Overview:

In the Pet Car Alarm project, students build and program a simple feedback and control loop. Feedback and control loops are central to many industrial systems and consumer products. The system uses three input modules; Two temperature sensors and a Hall effect magnetic field sensor. The system has four outputs that include two white LED lights (headlights), one continuous servo motor (controls the window) and also makes use of the TI-Innovator Hub's built-in speaker (horn honking). The continuous servo requires more power than the calculator can provide. Therefore, an external USB battery is needed. A program is written to read all three input parameters and logically compare them with critical set-point values to determine when to turn on the alarm. The project is presented in a series of small challenges that build the knowledge and skills required for the final open-ended challenge. The final challenge also relies on the students' understanding of important biology and Earth science topics that are relevant to optimizing the system the students ultimately design and refine.

Note: This project assumes the students have a working knowledge of the programming concepts in the Digital Mood Ring project. Please refer to that project as a review of concepts. The materials for the Digital Mood Ring project can be found here - link.

Possible NGSS topics to explore with students:

Disciplinary Core Ideas:

- HS-LS1-3 – Mechanisms of homeostasis
- MS-ETS1-1 to 1.4 – Engineering design
- MS-PS3-3 – Design that maximizes thermal energy transfer

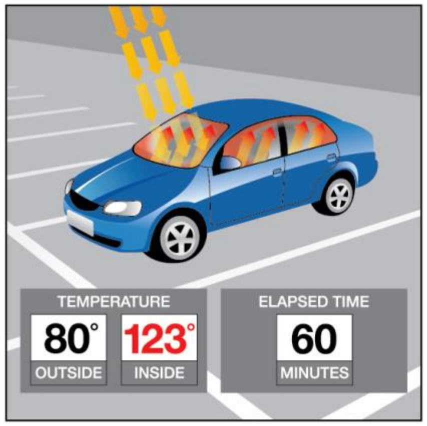
Science & Engineering Practices:

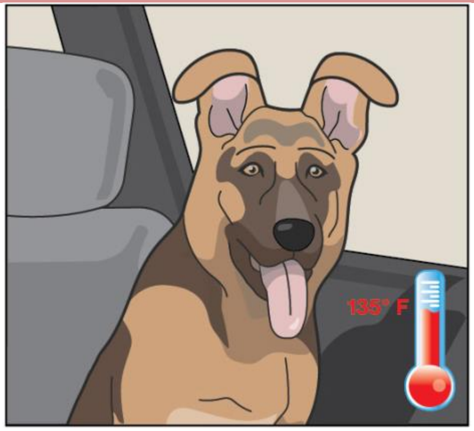
- Asking questions & defining problems
- Evaluate competing designs
- Constructing explanations & designing solutions

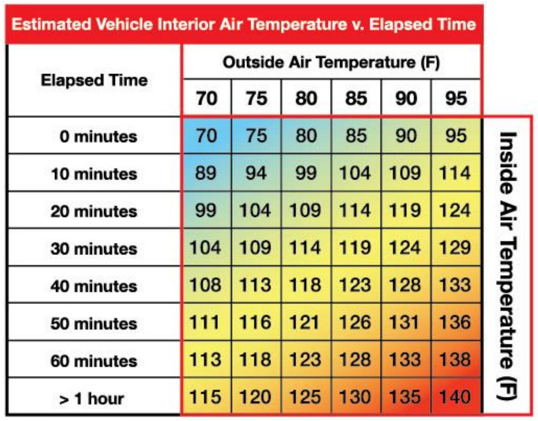
Crosscutting Concepts:

- Systems & System Models
- Cause & Effect
- Energy & Matter

Background:







Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed Time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Light to Heat
Each summer, there are horrifying stories of children and pets left in hot cars. Ultimately, most end up having a heat stroke and in many cases pass away. The inside of a car heats up so much faster than the outside due to the greenhouse effect. Rays of sunlight stream into the vehicle through the windows and strike, the light strikes the surfaces of the interior of the car. That visible light is absorbed and reradiated as infrared light. Infrared radiation has a larger wavelength than visible light. The infrared radiation is unable to escape back through the windows. The trapped radiation causes the temperature inside the car to rise faster than the outside temperature.

