Arduino Assignment

Traffic Light System

LIT
LIMERICK INSTITUTE
OF TECHNOLOGY

Joe Morais (K00254840)

Software Development Y2S2

Real Time Embedded Systems

Contents

Introduction

The purpose of this report is to outline and describe the steps taken to complete the Arduino Traffic Light assignment for the Real Time Embedded Systems module.

Requirements

The project requirements are to design and implement a solution for an Arduino circuit to simulate a traffic light system. The circuit include two sets of traffic lights, a photoresistor and 2 buttons. Further requirements were included, such as:

Sequence Timers: The sequence for the traffic lights uses default timers that can be modified in specific circumstances.

Sensors: The buttons simulate sensors to detect if there are cars at each traffic light. If there are no cars at the opposite traffic light, the timer for the green light (on the opposing side) can be reduced. Cars on both lights means the timers will retain their default value.

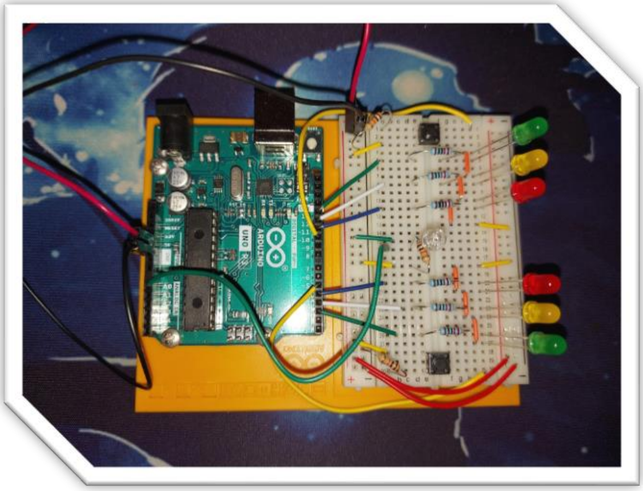
Day/Night: The photoresistor is used to simulate day and night. Depending on the amount of light entering the resistor, the maximum number of seconds for green lights are modified.

Solution

A solution for this problem was created using a circuit design and code created in the Arduino IDE. Several versions of the both the circuit and code were created and tested.

Circuit

The final version of the circuit can be seen below:

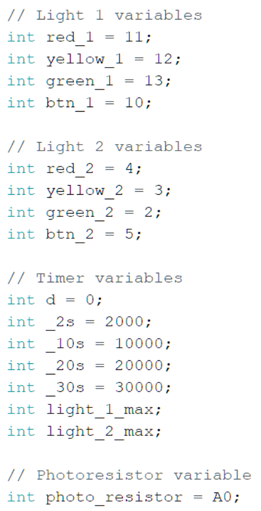


Coding & Implementation

The final version of the code included several methods, which are used to change the lights, change timers, and read inputs. The code is self-explanatory, but comments were also added for comprehensibility.

Global Variables

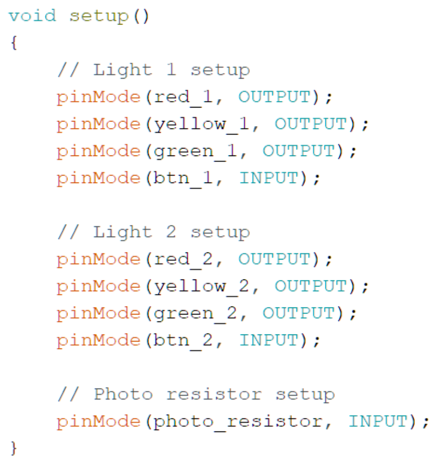
For readability purposes, variables were created for each specific component and their corresponding pin numbers. Variables were also used for the timers.



```
// Light 1 variables
int red_1 = 11;
int yellow_1 = 12;
int green_1 = 13;
int btn_1 = 10;

// Light 2 variables
int red_2 = 4;
int yellow_2 = 3;
int green_2 = 2;
int btn_2 = 5;

// Timer variables
int d = 0;
int _2s = 2000;
int _10s = 10000;
int _20s = 20000;
int _30s = 30000;
int light_1_max;
int light_2_max;

// Photoresistor variable
int photo_resistor = A0;
```

