COSMGrid: Configurable, Off-the-shelf Micro Grid

Jonathan Fürst
IT University of Copenhagen
Copenhagen, Denmark 2300
Email: jonf@itu.dk

Nik Gawinowski
IT University of Copenhagen
Copenhagen, Denmark 2300
Email: nikg@itu.dk

Sebastian Büttrich
IT University of Copenhagen
Copenhagen, Denmark 2300
Email: sbut@itu.dk

Philippe Bonnet
IT University of Copenhagen
Copenhagen, Denmark 2300
Email: phbo@itu.dk

***Abstract*—Access to modern energy services should be universally available by 2030. This is a goal of the United Nations. A promising approach to deliver on this commitment is based on microgrids that coordinate power generation, storage and usage in a local community. Microgrids constitute an attractive option in the presence of abundant renewable energy sources, and in the absence of robust transnational power grid infrastructure. An important problem is then to design cheap, resilient and configurable microgrids that can be assembled from off-the-shelf components and managed by non specialists. In this paper, we introduce COSMGrid, a microgrid platform based on commodity hardware and open source, open protocol software. The design of COSMGrid relies on a network of microcontrollers that monitor and control stand-alone power generation and storage nodes. As a result, COSMGrid can readily integrate existing stand-alone photovoltaic installations. COSMGrid can be configured based on the characteristics of the power electronics hardware that is available, or based on the power sharing policies agreed upon by a community of end-users. It is an important step towards a popular, open microgrid solution that can be appropriated by local communities in developing regions.**

I. Introduction

Populations in developing countries lack access to modern energy services. Estimates show that 1.4 billion people do not have access to electricity, that 1 billion people have only access to unreliable electricity networks and that 2.5 billion people rely on traditional biomass (mostly wood) for cooking [1]. The situation is particularly severe in rural areas where there is no access to a functioning national power grid. This is a significant problem with respect to the necessary reduction of CO_2 emissions. Indeed, relying on burning wood or on diesel motors for energy services is harmful for the environment (e.g., it causes desertification or pollution) and it is hindering further social and economic development (see e.g., [2], [3], [4]). This is why the United Nations have set it as a goal to achieve universal access to modern energy services by 2030 [1].

Two key enabling technologies that will support this evolution are renewable energy and ICT (Information and Communications Technology). Renewables allow a CO_2 free generation of electrical energy, while ICT plays an important supporting role in terms of their efficient usage. The geographical location of many developing countries makes photovoltaic (PV) power a suitable solution. Studies have shown for instance that PV can already compete or even be more economical than diesel power generation for rural solutions in Africa [5]. PV power has higher initial costs, but maintenance and operating costs are much lower. The lifetime cost for a small-scale system is less than for a diesel system [6].

PV systems can be stand-alone or connected to a microgrid. A microgrid connects a local community of producers and consumers. It can be sized from the scale of a handful houses up to a whole village. In a stand-alone system, PV generation is directly linked to sun insolation. As a result, there is no PV generation at night. Batteries are usually introduced to relax this constraint and to store energy produced during the daytime. Batteries must be dimensioned properly, so that energy outages are minimized. This is a complex problem. A solution is to over-engineer the installed system to guarantee a large range of operating modes. However, such over-engineering leads to high costs and low utilization on average. Microgrids solve this problem in an elegant and efficient way. Microgrids lessen the total energy storage needed because storage is physically or virtually shared between the participants of the microgrid, allowing a much more configurable, efficient and reliable system.

While numerous microgrid approaches have been proposed, they all require specific power electronics for their operation. By contrast, we present in this paper the design of COSMGrid, a microgrid that (i) is based on already existing hardware (e.g., that could be deployed in a legacy stand-alone system), (ii) relies on economical off-the-shelf power electronic hardware for the composition of the microgrid and (iii) has an "open" monitor and control architecture that is based on a popular microcontroller platform. We have implemented and evaluated our design in form of a testbed using AVR microcontrollers (Arduino).

