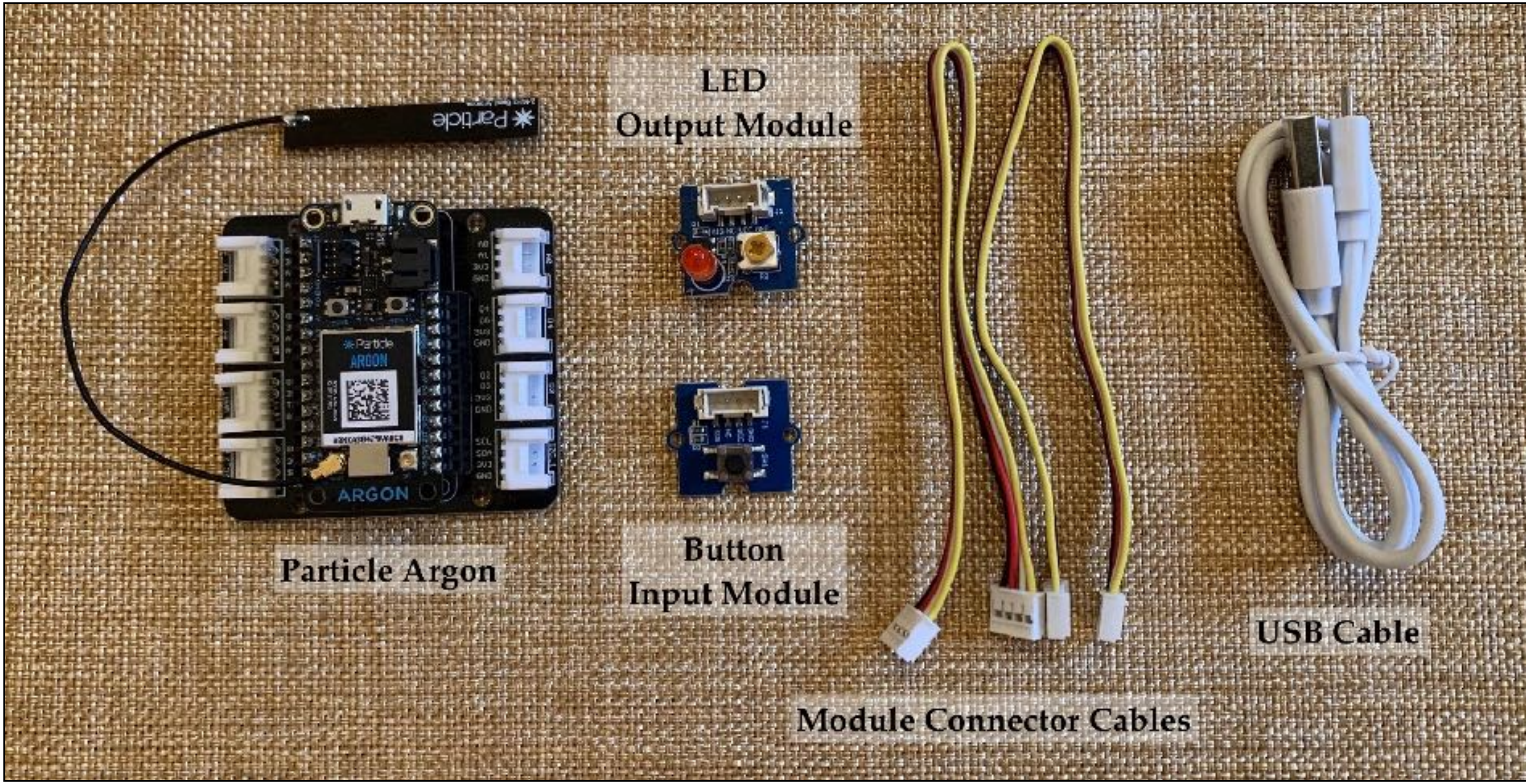
CURIE Academy, Summer 2021
Lab 2: Computer Engineering
Software Perspective

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

Materials Required for Lab 2



Computer Systems Stack

Layer	Example
Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	
Register-Transfer Level	Ripple Carry Adder
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

CURIE Lab 1

Computer Systems Stack

Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	
Register-Transfer Level	Ripple Carry Adder
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

