2018 ASEE Mid-Atlantic Section Spring Conference: Washington, District of Columbia Apr 6

The Design and Implementation of a Smart Switch Outlet Adapter

Dr. Sasan Haghani, University of the District of Columbia

Sasan Haghani, Ph.D., is an Associate Professor of Electrical and Computer Engineering at the University of the District of Columbia. His research interests include the application of wireless sensor networks in biomedical and environmental domains and performance analysis of communication systems over fading channels.

Mr. Mohammad Ali Rahimi, Rutgers, The State University of New Jersey

Student at Rutgers University studying for his Master's degree in Electrical and Computer Engineering. Co-founder of Arcane Reality, a Virtual Reality gaming company.

The design and Implementation of a Smart Switch Outlet Adapter

Ali Rahimi[1] and Sasan Haghani[1,2]

[1] *Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, 08854*

[2] *Department of Electrical and Computer Engineering, University of the District of Columbia, Washington DC 20008*

Abstract

This paper focuses on the development of a Wi-Fi smart plug capable of remotely switching a load on or off. It monitors energy consumption and collects data that is sent to a web application server where the user may interface with the smart plug. A relay is implemented in single phase. The user can communicate with the plug to switch the relay on or off.

Keywords

Smart Plug, energy monitoring, on-off control

I. Introduction and Background

There is an increasing interest in energy efficiency and focus on reducing the cost of embedded network sensors to decrease the overall cost of outlet-level metering. Organizations use IEEE802.15.4 to effectively deliver solutions for a variety of areas including consumer electronic device control, energy management, and home and commercial building automation. Reducing the electricity usage in buildings can make a substantial impact on greenhouse gas emissions. By focusing on energy consumption, smart plugs can manage and improve power efficiency [1]-[2], [4], and [6]. The United States is the top smart home market on the globe with installations in 9.7% of all households recorded in 2015 [9]. To further assist in the integration of smart energy saving devices, plug load metering provides medium potential energy saving at a reasonably low cost [10]. Most plug load meters available today sit at the wall outlet and sell for around $50 with wireless connectivity or $20-$30 without wireless connectivity [1]-[2]. This paper presents the outcomes of a course project on the design of a smart plug that incorporates measuring energy consumption as well as switch enabling. Unlike the design in [5] which uses an ESP-WROOM02 Microcontroller, this project utilizes an Arduino compatible device. This project utilizes a consumer energy monitoring product called Kill-A-Watt as the main energy metering component with a guaranteed error of 0.2%. This varies from the custom design built in [5] with an error less than 0.5%. The rest of this paper is organized as follows. In section II, an overview of the system design is provided describing the necessary hardware components and the overall hardware design, and the code that interacts with the embedded components in calculating voltage, current, and power values. In Section III, communication with the server and data analysis are provided. Finally, in Section IV, conclusions and future work are presented.

II. System Design

II.A High level System Overview

The smart plug is designed to be connected to an access point or Wi-Fi network. Data is communicated to a server and from the server communicated to the user. The user can interact with the server's web app via their portable device shown on Fig 1. The Kill-A-Watt measures AC power, while the Teensy pulls the data values from Kill-A-Watt's LM2902 operational amplifier. The Teensy reads data from the Kill-A-Watt, calculates a standard deviation based off the power usage, and sends that data to the server. The server provides time based schedule load switching following user requests by returning a value to the microcontroller to switch the relay on or off.

