TI Designs
Simple PID Control Reference Design With PRU®-ICSS Through Web Interface

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDEP0073	Design Folder
AM3358 Sitara Processor	Product Folder
DRV8833 Motor Driver	Product Folder
BeagleBone Black	Tool Folder

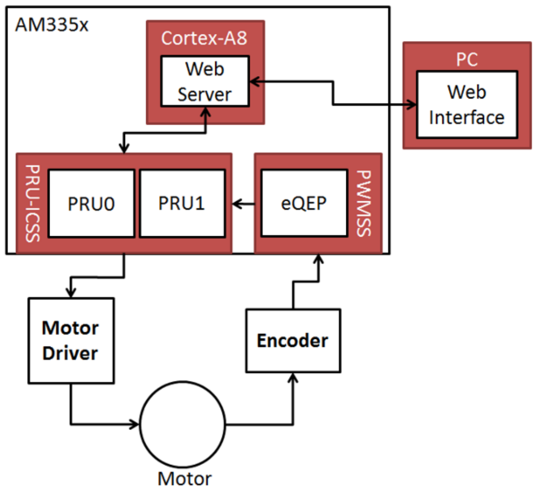
TI E2E™ Community — ASK Our E2E Experts

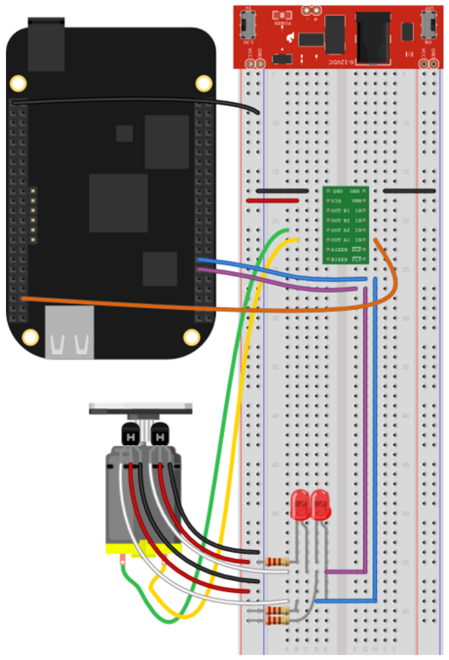
Design Features

- PWM Generation Using the PRU-ICSS Subsystem
- PRU®-ICSS Controlling the SOC eQEP Peripheral
- Simple PID Feedback Loop Implemented By the PRU-ICSS
- Inter-Processor Communication Between the ARM® Running Linux® and the PRU-ICSS
- ARM® Hosting a Web Interface for User Interaction

Featured Applications

- Simple Motor Control
- Appliances
- Robotics





1 Design Overview

This design demonstrates simple motor control with a PID feedback loop using the Programmable Real Time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS). A web interface is provided by the Cortex-A8 in the system for user control and interaction. This design provides a blueprint for task offloading and inter-processor communication that may be used when low latency and determinism are required.

2 System Description

The PRU-ICSS may be found on several TI processors, such as AM335x, AM437x, and AM57x; here, the Sitara™ AM335x is used on the BeagleBone Black.

The ARM® Cortex-A8 Processor in the AM335x device is used to host a web interface allowing users to view and set the speed of the motor, as well as the tuning parameters of the PID loop. The Remote Processor Messaging (RPMsg) library is used in the design to pass information back and forth between the PRU®-ICSS and Linux® on the ARM Cortex-A8 processor.

Inexpensive hobbyist parts and off-the-shelf components were used to keep costs low and make this design accessible for a wide audience.

3 Block Diagram

Figure 1 shows a high level block diagram. The PRU-ICSS controls the motor as well as the complete feedback loop through the encoder and eQEP. The ARM Cortex-A8 processor communicates with the PRU-ICSS and hosts a user interface.

