A Sillier Mousetrap: Logging Mouse Data to Adafruit IO with the Raspberry Pi

Created by Brennen Bearnes


Last updated on 2021-11-15 06:27:29 PM EST
# Table of Contents

## Overview

- Before You Begin 5

## Sensor Setup

## Software

- io-client-python 9
- Adafruit-MouseLogger 9

## Trap Setup

## Logging and Displaying Data on Adafruit IO

- Add Your Key to mousetrap.cfg 19
- Create Feeds for Each Sensor 20
- Create a Dashboard and Blocks to Display Feeds 21

## Optional SQLite Logging

- SQWhat? 22
- Installation & Database Setup 23
- The Basic Idea 23
- Log Some Mouses with Python 25

## Trap Placement & Next Steps

- Mouse-related Considerations 26
- Hardware Considerations 27
- Animal Welfare & Human Safety 27
- Final Pythonic Mouse Thoughts 28
I live in an old, less-than-airtight house not far from the Colorado foothills, and sometimes doors and windows get left open. Mostly this isn't a problem, as long as you don't mind the occasional wolf spider, but sometimes (and this is one of those times) there are mice.

Mice are cute, but they're also disgusting, noisy little disease vectors who steal my popcorn and run all over the counters being gross at three in the morning, so I bought some traps.
As it happens, I also just got an Adafruit IO beta account (https://adafruit.it/f58), and I was curious how hard it'd be to automatically log some concrete data about my mouse problem. Despite a few hiccups, it turns out to work pretty well.

To replicate this project, you'll need:

- A suitable mousetrap (https://adafruit.it/f59) - You could probably adapt most mechanical traps to the simple sensors used here with a little work; I went with a model of live trap that I've had good luck with in the past.
- A baseplate and fasteners for the trap - more on this in a moment.
- Peanut butter, tasty dry noodles, or other suitable bait.
- A Raspberry Pi (https://adafruit.it/ejq) - I used a model B+, but nearly any version should work.
- Internet connectivity, plus a WiFi dongle (http://adafruit.it/1012) or ethernet cable that reaches to your trap location.
- An Adafruit IO account - since IO is currently in limited beta (https://adafruit.it/f5a), we'll also explore how to log data to SQLite (https://adafruit.it/f5b) (and hey, that's pretty handy anyhow).
• A **PIR motion sensor** ([http://adafru.it/189](http://adafru.it/189)).
• A **door sensor** ([http://adafru.it/375](http://adafru.it/375)).
• **Jumper wires** ([http://adafru.it/266](http://adafru.it/266)).
• Mice.

### Before You Begin

This project should be fairly easy going for beginners, but it does lean on a few basic electronics and software skills. You might want to keep these other resources in mind:

- Basic command line usage:
  - [What is this "Linux", anyhow?](https://adafru.it/sdn)
  - [What is the Command Line?](https://adafru.it/sdo)
- [GPIO Setup](https://adafru.it/sdp)
- [Adafruit IO Basics](https://adafru.it/iDX)

### Sensor Setup

You'll need:

- A **PIR motion sensor** ([http://adafru.it/189](http://adafru.it/189)) ([detailed guide](https://adafru.it/sdq))
- A **contact switch reed / switch / door sensor** ([http://adafru.it/375](http://adafru.it/375))
- A handful of **jumper wires** ([http://adafru.it/794](http://adafru.it/794))

This project uses the same hardware as [Adafruit's Raspberry Pi Lesson 12. Sensing Movement](https://adafru.it/sdr), minus a breadboard and the Pi Cobbler. On a Pi model B with the Pi Cobbler, that setup looks like this:
Instead, I did the following (which should work with a B+ or a Pi 2):
From the PIR sensor to the Pi, connect:

- Power (red wire) → 5V
- GND (black wire) → GND
- Signal (yellow wire) → GPIO 18

From the door switch to the Pi, connect one end to a GND pin and the other to GPIO 23.