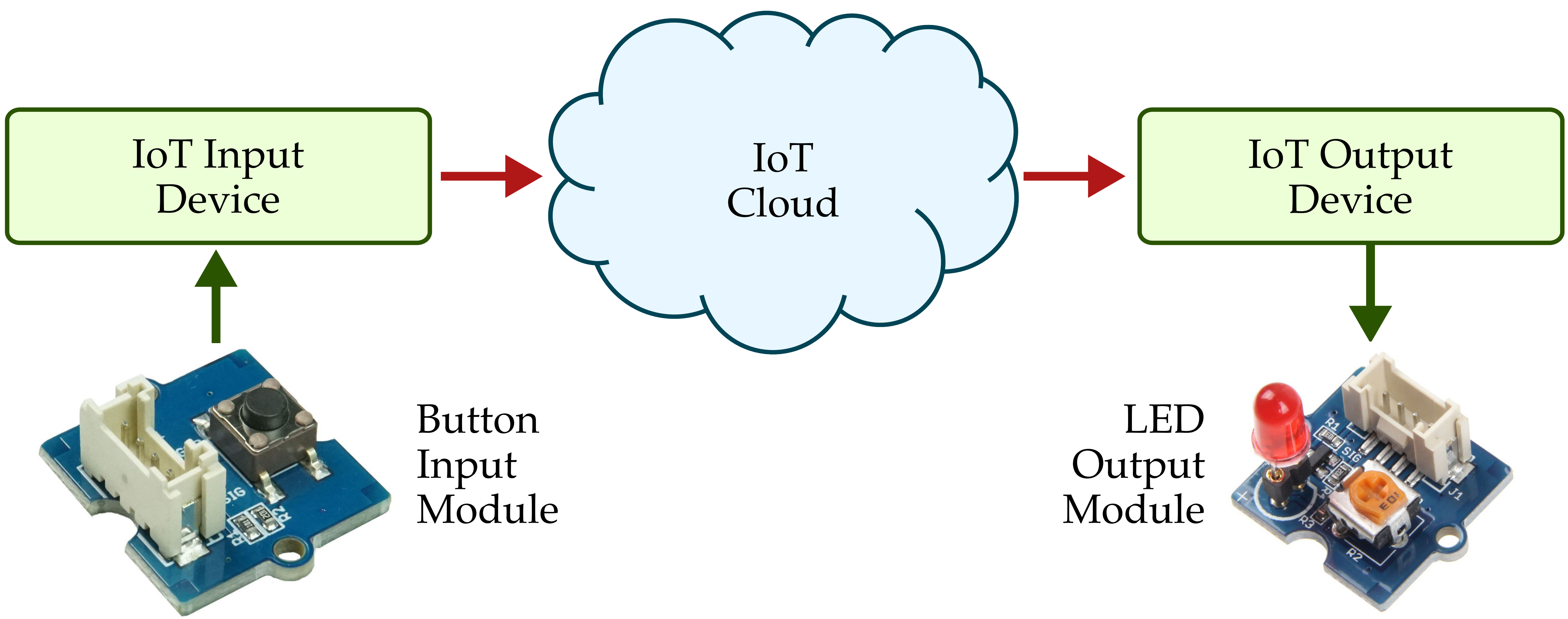
Computer Engineering

CURIE Lab 2

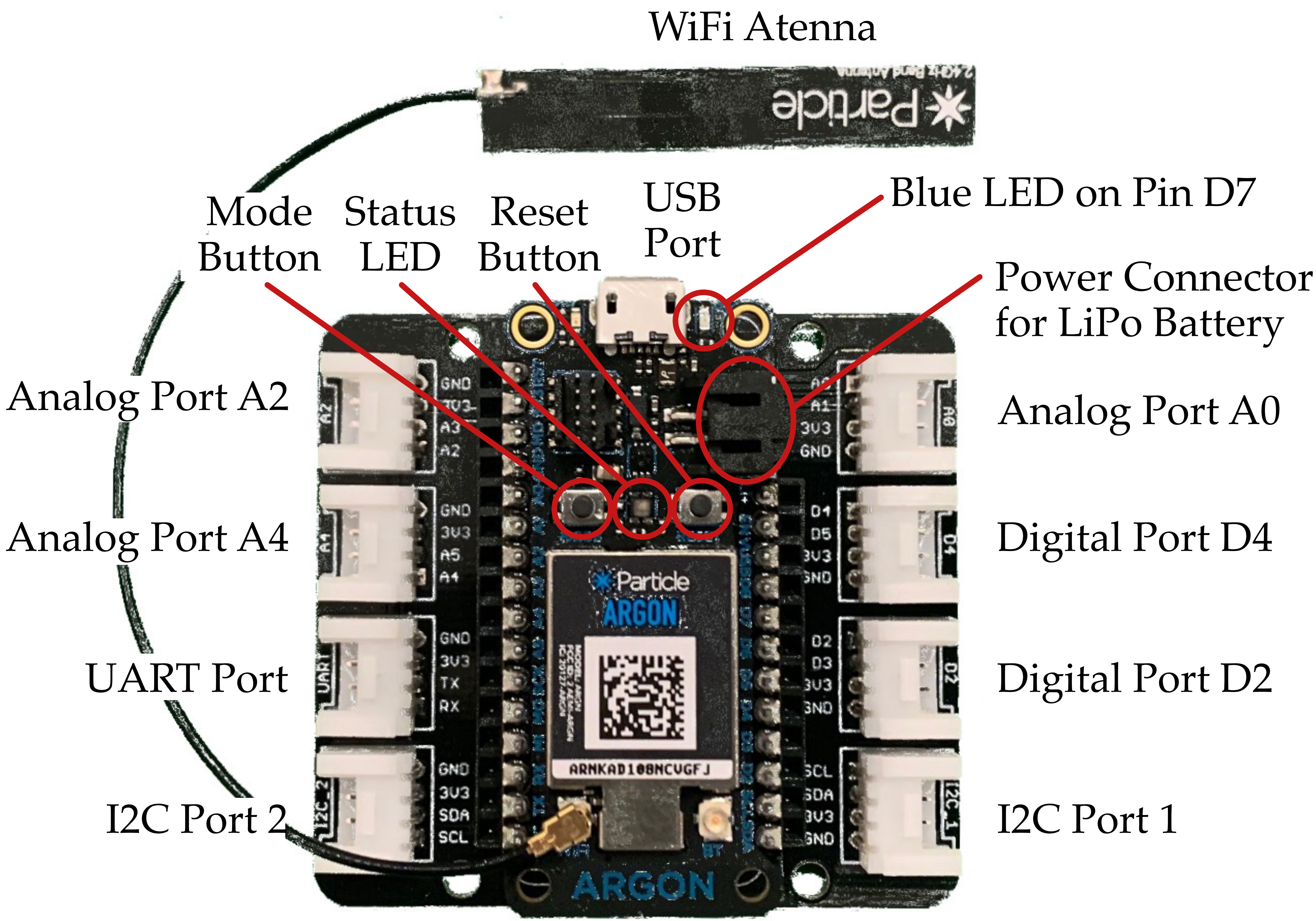
CURIE Lab 1

Application: “Smart Light” System

Application: “Smart Light” System



Particle Argon



Computer Systems Stack

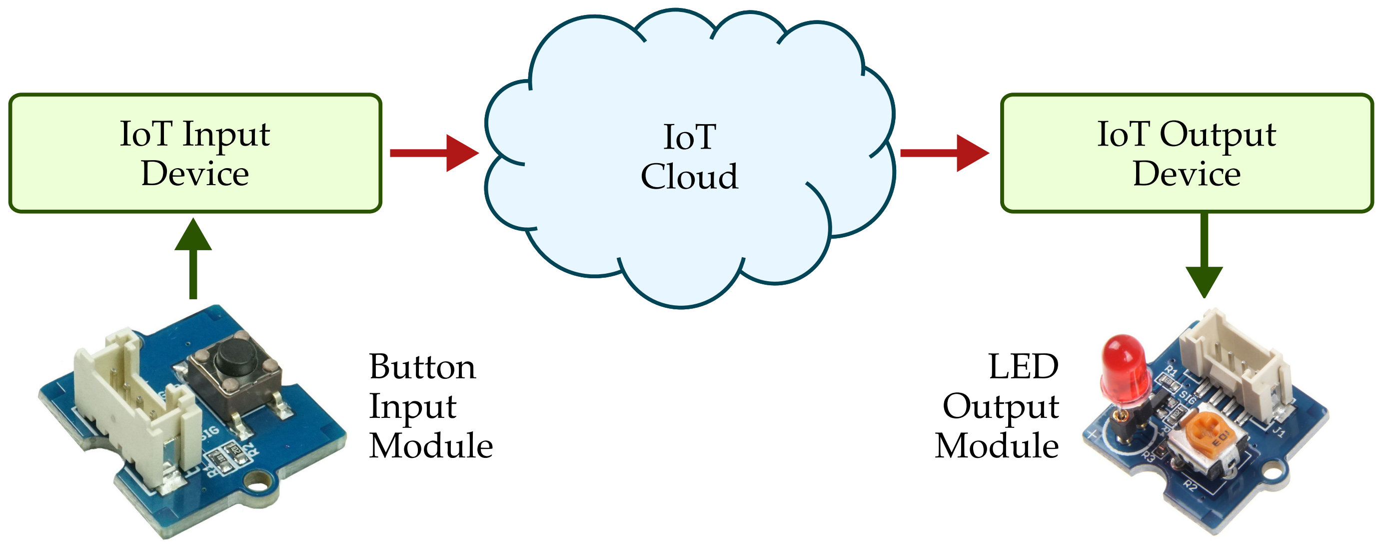
Application	Smart Light
Algorithm	**Flowchart**
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	
Register-Transfer Level	Ripple Carry Adder
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

