University of British Columbia
Electrical and Computer Engineering
ELEC291

Lab 4 – Temperature 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/

For this lab you are allowed to work with a partner.

References

C51 user manual included with the latest version of CrossIDE.

LM335 data sheet

AT89LP51RC2 reference manual.

Python reference manual. Available online.

Laboratory

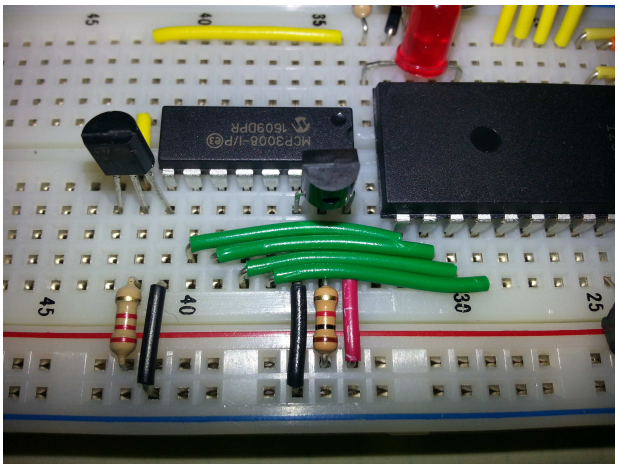
1) **Testing the Serial Port Using C with the AT89LP51RC2.** The program below prints "Hello, world!" in PuTTY running in a computer throughout the serial port of the AT89LP51RC2. Copy and paste it to Crosside and save it to 'hello.c'. To compile it under Crosside with C51, click 'Build' → 'Compile/link with C51'. Make sure you set the 'Complete path to C51.exe' correctly. After compiling, load the resulting '.hex' file to the AT89LP51RC2 micro controller.

```
//  AT89LP51RC2 "Hello, World!" example.
//  ~C51~

#include <stdio.h>
#include <at89lp51rd2.h>

#define CLK    22118400L // SYSCLK frequency in Hz
#define BAUD     115200L // Baud rate of UART in bps
#if ((CLK/(16L*BAUD))>0x100L)
#error "Can not set baud rate because (CLK/(16*BAUD)) > 0x100 "
```

```
#endif
#define BRG_VAL (0x100-(CLK/(16L*BAUD)))

unsigned char _c51_external_startup(void)
{
    AUXR=0B_0001_0001; // 1152 bytes of internal XDATA, P4.4 is I/O

    // Configure serial port and baud rate
    PCON|=0x80;
    SCON = 0x52;
    BDRCON=0;
    BRL=BRG_VAL;
    BDRCON=BRR|TBCK|RBCK|SPD;

    return 0;
}

void main (void)
{
    printf("Hello, World!\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, 2 stop bits, and Flow Control 'none'. Also make sure the 'Complete path of PuTTY.exe' field points to a valid location. Press the reset button of the AT89LP51RC2 to see "Hello, World!" printed in PuTTY.

2) **ADC Converter.** Available on Canvas is the test program 'ADC_SPI.c'. This program reads the analog inputs of an external ADC using SPI. The ADC available in your parts kit this year is the MCP3008 made by Microchip. Compile, load, and test the program. The picture below shows a LM335 temperature sensor output connected to analog input '0' of the MCP3008 as explained in the next point. The voltage measured from analog input '0' should be very close to 3V.

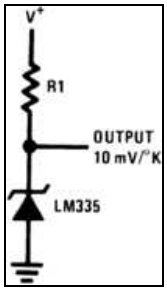


3) **Using Python to communicate with the AT89LP51RC2 Microcontroller.** The Python script shown below opens the serial port in the host computer, constantly

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 input of the MCP3008. 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 100 mili-seconds.

```
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 ('\r' will not work). 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 or touching it with the hot soldering iron available in the lab.



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 AT89LP51RC2 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 showing the temperature strip chart working.

Only one submission is needed from a team of two students. Please indicate with your submission the name and student number of the team members.