Electronic Design

By JAMES LOCKRIDGE, Systems Engineer,
Texas Instruments

Simple, Low-Cost Dynamometer Setup for Motor Testing (Part 2)

The wrap-up article for this two-part series reveals the steps that went into creating the dynamometer with the help of the team from the University of Texas at Dallas.

In Part 1, I overviewed the technical details that impacted the design decisions for the dynamometer. In this article, I describe the final dynamometer system that the students on the Motor Torque Test Bench team (MTTB) built for me.

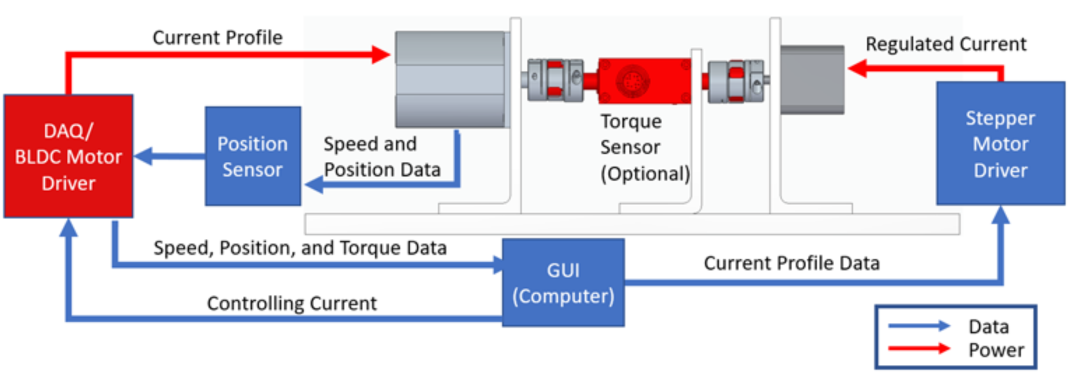
The Solution

Figure 1 shows a block diagram of the dynamometer's architecture. *Table 1* describes each of the dynamometer components with the reasons for selection.

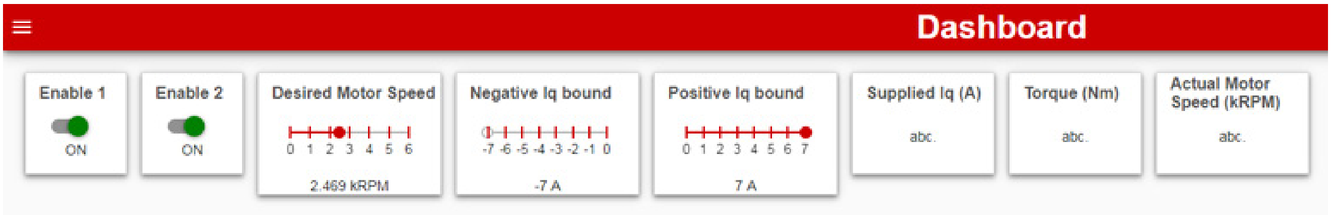
The total cost of the main components of the dynamometer is almost USD $303. This doesn't include the aluminum frame for mounting the motors, hardware, and machining costs (USD $140 material costs + USD $413 machining labor costs).

