CURIE Academy, Summer 2021
Lab 2 Handout: Computer Engineering – Software Perspective

Prof. Christopher Batten
School of Electrical and Computer Engineering
Cornell University

In this lab assignment, you will work in a group of three scholars to explore the field of computer engineering from the software perspective by incrementally programming a microcontroller in C++ to implement an IoT "smart light" system. Feel free to consult the lab notes as you work through the lab. In Part 1, you will compile and analyze the machine instructions for a simple Particle Argon C++ program. In Part 2, you will experiment with a button input module and LED output module connected to your Particle Argon. In Part 3, you will experiment with the Particle cloud. Finally in Part 4, you will build the complete IoT "smart light" system. After completing all four parts, all of the scholars will come together to create an IoT geolocation system so we can see where all of the scholars are participating from. For each part and subparts you will need to have an instructor observe the desired milestone and initial the appropriate box on the shared Google sign-off spreadsheet. Here is a list of each milestone:

- Part 1: Understanding the Connection Between Applications and Machine Instructions
  - Part 1.A: Test Simple Addition Program
  - Part 1.B: Examine Corresponding Machine Instructions
- Part 2: Understanding Input/Output Modules
  - Part 2.A: Experiment with LED Output Module
  - Part 2.B: Develop a Button-Controlled LED System
- Part 3: Understanding the IoT Cloud
  - Part 3.A: Experiment with Particle Variables
  - Part 3.B: Experiment with Sending Particle Events
  - Part 3.C: Experiment with Receiving Particle Events
- Part 4: Building an IoT System
  - Part 4.A: Develop a "Smart Light" System
  - Part 4.B: Share Photo or Video of Full-Adder on Slack

Before beginning, remove the following materials from your electronics prototyping kit and put all of the remaining items aside (see Figure 1).

- Particle Argon
- Button input module
- LED output module
- Two short module connector cables
- White USB cable

1. Understanding the Connection Between Applications and Machine Instructions

In this part, we will write our very first Particle Argon C++ program and examine the corresponding machine instructions. Our initial program will add two numbers together and then display the output by blinking an LED. **Revisit the lab notes for more information about the Particle Argon and the four sections of a Particle Argon C++ program.**

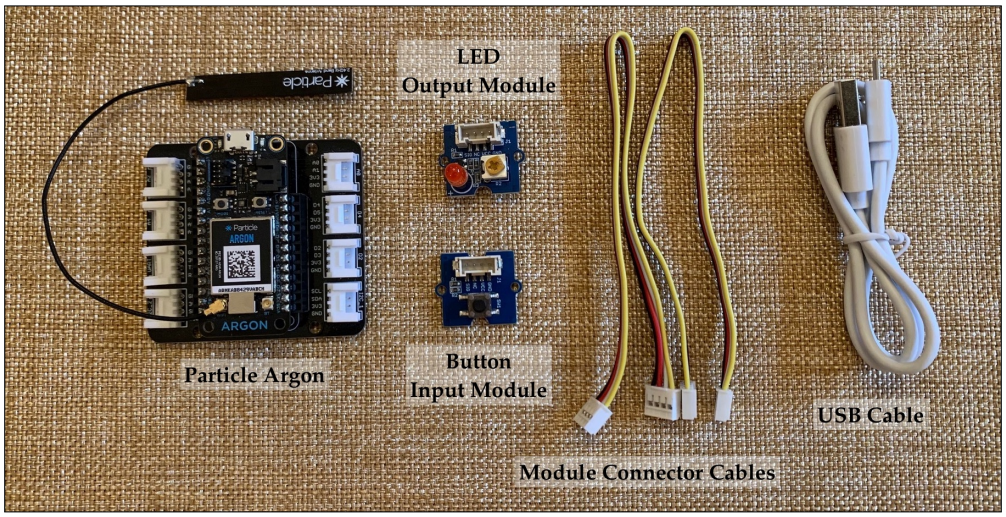

Figure 1: Materials Required for Lab 2

Complete the following steps before continuing:

- Ensure your Particle Argon is powered through the USB cable. Note you do not need to plug your Particle Argon into a laptop or workstation. The USB cable is only used for power, so you can use a USB power adapter.
- Ensure your Particle Argon status LED is breathing cyan.
- Log into the Particle development environment at `https://build.particle.io` using your group name and password provided by the instructors.
- Ensure your Particle Argon is selected by first clicking the *devices* icon and then clicking the yellow star next to the name of your device. Confirm that the name of your Particle Argon device is listed in the bottom right-hand corner of the Particle development environment.
- Try signaling your Particle Argon through the Particle development environment to ensure it is working and connected to the cloud.

1.A Test Simple Addition Program

Recall the program in Figure 2 from the lab notes. Recall that the `digitalWrite` routine will write a digital output pin with either a logic HIGH or a logic LOW. The `delay` routine will essentially wait for the given time specified in milliseconds. Notice that we are using a `for` loop on lines 34–39 to blink the LED several times; the number of times we blink the LED is equal to the sum of the two input numbers.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Enter the code from Figure 2. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon. Once the code has been uploaded, confirm that the blue LED is blinking the desired number of times. Try changing the values of `x` and `y` and recompile and reupload the program. Confirm the blue LED is blinking the desired number of times.

```
1  // Global constants for pin assignments and global variables
2
3  int led_pin = D7;
4
5  int x = 2;
6  int y = 3;
7  int z = 0;
8
9  // Helper functions
10
11 int add( int a, int b )
12 {
13   int sum;
14   sum = a + b;
15   return sum;
16 }
17
18 // The setup routine runs once when you press reset
19
20 void setup()
21 {
22   // Configure led_pin as digital output
23   pinMode( led_pin, OUTPUT );
24 }
25
26 // The loop routine runs over and over again
27
28 void loop()
29 {
30   // Do the addition
31   z = add( x, y );
32
33   // Blink LED z times
34   for ( int i = 0; i < z; i++ ) {
35     digitalWrite( led_pin, HIGH );  // Turn on the LED
36     delay(500);                     // Wait 0.5 seconds
37     digitalWrite( led_pin, LOW );   // Turn off the LED
38     delay(500);                     // Wait 0.5 seconds
39   }
40
41   // Wait four seconds
42   delay(4000);
43 }
```

Figure 2: Complete Example C++ Program

Sign-Off Milestone: Once you have experimented with your program, demonstrate for an instructor that the program can successfully add two new input values. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

Critical Thinking Questions to Discuss Within Your Group: What would happen to the LED if we replaced line 31 with `z = add( x, z )`? Explain why this would happen.

1.B Examine Corresponding Machine Instructions

In this subpart, we wish to examine the machine instructions that correspond to the `add` function in the high-level program language on lines 11–16 in Figure 2. To see these machine instructions, we will use an online tool called *Compiler Explorer* at this URL (see Figure 3):

- `https://godbolt.org`

Find where the drop-down box that says *x86-64 gcc 11.1* and choose *ARM gcc trunk (linux)*. This makes sure the compiler is generating the same kind of machine instructions used by the Particle Argon. Enter just the `add` function from lines 11-16 in Figure 2 into the left-hand box of Compiler Explorer. The compiler will generate the machine instructions in the right-hand box for you to examine. You can force the compiler to do more optimizations by entering in `-O3` into the *Compiler options* text box. Experiment with other code. For example, replace the + operator with - to do subtraction, * to do multiplication, or / to do division. Examine the machine instructions generated by the compiler.

Sign-Off Milestone: An instructor will choose one scholar in the group to share her screen and show the machine instructions corresponding to her new function which uses either the subtract (-), multiply (*), or divide (/) operator.

