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 © 2021.

## Goals

• To review the TM4C123 LaunchPad

• To familiarize yourself with Keil uVision5 for the ARM Cortex M processor

• To implement and test a data acquisition system.

## Review

• “How to program…” section located at the beginning of this laboratory manual,

• Read [Section 3.4 Interfacing to the TM4C123](https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch3_IntroToInterfacing.html#3_4)

• Read [Sections 6.3-6.5 Interrupts on the TM4C123](https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch6_DACSound.html#6_3),

• Read [Section 8.3 Details of the ADC on the TM4C123](https://users.ece.utexas.edu/%7Evalvano/Volume1/IntroToEmbSys/Ch8_ADC.htm#8_3)

## Starter files these projects are in ValvanoWare

• **ADCSWTrigger\_4C123** project

• **PeriodicSysTickInts\_4C123** project

• **GPIO\_4C123** project

• **ST7735\_4C123** project

**Pull the Lab1 project from GitHub**

**Team Size: 1 (each student performs Lab 1 on their own)**

## Required Hardware

EK-TM4C123GXL [http://www.ti.com](http://www.ti.com/tool/ek-tm4c123gxl?keyMatch=tm4c123g&tisearch=Search-EN-Everything)  $16.99

## Optional Hardware

Sitronix ST7735 Color LCD <http://www.adafruit.com/products/358> $19.99

Resistors

LED

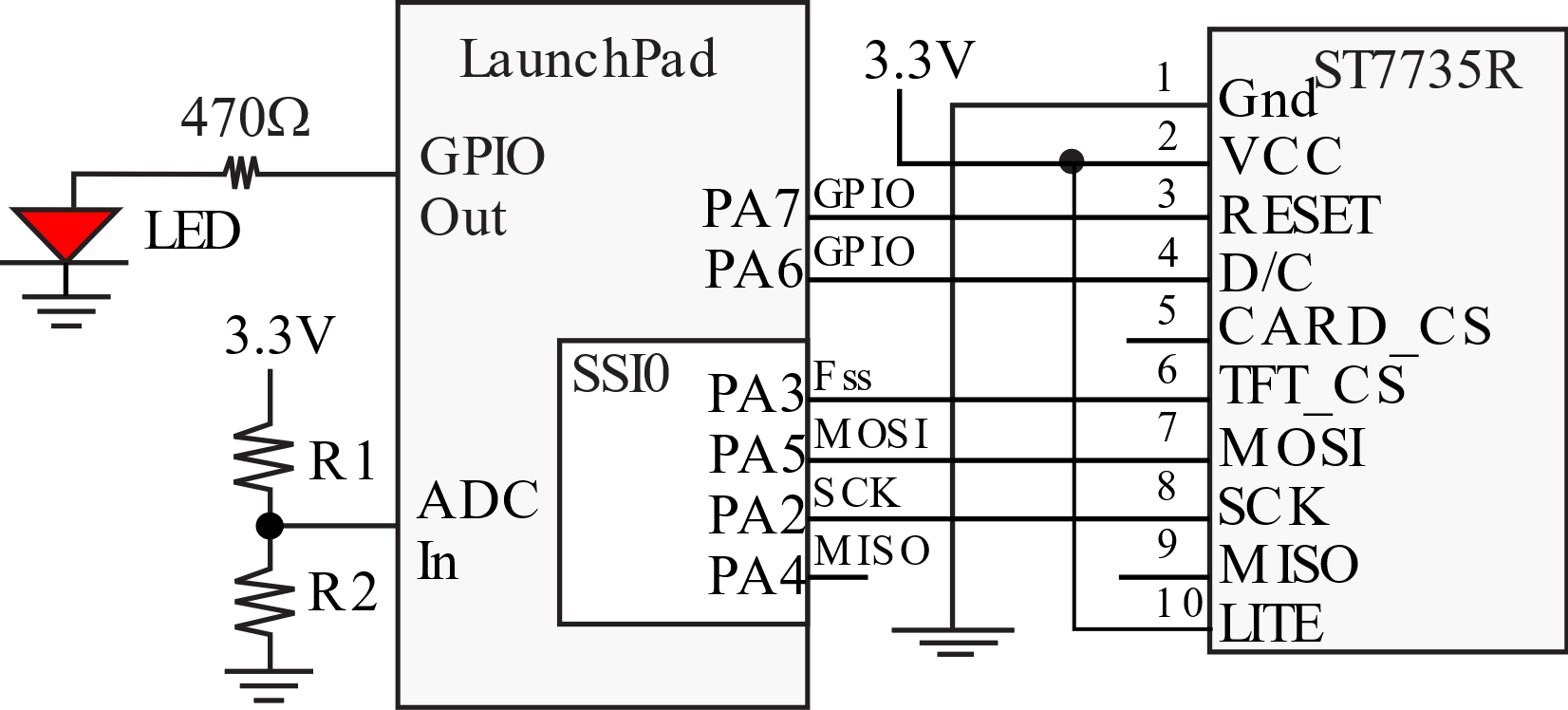
## Background

The objectives of this lab are to review the TM4C123 programming environment and put together some basic routines to create a simple data acquisition system as illustrated in Figure 1.1. The resistance *R1* is constant and known. The resistance *R2* is unknown. The goal is to measure resistance *R2* in ohms. The range of R2 is 0 to 100000 ohms. The GPIO output will provide a debugging heartbeat. The resistance measurement will be displayed on the LCD. The analog input voltage will be

*Vin* = 3.3V \**R2*/(*R1+R2*)

The 12-bit ADC sample will be

*Data* = integer(4095 \* *R2*/(*R1+R2*))



