**Senior Design Project:**

**Self-Powered Wireless Sensor Node for Environmental Monitoring**

Alec May

Curtis Nelson Jr.

Noel Villegas Jr.

Alvin Yang

Kevin Yang

April 29, 2016

Instructor: Radian Belu

Section: EE A438

**ABSTRACT**

The goal of this project was to design and build a self-powered wireless sensor node (WSN) for environmental monitoring. It is designed to measure temperature, pressure, humidity, and light intensity in harsh arctic environments like Alaska. We were able to find parts that can operate in as cold of temperatures as -25℃ while also meeting our low voltage requirement of 5 volts or lower. We have constructed a test WSN circuit that can successfully gather enough solar energy to operate our circuit and charge a battery for night operation. Our circuit can also successfully power sensors to measure environmental data while storing it on a SD card for future use. This project will continue next semester. Another undergraduate team will implement a micro wind turbine, RF transceiver, and an enclosure for the circuit. There are many challenges when implementing WSNs which must be investigated in order to acquire maximum benefits from the WSN aerospace applications. The success of our project may answer in part to some of these questions and challenges.

**TABLE OF CONTENT**

Background………………………………………………….…………………………… Page 3

Project Goal and Objective……………………………….....…………………………… Page 3

Introduction………………………………………………..…..…………………...……. Page 4

Design Methodology………………………..……………..……………………….……. Page 5

Analysis………………….………...…….……….……….…………………………...…. Page 7

Results & Discussion………...………….…………………………………………….... Page 10

Conclusion………………………………...……………………….…………………… Page 12

References…………………….…………..…………………………………………….. Page 13

Appendix A…………………………….….………………….………………………… Page 14

Appendix B……………………………………………………………………………… Page 22

**BACKGROUND**

There are many situations where data needs to be collected for long periods of time and it would not be cost effective to have an employee collect the data or have to perform routine maintenance on a data collector. This concern has led to the invention of the wireless sensor node (WSN). WSNs have been designed to stay in harsh environments to collect and transmit data back to the user with minimal physical interaction. The WSN consist of many components. There are nodes, gateways, and programmable software that can interact with the system to make it perform certain tasks. The device will have built in sensors that will measure and store data from its surrounding environment at certain time intervals throughout the day. These sensors can include light, humidity, temperature, pressure, etc. The WSN can send the data collected wirelessly usually by transmitting via RF radio transceiver. Since the WSN is made to collect data from its surrounding environment, the WSN must be built to withstand the weather conditions in the deployment location. That is why the components of the WSN must have specific tolerances and temperature operating ranges. The wireless sensor does require a long operating life, so practical energy harvesting methods, proper energy storage devices, along with a functioning power switching unit will ensure a long life span.

**GOAL AND OBJECTIVE**

The goal of our senior design project is to construct a WSN that can efficiently harvest its own energy then measure and record the temperature, humidity, light intensity, and pressure of the surrounding environment. It must also be able to withstand cold weather conditions here in Alaska. We built our system very robust by: utilizing solar energy harvesters, a super capacitor and a rechargeable battery to extend the life of our system, and an efficient data collecting scheme that utilizes the least amount of energy. This project is planned to be completed in two phases which consist of two different groups of students. The first team will be led by Alec May and then the project will be handed over to Samuel Cragle. The first team has been tasked with constructing and testing the WSN. This includes assembling an energy harvesting system, energy storage and transfer system, and the data collection and storage system. The second team will then be responsible for transmitting data from the node to a mother unit, constructing a weatherproof case, designing a unified printed circuit board (PCB) for the entire system, and refining the project as they see fit.

**INTRODUCTION**

WSNs have found a wide range of applications, including among others: structural and biomedical health, industrial process, environmental and habitat monitoring, and military surveillance. The project will have broad application throughout the Alaska, U.S. and is in line with the goals of NASA research plans to improve the monitoring and sensing capabilities and applications. Wireless sensors and sensor networks, an emerging technology area has found many applications within the aerospace industry. In addition, the use of WSN technology may fit very well in a variety of spacecraft applications, while NASA is looking to industry and universities to develop some of these new applications. It has long been known that eliminating wiring and wiring harnesses is reducing the mass of vehicle health monitoring systems, and transitioning from wired to wireless systems is an excellent way to reduce the amount of wiring.

For our Senior Design project, we collaborated with Alec May’s project for NASA, building a wireless sensor node (WSN) that is powered by renewable energy. Solar and wind energy harvesting techniques are the main focus. Currently, the WSN will gather energy from an array of solar panels. There will be a solar panel, micro wind turbine, super capacitor, and a rechargeable lithium ion battery supplying energy to the system. The Sensors need to measure temperature, humidity, pressure, and light. The goal of the project is to have the WSN manage the power between gathering and storing the power with the help of a microcontroller in the circuit. This project for NASA will take a few semesters to finish. The task for our group is to start a skeleton model, so that the following group next semester can continue the completion of the system.

Our group will be working with the solar panels, rechargeable battery, super capacitor, microcontroller, and sensors. We will hold off on the wind turbine for now due to time constraints, (The only micro wind turbine small and durable enough to meet our needs will arrive after this semester and was manufactured in Italy [Refer to Figure 16]) and we will have the next semester’s group implement the RF transceiver and construct the weatherproof housing for the device [12].

