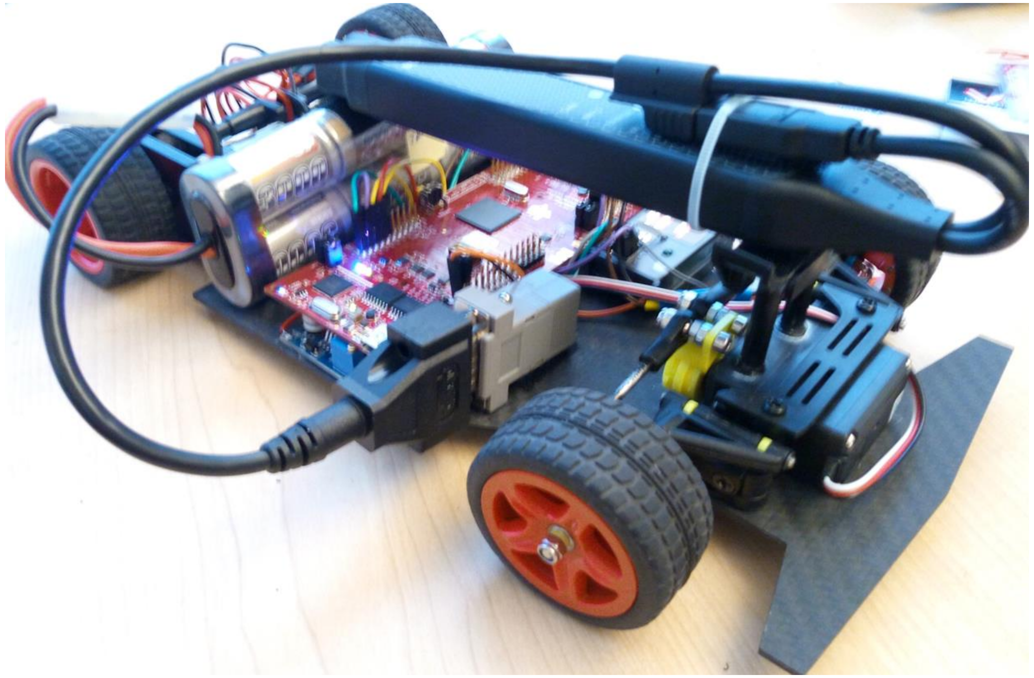
Remote Control Racecar Demo
Detailed Hardware Integration

By Jose Avendano, Christoph Hahn

Technical Objectives

- Traction Control
- Torque Vectoring
- Wireless CAN communication
- Use NXP cup car chassis



NXP Chassis Hardware Reused

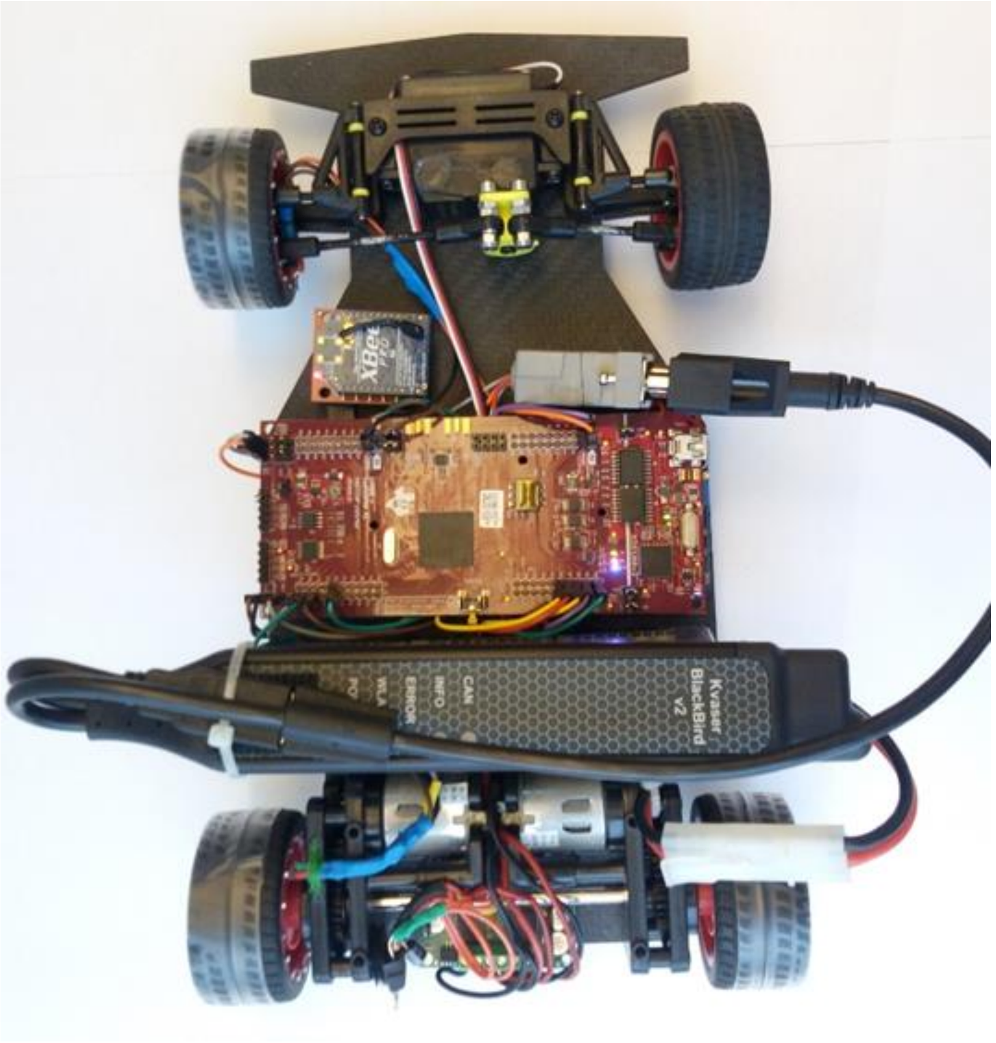
- NXP Cup chassis
  - 1x Futaba S3010 Servo motor
  - 2x DC motors (Standard motor RN260-CN-18130 7.2V, 3.8A)
- Battery – Conrad energy 7.2V 3000mAh

Additional Hardware

- TI C2000 Launchpad XL MCU
- Kvaser Blackbird V2
- Pololu Dual MC33926 Motor Driver
- RC Receiver
- Assorted power components

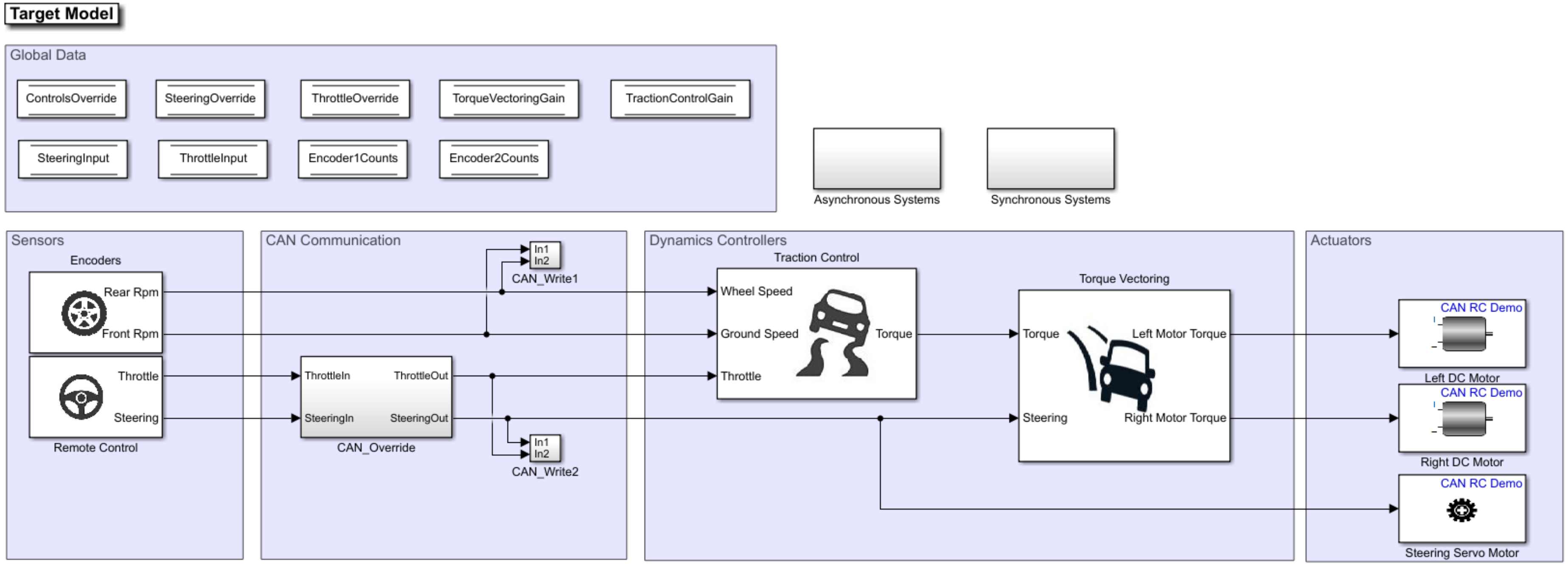
Peripherals Interacting with C2000 MCU

- Wheel Encoders (Wheel speeds)
- RC Receiver (Driver input)
- Serial xBee Receiver (Driver input)
- DC Motor Driver interface
- Servo Motor interface
- CAN protocol



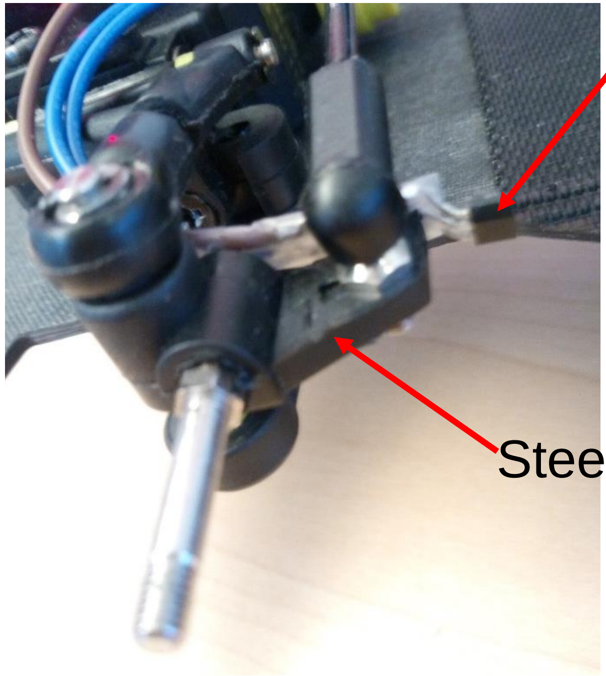
Simulink C2000 Modeling Approach

- Vehicle Dynamics Controller sample time is 0.01s, currently limited to encoder update rate (Can be changed but was a good compromise)
- Model structure:
  - High level controller
  - Asynchronous Systems (Interrupt based subsystems)
  - Synchronous Systems (Synchronized CAN communication)



Wheel Encoder Integration

- Hall effect sensor A1120EUA-T
- Digital output from magnet crossings
- Mounted on car chassis or steering knuckle
- 16 magnets mounted on wheel

