Challenge: Design Smart Irrigation System

TI-NSPIRE™ CXII PYTHON

Overview:

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*. In the smart irrigation project, students are challenged to build and program a smart irrigation system based on the engineering concept of a feedback and control loop. The system uses three input modules: a temperature-humidity sensor, a soil moisture sensor, and an ambient light level sensor. The system has a single output to a power control unit with a water pump connected. A program is written to read all four input parameters and logically compare them with critical set-point values and among each other to determine when to turn the water pump on and off. 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 ecology topics that are relevant to optimizing the system the students ultimately design and refine. *Digital Mood Ring link.

Possible NGSS topics to explore with students:

Disciplinary Core Ideas
- MS-LS2-3, LS2-5-Ecosystems Interactions
- MS-ESS2-4 – Water Cycle
- MS-ETS1-2, MS-ETS1-4 Design and evaluate solution to problem

Crosscutting Concepts
- Stability & Change
- Cause & Effect

Science and Engineering Practices
- Constructing explanations & designing solutions
- Evaluate competing designs
- Asking questions & defining problems

Background:

Identify the problem

Humans invented agriculture over 6000 years ago to produce more food than could be hunted and gathered from the environment. This increase in food availability produced a rapid increase in population that was totally dependent on agriculture for survival. Today, the world's large population requires sophisticated large-scale agriculture to keep everyone fed. Climate changes can have severe consequences on the food production required to keep the large human populations fed. Natural and man-made ecological disasters, such as the Dust Bowl of the 1930s, can have severe consequences on the dependent populations. A more recent example told in a 2016 article from The Guardian*, describes the devastating effects on the local population of a severe drought in Zimbabwe. Science and technology can help to optimize food production and mitigate the effects of climate change and poor farming practices.

*Link to The Guardian article.

Build a Solution

In this project, students are challenged to use science and technology to design and build a system that utilizes a limited amount of water in a "smart" way to grow crops. To achieve this, they need STEM skills and knowledge from the disciplines of ecology, biology, chemistry, electronics, logic and computer programming. To enable students to complete this project, they are challenged to complete smaller tasks that will provide the programming and electronics knowledge required. In addition, during the project, the teacher has the opportunity to teach scientific principles in a relevant and meaningful setting. Some of the following science principles could be worked into the activity.

- The effect of air temperature on the evaporation rate of water? Should water be used when it is hot or cool?

- How does soil type affect the percolation of water into the ground? Should water be delivered in a strong rush? Or should it be delivered slowly and in pulses?
- Does relative humidity affect the evaporation rate of water? Does warm air hold more water than cool air? What is relative humidity? Should water be delivered when relative humidity is high or low?
- How does the soil moisture level affect the rate that water can be absorbed? At what soil moisture level should water be delivered to crops?
- Does photosynthetic activity affect the uptake of water? Should plants be watered during daylight or night?

Python Quick Reference for Smart Irrigation Project

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 light_level sensor` 	
	`# object named lightsensor`
		# 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 sensor=sensor_type(“port”)	`lightsensor=light_level("IN 1")`	Creates a light_level sensor object named **lightsensor** connected to port IN 1. light_level is available from the TI Hub > Add Input Device menu. Note: = is the Python operator for storing or assigning values to a variable.
name_of_sensor.range(min,max)	`lightsensor.range(0,100)`	Scales the measured values read from the **light.sensor** object to the range of 0 to 100. **Note:** to see options for an object select the object name from the var key menu then press the “.” key.
var=name_of_sensor.measurement()	`light=lightsensor.measurement()`	Reads and stores the current measurement value of the **lightsensor** object into variable **light**. Note: **.measurement()** returns the current measured value of a sensor object. To see options for an object select the object name from the var key menu then press the “.” key.
text_at(row,”text”,”align”)	`text_at(3,“light level= “ +str(light),"left")`	This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable **light** has a value of 26, the following is displayed on row 3, aligned to the left: light level = 26 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”:`
	`light=lightsensor.measurement()`
	`text_at(3,”light level= "+str(t),"left")`
	`sleep(1)`
		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.

sleep(seconds)	`sleep(0.5)`	Pauses program for .5 seconds.

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.

