# Awesome Usage Monitor

A Capstone Project in 2017

# Project Overview

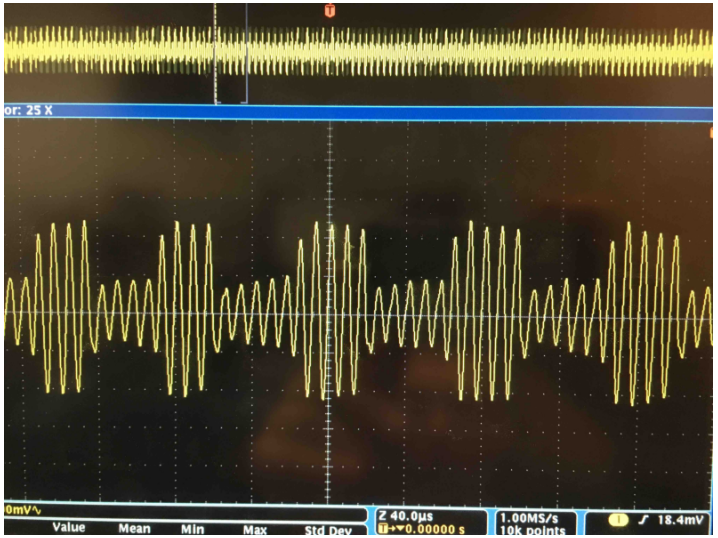
- In the EPL that are many expensive and dangerous tools.
- People could hurt themselves or damage tools.
- The Awesome Usage Monitor gives you control of who can use the tools.
- AUM also keeps logs of when used it and for how long.

# RFID

125kHz FSK signal.

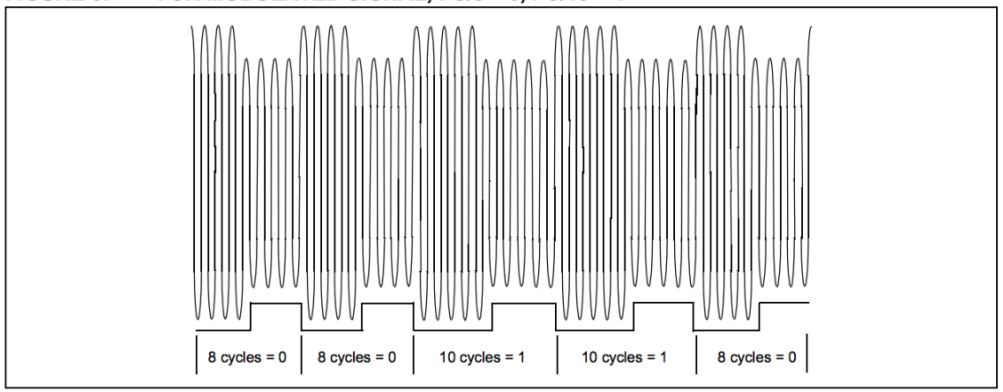
Compatible readers: HID Proxpro II, HID Proxpoint plus.. etc

Output: Wiegand protocol




PSU ID card sample data

**FIGURE 3: FSK MODULATED SIGNAL, FC/8 = 0, FC/10 = 1**

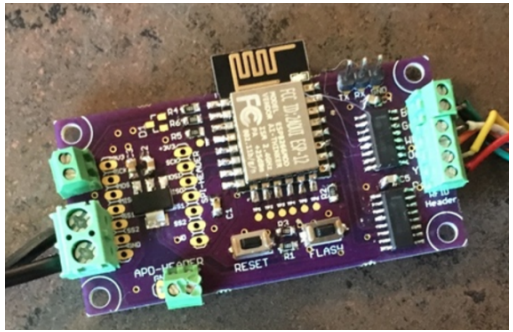


# Hardware

Based on ESP8266 12E Wifi Module:

Features:

