*Etherlux*®: A Novel Internet Controlled Smart Power Outlet System

Rounok Joardar

Copyright © 2016 Rounok Joardar
All rights reserved
• A novel internet connected power outlet that can be controlled over the worldwide web has been developed

• The system is centered around a Raspberry Pi and simple digital circuitry

• The main distinguishing features of this design are:
  • ability to control outlets well beyond simple on/off operations (e.g. timed on/off, duty cycle control, complementary behavior, etc. etc.)
  • continuously measure and log line voltage, current, and power
  • manual on / off override on each outlet
  • wireless (wifi) and wired network connectivity
  • low cost compared to similar units available today
  • easy upgrades to increase scale
  • robust performance

• The following slides describe in detail the internal workings of the system

Basic Concept Behind *Etherlux*

**Web Page**
- Point to Raspberry Pi's URL
- Click on button to send php request via internet to web server running on Raspberry Pi

**Raspberry Pi**
- Running Apache webserver
- Php request executes a local Python code
- Python code generates a specific digital waveform each time it is run
- Each button press on web page will run the python code one time

**State Machine**
- Logic circuit processes input bitstream and outputs a series of high or low signals

- [Diagram showing Vdd (5V), Relay, GPIO, and Gnd connections]
Simple Implementation of Concept

Raspberry Pi
- Python code (pulse.py) generates a single pulse each time it is run. Each button press on web page will run the python code one time.

Texas Instruments SN74HC109 +ve Edge Triggered J-K Flip-Flop

Hasco KLT1C6DC5 5V Relay

Vishay 4N33 Optoisolator

Clock signal (from Pi)
Flip flop output at Q (positive edge trig'd)
Etherlux in Action

- Etherlux outlet control web page on iPhone

This section shows ON/OFF and power status of each outlet

Each outlet can be toggled by clicking on its power button icon

3 preset programs are available to choose from

- Pgm 01: Turn off outlet 01, turn on outlets 02 and 03
- Pgm 02: Turn off all outlets
- Pgm 03: Pulse outlet 01 ten times at 0.2 Hz
Implementation Details

- Designed for 3 controlled outlets and one "always on" outlet
- GPIO port numbers 4, 17, and 27 on Raspberry Pi are used
- Each port is connected to clock pin of a JK flip flop
- J and K inputs of flip flops are tied to high (creating a toggle flip flop)
- Pull-down resistors and decoupling capacitors are used at GPIO ports to stabilize signals
Implementation Details (State Sensing)

- State of each outlet is determined by:
  - tapping the voltage at midpoint of the 300 ohm resistor at the input of the optoisolator associated with the outlet
  - feeding this voltage to a GPIO port configured as an input port (use 4.7k resistor to protect input port)
- If GPIO port reads digital high, then the corresponding outlet is considered ON
Implementation Details (Manual Override)

- Momentary ON (normally OFF) push button connected as shown between 5V supply and a GPIO input terminal

- Background process is run that constantly listens for rising edge at a specific GPIO input
  - if a rising edge is detected AND the input state remains high after 0.5 sec, a pulse is generated at the appropriate GPIO output terminal
  - this toggles the associated outlet

Python code (pulse.py) generates a single pulse each time it is run. Each button press on web page will run the python code one time.

Web Page

Raspberry Pi
- Python code (pulse.py) generates a single pulse each time it is run. Each button press on web page will run the python code one time

Texas Instruments SN74HC109 +ve Edge Triggered J-K Flip-Flop

Push switch (momentary ON)

To optocoupler and relay
Implementation Details (Metrology)

Web Page

Raspberry Pi
- Python code (pulse.py) generates a single pulse each time it is run. Each button press on web page will run the python code one time

Microcontroller

- Line voltage and load current are measured and recorded using the ADC inputs of a microcontroller; microcontroller output fed to Pi for final processing and display

Interface electronics (JK f/f, optoisolator, etc.) and relay

Current transformer

Voltage transformer

120V mains

Vdd (5V)

Gnd

GPIO

Amplifiers and level shifters
Implementation Details (Metrology)

- Oscilloscope traces of line voltage (blue) and load current (yellow)
Coding Details

