



**electric imp**

## **HackBright Workshop**

Matt Haines - @beardedinventor  
Betsy Rhodes - @betsyrhodes  
Gino Miglio - @collardgreens  
Tom Byrne - @ersatzavian

While you are waiting please make sure you have downloaded the 'Electric Imp' app for mobile device

# What are we doing tonight

---

Obligatory Internet of Things (IoT) Slide

Workshop

- Getting your imp online
- Controlling an LED with a button
- Controlling an LED with an API
- Controlling an LED with Twitter

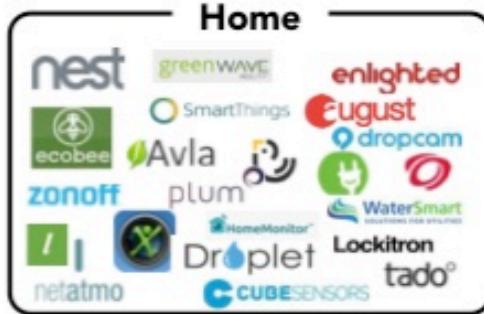
Take home project

- TempBug + SparkFun Data

<https://github.com/beardedinventor/hackbright2k15>



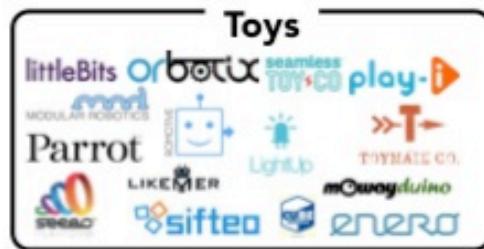
# The Internet of Things



## Internet of Things

500 Companies  
\$2.24B Funding

See the updated scan and more:  
[www.venturescanner.com/scans/internet-of-things](http://www.venturescanner.com/scans/internet-of-things)