*Figure 1.1. Simple data acquisition system to measure unknown resistance, R2.*

Program 1.1 shows the real-time data acquisition system

**uint32\_t R2; // measured unknown resistance in ohms**

**uint32\_t Data;**

**uint32\_t Flag;**

**int main(void){ //main10 use to run the data acquisition system**

**DisableInterrupts();**

**PLL\_Init(Bus80MHz);**

**ST7735\_InitR(INITR\_REDTAB);**

**ST7735\_FillScreen(0); // set screen to black**

**ST7735\_OutString("Real-time DAS\n");**

**ST7735\_OutString(EID);**

**ST7735\_OutString("\nR2= "); ST7735\_OutUDec5(R2); ST7735\_OutString(" ohms");**

**Pin\_Init(); // your function**

**ADC\_Init(); // your function**

**SysTick(); // your function**

**EnableInterrupts();**

**while(1){**

**if(Flag){ // wait for semaphore**

**Flag = 0;**

**R2 = Convert(Data);**

**ST7735\_SetCursor(4,2);**

**ST7735\_OutUDec5(R2);**

**}**

**}**

**}**

**// Interrupt service routine**

**// Executed at fs**

**void SysTick\_Handler(void){**

**Pin\_Out(1); // debugging heartbeat**

**Data = ADC\_In(); // sample**

**Flag = 1; // set semaphore**

**Pin\_Out(0); // debugging heartbeat**

**}**

*Program 1.1. Simple data acquisition system to measure unknown resistance, R2.*

## Specifications

You will develop and test these five functions

**Pin\_Init** Initialize a pin for GPIO output

**Pin\_Out** Set the output pin high or low

**ADC\_Init** Initialize a pin for ADC input

**Convert** Convert 12-bit ADC sample to measured resistance in ohms

**SysTick** Initialize SysTick for real time sampling

Which pin to use for the GPIO output, which pin to use for the ADC input, the value of R1, and the sampling rate will be selected at random based on your EID. Running the starter project will reveal these specifications.

## Procedure (do this during your lab period)

Since there is an autograder, all you need is the LaunchPad to complete this lab. Optionally, you may wire up the LED, resistors, and LCD to see the actual resistance measurement operate. If you do not have a LaunchPad but have a Windows machine, you can complete the lab in simulation. Copy this file

<https://www.dropbox.com/s/azig59lbopm1nmb/LaunchPadDLL.dll?dl=1> into this in \Keil\ARM\BIN folder. Select simulation in the debug settings. See Hint 3.

**0.** If you bought your LaunchPad used, ask the TA how to run the tester project.

**1.** Run the Lab1 project (enter your EID and run to see your specific requirements)

You can see the output on a serial terminal like TExaSdisplay or Putty (115200 bits/sec)

Using your value for *R1*, derive a function for *R2* (in ohms) given measured ADC *Data*

Floating-point code is not allowed for this **Convert** function.

**2.** Run these example projects as a review of ECE319K:

GPIO\_4C123 (review GPIO input/output)

PeriodicSysTickInts\_4C123 (review periodic interrupts)

ADCSWTrigger\_4C123 (review periodic timer interrupts and busy-wait ADC)

**3.** Experiment with the different features of uVision5 and its debugger. Familiarize yourself with the various options and features available in the editor/assembler/terminal. Edit, compile, download, and run your project working through all aspects of software development. In particular:

• learn how to remove all tabs (Edit->Advanced->UntabifySelection);

• learn how to comment and uncomment large sections of code;

• know how to compile, download, and debug a project;

• in the debugger run with and without View->PeriodicWindowUpdate;

• in the debugger observe assembly listings visualizing the correlation between C and assembly;

• set breakpoints, add global variables to watch window, observe I/O device registers in debugger;

**4.** Develop and debug your software modules. Please work **main1** through **main8** before running the grader with **main9**.

**5.** Run the grader, **main9**, to determine your score. The maximum score is 100

**Lab 1, ECE445L, EID=ABC1234**

**GPIO output pin = PB4**

**ADC input channel = 2**

**Sampling rate = 17 Hz**

**R1 = 2200 Ohms**

**Calling Pin\_Init**

**Calling Pin\_Out(0)**

**Calling Pin\_Out(1) Score = 20**

**Calling ADC\_Init Score = 40**

**Calling Convert 8 times Score = 80**

**Calling SysTick once Score = 100**

**6.** Optional run **main10** to observe the real-time DAS system.

**7.** Run **main11** to perform an empirical study to evaluate four implementations of a linear equation. Two implements use fixed-point, two use floating-point, two are written in assembly, and two are written in C. For each implementation, **main11** measures the total execution time. Make conclusions about implementing arithmetic operations on the Cortex M4.

## Deliverables (see Lab01Report.docx)

## Checkout (show this to the TA)

Lab 1 is different from other labs. For Lab1 simply upload the **Lab01Report.docx**. There is no checkout.

## Hints

1) Do not create a new project. Start with a project you know works, make a copy of the project and, then make small changes. It is good practice to make a small change and test it. Once you have some new code that works, make a back-up, so that when you add something that doesn’t work, you can go back to a previous working version and try a new approach. Please add documentation that makes it easier to change and use in the future. Your job is to organize these routines to facilitate subsequent laboratories.

2) It is also good practice to look at the assembly language created by the compiler to verify the appropriate function. Analyzing the assembly listing files is an excellent way to double-check if your software will perform the intended function. This is especially true when overflow, dropout, and execution speed are important. We have not found any bugs with this compiler. Most reported compiler bugs (my program doesn’t do what I want) turn out to be programmer errors or misunderstanding about the C language. However, if you think you’ve found a bug, email the source and assembly listing to the TA explaining where the bug is.

3) To simulate, put LaunchPadDLL.dll in \Keil\ARM\BIN, and set the debug options (as shown on the next page)

A screenshot of a social media post

Description automatically generated