Stayin' Cool!
Mammals such as humans, dogs, and cats all have ways to regulate temperature, thermoregulation, to maintain homeostasis. Humans sweat to increase the removal of heat from the body through evaporation. Dogs usually pant to remove heat although they also have a small number of sweat glands in the pads of their paws. Cats will sprawl out on surfaces that are relatively cool to help remove body heat. They will lick their paws and rub the saliva on warmer parts of their bodies to increase evaporative cooling which is a similar mechanism as sweating in humans. When thermal regulation mechanisms are unable to maintain homeostasis, that mammal will go into heat distress.

The problem with too much heat
Heat distress leads to brain impairment, dehydration, heart failure, cell swelling, and protein denaturation… and potentially death.
Conduct research into why mammals such as humans, dogs, and cats are unable to endure hot environments like closed cars in the summer for long periods. Then design a solution using technology to protect car occupants from heat distress. Refine your design until you have a working prototype. With your teacher's permission, compare your prototype with those of your classmates to determine which team has the "best" system.

Python Reference for Pet Car Alarm

For more on programming the TI-Innovator Hub with TI-Nspire CXII Python follow the links to the TI Hub Menu Map: TI-Nspire™ Python Programming > Python Menu Map > TI Hub Menu

Function	Example	Behavior
from module_name import *	`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.

# text comment	`# 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(frequency,time)	`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.

name_of output=device_type("port")	`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.

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

for index in range(stop value):
block
	`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.

name_of_led_object.blink(frequency, time)	`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.
name_of_device=output_device_type("port")	`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.

continous_servo_variable.set_cw (speed, time)
continous_servo_variable.set_ccw (speed, time)
	`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.

name_of sensor=sensor_type("port")	`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.

var=name_of_sensor.measurement()	`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(row,"text","align")	`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.
var="text string"	`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":
block
	`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.

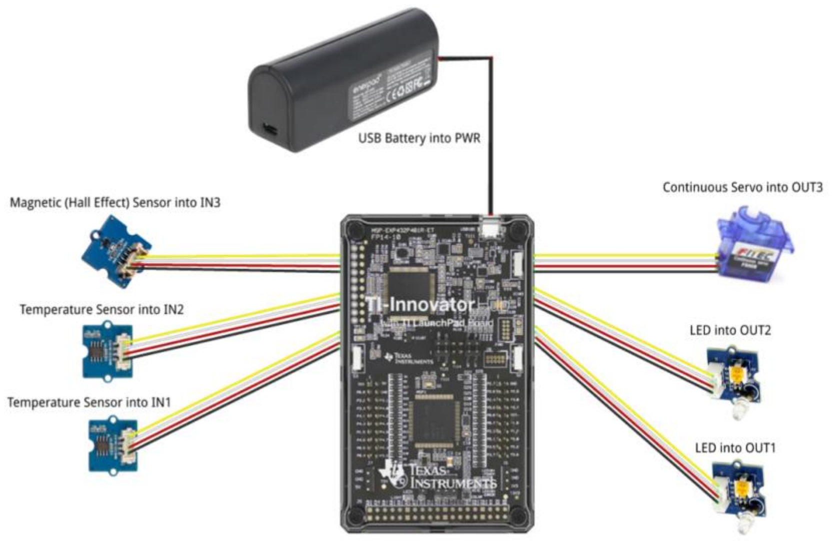
<Boolean expression>
value 1 operator value 2
	`2+3==6 (result is false)`
	`x+4>=y (if x=1 and y=3, the result is true)`
	`"enter"!="esc" (result is true)`
		Boolean expressions evaluate to either true or false. The examples show some of the relational operators available from the Built-ins > Ops menu.
		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 <Boolean expression>:
block
else:
block
	`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 <Boolean expression> and <Boolean expression>:
block
	`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.

Setup Project:

Students may work in groups of two, three or four.


