MEEN 408:
Introduction to Robotics

Lab Manual

Table 0.0.1: Revision History

Version	Date	Author	Reviewed	Details
1.0	11/24/2016	AJE	—	Document Creation

Table of Contents

List of Figures

List of Tables

Table 0.0.2: Acronyms and Abbreviations

Acronym or Abbreviation	**Meaning**
ROS	Robot Operating System
SSH	Secure SHell

A Note to The Reader

Digital Document

This document was prepared in LaTeX– a dynamic document processor. As such, the document is intended for digital use. Throughout, there are hyperlinks, highlighted in blue, like this.

1 Lab 1: ROS Installation and C++

2 Lab 2: Hello World with ROS

3 Lab 3: Topic Example

4 Lab 4: Control of Built-In LEDS

```
#include <stdio.h>
#include <unistd.h>
#include <iostream>
using namespace std;

int main() {
  cout << "LED Flash Start" << endl;

  FILE* LEDHandle = NULL;
  const char* LEDBrightness =
      "/sys/class/leds/beaglebone:green:usr3/brightness";
  for (int i = 0; i < 10; i++) {
    cout << "on " << i << endl;
    if ((LEDHandle = fopen(LEDBrightness, "r+")) != NULL) {
      fwrite("1", sizeof(char), 1, LEDHandle);
      fclose(LEDHandle);
    }
    sleep(1);
    cout << "off " << i << endl;
    if ((LEDHandle = fopen(LEDBrightness, "r+")) != NULL) {
      fwrite("0", sizeof(char), 1, LEDHandle);
      fclose(LEDHandle);
    }
    sleep(1);
  }
  cout << "LED Flash End" << endl;
  return 0;
}
```

5 Lab 5: GPIO and using C++

```cpp
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <iostream>
#define MAX 64
using namespace std;

int flashGPIOLED(int, int);

int main() {
  // readButton(117)
  flashGPIOLED(60, 5);
  return 0;
}

int flashGPIOLED(int GPIOPin, int times) {
  cout << "GPIO LED Flash Pin: " << GPIOPin << " start." << endl;

  // Create Strings that point to the GPIO Pin Value and Direction files
  FILE* LEDHandle = NULL;
  char setValue[4];
  char GPIOString[4];
  char GPIOValue[MAX];
  char GPIODirection[MAX];
  sprintf(GPIOString, "%d", GPIOPin);
  sprintf(GPIOValue, "/sys/class/gpio/gpio%d/value", GPIOPin);
  sprintf(GPIODirection, "/sys/class/gpio/gpio%d/direction", GPIOPin);

  // Export the Pin Number (this will make the Pin directory we can then
     use)
  // First we see if it is possible to create the directory. If not, quit.
  if ((LEDHandle = fopen("/sys/class/gpio/export", "ab")) ==
      NULL) {  // note that this opens the file
    printf("Cannot export the GPIO Pin");
    return 1;
  }
  // If the above works, export the pin:
  strcpy(setValue, GPIOString);                    // copy pin number to
     setValue
  fwrite(&setValue, sizeof(char), 2, LEDHandle);  // write set Value to
                                                  // LEDHANDLE
  fclose(LEDHandle);                              // and close the file

  // set pin direction
  if ((LEDHandle = fopen(GPIODirection, "rb+")) == NULL) {
    printf("Cannot open direction handle. \n");
    return 1;
  }
  strcpy(setValue, "out");
  fwrite(&setValue, sizeof(char), 3, LEDHandle);
  fclose(LEDHandle);

```

```
51    // flash the led on the gpio pin
52    for (int i = 0; i < times * 2; i++) {
53      if ((LEDHandle = fopen(GPIOValue, "rb+")) == NULL) {
54        printf("Cannot open value handle. \n");
55        return 1;
56      }
57      cout << &LEDHandle << endl;  // this just prints out the file pointer
            value.
58                                   // not sure why this is here
59      if (i % 2 == 1) {
60        strcpy(setValue, "0");
61      } else {
62        strcpy(setValue, "1");
63      }
64      fwrite(&setValue, sizeof(char), 1, LEDHandle);
65      fclose(LEDHandle);
66      sleep(1);
67    }
68
69    if ((LEDHandle = fopen("/sys/class/gpio/unexport", "ab")) == NULL) {
70      printf("Cannot unexport GPIO Pin.\n");
71      return 1;
72    }
73    strcpy(setValue, GPIOString);
74    fwrite(&setValue, sizeof(char), 2, LEDHandle);
75    fclose(LEDHandle);
76
77    cout << "GPIO LED Flash PIN: " << GPIOPin << " end" << endl;
78    return 0;
79  }
```

