Experiment No. 07:

Traffic Light Controller Implementation on Arduino UNO

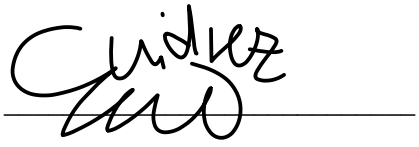
By: Theo Guidroz

Instructor: Dr. Jafar Saniie

ECE 441-001

Due Date: 04-19-2020

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature : 

I. Introduction

A. Purpose

The purpose of this lab is to introduce students to Arduino and programming in C. Using Tinkercad, students will build circuits to familiarize themselves with the Arduino UNO microcontroller.

B. Background

Arduino UNO is a microcontroller with an IC chip. It consists of a microprocessor, RAM and I/O ports, oscillator, I/O ports, and a reset button. This microprocessor is wildly used for prototyping embedded systems such as toys, elevators and other objects which use sensors. Arduino UNO is programmed in C, a high-level language. It is also an open-source microcontroller and includes Integrated Development Environment (IDE) and a USB communication port.

Arduinos are programmed in C because this language allows its users to modify the lower level of hardware while still being a high-level language. Created by Dennis Ritchie, it was designed to re-implement UNIX operating system which was originally written in assembly language. C is still commonly used to implement kernels of operating systems and embedded systems.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- A PC that runs Tinkercad

B. Procedure

1. Understand the basic functions related to I/O operations: pinMode(), digitalRead(), digitalWrite().

2. Design a traffic light system using pushbuttons as vehicle sensors.

3. Write the C code to implement the traffic light system.

4. Simulate the system and verify that it works.

III. Results and Analysis

A. Discussion

Schematics

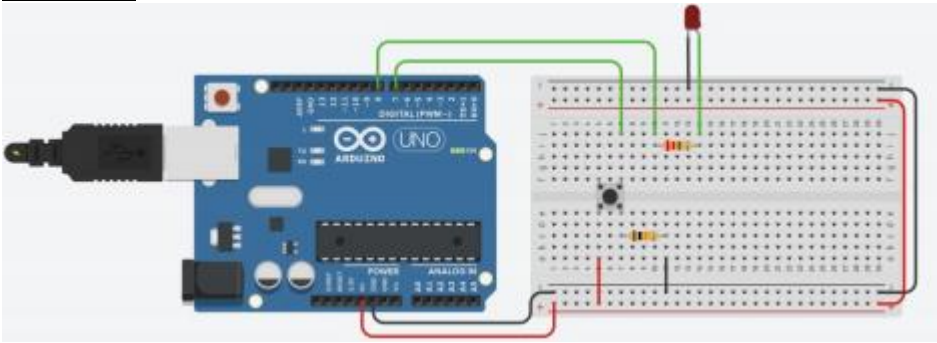


Figure 1: The circuit used to test the basic I/O function.