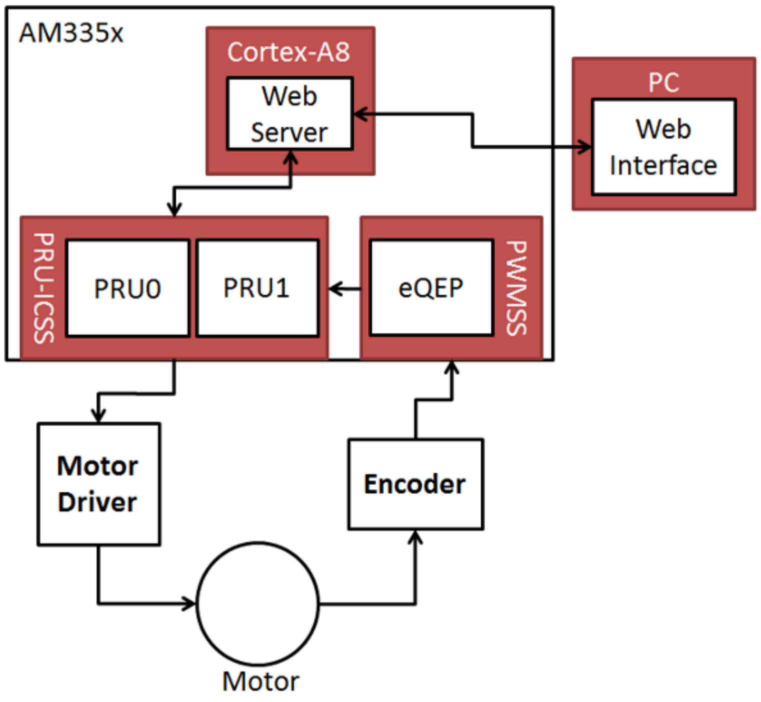


Figure 1. Block Diagram

4 System Design Theory of Operation

To achieve a set speed on a motor, the user must be able to read the current speed of the motor, compare that speed to the desired speed, and alter the signal driving the motor in response to both of those factors. Deterministic timing and the ability to respond quickly to real-world signals are ideal in a system such as this.

With Linux® running on the ARM Cortex-A8 it is very difficult, if not impossible, to achieve tight timings on a consistent basis. This is especially true when the processing load on the Linux system increases.

This design illustrates a system where all of the motor functions are offloaded from the ARM Cortex-A8 and placed into the PRU-ICSS. The PRU-ICSS includes two RISC cores each running at 200 MHz that are deterministic, have the ability to directly control I/O pins, read and write to the entire memory map, as well as send and receive messages to and from the ARM Cortex-A8 running Linux (see Figure 2).

In this design, a brushed DC motor has its speed controlled using a PWM signal that is generated by the PRU-ICSS. A wheel encoder kit is attached to the motor which provides RPM feedback to the Enhanced Quadrature Encoder Pulse (eQEP) module in the AM335x device. The PRU-ICSS uses its ability to read and write the full memory map of the device to control the eQEP module and get the DC motor's current speed. The motor's current speed along with a desired set speed are passed to a PID loop to calculate the required PWM at that moment to drive the motor to the desired speed. This loop is run continuously without using a single cycle from the ARM Cortex-A8 host processor that is running Linux.

Because the motor control and feedback are both offloaded from the ARM core, the ARM core has free cycles to host a web interface that allows users to view a graph of the motor's current speed graphed on top of the desired set point. The interface gives the user the ability to change the desired set point as well as a method to alter the PID constants used for loop tuning.

The Remote Processor Messaging (RPMsg) library from the PRU Software Support Package is used in this design to pass the motor speed from the PRU-ICSS to the ARM core, and also to pass the desired set speed and the PID loop tuning parameters from the ARM core to the PRU-ICSS. For more information on the RPMsg library see the PRU-ICSS Remoteproc and RPMsg Wiki page. Figure 2 shows the PRU subsystem block diagram.

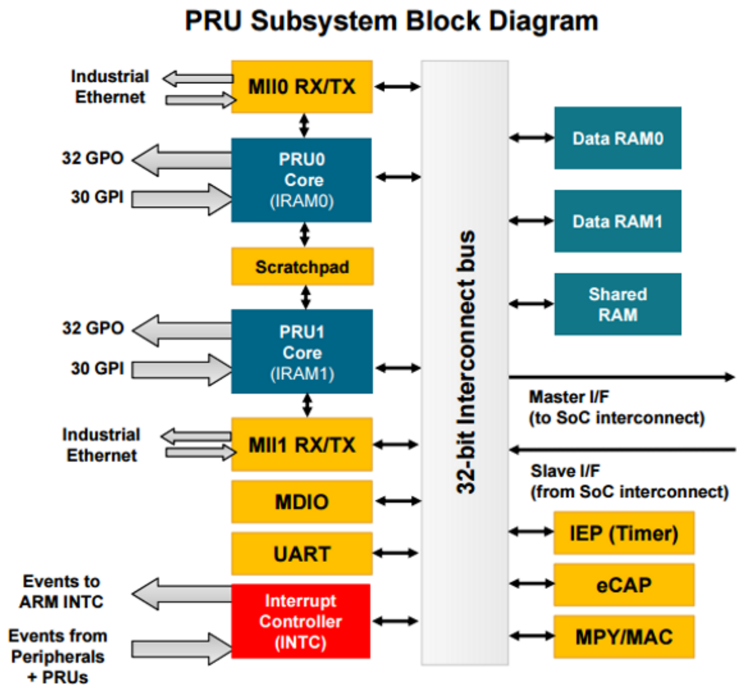


Figure 2. PRU Subsystem Block Diagram

5 Getting Started Hardware

5.1 Hardware Requirements

The following sections list the hardware that is required to get started.

