Use Sample Flight Data to Trigger your Microcontroller: Blue Origin New Shepard Flight Simulator

Topic: Use data to trigger a microcontroller
Suggested Grades: 6th - 12th
Summary: Use sample flight data from the TechRise flight simulator to trigger a microcontroller to light up.
Length: 30-45 minutes

Lesson Objectives
This lesson will show you how you can use the TechRise Flight Simulator data to trigger a microcontroller to do something. Whether you are a beginner or an expert in coding and microcontrollers, this lesson plan will help you learn how to use the Flight Simulator as a part of your TechRise experiment plan.

NASA TechRise Student Challenge Connection
As a part of the NASA TechRise Student Challenge, you will be sending an experiment to the edge of space! An astronaut won't be aboard the flight to perform your experiment, so you can program a microcontroller to execute your experiment for you. For example, if you want to take a picture of the curvature of earth from a high-altitude balloon, or mix two liquids when the rocket has achieved microgravity - you can use a microcontroller and flight data for that! Each flight vehicle has a computer on board that will send unique flight data to your microcontroller as it achieves key events throughout the flight. The flight data comes in a "packet" and each packet contains events and vehicle telemetry, which is basically data that tells you the physical state of the vehicle such as altitude, acceleration, etc. For instance, when the flight has achieved apogee (highest altitude in flight), your microcontroller can be programmed to recognize when apogee has been achieved. You then can program your microcontroller to perform a function at that point in time, e.g.: 'Once flight has achieved peak altitude, mix liquids together.'

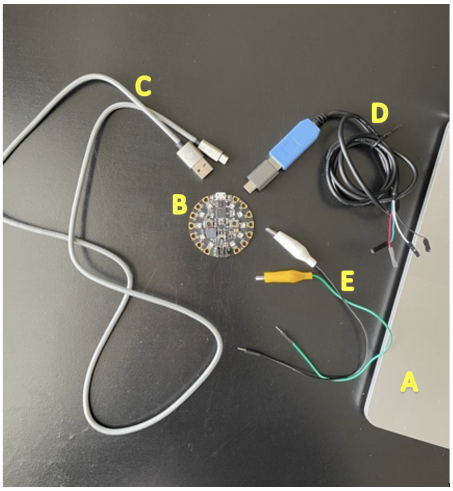
The challenge page has a TechRise Flight Simulator to help you understand the main events of each flight vehicle as well as give you access to sample flight data. The Flight Simulator stands in for the flight computer and provides a stream of sample flight data from a Blue Origin flight that can be accessed with a USB cord. This stream of data can be used in real time to trigger a microcontroller to do something, just like it would during flight. Keep in mind it is really important to understand the types of data the flight computer will be sending back to your payload so you can decide what to use as the trigger. In this lesson we will take a closer look at how this works.

On the TechRise challenge page, there is a Flight Simulator for each flight vehicle: Raven Aerostar, UP Aerospace, and Blue Origin. Take some time to look around in the simulators, notice their differences, and get comfortable with their features.

This lesson jumps right into programming the microcontroller based on Flight Simulator Data. For a deeper understanding of the Flight Simulator, and of the code used to program the microcontroller, please refer to background information at the end of the lesson.

Contents

- Getting Started: Download the Mu Editor, Circuit Playground Libraries, and FE Sim Data Libraries (Pages 2-3)
- Hardware Setup and Understanding the Flight Simulator (Pages 4-5)
- Use CircuitPython Code to Print the data from the Flight Simulator (Pages 5-7)
- Use Flight Simulator Data to Make the Neopixels Light Up (Pages 8-9)
- Troubleshooting and Background Information (Pages 9-10)



Disclaimer: This lesson uses CircuitPython code and the Circuit Playground Express microcontroller; however, you may use any microcontroller and language of your choice for your TechRise experiment.

