Outletify: A Project for CSCI 3308

Who

- Alan Moy
- Jonathan Wehrend
- Austin Glaser
- Joseph Vostrejs

Title: Outletify

Description

Outletify provides two key features: energy usage tracking, and home automation. Two distinct hardware components will make up a single Outletify system. Microcontrollers integrate with outlets to allow for low level control and usage detection, and a central server will be the main point of control for all microcontrollers. Energy usage tracking features will monitor and log the times outlets are in use, and the server will collect data and present it to the user with straightforward graphics and statistics over a web interface. This same web application will also serve as a hub for all home automation capabilities, allowing the user to control and automate household activities and appliances. By scheduling regular tasks and logging historical usage, Outletify will grant users finer control over their house - what's running, and what isn't.

Vision statement

Outletify is the simplest way to manage and monitor all the household technology you depend on. Lower your bills, reduce your environmental footprint, and make your life easier by leaving menial tasks to the machines.

Automated Tests

Because of the wildly different environments the different sections of our project run under, we require two unique testing setups.

Server

Running tests

To facilitate easy testing, we've created a top-level script to run all the server's tests. Called `run_tests`, it will invoke the various test suites necessary to test the server. Note that it may require some module installation.

Test Output

```
Running uWSGI server tests
.....
----------------------------------------------------------------------
Ran 5 tests in 7.115s

OK
Running Django tests
/usr/local/lib/python2.7/dist-packages/django/db/models/fields/__init__.py:1474:
RuntimeWarning: DateTimeField Usage.time_stamp received a naive datetime
(2015-11-11 20:20:00) while time zone support is active.
  RuntimeWarning)

.
----------------------------------------------------------------------
Ran 1 test in 0.004s

OK
Creating test database for alias 'default'...
Destroying test database for alias 'default'...
```

Sensor

The sensor uses a custom test framework. Embedded system development has some unique and not entirely common requirements, including:

- Small code footprint
- No dynamic memory allocation
- Portability to non-standard printing methods (no printf)

- Implemented entirely in C (no C++)

We evaluated several candidate frameworks (uUnit, embUnit, ACEUnit), and found them all to not entirely meet our requirements.

Our source code can be found in `sensor/system/test/microunit`. One interesting feature is that we use `setjmp()` and `longjmp()` to provide psuedo-exception test failure handling, which allows test failures to occur in nested functions (commonly in C test frameworks, test assertions are macros which simply return from their current function).

Running tests

The tests must necessarily run on the microcontroller's embedded hardware. Currently, we are compiling them in with our main application, where they run each time the system is reset or reprogrammed to provide instant feedback. They can be disabled with a makefile variable (`TEST`) to provide an easy method for generating a releasable binary.

Some tests are for hardware drivers. There is no way of verifying the intended effects of these drivers entirely within the software of the microcontroller. Professional firms will sometimes use an external device which can monitor system-level signals; however, these are quite expensive. We compromise by providing some simple prompts for the test runner, allowing us to log success/failure of external test elements.

Test Output

```
SUITE: switch_test_suite
 - TEST: starts_off                          PASS
 - Is the power output active? [y/n]: y
 - TEST: turn_on                             PASS
 - Is the power output active? [y/n]: n
 - TEST: turn_off                            PASS
 - TEST: toggle                              PASS
PASS
SUITE: pool_test_suite
 - TEST: allocate                            PASS
 - TEST: allocate_i                          PASS
 - TEST: can_free                            PASS
 - TEST: can_free_i                          PASS
 - TEST: allocate_all                        PASS
 - TEST: no_overlap                          PASS
PASS
```

User Acceptance Tests

Outletify — **UAT Test Plan Template**

Description	**Test if website can be accessed**
Testing Start Date	**List the Start Date of UAT**
Testing End Date	**List the End Date of UAT**
Test Case ID	**Site_01**
Module Name	**Web API**
Name of Tester(s):	
Pre-conditions	**Login Username & Password**
Dependancies	**web api**

Step	Test Step	Test Data	Expected result	Actual Result	Pass/Fail	Sign-off	Comments
1	Navigate to Login Page	Useremail@gmail.com	User should be able to Login	User is navigated to dashboard when logged in	Pass		
2	Provide Valid Username/Password	Password: abc123					
3	Click on Login Button						
4							
5							
6							
7							
Post Condition							

Outletify — **UAT Test Plan Template**

Description	**Retrieve Data**
Testing Start Date	**List the Start Date of UAT**
Testing End Date	**List the End Date of UAT**
Test Case ID	**Data_10**
Module Name	**Database**
Name of Tester(s):	
Pre-conditions	**Logged into web API**
Dependancies	**web api**

Step	Test Step	Test Data	Expected result	Actual Result	Pass/Fail	Sign-off	Comments
1	Navigate to Login Page	Useremail@gmail.com					
2	Provide Valid Username/Password	Password: abc123					
3	Click on Login Button						
4	Select Range of Dates to display data for	11/10/15 - 11/11/15	See all KW/H Data collected between the two timestamps	Data Displayed	Pass		
5	Click on Display Button						
6							
7							
Post Condition							

Outletify — **UAT Test Plan Template**

Description	**Test if visualization module is working**
Testing Start Date	**List the Start Date of UAT**
Testing End Date	**List the End Date of UAT**
Test Case ID	**Visualize_06**
Module Name	**Visualization**
Name of Tester(s):	
Pre-conditions	**Database working, able to Login**
Dependancies	

Step	Test Step	Test Data	Expected result	Actual Result	Pass/Fail	Sign-off	Comments
1	Navigate to Login Page	Useremail@gmail.com					
2	Provide Valid Username/Password	Password: abc123					
3	Click on Login Button						
4	Select Range of Dates to display data for	11/10/15 - 11/11/15					
5	Click on Visualize Button		View a graph of the collected data over time.	Raw data displayed	Fail		Visualization not fully implemented.
6							
7							
Post Condition							

VCS

Github

- https://github.com/austinglaser/csci3308-project