Hydrogen System

Time box 1 – 15-03-2011

This report covers Time-box 1 in the realization phase of the Hydrogen subproject carried out by Team 4. The Hydrogen subproject is a part of the overall Energy Hub Project.

The project is a mandatory part of the 4th semester at the Electronic Design Engineer education at AU-IBT.

The Project has been supervised by Klaus Kolle and Morten Jakobsen both teachers at the Electronic Design Engineering program.

Lasse Lykkegaard

Dennis Thomsen

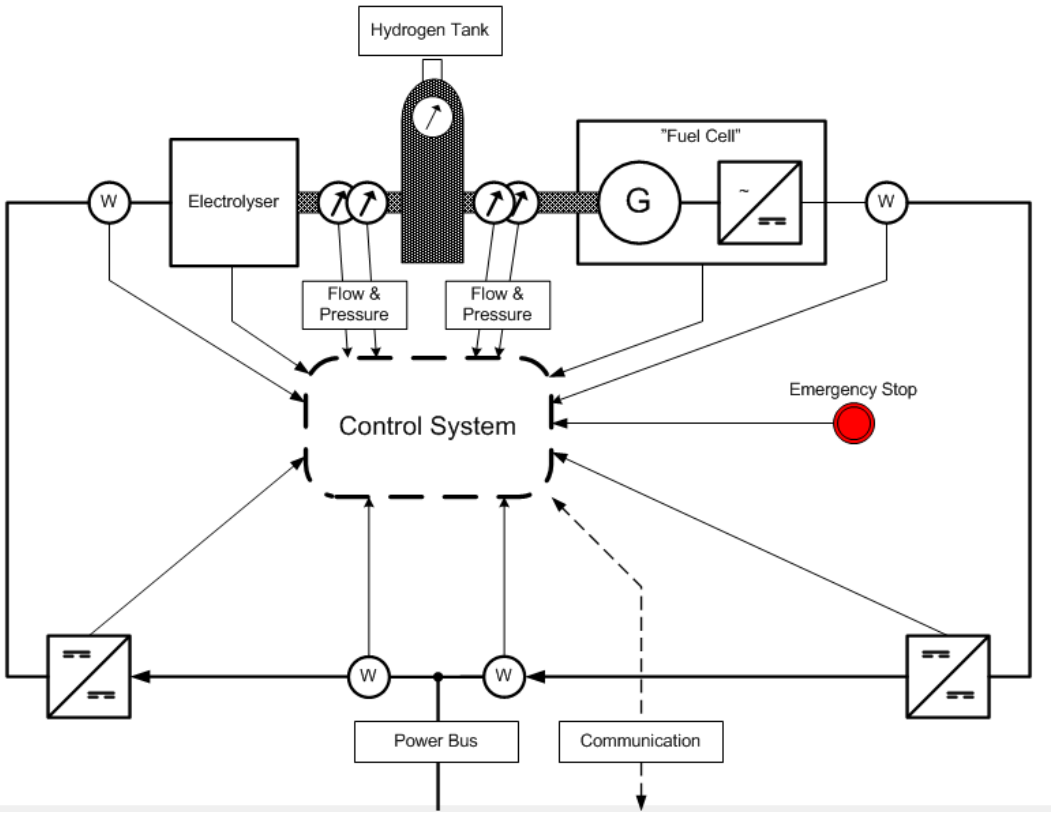
Knud Baastrup

Content

System Overview

Lasse Lykkegaard

Below figure show the System Overview created during the PRO3 Launch project



Change log

This section describes the overall project adaptations done in PRO4 compared to PRO3.

The altering specifications and the work in the realization phase have revealed some none coherent specifications which have been altered.

The functions getSensorData() does not need the arguments supplied, when working in the data model. These data is only necessary working on the bus...

The interface between the User-space application and the Data model was previously separated with two interfaces, one for the sensor class and one for the production system class. These two have been merged together.

The State Machine diagram of the Fuel Cell and Electrolyzer should be altered to include a super state as emergency shutdown. At least the emergency shutdown should be made a blocking condition. Super state is preferred

DEVELOPMENT STRATEGY

Knud Baastrup

The requirements defined during launch phase have been grouped into a number of time boxes as shown in Table 1.

The grouping has been done in order to satisfy an iterative deployment plan where the iterations results in product versions that can be deployed at the customer.

The grouping has furthermore ensured that the more time demanding HW components is initiated in the early time boxes to allow test on early mocked up prototypes prior to the final PCB versions.

The Fuel-cell is deployed late in the realization as it reuses most of the HW components used to support the electrolyzer.

Development and verification planning is done as part of the initial planning for each time box, which makes the planning more flexible in terms of who does what. EMC test facilities will however be scheduled in good time prior to the EMC verification.

Table 1 H2 requirements grouped into time boxes

ID	Requirements	Time box
G1.1	The H2 subsystem shall be able to produce hydrogen from electrical power using electrolysis and should as well store it in a hydrogen tank.	1
G1.2	The H2 subsystem shall be able to produce energy as electrical power using hydrogen fuel cell or any other device that can produce electrical power from hydrogen.	4
G1.3	The H2 subsystem shall be physical connected to a power-bus with a nominal voltage of 48 VDC. The voltage should be within 48 ±2 VDC with a maximum superimposed AC voltage of 2 V_{pp} in accordance with the Bus Voltage Specification. The power-bus will both deliver the power to be stored as hydrogen and consume the power produced using hydrogen fuel cell.	3
G1.4	The electrolyzer and Fuel cell shall not consume or deliver more than 3KW or 62.5A.	1, 4
G1.5	The H2 subsystem shall monitor input/output power-bus voltage and input/output power-bus current and notify the Hub and other subscribed observers on changes in voltage or current level.	2
G1.6	The H2 subsystem shall be able to deliver current efficiency data to other systems on request.	3
G1.7	The H2 subsystem should be able to deliver efficiency data measured over certain duration of time that at least should cover up to 24 hours.	2, 3
E1.1	It shall at any time be possible to start H2 production on request from Hub when all of the following conditions are fulfilled: o Defined delay between stop and start has passed. o Electrolyzer has not been manually stopped. o No power production ongoing. o Other (the system must be in ready state) Hub shall be notified once H2 production has started.	3

E1.2	It shall at any time be possible to stop H2 production on request from Hub either immediately or when defined delay between start and stop has passed. Hub shall be notified once H2 production has stopped.	3
E1.3	The H2 subsystem shall be able to stop H2 production and notify Hub if one of the following conditions takes place: o Sufficient power is no longer available for efficient H2 production. o Electrolyzer has been manually stopped due to malfunction	3
E1.4	It shall at any time be possible to manually stop H2 production immediately and notify Hub in case of emergency or for maintenance/test purpose.	3
E1.5	The H2 subsystem shall be able to produce hydrogen when sufficient power in terms of 1500W is available for the Electrolyzer.	1
E1.6	The H2 subsystem must be able to convert the power-bus voltage into the required input voltage for the given Electrolyzer. Alternatively the Electrolyzer should use the voltage given by the power bus despite an expected lower efficiency.	1
E1.7	Delay between start and stop of H2 production as well as delay between stop and start of H2 production should be configurable via local configuration interface/web interface to H2 subsystem.	3
E1.8	The H2 subsystem should monitor the input voltage and input current to the Electrolyzer to allow calculation of the converter efficiency for the converter required between power-bus and Electrolyzer.	2
E1.9	The H2 subsystem shall monitor the hydrogen flow and hydrogen pressure delivered by the Electrolyzer to allow calculation of the Electrolyzer efficiency.	2
F1.1	It shall at any time be possible to start power production on request from Hub when all of the following conditions are fulfilled: o Defined delay between stop and start has passed. o Fuel cell has not been manually stopped. o No hydrogen production ongoing. Hub shall be notified once power production has started.	4
F1.2	It shall at any time be possible to stop power production on request from Hub either immediately or when defined delay between start and stop has passed. Hub shall be notified once power production has stopped.	4
F1.3	The H2 subsystem shall be able to stop power production and notify Hub if one of the following conditions takes place: o Sufficient hydrogen is no longer available for efficient H2 production. o Fuel cell has been manually stopped.	4
F1.4	It shall at any time be possible to manually stop power production immediately and notify Hub in case of emergency or for maintenance/test purpose.	4
F1.5	The H2 subsystem shall be able to produce power when sufficient hydrogen can be delivered from hydrogen storage.	4
F1.6	The H2 subsystem shall be able to convert the output voltage from the given Fuel cell into the required input voltage for the power-bus.	4
F1.7	Delay between start and stop of power production as well as delay between stop and start of power production should be configurable via local configuration interface/web interface to H2 subsystem.	4
F1.8	The H2 subsystem shall monitor the input flow and input pressure to the Fuel cell to allow calculation of the Fuel cell efficiency	4

