Bert Zhao
EE 474

Abstract: This lab report provides an overview of lab 1 of EE/CSE 474. There is an introduction to the lab objective. There is a summary of the three tasks required in the lab including the procedure and result analysis of each task. There is also a conclusion and an overview of the skills learned in this lab.

Introduction: This is the first lab of EE 474 and it is focused on becoming familiar with the basic concepts of embedded programing and the IAR Workbench. We are also introduced to the TIVA TM4C board. The first part of the lab involves using the provided sample code to connect the TIVA to our PC. We learn how to download a simple blinking light program to the board. Next we use the on-board switches to control the on-board LEDs. Part two of this lab involves connecting peripherals to the TIVA board. We use buttons and LEDs to create a simple traffic light controller.

Procedure: In section A of this lab, learned how to connect the TIVA board to our PC. We covered the basics of how to enable the appropriate GPIO ports and pins and proper input/output configuration for the LEDs and on-board switches. We first created a program that blinked between different colors of lights and downloaded it to the board. There are red, green, and blue LEDs and the blinking lights included the eight different combinations of them. The second program in this section included the on-board switches to control the red and green LEDs. Switch one is used to control the red LED and switch two is used to control the green LED.

In section B of this lab, we learned how to interface peripherals to the TIVA board. We used two push buttons and green, yellow, and red LEDs to create a traffic light controller. The first button is used as a start/stop button. When the system first starts, this button is used to turn on the lights. The standard behavior of the lights is to just alternate between green and red. During normal operation, hitting the start/stop button will pause the light on the last lit light. The second button is the pedestrian crossing button. If this button is hit when the light is green, the next light will be yellow, and then hold at red until the button is released. If the current light is red, the pedestrian button will cause the red light to stay on until the button is released.

Results: The main result we wanted to achieve in section A of the lab was to connect the TIVA board to our PC and download a basic LED manipulation program. In the first part of section A, we created our own header files with definitions for the necessary registers for GPIO port F. In the second part of section A, we measured the corresponding pins to the on-board switches with an oscilloscope. The waveform for multiple presses of each button is show below.

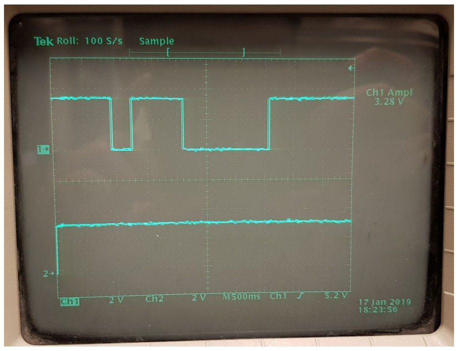


Figure 1: Oscilloscope waveform

Bert Zhao
EE 474

For the traffic controller, we created the circuit of two switches and three LEDs on a breadboard. The circuit diagram and picture of the complete circuit are shown below. I used GPIO port A and pins PA2 and PA3 were used for the start/stop and pedestrian crossing buttons respectively. PA5, PA6, and PA7 were used for the green, yellow, and red LEDs. To solve potential debouncing issues when using the push buttons, a delay was used following the detection of each potential button press.

