Portland State University

Electrical and Computer Engineering Capstone Final Report

Version 1.3

6/10/2015

Erebus Labs

Open Sensor Platform

Submitted By:

Colten Nye

Steve Peirce

Golriz Sedaghat

Supervisor:

Dr. Lisa Zurk

Sponsors:

Dr. Mike Borowczak

Dr. Andrea Burrows

Abstract

Current existing sensor platforms require programming/electronics experience which most K-12 students, beginners and hobbyists lack. In this project we built and designed an open-source data logging device allowing K-12 students to collect data from their environment using either the custom sensor boards or those of their own design.

Introduction

Data logging

Sensors are devices which detect and measure an input from the physical environment and output a value that can be human interpreted. A sensor platform is a base upon which one or more sensors can be read on a repeating interval over an extended duration.

Erebus Labs

The focus of our sponsor, *Erebus Labs & Consulting LLC*, is on the development of Secure Hardware/Software solutions with a focus on educational outreach and S.T.E.M. integration in K-20 classrooms. S.T.E.M. refers to education pertaining to Science, Technology, Engineering, and Mathematics. These topics are data and information centric. Erebus Labs feels that there is not enough emphasis in schools on the practice of collecting data. In this respect, one of their concerns has been regarding K-12 students having access to and making use of data acquisition systems (sensor platforms).

Problem Identification

There are a few common limiting factors of current sensor logging platforms in the context of K-12 classrooms:

- Closed design - Our sponsors would like to offer a solution that is fully open for users to tweak and add to as they see fit.
- Inaccessible price range - With the goal of generating interest in STEM type learning in schools, our sponsors would like to offer a product that has a price tag that will not turn away schools and teachers.
- Lack of support for sensors not offered by the manufacturer - If the user wants to be able to log data from a type of sensor that is not offered by the manufacturer, they may be out of luck. Our sponsors would like to make it simple to interface the device to a wide array sensing components.
- Technical skill requirement - Our target market may be as young as kindergarteners. Our sponsors would like the product to be accessible to even the most technically inexperienced users.

Solution:

Our project was to build a sensor platform solution that addresses all of the aforementioned shortcomings by making it open-source, inexpensive, flexible, and easy to use.

Methodology

Considerations

Basic requirements of logging sensor data

Sensors

A sensor platform must have sensors to read from. In our solution, we created two different types of compatible sensors for illustrating the system functionality.

Controller

A sensor platform must include something to perform the various tasks: scheduling, reading from sensors, formatting the data, storing it, etc. This generally takes the form of a microcontroller, as was the case for us.

Access to the data

The user must be given access to the data, whether it be piped out of a communication channel or stored in memory for later access. We chose to store the data on an SD card.

Special Considerations for our system

Interfacing with arbitrary sensors

One of our major requirements was for our system to be able to interface with a wide variety of sensors beyond the ones that we created/supported by the time our project ended.

Easy to use

One of our major goals is to make the system accessible to those with limited technical skills. Because of this, it was our design philosophy to make using the system very simple by abstracting away unnecessary aspects of the design away from the interface.

Open Source / hardware

Even though some parts of the design are abstracted away from the interface for the sake of user-friendliness, they are still accessible to the more advanced user, as we expose all software, firmware, and hardware for this project.

Inexpensive

Many sensor platforms offer a few bells and whistles that, while making the platform more powerful, are too technical for inexperienced users and also drive up the cost. Our solution is intended to be simple and straightforward to keep costs low.

Low power

Because the sensor platform has the potential to be collecting data on battery power for extended periods of time, one of the design considerations was to be energy efficient.

Analyzing collected data

There are myriad ways to analyze a data set. Our team decided that data analysis is outside the scope of our project. Instead, our design presents the data in a common format which is easily portable to many data analysis tools.