F1.9	The H2 subsystem should monitor the output current and output voltage delivered by the Fuel cell to allow calculation of the efficiency for the converter required between Fuel-cell and power-bus.	4
F1.10	The H2 subsystem shall be able to deliver a Green-score on request from Hub.	3

Extended project scope

Knud Baastrup

The PRO4 project has an extended project scope compared to the PRO3 Launch project. Figure 1 shows the PRO3 Launch project scope in the black dashed square along with the extensions that includes requirements to support WEB2, IDE1, EMB2(HW) and EMC1.

The extended project scope will be fulfilled in parallel with the original PRO3 scope and included in the time boxes along with the PRO3 requirements and will as well be visible in the deployment plan.

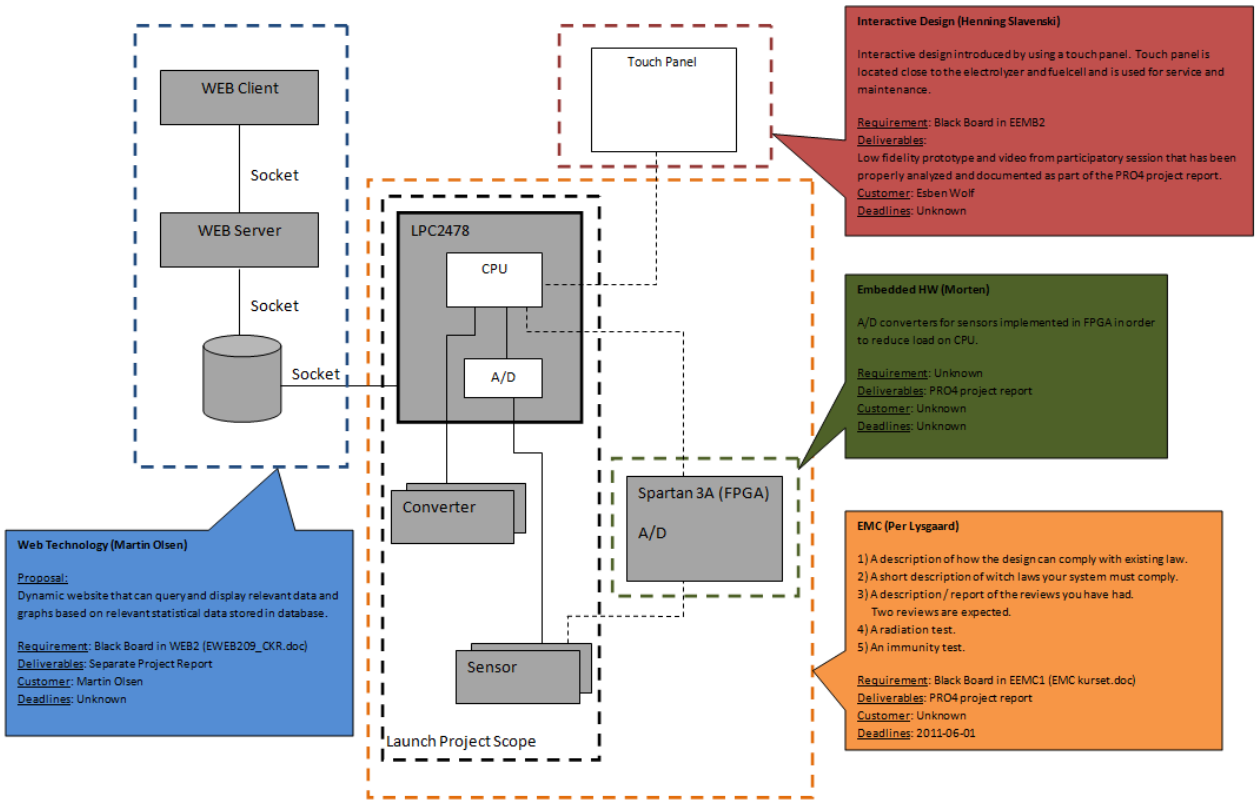


Figure 1: Overall Block Diagram with the extended PR04 project scope

DEPLOYMENT PLAN

Knud Baastrup

Each time box realization will result in a product version that gradually is extended with more and more functionality. The product version will be deployed at the customer premises and presented for the relevant stakeholders in order to get early feed-back from the customer and secure a continued commitment from the stakeholders.

Table 2 shows the functionality delivered a long with the product versions and as well the expected deployment date.

Table 2 Deployment Plan

Product Version	Functionality	Deployment Date
1	Start/Stop Electrolyzer and implement Database: The Electrolyzer can be started and stopped from a software application running in user-space on a LPC2478 Development kit. The power bus is simulated by using the XPR 60-100 power supply. Database design is settled and implemented on lene-lasse.dk Participatory session with low fidelity prototype completed. The product version consists of the following artifacts: • Electrolyzer (final version) • Converter/relay-switch (prototype 1) • LPC2478 Development kit (final version) • Driver for converter/relay-switch (final version) • H2 application (version 1) • Database design (final version) • Low fidelity prototype • Report documenting the design	15-03-2011
2	Measure efficiency of the Electrolyzer and implement dynamic web: It is possible to measure the efficiency of the Electrolyzer by measuring the energy consumed versus energy produced by the Electrolyzer. This requires the ability to measure voltage, current, flow and pressure. The measured data can be read-out from the H2 application sw. Dynamic web site can show measured data (data are simulated using scripts). The product version consists of the following artifacts: • Current sensor (prototype 1) • Voltage sensor (prototype 1) • Flow sensor (prototype 1) • Pressure sensor (prototype 1) • Driver for A/D converter (final version) • H2 application (version 2) • Dynamic web site (version 1) • Report documenting the design	29-03-2011

3	Communicate with Hub and connect to power bus:	12-04-2011
4	Introduce Fuel-cell and FPGA for (alternative) A/D conversion	03-05-2011
5	EMC verification:	17-05-2011

TIME BOX 1 SPECIFICATION

Knud Baastrup

This time box covers the below requirements, which must be realized in order to meet the functionality specified for product version 1 in the deployment plan in Table 2.

ID	Requirements (as specified during PRO3 Launch)
G1.1	The H2 subsystem shall be able to produce hydrogen from electrical power using electrolysis and should as well store it in a hydrogen tank.
G1.4	The electrolyzer and Fuel cell shall not consume or deliver more than 3KW or 62.5A.
E1.5	The H2 subsystem shall be able to produce hydrogen when sufficient power in terms of 1500W is available for the Electrolyzer.
E1.6	The H2 subsystem must be able to convert the power-bus voltage into the required input voltage for the given Electrolyzer. Alternatively the Electrolyzer should use the voltage given by the power bus despite an expected lower efficiency.

