ECE 445L Lab 2

Debugging, oscilloscope fundamentals, logic analyzer, dump profiles

This laboratory assignment accompanies the book, [*Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154*](https://www.amazon.com/Embedded-Systems-Real-Time-Interfacing-Microcontrollers/dp/1463590156), by Jonathan W. Valvano, copyright © 2024.

# Table of Contents

[Table of Contents 1](#_Toc173484554)

[Team Size 2](#_Toc173484555)

[Goals 2](#_Toc173484556)

[Review 2](#_Toc173484557)

[Starter Files 2](#_Toc173484558)

[Required Hardware 2](#_Toc173484559)

[Lab Overview 2](#_Toc173484560)

[Preparation 3](#_Toc173484561)

[Procedure 4](#_Toc173484562)

[Lab Checkout 7](#_Toc173484563)

[Lab Report 7](#_Toc173484564)

# Team Size

The team size for this lab is **2**.

# Goals

You should understand the following concepts by the end of the lab:

* real-time data acquisition
* time jitter
* critical sections
* methods to measure noise and calculate SNR
* probability mass functions (PMF)
* central limit theorem (CLT)

# Review

* Valvano Section 2.4 on GPIO, Chapter 10 of data sheet
* Valvano Sections 3.9, 5.9 on debugging,
* Valvano Section 5.3 on critical sections,
* Valvano Sections 5.7, 6.2 on periodic timer interrupts, Chapter 11 of data sheet
* Valvano Section 8.5 on the ADC, Chapter 13 of data sheet
* Valvano Sections 10.1, 10.4 and 10.5 on data acquisition systems and noise
* Logic analyzer instructions.

# Starter Files

* Example projects from <https://users.ece.utexas.edu/%7Evalvano/arm/>
  + PeriodicTimer2AInts\_4C123 (review periodic interrupts)
  + ADCSWTrigger\_4C123 (review periodic timer interrupts and busywait ADC)
* Starter project:
  + Lab 2 template provided on the GH Classroom repo.

# Required Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| Parts | Datasheet | Price | Source (**price source)** |
| EK-TM4C123GXL | EK-TM4C123GXL | $16.99 | **TI** |
| GP2Y0A21YK0F  or GP2Y0A41SK0F sensor |  | $11.85 | **Mouser** |
| 10 uF tantalum capacitor |  |  |  |

# Lab Overview

In this lab you will design a simple data acquisition system, calculating a physical parameter and measuring noise. The first option is to measure distance, and the second option is to measure resistance. You will use the oscilloscope, spectrum analyzer, and logic analyzer, and then learn how to profile code using dumps to measure sampling jitter and noise.

You are expected to learn how to use the oscilloscope, spectrum analyzer and logic analyzer in this class, so please ask your TA for a demonstration in the lab if you are unfamiliar with them.

To apply the **Central Limit Theorem**, we must assume the noise is random, i.e., the noise in each sample is independent from the noise in the other samples, and the noise has zero mean. Look up the ADC Sample Averaging Control (**ADC0\_SAC\_R**) register in Chapter 13 of the data sheet. The Central Limit Theorem (CLT) states: as the number of samples increases, the calculated average (your data) will approach the theoretical mean (true signal). The CLT also states that regardless of the original probability density function (PDF) of the noise, the PDF of the averaged signal will become Gaussian.

# Preparation

Preparation is performed during or before the W/TH lab session.

1. Write software to sample the ADC at 1000 Hz. You may use any of the functions on the class web site or the book. Three possible ADC inputs are PE4, PE5, or PE1. The 1000 Hz sampling rate should be created by a periodic timer, and the ADC synchronization should be busy wait.
2. Toggle PC7 three times in the ISR, and toggle PC6 once each time through the main loop. Compile, debug, and copy-paste the compiler-generated assembly code of the single toggle of PC6 in the main program. Initially, use this method to toggle a pin creating a critical section bug. The bug is induced by the nonatomic read-modify-write access to the shared global.

