Board 55: Work in Progress: Design and Implementation of an Advanced Electric Drive Laboratory using a Commercial Microcontroller and a MATLAB Embedded Coder

Mr. Bhanu Babaiahgari, University of Colorado, Denver

Mr. Bhanu Babaiahgari finished his master's program in 2015, at the University of Colorado Denver. He started his PhD at University of Colorado Denver supervised by Dr. Jae-Do Park in 2016. Since then he has been teaching Electric drives and Energy conversion laboratory as part-time grad instructor. He is an active researcher at Dr. Park's Energy and Power lab under Energy Conversion Research Force (ECRF). His current research interests are DC shipboard power systems protection and stability analysis, power electronics and coordinated energy management.

Dr. Jae-Do Park, University of Colorado, Denver

Jae-Do Park received his Ph.D. degree from the Pennsylvania State University, University Park, in 2007. Park is currently an Assistant Professor of electrical engineering at the University of Colorado, Denver. He is interested in various energy and power system research and education areas, including electric machines and drives, energy storage and harvesting systems, renewable energy sources, and grid-interactive distributed generation systems. Prior to his arrival at the University of Colorado, Denver, Park worked for Pentadyne Power Corporation in California as Manager of Software and Controls, where he took charge of control algorithm design and software development for the high-speed flywheel energy storage system. He also worked at the R&D Center of LG Industrial Systems, Korea, where he developed induction machine drive systems as a Research Engineer.

Work in Progress: Design and Implementation of an Advanced Electric Drive Laboratory using a Commercial Microcontroller and a MATLAB Embedded Coder

Abstract

This paper presents a design for an advanced instructional electric drives laboratory using commercial Texas Instruments (TI) C2000 microcontroller and MATLAB-based Embedded Coder toolbox. The main objective of this project is to familiarize students with real-time implementation of advanced electric machine drive concepts such as field-oriented vector control by programming high-performance industrial microcontrollers. Hands-on experience with electric drives is also provided through the operation and control of machines. This laboratory course, designed to follow the advanced lecture course on electric drives, aims to improve students' understanding of theory. Experiments, hardware and instruments for the proposed laboratory course are discussed.

Introduction

In the past, due to the convenience of their torque and speed control, DC machines were used most widely for adjustable speed applications. However, with the development of vector control, also known as field-oriented control (FOC), AC machines driven by variable-voltage variable-frequency (VVVF) inverters have become the norm[1,2,3]. AC machines, especially induction machines, are inexpensive and more rugged than their DC counterparts. This shift from DC to AC machines has continued due to the advancement of power electronics devices as well. With FOC techniques, an induction machine can be modeled like a separately excited DC machine through a series of transformations, so that DC machine-like control techniques can be applied to obtain good transient performance. Recently, efforts have been made to implement DSP-based electric machine and drive laboratories[4,5], but the required software and equipment have been unaffordable.

While TI provides software solutions to drive various machines via hardware, it is still challenging for students to understand the complicated interlinking of modules with many features and functions, typically in C language. However, today's students are familiar with writing MATLAB code and developing Simulink models for their courses. Graphical representation of Simulink programming is easier to learn and more intuitive. Furthermore, MATLAB has developed Embedded Coder, a toolbox that can translate MATLAB code and

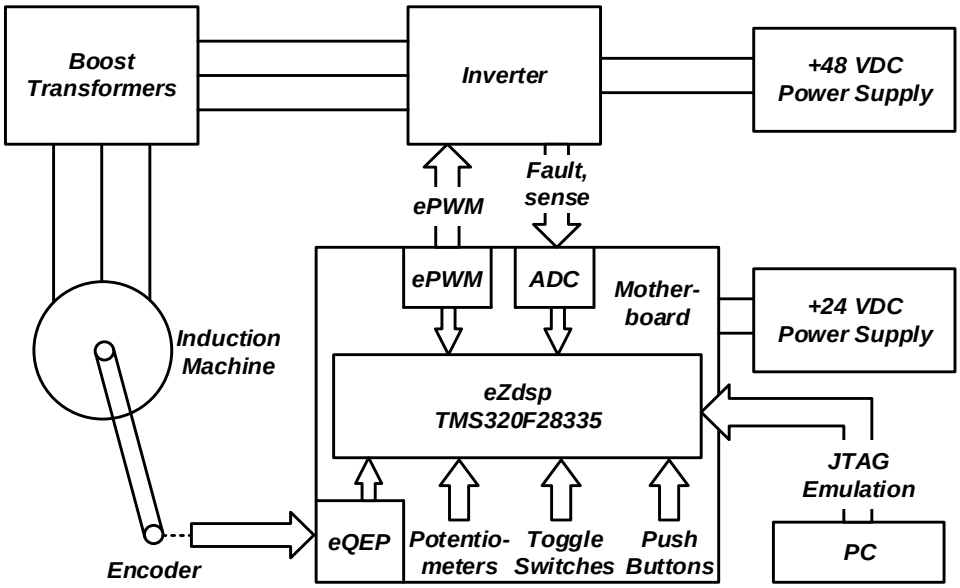


Figure 1: Functional diagram of the experimental hardware setup.

Simulink models into C and C++ for embedded microcontrollers. In this context, Simulink is used to design and model complex machine control algorithms and translate them into C code using the Embedded Coder toolbox.

This paper presents the design and implementation of an advanced electric drive laboratory using a commercial microcontroller development kit and MATLAB Embedded Coder, including hardware components, laboratory equipment setup, experiment sessions, and prototype test results. The laboratory is designed for graduate and advanced undergraduate students with moderate programming skills. Although it has not yet been offered, we expect 10-15 students in this course.

Hardware Design

The hardware design for the proposed laboratory focused on cost-effectiveness and reliability, as well as establishing a setup that is both realistic and safe. The design aims to place all equipment (i.e., power supplies, inverter, microcontroller, transformers and machine) on the same platform, so students can perform experimental tasks without the need for any modifications. The complete system and a functional diagram of the setup appear in Figures 1 and 2, respectively.

1. Microcontroller and Voltage Source Inverter

TI provides real-time control microcontroller units (MCUs) that use a 32-bit 28x-core to provide advanced digital signal processing in industrial system applications. In this work, an eZdsp

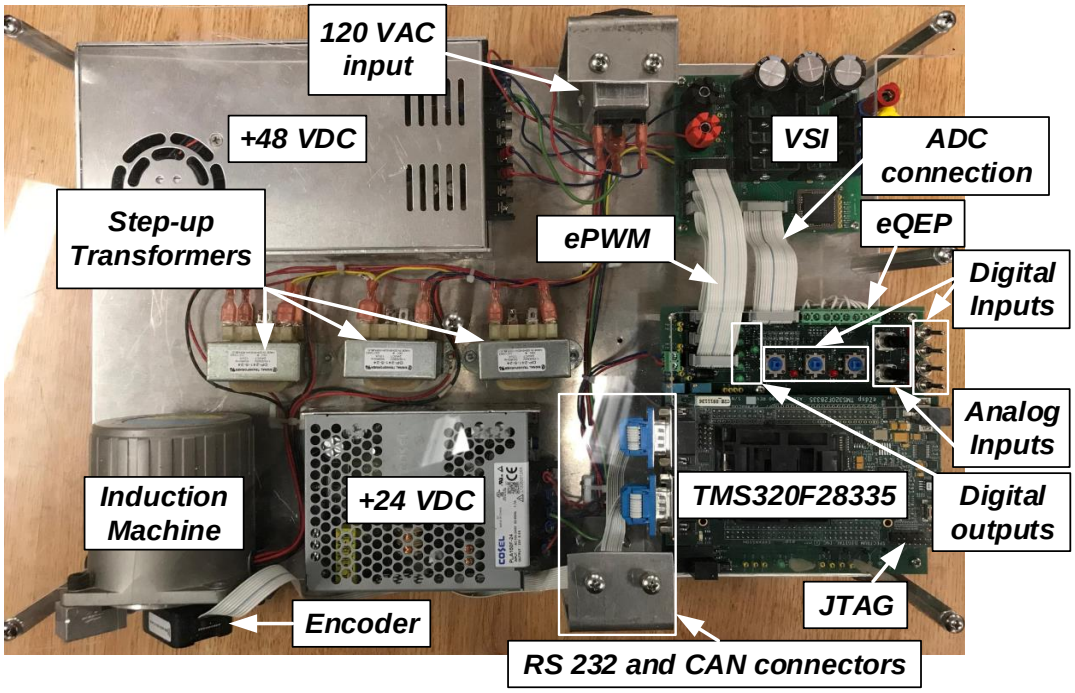


Figure 2: Experimental hardware setup.