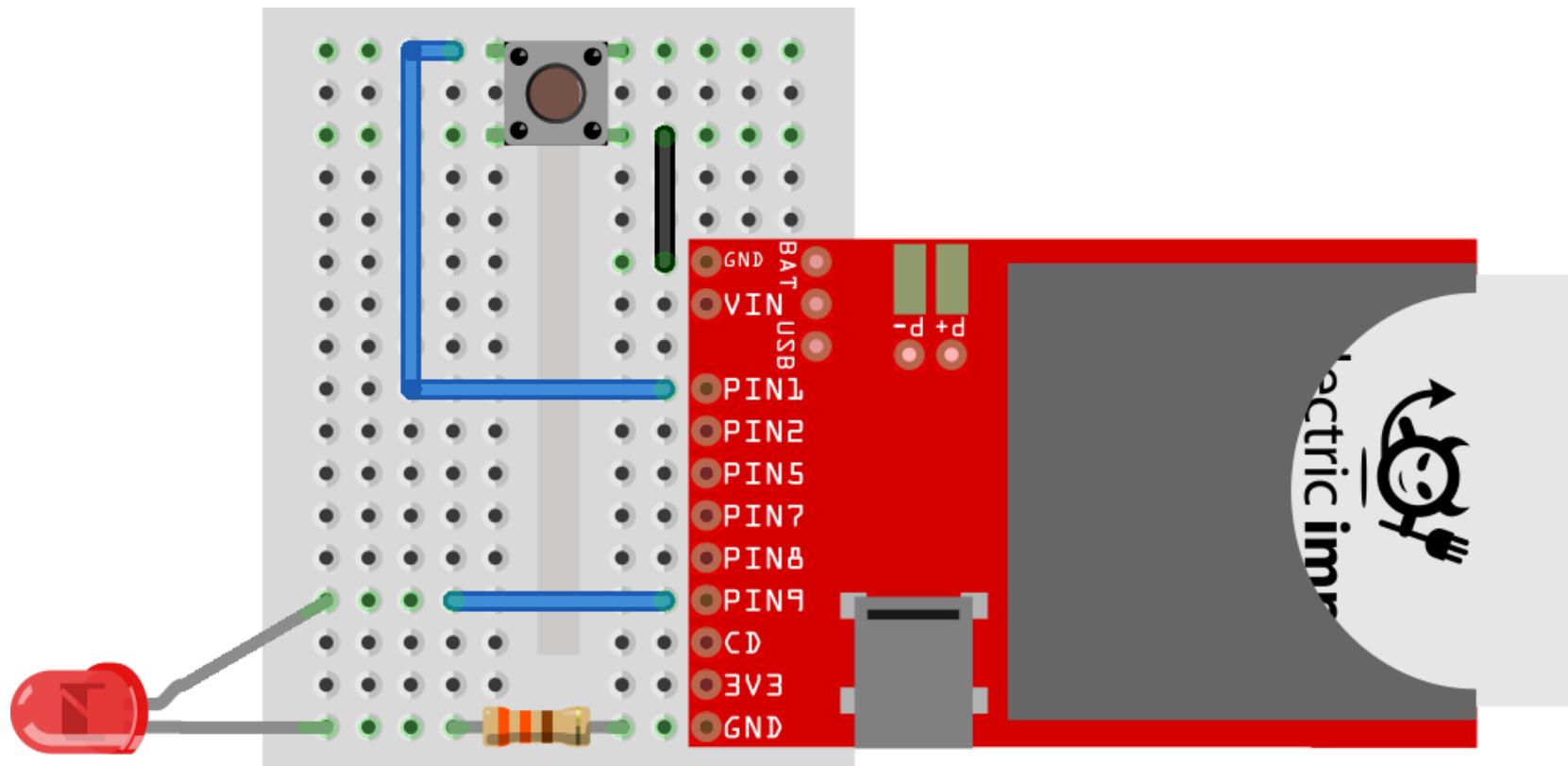
Venture Scanner



Let's get started



# Build a circuit



Made with  Fritzing.org

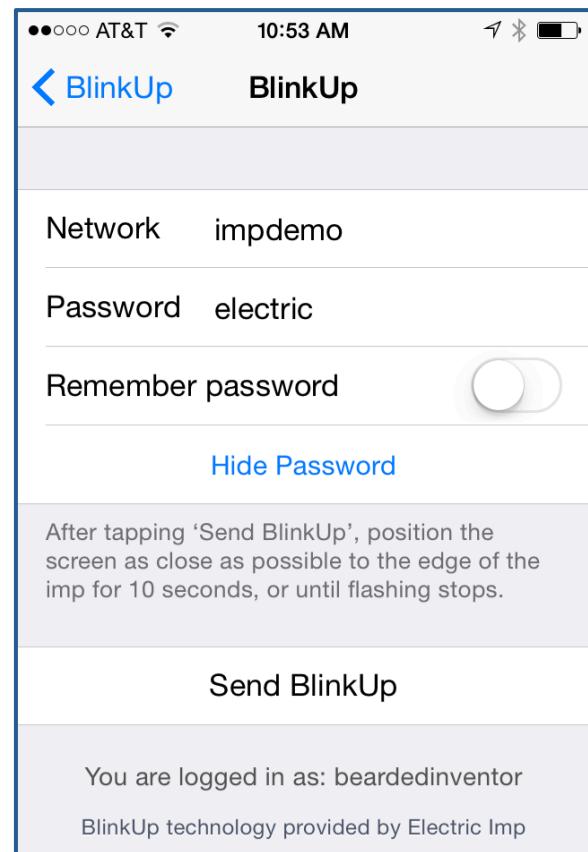


# Getting your imp online



# Getting your imp online

- Download the Electric Imp mobile app (available on iOS and Android app stores)
  - Sign into the app with your Electric Imp Account
  - Choose "Other Network"
  - Enter WiFi credentials:
    - **Network:** **impdemo**
    - **Password:** **electric**
- Power up your imp
  - The imp's internal LED must be blinking
- Click **Send BlinkUp** in the mobile app
- Hold screen of phone against end of Imp until BlinkUp completes
- Your device should now be blinking green (online)\*



\* Your imp will stop blinking after 1 minute to save power (it is still powered on)



# Example 1 – Buttons & LEDs



# Example 1 – Controlling an LED with a Button

The screenshot shows the electric imp web interface. On the left is a sidebar with a search bar, a 'Create New Model' button, and a list of devices categorized into 'Unassigned Devices', 'Active Models', and 'Inactive Models'. A specific device, '231c29b236a7c9ee', is selected and highlighted with a red arrow pointing to its gear icon. The main content area displays the message: 'There is nothing to see yet! Click on a device to start editing!'.

device name or id

Create New Model

Unassigned Devices 1

231c29b236a7c9ee

Active Models

Clyde

Forum Counter

ISS Overhead

KeenButton

Neopixel Driver

TempBug - Keen IO

Twitter Hat

Factory Firmware

Inactive Models

Code Operations Documentation Forums Status 1 beardedinventor

Hover here, then click the gear

There is nothing to see yet!  
Click on a device to  
start editing!