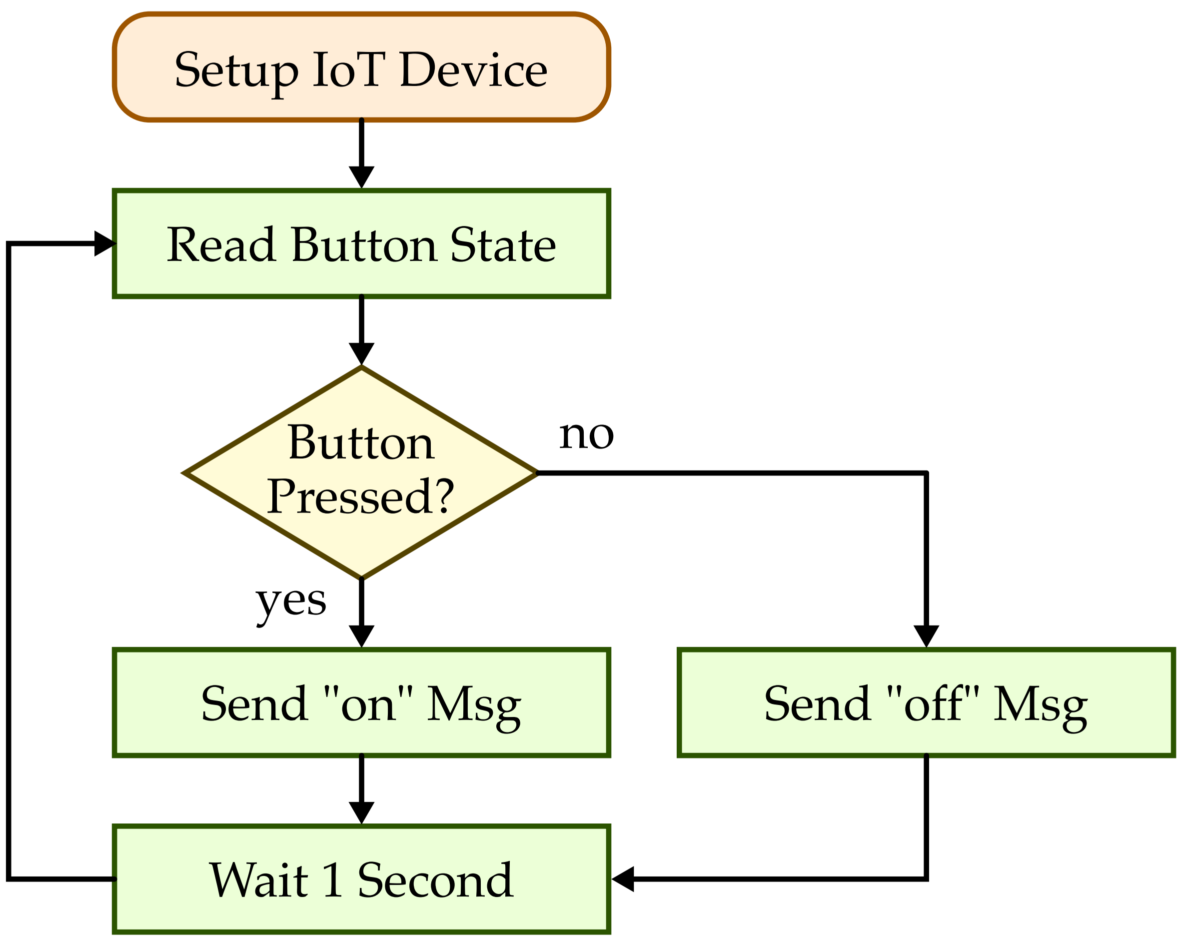
Computer Engineering

CURIE Lab 2

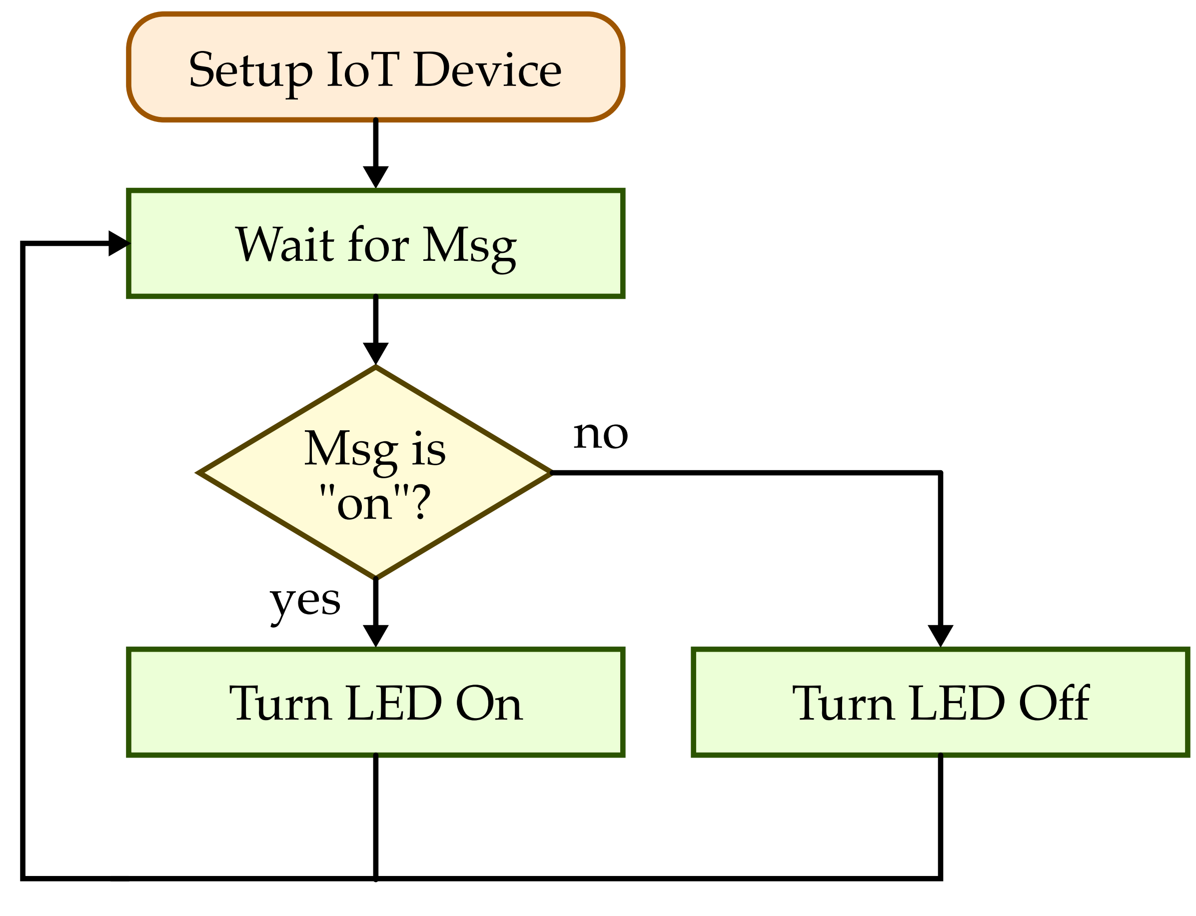
CURIE Lab 1

Algorithm: Flowcharts





(a) Flowchart for IoT Input Device



(b) Flowchart for IoT Output Device

Computer Systems Stack

Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	Ripple Carry Adder
Register-Transfer Level	
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

CURIE Lab 1

Programming Language: C++

```cpp
int a;       // declaration
a = 2;       // assignment
int b = 3;   // initialization

int c;
c = a + b;
```

(a)

```cpp
// function to add two integers
int add( int a, int b )
{
  int sum;
  sum = a + b;
  return sum;
}
```

(b)

```cpp
int c;
c = add( 2, 3 );
```

(c)

Figure 6: Example C++ Code Snippets

Programming Language: C++

```
int a;       // declaration
a = 2;       // assignment
int b = 3;   // initialization

int c;
c = a + b;
```

(a)

```
// function to add two integers
int add( int a, int b )
{
  int sum;
  sum = a + b;
  return sum;
}
```

(b)

```
int c;
c = add( 2, 3 );
```

(c)

Figure 6: Example C++ Code Snippets

```
int button_state;
button_state = read_button_state( button_pin );

if ( button_state == 1 ) {
  // send "on" msg
}
else {
  // send "off" msg
}

// wait 1 second
```

(a) Sketch of IoT Input Device Program

```
