Accepted at The 5th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC 2013).

Inspiring Energy Conservation Through Open Source Metering Hardware and Embedded Real-Time Load Disaggregation

Stephen Makonin*†, William Sung‡, Ryan Dela Cruz‡, Brett Yarrow‡, Bob Gill‡, Fred Popowich*, Ivan V. Bajić§
*Computing Science and §Engineering Science, Simon Fraser University
†Technology Centre and ‡School of Energy, British Columbia Institute of Technology
Email: {smakonin, popowich, ibajic}@sfu.ca, {Stephen_Makonin, Bob_Gill}@bcit.ca

***Abstract*—Utility companies around the world are replacing electro-mechanical power meters with new smart meters. These digital power meters have enhanced communication capabilities, but they are not actually smart. We present the cognitive power meter (c-meter), a meter that is actually smart. By using load disaggregation intelligence, c-meter is the realization of demand response and other smart grid energy conservation initiatives. Our c-meter is made of two key components: a prototype open source ammeter and an optimized embedded load disaggregation algorithm (μDisagg). Additionally, we provide an open source multi-circuit ammeter array that can build probabilistic appliance (or load) consumption models that are used by the c-meter. μDisagg is the first load disaggregation algorithm to be implemented on an inexpensive low-power embedded processor that runs in real-time using a typical/basic smart meter measurement (current, in A). μDisagg can disaggregate loads with complex power states with a high degree of accuracy.**

***Index Terms*—embedded software, energy conservation, load modelling, open source hardware, real-time systems**

I. Introduction

Currently, much of the world is focused on reducing energy consumption for both economical and ecological sustainability. One way for homeowners and occupants to reduce energy consumption is to understand how they consume energy and adjust their habits accordingly. As utility companies replace old electro-mechanical power meters with new smart meters there is hope that this can be achieved if the right information is presented to the occupants in a timely and convenient manner through an eco-feedback device or display mechanism. Currently, the display mechanisms promoted by some utility companies only report aggregate whole-house power and energy values. Occupants are not given any feedback as to how their appliances consume energy. Furthermore, with initiatives such as time-of-day usage charges (peak charges) and demand response (DR) [1] homeowners are left with little to no information to work from.

As we noted in a previous paper [2], it is doubtful that an occupant knows which appliance(s) need to be turned off to meet an opt-in DR request. With most advanced home automation systems this can be achieved, but at a high cost which is a barrier for the majority of home owners. One might think that a smart meter might do this. However, the smart meter has no *smarts* to help the occupant understand what appliances are running, it is just a digital meter with advanced communication capabilities.

By adding *smarts* to a smart meter we can help occupants participate in the opt-in DR requests. One way to help is to have intelligence that can discern what appliances are running from examining the whole-house power reading. This is called *load disaggregation* – first developed by Sultanem [3] and then Hart [4]. A smart meter with load disaggregation is what we call a *cognitive power meter* (c-meter, see Figure 1). In the next sections we will discuss the two key components of the c-meter: the Precision Ammeter hardware (see Section III) and the μDisagg load disaggregation firmware (see Section V). As well, we will discuss our Precision Ammeter Array hardware (Section IV) that can build up to 60 probabilistic appliance (or load) consumption models that are used by the c-meter. μDisagg is a new embedded load disaggregation algorithm that does not build from existing load disaggregation algorithms published by other researchers. Furthermore, it has been implemented on an inexpensive low-power embedded processor running in real-time using the basic current (I, in A) measurement – unlike any other load disaggregation algorithm to date. We close with a discussion on interfacing the c-meter to the smart grid to allow for participation in opt-in demand response (see Section VII).