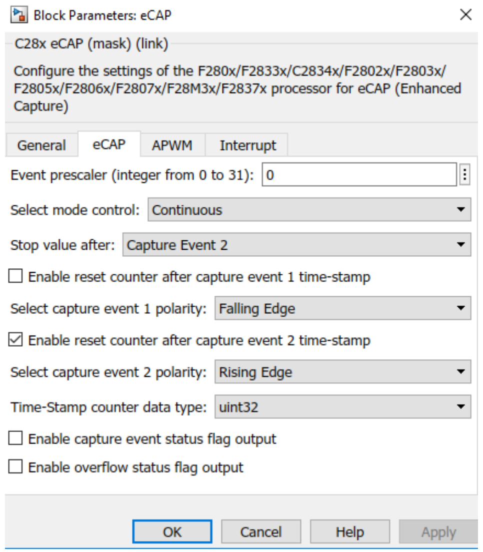


Sensor

Steering knuckle

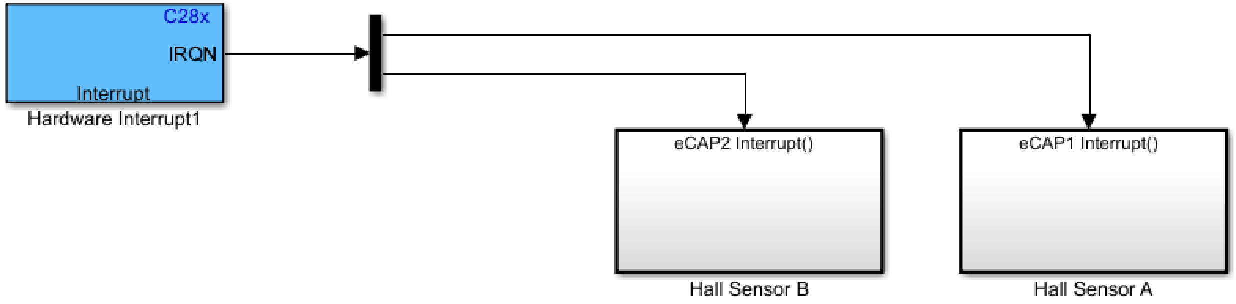
Wheel Encoder C2000 Implementation

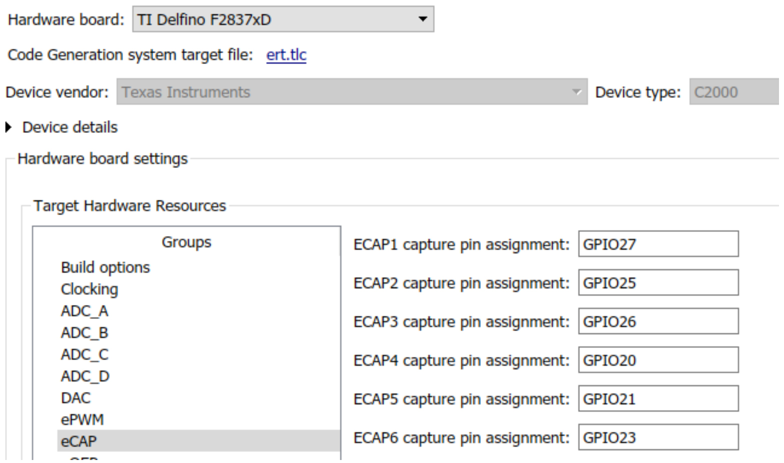
- TI eCAP Module (enhanced capture)
- Simulink eCAP block outputs timestamps of captured events
- For this implementation since sensor output is pulled-up, we want to capture a set of falling and rising edge (one magnet crossing).
- A counter is incremented in this Interrupt Service Routine (Triggered Asynchronous subsystem in Simulink)
- Check on timestamps prevents undesired counts from noise

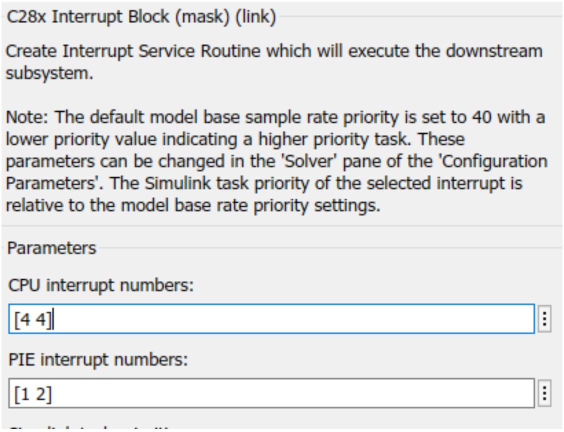


Wheel Encoder C2000 Implementation

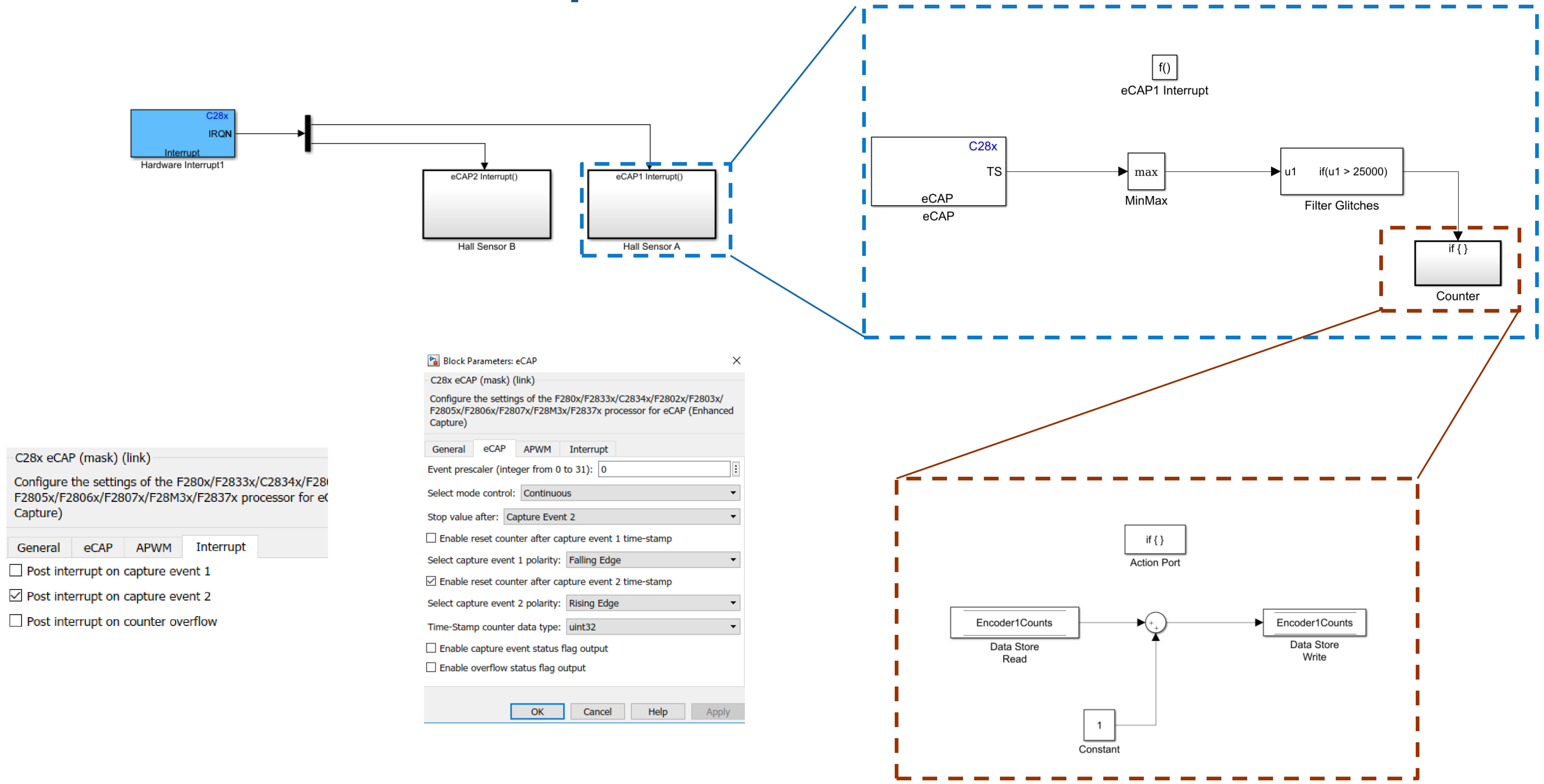
- Counts for encoder are updated asynchronously
- Interrupt is posted after a complete magnet crossing (rising edge)
- CPU and PIE interrupt numbers must be set to the corresponding eCAP modules
- eCAP pins must be set according to wiring on model's Hardware Implementation configurations





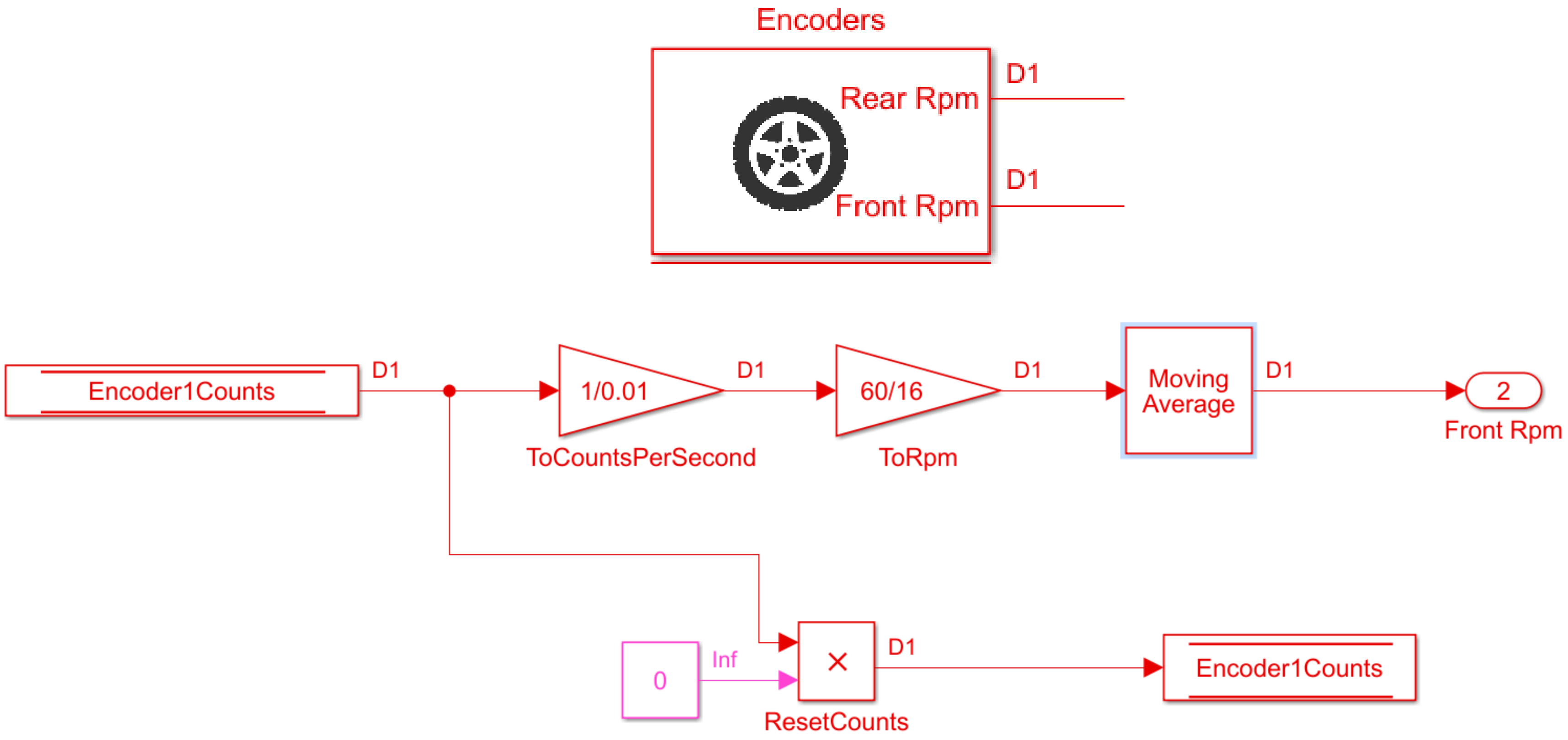


Wheel Encoder C2000 Implementation



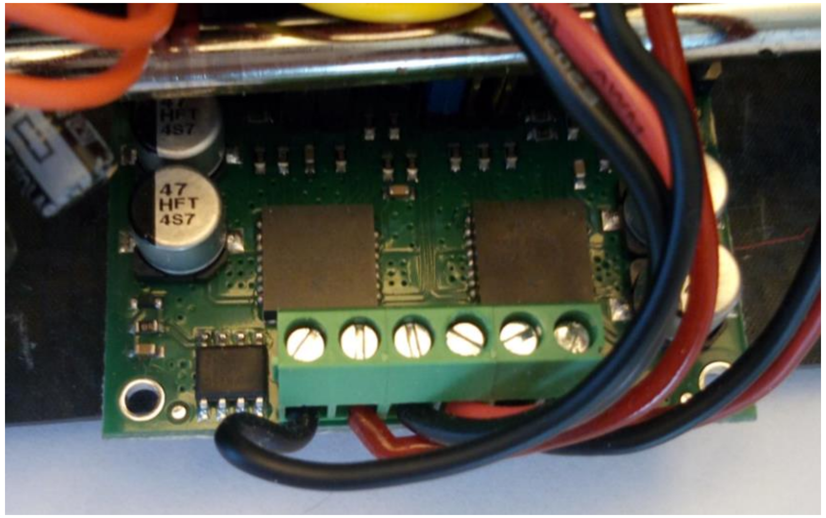
Wheel Encoder C2000 Implementation

- Counts retrieved and reset every 0.01s in main synchronous model
- Counts converted to revolutions per minute
- Moving average of 10 samples > Effective encoder sample rates is 0.1s