II. Related Work

The research on microgrids and its practical application follows two main architectural approaches. The first approach imitates the structure of the national grid by centralizing access to multiple power generation devices through a common battery bank. The battery bank is charged by different energy producers, for example a PV plant or a wind mill owned by different members of the community. An inverter is then providing energy from this battery bank to all households. Examples or microgrids applying this design can e.g. be found in [7].

The second approach is most interesting as it allows to integrate legacy devices. It connects multiple distributed energy producers in parallel. These producers are then interconnected through either a DC or an AC grid. Controlling parallel renewable power sources can be done in one of the following ways. First, control can follow a master-slave control scheme. An inverter converts the DC directly from the PV panel into

AC that is fed it into the grid. Off the shelf grid tie inverters (GTIs) are designed to work directly with solar panels without a battery buffer, and their internal control mechanism is always trying to track the maximum power point (MPPT), as power would be lost otherwise (see [8] [9]). The master inverter is setting the voltage and frequency for the whole microgrid. The slaves act as current sources, adjusting frequency and voltage accordingly (see e.g. [10]). Second, control can be established with an average current sharing control scheme that administers the distribution of current by using average control distribution signals. Third, control can be implemented autonomously at each distributed source by droop control. Droop control, in which the behavior of power generators is simulated, is the probably most used method. It is for instance used in [11]. Marwali et al. [12] combines both average current sharing control and droop control.

With COSMGrid, we follow the master/slave approach for connecting multiple producers and consumers in parallel. Our design relies on batteries to store the energy produced by renewable sources. We thus had to devise a power control mechanism to measure the output of inverters and regulate their behavior.

III. COSMGrid Design

The design of COSMGrid is modular, consisting of micro-grid nodes that can be freely connected with each other. Nodes can represent different entities, e.g. single households, building components, or a common infrastructure (e.g. street lightning). Each node is equipped with two inverters: a sine wave inverter and a grid-tie inverter. The sine wave inverter provides the voltage, frequency and phase for the whole microgrid. There is thus a single node in the system whose sine wave inverter is active at any point in time. This node is the master of the microgrid. The grid-tie inverter (GTI) converts the DC from the batteries (or directly from a renewable energy source) and feeds it into the microgrid as required. All nodes, except the master, are slaves and rely on their GTI to adapt their state to the microgrid conditions fixed by the master. Master and slave roles are not static. They change over the lifetime of the microgrid, depending on factors that may for example include the battery level and the actual PV production. The same goes for microgrids themselves. Nodes can theoretically change their membership (from microgrid to another) dependent on the current power requirements or user preferences.

There can be variations to the default design. Some nodes could just have one inverter and not use a battery storage at all. A possible system showing our design using two nodes can be seen in Figure 1.

The COSMGrid node design is based on a generic view of the underlying off-the-shelf hardware components (renewable source, sine wave inverter, GTI). As a result, COSMGrid relies on a monitor and control infrastructure, de-coupled from power electronics, to operate the microgrid.

Existing GTIs are adjusting to the frequency, voltage and phase but are always pushing the maximum available power into the grid. As our design is usually running the inverters from the batteries, this means that they will always provide the maximum amount up to their power ratings until the battery storage is empty. This raises two problems, (i) it will discharge the batteries very fast and more significant, (ii) it will most likely lead to a scenario where we have too much power inside the grid, leading to physical destruction of the grid infrastructure as well as power outages.

We do not aim at redesigning GTIs. Instead, we designed a monitoring, control and communication structure to adapt the behavior of off-the-shelf GTIs in our microgrid. As we are not implementing the current control internally by changing the software logic of the GTI, we need to be able to turn off load if there is not enough power available and we need to be able to turn off GTIs when there is too much power in the grid. We are thus faced with a scheduling problem, which is a classical problem in distributed systems. More specifically, the issues we need to address are: (a) How to detect whether there is enough/too much/not enough power available in the grid? (b) How to control which loads and GTIs should be turned on/off? Overall, the question is whether we can build a system that can live up to these requirements. Specifically, we should investigate whether we can turn on/off GTIs and loads fast enough with respect to the situation on the grid to avoid power outages and component destruction.