<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 moisture < 10:`
	`text_at(5,"The soil is dry","left")`
	`else:`
	`text_at(5,"The soil is moist","left")`
		Checks to determine if the value of variable **moisture** is less than 10. 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 the **moisture** value is less than 10, the text “The soil is dry” will be displayed on the calculator screen. When the value of moisture is 10 or greater the text “The soil is moist” will be displayed. Note: **if..else..** is available from the Built-Ins > Control menu.

if <Boolean expression> and <Boolean expression>:
block
	`if temperature<= 25 and humidity>=80:`
	`text_at(“it is cool and humid.”)`
		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 is less than or equal to 25 and the humidity is greater than or equal to 80, the message “It is cool and humid” will be printed on the calculator screen.

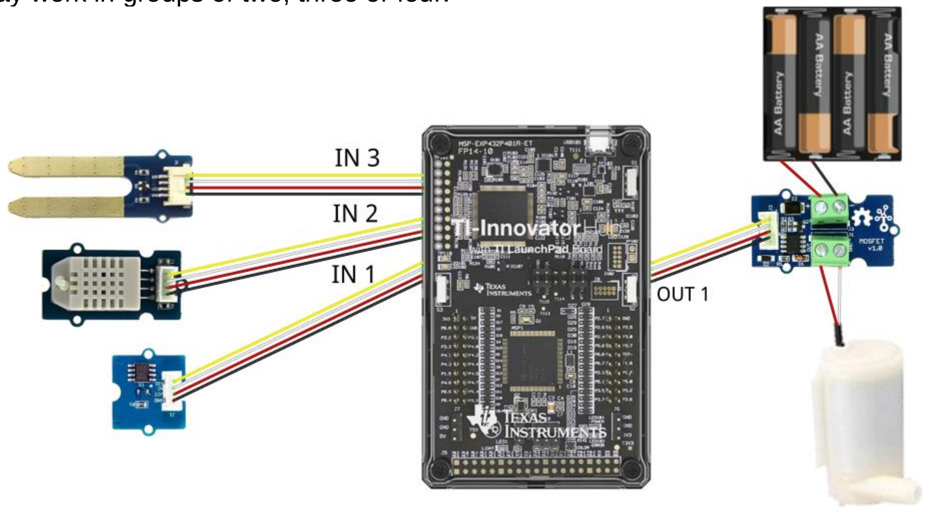
name_of_device=device_type("port")	`pump=analog_out("OUT 1")`	Creates an analog_out object named **pump** connected to port OUT 1. analog_out is available from the TI Hub > Add Output Device menu. Note: = is the Python operator for storing or assigning values to a variable.

name_of_output_device.set(value)	`pump.set(255)`
	`sleep(10)`
	`pump.set(0)`
		Sets the value of an analog_out device named **pump** to 255 (full power). The pump continues at full power until a value of 0 (off) is received after the sleep function causes the program to pause for 10 seconds. **Note:** The output device will continue using a setting value until a new setting value is received or until the power to the system is removed. Make sure to use the .set(0) or .off() output objective methods to turn the device off. To see options for an object select the object name from the var key menu then press the "." key.

var=input("prompt text")	`speed=int(input("pump speed (0-255)= "))`	Prompts the user to enter a value that will be stored as an integer to the variable **speed**. input() prompts the user to enter a text string. Int() from the. Built-Ins>Type menu changes the text string to an integer data type. The integer value is stored to the variable **speed** for later use in functions that require numeric values as inputs.

Setup Project:

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



Supplies:

- TI-Innovator Hub and cable
- TI Nspire CXII calculator
- Grove - Temperature & Humidity Sensor
- Grove – Soil Moisture Sensor
- Grove – Light Sensor
- Grove – MOSFET for power control module with 4xAA battery holder
- 4 x AA batteries to provide power to the pump (required)
- Water Pump with the plastic tube
- Possible supplies for building garden model:
- Drinking straws
- Duct Tape (always useful)
- Container for the plants, such as a 1-gallon milk
- Soil, perlite or some other growth medium

Student Activity:

Teacher Notes:

Challenge 1: Write a program named *c1* that continuously measures and displays the ambient light level. The program should:

1. connect a light level sensor to IN 1.
2. range the light level reading from 0 to 100.
3. use a While loop to continuously read and display light level every ½ second on the calculator screen.
4. enable the ESC key to quit the while loop and end the program.

