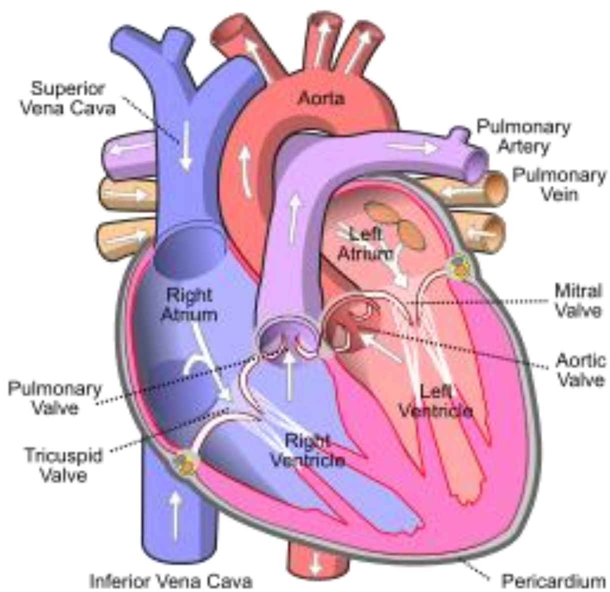
Challenge: Model the Human Four-Chamber Heart

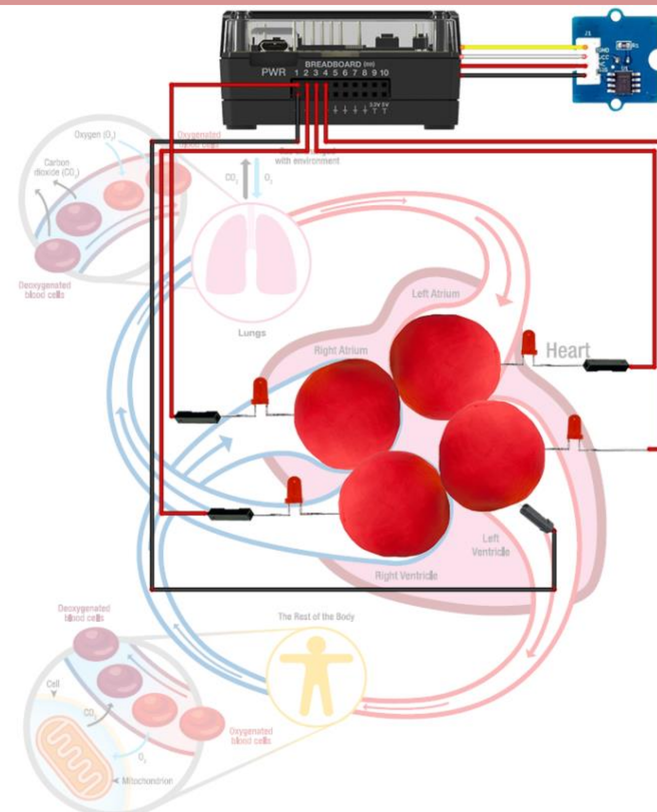
PROJECTS WITH THE TI-INNOVATOR™ SYSTEM (TI-NSPIRE CXII PYTHON) **The Plumber-Blood Circulation**

Overview:

Students explore the circulatory subsystem within the heart by building a model of the human four-chambered heart. In addition to the biology concepts, students have an opportunity to build electrical circuits and use coding to write a simple program that blinks four LEDs in the order of the filling of each chamber. This project explores:

- the anatomy of the heart chambers.
- blood flow through the circulatory system.
- the calculation or pulse rate in beats per minute.
- writing a program to control the model.
- LED electrical circuits.
- material conductivity.





