The Rise of a New HERO

We can rebuild it, stronger, faster... smaller?

or

Migrating the 2014 Zebracorn Robot from the cRIO Platform to the CTRE HERO

Another White (and Black) Paper by FRC Team 900 - The Zebracorns

Contact: support@team900.org

Project Summary

This project began as a thought exercise after one of our mentors had been playing with the new HERO board from Cross the Road Electronics (CTRE). The board was lightweight and versatile enough to be a very reliable robot control system. It was only a thought exercise until one day our trusty old 2014 robot (Hot Sauce), which we use frequently for outreach demos, broke. For some reason it could no longer drive without shaking violently and the control packets sent from the driver station were dropping out. Aside from not functioning, the violent shaking was clearly unsafe operation.

Image 1: Another Satisfied Customer at an Outreach Event

This robot had a unique shot in 2014 and could easily advance the ball down the field reducing cycle times considerably. It is considered by many[1] to be a pretty decent robot, not to mention our go to robot for outreach events, so we obviously couldn't just let it go without trying to revive it.

Instead of trying to reprogram the cRIO, we opted to replace it along with the motor controllers and the rest of the control system. Our goals were to restore the robot to working order so that we can continue to use it for demos while maintaining safety features such as the ability to disable the robot remotely, and keep it all to a minimal budget.

[1] Ok, not many but Karthik likes it and that's enough for us: https://www.chiefdelphi.com/forums/showpost.php?p=1585255&postcount=64

Fortunately, through a very generous donation from the kind folks at Cross the Road Electronics we were able to easily satisfy our goal for keeping this project to a minimal budget. They provided us with a HERO control board, six Talon SRXs, and some encoder breakouts. We really can't thank them enough for their support.

TLDR: Our robot needed a new brain so we asked the Wizard[2] for some help.

What is a HERO anyway?

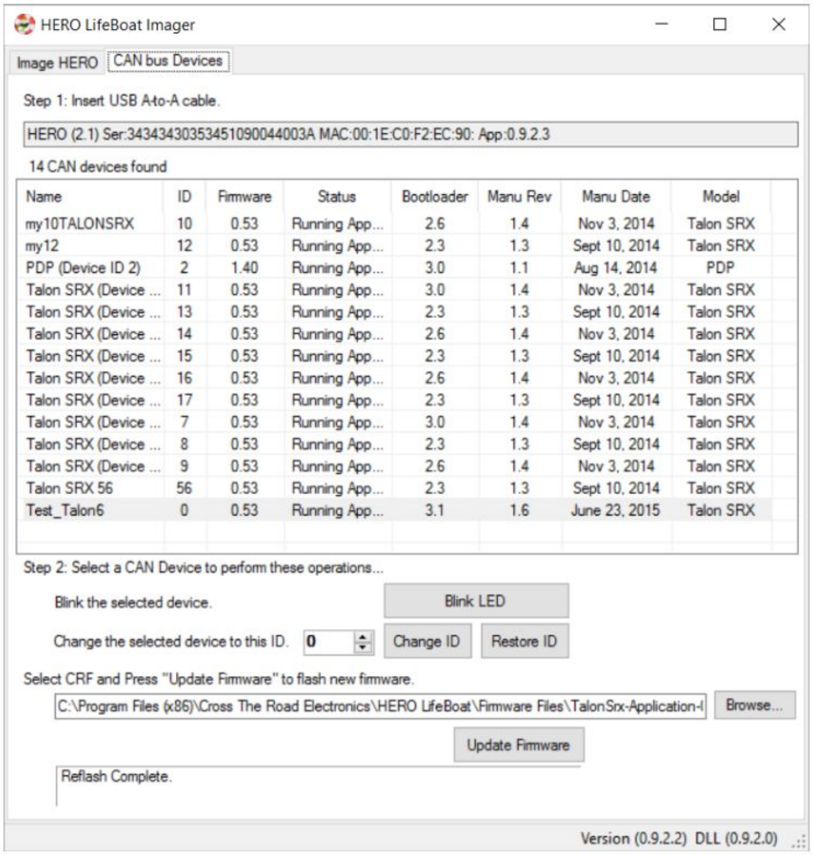
No, it's not a funny sandwich; that's a gyro. No, not a gyro, that's a thing that helps you orient the robot. Look, a HERO is a small embedded microcontroller that you can program in C# using Microsoft Visual Studio and is compatible with CAN based FIRST robotics system control components like Talon SRXs. It is also compatible with the Gadgeteer devices in the .NET Microcontroller Framework. It looks like this:

Image 2: The HERO we deserve.

It's built and sold by Cross the Road Electronics (CTRE). You can find out a lot more information about it at their website here: http://www.ctr-electronics.com/hro.html

Did we mention it also has USB support for HID controllers like the Logitech F310 and F710? No? Well, it does. It also has support for (re)setting CAN Device IDs using a USB cable and a handy program provided by CTRE. That program looks like this:

[2] The Wizard in this case is the amazing group of dedicated FIRST supporters working at Cross the Road Electronics who generously donated the parts necessary to complete this project. We are truly grateful for their support and assistance and this would not have happened without them.



Step 1: Insert USB A-to-A cable.

HERO (2.1) Ser:343434303534510900440003A MAC:00:1E:C0:F2:EC:90: App:0.9.2.3

14 CAN devices found

Name	ID	Firmware	Status	Bootloader	Manu Rev	Manu Date	Model
my10TALONSRX	10	0.53	Running App...	2.6	1.4	Nov 3, 2014	Talon SRX
my12	12	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
PDP (Device ID 2)	2	1.40	Running App...	3.0	1.1	Aug 14, 2014	PDP
Talon SRX (Device ...	11	0.53	Running App...	3.0	1.4	Nov 3, 2014	Talon SRX
Talon SRX (Device ...	13	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
Talon SRX (Device ...	14	0.53	Running App...	2.6	1.4	Nov 3, 2014	Talon SRX
Talon SRX (Device ...	15	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
Talon SRX (Device ...	16	0.53	Running App...	2.6	1.4	Nov 3, 2014	Talon SRX
Talon SRX (Device ...	17	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
Talon SRX (Device ...	7	0.53	Running App...	3.0	1.4	Nov 3, 2014	Talon SRX
Talon SRX (Device ...	8	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
Talon SRX (Device ...	9	0.53	Running App...	2.6	1.4	Nov 3, 2014	Talon SRX
Talon SRX 56	56	0.53	Running App...	2.3	1.3	Sept 10, 2014	Talon SRX
Test_Talon6	0	0.53	Running App...	3.1	1.6	June 23, 2015	Talon SRX

Step 2: Select a CAN Device to perform these operations...

Blink the selected device.

Change the selected device to this ID. 0

Select CRF and Press "Update Firmware" to flash new firmware.

C:\Program Files (x86)\Cross The Road Electronics\HERO LifeBoat\Firmware Files\TalonSrx-Application-

Reflash Complete.

