Sabatier Process Automation

Colorado State University – Pueblo

Faculty Advisor
Jude L. DePalma, Ph.D., Professor

Team Members
Charles Bryan, Alexander Klein, Dustin Clasby

April 6, 2015
Table of Contents

Acknowledgement .................................................................................................................................. iii

Introduction ........................................................................................................................................ 1

Executive Summary ......................................................................................................................... 1

Problem Statement .......................................................................................................................... 1

Operating Environment ................................................................................................................... 2

Intended User(s) and Intended Use(s) ............................................................................................ 2

Assumptions .................................................................................................................................... 2

Limitations ....................................................................................................................................... 3

Expected End Product and Other Deliverables .............................................................................. 3

Project Records and Activities ......................................................................................................... 3

  Background ................................................................................................................................. 3

  System Design ............................................................................................................................ 4

  PLC ............................................................................................................................................... 4

  HMI ............................................................................................................................................... 6

  Pump Design ............................................................................................................................ 7

Testing and Results .......................................................................................................................... 16

Resources and schedules .................................................................................................................. 16

  Financial requirement ................................................................................................................ 16

  Risk Management ...................................................................................................................... 16

  Project milestone and evaluation criteria .................................................................................... 17

Client information ........................................................................................................................... 18

  Faculty advisor information ...................................................................................................... 18

  Student team information ........................................................................................................ 18

Closure Material ............................................................................................................................... 19

  Project Evaluation ...................................................................................................................... 19

  Recommendations for Additional Work .................................................................................... 19

References ....................................................................................................................................... 20

Appendix A – PLC .......................................................................................................................... 21

Appendix B – Pump .......................................................................................................................... 22

Appendix C – Project Budget .......................................................................................................... 24
Acknowledgement

Many thanks must go to the faculty that has helped create the Sabatier Process Machine, specifically Huyesin Sarper, Ph.D., P.E., Paul Wallace, Laboratory Coordinator, and David Dillon, Ph.D., Associate Professor. Also, credit must be given to all of the students who have worked on the machine over the past few years.
Introduction

Executive Summary

The Sabatier Process Machine at Colorado State University-Pueblo exists as a proof of concept for a final Sabatier machine that could be sent to Mars. The machine uses a chemical reaction to convert hydrogen and carbon dioxide to water and methane. The water is then split into oxygen and hydrogen, with the hydrogen being recycled back into the reaction. The operation of the machine kept by the University previously required human supervision and interaction, but the process is now fully automated.

This project delivers four objects. The first object is the Automated Sabatier and Electrolysis machine. The Sabatier and electrolysis will function automatically without human monitoring. A PLC is used to automate the entire system. The second object is the Pump Controller. The pump will automatically adjust its speed based on the level of water in the tank. It is controlled by an Arduino-type microcontroller receiving an input from a sensor detecting the water level. The third object is the Computer Display to monitor the system. The display will show the current status of the system, as well as the total amount of methane that has been produced. The display is controlled by the Arduino-type microcontroller as well. The last object is the technical manual. The manual will be written so anyone who intends to work on the Sabatier has all of the information that they need. It will include a safety section, one-line electrical diagrams, a hard copy of the PLC and Arduino programs built for our project, and product data sheets. It is intended that this project be expanded or added to in future senior projects.

Problem Statement

Currently, there is a functioning Sabatier Process Machine located at Colorado State University-Pueblo. It was built as part of the Colorado Space Grant as a proof of concept. A final Sabatier machine could be sent to Mars ahead of any manned space mission. This machine uses a chemical reaction to convert hydrogen and carbon dioxide to water and methane. The water is then split into oxygen and hydrogen, with the hydrogen being recycled back into the reaction. The chemical reaction is given in the following equation,

\[ 4H_2 + CO_2 \rightarrow CH_4 + 2H_2O \]

The machine can produce jet fuel for the return journey and water and oxygen to sustain the astronauts during the mission. Operation of the machine kept by the university currently requires human supervision and interaction.

The Sabatier process machine needed constant monitoring. The start-up and shut down processes alone took one and four hour(s), respectively. To start the machine, the operator needed to turn on switches to heat the reaction and power on the temperature controller. The operator needed to open the gas valves for the process. During these periods, the reactor required constant monitoring to make sure that it did not overheat and to make sure that gas was passing through the reactor during these times to prevent it from burning up. When the
process was running, the speed of the pump that moved the hydrogen from the electrolysis tank had to be adjusted frequently to keep the water level in the tank from getting too high or too low. The water level fluctuated because the production of hydrogen is sporadic due to the varying temperature of the water. If the hydrogen production was too high, water would spill out of the tank, causing a dangerous chemical spill. If the hydrogen production was too low, water was sucked into the reactor, destroying it. As a proof of concept, this operation was unacceptable. The machine had to be made to run automatically, with no human supervision.

**Operating Environment**

The intended operating environment is the Martian surface. On the surface of Mars the Sabatier would be exposed to extreme cold temperatures, as well as an incredibly dusty and windy environment. As it is right now, however, the reactor is not required to function in that situation. Its current environment is in a room with low humidity, no wind, and no extreme temperatures. It is not intended to be dropped or thrown, and due to its weight there is no reason to believe it will be knocked off of the countertop, or otherwise disturbed. Likewise, for the purposes of testing and demonstration the machine will not operate in a high carbon dioxide environment, but be fed carbon dioxide gas from tanks.

**Intended User(s) and Intended Use(s)**

The intended users are trained students and faculty members. Everyone that is to operate this machine needs to be briefed on safety, as well as the functionality of the machine. It both uses and produces highly explosive gases, and it must be taken seriously.

The intended use of this machine is to produce methane that can be used as fuel on the surface of mars. Additionally, the oxygen and water that are by-products could be collected for breathing, burning, and drinking.