The monitoring and control infrastructure of COSMGrid relies on hardware as well as software components. It can be programmed to support a range of usage policies. For instance, we are envisaging the addition of a micro payment mechanism between node owners, transforming them into micro-energy companies. This is a topic for future work. With this modular and open design, our goal is to enable a grassroots based development of small microgrids that can be an alternative to the national grid in rural areas.

IV. COSMGrid Hardware

We implemented the COSMGrid design with a simple testbed based on two nodes and one type of load. We are deliberately using inexpensive power electronic devices to implement the system. By using inexpensive devices, we show that a basic working microgrid structure can be achieved at low cost. The total cost of approx. €250 for our solution of two inverters and its monitor and control infrastructure should be compared to the cost of commercial inverters that support parallel operations out of the box such as, (i) the Victron Energy MultiPlus, limited to 6 nodes in parallel, at €900, (ii) the Studer Xtender Series at €1000, or (iii) the SMA Sunny Island at €3000. Our basic setup consists of a 300 W pure sine wave inverter, a 300 W GTI and microcontroller / sensor parts[1]. Each node in our testbed, also includes a 100 W PV panel, a 100 Ah battery, and charge controller.

Our implementation of COSMGrid relies on a centralized architecture, i.e., we introduce a central unit to which all the nodes are connected. The central unit receives monitoring data from the nodes and sends control signals back to them. It also stores past data in order to learn from it and adapt its actions accordingly. We are using the ARM powered Raspberry Pi to implement the central unit due to its low cost, its

[1]The cost based on current (May 2013) prices in Denmark is as follows: 300 W GTI €75, 300 W Pure Sine Inverter €80, Micro Controller and Monitor Parts €100 (Arduino Uno €24, DC Current sensors €28, AC Current Transformer sensor €22, 8 channel relay €15, AC/AC Transformer Sensor €6, Resistors/Capacitors/Cables €5) for an approximate total of €250.

sufficient computing capabilities and its generic programming capabilities [13].

Each node uses an AVR microcontroller (Arduino platform) that is controlling and monitoring the node locally and communicating with the central unit. The sensors used to monitor the grid are:

- *Split Core Transformer.* For measuring alternating current after the inverters and at the node's load.
- *AC to AC Power Adapter.* For measuring AC voltage at the node's load.
- *Hall Effect Based Current Sensor.* For measuring direct current to and from the battery.
- *A Voltage Divider.* For measuring the voltage of the battery.

Mechanical relays are used to control the flow of current in our microgrid. This is necessary to balance the energy in the microgrid (demand and supply in the microgrid need always to match). A cable connection is used for communication between the central unit and the Arduino boards in our testbed. More mature iterations of our system will most likely rely on a form of wireless communication.

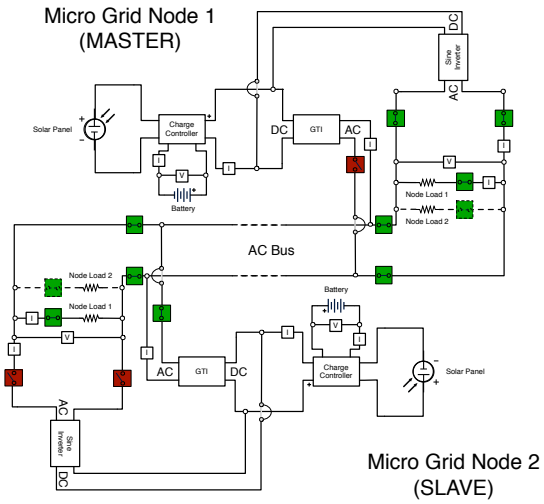

Fig. 1. COSMGrid Hardware Architecture

Figure 1 shows the position of the different sensors and relays for a single node. Starting from the DC side, we have one hall-effect based current sensor measuring the current going into the battery and one that is measuring what is going into the inverter. A voltage sensor is measuring the voltage at the battery. In this constellation we are able to monitor the battery level and how much we feed into the grid directly. The present production can easily be calculated with this information.

