



# Particle

**PARTICLE 101 – A LAP AROUND THE PARTICLE ECOSYSTEM**

**WHY PARTICLE**

**THE PARTICLE CLOUD & FRIENDS**

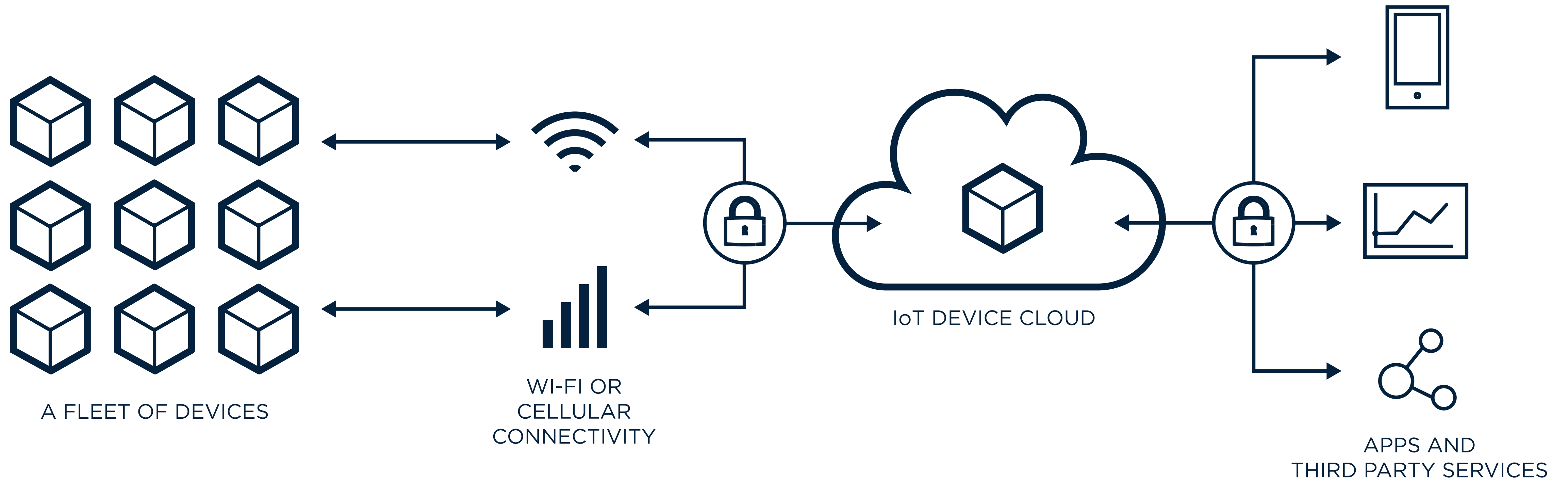
**CLAIMING YOUR FIRST DEVICE**

**WHY PARTICLE**

**THE PARTICLE CLOUD & FRIENDS**

**CLAIMING YOUR FIRST DEVICE**

# PARTICLE IS A FULL-STACK IOT DEVICE PLATFORM



**IOT DEVICE HARDWARE AND FIRMWARE**

**WI-FI AND CELLULAR MVNO**

**IOT DEVICE CLOUD**

**WEB/MOBILE APP SDKS AND INTEGRATIONS WITH THIRD-PARTY SERVICES**

# WE HAVE GROWN THE WORLD'S LARGEST IOT DEVELOPER COMMUNITY

## PARTICLE BY THE NUMBERS



**170,000**

~~140,000~~ developers

Developers love Particle. We have the largest IoT developer community in the industry



**170 countries**

Our customers' devices are deployed all over the world, from Argentina to Antarctica



**8,500 companies building with Particle**

According to IDC, we have the highest customer satisfaction rating of any IoT platform



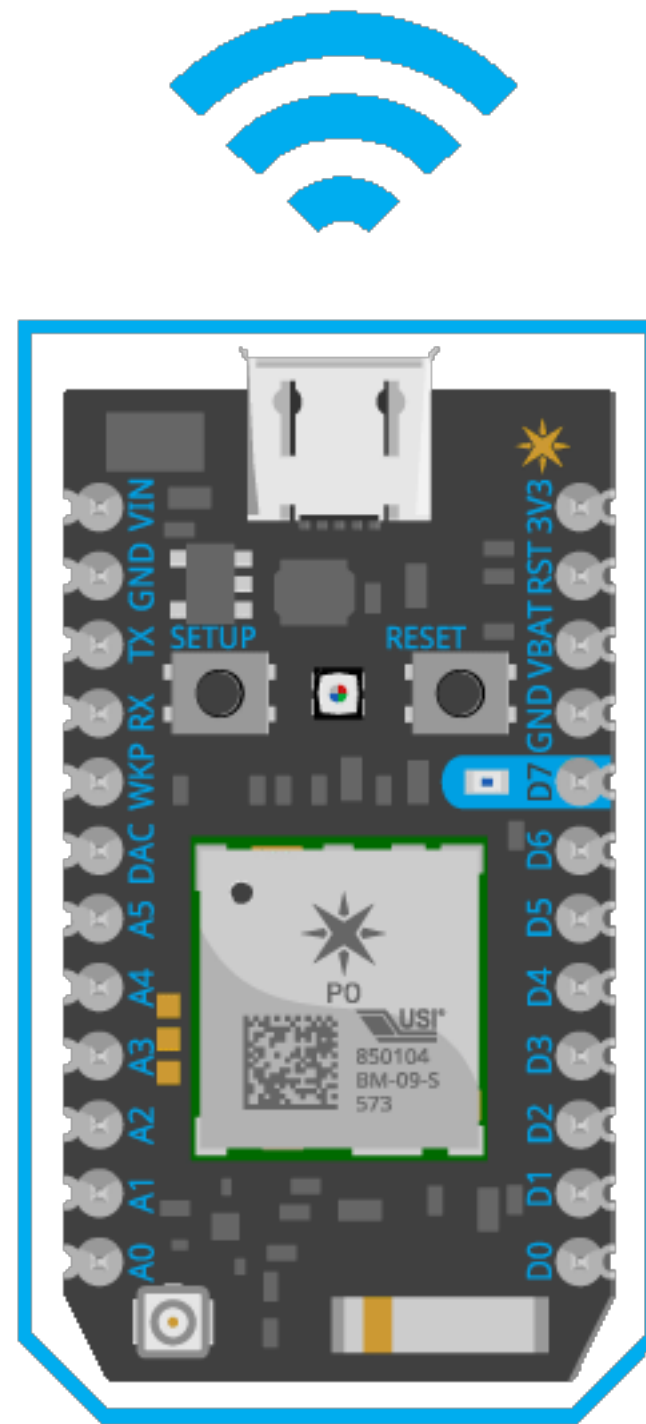
**500,000 devices**

We manage hundreds of thousands of devices sending billions of messages per month

**WE FOCUS ON SOLVING REAL PROBLEMS FOR REAL CUSTOMERS**

# WI-FI FOR PROTOTYPING AND PRODUCTION

WITH THE SAME DEVICE OS ON ALL MODULES, YOU CAN PROTOTYPE ON A PHOTON AND SCALE UP TO A P0 OR P1 WITH NO FIRMWARE CHANGES



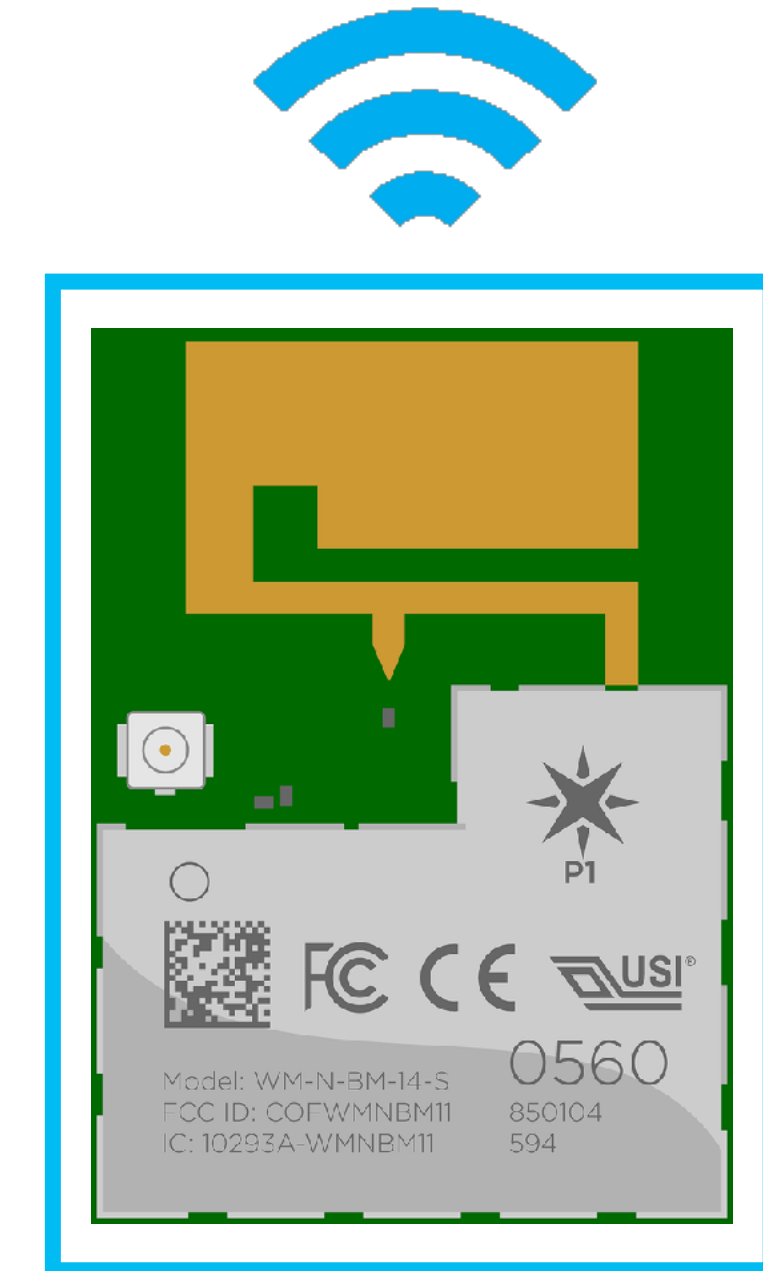
## Photon Development Kit

- Breadboard-friendly Dev Kit
- Available with or without headers



## P0

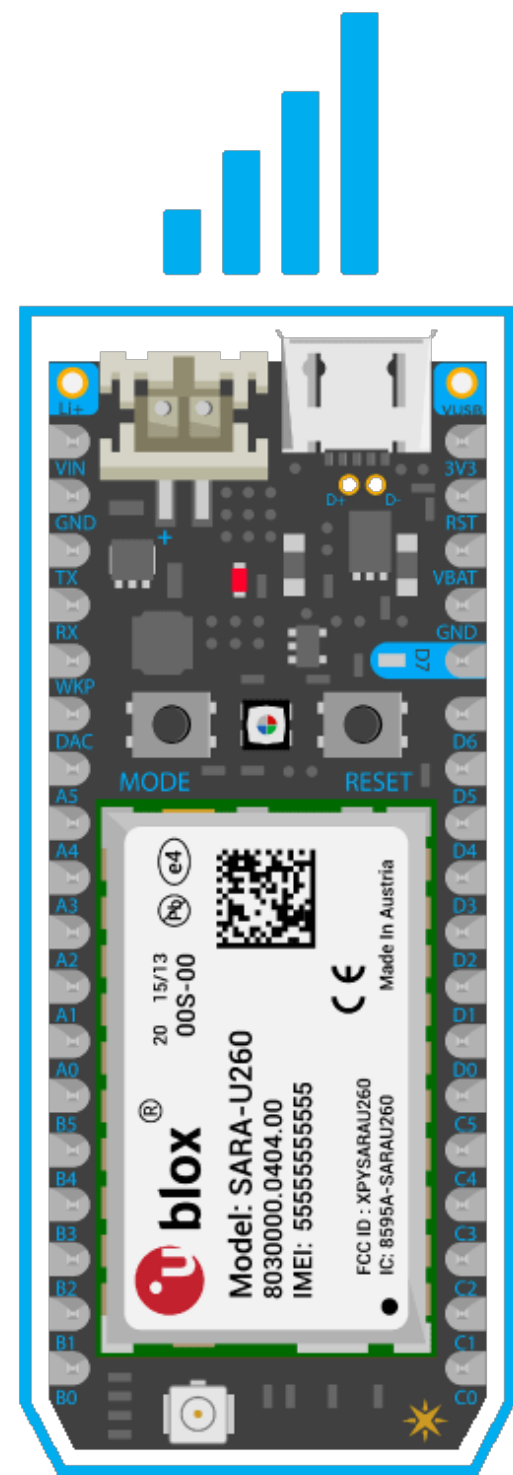
- Small, mass-production form-factor
- Available in mfg-ready reels



## P1

- Built-in PCB antenna and external antenna connector
- Available in mfg-ready reels

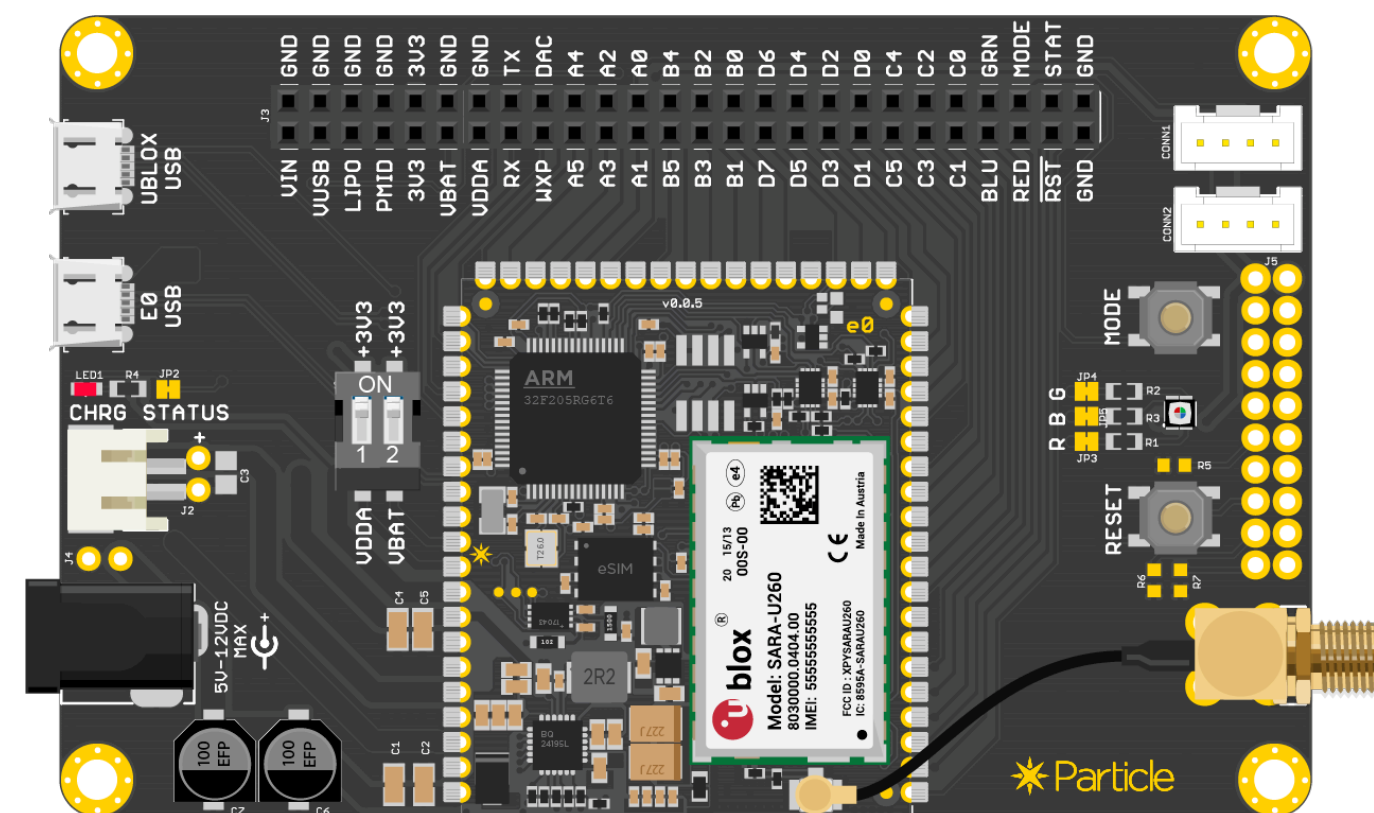
# CELLULAR FOR PROTOTYPING AND PRODUCTION



## Electron Development Kit

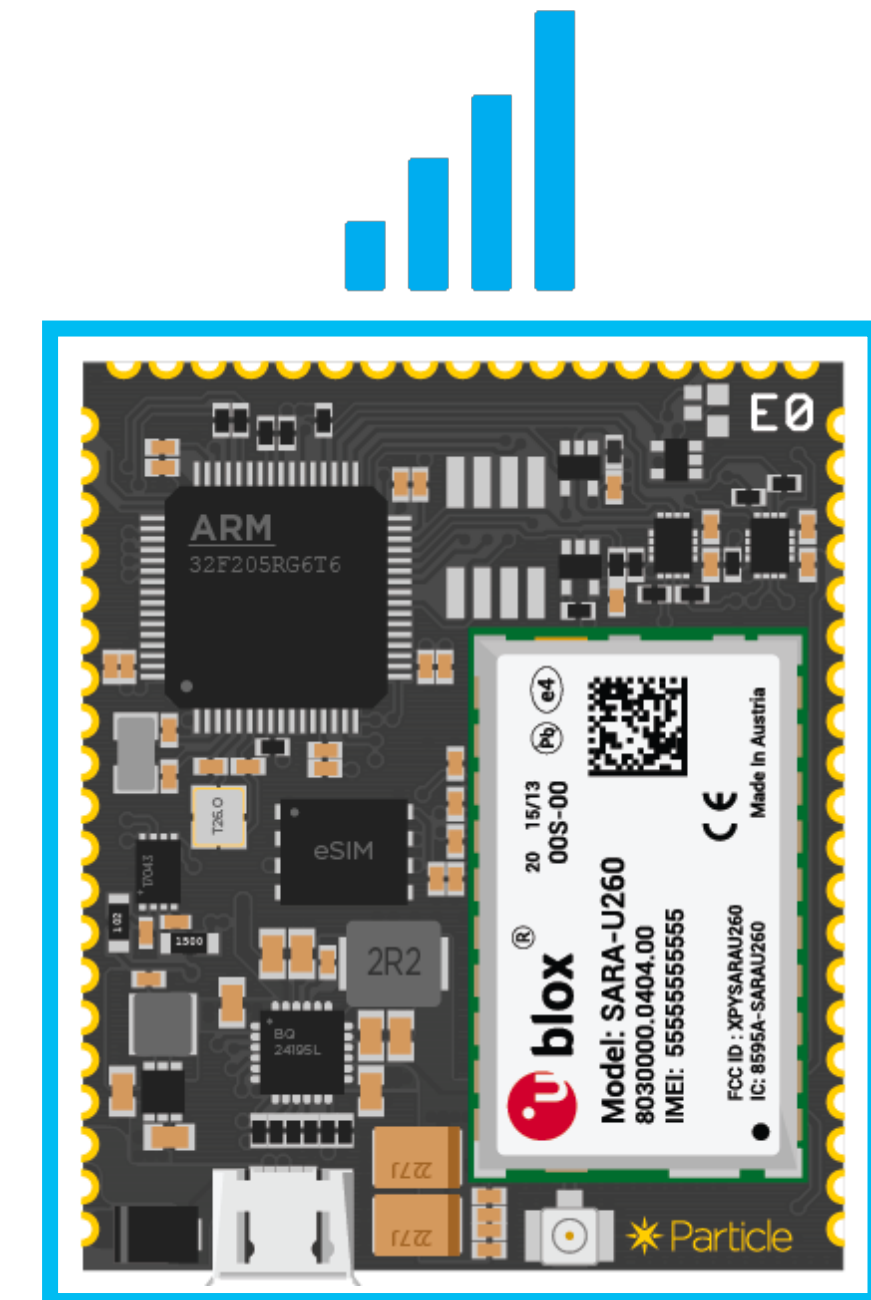
- Breadboard-friendly Dev Kit
- Available only with headers