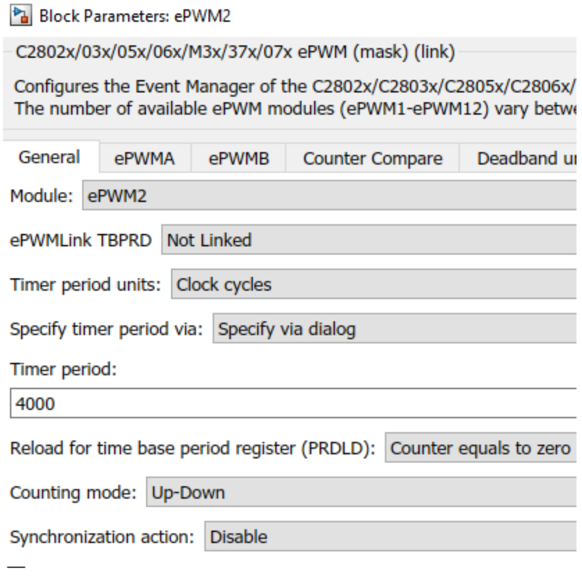
DC Motor Driver - Hardware

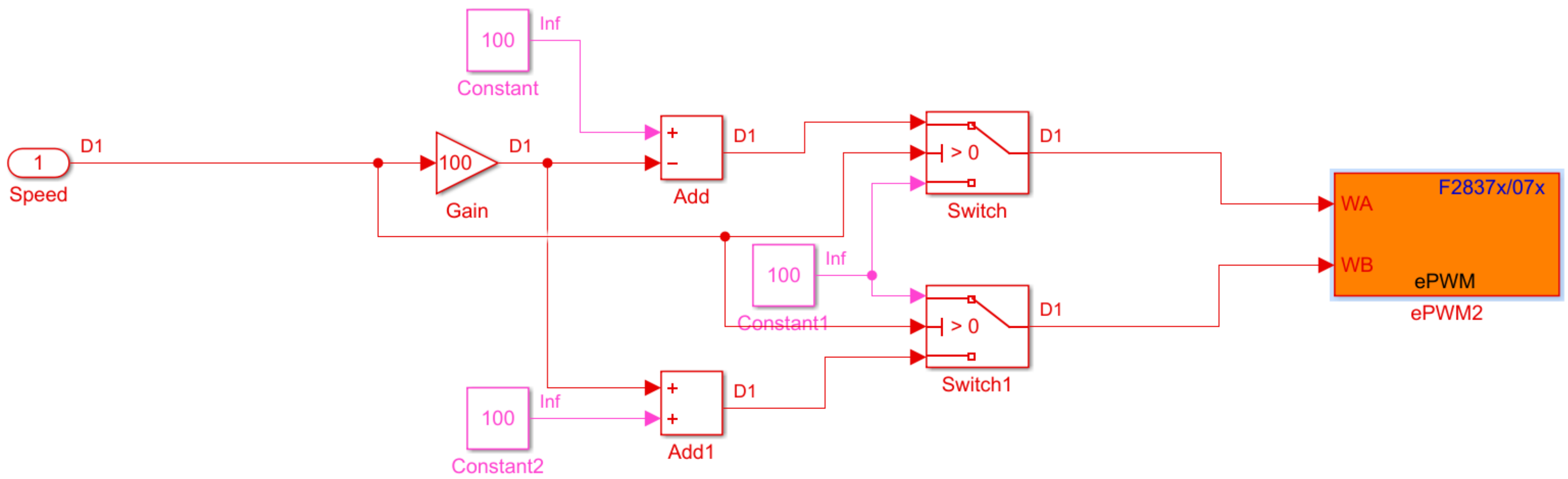
- Pololu Dual MC33926 Motor Driver Carrier
- Dual H-Bridge
- 2-pin interface per motor
- Use jumpers on Enable and D1/D2 pins
- Supports PWM up to 20kHz

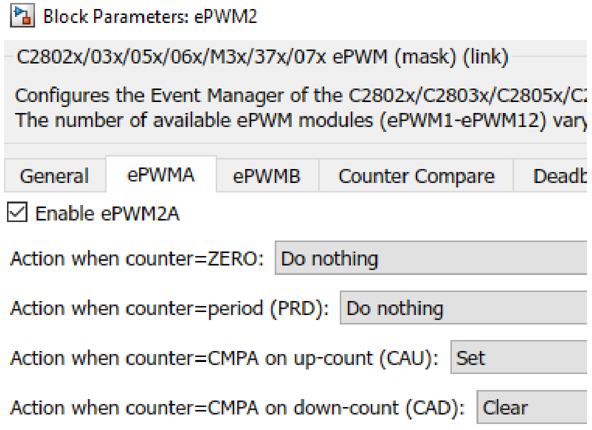


DC Motor Driver - Implementation

- TI ePWM modules are used to generate pwm
- Timer period must be less than 20kHz
- Use both A and B channels of one ePWM module for one motor
- Receive normalized desired speed: 1 Forward, -1 Reverse
- Set action on up-count and clear on down-count





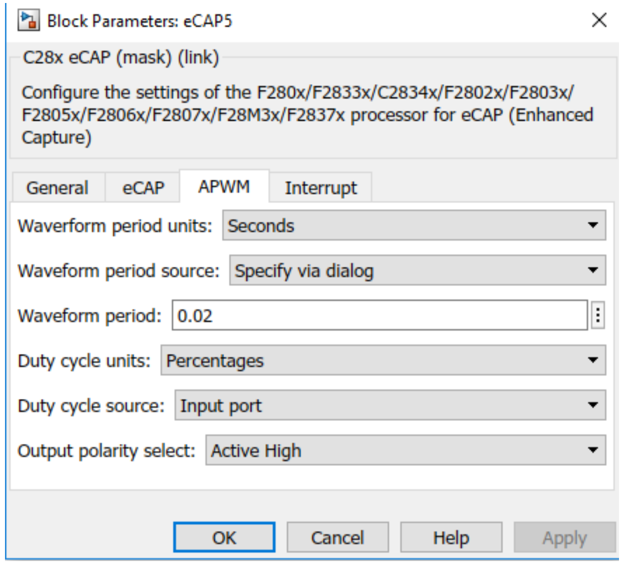


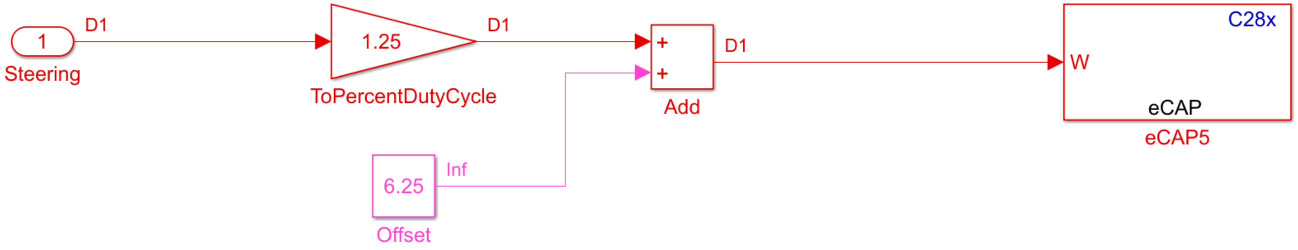
Servo Motor Steering Control - Hardware

- Futaba S3010 Servo Motor
- PWM Period 20ms
- PWM control range (1ms Left, 2ms Right)

Servo Motor Steering Control - Implementation

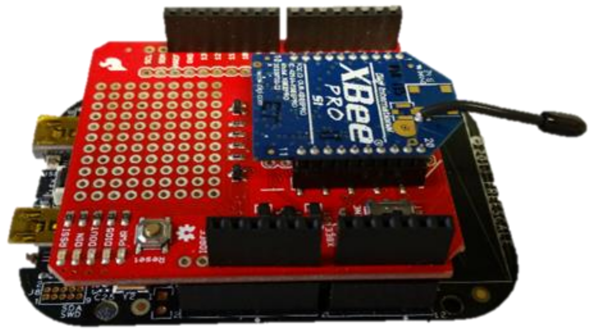
- APWM Option provided on eCAP modules
- APWM used instead of ePWM because of the much lower frequency (0.02s period vs 0.0013s max in ePWM)
- Normalized steering is the input (-1 Left, 1 Right)
- Offset and scaling may have to be adjusted for different motors

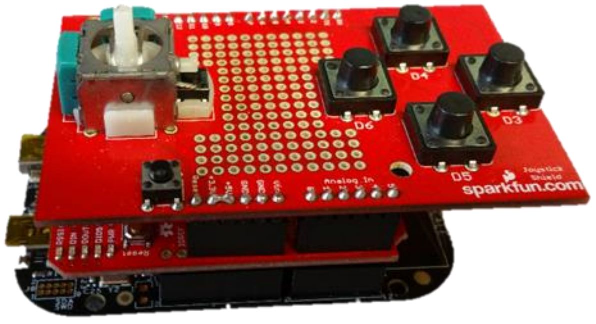


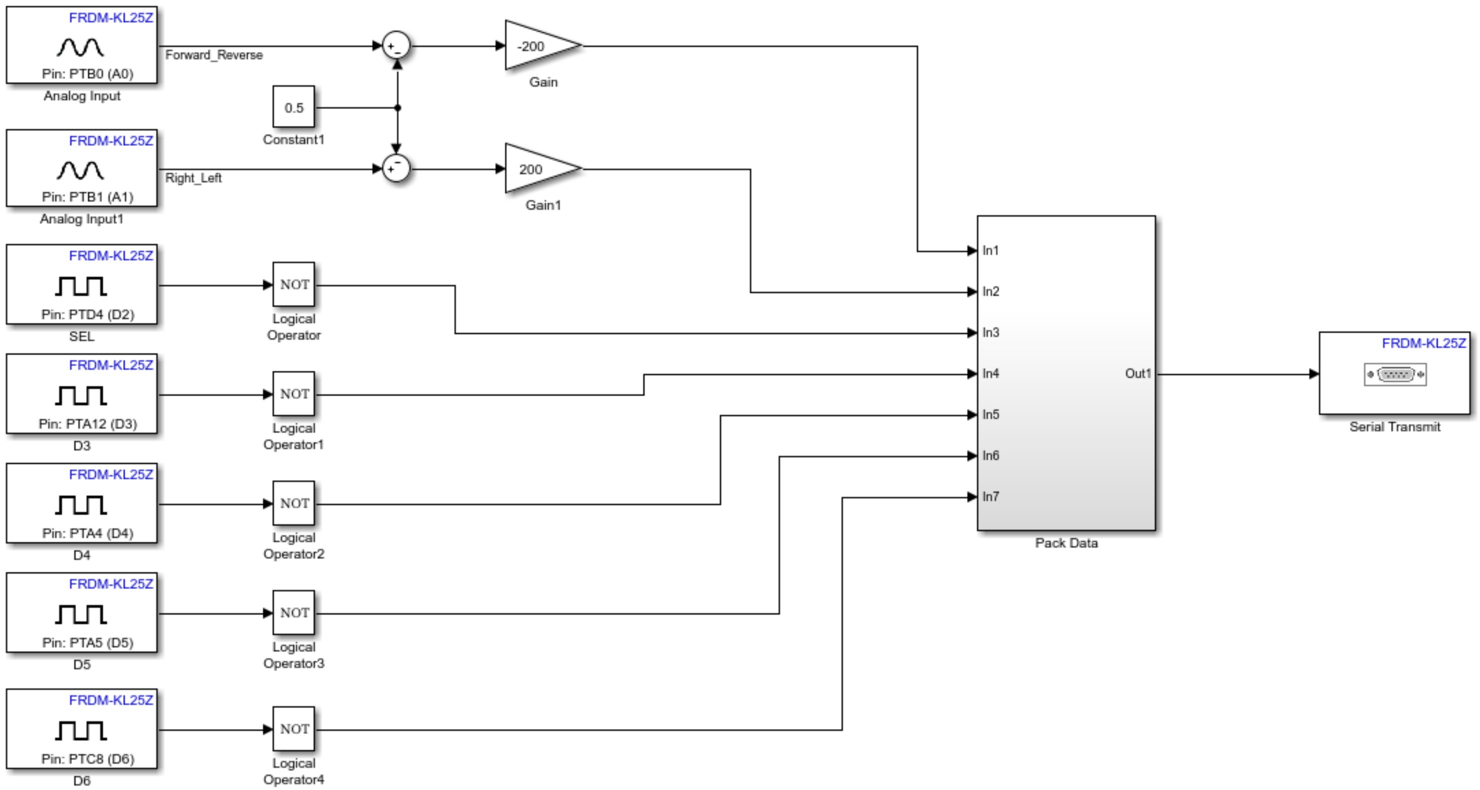


Serial Communication – Freescale KL25Z Transmitter

- Implemented through xBee RF network
- C2000 SCI block allows for packets to be received
- Header and terminator protocol setup