Moving on to the AC side we use three split core transformers to sensor the current coming out of the sine wave inverter and GTI, as well as the current going towards the microgrid. This combined with a voltage sensor allows us to calculate all relevant power values (active, apparent, reactive) and their physical direction.

For the control part we are using six relays in total. Two relays for both, live and neutral after our sine inverter and one relay at the live after the GTI. The reason for only using one relay after the GTI is that by design it will shut down if there is already one conductor (live or neutral cable) not connected due to its anti-islanding requirement.[2] Having the relays on the AC and not the DC side assures that their current limit will not be exceeded. In addition, we are using a relay to be able to cut off/turn on each load, and two relays to close/open the connection of a node's system towards the microgrid.

V. COSMGrid Software

In our centralized architecture, each node is monitoring its status and the grid status and communicating to the central unit. The central unit takes control decisions and communicates them back to the node, which executes them. In our testbed, we rely on Arduino motes to implement monitoring and control on the nodes, while we rely on Python scripts running on a Raspberry Pi for the central unit. In addition, a web server is running, providing a web interface for node owners and the grid administrator. Visualization is provided through a local *ThingSpeak* server. In the rest of the section, we describe the software system installed on the microcontroller at the node and on the central unit.[3]

A. Microcontroller at the Node

The AVR microcontroller has three main tasks: (i) it collects and processes data from various sensors, (ii) it communicates with the central unit and (iii) it controls the relays based on the commands it receives from the central unit or based on its self-protection mechanism logic. The key issues form a design point of view are whether we can collect precise enough data, and process it fast enough so that our system can control the microgrid in a timely fashion.

1) Monitoring the State of the Microgrid: All previously mentioned sensors deliver an analog voltage signal to the Arduino board. This analog signal is then converted by the analog-to-digital converter (ADC) to a digital signal. The ADC of the Arduino board provides a resolution of $10\,\mathrm{bit}$ that correspond to integer values from 0–1023 [15]. This fact is important when considering the accuracy for each sensor. Even a 100% precise sensor will not give 100% precise digital values. The $10\,\mathrm{bit}$ resolution, in combination with the Arduino's $5\,\mathrm{V}$ reference voltage, leads to $5\,\mathrm{V}/1024 = 4.9\,\mathrm{mV}$ as the smallest detectable voltage.

a) Sampling Rate: There are different constraints to our sampling rate. Firstly the ADC converter is limited by the clock speed of the processor. When the prescale factor is set to 128 with a clock speed of $16\,\mathrm{MHz}$ it relates to $16\,\mathrm{MHz}/128 = 125\,\mathrm{kHz}$ (see `wiring.c` [16]). One conversion from analog to digital takes 13 clock cycles which leads to $125\,\mathrm{kHz}/13 = 9600\,\mathrm{Hz}$. This value corresponds to the sampling rate of around 10.000 times per second mentioned from the Arduino data sheet.

We decided to choose integer types to be sent over the serial port. As integers are represented by $16\,\mathrm{bit}$, we will

[2] Anti-islanding protection is shutting the GTI down in case of a grid failure, like for example a power outage [14].

[3] The source code can be found on https://github.com/jf87/COSMGrid.

theoretically be able to send 7200 integer values per second.[4] To still maintain a higher sampling rate, we are going to calculate average values of sampled data. But more about this in the single sensor sections.