WITH THE SAME DEVICE OS ON ALL MODULES, YOU CAN PROTOTYPE ON AN ELECTRON AND SCALE UP TO AN E-SERIES WITH MINIMAL FIRMWARE CHANGES



## E-Series Eval Board

- Simple breakout board for evaluating the E-Series module
- Pins accessible via easy-to-use headers

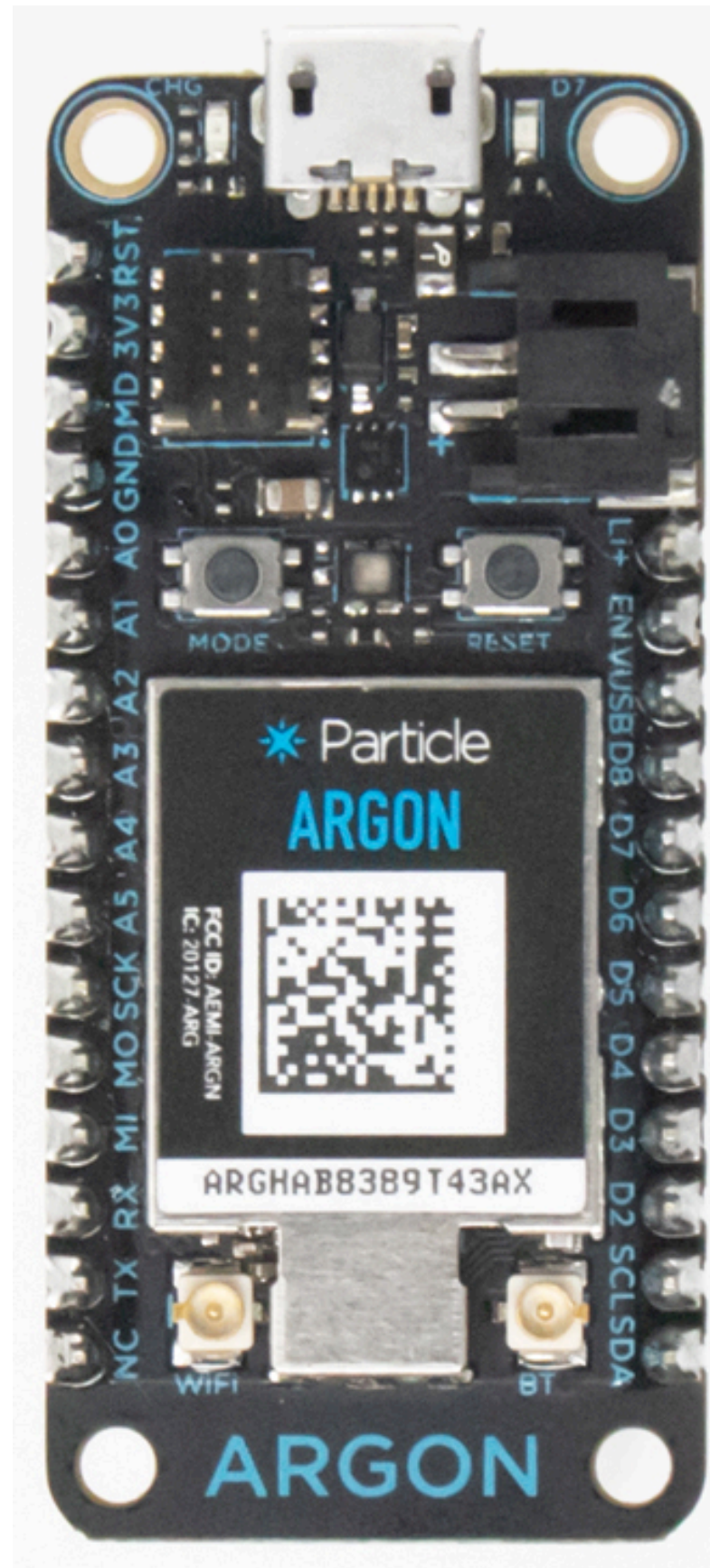


## E-Series Modules

- Castellated edges for easy PCB inclusion
- Available in mfg-ready trays



# PARTICLE 3RD GEN: MODERN DEVICES WITH BLE & MESH NETWORKING BAKED-IN



**Argon**

**Wi-Fi, BLE & Mesh**

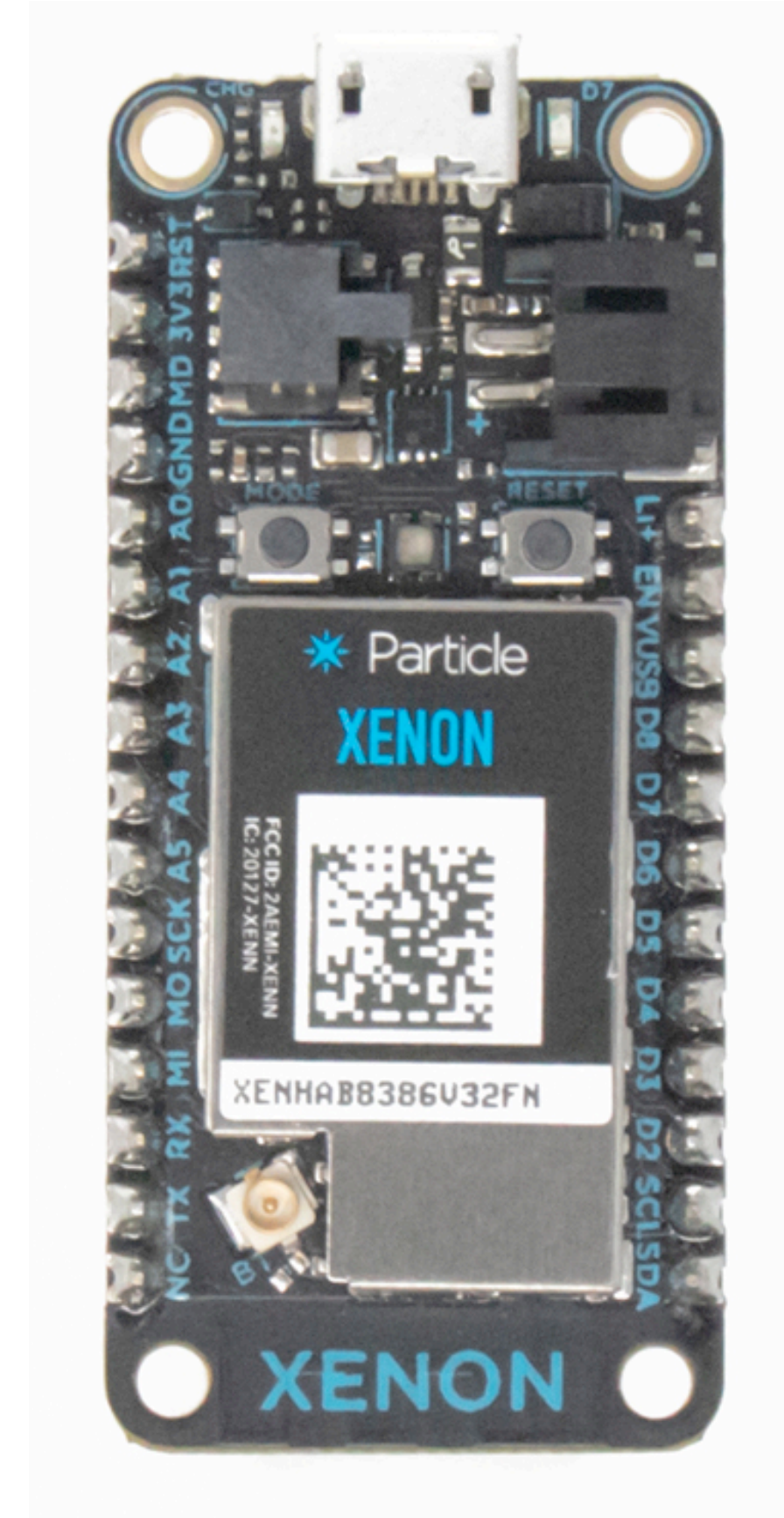
Can function as a gateway, repeater and/or endpoint



**Boron**

**LTE CAT-M1, BLE & Mesh**

Can function as a gateway, repeater and/or endpoint



**Xenon**

**BLE & Mesh**

Can function as a repeater and/or endpoint

**WHY PARTICLE**

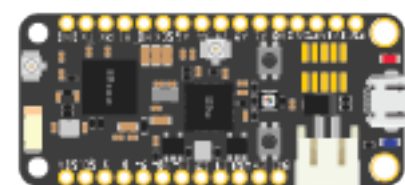
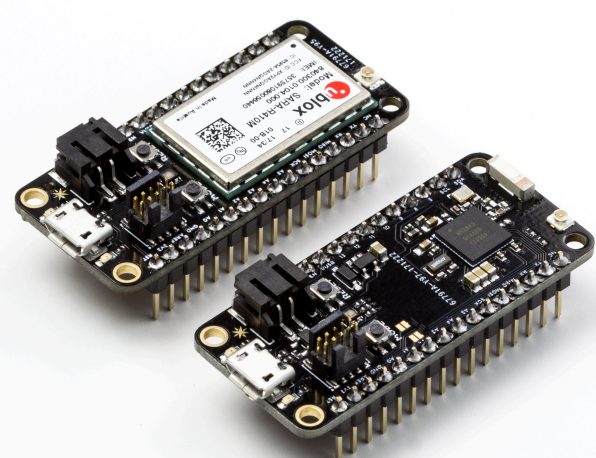
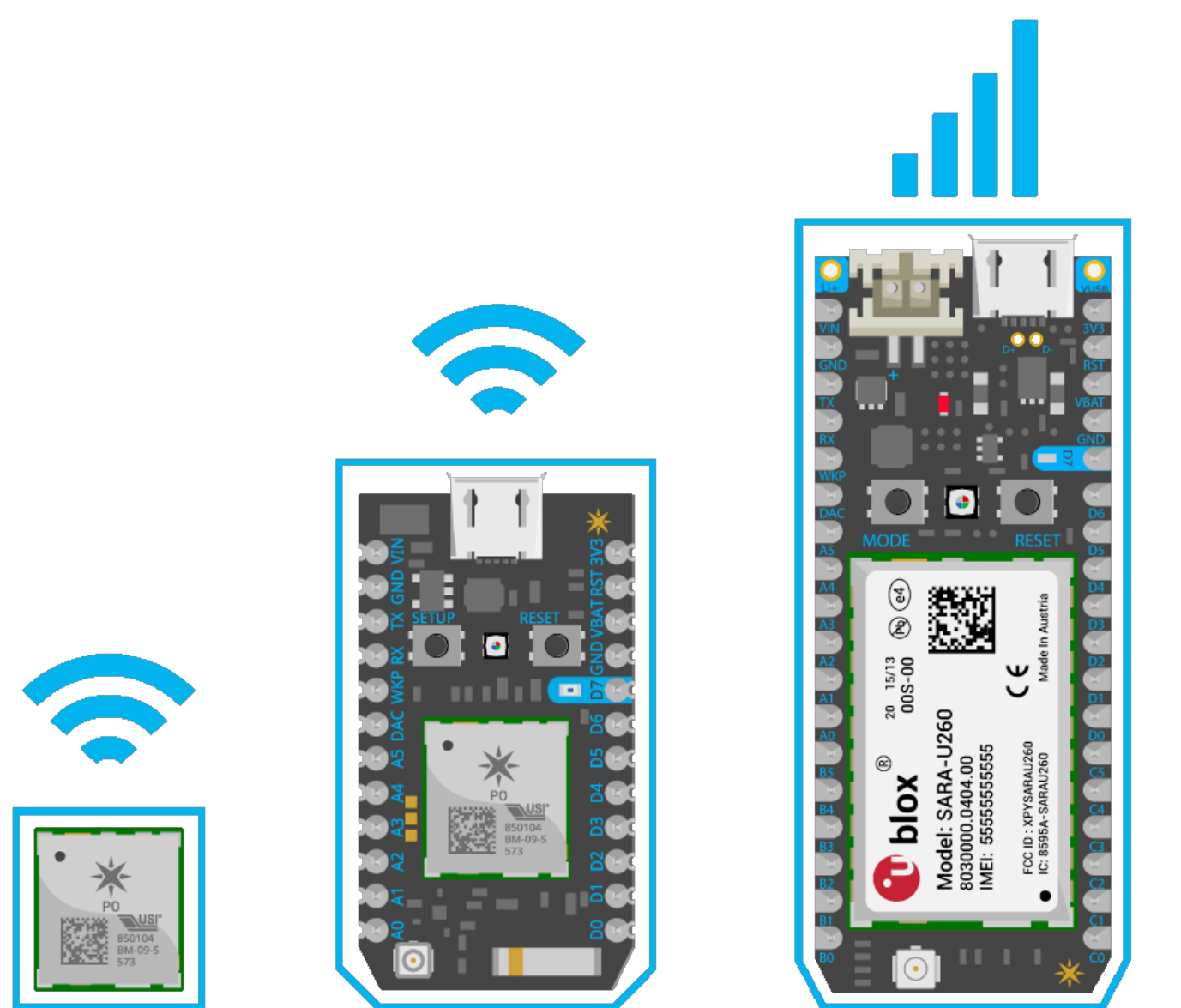
**THE PARTICLE CLOUD & FRIENDS**

**CLAIMING YOUR FIRST DEVICE**

# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

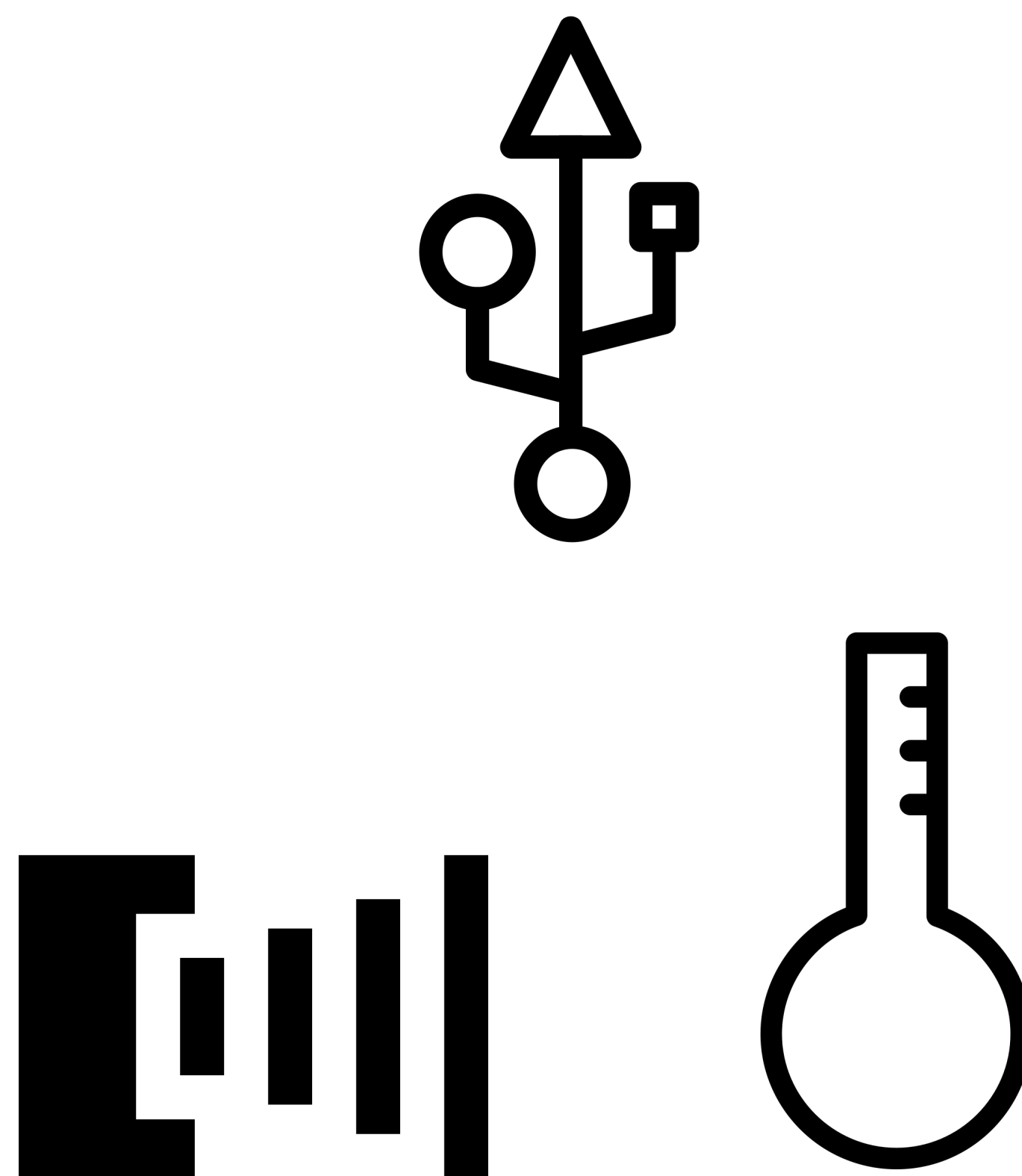
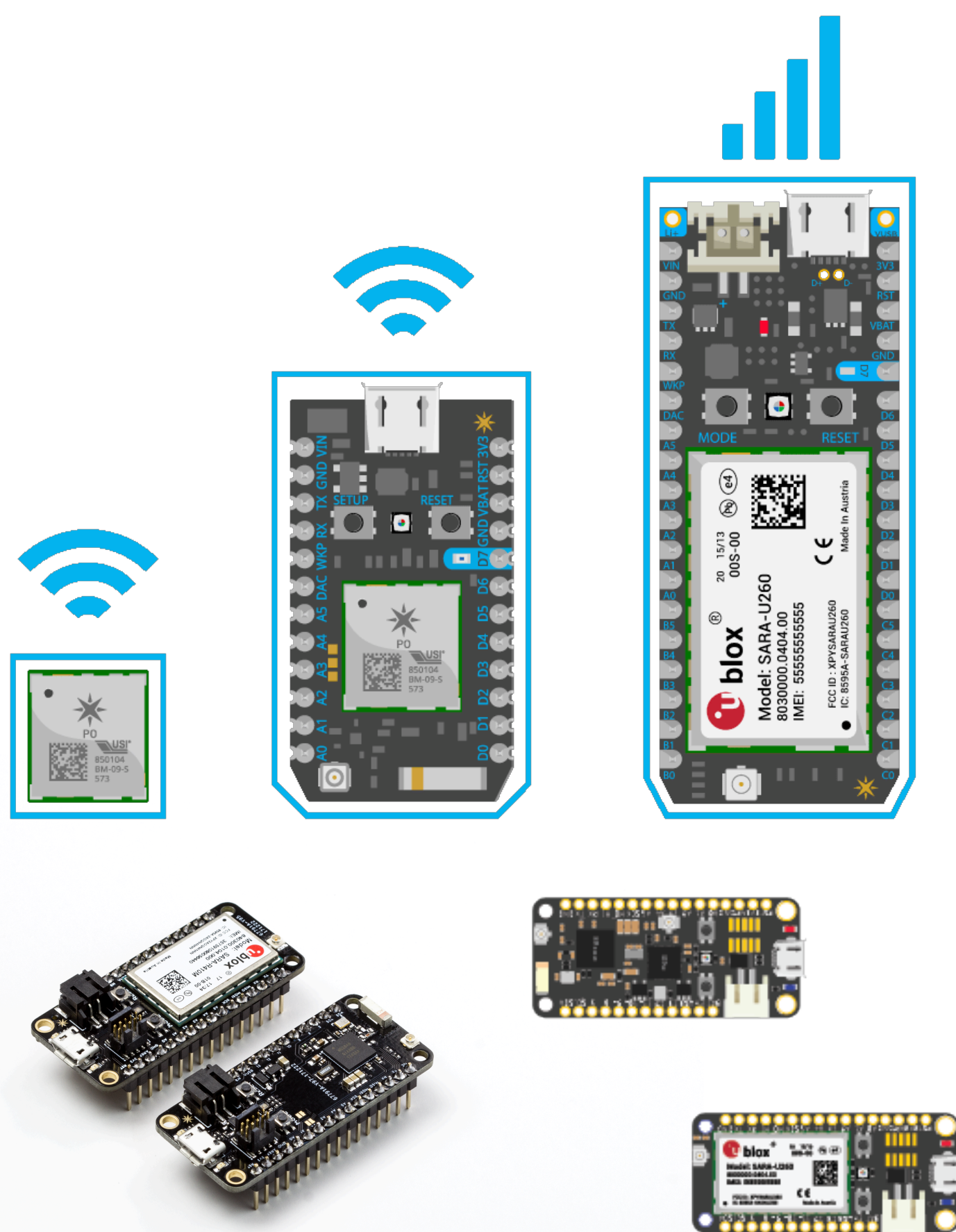
## DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION



# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

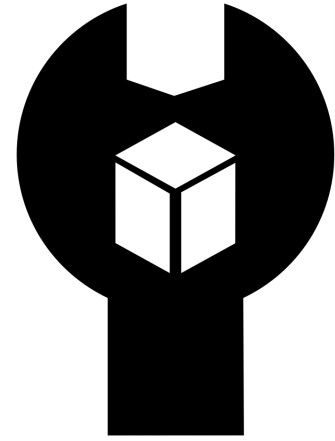
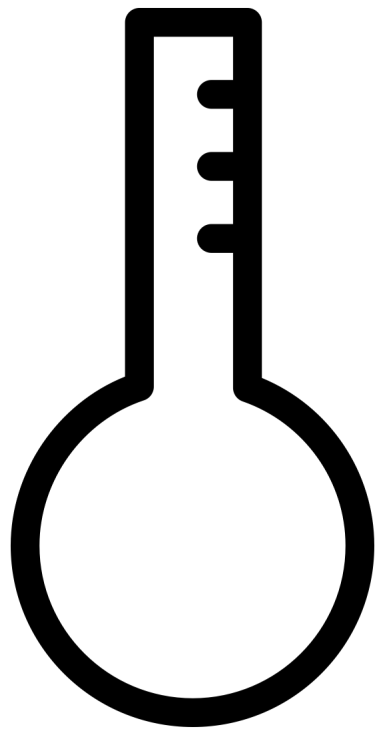
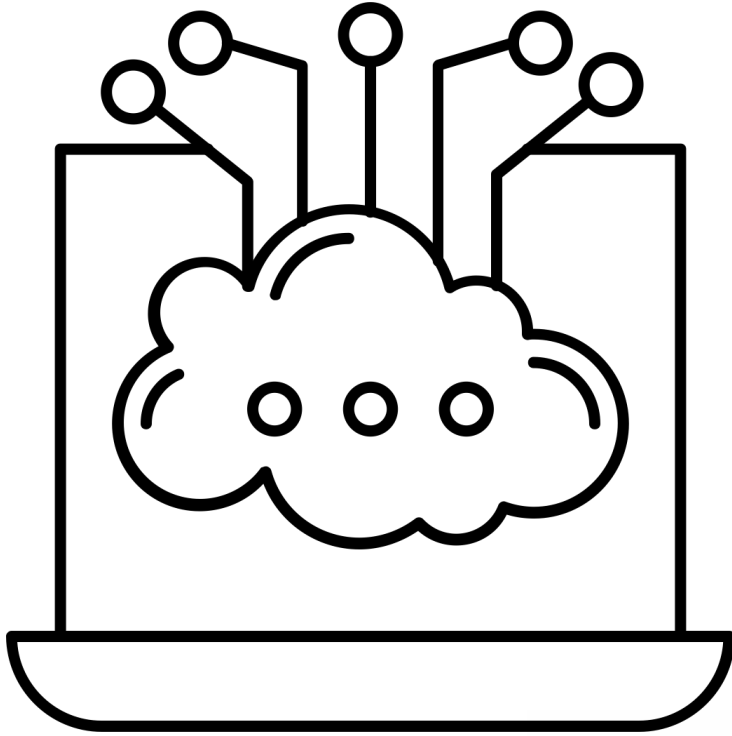
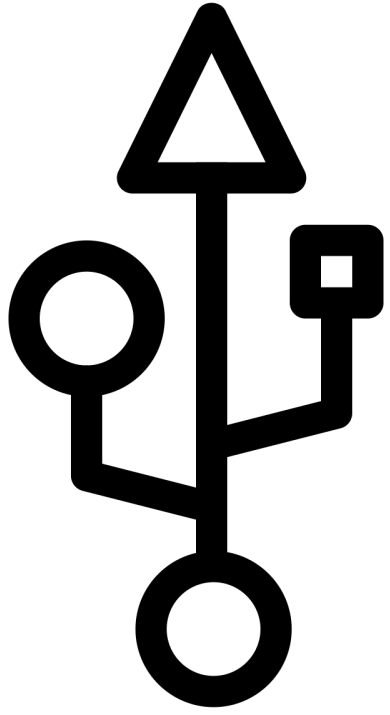
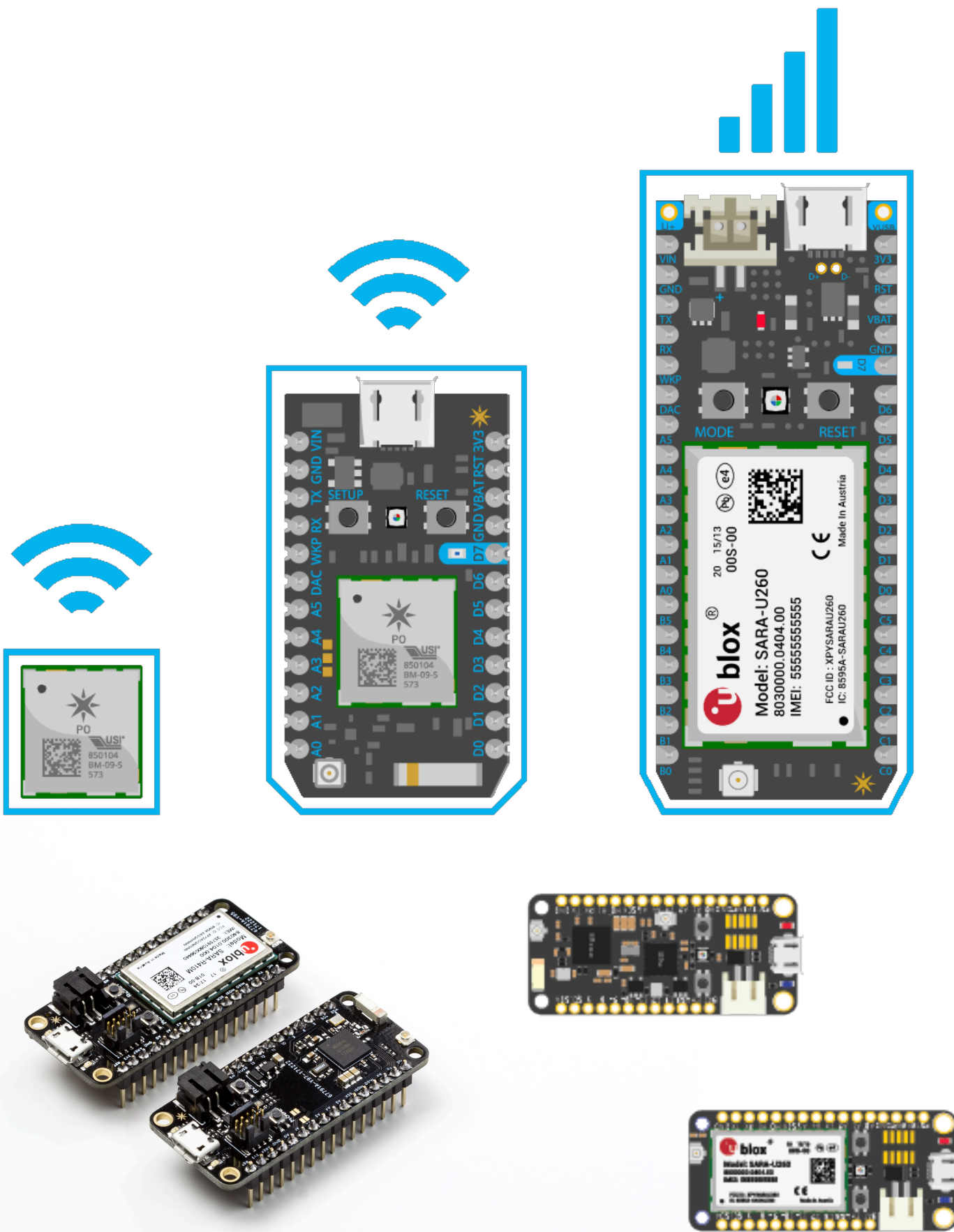


# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

DEVICE CLOUD & SOFTWARE TOOLS

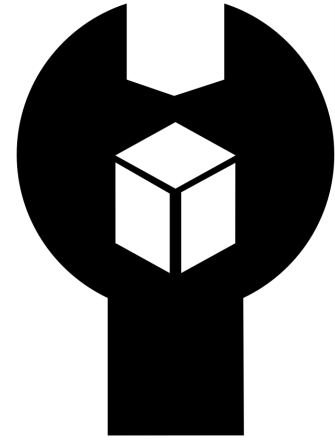
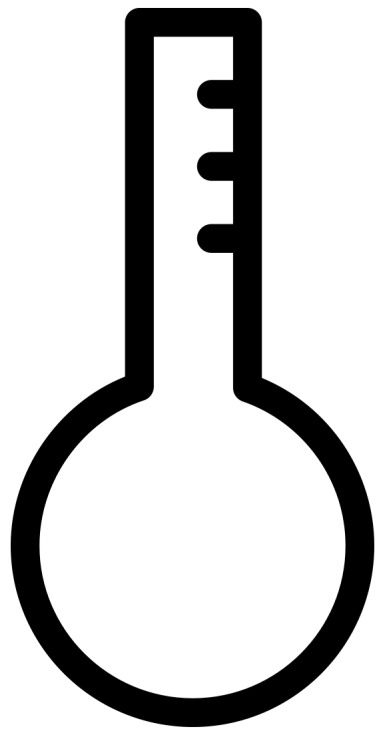
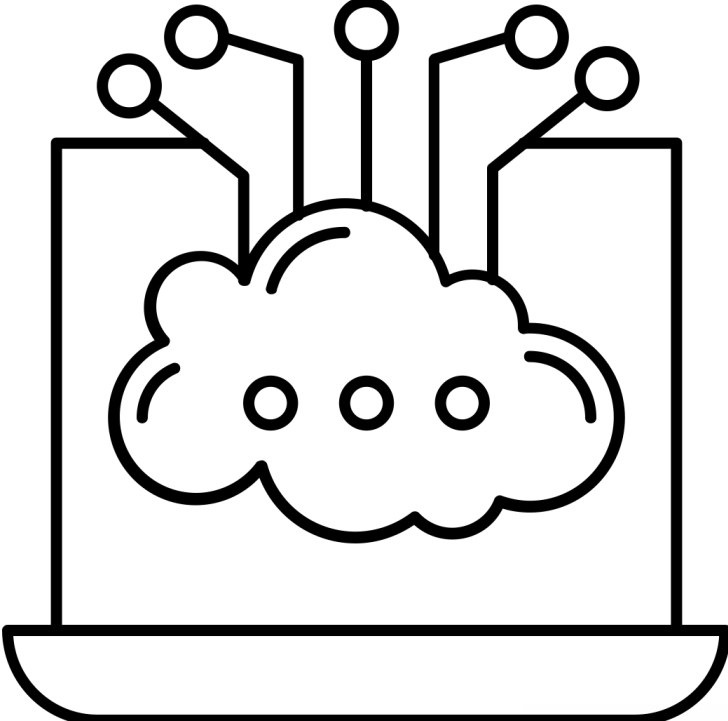
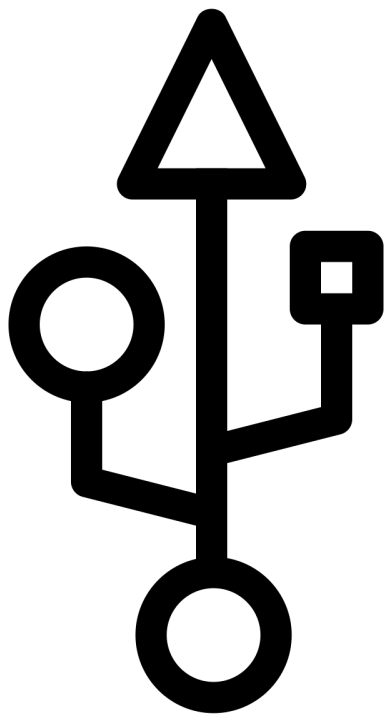
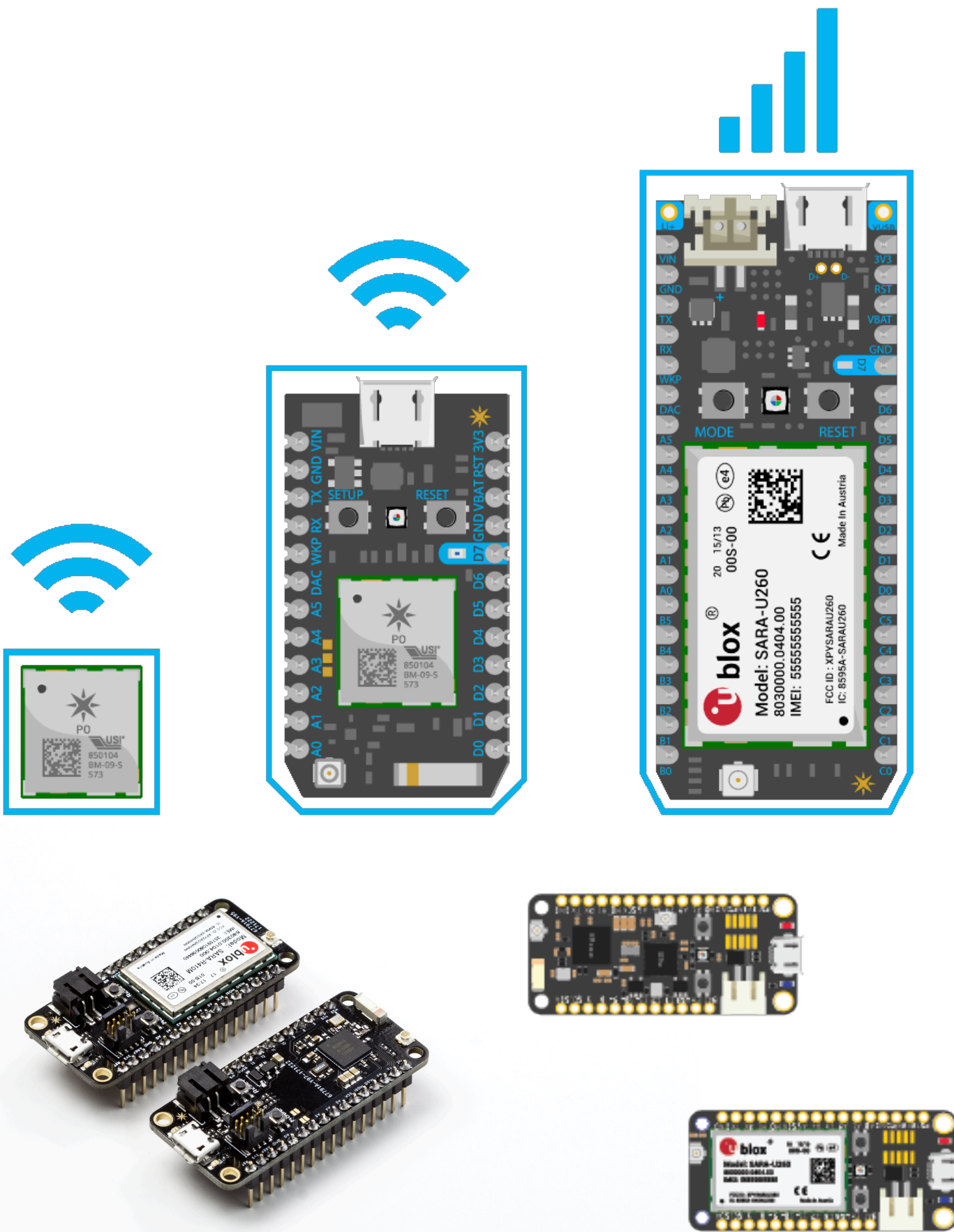


# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

DEVICE CLOUD & SOFTWARE TOOLS

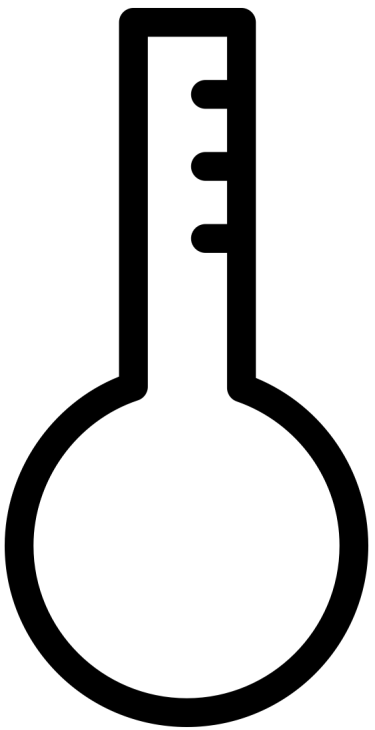
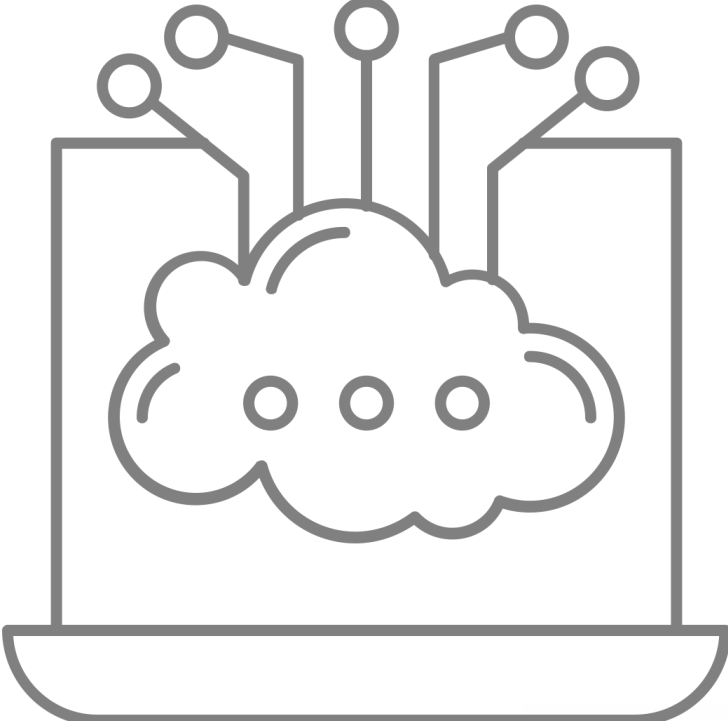
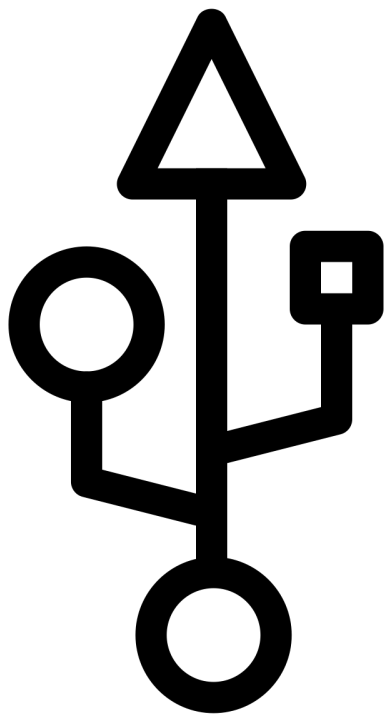
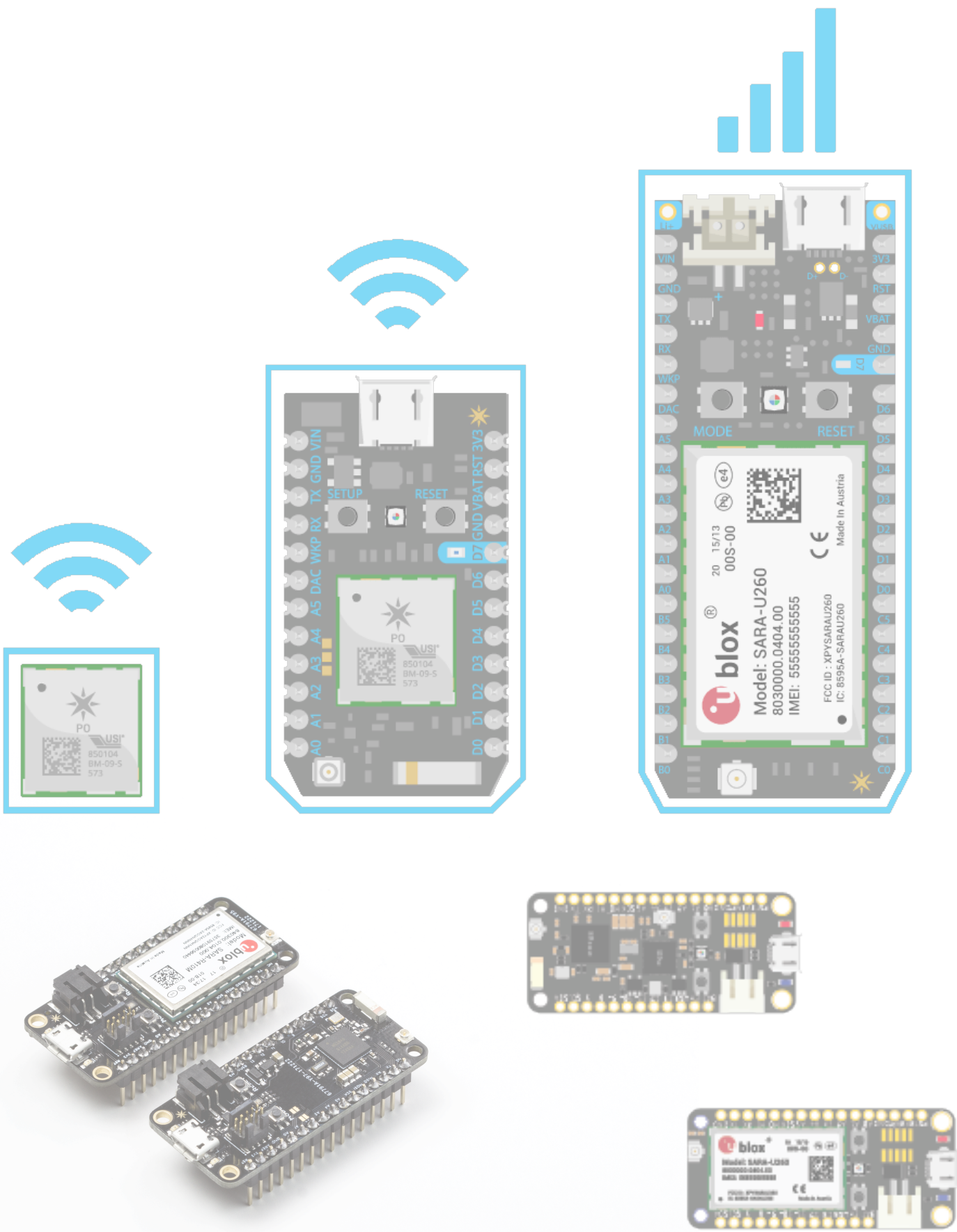


# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

DEVICE CLOUD & SOFTWARE TOOLS

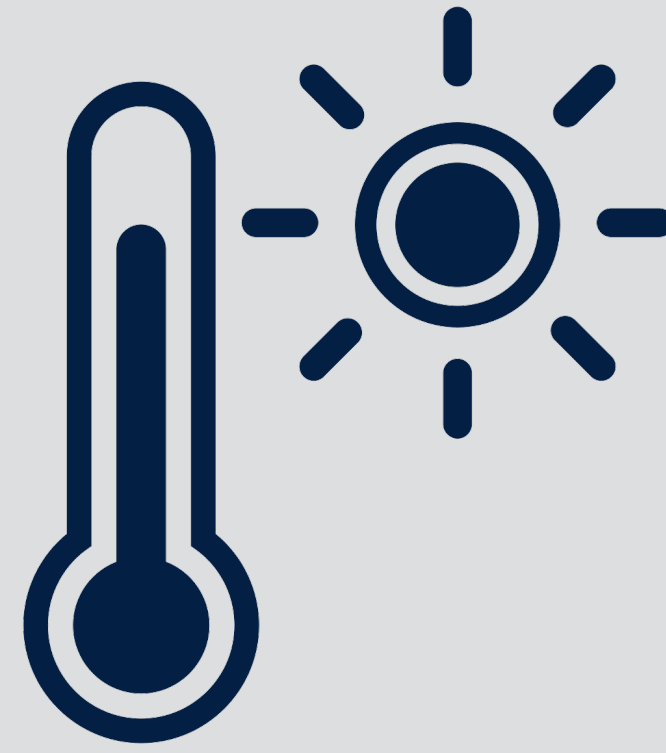




# PARTICLE DEVICE OS – POWERFUL FIRMWARE WITH TONS OF FEATURES



*Call a function, remotely*  
**Particle.function()**



*Fetch a variable, remotely*  
**Particle.variable()**



*Send an event to the cloud*  
**Particle.publish()**



*Listen for events*  
**Particle.subscribe()**

# ADVANTAGES OF USING FIRMWARE LIBRARIES

- \* Simplify interactions with sensors and actuators
- \* Reuse code across projects
- \* Leverage contributions from Particle's community of 165k developers

```
void Adafruit_SSD1306::display(void) {
  for (uint16_t i=0; i<(SSD1306_LCDWIDTH*SSD1306_LCDHEIGHT/8); i++)
  {
    // send a bunch of data in one tx-mission
    Wire.beginTransaction(_i2caddr);
    Wire.write(0x40);
    for (uint8_t x=0; x<16; x++) {
      Wire.write(buffer[i]);
      i++;
    }
    i--;
    Wire.endTransmission();
  }
}
```

# ADVANTAGES OF USING FIRMWARE LIBRARIES

- \* Simplify interactions with sensors and actuators
- \* Reuse code across projects
- \* Leverage contributions from Particle's community of 165k developers

```
void Adafruit_SSD1306::display(void) {  
  for (uint16_t i=0; i<(SSD1306_LCDWIDTH*SSD1306_LCDHEIGHT/8); i++)  
  {  
    // send a bunch of data in one tx-mission  
    Wire.beginTransaction(_i2caddr);  
    Wire.write(0x40);  
    for (uint8_t x=0; x<16; x++) {  
      Wire.write(buffer[i]);  
      i++;  
    }  
    i--;  
    Wire.endTransmission();  
  }  
}
```



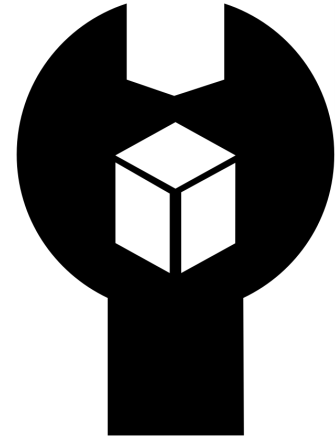
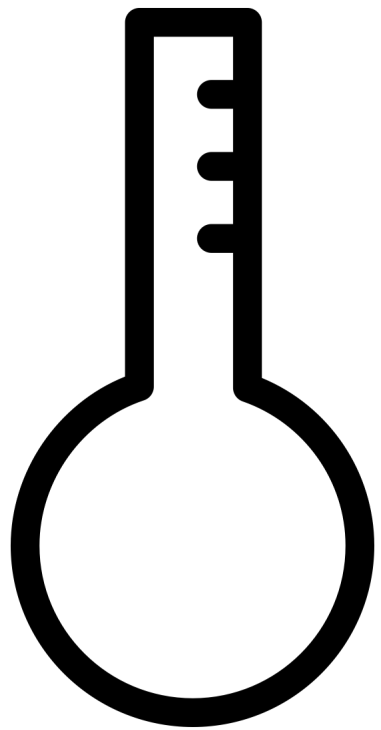
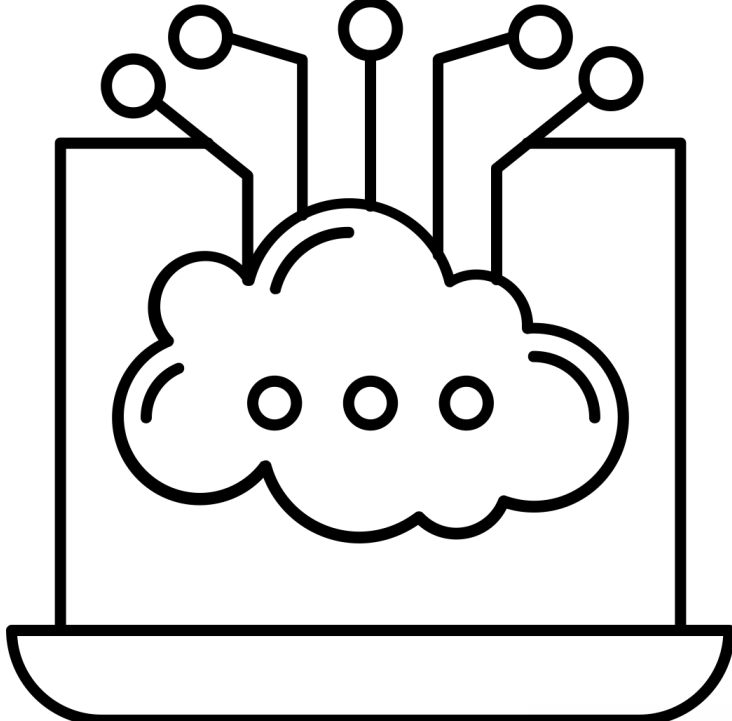
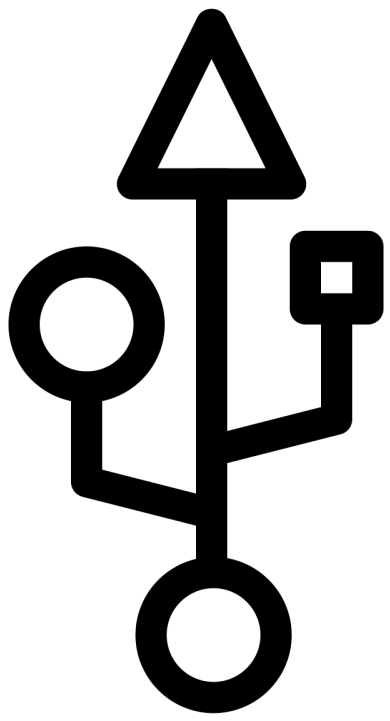
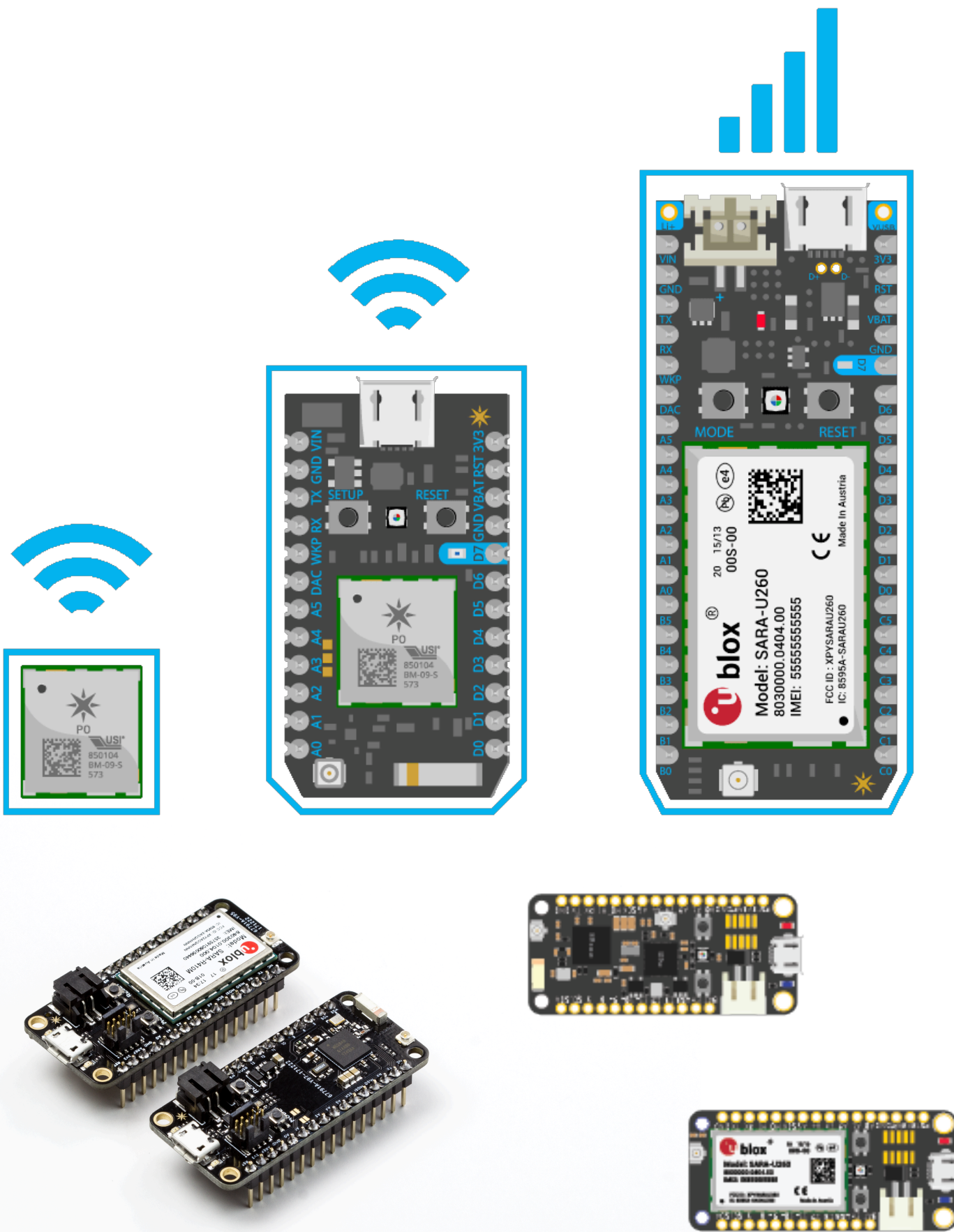
```
display.println("Temp");  
display.print((int)currentTemp);  
display.println("Humidity");  
display.print((int)currentHumidity);  
display.display();
```

# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

DEVICE CLOUD & SOFTWARE TOOLS

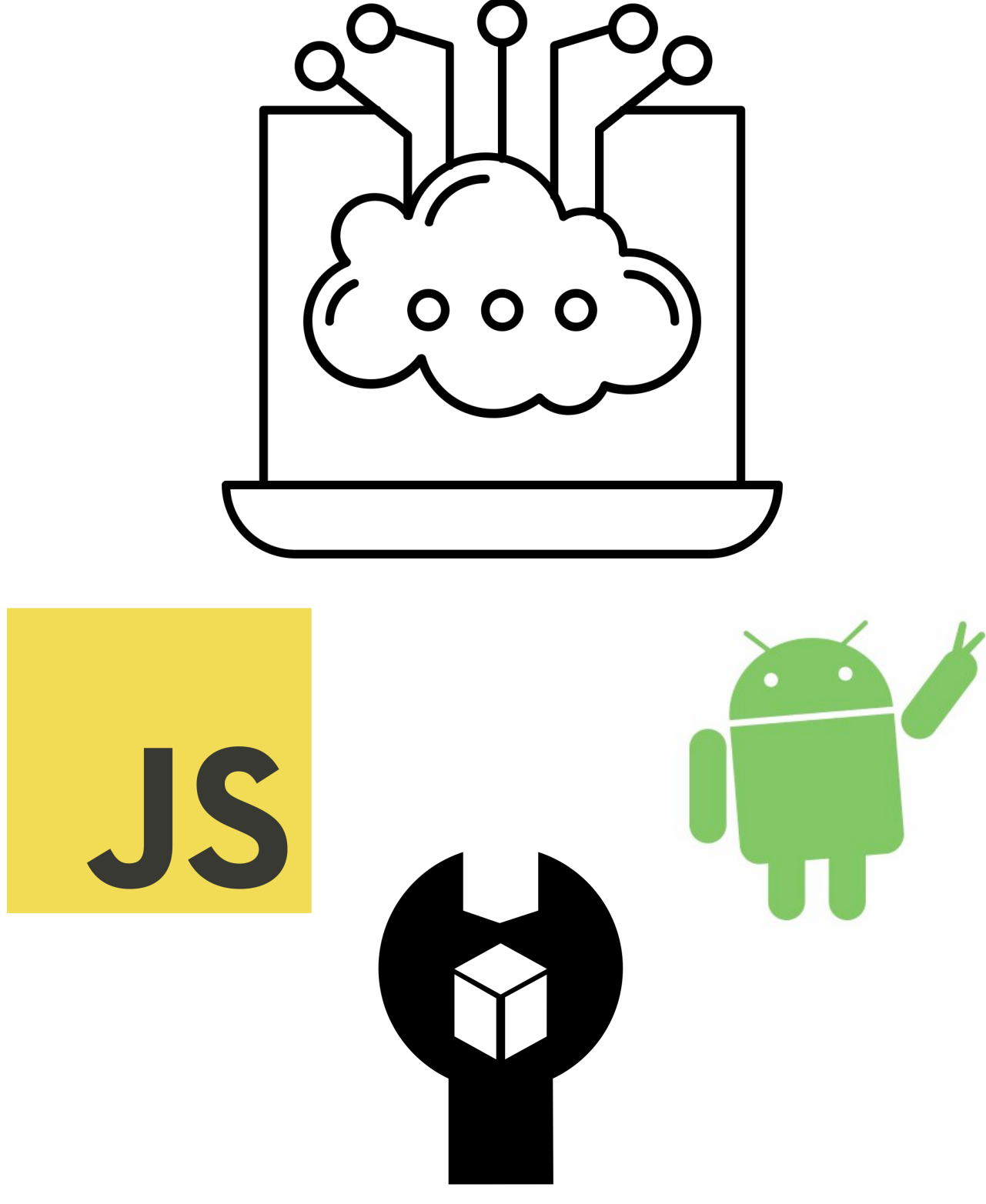
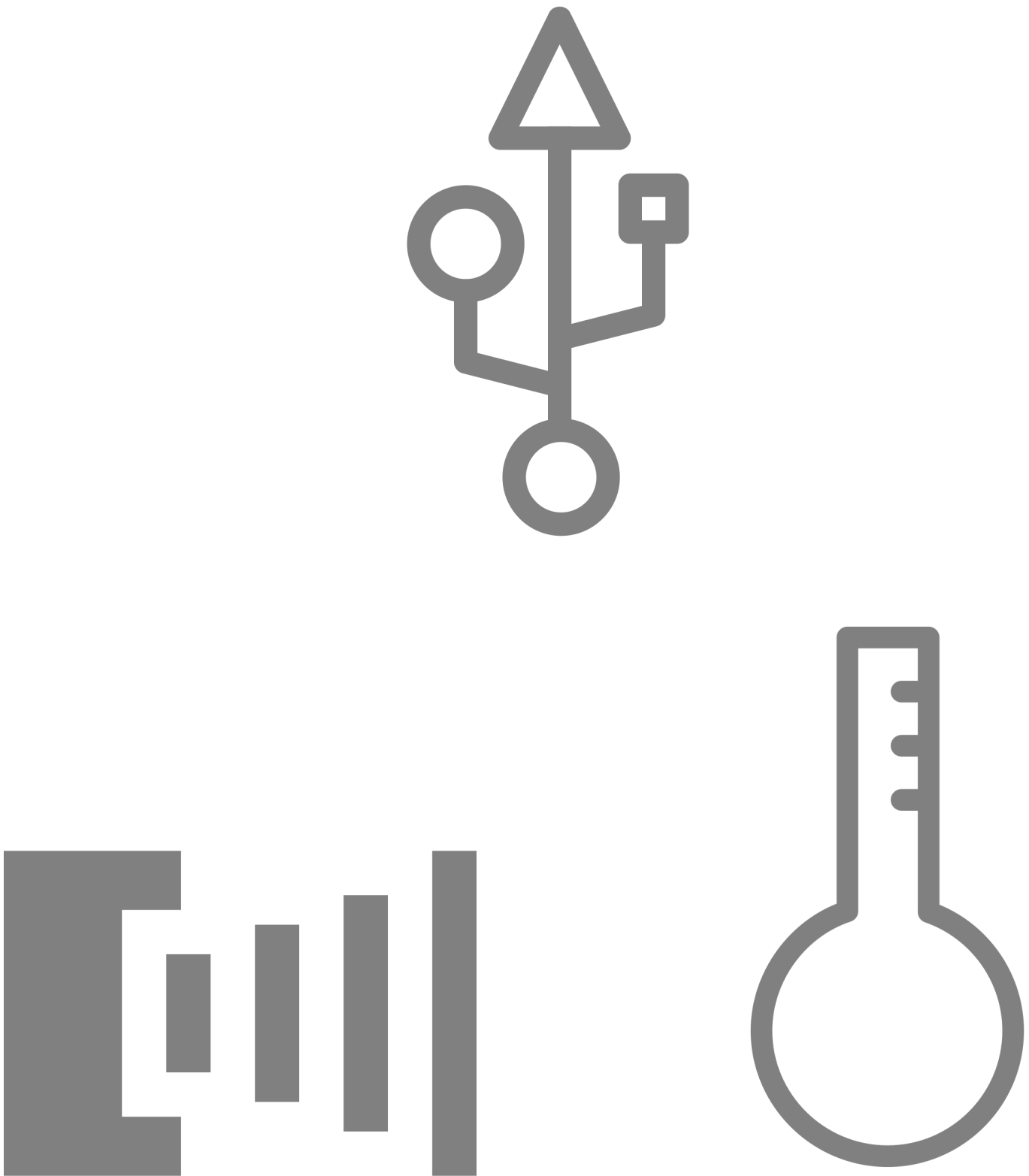
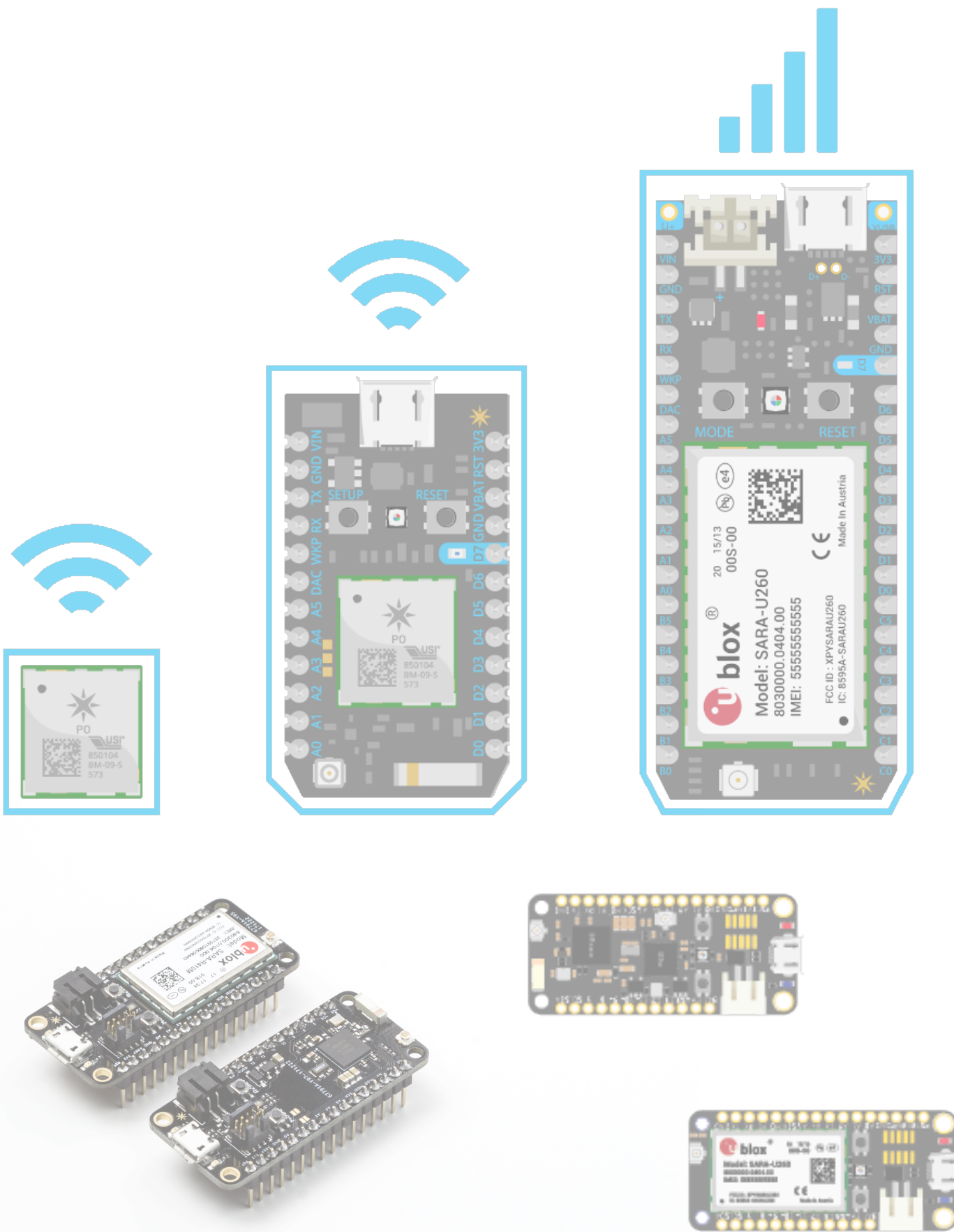


# THE PARTICLE ECOSYSTEM: HARDWARE, FIRMWARE, & SOFTWARE

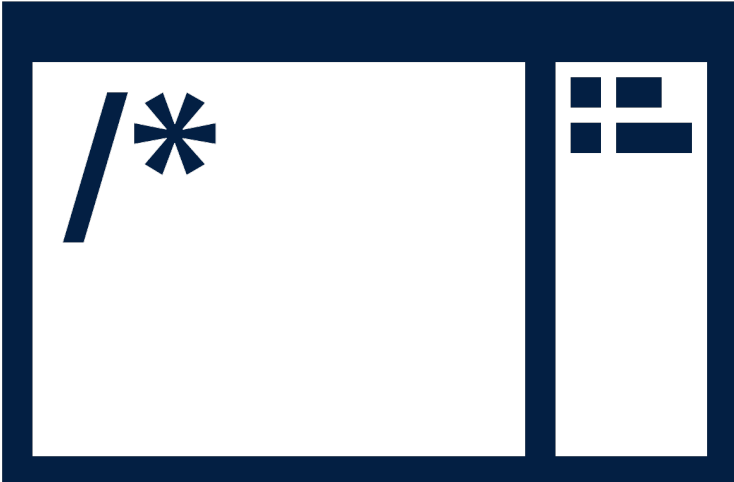
DEVICE HARDWARE FOR PROTOTYPING & PRODUCTION

DEVICE OS FIRMWARE & LIBRARIES

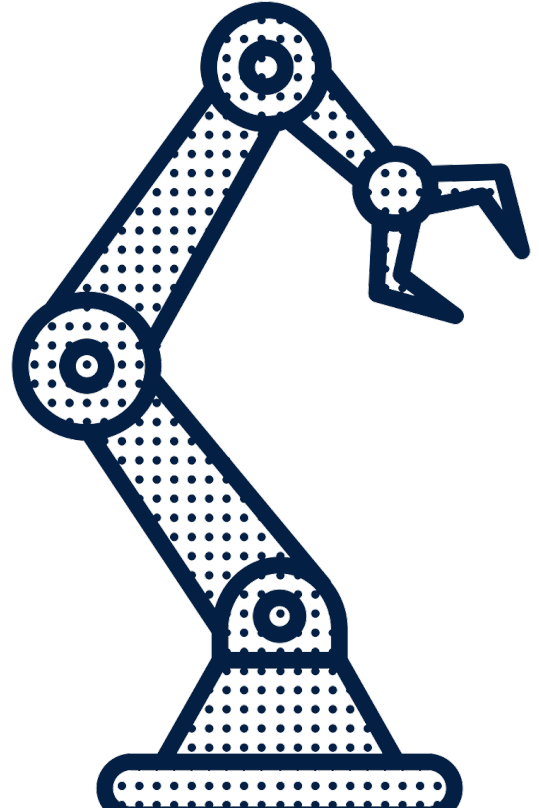
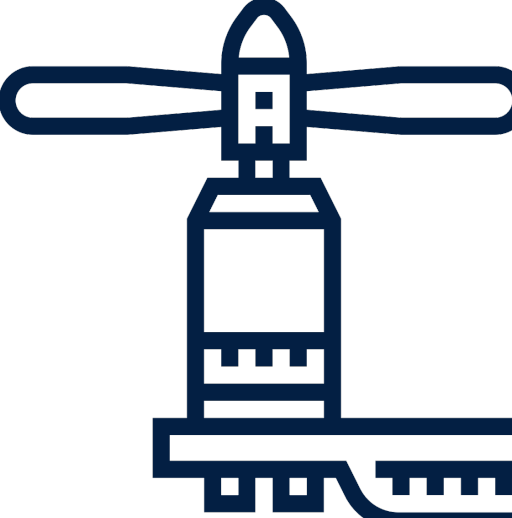
DEVICE CLOUD & SOFTWARE TOOLS



