Blinds IoT

An MQTT powered IoT Environment

Written and Produced By, Luke Jenson

June 14th, 2025

**Introduction**

The combination IoT device is comprised of an MQTT device operating out of a Python script. It functions to simulate temperature, light (lux), and sun location (radian) data publishing data to Make.com and ThingSpeak. This data can then be viewed in ThingSpeak for each field of data and get a live visual representation. Make.com takes the data and publishes it to a Google sheet, performs mathematical calculations to set the blinds, and publishes them through an HTTP request as raw JSON data to webhook.site. A second Python function reads data from ThingSpeak and then writes to a JSON document with the average Temperature. This dataset is imported to an HTML and Java Script environment to be graphed and displayed online.

**Generating Synthetic Data: ThingSpeak MQTT in Python**

An MQTT application was written with five functions. Three functions, temperature, sunlight, and radian, generate unique variables called payloads and web\_payloads for ThingSpeak and Make.com. The publish function posts the data depending on the passed-in channel and webhook URL. The main function loops through at 20-second intervals for 1440 seconds (allows the simulation of a 24-hour day in 24 minutes), calling each function once through a sixty-second period. (see attached MQTT\_Application.py file). The opening of the IoT device establishes the necessary information for publishing to Make and ThingSpeak (see Figure 0).

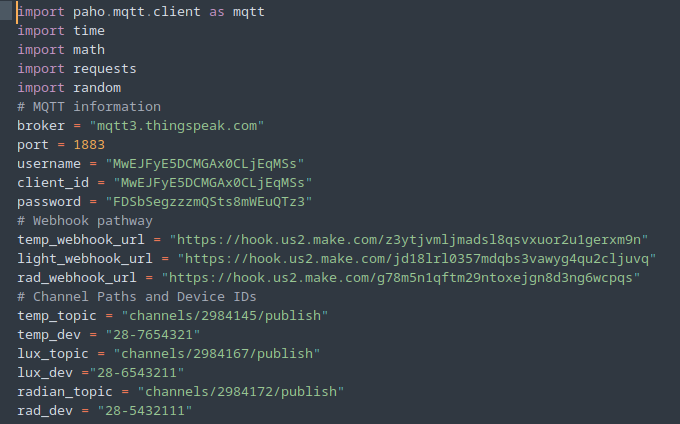
****

Figure 0: MQTT IoT simulator using ThingSpeak.

**ThingSpeak**

The basic channel settings for temperature and light had five fields. One field for the device ID, and four fields for locations where blinds would be positioned in a house or building (North, East, South, and West) (see Figure 1). The radian channel had two fields one for the position of the user relative to the sun (radians) and the device ID. Light data was successfully input over 24 measurements, showing eastern light to be high in the morning (where the sun rises) and high in the west in the evening (where the sun sets), with an even distribution of light for North and South facing windows. Sun positioning was a linear movement from 0pi to 2pi representing the earth’s circumference relative to the user's location. Temperature data was slightly offset to a degree but showed unilateral rise and fall from horizon to horizon (see Figures 2-4).