**Assumptions**

There are few assumptions with this project. One assumption is that the reactants that are being used are pure; however, since the CO₂ is coming from an outside source, AirGas, purity cannot be guaranteed. The H₂ is produced via electrolysis, but there is the possibility that it can get contaminated by H₂O(g). A chamber was built on top of the H₂ collection to condense the evaporated water, but that device lies outside the scope of this project.

The other assumption is that this project will not be the final state fit for space exploration. Because of that fact, the project will be left in a state where it can easily be altered and added to. Additionally, all of the technical specifications and detailed descriptions of our work will be available for all future students or faculty working with the device.
Limitations

There are some constraints that are not strict, but important to adhere to. The first is the intended operating environment, as mentioned above. The final version of the machine will need to operate in this environment. The second constraint is energy consumption. The final version of this machine will need to run off of battery or solar power. It will need to consume as little energy as necessary. It will not be possible to optimize the energy usage of the machine at this stage, but it will be considered in the design of the automation system. The final constraint is one of space. The final version will need to be as compact as possible. The system that will operate on Mars will probably need to be very large, but it must maximize the production per cubic centimeter. Again, this optimization will not be appropriate at this stage, but will be considered in the design.

Expected End Product and Other Deliverables

There are four main deliverables for this project. The first deliverable is the Automated Sabatier and Electrolysis. The Sabatier and electrolysis will function automatically without constant monitoring. A PLC will be used to automate the entire system. It will control several solenoid valves, as well as the thermal couple. The second deliverable is the Automatic Pump Control. The pump will automatically adjust its speed based on the level of water in the tank. It will be controlled by an Arduino-type microcontroller receiving an input from a sonic sensor detecting the water level. The third deliverable is the computer display to monitor the system. The display will show the current status of the system, as well as the total amount of methane that has been produced. The display will be controlled by the Arduino-type microcontroller, as well. The last deliverable is the Technical manual. The manual will be written so anyone who intends to work on the Sabatier has all of the information that they need. It will include a safety section, one-line electrical diagrams, a hard copy of the PLC and Arduino programs built for our project, and product data sheets.

A concise list of the deliverables and delivery date is given in Table 1.

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated Sabatier &amp; Electrolysis</td>
<td>5/1/14</td>
</tr>
<tr>
<td>Automatic Pump Control</td>
<td>5/1/14</td>
</tr>
<tr>
<td>Technical Manual</td>
<td>5/1/14</td>
</tr>
<tr>
<td>Computer Display and Monitoring System</td>
<td>5/1/14</td>
</tr>
</tbody>
</table>

Project Records and Activities

Background

The current machine is started by manually starting the electrolysis tank. This involves turning on the stirring motor and turning on the power to the electrolysis electrodes. The tank is observed until hydrogen has been produced. Meanwhile, the heaters for the reaction
chamber are turned on and monitored until it reaches nominal reaction temperature. The pump allowing the hydrogen to enter the reaction chamber is powered after the valve above the hydrogen electrolysis tank is opened. The hydrogen valve to the reaction chamber is opened. Finally, the valve to allow the carbon dioxide into the reaction chamber is opened.

After the start-up is accomplished, the electrolysis chamber must be supervised to regulate the rate of production of hydrogen. If too much hydrogen is produced, the pump speed must be increased, and if too little, decreased. When the machine is done, it must be carefully shutdown. The carbon dioxide valve must be shut. The hydrogen must be allowed to run through the reaction chamber while it cools. During this time, the electrolysis must continue to be supervised. Once the reaction chamber is cool, all valves must be closed and the electrolysis can be powered off. The entire cycle takes at a minimum, not including methane production time, which can vary, about five hours. The entire time the machine is running a person must be present to ensure proper operation.

System Design

To automate the system, a Programmable Logic Controller (PLC) and an Arduino-type microcontroller were used. The PLC functions as the primary operator for the system, turning everything on and off. The Arduino-type microcontroller automatically controls the flow rate of a DC pump based on the water level in the tank. Additionally, it also has a display linked to it showing the current progress/status of the machine. Essentially, automation comes in three main parts: system automation via PLC, automatic pump control, and a system monitoring display.

PLC

The PLC is the main control of the entire Sabatier automation system. Using ladder logic, it turns on and off each component of the Sabatier and electrolysis based on inputs from the Arduino-type controller and cascading timers.

The PLC system is made up for four main parts: a PLC, power supply, 16 port output expansion, and relay output boards. The PLC, power supply, and expansion bay are made by Koyo, and the relay output boards are made by Automation Direct. The wiring for the entire system can be seen below.
The power supply (PS) converts 120 VAC to 24 VDC at 1.3 amps. This is then supplied directly to the PLC to power the CPU and the digital logic of the PLC as well as the expansion bay. Additionally the PS provides power to the relays boards to fire the relays. Power calculations were necessary to make sure that the PS could run the entire system as shown in Table 2.

**Table 2  Power Calculations**

<table>
<thead>
<tr>
<th>Outputs/Relays</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>volts per output</td>
<td>24</td>
</tr>
<tr>
<td>amps per output</td>
<td>0.0167</td>
</tr>
<tr>
<td>power per output/relay (24 total)</td>
<td>0.4008</td>
</tr>
<tr>
<td>Total (Watts)</td>
<td>10.4208</td>
</tr>
</tbody>
</table>

**PLC**

<table>
<thead>
<tr>
<th>Volts</th>
<th>24</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amps</td>
<td>0.14</td>
<td>PS V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS Amps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total power available (Watts)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power remaining (Watts)</td>
</tr>
<tr>
<td>Total (Watts)</td>
<td>3.36</td>
<td></td>
</tr>
<tr>
<td>Total Consumed by the system (Watts)</td>
<td>14.7408</td>
<td></td>
</tr>
</tbody>
</table>
Since the PS gives us 31.2 Watts and the entire systems consumes only 14.7 Watts, there is more than enough power to run the PLC system as it stands, as well as any additions.

