lab 4
Urban Traffic Light in Assembly

EGc332– Microcontrollers

March 8, 2021

Christopher Lepore

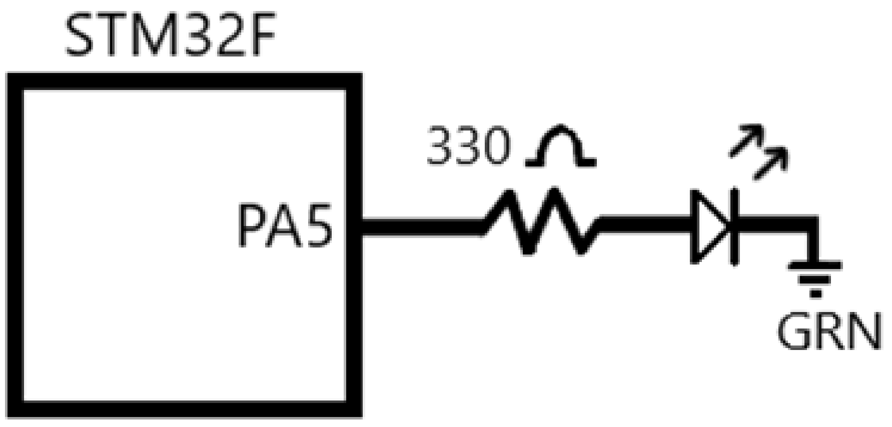
Introduction

In this lab we designed an urban traffic light control system in assembly for a Nucleo-64 microcontroller and used an assembly blinker program as a base. This introduced us to setting up a STM32F446RE circuit board using assembly code. This lab also introduced managing the microcontrollers registers and using assembly commands to assign port pins. The Blinker program simply cycles through turning an LED on and off with a second delay in between. The traffic light program cycles though green, yellow, and red LEDs for the north/south(NS) and east/west(EW) roads like a normal rural traffic light. In this cycle the green light stage lasts for 5 seconds and the yellow light stage lasts for 2 seconds. Since this traffic light is in an urban area there are no road sensors, therefore the code will loop back to the beginning (L1) and cycle forever. By using the Blinker program to design an urban traffic light system in assembly, a deeper comprehension of assembly coding with the STM32F446RE circuit board was achieved.

Design

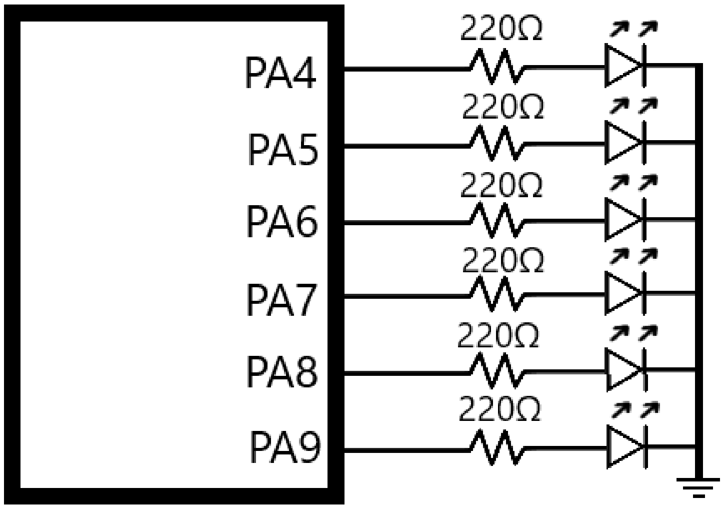
Hardware (Block Diagrams)

Blinker



For the Blinker circuit, port PA5 is set to output mode and then connected to a light emitting diode (LED). There is a 330-ohm resistor in between them to prevent the LED from burning out.

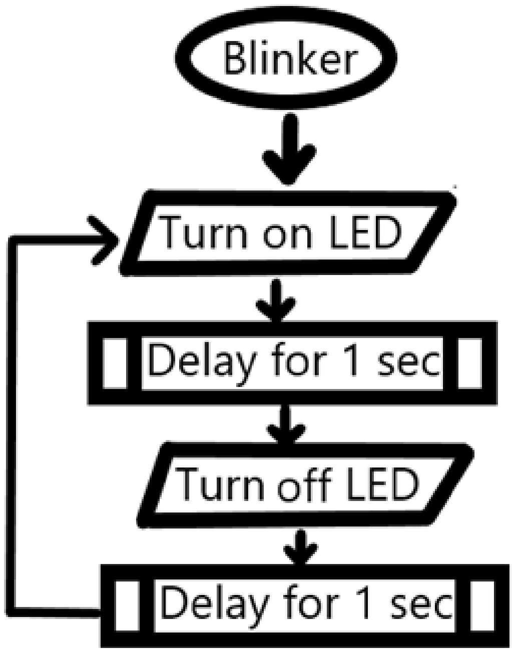
Urban traffic light



For the traffic light circuit, port PA4 to PA9 were set to output mode and then each connected to a light emitting diode (LED). There are 220-ohm resistors in between them to prevent the LED from burning out.

