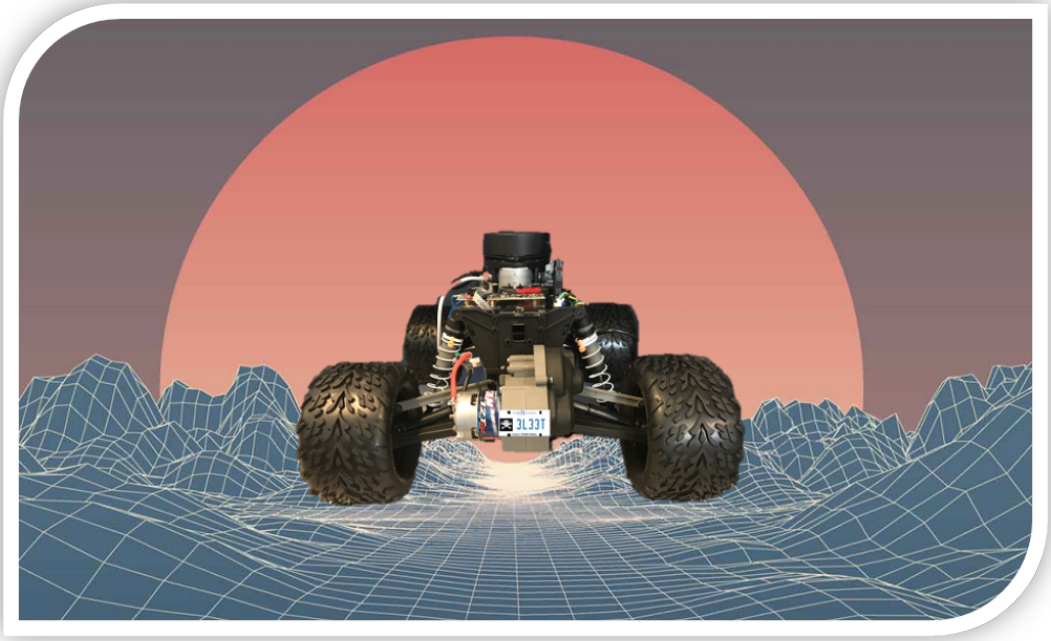
Build your own RoboCar

29Bit CANBus at 1/10th scale

Sean McKeever - spmckeever@gmail.com - @d0rkv4d3r



CONTENTS

CONTRIBUTORS

MKI Team

Shawn Nicolen @mortedamos - Design/Code/Help
Isabel Barbosa - Code/Build/Awesomeness
Jim Zondag - Prototyping/Fab/Crazy Ideas

MKII Team

Matt Bender - Code/Build
Isaac Hermann – Code/Build
Mike Donahue - Code/Build

THANKS

Cross The Road Electronics for their awesome products and killer support!
Killer Bees FRC 37
Car Hacking Village at DEFCON
@ac0rn
@Surge
@kateo
@Issac
@jsits
Java Hutt

WHAT IS A ROBOCAR?

The RoboCar is a 1/10th scale electric car with a 29bt CANBus. From inception they were designed to use commercially available hardware with open-source software so that they could be built by anyone. The name is not trademarked, we didn't bother to Google it, so there are other projects using the same name. The name stuck, and it's too late to turn back now. The core of the platform combines First Robotics Competition[1] components with a Raspberry Pi, and an open hardware CANBus device to replace the traditional radio and speed controllers of an R/C car. The cars are driven via CANBus injection from an SSH session over WiFi. Turing a fun toy into a fun demonstration of bad UX design. Feel free to skip the rest of this and go to chapter two if you want the hardware details.

Why would you do this? Most articles and guides[2] on CANBus and car hacking require real cars, or many trips to the junkyard to get vehicle components. The RoboCars are inexpensive (compared to a real car) and small enough to not really hurt anyone. They are an open platform so you can demo any new CANBus related hardware or software, and can help people learn CANBus in a safe controlled manner. The platform is slowly trudging towards autonomy, and my final goal is to have them follow me around on their own (Feel free to show me up and do this first).

