Adafruit IO and Python GUI

Martin Wirz, Coder and Maker at my own Code&More

martin.w.wirz@gmail.com

https://wirznet.internet-box.ch/code

https://github.com/trimchess

This tutorial introduces in how to handle the Adafruit IO system. It describes an application to steer actors from a Python Tkinter GUI and from the Adafruit dashboard. MQTT subscribers update the GUI and the GPIO port so you can steer your actors from "all over the world".

This tutorial was created during my introductory project in the Adafruit messaging system.

All code is available in my github project (trimchess/ledcmd)

Content

Prequisites

It is assumed you have installed a Raspberry Pi with all necessary tools and toys to steer a GPIO output from python. This document doesn't include any instructions on how to install a Pi or python components.

Hardware

Pi3+ with stretch but it should work with all other Pi's

PC with Windows 10

Adafruit IO configuration

A user account must be created for the Adafruit IO system (https://io.adafruit.com/). As a result of the registration, you get a username and an Adafruit IO key (AIO key). This information you need in your scripts to connect to Adafruit IO.

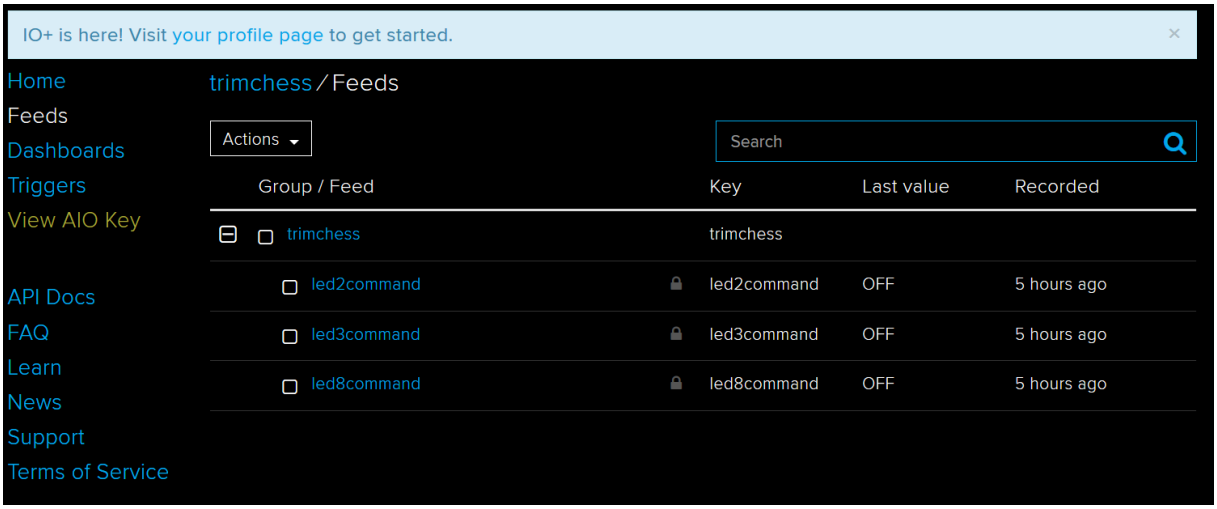
In the next steps, you must create feeds, a dashboard and buttons for the Adafruit Web GUI.

Feeds must be created in the feeds section of your account. Follow the Adafruit tutorials.

