

IT-Systeme in Produktion und Automatisierungstechnik

OPC UA Server

für das Smart Grid Modell

Übersicht

Das Smart-Grid-Modell (SGM) veranschaulicht, ein intelligentes Energienetz mit verschiedenen Strom- Erzeugern und Verbrauchern. Die Hardware des SGM wurde bereits durch mehrere Gruppen von Studierenden am CVH aufgebaut. Dazu gehörten sowohl Hardware-, als auch Softwarekomponenten.

Die Gruppe der Erzeuger auf dem SGM stellt sich aus den einer konventionellen Komponente (einem herkömmlichen Kraftwerk) und zwei erneuerbaren Energiequellen, einem Windrad und einem Photovoltaikpark, zusammen. Die Gruppe der Verbraucher, ist durch eine Ansiedlung von Häusern dargestellt. Alle diese Komponenten sind auf dem Modell miteinander verbunden. Bei der Verbindung handelt es sich um LED-Streifen, die sowohl grün als auch rot leuchten können. Grün steht hierbei für die erneuerbare Energie und Rot für die konventionelle. Das Windrad auf dem Modell verfügt über einen Motor, worüber die Windgeschwindigkeit veranschaulicht werden kann.

Die Ansteuerung der LED-Streifen, als auch des Windrads wird von einem Arduino übernommen, welcher per PWM die Farbe und Drehgeschwindigkeit einstellt. Dieser Arduino war nicht Teil der folgenden Arbeit, sondern wird lediglich über seine I2C-Schnittstelle angesprochen. Bis auf eine Aktualisierung des Raspberry-Pis auf das zum derzeit aktuellste Raspberry-Pi OS wurden keine Änderungen an der Hardware vorgenommen. Der Raspberry Pi wurde aktualisiert, damit die neuesten C++ Standards verwendet werden konnten.

Die bereits bestehende Software, die vorher auf dem Raspberry-Pi installiert war wurde gesichert und ist in diesem Repository unter dem Tag SS19 zu finden <https://gitlab.cvh-server.de/aco/smart-grid-modell/-/tree/ss19>. Dieser Tag stellt eine GUI bereit mit der verschiedenen Parameter des Modells angepasst werden können und ein Tageszeiten-Simulationsmodus ausgeführt werden kann. Diese Software bot jedoch keine Möglichkeit, um einen OPC UA Server sinnvoll zu integrieren, daher wurde die Software zur Ansteuerung des Modells in gänze neu geschrieben.

Entwicklungen

Als Vorbereitung zu den folgenden Entwicklungen wurde der Raspberry Pi aktualisiert, da das Betriebssystem zu sehr gealtert war, sodass die Software nicht auf den aktuellen Stand geupdated werden konnte.

I2C

Als erstes wurde eine I2C Modul entwickelt, welches es ermöglicht auf die I2C Schnittstelle des Raspberry-Pis und dadurch auf den Arduino zuzugreifen. Hierbei wurde im Gegensatz zu vorher auf die Verwendung von der Bibliothek WiringPi verzichtet, sondern es wurden direkt die von Linux bereitgestellten Schnittstellen genutzt. Eine Beispielanwendung, wie die Schnittstelle zu verwenden ist ist in der Datei *ComTest.cpp* bereitgestellt. Die zugehörige kompilierte Anwendung ist *manual_control.exe*. Mit dieser Anwendung können die Ausgängen des Arduinos gesteuert werden. Die Verwendung ist folgendermaßen:

```
// ./manual_control $register $value  
// for example
```

`./manual_control 2 50`

`// Sets the windmill to the speed of 50 (of max 255).`

Darauf aufbauen wurde ein *HardwareController* geschrieben, der eine leichtere Steuerung des Modells erlaubt. Erfasst die einzelnen Bereiche des Netzes zu logischen Gruppen zusammen die dann gemeinsam gesteuert werden können.

Smart Grid Modell

In Anlehnung an den bereits bestehenden Tagszeiten-Simulationsmodus wurde eine neue Simulation entwickelt, die ebenfalls einen Tagesablauf, mit Tageszeit abhängigem Verbrauch und Energieerzeugung, sodass zu verschiedenen Tagszeiten mal ein Überschuss an erneuerbaren Energien besteht und mal die Versorgung ausschließlich durch die konventionelle Energie gedeckt ist. Abhängig von der Energieverteilung innerhalb des Netz werden die LEDs geschaltet, um den Zustand zu visualisieren.

Die Uhrzeit kann von außen, aus Softwaresicht, eingestellt werden. Genauso kann das berechnen des Zustands für die nächste Stunde angestoßen werden, sodass sich das Modell von außen steuern lässt.