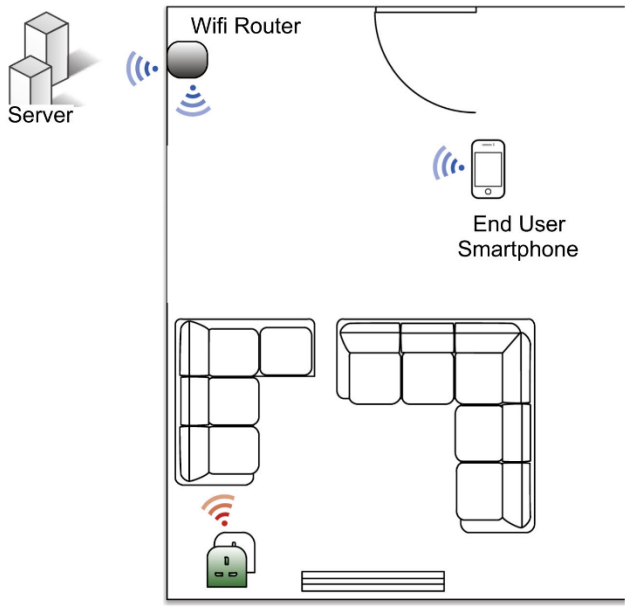

Figure 1: Communication System Design

II.B Hardware Design

There are four main components integrated together in designing the system: A low power Teensy Microcontroller, an ESP8266 Wi-Fi module, a 5V relay, and the Kill-A-Watt Energy monitoring device. Fig. 2 provides the schematic of the design and how the components are connected together.

Teensy Microcontroller

This paper employed a teensy 3.1. Its small size 1.4 by 0.7 inch and being inexpensive made it suitable for this project. It provides SPI, UART, I2C, PWM and generic GPIO. It can utilize the Arduino IDE.

ESP8266 Wi-Fi module

The ESP8266[8] is a self-contained SOC with integrated TCP/IP protocol stack that can provide any microcontroller Wi-Fi access. It has 1 MiB of built in flash memory. The ESP8266 allows the plug to communicate with multiple users through varying personal devices as shown in Fig.1.

5V Relay

The relay used is a single pole – double throw (SPDT) relay. They are used to switch high voltage and/or high current devices. The relay contains a relay driver used to control the relay. When the driver receives a 5V active high value, it signals the relay to switch. The relay itself contains 5 pins: Coil End 1, Coil End 2, Common (COM), Normally Close (NC), and Normally Open (NO). Coil End 1 and Coil End 2 are used to trigger the relay. Normally one end is connected to 5V while the other is connected to ground. The common pin is connected to one end of the load that is being controlled. Normally Closed or Normally Open are where the other end of the load will be placed. If placed on NC, the load remains connected before trigger. If placed on NO, the load remains disconnected before trigger.

Kill-A-Watt

The Kill-A-Watt is an energy monitoring system that can measure load voltage, Current, Wattage, and frequency at 0.2% accuracy. The Kill-A-Watt does not have any smart features such as the ability to send the data received. The purpose of this paper is to explore a solution where the data can be sent to a server and recorded and measured.

Hacking Kill-A-Watt

Utilizing the KAW, data is received from Kill-A-Watt's LM2902 quad op-amp, at the bottom of the KAW display board. Three pins from the LM2902 are tapped into containing the voltage signal, high current range signal, and low current range signal. These inputs are received and attenuated to run at 3.3V for the Teensy. A simple voltage divider solves this issue. A 60 Hz square wave is connected to Pin 1, the interrupt pin on the Teensy from the collector pin from a BJT on the KAW. All shown in Fig 2.

From Fig. 2, it is shown that the Teensy and ESP8266 use Serial Communication. The Tx pin of the Teensy is connected to the Rx pin of the ESP8266. The Tx pin of the ESP8266 is connected to the Rx pin of the Teensy. The relay contains 3 pins (VCC, IN2, GND). IN2 is connected to Pin 3 on the Teensy. The hot line of the power cable is connected to the common and NO (normally open) input on the relay. It is shown that the Kill-A-Watt has a male and female end connecting one side to the wall outlet and the other for the load device. As the relay turns off, the system too turns off.