Critical Thinking Questions to Discuss Within Your Group: The `add` instruction might very likely be implemented with a ripple-carry adder similar to what you built in Lab 1. How many full adders do we need in the ripple-carry adder if we can guarantee that the inputs are in the range 0–255?

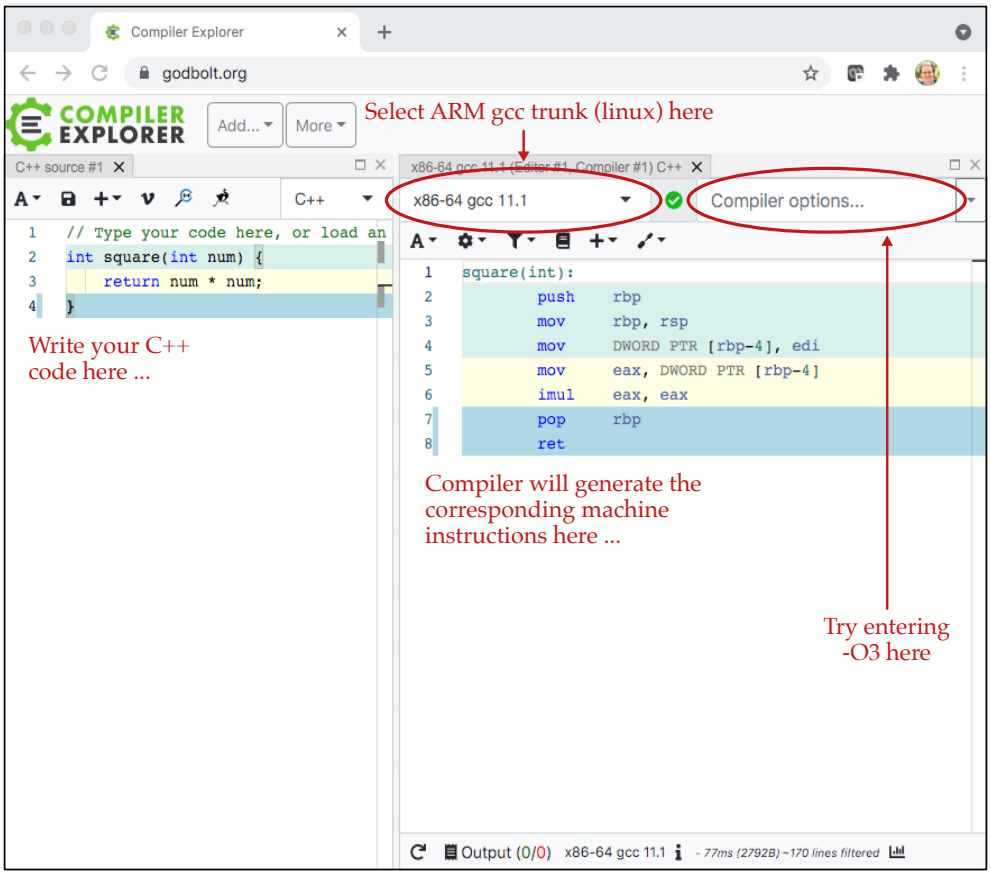

Figure 3: Compiler Explorer

2. Understanding Input/Output Modules

Now that we better understand the connection between applications and machine instructions, let's start to explore the various input and output modules. We will begin by blinking the LED output module, before adding the button input module to control the LED output module.

2.A Experiment with LED Output Module

Connect one end of a short module connector cable to the LED output module, and connect the other end to digital port D4 on the Particle Argon. Revisit the lab notes for more on the Particle Argon ports. Once we have made these connections we can now control the LED output module in the exact same way we controlled the blue LED on digital pin D7 in Figure 2. We simply need to specify a pin number D4.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Enter the code from Figure 4. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon. Once the code has been uploaded, confirm that the red LED is blinking. Try changing the delay values to make the red LED blink slower or faster.

