Whiskey as a Service

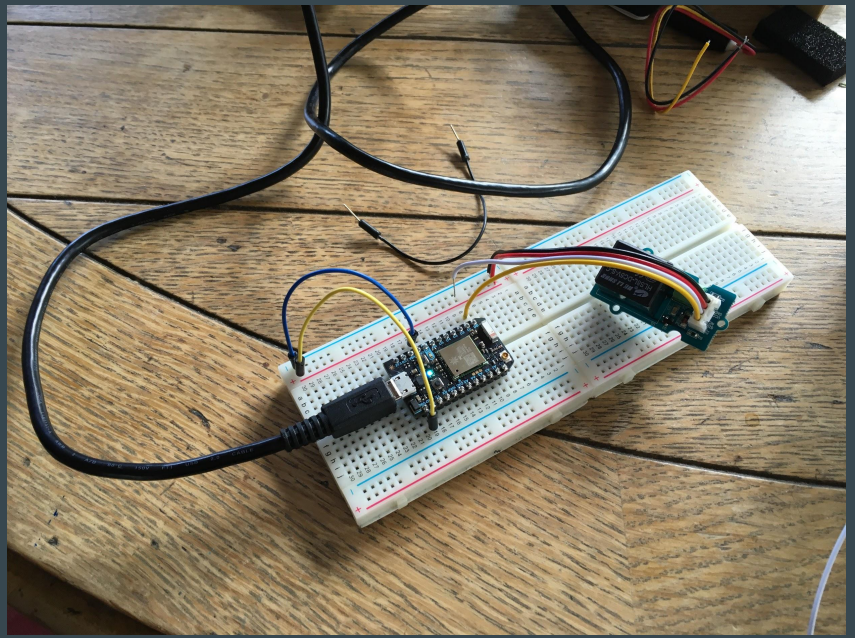
...

team BevOps

mission statement

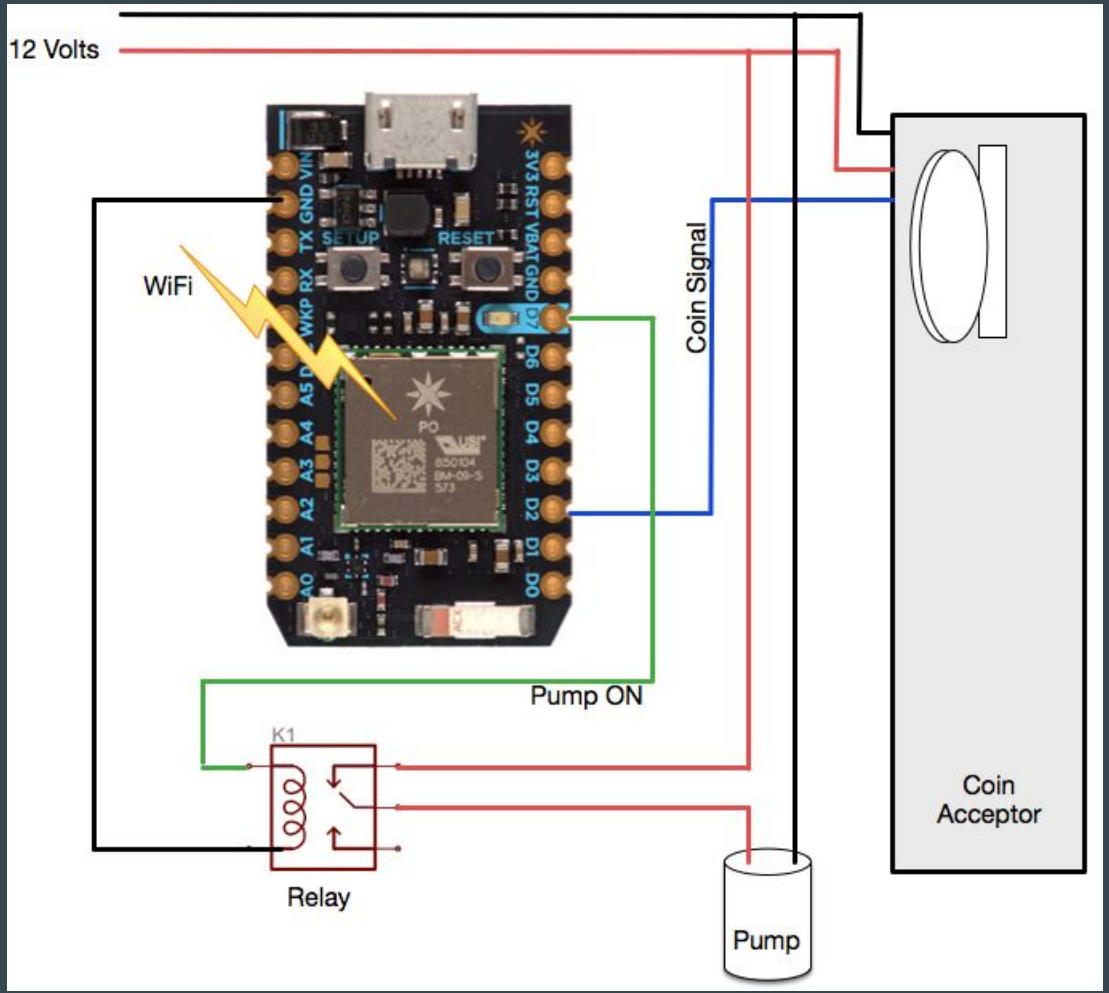
dispense whiskey with minimum efficiency and maximum amusement

