University of Dhaka

Department of Computer Science and Engineering

CSE-3116, Microcontroller Lab

3rd Year 1st Semester

Session: 2017-18

Experiment Number: 07

Name of the Experiment:

Simple one way traffic control using microcontroller.

Submitted by:
01. Amit Roy, Roll: 40
02. Tauhid Tanjim, Roll: 58
03. Esha Ferdous, Roll: 64

Submitted to:
1. Dr. Shabbir Ahmed, Professor, Dept. of CSE, DU
2. Dr. Sazzad Mohammad Samaun Imran, Associate Professor, Dept. of EEE, DU

Date of performance: 15 March, 2018.

Date of submission: 29 March, 2018.

Experiment No. : 07

Experiment Name:
Simple one way traffic control using microcontroller.

Objectives:
In this experiment we will design a microcontroller based system to control simple one way traffic. The objective of this experiment includes:

- Using IR sensor module to detect the existence of an object or vehicle and then control the traffic system of an one way road using green and red LEDs.
- Using a LCD to show from which direction of the one way road the traffic is allowed.

Microcontroller:

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are "embedded" inside some other device to control the features or actions of the device. Another name for a microcontroller, therefore, is "embedded controller." Microcontrollers are mostly designed for embedded applications and are heavily used in automatically controlled electronic devices such as cellphones, cameras, microwave ovens, washing machines, etc.

In our experiment we have used Arduino Uno R3. It is a microcontroller board based on a removable, dual-inline-package (DIP) ATmega328 AVR microcontroller. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs).

Arduino:

Arduino is an open-source microcontroller platform, 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. Since the boot loader has been programmed into the on board micro-controller, we can compile and upload sketches with only one software environment, that is, the Arduino Integrated Development Environment (IDE).

LED:

An LED is a small light (it stands for "light emitting diode") that works with relatively little power.

Current Limiting Resistor:

We should connect a current-limiting resistor when an LED is used or else the LED can become burned due to excessive current. In this experiment, the operating voltage of the LED is between 1.5V and 2.0V and the operating current is between 10mA and 20mA. The arduino Uno board can supply 5V power, the LED we choose works at 1.7V, 15mA. The current-limiting resistance equals total voltage subtracted by LED voltage, then divided by current. In this case, that would be (5-1.7)/0.015. Thus, the current-limiting resistance equals 220Ω.

Connecting an LED:

LEDs have polarity, which means they will only light up if you orient the legs properly. The long leg is typically positive, and should connect to a digital pin on the Arduino board. The short leg goes to GND; the bulb of the LED will also typically have a flat edge on this side.

Code:

To blink the LED takes only a few lines of code. The first thing we do is define a variable that will hold the number of the pin that the LED is connected to. We don't have to do this (we could just use the pin number throughout the code) but it makes it easier to change to a different pin. We use an integer variable (called an int).

```
int ledPin = 13;
```

We can also define an array if there are multiple LEDs. For example:

```
int ledPins[] = {0,1,2,3,4,5,6,7,8,9}
```

The second thing we need to do is configure as an output the pin connected to the LED. We do this with a call to the pinMode() function, inside of the sketch's setup() function:

```
void setup()
{
        pinMode(ledPin, OUTPUT);
}
```

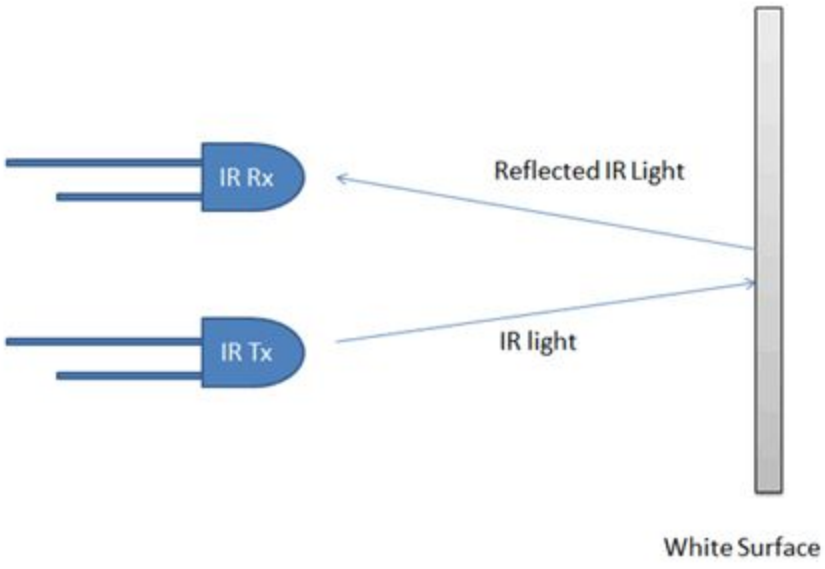
Finally, we have to turn the LED on and off with the sketch's loop() function. We do this with two calls to the digitalWrite() function, one with HIGH to turn the LED on and one with LOW to turn the LED off. If we simply alternated calls to these two functions, the LED would turn on and off too quickly for us to see, so we add two calls to delay() to slow things down. The delay function works with milliseconds, so we pass it 1000 to pause for a second.

```
void loop()
{
        digitalWrite(ledPin, HIGH);
        delay(1000);
        digitalWrite(ledPin, LOW);
        delay(1000);
}
```

