Paper ID #18658

BYOE: Student-built Versatile Platforms Integrate Solar-powered Microprocessor and Sensors for Chemical Engineering Data Acquisition

Rachel J. Monfredo, University of Rochester

Lecturer and Senior Technical Associate Department of Chemical Engineering Teach Freshman workshop, Junior and Senior Chemical Engineering laboratories.

David J. Schinsing
James Alkins, University of Rochester
Mr. Thor O. Olsen

BYOE: Student-built Versatile Platforms Integrate Solar-powered Microprocessor and Sensors for Chemical Engineering Data Acquisition

Abstract

Chemical Engineering freshmen at the University of Rochester were tasked with building their own solar-powered microprocessor systems to experience hands-on machine shop training, coupled with exposure to microprocessors, sensors, and data collection in a weekly workshop associated with their Green Energy course. The student-built unit was supported on plywood cut on a table or miter saw, with mounting holes created by a drill press. Students were taught the fundamentals of soldering to attach a power cable to a 2.5W solar panel to provide power to the microprocessor, and a small voltmeter panel to provide real-time voltage readings. Students had a variety of low-powered sensors (i.e. temperature, humidity, sound, light) to choose from. In alignment with the objectives of their Green Energy course curriculum, students, acting individually or on self-selected teams, were challenged to collect data from sensors chosen for creative application to some simulated aspect of green energy production or use—monitoring environmental effects, evaluating a future collection site, or assessing the production process itself. The on-board EEPROM enabled students to store up to 512 sensor readings on the microprocessor for subsequent transfer to a computer. Students were shown how to collect data generated by their solar-powered sensor and perform rudimentary calculations to understand the implications of the data. Forty-seven students presented their findings orally at the end of the semester (seven individuals, and fifteen 'teams' of two to four students). The experience exposed students early in the major to the use of sensors, microprocessors, Arduino software, (remote) data acquisition, and the data processing methods useful for their upper level unit operations and process control laboratory courses. Projects included evaluating the economic potential of solar panels or wind turbines installed on campus buildings, monitoring the temperature changes in a recyclable-material parabolic trough, and developing smart agriculture irrigation systems based upon soil moisture readings. Voluntary feedback from thirty-seven students at the end of the course indicated that more than two-thirds of the respondents 'Agreed or Strongly Agreed' to queries that the machine shop training was valuable, the hands-on assembly of components was enjoyable, and developing and running experiments was enjoyable. Nearly fifty percent of the class experienced an increased interest in green energy generation. Over ninety percent of the team-based respondents indicated that the opportunity to work on a team was valuable.

Introduction

In April 2014, graduating seniors at the University of Rochester requested a meeting with the chemical engineering department chair and professors, and the Dean and Asst. Dean of the Hajim School of Engineering and Applied Sciences to review and critique the chemical engineering curriculum. One of their requests was for the creation of a lab or hands-on project in the freshman introductory chemical engineering (CHE) course, CHE150, Green Energy. Students noted that other departments, specifically Mechanical and Biomedical engineering, had freshman courses incorporating such experiences, as do many other universities.[1-5] Their request reiterated comments heard annually from juniors taking the spring semester CHE246 unit operations laboratory during which machine shop training is provided in preparation for the senior laboratory course. The senior lab course is typically popular with students, with extensive positive end-of-college "Exit Survey" feedback, however, students comment that they wished they had had the opportunity to perform hands-on projects earlier, and that they had gotten into the campus machine shop earlier than second semester of their junior year:

> In general, this program starts hands on experimenting and training very late which is such a shame… I would have liked to have more training in the machine shop since that's a great thing to be able to put on your resume. Hands on experience is key for those people not going on to get a PhD. (Anonymous, post-course exit survey, 2012)

> … I think the machine shop training during CHE246 [the junior chemical engineering laboratory course] was very helpful and interesting, and wish that I had more time doing projects dealing with the machine shop… (Anonymous, post-course exit survey, 2012)

Additionally, instructors of the junior and senior laboratories find students lack basic assembly and implementation skills for computer-based data-acquisition (DAQ) measurements of common experimental parameters such as temperature, flow rate, and pressure. Students are provided ample theory in their core chemical engineering courses, and are generally capable of analyzing data collected on existing laboratory equipment. However, these same students rarely understand the physical components needed to set up and implement their own DAQ systems.

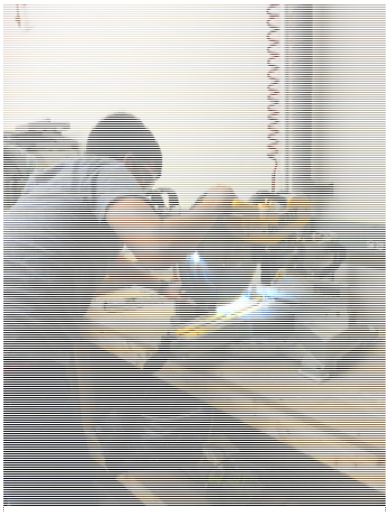


Figure 1. Students cut plywood on rip or chop saw.