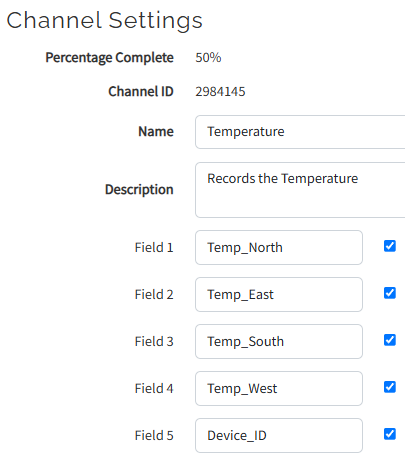
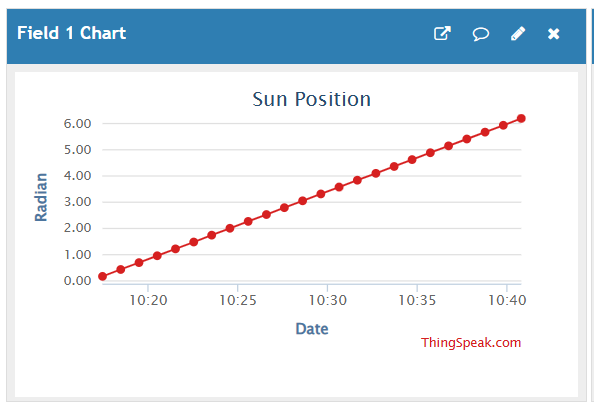