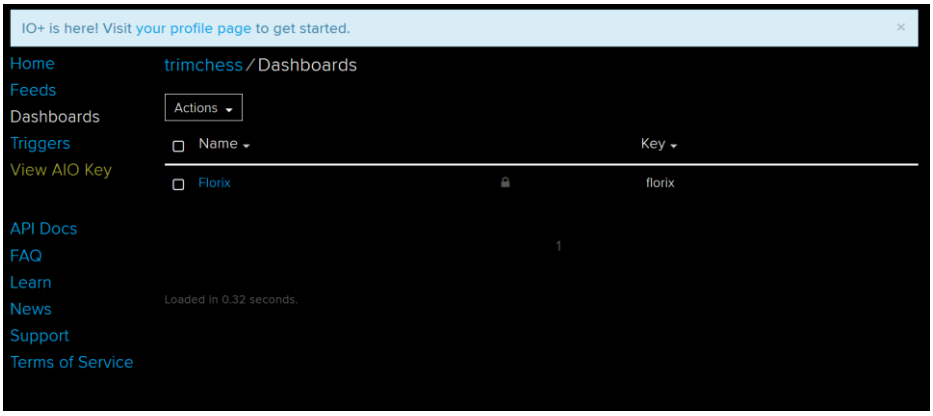
Feednames for the project

led1command, led2command, …., led8command

or less than eight but the feed names must be led<n>command; n=1 … 8) or you have to adapt the code.

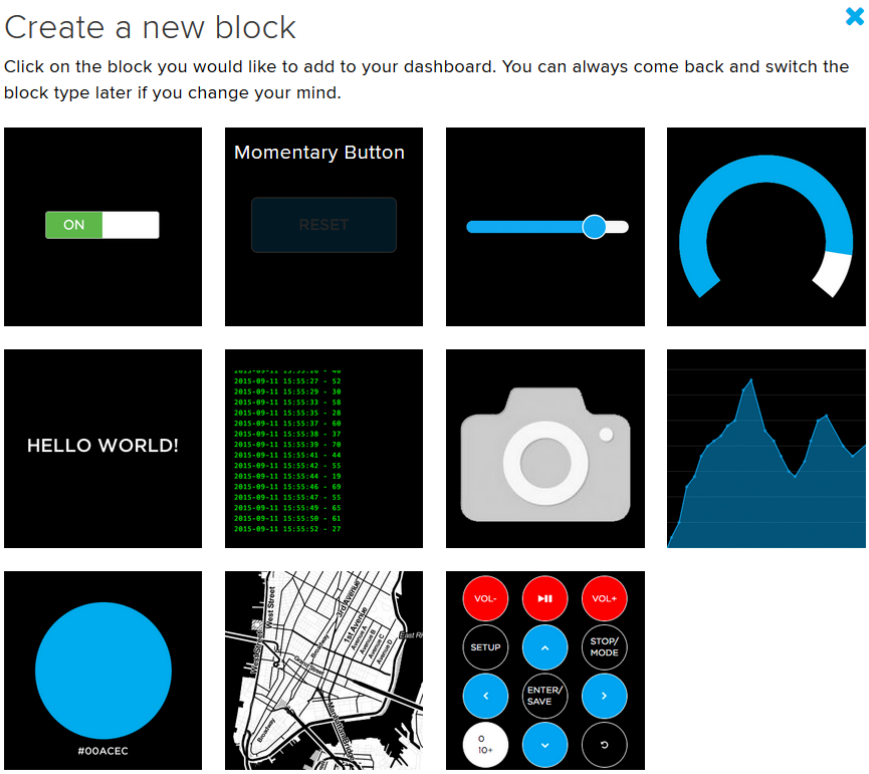


A dashboard should be created in your Adafruit user account. Follow the Adafruit tutorials.



Toggle blocks for the buttons are created in the Dashboard (LED 1, LED 2, LED 8, associated to the feeds led1command, led2command and led8command.

In your dashboard, you can create Blocks.with the + Button.

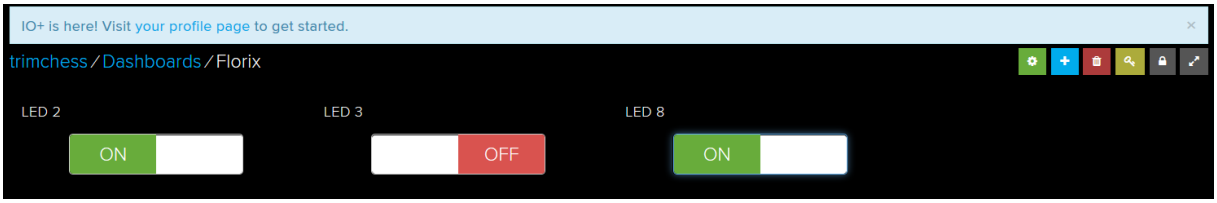


Select the Toggle Button symbol

Give the Button a name (LED 1, LED 2, LED 3) and select the appropriate feed for each button).

At the end, you have 3 buttons, one for each feed.

From this dashboard, you can switch your outputs by clicking the buttons.



Wiring the Raspberry PI GPIO

Wire the GPIO with leds, do not forget to use limiting resistances. In my project I used the following setup:

#setup GPIO

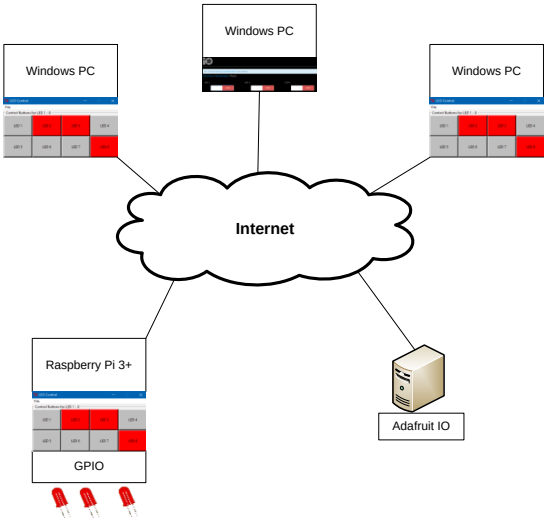
led_2 = GPIO 25 (header pin 22) associated to feed led2command

led_3 = GPIO 24 (header pin 18) associated to feed led3command

led_8 = GPIO 27 (header pin 13) associated to feed led8command

Python modules

Overview



Modules

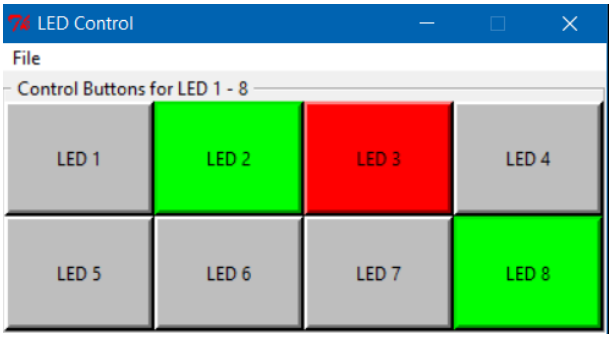
The developed software has three parts

- A GUI component with an integrated mqtt subscriber and feed steering functions
  - mainledgui.py, GUI
  - ledmqttsubs.py, mqtt subscriber
  - ledcmd.py, set/get feed data
- A component to switch a GPIO port when notified by the Adafruit IO system. This component is only used on the Raspberry Pi which steers the hardware
  - ledcmdmqtt.py

A module adafruitkey.py with the Adafruit IO data user name and AIO key

ADAFRUIT_IO_KEY	your adafruit_io_key
ADAFRUIT_IO_USERNAME	your adafruit_io_username

GUI



The GUI has static 8 buttons (LED_1 … LED_8)

Grey coloured buttons are not associated with a feed / GPIO port

Red coloured buttons are associated with a feed / GPIO port, state = OFF (LOW)

Green coloured buttons are associated with a feed / GPIO port, state = ON (HIGH)

mainledgui*.py*

The GUI main component. The component has its own mqtt subscription so it can work standalone on a client (without GPIO part). After creating and initializing the widgets and the mqtt subscription, the component waits for button clicks (method `on_buttonClick(self,event)`).

init:

`gui = GUI()`	Creates a GUi instance, creates the widgets and inizialize the button colours.