Teacher Guidance during Challenge 1:

- The light level sensor is a device that has a large range of responses to light intensity. As a result, the output is not linear with light intensity. In addition, the device does not measure with a particular unit. Instead, the device returns a raw value from 0 to 2^{14}. It is useful to use the RANGE command to scale the output from 0 to 2^{14} to 0 to 100.
- The sensor could be attached to any of the input ports 1, 2 or 3. It is essential that the port used in the statement that defines the sensor object matches the port that the physical sensor is plugged into.
- Connect the sensor as indicated in the picture above.
- The ambient light level is an important biological and ecological factor that affects phototaxis, primary productivity, photosynthetic rate, and ambient heating.
- Discuss with students how the ambient light level can affect when to water a plant.
- Use the Python functions descriptions described above to review the use of a while loop along with gekey() to continuously perform a task.
- Use the Python function descriptions above to review the use of the text_at() function for displaying values on the screen.
- Discuss with students how the use of the sleep() function affects the rate the system monitors light level.

Example program: **c1.py**

```
from ti_hub import *
lightsensor=light_level("IN 1")
lightsensor.range(0,100)
text_at(9,"Press [esc] to quit ","left")
while get_key() != "esc":
  light=lightsensor.measurement()
  text_at(3,"light level= "+str(light),"left")
  sleep(0.5)
```

Challenge 2: Write a program named *c2* that measures soil moisture every two seconds for a total of twenty times. The program should display if the soil is dry or moist based on the sensor reading.

Teacher Guidance during Challenge 2:

- In Challenge 1, all of the necessary steps of the program were given to the students. In Challenge 2, help the students to analyze the task and make a list of the necessary step to accomplish the task.
- Discuss the differences in the control structure between Challenge 1 and Challenge 2. The While loop provides indefinite program execution and the For loop provides definite program execution.
- The soil moisture sensor produces an output from 0 to 2^{14} based on the conductivity (dielectric permittivity)

of the soil which is dependent on water content. The sensor output is proportional with the volumetric water content; however, it is not calibrated in a particular unit or percentage. Like the light level sensor, it is useful to use the RANGE function to scale the sensor output from 0 to 100.

- The sensor could be attached to any of the input ports 1, 2 or 3. It is essential that the port used in the statement that defines the sensor object matches the port that the physical sensor is plugged into.
- Connect the sensor as indicated in the picture above.
- Discuss the different types of soils; clay, sand, and loam and how water interacts with each. Features to consider are the rate of absorption, water capacity, compaction, and runoff.

Clay	Sand	Loam
Soaks in water slowly	Soaks in water very quickly	Soaks moderately
Holds water very well	Holds water poorly	Holds water moderately
Fine grain texture	Coarse grain texture	Mixed grain texture
Severe erosion	Low erosion	Low erosion

- Discuss the best conditions for plant soil watering. Should the soil be allowed to dry out between watering? Do different plants require different condition?
- As an extension, you may determine values to indicate the relative ranking of soil moistures. To do this:
  1. Place soil into three different pots labeled, dry, moist and soaked.
  2. Add water to each pot to the labeled soil moisture.
  3. Consistently insert the soil moisture sensor tines (prongs) halfway into the soil of the pot labeled “dry” and run the sensor program and record the displayed soil moisture value.
  4. Repeat for the other two soil moisture levels.
  5. Use the three measured values in your program to rank the soil moisture of your plants and decide when to water the plant.

