

Measuring Motion

Student Activity

7 8 9 10 11 12



TI-Nspire



Coding



Student



60 min

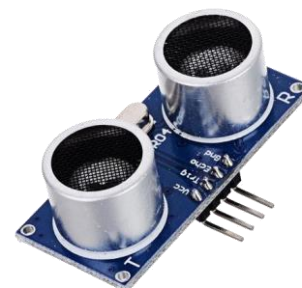
Ultrasonic Ranger (introduction)

Have you ever wondered how a car reverse sensor works? Most modern cars have reverse sensors fitted to the rear of the vehicle; many are now being produced with sensors on the front and side also. In this activity you will learn how to program your own reverse sensor prototype using the TI-Innovator Hub. The sensor is capable of measuring the distance to an object by transmitting (T) and the receiving (R) a series of ultrasonic pulses.

$$\text{speed} = \frac{\text{distance}}{\text{time}}$$

The speed of sound in air is approximately 330m/s. So if the emitted pulses take 0.01 seconds to return then the object is 3.3 metres away. The ultrasonic sensor used in this activity automatically converts the signal return time into a distance.

Once this distance has been measured, audible and visual warnings need to alert the driver that an object is close to the sensor. Both warning types need to change as the object gets closer. Why might it be useful to include both visual and audible responses?



Teacher Notes:

While this activity uses the calculator, the TI-Nspire computer software can also be used. The computer to calculator cable also connects to the TI-Innovator hub.

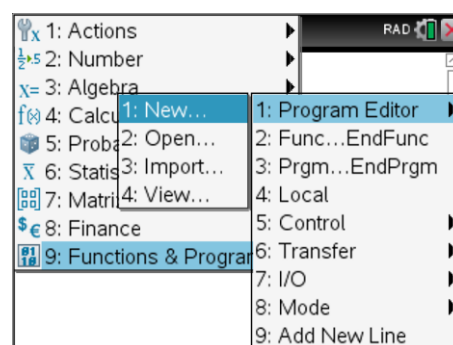
Connecting and Reading the Ranger

Instructions:

Start a new document; insert a Calculator application and a program.

Call the program: **Ranger**

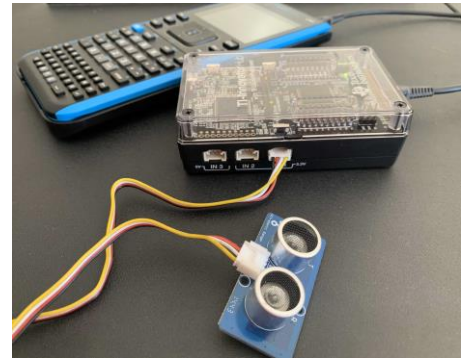
By default the Calculator application and Program will be shown on the same page. Press **Ctrl + 6** to ungroup the applications then navigate to page 1.2.



Connect the ultrasonic ranger sensor to input port 1 on the TI-Innovator using the ribbon cable.

Connect the TI-Innovator to the calculator using the USB cable.

The **B** end of the USB cable goes to the TI-Innovator. The letter **B** appears next to the data port on the TI-Innovator.

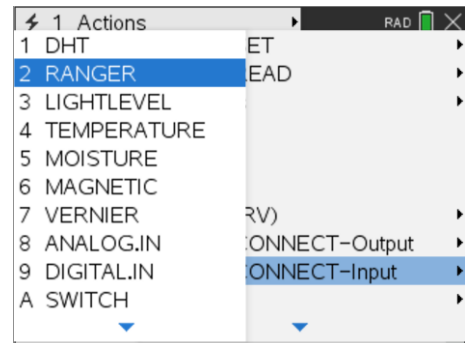


The first part of the program needs to identify the location of the ultrasonic ranger.

[menu] > Hub > Send "CONNECT-Input" > RANGER

As more than one Ranger can be connected to the TI-Innovator it is important to give it a number.

Enter a '1' to label this as Ranger 1.

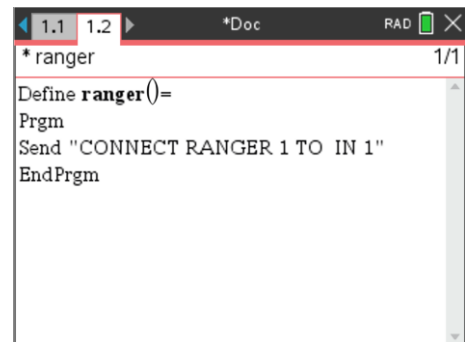


The next step, in the same instruction, requires us to identify where the Ranger sensor has been connected.

