Chapter

# Torque Control of a DC Motor with a State Space Estimator and Kalman Filter Applied in Electrical Vehicles

*Alex Archela, Dario Guilherme Toginho and Leonimer Flávio de Melo*

## Abstract

This work presents a study over a torque-generated speed control of free wheel attached to a DC motor, for use on traction of mobile vehicles. Also, it presents the discrete state space model of a DC model and the Kalman filter's equations and applications. This work presents a hardware-in-the-loop (HIL) system for design of a torque controller noticed that this process produces a faster design, coding, and parameter optimization of any embedded systems. The hardware used for the implementation of the system is discussed as the hardware-in-the-loop environment which makes possible the fast tuning and design of the system. In the absence of a torque sensor, this work uses the Kalman filter's estimated states torque and speed as feedbacks of the system.

**Keywords:** hardware-in-the-loop, Kalman filter, discrete state estimation, DC motor, closed-loop control, electrical vehicles, torque control, embedded control systems

## 1. Introduction

More attention has been taken on electrical vehicles for a series of motives, like price of oil increasing and concern about global environmental problems [1]. The study on this area has been searching for methods of increasing the energy efficiency, safety, stability, and performance of those systems [2, 3].

Another aim of the researchers is to develop efficient control strategies for navigation, but without an accurate control of the traction motors, the trajectory desired cannot be followed [4]. Traction control of vehicles can be very tricky though, because the friction coefficient between the tire and surface is nonlinear and uncertain [5]. This effect also happens for applications in mobile robots with torque higher than the surface friction; hence, the velocity control cannot assure that the trajectory will be followed.

The architecture of four independent motors actuating on each wheel is presented by [5, 6]. Usually, each wheel is controlled by a torque controller, direct or indirectly, as in [7], which proposes an optimum torque distribution strategy of four independent motors on an electric vehicle, considering its driving force and yaw moment ratios testing on slippery road conditions.

As pointed by [8], the torque of an electric motor can be generated quickly and accurately and also can be easily measured through an observer, making it possible to create a drive force observer. This work presents a study over the torque control of a DC motor, which the electromagnetic torque is estimated, with a discrete state space estimator and a Kalman filter [9].

## 2. System design

In this section, the mechanical system is described, which is the aim of the control study; it also presented the mathematical description of this model, so it can further be applied in this work.

### 2.1 Mechanical system

The choice of physical implementation for testing, setting, and tuning is a motor-wheel system, where a motor is attached to a suspended wheel with the use of a crown, pinion, and chain. This system provides an actuation of the motor on acceleration and braking of the wheel, making it possible to generate a full cycle of control. **Figure 1** presents the current system.

Notice that the motor is attached directly to the wheel instead of an axis, providing the possibility of a full cycle of control of the wheel speed and torque by the motor.

### 2.2 Dynamic model

The approach of the independent-driven wheels for vehicle traction results in a DC motor attached to a single wheel. Therefore, knowingly Eqs. (1)–(3) describe the electric DC motor states:

$$v(t) = R_a i_a(t) + L_a i'_a + e(t) \tag{1}$$

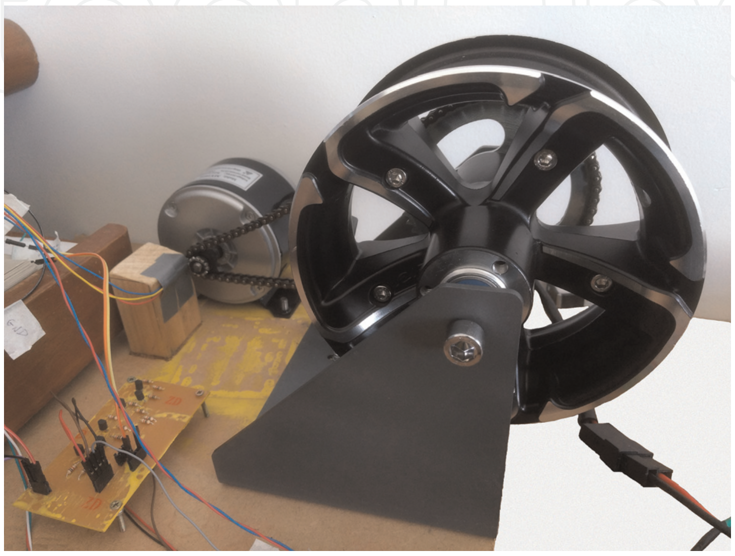

**Figure 1.**
*Suspended wheel attached to the traction motor by a crown-chain-pinion.*

$$e(t) = K_e\omega(t) \tag{2}$$

$$T_m = K_t i_a(t) \tag{3}$$

where $v(t)$ is the voltage supplied to the motor (V), $R_a$ the armature resistance ($\Omega$), $i_a$ the value of armature current (A), $L_a$ the armature inductance (H), $e(t)$ the back electromotive force (V), $K_e$ the back electromotive constant (V.s/rad), $\omega(t)$ the motor shaft speed (rad/s), $T_m$ the motor electromagnetic torque (N.m) and $K_t$ is the motor torque constant (N.m.A). Substituting (2) and (3) in (1) yields (4):

$$v(t) = R_a i_a(t) + L_a i'_a + e(t) \tag{4}$$

Apart from the DC motor, the single wheel attached to the motor alters the dynamic equations for the system. Applying Newton's second law of motion on the rotational movement results in Eq. (5):

$$J_T \frac{d\omega(t)}{dt} = T_m(t) - T_L(t) - B_T\omega(t) \tag{5}$$

Given that $B_T$ is the coefficient of viscosity (Kgm$^2$/s) of the system and $J_T$ is the moment of inertia (Kgm$^2$) generated by the sum of the motor shaft ($J_m$), the wheel ($J_r$) is given by Eq. (6):

$$e(t) = K_e\omega(t) \tag{6}$$

Considering ω(t) and $T_m$(t) the system states and $T_L$(t) is zero, it is possible to obtain the state space matrices for the DC motor, given Eqs. (4) and (5).

