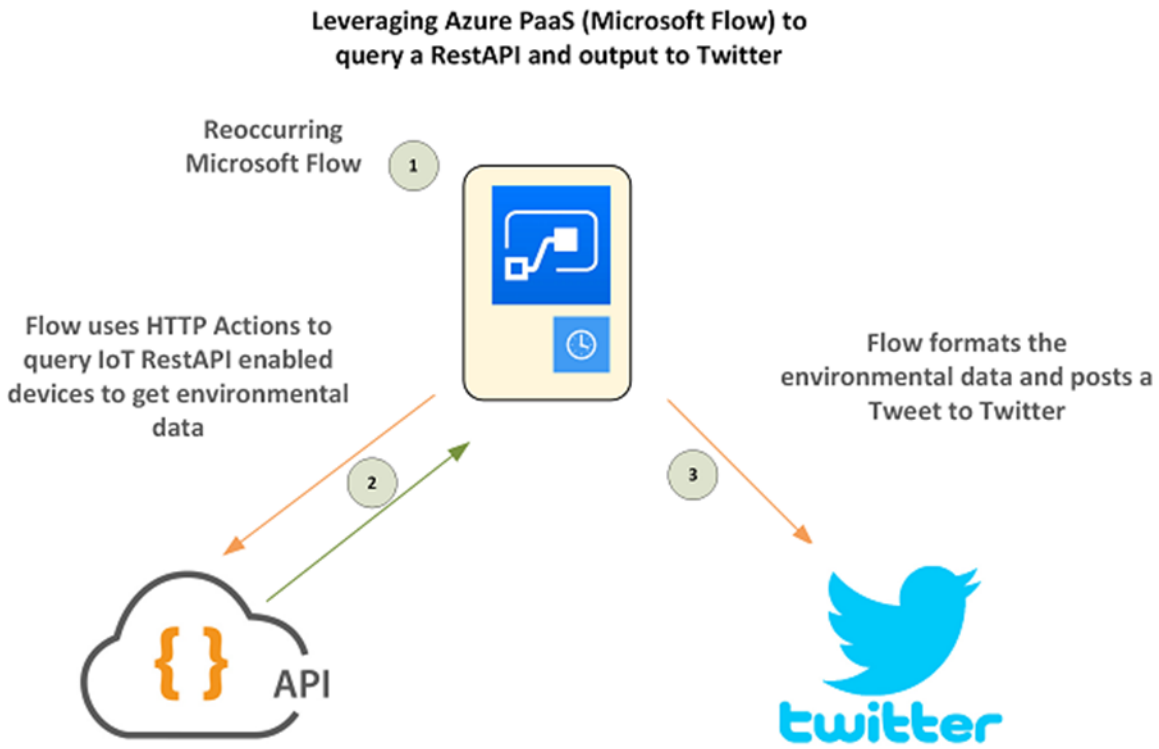
Introduction

Almost 18 months ago I wrote this post on integrating Twitter with Azure Functions to Tweet IoT data. A derivative of that solution has been successfully running for about the same period. Azure Functions have been bullet proof for me.
After recently implementing Microsoft Flow as detailed in my Teenager Notification Device post here I started looking at a number of the Azure Functions I have running and looked at what would be better suited to being implemented with Flow. What could I simplify by migrating to Microsoft Flow?
The IoT Twitter Function linked above was one the simpler Functions I had running that I've transposed and it has been running seamlessly. I chose this particular function to migrate as the functions it was performing were actions that Microsoft Flow supported. Keep in mind (see the Summary), that there isn't a one size fits all. Flow and Functions each have their place and often work even better together.



Comparison

Transposing the IoT Twitter Function App to Microsoft Flow provided me with the same outcome, however the effort to get to that outcome is considerably less. As a quick comparison I've compared the key steps I needed to perform with the Azure Function to enable the integration vs what it took to implement with Microsoft Flow.

	Azure Function	Microsoft Flow
Application Registrations	2	0
Credential sets required (in code)	3	0
Code/Script required (lines)	56	0
Third Party Modules Required	1	0

That's pretty compelling. For the Azure Function I needed to register an App with Twitter and I needed to create an Azure Function App Plan to host my Azure Function. With Microsoft Flow I just created a Flow.

