2006-496: USING ROCKETS TO UNIFY TOPICS IN AN ELECTRO-MECHANICAL ENGINEERING TECHNOLOGY INSTRUMENTATION COURSE

Dale Litwhiler, Pennsylvania State University-Berks

Dale H. Litwhiler is an Assistant Professor at Penn State Berks in Reading, PA. He received his B.S. from Penn State University, his M.S. from Syracuse University and his Ph.D. from Lehigh University all in electrical engineering. Prior to beginning his academic career in 2002, he worked with IBM Federal Systems and Lockheed Martin Commercial Space Systems as a hardware and software design engineer.

Using Rockets to Unify Topics in an Electro-Mechanical Engineering Technology Instrumentation Course

Abstract

Model rockets are being used at Penn State Berks to unify topics in an electro-mechanical engineering technology instrumentation course. Model rockets provide an exciting platform on which to carry many types of devices and sensors. Throughout the semester, several types of sensors and transducers are introduced and studied. Sensors include thermistors, micromachined accelerometers and integrated pressure transducers. The physics, construction and characteristics of these sensors are discussed in the course lectures. The students also receive hands-on experience with many of the sensors through the course's laboratory experiments. Analog to digital conversion techniques and data acquisition systems are also studied in this course. To help pull together the topics and concepts discussed in class, a rocket payload data acquisition system is employed. As each device is studied, its application to the payload system is presented and discussed. A thermistor is used to measure the air temperature at various altitudes. A micromachined accelerometer is used to measure the acceleration of the rocket during launch and throughout the mission. Integrated silicon pressure transducers are used to measure both altitude and speed of the rocket. The axial speed of the rocket is determined by using the body of the rocket as a Pitot tube together with a differential pressure transducer. The timing, power management, control, measurement and data storage for the entire payload is handled by an embedded PIC™ microcontroller. A rocket launch date is set near the end of the semester with a well-publicized formal countdown commenced well in advance of the launch to help promote interest and build excitement for the event. The students are active participants in the launch and recovery operations. The raw data collected during the flight is uploaded from the payload memory for interpretation and analysis by the students. A flight performance report based on the data is submitted by each student. This paper presents and discusses the details of the rocket system, the role of the project in the course and feedback from the students involved.

Introduction

The use of model rockets in engineering education is well documented [1-4] Students generally find working with model rockets an exciting way to learn engineering concepts. The experience usually takes the class out of the classroom and, due to the nature of model rocketry, this experience usually occurs on days of nice weather. The anticipation of a countdown and the thrill of watching the launch is an added bonus to the experience. Besides the pure enjoyment of a launch, the model rocket also makes an excellent platform on which to attach a plethora of useful educational payloads. The work presented here puts an emphasis on the sensor payload but also exploits the inherent fun as the catalyst to learning.

As part of a junior-level instrumentation and measurement theory course at Penn State Berks, a model rocket packed with sensors is used to help unify the concepts of the course. This course is part of the electromechanical engineering technology program. In this course, the theory of temperature, pressure, fluid flow, displacement, velocity, and acceleration measurement are covered in detail. Simultaneously, the laboratory component of the course provides the students the opportunity to get their hands on many of the sensors and transducers used to measure these physical quantities. To have a bit of fun with some of the sensors and help give the students a bigger picture of how they can be used, a portable data acquisition and logging system has been designed and implemented as a model rocket payload.

Throughout the semester, as the theory and characteristics of various sensors are studied, the application of each to the rocket payload system is also discussed. Using the datasheets for each of the sensors, the students convert the raw data into the desired measured quantity. The students also determine the relationship between the quantities measured by each sensor (if there is any) to help collaborate and substantiate the recorded data. The significance of the data derived from each sensor in determining part of the rocket mission performance is also an important concept of the exercise. A formal mission performance report is submitted by each student containing their interpretation of the flight data.

The model rocket launch date is scheduled near the end of the semester. This allows time for study of the pertinent devices and techniques prior to launch while still affording enough time for preparation of the flight report by the end of the semester. The countdown time is announced during each class meeting and is also displayed on the class webpage. This practice provides a cadence or metronome-like pace for the course which helps to keep a high level of student interest by providing a goal to work towards (or a light at the end of the tunnel?).

System Overview: Hardware

The physical layout is designed around the EZ Payloader model rocket manufactured by Quest Aerospacet[5]. A diagram of the rocket is shown in Appendix A at the end of this paper. This rocket was chosen because of its unique payload section which is separate from the parachute stowing area. In model rocketry the parachute is deployed by the, “Ejection charge” that is produced by the rocket engine after the thrust charge is depleted. The ejection charge effectively forces the rocket stages to separate and propels the parachute(s) out of the storage chamber. By having a separate payload section, the sensors can be shielded from the violent pressure effects of the ejection charge.

The diameter and length of the combined payload and hollow nose cone sections place constraints on the physical design. A two-sided printed circuit board (PCB) was designed to hold the control and data storage electronics. The PCB was designed with free software from ExpressPCB.[6] The sensors are located within the payload and nose sections but not directly on the main circuit board. This arrangement allows for different types and packages of sensors to be used in future projects without changing the circuit board design.

A block diagram of the data acquisition system hardware is shown in Figure 1. Power for the system is provided by a small 12V battery. The regulated 5V power supply for the electronics is controlled by the linear voltage regulator. An integrated circuit accelerometer is used to measure the acceleration of the rocket during launch and throughout the mission. An integrated silicon absolute pressure transducer is used to determine the altitude of the rocket by measuring the difference in atmospheric pressure during flight. The axial speed of the rocket is determined by using the body of the rocket as a Pitot tube together with a differential (gauge) pressure transducer. A nominal 3kΩ thermistor is used to measure the air temperature at various altitudes. Sensor data is stored in a non-volatile 64kbit CMOS EEPROM memory device. The timing, power management, control, measurement and data storage for the entire payload is handled by an embedded microcontroller. A momentary contact switch, blue LED and serial port connections are located on the exterior of the rocket and are used for operator intervention, status and data retrieval respectively as described later.

