

EXPERIMENT #7

Traffic Light Controller Implementation on Arduino UNO

1.0 Purpose

The purpose of this experiment is to introduce the user to the following items:

- Basic operation of the Arduino UNO board
- C programming on Arduino
- Integrated Development Environment (IDE) of Arduino
- Traffic light controller logic

The purpose of this exercise is to familiarize the student with the basic operation of the Arduino UNO board and Arduino IDE. Students are asked to implement traffic light controller on the Arduino using C programming. A simple LED and pushbutton circuit must be built on the breadboard to be interfaced with the Arduino UNO board.

2.0 Component Requirements

- 1 x Arduino UNO board and USB cable
- 2 x Red LED
- 2 x Yellow LED
- 2 x Green LED
- 2 x Pushbutton
- 6 x 220 Ohm & 2 x 10k Ohm Resistors
- 1 x Breadboard

3.0 Background

A. Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project. [1]

- B. For complex, modern traffic lights, often, the use of a micro-controller to handle all the timing functionality of the light is essential. Coordinating the three main signal lamps for each direction, left and right turn arrows, and walk and don't walk signals is not an easy task. It may also be beneficial to alter the timing of the light based upon traffic sensors and time of day. Handling all this functionality is simply too daunting a task for simple logic circuits. In addition, the programming capabilities of a micro-controller allow the timing to be altered relatively easily when necessary. With this said, there are some simple applications in which the cost of developing a micro-controller system and its corresponding software would simply not be justified. For these situations, a simple sequential logic circuit may be constructed to provide all the required functionality.

B-1. Simple Two-Way Intersection

The first logic that will be designed for this laboratory experiment is one for controlling a traffic signal at a very simple intersection. Consider an intersection of a north/south road and an east/west road, as shown in Figure 1 below.

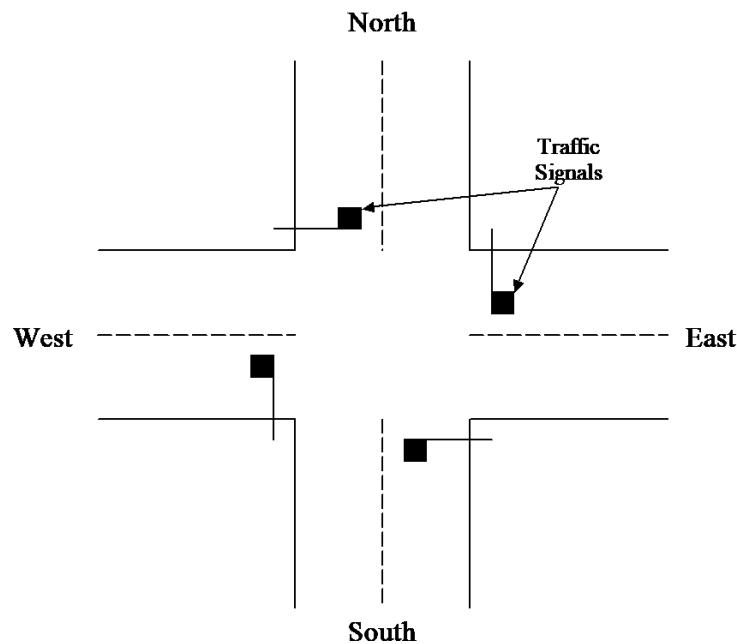


Figure 1 – Illustration of the intersection for the first traffic light controller.

For this first, very simple problem, each traffic signal only has three lamps: green for proceed, yellow for changing, and red for stop. A timer alone governs the changing of these lights and no additional sensor information is obtained regarding traffic flow or any other operating conditions. In addition, consider that the north/south road is busier than the east/west road; therefore, it is desirable to give longer green lights to the north/south road. The north/south road may be given ten time-unit green lights, while the east/west road is only given five time-unit green lights (for the purposes of this laboratory, a time-unit may be assumed to be the length of one clock cycle of a chosen frequency). Just as with a typical traffic light, when one direction is given a green light, the other direction must be given a red light. For the transition from green to red, a yellow light is given to the directions that had a

green light previously, for a duration of one time-unit. Finally, when the yellow light transitions to red, all four directions are provided with a red light for one time-unit, allowing for a margin of safety in the event someone decides to run the new red light. This functionality is summarized in the following flowchart.