void receive_msg( msg )
{
  if ( msg == "on" ) {
    // turn light on
  }
  else {
    // turn light off
  }
}
```

(b) Sketch of IoT Output Device Program

Figure 7: Sketch of C++ Programs for Smart Light

Computer Systems Stack

Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	**Particle OS**
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	Ripple Carry Adder
Register-Transfer Level	
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

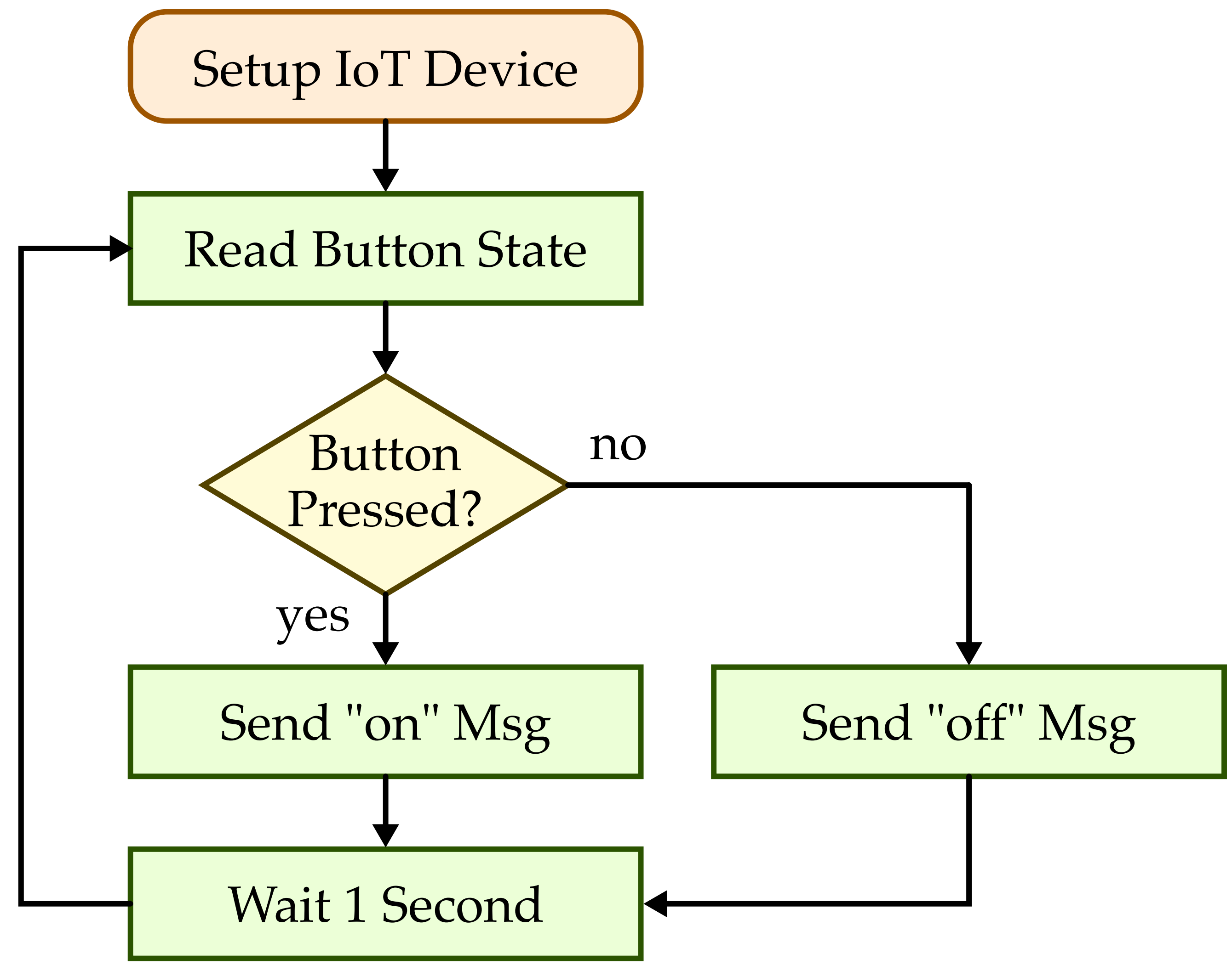
CURIE Lab 1

Compiler: Particle OS

```
// Global constants for pin assignments and global variables

int led_pin = D7;

int x = 2;
int y = 3;
int z = 0;

// Helper functions

int add( int a, int b )
{
  int sum;
  sum = a + b;
  return sum;
}

// The setup routine runs once when you press reset

void setup()
{
  // Configure led_pin as digital output
  pinMode( led_pin, OUTPUT );
}

// The loop routine runs over and over again

void loop()
{
  // Do the addition
  z = add( x, y );

  // Blink LED z times
  for ( int i = 0; i < z; i++ ) {
    digitalWrite( led_pin, HIGH );  // Turn on the LED
    delay(500);                     // Wait 0.5 seconds
    digitalWrite( led_pin, LOW );   // Turn off the LED
    delay(500);                     // Wait 0.5 seconds
  }

  // Wait four seconds
  delay(4000);
}
```