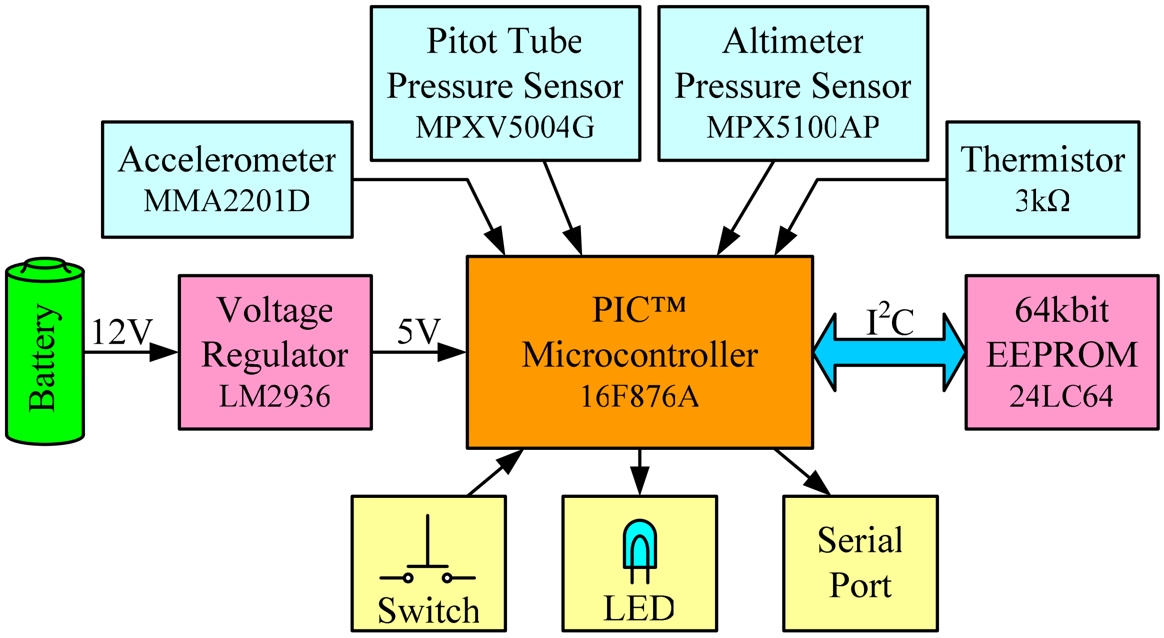


Figure 1. Model Rocket Payload Data Acquisition System

Weight is a primary concern for any rocket-launched system. To reduce system weight, a small battery is used to power the system. The total mission life is only about two minutes. This short life together with the measured system power requirements allowed for the use of a 12V battery typically used in automobile keyless entry system remote controllers (Radio Shack 23-279). The battery with wires attached (soldered) is shown in Figure 2.

The LM2936 +5V linear voltage regulator was chosen for the system because of its small quiescent current draw (nominally 15µA). This is an important feature because of the long periods of time in which the system is "sleeping." Battery energy is precious so during much of the mission, the microcontroller is in sleep mode. During this time, the current load on the 5V bus is very small (~ 40µA) therefore it is important that the voltage regulator not waste the battery energy. (Compare this with a quiescent current of about 3mA for the industry standard LM78L05!) Figure 3 shows the location of the voltage regulator on the circuit board (TO-92 package).

The axial acceleration of the rocket is measured with a Freescale™ MMA2201D ±40g surface micromachined integrated circuit accelerometer.[7] The students receive hands-on experience with an accelerometer in the same product family, but smaller acceleration range, during a prior laboratory experiment so they are quite familiar with the characteristics of the device. The MMA2201D provides a ratiometric zero-g output voltage of $0.5V_{DD}$. The ratiometric sensitivity of the accelerometer is 10mV/g/V. The accelerometer is mounted on an auxiliary circuit board (to facilitate connecting wires to the small lead frame) and connected to the main circuit board as shown in Figure 2.

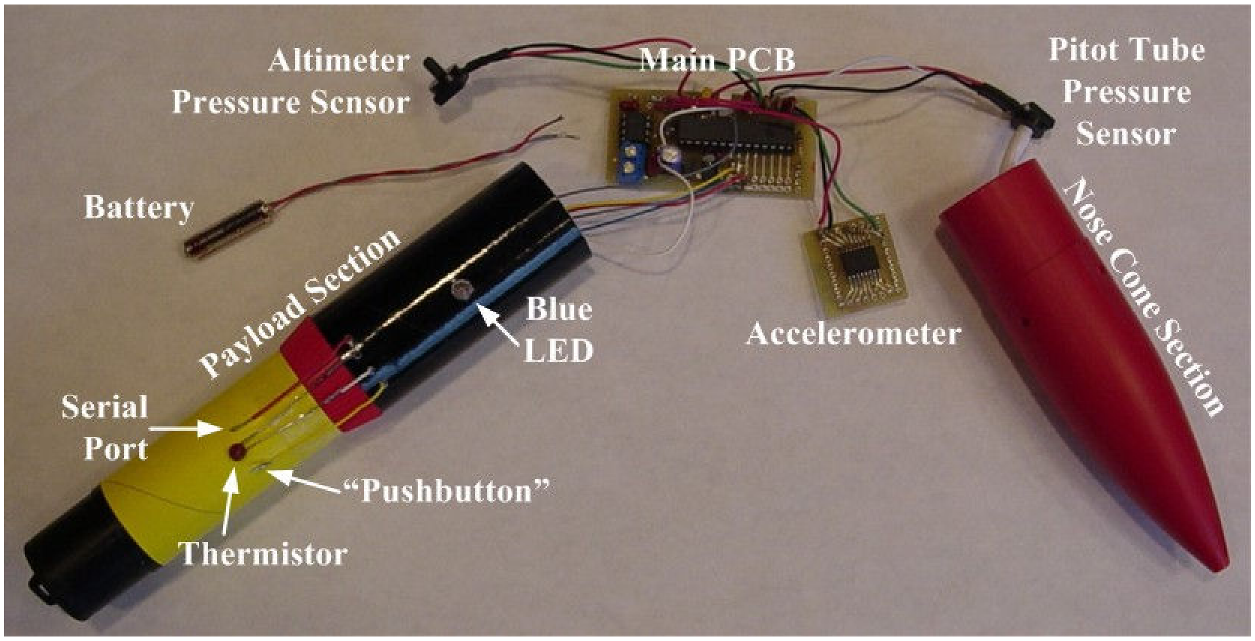


Figure 2. Photograph of Payload and Nose Cone Sections

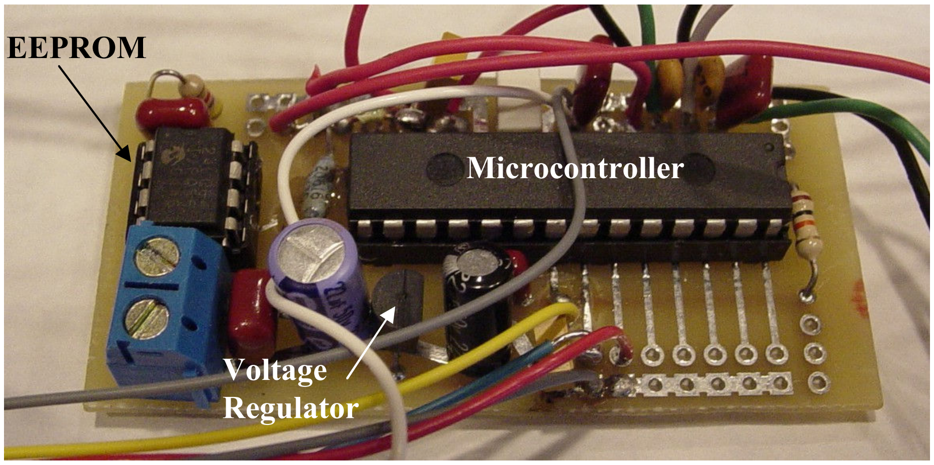


Figure 3. Photograph of Main Circuit Board

To measure the axial velocity of the rocket, the nose cone was modified to create a Pitot-static pressure probe. A small hole was drilled in the tip to allow plastic tubing to be passed through.

