John Danison

ECET 32900 – Lab 2 Report

1/31/2025

Lab Goal:

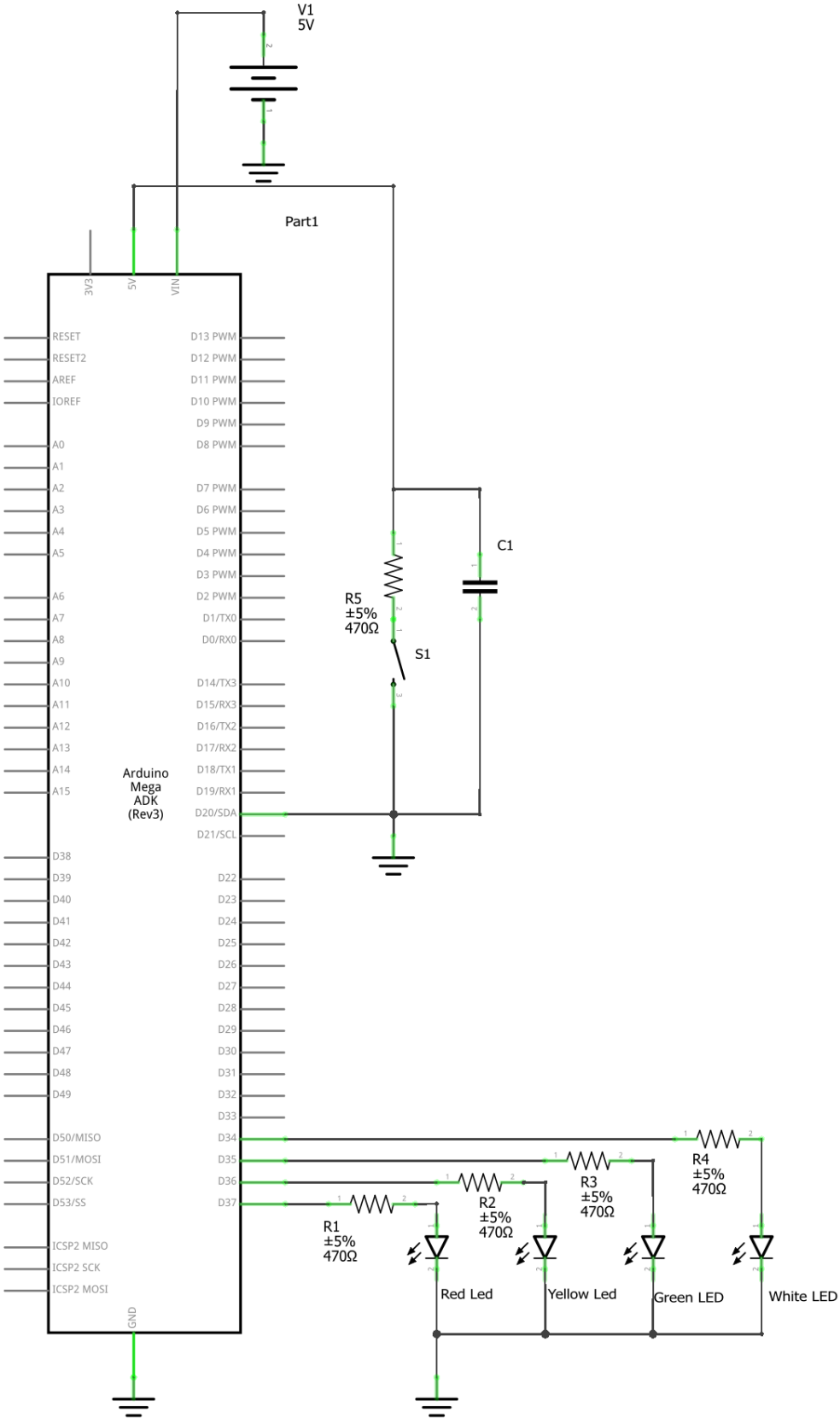
The goal of this lab was to recreate a traffic light pattern with a crosswalk button interrupt.

