University of British Columbia
Electrical and Computer Engineering
ELEC291/ELEC292

Module 3 – SPI and 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 the AT89LP51RC2 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 either Matlab or 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

A51 user manual included with the latest version of CrossIDE.

AT89LP51RC2 user manual and MCP3008 data sheet.

Python reference manual. Available online.

Pre-laboratory

1) Find the datasheet of the MCP3008 10-bit 8-channel ADC. Draw in your notebook its pin out and describe the function of each pin.

2) Find the datasheet of the LM335 integrated circuit. Draw in your notebook its pin out and explain why and how it is used. What is the temperature range of operation of the LM335?

3) What is the difference between a regular plot and a 'strip chart' plot in Matlab/Python?

Laboratory

1) **Testing the Serial Port of the AT89LP51RC2 Microcontroller.** Available in course web page you'll find the program 'hello.asm'. This program prints "Hello, world!" in PuTTY running in a personal computers throughout the serial port of the AT89LP51RC2 microcontroller. Compile, load, and test this program using the AT89LP51RC2 microcontroller and verify that you can receive the message through the serial port of a computer using PuTTY.

   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) **SPI communication.** Attach an MCP3008 ADC to the AT89LP51RC2 microcontroller. You'll need to communicate the ADC and the microcontroller using the Serial Peripheral Interface (SPI) standard. In order to do so, you'll need an assembly subroutine to perform SPI communication. Bellow is a subroutine that implements SPI in software (bit-bang SPI).

```
; These 'EQU' must match the wiring between the microcontroller and ADC
CE_ADC    EQU  P2.0
MY_MOSI   EQU  P2.1
MY_MISO   EQU  P2.2
MY_SCLK   EQU  P2.3

INIT_SPI:
    setb MY_MISO    ; Make MISO an input pin
    clr MY_SCLK     ; For mode (0,0) SCLK is zero
    ret

DO_SPI_G:
    push acc
    mov R1, #0      ; Received byte stored in R1
    mov R2, #8      ; Loop counter (8-bits)
DO_SPI_G_LOOP:
    mov a, R0       ; Byte to write is in R0
    rlc a           ; Carry flag has bit to write
    mov R0, a
    mov MY_MOSI, c
    setb MY_SCLK    ; Transmit
    mov c, MY_MISO  ; Read received bit
    mov a, R1       ; Save received bit in R1
    rlc a
    mov R1, a
    clr MY_SCLK
    djnz R2, DO_SPI_G_LOOP
    pop acc
    ret
```

   Modify the 'hello.asm' program from the previous point so that it continuously prints the voltage from channel 0 of the MCP3008 ADC into PuTTY every second or so. In order to read a voltage from the ADC using SPI you may find useful figure 6-1 in the MCP3008 data sheet.

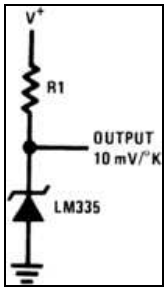
3) **Using Python to communicate with the AT89LP51RC2 Microcontroller.** The Python script shown below opens the serial port in the host computer, constantly reads and prints a received value, and finally closes the serial port when CTRL+C is pressed in Python's command console. Attach an LM335 temperature sensor to channel zero 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 second. To convert the voltage acquired from the LM335 sensor to temperature, a library of 32-bit arithmetic functions ('math32.inc') and an example file ('mathtest.asm') are available in the course web page for you to use.

```
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=2kΩ. To observe different temperature readings, you can **carefully** heat up the LM335 using the solder iron.



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 to Python in real time. Modify the provided script so it plots the data received from the serial port.

Demo the temperature strip-chart (in ºC) to you lab TA. Once again, you can use the solder iron to **carefully** heat the LM335 up! Don't forget to add extra functionality and/or features for bonus marks! Please upload to canvas:

a) Python code.

b) Assembly code.

c) ONE picture of the microcontroller system.

d) ONE Screen capture of temperature strip chart.

You can put all the files in a ZIP file and upload the ZIP file instead.