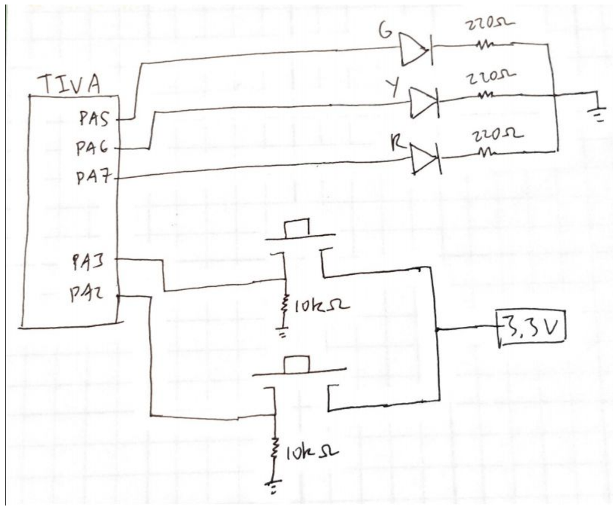


Figure 2: Circuit diagram

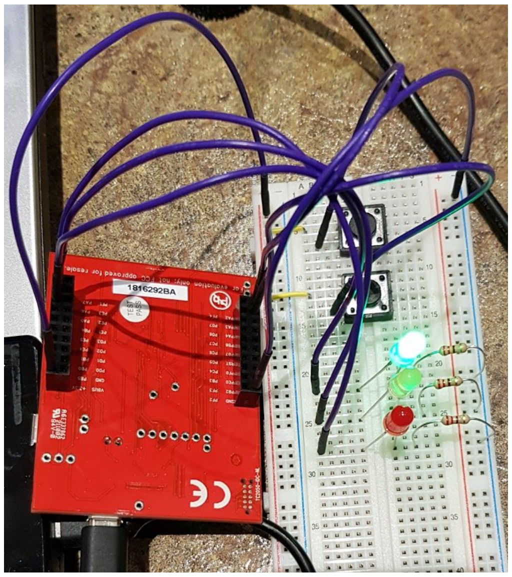
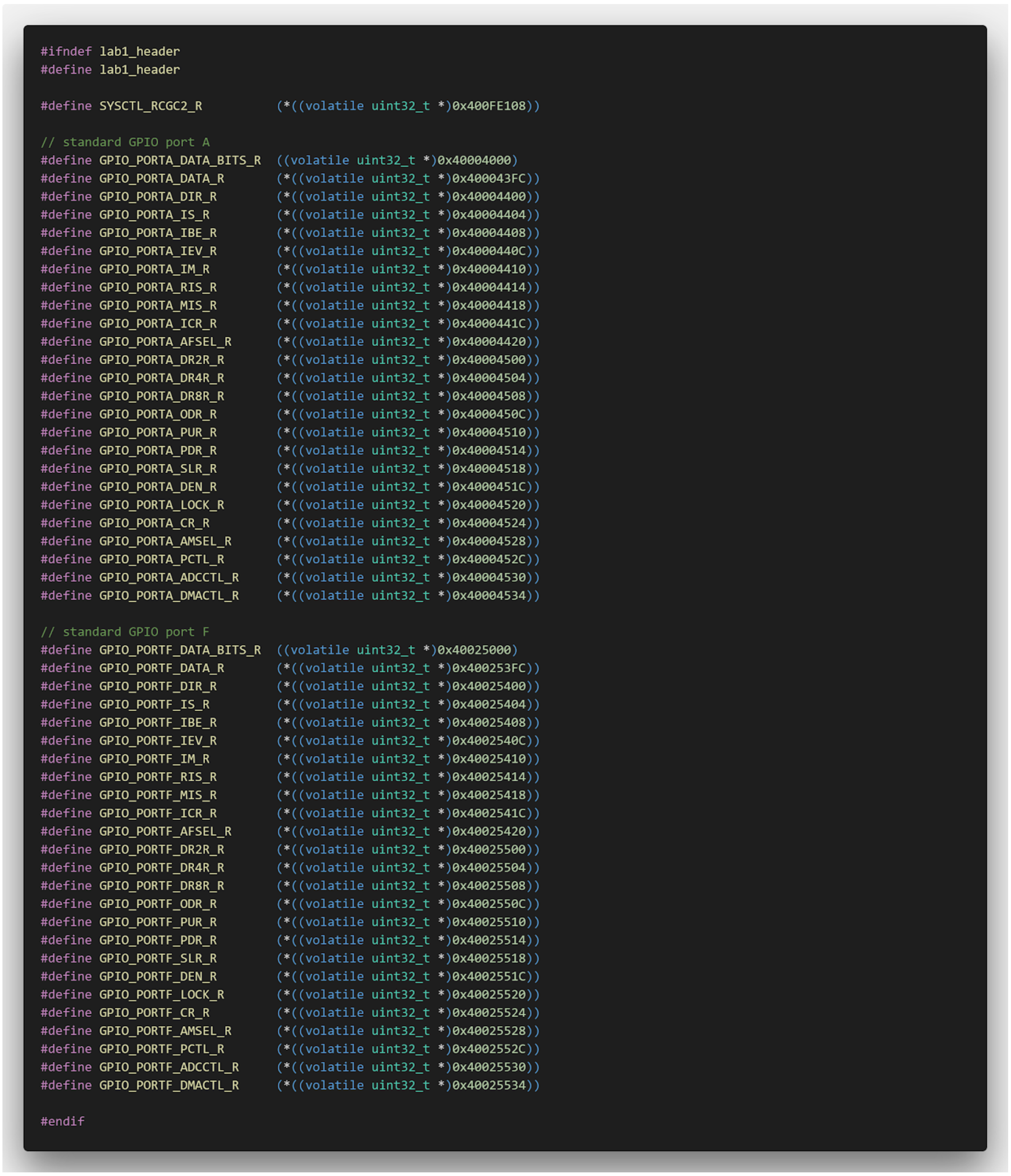