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

Supplies:

- TI-Innovator Hub and cable
- TI Nspire CXII calculator
- Grove Temperature sensor x2 (inside and outside measurement)
- Grove Hall effect magnetic proximity sensor
- Grove White LED Light x2
- Grove Continuous Servo motor
- Grove Cable x5
- External USB Battery w/ Micro USB Cable
- Fashion Doll Car, shoebox, or another object to model a car
- Small toy pet
- Small magnet (ceramic, ferrite or ceramic-ferrite) for pet collar
- Small piece of clear plastic to model car window
- Cellophane tape to attach magnet to pet and window to motor
- Cling wrap (e.g. Saran) to enclose model car
- Safety scissors

Student Activity:

Teacher Notes:

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)

Teacher Guidance during Challenge 1:

- `from` ti_hub `import` * brings in all of the functions on the Hub menus to be available for use in the program. This statement also establishes connection with the TI-Innovator Hub.
- Use a For loop to alternate between two different sound frequencies one octave apart.
- Loops are used to repeat a set of command. A For loop repeats a specified number of times. The programmer defines a For loop with four inputs: a counter variable, a beginning value for the counter variable, a stop value for the counter variable and an optional step value variable.
- The `for` index `in` range( ) form of the For loop assumes that the index variable starts at 0 and increases by a step of 1 with each loop. If n is less than the stop value, the loop continues to repeat. The block of statements to repeat begins with a colon and includes the indented lines that follow.
- `for` index `in` range( ) is found on the Built-ins>Control menu.
- Following sound.tone with the sleep() function assures that the tone will complete before the program begins to play the next sound.

Example program: **pc1.py**

```
from ti_hub import *
for n in range(5):
  sound.tone(440,1)
  sleep(1)
  sound.tone(880,1)
  sleep(1
```

Challenge 2: 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.

Teacher Guidance during Challenge 2:

- left_headlmap=led("OUT 1") and right_headlmap=led("OUT 2") define external output devices of the LED type with variable names of left_headlamp and right_headlamp. The statements associate the variables with physical ports on the Hub.
- Each device type has a set of related functions based on the capabilities of the device. For example, LED functions are on, off and blink. **Note:** You can see a dropdown menu of functions for a variable by typing a period after the variable name in the editor.
- Use a For loop to make the LED's turn off and on.
- Use sleep() to pause the program and create a blink effect. The number of seconds for the sleep() function determines how quickly the LED's will blink.
- Example program:

Example program: **pc2.py**

```
from ti_hub import *
left_headlamp=led("OUT 1")
right_headlamp=led("OUT 2")
for n in range(30):
  left_headlamp.on()
  right_headlamp.on()
  sleep(1)
  left_headlamp.off()
  right_headlamp.off()
  sleep(1)
```

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).

Teacher Guidance during Challenge :

- window_motor=continuous_servo("OUT 3") defines an external output device of the continuous servo motor type with the variable name window_motor. The statement associates the window_motor variable with the physical OUT 3 port of the Hub.
- Servo motors must be connected to the OUT 3 port because 5 volts are necessary to power the servo motor.
- Use .set_cw() (Clockwise) and .set.ccw() (Counter Clockwise) functions to turn the motor in the desired direction.
- .set_cw() and .set_ccw() require two inputs, speed (value between 0 for off and 255 for maximum speed) and time in seconds for the motor to be on.
- Typically you will follow the motor set functions with sleep() functions to assure that the motor completes the desired action before proceeding to the next function.
- Be sure external battery is attached to the PWR port of the TI-Innovator Hub and turned on just before running program.

Example program **pc3.py**

```
from ti_hub import *
window_motor=continuous_servo("OUT 3")
window_motor.set_cw(20,1)
sleep(2)
window_motor.set_ccw(20,1)
sleep(2)
```

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.

Teacher Guidance during challenge 4:

- temp_sensor_inside=temperature("IN 1") defines a sensor input of the temperature type with the variable name temp_sensor_inside. The statement associates temp_sensor_inside with the physical IN 1 port of the Hub.
- Use the .measurement() function of a temperature sensor to measure and return the current temperature reading.
- temp_inside=temperature_sensor_inside.measurement() stores the measurement value to a variable named temp_inside.
- While loops are useful when you would like a set of statements to be repeated as long as a a certain condition is true.
  - 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.
- text_at( ) has options to build a message by joining together pieces of text with the + operator.