All sampling tasks are structured in one function named `collectData()`, which is called from the `loop()` function and therefore itself looping during the whole lifetime of the node.

b) DC Voltage Sensor: We use a simple voltage divider to measure the battery voltage with the maximum input range of $5\,\mathrm{V}$ of the Arduino. The accuracy that can be theoretical achieved is a resolution of:

$$\frac{5\,\mathrm{V}}{1024} \cdot \frac{20\,\mathrm{V}}{5\,\mathrm{V}} = 10\,\mathrm{mV} \tag{1}$$

c) Directed Current Sensor: The hall-effect based current sensor is giving voltages from 0–5 V, corresponding to ampere values of -50 to $50\,\mathrm{A}$. This means that for a current of $0\,\mathrm{A}$, the sensor will theoretically return a value of exactly $2.5\,\mathrm{V}$ (511-512 as analog read). Its sensitivity is $40\,\mathrm{mV/A}$. With the ADC converter resolution of $1024\,\mathrm{bit}$ we can obtain the actual sensor resolution with:

$$\frac{5\,\mathrm{V}}{1024 \cdot 0.04\,\mathrm{V/A}} = 0.122\,\mathrm{A} \tag{2}$$

d) Alternating Current and Voltage Sensor: On the AC side we are using a split core transformer and an AC/AC adapter for measuring current and voltage respectively. Using this combination of a simultaneous measurement of current and voltage curves enables us to get precise values for the different power types.

The setup of a voltage and current sensor in combination with the Arduino platform is already for most parts been done by the *openenergymonitor.org* project. They provide a library that includes the AC power calculations. For AC it is more important than it is for DC to take samples in a relatively fast rate. Even assuming, voltage/current perfectly following a sinus curve, according to the *Nyquist theory,* a minimum of $100\,\mathrm{Hz}$ is needed [17].

To eliminate potential alias frequencies and to reproduce an accurate waveform, sampling frequencies of 10 times the input frequency are recommended. Using the default settings in the library there are around 53 samples taken per wave cycle, leading to around 2650 samples per second [18]. Modifications we added were functions for returning the RMS (Root mean square) voltage and active power.

2) Controlling the Flow of Power: The mandatory requirement to avoid failure within the microgrid is addressed by the control part. Two rationales for control procedures are defined: (i) to enforce a balancing policy, and (ii) to avoid faults in the grid. There are also two places where a control can be triggered: (i) in the logic of the Arduino itself, and (ii) inside the central unit. In the following we discuss the software of the Arduino. We discuss control in the central unit in B.

We abstract the switching of relays in the context of a state machine. We call each state a mode. Modes represent the settings for all the relays at one point in time. For example if we want a node to be disconnected from the grid and to only use its own produced power and stored energy, we set the relays to a predefined state. In this case, relays to the grid and GTI would be open whilst at the same time the relays to the load and sine wave inverter would be closed. This ensures that relays are never set in a way that contradicts to each other and results in hardware defects and failures.

a) Self Protection Function in the Arduino: Mode switches can also be triggered by an internal Arduino logic. If communication to the central control unit gets disrupted, or commands by the Arduino are not executed in time a simple safety function is implemented. The power flow in front of the master sine wave inverter is steadily monitored, as we must avoid power flowing the wrong direction inside the inverter. In case where the power monitored is about to reach zero, emergency actions are taken by cutting off the affected inverter from the grid.

3) Communication with the Central Unit: We differentiate between two different types of sensor information, (i) instant sensor information, and (ii) averaged sensor information. We require instant sensor information from the AC sensor and the AC voltage sensor on the AC side of the inverters, because we need to react quickly to stop power going into the inverters. For the other sensors it is sufficient to send their averaged data every minute. These data is only needed to implement the usage policies and to build up knowledge of the consumption and production patterns.

B. Central Unit

The central unit is performing three main tasks: (i) it is making decisions in regards to balancing the power in the microgrid; (ii) it is performing the collection of sensor data and storing it in a database; (iii) it is running a web application for visualization purposes and to allow node owners to make policy settings (which we do not describe here for lack of space).

The software is implemented in Python, using *pySerial* (http://pyserial.sourceforge.net) for communication with the Arduinos and *CherryPy* (http://cherrypy.org) as a web framework. So we are more or less following the client/server pattern. We leave the definition of a decentralized, peer-to-peer solution as future work.