Possible NGSS topics to explore with the student

Disciplinary Core Ideas:

- MS-LS1-3 – Body is a system of interacting subsystems.
- MS-LS1-3 – Chemical reactions release energy as matter moves through the organism

Science & Engineering Practices:

- Developing and using models
- Constructing explanations & designing solutions
- Asking questions & defining problems

Crosscutting Concepts:

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

Background:

The earliest and simplest forms of single-celled life developed metabolisms to conduct the molecular business of growth, repair, reproduction and energy harvesting. As single-celled organisms evolved into multicellular organisms, their body's required a system to move metabolites among the many cells that were not in direct contact with their environment. A circulatory system, that is, a kind of plumbing, evolved to move metabolites into and out of the body and among all of the cells of the body.

The circulatory systems of multi-cellular organisms become increasingly sophisticated moving up the phylogenetic tree of evolution. Sponges (Porifera) rely on simple diffusion while jellyfish (Cnidaria) rely on body pumping for circulation. At some level of size and complexity, diffusion and body pumping are not sufficient to circulate the required metabolites through the organism. A dedicated circulatory system is required to move the body's metabolites efficiently. A circulatory system needs a pump to push the fluid suspension of metabolites throughout the body. This pump is called a heart. The simplest heart is found in fish (ichthys); this heart has two chambers, an atrium, and a ventricle. One deficiency of the two-chambered heart is the mixing of oxygenated blood with unoxygenated blood. A more efficient and developed, the three-chambered heart is found in reptiles (turtles) with two atria and a single ventricle. The additional atria help to prevent the mixing of oxygenated blood with unoxygenated blood. The human (mammalian) four-chambered heart, two atria, and two ventricles are very efficient due to a complete separation of oxygenated blood from oxygenated blood within the organ.

The heart is a strong muscular organ which pumps blood through the body's circulatory system. The human heart is located between the two lungs within the chest. The heart pumps blood with a rhythm, called the pulse, which is informed by the autonomic nervous system. When the flight or fight response occurs in our brain, such as when we are suddenly scared, our heart rate will rapidly increase and we can feel our heart pound within our chest. Normal resting is between 60 and 100 beats per minute or BPM.

When a person exercises, such as running, the autonomic nervous system raises the heart rate without the person's conscious thought. The maximum heart for a young person is about 200; this maximum diminishes as a person ages. One measure of an athlete's fitness is their cardiac output, which is the amount of blood they can circulate from the lungs to their muscles. Cardiac output is the product of the heart's pulse rate by the heart's stroke volume. An average person's stroke volume is about 70 ml. One response to aerobic training is an enlargement of the stroke volume and the ability to maintain a fast heart rate. World-class aerobic athletes have large stroke volumes and can maintain fast heart rates for long periods, thus transporting large volumes of oxygen to their working muscles.

The function of the heart and lungs is to transport oxygen from the environment to the mitochondria within each cell of the body to oxidize the ingested sugar (glucose) from the digestive system to provide the energy required for life. When oxygen is breathed into the lungs, the oxygen diffuses into the blood's red blood cells flowing through the high surface area alveolar bed within the lung. The oxygenated blood moves through the circulatory system to the high surface area capillary bed where the oxygen diffuses into the mitochondria within the cells to participate in metabolism. The flow of blood through the four chambers of the heart occurs in the following steps:

1. Deoxygenated blood is returned from the body to the right atria through the vena calva.
2. From the right atrium, blood flows to the right ventricle via the tricuspid valve.
3. From the right ventricle, blood is pushed into the lungs for oxygenation via the pulmonary artery.
4. From the lungs, the oxygenated blood is returned to the heart's left atrium via the pulmonary vein.
5. From the left atrium, the blood flows to the left ventricle via the mitral valve.
6. From the left ventricle, the blood is pushed out to the capillary bed via the aorta.

