

KUMU A‘O CUBESAT: THERMAL SENSORS ON A CUBESAT

Tyson K. Seto-Mook
Department of Electrical Engineering
University of Hawai‘i at Mānoa
Honolulu, HI 96822

INTRODUCTION

A. Abstract

CubeSat is a project that will develop, test, launch, and operate a functional satellite at a fraction of the cost of a commercial satellite. The CubeSat project provides a way for the students to learn the various steps involved with mission design within a reasonable amount of time. Currently at the University of Hawai‘i at Mānoa, UH students are designing CubeSat called the Kumu A‘o CubeSat. The Kumu A‘o’s primary objective is to develop a fully functioning cubesat bus to serve as a prototype for future UH cubesat missions. CubeSat’s, in general, have many different mission. However, the Kumu A‘o’s mission is to observe the thermal environment a CubeSat endures while in space. A temperature gathering routine will be executed to gather thermal data on each side of the CubeSat. The design will be using thermistors as sensors and the MSP340 micro controller to gather the temperature data values. This will give information to the temperature ranges a CubeSat experiences while in space.

B. Background

The CubeSat is a valuable project to the industry and universities across the globe. Since commercial satellites are expensive and take years of planning, CubeSat is a great way for students to understand the basics of developing and operating satellite. The CubeSat will weigh about 1 kilo grams and will have a volume of 10 cm³. This is a much smaller satellite that will have a very reasonable cost, which will be a small fraction of manufacturing and launching large satellites, which is over \$300 M. It will also take a more reasonable amount of planning, unlike commercial satellites. CubeSat allows universities to engage in satellite experience and space exploration at low cost. CubeSat, in turn, allows students to gain valuable knowledge and experiences for future careers and aspirations. The CubeSat project is best fit to accommodate the space industry in collaboration with colleges.

The UH CubeSat project mission goals are to present an engineering demonstration performed by UH students. The main objectives are to develop, test, launch, and operate Hawai‘i’s first launched CubeSat. In addition, this project is to also create a reference model for future students pursuing the CubeSat project in the future. We will also try to create a plug and play template that will allow easy exchange between different types of payloads. To achieve this we are first starting out with the CubeSat kit, which is manufactured by Pumpkin Incorporated. This kit provides us with bear minimum hardware such as, a chassis and a flight module, to help jump start our CubeSat design. The kit will also save us time in starting the CubeSat from scratch. Also so that there is more focus towards the analysis of the subsystems and to house the payloads. Our CubeSat goal is to have it survive for a period of three months in space and send us status updates.

The CubeSat student team is currently comprised of nine undergraduate engineering students; two mechanical engineers and seven electrical engineers. A former CubeSat student Byron Wolfe and professor Lloyd French, with JPL experience, will be available as the mentors and advisors to the CubeSat team. They will provide advice to the team of both technical and non-technical issues. However, decisions of project matters will be established solely by team members and the project engineer.

This progress report will take a detailed look at the design phase of the thermal sensor module of the command and data handling subsystem.

HARDWARE DESIGN

This section provides a description of the thermal sensor circuit and each of its components.