**GPIO\_PORTC\_DATA\_R ^= 0x80; // toggle PC7**

**GPIO\_PORTC\_DATA\_R ^= 0x40; // toggle PC6**

Highlight the critical section in the assembly code. More specifically, specify between which instructions causes a bug if the interrupt were to occur. Be prepared to explain each line of this assembly code during checkout. Your assembly code should be like the following

0x00000DA4 4812 LDR r0,[pc,#72]

0x00000D9A 6880 LDR r0,[r0,#0x08]

0x00000D9C F0800002 EOR r0,r0,#0x02

0x00000DA0 491D LDR r1,[pc,#116]

**0x00000DA2 6088 STR r0,[r1,#0x08]**

1. The starter project has a function, **Timer1\_Init**, which activates timer 1 without interrupts. Reading **TIMER1\_TAR\_R** will return the 32-bit current time in 12.5ns units. This timer counts down. Write software to measure sampling jitter. You will read **TIMER1\_TAR\_R** every time you sample the ADC. Subtracting two times yields the actual ADC period in 12.5ns. If sampling were perfect each time difference would be exactly 80000 (1ms). Write code that records 1000 ADC samples (**DataBuf**) and 1000 time measurements (**TimeBuf**). Calculate sampling jitter as

*MaxT* = maximum(**TimeBuf**[i-1] – **TimeBuf**[i]) for **i** equals 1 to 999

*MinT* = minimum(**TimeBuf**[i-1] – **TimeBuf**[i]) for **i** equals 1 to 999

*Jitter* = *MaxT* – *MinT*

In a **hard-real time system** we do not report the typical or average behavior; rather, we report the worst-case timing. A hard real-time system guarantees every sample was properly collected.

1. Write software to measure the average and standard deviation of the 1000 ADC samples. Assuming the input is constant, we categorize the average as **Signal**, and the standard deviation as **Noise**. Calculate *SNR* as the average/standard deviation. Do not use floating point. There are many definitions for effective number of bits (*ENOB*). However, in this lab we use

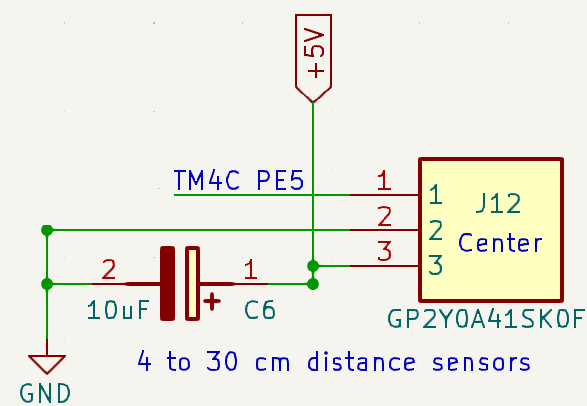
*ENOB* = log2(*SNR***)**

1. A probability mass function (PMF) is the number of observations plotted versus observed input. A system with a Gaussian noise process has a PMF shaped like a bell-curve. Observe the starter code includes the function **CreatePMF**, which calculates a PMF of the data. You will use this PMF to study the effects of signal averaging on the noise distribution according to the CLT.

# Procedure

Procedure is performed during or before the W/TH lab session.

1. **Connect the analog circuit** to an ADC input, PE4, PE5, or PE1. Option 1 is to measure distance with an IR sensor, see left side Figure 2.1a or Figure 2.1b. The 10 uF capacitor should be placed as close to the sensor as possible. Option 2 is to measure resistance *R*, see right side of Figure 2.1.

 A diagram of a circuit

Description automatically generated

Figure 2.1a. Possible hardware connections to create an analog input. The first option uses the IR sensor to measure distance. The second option measures resistance R. The 1k resistor is known and fixed, and the resistance R is unknown

A diagram of a computer

Description automatically generated

Figure 2.1b. Reduced function RSLK robot hardware connections to create an analog input.