Python Quick Reference for The Plumber: Model the Human Four-Chamber Heart

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	`# Define an external LED output device` `#  with variable name led1`	# 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.
name_of output=device_type("port")	`led1=("bb1")`	Creates an LED object named **led1** connected to breadboard port 1, "bb1". led is available from the TI Hub > Add Output Device menu. The drop down menu for port does not include breadboard ports. It is necessary to escape from the menu and type in bb1, bb2, etc. Note: = is the Python operator for storing or assigning values to a variable.

name_of_led_object.on()
name_of_led_object.off()
	`led1.on()`
	`sleep(1)`
	`led1.off()`
	`sleep(1)`
		Turns on an LED object named **led1** then pauses the program for 1 second before turning **led1** 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. Note: for index in range is found on the Built-ins>Control menu.

for index in list:
block
	`for c in [led1,led2,led3,led4]:`
	`c.on()`
	`sleep(1)`
	`c.off`
	`sleep(1)`
		Loops through the elements of the list one at a time. The index variable **c** is replaced with the list element. In this example, the list elements are LED output objects that were defined earlier in the program. First led1 is turned on then off, then led2 the same and so one through all of the list elements. Note: for index in list is found on the Built-ins>Control menu. List elements are enclosed in square brackets. Values, text strings, objects are some of the data types that can be used in a list.

sound.tone(frequency,time)	`sound.tone(440,1)`	Plays a tone through the TI-Innovator Hub speaker. In the example, plays the frequency 440 hertz (Hz) for 1 second. Note: sound.tone() is available from the Hub Built-in Devices>Sound Outputs menu.
name_of sensor=sensor_type("port")	`temp_sensor=temperature("IN 1")`	Creates a temperature sensor object named **temp_sensor** 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()	`t=temp_sensor.measurement()`	Reads and stores the current measurement value of the **temp_sensor** object into variable **t**. 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= " +str(t)+" °C","left")`	This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable **t** has a value of 26, the following is displayed on row 3, aligned to the left: temperature= 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.

while get_key() != "esc":
block
	`while get_key() != "esc":`
	`t=temp_sensor.measurement()`
	`text_at(3,"temperature= "+str(t),"left")`
		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 t<=27:`
	`period=.15`
	`else:`
	`period=.05`
		Checks to determine if the value of variable **t** is less than or equal to 27. 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 **t** is less than or equal to 27, the value .15 is stored to the variable **period**. When the value of **t** is or greater than 27, the value .05 is stored to the variable **period**. Note: **if..else..** is available from the Built-Ins > Control menu.

Challenge: Model the Human Four-Chamber Heart

Supplies and Equipment:

- TI Nspire CXII family calculator
- TI-Innovator Hub
- 4 x M/F Jumper Wires
- 1 x M/M Jumper Wires
- 4 x LED
- 4 x Toothpicks with labels
- 3oz. A conductive compound such as Play-Doh. Modeling clay or plasticine will not work
- Print of the Model Heart Build Sheet
- Grove Temperature sensor

Setup Project Model:

The four separate chambers of the heart are formed using Play-Doh. The heart's blood flow is modeled using the Innovator Hub and four LEDs controlled by a TI-BASIC program.

The model should include these events:

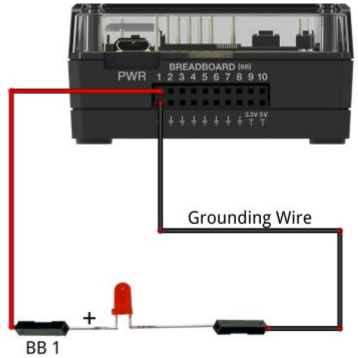
1. Unoxygenated blood flows into the right atrium via the superior vena cava.
2. From the right atrium, blood flows to the right ventricle via the tricuspid valve.
3. From the right ventricle, blood is pushed into the lungs for oxygenation via the pulmonary artery.
4. From the lungs, the oxygenated blood is returned to the heart's left atrium via the pulmonary vein.
5. From the left atrium, the blood flows to the left ventricle via the mitral valve.
6. From the left ventricle, the blood is pushed out to the capillary bed via the aorta.
7. An indication of a heartbeat and the calculation of heart rate

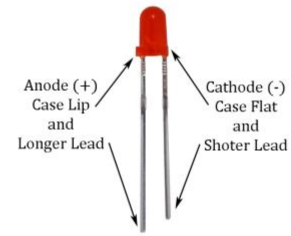
The Student Activity	The Teacher Activity
Sit in small groups of 2-3 students with calculators and supplies for this activity.	Introduce the overall engineering goal of the project and how the model should represent some of the electrical aspects of the human heart. The project will be built stepwise with three student challenges. Discuss possible prototypes. Have students draw a picture of their prototype and describe how it will work. Share ideas with the class.