# Example 1 – Controlling an LED with a Button

The screenshot shows the electric imp web interface. On the left, there's a sidebar with a search bar, a 'Create New Model' button, and a list of devices categorized into 'Unassigned Devices', 'Active Models', and 'Inactive Models'. Under 'Active Models', several items are listed: Clyde, Forum Counter, ISS Overhead, KeenButton, Neopixel Driver, TempBug - Keen IO, Twitter Hat, Factory Firmware, and Inactive Models. A large orange arrow points from the text 'Click on a device to start editing!' towards the 'Associated model:' dropdown. Another orange arrow points from the same text towards the 'Save Changes' button at the bottom right of the modal. The central part of the screen displays 'Device Settings' for a device with ID '231c29b236a7c9ee'. It shows the current name '231c29b236a7c9ee' and a note: 'You must assign the device to a model in order to change the name'. Below that is a dropdown menu titled 'Associated model:' containing 'Select model below, or type to create a new model.' with a search icon. The option 'Toggle' is listed, and 'Toggle - Create New Model' is highlighted with a blue selection bar. At the bottom of the modal are two buttons: 'Delete Device' (red) and 'Save Changes' (blue).



# Example 1 – Controlling an LED with a Button

The screenshot shows the electric imp web interface with three numbered arrows indicating steps:

- 1**: Points to the left sidebar where the "Toggle" model is selected.
- 2**: Points to the main content area showing the device logs for the selected device.
- 3**: Points to the "Build and Run" button at the top of the main content area.

**Main Content Area (Step 3):**

**Device Logs - 231c29b236a7c9ee**

```
2015-02-07 07:24:06 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 15:21:15 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 15:21:15 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 18:21:36 UTC-8 [Status] Device disconnected
2015-02-08 18:32:18 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 18:32:18 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 18:48:35 UTC-8 [Status] Device disconnected
2015-02-08 18:57:18 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 18:57:18 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-09 09:31:42 UTC-8 [Device] No firmware assigned; booting to blank firmware.
```



# Example 1 – Controlling an LED with a Button

The screenshot shows the electric imp web interface with the following components:

- Left Sidebar:** Lists "Unassigned Devices" (0), "Active Models" (Clyde, Forum Counter, ISS Overhead, KeenButton, Neopixel Driver, TempBug - Keen IO), "Toggle" (selected, 231c29b236a7c9ee), and "Inactive Models" (Twitter Hat, Factory Firmware).
- Header:** Includes a search bar ("device name or id"), "Create New Model" button, navigation tabs (Code, Operations, Documentation, Forums, Status 1), and a user dropdown ("beardedinventor").
- Middle Section:** A "Toggle" model page with a "Draft" status. It features three sections:
  - Agent Code:** An orange cloud containing the text "Agent Code".
  - Device - 231c29b236a7c9ee:** An orange cloud containing the text "Device Code".
  - Device Logs - 231c29b236a7c9ee:** An orange cloud containing the text "Device Logs".
- Bottom Section:** A log window titled "Device Logs - 231c29b236a7c9ee" showing the following log entries:

```
2015-02-07 07:24:06 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 15:21:15 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 15:21:15 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 18:21:36 UTC-8 [Status] Device disconnected
2015-02-08 18:32:18 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 18:32:18 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-08 18:48:35 UTC-8 [Status] Device disconnected
2015-02-08 18:57:18 UTC-8 [Exit Code] Imp restarted, reason: wifi outage
2015-02-08 18:57:18 UTC-8 [Status] Device Booting; 1.30% program storage used
2015-02-09 09:31:42 UTC-8 [Device] No firmware assigned; booting to blank firmware.
```



# Example 1 – Controlling an LED with a Button

---

<https://github.com/beardedinventor/hackbright2k15/tree/master/toggle>

- 1) Copy the code in `toggle.device.nut` into the **device editor**
- 2) Copy the code in `toggle.agent.nut` into the **agent editor**
  
- 3) Hit ‘Build and Run’
- 4) Play around with the button to see what happens!