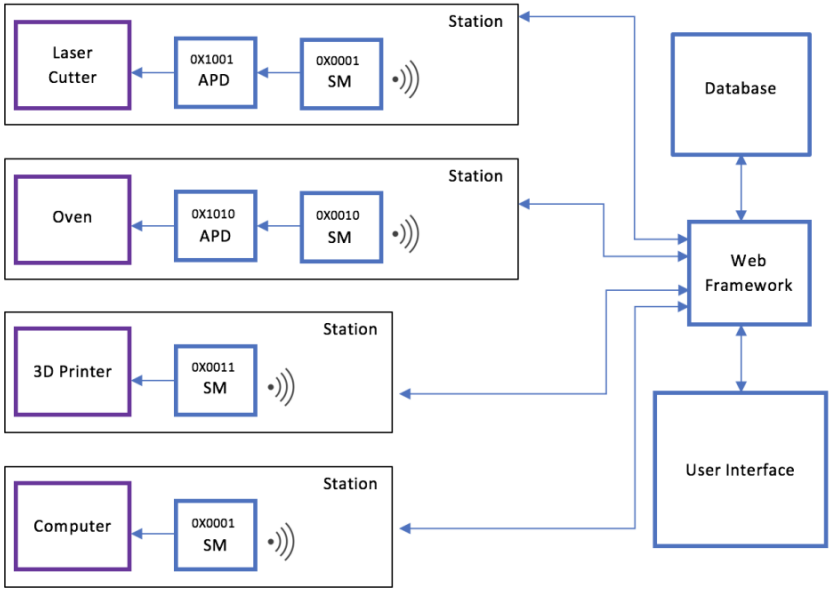
- 802.11 b/g/n
- Integrated TCP/IP protocol stack
- 32-bit MCU running at 80MHz
- 4M Flash memory
- Integrated 10-bit ADC
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO



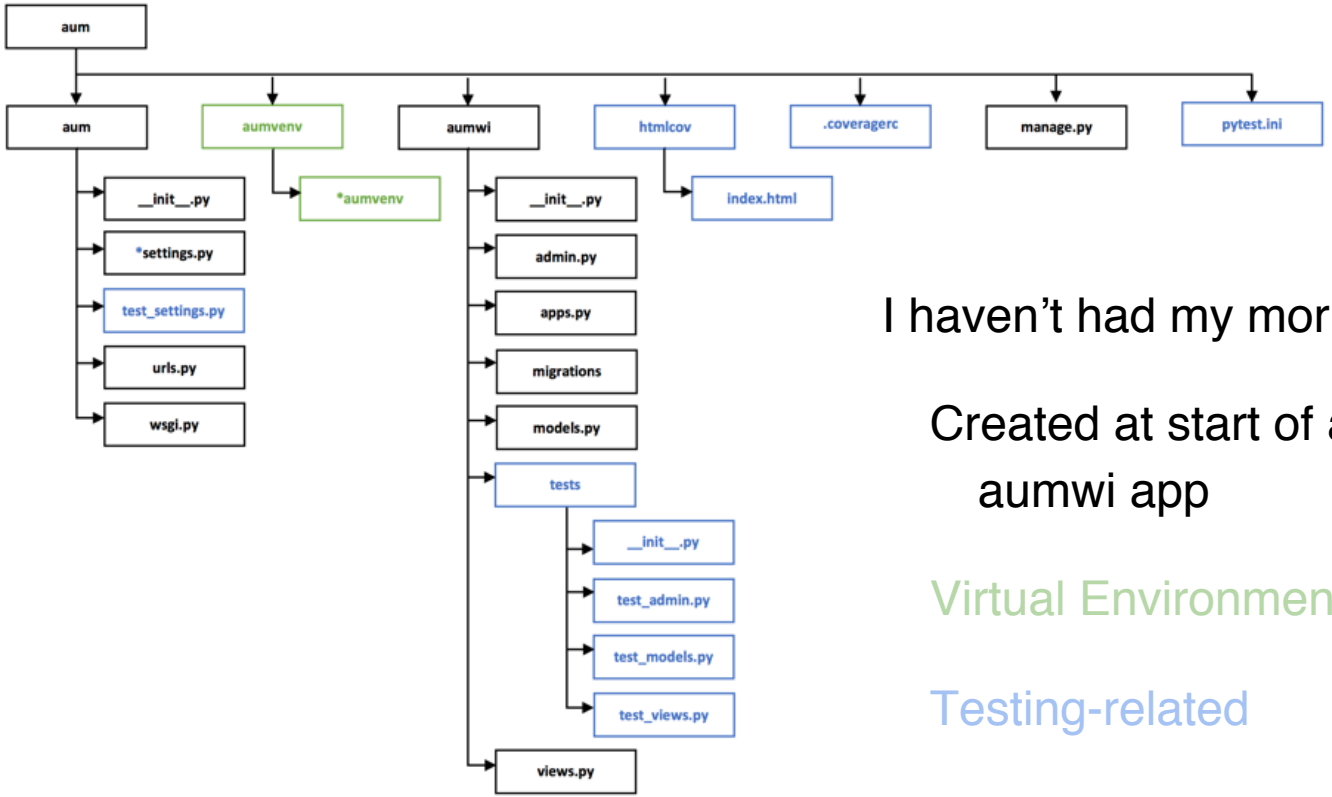


-Interfaces to isolated AC power switch
-Interfaces to HID scanner. (Wiegand protocol)

# Web Framework and Database



# Testing - General File/Folder Map




I haven't had my morning covfefe...

Created at start of aum project & aumwi app

Virtual Environment

Testing-related

# Virtual Environment (that green folder)

Do you have Python 2.6, 2.7, and 3.6 living on your computer?

Do you have different projects dependent on one of these three versions… depending on the project, of course.

Do you have one project using Django V 1.10.0 and another using V 1.11.2?

Are multiple people working on a project?

What about in the future?

**Use a virtual environment**

Overview

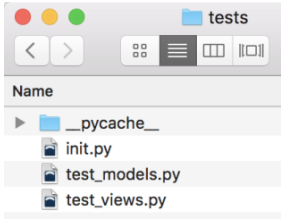
1. Install virtual environment
   $ pip install virtualenv
2. Set up a folder for your Django project (Do not call it “django”) and create a virtual environment in it.
   ***Set Python version when you create the venv***
   $ cd awesome_project_folder
   $ python3 -m venv myvenv
3. Activate the virtual environment
   $ source myvenv/bin/activate
4. Download needed items and get to work.
   (myvenv) ~$ pip install all the things!

# Testing - “All The Things”

Django: Django v 1.11.2 most recent used. --You can upgrade, but make sure tests still pass.

pytest: Testing tool with over 150+ external plugins that can run tests for Django.
py.test will:

- Find all files called “**test_.py**”
- Execute all functions called “**test_()***” on all classes that start with “**Test**”




```
class TestUser(TestCase):
    def test_init(self):
        obj = mixer.blend('aumwi.User')
        users = User.objects.all()
        self.assertEqual(len(users), 1), 'Should create a User instance.'
```

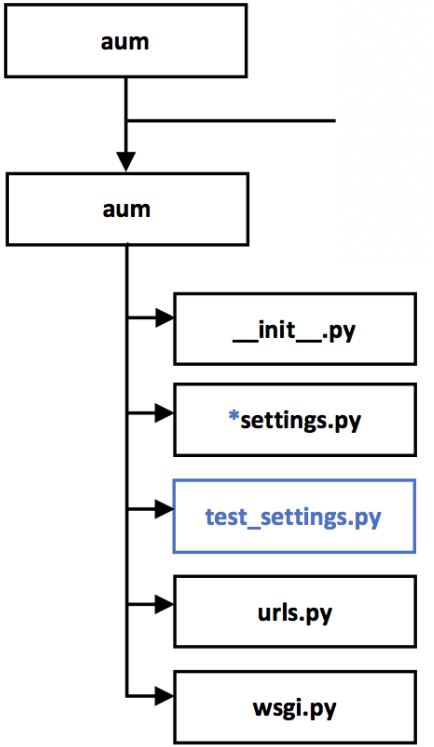
pytest-django: Less boilerplate: no need to import unittest, create a subclass with methods. Just write tests as regular functions. Use pytest marks to tell pytest-django your test needs database access.

pytest-cov: This plugin produces coverage reports.

pdbpp: Interactive source code debugger for Python programs; drop-in replacement for pdb.

mixer: Test fixture creator.

