Cornell University

Department of Computer Science

Active Suspension for a Baja SAE car

Jay Fetter
jdf258

January 8, 2018

Contents

1 Introduction

The goal of the Baja SAE competition is to design and build a single-seat, all-terrain, sporting vehicle whose driver is contained within the structure of the vehicle. The vehicle is to be a prototype for a reliable, maintainable, ergonomic, and economic production vehicle which serves a recreational user market, sized at approximately 4,000 units per year. The vehicle should aspire to market-leading performance in terms of speed, handling, ride, and ruggedness over rough terrain and off-road conditions.[1]

The Cornell Baja SAE team is one of the world's premier collegiate Baja teams with 5 first place finishes in the past 7 years as well as winning the Iron Team Award in the 2013-2014 race season. The Iron Team Award is given to the top-scoring team over the course of the three regional competitions and is the holy grail of Baja SAE.

One of the most important qualities of a Baja SAE car is a well-tuned suspension. The suspension of a vehicle provides a smooth and comfortable ride for the driver. In offroad racing, the suspension allows the vehicle to tackle intense terrain such as jumps and rock gardens. A suspension system can be modeled as a spring-mass system and ideally the system is critically damped to the external perturbations. Since Cornell's Baja cars typically use air shocks this spring rate is tuned via the air pressure in the shock. This gives a slightly non-linear spring rate compared to a coil-over shock, but has much less mass and more tunability.

The goal of this project is to provide an active suspension system for a Baja car and prove its feasibility for the Baja SAE competition. The system developed during this project will be strictly a prototype and will ignore concerns such as weight and serve as a proof of concept that a Baja SAE car can run active suspension within the competition rules. This system will use a sensor suite on the car to provide automatic damping adjustment of the shocks. As part of the project, this system will be integrated into an older competition Cornell Baja car, designated OD10[2].

1.1 Active Suspension and Baja SAE

An active suspension system allows for modifying spring rates dynamically. This can reduce pitch and roll and provide a better ride performance overall. The most important event in the Baja SAE competition is the 4 hour endurance race on the last day of competition. This is a circuit-based race where there could be up to 100 cars on the track at a time. This plays well into an active suspension system. By mapping areas of the track to different damping profiles, the vehicle can achieve an optimal lap time. This will allow the team to gain laps on the competition and subsequently earn more points. This would also reduce driver fatigue since the vehicle ride will be smoother.

There are many reasons why active suspension systems are not on every car.

[1] `http://bajasae.net/content/2018-BAJA-RULES-FINAL-2017-08-30.pdf`

There are no off-the-shelf active shocks that teams can purchase that would fit a Baja car. It is a complex electrical system which can easily be destroyed during a Baja race due to the mud, water, rocks, and other hazardous terrain. Baja SAE also mainly attracts mechanical engineers who are not as comfortable and familiar with electronics.

However, that changed just this year. Fox Racing introduced its live valve system. The Cornell Baja team has a very good relationship with Fox Racing and I was able to get a contact at Fox who was able to make a custom set of active shocks for this project, which will be discussed later. Recently, Polaris also introduced a production vehicle with active suspension. If the big players in industry are starting to recognize the market desire for active suspension systems, these systems will trickle down to the Baja SAE teams. Last year, Ecole De Technologie Superieure (ETS) ran an electric continuously variable transmission (CVT) and performed extremely well. While this is a different sort of electronic control system, it does demonstrate that a complex electrical system can survive the challenge of Baja with great success.

1.2 Previous Endurance Race Analysis

Previously, I was able to acquire GPS vs. Velocity information for the 2016 Baja SAE Endurance Races. I was able to use this data to make depth maps of the endurance race based on GPS location and velocity. This type of analysis is useful since it can be used to determine how fast the car was traveling over different grades of terrain.

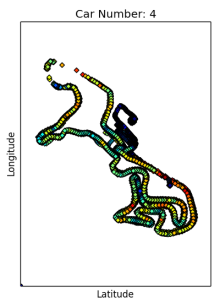

Figure 1: Endurance Race. Speed is indicated by color. Red is faster, Blue is slower.

This previous work served as an inspiration for this project. I hoped to be able to take GPS combined with roll, pitch, yaw, and 3-axis acceleration data and generate different depth plots. These GPS mappings would allow a system to determine the optimal damping given the state of the vehicle and the current location.

