EE202A/CS213A Homework 3 Report

Chenguang Shen
cgshen@ucla.edu

1. Summary

In this homework I implemented a simplified oscilloscope using mbed m0 (LPC1768) and mbed m3 (LPC11u24). The mbed m3 is used to generating signal wave at analog output pin18, while the m0 is used to sample the signal from analog input pin16 - pin20. The pyserial library of Python is used on the PC side to communicate with the m0. The python program is able to plot sample data from the m0, and it can accept user command to dynamically control the plotting. The github repo of this homework is https://github.com/chenguangshen/MbedOscilloscope.

2. System Organization

The organization of the software is shown in Figure 1:

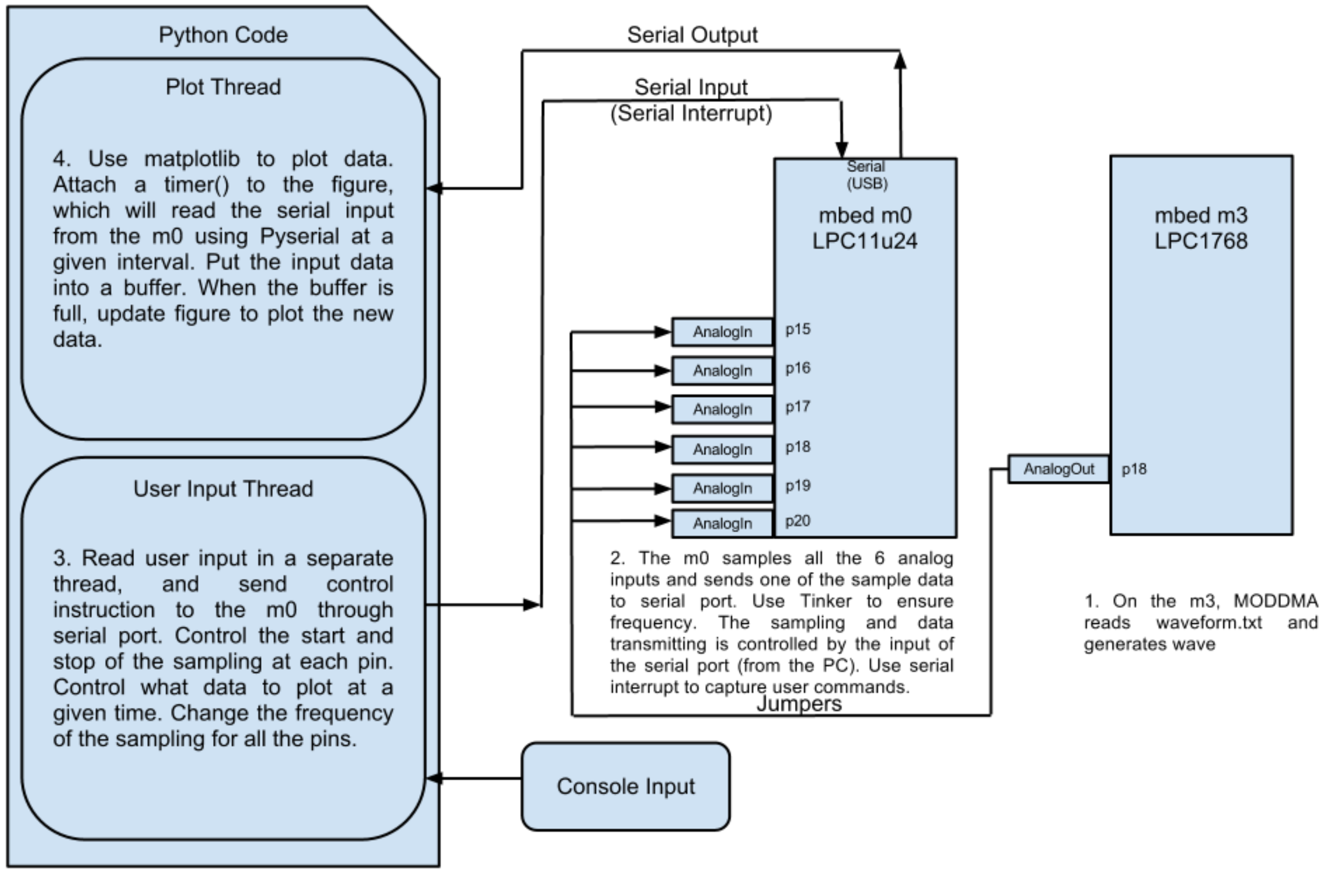
Python Code
Plot Thread
4. Use matplotlib to plot data. Attach a timer() to the figure, which will read the serial input from the m0 using Pyserial at a given interval. Put the input data into a buffer. When the buffer is full, update figure to plot the new data.
User Input Thread
3. Read user input in a separate thread, and send control instruction to the m0 through serial port. Control the start and stop of the sampling at each pin. Control what data to plot at a given time. Change the frequency of the sampling for all the pins.
Serial Output
Serial Input
(Serial Interrupt)
Serial
(USB)
mbed m0
LPC11u24
AnalogIn p15
AnalogIn p16
AnalogIn p17
AnalogIn p18
AnalogIn p19
AnalogIn p20
2. The m0 samples all the 6 analog inputs and sends one of the sample data to serial port. Use Tinker to ensure frequency. The sampling and data transmitting is controlled by the input of the serial port (from the PC). Use serial interrupt to capture user commands.
Jumpers
mbed m3
LPC1768
AnalogOut p18
1. On the m3, MODDMA reads waveform.txt and generates wave
Console Input

