CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

by Abigail Krich

Self-Powered Solar Data Logger

Abigail designed a microcontroller-based, self-powered solar data logger that uses a photodiode to measure solar insolation levels. The system converts the analog signal to a digital value that's stored in flash memory.

On a seemingly rare sunny day in Ithaca, New York, the sun delivers about 1,000 W of power per square meter and just begs to be put to some purpose besides browning the backs of the students lying out in the gorges. At up to 15% conversion efficiency, COTS photovoltaics (PVs) can turn that light into electricity. A growing number of Ithacans have heeded the sun's call and installed solar electric PV systems to power their homes and businesses. Ithaca may get 40% less solar insolation than San Diego, but it gets 25% more than Germany, the world leader in installed PV capacity. Tompkins County, where Ithaca is located, has roughly 2.9 W of installed solar capacity per person, which makes it second in the U.S. to only Palo Alto, California.

With the Finger Lakes, ridgelines, and valleys cutting through the region, clouds and fog levels vary significantly from one part of town to another. But weather data is only available from a few select locations. How is a potential PV buyer to know how much power her system will produce unless they can measure the incoming light? And how can a proud PV owner know how her system is performing or detect faults unless she can confirm the conversion efficiency? Expensive commercial data logging systems that cost thousands of dollars do this well, but they are entirely unreasonable for the small system owner.

For Bruce Land's ECE476 Microcontroller Design course at Cornell University (http://instruct1.cit.cornell.edu/courses/ee476/), I designed an inexpensive self-powered solar data logger to meet this need (see Photo 1). I built the system around an Atmel AVR STK500 development board that featured an ATmega32 microcontroller. You can leave the logger (untouched and isolated) in the field to collect data for months or years.

SYSTEM OVERVIEW

The solar-powered data logger features a photodiode that measures solar insolation levels and converts this analog signal to a digital value that's stored in flash memory. Every time the system logs a data point it also logs a time and date stamp so that the data can be downloaded to a PC and analyzed in the future.

When the system is logging, real-time data is displayed on its small LCD screen. The system displays several useful bits of information: the battery's voltage, the time and date, the length of time the system has been logging, and the length of time it can continue to log before running out of memory.

The logger has a dedicated solar power system that enables autonomous operation. A simple charge controller regulates the charging of a sealed battery (gel cell lead acid) by a small solar panel.

APPLICATIONS

When planning an off-grid solar electric power system (one that isn't connected to a larger power grid), the output must be matched closely with the load in order to provide sufficient power without considerable waste. Although the price of photovoltaics has dropped by almost fivefold in my lifetime, it is still quite expensive.[1] Good planning and system design ensure that you can provide sufficient power without having to buy more PV than necessary.

The power output from photovoltaics is directly related to the insolation level. Although seasonal and annual average insolation levels for most major U.S. cities are available on the Internet, cloud cover and other weather effects can be extremely localized depending on the topography. Thus, the data for large cities isn't always the same as the data for the smaller towns in its vicinity. In addition, this data is not available for every part of the world. While accurate predictions of power output are important for grid-tied solar electric systems (in which the grid is used as 'storage' for any excess electricity produced and drawn upon for any electricity shortfall), this mainly impacts the return on investment expectations rather than the system sizing.

PV power systems are the only devices currently available for generating electricity without any moving parts. This makes them brilliantly simple and easy to care for, which is a real benefit for homeowners who do not want to spend their weekends greasing bearings and performing regular system checks. But without any maintenance needs or means for visual inspection, it is easy for system faults to go undetected for quite some time.

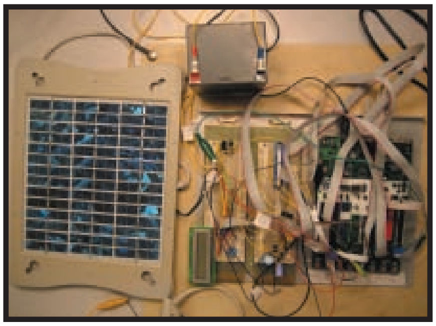

Photo 1—*Check out the complete solar data logger system with the PV panel, battery, and a mess of wires.*

Large commercial PV installations typically have sophisticated sensors and monitoring software that can detect system faults and activate an alarm when maintenance is needed. These features are usually too costly for residential sized PV systems. The central component of these monitoring systems is an insolation sensor whose output is compared with system power production. When the power production strays significantly from what would be expected given the insolation, an alarm is triggered and maintenance checks can be performed. Although you may not need a fully automated, integrated monitoring system, some means for determining your PV system's efficiency enables you to perform maintenance only when it's necessary and to have peace of mind at other times that everything is functioning properly.