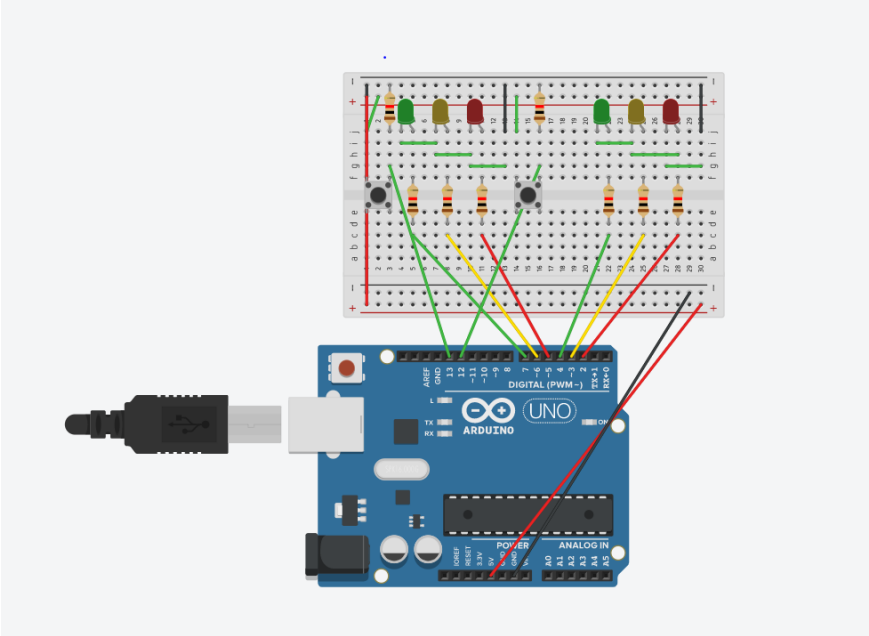


Figure 2: The schematic used to implement the traffic light with sensors.

Programs

```
void setup()
{
 pinMode(8, OUTPUT);
}
void loop()
{
 digitalWrite(8, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(8, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
}
```

Figure 3: Program used to learn how digitalWrite() work.

```
void setup()
{
 pinMode(8, OUTPUT);
 pinMode(7, INPUT);
}
void loop()
{
 if(digitalRead(7)){
 digitalWrite(8, HIGH);
 }
 else{
 digitalWrite(8, LOW);
 }
}
```

Figure 4: Program used to learn how digitalRead() work.

```
// North_South LEDs
#define r_ns 2
#define y_ns 3
#define g_ns 4
// East_West LEDs
#define r_ew 5
#define y_ew 6
#define g_ew 7


void setup()
{
// put your setup code here, to run once:
// initialize all LEDs as output
   pinMode(r_ns, OUTPUT);
   pinMode(y_ns, OUTPUT);
   pinMode(g_ns, OUTPUT);
   pinMode(r_ew, OUTPUT);
   pinMode(y_ew, OUTPUT);
   pinMode(g_ew, OUTPUT);
// Initialize all pushbuttons as input
      pinMode(12, INPUT);
      pinMode(13, INPUT);
}
void ChangeLedValue(byte number)
{
   // This changes the values of the LEDs depending on the bit pattern
   digitalWrite(r_ns, bitRead(number, 5));
    digitalWrite(y_ns, bitRead(number, 4));
   digitalWrite(g_ns, bitRead(number, 3));
   digitalWrite(r_ew, bitRead(number, 2));
```

```
   digitalWrite(y_ew, bitRead(number, 1));
   digitalWrite(g_ew, bitRead(number, 0));
}


void LightSequence()
{
begining:
    ChangeLedValue(B001100); // Set the initial stop-light values on the
LEDs
    if(digitalRead(12) || digitalRead(13)){
    // The sensor is trigerred on E/W road, activate light switching
        ChangeLedValue(B001100);
   // If so, we will start a timer that will change the stop-lights
    } else { // Else, keep looping until one of the sensors is activated
         goto begining;
    }
    delay(5000);    // When the sensor is triggered, allow 5 five seconds
before changing the lights
    ChangeLedValue(B010100); // Stop light change
    delay(2000);    // Small delay (since yellow light is short)
    ChangeLedValue(B100100); // Stop light change
    delay(2000);    // Small delay (since yellow light is short)
    ChangeLedValue(B100001); // Stop light change
    delay(10000);   // East/West light is green for 10 seconds
    ChangeLedValue(B100010); // Stop light change
    delay(2000);    // Small delay (since yellow light is short)
    ChangeLedValue(B100100); // Stop light change
    delay(2000);    // Small delay (since yellow light is short)
}


void loop()
```

```
{
    LightSequence();      // Infinite loop that restarts the process
}
```

Figure 5: Program used for the traffic light system.

Questions

1. **A schematic diagram of your hardware design.**

   See figure 2.

2. **A fully commented listing of the traffic light controller program.**

   See figure 5.