## 2.3 State space of a DC motor

Selecting the states previously elected and knowing that v(t) and $T_L$(t) are inputs of the system, Eqs. (4) and (5) can be rewritten as (7) and (8):

$$\dot{x} = Ax + Bu \tag{7}$$

$$y = Cx + Du \tag{8}$$

resulting in (9) and (10)

$$\begin{bmatrix} \dot{\omega}(t) \\ \dot{T_m}(t) \end{bmatrix} = \begin{bmatrix} \dfrac{-B_m}{J_m} & \dfrac{1}{J_m} \\ \dfrac{-K_eK_t}{L_a} & \dfrac{-R_a}{L_a} \end{bmatrix} \begin{bmatrix} \omega(t) \\ T_m(t) \end{bmatrix} + \begin{bmatrix} 0 & -1/J_m \\ K_t/L_a & 0 \end{bmatrix} \begin{bmatrix} v(t) \\ T_L(t) \end{bmatrix} \tag{9}$$

$$R = E\left[\nu\nu^T\right]. \tag{10}$$

Therefore, the output of the system given by Eq. (10) is only the current rotor speed, once there is no torque sensor integrated on the system. Also the matrix D is equal to zero, because there is no interference on the input of the system directly to the output.

Although the state space obtained describes the motor inputs and outputs for a continuous time application, these matrices cannot be applied on a digital discrete system. Hence, for an implementation of the digital control, it is needed to obtain the state space discrete model described by (9) and (10) with a sample period of $\tau$. For that, (7) becomes (11):

$$P[k] = \Phi\tilde{P}[k-1]\Phi^T + Q \tag{11}$$

where

$$T_m = K_t i_a(t) \tag{12}$$

and

$$J_T \frac{d\omega(t)}{dt} = T_m(t) - T_L(t) - B_T\omega(t) \tag{13}$$

## 3. System implementation

In this section, the hardware used for the development of this work as the introduction of the hardware-in-the-loop applied system is described. Also, the motor and mechanical parameters for tuning the controllers are presented.

### 3.1 System description

The hardware-in-the-loop (HIL) process is composed of the real physical system, to be controlled, activated by an embedded microcontroller which communicates directly to the virtual environment of design in the computer, as illustrated in **Figure 2**. The embedded controller is made through the BeagleBone Black, programed through the software MATLAB/Simulink, allowing fast prototyping and tuning of the controller settings and observation of the results.

As illustrated in **Figure 2**, the controller design is made through MATLAB/Simulink, where it is also possible to tune the controller parameters and observation of the results in real time, without compromising the experiment execution. The results can be observed through real-time graphs that can also be stored for later analysis.

The controller design is created through Simulink block diagram that automatically generates a code compatible to the microcontroller and downloads it to the embedded controller system. The embedded system, on the other hand, is dedicated to execution of the motor control, estimating states and sensors reading.

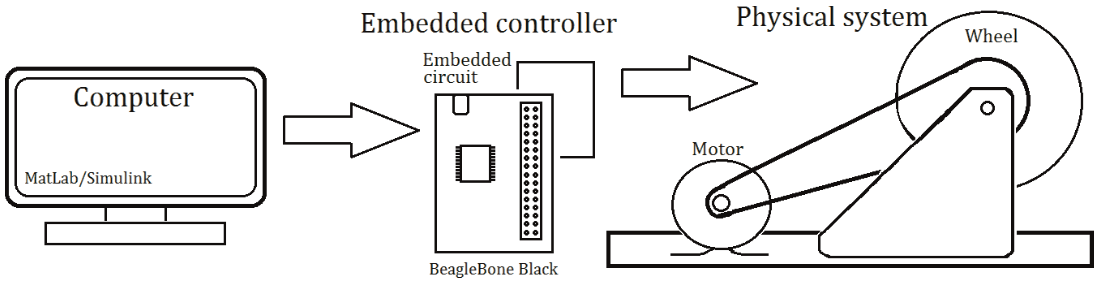

**Figure 2.**
*Illustration of hardware-in-the-loop system.*

| Ra [Ω] | La [μH] | Ke [Vs/rad] | Kt [Nmrad/s] | Bm [$Kgm^2/s$] | Jm [$Kgm^2$] |
|---|---|---|---|---|---|
| 4.735 | 344.6 | 0.0236 | 0.0424 | 0.15379 | 0.0005 |

**Table 1.**
*Parameters of the mechanical system.*

### 3.2 Hardware

A DC motor model MY1016 is used for the traction of the wheel for the studied system, which has a nominal voltage of 24 V, current of 13.7 A, maximum velocity of 2650 RPM, and nominal power of 250 W. For the system power supply, two batteries of 12 V and 7.0 Ah are used, powering the motor driver.

The motor velocity is obtained through a Hall effect sensor u1881 and six pairs of magnets attached to the motor shaft with alternated polarities, so when the motor spins, it produces six rising and falling edges per revolution.

For the crown-chain-pinion set, a 11 teeth pinion on the motor axis is used, while the wheel has a 55 teeth crown, generating a mechanical advantage of 5, meaning the motor axis has a rotational speed five times faster than the wheel; meanwhile, its torque is five times higher.

With a diameter of 18.0 cm in the wheel, the maximum speed expected by the system is 17.98 km/h, resulting in a system speed high enough for the analysis used in this work. The embedded system is controlled by a BeagleBone Black ARM Cortex-A8 microcontroller, which has up to 1 GHz of frequency. The parameters that describe the state space for the torque control are provided in **Table 1**.

## 4. Hardware-in-the-loop

This section describes hardware-in-the-loop process used in this work; it also presents the configuration and workflow of the current system.

### 4.1 MATLAB/Simulink embedded coder

The system in hardware-in-the-loop allows the fast prototyping of a control system, allowing the observation of speed, control signal graphs, and observation of states in real time, provided by the embedded board BeagleBone Black. It also provides a result much more closer to reality than the pure simulation [10].

Once the coding does not need to be tested in a complete real situation, the presence of HIL system turns out to be a low-cost solution for engineering problems [11]. Allowing the code to be directly generated and optimized by the embedded coder from the Simulink environment [12].

This processor is responsible for sensor reading and processing the actual speed, given the sensor signals provided, as estimation of the current torque value. Besides, the PWM signals controls the motor through its driver. **Figure 3** illustrates the HIL system workflow.

