TELEMETRY SYSTEM FOR REAL-TIME MONITORING OF AN OFFROAD RACECAR

Undergraduate Students: Kyle Boyer, Laura Brubaker, Kyle Everly, Richard Herriman, Mark Sackett, Huy Tran
Faculty Adviser: Dr. Michael Marcellin
Department of Electrical and Computer Engineering
The University of Arizona, Tucson, AZ 85721

ABSTRACT

The University of Arizona Baja racing team competes annually in a grueling four-hour off-road endurance race which subjects vehicles to an array of obstacles such as jumps, boulders, and mud bogs. This paper examines the telemetering system created by the UA Baja Team to monitor a range of critical aspects of the car with the goal of detecting and identifying possible mechanical failures and areas with potential for improvement. Running on an Arduino Mega, the system stores all gathered data to an SD card and transmits it back to the pit wirelessly for real-time analysis.

INTRODUCTION

BACKGROUND

Every year, the University of Arizona Baja racing team joins a host of other schools from around the world to compete in a grueling three-day competition culminating with a four-hour off-road endurance race. Teams are ranked based on the capabilities of their vehicle, including maneuverability and acceleration, as well as the design process and overall marketability of the vehicle. A heavy emphasis is placed on innovation and data-driven design, which is where the telemetry system comes into play.

Back in 2012, we decided that we wanted to know how fast our car could go. There are many ways of accomplishing this. Among the available options are laser timing gates, radar guns, or simply following behind the vehicle with a regular car, and watching the speedometer. After
attempting this last method, it occurred to someone that it would be much simpler if we could just put a speedometer on the Baja car itself.

We did just that and everyone loved it. But then we thought, why stop there?

OVERVIEW

This paper is a record of our design decisions since then, and the system to which they have given rise. We will start with an overview of the purpose of the system as it stands today, and why we decided to build it from scratch. From there we will cover core features of the system, the software that drives it, and some other factors that influenced the design.

PURPOSE

In any system with moving parts, there is a risk of things breaking. When that system is a hand built race car flying off jumps on rocky terrain, the likelihood of things breaking increases considerably. Our system is designed to address this problem in two ways. The first goal of the system is to provide performance data on various pieces of the car, to aid in the design process. The second is to gather live data during a race, allowing the pit crew to more quickly identify problems and get the car back on the track.

However, the most important goal of this project is to provide its participants with knowledge and experience. There are products on the market that encompass most of the features we were looking for. However, none of them can provide the learning experience of building the system from scratch. In the following we will discuss the system that evolved to fulfill those goals.

FEATURES

DASH-MOUNTED DISPLAY

The dash-mounted display is the only part of the system accessible to the driver. Visible from the outside is the LCD display, providing the canvas for the speedometer readout, and occasional flashing alerts from the control console or the system itself. Next to the screen is the multi-purpose panic button, which allows the driver to push a data update to the remote console, even when live viewing is disabled. This feature may be used to notify the pit that the car has stopped in the event of voice communication being lost. The display housing also holds the GPS module.
GPS/MOTION DATA

In addition to providing location coordinates, the GPS module performs the velocity calculations that feed the speedometer. In fact, the GPS module was one of three options we originally considered for calculating speed.

One of the other options was an accelerometer, the data from which can be integrated to find the current speed, provided that at some point in the past the speed is known. While we have integrated an accelerometer for testing purposes, our current board design inhibits us from taking full advantage of available libraries, making it an infeasible option for this generation of the board. This will be revisited in the next iteration.

The current version does make use of the accelerometer to the extent of warning the pit crew if the vehicle is inverted. It also logs raw data from both the accelerometer and the attached gyroscope to the SD card for later analysis.

TACHOMETER

The third option for determining the speed of the car is a tachometer. Because we use a CVT (Continuously Variable Transmission), the speed of the engine is only loosely related to the speed of the wheels. Similarly, because of the uneven and frequently slippery nature of the track, even the speed of the wheels is not always a reliable indicator of vehicle speed.

In spite of that, we have included two tachometers in our design. Rather than estimating the vehicle speed, they are intended to measure the rotational speed of the input and output of the CVT, telling us the effective gear ratio.