5.1.1 BeagleBone Black

See http://www.beagleboard.org/black for more information and to purchase a BeagleBone Black from a vendor. See the documentation posted on the TI Design Website for the BeagleBone Black System Reference Manual and *Technical Reference Manual* (SPRUH73M) for the AM335x. A 5-V power supply or a USB cable is required to power the BeagleBone Black as well as a USB-to-Serial cable for console output. An Ethernet® cable is also required to connect the BeagleBone Black to a network to access the web page that is hosted.

5.1.2 DRV8833 Motor Driver

The DRV8833 device provides a dual H-bridge motor driver solution for toys, printers, and other mechatronic applications. This TI Design shows an example of a PWM signal being sent from the BeagleBone Black to the DRV8833 in order to spin a brushed DC motor.

See https://www.pololu.com/product/2130 to purchase the DRV8833 carrier board that was used in this design from Pololu™.

5.1.3 Wheel Encoder Kit

A wheel encoder kit is used to provide RPM feedback from the motor to the host processor. The kit includes two Hall Effect sensors and two 8-pole magnets that mount on the DC motor shaft. See https://www.sparkfun.com/products/12629 to order the encoder kit used in this design.

5.1.4 Brushed DC Motor

See https://www.sparkfun.com/products/11696 for an inexpensive hobby motor from SparkFun Electronics™ that is used in this design.

5.1.5 Power Supply

An external power supply is required to provide power for the motor. The power supply must be capable of supplying 3.3 V and at least 800 mA. See https://www.sparkfun.com/products/13032 to view the power supply used in this design.

5.1.6 PRU Cape (Optional)

The PRU cape has LEDs that may be useful for debug purposes. During the development process the LEDs connected to PRU-ICSS core 1 were used to indicate whether or not the quadrature encoder peripheral had experienced an overflow condition. See http://www.ti.com/tool/PRUCAPE to order the PRU cape.

5.1.7 Miscellaneous

A few resistors, LEDs, and a breadboard are also required to connect together all of the pieces of the design. The included schematic shows the connections to be made on the breadboard and the bill of materials gives the values of the resistors required.

5.2 Hardware Setup

See the schematic in Figure 7 while completing this section.

1. Connect the DRV8833 to the BeagleBone Black and the motor.
   (a) Connect the GND connections between the external power supply and the BeagleBone Black P9 → 1 and P9 → 2 to ensure the generated PWM may be read from the DRV8833. Do not connect the power connections between the boards.
   (b) Connect the DRV8833 VIN to the external power supply input (**not** the 3.3 V from the BeagleBone Black).
   (c) Connect P9 → 42 (the pin generates the PWM) onto the BeagleBone Black to *A1_IN* on the DRV8833 carrier board. (Optionally, connect *A2_IN* to the external 3.3 V power to enable brake mode on the DRV8833.
   (d) Connect the two inputs on the motor to *A1_OUT* and *A2_OUT* from the DRV8833 carrier board.
2. Mount the magnetic encoder on the motor as shown in Figure 3.
   (a) If using the hobby gear motor recommended for this design, pliers may be required to move the white sprocket down before the disc may be pressed onto the shaft.
   (b) Slide the rubber and metal disc assembly onto the shaft, with the rubber side facing down.
   (c) Attach the two Hall Effect sensor bodies to the motor using a zip-tie as shown in Figure 3. A 90º bend on the sensor is required, with the rounded side of the sensor facing the encoder disc.

