University of Windsor Electrical and Computer Engineering Department

06-88-443 Embedded System Design

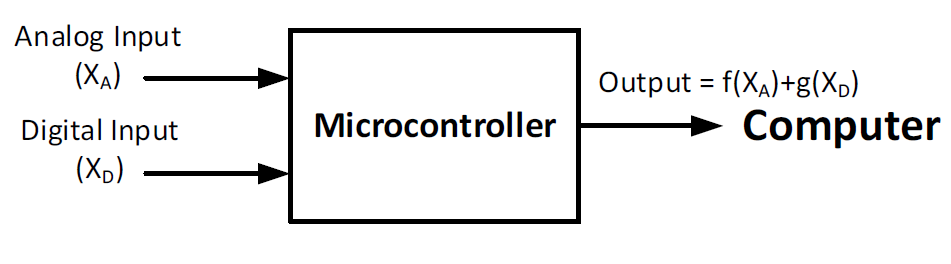
Project 2: Data Acquisition System

# Instructor: Arash Ahmadi

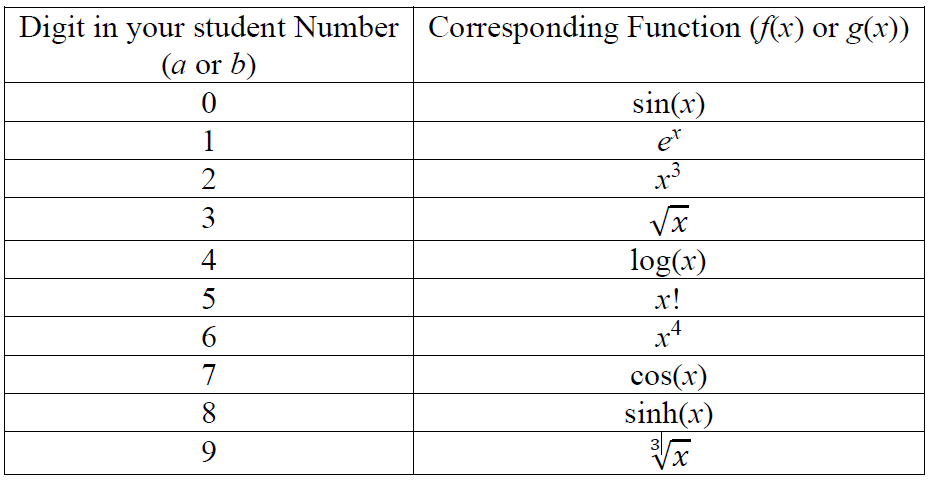
|  |  |
| --- | --- |
| Group Member: Daksh Patel | ID: 104 030 031 |

# Introduction

The objective of this project is to design and implement a data acquisition system and carry out a series of calculations based on student numbers. The instructions provided describe the goals as: Design and implement a simple data acquisition system, as depicted below, which reads a digital (XD) and an analog (XA) input, calculates f(XA) + g(XD) and sends the results to a host continuously. The figure below is provided in the project assignment document.



As shown from this diagram, there are analog and digital inputs that are used to source the assignment data for calculations that are then carried out by the microcontroller. In addition, numerical functions f and g as in the table provided identify the functions to be used based on the last two digits of our student numbers (a and b). The last two digits being 6 and 6 resulting in both functions being x4. Which is represented in the microcontroller code as multiplying the variable onto itself 4 times. Showing the differences between Project 1 where this was done on FPGA at module level. It shows how much simpler higher-level languages and hardware are while still being able to do increasingly complex things.



* Last digit (a) indicates f(x) as in table
* Second last digit (b) indicates g(x) as in table

# Our Solution

The project assignment gave us the option of using any current projects that we are pursuing, including our Capstone Design projects. members of our team are working on the data acquisition system for the University of Windsor Rocketry Team, which has specifications that closely match the assignment. Dr Ahmadi approved of us pivoting to this project as a substitute for the standard assignment since it was an acceptable match. below are some of the characteristics of the design project that mirror our ESD assignment:

* It is a project that is specifically designed to gather analog and digital data from an array of sensors that make up the rocket payload for the 2018 UWindsor Rocketry Team Payload design.
* It includes the use of some of the same microcontroller architectures covered by this course (Arduino)
* There are some mathematical calculations carried out to correctly capture and measure the data provided by the sensors.