HARDWARE

The system schematic is shown in Figure 1. The pins of the STK500 board are depicted along the bottom. Note that the RXD and TXD pins for the RS-232 connection and the data flash are also located on the board.

Port A connects to the photodiode and the switches. Port B also connects with the switches as well as the data flash. Port C controls the LCD and Port D sends data to the RS-232 connection and the field effect transistor. These ports connect to the ATmega32 microcontroller.

POWER SUPPLY

The logger has a dedicated solar-charging system to enable for autonomous operation. It was built around a 3.2-W solar panel a Volkswagen dealer gave me to accessorize the originally diesel-fueled Volkswagen Golf TDI that I bought and converted to run on vegetable oil. But that car is another story.

Even without much optimization for efficiency, the power needs of the logger are very small. The maximum power drawn by each of the main components is 5 mW for the LCD, 75 mW for the ATmega32, and 75 mW for the photodiode for a total of 155 mW. If all components run at maximum power at all times, this would amount to 3.72 Wh/day. This does not account for losses in the voltage regulator or other minor components, but there is plenty of energy available for the system so this is not a problem.

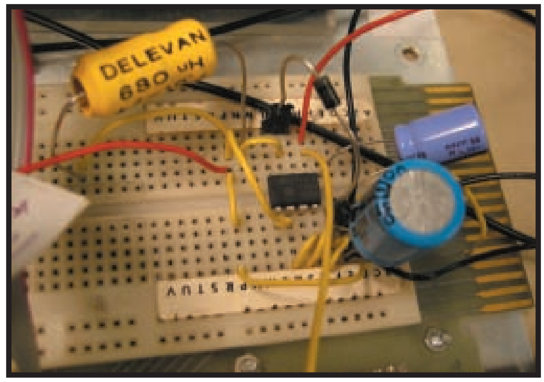

Photo 2—*The LM2574 step-down switching regulator maintains the voltage supply to the logger at 5 V.*

The battery I chose was a 12-V, 5-Ah sealed gel cell lead acid deep cycle battery. This type of battery is the most cost effective when size and weight are not a concern, but safety, ease of handling, and the ability to deep-cycle the battery are. Although a 12-V, 5-Ah battery could provide 60 Wh, draining any battery too low (even a deep cycle battery) can cause damage. Five days of energy storage in Ithaca is considered conservative, but due to the reliability needs of an autonomous system such as this, it is worth having a good factor of safety. Note that 37 Wh of useful storage enables the system to ride through 10 days with no sun, giving plenty of leeway with the battery chosen.

Even with enough storage, it is necessary to be sure that the energy balance of the system is kept positive or the battery will eventually drain. Ithaca averages 2.3 sun hours per day in the winter.[2] What this means is that a solar cell rated at 1 W would produce 1 Wh of electricity per sun hour. My 3.2-W solar panel would therefore be able to produce 7.4 Wh (on average) of electricity per day during the darkest time of year in Ithaca if kept at its maximum power point. Because there is no maximum power point tracking in this system, it can be expected that the panel will produce about 5.2 Wh per day in the winter, which still far exceeds the maximum load expected.

The PV panel, which is rated at 18.8 V at its maximum power point, was designed for trickle charging a 12-V car battery and so could be directly connected to the data logger. However, to prevent battery damage from overcharging, I needed a way to disconnect the panel once the battery was fully charged. Using a BUZ71 field effect transistor and a polling routine, the panel was effectively disconnected when the battery voltage rose above 12 V.

Depending upon the state of charge, the battery voltage will float around 12 V but will not remain steady. The logger components required a constant 5-V power supply. The ON Semiconductor LM2574, a 0.5-A, 5-V step down switching regulator (buck converter) regulates the voltage from the battery to the level needed for the system components (see Photo 2). This regulator has a typical efficiency of 72%, which is much higher than resistance-based voltage regulation.

An unresolved and bizarre result of running the logger off of the solar power supply as compared to the standard AC/DC power supply was that the on-off switch on the STK500 ceased to function. The only way to turn the logger on or off when connected to the solar power supply was to actually disconnect a battery lead.

PHOTOSENSOR

A Texas Instruments OPT101—a monolithic photodiode and single-supply transimpedance amplifier—is used to sense the incoming solar insolation level. Natural sunlight ranges in intensity from 0 to approximately 1,000 W per square meter, but the OPT101 puts out its maximum voltage at roughly 10 W per square meter of incident insolation. With hardly any light striking the sensor, it reached its upper limit. It was thus necessary to attenuate the intensity of the light striking the photodiode to increase the range over which the sensor could differentiate intensity.