Sign-Off Milestone: Show an instructor that the red LED is blinking. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

Critical Thinking Questions to Discuss Within Your Group: What would happen if we change the delay on line 11 to a very small value (e.g., 100)? What would happen if we instead change the delay on line 13 to a very small value (e.g., 100)?

```
1  int led_pin = D4;
2
3  void setup()
4  {
5    pinMode( led_pin, OUTPUT );
6  }
7
8  void loop()
9  {
10   digitalWrite( led_pin, HIGH );  // Turn on the LED
11   delay(1000);                    // Wait 1 second
12   digitalWrite( led_pin, LOW );   // Turn off the LED
13   delay(1000);                    // Wait 1 second
14 }
```

Figure 4: Code for LED Output Module

2.B Develop a Button-Controlled LED System

Connect one end of a short module connector cable to the button input module, and connect the other end to digital port D2 on the Particle Argon. Revisit the lab notes for more on the Particle Argon ports. Once we have made these connections we can now control the LED output module using the button input module.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Start with the template code from Figure 5 and fill in the code for the `loop` function. You will need to use a conditional statement. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon. Once the code has been uploaded, confirm that the red LED turns on when the button is pressed. *Hint: Take a look at Figure 9(a) in the lab notes for an example of a C++ conditional statement.*

Sign-Off Milestone: Show an instructor that the red LED turns on when the button is pressed. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

```
1  int led_pin      = D4;
2  int button_pin   = D2;
3  int button_state = -1;
4
5  void setup()
6  {
7    pinMode( led_pin,    OUTPUT );
8    pinMode( button_pin, INPUT  );
9  }
10
11 void loop()
12 {
13   // read the button value
14   button_state = digitalRead( button_pin );
15
16   // Add code here to check the button value ...
17   // ... if button value is one, turn on LED.
18   // ... if button value is zero, turn off LED.
19
20   // wait 1 second before reading button again
21   delay(1000);
22 }
```

Figure 5: Code for Button-Controlled LED System

3. Understanding the Particle Cloud

In the previous part, we developed a button-controlled LED system, but this system did not communicate with the Particle cloud. In this part, we will first connect the button input module to the Particle cloud and then we will separately connect the Particle cloud to the LED output module.

3.A Experiment with Particle Variables

We can use a *Particle variable* as a way to read variables stored on your Particle Argon from the Particle cloud. We simply need to include a call to the `Particle.variable` function to tell the Particle OS which variables we wish to monitor from the cloud. Remove the LED output module, and connect the button input module to digital port D4.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Enter the code from Figure 6. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon.

Once the code has been uploaded then open the Particle console by clicking the *console* icon in the Particle development environment. Click on the *devices* icon in the Particle console and choose your device from the list. Recall the figure from the lab notes also shown in Figure 7. Find the `button_state` Particle variable in the *variables* section in the lower right-hand corner. Click *get* to refresh the variable. Press the button, click *get*, release the button, click *get*. Confirm that the variable correctly reflects the state of the button on your IoT device.

Sign-Off Milestone: An instructor will choose one scholar in the group to share her screen and show that the Particle variable correctly reflects the state of the button. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

```
1  int button_pin   = D4;
2  int button_state = -1;
3
4  void setup()
5  {
6    pinMode( button_pin, INPUT );
7    Particle.variable( "button_state", button_state );
8  }
9
10 void loop()
11 {
12   // read the button value
13   button_state = digitalRead( button_pin );
14
15   // wait 1 second before reading button again
16   delay(1000);
17 }
```

