

# Adaptive Cruise Control

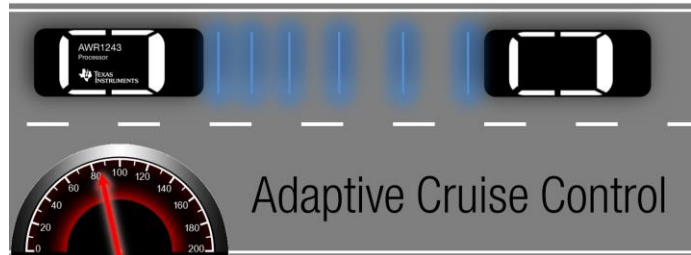
## Teacher Notes & Answers

7 8 9 10 11 12



## Introduction

Basic cruise control is where the car's computer automatically adjusts the throttle so that the car maintains a constant speed. This is a great feature when the car is travelling on a straight open highway with no other vehicles nearby. Problems occur when the car negotiates a corner, travels down a steep hill or encounters another vehicle travelling in the same direction but somewhat slower. Adaptive cruise control accounts for all of these alternatives. In this activity you will learn how to control the speed of a vehicle (TI-Innovator Rover) based on data being collected from its sensor(s).



In this activity you will use the sensor on the front of Rover to make adjustments to Rover's speed.

To see some of the features of Adaptive Cruise Control, check out the video: <http://bit.ly/AdaptiveCruise>

### Teacher Notes:

The video makes a nice introduction to the various types of 'adaptive cruise control' and a constant point of reference as students work through the activity. It is assumed that students have completed some basic programming on TI-Nspire prior to commencing this activity. The 10 Minutes of Coding section on the Texas Instruments Australia website is a great place for students to start.

<https://education.ti.com/en-au/activities/ti-codes>

Students should do all work with Rover on the floor consisting of a smooth, hard and clean surface. Cardboard boxes provide good obstacles for Rover to 'see'. This too can generate discussion. What if the objects are not smooth and flat? How might the sound waves bounce when they strike such a surface? What implications does this have for 'real' autonomous vehicles? What sort of waves should they be using?

## Reading Rover's Sensor

Some Adaptive Cruise Control (ACC) systems will bring the vehicle to a stop if the vehicle in front has already stopped and then resume moving as the vehicle in front moves off. This is the aim of the first part of this investigation. To achieve this result you will need to know some basic commands:

- Send "RV FORWARD"
- Send "READ RV.RANGER"
- Send "RV STOP"
- If ... Then



Create a new TI-Nspire document and insert a program titled:

ACC

The first line of any Rover program starts with:

Send "RV CONNECT"

The next step is to read the Ranger on the front of Rover.

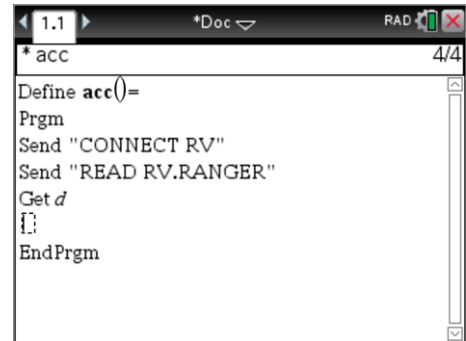
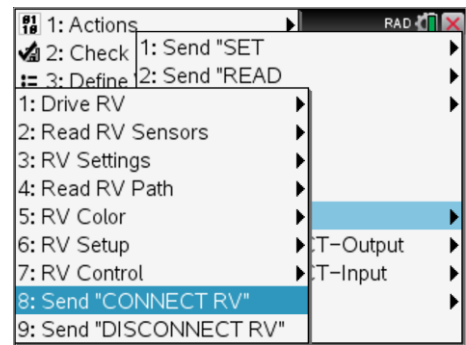
Hub > Rover (RV) > Read RV Sensors >

Send "READ RV.RANGER"

Reading "Ranger" is not enough; the result must also be returned to the calculator and stored.

Get d

Note that 'd' is simply a variable where the result will be stored. For simplicity it has been called *d* to represent the distance.



### Tip



"Display" commands are useful tools to make sure your program is working as expected. To see what your program is doing it is a good idea to place the following command after the 'get' statement:

DispAt 1,"Distance: ",d

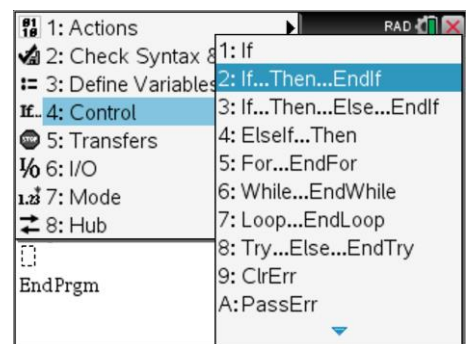
If Rover is too close to the vehicle in front it should remain stationary. If on the other hand the vehicle is more than 20cm away it may be appropriate for Rover to move.