Challenge 1: Connect an LED to BB1 and blink at a rate of 2 Hz (flashes per second) for 10 seconds.

Guidance during challenge 1:

1. Review concepts from 10 Minutes of Code for the Innovator Hub including the for loop.
2. Review the Mood Ring project as an example of a feedback and control system.
3. Ensure students can identify the polarity of an LED.
4. Demonstrate to students how to insert an LED into the receptacle of an M/F jumper wire. The anode (+) wire pin should be inserted into the BB 1, and the cathode (-) wire pin should be inserted into the ground hole.
5. Show students how to open a new file and insert a program named “blink”
6. Demonstrate the code necessary to blink at 2 Hz (per second) for 10 seconds.
7. Challenge students to blink the LED blink at a rate of 5 Hz for 20 seconds.





Example program: **blink.py**

```
from ti_hub import *
led1=led("bb1")
for n in range(20):
  led1.on()
  sleep(.25)
  led1.off()
  sleep(.25)
```

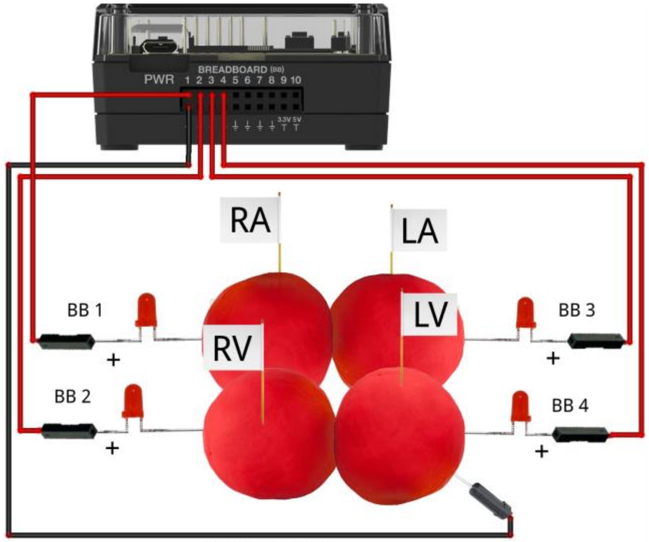
Challenge 2:

- Use four pieces of Play-Doh to model the four heart chambers.
- Place build sheet on the desk and place Play-Doh pieces on sheet.
- Label each piece of Play-Doh with the appropriate chamber name.
- Add three additional LEDs to the Hub connected to BB 2 thru BB 4.
- Insert the negative lead of each LED into each piece of Play-Doh.
- Add a single grounding wire from any of the pieces back into any ground on the BB connector of the Hub.
- Write a program to blink the four LEDs in sequence 1,2,3,4. Be sure to model the blinking order to match the blood flow order. The right atrium fills first while the left ventricle fills last.
- Add a beep to your program to sound like an EKG machine after each cycle of the heart. For example, the program should run for at least 30 seconds.