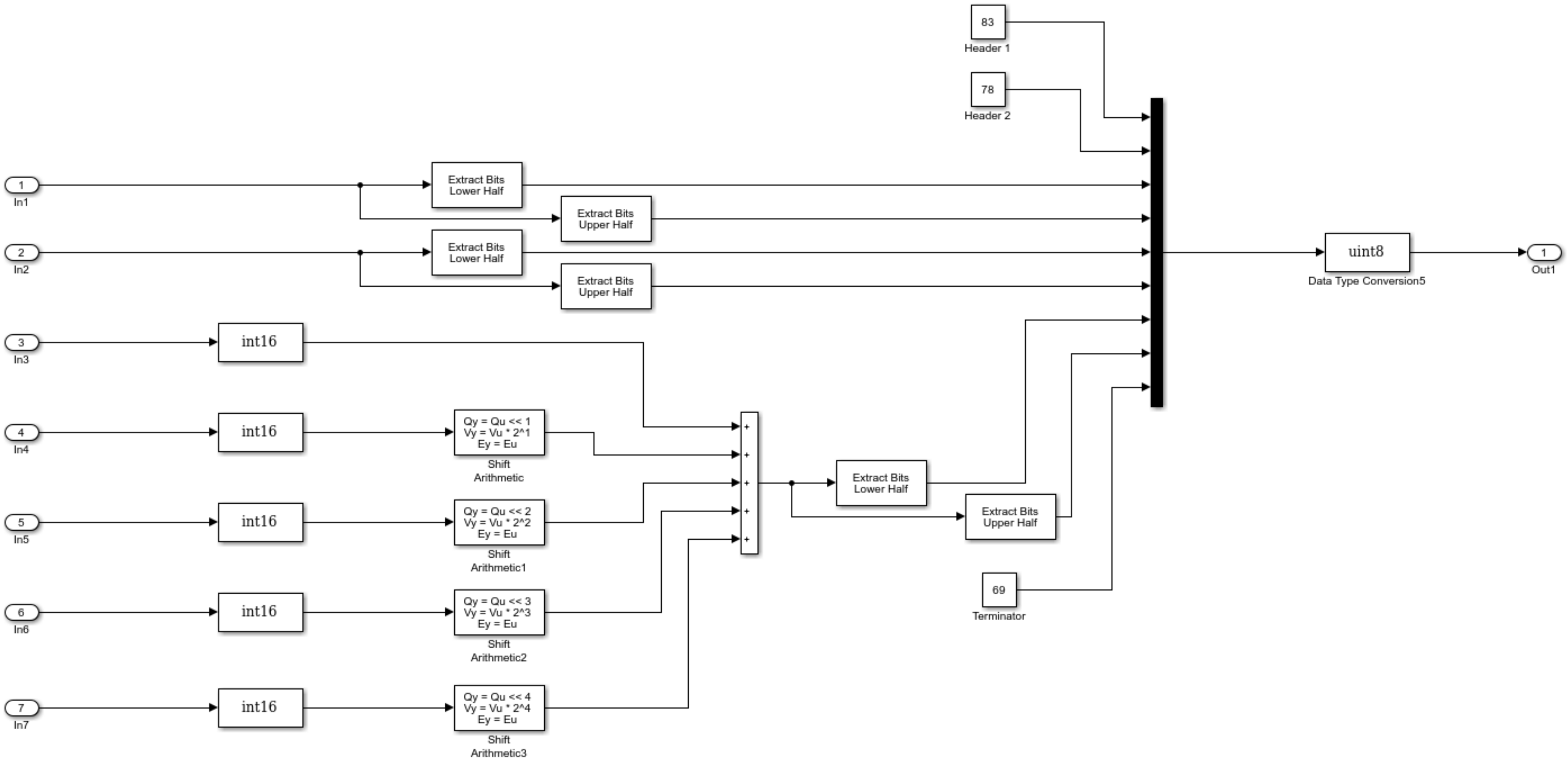






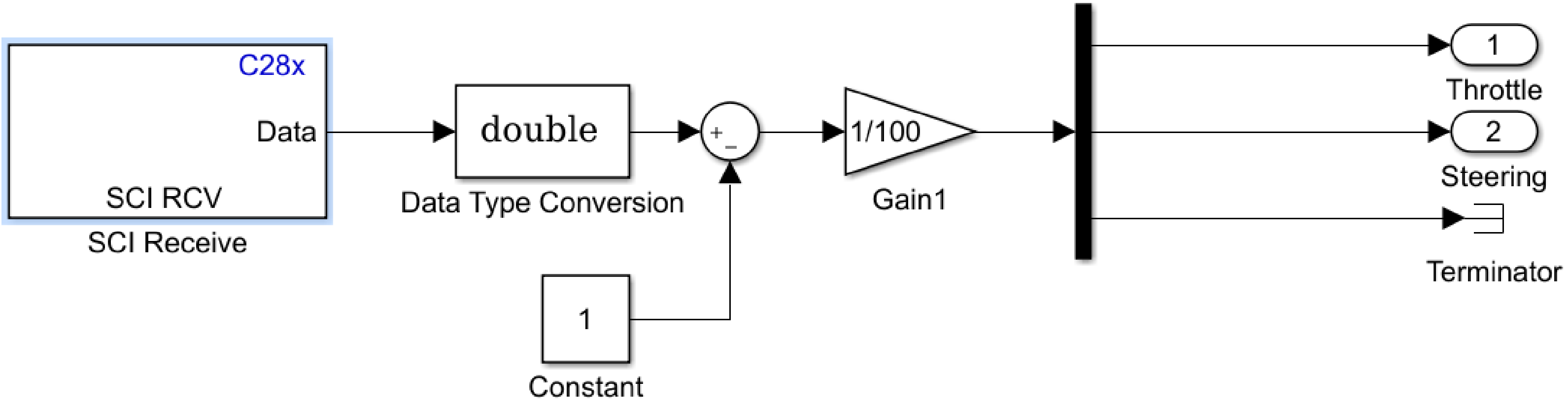
Serial Communication – Freescale KL25Z Transmitter

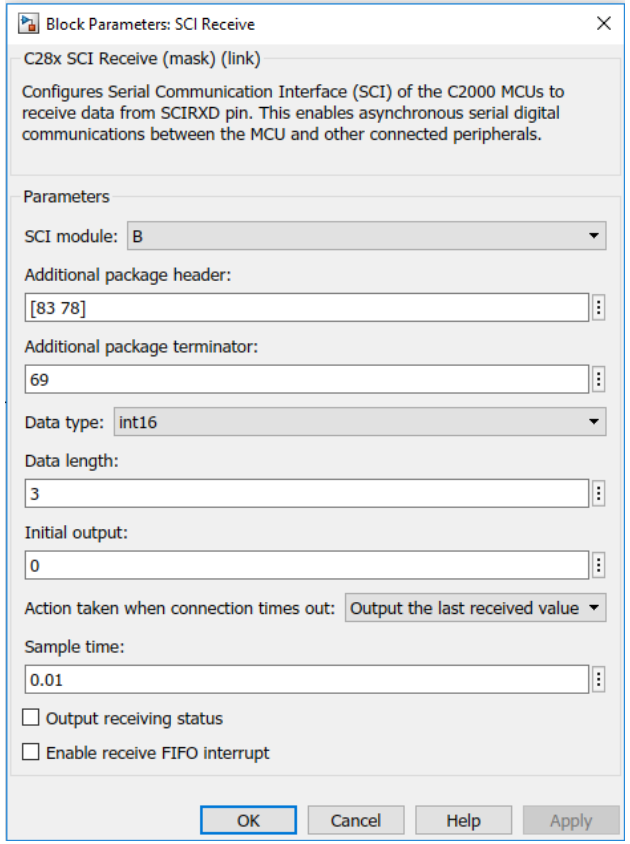
- Data packing implemented in Simulink
- Allows for custom messages to be sent along with throttle and steering commands



Serial Communication – C2000 Receiver

- SCI block
- Receive three int16
- Third int16 has packet binary data from KL25Z controller buttons





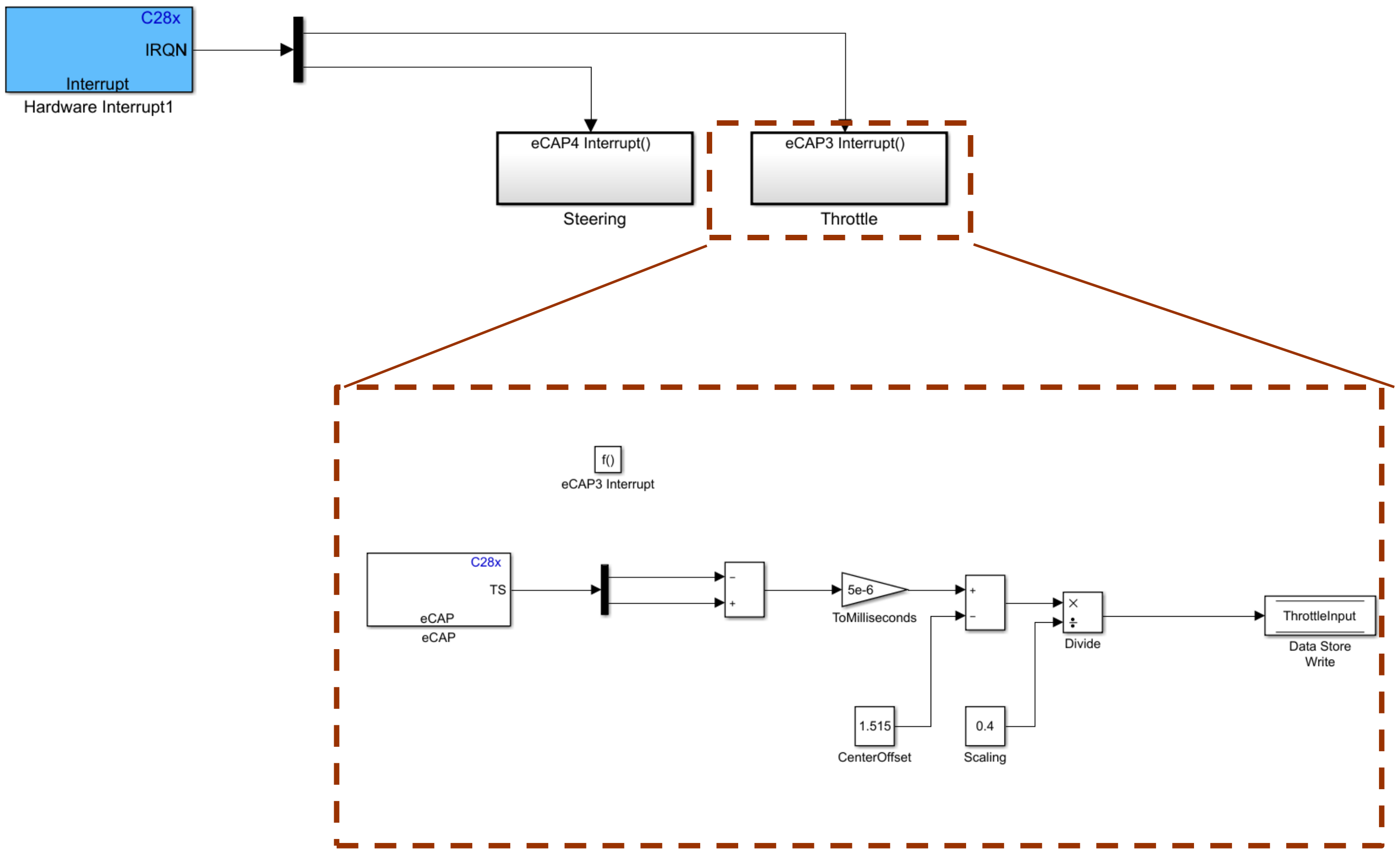
Remote Control Receiver

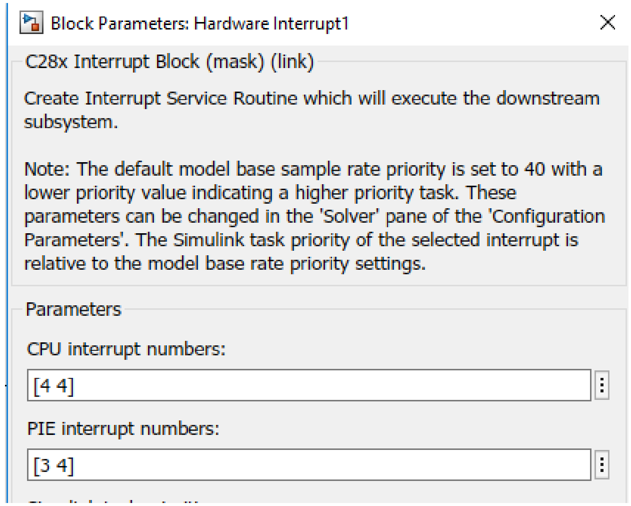
- For use with commercial hobby RC receivers and transmitters
- 50Hz PWM Frequency
- 1ms pulse length
- eCAP implementation in Simulink