1.3 Vehicle Platform

The vehicle platform I am using for this project is the 2013-2014 Cornell Baja vehicle, OD10. This car was the Iron Team winner, but had fallen into disrepair after a crash in Fall, 2014. The benefit of using this car was that it had plenty of spare components, a robust drivetrain, and a trailing link suspension. The trailing link suspension provides more clearance in the rear for mounting larger shocks compared to the H-Arm suspension used on subsequent vehicles.

There were only a couple of parts that needed to be remachined for the car. I needed to remake several bushings for the CVT as well as a new set of tungsten weights. I rebuilt the entire brake system and redesigned the brake pushrods to be easier to machine.

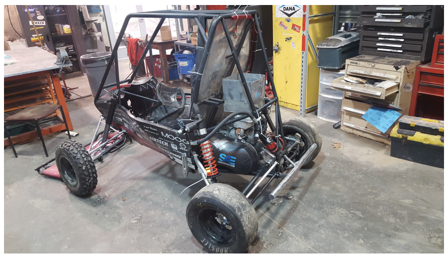
Figure 2: OD10 with active shocks

Even though there were plenty of spares, it takes a lot to rebuild a Baja car. I had stored the car in Rochester with a couple of Baja alumni due to my inability to find space on campus. I reached out to several labs and started several e-mail chains, but they all fell through. This meant that I had to take several day trips out to Rochester where the car was initially stored to work on it. After Thanksgiving break, I was able to move the car into GM lab in the basement of Upson to do my final repairs and integrate everything together.

1.4 Partnership with Fox Factory

In order to implement active suspension, I needed to have shocks that supported damping control. I considered building/modifying shocks myself, but decided to reach out to an actual shock manufacturers first.

I contacted Magneride[2] , who make magnetically controlled dampers for automobiles. My initial contacts went well, but they only offered to consult with me on the project. This meant that I would have to acquire a set of their shocks first which would be very expensive.

The next place I contacted was Fox Racing. Fox has been a sponsor for the Cornell Baja team for a long time and has a good relationship with the team. While Fox did not really have any public information released about making active shocks for off-road, I heard some rumors that they were working on it and felt that it was worth reaching out to them to see if they could help. I had some initial e-mail correspondence with the usual Baja contact and was directed to an engineer in the Off-Road division who could potentially help.

[2] `http://www.magneride.com`

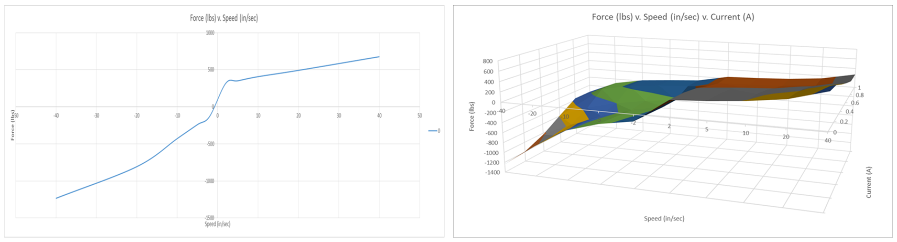

Figure 3: Damping Curve for Regular Shock (left) v. Damping Curve for Active Shock (right)

I was able to talk to him over the phone and he thought the project was interesting and was willing to help me acquire some shocks. He was able to get me four custom Podium 2.5 Live Valve shocks. He had them cut to length so they would fit on OD10. He was always able to answer email questions and talk over the phone and he really helped this project come together.

He ran the shocks on a shock dynamometer before sending them to me so I could have an idea of the current to damping relationship. This relationship is shown in figure 3.

In order to get the shocks to work for OD10, I needed to design and make 8 new shocks spacers. After making these spacers, I was able to mount the shocks on the rear of the car.

I had some clearance issues in the front of the car which could be solved by making bent upper-a arms, but I decided to put that off and focus on the electronics.

1.5 Dumbo

Dumbo is the current electrical data logging system for the Cornell Baja team. This system was developed last year (2016-2017) by Raghava Kumar and me and is able to log and wirelessly stream data from the car.

I decided to use this system for the data processing for my project for several reasons. It has support for many of the sensors I wished to interface with, an existing software library, and a modular wiring harness. By slightly modifying this system, I would not need to redo a lot of wiring and sensor interfacing. It also would let me connect wirelessly to the car which is very convenient.

