PteroDAQ Documentation

extracted from **Applied Analog Electronics**

Kevin Karplus
Edition 1.1—September 24, 2020

Published by the author through Leanpub

Because this document is an appendix extracted from the textbook *Applied Analog Electronics: a first course in electronics* [2] , there may be a few pointers to figures or sections that are not part of this appendix. These missing cross-references are indicated with a "??" in place of the figure or section number. If you really need the cross-references, you will have to buy that book, but I believe that this is useful even without the cross-references.

Published through Leanpub.
Available at https://leanpub.com/applied_analog_electronics

Leanpub

1: PteroDAQ documentation

PteroDAQ is a data-acquisition system that works with a microcontroller board connected via a USB cable to a host computer. The microcontroller board can be any of a number of different boards. The choice of board provides some differences in functionality besides the channel numbers listed in Table 1.1 (e.g., voltage ranges on the inputs, resolution of analog-to-digital converters, and sampling frequency). The pinout for each board is available from the manufacturer of the board—from PJRC for the Teensy boards [3], from NXP for the FRDM-KL25Z board [1], and from the various manufacturers for Arduino and Arduino-clone boards.

The microcontrollers used in this course contain an *analog-to-digital converter (ADC)*, that converts voltages into numbers. Depending which microcontroller board is used, the converters may produce 10-bit numbers (0 through 1023) or 16-bit numbers (0 through 65535). The full-scale voltage range is either 0 to 3.3 V or 0 to 5 V, again depending on the microcontroller board.

The FRDM-KL25Z and Teensy boards have 16-bit ADC converters (that is, the range of the numbers is 0 to $2^{16} - 1 = 65535$), while the Arduino boards have only a 10-bit resolution (that is, values from 0 to 1023). The FRDM-KL25Z can measure voltages from any of several different pins (PTB0, PTB1, PTB2, PTB3, PTC0, PTC1, PTC2, PTD1, PTD5, PTD6, PTE20, PTE21, PTE22, PTE23, PTE29, PTE30) as *single-ended* inputs—that is the value reported is the difference between the voltage on the pin and the ground pin on the microcontroller. The Teensy boards can measure voltage on A0 through A12 (Teensy LC) or A0 through A14 (Teensy 3.1 or 3.2). The Arduino boards can measure voltages on any of the 6 analog input pins A0 through A6 (the Arduino Mega has A0 through A15).

The FRDM-KL25Z also has two *differential* channels: E20 − E21 and E22 − E23. The outputs of these channels are from $-2^{15} = -32768$ to $2^{15} - 1 = 32767$. The Teensy boards also have differential channels: A10 − A11 on all and A12 − A13 on Teensy 3.1 and 3.2. The Arduino boards do not have differential channels.

The voltage on the analog pins must be limited to be between the power rails of the microcontroller. That is, on the FRDM-KL25Z and Teensy boards $0 \le V_A \le 3.3\,V$ and on most Arduino boards $0 \le V_A \le 5\,V$ (some Arduino boards are 3.3 V). Putting negative voltages or larger positive voltages on the board may damage the chips on the board. A differential channel may have a negative number (for example, if PTE20 < PTE21), but the voltages on each pin must stay between 0 and +3.3 V with respect to the ground pin of the microcontroller.

With the currently supported microcontroller boards, it is not possible for PteroDAQ to measure voltages simultaneously on different inputs, but the microcontroller can switch the ADC rapidly from one input to another, so that measurements can be made at *almost* the same time. Some microcontrollers (including the one on the Teensy 3.1/3.2) have multiple analog-to-digital converters, and PteroDAQ is likely to be extended to handle multiple ADCs, to get nearly simultaneous measurement and a faster sampling rate. (See Section **??** for an explanation of sampling rate and why it is important.)

Board names	channels analog	differential	digital	frequency	trigger
Arduino Uno Arduino Duemilanove Arduino Diecimila Arduino Ethernet Arduino Pro Arduino LilyPad	8	0	20	0	2
Arduino Mini Arduino Nano Arduino Pro Mini Arduino Fio	10	0	20	0	2
Arduino Mega	17	0	14	0	6
Arduino Leonardo Arduino Yun Arduino Micro Arduino Robot Arduino Esplora Arduino LilyPad USB	14	0	23	0	5
FRDM-KL25Z	18	2	54	2	18
Teensy 3.1 Teensy 3.2	24	2	18	5	18
Teensy LC	15	1	14	3	12

Table 1.1: This table lists the boards supported by the PteroDAQ data-acquisition system, with some indications of the capabilities of each board. The table is derived from the PteroDAQ `boards.py` *file, using the names that PteroDAQ uses, and so the names used may not be the official names of the boards. The "trigger" channels are the digital inputs that can be used to start a sample, rather than using the timer. Warning: the FRDM-KL25Z code has not been maintained recently, and may no longer be compatible with the newer versions mbed compiler and library.*

The ADC is *ratiometric*, which means that the output value for the 16-bit analog-to-digital converters is $65536 V_\mathrm{A}/V_\mathrm{REFH}$, where V_A is the voltage on the analog input pin, and V_REFH is a reference voltage being compared to. (For differential inputs on the FRDM-KL25Z and Teensy boards, the scaling is $32768 V_\mathrm{A}/V_\mathrm{REFH}$, as the 65536 values map to twice as wide a range: $-V_\mathrm{REFH}$ to $+V_\mathrm{REFH}$.) The default reference voltage in the PteroDAQ software we are using is the power-supply voltage to the board, which the system measures to convert the raw readings into volts.

