# Table of Contents

## Overview 3

### Adafruit IO Setup 3
- Creating the Digital Feed 4
- Adding the Toggle Block 5

### Arduino Wiring 6
- Arduino Wiring 6

### Arduino Setup 7

### Arduino Network Config 8
- WiFi Config 8
- FONA Config 9
- Ethernet Config 9

### Arduino Code 10

### Python Wiring 12
- Parts 12
- Wiring 13

### Python Setup 14
- Update your Pi and Python 14
- Make sure you’re using Python 3! 15
- Install Python Libraries 15
- Installing Adafruit Blinka Library 15
- Installing Adafruit IO Python Library 16
- Downloading Example Code 16

### Python Code 17
- Running the Code 18
- Code 19

### Adafruit IO FAQ 20
- Encountering an issue with your Adafruit IO Arduino Project? 20
Overview

This guide is part of a series of guides that cover the basics of using Adafruit IO. It will show you how to turn a LED on and off from Adafruit IO using any modern web browser.

If you haven’t worked your way through the Adafruit IO feed and dashboard basics guides, you should do that before continuing with this guide so you have a basic understanding of Adafruit IO.

- Adafruit IO Basics: Feeds
- Adafruit IO Basics: Dashboards

You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

- Adafruit Feather HUZZAH ESP8266 Setup Guide

If you have gone through all of the prerequisites for your selected hardware, you are now ready to move on to the Adafruit IO setup steps that are common between all of the hardware choices for this project. Let’s get started!

Adafruit IO Setup

The first thing you will need to do is to login to Adafruit IO and visit the Settings page.
Click the VIEW AIO KEY button to retrieve your key.

A window will pop up with your Adafruit IO. Keep a copy of this in a safe place. We'll need it later.

Creating the Digital Feed

Next, you will need to create a feed called Digital. If you need help getting started with creating feeds on Adafruit IO, check out the Adafruit IO Feed Basics guide (https://adafruit.ioA).
Adding the Toggle Block

Next, add a new Toggle Block to a new or existing dashboard. Name the block whatever you would like, and set On Text to a value of 1 and Off Text to a value of 0. Make sure you have selected the Digital feed as the data source for the toggle.

If you need help getting started with Dashboards on Adafruit IO, check out the Adafruit IO Dashboard Basics guide (https://adafruit.io/f5m).

When you are finished editing the form, click Create Block to add the new block to the dashboard.
Next, we will look at wiring the circuit.

Arduino Wiring

You will need the following parts for this tutorial:

- 1x Adafruit IO compatible Feather
- 2x jumper wires
- 1x 560 ohm resistor
- 1x 10mm LED

You will need to connect the following pins to the LED and resistor:

- Feather GND to LED cathode (short leg)
- Feather Pin 5 to one leg of the 560 ohm resistor
- LED anode (long leg) to the second leg of the 560 ohm resistor

Note: Resistors are not polarized, so the 560 ohm resistor can be connected to the circuit in either direction.
Next, we're going to set up our Arduino.

Arduino Setup

You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

- Adafruit Feather HUZZAH ESP8266 Setup Guide

You will need to make sure you have at least version 2.3.1 of the Adafruit IO Arduino library installed before continuing.

For this example you will need to open the adafruitio_07_digital_out example in the Adafruit IO Arduino library.
Next, we will look at the network configuration options in the sketch.

Arduino Network Config

To configure the network settings, click on the config.h tab in the sketch. You will need to set your Adafruit IO username in the IO_USERNAME define, and your Adafruit IO key in the IO_KEY define.

WiFi Config

WiFi is enabled by default in config.h so if you are using one of the supported WiFi boards, you will only need to modify the WIFI_SSID and WIFI_PASS options in the config.h tab.
FONA Config

If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in config.h

```c
// comment out the following two lines if you are using fona or ethernet
#include "AdafruitIO_WiFi.h"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Next, remove the comments from both of the FONA config lines in the FONA section of config.h to enable FONA support.