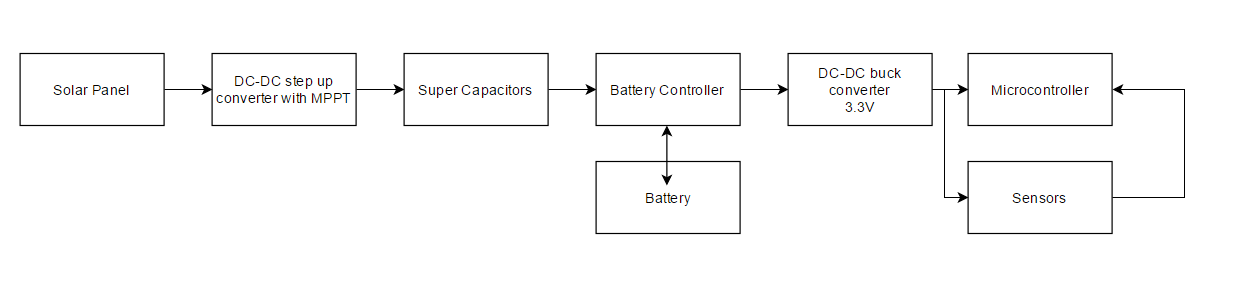


Figure (1): Block Diagram of the WSN

**DESIGN METHODOLOGY**

When we began our project we met with our client Dr. Radian Belu where he presented us with a block diagram of a typical WSN. He also gave us many articles and papers from other design groups who had constructed unique WSNs. We then began to research how WSNs functioned and then began to construct a table for each individual part. The table specified such information as input voltage, current, power consumption, temperature, size, shipping time, etc. Below is a table that shows the parts we selected. Following Table 1 is a brief explanation of why we selected them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PART** | **MANUFACTURER** | **PRICE** | **QUANTITY** | **TOTAL** |
| Solar Panel | Radio Shack | $10.20 | 4 | $40.80 |
| Super Capacitor | Panasonic | $3.91 | 1 | $3.91 |
| Battery manager | Linear Technology | $4.95 | 1 | $4.95 |
| Battery | SparkFun | $12.95 | 1 | $12.95 |
| Buck-Boost Converter | SparkFun | $14.95 | 2 | $29.90 |
| MicroSD Card Reader | SparkFun | $3.95 | 1 | $3.95 |
| Temperature/Humidity/Pressure Sensor | Sparkfun | $19.95 | 1 | $19.95 |
| Photosensor | Seeed Studio | $9.90 | 1 | $9.90 |
| Microcontroller | Arduino | $9.95 | 1 | $9.95 |
| Arduino Programmer | Arduino | $14.95 | 1 | $14.95 |
| Real Time Clock | Donop | $2.49 | 1 | $2.49 |
|  |  |  |  | **$153.70** |

Table 1: Components of WSN

Solar Panel

The solar panels that we chose were the 1.5 watts Solar Panel 4.5 volts from Radio Shack. It has dimensions of 5.11 by 3.3 inches, which is an ideal size for our design. We used four solar panels in parallel to meet the voltage requirement for the boost converter. The required voltage range is 0.3V - 7.0 volts.

Super Capacitor

We chose to use a 1.5 farads 5.5 volts supercapacitor to power our system. This capacitor should have the ability to supply enough current to power our system and should be able to charge quickly.

Battery Charger & Battery

The Polymer Lithium Ion Rechargeable Battery - 2000mAh from SparkFun was selected as our battery and the Linear Technology’s Li-Ion/Polymer Shunt Battery Charger System with Low Battery Disconnect (PRT LTC4071) was chosen to be our power switching unit (hereafter referred to as ‘battery charging unit’ or ‘battery management unit’). We were drawn to this battery because of its high energy density and life cycles. It also runs at 3.7 volts which can work in conjunction with our battery charger that has a floating voltage of 4.0 volts. Generally you should not overcharge a battery because it reduces battery life and if charged too much can cause the battery to explode. However, the battery charger has an internal shunt that will activate at 4.2 volts so we should not have to worry about explosions. Together, these two parts will allow our system to draw energy from just the super capacitor during day time operation. The energy supplied by the super capacitor in conjunction with the battery management unit will recharge the battery such that it is ready to use for night operation. We expect to extend the life time of our system compared to just using a battery or just using a supercapacitor.

Buck Boost Converters

We chose the Ultra-low power boost converter harvester from Texas instruments to use to implement our design. The boost converter will manage the voltage between the solar panels and the supercapacitors, taking in voltages as low as 330 millivolts and boosting it to 5.0 volts. A second buck boost converter was used after the battery management to supply the Microcontroller and sensors with 3.3 volts.

Microcontroller

The microcontroller we selected was the Arduino Pro Min 328. There were 2 models of this microcontroller available and we chose the one with 3.3 volts. This particular was chosen because it met our main requirements of an input voltage of 3.3 volts, operating temperature between -40 and 85 ℃, and its small size. Another reason we chose to use an Arduino microcontroller was due to the various libraries available. Having these libraries made the coding tasks much simpler.

Humidity/Temperature/Pressure Sensor

We decided to use the SparkFun BME280 Atmospheric Sensor Breakout to measure the humidity, temperature, and pressure of the environment. The BME280 had all three sensors incorporated onto a very small integrated circuit and operated at the same 3.3 volts as the rest of our components along with a low supply current. It was also able to perform in a temperature range of -40 to 85 ℃ which made it suitable for the Alaska climate.

Photo Sensor

In order to be able to measure the amount of sunlight our WSN was receiving, we purchased the Grove SI1145 Sunlight Sensor. This sensor operated at temperatures as low as -40 ℃ and was able to use an input voltage of 3.3 volts making it usable in our system. It didn’t consume a lot of power and was already an integrated circuit so we didn’t have to add any external resistors and capacitors.

MicroSD Card Reader

The microcontroller does not have enough internal memory to store all of the various sensor data we are logging. Therefore, a microSD card reader is needed so we can write the data onto an external microSD card. The microSD card reader we chose was the SparkFun microSD transflash breakout chip. The chip was small in size and bought ready to use. All we had to do was simply solder in header pins.

Micro Wind Turbine

In the design of the WSN, we would like to implement multiple energy harvesting techniques. After weeks of researching, we found a 10W Polar wind generator from a company in Italy called Wind Kinetic. Operating in as low of temperatures as -70℃ and producing energy with as low of wind speed as 4.5 mph, this component offers great outputs in Alaskan arctic weather that will meet our systems current and voltage specifications [Refer to Appendix A- Figure 16].

RF Transmitter

We chose an Xbee 1mW wire antenna to implement in our design of the WSN. This product is from Sparkfun. The performance of the Xbee antenna gives us an outdoor RF line-of-sight range of up to 300ft. The wire antenna has a transmitting output power of 1mW, operates within a temperature range of -40℃ to 85℃, requires an operating voltage between 2.8-3.4 volts, and comes on an integrated circuit. All of these specifications are ideal for our system.

While selecting our parts and building our circuit we ran into a few problems. We originally desired most of our components to be surface mount because in general, they're significantly smaller than through hole or IC chips and they consume less power. We found that many of these parts were too difficult for us to solder by hand using the tools available at UAA. Many parts were then replaced with a through hole counterpart. However, we purchased multiple battery management surface mount components because we couldn’t find a through hole equivalent. We then used this as an opportunity to construct a printed circuit board (PCB) and learned advanced soldering techniques from an experienced undergraduate soldering machine, Lance Leber.

**ANALYSIS**

Our system had several components that needed to be tested before we connected them together. Most of the measurements for the voltages, currents, and power consumption came from using the Agilent 24410A 6 ½ Digital Multimeter from the electrical engineering labs at the University of Alaska Anchorage. We wanted to make sure that the output voltages and output currents from each component were acceptable for the inputs of the subsequent parts in the system.