# Analog I/O Theory in Microcontrollers

This ESD course covers an entire chapter on ports and protocols and a number of our lectures covered how to properly select and implement I/O communication when using embedded systems. Several portions of the learning material applied to this assignment:

1. Analog and digital communication are the two main approaches considered when examining ports and communication protocols.
   1. Analog communication has infinite resolution in terms of its dependant values (amplitude, voltage, loudness, velocity etc.)
   2. Digital communication is made up of a finite number of digits or levels that represent the sampled dependant value. It has a given minimum resolution.
   3. Continuous and discrete values represent the characteristics mentions about the two properties of signals mentioned above. they can apply to both dependant values such as voltage as well as independent values such as time.
2. There are various communication protocols that are typically used in embedded systems. Selection of the appropriate communication protocol is crucial in system design because it affects
   1. Chapter 1 considerations such as costs, timelines, amount of labour input
   2. Compatibility considerations when selecting hardware, design constraints and software strategies that will affect performance, robustness and ease of use
   3. Protocol features that may be of consideration include
      1. Error correction (built-in/updatable/configurable)
      2. Physical properties associated with the technology (length, temperature, resistance, line-of-sight, noise and interference and other operational constraints
      3. Speed and bandwidth
      4. Protocol Architecture characteristics such as serial/parallel, wired/wireless, adaptability/retro-adaptability

The most commonly used protocols for microcontrollers is likely to be serial communication. Serial communication works by sending one bit at a time between senders and recipients. The serial protocol includes parity checking, a preset *Baud* rate (communication speed setup) and start and stop bits. Analog communication in these devices typically involves sending a set voltage, of a given value relative to a common ground as well as a particular frequency and/or duty cycle. Serial communication is great for such embedded systems because:

* It is relatively cost-effective (down to a wire, and necessary for our Capstone)
* It is a good bandwidth match for embedded systems, which are usually used for on-site, low data throughput tasks requiring short length connections that require no adapters or special connectors

# Procedure

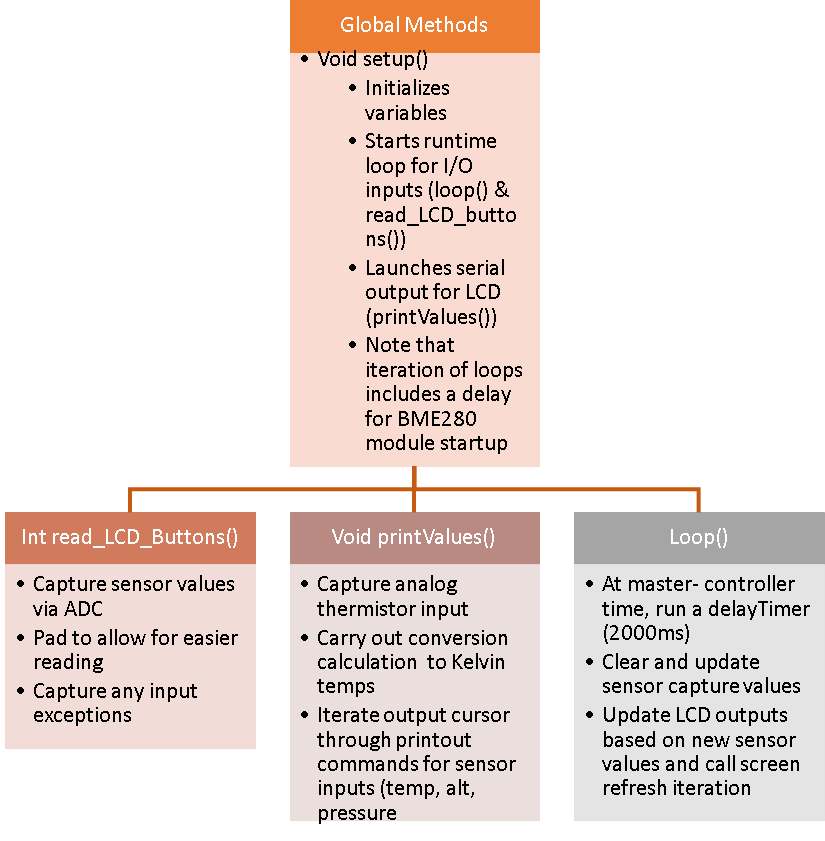
The following steps were taken to implement the project goals

