New Paltz
STATE UNIVERSITY OF NEW YORK

lab 5
Rural Traffic Light in assembly

EGc332– Microcontrollers

March 13, 2021

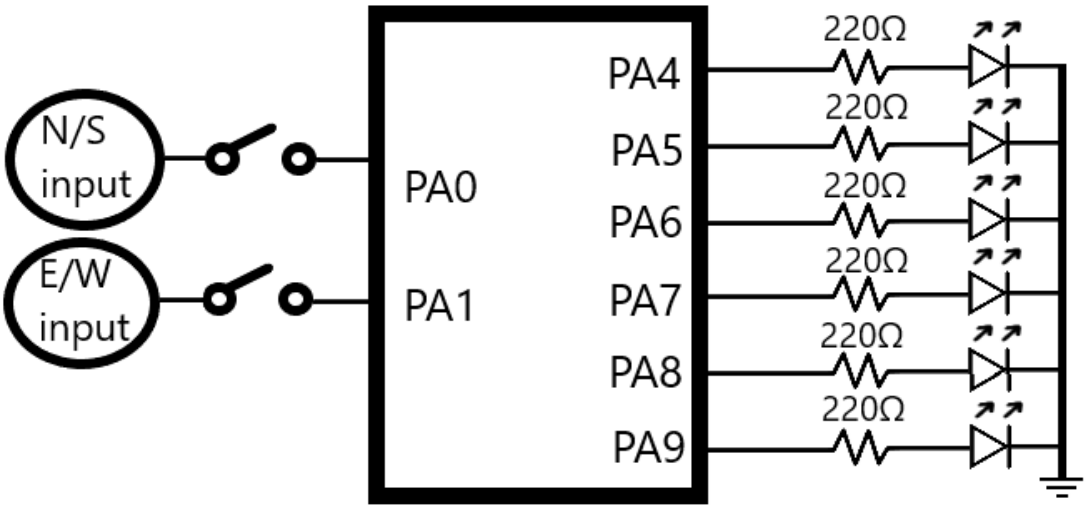
Christopher Lepore

Introduction

In this lab we designed a rural traffic light control system in assembly for a Nucleo-64 microcontroller. This introduced us to setting up and coding internal pull up resistors in assembly. This lab also introduced using AND and branch commands to read input port pins on the STM32F446RE circuit board. 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 a rural area there are two road sensors. When the NS sensor is logic 1 the NSG and EWR LEDs will stay on till the sensor switches to logic 0. The same goes for the EW sensor and the NSR and EWG LEDs. By designing this rural traffic light, a deeper comprehension of coding input ports in assembly with the STM32F446RE circuit board was achieved.