2 Implementation

2.1 Electrical Hardware

In order to control the solenoids in the shocks, I needed to develop a current based solenoid controller. The first step in this process was defining the requirements for the module. The following requirements were decided:

– Ability to control 2 solenoids

– Ability to source up to 1.2A per solenoid

– CAN interface

– An onboard accelerometer

– STM32 microcontroller

– UART interface for debugging

– Test Points

With those requirements in mind, I went ahead and planned on running a two revision board design cycle. This would allow me to speed up my development and not worry about being perfect on the first revision.

I decided on using an STM32F429 microcontroller due to my access to development boards and familiarity with the software. I then chose power supplies based on their current ratings and that I would pass unregulated 12V battery power to solenoids.

For revision 1, I chose the Texas Instruments DRV110APWR chip as my solenoid controller. The proportional solenoids use a current spectrum to control position and I thought that I could control the current output of the DRV110APWR with a Digital-to-Analog (DAC) signal. This turned out to be incorrect and an error on the design tool for the chip. On revision 1, the CAN pins on the microcontroller were damaged/non-functional. This resulted in several days of trying to figure out why CAN was not working before building my own makeshift CAN modules using extra CAN transceivers and development boards and verifying that it was indeed a hardware problem.

On revision 2, I switched to using an Infineon TLE7241E proportional solenoid control chip. This chip is everything I could want out of a solenoid control chip. It is programmable via a SPI interface and supports 2 solenoids up to 1.2A each. This is essentially my application so I was very pleased with the chip. I also switched the CAN transceiver from a 5V chip to a 3.3V chip. I thought this would allow me to remove the 5V regulators from my board (which I did and was very excited about), but that was a mistake as the TLE7241E requires a 5V supply. This was not a difficult rework as I made sure to run traces out of the chip then use a via to connect them to the power plane instead of running to a via underneath the chip.

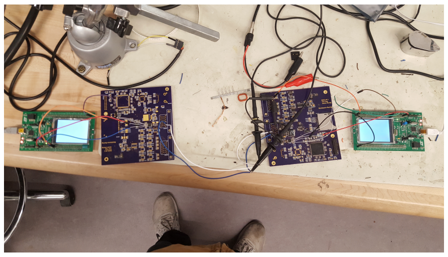

Figure 4: Makeshift CAN setup

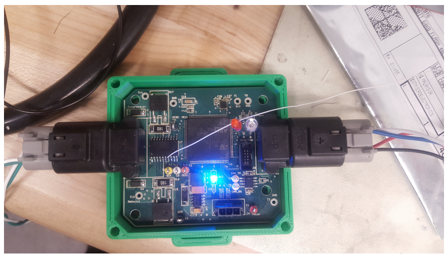

Figure 5: Assembled PCB.

2.1.1 PCB Layout

My main concern for PCB layout on this board was ensuring good thermal properties. This board could potentially be seeing upwards of 2A being drawn by a single chip and I wanted to make sure that there would not be any heating issues. I used a 4 layer board design with the top and bottom layers as routing layers and the inner layers as ground and power. The power plane was segmented into unregulated 12V from the battery and 3.3V regulated. The 5V for the TLE7241E is supplied by an offboard regulator.

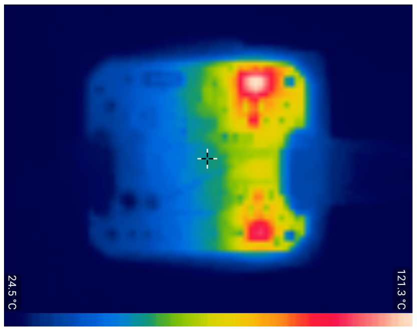

Figure 6: Thermal image of PCB during operation. Camera uncalibrated.

2.2 Dumbo Modifications

In order to get Dumbo to work for this project, I needed to modify the system to incorporate CAN. The framework had skeleton CAN code for receiving and logging CAN messages, but no work was done on actually implementing CAN and getting it all setup. I was able to get CAN receiving and logging working rather easily with some slight issues (the CAN ID did not log, which makes the CAN messages undecodable).

The next step I took in modifying the Dumbo system was updating the configuration tool to be able to send CAN messages over WiFi. The configuration tool is called py_tool and was written in Python. This program allows the user to connect to Dumbo and manage configuration and logging through a command line interface. The messaging protocol uses Protobufs [3] and I needed to modify many different parts of the system to incorporate my desired functionality.