There are three main groups that the PLC controls: heating system, electrolysis, and the solenoid valves. Each of these three groups has been isolated to a dedicated relay output board. The exact address of each device can be found in the I/O list in Appendix A.

The PLC was programmed with ladder logic using the CLICK software provided by Koyo.

The program consists of a main program and 2 subroutines. The main program initiates the startup and shut down sequences contained in the corresponding subroutines. Both start up and shut down are run by cascading timers whose functionalities are nested inside of each other. The timers are based on total minutes depending on the time that it takes to complete each step of the corresponding process.

To start the machine, press the ON switch on the touch screen. The PLC takes this input and begins the startup subroutine. After the start up subroutine is finished, it continues running to maintain the bits that have been turned on. When the machine must be turned off, the OFF button is pressed, starting the OFF subroutine. This does not shut down the start routine; it alters it by opening normally closed internal bits built into the startup that keep each component on once the startup is finished. These bits are opened by a sequence of timers in the same way that the startup routine functions. The full program can be found in the index.

HMI

The human interface, or HMI, that is used is a 4.3 inch, resistive touch display made by 4D systems. It is programmed using 4D Systems VisiGenie software to build the visual aspect of the screen in conjunction with Arduino software to control the on screen objects functionality. This means that in order for the screen to function properly, it must be connected to an Arduino or a similar microcontroller. For the purpose of this project we are using a SparkFun Redboard.
The purpose of the HMI is to initiate the startup and shut down sequences, as well as monitor the ongoing processes. As previously mentioned there are two phases to programming the screen, visual and functionality.

Using the 4D IDE multiple screens were created corresponding to the Electrolysis and the Sabatier. Additionally, there were several informational screens built in case someone does not first look at the manual. Many of the objects are programmed to display readings from the other processes being run by the Arduino-type microcontroller. This is achieved through the Arduino programming. A special library full of commands made by 4D had to be imported into the software. It takes these values and places them in the corresponding Genie object, whether that be an LED digits display or a meter. Images of each screen as well as the full Arduino code can be found in the index.

Pump Design

The pump is a Pergro Grobot Peristaltic Pump. The peristaltic pump is driven by a DC motor. The power-voltage curve is shown in Figure 5. The slope of this curve is the amperage through the motor circuit. The data for this curve is included in Appendix B.
The volume of medium pumped through the peristaltic pump depends on the voltage applied to the DC motor. Figure 6 shows the pump displacement for the voltage applied to the pump. The relationship is nearly linear, as shown by the line of best fit. The formula for this relationship is given on the graph. The data for this graph is included in Appendix B.

![Graph: Pump Voltage vs Pump Volume Displacement](image)

**Figure 6  Voltage v Displacement**

From this linear equation, it is straightforward to estimate the amount of medium pumped based on the voltage applied to the motor. This equation is the method used to estimate the hydrogen introduced into the Sabatier reactor. Since hydrogen is the limiting reactant, the amount of methane produced can also be calculated.

The electrical circuitry for the pump consists of an operational amplifier and a DC power supply. The Arduino-type microcontroller takes the ADC reference value of the converted voltage from the sensor and sends it out to the operational amplifier (Op Amp) by converting the digital signal in the Arduino-type microcontroller to a pulse width modulation signal (PWM). The PWM is as close as the Arduino-type microcontroller can get to an analog signal. PWM is generated from an 8-bit value, meaning there are $2^8$ (256) possible different values. The Arduino-type microcontroller does this by modulating the duty cycle of a square wave (a signal that switches on and off). The lower the value, the slower the duty cycle and the higher the value, the faster the duty cycle. So, a 0 value would be equal to 0 volts, and a 255 value would be equal to 5 volts.

An Op Amp is a device that receives an input of a certain voltage and outputs a higher voltage that the receiving device needs to function properly. As a reference voltage enters the Op Amp at one input, it changes the equilibrium of the Op Amp. As a result, the Op Amp tries to come back into balance by changing the voltage output to get both inputs into the Op Amp equal again. Since the resistors are at fixed value, the only way for the Op Amp to get back to equilibrium is to change the voltage output.

The Op Amp being used between the Arduino-type microcontroller and the peristaltic pump is wired as a non-inverting type. This amplifier is an OSM LM67ST-ND. The op-amp was wired for a 12/5 gain, so that it would deliver voltage on a range from 1 – 12 V as 0 – 5 V is provided from the Arduino-type microcontroller, and with enough current to run the motor. The voltage used for the op-amp was a single supply 20 volts DC from the DC power supply. The power supply can supply 3A, which is more...
than enough, as can be seen from the power-voltage curve above. The wiring diagram for the circuit is shown in Figure 7. Design guidance for the single supply op amp was given by the Texas Instruments, “Single Supply Op Amp Design Techniques” [1].

The equation that describes the op amp circuit can be derived from the voltage divider law.

\[ V_{out} = V_{in} \left( \frac{R_1}{R_1 + R_2} \right) + V_{cc} \left( \frac{R_2}{R_1 + R_2} \right) \]

The graph below demonstrates the relationship between power supplied by the Arduino-type microcontroller and the power output to the DC pump motor. Figure 8 also contains the linear best fit function. The formula for this function is displayed on the graph.
The data for this graph is provided in Appendix B. This formula relates the voltage from the Arduino-type microcontroller to the voltage provided to the pump. The relationship from the Arduino-type microcontroller output to the amount of medium pumped can now be determined, and this relationship is provided in the Arduino code.