Example program: **c2.py**

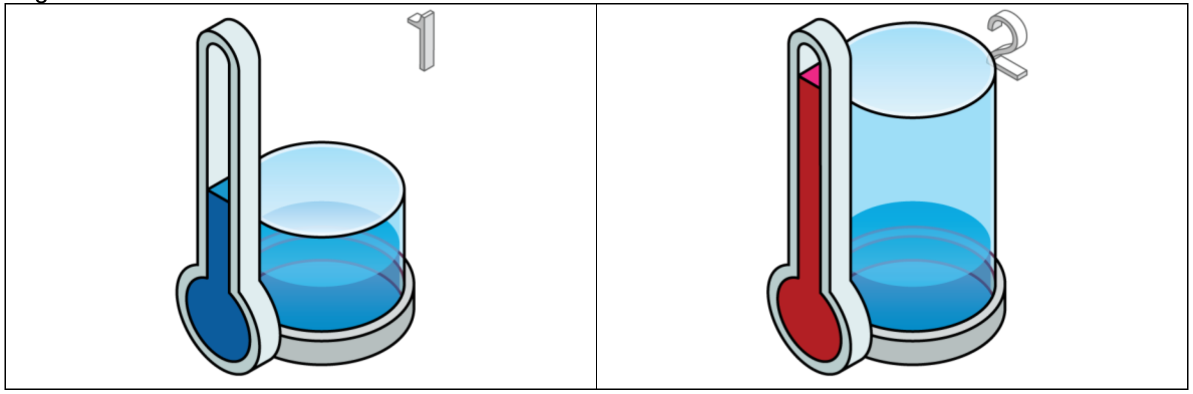
```
from ti_hub import *
moistsensor=moisture("IN 3")
moistsensor.range(0,100)
for n in range(20):
  moisture=moistsensor.measurement()
  text_at(3,"moisture level= "+str(moisture),"left")
  if moisture < 10:
```

```
  text_at(5,"The soil is dry","left")
else:
  text_at(5,"The soil is moist","left”)

sleep(1)
```

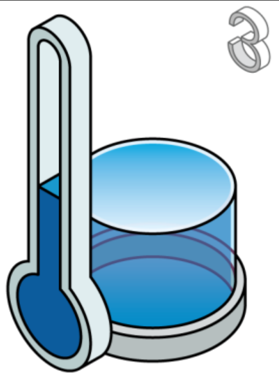
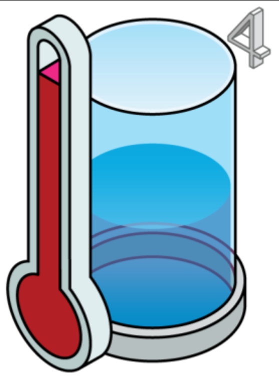
Challenge 3: Write a program named *c3* that connects the Digital Humidity and Temperature (DHT) sensor. Read 20 measurements at two-second intervals and display with an appropriate message. Use a decision tree based on the temperature and relative humidity measurements to determine the present watering conditions as indicated in the graph below. Display an appropriate message with each of the four cases.

Teacher Guidance during Challenge 3:

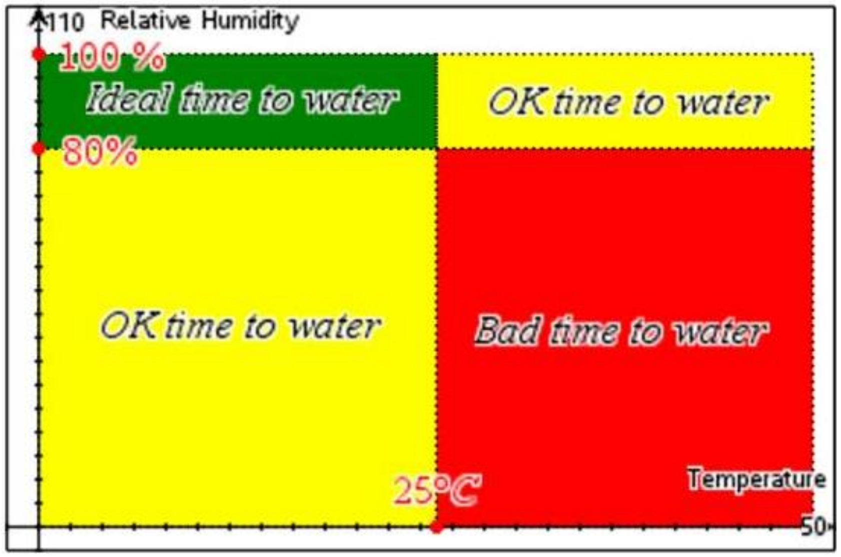
- The amount of water from the ground that evaporates into the air is dependent on the ambient temperature and relative humidity.
- For a particular temperature, there is a maximum amount of water that can evaporate and exist as vapor in the air. This is referred to as the saturation vapor pressure and is calculated using the Clausius-Clapeyron relation.
- Image #1 illustrates when the temperature is low, the corresponding maximum amount of water vapor the air can hold (saturation vapor pressure) is also low as represented by the small container. In image #2, the temperature is high and the maximum amount of water vapor the air can hold is also high as shown by the larger container.