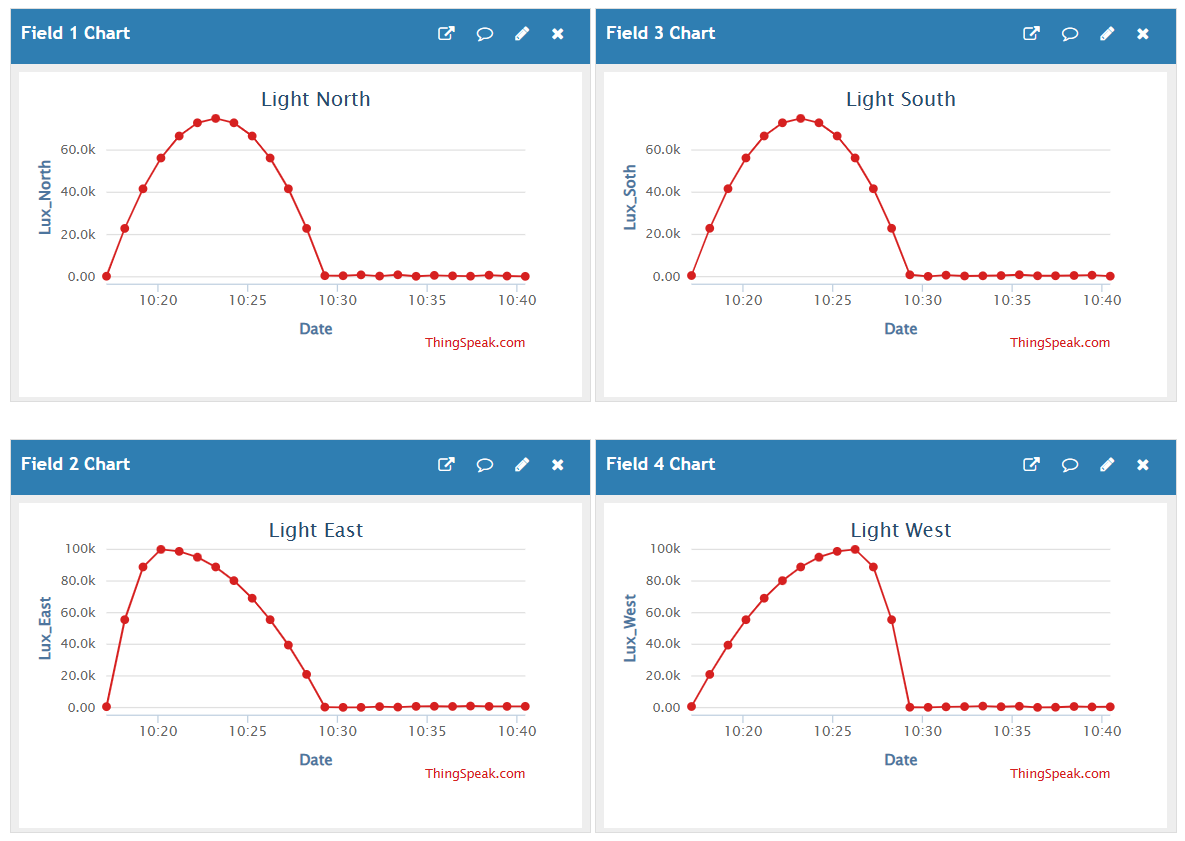
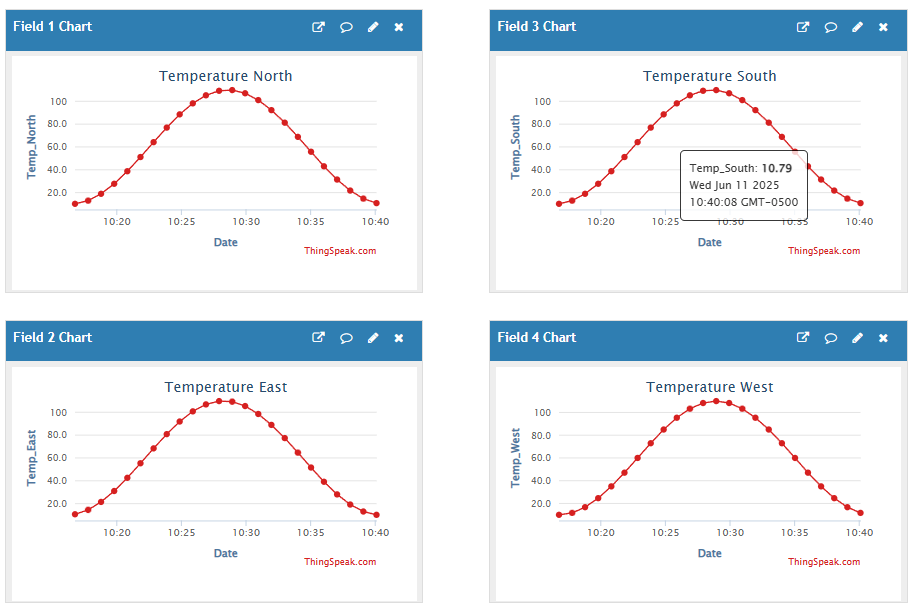
****