II. Background

With a c-meter, an in-home display (IHD) can display a list of appliances or combination of appliances that the occupant can choose from to meet an opt-in DR request in a timely and confident fashion. *So why is there no c-meter?* Load disaggregation in its current form is not feasible for the typical home, although some initial attempts in devices such as the TED-5000 have allowed use in restricted contexts[1]. There is an optimization problem–having the algorithm fit on an embedded processor. Some systems, depending on the machine learning technique, can have a computational cost of $O(n^m)$, where n is the number of states and m is the number of appliances/loads. This results in an optimization problem when wanting to

[1]See the "I want to view more than my overall usage. How can I monitor an individual appliance?" FAQ question at http://www.theenergydetective.com/faq (last accessed January 26, 2013).

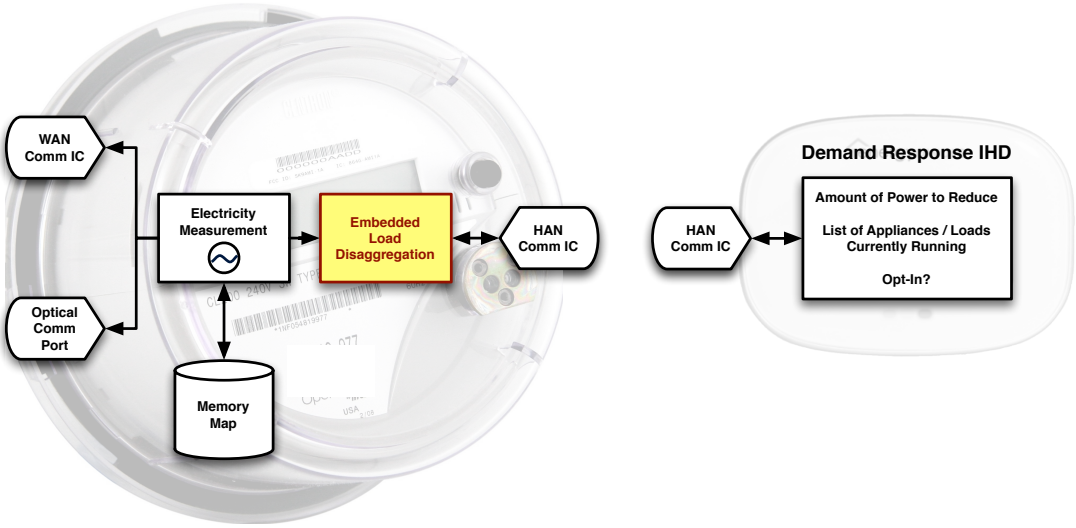

Fig. 1. The physical components in our Demand Response Model: (left) the cognitive power meter or c-meter and (right) the demand response in-home display. Note that the Embedded Load Disaggregation module only sends data to devices on the HAN side due to privacy concerns.

implement load disaggregation on an embedded processor. We can make some assumptions to mitigate the problem of exponential computational cost. Using discrete measurements, disaggregation is bound linearly by the maximum current draw of the house (in Amperes) and of each load. We can also bind the number of loads to disaggregate to only those loads that are deferrable (e.g. clothes dryer) and give occupants the best opportunity to conserve energy. We feel that the number of desired loads to disaggregate would be 10 – Ziefman [5] has considered between 10 to 20 appliances although in his work he only ever evaluates 9 appliances.

There are many privacy concerns that involve load disaggregation which centre around utility companies being able to tell what appliances a homeowner is using, and having the utility company turn off appliances without a homeowner's consent. Our opinion is that the intelligent load disaggregation part of the c-meter needs to exist on the Home Area Network (HAN) side of the meter as we noted in Figure 1. If the load disaggregation module only communicates with devices on the HAN then privacy concerns should be alleviated [2].

III. The Precision Ammeter

Many researchers have focused on using smart meter data for their load disaggregation algorithms [5]–[11]. In particular they use real power measurements (in W). However, after we performed a study on detailed and long term information available in our AMPds dataset [12] we found that current (I) when compared to real power (P) was a better measurement to use.