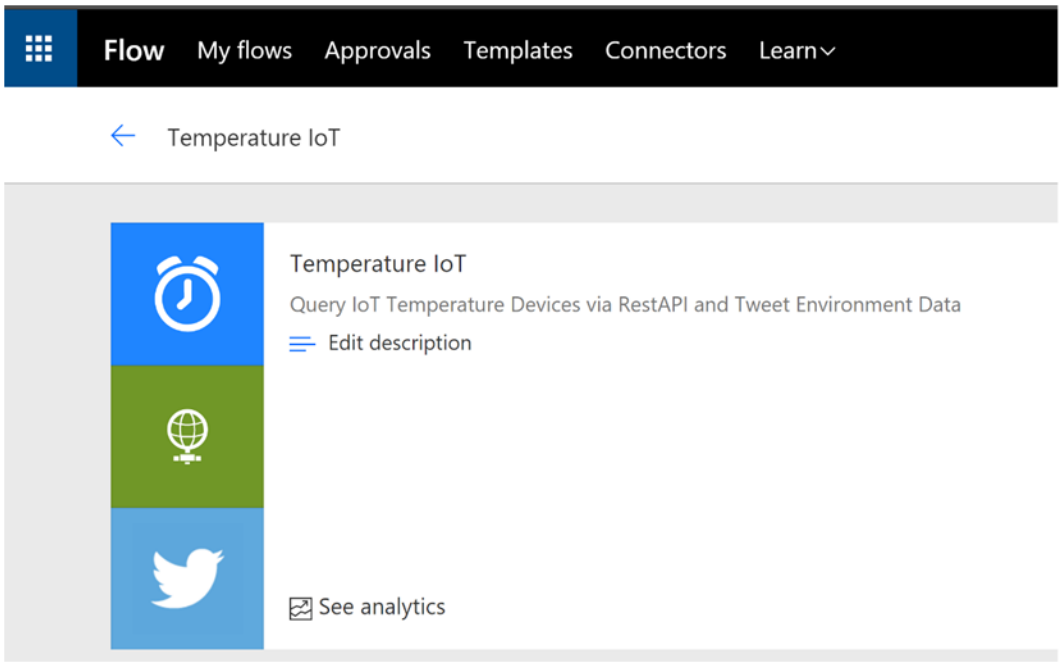
To setup and configure the Azure Function I needed to set up Deployment Options to upload the Twitter PowerShell Module (this is the third-party module), and I needed to store the two credential sets associated with the Twitter Account/App. In Microsoft Flow I just chose Twitter as an Action and provided conscent to the oAuth2 challenge.

Finally for the logic of the Azure Function I had to write the script to retrieve the data, manipulate it, and then post it to Twitter. In Microsoft Flow it was simply a case of configuring the workflow logic.

