CMPE12L: Lab Assignment 5-6

The C Language

Due December 6th

200 Points (double for lab5 and lab6)

Objective

In this lab, you will be introduced to the C programming language. You will utilize C in order to control hardware external to the RasberryPi.

Task 1 (30 points)

Quick Summary

Toggle a LED in the C programming language from your RasberryPi.

Detailed explanation

Use the wiringPI library to control the led. Instead of using the command line as in lab4, use a simple C program.

```
#include <stdio.h>
#include <wiringPi.h>

int main()
{
  printf("Raspberry Pi blink\n");

  int pin = 7;

  wiringPiSetup();
  pinMode(pin, OUTPUT) ;

  int i;
  for (i=0;i<10;i++) {

    printf("led on\n");
    digitalWrite(pin, HIGH);

    sleep(1);

    printf("led off\n");
    digitalWrite(pin, LOW);

    sleep(1);
  }

  return 0;
}
```

Compile with gcc and use the wiringPi library (be careful, it is case sensitive)

```
gcc  [your ucsc username]_lab5_task1.c -lwiringPi

To run it, you need superuser access
./a.out

# This works
sudo ./a.out
```

Task 2 (30 points)

Quick Summary

Toggle two LED C program

Detailed explanation

In this demo, you use a similar setup from task 1, but you need to connect an additional LED light (2 total) and reading from stdin.

The following program reads from the terminal without waiting for the return key being pressed. Modify it so that one led lights up if the user pushes a vowel, and the other illuminates if a consonant is pressed. When the program finishes, both LEDs should be off. E.g: if red is vowel, and blue is consonant, when I type “hello” it should illuminate in this sequence (blue, red, blue, blue, red). Both LEDs should not be on at the same time.

```
#include <stdio.h>
#include <stdlib.h>
#include <termios.h>


struct termios initial_settings,new_settings;

int main(int argc, char **argv) {

  tcgetattr(0,&initial_settings);
  new_settings = initial_settings;
  new_settings.c_lflag &= ~ICANON;
  new_settings.c_lflag &= ~ECHO;
  new_settings.c_lflag &= ~ISIG;
  new_settings.c_cc[VMIN] = 0;
  new_settings.c_cc[VTIME] = 0;

  tcsetattr(0, TCSANOW, &new_settings);

  while(1) {
    int n = getchar();

    if(n != EOF) {
      int key = n;

      if(key == 27 || key == 10) {
        break;
      }

    printf("[%c:%d]\n",key,key);
    }
  }

  tcsetattr(0, TCSANOW, &initial_settings);

  return 0;
}
```

Task 3 (30 points)

Quick Summary

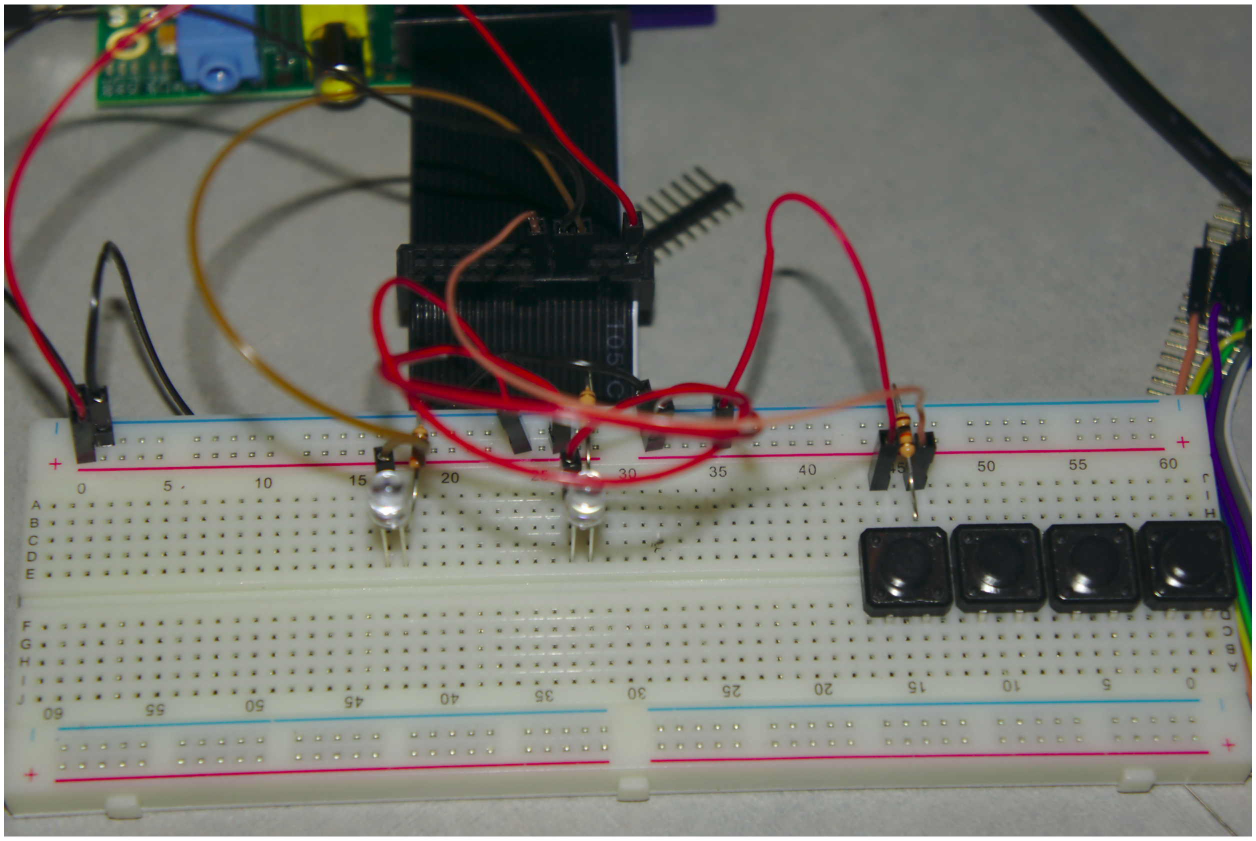
Read from a switch

Detailed explanation

In this task, you are going to connect a single push button switch to either VDD(+) or VSS(-), and you will be able to read, with the ‘gpio read’ command, either a zero (vss) or a one (vdd) at the command prompt.

Connect positive(3.3v) and negative(0v) wires to the breadboard (same as in lab 4) from the ribbon cable attached to the pi. Devise a way to wire either positive 3.3v or ground to the input of the push button. The output of the button should be connected back into a suitable pin on the ribbon cable so that the ‘gpio read’ command will register either a 0 or a 1 when the button is pressed (remember to set the correct mode with ‘gpio mode’ before).

After you can manually read the value of the gpio pin and you can see that pressing the button changes the value of the pin (by calling the command at the command line), create a C program that will constantly print out the value of the pin. If done correctly, you will see constant output at the command prompt that will change between 0 and 1 depending on the state of the button.
The photograph for a possible cabling solution:



In the image, only of of the switches is being used, but they are setup for the last task of the class (simon color game).
A very clean connection of 4 switches can be seen in this online demo
http://rcarduino.blogspot.com/2012_07_01_archive.html
Notice that the demo uses an arduino, we have a raspberry PI. The difference is that you need to pull the VDD, VSS, and GPIO from different pins. In my picture, I used GPIO0 (pin 17) if you use lab4 pictures.

Task 4 (30 points)

Quick Summary

Simon says with keyboard (no switches)

Detailed explanation

In this part, you need to connect 4 leds. To control what the simon says, the user can type the characters (a, s, d, e) in the keyboard that correspond to (red, green, blue, white) in the breadboard. A “c program” starts showing a single color (e.g: red), then the user has to type “a”. Otherwise all the colors blink and the game starts. If the user correctly typed “a”, then it shows red and some other random color (e.g: blue). Then the user has to type first “a” and the “d”. The simon says game should be able to memory 100 patterns.

Task 5 (60 points)

Quick Summary

Simon says with keyboard and/or switches

Detailed explanation

In this part, you need to connect 4 leds and the 4 switches. The game can be controlled either with the keyboard like task 4 or with the switches. Map the keys in the same order as the leds are shown in the breadboard (red, green, blue, white).
In this version of the program, you are FORCED to have small functions. You can have as many spaces or comments as you want, but there should be less than 10 statements per function code. “if (a) { a++; }” counts as 2 statements (if and a++). “if (a) { a++; }else{c=1;d=2;}” counts as 4 statements (if/else, a++, c=1, d=2). This constraint is to force you to structure your code and to have several functions.

Lab Write-up Requirements (20 points)

In the lab write-up we will be looking for a coherent report including the following sections:

- Introduction/Purpose
- Procedure
- Algorithm/Other Data
- Other Information
- Conclusion

You may add additional sections if you feel they are necessary.

In the lab report, be sure to address the following questions:

- What does sleep() do? What units of time does it use?
- What does EOF mean?
- What do each of the 3 #include do? Specify what including them allows you to do.
- Describe what each of the functions do and terms mean in task 2 (i.e: `tcsetattr`, `TCSANOW`, etc.)
- What is a struct in C? What is it used for? Give an example using pseudo-code.
- What is “\n” used for? When would you use it when programming?
- What is the difference between using brackets <> and quotations “” when #include-ing? Explain the difference in file type that would require brackets or

quotations
- Why do you return 0 at the end of *main()*?
- How many times will the *for* loop in task 1 run? Give an exact number.
- What different pins could be used as inputs for task 3, besides the one you chose?

Please save the report in PDF format as [your UCSC username]_lab5_report.pdf

Lab Submission

Please submit your lab on eCommons. We will be looking for the following files:
- [your ucsc username]_lab5_task1.c
- [your ucsc username]_lab5_task2.c
- [your ucsc username]_lab5_task3.c
- [your ucsc username]_lab5_report.pdf

Information on submitting assignments is available at http://its.ucsc.edu/ecommons/documentation/student/assignments-tool.html.

You can see a video on this submission process here at http://www.youtube.com/watch?v=doHDSDK9c3I.

Make sure to confirm that your assignment is SAVED and SUBMITTED before the deadline. You may resubmit your assignment an unlimited number of times up until the due date.

Check-Off (180 Points)

For this lab, as with all other labs, you will need to demonstrate your lab to the TA or tutor to get it signed off in addition to submitting it to eCommons. TO SAVE TIME, PLEASE ASK TO BE CHECKED OFF **ONLY** AFTER YOU HAVE SET UP THAT SPECIFIC PART. To get full points for check off, make sure:

- ❏ (20 pts) Part 1: Show that the program works
- ❏ (5 pts) Part 1: Answer a question regarding errors that would occur if you didn't include the '.h' files
- ❏ (25 pts) Part 2: Explain the conditionals that need to be checked and show that it works
- ❏ (5 pts) Part 2: Answer a question regarding more efficient ways to implement this part
- ❏ (20 pts) Part 3: Show that toggling the button gives the correct output
- ❏ (5 pts) Part 3: Be able to justify why you chose to tie the button to either 3.3v or 0v.