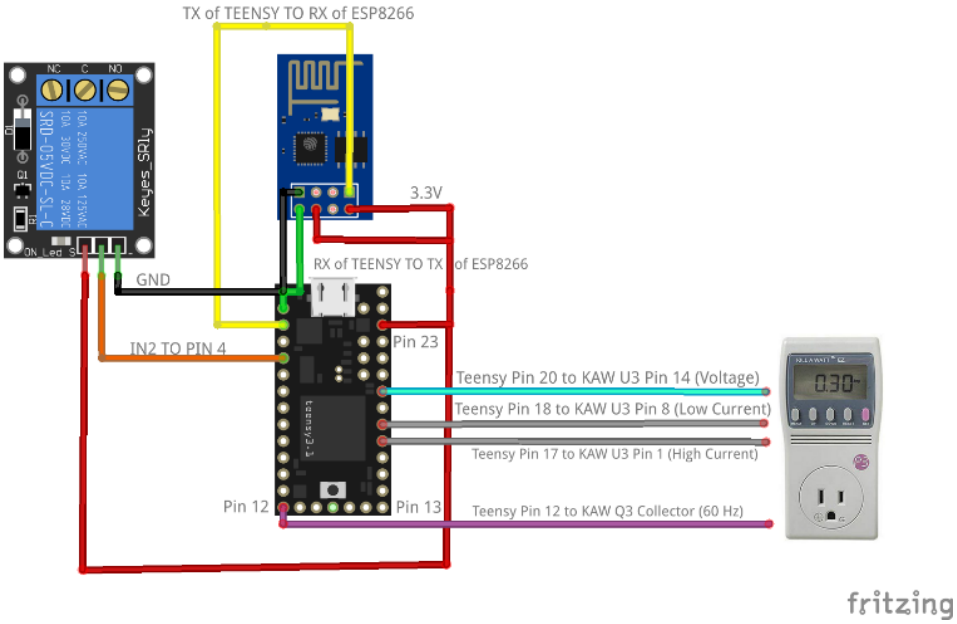

Figure 2: Component Integration

II.C Software design

The software design portion consists of Part A which focuses on the microcontroller receives data outputted by the pins of the Kill-A-Watts LM2902 operational amplifier. Using a built in Arduino function 'analogRead', an array is created for the high and low current, and voltage values. In part B, an average of these array values are taken and the RMS value is calculated for power, current, and voltage.

A. Value Measurement

Teensy runs a toggle that is activated on the Rising and falling edge from Q3, the collector on the Kill-A-Watt. Every time an interrupt is ran MeasState is incremented by one. Once MeasState has the correct parameters, the values are recorded. The code for this process is shown in Fig. 3.

B. *Average Value Calculation*

As stated in the introduction of this section, an array is created for the high and low current and voltage values. The average of each value is calculated by taking the sum of each element contained in the array for the high and low current and voltage. The sum is then divided by the number of elements to give the average. By calculating the standard deviation, we are also calculating the RMS. Within the second for-loop displayed in Fig. 4, the sum is taken again containing the difference between each element of the sampled value and the average of that value squared pertaining to the high and low current and voltage. The average of that sum is taken and then square rooted to give the root mean square. The root mean square of each signal is now calculated giving the final value for

V[], A[], and a[] which represent the voltage, high current, and low current value. The power W[], and w[] are calculated where W[] represents the power of a device where the current is above 2A, while w[] represents the power of a device where the current is below 2A.

```
int n = 0;
measState = 0; // 0 is partial half cycle
toggle = !toggle;
if (toggle) {
  attachInterrupt(HzINT, AC60HZ, RISING);
} else {
  attachInterrupt(HzINT, AC60HZ, FALLING);
}
while (measState < 2); // skip partial cycle
//digitalWrite(LED,HIGH);
detachInterrupt(HzINT);
while ((measState < 3) && (n < MaxN)) { // measure 1 cycle
  //digitalWrite(LED,HIGH);
  A[n] = analogRead(ASENS); // 20.4 us
  V[n] = analogRead(VSENS); // 20.4 us
  a[n] = analogRead(aSENS); // 20.4 us
  n++;
  //digitalWrite(LED,LOW);
  // time difference between voltage and current readings is about 20.4 us
  // 20.4 us is about 0.44 degrees at 60 Hz, good enough!
  // target 50 readings over 1 cycle @ 60Hz
  // reading time: 3 * 20.4 us * 50 = 3.06 ms
  // total dead time: (1/60) - 3.06 ms = 13.61 ms
  // dead time between readings: 13.61 ms / 50 = 272 us
  delayMicroseconds(280); // fine tuned for 50 readings @ 60 Hz
}
```

