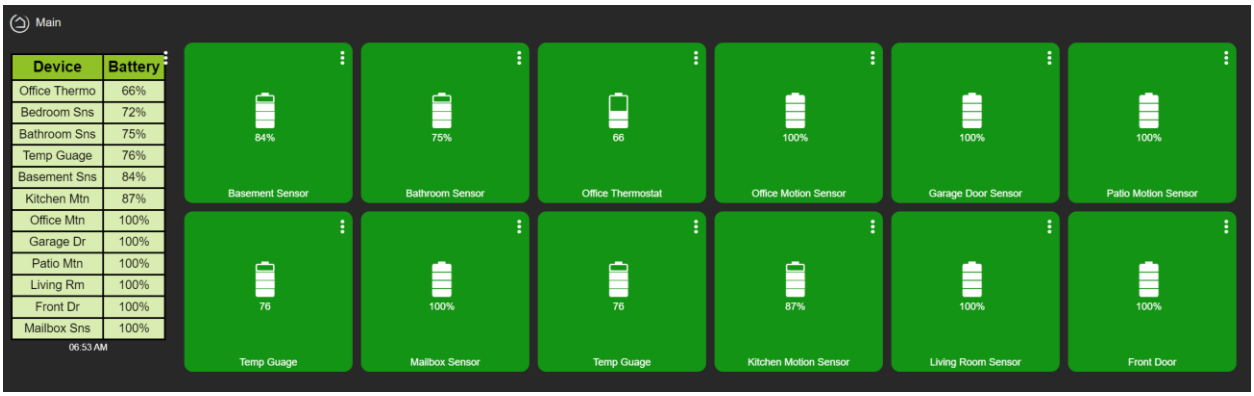
Tile Builder Standard Help
Version 1.0

Table of Contents

Introduction

Tile Builder is a novel way of presenting data on a Hubitat Dashboard. Rather than each tile being a single unique device, Tile Builder allows data from multiple devices to be presented on the same tile in a highly customizable tabular format.

It is entirely native to Hubitat and does not require any third-party elements or knowledge of CSS to achieve quite impressive results. The image below shows how one of these tables vs the traditional method of using individual tiles.



One of the biggest benefits of Tile Builder is that it allows information to be presented at a much higher density and with a much greater degree of control over color, style, placement, effects etc. without any technical knowledge.

There are many devices that only gather data without offering any control. For example, temperatures, motion, humidity, battery, contacts, presence, leaks etc. all merely present data without any corresponding action. The example below is built entirely using Tile Builder Advanced except for the 2 thermostat controls and the 3 image tiles for cameras. I place all the action items on separate dashboards which are only used when something does not seem right on the main page. The various styles show the level of control available in a Tile Builder tile.

How Tile Builder Works

There are 4 components to Tile Builder all of which are required for installation.

1) **Tile Builder Parent App**
2) **Attribute Monitor** (child app) – Generates device tables using a single attribute such as the first battery example.
3) **Activity Monitor** (child app) – Generates device tables using the 'Last Activity' attribute and can be used to monitor both active and inactive groups of devices.
4) **Tile Builder Storage Driver** – Device driver used for storing **Tile Builder** data.

The Tile Builder parent app is the primary organizing app.

Tiles are generated by one of the two child apps and organized under the parent app. When tiles are generated, the results are stored in the **Tile Builder Storage Device** in a specific named tile attribute (tile1 – tile25). This attribute is then placed onto the dashboard as shown previously.

Attribute Monitor uses an event subscription model. If the value of one of the monitored attributes changes the table is automatically regenerated and is updated on the dashboard. This happens within fractions of a second without any perceptible delay.

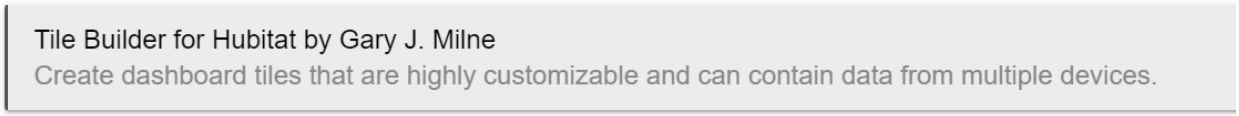
Activity Monitor only relates to the **Last Activity** time on a device and uses a timed model. A query is constructed via the child app such as 5 most inactive battery devices or devices that have been inactive over a week etc. A refresh interval is assigned from minutes to months. When that interval is reached the table is regenerated, the results are stored in the **Tile Builder Storage Device** and immediately displayed in the dashboard.

The Hubitat dashboard has a limit of 1,024 bytes for any attributes that are display and **Tile Builder** operates within this constraint without tricks or workarounds. The simple rule of thumb is the more data you want to display the less features can be displayed. But don't worry, **Tile Builder** offers a great deal of optimization and guidance to help you find that balance point. When you build a tile the size of the tile and which components are active is always displayed along the bottom.

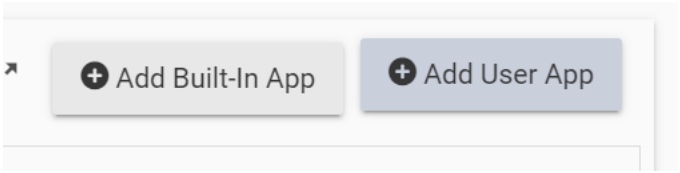
I have been able to display up to 20 devices in the most basic of tables. But if you want a moderate level of customization then 8-12 devices is probably more realistic. With all display features turned on, including highlights and animations (Advanced Version), 4-8 devices is probably a reasonable estimate.

Tile Builder Installation

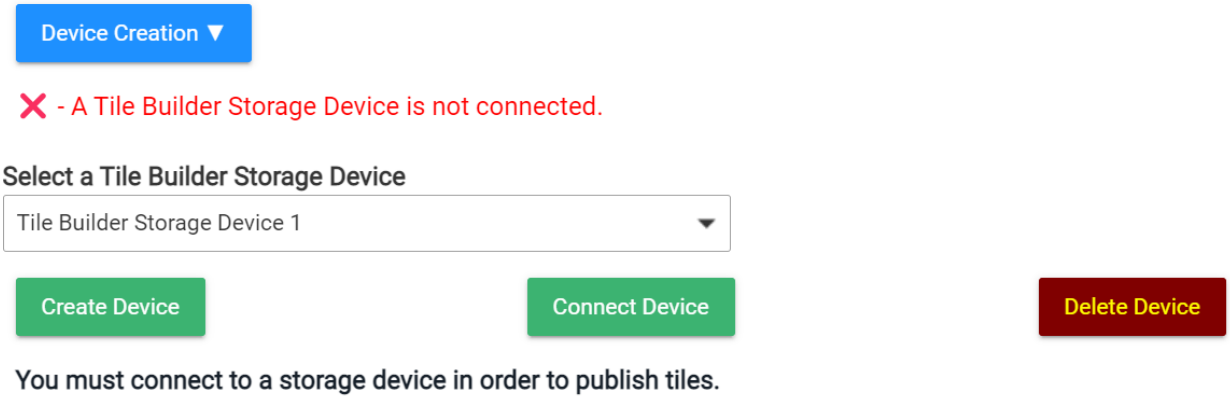
Tile Builder is listed in Hubitat Package manager. Choose to install by tags and select the **Dashboards** tag. Select **Tile Builder for Hubitat** and complete the installation process. This will place the code on your hub but there are a few steps to complete the installation.