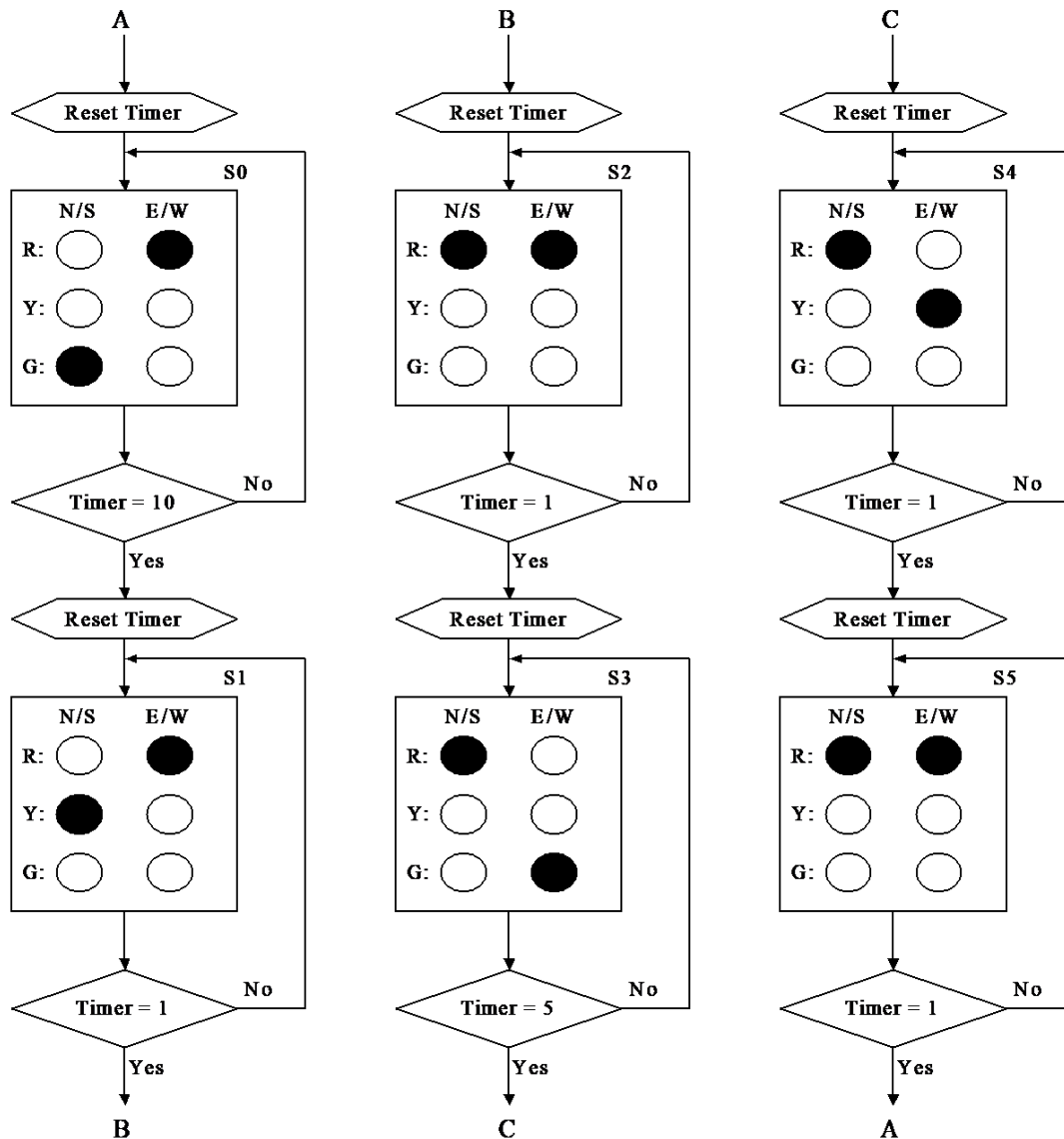


Figure 2 – Flowchart for the simple traffic light controller.

B-2. Two-Way Intersection with Traffic Sensors

For the second circuit to be developed in this laboratory experiment, consider that the previous traffic light no longer satisfies the needs of the traffic patterns on these two streets. If the traffic flow on the north/south street increases drastically while the traffic on the east/west street remains very low, the timing that was provided in the previous section could produce undesirable delays on the north/south road. If there are no cars waiting for the red light on the east/west street, there is no reason to change the north/south signal from green to

red. When a car arrives at the east/west traffic light, the same timing presented in the previous section may be initiated. In order to achieve this functionality, automobile sensors are inserted at the intersection as shown in the following figure.

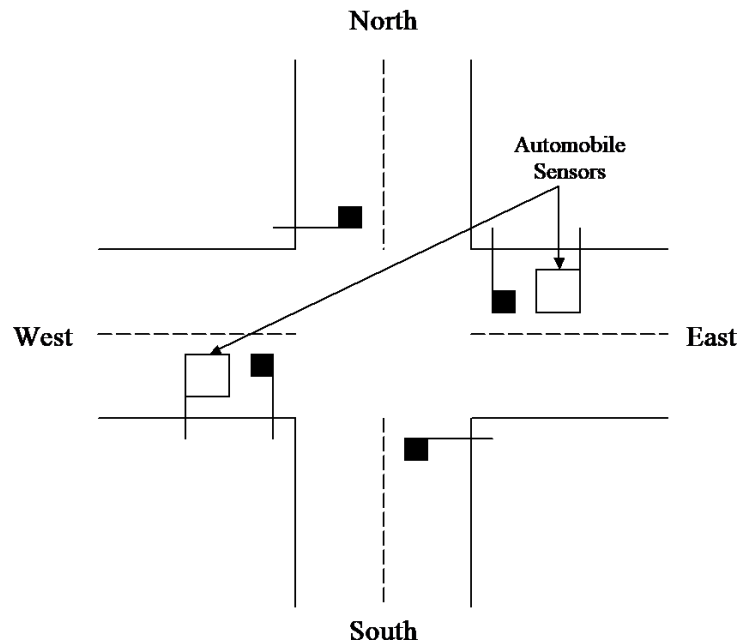


Figure 3 – Illustration of the intersection with the automobile sensors added.

The addition of the automobile sensors allows the traffic light controller to adapt to changing traffic patterns. This will allow the north/south street to have an unrestricted traffic flow if no cars are waiting at either of the cross-street traffic signals. The functionality of these automobile sensors, in conjunction with the traffic light timing, is summarized in Figure 4.