3. **Write a brief discussion on (a) what is Arduino, (b) what Arduino can achieve and what is the shortcoming. (At least 200 words).**

   a. Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards can read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. Most Arduino boards are microcontrollers that fit on one Integrated Chip. They usually feature a CPU, RAM, and I/O ports, and software to implement programs written in C. Arduino boards are commonly used for prototyping since sensors can easily be attached to them and output devices such as LEDs can be programmed to react to the sensors. Programs can be loaded on most Arduino boards using an IDE and this makes it very simple to control both hardware and software of a specific design.
   b. Arduino boards are very versatile and can be used for an endless amount of applications. The most common use of Arduinos is to control other devices such as sensors and LEDs. When designing an embedded system, it is important to optimize the hardware and the software to reduce manufacturing costs. The best way of optimizing is by prototyping and Arduino boards are the perfect tools for that. Moreover, Arduino boards are wildly used by students for projects since they can mimic real-world solutions on the board. Designing the traffic lights is one example. On the other hand, Arduino limits the details of hardware complexity. There is so much that can be learned about hardware with Arduinos since it does most of the work. Arduinos are also not meant for large scale production since its purpose is very vague. It would be very expensive to mass-produce an embedded system for a specific purpose using Arduinos.

Describe Implement a clock with LCD display in TinkerCAD. The time has to be formatted in HH:mm:ss MM/dd/yyyy. The initial time can be hard coded in your program. A schematic diagram of your hardware design and a fully commented listing of the program need to be included in the lab report.

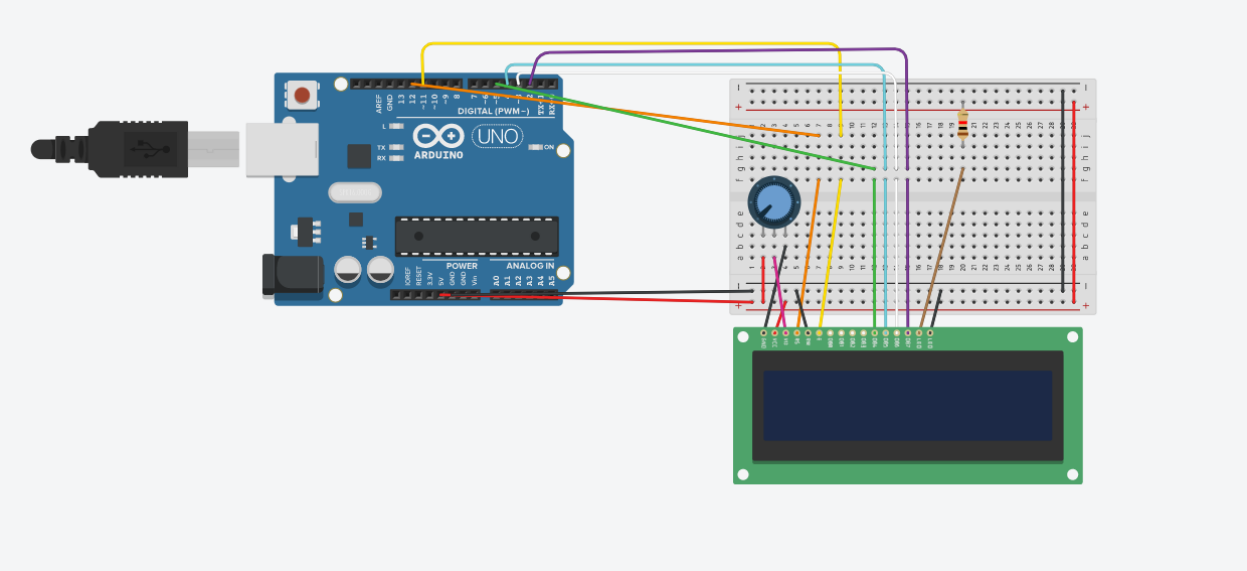


Figure 6: Clock schematic.