The time box will as well include the database design required for the WEB2 extended scope and some preparations for the next time box

Development Plan

All

Timebox 1	W8	W9	Week 10							Week 11						
Task			M	T	W	T	F	S	S	M	T	W	T	F	S	S
Converter/Relay circuit (PT1)					L	L	L	L	L			L	L	L	L	L
Flow Sensor (Identify and Order)					D	D	D	D	D			D	D	D	D	D
Pressure Sensor (Identify and Order)					D	D	D	D	D			D	D	D	D	D
Voltage Senser (Identify & Order)		K	K											K	K	K
Current Senser (Identify & Order)		K	K											K	K	K
Embedded computer platform	A	A														
Driver to converter		K		K	K	K	K	K	K	K	K	K	K			
Data model for converter		K		K	K	K	K	K	K	K	K	K	K			
Remaining data model																
Webserver configuration	L	L														
Database Design			D	D						D	D					
Implement Database																
Low fidelity prototype			L	L						L	L					

L: Lasse Lykkegaard
K: Knud Baastrup
D: Dennis Thomsen
A: All

Verification Plan

Knud Baastrup

A set of Product Acceptance test that match each requirement were prepared during the Launch project. The following 4 tests were defined for the requirements included in this timebox. Some remarks have been added for some of the tests to suggest some

alternative measures that are needed due to some functionality still missing, but planned for later timebox realizations.

The Product Acceptance tests shown in Table 3 will be performed prior to the deployment of timebox 1.

Table 3 Product Acceptance test relevant for timebox 1 deployment

ID	Acceptance Tests (as specified during PRO3 Launch)	Remarks
G1.1	The test can be performed using the procedure described below: 1. Start the hydrogen production 2. Read the flow and pressure sensors 3. Alternating/increasing value insures ongoing hydrogen production. 4. Increasing pressure in hydrogen tank confirms the process of hydrogen storage.	There is currently no sensor support, but it is possible to see the hydrogen flow in a transparent hose. Hydrogen will not be stored in a tank.
G1.4	The test can be performed using the procedure described below: 1. Measure current and voltage on output/input of the fuel cell/electrolyzer while the system is at work. 2. If the measurements are not larger than the required 3KW or 62.5 A, the requirements are fulfilled.	For now only tested for electrolyzer. Test with Fuelcell is done during timebox 4 verification.
E1.5	The test can be performed using the procedure described below: 1. Insure that sufficient power in terms of 1500W can be delivered from power-bus, by measuring the current/voltage on the input. Make readings from the flow and pressure sensors, if the values alternate/increase – the hydrogen production is started.	There is currently no sensor support, but it is possible to see the hydrogen flow in a transparent hose.
E1.6	The test can be performed using the procedure described below: 1. Perform multiple conversance tests in the lab environment in order to insure the both stability and efficiency of the PowerBusToElectrolyzerConverter. 2. Once the lab test is performed, the system can be started. Measurements of voltage must be taken from Power-bus output, as well as measurements of the input of the electrolyzer. 3. The measurements from power-bus output must meet the required input value of the electrolyzer.	There is currently no funding to in source a 48VDC to 60VDC Converter. We test will just verify that the interconnected relay does not impact the powerbus voltage of 48VDC.

Test cases to verify the functionality in more detail will be developed in parallel with the implementation and the test results will be documented.

CONVERTER/RELAY-SWITCH

Start and stop of the electrolyzer is managed by enabling and disabling the converter located between the power bus and the electrolyzer.

The electrolyzer is able to run direct from the power bus voltage, so the initial version of the converter will just act as a relay and not do any conversion of the power bus voltage.

Using the power bus voltage of just 48 VDC for the electrolyzer, will impact the amount of hydrogen produced. This is acceptable here in the early stages of the project as it despite the lower efficiency still allow the electrolyzer to operate and in that way allow an early deployment without investing too much money or time into a more expensive converter solution.

Class Design

Knud Baastrup

The converter is defined by its Converter class in the data model as already specified during launch. The Converter class is an aggregation of the ProductionSystem class that defines the electrolyzer and the fuel cell. See Figure 2.

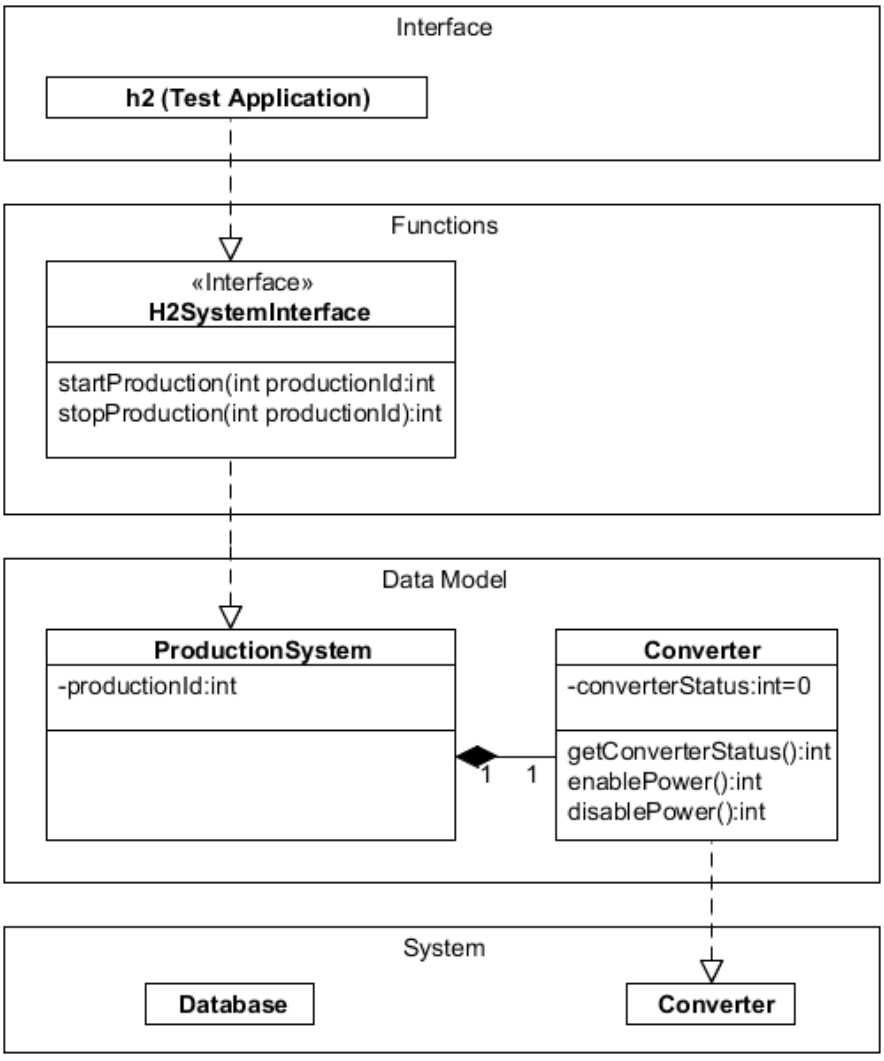


Figure 2: Class design for H2 System

The data model will for now just support what is needed in order to fulfill the functions for starting and stopping the electrolyzer. The interface that is defined by the class H2SystemInterface, is for that reason limited to the two functions startProduction() and stopProduction().

A simple h2 test application has been created in order to test the functions provided via the H2SystemInterface class. This test application will later be replaced with an interface towards the Hub.

Class Implementation

Knud Baastrup

The actual implementation of the classes can be explored via below HTML pages that have been created using Doxygen.

http://www.lene-lasse.dk/doxygen/h2_app/index.html

Use Case Scenarios

Knud Baastrup

The sequence diagram below show how the currently supported functions have been implemented.

The h2 test application instantiate the H2SystemInterface object that support the required functions to start and stop the hydrogen production. The H2SystemInteface object instantiates the required part of the data model, i.e. an object representing an electrolyzer and an object representing a fuel cell. The converter objects are created in the constructor of respectively the electrolyzer and fuel cell as the converter is an aggregation to the ProductionSystem. The converter is likewise deleted in the destructor of the ProductionSystem.

A request to start hydrogen production can be fulfilled unless the fuel cell is already running, which require a check prior to enabling the hydrogen production as shown in Figure 3.