Table I shows the result of an analysis we performed on 473,232 data points (per min readings) over 11 months. We found that real power (W) readings had a high degree of fluctuation (as high as $59\times$, see Table I) compared to current (A). This is due in part to the meter using two sensor readings (current and voltage) that can both fluctuate independently to measure real power. Formulae for apparent and real power are shown in (1)

TABLE I
Current (in A) *vs* Real Power (in W) Comparison

ID	Load	Distinct I	Distinct P	Flux
BME	Basement Plugs & Lights	8	387	48×
CDE	Clothes Dryer	20	632	32×
CWE	Clothes Washer	14	720	51×
DWE	Dishwasher	8	270	34×
FGE	Kitchen Fridge	17	525	31×
FRE	HVAC/Furnace	7	298	43×
HPE	Heat Pump	33	1268	38×
TVE	Ent TV/PVR/AMP	7	415	59×
WOE	Wall Oven	21	646	30×

$$\begin{aligned} S &= I{\cdot}V \ , \\ P &= S{\cdot}\cos(\Theta) = I{\cdot}V{\cdot}\cos(\Theta) \ , \end{aligned} \tag{1}$$

where S is apparent power, P is real power, and Θ is the angle between voltage (V) and current (I). Compounding this problem is the fact that branch circuit power meters (BCPM) measure current at each breaker (using current transformers or CTs) but measure power from one spot on the breaker power panel.

A. Hardware Design

There are a number of open source hardware and software development platforms available to build prototype systems. We chose the Arduino (*http://www.arduino.cc*) platform because of its popularity, support ecosystem, and commitment to open source. From what we observed in the previous section we designed an open source Precision Ammeter (see Figure 3) that would measure the whole-house current used by μDisagg.

We studied projects such as Open Energy Monitor (*see http://openenergymonitor.org/emon/*) that sample both voltage and current waveforms using integral equations. These methods proved to be processing intensive for the microprocessors

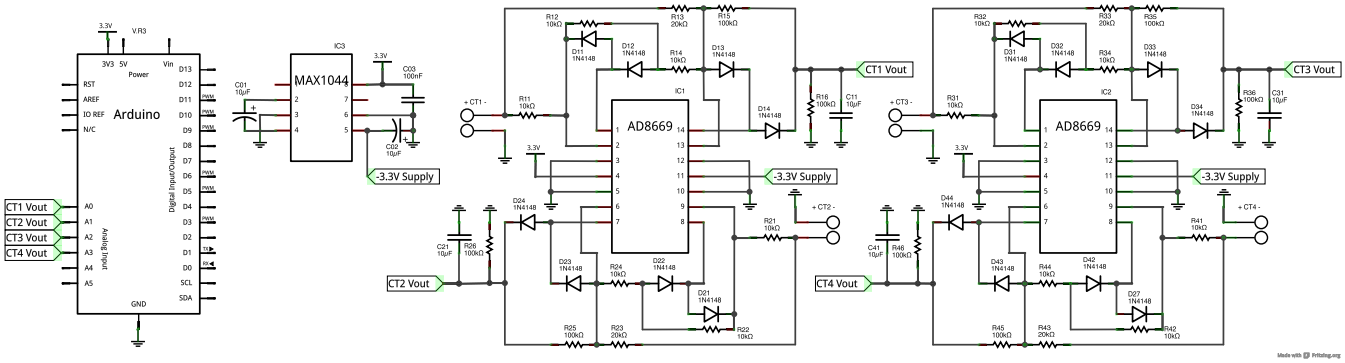

Fig. 2. The schematic of our open source Precision Ammeter for use with the Arduino Due which can read four CTs.