Figure 6: Code for Button Particle Variable

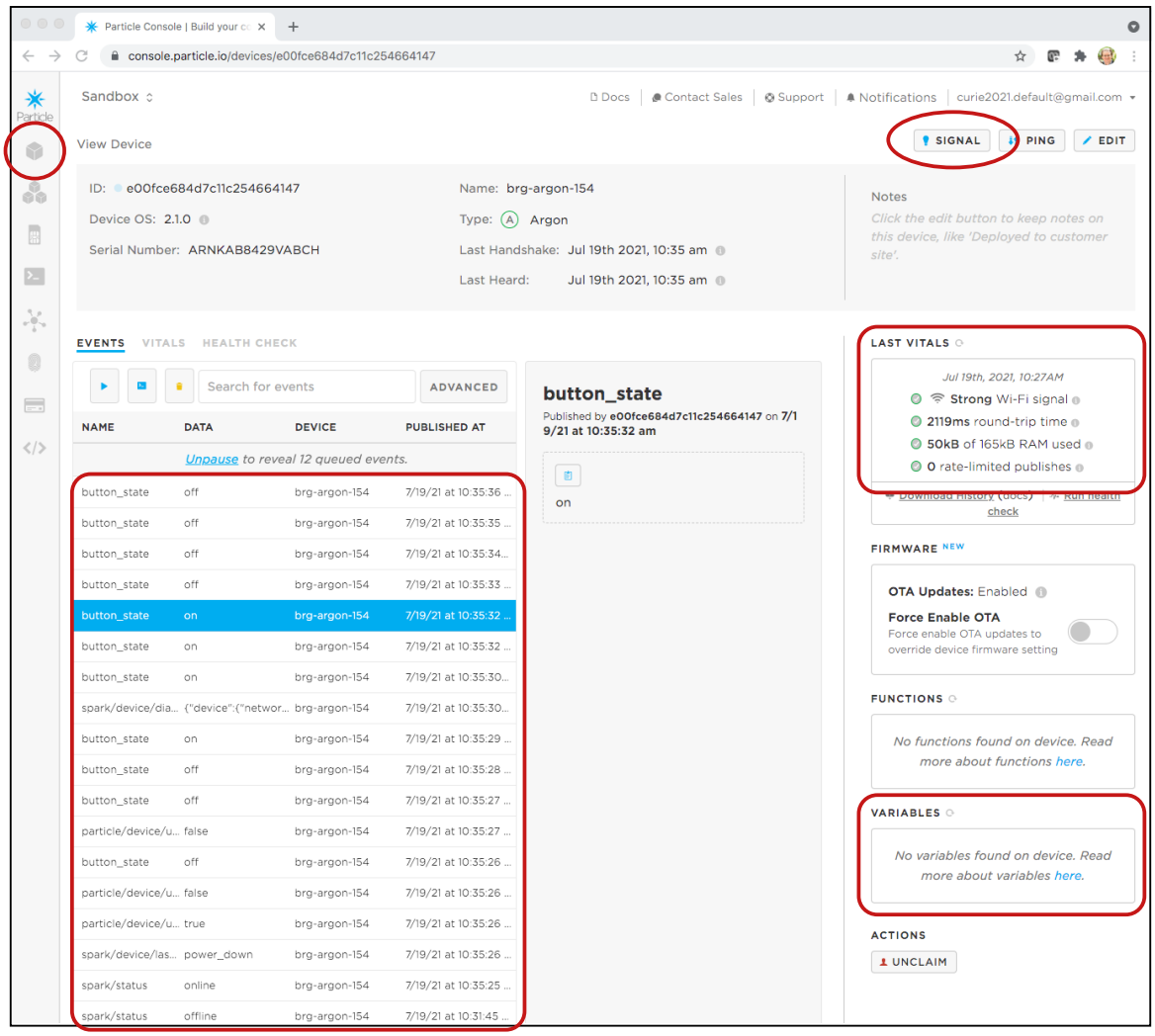

Figure 7: Particle Console Device Page

3.B Experiment with Sending Particle Events

Particle variables are useful for debugging, but they do not enable Particle Argons to communicate with each other. We will need to use Particle events (also called messages) where IoT input devices send events into the cloud and IoT output devices receive events from the cloud. To sending a Particle event to the Particle cloud, We simply need to include a call to the `Particle.publish` function to tell the Particle OS about the event and the data we want to send to the cloud.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Enter the code from Figure 8. To make sure each scholar's events are unique, use your name as part of the event name. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon.