- On any given day, the air will contain a certain amount of water, this is the water vapor pressure as illustrated by the darker blue volume in images 1, 2, 3, and 4. The water vapor pressure may be less than or equal to the maximum saturation vapor pressure.
- Image #3 illustrates a cool day with much water in the air. Notice the container is small and the water in the air just fills its container. Contrast with image #4 that illustrates a warm day with the same amount of water in the air as image #3.

- The relative humidity in image 3 is 100% because the amount of vapor in the air is 100 % of the maximum the air can hold.
- The relative humidity in image #4 is 50% because the amount of vapor in the air is 50% of the maximum the air can hold
- Relative humidity is the percentage of the measured vapor pressure divided by the theoretical saturation vapor pressure. Thus, relative humidity does not inform how much water is in the air, instead, the percentage of the theoretical maximum amount of water that is in the air at a particular temperature.
- When relative humidity is 100%, and water evaporates from the ground, water vapor will precipitate back into the liquid phase and fall as rain and there will be no net transfer of liquid water from the ground into the air.
- There are many inferences that can be made from the above concept.
    - On a hot day, more water will evaporate from the ground to produce a relative humidity of 70% compared to a cold day with 70% relative humidity. This is because on a hot day there is a greater saturation vapor pressure and more water in the air is needed to establish 70%.
    - When relative humidity is high, less evaporated water is required to reach the maximum saturated vapor density at a given temperature.
    - To minimize the net transfer of liquid water on the ground into water vapor in the air (evaporation), water should be delivered when it is cool and the relative humidity is high.



- Help the students to interpret the graph above and determine the conditions for each of the four regions. Use these conditions to construct an if-elseif-else decision tree in their programs.
    - When the temperature is less than or equal to 25 and the relative humidity is greater than or equal to 80, it is cool and humid and an ideal time to water (green).
    - When the temperature is greater than 25 and relative humidity is greater than or equal to 80, it is hot and humid and an OK time to water (yellow).
    - When the temperature is less than or equal to 25 and relative humidity is less than 80 it is cool and dry and an OK time to water (yellow).
    - When the temperature is greater than 25 and relative humidity is less than 80, it is hot and dry and a bad time to water (red).
- Connect the sensor as indicated in the picture above.
- The DHT sensor requires a few seconds to "warm up" and begin communicating with the Hub. During this period, the temperature is reported to be absolute zero (-273 **°C)**. The program uses a While loop to continuously read the DHT every one second until it reports a value greater than absolute zero.

Example program **c3.py**

```
from ti_hub import *
htsensor=dht("IN 2")
```

```
for n in range(20):
  temperature=htsensor.temp_measurement()
  humidity=htsensor.humidity_measurement()
  text_at(4,"sample number= "+str(n+1),"left")
  text_at(5,"temperature= "+str(temperature),"left")
  text_at(6,"humidity= "+str(humidity),"left")
  if temperature<=25 and humidity>=80:
    text_at(8,"Cool and Humid. Best time to water","left")
  elif temperature>25 and humidity>=80:
    text_at(8,"Hot and Humid. OK time to water","left")
  elif temperature<=25 and humidity<80:
    text_at(8,"Cool and Dry. OK time to water","left")
  else:
    text_at(8,"Hot and Dry. Bad time to water","left")
  sleep(1)
```

Challenge 4: Write a program named *c4* using a While loop to continuously measure and display a dashboard of all of sensor value readings. The user should be able to stop the monitoring by pressing the ESC key.

Teacher Guidance during challenge 4:

- This challenge could be skipped if there is insufficient time to complete the project. However, it is a major step toward the final project.
- The program uses most of the code from the previous three challenges.
- All of the sensors need to be connected as they were in the previous challenges.
- Use a while loop with the escape key as an exit.
- This program could be modified for the final challenge by adding the next challenge and control logic.