TABLE 1: PART SELECTION

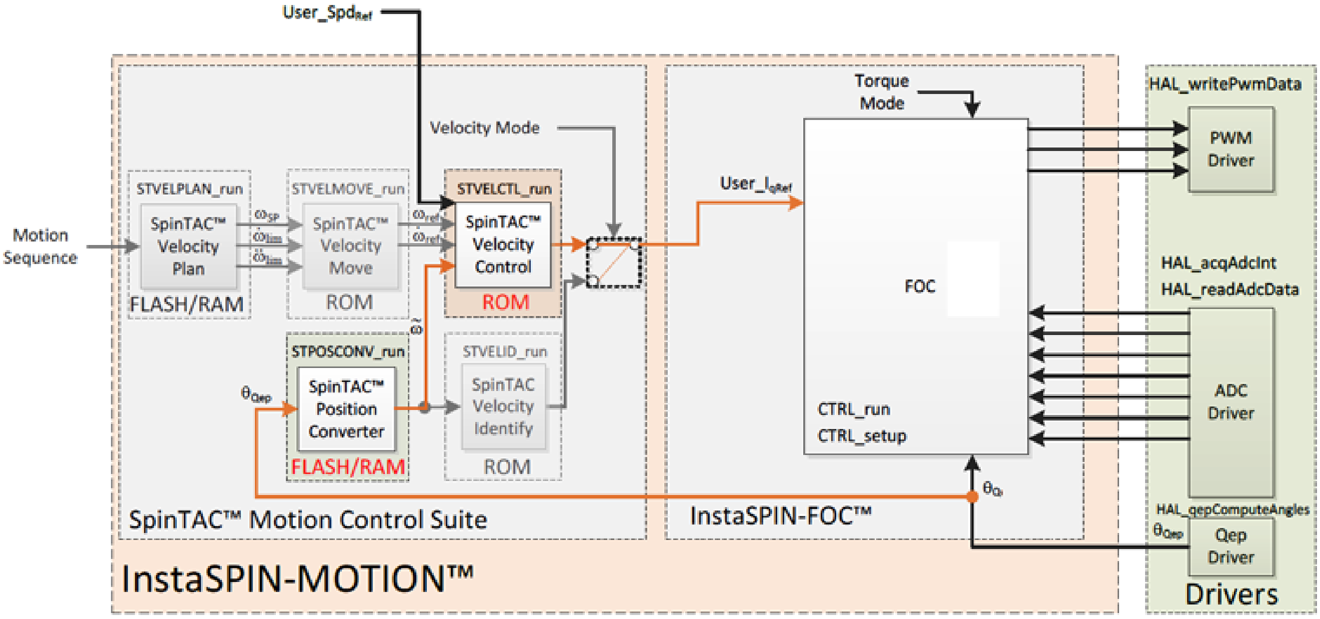
Subsystem Component	Part Number	Reasons for Selection	Price
Control algorithm	InstaSPIN-MOTION	The MTTB team was able to modify Lab12b from MotorWARE to implement sensored torque control with a quadrature encoder. The firmware provides global variables that are easy to interface with GUI Composer.	Free
Embedded controller	LAUNCHXL-F28069M	This C2000 LaunchPad has firmware examples for implementing Lab12b. It also has board connections and level translators for interfacing with a quadrature encoder.	$25
Motor- driver inverter	BOOSTXL-DRV8305EVM	This board drives the PMSM motor, interfaces with the C2000 LaunchPad, and is supported by the MotorWare Labs.	$79
Active load (PMSM)	Teknic M-2310P-LN-04K	This PMSM has a high-pole count (8 poles) for smooth motion, an integrated quadrature encoder, and interfaces with the LaunchPad as part of the LVSERVOMTR kit. The parameters of this motor are already configured in the userj1.h header file. Choosing another motor is possible, but the parameters must be determined by running earlier MotorWare labs.	$199



