FAMU-FSU College of Engineering
Department of Electrical and Computer Engineering

Final Report

Project Title: SAE Baja Data Acquisition System
Team #: 5
Website: http://www.eng.fsu.edu/ece/senior_design/2015/team05/

Student Team Members:

Chris Riker, Computer Engineering (Email: cdr11c@my.fsu.edu)

Hebe Perez, Computer Engineering (Email: hp09d@my.fsu.edu)

Dewey Williams, Computer Engineering (Email: dmw10g@my.fsu.edu)

Tyler Dudley, Electrical Engineering (Email: dudleyt2004@gmail.com)

Project Coordinator: Dr. Michael Frank

Additional Reviewer: Dr. Linda DeBrunner

Submitted in partial fulfillment of the requirements for
EEL4914C – ECE Senior Design Project II

February 6, 2015

Table of Contents

Project Executive Summary

The Society of Automotive Engineers encourages college students worldwide to apply classroom knowledge to real world design concepts through their annual competitions. SAE International hosts five categories of competition that include Aero, Clean SnowMobile, Formula, Baja, and Supermileage. The SAE team at the FAMU-FSU College of Engineering actively participates in the Baja competition each year and subsequently builds upon their designs from previous competitions for vehicle improvement.

Previously, the SAE Baja Team has made changes to the vehicle based on estimates of the performance of the vehicle. This year, Team E#5 has introduced a quantitative approach to the Baja design process by adding a data acquisition system (DAQ). The DAQ records values from various vehicle systems and stores it for later use. DAQ subsystems include measurement for speed and acceleration, as well as uptime data and a fuel timer. In addition to recording and storing data, the DAQ system includes a wireless communications component which allows the vehicle to be monitored from an external station. This monitoring includes collected sensor data, as well as rudimentary communication between the pit crew and driver. This allows the pit crew to prepare for the driver and decrease the time it takes for the vehicle to get back on the track.

The system follows the competition design rules outlined by SAE International and was designed to hold up against the rough terrain of a Baja competition track. The system was complete in time for the April 9th, 2015 Baja competition in Auburn, Alabama and was completed with a budget of $750 with the aid of contributions from Texas Instruments.

Introduction

1.1 Acknowledgements

The SAE Baja DAQ design team would like to acknowledge our project advisors, Dr. Frank and Dr. Linda DeBrunner, for any advice, input and ideas offered on how to proceed during the design phase of the DAQ. The team would like to acknowledge the FAMU-FSU Society of Automotive Engineers for the sponsorship and advice they provided and for including this design team in this year's Baja car. The team would also like to thank Arrigo Dodge of South Florida for their contribution of four tire pressure monitoring system sensors to the Baja DAQ team. Finally, the team would also like to thank the FAMU-FSU Electrical and Computer Engineering department for their financial contributions towards the project.

1.2 Problem Statement

The acquisition of data can be a very important part of efficiently optimizing designs of many kinds, especially those involving mechanical systems such as automobiles, provided that the data collected is used in an appropriate manner. The design team aims to design and implement a Data Acquisition System (or DAQ) that will collect and store important data and ultimately make the FAMU-FSU Baja team's vehicle more successful in this year's competition.

In previous competitions, the SAE Baja team had little quantitative information to use when making changes to the vehicle. The team often wasted valuable race time by entering the pits when it was not necessary. The DAQ will inform the SAE Baja team of the proper changes and procedures needed to improve the vehicle mid-race, allowing them to focus less on problem identification and more on problem solving. The DAQ will inform the driver and pit crew of any issues with the vehicle, as well as provide rudimentary communication between the driver and the crew. Data collected by the system is available in the pit wirelessly and in real time.

1.3 Operating Environment

The Baja vehicle to which the DAQ system is mounted is driven on rough terrain in a harsh outdoor environment during competition. The system is subject to shock and vibration due to the rough terrain, rain, dust, dirt, mud, rocks, heat (both due to weather and ambient engine heat) and RF interference from other similar systems on opposing vehicles during competitions. The system needs to withstand temperatures up to at least 120 degrees Fahrenheit. In designing the system, the team has made considerations for the inclement conditions which the system faces during the SAE Baja competition.

1.4 Intended User(s) and Intended Use(s)

1.4.1 Intended User(s)

The intended users of the DAQ are the members of the FAMU-FSU SAE Baja team. As all members of the SAE Baja team must be at least 18 years of age (for liability reasons), any and all users of the DAQ must also be at least 18 years of age. Any user of the DAQ must also be adequately trained in the use of the system. One of the important attributes of the system, as outlined in the Needs Analysis and Requirements Specifications, was usability. This means that the system will be easy to use so that the Baja team can focus on the competition as opposed to the DAQ operation. Thus, just about any engineer on the Baja team should be able to use the DAQ. Only DAQ team members should exchange or modify the hardware components of the system itself.

1.4.2 Intended Use(s)

The DAQ is designed to be used by the Baja team members to evaluate various data points about the vehicle and make changes based on the data collected. To begin using the DAQ, the Baja team has to simply turn the system on and begin running it. Once powered on and instantiated, the DAQ begins collecting and displaying data to the users. The received data is written to a storage device so that the Baja team can analyze it after testing or competition. During the competition, the Baja team is able to use this data to make adjustments to the vehicle as needed. Ultimately, it is up to the Baja team to determine what is to be done with the collected and displayed data.

1.5 Assumptions and Limitations

1.5.1 Assumptions

- All components used in the design of the DAQ were chosen such that their specifications meet or exceed the requirements of the system.
    - The wireless data transmission module(s) will have sufficient range to reach the vehicle at one or more points on the track
    - The main power source and the subsystem power sources will allow at minimum 8 hours of system operation
    - The storage medium is large enough to accommodate all collected data
        - It is estimated that 1 GB will be more than enough to accomplish this
        - 40 bytes per line * 18000 seconds (5 hours) * 1 Hz = 720KB
- The main MCU has enough I/O pins to be able to interface with every subsystem of the DAQ.
- The main MCU is mounted inside of a protective enclosure to shield it from the competition conditions.
- The end product is to be used only on the Baja vehicle.

- The system is subject to inclement conditions outlined above in **Section 1.3 - Operating Environment.**

1.5.2 Limitations

- The DAQ will be limited in some ways by the Baja competition rules.
    - The engine of the vehicle cannot be modified in any way.
    - The DAQ must use its own power source separate from the Baja vehicle's power source.
    - The system cannot interfere with any essential vehicle systems.
- The system shall be designed within an initial budget of $750.
- The system shall be able to withstand temperatures up to 120 degrees Fahrenheit.
- The display shall be easily readable by the driver of the vehicle.

1.6 Expected End Product and Other Deliverables

Before April 9th, 2014, the team delivered a data acquisition system to the SAE Baja team. This system collected, stored, and displayed important data about the Baja vehicle to the team during the SAE Baja competition in Auburn, AL. This data includes acceleration and speed. This data was written to removable media so that the Baja team can import it to a computer and analyze it as they see fit. The system consisted of two sections, the collection system mounted other vehicle (outlined in section 1.6.1), and the pit station where a readout of the collected data was to be given to the pit crew (outlined in 1.6.2).

The Baja team also received a copy of each deliverable report upon completion, so that they may stay up to date on the development of the DAQ. The Baja team also received schematics of the system, top-level block diagrams, and data sheets and information about each component used. Additionally, they were given a copy of all source code used in development of the project, so that future Baja teams may learn and improve on the design further.

The end product is a standalone data acquisition system that collects data about the following subsystems:

- Acceleration
- Speed
- Time elapsed since a refuel

The data is logged onto a removable memory host that can stored for later use and be sent to the pit crew so that they may monitor the status of the vehicle remotely. The data will also be displayed on the vehicle's subsystem for driver awareness. In addition, the pit crew can be alerted by the driver ahead of time if a pit stop is necessary, so that the crew can prepare.

1.6.1 On-vehicle Data Collection

The actual data acquisition system itself is mounted to the vehicle. The unit collects data about the vehicle from several sensors, stores that data to a removable media device in a CSV file, and displays some of that data to the driver and the pit crew. The vehicle also has an on-screen indication that warns the driver that the pit is requesting a return. The unit consists of the following:

- 1 - Hercules LAUNCHXL-TMS57004
- 2 - MSP430G2553IN20 Microcontrollers
- 1 - TI DRV5053 Hall effect sensors
- 1 - LSM303 accelerometer
- 1 - Xbee-PRO 900HP wireless transceiver
- 2 - $LiFePO_4$ 24Ah batteries
- 1 - High-gain Dipole Antenna
- 1 - Adafruit 2.2" TFT display
    - Driven by an ILI9340 display chip
    - Onboard SD card slot

1.6.1 Pit Unit

The pit unit includes a display for the collected information as well as mirrored indicator lights, one for a driver-to-pit signal, and one for a pit-to-driver signal. This unit includes the following items:

- 1 - Raspberry Pi (B+)
- 1 - Xbee-PRO 900HP wireless transceiver
- 1 - Display

1.6.2 Deliverable Reports

The DAQ team has completed 3 milestones in the Fall semester, and 2 more in the Spring semester. The SAE Baja team is provided with copies of each deliverable report as they are completed. This is to ensure that the Baja team is satisfied with the progress of the DAQ and all of its components, and to ensure that the system fulfills the needs set forth by the team. The completed milestones are as follows:

- Needs Analysis and Requirements Specifications (Completed September 18th, 2014)
- Project Proposal and Statement of Work (Completed October 17th, 2014)
- System-Level Design Review (Completed November 14th, 2014)
- Detailed Design Review & Test Plan (Completed February 6th, 2015)

1.6.3 Documentation and Configuration Files

Along with the actual system itself, the Baja team will be provided with documentation about the implementation of the system. This includes a top-level block diagram, information about each component, and all source code for the project. Those items are outlined below.

1.6.3.1 Top-Level Block Diagram

The top-level block diagram of the system will provide the Baja team and any future DAQ teams with an idea of how the system is laid out at a glance. This diagram shows each different DAQ component and illustrates how they interface with each other. The diagram is stored as a PDF to allow for portability between platforms.

1.6.3.2 Component Information

The Baja team has been provided with information about each component used in the system. For each component used in the DAQ, the Baja team received:

- Datasheet
- Part number
- Information about how the part is interfaced with the overall system

This information will allow the current and future Baja teams to easily find and replace any failed components with the same components or components with similar specifications. In the event that future teams wish to improve on the DAQ design, they will be able to choose new components that will be compatible with the rest of the system.

1.6.3.3 Source Code

All of the source code used to implement the system has been given to the SAE Baja team. This will allow future members of the SAE Baja team to learn from the DAQ team's implementation of the system and improve on the design further.

2 System Design

2.1 Overview

A top-level block diagram for the overall design of the system is found above in **figure**. The design consists of two sections, the pit unit, which allows the pit crew to monitor the vehicle remotely and the on-vehicle DAQ system, which collects, stores, displays and transmits data from the sensor network. The vehicle MCU polls data from the sensors, formats the data, write the data to an SD card, and transmits data to the pit unit via an XBee-PRO 900HP. The system is powered by two 3.2V $LiFePO_4$ batteries connected in series. The on-vehicle display is a 2.2" SPI LCD display, and displays the speed of the vehicle as well as other collected information.

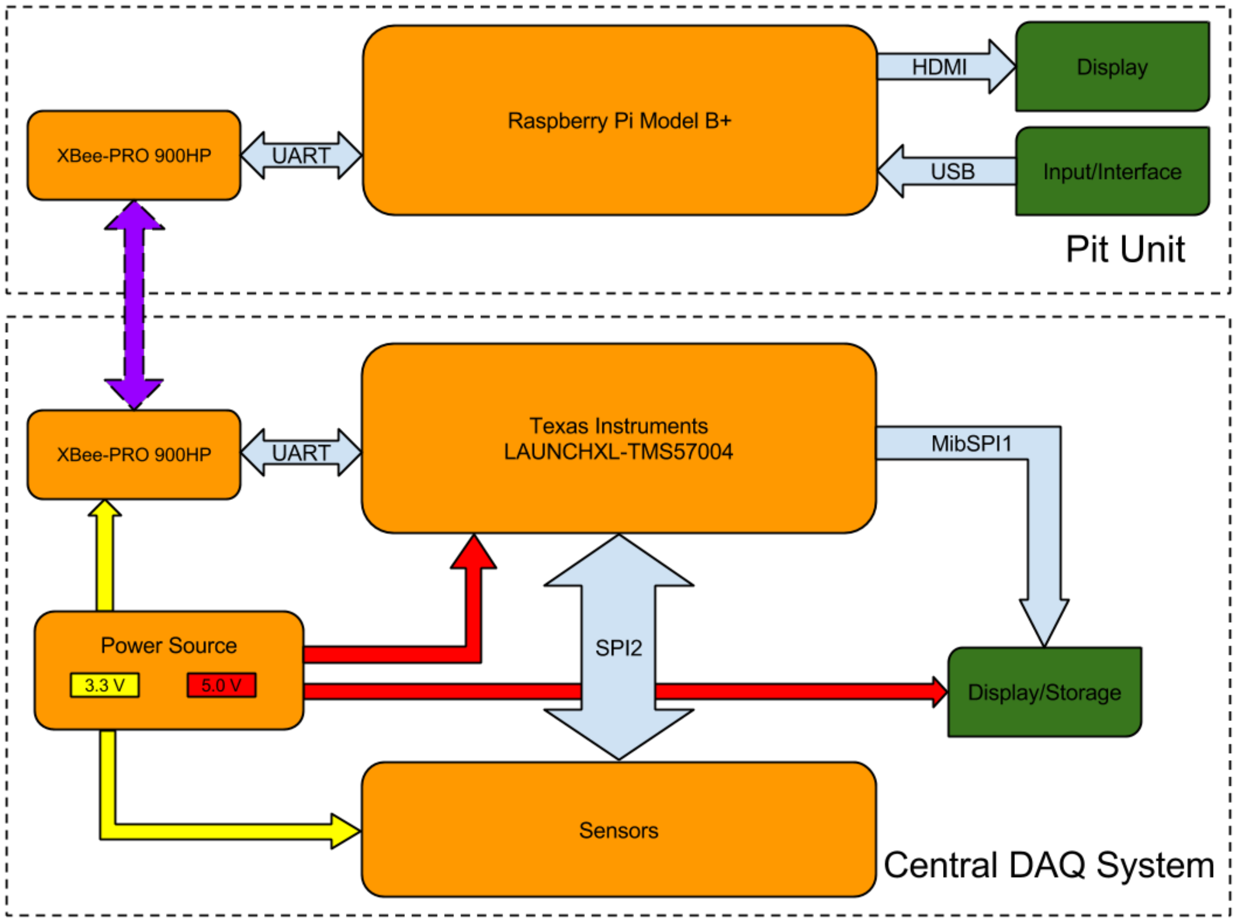


Figure 1: Top-Level Block Diagram of the entire system

2.2 Major Components and Component Requirements

2.2.1 On-Vehicle Data Collection Unit

The unit mounted on the vehicle contains the main microcontroller, a power source, a sensor network, a wireless transceiver and antenna, a microSD card slot, and a display as illustrated by the top-level block diagram in **figure 1**. The main microcontroller polls data from the sensor microcontrollers--which continually collects analog outputs from the sensors they are connected to, converting the analog inputs to digital representations, and converting the digital representations to the desired units (e.g. MPH, m/s^2, etc.). If polled, the sensor microcontrollers send their final calculated values to the main microcontroller via the SPI bus. Each sensor microcontroller draws power from the power regulation circuit.

2.2.1.1 Main Microcontroller

The main on-vehicle microcontroller is the central hub of information flow for the entire system. It collects data from each sensor via SPI and transmit or relay that data to a number of endpoints including a display for the driver, a log file on an SD card, and the remote pit unit.

The main MCU must be fast enough to perform data collection, display, and storage in near-real-time. It must have a number of interfacing options including hardware SPI, and it must have enough GPIO and function pins to provide support for all the current DAQ functions as well as any future additions the system might have. Additionally, the MCU must be reasonably priced and should have low power draw.

2.2.1.2 Accelerometer Module

The accelerometer, along with most of the other sensors used in the system, is connected to its own microcontroller. The accelerometer is read and written to over I^2C. The accelerometer is polled for its x- and y- axis readings by an MSP430 and the data is sent over SPI to the Hercules. The acceleration data can be used by the Baja team to increase performance in the acceleration portion of the competition.
The LSM303 accelerometer has a measurement range of +/- 2g which is plenty for (wheeled vehicles typically only reach accelerations of 0.4-0.6g, so a low measurement range reduced cost and increase accuracy).
The ST Micro LSM303 chip has an onboard 3-axis magnetometer which can be used to determine the heading of the vehicle. This feature is not yet incorporated into the DAQ system.

2.2.1.4 Hall Effect Sensor Module

Hall effect sensors increase their voltage output in the presence of a magnetic field. The Hall effect sensor is used to measure the speed of the vehicle by detecting the spike in the surrounding magnetic field caused by a magnet passing the sensor. The time it takes between two pulses is recorded by the MSP430, and a speed is calculated from that. Once calculated, the speed of the vehicle is sent to the main MCU and displayed on the on-vehicle display. The use of a discrete microcontroller to calculate speed is useful because the microcontroller must essentially poll the output of the sensor continuously.

2.2.1.6 Sensor Data Bus

The sensors and main MCU are connected by a single data bus which simplifies programming, allows for future expansion, and reduces pin use on the main MCU. The bus used is 4-wire SPI, using individual chip select lines to differentiate devices on the bus. The various sensor MCUs transmit data to the Hercules LaunchPad when their chip select lines are pulled low. Other peripheral devices may require input from the main MCU to be sent on the bus before data can be sent.

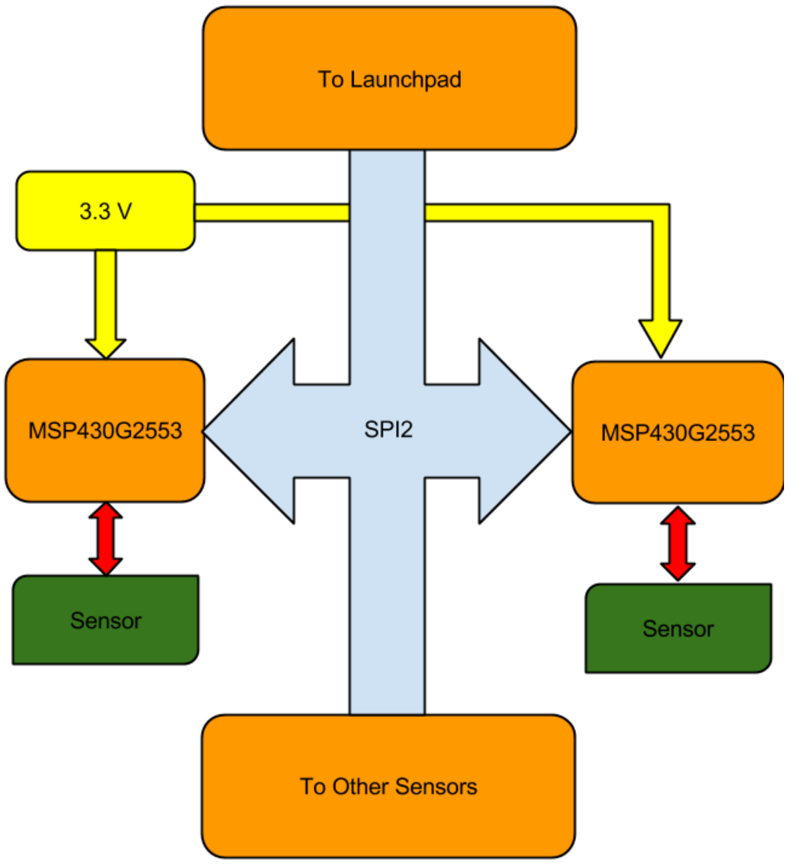


Figure 2: Block Diagram of the sensor network

2.2.1.7 Wireless Transceiver

The wireless transceiver allows the on-vehicle system to communicate with the pit unit remotely, providing the pit crew with valuable data while the vehicle is racing. This allows the crew to prepare for pit stops in advance and thereby reduce the overall time spent in the pit. There are two identical transceivers, one at each endpoint, which send data to one another.

2.2.1.8 Display

The display allows the driver to read important information about the vehicle such as the speed. Other messages such as system errors or warnings can also be displayed. This allows the driver to quickly determine when a pit stop is needed, and aids the driver in dynamic competitions as performance data can be displayed in real time.

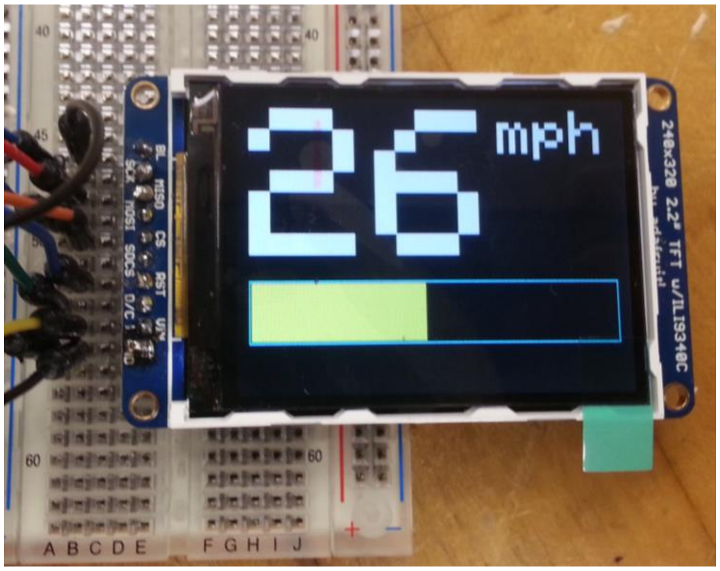


Figure 3: Initial Mock-up of the driver display

2.2.1.9 Storage

Removable storage is used to log data from the DAQ system for later analysis. The log is stored in CSV format to allow portability and ease of reading and writing. CSV is a very simple open format which is readable by a very wide range of free and proprietary software. An SD card is used as the removable media due to its ease of implementation and portability.

2.2.2 Pit Unit

The pit unit collects data from the DAQ system wirelessly and allows the pit crew to read the status of the vehicle remotely. It consists of a wireless transceiver (identical to the vehicle's transceiver) to retrieve data, and a processor to record or display that data to the crew.

2.3 Performance Assessment

Most of the requirements set forth in the Needs Analysis and Requirements Specifications were met upon completion of the project.. Some requirements were tabled indefinitely due to the solutions not being financially viable, while others were tabled due to, among other things, their lack of usefulness.

2.3.1 Functional Requirements

Most of the functional requirements were met upon completion of the project. The system was not able to accurately measure and display the speed of the vehicle (**REQF-001**). The system was able to accurately measure the linear acceleration of the vehicle (**REQF-003**). There are visual alerts (**REQF-007**) on the displays both on the vehicle and on the pit unit, alerting the driver that the pit requests a return, or alerting the pit that the driver is returning. The on-vehicle system has an easy-to-read display that shows the speed of the vehicle (**REQF-009**) as well as other pertinent information, such as a fuel timer and the status of the driver-to-pit and pit-to-driver notifications. The data collected by the system is written to an SD card in a *.csv file (**REQF-010**).

Some functional requirements were tabled due to their financial viability, their lack of usefulness, or engineering design concerns. The tire pressure measurement **(REQF-005)** was tabled indefinitely due to a lack of person-hours and monetary resources. The suspension travel measurement (**REQF-006**) was tabled indefinitely due to the cost of implementation using market solutions. It was determined that designing sensors was beyond the scope of this project given the time and material cost it would take to design and manufacture suspension travel sensors. The voice communication between the driver and the pit (**REQF-008**) was tabled due to the range concerns with the wireless transceivers. The design team met with the SAE Baja team and determined that the feature, which would have hampered the rest of the design, had limited usefulness. The Baja team determined that, if need be, a marketed voice communication system would be purchased and installed by them, independent of the DAQ team. The measurement and report of fuel level **(REQF-002**) has been modified after being informed of fuel tank modification constraints in the SAE international design rules. More information can be found in **Section 2.4.5.**

2.3.2 Non-Functional Requirements

The design team ultimately ran into some unforeseen expenses, and overran the given budget of $750 by $0.97. The system essentially remained within the budget provided by the FAMU-FSU Department of Electrical and Computer Engineering, as per **REQN-001**. The project was completed before the April 9th, 2014 competition in Auburn, Alabama (**REQN-002**). The on-vehicle system is powered by its own power source, explored in detail in **Sections 3.1.3** (**REQN-003**)**.** The system is powered on or off by a switch located outside of the system's protective housing (**REQN-004**). Since most of the sensors are housed independently of each other and the main MCU box, subsystems can be taken out without affecting the operation of the overall system (**REQN-005**). However, the fact that most of them will be mounted to custom PCBs could make repairing those subsystems a lengthy procedure if extra modules are not on hand and assembled. The code written was optimized as much as possible to ensure that the system was running as efficiently as possible (**REQN-006**).

2.3.3 Environmental and Health & Safety Requirements

The DAQ's power source is recyclable and lasts for a large number of recharge cycles (**EHS-001**). The system did/does not impede the function of any essential Baja systems such as brakes, steering, or electrical systems that are on the vehicle for safety reasons (headlights, brake lights, reverse lights) as per **EHS-002**. The system was securely attached to the vehicle to prevent creating a safety hazard for other drivers as per **EHS-003**, and remained so during the entirety of the Auburn, Alabama competition. Project boxes for the main MCU as well as the discrete sensors were chosen such that they will be protected from the elements as per **EHS-004**.

2.3.4 Usability Requirements

The user interface of the system is easy to read and use by anybody, as per **REQU-001**. The system's modular design allows the Baja team great freedom in placement of components around the vehicle, and allows them to remove and replace system components as needed to perform vehicle maintenance or other work. The system is powered on or off by a switch located on the exterior of the main DAQ housing so that the team can enable or disable the system as needed (**REQU-002**).