I first defined a CAN messaging Protobuf and compiled using the Protobuf version on Dumbo. Then, I had to modify py_tool to interpret and pack the new message correctly. Then I had to modify the C++ code running on the Raspberry Pi in Dumbo to also interpret the new messages and send them over to the microcontroller. Finally, I modifying the firmware on the microcontroller to accept the new messaging format and interpret them correctly.

2.3 Software

I used the STM32F429 microcontroller for this project[4]. This is a 32-bit ARM based microcontroller from ST Microelectronics and I have done a lot of design work with in the past.

The programming environment I set up was based of the STM32CubeMX framework[5] that ST provides. STM32CubeMX is a pin planner and code generator. When used correctly it will generate the initialization code for the peripherals you want to use. This covers things like CAN clock rates to GPIO interrupts. I used STM32CubeMX to generate code for both Makefiles and MDK-Arm environments. On a windows machine, I setup Keil Vision IDE[**?**] and used that for debugging and development.

The embedded software for these modules was simple. I needed them to interpret CAN messages and send out SPI commands to the solenoid control chips. I started by writing the CAN driver. This proved more challenging that I initially anticipated. Since I did not have a working CAN bus setup to integrate my module into, I had no reference to compare against. This made it very difficult to debug the hardware vs. software issues. I eventually decided to setup a bare bones CAN bus system using two development boards with wires jumped to two CAN controller chips. Using this setup, I was able to get the CAN bus software running on the development boards. Since CAN is a bus system, I left this system intact and added my boards to it. This would allow me to verify if I was having a software vs. hardware problem.

After I got CAN up and running on my boards, I moved on to adding CAN functionality to the Dumbo system. For the purposes of this project, I was not concerned with preserving the existing functionality of Dumbo so I used a CAN_RX interrupt for receiving CAN messages. This would not mesh well when Dumbo is busy reading analog values, but for my purposes it worked. A better solution would be to use DMA by mapping the memory controller of the CAN receive array to a local array that could be read when sending information to the Raspberry Pi. This would be similar to the way UART is handled on Dumbo right now.

[3]`https://developers.google.com/protocol-buffers/`

[4]`http://www.st.com/en/microcontrollers/stm32f429zi.html`

[5]`http://www.st.com/en/development-tools/stm32cubemx.html`

The next step was writing the SPI driver for the TLE7241 solenoid control chip. The TLE7241 has an interesting SPI protocol where the master device sends a command, the master needs to send another command to receive the response to the first command. After wrapping my head around that quirk as well as discovering that I needed to power the chip with 5V rather than 3.3V, I was able to control the solenoids over SPI. Then I wrote some functions to generalize the messaging protocol.

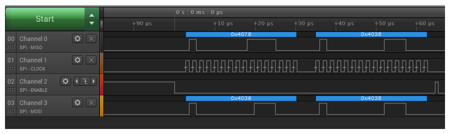

Figure 7: Example SPI Communication.

3 Future Work

3.1 Realistic System for the Cornell Baja Team

The shocks for this project should not go on a competition Baja car. One of the shocks that I used weighs about 19lbs which is 16lbs heavier than the current Fox Float 3s the team uses. For the purposes of this project, I did not consider making sure my system could immediately be run on the upcoming Baja car. This would have required too much mechanical design work. However, the electrical systems I made are able to control any proportional solenoid. This means that if the team can figure out how to modify the Float 3 by either working with Fox directly or modifying the shocks themselves, they would be able to run active suspension.

There are a couple of changes to the system that I would like to outline if the team considers going forward with this project.

One of the biggest obstacles I faced when designing my system was trying to incorporate it into the existing Dumbo framework. I did this because I wanted to use a decentralized approach and underestimated the robustness of the Dumbo software stack. If this was a system that would be run on the car, it should follow the criteria every part on the car goes through:

1. Minimize weight, manufacturing time, and overall footprint
2. Be easy to install and use
3. Reduce cost as much as possible
4. As simple as possible

A better system would move the solenoid control to a centralized location that also does the computing. This system should be similar to the Dumbo system in that it uses a Raspberry Pi and STM32F4 microcontroller, but should have its own unique software stack. This board should be able to do all the control for the suspension by itself as well as read in sensory information. This will let the software stack be much neater since it does not need to contain all the logging information and can just focus on the task of solenoid control. The raspberry pi should be responsible for doing high level computation and data analysis while the microcontroller manages dealing with incoming information in real-time.

