Robust Implementation of Distributed Algorithms for Control of Distributed Energy Resources

Stanton T. Cady[†], Alejandro D. Domínguez-García[†] and Christoforos N. Hadjicostis[†‡]
[†]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL
[‡]Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus
E-mail: {scady2, aledan}@ILLINOIS.EDU, chadjic@UCY.AC.CY

***Abstract*—This paper discusses the hardware implementation of distributed control strategies that could be used to enable the utilization of distributed energy resources for the provision of grid support services. Although the capacity of individual resources may be small, the large number of them present in many distribution networks suggests that under proper coordination and control, the collective capacity of the resources could provide useful system-level ancillary services. The software necessary to implement the control strategies is discussed and experimental results are presented.**

I. Introduction

It has been acknowledged that distributed energy resources (DERs) have the potential to provide ancillary services to the power system at the distribution level [1], [2], [3]. One example is using inverter-interfaced DERs (e.g., photovoltaic (PV) systems or motor drives with active rectifier inputs) to provide reactive power support. Although the primary function of these power electronics-based systems is to provide active power, many of them are capable of producing reactive power if appropriately controlled [4]. Another example is utilizing distributed energy storage (e.g. plug-in hybrid electric vehicles (PHEV) or uninterruptible power supplies (UPS)) to control active power for up and down regulation. Such resources could provide energy peak-shaving during hours of high demand and load leveling at night [5].

Proper coordination and control of DERs is essential to enable the extra functionality needed to provide ancillary services. One potential control architecture relies on a centralized strategy where each DER is coordinated from a central controller. In this case, the central controller issues a command to each DER to provide a certain amount of, for example, active or reactive power so that the DERs collectively meet the total demand. In addition to complete knowledge of the available DERs at any given time, a communication network connecting the central controller with each distributed resource is required to realize this control strategy.

The focus of this paper is on the implementation and experimental validation of alternative strategies for coordinating and controlling DERs in a distributed manner [6]. Compared to a centralized approach, distributed strategies offer several potential advantages: i) a communication link between a controller and all of the distributed resources is not required, reducing costs associated with communication infrastructure, ii) complete knowledge of the distributed resources available is not required and iii) the effects of faults and/or unpredictable behavior of the DERs can be more easily isolated, resulting in an overall network that is more resilient. The distributed control strategy we propose to use allows each distributed resource to make a local control decision based on information attained by communicating with other "close-by" resources. The result of the collective local control decisions should have the same effect as a centralized control strategy. We pursue this approach by demonstrating distributed algorithms that would enable DERs to be used for grid support.

In our setup, DERs outfitted with a wireless transceiver can be thought of as nodes in a stationary, yet unplanned ad-hoc network. Through an iterative process in which each node exchanges information with neighboring nodes, each DER in the network computes the amount of resource it needs to provide, such that collectively, the nodes meet the overall system demand. We illustrate the implementation of algorithms that solve this distributed resource allocation problem using commercial hardware and protocols that are robust to asynchronous communication.

The remainder of this paper is organized as follows. Section II provides some background on graph theory and presents the two iterative distributed algorithms used. Section III describes the hardware and software used to implement the algorithms and presents results for a 4-node network. Section IV presents concluding remarks.

II. Distributed Control Algorithm Description

The nodes where resources are located and the exchange of information between them can be described by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the vertex set with each vertex corresponding to a node, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges representing communications links between nodes. The directed edge $(j, i) \in \mathcal{E}$ if node j can receive information from node i while the edge is undirected if $(j, i) \in \mathcal{E}$ and $(i, j) \in \mathcal{E}$, i.e. the communication link between node j and node i is bidirectional. All nodes that can transmit information to node j are said to be neighbors of node j and are contained in the set $\mathcal{N}_j = \{i \in \mathcal{E} : (j, i) \in \mathcal{V}\}$. The number of nodes that have node j as a neighbor, i.e., the nodes to which node j can transmit information, is called the out-degree of j and is denoted $\mathcal{D}_j^+$.

A. No Constraints on Node Capacity

We first consider the case where there are no limitations on node capacity. Although this is unrealistic, the algorithm provided is the basis for solving the constrained case discussed next.