Version (0.9.2.2) DLL (0.9.2.0)

Image 3: LifeBoat Software from CTRE

Out With the Old

The first thing we needed to do was rip out the old control system. A few of our dedicated students ripped it out in short order. It involved tearing out the old cRIO, the old Jaguars, a 2CAN (2.0, of course), the power distribution system, a D-Link router, an Axis Camera, and a lot of wires. Here are some action shots of the teardown process:

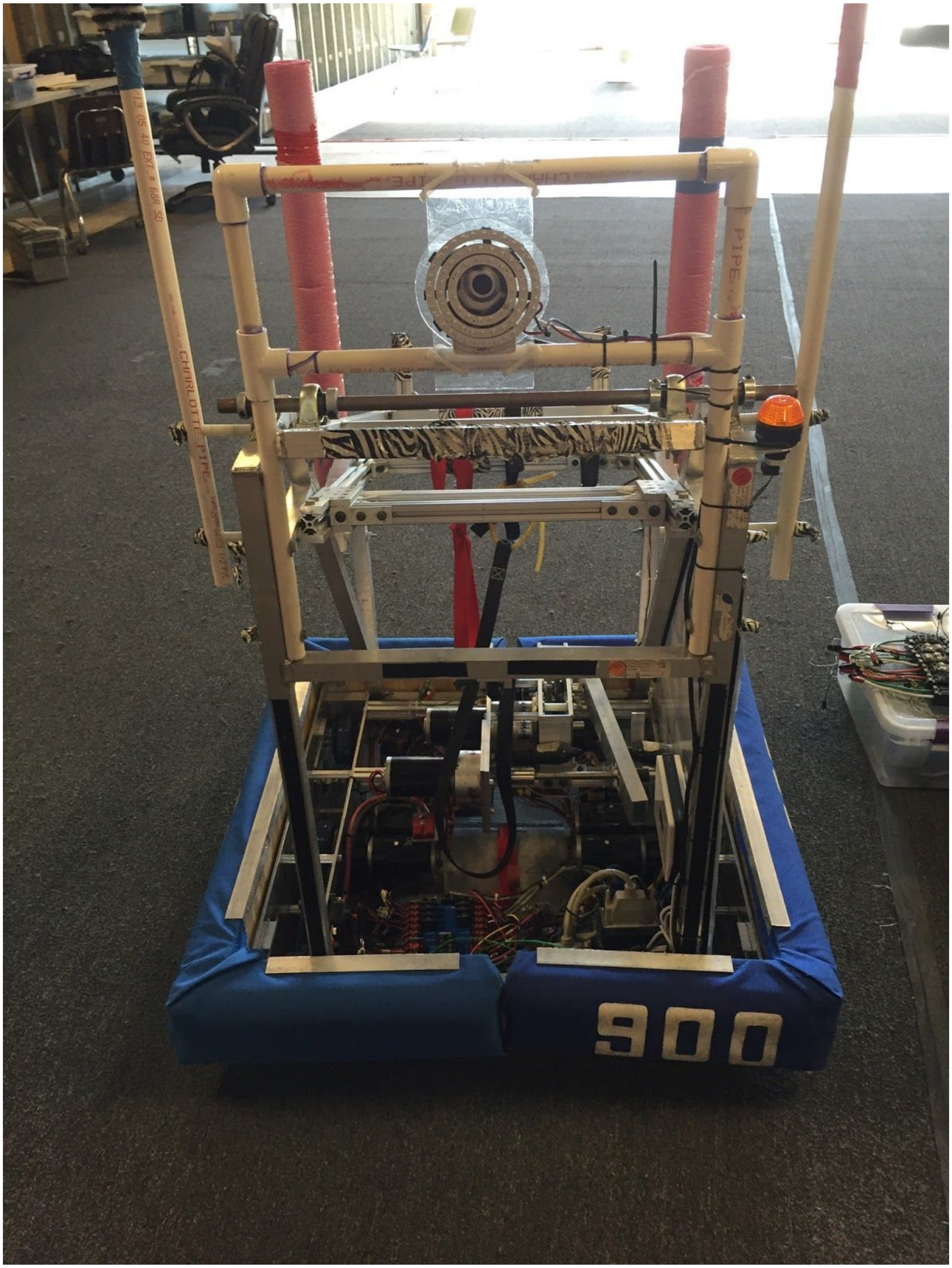


Image 4: The Patient Before Surgery

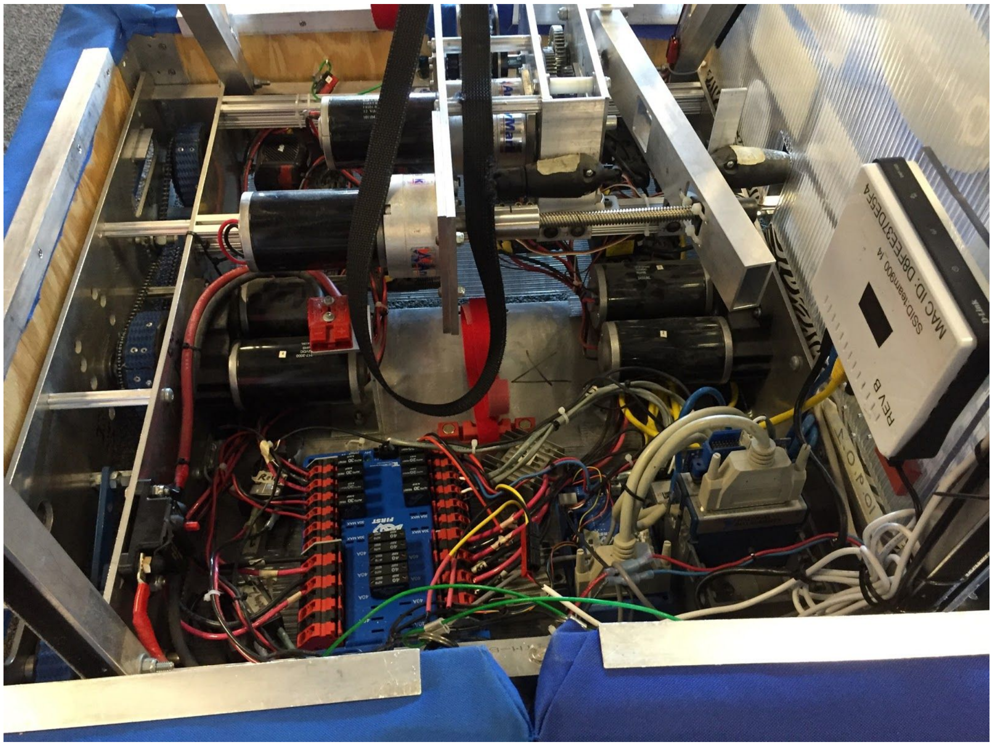


Image 5: What a mess…

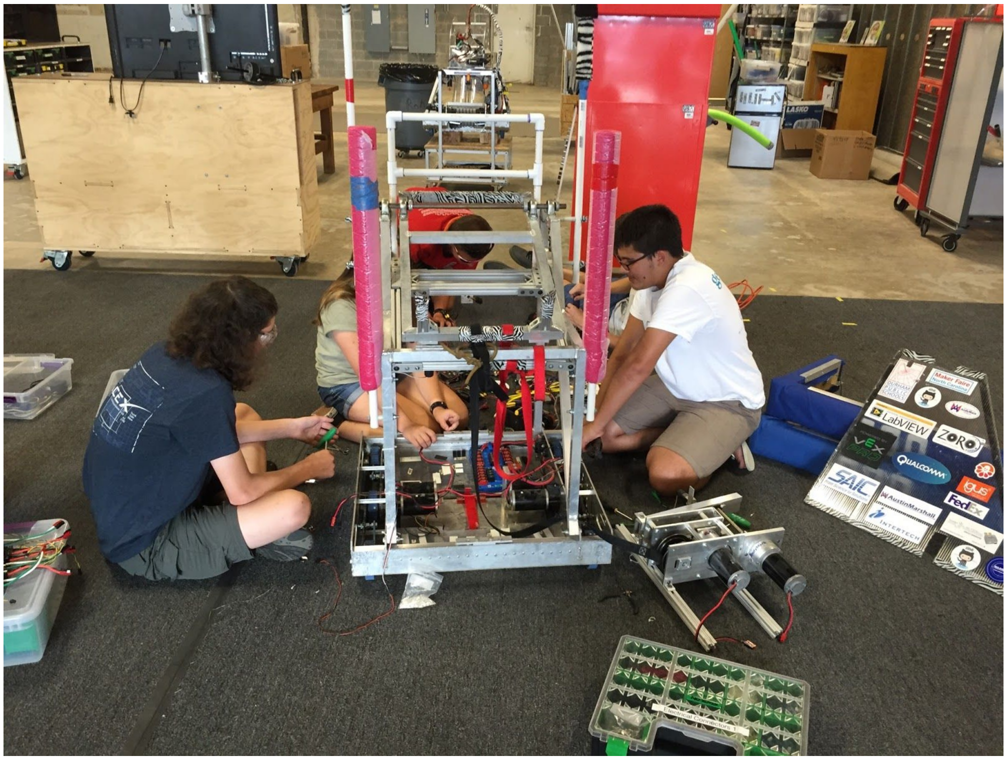


Image 6: Taking it all apart.

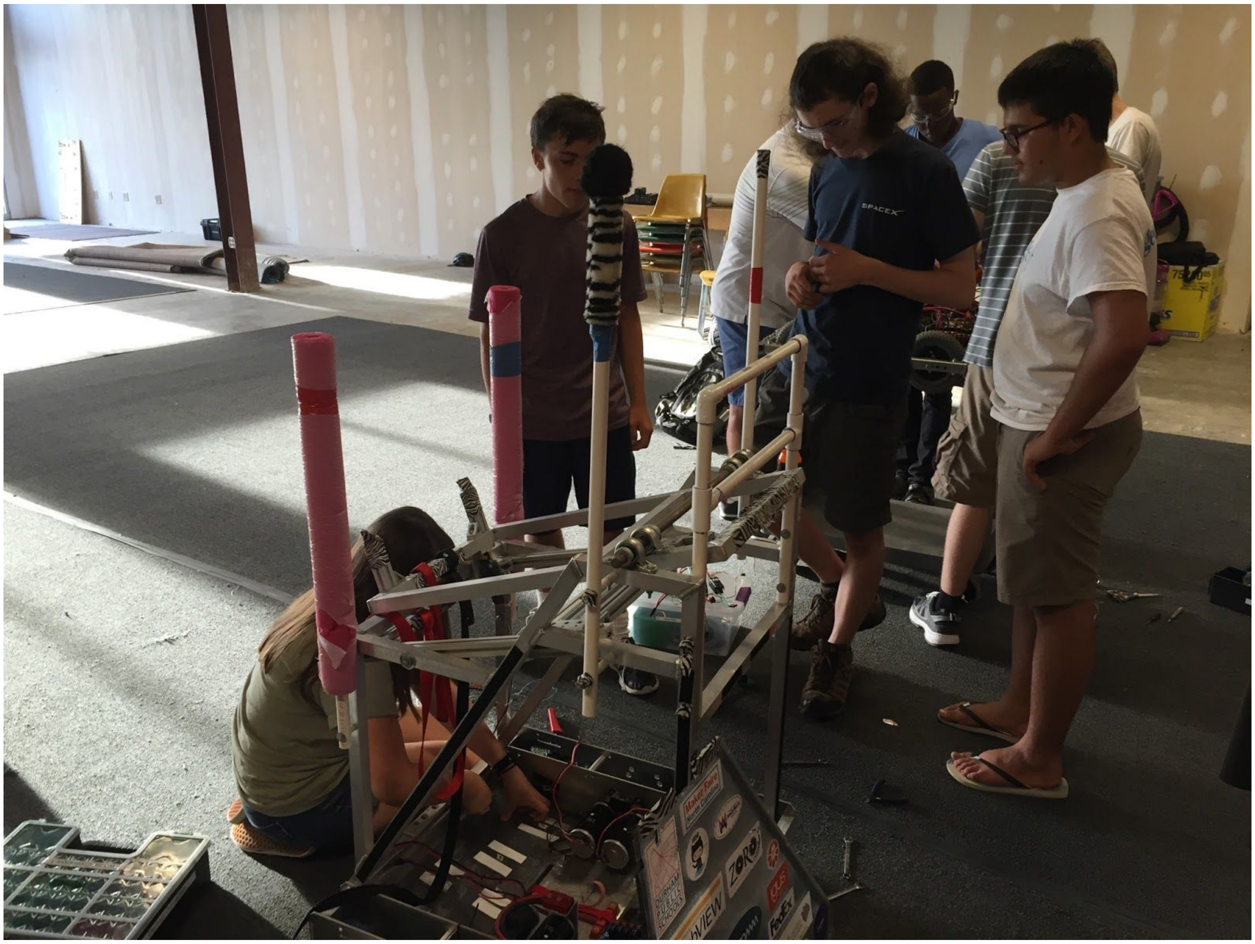


Image 7: Highway project. One person works while everyone else watches...