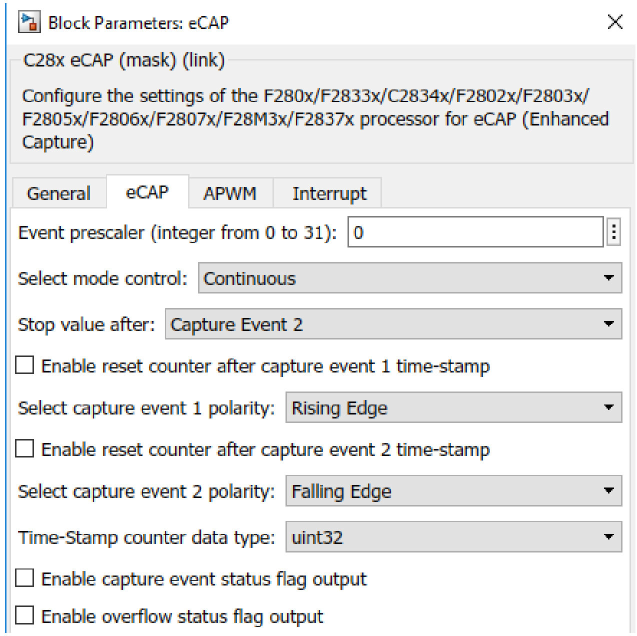
Remote Control Receiver – C2000 Implementation

- Similar eCAP setup as encoders
- Capture Timestamps of rising and falling edges
- Scale from Clock cycles to milliseconds and apply offset and scaling
- Assign pins in hardware implementation pane according to wiring

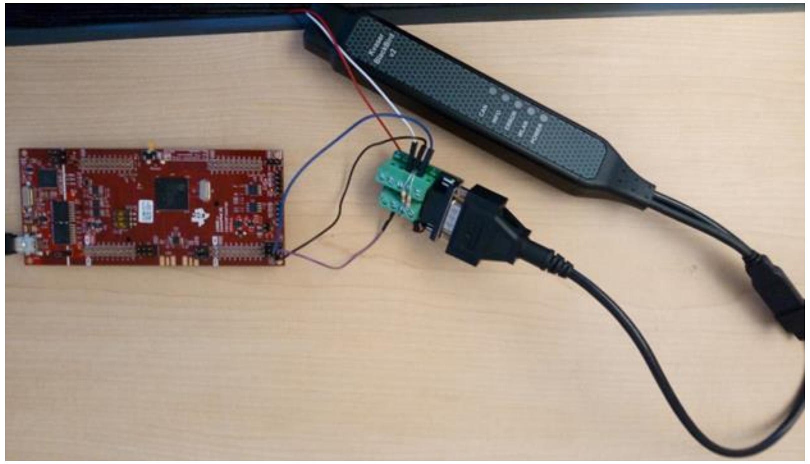
Remote Control Receiver – C2000 Implementation







CAN Communication – Hardware Setup



- Kvaser Blackbird V2 provides CAN over WIFI and is compatible with Simulink
- GND CAN-HIGH and CAN-LOW signals have to be connected to MCU
- 120Ohm terminating resistors are necessary
  - TI LaunchPad XL already has an integrated resistor
  - External resistor added to Kvaser Db9 connector
- Kvaser requires 9-40V. Therefore, a DC-DC converter was used to boost 7.4V battery power

CAN Communication

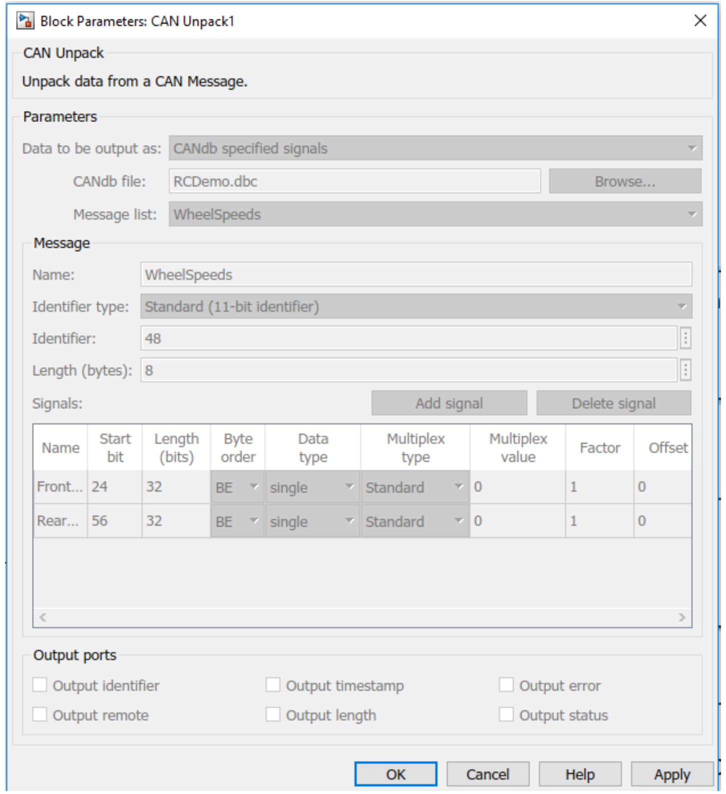
- Implemented using a CAN Database file (dbc)
  - Generated using Vector db++ Editor (RCDemo.dbc)
  - Contains message and signal specifications
- Contains telemetry from vehicle and controller parameters
- Most parameters are 32 bit Floating point numbers

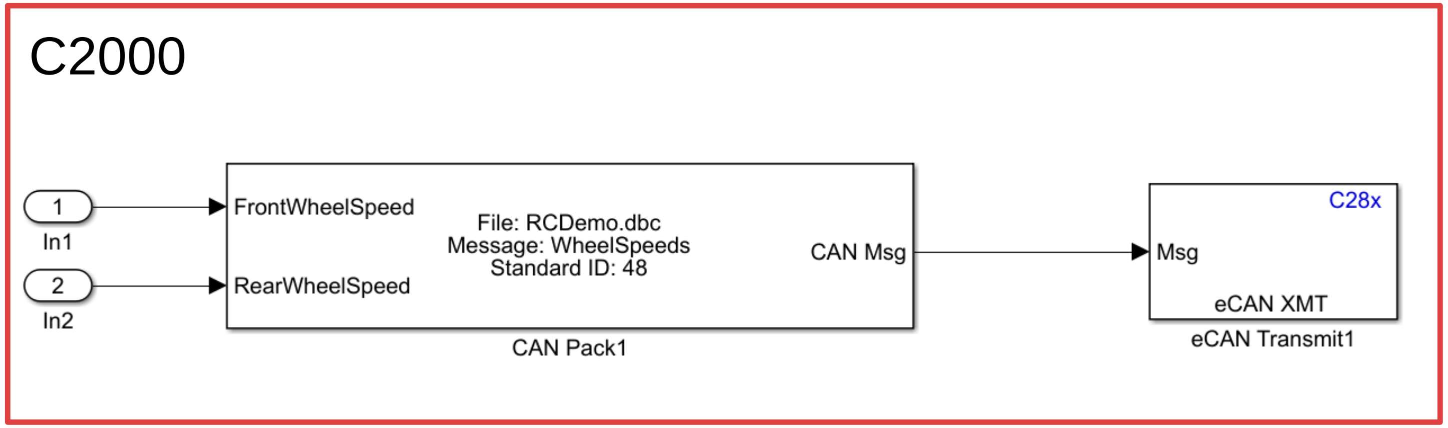
CAN Network Messages:

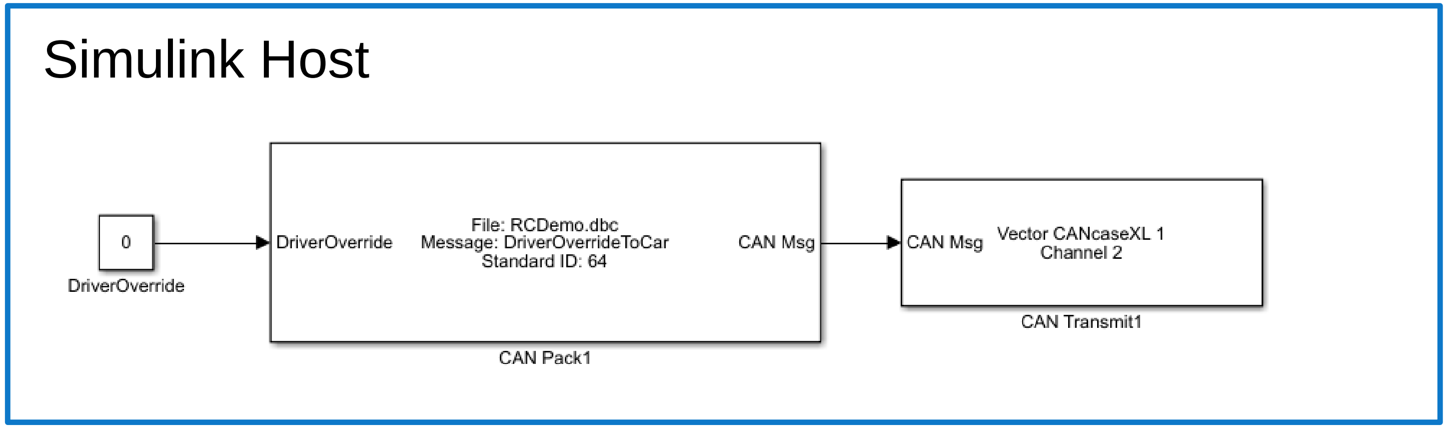
- DriverControlsFromCar
  - ThrottleInput
  - SteeringInput
- DriverControlsToCar
  - ThrottleInput
  - SteeringInput
- DriverOverrideFromCar
  - DriverOverride
- DriverOverrideToCar
  - DriverOverride
- DynamicsControllerGains
  - TorqueVectoringGain
  - TractionControlGain
- WheelSpeeds
  - FrontWheelSpeed
  - RearWheelSpeed

CAN Communication

- Use CAN Pack/Unpack along with CAN Transmit and Receive to share data on the CAN Bus
- DBC file contains packing information

