As you work your way through this Step 6 tutorial, you will send data from all three sensors over the cellular network via the Notehub. Towards the end of the tutorial, your system will attempt to get a strong GPS signal. This is best achieved if you can test your system next to a large window that has a relatively clear view of the sky. You are provided a long USB cable to help position your Blues Starter Kit and connected sensors close to a window while still allowing you to work at a nearby table.

Adding the Notecard - Sending your data

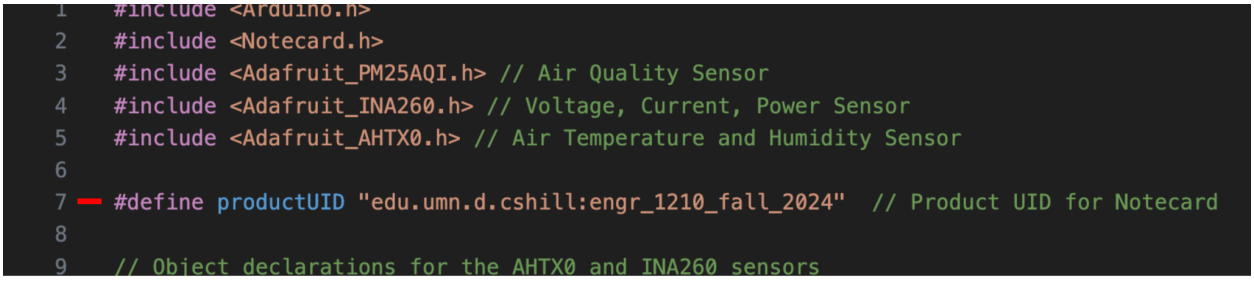
---

1. Adding the Notecard - Sending your data

a. Visit the ENGR1210_AQM_Station GitHub Repository.

b. Start by adding in the product UID line from notecard.cpp into your main.cpp project file.

i. ***NOTE:*** Do not use the same exact product UID shown in the image below. Use the product UID you set up earlier in the tutorials!



```
#include <Arduino.h>
#include <Notecard.h>
#include <Adafruit_PM25AQI.h> // Air Quality Sensor
#include <Adafruit_INA260.h> // Voltage, Current, Power Sensor
#include <Adafruit_AHTX0.h> // Air Temperature and Humidity Sensor

#define productUID "edu.umn.d.cshill:engr_1210_fall_2024"  // Product UID for Notecard

// Object declarations for the AHTX0 and INA260 sensors
```

c. Add in the Notecard object and all the variables the Notecard will set. Pay close attention to the format of each. For example, the second “char” line is for representing the month, and uses the variable “mM” (first lowercase, then uppercase “M”).

```
6
7   // Object declarations for the AHTX0 and INA260 sensors
8   Adafruit_AHTX0 aht;
9   Adafruit_INA260 ina260;
10  Adafruit_PM25AQI aqi;
11  Notecard notecard;
12
13  // Variables to store time and location data
14  char yyyy[5];
15  char mM[3];
16  char dd[3];
17  char hh[3];
18  char mm[3];
19  char ss[3];
20  double lat;
21  double lon;
22
23  // Variables to store sensor measurements
24  float temperature;
25  float humidity;
```

d. Next, you will need to insert the Notecard's initialization code into your setup function. Copy lines 32 through 65 in notecard.cpp and paste them into the bottom of the setup function just after the existing setup code. You still want to paste this inside of your setup() function, so make sure that it lands after the initialization for the air quality sensor, but before the curly-brace that "closes" the setup() function.