TMS320F28335 microcontroller DSP starter kit (DSK) was selected to perform the experiment sessions. It features 150 MHz operating speed, on-chip 32-bit floating-point unit, 68 kbytes on-chip RAM, 512 kbytes on-chip flash memory, 12-bit analog-to-digital converter (ADC) with 16 input channels, enhanced pulse-width modulation (ePWM) channels, RS-232 connector with line driver, multiple expansion connectors for analog input/output (I/0) pins, and an on-board JTAG emulation connector. Furthermore, the DSK has an enhanced quadrature encoder (eQEP) module, intended to acquire position, direction, and speed information from rotating machines for high-performance motion control applications.

The hardware setup also includes a custom voltage source inverter (VSI) that uses a 48 VDC supply and 3.3V ePWM digital signals from the microcontroller to generate up to 24 VAC peak output. The ePWM digital signals drive the six MOSFET devices using an IRS2336 integrated gate driver. Three high-voltage discharge capacitors are placed before the AC output stage of the VSI, which maintains solid DC input voltage. Furthermore, the VSI has one voltage sensor for DC voltage and three current sensors that feed the microcontroller's ADC channels.

To accommodate the microcontroller and its associated modules (ADC, ePWM, and eQEP), a custom motherboard powered by a 24 VDC supply was developed. To implement user control interfaces for speed, run/stop, and rotational direction, it has additional control inputs – potentiometer, toggle and pushbutton switches, respectively. The bill of materials is listed in Table 1.

Description	Part number	Manufacturer	Quantity	Price($)
Induction machine	ACP-M-4IK25A-SU	PeeiMoger	1	89.24
Microcontroller	eZdsp TMS320F28335	Texas Instruments	1	549.00
Motherboard	Custom	-	1	199.75
Inverter	Custom	-	1	117.00
Step-up transformers	DP-241-5-24	Signal Transformer	3	34.80
+48 VDC supply	SP-320-48	MEAN WELL USA Inc.	1	85.73
+24 VDC supply	PLA150F-24	Cosel USA, Inc.	1	65.38
Encoder	ENC-A5DI-1024-313-H-G	Anaheim Automation	1	64.00
Hex standoffs	Generic	-	10	4.31
22 AWG wire	Generic	-	1	10.00
Polycarbonate panel	Generic	In-house machine shop	1	5.00
Terminal connectors	Generic	-	20	5.20
Miscellaneous	-	-	-	20.00
Total				**$1,249.41**

Table 1: Bill of materials for one experimental setup. It should be noted that the price could vary according to the purchasing quantity and time.

2. Induction Machine and Encoder

In this laboratory, a 4-pole AC induction machine (ACP-M-4IK25A-SU, PeeiMoger) was selected. It comes in a small frame with standard 6-lead configuration and is wired to run on a 3, 60Hz, 230VAC supply. The machine is rated at 25 W at 1625 rpm, and mounted on an aluminum plate built by the in-house machine shop. Additionally, a differential optical encoder is attached to the end of the rotor shaft to measure rotational speed, which is powered through the motherboard connection. The encoder feeds the signals to the microcontroller's eQEP module, which counts quadrature pulses and measures time.

3. Step-Up Transformers

The maximum peak output voltage generated using VSI is half of the DC input voltage, which is 24 VAC. Since the machine requires 230 VAC, three single-phase 1:10 transformers are used to step up the voltage level for each phase. Additionally, the transformers provide galvanic isolation to the machine.

4. DC Power Supplies

In this laboratory, two AC-DC switching power supplies (48 and 24 VDC) are used to power the VSI and motherboard. Both power supplies use 1, 60Hz, 100-240VAC input and connect to a 120VAC power socket through a switch, so that power can be cut when needed.

5. Development Tools and Instruments

TI has a software program called Code Composer Studio (CCS) [6], an integrated development environment (IDE) tool for their processors and controllers. The software includes a suite of tools, such as optimizing C/C++ compiler, source code editor, project build environment, debugger, and many other features that are used to develop and debug embedded applications; however, users are required to develop complex C/C++ code. Alternatively, MATLAB Embedded Coder – along with support packages for TI microcontrollers – generates C and C++ code from Simulink models and MATLAB functions and sends it to CCS, where a real-time executable is generated and downloaded to the target hardware. Figure 3 illustrates the MATLAB TI embedded software design flow. Furthermore, a Simulink model is shown in Figure 4, and the C code generated by the Embedded Coder appears in the Appendix.

A computer with pre-installed firmware development packages is provided to each student, as well as generic industrial instruments like the digital multimeter and oscilloscope.

