Smart Irrigation System

In this TI-Innovator™ project, you will design a smart irrigation system that could be used to monitor and meter water from a rain collection cistern that might be used to irrigate a small family garden in Zimbabwe. This model also applies to other related scenarios where "smarter water" usage makes sense. For example, water restrictions are often put in place during the hot summer months in areas of excessively hot and dry climates. A smart water irrigation system could alleviate some of these restrictions as well, if people are smarter about the way they water their yards.

You will have to utilize math skills, computer programming and engineering to design and build a smart watering system to solve a real world problem like the drought in Zimbabwe or, even the problem right in your backyard!

Background:

A drought in South Africa has caused a famine in Zimbabwe and local students are quitting school and soccer to stay home and help grow food for the family. What can be done differently in regards to the watering of crops to allow students to stay in school, instead of working the fields?

Your Challenge:

You are challenged to help solve this problem by designing and building a smart irrigation system to manage a limited amount of water collected in a cistern to irrigate a garden.

Activity Materials:

- TI-84 Plus CE calculator
- TI-Innovator Hub
- 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
- Fast germinating seeds like radish, lettuce or similar.

Project Challenges

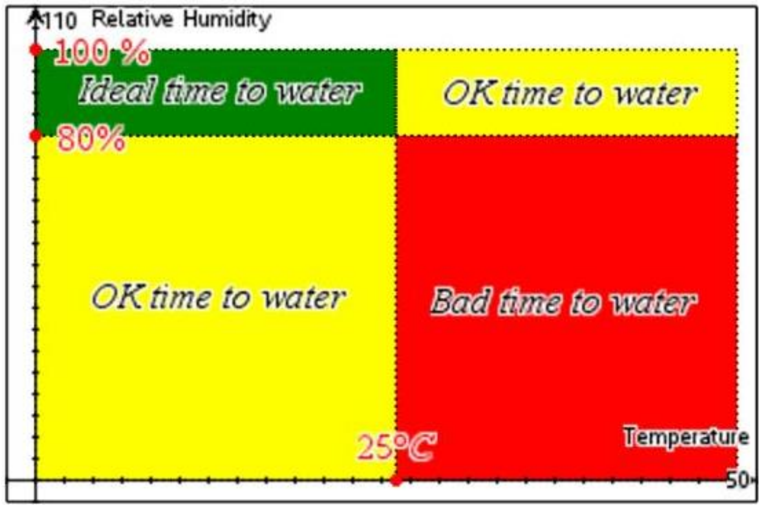
Skill Building Challenge 1: Write a program named C1 that continuously measures and displays the ambient light level. The program should:

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 clear key to quit the while loop and end the program.

Skill Building 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.

Skill Building Challenge 3: Write a program named C3 that connects the Digital Temperature and Humidity (DHT) sensor. Read 20 measurements at two-second intervals and display with an appropriate message.

1. 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.



Skill Building 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 clear key.

Skill Building Challenge 5: Write a program named C5 to connect the pump power module and run the pump for 20 seconds. Be sure to turn the pump off.

1. Try setting the pump power to different values.
2. Try to estimate the pump flow rate in mL/sec.

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 display. Use your knowledge of ecology, biology, and Earth science to determine what the best conditions are to water your garden. When the condition is correct, set the pump to deliver water at a rate that is best 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 false, set the pump to zero.

Example Programming Commands for the project

Example Code	Behavior
`Send("CONNECT LIGHTLEVEL 1 TO IN1")`	Associates the first LIGHTLEVEL object with a light sensor plugged into port IN1 on the Hub.
`Send("SET ANALOG.OUT 1 TO 128")`	Turns on an analog.out1 object, such as a pump, to a power setting of 128
`Send("RANGE LIGHTLEVEL 1 0 100")`	Scales the measured values read from LIGHTLEVEL 1 to return in the range 0 to 100.
`Send("READ MOISTURE 1")`	Reads one measurement from the first moisture sensor.
`Get(M)`	Stores the moisture measurement into the variable named M. *Note a get command must immediately follow a read command. The value stored will contain the measurement from the immediately preceding READ command."

`Output(3,1,"MOISTURE LEVEL= ")`
`Output(3,17,M)`
	When variable M has a value of 26, "MOISTURE LEVEL = 26" is displayed on line 3 of the calculator.