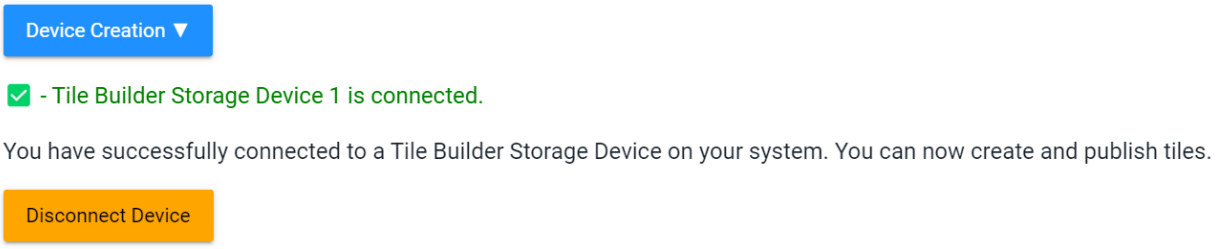
1. Go to the Apps tab and click on **Add User App**



2. Select **Tile Builder** from the list of available apps.
3. **Tile Builder** will install and bring you to the parent screen.
4. **Select the appropriate License Type (see paragraph below on licensing).**
5. Under **Device Creation** you must first create the storage device and then connect the app to the device. Just use the default device for now.



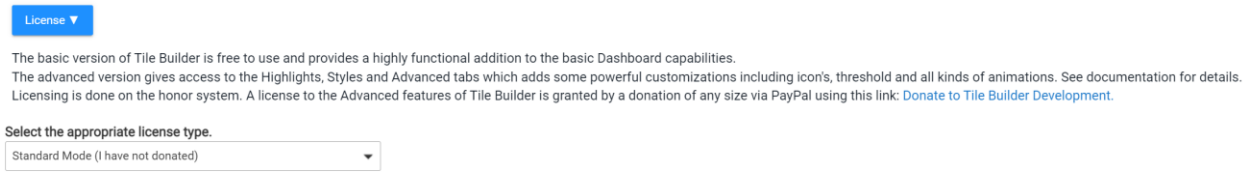
6. Once the device is created and connected it will look like this.



7. We can now create our first tile.

Licensing

Tile Builder has Standard and Advanced versions. The Standard version has a great degree of functionality and is entirely free to use as much as you wish. The Advanced version adds some very powerful features such as highlighting, thresholds, styles, overrides and a lot more.

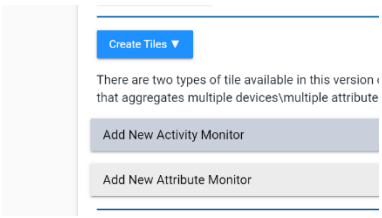


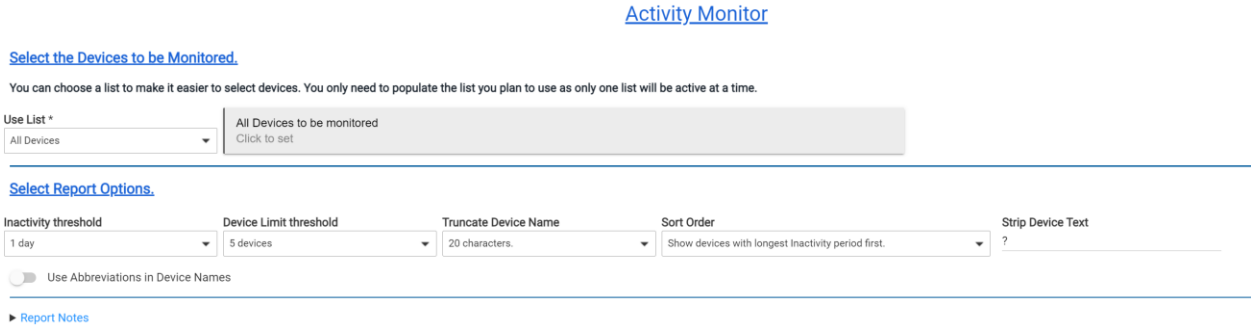
To license the Advanced version, you must make a monetary contribution via the link in the app. There is no specific price, but if you can tip a barman $2 for the time it takes to pour a beer then the 500+ hours that I have put into this should be worth something. When you donate, please include your Hubitat Community ID and I'll put you on my nice list with the highest priority for answering questions, bug fixing and feature requests.

Your donations free me up from other money generating activities and will allow me to add some additional modules to **Tile Builder**. I have several additions in mind, whether those get built and released largely depends on whether there is any room in the Hubitat ecosystem for a developer supported model.

Creating a Tile Using Activity Monitor

Within the **Tile Builder** parent app go to the section called **Create Tile** and select **Add New Activity Monitor**. The **Activity Monitor** main screen will be displayed. The top part of the display controls which devices will be monitored and how that information will be displayed.

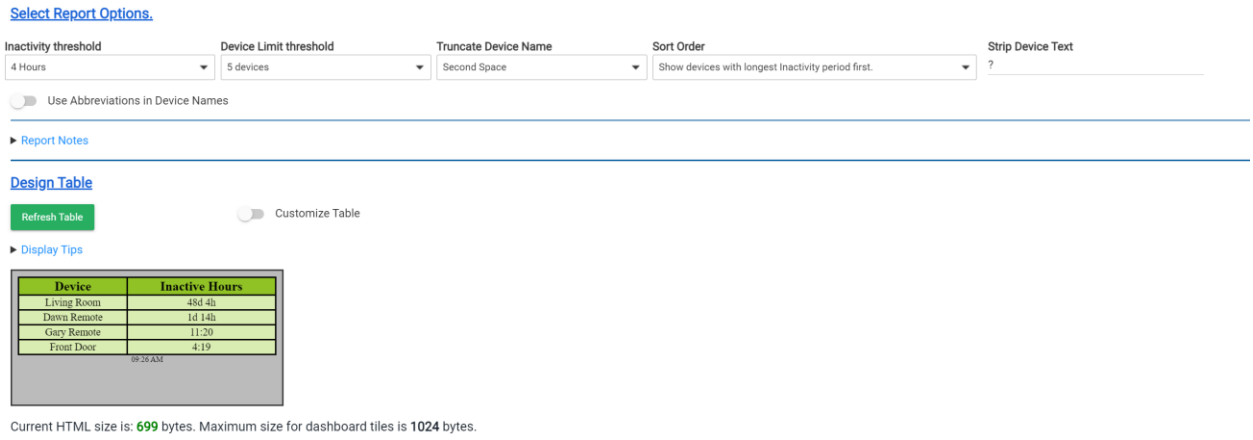




We will create a sample looking for inactive battery devices. Note: I have some battery devices that fail while the battery still claims to be at 100% so using the percentage alone is not a guarantee of catching a failed battery.