BILL OF MATERIALS - WHAT DO YOU NEED?

Before we dive in, a warning. I am a hack, not an engineer. Everything documented here is a best effort, and has resulted in letting the magic smoke out of lots of stuff. If something seems wrong, it probably is; I welcome any feedback, corrections, and name calling resulting from trying to follow my instructions.

The RoboCars are based on 1/10th Sale R/C cars, brushed motor cars are both cheaper and better suited to low-speed operation. Brushless motors don't like running at low revolutions; causing stuttering, and will eventually fail if regularly run at low speeds. To keep this simple, we started with complete ready to run cars, so we can test them first and know that we have a solid vehicle to start building on. You do wind up buying more than you need, but can use the leftovers for other projects.

I'll try to explain how each component is used, and what it is wired to. The hows of packaging and wiring is left as an exercise for the reader/builder, but some reference photographs and diagrams are included for help. There are certainly other ways to do this, the goal of the project was to purchase everything, and focus on building a working prototype. Since the MKI cars were built in 2016, there are a couple new boards that could handle what our Raspberry Pi + CANTact(or CANable) accomplishes, but we already had a working solution and stayed with what worked.

The Bill of Materials

NOTE: Links are for reference only and do not indicate any endorsement from me or permission from them. IANAL, YMMV, Etc.

RoboCar MKII BOM		
Item	**Price EA**	**Notes**
Raspberry Pi 3 Model B	$35.00	Upgrade from Pi Zero to more RAM and speed
Raspberry Pi Camera Board V2	$29.95	Cameras for RoboCar
Camera Case	$2.95	Case to protect Camera - Need new case for V2 camera
CANAble	$25.00	Provide CAN injection
CANifier	$39.99	Provides lights via CAN Bus
Hero Development Board	$59.99	Brains of the RoboCar
Talon SRX	$89.99	Provides DC Motor Control and CAN Bus for RoboCar
Voltage Regulator Module	$44.99	Voltage Regulator for steering
Gadgeteer Breakout Module	$5.99	Provides PWM for Steering (5pack)
Anderson Powerpole Red	$0.32	Interconnects for wiring
Anderson Powerpole black	$0.32	Interconnects for wiring
Anderson Powerpole Contact #10-14 AWG	$0.45	Large gauge terminals
Anderson Powerpole Contact #12-16 AWG	$0.21	Medium gauge terminals
Anderson Powerpole Contact #16-20 AWG	$0.20	Small gauge terminals
Weidmuller Ferrule - 18 AWG	$3.99	PCB Connections (pack of 20)
SRX/Gadgeteer Data Cable Kit	$29.99	Make custom Data Cables
Traxxas Stampede 1/0th Scale R/C Truck	$215.99	The base of the Robo Car
3000 mAh MiMH 8.4V Battery cell	$79.98	Extra batteries
4MM connector terminals	$8.69	For motor connection to Talon
Traxxas AC to DC converter	$24.16	Charge Batteries
	Total	$698.15

The Car

The Traxxas Stampede, even in its least expensive form is capable of 35 miles per hour. This is very fast for something driven from a shell prompt. This was solved in software by limiting the car to 17% throttle, which at a full charge is a brisk walk or jog to catch a runaway car. This is still plenty fast, but mostly controllable. Additionally, the Stampede is prone to wheelies, which

is fun; but potentially catastrophic with exposed electronics. To reduce the chances of flipping the car over under sudden acceleration, the batteries were mounted underneath the chassis, and suspension adjusted to its lowest possible setting. While the car is still capable of wheelies, these modifications seem to have solved the issue.

Testing

This part of the build is the easiest, unpack your car, and charge the battery. Throw some batteries into the radio and give the car a test drive to make sure everything is working before you start tearing it apart. It will save your sanity later to know that your motor and steering servo work as they should, and that the steering is centered properly. Pop some wheelies, do some donuts, and enjoy; then tear it apart. You will need to remove

1. The Electronic Speed Controller
2. The Radio Receiver
3. The battery mounts

Fab Work

Give the chassis a good going over, and start figuring out where you'll place all your new components. The chassis won't hold everything as-is, so you'll need to fabricate an additional surface to mount a lot of the new hardware. (See build notes for examples) For the MKI cars, metal desk drawer dividers were pilfered, insulated with foam core, and zip-tied to the top of the chassis across the shock towers, for the MKII cars, we used Lexan Polycarbonate cut to shape. You should be able to find this at any hardware store. The original body mounts are excellent quick-release mounting points for this, our design opted to only use the rear mount, and small zip ties to secure the front.