Our Approach

After weeks of brainstorming, we converged on the general design This design includes a controller that exposes several analog ports and an I2C bus for interfacing with sensors and has an SD card slot for storing collected data. The controller would be powered via USB port. The controller would be configured from a desktop computer via the USB port, though using rs232 protocol, since USB protocol is unnecessary and quite complicated. The desktop computer would be running software (of our own design) as a Google Chrome App. We would also design and prototype two different types of sensors to test the two types of sensor interfaces.

User Interface

The user interface (UI) is the means by which the user configures the controller to read from certain sensors on user-defined time intervals, as well as other controller options.

We decided to use a Google Chrome App as the platform to build the user interface, primarily because the app would function on all major operating systems without specialization. Additionally, Chrome Apps gain access to a few key pieces of hardware on the host computer, specifically the serial ports for our purposes.

Sensor Description Files

To support the the goal of allowing use of arbitrary sensor types, the UI needs to know what kind of sensors are available. To support this, the UI loads Sensor Description Files (SDF) from a directory indicated by the user. Each SDF describes all necessary information about a sensor type in order to read from that sensor. An SDF for a particular sensor can be acquired if it already exists, or created if the user is more technically inclined.

Technologies used

Google Chrome App

A Chrome App is a hybrid between a website and a program. It is built like, and has all the resources, of a web application, yet is stored and run from the local disk, regardless of internet connectivity.

Chrome APIs

Google Chrome offers a few Application Programming Interfaces (APIs) to gain access to hardware resources.

- **serial**

  This API exposes the machine's serial communication ports (COM, for Windows), as well as functions for using them. This is how we are able to transmit information from the UI to the controller.

- **filesystem**

  This API exposes the machine's filesystem, which allows us to load sensor definition files, as well as save and load the device configuration for later use.

HTML

HyperText markup Language (HTML) is a convention for describing how visual elements are displayed in a web browser, or in our case, the App.

JavaScript

Javascript (JS) is a non-compiled programming language used with web applications. This is the language that does all the heavy lifting for the User Interface.

- **AngularJS**

  This is a very powerful JS library. It's main advantage is 2-way binding between JS and HTML, meaning when a value is changed in the script, that value is updated in real time on screen, and vice-versa.

JSON

JavaScript Object Notation (JSON) is a data exchange format. It is used to for the SDFs, as well as saving and loading the configuration to local disk.

CSS

Cascading Style Sheets (CSS) is the technology used to ensure a set of HTML documents are visually consistent when displayed.

- **Twitter Bootstrap**

  Bootstrap is a very popular, very powerful CSS library.

VirtualCOM

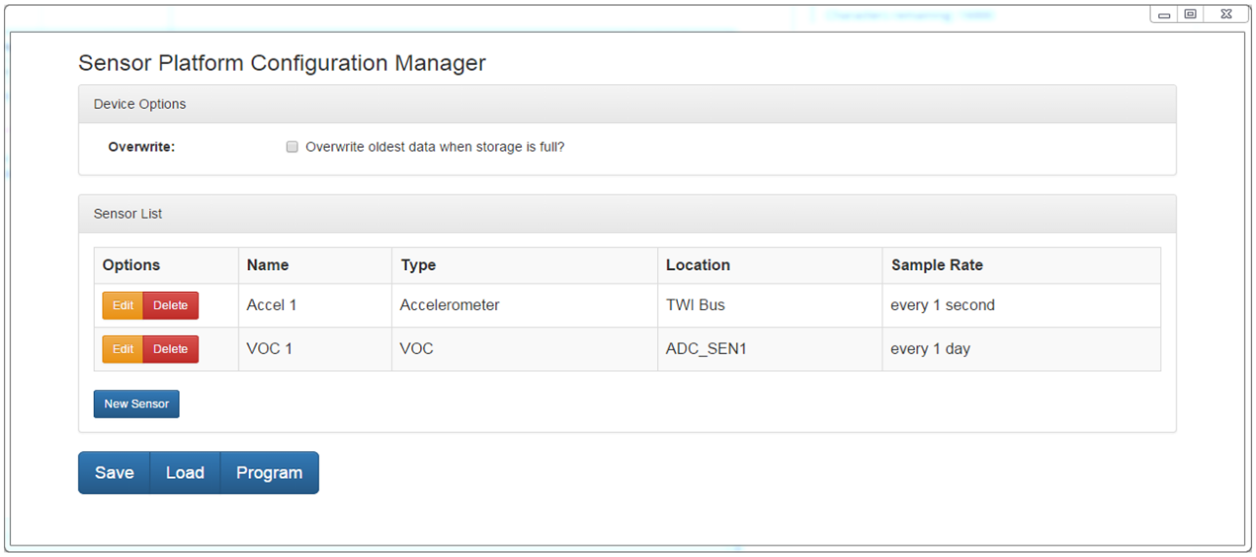
This software creates a virtual serial connection in the host machine. When running, the host machines sees two new COM ports, which are virtually bridged together. This allows the developer to attach the User Interface to one of the COM ports, while the other is attached to a terminal emulator. Thus, any output of UI is piped to the terminal.

