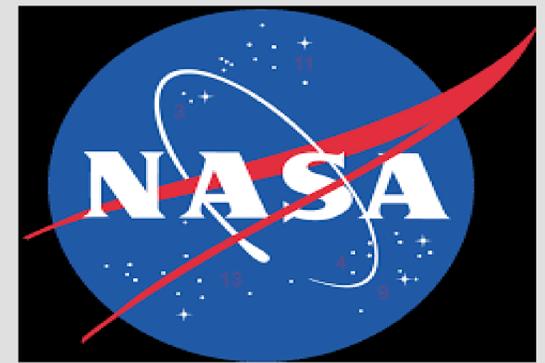


Intelligent Software for Nanosatellites

Interns: Jarmal Johnny, Edouard Michel, Valina Maitland, Tania Nelzy
Faculty Mentors: Dr. Shermane Austin | Prof. Michael Jiang



ABSTRACT

NASA's CubeSat launch initiative provides opportunities for students to get exposure to NASA activities and gain interest in STEM careers. CubeSats are nanosatellites and software development plays an important role in control. The software for controlling, tracking, and communicating with the satellite has to be developed following best practices and ensuring that it is robust and able to tolerate and recover from the unexpected.

Software engineering best practices and tools (e.g. for example, specification requirements, agile development, change control, extreme programming, etc.) can be used to achieve software of quality. In addition, artificial intelligence techniques can be used to help automate or improve the software development process. Example cases include automated tools for program modeling, verification and validation, automated generation of test units, and automated verification of unit test coverage and adequacy. AI techniques can help even beyond the development phase.

Despite following best practices, when the software is used in critical missions, it might be very risky to assume that the software is free of faults. Further, if the system is to be used in a dynamic and complex environment, it might face unexpected situations not considered during its design.

Autonomic Computing (AC), a vision and architecture proposed by IBM, relies on the use of AI techniques to build computing systems with self-managing characteristics so that they are able to adapt to unpredictable changes while hiding intrinsic complexity to operators and users. An AC system makes decisions on its own, using high-level policies. It has sensors for self-monitoring, effectors for self-adjustment, and knowledge and planner adapters for exploiting policies based on self- and environment awareness.

Introduction/Background

The primary objective is the design, implementation and testing of CubeSat prototype system based on AC, defining policies, effectors and required transformations based on a standard suite of CubeSat subsystems including Command and Data Handling, Electrical Power System, Science, Attitude Determination and Control (ADC) and Communications. The objective involves the investigation of preliminary AC computational capacity, load-sharing, power constraints, defined system requirements, etc.

The testbed is built around an Arduino, an open source tool utilized for developing electronic devices which includes a high-level library for training purposes. It simplifies the code needed to interface with sensors, control physical outputs, and implement communication. A data shield is an attachment for the Arduino that allows us to interface with an SD card without using any pins. A DHT22 sensor measures temperature, humidity and heat index. The Light/IR sensor precisely measures illuminance in diverse lighting conditions. The ADXL335 is a three axis accelerometer that measures the acceleration of gravity on the x, y and z-axes.

METHODS

Integration Procedure

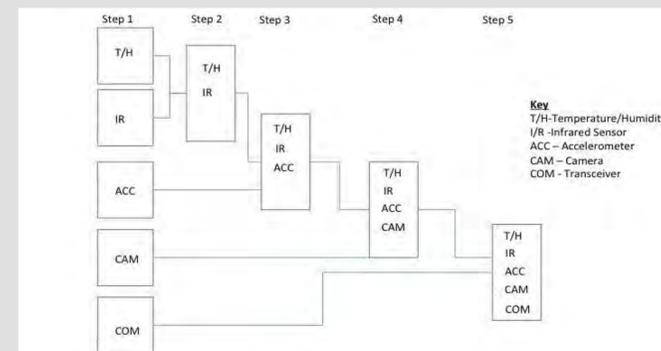


Figure 1: A general plan of our sensor integration process. First we work on them individually to familiarize ourselves with the sensor then we integrate it with the others. The integration includes both hardware and software.

The testbed system includes two Atmega 328 microcontrollers communicating via a system bus, thermal monitoring with temperature/humidity sensors, a low-profile IR sensor, a low-resolution camera, an accelerometer, internal storage, GPS receiver, and transceiver. Defined system requirements will also include the integration of a reaction wheel at later stages. Preliminary integration and testing of the component suite has been underway to provide guidance for power, data and link budgets and telemetry needs as an initial basis for system and resource requirements.

Prototype Schematic

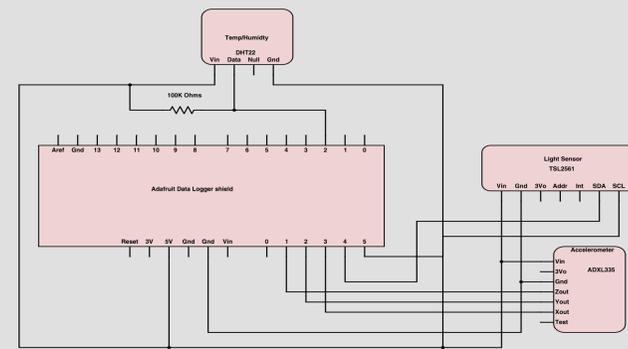


Figure 2: Our wiring schematic. Visually details all the connections between the sensors and the Arduino data logger, which is connected on top of the Arduino.

RESULTS

This investigation is still in preliminary stages. Only a partial integration and testing of the testbed system has been completed. Identification of risks, potential faults and required transformations and budget determinations have not yet been completed and microcontroller trades are needed.