[menu] > Hub > Settings > TO

Next, select the port that the ultrasonic ranger has been connected to:

[menu] > Hub > Ports > IN 1



Running the program to date would let the microprocessor in the TI-Innovator know what sensor has been connected and to which port it has been connected. The next set of instructions will get the TI-Innovator to read the distance, transfer that measurement to the calculator and then display it on the calculator screen.

On a new program line:

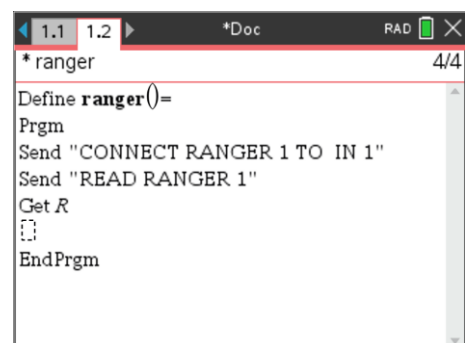
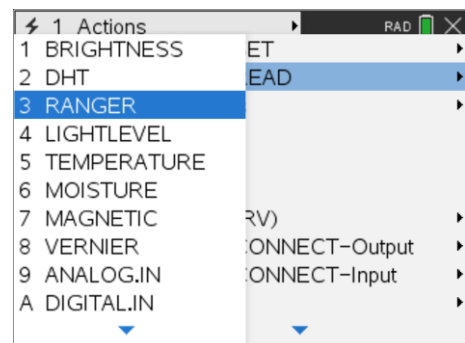
[menu] > Hub > Read > RANGER

Once this command has been entered, don't forget to let the program know which Ranger you need to read. (Ranger 1)

The TI-Innovator has captured the measurement; it now needs to be transferred to the calculator. On a new program line:

[menu] > Hub > Get

This command does as it suggests, it gets the distance value from the TI-Innovator. The command then needs to store this value in a variable on the calculator. In the example shown opposite an 'R' has been used to store the value returned from the Ranger.



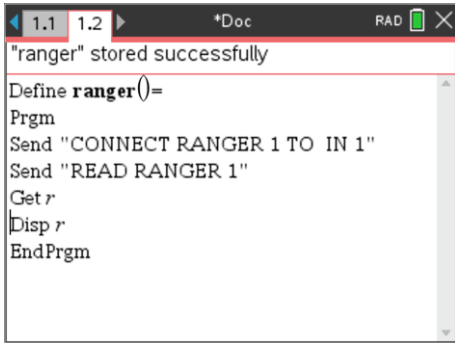
Now that the value is stored on the calculator it can be retrieved and used in the Calculator application (Page 1.1) and displayed immediately.

 > I/O > Disp

Text can be added to the Display command:

Disp "Distance",R

Alternatively the program can simply return the distance.



```
"ranger" stored successfully
Define ranger()=
Prgm
Send "CONNECT RANGER 1 TO IN 1"
Send "READ RANGER 1"
Get r
Disp r
EndPrgm
```



Programming commands can be typed directly rather than using menus. Whilst most commands are not CASE sensitive, TI-Innovator commands that include quotation marks are both case and space sensitive. Using the menus for these commands is recommended.

The program is now ready to run.

Press: Ctrl + R

This will launch a new Calculator application and paste the program name ready to run.

The screen opposite shows a sample result. The ranger sensor has detected an object 1.92 metres away.



```
ranger()
-----
1.922688
-----
Done
```

Repeated Reading

A single reading for the sensor is great, but for most applications the sensor is triggered and continues to read until stopped by either a specific value or event. In the case of a car reverse sensor it is triggered by the car being put in reverse and switched off when the car is taken out of reverse.

Navigate back to the Ranger program.

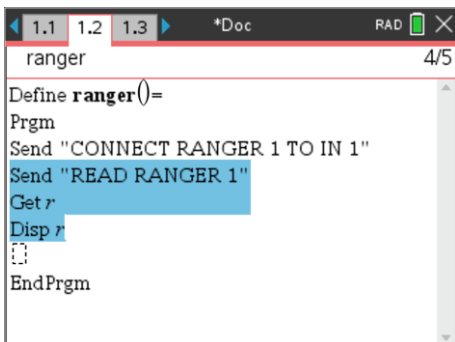
Use the Shift and Arrow keys to highlight the commands that measure and display the ultrasonic ranger readings, these are the commands that will be repeated.