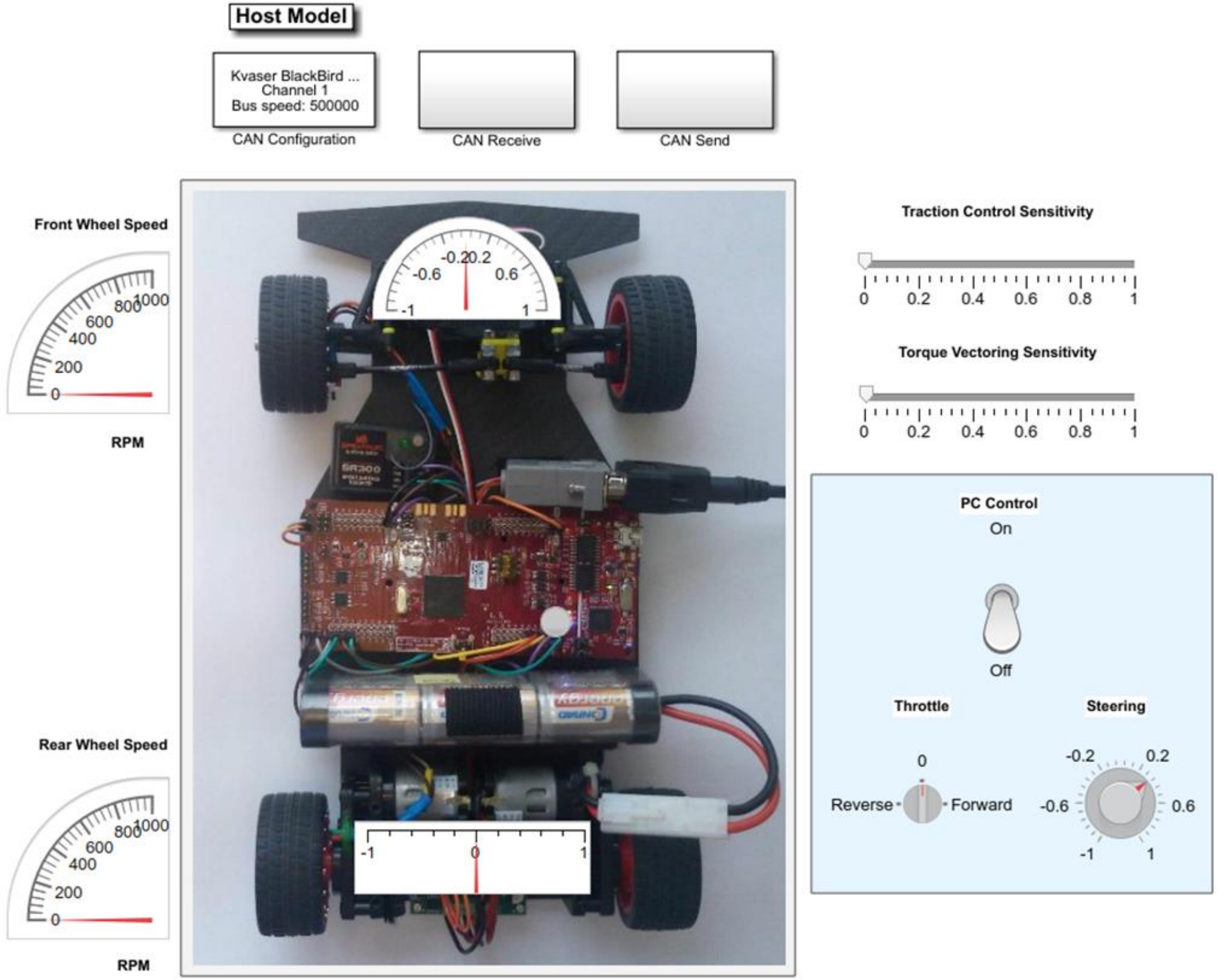


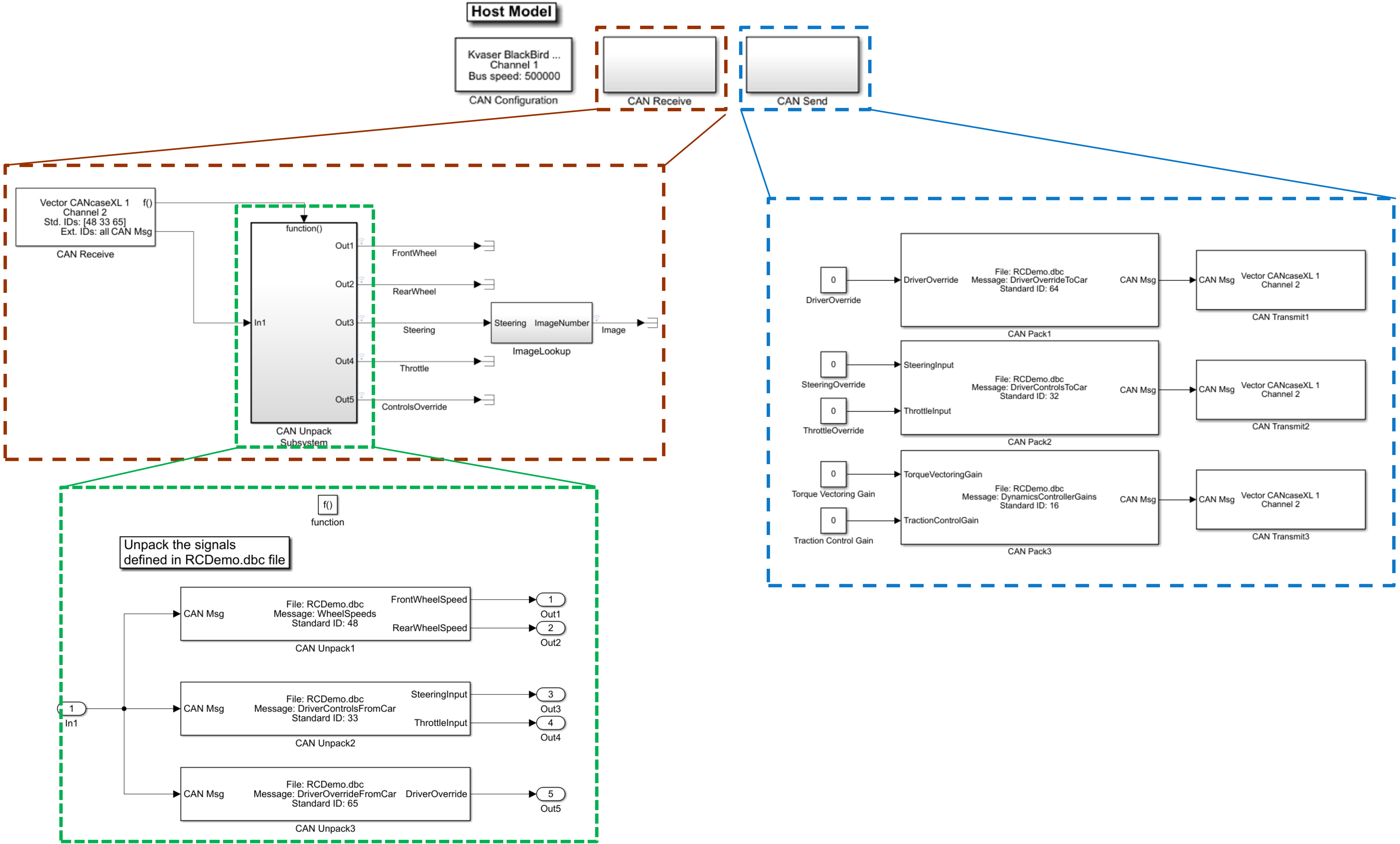




Host Simulink Model

- Use CAN Hardware support package to obtain access to CAN Network
- Transmit Messages to CAN Network
- Visualization of vehicle state using dashboard blocks
- Remote control through PC Host

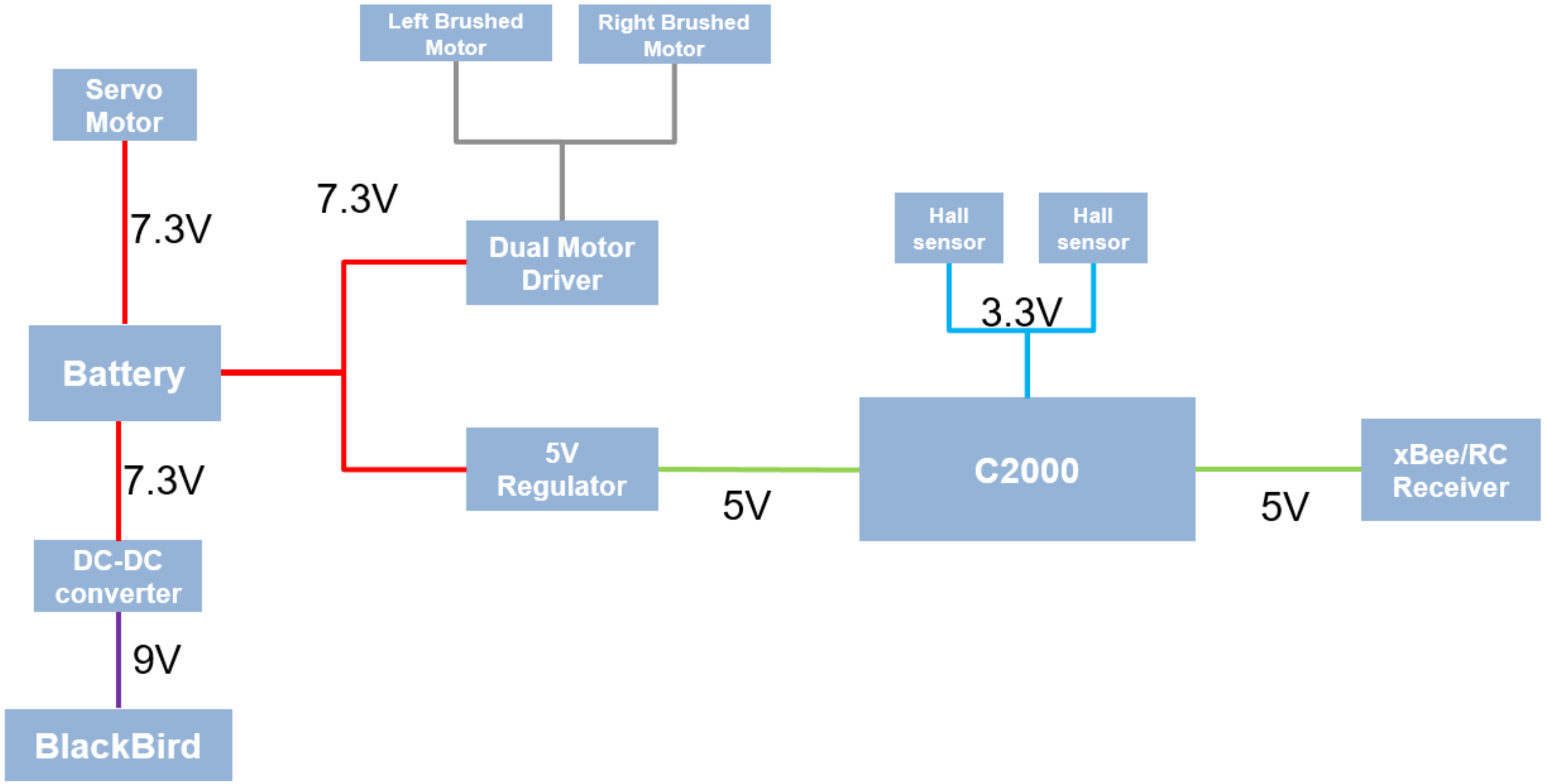

Host Model
Kvaser BlackBird ...
Channel 1
Bus speed: 500000
CAN Configuration
CAN Receive
CAN Send
Front Wheel Speed
RPM
Rear Wheel Speed
RPM
Traction Control Sensitivity
Torque Vectoring Sensitivity
PC Control
On
Off
Throttle
Reverse
Forward
Steering


Host Model
Kvaser BlackBird ...
Channel 1
Bus speed: 500000
CAN Configuration
CAN Receive
CAN Send
Vector CANcaseXL 1
Channel 2
Std. IDs: [48 33 65]
Ext. IDs: all CAN Msg
f()
CAN Receive
function()
In1
Out1
Out2
Out3
Out4
Out5
CAN Unpack
Subsystem
FrontWheel
RearWheel
Steering
Throttle
ControlsOverride
Steering
ImageNumber
ImageLookup
Image
0
DriverOverride
File: RCDemo.dbc
Message: DriverOverrideToCar
Standard ID: 64
CAN Msg
CAN Pack1
Vector CANcaseXL 1
Channel 2
CAN Transmit1
SteeringOverride
SteeringInput
ThrottleOverride
ThrottleInput
File: RCDemo.dbc
Message: DriverControlsToCar
Standard ID: 32
CAN Pack2
CAN Transmit2
Torque Vectoring Gain
TorqueVectoringGain
Traction Control Gain
TractionControlGain
File: RCDemo.dbc
Message: DynamicsControllerGains
Standard ID: 16
CAN Pack3
CAN Transmit3
f()
function
Unpack the signals
defined in RCDemo.dbc file
CAN Msg
File: RCDemo.dbc
Message: WheelSpeeds
Standard ID: 48
FrontWheelSpeed
RearWheelSpeed
CAN Unpack1
File: RCDemo.dbc
Message: DriverControlsFromCar
Standard ID: 33
SteeringInput
ThrottleInput
CAN Unpack2
File: RCDemo.dbc
Message: DriverOverrideFromCar
Standard ID: 65
DriverOverride
CAN Unpack3
1
In1
Out1
Out2
Out3
Out4
Out5

Appendix A - C2000 Peripheral Connections Table

Device	Description	CPU GPIO	Launchpad Pin	C2000 Module
Hall Sensor 1	Front Encoder Ticks	27	52	eCAP1
Hall Sensor 2	Rear Encoder Ticks	25	51	eCAP2
Motor Driver In1	Motor 1 Direction 1	1	39	ePWM1A
Motor Driver In2	Motor 1 Direction 2	0	40	ePWM1B
Motor Driver In3	Motor 2 Direction 1	3	37	ePWM2A
Motor Driver In4	Motor 2 Direction 2	2	38	ePWM2B
Servo Control	PWM Output For Steering servo	4	36	eCAP5
RC Receiver 1	PWM Input From Receiver (Throttle)	26	53	eCAP3
RC Receiver 2	PWM Input From Receiver (Steering)	64	54	eCAP4
xBee Serial RX	RS232 Receive Pin	11	75	SCI B

Appendix B – Power Distribution Diagram



Appendix C – Detailed Parts List

Parts reused from NXP cup chassis*:

- Front and rear wheel assemblies
- Steering servo Futaba S3010
- Motors (RN260-CN-18130 7.2V – 3.8A)
- 7.2V 3000mAh battery pack

*Was available here: https://community.nxp.com/docs/DOC-1284
Potential alternative, but expensive and might also be going out of production: https://developer.arm.com/academia/arm-university-program/for-educators/mechatronics-and-robotics/hardware
Consider modifying an off-the-self RC car with independent motors. See similar project:
https://hackaday.com/2014/07/12/independent-wheel-drive-rc-car/
Or if you are ok with something for low speeds check out amazon for "robot smart car" or "smart car kit" for other potential options.

