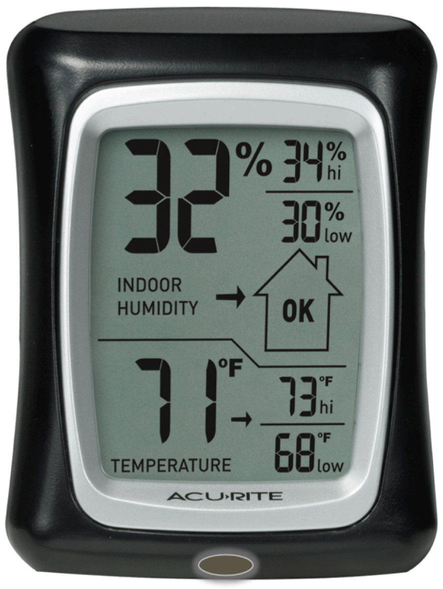
# Lab 5: Embedded Systems Testing
## Part 1
## Embedded Systems Testing



Write and test the embedded code needed to run an electronic device with the following features:

The device will read and display the following:

1. **Current relative humidity** and **temperature** levels
2. The **maximum** and **minimum** readings of both humidity and temperature since it was last reset by the user
3. The current **trends** of both humidity and temperature.
   a. **The humidity trend** will compare the latest two humidity readings and determine if humidity is increasing, decreasing, or stable.
   b. **The temperature trend** will compare the latest two temperature readings and determine if temperature is increasing, decreasing, or stable.
4. **Humidity status Check**: The device will display a humidity status:
   a. *OK* if the relative humidity is between 25% and 55%
   b. *High*: if relative humidity is above 55%
   c. *Low*: if relative humidity is below 25%
5. The device has a **reset** button to reset the maximum and the minimum values of both humidity and temperature.

**Input:**
1- The device will read the **current relative humidity** from an input humidity sensor
Allowed input range: 0% – 100% relative humidity
2- The device will read the **current temperature** from an input temperature sensor
Allowed input range: 0 – 125 degrees Fahrenheit

**Sensitivity** of temperature sensor is one degree Fahrenheit
**Sensitivity** of humidity sensor is 1%

**Output** (see next page for illustration):
Your embedded code will display the following values:

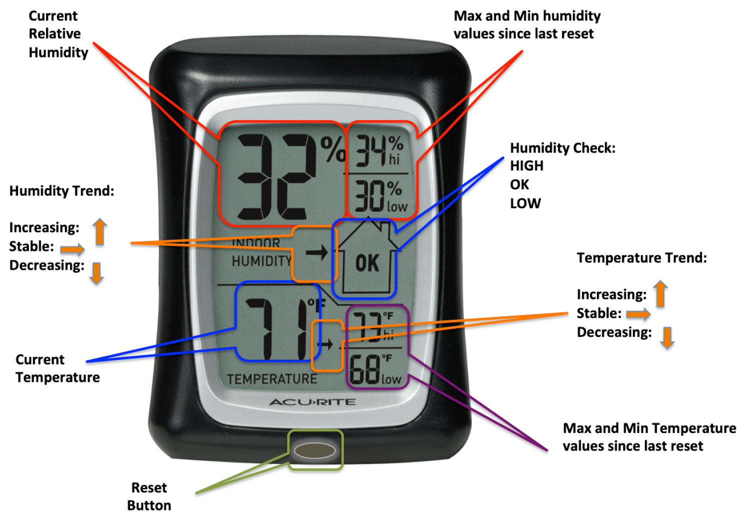
For Relative Humidity:

1. The current relative humidity
2. The maximum relative humidity reading since last reset
3. The minimum relative humidity reading since last reset
4. The relative humidity trend (up, down, or stable)
5. Humidity status (Hi, OK, or Low)

For Temperature:

1. The current temperature
2. The maximum temperature reading since last reset
3. The minimum temperature reading since last reset
4. The temperature trend (up, down, or stable)

*See next page for demonstration* →



Part 1: Deliverables:

1- Submit a working source code to run this embedded system.
   a. For **MANUAL STEPWISE** testing purpose, the **input** can be taken from the keyboard, one pair at a time: Since we are not deploying the embedded system in a device yet, the input will be simulated as prompting the user (tester) to input one pair of keyboard values for current temperature and humidity.
   b. The **output** can be simple alphanumeric output values displayed as text; no need for graphics.