Software

Before attaching sensors to the trap itself, it's a good idea to make sure everything works and logs data correctly. We'll start with a stock Raspbian install, and rely on a handful of software components:

- The occi configuration helper from Adafruit's Raspberry-Pi-Installer-Scripts (https://adafruit.it/CJr).
• tmux and mosh  
• The Adafruit IO Python Library ([https://adafruit.io/eli](https://adafruit.io/eli)).  

Log in to your Pi by way of SSH or serial cable, or get a terminal. As an optional first step, run the following command to install tmux and mosh:

```
sudo apt-get install tmux mosh
```

`tmux` is a terminal multiplexer I recommend you use to edit code and run long-running scripts inside of. It's a completely optional step, but if you're interested, you can read the Use a Terminal Multiplexer ([https://adafruit.io/jWB](https://adafruit.io/jWB)) section in our shell magic ([https://adafruit.io/sds](https://adafruit.io/sds)) guide.

Next, I used occi ([https://adafruit.io/CJr](https://adafruit.io/CJr)), our configuration helper provided with Occidentalis, to configure the Pi's hostname and wireless connection. Begin by cloning the Raspberry-Pi-Installer-Scripts Git repository and running occi's built-in installer function:

```
git clone https://github.com/adafruit/Raspberry-Pi-Installer-Scripts.git
cd Raspberry-Pi-Installer-Scripts
sudo ./occi --install
```

I edited `/boot/occidentalis.txt` to look like the following:

```bash
hostname=mousetrap
wifi_ssid=[my wireless router's network name here]
wifi_password=[my wireless password here]
```

This way, with a wifi dongle plugged in, my Pi will always connect to the wireless network at boot, and I can log in via SSH ([https://adafruit.io/jsE](https://adafruit.io/jsE)) as pi@mousetrap.local
instead of worrying what IP address it's been assigned. This is really handy when the Pi is sitting in a corner next to a mousetrap instead of on my desk near a keyboard and monitor.

io-client-python

mouse.py will rely on Adafruit's Python client for IO (https://adafruit.it/eli), which provides a clean, simple interface for talking to the service. Clone and install this like so:

```bash
git clone https://github.com/adafruit/io-client-python.git
cd io-client-python
sudo python setup.py install
```

Adafruit-MouseLogger

Finally, clone the Adafruit-MouseLogger repository (https://adafruit.it/f4P) from GitHub:

```bash
cd ~
git clone https://github.com/adafruit/Adafruit-MouseLogger.git
cd Adafruit-MouseLogger
```

Once you have mouse.py, you should be able to test your sensors. Just run like so:

```bash
sudo ./mouse.py
```

Fiddle with the sensors and see if you get readings - this might take a few seconds, but you should start seeing output like so:
Trap Setup

Here's the trap I used. It's balanced so that once the mouse goes inside after bait, it tips backwards and the door snaps shut:
And here's the trap with the door sensor's magnet attached to the back end, weights on the front of the door to balance this out, and a couple of screws to act as pivot points:
The screws seem to work well about 10mm up from the pivot point on the bottom of the trap:
I figured out a good height for the magnet by setting the end of the trap on a flat surface next to the sensor and marking spots for mounting holes with a pocket knife (the end comes off to put bait in, so it's easy to work with):
I discovered that I needed to add counterweights to the door so that the trap would stay (just barely) open. I used a couple of extra small screws and those little nuts that come with [Presta valves on bike inner tubes](https://adafru.it/f5h) for this, but just about any random small hardware should work:
It's a good idea to test the balance of the trap with bait already in the cap, because you may have to adjust weight:

In the past, I've had good luck with dry noodles and seeds, but I went with peanut butter this time around in part because it sticks to the trap and thus can be less finicky for balance.

Once I had the trap add-ons figured out, I needed some sort of platform to mount the door sensor and trap, so it'd predictably trigger the sensor when trapped. I went with a spare sheet of acrylic and a couple of these self-adhesive cable clips (https://adafruit.it/CfB) I had lying around. In a pinch, a square of cardboard or plywood and
some u-staples would probably work just as well. One benefit of the cable clips is that they snap open and closed, so it's easy to separate the trap once you catch a mouse.