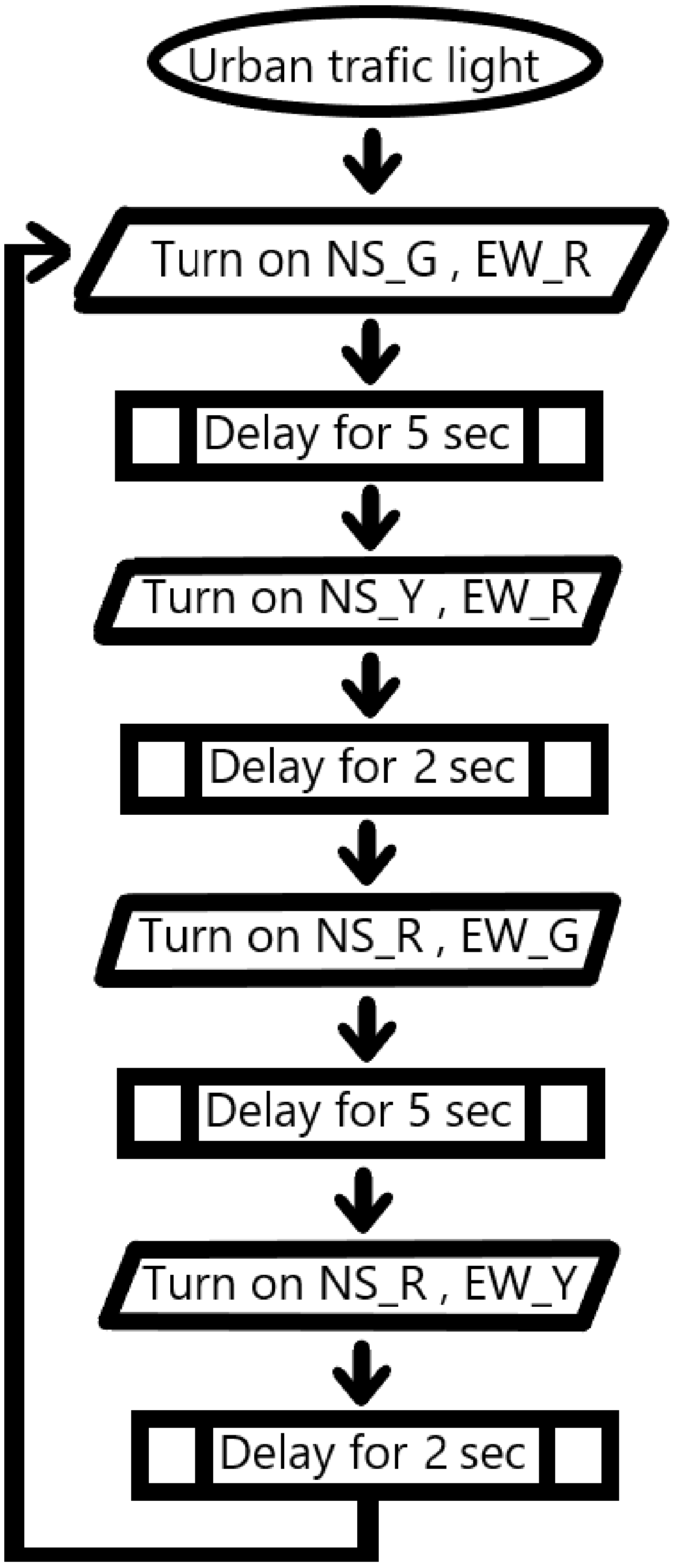
Software (Flow Charts)

Blinker



In the Blinker program, the code will cycle from turning the LED on and off with a 1 second delay in between each transition. Since there is no function block the cycle will go on forever without changing

Urban traffic light

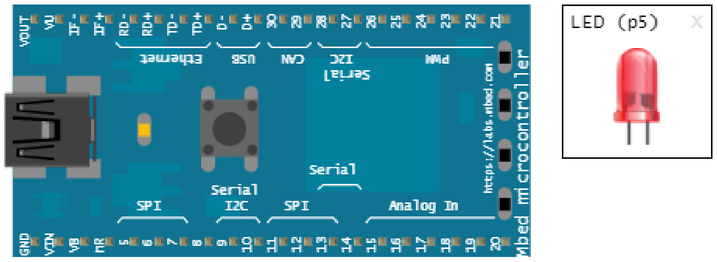


In the urban traffic light program, the code will cycle just like a normal traffic light, but the green LEDs duration are reduced to 5 seconds and the yellow LEDs duration are reduced to 2 seconds. Since there is no function block the cycle will go on forever without changing.