A custom sensor had to be created for the electrolysis tank to measure the water level. Its casing is made of polypropylene to reduce any reaction the electrolysis tank may have with the sensor. A five volt source is sent to the sensor from the Arduino-type microcontroller down inside the sensor tube through a bare high resistive wire where it is then routed to ground. Another wire stripped bare is placed opposite the resistive wire in the sensor to return a voltage back to the Arduino-type microcontroller. Contact between these two wires is made by the water in the electrolysis tank. The circuit for the sensor is shown in Figure 9. The Arduino-type microcontroller reads the voltage coming from the sensor by converting the analog signal to a digital signal using a 10-bit value analog to digital converter (ADC) within the Arduino-type microcontroller. 10-bit means that the ADC can subdivide (or quantize) an analog signal into $2^{10}$ (1024) different values. In this case, 0 volts is equal to 0 and 5 volts is equal to 1023 in the Arduino mapping system.
To find the voltage that is being output to the Arduino from the sensor, the resistance of the resistive wire is separated into two parts by the height of the water. At this division, a measurement is taken measured from the bottom of the sensor. The length above the water level is called R1 (resistance 1) and the length below the water as R2 (resistance 2). A formula is used to calculate $V_{out}$ (voltage out) to the Arduino using these resistances and multiplying them by the $V_{in}$ (input voltage) as can be seen in the equation below.

$$\frac{R_2}{R_1 + R_2} \cdot V_{in} = V_{out}$$

From the equation the denominator $R_1 + R_2$ is equal to about 10 $\Omega$ (each inch represents about 1 $\Omega$), and the numerator $R_2$ is the measured distance from the bottom of the sensor. Multiplying the result of the resistors to a $V_{in}$ of 5V results in $V_{out}$ going to the Arduino. Since the Arduino must quantize the analog voltage signal to a reference value that the microcontroller can read by use of the ADC seen in the equation below.

$$\frac{1023 \cdot V_{out}}{5} = \text{ADC Reference Value to Arduino}$$

The numerator 1023 is the maximum reference value achievable multiplied by $V_{out}$, which is the voltage being read by the sensor.

In order to design a controller for the pump, it was necessary to model the pump as a transfer function. Normally a dc motor is modeled as a second order function, but this specific motor has a gearbox attached, which provides friction. The friction of the pump and gearbox dampens the motor output enough that the system could be modeled as a first order transfer function. Measurements were made of the pump approaching a reference position in a simple feedback loop. The graph of these measurements is shown in Figure 10.

Figure 10  Pump System Time to Equilibrium
For these tests, the pump was given a reference voltage of 1.675 V. This voltage correlated to water level under atmospheric pressure. The Arduino-type microcontroller reads in voltage from 0 to 5 volts and relates this value numerically on a scale of 0 – 1023. 1.675 V relates to a numeric value of 337 in the Arduino-type microcontroller. As can be seen in the graph the gain from the pump is 1. Also, from this graph the time taken to reach 63% of the final value (the time constant in exponential decay) is 735 seconds. An additional test was run and the result was 637 seconds. The additional test is included in Appendix B. The transfer function can be modeled as follows:

\[ G_{CL}(s) = \frac{K}{\tau s + 1} = \frac{1}{685s + 1} \]

This transfer function is for the closed loop function, the open loop transfer equation can be solved for and is modeled as follows:

\[ G_p(s) = \frac{1}{685s} \]

The interesting part of this open loop transfer function is that it is marginally stable. This fact explains why previous attempts to use a static pump speed did not lead to a steady state water level. The controller is a proportional integrator (PI). From previous experience, the PI controller works well for DC motors. A proportional controller may theoretically work for the system, but in reality, as the feedback loop drives the error to zero the output voltage becomes too low to drive the motor. This results in a system that will approach the target, but never reach it. An integrator controller makes up for this deficiency by summing the error until it produces enough voltage to bring the motor to the target. The form for this controller, in continuous time, is modeled as follows:

\[ D(s) = K_p + \frac{K_i}{s} \]

Putting the controller in line with the plant and running a feedback loop the closed loop transfer function is modeled as follows:

\[ \frac{K_p s + K_i}{685 s^2 + K_p s + K_i} = \frac{0.00146 K_p s + 0.00146 K_i}{s^2 + 0.00146 K_p s + 0.00146 K_i} \]

This can be compared to the known characteristic equation for a second order transfer function [2]:

\[ \frac{N(s)}{s^2 + 2\omega \zeta s + \omega^2} \]

This allows the function to be tuned based on the factors of \( \zeta \) and \( \omega \). A settling time of 700 seconds and a damping ratio of 0.707 were selected.

\[ 0.00146 K_p = 2\omega \zeta \]
\[ 0.00146 K_i = \omega^2 \]

Solving for \( K_i \) and \( K_p \) gives,

\[ K_p = 7.825, K_i = 0.0447 \]
The controller transfer function is then:

\[ D(s) = \frac{7.825s + 0.0447}{s} \]

Using the Controller found above, it is important to test the controller in simulation to ensure proper function. The first test is to check the operation in a continuous time function, and then convert that to a discrete time function. In MATLAB the transfer function was created as follows:

\[ H = \frac{7.825 s + 0.0447}{685 s^2 + 7.825 s + 0.0447} \]

Continuous-time transfer function.

The function \( H \) represents the closed loop transfer function of the whole system. Inputting a reference value will cause the system to respond by reducing the error to zero. This was simulated by using the step function in MATLAB with an input of -30, which simulates the drop in water level seen in the experiment. The graph in Figure 11 is the system response.

**Figure 11  Step Response**
As can be seen in the graph, the system settles on -30 which correlates to the water level at atmospheric pressure from the experiment start point. In Figure 12 there are two graphs. One is the feedback response of the motor. The second is the feedback response with the controller.