1. On the **Use List** dropdown select **Battery Devices**.
2. Click on the selector now titled **Battery Devices to be Monitored**. This is a filtered list of all your devices that have a battery capability. Select all the devices.
3. The **Inactivity Threshold** will limit the results to only those devices whose inactivity is greater than the threshold value. The default is 1 day, let's change that to 4 hours.
4. The **Device Limit Threshold** limits the display to no more than this number, but it may be less. We will leave that at the default of 5.
5. The **Truncate Device Name** allows you to shorten the device name and is useful in shrinking the amount of data space required, making devices names more consistent in length and to reduce text wrapping. In this example I chose to truncate at the second space.
6. The **Sort Order** allows you to sort the table by oldest or youngest. Given we are looking for devices that might have a failed battery we will set it to **Show devices with longest Inactivity period first**.
7. The **Strip Device Text** allows you to specify a string that will be completely eliminated from the device names. I sometimes use ascii characters such as ! or ~ to group like devices together in the device table and device picker. I can strip those characters from the final display name. We can leave this at the default for this example.
8. The **Use Abbreviations in Device Names** is an option for reducing the size of the data in the result set. With this enabled the word " Room" becomes " Rm", " Sensor" becomes "Sns" etc. When trying to cram a lot of data into a table this can be a very useful option but mostly of use in Attribute Monitor where the result sets are longer. We will leave this turned off here.

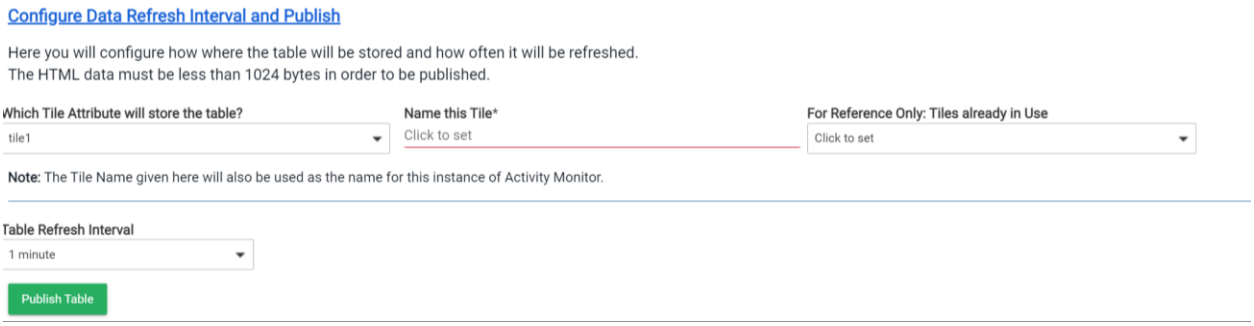
Based on these parameters my report looks like this. Only 4 battery devices have not been active in the last 4 hours. The **Living Room** device (a sensor) obviously needs a battery replaced despite the fact the last report stated the battery was at 100%.



All the table customizations are hidden unless the **Customize Table** option is checked. We will come back to that, but at this point there is nothing else that is absolutely required and we can publish the table as it currently is.

Publishing a Tile

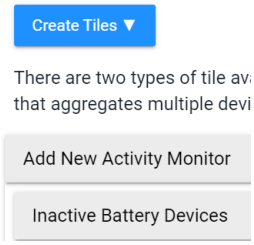
The publishing options are shown below.



There are just a few steps.

1. Select the **Tile Attribute** to store the table in. Tile attributes are tile1 – tile 25. We will use tile1 in this case.
2. **Name the Tile**. I'm going to call it **Inactive Battery Devices – Tile 1**. This is also the name that will be used when looking at the **Tile Builder** parent app. I recommend you append the name with the tile number so you can see it on the parent screen.
3. Select a **Table refresh interval**. No need to make the hub work too hard. In this case I think hourly would be reasonable.
4. With those values set click on **Publish Table**.
5. Click on **Done** to close the **Activity Monitor** app.

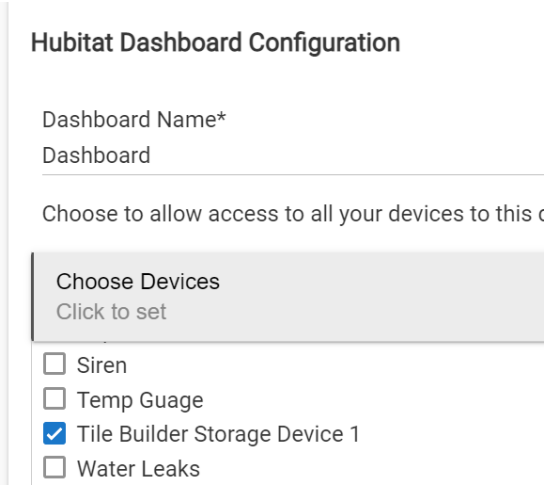
6. You should see your new tile listed under the **Activity Monitor** child app like this.



You can go back and edit this tile any time by clicking on this button.

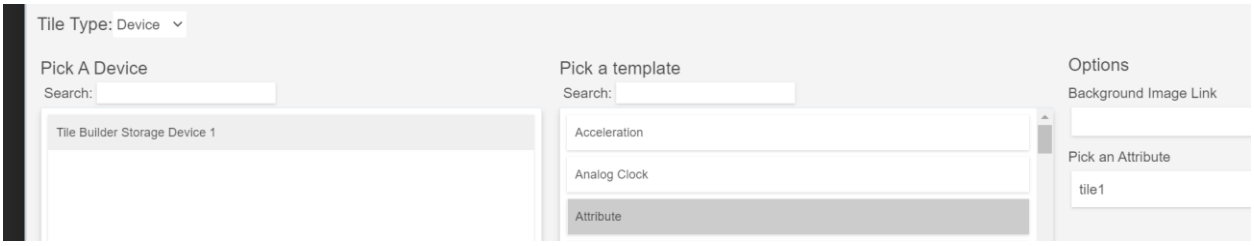
Dashboard Integration

We are done with **Tile Builder** for the moment and now we can add the newly generated tile to the dashboard of our choice. Before we can do that, we must first grant the dashboard access to the **Tile Builder Storage Device** as shown below.

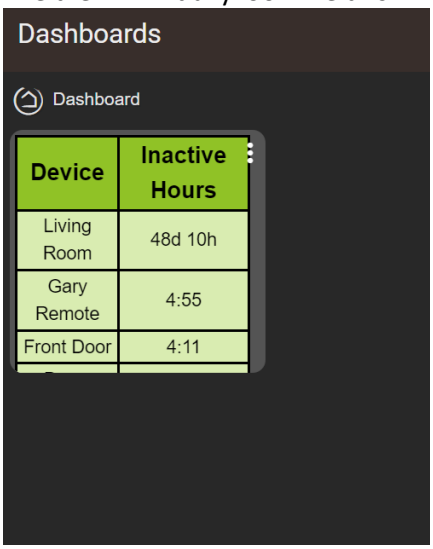


Note: Dashboard does not need access to the underlying devices that provide data for the table.