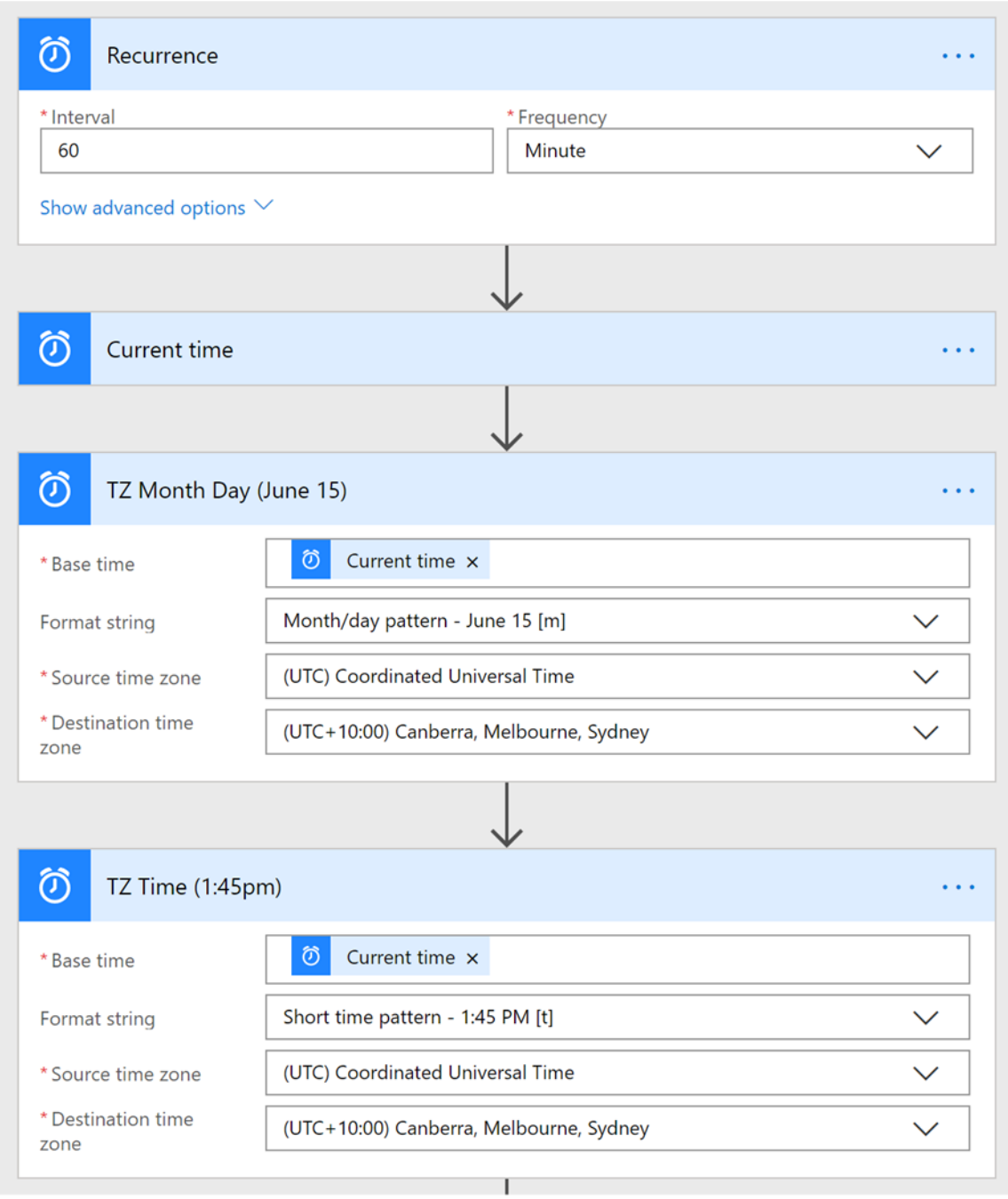
Microsoft Flow

As detailed above, the logic is still the same. On a schedule, get the data from the IoT Devices via a RestAPI, manipulate/parse the response and output a Tweet with the environment info. Doing that in Flow though means selection of an action and configuring it. No code, no modules, no keys.

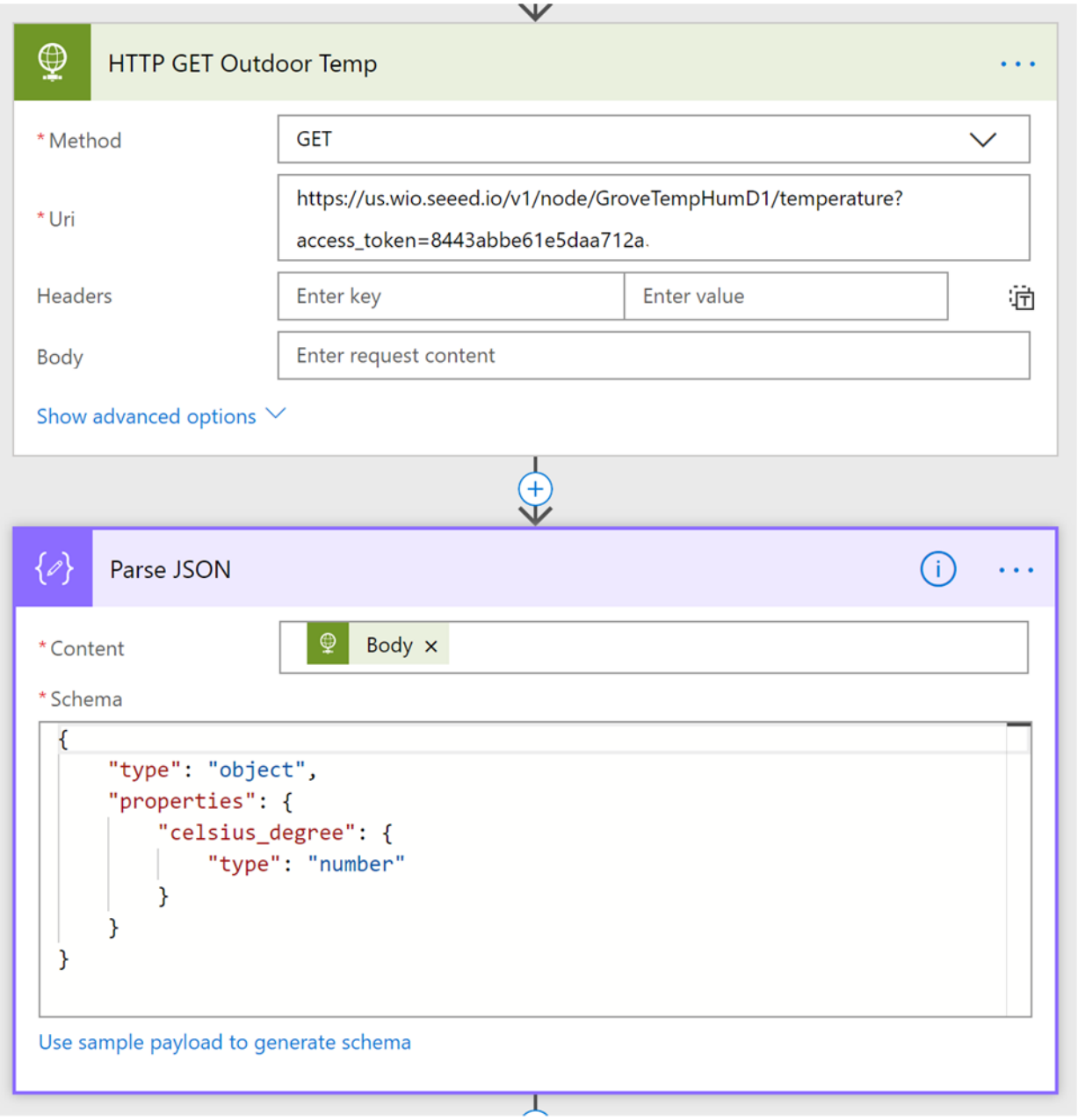
Below is a resultant Flow (overview) to achieve the same result as my Azure Function that I originally implemented as an Azure Function as detailed here.