Procedure:

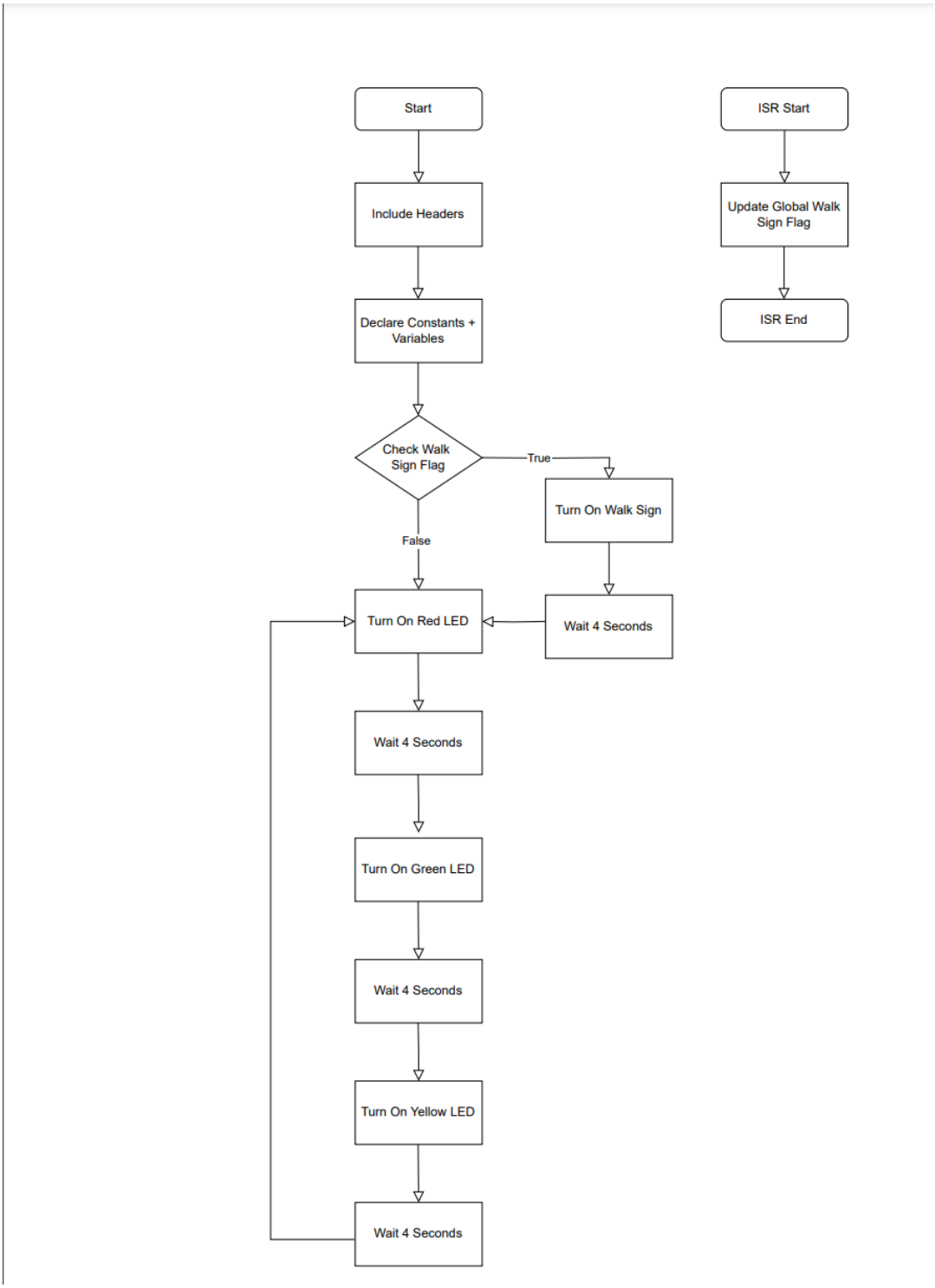
The procedure for this lab was very straightforward. I started with reviewing lab instructions and getting specifications answered by the Lab TA. Afterwards, the hardware for recreation was selected using an ATMega 2560 microcontroller, 4 simple LEDs in red, green, yellow, and white, and a simple push button. With components selected, a schematic was drawn to connect each of these items. Once completed, a flowchart was created in preparation for programming the microcontroller. Then the program was created, and a functional prototype was created. After testing, the system was checked off by Professor Panigraphi.

Lab Results:

Below is the schematic for the system.


V1
5V
Part1
3V3
5V
VIN
RESET
RESET2
AREF
IOREF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
D49
D50/MISO
D51/MOSI
D52/SCK
D53/SS
ICSP2 MISO
ICSP2 SCK
ICSP2 MOSI
GND
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D3 PWM
D2 PWM
D1/TX0
D0/RX0
D14/TX3
D15/RX3
D16/TX2
D17/RX2
D18/TX1
D19/RX1
D20/SDA
D21/SCL
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37
Arduino
Mega
ADK
(Rev3)
C1
R5
±5%
470Ω
S1
R1
±5%
470Ω
R2
±5%
470Ω
R3
±5%
470Ω
R4
±5%
470Ω
Red Led
Yellow Led
Green LED
White LED

Below is the Flow Chart for the system.