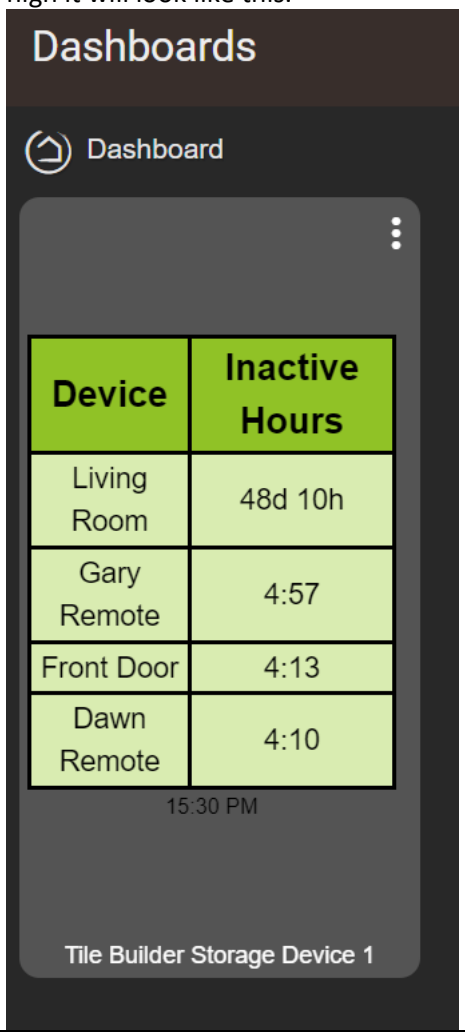
Next, we must add the newly generated tile to the Dashboard in the same way we would add any other device as shown below.



The tile will initially look like this.



By changing the dimensions of a tile to 1 wide x 2 high it will look like this.

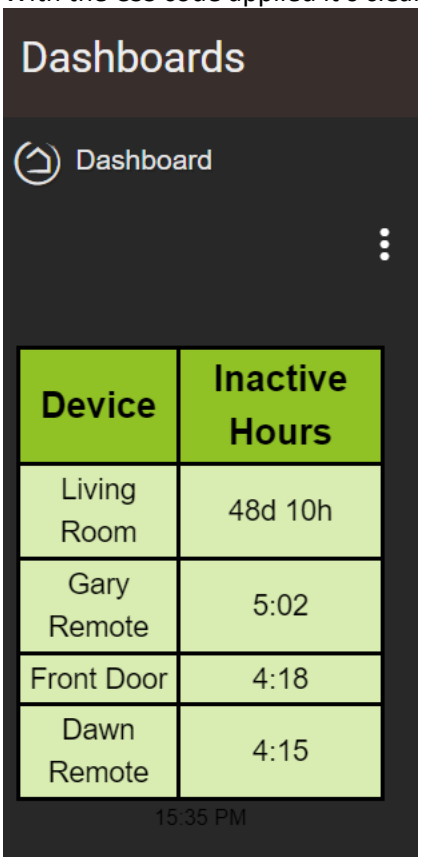


If you wish to you can improve the look of the tile by adding these lines to your dashboard CSS.

#tile-1 {background-color: rgba(128,128,128,0.0) !important;}
#tile-2 {background-color: rgba(128,128,128,0.0) !important;}
#tile-1 .tile-title, #tile-2 .tile-title {visibility: hidden; display: none;}

These make the tile background transparent and then hide the name of the device.
Note: The tile number referred to in the above CSS code has no bearing on the tile numbers assigned in the **Tile Builder** app.

With the CSS code applied it's cleaned up a little bit.



Not bad, it has the basic information but it's not going to win any design awards. Here are a few things I don't like about it so far.

1. The font is rather boring.
2. It's not clear what the table represents.
3. It's hard to the read timestamp at the bottom of the time. (Last time it was generated.)
4. The table border blends into the background.
5. Green is not my favorite color.

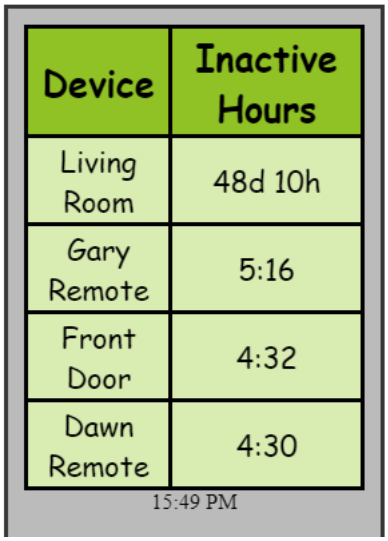
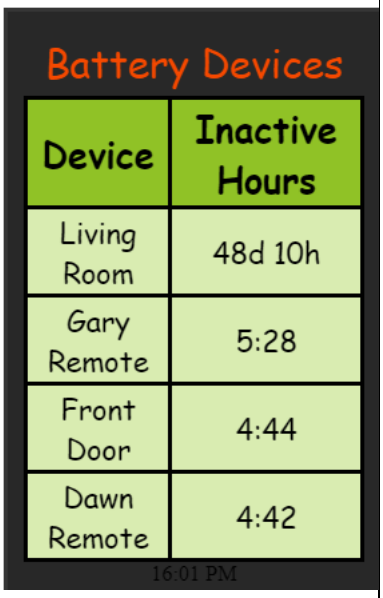
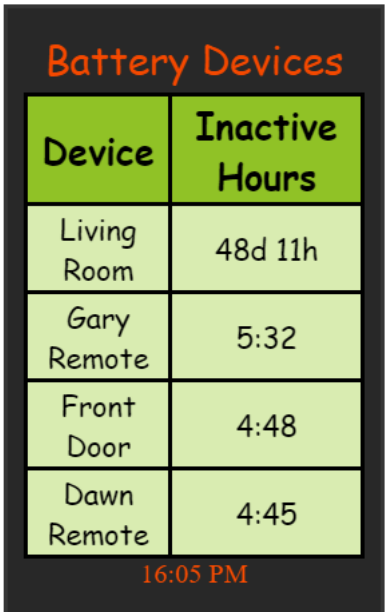
Let's go back and fix these and make it a little more pleasing.

Editing Activity Monitor Tiles

Let's open the tile in **Activity Monitor** again. This time we will click on **Customize Table**. In the standard version it looks like this.

Before we start, let's go to the **General** tab and change the **Select Tile Preview Size** to match our desired size. In this case **1 x 2**. Then click on **Dashboard Color** and use the eye dropper tool to grab the color from your dashboard. This makes the preview more useful, especially when selecting colors.

The following table shows the progression of the table as the different elements are changed.

General Tab Select font Comic Sans.	**Title Tab - Add Title** Size 110%, new color.	**Footer Tab** Size 75%, new color.
		

Borders Tab – Change Border
Type: Ridge, Width: 4, new color.

Header Tab & Row Tab
Change background color.

Title Tab
Add a title shadow.

As you can see it is quite easy to make changes to the table and the number of variations is limitless. The purpose of most of the controls is obvious from the name. Where some explanation is needed it is included under a notes section that exists under each tab. The following example is from the **Border** tab.

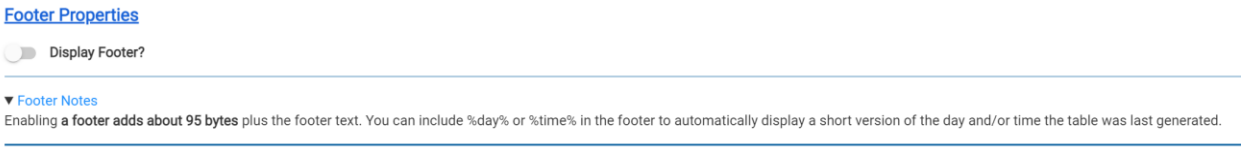
▼ Border Notes

Border Radius applies to each individual cell, not the table as a whole. A **border adds about 85 bytes**.
Border padding takes precedence over Header text and Row text padding. Header and Row padding settings are ignored whenever Border padding is > 0.
Using a setting of 'Border Mode' = Seperate on the General Tab can give the appearance of a border but consumes less space as borders can be turned off.

Tile Size Limits

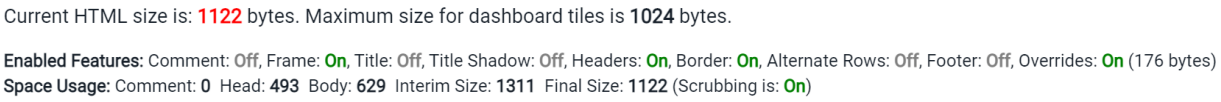
As previously mentioned, the maximum size of data published to a dashboard via a device attribute is 1,024 bytes. This is not a lot of room to work with, so it is helpful to be cognizant of the size of the tile you have created and what components are contributing to that size. **Tile Builder** has some capabilities built in to help optimize the tables.

1) Under the **Notes** for each section the added payload from enabling a given option is displayed as shown in this example.



2) The area below the table preview shows the current size of the table, the compressed size of the table and which options are currently enabled. For example:

3) **Scrubbing** removes all excess characters from the HTML making it extremely dense and is automatically turned on.
4) When the 1,024 byte limit is exceeded the size will be displayed in red and the **Publish** button will be greyed out (not shown).



There are multiple strategies to keep the size of the tile within the limits and these tools will help you find that optimal point more quickly. It is important to note that changing the amount of data in the table is an equally important strategy to sizing tiles appropriately.

Note: If you have **Tile Builder Advanced** you can load the style called **Everything Off** which uses all the HTML default values to render the minimal tile size. You can then add back\enable just the components that you deem necessary. If you have **Tile Builder Advanced** you will also see **Styles** and **Advanced** tabs available. You can find details of these in the separate **Tile Builder Advanced Help** document.

Creating a Tile Using Attribute Monitor

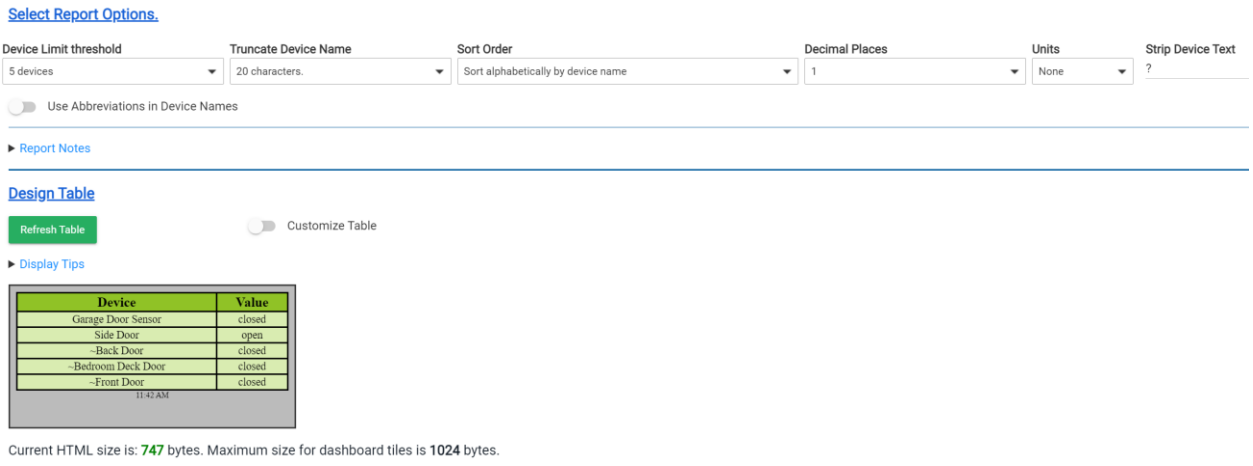
The process for doing this is very similar to that for creating an **Activity Monitor** tile so I will only focus on the differences.

In **Tile Builder** parent app click on **Add New Attribute Monitor**

We will create a sample to monitor the contact status of doors.

1. On the **Select Attribute to Monitor** dropdown select **Contact**.
2. Click on the **Select Devices to Monitor**. This is a filtered list of all your **Contact** devices. Select all your doors.

In my example the screen now looks like this.

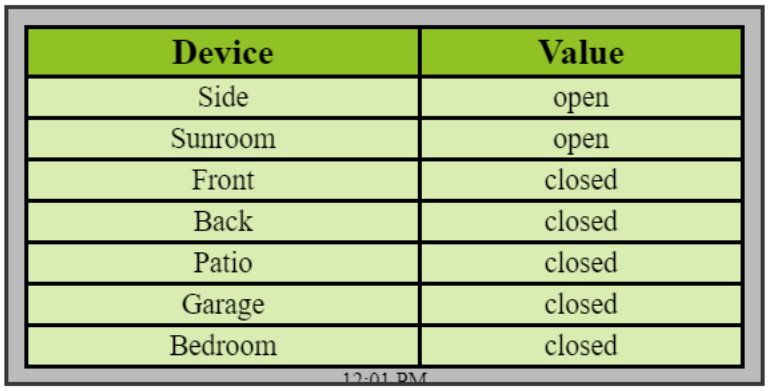


Let's go through each of the report options.

1. **Device Limit Threshold**: You can limit the number of devices displayed to be less than the number of devices being monitored. For example, I might monitor 12 temperatures but only display the 5 highest ones by using the appropriate sort order. In this case we will set the device limit to 7 so we see all the doors at once.
2. **Truncate Device Name**: Allows you to shorten the device name and is useful in shrinking the amount of data space required, making devices names more consistent in length and to reduce text wrapping. In this example I chose to truncate at the first space because I don't need the word door to be on every line.
3. **Sort Order**: Allows you to sort the table by device name, value, or value reverse. I care more about open doors so I will have them sort first by selecting the **Reverse Sort Alphabetically by Value** option.
4. **Decimal Places**: Applies only to floating point values.

5. **Units**: You can choose the type of units to display alongside the value. Not relevant here.
6. **Strip Device Text**: Allows you to specify a character or string that will be eliminated from the device names display. I sometimes use ascii characters such as ! or ~ to group like devices together in the device table and device picker. In this case my wired contacts have a ~ as the first character and it looks unsightly. I can strip those characters from the final display name by entering the ~ character in this field.
7. The **Use Abbreviations in Device Names** is an option for reducing the size of the data in the result set. With this enabled the word " Room" becomes " Rm", " Sensor" becomes "Sns" etc. We will leave this turned off here as we have truncated device names at the first word so they are pretty short already.

Our table now looks like this.

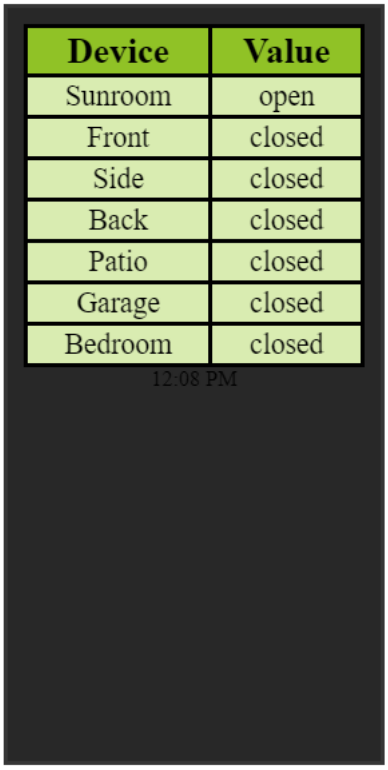


Device	Value
Side	open
Sunroom	open
Front	closed
Back	closed
Patio	closed
Garage	closed
Bedroom	closed

12:01 PM

As you can see the table is tight, the footer is getting cut off and I'd like to add a title to it so I'm going to go to the General tab and change the tile preview to **1x2** as I think that will be a better fit. I also changed the Dashboard Color to match my actual dashboard to give a more accurate preview. The table below shows the progression of the tile preview as I make each of the changes listed.

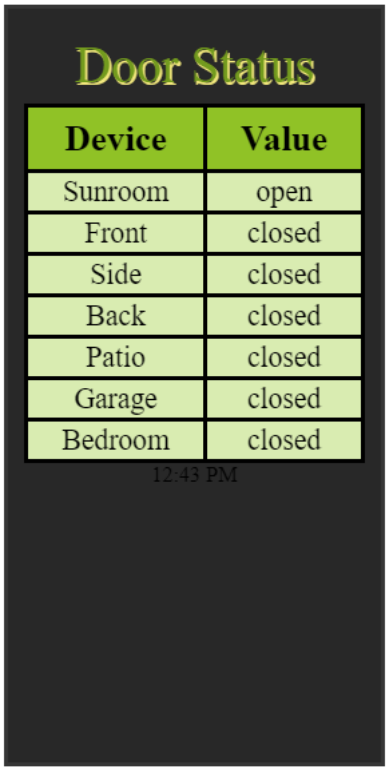
General Tab
Tile preview **1x2**, Dashboard color.



Device	Value
Sunroom	open
Front	closed
Side	closed
Back	closed
Patio	closed
Garage	closed
Bedroom	closed

12:08 PM

Title Tab
Add Title, color, padding, shadow.

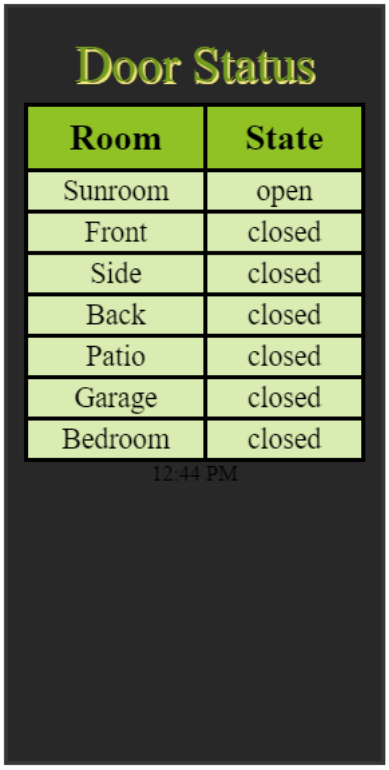


Door Status

Device	Value
Sunroom	open
Front	closed
Side	closed
Back	closed
Patio	closed
Garage	closed
Bedroom	closed

12:43 PM

Headers Tab
Header text



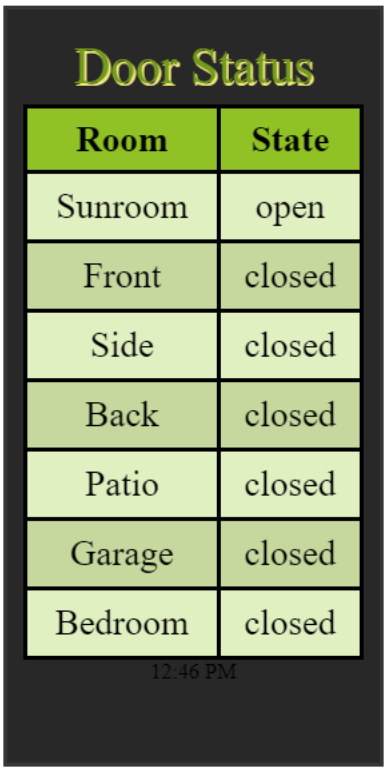
Door Status

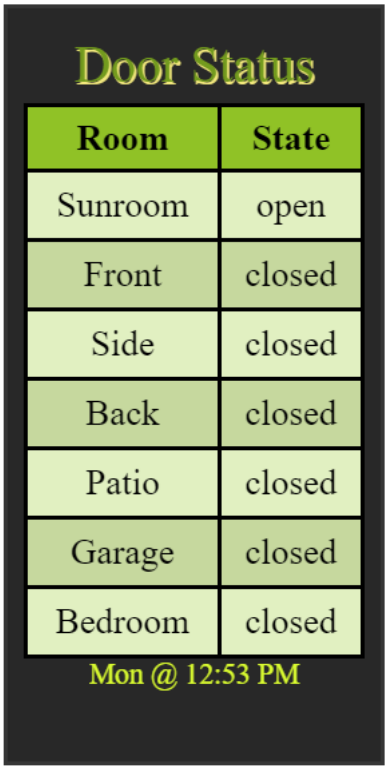
Room	State
Sunroom	open
Front	closed
Side	closed
Back	closed
Patio	closed
Garage	closed
Bedroom	closed

12:44 PM

Rows Tab
Text padding, alternate row color, background opacity.

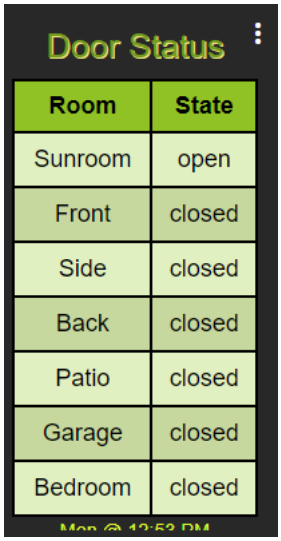
Footer Tab
Footer text %day% @ %time%, text color.