Figure 3: Arduino value read

```
KAW.N = n; // number of readings
// compute offset levels
float VSum = 0, ASum = 0, aSum = 0, WSum = 0, wSum = 0;
for (int i = 0; i < n; i++) {
  VSum += V[i];
  ASum += A[i];
  aSum += a[i];
}
float aV, aA, aa; // averages
aV = VSum / n;
aA = ASum / n;
aa = aSum / n;
// compute std.dev. = AC RMS
VSum = ASum = aSum = 0;
for (int i = 0; i < n; i++) {
  VSum += (V[i] - aV) * (V[i] - aV);
  ASum += (A[i] - aA) * (A[i] - aA);
  aSum += (a[i] - aa) * (a[i] - aa);
  WSum += (V[i] - aV) * (A[i] - aA);
  wSum += (V[i] - aV) * (a[i] - aa);
}
KAW.V = sqrt(VSum / n);
KAW.A = sqrt(ASum / n);
KAW.a = sqrt(aSum / n);
Serial.println("RAW a");
Serial.println(KAW.a);
KAW.W = WSum / n;
KAW.w = wSum / n;
```

Figure 4: Calculating RMS values

IV. Server Communication and Data Analysis

A. ThingSpeak Communication

The data collected from the Teensy is sent to ThingSpeak [4] via the ESP8266. ThingSpeak is an open IOT platform with MATLAB analytics. Using TCP protocol to start the connection, AT+CIPSTART command is invoked to connect to the ThingSpeak server. An HTTP Post request is sent to ThingSpeak with the power and voltage values shown on Table 1 of a Hakko Fx888d 70W Soldering Iron[11]. A HTTP GET request is made sending a string "On" or "Off" which is received and parsed through. When either String is received, Pin 3 on Teensy is HIGH or LOW. Because the connection from the hot wire is connected to Normally Open, grounding the pin actually turns the relay on.

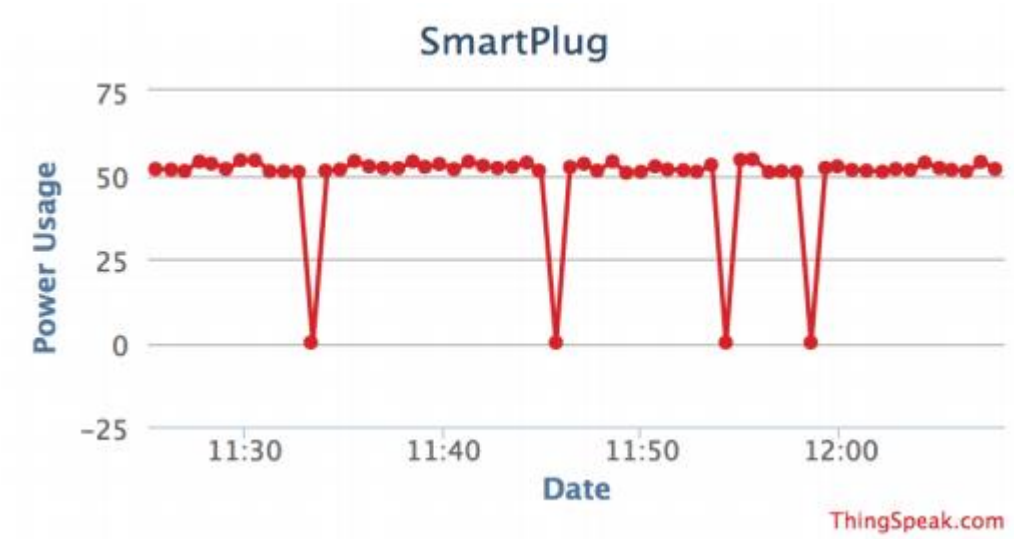

Table 1: ThingSpeak Values recorded

B. ThingSpeak Analytic functions

ThingSpeak provides a function called TalkBack. It enables any device to act upon queued commands. A string containing characters "On" or "Off" are received by the microcontroller commanding the plug to turn on or off. Fig. 5 provides an example of when an "On" string command is sent via HTTP GET request to the Teensy. ThingSpeak also provides a function called TimeControl to perform TalkBack function at a specific time or on a regular schedule. Fig. 6 depicts a TimeControl event sending an "Off" string.