Energy Collecting

For this part of our design, it contained the four solar panels and the buck-boost converter. Putting all the solar panels in parallel gave us an output voltage of 4.98 volts with an output current of .334 amperes. After a load was connected to the solar panels, the voltage went down to 0.91 volts.

From the output of the solar panels, it goes into the input of the buck-boost converter. The buck boost-converter has a selectable voltage output of 3 volts and 5 volts with an under voltage protection of 2.6 volts. Refer to Appendix A - Figure 6 for the schematics of the buck-boost converter. We needed to take out the under voltage protection by shorting a capacitor and a resistor on the IC board itself. After taking out the under-voltage protection, we were able to boost the output voltage to 4.99 volts. We connected a Zener diode between the buck-boost and the super capacitor so there is no power flowing back into the solar panels.

Storing and Delivering Energy

This section is comprised of the supercapacitor, battery charging unit, rechargeable battery, and buck-boost converter. We first worked on a design that would draw energy more frequently from the supercapacitor than the battery since supercapacitors have more charge cycles than rechargeable batteries. The original design implemented a CPLD and switches to determine which energy storage device should the circuit pull operating power from. This idea was left out because of time constraints and the need to work with surface mount parts which could put us behind our deadline if we made a mistake soldering them. The design was changed and simplified so we could persevere.  We then waited on the final parts to arrive so that we can attach the system together and begin testing. But, we hope to return to the previous design before the end of the semester and pass on our thoughts to the next group.

From measuring the output voltage from the supercapacitor, we noticed that the battery management would not turn on until the supercapacitor charged up to 4.01 volts. Then the battery management unit will determine whether the battery or supercapacitor will power the sensors.

The output of the battery management unit was giving an output voltage of about 3.8 volts. The voltage then needed to go down because the Arduino and the sensors operated at a low voltage of 3.3 volts. Managing the buck-boost converter just like before, we were able to buck the voltage down to 3.2 volts.

Designing the PCB Board

When designing a Printed Circuit Board (PCB), it is vital that one gains an understanding of a product’s preferred schematic which can be found in the datasheet. First, we had to determine the need for designing a PCB for our system. Fortunately, a few of our components that would have needed a PCB already came on one. We then narrowed our project down to a “test bench” implementation. This meant we wanted each of the components in our system to be able to easily swap without needing to redesign the final circuit. Our idea is, this could provide future improvements by simply swapping to more efficient components with little concern of system faults. We then decided on a need for the design of only one PCB for the battery charging unit because it is a surface mount component.

The next step was to self-teach ourselves how to design a PCB using foreign software. We chose to use Eagle PCB Software to design the typical application schematic for the battery management unit [Refer to Appendix - Figures 13 & 14]. After hours of online tutorials and videos we were able to move to the fabrication of a PCB. Luckily, UAA provided us with an undergraduate (Alejandro Johnson) to aid us while fabricating the PCB. The milling process took roughly an hour to complete. Unfortunately, troubleshooting and burning out one of our battery management chips was a slight deadline set back which we were able to come back from [Refer to Appendix - Figure 15].

Energy Consumption and Programming the Microcontroller

This section consisted of the sensors and microcontroller. We started by compiling a list of options for each sensor type. The four sensors we needed were temperature, pressure, humidity, and light. The requirement for our sensors were an input voltage in the range of 3.3 volts and be able to operate in very cold temperatures. From the list, we selected the best overall option for each sensor and ordered them. In addition to our microcontroller and sensors, two additional components were needed. The first additional component was a SD card reader. The Arduino Pro Min doesn’t have much internal memory and it was mostly used up for our code. By getting a SD card reader, we were able to store our sensor reading onto an external memory card. The second component needed was a real time clock. Our Arduino is only able to stay asleep for a maximum of eight seconds. Using a real time clock allowed our Arduino to stay asleep longer and be able to record the date and time of each measurement.

The block diagram in Figure 2 shows what the Arduino was programmed to do. The first task to do when the Arduino turns on is initialize all of our components. Load the settings onto our sensors and give enough time for the components to start up. As soon as that’s done, the Arduino will go to sleep to save power. At the start of every new hour, the real time clock will send an interrupt to wake up the Arduino. Next, we had the Arduino record a measurement from each sensor every second for five seconds. Then we had it compute the average of the measurements and store it into the SD card. After that, we had the Arduino go back to sleep and restart the cycle. The code written for our Arduino can be seen in Appendix B.

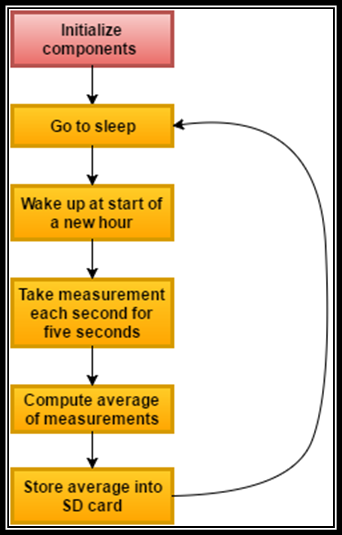


Figure (2): Block Diagram for Arduino Procedure

**RESULTS & DISCUSSION**

Once our system was built and functioning, we let it record data outside for a day. The weather conditions the WSN experienced started off sunny, got cloudy with light rain, and then got sunny again. The graphs in Figures 3 & 4 shows the field test data collected. As you can see, the sensors followed the weather pattern and behaved as we expected them to.

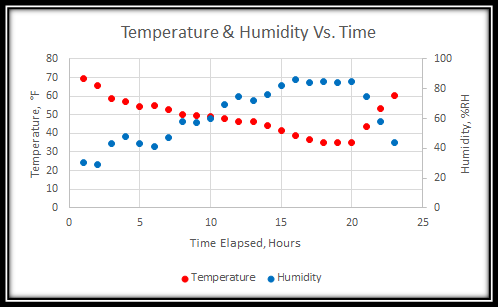


Figure (3): Temperature and Humidity Sample Data Collected

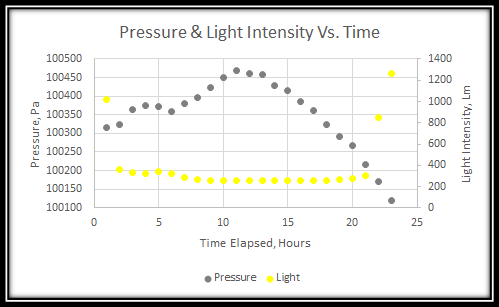
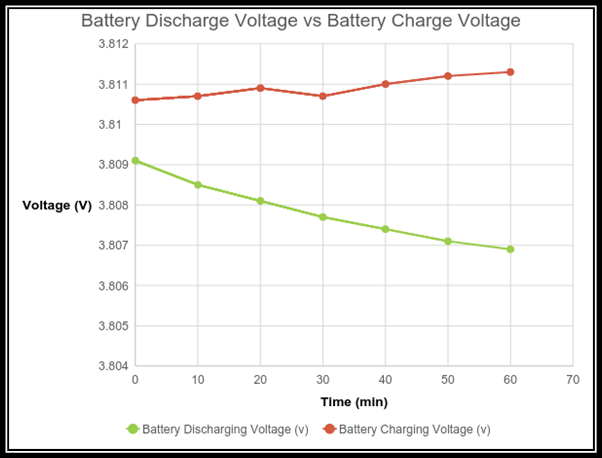