e. Next, add the following function prototypes for the Notecard.

```
44  uint16_t particles_100um;
45
46  // Function prototypes
47  void Read_AHTX0();
48  void Read_INA260();
49  void Read_PM25AQI();
50  void Print_Data();
51  void Notecard_Find_Location();
52  void Send_Data();
53  void Set_Time_Location(J *rsp);
54
55  void setup()
```

f. In the previous step, you just added three new function names (Notecard_Find_Location, Send_Data, and Set_Time_Location). Next, you need to add the function code. Locate the function's code in notecard.cpp, then copy and paste in these functions to the bottom of your main.cpp code file.
g. Within the loop function, add a call to the **Notecard_Find_Location()** function, followed by a call to the **Send_Data()** function. See the next image as an example.
    i. ***IMPORTANT:*** There is no need to add a call to the **Set_Time_Location()** function, this is called from within the **Notecard_Find_Location()** function.

```
120  void loop()
121  {
122    Read_AHTX0();  // Read the sensor data
123    Read_INA260();  // Read the sensor data
124    Read_PM25AQI();  // Read the sensor data
125
126    // Execute the tasks: location and data sending
127    Notecard_Find_Location();
128    Send_Data();
129
130    Print_Data();  // Print the data to Serial
131
132    delay(5000);  // Wait for 5 seconds before reading again
133  }
134
```

h. We want to send all the data we collected to Notehub. Add a line for each piece of data collected from the three sensors. The lines for data from the ***Notecard*** are already present in the code and shown below. ***It is up to you to add new lines for the other sensor data we need to send!*** You should be adding 2 lines for the temperature and humidity sensor, 3 lines for the current, voltage, and power sensor, and 12 lines from the air quality sensor (in addition to the 8 lines added in the image below for the Notecard date, time, and GPS information).
    i. ***IMPORTANT***: In each line you add, make sure the string next to the variable name matches ***exactly*** the variable name. You should not have modified any of the variable names previously. This ensures that the destination we route our data to can intake it properly.

```
261  void Send_Data()
262  {
263    // Create a Notecard request to send data (date/time and extra info)
264    J *req = notecard.newRequest("note.add");
265    if (req != NULL)
266    {
267      JAddStringToObject(req, "file", "data.qo");  // Store data in "data.qo" file
268      JAddBoolToObject(req, "sync", true);  // Request immediate sync
269      J *body = JAddObjectToObject(req, "body");
270      if (body)
271      {
272        // Send time data
273        JAddStringToObject(body, "YYYY", yyyy);
274        JAddStringToObject(body, "MM", mM);
275        JAddStringToObject(body, "DD", dd);
276        JAddStringToObject(body, "hh", hh);
277        JAddStringToObject(body, "mm", mm);
278        JAddStringToObject(body, "ss", ss);
279
280        //Send location data
281        JAddNumberToObject(body, "lat", lat);
282        JAddNumberToObject(body, "lon", lon);
283
284        Following the code above, add lines to send
285        the data collected from the three sensors
286      }
287
288      notecard.sendRequest(req);  // Send the request to the Notecard
289    }
290  }
```

These should match for every line

i. Inside the Print() function, add the code to print the data collected by the Notecard to the Serial Monitor window. See the image below. My preference is to put this first, before the sensor output lines.

```
254    Serial.print("YYYY: ");
255    Serial.println(yyyy);
256    Serial.print("MM: ");
257    Serial.println(mM);
258    Serial.print("DD: ");
259    Serial.println(dd);
260    Serial.print("hh: ");
261    Serial.println(hh);
262    Serial.print("mm: ");
263    Serial.println(mm);
264    Serial.print("ss: ");
265    Serial.println(ss);
266
267    Serial.print("latitude: ");
268    Serial.println(lat);
269    Serial.print("longitude: ");
270    Serial.println(lon);
```

j. Test your code! You will likely only obtain a GPS connection if you are outside or near a big window. Otherwise, the attempt to establish a GPS connection will timeout. Your serial output should have a field for every datapoint from the three sensors and the Notecard.

k. ***IMPORTANT!*** Once you upload the sketch to the Swan and watch the Serial Monitor, you will see frequent messages appear that indicate it is in "status": "GPS search" and then indicate a number of seconds. Look in your code, specifically in the function called Notecard_Find_Location(). You will see a line with the text "size_t timeout_s = 600; // 10-minute timeout for finding a location". This indicates that the GPS will search for 600 seconds (10 minutes) to get a high quality fix with satellites to determine an accurate location. It will search for up to 10 minutes before "timing out" and moving on with the next steps.