The following first-semester freshman project was implemented in the fall 2016 semester to address these concerns. The main goals were to expose chemical engineering freshmen to machine shop facilities, and to microprocessors and different sensors as a means of collecting data. The multi-week, hands-on project encouraged creative use of sensors through applications of the students' choosing, and in concert with their Green Energy course goals, exposed students to the challenges of green energy/power generation. The over-arching implication being that the projects were to use green energy (as supplied by the solar panel) to evaluate other authentic forms of green energy.[6, 7] The requirements of the final project were intentionally vague: to find a green energy application for their chosen sensor to some simulated or actual aspect of green energy production (i.e. monitoring environmental effects, evaluating a future collection site, or assessing the production/implementation process itself). Students were told they needed to come up with a means of interpreting the raw data the sensor generated (typically a voltage signal) and apply a calibration method in order for the data to have meaning (i.e. use an ice bath and boiling water for the waterproof temperature sensor). The intent was to employ a form of hybridized discovery learning with project-based learning to provide students an open-ended challenge to investigate more deeply one of the various forms of green energy they were learning about in the main course;[8] the kind of "creative and critical thinking exercise" Felder suggests to make their coursework more meaningful and memorable.[9] The overall project was explained to students early in the semester, with weekly descriptions of one or another sensor that would be available. This was intended to motivate the students to consider the forms of green energy as they were being taught, from the perspective, 'how could I evaluate this?' and to generate projects which would, as Blumenfeld et al. claim "build bridges between phenomena in the classroom and real-life experiences."[10, 11] In the absence of a formal laboratory space associated with the Green Energy course, the students were expected to generate their own 'laboratory setting' within the surrounding campus or beyond—to create a legitimate (if simulated, in some instances) real-world application. In addition, students were encouraged to work on teams if they chose (up to four members). The success of the 'real-world application' challenge as a teamwork effort is the topic of a future paper.

Figure 2. Students solder components to solar panel.

Accomplishments included fabrication of a solar-powered DAQ system, submission of a written proposal for an intended application of their sensor(s), and an oral presentation to the class of the project results. Students voluntarily completed an end-of-semester survey on their achievement of the intended goals. Forty-seven students completed the course (eight were not declared chemical engineers or freshmen); an additional five dropped the course within the first few weeks of class. Thirty working solar-panel boxes were completed for twenty-three individual or team projects (all students completed initial machine shop training), and thirty-seven surveys were completed (79% response rate). One professor lectured for 75 minutes, twice weekly on material derived from Richard Dunlap's *Sustainable Energy*[12] for the 14-weeklong semester. A second professor led the 'workshop' portion of the class meeting once weekly for 75 minutes: the first half of the semester focused on readings and assignments from Raymond Landis'

Studying Engineering: A Roadmap to a Rewarding Career[13] while students built the solar panel boxes in a campus machine shop outside of class time. Students were introduced to the sensors during the second half of the weekly workshop sessions. Students were shown how to wire sensors to breadboards or directly to the Arduino-compatible microprocessors (the less-expensive RedBoard from SparkFun.com, also the source of the low-powered sensors) mounted on the solar panel boxes. Operation of pre-existing Arduino software code (developed specifically for use with the solar-powered units, included in Appendix) demonstrated the process of data collection with the microprocessor and how to transfer data to a computer. The workshop ended with the final oral presentations.

Apparatus Design and Assembly

Students completed a brief safety course prior to working in the machine shop. All components, shop equipment, and staff support were available to students in one-hour sessions. Most students completed their units in two to three sessions. The solar panel boxes (SPBs) had a series of inexpensive components to enable versatile functionality. A Sterilite® 6-quart shoe box served as the housing, with the lid becoming the base of the structure. A 10.5” x 5.75” x 0.25” plywood piece was cut by students on a rip or chop saw (Figure 1). A drill press was utilized for drilling holes for mounting the plywood to the Sterilite® base, and for holes to mount a SparkFun Electronics RedBoard©, and a 0.25” panel meter. A 3.29” x 2.15” x 0.33” breadboard was mounted with pre-attached double-stick tape. Soldered to the underside of an Allpowers solar panel (130mm x 150mm, 2.5W/500mAh mini-encapsulated solar cell epoxy solar panel) were the wires of the panel meter, wires for alligator clips for additional optional accessories, and a barrel jack plug cable for powering the RedBoard (Figure 2). The solar panel was held in place with hot-melt adhesive at each corner (Figure 3). See Appendix A for a complete parts list. The Sterilite® box provided storage space to the components as the ‘lid’, as well as protection from environmental hazards and backpack storage.

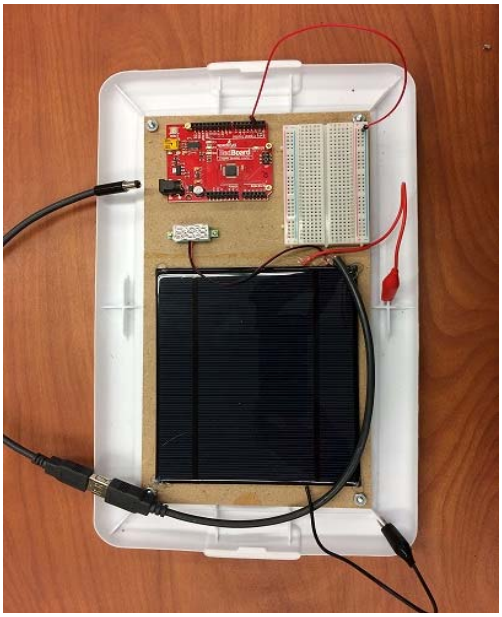


Figure 3. Completed unit with solar panel, breadboard, voltmeter panel, microprocessor, power and accessory cables.

Collection Methodology

Sensors for the project were chosen from the SparkFun Electronics website for low operating current and voltage requirements. Simplicity of wiring and data acquisition were secondary considerations. Some sensors as breakout boards required additional wiring to enable flexible application in novel contexts generated by the students. Table 1 provides a chart of the sensors used, voltage and current requirements, wiring considerations, and the form of the output signal.