# Example 2 – LED API



# Example 2 – LED API

The screenshot shows the electric imp web interface. The top navigation bar includes links for Code, Operations, Documentation, Forums, Status (1), and a user account dropdown for 'beardedinventor'. The main content area is titled 'Toggle' and shows the URL <https://agent.electricimp.com/R05HOrveo4II>. The status is 'running'. On the left, a sidebar lists 'Unassigned Devices', 'Active Models' (including Clyde, Forum Counter, ISS Overhead, KeenButton, Neopixel Driver, TempBug - Keen IO, Toggle, Twitter Hat, Factory Firmware, and Inactive Models), and a 'Create New Model' button. The main panel displays the device code:

```
Agent - https://agent.electricimp.com/R05HOrveo4II running
1 server.log("Hello from the agent!");

Device - 231c29b236a7c9ee online
1 // assign pins we're using to global variables
2 led <- hardware.pin9;
3 btn <- hardware.pin1;
4
5 // track the current state of the LED in a global variable
6 state <- 0;
7
8 // configure LED and set initial state to OFF (0)
9 led.configure(DIGITAL_OUT, state);
10
11 btn.configure(DIGITAL_IN_PULLUP, function() {
12     // software debounce
13     imp.sleep(0.02);
14     // read the state
15     local btnState = btn.read();
16
17     // 1 means the button was released
18     if (btnState == 1) {
19         // flip the state
20         state = 1 - state;
21         // log a message
22         server.log("Set LED to: " + state);
23         // write the new state to the led pin
24         led.write(state);
25     }
26 });

Device Logs - 231c29b236a7c9ee
Clear Log
2015-02-09 09:45:59 UTC-8[Agent] Hello from the agent!
2015-02-09 09:45:59 UTC-8[Status] Downloading new code; 1.79% program storage used
```

An orange arrow points to the gear icon next to the device name '231c29b236a7c9ee' in the sidebar. A callout text says 'Hover here, then click the gear'.



# Example 2 – LED API

The screenshot shows the electric imp web interface. On the left, the sidebar lists various device models: Unassigned Devices, Active Models (Clyde, Forum Counter, ISS Overhead, KeenButton, Neopixel Driver, TempBug - Keen IO, Toggle, Twitter Hat), Factory Firmware, Inactive Models (2lemetry ThingFabric, APIStrat Demo, ArrayTest), and a status bar at the top.

In the center, a modal window titled "Device Settings: 231c29b236a7c9ee" is open. It shows the "Name" field set to "231c29b236a7c9ee". The "Associated model:" dropdown is open, displaying "Toggle", "LED API", and "LED API - Create New Model", with "LED API - Create New Model" highlighted. At the bottom of the modal are "Delete Device" and "Save Changes" buttons.

On the right, the main content area displays the device's code. The code is a sketch for an Arduino-like environment:

```
server.on("button", function() {
    // read the state
    local btnState = btn.read();
    // 1 means the button was released
    if (btnState == 1) {
        // flip the state
        state = 1 - state;
        // log a message
        server.log("Set LED to: " + state);
        // write the new state to the led pin
        led.write(state);
    }
});
```

Below the code, the "Device Logs" section shows the following entries:

```
2015-02-09 15:20:30 UTC-8[Agent] ERROR: wrong indexes
2015-02-09 15:20:30 UTC-8[Agent] ERROR: at _encode:140
2015-02-09 15:20:30 UTC-8[Agent] ERROR: from _oAuth1Request:157
2015-02-09 15:20:30 UTC-8[Agent] ERROR: from stream:79
2015-02-09 15:20:30 UTC-8[Agent] ERROR: from main:207
2015-02-09 15:20:30 UTC-8[Status] Downloading new code; 1.99% program storage used
2015-02-09 15:20:48 UTC-8[Agent] Opening stream for: beardedinventor
2015-02-09 15:20:48 UTC-8[Status] Device Booting; 1.99% program storage used
2015-02-09 15:28:24 UTC-8[Agent] Hello from the agent!
2015-02-09 15:28:24 UTC-8[Agent] Hello from the agent!
```



## Example 2 – LED API

---

<https://github.com/beardedinventor/hackbright2k15/tree/master/api>

- 1) Copy the code in `api.device.nut` into the **device editor**
- 2) Copy the code in `api.agent.nut` into the **agent editor**
  
- 3) Hit Build and Run
  - \* The URLs you need to control your LED should be in the **Device Logs pane**
- 4) Copy and paste the URLs into a new browser window (or cURL, etc) to control your LEDs
  - \* These are endpoints on the Internet now, so anyone with the URL can control them
  
- 5) Bonus Points – Build a webpage/script to control your/your friends' LEDs



# Example 3 – Twitter Indicator



# Example 3 – Twitter Indicator

Move your cursor over here, then click the gear

Code Operations Documentation Forums Status 1 beardedinventor ▾

device name or id Create New Model

Unassigned Devices 0

Active Models

- Clyde
- Forum Counter
- ISS Overhead
- KeenButton
- LED API
- 231c29b236a7c9ee
- Neopixel Driver
- TempBug - Keen IO
- Twitter Hat