Figure (4): Pressure and Light Sample Data Collected

To check if our system was able to harvest solar energy, we measured the battery’s voltage during a sunny day. The graph below shows the battery’s voltage when it was charging and discharging for an hour duration. To simulate night time for discharging, we unplugged the solar panels from our system. From the graph below, you can see that the battery is in fact charging. However, the battery is discharging more than it is charging making the overall battery voltage to gradually decrease. Hopefully by adding the wind turbine, our battery will be charging more allowing our system to operate longer.

  
Figure (5): Battery Charging and Discharging

To make the WSN last longer, we turned off features on the Arduino that were not being used. Features that were turned off were the analog to digital converter, some timers, and serial communication. Table 2 shows the current consumption of the Arduino with features enabled and disabled. You can observe the current going down with the features disabled making our WSN consume less power.

|  |  |  |
| --- | --- | --- |
|  | Operation  Current (mA) | Sleep  Current (mA) |
| Default | 3.75 | 0.27 |
| Extra Features  Off | 3.41 | 0.19 |

Table 2: Current Consumption of Arduino

**CONCLUSION**

The sensor node we built performed adequately. Our solar panels were able to harness enough energy to start our system as well as charge our battery and supercapacitor. The Arduino controlled when our sensors measured data and recorded the data onto the microSD card. The second phase of this project will be continued by Samuel Cragle. His team will add a micro wind turbine, antenna, protective box, and a printed circuit board. The micro wind turbine will be added to help our system charge overnight. An antenna will allow our sensor node to be able to wirelessly transmit the sensor data to a local computer. The mechanical case will support the micro wind turbine and protect our circuits from rain and snow. Lastly, a printed circuit board will make our system a lot more compact and efficient.

There are many challenges when implementing WSNs which must be investigated in order to acquire maximum benefits from the WSN aerospace applications. Power, volume, and mass must be reduced. Issues such as power supply, electromagnetic interference, ionizing radiation, thermal dissipation, vibration, shock, and harsh environments must be addressed. Despite the challenges, wireless technology offers benefits that cannot be ignored. The success of our project may answer at least in part to some of these questions and challenges.

**REFERENCES**

[1] Chris Schmidbauer, Dave Watson. A Digitally-Controlled and Portable Photovoltaic Power Source. The Department of Electrical and Computer Engineering Drexel University. 2014

[2] Christensen, J. (n.d.). *Arduino Library for Maxim Integrated DS3232 and DS3231 Real-Time Clocks*. Retrieved from https://github.com/JChristensen/DS3232RTC

[3] ForceTronics. (n.d.). *How to Save Power with Arduino*. Retrieved from http://www.instructables.com/id/How-to-Save-Power-with-Arduino/

[4] Linear Technology Corporation. (2010). *Linear Technology LTC4071 – Lithium-Ion/Polymer Shunt Battery Charger System with Low Battery Disconnect.* [PDF document]. Retrieved from http://cds.linear.com/docs/en/datasheet/4071fc.pdf

[5] Matthew Martino, Jordan Varley. A Wireless Sensor Node Powered by a PV/SuperCapacitor/Battery Trio. The Edward S. Rogers Sr. Department of Electrical and Computer Engineering University of Toronto. 2012

[6] R. Belu, Wireless Sensor Networks: Energy Harvesting Methods, in Encyclopedia of Energy Engineering & 4Technology (Online) (Ed: Dr. Sohail Anwar), 2014(DOI: 10.1081/E-EEE-120048435 /21 pages.).

[7] Silicon Labs. (2013). Grove-Sunlight Sensor(September 2, 2015). Retrieved from the SeeedStudio Wiki: http://www.seeedstudio.com/wiki/Grove\_-\_Sunlight\_Sensor

[8] SparkFun. (2015). *SparkFun BME280 - SparkFun Atmospheric Sensor Breakout.* Retrieved from https://www.sparkfun.com/products/13676

[9] SparkFun. (2015). *SparkFun PRT-10255 ROHS - LiPower - Boost Converter.* Retrieved from https://www.sparkfun.com/products/10255

[10] *SparkFun microSD Transflash Breakout*. (n.d.). Retrieved from https://www.sparkfun.com/products/544

[11] *Using the Arduino Pro Min 3.3V.* (n.d.). Retrieved from https://learn.sparkfun.com/tutorials/using-the-Arduino-pro-mini-33v

[12] Wind-Kinetic. (2014). *WindKinetic  – The Polar Range Advanced Vertical Axis Wind Generators.* [PDF document]. Retrieved from http://wind-kinetic.com/media/newpdf/WindKinetic\_Polar\_Range\_VAWT\_brochure.pdf