Figure 3: Picture of circuit

Conclusion: Overall, this lab provided a key overview and introduction to embedded programming. We started becoming more familiar with the TIVA TM4C board and its functions. We created three basic programs; blinking LEDs, on-board switch controlled LEDs, and a traffic light controller. We also learned more general skills for programming in C such as creating header files. The fundamental skills we learned in this lab will be important for upcoming projects in this course.

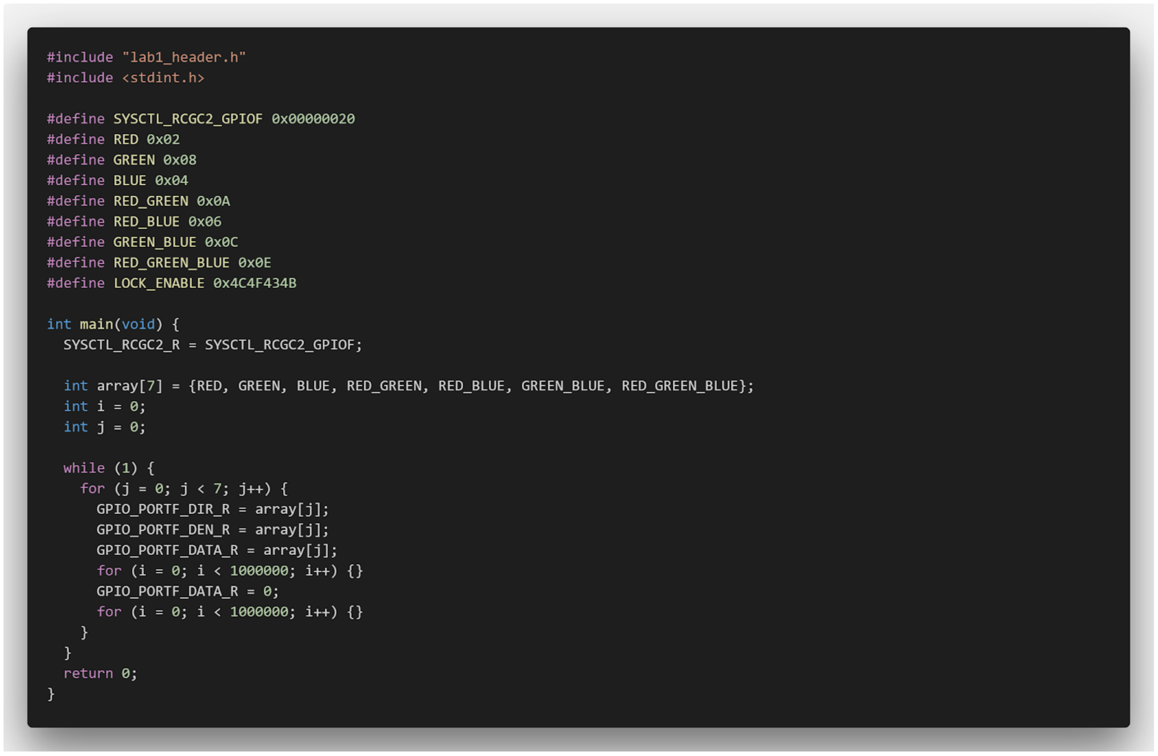
Bert Zhao
EE 474

Appendix:



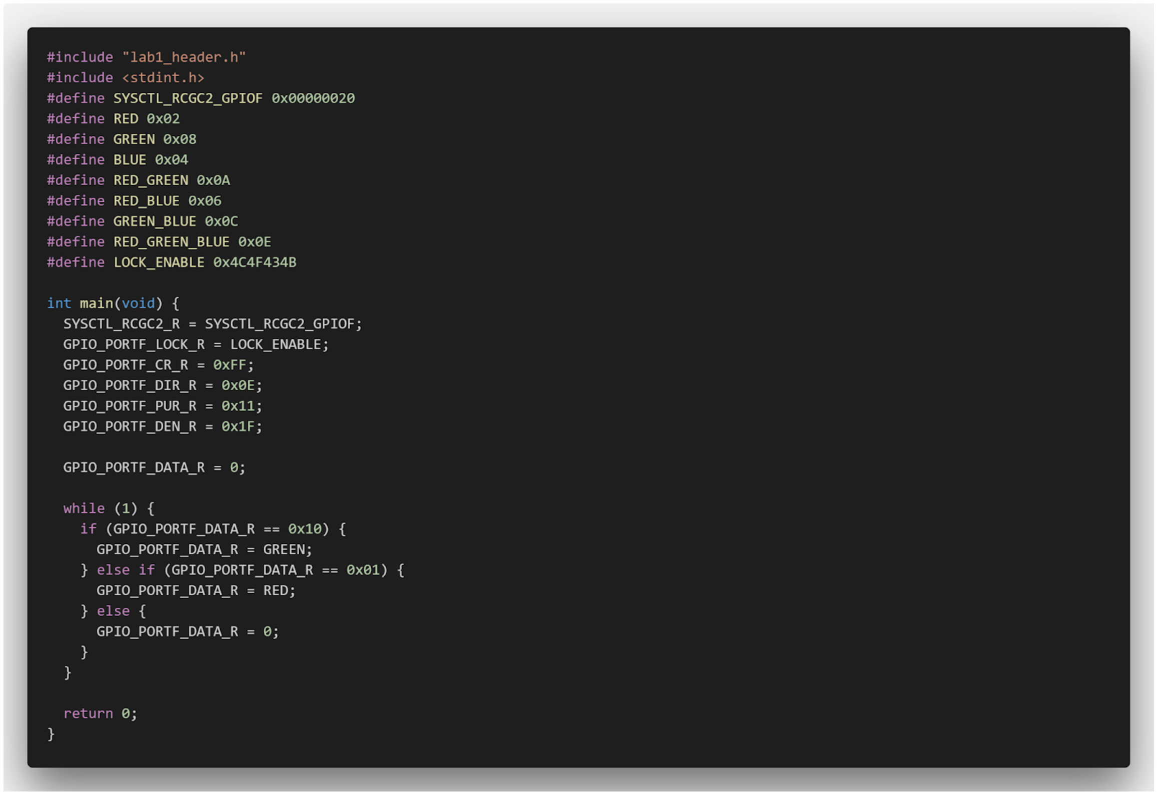
```
#ifndef lab1_header
#define lab1_header

#define SYSCTL_RCGC2_R          (*((volatile uint32_t *)0x400FE108))

// standard GPIO port A
#define GPIO_PORTA_DATA_BITS_R  ((volatile uint32_t *)0x40004000)
#define GPIO_PORTA_DATA_R       (*((volatile uint32_t *)0x400043FC))
#define GPIO_PORTA_DIR_R        (*((volatile uint32_t *)0x40004400))
#define GPIO_PORTA_IS_R         (*((volatile uint32_t *)0x40004404))
#define GPIO_PORTA_IBE_R        (*((volatile uint32_t *)0x40004408))
#define GPIO_PORTA_IEV_R        (*((volatile uint32_t *)0x4000440C))
#define GPIO_PORTA_IM_R         (*((volatile uint32_t *)0x40004410))
#define GPIO_PORTA_RIS_R        (*((volatile uint32_t *)0x40004414))
#define GPIO_PORTA_MIS_R        (*((volatile uint32_t *)0x40004418))
#define GPIO_PORTA_ICR_R        (*((volatile uint32_t *)0x4000441C))
#define GPIO_PORTA_AFSEL_R      (*((volatile uint32_t *)0x40004420))
#define GPIO_PORTA_DR2R_R       (*((volatile uint32_t *)0x40004500))
#define GPIO_PORTA_DR4R_R       (*((volatile uint32_t *)0x40004504))
#define GPIO_PORTA_DR8R_R       (*((volatile uint32_t *)0x40004508))
#define GPIO_PORTA_ODR_R        (*((volatile uint32_t *)0x4000450C))
#define GPIO_PORTA_PUR_R        (*((volatile uint32_t *)0x40004510))
#define GPIO_PORTA_PDR_R        (*((volatile uint32_t *)0x40004514))
#define GPIO_PORTA_SLR_R        (*((volatile uint32_t *)0x40004518))
#define GPIO_PORTA_DEN_R        (*((volatile uint32_t *)0x4000451C))
#define GPIO_PORTA_LOCK_R       (*((volatile uint32_t *)0x40004520))
#define GPIO_PORTA_CR_R         (*((volatile uint32_t *)0x40004524))
#define GPIO_PORTA_AMSEL_R      (*((volatile uint32_t *)0x40004528))
#define GPIO_PORTA_PCTL_R       (*((volatile uint32_t *)0x4000452C))
#define GPIO_PORTA_ADCCTL_R     (*((volatile uint32_t *)0x40004530))