Commands

Position	Command ID	Command string
1	10147559	On

Figure 5: ThingSpeak Values recorded

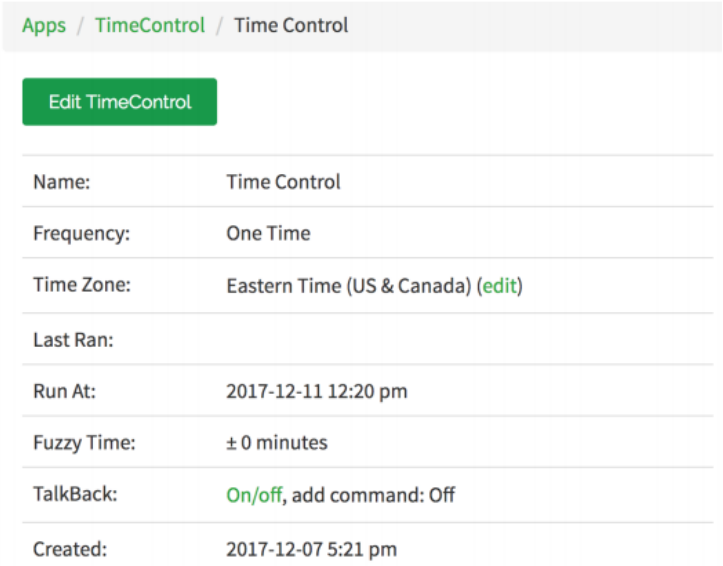

Figure 6: ThingSpeak Time Control

V. Conclusions and Future Work

In this paper, we have detailed the design and characterization of a smart plug capable of remote switching a load on or off. This plug monitors energy consumption and collects data sent to an open IOT platform called ThingSpeak, where a user can interact with the device through HTTP GET requests sending a string that is received and parsed by the Teensy microcontroller. This design demonstrates a hackable method utilizing very few components to make a Kill-A-Watt into a smart IOT plug. However, the Kill-A-Watt is not the best sensor to use to track power because the values received by the Teensy need to be mapped and calibrated to match the value of power consumed. This is due to not knowing the power factor of any load device. The future revision of this product will have a custom PCB board. It will have its own dedicated server that handles wireless communication. Added features will include a weekly summary of your power consumption for not just 1 smart plug but multiple.

VI. References

1. “Things,” SmartThings. [Online]. Available: https://shop.smartthings.com/things. [Accessed: 04-Mar-2018].

2. Electricity Monitors & Energy Savers | P3. [Online]. Available: http://www.p3international.com/products/energy-savers.html. [Accessed: 04-Mar-2018].

3. “Understand Your Things,” IoT Analytics - ThingSpeak Internet of Things. [Online]. Available: https://thingspeak.com/. [Accessed: 04-Mar-2018].

4. A. Zipperer *et al.*, "Electric Energy Management in the Smart Home: Perspectives on Enabling Technologies and Consumer Behavior," in *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2397-2408, Nov. 2013.

5. Y. Thongkhao and W. Pora, "A low-cost Wi-Fi smart plug with on-off and Energy Metering functions," *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Chiang Mai, 2016, pp. 1-5.

6. M. by T. Dhar, “ESP8266 IoT Energy Monitor,” Hackster.io, 13-Sep-2017. [Online]. Available: https://www.hackster.io/whatnick/esp8266-iot-energy-monitor-b199ed. [Accessed: 04-Mar-2018].

7. M. Tranchemontagne, “Hooked on Arduino & Raspberry Pi,” Moteino Kill-A-Watt: Hardware, 01-Jan-1970. [Online]. Available: http://www.mikesmicromania.com/2013/04/moteino-kill-watt-hardware.html. [Accessed: 04-Mar-2018].

8. “ESP8266,” ESP8266 - NURDspace. [Online]. Available: https://nurdspace.nl/ESP8266. [Accessed: 05-Mar-2018].

9. “Smart Homes and Home Automation 5th Edition,” – Giiresearch. [Online]. Available: https://www.giiresearch.com/report/ber203895-smart-homes-home-automation.html. [Accessed: 05-Mar-2018].

10. Neep, “The Smart Energy Home: Strategies to Transform the Region,” *http://neep.org*, Oct-2016. [Online]. Available: http://neep.org/sites/default/files/resources/SmartEnergyHomeStrategiesReport_3.pdf. [Accessed: 05-Mar-2018].

11. “HAKKO | Soldering iron | HAKKO FX-888D,” *Global Sites*. [Online]. Available: http://www.hakko.com/english/products/hakko_fx888d_set.html. [Accessed: 05-Mar-2018].