1. Identify the set of sensors and input devices required for the rocket payload
   1. Temperature, and pressure (analog input 13 and digital input 24)
2. Separate these sensors into integrated circuits or direct input based on the available hardware. Also, identify the needed libraries to implement serial communication using Arduino
3. Test the sensors during the pre-flight stage to see if they accurately represent expected results

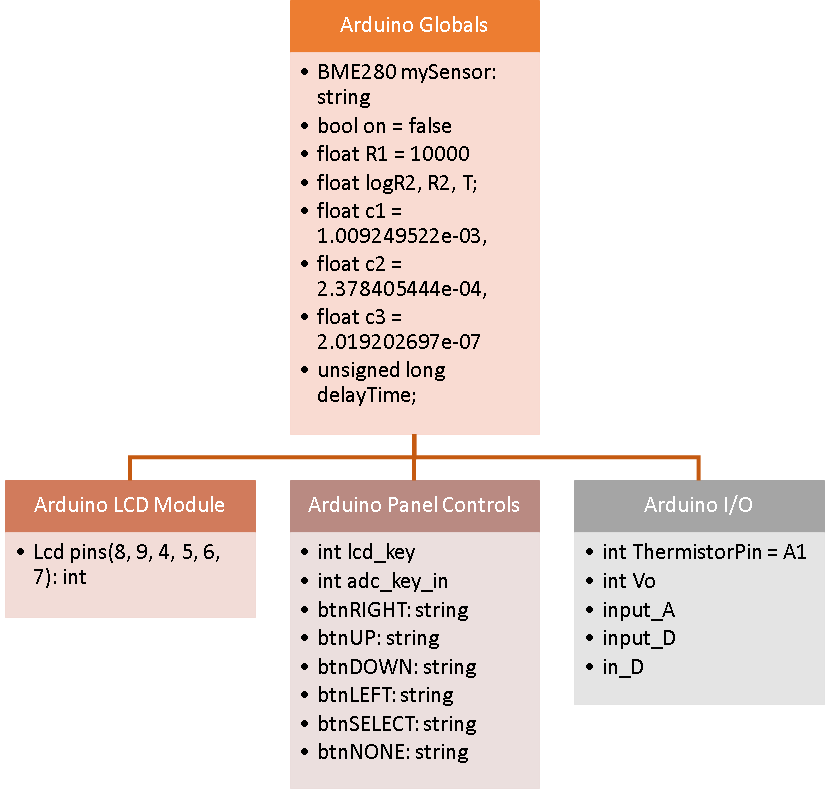
Additional complexity of the project is not explained here since it is not a part of this assignment and includes constraints that are irrelevant and may seem unnecessary or incorrect if mentioned without knowledge of the full Capstone Design requirements. Our example highlights the use of some of the sensors.