The area between the tubing and the cone is sealed with glue (Silicone RTV). The tubing exposed on the outside of the nose tip was trimmed flush with the tip. This forms the total pressure port of the probe. The other end of the tubing is connected to the input port of the differential pressure sensor (Freescale™ MPXV5004G). Four small equally spaced holes were drilled around the circumference of the nose cone beyond the tapered section. These holes form the static pressure ports of the Pitot tube.[8,9] Figure 4 shows a photograph of the modified nose cone section. The air pressure within the payload section is therefore the static pressure of the Pitot-static probe. The difference between the total pressure and static pressure is measured by the sensor and used to estimate the axial speed as,[8]

$$v = \sqrt{\frac{2\left(P_{Total} - P_{Static}\right)}{\rho_{Air}}} \qquad (\rho_{Air} \text{ is the density of air})$$

The sensitivity of the pressure sensor is ratiometric with the supply voltage (for $V_S = 5.0V \pm 0.25Vdc$). The output voltage to supply voltage ratio is given by the following transfer function:[10]

$$\frac{V_{Out}}{V_S} = \left(0.2P + 0.2\right) \qquad (P \text{ is pressure in kPa})$$

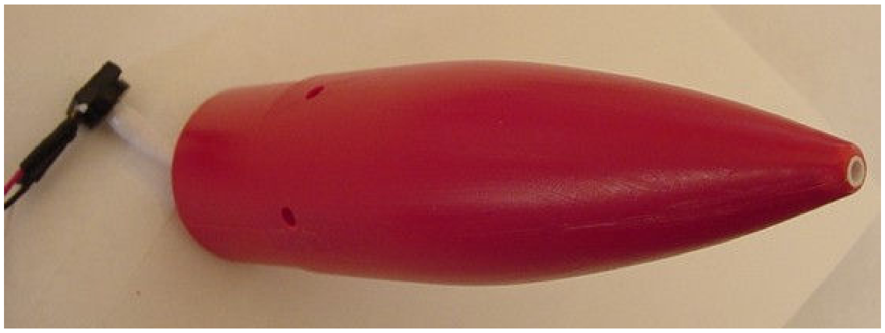


Figure 4. Rocket Nose Cone Modified to form Pitot-Static Probe

During the semester, the MPXV5004G is discussed and analyzed in detail in class. The students also perform manometer experiments in the laboratory with this device. The characteristics of the device are measured and possible applications, including Pitot-static probes are studied.

The altitude of the rocket is estimated by measuring the difference in atmospheric pressure with respect to that of the launch pad. The pressure change is related to altitude (h) using the following expression:

$$h = \frac{\Delta P}{g\rho_{Air}} \qquad (g = 9.81 \text{ m/s}^2)$$

The pressure is measured with an integrated silicon absolute pressure sensor (Freescale™ MPX5100AP). This sensor is housed in the payload section. The static atmospheric pressure inside the payload section is that of the surrounding atmosphere due to the static port holes required for the Pitot tube as described earlier. The full range of this sensor is 15 – 115 kPa (2.2

– 16.7 psi). The change in output voltage produced by the altitude change of the rocket is quite small but is still measurable and sufficient to describe the altitude profile with rough accuracy. The altimeter sensor scheme will be upgraded in future flights. The sensitivity of the pressure sensor is ratiometric with the supply voltage (for $V_S = 5.0V \pm 0.25Vdc$). The output voltage to supply voltage ratio is given by the following transfer function:[11] (Note: The referenced datasheet wrongly indicates "+" rather than "-" in the following equation!)

$$\frac{Vout}{Vs} = 0.009P - 0.095 \qquad \text{(P is pressure in kPa)}$$

Again, the behavior and applications of the MPX5100AP are studied by the students during the semester. Adding this device to the payload sensor suite helps them understand where it can be used (and where it may not be the best choice).

The atmospheric temperature is measured during the flight using a thermistor (Jameco part number 207466CE) that is mounted on the exterior of the payload section as shown in Figure 2. The thermistor resistance is a nonlinear function of its temperature. To determine the thermistor resistance, the device is placed in a voltage divider circuit with a fixed 3.3kΩ resistor as shown in Figure 5.

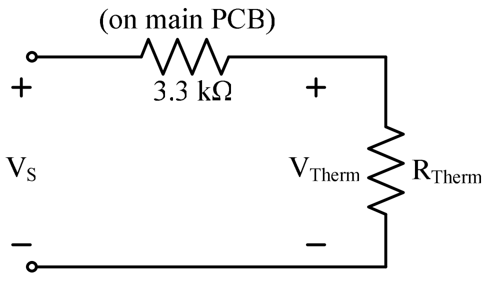


Figure 5. Thermistor Voltage Divider Circuit

The thermistor voltage can be expressed as a ratiometric function of the voltage divider input voltage. By measuring this ratio (as is done with the analog to digital converter), the thermistor resistance can be calculated:

$$R_{Therm} = \frac{\frac{V_{Therm}}{V_S}(3.3k\Omega)}{\left(1 - \frac{V_{Therm}}{V_S}\right)}$$

Finally, the thermistor resistance is converted to temperature using the exponential relationship:

$$T(°C) = \left[\frac{1}{\beta}\ln\left(\frac{R}{R_0}\right) + \frac{1}{T_0}\right]^{-1} - 273.15$$

The students were required to determine β from calibration data obtained for this particular thermistor. Thermistors of this family are also studied in class. The static and dynamic behavior of these devices is also measured by the students in the laboratory. Because of the slow thermal response of the thermistor compared to the ascent time of the rocket, temperature data is only acquired during the parachuted descent phase of the flight.

The data acquired during the mission is stored in a CMOS EEPROM memory device (Microchip™ 24LC64). This device has a capacity of 64kbit arranged as 8kbytes of 8 bit words. Data is written to and read from the device using the I^2C serial data bus protocol.[12] This device was chosen because it has sufficient capacity for the amount of data acquired and it requires very little power to operate.

All of the data acquisition, control and timing is performed by the Microchip™ PIC™ 16F876A microcontroller unit (MCU).[13] This device was chosen for its multiple analog inputs with a 10-bit A/D converter, multiple digital inputs and outputs, low power requirements, ease of programming, and the author's personal experience with its use. The MCU is located on the main PCB as shown in Figure 3. A schematic of the main PCB is included in Figure 6. The software for the MCU was written and compiled using the PICBASIC PRO™ Compiler from microEngineering Labs, Inc.[14] The software is discussed in detail later in this paper.