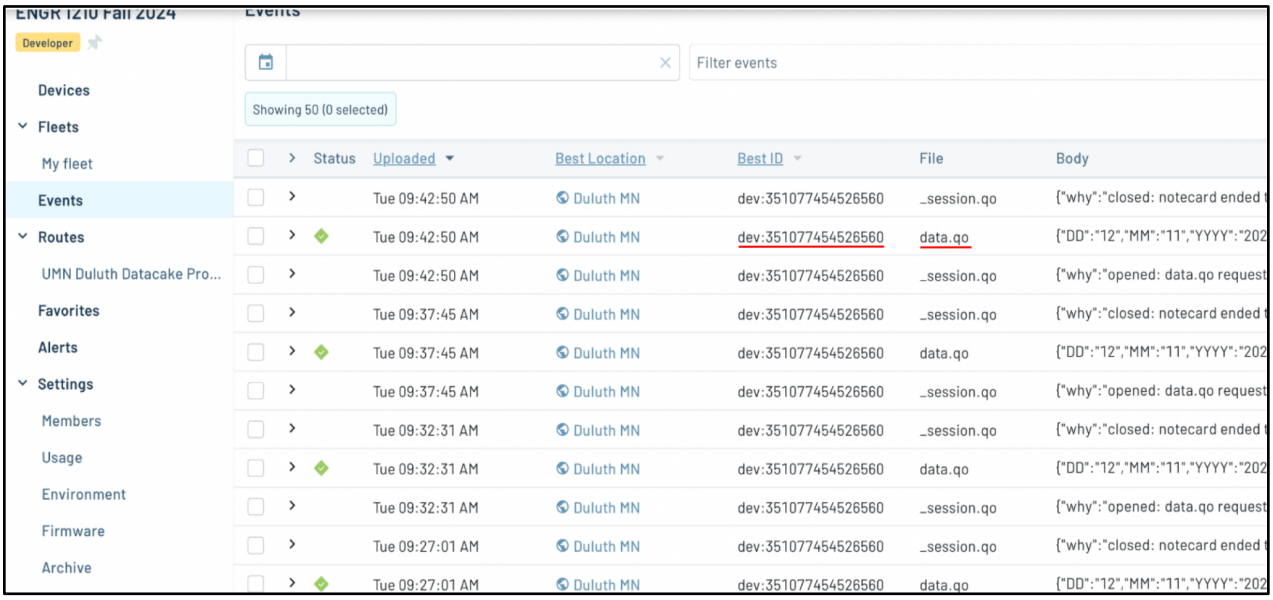
   i. Up until now, we have been fairly decent locations by triangulating between multiple cell network towers. When the stations are located outside, it should reliably get GPS coordinates and the date/time through that approach. While it is not essential for this project to have GPS location for every single message (our stations hopefully should not be moving around once deployed), it provides a deeper learning experience in case some day you want to explore developing a tracking device!

   ii. Try it once with the 10-minute interval and see if you get a successful GPS fix. If you do, excellent! Proceed with the next steps.