Consider a network consisting of $n+1$ nodes with 1 leader node that knows the total amount of resource (e.g. active or reactive power) ρ_d to be supplied by the network. Let x_j be the active or reactive power supplied by each node and define the collective active or reactive power supplied by the nodes as $\rho = \sum_{j=1}^{n} x_j$. Assume the leading node can send information to $l \geq 1$ nodes and initially commands ρ_d / l units of active or reactive power from each of them. After the leading node sends the initial command, it will not communicate further with the remaining nodes unless ρ_d changes.

Given that there are no constraints on node capacity, a simple solution would be to have each of the l nodes provide ρ_d / l units of active or reactive power while the remaining $n-l$ nodes provide none. This strategy could not be adapted to the constrained case, however, if the l nodes the leader initially sends a command to cannot collectively meet the demand. Instead, we use an iterative algorithm that, after m steps, allows the remaining n nodes to split the total demand in order to collectively provide ρ_d.

Let $\pi_j[k]$ be the value of the proposed algorithm at node j at iteration k, and define the corresponding vector of values as $\pi[k] = \left[\pi_1[k], \pi_2[k], \ldots, \pi_j[k], \ldots, \pi_n[k]\right]'$. The iterative algorithm we use will update the value at node j based upon i) its current value, $\pi_j[k]$, and ii) the current value of the neighbors of node j (nodes that can send information to j), $\pi_i[k], i \in \mathcal{N}_j$. One way to achieve this is for each node j to equally split its current value among itself and the nodes that have j as a neighbor, i.e. the nodes that j can transmit information to [7]. Thus, for each node j,

$$\pi_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\pi_j[k] + \sum_{i \in \mathcal{N}_j} \frac{1}{1+\mathcal{D}_i^+}\pi_i[k], \qquad (1)$$

where $\mathcal{D}_i^+$ is the out-degree of node i. The initial conditions in (1) at node j are set to $\pi_j[0] = \rho_d / l$ if information can be received from the leading node and $\pi_j[0] = 0$ otherwise. Provided the directed graph describing the communication links between nodes has a single recurrent class, the steady state solution of (1) is unique (for a proof see [6]).

After m iterations, we set $x_j = \pi_j[m]$ and we have that $\rho = \rho_d$. Note that this algorithm does not necessarily split the demand evenly among all the nodes but it does ensure that the overall demand is collectively met.

B. Constraints on Node Capacity

Building upon the discussion presented in the previous section, we now address the case where nodes have limits on the amount of active or reactive power they can provide. Let x_j^{min} and x_j^{max} for $j = 1, 2, \ldots, n$, be the minimum and maximum active or reactive power that node j can provide and define the corresponding minimum and maximum capacity vectors as $x^{min} = \left[x_1^{min}, x_2^{min}, \ldots, x_n^{min}\right]'$ and $x^{max} = \left[x_1^{max}, x_2^{max}, \ldots, x_n^{max}\right]'$, respectively. As before, we define the collective active and reactive power supplied by the nodes as $\rho = \sum_{j=1}^{n} x_j$, and ρ_d to be the total resource demand. Let $\chi^{min} = \sum_{j=1}^{n} x_j^{min}$ and $\chi^{max} = \sum_{j=1}^{n} x_j^{max}$ be the lower and upper limits on the collective amount of active or reactive power the nodes can provide. We assume that the collective active or reactive power supplied by the nodes is capable of meeting the demand, such that $\chi^{min} \leq \rho_d \leq \chi^{max}$.

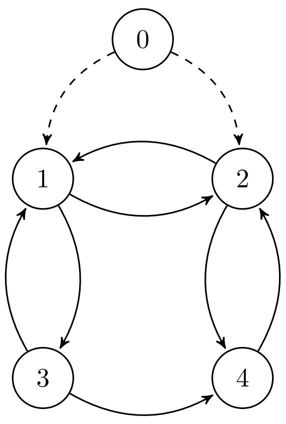


Fig. 1. Graph of 4-node network with leader node

To account for the constraints, two of the iterative algorithms given in (1) are run in parallel at every iteration. We denote the values of each algorithm at node j at iteration k as $\pi_j[k]$ and $\mu_j[k]$ and each node updates its values as

$$\pi_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\pi_j[k] + \sum_{i \in \mathcal{N}_j} \frac{1}{1+\mathcal{D}_i^+}\pi_i[k], \qquad (2)$$

$$\mu_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\mu_j[k] + \sum_{i \in \mathcal{N}_j} \frac{1}{1+\mathcal{D}_i^+}\mu_i[k], \qquad (3)$$

where $\mathcal{D}_i^+$ is the number of nodes that i can transmit information to. The initial conditions in (2) are set to $\pi_j[0] = \rho_d / l - x_j^{min}$ if j is a neighbor of the leading node and $\pi_j[0] = -x_j^{min}$ otherwise and the initial conditions in (3) are set to $\mu_j[0] = x_j^{max} - x_j^{min} := \Delta x_j > 0$.

After m iterations, we set

$$x_j = x_j^{min} + \frac{\pi_j[m]}{\mu_j[m]}\Delta x_j \qquad (4)$$

and we have that $\rho = \rho_d$ and $x_j^{min} \leq x_j \leq x_j^{max}$. As in the unconstrained case, this algorithm does not necessarily split the demand evenly among all nodes but it does ensure that the overall demand is collectively met and the resource being demanded from each node does not exceed its upper and lower capacity limits.

III. Experimental Verification

In order to verify the feasibility of the proposed coordination algorithms to control a network of distributed energy resources, we use commercially available hardware to implement

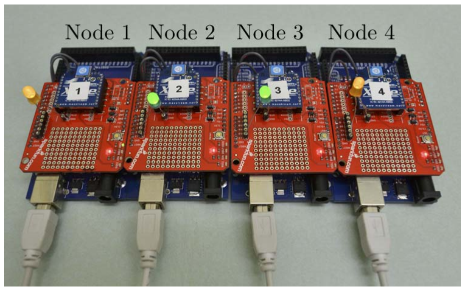


Fig. 2. Communication/processing nodes

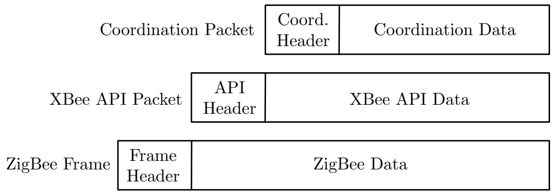


Fig. 3. Communication protocol stack

the 4-node network illustrated in Figure 1. The hardware and software setup, algorithm implementation details and experimental results are discussed next.

A. Hardware and Software Setup

The hardware used to implement the 4-node network is shown in Figure 2. Each node consists of an Arduino Mega prototyping board with AVR ATmega2560 8-bit microcontroller [8], a SparkFun Electronics XBee shield [9] and a MaxStream XB24-DMCIT-250 revB XBee module [10].

The Arduino environment was used to upload software to each device and monitor the serial output of the nodes. Each XBee is put in AP mode and the three-layered communication protocol stack illustrated in Figure 3 is implemented. The lowest layer of the stack is the IEEE 802.15.4 protocol and is implemented on the XBee chips. The middle layer is the xbee-arduino API [11] modified to enable simultaneous wireless communication with other nodes and wired communication with the computer and to facilitate the time synchronization mechanism to be discussed later. The top layer protocol was designed to transport the values of the distributed algorithms and inform each node which algorithm is being used.

B. Distributed Control Algorithm Implementation

In order to take advantage of the wireless medium used for communication, all packets used to exchange information for the distributed algorithms are broadcast and are not acknowledged upon receipt. While this removes the need for every node to know the addresses of its neighbors, the lack of acknowledgements results in the distributed algorithms being sensitive to time synchronization errors. In order to ensure convergence, all nodes are synchronized to a common reference before beginning the exchange of information.

The close proximity of the nodes during testing required that each node be programmed to ignore messages received from some of its neighbors in order to create a partially connected network. Upon startup, the neighborhood of each node and a list of nodes to ignore is stored into memory as shown in Procedure 1. After loading these lists, each node attempts to synchronize its clock with the other nodes in the network.

Procedure 1: Startup sequence

```
begin
    Build neighborhood
    Create ignore list
    Store initial conditions and capacity constraints
    Synchronize time
```

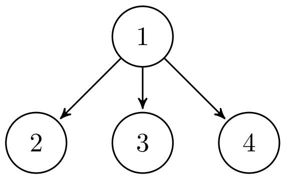


Fig. 4. Graph of time synchronization network

The time synchronization protocol used is based on the hierarchy referencing time synchronization (HRTS) protocol proposed in [12]. With only three packets, HRTS allows several nodes to synchronize clocks with a reference node. For our setup, node 1 was selected to be the reference node to which nodes 2, 3 and 4 were synchronized, forming the network illustrated by the graph shown in Figure 4. To begin the time synchronization process, node 1 broadcasts a `sync_begin` packet at time t_1, specifying a randomly chosen node from its neighborhood, e.g. node 3. Node 3 responds with a unicast packet containing t_2, the time the `sync_begin` packet was received and t_3, the time the response packet was sent. All other nodes, e.g. nodes 2 and 4, store t_2', the time the `sync_begin` packet was received, but do not respond. At time t_4, node 1 receives the response packet from node 3 and now owns all of the time stamps required to determine the clock offset,

$$d = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}. \tag{5}$$

Node 1 broadcasts a final packet containing d and t_2 and node 3 sets its local clock to be $T = t + d$, where t is the pre-synchronization local clock reading. Similarly, nodes 2 and 4 set their local clocks to be $T = t + d + t_2 - t_2'$. Each Arduino uses `timer0_millis` for local time keeping and the wiring API for the microcontroller was modified to enable the clocks to be adjusted to the synchronized time.

According to the HRTS protocol, the time stamps used for synchronization should be generated at the lowest protocol layer possible to minimize error resulting from data propagating up the stack. For our implementation, the bottom layer of the stack was inaccessible so all time stamps were generated at the middle layer. This results in an error of approximately $50\,ms$ after synchronizing the clocks of the nodes. The effects of this error are mitigated by preventing nodes from transmitting information in the first and last 50 ms of each iteration.

Procedure 2: Basic coordination algorithm

```
Input: initial command, constraints
Output: new active or reactive power command
begin
    generate random transmit time;
    foreach iteration do
        begin timer;
        while timer > iteration period do
            look for incoming packet;
            if packet available then
                store incoming value(s);
            if transmit time = time elapsed then
                broadcast current value(s);
        compute next value;
    compute final command;
```

Once the clocks of all nodes have been synchronized, one of the previously discussed coordination algorithms begins. To ensure that all nodes maintain synchronism throughout the evolution of the coordination algorithm, the number of iterations to be performed and the period of each iteration is specified beforehand. Additionally, the initial conditions of each node is stored in memory as well as any capacity constraints. The function outlined in Procedure 2 describes the basic routine that each node executes to participate in the coordination algorithm. This function requires the initial command and optionally accepts any capacity constraints as arguments. To avoid collisions, each node generates a random transmit time within the iteration period that it will use to broadcast its value(s) to its neighbors throughout the iterative process. While the purpose of our experimental setup was to design and verify communication protocols that enable a network of nodes to iteratively solve a resource allocation problem, the value computed at the end of the iterations could be used to control an actuator capable of providing, for example, active or reactive power.

C. Experimental Results

Both of the distributed algorithms discussed in Section II were implemented and tested. For each case, the network of nodes represented by the graph in Figure 1 is implemented and the total resource demand from the network is $\rho_d = 1$. We assume that the initial values at all nodes are zero and that node 0 is the leader node which splits the initial command ρ_d in half and sends it to nodes 1 and 2. In each case, the iteration period for all nodes was set to be 500 ms and 14 iterations were performed. We first discuss results for the case where there are no constraints on node capacity and then discuss results

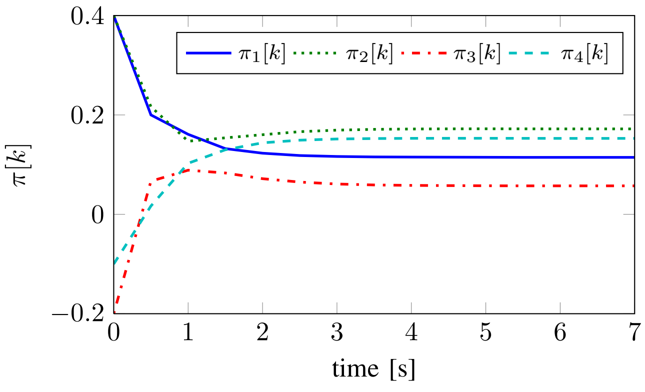


Fig. 5. Unconstrained Results

for the case when all nodes have upper and lower bounds on the amount of resource they can provide.

1) Unconstrained Capacity: According to (1), each node updates its values as follows

$$\begin{aligned}\pi_1[k+1] &= \frac{1}{3}(\pi_1[k]+\pi_2[k]+\pi_3[k]),\\ \pi_2[k+1] &= \frac{1}{3}(\pi_1[k]+\pi_2[k])+\frac{1}{2}\pi_4[k],\\ \pi_3[k+1] &= \frac{1}{3}(\pi_1[k]+\pi_3[k]),\\ \pi_4[k+1] &= \frac{1}{3}(\pi_2[k]+\pi_3[k])+\frac{1}{2}\pi_4[k],\end{aligned} \tag{6}$$

with $\pi_1[0] = \pi_2[0] = \frac{1}{2}$ and $\pi_3[0] = \pi_4[0] = 0$. Letting $\pi[0] = \big[\pi_1[0], \pi_2[0], \pi_3[0], \pi_4[0]\big]'$, we have

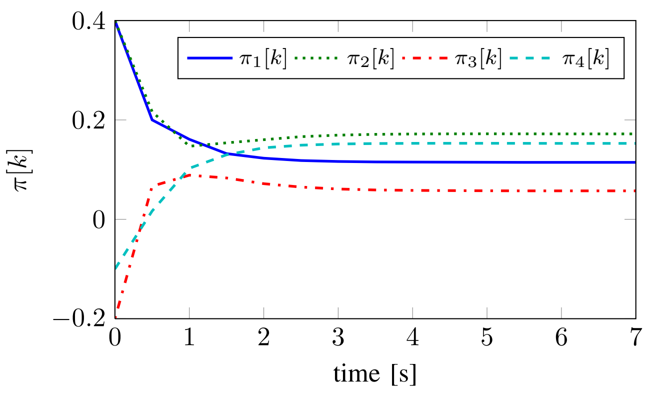
$$\begin{aligned}\pi[k+1] &= P\pi[k],\\ \pi[0] &= \big[\tfrac{1}{2}, \tfrac{1}{2}, 0, 0\big]',\end{aligned} \tag{7}$$

where

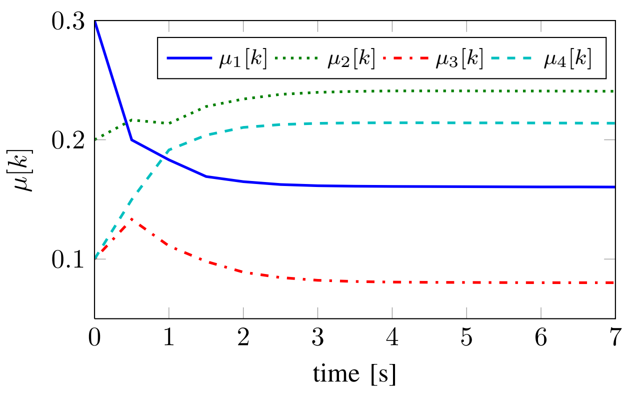
$$P = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 1/3 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/2 \end{bmatrix}. \tag{8}$$

Figure 5 shows the evolution of the values of π computed at each node throughout the iterative process. It can be seen that the nodes converge to their steady state values in approximately 5 seconds, or 10 iterations. After performing 14 iterations, each node returns the computed amount of resource it should provide and we have $x = \pi[14] = \big[0.2298, 0.3447, 0.1194, 0.3064\big]'$. This case is an example where the nodes do not equally split the total resource demand among themselves due to the directed edge between nodes 3 and 4.

2) Maximum and Minimum Constrained Capacity: To account for capacity constraints at each node, the algorithms given in (2) and (3) are executed in parallel. According to (2), the first set of values are updated at each node as given in (6)



(a) Evolution of $\pi[k]$



(b) Evolution of $\mu[k]$

Fig. 6. Constrained Results

while the second set of values are updated according to (3) as follows

$$\begin{aligned} \mu_1[k+1] &= \frac{1}{3}(\mu_1[k] + \mu_2[k] + \mu_3[k]), \\ \mu_2[k+1] &= \frac{1}{3}(\mu_1[k] + \mu_2[k]) + \frac{1}{2}\mu_4[k], \\ \mu_3[k+1] &= \frac{1}{3}(\mu_1[k] + \mu_3[k]), \\ \mu_4[k+1] &= \frac{1}{3}(\mu_2[k] + \mu_3[k]) + \frac{1}{2}\mu_4[k]. \end{aligned} \tag{9}$$