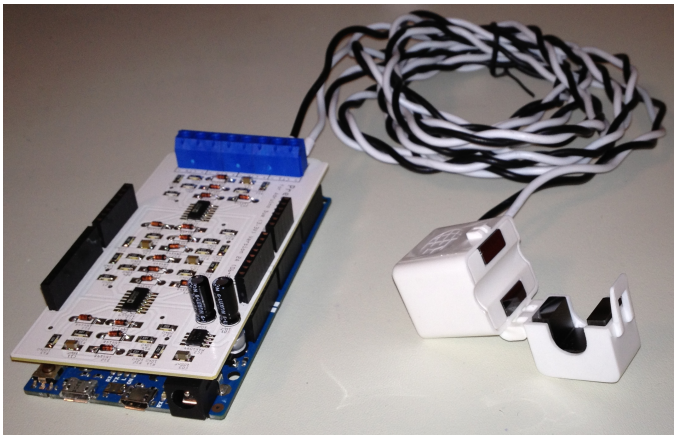

Fig. 3. The open source Precision Ammeter using the Arduino Due and split-core CT. A maximum of 4 CTs can be connected.

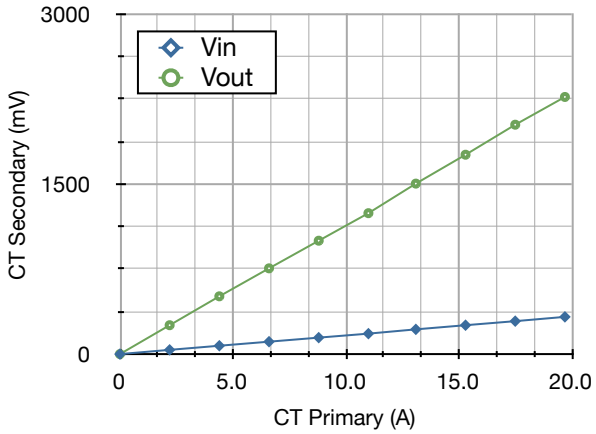

Fig. 4. Op-Amp circuit linearity test results. Using a 20A CT we tested the full CT secondary range (0–333mV). *Vin* is the voltage input the op-amp circuit (also the CT secondary voltage) and *Vout* is the voltage output of the op-amp circuit that is used by the ADC. Loads were created using 300W indecent light bulbs.

and thus we decided to design circuitry that would accomplish the same task, but without having the microprocessor extensively sample the waveforms (see Figure 2 for the schematic and Figure 3 for the final product picture).

With our choice of having a current transformer to sample the currents of a specific load, we obtained a voltage reading that would be directly fed into our *analog to digital converter* (ADC) on the Arduino Due board. From the output of the current transformer we obtained a voltage reading that would range from 0–333mV. This value is sinusoidal and thus we rectified this signal to obtain instantaneous values of current being fed through to the load.

The specifications of the Arduino Due board noted that input ports are limited to a maximum input voltage of 3.3V to ensure that no damage occurs to the board. The Arduino Due board allows its users to work with power supplies of 3.3V. Thus to power up our operational amplifiers (op-amps) we chose to use the 3.3V power supply along with the MAX1044 charge pump to create the -3.3V power supply. This allowed our op-amps to output a maximum of 3.3V and a minimum of -3.3V. As op-amps saturated before reaching the minimum or maximum output voltages, the output would never actually reach 3.3V and the microprocessor's port maximum input voltage would never be exceeded. We use a gain of 5 (100kΩ ÷ 20kΩ).

We performed a linearity test (see Figure 4) to verify that the op-amp circuit we designed did not saturate when rectifying an amplifying the CT secondary range (0–333mV). The CTs we use (primary 200A, 100A, 50A, and 20A) all have the same secondary output range (0–333mV). The 20A CT was used for testing due to limitations and safety. We found that due to different tolerances with resistors the we had an actual gain of 6.9 not 5.

B. Firmware Design

The firmware of our ammeter was designed to be simple and accurate when measuring current. To provide accurate uniform sampling of each CT, we used a timing interrupt running at 1kHz. We used sliding window averaging (the last 1000 samples) to smooth out any jittering and sudden spikes in current readings. An I^2C interface was also provided to allow other equipment to receive measurement readings.

IV. The Precision Ammeter Array

The Precision Ammeter Array (a-array) is a multi-circuit ammeter that consists of a number of *element boards* or expansion boards that allow the a-array to expand the number

of CTs being metered and the Data Logger. The a-array allows us to monitor a large number of breakers to build load models that μDisagg can use for load disaggregation. The cost of the CTs alone make the a-array an unlikely candidate for a permanent multi-point monitoring solution; hence, the continued need for a small ammeter and load disaggregation (the c-meter).

A. Hardware Design

Using the same op-amp circuitry (Figure 2) we extended the design to create the element board. We chose to use the Teensy 3.0 (*http://www.pjrc.com/store/teensy3.html*) open source board rather than the Arduino Due, due to its small footprint. Each element board can monitor up to 12 CTs and communicate with the Data Logger via a SPI serial interface. The Data Logger can have a maximum of 5 element boards connected to it. This means a total of 60 CTs can be metered. The limitation in the number of element boards exists mainly due to the fact that there needs to be a separate *chip select line* per element board.

B. The Data Logger

The Data Logger is based on the Arduino Due and uses the SD Card Shield for logging data. The interface cable that is used to carry power, ground and the SPI lines was soldered to the prototyping area of the SD Card Shield. Raw Ampere readings, as well as load model data, is stored on an SD card at a maximum rate of 1Hz. After a period of between 2 weeks to 1 month of monitoring, the load models are downloaded and converted (using a Python script) into a C header file format and then the data is compiled and uploaded to the c-meter. This is currently a manual process and there are plans to have this automated using a wireless protocol such as ZigBee or WiFi (802.11). These load profiles are the *probability mass functions* discussed in the next section.

V. μDisagg: Load Disaggregation

Modern home appliances (even LED lighting) now have embedded electronics that allow for different modes of operation creating complex behaviours and not the *simple on/off* behaviour[2] researchers like to disaggregate [5]. The real challenge for load disaggregation algorithms is the need to detect these complex, finite-state appliances and loads. Our load disaggregation algorithm μDisagg can handle these types of loads. Our philosophy is that a load has loads at any level– be it an appliance, room, home, or neighbourhood. For each load we build a prior knowledge model (a probability mass function) at any level and then disaggregate it. The next two subsections discuss our load disaggregation algorithm [12].

A. Single-Measurement Disaggregation

Let there be l independent discrete random variables $X_1, X_2, \ldots, X_l$, corresponding to current draws from l loads.

[2]Hart [4] identified four basic types: simple on/off, finite-state, constantly on, and continuously variable.

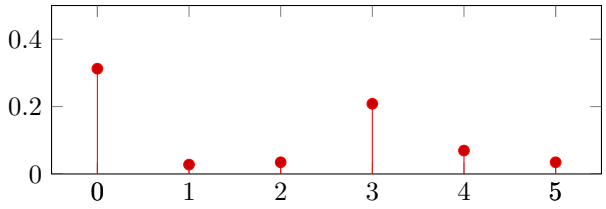

Fig. 5. A stem diagram of the example PMF.

Each X_i is the deci-Ampere (dA) measurement of a metered electric load with a *probability mass function* (PMF) of $p_{X_i}(x)$, where i is the load index $i \in \{1, 2, ..., l\}$, x is a number from a discrete set of possible measurements $x \in \{0, 1, ..., m_i\}$, and m_i is the upper bound imposed by the breaker that the i-th load is connected to. For example, with dA measurements on a 15A breaker, we would have $m_i = 150$. The PMF $p_{X_i}(x)$ is defined as follows:

$$p_{X_i}(x) = \begin{cases} \Pr[X_i = x], & x \in \{0, 1, \ldots, m_i\}, \\ 0, & \text{otherwise}, \end{cases} \quad (2)$$

where $\Pr[X_i = x]$ is the probability that the current draw of the i-th load is x. For example, if the PMF of X_i is $(0.10, 0.05, 0.25, 0.40, 0.20, 0, 0, ...)$ for $x \in \{0, 1, 2, 3, 4, 5, 6, ...\}$, then $\Pr[X_i = 2] = 0.25$, so the probability of the i-th load drawing 2 dA (i.e., 0.2 A) is 0.25.

The probability $\Pr[X_i = x]$ is estimated from measurements over a sample period. For example, if over T measurements, the current draw x was recorded j times, then $\Pr[X_i = x] = \frac{j}{T}$. During the sample period each load is metered at a consistent rate of one measurement per minute. This rate determines the time resolution at which the load disaggregation will be performed.

TABLE II
An Example PMF

x	0	1	2	3	4	5
j	900	80	100	600	200	100
$p_{X_i}(x)$	0.3125	0.0278	0.0347	0.2083	0.0694	0.0347
s	0	1				

Peaks in PMF $p_{X_i}(x)$ are designated as probable load states $s \in \{0, 1, \ldots, S_i\}$, where $S_i + 1$ is the number of states. The states are assigned by quantizing the range of possible measurements $[0, m_i]$ (without gaps) so that each quantization bin contains one peak of the PMF. For example, in Figure 5 and Table II, there are two peaks in the PMF, so we would model this PMF by a two-state model, with states being indexed $\{0, 1\}$. The probability of each state is the total probability mass within its quantization bin.

B. MAP disaggregation

A single measurement of the whole-house current draw is given by

$$Z = X_1 + X_2 + ... + X_l, \quad (3)$$

where Z is the sum of current draws by all loads. We want to be able to determine the values of X_i's from the value of Z. In general, there are multiple combinations of X_i's that would produce any given Z, but not all of them have the same probability. We want to find the combination $(X_1, X_2, ..., X_l)$ that is the most probable, given the sum Z, i.e., the one that maximizes the posterior probability $\Pr(X_1, X_2, ..., X_l|Z)$. Note that the conditional probability $\Pr(Z|X_1, X_2, ..., X_l) = 1$ if $\sum_{i=1}^{l} X_i = Z$, and 0 otherwise, so by Bayes' rule we have

$$\begin{aligned} &\Pr(X_1, X_2, ..., X_l|Z) \\ &\quad = \frac{\Pr(Z|X_1, X_2, ..., X_l)\Pr(X_1, X_2, ..., X_l)}{\Pr(Z)} \\ &\quad = \begin{cases} \frac{\Pr(X_1, X_2, ..., X_l)}{\Pr(Z)} & \text{if } \sum_{i=1}^{l} X_i = Z, \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{4}$$

Since $\Pr(Z)$ is common to all combinations, it does not make a difference to their rank ordering in terms of probability. Hence, the MAP solution is the one with the highest probability $\Pr(X_1, X_2, ..., X_l)$ such that $\sum_{i=1}^{l} X_i = Z$. Since the load current draws are assumed independent, we have $\Pr(X_1, X_2, ..., X_l) = \prod_{i=1}^{l} \Pr(X_i)$.

VI. Experimental Setup & Results

We took our load disaggregation algorithm [12] and made extensive changes so that it can run on an embedded processor. First, we changed the whole-house measurements from deci-Amperes (dA) to whole Amperes (A). We did this to reduce the amount of distinct current readings which in turn: (1) reduced the amount of processing time, and (2) reduced the amount of measurement inaccuracies when using different meters. Different meters have different sensors and different measurement circuits. When comparing one meter with another (even if other meters are highly accurate) the current readings from the metered loads can differ by 50mA to 200mA. Secondly, we rewrote our initial Python algorithm in C++ which allowed us to simplify and optimize the code for running on an embedded processor.

Working with embedded processors always presents a challenge both in processing power and in space (memory and storage). To this end, we needed to limit how we implemented MAP. Through data analysis we found that the *ground truth* answer nearly always is found in the three most probable current levels when disaggregating each load. Thus, we modified MAP to only store the three most probable current levels for each load. Then we ran MAP disaggregation to determine the most like combination of loads running.

We chose 9 diverse loads (or sub-meters) to disaggregate from our AMPds dataset [12]. Seven sub-meters containing a single load (CDE, CWE, DWE, FGE, FRE, HPE, WOE), and 2 sub-meters containing multiple loads (BME, TVE). All loads can be considered *finite-state* loads. FRE (HVAC/Furnace) is mainly a *constantly on* load consisting of a fan and thermostat. CWE (Clothes Washer) is a front load washer with a variable speed spinning drum so it is a *finite-state* load combined with a *continuously variable* load. Our initial Python algorithm [12] was used as a baseline.

A. The Results

Table I shows testing results using our Baseline and μDisagg algorithms, their run times and accuracies. The disaggregation of 524,544 data points went from an average run time of 24 minutes for the baseline algorithm, down to an average of 43 seconds for μDisagg). Nine experiments (one for each sub-meter) were run to see if single loads could be identified (or disaggregated) from the whole house reading. Accuracy is based on the correctness of both: (1) the load being correctly identified, and (2) Ampere amount is the same as ground truth. We conducted our experiments using a Python implementation of the algorithm on a MacBook Air with a 1.8 GHz Core i7 CPU and 4 GB of memory.

TABLE III
Load Disaggregation Results

ID	Baseline		μDisagg		Accuracy
	Time	Accuracy	Time	Accuracy	Improvement
BME	23 min	77.0%	39 s	87.0%	+10.0%
CDE	16 min	97.9%	39 s	98.9%	+1.0%
CWE	33 min	97.4%	46 s	97.7%	+0.3%
DWE	19 min	97.3%	41 s	97.3%	0.0%
FGE	32 min	55.0%	48 s	64.5%	+9.5%
FRE	21 min	33.8%	39 s	86.5%	+52.7%
HPE	30 min	84.7%	52 s	94.1%	+9.4%
TVE	20 min	57.0%	39 s	90.4%	+33.4%
WOE	18 min	99.5%	45 s	99.2%	-0.3%

A second experiment was performed to test the accuracy of μDisagg to predict all 9 loads at the exact current level for each time period. An accuracy 44.6% was achieved when all 524,544 data points. If we relax our accuracy measure by $\pm$1A (an approximation) so long as there is not a transition OFF to ON from a load, or visa versa then we get an accuracy of 53.0%. Having exact accuracy for current levels may only be important when there are changes in state (e.g. an load that is ON turning OFF). This condition is important because μDisagg might indicated that a load was ON even thought it was not or μDisagg might determine a load was OFF even though it was ON – in both cases this would be counted as an in accurate result. When examining the disaggregation results the majority of the errors come from FGE (the kitchen fridge) which cycles frequently enough between OFF and ON (cooling) where there is no dominant state like with the other loads selected. If we remove the disaggregation of the fridge from our tests then the accuracy improves to 79.7%.

After implementing μDisagg on the Arduino Due, we found it took an average of 3.78ms to disaggregate 9 appliances on the Ardunio Due (an ARM Cortex-M3 processor). This allowed us to run μDisagg the Precision Ammeter instead of our initial design plans that had two Arduino boards (one for the ammeter and the other for disaggregation) communicating via I^2C. Note that it took about 2 minutes to upload the house

model data (used for disaggregation) to the Arduino Due, but this only needed to be done once.

B. Further Analysis

It may be counterintuitive to think that we can have better accuracy with coarser measurement values, but in some cases this approach can indeed result in improved accuracy, specifically in cases involving jitter removal. For example, the more fine-grained dA current readings for FRE (the constantly-on furnace fan) shows a lot of flux because the fan has a variable speed motor. As the filter gets clogged with dust over time, the motor has to work harder to push the same volume of air (this gets reset when the air filter is changed). When FRE is measured in dA there can be a spread of high probability measurements from 1A–1.6A and 1.8A–2.3A all being of the state *fan on*. If FRE current is measured in A instead of dA, then the spread becomes two values (1A–2A) with an increased probability.

However, using A increments instead of dA increments does not allow us to identify vampire loads or appliances in standby mode. For example, the CDE (Clothes Dryer) has a standby mode of 400mA which is now seen as being 0A. So there are trade-offs, but those can be good or bad depending on our disaggregation goals. For instance, if we are only interested in identifying large deferrable loads for demand response then identifying vampire loads may not be the priority.