Next, MATLAB was used to convert the controller to a discrete time function that can be utilized in Simulink. Using Simulink, the complete transfer function is simulated for a digital controller and continuous time motor, as shown in Figure 13.

In Figure 14, the step response of the digital system is graphed.
As the simulation works the next step is to construct the actual system for testing. The controller designed above is converted into a difference equation as follows,

\[
\frac{U(z)}{E(z)} = \frac{7.825z - 7.825}{z - 1}
\]

\[
U(z)(z - 1) = E(z)(7.825z - 7.825)
\]

\[
U(z)(1 - z^{-1}) = E(z)(7.825 - 7.825z^{-1})
\]

\[
u(n) - u(n - 1) = 7.825e(n) - 7.825e(n - 1)
\]

\[
u(n) = u(n - 1) + 7.825e(n) - 7.825e(n - 1)
\]

This function is used in the Arduino code to control the pump.
Testing and Results

As stated above, the intention of this project was to automate the existing system. Successful testing was conducted that indicates that the automated system functions as well as it did manually. All systems function as expected and run for the correct time. The pump control functions as designed. No output improvement was expected, and none was indicated in the testing. No greater efficiency was expected, and none was measured. Future testing will be required to ensure system runs satisfactorily over required varying time periods.

Resources and schedules

Financial requirement

An itemized list of the components necessary for the completion of the project as well as budget for the project is included in Appendix C.

Risk Management

Risk Analysis was done on possible problems that may occur to prevent deliverables. The possible problems and solutions are shown in Table 3.
### Table 3  Risk Assessment

<table>
<thead>
<tr>
<th>Problem that may occur to prevent deliverables</th>
<th>Likelihood</th>
<th>Impact</th>
<th>Proposed Solution</th>
</tr>
</thead>
</table>
| Team member quits                            | 1          | 5      | • Ensure no knowledge/program/paper is held by a single person.  
• All work will be shared via computer folder.  
• Clearly track all member responsibilities. Tasks can be taken up upon member leaving.  
• No task is solely owned by an individual. At least two members are involved in every task. |
| Ordered equipment does not arrive (or is not what was ordered) | 5          | 2      | • Reorder equipment.  
• Use a different supplier. |
| Ordered equipment does not work                | 5          | 5      | • Use warranty to replace equipment.  
• Reorder equipment.  
• Use a different supplier. |
| Components break                              | 3          | 3      | • Buy replacement part.  
• Keep spares on hand. |
| Components work, but system function is unacceptable | 2          | 7      | • Redesign system.  
• Ensure proper simulation. |
| Damage to existing system                      | 3          | 7      | • Repair or rebuild system.  
• Use known spare parts to rebuild system. |
| Lack of expertise to complete system           | 1          | 2      | • Obtain outside consulting to complete project.  
• Research and Learn new material. |

**Project milestone and evaluation criteria**

Milestones accomplished by the dates that are mentioned in the schedule will determine if the deadlines are reached and the project is successful. On a weekly basis, reaching deadlines kept the project on time and ensured delivery of the completed project by
the delivery date. Below is Table 4, which displays the milestones for the project and dates of completion.

**Table 4  Milestones**

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing current process.</td>
<td>12-Jan</td>
<td>12-Feb</td>
</tr>
<tr>
<td>Purchasing parts.</td>
<td>12-Jan</td>
<td>19-Jan</td>
</tr>
<tr>
<td>Test the Parts for correct functionality.</td>
<td>19-Feb</td>
<td>20-Feb</td>
</tr>
<tr>
<td>Install the Parts to their respective locations.</td>
<td>20-Feb</td>
<td>26-Feb</td>
</tr>
<tr>
<td>Program the PLC.</td>
<td>1-Mar</td>
<td>31-Mar</td>
</tr>
<tr>
<td>Find the Transfer Function of the Motor for the Pump.</td>
<td>1-Mar</td>
<td>31-Mar</td>
</tr>
<tr>
<td>Develop the Control System.</td>
<td>1-Mar</td>
<td>31-Mar</td>
</tr>
<tr>
<td>Program the Arduino-type microcontroller.</td>
<td>13-Mar</td>
<td>1-Apr</td>
</tr>
<tr>
<td>Test the Pump Control.</td>
<td>13-Mar</td>
<td>1-Apr</td>
</tr>
<tr>
<td>Test the entire system.</td>
<td>6-Apr</td>
<td>11-Apr (Continuing)</td>
</tr>
<tr>
<td>Write the Manual.</td>
<td>1-Apr</td>
<td>11-Apr (Continuing)</td>
</tr>
</tbody>
</table>

**Client information**

Colorado Space Grant Consortium  
c/o Jude L. DePalma, Professor  
Department of Engineering  
Colorado State University – Pueblo  
2200 Bonforte Blvd.  
Pueblo, CO 81001

**Faculty advisor information**

Jude L. DePalma, Ph.D., Professor  
Department of Engineering  
Colorado State University – Pueblo  
2200 Bonforte Blvd.  
Pueblo, CO 81001

**Student team information**

Dustin Clasby  
dustin.clasby@gmail.com  
(719) 369-4524
In conclusion, there is a functioning Sabatier Process Machine at Colorado State University-Pueblo. It exists as a proof of concept for a final Sabatier machine that could be sent to Mars ahead of any manned space mission. The machine uses a chemical reaction to convert hydrogen and carbon dioxide to water and methane. The water is then split into oxygen and hydrogen, with the hydrogen being recycled back into the reaction. Operation of the machine kept by the University previously required human supervision and interaction, but is now fully automated. The testing confirms that this system is operational to the specifications. The risk management system was utilized as described.