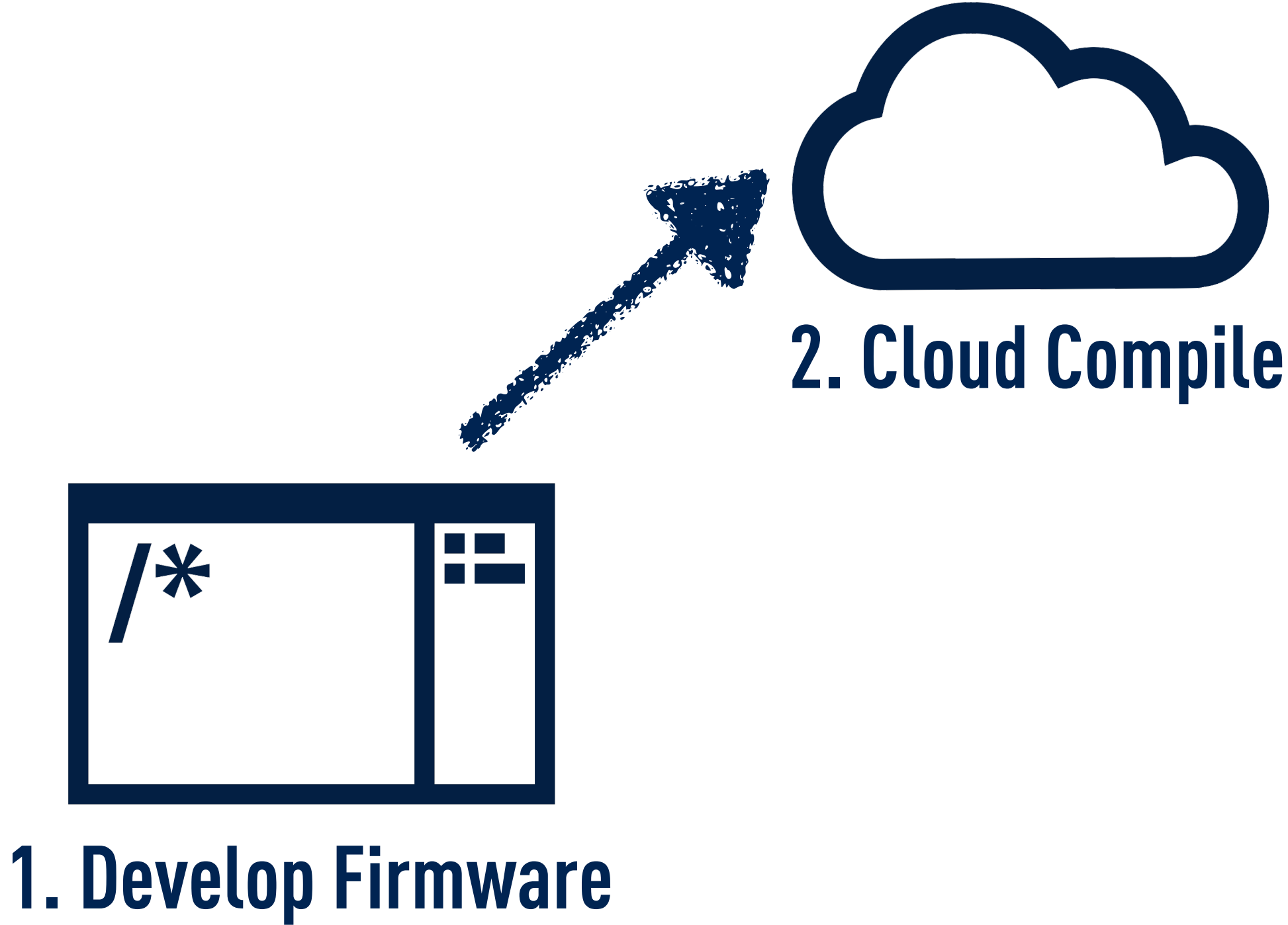
# OVER THE AIR (OTA) DEVICE UPDATES



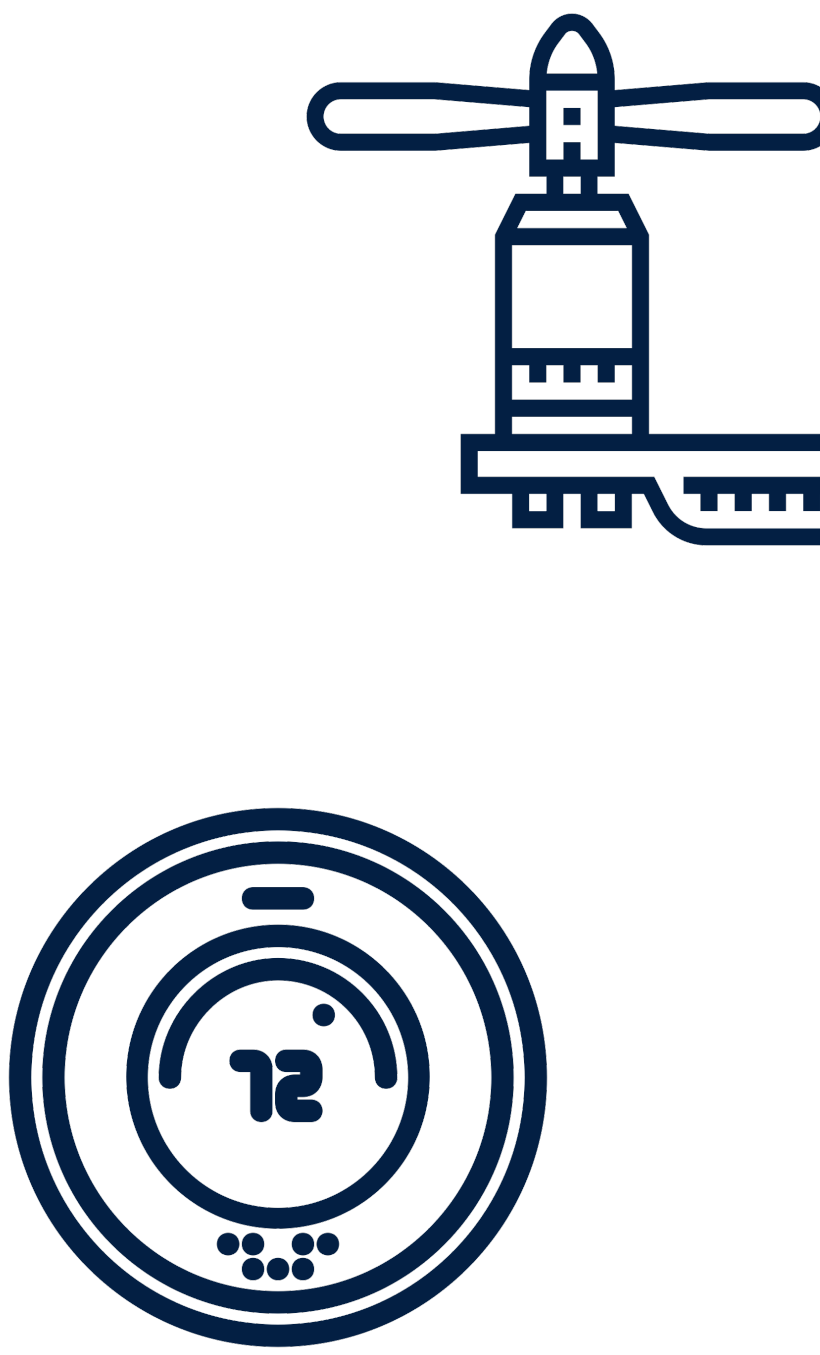
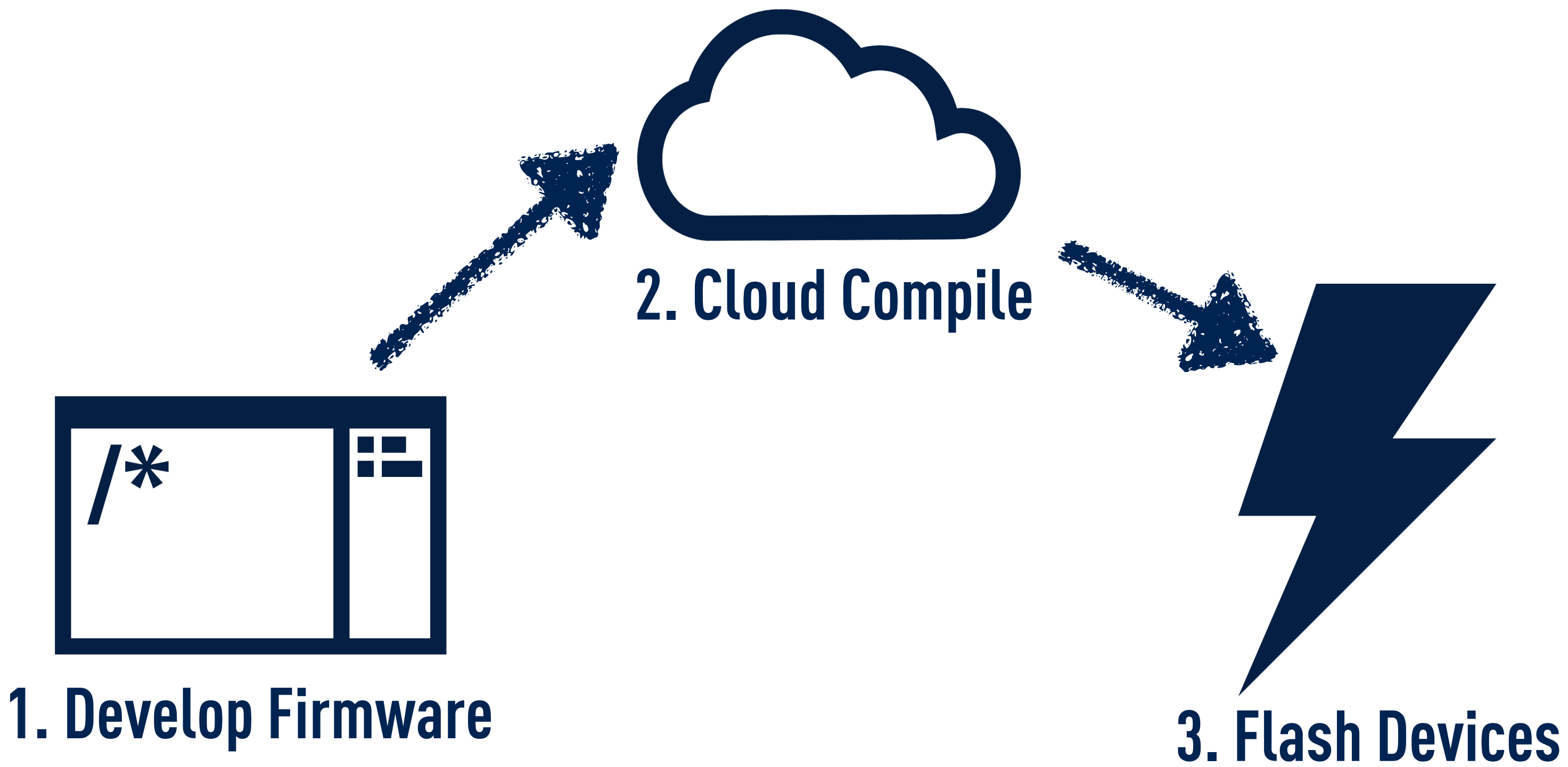
1. Develop Firmware