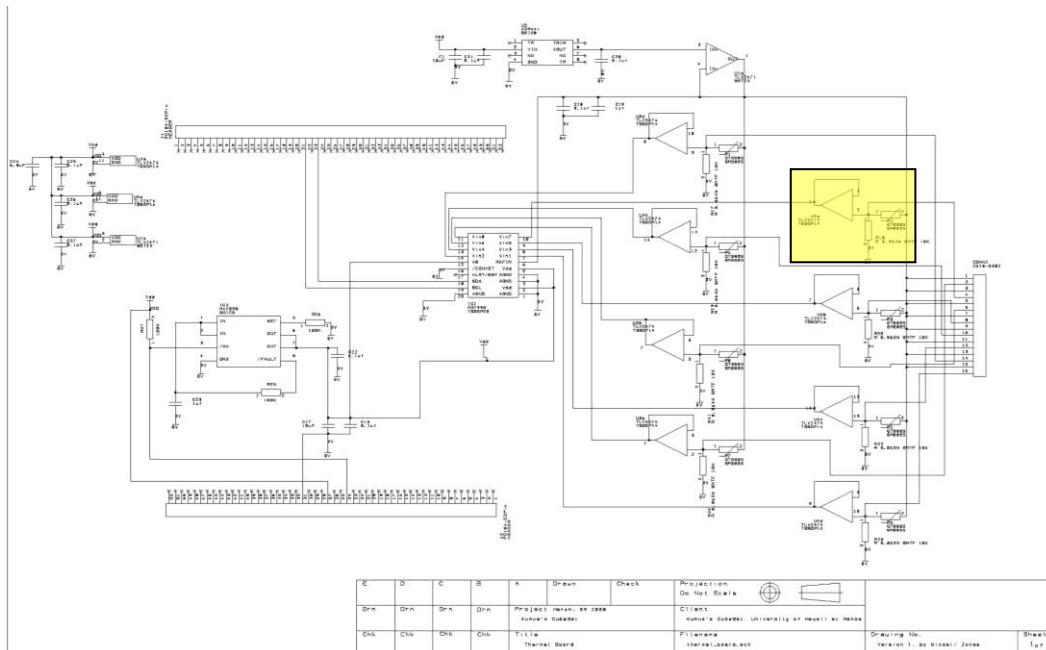


Figure A – Thermal hardware design layout

A. General Overview

Figure A shows our schematic design layout for our thermal sensor circuit. Starting from the top, we have our voltage reference chip which outputs an accurate low noise 2.5 volt signal to our thermistor sensors. It is then fed through a voltage buffer to increase the signal's current so that more power may be drawn from that signal. This signal is then connected to our voltage divider circuit which contains the thermistor. From there the signal is passed through another voltage buffer, which is placed here to maintain signal accuracy, then to the analog-to-digital converter. The analog-to-digital converter then converts the signal and sends the converted data to the MSP430 via the I2C protocol. Which is then stored into memory.

B. Voltage Reference (ADR441)

The ADR441 chip is a voltage reference chip that provides an accurate low noise signal to the thermister voltage divider circuits. This chip is used to maintain data accuracy for our temperature readings.

C. Thermister

Thermisters will be used as our sensors to measure the thermal characteristics of our CubeSat while orbiting. A thermister is a resistor which varies its value depending on its temperature. This sensor was chosen because of its temperature sensing range. Although the circuitry involved in implementing these sensors are more complicated than our other leading choices, these thermisters had a range that was more suited for a space environment. We also chose to use these as our sensors because they are low cost and had good accuracy. The highlighted portion of Figure A shows how we will be using thermister in our circuit. Through this set up we can use the voltage divider equation to get the voltage across the thermister to check its voltage value. We then send that signal through a voltage buffer, which will help maintain signal accuracy, and then to the ADC. Once we've gotten the data from the analog to digital converter, we can then figure out the temperature taken by using the voltage divider equation again to get the resistant value of the thermister and comparing it to its resistor to temperature chart.

D. Analog to Digital Converter (AD7998)

The analog output from the voltage divider is an input into the analog to digital converter's (ADC) input pins, pins 7 through 14. Pin 6 is used as the ADC's voltage reference and it will be getting 2.5 volts. Pins 18 and 19 are the I2C communication pins, where 18 is the SDA data line and 19 is the SCL data line. The address select pin will use pin 32 of the PC104 bus so we can control the I2C address of the ADC.

To start a conversion we must first set the configuration register. We can do this by sending a write sequence through I2C to the ADC. Then we can set the address pointer to the configuration register and then writing 16 bits of data to it. In the 16 bits the first 4 bits are "don't care bits," which are then followed by the input select bits. These are the bits you set to logic 1 in order to tell the ADC which inputs we want to be converted. The last 4 bits of data are alert bits which we will not be using. To retrieve data, we first have to set the address pointer back to the conversion results register then start a read sequence to the ADC through I2C. During the read sequence, the ADC powers up and automatically starts its conversion which takes about 3 micro seconds. The results are then stored into the ADC's conversion results register. The ADC will send 16 bits of data back to the microcontroller with the most significant bit being an alert bit, if we have set any alerts, followed by 3 bits of data stating which input the next 12 bits of data was taken from.