- Apache webserver installed and running on Raspberry Pi
- This is the simple html code that is present in the /var/www directory on the Raspberry Pi (file = lights.html)
- Sends request to the webserver to run php file "runPython.php" with the value "port04" or "port17" or "port27" depending on which outlet the user chooses to toggle

```html
<html>
<body>
<form action="runPython.php" method="post">
<input type="checkbox" name="lightNum[]" value="port04">Light01
<br>
<input type="checkbox" name="lightNum[]" value="port17">Light02
<br>
<input type="checkbox" name="lightNum[]" value="port27">Light03
<br>
<br>
<input type=submit value="Continue">
</form>
</body>
</html>
```
This is the php code that is present in the /var/www directory on the Raspberry Pi (file = runPython.php)

Upon receiving request from the internet the webserver runs this file

The php code in turn executes a Python script named "pulseGen.py" with the value "port04" or "port17" or "port27" depending on which outlet the user chose to toggle.
import RPi.GPIO as GPIO
from time import sleep
import sys

GPIO.setmode( GPIO.BOARD )
GPIO.setup( 4 , GPIO.OUT )
GPIO.setup( 17 , GPIO.OUT )
GPIO.setup( 27 , GPIO.OUT )
GPIO.output( 4, False)
GPIO.output( 17, False)
GPIO.output( 27, False)

port = sys.argv[1]
pulseWidth = 5
portNum = 4
if port == "port04":
    portNum = 17
if port == "port17":
    portNum = 27
if port == "port27":
    portNum = 7
print( 'Active port = ' + str(portNum) )

if (portNum == 4 or portNum == 17 or portNum == 27):
    i = 1
else:
    i = 4
    #print( str(i) )
while i < 2:
    print( 'Generating pulse on port ' + str(portNum) )
    GPIO.output( portNum , True )
sleep ( pulseWidth )
    GPIO.output( portNum, False)
sleep ( pulseWidth )
i += 1
    print( 'Done' )
GPIO.cleanup()

- This is the Python code that is present in the /var/www directory on the Raspberry Pi (file = pulseGen.py)
- When the php code executes this Python script, the GPIO output at "port04" or "port17" or "port27" is toggled, depending on which outlet the user chose
Final Assembly

This is the final assembled view of the "Etherlux" web controlled power outlet box.

Raspberry Pi with wi-fi adapter and other electronics and relays are located inside the enclosure.

Internal view is shown in next slide.
Final Assembly (contd..)

• Internal view of the "Etherlux" web controlled power outlet box
### Parts List & Cost

<table>
<thead>
<tr>
<th>Item</th>
<th>Qty</th>
<th>Cost ea.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi Zero</td>
<td>1</td>
<td>$5.00</td>
<td>$5.00</td>
</tr>
<tr>
<td>Relay (KLT1C6D5)</td>
<td>3</td>
<td>$0.95</td>
<td>$2.85</td>
</tr>
<tr>
<td>Optoisolator (4N33)</td>
<td>3</td>
<td>$0.45</td>
<td>$1.35</td>
</tr>
<tr>
<td>Dual JK Flip Flop (SN74C109)</td>
<td>2</td>
<td>$0.53</td>
<td>$1.06</td>
</tr>
<tr>
<td>Resistors, capacitors, diodes, wires</td>
<td>x</td>
<td>Varying</td>
<td>$3.00</td>
</tr>
<tr>
<td>Breadboard</td>
<td>1</td>
<td>$4.00</td>
<td>$4.00</td>
</tr>
<tr>
<td>Case</td>
<td>1</td>
<td>$10.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>MicroSD card (8 Gb)</td>
<td>1</td>
<td>$4.00</td>
<td>$4.00</td>
</tr>
<tr>
<td>WiFi adapter</td>
<td>1</td>
<td>$6.00</td>
<td>$6.00</td>
</tr>
<tr>
<td>Outlet receptacles</td>
<td>3</td>
<td>$1.25</td>
<td>$3.75</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>$41.01</strong></td>
</tr>
</tbody>
</table>

- Similar commercial grade item costs over $100 (e.g. webpowerswitch.com)
Several “lessons learned” along the way:

• Cross-talk arising from switching large loads is a problem (e.g. state of outlet 1 changes unexpectedly when outlet 2 is toggled)
  • Isolate relay power supply from Pi and digital Vdd
  • Use solid state relays with zero-crossing detection

• Must use pull-down resistors at GPIO outputs
  • Drift in GPIO output pins result in false clock signal at JK flip flop