Table 1. Sensors used by students and supported by Arduino code for data collection on RedBoard microprocessor.

Sensor	SparkFun Part No.	Cost per unit*	Voltage	Operating Amps	Add'l Wiring	Data from Arduino code
2.2" Flex Sensor	SEN-10264	$7.95	5V	--	Voltage and Ground wires, 47kΩ resistor	Analog output
Ambient Light Sensor	BOB-08688	$4.95	3.3V or 5V	--	Voltage, Signal and Ground wires	Analog output
Digital (Ambient) Temperature Sensor, TMP102	SEN-11931	$5.95	3.3V	10μA	Voltage, Ground, SDA, SCL, ADD0 wires	Temperature in Celcius when multiplied by constant
Force Sensitive Resistor, square	SEN-09376	$7.95	5V	--	Voltage and Ground wires, 3.3kΩ resistor	Analog output
Force Sensitive Resistor, round	SEN-09375	$6.95	5V	--	Voltage and Ground wires, 3.3kΩ resistor	Analog output
Relative Humidity and Temperature Sensor	RHT03	$9.95	5V	40-50μA standby, 1-1.5mA measuring	N/A	Temperature in Celcius when multiplied by constant; percent relative humidity
Soil Moisture Sensor	SEN-13322	$4.95	5V	--	Voltage, Signal and Ground wires	Analog output
Sound Detector	SEN-12642	$10.95	5V	--	Voltage and Ground wires, 3 Signal wires for Audio, Gate and Envelope	Analog output from the envelope wire
Waterproof Temp Sensor	TMP30 (not Sparkfun)	N/A	5V	--	Voltage, Signal and Ground wires	Analog output

*Cost values as of August 2016

A special Arduino code was written to enable students to bring their sensor (wired to the SPB) outdoors to be powered by the solar panel (see Appendix B for sample code). The EEPROM provided storage of up to 1024 bytes, or 512 data points. Students were required to enter the number of data points they wished to collect (especially if multiple sensors were employed simultaneously) and how often data would be collected (in milliseconds). A jumper wire controlled the onset of data collection or retrieval from the microprocessor to the Arduino's Serial Monitor when the unit was reconnected to a computer (see Appendix C for instructions for collecting data). A separate data collection mode, utilizing power through a computer USB port, could be used in the absence of sunlight, and for testing and calibration. Part of the project's

objective was for students to make the sensors' "analog output" signals 'meaningful' by establishing their own calibration criteria.

Challenges and Student Feedback

By far the greatest challenge of the project was preparing all of the sensors—wiring and testing SparkFun break-out boards—and ensuring the Arduino software interfaced properly with each sensor. Several students responded in the end-of-semester survey that they wished they had had more time to work with the sensors beyond the three weeks provided. Other challenges included multiple students with color vision deficiencies who incorrectly wired sensors to their breadboard/microprocessor, students' failure to detect loose wires when systems weren't working, and complaints that the lack of sunlight in upstate New York interfered with students' ability to collect data. These were far outweighed by the popularity of the project and the initiative students demonstrated collecting data in novel and challenging contexts: students seeking out building facilities managers for access to rooftops and otherwise inaccessible windows to collect light exposure or wind pattern data (utilizing the sound sensor) with supporting economic feasibility studies; a student bringing home his SPB over the Thanksgiving break to gather water temperature data upstream and downstream from a low head hydroelectric facility to evaluate the environmental impact; or continuous multi-day attempts to gather soil moisture data (indoors) as various houseplants transitioned from drenched to dehydrated states to investigate the application of soil moisture sensors in farming irrigation systems. One team of students successfully demonstrated the operation of a parabolic trough, made from recycled soda bottles and aluminum foil, utilizing the waterproof temperature sensor.

Thirty-seven students completed a voluntary, anonymous end-of-semester survey providing feedback on various components of the project, a seventy-nine percent response rate. Response categories included Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree. The project inspired increased interest in green energy and in the chemical engineering major for a large portion of students in both categories (Figures 4 and 5). Overwhelmingly positive responses, where combined Agree and Strongly Agree totals exceeded two-thirds of the respondents, were in the students' perception of the value of machine-shop training (84%) and the hands-on assembly of the boxes (76%), the value of microprocessors for collecting data (68%) and slightly-less, their enjoyment of developing and running the experiments (65%), the numbered likely tempered by those with wiring problems.

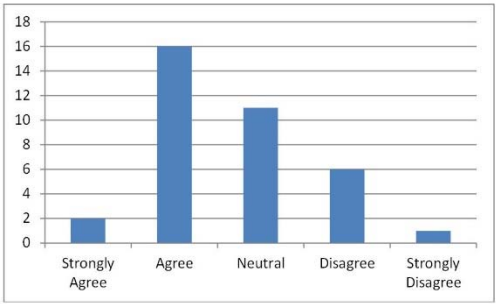


Figure 4. Student responses to "This project increased my interest in Green energy generation."

The workshop was not intended to teach students how to write computer code in the Arduino language, so some concern was expressed by instructors that students would not be able to use the provided code to collect data. Three separate class periods were dedicated to demonstrating the use of the SparkFun RedBoards, breadboards, and the operation of the Arduino code with the sensors. Handouts with step-wise instructions for collecting data were provided, in addition to wiring diagrams and detailed instructions accompanying individual sensors (sample provided in Appendix D). Handouts were distributed after students submitted one-page proposals for their intended sensor application. Multiple, open office hours were offered to students for additional help. Survey responses indicated a mixed response that sufficient background in Arduino programming language was provided to collect data (Figure 6). This indicates a quarter of the class still struggled with this aspect of the project, a problem that needs to be addressed in future course iterations.