Once you've planned out where to mount your components break out a drill and start making holes for everything. Be creative, brass motherboard standoffs plus screws are excellent ways to securely mount things, and will add rigidity to your Lexan plate should you opt to use one. Small Zip ties can also provide solid mounting for odd shaped components like the Talon SRX, just make sure you've got plenty so you can remove things and re-install them as needed.

NiMH Battery

The Traxxas batteries come with high-quality keyed connectors that do a good job of ensuring connections aren't shorted and polarity is correct. I'd recommend sourcing originals from Traxxas, which should be available from your local hobby shop for a few dollars each. I tried some off of Amazon, that looked to be OEM, and they did not work out well. The battery connection will need to power the Talon SRX directly, along with the VRM, be careful soldering all of this together as it is easy to make a connection that won't fit into the terminals when you're done.

The FRC Components

Hero Development Board

This contains the C# code that translate CANBus commands into throttle steering and any other actions you'd like to program into your RoboCar. When mounting it, be sure it's protected from

impact, and easy to reach for wiring purposes. All of the FRC devices need to talk to this board, so be sure to mount it somewhere you can access easily.

Talon SRX

This replaces the Electronic Speed controller, and provides the CANBus for the RoboCar. This is the only component that is location dependent, as the wires aren't very long and must reach the drive motor. I recommend using the same barrel connectors the Stampede originally used, which should be available at your local hobby shop. There are four sets of wires on the Talon, the motor drive wires (White and Green) the power wires (Red and Black) connect to the battery; the green and yellow wires are CANBus leads which connect to the Hero and CANBus interface.

CANifier

This was included to control the LED Headlights, and should be able to activate any other device via the CANBus. We did not get this working in the MKII build, but is included here as an option for CAN activated devices.

Voltage Regulator Module (VRM)

This provides power to all the FRC components of the car, and is wired straight to the NiMH battery along with all the FRC Components

Gadgeteer Breakout Module

This provides the Pulse Width Modulation (PWM) control for the steering servo. It wires to the Hero via the SRX/Gadgeteer Data Cable, which is a small ribbon cable I typically mount it close to the hero, and use secure quick connect to wire it to the steering servo.

CANBus Interface

CANTact or CANAble:
The CANTact, and its mini clone the CANAble "is an open source Controller Area Network (CAN) to USB interface for your computer." It's wired directly into the CANBus from the Talon, and connected to the Raspberry Pi via a USB cable. Details on setting this up can be found under “Chapter 3. How to make this Stuff Work”

Raspberry Pi

Raspberry Pi 3 Model B(or 3B+):
The RoboCars need an interface, and that's where the Raspberry Pi comes in, we used RPi 3Bs for this project but the installation has been tested on a 3B+. The MKI cars used a RPi ZeroW, which has enough horsepower but limited USB connections. For ease of installation, and processing power a 3B or 3B+ is recommended. In this design the Pi is powered from a USB battery typically used for phone and tablet charging. The VRM is slightly under voltage for this task, and having a separate battery for the Pi allows car-side battery swaps without shutting down the Pi. A 6000mAh will run the Pi3B for at least 4 hours.

In the MKII design the Pi is mounted at the very front of the car on the Lexan plate, to allow the camera cable to reach the front mounted camera. Aside from power and camera, you'll also need a USB cable to connect you CANBus Interface card (Cables vary based on your specific model) shorter is better here, as long wires can dangle and get snagged easily. NOTE: There is such a thing as POWER ONLY USB A to OTG cables, especially the short cables provided with phone charger batteries. Make sure your cable can send data if you need it to, or you'll waste a week trying to figure out why your CAN Bus Interface isn't working on your Pi but is on your computer. Or just use the same cable for both use-cases :)

Raspberry Pi Camera Board V2:
Possibly the most straightforward component in the RoboCar, the Camera provides a front facing camera for navigation. Find a safe spot to mount it to the chassis, I used a "Raspberry Pi Camera Case" to protect these, but wound up with a case for a V1 camera, not a V2. The lens is different, so the hole in the case had to be widened with a small drill, but it's easier to mount than the tiny screws the camera board uses. You may also want to purchase a longer camera cable, so that your Pi can be mounted in a different location, the location we used stretched the cable tightly over the GPIO pins, which will eventually tear the cable and kill the camera connection.

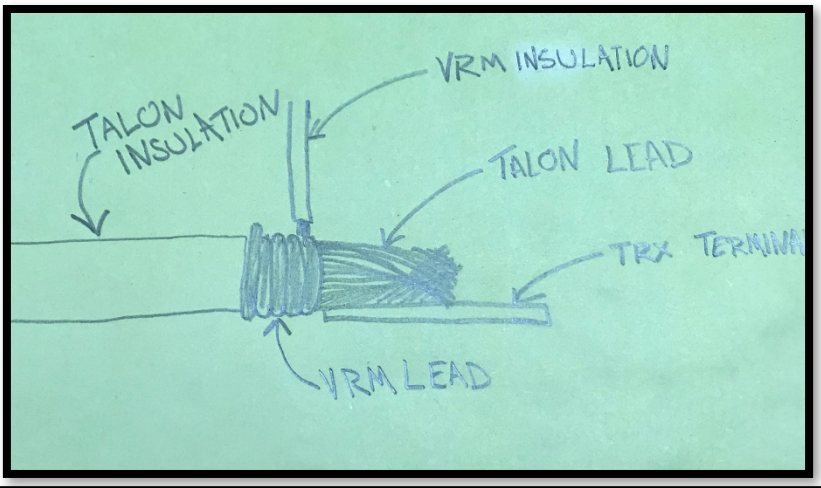
Anker 10000mAh battery:
While the VRM may have enough power to run the Pi in addition to the rest of the RoboCar, it's at the bottom end of what the Pi needs, and motors kicking on and off may cause brownouts for the Pi; so we just use a separate inexpensive USB battery to power the Pi. The Anker is just a suggestion, we are using a different battery that is only 6000mAh, but I don't have a handy link to them anymore. Anything similar will work, and of course higher mAh ratings will provide longer run times. The only constraint is size. This wires direct to the Pi's power connection via a USB A to OTG cable. The 6000mAh battery will keep a Pi3B going for over 5 hours.

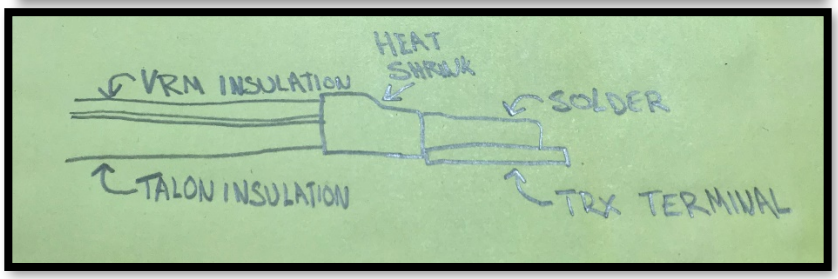
Wiring the RoboCar

The VRM is wired to almost everything, CTR has wiring size recommendations and they are worth following. I start with the Talon power and motor wires, as they are the largest and require the most heat. Always be sure to insulate any exposed connections with heat shrink, and take your time to get solid solder joints.

Wire the white and green wires to the drive motor of your car, the polarity is labeled on the talon, and should be labeled on the motor, or in the car's instructions. As previously noted, try and use the same connections as the manufacturer used for the speed controller, or replace them entirely with different connectors, while you can certainly just solder the wires to each other, it makes troubleshooting and field replacement much harder. We did this on the MKI cars, don't be like us.