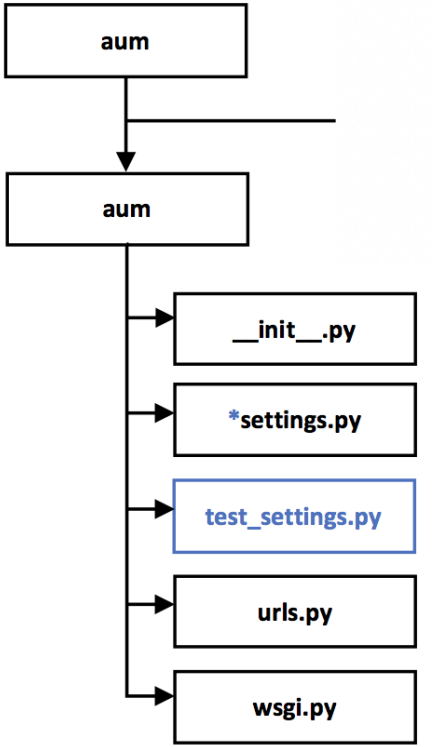
# A word of CAUTION...




aum (1st level): Root folder. Name doesn't matter. It's whatever you call it and is changeable.

aum (2nd level): Settings folder. You cannot change the name because it's in the settings. For continuity, this folder name should match between all parties working on the project in different environments.

# Settings and Test Settings




## Settings

settings.py

```
30  # Application definition
31
32  INSTALLED_APPS = [
33      'django.contrib.admin',
34      'django.contrib.auth',
35      'django.contrib.contenttypes',
36      'django.contrib.sessions',
37      'django.contrib.messages',
38      'django.contrib.staticfiles',
39      'aumwi', # <-- Add aumwi app here
40  ]
```

- Tell Django the name of your app.

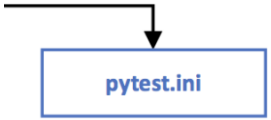
## Settings for testing

test_settings.py

```
1  #test_settings.py
2
3  from .settings import *
4
5  DATABASES = {
6      "default": {
7          "ENGINE": "django.db.backends.sqlite3",
8          "NAME": "memory",
9      }
10 }
11
12 EMAIL_BACKEND = 'django.core.mail.backends.locmem.EmailBackend'
```

- Use in-memory db. (Fast!)
- Set email to local mem. (You didn't really want to send that email, right?)

# Pytest - Test Settings and Options



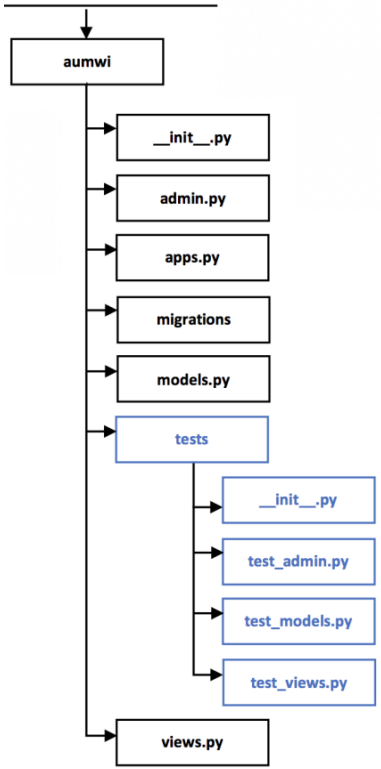

pytest.ini

```
1  [pytest]
2  DJANGO_SETTINGS_MODULE = aum.test_settings
3  addopts = --nomigrations --cov=. --cov-report=html
4
```

Tell pytest where the test settings file is.