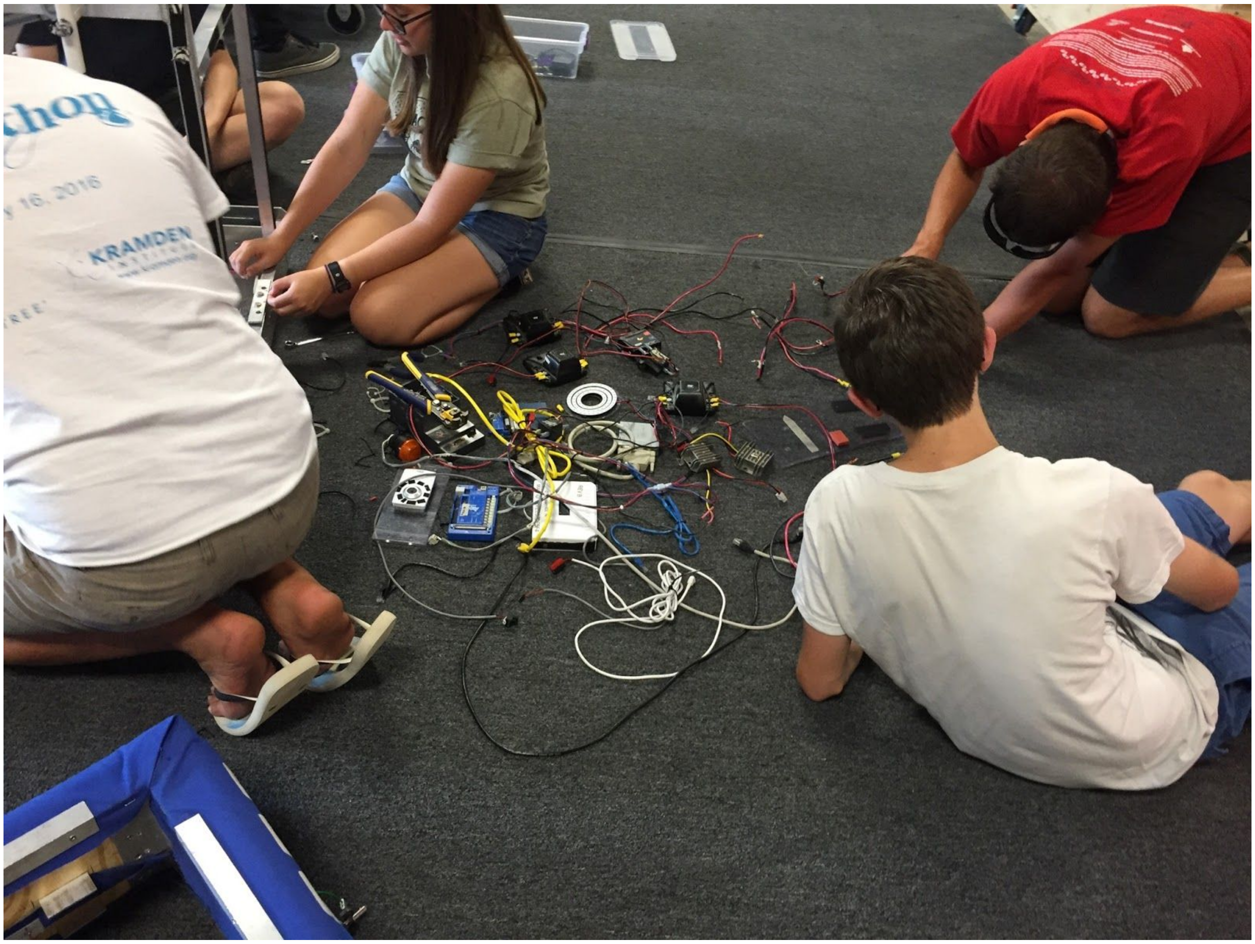


Image 8: Alas poor Hot Sauce! We knew her, Patricia[3].

In With the New

We anxiously awaited the arrival of our new Talons from CTRE. We worked on some code while we waited:

[3] This is both a literary reference along with an inside-joke for our team. Patricia, we miss you and hope you are doing well.

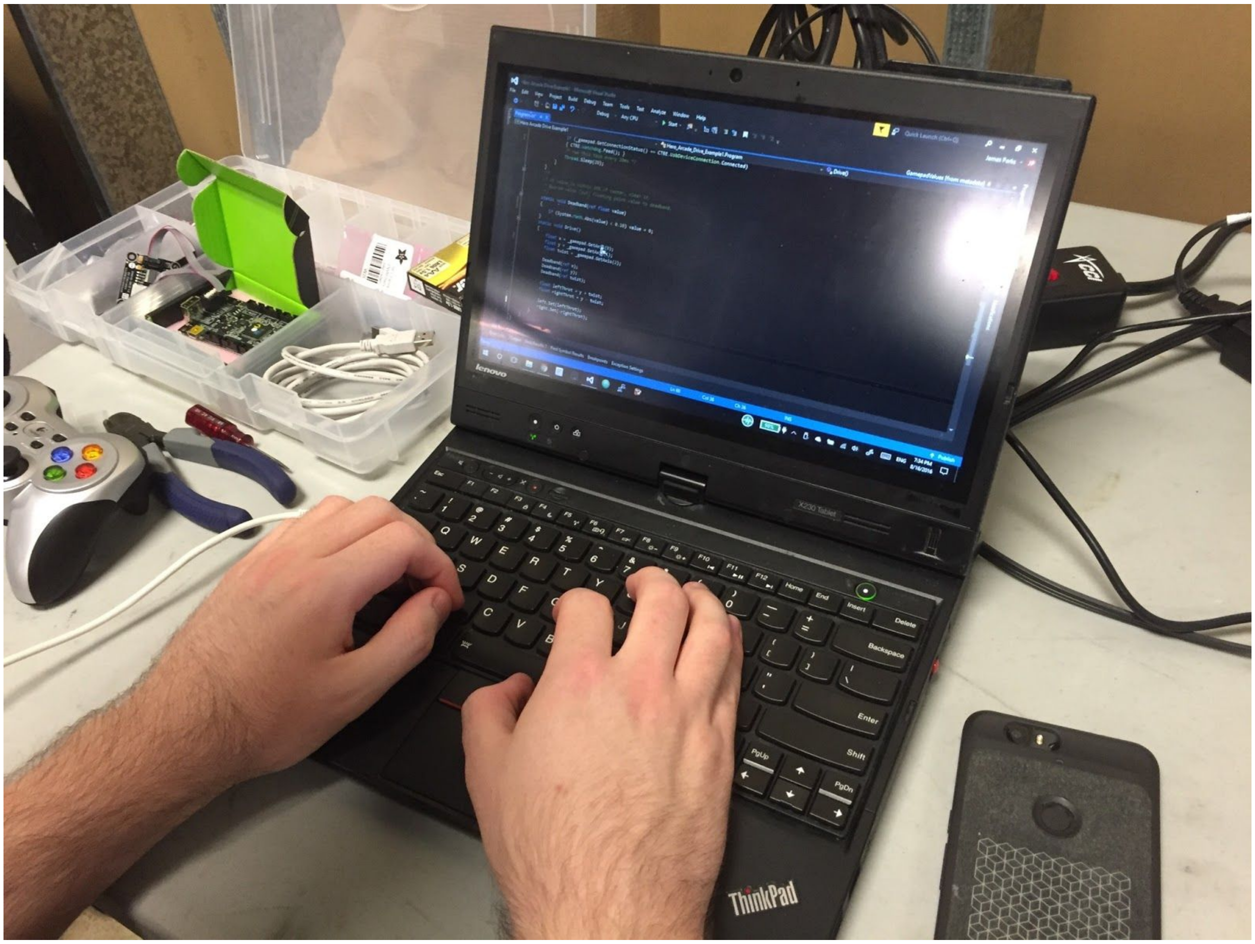


Image 9: Coding away.

Talons are a lot more compact than the Jaguars. Not to mention that they offer up more features, better performance, are completely sealed, and they look good too:

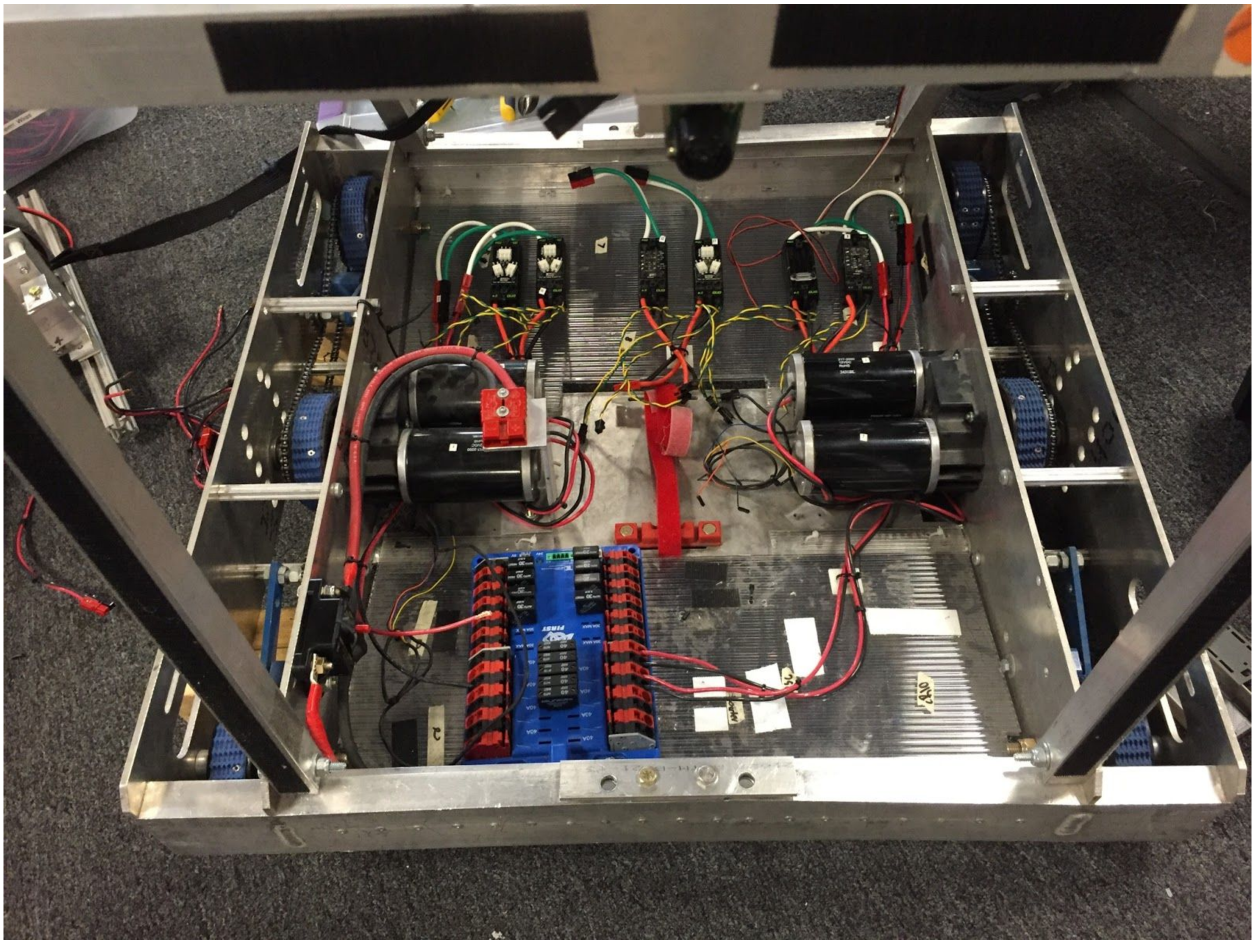


Image 10: My, what tiny Talons you have!

Once we had the motor controllers on the robot, it was a matter of tying all of the components to the power distribution system. Since the cRIO, Radio, 2CAN, and Camera had been removed it was a much cleaner wiring job:

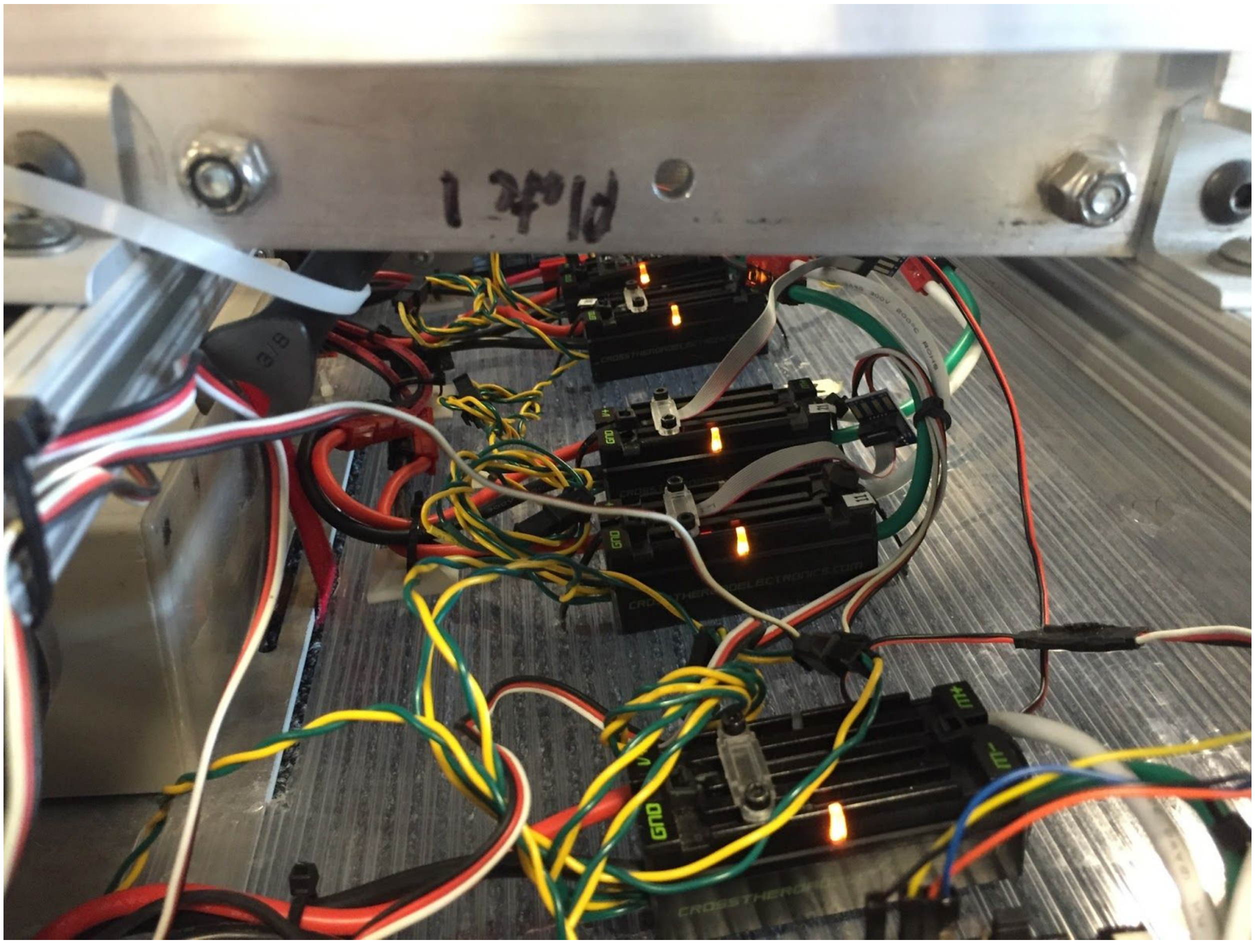


Image 11: Shiny lights!

The HERO board is powered off a 12V source so replacing the cRIO was quite simple and it could even use the same 12V power port.

Once the power wiring had been completed it was time to work on the wiring for the control pathways. This means you can wire CAN if you try.