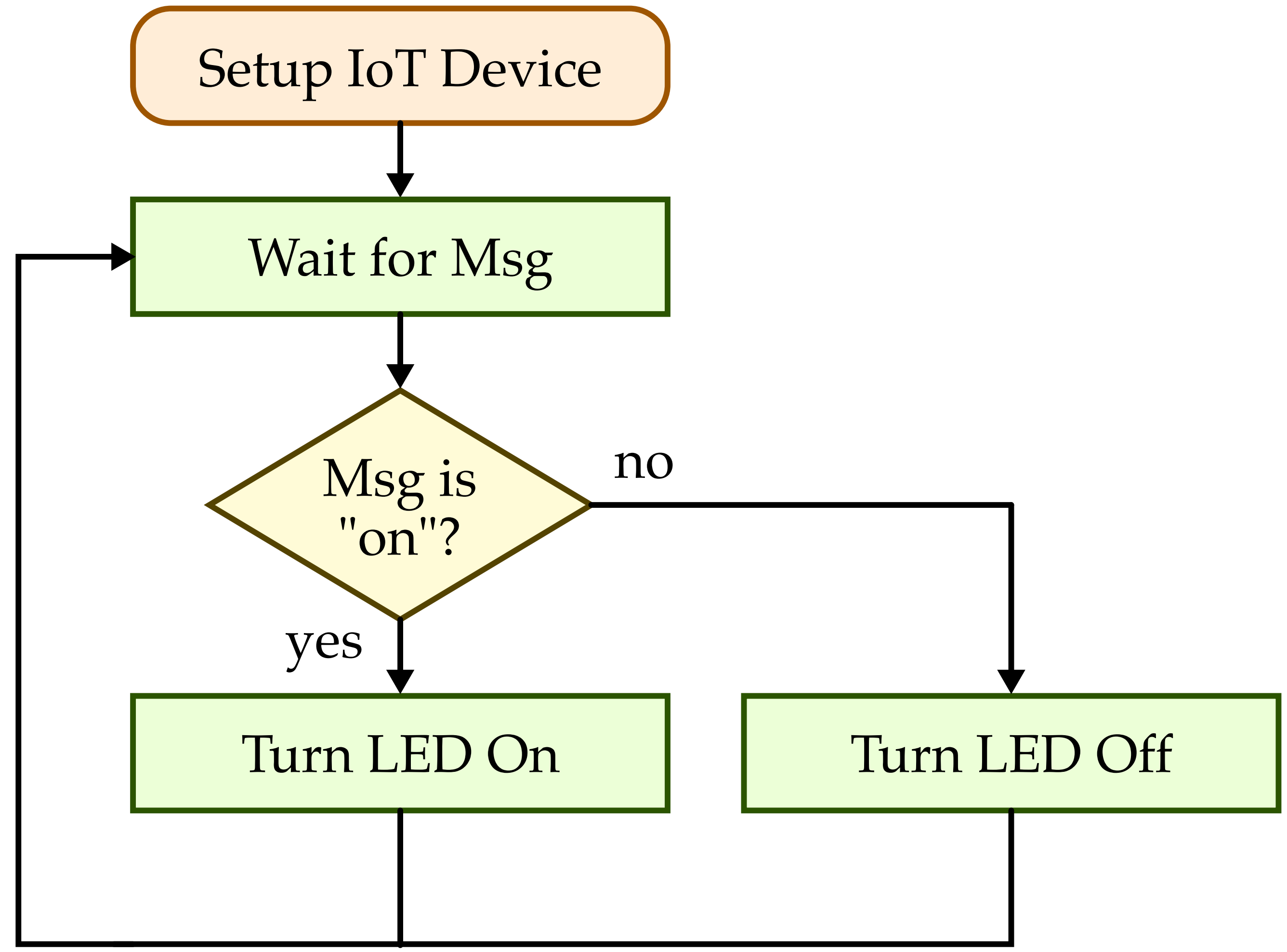
Compiler: Particle OS

```
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);
}
```



Compiler: Particle OS

```
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
}
```



Computer Systems Stack

Layer	Example
Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	Ripple Carry Adder
Register-Transfer Level	
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