1. Shown is a block diagram of dynamometer.



2. Using Lab12b firmware, this GUI was created.



3. The Lab 12b FOC control algorithm is applied to regulate motor speed.

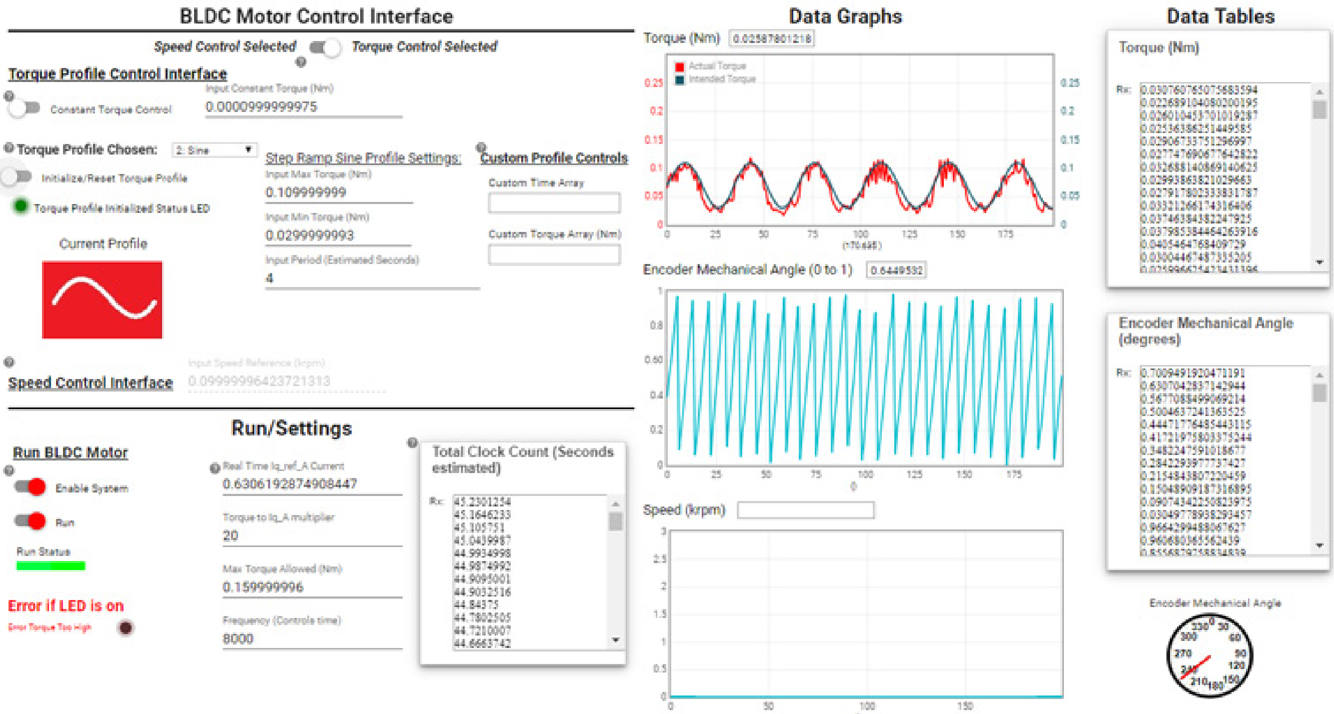
Initial GUI and Firmware—Speed Control

Chris Clearman's article and the advice he gave me offline helped the MTTB team decide on the hardware setup. The team and I experimented with the InstaSPIN-MOTION firmware as Chris recommends in the article. Using the Lab12b firmware example from MotorWare, I made an initial GUI (graphic user interface) for the dynamometer *(Fig. 2)*.

I created a new GUI Composer project (myTI login required) following the steps for MSP432 XDS interface (more helpful GUI Composer information on the "Getting Started" page). I bound the indicators and controls to the following variables required by Lab12b. The ".$q24" addition on the end of the variable name helps GUI Composer to

TABLE 2: BINDING GUI COMPOSER CONTROLS TO INSTASPIN VARIABLES IN LAB 12B

Control	Lab12b Variable	Purpose
Enable 1	gMotorVars.Flag_enableSys	Begins the operating for(;;) loop with the control routines
Enable 2	gMotorVars.Flag_Run_Identify	Determines rotor resistance before driving
Desired motor speed	gMotorVars.SpeedRef_krpm.$q24	Sets the intended speed of the motor
Negative Iq bound	gMotorVars.SpinTAC. VelCtlOutputMin_A.$q24	Sets the maximum torque in the negative/positive direction that the control routine will apply to the motor to try to achieve the desired motor speed
Positive Iq bound	gMotorVars.SpinTAC. VelCtlOutputMax_A.$q24	
Supplied Iq (A)	gMotorVars.IqRef_A.$q24	Reports the actual Iq current that the control routine commands to set the motor torque
Torque (Nm)	gMotorVars.Torque_Nm.$q24	Reports the estimated motor torque determined from the motor parameters configured in the userj1.h header file
Actual motor speed (kRPM)	gMotorVars.SpeedQEP_krpm.$q24	Reports the motor speed determined by the encoder