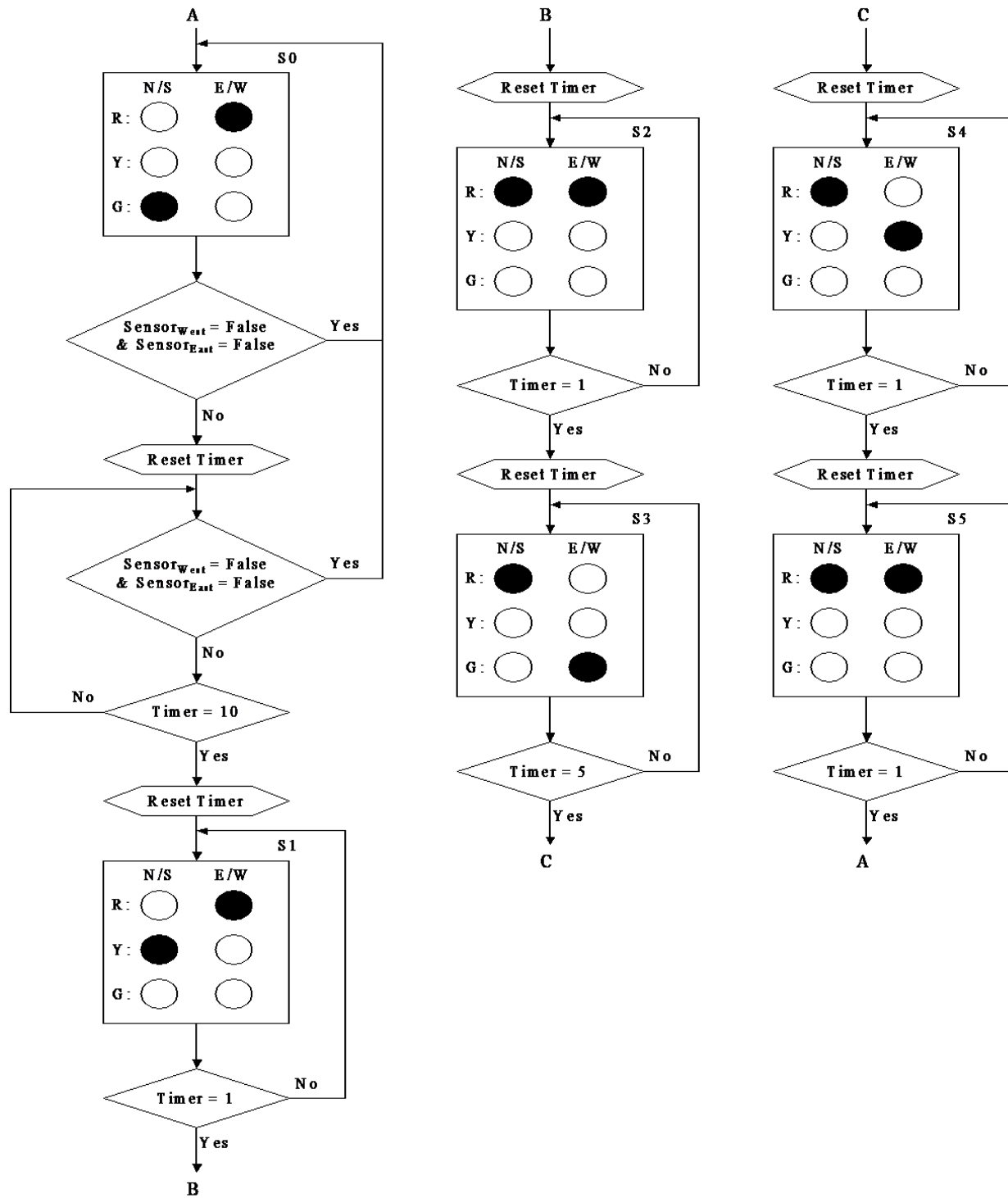


Figure 4 – Flowchart for the traffic light controller including automobile sensors.

4.0 Statement of Problem

In this lab, the students will be introduced to the Arduino Microcontroller and its Integrated Development Environment (IDE). Students are required to build a simple traffic light controller system with the Arduino UNO development board. A circuit with LEDs and pushbuttons will be built on the breadboard as the peripheral device of the microcontroller. It will also familiarize the students with programming in C language to control microcontrollers. By completing this laboratory experiment, students will gain confidence in building more complex embedded system in the future.

5.0 Preliminary Assignment

1. Follow the instructions in Appendix A to learn how to simulate a circuit with the Arduino UNO R3 controller on TinkerCAD. Scheme your own circuit and simulate the traffic light controller system presented in 3.0 Background B-1 Simple Two-Way Intersection. Show your result to the TA in the beginning of the lab.
2. Write a C program for the Arduino UNO development board, following the behavior and logic presented in 3.0 Background B-2 Two-Way Intersection with Traffic Sensors. Simulate it in TinkerCAD before coming to the lab. Show your result to the TA in the beginning of the lab.

6.0 Procedure

PART A – Introduction to Arduino and its Integrated Development Environment (IDE)

In this part, you will familiarize yourself with Arduino and its IDE by go through a step by step tutorial.

1. Digital I/O usage

The digital I/O pins on the Arduino allow you to connect sensors, LED, BTN, and various peripherals. Learning how to use them will allow you to use Arduino to implement a lot of useful designs. There are 3 basic functions that are related to digital I/O operations.

- pinMode()
- digitalWrite()
- digitalWrite()

In Figure 5 is a schematic that contains a LED as the digital output and a push button as the digital input. We will use this circuit to get to understand how these three functions work. Please build the circuit on the bread board and connect it to Arduino UNO development board according to Figure 5. The resistor used for LED is 220 Ohm to limit the current draw for the digital pin. The pull up resistor used for the push button is 10k Ohm.