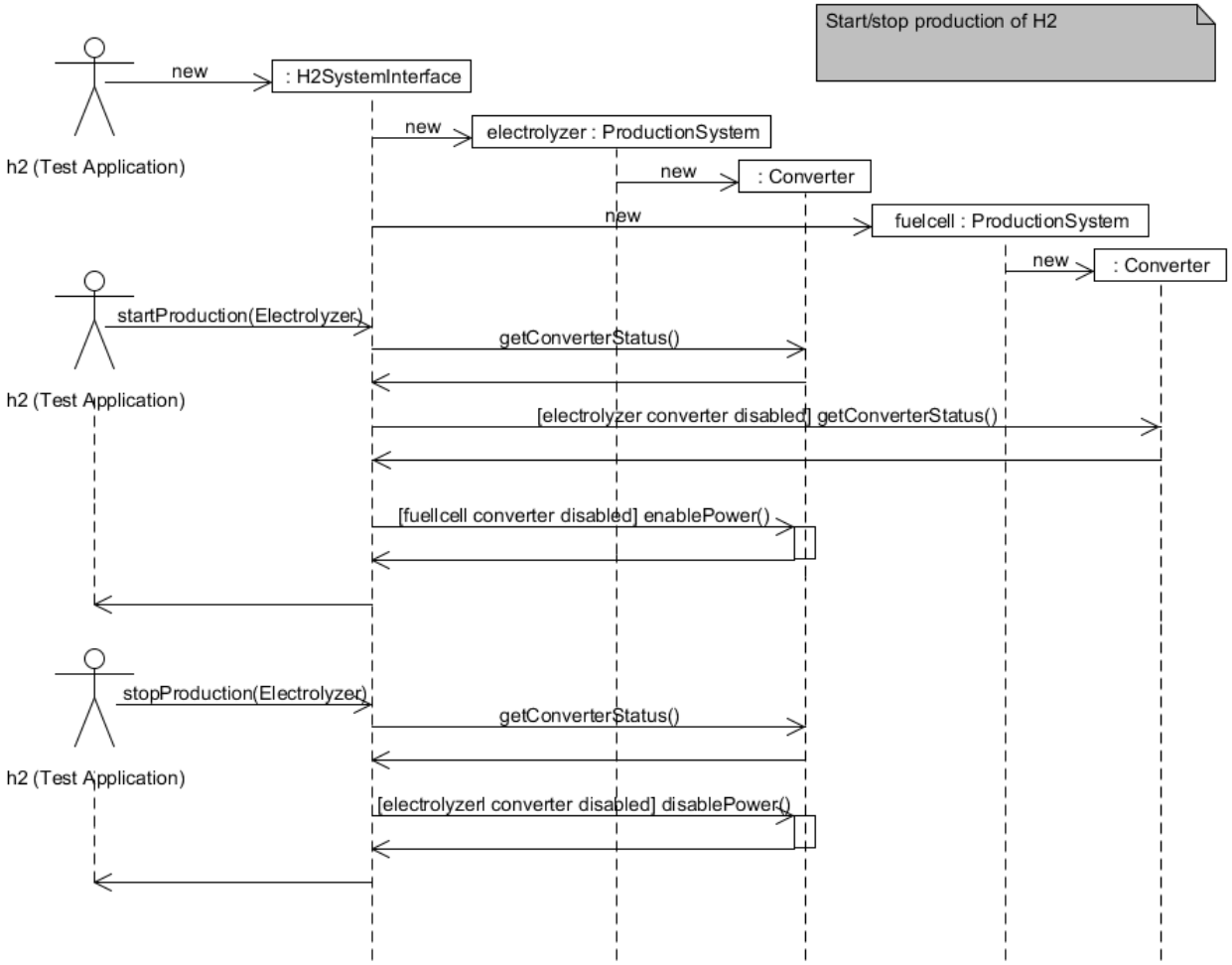


Figure 3: Sequence diagram for start/stop production of H2

Driver support in kernel

Knud Baastrup

The converter enables and disables the power via a driver running in kernel space with the required functionality to set and clear a GPIO pin on the LPC2478 CPU. Figure 4 shows how the converter is processing the request to enable and disable power.

The converter aggregated by the electrolyzer has been hard coded to use the driver named /dev/converter0 and the converter aggregated by the fuel cell has been hard coded to use driver named /dev/converter1. The 2 drivers share the same major number that can be allocated either statically or dynamically depending on the definition:

```
#define CONVERTER_MAJOR 249      // Set to 0 for dynamic allocation
```

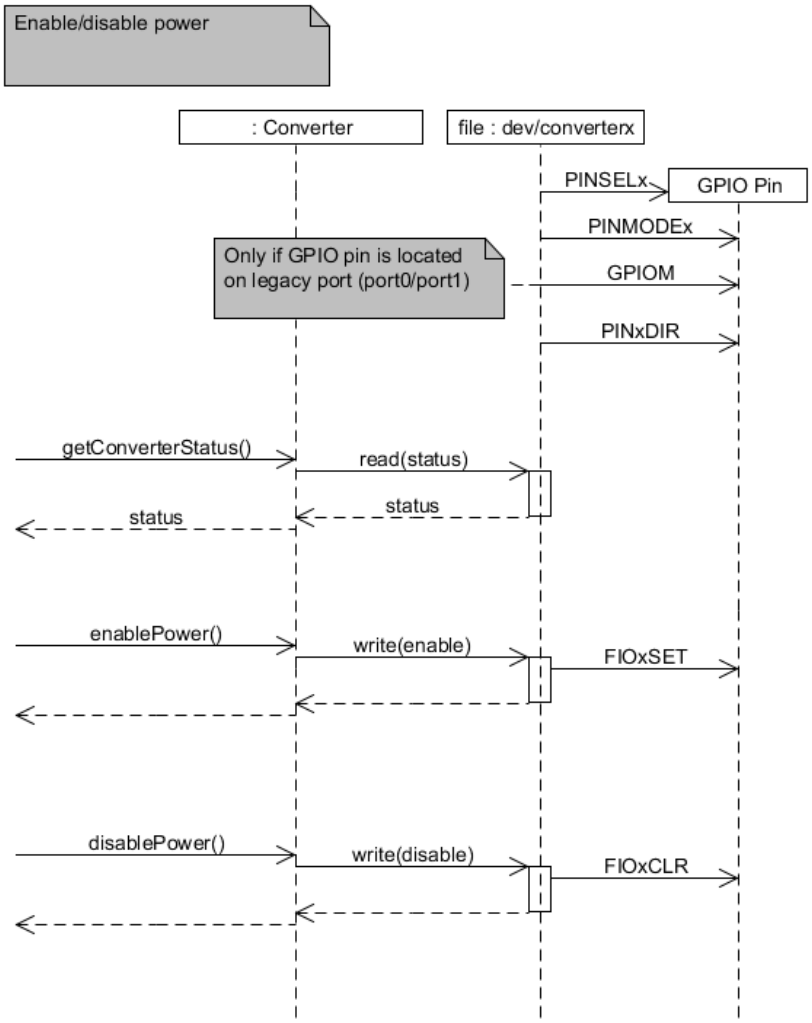


Figure 4: Sequence diagram for enable/disable power of converter

The GPIO pin to be configured can be set in the header file by using below definitions:

```
#define FIOPORT_DEV0 2          // GPIO Port for device 1
#define PORTPIN_DEV0 21         // GPIO Port Pin for device 1

#define FIOPORT_DEV1 2          // GPIO Port for device 2
#define PORTPIN_DEV1 23         // GPIO Port Pin for device 2
```

It is possible to set a LED (P2.10) to follow the GPIO pin by using below definitions:

```
#define USE_LED 0               // Set P2.10 LED to follow port.
                                // Use 0 to follow pin on device 0.
                                // Use 1 to follow pin on device 1.
                                // Use -1 to disable LED support.
```

It has been chosen to save the status of the converter in the device structure allocated for each device, which then eliminate the need for a reading the GPIO pin when supporting a read requests for the converter status. Alternatively, we could have read out the actually GPIO pin for each read request and saved the space required to store the converter status in the device structure.