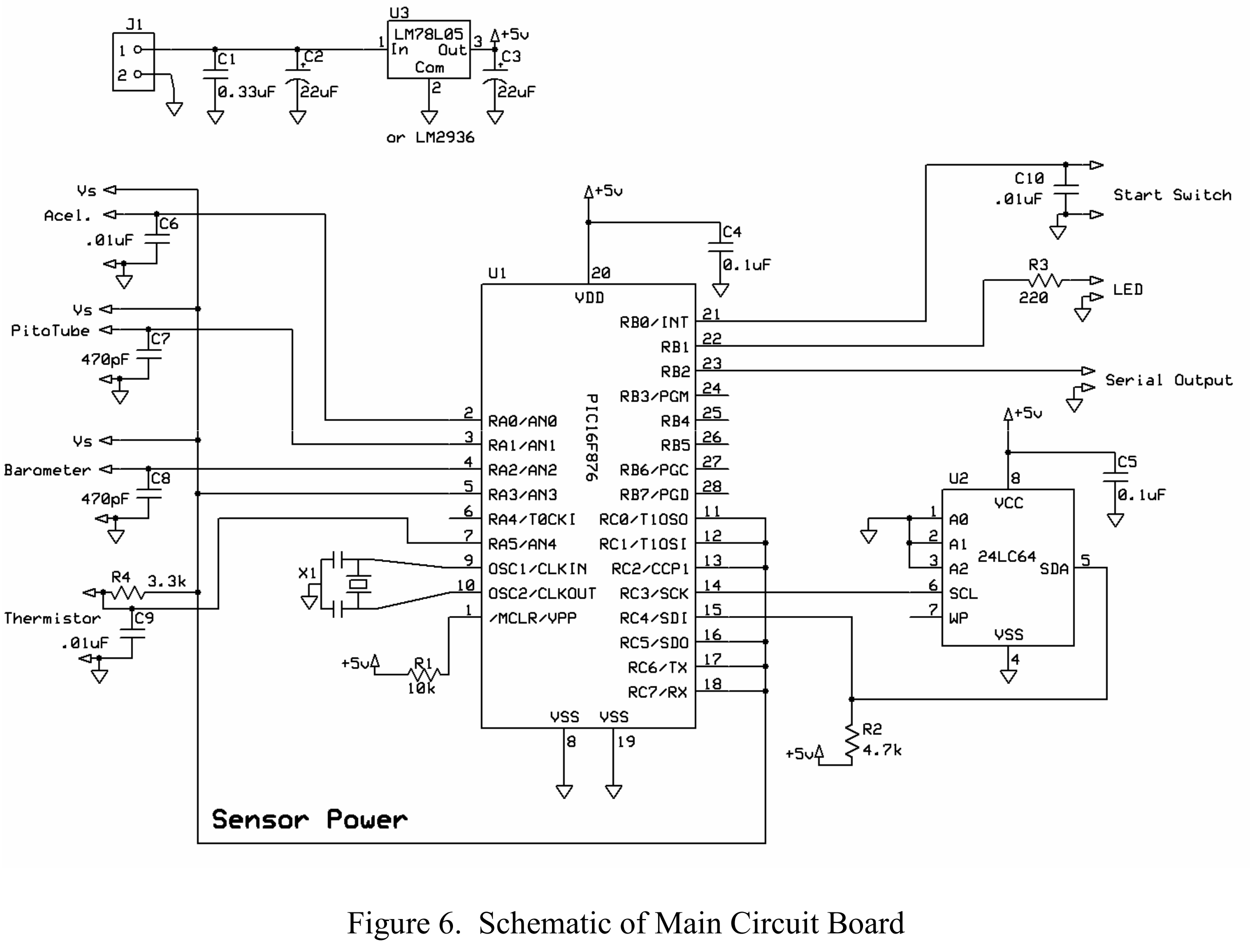


Figure 6. Schematic of Main Circuit Board

For the rocket data acquisition system, the MCU instruction clock is derived from a 10MHz ceramic resonator located on the main PCB. The MCU analog inputs are configured as four ground-referenced (single-ended) measurement inputs. The reference voltage for the A/D converter is configured as an external voltage applied to analog input 3 (AN3). Power for the sensors is provided through six of the digital I/O pins connected in parallel. In this way, the power to the sensors can be turned on and off under software control to help conserve battery energy and no external transistor switch is required. Other digital I/O pins are used as user interface controls.

One digital input pin of the MCU is connected to an external momentary contact switch. This switch (which is just a piece of wire protruding through the payload section that is momentarily touched to the ground side of the thermistor) is used to wake the MCU from "Sleep" mode as is described in the software section. A digital output pin is connected to a bright blue LED located on the outside of the payload section. This LED is used to provide status to the user during launch and recovery as described later. Finally, another digital output pin is used as the serial output port to upload the stored data to a PC after the mission. Again, this is just a piece of wire protruding through the payload section to which an external serial cable can be clipped.

System Overview: Software

The control and data acquisition software for the rocket MCU was written and compiled using the PICBASIC Pro compiler. This is a higher level language that has commands and structure in a BASIC-like manner. There are many special function commands built into PICBASIC Pro that provide for quick software development. Some of these routines include accurately timed pauses, I2C bus communication and serial port (RS-232) protocols. This programming environment also allows assembly code instructions to be embedded within the BASIC code. This is especially helpful for routines that require tight control of execution timing.

Figure 7 shows a simplified flowchart of the rocket mission software. A detailed listing of the PICBASIC Pro code is included in Appendix B. The mission can be broken into five main phases: The pre-launch phase, the countdown phase, the launch-ascent phase, the descent phase and the data recovery phase. The block diagram and code listing are color coordinated to indicate these phases.

The pre-launch phase begins when the battery is connected to the main PCB. The MCU powers up and configures the analog and digital inputs and outputs. The MCU then enters sleep mode to conserve battery energy. This gives the operator time to stow the sensors and the main PCB, install the nose cone section, and place the rocket on the launch pad. The system is now ready and waiting for the external switch to be pressed to wake it up.

When the external switch is pressed, the MCU awakens and begins the countdown phase. During this phase, the sensors are powered up, their output voltages are measured and stored in memory, and powered back down again. The blue LED is flashed for 100ms. This process repeats once each second for 10 seconds. The sensor measurements provide a baseline for calibration and the flashing LED provides a countdown cadence for the launch team and onlookers. (During the final second, the LED remains on for one second to indicate to the launch operator that it is time to press the ignition button to start the rocket engine.)