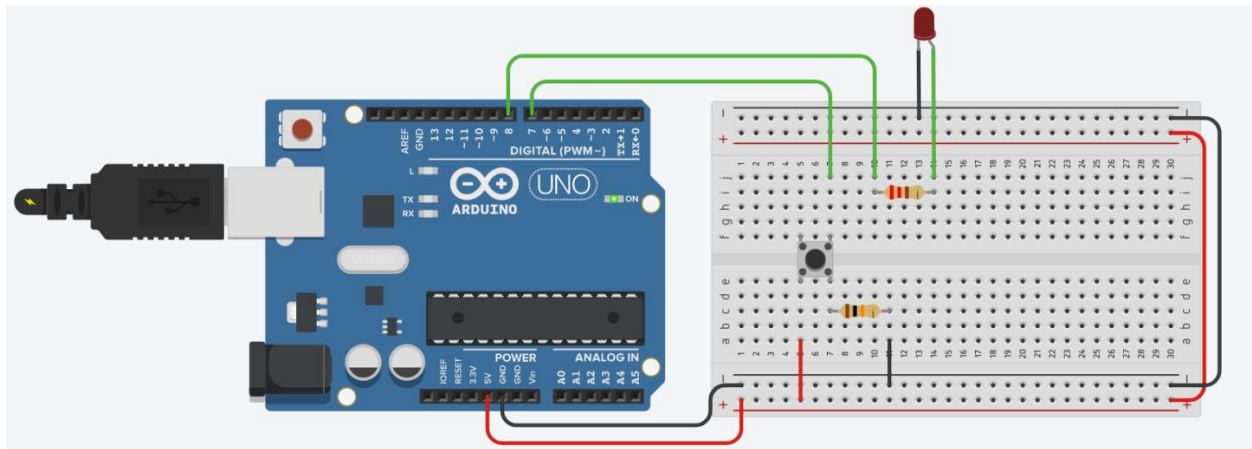


Figure 5. Arduino Circuit with Push Button and LED.

Follow Appendix B and load the following program to the Arduino UNO to learn how `digitalWrite()` work.

```
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)
}
```

Load the following code to Arduino UNO to learn how `digitalRead()` work.

```
void setup()
{
  pinMode(8, OUTPUT);
  pinMode(7, INPUT);
}

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

PART B – Traffic light controller implementation on Arduino UNO

1. In this part, you are going to implement the traffic light controller of a Two-Way Intersection described in 3.0 Background B-1 Simple Two-Way Intersection.

Using the schematic from your prelab, build the circuit on the breadboard. Since the traffic light on the same direction are identical, we use 6 LEDs to build the circuit for the traffic light controller. Use the following table for pinout setup of these LEDs.

Digital Pin 2	OUTPUT	Red North/South Direction
Digital Pin 3	OUTPUT	Yellow North/South Direction
Digital Pin 4	OUTPUT	Green North/South Direction
Digital Pin 5	OUTPUT	Red East/West Direction
Digital Pin 6	OUTPUT	Yellow East/West Direction
Digital Pin 7	OUTPUT	Green East/West Direction

Loading the program in Table 1.1 to the Arduino UNO. Show your result to your TA.

2. In this part, you are going to implement the traffic light controller of a Two-Way Intersection with **Traffic Sensors** described in 3.0 Background B-2 Two-Way Intersection with Traffic Sensors.

For easier implementation, the vehicle sensor will be replaced by the push button. When the button is pushed, it represents for cars detected. Using the same circuit finished in part 1, you can add two push buttons to the circuit. With the modified circuit, you must write the C code to implement the program logic that is described in Figure 4. Use the following table for pinout setup for the push buttons.

Digital Pin 12	INPUT	Automobile Sensor on West
Digital Pin 13	INPUT	Automobile Sensor on East

Finish the code, load it to the Arduino UNO board. Show your result to your TA.

7.0 Discussion

1. A schematic diagram of your hardware design. (10pts)
2. A fully commented listing of the traffic light controller program. (5pts)
3. Write a brief discussion on (a) what is Arduino, (b) what Arduino can achieve and what is the shortcoming. (At least 200 words). (15pts)

4. 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. (20pts)

References

- [1] "Arduino," [Online]. Available: <https://www.arduino.cc>
- [2] "TinkCAD", [Online]. Available: <https://www.tinkercad.com>

Table 1.1 Sample Program No. 1