Ideally, this would have been done with neutral density filters, but I didn't have any on hand. I used the next best scientifically accurate and available tool: electrical tape. Two layers of electrical tape covering the photodiode were found to be quite effective. This solution enabled the sensor to give reasonably reliable readings over the full range of expected intensities (see Photo 3).

The microcontroller's ADC has a maximum value of 255. Multiplying by a factor of four roughly converted the ADC reading to watts per square meter. Ideally, this system would be calibrated and the factor would be more precise than four, but this approximation gave reasonable results with a range of readings from 0 to 1,020 W per square meter.

I used the ATmega32-based STK500 development board because of its integral flash memory and switches. However, if this system were commercialized or rebuilt, it is fairly obvious that the STK500 would not necessarily be used. It has many features that are unnecessary for this project. A far simpler and more compact board can be designed.

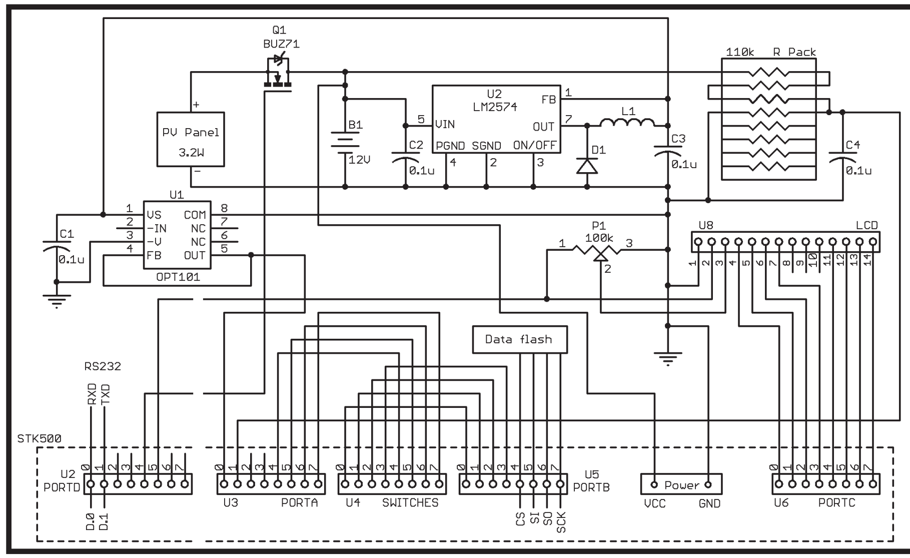

Figure 1—*The system's schematic shows the STK500 and its connections with each of the other components.*

PROGRAM

An interrupt-driven program runs the system and enables accurate timing. An interrupt service routine decrements a series of task timers once per millisecond. The main task in the program calls various subroutines at predetermined intervals when their task timers run out and the appropriate flags are set to enable a task to run. Each of the eight buttons is polled separately once every 30 ms with a state machine to debounce the button as it stabilizes after a transition. The only button that is not debounced is the LCD Wake button because bouncing is not a concern with this function. The debouncing routine is based on code written by Bruce Land.

Each time the system is turned on or reset, it welcomes the user and guides her through the set-up process (see Photo 4). During set-up, you set the system time and date, select the logging frequency, decide whether to clear any stored data or to continue by appending future data, and indicate when to actually begin logging after the system is in place.

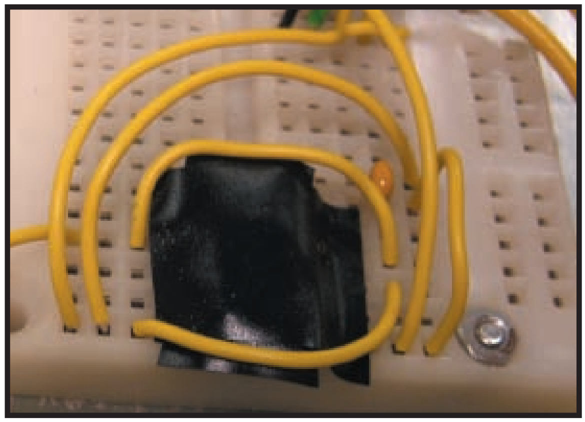

Photo 3—*The photosensor is covered with two layers of electrical tape to provide a larger range of sensitivity. The inset shows the sensor without the tape cover.*

In order to enable data appending after a reset, the memory pointers have to be stored in nonvolatile memory and initialized only when the chip is programmed, not each time the system is reset. The ATmega32 contains 1,024 bytes of data EEPROM memory organized as a separate data space in which single bytes can be read and written. The pointers to the flash memory as well as a log of how long the system has been logging are kept in EEPROM. During operation, you can reset the time if needed or change the logging frequency without interrupting the data collection. If the logging frequency were changed, it would be impossible to calculate how long the system had been logging unless a running tally had been kept.