**APPENDIX A**

BUCK BOOST CONVERTER

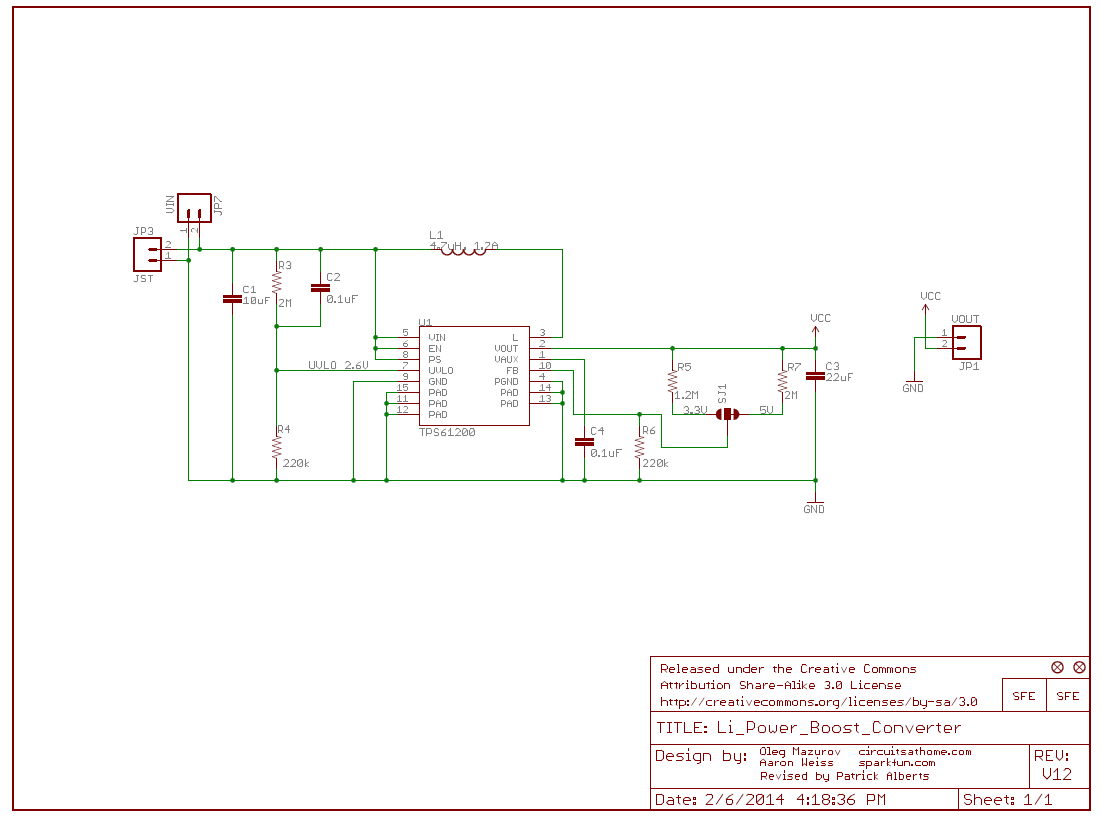


Figure (6): Lithium-Ion Buck Boost Converter Schematic

ARDUINO

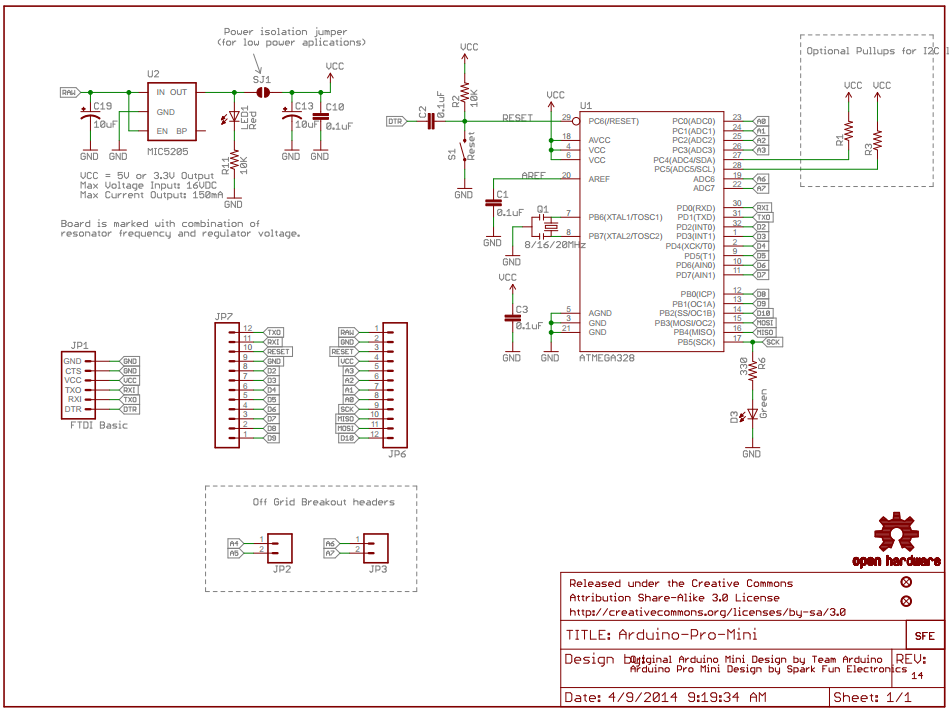
****

Figure (7): Arduino Pro Mini 328 Schematic

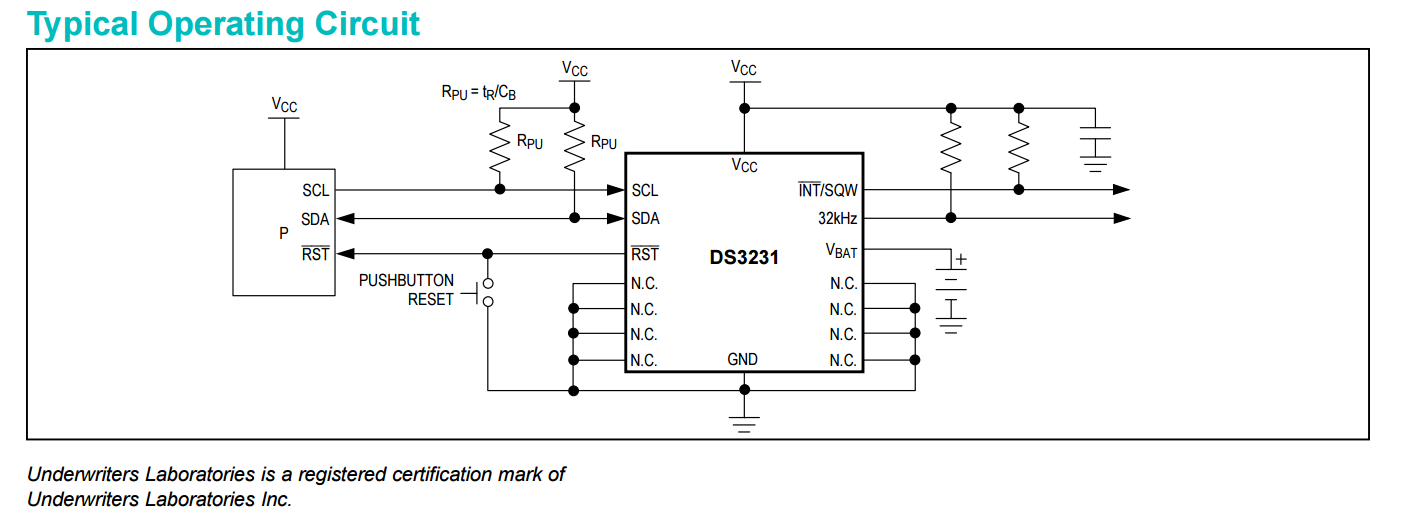


Figure (8): DS3231 Real Time Clock Schematic

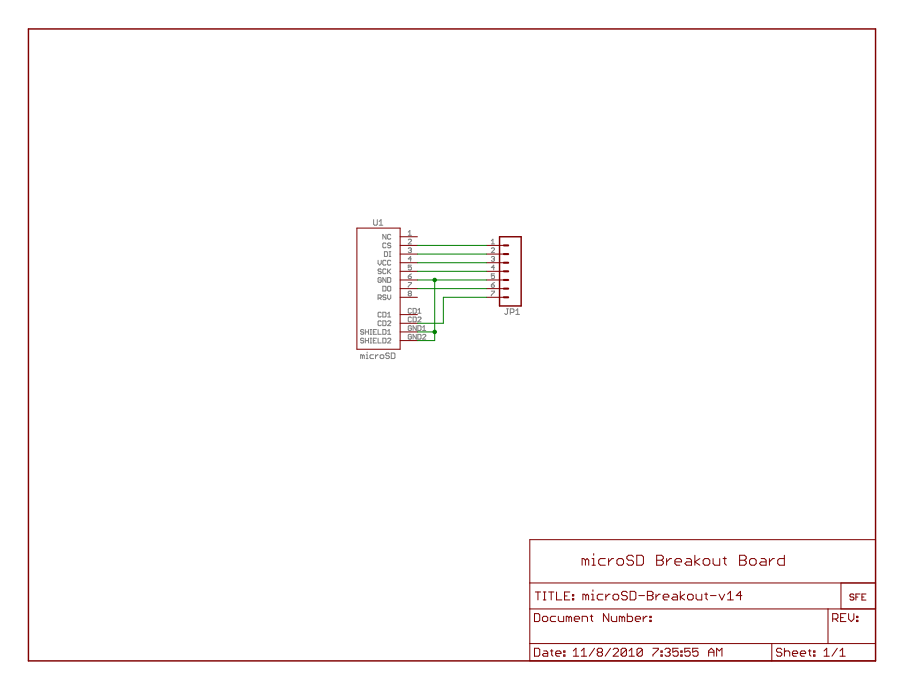


Figure (9): MicroSD Card Reader Schematic

SUNLIGHT SENSOR

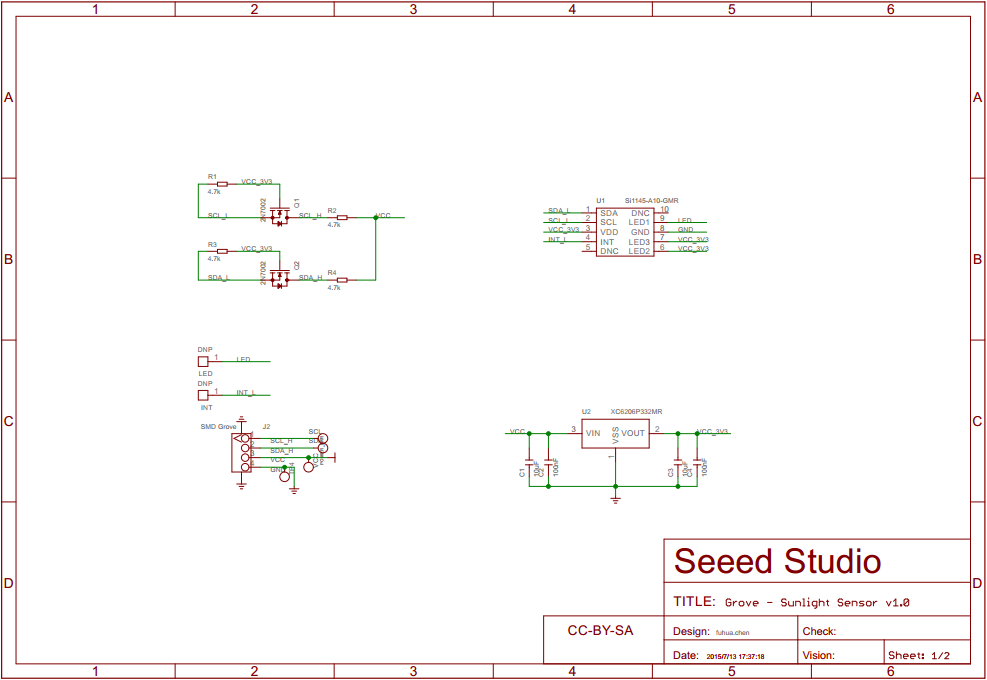


Figure (10): Si1145 Sunlight Sensor Schematic

3 IN 1 SENSOR

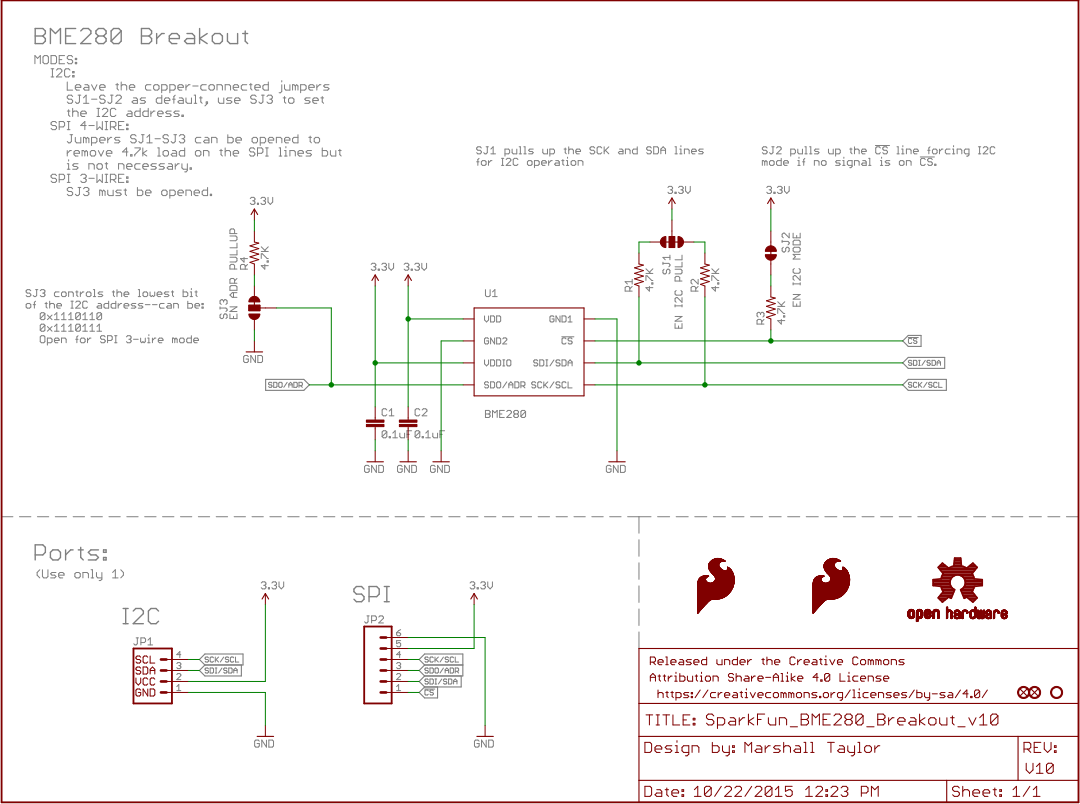


