Data Acquisition and Sensor Circuits For The SuPER Project

By

Gustavo Vasquez Jr.

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2006
Abstract

This senior project focuses on an interface between a data acquisition device and three sensors using a Linux operating system. The sensors will sense voltage, current and temperature of the SuPER system and display the respective values on a computer screen. The project will use a source code in C language that enables the interface of the sensor circuits and the data acquisition device. The sensors will be placed on a cart that contains a photovoltaic source (a solar panel), a battery, a DC-to-DC converter, and loads. From this the user can run the source code and be able to know the temperature, current and voltage from the solar panel, battery, and DC-to-DC converter; and from the loads, the voltage and current. Once all the data is acquired, the source code saves the data to a comma separated value file. This will allow the user to export the data to a spreadsheet and be able to plot out graphs, such as, voltage versus current.
Acknowledgements

I would like to dedicate my senior project to my parents, Gustavo and Esther Vasquez, my brothers, Arturo, Omar, Emanuel and Moises, my sisters, Laura, Ana and Rocio, and my best friend Alma Manriquez. All who have given me their love and support throughout my educational endeavors. I would also like to give a special thanks to Alan Jeffrey Ross, who was a good friend and most of all a great teacher.
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>I</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>II</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>III</td>
</tr>
<tr>
<td>List of Figures</td>
<td>IV</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>II. Background</td>
<td>3</td>
</tr>
<tr>
<td>III. Requirements</td>
<td>6</td>
</tr>
<tr>
<td>IV. Characteristics of Voltage, Current and Temperature Sensors</td>
<td>8</td>
</tr>
<tr>
<td>V. Design and Construction</td>
<td>11</td>
</tr>
<tr>
<td>VI. Testing and Results</td>
<td>16</td>
</tr>
<tr>
<td>VII. Conclusion</td>
<td>20</td>
</tr>
<tr>
<td>VIII. Bibliography</td>
<td>21</td>
</tr>
</tbody>
</table>

*Appendices*

A. Source Code in C Language                                          | 22   |
B. Printed Circuit Board                                               | 25   |
C. Analysis of Senior Project Design                                   | 30   |
List of Figures

Figure 1 SuPER System Daily Use ................................................................. 3
Figure 2 Interface between the Sensor Board, NiDAQ and PC ...................... 6
Figure 3 Current Sensors Offset Voltage .................................................... 8
Figure 4 Temperature Sensors Output Voltage at Room Temperature .......... 9
Figure 5 The Current Sensor Circuit .......................................................... 13
Figure 6 The Temperature Sensor Circuit ................................................... 14
Figure 7 The Voltage Sensor Circuit ............................................................. 14
Figure 8 Values displayed from Command Prompt ...................................... 17
Figure 9 Temperature Sensor Results ......................................................... 18
Figure 10 Printed Circuit Board with resistors, capacitors & voltage regulator 19
Figure 11 Schematic of the Op-Amps on the Printed Circuit Board ............... 27
Figure 12 Schematic of the Sensors on the Printed Circuit Board ................. 28
Figure 13 The Printed Circuit Board Layout ............................................... 29
I. Introduction

The twenty-first century is known for low cost, efficient technological electronics that make peoples’ life much easier. More than often, these electronics are run on electricity. However, according to Dr. Harris, an estimated one third of the world population does not have access to electricity, that’s approximately 2 billion people without electrical means [2]. Through the Sustainable Power for Electrical Resources (SuPER) project, an electrical system can be designed to provide electricity to a population that does not have access to electrical power, refrigeration and, most important, lighting. If the SuPER system can become a reality, the standard of living for the 2 billion people living without electricity can be raised.

I was introduced to the Sustainable Power for Electrical Resources (SuPER) project when I read professor James G. Harris’s *White Paper*, published in July 15, 2005 [1]. The paper discusses the idea of building a sustainable source of electrical power system with a 20-year life cycle. This system will be low cost and affordable for a single family in under developed countries. The system is composed of a photovoltaic (PV) source, which will be used to provide power to a household along with a battery for energy storage. Since the power coming from the photovoltaic is DC, the system will only have DC outputs. This will allow the system to only have DC loads, such as, LED’s for lighting, battery charger, and other DC appliances.

My senior project will be a subsystem that focuses on the idea of being able to track the voltage, current and temperature throughout the SuPER system
at different locations using sensors. A location of interest is between the photovoltaic source (solar panel) and the service panel of the system. Other important locations are between the various loads, the circuit breaker panel, and the battery. In order to track the voltage, current and temperature, a device from National Instruments called the USB 6009 data acquisition (NiDAQ) will be used [7]. This device comes with eight analog inputs channels, two analog output channels, twelve digital input/output channels and a 32-bit counter with a full speed USB interface. Using this data acquisition device will make the SuPER system efficient, because the user will be able to know how the system is behaving in terms of voltage, current and temperature.