1) Controlling the Nodes: From the central unit point of view, microgrid nodes are like black boxes in terms of their available PV power, their inverter's power ratings and amount of battery storage. New nodes can be connected to the microgrid, whereas some nodes might be disconnected during the lifetime of the grid. In the following, we describe how we deal with node connection and disconnection. Nodes connected to the central unit are identified and assigned a unique ID. This ID is saved in the EEPROM (Electrically Erasable Programmable Read-Only Memory) to ensure consistency with the naming of the nodes. Once a node is connected to the grid, it will keep its unique ID forever and no two nodes will have the same ID. A default start mode is defined for a new node connecting to the grid. After a first set of sensor values from all the nodes connected to the microgrid is sent to the central unit, the central unit choses a node to be the master node. This is based on

[4] $\frac{115200\,\mathrm{bit/s}}{16\,\mathrm{bit/integer}} = 7200\ \mathrm{integer/second}$.

two factors, the battery level and the energy production. The central unit regularly checks for new nodes that haven't been connected to the microgrid yet. The central unit ensures the safety of the microgrid by taking actions in case of (i) too much power in the microgrid, and (ii) not enough power in the microgrid: (i) too much power in the grid is checked by monitoring the active power in front of the inverter of the master node (sine wave inverter). Precisely, if current from the microgrid is about to flow into the inverter, it is a sign that there is too much power in the grid. Countermeasures are, firstly the increase of load, and secondly the decrease of power by turning off a GTI. (ii) Not enough power in the grid is checked by monitoring voltage. When the voltage level drops significantly, there is not enough power available. Countermeasures are firstly, the decrease of load by switching on a node (e.g., it is set in a mode where the local load is powered by the sine wave inverter and the GTI is providing power to the grid simultaneously) and secondly, the decrease of load by actually switching load off (introducing load priority to influence the scheduling of load switch-off is a topic for future work).

2) Data Collection and Storage: Apart from the control part, sensor data are collected for visualization purposes. Our system uses two databases. One contains just very recent values necessary for decisions to keep the microgrid alive. The second database contains averaged values that are used for visualization and in the future for control decisions based on policies or past learnt effects.

VI. Testbed Evaluation

The evaluation of our microgrid design focuses on whether our testbed can react fast enough to changing conditions. More specifically, we study how our system reacts to overload in the microgrid or how fast our system can turn on/off a GTI.

We measure the conditions in the microgrid with a basic oscilloscope (PicoScope 3204, bandwidth 60 MHz and maximum sampling Rate 500 MS/s). In stand-alone mode we reach an efficiency of 81% at the beginning of the load spectrum of the inverter, 84% in the middle and 82% at the top. Our chosen GTI shows an efficiency of 72%. For the whole microgrid we measured a total efficiency of 78%, when sine wave inverter and GTI are run in parallel. Efficiency is defined in terms of how much power is preserved after the losses due to conversion from DC (of battery or solar panel) to AC and the losses due to the grid forming process between parallel inverters.

A. Reaction to Overload in the Microgrid

This experiment aims at showing the reaction time (latency) of our grid control mechanism with respect to an overload in the grid. An overload in this case means that we switch on loads that collectively require more power than is available power in the grid at that point in time.

Tests are done with two nodes in parallel. Loads consist of resistive loads in form of 100 W incandescent light bulbs that can be switched on or off individually. One node is the master node of the system and the other node runs as GRIDSUPPORTED (in a mode where it does not contribute to the grid). Then we switch too much load on at the slave node (at 40 ms on the x-axis on figure 2). This triggers a startup

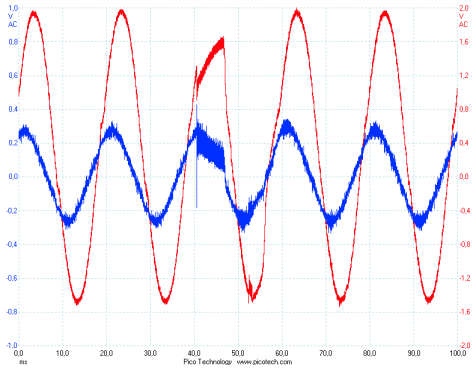

Fig. 2. Reaction to Overload

of the GTI and relay switches (so that the node actually feeds power into the microgrid). Measurement of voltage and current is done at the master node.