1. **Measuring noise with the DVM**. Configure the system with a constant distance or fixed resistor R. Take measurements at the ADC input (e.g., PE5). Measure DC voltage (signal), and then measure AC voltage (noise). Calculate signal to noise ratio, SNR = signal/noise, ENOB=log2(SNR). Note: these calculations will vary with the amplitude of the input. Perform one measurement where the analog voltage is between 1.65 and 3.3V.
2. **Measuring noise with the** **oscilloscope**. Use an oscilloscope to view the voltage at the ADC input (e.g., PE5). Using DC coupling mode measure DC voltage (signal). Using AC coupling mode measure AC RMS voltage (noise). Calculate signal to noise ratio, SNR = signal/noise, ENOB=log2(SNR). Note: these calculations will vary with the amplitude of the input. Perform one measurement where the analog voltage is between 1.65 and 3.3V. The image should show the magnitude of the noise in either AC RMS or DC Peak-to-Peak. This can be done via the cursors or adding a dynamic measurement. See Figure 2.2.
3. **Observing the noise profile with a spectrum analyzer**. Use an oscilloscope to view the voltage at the ADC input (e.g., PE5). Using spectrum analyzer mode observe the noise versus frequency. Select a sampling rate that allows you to see frequencies from 0 to 1000 Hz. See Figure 2.3.
4. **Measuring noise using software**. Using the same configuration as steps 2 – 4, run the software sampling (without hardware averaging) and report the calculations of Signal, Noise, and SNR from your software. Calculate ENOB=log2(SNR)
5. **Measuring time jitter.** Report the time jitter of your real-time data acquisition system.
6. **Observing the critical section.** Connect a scope or logic analyzer to PC7 and PC6. PC7 should toggle 3 times every 1ms. Capture a trace like Figure 2.4 showing the effect of the critical section bug. Edit the software removing the bug. There are lots of ways to remove the bug, pick one to implement and discuss the other options in the report. Figure 2.5 shows the proper behavior.
7. **Explore the behavior of the CLT.** Configure the system with a constant distance or fixed resistor R. Make a table of SNR and ENOB, as calculated by your software, as a function of hardware averaging, with ADC0\_SAC\_R = 0 – 6. Pick your choice for averaging and download and plot the PMF for no averaging and your choice, like Figure 2.6. Note: The PMF data are output to the PC via UART0, and you can capture the data with a terminal program like Putty or TExaSdisplay. If you compare two PMFs with the same SAC value, you will not get the same result because the noise is not stationary
8. **Calibrate the data acquisition system.** Use four measurands (distance or resistors) of known value. Using the averaging settings chosen in step 8, record the **Signal** for each measurand. Using a nonlinear equation fit the four calibration points to determine the calibrations constant A and B. See the **Lab2Calibration.xlsx** file. Calculate the average error of the calibration.

Option 1) Distance (mm) d = A/(Signal + B);  
Option 2) Resistance (ohms) R = (Signal\*A)/(B - Signal)

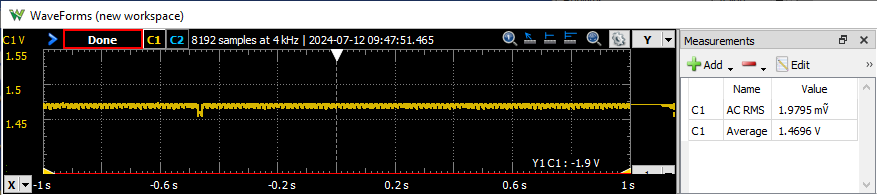


Figure 2.2. Analog voltage versus time measured with a real oscilloscope. Signal =1.5V, Noise=2mV, SNR = 1.7/0.055= 742, which is ENOB= log2(742) =9.5 bits.