#define GPIO_PORTA_DMACTL_R     (*((volatile uint32_t *)0x40004534))

// standard GPIO port F
#define GPIO_PORTF_DATA_BITS_R  ((volatile uint32_t *)0x40025000)
#define GPIO_PORTF_DATA_R       (*((volatile uint32_t *)0x400253FC))
#define GPIO_PORTF_DIR_R        (*((volatile uint32_t *)0x40025400))
#define GPIO_PORTF_IS_R         (*((volatile uint32_t *)0x40025404))
#define GPIO_PORTF_IBE_R        (*((volatile uint32_t *)0x40025408))
#define GPIO_PORTF_IEV_R        (*((volatile uint32_t *)0x4002540C))
#define GPIO_PORTF_IM_R         (*((volatile uint32_t *)0x40025410))
#define GPIO_PORTF_RIS_R        (*((volatile uint32_t *)0x40025414))
#define GPIO_PORTF_MIS_R        (*((volatile uint32_t *)0x40025418))
#define GPIO_PORTF_ICR_R        (*((volatile uint32_t *)0x4002541C))
#define GPIO_PORTF_AFSEL_R      (*((volatile uint32_t *)0x40025420))
#define GPIO_PORTF_DR2R_R       (*((volatile uint32_t *)0x40025500))
#define GPIO_PORTF_DR4R_R       (*((volatile uint32_t *)0x40025504))
#define GPIO_PORTF_DR8R_R       (*((volatile uint32_t *)0x40025508))
#define GPIO_PORTF_ODR_R        (*((volatile uint32_t *)0x4002550C))
#define GPIO_PORTF_PUR_R        (*((volatile uint32_t *)0x40025510))
#define GPIO_PORTF_PDR_R        (*((volatile uint32_t *)0x40025514))
#define GPIO_PORTF_SLR_R        (*((volatile uint32_t *)0x40025518))
#define GPIO_PORTF_DEN_R        (*((volatile uint32_t *)0x4002551C))
#define GPIO_PORTF_LOCK_R       (*((volatile uint32_t *)0x40025520))
#define GPIO_PORTF_CR_R         (*((volatile uint32_t *)0x40025524))
#define GPIO_PORTF_AMSEL_R      (*((volatile uint32_t *)0x40025528))
#define GPIO_PORTF_PCTL_R       (*((volatile uint32_t *)0x4002552C))
#define GPIO_PORTF_ADCCTL_R     (*((volatile uint32_t *)0x40025530))
#define GPIO_PORTF_DMACTL_R     (*((volatile uint32_t *)0x40025534))

#endif
```

