Philip Geramian
CSE-671 S2016
Lab #1 Report

In the first lab of CSE-671 we are tasked with doing some basic control of outputs using the Beaglebone Black embedded system. For the first task we are required to control the onboard LEDs in such a way that we can blink one of them at different rates. The second task is to do the same, but with an external LED, which requires the use and control of one of the Beaglebone's GPIO pins.
For this lab I took the time to write two simple classes in C++, one for the onboard LEDs, and the other a more general GPIO class that covers the important functionality. Both classes can be found at the end of this report in the appendix.
For the onboard LEDs I chose to use USR1 due to the fact that the operating system did not seem to be using this one, while the other three seemed to be in use by the OS. For the external LED I connected a LED between GPIO 60 (also known by GPIO1_28) P9.12, and ground on P9.2. In each class I included a macro protected test main that preformed the functionality of this lab by setting the LED on and off at various intervals.
For this lab my thought process was to make it such that material used here could be applied further into the semester if needed, thus why I made classes for this easy file I/O, so that at a later date I may use these classes in other projects as needed without having to rewrite them. After I wrote the outline of the class I was able to use the "usleep()" function to delay the changing of the output between a 1 and a 0. This was followed by another usleep() after that to keep the system at a zero for the same time. This was then all wrapped in a "for()" loop to repeat it several times to show the blinking. The only major difference between my code for the external LED was that my class allows for the direction to be changed, so I had to set the GPIO direction to "out" such that I could use it to control the LED.
Overall this lab was fairly simple, both in part do to the instructions provided in the lab assignment and my prior experience with the Beaglebone Black from both my Undergraduate career and through my TA position within the College. For myself I found it a good personal excuse to rewrite my GPIO class that I had written back in 2014, and was in dire need of replacement. For someone who is not familiar with either the Beaglebone Black, Linux, or embedded systems as a whole I could imagine how this lab could be difficult to some extent and thus see why a task that seems so simple was chosen as the starting point for the semester.

Appendix:

GPIO.h

```
#ifndef BBPDGGPIO_H
#define BBPDGGPIO_H

#include <string>
class GPIO
{
	public:
		GPIO();
		GPIO(int pin);
		void Set(int value);
		int Get();
		void Mode(std::string mode);
		~GPIO();
	private:
		int _pin;
		int _mode;
		int _value;
};
#endif
```

GPIO.cpp

```
#include <iostream>
#include <cstdlib>
#include <unistd.h>
#include <sys/stat.h>
#include <fstream>
#include <string>
#include "GPIO.h"

GPIO::GPIO(){}

GPIO::GPIO(int pin)
{
	if (getuid()!=0)
	{
		std::cout<<"Please rerun as root!\n";
		exit(1);
	}
	_pin=pin;
	std::fstream in;
	in.open("/sys/class/gpio/export",std::ios::out);
	if (!in.good())
	{
		std::cout << "can't open file /sys/class/gpio/export\n";
	}
	in<<pin;
	in.close();

}

GPIO::~GPIO()
{
	if (getuid()!=0)
	{
		std::cout<<"Please rerun as root!\n";
		exit(1);
	}
	std::fstream in;
	in.open("/sys/class/gpio/unexport",std::ios::out);
	if (!in.good())
	{
		std::cout << "can't open file /sys/class/gpio/unexport\n";
	}
```

```
	in<<_pin;
	in.close();

}

void GPIO::Set(int value)
{
	char val[50];
	_value=value;
	sprintf(val,"/sys/class/gpio/gpio%d/value",_pin);
	std::fstream in(val,std::fstream::out);
	if (!in.good())
	{
		std::cout << "can't open file " << val << "\n";
	}
	in<<value;
	in.close();
}
int GPIO::Get()
{
	char val[50];
   sprintf(val,"/sys/class/gpio/gpio%d/value",_pin);
	std::fstream in(val,std::fstream::in);
	if (!in.good())
	{
	std::cout << "can't open file " << val << "\n";
	return 1;
	}
	in>>_value;
	in.close();
	return _value;
}
void GPIO::Mode(std::string mode)
{
	char direc[50];
	sprintf(direc,"/sys/class/gpio/gpio%d/direction",_pin);
	std::fstream in(direc,std::fstream::out);
	if (!in.good())
	{
		std::cout << "can't open file " << direc<< "\n";
		return;
	}
	for (size_t i = 0; i < mode.length(); i++)
		mode[i] = tolower(mode[i]);
	in<<mode;
	in.close();
	return;
}
#ifdef TEST_GPIO
int main()
{
	GPIO LED(60);
	LED.Mode("OUT");
	for(int i=0;i<10;i++)
	{
		LED.Set(1);
		usleep(250000);
		LED.Set(0);
		usleep(250000);
	}
	for(int i=0;i<20;i++)
	{
		LED.Set(1);
		usleep(50000);
		LED.Set(0);
		usleep(50000);
	}
	return 0;
}
```

```
#endif
```

LEDS.h

```
#ifndef BBPDGLED_H
#define BBPDGLED_H

#include <string>
class LEDS
{
	public:
		LEDS();
		LEDS(int pin);
		void Set(int value);
		~LEDS();
	private:
		int _pin;
};
#endif
```

LEDS.cpp

```
#include <iostream>
#include <cstdlib>
#include <unistd.h>
#include <sys/stat.h>
#include <fstream>
#include <string>
#include "LEDS.h"

LEDS::LEDS(){}

LEDS::LEDS(int pin)
{
	if (getuid()!=0)
	{
		std::cout<<"Please rerun as root!\n";
		exit(1);
	}
	if(pin<0|| pin>4)
	{
		std::cout<<"value out of range!\n";
		exit(1);
	}
	_pin=pin;
}

LEDS::~LEDS(){}
void LEDS::Set(int value)
{
	char val[50];
	sprintf(val,"/sys/class/leds/beaglebone:green:usr%d/brightness",_pin);
	std::fstream in(val,std::fstream::out);
	if (!in.good())
	{
		std::cout << "can't open file " << val << "\n";
	}
	in<<value;
	in.close();
}
#ifdef TEST_LEDS
int main()
{
	LEDS LED(1);
	for(int i=0;i<10;i++)
	{
		LED.Set(1);
		usleep(250000);
		LED.Set(0);
```

```
                usleep(250000);
        }
        for(int i=0;i<20;i++)
        {
                LED.Set(1);
                usleep(50000);
                LED.Set(0);
                usleep(50000);
        }
        return 0;
}
#endif
```