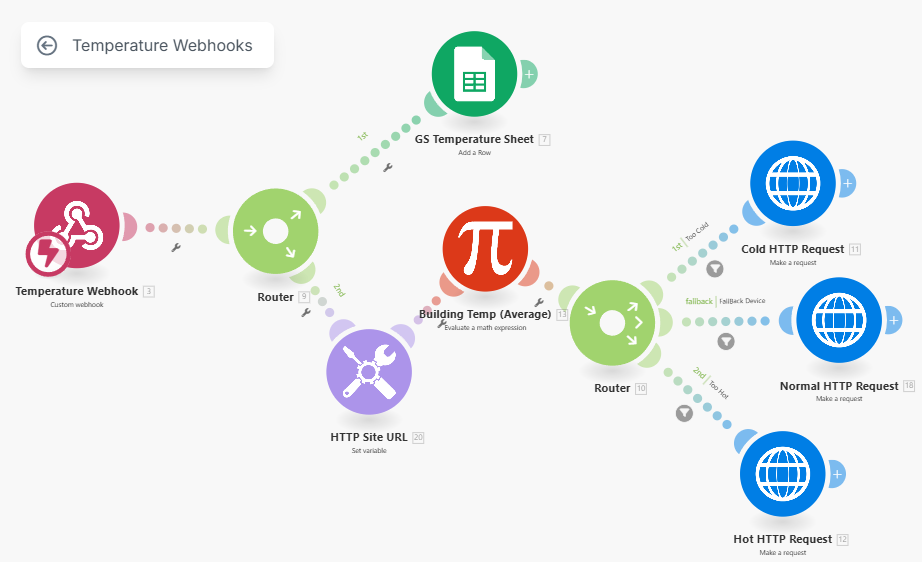
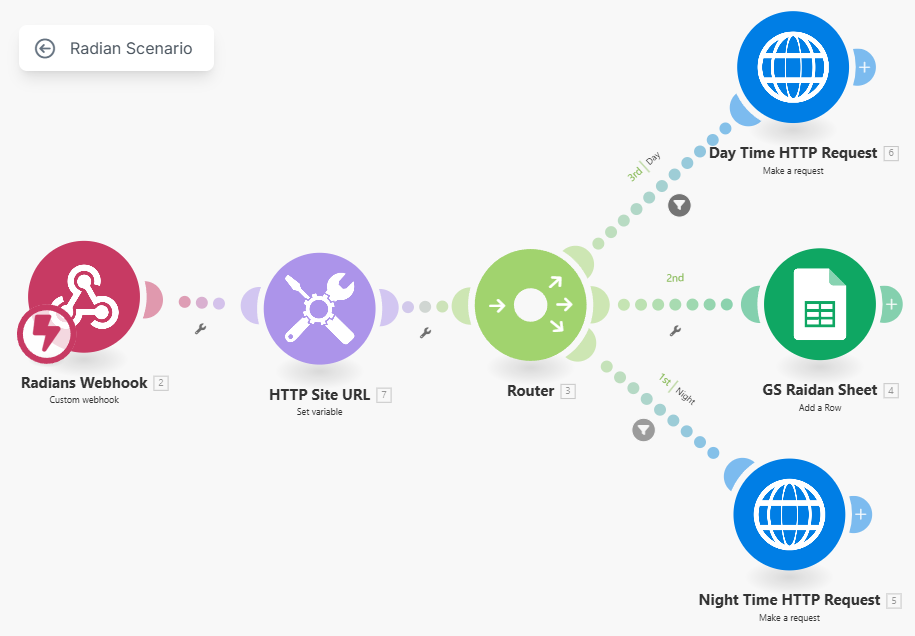
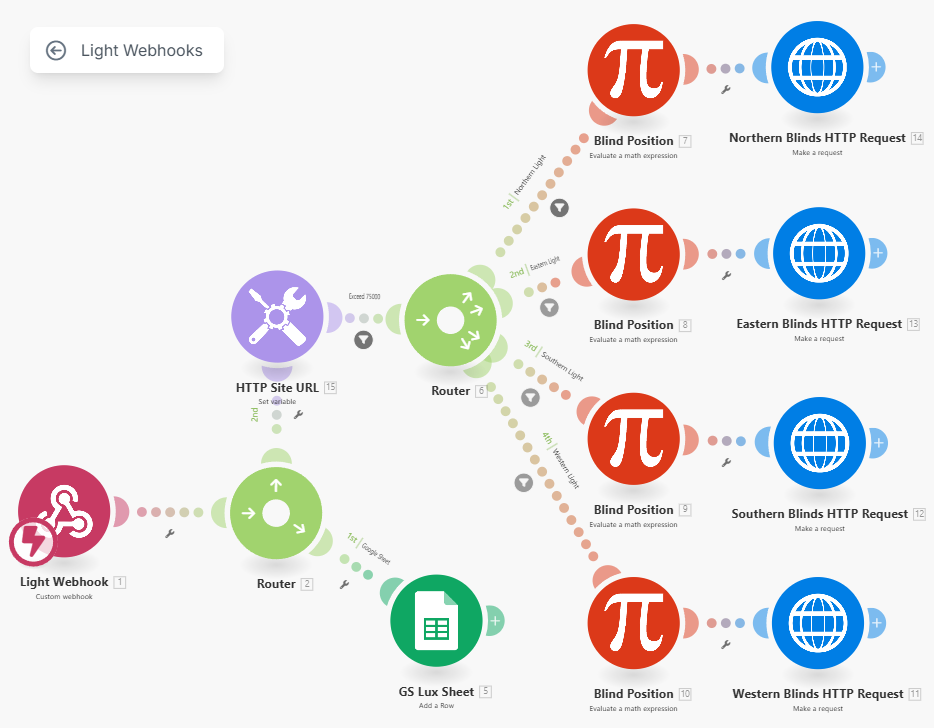
Figure 1: Example ThingSpeak Channel settings (Temperature Channel mirrors Light Channel).