Code 1: lab1_header.h



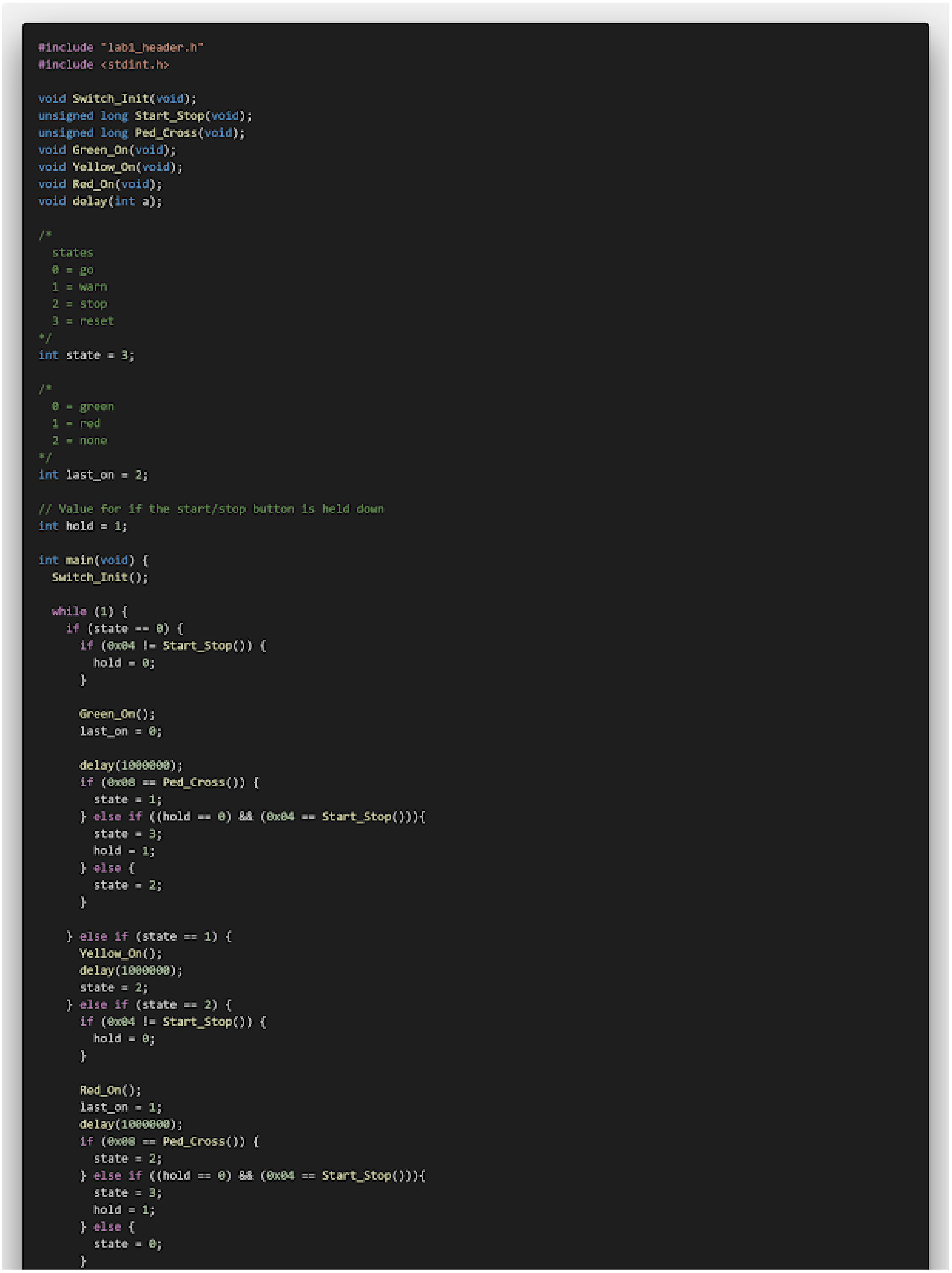
```c
#include "lab1_header.h"
#include <stdint.h>

#define SYSCTL_RCGC2_GPIOF 0x00000020
#define RED 0x02
#define GREEN 0x08
#define BLUE 0x04
#define RED_GREEN 0x0A
#define RED_BLUE 0x06
#define GREEN_BLUE 0x0C
#define RED_GREEN_BLUE 0x0E
#define LOCK_ENABLE 0x4C4F434B

int main(void) {
  SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

  int array[7] = {RED, GREEN, BLUE, RED_GREEN, RED_BLUE, GREEN_BLUE, RED_GREEN_BLUE};
  int i = 0;
  int j = 0;

  while (1) {
    for (j = 0; j < 7; j++) {
      GPIO_PORTF_DIR_R = array[j];
      GPIO_PORTF_DEN_R = array[j];
      GPIO_PORTF_DATA_R = array[j];
      for (i = 0; i < 1000000; i++) {}
      GPIO_PORTF_DATA_R = 0;
      for (i = 0; i < 1000000; i++) {}
    }
  }
  return 0;
}
```