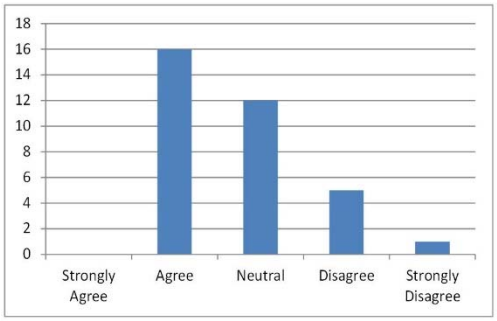


Figure 5. Student responses to "This project increased my interest in my major." 29 responses indicated Chemical Engineering major.

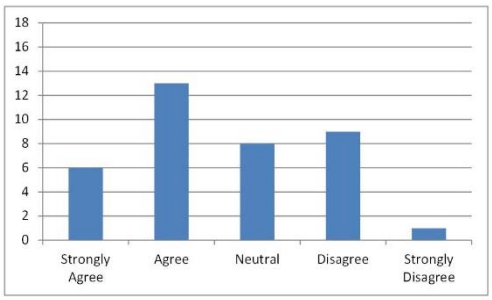


Figure 6. Student responses to "Sufficient background to Arduino programming language was provided to collect data."

Additional changes would include allotting more time in the machine shop to complete the preparation of materials and assembly of the SPBs; two 1-hour sessions were insufficient in many cases. Early on it was identified that wood screws failed to secure components to MDF board, so a switch was made to quarter inch plywood. Subsequent observations suggest three-eighths plywood would be more successful. Mini USB cables connected the solar panel to the microprocessor, but initial testing indicated unregulated voltage levels generated by the solar panel exceeded the specified 5V microprocessor input voltage at the mini USB port. A barrel jack cable retrofitted to the USB cable connected the solar panel to the RedBoard barrel jack which could tolerate variations from 7-15V. Initial cost estimates predetermined that students would share materials. Thirty of a possible forty units were completed. Suggestions that a lab fee attached to future classes to allow each student to make and keep their own SPB were met with mixed reviews. More in-class time needs to be spent to ensure students wire their chosen sensors properly, and are able to collect meaningful data.

The opportunity to work on a team was unexpectedly the most popular aspect of the project. Ninety-four percent of the respondents who admitted working on a team Agreed or Strongly Agreed that they enjoyed working with classmates on their self-selected teams of two to four students (seven chose to work alone). Student oral presentations indicated the cohesive nature of the formed teams. In order to accommodate twenty-three presentations over two course periods, students were allotted defined amounts of time to present. Students, formally dressed, delivered

organized, well-researched, and obviously practiced PowerPoint presentations. Student teams sought ‘public speaking’ tutors available on campus to critique their delivery. Many students invested significant amounts of time to gather data under harrowing conditions—interactions between presenters on teams indicated the level of comradery developed among classmates unknown to each other at the start of the semester.

Conclusions

Empowering freshman to build and use solar-powered microprocessor systems to collect data from sensors was considered an overall success. Chemical engineering freshmen were introduced to campus machine shop facilities five semesters earlier than previously practiced, and enabled to engage in remote data collection in student-generated contexts around topics of green energy power generation, application, and environmental effects. Students utilized multiple resources across campus to complete data collection, perform background research, and prepare for their oral presentations. The project not only introduced students to aspects of data collection, calibration and regression analysis needed for their major, it also provided freshmen chemical engineering students the opportunity to work with each other and collaborate on teams as demonstrated through professional, informative and organized final project reports. Any enduring impact of the workshop exercise on students’ performance in their upper level laboratory and process control courses will be evaluated as these students enter those upper-level courses.

Acknowledgments

The authors wish to thank Dean Wendi Heinzelman and Assistant Dean James Zavislan for their support and for underwriting the major costs of this project, and to Sandra Willison of the Dept of Chemical Engineering for ‘finding’ the remaining needed funds. Special thanks to Prof. Mitchell Anthamatten for enabling and encouraging his Green Energy course recitation time to be used for this workshop; to Larry Kuntz for sharing his electrical knowledge for the initial designs; to Moriana Garcia for assistance and training on library resources for the students, and to Prof. F. Doug Kelley for promoting hands-on experimental work with DAQ systems in the labs.

References

1. Bernstein, A.; Gaudet, C.; Lyons, L.; Sherman, D.; Shikari, A.; Stewart, K.; Stilson, B.; Ward, J.; Zabrodsky, A., Curriculum meeting with the CHE Department Chair, Professors and Hajim School of Engineering and Applied Science Deans Robert Clark and James Zavislan. Monfredo, R., Ed. April 18, 2014.

2. Moor, S. S.; Saliklis, E. P.; Hummel, S. R.; Yu, Y.-C., A Press RO System: An Interdisciplinary Reverse Osmosis Project for First-Year Engineering Students. *Chemical Engineering Education* **2003,** *37* (1), 38-44.

3. Farrell, S.; Hesketh, R. P.; Savelski, M., A Respiration Experiment to Introduce ChE Principles. *Chemical Engineering Education* **2004,** *38* (No. 3, Summer), 7.

4. Coronella, C., Project-based learning in a first-year chemical engineering course: Evaporative cooling. In *113th Annual ASEE Conference and Exposition, 2006, June 18, 2006 - June 21, 2006*, American Society for Engineering Education: Chicago, IL, United States, 2006.

5. Menicucci, J.; Duffy, J.; Palmer, B., Hands-on introduction to chemical and biological engineering. In *114th Annual ASEE Conference and Exposition, 2007, June 24, 2007 - June 27, 2007*, American Society for Engineering Education: Honolulu, HI, United States, 2007.

6. Slater, C. S.; Hesketh, R. P.; Fichana, D.; Henry, J.; Flynn, A. M.; Abraham, M., Expanding the frontiers for chemical engineers in green engineering education. *International Journal of Engineering Education* **2007,** *23* (2), 309.

7. Crippen, K. J.; Boyer, T. H.; Korolev, M.; de Torres, T.; Brucat, P. J.; Chang-Yu, W., Transforming Discussion in General Chemistry With Authentic Experiences for Engineering Students. *Journal of College Science Teaching* **2016,** *45* (5), 75-83.

8. Prince, M.; Felder, R., The Many Faces of Inductive Teaching and Learning. *Journal of College Science Teaching* **2007,** *36* (5), 14-20.

9. Felder, R. M., Why are you teaching that? *Chemical Engineering Education* **2014,** *48* (3, Summer), 131-132.

10. Blumenfeld, P. C.; Soloway, E.; Marx, R. W.; Krajcik, J. S.; Guzdial, M.; Palincsar, A., Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational psychologist* **1991,** *26* (3-4), 369-398.

11. Ryan, R. M.; Deci, E. L., Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* **2000,** *25* (1), 54-67.

12. Dunlap, R. A., *Sustainable Energy*. Cengage Learning: Canada, 2015.

13. Landis, R. B., *Studying Engineering: A Road Map to a Rewarding Career*. Discovery Press: Los Angeles, CA, 2013.

Appendix A

Bill of Materials for 1 Solar Panel Box

Item	Source	Item #	Unit Price [1]
Sparkfun RedBoard programmed with Arduino	SparkFun.com	DEV-12757	17.95
USB Mini-B Cable, 6ft	SparkFun.com	CAB-00598	1.95
ALLPOWERS 2.5W 5V/500mAh Mini Encapsulated Solar Cell Epoxy Solar Panel DIY Battery Charger Kit for Battery Power 130x150mm	Amazon.com		7.99
7" Jumper Wires for RedBoard M/M - 30AWG (30 pack)	SparkFun.com	PRT-11026	1.95
Mini breadboard 3.29 x 2.15 x 0.33" Fits wire sizes 29-20AWG	Newark.com	99W1759	2.51
DC Power Connector (barrel jack) Plug, male, cable mount, 2.1mm*	Newark.com	#34C3282	0.75
Red 22 awg wire for barrel jack*, sensors, 25ft	SparkFun.com	PRT-08865	2.95
Black 22 awg wire for barrel jack*, sensors, 25ft	SparkFun.com	PRT-08867	2.95
Sterlite 6 Qt shoebox	Big Lots		1.00
1/4" Mini-Panel-Meter	MPJA.com	32320 ME	2.59
		Preliminary total	42.59

*Pre-wired barrel jacks may also be available

Sensors--see Table 1 Additional

Assorted Parts

Machine shop: plywood, size 1 screws for panel meter, size 2 screws for microprocessor, nuts, solder, glue gun sticks — Additional

Breakout board wiring: colored wires (can also use Sparkfun jumper wires), solder; resistors for circuits — Additional