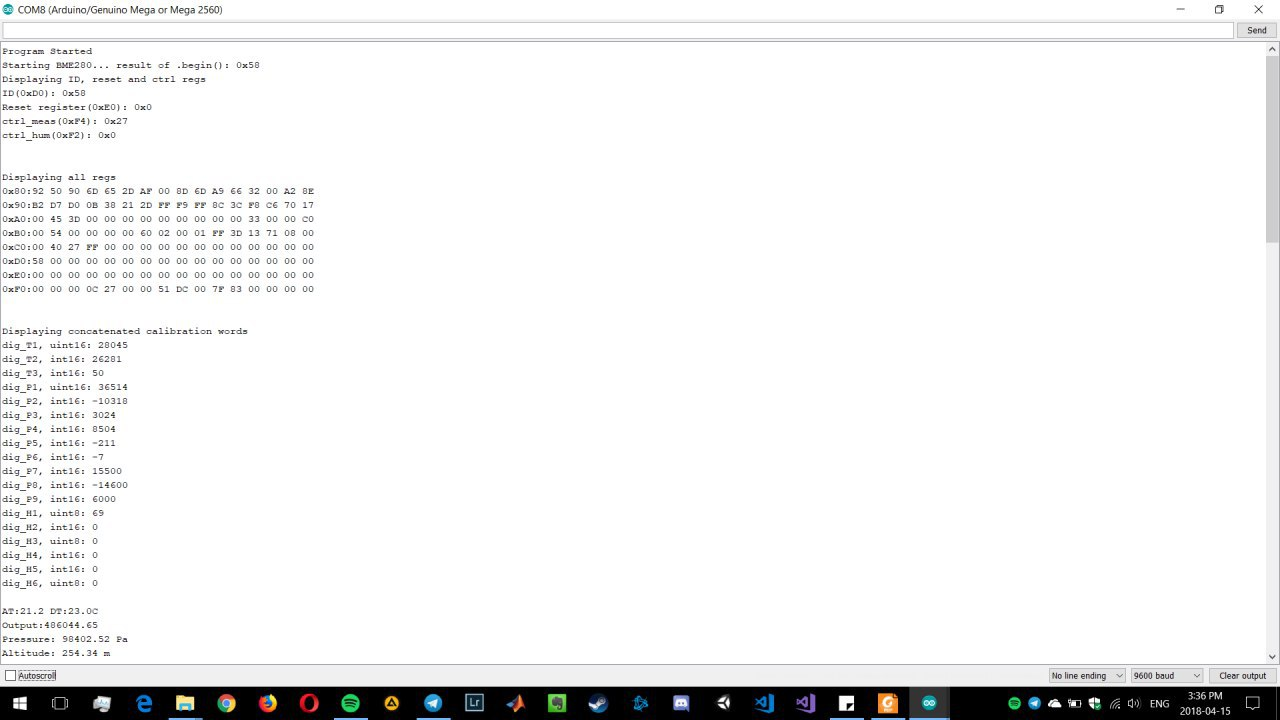
# Data/Diagrams

The following simplified UML diagrams outline the code structure for the methods and variables utilized in the system’s setup. The two diagrams are for the methods and data respectively.