Factory Firmware

Inactive Models

- 2lemetry ThingFabric
- APIStrat Demo
- ArrayTest

LED API  
231c29b236a7c9ee / device online / agent running / agent link

Build 4 ✓ Check ▶ Build and Run ⚡ Promote to Ops

Agent - https://agent.electricimp.com/R05H0rveo4I1 running

```
1 // when an HTTP request comes into our agent:
2 http.onrequest(function(req, resp) {
3     // always wrap in try catch
4     try {
5         if ("led" in req.query) {
6             // convert query parameter to an integer
7             // (This will throw and err if it's not an integer)
8             local state = req.query.led.tointeger();
9             // if state is 1 or 0, send a message to our device
10            if (state == 1 || state == 0) {
11                device.send("led", state);
12            }
13        }
14    }
15    // 200 OK is a pretty standard response
16    resp.send(200, "OK");
17    // if we hit an error, send a 500 response with the error
18    resp.send(500, "Internal Agent Error - " + ex);
19 }
20 });
21 });
22
23 server.log("Turn LED on: " + http.agenturl() + "?led=1");
24 server.log("Turn LED off: " + http.agenturl() + "?led=0");
```

Device - 231c29b236a7c9ee online

```
1 // Assign and configure pins we're using
2 led <- hardware.pin9;
3
4 // configure LED and set initial state to OFF (0)
5 led.configure(DIGITAL_OUT, 0);
6
7 agent.on("led", function(state) {
8     led.write(state);
9     server.log("Set LED to " + state);
10});
```

Device Logs - 231c29b236a7c9ee

Clear Log

```
2015-02-09 15:38:36 UTC-8[Agent] Turn LED on: https://agent.electricimp.com/R05H0rveo4I1?led=1
2015-02-09 15:38:36 UTC-8[Agent] Turn LED off: https://agent.electricimp.com/R05H0rveo4I1?led=0
2015-02-09 15:38:36 UTC-8[Status]Device Booting; 1.56% program storage used
```



# Example 3 – Twitter Indicator

The screenshot shows the electric imp web interface. On the left, the sidebar lists various devices and models. A large orange arrow points from the sidebar to the 'Associated model:' dropdown in the central modal window. Another large orange arrow points from the 'Save Changes' button in the modal back to the main interface.

**Device Settings: 231c29b236a7c9ee**

Name: 231c29b236a7c9ee

Associated model:

- LED API
- Twitter Indicator
- Twitter Indicator - Create New Model

Code (JavaScript):

```
1 // when
2 http.onEvent("led", function(state) {
3     if (state === "on") {
4         led.setInitialState(0);
5         led.set("on");
6     } else {
7         led.set("off");
8     }
9 });
10
11 // 200 OK is a pretty standard response
12 resp.send(200, "OK");
13
14 } catch(ex) {
15     // if we hit an error, send a 500 response with the error
16     resp.send(500, "Internal Agent Error - " + ex);
17 }
18
19 server.log("Turn LED on: " + http.agenturl() + "?led=1");
20 server.log("Turn LED off: " + http.agenturl() + "?led=0");
```

Device Logs - 231c29b236a7c9ee

```
2015-02-09 15:38:36 UTC-8 [Agent] Turn LED on: https://agent.electricimp.com/R05HOrveo4I1?led=1
2015-02-09 15:38:36 UTC-8 [Agent] Turn LED off: https://agent.electricimp.com/R05HOrveo4I1?led=0
2015-02-09 15:38:36 UTC-8 [Status] Device Booting; 1.56% program storage used
```