When I first published the result to the dashboard it looked like the entry in the before column. Notice how the footer is partially cut off. This demonstrates that while the preview is a close approximation it is not exact. It's an easy fix, I went back to the Rows tab and changed the **Text Padding** from 6 to 5 and then everything displayed as expected.

Before



After changes to row text padding.

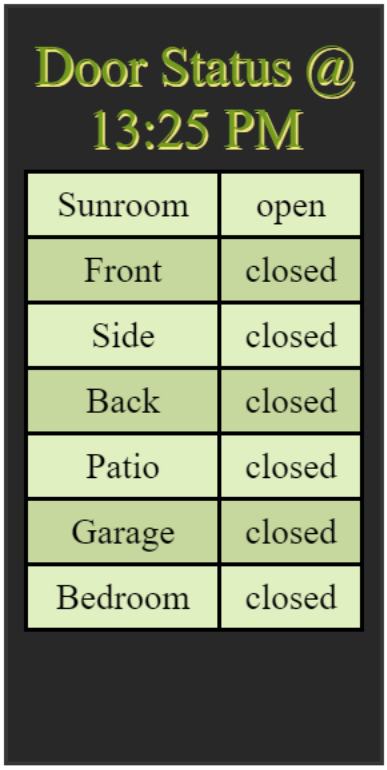
Tile Size

In this case the final tile size was 1,008 bytes after scrubbing and close to the upper limit of 1,024. As a **closed** door uses 2 bytes more than an **open** door, we need to leave a some buffer space to handle the resulting variations in tile size.

Current HTML size is: **1008** bytes. Maximum size for dashboard tiles is **1024** bytes.

Enabled Features: Comment: Off, Frame: Off, Title: On, Title Shadow: On, Headers: On, Border: On, Alternate Rows: On, Footer: On, Overrides: Off (0 bytes)
Space Usage: Comment: **0** Head: **546** Body: **462** Interim Size: **1266** Final Size: **1008** (Scrubbing is: On)

If we wished to add more lines of data, we would have to cut some of the features that were turned on as shown above. In this case I'd think about turning off the header rows as the title tells me what the rest of the table is about. I would also consider moving the footer text into the Title. If I do both of those things the tile now looks like this:



If we look at the HTML info, we can see the tile size has shrunk from 1,008 bytes to 789. That is a big difference and would be sufficient to add another 5-10 rows of data if desired.

Current HTML size is: **789** bytes. Maximum size for dashboard tiles is **1024** bytes.

Enabled Features: Comment: Off, Frame: Off, Title: On, Title Shadow: On, Headers: Off, Border: On, Alternate Rows: On, Footer: Off, Overrides: Off (0 bytes)
Space Usage: Comment: **0** Head: **470** Body: **319** Interim Size: **1000** Final Size: **789** (Scrubbing is: On)

Appendix 1- Advanced Topics

Embedded HTML Tags

Anywhere you can enter text you can wrap it inside HTML tags. But rather than using **<>** you must use **[]** as normal html tags are rejected by the Hubitat interface. For example, you could enter **[u]Door Status[/u]** in the title field and the title would be displayed in underline, **[b]Door Status[/b]** and it would display in bold. Multiple html tags can be used if they follow html conventions.

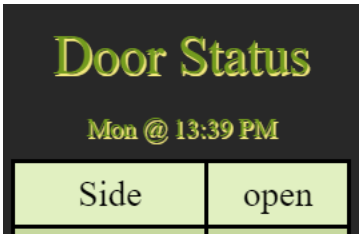
Macros

The following values are macros that will be expanded in the final html.

- **%day%** will be replaced by a short version of the day name
- **%time%** will be replaced by a 24 hr time including AM\PM
- **%units%** will be expanded into the Units chosen, if any.

In the above example the **Door Status** title text was replaced with **Door Status @ %time%.**

Using the text **Door Status [br][font size=2]%day% @ %time%[font]** as the title would look like this.



You can use these macros in any text field combined with html tags.

Publish and Subscribe Model

When you click on the **Publish and Subscribe** button, **Tile Builder** creates an event subscription to each of the selected devices and chosen attribute. **Attribute Builder** then remains dormant until such time as one of the monitored attributes changes, in the above case a door opens or closes, at which point the table is immediately regenerated and published. This is a highly efficient model, and **Tile Builder** tiles will only regenerate when the underlying data has changed. Tile updates will appear on the dashboard at the same speed as a typical device tile.

Hubitat CPU Utilization

You can view the performance of **Tile Builder** apps under **Logs\App Stats**. In the below example you can see that the **Battery Status** has only been initiated 5 times since the last reboot (reboots nightly) and has consumed a total of 0.004% of total CPU. This is not unexpected as battery percentage changes do not occur often.

Name	Total, ms	Count	Avg, ms	% of busy	% of total	State size
Add device	0	0	0	0.0	0.000	71
Advanced Button Controller	0	0	0	0.0	0.000	21
Amazon Echo Skill	3,987	57	70	0.2	0.011	1,011
Basement Lights (Paused)	333	8	42	0.0	0.001	2,018
Battery Status	1,568	5	314	0.1	0.004	9,808

In addition to the **Tile Builder** apps utilizing the CPU the larger consumer is the **Tile Builder Storage Driver**. You can track this in Logs\Device Stats as shown below.

In the instance shown below there are 5 published tiles, and the **Tile Builder Storage Device** is about as busy as a thermostat.

Name	Total, ms	Avg, ms	% of busy	% of total
Envisalink	1,362,152	244	93.5	3.832
Tile Builder Storage Device 1	19,696	13	1.4	0.055
House Thermostat - TCC	17,424	143	1.2	0.049
Temp Guage	15,736	11	1.1	0.044
Office Thermostat	14,270	10	1.0	0.040

Final CPU utilization in any given case will be a function of how many tiles you have published and especially how frequently those tiles are updated. For this reason, publishing rapidly changing data such as power consumption needs to be done thoughtfully.

Summary

As you can see, **Tile Builder** offers a great deal of flexibility in how to display data onto a Hubitat dashboard. What you have learned so far should keep you busy for a while. While Tile Builder has a lot to offer, **Tile Builder Advanced** is a big step up in terms of flexibility and control. The key features of Tile Builder are described below.

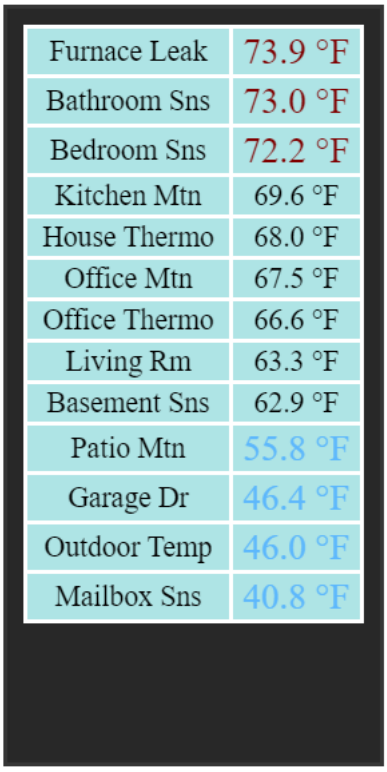
Thresholds: Allow for the formatting and substitution of results based upon their values using font color, font size, icons and even animation.