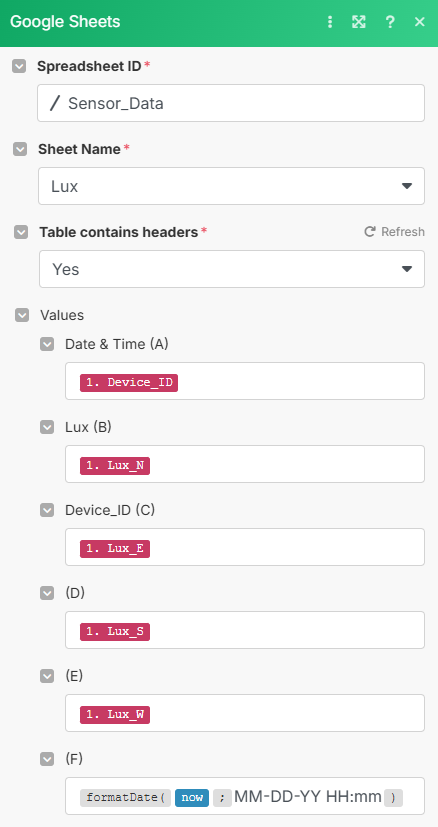
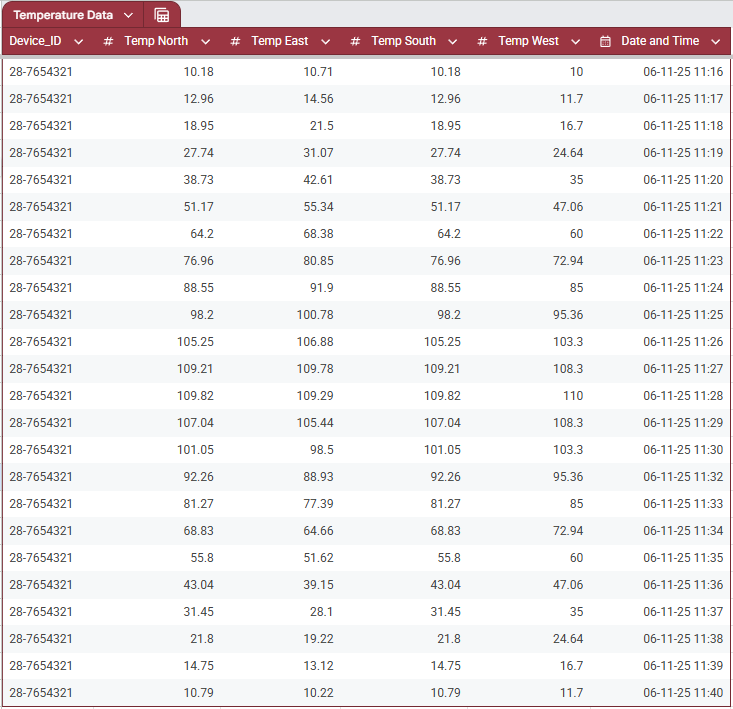
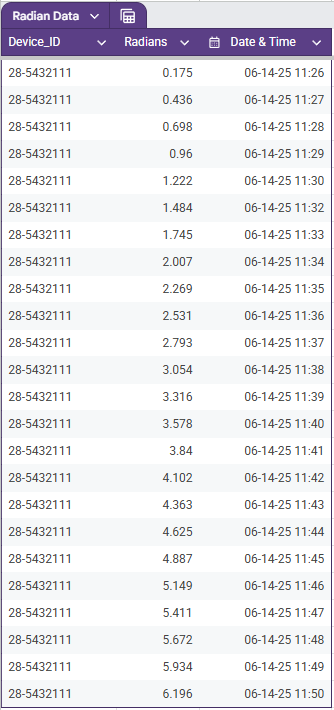
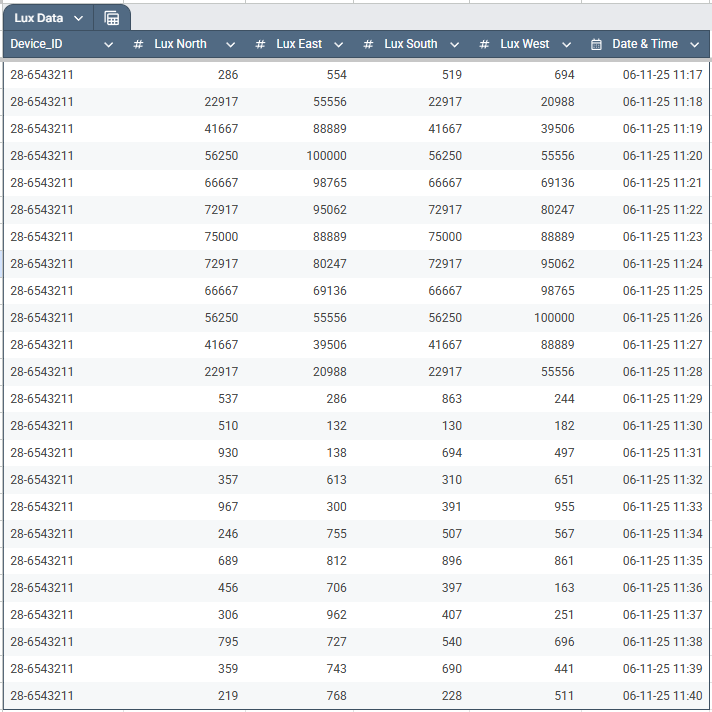
****