A semaphore has been introduced for the write request to ensure that only one device at a time can update the converter status saved in the device structure. It is to some extend overkill as we will not have more than one process that actually makes use of this device, but who knows what the future will bring.

The actual implementation of the driver can be explored via below HTML pages that have been created using Doxygen.

http://www.lene-lasse.dk/doxygen/h2_drv_converter/index.html

HW Design

Lasse Lykkegaard

Initially we projected a solution with an optocoupler separating the EA-board and the power control circuit, but since the two shares supply, the galvanic separation did not improve the system. The next idea was to implement the circuit with an unity gain operational amplifier to exploit its high input resistance and low output resistance. The high current from the relay coil and voltage spikes was still to be kept out of the op amp and a mos fet should conduct the current. The op amp was overkill to make a simple switch.

A very simple solution with an transistor and a base resistor was the next solution, small signal transistors would be pressed to their limit conduction the coil current and the spikes occurring when switching. A power transistor has a significantly lower hfe/ß of 20-30 resulting in a large current pulled from the EA-board.

A MOSFET is voltage controlled (charge) no current is taken from the circuit controlling the MOSFET, our problem was the low control voltage of 3.3VDC, special MOSFET with a low threshold voltage solved that problem (though our supplier blow us off by not ordering a component with equal specs as the wanted one which was out of stock) since we couldn't wait a week a new solution had to made.

A standard Common-Emitter circuit has the disadvantage that it inverts the signal, so if the EA-board was disconnected the relay would turn on the connected equipment. For safety reasons this was not wanted. If the output was moved to the Emitter it would become an Emitter follower, which well follows the base voltage, there by nothing gained.

Eventually we decided us for cascading two transistors in what is known as a Darlington bridge – the two transistors is available merged into one discrete component but, this unfortunately have some other unwanted specifications higher voltage drop base-emitter. Therefore it was built by two transistors. I order to make sure the base of the power transistor is not floating a pull down 47 kΩ resistor is added

With this setup the GPIO current is lower than 0.5 mA. A diode over the relay coil is a must to minimize the spikes from the coil, which would have killed the power transistor even though it is heavily over sized. Simulated the peaks without the diode is close to 14 kV. We chose a fast diode a Schottky.

The 10 kΩ pull-down resistor at the FIO pin ensures that the pin is low during reset of LPC2478 board where its initialized reset value is input.

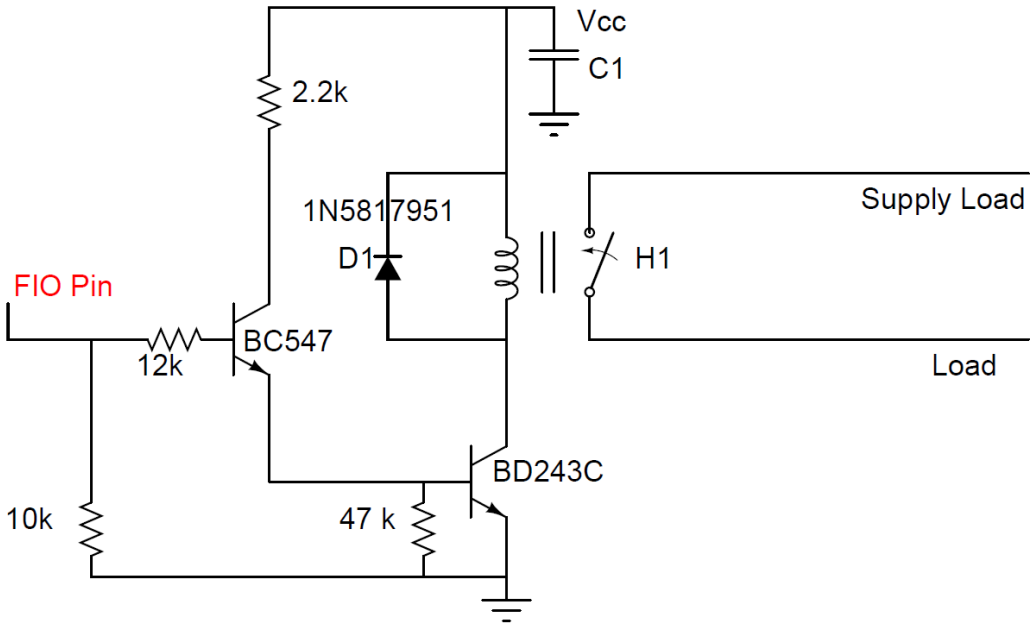


Figure 5: Digram of the Relay circuit

Verification

The verification for timebox 1 includes a number of functional tests and as well the product acceptance tests required prior to deployment.

Functional tests

The following test cases were executed to verify the functionality delivered in timebox 1

Test Case ID	**1.1**	
Description	Verify that it is possible to start and stop hydrogen production via h2SystemInterface using h2 test application.	
Preconditions	Electrolyzer NOT started Fuelcell NOT started	
Command/Action/Steps	**Expected Result/Verification**	**PASSED**
`$ h2 1 0`	Relay should be activated	OK
`$ h2 2 0`	Relay should be deactivated	OK

Test Case ID	**1.2**	
Description	Verify that it is not possible to start hydrogen production if Fuellcell is already started.	
Preconditions	Electrolyzer NOT started Fuelcell started	
Command/Action/Steps	**Expected Result/Verification**	**PASSED**
`$ h2 1 0`	Relay should NOT be activated. The error code `-PROD_START_REJECTED` should be returned.	OK

Test Case ID	1.3	
Description	Verify that it is rejected to start the electrolyzer if it has already been started.	
Preconditions	Electrolyzer started Fuelcell NOT started	
Command/Action/Steps	**Expected Result/Verification**	**PASSED**
`$ h2 1 0`	Relay should continue to be activated. The error code `-PROD_ALREADY_STARTED` should be returned.	OK

Test Case ID	1.4	
Description	Verify that an attempt to start production of hydrogen without any device driver loaded is properly rejected and do not change the state of the system.	
Preconditions	Electrolyzer NOT started Fuelcell started	
Command/Action/Steps	**Expected Result/Verification**	**PASSED**
`$ h2 1 0`	The error code `-OPEN_DEV_FAILED` should be returned.	OK
`$ lib/modules/converter_load`	`Converter module has been loaded`	OK
`$ h2 1 0`	Relay should be activated	OK

Test Case ID	1.5	
Description	Verify that a reset of the LPC2478 Development kit do not cause any flicker on the relay.	
Preconditions	LPC2478 development kit up running with device drivers loaded Electrolyzer NOT started Fuelcell NOT started	
Command/Action/Steps	**Expected Result/Verification**	**PASSED**
Activate the reset button	The delay should be kept deactivated. No flicker.	OK
`$ lib/modules/converter_load`	The delay should be kept deactivated. No flicker.	OK

Product Acceptance tests

The product acceptance tests were passed with the remarks given as part of the timebox specification in Table 3.

DATABASE

In WEB1 last semester we built a static webpage for our system which this semester needed to be made interactive though the use of php and mysql.

On this webpage there should be displayed among other things the current data for electrolyzer or the generator, the average data over a time period and either a table or graph for seeing changes over time.

For this we need a database of previously recorded data.

Database structure

Dennis Thomsen

The first idea for the database structure was to have a table for the Production System containing all data that was close to being static, meaning that most of it wouldn't be changed after initialization. We did not see any value in being able to see the old values of these variables, therefore there was no history. The current status of the system i.e. if the system was on or off, was also in the table, because for that data we did want a record.