```
// 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

//time_base define the running speed
//Set it to 1000 is in real time
//Set it to a smaller value for debugging
#define time_base 500

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);
}

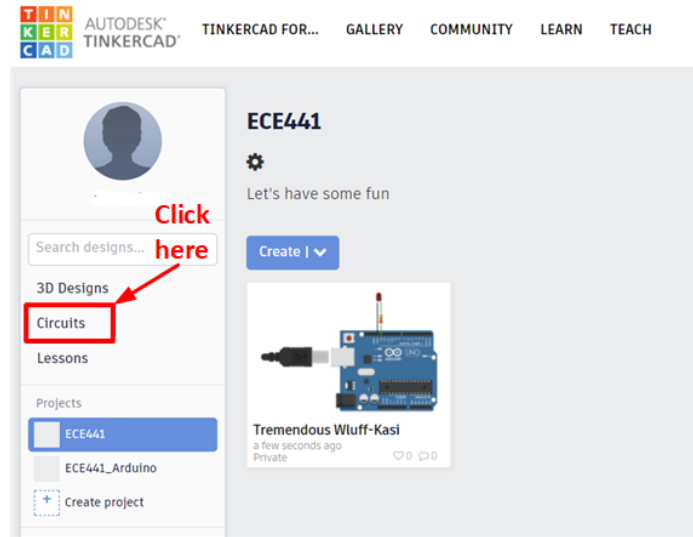
void ChangeLedValue(byte number)
{
    digitalWrite(r_ns, bitRead(number, 5));
    digitalWrite(y_ns, bitRead(number, 4));
    digitalWrite(g_ns, bitRead(number, 3));
    digitalWrite(r_ew, bitRead(number, 2));
    digitalWrite(y_ew, bitRead(number, 1));
    digitalWrite(g_ew, bitRead(number, 0));
}

void LightSequence()
{
    begining:
    ChangeLedValue(B001100);
    delay(time_base * 10);
    ChangeLedValue(B010100);
    delay(time_base * 1);
    ChangeLedValue(B100100);
    delay(time_base * 1);
    ChangeLedValue(B100001);
    delay(time_base * 5);
    ChangeLedValue(B100010);
    delay(time_base * 1);
    ChangeLedValue(B100100);
    delay(time_base * 1);
}