Figure (11): SparkFun BME280 Atmospheric Sensor Schematic

BATTERY MANAGEMENT

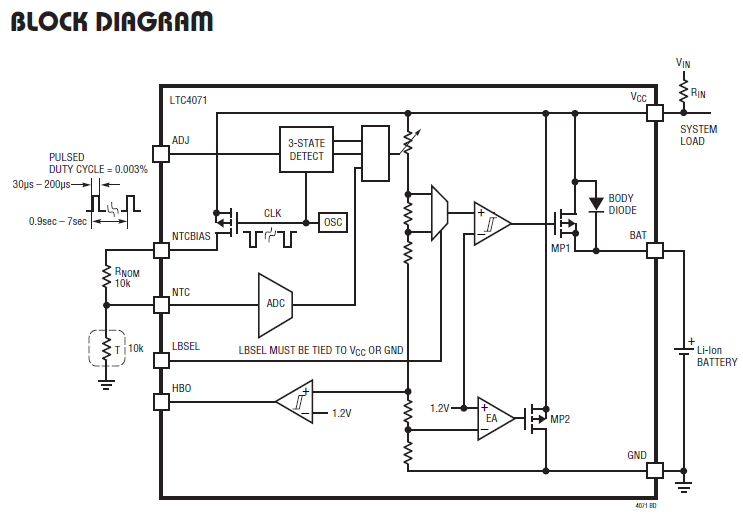


Figure (12): LTC4071 Battery Management Unit Schematic

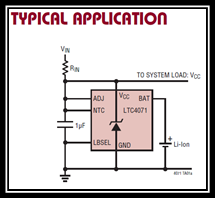
****

Figure (13): LTC4071 Battery Management Application Schematic

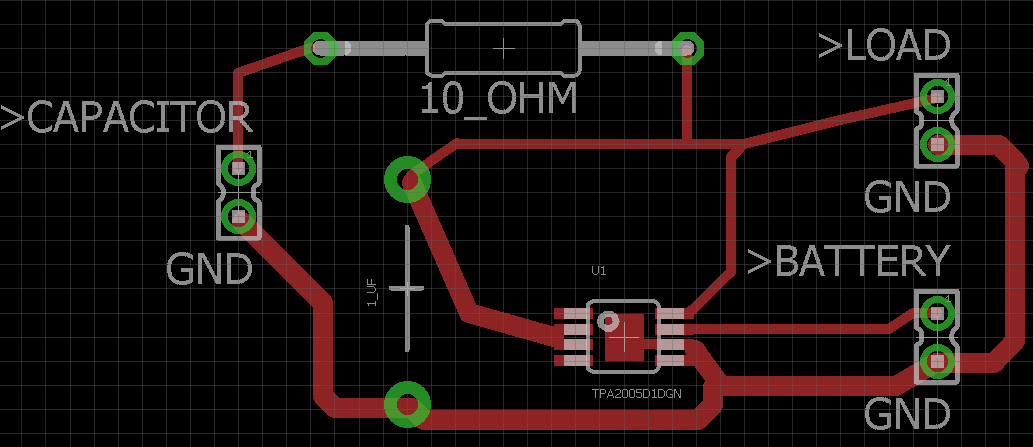


Figure (14): Eagle PCB Software Battery Management Design

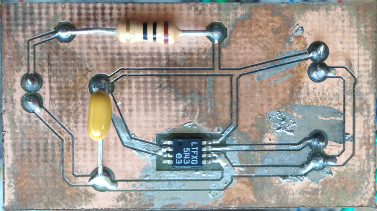


Figure (15): Physical PCB of Battery Management Circuit

Micro Wind Turbine



Figure (16): Wind-Kinetic 10W Micro Wind Generator

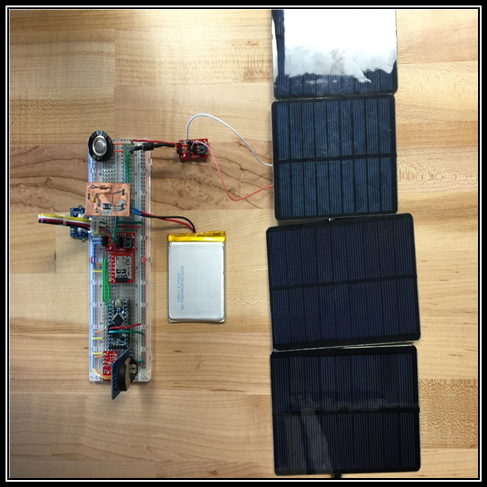


Figure (17): Completed Prototype

**APPENDIX B**

Arduino Code:

#include <stdint.h>

#include "SparkFunBME280.h"

#include "Wire.h"

#include "SI114X.h"

#include "Arduino.h"

#include <SPI.h>

#include <SD.h>

#include <avr/sleep.h>

#include <avr/power.h>

#include <DS3232RTC.h>

#include <Time.h>

//Global sensor object

BME280 mySensor;

SI114X SI1145 = SI114X();

tmElements\_t tm;

int pin2 = 2;

unsigned int sampleNumber = 0;

int CS\_pin = 10;

float temp = 0;

float pressure = 0;

float humidity = 0;

float light = 0;

void pin2Interrupt(void) {

 detachInterrupt(0);