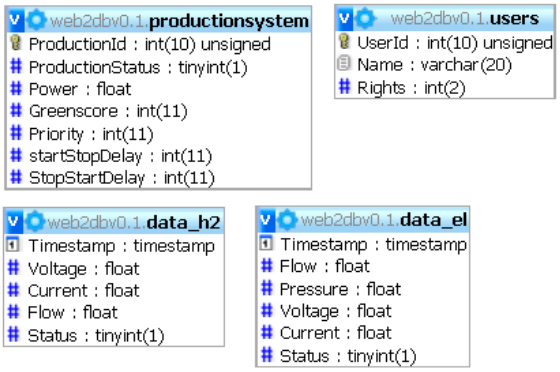


Figure 6: First version of the Database structure

The idea behind storing the status of the system is that we can use it to calculate how much time the machine was actually running, when averaging data over a time period.

In the design was the "users" table, which the idea behind was to allow some users admin rights. The two tables "data_h2" and "data_el" where meant for all the data we needed a history of for the two production systems "voltage", "current", "flow" ,"pressure" and "status" with the "timestamp" for when the data was recorded being the primary key.

Voltage, current, flow and pressure is all the base values we need for the calculations, we could of cause have stored the calculated values as the effect in and out of the systems, but in case we needed to change the formals for any of the calculation, we thought it best to keep the results out of the database, also calculated vales can be placed in some quick look up tables for quick reference.

As the two systems don't have exactly the same variables, and also what count's as inputs for one system, is outputs for the other, it was separated into two tables.

Moving on with the design there was a desire to separate the data from each sensor type into its own table and then do a many-to-one link with the production system, with each table having a composite key made up of the sensor_id and prod_id, as seen in Figure 7.

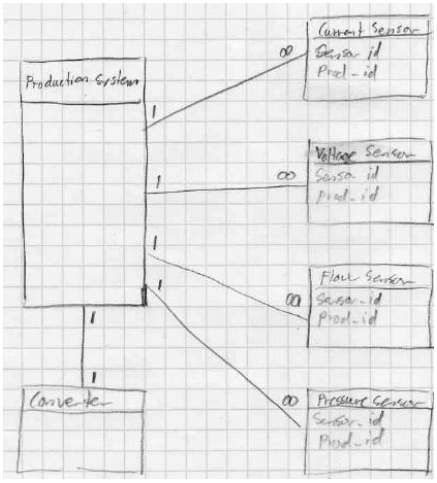


Figure 7: First draft of a database structure with sensor data split into separate tables

There are many benefits to this approach, it is easier to add on additional sensors, and adding raw data for specific sensors, that is updates, also extracting only the values needed for certain efficiency calculation will be easier and it is still possible to group by day, week and month, though this will require some joints, on the downside the complexity of the queries for fetching data will be slightly higher.

In order to eliminate redundancies in a database and ensure its integrity, the structure goes though a normalization process, using a set of rules called normal forms, where the third form is the highest we use.
For the structure of a database to be third normal form (3NF) compliant it has to follow the requirements for 3NF, and also live up the first normal form (1NF) and the second normal form (2NF).

The requirements for which can be summarize with
"Requiring existence of "the key" ensures that the table is in 1NF; requiring that non-key attributes be dependent on "the whole key" ensures 2NF; further requiring that non-key attributes be dependent on "nothing but the key" ensures 3NF[1]."

What this means is that, for 1NF to hold, each column must contain only a single value, and there mustn't be repeating groups of similar data. 2NF means that any data in non-key columns of the table, that don't depend on the key, should be separated into its own tables. And finally if changing a value in one column requires change another value, then the structure violated 3NF

FINAL VERSION
Using this knowledge the Final version of the database was built, seen in Figure 8, where we tried to keep the advantages of splitting the system down to the sensors but at the same time cut down on the number of tables, thus making it third normal form compliant.

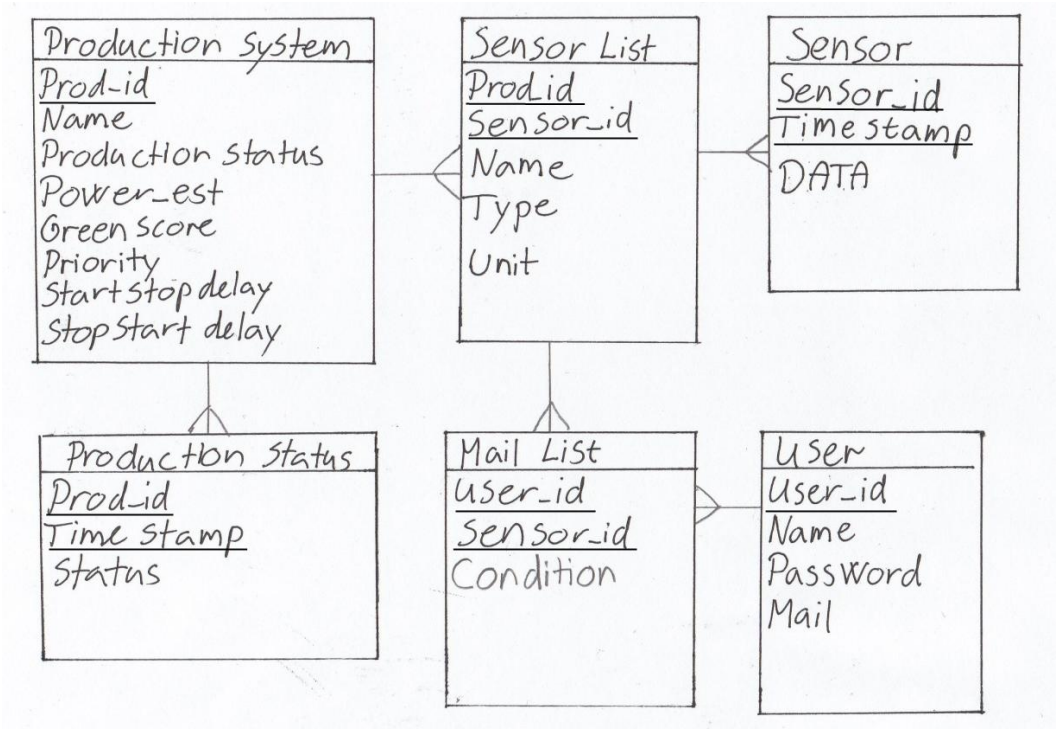


Figure 8: Final database structure, Primary keys are underlined with black

[1] Taken from http://en.wikipedia.org/wiki/Third_normal_form

The handling of the sensors has been split into two tables called “sensor_list” and “sensor”. The table, “sensor_list”, is identified with a composite key of “production_id” and “sensor_id” where the “production_id” part of the key refer to the production system for which the sensor belongs. The “name” column can be “Ampere_in” or maybe the part number of the device used for the measuring. The “type” column is meant to be the same for all ampere meters e.g. “Ampere” and the “unit” column would then be [A]. The “unit” can be fetched when building graphs or tables for the website, so there isn’t some hard coding that have to be fixed when adding a new type of sensor.

All data from the sensors is then stored in a table called “sensor” with composite key of “Sensor_id” and “Timestamp”, this table is linked to “Sensor_List “with a many-to-one relationship as each sensor can have lots of data reading but each data reading only belongs to one sensor, likewise “Sensor_List” is linked to “Production_System” by the same logic that each subsystem can have many sensors but each sensor only have one “Production_System”.

The “Production_System” table is still almost the same, as in the first version, the only change, is the addition of the “Name” column, linked to it besides the “Sensor_List” table, is a table called “Production_status”, that holds the history of the “ProductionStatus” column. This is placed there in the design as we still want that data, but didn’t think it belonged in the “sensor” table.

Rather than using the “user” table for an admin system, as first intended, it has been change in the current version to support a mail list where users can subscribe to different sensors, so they would get some automated email from the system, regarding the sensors they are subscribing to. For the structure of the database this means we need to link “Sensor_List” and “User” together, with a many-to-many relationship as a user can subscribe to many sensors, and many users can be subscribed to one sensor. This many-to-many relationship between “Sensor_List” and “User”, is resolved by an intermediate table that has a many-to-one relationship with both “Sensor_List” and “User”, this table was named “Mail_List”, and have a composite key of “User_id” and “Sensor_id”. To this table was added a column “Condition” (which may be split into more columns in the future) the idea behind which where that the user could choose only to receive an email when say the pressure in line to generator went over 2.4 Bar.

