Fall 2017 Embedded Systems Lab

Introduction to Metronome

A. Project Definition

In this Embedded Systems Design Lab, a simple design and implementation of an embedded system will be performed in order to establish capability of embedded system design for various fields in the future. Especially, a project which requires no additional hardware is formulated: **Design of a Metronome**. It will be designed and implemented during this semester.

In the Wikipedia, metronome is explained as follows:

A **metronome** is any device that produces regular, metrical ticks (beats). These ticks represent a fixed, regular aural pulse; some metronomes also include synchronized visual motion (e.g. pendulum-swing). The metronome dates back to the early 19th century. Though the metronome was conceived as a tool for music, some musicians consider it to be a highly controversial tool in this respect: there are musicians who reject the metronome altogether.

Types of metronomes are as follows:

- Mechanical metronome
- Electronic metronomes
- Software metronomes

This image was originally posted to Flickr by Paco Vila at http://flickr.com/photos/48423254@N00/12294167. It was reviewed on 10:31, 19 April 2007 (UTC) by the FlickreviewR robot and confirmed to be licensed under the terms of the cc-by-2.0.

The Seiko DM50 is a lightweight clip-style metronome about the same size as a box of matches. Tempo range 30-250 bpm, Beat Accents 1-7, 1/8th, triplet and 1/16th notes. It has a 4-step volume control (off, low, mid, loud) and even a clock!

Project Definition

Design and implement a metronome using an embedded system having user input from the keyboard and output to LED (Light Emitting Diode) display output.

Required specifications are as follows.

Specification for metronome

User input

Input device	Keyboard. Local PC or Remote PC
Contents of input	
Tempo	Beats per minute. 30 to 200.
Time signature	2/4, 3/4, 4/4, or 6/8.
Start	
Stop	

Single key user input without Enter key.

LED Output

Beats (LED lamps on) with respect to time

Period	Given by Tempo input.
Number of LED lamps	4. Controlled by Time signature input. (3 strong, 2 medium, 1 weak)
Duty	50 %

Number of LED lamps output

Time signature 2/4:	3 1
Time signature 3/4:	3 1 1
Time signature 4/4:	3 1 2 1
Time signature 6/8:	3 1 1 2 1 1

Remote Text Output

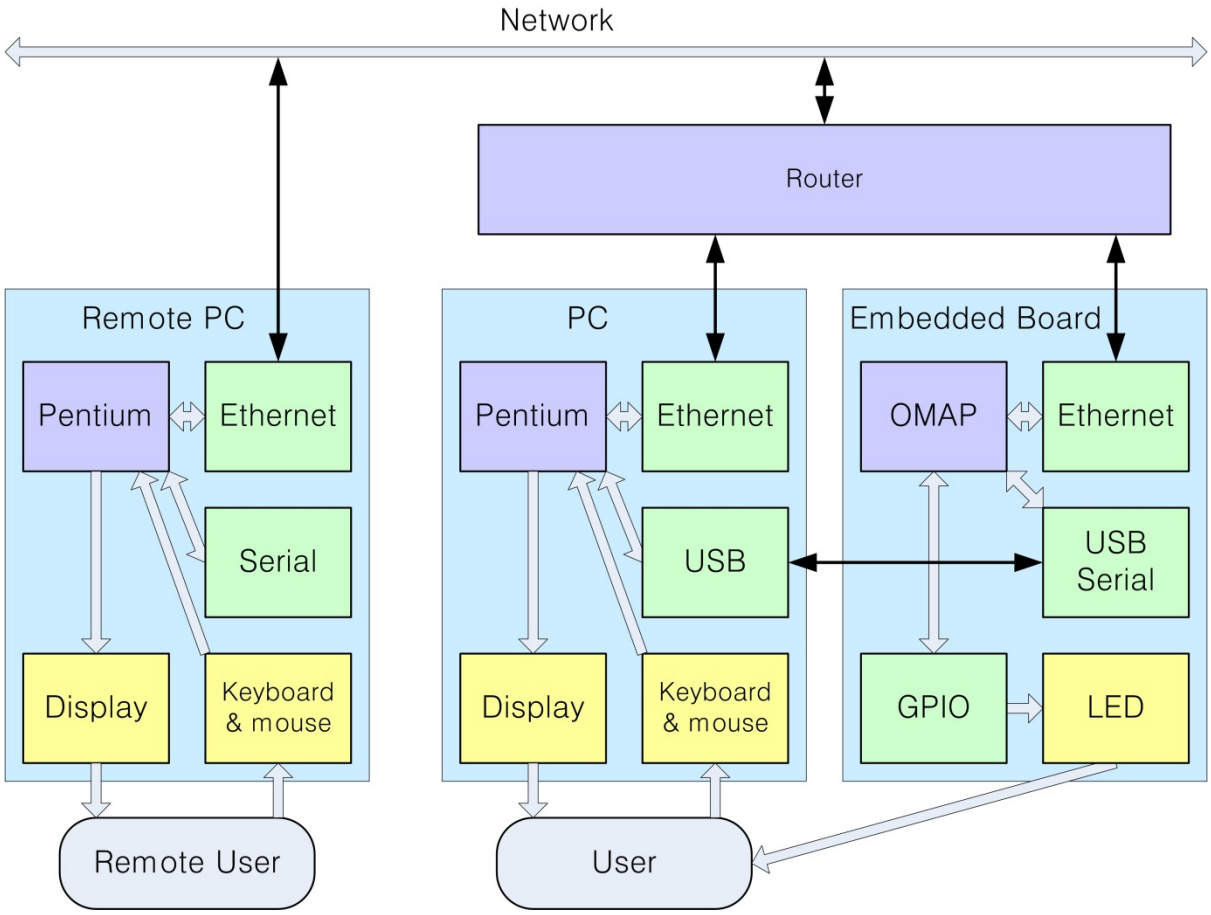
Text output with respect to time

Period	Given by Tempo input. One character display per each note.
Intensity	Denote by characters of #, +, and ! (# strong, + medium, ! weak)

Character pattern

Time signature 2/4:	# !
Time signature 3/4:	# ! !
Time signature 4/4:	# ! + !
Time signature 6/8:	# ! ! + ! !