Figure 1: Software Organization

I have two versions of my program: single and multiple. The single version plots one figure at a time, and the user can select which analog input pin to plot at a time. Single version can achieve 1-1kHz sampling frequency. The multiple version plots 6 figures at a time, each corresponds to an analog input. However, it cannot achieve high frequency due to the limitation of serial port.

Here's some explanation about my code:

- (1) **Mbed m3:** I used the example code in MODDMA [1] to generate waves on the m3 side. The reason to use DMA is to speed up the data sending so that the output signal wave can achieve the 1kHz maximum. The program supports 3 types of waves: sine wave, square wave, and triangle wave. For simplicity, all of the three waves are stored in a buffer with the size of 360. After being generated, the buffer will be passed to the MODDMA for sending.
- (2) **Mbed m0:** I use Ticker to implement periodically sampling of analog input. the read_u16() function of analog input is used, instead of read().
  For single version, there are 6 Tickers, and each corresponds to an analog input. If one analog input is being sampled, the corresponding sample() function is attached to its tinker as callback. The sample() function only samples the signal from analog input pin, but does not print the value to the serial port. If user selects to plot the data from one analog input, the corresponding plot() function will be attached to its tinker as callback. The plot() function not only samples data, but also sends data over serial port. In order to input speed, I used putc() to send each digit of the analog input. On the python side I scaled the value back to the range [0, 3.3].
  For the multiple version, only one tinker is used, and a sample() function attached as callback. The callback function only samples the analog inputs that have been started by the user.
  In both case, Serial Interrupt is used to accept user command.
  If user changes the sampling frequency, the callback function (sample() or plot()) on the corresponding Tinker will be detached, and a callback with new frequency will be attached.
- **Python Code:**
  - (3) The main thread at python side uses matplotlib [3] to plot sample data. A timer is attached to the figure, and it will read sample data from serial input and save them to buffer at a given interval (1ms in my case). Pyserial [2] is used for serial communication. When the buffer size reaches a certain threshold, the figure will be updated using the new data in buffer. In addition, the label of X axis will change dynamically according to the current frequency.
  - (4) There is another thread at python side which is used to prompt a "mbed>" and receive user input. User can start, stop the sampling, change sampling frequency, and choose which analog input to plot (only for the single version). Note that at each time only one figure is plotted. On the m0 side these command will be captured by the serial interrupt. The accepted user command is shown as the following:

    start <pin15|pin16|pin17|pin18|pin19|pin20|all>
    Start sampling on an an input pin (or all pins). Pin status will be displayed.
    stop <pin15|pin16|pin17|pin18|pin19|pin20|all>
    Stop sampling on an input pin (or all pins). Pin status will be displayed.

freq [1..1000]
Change the sampling frequency on all pins.
plot *<pin15|pin16|pin17|pin18|pin19|pin20>*
Plot the data from a given input pin. (ONLY FOR THE SINGLE VERSION)
status
Show the sampling status of all pins.
exit
Exit the program elegantly.

3. Some Issues

As a CS student this is almost the first time I work on a sampling problem. Also this is the first time I use python to finish a homework. Therefore it took me some time to get the basic knowledge, and I still have some unsolved issues regarding this homework.
(1) In order to increase transfer speed, I change the baud rate of the serial port to be 115200 (original value is 9600). At this rate data can be transferred at 1kHz frequency. However, the noise also increases when baud rate is 115200. Sometimes the plotted line is not very smooth.
(2) The voltage of the USB port makes a difference. I once used a USB hub to connect the 2 Mbeds to my computer, but found some uncertainties about the serial data transmission.
(3) The multiple version has some bugs in that sometimes lots of error will be seen.

4. Reference

[1] MODDMA, http://mbed.org/cookbook/MODDMA
[2] Pyserial, http://pyserial.sourceforge.net/
[3] Matplotlib, http://matplotlib.org/