resistance-A-2 > 75
  set turnstatus to 1
```

10. Finally you may have noticed something funny about how the steering wheel works. It’s backwards! You have to change the block of code that sets the variable “turnstatus” so that instead of 5 it sets “turnstatus” to 1 and vice versa. This is because the “slider” sensor on the board is hooked up the reverse of how the dial works.

Below is an example of how my code changed. There are multiple ways you can do this, see how many other fixes you can figure out.

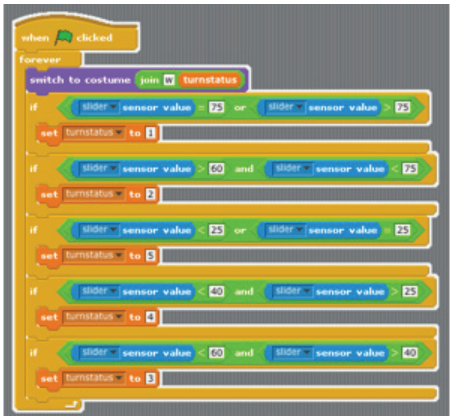
Name:
Date:

// Setup:

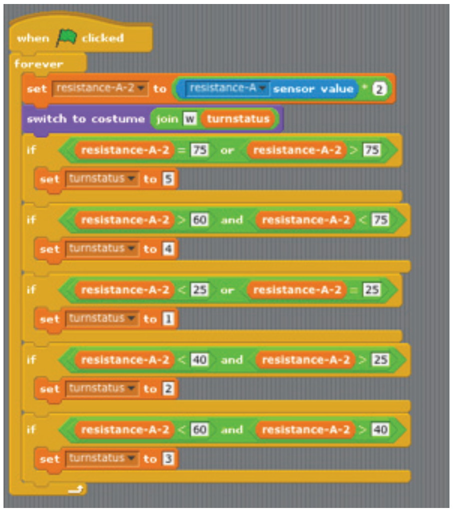


```
when [flag] clicked
forever
  set resistance-A-2 to (resistance-A sensor value * 2)
  switch to costume (join W turnstatus)
  if <(resistance-A-2 = 75) or (resistance-A-2 > 75)>
    set turnstatus to 5
  if <(resistance-A-2 > 60) and (resistance-A-2 < 75)>
    set turnstatus to 4
  if <(resistance-A-2 < 25) or (resistance-A-2 = 25)>
    set turnstatus to 1
  if <(resistance-A-2 < 40) and (resistance-A-2 > 25)>
    set turnstatus to 2
  if <(resistance-A-2 < 60) and (resistance-A-2 > 40)>
    set turnstatus to 3
```

11. Finally you may have noticed something funny about how the steering wheel works. It's backwards! You have to change the block of code that sets the variable "turnstatus" so that instead of 5 it sets "turnstatus" to 1 and vice versa. This is because the "slider" sensor on the board is hooked up the reverse of how the dial works.



```
when [flag] clicked
forever
  switch to costume (join W turnstatus)
  if <(slider sensor value = 75) or (slider sensor value > 75)>
    set turnstatus to 1
  if <(slider sensor value > 60) and (slider sensor value < 75)>
    set turnstatus to 2
  if <(slider sensor value < 25) or (slider sensor value = 25)>
    set turnstatus to 5
  if <(slider sensor value < 40) and (slider sensor value > 25)>
    set turnstatus to 4
  if <(slider sensor value < 60) and (slider sensor value > 40)>
    set turnstatus to 3
```



```
when [flag] clicked
forever
  set resistance-A-2 to (resistance-A sensor value * 2)
  switch to costume (join W turnstatus)
  if <(resistance-A-2 = 75) or (resistance-A-2 > 75)>
    set turnstatus to 5
  if <(resistance-A-2 > 60) and (resistance-A-2 < 75)>
    set turnstatus to 4
  if <(resistance-A-2 < 25) or (resistance-A-2 = 25)>
    set turnstatus to 1
  if <(resistance-A-2 < 40) and (resistance-A-2 > 25)>
    set turnstatus to 2
  if <(resistance-A-2 < 60) and (resistance-A-2 > 40)>
    set turnstatus to 3
```

12. Below is an example of how my code changed. There are multiple ways you can do this, see how many other fixes you can figure out.

Don't worry about being too rough with your prototype steering wheel. If it breaks, don't worry; you can always rebuild it, only better and stronger. Maybe next time you want to add a horn? Or a shifter? Or a gas and brake pedal?