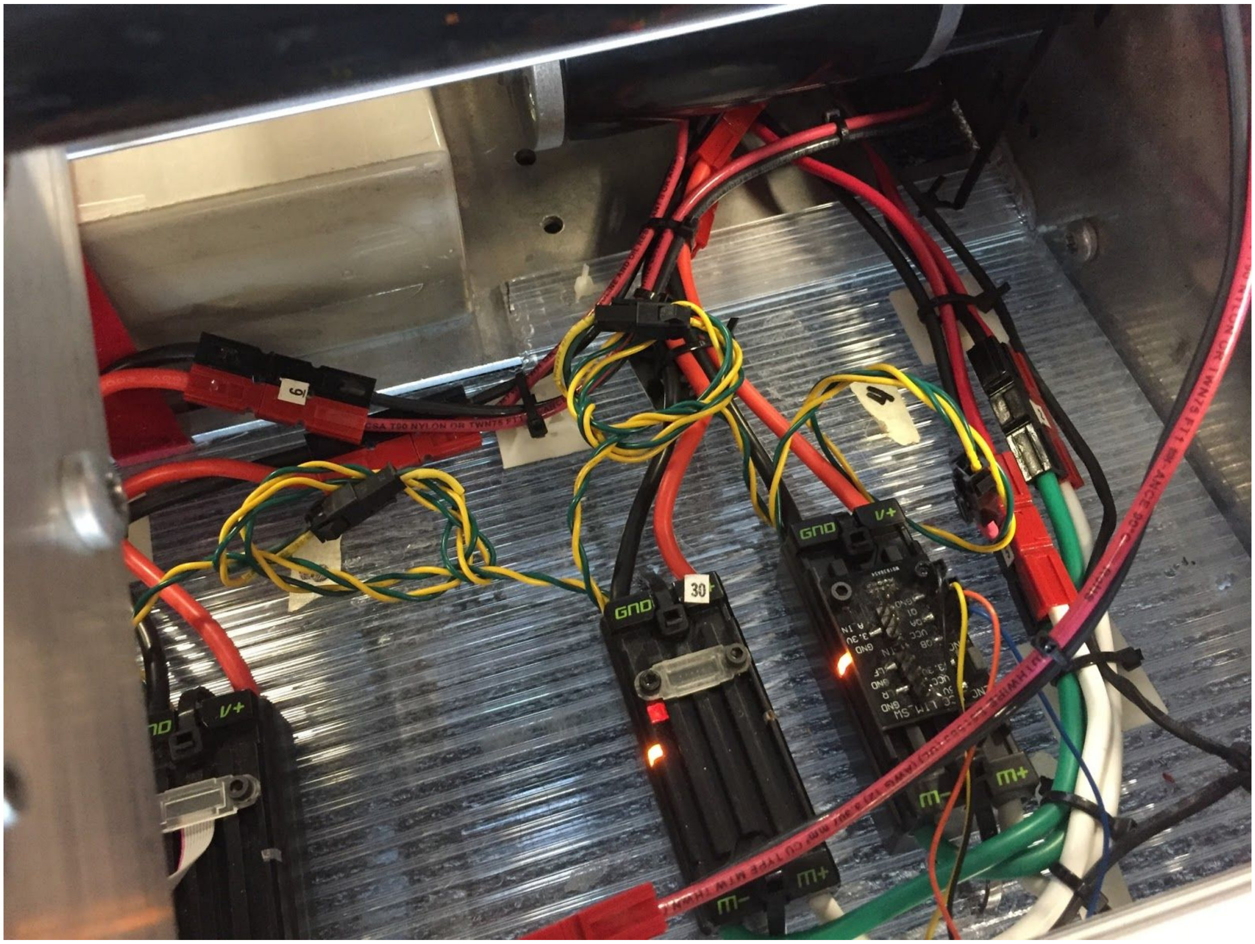


Image 12: Look closely and you can spot the terminating resistor on the end.

The HERO has a built-in 120Ohm terminating resistor for the CAN bus so it was one endpoint of the system. It was connected to the Talon SRXs that were then terminated on the far side with another 120Ohm resistor. This resistor was required to complete the CAN bus properly.

Protecting Our HERO

We designed a case for our HERO board. It’s not perfect but it gave us a chance to try out OnShape for designing robot parts. Here is a screenshot of the case in OnShape:

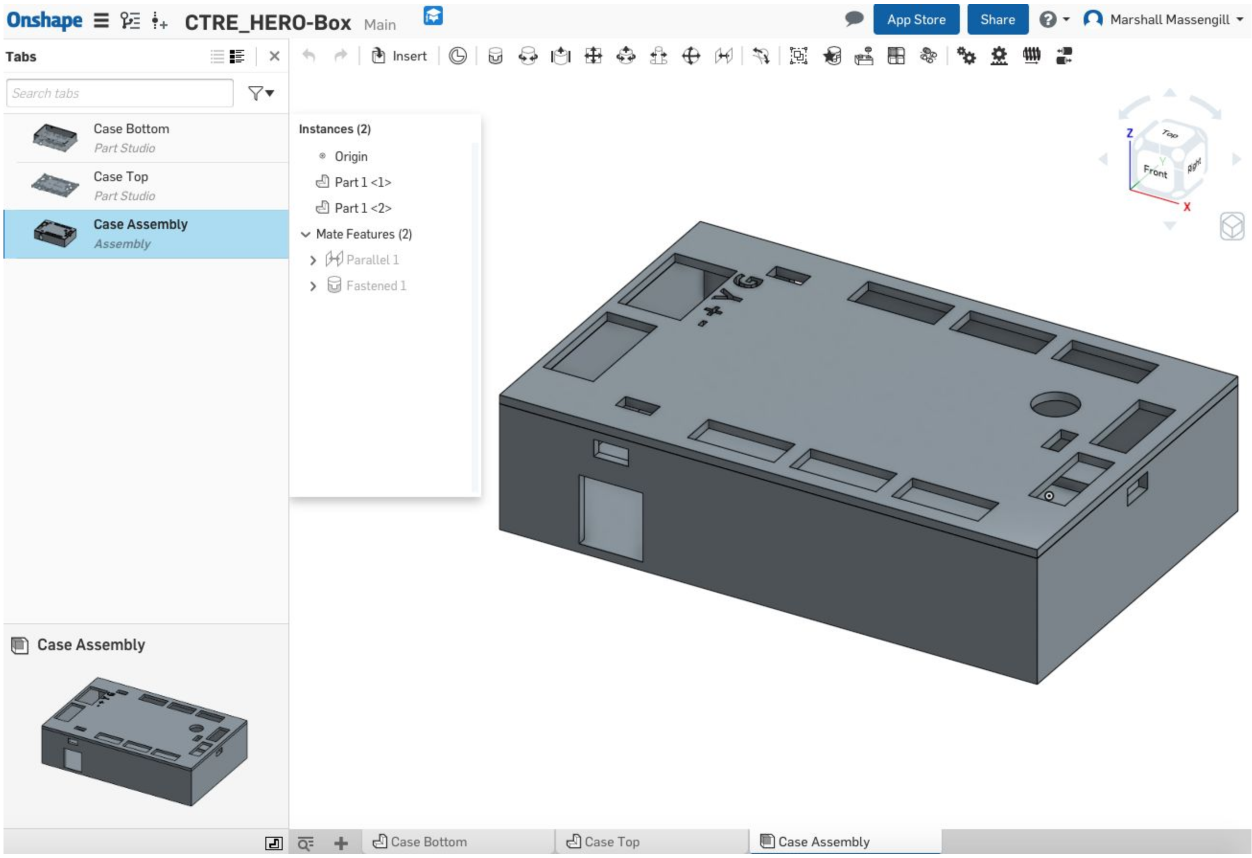


Image 13: The Case in OnShape

You can get a copy of the case via OnShape here: https://cad.onshape.com/documents/2c322ff1002abb330b7c6c01/w/2861d574639c6da318e713d4/e/d7908d028499d59a58c127af

We then printed our case using a 3D printer and PLA filament. This is what the case looked like mounted to the robot:

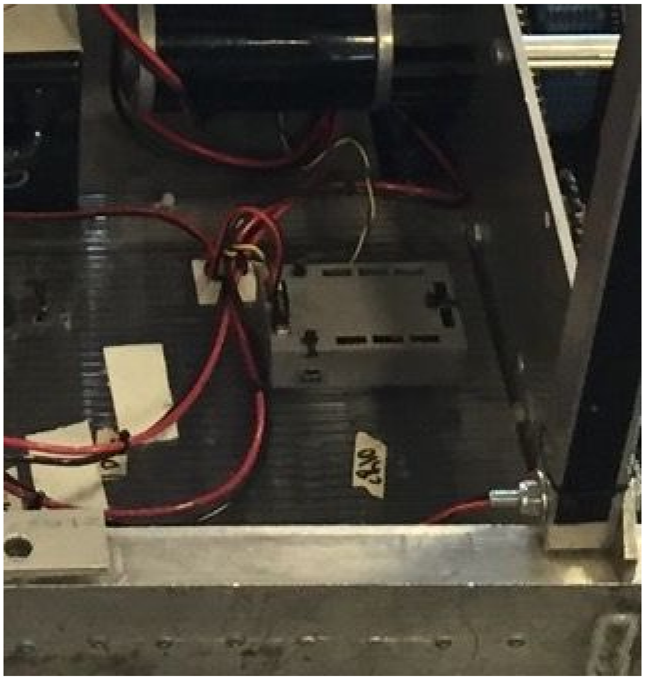


Image 14: The Case in Action… look at it not move.

As our season ramps up this year we are going to have a student design and cut a new case for the HERO out of acrylic on a laser cutter. This will showcase the HERO board as well as provide an opportunity for a student to complete a project on their own. Additionally, we will add conformal spray coating to the HERO once we mask out the headers to add some additional protection.

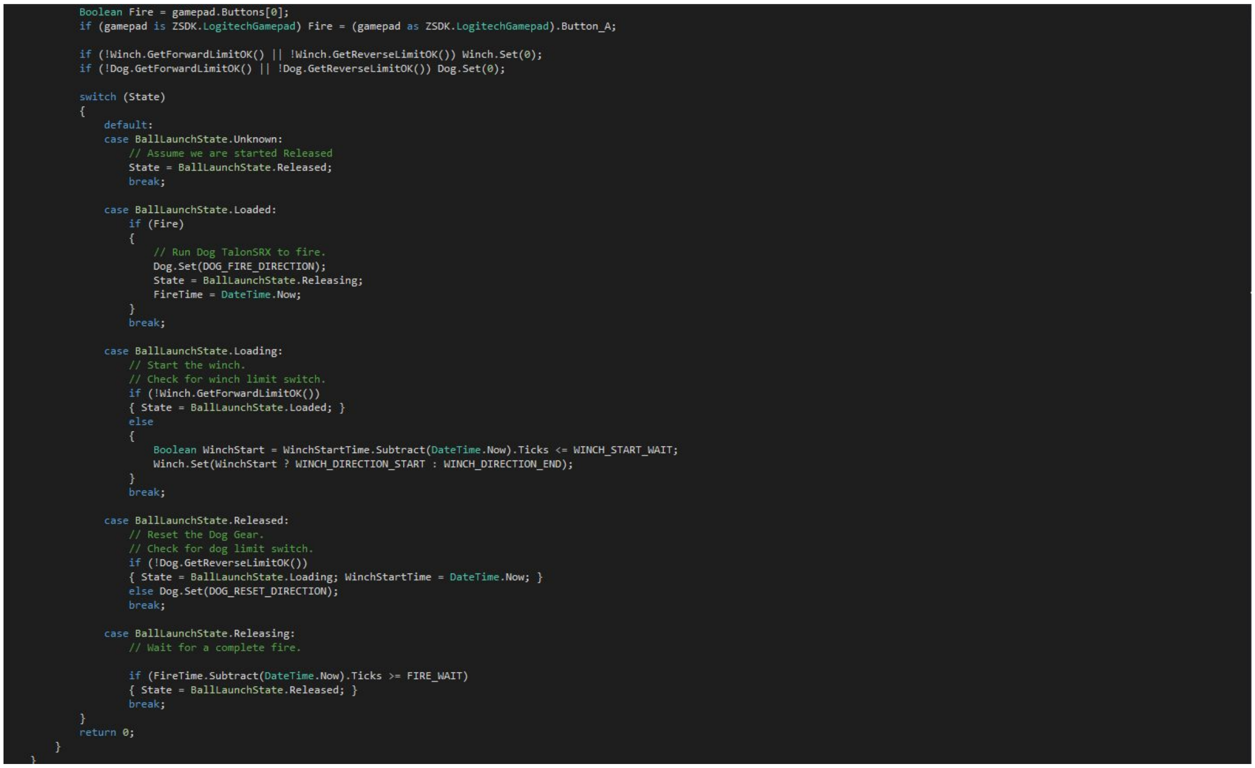
Programming the HERO

Teams that use Java will probably have the easiest time using a HERO board, followed by C++ teams then LabVIEW teams. Since we were new to the CTRE-HERO, and wanted to test some of the experimental features in the future, we decided to use the 4.4.1.0 beta release of the SDK.

Due to the limited time before an upcoming demo, this iteration of the code was written by one of our mentors who was fluent in C#. After taking an hour or two to read the documentation and some of the sample projects, we were able to have the robot driving around using %VBus in under 15 minutes. The next step on our conquest of updating our robot was to re-implement the functionality of our mechanism. We accomplished this through the use of a state machine of 6 different states. We also decided to write a simple module class and hero class to make development easier and allow the breakout of the robot’s components.

The biggest headache we ran into was the .NET Micro Framework itself as it lacks some of the niceties of its big brother on the desktops/servers.

We would like to see wireless network support in the future for a better safety option. At the moment the recommended safety procedure is to use a wireless Logitech gamepad F710 and use the input mode switch on it to enable and disable the robot.



```
        Boolean Fire = gamepad.Buttons[0];
        if (gamepad is ZSDK.LogitechGamepad) Fire = (gamepad as ZSDK.LogitechGamepad).Button_A;

        if (!Winch.GetForwardLimitOK() || !Winch.GetReverseLimitOK()) Winch.Set(0);
        if (!Dog.GetForwardLimitOK() || !Dog.GetReverseLimitOK()) Dog.Set(0);

        switch (State)
        {
            default:
            case BallLaunchState.Unknown:
                // Assume we are started Released
                State = BallLaunchState.Released;
                break;

            case BallLaunchState.Loaded:
                if (Fire)
                {
                    // Run Dog TalonSRX to fire.
                    Dog.Set(DOG_FIRE_DIRECTION);
                    State = BallLaunchState.Releasing;
                    FireTime = DateTime.Now;
                }
                break;

            case BallLaunchState.Loading:
                // Start the winch.
                // Check for winch limit switch.
                if (!Winch.GetForwardLimitOK())
                { State = BallLaunchState.Loaded; }
                else
                {
                    Boolean WinchStart = WinchStartTime.Subtract(DateTime.Now).Ticks <= WINCH_START_WAIT;
                    Winch.Set(WinchStart ? WINCH_DIRECTION_START : WINCH_DIRECTION_END);
                }
                break;

            case BallLaunchState.Released:
                // Reset the Dog Gear.
                // Check for dog limit switch.
                if (!Dog.GetReverseLimitOK())
                { State = BallLaunchState.Loading; WinchStartTime = DateTime.Now; }
                else Dog.Set(DOG_RESET_DIRECTION);
                break;

            case BallLaunchState.Releasing:
                // Wait for a complete fire.

                if (FireTime.Subtract(DateTime.Now).Ticks >= FIRE_WAIT)
                { State = BallLaunchState.Released; }
                break;
        }
        return 0;
    }
}
```

Image 0b1111: A screenshot of the start of the ball state machine.