Code 2: Section A Task 1

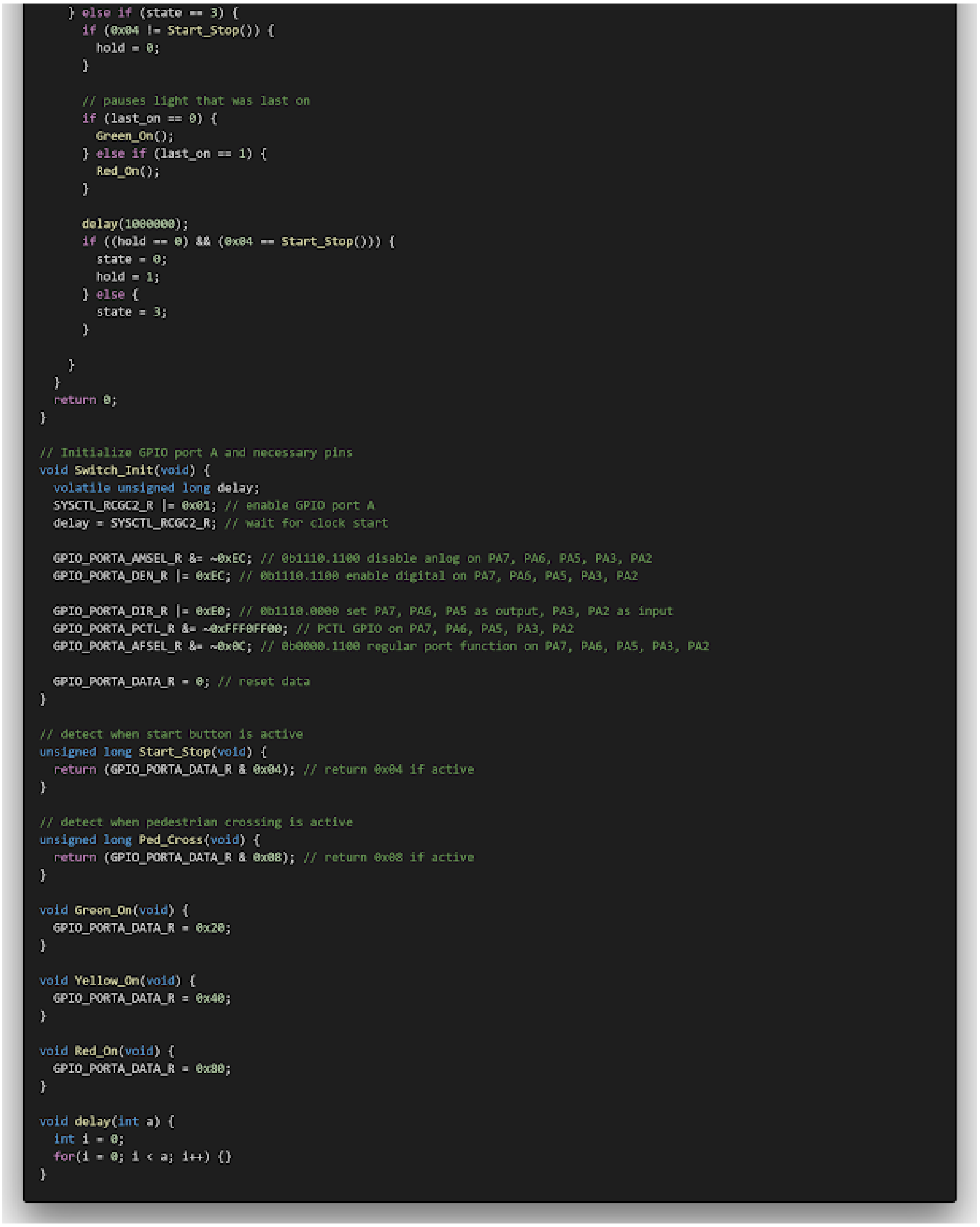


```c
#include "lab1_header.h"
#include <stdint.h>
#define SYSCTL_RCGC2_GPIOF 0x00000020
#define RED 0x02
#define GREEN 0x08
#define BLUE 0x04
#define RED_GREEN 0x0A
#define RED_BLUE 0x06
#define GREEN_BLUE 0x0C
#define RED_GREEN_BLUE 0x0E
#define LOCK_ENABLE 0x4C4F434B

int main(void) {
  SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
  GPIO_PORTF_LOCK_R = LOCK_ENABLE;
  GPIO_PORTF_CR_R = 0xFF;
  GPIO_PORTF_DIR_R = 0x0E;
  GPIO_PORTF_PUR_R = 0x11;
  GPIO_PORTF_DEN_R = 0x1F;

  GPIO_PORTF_DATA_R = 0;

  while (1) {
    if (GPIO_PORTF_DATA_R == 0x10) {
      GPIO_PORTF_DATA_R = GREEN;
    } else if (GPIO_PORTF_DATA_R == 0x01) {
      GPIO_PORTF_DATA_R = RED;
    } else {
      GPIO_PORTF_DATA_R = 0;
    }
  }

  return 0;
}
```