ExtraPuTTY

This software was the UI developer's terminal emulator of choice.

Results

The user Interface met and exceeded the expectations of the Sponsors. It is easily acquired, and successfully allows the user to select from available sensors that are dynamically loaded at runtime, specify parameters about the sensors, and program the controller with this information.


Sensor Platform Configuration Manager

Device Options

Overwrite: Overwrite oldest data when storage is full?

Sensor List

Options	Name	Type	Location	Sample Rate
Edit Delete	Accel 1	Accelerometer	TWI Bus	every 1 second
Edit Delete	VOC 1	VOC	ADC_SEN1	every 1 day

Features

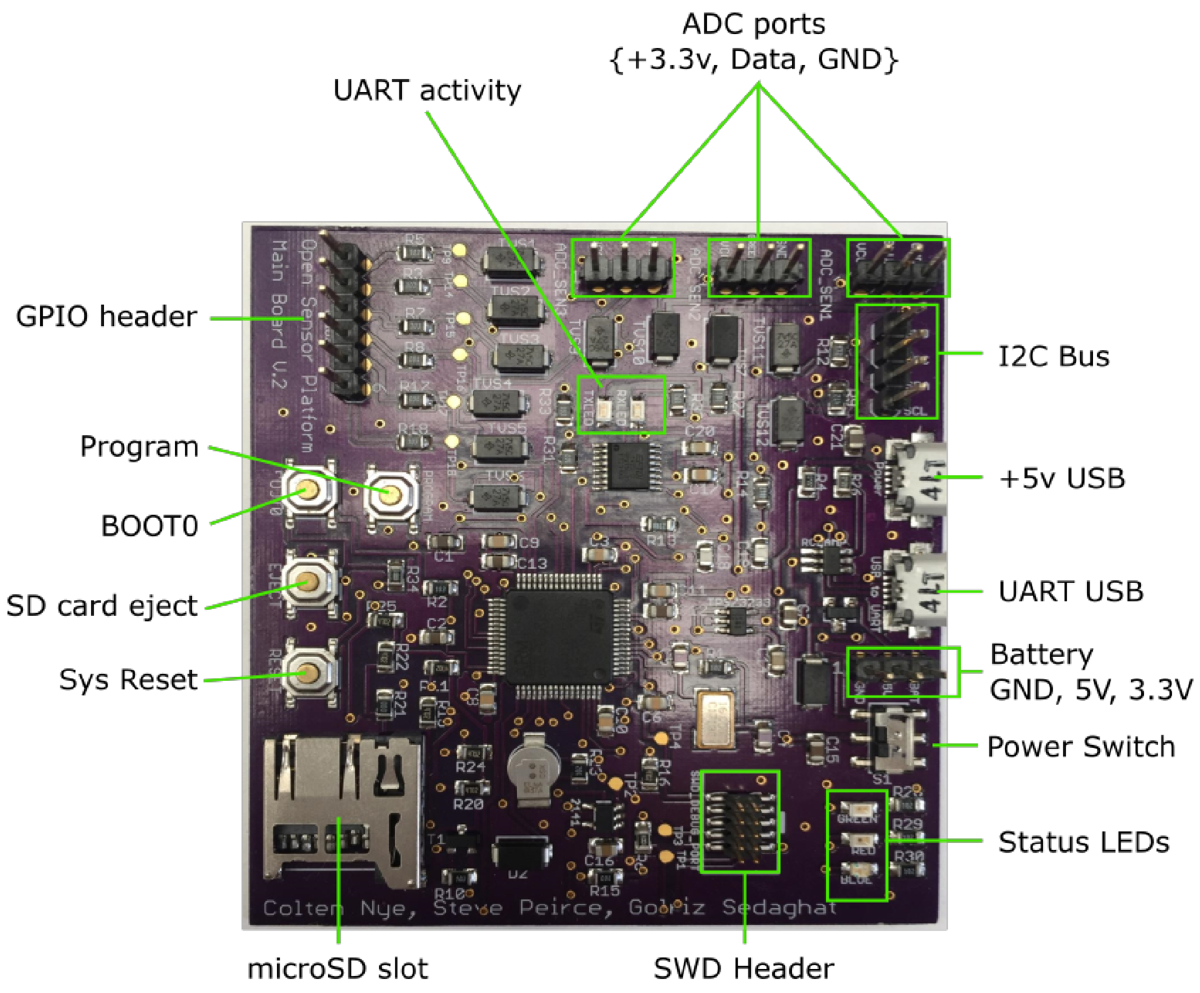
- Extensive validation. There is virtually no way to specify a sensor that has invalid parameters.
- Ability to Edit or Copy a sensor.
- Serial connection protection. Greatly lowers the rate of problems stemming from persistent serial connections
- Saving and Loading the configuration for later use.
- Visual aesthetic. The UI is very simple and easy to understand.