Let the minimum amount of resource the nodes can provide be $x^{min} = \left[0.1, 0.1, 0.2, 0.1\right]'$ and the maximum amount of resource be $x^{max} = \left[0.4, 0.3, 0.3, 0.2\right]'$. The maximum constraints are chosen so that $\chi^{max} = 1.2 > \rho_d$, i.e. the nodes can collectively provide the total resource demanded. The initial conditions are $\pi_1[0] = \pi_2[0] = 0.4$, $\pi_3[0] = -0.2$, $\pi_4[0] = -0.1$, $\mu_1[0] = 0.3$, $\mu_2[0] = 0.2$ and $\mu_3[0] = \mu_4[0] = 0.1$. Letting $\pi[0] = \left[\pi_1[0], \pi_2[0], \pi_3[0], \pi_4[0]\right]'$ and $\mu[0] = \left[\mu_1[0], \mu_2[0], \mu_3[0], \mu_4[0]\right]'$, we have

$$\begin{aligned} \pi[k+1] &= P\pi[k] \\ \pi[0] &= \left[0.4, 0.4, -0.2, -0.1\right]' \\ \mu[k+1] &= P\mu[k] \\ \mu[0] &= \left[0.3, 0.2, 0.1, 0.1\right]', \end{aligned} \tag{10}$$

where P is given in (8). The evolution of π and μ at each node is shown in Figure 6. The plots illustrate that similar to the constrained case, both values converge to their steady state values in approximately 5 seconds, or 10 iterations. After performing 14 iterations, each node computes the amount of resource it should provide according to (4) and we have $x = \left[0.3143, 0.2428, 0.2714, 0.1714\right]'$. As in the constrained case, the nodes do not evenly split the total demand among themselves but they do reach a solution that collectively meets the total demand without exceeding any individual capacity constraints.

IV. Concluding Remarks and Future Work

Proper coordination and control can expand the functionality of DERs and enable them to provide ancillary services to power systems. We have discussed the implementation of distributed control strategies that can be used to determine the amount of resource each DER should provide in a distributed fashion and have demonstrated these algorithms through the use of commercial hardware. The experimental results demonstrate that the communication protocols proposed are robust to asynchronous communication. Further work will expand the protocols to be robust to unreliable communication links.

References

[1] G. Joos, B. Ooi, D. McGillis, F. Galiana, and R. Marceau, "The potential of distributed generation to provide ancillary services," in *Proc. of IEEE Power Engineering Society Summer Meeting*, vol. 3, Seattle, WA, 2000.

[2] M. Baran and I. El-Markabi, "A multiagent-based dispatching scheme for distributed generators for voltage support on distribution feeders," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 52 –59, Feb. 2007.

[3] K. Turitsyn, P. Šandulc, S. Backhaus, and M. Chertkov, "Distributed control of reactive power flow in a radial distribution circuit with high photovoltaic penetration," in *Proc. IEEE Power and Energy Society General Meeting*, july 2010.

[4] A. D. Dominguez-Garcia, C. N. Hadjicostis, P. T. Krein, and S. T. Cady, "Small inverter-interfaced distributed energy resources for reactive power support," in *Proc. Applied Power Electronics Conference and Exposition*, March 2011, pp. 1616 –1621.

[5] C. Guille and G. Gross, "A conceptual framework for the vehicle-to-grid (v2g) implementation," *Energy Policy*, vol. 37, no. 11, pp. 4379 – 4390, 2009.

[6] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proc. IEEE International Conference on Smart Grid Communications*, Gaithersburg, MD, October 2010, pp. 537 –542.

[7] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215 –233, January 2007.

[8] (2011, April) Arduino. [Online]. Available: http://www.arduino.cc

[9] (2011, April) Sparkfun electronics. [Online]. Available: http://www.sparkfun.com/

[10] (2011, April) Digi international inc. [Online]. Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp

[11] A. Rapp. (2011, April) xbee-arduino. [Online]. Available: http://code.google.com/p/xbee-arduino/

[12] H. Dai and R. Han, "Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, pp. 125–139, January 2004. [Online]. Available: http://doi.acm.org/10.1145/980159.980173