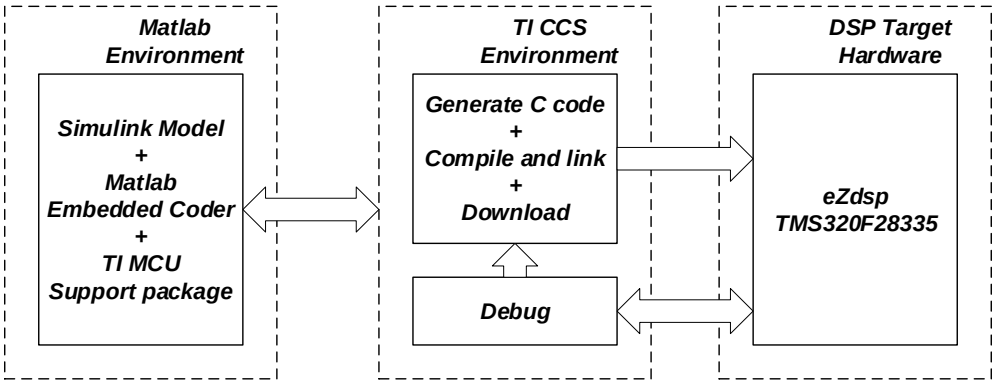


Figure 3: MATLAB embedded coder with TI support package software design flow.

6. Safety Aspects

Safety and protection measures will be taken at all times while students are performing experiments. Motor connections are pre-soldered and completely insulated to avoid short-circuiting. The hardware setup is covered with a transparent polycarbonate panel to avoid accidental contact. Fully insulated quick connect crimp terminal connectors are used for all wiring for added safety. All electrical connections ensure good contact, and no copper is exposed.

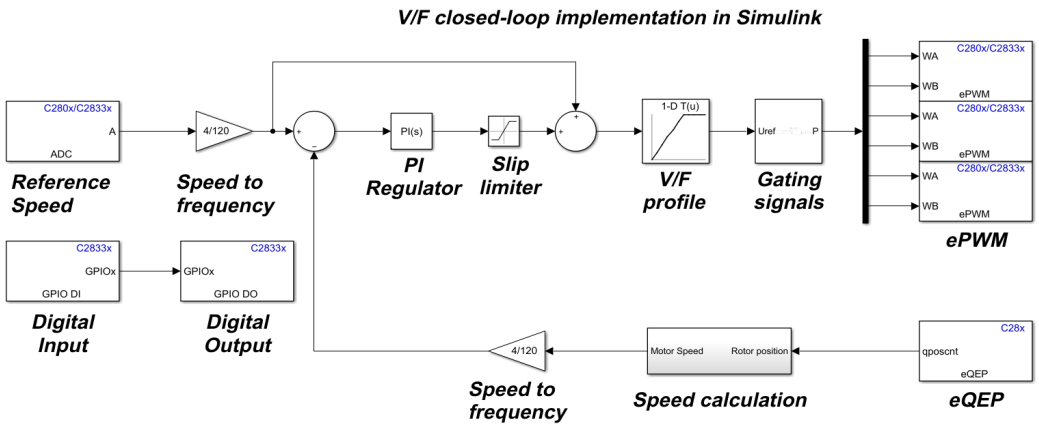


Figure 4: V/F closed-loop experiment session implementation in simulink.

Experiments

The proposed advanced electric drives laboratory comprises five experiment sessions, briefly discussed here. Each session requires students to build a model in MATLAB/Simulink using the equations provided at the start of the session. Then the code is debugged and deployed to the target microcontroller using the MATLAB Embedded Coder. Troubleshooting is done as needed. Students are provided with step-by-step procedures and exercises to meet learning objectives. At the end of each chapter, students are asked to solve problems using components of the example model. Each student submits an analysis of the results they obtained from the model and their measurements for the tasks.

1. Induction Machine Parameter Identification

To apply the vector control algorithm, machine parameters should be identified for setting of gains, reference and feed-forwarding values of regulators. Locally sensed values, introduced in[6], are utilized to estimate machine parameters in this session. According to this theory, students are asked to implement a particular machine operating condition using the microcontroller and inverter so that the parameters can be estimated. For example, the stator transient inductance (σL_s) is estimated by turning on the top switch of a-phase only in the VSI for a time duration of t, so that the peak a-phase current is around a rated value. Using the sensed values of VSI voltage and current, σL_s can easily be determined. This step is repeated 3-5 times to find the average value of estimated σL_s.

