micro:bit with expansion board **Intruder Alert!**

The Ultrasonic Ranger can be used to detect a change in distance. This concept can be used to build an alarm-type system that triggers lights and sounds when the distance is not within an allowed range. This activity develops the basic tools for an alarm system and the idea can be adapted to lots of different applications.

This alarm will trigger when a small box containing the ranger is opened.

---

1. Before you begin, be sure that:
   - your **micro:bit** is connected to your TI-84Plus CE Python
   - your **micro:bit** is inserted in the **expansion board**.
     (Grove shield shown here)
   - *the Grove accessories you will need are coming up…*

2. Your micro:bit should look like this when it has power from the TI-84 Plus CE Python graphing calculator:
   Recall that the display on the **micro:bit** is showing the TI logo, an icon of the state of Texas with a bright spot near Dallas, the home of **Texas Instruments, Inc**.

3. This activity uses three external devices: the ultrasonic ranger, a speaker, and the LED that was used in the Light Switch activity. On the LED board, remember to turn the yellow dial (a potentiometer) all the way to the right to get the maximum brightness. Make the following micro:bit connections:
   pin0: speaker
   pin1: ranger
   pin2: LED

   Note micro:bit version 2 has an on-board speaker. An external speaker can also be used. We will also use the micro:bit display board to flash lights in a custom pattern.

4. Start a new Python program. We named it **INTRUDR1**. Add the micro:bit module to your code using **<Fns…> Modul <Add-On> micro:bit**:

   from microbit import *

   Tip: If the message 'micro:bit not connected' ever appears, just unplug the micro:bit and plug it in again (reset).

5. With the **micro:bit** module in the editor, add some additional components using the **[math] micro:bit…** menus.
   Grove Devices will add both the **mb_grove** and **mb_pins** modules.
   Music adds both **mb_music** and **mb_notes** modules.

6. The program is divided into two parts: a loop that 'watches' the distance and another loop that produces the alarm effects: lights and sounds. There are two **while** loops in the program…

7. We've added a statement to clear the micro:bit display and to show that the alarm is set on the calculator screen.

The first loop simply monitors the distance from the ranger that is connected to pin1 on the expansion board:

dist = grove.read_ranger_cm(pin1)
while dist < 10:
dist = grove.read_ranger_cm(pin1)

Get **grove.read_ranger_cm()** from **[math] grove…** Complete the statement by adding the variable name and the pin variable.

Recall that the ranger will be inside a small (<10cm high) closed box with a lid. The ranger (not the micro:bit) is placed on the inside bottom of the box facing upward to detect when the box cover is opened. If the lid is closed this loop will continue. When the lid is opened this loop ends.

8. **Sounding the alarm:**

The second loop creates the alarm effects. If you have an external speaker and LED you can use those to produce lights and sound. You can also use the on-board display and speaker (*speaker is on micro:bit version 2 only*).

We will use the **[clear]** key to end the program:

while not escape():

9. To make the LED turn on and an alarm to sound, use the functions

grove.power(pin2, 100)
music.pitch(440, 100)

grove.power( ) is found on **[math] grove… Output** menu
pin2 is the connection for the LED
100 is full intensity (brightness)

music.pitch( ) is found on **[math] music…**
440 is 440Hz which is middle A on the music scale
100 is 0.1 seconds (100 milliseconds)

10. To make the LED blink and the alarm to sound 'alarming' add two more statements to turn the LED off and change the musical note:

grove.power(pin2, 0)
music.pitch(220, 100)

You can choose other frequencies for the musical tones.

11. Let's add a flashing **display** effect on the micro:bit using
display.show( ).
Remember to *first* import **mb_disp** for the display functions.

The display can show any of the images in the built-in collection or you can make your own custom images using the **Image( )** function (shown here) found on **[math] display…**.

These image definitions can go at the top of your program (below the import section and before any loops).

12. Here are two sample custom images:
The variable names are **img_alarm1** and **img_alarm2**
The argument is a "string" of 25 digits (each from 0 to 9 indicating brightness) for each of the 25 pixels on the display. Each row of 5 values is separate by a colon (:).

It is convenient to write the string on five separate lines to arrange the digits as they will appear on the display (in a 5x5 grid):

Image("90909:"
"09090:"
"90909:"
"09090:"
"90909")

But it is also possible to write the string in one long line:
Image("90909:09090:90909:09090:90909")
or with extra quotes:
Image("90909:""09090:""90909:""09090:""90909")
Python will concatenate the separate strings for you!

13. img_alarm1 lights up every other pixel on the display in a 'checkerboard' pattern.
    img_alarm2 does the opposite pattern.

14. Finally, to get the display to flash when the alarm goes on, add two statements to your alarm code to display the two images (at the right moments)

    display.show( ) is on **[math] micro:bit… display…**
    and just type the names of your custom image variables as the arguments.

    This image also shows the two optional arguments for display.show(): **delay** *is in ms,* **wait**=*True tells the system to wait until the image is displayed for the proper time before proceeding to the next statement.*

15. The effect is alarming!

    Note: the video shows a portable amplified speaker connected to the micro:bit using a custom cable. (plug-n-play!).

<see CE Intruder.mp4>