Example program: **pc4.py**

```
from ti_hub import *
temp_sensor_inside=temperature("IN 1")
text_at(9,"Press escape to quit ","left")
while get_key() != "esc":
  temp_inside=temp_sensor_inside.measurement()
  text_at(5,"temp inside care= "+str(temperature)+" °C","left")
  sleep(0.5)
```

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.

Teacher Guidance during challenge 5:

- The Hall effect sensor detects the presence (or absence) of a magnetic field with south polarity. For the Hall sensor you will use Magnetic sensor object type from the Hub Add Outputs menu.
- Sensor values are related to voltage measurements. When the south pole of the magnet is close, readings are usually under 200.
- It is recommended that IN3 is used to supply enough power to the sensor.
- Use the if..else control structure with boolean comparative statements to determine if a magnet is close or not.

Example program: **pc5.py**

```
from ti_hub import *
magnet_sensor=magnetic("IN 3")
text_at(9,"Press escape to quit”,"left")
while get_key()!= "esc":
  magnet_value=magnet_sensor.measurement()
  text_at(3,"magnetic sensor (bits)= "+str(magnet_value),"left")
  if magnet_value<200:
    text_at(5,"magnet is present    ","left")
  else:
    text_at(5,"magnet is not present","left")
  sleep(.5)
```

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.

Teacher Guidance during the Final Challenge:

- This challenge has students create a system that reads data from inputs such as temperature and the Hall effect magnetics sensor which in turn controls LED's, sounds through the speaker, and the continuous servo motor.

Example program: **petcaralarm.py**

```
from ti_hub import *
# Define sensor inputs
temp_sensor_inside=temperature("IN 1")
temp_sensor_outside=temperature("IN 2")
magnet_sensor=magnetic("IN 3”)
# Define alarm response outputs
left_headlamp=led("OUT 1")
```

```
right_headlamp=led("OUT 2")
window_motor=continuous_servo("OUT 3")
# Display system information
text_at(1,"Pet Car Alarm is on ","left")
text_at(10,"Press escape to turn off","left")
# Set initial value for window as closed
window=”closed"
# Set up main while loop for the alarm system
while get_key() != "esc":
  # Read sensors and store values to variables
  temp_inside=temp_sensor_inside.measurement()
  temp_outside=temp_sensor_outside.measurement()
  magnet_value=magnet_sensor.measurement()
  # Report status of alarm system
  # Window closed or open, pet in car,
  # temperature values
  if magnet_value<200:
    text_at(5,"Pet in car    ","left")
  else:
    text_at(5,"Pet not in car","left")
  if window=="closed”:
    text_at(6,"Window is closed","left")
  else:
    text_at(6,"Window is open  ","left")
  text_at(7,"Temperature inside= "+str(temp_inside),"left")
  text_at(8,"Temperature outside= "+str(temp_outside),"left")
  # Detect an alarm situation
  if temp_inside>25 and magnet_value<200:
    # Alarm detected response
    text_at(3,"Warning! Pet is in heat distress","left")
    left_headlamp.blink(5,3)
    right_headlamp.blink(5,3)
    sound.tone(440,.5)
    sleep(.5)
    sound.tone(880,.5)
    sleep(.5)
    # Open window, if closed
    if window==”closed":
      window_motor.set_cw(30,.25)
      sleep(.25)
      window="open"
# If no problem detected,
# Set alarm outputs off
  else:
    text_at(3,"No problems detected             ","left")
    left_headlamp.off()
    right_headlamp.off()
    #  If open, close window
    if window=="open":
      window_motor.set_ccw(30,.25)
      sleep(.25)
      window="closed"
  sleep(.5)
# When [esc] pressed, close down system
text_at(1,"Pet Car Alarm is off","left")
left_headlamp.off()
right_headlamp.off()
```