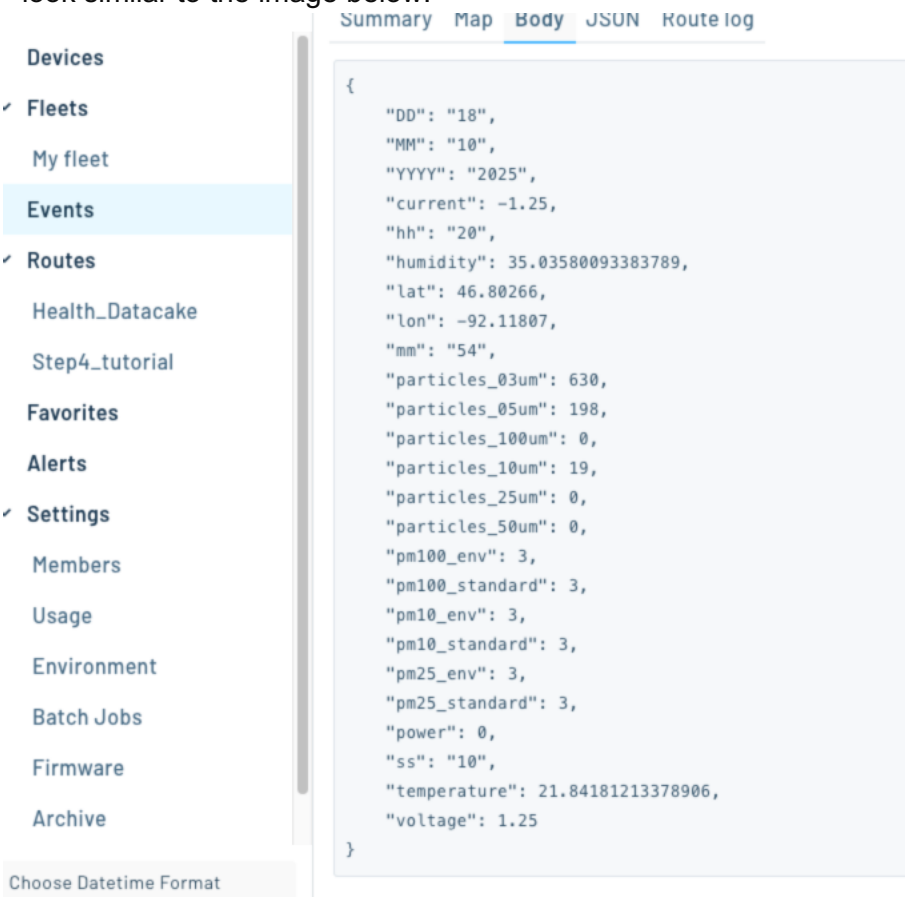
   iii. If you don't get a GPS fix, you can reduce the timeout duration to be 60 seconds. However, we need to remember to change this back prior to deploying in the field.

   iv. If you change the timeout duration, save and re-upload your sketch.

l. **Check Notehub**. You should see an event corresponding to your Notecard's Device ID with a *data.qo* file.



m. Click on the event to open the event summary. Then click on the Body tab. You should see all the data collected from the sensors and the Notecard. It should look similar to the image below.