A series of flags and state machines are used throughout the program to prevent erroneous user input from activating a section of code out of order. At any given point in the program, only the relevant buttons are active and their functions change as shown in the button labels in Photo 5.

The battery voltage and the photodiode output are fed into two channels of the ATmega32's ADC. The ADC has a maximum input of 5 V. So, in order to read the battery voltage, it was necessary to use a voltage divider to guarantee that the input to the ADC was within range. Initially, the system sampled only the light level with the frequency at which the user wanted data stored. However, with logging frequencies of 1 min. to 1 h, this did not allow for a satisfying real-time display that showed changes in light intensity. Moreover, it allowed less accuracy if only one sample was taken per stored data point. Instead, the final design

Photo 4—*The data logger greets users before leading them through the system setup.*

reads the ADC once per second through a task that enables an ADC interrupt routine and starts a conversion. The ADC interrupt automatically switches between the two channels for the photodiode and the battery voltage, reading the battery voltage once for every five times the photodiode is read.

To get an accurate battery voltage reading, the PV panel is disconnected just before the reading is taken and then reconnected immediately after if the battery voltage is below 13 V. If it is at 13 or above, the PV panel is not reconnected to prevent overcharging.

The instantaneous readings are displayed to the LCD screen and kept in a running average over the logging interval. When the logging interval is complete, a data point is stored in the flash memory and the running average is cleared.

To be able to run for a useful length of time, the data logger needed an external memory for storing measured data. The older STK500 boards have flash memory chips built into them (as can be seen on the upper right of the board in Photo 5), making this an obvious choice. Unfortunately, I did not have any instruction set for interfacing with the flash memory. An earlier ECE476 project used this flash memory. Based on the project's code and comments, I was able to learn how the flash memory interface worked.[3] The designers used a program named dFlash, which was written by Terje Frostad of Atmel Norway. I e-mailed the Atmel AVR technical support team and received permission to use dFlash in my project as well. The dFlash program provides a set of routines to interface with the flash memory by writing to a 264-byte buffer. This buffer is then written to one of 1,024 264-byte pages of the flash memory.

Because the routines take byte-size data as input, I had to break the insolation value (stored as an integer) into 2 bytes and then put it back together again when storing or retrieving it. The 2-byte insolation together with the time and date stamp meant that each time the system logged a data point it used 8 bytes of memory. This allowed the system to log 33,792 data points. The logging interval is user-selectable from 1 min. to 1 h. At an interval of 1 min., the system is able to continuously log data for 23.4 days before filling the memory. At an interval of 1 h, the system is able to log for 3.8 years.

One thing I did not realize that caused me a considerable headache was that PortB.4:7 is used for interfacing with the flash. I had initially been using PortB as input for the switches, but I found that three of the switches ceased functioning properly. When I realized that dFlash was reinitializing these pins, I was able to switch these buttons to PortA.

The flash memory can be read either through a buffer or directly, the latter being my choice for the logger. If you were to call for the data to be retrieved before the buffer had filled and written the data to flash, there would be no data to retrieve. Thus, it was necessary to write the buffer to flash memory just before the data retrieve routine was entered as well as when the buffer became full. It was also necessary to give the microcontroller a brief period to finish writing the buffer before the read command was executed or the same problem would occur.

The read command is executed when you press the Retrieve Data button (active only when the logger is stopped). The logger must be connected to a computer using an RS-232 cable with straight-through connection. You should start a simple terminal program on the PC (e.g., HyperTerminal) set to 9,600 bps, no parity, 1 stop bit, and no flow control.

When the Retrieve Data button is pressed, the system prints identifying header rows followed by a row for each data point logged with the time and date at which it was stored. The LCD displays the message "Uploading

Data" followed by the time and insolation value presently being uploaded. Upon reaching the end of the data, a footer row is printed to the terminal program and the LCD displays the message "Done Uploading Please Restart." The data in the terminal program on the PC may then be copied into another program for storage or analysis. If an incomplete transmission was made, you may reset the logger and choose to upload the data at the first prompt. The data may be uploaded as many times as the user desires until the memory is cleared.