Verification

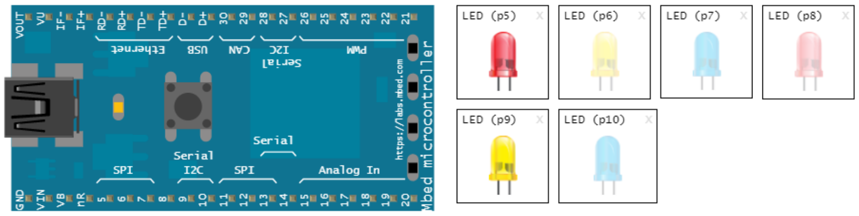
MBED simulation

Blinker



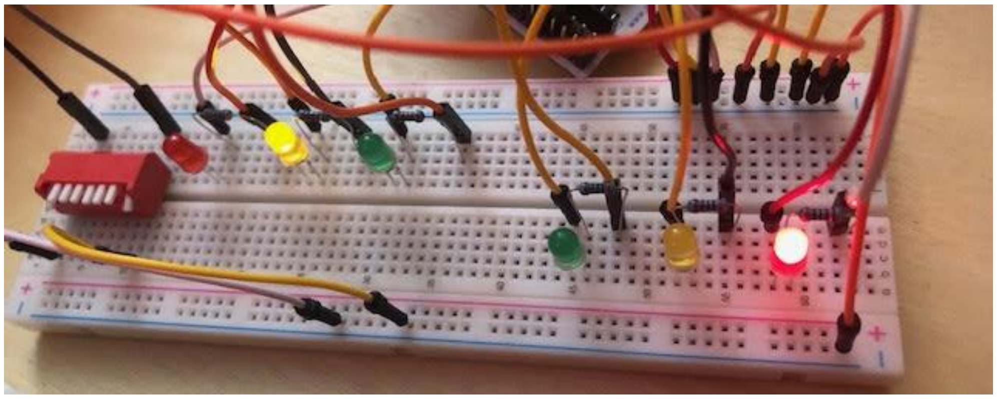
MBED simulation circuit

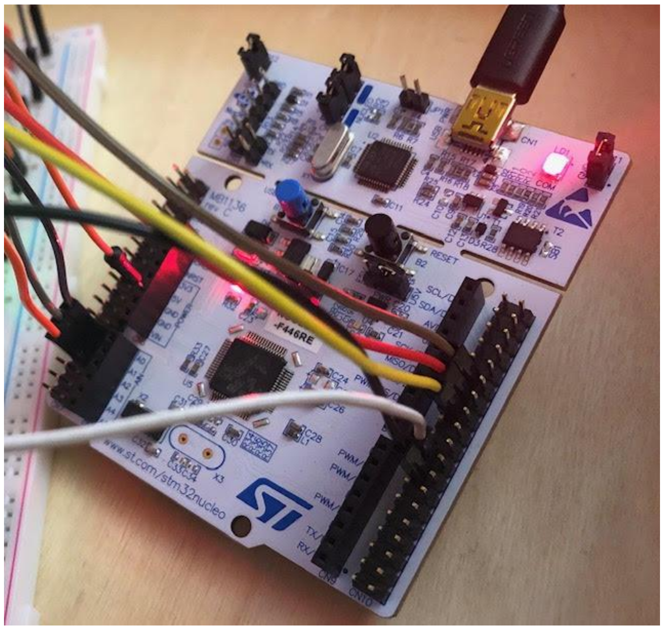
Urban traffic light



MBED simulation circuit

Nucleo and breadboard circuit


Breadboard wiring


NUCLEO circuit board

Summary and conclusion

In the Blinker program a LED is turned on and off every second. This is done in two phases. Phase one is the prep for the program which involves enabling the GPIOA clock, setting PA5 to output mode, and setting up the reset handler and exports. Phase two is the meat of the program begins at the beginning of the L1 loop. This phase states that port PA5 should alternate from on to off. This is done by calling a delay function in between setting the port on or off. One of the problems encountered was when the paths for the code got messed up. This was solved by starting a new project from scratch.

In the urban traffic light program, port PA4 to PA9 will be wired to NSG, NSY, NSR, EWG, EWY, and EWR, respectively. The traffic light program was created by taking the blinker program and changing the L1 loop and the output values to make a traffic light cycle. The traffic light cycles through 4 stage like the Blinker program's 2 stages(on/off). Stage 1 has NSG and EWR LEDs on, stage 2 turns NSY LED and EWR on, stage 3 turns NSR and EWG on, and stage 4 turns NER and EWY on. The code will cycle infinitely and have a delay of 5 seconds in between stage 1 to 2 and 3 to 4. It will also have a 2 second delay in between stage 2 to 3 and 4 to 1.