Add options like desired coverage report format.

# Tests Folder and Files



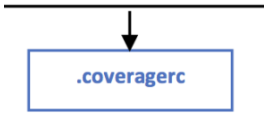

*Tests don't have to live in one file*

Get rid of the tests.py file that was automatically created by Django.

Create a folder your tests can live in

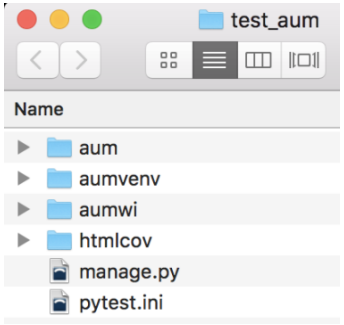
Add an __init__.py file to automatically detect tests.

# Test Coverage



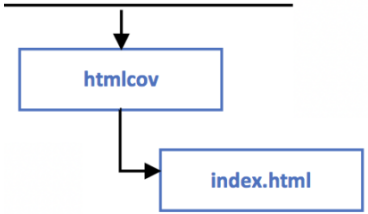

.coveragerc

```
[run]
omit =
  *apps.py,
  *migrations/*,
  *settings*,
  *tests/*,
  *urls.py,
  *wsgi.py,
  *manage.py,
  *aumvenv/*
  *aum/__init__.py,
  *aumwi/__init__.py,
```




.coveragerc: Hidden file!

Allows you to omit items you do not want to see in the coverage report.




htmlcov: Houses callable index file to show you the test coverage report.

$ open htmlcov/index.html

# Running Tests

test_aum — -bash — 98×30

```
(aumvenv) MacBook-Pro:test_aum NovaDrop$ py.test
========================================= test session starts ==========================================
platform darwin -- Python 3.6.1, pytest-3.1.2, py-1.4.34, pluggy-0.4.0
Django settings: aum.test_settings (from ini file)
rootdir: /Users/NovaDrop/test_aum, inifile: pytest.ini
plugins: django-3.1.2, cov-2.5.1
collected 9 items

aumwi/tests/test_models.py .F......
aumwi/tests/test_views.py .

---------- coverage: platform darwin, python 3.6.1-final-0 -----------
Coverage HTML written to dir htmlcov


================================================ FAILURES =================================================
_____________________________________ TestUser.test_init _____________________________________

self = <test_models.TestUser testMethod=test_init>

    def test_init(self):
        obj = mixer.blend('aumwi.User')
>       assert obj.pk == 1, 'Should create a User instance.'
E       AssertionError: Should create a User instance.
E       assert UUID('90d5141c-6640-427d-875b-cf31b682fbd8') == 1
E        +  where UUID('90d5141c-6640-427d-875b-cf31b682fbd8') = <User: mpakOUGuZzzlbfcLqNJB>.pk

aumwi/tests/test_models.py:15: AssertionError
============================== 1 failed, 8 passed in 2.70 seconds ==============================
(aumvenv) MacBook-Pro:test_aum NovaDrop$
```

test_aum — -bash — 71×30

```
(aumvenv) MacBook-Pro:test_aum NovaDrop$ py.test
======================== test session starts ========================
platform darwin -- Python 3.6.1, pytest-3.1.2, py-1.4.34, pluggy-0.4.0
Django settings: aum.test_settings (from ini file)
rootdir: /Users/NovaDrop/test_aum, inifile: pytest.ini
plugins: django-3.1.2, cov-2.5.1
collected 9 items

aumwi/tests/test_models.py ........
aumwi/tests/test_views.py .

---------- coverage: platform darwin, python 3.6.1-final-0 -----------
Coverage HTML written to dir htmlcov


====================== 9 passed in 2.64 seconds ======================
(aumvenv) MacBook-Pro:test_aum NovaDrop$
```

# Running Coverage Reports

(myvenv) ~$ open htmlcov/index.html