# OVER THE AIR (OTA) DEVICE UPDATES

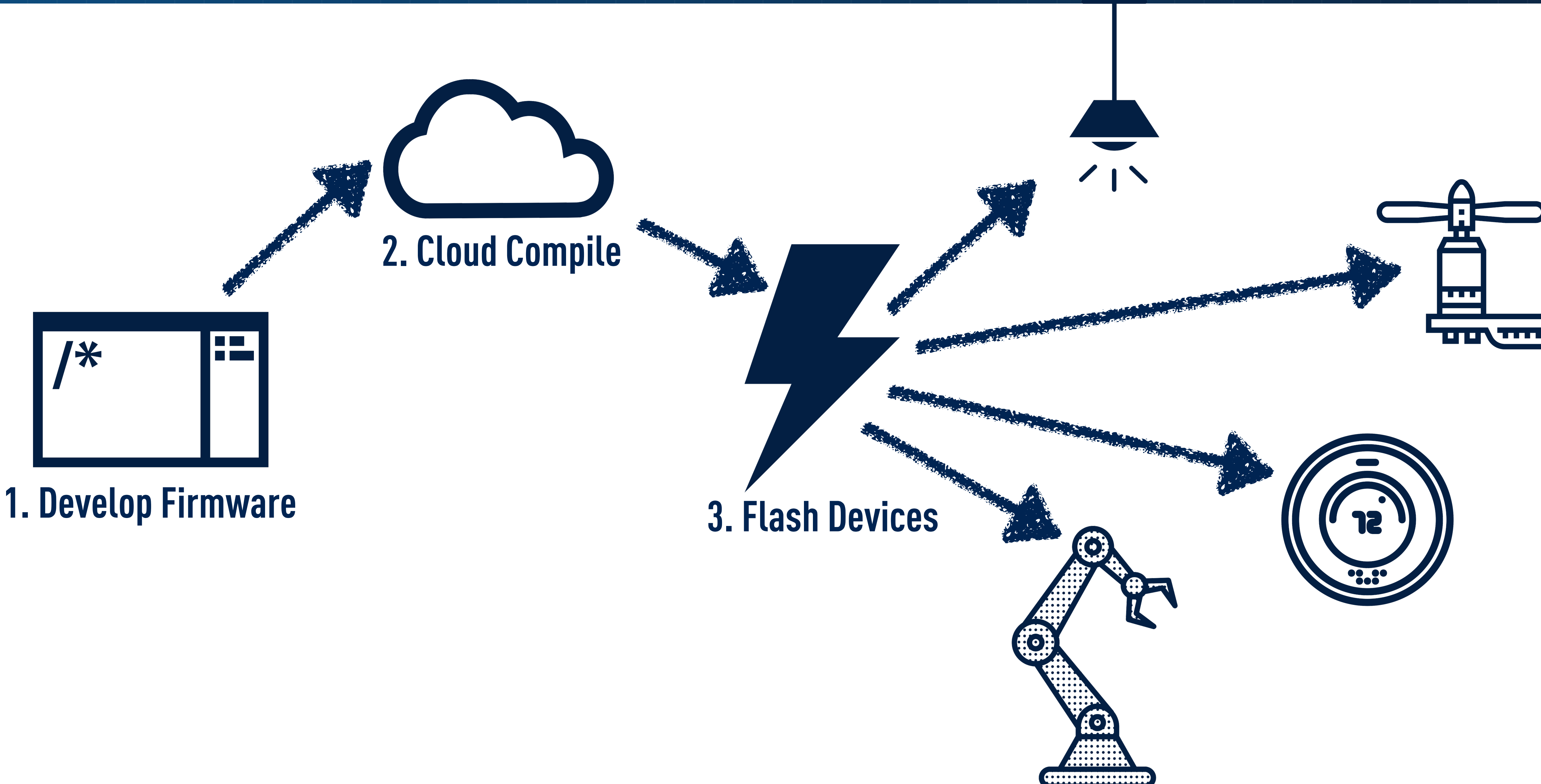


# OVER THE AIR (OTA) DEVICE UPDATES





# OVER THE AIR (OTA) DEVICE UPDATES



# IDES AND DEVELOPER TOOLING

## PARTICLE BUILD (WEB)

## PARTICLE WORKBENCH

## PARTICLE CLI

The screenshot shows the Particle Build web interface. On the left, there's a sidebar with 'Particle Apps' and 'My apps'. The main area displays a code editor with C++ code for a Tinker application. The code includes comments and function definitions for digital and analog read/write operations. Below the code, there's a 'Please enter the title to create your first app.' prompt.

```
1-/* Function prototypes -----*/
2 int tinkerDigitalRead(String pin);
3 int tinkerDigitalWrite(String command);
4 int tinkerAnalogRead(String pin);
5 int tinkerAnalogWrite(String command);
6
7-/* This function is called once at start up -----*/
8 void setup()
9 {
10 //Setup the Tinker application here
11
12 //Register all the Tinker functions
13 Particle.function("digitalread", tinkerDigitalRead);
14 Particle.function("digitalwrite", tinkerDigitalWrite);
15 Particle.function("analogread", tinkerAnalogRead);
16 Particle.function("analogwrite", tinkerAnalogWrite);
17
18 }
19
20-/* This function loops forever -----*/
21 void loop()
22 {
23 //This will run in a loop
24 }
25
26-----
27 * Function Name : tinkerDigitalRead
28 * Description : Reads the digital value of a given pin
29 * Input : Pin
30 * Output : None
31 * Return : Value of the pin (0 or 1) in INT type
32 Returns a negative number on failure
33-----
34 int tinkerDigitalRead(String pin)
35 {
36 //convert ascii to integer
37 int pinNumber = pin.charAt(1) - '0';
38 //Sanity check to see if the pin numbers are within limits
39 if (pinNumber < 0 || pinNumber > 7) return -1;
40
41 if (pin.startsWith("D"))
42 {
43 pinMode(pinNumber, INPUT_PULLDOWN);
44 return digitalRead(pinNumber);
45 }
46 else if (pin.startsWith("A"))
47 {
48 pinMode(pinNumber, INPUT_PULLDOWN);
49 return digitalRead(pinNumber);
50 }
51 return -2;
52 }
53
54-----
55 * Function Name : tinkerDigitalWrite
56 * Description : Sets the specified pin HIGH or LOW
57 * Input : Pin and value
58 * Output : None
59 * Return : 1 on success and a negative number on failure
60-----
61 int tinkerDigitalWrite(String command)
62 {
63 bool value = 0;
64 //convert ascii to integer
65 int pinNumber = command.charAt(1) - '0';
66 //Sanity check to see if the pin numbers are within limits
67 if (pinNumber < 0 || pinNumber > 7) return -1;
68
69 if (command.substring(3,7) == "HIGH") value = 1;
70 else if (command.substring(3,7) == "LOW") value = 0;
71 else return -2;
72
73 if (command.startsWith("D"))
74 {
75 pinMode(pinNumber, OUTPUT);
76 digitalWrite(pinNumber, value);
77 return 1;
78 }
79
80 else if (command.startsWith("A"))
81 {
82 pinMode(pinNumber, OUTPUT);
83 digitalWrite(pinNumber, value);
84 return 1;
85 }
86 }
87
88-----
89 * Function Name : tinkerAnalogRead
90 * Description : Reads the analog value of a given pin
91 * Input : Pin
92 * Output : None
93 * Return : Value of the pin (0 or 1) in INT type
94 Returns a negative number on failure
95-----
96 int tinkerAnalogRead(String pin)
97 {
98 //convert ascii to integer
99 int pinNumber = pin.charAt(1) - '0';
100 //Sanity check to see if the pin numbers are within limits
101 if (pinNumber < 0 || pinNumber > 7) return -1;
102
103 if (pin.startsWith("D"))
104 {
105 pinMode(pinNumber, INPUT_PULLDOWN);
106 return digitalRead(pinNumber);
107 }
108 else if (pin.startsWith("A"))
109 {
110 pinMode(pinNumber, INPUT_PULLDOWN);
111 return digitalRead(pinNumber);
112 }
113 return -2;
114 }
115
116-----
117 * Function Name : tinkerAnalogWrite
118 * Description : Sets the specified pin HIGH or LOW
119 * Input : Pin and value
120 * Output : None
121 * Return : 1 on success and a negative number on failure
122-----
123 int tinkerAnalogWrite(String command)
124 {
125 bool value = 0;
126 //convert ascii to integer
127 int pinNumber = command.charAt(1) - '0';
128 //Sanity check to see if the pin numbers are within limits
129 if (pinNumber < 0 || pinNumber > 7) return -1;
130
131 if (command.substring(3,7) == "HIGH") value = 1;
132 else if (command.substring(3,7) == "LOW") value = 0;
133 else return -2;
134
135 if (command.startsWith("D"))
136 {
137 pinMode(pinNumber, OUTPUT);
138 digitalWrite(pinNumber, value);
139 return 1;
140 }
141
142 else if (command.startsWith("A"))
143 {
144 pinMode(pinNumber, OUTPUT);
145 digitalWrite(pinNumber, value);
146 return 1;
147 }
148 }
149
150-----
151 * Function Name : tinkerAnalogWrite
152 * Description : Sets the specified pin HIGH or LOW
153 * Input : Pin and value
154 * Output : None
155 * Return : 1 on success and a negative number on failure
156-----
157 int tinkerAnalogWrite(String command)
158 {
159 bool value = 0;
160 //convert ascii to integer
161 int pinNumber = command.charAt(1) - '0';
162 //Sanity check to see if the pin numbers are within limits
163 if (pinNumber < 0 || pinNumber > 7) return -1;
164
165 if (command.substring(3,7) == "HIGH") value = 1;
166 else if (command.substring(3,7) == "LOW") value = 0;
167 else return -2;
168
169 if (command.startsWith("D"))
170 {
171 pinMode(pinNumber, OUTPUT);
172 digitalWrite(pinNumber, value);
173 return 1;
174 }
175
176 else if (command.startsWith("A"))
177 {
178 pinMode(pinNumber, OUTPUT);
179 digitalWrite(pinNumber, value);
180 return 1;
181 }
182 }
183
184-----
185 * Function Name : tinkerAnalogWrite
186 * Description : Sets the specified pin HIGH or LOW
187 * Input : Pin and value
188 * Output : None
189 * Return : 1 on success and a negative number on failure
190-----
191 int tinkerAnalogWrite(String command)
192 {
193 bool value = 0;
194 //convert ascii to integer
195 int pinNumber = command.charAt(1) - '0';
196 //Sanity check to see if the pin numbers are within limits
197 if (pinNumber < 0 || pinNumber > 7) return -1;
198
199 if (command.substring(3,7) == "HIGH") value = 1;
200 else if (command.substring(3,7) == "LOW") value = 0;
201 else return -2;
202
203 if (command.startsWith("D"))
204 {
205 pinMode(pinNumber, OUTPUT);
206 digitalWrite(pinNumber, value);
207 return 1;
208 }
209
210 else if (command.startsWith("A"))
211 {
212 pinMode(pinNumber, OUTPUT);
213 digitalWrite(pinNumber, value);
214 return 1;
215 }
216 }
217
218-----
219 * Function Name : tinkerAnalogWrite
220 * Description : Sets the specified pin HIGH or LOW
221 * Input : Pin and value
222 * Output : None
223 * Return : 1 on success and a negative number on failure
224-----
225 int tinkerAnalogWrite(String command)
226 {
227 bool value = 0;
228 //convert ascii to integer
229 int pinNumber = command.charAt(1) - '0';
230 //Sanity check to see if the pin numbers are within limits
231 if (pinNumber < 0 || pinNumber > 7) return -1;
232
233 if (command.substring(3,7) == "HIGH") value = 1;
234 else if (command.substring(3,7) == "LOW") value = 0;
235 else return -2;
236
237 if (command.startsWith("D"))
238 {
239 pinMode(pinNumber, OUTPUT);
240 digitalWrite(pinNumber, value);
241 return 1;
242 }
243
244 else if (command.startsWith("A"))
245 {
246 pinMode(pinNumber, OUTPUT);
247 digitalWrite(pinNumber, value);
248 return 1;
249 }
250 }
251
252-----
253 * Function Name : tinkerAnalogWrite
254 * Description : Sets the specified pin HIGH or LOW
255 * Input : Pin and value
256 * Output : None
257 * Return : 1 on success and a negative number on failure
258-----
259 int tinkerAnalogWrite(String command)
260 {
261 bool value = 0;
262 //convert ascii to integer
263 int pinNumber = command.charAt(1) - '0';
264 //Sanity check to see if the pin numbers are within limits
265 if (pinNumber < 0 || pinNumber > 7) return -1;
266
267 if (command.substring(3,7) == "HIGH") value = 1;
268 else if (command.substring(3,7) == "LOW") value = 0;
269 else return -2;
270
271 if (command.startsWith("D"))
272 {
273 pinMode(pinNumber, OUTPUT);
274 digitalWrite(pinNumber, value);
275 return 1;
276 }
277
278 else if (command.startsWith("A"))
279 {
280 pinMode(pinNumber, OUTPUT);
281 digitalWrite(pinNumber, value);
282 return 1;
283 }
284 }
285
286-----
287 * Function Name : tinkerAnalogWrite
288 * Description : Sets the specified pin HIGH or LOW
289 * Input : Pin and value
290 * Output : None
291 * Return : 1 on success and a negative number on failure
292-----
293 int tinkerAnalogWrite(String command)
294 {
295 bool value = 0;
296 //convert ascii to integer
297 int pinNumber = command.charAt(1) - '0';
298 //Sanity check to see if the pin numbers are within limits
299 if (pinNumber < 0 || pinNumber > 7) return -1;
300
301 if (command.substring(3,7) == "HIGH") value = 1;
302 else if (command.substring(3,7) == "LOW") value = 0;
303 else return -2;
304
305 if (command.startsWith("D"))
306 {
307 pinMode(pinNumber, OUTPUT);
308 digitalWrite(pinNumber, value);
309 return 1;
310 }
311
312 else if (command.startsWith("A"))
313 {
314 pinMode(pinNumber, OUTPUT);
315 digitalWrite(pinNumber, value);
316 return 1;
317 }
318 }
319
320-----
321 * Function Name : tinkerAnalogWrite
322 * Description : Sets the specified pin HIGH or LOW
323 * Input : Pin and value
324 * Output : None
325 * Return : 1 on success and a negative number on failure
326-----
327 int tinkerAnalogWrite(String command)
328 {
329 bool value = 0;
330 //convert ascii to integer
331 int pinNumber = command.charAt(1) - '0';
332 //Sanity check to see if the pin numbers are within limits
333 if (pinNumber < 0 || pinNumber > 7) return -1;
334
335 if (command.substring(3,7) == "HIGH") value = 1;
336 else if (command.substring(3,7) == "LOW") value = 0;
337 else return -2;
338
339 if (command.startsWith("D"))
340 {
341 pinMode(pinNumber, OUTPUT);
342 digitalWrite(pinNumber, value);
343 return 1;
344 }
345
346 else if (command.startsWith("A"))
347 {
348 pinMode(pinNumber, OUTPUT);
349 digitalWrite(pinNumber, value);
350 return 1;
351 }
352 }
353
354-----
355 * Function Name : tinkerAnalogWrite
356 * Description : Sets the specified pin HIGH or LOW
357 * Input : Pin and value
358 * Output : None
359 * Return : 1 on success and a negative number on failure
360-----
361 int tinkerAnalogWrite(String command)
362 {
363 bool value = 0;
364 //convert ascii to integer
365 int pinNumber = command.charAt(1) - '0';
366 //Sanity check to see if the pin numbers are within limits
367 if (pinNumber < 0 || pinNumber > 7) return -1;
368
369 if (command.substring(3,7) == "HIGH") value = 1;
370 else if (command.substring(3,7) == "LOW") value = 0;
371 else return -2;
372
373 if (command.startsWith("D"))
374 {
375 pinMode(pinNumber, OUTPUT);
376 digitalWrite(pinNumber, value);
377 return 1;
378 }
379
380 else if (command.startsWith("A"))
381 {
382 pinMode(pinNumber, OUTPUT);
383 digitalWrite(pinNumber, value);
384 return 1;
385 }
386 }
387
388-----
389 * Function Name : tinkerAnalogWrite
390 * Description : Sets the specified pin HIGH or LOW
391 * Input : Pin and value
392 * Output : None
393 * Return : 1 on success and a negative number on failure
394-----
395 int tinkerAnalogWrite(String command)
396 {
397 bool value = 0;
398 //convert ascii to integer
399 int pinNumber = command.charAt(1) - '0';
400 //Sanity check to see if the pin numbers are within limits
401 if (pinNumber < 0 || pinNumber > 7) return -1;
402
403 if (command.substring(3,7) == "HIGH") value = 1;
404 else if (command.substring(3,7) == "LOW") value = 0;
405 else return -2;
406
407 if (command.startsWith("D"))
408 {
409 pinMode(pinNumber, OUTPUT);
410 digitalWrite(pinNumber, value);
411 return 1;
412 }
413
414 else if (command.startsWith("A"))
415 {
416 pinMode(pinNumber, OUTPUT);
417 digitalWrite(pinNumber, value);
418 return 1;
419 }
420 }
421
422-----
423 * Function Name : tinkerAnalogWrite
424 * Description : Sets the specified pin HIGH or LOW
425 * Input : Pin and value
426 * Output : None
427 * Return : 1 on success and a negative number on failure
428-----
429 int tinkerAnalogWrite(String command)
430 {
431 bool value = 0;
432 //convert ascii to integer
433 int pinNumber = command.charAt(1) - '0';
434 //Sanity check to see if the pin numbers are within limits
435 if (pinNumber < 0 || pinNumber > 7) return -1;
436
437 if (command.substring(3,7) == "HIGH") value = 1;
438 else if (command.substring(3,7) == "LOW") value = 0;
439 else return -2;
440
441 if (command.startsWith("D"))
442 {
443 pinMode(pinNumber, OUTPUT);
444 digitalWrite(pinNumber, value);
445 return 1;
446 }
447
448 else if (command.startsWith("A"))
449 {
450 pinMode(pinNumber, OUTPUT);
451 digitalWrite(pinNumber, value);
452 return 1;
453 }
454 }
455
456-----
457 * Function Name : tinkerAnalogWrite
458 * Description : Sets the specified pin HIGH or LOW
459 * Input : Pin and value
460 * Output : None
461 * Return : 1 on success and a negative number on failure
462-----
463 int tinkerAnalogWrite(String command)
464 {
465 bool value = 0;
466 //convert ascii to integer
467 int pinNumber = command.charAt(1) - '0';
468 //Sanity check to see if the pin numbers are within limits
469 if (pinNumber < 0 || pinNumber > 7) return -1;
470
471 if (command.substring(3,7) == "HIGH") value = 1;
472 else if (command.substring(3,7) == "LOW") value = 0;
473 else return -2;
474
475 if (command.startsWith("D"))
476 {
477 pinMode(pinNumber, OUTPUT);
478 digitalWrite(pinNumber, value);
479 return 1;
480 }
481
482 else if (command.startsWith("A"))
483 {
484 pinMode(pinNumber, OUTPUT);
485 digitalWrite(pinNumber, value);
486 return 1;
487 }
488 }
489
490-----
491 * Function Name : tinkerAnalogWrite
492 * Description : Sets the specified pin HIGH or LOW
493 * Input : Pin and value
494 * Output : None
495 * Return : 1 on success and a negative number on failure
496-----
497 int tinkerAnalogWrite(String command)
498 {
499 bool value = 0;
500 //convert ascii to integer
501 int pinNumber = command.charAt(1) - '0';
502 //Sanity check to see if the pin numbers are within limits
503 if (pinNumber < 0 || pinNumber > 7) return -1;
504
505 if (command.substring(3,7) == "HIGH") value = 1;
506 else if (command.substring(3,7) == "LOW") value = 0;
507 else return -2;
508
509 if (command.startsWith("D"))
510 {
511 pinMode(pinNumber, OUTPUT);
512 digitalWrite(pinNumber, value);
513 return 1;
514 }
515
516 else if (command.startsWith("A"))
517 {
518 pinMode(pinNumber, OUTPUT);
519 digitalWrite(pinNumber, value);
520 return 1;
521 }
522 }
523
524-----
525 * Function Name : tinkerAnalogWrite
526 * Description : Sets the specified pin HIGH or LOW
527 * Input : Pin and value
528 * Output : None
529 * Return : 1 on success and a negative number on failure
530-----
531 int tinkerAnalogWrite(String command)
532 {
533 bool value = 0;
534 //convert ascii to integer
535 int pinNumber = command.charAt(1) - '0';
536 //Sanity check to see if the pin numbers are within limits
537 if (pinNumber < 0 || pinNumber > 7) return -1;
538
539 if (command.substring(3,7) == "HIGH") value = 1;
540 else if (command.substring(3,7) == "LOW") value = 0;
541 else return -2;
542
543 if (command.startsWith("D"))
544 {
545 pinMode(pinNumber, OUTPUT);
546 digitalWrite(pinNumber, value);
547 return 1;
548 }
549
550 else if (command.startsWith("A"))
551 {
552 pinMode(pinNumber, OUTPUT);
553 digitalWrite(pinNumber, value);
554 return 1;
555 }
556 }
557
558-----
559 * Function Name : tinkerAnalogWrite
560 * Description : Sets the specified pin HIGH or LOW
561 * Input : Pin and value
562 * Output : None
563 * Return : 1 on success and a negative number on failure
564-----
565 int tinkerAnalogWrite(String command)
566 {
567 bool value = 0;
568 //convert ascii to integer
569 int pinNumber = command.charAt(1) - '0';
570 //Sanity check to see if the pin numbers are within limits
571 if (pinNumber < 0 || pinNumber > 7) return -1;
572
573 if (command.substring(3,7) == "HIGH") value = 1;
574 else if (command.substring(3,7) == "LOW") value = 0;
575 else return -2;
576
577 if (command.startsWith("D"))
578 {
579 pinMode(pinNumber, OUTPUT);
580 digitalWrite(pinNumber, value);
581 return 1;
582 }
583
584 else if (command.startsWith("A"))
585 {
586 pinMode(pinNumber, OUTPUT);
587 digitalWrite(pinNumber, value);
588 return 1;
589 }
590 }
591
592-----
593 * Function Name : tinkerAnalogWrite
594 * Description : Sets the specified pin HIGH or LOW
595 * Input : Pin and value
596 * Output : None
597 * Return : 1 on success and a negative number on failure
598-----
599 int tinkerAnalogWrite(String command)
600 {
601 bool value = 0;
602 //convert ascii to integer
603 int pinNumber = command.charAt(1) - '0';
604 //Sanity check to see if the pin numbers are within limits
605 if (pinNumber < 0 || pinNumber > 7) return -1;
606
607 if (command.substring(3,7) == "HIGH") value = 1;
608 else if (command.substring(3,7) == "LOW") value = 0;
609 else return -2;
610
611 if (command.startsWith("D"))
612 {
613 pinMode(pinNumber, OUTPUT);
614 digitalWrite(pinNumber, value);
615 return 1;
616 }
617
618 else if (command.startsWith("A"))
619 {
620 pinMode(pinNumber, OUTPUT);
621 digitalWrite(pinNumber, value);
622 return 1;
623 }
624 }
625
626-----
627 * Function Name : tinkerAnalogWrite
628 * Description : Sets the specified pin HIGH or LOW
629 * Input : Pin and value
630 * Output : None
631 * Return : 1 on success and a negative number on failure
632-----
633 int tinkerAnalogWrite(String command)
634 {
635 bool value = 0;
636 //convert ascii to integer
637 int pinNumber = command.charAt(1) - '0';
638 //Sanity check to see if the pin numbers are within limits
639 if (pinNumber < 0 || pinNumber > 7) return -1;
640
641 if (command.substring(3,7) == "HIGH") value = 1;
642 else if (command.substring(3,7) == "LOW") value = 0;
643 else return -2;
644
645 if (command.startsWith("D"))
646 {
647 pinMode(pinNumber, OUTPUT);
648 digitalWrite(pinNumber, value);
649 return 1;
650 }
651
652 else if (command.startsWith("A"))
653 {
654 pinMode(pinNumber, OUTPUT);
655 digitalWrite(pinNumber, value);
656 return 1;
657 }
658 }
659
660-----
661 * Function Name : tinkerAnalogWrite
662 * Description : Sets the specified pin HIGH or LOW
663 * Input : Pin and value
664 * Output : None
665 * Return : 1 on success and a negative number on failure
666-----
667 int tinkerAnalogWrite(String command)
668 {
669 bool value = 0;
670 //convert ascii to integer
671 int pinNumber = command.charAt(1) - '0';
672 //Sanity check to see if the pin numbers are within limits
673 if (pinNumber < 0 || pinNumber > 7) return -1;
674
675 if (command.substring(3,7) == "HIGH") value = 1;
676 else if (command.substring(3,7) == "LOW") value = 0;
677 else return -2;
678
679 if (command.startsWith("D"))
680 {
681 pinMode(pinNumber, OUTPUT);
682 digitalWrite(pinNumber, value);
683 return 1;
684 }
685
686 else if (command.startsWith("A"))
687 {
688 pinMode(pinNumber, OUTPUT);
689 digitalWrite(pinNumber, value);
690 return 1;
691 }
692 }
693
694-----
695 * Function Name : tinkerAnalogWrite
696 * Description : Sets the specified pin HIGH or LOW
697 * Input : Pin and value
698 * Output : None
699 * Return : 1 on success and a negative number on failure
700-----
701 int tinkerAnalogWrite(String command)
702 {
703 bool value = 0;
704 //convert ascii to integer
705 int pinNumber = command.charAt(1) - '0';
706 //Sanity check to see if the pin numbers are within limits
707 if (pinNumber < 0 || pinNumber > 7) return -1;
708
709 if (command.substring(3,7) == "HIGH") value = 1;
710 else if (command.substring(3,7) == "LOW") value = 0;
711 else return -2;
712
713 if (command.startsWith("D"))
714 {
715 pinMode(pinNumber, OUTPUT);
716 digitalWrite(pinNumber, value);
717 return 1;
718 }
719
720 else if (command.startsWith("A"))
721 {
722 pinMode(pinNumber, OUTPUT);
723 digitalWrite(pinNumber, value);
724 return 1;
725 }
726 }
727
728-----
729 * Function Name : tinkerAnalogWrite
730 * Description : Sets the specified pin HIGH or LOW
731 * Input : Pin and value
732 * Output : None
733 * Return : 1 on success and a negative number on failure
734-----
735 int tinkerAnalogWrite(String command)
736 {
737 bool value = 0;
738 //convert ascii to integer
739 int pinNumber = command.charAt(1) - '0';
740 //Sanity check to see if the pin numbers are within limits
741 if (pinNumber < 0 || pinNumber > 7) return -1;
742
743 if (command.substring(3,7) == "HIGH") value = 1;
744 else if (command.substring(3,7) == "LOW") value = 0;
745 else return -2;
746
747 if (command.startsWith("D"))
748 {
749 pinMode(pinNumber, OUTPUT);
750 digitalWrite(pinNumber, value);
751 return 1;
752 }
753
754 else if (command.startsWith("A"))
755 {
756 pinMode(pinNumber, OUTPUT);
757 digitalWrite(pinNumber, value);
758 return 1;
759 }
760 }
761
762-----
763 * Function Name : tinkerAnalogWrite
764 * Description : Sets the specified pin HIGH or LOW
765 * Input : Pin and value
766 * Output : None
767 * Return : 1 on success and a negative number on failure
768-----
769 int tinkerAnalogWrite(String command)
770 {
771 bool value = 0;
772 //convert ascii to integer
773 int pinNumber = command.charAt(1) - '0';
774 //Sanity check to see if the pin numbers are within limits
775 if (pinNumber < 0 || pinNumber > 7) return -1;
776
777 if (command.substring(3,7) == "HIGH") value = 1;
778 else if (command.substring(3,7) == "LOW") value = 0;
779 else return -2;
780
781 if (command.startsWith("D"))
782 {
783 pinMode(pinNumber, OUTPUT);
784 digitalWrite(pinNumber, value);
785 return 1;
786 }
787
788 else if (command.startsWith("A"))
789 {
790 pinMode(pinNumber, OUTPUT);
791 digitalWrite(pinNumber, value);
792 return 1;
793 }
794 }
795
796-----
797 * Function Name : tinkerAnalogWrite
798 * Description : Sets the specified pin HIGH or LOW
799 * Input : Pin and value
800 * Output : None
801 * Return : 1 on success and a negative number on failure
802-----
803 int tinkerAnalogWrite(String command)
804 {
805 bool value = 0;
806 //convert ascii to integer
807 int pinNumber = command.charAt(1) - '0';
808 //Sanity check to see if the pin numbers are within limits
809 if (pinNumber < 0 || pinNumber > 7) return -1;
810
811 if (command.substring(3,7) == "HIGH") value = 1;
812 else if (command.substring(3,7) == "LOW") value = 0;
813 else return -2;
814
815 if (command.startsWith("D"))
816 {
817 pinMode(pinNumber, OUTPUT);
818 digitalWrite(pinNumber, value);
819 return 1;
820 }
821
822 else if (command.startsWith("A"))
823 {
824 pinMode(pinNumber, OUTPUT);
825 digitalWrite(pinNumber, value);
826 return 1;
827 }
828 }
829
830-----
831 * Function Name : tinkerAnalogWrite
832 * Description : Sets the specified pin HIGH or LOW
833 * Input : Pin and value
834 * Output : None
835 * Return : 1 on success and a negative number on failure
836-----
837 int tinkerAnalogWrite(String command)
838 {
839 bool value = 0;
840 //convert ascii to integer
841 int pinNumber = command.charAt(1) - '0';
842 //Sanity check to see if the pin numbers are within limits
843 if (pinNumber < 0 || pinNumber > 7) return -1;
844
845 if (command.substring(3,7) == "HIGH") value = 1;
846 else if (command.substring(3,7) == "LOW") value = 0;
847 else return -2;
848
849 if (command.startsWith("D"))
850 {
851 pinMode(pinNumber, OUTPUT);
852 digitalWrite(pinNumber, value);
853 return 1;
854 }
855
856 else if (command.startsWith("A"))
857 {
858 pinMode(pinNumber, OUTPUT);
859 digitalWrite(pinNumber, value);
860 return 1;
861 }
862 }
863
864-----
865 * Function Name : tinkerAnalogWrite
866 * Description : Sets the specified pin HIGH or LOW
867 * Input : Pin and value
868 * Output : None
869 * Return : 1 on success and a negative number on failure
870-----
871 int tinkerAnalogWrite(String command)
872 {
873 bool value = 0;
874 //convert ascii to integer
875 int pinNumber = command.charAt(1) - '0';
876 //Sanity check to see if the pin numbers are within limits
877 if (pinNumber < 0 || pinNumber > 7) return -1;
878
879 if (command.substring(3,7) == "HIGH") value = 1;
880 else if (command.substring(3,7) == "LOW") value = 0;
881 else return -2;
882
883 if (command.startsWith("D"))
884 {
885 pinMode(pinNumber, OUTPUT);
886 digitalWrite(pinNumber, value);
887 return 1;
888 }
889
890 else if (command.startsWith("A"))
891 {
892 pinMode(pinNumber, OUTPUT);
893 digitalWrite(pinNumber, value);
894 return 1;
895 }
896 }
897
898-----
899 * Function Name : tinkerAnalogWrite
900 * Description : Sets the specified pin HIGH or LOW
901 * Input : Pin and value
902 * Output : None
903 * Return : 1 on success and a negative number on failure
904-----
905 int tinkerAnalogWrite(String command)
906 {
907 bool value = 0;
908 //convert ascii to integer
909 int pinNumber = command.charAt(1) - '0';
910 //Sanity check to see if the pin numbers are within limits
911 if (pinNumber < 0 || pinNumber > 7) return -1;
912
913 if (command.substring(3,7) == "HIGH") value = 1;
914 else if (command.substring(3,7) == "LOW") value = 0;
915 else return -2;
916
917 if (command.startsWith("D"))
918 {
919 pinMode(pinNumber, OUTPUT);
920 digitalWrite(pinNumber, value);
921 return 1;
922 }
923
924 else if (command.startsWith("A"))
925 {
926 pinMode(pinNumber, OUTPUT);
927 digitalWrite(pinNumber, value);
928 return 1;
929 }
930 }
931
932-----
933 * Function Name : tinkerAnalogWrite
934 * Description : Sets the specified pin HIGH or LOW
935 * Input : Pin and value
936 * Output : None
937 * Return : 1 on success and a negative number on failure
938-----
939 int tinkerAnalogWrite(String command)
940 {
941 bool value = 0;
942 //convert ascii to integer
943 int pinNumber = command.charAt(1) - '0';
944 //Sanity check to see if the pin numbers are within limits
945 if (pinNumber < 0 || pinNumber > 7) return -1;
946
947 if (command.substring(3,7) == "HIGH") value = 1;
948 else if (command.substring(3,7) == "LOW") value = 0;
949 else return -2;
950
951 if (command.startsWith("D"))
952 {
953 pinMode(pinNumber, OUTPUT);
954 digitalWrite(pinNumber, value);
955 return 1;
956 }
957
958 else if (command.startsWith("A"))
959 {
960 pinMode(pinNumber, OUTPUT);
961 digitalWrite(pinNumber, value);
962 return 1;
963 }
964 }
965
966-----
967 * Function Name : tinkerAnalogWrite
968 * Description : Sets the specified pin HIGH or LOW
969 * Input : Pin and value
970 * Output : None
971 * Return : 1 on success and a negative number on failure
972-----
973 int tinkerAnalogWrite(String command)
974 {
975 bool value = 0;
976 //convert ascii to integer
977 int pinNumber = command.charAt(1) - '0';
978 //Sanity check to see if the pin numbers are within limits
979 if (pinNumber < 0 || pinNumber > 7) return -1;
980
981 if (command.substring(3,7) == "HIGH") value = 1;
982 else if (command.substring(3,7) == "LOW") value = 0;
983 else return -2;
984
985 if (command.startsWith("D"))
986 {
987 pinMode(pinNumber, OUTPUT);
988 digitalWrite(pinNumber, value);
989 return 1;
990 }
991
992 else if (command.startsWith("A"))
993 {
994 pinMode(pinNumber, OUTPUT);
995 digitalWrite(pinNumber, value);
996 return 1;
997 }
998 }
999
1000-----
1001 * Function Name : tinkerAnalogWrite
1002 * Description : Sets the specified pin HIGH or LOW
1003 * Input : Pin and value
1004 * Output : None
1005 * Return : 1 on success and a negative number on failure
1006-----
1007 int tinkerAnalogWrite(String command)
1008 {
1009 bool value = 0;
1010 //convert ascii to integer
1011 int pinNumber = command.charAt(1) - '0';
1012 //Sanity check to see if the pin numbers are within limits
1013 if (pinNumber < 0 || pinNumber > 7) return -1;
1014
1015 if (command.substring(3,7) == "HIGH") value = 1;
1016 else if (command.substring(3,7) == "LOW") value = 0;
1017 else return -2;
1018
1019 if (command.startsWith("D"))
1020 {
1021 pinMode(pinNumber, OUTPUT);
1022 digitalWrite(pinNumber, value);
1023 return 1;
1024 }
1025
1026 else if (command.startsWith("A"))
1027 {
1028 pinMode(pinNumber, OUTPUT);
1029 digitalWrite(pinNumber, value);
1030 return 1;
1031 }
1032 }
1033
1034-----
1035 * Function Name : tinkerAnalogWrite
1036 * Description : Sets the specified pin HIGH or LOW
1037 * Input : Pin and value
1038 * Output : None
1039 * Return : 1 on success and a negative number on failure
1040-----
1041 int tinkerAnalogWrite(String command)
1042 {
1043 bool value = 0;
1044 //convert ascii to integer
1045 int pinNumber = command.charAt(1) - '0';
1046 //Sanity check to see if the pin numbers are within limits
1047 if (pinNumber < 0 || pinNumber > 7) return -1;
1048
1049 if (command.substring(3,7) == "HIGH") value = 1;
1050 else if (command.substring(3,7) == "LOW") value = 0;
1051 else return -2;
1052
1053 if (command.startsWith("D"))
1054 {
1055 pinMode(pinNumber, OUTPUT);
1056 digitalWrite(pinNumber, value);
1057 return 1;
1058 }
1059
1060 else if (command.startsWith("A"))
1061 {
1062 pinMode(pinNumber, OUTPUT);
1063 digitalWrite(pinNumber, value);
1064 return 1;
1065 }
1066 }
1067
1068-----
1069 * Function Name : tinkerAnalogWrite
1070 * Description : Sets the specified pin HIGH or LOW
1071 * Input : Pin and value
1072 * Output : None
1073 * Return : 1 on success and a negative number on failure
1074-----
1075 int tinkerAnalogWrite(String command)
1076 {
1077 bool value = 0;
1078 //convert ascii to integer
1079 int pinNumber = command.charAt(1) - '0';
1080 //Sanity check to see if the pin numbers are within limits
1081 if (pinNumber < 0 || pinNumber > 7) return -1;
1082
1083 if (command.substring(3,7) == "HIGH") value = 1;
1084 else if (command.substring(3,7) == "LOW") value = 0;
1085 else return -2;
1086
1087 if (command.startsWith("D"))
1088 {
1089 pinMode(pinNumber, OUTPUT);
1090 digitalWrite(pinNumber, value);
1091 return 1;
1092 }
1093
1094 else if (command.startsWith("A"))
1095 {
1096 pinMode(pinNumber, OUTPUT);
1097 digitalWrite(pinNumber, value);
1098 return 1;
1099 }
1100 }
1101
1102-----
1103 * Function Name : tinkerAnalogWrite
1104 * Description : Sets the specified pin HIGH or LOW
1105 * Input : Pin and value
1106 * Output : None
1107 * Return : 1 on success and a negative number on failure
1108-----
1109 int tinkerAnalogWrite(String command)
1110 {
1111 bool value = 0;
1112 //convert ascii to integer
1113 int pinNumber = command.charAt(1) - '0';
1114 //Sanity check to see if the pin numbers are within limits
1115 if (pinNumber < 0 || pinNumber > 7) return -1;
1116
1117 if (command.substring(3,7) == "HIGH") value = 1;
1118 else if (command.substring(3,7) == "LOW") value = 0;
1119 else return -2;
1120
1121 if (command.startsWith("D"))
1122 {
1123 pinMode(pinNumber, OUTPUT);
1124 digitalWrite(pinNumber, value);
1125 return 1;
1126 }
1127
1128 else if (command.startsWith("A"))
1129 {
1130 pinMode(pinNumber, OUTPUT);
1131 digitalWrite(pinNumber, value);
1132 return 1;
1133 }
1134 }
1135
1136-----
1137 * Function Name : tinkerAnalogWrite
1138 * Description : Sets the specified pin HIGH or LOW
1139 * Input : Pin and value
1140 * Output : None
1141 * Return : 1 on success and a negative number on failure
1142-----
1143 int tinkerAnalogWrite(String command)
1144 {
1145 bool value = 0;
1146 //convert ascii to integer
1147 int pinNumber = command.charAt(1) - '0';
1148 //Sanity check to see if the pin numbers are within limits
1149 if (pinNumber < 0 || pinNumber > 7) return -1;
1150
1151 if (command.substring(3,7) == "HIGH") value = 1;
1152 else if (command.substring(3,7) == "LOW") value = 0;
1153 else return -2;
1154
1155 if (command.startsWith("D"))
1156 {
1157 pinMode(pinNumber, OUTPUT);
1158 digitalWrite(pinNumber, value);
1159 return 1;
1160 }
1161
1162 else if (command.startsWith("A"))
1163 {
1164 pinMode(pinNumber, OUTPUT);
1165 digitalWrite(pinNumber, value);
1166 return 1;
1167 }
1168 }
1169
1170-----
1171 * Function Name : tinkerAnalogWrite
1172 * Description : Sets the specified pin HIGH or LOW
1173 * Input : Pin and value
1174 * Output : None
1175 * Return : 1 on success and a negative number on failure
1176-----
1177 int tinkerAnalogWrite(String command)
1178 {
1179 bool value = 0;
1180 //convert ascii to integer
1181 int pinNumber = command.charAt(1) - '0';
1182 //Sanity check to see if the pin numbers are within limits
1183 if (pinNumber < 0 || pinNumber > 7) return -1;
1184
```

