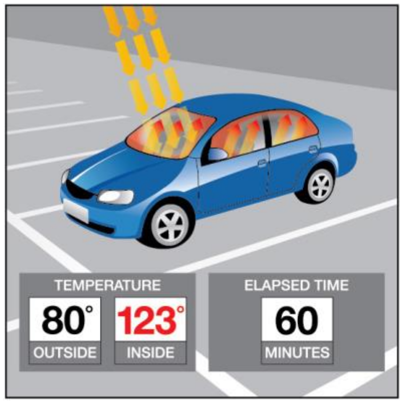
Pet Car Alarm

In this TI-Innovator™ project, you will explore the science behind the greenhouse effect and apply your knowledge to design a product to solve a real-world problem of pets dying due to owners leaving them in hot cars. You will have to utilize math skills, computer programming and engineering to design and build a smart pet alarm system for a model car. A car equipped with a "smart" pet alarm could prevent harm to a pet left inside a hot car by taking action to cool the interior and notify the owner of impending pet harm.



Background:

Pets suffer when left unattended in a car with the windows rolled up on a hot sunny day. The temperatures inside a car may reach greater than 40°F above the outside ambient temperature within an hour due to a greenhouse effect within the closed car. A car equipped with a pet-smart alarm could prevent harm to a pet left inside a hot car by taking action to cool the interior and notify the owner of impending pet harm.

Your Challenge:

Understand the science behind the greenhouse effect and use math, computer programming and engineering to design and build a pet-smart alarm system using a model car.

Activity Materials:

- TI-Innovator™ Hub
- TI-Nspire CXII
- Hall Effect (magnetic) Sensor
- Temperature Sensor(s)
- 2 White LEDs
- Continuous Servo Motor
- External Battery for TI-Innovator Hub
- Small magnet
- Model Car (or other container)
- Plastic Pet
- Cling wrap
- Tape
- Safety Scissors

Project Challenges

Skill Building Challenge 1: Write a program that will play two sounds for 1 second each in a For loop that repeats five times.
For example: sound.tone(440,1)

Skill Building Challenge 2a: Use a For loop to blink two external LED's.
Try to blink 30 times with 1 second on and 1 second off for each blink.

Skill Building. Challenge 2b: Try Challenge 2 once more using the .blink(frequency,time) function available for LED output devices. This time you can blink repeatedly without a For Loop.

Skill Building Challenge 3: Connect a continuous servo motor to the TI-Innovator Hub and cause it to rotate clockwise (CW) and then in the opposite direction, counterclockwise (CCW).

Skill Building Challenge 4: Connect a temperature sensor to the TI-Innovator Hub and display the temperature on the calculator.
Use a While loop to read and display temperature values until the escape key is pressed.

Skill Building Challenge 5: Connect the Hall effect magnetic proximity sensor, which determines if the south pole of a magnetic field is close to the sensor.

Display "Magnet is present" or "Magnet is not present" based on the reading of the Hall effect sensor and the position of the magnet.

Final Challenge: Develop a car alarm system that determines if a pet is present (magnet) and if the temperature inside the car is above a critical threshold before triggering the alarm.

Example Programming Statements for the project

Example	Behavior
`from ti_hub import *`	Imports all the functions in the ti_hub module for use in the program. The ti_hub module includes all the necessary additions needed for project.

`# defines a temperature sensor`
`# object named temp_sensor_inside`
	# at the beginning of a line denotes a comment. Comments are a "best practice" by programmers to annotate their code. Comment statements are ignored when the program is run. In the TI-Nspire CXII Python editor, [ctrl]+[T] toggles the statement of the current cursor location from a comment to a statement that will be run.

`sound.tone(440,1)`
`sleep(1)`
	Plays a tone through the TI-Innovator Hub speaker. In the example, plays the frequency 440 hertz (Hz) for 1 second. Following sound.tone with a sleep function assures that the tone will complete before the program begins the next sound function. Note: sound.tone() is available from the Hub Built-in Devices>Sound Outputs menu.

`left_headlamp=("OUT 1")`	Creates an LED object named **left_headlamp** connected to port OUT 1. led is available from the TI Hub > Add Output Device menu. Note: = is the Python operator for storing or assigning values to a variable.