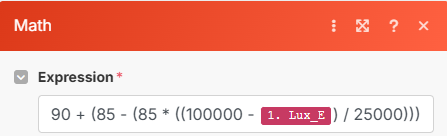
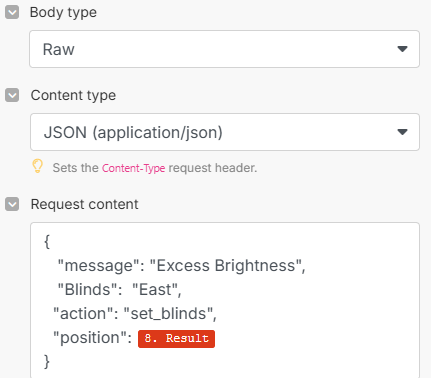
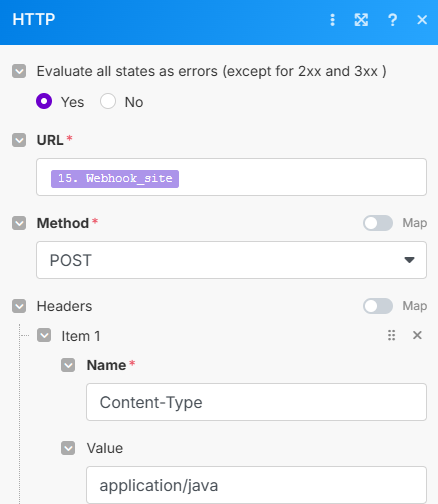
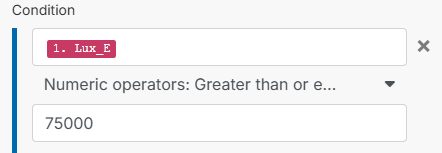
Figures 2-4: ThingSpeak data being populated in field charts restricted to 24 points.