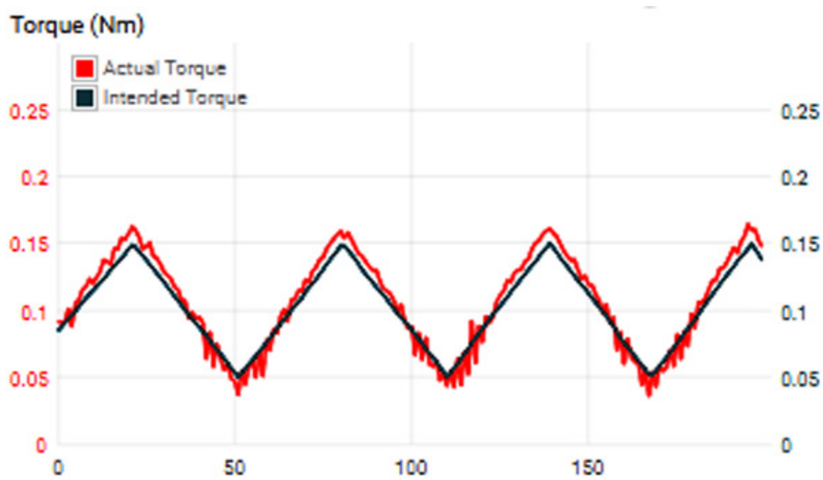


4. This is the dynamometer GUI created by the Motor Torque Test Bench Team at UTD.

display IQ-math variable types correctly. Otherwise, it would interpret them as a "long" and the values would not make sense. *Table 2* shows how I bound the variables to the controls and describes the function of each control.

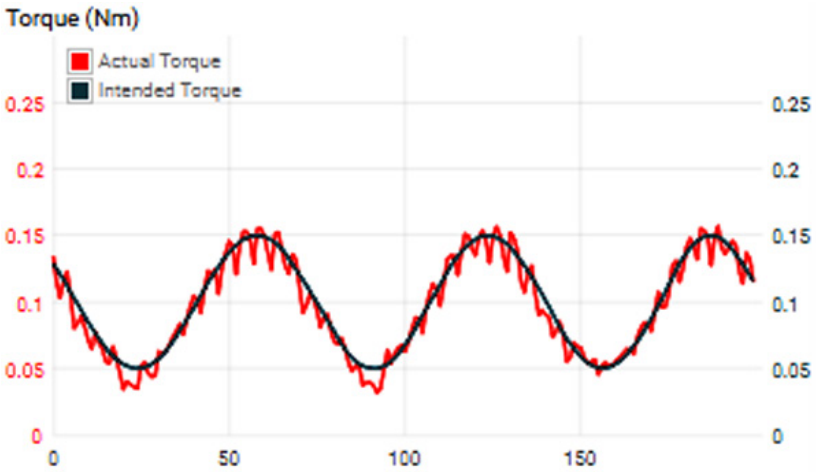
Figure 3 depicts the block diagram structure of the Lab12b code. The SpinTAC algorithm works to regulate the speed of the motor based on the encoder's signals. If an external torque begins to slow the motor, the SpinTAC controller increases the current (same as from Equation 3 from the first article) in order to boost the PMSM torque.

Following Chris's blog, Lab 12b can implement a basic dynamometer by setting the negative and positive bounds to a current level that produces the desired torque when the speed reference is 0 kRPM. As the motor under test (MUT) spins the PMSM, the SpinTAC controller increases the current (and therefore, the torque) to try to bring the rotor speed back to 0 kRPM. Eventually, the error in the SpinTAC loop increases to a point where the required current is at its positive or negative limit. This worked great for initial testing, but it requires more modifications to fully control the torque.