`mqtt.init(gui)`	Initialize the mqtt subscription with a reference to the mainledgui component. So a feed message can update the GUI.
`mqtt.run()`	Starts the mqtt loop
`gui.mainloop()`	GUI for ever loop (main loop)

The class methods have a short documentation in the class itself

ledmqttsubs*.py*

mqtt listener for the GUI. Main methode is

`on_message()`	Updates the GUI's button colours depending on the received message (feed, state of feed

ledcmd.py

Helper to send and read feed data

`sendFeed(self,ledFeed)`	Sends data to a feed according to a button click
`getFeedState(self,ledFeed)`	Reads the actual state of a feed

Feed subscription for GPIO switching

ledcmdmqtt.py

Subscribes to feeds and switches a GPIO port when receiving data from a subscribed feed. Can run standalone on a Raspberry with GPIO connections.

Initializing the GPIO

```
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_2, GPIO.OUT)
GPIO.setup(led_3, GPIO.OUT)
GPIO.setup(led_8, GPIO.OUT)
```

mqtt connection and subscription

```
client.connect()
client.loop_background()
```

mqtt callbacks

```
connected(client)
```

Called after succssfull connection. Subscribes the feeds

```
message(client, feed_id, payload)
```

Called when a message from a feed is received. Parse the message and switches the appropriate GPIO with the appropriate value.

`disconnected(client)`

Called when a disconnect event is received. Tries to reconnect to Adafruit IO.

Let it run

On the Raspberry Pi with GPIO access start

```
python ledcmdmqtt.py
```

```
python mainledgui.py
```

On Windows PC or other Linux clients only

```
python mainledgui.py
```

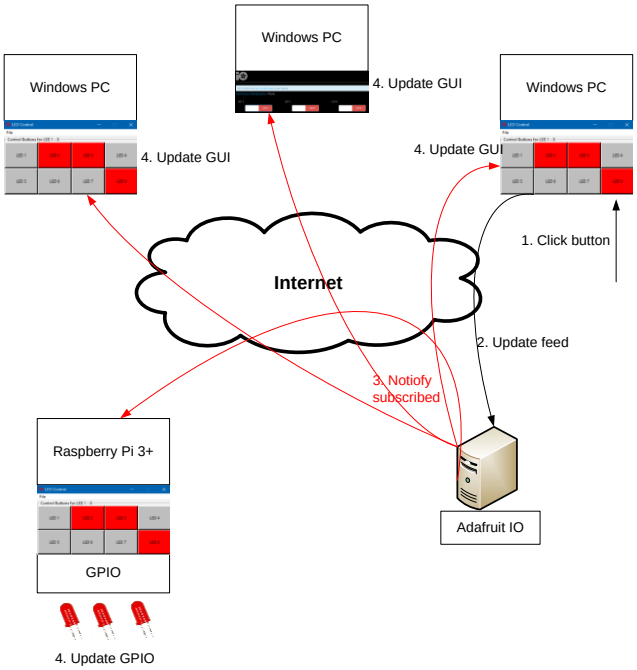
Click buttons and check the parallel update of LED's GUI's and the feeds in your Adafruit dashboard.

Message flow

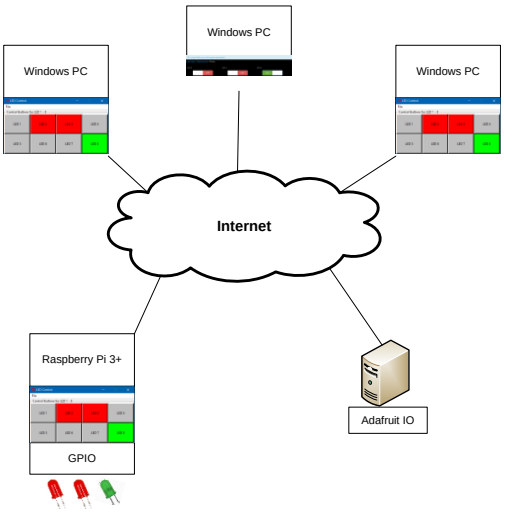
See below several diagrams depending on the actions after a button click:

- button click
- feed update
- subscriber update
- GUI update (Tkinter GUI and Adafruit Web GUI)
- GPIO update

View at start



View after update of GUI and GPIO



Detailed message flow