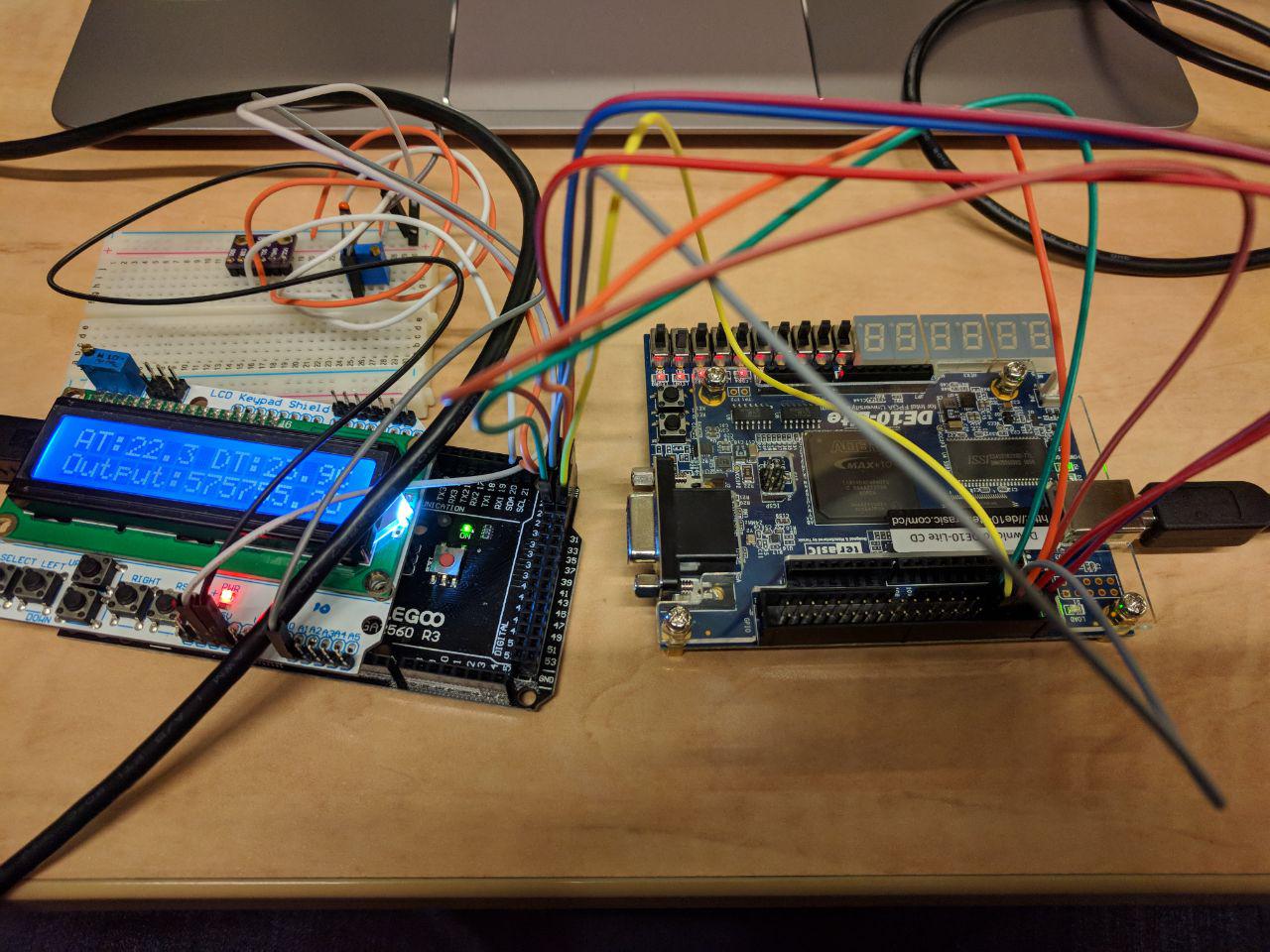


*Figure 1: Method UML Chart*

  
*Figure 2, Variable UML Chart*

**

*Figure 3: Overview of the project, which provides a visual aid in the data Being Passed through I/O and displayed in the LCD*

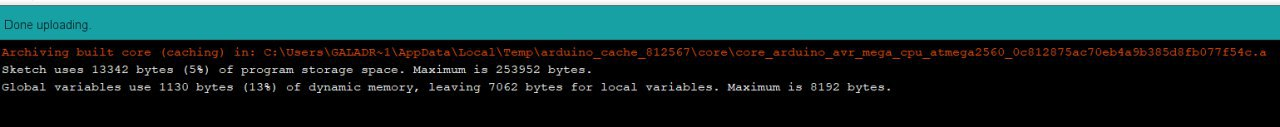


*Figure 4: Overview of the Project and the LCD Screen with the Temperature Output*

# Arduino Compiled Program Size

A key part of the assignment asked for the size of the program. Shown below is that data, with an accompanying image.

* Sketch uses 13342 bytes (4%) of total program storage
* Global variables use 770 bytes (9%) of dynamic memory



# Project Code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Project2.ino

Read values from BME280 Arduino and Thermistor to display on both Serial Window and LCD Screen

Version: 0.3

Apr 15, 2018

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <LiquidCrystal.h>

#include <stdint.h>

#include "SparkFunBME280.h"

#include "Wire.h"

#include <NeoTee.h>

#include <Math.h>

// define some values used by the panel and buttons

int lcd\_key = 0;

int adc\_key\_in = 0;

int ThermistorPin = A1;

int Vo;

float R1 = 10000;

float logR2, R2, T;

float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;

unsigned long delayTime;

double out, input\_A, input\_D, in\_D;

bool on = false;

// select the pins used on the LCD panel

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

//Global sensor object

BME280 mySensor;

//Print out Data cleanly

Print \*outputs[] = { &Serial, &lcd }; // <-- list all the output destinations here

NeoTee tee( outputs, sizeof(outputs)/sizeof(outputs[0]) );

void setup() {

Serial.begin(9600);

mySensor.settings.commInterface = I2C\_MODE;

mySensor.settings.I2CAddress = 0x76;

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

mySensor.settings.tStandby = 0;

mySensor.settings.filter = 0;

mySensor.settings.tempOverSample = 1;

mySensor.settings.pressOverSample = 1;

mySensor.settings.humidOverSample = 1;

Serial.print("Program Started\n");

Serial.print("Starting BME280... result of .begin(): 0x");

//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.

