Lab Manual

CYSL – Embedded System Security Lab

Topic: Analog Voltage Control and
PWM Generator

Class: **BSCYS**

Semester: **IV**

Session: **Spring, 2022**

Instructor: Dr. Naveed Bhatti

Lab Instructor: Mr. Salman Ahmad

Air University Islamabad

FACULTY OF COMPUTING & ARTIFICAL INTELLIGENCE

Department of Cyber Security

Objectives

In this lab, we will perform:

- PWM generation using Arduino digital pin.
- how to dynamically make the LED fade in/fade out when turning the potentiometer knob
- how to control the brightness of LED with LDR

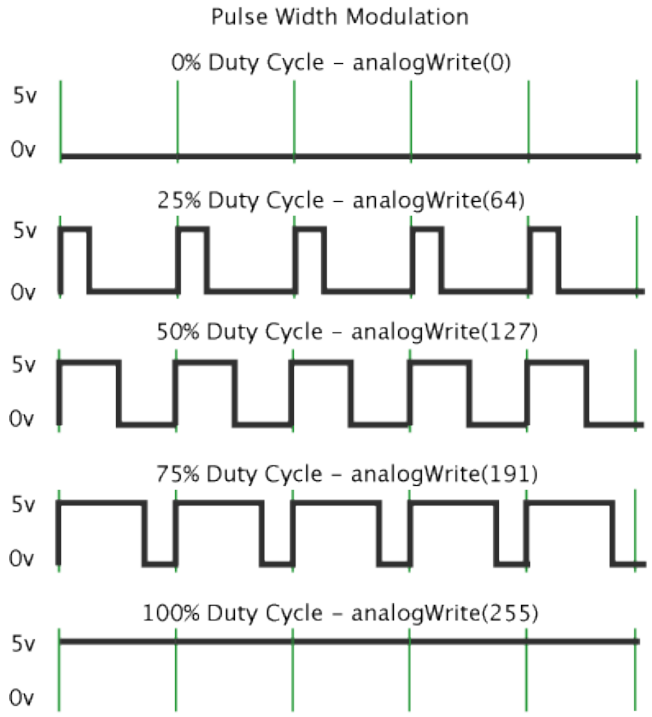
1. Description

In the scenarios where we want the analog outputs. As Arduino don't have analog output, so we use the pulse width modulation (PWM) phenomenon to generate it. Arduino's analog inputs (pins marked A0-A6) can detect a gradually changing electrical signal, and translates that signal into a number between 0 and 1023. Subsequently it reads an analog input pin, maps the result to a range from 0 to 255 and uses the result to set the PWM of an output pin.

2. PWM (Pulse Width Modulation)

Pulse Width Modulation is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between the full Vcc of the board (e.g. 5 V) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and Vcc controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