The original naming and location of the sensor does not need to be repeated.

A 'while' loop will be used to repeat the sensor readings. With the text highlighted:

 > Control > While...EndWhile

The **While** and **EndWhile** commands effectively bracket the instructions to be repeated.



```
ranger 4/5
Define ranger()=
Prgm
Send "CONNECT RANGER 1 TO IN 1"
Send "READ RANGER 1"
Get r
Disp r
EndPrgm
```

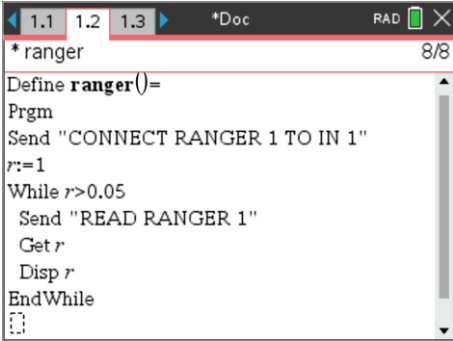
The condition that determines when the loop will operate is the distance 'r'.

Include the condition for the while loop:

While $r > 0.05$

An additional line needs to be added immediately before the While loop to ensure that the value of R is satisfied initially.

R := 1



```
* ranger
Define ranger()=
Prgm
Send "CONNECT RANGER 1 TO IN 1"
r:=1
While r>0.05
  Send "READ RANGER 1"
  Get r
  Disp r
EndWhile
[ ]
```

Question: 1.

Run the Ranger program. Move your hand towards and away from the sensor.

a) At what distance does the program stop measuring?

Answer: The loop terminates when a measurement is recorded below 0.05m. This means the value of r could be as low as zero when the loop is terminated.

b) Suggest a way to improve the output of the program.

Hint: Check other display options in the menu.

Answer: As the measurements scroll down the screen the DispAt command is better for this output.

Example: DispAt 10,0,"Distance",r

A more extreme example would be to use the drawing canvas to 'show' the object.

Audible Warnings

Vehicle reverse sensors generally have an audible system that provides an indication of the distance to surrounding objects. As the vehicle gets closer to the object the frequency or volume of the sound may increase. Mathematically speaking this is called an 'inverse' relationship. As one value is decreasing (distance to the object) the output is increasing (sound frequency).

Navigate back to the Ranger program.

The aim here is to provide an audible warning if the distance between the ultrasonic ranger and an object is small.

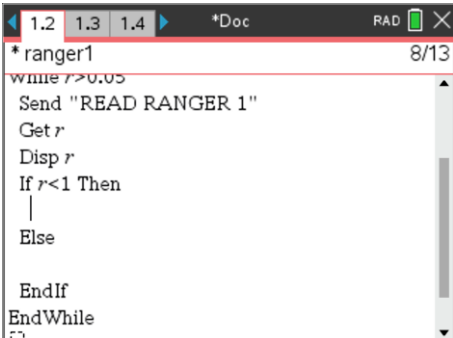
Add a new line between the display command and the end of the while loop. The condition will start here:

"IF" the distance is small.

 > **Control** > **If...Then...Else...EndIf**

The first condition is for the distance to be less than 1m:

r < 1



```
* ranger1
while r>0.05
  Send "READ RANGER 1"
  Get r
  Disp r
  If r<1 Then
  |
  Else
  EndIf
EndWhile
[ ]
```

If the distance between the sensor and the object is less than 1m then a sound needs to be generated:

 > Hub > Send "SET > SOUND

The syntax for the sound command allows for a frequency and time if required. In this case the sound will be calculated (evaluated) from an expression.

 > Hub > eval(

The expression to be evaluated is: $100 / r$

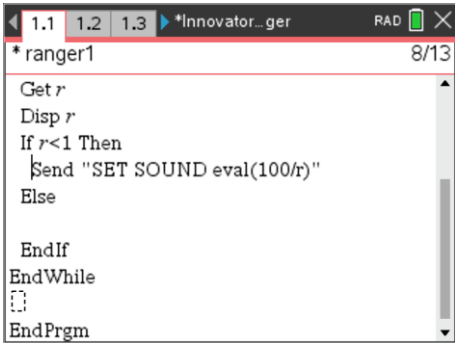
If the distance is not less than 1m, the sound can be turned off. The **Else** section of the statement is where this instruction needs to be located:

 > Hub > Send "SET > SOUND

The sound needs to be turned off when $r \geq 1$:

 > Hub > Settings > OFF

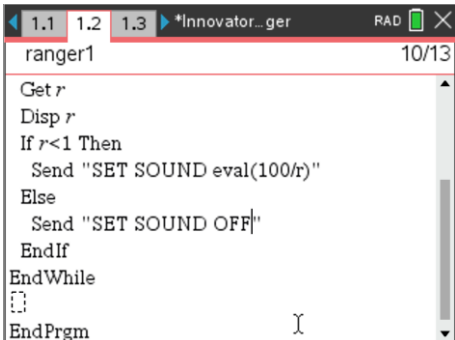
The program is now ready to run. Press Ctrl + R and run the program.



```

1.1 1.2 1.3 *Innovator...ger RAD
*ranger1 8/13
Get r
Disp r
If r < 1 Then
  Send "SET SOUND eval(100/r)"
Else
EndIf
EndWhile
EndPrgm

```



```

1.1 1.2 1.3 *Innovator...ger RAD
ranger1 10/13
Get r
Disp r
If r < 1 Then
  Send "SET SOUND eval(100/r)"
Else
  Send "SET SOUND OFF"
EndIf
EndWhile
EndPrgm

```

Question: 2.

Run the Ranger program. Move your hand towards and away from the sensor.

- a) At what distance does the sound cut in / out?

Answer: The sound cuts in when an object is closer than 1m and stops shortly after the program ends or when the object is further away than 1m.

- b) Describe how the sound changes as an object gets closer to the sensor?

Answer: The frequency gets higher as the object gets closer. Sample calculations are appropriate here. Example: 0.1m to object produces a 1000Hz sound.

- c) Compare the sound changes to that of a vehicle reverse sensor.

Answer: Most vehicle sensors produce a 'beep' rather than a continuous sound. The beeps get closer and closer together as the object gets closer to the sensor.

Teacher Notes:

It is possible to create a 'beep' rather than a continuous sound. The 'sound' command can also incorporate a time. By placing the sound command in a small loop it will become more like a 'beep'. Note that this could become a separate program "Beep()" and be called up from the Ranger program. This makes it possible for students to complete their Ranger program before focusing on the sound.

Question: 3.

The mathematical formula used to change the sound as an object gets closer to the sensor can be changed from the current formula: $100/r$ to a more appropriate one. The formula may be defined as a function or entered directly into the evaluate command: eval()

An appropriate frequency range for the sound could be from 100Hz to 2000Hz.

Create a new formula for distances from 1m down to 0.05m to generate the above frequency range.

Answer: Answers to this question will vary. Students can use a linear function, reciprocal, exponential or a range of other functions that decrease over the selected domain. Students may be encouraged to experiment with a number of different functions, perhaps even hearing the difference between the curvature (concave up / down).

Points as (r, f) r = distance and f = frequency should include: (1, 100) and (0, 2000).

Note that it is important to consider that r could become equal to zero given that the While loop is terminated *after* a reading where $r < 0.05$ is obtained.

$$\text{Samples: } f = 2000 - 1900r \text{ or } f = \frac{1}{r - 1.0005} + 2001 \text{ (approx) or } f = 2000 \cdot \left(\frac{1}{20}\right)^x$$

Visual Warnings

Vehicle reverse sensors are also accompanied with visual support. While it is not possible to connect a camera to the TI-Innovator, it can produce a light signal that changes colour depending on the distance to the object. The RGB diode on the TI-Innovator hub requires a mixture of three inputs:

Red 0 to 255

Green 0 to 255

Blue 0 to 255

These instructions are provided as a guide. You will be invited to change the colour settings later.

If the distance to a nearby object is less than 1m **then** the colour of the warning light will progressively change from green to red.

This command needs to be entered into the **If...Then..Else...EndIf** statement.

 > **Hub > Send "SET > COLOR**

Red is evaluated as: $255 - 255r$

Green is evaluated as: $255r$

Blue is left as zero: 0

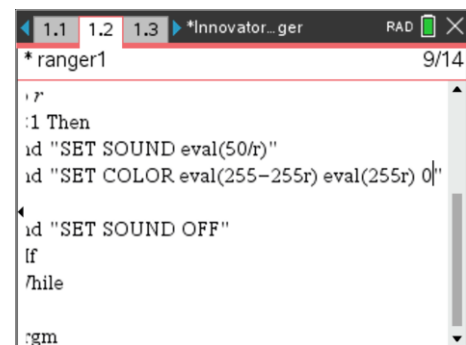
If the distance to the nearby object is greater than 1m then the colour of the warning light will remain green.