Styles: Allows you to save any group of settings into a style and apply them to as many tiles as you wish. **Tile Builder** also has multiple built in styles that you can use or modify and create your own. Styles can be shared on Hubitat Community forums as simple cut and paste text strings. The style example table below uses one set of data and simply applies different styles to it.

Advanced: See the style table below for some of the amazing effects available with the Advanced version. Not shown are animation effects.

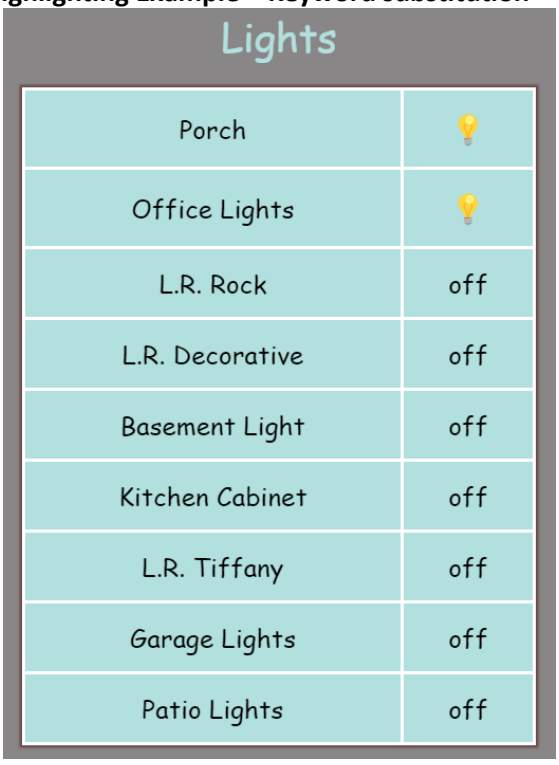
A few examples of these customizations are shown in the tables below. See the Advanced guide for full details and more examples.

Highlighting Example – Value Highlighting



Furnace Leak	73.9 °F
Bathroom Sns	73.0 °F
Bedroom Sns	72.2 °F
Kitchen Mtn	69.6 °F
House Thermo	68.0 °F
Office Mtn	67.5 °F
Office Thermo	66.6 °F
Living Rm	63.3 °F
Basement Sns	62.9 °F
Patio Mtn	55.8 °F
Garage Dr	46.4 °F
Outdoor Temp	46.0 °F
Mailbox Sns	40.8 °F

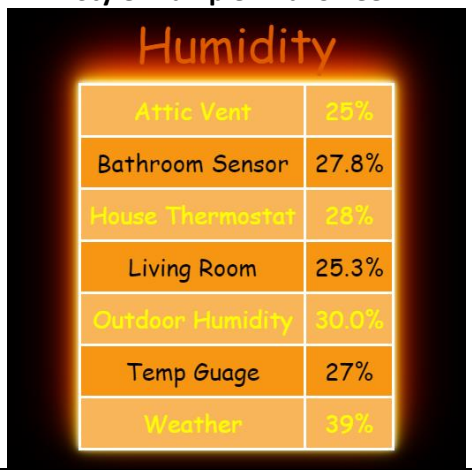
Highlighting Example – Keyword substitution



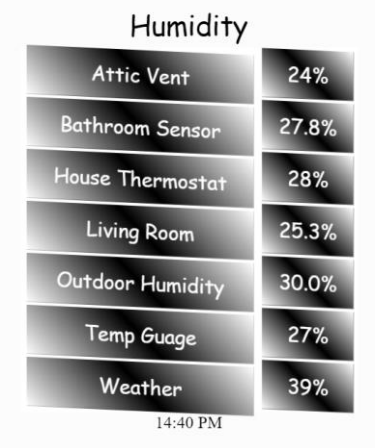
Lights

Porch	💡
Office Lights	💡
L.R. Rock	off
L.R. Decorative	off
Basement Light	off
Kitchen Cabinet	off
L.R. Tiffany	off
Garage Lights	off
Patio Lights	off

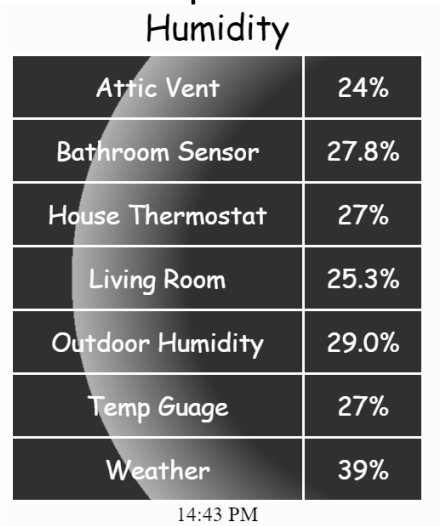
Style Example - Halloween



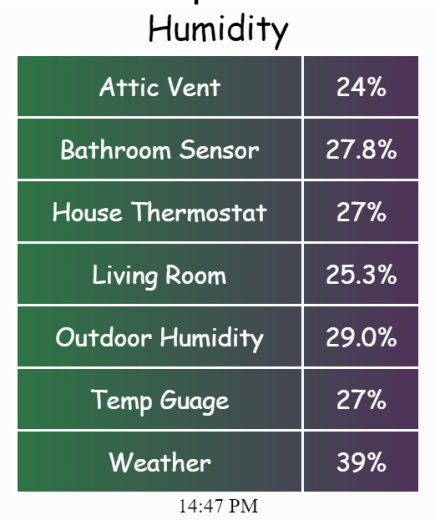
Style Example – Black and White



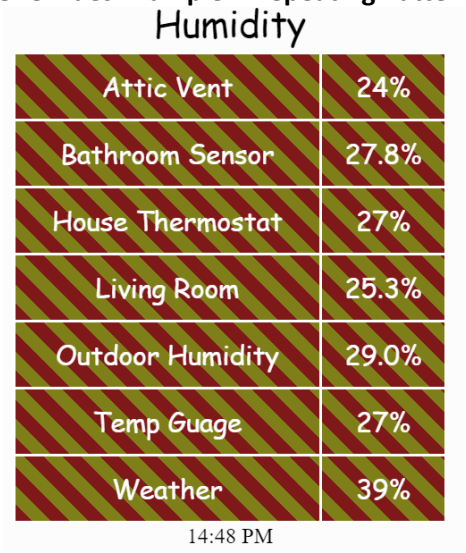
Overrides Example – Radial Gradient



Overrides Example – Color Gradient



Overrides Example – Repeating Pattern



Overrides Example – 3D Effect