University of British Columbia
Electrical and Computer Engineering
ELEC291/ELEC292

Module 4 –Data Logging using Python

Introduction

Embedded systems are often designed to perform simple or repetitive tasks while connected to larger computers. Examples of such systems are found in many of today's computers: the mouse, keyboard, memory sticks, hard drive controllers, etc. For this module you will build one of such devices: an embedded digital thermometer using a microcontroller system. The digital thermometer will serially transmit the temperature to a personal computer using the serial port. You will program the personal computer using Python to receive the temperature and conveniently present it in real time using a strip chart plot.

There are many free python distributions available. One that has all the functionality to complete this laboratory module is WinPython version 3 available at:

https://winpython.github.io/

References

C51 user manual included with the latest version of CrossIDE.

LM335 data sheet

EFM8LB12 reference manual.

Python reference manual. Available online.

Laboratory

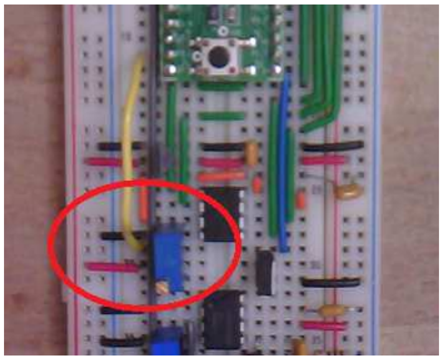
1) **Testing the Serial Port Using C with the EFM8 board.** The program below prints "Hello, world!" in PUTTY running in a computer throughout the serial port of the EFM8LB1. Copy and paste it to Crosside and save it to 'hello.c'. To compile it under Crosside with C51, click 'Build' → 'Compile/link with C51'. Make sure you set the 'Complete path to C51.exe' correctly. After compiling, load the resulting '.hex' file to the EFM8 board.

```
#include <EFM8LB1.h>
#include <stdio.h>

#define SYSCLK      24500000L  // SYSCLK frequency in Hz
#define BAUDRATE      115200L  // Baud rate of UART in bps

char _c51_external_startup (void)
{
        // Disable Watchdog with key sequence
        SFRPAGE = 0x00;
        WDTCN = 0xDE; //First key
        WDTCN = 0xAD; //Second key
```

```
		VDM0CN |= 0x80;
		RSTSRC = 0x02;

		// Use a 24.5MHz clock
		SFRPAGE = 0x00;
		CLKSEL  = 0x00;
		CLKSEL  = 0x00;
		while ((CLKSEL & 0x80) == 0)

		P0MDOUT |= 0x10; // Enable UART0 TX as push-pull output
		XBR0     = 0x01; // Enable UART0 on P0.4(TX) and P0.5(RX)
		XBR1     = 0X00;
		XBR2     = 0x40; // Enable crossbar and weak pull-ups

		// Configure Uart 0
		SCON0 = 0x10;
		CKCON0 |= 0b_0000_1000 ; // Timer 1 uses the system clock.
		TH1 = 0x100-((SYSCLK/BAUDRATE)/2L);
		TL1 = TH1;      // Init Timer1
		TMOD &= ~0xf0;  // TMOD: timer 1 in 8-bit auto-reload
		TMOD |=  0x20;
		TR1 = 1; // START Timer1
		TI = 1;  // Indicate TX0 ready

		return 0;
	}

	void main (void)
	{
		printf( "Hello, world!\r\n" );
	}
```

PUTTY is a free Telnet/SSH/Serial terminal that can be downloaded from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. It is possible to launch PUTTY directly from CrossIDE by pressing <Control>+T. Configure PUTTY to 115200 baud, 8 bits, parity none, 1 stop bits, and Flow Control 'none'. Also make sure the 'Complete path of PuTTY.exe' field points to a valid location.

2) **ADC Converter.** Available on Canvas is the test program 'ADC_EFM8.c'. This program reads the analog voltage at some pins of the microcontroller using the built-in 14-bit analog to digital converter (ADC). Compile, load, and test the program. The picture below shows a 10k potentiometer acting as a voltage divider, with the output connected to one of the ADC inputs (P2.2) to test the provide program.

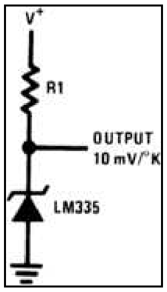


3) **Using Python to communicate with the EFM8LB1 Microcontroller.** The Python script shown below opens the serial port in the host computer, constantly

reads and prints a received value, and finally closes the serial port when CTRL+C is pressed in Python's command console. Attach an LM335 temperature sensor to one ADC capable pin of the EFM8 microcontroller board. Modify the program you wrote for the previous point so it converts the acquired value to temperature and transmits it through the serial port every second.

```
import time
import serial

# configure the serial port
ser = serial.Serial(
    port='COM1',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_TWO,
    bytesize=serial.EIGHTBITS
)
ser.isOpen()

while 1 :
    strin = ser.readline()
    print (strin)
```

The script above assumes you are using COM1. For other serial ports, adjust accordingly. Also, Python expects a new line escape sequence ('\n') for each received value from the microcontroller. Connect the LM335 as shown in the figure below. Make V^+=5V and R1=2.0kΩ. To observe different temperature readings, you can heat up the LM335 by pressing it with your fingers.



4) **Temperature strip-chart using Python.** The script 'stripchart_sinewave.py' shows how to implement strip-charts in Python. A strip-chart can be used to plot the temperature transmitted from the EFM8 microcontroller board to Python in real time. Modify the provided script so it plots the data received from the serial port. Don't forget to add extra functionality and/or features for bonus marks! Upload to canvas:

a) Python code.

b) C code.

c) ONE good resolution picture of the microcontroller system with the LM335 attached.

d) A video of showing the temperature strip chart working.