[1] 2016 list price; may depend upon quantity discount

Appendix B

recorder.ino

```
/* Code available upon request. Email author. */

/* Code to be used for Solar Panel Boxes, written by
   David J. Schinsing, 2016. Different pieces of code are
   present to accommodate different operating modes i.e., either
   in Solar Panel data collection mode or in test mode, for continuous
   data presented on the Serial Monitor of this Arduino sketch. Default
   operating mode is for continuous data, plugged into computer (Thor
   Mode 1). To activate different modes, follow instructions provided
   separately.

   To enable short-term data storage on the RedBoard while powered by a
   solar panel, the system utilizes the EEPROM library.  Data is stored
   or released depending upon the placement of a jumper wire, called
   here a 'mode jumper'. The mode jumper applies a ground to a digital
   pin. This is the active state: mode jumper in, and the pin reads LOW.
   The pin is configured as INPUT_PULLUP so when the mode jumper is
   removed, the pin rises to HIGH, which is the inactive state.
*/

#include <Wire.h>

#define MODE_JUMPER 2   // mode jumper is digital pin 2
#define LED_PIN 13      // LED on digital pin 13

/* The number of samples and the sample rate is defined here. Consider
   the sensor, how fast does the physical property change?, what is
   the duration of the event you're measuring?
*/
#define N 59            // number of samples
#define TICK 900000     // sample every 15 minutes

/* How many analog channels do you need? The SparkFun RedBoard has
   6 analog inputs, or channels. Set CHANNELS to how many you want. I'll
   read A0 first, then A1, and A2, ... up to however many you need.
*/
#define CHANNELS 1      // maximum of 6 channels

/* Are you using the TMP102 sensor? If not, comment this out. */
//#define TMP102
/* Are you using the RHT03 sensor? If not, comment this out. */
#define RHT03

/* A word about memory: When recording, data from the analog channels
   is saved in EEPROM. EEPROM is limited to 1024 bytes on the
   ATmega328/P. Each sample of data takes 2 bytes. Do the math. Or
   read the table:

   CHANNELS(+TMP102)   ... maximum N (samples)
          1      ...        512
          2      ...        256
          3      ...        170
          4      ...        128
          5      ...        102
          6      ...         85

   Just make sure N*CHANNELS <= 512. I'm not going to check in the code.
   And don't set N or CHANNELS to zero. That's dumb, and I'm not going
   to test for that, either.
*/

#include <EEPROM.h>

// Arduino runs this first, and only once

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(MODE_JUMPER, INPUT_PULLUP);

  ACSR = 0;

#ifdef TMP102
  tmp102_setup();
#endif
#ifdef RHT03
  rht03_setup();
#endif
#ifdef DS18B20
  ds18b20_setup();
#endif
}

void flashLED() {
  digitalWrite(LED_PIN, HIGH);
  delay(150);          // ms delay
  digitalWrite(LED_PIN, LOW);
  delay(150);
}

/* Record Mode: Because we got here, the mode jumper is
   in. Data collection is armed, waiting for mode jumper
   removal.
*/
void record() {
  int n, channel;
  int address = 0;      // EEPROM address starts at 0 (Arduino provided a nice
EEPROM interface!)
  int value;
#ifdef RHT03
  int temp;
#endif

  // wait for the mode jumper to be removed
  while (digitalRead(MODE_JUMPER) == LOW)   // while the jumper is installed, ...
    flashLED();                             // ... blink

  delay(TICK / 2);

  for (n = 0; n < N; n++) {       // for N samples, ...
    for (channel = 0; channel < CHANNELS; channel++) {    // for each CHANNEL,
...
      value = analogRead(A0 + channel);   // ... read sensor, ...
      EEPROM.put(address, value);      // ... and stuff it in the EEPROM
      address += sizeof(value);        // next address
    }
#ifdef TMP102                          // for TMP102, ...
    value = tmp102();                  // ... read sensor, ...
    EEPROM.put(address, value);        // ... and stuff it in the EEPROM
    address += sizeof(value);          // next address
#endif
#ifdef RHT03
    rht03(&temp, &value);              // ... read sensor, ...
    EEPROM.put(address, temp);         // ... and stuff it in the EEPROM
    address += sizeof(temp);           // next address
    EEPROM.put(address, value);        // ... and stuff it in the EEPROM
    address += sizeof(value);          // next address
#endif

#ifdef DS18B20
    value = ds18b20();                 // ... read sensor, ...
    EEPROM.put(address, value);        // ... and stuff it in the EEPROM
```

```
    address += sizeof(value);          // next address
#endif
    flashLED();                    // look alive!
    delay(TICK);                   // ms delay
  }
}

void playback() {
  int n, channel;
  int channels = CHANNELS
#ifdef TMP102
                 + 1
#endif
#ifdef RHT03
                 + 2
#endif
#ifdef DS18B20
                 + 1
#endif
                 ;
  int address = 0;    // EEPROM address starts at 0

  Serial.begin(9600); // initialize
  while (!Serial);    // wait for port to open

  Serial.print("\n\nSAMPLE");
#if CHANNELS > 0
  Serial.print(", A0");
#endif
#if CHANNELS > 1
  Serial.print(", A1");
#endif
#if CHANNELS > 2
  Serial.print(", A2");
#endif
#if CHANNELS > 3
  Serial.print(", A3");
#endif
#if CHANNELS > 4
  Serial.print(", A4");
#endif
#if CHANNELS > 5
  Serial.print(", A5");
#endif
#ifdef TMP102
  Serial.print(", TMP102");
#endif
#ifdef RHT03
  Serial.print(", RHT03(temperature), RHT03(humidity)");
#endif
#ifdef DS18B20
  Serial.print(", DS18B20");
#endif
  Serial.print("\n");

  for (n = 0; n < N; n++) {       // for N samples, ...
    Serial.print(n);
    Serial.print(", ");
    channel = 0;
    do {
      int value;
      EEPROM.get(address, value);
      address += sizeof(value); // next address
      Serial.print(value);
      channel++;
      if (channel == channels) break;
      Serial.print(", ");
    } while (1);
    Serial.print("\r\n");
  }
}

// THOR Mode: use this for testing sensors, or setting up your serial
communication
#define THOR 0
//#define THOR 1        // don't check MODE_JUMPER, just read and display
continuously
void thor() {
  int value;
#ifdef RHT03
  int temp;
#endif

  Serial.begin(9600); // initialize
  while (!Serial);    // wait for port to open

  do {
    int channel;
    Serial.print("---, ");
    channel = 0;
    do {
      int value;
      value = analogRead(A0 + channel);    // ... read sensor, ...
      Serial.print(value);
      channel++;
      if (channel == CHANNELS) break;
      Serial.print(", ");
    } while (1);
#ifdef TMP102
    value = tmp102();   // left as an exercise to the user to multiply these
numbers by 0.0625 degrees C (TMP102 only!)
    Serial.print(", ");
    Serial.print(value);
#endif
#ifdef RHT03
    rht03(&temp, &value);
    Serial.print(", ");
    Serial.print(temp);
    Serial.print(", ");
    Serial.print(value);
#endif
#ifdef DS18B20
    value = ds18b20();
    Serial.print(", ");
    Serial.print(value);
#endif
    Serial.print("\r\n");
    delay(500);    // slower
  } while (1);
}

// Arduino runs this second. This code never returns. See the infinite loop at
the
// bottom. Is this atypical Arduino code? Probably.

void loop() {
  delay(1000);
#if THOR
  thor();           // never to return ...
#endif

  if (digitalRead(MODE_JUMPER) == LOW) {   // is the jumper installed?
    record();       // mode jumper in, ... record mode
  } else {
    playback();     // mode jumper out, ... playback
  }
```

```
  // turn LED on:
  digitalWrite(LED_PIN, HIGH);   // done recording or playing back, ... turn LED
on
  while (1);      // power-hungry stall (I wonder if Arduino has a sleep mode?)
}
```