This project delivers four objects. The first is the Automated Sabatier and Electrolysis machine. The Sabatier and electrolysis will function automatically without human monitoring. A PLC is used to automate the entire system. The second is the Pump Controller. The pump will automatically adjust its speed based on the level of water in the tank. It is controlled by an Arduino-type microcontroller receiving an input from a sensor detecting the water level. The third is the Computer Display to monitor the system. The display will show the current status of the system, as well as the total amount of methane that has been produced. The display is controlled by the Arduino-type microcontroller as well. The last is the technical manual. The manual will be written so anyone who intends to work on the Sabatier has all of the information that they need. It will include a safety section, one-line electrical diagrams, a hard copy of the PLC and Arduino programs built for our project, and product data sheets. It is intended that this project be expanded or added to in future senior projects.

Recommendations for Additional Work

There are four recommendations for additional work on this project. The first is to move everything electrically to the solar panel and batteries. Currently, the electrolysis tank is run off of the solar panel and the batteries. However, the intention is that all systems will run off of the battery. All of the devices that were chosen, save the solenoid valves, are powered by 24 VDC, the same as the battery. When the solar panel is expanded to provide more power, the systems can be powered through the battery without needing to order new parts.
The second recommendation is to establish a new data baseline. During this project, changes were made to the power systems, and unrelated to this project, the solar panel and batteries. It will be important for future projects to establish a new baseline for methane and water production.

The third recommendation is to re-introduce the cryo-chamber to system. This system was created to separate methane and water, and then store methane in a freezing chamber. This machine lay outside the scope of this project.

The last recommendation is to change the electrolysis chamber to reset after each run of the system. After a run of the system, hydrogen and oxygen may remain in the electrolysis chambers. This gas should be released to prevent hazards and to ensure proper function of the pump system.

References


# Appendix A – PLC

<table>
<thead>
<tr>
<th>INPUTS</th>
<th></th>
<th></th>
<th></th>
<th>OUTPUTS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE TYPE</td>
<td>DEVICE INFO</td>
<td>GROUP NAME</td>
<td>PLC ADDRESS</td>
<td>DEVICE TYPE</td>
<td>DEVICE INFO</td>
<td>GROUP NAME</td>
<td>PLC ADDRESS</td>
</tr>
<tr>
<td>Relay</td>
<td>ON relay controlled by arduino</td>
<td>Arduino</td>
<td>x001</td>
<td>Controller</td>
<td>Heat controller/Gauge</td>
<td>Reactor</td>
<td>y001</td>
</tr>
<tr>
<td>Relay</td>
<td>OFF relay controlled by arduino</td>
<td>Arduino</td>
<td>x002</td>
<td>Heater</td>
<td>Sabatier heat 1</td>
<td>Reactor</td>
<td>y002</td>
</tr>
<tr>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>x003</td>
<td>Heater</td>
<td>Sabatier heat 2</td>
<td>Reactor</td>
<td>y003</td>
</tr>
<tr>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>x004</td>
<td>Heater</td>
<td>Sabatier heat 3</td>
<td>Reactor</td>
<td>y004</td>
</tr>
<tr>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>x005</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>y005</td>
</tr>
<tr>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>x006</td>
<td>Electrolysis valve System Output Valve</td>
<td>Sol</td>
<td>y101</td>
<td></td>
</tr>
<tr>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>(SPARE)</td>
<td>x007</td>
<td>Solenoid</td>
<td>H2 Valve</td>
<td>Sol</td>
<td>y102</td>
</tr>
<tr>
<td>Solar power</td>
<td>Stars the Electrolysis Stirs the tank turns on</td>
<td>Elec</td>
<td>y109</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC Motor</td>
<td>Elec</td>
<td>y110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pump PS</td>
<td>Elec</td>
<td>y111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Starts and maintains run status of the sabatier

1. \text{test_off X002} \rightarrow \text{Call Startup}

2. \text{Start heat T3} \rightarrow \text{Enable}

   \text{Timer (ON Delay) T4}

   \text{Current Value Unit Not Retained sec}

   \text{SetPoint 10}

   \text{Current TD4}

   \text{Output}

Starts shut down sequence based on button being pressed or the time limit being reached

3. \text{test_off X002} \rightarrow \text{Call Shut Down}

4. \text{time T4}

\text{(END)}
Beginning of shutdown sequence

- turn off heat and close CO2 valve
- Starts the cooldown timer

-shuts down electrolysis
- Begins close valve delay

Subroutine Program: Shut Down

Page 2 of 5 (Total Pages)
- Closes remaining valves
- Starts control shut down delay timer

Turns off the heat control

Enable

SetPoint: 5
Current: TD8

Turn off Control_Delay: T8
Turn off Control: Y504
Timers (ON Delay): T8
Current Value: Not Retained
Unit: sec

Close_valve_delay: T7
Close_Valves: Y503

Return
Begin of Startup (1 sec delay)

- Timer(ON Delay) T1
  - Current Value Unit
    - Not Retained
    - Start_Delay
- SetPoint
  - 1
- Current
  - TD1

**Start up and opens all of the valves**

1. Enables
2. Sets

- opens the valves and turns on heat control
  - Start_Delay T1
- turns off everything associated with electrolysis
  - Elec_Shutdown Y502
- Electrolysis valve
  - Elec_Sol Y101

**Closes remaining valves**

- Electrolysis valve
  - SYS_Out_Sol Y102

**Closes remaining valves**

- Electrolysis valve
  - SYS_Out_Sol Y102

**Turns off heat and closes CO2 valve**

- Electrolysis valve
  - CO2_Sol Y104

**Turns off heat control**

- Electrolysis valve
  - heat control Y001

**Start up (1 sec delay)**

- Timer(ON Delay) T2
  - Current Value Unit
    - Not Retained
    - start_Electrolysis T2
- SetPoint
  - 2
- Current
  - TD2