3. Task: LED brightness control using potentiometer/LDR

3.1 Components Required

1. Oscilloscope (Optional)
2. Arduino UNO
3. Breadboard
4. Resistors 470, 10 k ohm
5. LED
6. Potentiometer
7. LDR

3.2 Schematic Diagram

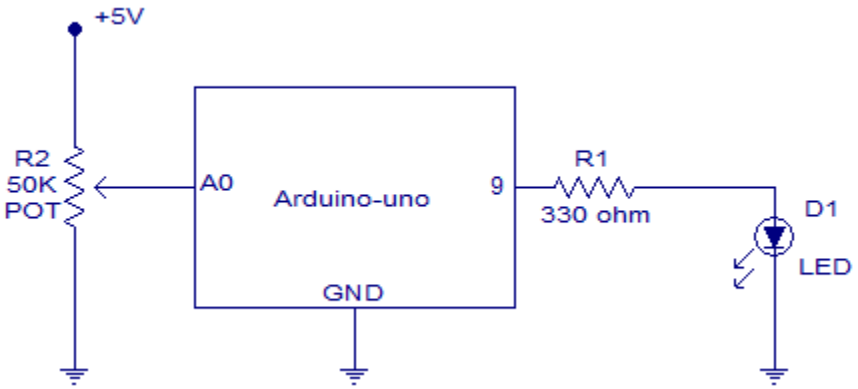


Fig 1: Schematic diagram of PWM control of LED brightness using potentiometer

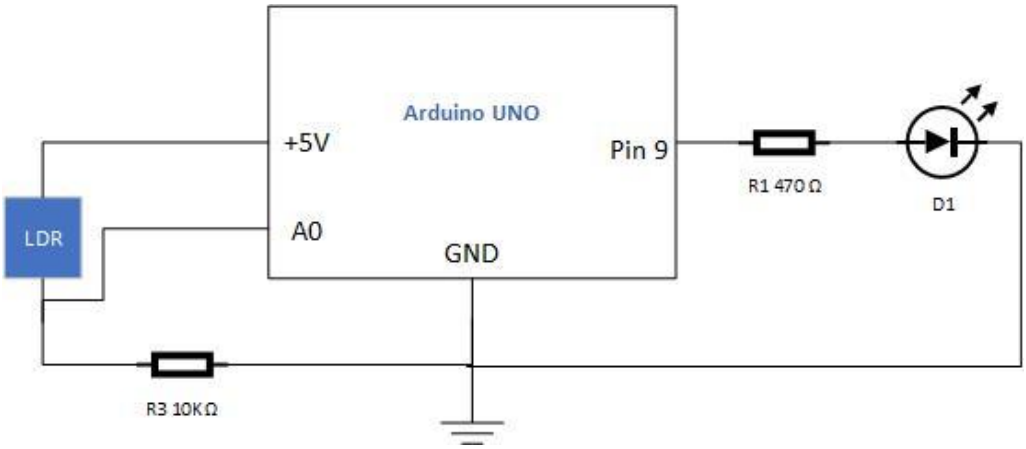


Fig 2: Schematic diagram of PWM control of LED brightness using LDR

3.3 Arduino Code:

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to
int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
```

```
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(20);
}
```

3.4 Procedure

1. Launch the Arduino IDE. Set the comm port
    - Tools > Port > Comm___ (It can be verified from the device manager)
2. Set the corresponding microcontroller board
    - Tools > Boards > Arduino AVR boards > Arduino UNO
3. Now connect your PC with Arduino. Then write your program in the IDE and Upload it.
4. For potentiometer. Connect the circuit as shown in the Fig 1. For LDR go to step 7.
    - Plug the 3 legs of the potentiometer to 3 different lines on the breadboard.
    - Connect the extreme left (or right) leg to GND.
    - Connect the other extreme leg to 5V on the Arduino.
    - Add a wire between the middle pin to the analog pin A0.
5. Now as we turn the knob of the potentiometer, the minimum knob position will correspond to the minimum brightness – LED turned off. And the maximum position will correspond to the maximum brightness – LED with full intensity.
6. Observe the output waveform on the oscilloscope.
7. Connect the circuit as shown in the Fig 2. LDR has two terminals

- Connect one end with 5V on the Arduino
- Connect other end with analog input A0 and to the ground via a 10 kΩ resistor.

8. Current passing through LDR increases/decreases depending on the amount of light thrown on it.
9. Lastly, open the serial monitor to observe your sensor values. It's likely that real world values will not extend all the way to 0 or all the way to 1023, depending on your lighting conditions. Feel free to adjust the 0-1023 range to your observed minimum and observed maximum in order to get the maximum brightness expression range on the LED.

4. Assignment 2:

Using the PWM phenomenon, control the buzzer tone (frequency) in accordance with some varying analog input values. Demonstrate the circuit and submit the code as an assignment.
Whereas syntax of tone() function is given below.

4.1 tone() Function:

It generates a square wave of the specified frequency on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Syntax:

```
tone(pin, frequency, duration) // here duration is optional.
```

pin: Arduino pin on which tone is generated.
frequency: frequency of the tone in hertz. Allowed data types: unsigned int.
duration: duration of the tone in milliseconds. Allowed data types: unsigned long.

5. Assignment 3:

Build a circuit that raises an alarm when there is a sufficiently dark in a room. Demonstrate the circuit by covering the photo-resistor to simulate darkness. Submit the code written.