CURIE Lab 1

Compiler: Particle Devel Environment

Flash

Verify

Save

Code

Library

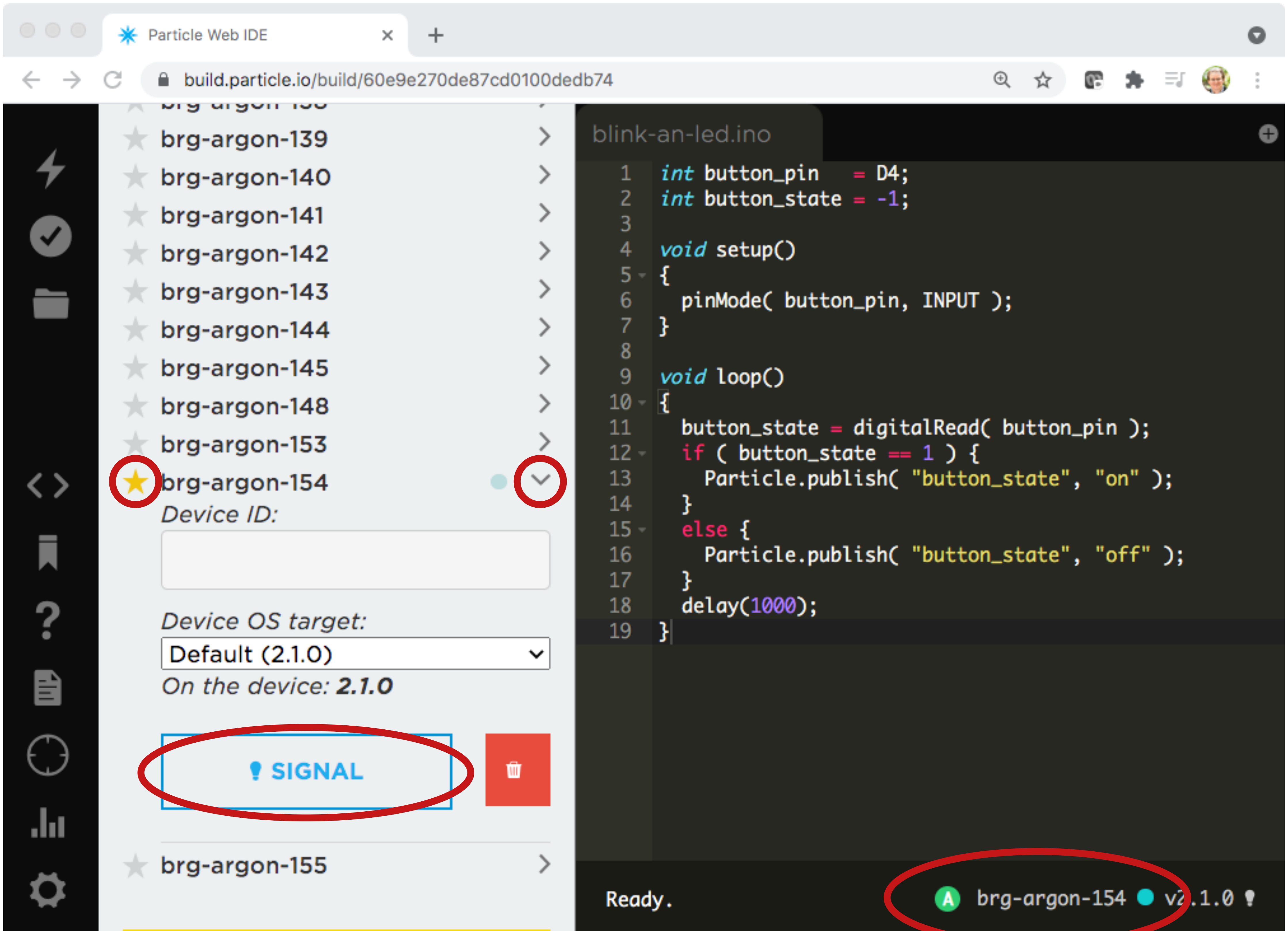
Help

Docs

Devices

Console

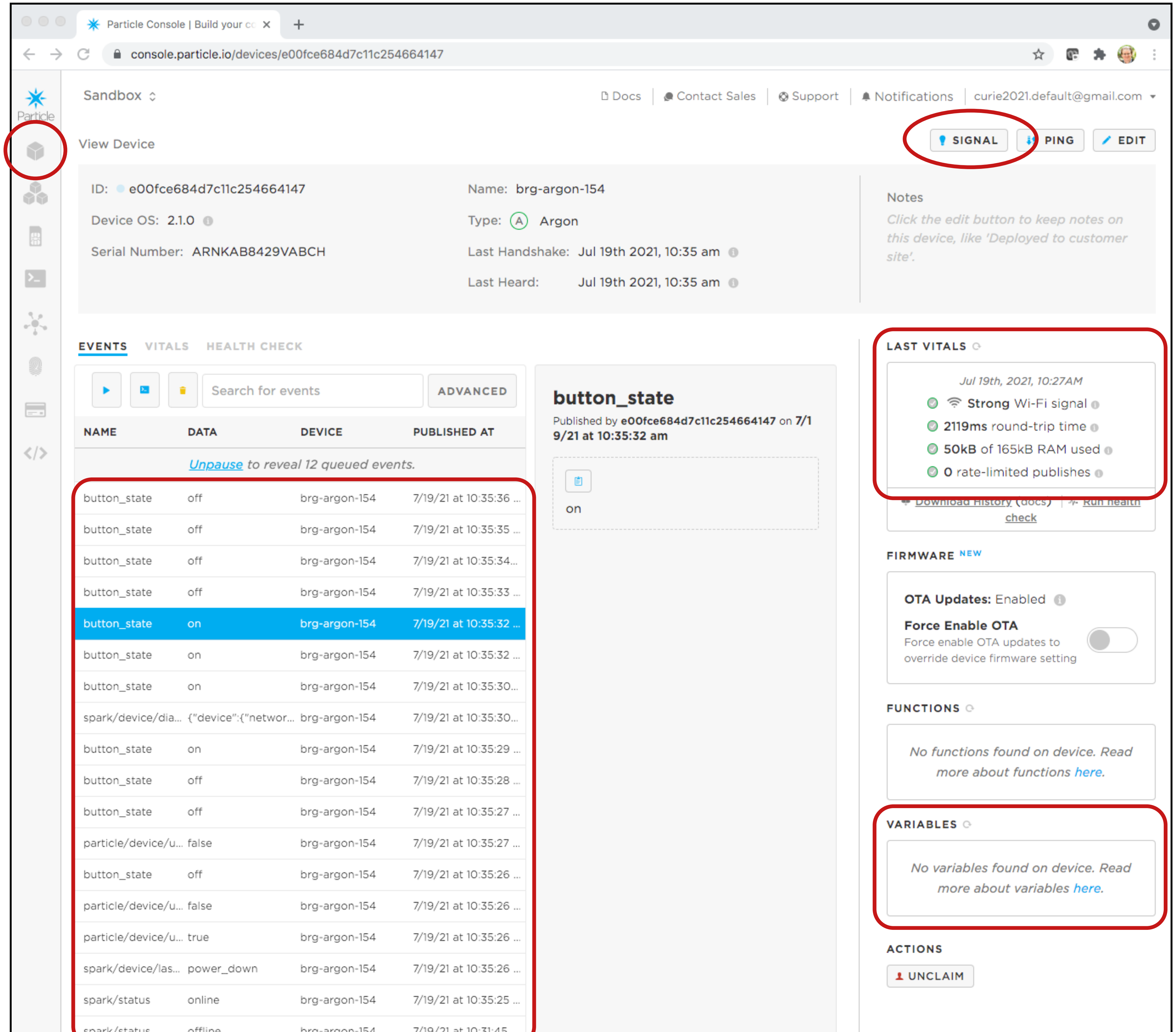
Settings



Compiler: Particle Devel Environment

- Always confirm your Particle Argon status LED is breathing cyan
- Always confirm that your Particle Argon is selected in the device list as indicated by the yellow star
- Always confirm that your Particle Argon is selected as indicated by the device name in the lower left-hand corner
- Always prefix the names your Particle Argon C++ programs with your first name (e.g., jane-blink-led)