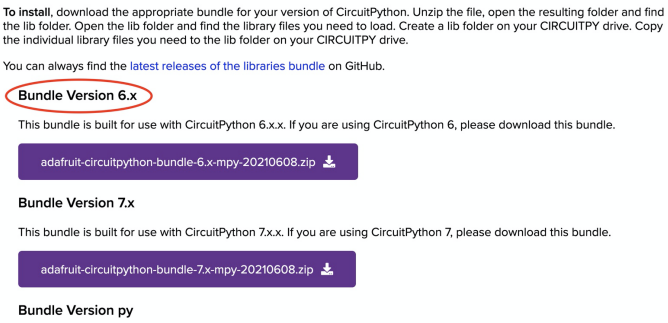
Materials
A. Computer
B. Adafruit Circuit Playground Express Microcontroller Board (linked here)
C. Micro USB Cord (like the one linked here.)
D. USB to TTL Serial Cable (Like the one linked here.)
E. (2) Alligator Clip Jumpers with Gator to Male Leads (Like the ones linked here.)
F. Blue Origin Flight Simulator
G. TechRise Microcontroller Workshop Library Files

Getting Started: Download the Mu Editor, Circuit Playground Libraries, and FE Sim Data Libraries

In this section we will download all the tools we need in order to program the Circuit Playground microcontroller to do something based on the NASA TechRise Flight Simulator Data.

1. First, we will need to make sure that the Mu Editor is downloaded. If you do not have the Mu Editor downloaded, please refer to Getting Started with Microcontrollers -Circuit Playground Express Lesson "Set up the Mu Editor" for instructions on how to get started.

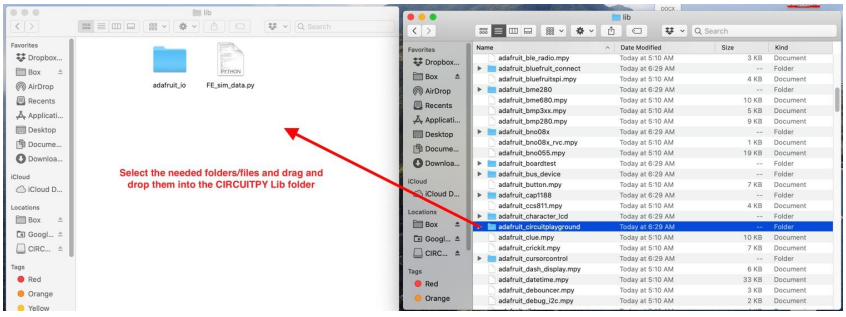
2. Next we are going to ensure we have access to the Neopixel library mentioned above. If you completed lesson 1, you already have the Neopixel library downloaded. If not, please follow the instructions. Go to https://circuitpython.org/libraries to download the CircuitPython libraries that need to be installed to your Circuit Playground Express. You will download the most recent Bundle Version (Circled in picture below). Keep in mind, the bundles are updated frequently and you may see a different version number on the CircuitPython site than the one pictured below. The top version will most always be the version to be downloaded.



3. The file will be downloaded as a Zip file. Open the file to unzip it. Once the file is open, click to open the folder called "lib". You will see many files open up.

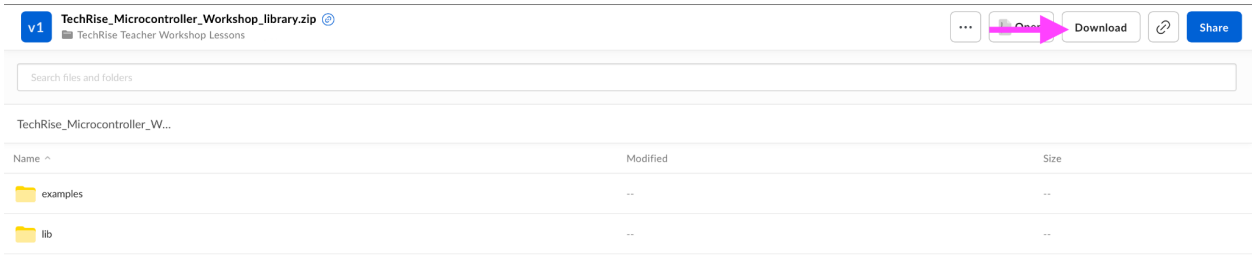
4. Now, click on your CIRCUITPY drive and click to open the folder labeled "lib". You do not need all of the files in the lib folder, you only need the ones listed below. In the downloaded Zip file under "lib", search for and select the files/folders listed below and then drag them into the CIRCUITPY folder "lib". (See photo below) Now your microcontroller will be loaded with the libraries needed for it to be programmed!

    a. Adafruit_circuitplayground [folder]
    b. Adafruit_bus_device [folder]
    c. Adafruit_lis3dh.mpy [file]
    d. Adafruit_thermistor.mpy [file]
    e. Neopixel.mpy [file]