## Example 3 – Twitter Indicator

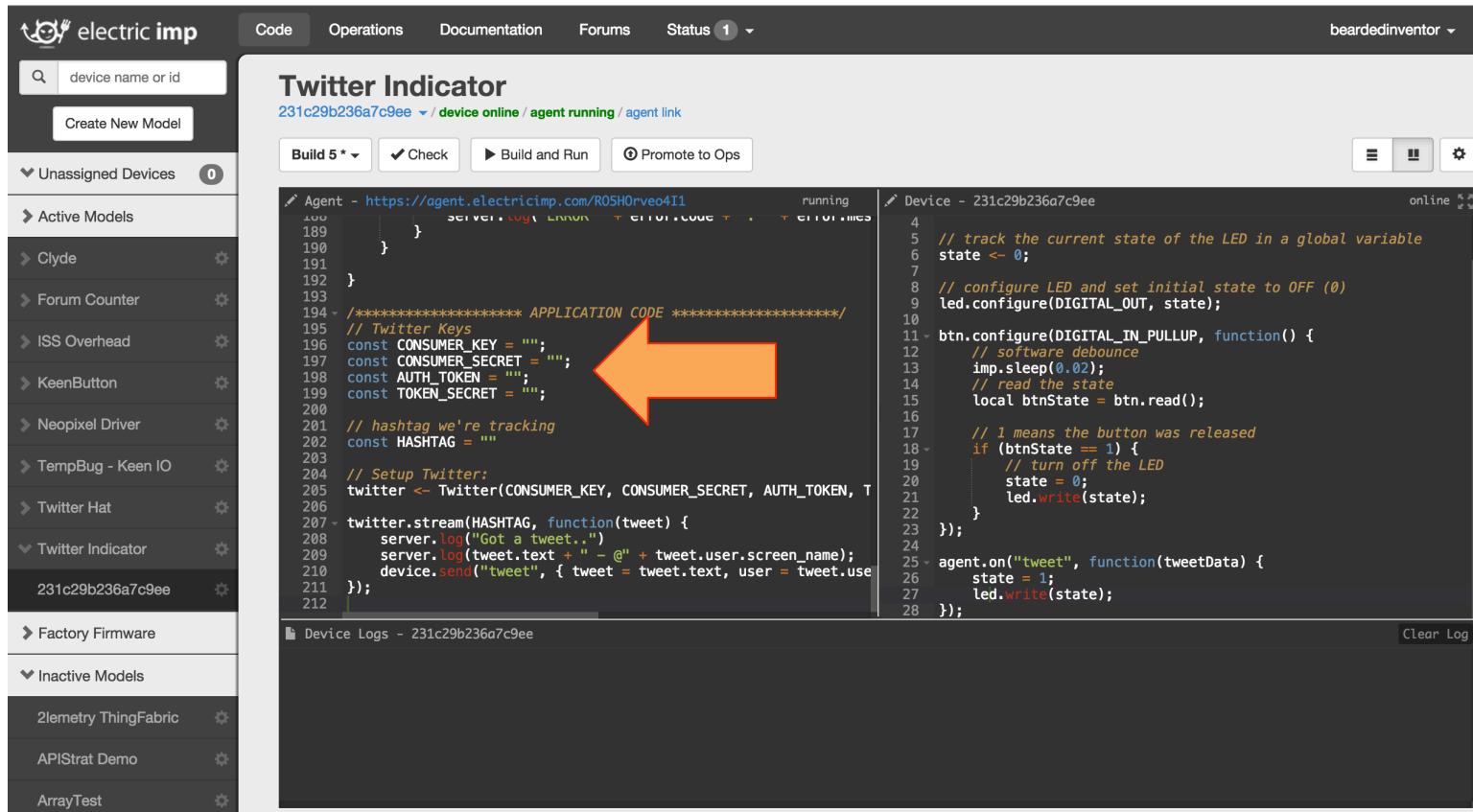
---

<https://github.com/beardedinventor/hackbright2k15/tree/master/twitter>

- 1) Copy the code in `twitter.device.nut` into the **device editor**
- 2) Copy the code in `twitter.agent.nut` into the **agent editor**
  
- 3) Setup your Twitter application
- 4) Copy the required tokens into your code
  
- 5) Hit ‘Build and Run’ and try tweeting with your tracked term to see what does



# Example 3 – Twitter Indicator



The screenshot shows the electric imp web interface with the following details:

- Header:** electric imp, device name or id, Create New Model, Status 1, beardedinventor.
- Left Sidebar:** Unassigned Devices (0), Active Models (Clyde, Forum Counter, ISS Overhead, KeenButton, Neopixel Driver, TempBug - Keen IO, Twitter Hat, Twitter Indicator, 231c29b236a7c9ee), Factory Firmware, Inactive Models (2lemetry ThingFabric, APIStrat Demo, ArrayTest).
- Central Area:**
  - Twitter Indicator:** Device ID: 231c29b236a7c9ee, Status: running.
  - Code Editor:** Shows the source code for the Twitter Indicator agent. The code includes setup for Twitter keys and a stream handler. A large orange arrow points to the application code section (lines 194-212).

```
189     }
190   }
191 }
192 }
193 */
194 // ***** APPLICATION CODE *****
195 // Twitter Keys
196 const CONSUMER_KEY = "";
197 const CONSUMER_SECRET = "";
198 const AUTH_TOKEN = "";
199 const TOKEN_SECRET = "";
200
201 // hashtag we're tracking
202 const HASHTAG = ""
203
204 // Setup Twitter:
205 twitter <- Twitter(CONSUMER_KEY, CONSUMER_SECRET, AUTH_TOKEN, T
206
207 - twitter.stream(HASHTAG, function(tweet) {
208   server.log("Got a tweet..")
209   server.log(tweet.text + " - @" + tweet.user.screen_name);
210   device.send("tweet", { tweet: tweet.text, user: tweet.use
211 });
212 }
```
  - Device Logs:** Device Logs - 231c29b236a7c9ee, Clear Log.