2. Volts-per-Hertz (V/F) Control

The objective of this experiment session is to implement and test the Volts-per-Hertz (V/F) control in open- and closed-loop modes. An overall block diagram implementing a closed-loop V/F induction machine drive appears in Figure 5. The measured frequency (F_r) obtained from the

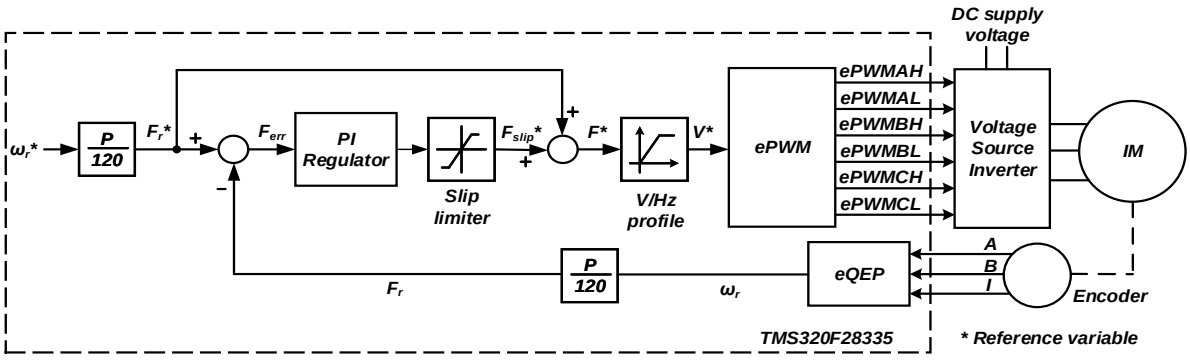


Figure 5: Block diagram of V/F closed-loop drive implementation.

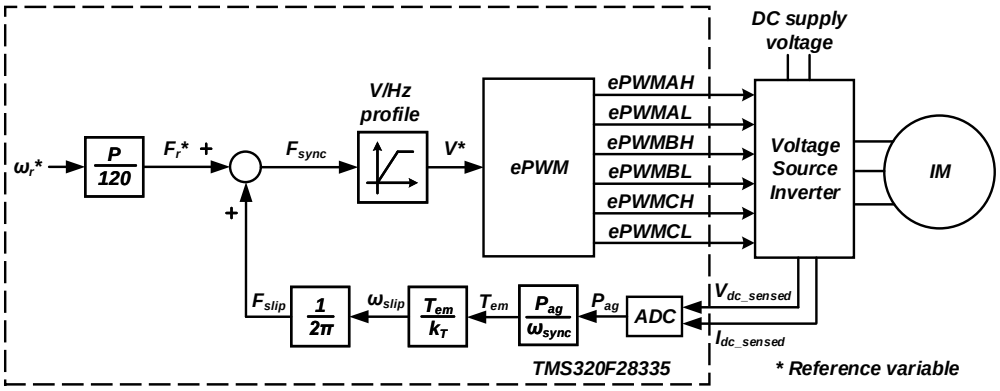


Figure 6: Block diagram of slip compensation implementation.

eQEP module is compared with a reference frequency value (F_r^*) to generate an error (F_{err}) value. Using a PI regulator and a feed-forward signal, the reference value of frequency (F^*) is generated, which is then used to calculate the voltage reference value (V^*) by looking up a preset V/F profile. Finally, the ePWM module in the MCU is programmed to generate the PWM gating signals to the VSI, which generates the 3-ϕ AC voltage. The test is conducted for different ω_r^*; to observe the V^* trend.

3. Slip Compensation

The objective of this experiment session is to implement and test the slip compensation control technique, in which the slip speed (ω_{slip}) and slip frequency (F_{slip}) are estimated using voltage (V_{dc}) and current (I_{dc}) measurements sensed from VSI. The estimated F_{slip} is then added with the frequency reference (F_r^*) to obtain the machine synchronous speed (F_{sync}). Voltage reference value (V^*) is calculated by looking up the V/F profile and applying the same approach as used in Session 2 for generating 3-ϕ AC voltage. The overall block diagram implementing a slip compensation technique appears in Figure 6.