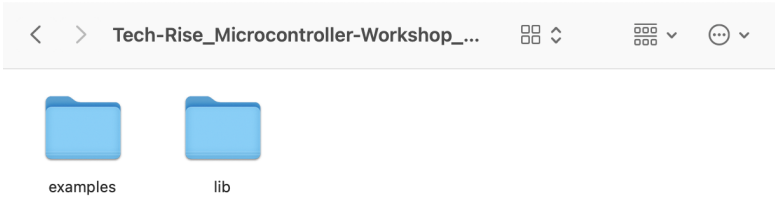
5. In addition to the Circuit Playground Libraries, we also will need to utilize a library that will allow us to stream the data from the Flight Simulator to the microcontroller. Future Engineers has provided both the FE Sim Data library and Sample Code for each Flight Vehicle. Download the TechRise Microcontroller Workshop Library Zip File here.

6. To download, click the link and then click “download”.

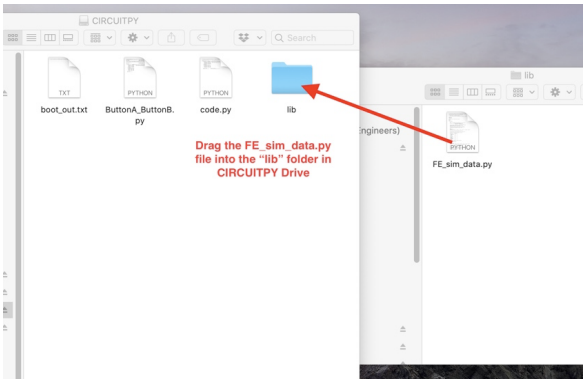


7. The folder will download as a Zip file to your computer. Once it is downloaded, click the folder to open it and unzip the file.

8. Once you have opened the folder, you will see two folders: “examples” and “lib”. First open the “lib” folder. This folder contains the FE_sim_data library that we need to copy to our CIRCUITPY drive similarly to how we downloaded libraries previously. This library will process the data coming in from the simulator.



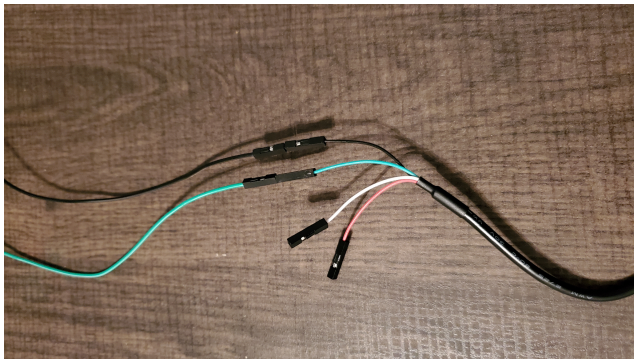
9. Once the “lib” folder is open, click to open your CIRCUITPY drive. We will now drag the FE_sim_data.py file from the “lib” folder into our “lib” folder in our CIRCUITPY drive (see picture below). Now we are ready to import this library and use it in our code!



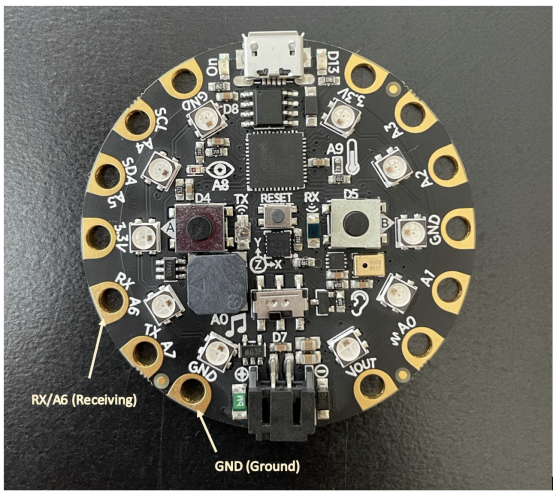
Hardware Setup and Understanding the Flight Simulator

1. Now let's set up our microcontroller to get it ready to receive data from the Flight Simulator. First, ensure that your micro USB cord is plugged into the USB port onto your computer and plugged into the microcontroller.