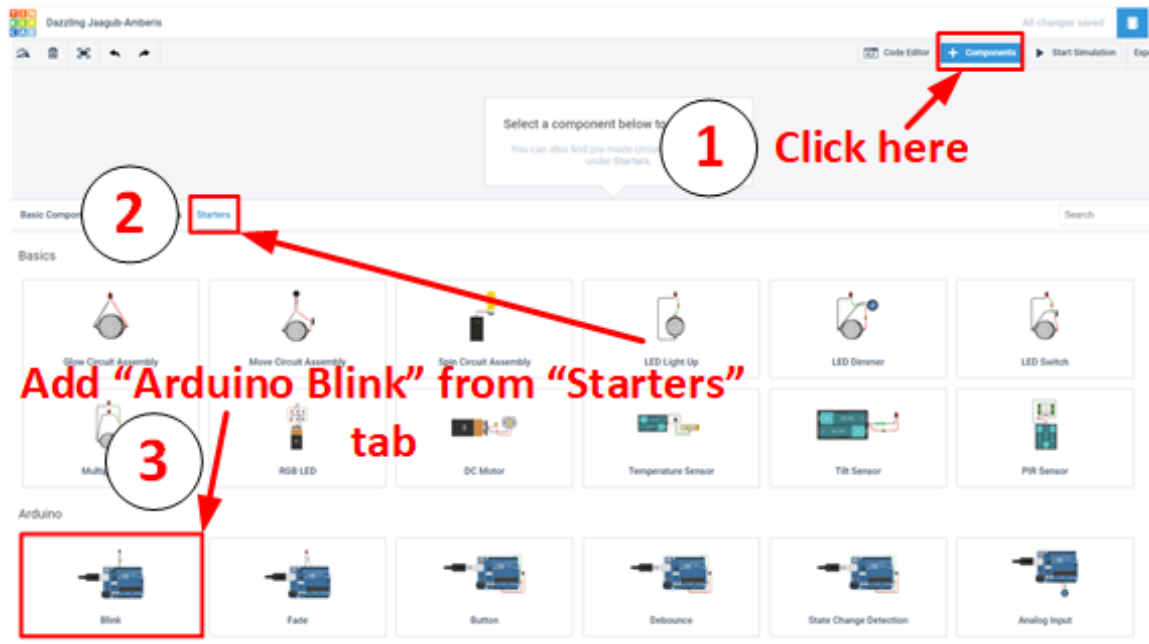
void loop()
{
    // put your main code here, to run repeatedly
    LightSequence();
}
```

Appendix A. Instruction for Simulation in TinkerCAD

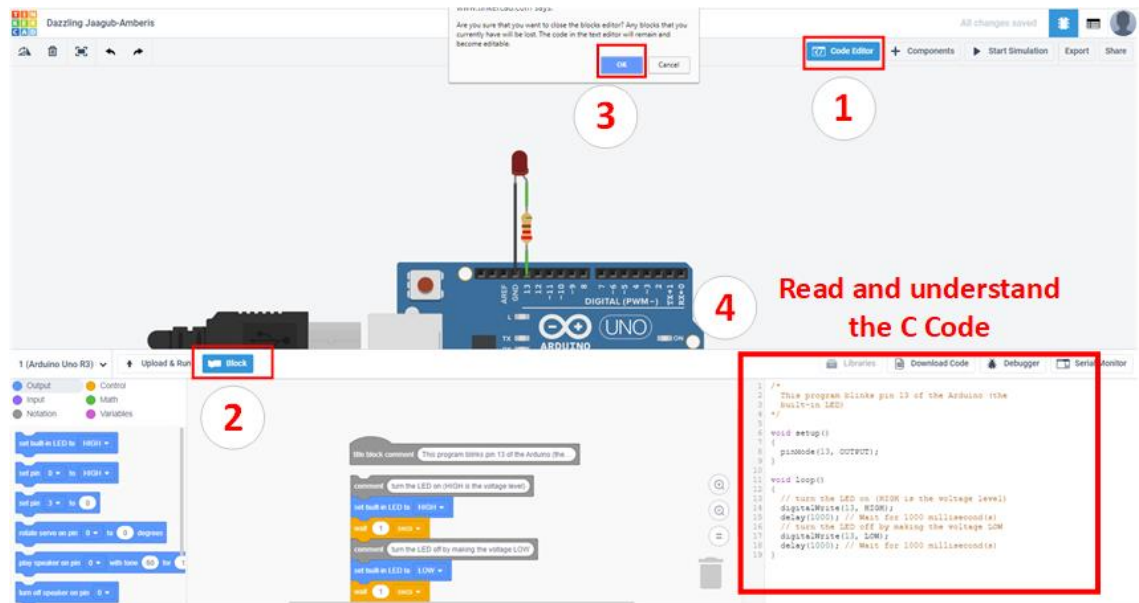
1. Go to the website: <https://www.tinkercad.com>
Register for Autodesk account using your Illinois Tech email address
Click "**Circuits**" on the left panel



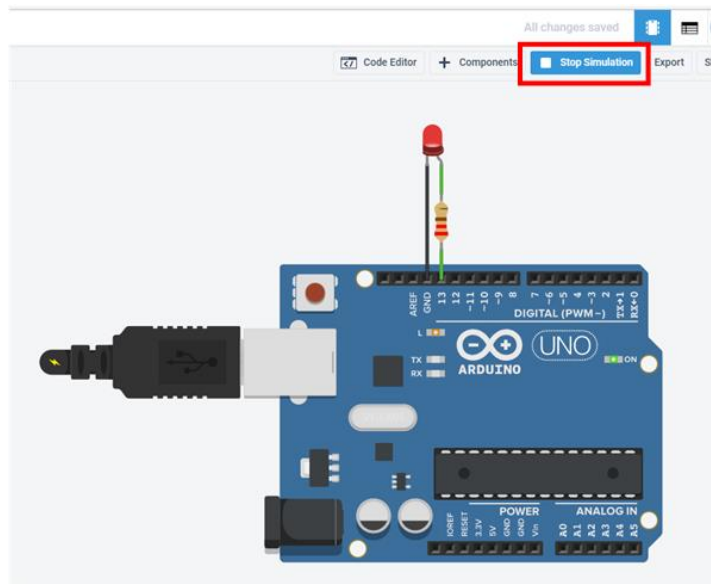
2. Click "**Create**" -> "**Circuit**" -> "**+ Component**" and add "**Arduino Blink**" from the "**Starters**" Tab.



- Go to **“Code Editor”** and disable **“Block”** function since we don’t want to program it using the block diagram. Read the C code to gain a better understanding of the program flow.



- Click **“Start Simulation”** and observe the result.

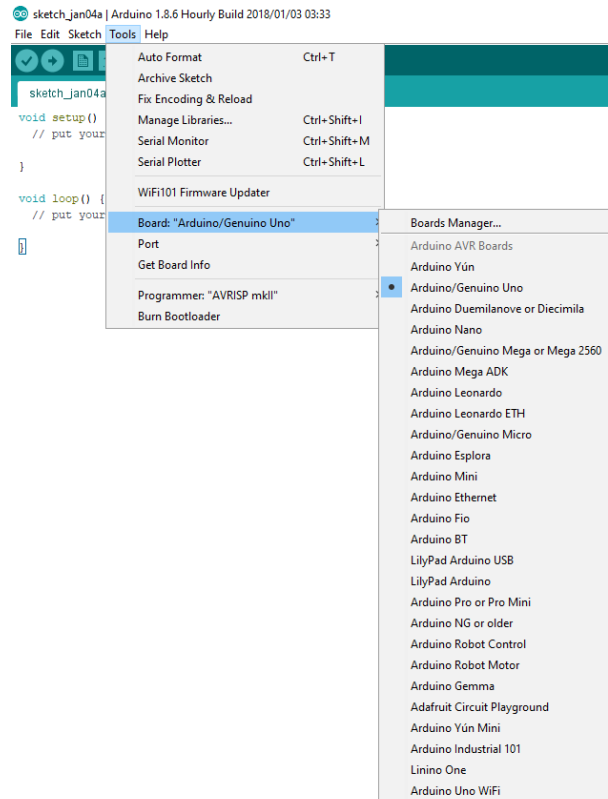