Throughout this report, the background, requirements, characteristics of the voltage, current and temperature sensors, design and construction, testing and conclusion of the subsystem will be explain. The background section will present the context of my senior project. The system requirements can be found in the requirement section with the characteristics of each of the sensors in the following section. The design and construction section has the actual circuit diagram that was used in the SuPER system. I also added two appendices; appendix A has the source code that was used to interface the sensor circuit and the data acquisition device. Appendix B explains the design of the printed circuit board.
II. Background

Previous to the idea of the SuPER system, there have been a number of solar systems built in recent years. For example, the SunWize System offers a portable solar power generator that can generate DC or AC electricity. This system can power small appliances such as lights, small televisions, fans, laptops, and small power tools. This system also requires little maintenance and provides 190-watts per day of 12 volts DC. However, the system is mostly marketed for emergency power applications and its manufacturer suggested retail price (MSRP) is $1,496 [3].

Unlike the SunWize system, the SuPER system will consist of a 12-volt DC output to supply power to several loads. Those loads include lighting, refrigeration, a computer, battery charging, and a pump. A summary of the system daily use is shown below [1]:

<table>
<thead>
<tr>
<th>Daily Source</th>
<th>Watts/hr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solar Energy Production</td>
<td>400</td>
</tr>
<tr>
<td>Lighting</td>
<td>60</td>
</tr>
<tr>
<td>Pump/Motor</td>
<td>187</td>
</tr>
<tr>
<td>Computer</td>
<td>50</td>
</tr>
<tr>
<td>Refrigerator</td>
<td>50</td>
</tr>
<tr>
<td>Battery Charging</td>
<td>50</td>
</tr>
<tr>
<td>Total Energy use</td>
<td>397</td>
</tr>
</tbody>
</table>

Figure 1 SuPER System Daily Use

The SuPER system cost goal is set to be under $500 dollars, including the replacement of the battery every five years. And if all goes as planned, this turns out to be $2-3 dollars per month to the owner, thus making it affordable to any family no matter their income.
The subsystem that will be designed for my senior project will have three types of sensors, which are voltage, current and temperature. The current sensors that the subsystem will have are the ZAP25 and the ZAP50. The ZAP25 is rated to sense a maximum current of 25 amperes (amps) [4]. The output voltage of the sensor is proportional to the supply voltage. Meaning, at a sensed current that is equal to 0 Amps the output voltage will be the supply voltage divided by two \( (V_o = V_s / 2) \). For example, if you give the current sensor a supply voltage of 5-volts’ its output voltage would read 2.5-volts. The same applies to the ZAP50 current sensor but with a maximum rated current of 50 amperes.

The LM50 SOT-23 Single Power Supply Centigrade Temperature Sensor will be use to display the temperature of the SuPER system. The LM50 is a precision temperature sensor that can sense temperatures that range from \(-40\) degree Celsius to 125 degree Celsius using a single positive supply [5]. The sensor output voltage is proportional to the Celsius temperature \((10 \text{ mV} / ^\circ\text{C})\) and has a DC offset of .5-volts. This offset allows voltage readings to be negative without supplying a negative voltage, and important in our subsystem because no negative voltage will be supplied.

The voltage sensor will use an operational amplifier (Op-Amp) to sense the voltage coming from the photovoltaic source (solar panel). The characteristics of the voltage sensor will depend on the type of Op-Amp the subsystem will use. The current and temperature sensors will be designed with their voltage outputs feeding into the input of an operational amplifier (Op-Amp). The maximum output voltage the ZAP50 sensor can output will be 3.6-volts when
supplying 5-volts to the sensor. The temperature sensor will also have a supply voltage of 5-volts; therefore, the maximum voltage it can output will be 1.75-volts at 125 degree Celsius. In order to keep an accurate track of the current and temperature of the SuPER system the sensors must be able to output higher voltage readings than their maximum voltages. The system will need to have greater output voltages because the system will have high voltages and would like to be able to read those higher voltages.