This instruction needs to be entered into the appropriate section of the **If...Then...Else...EndIf** command.

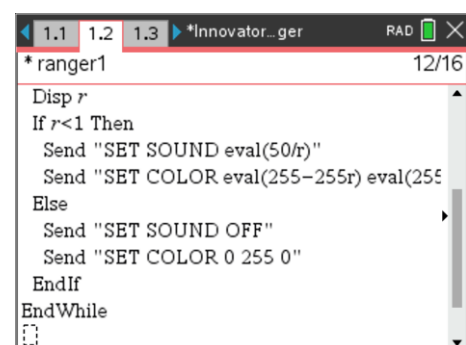
 > **Hub > Send "SET > COLOR**

Set the colours as: $0 \ 255 \ 0$

At the end of the program it is appropriate to insert another SET COLOR command to make sure the diode is turned off completely.



```
* ranger1 9/14
r
:1 While
:1 Then
  id "SET SOUND eval(50/r)"
  id "SET COLOR eval(255-255r) eval(255r) 0"
:1 Else
  id "SET SOUND OFF"
if
/while
:gm
```



```
* ranger1 12/16
Disp r
If r<1 Then
  Send "SET SOUND eval(50/r)"
  Send "SET COLOR eval(255-255r) eval(255r) 0"
Else
  Send "SET SOUND OFF"
  Send "SET COLOR 0 255 0"
EndIf
EndWhile
[]
```

Question: 4.

When the distance from the sensor to the object is 1m, determine the colour of the RGB diode.

Answer:

$$\text{Red} = 255 - 255r. \quad \therefore \text{Red} = 0$$

$$\text{Green} = 255r. \quad \therefore \text{Green} = 255$$

$$\text{Blue is independent of } r \quad \therefore \text{Blue} = 0$$

Question: 5.

Design a new colour range for the sensor. Remember the value of R potentially varies from 0 to 1 when the colour needs to change. Each colour red (R), green (G) and blue (B) can vary between 0 (Off) and 255 (On). Values outside this range will generate an error.

Record your equations and the corresponding colours generated as an object moves closer to the sensor.

Answer: Answers will vary. Students can define the functions using functional notation, including a piecewise function so as to avoid values that are not permitted. The level of complexity can be increased depending on the students involved. In support of their answers student may also want to consider the overall brightness and argue that this might involve the total output to be constant.

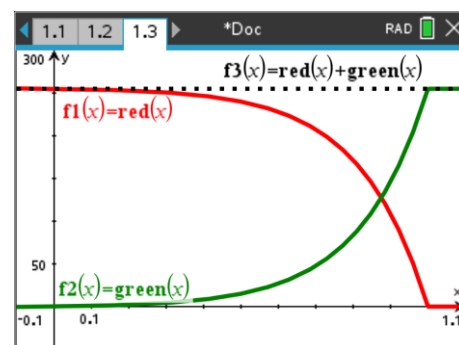
Example: (non-linear)

```

Define red(x) = { 256 - 256^x, x < 1
                0, x >= 1
Done

Define green(x) = { 256^x - 1, x < 1
                  255, x >= 1
Done

```

**Question: 6.**

Suggest ways that the ranger program could be improved or made more like that of a motor vehicle set up.

Answer: Answers will vary. One of the most obvious recommendations within the scope of the TI-Innovator set up is to have a 'beep' rather than a continuous noise. Students should comment that a beeping sound is more likely to be heard over a continuous sound and that higher frequency sounds are potentially more irritating and likely to draw attention than lower frequencies. Lower frequencies also use more energy and generally require larger output devices such as speakers.

Colour is another obvious option. Students may suggest a brighter light or perhaps an array. Students could also make a 'radar' type shape with cut-outs as a cover for a box and place the innovator inside the box so as to give the impression it is producing the radar image.

Teacher Notes:

For a more extensive option, students could use the breadboard outputs to create their own array of diodes. The number of LED's that switch on can be programmed using either defined functions or IF statements, in this way students can learn more about 'piecewise' functions and how they resemble the programming command IF ... THEN ... ELSE.