**Make.com, webhook.site, and Google Sheets**

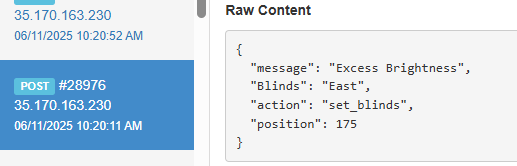
Three scenarios were created to capture and transform data, radian scenario, temperature scenario, and light scenario (see Figures 5-7). The first function of the sheets was to populate a Google spreadsheet with a set of coordinated measurements as they were being made available. The values for device ID and measurements were passed in with an additional date and time stamp (see Figures 8-11). The simultaneous routes are connected to HTTP requests passing in an established webhook.site URL variable (webhook.site URL stored in a variable). An example of this occurring is with the eastern-facing blinds when the light value exceeds 75,000lux the position of the blinds is recalculated in the math position function and then passed into the HTTP request (see Figures 12-14). Successful HTTP requests were published with the necessary calculated values for determining blind tilt under different lux, temp, and radian conditions(see Figures 15-17).

Figures 5-7: Light webhook, Radian Webhook, and Temperature Webhook take in MQTT data and route to Google Sheets and HTTP requests, webhook.site, under specified conditions.

Figures 8-11: An example of how the data was formatted in Google Sheets before being published to individual sheets under the spreadsheet Sensor\_Data.



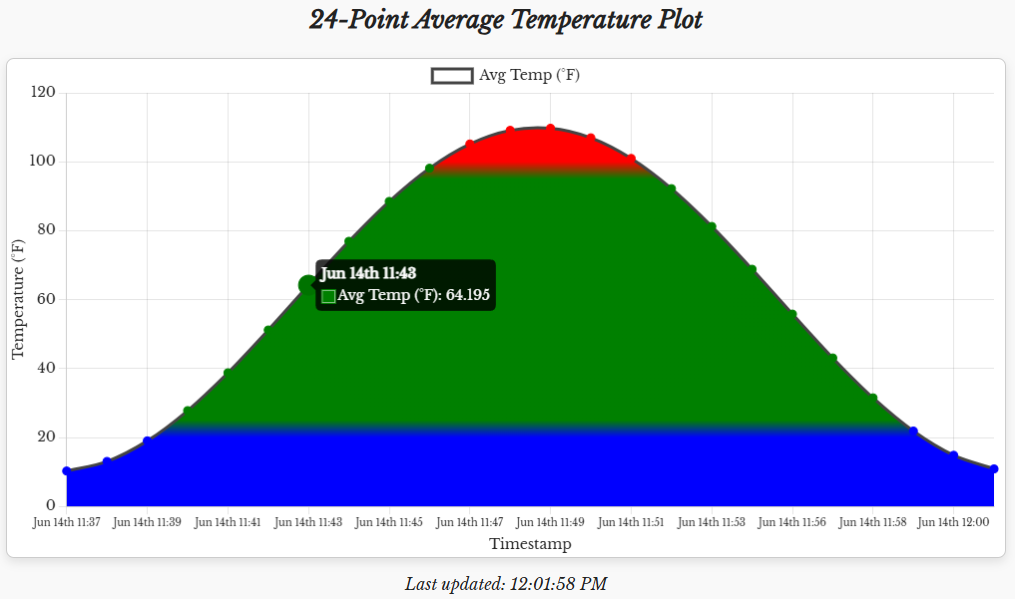
Figures 12-14: Conditions of Eastern Lux value exceeding 75,000 trigger mathematical recalculation of blinds measured in degrees. The results are passed to webhook.site URL for publish of raw JSON data.



Figures 15-17: webhook.site receiving raw JSON text from all three make scenarios with position values for the blinds.

**HTML Live Average Temperature**

A secondary Python script was loaded with the API URL for reading the most recent temperatures published to ThingSpeak. The Python script took the average temperature of all fields and dumped the average temperature to a JSON file (see attached Generate\_avgTemps\_JSON\_file.py). The JavaScript file had several functions including generating a formatted time stamp, creating and updating the chart, and loading the JSON data for use (see attached Blind\_System\_Management.js). The HTML file is brief and imports the style sheet, the Javascript, and the chart functions then displays the generated graph. (Java Scripts fetch() function is blocked when loading the HTML site directly, a local server was executed to abide by web browser security features.) The HTML was successful at loading the average temperatures over a rolling 24-point time slot and presented the data as being under, within, or over temperature (see Figure 18).

Figure 18: Plot of average temperatures with color coding for times when temperature was over, within, or under temperature thresholds.

**Results**

An IoT application with multiple functioning components has been created. An MQTT ThingSpeak device generates data successfully, and three other tools capture and transform the data for applicable uses. Blinds can now be automatically changed to varying degrees to adjust for the time of day, light conditions, and temperature. Meta-analysis allows us to track the temperatures experienced throughout the day giving us a full-scope analysis of the building temperature. Future iterations would minimize and combine Python functions to run together rather than independently and remove unnecessary data handling like webhook.site.