The three sensors will be connected to three out of the eight analog input channels of the NiDAQ device. The NiDAQ device will read the input values from the output of the Op-Amps and then display the voltage value on a command prompt. However, to make it easier for the user to know exactly what the current and the temperature is, the subsystem will be designed to make the conversion in the software to display the values in engineering units; voltage in volts, current in amps and temperature in degree Celsius. The goal of the project is to interface the voltage, current and temperature sensors with the NiDAQ device using a Linux operating system.
III. Requirements

To achieve the goal of having the data acquisition device display the correct values for each of the sensor outputs, the following requirements must be met. First, it is important to check the current sensors for accuracy at half amps steps from 0 to 25 Amps. Second, verify the temperature sensors voltage at a couple of extreme conditions such as, cold and hot temperatures. From this it can be concluded whether or not the sensors can accurately detect a change of about 1 or 2 degree Celsius. In addition, the NiDAQ device must be verified to ensure that the digital outputs work on all available channels and that the analog inputs generate the expected results when compared against a digital multimeter.

Another requirement is to write a C language source code that would interface the Linux machine, the sensors and NiDAQ, and be able to output the voltage, current and temperature every certain amount of seconds as determined by the user. Figure 2 shows the block diagram of the interface.

![Figure 2 Block Diagram of the Interface between the Sensor Board, NiDAQ and PC](image-url)
The program would output the values of the readings in the following format: voltage, current, temperature and have a time stamp associated with each displayed value. As part of the requirements, the data needs to be written to a text file and to an Excel spreadsheet. Once the data is transferred into a spreadsheet, the user can use the data to plot graphs such as, voltage vs. current and temperature vs. time. Thus making it easier for the user to see the functionality of the SuPER system.

The subsystem also requires a design of operational amplifier circuitry for each of the three sensors. Once the design is set and the best operational amplifier is chosen for the system, the next step will be to verify the measurements with the NiDAQ device using a breadboard. After the measurements have been verified on the breadboard, the next step will be to integrate the circuitry to a printed circuit board (PCB). The last requirement will be to standardize the sensor data variables names and data storage. For example, the solar panel will have three sensors voltage, current and temperature, so the variable names would be voltage PV out, current PV out and temperature PV. The same will apply to the battery, DC to DC converter and the load sensors. However the loads will not have a temperature sensor.
IV. Characteristics of Voltage, Current and Temperature Sensors

As was stated earlier in the background section, the current sensor has an offset voltage of \( V_o = V_s / 2 \), when sense current is equal to zero. The first step to verify the accuracy of the sensor is to confirm that the ZAP25 and the ZAP50 offset voltage is approximately 2.5-volts. The figure below shows the offset of each sensor.

<table>
<thead>
<tr>
<th>Current Sensor</th>
<th>Input Voltage</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZAP25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4.974</td>
<td>2.527</td>
</tr>
<tr>
<td>2</td>
<td>4.974</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>4.973</td>
<td>2.529</td>
</tr>
<tr>
<td>4</td>
<td>4.973</td>
<td>2.519</td>
</tr>
<tr>
<td>5</td>
<td>4.973</td>
<td>2.520</td>
</tr>
<tr>
<td>6</td>
<td>4.973</td>
<td>2.529</td>
</tr>
<tr>
<td>7</td>
<td>4.973</td>
<td>2.516</td>
</tr>
<tr>
<td>8</td>
<td>4.973</td>
<td>2.507</td>
</tr>
<tr>
<td>ZAP50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4.977</td>
<td>2.481</td>
</tr>
<tr>
<td>2</td>
<td>4.977</td>
<td>2.476</td>
</tr>
</tbody>
</table>

Figure 3 Current Sensors Offset Voltages

According to the specification sheet, the offset voltage is half of the supply voltage at plus or minus one percent. And from figure 3, all the current sensors display the expected offset except for ZAP25 sensor number 2. Another characterization was to see that the current sensors are actually outputting the correct voltage. This was done at half amps steps from 0 to 25 Amps with a resistor that can handle at least 25 amperes, a power supply, two digital multi-meters and a breadboard.
The characterization of the LM50 temperature sensors was to verify the accuracy of the output voltage according to the specification sheet. At a given temperature the conversion factor is 10 mV / °C and then add the DC offset of 0.5-volts. The easiest way to verify that the sensor is outputting the expected voltage is to get a thermometer to measure the temperature in the room, multiply the conversion factor to the measured temperature from the thermometer and then add the DC offset voltage. The following figure shows the output voltage of the temperature sensors at room temperature.

<table>
<thead>
<tr>
<th>Temperature Sensor</th>
<th>Input Voltage</th>
<th>Temperature Celsius</th>
<th>Output Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.00</td>
<td>21.4</td>
<td>0.714</td>
</tr>
<tr>
<td>2</td>
<td>5.00</td>
<td>21.3</td>
<td>0.708</td>
</tr>
<tr>
<td>3</td>
<td>5.00</td>
<td>21.4</td>
<td>0.714</td>
</tr>
</tbody>
</table>

Figure 4 Characterizations of Temperature Sensors

According to the Wikipedia Encyclopedia, room temperature is measured to be at 21 to 23 degrees Celsius [8]. The temperature Celsius column shown in figure 4 was an actual measured temperature taken from a thermometer. From the data above, it can be concluded that the temperature sensors are reading the correct temperature except for sensor number two. The following step will be to test the sensors at various temperature ranges to verify that the sensors can actually detect and display the correct temperature.