This introduced us to setting up a STM32F446RE circuit board using assembly code, managing the microcontrollers registers, and using assembly commands to assign port pins. This was done by converting an assembly Blinker program into an urban traffic light and experiencing assembly code for the first time. Due to the lessons above this lab has prepared students to begin designing and coding more complex projects in assembly language.

Appendix (Programs)

Blinker:

```
        EXPORT  __Vectors
        EXPORT  Reset_Handler
        AREA    vectors, CODE, READONLY

__Vectors   DCD     0x10010000  ; 0x20008000    ; Top of Stack
            DCD     Reset_Handler               ; Reset Handler

RCC_AHB1ENR equ 0x40023830
GPIOA_MODER equ 0x40020000
GPIOA_ODR   equ 0x40020014

        AREA    PROG, CODE, READONLY
Reset_Handler
        ldr     r4, =RCC_AHB1ENR    ; enable GPIOA clock
        mov     r5, #1
        str     r5, [r4]

        ldr     r4, =GPIOA_MODER
        ldr     r5, =0x00000400     ; set PA5 as output
        str     r5, [r4]

L1      ldr     r4, =GPIOA_ODR
        mov     r5, #0x20           ; turn on LED
        str     r5, [r4]

         mov    r0, #1000
        bl      delay

        ldr     r4, =GPIOA_ODR
        mov     r5, #0x00           ; turn off LED
        str     r5, [r4]

        mov     r0, #1000
        bl      delay
        b       L1                  ; loop forever
```

```
; delay milliseconds in R0
delay       ldr     r1, =5325
DL1         subs    r1, r1, #1
            bne     DL1
            subs    r0, r0, #1
            bne     delay
        bx      lr

        end
```

Urban Traffic Light:

```
        EXPORT  __Vectors
        EXPORT  Reset_Handler
        AREA    vectors, CODE, READONLY

__Vectors   DCD     0x10010000  ; 0x20008000    ; Top of Stack
        DCD     Reset_Handler               ; Reset Handler

RCC_AHB1ENR equ 0x40023830
GPIOA_MODER equ 0x40020000
GPIOA_ODR   equ 0x40020014

        AREA    PROG, CODE, READONLY
Reset_Handler
        ldr     r4, =RCC_AHB1ENR    ; enable GPIOA clock
         mov         r5, #1
        str     r5, [r4]

        ldr     r4, =GPIOA_MODER
        ldr     r5, =0x00055500             ; set pin A4 - A9 to output mode
        str     r5, [r4]

L1      ldr     r4, =GPIOA_ODR
        mov     r5, #0x00000210             ; turn on NSG/EWR
        str     r5, [r4]

         mov     r0, #5000
        bl      delay

        ldr     r4, =GPIOA_ODR
        mov     r5, #0x00000220             ; turn on NSY/EWR
        str     r5, [r4]

         mov     r0, #2000
        bl      delay

         ldr     r4, =GPIOA_ODR
        mov     r5, #0x000000C0             ; turn on NSR/EWG
        str     r5, [r4]
```

```
         mov     r0, #5000
        bl      delay

        ldr     r4, =GPIOA_ODR
        mov     r5, #0x00000140             ; turn on NSR/EWY
        str     r4, [r4]

         mov     r0, #2000
        bl      delay

        b       L1                 ; loop forever

; delay milliseconds in R0
delay       ldr     r1, =5325
DL1         subs    r1, r1, #1
             bne     DL1
            subs    r0, r0, #1
            bne     delay
            bx      lr

            end
```

Embedded system - Blinker:

```
#include "mbed.h"

DigitalOut led(p5);

int main() {
    while (1) {
        led = !led;

        wait_ms(500);
    }
}
```

Embedded system - Urban traffic light:

```
#include "mbed.h"

DigitalOut NSR(p5);
DigitalOut NSY(p6);
DigitalOut NSG(p7);
DigitalOut EWR(p8);
DigitalOut EWY(p9);
DigitalOut EWG(p10);
```

```
DigitalIn NS_sensor(p11);
DigitalIn EW_sensor(p12);

int main() {
   while (1) {
      NSG = 1; EWR = 1;    /* turn on NSG, EWR */
      wait_ms(5000);

      while(EW_sensor){wait_ms(1);}

      NSG = 0; EWR = 0;
      NSY = 1; EWR = 1;   /* turn on NSY, EWR */
      wait_ms(2000);
      NSY = 0; EWR = 0;
      NSR = 1; EWG = 1;   /* turn on NSR, EWG */
      wait_ms(5000);

      while(NS_sensor){wait_ms(1);}

      NSR = 0; EWG = 0;
      NSR = 1; EWY = 1;    /* turn on NSR, EWY */
      wait_ms(2000);
      NSR = 0; EWY = 0;
   }
}
```