As shown in **Figure 3**, the design of the project is previously created in Simulink virtual environment of design. Through the use of code generator provided by MATLAB, the environment generates a compatible code to the embedded controller and downloads the code.

Once the code is downloaded, the embedded system still maintains a communication in background with the physical computer but, in second plane, without compromising the current experiment. This communication provides the value of

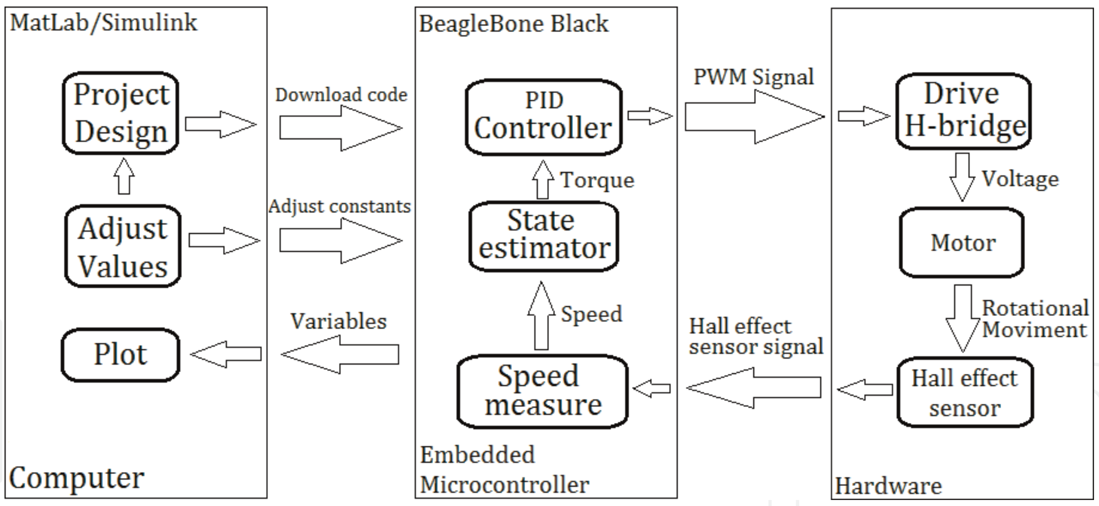

**Figure 3.**
*Schematic of a hardware-in-the-loop system design.*

specific variables, so the results can be observed in real time. Also, it is possible that new adjusts in variables inside the controller designed can be transferred to the embedded controlled in the current experiment, without the necessity of terminating the simulation and starting again.

The embedded board is responsible for all the execution of the digital controller; hence, the speed sensor is an input to the microcontroller, which processes the information and estimates the respective speed; this speed works as an input to the Kalman filter that works not just as an estimator for the torque but also filters the noise generated by speed sensor. By adding a Kalman filter, the vibration of the system is reduced, creating a much more efficient controller [13]. As a result, the current torque works as an input for the feedback of a PID controller, producing PWM signals for the motor drive.

For the physical system, the input is the PWM signals that provide not just the information of the amplitude in which the motor will be activated but also its direction, depending on the current PWM activated. This signal activates the interface of activation of the motor producing the correction on the motor torque and speed. As the motor axis rotational speed produces a rotational movement on the motor axis, the Hall effect speed sensor produces a signal that works as an output, providing a feedback for closed-loop control system [14].

## 4.2 MATLAB/Simulink configuration

For the implementation of the HIL system, it is needed to follow some configuration steps. The first step is to assure that MATLAB has support for the desired embedded board; in this present research, it was chosen to use the BeagleBone Black board. The support packages can be found in Add-Ons → Get Hardware Support Packages.

Those which are necessary to successfully communicate with MATLAB are ARM Cortex-A: Generate code optimized for ARM Cortex-A processor, providing the possibility for MATLAB to generate code compatible to an ARM processor Cortex-A.

BeagleBone Black development board interact with MATLAB which in turn generates an optimized code necessary for HIL implamentation.

After the download and installation of each package, it is needed to connect the board to MATLAB and open Simulink configurations in model configuration parameters to properly set the communication with the board, as shown in **Figure 4**.

The configuration parameters are set automatically once the previous installations are successfully done; however, for high demanding control system in HIL which are time dependent, it is necessary to run the communication in the second plane. To configure the HIL in the background, it is necessary to check the option Run external mode in a background thread in the External mode menu.

Another parameter configuration menu panel that is very important for a successfully code generation is depicted in **Figure 5**.

Language C and the system target file, ert.tlc, are necessarily set for a successful download of the designed control system, so the code can be understood by the embedded board. This menu also allows the generation of the code for external applications without running in HIL.

For the application of the HIL, after the proper design of the controller has been set, it is necessary to change the mode of execution from normal to external, as

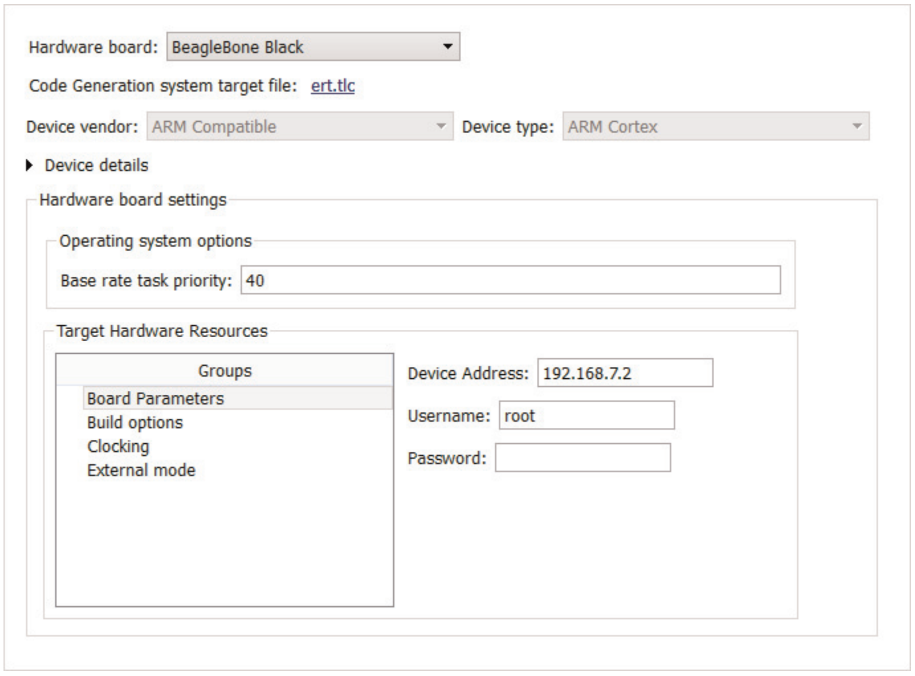
Hardware board: BeagleBone Black
Code Generation system target file: ert.tlc
Device vendor: ARM Compatible
Device type: ARM Cortex
Device details
Hardware board settings
Operating system options
Base rate task priority: 40
Target Hardware Resources
Groups
Board Parameters
Build options
Clocking
External mode
Device Address: 192.168.7.2
Username: root
Password:

**Figure 4.**
*Interface of parameter configuration for the hardware communication.*

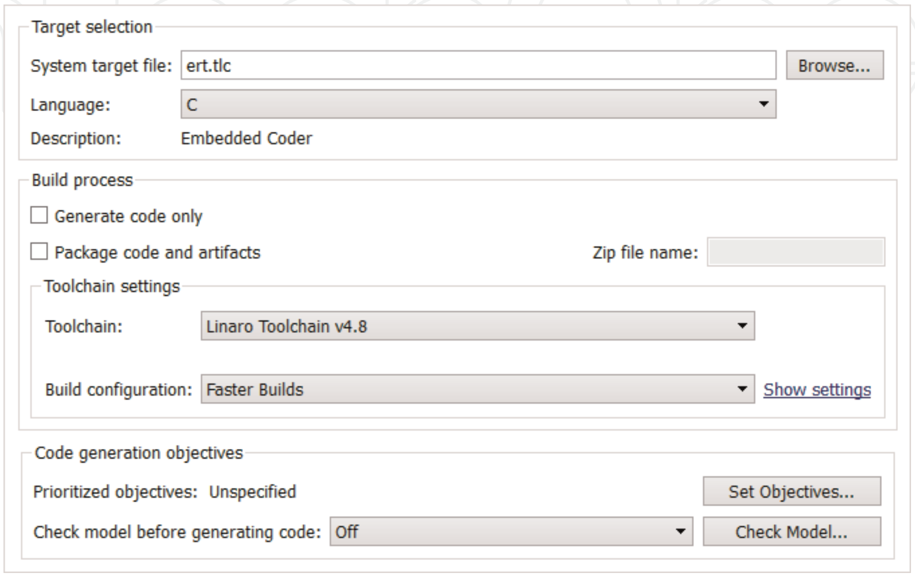
Target selection
System target file: ert.tlc
Browse...
Language: C
Description: Embedded Coder
Build process
☐ Generate code only
☐ Package code and artifacts
Zip file name:
Toolchain settings
Toolchain: Linaro Toolchain v4.8
Build configuration: Faster Builds
Show settings
Code generation objectives
Prioritized objectives: Unspecified
Set Objectives...
Check model before generating code: Off
Check Model...

**Figure 5.**
*Interface of parameter configuration for the code generator.*

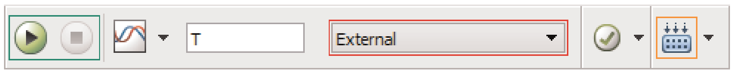

**Figure 6.**
*Illustration for running the code in HIL.*

shown in **Figure 6**, and run the simulation button, instead of deploying on the hardware. While the code is running, Simulink will automatically generate a code and print a report of the code generated and also automatically downloads the code into the embedded board and starts the experiment.

## 5. Controller design

The torque controller as its design as a Simulink block diagram and also the torque-generated speed controller are described in this section, where it also presented its design for application.

### 5.1 Torque controller

Motor's speed could present a high slipping ratio and some times lost of traction, in extreme situations, once the controller has no information about traction and real acceleration, which might be dangerous.

**Figure 7** shows a design of a torque controller for the motor, in which the reference is an input for the system. The sensor signal is then read and processed and assigned as one of the Kalman filter inputs, aside from the control signal. The Kalman filter is responsible for filtering noises from the sensor signal and also estimates the current torque, since there is no torque sensor attached to the system.

As a torque control, the output of the Kalman filter which estimates the real current speed is ignored, and the current torque is assigned as the feedback for the controller. The resulting error signal is the input of the discrete PID controller, resulting on a control signal.

This control signal is a feedback to the Kalman filter and also it is normalized, and after passing through a block of separation and generation of the protected PWM signals, the system outputs both PWM signals as generated.