2.3.5 Reliability Requirements

The system is expected to run for many hours due to careful selection of components and battery technology (**REQR-001**) and allow for extra run-time for pre-race testing and adjustments, and also to compensate for extra power draw unforeseen by the design team. The modular design of the system simplifies component replacement in the event of component failure (**REQR-002**). The Baja team was consulted on system mounting locations to ensure that the system is safe and secure on the vehicle **(REQR-003)**.

2.3.6 Final Requirements Analysis

Table below shows a tabulated list of the requirements of the system as laid out on hte Needs Analysis and Requirements Specifications, indicating whether or not each requirement was met.

2.4 Design Process

2.4.1 On-Vehicle Data collection System

2.4.1.1 Central MCU

The central MCU was chosen to be both fast and inexpensive, with enough interfacing and expansion capabilities to facilitate future improvements and unforeseen design changes.

There were four development boards considered for use as the central controller. Of those four, two had features that better suited the needs of the main controller. These two were the Intel Galileo and the Hercules LAUNCHXL-TMS57004 from Texas Instruments. The Intel Galileo was rejected due to the high cost ($65.25) and the Hercules was chosen because of its automotive and safety focus.

Table 1: Criteria/Justification for Central MCU selection

Criteria	Justification
Flash memory size	The MCU flash memory size must be large enough to accommodate software written for the DAQ
GPIO pin count	A larger amount of GPIO pins will ensure that there will be enough pins for the sensors and future project expansion.

SPI support	After deciding to use SPI for data communication, choosing an MCU with built in SPI support would shorten and simplify development.
Clock speed	The MCU should be able to collect data in real time. Higher clock speed means a higher sampling rate can be achieved.
Price	The limited project budget was considered
Power Consumption	The system will be running off a limited battery supply. Power consumption from the MCU must be kept to a minimum.

The TI Hercules TMS570LS0432 is an 80MHz microcontroller featuring dual ARM Cortex R4 cores in lockstep. The Hercules' interfaces include a UART (or SCI), and 3 SPI interfaces, as well as analog inputs and both dedicated and multiplexed GPIO, offering great expandability and integration.

2.4.1.2 Sensors

2.4.1.2.1 Sensor Data Bus

The data transmission bus is used to move data around the vehicle, and primarily allows the main MCU to retrieve and collect data from each sensor. The same bus type is used to display and record the sensor data.

Table 2: Criteria/Justification for sensor data bus selection

Criteria	**Justification**
Simple Frame Structure	A simple frame structure will simplify code and reduce unnecessary data being transmitted on the bus.
Low Pin Requirement	Conserving pins on the main MCU will allow future developers to add more to the system at a later time.
Onboard Hardware	Hardware or hybrid implementations should be available on-chip and at a low cost. Because there are many devices on the bus, adding hardware can become very costly.

A number of different serial communications standards were considered, including I^2C and CANBus, but the team chose SPI due to its simplicity in slave selection and its full duplex operation. While I^2C was a main contender, its use of an addressed frame was determined to be unnecessary and costly in both code and in data transmissions (SPI's higher pin requirement was decidedly less costly than the use of an addressed frame). Additionally, CANBus requires external hardware and is very feature-heavy and could not be fully or efficiently utilized by the current system. 4-wire SPI is also supported natively in SD cards, which greatly simplifies data storage.

2.4.1.2.2 Sensor Microcontrollers

In deciding how to deal with the sensor outputs, the design team decided on using a data bus to reduce the number of pins used on the main microcontroller. The design team decided to connect each sensor to its own discrete microcontroller, which will format collected samples and properly transmit them on the SPI bus. In choosing this design paradigm, some of the calculations that would have been done by the main microcontroller can be offloaded onto the sensor microcontrollers.

When deciding on which microcontroller to use for this purpose, the design team was looking for a controller that was low-cost, through-hole mounted, had a minimal number of pins, and had enough flash memory to achieve the task for which it will be used.

Table 3: Criteria/Justification for sensor MCU selection

Criteria	**Justification**
Low-cost	Since there will be several of these microcontrollers, their cost should be kept as low as possible.
Through-hole Mounting	Through-hole mounting was preferred to make the milling of the PCBs in-house as easy as possible, reducing the risk of having to outsource custom PCB milling (see **6.4.2**)
Minimal Pin Count	Minimal pin count was desired again to make the milling of the PCBs in-house as easy as possible, as well as to avoid wasted features.
Flash Memory	The team wanted to be sure that the microcontroller selected had enough flash memory to hold the code that it would be programmed with to avoid wasting money.

The DAQ system utilizes Texas Instruments MSP430G2553 microcontrollers for all sensors. It has dual onboard USCIs (one SPI/UART and one SPI/I^2C), and plenty of GPIO and analog inputs.

2.4.1.2.3 Speedometer

The speedometer module measures the speed of the vehicle using a Hall Effect sensor which can detect the presence of a magnetic field. The Hall Effect sensor is placed at a stationary location on the vehicle, and a permanent magnet is affixed to a nearby rotating shaft. By measuring the time between pulses from the Hall Effect sensor, rotational velocity can be calculated, and from there the linear speed of the vehicle can be found.

When considering different Hall Effect sensors, the design team considered different sensors based on the criteria outlined in **Table 4** below.

Table 4: Criteria/Justification for Speedometer implementation selection

Criteria	**Justification**
Through-hole or Frame Mounting	Like the other sensors and sensor MCUs considered, through-hole mounting was desired in order to simplify the milling of custom PCBs due to the risk outlined in **6.4.2**
Analog Output	Some components that are called "Hall effect sensors" are actually switches, which are switched "on" with the presence of a magnetic field, and switched "off" again when the presence of a magnetic field is detected again. The design team desired a Hall effect sensor that output an analog voltage whenever a magnetic field was detected.
Cost	Hall effect sensors can be very expensive depending on how robust (durable) they are as well as the type of output they produce. Industrial Hall effect sensors can cost up to $40/unit.

The team decided on a Texas Instruments DRV5053 Analog-Bipolar Hall effect sensor, which was provided by TI as part of the Innovation Challenge.

2.4.1.2.4 Accelerometer

The accelerometer module is used to measure (most importantly) the linear acceleration of the vehicle, allowing the Baja team to tune the vehicle to their liking for the specific terrain/event. When looking for an accelerometer, the design team considered the characteristics outlined in **Table 5.**

Table 5: Criteria/Justification for the selection of the accelerometer

Criteria	**Justification**
Mounting Style	Through-hole mounting was preferred. As with the other PCB-mounted components, using surface mount components was not viable for the in-

	house PCB milling. See **6.4.2** for associated risks.
Bandwidth	The bandwidth determines how often an accurate acceleration measurement can be taken. The bandwidth should be high enough such that the acceleration can be accurately represented without being
Sensitivity	The sensitivity of an accelerometer determines range of accurate acceleration measurement. Given that the vehicle will likely not reach anywhere near 1g, accelerometers with excessively high sensitivities were ignored, as they are generally more expensive with less sensitive accelerometers.
Cost	Accelerometers can be very expensive and have a wide range of price points. Given the fairly slim budget of the project, the chosen accelerometer was fairly low-cost.

The team decided on an ST Micro LSM303 accelerometer from Adafruit. The ST Micro LSM303 is a digital accelerometer and magnetometer package which offers 3-axis acceleration measurement +/- 2g (selectable) with 16 bit resolution. It also offers an onboard 3-axis magnetometer also with 16 bit resolution.

2.4.1.2.5 Fuel Level Measurement

Upon reading through the updated rulebook for the SAE Baja series competition, the design team discovered that no holes (even patched holes) are allowed to be in the provided fuel tank. For this reason, the design team decided to forego implementing this subsystem. Many other SAE Baja DAQ teams have faced the same issue, and most have turned to simply implementing a timer instead of actual fuel level measurement, as most solutions seem to fail technical inspection, and thus have to be removed from the vehicles. For this system, a timer was implemented instead that counts down from a specified time, and can be reset either from the pit unit, or using the driver's notification button

2.4.1.3 Display

The vehicle display was required to be compact, bright, and simple to integrate. The team chose a 2.2" TFT display from Adafruit with an integrated ILI9340 display driver which communicates over SPI. This display was chosen based on the Hercules' interface options, and the availability of sample code from Adafruit which shortened the required development time.

Design of the driver interface was done in GIMP, an open source image editor. The interface was designed with the graphics limitations in mind, such as the difficulty of drawing complex shapes.

Any non-rectangular graphics required were composed in GIMP and incorporated into the interface bitmap, and dynamic elements such as sensor data or notifications were drawn over this bitmap. Graphics are stored in a raw "565" image format on the SD card, and drawn on the display at boot.

2.4.1.4 Storage

The two storage options considered for this block were USB Mass Storage and Secure Digital (SD). The benefits and drawbacks of both had to be weighed, as they both are well suited to complete this task. Below is a summary of the features of both storage options as they apply to the DAQ storage needs.

Table 6: Criteria used for selection of the storage medium

Criteria	**USB Mass Storage**	**SD**
Compatibility	Can be used on any modern machine	Many computers do not have built in SD card readers
Portability	Flash drives are very portable. Longer flash drives can break under stress	Small and portable. Can be easily lost or broken.
Implementation	Requires a host controller	Built in SPI capability

As seen above, one of the downsides to using USB Mass Storage is that a host controller is required to communicate between the MCU and the USB device. This could be costly in finances as well as extra development time and complexity. SD supports SPI natively which is also found in hardware on the MCU, greatly easing the hardware development and design.

It is also worth noting the LCD display chosen comes with a built in SD card slot which has saved the team some time in development by reducing hardware components, and allowing both the SD card and LCD display to be accessed on one SPI bus using an additional chip select line on the display board.

2.4.1.5 Power System

When considering which battery technology to use, the team considered several factors to make the decision. Since the batteries are the most expensive components of the entire system, it was vital that the team considered all available options and made the right choice first time. The criteria used to evaluate battery technologies are outlined below in **Table 7.**

Table 7: Criteria/Justification for power system selection

Criteria	Justification
Safety	Some battery technologies are infamous for having catastrophic failures. For example, Lithium ion and Lithium polymer batteries are subject to explosion or fire if they are overcharged, overdrawn, or connected improperly. Lithium polymer batteries come in pouches, and as such are prone to crushing and puncture, which can lead to a catastrophic failure. Safety is of the utmost importance in the Baja competition. As such, safety factored heavily into the decision.
Price/Capacity	Batteries are very expensive. The batteries will be the most expensive components in the entire system. As such, the $USD/capacity also factored heavily into the decision.
Nominal cell voltage	Nominal cell voltage was a fairly important part of the decision. Some batteries had nominal cell voltages that were far too low or far too high to be viable for the voltage requirements of the components of the system.

The team ultimately decided to use Lithium Iron Phosphate batteries. They come in small form factors, provide a large capacity for a good price, and eliminate all of the safety concerns associated with Lithium ion and Lithium polymer batteries.

In order to regulate the voltage of the batteries, switching voltage regulators are used. The DAQ system uses Texas Instruments TL2575 voltage regulators in 3.3 V and 5 V variants, provided by TI.

2.4.1.6 Enclosures

The enclosures used in the DAQ system must be both waterproof and resistant to shock or other physical trauma. For the smaller sensor enclosures, a 4”x3”x2” enclosure from Estone was selected, which offers a waterproof seal and a mounting flange for attaching to the vehicle, as well as internal screw posts for attaching components inside.

2.4.2 Pit Crew Unit

The Pit Unit displays data collected from the system on a screen so that the pit crew can see the status of the vehicle at a glance.

2.4.2.1 SBC

The central component of the pit unit is the single board computer (or SBC). The use of a SBC simplifies the display of collected data. The decision to use the Raspberry Pi B+ as the pit crew

SBC was made when comparing development boards for the main MCU. The features were impressive but with its optimization for graphic applications, the Raspberry Pi was not a match for what was needed as the DAQ system's main controller. Since the pit crew module relies heavily on the display of data to the sideline team members, it was determined that the Raspberry Pi was better suited for this module.

2.4.2.2 Display

The display used is a 7" TFT LCD display which operates on 12V power and has a composite video input which can be connected directly to the Raspberry Pi's composite out. The display can handle the Pi's 720x480 composite output with reasonable clarity, and is small and lightweight. It also includes a number of mounting options and a remote control.

2.4.3 Wireless Communication

The team chose to use the XBee-PRO 900HP due to its long range and low transmit power draw (as compared with its output power). This device operates in the 900 MHz band and offers both serial communication via UART and I/O pin mirroring capabilities. Models are available that offer mesh networking, or simple Point-Multipoint operation. The DAQ system uses the point-multipoint model and communicating with the XBee via UART on both ends.

2.4.4 Software Development

All of the code for the on-vehicle system was written in the C programming language. Some libraries were developed/ported from other systems for this project, including:

- Software I^2C library, for the MSP430 microcontroller
- Display library for the Ilitek ILI9340, for use on the Hercules LaunchPad
- SD card interface library based on FatFS, for use on the Hercules LaunchPad

The primary development environment is Texas Instruments' Code Composer Studio on Windows, with Raspberry Pi development being done using Qt Creator. Any code developed by the DAQ team is offered as open source code without a license, and is available on the team's GitHub which is accessible through the team website. The link is provided below.

https://github.com/hp09d/SAE-Baja-Data-Acquisition

2.4.5 PCB Design

Three PCBs were designed for use in the DAQ system: a board for sensor microcontrollers, a board for power regulation, and a board to break out the necessary connections on the Hercules.

Boards were designed in CadSoft EAGLE, and milled in-house at the FAMU-FSU College of Engineering.

2.5 Overall Risk Assessment

2.5.1 Financial Risks

2.5.1.1 *Financial Risk 1: Component Destruction*

Description: Electrical components are very sensitive to the presence of static electricity, and can be permanently damaged by a static discharge. Being mounted on an off-road vehicle, some components could have potentially been destroyed by a vehicle roll-over or inclement weather conditions. Components could have also been destroyed if they were connected incorrectly, e.g. if a component is connected to the wrong voltage levels or draws more current than it is rated for. The Destruction of any components of the DAQ could potentially consume a large portion of the budget (depending on which component is destroyed) and/or prevent the team from delivering the fully functional system on time.

Probability: High
While destruction by static shock was the least likely situation to occur, it was still a potential problem. Being mounted on an off-road vehicle that will likely be rolled over frequently, components could have potentially been crushed depending on where they are mounted on the vehicle. This is the most likely scenario for component destruction.

Consequences: Tolerable to Serious
The consequences of component failure vary depending on which component is destroyed. Destruction of an XBee, a Raspberry Pi, or one or both of the batteries would have been catastrophic, as they are the most expensive components in the system. The XBee and the batteries are mounted on the vehicle. As such, they are the most expensive, most vulnerable components in the entire system. On the contrary, destruction of a small sensor or some other cheap component would have tolerable consequences, as their financial impact is minimal.

Strategy:
When choosing housings for the components found on the vehicle, the structural integrity of the enclosures was carefully considered, as was their mounting points. Components were optimally mounted in places that were less likely to bear the full weight of the vehicle upon a roll over. Since components could also be destroyed by static shock, anti-static wrist bands were worn whenever expensive electrical components are handled. Additionally, special care was taken when connecting components to power sources to ensure that they are not overloaded with voltage or current.

2.5.1.2 *Financial Risk 2: Component Failure*

Description: At any time, electrical components can fail for a variety of reasons related to manufacturing defects or heavy use. As with component destruction, component failure could potentially consume a large portion of the budget and/or prevent the team from delivering the fully functional system on time.

Probability: Low
Electrical components are manufactured to be very reliable and seldom fail spontaneously under normal use. However, considering the project's intended operating environment, it may have been possible for a component's longevity to be affected by heat or vibration.

Consequences: Tolerable to Serious
For the same reasons as component destruction, the consequences of component failure vary from tolerable to serious depending on which of the parts fails. Failure of expensive parts could have heavily impacted the project budget, while failure of inexpensive parts would have has minimal impact. Any failed part would have affect the project's progress and may negatively affect its completion time.

Strategy: Since component failure would have been caused by manufacturing defects, mitigation of this risk was near-impossible. The design team has no control over the manufacturing of the components used. The only mitigation strategy relating to overuse would be to limit excessive component use, although the amount of use required to cause a spontaneous is very high. As such, failure due to overuse is extremely unlikely.

2.5.1.3 *Financial Risk 3: Project Cost Overrun*

Description: There was a potential that the actual cost of the project may be above that of the preliminary estimations. As with any project, unforeseen issues and changes can occur during project development which could cause the project to cost more than anticipated.

Probability: Low
The preliminary budget estimate came very close to the originally allotted \$600, and did not take into account, among other things, pin headers, voltage regulators, component housing, mounting hardware, programmers or antennae. The final budget was increased to \$750, but overrun could still occur due to unforeseen expenses or component replacement.

Consequences: Serious
Budget overrun would have required the team to explore other funding options to complete the project. If extra funding was not found, or the funding is insufficient, then features may have to

be cut, or project specifications may not be fully met. If an overrun is anticipated, the team will begin seeking external funds as soon as possible to avoid a delay in project progress.

Strategy: The preliminary budget analysis was done using a worst case scenario, providing some monetary cushion in case unexpected expenses arise. The team took steps to attempt to procure some components from Texas Instruments for free in order to eliminate some of the costs of the project.

2.5.3 Safety Risks

2.5.3.1 Safety Risk 1: *System Mounting Failure*

Description: It was possible that the system could become detached from the vehicle via a mounting failure. This poses a safety risk to not only the driver of the FAMU-FSU Baja car, but every driver in the competition as well as spectators of the competition.

Probability: Low
As there are no components that are particularly heavy or bulky, mounting was expected to be simple and secure. Even so, unexpected complications can occur, and given the operating environment of the vehicle and the location of some subsystems mounting could prove to be more difficult or precarious than anticipated.

Consequences: Catastrophic
The safety of the Baja team's driver as well as the other competing drivers is of the utmost importance in the SAE Baja Series. A system mounting failure could pose potential safety risks to all of the drivers in the competition. As such, a system mounting failure was to be avoided at all costs.

Strategy: In order to mitigate this risk, the team worked closely with the SAE Baja team when working on mounting hardware and mounting points. When necessary, a member of the Baja team provided welding services in order to ensure that the system was safely and securely mounted to the vehicle at all times. Additionally, all screws used will be secured with thread locking adhesives.

2.5.3.2 Safety Risk 2: Water Damage

Description: During competition or in testing, the Baja car may be used in inclement weather or in situations that require navigation of a water hazard. Water poses certain risks to electronics and electrical systems and steps must be taken to ensure the safety of the system and anyone working on it in such conditions.

Probability: Low
The components of the DAQ system are housed in water resistant housings to prevent water or other materials from interfering with the DAQ system's performance. Components were placed with care to maximize the safety of the components and prevent unnecessary exposure to physical risks.

Consequences: Catastrophic
Water damage to any subsystem could have caused it to become inoperable or in the worst case, caused electrical damage to other parts of the system as a whole. Water present in the system could pose safety risks to the vehicle operator or any crew working on the vehicle including electrical shock, fire, or chemical leaks from the batteries. Additionally, this damage may be irreversible and could pose a financial risk as well.

Strategy: In order to mitigate the risk of water damage, great care will be taken to properly house and protect all components. Fuses will be placed in key locations of the power system to prevent any electrical damage or overall system failure in the event of a short circuit. Additionally, great care must be taken by the Baja and DAQ teams when disassembling or modifying any electrical systems on the vehicle.

2.5.4: Design Risks

2.5.4.1 Design Risk 1: Tire Pressure Monitoring System
Description: The team found during their research of potential solutions for the Baja TPMS that there exists virtually no marketed solutions for ATV tires. While the project team was donated 4 automobile TPMS sensors, there is a chance that the team will be unable to interface with the sensors, as they require a certain command to be sent in order for the sensors to return their 69-bit long data stream. If the team is unable to get the sensors provided to return usable data, the TPMS may have to be scrapped, as designing custom TPMS sensors would require more time and money that the design team does not have.

Probability: High
The sensors that were donated to the team, as well as basically any other commercially available automobile TPMS sensors, were designed using a proprietary activation code. As such, they will require a significant amount of reverse engineering and guess work in order to interface with them properly. If the team is not able to successfully interface with the sensors, the tire pressure measurement may be shelved entirely.

Consequences: Tolerable
The tire pressure monitoring is one of the less important subsystems of the DAQ. The Baja team rarely has problems with the tires that are not immediately evident by the driver based on the feel

of the car, or the pit crew based on visual inspection - even from a distance. The tire pressure monitoring system on the Baja vehicle was almost scrapped completely upon learning that the sensors available on the market may not be sensitive enough to measure below 10 PSI. With that in mind, failure to produce this part of the system would not be detrimental to the project.

Strategy: The team was in contact with representatives from a company that manufactures replacement TPMS sensors, and has gathered some information regarding data types and frame information.

2.5.4.2 Design Risk 2: Wireless Vehicle-to-Pit Communication

Description: The team was hard-pressed to find information about the urban data transmission range using the 2.1 dB dipole antenna. The XBee-PRO 900HP is capable of a 4-mile line-of-sight with a 2.1 dB antenna at 200 kbps, but no information was given about the urban range. If the transmission range did not meet the requirements, the pit may have only received a transmission once per lap, or whenever the vehicle is in range. Depending on the topology of the endurance course, this could last anywhere from seconds to minutes.

Probability: Moderate
When considering wireless transceivers, the design team came across a benchmark for a wireless transceiver module—the XTend—that listed a 28-mile line-of-sight range using a 2.1 dB dipole antenna, which gave optimism to the team when considering the needs and requirements of the project. During component selection, however, the design team found that the specific model that was rated at 28 miles LOS was very expensive, well beyond the means of the project given the provided budget. Given that, the design team had to choose a less-powerful XBee module, which is only rated at 4 miles LOS, leading to concerns about the unit's ability to reach the required 3-mile LOS range that was given in **CONS-005**, outlined in the Needs Analysis and Requirements Specifications.

Consequences: Tolerable
Since the 28-mile range modules are very expensive--$180 per unit--they were out of the range of attainability for the design team. The modules would cost almost half of the given $750 budget. Even if the team were to obtain outside sponsorships, receiving the amount that is needed to use the XTend modules was unlikely. As such, the team had to simply work with the hand they were dealt. If the XBee-PRO 900HP does not reach the proper range, the pit crew may only receive data transmissions once per lap, or only when the unit is in range.

Strategy: If the XBee-PRO 900HP did not give the design team the required range, the pit unit may have only been able to receive data transmissions once per lap, or whenever the vehicle was in range. Due to the unknown topology of the endurance courses, whether the transmissions occur once per lap or during the whole race remained unknown until the time of the race. Unless the design team can receive funding for ~$400, the team simply had to live with what the XBee-PRO 900HP gave in terms of range and transmission accuracy.

2.5.4.3 Design Risk 3: In-house Sensor MCU Printed Circuit Boards

Description: The team decided to buy discrete MCUs for each sensor and fabricate custom PCBs for each one. If the PCB milling was not possible or failed in-house, the team may have had to use perfboard to make their own circuits, which would be fairly time consuming, or seek outsourced manufacturing, which could have potentially add a significant amount of cost to the project.

Probability: Moderate
There was a chance that the PCB milling facility at the FAMU-FSU College of Engineering was unable to fulfill the requirements of the design team. The traces may have been located too close together in the CAD design for the milling machine to mill the PCB without issues.

Consequences: Tolerable
Failure to have the PCBs created in-house could have had a large financial impact on the project. This financial impact could have been mitigated by changing the design paradigm of the system, which could ultimately affect the timely completion of the project. If the in-house PCB milling was not possible, the design team would have been tasked with deciding whether to pay a potentially large sum of money to have PCBs manufactured elsewhere or to build the circuits using perfboard.

Strategy: The actual milling of the printed circuit boards was out of the control of the design team. The team received the mill tolerances from Donte Ford and adhered to these specifications in order to maximize the possibility that the in-house PCB milling would be successful. If the in-house milling was not successful, the design team would have first looked into outsourced manufacturing before deciding to build the circuits by hand on perfboard.

2.6 Final Risk Assessment

Any risks that the team had previously foreseen that materialized will be explored further in this section. Any risks outlined in **section 2.5** that do not appear in this section can be safely assumed to have not materialized during the design, development, and use of the system.

2.6.1 Financial Risks

2.6.1.1 Component Destruction

Some components of the system ended up being destroyed in one way or another. With that in mind, it should be noted that the mounting of all of the components onto the vehicle was a success, as none of the components became detached during the competition or were destroyed due to the vehicle rolling over.

2.6.1.1.1 XBee PRO-900HP
At some point during the installation of the system, the XBee module for the pit unit was mistakenly connected to an input voltage that it was not rated to handle. Originally, component destruction was considered a financial risk. However, this risk did not materialize in a financial fashion. The team discovered that the XBee module was no longer operational the morning of the Auburn, Alabama Baja competition. As such, the destruction of this component affected the operation of the system, rather than the budget of the project. In reaction to the loss of this vital component of the system, the Hercules LaunchPad on-board the vehicle was reprogrammed so allow the driver to reset the fuel timer and start/stop the SD card data logging from the button previously used as the driver-to-pit signal. Two-way communication between the vehicle and the pit was a vital portion of this project, so the loss of this component was devastating to the usefulness of the system during competition.

2.6.1.2 On-Vehicle Display

As with the XBee, the destruction of the on-vehicle display did not materialize as a financial risk, as the destruction of this component was also discovered on the morning of the Auburn, Alabama competition. At some point during the transportation of the vehicle from Tallahassee, Florida to the competition site in Opelika, Alabama, the on-vehicle display became cracked through the center, making a large of the portion of the screen unusable. It became clear quickly that the screen would become completely broken soon into the competition, as the conditions of the course were much worse worse than what the system was subject to during transportation. As with the XBee, the destruction of this component was also detrimental to the overall usefulness of the system during competition. Destruction of the screen due to shock was not something that the team had considered during the design process. See **section** on future considerations for the project for more information about this part of the system.

2.6.2 Safety Risks
Thankfully, none of the safety risks that the team had considered initially materialized. The system stayed securely mounted to the vehicle throughout the entire duration of the competition, and nobody was injured during the design/installation of the system (save from some minor soldering iron burns).