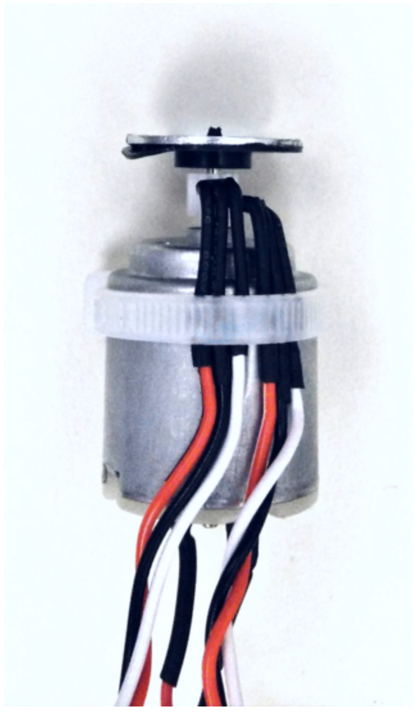


Figure 3. Encoder Setup

   (d) Some overlap on the sensors may be required to achieve the correct offset for the quadrature encoder input of approximately 50% phase offset.
   (e) Tuning may be done once the sensors are wired and the LEDs are added.
   (f) The phase offset may be checked on an oscilloscope, and the ouput must be similar to Figure 4.

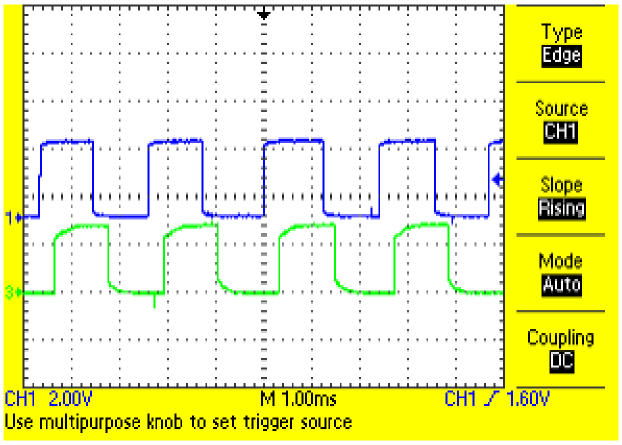


Figure 4. Encoder Phase Offset

3. Connect the quadrature encoder wires to the BeagleBone Black.
   (a) Connect the red wire to 3.3 V from the external power supply.
   (b) Connect the black wire to GND.
   (c) Attach the pull-up resistors between 3.3 V from the external power supply and white signal out from the Hall Effect sensor.
   (d) TI recommends adding LEDs to each signal wire with each LED anode attached to signal, and current limiting resistors between the LED cathode and GND.
   (e) Attach the signals from signal from the white wire on the Hall Effect sensors to pins P8 → 33, and P8 → 35 on the BeagleBone Black.

6 Getting Started Software

6.1 Software Requirements

To use the demo software provided with this TI Design, the user must have the following software downloaded (latest versions at the writing of this document).

- AM335x Linux Processor SDK v02.00.02.11
- PRU-ICSS PID Motor Demo Software Package v01.00

6.2 AM335x Linux Processor SDK Installation

This design was written using a 64-bit version of Ubuntu 14.04 LTS host machine.

- Linux Processor SDK v02.00.02.11 download page: Processor SDK Download Page
- Installation instructions: Processor SDK Linux Installer
- Additional libraries if you are running a 64-bit distro: 64-bit Ubuntu Support
- Processor SDK Linux Getting Started Guide

6.3 PRU-ICSS PID Motor Drive Demo Software Package Installation

An installer for the demo software is included on the TI Design page , and must be installed on the same Ubuntu host computer that the Processor SDK was installed on. The name of the installer is ***pru-icss-pid-motor-demo-v1.0.bin*** and may be installed by navigating to the directory where the installer was downloaded. Type:

#./pru-icss-pid-motor-demo-v1.0.bin

You may install the package anywhere on the Ubuntu development machine, however, TI recommends that you place the package in the ***example applications*** folder at the root of the Linux Processor SDK v02.00.02.11 installation.

6.4 Configure Pin Muxing

A device tree include file is provided in the PRU-ICSS pid motor demo software package in the ***sw/dtsi/*** folder. Copy the include file (**am335x-boneblack-prupid.dtsi**) into the ***dts*** folder of the Linux Kernel source of the Processor SDK. The dts folder may be found at ${Processor SDK}/board-support/linux-X.XX.XX*/arch/arm/boot/dts/

Once the **am335x-boneblack-prupid.dtsi** file is in the **dts** folder, you must add an include statement to the ***am335x-boneblack.dts*** file (that is already located in the dts folder), so that the new pin muxing is included and applied to the BeagleBone Black device tree binary (am335x-boneblack.dtb). Add the line **#include "am335x-boneblack-prupid.dtsi"** at the END of the ***am335x-boneblack.dts*** file.