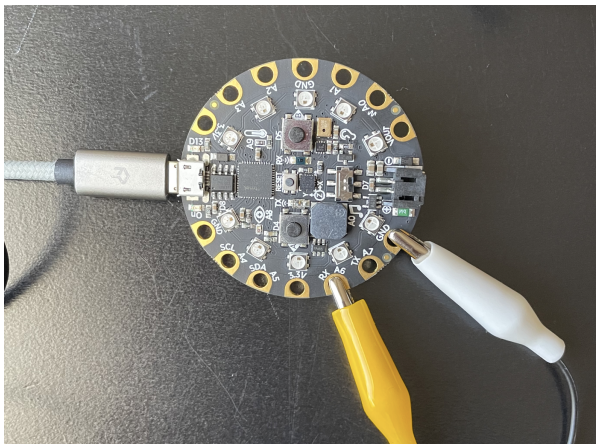
2. Next, plug your USB to TTL serial cord into your computer in a second USB port. The USB TTL serial cord will stream the data from the Flight Simulator to the microcontroller. You will connect the male side of one alligator clip to the black cord coming out of the USB/TTL cord. Then, connect a second alligator clip to the green cord coming out of the USB/TTL cord. The white and red cords will remain unused. *Keep in mind that you may use any color of alligator wire, but for the purpose of keeping things organized, we use a green to connect to the green TTL cord, and a black to connect to the black TTL cord. See picture:



3. Next we will be connecting our alligator wires to the RX and GND pins seen in the photo below:



4. Connect the TX (Transmitting) wire (green wire) to the RX (Receiving- labeled RX A6) pad on the Circuit Playground microcontroller. Then connect the Ground wire (black wire) to the GND pad on the Circuit Playground. The alligator clips can clip right onto the pads on the microcontroller as seen in the picture below:



5. Now, let's make sure our Mu Editor knows that it needs to program our Circuit Playground Microcontroller. Because we have two USB cords connected, sometimes Mu will think they are both a microcontroller. Make sure that you select "CircuitPython Board" as the microcontroller in the bottom right hand corner of the Mu Editor (see pic below). Hit "Save" in the Mu Editor.

```
            print("YAY! CRUISE ALITITUDE ACHIEVED, COLLECTING DATA!")
            pixels.fill(BLUE)
        elif event == 'Ascend':
            print("YAY! LIFTOFF ACHEIVED")
            pixels.fill(ORANGE)
        elif event == 'Pre-Launch':
            print("YELLOW ALERT, PREPARING FOR LAUNCH")
            pixels.fill(YELLOW)
        elif event == 'Initialize':
            print("SYSTEM INITIALIZED")
            pixels.fill(RED)
        else:
            pixels.fill(0)
else:
    pixels.fill(GREEN)
```

CP210x
CircuitPython board

6. Next, open up the Blue Origin Flight Simulator using **Google Chrome** for the best functionality. https://www.futureengineers.org/simulator/blue-origin/new-shepard/

7. Scroll to the bottom and you will see a button that has a picture of a USB icon, a “Play”, and a “Rewind” button. Click the USB button, and choose “CP2102 USB to UART Bridge controller”. Once connected, the USB icon should light up in green as seen in the picture below. This is connecting the flight simulator data to our USB to TTL cord. Now that it is green, once we write some code to stream the data and do something with it, that data will travel from the flight simulator through the USB to TTL cord to our microcontroller.

8. Next, take some time to become familiar with Blue Origin flight simulator and its features. Press play to see the vehicle move through the different stages of flight. You can speed it up or slow it down using the dropdown button at the bottom of the screen.

You will notice that the simulator highlights certain key events that the vehicle achieves within the flight. The key events that are highlighted in the Blue Origin Flight Simulator are:

a. Payload Power On (experiments are ready for flight)
b. Liftoff (vehicle has launched)
c. MECO (Main Engine Cut Off)
d. Capsule Separation (capsule separates from the rocket.)
e. Coast Start (microgravity begins)
f. Apogee (vehicle reaches highest altitude in flight)
g. Coast End (microgravity ends)
h. Drogue Chutes (first set of parachutes deploy)
i. Main Chutes Deployed (main set of parachutes deploy)
j. Touchdown (vehicle lands back on Earth)

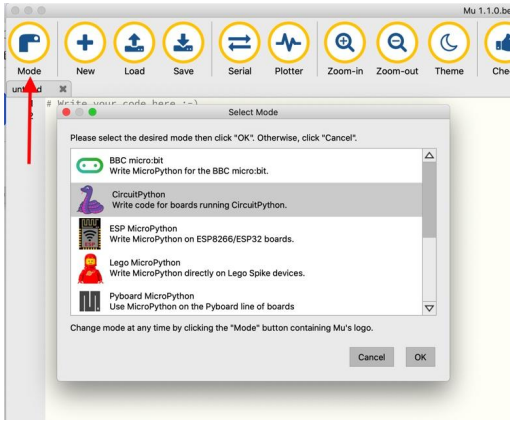
Detailed Tech Sheets of each flight provider can be found on the TechRise Challenge Page (https://www.futureengineers.org/nasatechrise).