The voltage sensor was the easiest sensor to characterize because it depends on which operational amplifier works the best for the system. A good operational amplifier will be one that uses a single power supply because there is
no load that will use negative supply voltage. Also, the temperature sensors only use a single power supply.
V. Design and Construction

The design phase of the subsystem was to design two circuits that will amplify the input voltage of the current and temperature sensors. The voltage sensor was designed differently because the voltage coming from the solar panel can be as high as 40-volts. The design needs to be able to step down the incoming voltage below 12-volts or else the output voltage of the operational amplifier will saturate.

The ZAP50 has a rated maximum output voltage of 3.64-volts at 50 amperes. For simplicity, I raised it to 4-volts to be on the safe side. Since the SuPER system uses 12-volts throughout the system, I designed the amplification (of the output voltage of the sensor) by 2.5 because the output voltage must remain under 12-volts. This will always be the case because the maximum output voltage the sensor can output is 3.64-volts then; multiplying it by 2.5, the output will never reach 12-volts. The circuit design will be based on a non-inverting amplifier. The way this works is the output voltage from the current sensor will be input into the non-inverting input (+) of the operational amplifier. Then, the gain of the amplifier will be based on resistor one (R1) and resistor two (R2), which are connected to the inverting input (-) of the operational amplifier. The gain of the operational amplifier is as follows:

\[ V_o = A_{\text{GAIN}} \times V_i, \]

where \( A_{\text{GAIN}} = 1 + \frac{R_2}{R_1} \)
The input voltage is 4-volts and we want the output voltage to be less than 12-volts. Therefore, the design of the circuit was set to output 10 volts, to have a gain of 2.5-volts. This will set R2/R1 to be equal to 1.5. Now, a value can be set for R2 and R1 can be solved. This design also works for the ZAP25 because its rated maximum voltage is 3.425 thus making it easier to only design one circuit for both current sensors.

The temperature sensor circuit was also designed with a non-inverting operational amplifier. The maximum output voltage the sensors can output is 1.75-volts, when sensing a temperature of 125 degree Celsius. Again, to keep it simple, I raised it to 2-volts. The input voltage is 2-volts and the system requires the output voltage to be less than 12-volts. Again, set the output voltage to equal 10-volts. This will make the amplification gain to be 5 resulting in R2/R1 to equal 4. Knowing this, one can set a resistor value for R2 and then solve for R1.

As was stated earlier, the voltage sensor depends on the operational amplifier that will be used. But in order to step down the voltage coming from solar panel, a voltage divider will be needed. The voltage divider works perfect because it allows the photovoltaic source to output the 40-volts, while the voltage divider will step it down to 10-volts. The type of amplifier that would be designed for the voltage sensor is a voltage follower. This type of circuit will allow for very high input resistance; but this does not matter because the voltage divider will handle voltage from the solar panel. The voltage divider will be connected to the non-inverting (+) input of voltage follower. The inverting input (-) will be
connected to the output of the voltage follower making the output follow the voltage of the input.

There is an important reason why the output voltage of the amplifiers needs to be set to 10-volts. This is because the NiDAQ device can only handle a maximum voltage of 10-volts. However, in the software there will be equations that will display the correct values for voltage, current and temperature. The current and temperature equations will read in the values from the NiDAQ device and be divided by their gain of 2.5 and 5 respectively. Then, their offsets will be subtracted and lastly divided by their conversion factor. The voltage sensor will be multiplied by four since the hardware divides the voltage by four.

The construction of the sensor circuits are shown below:

![Current Sensor Circuit](image)

Figure 5 The Current Sensor Circuit
Figure 6 The Temperature Sensor Circuit

Figure 7 The Voltage Sensor

The values of the resistors for each circuit were chosen according to what is available in industry but making it as simple as possible. The operational amplifier that was chosen for these circuits was the LM324. The LM324 is a low power quad operational amplifier. It consists of four independent high gain operational amplifiers. The most important specification is that it uses a single power supply over a wide range of voltages. Another factor is that with one
LM324 chip you can power all three-sensor circuits. This will be a big significance when it comes to cost. The SuPER system will have a minimum of seventeen sensors and with the LM324 the system will only need five chips, instead of seventeen individual Op-Amps, if using a 741 Op-Amp.
VI. Testing and Results

As was mentioned above, the goal of the project is to interface the voltage, current and temperature sensors with the NiDAQ device using a Linux operating system. To see if the current sensors are sensing the correct current, I measure the current at half amp steps, from 0 – 25 amps. The test was done using a resistor that can handle at least 25 amperes, a power supply, two digital multi-meters and a breadboard. The problem with testing at high amperes is the digital multi-meters that we have in the lab can only measure up to 10 amperes. From the data acquired, up to 10 amperes, you can assume that the current sensor will be displaying the correct values there after.