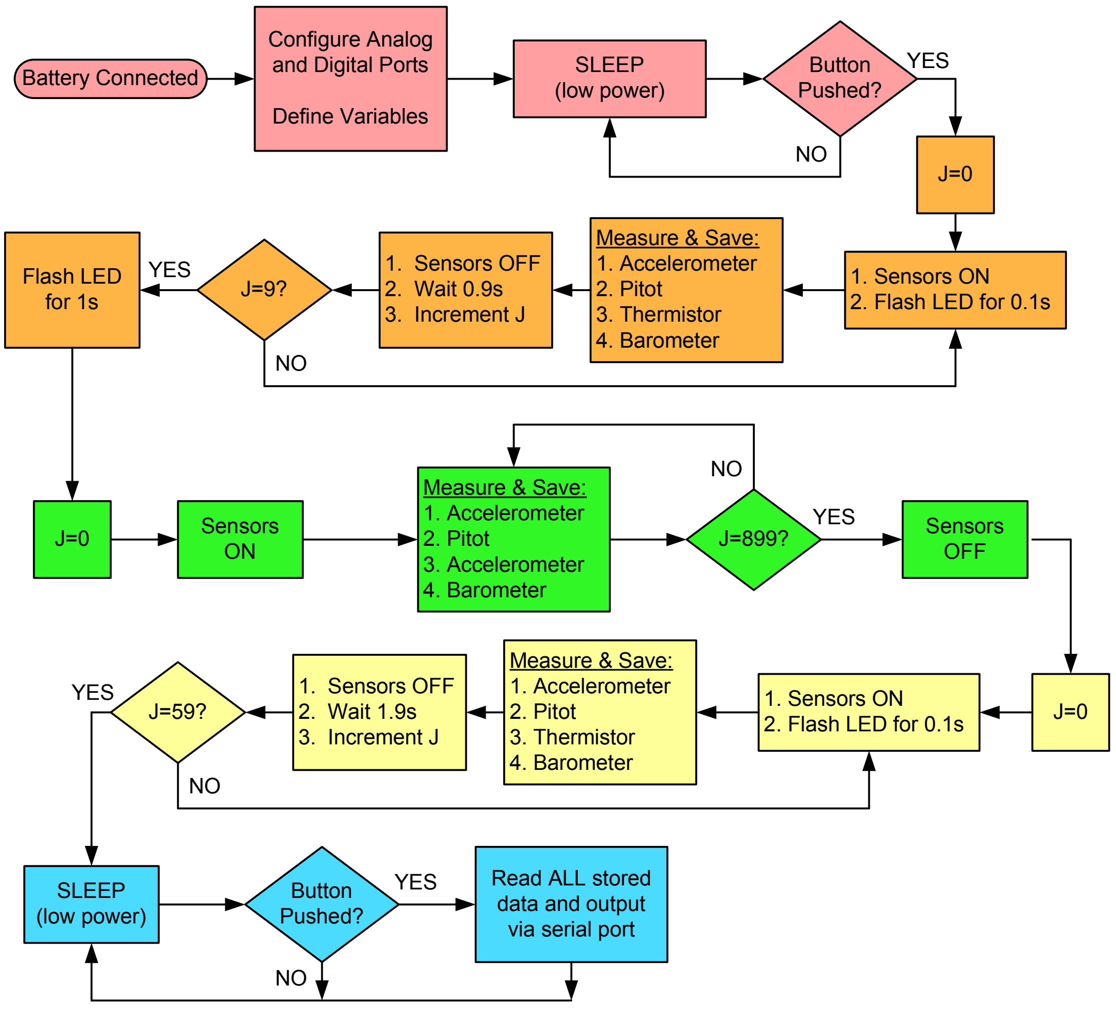


Figure 7. Flowchart of the Rocket Mission Software

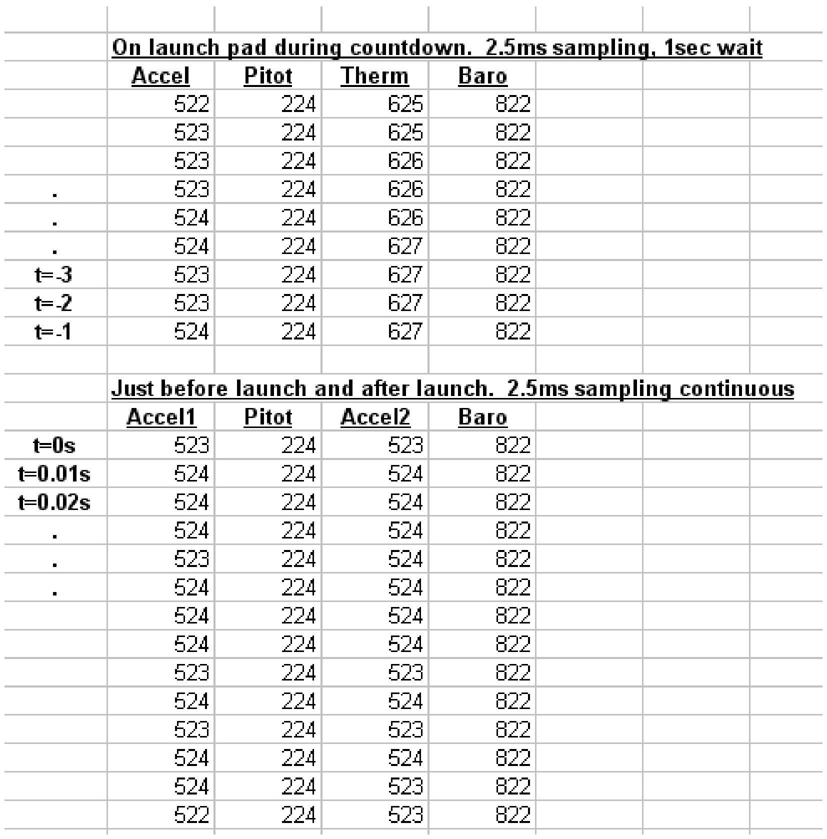
During the launch-ascent phase, the data acquisition is performed continuously at a 400Hz rate. The sensor measurement sequence is as follows: 1) Accelerometer, 2) Pitot pressure, 3) Accelerometer again, 4) Altimeter pressure. The accelerometer measurement is repeated to increase the sampling frequency of this sensor because of the higher frequency content possible. Data is written to EEPROM after each set of four measurements is complete. The thermistor is not measured at all during this phase because of its long time constant compared with the ascent time of the rocket. The launch-ascent phase lasts for 9 seconds. This is more than enough time for the rocket to reach apogee.

Immediately after the nine second launch-ascent phase, the descent phase begins. In this phase, the parachute has been deployed and the rocket is slowly floating back toward earth. Sensor data is now taken from all four sensors at two second intervals. Between measurements, the sensors are powered down to conserve battery energy. This phase lasts for 120 seconds which should be adequate for the rocket to return to the ground. The system then returns to the low power sleep mode awaiting recovery.

After the rocket has been retrieved, the data recovery phase begins. The external serial port wire and ground wire are connected to the serial port of a PC. When the rocket's external switch is pressed, the MCU awakens, reads back all of the saved data, and sends it to the serial port connection in spreadsheet format (four columns, space delimited) with blank rows separating the various phases of data. The MCU then goes to sleep again. This process can be repeated as needed to recovery the stored data. A simple LabVIEW serial port reading application is used to display and save the downloaded data.

Data and Results

Because of the configuration of the A/D reference voltage input as discussed earlier, the data is in ratiometric format. Each measurement provides an integer value between 0 and 1023 which represents what fraction of 1023 is measured. Figure 8 shows a sample of the data displayed in Excel. The students can use Excel to convert this data to the appropriate sensor output using the ratiometric transfer functions shown earlier.



	On launch pad during countdown. 2.5ms sampling, 1sec wait			
	Accel	**Pitot**	**Therm**	**Baro**
	522	224	625	822
	523	224	625	822
	523	224	626	822
.	523	224	626	822
.	524	224	626	822
.	524	224	627	822
t=-3	523	224	627	822
t=-2	523	224	627	822
t=-1	524	224	627	822
	Just before launch and after launch. 2.5ms sampling continuous			
	Accel1	**Pitot**	**Accel2**	**Baro**
t=0s	523	224	523	822
t=0.01s	524	224	524	822
t=0.02s	524	224	524	822
.	524	224	524	822
.	523	224	524	822
.	524	224	524	822
	524	224	524	822
	524	224	524	822
	523	224	523	822
	524	224	524	822
	523	224	523	822
	524	224	524	822
	524	224	523	822
	522	224	523	822

Figure 8. Excerpt from Excel Spreadsheet of Rocket Mission Data

Appendix C shows a presentation of launch-ascent phase results submitted by one of the course's students. The acceleration data very closely follows the shape and timing of the thrust curve for the C6-3 engine provided by the rocket engine manufacturer.[15]

Several students performed discrete integration using Excel and MathCAD on the accelerometer data to obtain an estimate of rocket velocity with excellent correlation between the expected and measured values. Similarly, students were able to achieve very good results by performing discrete integration on the velocity data to estimate the rocket altitude. The small change absolute pressure sensor output voltage together with the 10 bit resolution of the A/D converter produces poor altitude resolution. The integrated velocity data provides a higher resolution estimate of the altitude however, the initial value offset produces some ambiguity in the actual altitude result.