Use CircuitPython Code to Print the Data from the Flight Simulator

We know that the Flight Simulator gives us sample flight data, but we do not yet know exactly what that data looks like. We are going to write a program that streams the sample flight data to our microcontroller, and then we are going to have the microcontroller print that data so we can see it.

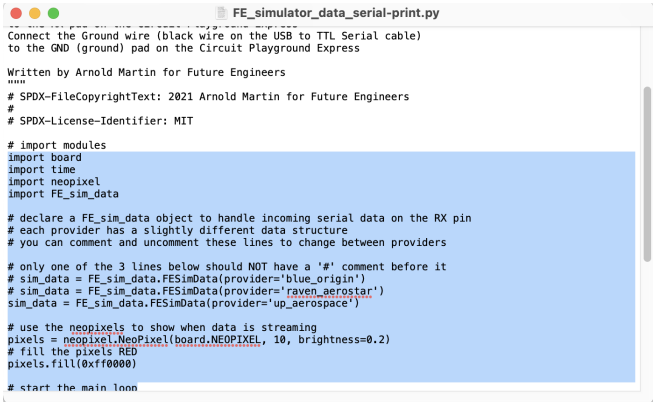
1. Future Engineers has provided the code needed to print the data from the Flight Simulator. To obtain the code, go back to the downloaded *“FE_sim_data_examples”* folder you downloaded to your computer. Click to open the *“FE_simulator_data_serial-print.py”* file and it will open in a text editor on your computer. You will see an entire coding program written out that we will soon copy and paste into our Mu Editor.

2. Next open up your Mu Editor. Click “Mode” and then click “CircuitPython”. This sets the Mu Editor up to write code in the CircuitPython which is the coding language used to program our Circuit Playground board.



3. Next, click the “Serial” button at the top of the page. A panel will open up at the bottom of the Mu Editor which will serve as our serial monitor. The serial monitor is essentially like a computer screen. Our microcontroller does not have a screen to show us what it’s doing when we perform certain functions, and so the serial monitor operates like it’s screen. Once we write our code to print the flight data to our serial monitor, we will see the data streaming in this window. For more information on the Serial Monitor and other functions of the Mu Editor, please refer to the Getting Started with Microcontrollers Lesson-Circuit Playground Express.

4. Go back to the text editor, and select the code beginning from “import board.” Copy and paste the code into your Mu Editor starting at line 1.



5. Once pasted into the Mu Editor, you will notice that any line of code preceded by “ # “ is grayed out and will not be processed by the microcontroller. The grayed out lines are showing comments made in the code for learning/instructional purposes.

6. Note that this sample code can be used for each of the Flight Simulators by changing the parameters in the “*sim_data =*” line of code. To change the parameters, simply remove the “ # “ from the line that you are using. Since we are using the Blue Origin Flight Simulator, remove the “#” from line 11. The microcontroller can only utilize data from one flight simulator at a time, so make sure that the two other simulators in lines 12 and 13 are grayed out. See photo below.

code.py

```
import board
import time
import neopixel
import FE_sim_data

# declare a FE_sim_data object to handle incoming serial data on the RX pin
# each provider has a slightly different data structure
# you can comment and uncomment these lines to change between providers

# only one of the 3 lines below should NOT have a '#' comment before it
sim_data = FE_sim_data.FESimData(provider='blue_origin')
# sim_data = FE_sim_data.FESimData(provider='raven_aerostar')
# sim_data = FE_sim_data.FESimData(provider='up_aerospace')

# use the neopixels to show when data is streaming
pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=0.2)
# fill the pixels RED
pixels.fill(0xff0000)

# start the main loop
while True:
    # at the top of the loop, run the update() method for the FE_sim_data object
    sim_data.update()

    # check if the data is currently streaming...
    if sim_data.streaming:

        # fill the pixels green to indicate the data is streaming
        pixels.fill(0x00ff00)

        # check to see if new data is available...
        if sim_data.new_data:
            # print the new data to the serial monitor (over usb)
            print(sim_data.data)
            # print the current event to the serial monitor (over usb)
            print(sim_data.event)

    else:
        # since the data is not streaming fill the pixels RED
        pixels.fill(0xff0000)