After switching on too much load the inverter(s) can not keep holding the voltage and current, causing a distortion and drop (between 40 and 46 ms on the x-axis on figure 2). The grid's sensors measure this drop and the micro controller is executing counter measure commands in form of relay switches. In figure 2 the system reacts in only 20 ms. During several tries we came to values ranging from 20 to 600 ms.

The reaction time seems to depend on (i), the current execution state of the software when the load is switched on and (ii), on the magnitude of the load that is switched on (50 W, 100 W, 200 W etc.). We cannot predict the current running state of the software when an overload occurs. As such we cannot change this fact. It is however possible to change the sensitivity to voltage drops in our software. For example reacting already to small drops. But we need to make a tradeoff between normal fluctuations in the voltage curve (for example the decrease in voltage due to switching on of a load) and the drop in voltage due to a "real" overload.

B. Grid Tie Inverter Startup Process

An important element in our microgrid is the start-up time of the GTI. Switching a GTI on, it has to track the voltage and current curves of the existing grid. After monitoring several cycles it will start pushing power slowly into the grid. This experiment will show us factors that influence the latency (e.g. the type of load). Understanding the exact delay will be important for further software improvements.

Tests are done with two nodes in parallel. Loads consist again of resistive loads in form of 100 W incandescent light bulbs that can be switched individually. Then we switch on too much load to cause the GTI to start up. Time is measured until the GTI is fully operational.

As figure 3 shows the GTI slowly pushes more power into the grid until it reaches its maximum ($300W$) after $52s$. In average our measurements showed a start-up time of $50s$. After reaching the maximum, the power output fluctuates by 50 W. Several experiments have shown that the actual load in the grid does not have an affect on this time.

We did not find any correlation between the load and the start-up latency of the GTI. Because we cannot change this

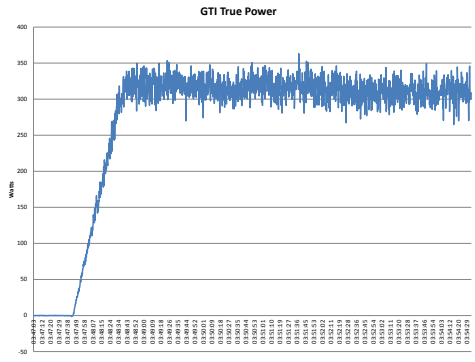

Fig. 3. GTI Active Power Output

timing constraint, we need to consider it in our software and design of the microgrid to avoid power outages due to the start-up process.

VII. Conclusion

In this paper, we presented the design, implementation and evaluation of COSMGrid, a microgrid based on off-the-shelf components. Basing the system on existing power electronics hardware lowers the costs and thereby the entry level for potential users, but it also allows us to transform existing stand-alone systems into systems that are part of a microgrid. Lowering the entry level for potential users is especially important in regards to the high initial costs of PV power [6]. This is supported by the modular structure of our design. We want to minimize common costs for a microgrid community. Many systems rely for instance on a common battery bank. Participants need to agree and invest together on it. Allowing our system to be built on existing hardware of a stand-alone system allows the incremental composition of a microgrid. Some people may start connecting themselves from the beginning to the microgrid, while others join it gradually, dependent on their available resources. Future work includes more thorough evaluation with multiple nodes in the presence of a variety of threats to the microgrid operation, the design of a decentralized monitoring and control solutions, as well as the deployment of COSMGrid in a range of use cases.

References

[1] U. N. D. Programme, "Energy for a sustainable future: The secretary-general's advisory group on energy and climate change summary report and recommendations," UNDP, Tech. Rep., New York, 2010.