`left_headlamp.on()`
`sleep(1)`
`left_headlamp.off()`
`sleep(1)`
	Turns on an LED object named **left_headlamp** then pauses the program for 1 second before turning **left_headlamp** off then pauses the program again for 1 second.
	Note: To see options for an object paste the object name from the var key menu then press the period key.

`sleep(.5)`	Pauses program for .5 seconds. sleep() is found on the Hub Commands menu.

`for n in range(10):`
` print(n)`
	Repeats the statements in the block ten times, printing the value of the index variable, **n**, as 0,1,2,…9. The index variable n starts at 0 and increases by 1 with each loop. If **n** is less than the stop value, 10, the loop continues to repeat.

`left_headlamp.blink(0.5,60)`	Blinks (turns on and off) an LED object named **left_headlamp** once every 2 seconds (one half a blink per second) for 60 seconds. Frequency is typically measured in Hertz, cycles per second. Note: To see options for an object paste the object name from the var key menu then press the "." key.

`window_motor=continuous_servo("OUT 3")`	Creates a continuous servo device object named **window_motor** connected to port OUT 3. Continuous Servo is available from the TI Hub > Add Output Device menu. The Servo motor requires 5 volts and must be connected to port OUT 3. An external battery must be plugged into the HUB PWR port and turned on before running programs that include the Servo motor. Note: = is the Python operator for storing or assigning values to a variable.

`window_motor.set_cw(20,1)`
`sleep(2)`
`window_motor.set_ccw(20,1)`
`sleep(2)`
	Sets the continuous servo motor named window_motor to turn clockwise at a speed of 20 (speed inputs range between 0 for off to 255 for maximum speed) for 1 second, then pauses the program for 2 seconds before moving on to set the motor to turn counter clockwise at a speed of 20 for 1 second. Follow the motor set functions with sleep() functions to assure that the motor completes the desired action before proceeding to the next function.

`temp_sensor_inside=temperature("IN 1")`	Creates a temperature sensor object named **temp_sensor_inside** connected to port IN 1. temperature is available from the TI Hub > Add Input Device menu. Note: = is the Python operator for storing or assigning values to a variable.
`temp_inside=temp_sensor_inside.measurement()`	Reads and stores the current measurement value of the **temp_sensor_inside** object into variable **temp_inside**. Note: **.measurement()** returns the current measured value of a sensor object. To see options for an object paste the object name from the var key menu then press the period key.

`text_at(3,"temperature inside car= "`
`+str(temp_inside)+" °C","left")`
	This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable **temp_inside** has a value of 26, the following is displayed on row 3, aligned to the left:
	temperature in car= 26 °C
	text_at() is available from the TI Hub>Commands menu.
	Note: The **str()** function converts a numeric value to a string. The + operator is used to join two strings. **str()** is available from the Built-ins> Type menu.
	Note: Degree, percent and other special characters are available from the ?! key menu in the lower right of the TI-Nspire keyboard.

`window="closed"`	Sets the value of a variable named window to be the text string, "closed". Variables can contain many different data types, including floating point numbers, integers, text strings and lists.

`while get_key() != “esc”:`
` temp_inside=temp_sensor_inside.measurement()`
` text_at(3,”temperature inside car=`
`"+str(temp_inside)+” °C”,"left")`
` sleep(0.5)`
	Defines a while loop that will continue until the escape key is pressed.
	While loops repeat the statements in the block if the condition at the top of the loop is true. In the example, looping continues until the escape key is pressed. Not pressing a key or pressing any key but escape means that get_key() will return a value that is not equal to “esc”. The loop condition is true and looping continues. If the escape key is pressed, get_key() returns “esc”. The condition will evaluate as “esc” not equal to “esc”, which is false. A false result means that the loop statements are not repeated. Program execution skips to the statement just after the loop. Note: The block starts with a **colon** and includes the indented lines that follow. while get_key() != “esc”: is available from the TI Hub > Commands menu.

`2+3==6 (result is false)`
`x+4>=y (if x=1 and y=3, the result is true)`
`“enter”!=”esc” (result is true)`
	Boolean expressions evaluate to either true or false. The examples show some of the relational operators available from the Built-ins > Ops menu.
	Note: == is the Python operator to check equality. >= is the Python operator to check whether the value to the left is greater than or equal to the value on the right. != is the Python operator to check inequality.