If the power-supply voltage is likely to fluctuate, then it is worth using one channel to record the 1 V bandgap reference built into the Teensy LC board—you can then correct for the fluctuation (if it is not too fast), by dividing the analog channel reading by the 1 V reading.

The ARM-based boards (FRDM-KL25Z and Teensy) boards can be used to measure the

frequency of digital signals, by counting the number of rising edges in each sample time, but that capability has not been implemented for the Arduino boards. The FRDM-KL25Z and Teensy LC boards can measure up to 3.9 MHz, but use a 20-bit counter to count pulses, so the frequency being measured must be no more than a million times the sampling rate. The Teensy 3.1/3.2 can measure up to 4 MHz, but uses only a 15-bit counter, so the measured frequency can only be up to 30,000 times the sampling rate.

For each channel, PteroDAQ provides a *sparkline* that plots the last few samples recorded (updated ten times a second). PteroDAQ also displays the last sample measured, the average of all the samples (labeled "DC"), and the standard deviation of all the samples (labeled "RMS").

The *RMS* stands for "Root Mean Square", which is a common way for measuring the magnitude of time-varying signals. There are two common ways of defining RMS voltage used in voltmeters: AC and AC+DC.

- The *AC+DC* definition squares the observed voltage, averages the squared signal over some time period, then takes the square root. As shown in Section **??**, this definition is useful for computing how much heat is dissipated in a component.
- The *AC* definition first computes the average voltage, then squares the difference from that average, and again takes the square root of the average of the square—this is the same concept as standard deviation in statistics, and serves the same purpose of measuring how much a signal is changing.

Because many of our sensors and circuits will have a small changing signal with a large DC offset, the AC definition is usually the more useful one for us, and that is what PteroDAQ reports. You can compute the AC+DC RMS voltage, if needed, as $V_{\mathrm{AC+DC}} = \sqrt{V_{\mathrm{DC}}^2 + V_{\mathrm{RMS}}^2}$.

The DC and RMS values can be used to measure steady-state signals (either DC or AC) with higher precision than single samples. In some of the places where an AC voltmeter is called for in the labs, PteroDAQ can be substituted—the crucial questions are whether the voltages are in a range suitable for the analog-to-digital conversion and whether the aliasing due to PteroDAQ's sampling causes high frequency signals to be misinterpreted.

There are some limitations of the FRDM-KL25Z and Teensy boards that are important for this course:

- The microcontroller can only record voltages, frequencies, digital values, and time stamps, so we need to convert other signals (like current or capacitance) into signals that we can measure.
- The highest voltage allowed on any pin is 3.3 V and the lowest is 0 V.
- There are only 15–24 channels that can be used for single-ended voltage input (and only 1–2 that can be used for differential input).
- The resolution is only 16 bits (65536 different values). With V_{REFH} at the default value of 3.3 V, the step size is about 50 μV for single-ended measurements and 101 μV for differential measurements. The accuracy is much worse than the resolution, but

is difficult to estimate ahead of time—repeatability is probably only ±1 mV, and the absolute accuracy probably only ±3%. On the Arduino boards, the resolution is only 10 bits (1024 different values) and the step size about 5 mV.

- The ADC gets inaccurate when near the power rails or reporting small numbers—you usually want to have all inputs stay at least 2.5 mV from either power rail (50 counts on 16-bit ADC), and measure differential signals that are at least 1 mV (20 counts).

- Readings are not simultaneous, but the analog-to-digital converter is quickly switched from one input to another. It takes between 2.2 μs and 128 μs for each conversion on the FRDM-KL25Z and Teensy boards, depending on how much hardware averaging is done.

  If you pick too high a sampling rate for the number of channels and degree of hardware averaging selected, so that the microcontroller can't do the conversions in time, PteroDAQ reports "Warning: triggering too fast, next trigger before finished". On the FRDM-KL25Z and Teensy LC boards at 32× hardware averaging, this happens at about 6.9 kHz, and on the Teensy 3.1/3.2 at about 10.3 kHz.

  At 4× averaging, much higher sampling rates are possible, and limitations on the host computer occur before the sampling-rate limit is reached (the Teensy 3.1/3.2 can do 35 kHz sampling, if the host computer can read the data fast enough).

- The FRDM-KL25Z has limited memory size for queuing recordings and will drop data points if its queue is full. Each sample requires 8 bytes for a time stamp and 2 bytes per analog channel, so the 8 kB queue (half the RAM) on the board can only hold about 820 samples of a single channel or 683 samples of 2 channels. That means it can hold about 2.7 seconds worth of data at a 300 Hz sampling rate, or over a minute of data at 10 Hz.

  The Teensy 3.1 and 3.2 have a 32 kB queue, and the Teensy LC has only a 4 kB queue.