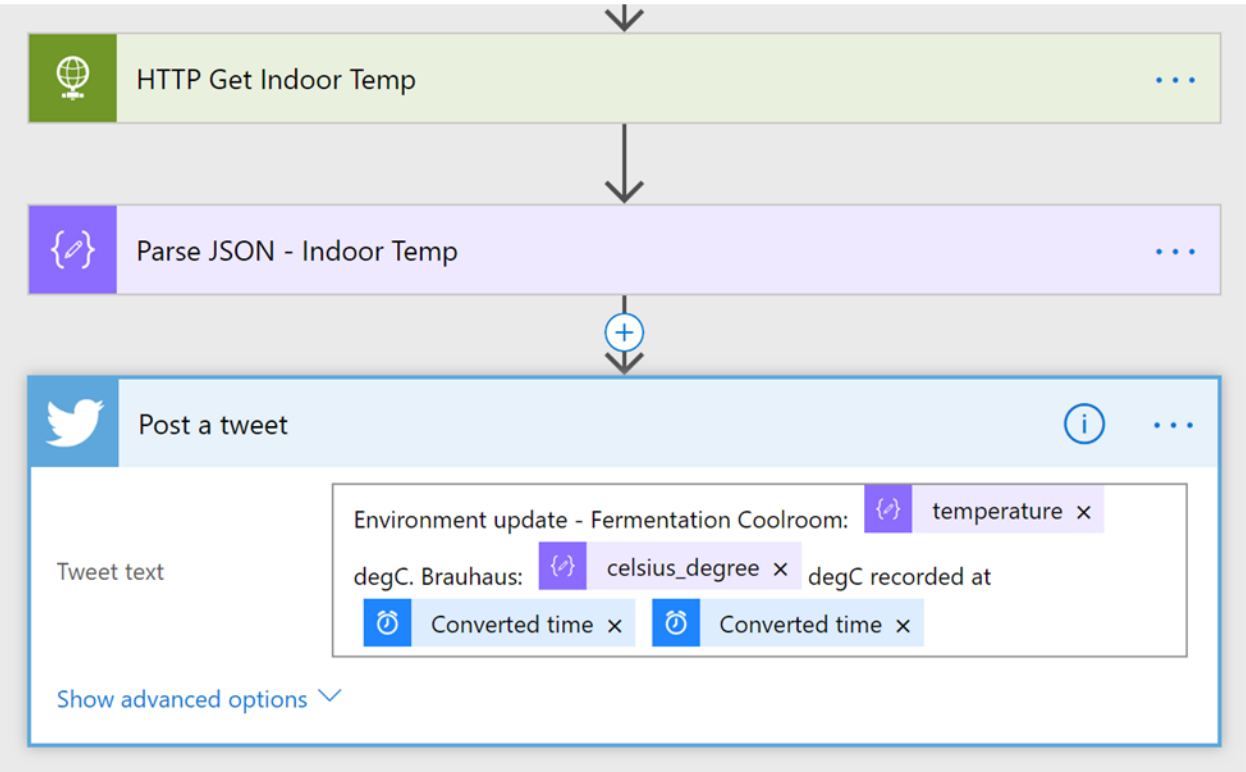
The schedule part is triggered hourly. Using Recurrence it is easy to set the schedule (much easier than a CRON format in Azure Functions) complete with timezone (within the advanced section). I then get the Current time to allow me to acquire the Date and Time in a format that I will use in the resulting tweet.



Next is to perform the first RestAPI call to get the data from the first of the IoT devices. Parse the JSON response to get the temperature value.



Repeat the above step for the other IoT Device located in a different environment and parse that. Formulate the Tweet using elements of information from the Flow.



Looking at Twitter we see a resultant Tweet from the Flow.

Summary

This is a relatively simple flow. Bare in mind I haven't included any logic to validate what is returned or perform any conditional operations during processing. But very quickly it is possible to retrieve, manipulate and output to a different medium.