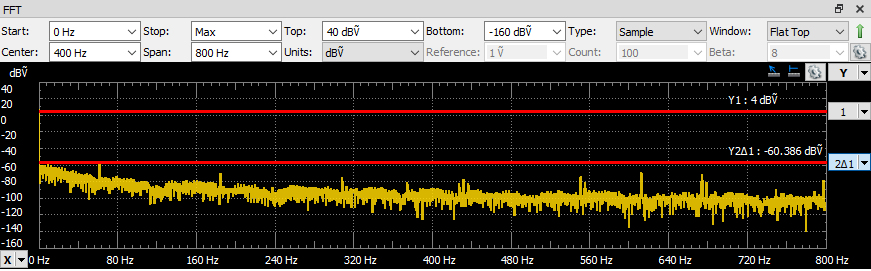


Figure 2.3. Analog voltage versus frequency measured with a real spectrum analyzer. Signal = 4dB, Noise=-60dB, SNR=64dB, 1064/20 = 1584, ENOB=log2(1584) = 10.6 bits.

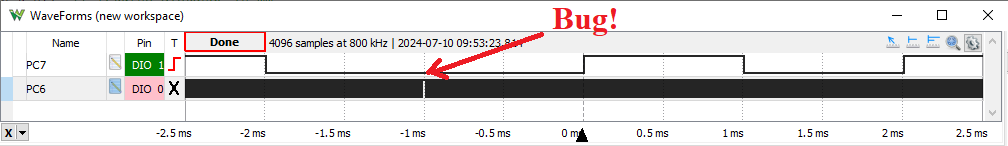


Figure 2.4. Logic analyzer trace showing the critical section bug. PC7 was toggled three times every 1ms, but right at time=-1ms the toggles are missing.

A screenshot of a computer

Description automatically generated  
Figure 2.5. Zoomed in view. Note that the main program does not run (As evidenced by PC6 no longer toggling) while the Timer ISR is running. Also note that the time to execute the Timer ISR is about 10us (this is the time between the first and third toggle of PC7; 8us of this 10us occurs converting the ADC, which is running with ADC0\_SAC\_R=0).

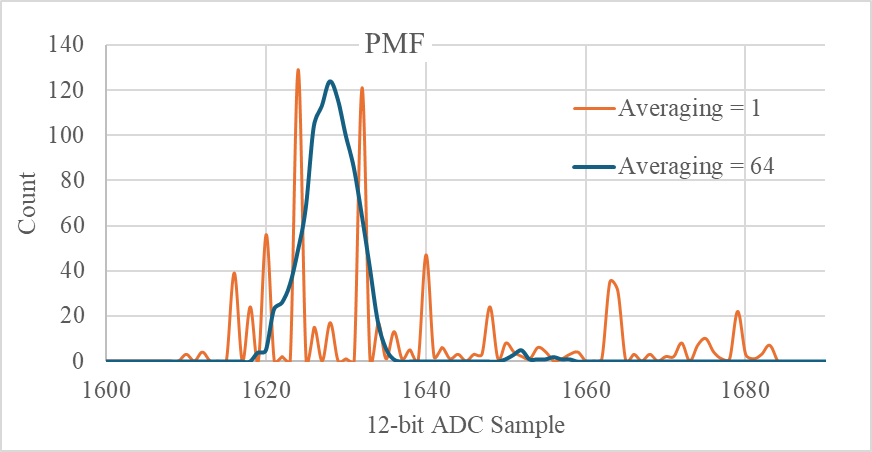


Figure 2.6. The input distance was constant, the ADC was sampled 1000 times at 1 kHz. The PMF is the count versus ADC sample.

Note: If we were to use timer-triggered ADC sampling, even with hardware averaging, there would be no jitter

# Lab Checkout

The lab checkout is performed during the M/T lab session.

You should be able to demonstrate:

* Your understanding of the logic analyzer and scope features listed.
* Any of the deliverables: how the data were collected and what they mean.

# Lab Report

The lab report shall be submitted by the Friday after the second lab section.

You should complete the Lab02Report.docx file with your data and answers then submit the completed file to Canvas