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// The initial times below check:
// If the day increments correctly if hour > 23
// If the hour increments correctly if minute > 59
// If the minute increments correctly if second > 59
// If Feb 28th goes to Feb 29th because the year is a leap year

  int hour = 23;   // Initial hour
  int minute = 59; // Initial minute
  int second = 58; // Initial second
  int month = 2;   // Initial month
  int day = 28;    // Initial day
  int year = 2020; // Initial year

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  loop();
}

void set_time(){
```

```
  // Check if second has gone over 59; if so, increment minute set second = 0
  if(second > 59){
      ++minute;
    second = 0;
  }

  // Check if min has gone over 59; if so, increment hour set min = 0
  if(minute > 59){
    ++hour;
    minute = 0;
  }

  // Check if hour has gone over 23; if so, increment day set hour = 0
  if(hour > 23){
      ++day;
    hour = 0;
  }

  // Jan, March, May, July, August, October, and December have 31 days
  if(day > 31 && (month == 1 || month == 3 || month == 5 ||
                  month == 7 || month == 8 || month == 10)) {
    ++month;
    day = 1;
  // If the month is December, add 1 year
  } else if (day > 31 && month == 12){
      ++year;
    month = 1;
    day = 1;
  }

  // Check for leap year, if month is February and day is 28th
  // If the year is divisible by 4, and divisible by 100 AND 400,
  // it is a leap year. Otherwise, it is not, and you must increment the
month
  // and reset the day.

  if(day > 28 && month == 2){
     if((year % 4 != 0) && ((year % 400 != 0) ||
                         (year % 100 == 0))){
      ++month;
        day = 1;
     }
  }

  // April, June, September, and November have 30 days
  // Therefore, you should only check if day has gone over 30 days
  // to increment the month correctly

  if(day > 30 && (month == 4 || month == 6 || month == 9 ||
                  month == 11)) {
    ++month;
    day = 1;
  }

}

void loop() {
```

```
// set the cursor to column 0, line 1
// (note: line 1 is the second row, since counting begins with 0):
lcd.setCursor(0, 0);

// This will always print two digits for the hour field (i.e. 04)
if(hour < 10){
    lcd.print("0");
}

// Print hour at location (0,0)
lcd.print(hour);
lcd.setCursor(2, 0);
// This will print the ‘:’ character for HH:mm:ss format
lcd.print(":");
lcd.setCursor(3, 0);
// This will always print two digits for the minute field (i.e. 04)
if(minute < 10){
    lcd.print("0");
}
lcd.print(minute);
// Print hour at location (3,0)
lcd.setCursor(5, 0);
// This will print the ‘:’ character for HH:mm:ss format
lcd.print(":");
lcd.setCursor(6, 0);
// This will always print two digits for the second field (i.e. 04)
if(second < 10){
    lcd.print("0");
}
// Print hour at location (6,0)
lcd.print(second);

// For the date format, we will print it on the second row

// Print month at (0,1)

lcd.setCursor(0, 1);
if(month < 10){
    lcd.print("0");
}
lcd.print(month);

// This will print the ‘/’ character for MM:dd:yyyy format

lcd.setCursor(2, 1);
lcd.print("/");

// Print day at (3,1)

lcd.setCursor(3, 1);
if(day < 10){
    lcd.print("0");
}
lcd.print(day);

// This will print the ‘/’ character for MM:dd:yyyy format
```

```
  lcd.setCursor(5, 1);
  lcd.print("/");

  // Print year at (6,1)

  lcd.setCursor(6, 1);
  lcd.print(year);
  // Delay 1 second and increment the second
  delay(1000);
  ++second;
  // Infinite loop to check time
  set_time();
}
```

Figure 7:

IV. Conclusions

For this lab, students became aware of the main functions of an Arduino Uno and its importance. The schematics were built on TinkerCAD and both programs were successful when simulated. The clock even took into consideration leap years! Students became more comfortable coding in C and realized its importance in the embedded system world. Overall, this lab was a success and students can now comfortably set up an Arduino UNO and program it.

References

[1] Experiment 7 Lab Manual

[2] "Arduino," [Online]. Available: https://www.arduino.cc

[3] "TinkCAD", [Online]. Available: https://www.tinkercad.com