Shortcomings

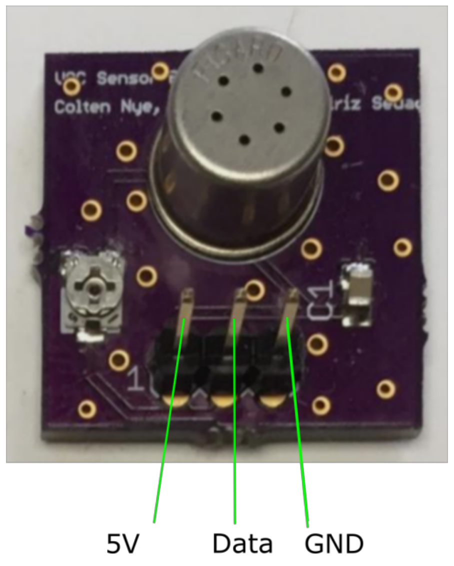
- Loading sensors is not validated. If the JSON format is invalid, the sensor will not be loaded, and without warning. If the sensor description contains invalid fields, it may cause corrupted configuration for the controller, or may crash the UI.

Hardware / Firmware

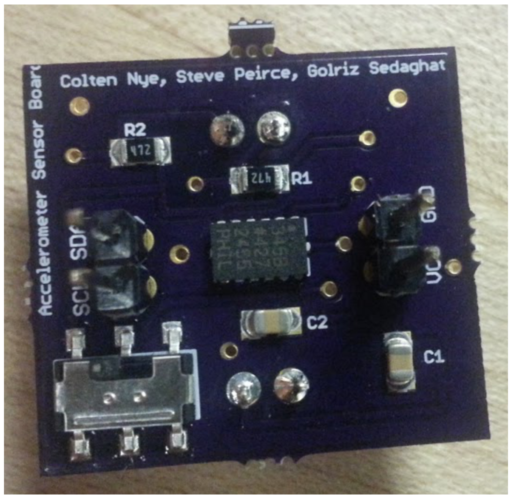
We designed the platform to support ten I2C and three analog sensors. The employed processor on the main board is 32-bit ARM based microcontroller, STM32F205 by ST Microelectronics, which communicates with the sensor boards, reads the collected data from them on user specified intervals, and logs the collected data to a micro-SD card. The open hardware/software design of the main board allows the more advanced user to design a custom version of the board, as well as to upload their own custom binaries via the single wire debug header.



