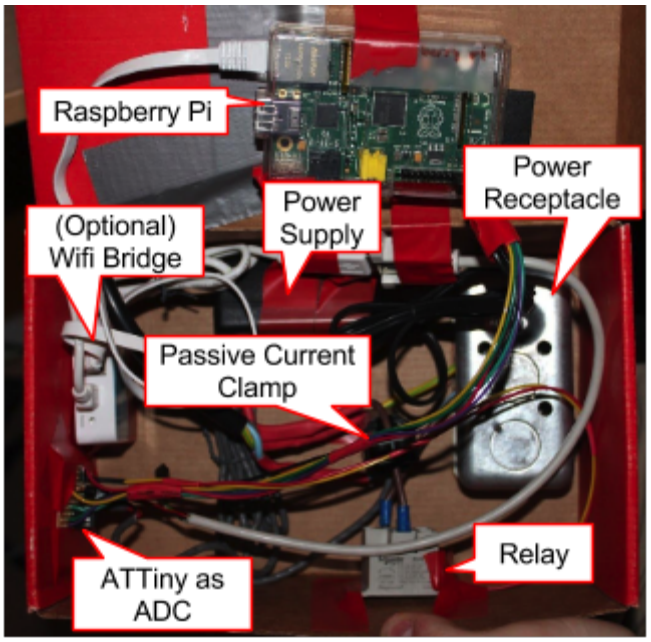
WifiWatt

Carnegie Mellon University
Electrical & Computer ENGINEERING
BUILD18

WifiWatt implements a network of current sensing and controlling nodes that be used to easily retrofit existing electrical systems. Our proof-of-concept system is built almost entirely of off the self parts. The system uses a central server for data logging, communication to the nodes, and serving a real-time web page along with several nodes for current data and control. Each node plugs into a regular wall outlet and provides 2 measured and switched receptacles.

Our project won Build18 2013's *Outstanding Project* award.



The hardware in a single node.

Measuring Current

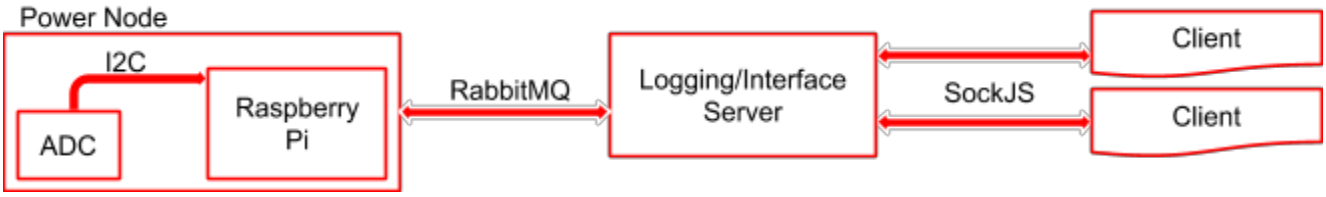
We chose to measure current non-invasively with a current clamp. We then read the small induced current over a known resistance. An ATtiny microcontroller serves as an Analog to Digital Converter (ADC), and we perform the necessary RMS calculation on the Raspberry Pi with it's Floating Point Unit. This configuration enables the collection of current data without worrying about dangerous voltage spikes on the power line damaging our other low-voltage electronics. Like the power control circuitry, all the DC measurement circuitry is physically and electrically isolated from the AC circuit.