(If you go with acrylic, make sure to drill pilot holes for any screws.)

Once that's taken care of, the whole thing comes together:
The magnet should always land within a few mm of the door sensor, but there's a fair amount of tolerance here - they don't have to touch for the contact to register.

You might want a mount for the PIR sensor - I put together a quick one out of Lego blocks:
For a polished build, you'd probably want to mount the Raspberry Pi and the PIR sensor to the baseplate as well, enclosing wires to avoid getting them chewed on by your test subjects, but at this point you should be able to do a test deployment of the whole system.
Logging and Displaying Data on Adafruit IO

We've got a detailed series of guides on Adafruit IO basics, so this section will limit itself to the high points.

First, you'll need an Adafruit IO account and an AIO key. At the time of this writing, IO is still in a limited beta.

Once logged in, you can find the value of your key by clicking "AIO KEY" in the upper righthand side of your IO dashboard. It'll be a long, random-looking hash string.

Add Your Key to mousetrap.cfg

Open up mousetrap.cfg in Nano (or your editor of choice), and find the key option in the io section:
You'll need to uncomment this value (remove the `#` at the beginning of the line) and replace `xxx` with your key.

Create Feeds for Each Sensor

Next, you'll need to create two feeds by clicking:

- Mouse Problems
- Trap Activity

Click on "CREATE FEED" and do something like the following for each:

Mine look like this:
Create a Dashboard and Blocks to Display Feeds

Next, you'll need to create a new dashboard by clicking "Your Dashboards" on the menu and then "CREATE DASHBOARD" on the upper righthand side.

Once that's done, add some blocks to display each feed. You can see my feed here (https://adafru.it/f5l) - I went with a line chart for each feed and a gauge for the Mouse Problems feed.

Experiment with this! You might find a different display layout more interesting or useful.
Optional SQLite Logging

Suppose you:

- Don't have an IO account yet
- ...or don't have a network near your mouse problem
- ...or would rather keep your mouse data exclusive to hardware you control
- ...or just want a version you can use to build visualizations locally?

Totally cool! Let's talk about SQLite for a minute.

SQWhat?

The first thing you need to know is that SQLite, as the name implies, is a database system that implements a lightweight version of SQL, the Structured Query Language (https://adafru.it/f5n).

As computer languages go, SQL is both an old technology and an incredibly important one. It was first defined back in the 1970s, and it's in use by, among other things, just about every big web site on the planet. The guide you're reading right now, for example, is stored in a relational database queried with SQL.

Most full-featured SQL databases - MySQL/MariaDB, PostgreSQL, etc. - are big, complicated pieces of software that run as server processes and require a healthy
dose of know-how to configure. They'll run on the Raspberry Pi, but they might be a lot of overhead for a small project like this one.

SQLite, by contrast, implements only a fraction of the features of bigger systems, but it keeps its data in a single file and the basics of use can be learned in a few minutes.

### Installation & Database Setup

```
sudo apt-get install sqlite3
```

And then, from the root of the Adafruit-MouseLogger repo, run `./setup_sqlite.sh`:

```
@mousetrap ~/Adafruit-MouseLogger $ ./setup_sqlite.sh
```

Now there should be a file called `mice.db`. You can explore its contents by running `sqlite3 mice.db`.

### The Basic Idea

In an SQLite database, data is stored in rows, in tables containing pre-defined columns. The definition of each table is called a schema. A schema is defined for a given table by using a `CREATE` statement. You can see the create statement for a given table in SQLite by running `:schema tablename`, like so:

```
The setup script defines two tables: One called `events`, and one called `event_types`. It also defines a couple of pre-existing event types. You can see these by using a `SELECT` statement:

```
SELECT * FROM event_types;
```

```
sqlite> S
```

©Adafruit Industries
The * here is a lot like a wildcard in the shell ([https://adafruit.it/sdu](https://adafruit.it/sdu)) - it means "all of the fields". If you wanted just the names, of the event types, you could instead write a query like:

```sql
SELECT name FROM event_types;
```

You don't have to put things like SELECT in caps - SQLite doesn't care - but always remember to end every statement with a semicolon!

If you were going to log a tripping of the motion sensor by hand, you'd write an INSERT like this:

```sql
INSERT INTO events (timestamp, value, event_type) VALUES (DATETIME(), 1, 1);
```

In this statement:

- **events (timestamp, value, event_type)** specifies the table and the fields you're providing values for
- **VALUES (DATETIME(), 1, 1)** provides the values
  - **DATETIME()** is a function that returns a string containing the current system date and time
  - The first **1** is our value
  - The second **1** corresponds to the **id** of the event type for motion detection

Try running this a few times in a row with some different values (as in the shell, you can press the up arrow to go back to the previous query) and doing a

```
SELECT * FROM events:
```
Log Some Mouses with Python

`mouse.py` is written so that if it sees a file called `mice.db` in the directory it's run from, it'll automatically add event rows. Make sure you've run `setup_sqlite.sh`, and then run `mouse.py` and trigger the sensors a few times:

Now, you should be able to open the database with SQLite, and read back these events with `SELECT * FROM events`:

Suppose you don't really care about the values, but do want to know the timestamp and the name of the event type, instead of just its numeric id? Try this:

```
SELECT e.timestamp, et.name FROM events e JOIN event_types et ON e.event_type = et.id;
```
SQL is a deep topic, and if you're anything like me, it might take a while for it to really click, but the basics can take you quite a ways.

For more on SQLite, check out this detailed writeup over at Python Module of the Week (https://adafru.it/f5p).

---

**Trap Placement & Next Steps**

Once you have a known-good script configuration and the sensors are tested to your liking, you'll need to figure out where to set your trap and sensor platform.

**Mouse-related Considerations**

- Mice tend to travel along baseboards, behind furniture, and generally in places of concealment.
- Mice like to repeat routes and paths - if you've seen them running along a particular wall, they'll probably do it again.
• The opening of the trap should be near or along an expected mouse path. It may seem silly, but if you imagine them traveling on a straight line, and put the trap in the path so that they’d just run right inside, you’ll probably be successful.
• Remember that mice are well-known to chew on electrical wires and components. I haven’t lost any hardware to this yet during this project, but it would in no way surprise me.

Hardware Considerations

• Power: Consider placement near an outlet, or use a decent extension cord.
• WiFi: The baseboard where I set my trap is just at the outside limit of range for the wifi dongle I used; you may want to test this before committing to a location.
• Drafts & temperature fluctuations: The PIR sensor can get a lot of false positives on drafty baseboards or other areas with a lot of temperature variation on a surface area. You will likely have to test this, and adjust its positioning until it’s stable.

Animal Welfare & Human Safety

It's worth noting that there's some debate as to whether live traps are more humane than traditional kill traps. There's an argument that mice habituated to human dwellings, when released in the wild, are unlikely to survive anyway. The Adafruit Learning System is not the place to hash out that argument, but please proceed with care when engaging in any project involving live animals.

With that disclaimer out of the way, a couple of specific notes about live traps like the one I used for this project:

• You should never leave traps set when you expect to be absent from the area for any period of time longer than a few hours.
• Traps should be emptied relatively quickly, or mice inside can suffer badly from dehydration and confinement.
• Even a seemingly unbaited trap, especially if it smells like previous mice, can quickly catch a rodent.
• Don’t be surprised if you catch more than one mouse at a time.

Finally, please remember that wild rodents can carry diseases transmissible and possibly even fatal to humans. Once traps have been used, please exercise caution in their handling, and the handling of other accessories and hardware which have been
exposed to mice. You might consider using gloves to handle traps containing rodents; you should certainly wash your hands thoroughly after touching used traps. In documenting this project, I used a mild bleach solution to wipe down the outside of the trap and the baseplate before taking photos.

Final Pythonic Mouse Thoughts