rht03.ino

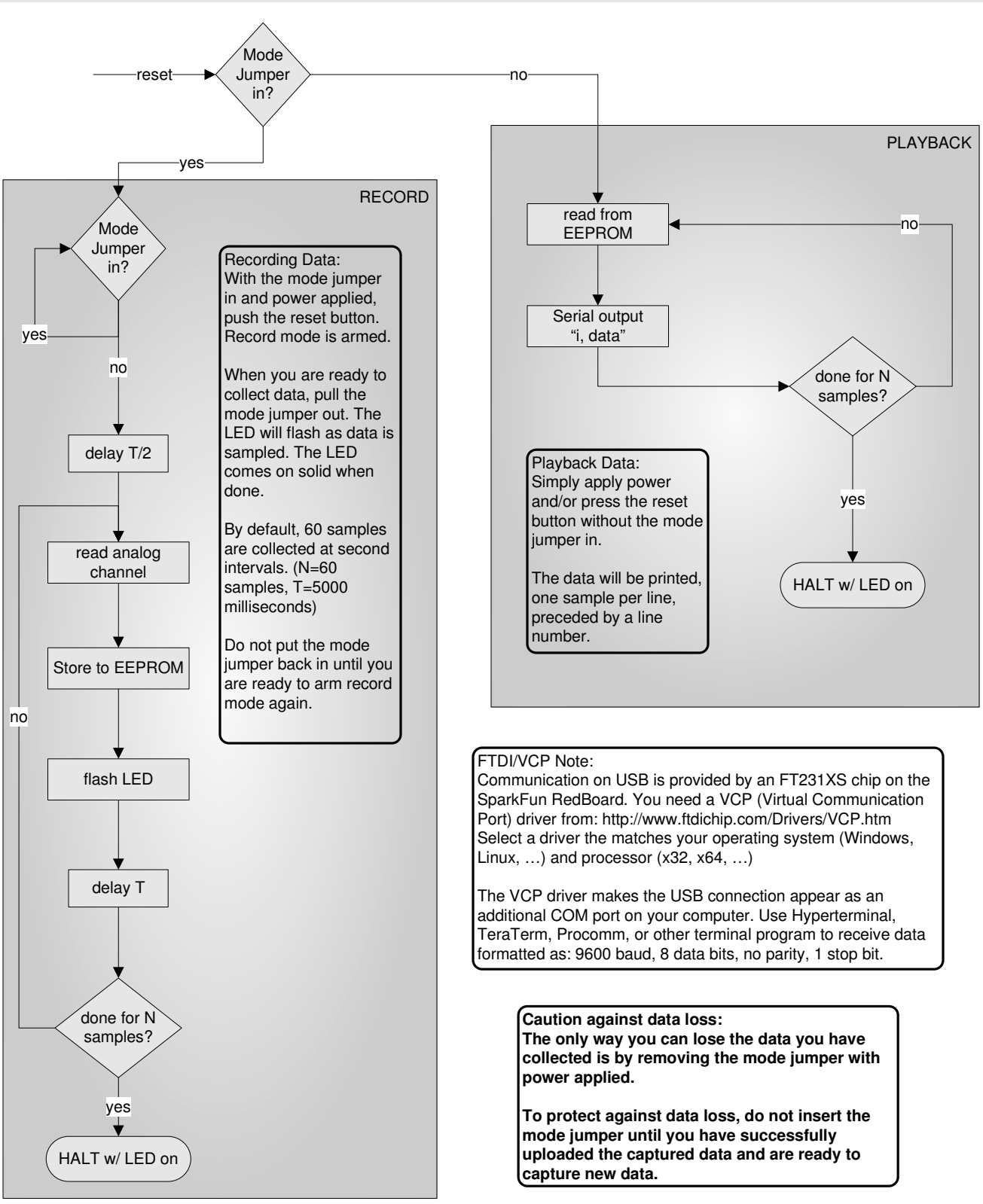
```
#define RHT03_1 9
#define RHT03_2 8
#define RHT03_4 7

void rht03_setup()
{
  TCCR1A = 0;
  TCCR1B = 0x82;   // /8 prescaler

  pinMode(RHT03_1, OUTPUT);
  digitalWrite(RHT03_1, HIGH);
  pinMode(RHT03_2, INPUT_PULLUP);
  pinMode(RHT03_4, OUTPUT);
  digitalWrite(RHT03_4, LOW);
}

static int low()   /// return the time it went low
{
  while (digitalRead(RHT03_2) == HIGH);
  return TCNT1;
}

static int high()   // return the time it went high
{
  while (digitalRead(RHT03_2) == LOW);
  return TCNT1;
}

int rht03(int *temp, int *humid)
{
  int i;
  int x, y;
  int data[5];

  pinMode(RHT03_2, OUTPUT);
  digitalWrite(RHT03_2, LOW);
  delayMicroseconds(2000);   // 1000 uS minimum
  pinMode(RHT03_2, INPUT_PULLUP);

  delayMicroseconds(50);   // get away from the noise

  y = low();
  x = high();
  y = low();
  for (i = 0; i < 5; i++) {
    x = high();
    y = low();
    if (y - x > 100) data[i] = 0x80; else data[i] = 0;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x40;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x20;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x10;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x08;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x04;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x02;
    x = high();
    y = low();
    if (y - x > 100) data[i] |= 0x01;
  }

  if (((data[0] + data[1] + data[2] + data[3]) & 0xff) != data[4]) {
    *temp = 0^-1;
    *humid = 0^-1;
  } else {
    *temp = data[2] << 8 | data[3];
    *humid = data[0] << 8 | data[1];
  }
}
```