NOTE: Each group account has a limit on the total number of events that can be sent/received to the Particle cloud per month. To stay under this limit, try to avoid sending too many events. Try not to send an event more than once a second and turn off your Particle Argon when you are not using it.

Once the code has been uploaded then open the Particle console by clicking the *console* icon in the Particle development environment. Click on the *events* icon in the Particle console. Press the button input module and watch for the corresponding events to show up in the event list.

Sign-Off Milestone: An instructor will choose one scholar in the group to share her screen and show that the Particle console shows the correct event when the button is pressed. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

```
1  int button_pin   = D4;
2  int button_state = -1;
3
4  void setup()
5  {
6    pinMode( button_pin, INPUT );
7  }
8
9  void loop()
10 {
11   // read the button value
12   button_state = digitalRead( button_pin );
13
14   // NOTE: Replace SCHOLARS_FIRST_NAME with your first name!
15   if ( button_state == 1 ) {
16     Particle.publish( "SCHOLARS_FIRST_NAME_button_state", "on" );
17   }
18   else {
19     Particle.publish( "SCHOLARS_FIRST_NAME_button_state", "off" );
20   }
21
22   // wait 1 second before reading button again
23   delay(1000);
24 }
```

Figure 8: Code for Sending Particle Events

3.C Experiment with Receiving Particle Events

The previous part experimented how to send Particle events, and in this part we will experiment with how to receive Particle events. Receiving events is a bit more complicated. We need to include a call to the `Particle.subscribe` function to tell the Particle OS what *handler* function to call whenever a specific kind of event is received, and then we need to actually process the event in the handler function. Remove the button input module and connect the LED output module to digital port D4.

Create a fresh program by clicking the *code* icon and then clicking *create new app*. **It is very important that you always prefix the name of your programs with your first name to ensure that all members of the group are editing their own unique files!** Enter the code from Figure 9. To make sure each scholar's events are unique, use your name as part of the event name. **Always confirm you are working with your specific Particle Argon by checking the device name in the lower right-hand corner.** Compile and upload your program to the Particle Argon using the *flash* (lightning) icon.

Once the code has been uploaded then open the Particle console by clicking the *console* icon in the Particle development environment. Click on the *events* icon in the Particle console. Recall the figure from the lab notes also shown in Figure 10. We can use the Particle console to not just monitor events but also to send new Particle events directly from the cloud. Click on the little blue arrow head. Enter in the name of your event and either "on" or "off". Then click *publish*. Verify that the LED output module correctly turns on or off.

Sign-Off Milestone: An instructor will choose one scholar in the group to share her screen and show she is able to turn the LED output module on/off from the Particle console using events. Ideally, all scholars in the group should sign-off this milestone before moving on to the next milestone. Work together to help each other achieve the milestone!

```
1  int led_pin = D4;
2
3  void receive_msg( const char* event, const char* msg )
4  {
5    if ( strcmp( msg, "on" ) == 0 )
6    {
7      digitalWrite( led_pin, HIGH );
8    }
9    else {
10     digitalWrite( led_pin, LOW );
11   }
12 }
13
14 void setup()
15 {
16   pinMode( led_pin, OUTPUT );
17   Particle.subscribe( "NAME_toggle_led", receive_msg );
18 }
19
20 void loop()
21 {
22   // empty
23 }
```