This can be coded as:

```
If d>0.2 Then
  Send "RV Forward 1.5"
Endif
```

The Drive command is available from:

Hub > Rover (RV) > Drive RV > FORWARD



Place Rover on a smooth flat surface, preferably the ground (so it will not fall off the table). Place a box in front of Rover, approximately 15cm away. Press Ctrl + R to save and run the program.

**Question: 1**

Does Rover do anything?

Answer: No. As the distance is less than 20cm the "IF" statements tells Rover to Stop. As only one measurement is made Rover will not check again to see if the box has been moved.

**Question: 2**

Move the box so that it is approximately 40cm away. Press [Enter] to run your program again. What happens this time?

Answer: Rover moves forward approximately 15cm regardless of how much further the box is placed, provided the box is at least 20cm away.

**Question: 3**

Change the: Send "RV FORWARD" command to read: Send "RV FORWARD eval(10d-0.2)" Press Ctrl + R to run the program again. Try several different locations for the box and explain what is happening.

Answer: Rover moves forward and stops very close to the box. (On some occasions Rover may hit the box) The formula:  $10d - 0.2$  uses the distance from Rover (measured in metres) and converts it into a forward command which is in multiples of 10cm but removes just 2 centimetres from the overall distance. So if the measurement is slightly inaccurate Rover can hit the object in front. The 0.2 therefore describes just how close Rover stops in front of the object.

If the box is moved during this time it does not change how far Rover moves ... the distance is still only measured once.

So far the program only takes one look at the object in front. What if the object moves? Rover must repeatedly look at the object in front and adjust its movements accordingly.

Two adjustments need to be made to the program.

In the IF statement, an 'ELSE' option is included:

```
Else
  Send "RV STOP"
```

Now if Rover is travelling towards the box and the box gets closer than 0.2m away, Rover will stop immediately.

The second change is to make the program run repeatedly. This will be done with a "WHILE" loop.

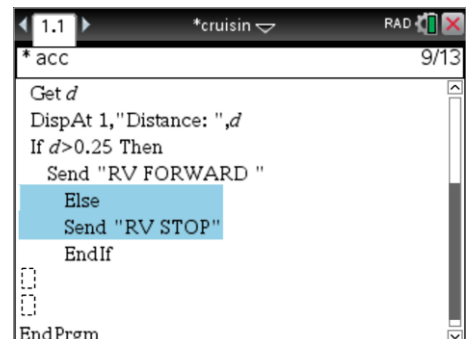
Use the SHIFT key to highlight all the programming code from:

```
Send "READ RV.RANGER"
...
EndIf
```

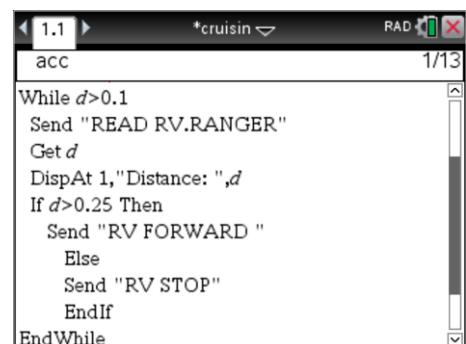
From the menu select:

Control > While ... EndWhile

Use the condition shown opposite: " $d > 0.1$ "



```
1.1 | *cruisin | RAD | 9/13
*acc
Get d
DispAt 1, "Distance: ", d
If d > 0.25 Then
  Send "RV FORWARD "
Else
  Send "RV STOP"
EndIf
EndPrm
```



```
1.1 | *cruisin | RAD | 1/13
acc
While d > 0.1
  Send "READ RV.RANGER"
  Get d
  DispAt 1, "Distance: ", d
  If d > 0.25 Then
    Send "RV FORWARD "
  Else
    Send "RV STOP"
  EndIf
EndWhile
```

**Warning**

The loop will only be active 'while' the value of  $d$  is greater than 0.1. An additional statement can be added to the program before the start of the While loop:  $d:=1$ .

Alternatively the value of  $d$  can be set from the Calculator application prior to running the program.

**Question: 4**

Run the new version of the program and explore what happens whenever the box is moved away or toward Rover.

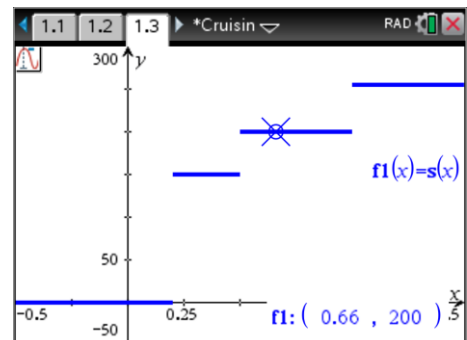
**Answer:** The inclusion of the While loop means that Rover is continually checking the Ultrasonic sensor. If the box is moved a new location is recorded and Rover responds accordingly. This is more like the adaptive cruise control where the vehicle may come to a complete stop if the vehicle in front stops, automatically resume when the vehicle in front moves again. The program stops completely if the distance to the object is less than 1cm.