Although The MSP430 microcontroller does have ADC pins, we chose to use an external ADC in order to reduce the cost of pin consumption. We also chose to use an external ADC to have the ability to power down the whole interface to reduce the cost of power consumption.

E. Inter-Integrated Circuit (I2C)

The I2C is a communication protocol with a five step process. The process consists of the start, address, acknowledge, data transfer and stop steps.

The start state is initiated by the bus master, which is typically a microcontroller. To issue a start condition, the bus master pulls the Serial Data line (SDA) line to a low then pulls the Serial CLock line (SCL) line to a low. This sequence of signals on the bus lines indicate to the bus slaves that an address will be broadcasted on the bus lines.

The address state is initiated by the bus master by broadcasting a 7 bit address of the device it wishes to communicate with on the SDA bus line. Along with 7 bits, an 8th bit is also sent to notify the device if it will be read from or written to. If the address matches the device, it will then proceed to the acknowledge state. If the address does not match the device, it ignores the rest of the following data until a stop signal is broadcasted on the bus.

The acknowledge state is a signal sent back from the chosen slave device to the bus master. This tells the master that the slave is ready to communicate. The signal is given by having the device pull the SDA line low until the master generates one clock signal on the SCL line.

The data transfer state is then initiated. This is where the slave device sends or receives 8 bits of data to or from the master. When all the data is finished being sent, a stop signal is broadcasted on the I2C bus lines.

The stop state is initiated by the bus master driving both bus lines back to high; first starting with driving the SCL line to a high signal followed by driving the SDA line to high signal.

F. MSP430 Microcontroller

The MSP430 is an ultra-low-power 16-bit RISC mixed-signal processor which is used for low cost and low power consumption systems. The microcontroller can be programmed in either C++ or C. since I am more familiar with C rather than C++ we will be programming the microcontroller in C. For this particular interface we will be communicating from the microcontroller to the ADC through I2C.

G. Power Switch (MAX890)

The MAX890 is a low voltage p-mos mosfet power switch. It will be used to shut down the whole thermal sensor system. The reason why we have this switch is so that we can save on power consumption. Since we are unsure how much power we may really need for the whole CubeSat, it is safer to be able to shut down the thermal sensors and turn it on only when we need to gather sensor readings.

SOFTWARE DESIGN

This section provides a description of what my software design for the thermal sensor circuit.

A. General Overview

Since there are many programmers working on the CubeSat, each routine may be implemented differently. This creates uncertainties on how my software routine will fit in with the main C&DH program.

The solution to this problem was to create a thermal routine so that it is independent of the main C&DH program. This will allow the routine to work well with other routines without conflicting with other routines. Not only will it be easier for the separate routines, but it will also keep them well organized.

For example, Instead of having one function read a sensor and calculate its value individually, there are two functions. One function that will read all sensors and store them in memory and, another that will read and convert the sensor value from memory when needed.

This also allows us to save power by turning on the system, read all 8 sensors at once, then immediately turning off the thermal system. Then analyze the sensor values at a later time. On the other hand, keeping the whole system on while individually reading and analyzing a sensor value will be inefficient in time and power.

B. C Language Structure Design

The C structure will contain 8 integer fields; sensor1 through sensor8. Each field will hold its respected sensor value taken from the ADC. Along with the field of this structure will contain several member functions. Each member function will have a unique task in manipulation of the structure field's. A member function is a function which is exclusive to the structure it is declared in. This minimizes data corruption from other routines within the system.

C. Data Members

In this C structure, there will be eight data members. The data members are unsigned characters (integers) that will hold 16 bits of data retrieved through I2C from the ADC. The most significant bit will be an alert bit, followed three bits that tells us which sensor was read, and finally the last twelve bits are the value of the sensor.

D. Member Functions

This object will contain several member functions, which can be placed into three different function types. There is the constructor/destructor, accessor, or mutator function type.