Setup

The setup function simply involved assigning each component to the correct pin number and their function (INPUT/OUTPUT).



```
void setup()
{
    // Light 1 setup
    pinMode(red_1, OUTPUT);
    pinMode(yellow_1, OUTPUT);
    pinMode(green_1, OUTPUT);
    pinMode(btn_1, INPUT);

    // Light 2 setup
    pinMode(red_2, OUTPUT);
    pinMode(yellow_2, OUTPUT);
    pinMode(green_2, OUTPUT);
    pinMode(btn_2, INPUT);

    // Photo resistor setup
    pinMode(photo_resistor, INPUT);
}
```

Loop

The loop first checks if it is day or night, then sets the max timers as described in the project specification document. It then enters the sequence and calls methods that changes the lights to specific colours.

```
void loop()
{
    // Check if daylight is detectable and set max values
    if(analogRead(photo_resistor) > 100)         // Set day timers
    {
        light_1_max = _10s;
        light_2_max = _10s;
    }
    else                                         // Set night timers
    {
        light_1_max = _30s;
        light_2_max = _20s;
    }

    light_1_green(light_1_max);                  // Set light 1 to green for max num of seconds
    light_1_yellow();                            // Set light 1 to yellow

    both_red();                                  // Set both lights to red

    light_2_green(light_2_max);                  // Set light 2 to green for max num of seconds
    light_2_yellow();                            // Set light 2 to yellow

    both_red();                                  // Set both lights to red
}
```

Light 1 Green

This method requires a *max_time* parameter, which is set based on the value of the photoresistor, and then passed to it in the loop function.

To set light 1 to green, the program clears all lights, turns on green on light 1, and turns on red on light 2. It then calls the *green_wait()* method.

```
/*  Sets light 1 to green
 *  @param max_time the maximum time the light can stay green for (defined by photoresistor) */
void light_1_green(int max_time)
{
    clear_all();                                 // Clear all lights
    digitalWrite(green_1, HIGH);                 // Light 1 green on
    digitalWrite(red_2, HIGH);                   // Light 2 red on
    green_wait(btn_1, btn_2, max_time);          // Wait between 2s and max_time, while waiting for input
}
```

Light 1 Yellow

To set light 1 to yellow, the program clears all lights, turns on yellow on light 1, turns on red on light 2, and waits 2 seconds.

```
/*  Sets light 1 to yellow */
void light_1_yellow()
{
    clear_all();                                 // Clear all lights
    digitalWrite(yellow_1, HIGH);                // Light 1 yellow on
    digitalWrite(red_2, HIGH);                   // Light 2 red on
    delay(_2s);                                  // Wait 2s
}
```

Light 2 Green

This method requires a *max_time* parameter, which is set based on the value of the photoresistor, and then passed to it in the loop function.

To set light 2 to green, the program clears all lights, turns on green on light 2, and turns on red on light 1. It then calls the *green_wait()* method.

```
/*  Sets light 2 to green
 *  @param max_time the maximum time the light can stay green for (defined by photoresistor) */
void light_2_green(int max_time)
{
    clear_all();                            // Clear all lights
    digitalWrite(green_2, HIGH);            // Light 2 green on
    digitalWrite(red_1, HIGH);              // Light 1 red on
    green_wait(btn_2, btn_1, max_time);     // Wait between 2s and max_time, while waiting for input
}
```

Light 2 Yellow

To set light 2 to yellow, the program clears all lights, turns on yellow on light 2, turns on red on light 1, and waits 2 seconds.

```
/*  Sets light 2 to yellow */
void light_2_yellow()
{
    clear_all();                            // Clear all lights
    digitalWrite(yellow_2, HIGH);           // Light 2 yellow on
    digitalWrite(red_1, HIGH);              // Light 1 red on
    delay(_2s);                             // Wait 2s
}
```

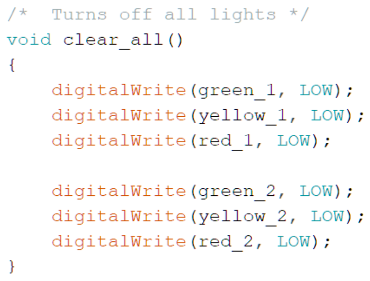
Both Lights Red

To set both lights to red, the program clears all lights, turns on red on light 1, turns on red on light 2, and waits 2 seconds.

```
/*  Sets both lights to red */
void both_red()
{
    clear_all();                            // Clear all lights
    digitalWrite(red_1, HIGH);              // Light 1 red on
    digitalWrite(red_2, HIGH);              // Light 2 red on
    delay(_2s);                             // Wait 2s
}
```

