ECEN301 Embedded Systems Lab 7
Introduction to Embedded Linux

Daniel Eisen 300447549

October 16, 2020

1 Overview

The ARM processor that is core to the BeagleBone black is powerful enough to run an embedded operating system (In this case we use a Linux install). The benefit of using an preinstalled OS environment for developing a project over say a bare-metal implementation is that it enable a level of decoupling from the specific hardware and can be transferable between different devices running the same environment, make use of prewritten peripheral drivers and system libraries. These are particular useful when working on a larger/more complex piece of software with say networking etc.

2 Methodology

Due to the UNIX OS we can use an emulated serial connection via the USB port for all our connections.
One thing of importance/note is that everything within a UNIX operating system is defined/controlled via files, this includes thing like physical LEDs on the dev board.

This means to alter the output to an individual LED we write to the
`/sys/class/leds/beaglebone:green:usr`*N* directory with individual configuration files such as /brightness, and /trigger.
As these are file operations, the code we write is not limited to binary compilations i.e. from c-code. We can make use of the OS's shell scripting, a Python interpreter to call sys operation and do IO or we can if fact write c code and directly compile it on system with the install c build system (GCC).
We ran multiple prewritten scripts/programs using the above methods then expanded by extending the function of the c program to access multiple LEDs to accept a ASCII char and display its 4 least significant bits.
I achieved this by looping 4 times over a bit shift and masking operation to isolate the nth bit in from the char byte and write it to the brightness file.

Questions

1.
   - cd: change directory
   - ls: prints contents of current directory to terminal
   - mkdir: make directory
   - rm: file/folder removal with optional recursive function.

Appendix

```c
/** Simple On-board LED flashing program - written in C by Derek Molloy
*    simple functional struture for the Exploring BeagleBone book
*
*    This program uses USR LED 3 and can be executed in three ways:
*         makeLED on
*         makeLED off
*         makeLED flash  (flash at 100ms intervals - on 50ms/off 50ms)
*         makeLED status (get the trigger status)
*
* Written by Derek Molloy for the book "Exploring BeagleBone: Tools and
* Techniques for Building with Embedded Linux" by John Wiley & Sons, 2014
* ISBN 9781118935125. Please see the file README.md in the repository root
* directory for copyright and GNU GPLv3 license information.            */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LED0_PATH "/sys/class/leds/beaglebone:green:usr0"
#define LED1_PATH "/sys/class/leds/beaglebone:green:usr1"
#define LED2_PATH "/sys/class/leds/beaglebone:green:usr2"
#define LED3_PATH "/sys/class/leds/beaglebone:green:usr3"

void writeLED(char path[], char filename[], char value[]);
void removeTrigger();

int main(int argc, char* argv[])
{
    int c;
    char value[4];
    while(1){
        c = getchar();
        getchar();

        for (int i = 3; i >= 0; --i){
            sprintf(value, "%c",(c & (1 << i)) ? '1' : '0');
            removeTrigger();
            if (i==3) {
                writeLED(LED3_PATH, "/brightness", value);
            }else if (i==2) {
                writeLED(LED2_PATH, "/brightness", value);
            }else if (i==1){
                writeLED(LED1_PATH, "/brightness", value);
            }else if (i==0) {
                writeLED(LED0_PATH, "/brightness", value);
            }else {
                printf("default \n");
            }
        }
    }
    return 0;
}

void writeLED(char path[], char filename[], char value[])
{
    FILE* fp;   // create a file pointer fp
    char  fullFileName[100];  // to store the path and filename
    sprintf(fullFileName,"%s%s",path, filename); // write path and filename
    fp = fopen(fullFileName, "w+"); // open file for writing
    fprintf(fp, "%s", value);  // send the value to the file
    fclose(fp);  // close the file using the file pointer
}

void removeTrigger()
{
    writeLED(LED0_PATH,"/trigger", "none");
    writeLED(LED1_PATH,"/trigger", "none");
    writeLED(LED2_PATH,"/trigger", "none");
    writeLED(LED3_PATH,"/trigger", "none");
}
```