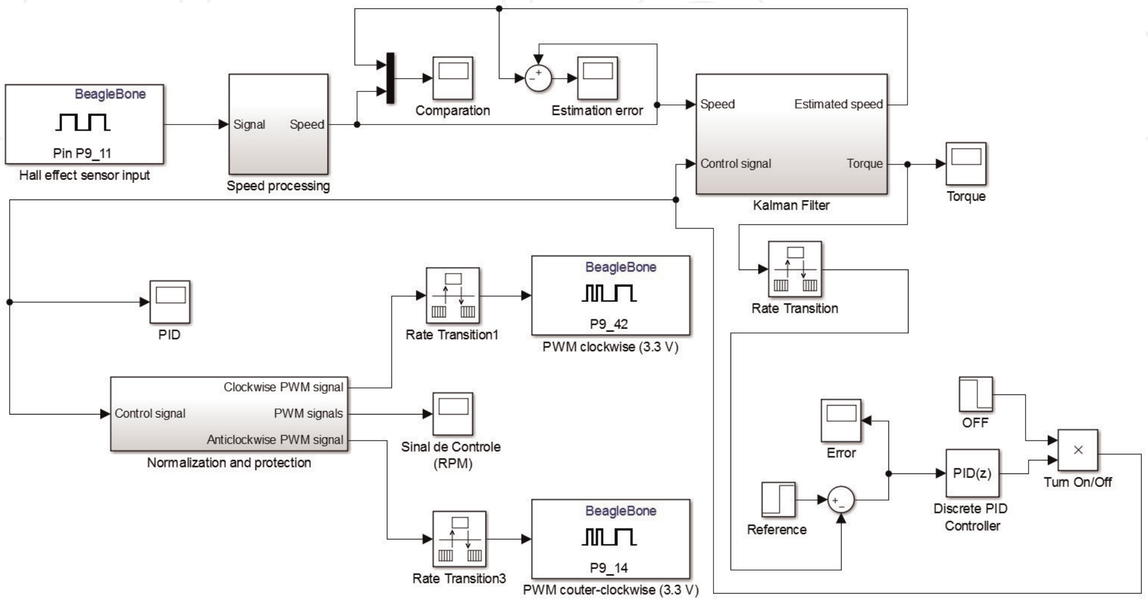

**Figure 7.**
*Torque controller design.*

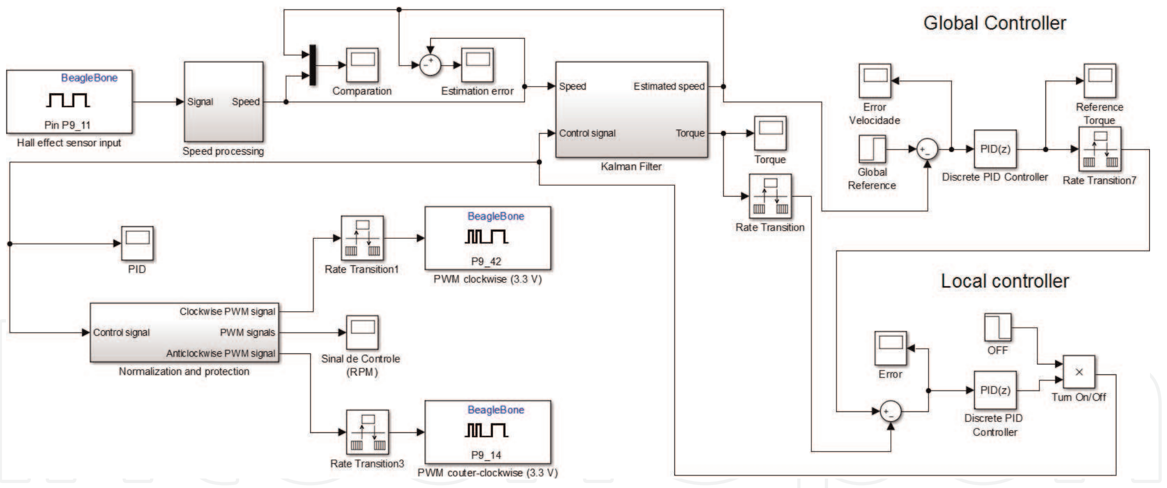

**Figure 8.**
*Torque-generated speed controller design.*

### 5.2 Torque-generated speed controller

A second option for the solution of the pure speed controller problems is the application of a torque-generated speed controller, where the controller sets the speed as desired, but it is tuned by a torque controller. This method keeps the torque monitored and can be easily limited and used for noticing traction of the wheel.

The significant difference from the torque controller to the design of the present controller is that the estimated speed is not ignored any longer, but its value is used as a feedback of a global controller, where the speed is controlled by another discrete PID controller.

The resulting signal of this global controller is a torque reference for the previous controller, called local controller. This system then provides a stable speed controller with a monitored torque that provides more safety for the system (**Figure 8**).

## 6. Results

The results of this work are presented and discussed in this section. Each result is presented with graphs.

### 6.1 Torque controller

The torque controller for a free wheel has low stability, as higher the speed lowers the current torque, but the tuning of a good torque controller allows the increase of stability for other types of controllers.

It is noticeable in **Figure 9** that the torque controller manages to stabilize around 0.15 and 0.02 Nm, in different states of time, which happens because of the control correction.

The controller actuation can be seen in **Figure 10**, where the fast response of the controller to correct the stability region in a small period of time is noticeable.

Also it is possible to observe in **Figure 11** the error signal, where it notices the correction of the controller with the time, and visualize the challenge of stability of a torque control in the absence of traction.

### 6.2 Torque-generated speed controller

Once the pure torque control proves to be challenging, the proposition of developing a torque-generated speed controller seems to produce better results adding the best characteristics of each controller. **Figure 12** shows the signal reference of the local controller generated by the global controller and notices the stability of the current system.

As the response to the reference signal, the local controller alters the torque-generated speed controller which controls the final speed. **Figure 13** shows the current torque of the system during the experiment, and it is possible to notice not only the stability of the controller but also an oscillation of the state resultant from the chain that attaches the motor and wheel.

The variation of the resultant torque is caused by the control signal; this signal activates the motor through a driver using two PWM signals that varies from 0 to 1, and each one controls the direction of the motor activation. **Figure 14** shows the PWM signals and its values.

It can be noticed that, periodically, the controller generates a reverse pulse of PWM, caused by the mechanical system chain oscillation, but the controller proves to be stable as the result can be seen in **Figure 15**.