```

7. Hit the “Save” Button in the Mu Editor. Your microcontroller should light up in Red! If yours is not lit up, please double check the following:

    a. Make sure the alligator clips and all the cords are connected correctly.
    b. Make sure the Blue Origin Flight Simulator is open in the web browser, and that the USB icon on the page is green.
    c. In the bottom right hand corner of the Mu Editor, make sure CircuitPython board is selected:

8. Next, go to the Blue Origin Flight Simulator and Press the Play button, and then click your Mu Editor and look down into the Serial Monitor panel to see the data streaming in from the flight simulator!

Now that we have learned how to send and receive data to our microcontroller using Circuit Python and the Mu Editor, we can add in the flight simulator. The flight simulator will provide your microcontroller with a stream of sample flight data similar to what a winning TechRise experiment will get from its flight vehicle (balloon or rocket) computer. In the next section, we will program a microcontroller to take in the sample flight data from the simulator and use it to trigger an event (ex. turning on lights) at a specific point within the data (ex. lift off, apogee or landing).

Use Flight Simulator Data to Make the Neopixels Light Up

1. Keep all of the hardware set up as is. Go back to the Mu Editor, select all of the code and erase it for now.

2. Next, go back to the downloaded *"FE_sim_data_examples"* folder you downloaded to your computer. We will now click to open the "examples" folder. You can use any of the examples given, but for this lesson plan's purposes, let's use Blue Origin as our example. Click to open the *"MC_BLUEData_demo"* file and it will open in a text editor on your computer, just as the previous code file did.

3. The text includes some simple instructions at the beginning. For now, we want to highlight the part starting from where it says "import board"(see picture below to see where you should start highlighting). Highlight from "import board" all the way to the end and hit Copy, and then Paste it into your Mu Editor.

```
import board
import busio
import time
import neopixel
import FE_sim_data

# setup FE_sim_data object for blue origin rocket flight
sim_data = FE_sim_data.FESimData(provider='blue_origin')

# declare neopixels to visulize events
pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=0.2)

# color constants for pixel colors
RED = 0xff0000
GREEN = 0x00ff00
BLUE = 0x0000ff
CYAN = 0x00ffff
MAGENTA = 0xff00ff
YELLOW = 0xffff00
ORANGE = 0xff8000
WHITE = 0xffffff

pixels.fill(RED)

event = None
pre_event = None

event_list = ['@', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M']

while True:
    # read the serial data from FE_sim_data object
    sim_data.update()

    if sim_data.streaming:
        if sim_data.new_data:
            print(sim_data.data)
            print(sim_data.event)
            # events arrive as a single character
            event = sim_data.event

        # if statement to check for change in the event data
        if event != pre_event:
            # check that it is a good event and not bad DATA
            if event in event_list:
                # reset the previous event
                pre_event = event

            # if...elif responding to blue_origin event data
            if event == 'J':
                print("Main chute deployed, floating safely to Earth!")
                pixels.fill(CYAN)
            elif event == 'I':
                print("Drogue chute deployed")
                pixels.fill(ORANGE)
            elif event == 'H':
                print("End Coast")
                pixels.fill(YELLOW)
            elif event == 'G':
                print("APOGEE ACHIEVED!")
                pixels.fill(WHITE)
            elif event == 'F':
                print("Begin Coast")
                pixels.fill(BLUE)
            elif event == 'E':
                print("Separation Commanded")
                pixels.fill(YELLOW)
            elif event == 'D':
                print("Main Engine Cutoff")
                pixels.fill(ORANGE)
            elif event == 'C':
                print("Engines are burning, we have liftoff!")
                pixels.fill(RED)
            else:
                pixels.fill(0)
    else:
        pixels.fill(GREEN)
```

Let's take a moment to look at this written program to see what the code is telling the microcontroller to do. This code streams data from the flight simulator and then tells the microcontroller to do something when the data is at a specific event. Take note of the "if statements." These statements are basically telling the microcontroller, "If the flight vehicle has hit a certain key event within flight (e.g.: "Drogue chute deployed!"), make the neopixels light up with a certain color, AND print a statement describing the event." You may also notice the "else" statement at the bottom. This is telling the microcontroller, "if the flight vehicle is at some other event within flight, light the neopixels up with green color."