```c
// uncomment both fona config lines
#include "AdafruitIO_FONA.h"
AdafruitIO_FONA io(IO_USERNAME, IO_KEY);
```

Ethernet Config

If you wish to use the Ethernet Wing to connect to Adafruit IO, you will need to first comment out the WiFi support in config.h

```c
// uncomment both ethernet config lines
#include "AdafruitIO_WiFi.h"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```
Next, remove the comments from both of the Ethernet config lines in the Ethernet section of config.h to enable Ethernet Wing support.

Next, we will look at how the example sketch works.

Arduino Code

The adafruitio_07_digital_out example uses digital pin 5 by default on all boards, and that can be modified by changing the LED_PIN define.

Note: If you are using the WICED Feather, you will need to change the LED_PIN define to PC5 instead of the default setting of 5.

```cpp
#define LED_PIN 5
```

The next chunk of code sets up an Adafruit IO Feed instance for a feed called digital.

```cpp
// set up the 'digital' feed
AdafruitIO_Feed *digital = io.feed("digital");
```
In the setup function, we set the LED_PIN as a digital output, and connect to Adafruit IO. We also attach a function called handleMessage to the digital feed that will be called whenever your device receives messages for that feed.

The code will wait until you have a valid connection to Adafruit IO before continuing with the sketch. If you have any issues connecting, check config.h for any typos in your username or key.

```cpp
void setup() {
    // set led pin as a digital output
    pinMode(LED_PIN, OUTPUT);

    // start the serial connection
    Serial.begin(115200);

    // wait for serial monitor to open
    while(!Serial);

    // connect to io.adafruit.com
    Serial.print("Connecting to Adafruit IO");
    io.connect();

    // set up a message handler for the 'digital' feed.
    // the handleMessage function (defined below) will be called whenever a message is
    // received from adafruit io.
    digital-&gt;onMessage(handleMessage);

    // wait for a connection
    while(io.status() &lt; AIO_CONNECTED) {
        Serial.print(".");
        delay(500);
    }

    // we are connected
    Serial.println();
    Serial.println(io.statusText());
    digital-&gt;get();
}
```

Next, we have the main `loop()` function. The first line of the loop function calls `io.run();`; this line will need to be present at the top of your loop in every sketch. It helps keep your device connected to Adafruit IO, and processes any incoming data.

```cpp
void loop() {
    // io.run(); is required for all sketches.
    // it should always be present at the top of your loop
    // function. it keeps the client connected to
    // io.adafruit.com, and processes any incoming data.
    io.run();
}
```

The final chunk of code is the handleMessage function. This is the function that is called whenever the digital feed gets a message.
We use the `data->toPinLevel()` function to convert the incoming data to either LOW or HIGH, and set the state of the LED_PIN to that value.

```cpp
// this function is called whenever an 'digital' feed message
// is received from Adafruit IO. it was attached to
// the 'digital' feed in the setup() function above.
void handleMessage(AdafruitIO_Data *data) {
    Serial.print("received <- ");
    if(data-&gt;toPinLevel() == HIGH)
        Serial.println("HIGH");
    else
        Serial.println("LOW");

    // write the current state to the led
digitalWrite(LED_PIN, data-&gt;toPinLevel());
}
```

Upload the sketch to your board, and open the Arduino Serial Monitor. Your board should now connect to Adafruit IO.

```
Connecting to Adafruit IO....
Adafruit IO connected.
```

Toggle the button on your Adafruit IO dashboard, and you should see the following in the Arduino Serial Monitor.

```
received &lt;- HIGH
received &lt;- LOW
received &lt;- HIGH
received &lt;- LOW
```

You can now toggle the button on your Adafruit IO dashboard, and you should see your LED turn on and off.

---

**Python Wiring**

We're going to use a combination of the Adafruit IO Client Library and Adafruit's CircuitPython to control a Raspberry Pi over Adafruit IO.