Figure 9: Code for Receiving Particle Events

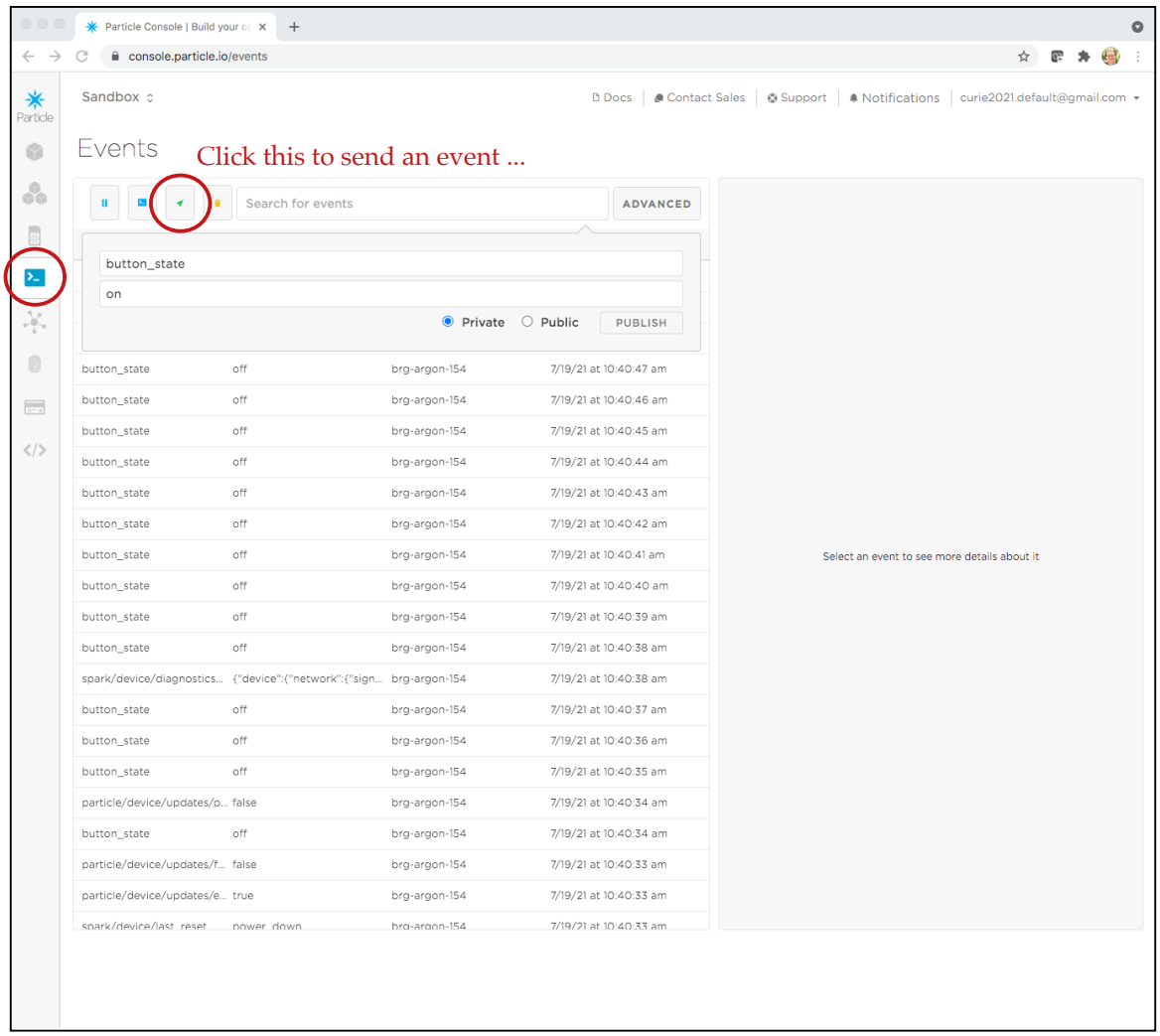

Figure 10: Particle Console Events Page

4. Building an IoT System

In the previous part, we learned how to use Particle events. In this part we will now put together the fully integrated "smart light" system as shown in Figure 11 from the notes.

4.A Develop a "Smart Light" System

Start by choosing which scholar(s) will implement the IoT input device and which scholar(s) will implement the IoT output device. You will need at least one scholar implementing each type of device. For the IoT input device, connect the button input module to digital port D4. For the IoT output device, also connect the LED output module to digital port D4.