# SDKS FOR MOBILE AND WEB DEVELOPMENT

```
myPhoton!.getVariable("temperature",
  completion: { (result:Any?, error:Error?) → Void in
    if let _ = error {
      print("Failed reading temperature from device")
    }
    else {
      if let temp = result as? NSNumber {
        print("Room temp is \(temp.stringValue) degrees")
      }
    }
  })
```



```
var brew = particle.callFunction({
  deviceId: 'DEVICE_ID',
  name: 'brew',
  argument: 'D0:HIGH',
  auth: token
});

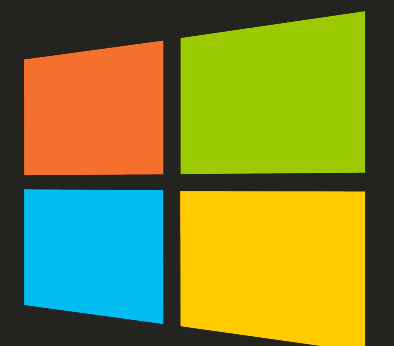
brew.then(
  (data) ⇒ console.log('Function called:', data),
  (err) ⇒ console.log('An error occurred:', err);
);
```



```
List<ParticleDevice> devices = ParticleCloudSDK.getCloud().getDevices();
for (ParticleDevice device : devices) {
  if (device.getName().equals("myDevice")) {
    doSomethingWithMyDevice(device);
    break;
  }
}
```



```
ParticleDevice myDevice = null;
List<ParticleDevice> devices =
ParticleCloud.SharedCloud.GetDevicesAsync();
foreach (ParticleDevice device in devices)
{
  if (device.Name().equals("myDeviceName"))
    myDevice = device;
}
```



# PARTICLE DEVICE CLOUD & CONSOLE

Particle Docs | Contact Sales | Support | bsatrom@gmail.com

Devices > View Device PING EDIT

ID: 3b001e000247363339343638 Name: trash-panda  
Device OS: 0.8.0-rc.8 Type: Photon  
Serial Number: PHHMAB819ZY6QXD Last Handshake: Jul 12th 2018, 9:32 am

Notes  
Click the edit button to keep notes on this device, like 'Deployed to customer site'.

**EVENT LOGS** DIAGNOSTICS NEW

spark/flash/status device came online spark/device/last\_reset spark/device/diagnostics/update

Event Name	Data	Published At
spark/device/diagnostics/update	{"device":{"system":	July 12th at 9:32:45 am
spark/device/last_reset	user	July 12th at 9:32:45 am
device came online		July 12th at 9:32:45 am
spark/flash/status	success	July 12th at 9:32:33 am
spark/flash/status	started	July 12th at 9:32:33 am

PAUSE SEE IN TERMINAL PUBLISH EVENT

**DEVICE VITALS NEW**

Jul 12th, 2018, 09:31AM

- Strong Wi-Fi signal
- 0 disconnect events
- 67ms round-trip time
- 0 rate-limited publishes
- 30kB of 81kB RAM used

[Download History](#) | [Run diagnostics](#)

**FUNCTIONS**

f updateFName  
Argument CALL

f updateLName  
Argument CALL

**VARIABLES**

✓ wearerFName (string) GET

✓ wearerLName (string) GET

# PARTICLE DEVICE CLOUD & CONSOLE

The screenshot displays the Particle Device Cloud & Console interface. At the top, there are navigation links for Docs, Contact Sales, Support, and a user profile for bsatrom@gmail.com. The main header shows 'Devices > View Device' with 'PING' and 'EDIT' buttons.

Device details include:  
ID: 3b001e000247363339343638  
Name: trash-panda  
Device OS: 0.8.0-rc.8  
Serial Number: PHHMAB819ZY6QXD  
Type: Photon  
Last Handshake: J

The 'DIAGNOSTICS' section is highlighted with a 'NEW' badge and a 'RUN TESTS' button. A large green box contains the message: 'EVERYTHING LOOKS GOOD! All diagnostic tests have passed. This device is healthy.'

Below this, two columns show 'Device Vitals' and 'Device Cloud' status, both marked as 'Healthy'.  
**Device Vitals:**  
- Strong Wi-Fi signal  
- 0 disconnect events  
- 70ms round-trip time  
- 0 rate-limited publishes  
- 30kB of 81kB RAM used

**Device Cloud:**  
- API  
- Device Service  
- Webhooks

The 'EVENT LOGS' section shows a table of events with columns for 'EVENT NAME' and 'DATA'.

EVENT NAME	DATA
spark/device/diagnostics/update	{"device":{"system":
spark/device/last_reset	user
device came online	
spark/flash/status	success
spark/flash/status	started

At the bottom, there is a dropdown menu for 'wearerLName (string)' with a 'GET' button.

# INTEGRATIONS FOR EXTENDING YOUR IOT SOLUTIONS TO OTHER CLOUDS

The screenshot shows the Particle Integrations page. The top navigation bar includes links for Docs, Contact Sales, Support, and the user email bsatrom@gmail.com. The main heading is "Integrations". There are five integration cards displayed:

- Azure IoT Hub**: Includes brew\_temp, any device, and tangible-iot-hu...
- Google Cloud Platform**: Includes googlePublish, brew\_buddy\_2, and brew-reading.
- Webhook**: Includes BatteryState, any device, and everlive.com.
- Webhook**: Includes BatchTemp.
- Webhook**: Includes BrewStageTemp, any device, and firebaseio.com.

A dashed box labeled "NEW INTEGRATION" with a plus sign is located at the bottom center of the page.

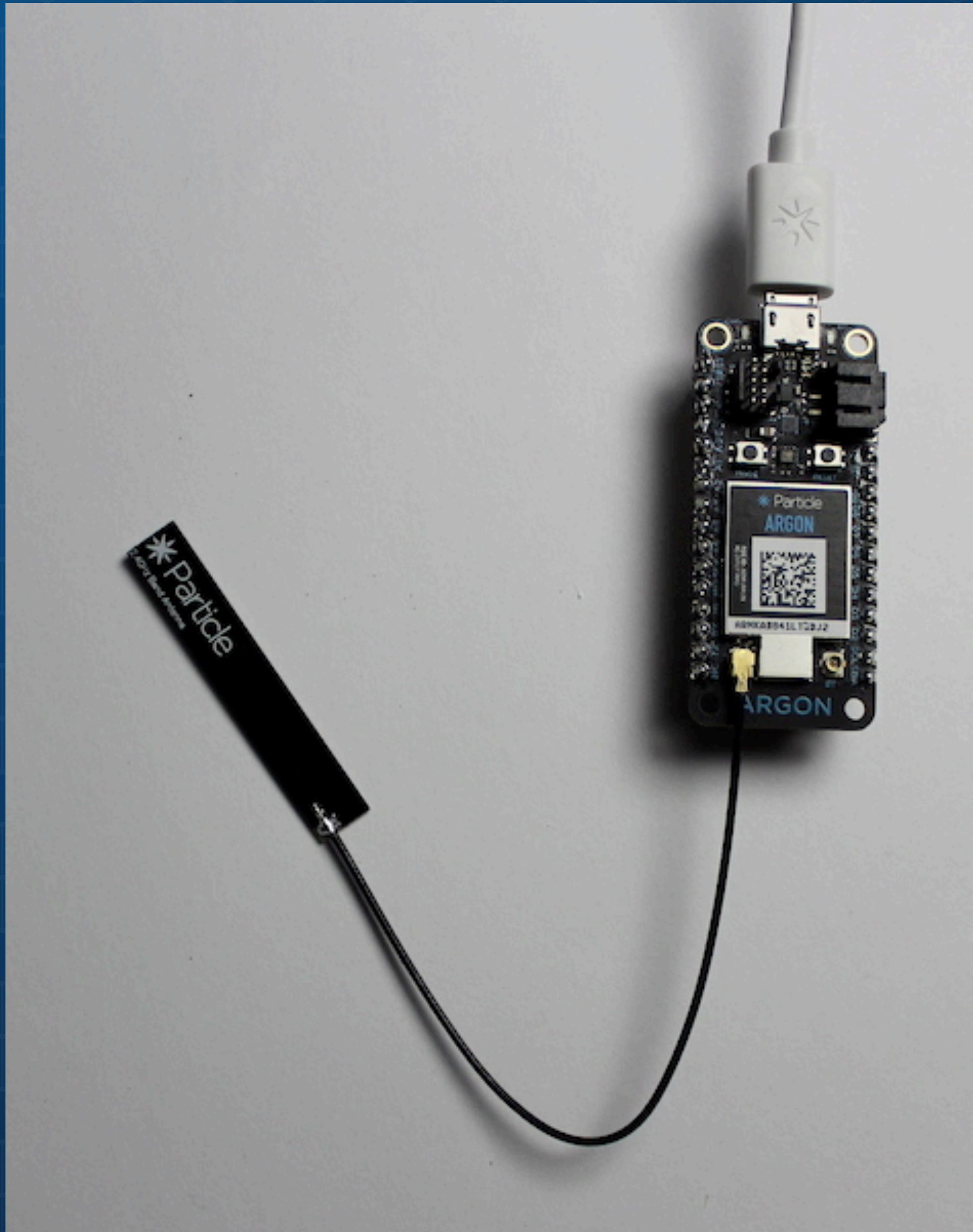
The screenshot shows the "Integrations > New Integration" page. The left sidebar is visible. The main content area lists four integration options:

- Google Maps**: Geolocate Particle devices via visible Wi-Fi access points or Cellular towers.
- Azure IoT Hub**: Stream Particle device data into the Azure ecosystem.
- Google Cloud Platform**: Tie into an enterprise grade suite of cloud-based data storage and analysis tools.
- Webhook**: Push Particle device data to other web services in real-time.

**WHY PARTICLE**

**THE PARTICLE CLOUD & FRIENDS**

**CLAIMING YOUR FIRST DEVICE**



workshop@reflektor.dk

### Get your Argon ready for setup

Plug your device into a power source

I have attached the antenna and see that the device is blinking blue

**NEXT**

USE WITH ETHERNET?

Toggle Ethernet Featherwing setup





# LAB PREREQUISITES

<https://part.cl/accelerate>

- \* Create a new Particle account at [login.particle.io](https://login.particle.io)
- \* Install Particle Workbench (or the Workbench VS Code Extension)
- \* Install the Particle iOS or Android App
- \* Install the Particle CLI