2.6.3 Design Risks
2.6.3.1 Tire Pressure Monitoring System
The team did not have the resources/time required this year to be able to implement a tire pressure monitoring system. Given the difficulty of extracting information about the framework of the transmissions made by the sensor as well as a shortage of resources (both monetary and man hours), the team decided that this feature was not important enough to sacrifice the quality

of the rest of the system to attempt to implement it. See **section** on future considerations for the project for more information about this part of the system.

3 Design of Major Components/Subsystems

3.1 Central Data Acquisition Unit

Figure 4 below shows a top level pin diagram on the on-vehicle DAQ system. It shows how components are connected on a pin-by-pin basis, and shows which components are connected to which interfaces on the main MCU.

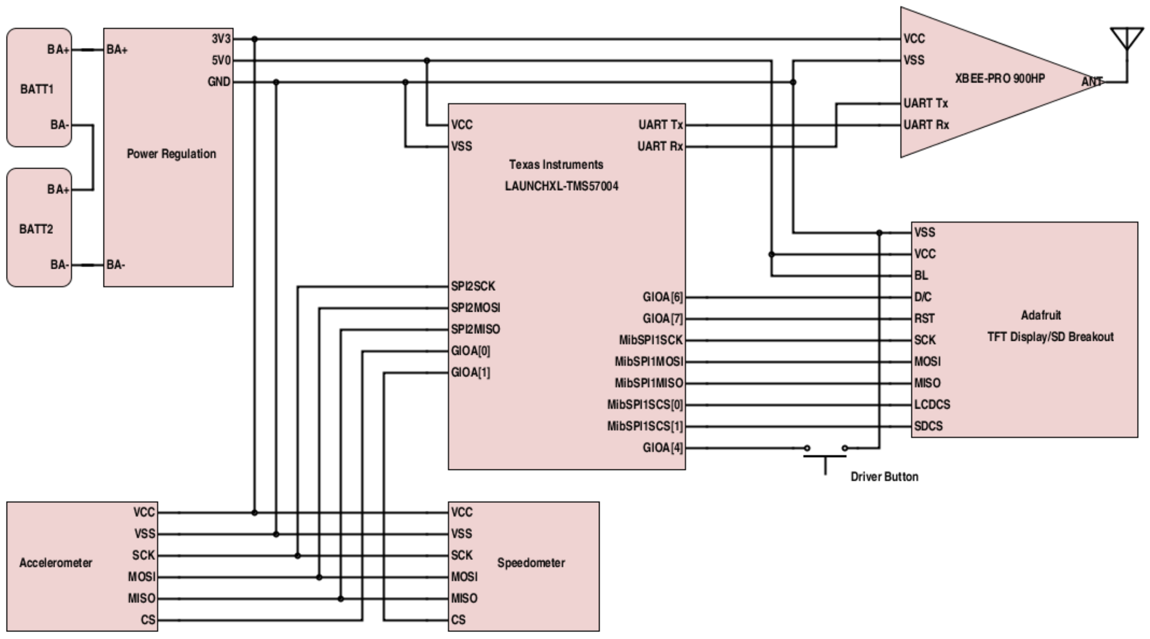


Figure 4: Top-level pin diagram of the on-vehicle system

3.1.1 Main Microcontroller

The microcontroller serves as the central control unit of the data acquisition system, and executes the control flow shown in **Figure 5.** Information from the sensor bus is sent into the MCU to be organized and stored into a log file.

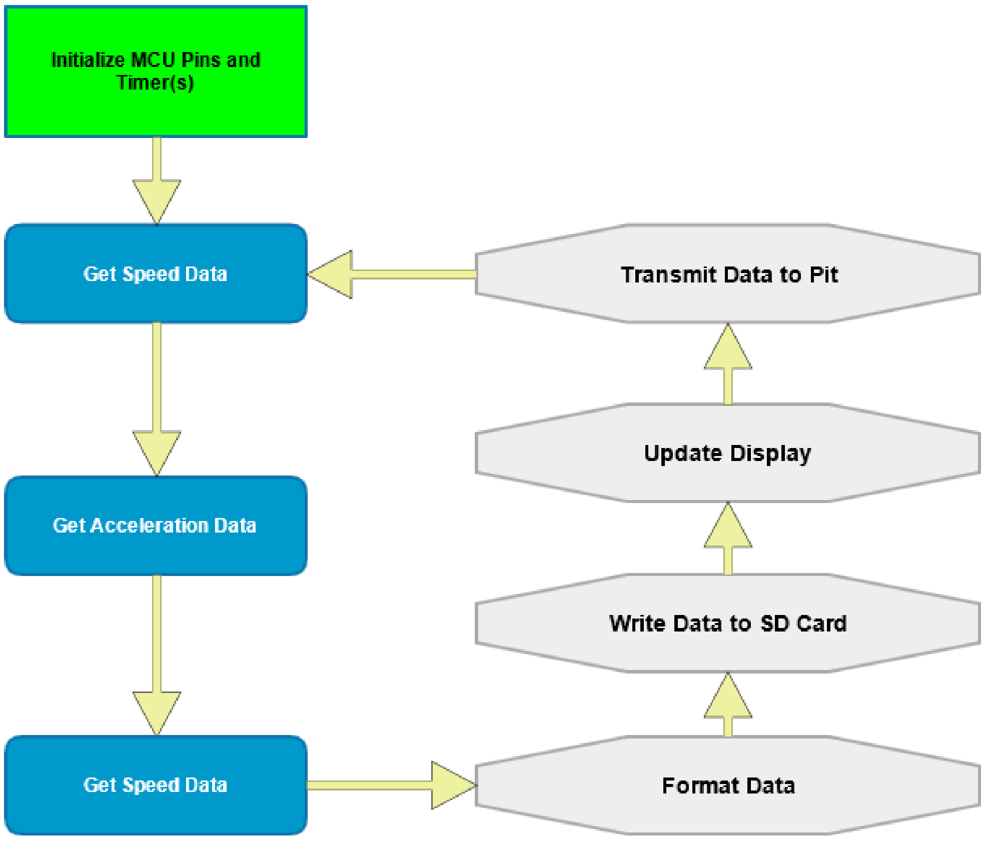


Figure 5: Control flow executed by the Hercules MCU

The LAUNCHXL-TMS57004 hosts a TI Hercules TMS570LS0432 MCU utilizing the ARM Cortex-R4 architecture and a 32 bit word length. It features dual cores in lock-step with error correction on MCU outputs and onboard memory, and can notify the system of errors via a dedicated error state pin. One of the most useful features of this MCU is Multi-Buffer SPI (MibSPI) which stores up to 128 words of queued data to be sent or received while the MCU executes other instructions. Multi-Buffer SPI also includes features such as transfer groups (which can be sent from or received to via interrupts), and interrupt outputs which correspond to successful sends/receives or transmission errors.

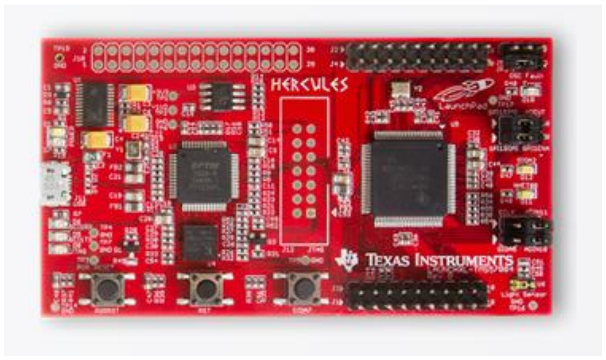


Figure 6: Hercules MCU

Many needed pins such as all MibSPI pins and a number of timer and GPIO pins are already broken out on the standard 40-pin Launchpad headers, but the optional 60pin header had to be installed in order to access the secondary SPI pins (which are also multiplexed on-chip and require some configuration to enable).

3.1.2 Sensor Network

The sensors, each driven by their own discrete microcontroller, collect data about the vehicle and transmit that data over the SPI bus to the main microcontroller. The sensor network will be broken down in detail in this section.

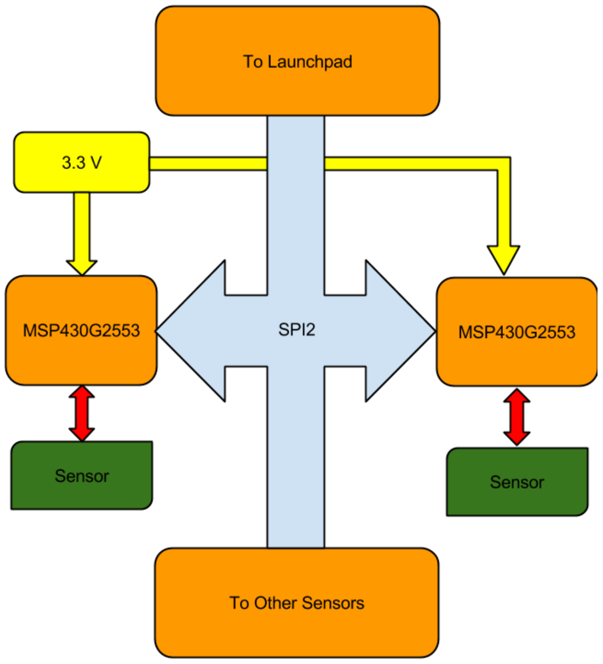
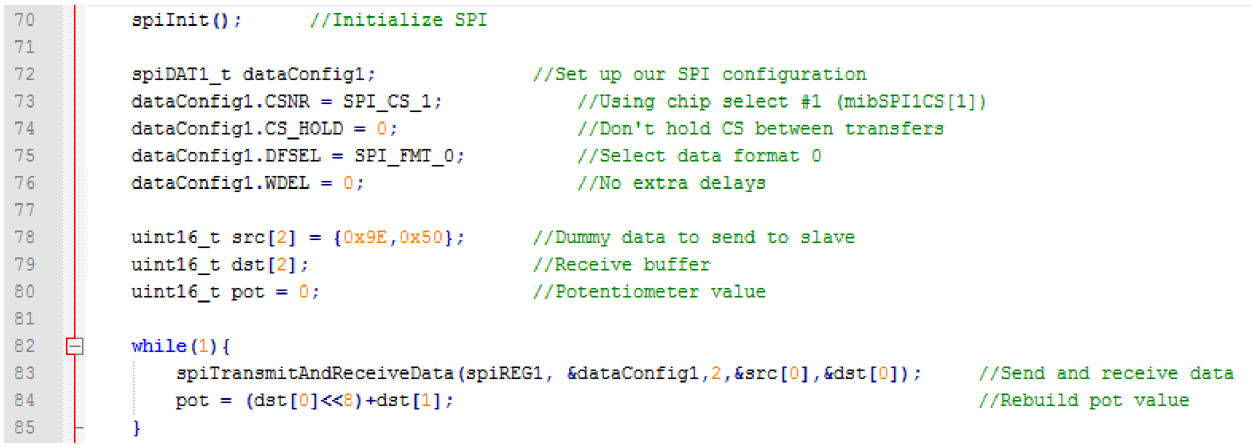


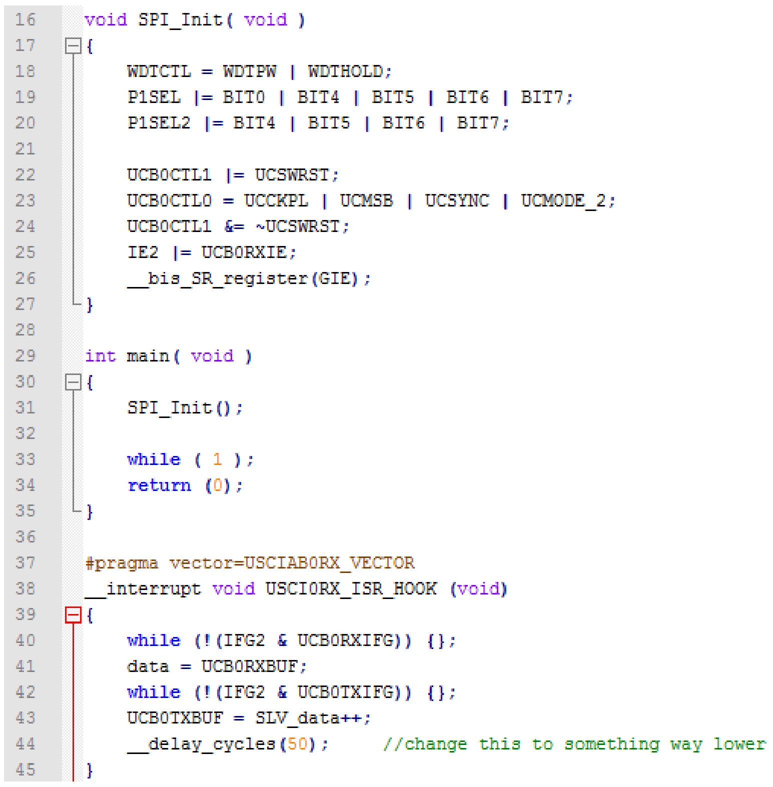
Figure 7: Top-level block diagram for the sensor network

The code in **figure 8** shows how to set up the Hercules' SPI interface to collect sensor data, and **figure 9** shows some sample slave code running on an MSP430. The full code is available on the team GitHub, accessible from the team website.



```c
spiInit();      //Initialize SPI

spiDAT1_t dataConfig1;            //Set up our SPI configuration
dataConfig1.CSNR = SPI_CS_1;          //Using chip select #1 (mibSPI1CS[1])
dataConfig1.CS_HOLD = 0;              //Don't hold CS between transfers
dataConfig1.DFSEL = SPI_FMT_0;        //Select data format 0
dataConfig1.WDEL = 0;                 //No extra delays

uint16_t src[2] = {0x9E,0x50};    //Dummy data to send to slave
uint16_t dst[2];                  //Receive buffer
uint16_t pot = 0;                 //Potentiometer value

while(1){
    spiTransmitAndReceiveData(spiREG1, &dataConfig1,2,&src[0],&dst[0]);    //Send and receive data
    pot = (dst[0]<<8)+dst[1];                                              //Rebuild pot value
}
```

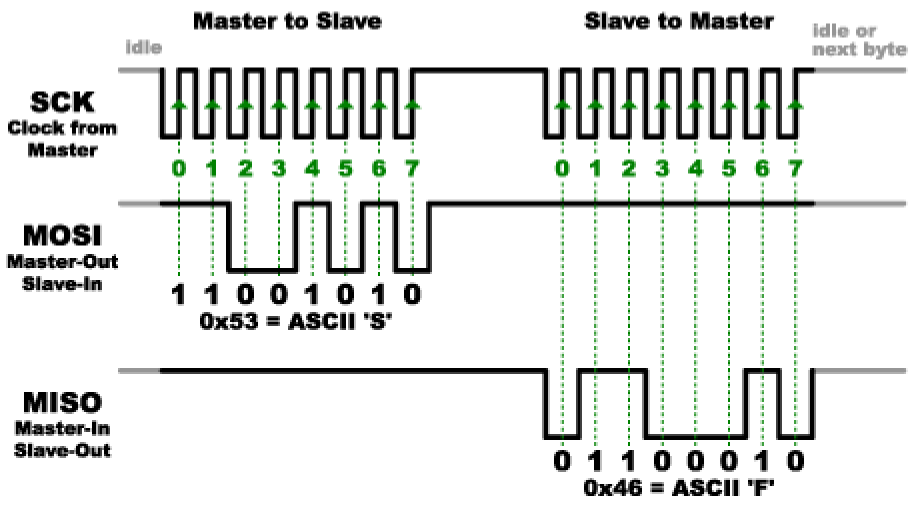
Figure 8: SPI Interface setup code



```c
void SPI_Init( void )
{
    WDTCTL = WDTPW | WDTHOLD;
    P1SEL |= BIT0 | BIT4 | BIT5 | BIT6 | BIT7;
    P1SEL2 |= BIT4 | BIT5 | BIT6 | BIT7;

    UCB0CTL1 |= UCSWRST;
    UCB0CTL0 = UCCKPL | UCMSB | UCSYNC | UCMODE_2;
    UCB0CTL1 &= ~UCSWRST;
    IE2 |= UCB0RXIE;
    __bis_SR_register(GIE);
}

int main( void )
{
    SPI_Init();

    while ( 1 );
    return (0);
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR_HOOK (void)
{
    while (!(IFG2 & UCB0RXIFG)) {};
    data = UCB0RXBUF;
    while (!(IFG2 & UCB0TXIFG)) {};
    UCB0TXBUF = SLV_data++;
    __delay_cycles(50);     //change this to something way lower
}
```

Figure 9: Sample MSP430 slave code

3.1.2.1 Data Transmission Bus



From learn.sparkfun.com

The team implemented both a single-byte and multi-byte SPI communications framework for the MSP430 microcontrollers. For single-byte data such as the speed sensor, the MSP430 simply updates its transmit buffer each time new data is available, however the multi-byte framework is more complex and requires the use of a preamble in order to properly align the bytes at sender and receiver.

3.1.2.2 Sensor Microcontrollers

Texas Instruments MSP430G2553 microcontrollers are used to take sensor readings and transmit them over the SPI bus to the main MCU. A PCB was designed and milled to house the MSP430, provide reset line stabilization, and break out the necessary connections for easy access. The boards also feature an optional reset button, and a 100 uF filter capacitor on the power input to the MCU.

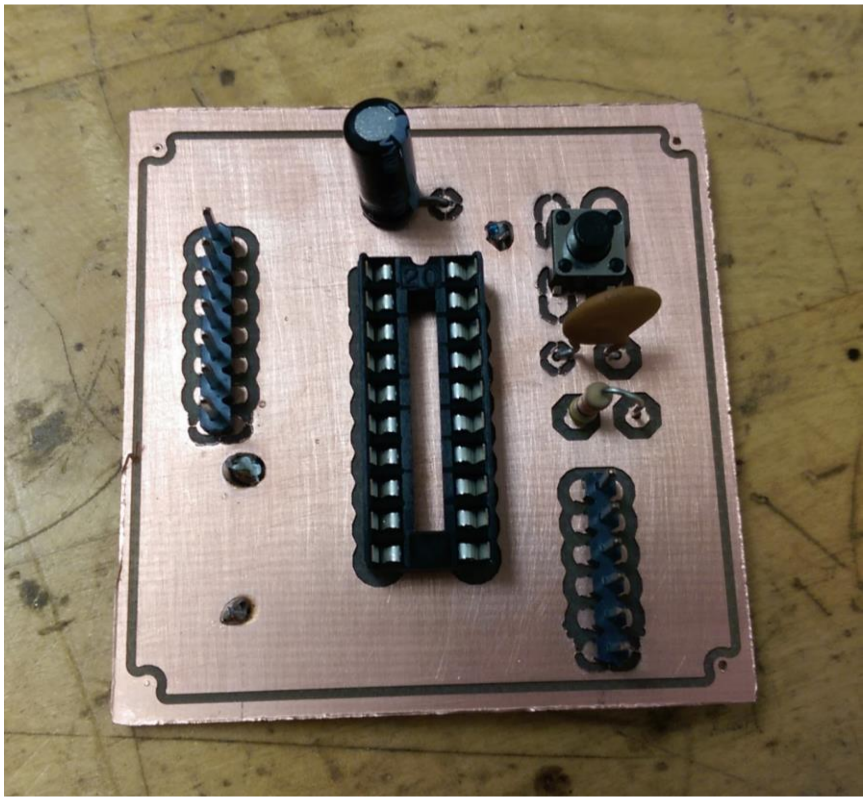


Figure 10: Sensor MCU PCB designed at the FAMU-FSU College of Engineering

This board is used as a platform for building sensor modules for the Baja car. It breaks out power, ground, and both serial interfaces (UCA and UCB), as well as analog pins 0-2, 4, and 5. The pinout of the UCB header matches with pins 2-7 of the UCA/ANALOG header for easily switching interfaces. More info about this board, as well as EAGLE and Gerber files can be found on the team GitHub.

3.1.2.3 Accelerometer Module

The accelerometer, along with most of the other sensors used in the system, is connected to a discrete MSP430G2553 microcontroller following the basic control flow shown in **figure 12**. The acceleration values are sent to the main microcontroller, where it is written to the log file and sent to the pit unit for display.

Due to development complications with the MSP430's onboard I^2C interface, communication with the LSM303 accelerometer is done using a "bit banging" technique whereby I^2C communication is emulated by the manipulation of GPIO pins on the MSP430. This technique is much slower than hardware, and so samples from the accelerometer take a significant amount of time to obtain.

The accelerometer module sends dual-axis acceleration data using the multi-byte SPI framework. It utilizes a two byte preamble, followed by 4 bytes of acceleration data - 2 bytes for each axis. It also implements a sample-and-hold policy by which a sample is requested from the main microcontroller by initiating a transfer and receiving the first byte of the preamble. This first byte is held in the buffer until a sample is obtained at which point the remainder of the preamble and the data is transmitted byte wise as normal. This maximizes the transfer time of the acceleration data while also allowing the MSP430 to perform its sampling while the Hercules is performing other tasks.

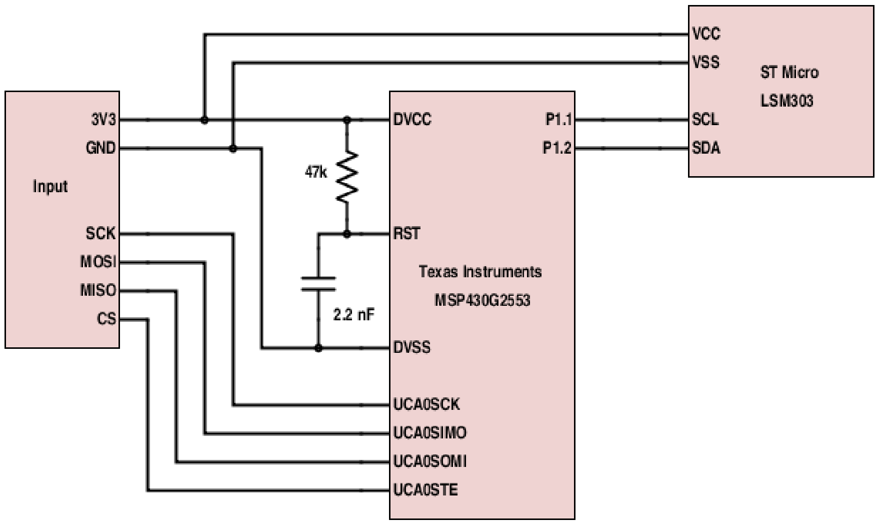


Figure 11: Pin-level diagram of the LSM303 sensor & its MCU

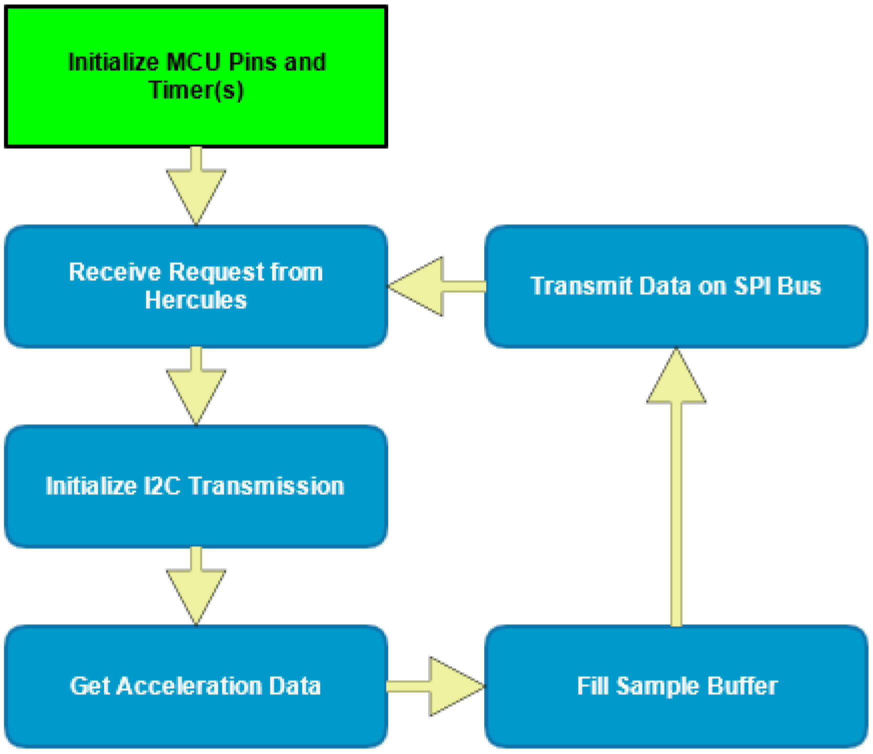

Initialize MCU Pins and Timer(s)
Receive Request from Hercules
Transmit Data on SPI Bus
Initialize I2C Transmission
Get Acceleration Data
Fill Sample Buffer

Figure 12: Control flow executed by the LSM303 Accelerometer MCU

3.1.2.4 Hall Effect Sensor Module

Hall Effect sensors increase their voltage output in the presence of a magnetic field. The Hall effect sensor will be used to measure the speed of the vehicle by affixing a permanent magnet to the drivetrain of the vehicle, and using the sensor to read when the magnet passes by. The Hall Effect sensor MCU will execute the control flow found below in **figure 14**. Once calculated, the speed of the vehicle is sent over the SPI bus using a single-byte transfer, and is displayed on the on-vehicle display.