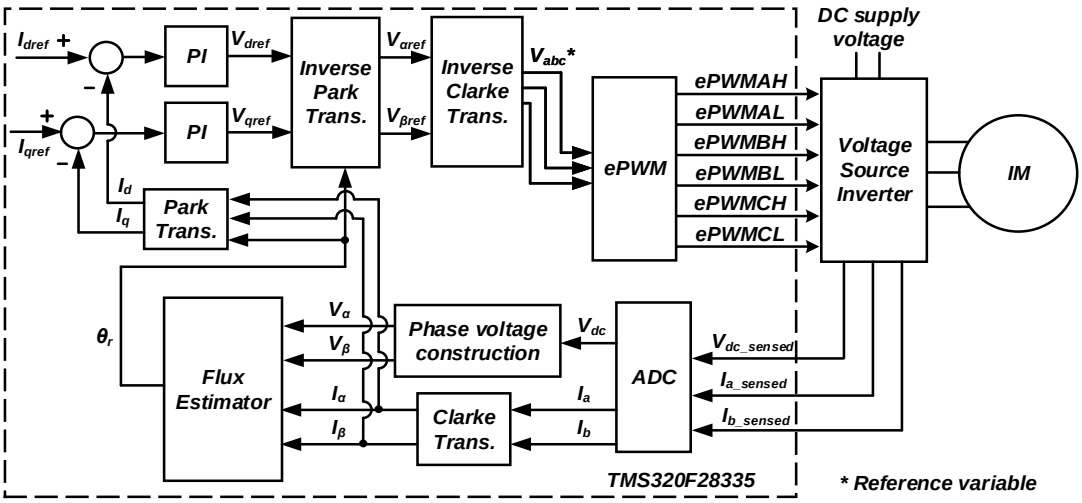
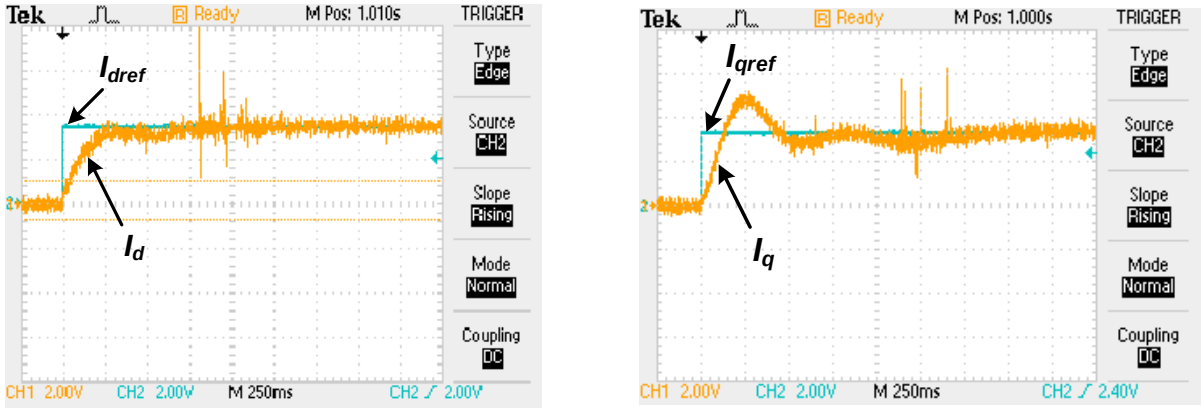


Figure 7: Block diagram of rotor-flux oriented vector control technique implementation.



(a) (b)

Figure 8: Rotor-flux oriented vector control experimental results. (a) shows the response of i_d current for a step change in i_{dref} current. (b) shows the response of i_q current for a step change in i_{qref} current.

4. Rotor-Flux Oriented Vector Control

Topics covered in this session include the implementation of Clarke and Park transformations to convert the phase currents (I_a and I_b) in abc reference frame to dq currents (I_d and I_q) in rotating dq reference frame. A flux estimator block is modelled to estimate rotor position using the voltages (V_α and V_β) and currents (I_α and I_β) in stationary $\alpha\beta$ reference frame. Two current regulators are designed to regulate I_d and I_q, respectively, to a set reference value I_{dref} and I_{qref}.

The output of PI regulators (V_{dref} and V_{qref}) is then utilized to obtain $V_{\alpha ref}$ and $V_{\beta ref}$ by applying an inverse Park transformation, which are then used as a source to generate PWM gating signals. The overall block diagram and experimental results of current control implementation are shown in Figures 7 and 8, respectively. As can be seen, the currents I_d and I_q respond for a step change in reference currents I_{dref} and I_{qref} without steady-state error. Gain tuning and anti-windup can be included as additional modules.

5. Sensorless Speed Control

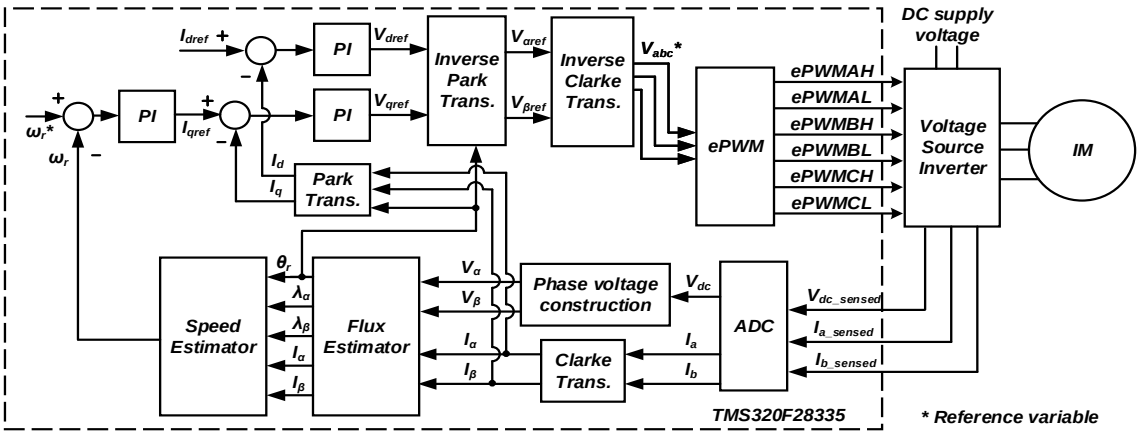


Figure 9: Block diagram of speed control technique implementation.

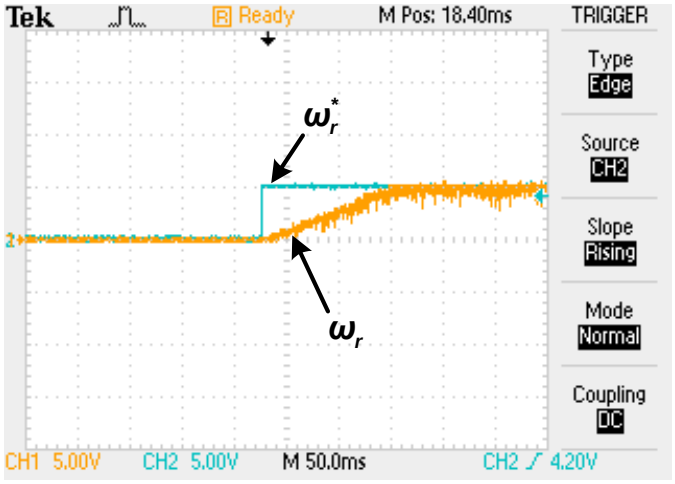


Figure 10: Sensorless speed control experimental results showing the induction machine speed (ω_r) response for a step change in the reference speed (ω_r^*).

In this final experiment session, a speed control without the encoder is implemented. To estimate machine speed, a speed estimator block is modeled by utilizing the information obtained from the flux estimator model. The overall block diagram and experimental result of speed control implementation are shown in Figures 9 and 10, respectively. As can be seen, the induction machine speed (ω_r) responds well for a step change in the reference speed (ω_r^*).

Conclusion and Future Work

This paper presents a design for the instructional advanced electric drives laboratory and its implementation . Off-the-shelf TI microcontroller DSK and industrial components were utilized for their cost-effectiveness and reliability as well as the hands-on experience they offer with induction machine drive systems, up-to-date tools, and embedded programming. For future work, widely used serial communication interface (SCI), controller area network (CAN) bus and serial peripheral interface (SPI) can be considered for improved user interface and peripheral expansion.

References

[1] S. A. Shirsavar, B. A. Potter, and I. M. L. Ridge. Three-phase machines and drives-equipment for a laboratory-based course. *IEEE Transactions on Education*, 49(3):383–388, Aug 2006. ISSN 0018-9359. doi: 10.1109/TE.2006.879266.

[2] N. Mohan, W. P. Robbins, P. Imbertson, T. M. Undeland, R. C. Panaitescu, A. K. Jain, P. Jose, and T. Begalke. Restructuring of first courses in power electronics and electric drives that integrates digital control. *IEEE Transactions on Power Electronics*, 18(1):429–437, Jan 2003. ISSN 0885-8993. doi: 10.1109/TPEL.2002.807120.

[3] R. S. Balog, Z. Sorchini, J. W. Kimball, P. L. Chapman, and P. T. Krein. Modern laboratory-based education for power electronics and electric machines. *IEEE Transactions on Power Systems*, 20(2):538–547, May 2005. ISSN 0885-8950. doi: 10.1109/TPWRS.2005.846237.

[4] Shuhui Li. Laboratory restructuring and development for the course of electric machinery using software and hardware it tools. *age*, 10:1, 2005.

[5] Razvan C Panaitescu, Ned Mohan, William Robbins, Philip Jose, Todd Begalke, C Henze, Tore Undeland, and E Persson. An instructional laboratory for the revival of electric machines and drives courses. In *Power Electronics Specialists Conference, 2002. pesc 02. 2002 IEEE 33rd Annual*, volume 2, pages 455–460. IEEE, 2002.

[6] Seung-Ki Sul. *Control of Electric Machine Drive Systems*. IEEE, 2011. ISBN 9780470876541. doi: 10.1002/9780470876541.app1.

Appendix:

C code sample generated by MATLAB embedded coder for V/F closed-loop implementation

```c
#include "V_F_Closed_loop_Experiment.h"
#include "V_F_Closed_loop_Experiment_private.h"

/* Block signals (default storage) */
B_V_F_Closed_loop_Experiment_T V_F_Closed_loop_Experiment_B;

/* Real-time model */
RT_MODEL_V_F_Closed_loop_Expe_T V_F_Closed_loop_Experiment_M_;
RT_MODEL_V_F_Closed_loop_Expe_T *const V_F_Closed_loop_Experiment_M =
  &V_F_Closed_loop_Experiment_M_;
static void rate_monotonic_scheduler(void);
static uint16_T adcInitFlag = 0;
/*
* Set which subrates need to run this base step (base rate always runs).
* This function must be called prior to calling the model step function
* in order to "remember" which rates need to run this base step.  The
* buffering of events allows for overlapping preemption.
*/
void V_F_Closed_loop_Experiment_SetEventsForThisBaseStep(boolean_T *eventFlags
    )
{
  /* Task runs when its counter is zero, computed via rtmStepTask macro */
  eventFlags[1] = ((boolean_T)rtmStepTask(V_F_Closed_loop_Experiment_M, 1));
}
/*
*This function updates active task flag for each subrate
* and rate transition flags for tasks that exchange data.
* The function assumes rate-monotonic multitasking scheduler.
* The function must be called at model base rate so that
* the generated code self-manages all its subrates and rate
* transition flags.
*/
static void rate_monotonic_scheduler(void)
{
/* Compute which subrates run during the next base time step.  Subrates
* are an integer multiple of the base rate counter.  Therefore, the subtask
* counter is reset when it reaches its limit (zero means run).
*/
(V_F_Closed_loop_Experiment_M->Timing.TaskCounters.TID[1])++;
if ((V_F_Closed_loop_Experiment_M->Timing.TaskCounters.TID[1]) > 9) {/* Sample
    time: [0.001s, 0.0s] */
   V_F_Closed_loop_Experiment_M->Timing.TaskCounters.TID[1] = 0;
  }
}

/* Model step function for TID0 */
void V_F_Closed_loop_Experiment_step0(void) /* Sample time: [0.0001s, 0.0s] */
{
  {                                     /* Sample time: [0.0001s, 0.0s] */
    rate_monotonic_scheduler();
```

```
  }

/* S-Function (c280xgpio_do): '<Root>/Digital Output' incorporates:
 *  Constant: '<Root>/Constant'
*/
{
if (V_F_Closed_loop_Experiment_P.Constant_Value)
    GpioDataRegs.GPASET.bit.GPIO7 = 1;
    else
    GpioDataRegs.GPACLEAR.bit.GPIO7 = 1;
}

/* S-Function (c280xqep): '<Root>/eQEP' */
  {
    V_F_Closed_loop_Experiment_B.eQEP = EQep1Regs.QPOSCNT;/*eQEP Position
    Counter*/
  }
}

/* Model step function for TID1 */
void V_F_Closed_loop_Experiment_step1(void) /* Sample time: [0.001s, 0.0s] */
{
/* S-Function (c280xadc): '<Root>/ADC' */
  {
    AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;  /* Reset SEQ1 module*/
    AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;/*clear INT sequencer*/
    AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;  /* Software Trigger*/
    while (AdcRegs.ADCST.bit.INT_SEQ1 == 0) {
    }  /*Wait for Sequencer INT bit to clear */
    asm(" RPT #11 || NOP");
  V_F_Closed_loop_Experiment_B.ADC[0] = (AdcRegs.ADCRESULT0) >> 4;
  V_F_Closed_loop_Experiment_B.ADC[1] = (AdcRegs.ADCRESULT1) >> 4;
  V_F_Closed_loop_Experiment_B.ADC[2] = (AdcRegs.ADCRESULT2) >> 4;
}
```