5. Triangle torque profile

6. Square-wave torque profile



7. Sinusoidal torque profile

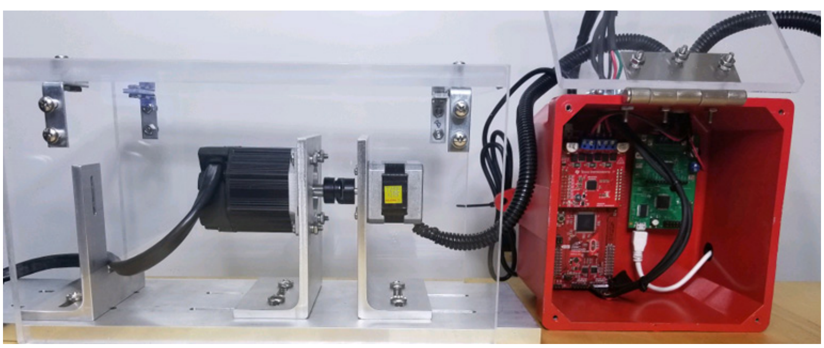
The next step was to modify the Lab 12b firmware to remove the speed-control loop. I did this by disabling the SpinTAC loop and modifying Lab 12b using some of the instructions in Motorware Lab 4. Lab 4 implements a torque-control loop, but it doesn't use an encoder, which is why I stuck with the Lab 12b code. All of the necessary modifications can be done within the proj_lab12b.c file.

The University of Texas at Dallas (UTD) team further modified the firmware to implement torque profiles and data collection in the GUI Composer. *Figure 4* shows the GUI they created. It reports torque, mechanical angle, speed and a timing reference. In addition, it allows the user to select speed control, torque control, and torque profiles.

Figures 5-7 show the plots for the configurable torque profiles with the estimated motor torque.

The UTD students documented their project on the TI University hackster.io page. Their GUI and firmware are both available there for reuse. *Figure 8* shows the final fully-assembled dynamometer system testing a stepper motor.

The dc power-supply rails of the stepper-motor driver board and the DRV8305 BoosterPack (PMSM driver board) are connected together on the same power supply. This allows the power to recirculate between the PMSM and stepper when the PMSM goes into its generator region. Otherwise, the power from the PMSM would need to be dissipated some other way, like adding an external load.



8. Here, the fully-assembled dynamometer is testing a stepper motor.

Conclusion

At the end of our eight-month journey, the MTTB team truly impressed me with their accomplishments. We all learned an incredible amount about different motor types, motor construction, and motor control. We carefully thought through many design concerns, and I now have an excellent dynamometer to use for motor testing.

For future development, I would like to enable an option to interface the dynamometer with LabView and add a feature that allows me to drive the motor under test (MUT) to a stall condition without overdriving it. I would also check out the new C2000WARE-MOTORCONTROL-SDK, which released toward the end of our project.

JAMES LOCKRIDGE is a systems engineer within TI's motor-drives business.

Acknowledgements

• University of Texas at Dallas UTDesign staff and students (listed as contributors on the Hackster page)
• Chris Clearman for recommending the LaunchPad, BoosterPack, and InstaSPIN firmware to get us started on development.
• David Magee, Rajan Narasimha, and Stephen Fedigan for our discussions on dynamometer designs earlier in the project.

Resources

Acarnley, Paul P. *Stepping motors: a guide to theory and practice*. 4th ed., Institution of Engineering and Technology, 2007.

Krause, P. Wasynczuk, O. Pekarek, S. *Electromechanical Motion Devices*. IEEE. 2012.

"Hysteresis Brakes and Clutches." 2nd edition. www.magtrol.com. November 2011.

"Understanding the Distinctions Among Torque Ripple, Cogging Torque, and Detent." *Tech Papers*. Motion Control Resources. www.motioncontrolonline.org. 15 August 2013.

Akin, B. Bhardwaj, M. Warriner, J. "Trapezoidal Control of BLDC Motors Using Hall Effect Sensors." www.ti.com. April 2011.