**Extension – Adjusting the Throttle****Teacher Notes:**

As the subheading suggests, this part of the activity changes Rover's speed depending on the distance to the vehicle (object) in front. Rather than using a set of nested "IF" statements a piecewise function is used. The piecewise function helps introduce some more advanced mathematics to students and helps show the relationship between programming and higher level mathematics.

One significant advantage of the Piecewise function is that it can be graphed (Graph Application) and then traced so that students can see how it works.

Note that the function is defined in terms of  $x$  so that it can be graphed easily, but also to reinforce the concept that  $x$  is a variable and the 'quantity' (distance) that is currently stored in  $d$  will be placed in ' $x$ ' or "substituted into the function".



The program so far has Rover travelling at full speed when the road ahead is clear and stopping when an obstacle is detected. Adaptive cruise control fills provides more flexibility by gradually slowing as an object appears closer. It is possible to adjust the speed of Rover's motors gradually using slightly different commands.

"SET RV MOTORS LEFT # RIGHT # "

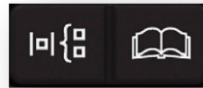
The # symbol used here represents a numerical value between -255 and +255 where the negative indicates counter-clockwise (CCW) rotation of the motor and positive is clockwise (CW). It must be noted however that 'small' values will fail to provide sufficient power to rotate the motors. A series of "IF" statements could be used to vary the power to each motor, however in this activity a "Piece-wise function" will be used.

**Teacher Notes:**

The possible values for LEFT and RIGHT are based on a 16 bit – binary number producing 256 states.

Navigate to a Calculator application. A rule or mathematically speaking, a 'piecewise-function' will be defined that determine the amount of power sent to each motor.

### Actions > Define



Type in the function name:  $s(x) =$

From the templates menu select the piecewise template.

In the piecewise template select "4" pieces.

The image opposite shows that the definition for the first two pieces. When  $x$  (distance) is larger than 1m the function will return the value 255, this means the motor will receive full power and rotate CW.

If the distance is between 0.5m and 1m then the motor will receive 200 units of power, almost 80%.

Complete the definition so that rover's motors will receive 150 units of power when the distance is between 0.2m and 0.5m; and no power if the distance is less than 0.2m.

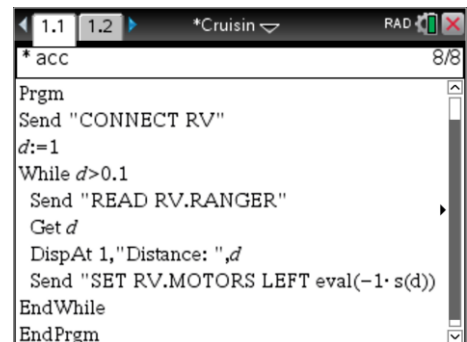
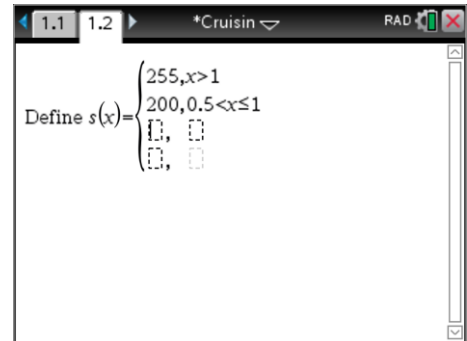
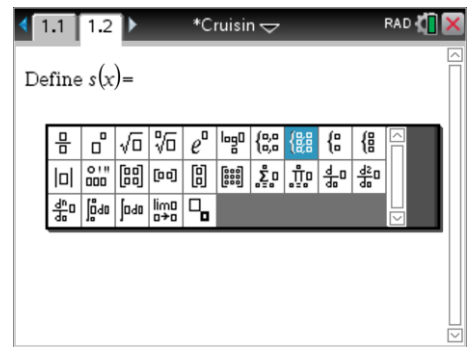
Return to the ACC program. Remove the entire IF statement and replace it with the SET RV.MOTORS command.

Hub > Rover (RV) > RV Control > SET RV.MOTORS

The LEFT and RIGHT commands can be obtained from the Rover menu – RV SETTINGS. The 'eval' command is an abbreviation of the word 'evaluate' and can be obtained from the HUB menu.

Send "SET RV.MOTORS LEFT eval(-1\*s(d)) RIGHT eval(s(d))"

Press Ctrl + R to save and run the program.



### Question: 5

How does Rover respond when an object is placed or moved along in front as it is driving?

**Answer:** Depending on the distance, Rover may speed up, slow down or stop. The distance increments are defined by the piecewise function.

### Question: 6

The evaluation contained in the LEFT argument of the RV.MOTORS command multiplies the result by negative 1. What does this do and why is it necessary?

**Answer:** The left hand motor must turn in the opposite direction since it is effectively facing the opposite way. The negative value makes the motor turn counter-clockwise.

### Question: 7

The magnitude of the power sent to via the LEFT and RIGHT arguments is the same. How would Rover move if these values were different? How could this be used for obstacle avoidance?

**Answer:** If the motors receive different amounts of power it will cause Rover to turn towards the left or right, this could be used to help Rover 'change lanes' or to navigate around an object.