## Example 3 – Twitter Indicator

<http://apps.twitter.com>

The screenshot shows the Twitter Application Management interface. At the top, there's a blue header bar with the Twitter logo and the text "Application Management". On the right side of the header is a user profile icon. Below the header, the main content area has a title "Twitter Apps". A central message box contains the text "You don't currently have any Twitter Apps." and a "Create New App" button. An orange arrow points to the "Create New App" button. At the bottom of the page, there's a footer with links for "About", "Terms", "Privacy", and "Cookies", and a copyright notice "© 2015 Twitter, Inc.". There's also a "Tweet" button with a count of "4,069".



# Example 3 – Twitter Indicator

Twitter Application Management

Create an application

**Application Details**

**Name \***  
beardedinventor\_agents

*Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.*

**Description \***  
Twitter app for Electric Imp agents

*Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.*

**Website \***  
<http://www.electricimp.com>

*Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.  
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)*

**Callback URL**

*Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.*



# Example 3 – Twitter Indicator

Developer Agreement

Last Update: October 22, 2014.

This Twitter Developer Agreement ("Agreement") is made between you (either an individual or an entity, referred to herein as "you") and Twitter, Inc., on behalf of itself and its worldwide affiliates (collectively, "Twitter") and governs your access to and use of the Licensed Material (as defined below).

PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY, INCLUDING WITHOUT LIMITATION ANY LINKED TERMS AND CONDITIONS APPEARING OR REFERENCED BELOW, WHICH ARE HEREBY MADE PART OF THIS LICENSE AGREEMENT. BY USING THE LICENSED MATERIAL, YOU ARE AGREEING THAT YOU HAVE READ, AND THAT YOU AGREE TO COMPLY WITH AND TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND ALL APPLICABLE LAWS AND REGULATIONS IN THEIR ENTIRETY WITHOUT LIMITATION OR QUALIFICATION. IF YOU DO NOT AGREE TO BE BOUND BY THIS AGREEMENT, THEN YOU MAY NOT ACCESS OR OTHERWISE USE THE LICENSED MATERIAL. THIS AGREEMENT IS EFFECTIVE AS OF THE FIRST DATE THAT YOU USE THE LICENSED MATERIAL ("EFFECTIVE DATE").

IF YOU ARE AN INDIVIDUAL REPRESENTING AN ENTITY, YOU ACKNOWLEDGE THAT YOU HAVE THE APPROPRIATE AUTHORITY TO ACCEPT THIS AGREEMENT ON BEHALF OF SUCH ENTITY. YOU MAY NOT USE THE LICENSED MATERIAL AND MAY NOT ACCEPT THIS AGREEMENT IF YOU ARE NOT OF LEGAL AGE TO FORM A BINDING CONTRACT WITH TWITTER, OR YOU ARE

Yes, I agree

Create your Twitter application



# Example 3 – Twitter Indicator

Application Management

Your application has been created. Please take a moment to review and adjust your application's settings.

## beardedinventor\_agents

Test OAuth

Keys and Access Tokens   Permissions

Twitter app for Electric Imp agents  
<http://www.electricimp.com>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to [authenticate](#) requests to the Twitter Platform.

Access level	Read-only ( <a href="#">modify app permissions</a> )
Consumer Key (API Key)	gcc9D1hPIBTt6uPtnfdLfzhXN ( <a href="#">manage keys and access tokens</a> )



# Example 3 – Twitter Indicator

Application Management

beardedinventor\_agents

Test OAuth

Details    Settings    Keys and Access Tokens    Permissions

**Access**

What type of access does your application need?

*Read more about our [Application Permission Model](#).*

Read only

Read and Write

Read, Write and Access direct messages

**Note:**

*Changes to the application permission model will only reflect in access tokens obtained after the permission model change is saved. You will need to re-negotiate existing access tokens to alter the permission level associated with each of your application's users.*

Update Settings



# Example 3 – Twitter Indicator

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	gcc9D1hPIBTt6uPtnfdLfzhXN
Consumer Secret (API Secret)	FKCc6KbTQyfmXm7dTiv2t49R3a8witzTBQyLA9pjnfJSoY8aCW
Access Level	Read-only ( <a href="#">modify app permissions</a> )
Owner	BeardedInventor
Owner ID	24045908

## Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

## Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

## Token Actions

[Create my access token](#)



# Example 3 – Twitter Indicator

Keep the "Consumer Secret" secret. This key should never be human-readable in your application.