6 Lab 6: ADC and C++

```cpp
#include <stdio.h>
#include <unistd.h>
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

int main() {
  cout << "ADC Start" << endl;
  FILE* ADCHandler = NULL;
  const char* ADCVoltage = "/sys/bus/iio/devices/iio:device0/
     in_voltage5_raw";

  char ADCVoltageRead[5] = {0};
  int Voltage;

  while (1) {
    if ((ADCHandler = fopen(ADCVoltage, "r")) != NULL) {
      fread(ADCVoltageRead, sizeof(char), sizeof(ADCVoltageRead - 1),
            ADCHandler);
      fclose(ADCHandler);
      stringstream ss(ADCVoltageRead);
      ss >> Voltage;
      cout << Voltage << endl;
      //        printf("%s", ADCVoltageRead);
      usleep(50000);
    }
  }
  cout << "ADC End" << endl;
  return 0;
}
```

7 Lab 7: PWM and C++

```cpp
#include <stdio.h>
#include <unistd.h>
#include <iostream>
using namespace std;

#define PERIOD 1000000

int main() {
  cout << "PWM Start" << endl;

  FILE* PWMHandle = NULL;
  const char* PWMPeriod = "/sys/class/pwm/pwmchip0/pwm0/period";
  const char* PWMDutyCycle = "/sys/class/pwm/pwmchip0/pwm0/duty_cycle";
  const char* PWMEnable = "/sys/class/pwm/pwmchip0/pwm0/enable";
  char setValue[10];

  // Set PWM period, duty cycle, and enabled status
  if ((PWMHandle = fopen(PWMPeriod, "r+")) != NULL) {
    fwrite("1000000", sizeof(char), 7, PWMHandle);
    fclose(PWMHandle);
  }
  if ((PWMHandle = fopen(PWMDutyCycle, "r+")) != NULL) {
    fwrite("0", sizeof(char), 1, PWMHandle);
    fclose(PWMHandle);
    // cout << "DutyCycle " << sizeof(PERIOD/2)/sizeof(char) << endl;
  }
  if ((PWMHandle = fopen(PWMEnable, "r+")) != NULL) {
    fwrite("1", sizeof(char), 1, PWMHandle);
    fclose(PWMHandle);
  }

  // increase the duty cycle from 0% to 100% in 10 seconds smoothly
  double timeToFullLight = 10; //seconds
  int numberOfIncrements = 1000;
  for (int i = 0; i < numberOfIncrements; i++) {
    cout << "count " << i << endl;
    sprintf(setValue, "%d", int(PERIOD * i / numberOfIncrements));
    if ((PWMHandle = fopen(PWMDutyCycle, "r+")) != NULL) {
      fwrite(setValue, sizeof(char), sizeof(setValue), PWMHandle);
      fclose(PWMHandle);
    }
   usleep(int(timeToFullLight/numberOfIncrements*1000000));
  }

  // reset duty cycle to 0 and disable
  if ((PWMHandle = fopen(PWMDutyCycle, "r+")) != NULL) {
    fwrite("0", sizeof(char), 1, PWMHandle);
    fclose(PWMHandle);
    // cout << "DutyCycle " << sizeof(PERIOD/2)/sizeof(char) << endl;
  }
  if ((PWMHandle = fopen(PWMEnable, "r+")) != NULL) {
    fwrite("0", sizeof(char), 1, PWMHandle);
```

```
53      fclose(PWMHandle);
54    }
55
56    cout << "PWM End" << endl;
57    return 0;
58  }
```

8 Lab 8: Quadrature Encoder

8.1 eQEP Setup

8.2 eQEP with C++

```
content...
```

9 Lab 9: Putting it all Together—Making your own C++ Library

10 Lab 10: ROS Network Setting

```
content...
```

11 Lab 11: Servomotor

```
content...
```

12 Lab 12: DC Motor, BBB, and ROS

```
content...
```