NOTE: The user must rebuild the kernel, device tree, and modules in the following section. The user must make sure that the changes made in this section are complete before moving on. If your pin muxing is not configured correctly, the demo will not work.

6.5 Build the Linux Kernel and Device Tree

From the ***${Processor-SDK}/board-support/linux-X.XX.XX*/*** directory, run the following bolded commands. Make sure to add the cross-compile toolchain to your PATH variable first. If you do not understand any of the following commands, see the Processor SDK Linux Kernel Users Guide.

```
# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean

# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- tisdk_am335x-evm_defconfig

# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage

# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x-boneblack.dtb
```

NOTE: In this version of the Linux Processor SDK (v02.00.02.11) the modules required to load firmwares into the PRU-ICSS and enable RPMsg communication are included out of the box in the provided kernel and file system. You are only required to update the symbolic links in the ***/lib/firmware/*** directory of your file system to point to the new PRU-ICSS firmwares that you want to load and run (this is mentioned again in Section 6.6). Previous versions of the SDK required modification of the ***.config*** file to enable PRU-ICSS Remoteproc and RPMsg support.

6.6 Compiling the ARM User Space Binary and PRU-ICSS Firmwares

You must now compile the ARM binary as well as the two PRU firmwares required to run the demo.

1. Navigate to the top-level directory of the PRU-ICSS PID Motor Demo Software Package on the host machine (referred to as $ { PRU-PID-MOTOR-DESIGN }).
2. A Makefile is provided for building both PRU-ICSS firmware images and the ARM user space binary.
3. To successfully compile these binaries, the following environmental variables must be manually exported (modify to reflect the correct path on your system).

   ARM Cross-Compile Toolchain

   - export ARM_CCT=${HOME}/ti-processor-sdk-linux-am335x-evm-02.00.02.11/linux-devkit/sysroots/x86_64-arago-linux/usr/bin

   PRU Code Generation Tools

   - export PRU_CGT=${HOME}/ti-processor-sdk-linux-am335x-evm-02.00.02.11/linux-devkit/sysroots/x86_64-arago-linux/usr/share/ti/cgt-pru

   PRU Software Support Package

   - export PRU_SSP=${HOME}/ti-processor-sdk-linux-am335x-evm-02.00.02.11/example-applications/pru-icss-4.0.2
4. To build the binaries, navigate to the ***sw*** folder and execute the ***make*** command:

   ```
   # cd ${PRU-PID-MOTOR-DESIGN}/sw/
   # make
   ```

NOTE: This top-level file makefile calls makefiles in each project folder, and may be called directly to rebuild each binary separately if desired.

If nothing was missing, the binaries are located in the newly created ***gen*** folder inside each of the three project folders:

ARM user space application binary:

- $ {PRU-PID-MOTOR-DESIGN}/sw/prumsg/gen/prumsg

PRU-ICSS core 0 firmware binary:

- ${PRU-PID-MOTOR-DESIGN }/sw/prupid_fw_0/gen/prupid_fw_0.out

PRU-ICSS core 1 firmware binary:

- ${PRU-PID-MOTOR-DESIGN }/sw/prupid_fw_1/gen/prupid_fw_1.out

6.7 Copying Files to the File System of the Board

Now that all modules, binaries, and firmware have been built, you must move them over to the file system.

NOTE: During development, there are multiple ways to provide a kernel, device tree, file system, and kernel modules. Use whatever method you would like (SD card, TFTP boot, NFS server). It is expected that the user knows how to take the kernel image and device tree built in Section 6.5, and boot the board. If you are struggling with this, TI recommends to go through the Processor SDK Linux Getting Started Guide before returning to this design. Use the rebuilt kernel (zImage) and modified device tree (am335x-boneblack.dtb) built in Section 6.5 in addition to the following steps.

1. Copy the ARM binary that was built to the */usr/bin* directory of the board's file system.
   - ${PRU-PID-MOTOR-DESIGN}/sw/prumsg/gen/prumsg
2. Copy the ***pru_pid_server*** directory from the demo software installation directory to the ***/usr/share/matrix-gui-2.0/*** directory of your board’s file system.
   - ${PRU-PID-MOTOR-DESIGN}/sw/pru_pid_server/
3. Copy the two PRU-ICSS firmwares into the ***/lib/firmware/pru/*** directory of your board’s file system (***/lib/firmware/pru/***).
   - ${ PRU-PID-MOTOR-DESIGN }/sw/prupid_fw_0/gen/prupid_fw_0.out
   - ${ PRU-PID-MOTOR-DESIGN }/sw/prupid_fw_1/gen/prupid_fw_1.out
4. Update the existing symbolic links in the ***/lib/firmware/*** directory of your board’s file system to point to the new firmwares.
   - am335x-pru0-fw → /lib/firmware/pru/prupid_fw_0.out
   - am335x-pru1-fw → /lib/firmware/pru/prupid_fw_1.out

You now have the ARM application, the web server page, and two PRU-ICSS firmwares copied over from your development machine to the file system of the board.