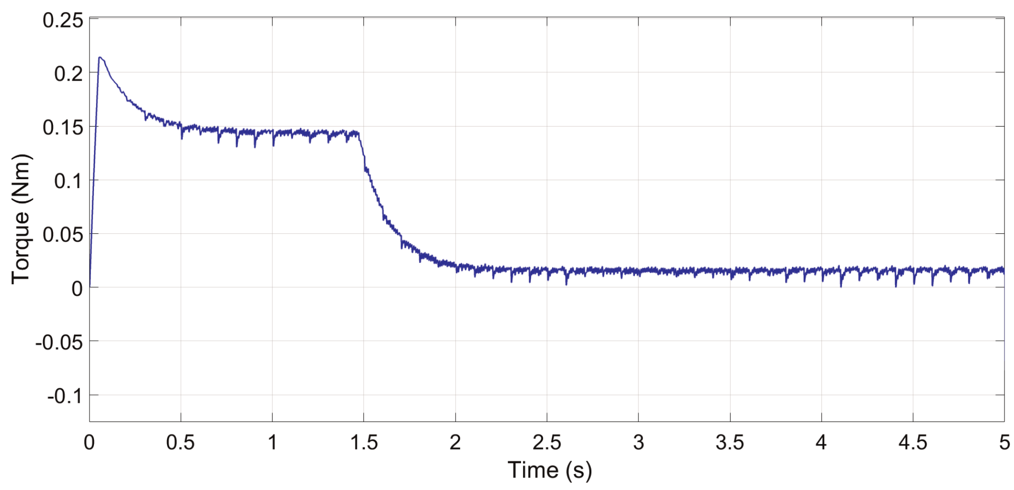

**Figure 9.**
*Torque generated.*

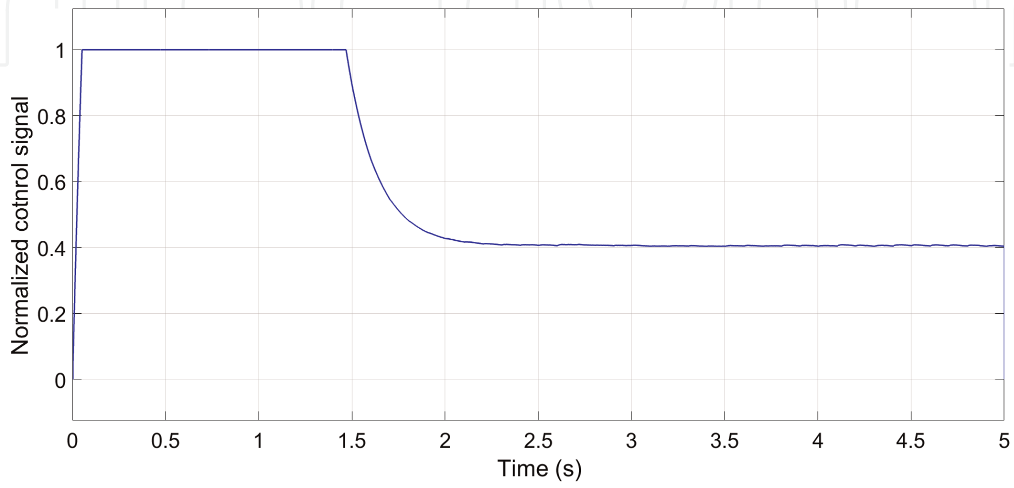

**Figure 10.**
*Normalized control signal.*

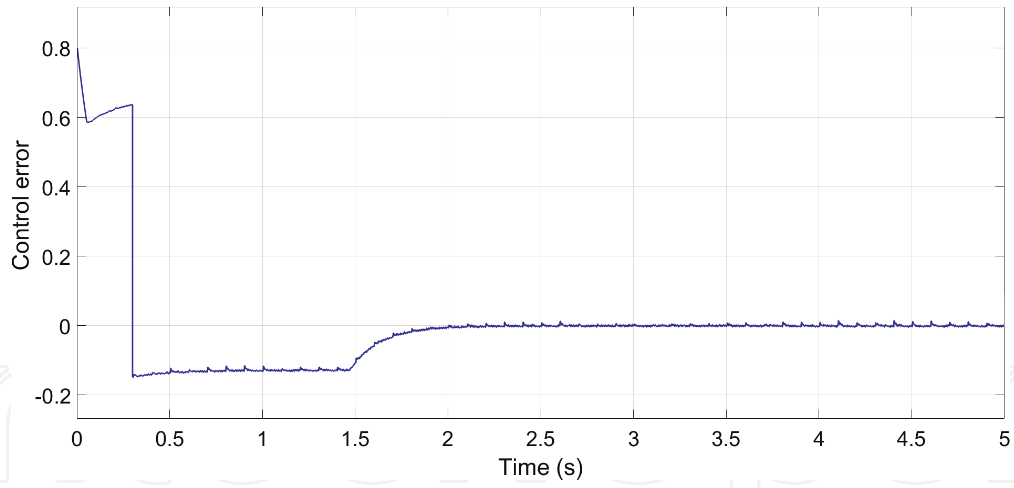

**Figure 11.**
*Torque controller error.*

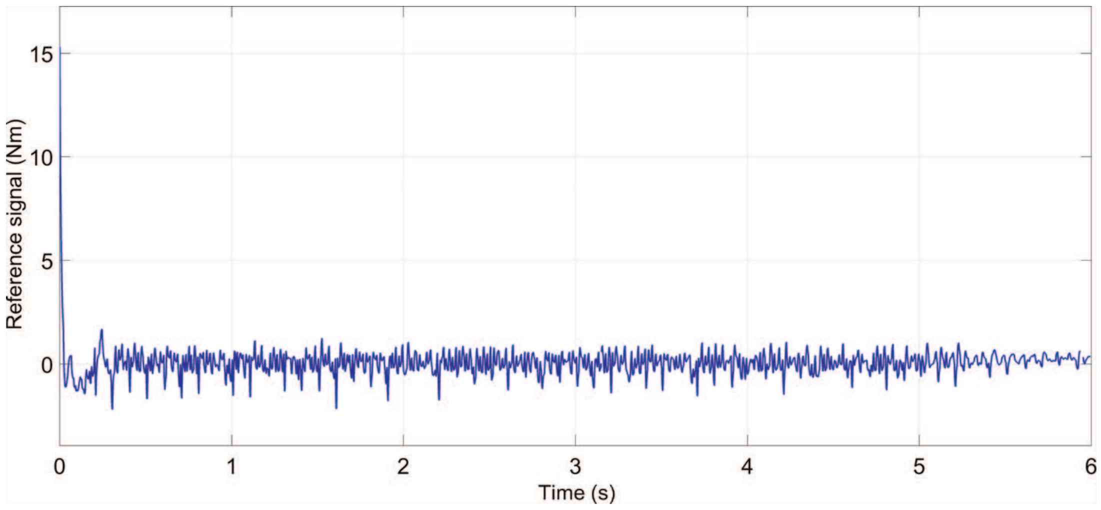

**Figure 12.**
*Reference signal.*

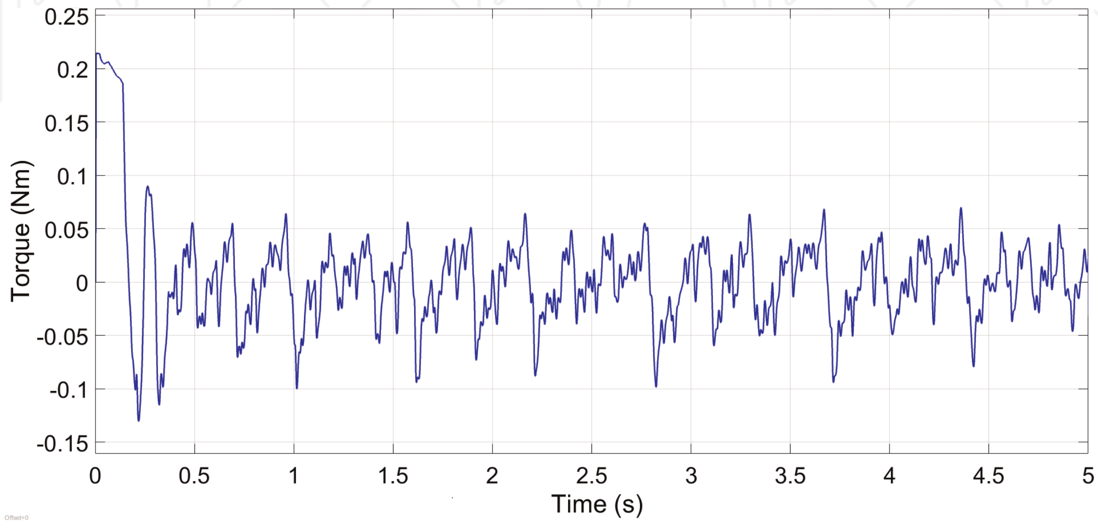

**Figure 13.**
*Torque.*

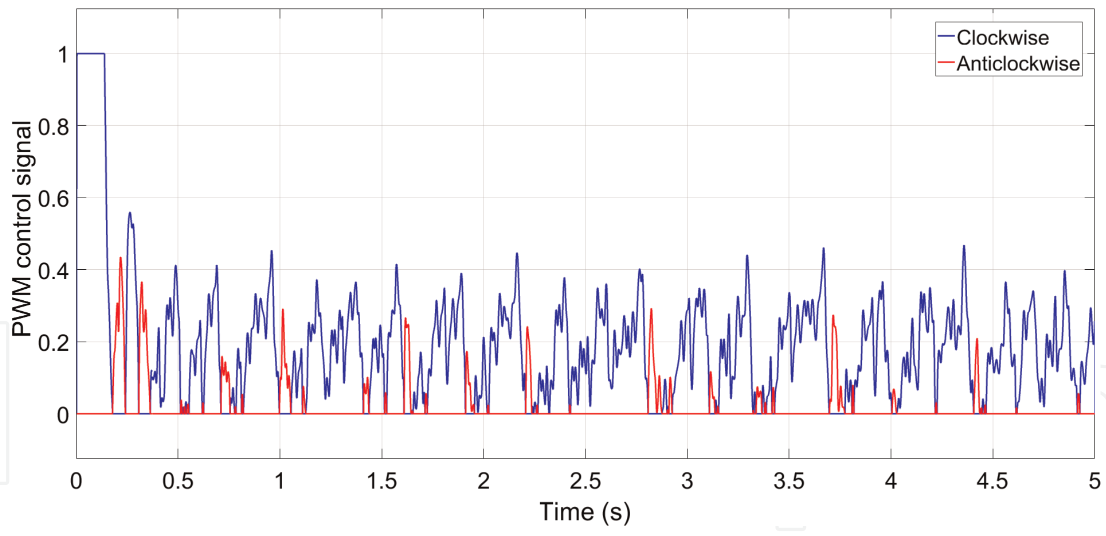

**Figure 14.**
*Control PWM signal.*

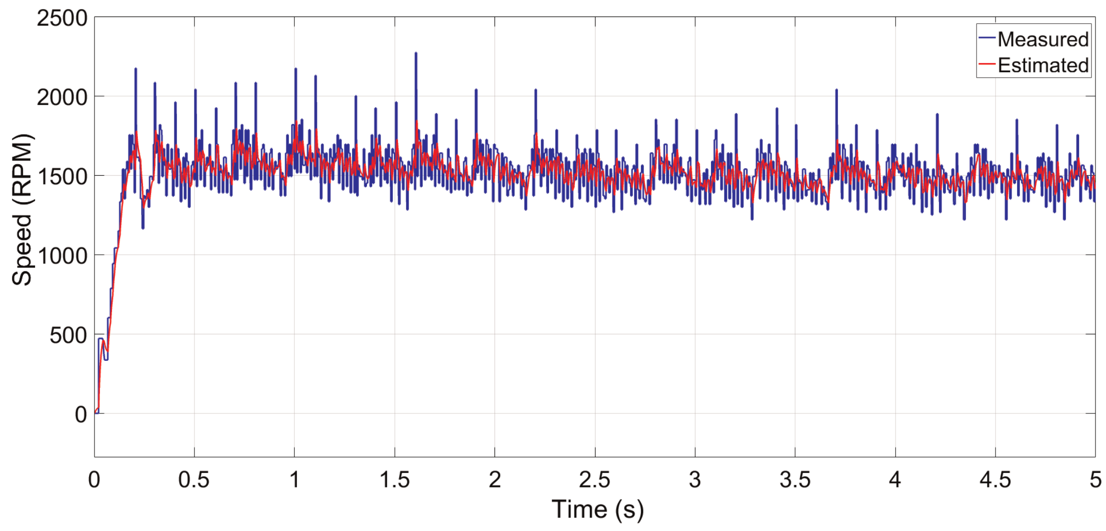

**Figure 15.**
*Speed.*

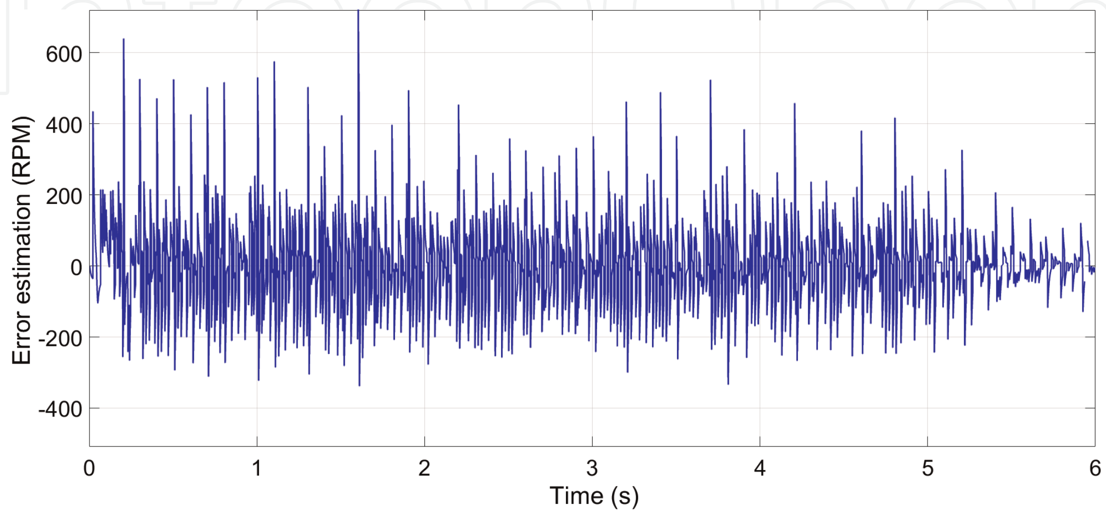

**Figure 16.**
*Error Kalman.*

**Figure 15** shows the resulting signal of the controller that stabilizes on the desired speed. The mechanical oscillation is still present, but the controller transient shows to be more smooth than a pure speed controller, allowing a better control of slippery traction situations.

Also **Figure 16** proves the use of a Kalman filter instead of a simple state observer for the torque estimation. The figure shows the noise removed from the speed reading without generating an offset of this signal, resulting in a more stable controller.

## 7. Conclusion

This work presents a hardware-in-the-loop system as a solution for fast prototyping, tuning, observation of results, and coding. The process of design by the use of Simulink block diagrams and then automatically generating a code results in a much easier and faster way of designing a code for a complex controller, once the process of coding human error is completely eliminated. Also the possibility of adjusting variables in real time turns out to be a good tool for minor tuning of the controllers, when differences between the real system differs from the mathematical model. Also this work