  This buffer can be filled as fast as the processor can collect data, and the PteroDAQ software transfers the data as quickly as it can to the host computer, which results in different speeds with different microcontroller boards and different host computers.

- When not doing large amounts of hardware averaging, the main limiting factor on the sampling rate for the PteroDAQ system is how fast the host computer can read the samples over the USB connection, to keep the queue from filling up. This is dependent on the microcontroller board, the operating system on the host, and the Python program on the host.

  The FRDM-KL25Z at 32× averaging can generally record a single channel at 6.3 kHz indefinitely, even with slow host computers—the limitation is mainly from the time taken to do the conversions.

  At 4× averaging, with my old, slow MacBook Pro laptop, I can record for about 20 s at 17 kHz or 35 s at 15 kHz, before the FRDM-KL25Z queue fills up and the PteroDAQ system warns that samples are being dropped. On faster machines, the sampling rate can go to 20 kHz, but there is always a chance that the host operating system will temporarily take enough processing power away from the Python job that a few samples will be lost, even at only 10kHz.

board	1 analog channel	2 analog channels	3 analog channels
Arduino (ATMega386)	267 μs	415 μs	563 μs
FRDM KL25Z (1×)	47 μs	50 μs	56 μs
Teensy LC (1×)	44 μs	48 μs	56 μs
Teensy 3.1 (1×)	44 μs	48 μs	51 μs
FRDM KL25Z (32×)	146 μs	276 μs	405 μs
Teensy LC (32×)	148 μs	276 μs	403 μs
Teensy 3.1 (32×)	134 μs	258 μs	381 μs

Table 1.2: Measured fastest sample periods for various microcontroller boards running PteroDAQ. The three NXP processors all have similar performance, because they have the same underlying ADC hardware. The Teensy 3.1 was overclocked to 96 MHz, to get similar performance on the ADC as the Teensy LC running at 48 MHz. The extra processor power of the Teensy 3.1 makes a small difference with 32× averaging, but at 1× averaging, the speed is limited by how fast the Python program on the host can run to keep the data queue from overflowing, so all three boards take about 50 µs a sample, with only slight increases as the number of channels increases.

The larger queue on the Teensy 3.1/3.2 does not seem to help much—I still see samples dropped after the first 500,000 or so at 15–17 kHz. On a faster (2.7GHz Core i5) iMac, I can get up to 5,000,000 samples at 20 kHz or 2,000,000 at 25 kHz, but eventually the operating system makes the Python program take too long a pause and the queue on the Teensy fills up.

Table 1.2 shows how small a sample period I could use on several boards to run for several minutes with my iMAC as a host computer. I used 1× and 32× sampling on the boards that supported hardware averaging, The Teensy boards were compiled for fast code rather than small code, at the default speed settings (48 MHz for the Teensy LC and 96 MHz for the Teensy 3.1).

For some applications, missing a few samples and having an irregular sampling rate is not a major problem, and the system adapts to the communication rate, producing samples as fast as they are consumed by the host computer.

- The FRDM-KL25Z and Teensy analog-to-digital converters were designed assuming that the input comes from a low-impedance source—one that can provide a reasonably large current to charge or discharge capacitors.

  With a high-impedance source (over about 1 kΩ), hardware averaging does not improve the signal-to-noise ratio, as noise injected onto the analog pin by the sampling hardware itself is not removed by the source before the next sample is taken. For high-impedance sources, using no hardware averaging and a slow sampling rate may result in more accurate measurement than doing the hardware averaging.

  Section **??** discusses a way to avoid having to connect high-impedance inputs directly to the analog-to-digital converter, to avoid this problem.

- Many of the labs in this book call for using the microcontroller board as a power supply for the circuit being tested, to ensure that voltages being measured stay within the allowed input voltage range.

  The FRDM-KL25Z 3.3 V power supply uses a low-dropout voltage regulator (LDO) that can handle up to 1A, and so using the board as a power supply is limited primarily by the current limit from the USB power source (500 mA).

  The Arduino boards use the same series of LDOs, so can supply up to 1 A when powered by a separate power supply, but only 500 mA when powered over the USB line. The 5 V supply from the USB line is often quite noisy, making a rather poor voltage reference.

  The Teensy LC and 3.1 boards use a much more limited voltage regulator built into the microcontroller chip itself. They are limited to providing about 100 mA as a 3.3 V power supply. The Teensy 3.2 is almost identical to the Teensy 3.1, but can supply 250 mA from its 3.3 V supply.

References

[1] Freescale Semiconductor (NXP). *FRDM-KL25Z Quick Reference Card*. URL: https://www.nxp.com/docs/en/supporting-information/FRDM-KL25Z-Quick-Reference-Card.pdf (visited on 15 Nov. 2018) (cit. on p. 1).

[2] Kevin Karplus. *Applied Analog Electronics: a first course in electronics*. 29 July 2019. URL: https://leanpub.com/applied_analog_electronics (cit. on p. ii).

[3] PJRC. *Teensy Pin Assignments*. URL: https://www.pjrc.com/teensy/pinout.html (visited on 15 Nov. 2018) (cit. on p. 1).