Web server configuration

Lasse Lykkegaard

When multiple people are working with server generated HTML and SQL-database data, you cannot test files locally, you need to have a development server with sufficient programs installed. Bren or our own local server could be equipped with MySQL, php and apache2, but then programming from outside the school had not been possible. Unlike last year it's the schools does not supply a server to the project.

We therefore chose to place our web-development server on a internet server. Physically located somewhere in Germany. Accessible from everywhere.

The server is running Ubuntu version 8. with MySQL ver 5.

There set up a group user for the database and the HTML pages have been moved to www.lene-lasse.dk which is the temporary domain for our web page. Its should not affect other sites and databases running on the server.

Its possible to connect through FTP port 21 and administrate the SQL-DATABASE with phpmyadmin or login through SSH and manage though a SQL-client.

The server has like Bren git installed but this does not work with FTP uploads

LOW FIDELITY PROTOTYPE

Lasse Lykkegaard

In this part of the project, the prototype is presented very early in the process. Before any line of code or piece of hardware is ordered. By that we have full flexibility and no limits in changing the product according to the feedback.

The prototype is a low-fidelity prototype of a User-interface-system. We have provided a suggestion to the different UI-screens on paper – If nice-looking designed UI-interface is presented, the focus is changed to design and layout instead of functionality.

Mr. Esben Wolf from the Alexandra Institute has consented to help us with the service maintenance interface for the hydrogen system. We had a short meeting with Mr. Wolf to align our common requirements and ideas. Mr. Wolf stressed the fact that is should be easy and intuitively made. Since many of the service technicians could have dyslexia and/or were foreigners.

Mr. Wolf pointed out some major key objects of the GUI[2], which will try to carry out.

1. Intuitive menu system
2. interactive structure with pressure sensitive function
3. overview of the system
4. Zoom able overview
5. Alarms at front page
6. Cable free interaction
7. Guides at difficult places

We took Mr. Wolf's ideas and demands and tried to fetch them into a prototype of the interaction screen. Mr. Wolf did agree to participate in a participatory design session. This session was videotaped and is to be showed and analyzed further. At this session we showed Mr. Wolf our GUI-ideas with a low fidelity prototype of the system. Mr. Wolf was able to interfere and alter whatever he wanted since everything was implemented in cardboard paper.

The screens contained no system unique texts - all gibberish to avoid focus at nonwanted areas for example the actual text. The session's goal was to determine the navigation, buttons, contents.

[2] Graphical User Interface

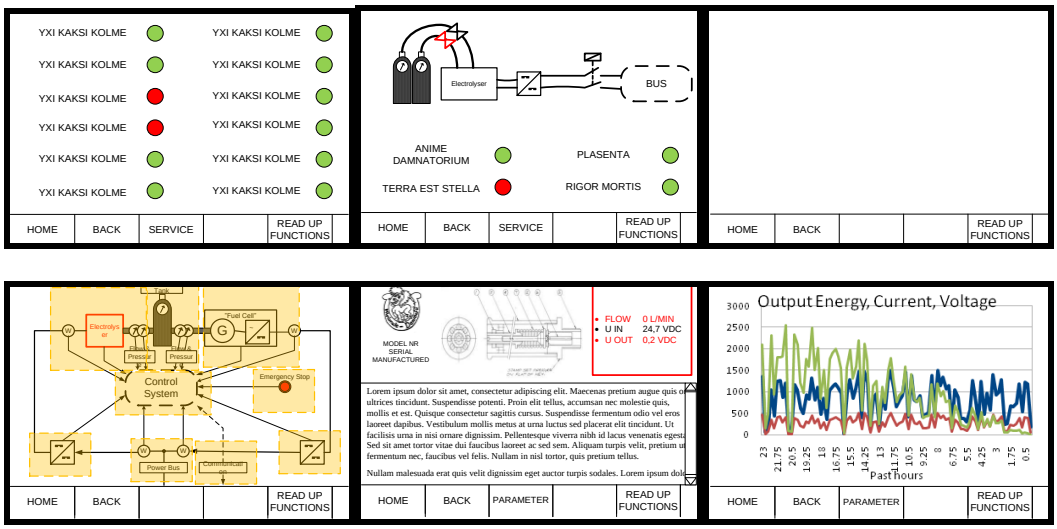


Figure 9 - Screen suggestions

The screens presented to Mr. Esben Wolf, all open for changes and in a 1:1 scale about 10 inches

A blank screen was supplied as well as extra “buttons” in case of changes during the session. A button with video guide was added on request from Mr. Wolf.

OTHER TASK

Other task includes primarily tasks carried out in preparation for late time box development.

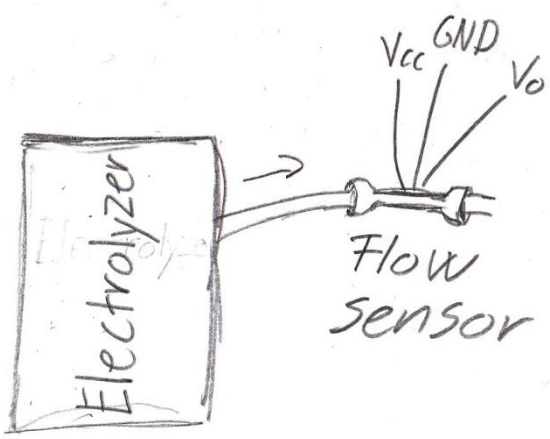
Identify/Order Flow/Pressure sensor

Dennis Thomsen

Flow sensor

Dennis Thomsen

A Flow sensor is needed both for the Electrolyser and Generator subsystems, such a flow is inserted into the gas line, and calculate the flow based on two measurements of pressure of the gas, before and after a narrowing inside the sensor.



Minimal Requirements for Flow Sensors:

Analogue output
Electrolyser subsystem: 4.3 L/m
1 bar
Generator subsystem: 30 L/m
2 bar

The reason we would like a analogue output is that, it make it all far easier for us to handle than a digital output, as we can just use a A/D converter to handle the inputs and we can normalized all the input signals to be in the range 0-3.3V, by building small circuits.

During testing performed last semester we found the flow from the electrolyser to be about 4.3 L/m as the gas isn't being compressed, the sensor just needs to work under Atmospheric pressure that is about 1 bar.

The flow sensor for the Generator subsystem, is a another matter, as the operating pressure for the motor is 2 bar the sensor should be able to handle at least that, and based on information from reports by teams that have previously work on the generator, and also testing done by ourselves (as didn't have a device for measuring flow that we could insert into the system between the tank and the motor, we had to make due, as the there is faucet on the motor that can adjust the flow rate into it, we had to try and find the maximum, this was done by starting the motor and adjusting the pressure of the line to 2 bars, and the disconnect the line from where it was attached to the motor, and then measured how long time it took for it to displaced 500 ml of water, this time was measured to be about 1 sec, which translates to 30L/m), the maximum flow rate for system seems to lie around 30 L/m.

Sensors to be used are:

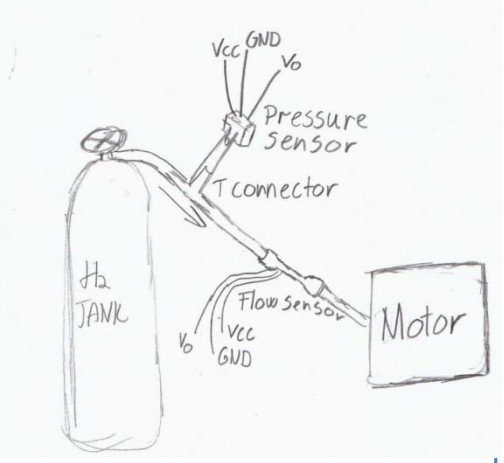
FTAL020NU (SensorTechnics)
Operating Range: 0 – 20 LPM
Vcc: 8-15 V DC Normal: 10 V
Maximum Pressure: 50 PSI (3.4 bar)
Minimum Output Voltage: 0.95 – 1.05 V DC Normal: 1 V
Maximum Output Voltage: 5 V

D6F50A5000 (Omron)
Operating Range: 0 – 50 LPM
Vcc: 10.8-26.4 V DC
Maximum Pressure: 500 kPa (5 bar)
Minimum Output Voltage: 0 – 1 V
Maximum Output Voltage: 5 – 5.7 V

The sensor from SensorTechnics is a left over from an old project involving the electrolyser, while the sensor from Omron has just been ordered home. We have of course tested the SensorTechnics to make sure it still works, which seems to be the case, though we don't have any way checking the calibration of the sensor, as we don't have any source with a known flow rate.