Consumer Key	gcc9D1hPIBTt6uPtnfdLfzhXN
Consumer Secret (API Secret)	FKCc6KbTQyfmXm7dTiv2t49R3a8witzTBQyLA9pjnfJSoY8aCW
Access Level	Read, write, and direct messages ( <a href="#">modify app permissions</a> )
Owner	BeardedInventor
Owner ID	24045908

## Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

## Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	24045908-CboJCNH2P2uC8BzMzgbozagNsstFrVzfZ6Oe8izli
Access Token Secret	KVNSLxtwqYrSp1ezlmNjgkahY1X4hqJCCBj3ybvvmX2n
Access Level	Read, write, and direct messages
Owner	BeardedInventor
Owner ID	24045908



# Example 3 – Twitter Indicator

The screenshot shows the electric imp web interface with the "Twitter Indicator" model selected. The left sidebar lists various device models, with "Twitter Indicator" highlighted by a red arrow. The main workspace displays the model's code. Two orange arrows point to specific lines of code: one to the consumer key and secret constants, and another to the "agent.on('tweet', function(tweetData)" event handler.

```
Agent - https://agent.electricimp.com/R05H0rveo4I1      running
188
189 }
190 }
191 }
192 }
193 */
194 // ***** APPLICATION CODE *****/
195 // Twitter Keys
196 const CONSUMER_KEY = "gcc9D1hPIBTt6uPtnfdLfzhXN";
197 const CONSUMER_SECRET = "FKCc6KbTQyfmXn7d1v2t49R3a8witzTBQyLA9
198 const AUTH_TOKEN = "24045908-CboJCNH2P2uC8BzMzbogazNsstFrVzfZ6
199 const TOKEN_SECRET = "KVNSLxtwgYrSpiezlmjgkahyY1X4hdJCCh3ybvv
200
201 // hashtag we're tracking
202 const HASHTAG = "beardedinventor"
203
204 // Setup Twitter:
205 twitter <- Twitter(CONSUMER_KEY, CONSUMER_SECRET, AUTH_TOKEN, T
206
207 - twitter.stream(HASHTAG, function(tweet) {
208   server.log("Got a tweet..")
209   server.log(tweet.text + " - @" + tweet.user.screen_name);
210   device.send("tweet", { tweet = tweet.text, user = tweet.use
211 });
212
Device - 231c29b236a7c9ee
4
5 // track the current state of the LED in a global variable
state <- 0;
6
7 // configure LED and set initial state to OFF (0)
led.configure(DIGITAL_OUT, state);
8
9 btn.configure(DIGITAL_IN_PULLUP, function() {
10   // software debounce
11   imp.sleep(0.02);
12   // read the state
13   local btnState = btn.read();
14
15   // 1 means the button was released
16   if (btnState == 1) {
17     // turn off the LED
18     state = 0;
19     led.write(state);
20   }
21 });
22
23 agent.on("tweet", function(tweetData) {
24   state = 1;
25   led.write(state);
26 });
27
28
Device Logs - 231c29b236a7c9ee
Clear Log
2015-02-09 15:20:48 UTC-8[Agent] Opening stream for: beardedinventor
2015-02-09 15:20:48 UTC-8[Status]Device Booting; 1.99% program storage used
```



# Take home project - TempBug



# TempBug + SparkFun Data

---

[instructables.com/id/TempBug-internet-connected-thermometer](https://www.instructables.com/id/TempBug-internet-connected-thermometer)

Follow the circuit wiring diagram

- Ignore the battery clip and capacitor if you're powering with USB
- To build the battery version you'll need a soldering iron, so check out a local hackerspace!!



## Additional Resources

---

Getting Started

[electricimp.com/docs/gettingstarted](http://electricimp.com/docs/gettingstarted)

API Docs

[electricimp.com/docs/api](http://electricimp.com/docs/api)

Squirrel Docs

[electricimp.com/docs/squirrel](http://electricimp.com/docs/squirrel)

GitHub

[github.com/electricimp](http://github.com/electricimp)

Community Blog

[community.electricimp.com](http://community.electricimp.com)

Forums

[forums.electricimp.com](http://forums.electricimp.com)

# Questions?