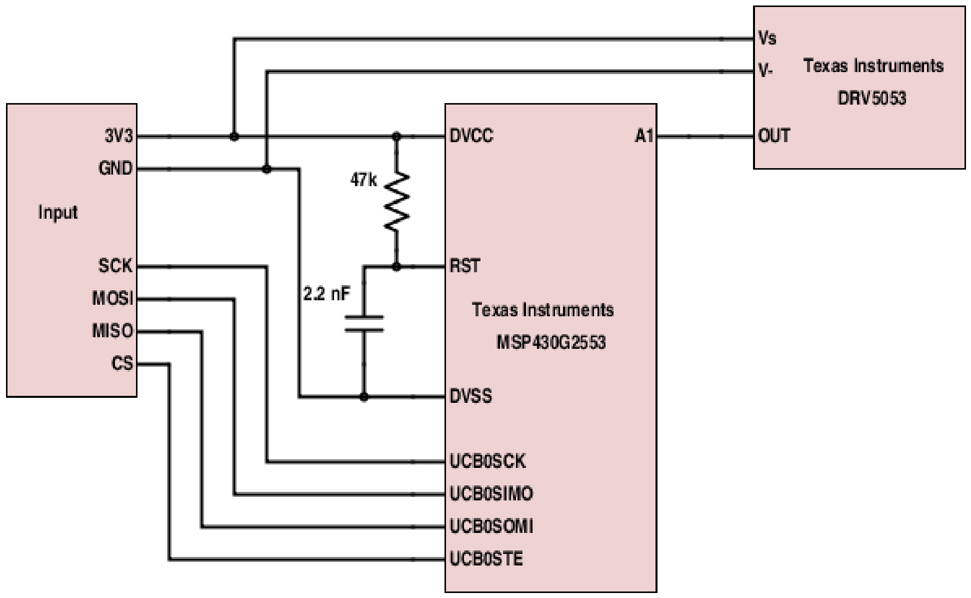


Figure 13: Pin-level diagram of the MSP430 MCU connected to the Hall Effect sensor

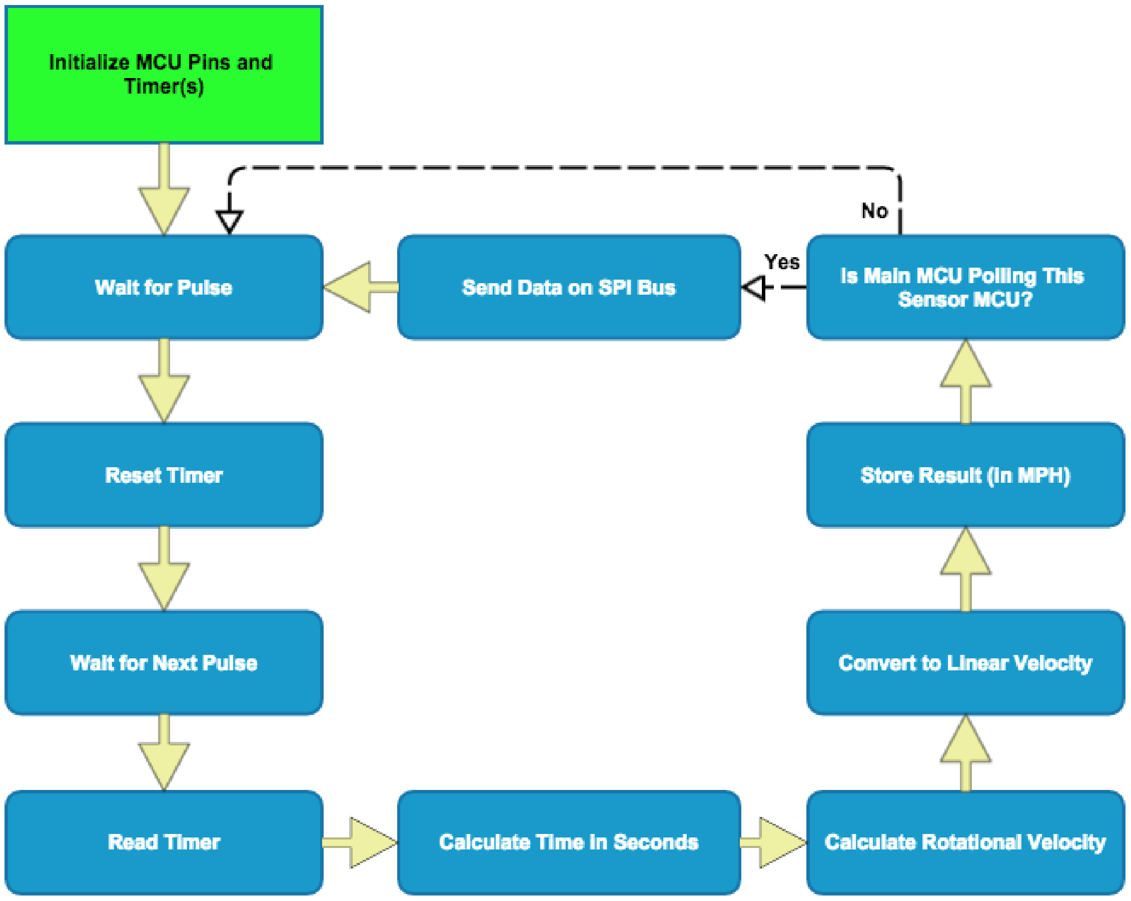

Initialize MCU Pins and Timer(s)
No
Wait for Pulse
Send Data on SPI Bus
Yes
Is Main MCU Polling This Sensor MCU?
Reset Timer
Store Result (in MPH)
Wait for Next Pulse
Convert to Linear Velocity
Read Timer
Calculate Time in Seconds
Calculate Rotational Velocity

Figure 14: Control flow executed by the Hall Effect sensor MSP430 MCU

3.1.3 Power System

The DAQ system requires its own power source, as per competition rules. The DAQ system's power system consists of two Lithium Iron Phosphate batteries and a circuit to regulate the battery voltage (~6.6V) down to 3.3 V and 5 V outputs for use by the other subsystems.

Table 8: Power Analysis of the system

Component	Voltage Rail	Active (mA)	Passive (mA)	% Active	# of Components	Average Power Draw (mW)
Hercules LaunchPad	5	92		100%	1	445.45
MSP430G2553	3.3	420 uA		100%	4	1.386
LSM303DLHC Accelerometer	3.3	110 uA		100%	1	0.363
XBee-PRO 900HP	3.3	215	29	30%	1	279.84
DRV5053 Hall Effect sensor	3.3	2.7		100%	1	8.91
Adafruit LCD Display	5	68		100%	1	340
SD Card Slot	5	66	6.60	30%	1	122
				Total Average Power Draw (mW)		**1197.949**

The block diagram in **Figure 1** shows the voltages that each subsystem requires, and this system is also shown in the pin level diagram in **Figure 4**.

In order to regulate the voltage of the batteries, switching voltage regulators are used. The DAQ system uses Texas Instruments TL2575 voltage regulators, in 3.3 V and 5 V variants. The regulator circuits include a schottky diode, a high current inductor, and a capacitor as per the datasheet. **Figure 15** shows the circuit implemented. On the output side, the DAQ uses a 1N5819 diode, a 150 uH inductor, and a 330 uF capacitor.

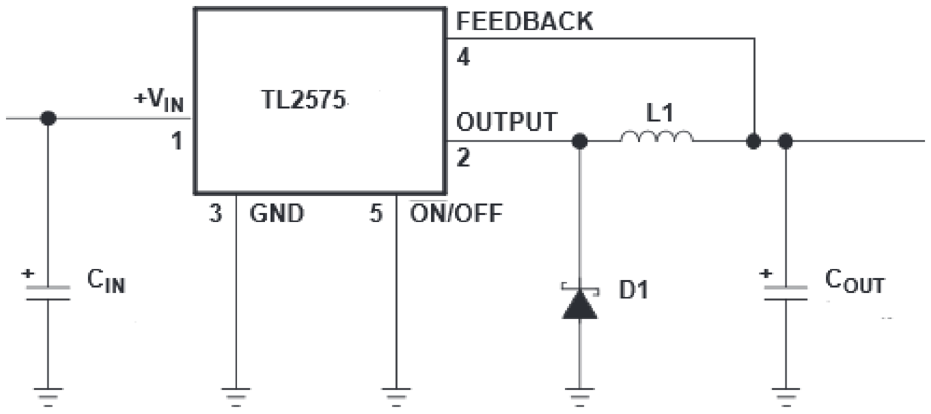


Figure 15: TL2575 voltage regulator circuit

A PCB was designed to house both TL2575 circuits, and provide both 5 and 3.3 V outputs from the nominal battery voltage.

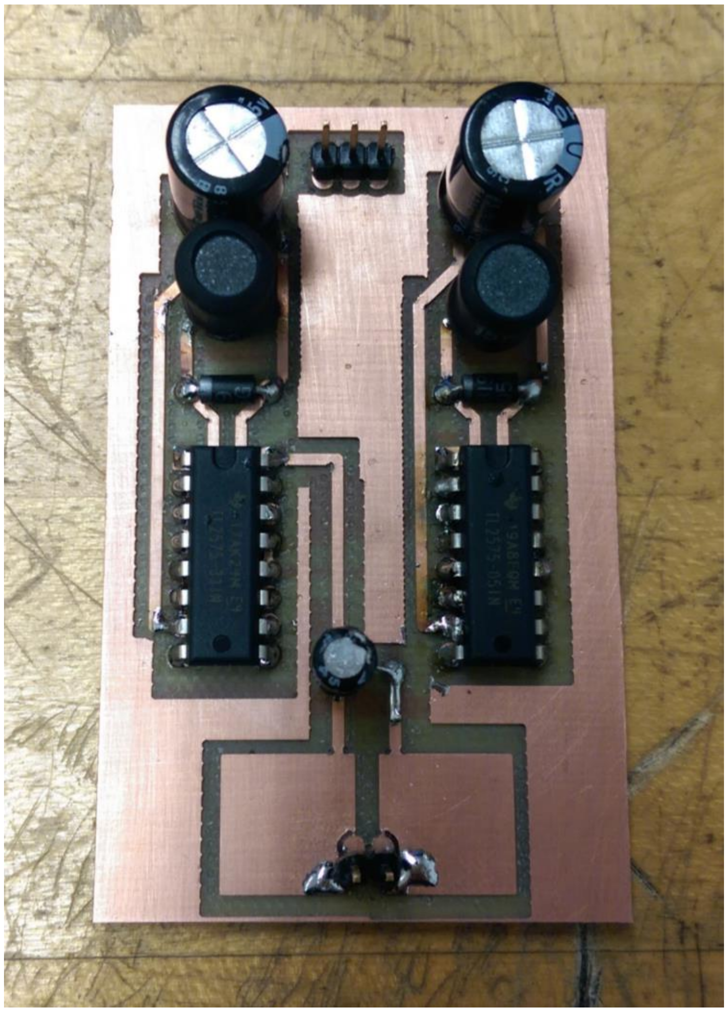


Figure 16: Voltage regulation PCB fabricated at the FAMU-FSU College of Engineering

3.1.4 Wireless Communication

An XBee PRO 900 HP RF transceiver pair is used to create a wireless link between the pit unit and on-vehicle DAQ system. The on-vehicle MCU sends serial data to the transceiver via the UART (or SCI). That data is repeated on the other end by the paired module and in turn received by the UART on the Raspberry Pi.

Figure 17

Data transmitted via the XBee includes a status byte (containing flag bits for the status of individual systems), an 8 bit speed value, and an 8 bit accelerometer value. The frame structure also includes a three byte preamble to ensure byte alignment.

3.1.5 Display/Storage

Once the main MCU collects and processes information from the sensors, the data is sent through a multi-buffer SPI bus (MibSPI) to the SD card and LCD display. An illustration of this subsystem is shown below.

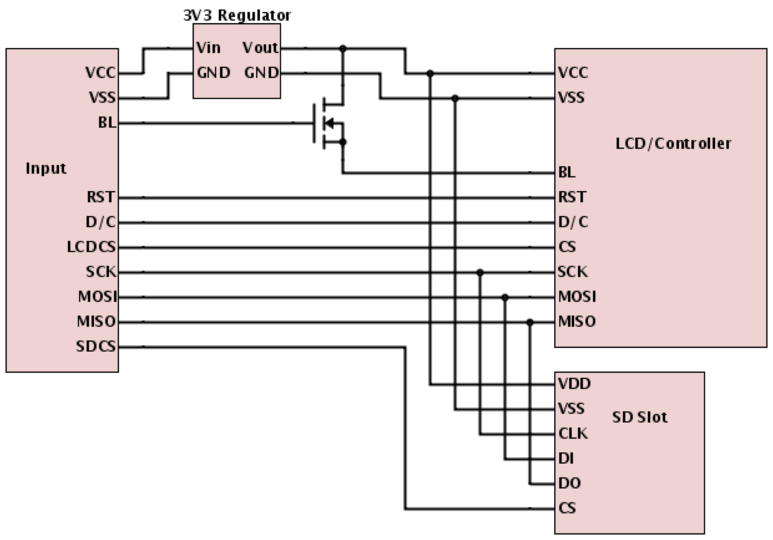


Figure 18: Pin-level diagram of the Adafruit display

3.1.5.1 Display

The display is mounted onto the frame in the direct line of vision of the driver and displays the speed and fuel timer, as well as any set notifications from the main MCU.

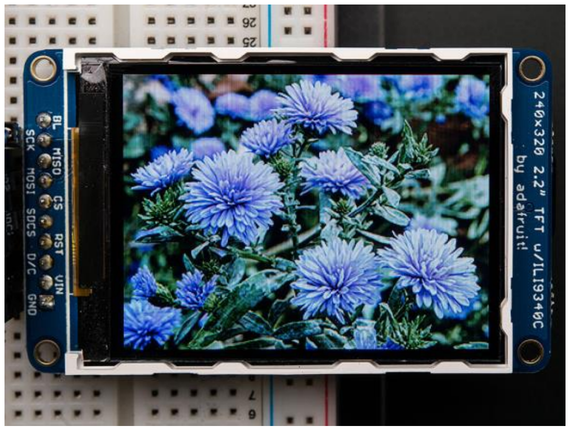
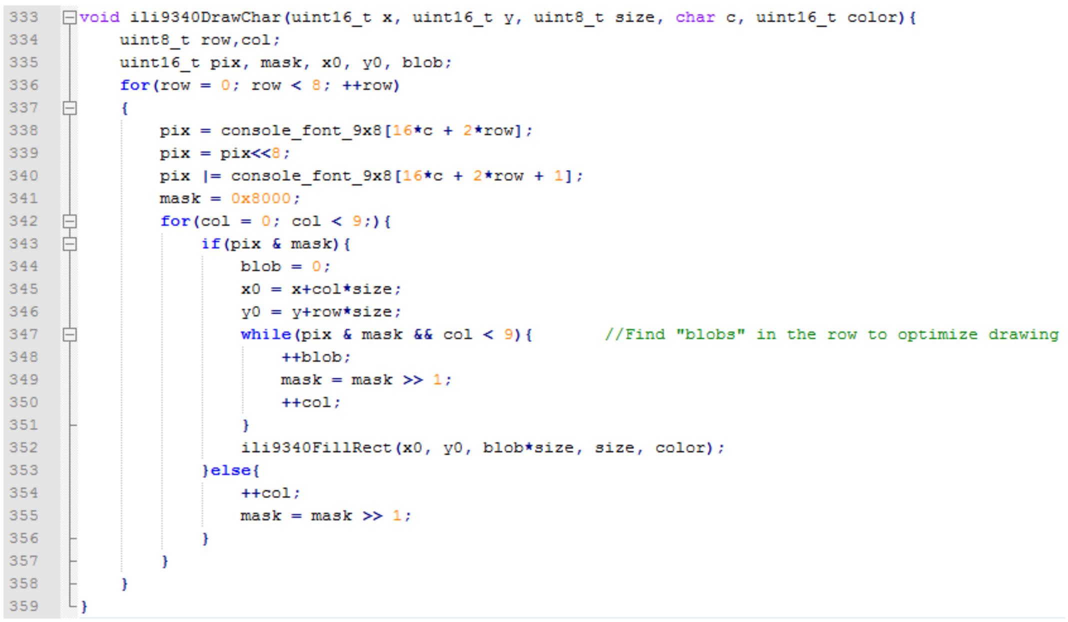


Figure 19: Adafruit 2.2" TFT Display

The display accepts commands and data over SPI. Writing to the display buffer involves defining a rectangular draw area, then sending 16-bit pixels until drawing is done or until the entire draw area has been filled. Driver code for this display can be found on the team GitHub. The routine in **figure 20** uses a rectangle drawing routine for writing text to the display using a simple 9x8 font.



```
void ili9340DrawChar(uint16_t x, uint16_t y, uint8_t size, char c, uint16_t color){
    uint8_t row,col;
    uint16_t pix, mask, x0, y0, blob;
    for(row = 0; row < 8; ++row)
    {
        pix = console_font_9x8[16*c + 2*row];
        pix = pix<<8;
        pix |= console_font_9x8[16*c + 2*row + 1];
        mask = 0x8000;
        for(col = 0; col < 9;){
            if(pix & mask){
                blob = 0;
                x0 = x+col*size;
                y0 = y+row*size;
                while(pix & mask && col < 9){       //Find "blobs" in the row to optimize drawing
                    ++blob;
                    mask = mask >> 1;
                    ++col;
                }
                ili9340FillRect(x0, y0, blob*size, size, color);
            }else{
                ++col;
                mask = mask >> 1;
            }
        }
    }
}
```

Figure 20: Routine that reads the font from program space and writes an ASCII character to the display

3.1.5.2 Storage

Once data has been collected and processed by the MCU, the data is then be written to a storage medium to be used for offline reference.

FatFS, an open source FAT file system library was used to allow the Hercules MCU to interface with the formatted SD card. FatFS provides a high level framework for navigating the file structure, as well as reading and writing to files, however low level driver code must be written to initialize the card, and to read and write 512 byte blocks over the SPI bus. The complete driver code uses the mibSPI module on the Hercules MCU to more efficiently control the SD card. This code can be found on the team GitHub.

3.1.6 Central MCU Housing

The center of the on-vehicle system was housed in a Plano Molding polycarbonate waterproof case. These cases are reasonably priced at $30 and the clamps and waterproof seal held up very well to the harsh operating environment outlined in **Section 1.3**. The box was altered by drilling holes for waterproof cable grommets, and is held in place with a custom 3-point steel bracket which was bolted to the frame of the vehicle. The batteries and circuit boards inside are affixed to the case with industrial strength Velcro, which held the parts secure even in intense racing conditions.

When choosing a project box, cost and size were the main factors. Many of the boxes large enough to accommodate the system (at least 10” long and 4” deep) were very expensive, and many boxes in the same price range as the Plano box were far too small.

3.2 Pit Unit

The pit unit is run with a Raspberry Pi Model B+. It receives periodic transmissions from the main MCU whenever the vehicle is in range (see design risk **6.4.2**). This unit serves as a supplement to the onboard vehicle data acquisition. As the Raspberry Pi offers a full operating system, the pit unit is operated using a standard keyboard and mouse which allows it to be used with minimal instruction or prior training for members of the pit crew.

3.2.1 Computer

The pit unit runs a GUI application that displays all of the pertinent data about the vehicle, including the status of DAQ subsystems, and selected sensor data. This GUI application was developed with Qt to quickly and easily create the interface itself, as well as simplifying the required polling of the UART, and timing of interface components.

3.2.2 Display

The pit unit includes a display to allow the crew to view the data transmitted from the DAQ system. The pit unit uses a 7” composite video display which operates on 12 V DC. The display is portable, and provides sufficient resolution for its purpose. The display is connected to the Raspberry Pi by a standard 3.5mm to RCA cable.

4 Test Plan

The data acquisition system subsystems were first tested individually, and then tested again as a whole system.

4.1 Component Test Plan

The first step in the test plan was to evaluate the performance of individual components to determine progression into the next stage. Performance evaluation includes identifying faulty hardware and/or user written software that occurred during initial development.

4.1.1 Accelerometer Module

Initial testing of the accelerometer involves connecting the sensor to a MSP430 Launchpad from TI. After successful connection, the accelerometer was then turned such that one of its axes was directly aligned with the gravitational pull of the earth. Knowing this value to be approximately 9.81 m/s^2 (1 G) the team could verify the accelerometer module to be functioning by comparing the value given by each axis to the gravity of earth.

4.1.2 Magnetometer

Because both the magnetometer and accelerometer are onboard the LSM303 chip, the testing of the magnetometer was performed at the same time as the accelerometer test, by pointing the module in different directions and examining the change in the magnetometer output. A stronger magnetic field can also be introduced in order to further test the module.

4.1.2 Hall Effect Sensor Module

A magnet was attached to a rotating item with a known rate of revolution. The Hall Effect sensor module was used to measure and calculate the speed of the rotating item. The calculated rate of revolution was compared to the known rate of revolution to determine whether or not the module was correctly calculating speeds based on the #defined circumference of the object as well as correctly detecting rising and falling edges of the magnetic pulses.

4.1.4 Wireless Communication Module

Testing this module involved testing the performance of the hardware out of the box to rule out errors from manufacturing. The XBee Pro modules were each connected to a computer via a USB explorer device. Using the X-CTU software, the XBees were configured and messages sent to one another through a console window. Operational condition was determined by verifying that the message received from one device matched the message sent from the other device.

The next form of testing was done by transmitting messages through an XBee from a computer and checking the Raspberry Pi connected to the receiving XBee to determine whether the signal transmission was a success. The test was repeated using the Hercules LaunchPad. Lastly, a transmission test between the Pi and Hercules will be performed to ensure proper final operation.

4.1.5 SD card storage

Once the low level driver code was developed, the read and write operations were be tested by using a PC to write test files and to verify that files written back to the card are intact and readable.

4.1.6 Display

Once example code was ported to utilize the Hercules' mibSPI mode, simple shapes such as lines and rectangles can be drawn to the display, confirming its full working condition. More advanced operations such as bitmap and font drawing capabilities are also available.

4.2 System and Integration Test Plan

Once individual system component testing was complete and all the subsystems are deemed to be operational, testing will then move into the integration phase. The system integration phase was separated into two categories of testing: the integration components and system testing.

As part of the integration components testing of the system integration phase, aspects of the data acquisition system required to combine subsystems were tested. This included items such as the

SPI communication bus and voltage regulation. Once testing for these integration components was complete, all of the subsystems were installed on the vehicle and tested as a whole system as part of system testing.

4.2.1 Integration Components

The following are components not directly part of the data acquisition subsystem set but still crucial for subsystem integration.

4.2.1.1 MSP430 ADC

The Hall Effect sensor outputs a voltage based on the strength of the magnetic field around it, which needs to be converted to a digital value using an ADC. For the Hall Effect sensor and all other analog devices, the analog output was connected to the onboard 10-bit ADC on the MSP430G2553 microcontroller. Testing the built in ADC involves developing baseline code to initialize and read from the ADC, as well as some external hardware to generate a known analog signal.

4.2.1.2 3.3V Regulator

A 3.3V input voltage is required to operate several devices in the DAQ. A switching voltage regulator circuit is used to provide this voltage from the higher voltage of the battery. To test the voltage regulator, 6-7V supplied to the circuit and the output voltage and the ripple were measured using an oscilloscope.

4.2.1.3 5V Regulator

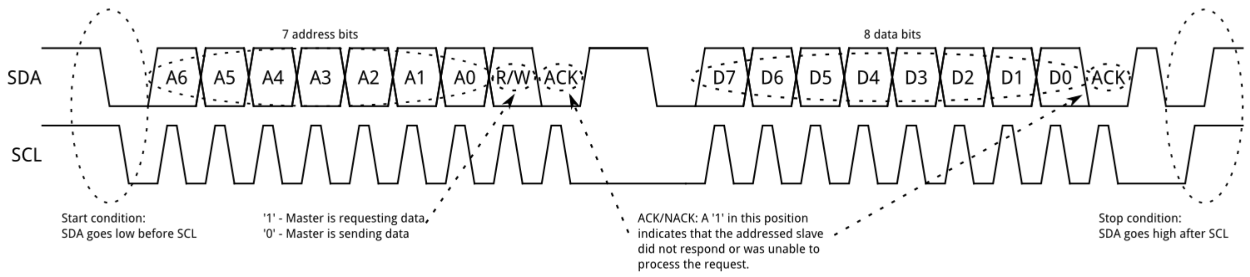
Several devices also require a 5V input voltage. The testing process for this regulator was the same as the process used for the 3.3V outlined above with the output voltage tested for 5V.

4.2.1.4 SPI

Both the Hercules and MSP430 devices need to communicate using the Serial Peripheral Interface, or SPI. Their respective interfaces were connected to an oscilloscope and their outputs viewed in order to observe the proper operation of the modules. Additionally, slave devices were connected and the ability to collect and send data was tested.

4.2.1.5 I^2C

The I^2C serial data bus is used to collect digital readings from the accelerometer. The accelerometer (and thus the I2C bus), was tested by reading/writing from/to the accelerometer with an MSP430. Similar to SPI, I^2C data transmissions could be examined using an oscilloscope to confirm that the devices are communicating properly.


7 address bits
8 data bits
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Start condition: SDA goes low before SCL
'1' - Master is requesting data, '0' - Master is sending data
ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.
Stop condition: SDA goes high after SCL

I2C timing diagram From learn.sparkfun.com

4.2.1.6 Battery Life and Power Consumption

The batteries arrived with cell voltage and charge labeled as measured in the factory. Those voltages were confirmed with a multimeter measurement. When the entire system was connected and running off of power, the current draw was measured at ~258 mA, less than originally anticipated. By the teams calculations, the batteries should last somewhere around 40 hours. As such, the team decided that it was not necessary to test the full battery life of the system. As of the writing of this paper, the batteries have gone uncharged and have not dropped in voltage at all.

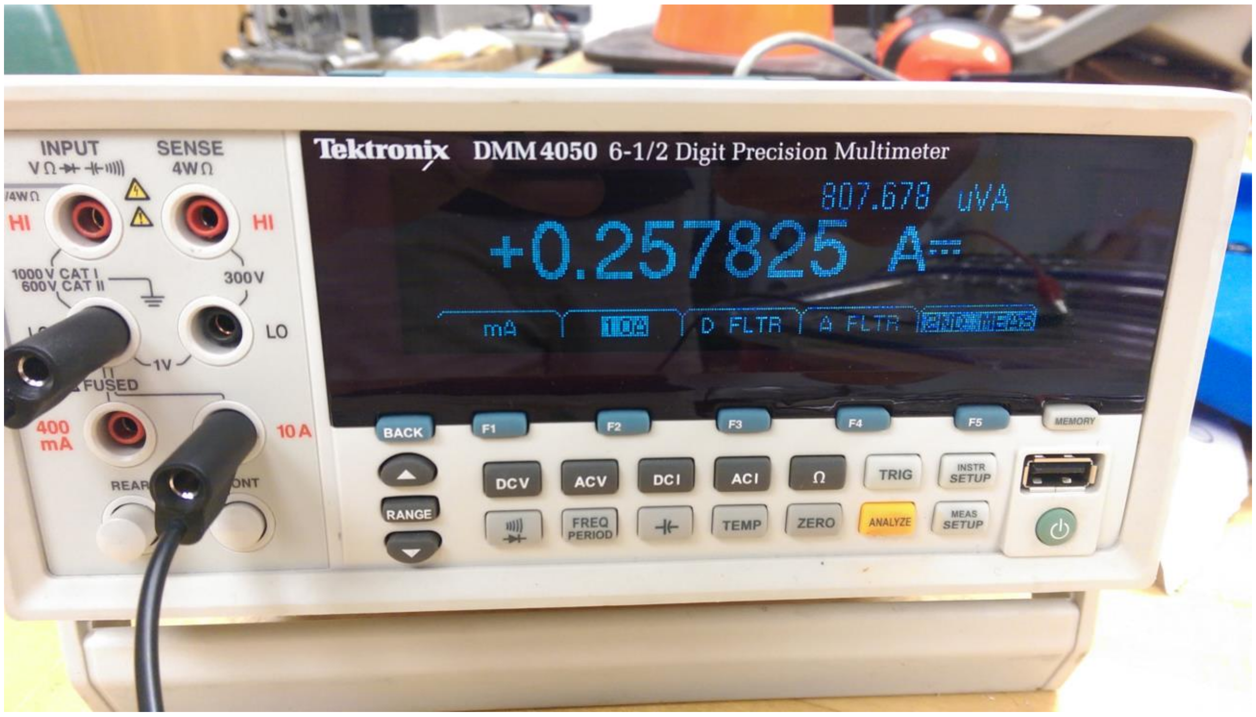


Figure 21: Total current draw of the running system

4.2.1.7 Project Box

This test allowed the team to assess the performance of the project enclosure seal when subjected to moisture. The test was performed by placing paper towels inside of the Plano project box and submerging the whole box under a body of water for a small period of time. The box and the paper towels were examined for any leakage.

4.2.2 System Testing

The following are tests for each DAQ subsystem after vehicle installation.

4.2.2.1 Wireless Communication

Testing for this subsystem began by manually setting constant sensor values on the Hercules and transmitting these values using the XBee modules to the Raspberry Pi in a lab setting. The data displayed on the Pi was checked for accuracy.

The next form of testing was done by pressing the driver's notification button on the vehicle and checking the display on the raspberry pi at a simulated pit crew station to determine whether the signal transmission was a success. The pit crew toggle button was pressed and the driver display examined to determine if reverse communication was a success. The data displayed on the Raspberry Pi was inspected to confirm that the data coming in falls in a realistic range.

4.2.2.2 Accelerometer

Given that preliminary tests are done on the accelerometer module, it is certain that the module will output correct values when installed on the vehicle. Some sanity checks can be performed to verify that extreme or obviously incorrect values are not being reported to the main MCU.

4.2.2.3 Hall Effect Sensor

The Hall Effect sensor module will use the rotations per second value collected from the sensor to calculate the speed of the vehicle in miles per hour. To verify the data calculated by the Hall Effect sensor module, a car with an accurate speedometer will run next to the car on a short stretch, and the speed readings will be compared. Any error can be corrected in the MCU's calculations.

4.3 Summary of Test Plan Status

The table below provides a summary of the completed subsystem tests. All component tests were completed by 4/4/2015. The installation of the DAQ ran right up until the last minute, with the entire system powering on the day of the Baja team's departure to Auburn, Alabama.

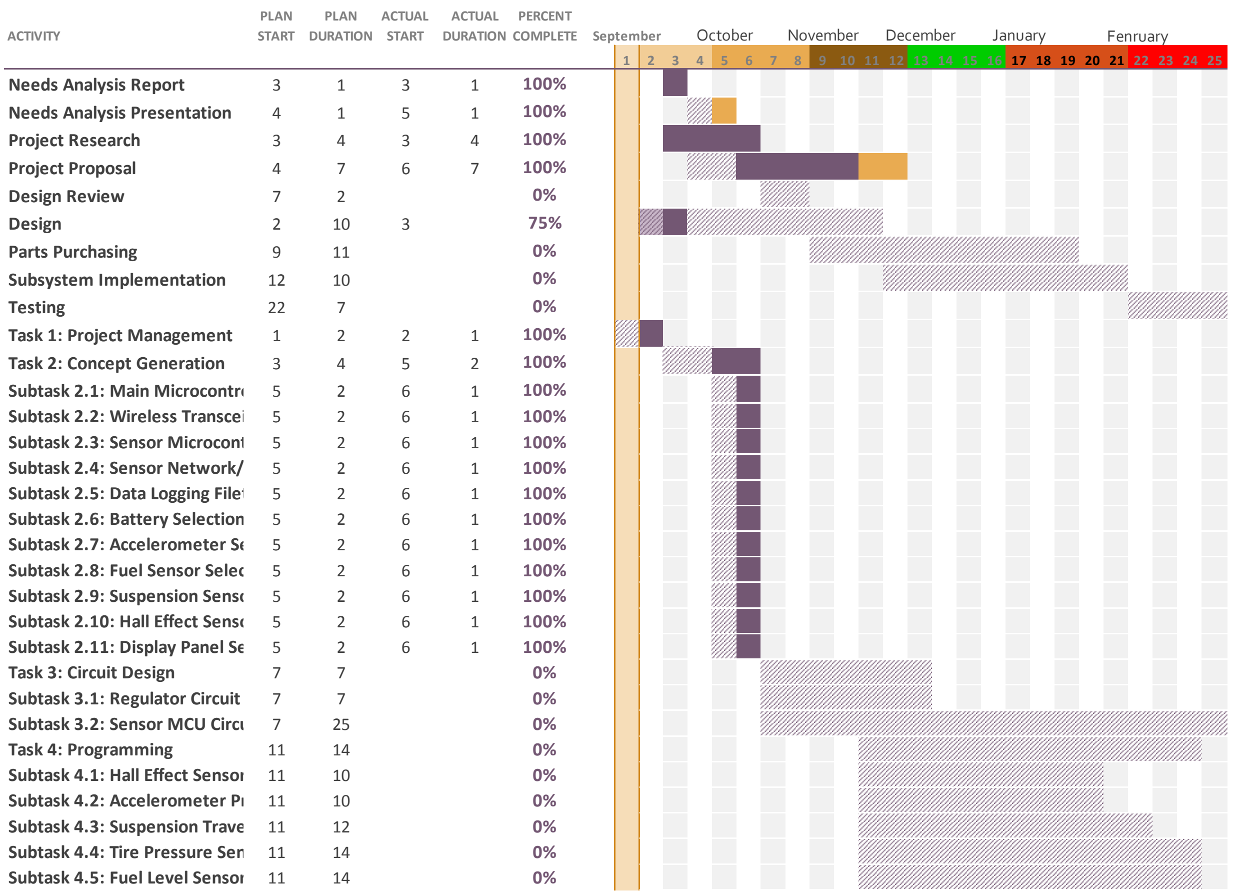
Table 9: Test Plan table

Title	Test Type	Attempt 1	Tester	Result
Accelerometer Calibration Test	Component	1/9/2015	Chris	Pass
Hall Effect Sensor Test	Component	2/15/2015	Chris	Pass
Xbee Baseline Test	Component	12/5/2014	Chris/Dewey	Pass
Xbee Short Range Test	Component	12/5/2014	Chris/Dewey	Pass
Xbee Long Range Test	Component	Not completed		
Xbee Raspberry Pi - Communication Test	Component	1/16/2015	Hebe	Fail

Hercules SD Card Test	Component		Dewey	Pass
Hercules Display Test	Component	1/14/2015	Dewey	Pass
LSM303 Calibration test	Component	2/20/2015	Chris	Pass
MSP430 ADC10 Test	Integration	1/25/2015	Chris	Pass
3.3V Switching Regulator Circuit Test	Integration	2/1/2015	Dewey	Fail
5V Switching Regulator Circuit Test	Integration	2/1/2015	Dewey	Fail
Hercules SPI Test	Integration	1/12/2015	Dewey	Pass
MSP430 SPI Slave Test	Integration	1/21/2015	Chris	Pass
MSP SPI Slave Mode with ADC10 Test	Integration	1/15/2015	Chris/Dewey	Pass*
MSP430 I2C	Integration	2/20/2015	Chris	Pass
I2C and SPI MSP430 Test	Integration	3/1/2015	Chris/Dewey	Pass
Battery Life Test	Integration		Not completed	
Project Box Waterproof Seal Test	Integration	3/5/2015	Chris	Pass
Plano Waterproof Seal Test	Integration	3/20/2015	Chris	Pass
MSP430 PCB Test	Integration	3/15/2015	Dewey	Pass
Xbee between Hercules and RPi Test	Integration	1/30/2015	Dewey	Pass
Hall Effect Sensor Accuracy Test	Integration	3/20/2015	Chris	Pass
Hall Effect Sensor Project Box Test	Integration	3/16/2015	Chris	Pass
Full System Test	Integration	4/4/2015	Chris/Dewey	Pass
Xbee Raspberry Pi - Communication Test 2	Component	1/23/2015	Dewey	Pass
3.3V Switching Regulator Circuit Test 2	Integration	3/5/2015	Hebe/Dewey/Chris	Pass
5V Switching Regulator Circuit Test 2	Integration	3/6/2015	Hebe/Dewey/Chris	Pass

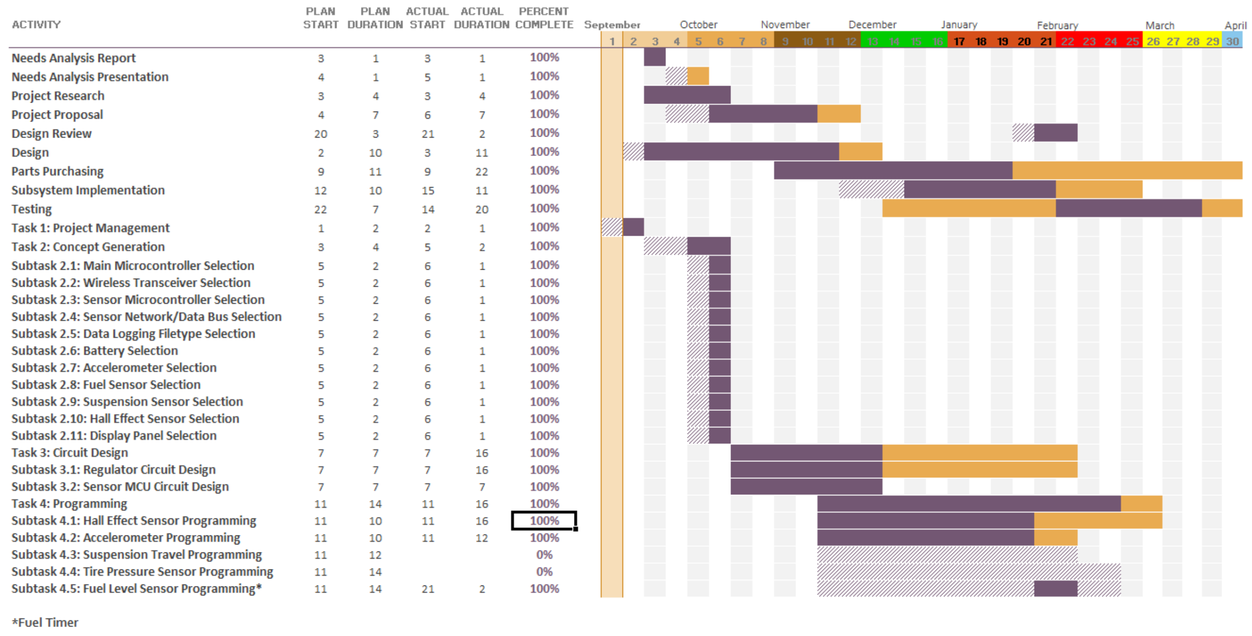
5 Schedule

Figure 22 below shows the preliminary schedule. The final schedule can be seen in **Figure 23**. The development of the system took much longer than anticipated, even with the descoping of some of the planned subsystems. With the chosen components and wide variety of communication protocols used, the project ended up being much more ambitious than any of the design team anticipated. Even with the project going far over the initial schedule estimate, the system was still complete in time for the Auburn, Alabama competition. However, the two-way communication between the pit and the driver was lost before the system could be used in competition. Still, the team considers the project a success.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Needs Analysis Report	3	1	3	1	100%
Needs Analysis Presentation	4	1	5	1	100%
Project Research	3	4	3	4	100%
Project Proposal	4	7	6	7	100%
Design Review	7	2			0%
Design	2	10	3		75%
Parts Purchasing	9	11			0%
Subsystem Implementation	12	10			0%
Testing	22	7			0%
Task 1: Project Management	1	2	2	1	100%
Task 2: Concept Generation	3	4	5	2	100%
Subtask 2.1: Main Microcontr	5	2	6	1	100%
Subtask 2.2: Wireless Transcei	5	2	6	1	100%
Subtask 2.3: Sensor Microcont	5	2	6	1	100%
Subtask 2.4: Sensor Network/	5	2	6	1	100%
Subtask 2.5: Data Logging File	5	2	6	1	100%
Subtask 2.6: Battery Selection	5	2	6	1	100%
Subtask 2.7: Accelerometer Se	5	2	6	1	100%
Subtask 2.8: Fuel Sensor Selec	5	2	6	1	100%
Subtask 2.9: Suspension Sens	5	2	6	1	100%
Subtask 2.10: Hall Effect Sens	5	2	6	1	100%
Subtask 2.11: Display Panel Se	5	2	6	1	100%
Task 3: Circuit Design	7	7			0%
Subtask 3.1: Regulator Circuit	7	7			0%
Subtask 3.2: Sensor MCU Circu	7	25			0%
Task 4: Programming	11	14			0%
Subtask 4.1: Hall Effect Sensor	11	10			0%
Subtask 4.2: Accelerometer Pr	11	10			0%
Subtask 4.3: Suspension Trave	11	12			0%
Subtask 4.4: Tire Pressure Sen	11	14			0%
Subtask 4.5: Fuel Level Sensor	11	14			0%

Figure 22: Initial schedule



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Needs Analysis Report	3	1	3	1	100%
Needs Analysis Presentation	4	1	5	1	100%
Project Research	3	4	3	4	100%
Project Proposal	4	7	6	7	100%
Design Review	20	3	21	2	100%
Design	2	10	3	11	100%
Parts Purchasing	9	11	9	22	100%
Subsystem Implementation	12	10	15	11	100%
Testing	22	7	14	20	100%
Task 1: Project Management	1	2	2	1	100%
Task 2: Concept Generation	3	4	5	2	100%
Subtask 2.1: Main Microcontroller Selection	5	2	6	1	100%
Subtask 2.2: Wireless Transceiver Selection	5	2	6	1	100%
Subtask 2.3: Sensor Microcontroller Selection	5	2	6	1	100%
Subtask 2.4: Sensor Network/Data Bus Selection	5	2	6	1	100%
Subtask 2.5: Data Logging Filetype Selection	5	2	6	1	100%
Subtask 2.6: Battery Selection	5	2	6	1	100%
Subtask 2.7: Accelerometer Selection	5	2	6	1	100%
Subtask 2.8: Fuel Sensor Selection	5	2	6	1	100%
Subtask 2.9: Suspension Sensor Selection	5	2	6	1	100%
Subtask 2.10: Hall Effect Sensor Selection	5	2	6	1	100%
Subtask 2.11: Display Panel Selection	5	2	6	1	100%
Task 3: Circuit Design	7	7	7	16	100%
Subtask 3.1: Regulator Circuit Design	7	7	7	16	100%
Subtask 3.2: Sensor MCU Circuit Design	7	7	7	7	100%
Task 4: Programming	11	14	11	16	100%
Subtask 4.1: Hall Effect Sensor Programming	11	10	11	16	100%
Subtask 4.2: Accelerometer Programming	11	10	11	12	100%
Subtask 4.3: Suspension Travel Programming	11	12			0%
Subtask 4.4: Tire Pressure Sensor Programming	11	14			0%
Subtask 4.5: Fuel Level Sensor Programming*	11	14	21	2	100%

*Fuel Timer

Figure 23: Final Schedule

6 Final Budget and Justification

The initial budgetary analysis originally proposed in the Project Proposal & Statement of Work can be seen below in **Table 10**, followed by the final budgetary analysis in **Table 11.** Clearly, the project required much more funding than initially anticipated. Even with the donation of components from Texas Instruments, the team still overran the provided $750 budget.

When completing the initial budgetary analysis, only the major components of the system were priced out and considered. The team did not include mounting hardware, headers, sockets, circuit elements, cable, display components for the pit unit, or enclosures (among other miscellaneous parts). The inclusion of these items as well as some design changes and unforeseen expenses pushed the actual expenses far beyond initial estimates.

One major unforeseen expense was the hazardous materials shipping charge added to the batteries, which amounted to $40 USD. This extra expense was one that the team did not anticipate, and one that was not readily listed on the website from which the batteries were purchased. In addition to this extra expense, the addition of all of the headers, sockets, cables, circuit elements, and mounting hardware added a significant amount of cost to the project.

In addition to unforeseen expenses, some design changes were made that either warranted the purchasing of new components, or made some purchased components useless to the project. Upon completing the code for the ADXL335 accelerometer, it was apparent that the +/- 3G range was too wide, to get a reading from the 10-bit ADC that was accurate enough. Thus, the new LSM303 digital accelerometer was purchased. The +/- 2G range coupled with the built-in 16-bit ADC gave the team enough accuracy to be able to take meaningful measurements of the vehicle's acceleration. In addition, the voltage regulators that were purchased went unused, as Texas Instruments donated switching regulators to the team as part of the Innovation Challenge entry.

Despite the unforeseen expenses and the unused components, the design team did their due diligence in selecting components. As such, nothing needed to be replaced as a result of incompatibility issues. Additionally, the team was, for the most part, diligent in their handling of the components throughout much of the duration of the project. At the end, however, an XBee module was accidentally destroyed which greatly affected the functionality of the system. This did not affect the budget as this occurred too close to the end of the project (at the Auburn, Alabama competition) to warrant the purchase of a new XBee.

Table 10: Initial budgetary analysis

Item	Quantity	Price Per Unit	Total Price
Hercules LaunchPad	1	$19.99	$19.99
MSP430G2332IN20	10	$1.86	$18.60
SD BoosterPack	1	$9.99	$9.99
ADXL335 Accelerometer	1	$14.95	$14.95
Fuel Sender	1	$25.00	$25.00
Xbee-PRO 900HP	2	$39.00	$78.00
LiFePO4 Battery	2	$40.99	$81.98
LiFePO4 Battery Charger	1	$19.95	$19.95
Magnets (30 pk)	1	$6.88	$6.88
Hall Effect sensor (5 pk)	1	$4.49	$4.49
Blank PCB	3	$3.95	$11.85
2.2" TFT SPI Display	1	$20.00	$20.00
Raspberry Pi (Model B)	1	$35.00	$35.00
	Extra Parts		$50.00
	Shipping		$150.00
	Total		**$546.68**

Table 11: Final budget report

Item	Quantity	Price	Item Total
XBee-PRO 900HP, P-MP, RPSMA	2	$39.00	$78.00
2.1dB 900MHz Articulating Dipole Antenna	2	$9.00	$18.00
XBee Explorer USB	1	$24.95	$24.95
XBee Breakout Board	2	$2.95	$5.90
Breakaway Headers – 2.54mm	3	$1.50	$4.50
10 pin Headers – 2mm	4	$1.00	$4.00
Raspberry Pi Model B+	1	$39.95	$39.95
ADXL335 Accelerometer Breakout	1	$14.95	$14.95
Dual Female USB Type A Connector	1	$1.39	$1.39
ROHM 5V LDO Voltage Regulator	2	$1.76	$3.52
ROHM 3.3V LDO Voltage Regulator	2	$2.17	$4.34
3M 20 pin DIP socket	8	$0.42	$3.36
8GB Micro SD Card	2	$6.49	$12.98
MSP430G2553 Chips	5	$2.80	$14.00
Tenergy 3.2V 24Ah LiFePO4 Battery	2	$39.99	$79.98
Tenergy 6.4V 1A 2-cell Charger	1	$9.99	$9.99
Neodynium Magnets, 10pk	1	$5.99	$5.99
2.5mm to RCA A/V Cable	1	$4.59	$4.59
12V 7" TFT Display	1	$26.20	$26.20
Cable Grommets, 30pk	1	$9.28	$9.28

Plano Large Waterproof Case	1	$50.12	$50.12
Estone Waterproof Enclosure	6	$3.79	$22.74
LSM303 Accelerometer+Magnetometer	1	$14.95	$14.95
2.2” 18-bit color TFT LDC display w/ MicroSD	1	$24.95	$24.95
30-pin Male Header	3	$1.47	$4.41
30-pin Female Header	3	$2.13	$6.39
20-pin Female Header	2	$1.53	$3.06
3M 20-pin DIP Socket	3	$0.58	$1.74
47k Resistor	10	$0.05	$0.54
2.2 nF Ceramic Capacitor	10	$0.08	$0.80
100 uF Capacitor	10	$0.12	$1.20
330 uF Capacitor	5	$0.18	$0.90
150 uH Inductor	5	$0.42	$2.10
1N5819 Schottky Diode	5	$0.13	$0.65
50 ft Cable – 9 conductor 24AWG	1	$54.04	$54.04
6x6 Blank Copper Clad PCB	2	$9.49	$18.98
2”x4” Industrial Velcro Strips	4	$2.97	$11.88
Black Zip Ties	1	$6.47	$6.47
8”x10” Polycarbonate Lexan Sheet	1	$3.98	$3.98
2 Oz Gorilla Glue	1	$4.97	$4.97
#6x3/8 Sheet Metal Screws	1	$1.18	$1.18
M6 Lock Washers	3	$0.75	$2.25
6mm Hex Nuts	3	$0.58	$1.74
M6 Cap Screws	3	$0.90	$2.70
36” ZMAX Steel Strap	3	$2.98	$8.94
M2.5x20mm Machine Screws	4	$0.37	$1.48
M2.5 Hex Nuts	4	$0.23	$0.92
M2.5 Washers	4	$0.17	$0.68
Shipping & Handling + Tax			$134.33
Discounts			-$3.99
Total			**$750.97**

7 Future Considerations

As this project may be passed down to other design teams in the future, there are some considerations that the team would recommend that anybody picking up this project in the future take into account when making their design decisions.

7.1 Operating Environment

When designing the DAQ system, the team had little information about the environment in which these Baja competitions take place. At the Auburn, Alabama competition, the actual events that the vehicle participated in occurred quite far away from the where the paddocks were located (i.e. the pit – where vehicle repairs take place) as well as from the refueling station. With this in mind, future teams should seriously consider the purchase of more expensive wireless transceivers (e.g. the Xtend transceivers, which still have a relatively low rated urban range) if it is within the provided budget. Although the team did not get to use the XBee communication in competition (the pit XBee was inadvertently destroyed before arriving), it is unknown whether or not they would have been sufficient in the environment in which the Auburn, Alabama competition took place. However, it is unlikely that the XBee would have given an urban range (given the amount of signal interference due to trees, etc.) that was sufficient enough to prove useful in the competition.

In relation to the location of the paddock, another issue the team ran into was the lack of portability of the pit unit. When designing the system, the team was under the impression that power would be available to run the pit unit on (and that the paddocks would be relatively close to the location of the endurance event). Future teams should take the location of the paddocks in the competition site into consideration when designing the pit unit. Had the unit been more portable and run on its own power supply, the system may have proved to be very useful (barring the destruction of the wireless transceiver) during the competition.

At competition, the vehicle had to be frequently modified due to technical inspections and damage during the dynamic events and the endurance race. The next DAQ implementation should be designed in a way so that it is easily removable while still being rigidly mounted. Doing this will decrease the probability of data acquisition system damage due to frantic removal of the DAQ system to access/repair the car during time critical events. A more user friendly version of the DAQ system would include using less cable and a smaller main component box.

It should be noted that the environments in which these competitions are held are volatile. A competition at another location could be held in a completely different environment/topology than the Auburn, Alabama competition.

7.2 Tire Pressure Monitoring System

In order to develop subsystem, future teams should try their best to find sensors that are well documented and easy to use. However, most commercially-available TPMS sensors operate using proprietary data frameworks. While the team was aware that the sensors automatically transmit temperature and pressure data every 60 seconds, the framework of that data was unknown, and that information was difficult to receive through Schrader (the sensor manufacturer). It is recommended that, if using Schrader sensors, they be contacted very early and very persistently in order to receive the necessary information to be able to develop this subsystem. Additionally, indirect TPMS is another viable option that can be developed with speed measurement of each wheel on the vehicle. With that in mind, the usefulness of this subsystem is questionable. The team noticed that during competition, tires were more likely to be blown out completely due to the rough nature of the course than to be gradually deflated over time.

7.3 On-Vehicle Display

The structural integrity of the display is something that none of the design team had considered when choosing components for the system. Upon turning the system on the morning of the competition, the team discovered that the display was broken, making a significant amount of screen area unusable. Future teams should consider a more robust display method. Analog gauges and warning lights, much like those found in automobiles, may be more reliable in the rough conditions inherent in Baja competitions. Seven-segment displays may also prove worthy in this situation. An important note to make is the fact that the SPI TFT display used has a built-in microSD slot. As such, if the screen is replaced with something more robust, a different discrete microSD slot must be used. Additionally, using a more robust analog gauge or seven-segment display would mean that the user interface would be far less pleasing to look at, which is, in the opinion of the design team, far less important than reliability.