Pressure sensor

Dennis Thomsen



A pressure sensor is needed for our generator subsystem as the motor runs on compressed hydrogen gas, this sensor will be insert into the gas line using a t-connecter.

Minimal Requirements for Pressure Sensor to Generator subsystem:
Analogue output
Generator subsystem: 2 bar

The reasoning behind using a sensor with an analogue output is the same as for the flow sensors and we need a sensor that can handle at least 2 bar, as that is the operating pressure for the motor in the generator subsystem.

The following Sensor has been ordered

MPX4250AP (Freescale)
Vcc: 4.85 - 5.35 V DC Normal 5.1 V
Operating Pressure Range: 20 to 250kPa (0.2-2,5bar)
Minimum Output Voltage: 0.133 - 0.264V DC Normal: 0.204 V
Maximum Output Voltage: 4.826 - 4.966 V DC Normal: 4.896 V

This is a relatively cheap pressure sensor and it lives up to our requirements and it is also capable of handling up to 1000 kPa (10 bar) for a short time.

Preliminary Circuit design

My current thought about the boards for the sensors are similar

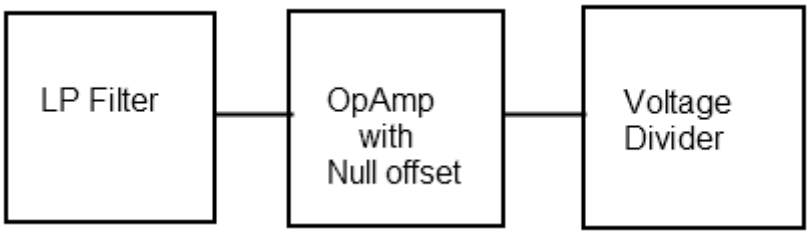
First part should be a Low pass filter, for obvious reasons, this is placed first so noise doesn't affect the other parts.

A opamp with null offset is then needed to place, the voltage point for the sensors that indicate either 0 flow or pressure at 0 V, this is done so we get the full range from 0-3.3 V into the ADC's giving so a higher degree of accuracy.

For the MPX4250AP (the pressure sensor) we need only adjust the level about 200mV so I am thinking that this can be done with a Non-inverting Opamp with Null offset of course.

Will for the two flow sensor a differential opamp and Null offset might be needed, where 0.5V and 1V are subtracted respectively for the D6F50A5000 and FTAL020NU.

And finally a voltage divider to bring the maximum voltage down to 3.3 V



Fuel Cell load test

Lasse Lykkegaard

Pressurized hydrogen is needed for the "Fuel Cell" to run, therefore the test is postponed until more bottled H_2 arrives. The electrolyzer cannot pressurize hydrogen, therefore we are depending on external supply of hydrogen.

The test is fully projected and is to clarify the actual power production of the fuel cell. The calorific value of hydrogen is higher than regularly gasoline measured in J/kg. Since hydrogen is a gas it can be compressed unlike gasoline. So the calorific value measured as J/L can vary according to the pressure of the Hydrogen. But at 1 atm. the calorific

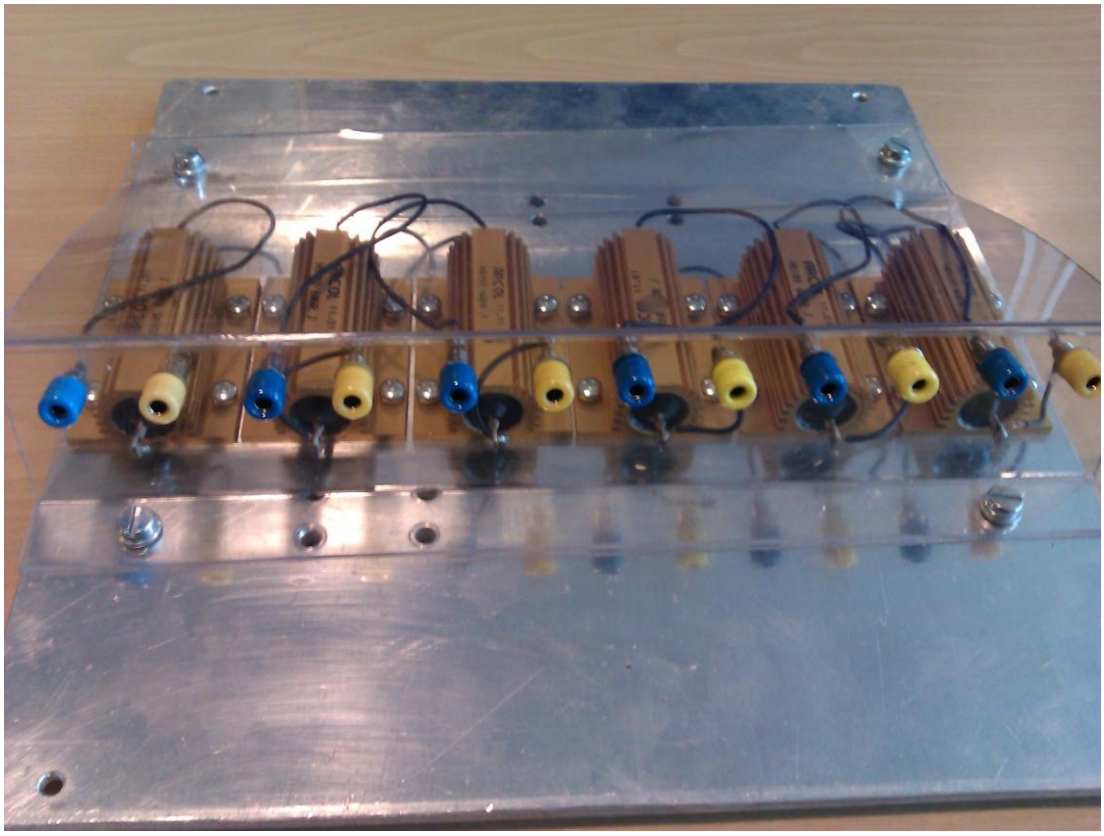
value of one liter of hydrogen is significantly lower than one liter of gasoline, about one third.

The carburetor of a four stroke engine mixes air and gasoline, due the vacuum produced by the piston movements the mixture is sucked into the combustion chamber, and ignited. The rebuilt version running on hydrogen approximately the same is happening. Hydrogen under pressure is mixed with oxygen from the air producing oxyhydrogen. The mixture is directed in the combustion chamber and ignited. We do not know the pressure and cannot calculate the actual amount of energy our fuel cell can produce.

A low tech test is rigged up, without any triac regulated devices – a triac regulator will result in a non uniform current load. We don't know the specifications of the converter yet, thereby the test is based on pure ohmic loads.

We expect the unit to produce less energy than a similar one running on gasoline. 500W halogen spots shall produce the coarse test and are accompanied by 100W power resistors. The power resistors though cannot extract more than 78W due to interval of the E12 resistor scale and the fixed voltage of 230VAC. The 78W is thereby the resolution of the test.

Below picture shows the load resistors to be used for the load test of the Fuel cell.



REFERENCES

- http://www.eudp.net
- Physics for Scientists and Engineers with modern physics by F.G.T.
- FTAL020NU Datasheet
- D6F50A500 Datasheet
- MPX4250AP Datasheet