The results obtained for the atmospheric temperature were not as impressive. Figure 9 shows a plot of temperature versus altitude that was calculated by one of the students. Notice that the total temperature change is only about one degree Fahrenheit. This together with the poor altitude resolution makes this data inconclusive.

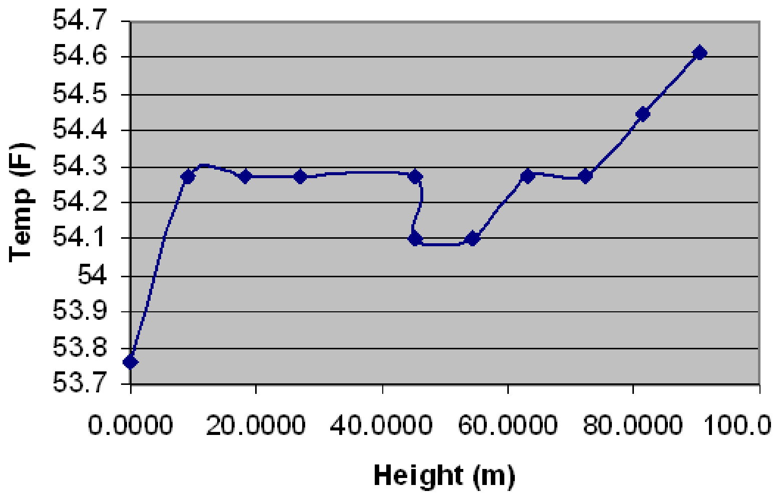


Figure 9. Plot of Temperature vs. Altitude

Conclusions and Outcomes

A model rocket provides an excellent vehicle on which to attach a host of instruments relevant to a modern engineering instrumentation course. Even the small suite of sensors presented here produces a plethora of useful and interesting data. The nature of a model rocket flight provides a rich set of parameters to measure. The choice of sensors and instrumentation helps to produce relationships between the data with which the students are familiar: acceleration, velocity and displacement. In this way, the rocket mission provides a unifying experience for the students. The hope is that this experience helps the students see the bigger picture of possible applications of the devices studied in the course. The anticipation and excitement of the launch are also very effective in attracting and maintaining student interest.

The cost of the rocket system materials was very reasonable. The rocket and launch accessories were obtained from a local hobby shop. The sensors and electronic components are typically available through common electronics distributors however those used here were obtained as free samples directly from the manufacturers. The circuit boards for the microcontroller and the accelerometer were part of a larger PCB containing other circuits for engineering use. The entire PCB was obtained for $59 from ExpresssPCB.com. Table 1 shows a breakdown of system costs.

Table 1. Rocket System Material Costs

Item	**Cost**
Quest Aerospace EZ-Payloader Rocket	$11.00
Rocket Engines (3 per pack)	7.00
Launch Pad and Controller	15.00
Circuit Board (Piggy-backed with other circuits)	59.00
Total	$92.00

The construction and modification of the rocket requires only simple tools and basic skills. The Pitot-static probe construction seems simplistic but excellent velocity results were obtained. The velocity profile fits very well with the integrated acceleration data profile. (Or conversely, the acceleration profile fits very well with the differentiated velocity data profile)

The microcontroller and software used here was chosen for its familiarity to the author. An alternate MCU and development software could be used to tie the rocket project to an associated microprocessor course to even further unify topics. In the work presented here, the emphasis is on the sensors rather than the MCU.

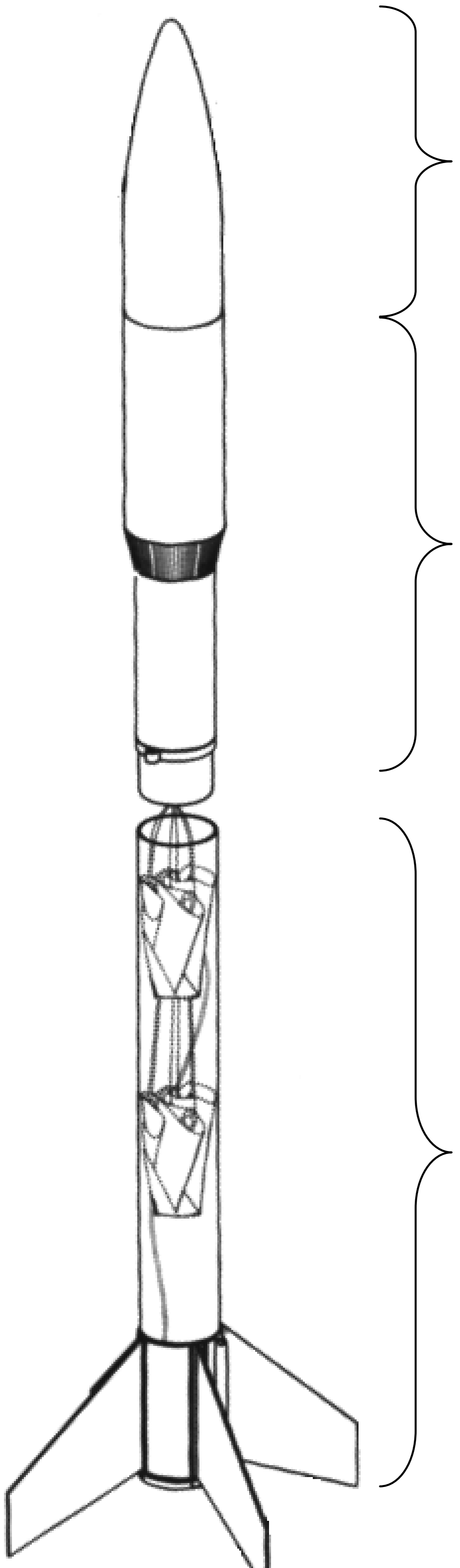
The analysis performed by the students shows a high level of interest and enthusiasm. During the semester as each sensor is studied and its role in the rocket payload is discussed, the students have at least one concrete reason for *why* they are learning the material. This is a great motivator. The students are very motivated to analyze the data to find out how many “gs” (ghēz) the rocket “pulled,” and how fast and high the rocket went. In this class, the students really are *rocket scientists.*

Acknowledgements

The author wishes to thank the students of the Penn State Berks EMET 330 course for their participation in the rocket launching experiment and activities. A special thanks to the Penn State Berks electrical engineering technology laboratory supervisor, Mr. Jeff Wike, for his advice and help with assembling the components of the rocket system and launch day activities.