Clear All Lights

This method turns off all lights.



```
/*  Turns off all lights */
void clear_all()
{
    digitalWrite(green_1, LOW);
    digitalWrite(yellow_1, LOW);
    digitalWrite(red_1, LOW);

    digitalWrite(green_2, LOW);
    digitalWrite(yellow_2, LOW);
    digitalWrite(red_2, LOW);
}
```

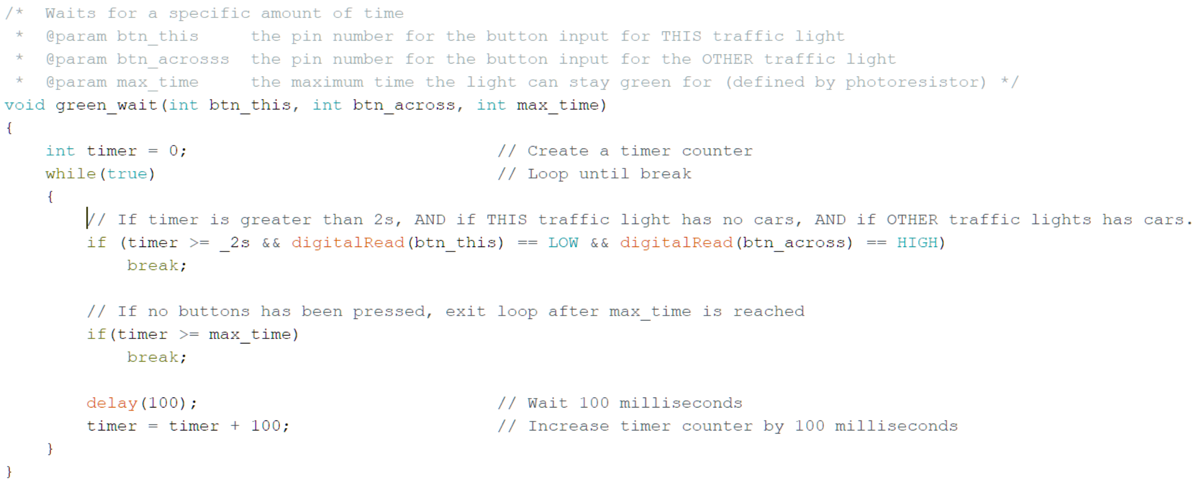
Green Light Timer

The method *green_wait* is be used by both set of traffic lights to wait for a specific amount of time and to read input from the buttons.

The method takes three parameters: the button number for the current traffic light that is green, the button number for the other traffic light, and the maximum amount of time the current traffic light should stay green.

A timer counter is created to keep track of how much time has passed at each iteration. A loop begins and it checks if the minimum amount of time has been reached (2s), if the button for the current traffic light is off, and if the button for the other traffic light is on. If these conditions are all met, the loop ends.

The method also checks whether the maximum time has been reached, in which case, it also exits the loop. Each iteration has a delay of 100 milliseconds. A longer delay would cause the input to not be read accurately. The time is tracked by incrementing the timer counter variable by 100 milliseconds after each iteration.



```
/*  Waits for a specific amount of time
 *  @param btn_this     the pin number for the button input for THIS traffic light
 *  @param btn_acrosss  the pin number for the button input for the OTHER traffic light
 *  @param max_time     the maximum time the light can stay green for (defined by photoresistor) */
void green_wait(int btn_this, int btn_across, int max_time)
{
    int timer = 0;                              // Create a timer counter
    while(true)                                 // Loop until break
    {
        // If timer is greater than 2s, AND if THIS traffic light has no cars, AND if OTHER traffic lights has cars.
        if (timer >= _2s && digitalRead(btn_this) == LOW && digitalRead(btn_across) == HIGH)
            break;

        // If no buttons has been pressed, exit loop after max_time is reached
        if(timer >= max_time)
            break;

        delay(100);                             // Wait 100 milliseconds
        timer = timer + 100;                    // Increase timer counter by 100 milliseconds
    }
}
```