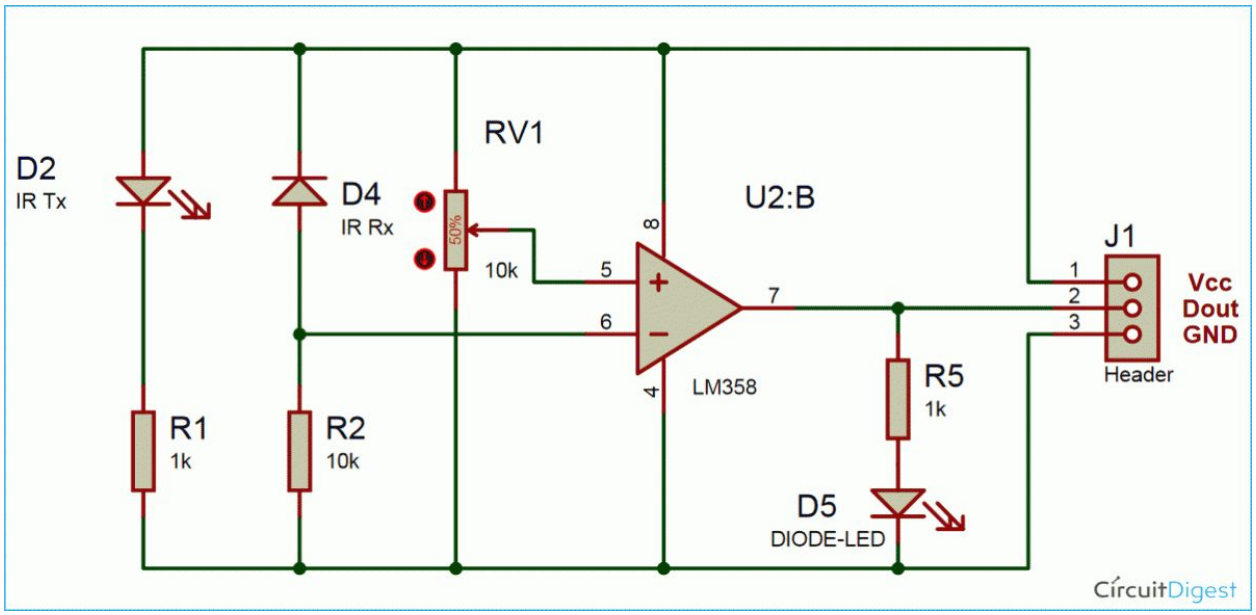
IR Module:

The IR sensor would observe an interruption and provide an input to the controller which would control the green and red LED on the both ends of the one way road depending on whether a traffic is entering or exiting the one way road and the current status of incoming and outgoing roads will be shown on a 16x2 LCD through the controller.

When any vehicle enters the one way road, the IR sensor for the incoming vehicles will get interrupted by the vehicle and then the other IR sensor will not work because we have added a delay for a while.



In this section we have used two IR sensor modules which contain IR diodes, potentiometer, Comparator (Op-Amp) and LED's. Potentiometer is used for setting reference voltage at comparator's one terminal and IR sensors sense the object or person and provide a change in voltage at comparator's second terminal. Then comparator compares both voltages and generates a digital signal at output. Here in this circuit we have used two comparators for two sensors. LM358 is used as comparator. LM358 has inbuilt two low noise Op-amp.