Great Success!

The project was a success despite some minor setbacks with accomplishing everything we had hoped to do in a shorter timeframe. Here is the 2014 Zebracorn Robot alive and well once again:

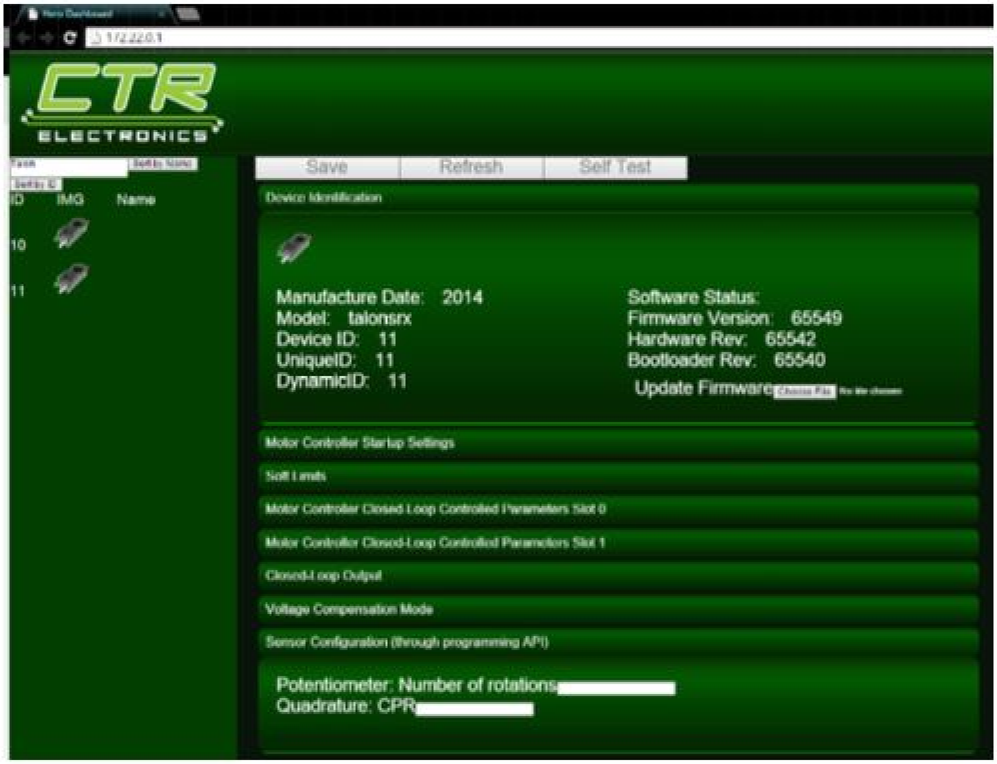
Image 16: It’s kinda hard to see in the background but students were learning about it and we didn’t have the heart to ask them to move.

Image 17: One final picture of the robot in our new space

Lessons Learned

After we had the robot working the first thing we noticed was that our demonstration bag was a little lighter since we no longer needed to carry a driver station laptop with us to our demos. This is really nice since we don't have to worry about the laptop being taken by someone or powering it. We do need an extra set of AA batteries for the wireless controller though just in case.

Wiring the limit switches for the mechanism took some time and patience. It was not a difficult task but we wanted to ensure it was done correctly and that our robot still operating the same way it did with the cRIO. There was not an easy method to test if a limit switch was triggered by the Talons so we wrote a bunch of debug code and just methodically worked the problem. It was a bit more of a brute-force solution than we would have liked but it worked out well. We are hoping that the next release of HERO software includes the new web interface, which should make this kind of work easier:


1/72.22.0.1
CTR ELECTRONICS
Save
Refresh
Self Test
ID
IMG
Name
10
11
Device Identification
Motor Controller Startup Settings
Soft Limits
Motor Controller Closed Loop Controlled Parameters Slot 0
Motor Controller Closed-Loop Controlled Parameters Slot 1
Closed-Loop Output
Voltage Compensation Mode
Sensor Configuration (through programming API)

Manufacture Date: 2014
Model: talonsrx
Device ID: 11
UniqueID: 11
DynamicID: 11

Software Status:
Firmware Version: 65549
Hardware Rev: 65542
Bootloader Rev: 65540
Update Firmware

Potentiometer: Number of rotations
Quadrature: CPR

Image 17: HERO Web Interface

The Zebracorns are part of the cult of the orange safety light, or robot signal light as the FRC elders refer to it. It's just a 12v Allen-Bradley industrial safety LED, but it adds an obvious visual notification that the robot is powered on and enabled. It's also a competition requirement for FRC robots. We rely on this light for safely demonstrating our robot so it needed to be in any new control system. The good news is that there will be a way to do this in the future (more on this later).

The bad news is that we had to implement a sort of hack to make the light work for now. To make it work for now we simply removed the jumper that FRC has teams place on the light and then wired it directly to 12v power. This means that when the robot is powered, the light will flash. This is not ideal operation as we would like to have the light flash when the robot is "enabled" and stay solid when the robot is just powered up.

Continuing on with the safety theme, because there is no Driver Station, there is no simple method to disable the robot so we used the input switch on the back of the Logitech controller. We have code that looks for the input mode and will quickly disable the robot when switched off.

It works very well, and despite not having an emergency stop, we believe this is safe enough for performing demonstrations with our robot.

Next Steps

As we mentioned, we needed to have this robot working in a hurry for a demo so one of our mentors wrote the code to use %Vbus rather than closed-loop PID. Our aim is to have students re-implement the drive code using the encoders as soon as we can get them settled in for the year.

Along with getting the encoders functioning, we will need to switch over our encoders to the CTRE breakout boards. We have experimented in the past with our own breakouts and though we aren't the biggest fans of ribbon cables, we have to admit that they work really well with the Talon SRXs and we've finally been won over by their convenience.

We would like to eventually replace the old FRC Power Distribution Board with a new CTRE Power Distribution Panel. This will allow us to extend our CAN bus to monitor the power on the robot and we might even use a Gadgeteer LCD panel to display some of that information.

As mentioned above, we will be redoing the case for the HERO in acrylic so we can showcase the board. Along with that we will mask out the headers and spray the board with a conformal coating to make it a little more resilient.

Lastly, we really want to get our safety light working. This will mean we need to write some code for it and interface it with the HERO. Fortunately, CTRE has recently released a driver module for the HERO that will allow it to control high current loads like our light or solenoids:

Image 18: Driving with a HERO

Thank You

A huge thank you to CTRE for sponsoring this project and providing us with the resources we needed to complete it. We're very grateful for your support!

The Zebracorns ❤

CTR

ELECTRONICS

Image 19: Our HERO!

Resources

Grab the latest HERO SDK, Documentation, and Programs from here:
http://www.ctr-electronics.com/hro.html#product_tabs_technical_resources

Our source code is available here:
https://github.com/FRC900/Aerial-HERO