Compiler: Particle Devel Environment



Computer Systems Stack

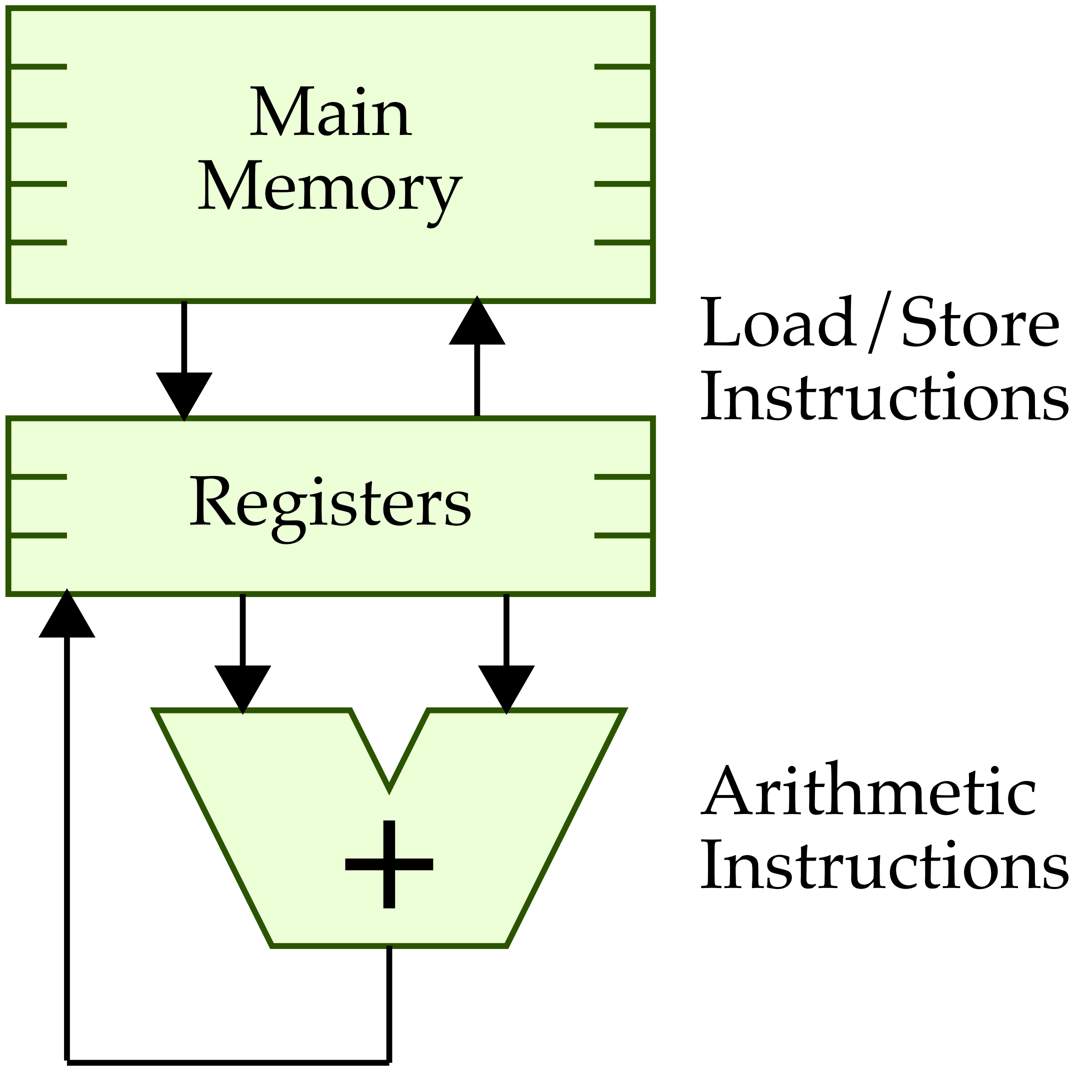
Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	
Register-Transfer Level	Ripple Carry Adder
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

CURIE Lab 1

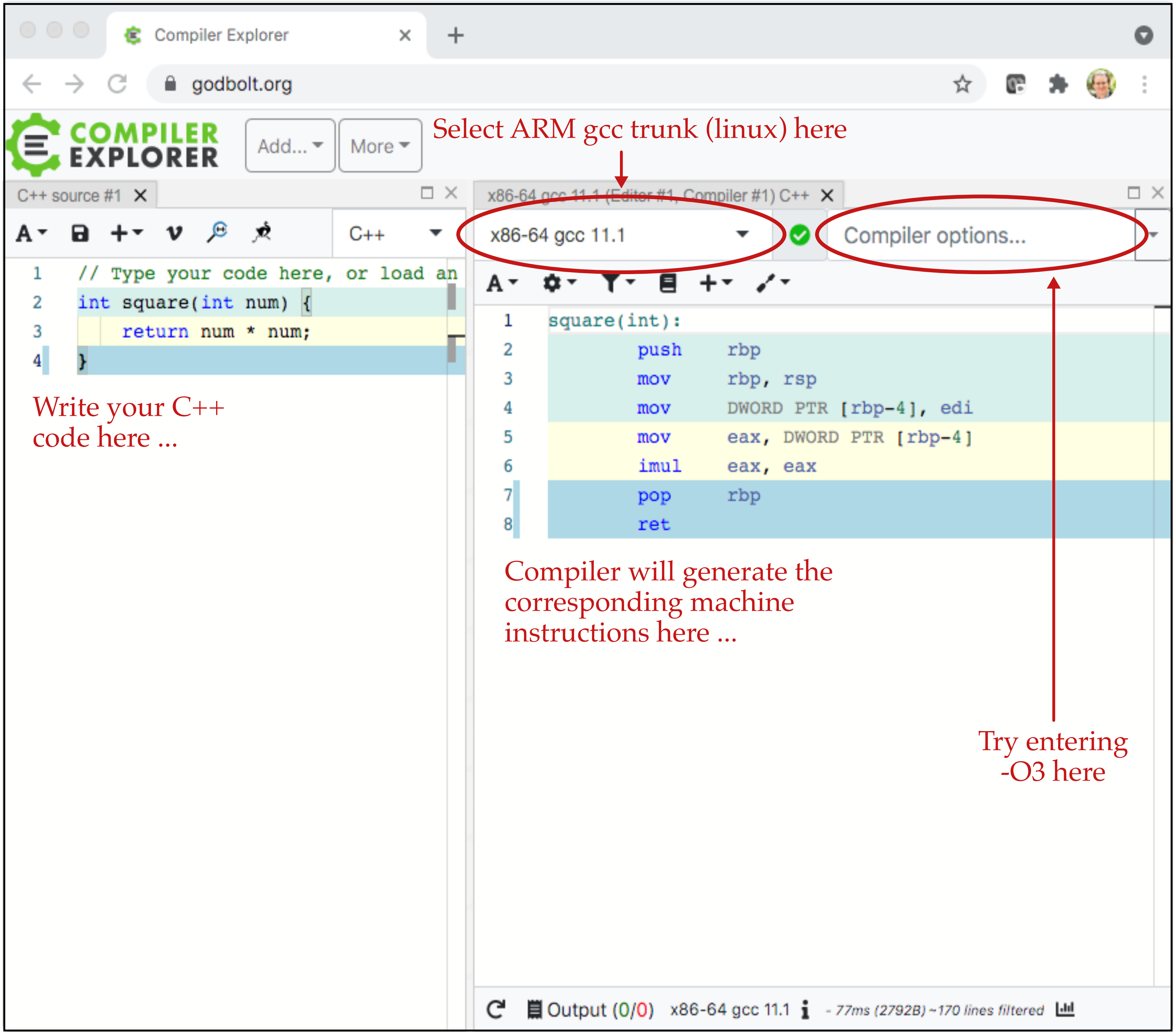
ISA: ARM Machine Instructions




```c
int add( int a, int b )
{
  int sum;
  sum = a + b;
  return sum;
}
```

```
# load two values from main
# memory into two registers
ldr     r2, [r7, #4]
ldr     r3, [r7]

# do the actual addition
add     r3, r3, r2

# store the sum from a register
# back into main memory
str     r3, [r7, #12]
```