[2] T. A. Benjaminsen, "Fuelwood and desertification: Sahel orthodoxies discussed on the basis of field data from the gourma region in mali," *Geoforum*, vol. 24, no. 4, pp. 397–409, 1993.

[3] E. Audu, "Fuel wood consumption and desertification in nigeria," *International Journal of Science and Technology*, vol. 3, no. 1, 2013.

[4] R. M. Hassan and G. Hertzler, "Deforestation from the overexploitation of wood resources as a cooking fuel: a dynamic approach to pricing energy resources in sudan," *Energy economics*, vol. 10, no. 2, pp. 163–168, 1988.

[5] S. Szabó, K. Bódis, T. Huld, and M. Moner-Girona, "Energy solutions in rural africa: mapping electrification costs of distributed solar and diesel generation versus grid extension," *Environmental Research Letters*, vol. 6, no. 3, p. 034002, 2011.

[6] M. Kolhe, S. Kolhe, and J. Joshi, "Economic viability of stand-alone solar photovoltaic system in comparison with diesel-powered system for india," *Energy Economics*, vol. 24, no. 2, pp. 155–165, 2002.

[7] S. B. Bekiarov and A. Emadi, "Uninterruptible power supplies: classification, operation, dynamics, and control," in *Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE*, vol. 1. IEEE, 2002, pp. 597–604.

[8] S. Kjaer, J. Pedersen, and F. Blaabjerg, "A review of single-phase grid-connected inverters for photovoltaic modules," *Industry Applications, IEEE Transactions on*, vol. 41, no. 5, pp. 1292–1306, 2005.

[9] B. Yang, W. Li, Y. Zhao, and X. He, "Design and analysis of a grid-connected photovoltaic power system," *Power Electronics, IEEE Transactions on*, vol. 25, no. 4, pp. 992–1000, 2010.

[10] W.-C. Lee, T.-K. Lee, S.-H. Lee, K.-H. Kim, D.-S. Hyun, and I.-Y. Suh, "A master and slave control strategy for parallel operation of three-phase ups systems with different ratings," in *Applied Power Electronics Conference and Exposition, 2004. APEC'04. Nineteenth Annual IEEE*, vol. 1. IEEE, 2004, pp. 456–462.

[11] J. M. Guerrero, J. C. Vasquez, J. Matas, M. Castilla, and L. G. de Vicuna, "Control strategy for flexible microgrid based on parallel line-interactive ups systems," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 726–736, 2009.

[12] M. N. Marwali, J.-W. Jung, and A. Keyhani, "Control of distributed generation systems-part ii: Load sharing control," *Power Electronics, IEEE Transactions on*, vol. 19, no. 6, pp. 1551–1561, 2004.

[13] Raspberry Pi - An ARM GNU/Linux box for 25 Dollars. Take a byte!, Project Website. [Online]. Available: http://www.raspberrypi.org/

[14] R. M. Hudson, T. Thorne, F. Mekanik, M. R. Behnke, S. Gonzalez, and J. Ginn, "Implementation and testing of anti-islanding algorithms for ieee 929-2000 compliance of single phase photovoltaic inverters," in *Photovoltaic Specialists Conference, 2002. Conference Record of the Twenty-Ninth IEEE*. IEEE, 2002, pp. 1414–1419.

[15] [Online]. Available: http://arduino.cc/en/Tutorial/AnalogInputPins

[16] wiring.c - arduino - Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. - Google Project Hosting, Project Website. [Online]. Available: http://code.google.com/p/arduino/source/browse/trunk/hardware/arduino/cores/arduino/wiring.c

[17] Nyquist rate - Wikipedia, the free encyclopedia, Project Website. [Online]. Available: http://en.wikipedia.org/wiki/Nyquist_rate

[18] Explanation of the phase correction algorithm — OpenEnergyMonitor, Project Website. [Online]. Available: http://openenergymonitor.org/emon/buildingblocks/explanation-of-the-phase-correction-algorithm