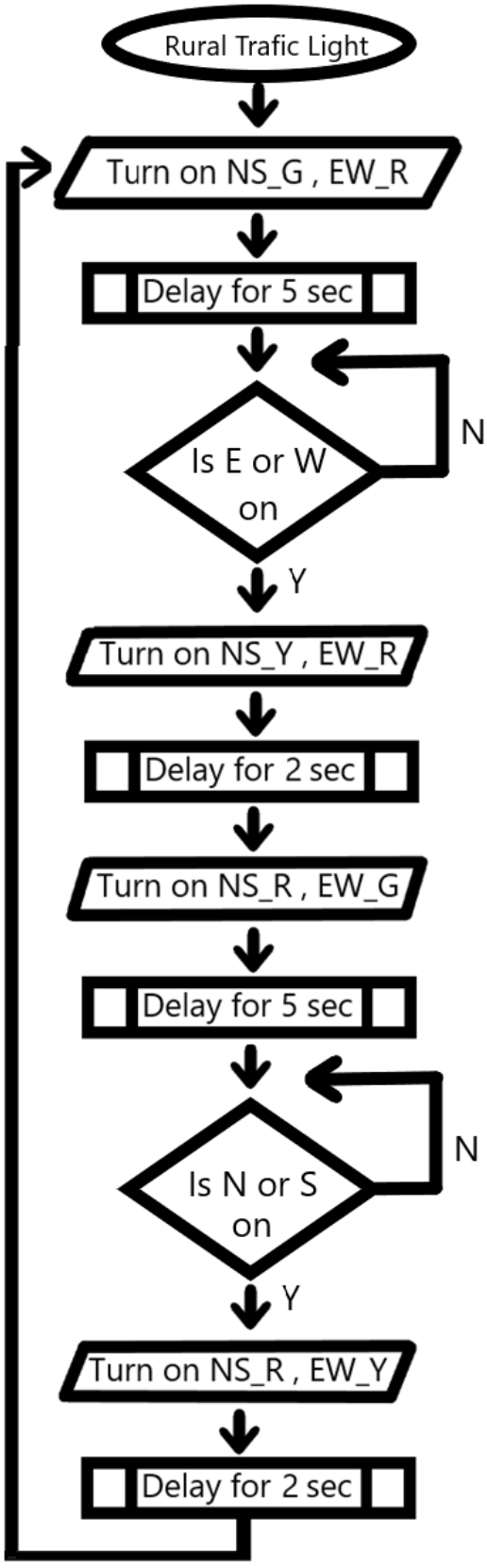
Design

Hardware (Block Diagrams)



For the rural 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. Port PA0 and PA1 were set to input mode for the NS and EW sensors.

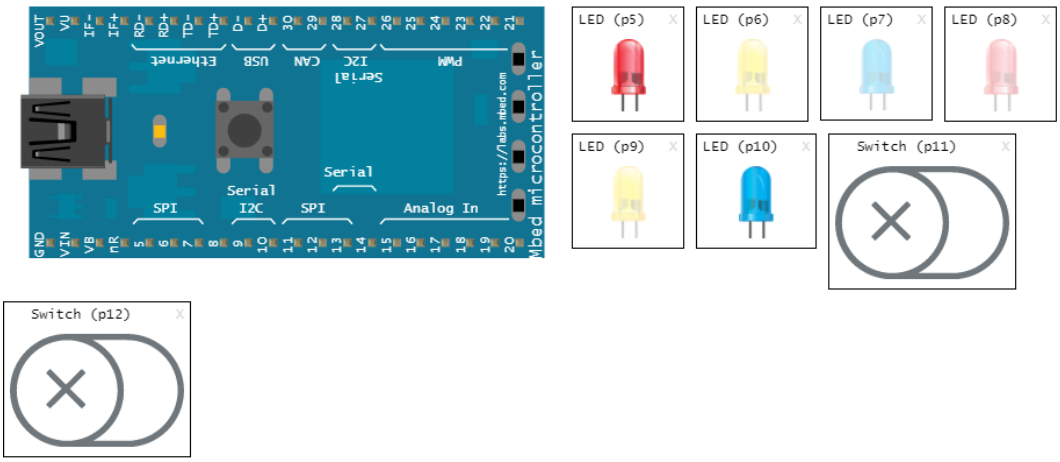
Software (Flow Charts)



In the rural 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. When the NS sensor is logic 1 the NSG and EWR LEDs will stay on till the sensor switches to logic 0. The same goes for the EW sensor and the NSR and EWG LEDs.