6.8 Booting and Running the Demo

Everything is now in place and the board is ready to be booted. Make sure the 3.3-V FTDI UART cable is connected to the BeagleBone Black, and a terminal emulator program (Minicom, picocom, screen, etc) is ready to communicate.

1. Power up the BeagleBone Black
2. During the boot process the PRU-ICSS firmware has been loaded and running, and is polling for commands from the ARM host.
3. Make sure the Ethernet cable is plugged into the BeagleBone Black.
4. Make sure you have an IP address.
5. Note down your IP address from the following command:
   - **#ifconfig**
6. Move to the ***pru_pid server*** folder that you copied and start the server.
   - **# cd /usr/share/matrix-gui-2.0/pru_pid_server/**
   - **# ./start-server.sh**

   You will now see the RPM, PWM, and set point numbers scrolling by on the terminal. This is the data that is being passed back and forth between the ARM and the PRU-ICSS cores and also the data that is being served up on the associated web page.
7. Open a web browser (Google Chrome™ was used for testing)
8. Navigate to the IP address of your board, followed by the folder name you copied above and hit enter
   - **192.168.0.12/pru_pid_server/**

NOTE: This assumes that your IP address found in step 5 was 192.168.0.12.

You should now see a web interface that shows a graph of the current set point, the current RPMs of the motor, and the current PWM output. There is also a slider to modify the set point of the motor as well as input boxes to modifying the PID constants for tuning. Figure 6 shows the web interface.

NOTE: If the console output or the web GUI reports 0 RPMs from the motor, you must check that you are using the device tree binary that was created using the ***dtsi*** file provided in the demo software package. If this device tree binary is not used, the pin muxing is not configured correctly and the device is not able to read the current speed of the motor.

6.9 Software Disclaimer

The software provided has been proven to build and run correctly on the 2.0.2.11 version of the Linux Processor SDK. Using a different version of the SDK or different hardware components has not been tested and must be verified by the user.

7 Test Setup

Figure 5 shows the test setup used to verify this TI Design. The optional PRU cape was used for the debug LEDs and all the external connections were made on a breadboard.

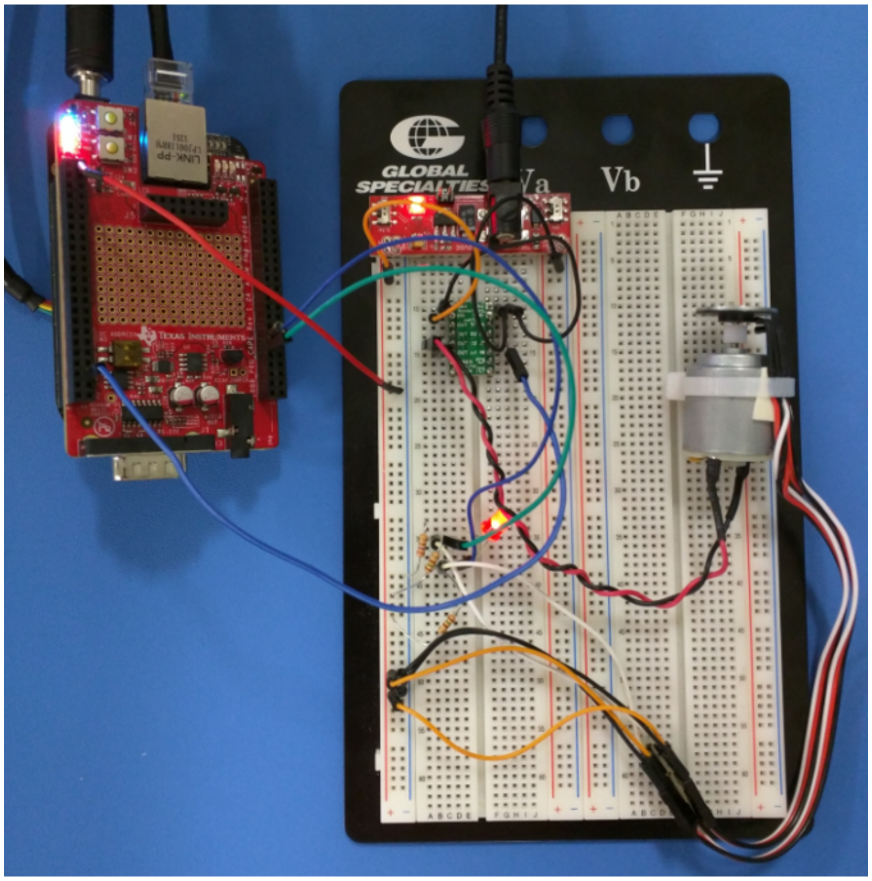


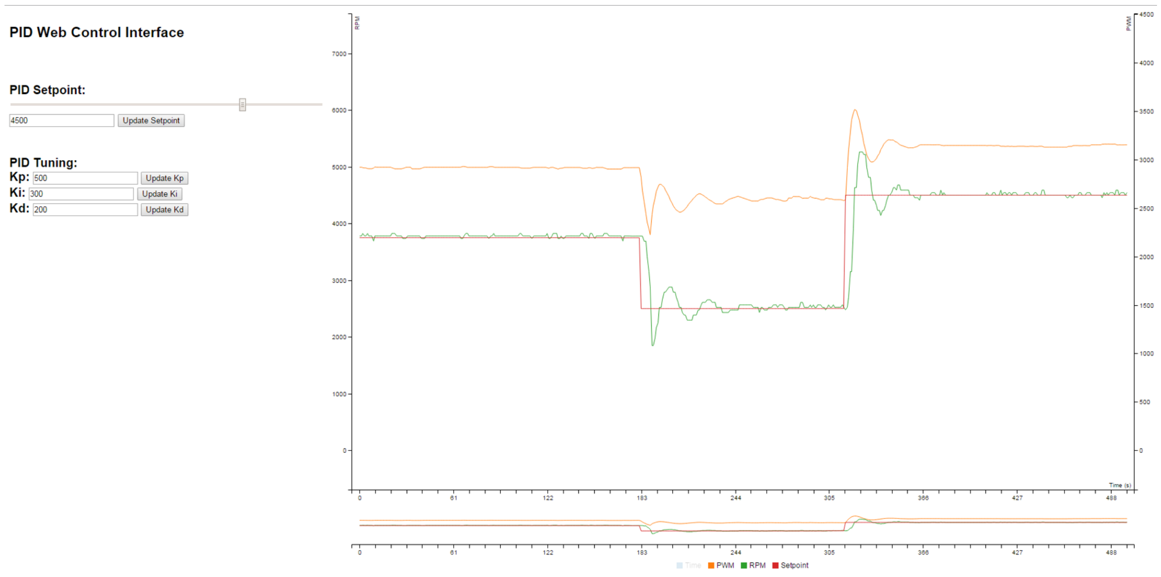
Figure 5. Breadboard Setup

8 Test Data

With the provided schematic and software, we were successfully able to provide an RPM set point, and then have the PRU-ICSS use the encoder feedback to produce a PWM signal that spins the motor very close to that set point. All of this information may be viewed and set from the web interface hosted by the board shown in Figure 6.

8.1 Web Interface Example

Figure 6 shows an example of the output that is displayed on the web server interface. Observe that the actual speed of the motor tracks the desired set point very closely.



PID Web Control Interface

PID Setpoint:

4500 Update Setpoint

PID Tuning:

Kp: 500 Update Kp

Ki: 300 Update Ki

Kd: 200 Update Kd

Figure 6. Web Interface

9 Design Files

9.1 Schematics

To download the schematics for each board, see the design files at http://www.ti.com/tool/TIDEP0073. Figure 7 shows the PRU-ICSS PID motor demo schematic.

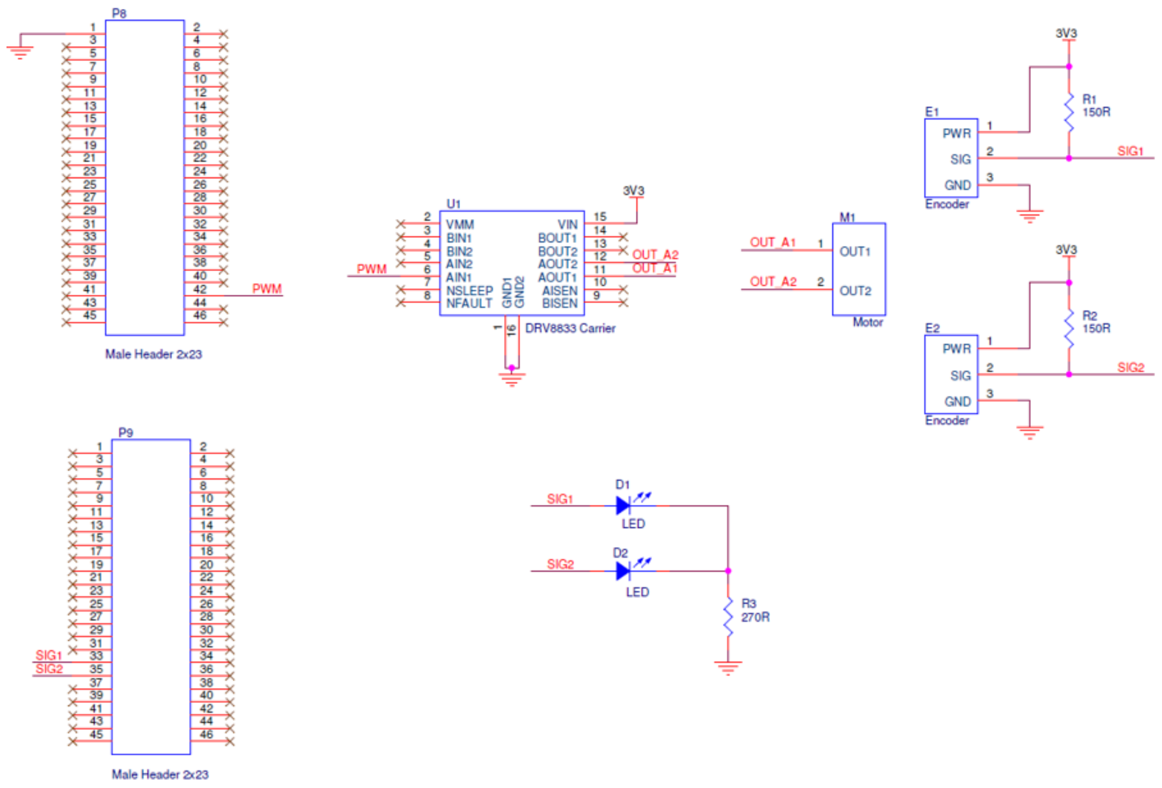


Figure 7. PRU-ICSS PID Motor Demo Schematic

9.2 Bill of Materials

To download the Bill of Materials (BOM) for this design, see the design files at http://www.ti.com/tool/TIDEP0073

Table 1 lists the Bill of Materials (BOM) for this design.

Table 1. Bill of Materials

VALUE	DESCRIPTION	DESIGNATOR	FOOTPRINT	MANUFACTURER	MANUFACTURER PART NUMBER	QUANTITY
LED	LED RED DIFF 3MM ROUND T/H	D1, D2	2DIN	Rohm Semiconductor™	SLR-343VRT32	2
Male Header 2 × 23	CONN HEADER MALE 46PS.1" DL GOLD	P8, P9	HDR2×23	–	–	2
150 R	RES 150 Ω 1/4 W 1% AXIAL	R1, R2	2DIN	Yageo®	MFR-25FBF52-150R	2
270 R	RES 270 Ω 1/4 W 1% AXIAL	R3	2DIN	Yageo	MFN-25FRF52-270R	1
DRV8833 Carrier	DRV8833 Dual Motor Driver Carrier Board	U1	16DIN	Pololu	DRV8833 Carrier Board	1
Motor	Brushed DC Motor	M1	N/A	SparkFun	ROB-11696	1
Encoder	Wheel Encoder	E1, E2	N/A	SparkFun	ROB-12629	2

10 Software Files

To download the software files for this design, see the files at http://www.ti.com/tool/TIDEP0073.

11 Terminology

PRU-ICSS: Programmable Real-Time Unit – Industrial Communications Sub Systems

eQEP: Enhanced Quadrature Encoder Pulse

BOM: Bill of Materials

E2E: Engineer to Engineer: http://e2e.ti.com/

12 About the Authors

JASON MERLO is a Catalog Embedded Processors Intern at TI, where he is developing reference design solutions for the PRU-ICSS.

JASON REEDER is a Catalog Embedded Processors Software Applications Engineer at TI.