Guidance during challenge 2:

1. Be sure students insert the LEDs correctly with the polarity identified in the previous diagrams.
2. Discuss the requirement to define an LED object for each heart chamber that is associated with a unique breadboard pin number.
3. Discuss the concept of nested loops (loop within a loop).
4. Be sure the two loops have different index variables.
5. Notice that the nested inner loop cycles through a list of the heart chambers object names defined above.
6. The sound.tone function should be short. Have students explore frequencies and durations to produce the desired effect.
7. The “Heart Rate” of the model can be calculated from the code and expressed in Beats Per Minute (BPM):
    - The inner “for” loop cycles four times for every heartbeat.
    - There are two sleep()’s of .075 seconds per cycle.
    - The period for one heartbeat of the model is:
      2 x 4 X .075 = 0.6 seconds/beat.
    - Heart rate is the reciprocal of heartbeat period or 1/.6 = 1.67 beat per second.
    - Convert heart rate from units of beats per second to units of beats per minute.
    - Example calculation:

$$\frac{1\ Beat}{4 \times 2 \times .075\ sec} \times 60\frac{sec}{min} = 100\frac{beats}{min}$$



Example program: **heartbeat.py**

```
from ti_hub import *
ra=led("bb1")
rv=led("bb2")
la=led("bb3")
lv=led("bb4")
for n in range(30):
  for c in [ra,rv,la,lv]:
    c.on()
    sleep(.075)
    c.off()
    sleep(.075)
  sound.tone(440,.1)
```


- What would you change in the program to speed up or slow down the flashing? Calculate the “heart rate” for this program.
- What happens when you separate the chambers? Can you explain your observation?

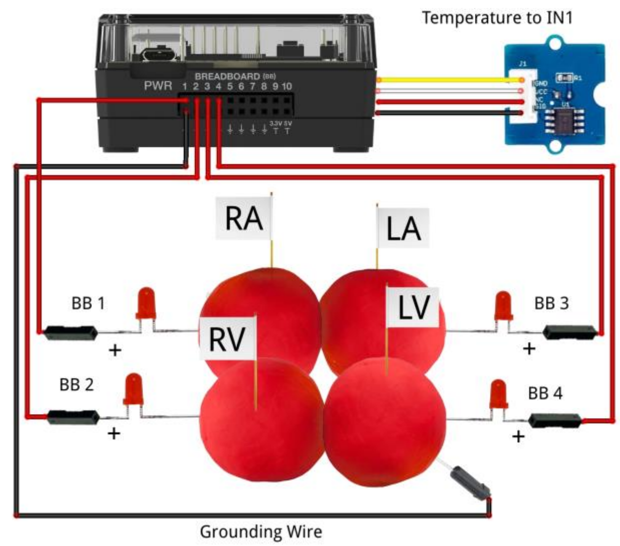
8. Have the students observe closely to see the pattern of the LED’s representing the heart chambers turning on. Try a sleep value of .125 to slow down the rate and make the pattern easier to observe.

Challenge 3: The autonomic nervous system controls the human heart rate. When frightened, we do not need to think to make our heart beat faster. When our body becomes overheated, our heart rate elevates to try to cool us.

Use the Grove temperature sensor to modify the heart rate of the model. Change the program from Challenge 2 such that when body temperature goes up, heart rate goes up.

Guidance during challenge 3:

1. Insert a Grove temperature module into IN 1 on the Hub
2. Define a temperature sensor object associated with port IN 1.
3. Use a while get_key( ) !=”esc” loop to run the model indefinitely until the escape key is pressed to quit.
4. Measure the temperature sensor value once each loop cycle and store the value to a variable.
5. Use a variable called “period” that programmatically changes the duration of sleep()
6. Use an if..else decision structure with the measured temperature to set the period.
7. What are the symptoms of heat stroke? How does heart rate help the body to cool?



Example program: **heartmodel.py**

```
from ti_hub import *
ra=led("bb1")
rv=led("bb2")
la=led("bb3")
lv=led("bb4")
temp_sensor=temperature("IN 1")
text_at(9,"Press escape to quit","left")
while get_key() != "esc":
  t=temp_sensor.measurement()
```

```
if t<=27:
  period=0.15
else:
  period=0.05
for c in [ra,rv,la,lv]:
  c.on()
  sleep(period)
  c.off()
  sleep(period)
text_at(3,"temperature= "+str(t),"left")
bpm=(1/(2*4*period))*60
text_at(4,"bpm= "+str(bpm),"left")
sound.tone(440,.1)
```