Darauf aufbauend wurde das SGM in einen Zustandsautomaten gekapselt, dieser verwaltet lediglich die zwei Zustände Simulation oder Manuelle-Steuerung. Im Simulationsmodus, wird in einem regelmäßigen Zeitintervall der Zustand für die nächste Stunde des Modells berechnet und eingestellt. Im manuellen Modus, wird die Zeit festgehalten und das Modell verweilt in diesem Zustand. Von außen kann jedoch die Uhrzeit manuell geändert werden, wodurch sich der Zustand des Modells ändert.

OPC UA Server

Für die Ergänzung des SGM wurde auf die Bibliothek *FreeOpcUa* <https://github.com/FreeOpcUa/freeopcua> zurückgegriffen. Dies ist eine unter LGPL stehende Software, mit der sowohl OPC UA Server als auch Clients geschrieben werden könne. Die Bibliothek wurde als Git-Submodule eingebunden.

Es werden auch Code-Beispiele z.B. https://github.com/FreeOpcUa/freeopcua/blob/master/src/examples/example_server.cpp für einen Server bereitgestellt. Aufbauen auf diesem Beispiel wurde der OPC UA Server für das SGM implementiert.

Der OPC UA Server soll es ermöglichen, den Simulationszustand zu simulieren und gleichzeitig Verbraucher im Modell ein und ausschalten können. Zur Steuerung stellt der OPC UA Server daher drei Steuerungs-Variablen zur Verfügung.

sim_mode_enabled Über diese Variable vom Typ `boolean` ist es möglich zwischen den Modi Simulation und manueller Steuerung zu wählen.

time Über diese Variable vom Typ `int` lässt sich eine Uhrzeit zwischen 0 und 23 Uhr einstellen.

producing Über diese Variable vom Typ `boolean` kann im Modell gesteuert werden, ob ein zusätzlicher Verbraucher (zum Beispiel eine Industrieanlage) eingeschaltet oder ausgeschaltet werden soll

Darüber hinaus gibt der OPC UA Server auch den Zustand der Simulation nach außen frei. Diese veröffentlichten Variablen können jedoch nicht manipuliert werden.

excess_power Im Netz verfügbare (überschüssige) erneuerbare Energie.

power_production Insgesamt bereitgestellte Energie, sowohl erneuerbar als auch konventionell.

renewable_pwer Produzierte erneuerbare Energie.

sun_power Durch den Photovoltaikpark erzeugte Energie.

used_power Kommulierte Energie aller Verbraucher.

wind_power Druch das Windrad erzeugte Energie.

DisplayName	BrowseName	NodeId
Root	0:Root	i=84
Objects	0:Objects	i=85
Server	0:Server	i=2253
smart_grid_model	2:smart_grid_model	ns=2;i=2001
controls	2:controls	ns=2;i=2009
producing	2:producing	ns=2;i=2011
sim_mode_enabled	2:sim_mode_enabled	ns=2;i=2012
time	2:time	ns=2;i=2010
state	2:state	ns=2;i=2002
excess_power	2:excess_power	ns=2;i=2007
power_production	2:power_production	ns=2;i=2005
renewable_power	2:renewable_power	ns=2;i=2008
sun-Power	2:sun-Power	ns=2;i=2003
used_power	2:used_power	ns=2;i=2006
wind_power	2:wind_power	ns=2;i=2004
Types	0:Types	i=86
Views	0:Views	i=87

Abbildung 1: OPC UA Server SGM Variablen

Installation

Für die Entwicklung der Software kann jeglicher Linux-Desktop verwendet werden. Für Debian basierte Linux-Distributionen ist die Installation am einfachsten.

Die zu dem Projekt gehörende Quellcode befindet sich im CVH GitLab unter: <https://gitlab.cvh-server.de/aco/smart-grid-modell>.

Zur Entwicklung werden die folgenden Tools benötigt:

- git
- gcc
- make
- cmake

Für das Testen des OPC Server bietet sich darüber hinaus noch das Tool *opcua-client* an. Dieses kann über die Python Paketverwaltung pip mit dem Befehl `pip install opcua-client` installiert werden.

Das Projekt kann mit dem folgenden Befehl heruntergeladen werden.

`git clone --recurse-submodules https://gitlab.cvh-server.de/aco/smart-grid-modell`

Für die Installation wurde das Skript `install.sh` bereitgestellt. Sind alle nötigen Abhängigkeiten installiert wird hiermit das gesamte Projekt bebaut. Auf dem Raspberry-Pi kann das Kompilieren einige Minuten dauern.

Der Grund dafür ist, dass die FreeOpcUa Bibliothek mitgebaut werden muss. Die Abhängigkeiten, die für die Bibliothek installiert werden müssen, sind in der Datei `debian.soft` der FreeOpcUa Bibliothek aufgelistet.