Initially, the program was written with each of the subroutines that updated a parameter sending its update to the LCD. This avoided updating the LCD more frequently than necessary. However, when power-saving code was incorporated to turn the LCD screen off when the system was idle by driving it with a port pin, the system would freeze every time it went idle. It turns out that the LCD sends and receives messages to and from the microcontroller. If the LCD is off when the microcontroller sends it a message,

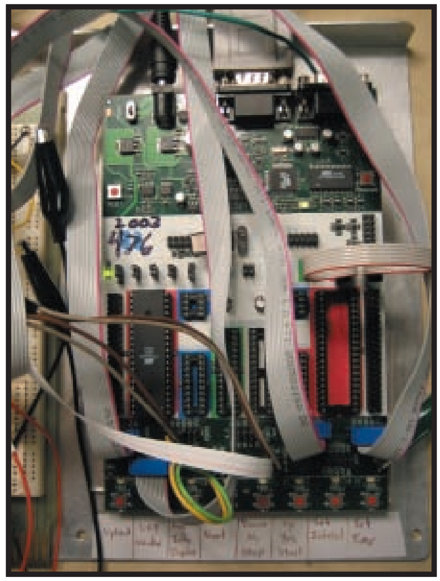

Photo 5—*Check out the STK500 board with labels for each of the eight buttons.*

it is not able to respond and the microcontroller hangs, waiting. The program thus had to be rewritten to prevent the LCD from being called when the LCD screen was turned off. I did this by writing a single routine that printed to the LCD screen. A state machine and a variety of flags controlled it so that the correct message would be displayed.

Additionally, there was some difficulty at times with the LCD not making proper contact with the surplus whiteboard I was using. When the LCD lost contact momentarily, it would cause the system to freeze. Thus, I added an extra new whiteboard solely to hold the LCD screen securely because I was not able to solder the borrowed LCD to my project. Another problem I had with the LCD was when the STK ground was not properly connected to the system ground, the negative terminal of the battery. This happened during testing with the STK500 plugged into an AC/DC power supply. When the two grounds were not at equal voltages, the voltage across the LCD was not in its operating range. The positive voltage was coming from the STK port pin and the negative voltage from the battery negative terminal. Once the two systems were joined by connecting to the STK ground pin (or also powering the STK from the battery), the LCD resumed normal operation.

IMPROVEMENTS

If I rebuild this system for fun or commercialization, it will need to be packaged in a weatherproof enclosure to allow for outdoor operation. Additionally, since not everyone has a spare PV panel lying around the house, it would be important to increase the system's efficiency to allow for a reduced panel size because this is the most expensive component. This can be done in many ways.

The first way would be to have a maximum power point tracking charge controller rather than the simple on-off switch used in this project. This would keep the voltage of the solar cells at the maximum power point on the I-V curve. It would effectively extract about 30% more power from the panel than allowing the battery to determine the panel's operating voltage. The second way to improve efficiency would be to reduce the power consumption of the system. Because I had plenty of power from my PV panel and sufficient battery capacity, there was no need for this system to do more than turn the LCD screen off when idle. However, it would be possible to reduce power needs further by reducing the chip's clock speed, letting the chip go idle between readings or at night, and eliminating the LEDs on the STK500 board or using another board entirely.

The sun was shining in my eyes as I started working on this article last summer. I think I'll soon have to join the fold and get a PV system of my own to go with the logger. ■

Abigail Krich (ajk28@cornell.edu) holds a B.S. in biological and environmental engineering and a Master's degree in electrical engineering from Cornell University. She is now a project developer at Tamarack Energy, a renewable energy development company. She has previously worked at the National Renewable Energy Laboratory's National Wind Technology Center and at Northern Power Systems.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/198.

REFERENCES

[1] R. Wiser, M. Bolinger, P. Cappers, and R. Margolis, "Letting the Sun Shine on Solar Costs: An Empirical Investigation of Photovoltaic Cost Trends in California," LBNL-59282, NREL/TP-620-39300, 2006, http://eetd.lbl.gov/ea/ems/reports/59282.pdf.

[2] Solar Insolation for Major U.S. Cities, Advanced Energy Group, www.solar4power.com/solar-power-insolation-window.html.

[3] S. Jean-Louis and C. Pandarinath, "The Big Red Guide," ECE476 Final Design Project, Cornell University, 2004, http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2004/sj74/main.htm.

RESOURCE

Datasheets, dFlash.c, and the User Guide, http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2006/ajk28/ajk28/index.html.