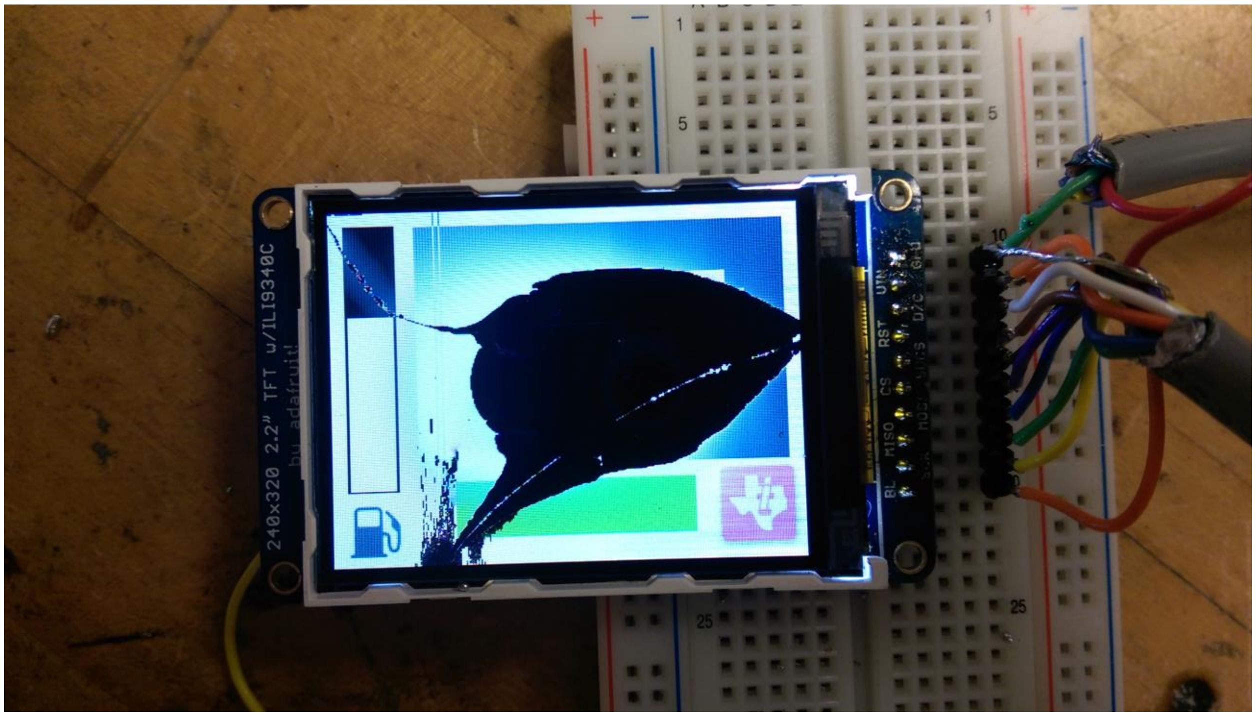


Figure 24: Damaged driver display

7.4 Speed Measurement

During the development of the speed measurement subsystem, the code was developed as if the magnet would be directly in front of the sensor with very little space in between the two. The DRV5053 Hall Effect sensor outputs 1V DC when there is no magnet near the sensor, and 2V DC when there is a magnet directly in front of the sensor. Given a reference voltage of 2.5V minimum, this gives a very small range of ADC values to work with when detecting rising and falling edges. The team ran into issues when the magnets and sensor were finally mounted to the vehicle. There were very few mounting options to work with. At the final mounting position, the magnets were much farther away from the sensor than they were when the subsystem was being developed. This greatly reduced the range of ADC values between the magnet's farthest and closest position to the sensor. So much so that finding rising and falling edges became near-impossible.

Perhaps with more development time this problem could have been resolved, but due to the schedules of the Baja team and the DAQ team there was very little (read: none) time for debugging after the system was installed on the vehicle. It is believed that a more sensitive sensor with a wider output range would be easier to work with given the mounting situation with the current vehicle.

8 Conclusion

Overall, the design team is happy with the final product. Even after some of the subsystems and features were ultimately cut from the project for various reasons, the team still feels that the project was extremely successful, especially given the ambitiousness of the project. Although the team was not able to implement all of the data acquisition subsystems that they would have liked, the two-way communication between the driver and the pit is something that could prove to be very useful, and something that the Baja team was very excited about. Unfortunately, with the destruction of the pit unit's XBee module, this usefulness was not actually able to be tested in a competition situation, which was what the system was originally designed for.

Initially, the design team had great hopes for how the DAQ system would propel the Baja team to a higher placement. However, the problems that the Baja team faced during the Auburn, Alabama competition were not things that could have been solved/helped by the DAQ system. Additionally, with the use of walkie-talkies and the sheer amount of time it took for the vehicle to be towed off of the course and to the paddocks (as it is when something major goes wrong), the Baja team had fair warning of the vehicle's arrival. Perhaps if the pit station were set up near the refueling area it would have proved useful, however the distance between the refueling station and the paddocks made it unquestionable as to what the driver was returning for (e.g. the driver would never return to the refueling area for a flat tire, and would never return to the paddocks to refuel). However, it is not to say that the application of the DAQ system to tuning the vehicle would have proved useless, as the system can still collect some very useful information when it comes to tuning the vehicle for specific events. With the timeline of project completion, however, the Baja team did not get any time to actually use the system for tuning purposes, as very little (if any) tuning is done at the competition.

Although the system lost the bulk of its functionality right at the beginning of the Auburn, Alabama competition, completing this project still turned out to be a great learning experience for all of those involved, including some of the Baja members that showed a genuine interest in the system and how it works. The use of various communication protocols along with the development of data-transmission frameworks was invaluable experience that will surely prove useful in real-world applications. All things considered, the design team is very happy with the product that it produced and the lessons that came with the production of that product, and hopes that any future DAQ teams receive the same learning experience.

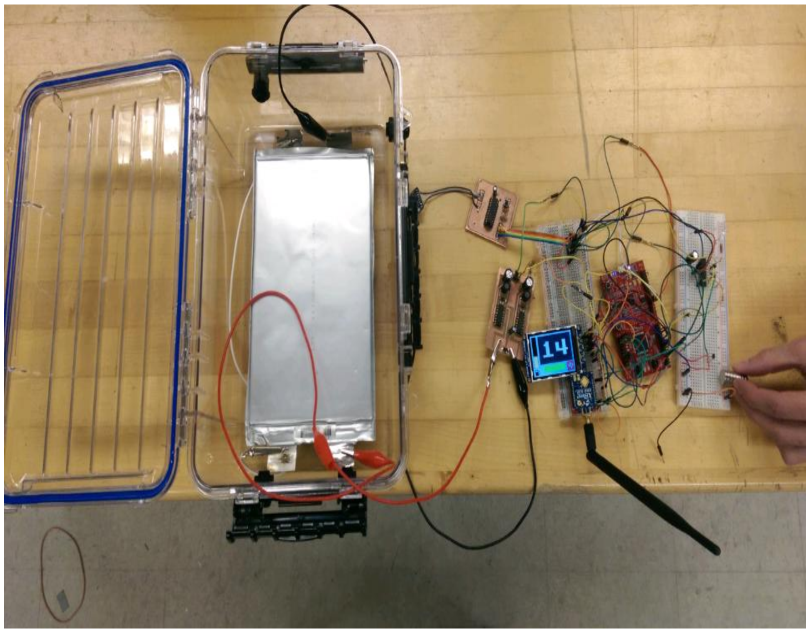
Appendix A

User Manual

SAE Baja Data Acquisition System User Guide

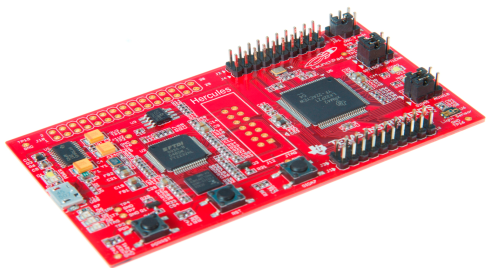
Introduction

Welcome to the user manual of the Baja Data Acquisition System(DAQ) designed by the Senior Design Team E #5. The following guide will show how to properly operate and maintain the DAQ system while in the user's care. The system was designed for the 2015 SAE Baja car to record values from various vehicle systems and will be stored for later use. DAQ subsystems include speed and acceleration measurements, as well as a timer to estimate fuel level. In addition to recording and storing data, the DAQ system will transmit the collected data wirelessly to a pit unit, along with a driver-to-pit notification so that the driver can notify the pit of his/her return. The image below illustrates the main components of the DAQ system.



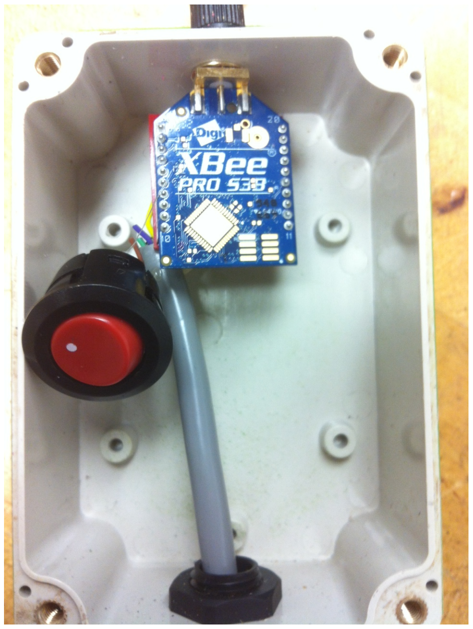
What's Included

- 2 - $LiFePO_4$ 24Ah batteries
- 1- Charger
- 1 - Adafruit 2.2" TFT display inside enclosure
- 1- Notification Button
- 1 - LSM303 accelerometer inside enclosure
- 1 - Plano waterproof enclosure. Contents include:
    - Hercules LAUNCHXL-TMS57004
    - Voltage Regulator circuit (on custom PCB)
    - Hall effect sensor circuit (on custom PCB)
- 1 - Texas Instruments DRV5053 Hall effect sensor
- 1 - Xbee-PRO 900HP wireless transceiver inside enclosure
- 1 - Raspberry Pi (B+)
- 2 - Xbee-PRO 900HP wireless transceiver
- 1 - 7" TFT 12V Display



Getting Started

Before installation of the data acquisition system, ensure the batteries are fully charged. To do this:



- Make sure the DAQ system on/off switch is in the off position (pushed towards the dot). The on/off switch can be found inside of the Xbee enclosure.
- Using a voltmeter, measure the voltage across the terminals of the batteries located inside of the plano box. If the voltmeter reads 6.4 V or greater, the batteries are fully charged and the DAQ system is ready for installation. Otherwise, continue reading for battery charging instructions.
- Next, disconnect the battery connections in the plano box by first disconnecting the red voltage source cable and following with the black ground cable.
- With the charger provided on hand, slowly remove the battery from its housing being careful not to damage the batteries, as they can bend under force.
- Attach the charger’s alligator clips to the leads of the batteries, making sure to double check the polarity of the batteries.
- Plug in the charger and wait for the charger’s LED to change from red to green.
- Once fully charged, return the batteries to the main enclosure and reconnect the DAQ system to the batteries, again making sure to check the polarity.

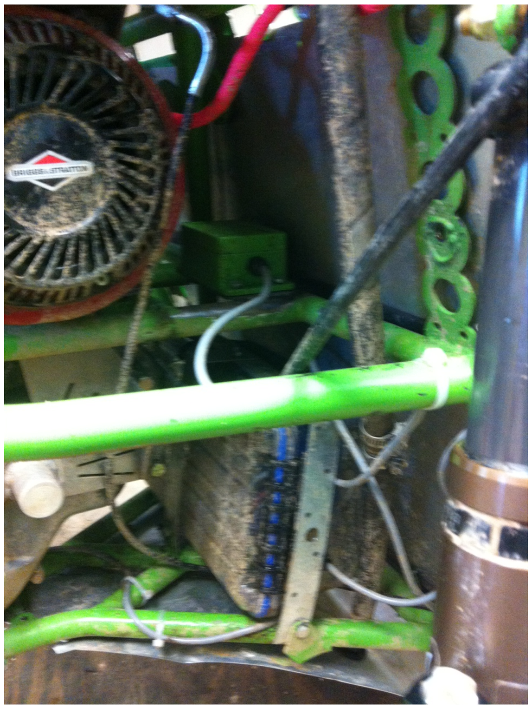
Installation

The DAQ system comes pre soldered and ready to use out of the box. The only setup needed is the mounting of the system onto the vehicle. The details of this instruction set will vary with different vehicles.

*Note: Due to the rugged environment Baja cars are typically used, be sure the components are rigidly mounted. Ensure that all project boxes have washers underneath the bolt heads in order to avoid damage to the enclosures.

Installing the on-vehicle system:

- Begin by placing the accelerometer enclosure in an area of the vehicle as close to the center of mass as possible.
- Next, insert the main enclosure into the vehicle centered around tabs welded onto the car.
- Place the three sided main enclosure bracket around the plano box and align the holes to the tabs. Use bolts to hold the bracket in place.
- Find a place with as little obstruction as possible but still protected from vehicle rolls.



This is where the XBee enclosure will be. Align the mounting holes on the enclosure to a structurally rigid area and secure with bolts.

- The display enclosure should be placed such that the driver can clearly view the display. There are already brackets mounted onto the vehicle on which the system was installed. If installing on a new vehicle or if the bracket is no longer present, it would be advisable to fabricate a new bracket. This should be welded onto the frame, and the box should be securely attached with bolts.
- The drivers button should be securely zip tied along the frame to an area easily accessible to the driver, or optimally attached to a tab welded into the frame.
- Given that the hall effect sensor is the most difficult to remove, this should be left last to install and all other DAQ system pieces should be secured in place before installing the hall effect sensor. Begin by removing the bolt of one of the rotating axles directly proportional to the rotating tire.
- Align the hall effect sensor plate hole to the bolt hole and replace the bolt.
- Repeat with a neighboring axle bolt. Take care not to disalign the rotating axle.

Assembling the pit crew station:

- Start by connecting the display's A/V cable to the Raspberry Pi A/V port
- Attach the XBee shield onto the Pi's expansion header, and plug the XBee into its socket
- Using an inverter connected to a 12V battery, connect the displays power terminals directly to the terminals of the 12V battery and plug in the Raspberry Pi to the inverter using a micro USB cable connected to a powered USB hub.

Operation

Once installed and powered on using the switch inside the XBee box, the data acquisition system will immediately begin collecting data and wirelessly communicating to the pit station via the XBee modules.

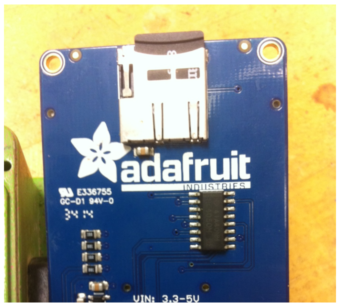
To turn on:

- Open the XBee enclosure and turn the switch to the ON position (away from the dot)

To read stored data:

While holding the display enclosure

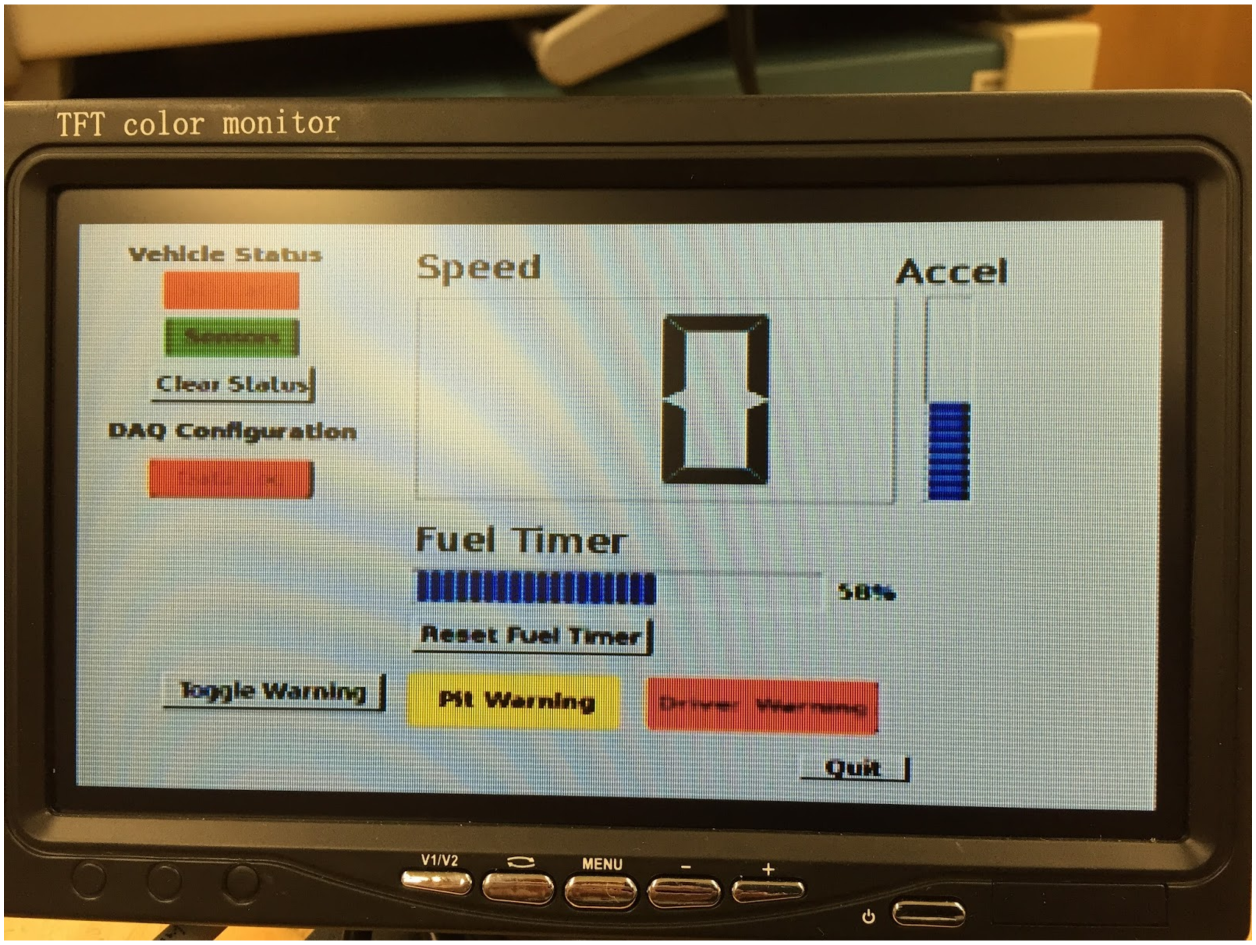
- Make sure the data acquisition system is turned off
- Loosen the grommet holding the cable going into the enclosure
- Unscrew the corners and open the lid
- An SD card is located on the back of the Adafruit display
- Push the SD card in and the slot will push out the SD card.

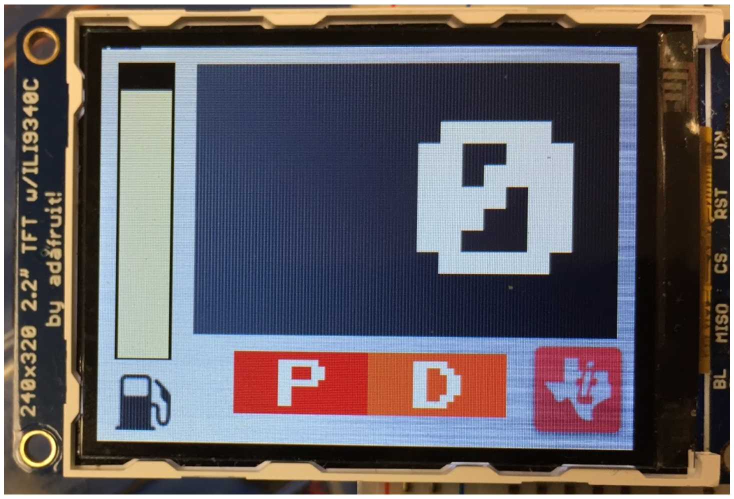


- Remove and insert into a SD compatible machine

For wireless communication:

- Connect a keyboard and mouse to the USB ports of the provided raspberry pi B+
- Power on the raspberry pi B+ by connecting the microUSB cable
- Log into the system using the default username and password
    - Username: pi
    - Password: raspberry
- Use the command “startx” to bring up the Raspbian desktop
- From the desktop, open up a command terminal
- Change to the display test directory using the command “cd ./guitest”
- Run the display text program with “./guitest”
- If the DAQ system is running, data should begin appearing on the screen
    - Orange status boxes indicates that the DAQ system is not running, or it is out of range
    - Red status boxes indicate that there has been some kind of failure on whatever module the indicator is for.
    - The “Toggle Pit Warning” button will send a signal to the driver, signaling him/her to return to the pit. This warning signal will only appear on the drive display if the on-vehicle unit is in range of the pit station. **Figure** shows an example of both driver-to-pit and pit-to-driver warnings on the driver display.
    - The “Clear Status” button will clear all of the status indicators, setting them back to their default states (orange) until a new transmission is received
    - The “Data Log” button serves two purposes. Its color indicates the status of the data logging portion. Red: Error, Green: Running, Orange: Paused. The user can also click the button to start/stop data logging.
        - It is imperative that the logging be stopped completely before the system is powered down to avoid corrupting the SD card.
    - The status will clear automatically after 2 seconds if no transmission is received
    - The application can be exited by clicking the “Quit” button.


TFT color monitor
Vehicle Status
Sensors
Clear Status
DAQ Configuration
Speed
Accel
Fuel Timer
58%
Reset Fuel Timer
Toggle Warning
Pit Warning
Driver Warning
Quit
V1/V2
MENU
-
+


240x320 2.2" TFT w/ILI9340C
by adafruit!
P
D
BL
MISO
CS
RST

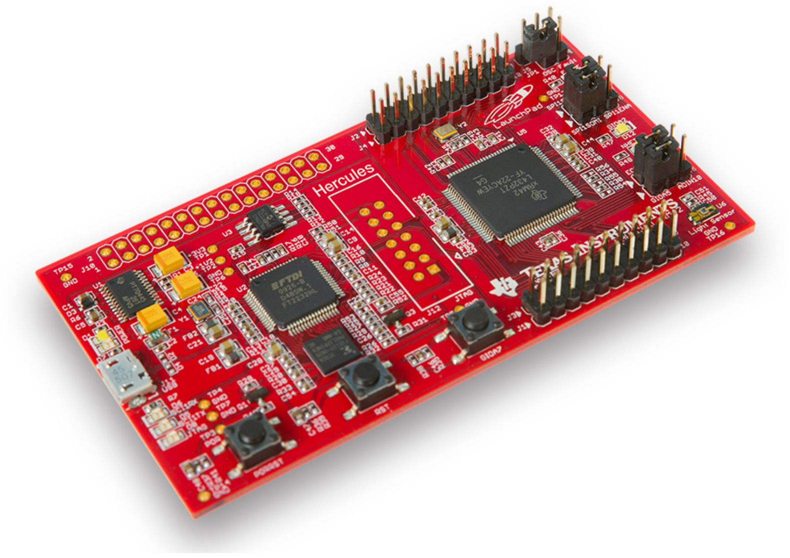
Maintenance

To properly maintain the DAQ system:

- Make sure the batteries are fully charged before each use
- Be sure data logging is paused before shutting down the DAQ system, and do not remove the SD card while the system is running
- Turn off the DAQ system after each use (using the switch inside the XBee enclosure).
- Clean the enclosures after each off road drive
- Wipe off excess dirt from the hall effect sensor as needed
- Store pit crew station items in a dry room temperature area when not in use

Major Component Specifications

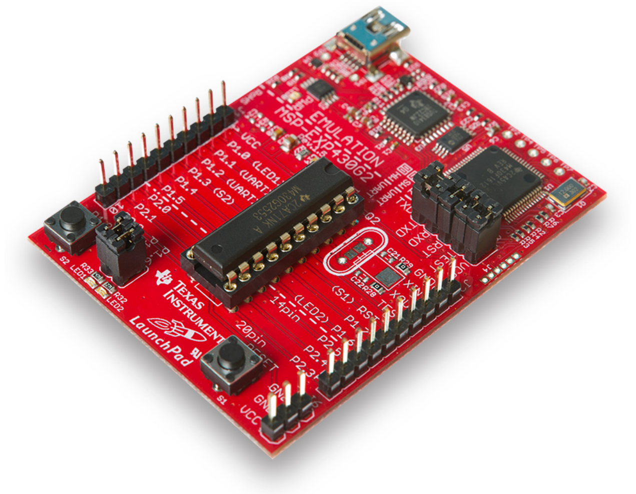
Hercules LAUNCHXL-TMS57004



Clock Frequency (MHz)	80
ADC	12-bit (16ch)
MibSPI	1
SPI	2
SCI/LIN	1
GPIO	45 (8 dedicated)
Operating Temperature Range (C)	-40 to 125

Core Supply (Volts)	1.2
IO Supply (V)	3.3
CPU	ARM-Cortex-R4

MSP430G2553 MCU



Frequency (MHz)	16
GPIO	24
I2C	1
SPI	1
UART	1
ADC	ADC10 - 8ch
Timers - 16-bit	2
Min VCC	1.8
Max VCC	3.6
Operating Temperature Range (C)	-40 to 85

XBee-PRO 900HP

Processor	ADF7023 transceiver, Cortex-M3 EFM32G230 @ 28 MHz
Frequency Band	902 to 928 MHz, software selectable channel mask for interference immunity
RF Data Rate	10 Kbps
Indoor/Urban Range	10 Kbps: up to 2000 ft (610 m); 200 Kbps: up to 1000 ft (305 m)
Outdoor/ Line-Of-Sight Range	10 Kbps: up to 9 miles (14 km); 200 Kbps: up to 4 miles (6.5 km) (w/ 2.1 dB dipole antennas)
Data Interface	UART (3V), SPI
GPIO	Up to 15 Digital I/O, 4 10-bit ADC inputs, 2 PWM outputs
Supply Voltage	2.1 to 3.6 VDC

DRV5053 Analog-Bipolar Hall Effect Sensor

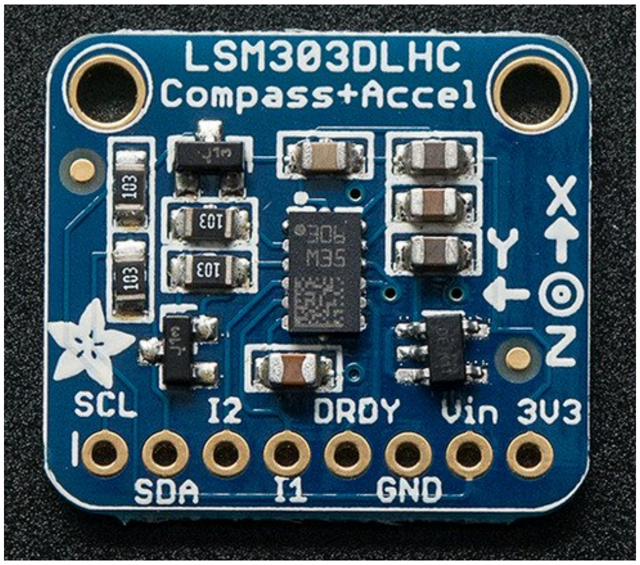
Supply Voltage (Vcc) (Min) (V)	2.5
Supply Voltage (Vcc) (Max) (V)	38
Output	0.2 V to 1.8 V
Type	Analog Bipolar
Operating Temperature Range (C)	-40 to 125

TL2575 Switching Voltage Regulators

Vin (Min) (V)	4.75
Vin (Max) (V)	40
Vout (Min) (V)	5

Vout (Max) (V)	5
Iout (Max) (A)	1
Operating Temperature Range (C)	-40 to 125

LSM303 accelerometer



Linear Acceleration	±2g/±4g/±8g/±16g
Data Output	16-bit
Interface	I^2C
Supply Voltage	2.16 V to 3.6 V

Appendix B

Test Forms

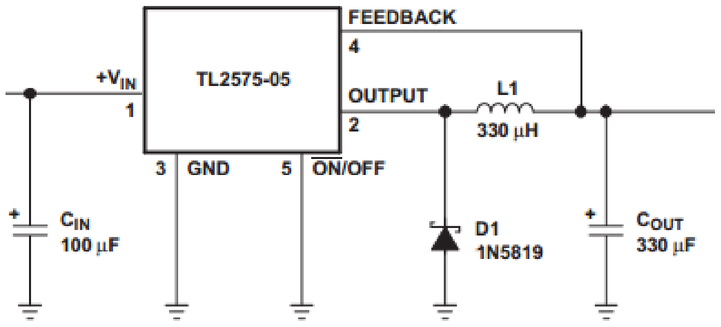
Scheduled Test Reporting Form

Test Item:	3.3V Switching Regulator Circuit	
Tester Name:	Dewey	
Test Date:	2/1/2015	Test ID No: 13
Test Time:	8 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Failed

Test Objective:
The objective of this test was to determine whether or not the 3.3V switching regulators supplied the correct voltage with a minimum amount of ripple.