With in the constructor and destructor type functions there will be two functions, an initiation and shutdown function. The initiation function will clear current sensor values, turn on the thermal system, and ready the ADCs registers for conversions. The shutdown function will turn off the thermal system.

With in the accessor function type there will be one main function, the get sensor function. This function will only be allowed to read, from the desired sensor value in memory, and return its value to the main command and data handling program.

With in the mutator function type there will also be one main function, the read sensor function. This is the function that will communicate with the ADC, through I2C, and attain the sensor's value and store them in memory.

E. Integration of thermal C structure

By keeping everything exclusive to the member functions only, it will keep program well organize and easy to use. The main command and data handling program will now only have 3 functions to call instead of having several lines to execute. Main program will just have to run the initiation, then read sensor, then shutdown function and it will have its sensor values in memory ready to be analyzed and processed.

F. Flow charts

In figures B and C are of the command and data handling flow charts. Tracing the flow charts will show how the thermal sensor routine fits with in the big picture of the main satellite program.

Flow Chart of Thermal System Temperature gathering

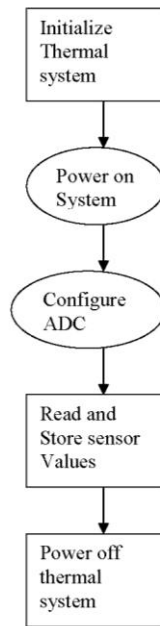


Figure D – Thermal Temperature gathering Flow chart

TEST PLAN

A. Hardware Testing

The first component to be tested will be the ADC. After a successful testing of the I2C communication we can now test the ADC. To test the ADC we will set up the circuit and program the micro controller to collect and display a set value in one of its ADC channels. This can be done with an external power supply. First we will test the limits of the ADC, meaning that we will test its max value and lowest value. This can be done by setting a channel of the ADC equal to the reference voltage of the ADC. There is a successful read if all bits displayed are high logic. If the result do not math the expected results then there maybe a problem with the ADC. We can verify this by also setting the channel to zero where we should get all low logic values. If all tests have passed, the ADC is verified to be in working condition.

The second component to test will be a thermister circuit. Since I2C and the ADC are validly working, we can now only worry about this circuitry. We will first start by setting up a circuit that will connect the processor, ADC, and a thermister together. The test will be to take an average temperature using an out side source then compare them to the temperature given by the ADC from the termister. If the temperature from the thermister is not similar, there is a circuit problem that will have to be debugged. If they are, we can verify it by heating or cooling the thermister to see if the temperature changes in its expected direction.

Once these individual components have been tested we can now move into the finalizing phase of the project.

B. Software Testing

To test the software we will start by testing the I2C communication. This test will be done with an LCD screen which can be written to through I2C. If this is successful, then I know that I2C is working. If it isn't there maybe code errors or hardware errors. An oscilloscope will be used to test for hardware failures. If there are no hardware failures, we should see square wave signals the oscilloscope for both the I2C lines. If there is a wave signal then we have a software problem. Debugging the code would be the next step. After we have successful communication, I2C is verified and now is proven to no be a problem of the system. Other than this the software should be the fairly easy to debug.

CONCLUSION

For the current status, for the thermal system, the research and design phase is completed. Since the design phase had just been completed, we haven't yet tested the circuit to find problems. Theoretically, this circuit should work; however, we may have issues with accuracy of the gathered data. The accuracy of our temperature depends on accuracy of our system. The next step is to prototyping, debugging and ultimately finalizing our system.

ACKNOWLEDGMENTS

I would like to thank the Hawai'i Space Grant Consortium for providing the opportunity for undergraduate research in sciences. I would also like to thank my mentors Lloyd French and Byron Wolfe who gave advices and supported me throughout the semesters.

REFERENCES

PC. (2008, December 9). In Wikipedia, The Free Encyclopedia. Retrieved 08:49, December 19, 2008, from <http://en.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=256809889>

Maxim-ic Website: Power Switch Available [Online]: <http://www.maxim-ic.com/>

Analog Devices Website: Analog to Digital Converter
Available [Online]: www.analog.com/en/analog-to-digital-converters/