**Output**

- Electrolysis valve
  - Output
Turns on electrolysis system

-start_Electrolysis \( \Rightarrow \) T2
-Elec_Shutdown \( \Rightarrow \) Y502
-power to Electrolysis \( \Rightarrow \) Y109
-DC stir motor \( \Rightarrow \) Stir \( \Rightarrow \) Y110
-power to OP amp \( \Rightarrow \) Pump power \( \Rightarrow \) Y111

Begins the delay for heat start up

-start_Electrolysis \( \Rightarrow \) T2
-Enable \( \Rightarrow \) T3
-SetPoint \( = 3 \)
-Current \( = TD3 \)
-Timer(ON Delay) \( T3 \)

Powers up the heaters based on start heat timer

-Start heat \( \Rightarrow \) T3
-Heat NC Shutdown \( \Rightarrow \) Y501

-heats: heat1 \( \Rightarrow \) Y002
-heats: heat2 \( \Rightarrow \) Y003
-heats: heat3 \( \Rightarrow \) Y004

Return
Appendix B – Pump

<table>
<thead>
<tr>
<th>Op Amp Circuit Vin v Vout</th>
<th>Vin</th>
<th>Vout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1.76</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>2.71</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.72</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>4.65</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5.65</td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>6.58</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7.57</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>8.52</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>9.5</td>
</tr>
<tr>
<td></td>
<td>4.5</td>
<td>10.45</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>11.44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pump Voltage vs Pump Displacement and Power</th>
<th>Volts</th>
<th>ml/min</th>
<th>amps</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>27</td>
<td>0.34</td>
<td>1.36</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>33</td>
<td>0.34</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>41</td>
<td>0.34</td>
<td>2.04</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>48</td>
<td>0.34</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>65</td>
<td>0.34</td>
<td>3.06</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>81</td>
<td>0.41</td>
<td>4.51</td>
</tr>
<tr>
<td></td>
<td>11.43</td>
<td>84</td>
<td>0.44</td>
<td>5.0292</td>
</tr>
</tbody>
</table>
Sensor Reading of Voltage on scale from 0 - 1023

Pump to Equilibrium Test #2

Time (s)
## Appendix C – Project Budget

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Code</th>
<th>Date</th>
<th>Quantity</th>
<th>Rate/Unit</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Material A</td>
<td>0001</td>
<td>01/10</td>
<td>100</td>
<td>$100.00</td>
<td>$10,000.00</td>
</tr>
<tr>
<td>2</td>
<td>Material B</td>
<td>0002</td>
<td>01/15</td>
<td>50</td>
<td>$200.00</td>
<td>$10,000.00</td>
</tr>
<tr>
<td>3</td>
<td>Material C</td>
<td>0003</td>
<td>01/20</td>
<td>75</td>
<td>$150.00</td>
<td>$11,250.00</td>
</tr>
<tr>
<td>4</td>
<td>Material D</td>
<td>0004</td>
<td>01/25</td>
<td>25</td>
<td>$300.00</td>
<td>$7,500.00</td>
</tr>
<tr>
<td>5</td>
<td>Material E</td>
<td>0005</td>
<td>01/30</td>
<td>30</td>
<td>$250.00</td>
<td>$7,500.00</td>
</tr>
</tbody>
</table>

**Total Cost:** $45,750.00
Appendix D – HMI Code

==========================================================================================================

**************

==========================================================================================================

**************

==========================================================================================================

**************

Colorado State University - Pueblo

School of Engineering

Date Created : 4/2/2015

Project Name : Sabatier HMI Program

Description : HMI display/control of the Sabatier Reactor

Limitations : (List current limitations of the code)

While program is off or loading, pump is out of control (max)

==========================================================================================================

**************

- IMPORT REQUIRED LIBRARIES  ---------------------------------------------------------------

==========================================================================================================

**************

#include <genieArduino.h>

==========================================================================================================

**************

- GLOBAL DEFINITIONS  ---------------------------------------------------------------

==========================================================================================================

**************

#define COM_LINK 9600

#define VOLTAGE_MAX 500  //map function works for integers, using this value will give decimal on 0-5 scale

#define VOLTAGE_MIN 0

#define SENSOR_MAX 1023

#define SENSOR_MIN 0
#define PUMP_MAX 255
#define PUMP_MIN -255
#define PUMP_DC_OFFSET 94 //Based on tests of average hydrogen production
#define MIN_PRUN_VOLTAGE 3 //The minimum voltage to run pump motor
#define RANGE_MAX 30 //This range is based on empirical measurements of sensor
#define RANGE_MIN -30

}// ARDUINO PIN DEFINITIONS

#define PUMP_OUT_PIN 9
#define SENSOR_IN_PIN 0 //Analog input 0
#define ON_PIN 5
#define OFF_PIN 3

}// GLOBAL VARIABLES

int pumpV = 0; //Numeric scale output of Arduino to OpAmp (0-255)
int prev_pumpV = 0; //Stored value of the pumpV previously sent to the opAmp
float pVolts = 0; //Voltage output of Arduino to OpAmp
int wHeight = 0; //water level in inches
float pVolume = 0; //mL/min of hydrogen
float opAmpV = 0; //Voltage from OpAmp to drive pump
int sensorV = 0; //Numeric scale input of Arduino from sensor (0-1023)
int error = 0; //Sensor error (0 - 255)
int prev_error = 0; //Stored value of the error previously read from the Sensor
int pV = 0; //dummy variable for pump voltage calculations
int refV = 337; //set at the beginning of each cycle

boolean init_state = true;

int prevTime = millis();

//int rocker0_val = 0; uncomment and set serial to check for changes. Make sure to uncomment corresponding below

//int rocker1_val = 0; uncomment and set serial to check for changes. Make sure to uncomment corresponding below

//long prevTime = 0; //For testing

Genie genie;

#define RESETLINE 4 // Change this if you are not using an Arduino Adaptor Shield Version 2 (see code below)

void setup()
{

    // NOTE, the genieBegin function (e.g. genieBegin(GENIE_SERIAL_0, 115200)) no longer exists. Use a Serial Begin and serial port of your choice in

    // your code and use the genie.Begin function to send it to the Genie library (see this example below)

    // 200K Baud is good for most Arduinos. Galileo should use 115200.