tmp102.ino

```
#include <Wire.h>

#define TMP102_I2C_ADDRESS 0x48 /* This is the I2C address for our chip.
  This value is correct if you tie the ADD0 pin to ground. See the datasheet for
some other values. */

void tmp102_setup() {
  Wire.begin(); // start the I2C library
}

signed int tmp102() {
  signed short firstbyte, secondbyte; //these are the bytes we read from the
TMP102 temperature registers

  Wire.beginTransmission(TMP102_I2C_ADDRESS); //Say hi to the sensor.
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(TMP102_I2C_ADDRESS, 2);
  Wire.endTransmission();

  firstbyte = Wire.read();
  /*read the TMP102 datasheet - here we read one byte from
    each of the temperature registers on the TMP102*/
  secondbyte = Wire.read();
  /*The first byte contains the most significant bits, and
    the second the less significant */

  return (firstbyte << 8 | secondbyte) >> 4;
}
```

Appendix C: Instructions for Collecting Data

Data Record and Playback on a SparkFun RedBoard



FTDI/VCP Note:
Communication on USB is provided by an FT231XS chip on the SparkFun RedBoard. You need a VCP (Virtual Communication Port) driver from: http://www.ftdichip.com/Drivers/VCP.htm Select a driver the matches your operating system (Windows, Linux, …) and processor (x32, x64, …)

The VCP driver makes the USB connection appear as an additional COM port on your computer. Use Hyperterminal, TeraTerm, Procomm, or other terminal program to receive data formatted as: 9600 baud, 8 data bits, no parity, 1 stop bit.

Caution against data loss:
The only way you can lose the data you have collected is by removing the mode jumper with power applied.

To protect against data loss, do not insert the mode jumper until you have successfully uploaded the captured data and are ready to capture new data.

Appendix D Sample Handout for Sensor wiring and use

2.2” Flex Sensor

FlexSens _____

Sparkfun.com SEN-10264

Description: A simple flex sensor 2.2" in length. As the sensor is flexed, the resistance across the sensor increases.

Note: Please refrain from flexing or straining this sensor at the base (where it attaches to red and black wires). It can be broken, and doesn’t provide a reliable signal at this location.

Hardware/Software needs:

This sensor has two wires: one red wire, which needs to be connected to a voltage source (ie. 5V) and a black wire which needs to be connected to ground (GND) via a voltage dividing resistor.

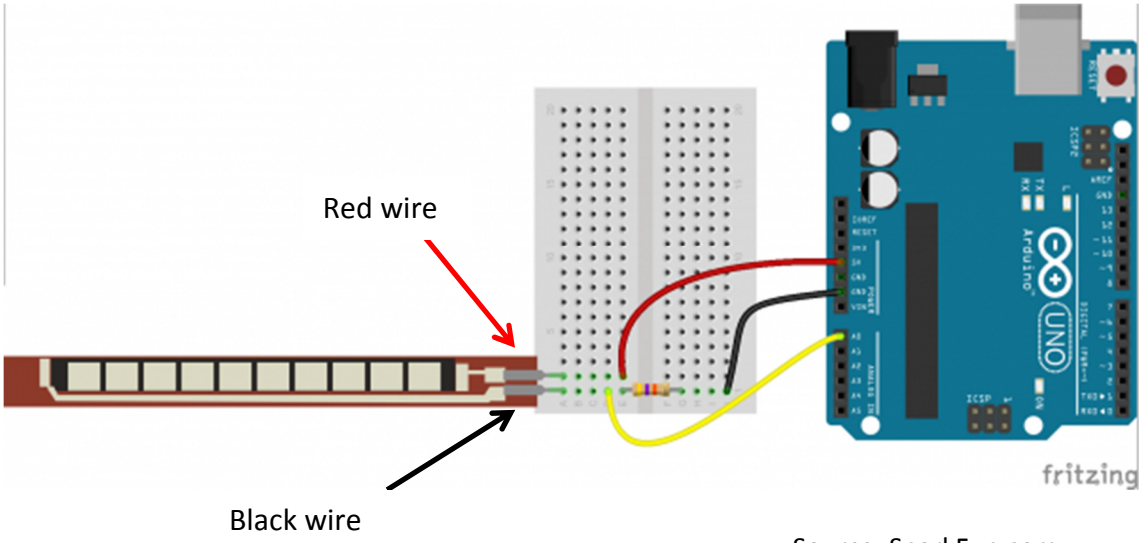
A 47K ohm resistor and jumper wires are needed. See the diagram below for wiring between your RedBoard (essentially the same as the Uno Board pictured below), the breadboard and the sensor. See also the Sparkfun website: https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide#example-program

Red wire to breadboard to 5V via jumper wire

Black wire to breadboard to colored jumper wire to A0 and to 47KΩ resistor to ground.

Utilizing the RecorderA0 program in the test mode will enable you to determine how the values change from a flat to fully flexed configuration

The Sparkfun site also suggests uses for the sensor.



Source: SparkFun.com