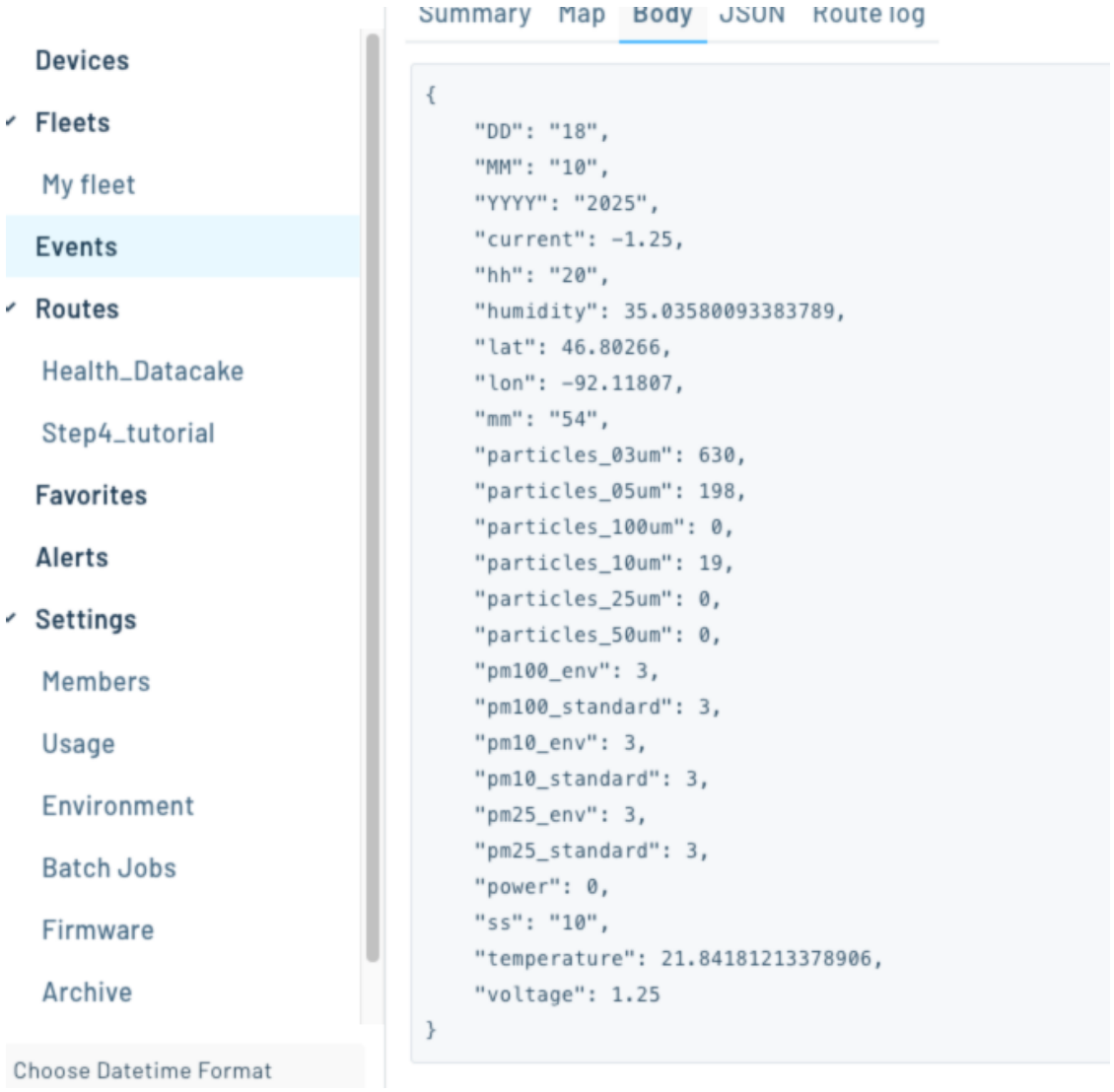


n. We want to force the main loop to take exactly 5 minutes to complete (we'll later change this to 15 minutes for field deployments). To do this, we will make the code delay until 5 minutes have elapsed. Modify your code to match what is shown in the image below. Instead of deleting the line for the delay() command, add two "//" lines before it to comment it out (should turn green, like other comment lines in the code document). The delay() line is not shown in the image below, but you can keep it in your code as long as you comment it out.
o. Save and upload your sketch file, then monitor your Events tab in Notehub.

```
120  void loop()
121  {
122    unsigned long startTime = millis();  // Record the start time
123
124    Read_AHTX0();  // Read the sensor data
125    Read_INA260();  // Read the sensor data
126    Read_PM25AQI();  // Read the sensor data
127
128    // Execute the tasks: location and data sending
129    Notecard_Find_Location();
130    Send_Data();
131
132    Print_Data();  // Print the data to Serial
133
134    // Spin the CPU until 5 minutes (300,000 ms) have passed
135    while (millis() - startTime < 300000)
136    {
137      // Busy-wait to ensure the loop runs for exactly 5 minutes
138    } // Change to 15 minutes for deployment.
139  }
```

What to Submit

- In the instructions above, you have been asked to take a screenshot from the Notehub Events tab showing the Body of your message sent. It should include all variables for the date, time, GPS location, and three sensors integrated into the system, similar to what is shown in the image below. Save this single image as a JPG to upload to the assignment on Canvas.
- Ideally you can get a successful GPS signal. If you're struggling to get a GPS fix, the Body of the message will show zeros or blank values for the lat, lon, DD, MM, YYYY, hh, mm, and ss variables. If you're unable to get a solid GPS connection indoors for this assignment, that is okay.



Congratulations! You have now finished Step 6 - Sending all sensor data to Notehub! Your group is one step closer to having a fully operating real-time IoT field station!