    Serial.begin(9600);

    genie.Begin(Serial);

    genie.AttachEventHandler(myGenieEventHandler); // Attach the user function Event Handler for processing events

    // Reset the Display

    // THIS IS IMPORTANT AND CAN PREVENT OUT OF SYNC ISSUES, SLOW SPEED RESPONSE ETC

    pinMode(RESETLINE, OUTPUT); // Set D4 on Arduino to Output (4D Arduino Adaptor V2 - Display Reset)

    digitalWrite(RESETLINE, 1); // Reset the Display via D4

    delay(100);

    digitalWrite(RESETLINE, 0); // unReset the Display via D4

    delay (3500); //let the display start up after the reset (This is important)

    //Turn the Display on (Contrast) - (Not needed but illustrates how)

    genie.WriteContrast(7); // 1 = Display ON, 0 = Display OFF.
// For uLCD43, uLCD-70DT, and uLCD-35DT, use 0-15 for Brightness Control, where 0 = Display OFF, though to 15 = Max Brightness ON.

// Write a string to the Display to show the version of the library used

pinMode(ON_PIN, OUTPUT); // ON
pinMode(OFF_PIN, OUTPUT); // OFF
pinMode(PUMP_OUT_PIN, OUTPUT);
// Serial.println("System ready...");

}

void loop()
{
    /* static long waitPeriod = millis();
    static int gaugeAddVal = 1;
    static int gaugeVal = 50; */
    genie.DoEvents(); // This calls the library each loop to process the queued responses from the display
    genie.WriteObject(GENIE_OBJ_GAUGE, 0, wHeight); // Shows height of water on the gauge object
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0, opAmpV); // Displays voltage being put to the pump through the op amp
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 1, pVolts); // Displays the voltage going from the arduino to the op amp
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 2, pVolume); // Shows the rate at which Hydrogen is being created
    // genie.WriteObject(GENIE_OBJ_LED_DIGITS, 3, (total methane produced variable)); // Insert total methane variable and uncomment to display total methane
    // genie.WriteObject(GENIE_OBJ_LED_DIGITS, 4, (methane rate variable)); // Insert methane rate variable and uncomment to display
    pump();
}

/////////////////////////////////////////////////////////////////
void myGenieEventHandler(void)
{

genieFrame Event;
const int index = 0;
genie.DequeueEvent(&Event);
if (genie.EventIs(&Event, GENIE_REPORT_EVENT, GENIE_OBJ_ROCKERSW, 0) || //Checks rocker switch
0 for any changes
    genie.EventIs(&Event, GENIE_REPORT_OBJ, GENIE_OBJ_ROCKERSW, 0))
{
    digitalWrite(ON_PIN,HIGH); //If changed sets ON_PIN to high
    //rocker0_val = genie.GetEventData(&Event); uncomment and add serial in order to check proper
    input

}

if (genie.EventIs(&Event, GENIE_REPORT_EVENT, GENIE_OBJ_ROCKERSW, 1) || //Checks rocker switch
1 for any changes
    genie.EventIs(&Event, GENIE_REPORT_OBJ, GENIE_OBJ_ROCKERSW, 1))
{
    digitalWrite(OFF_PIN,HIGH); //If change occurs sets OFF_PIN to HIGH
}
//rocker1_val = genie.GetEventData(&Event); un comment and add serial in order to check proper input

}

}

void pump(void)
{
    sensorV = analogRead(SENSOR_IN_PIN);
    if(init_state){ //first reading is taken at equilibrium. later loops drive to equilibrium
        refV = sensorV;
        init_state = false;
    }
    prev_error = error;
    error = (int)(map((sensorV-refV), RANGE_MIN, RANGE_MAX, PUMP_MIN, PUMP_MAX)); //maps input error to output voltage
    pV = prev_pumpV + (7.825* error) - (7.825* prev_error) + PUMP_DC_OFFSET;//pump transfer function
    pV = max(pV, VOLTAGE_MIN); //Ensures the voltage never drops below zero (i.e. the pump cannot run backward)
    pV = min(PUMP_MAX, pV); //Ensures the voltage output is never above the max
    prev_pumpV = pumpV;
    pumpV = pV;
    pVolts = (map(pumpV, VOLTAGE_MIN, PUMP_MAX, VOLTAGE_MIN, VOLTAGE_MAX))/100;
    //wHeight = //Sensor Inches Transfer Function when known
    opAmpV = 1.9338*pVolts + 1.7609;  //Op Amp Voltage Transfer Function
    if(opAmpV < MIN_PRUN_VOLTAGE) {
        pVolume = 0;
    } else {
        pVolume = 7.8366*opAmpV - 5.6729; //Pump Displacement Transfer Function
    }
    analogWrite(PUMP_OUT_PIN, pumpV);
//Event.reportObject.cmd is used to determine the command of that event, such as an reported event
//Event.reportObject.object is used to determine the object type, such as a Slider
//Event.reportObject.index is used to determine the index of the object, such as Slider0
//genie.GetEventData(&Event) us used to save the data from the Event, into a variable.
}