Example program: **c4.py**

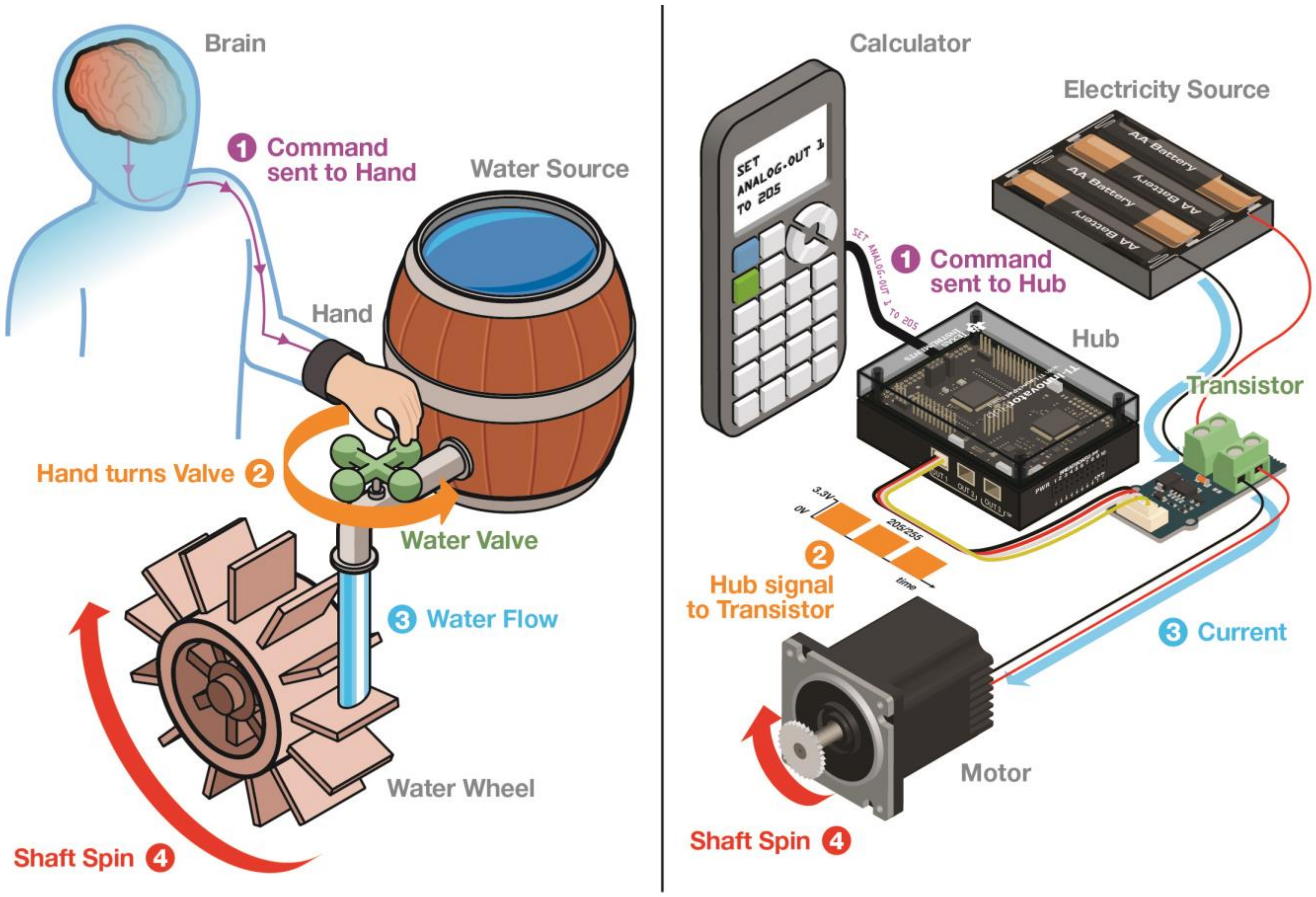
```
from ti_hub import *
lightsensor=light_level("IN 1")
lightsensor.range(0,100)
moistsensor=moisture("IN 3")
moistsensor.range(0,100)
htsensor=dht("IN 2")

text_at(9,"Press [esc] to quit ","left")
```

```
while get_key() != "esc":
  light=lightsensor.measurement()
  moisture=moistsensor.measurement(
  temperature=htsensor.temp_measurement()
  humidity=htsensor.humidity_measurement()
  text_at(3,"light level= "+str(light),"left")
  text_at(4,"moisture= "+str(moisture),"left")
  text_at(5,"temperature= "+str(temperature)+"°C","left")
  text_at(6,"humidity= "+str(humidity)+"%","left")
  sleep(1)
```

Challenge 5: Write a program named *c5* to connect the pump power module and run the pump for 10 seconds. Be sure to turn the pump off at the end of the program. Try setting the pump power to different values. Try to estimate the pump flow rate in mL/sec.