prototyping



coin acceptor and pump

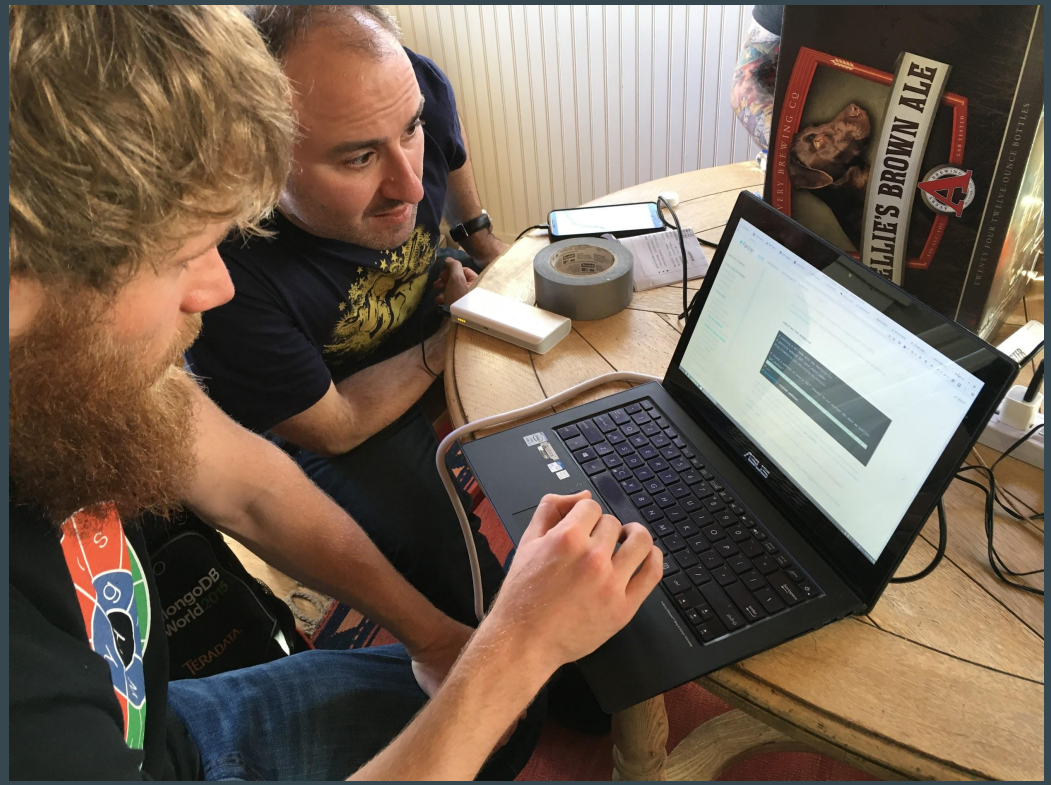
vending schematic



vending code

```
const int coinPin = 2;
const int pumpPin = 7;

void setup() {
 Serial.begin(9600);              // start serial communication
 pinMode(pumpPin, OUTPUT);        // pump relay pin as output
 pinMode(coinPin, INPUT);         // sets the coin pin as input
}

void loop() {
 if (digitalRead(coinPin) == HIGH) {
     Particle.publish("drink_pour");// publish event to cloud
     digitalWrite(pumpPin, HIGH);   // closes the relay
     delay(30000);                  // waits for 30 seconds
     digitalWrite(pumpPin, LOW);    // opens the relay
 }
}
```

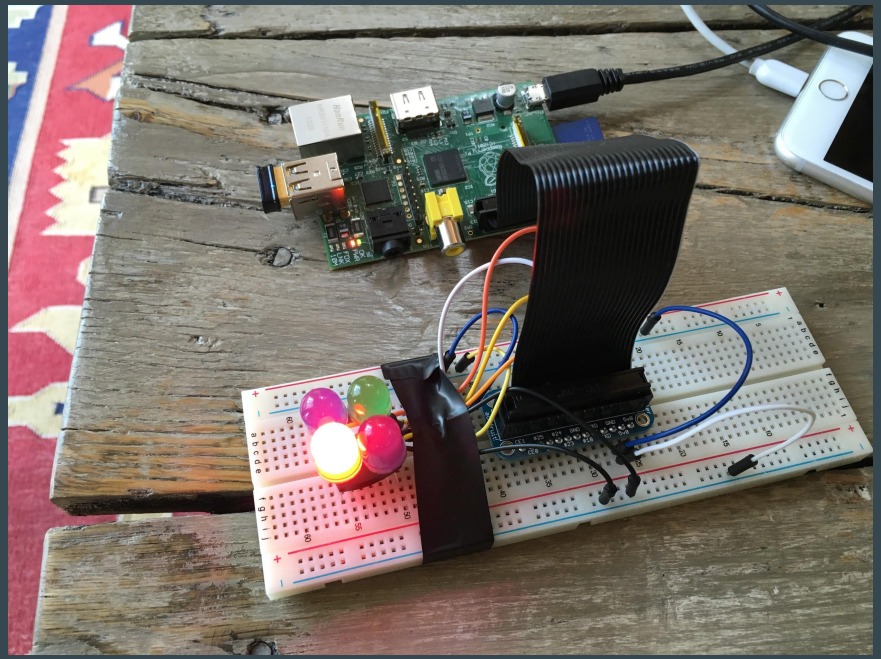
slack all the bots!



slack publish code

```
const https = require('https');

var particleUrl = "https://api.particle.io/v1/devices/
var slackUrl = "https://slack.com/api/chat.postMessage

function processResponse(response) {
  response.setEncoding('utf8');
  response.on("data", function(data) {
    console.log(data);
    if (/drink_pour/.test(data)) {
        https.get(slackUrl, done);
    }
  });

  response.on("error", console.error);

  response.on("end", function() {
    console.log("closed connection to particle");
  });
}

function done(response) {
 console.log("response from slack");
}

https.get(particleUrl, processResponse);
```

blinkey lights



blinkey code

```
seq = [7, 11, 13, 15]  # Pin sequence
GPIO.setmode(GPIO.BOARD)

while True:
    print "Checking last message on slack channel"
    response = json.loads(urllib2.urlopen(GET_LAST_MESSAGE_URL).read())
    text = response['messages'][0]['text']

    if text == "Drink being poured!":
        print "Triggering party. Notifying channel. "
        urllib2.urlopen(NOTIFY_CHANNEL_URL).read()

        for j in seq:
            GPIO.setup(j, GPIO.OUT)

        for i in range(0, 8):
            print "Iteration: " + str(i)
            for j in seq:
                GPIO.output(j,True)
                time.sleep(1)
                GPIO.output(j,False)

        print "Done"
        GPIO.cleanup()

    else:
        print "Sleeping for 3 seconds"
        time.sleep(3)
```

the form factor