presents a torque controller for a traction motor to solve the slip problem created by high responsible speed controllers, and finally, this work concludes that a torque-generated speed controller shows to be more reliable than both the previous solutions, once the designer has more control over the traction of the vehicle without losing the control of the speed of the vehicle.

## Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brazil (CAPES)—Finance Code 001.

## Author details

Alex Archela, Dario Guilherme Toginho and Leonimer Flávio de Melo*
State University of Londrina (UEL), Londrina, PR, Brazil

*Address all correspondence to: leonimer@uel.br

## References

[1] Yin D, Hori Y. A new approach to traction control of EV based on maximum effective torque estimation. In: 34th Annual Conference of IEEE Industrial Electronics; 2008

[2] Yin D, Hori Y. Traction control for EV based on maximum transmissible torque estimation. International Journal of Intelligent Transportation Systems Research. 2010;**8**(1):1-9

[3] Magallan GA, De Angelo CH, Garcia GO. Maximization of the traction forces in a 2WD electric vehicle. IEEE Transactions on Vehicular Technology. 2011;**60**(2):369-380

[4] Heikkinen J, Serykh EV, Minav T. A control strategy for an autonomous differential drive mobile robot. In: 2017 IEEE II International Conference on Control in Technical Systems (CTS); 2017

[5] Kuntanapreeda S. Traction control of electric vehicles using sliding-mode controller with tractive force observer. International Journal of Vehicular Technology. 2014;**2014**:829097

[6] Mutoh N. Driving and braking torque distribution methods for front- and rear-wheel-independent drive-type electric vehicles on roads with low friction coefficient. IEEE Transactions on Industrial Electronics. 2012;**59**(10): 3919-3933

[7] Li B, Goodarzi A, Khajepour A, Chen S, Littkouhi B. An optimal torque distribution control strategy for four-independent wheel drive electric vehicles. Vehicle System Dynamics. 2015;**53**(8):1172-1189

[8] Hori Y. Future vehicle driven by electricity and control-research on four-wheel-motored "UOT Electric March II". IEEE Transactions on Industrial Electronics. 2004;**51**(5):954-962

[9] Kalman RE. A new approach to linear filtering and prediction problems. Journal of Basic Engineering. 1960; **82**(1):35-45

[10] Guazzelli PRU, De Oliveira CMR, De Castro AG, Pereira WCA, De Aguiar ML. Electric vehicle hardware-in-the-loop simulation with differentiator optimised by genetic algorithm. In: 2016 12th IEEE International Conference on Industry Applications (INDUSCON); IEEE; 2016

[11] Araújo JFB, Pedrosa HM, Rodrigues MCBP, Barbosa PG. Real-time "hardware-in-the-loop" simulation of components of an electric vehicle powertrain: Modeling and implementation. In: 2016 12th IEEE International Conference on Industry Applications (INDUSCON); IEEE; 2016

[12] Garrido S, Moreno L. Mobile robot path planning using Voronoi diagram and fast marching. In: Robotics, Automation, and Control in Industrial and Service Settings (IGI-Global); 2015

[13] Archela A, Toginho DG, de Melo LF. Torque control of a DC motor with a state space estimator and Kalman filter for vehicle traction. In: 2018 13th IEEE International Conference on Industry Applications (INDUSCON); São Paulo, Brazil; 2018. pp. 763-769. DOI: 10.1109/ INDUSCON.2018.8627285

[14] Philips CL, Nagle HT. Digital Control System Analysis and Design. 4th ed. Upper Saddle River, NJ, USA: Prentice Hall Press; 2007. ISBN: 0130812226, 9780130812223