`0→K`
`While K≠45`
`Send("READ MOISTURE 1")`
`Get(M)`
`Output(3,1,"MOISTURE LEVEL= ")`
`Output(3,17,M)`
`getKey→K`
`Wait 1`
`End`
	The commands inside the While structure are looped until the clear key is pressed. The loop continues while logical expression K ≠ 45, is true. The variable K is initially assigned a value of zero so the while loop will execute at least once. The getKey() function monitors the keypad and returns a two digit value of the row and column of the last key pressed. The clear key is located at row 4 and column 5.

`If H≥ 50 and L<10`
`Then`
`Output(3,1,"PUMP IS ON")`
`End`
	The "and" operator compares two separate Boolean expressions concurrently. When both expressions are true, the operation is true.

`If L< 50 or M<10`
`Then`
`Output(3,1,"PUMP IS ON")`
`End`
	The "or" operator compares two separate Boolean expressions concurrently. When either or both expressions are true, the operation is true.

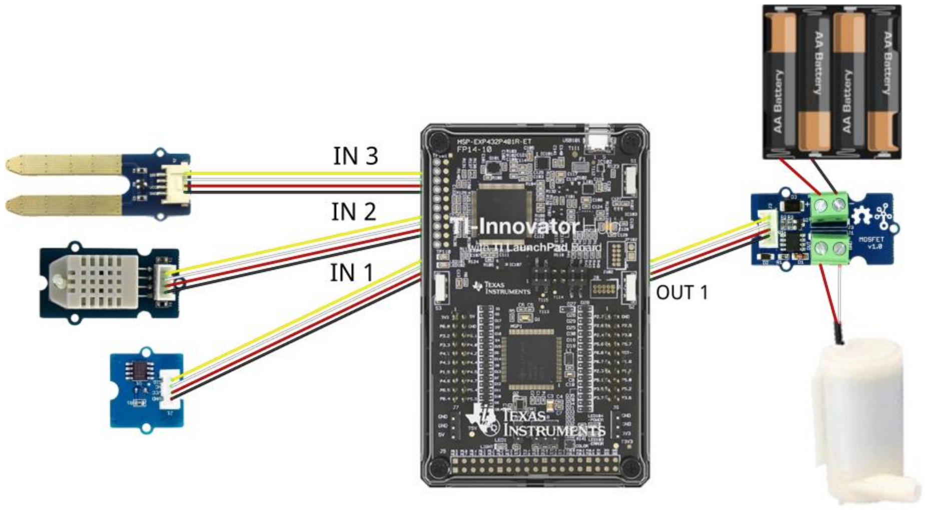
`If L<20 and T<25 and H>80 or M<10`
`Then`
`Output(6,1,"GOOD TIME TO WATER")`
	The example decision tree has a Boolean expression with corresponding statements to execute if true. It also has an Else condition that executes corresponding statements when the Boolean expression is false. This Else condition ensures that a set of statements will always be executed. When this decision tree executes, focus proceeds from top-down. If the Boolean

`Else`
`Output(6,1,"BAD TIME TO WATER ")`
`End`
	expression is true, the corresponding statements are executed and the decision tree is immediately exited. In the example, if T=30 then the first expression is false and the <statements 1> are skipped and the <statements 2> after the Else are executed.

Calculator Notes:

- Press the [prgm] key to see a menu option for creating a new program.
- Press the [prgm] key to see a menu option for selecting programs to edit.
- While in the Program Editor, press [prgm] key to see menu items for programs.
- While in the Program Editor, press [alpha] [f5] to see editing options.
- While in Program Editor, use 2nd Left Arrow and 2nd Right Arrow to jump your cursor to the beginning and end of the line.
- Press [2nd] [quit] to leave the Program Editor and return to the home screen.
- While on the home screen, Press the [prgm] key and select the execution menu option for selecting a program names to paste to the home screen.
- While on the home screen, Press [enter] to run a program named on the home screen.
- The Home Screen "remembers" the last command entered. Press Enter after a program has run to run the program again.
- To stop ("break") a running program press and hold the ON key until you receive a dialogue box.

Sensor and actuator Hub connections


IN 3
IN 2
IN 1
OUT 1
TI-Innovator
AA Battery