By combining this data about the ratios between the two pulleys with velocity and acceleration data, we will be able to tune the CVT more effectively. This data will also be provided to the CVT manufacturer with feedback.

Our tachometers are custom magnetic encoders consisting of a Hall effect sensor and a ring of neodymium magnets arranged so the exposed poles alternate around each shaft. The sensor is sensitive to both the north and the south poles of the magnets, and latches in both directions. This eliminates the need for complicated filtering of the output signal.

To read the sensor, the system counts the number times the magnetic field has flipped during a time interval. Angular speed is calculated using the equation below. Note, the 60000 milliseconds converts the measurement into revolutions per minute and is needed as the time
interval is quantized in milliseconds. A measurement interval much longer than one revolution helps reduce error caused by missed flips.

\[
RPM = \frac{\text{flip count} \times 60000 \text{ (milliseconds)}}{\text{number of magnets} \times \text{measurement time interval} \text{ (milliseconds)}} \tag{1}
\]

Because the sensor resides inside the transmission housing, which is neither waterproof nor dust-proof, the sensor must be very durable. We experimented with cementing the sensor inside a drilled-out aluminum or steel bolt, which makes mounting easier. Unfortunately, the metal acts as a faraday cage and dampens the magnetic field to undetectable levels.

Next we tried covering the sensor with heat-shrink tubing and filling the end with epoxy resin. This works very well for durability (it works while completely submerged in water), and is very accurate: we tested it by mounting the magnets on a lathe, and our reading is consistently within 20 RPM of the lathe readout. That said, it has proven to be very difficult to mount securely in close enough proximity to the shaft.

We are currently experimenting with non-conductive fasteners which mount securely without impeding the magnetic fields we are trying to measure.

**TEMPERATURE**

We used temperature sensors that operate from -40°C to +125°C with a tolerance of ±°2 before calibration [1]. The wide temperature range is important as the sensors are mounted in the gearbox and CVT which get very hot. The plastic sensor is cemented into a handmade aluminum housing with threads on one end. The housings are screwed into the gearbox and back of the CVT cover. We use aluminum for the housings because it has a high thermal conductivity which means it does not adversely affect the temperature reading. Being made of the same material as the gearbox and CVT enclosure, thermal expansion does not create any gaps. This is particularly helpful in the case of the gearbox, which is full of oil.

The temperature was calculated from the sensor value using the equation below. This equation was tested and calibrated by placing the sensor in an ice bath and finding the intercept that lead to the calculated value being 0 °C.

\[
\text{Temperature in Celsius} = \left(\frac{500}{1023}\right) \times \text{Sensor Value} - 44 \tag{2}
\]
BRAKE PRESSURE
Brakes are one of the most important things you don’t want to fail on an off road vehicle. In consideration of this, we put pressure sensors on the brake lines to help track and predict if the brake are failing. The pressure sensors monitor both the front and rear brake systems. The sensors themselves are industrial grade; they can measure pressures up to 3000 psi and are shockproof and waterproof. The sensors also allow us to more easily fine tune or repair the brake system.

STEERING
The implementation of the steering sensor was an exercise in prototyping and adaptability. We strove to find a method of recording the position of the steering wheel in real time, through a robust design able to hold up to the stresses experienced out on the track. The final design consists of a ten turn potentiometer, a 3D-printed mounting bridge, and a 3D-printed 1-1 gear system. We tried several designs, including a belt transfer system, a double sided bridge of double height, and a variety of gear radius and teeth sizes, eventually deciding on the simplistic three part system described above. Never before attempted by our Baja Team, the steering sensor provides critical data for fine-tuning our car and is a major part of our data stream allowing the team to accurately recreate the events of a race post competition.

TIE ROD INTEGRITY
One of the most likely things to break in an off-road vehicle is a tie rod, a shaft of metal that connects the steering rack to the front wheels. Tie rods are important because they make the front wheels steer. We added integrity sensors to the tie rods, in order to warn the pit crew early in the event of a tie rod breaking. The sensors monitor continuity along the tie rod, and are triggered by any break in continuity. In addition to these initial sensors, we are also implementing flex sensors to detect bent tie rods. The flex sensors function similarly to the continuity sensors, however the connection is broken at a certain angle, rather than only if the tie rod itself breaks.

STATUS INDICATORS
The core and peripheral modules each have several LEDs to indicate the status of various components. If the power supply is attached backwards, a red LED glows. Each voltage regulator has a green status indicator. Three software controlled LEDs (two by the Arduino, one by the XBee) can be configured to convey a variety of different conditions.
COMMUNICATIONS

The complete system consists of several peripheral modules, each of which communicates with
the core module via a serial interface. The SD card reader uses SPI, the IMU uses I2C, and
everything else uses UART.

The core module communicates with the remote console using a serial protocol passed through
two XBee radios in transparent mode. An XBee-PRO 900 XSC S3B operating at 900 MHz on
full power has a line of sight range of 4 to 9 miles [2]. Mounting one antenna atop a flagpole
significantly improves achievable range in hilly terrain.

SOFTWARE

CORE MODULE

The core module aggregates data from all attached sensors with data from all other modules and
logs it to a microSD card. It also handles wireless communications with the remote console via
XBee and any necessary interaction with the driver.

PERIPHERAL DATA ACQUISITION

The peripheral data acquisition module is designed to extend the capabilities of the core system.
It communicates with the core system via a single serial line, and is intended to minimize the
quantity of wire running through the vehicle.

Distributing sensors among modules also increases the maximum sample rate, as each module
performs the necessary averaging and extrema detection operations for its attached sensors.

On the current iteration of our vehicle, only one peripheral module is in use. It is mounted behind
the firewall (a large metal plate separating the driver from the engine), and monitors the
temperature of the gearbox and the transmission, as well as the input and output RPMs of the
transmission. This module also contains an accelerometer as it is closer to the car’s center of
gravity.

REMOTE DATA VISUALIZATION AND CONTROL CONSOLE

The remote console serves two functions. First, it is the main control panel for the system. From
the console it is possible to pause and restart data logging, change the name of the log file,
modify the sample rate, and send messages to the driver.
The second function handled by the remote console is live visualization. The remote console receives data from the core module via an XBee and then interprets and displays the data on a virtual dashboard. The console can be configured to update at set intervals, or to update on user interaction.

At present, the update interval is the same as the log interval. When the console is set to update on request, either the update button on the console or the multi-purpose panic button on the dash can be used to force an update. Forcing an update when the panic button is pressed ensures that the pit crew is notified promptly. This feature is also useful during testing to display only data points of interest on the console. The core module continues to log to the SD card even if the remote console does not request an update.

The system has a variable log frequency, to reflect the requirements of different events. Without modifying the code, data can be written to the SD card as frequently as eight times per second, as infrequently as once every ten seconds, or any frequency in between.

DATA LOGGING

The microSD reader is integrated into the core module, so that even if peripheral modules become disconnected, data can still be stored in nonvolatile memory. We originally attempted to implement a custom binary format, but found that the CSV format is nearly as fast, and can be opened without modification by most data analysis software.

In the data file each line begins with a unique key. Whenever the system starts it scans the SD file for the most recent index, retrieves that number, and continues logging from there. Because the file grows continuously, it can take more than a minute to find the most recent line. To mitigate this, we tried to skip to the end of the file and read backward to find the most recent key. We quickly discovered that the library we are using does not support this. Additionally, because lines vary in length depending on the presence of the peripheral module, it is infeasible to seek directly to the correct position. We have found that we can shorten the scan time by skipping nearly to the end of the file and scanning from there. We chose 5000 bytes before the end of the file as an arbitrary constant, since it is far longer than the longest line and short enough to prevent noticeable lag at startup.