Block Diagram of Metronome



Overview of Embedded Systems Lab

Metronome

Lab	Topic	Description	Input	Embedded software	Output
1	Cross-development system	Construct cross-development system and debug application	-	Simple embedded application program.	Text string
2	LED display	Program for LED display output	-	Program for LED display. Given tempo 60 and time-signature 6/8	LED display
3	Serial input	User input interface with single keys	User input via single keys	Get user input and drive LED accordingly.	LED display
4	Interrupt	Timebase generator using interrupt	User input via single keys	Timer interrupt signal handler for tempo.	LED display
5	Network input	Remote user input/output via Ethernet.	Remote user input via single keys	Get remote user input via Ethernet.	LED display and Remote text display

B. Lab Equipments

Equipments List

1 IBM PC with Windows 10 and Ubuntu 16.04 64-bit (Dual boot).

1 1 Gbps 4 port Router

1 Beaglebone set from beagleboard.org with
- 1 Beaglebone board (Credit card size)
- 1 USB cable.
- 1 Ethernet cable
- 1 5V 2A Power adaptor

1 4 GB micro SD (for program storage)

Rental of embedded board

The embedded board named Beaglebone can be borrowed for each group with duration of one semester with the following conditions.

1) Rent of Beaglebone set: From Technical Staff Gook Young Yoon in the Technical Staff Office.

2) Check the board: When you rent a board, please check if it is functioning correctly. If not, ask Technical Staff for a new board.

3) Usage: Use the board carefully. The board is not so strong! Check if everything is OK before you apply power to the board. Malfunction of the board should be fixed by each student group. A board with severe malfunction may be exchanged from the Technical Staff with a certain expense.

4) Returning the board: The embedded board set should be returned to Technical Staff, by the end of this semester. The board should be operational.

Technical Staff Office	Room 2358, N5 building
Laboratory	Room 2353, N5 building

C. Notes on Experiments

Embedded Systems Lab

I. Prereport for Lab

1. Understand specification and technical backgrounds

Read and understand the Lab text and related reference materials. Clearly understand the specification and technical backgrounds of each Lab.

2. Software program design procedure

Design your group's own software algorithm and program using C language with the following procedure.

a) Clarify the specification of the program: What are the input and the output, and what is the required function.
b) Divide into manageable functions if the program seems too big.
c) Design and write down your own algorithm for each function.
d) If the control flow is somewhat complex, drawing the flowchart may be helpful. For sequential process, flowchart is not necessary in general.
e) If multiple processes are running in parallel, drawing UML (Unified Modeling Language) sequence diagram is helpful, since it clearly shows multiple processes interacting with respect to time. http://www.websequencediagrams.com/ is a good site to draw your sequence diagrams instantly on the web.
f) Define your data to be used by user program. Use structure variable if required. Specify the type (binary, byte, integer, float, double, etc). Usually the data with binary, byte, integer will be designated as 'int'.
g) Code your header file containing function definitions, data, and define statements.
h) Code your functions and modules. Comment heavily in order to clarify what is your intention. Include exceptions: You should not assume that all the functions are correct. A function may return with various errors, and you have to handle these errors, in order not to proceed with errors.

II. Progress of Lab.

1. Check Lab equipment.

Check all the equipment on the laboratory desk using suitable methods. For successful lab, all equipment including PC, router, and embedded board should be error-free obviously.

2. Software implementation

Two students in each group may perform software test cooperatively.

A. Design your program containing algorithm, input/output.

Prepare two versions: Algorithmic program (hardware independent portion of your program) and Algorithmic + input/output program.

B. Develop Algorithmic program.

Using the Linux PC, edit, native-compile the Algorithmic program.
Re-edit if compile errors.
Use the debugger 'gdb' to check the operation of the program.
Repeat Edit, Compile, and Debug until all bugs are gone.

C. Develop Algorithmic plus input/output program.

Using the Linux PC, edit, cross-compile the Algorithmic plus input/output program.
Re-edit if compile errors.

On Beaglebone:
Connect cables and Turn on the power of Beaglebone carefully.
Download cross-compiled code to Beaglebone (or use NFS).
Use the debugger 'gdb' to check the operation of the program.
Repeat Edit, Compile, and Debug until all bugs are gone.

Note. Some input/output routines are time-dependent and hence it is hard to use gdb. You may store interim variables to memory, and check them later.

3. Demonstration

When all the hardware and software operation satisfies the problem requirement, demonstrate to TA.

4. Rearrangements

(1) Modify the flowchart and the program to make the final program listing. Print-out 2 copies of the program listing for two students in the group.

(2) Rearrange equipment on the laboratory desk for the next group.

III. Writing the main report

A final report for each lab is to be written and submitted to TA, which should include the following:

1) Introduction
2) Hardware block diagram
3) Software flowchart, program listing
4) Explanation of experimental results
5) Discussions
6) References

Caution: ***Each student should make his/her own report.*** *Even though the experimental results are the same for two students in the same group, discussions should be different.*