Test Description/Requirements:
Requirements:
1: One TL2575 3.3V switching regulator
2: Oscilloscope
3: Power source
4: One 100uF capacitor
5: One 330uF capacitor
6: One 150uH inductor
7: One 1N5819 Schottky diode

Process:
The TL2575 switching regulator was hooked up into the circuit as shown in **figure** (5V regulator shown, same circuit). A power supply was used to supply the regulator with ~6-7V of input voltage. An oscilloscope was then connected across C_{out} and the output voltage and ripple were observed.



Anticipated Results:
The regulator will regulate the input voltage to 3.3 volts with a ripple of 50mV - 150mV.

Requirement for Success:
The test was a success if the regulator sufficiently regulated the input voltage to 3.3V, and minimized the amount of ripple. The data sheet for the IC specifies an achievable ripple of 50mV to 150mV.

Actual Results:
The voltage regulator sufficiently regulated the voltage input voltage to 3.3V. However, there was a very large amount of ripple ($800mV_{PP}$ - $1V_{PP}$).

Reason for Failure:
Insufficient output filtering, improper testing environment

Recommended Fix:
Try to add the second-stage ripple filter, measure voltage and ripple again.

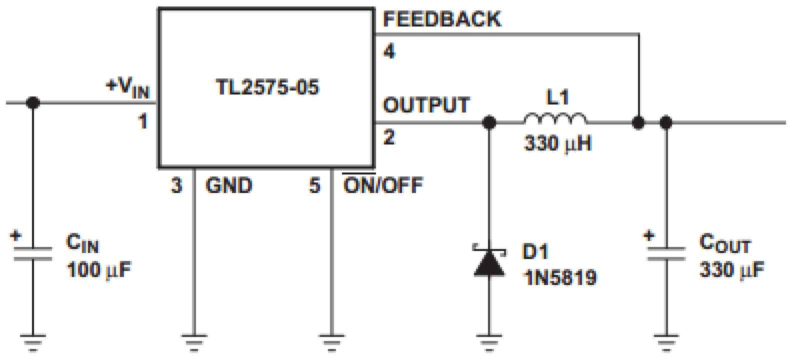
Scheduled Test Reporting Form

Test Item:	5V Switching Regulator Circuit	
Tester Name:	Hebe Perez	
Test Date:	3/05/2015	Test No: 15
Test Time:	5 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test was to determine whether or not the 5V switching regulators supplied the correct voltage with a minimum amount of ripple.

Test Description/Requirements:
Requirements:
1: One TL2575 5V switching regulator
2: Oscilloscope
3: Power source
4: One 100uF capacitor
5: One 330uF capacitor
6: One 150uH inductor
7: One 1N5819 Schottky diode

Process:
The TL2575 switching regulator will be hooked up into the circuit as shown in **figure** (5V regulator shown, same circuit). A power supply will be used to supply the regulator with ~6-7V of input voltage. An oscilloscope will then be connected across C_{out} and the output voltage and ripple will be observed.



Anticipated Results:
The regulator will regulate the input voltage to 5V with a ripple of 50mV - 150mV.

Requirement for Success:
This test was considered a success if the regulators displayed the rated 50mV – 150mV of ripple.

Actual Results:
The regulators circuits were connected as outlined in the datasheet. When analyzed with an oscilloscope, the regulators stayed within their 50mV – 150mV of ripple rating.

Reason for Failure:
N/A

Recommended Fix:
N/A

Scheduled Test Reporting Form

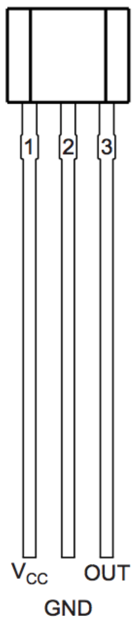
Test Item:	Hall Effect Sensor Basic Operation		
Tester Name:	Christopher Riker		
Test Date:	2/15/2015	Test No:	4
Test Time:	8 PM	Test Type:	Component Test
Test Location:	CoE Senior Design Lab	Test Result:	Pass

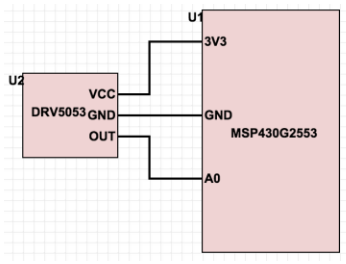
Test Objective:
The objective of this test is to verify that the hall effect sensor is able to detect the presence of a magnet as it passes by the sensor. This is to ensure that the sensor is operational and will be able to be used to detect a magnet passing by that is attached to a wheel or axle.

Test Description/Requirements:

Requirements:
1: One DRV5053 analog bipolar Hall Effect sensor
2: One neodymium magnet
3: One MSP-EXP430G2 LaunchPad development board

Process:
The MSP430 ADC10 module will be configured for use. The Hall Effect sensor will be connected to the LaunchPad on a breadboard. The VCC pin will be connected to the 3.3V output of the MSP-EXP430G2, and the GND pin will be connected to the ground. The OUT pin will be connected to the analog input port A0 (pin 1.0). Some code will be written to do A/D conversion on the Hall Effect sensor output continually. With the debugger running, a neodymium magnet will be passed in front of the Hall Effect sensor. The ACD output will be observed to determine whether or not it rises when the magnet is in front of the sensor.





Anticipated Results:
The voltage output of the Hall Effect sensor will rise as the magnet moves close to the sensor. The voltage output will peak when the magnet is directly in front of the sensor.

Requirement for Success:
The test will be successful if the voltage output of the sensor peaks when the magnet is directly in front of the sensor, telling the tester(s) that the Hall Effect sensor is properly detecting a magnetic field.

Actual Results:
The voltage output of the sensors varied with the distance of the magnet to the sensor. When the magnet was directly in front of the sensor, the output voltage was at 2 V, the sensor's maximum output. When the magnet was far away from the sensor, the output voltage was at 1 V, as specified in the data sheet.

Reason for Failure:
N/A

Recommended Fix:
N/A

Other Comments:

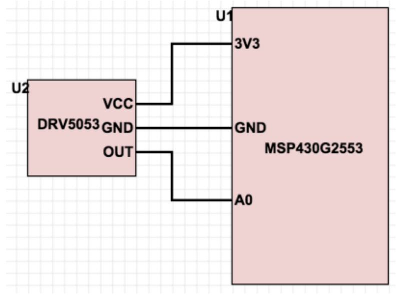
Scheduled Test Reporting Form

Test Item:	Hall Effect Sensor in Project Box	
Tester Name:	Christopher Riker	Tester ID No: 5174
Test Date:	03/16/2015	Test No: 2.5
Test Time:	7 PM	Test Type: Test
Test Location:	CoE Senior Design Lab	Test Result: TBD

Test Objective:
The objective of this test is to determine whether or not the hall effect sensor will be able to detect pulses from a passing magnet when it is inside the waterproof project box.

Test Description/Requirements:
Requirements:
1: Waterproof project box
2: One DRV50530 Bipolar Analog Hall Effect Sensor
3: One neodymium magnet
4: One MSP-EXP430G2 LaunchPad

Process:
The MSP430 ADC10 will be configured for use, and the hall effect sensor will be connected to the LaunchPad appropriately, as done in previous tests (see **Figure 1** below). The sensor will be placed on a breadboard behind the lid of the project box. A neodymium magnet will then be passed in front of the lid past the sensor while the MSP430 ACD10 is continually reading and converting the voltage output of the hall effect sensor. The digital output from the ADC10 will be observed to determine whether or not the hall effect sensor will be able to detect a passing magnet through the lid or walls of the small project boxes.



Anticipated Results:
Since the neodymium magnets are very strong, the testers expect that the hall effect sensor will be able to detect the change in the magnetic field as the magnet passes in front of the lid of the project box.

Requirement for Success:
The test will be considered a success if the hall effect sensor can detect the pulses from the passing magnet effectively enough to be used for speed measurement (e.g. in high-frequency applications).

Actual Results:
The sensor's output was varied by the magnet, just as it was without the project box there. When the magnet was directly in front of the sensor through the box, the output of the sensor was 2 V. When the magnet was not near the sensor, the output was 1 V, as per the data sheet. It should be noted that the output dropped to 1 V sooner than when the project box was not in between the magnet and the sensor.

Reason for Failure:
TBD

Recommended Fix:
TBD

Scheduled Test Reporting Form

Test Item:	Hercules + ILI9340 Display Test		
Tester Name:	Dewey Williams		
Test Date:	1/14/2015	Test No:	11
Test Time:	1 PM	Test Type:	Test
Test Location:	CoE Senior Design Lab	Test Result:	Pass

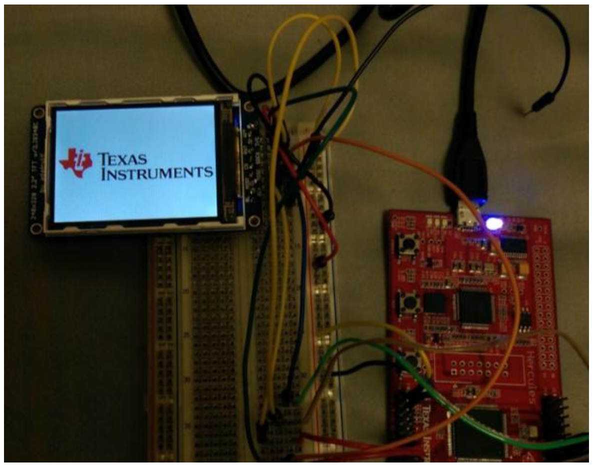
Test Objective:
The objective of this test is to develop code for the Hercules that will successfully initialize and write pixel data to a ILI9340-driven LCD display.

Test Description/Requirements:
Requirements:
TI Hercules TMS570LS04x with mibSPI capabilities
ILI9340-driven display (We used a 2.2" TFT LCD Display from Adafruit)

Process:
Following the ILI9340 datasheet and example code from Adafruit (written for the ATMega328), develop code to properly reset and initialize the display. Use the SPI1/mibSPI interface to attach to the data and clock lines, and two GPIO pins to connect to the D/C and RST lines on the display (additionally, the backlight can be PWM driven if desired). Configure the SPI interface and transfer groups as desired using HalCoGen (at least one transfer group must be a single 8 bit word as in traditional SPI for sending commands), and verify that the display turns on in an initialized state. Use the datasheet and example code to insert pixel data into the display buffer for output on the display.

Anticipated Results:
The display will be properly reset and initialized on startup of the Hercules, and pixel data will be sent and displayed on the screen.

Requirement for Success:
In order to be a success, the driver code should be able to reset and initialize the display, send commands to manipulate screen output, and send arbitrary pixel data to be displayed.



Actual Results:
The Adafruit example code was partially ported to be used on the Hercules by simply replacing AVR-specific function calls with those provided by HalCoGen. To verify operation, simple lines were drawn and the "rotate" command was sent to the display. Additionally, a full resolution bitmap was uploaded to the Hercules and drawn on the display using these techniques. The test was a success.

Reason for Failure:
N/A

Recommended Fix:
N/A

Other Comments:
The developed code was further optimized to utilize the mibSPI more effectively by prebuffering pixel data into 24 16-bit buffers, which greatly increased the performance of drawing large shapes and bitmaps. Additionally, text writing capability was added for writing data or text to the display. Development of this code is ongoing.

Scheduled Test Reporting Form

Test Item:	Hercules SPI	
Tester Name:	Dewey Williams	
Test Date:	1/12/2015	Test No: 15
Test Time:	5 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to verify correct operation of the Hercules' SPI interfaces in master mode. Only the SPI1/mibSPI interface will be tested, as the pins necessary for the other interfaces are not available until the additional MCU pin header is populated.

Test Description/Requirements:
Requirements:

TI Hercules TMS570LS04x MCU
Oscilloscope
Known-working SPI slave device

Process:
Use HalCoGen to configure the SPI1 (legacy) interface for testing. Program the Hercules to send some dummy data over SPI, and view the SCK, SIMO, and CS lines with an oscilloscope and verify correct operation. Next, attach a slave device and attempt to communicate. Verify any received data using a debug tool. An oscilloscope may be used to view the SCK, SIMO, SOMI, and CS pins again to verify full duplex operation. Repeat these steps using the mibSPI interface.

Anticipated Results:
The Hercules will be able to send and receive data over SPI using both legacy and mibSPI modes.

Requirement for Success:
The test is a success if the Hercules is able to communicate with a slave device in both legacy and mibSPI modes.

Actual Results:
The Hercules was able to successfully communicate with our ILI9340-driven display using both legacy and mibSPI modes. An oscilloscope was used to examine the behavior of the mibSPI module while transmitting multiple bytes. Correct operation was again verified by connecting the Hercules to a MSP430 configured as an SPI slave and checking the data at both ends using a debug tool.

Reason for Failure:
N/A

Recommended Fix:
N/A

Other Comments:

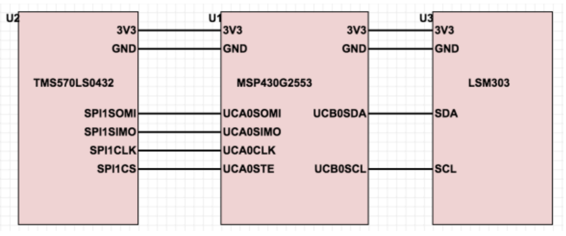
Scheduled Test Reporting Form

Test Item:	MSP430 I2C Master Mode w/ SPI Slave Mode	
Tester Name:	Christopher Riker& Dewey Williams	
Test Date:	3/01/2015	Test No: 19
Test Time:	7 PM – 9 PM	Test Type: Integration Test
Test Location:	College of Engineering Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to determine whether or not the MSP430G2553 can be used in I^{2}C master mode and SPI slave mode at the same time. This will be used for the accelerometer module, as the acceleration data must be collected from the digital accelerometer over I^{2}C and then sent to the Hercules LaunchPad over SPI.

Test Description/Requirements:
Requirements:
1: One TMS570LS04x (Hercules LaunchPad)
2: One MSP-EXP430G2 LaunchPad
3: One LSM303 Accelerometer/Magnetometer
4: Breadboard

Process:
The Hercules LaunchPad will be configured for SPI Master mode (already done). The MSP430 will be configured for SPI Slave mode (already done) using the USCIB0 module. The MSP430 will then be configured for I^{2}C master mode using the USCIA0 module. The LSM303 accelerometer will then be connected to the MSP430. A pin-level diagram of this connection can be found below. The MSP430 will be configured to continually poll the accelerometer for acceleration data, with the Hercules requesting SPI transmissions every so often. A variable on the MSP430 that stores the accelerometer data collected over I2C will be watched to ensure that the MSP430 is indeed collecting data from the accelerometer. The data transmitted over SPI will be stored in some variable, and that variable will be watched to determine whether not the MSP430 is successfully transmitting data to the Hercules over SPI.



Anticipated Results:
Since the MSP430 has two separate USCI modules, the design team does not expect any issues stemming from using both of them in tandem.

Requirement for Success:
The test will be considered a success if the MSP430 can collect data from the LSM303 accelerometer over I^{2}C while sending data to the Hercules over SPI.

Actual Results:
Once a multi-byte SPI framework was developed, the testers were able to get the MSP430 to send correct accelerometer data to the Hercules LaunchPad over SPI.

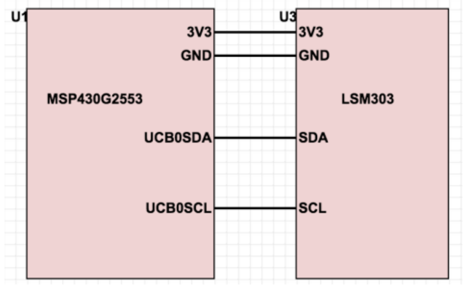
Scheduled Test Reporting Form

Test Item:	Accelerometer Calibration	
Tester Name:	Christopher Riker	
Test Date:	2/20/2015	Test No: 1
Test Time:	9 PM	Test Type: Component Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to determine whether or not the accelerometer outputs the correct values when a known, constant acceleration is placed on it.

Test Description/Requirements:
Requirements:
1: LSM303 Accelerometer/Magnetometer
2: MSP430G2553 Microcontroller
3: MSP-EX430G2 LaunchPad
4: MSP430G2553 Microcontroller
5: Breadboard

Process:
The LSM303 accelerometer will be connected to the MSP430 LaunchPad, which will be configured for I2C master mode. The accelerometer will be placed in a breadboard and hooked up using the pin diagram found in **figure**. The accelerometer will then be turned such that the gravitational acceleration from the earth is applied to each of its axes. The output of these axes will be examined to ensure that they are equivalent to 9.81 m/s^2.



Anticipated Results:
The LSM303 accelerometer will output 1G of acceleration for each of the axes when they are subjected to the gravitational acceleration from the earth.

Requirement for Success:
The test will be considered a success if the accelerometer accurately measures the acceleration due to the earth's gravitational force.

Actual Results:
The accelerometer's outputs are in the form of a 16-bit signed number. I.e., when 0 G's of gravitational acceleration are applied to an axis, the output is around 0 when the values are read into a signed integer. The tester discovered this when the output value was overflowing an unsigned integer when switching from a positive to a negative acceleration. Given a +/- 2G range, 1 G of acceleration should read right around 16,000 when the readings are placed into a signed integer. Each axis exhibited this behavior, outputting a value around 16,000 when 1 G was placed on the axis.

It should be noted that these values varied slightly, and each axis had a "characteristic" value that it output for 1 G of acceleration.

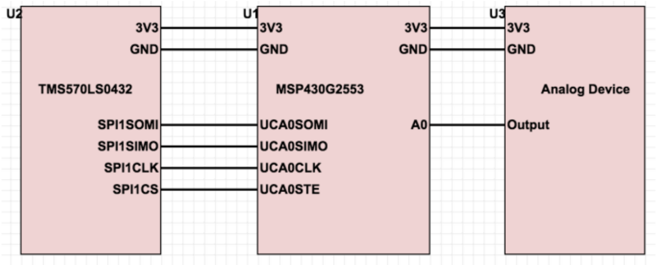
Scheduled Test Reporting Form

Test Item:	MSP430G2553 SPI slave mode w/ ADC10 conversion	
Tester Name:	Dewey Williams & Christopher Riker	
Test Date:	1/15/2015	Test No: 17
Test Time:	5 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Failure

Test Objective:
The objective of this test is to determine if the MSP430 microcontroller is capable of doing efficient A/D conversion while communicating with a master device over SPI.

Test Description/Requirements:
Requirements:

1: One MSP-EXP430G2 LaunchPad
2: One SPI master device
3: One potentiometer
4: Breadboard
5: Jumper wires



Process:
The TMS570LS04x LaunchPad will be configured in SPI master mode using HalCoGen. The MSP430's ADC10 module will be set up to continually convert on one channel (sequence mode 2). The potentiometer will be connected to VCC and Ground, with the wiper being connected to analog port A0. Code will be written for the TMS570LS04x that continually begins SPI transmissions with the MSP430 and stores the received message in some variable. Both chips will be programmed and the debuggers started. The tester(s) will observe the variable and how it changes when turning the potentiometer knob, noting how quickly the variable updates.

Anticipated Results:
The variable that stores the transmitted data should update in real-time with the turning of the potentiometer knob. Since the hardware USCI module and the ADC module are implemented on-chip separately, no troubles are anticipated by the testers.

Requirement for Success:
The test will be considered successful if the MSP430 is able to efficiently convert analog readings to digital signals, and the master device receives those signals in real-time.

Actual Results:
Initially, the test seemed to be a failure. However, upon checking wire connections, the testers discovered that one of the wires was connected to the wrong slot in the breadboard. When this was corrected, the test did not go as anticipated. The variable took 2-3 transmissions to update when turning the potentiometer wheel.

Reason for Failure:
Upon further consideration, the testers discovered that a line of code in the USCI0RX interrupt vector `__delay_cycles(10000);` may have been the culprit.

Recommended Fix:
The `__delay_cycles()` call is necessary, however 10,000 cycles may be overkill. The number should be reduced drastically, which may lead to a successful future test.

Other Comments:

Scheduled Test Reporting Form

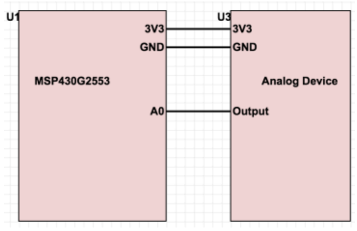
Test Item: MSP430 ADC10	
Tester Name: Christopher Riker	
Test Date: 1/25/15	Test No: 12
Test Time: 5 PM	Test Type: Integration Test
Test Location: CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to demonstrate the ADC functions of the MSP430G2553 and to generate baseline code to be used in sensor applications.

Test Description/Requirements:
Requirements:

MSP430G2553 microcontroller
Flash programmer and debugger (such as an MSP-EXP430G2 Launchpad)
Analog device (such as an analog hall effect or potentiometer)

Process:
Create a program using example code and other resources which will receive data from the ADC and store it to a variable at some constant interval. Use a debugging utility to view the contents of variables and registers to verify that the collected information is correct.



Anticipated Results:
Data stored in the variable will reflect the position of the potentiometer or the position of the accelerometer.

Requirement for Success:
In order for this test to be a success, the MCU must be able to activate and retrieve data from the ADC, and store that data for later use. The ADC should be able to capture the full range of the attached analog device with a reasonable degree of accuracy.

Actual Results:
The MSP430 was connected to a potentiometer on ADC channel 0 (pin 1.0), and was programmed to continually activate and retrieve data from the ADC. The potentiometer was moved to various positions and the debugger was used to verify the data gathered from the ADC. For each position, the captured data was accurate within a reasonable margin, so the test is considered a success.

Reason for Failure:
N/A

Recommended Fix:
N/A

Other Comments:
Code samples for this test are available in the full report. This code will be assembled into an easy to use driver library to and included in the software for each analog sensor MCU.

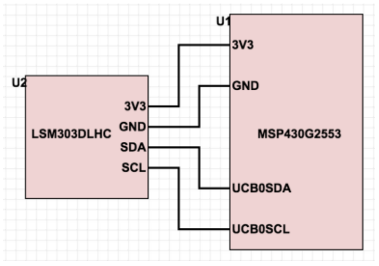
Scheduled Test Reporting Form

Test Item:	MSP430G2553 I^2C (Master mode)	
Tester Name:	Christopher Riker	
Test Date:	2/20/2015	Test No: 18
Test Time:	7-9 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to determine whether or not the testers can get the MSP430G2553 microcontroller configured and working in I^2C master mode. This will be used to communicate with the digital accelerometer/magnetometer.

Test Description/Requirements:
Requirements:
1: MSP-EXP430G2 LaunchPad (with an MSP430G2553)
2: LSM303 Accelerometer/Magnetometer
3: Breadboard

Process:
The MSP430G2553 will first be configured as an I2C master device (using Grace or similar). The LSM303 will then be connected to the proper pins of the MSP430 (pin diagram found in **figure**). Code will be written that polls the acceleration values from the LSM303 continually. The variable used to store the accelerometer output will be observed to determine whether or not the MSP430 is receiving communication from the accelerometer.



Anticipated Results:
The MSP430 will be configured correctly such that is able to communicate with the LSM303 and receive accurate readings from it.

Requirement for Success:
The test will be considered a success if the tester is able to get the MSP430 communicating with the LSM303 over I2C.

Actual Results:
The tester was unable to get the hardware USCI module to work in I^2C mode. Initially, the module was clocking and sending bytes of data to the sensor. However, after a break in testing/development, the USCI module was no longer clocking or sending bytes of data. The code remained the same between these two tests. The tester tried to write entirely new code in order to get the USCI module to work, however this was unsuccessful. As such, software I2C was explored (i.e. “bit banging”). Once an existing I2C library was ported for use on the MSP430 in Code Composer Studio, and a library for the LSM303 was written, the tester was able to get the MSP430 to successfully communicate with the LSM303 accelerometer/magnetometer.