While it is theoretically possible to use the SD card in a mode supporting parallel data lines, the reader on the board only supports SPI. This limits the theoretical maximum data transfer rate, but in practice it is still faster than the rest of the system, and is therefore not a problem.
In the process of logging data to the SD card, the core module sends a request for data to the peripheral module. The peripheral module then performs any final calculations before converting the numerical values to a string of ASCII characters to send over the serial line. Once the entire string has arrived, the core module simply concatenates it with its own data string, and writes it to the SD card. If no data arrives, or the data is corrupted, an error code is concatenated instead. Because the serial buffer on the Arduino Mega is 64 bytes and the data string consistently exceeds this limit, the peripheral module breaks up messages into smaller blocks. Specifically, the peripheral module transmits 60 characters then waits for an acknowledgment from the core module. Both modules have an initial timeout of 50 milliseconds. In case the peripheral module comes disconnected during a transmission, the core module also has a timeout of 5 milliseconds while it waits for each character.

CUSTOM LCD BACKPACK FIRMWARE

The LCD screen is controlled using a serial backpack, which accepts commands over a serial line and controls the screen itself using a parallel bus. The stock firmware supports eight pixel high ASCII characters, which are too small for the driver to read during the race. Because the firmware is open source, we were able to modify it to display numbers the full height of the screen.

DESIGN CONSIDERATIONS

THE SHIELD

The core and peripheral modules each consist of an Arduino Mega and a custom Arduino compatible shield. The Arduino is well suited to this task because of its small form factor and many communication interfaces. Being both inexpensive and open source, it has a thriving developer community, making documentation and third party libraries easier to find.

While it would be possible to integrate an Atmega2560 (the microcontroller at the heart of the Arduino Mega) into our board directly without losing the benefit of third-party libraries and documentation, this would require surface-mounting the microcontroller to the board. Using an Arduino makes the brain of the system replaceable in an emergency.

There are several advantages to using a custom printed shield versus one of the stock shields. First of all, because we designed it specially for our system it is very compact yet fully capable of completing all the processes required of it. The shield is designed as a working prototype, so it has a lot of extra exposed pins allowing for further changes. Furthermore, each sensor has its own connector allowing great customizability.
Another feature is that the shield is used in two different modules. Both the core and peripheral modules use the same shield (Figure 1) but with different components soldered on. This reduces production costs significantly, and allows any module to be reprogrammed as the core module should the core module develop a fault.

One disadvantage of the shield design is the large number of sensor connectors. Each connector adds bulk to the board with in turn makes its housing larger. It is particularly difficult to fit the entire assembly into the housing once all the wires were attached. Also, because each sensor connector has its own power and ground there are a lot of redundant wires going to the connector on the inside of the electronics housing. The connector we used this year is actually designed to mount directly to a PCB as a through-hole component. By redesigning the shield to take advantage of this, we will be able to eliminate all of the connectors used to interface with external components.

Figure 1

MODULAR DESIGN

The system is designed from the boards up to be highly modular. Nearly every major component can be replaced in the event of a malfunction. We designed a modular board with the arduous nature of the race and life of the car in mind. We expected components to break and wanted to be able to replace them.
WATERPROOFING

Along with designing in a high degree of modularity we also planned for the harsh environment by designing tough 3D printed enclosures. These were designed specially to fit the boards, securely mount on the car, and be resistant to physical assault and water. The housing consists of an inner and outer shell fabricated in two pieces then sealed together with silicone. Each module connects to the vehicle's wire harness via a single high-quality waterproof connector.

SHOCK RESISTANT

Along with taking precautions to avoid physical damage, we also implemented several measures to avoid electrical damage. To begin, we put a ground plane on each side of the shield to improve heat dissipation. We also put a reverse polarity diode on the primary power connector to stop inverted connections from destroying the components. In future we are considering placing ferrite chokes on the power and ground connections, as there is some risk of shock due to sharing a common ground with the spark plug.

CONCLUSIONS

In conclusion, we view our telemetry system as a successful first step in what will hopefully be a long line of exciting systems to come. It is by far the most complex electrical system the UA Baja Team has ever seen. Not only has the system been a great success in coordination and organization but it has proved a tremendous learning experience. We are excited to take the experiences of this year to make next year’s system even more impressive.

ACKNOWLEDGEMENTS

We would like to thank Dr. Michael Marcellin for making this project possible and the rest of the UA Baja Team for putting up with the clouds of blue smoke.

CITATIONS AND OTHER REFERENCES