Die Schritte die das Installations Skript übernimmt sind die folgenden. Als erstes wird das Verzeichnis `build` angelegt und in dieses gewechselt. Darin wird der Befehl `cmake ..` ausgeführt. Damit werden die GNU Makefiles generiert, mit denen das Projekt kompiliert werden kann. Das Kompilieren kann dann mit dem Befehl `make` angestoßen werden. Beschleunigt werden kann der Vorgang, indem mit der Option `-j2` die Anzahl der Threads angegeben werden, mit denen parallel kompiliert wird.

Beim manuellen kompilieren ist darauf zu achten, dass die ausführbaren Dateien in dem Verzeichnis `build/libs/freeopcua/bin/` abgelegt werden. Das Installations-Skript kopiert die `smart_grid.exe` zurück in das `build` Verzeichnis.

Ausführen

Gestartet werden kann der OPC Server indem die ausführbare Datei gestartet wird. Beim beenden des Servers, werden die Ausgänge des Modells nicht wieder zurück gesetzt. Daher leuchten die LEDs weiter und auch das Windrad dreht sich weiter.

Zum ausschalten der LEDs können zwei Skripte benutzt werden die auf das oben genannte Tool `manual control` zurückgreifen. Damit können entweder alle Komponenten ausgeschaltet werden, oder nur das Windrad.

- `windmill_off.sh`
- `all_off.sh`

Alternativ kann das Skript `start_server.sh` verwendet werden. Mit dem Skript wird nach dem dem Beenden, durch das Drücken der Taste `q` alle Ausgänge wieder ausgeschaltet.

Der OPC Server ist anschließend unter der lokalen IP Adresse (vergeben über DHCP) oder wenn möglich über den Hostnamen `raspberrypi` und dem Port 4840.

Um sich per ssh mit dem Raspberry-Pi zu verbinden werden die Zugangsdaten Benutzername: `pi` und Passwort: `pi` benötigt.

In dem OPC UA client können dann zum Beispiel die Daten betrachtet werden. So wie in Abbildung 2 dargestellt.

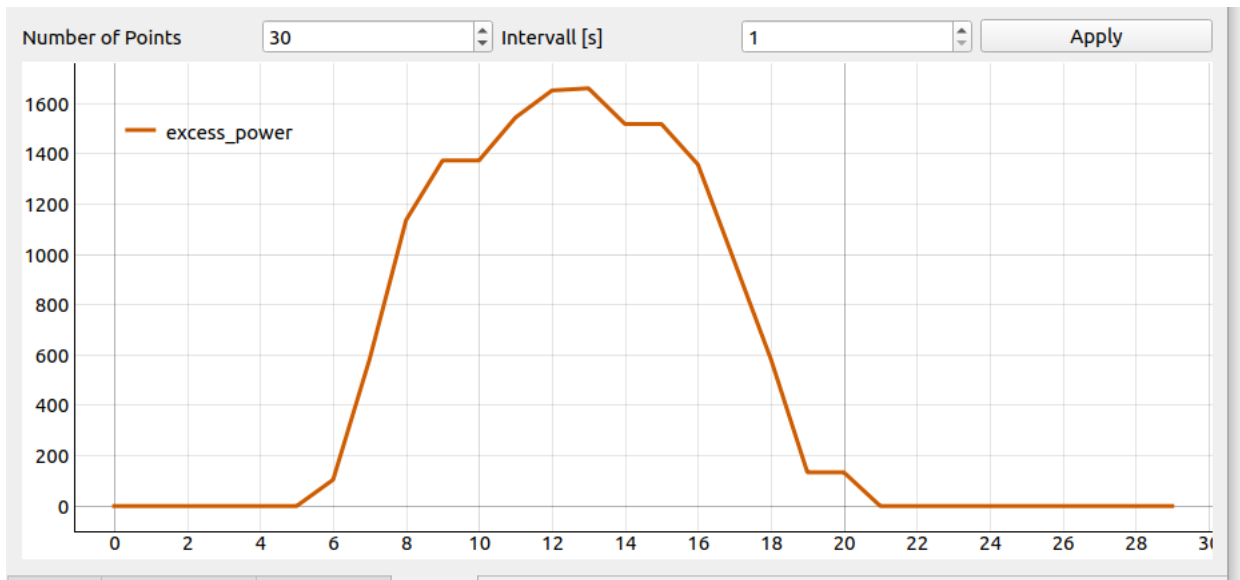


Abbildung 2: Überschüssige Energie im Zeitverlauf

Ressourcen

- <https://gitlab.cvh-server.de/aco/smart-grid-modell>
- <https://www.raspberrypi.org/>
- <https://github.com/FreeOpcUa/freeopcua>
- <https://cmake.org/>

Copyright © 2020 Armin Co

Lizenz: CC-by-sa (Version 3.0) oder GNU GPL (Version 3 oder höher)

Sie können diesen Text einschließlich \LaTeX -Quelltext herunterladen unter:

<https://gitlab.cvh-server.de/aco/smart-grid-modell>