ISA: ARM Machine Language



Lab 2 Overview

- Part 1.A Test Simple Addition Program
- Part 1.B Examine Machine Instructions
- Part 2.A Experiment with LED Output
- Part 2.B Experiment with Button Input
- 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.A Develop a “Smart Light” System
- Part 4.B Share Photo or Video of IoT System
- Experiment with IoT Geolocation System

Let’s write a simple blinking LED program

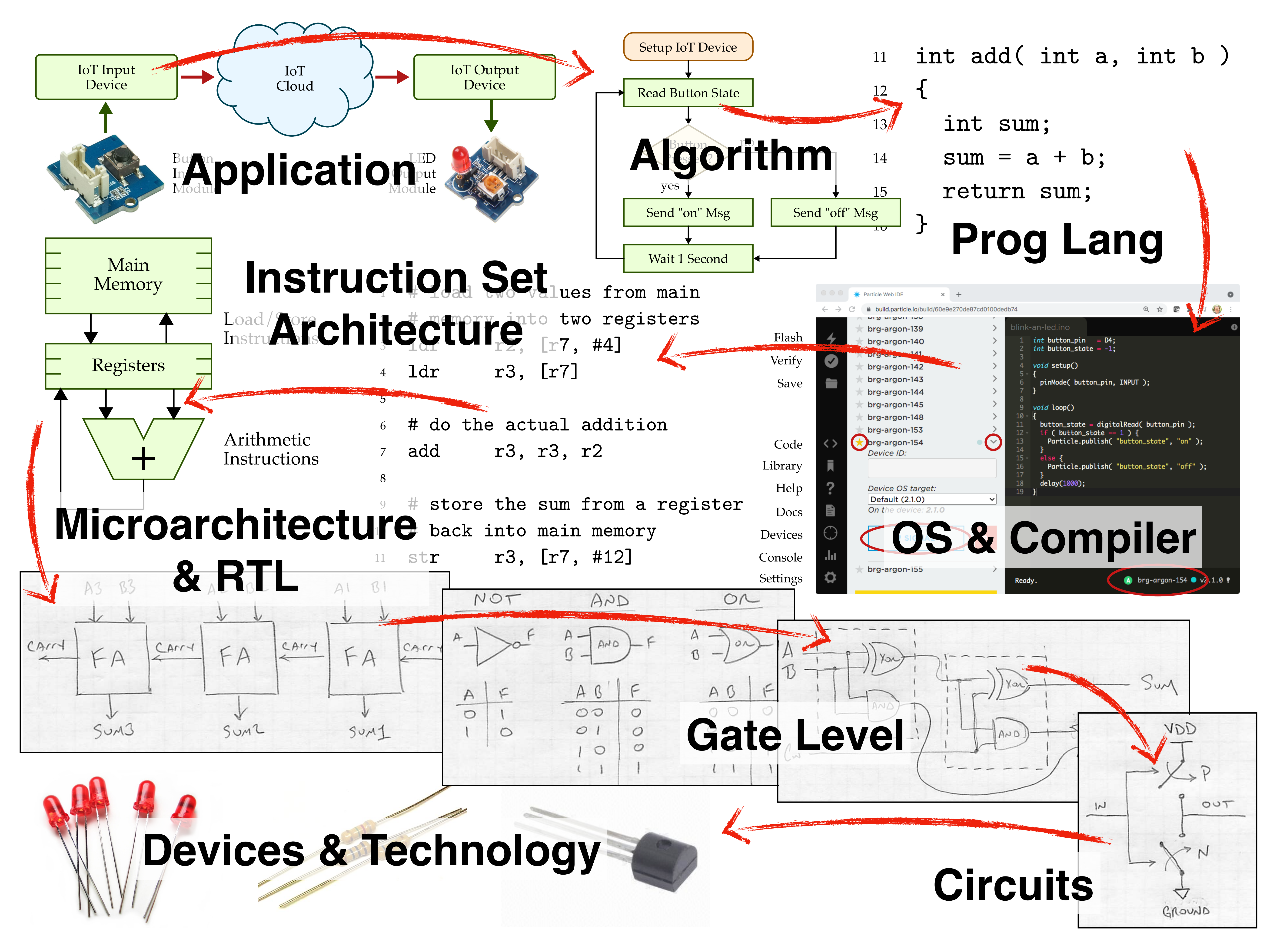
Computer Systems Stack

Application	Smart Light
Algorithm	Flowchart
Programming Language	C++
Operating System	Particle OS
Compiler	Particle Development Environment
Instruction Set Architecture	ARM Machine Instructions
Microarchitecture	Ripple Carry Adder
Register-Transfer Level	
Gate Level	NOT, AND, OR, XOR
Circuits	Inverter
Devices	Resistors, LEDs, Transistors
Technology	

Computer Engineering

CURIE Lab 2

CURIE Lab 1



Application
IoT Input Device
IoT Cloud
IoT Output Device
Algorithm
Setup IoT Device
Read Button State
yes
Send "on" Msg
Send "off" Msg
Wait 1 Second
Prog Lang
int add(int a, int b)
{
int sum;
sum = a + b;
return sum;
}
Instruction Set Architecture
Main Memory
Registers
Load/Store Instructions
Arithmetic Instructions
# load two values from main
# memory into two registers
ldr r2, [r7, #4]
ldr r3, [r7]
# do the actual addition
add r3, r3, r2
# store the sum from a register
# back into main memory
str r3, [r7, #12]
OS & Compiler
Flash
Verify
Save
Code
Library
Help
Docs
Devices
Console
Settings
Microarchitecture & RTL
CARRY
FA
SUM3
SUM2
SUM1
Gate Level
NOT
AND
OR
XOR
SUM
Circuits
VDD
IN
OUT
P
N
GROUND
Devices & Technology