Teacher Guidance during challenge 5:

- The Innovator Hub can control the amount of power delivered to the submersible pump using the power module with an attached battery pack. The power module has a special type of transistor called a MOSFET mounted on the unit. The illustration below makes an analogy between electricity flow in a circuit and water flow through a system to help students to better understand how a transistor works in an electrical circuit.



- When the Python set output value function for an analog_out object is executed (e.g. pump.set(205)), the Hub creates a control signal on the OUT port. The execution of the Python set analog_out. function is similar to the brain sending a command to the hand in step 1 of the analogy.
- When a power module is connected to the Hub's OUT port, that control signal is sent to the transistor. This signal causes the transistor to adjust the amount of electricity that will flow through it. In step 2 of our water analogy, this signal can be thought of as the hand on turning the faucet.
- When the transistor is turned on, more current flows through it and into the motor. Like in step 3 of the analogy, when the faucet is opened, more water flows onto the water wheel.
- When more electrical current flows through the motor, greater power is delivered to the shaft causing it to turn faster. As in step 4 of the analogy, greater water flow onto the paddles of the water wheel cause it to

turn faster.

- In this challenge, the power module is used to control the pump. A pump is just a motor with an impeller and housing attached to its shaft.
- Note: The power module only needs to be SET once. It will keep the present setting until a new command is sent to the module or until power is removed from the Hub. Be sure to SET the power module off (e.g. pump.set(0) or pump.off() for an analog_out object named **pump**.) before exiting the program. If your program fails to turn off the module, the pump will continue to run even after the program has quit.

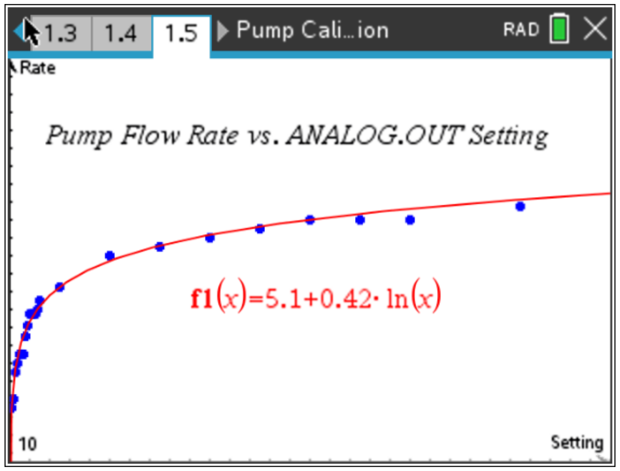
Example program: **c5.py**

```
from ti_hub import *
pump=analog_out("OUT 1")
pump.set(255)
sleep(10)
pump.set(0)
```

- Once students understand how to control the water pump, discuss how excessive watering may lead to water runoff, soil compaction, and nutrient leaching.
- Discuss with students the best flow rate of watering and if watering should be pulsed with a soak-in period.
- As an extension, the students could calculate the flow rate of the pump. To do this, run pump tubing into an empty graduated cylinder and fill the cylinder for a measured amount of time. Use the water volume and run time to calculate the flow rate. Challenge students to find how changing the power value setting of the pump changes the flow rate.

Example Extension Program: **pumpcalibrate.py**

```
from ti_hub import *
pump=analog_out("OUT 1")
for n in range(10):

  speed=int(input("pump speed (0-255)= "))
  pump.set(speed)
  sleep(10)
  pump.set(0)
```



Final Challenge: Write a program that uses all of the sensors in the previous challenges. Continuously monitor light level, soil moisture, temperature, and humidity and display the current value on the calculator display. Use your knowledge of ecology, biology, and earth science to determine the best conditions to water your garden. When the conditions are favorable, set the pump to deliver water at the best rate for your garden.