Scheduled Test Reporting Form

Test Item:	Sensor Board Testing	
Tester Name:	Dewey Williams & Christopher Riker	
Test Date:	3/15/2015	Test No: 22
Test Time:	6 PM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to verify the correct working condition of circuit boards designed and milled at the College of Engineering.

Test Description/Requirements:
Requirements:

Milled and populated sensor board
Programmed MSP430G2553 MCU

Process:
Program the MSP430 to toggle each pin in turn (for all pins broken out on the circuit board), and insert into the socket. Apply 3.3v to the designated pin and connect the ground pin to ground. Test each pin on the headers with an oscilloscope or multimeter to verify that the pin is properly connected to the MCU.

Anticipated Results:
The milled circuit board will function as designed, and the MCU will have proper control over each pin.

Requirement for Success:
In order to be a success, the MCU must be able to toggle each header pin on the board. The MCU must also be in a stable state (that is, no strange resets or other odd behavior) to ensure that the chip and board can be safely embedded.

Actual Results:
The MSP430 was inserted into the populated PCB and functioned as expected. The MCU had control over each pin, indicating that each pin that was broken out was properly connected to the MCU.

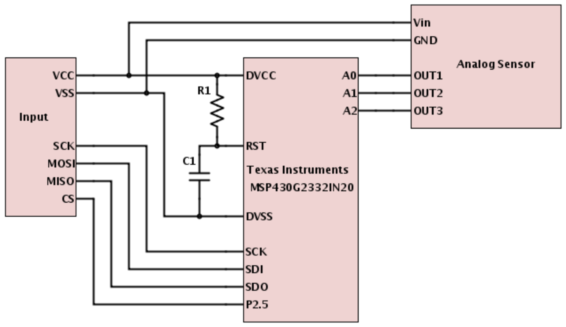
Scheduled Test Reporting Form

Test Item:	MSP430 SPI Slave Mode		
Tester Name:	Christopher Riker		
Test Date:	1/21/15	Test No:	16
Test Time:	5 PM	Test Type:	Integration Test
Test Location:	CoE Senior Design Lab	Test Result:	Pass

Test Objective:
The objective of this test is to demonstrate the SPI functions of the MSP430's USCI and to generate baseline code to be used in SPI slave sensor applications.

Test Description/Requirements:
Requirements:

MSP430G2553 microcontroller
Flash programmer and debugger (such as an MSP-EXP430G2 Launchpad)
Known-working SPI master device (such as another microcontroller)



Process:
Create a program using example code and other resources which will send SPI data (such as an increasing counter or some constant value) when data is sent by a master device. Connect the SPI and ground pins of each device and view the contents of internal registers and memory using a debugging utility to verify the correct transmission of data. Two similarly programmed MCUs may be connected to different chip selects on the master in order to verify that inactive devices will not interfere with other transmissions on the bus, or that these other transmissions will not affect the inactive slaves. An example pin-level layout can be found below.

Anticipated Results:
Data sent by the slave will be correctly received at the master, and vice versa.

Requirement for Success:
In order for this test to be a success, the slave device must successfully transmit the most current internal data when polled by the master, and it should also successfully receive data from the master, in order to verify full duplex operation. The slave devices must also be completely inactive on the bus unless their chip select (or Slave Talk Enable in the case of the MSP430s) is pulled active in order to prevent interference.

Actual Results:
The MSP430s were programmed to send an incrementing count when polled by the master device (the TI Hercules TMS570LS0432) using UCB0 (pins 1.4-1.7). Each slave was programmed with a different starting count so that the results could be properly identified. After some development and debugging, data was correctly sent and received between the master and selected slave devices, and no interference was detected so this test is considered a success.

Other Comments:
Code samples for this test are available in the full report. This code will be assembled into an easy to use driver library to and included in the software for each sensor MCU.

Scheduled Test Reporting Form

Test Item:	Plano 147000 Waterproof Case	
Tester Name:	Christopher Riker	
Test Date:	3/20/2015	Test No: 21
Test Time:	4 PM	Test Type: Integration Test
Test Location:	College of Engineering	Test Result: Pass

Test Objective:
The objective of this test is to test whether or not the Plano 147000 waterproof seal will hold up to the environment in which it will be used in.

Test Description/Requirements:
Requirements:

1: Plano 147000 Waterproof Case
2: Water source
3: Paper towels

Process:
The case will be lined with paper towels. This will make it easy to tell whether or not water penetrated the box. The case will then be submerged just under the surface of some water source (location TBD). The box will then be removed, and the paper towels will be examined to determine whether or not water penetrated the box. This test will be sufficient in determining whether or not the box will protect the components inside of it from the elements.

Anticipated Results:
The moisture will be successfully kept outside of the box (as advertised), and the paper towels inside will remain dry.

Requirement for Success:
The test will be considered a success if the box keeps any and all moisture out.

Actual Results:
As expected, the Plano box kept moisture out, indicating that the components will be kept safe from the elements.

Other Comments:
It should be noted that the box was tested without the “waterproof” grommets installed. Installing these grommets could potentially introduce some unreliability into the protection that the box provides.

Scheduled Test Reporting Form

Test Item:	Project Box Waterproof Seal Test	
Tester Name:	Christopher Riker	Tester ID No: 5174
Test Date:	3/05/2015	Test No: 2.5
Test Time:	4 PM	Test Type: Test
Test Location:	College of Engineering	Test Result: Pass

Test Objective:
The objective of this test is to determine whether or not the waterproof project boxes obtained for each of the discrete components are actually waterproof.

Test Description/Requirements:
Requirements:
1: Project boxes with seals cut to the correct length
2: Paper towels
3: Water source

Process:
The seals of the small project boxes first need to be cut to the correct length. Once that is done, the boxes will be filled with paper towels, and the tops will be screwed securely on. The boxes, one by one, will be submerged just under the surface of the water source (location TBD). The boxes will then be opened up and the paper towels inside examined to determine whether or not water got into the boxes.

Anticipated Results:
The project boxes will successfully keep water out and the paper towels inside will remain completely dry.

Requirement for Success:
The test will be considered a success if the boxes keep the water out and the paper towels inside remain completely dry.

Actual Results:
The un-modified boxes successfully kept the moisture out.

Other Comments:
As with the Plano box, introducing the grommets to the box could potentially reduce the reliability of the protection provided by the project boxes.

Scheduled Test Reporting Form

Test Item:	Full System Test	
Tester Name:	Dewey Williams & Christopher Riker	
Test Date:	4/4/2015	Test No: 2.5
Test Time:	3:30 AM	Test Type: Integration Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to connect the entire system together and run it under battery power, with all subsystems operation and working.

Test Description/Requirements:
Requirements:
1 – Assembled & Running Pit unit
1 – Assembled and running on-vehicle unit

Process:
After the completion of the software development, the entire system is to be hooked up and run under battery power. The sensor MCUs will be inserted into the MCU PCBs, and the components will draw power from the 3.3V and 5V regulator circuits (on their PCB). This test is essentially to ensure that the integration of each of the discrete subsystems works smoothly.

Anticipated Results:
The system will run fully operational (display working and displaying data, SD card logging, etc.) on battery power and transmit actual sensor data to the pit unit.

Requirement for Success:
In order for this test to be successful, the system should run on battery power and send actual sensor data to the pit unit.

Actual Results:
The system ran off of the LiFePO4 batteries and transmitted actual sensor data to the pit unit while collecting, storing, and displaying actual sensor data.

Scheduled Test Reporting Form

Test Item:	Xbee - Raspberry Pi Communication	
Tester Name:	Dewey Williams	
Test Date:	1/23/2015	Test No: 9
Test Time:	8 PM	Test Type: Component Test
Test Location:	CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test is to connect an XBee module to the Raspberry Pi so that transmissions may be read by the operating system.

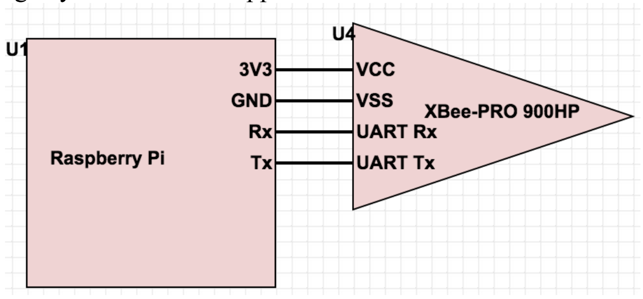
Test Description/Requirements:
Requirements:

XBee Module
Raspberry Pi running Raspbian
XBee Explorer
A host to send test messages

Process:

First, attach an XBee module to a working host (such as a PC or a microcontroller). Next, connect another XBee to the XBee Explorer and connect it to the Raspberry Pi via USB. Use a serial monitor application to attach to the Explorer's port (by default, the explorer will appear as "/dev/ttyUSB0"), and verify that test transmissions are properly sent and received.

Another method is to use the Pi's UART to connect to the XBee. Use of the UART by applications requires disabling some default configurations in Raspbian (more information on this can be found online). The UART is located by default at /dev/ttyAMA0. By connecting the XBee to the Tx and Rx pins on the Pi's expansion header, the XBee can be reached using any serial monitor application.



Anticipated Results:

The XBee will connect to the Raspberry Pi and respond to commands and transmitted data.

Requirement for Success:

The test is a success if the Raspberry Pi is able to send and receive messages via the XBee at the OS level. This is required in order to later develop the GUI application to display data from the vehicle.

Actual Results:

The Pi was able to receive messages from another XBee attached to a PC running X-CTU. The USB connection was tested using a serial monitor application, and UART was tested using a short C program based on example code found online. The Pi was also able to send commands to the XBee (such as +++ for AT mode), as well as send messages to the other host.

Scheduled Test Reporting Form

Test Item: XBee Baseline Wireless Transmission Test	
Tester Name: Dewey Williams & Christopher Riker	
Test Date: December 10th, 2014	Test No: 6
Test Time: 4 PM	Test Type: Component Test
Test Location: CoE Senior Design Lab	Test Result: Pass

Test Objective:
The objective of this test was to determine whether or not the XBees arrived in an operational condition, and were able to send/receive transmissions.

Test Description/Requirements:
Requirements:
1: Two XBee-PRO 900HP wireless transceivers
2: Two computers with X-CTU installed
3: Two XBee USB Explorers

Process:
The XBee-PRO modules were connected to the computers via the USB explorers. The testers first verified that the XBees were responding to commands by sending the command +++, which puts the XBee in AT mode. A unique network identifier was then configured to prevent interference with other devices in the area. The XBees were then connected to the network using the connect button. The console window of one XBee was used to send packets from one XBee to the other, and vice versa. The testers confirmed that the packets were received and were transmitted without error.

Anticipated Results:
The XBee modules will return "OK" when placed in AT mode. After that, the XBee modules will send and receive packets to one another without error or packet loss.

Requirement for Success:
The test will be a success if the XBee modules both return "OK" when placed in AT mode, and if they both send and receive packets to one another without error or packet loss. Upon success, the XBee modules will be declared operational and no further action will be taken.

Actual Results:
The XBee modules both returned "OK" when plated into AT mode, and the devices were able to exchange packets without error in the transmissions.

Reason for Failure:
Test passed

Recommended Fix:
N/A

Other Comments:

Scheduled Test Reporting Form

Test Item:	XBee communication between Hercules LaunchPad and Raspberry Pi	
Tester Name:	Dewey Williams	
Test Date:	1/30/2015	Test No: 23
Test Time:		Test Type: Integration Test
Test Location:	College of Engineering	Test Result: N/A

Test Objective:
The objective is to test is to determine whether or not the code developed for communication between the XBee modules connected to the Hercules and the Raspberry Pi operates as intended.

Test Description/Requirements:
Requirements:

1: Two XBee-PRO 900HP Module
2: Raspberry Pi running Raspbian
3: TI TMS570LS04x LaunchPad

Process:
Code will be developed for the Hercules LaunchPad and the Raspberry Pi that utilizes UART to send messages back and forth between the two devices via the XBee modules. The Hercules will generate some kind of arbitrary data (e.g. an incrementing variable placed into a string) and send that data to the Raspberry Pi. The Raspberry Pi will then receive that data and echo it to the console. The data will be checked for accuracy and completeness.

The XBee module has already been successfully tested to work with the Raspberry Pi's UART interface (using X-CTU on a laptop to send messages to the Raspberry Pi). The purpose of this test is to ensure that the design team can get the Hercules to do the same.

Anticipated Results:
The team anticipates that the Hercules and Raspberry Pi will be able to effectively communicate (i.e. without erroneous transmissions) with one another with the XBee modules using their respective UART interfaces.

Requirement for Success:
The test will be considered successful if the Hercules and Raspberry Pi are able to communicate with each other via the XBee modules over their respective UART interfaces without erroneous transmissions.

Actual Results:
The Hercules and the Raspberry Pi were able to successfully communicate with each other via the XBee modules.

Scheduled Test Reporting Form

Test Item:	Vehicle-to-Pit Communication	
Tester Name:	Christopher Riker, Dewey Williams, & Hebe Perez	
Test Date:	4/7/2015	Test No: 24
Test Time:	4 PM	Test Type: Integration Test
Test Location:	College of Engineering	Test Result: Pass

Test Objective:
The objective is to test the functionality of the wireless communication module as part of the data acquisition system when the system was installed on the vehicle.

Test Description/Requirements:
Requirements:

1- XBee Module
2- Raspberry Pi running Raspbian
3- TI Hercules TMS570LS04x

Process:
This test was completed when the full system was installed on the vehicle. The system was turned on, and the pit unit was connected. The full system, pit unit and vehicle unit, had already been tested in a lab setting. This test was completed mostly as a validation of a correct install, and to ensure that nothing was broken during installation.

Anticipated Results:
Thorough testing of the components before vehicle installation gives confidence to successful wireless communication after system integration.

Requirement for Success:
In order for this test to be a success, the system should function exactly as it did before it was installed on the vehicle. This means that the driver-to-pit and pit-to-driver warning lights functioned correctly, all of the status indicators on the pit unit functioned correctly, and sensor data sent to the pit unit matched that which was displayed on the on-vehicle display.

Actual Results:
The system functioned as expected. All of the indicator/status lights worked, and the sensor data displayed by the Raspberry Pi matched that which was displayed on the on-vehicle display.

Scheduled Test Reporting Form

Test Item:	XBee Short Range Test	
Tester Name:	Christopher Riker & Dewey Williams	
Test Date:	12/5/2014	Test No: 7
Test Time:	4 PM	Test Type: Component Test
Test Location:	Innovation Park	Test Result: Pass

Test Objective:
The objective of this test was to test whether or not the XBee-PRO 900HP modules would send and receive errorless packets at a distance of 0.25 miles (non-LOS). This was a preliminary test for a longer range test.

Test Description/Requirements:
Requirements:

1: Two XBee-PRO 900HP wireless transceivers
2: Two XBee USB explorers
3: Computers with X-CTU installed

Process:
Since the XBee modules were already configured with a unique network identifier, they required no setup other than to be connected to the tester's computers. The testers found 0.25 miles of space in the Innovation Park area. The space found did not allow the testers to get a line of sight between the XBee modules. Once the testers were in their respective spots, X-CTU was used to send packets from one XBee module to the other. Both long and short messages were sent to test whether or not large packets of information could be transmitted over this range. After testing transmission, the testers compared their X-CTU consoles to ensure that the data was transmitted without error. The range of the testing area was measured with an automobile odometer to be 0.25 miles.

Anticipated Results:
The XBee modules were expected to be able to send and receive packets at this range with little to no error.

Requirement for Success:
The test was considered a success if the testers were able to send packets to each other through X-CTU and those packets were received without packet loss or error.

Actual Results:
The XBee modules were able to send and receive packets to and from one another without error.

Reason for Failure:
TBA

Recommended Fix:
TBA

Other Comments:
There was one small anomaly in one of the XBee module's console windows that was not associated with any message that was sent by the other. It is unknown what the cause of this anomaly was.

Scheduled Test Reporting Form

Test Item:	Hercules SD Card R/W	
Tester Name:	Dewey Williams	
Test Date:	02/21/2015	Test No: 10
Test Time:	9pm	Test Type: Component Test
Test Location:	College of Engineering Lab	Test Result: Success

Test Objective:
This test will establish the capability of the Hercules MCU to read and write data to/from a microSD card. It will require the development of new code based on an existing file system framework.

Test Description/Requirements:
Requirements:
TI Hercules TMS570LS04x MCU
microSD card and slot

Process:
Connect the pins of the microSD card to the SPI1/mibSPI interface of the Hercules. Using FATFS to handle high level file operations, add the necessary low level driver code to send commands and data to the SD card. Verify that data can be read effectively by using a debug tool or another data output medium to display the collected data. Verify that data can be written effectively by performing a write operation using the Hercules and checking the data using a PC or another known-good card reading device. Circular reads and writes using the Hercules alone may also be tested to check the integrity of arbitrary data written to the card.

Anticipated Results:
Pending software and driver code development, SD functionality should be achieved using the above method.

Requirement for Success:
In order for this test to be a success, the final code should be able to read files created on a PC, and should be able to write files to be read with a PC using a FAT file system. The files should be of arbitrary length and content as this code may be used in different DAQ system operations in the future.

Actual Results:
After consulting example code and following through with the development steps described above, SDHC read and write capability was ported to the Hercules. This test was used as a baseline for further development in the DAQ's data logging subsystem, and was also used to read graphics from the card to draw on the driver display.

Reason for Failure:
N/A

Recommended Fix:
N/A

Other Comments:

Appendix C

Software Guide

Appendix C – Software

All of the code developed for this project can be found in the GitHub, linked elsewhere in this document as well as on the team webpage.

Hercules Main Routine

sys_main.c contains the main routine for the Hercules MCU. The necessary TI-provided drivers can be generated from the included *.hcg and *.dil files by opening them in HALCoGen. The main routine also requires the ILI9340 display library and SDHC library.

The Hercules is responsible for collecting sensor data and sending it to its destination - that is, the driver display, the pit unit, and the SD card - as well as monitoring the status of the subsystems and potentially handling any errors that may arise.

A bulk of the routine's responsibilities are handled in the main loop. There, the status of the sensor network is determined by polling the accelerometer (since the accelerometer's data collection routine is the most complex, it is seen as the main point of failure for that subsystem). Then it collects speed data and filters those samples to calculate the min, max, and average values which will be written to the log if data logging is enabled at the pit unit.

Because the accelerometer sampling is slower than other operations, its collection is split into two pieces - the request for a sample, and the actual receipt of data from the module. This way, the time spent waiting for a new sample can be used to draw on the display, log data, etc. and not in empty wait loops.

Data transmission to the pit is handled by an RTI (real time interrupt), configured to trigger every half second. This interrupt also handles other timed functions such as the fuel timer and triggering writes to the SD card.

Transmissions received from the pit unit are handled in an SCI receive interrupt which first checks for a particular series of bytes (the preamble), and then accepts two bytes signifying a command. A few commands are supported such as setting a notification, toggling the data log, and resetting the fuel timer. These commands take the form of simple two-character ascii strings. For instance, "lo" is the command to toggle data logging.

Hercules ILI9340 Display Library

The ILI9340 display library was developed to interface the Hercules with the Adafruit display used on the vehicle. The library was partially ported from the provided Adafruit code, and some extra features were added as they were required. The library uses the Hercules' mibSPI module to quickly and efficiently draw simple shapes, text, and images to the display.

The library requires that a number of configurations be made in HALCoGen. The mibSPI module must be configured with certain transfer groups as described here:

- Single 8bit transfer at 7000 kHz for commands
- Single 16bit transfer at 16000 kHz for pixel data
- Multiple 16bit transfers (configurable length) at 16000 kHz for pixel data

The transfer group numbers, and the size of the multi-pixel transfer can be configured in *ili9340.h*. Note that for each data format, the "clock phase" box must be checked.

One version of the library uses the FatFS SDHC library to add raw 565 image drawing capability. The other version removes this feature for when SD card capability is not required or desired.

The driver incorporates a 9x5 console font which is stored in program space, allowing for operation without an SD card inserted.

Hercules SDHC Library

The Hercules SDHC library is a port of FatFS, a free and open source FAT file system library which provides all high level file system traversal and file read/write capability. Low level driver code had to be developed which uses the Hercules' mibSPI module to both initialize the card and send and receive 512 B blocks from the card. This driver code can be found in the file *diskio.c*. Currently, only SDHC cards are supported.

The SDHC library requires a few mibSPI transfer groups be set up in HALCoGen and configured in *diskio.c*:

- Single byte transfer at 7000 kHz
- 6 byte transfer at 7000 kHz for commands
- 64 byte transfer at 7000 kHz for large data blocks
- 10 bytes at 7000 kHz with no CS asserted (for initializing the card)

Note that for each data format, the "clock phase" box must be checked.

More info about FatFS can be found here:

http://elm-chan.org/fsw/ff/00index_e.html

MSP430 Accelerometer Module

Overview

This module, upon request from the Hercules, polls the LSM303 accelerometer for X- and Y-axis acceleration values (2-bytes each). Those values are then transmitted to the Hercules byte-by-byte via SPI, using a framework developed for multi-byte SPI transmissions between the Hercules and the MSP430. The framework developed is as follows:

2-byte preamble: 0xAE (always kept in UCB0TXBUF when data is not ready to be transmitted) followed by 0xBB (first element in the sample buffer, indicates to the Hercules that a transfer is beginning).
4-bytes of data: X-axis high byte, X-axis low byte, Y-axis high byte, Y-axis low byte.

All of this is controlled in the file *main.c*, which sets up the MSP430 clock to run at 16 MHz, initializes the USCI-B module in SPI mode, initializes the software I2C, and sends the proper bytes to the LSM303 to set it up in its default state (more info in the LSM303 data sheet).

Once a transfer is requested by the Hercules, a variable called *"transferRequested"* is set to 1, to indicate that a transfer has been requested. The MSP430 will continue to send out 0xAE over the SPI bus while the accelerometer is read. Once complete, *"accelReadComplete"* is set to 1 to indicate that the read is complete and the data is ready to be transmitted to the Hercules. The MSP430 then sends a 0xBB over the SPI bus, followed by 4-bytes of acceleration data.

Software I2C and LSM303 Operation

The MSP430 writes/reads data to/from the LSM303 accelerometer using a software I2C "bit banging" technique, whereby the GPIO pins (selectable by the programmer) of the MSP430 are manipulated through software, rather than through the built-in hardware module. The software I2C library is essentially a ported Arduino library , edited for use with the MSP430 (created by Rei VILO, edited by Christopher Riker). This was used when the hardware USCI module stopped working in I2C mode.

The LSM303 accelerometer is designed to be read in "burst mode." Essentially, the address of the register that is to be read first is written to the accelerometer, with the bit in position 7 set high, i.e. START_ADDRESS | (1 << 7). This sets the registers to auto-increment, and each successive transmission contains the byte in the next register. A library was written for the LSM303, and examples of the use of the functions created can be seen in the final LSM303 code found on the GitHub. Calling the function *ReadAccel* (reads all 3 axes), *ReadAccelXY* (only reads X and Y axis), and *ReadMag* (reads all 3 axes) will handle all of this, depending on what the programmer requires. The code, found on GitHub through the team's web page, is well commented, and any future teams should be able to understand it quite easily.

MSP430 Hall Effect Sensor Module

This module uses simple analog to digital conversion, along with a single-byte SPI transmission to transfer the final calculation from the MSP430 to the Hercules LaunchPad. Since the top speed of the vehicle is 37 MPH (as per the Baja team), the team decided to use an 8-bit, UQ6.2 fixed-point format for data transmission.

The maximum pulse rate was found to be ~10 Hz, giving a Nyquist rate of ~20 Hz for sampling of the Hall Effect sensor. Given such a slow sampling rate, the team decided to use floating point operations to simplify the programming of this module, while still maintaining the 20 Hz Nyquist rate. With the whole system running, the Hercules was able to sample the sensors at a rate of ~24 Hz, and the Hall Effect sensor module undoubtedly calculated speed values much faster than that.

In order to accomplish speed measurement, the MSP430 continually converts analog readings on analog channel A1 to digital values. Once a value has reached high enough, the MSP430 considers this to be the beginning of a pulse. Once the first rising edge has been detected, the timer is reset, and the MSP430 continually converts readings on analog channel A1 to digital values, and waits for a falling edge. Once again, the MSP430 converts analog readings to digital values, waiting for the next rising edge. Once the next rising edge has been found, the value contained in the timer is read and stored in the variable called *count*, and the speed is calculated using hard-coded #define constants and floating-point operations. Once the speed has been found, the fixed-point representation is constructed as follows:

1. The whole part of the value is subtracted off and stored in the upper 6-bits of *fixed_point_speed*.
2. If the leftover speed is >= 0.5, the first fractional bit is set high. If not, it is kept low.
3. 0.5 is subtracted off of the speed
4. If the leftover speed is >= 0.25, the second fractional bit is set high. If not, it is kept low.
5. *fixed_point_speed* is placed in the USCI transmit buffer for transfer to the Hercules over SPI.

A timer interrupt was enabled and an ISR was created to handle a time-out situation. If ~1 second passes, the timer interrupt is triggered, and the USCI transmit buffer is set to 0 and the pulse count is reset. This prevents the MSP430 from sending out garbage values when the vehicle is stopped or otherwise.

Raspberry Pi GUI

The Raspberry Pi GUI was created in C++ using Qt Creator, and compiled using the Qt utilities *uic* and *qmake*. The GUI implements two QTimer instances, one set for 0ms to check for incoming transmissions, and another at 2000ms to clear the status indicators if no transmission has been received. On a new transmission, the Pi checks for a 3 byte preamble, and then accepts a status byte and 3 data bytes. The Pi does not store or analyze any data, it simply collects it for display to the pit crew. The interface includes buttons to start or stop data logging, manually clearing the status indicators, and setting the pit crew notification to be displayed to the driver.

Appendix D

Datasheets

Consult the project website for part datasheets

http://eng.fsu.edu/ece/senior_design/2015/team05/datasheets/