Code 3: Section A Task 2



```c
#include "lab1_header.h"
#include <stdint.h>

void Switch_Init(void);
unsigned long Start_Stop(void);
unsigned long Ped_Cross(void);
void Green_On(void);
void Yellow_On(void);
void Red_On(void);
void delay(int a);

/*
  states
  0 = go
  1 = warn
  2 = stop
  3 = reset
*/
int state = 3;

/*
  0 = green
  1 = red
  2 = none
*/
int last_on = 2;

// Value for if the start/stop button is held down
int hold = 1;

int main(void) {
  Switch_Init();

  while (1) {
    if (state == 0) {
      if (0x04 != Start_Stop()) {
        hold = 0;
      }

      Green_On();
      last_on = 0;

      delay(1000000);
      if (0x08 == Ped_Cross()) {
        state = 1;
      } else if ((hold == 0) && (0x04 == Start_Stop())){
        state = 3;
        hold = 1;
      } else {
        state = 2;
      }

    } else if (state == 1) {
      Yellow_On();
      delay(1000000);
      state = 2;
    } else if (state == 2) {
      if (0x04 != Start_Stop()) {
        hold = 0;
      }

      Red_On();
      last_on = 1;
      delay(1000000);
      if (0x08 == Ped_Cross()) {
        state = 2;
      } else if ((hold == 0) && (0x04 == Start_Stop())){
        state = 3;
        hold = 1;
      } else {
        state = 0;
      }
```



```c
    } else if (state == 3) {
      if (0x04 != Start_Stop()) {
        hold = 0;
      }

      // pauses light that was last on
      if (last_on == 0) {
        Green_On();
      } else if (last_on == 1) {
        Red_On();
      }

      delay(1000000);
      if ((hold == 0) && (0x04 == Start_Stop())) {
        state = 0;
        hold = 1;
      } else {
        state = 3;
      }

    }
  }
  return 0;
}

// Initialize GPIO port A and necessary pins
void Switch_Init(void) {
  volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x01; // enable GPIO port A
  delay = SYSCTL_RCGC2_R; // wait for clock start

  GPIO_PORTA_AMSEL_R &= ~0xEC; // 0b1110.1100 disable anlog on PA7, PA6, PA5, PA3, PA2
  GPIO_PORTA_DEN_R |= 0xEC; // 0b1110.1100 enable digital on PA7, PA6, PA5, PA3, PA2

  GPIO_PORTA_DIR_R |= 0xE0; // 0b1110.0000 set PA7, PA6, PA5 as output, PA3, PA2 as input
  GPIO_PORTA_PCTL_R &= ~0xFFF0FF00; // PCTL GPIO on PA7, PA6, PA5, PA3, PA2
  GPIO_PORTA_AFSEL_R &= ~0x0C; // 0b0000.1100 regular port function on PA7, PA6, PA5, PA3, PA2

  GPIO_PORTA_DATA_R = 0; // reset data
}

// detect when start button is active
unsigned long Start_Stop(void) {
  return (GPIO_PORTA_DATA_R & 0x04); // return 0x04 if active
}

// detect when pedestrian crossing is active
unsigned long Ped_Cross(void) {
  return (GPIO_PORTA_DATA_R & 0x08); // return 0x08 if active
}

void Green_On(void) {
  GPIO_PORTA_DATA_R = 0x20;
}

void Yellow_On(void) {
  GPIO_PORTA_DATA_R = 0x40;
}

void Red_On(void) {
  GPIO_PORTA_DATA_R = 0x80;
}

void delay(int a) {
  int i = 0;
  for(i = 0; i < a; i++) {}
}
```

Code 4: Section B Task 1