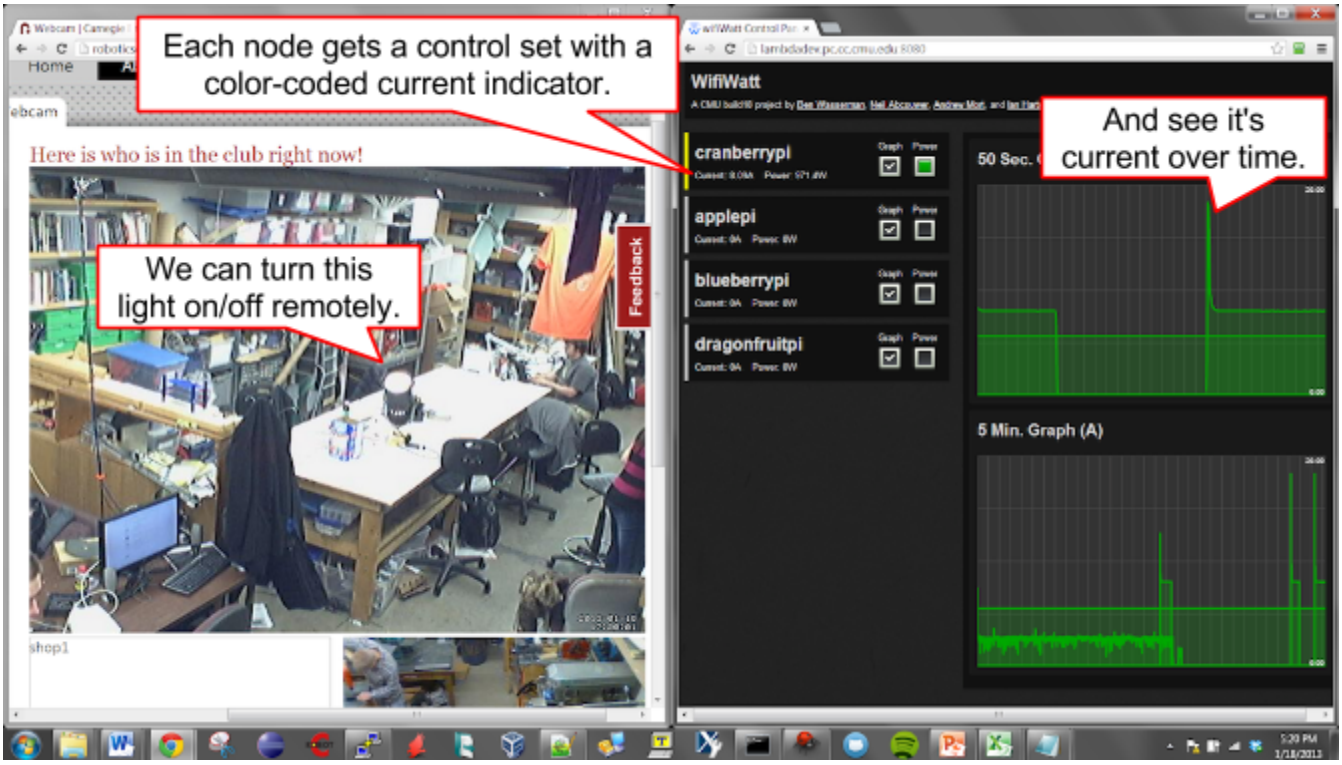
Controlling Power

We used a mechanical relay with a simple driver circuit to open and close the 120V AC circuit. Using a mechanical rather than solid state relay maintains the physical and electrical separation between high-voltage AC and low-voltage DC circuitry. Since finding a relay with low enough current and voltage demands to be satisfied by the Raspberry Pi's 3.3V General Purpose IO (GPIO) pins is demanding, we built an additional transistor circuit capable of driving a standard 5V relay.

Communications

Since this project brought together many different processing and sensing units, intercommunication was essential. The ADC to Raspberry Pi connection uses a simple I2C (TWI) serial protocol. Each node's Raspberry Pi connects over a standard ethernet or wifi connection to the RabbitMQ (http://www.rabbitmq.com/) messaging server. The messaging server handles communications between the nodes and the logging/interface server. Client web pages can then connect to the interface server in real time with websockets or xhr long polling. The protocol decision is managed by the SockJS library. These interactions are visualized below.





Interface

The WifiWatt web interface (seen above right) uses real-time communications with the logging/interface server as described above. Embedded javascript handles sending the server user events - like requesting a graph of current data or turning off a node - and updating the graphs with new data received on-demand from the server. All buttons and links on the page are standard html elements with styling in order to degrade gracefully. Real-time canvas-based graphs are generated with the Smoothie Charts (http://smoothiecharts.org/) script.

Future

Although our proof-of-concept system runs the messaging, logging, and interface servers on the same machine, the system can easily be expanded due to the RabbitMQ messaging foundation. New node data can be read off the queue by multiple logging servers, for example. Rabbit ensures that this new data will only be processed once. Commands can also be sent from different interface servers to a node; as long as the interface server has the node's address, rabbit will handle delivery. Our choice of messaging technologies means our system can scale from managing a single room to CMU's entire campus effortlessly.

Additionally, we chose to use off-the-shelf hardware due to our time constraints for this project. However, each node could easily be consolidated on a single board. This could easily cut the cost per node in half. Since we built 4 nodes with our $250 budget (approx. $60 each), this represents a significant savings to an already low cost.

About Build18

Build18 is an Electrical and Computer Engineering (ECE) focused hackathon that takes place the first week of Spring semester at Carnegie Mellon University. Project groups are formed and parts are ordered before we leave for winter break. Each project receives a $250 budget thanks to the program sponsors. All project work is completed within the week preceding Demo Day on Friday. Find out more at http://build18.ece.cmu.edu/.

Ben Wasserman
bwasserm@
andrew.cmu.edu

Ian Hartwig
ihartwig@
andrew.cmu.edu

Andrew Mort
amort@
andrew.cmu.edu

Neil Abcouwer
nabcouwe@
andrew.cmu.edu