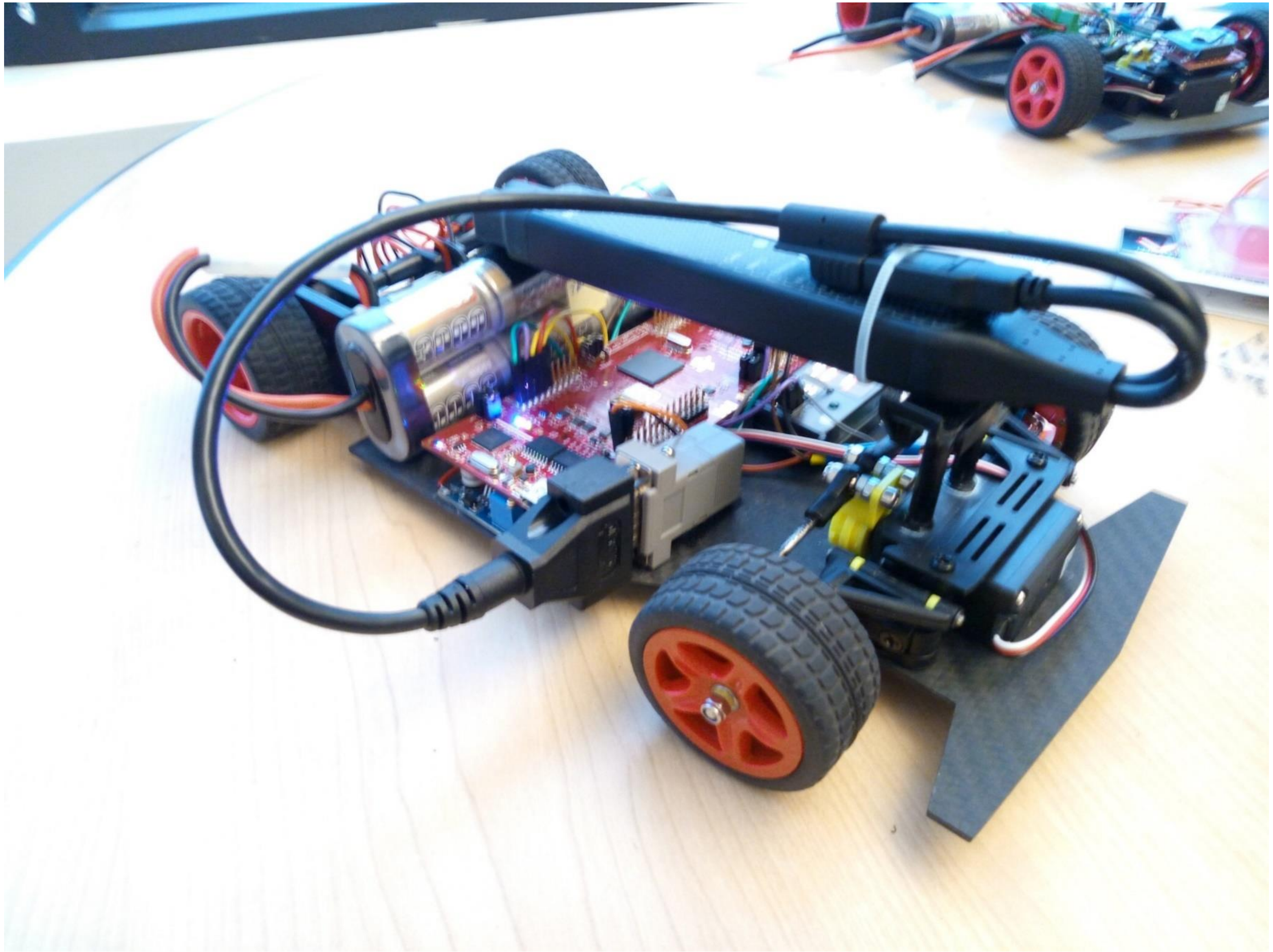
Added parts for demo:

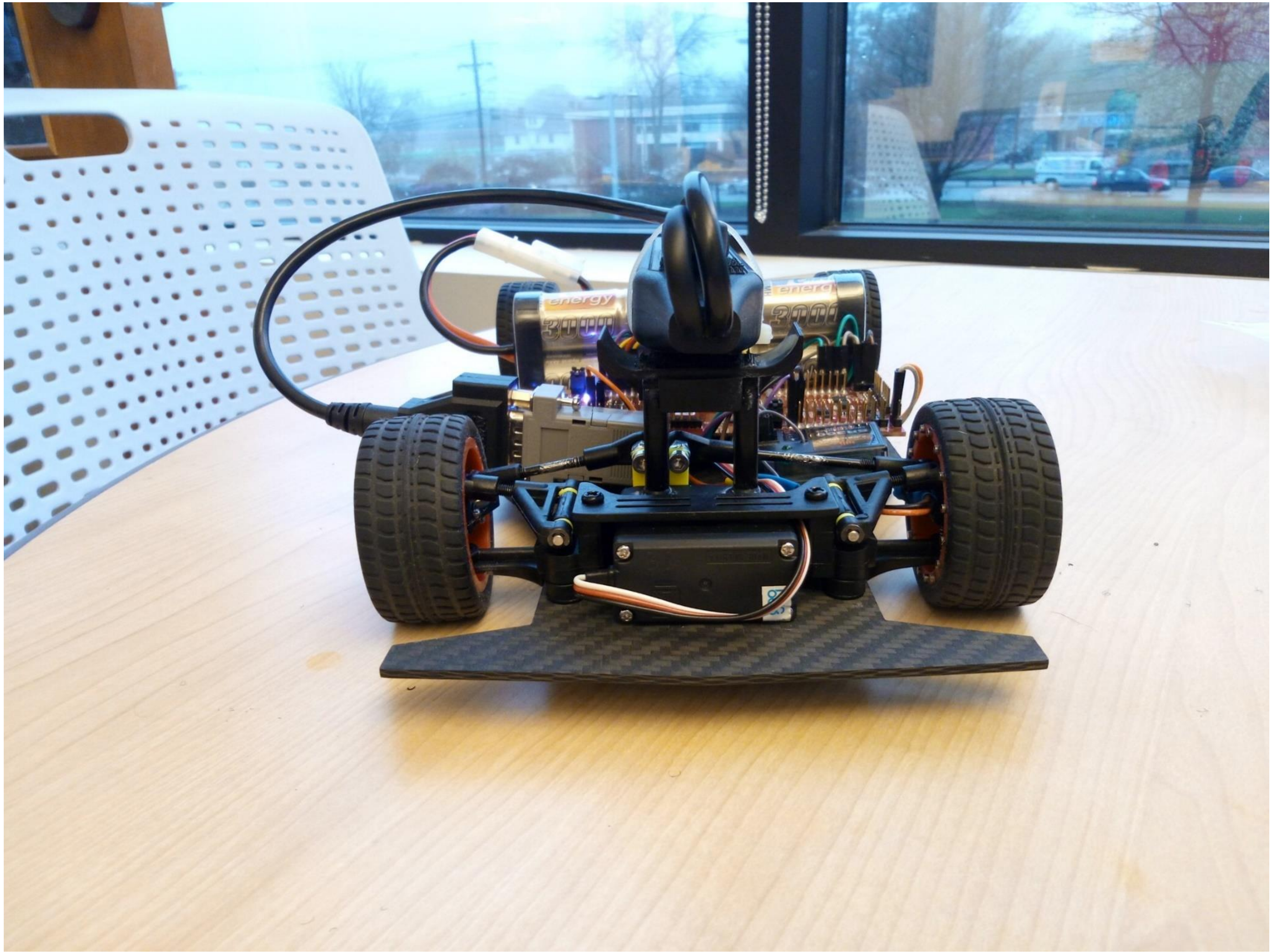
- A3144 hall effect sensor
- 16 Magnets per encoder wheel, Neodymium-Micromagnets 2 x 1 (D x t) mm
- Pololu Dual MC33926 Motor Driver carrier
- Texas Instuments Launchpad XL (F28379D)
- Kvaser Blackbird v2
- DB9 connector (added terminating resistors for CAN)
- XL6009 DC-DC Boost voltage converter
- L7805CV voltage regulator with filter capacitors
- RC controller and receiver- FlySky FS-GT2E
- Custom carbon fiber plate for chassis: Thickness 2.5mm (dxf file attached)

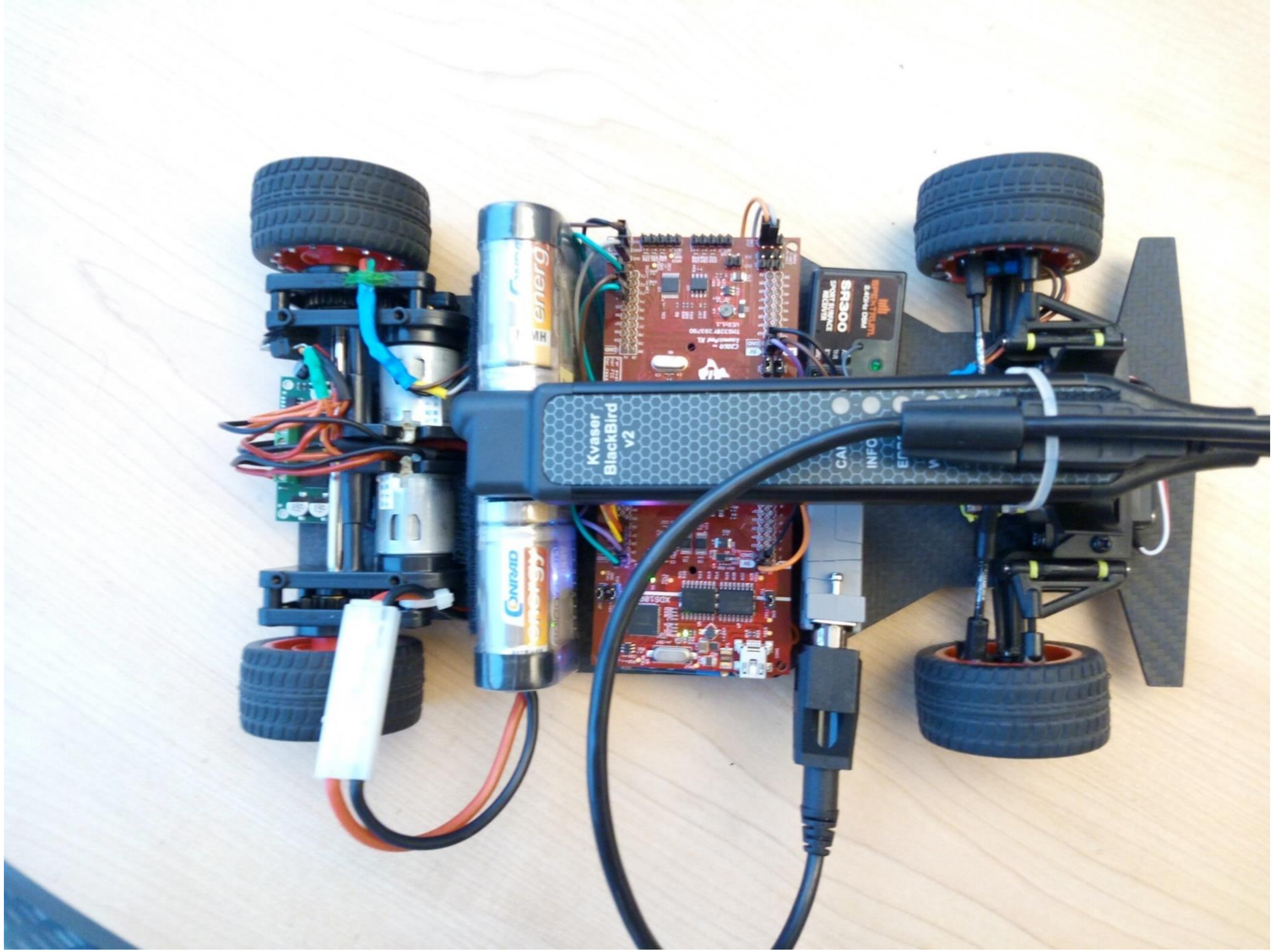
Miscellaneous components:

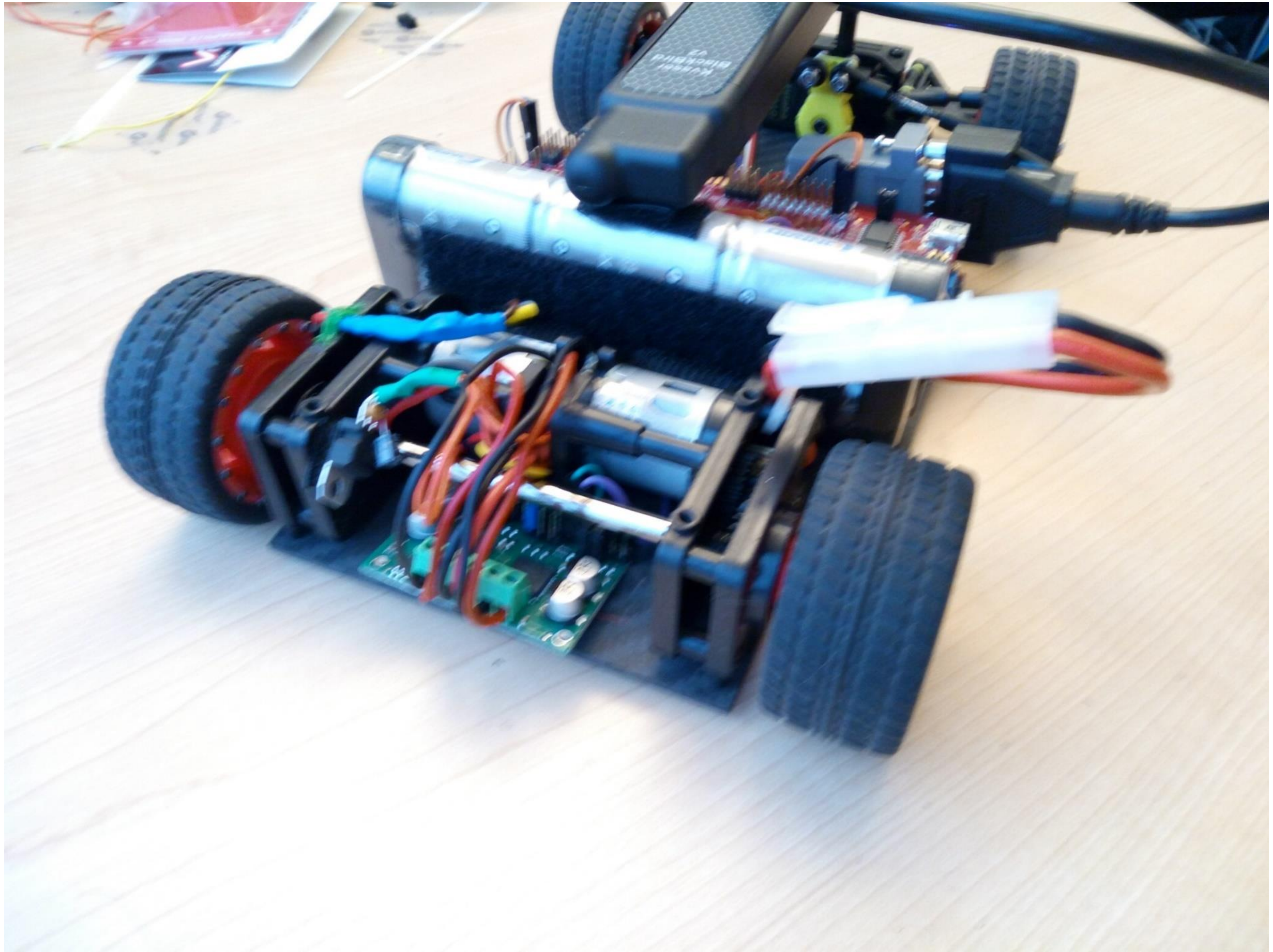
- Assorted wiring
- Heat shrink tubing
- Velcro (To attach battery and components to chassis)
- Super glue (Combined with thread to secure sensors)
- Tamiya connectors for battery
- Female jumper wires

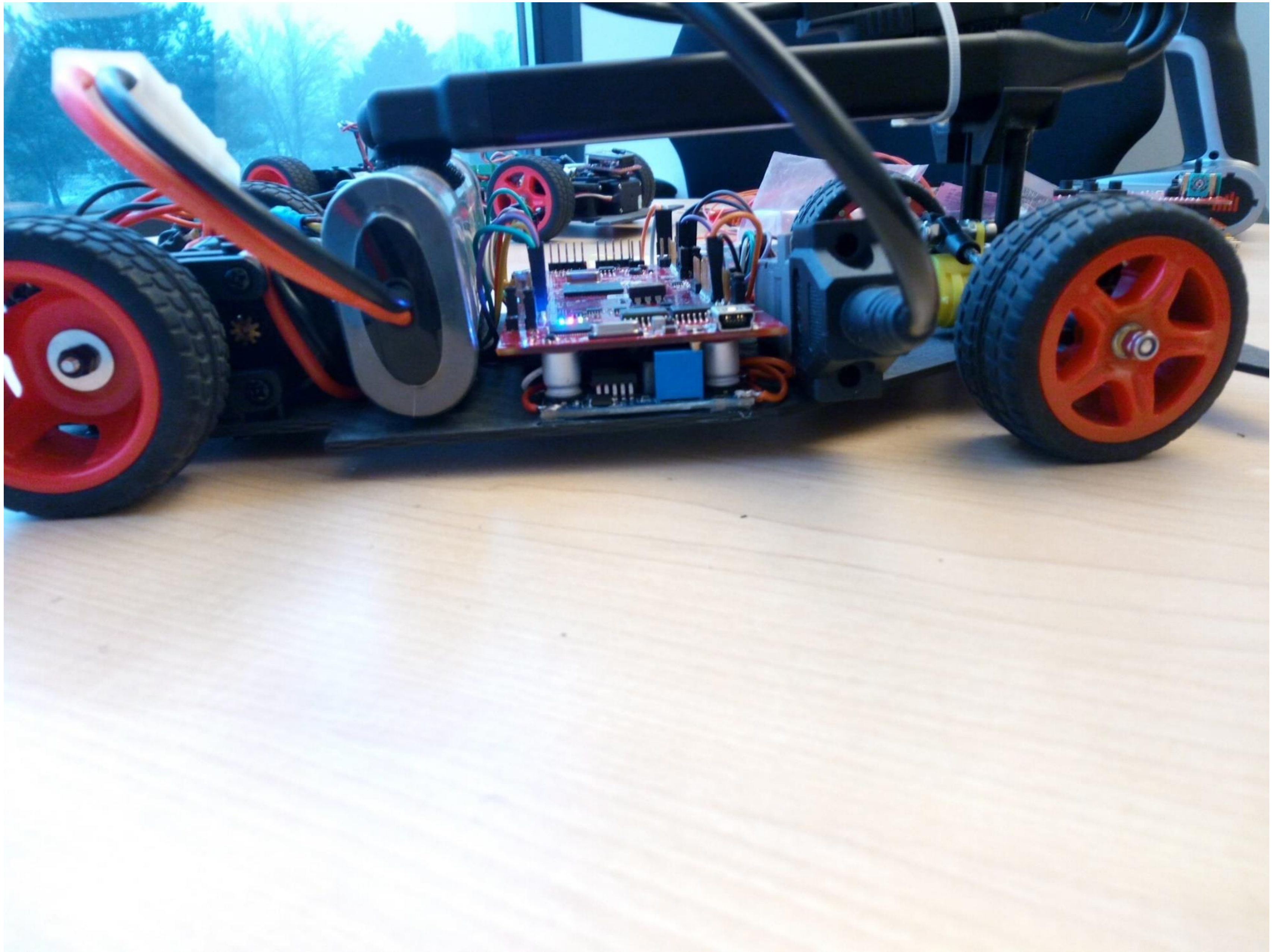
Appendix C – Additional Pictures of Assembled Hardware

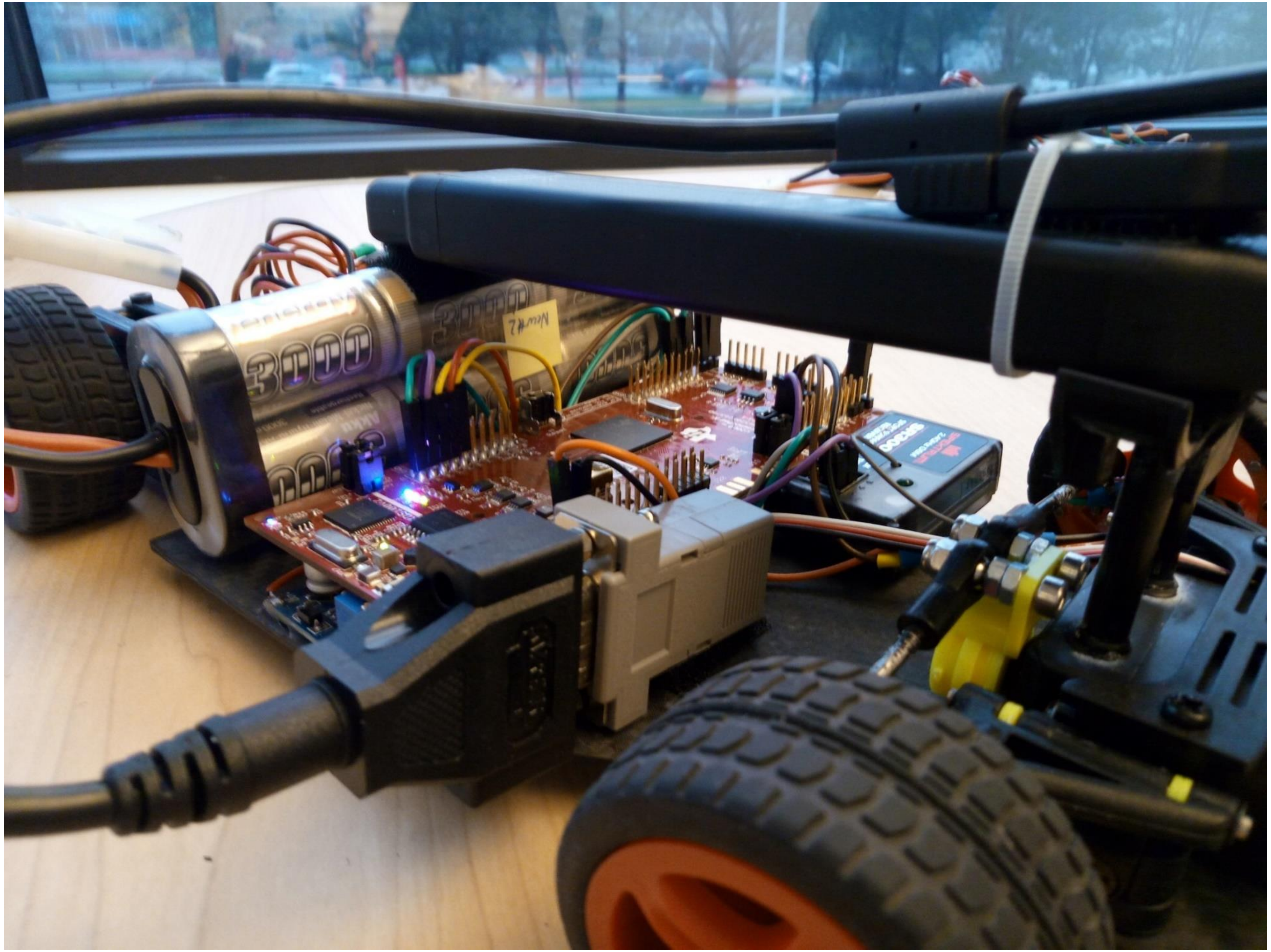



energy


Kvaser
BlackBird
v2






3000