Lab 9: GUI for a Teensy controlled LED and Potentiometer
CSE 2100-001

Apar Pokhrel

November 5, 2019

Date Performed: October 22, 2019
Partners: Apar Pokhrel
Bivash Yadav

1 Objective and Description

Watch the video posted on YouTube for Lab-9.
Get a new git clone for cse2100 from github. This lab is associated with the files in the GUI_Teensy folder.

You will also need to build a Teensy based circuit with an RGB LED and a potentiometer (similar but not exact same circuit you have already built once). The schematic is in Figure 1 and the lab-9 video show the circuit built.

Your next task is to finish the Teensy code under Teensy_code/led_and_pot_serial; your code is supposed to go under "while (isRunning)". You may notice some changes in this code (e.g., to overcome an analog-write bug in the Teensy 3.2 micro-controller board's Arduino interface we increased the PWM resolution from 8 bits to 16 bits; this way you can actually mostly turn off the LEDs). You can also see that if you send a properly formatted packet with the first payload byte equaling the ASCII 'L', then the next three bytes are going to represent the brightness values for the red, the green, and the blue part of the LED respectively.
You will need to add code that obtains an ADC value from the potentiometer and if this value has changed since the last time the value was obtained then you need to assemble a properly formatted packet and send it off the serial port. The properly formatted packet is framed the same way we did in previous labs (e.g., by calling sendPacket()); the payload for this packet is three bytes as follows. The first payload byte should be the ASCII value for capital 'P' (signaling the other side that the incoming packet contains a reading from a potentiometer). The ADC conversion on the potentiometer results in a 10-bit number (between

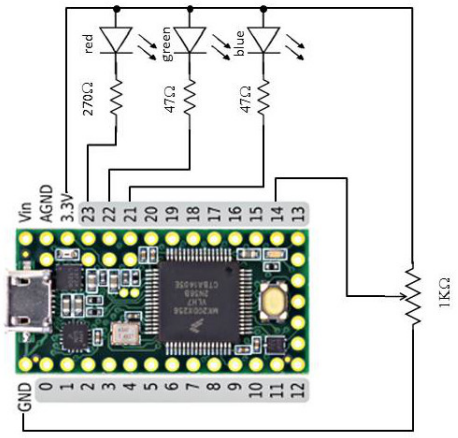

Figure 1: Lab 9 schematic

0 and 1023); you will need to separate this value into two bytes: the high-byte containing the upper (most significant) two bits and the low-byte containing the eight least significant bits. After the 'P' then you will first need to send the high-byte then the low byte (three bytes of payload all together).

You can test your code by running the TeensyControl executable from the Executable_LED_Controller directory. Make sure that the port description on top of this GUI is correct, open the port and try to control the LEDs and receive the potentiometer's value by turning it.
Show your success to the TA.

Your next task is to recreate the TeensyControl executable for which you can find a skeleton in LED_Controller. Complete the GUI, and the missing parts in main.cpp, global.h and serialthread.cpp.

The serialthread.cpp file contains the implementation of the thread that handles reading from the serial port (this is where you will need to implement creating packages from received bytes, validating the packets, and checking if there is an incoming 'P'-command; hint: you can find implementations for most of these in the Teensy code...).
Once you are done, demonstrate your work to the TA starting with a clean build of your project. Turn in both this document as well as your source code (cleaned of object files and executables).

1.1 Definitions and Quick Questions

thread: An independent path of execution within a process, running concurrently with other threads within a shared memory space

mutex: A mutual exclusion object that prevents two properly written threads from concurrently accessing a critical resource

What's the difference between the "Fill Level" attribute of a Scale widget and its associated ad
The fill level determines the maximum value the user is allowed to drag the scale widget to.Its associated maximum value is the the maximum accepted.

How do you start a thread in GLib and how do you find its id? The command g-thread-new() creates the thread and is identified using g-thread-self().