VII. Smart Grid Interface

The c-meter will play a pivotal role in the DR interaction between utility company and house occupants–not just for the initial DR request.Load disaggregation also will play a part in the verification of the accepted DR request [11]. The utility company needs to verify houses that accept DR requests actually did turn off the appliances they selected to meet the request. However, there may be interesting issues that arise from this. For instance, say occupants accept a DR request and respond that they will stop using the clothes dryer (which was consuming 5kW) but unexpectedly the heat pump turns on and starts to consume 6kW.

Without load disaggregation the utility company might conclude that the occupants did not mean the DR request and no financial incentive would be given. With load disaggregation the utility company knows what has happened and indeed the occupants did meet the DR request. If the utility company does need the occupants to further reduce consumption by shutting off the heat pump another DR request can be sent.

VIII. Conclusions

We have presented our prototype Cognitive Power Meter (c-meter) that uses an open source ammeter which we have designed along with the Precision Ammeter Array (a-array) used to build the probabilistic appliance (or load) consumption models used for disaggregation. Using the hardware platform we have implemented our real-time embedded load disaggregation algorithm called μDisagg. μDisagg is the first load disaggregation algorithm to be implemented on an inexpensive low-power embedded processor that runs in real-time using a typical/basic smart meter measurement (current, in A).

Although μDisagg is still in the initial stages of development, it has high accuracies (most $> 90\%$) when determining what individual loads are running in the house and good accuracy when determining the combination of loads running using an approximation (about 80%). Our future work includes: investigating how and if we can simplify building the house model, and adding the ability to recognize new loads as new appliances over time.

Acknowledgments

Thanks to Technical Support at Analog Devices for helping with the final ammeter design. Special thanks to the BCIT School of Energy for providing lab space and research seed funding. Research partly supported by grants from the National Sciences and Engineering Research Council (NSERC) of Canada, and the Graphics, Animation, and New Media Network of Centres of Excellence (GRAND NCE) of Canada.

Schematics & Source Code

All schematics and firmware source code have been released as open source and can be downloaded from *https://github.com/smakonin/c-meter*. Our load disaggregation algorithm μDisagg has not been released as open source as of yet, it is currently under active development.

References

[1] B. Seal and R. Uluski, "Integrating Smart Distributed Energy Resources with Distribution Management Systems," *The Electric Power Research Institute (EPRI)*, 2012.

[2] S. Makonin, F. Popowich, and B. Gill, "The Cognitive Power Meter: Looking Beyond the Smart Meter," in *Electrical and Computer Engineering (CCECE), 2013 26th IEEE Canadian Conference on*, 2013, pp. 1–5.

[3] F. Sultanem, "Using appliance signatures for monitoring residential loads at meter panel level," *Power Delivery, IEEE Transactions on*, vol. 6, no. 4, pp. 1380–1385, 1991.

[4] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[5] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 1, pp. 23 –31, 2012.

[6] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *11th International Conference on Data Mining*, 2010, pp. 747–758.

[7] J. Kolter, S. Batra, and A. Ng, "Energy disaggregation via discriminative sparse coding," in *Proc. Neural Information Processing Systems*, 2010.

[8] J. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," *Journal of Machine Learning Research - Proceedings Track*, vol. 22, pp. 1472–1482, 2012.

[9] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012.

[10] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 1, pp. 76–84, 2011.

[11] D. Bergman, D. Jin, J. Juen, N. Tanaka, C. Gunter, and A. Wright, "Non-intrusive load-shed verification," *IEEE Pervasive Computing*, vol. 10, no. 1, pp. 49–57, 2011.

[12] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, "AMPds: A Public Dataset for Load Disaggregation and Eco-Feedback Research," in *Electrical Power and Energy Conference, The Annual (EPEC)*, 2013, pp. 1–6.