The temperature sensors were tested using a power supply, digital multi-meter, a thermometer and a breadboard. Once the characterization of each temperature sensor was measured, the next step was to test the sensor under two different conditions. First condition was to place the sensor inside my refrigerator and make sure it can sense low temperatures. Second condition was to place the sensor in a place where they can measure high temperatures. I placed the sensors in a box and used a 1 ohm 10 watt resistor to dissipate heat to warm up the box, then check to see if the sensors are sensing the expected temperature.

The voltage sensor was tested using a 3 to 1 voltage divider connected to the non-inverting input of the LM324 operational amplifier. Simply, supply a voltage to the input ($V_{in}$) of figure 6 and verify that the output voltage ($V_{out}$) of the sensor is displaying the correct voltage. For example, if you input 12-volts, the
voltage sensor should be outputting 4-volts. For further verification, the following equation can help:

$$V_{out} = \frac{V_{in} \cdot R2}{R1 + R2}$$

The last part of the testing phase was to actually implement all three-sensor circuits onto a printed circuit board and verify that it is performing correctly.

The overall performance of the subsystem was satisfactory; each sensor circuit outputted the expected voltage. The interface between the sensor and the data acquisition device worked as planned. The figure below shows how the command prompt displays the output values for the voltage, current and temperature sensors.

<table>
<thead>
<tr>
<th>Voltage(V)</th>
<th>Current(A)</th>
<th>Temperature(C)</th>
</tr>
</thead>
</table>

---------- PV Averages ---------
voltage = 5.086217
current = 0.067721
temperature = 21.251606

<table>
<thead>
<tr>
<th>Voltage(V)</th>
<th>Current(A)</th>
<th>Temperature(C)</th>
</tr>
</thead>
</table>

---------- PV Averages ---------
voltage = 5.084588
current = 0.056681
temperature = 21.384538

Figure 8 Values displayed from Command Prompt
As seen from figure 8, with each value there is a time stamp that is associated with each reading. Also, to make the data more presentable, after each five readings the command prompt displays the average for those five readings. When the code is running, it would display values every two seconds. The interface allows the user to take data every 10 seconds, 30 seconds, or every minute; it all depends on the user.

The following figure shows the results of the temperature sensor being expose to warm and cold temperatures.

<table>
<thead>
<tr>
<th>Temperature Celsius</th>
<th>V_{out} (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.70</td>
<td>0.61</td>
</tr>
<tr>
<td>11.60</td>
<td>0.62</td>
</tr>
<tr>
<td>13.40</td>
<td>0.63</td>
</tr>
<tr>
<td>14.10</td>
<td>0.64</td>
</tr>
<tr>
<td>14.50</td>
<td>0.65</td>
</tr>
<tr>
<td>15.90</td>
<td>0.66</td>
</tr>
<tr>
<td>16.40</td>
<td>0.66</td>
</tr>
<tr>
<td>17.90</td>
<td>0.68</td>
</tr>
<tr>
<td>19.40</td>
<td>0.69</td>
</tr>
<tr>
<td>20.30</td>
<td>0.70</td>
</tr>
<tr>
<td>21.30</td>
<td>0.71</td>
</tr>
<tr>
<td>33.10</td>
<td>0.83</td>
</tr>
<tr>
<td>33.20</td>
<td>0.83</td>
</tr>
<tr>
<td>33.90</td>
<td>0.84</td>
</tr>
<tr>
<td>34.90</td>
<td>0.85</td>
</tr>
<tr>
<td>35.90</td>
<td>0.86</td>
</tr>
<tr>
<td>39.30</td>
<td>0.89</td>
</tr>
<tr>
<td>40.00</td>
<td>0.90</td>
</tr>
<tr>
<td>56.80</td>
<td>1.07</td>
</tr>
<tr>
<td>65.90</td>
<td>1.16</td>
</tr>
</tbody>
</table>

Figure 9 Temperature Sensor Results

Figure 9 shows that the sensors can sense various temperatures ranging from cold temperatures to high temperatures. I could have gone higher than 65.9
degree Celsius but this is already sensing a temperature of 149 degrees Fahrenheit. I don’t anticipate the system will ever reach this high of a temperature.