Serial.println(mySensor.begin(), HEX);

Serial.print("Displaying ID, reset and ctrl regs\n");

Serial.print("ID(0xD0): 0x");

Serial.println(mySensor.readRegister(BME280\_CHIP\_ID\_REG), HEX);

Serial.print("Reset register(0xE0): 0x");

Serial.println(mySensor.readRegister(BME280\_RST\_REG), HEX);

Serial.print("ctrl\_meas(0xF4): 0x");

Serial.println(mySensor.readRegister(BME280\_CTRL\_MEAS\_REG), HEX);

Serial.print("ctrl\_hum(0xF2): 0x");

Serial.println(mySensor.readRegister(BME280\_CTRL\_HUMIDITY\_REG), HEX);

uint8\_t memCounter = 0x80;

uint8\_t tempReadData;

lcd.begin(16, 2); // start the library

lcd.setCursor(0,0);

delayTime = 1000;

Serial.println();

}

void loop() {

printValues();

delay(delayTime);

lcd.clear();

} //Loop end

void printValues() {

Vo = analogRead(ThermistorPin);

R2 = R1 \* (1023.0 / (float)Vo - 1.0);

logR2 = log(R2);

T = (1.0 / (c1 + c2\*logR2 + c3\*logR2\*logR2\*logR2));

T = T - 273.15;

in\_D = mySensor.readTempC();

input\_A = T\*T\*T\*T,1;

input\_D = in\_D\*in\_D\*in\_D\*in\_D;

out = input\_A + input\_D;

lcd.setCursor(0,0);

tee.print("AT:");

tee.print(T, 1);

tee.print(" DT:");

tee.print(mySensor.readTempC(), 1);

tee.println("C");

lcd.setCursor(0,1);

tee.print("Output:");

tee.println(out);

//tee.print( ',' );

Serial.print("Pressure: ");

Serial.print(mySensor.readFloatPressure(), 2);

Serial.println(" Pa");

//tee.print( ',' );

Serial.print("Altitude: ");

Serial.print(mySensor.readFloatAltitudeMeters(), 2);

Serial.println(" m");

Serial.println();

}

# Discussion

To explain the code used let’s start with going through the included libraries. Math is used for the various calculations performed such as the ones required for the Analog Temperature and the project requirements. LiquidCrystal and stdint are required for the LCD Screen functions. SparkFunBME280 is for the digital sensor along with Wire. NeoTee is to output the results to both Serial Terminal and LCD Screen at the same time. There is some additional output detailed in the Serial terminal such as Pressure but not shown on the LCD Screen due to wanting to show the basics all the time. It was chosen to show the Analog and Digital temperatures on the LCD Screen.

After the libraries variables are declared. Some of them are related to the Analog Temperature to calculate some specific values while other variables are there to be global so that all functions can access them. LCD screen pins are declared and sensor is declared as an object. Next the NeoTee code is organized and outputs selected.

Setup performs the required sensor setup and the LCD screen and baudrate declarations. The loop calls the printValues function and clears the LCD every loop to refresh. The sensors part of the code is the printValues function where the analog temperature values are calculated. The required project calculations are performed. And the values are neatly displayed.

The data acquisition design procedure was simple to implement considering the assignment specifications and the concurrent course objectives that we had covered. It was good practice for applying some of the theory that we have covered in the course as well as an opportunity to consider some of the constraints that we have a better grasp after covering the earlier chapters in the course. Areas where we would like to improve such a design project include assessing our design’s performance under the conditions of the rocket’s launch and payload dispatch.

While usage of communication protocols in this example was easy, we believe that a set number of parameters to use as a benchmark would have allowed us to spend more time intuitively looking at areas of improvement or better use. Our results matched the desired goals of correctly implementing the data transfer and device commands.

# Conclusion

The overall project goal was achieved. In addition, the required tasks were completed without us reaching any of the obvious failures (limited space due to bloated software/poor selection of hardware). We managed to track the data transfer as well as capture this for observation and assignment reporting.

# References

1. Course Materials: Embedded Systems Design: A Unified Hardware and Software Introduction, Vahid, Frank, Givargis, Tony. John Wiley & Sons, 2002
2. Course Slides and Lecture Notes
3. Arduino ATMEGA2560 Datasheet, Accessed March 24 2018, <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf>, www. microchip.com
4. Serial Communication Tutorial, Sparkfun Electronics, <https://learn.sparkfun.com/tutorials/serial-communication/all>, Accessed March 25 2018.