Once the motor side of the talon is wired, you'll need to wire up the power side with a battery lead. Buy extra connectors because it may take a few tries to get this right. This connection has to go to both the battery and the VRM, which is usually a tight if not impossible fit for the battery leads. I wound up stripping back about twice as much insulation on the talon wires, and attaching the VM leads with extra bare wire to spare. This leave the original wire diameter at the end which is just small enough to fit inside the TRX leads from Traxxas. The following illustrations may help.





Once you've got the Talon connections done, route the wires you added for the VRM to the **12V In** leads.

Next you'll need to wire up the Hero Development board to the **12V2A** output of the VRM, these attach to the **V+ V-** on the Hero. This is also a good time to attach one set of the CANBus leads from the Talon to the Hero.

The Steering Servo will also need to be connected to the VRM on the **5V2A** output, to do this, it's easiest to just remove the original connector from the three wires coming from the server. Red and Black are what you'd expect them to be. These wires are much smaller than the VRM terminals are designed for, so get creative with making some new leads (I just used some of the other wire and heat shrink). The White wire on the servo is the PWM signal line, which gets soldered to line 7 on the Gadgeteer Breakout Module. The module also is wired to the PY port on the Hero with a CTR Ribbon cable. I make my own with the CTR kit to avoid having excess ribbon cable to deal with.

Wire up your CANTact or CANAble to the second set of CANBus leads from the Talon, connect that to a USB Port on the Pi, add a power line for the Pi and your RoboCar is ready for code!

HOW TO MAKE THIS STUFF WORK

Once you've done all your fabrication, mounting and wiring work, you're ready to start getting this working.

Hero Setup

The Hero needs code, and that is located in RoboCarMKII.txt This will establish your Arbitration IDs for the CANBus and how the RoboCar will respond to them. You can find this code in the GitHub repository.

Raspberry Pi Setup

Setup the Pi with vanilla Raspbian or NOOBS following your favorite setup guide, and get it connected to WiFi. You'll need to enable SSH, and the Camera in Raspbi-config then proceed with getting the CANBus device working.

Before using the CANTact or CANAble, you'll need to update them to the CandleLight Firmware, available here: https://wiki.linklayer.com/index.php/CandleLightFirmware

The CANBus setup is based on work documented here: https://github.com/mortedamos/vehicle-hacking/wiki/Vehicle-Hacking-Setup-Guide:-Part-2:-Physical-CAN-Interface [3]

1. Enable CAN in your kernel with
   `sudo modprobe can`
2. Plug in your device, and identify what port has been assigned to it with the dmesg command. Output will look like this:

```
[  844.758401] usb 2-2.2: new full-speed USB device number 5 using uhci_hcd
[  844.865131] usb 2-2.2: New USB device found, idVendor=ad50, idProduct=60c4
[  844.865135] usb 2-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber
=3
[  844.865137] usb 2-2.2: Product: CANtact b20
[  844.865138] usb 2-2.2: Manufacturer: CANtact
[  844.865139] usb 2-2.2: SerialNumber: 00000000001A
[  844.908111] cdc_acm 2-2.2:1.0: ttyACM0: USB ACM device
```

3. Create a serial link to your CAN device with slcand
   `sudo slcand -o -c -s8 /dev/ttyACM0 can0`
4. Bring the new interface up and make sure it's working
   `sudo ip link set up can0`
   `ifconfig can0`

If ifconfig outputs anything other than an error message your device is working. No packets have been transmitted yet, so you will only have Tx packets if anything.

5. Adjust the Tx buffer size, as our 29bit frames will fill it very quickly. This is done with the following command
   `Sudo ip link set can0 txqueuelen 80000`

FUTURE PROJECTS

For our MKII cars we added LiDAR, cameras, Lights and lots of hopes and dreams. Version two of this guide will include details on this hardware and the setup (that we managed to complete)

REFERENCES

[1]Cross The Road Electronics GitHub
https://github.com/CrossTheRoadElec

[2]The Car Hacker's Handbook
http://opengarages.org/handbook/

[3]Shawn Nicolen's full Vehicle-Hacking guide
https://github.com/mortedamos/vehicle-hacking