As was mentioned before, the final test was to implement the sensor circuits onto a printed circuit board. Figure 10 shows the printed circuit boards, but for more details refer to Appendix B.

Figure 10 Printed Circuit Board with resistors, capacitors & voltage regulator
VII. Conclusion

This subsystem shows a valid interface between the sensor circuits and the data acquisition device. The data acquired from the subsystem can have significant value as far as what the user will do with the data. The user can plot graphs to see voltage versus current and from this be able to see how much power is being used in the system. The implementation of the printed circuit board was a great idea; each of the loads, the battery, solar panel and the DC-to-DC converter will have its own sensors. This will also make wiring issues disappear because the PCB has connector pins that will be connected into the NiDAQ device.

The difficulties that I faced when performing this project were learning how to use Linux and making sure that I was supplying exactly 5-volts to the input of the current and temperature sensors. This was essential when trying to characterize each of the sensors. Previous to this project, I had never used the Linux operating machine. Although, it was a challenge in the beginning, I am pleased with the overall performance of my senior project.

Future recommendations would be to find out whether or not you can run my source code on a Window operating machine. This will allow the user to compare data with other instruments such as the Winverter. Another recommendation would be to research how to store excess energy once the battery is fully charged, thus making the system more efficient.
VIII. Bibliography


[2] Harris, James G. Graduate Seminar: Development of Sustainable Power for Electrical Resources September 30, 05 EE 563 Graduate Seminar presentation 9/30/05: Development of Sustainable Power for Electrical Resources - SuPER System


Appendix A - Source Code in C Language

/***************************************************************************/
/*
* ANSI C Example program:
*  contAcquireNChan.c
*
* Description:
*  This example demonstrates how to continuously acquire data on
*  multiple channels using the DAQ device.
*
******************************************************************************/

#include "NIDAQmxBase.h"
#include <stdio.h>
#include <time.h>

#define DAQmxErrChk(functionCall) { if( DAQmxFailed(error=(functionCall)) ) {goto Error; } }
#define NUM_TO_OUTPUT 5

int main(int argc, char *argv[])
{
    //system function parameters
    FILE*fp;
    time_t time;
    struct tm *timeinfo;
    char *s_time;

    //data manipulation parameters
    float64 voltage_sum=0, current_sum=0, temperature_sum=0;
    float64 voltage_avg=0, current_avg=0, temperature_avg=0;
    char response[10];

    // Task parameters
    int32 error = 0;
    TaskHandle taskHandle = 0;
    char errBuff[2048]={"\0"};
    int32 i,j;
    time_t startTime;
    bool32 done=0;

    // Channel parameters
    char chan[] = "Dev1/ai0, Dev1/ai1, Dev1/ai2";
    float64 min = -10.0;
    float64 max = 10.0;
// Timing parameters
char clockSource[] = "OnboardClock";
UInt64 samplesPerChan = 5;
float64 sampleRate = 500.0;

// Data read parameters
#define bufferSize (UInt32)1000
float64 data[bufferSize * 3];
int32 pointsToRead = bufferSize;
float64 timeout = 10.0;

printf("Press Ctrl-C to exit\n");
DAQmxErrChk (DAQmxBaseCreateTask("", &taskHandle));
DAQmxErrChk (DAQmxBaseCreateAIVoltageChan(
taskHandle, chan," .DAQmx_Val_RSE, min, max, DAQmx_Val_Volts, NULL));
DAQmxErrChk (DAQmxBaseCfgSampClkTiming(
taskHandle, clockSource, sampleRate, DAQmx_Val_Rising, DAQmx_Val_ConfSamps, samplesPerChan));
DAQmxErrChk (DAQmxBaseCfgInputBuffer(taskHandle, 100000*3)); // use a 100,000 sample DMA buffer
DAQmxErrChk (DAQmxBaseStartTask(taskHandle));

// open a file to write data to
// fp = fopen("Super_Output.txt", "w+");
fp = fopen("Super_Output.csv", "w+");

for (j = 0; j < 100; j++){
    startTime = time(NULL);
    while( time(NULL)<startTime+1 ) {
        DAQmxErrChk (DAQmxBaseReadAnalogF64(
taskHandle, pointsToRead, timeout, DAQmx_Val_GroupByScanNumber, data, bufferSize* 3, &pointsRead, NULL));
        printf("Voltage(V)\t\tCurrent(A)\t\tTemperature(C)\n");
        fprintf(fp, "Voltage\t\tCurrent\t\tTemperature\n");
        for (i = 0; i < NUM_TO_OUTPUT; i++)
            { // Engineering Units
                float64 voltage_pv_out;
                float64 temperature_pv;
                float64 current_pv_out;
                float64 voltage_batt_out;
                float64 temperature_batt;
                float64 current_batt_out;
                float64 voltage_DC_out;
                float64 temperature_DC;
            } // end of for loop
float64 current_DC_out;

time(&t_time);
timeinfo = localtime(&t_time);
s_time = asctime(timeinfo);
s_time[strlen(s_time)-1] = 0;