References

1. Hunt, T. S., Miller, D. P., Ortega, E., Striz, A. G., “Rocketry: System Development Experiences and Student Outreach,” Proceedings of the American Society for Engineering Education Annual Conference and Exposition, 2004.
2. Bales, J. W. and Consi, T. R., “Smart Rockets: A Hands-on Introduction to Interdisciplinary Engineering,” Proceedings of the American Society for Engineering Education Annual Conference and Exposition, 2003.
3. Niemi, E. E., “Using Model Rocketry to Introduce Students to Aerospace Engineering,” Proceedings of the American Society for Engineering Education Annual Conference and Exposition, 2002.
4. Bales, J. W. and Consi, T. R., “Smart Rockets: Data Acquisition in Model Rocketry,” Circuit Cellar INK, Issue 98, pp. 12-23, 1998, Circuit Cellar Inc., Vernon, CT. (Also available online at: http://www.circuitcellar.com/library/print/0998/Consi98/index.htm)
5. http://www.questaerospace.com/
6. http://www.ExpressPCB.com
7. http://www.freescale.com/files/sensors/doc/data_sheet/MMA2201D.pdf
8. Figliola, R. S., and Beasley, D. E., *Theory and Design for Mechanical Measurements*, 3rd ed., John Wiley & Sons, Hoboken, NJ, 2000, pp. 375-376.
9. Doebelin, E. O., Measurement Systems Application and Design, 4th ed., McGraw-Hill, 1990, pp. 528-529.
10. http://www.freescale.com/files/sensors/doc/data_sheet/MPXV5004G.pdf
11. http://www.freescale.com/files/sensors/doc/data_sheet/MPX5100.pdf
12. http://ww1.microchip.com/downloads/en/devicedoc/21189f.pdf (Microchip™ 24LC64 datasheet website.)
13. http://ww1.microchip.com/downloads/en/devicedoc/30292c.pdf (Microchip™ 16F876A datasheet website.)
14. http://www.melabs.com/
15. http://www.estesrockets.com/Estes_Time_Thrust_Cu1100.html



<u>Quest™ E-Z Payloader™</u>

Overall Length: 21.5 inches

Rocket Weight Empty: 65.2 grams

Rocket Weight
with engine and payload: 135 grams

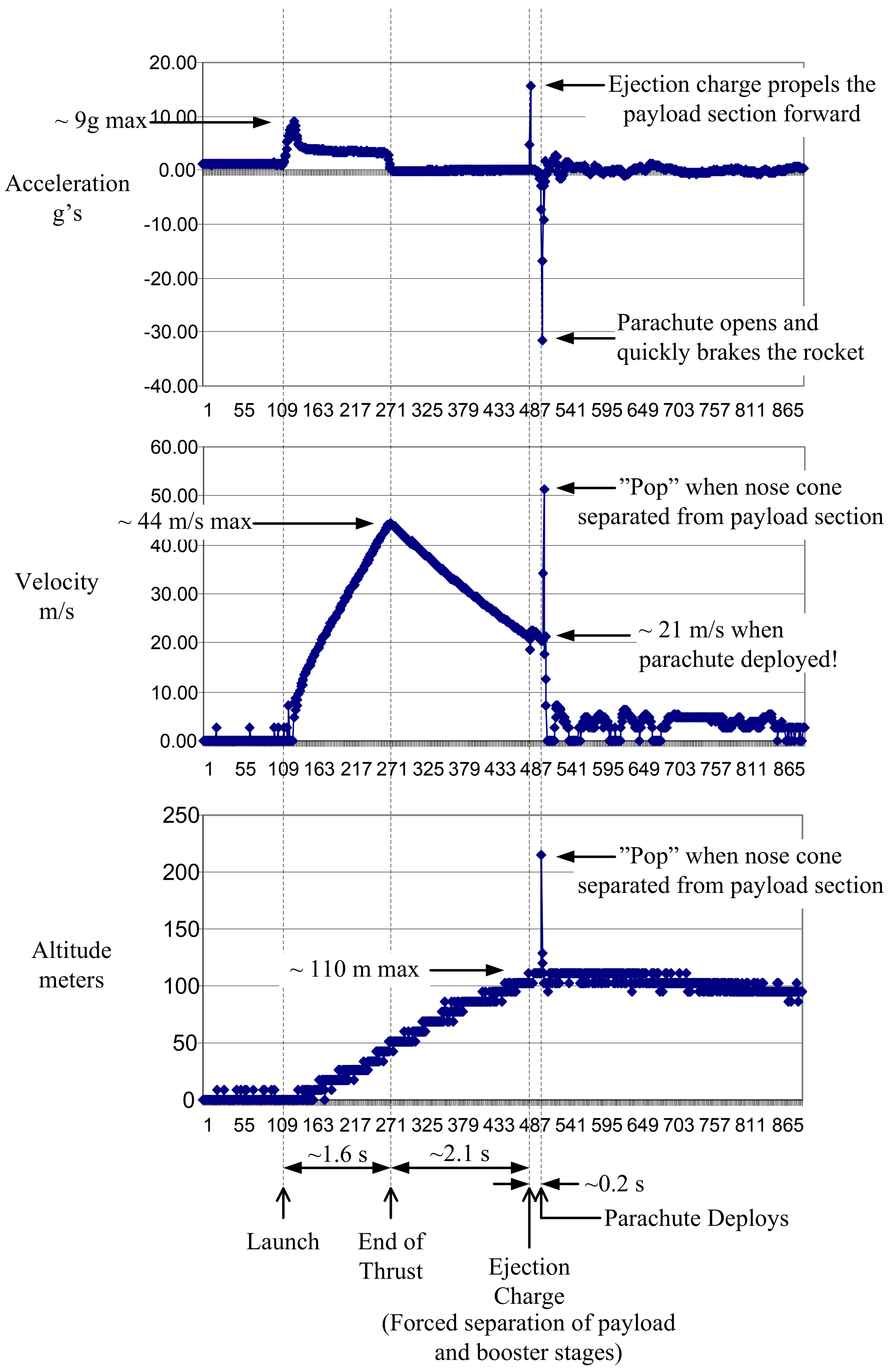
Appendix A. Quest™ E-Z Payloader™ Model Rocket Diagram