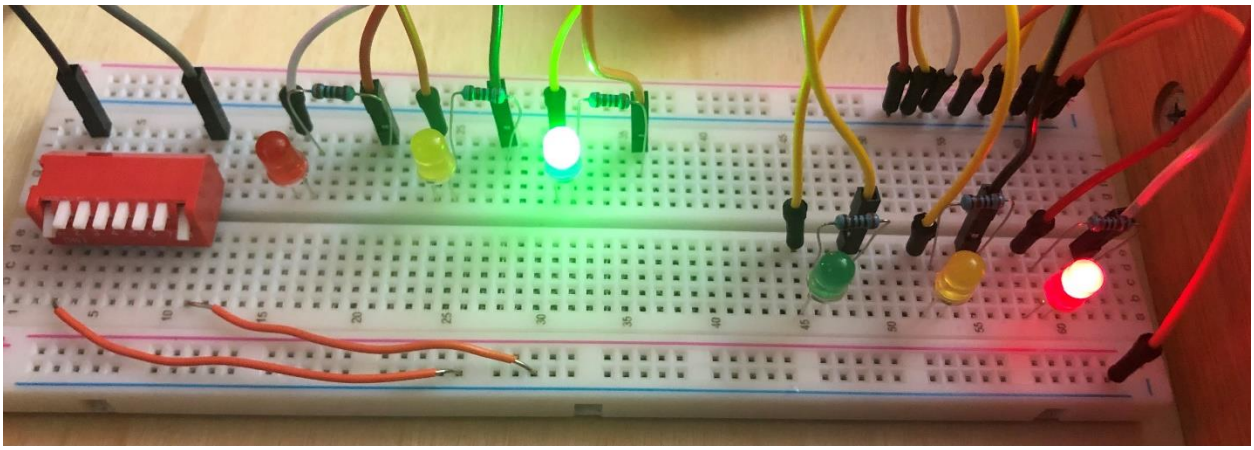
Verification

MBED simulation

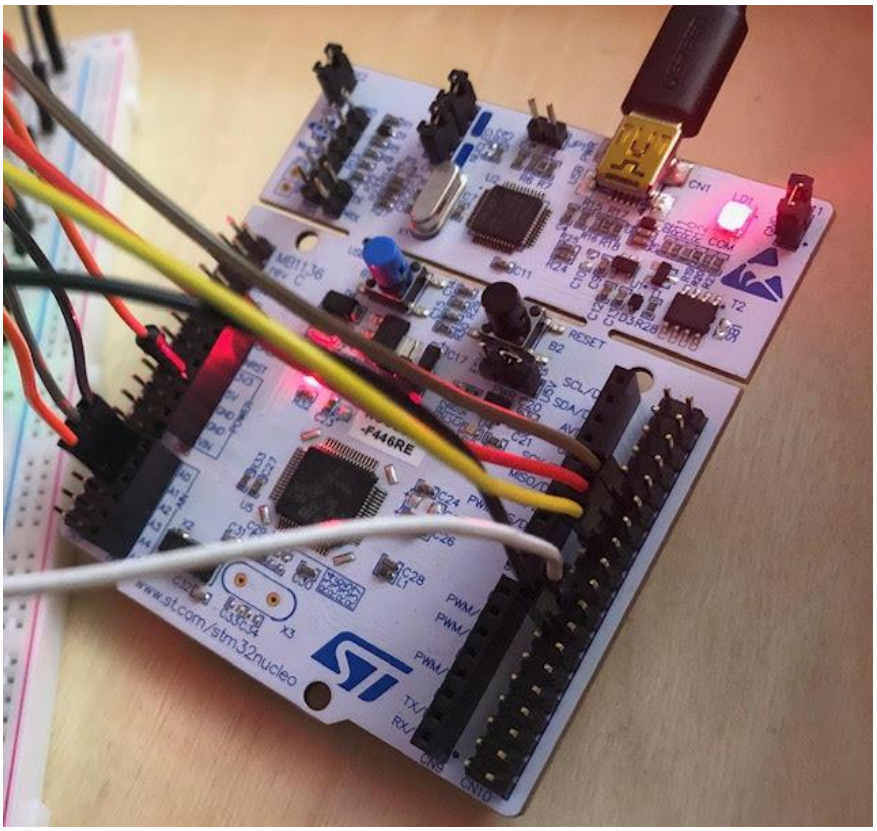


MBED simulation circuit

Nucleo and breadboard circuit



Breadboard wiring



NUCLEO circuit board

Summary and conclusion

In the traffic light circuit, port PA4 to PA9 will be wired to NSG, NSY, NSR, EWG, EWY, and EWR, respectively. The traffic light cycles through 4 stages. 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. If the input port PA0 (NS sensor) is logic 1 at stage 1 then the program will stay at that stage till PA0 is set to logic 0. Also, if the input port PA1 (EW sensor) is logic 1 at stage 3 then the program will stay at that stage till PA0 is set to logic 0. The sensor functions by loading the value from the memory location of desired port into a register and then ANDing it with the 32-bit immediate value that represents logic 1. Then it branches to the beginning of the sensor stage such as the “L2” loop using the BNE command which will loop only if the value read is equal to logic 1 for that port.

This introduced us to setting up and coding internal pull up resistors in assembly along with using AND and branch commands to read input port pins on the STM32F446RE circuit board. This was done by analyzing the schematic to learn what the 32-bit value is for the GPIOA_PUPDR and GPIOA_IDR and by using a ANDS with a BNE branch to make the sensors. Due to the lessons above this lab has prepared students to begin designing and coding more complex projects with inputs in assembly language.

Appendix (Programs)

Rural 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

GPIOA_PUPDR equ 0x4002000C
GPIOA_IDR   equ 0x40020010

        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]

        ldr             r4, =GPIOA_PUPDR
        ldr             r5, =0x00000005                 ; sets A0 and A1 to pull up resistor
        str             r5, [r4]


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

```
        mov     r0, #5000
        bl      delay


L2      ldr     r4, =GPIOA_IDR
        ldr     r5, [r4]
        ANDS    r4, r5, #1
        BNE     L2

        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

L3      ldr     r4, =GPIOA_IDR
        ldr     r5, [r4]
        ANDS    r4, r5, #2
        BNE     L3

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