- Following the same procedure, simulate **“Button”**, **“Debounce”** from **“Starters”**. Practice with the tool to gain a better understanding of building peripheral circuit and programming the microcontroller.
- Refer to the **3.0 Background B-1 Simple Two-Way Intersection** and build the circuit in Tinkercad. Use the code from Table 1.1 and simulate the circuit. Show your result to your TA in the beginning of the lab.

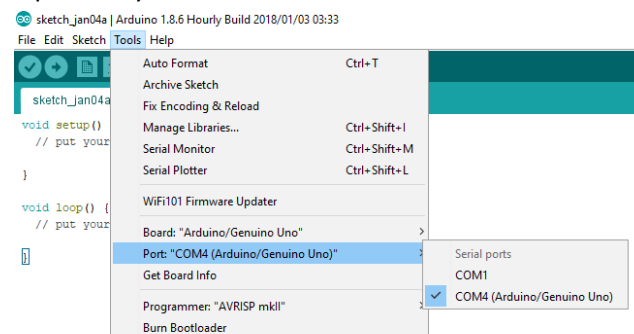
Appendix B. Load compiled code into your Arduino UNO device




1. Click the icon to open **Arduino IDE**
2. Set the proper development board (Board: "Arduino/Genuino Uno")



3. Select the proper COM port for your board



4. Type the code from the Table into IDE. Compile your code with the button  and upload it to the Arduino UNO using button 