The two provided custom sensors are Volatile Organic Compound (VOC) sensor and accelerometer. VOC's are Carbon-based airborne chemicals. A VOC sensor's conductivity changes in the presence of a detectable gas; the change in the conductivity converts to an analog output signal. Our VOC sensor board employs TGS2602 VOC sensor model which senses VOC's, such as Ammonia and Hydrogen Sulfide. It can be powered directly through the main board by connecting the 5V pin of the sensor to the middle pin of the battery header on the main board. The potentiometer on the VOC sensor board controls the applied voltage to the data pin which in turn sends analog readings to the processor.



The accelerometer sensor communicates over I2C with the main board. The employed accelerometer is Analog Devices' ADXL345. It is a small, ultra-low Power, 3-Axis and popular with hobbyists. The three-pole switch on the board provides two alternate I2C addresses. When the switch is in the ON position (as labeled in figure 3) the I2C address for the device is 0x1D; on the other hand, when the switch is in the OFF position (as labeled in figure 3) the alternate I2C address is 0x53. This would allow employing two identical sensors on the same bus.


Colten Nye, Steve Peirce, Golriz Sedaghat
Accelerometer Sensor Board
R2
R1
SDA
SCL
GND
VCC
C2
C1

The system's specifications are as follows:

Category	Specification
Configuration	Ability to communicate with: I2C sensors (stackable I2C sensor boards) Up to 3 analog sensors
Power	3.3 V operating voltage USB Connector 5V 3x1.5 AA Battery
Clock Speed	16 MHz
Memory	Micro SD Card connector (stores data collected by sensors) Microcontroller (STM32F205) has Up to 1 MB Flash Microcontroller (STM32F205) has 128+4 kB RAM
Communication	I2C (Inter-Integrated Circuit) ADC (Analog to Digital Converter) USB (Universal Serial Bus) SWD (Serial Wire Debug Interface) USART (Universal Synchronous/Asynchronous Receiver/Transmitter)
User Interface	Platform independent Web-application
Supported Operating Systems	Windows
Customization	Main board can be customized over SWD port
Cost	$ 45.8 per main board

Conclusion

In this project we designed a sensor platform supporting analog and I2C sensor types. Our design consisted of the main board and two custom sensor boards: VOC analog sensor and I2C accelerometer sensor. The system is completely an open hardware/software design which allows the user to customize the system in their desired way. Although the current system is an efficient design, there is always room for improvements. The employed microcontroller on the main board, STM32F205, is a very powerful processor with many capabilities. The processor possesses multiple features for optimizing the power consumption of the system; as well as capability of USB to asynchronous serial data transfer interface; due to lack of time we were not able to utilize all of these properties which would turn the system into an even more efficient and powerful design.

References

GUI Technologies:

Chrome Apps: https://developer.chrome.com/home

Chrome serial API: https://developer.chrome.com/apps/serial

Chrome filesystem API: https://developer.chrome.com/apps/fileSystem/

HTML: http://www.w3schools.com/html/

CSS: http://www.w3schools.com/css/

Bootstrap: http://getbootstrap.com/2.3.2/

JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript

AngularJS: https://angularjs.org

JSON: http://json.org/

COM port emulator: http://free-virtual-serial-ports-emulator.software.informer.com/

ExtraPuTTY: http://www.extraputty.com/

Revision History

Version #	Revision Date	Author	Comments
1.0	6/8/2015	Golriz Sedaghat	Creating the initial version
1.1	6/9/2015	Steve Peirce	Minor Revisions
1.3	6/9/2015	Colten Nye	Major revision, addition of content