**Parts**
1 x Raspberry Pi 3 - Model B+
The Raspberry Pi is a small linux board compatible with Adafruit IO projects.

https://www.adafruit.com/product/3775

If you're following along with a Raspberry Pi, we use the T-Cobbler Plus for the IO Basics Projects. This add-on prototyping board lets you easily connect a Raspberry Pi (Raspberry Pi Model Zero, A+, B+, Pi 2, Pi 3) to a solderless breadboard:

1 x Assembled Pi T-Cobbler Plus
GPIO Breakout - Pi A+, B+, Pi 2, Pi 3, Zero

https://www.adafruit.com/product/2028

1 x Green LED
Diffused Green 5mm LED

https://www.adafruit.com/product/298

1 x Jumper Wires
Breadboarding wire bundle.

https://www.adafruit.com/product/153

Wiring

You'll need to make the following connections:

• Pi GND to LED Cathode (short LED leg).
• Pi Pin 5 to one leg of the 560 ohm resistor.
• LED Anode (long LED leg) to the other leg of the 560 ohm resistor.
  ◦ note: resistors are not polarized, so the 560 ohm resistor can be connected to the circuit in either direction.
Python Setup

If you're following along with a Raspberry Pi, Beaglebone or any other supported small linux computer, we'll use a special library called adafruit_blinka (https://adafruit.it/BJS) (named after Blinka, the CircuitPython mascot (https://adafruit.it/BJT)) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides. It's CircuitPython, on Pi!

Update your Pi and Python

The latest Raspbian (currently this is 'Stretch') is required for the installation of Adafruit IO + Blinka.

In this page we'll assume you've already gotten your Raspberry Pi up and running and can log into the command line.

Go ahead and ssh into your Raspberry Pi via terminal or a ssh client:

    ssh pi@raspberrypi.local

Run the standard updates:

    sudo apt-get update
**sudo apt-get upgrade**

and

**sudo pip3 install --upgrade setuptools**

Make sure you're using Python 3!

The default python on your computer may not be python 3. Python 2 is officially discontinued and all our libraries are Python 3 only.

We'll be using **python3** and **pip3** in our commands, use those versions of python and pip to make sure you're using 3 and not 2

Install Python Libraries

Installing Adafruit Blinka Library

Now you're ready to install all the python support

Run the following command to install the Raspberry PI GPIO library:

```
pip3 install RPI.GPIO
```

Run the following command to install adafruit_blinka

```
pip3 install adafruit-blinka
```
The computer will install a few different libraries such as `adafruit-pureio` (our ioctl-only i2c library), `spidev` (for SPI interfacing), `Adafruit-GPIO` (for detecting your board) and of course `adafruit-blinka`

**Installing Adafruit IO Python Library**

![Adafruit IO Python Library](image)

We'll also need to install the [Adafruit IO Python Client Library](https://adafruit.io/eli) to communicate with Adafruit IO.

Run the following command to install the Adafruit IO Client for Python:

```bash
pip3 install adafruit-io
```

If the installation gives you 'insufficient permissions' errors, add 'sudo' before the call to `pip3`

**Downloading Example Code**

The example code is contained within the Python IO Client's examples/basics subdirectory.

Navigate to the root directory of your Pi:

```bash
cd ~
```

Then, download the latest version of the adafruit/io-client-python repository by running:

```bash
git clone https://github.com/adafruit/io-client-python.git
```

Navigate to that folder's example folder for the examples:
That's it! We're all set up.

Next, let's upload some code and learn how it works.

---

**Python Code**

Before we run the script, we'll need to change `ADAFRUIT_IO_USERNAME` and `ADAFRUIT_IO_KEY` to the username and key for your Adafruit IO account.