For example, for humidity the output will be:

```
Current Humidity:  57%
Maximum Humidity:  59%
Minimum Humidity:  31%
Humidity Trend:    Increasing
Humidity Check:    High
```

Do the same for temperature

**Make sure your code compiles and runs correctly.**
**Take screenshot of 3 different input pairs (three pairs of temp & humidity) and the corresponding output values and submit them with your source code.**

Tips:

- Make sure you initialize your values **correctly** after each reset.
- Think of the Reset as a function or a class
- Java has multiple time libraries that you could use if needed
- Select your sensor reading intervals (and hence your reading refresh rate) wisely. Reading too frequently (e.g. every second) will deplete the battery fast. Reading every few minutes will provide low-quality feedback (with high latency.)

# Lab 5: Embedded Systems Testing
## Part 2
## Advanced Embedded Systems Testing

**Question a)** Design at least **two** test cases (using assertion) to test each of the output values separately.
Write your test criterion
Write your complete test requirements set TR
Write your complete test set T

*Hints:* *Write a different test case for each of the following outputs: trend, max, and min*
*You will need to input at least two values for each test before you check the output*

Each test will include test code (script) with input values **or input sequences** as needed for each test

**Question b)**
What is the total number of test cases?
What is the total number of test types?
*Hint: Changing the input and expected values of a test case generates a different test case of the same test type.*
What is the size of your test set?
Run your tests and take a screenshot of each one of them.
Submit your test scripts

**Question c)** Data-Driven Testing to avoid test code bloating:
*Hint: Review slides 21 & 22 of chapter 3 in the textbook, the "adding two numbers" example", @Parameters*

Write the following tests using the given values in two different testing styles, style 1 & style 2:

**Style 1:** One sequence of seven "Temperature, Humidity" pairs, followed by ONE test at the end of the given sequence (i.e. **one test for the entire sequence** of the seven input pairs).
This extended sequence testing is often used to test non-functional requirements before and after hardware/software integration like checking for memory leaks, overheating, component saturation, or excessive battery drainage. A typical "extended sequence test" could have thousands of sequence values automatically generated, stored, and read from an array.
In this case, the entire sequence will be **One test case** and **One test type**

**Style 2:** One pair of "Temperature, Humidity" input followed by a test; repeat seven times
(i.e. seven independent tests: **One test for each pair of input values and their expected output**).
In this case, the sequence will be **Seven test cases, all have the same test type**

i. Use the following **temperature** sequence after a reset: **66, 68, 69, 67, 63, 59, 53**
(Use **any** arbitrary value for the corresponding humidity readings)

Submit your test scripts for both style 1 & style 2
Take a screenshot of the output values after **each test** is run (style 1 & 2) and include them in your deliverable

ii. Use the following **relative humidity sequence** (in %) after a reset: **53, 51, 48, 49, 54, 56, 56**
(Use **any** arbitrary value for the corresponding temperature readings)

Submit your test scripts for both style 1 & style 2
Take a screenshot of the output values after **each test** is run (style 1 & 2) and include them in your deliverable

**Question c)** Refactoring:
You will notice that the algorithms used for humidity and temperature are similar (for calculating the **max**, the **min**, and the **trend**).
Refactor your code (if needed) to use one algorithm that could be used in both humidity and temperature calculations.

**Part 2 Deliverables:**
i) Submit your test scripts as explained above

ii) Include one screenshot of the 9 output values after each of the two sequences (question b) are read by the system in both styles 1 & 2

iii) Submit your refactored code (with comments)

iv) Answer **question b** above

v) Answer the following questions:

  a. What is the difference between testing the 7 inputs **in a sequence** and testing them **individually**. How are the two test cases **designed**? (use narrative description, no test code needed.)
  b. As the same person who developed, refactored, and tested the code, does your refactored code make it easier or harder to test the system, explain with examples
  c. If you received the refactored code (written by another developer) to just test it, would it be easier or harder than case iv) above?
  d. Would you prefer to test the original code or the refactored code, if both were written by another developer?