//get the Voltage from Solar Panel
voltage_pv_out = (data[3*i] * 4);
//get the current from Solar Panel
current_pv_out = ((data[3*i+1] / 2.5) - 2.50) / .037;
//get the Temperature from Solar Panel
temperature_pv = ((data[3*i+2] / 5) - .5) / .010;

//get the Voltage from Battery
voltage_batt_out = (data[3*i] * 4);
//get the current from Battery
current_batt_out = ((data[3*i+1] / 2.5) - 2.50) / .037;
//get the Temperature from Battery
temperature_batt = ((data[3*i+2] / 5) - .5) / .010;

//get the Voltage from DC-DC
voltage_DC_out = (data[3*i] * 4);
//get the current from DC-DC
current_DC_out = ((data[3*i+1] / 2.5) - 2.50) / .037;
//get the Temperature from DC-DC
temperature_DC = ((data[3*i+2] / 5) - .5) / .010;

//get sums for a later average calculation
voltage_sum += voltage_pv_out;
current_sum += current_pv_out;
temperature_sum += temperature_pv;

voltage_sum2 += voltage_batt_out;
current_sum2 += current_batt_out;
temperature_sum2 += temperature_batt;

voltage_sum3 += voltage_DC_out;
current_sum3 += current_DC_out;
temperature_sum3 += temperature_DC;
//output the data to console
    printf("PV_OUT %lf, %s\t", voltage_pv_out, &s_time[11]);
    printf("PV_OUT %lf, %s\t", current_pv_out, &s_time[11]);
    printf("PV %lf, %s", temperature_pv, &s_time[11]);
    printf("\n");

    printf(" VOLTAGE_BATT_OUT %lf, %s\t", voltage_batt_out, &s_time[11]);
    printf(" CURRENT_BATT_OUT %lf, %s\t", current_batt_out, &s_time[11]);
    printf("TEMPERATURE_BATT %lf, %s", temperature_batt, &s_time[11]);
    printf("\n");

    printf(" VOLTAGE_DC_OUT %lf, %s\t", voltage_DC_out, &s_time[11]);
    printf(" CURRENT_DC_OUT %lf, %s\t", current_DC_out, &s_time[11]);
    printf("TEMPERATURE_DC %lf, %s", temperature_DC, &s_time[11]);
    printf("\n");

    //output data to file...
    fprintf(fp, " %lf, %s\t", voltage_pv_out, s_time);
    fprintf(fp, " %lf, %s\t", current_pv_out, s_time);
    fprintf(fp, " %lf, %s", temperature_pv, s_time);
    fprintf(fp, "\n");

    fprintf(fp, " %lf, %s\t", voltage_batt_out, s_time);
    fprintf(fp, " %lf, %s\t", current_batt_out, s_time);
    fprintf(fp, " %lf, %s", temperature_batt, s_time);
    fprintf(fp, "\n");

    fprintf(fp, " %lf, %s\t", voltage_DC_out, s_time);
    fprintf(fp, " %lf, %s\t", current_DC_out, s_time);
    fprintf(fp, " %lf, %s", temperature_DC, s_time);
    fprintf(fp, "\n");

    //end for

    //get averages
    voltage_avg = voltage_sum / NUM_TO_OUTPUT;
    current_avg = current_sum / NUM_TO_OUTPUT;
    temperature_avg = temperature_sum / NUM_TO_OUTPUT;
    printf("\n---------PV Averages---------\n");
    printf("voltage = %fn", voltage_avg);
    printf("current = %fn", current_avg);
    printf("temperature = %fn", temperature_avg);

    //reset averages
    voltage_sum = 0;
    current_sum = 0;
    temperature_sum = 0;
}} //end while

}} //end for
//printf("\nAcquired %d total samples.\n",totalRead);

Error:
   if( DAQmxFailed(error) )
       DAQmxBaseGetExtendedErrorInfo(errBuff,2048);
   if(taskHandle != 0) {
       DAQmxBaseStopTask (taskHandle);
       DAQmxBaseClearTask (taskHandle);
   }
   if( DAQmxFailed(error) )
       printf("DAQmxBase Error: %s\n",errBuff);
       return 0;
}} //end main
Appendix B – Printed Circuit Board

The implementation of the PCB designed has great advantages over keeping the sensor circuits on a breadboard. First, my breadboard does not have enough space to fit all seventeen sensors. Then the location of the breadboard will become a problem as far as where to place it to accommodate the solar panel, battery, DC-to-DC converter, and the loads. The PCB allows for each of the equipment to have its own sensor boards. The following figure shows the schematic of the PCB.