- If you need the AIO Key, navigate to your Adafruit IO Profile (https://adafruit.io/BmD)

```python
# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'
```

Next, we'll create an instance of the Adafruit IO Client and set up the feed we created earlier.

```python
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
digital = aio.feeds('digital')
```

The next chunk of code sets up the LED to digital pin 5 (`board.D5`) and sets the direction of the LED to an output.

```python
# led set up
led = digitalio.DigitalInOut(board.D5)
led.direction = digitalio.Direction.OUTPUT
```

To set up the led, assign it to digital pin 5:

```python
led = digitalio.DigitalInOut(board.D5)
```

We want the LED to light up. To do this, its pin needs to be set to an output:

```python
led.direction = digitalio.Direction.OUTPUT
```
Inside the `while True` loop, the code retrieves the current value of the feed and sets the LED to the value.

```python
while True:
    data = aio.receive(digital.key)
    if int(data.value) == 1:
        print('received &lt;- ON\n')
    elif int(data.value) == 0:
        print('received &lt;- OFF\n')

    # set the LED to the feed value
    led.value = int(data.value)

    # timeout so we dont flood adafruit-io with requests
    time.sleep(0.5)
```

Running the Code

Make sure you're within the `/io-client-python/examples/basics` directory.

If you're not sure which directory you're in, you can check this by running `pwd` and you should see the following output from your terminal:

```
~/io-client-python/examples/basics
```

Let's run the script. In your terminal, run:

```
python3 digital_in.py
```

Toggle the button on your Adafruit IO dashboard, and you should see the following in the terminal of your Pi.
You can now toggle the button on your Adafruit IO dashboard, and you should see your LED turn on and off.

Code

```python
# 'digital_out.py'
# Example of turning on and off a LED from the Adafruit IO Python Client

# Import standard python modules
import time

# import Adafruit Blinka
import digitalio

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed, RequestError

# Set to your Adafruit IO key.
# Remember, your key is a secret, so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'

# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

try:
    digital = aio.feeds('digital')
except RequestError:
    feed = Feed(name='digital')
    digital = aio.create_feed(feed)

# led set up
led = digitalio.DigitalInOut(board.D5)
led.direction = digitalio.Direction.OUTPUT

while True:
    data = aio.receive(digital.key)
    if int(data.value) == 1:
        print('received <- ON\n')
    elif int(data.value) == 0:
        print('received <- OFF\n')

    # set the LED to the feed value
    led.value = int(data.value)

    # timeout so we dont flood adafruit-io with requests
    time.sleep(0.5)
```
Encountering an issue with your Adafruit IO Arduino Project?

If you're having an issue compiling, connecting, or troubleshooting your project, check this page first.

Don't see your issue? Post up on the Adafruit IO Forum with your issue (https://adafru.it/plC).

I encounter the following error when compiling my sketch:

```
fatal error: Adafruit_MQTT.h: No such file or directory, #include "Adafruit_MQTT.h"
```

The Adafruit IO Arduino library is dependent on our Adafruit IO MQTT Library.

To resolve this error, from the Arduino IDE, navigate to the Manage Libraries... option in the Sketch -> Include Library menu.

To resolve this error, from the Arduino IDE, navigate to the Manage Libraries... option in the Sketch -> Include Library menu.
My Serial Monitor prints "..." endlessly after the "Connecting to Adafruit IO" message

Your board is not connecting to Adafruit IO, but why? Let's find out:

First, check in `config.h` that you have the correct `IO_USERNAME`, `IO_KEY`, `WIFI_SSID`, and `WIFI_PASS` are set correctly.

Next, we're going to modify the while loop which waits for an IO connection in your sketch. Change the line in the status check loop from `Serial.println(.)` to `Serial.println(io.statusText());`

```cpp
// wait for a connection
while(io.status() < AIO_CONNECTED) {
  Serial.println(io.statusText());
  delay(500);
}
```

Verify and re-upload the sketch. If you're receiving a Network disconnected error message, the board is not able to talk to the internet. Re-check your hardware, connections, and router settings.

If it's still not showing Adafruit IO connected, check the IO status on the Adafruit Status page (https://adafruit.it/Oc0) to make sure the service is online.

My data isn't displaying, is Adafruit IO's [service/MQTT/API] down?

Possibly - you can check IO status on the Adafruit Status page (https://adafruit.it/Oc0).

Is my data being sent properly? Am I sending too much data?

There's a monitor page built-into Adafruit IO (https://adafruit.it/DOK) which provides a live view of incoming data and error messages. Keep this page open while you send data to your Adafruit IO devices to monitor data and errors.