# ECEN301 Embedded Systems Lab 8
# Cross compiler IDE and GNU debug

Daniel Eisen 300447549

October 22, 2020

## 1 Objectives

Up until now we have either been writing small direct programs, flashing via JTAG or just writing and editing programs directly on the hardware. Sometimes even compiling it with build tools on the embedded OS.
For the large majority of real world cases, a developer would have a defined and setup dev environment that support debugging and cross-compiling on the host machine and only transferring/flashing the compiled binaries of the project to the device. This frees up resources in the device, makes use of much more powerful hardware for compiling and using and allows for a much more user friendly interface for development than terminal text-editor (though some may say they prefer it).

## 2 Methodology

To setup cross-compilation a new Code Composer project must be pointed to use the compiler and set of build tools and libraries that are for the device/device family in order to build a compatible set of binaries for the hardware. This is done with by setting the projects environmental/build variable to the correct file/dir. paths.
Now we are able to build a native executable for the embedded Linux install on any machine/OS, even Windows if its behaving.
To enable the transfer and remote execution and debugging of the project the beaglebone run a virtual SSH interface that we interacted with over USB and connected to like any other via the Remote Application config in CC.

### 2.1 Cylon

To write a small LED animation I extended teh clear LEDS function to clear all LEDS by accessing their brightness files. Then I added a "cylon" input argument and looped through the forward and back steps of the animations. See below.

## Appendix

```
/*
 * bbb_gpio_test.cpp
 *
 Simple On-board LED flashing program - written by Derek Molloy
 for the ee402 module

 This program uses USR LED 0 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)
 */

#include <iostream>
#include <fstream>
#include <string>
#include <cstdio>
#include <unistd.h>
using namespace std;

#define LED0_PATH "/sys/class/leds/beaglebone:green:usr0"
char LED_PATH[100] = "/sys/class/leds/beaglebone:green:usr1";

void removeTrigger()
{
    // remove the trigger from the LED
    std::fstream fs;
    for (int i = 0; i < 4; ++i)
    {
        sprintf(LED_PATH, "/sys/class/leds/beaglebone:green:usr%d/trigger", i);
        fs.open(LED_PATH, std::fstream::out);
        fs << "none";
    }
    fs.close();
}

void clearLEDS()
{
    std::fstream fs;
    fs.open("/sys/class/leds/beaglebone:green:usr0/brightness",
            std::fstream::out);
    fs << "0";
    fs.close();

    fs.open("/sys/class/leds/beaglebone:green:usr1/brightness",
            std::fstream::out);
    fs << "0";
    fs.close();

    fs.open("/sys/class/leds/beaglebone:green:usr2/brightness",
            std::fstream::out);
    fs << "0";
    fs.close();

    fs.open("/sys/class/leds/beaglebone:green:usr3/brightness",
            std::fstream::out);
    fs << "0";
    fs.close();
}

int main(int argc, char* argv[])
{
    if (argc != 2)
    {
        cout << "Usage is makeLED and one of: on, off, flash or status" << endl;
        cout << "e.g. makeLED flash" << endl;
    }

    string cmd(argv[1]);
    std::fstream fs;
    cout << "Starting the LED flash program" << endl;
    cout << "The LED Path is: " << LED0_PATH << endl;

    // select whether it is on, off or flash
    if (cmd == "on")
    {
        removeTrigger();
        fs.open(LED0_PATH "/brightness", std::fstream::out);
        fs << "1";
        fs.close();
    }
    else if (cmd == "off")
    {
        removeTrigger();
        fs.open(LED0_PATH "/brightness", std::fstream::out);
        fs << "0";
        fs.close();
    }
    else if (cmd == "flash")
    {
        fs.open(LED0_PATH "/trigger", std::fstream::out);
        fs << "timer";
        fs.close();
        fs.open(LED0_PATH "/delay_on", std::fstream::out);
        fs << "50";
        fs.close();
        fs.open(LED0_PATH "/delay_off", std::fstream::out);
        fs << "50";
        fs.close();
    }

    else if (cmd == "cylon")
    {
        removeTrigger();
        std::fstream fs;
        while (1)
        {
            for (int i = 1; i < 4; ++i)
            {
                clearLEDS();
                sprintf(LED_PATH,
                        "/sys/class/leds/beaglebone:green:usr%d/brightness", i);
                fs.open(LED_PATH, std::fstream::out);
                fs << "1";
                fs.close();

                for (int d = 0; d < 10000000; ++d)
                    ;
            }

            for (int i = 2; i > -1; --i)
            {
                clearLEDS();
                sprintf(LED_PATH,
                        "/sys/class/leds/beaglebone:green:usr%d/brightness", i);
                fs.open(LED_PATH, std::fstream::out);
                fs << "1";
                fs.close();

                for (int d = 0; d < 10000000; ++d)
                    ;
            }
        }

    }
    else if (cmd == "status")
    {
        // display the current trigger details
        fs.open( LED0_PATH "/trigger", std::fstream::in);
        string line;
        while (getline(fs, line))
            cout << line;
        fs.close();
    }
    else
    {
        cout << "Invalid command" << endl;
    }
    cout << "Finished the LED flash program" << endl;
    return 0;
}
```