}

void enterSleep(void) {

 /\* Setup pin2 as an interrupt and attach handler. \*/

 attachInterrupt(0, pin2Interrupt, LOW);

 delay(100);

 set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN);

 sleep\_enable();

 sleep\_mode();

 /\* The program will continue from here. \*/

 /\* First thing to do is disable sleep. \*/

 sleep\_disable();

}

void setup()

{

 //\*\*\*Driver settings\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

 //commInterface can be I2C\_MODE or SPI\_MODE

 //specify chipSelectPin using Arduino pin names

 //specify I2C address.  Can be 0x77(default) or 0x76

 //For I2C, enable the following and disable the SPI section

 ADCSRA = 0;

 power\_adc\_disable();

 power\_timer1\_disable();

 power\_timer2\_disable();

 power\_usart0\_disable();

 mySensor.settings.commInterface = I2C\_MODE;

 mySensor.settings.I2CAddress = 0x77;

 //For SPI enable the following and dissable the I2C section

 //mySensor.settings.commInterface = SPI\_MODE;

 //mySensor.settings.chipSelectPin = 10;

 //\*\*\*Operation settings\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

 //renMode can be:

 //  0, Sleep mode

 //  1 or 2, Forced mode

 //  3, Normal mode

 mySensor.settings.runMode = 3; //Normal mode

 //tStandby can be:

 //  0, 0.5ms

 //  1, 62.5ms

 //  2, 125ms

 //  3, 250ms

 //  4, 500ms

 //  5, 1000ms

 //  6, 10ms

 //  7, 20ms

 mySensor.settings.tStandby = 0;

 //filter can be off or number of FIR coefficients to use:

 //  0, filter off

 //  1, coefficients = 2

 //  2, coefficients = 4

 //  3, coefficients = 8

 //  4, coefficients = 16

 mySensor.settings.filter = 0;

 //tempOverSample can be:

 //  0, skipped

 //  1 through 5, oversampling \*1, \*2, \*4, \*8, \*16 respectively

 mySensor.settings.tempOverSample = 1;

 //pressOverSample can be:

 //  0, skipped

 //  1 through 5, oversampling \*1, \*2, \*4, \*8, \*16 respectively

    mySensor.settings.pressOverSample = 1;

 //humidOverSample can be:

 //  0, skipped

 //  1 through 5, oversampling \*1, \*2, \*4, \*8, \*16 respectively

 mySensor.settings.humidOverSample = 1;

 //Calling .begin() causes the settings to be loaded

 delay(10);  //Make sure sensor had enough time to turn on. BME280 requires 2ms to start up.

 mySensor.begin();

 while (!SI1145.Begin()) {

    delay(1000);

 }

 //CS pin is an output

 pinMode(CS\_pin, OUTPUT);

 //Card will draw power from pin 8 so set it high

 //pinMode(pow\_pin, OUTPUT);

 //digitalWrite(pow\_pin, HIGH);

 // check if card ready

 SD.begin();

 pinMode(pin2, INPUT);

 //setTime(21,26,00,12,4,2016); //use to set/reset time of RTC

 //RTC.set(now());

 RTC.squareWave(SQWAVE\_NONE);

 RTC.setAlarm(ALM1\_MATCH\_MINUTES,0,0,0);

 RTC.alarmInterrupt(ALARM\_1, true);

 RTC.alarm(ALARM\_1);//reset alarm2 status

 enterSleep();

}

void loop(){

 //Each loop, take a reading.

 //Start with temperature, as that data is needed for accurate compensation.

 //Reading the temperature updates the compensators of the other functions

 //in the background.

 temp = 0;

 pressure = 0;

 humidity = 0;

 light = 0;

 RTC.read(tm);

 if(tm.Minute == 0 && tm.Second == 0 ) {

    while(sampleNumber<5){

     temp=mySensor.readTempF()+temp;

     pressure = mySensor.readFloatPressure()+pressure;

     humidity = mySensor.readFloatHumidity() + humidity;

     light = SI1145.ReadVisible() + light;

     File dataFile  = SD.open("test.txt", FILE\_WRITE);

     //check if file is successfully opened

     if (dataFile) {

       dataFile.print(mySensor.readTempF()); // write to SD card

       dataFile.print(",");

       dataFile.print(mySensor.readFloatHumidity());

      dataFile.print(",");

       dataFile.print(mySensor.readFloatPressure());

       dataFile.print(",");

       dataFile.print(SI1145.ReadVisible());

       dataFile.print(",");

       RTC.read(tm);

       dataFile.print(tm.Hour, DEC);

       dataFile.print(":");

       dataFile.print("00");

       dataFile.print(",       ");

       dataFile.print(tm.Month,DEC);

       dataFile.print('/');

       dataFile.print(tm.Day, DEC);

       dataFile.print('/');

       dataFile.print(tm.Year+1970, DEC);

       dataFile.print("   ");

       dataFile.print(tm.Hour, DEC);

       dataFile.print(':');

       dataFile.print(tm.Minute, DEC);

       dataFile.print(':');

       dataFile.println(tm.Second, DEC);

       dataFile.close(); //have to close file when done writing

       }

     delay(1000); //delay for 1 second

     if (sampleNumber==4){

       temp=temp/5;

       pressure=pressure/5;

       humidity=humidity/5;

       light = light/5;

       // Open file to write to.

       //File will automatically be created if non existent

       File dataFile  = SD.open("test.txt", FILE\_WRITE);

     //check if file is successfully opened

     if (dataFile) {

       dataFile.print(temp); // write to SD card

       dataFile.print(",");

       dataFile.print(humidity);

       dataFile.print(",");

       dataFile.print(pressure);

       dataFile.print(",");

       dataFile.print(light);

       dataFile.print(",");

       RTC.read(tm);

       dataFile.print(tm.Hour, DEC);

       dataFile.print(":");

       dataFile.print("00");

       dataFile.print(",       ");

       dataFile.print(tm.Month,DEC);

       dataFile.print('/');

       dataFile.print(tm.Day, DEC);

       dataFile.print('/');

       dataFile.print(tm.Year+1970, DEC);

       dataFile.print("   ");

       dataFile.print(tm.Hour, DEC);

       dataFile.print(':');

       dataFile.print(tm.Minute, DEC);

       dataFile.print(':');

       dataFile.println(tm.Second, DEC);

       dataFile.close(); //have to close file when done writing

       } //end of datafile if

     RTC.alarm(ALARM\_1);//reset alarm2 status

       }//end of if sample==4

     sampleNumber++;

    }//end of while loop

 }//end of if new second

 sampleNumber=0;

 enterSleep();

}//end of main loop