The code for the IoT input device is shown in Figure 12(a) and the code for the IoT output device is shown in Figure 12(b). Notice how the event name no longer includes your name as a prefix. This is critical since this will enable all of your Particle Argon devices to send and receive the same event and thus enables them to communicate with each other! Each scholar should create a new program with the code for their device based on wether they are implementing the IoT input device or the IoT output device. Make sure to prefix the name of your program with your first name!

Once everything is connected and all devices have been programmed, then test out your "smart light" system. If any scholar with an IoT input device pesses her button, then the LEDs on all of the IoT output devices should turn on. When the scholar releases her button, then all of the LEDs should turn off. Try using the the Particle console to monitor the events!

Sign-Off Milestone: Demonstrate the fully working "smart light" system to an instructor.

4.B Share Photo or Video of "Smart Light" on Slack

Now that you have your "smart light" working, take a photo of your final device, code, and/or record a short video of your system in operation. Then upload your photo and/or video to Slack using the `#lab2-final-media-milestone` channel. It would be great if you could maybe even have someone else take a photo of you either hooking up your device or showing it off for the camera!

Sign-Off Milestone: Show your uploaded photo and/or video to an instructor.

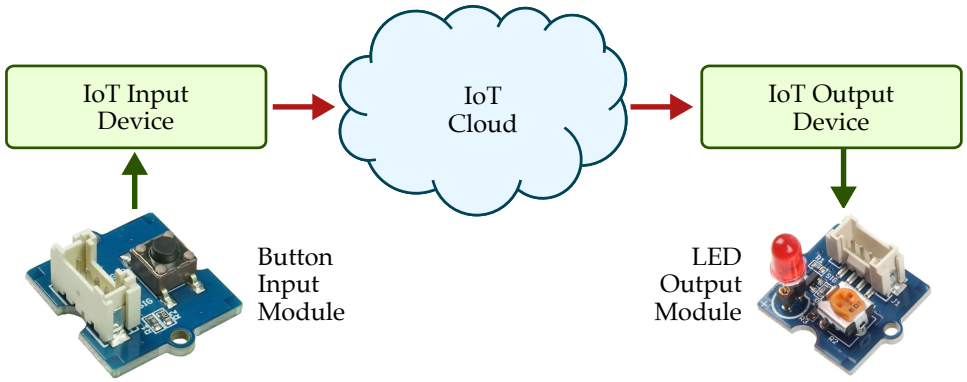

Figure 11: Diagram of Simple "Smart Light" System

```
int button_pin   = D4;
int button_state = -1;

void setup()
{
  pinMode( button_pin, INPUT );
}

void loop()
{
  button_state = digitalRead( button_pin );
  if ( button_state == 1 ) {
    Particle.publish( "button_state", "on" );
  }
  else {
    Particle.publish( "button_state", "off" );
  }
  delay(1000);
}
```

(a) IoT Input Device Program

```
int led_pin = D4;

void receive_msg( const char* event, const char* msg )
{
  if ( strcmp( msg, "on" ) == 0 ) {
    digitalWrite( led_pin, HIGH );
  }
  else {
    digitalWrite( led_pin, LOW );
  }
}

void setup()
{
  pinMode( led_pin, OUTPUT );
  Particle.subscribe( "button_state", receive_msg );
}

void loop()
{
  // empty
}
```

(b) IoT Output Device Program

Figure 12: Complete C++ Programs for Smart Light

Next Steps

Parts 1–4 are the required portions of the lab. If you complete these parts quickly, you can start to get ready for a final activity which will involve all of the scholars coming together to experiment with an IoT geolocation system. To prepare connect the LED output module to digital port D4 and the button input module to digital port D2. Then take a few minutes to read this Khan Academy article about geolocation:

- `https://tinyurl.com/92zywfs5`

We will be using the Wi-Fi positioning system strategy to geolocate all of the scholars' Particle Argon devices.