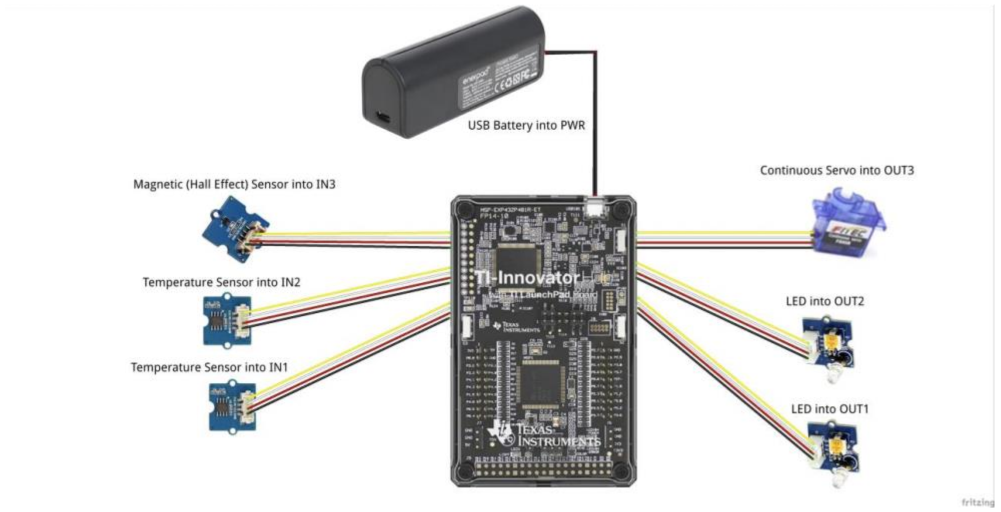
`  if magnet_value <200:`
`    text_at(5,"magnet is present    ","left")`
`  else:`
`   text_at(5,"magnet is not present","left")`
	Checks to determine if the value of variable **magnet_value** is less than 200. If the statement is “true” then the statements in the **if** block are executed. If the statement is “false” then the statements in the **else** block are executed. In the example, when **magnet_value** is less than 200, the text “magnet is present” will be displayed on the calculator screen. When the value of **magnet_value** is 200 or greater the text “magnet is not present” will be displayed. Note: **if..else..** is available from the Built-Ins > Control menu.

`if temp_inside>25 and magnet_value<200:`
`  text_at(“pet in danger”)`
	If both expressions are true the **and** function is “true”, then the block is executed. Otherwise, the **and** function returns false, and the block is skipped. In the example, when the temperature value is greater than 25 and the magnet value is less than 200, the message “pet is in danger” will be printed on the calculator screen. Note: **if..** is available from the Built-Ins > Control menu.

Calculator Notes:

- On the Home screen press 4:Current (or the Home key) to return to your document file.
- On the Home screen Press 1:New to create a new document file.
- You create and edit programs in a Python app program editor page. You run programs from a Python app Shell page.
- Use the [menu] key to see the options for your current app.
- ctrl-B is the shortcut from the Run menu to check the syntax and save changes to your program.
- In the Python editor ctrl-R is the shortcut from the Run menu to check the syntax and save changes and to run the program on the following Python Shell page.
- Use ctrl-R in the Shell to re-run a program.
- Press [enter] to run the statement entered on the line of a Python Shell page.
- Find your function names in the Python app by pressing the [var] (variables) key.
- To see options for an object paste the object name from the var key menu into the Python editor then press the period key.
- Move from page to page by using ctrl-left arrow and ctrl-right arrow or by using the touchpad pointer to click on the desired page tab.
- ctrl-doc (+page) will add a blank page to your document.
- ctrl-Z will undo your last action.
- To stop ("break") a program press and hold the ON key until you receive a dialogue box.
- ctrl-S is the shortcut for saving your entire document file. Do this periodically to save your work.

Sensor and actuator Hub connections


USB Battery into PWR
Magnetic (Hall Effect) Sensor into IN3
Temperature Sensor into IN2
Temperature Sensor into IN1
Continuous Servo into OUT3
LED into OUT2
LED into OUT1
TI-Innovator
fritzing