You may also notice some lines of code that begin with "# ". These are simply comments that are telling us what those lines of code are doing. Any statement that begins with "# " will not be processed by the microcontroller and is for learning purposes only.

4. In the Mu Editor, click "Save" and make sure to save the code as 'code.py' - it will ask you if you would like to replace the code.py file, and you will click "Yes" to replace it. If everything is working correctly, the neopixels on the microcontroller will light up in green.

5. Click back over to the Blue Origin Flight Simulator (https://www.futureengineers.org/simulator/blue-origin/new-shepard/). Ensure that the USB icon at the bottom of the page is lit up green. Press Play and watch your microcontroller light up in different colors as the vehicle goes through the different events of flight!

Troubleshooting

1. If after saving the file to code.py you find it is still not running, check the following:
    a. Did you exit out of the REPL? Click into the bottom panel and hit Control-C after a ">>>" to ensure that the REPL has stopped running any previous code you wrote there. The main window of code will not run if anything is still running within the REPL. Once you hit Control-D, you should see your code start running. If the REPL is no longer open, simply click "Serial", click into it, and follow the same instructions.

2. Ensure the simulator is connected to your USB TTL cord. You will know it is connected if the USB icon is lit up green. If yours is connected and still not working, try unclicking.

3. Ensure the alligator clips and cords are all connected correctly.

4. Ensure "CircuitPython Board" is selected in the bottom right hand corner of the Mu Editor.

5. Did you exit out of the REPL? Click into the bottom panel and hit Control-C after a ">>>" to ensure that the REPL has stopped running any previous code you wrote there. The main window of code will not run if anything is still running within the REPL. Once you hit Control-C, and then hit Save, you should see your code start running. If your code does not start running, hit Control-D to reload and it should start after that.

6. Click the "Check" button at the top of the Mu Editor - it will show you if you have any Syntax Errors (too many spaces, no indent, etc.)

    a. If you do have syntax errors, check the following: Did you include all the correct brackets, quotation marks and indents?

7. If you are still experiencing errors, please check CircuitPython's troubleshooting link: https://circuitpython.readthedocs.io/en/latest/docs/troubleshooting.html , or reach out to us via email at support@futureengineers.org

Background Information

Flight Simulator

The TechRise flight data simulator is a web-based tool to help you learn about the NASA TechRise Student Challenge vehicles. It shows the main events of the flight vehicle and streams sample vehicle telemetry data through a USB. To use it you can connect your experiment's microcontroller to your PC, Mac or Chromebook using a USB. In this lesson the Blue simulator the your microcontroller receives data such as flight state, altitude, pressure, orientation - the similar to the data it will receive in flight. You can use this data to cue motors, read from sensors, take photos at a specific point in flight, and do whatever your payload requires.

The chart below shows an explanation of CircuitPython commands that are used within the code provided by Future Engineers. These commands handle the incoming data streaming from the flight simulator.

CircuitPython Command	What it does	Data it produces
Sim_data.update	Checks for new data at top of loop repeatedly	Nothing
Sim_data.streaming	Checks if data has been received within the last second	"True" if yes, "False" if no
Sim_data.new_data	Checks if new data is available	"True" if yes, "False" if no
Sim_data.data	Reads data	Returns entire data string in a list type [list, of, values, in, a, list, type]
Sim_data.event	Returns the last event received for the current flight provider	Blue Origin: "A, B, C, D, etc." UP: "LIFTOFF, BOOSTER BURNOUT, etc." Raven: None

During this lesson we experienced how to stream and utilize sample flight data for a TechRise Experiment. A microcontroller was programmed to light up in different colors at specific events in the data. Maybe you want to record the temperature at the highest point in flight (apogee), or mix liquids together when microgravity starts (coast start). The code provided in this lesson can be altered to do just that!

Vocabulary List

- TechRise Flight Simulator
- Flight Data
- Vehicle Telemetry
- Data Trigger
- Data Packet
- Flight Events
- MECO
- Coast Start
- Coast End
- Apogee
- Drogue Chutes
- Main Chutes
- Touchdown

For more information and resources for the NASA TechRise Challenge, visit https://www.futureengineers.org/nasatechrise

For support, please reach out to us at support@futureengineers.org