Test your system with these conditions:

- light level < 20
- soil moisture < 10
- temperature < 25
- humidity > 80

When these conditions are true, set the pump on to 255. When the conditions are

Teacher Guidance during the Final Challenge:

- Discuss with students the concept of optimization and how that is expressed in the logic of the program.
- For each sensor, have students consider the conditions suited to watering. These values are called sensor set-points. For example, a set-point for temperature could be 25°C and watering should only occur when the measured temperature is less than the temperature set-point.

Sensor	Sensor Set-Point	Set-point condition	Conditional expression
Light Level	20	<	light<20
Air Temperature	25	<	Temperature<25
Air Humidity	80	>	humidity>80
Soil Moisture	10	<	moisture<10

- Once the four conditional expressions are determined in the table above, use the logical operations of AND and OR to link those expressions together to be evaluated by a single If-Then-Else conditional statement.
- When AND is used, both conditional expressions must be TRUE for the statement to be TRUE.
- For example, to construct a statement that requires the temperature to be less than the set-point of 25°C and the humidity is greater the set-point of 80% to water is:

```
if temperature<25 and humidity>80:
  pump.set(255)
else:
```

false, set the pump to zero.

```
  pump.set(0)
```

This statement ensures that the pump will turn on at maximum flow rate only when the temperature is cool and the humidity is high.

- When OR is used, either or both conditional expressions must be TRUE for the OR statement to be TRUE.
- For example, to construct a statement that requires the light to be less than the set-point of 20 or the moisture to be less than the set-point of 10 to water.

```
if light<20  or  moisture<10:
  pump.set(255)
else:
  pump.set(0)
```

The pump will turn on when either it is dark or the soil is dry. Note, the pump will also turn on when it is dark and the moisture is low.

- It may be difficult for a student to consider 4 conditional expressions concurrently. It may be best, to begin with only one condition and then add an additional condition with a logical operation. Once successful with to conditional expression, a student can add additional conditions until all four are utilized.
- The pump control statement in the example below does the following: **1)** runs the pump under any environmental conditions if the soil is very dry, soil moisture measurement less than 10, or **2)** runs the pump when the environmental conditions are good for watering if the soil is moderately dry, soil moisture measurement less than 60. The pump will not run under any environmental conditions if the soil is wet. Note: Experiment by taking soil moisture measurements with your soil to determine whether 10 and 60 are good values for your garden environment.

```
if light<20 and temperature<25 and humidity>80 and moisture<60 or moisture<10:
  text_at(7,"good time to water","left")
else:
  text_at(7,"bad time to water","left")
```

- You can save time by building the final program from a copy of the c4.py dashboard program. Go to the page with the c4 program code. Press Menu>Actions>Create Copy. Name your new program **smartwater**. The c4 program sets up sensors and a While loop to take measurements. You will need to define the pump analog_out object in the sensor set up section at the top of the program. You will add if-then-else logic and pump statements within the loop.

Example program: **smartwater.py**

```
from ti_hub import *
#Set up sensors and pump
lightsensor=light_level("IN 1")
lightsensor.range(0,100)
moistsensor=moisture("IN 3")
moistsensor.range(0,100)
htsensor=dht("IN 2")
pump=analog_out("OUT 1")

# Set up While loop
# that will run until esc is pressed
text_at(9,"Press [esc] to quit ","left")
while get_key() != "esc":
# Take measurements and save to variables
  light=lightsensor.measurement()
  moisture=moistsensor.measurement(
  temperature=htsensor.temp_measurement()
  humidity=htsensor.humidity_measurement()
# Display measurement variable values
# ° and % are available from ?! key menu
  text_at(3,"light level= "+str(light),"left")
  text_at(4,"moisture= "+str(moisture),"left")
  text_at(5,"temperature= "+str(temperature)+"°C","left")
  text_at(6,"humidity= "+str(humidity)+"%","left")

# Set up if-then-else logic to run the pump
# when conditions are good and soil is not wet
# or when the soil is dry
   if light<20 and temperature<25 and humidity>80 and moisture<60 or moisture<10:
     text_at(7,"good time to water","left")
     pump.set(255)
   else:
     text_at(7,"bad time to water","left")
     pump.set(0)
# Pause the loop for 1 second
  sleep(1)
# Shut off pump when you escape from the loop
pump.set(0)
```