LCD Display:

LCD stands for Liquid Crystal Display. We will refer to it as either an LCD or simply a display.

In this experiment we used a LCD display to show when the LED brightness is intensifying or dimming as we press the push button.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A Read/Write (R/W) pin that selects reading mode or writing mode

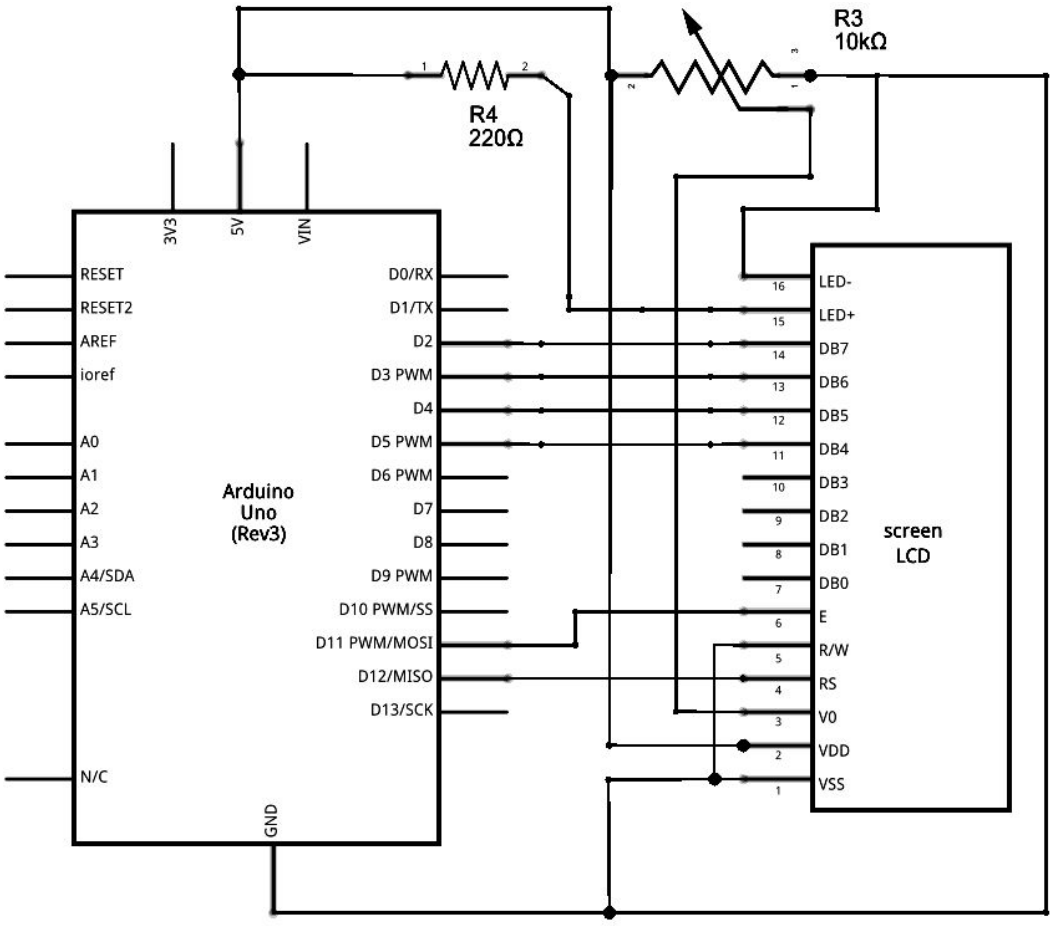
An Enable pin that enables writing to the registers

8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display constrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

Circuit:



To wire your LCD screen to your board, connect the following pins:

LCD RS pin to digital pin 12
LCD Enable pin to digital pin 11
LCD D4 pin to digital pin 5
LCD D5 pin to digital pin 4
LCD D6 pin to digital pin 3
LCD D7 pin to digital pin 2

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

Equipment:

1.A computer (Windows, Mac, or Linux).
2.An Arduino compatible microcontroller.
3. Arduino Software
4. A breadboard
5. A USB A-to-B cable
6. Male/Male, Male/Female Jumper Wires
7. IR Sensors
8. LEDs
9. Push Button
10. LCD Display
11. Potentiometer

Experiment:

Circuit Diagram:

Program Source Code:

Simple one way traffic control using microcontroller.

```
#include<LiquidCrystal.h>


const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int red1    = 6;
const int green1  = 7;
const int red2    = 8;
const int green2  = 9;



const int in = 14;
const int out = 19;



void OUT()
{
	lcd.clear();
	lcd.print("Lane1: OFF");
	lcd.setCursor(0,1);
	lcd.print("Lane2: ON");


	digitalWrite(green2,HIGH);
	digitalWrite(red2,LOW);
	digitalWrite(red1,HIGH);
	digitalWrite(green1,LOW);

	delay(3000);
}


void IN()
{

	lcd.clear();
	lcd.print("Lane1: ON");
	lcd.setCursor(0,1);
	lcd.print("Lane2: OFF");


	digitalWrite(green1,HIGH);
	digitalWrite(red1,LOW);
	digitalWrite(red2,HIGH);
	digitalWrite(green2,LOW);
```

```
	delay(3000);
}
void NEUTRAL()
{
	lcd.clear();
	lcd.print("Lane1: ON");
	lcd.setCursor(0,1);
	lcd.print("Lane2: ON");


	digitalWrite(green1,HIGH);
	digitalWrite(red1,LOW);
	digitalWrite(green2,HIGH);
	digitalWrite(red2,LOW);


	delay(100);
}


void setup()
{
	pinMode(in, INPUT);
	pinMode(out, INPUT);


	pinMode(red1,OUTPUT);
	pinMode(green1,OUTPUT);
	pinMode(red2,OUTPUT);
	pinMode(green2,OUTPUT);


	lcd.begin(16,2);
	lcd.clear();
	lcd.print(" TRAFFIC CONTROL");
	delay(2000);

	lcd.clear();
	lcd.print("Lane1: ON");
	lcd.setCursor(0,1);
	lcd.print("Lane2: ON");



	digitalWrite(green1,HIGH);
	digitalWrite(red1,LOW);
	digitalWrite(green2,HIGH);
	digitalWrite(red2,LOW);
	delay(1000);
```

```
}


void loop()
{
	if(digitalRead(in) && !digitalRead(out))
	{
		IN();
	}
	else if(!digitalRead(in) && digitalRead(out))
	{
		OUT();
	}
	else
	{
		NEUTRAL();
	}
}
```

Procedure:

Part1: Designing the circuit:

We designed the circuit by connecting two IR sensors, LED, LCD display, potentiometer and resistors as defined above. We used the IR sensor to detect the appearance of a vehicle on the one way road. The LEDs are controlled depending on whether the incoming and outgoing roads are busy or not and the status of the incoming and outgoing roads are shown on the display of the LCD.

Part 2: Connecting Arduino to the computer

We launched the Arduino IDE from the computer then created a new sketch. Then, we connected the Arduino UNO board with the computer through a USB A-to-B cable.

Part 3: Compiling and Uploading the sketch to the arduino

Now we need to write the source code for the experiment.

We included the LiquidCrystal.h library so that we can use the LCD display. Then we set up the pins of the LCD display as follows

RS to pin 12
EN to pin 11

D4 to pin 5
D5 to pin 4
D6 to pin 3
D7 to pin 2

We connected the LED red1, green1, red2, green2 pins to the digital pins 6,7,8 and 9 of the arduino respectively and the incoming road sensor with the pin no 14 (A0) and outgoing road sensor with the pin no 19(A5).

We declared three funcion IN(), OUT() and NEUTRAL().

When incoming road is busy and outgoing is free
LED green1 is HIGH
LED red1 is LOW
LED green2 is LOW
LED red2 is HIGH

When incoming road is free and outgoing is busy
LED green1 is LOW
LED red1 is HIGH
LED green2 is HIGH
LED red2 is LOW

When both the incoming and outgoing road is free
LED green1 is HIGH
LED red1 is LOW
LED green2 is HIGH
LED red2 is LOW

The status of each road is always shown on the LCD display.

Conclusion:

In this experiment we used IR sensor to detect the appearance of any object or vehicle in front of it. IR sensor is environment dependent. So, we need to control it by rotating the potentiometer of it. Besides, since two IR sensors are placed one after another so we used 3000 milliseconds delay so that when one IR sensor is detecting radiation, the other IR sensor is not be affected. Finally, we are successful designing a simple one way traffic controller using microcontroller and completed the experiment without any errors.