Below is the source code used for the system.

Lab2 > src > main.cpp > ...

You, 7 days ago | 2 authors (You and one other)

```cpp
/**
 * Filename: main.cpp
 * Author: John Danison
 * Date: 1/24/2025
 *
 * ECET 32900 Lab 2 - Interrupts Review
 * Description:
 *  This lab simplifies and simulates a traffic light sequence. The program will:
 *    1) Turn on a Red LED
 *    2) Wait 4 Seconds
 *    3) Turn off the Red LED and Turn on a Green LED
 *    4) Wait 4 Seconds
 *    5) Turn off the Green LED and Turn on a Yellow LED
 *    6) Wait 4 Seconds
 *    7) Turn off the Yellow LED and Repeat back to step 1
 *
 *    ISR)
 *     1) When a button is pressed and INT0 is triggered, the current stoplight LED will clear and a 4th White LED will turn on.
 *     2) Wait 4 seconds
 *     3) Continue with Stoplight sequence
 */

/* Libraries */
#include <Arduino.h>         // Standard Arduino Library
#include <avr/io.h>          // General Definition
#include <avr/interrupt.h>   // General Interrupt

/* Global Variables */
volatile bool walkSignISR = false;

      John Danison, 7 days ago • Added Lab 2 Platformio Framework
/* Global Constants */
int redStopLightLED = PORTC0;
int yellowStopLightLED = PORTC1;
int greenStopLightLED = PORTC2;
int whiteWalkSignLED = PORTC3;

/* Function Prototypes */
void init_io(void);
void init_interrupt(void);
void walkSignInterrupt(void);
```

```cpp
/* Setup Function */
void setup() {
  // Initializations
  init_io();        // IO Initialization
  init_interrupt();

  Serial.begin(9600);
  Serial.println("Setup Done");

  // Enable global interrupt
  sei();
}

/* Main Loop */
void loop() {
  /* Check for walk sign ISR */
  if (walkSignISR) {
    PORTC = 0x00;
    PORTC ^= (1 << whiteWalkSignLED);
    Serial.println("White Light");
    delay(4000);
    walkSignISR = false;
  }

  /* Normal Stoplight Sequence */
  // Red LED
  PORTC = 0x00;
  PORTC ^= (1 << redStopLightLED);
  Serial.println("Red Light");
  delay(4000);

  // Green LED
  PORTC = 0x00;
  PORTC ^= (1 << greenStopLightLED);
  Serial.println("Green Light");
  delay(4000);

  // Yellow LED
  PORTC = 0x00;
  PORTC ^= (1 << yellowStopLightLED);
  Serial.println("Yellow Light");
  delay(4000);
}
```

```
/* Custom Functions */
// Function for IO initialization
void init_io(void)
{
  // PORTC is the output for the Stop Lights LEDs
  DDRC = 0xFF;
  PORTC = 0x00;

  // PORTD.0 as INT0 Input
  DDRD = 0x00;
  PORTD = 0x00;
}

// Function for Interrupt initialization
void init_interrupt(void) {
  EICRA = (1 << ISC10) | (1 << ISC11);
  EIMSK = (1 << INT1);
}

/* Interrupt */
ISR(INT1_vect) {
  walkSignISR = true;
}
```

The system works by running a default light sequencing order. The code will start with running a Red LED for 4 seconds, then running the Green LED for 4 seconds and running the Yellow LED for 4 seconds before returning back to the Red LED. This sequence loops forever. Before the Red LED is turned on, the program checks a global variable that holds the current state of the walk sign button. If a user presses the walk sign button, a flag is raised and at the beginning of the next light cycle, the walk sign will turn on for the user. This means that there is a delay between hitting the button and the walk sign turning on.

Conclusion:

This lab was useful for learning the process behind designing embedded systems while reviewing fundamentals taught in previous courses. While I didn't learn anything new technically, I am learning the format of the class and what to expect from week to week. This lab also made me think about how the current traffic light sequencing is controlled, and what tricks are used to maximize traffic flow while minimizing accidents.

References

ECET 27900. (Spring 2024). *Laboratory resources used as reference for weekly lab*. Purdue University.

ECET 32900. (Spring 2025). *Peer collaboration for debugging assistance*. Purdue University.

Electronicshub. (2021). *Arduino Mega pinout*. Retrieved from https://www.electronicshub.org/wp-content/smush-webp/2021/01/Arduino-Mega-Pinout.jpg.webp

Microchip Technology. (n.d.). *ATmega640/1280/1281/2560/2561 datasheet* (Atmel 2549). Retrieved from https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf