Unit 5: Rover's Sensors | **Skill Builder 2: Off the Wall**

In this lesson, you will use the TI-Innovator Rover's Ultrasonic Ranger to avoid an obstacle.

Objectives:

- Read the distance to an object in front of Rover
- If the distance is 'too small', turn around and continue driving

Teacher Tip: In this project, students will encounter a fairly common issue when working with Rover: The TI-84 works much faster than the Rover. In the LED lessons of the last unit, the calculator had to wait until Rover finished a forward or a turn command before issuing the next instruction in the code.
The same problem arises here, but the effect (error) is more dramatic. The Rover struggles between moving, turning, and stopping. Adding another wait (in the right place) solves the problem.

As Rover is driving, your program can use the Ultrasonic Ranger to detect an obstacle ahead and interrupt Rover to take another route. In this project, the Rover will drive back and forth between two opposite walls.

Teacher Tip: This project works well using hardcover books standing on end. Be sure the books are perpendicular to the Rover both vertically and horizontally. Otherwise, the Ranger will not 'see' the obstacle because the (silent) tone does not bounce (echo) directly back to the sensor.

1. Begin a new Python program using the Rover template.

   Get Rover moving using:

   rv.forward (100)

   Yes, that's 10 meters, or about 39 feet, but there will be an obstacle in the way.

2. Read the distance in front of Rover:

d = rv.ranger_measurement()

As long as there's room to proceed, keep going. But if Rover is too close to an obstacle, turn around. How close is 'too close'? That is for you to determine.

3. Add a **while** loop to continue monitoring the distance as long as the obstacle is 'far away':

while d > ??? :

(Replace the question marks (???) with a 'small' number.)

♦♦d = rv.ranger_measurement()

For now, just stop Rover when the loop ends.

rv.stop()

Notes: **while** is on **<Fns…> Ctl**
> is on **[test]** (**[2nd] [math]**)
rv.stop() is on **[math] ti_rover… Drive**

Note: **rv.stop( )** *executes as a 'pause' and then* **rv.resume( )** *continues with the Rover movements in the queue. But, if another movement command is executed after* **rv.stop( )**, *then the movement queue is cleared.*

Test your program now by pointing Rover towards a 'wall' (less than 40 feet away) and **<Run>** the program. Rover should stop before hitting the 'wall'.

Teacher Tip: Your hand works well as an obstacle, too.
rv.stop() executes as a pause and then **rv.resume( )** *continues* with the Rover movements in the queue. If another movement command is executed after **rv.stop()**, then the movement queue is cleared. This again is a bit different than programming TI-Innovator™ Rover in TI-Basic.

The principle above: Get Rover moving, and then 'monitor' the distance. The drive command is queued, but the **measurement()** function is not so the ranger can continuously take a reading while Rover is moving.

4. When Rover encounters the obstacle, the Rover should turn around. After Rover stops, have the Rover turn around using:

rv.left(180) or **rv.right(180)**

5. After turning around, repeat the entire process again by making the whole program (so far) part of a **while not escape( )** loop. Remember that Rover is moving, so it might be tricky to press the **[clear]** key. Place the statement

while not escape( ):

at the beginning of your code (after the imports).

Indent all the statements *below* this **while** statement to make them the **while** **block**. On each line select **<Tools> Indent**. Note that one line is already indented because it's part of another while loop. When one loop is embedded inside another the structures are 'nested'.

6. Before running your program, add one more statement to the main loop. After the turn around statement (ours is **rv.right(180)**) add

rv.wait_until_done().

This is necessary because the turn command is part of the drive queue so the program will immediately proceed to read the ranger measurement which is *not* in the drive queue. Since the wall is still there, Rover will turn around again right away. The program must wait until Rover turns all the way around before driving forward again.

<Run> the program with Rover between two 'walls' or use your hand as a wall. Rover needs about 20cm to turn around since the stern is longer than the bow. **while d > 0.2** gives Rover room to turn around. Try the program with and without the last **wait**… statement.