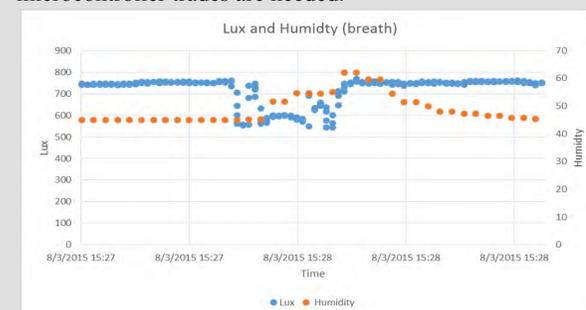


Figure 3: This figure details the calculated lux and the humidity when someone exhales on the sensor. The dip in lux represents when the person blocked some of the light to the light sensor. An increase in humidity can be seen occurring immediately after the event representing the breath.

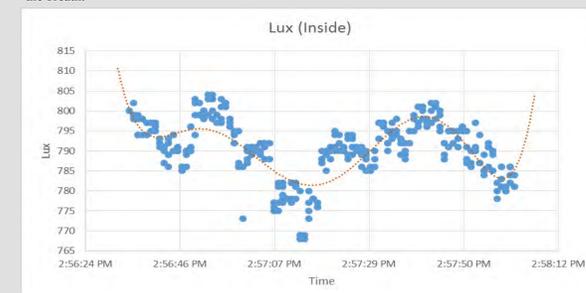


Figure 5: This figure represents the lux calculated inside of a building. There is an unexpected sinusoidal trend that we believe may represent this particular light source inside the building, dimming and brightening at pseudo-regular rates.

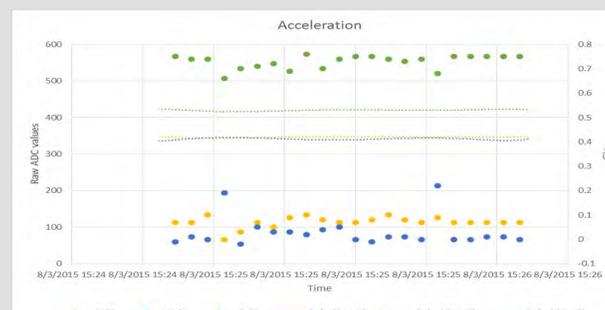


Figure 4: The ADC on the Arduino converts voltages to a value between 0 and 1023. These are then converted to g's (which is a force caused by acceleration similar to gravity factor) using a formula and knowledge of the unit obtained from the datasheet. This accelerometer is 3-axis which detects x, y, and z accelerations.

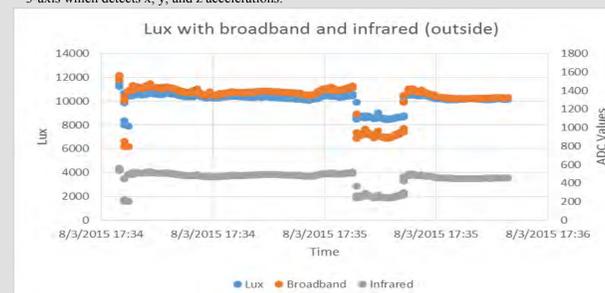
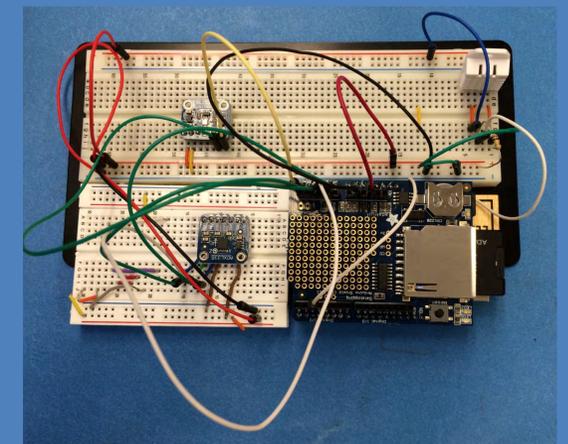


Figure 6: This figure represents light data taken outside of our college building. The values on the left represent Lux which is the magnitude of the energy of visible light adjusted to account for how humans perceive light. The values on the right represent the values from the Analog Digital Converter (ADC) on the sensor itself which is used to calculate Lux. Broadband represents the value for the entire range the sensor detected (visible and Infrared light) while IR represents just IR light. It was a bright and hot day and the unit experienced direct sun-light. Half way through 17:35 (5:35 PM) we can see a dip across all values. This occurred when the unit was shielded from the sun by cloud cover.

CONCLUSION

It is premature to draw conclusions about the feasibility of implementing an AC system for CubeSat, and additional investigation is necessary and ongoing. This is due to time constraints; since this was only a seven week program. Presently, the IR sensor, Temperature/Humidity, and Accelerometer are interfaced with the data logger and the micro-controller. Some of the challenges we are facing at this stage includes getting the camera to communicate with the Arduino and low memory constraint on the Atmega 328 since it only contains 32KB. Therefore, the code needs to be optimized without sacrificing functionality. In addition, the power used by the sensors needs to be calculated. During the testing phase, we will be using solar cells to provide typical IU CubeSat power constraints. This prototype will be tested in the coming months on a high-altitude balloon launch.



REFERENCES

- [2] www.arduino.cc
- [3] [http://www.nasa.gov/sites/default/files/files/ELaNa-II-Factsheet-508\(1\).pdf](http://www.nasa.gov/sites/default/files/files/ELaNa-II-Factsheet-508(1).pdf)
- [4] <http://www1.cuny.edu/mu/forum/2013/11/14/medgar-evers-college-set-to-launch-satellite-on-nasa-rocket/>

ACKNOWLEDGEMENT

NASA, New York City Research Initiative, New York NASA Space Grant Consortium, Goddard Institute of Space Studies, and GSFC Office of Education contributed greatly to this research project.