Figure 11 Schematic of the Op-Amps on the Printed Circuit Board
Figure 12 Schematic of the Sensors on the Printed Circuit Board

Figure 11 shows the input of the sensors with their respective operational amplifier. It also shows a voltage regulator, the LM340, that supplies 5-volts to the current and temperature sensors. Figure 12 shows, how the current and temperature sensors are connected to the schematic on figure 11, the voltage divider is shown on the right hand side.
Figure 13 shows the PCB layout.

Figure 13 The Printed Circuit Board Layout
Summary of Functional Requirements

My senior project interfaces three sensors circuits with a data acquisition device using Linux operating machine. The three sensors that were used in my project are voltage, current and temperature. My senior project also contains a source code in C language that enables the interface of the sensor circuits and the data acquisition device. The sensors will be placed on a cart that contains a photovoltaic source (a solar panel), a battery, a DC-to-DC converter, and loads. From this the user can run my source code and be able to know the temperature, current and voltage from the solar panel, battery, and DC-to-DC converter, and from the loads the voltage and current. Once all the data is acquired, the source code saves the data to a CSV file. This file is called, comma, separated value, which allows the user to export the data to a spreadsheet and be able to plot out graphs, such as, voltage versus current. The last part of my senior project was to implement my sensor circuitry onto a printed circuit board. The board works and is ready for operation.

Primary Constraints

One constraint I faced was, keeping a constant supply voltage of 5-volts to the input of the temperature and current sensors when characterizing the sensors. The power supply that I used in the lab says it supplies a constant voltage of 5-volts but this is not true. It actually outputs 5.1-volts and this makes a big difference when trying to measure the offset of the current sensors. Another issue was the data acquisition device default setting. For some reason
the default setting is not set to the default so when testing the device the expected values were not correct. This minor detail was a pain in the neck.

**Economic**

The only cost that I can claim to my senior project is the thermometer that cost $14 and the bag of resistors that cost $3. The original development for project was set for at least a quarter and a half. I had to wait for my colleagues to set up the Linux operating system on the computer. Then make sure that the drivers for the NiDAQ were compatible with Linux. The actual development of my project was about nine weeks. The first four weeks was developing the source code for the interface and the rest of the weeks were for testing and finalizing the design.

**Environmental**

My senior project is based on solar power energy thus making it safe for the environment. The solar power will supply voltage to a refrigerator, a battery charger, pump, LED lights and a TV. The goal is to keep everything DC for low power consumption.

**Manufacturability**

My senior project is based on a five-year plan to design a prototype system that will be able to only supply DC voltage to various loads. My part in the project is having a sensor board to be able to sense the voltage, current and temperature of the system, along with the interface of the computer or laptop with Linux and the data acquisition device. The sensor board should not be hard to
manufacture. The only issue I see is to find the best way to place the sensors on the printed circuit board.

**Sustainability**

The SuPER system will have a life cycle of twenty years and with no maintenance. However, the owner must replace the battery of the system every five years. This project is all about sustainable resources; just look at the name of the project Sustainable Power for Electrical Resources (SuPER). The system will have a solar panel supplying power to various loads as explain the summary. For future upgrades, the system can use other energy storage other than the battery. When the battery is fully charge, the solar panel is still supplying voltage so why not have a hydrogen cell to store more energy. This will help the system be more efficient.

**Ethical**

Some ethical issues relating to the design of this project is the manufacturing of the sensor boards. In order to have the proper sensing the sensors will need to be properly tested. Some might assume that the sensors work and not verify the sensors.

**Health and Safety**

I am too not sure about health issues but as far as safety the owner will need to handle the system with care. The SuPER project is rated to have currents of 30 amps and voltage of forty volts or higher.

**Social and Political**

The social impact that this project will have on people that don’t access to power will be tremendous. This system will give them a way to refrigerate food,
have lighting, and overall just have access to power. In the U.S. we take power for granted because we have it but imagine if you didn’t and how much your life will change. Politically, I can see certain governments purchasing the system to help those individuals in their country that can’t afford the system and have some type payback plan.

*Development*

I can honestly say that working in this project has given me good background knowledge about DC-to-DC converters and power systems without prior knowledge. I also learned how to interface three sensors with the NiDAQ to a Linux operating machine. This is something that I am very proud of myself. The last thing I learned from this project is how to trouble shoot when expected values are not correct.