This would use a Infineon TLE7242-2G which can control 4 solenoids. The Float 3 shocks do not have a compression and rebound adjustment like the Podium 2.5 shocks and it would be easiest to modify the Float 3 to only have a single compression and rebound controlling solenoid. Theoretically, if the system is fast enough it would be able to adjust the solenoid on the compression stroke and then again on the rebound stroke. If there was a need to control 8 solenoids, Infineon makes the TLE8242-2L which can do that.

The board should have a SPI connection with the Raspberry Pi for high speed communication. The Raspberry Pi should have a similar setup to the Dumbo system and can reuse a lot of the codebase.

3.1.1 Manpower Requirements

Developing an active suspension setup has many components. There is the electrical control hardware, electrical control software, high level software, and the shocks themselves. There would need to be a dedicated electrical engineer and two embedded software engineers for finishing up the electronic control, at least two mechanical engineers working on the shocks, and at least one software engineer working on the high level control software. I would recommend starting the process in the Spring semester due to the long lead times working with outside vendors and the amount of new design work that needs to be done.

The electrical setup is in a very good place. The system that was designed for this project, while not ideal for a competition car, allows the user to control the damping of the shocks. The work for this portion of the project would be moving the existing schematics over to the centralized system outlined above. The embedded software is also in a good place. The software would need to be ported over to support the different solenoid controller chips, but the drivers I wrote for this project are a good starting point for that.

On the mechanical engineering side, there is a lot of work. They would need to either work with Fox on developing a Float 3 active suspension setup or figure out how to modify the existing shocks to work with proportional solenoids. Fox has their Quick Switch 3-position (QS3) system, which is similar to the system that was used on the Podium 2.5 shocks before the active solenoids were integrated. The QS3 system comes with the FACTORY SERIES FLOAT 3

EVOL QS3-R[6] shocks which weigh approximately 4.4lbs[7]. This is about a 0.7lb increase over standard Float 3 shocks.

The high level software has the most flexibility. A bare-bones team of one engineer should be able to get the Raspberry Pi to STM32F4 communication and be able to have some basic system of control working by the end of a single semester. If more manpower can be dedicated to the project, getting a basic machine learning model done in a single semester is possible. I would recommend trying to replicate previous work modeling dynamic systems using reinforcement learning [8].

3.2 Dumbo

As one of the first people to try and modify the existing Dumbo software stack, I was dissatisfied with how difficult it was to get a change in. The software did not feel robust and other functionality kept breaking even if it was not related to the functionality that I was trying to incorporate. I believe that this system should be reworked for a more stable software stack. The use of Protobufs is a good start, but I think that they should be stored in a centralized place, which will make it easier to work with.

I would recommend working on more code generation based on the Protobuf files. This would allow the user to make sure that any code related to the changes being made gets those changes incorporated into it. This could be a simple script that manages Protobuf compilation and copies the files to the needed places. This would make it easy for people to work on Dumbo in the future. Since there exists both the Python and C++ generated Protobuf files, it can be difficult to keep track of them. I believe the py_tool interface should be rewritten in C++ to keep the languages consistent in the system. This would eliminate the need to generate Python Protobuf files.

There are a lot of moving parts for this system and a lot of ways that it can go wrong. If the team wants this system to be a data logger for many years to come, they must take ownership of the software and work to make it more robust and user-friendly.

4 Acknowledgements

I would not have been able to accomplish this project without the help of many people. I would like to thank my advisor, Bruce Land, for helping me with any issues I came across. The team at Fox for supplying the shocks, especially Ivan Tong for working with me and Issac Chapluk for building the custom set of shocks. Brent and Torin from G&C Engineering for helping store and repair OD10 in their shop in Rochester. I would like to thank my parents as well for their support and interest in my project. Finally, I would like to thank the

[6] `https://www.ridefox.com/2016/product.php?m=snow&t=shocks&p=85`
[7] `http://www.carbonsled.com/F3QS3-tech-BSS.html`
[8] `http://ai.stanford.edu/~ang/papers/iser04-invertedflight.pdf`

Cornell Baja Racing team for allowing me to use their lab space for electrical development and parts of the rebuild.

5 Appendix

Github for code, altium files: `https://github.com/Cornell-Baja/Active_Suspension`
To request access please reach out to: jdf258@cornell.edu