Appendix B Starts Here…

```
‘ PicBasic Pro program to take 10 10-bit A/D voltage readings of 4 sensors and save the readings to external I2C EEPROM
‘ while in flight on a model rocket.  The sensors are powered through the PORTC output pins tied in parallel.
‘
‘ This version does not use the ADCIN function.
‘ Connect analog inputs as follows:
‘ Accelerometer output to channel-0 (AN0, RA0)
‘ Differential pressure device output to channel-1 (AN1, RA1)
‘ Absolute pressure device output to channel-2 (AN2, RA2)
‘ Thermistor voltage divider is fed to channel-4 (AN4, RA5)
‘ AN3, RA3 is used as the positive side of the reference voltage for the accelerometer, pressure and thermistor sensors
‘

        INCLUDE “modedefs.bas”

DEFINE OSC 10           ‘Using 10MHz oscillator
DEFINE I2C_SCLOUT 1     ‘Set I2C clock output to bipolar instead of needing an external pullup.

Addata  VAR  WORD[4]    ‘Create array to store A2D data
Adsetup VAR  BYTE[4]    ‘Create array to hold A2D setup
Eeaddr   VAR WORD               ‘EEPROM address counter
Temp1     VAR WORD      ‘temporary variable
I  VAR  BYTE            ‘FOR loop index
j  VAR  WORD            ‘FOR loop index
SO      CON     2       ‘ Define serial out pin RB2
TRISA = %11111111       ‘ Set PORTA to all input
TRISB = %11110001       ‘ Set PORTB to all input except RB1 and RB2 (and RB3 for debug loop tests)
TRISC = %00000000
Eeaddr = 0
PORTB = 0               ‘Start out with all PORTB outputs low (Otherwise the serial port pin has no START)
I2Cread PORTC.4,PORTC.3,$A0,Eeaddr,[Temp1]          ‘Dummy read to get the CLK and Data lines in the right states
PORTC = PORTC & %00011000       ‘Set power off to sensors
ADCON1 = %11000001      ‘Right just, 16 Tosc, Vref on AN3

Adsetup[0] = %01000001  ‘Setup □ariable to set ADCON0: 16 Tosc, CH0, No start yet, Power up A/D
Adsetup[1] = %01001001  ‘Setup □ariable to set ADCON0: 16 Tosc, CH1, No start yet, Power up A/D
Adsetup[2] = %01010001  ‘Setup □ariable to set ADCON0: 16 Tosc, CH2, No start yet, Power up A/D
Adsetup[3] = %01100001  ‘Setup □ariable to set ADCON0: 16 Tosc, CH4, No start yet, Power up A/D

ADCON0 = %01000000      ‘16 Tosc, CH0, Power off A/D
OPTION_REG = %00000100          ‘Pull-ups on, Falling RB0, Assign Pre to TMR0, 32:1 prescalar
INTCON = %00010000      ‘GIE=0,PEIE=0,TMR0IE=0,INTE=1,RBIE=0,TMR0IF=0,INTF=0,RBIF=0

@ Sleep  ;Go to sleep waiting for falling edge of RB0

For j = 0 TO 9

        PORTB.1 = 1             ‘Turn on LED
        Pause 100
        PORTB.1 = 0             ‘Turn off LED
        PORTC = PORTC | %11100111       ‘Power up the sensors
        Pause 50
        TMR0 = 60       ‘Preset TMR0 so that it overflows after 2.5ms with 10MHz oscillator
        For I = 0 TO 3

                ADCON0 = Adsetup[i]
                Loop1: IF INTCON.2=0 Then Loop1    ‘Loop waiting for TMR0 to overflow
    TMR0 = 60
    ADCON0.2=1 ‘Start A/D conversion
    INTCON.2 = 0        ‘Clear TMR0 overflow bit
```

```
        Loop2: IF ADCON0.2 Then Loop2             ‘Wait for A/D to complete conversion
        Temp1.highbyte = ADRESH                   ‘get highbyte of A/D conversion
                      Temp1.lowbyte = ADRESL              ‘get lowbyte of A/D conversion
                      Addata[i]=Temp1

            Next i
            PORTC = PORTC & %00011000         ‘Set power off to sensors
            I2Cwrite PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4] ‘Write data to EEPROM
            Eeaddr = Eeaddr + 8                               ‘Increment EEPROM address pointer
            Pause 840

Next j

PORTB.1 = 1                               ‘Turn on LED
Pause 950
PORTC = PORTC | %11100111             ‘Power up the sensors
Adsetup[2] = %01000001       ‘Setup □ariable to set ADCON0: 16 Tosc, CH0 AGAIN!, No start yet, Power up A/D
Pause 50
TMR0 = 60                    ‘Preset TMR0 so that it overflows after 2.5ms with 10MHz oscillator
INTCON.2 = 0       ‘Clear TMR0 overflow bit
PORTB.1 = 0                               ‘Turn off LED

For j = 0 TO 899

            For I = 0 TO 3

                      ADCON0 = Adsetup[i]
        Loop3: IF INTCON.2=0 Then Loop3           ‘Loop waiting for TMR0 to overflow
        TMR0 = 60     ‘Preset TMR0 so that it overflows after 2.5ms with 10MHz oscillator
        ADCON0.2=1 ‘Start A/D conversion
        INTCON.2 = 0             ‘Clear TMR0 overflow bit
        Loop4: IF ADCON0.2 Then Loop4             ‘Wait for A/D to complete conversion
        Temp1.highbyte = ADRESH                   ‘get highbyte of A/D conversion
                      Temp1.lowbyte = ADRESL  ‘get lowbyte of A/D conversion
                      Addata[i]=Temp1             ‘Place all 16 bits of A/D conversion into array position

            Next i

            I2Cwrite PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4] ‘Write data to EEPROM
            Eeaddr = Eeaddr + 8                               ‘Increment EEPROM address pointer

Next j

PORTC = PORTC & %00011000         ‘Set power off to sensors
Adsetup[2] = %01010001       ‘Setup □ariable to set ADCON0: 16 Tosc, CH2, No start yet, Power up A/D

For j = 0 TO 59

            PORTB.1 = 1                   ‘Turn on LED to help see it when it is coming down
            Pause 50
            PORTB.1 = 0                   ‘Turn off LED
            PORTC = PORTC | %11100111         ‘Power up the sensors
            Pause 50
            TMR0 = 60            ‘Preset TMR0 so that it overflows after 2.5ms with 10MHz oscillator

            For I = 0 TO 3

                      ADCON0 = Adsetup[i]
                      Loop5: IF INTCON.2=0 Then Loop5   ‘Loop waiting for TMR0 to overflow
        TMR0 = 60
        ADCON0.2=1 ‘Start A/D conversion
        INTCON.2 = 0                'Clear TMR0 overflow bit

        Loop6: IF ADCON0.2 Then Loop6              'Wait for A/D to complete conversion
        Temp1.highbyte = ADRESH                    'get highbyte of A/D conversion
                    Temp1.lowbyte = ADRESL                 'get lowbyte of A/D conversion
                    Addata[i]=Temp1

            Next i

            PORTC = PORTC & %00011000       'Set power off to sensors
            I2Cwrite PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4] 'Write data to EEPROM
            Eeaddr = Eeaddr + 8                              'Increment EEPROM address pointer
            Pause 1890

    Next j

    Loop7: INTCON.1 = 0         'Clear the RB0 interrupt flag to get ready to sleep

    @ Sleep                     ;Sleep here waiting for falling edge on RB0.
    Pause 1000                  'just a short pause to let the user get ready for the data spew

    Eeaddr = 0

    For j = 0 TO 9

            I2Cread PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4]  'Read back the pre-launch data
            Eeaddr = Eeaddr + 8

            SerOut SO,N9600,[#Addata[0],32,#Addata[1],32,#Addata[2],32,#Addata[3],10]         'send it out RS232ish

    Next j

    SerOut SO,N9600,[10]        'Insert blank line between sections

    For j = 0 TO 899

            I2Cread PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4]  'Read back the high rate flight data
            Eeaddr = Eeaddr + 8

            SerOut SO,N9600,[#Addata[0],32,#Addata[1],32,#Addata[2],32,#Addata[3],10]         'send it out RS232ish

    Next j

    SerOut SO,N9600,[10]        'Insert blank line between sections

    For j = 0 TO 59

            I2Cread PORTC.4,PORTC.3,$A0,Eeaddr,[STR Addata\4]  'Read back the low rate flight data
            Eeaddr = Eeaddr + 8

            SerOut SO,N9600,[#Addata[0],32,#Addata[1],32,#Addata[2],32,#Addata[3],10]         'send it out RS232ish

    Next j

    GoTo Loop7
```

…Appendix B Ends Here



Appendix C. Flight Data: Launch-Ascent Phase