Mini Project 2023

DECEMBER 2

EGR 102 Mini-Project

Authored by: Raymond Carroll

UK EGR 102
Foundations of Engineering Computing
First Year Engineering Program

Mini Project

Overview

This project allows for the easy classification of resistors. There is a servo that will use the attached hand to point to the resistors value. There is also a ARGB light that will be able to indicate if the resistor is within a 5% tolerance. The green and red button allow for easy interaction with the attached program, making it even easier for the end user. Another additional feature is the use of a buzzer, which creates audible queues for program interaction. This device meet or exceeded all requirements detailed in the project description.

Sensor Design

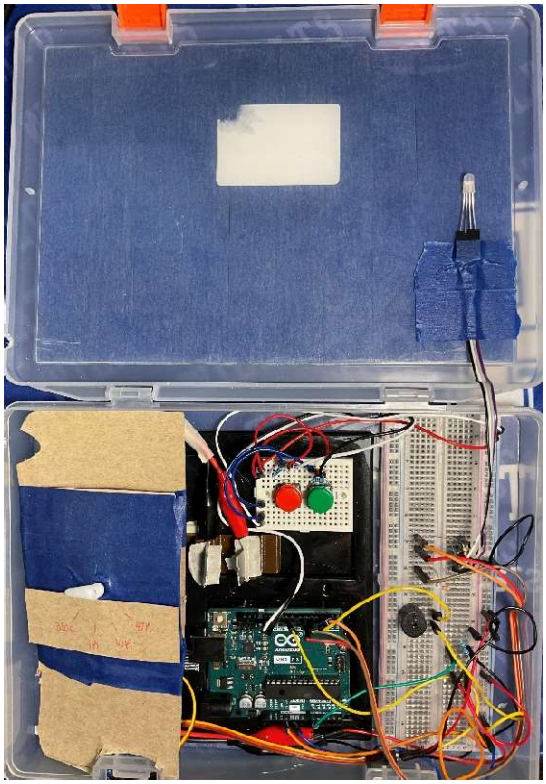

Figure 1 Picture of the project build

In this project, the containment of all the components is something that was considered. For this, I used a plastic box that I had laying around. I also wanted to make sure that the resistor reading poles were easily assessable. I also wanted to make sure that the resistor read button (green) and program exit (red) buttons were easily accessible, that is why the wires are routed elsewise.

As a part of the calculations for the variable (user-defined) resistor a voltage divider is used. This is a method that uses a known value resistor (10K Ω in our case) and the 5-volt output of the Arduino to calculate the variable resistor.

Circuit Design

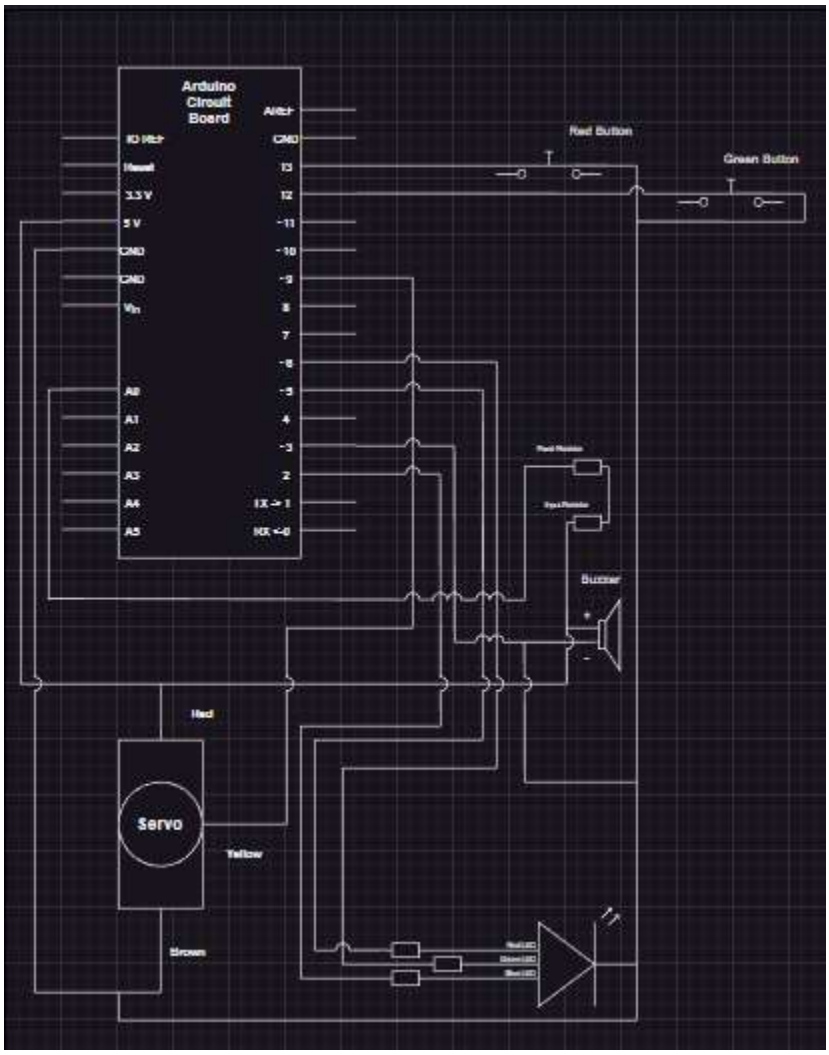

Figure 2 Circuit Design

Figure 2 details a schematic that illustrates all of the components within my project. There are a couple of things to note within the schematic. Firstly, a 10KΩ resistor is used for the fixed resistor. This is because this 10KΩ resistor is a 1% tolerance, meaning that the actual resistance from the resistor is within 1% of the nominal value (In this case, within 10Ω). Secondly, some items are connected to PWM pins. The definition of PWM is pulse width modulation, and this allows a digital pin to act like an analog pin in a regard. The servo is connected to D9, which is a PWM pin. The buzzer is also connected to a PWM pin, D3. Both devices require the pins that they are assigned to. The red LED is connected to a PWM pin but does not take advantage of the PWM functionality.

This also depicts the use of the voltage divider, which allows the measurement of the variable resistor's value.

Data Taken From Sensor

Sample	Average reading	Maximum Reading	Minimum Reading
330 Ohm	0.87 V	0.8553 V	0.8798 V
1K Ohm	2.5092 V	2.5220 V	2.5220 V
10K Ohm	4.5415 V	4.5455 V	4.5357 V
47K Ohm	4.8308 V	4.8338 V	4.8289 V

Table 1 Sensor Readings

This table depicts the voltages with each of the different resistor values. This information creates the if/elseif states that decided the resistor values.

Decision Tree

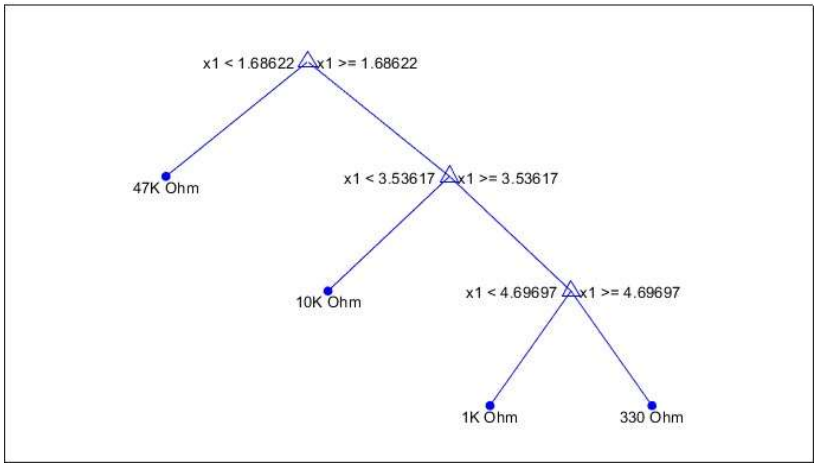

Figure 3 Decision Tree Image

This decision tree allowed me to make decisions based off the voltage values. This data was used to create the if/elseif statement that is used to decide what the resistor value is.

Final Performance

Resistor Value	Test 1	Test 2	Test 3	Test 4	Test 5
330 Ω	Fail	Pass	Fail	Fail	Fail
1K Ω	Fail	Pass	Pass	Pass	Pass
10K Ω	Pass	Pass	Pass	Pass	Pass
47K Ω	Pass	Pass	Pass	Pass	Pass

Table 2 Resistor Classification Results

The device was 100% accurate in identification of the resistors. The device also was able to identify the tolerance of the 10K Ω resistors and greater. The device had issues identifying the tolerance of 330 Ω & 1K Ω resistors. The pass rate of 1K Ω resistors was 80%, and the 330 Ω had a pass rate of 20%. This is to partly to blame Arduino's resolution of voltage.

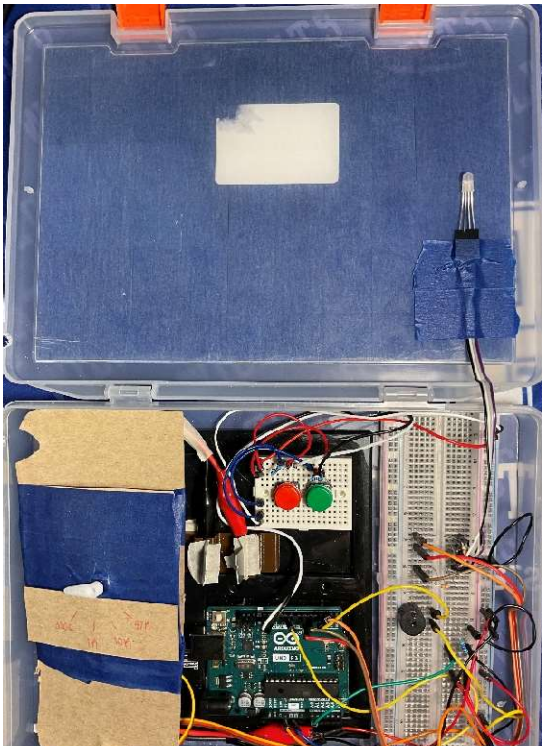

Figure 4 Output Picture

The output device has several features. The device uses an ARGB for a tolerance light, a buzzer for interaction with the device, and a servo for the output. The device also uses a green and red button for interaction with the program.

Appendix

Appendix A: Control Program Flow Chart

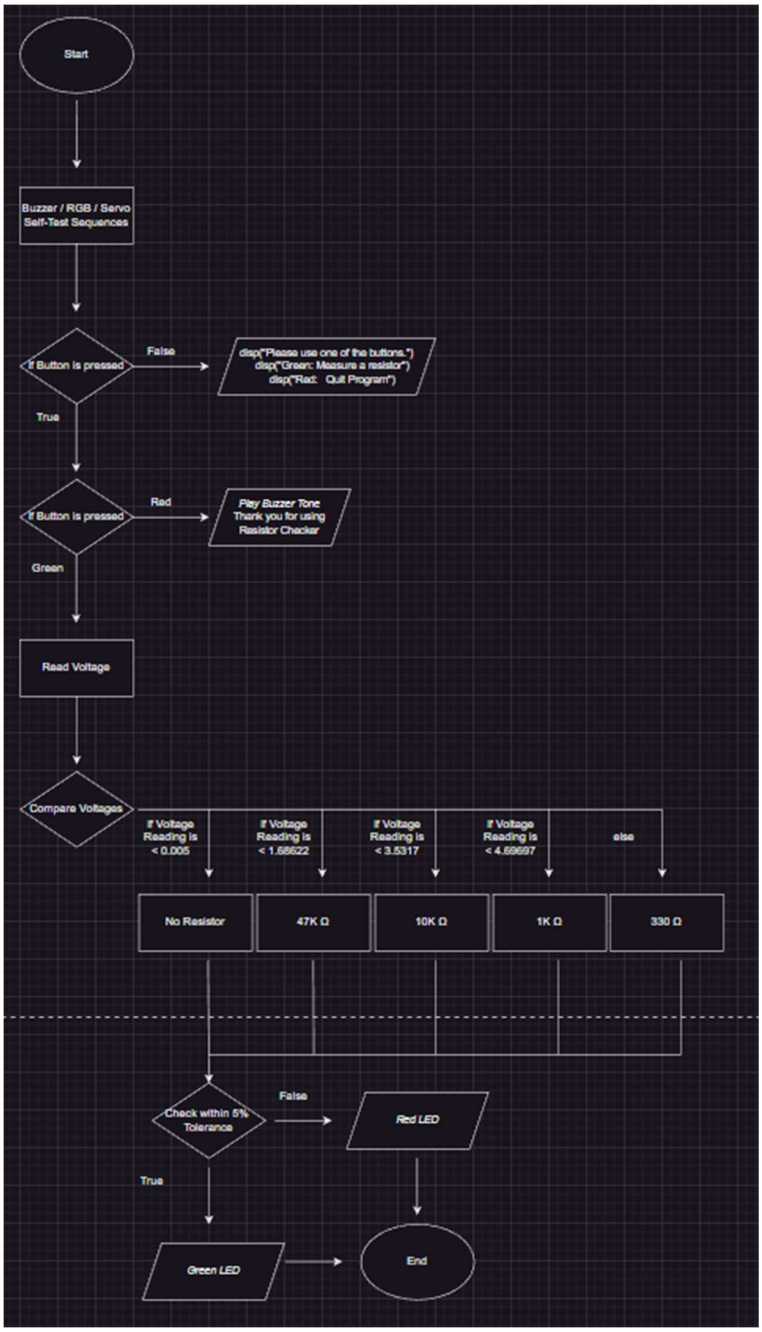

Figure 5 Flowchart

Appendix B: Control Program Code

```
%{
EGR 102 Mini Project - Meausure the value of a resistor
Authors:    Raymond Carroll
Assignment: EGR 102-016 Mini Project
Changed:    1 Novemeber 2023

Purpose:
  Measure the value of a reistor in the reistor holder. Also determine the
  tolerace of the resistor.
%}

% Clear Command Window
clc
%Clear all varables within the workspace
clear all

global a redLEDPin greenLEDPin blueLEDPin s voltageReading;

% Create a new Arduino object
a = arduino(); % Replace 'COM3' with your actual COM port

% Defines all pins
buzzerPin = 'D3';
redLEDPin = 'D5';
greenLEDPin = 'D6';
blueLEDPin = 'D2';
servoPin = 'D9';
VoltagePin = 'A0';
greenButton = 'D12';
redButton = 'D13';

%Sets resistor defualt tolerance (inculded the fixed resistor)
toleranceValue = 0.06;

%Define Servo
s = servo(a, servoPin, 'MinPulseDuration', 700*10^-6, 'MaxPulseDuration', 2300*10^-6);

%Please Wait Message
disp("Please wait, the system is starting.")

%Ensure that servo is at postion 0
writePosition(s, 0);

clc
disp("Now completeing: Buzzer Self-Test")
%Buzzer Self-Test
playTone(a, buzzerPin, 1200, .5);
pause(1);
playTone(a, buzzerPin, 2400, .5);
pause(1);
playTone(a, buzzerPin, 3600, .5);
pause(1);
playTone(a, buzzerPin, 4800, .5);
```

```
pause(1);

clc
disp("Now completeing: LED Self-Test")
%LED Self Test
RGB_LED_Handler(1, 0, 0, .5);
RGB_LED_Handler(0, 0, 0, 0);
RGB_LED_Handler(0, 1, 0, .5);
RGB_LED_Handler(0, 0, 0, 0);
RGB_LED_Handler(0, 0, 1, .5);
RGB_LED_Handler(0, 0, 0, 0);

clc
disp("Now completeing: Servo Self-Test")
%Servo Self Test
for servoPostion = 0:0.2:1
    if servoPostion <= 180
        RGB_LED_Handler(1, 1, 0, .025);
    else
        RGB_LED_Handler(0, 0, 0, 0);
    end
    writePosition(s, servoPostion);
    currentServoPostion = readPosition(s);
    currentServoPostion = currentServoPostion * 180;
    pause(.05);
    RGB_LED_Handler(0, 0, 0, 0);
end

%Ensure that servo is at postion 0
writePosition(s, 0);

% Completes two more please wait flashes
RGB_LED_Handler(1, 1, 0, .025);
RGB_LED_Handler(0, 0, 0, 0);
RGB_LED_Handler(1, 1, 0, .025);
RGB_LED_Handler(0, 0, 0, 0);

clc
buttonActivated = 'False';
while buttonActivated == 'False'
    clc
    disp("Please use one of the buttons.")
    disp("Green: Measure a resistor")
    disp("Red:   Quit Program")
    if readDigitalPin(a, greenButton) == 0
        clc
        playTone(a, buzzerPin, 1200, 1);
        voltageReading = readVoltage(a, VoltagePin);
        resistorType = ResistorClassifer(voltageReading);
        resistorValue = (5 - voltageReading)/(voltageReading / 10000);
        if resistorValue < 10
            continue
        elseif resistorValue >= (resistorType * (1 - 0.06)) && resistorValue <=
(resistorType * (1 + 0.06))
            RGB_LED_Handler(0, 1, 0, 10);
        else
```

```
            RGB_LED_Handler(1, 0, 0, 10);
        end
        RGB_LED_Handler(0, 0, 0, 0);
        writePosition(s, 0);

    elseif readDigitalPin(a, redButton) == 0
        playTone(a, buzzerPin, 3600, 1);
        clc
        disp('Thank you for using Resistor Check')
        break
    end
end

function [resistorClassification] = ResistorClassifer(voltageReading)
global s
if voltageReading <= 0.005
    resistorClassification = '0';
    writePosition(s, 0);
    disp("There has been an error, or there is no resistor")
elseif voltageReading <= 1.68622
    resistorClassification = 47000;
    writePosition(s, .8);
elseif voltageReading <= 3.5317
    resistorClassification = 10000;
    writePosition(s, .6);
elseif voltageReading <= 4.69697
    resistorClassification = 1000;
    writePosition(s, .4);
elseif voltageReading <= 5
    resistorClassification = 330;
    writePosition(s, .2);
else
    resistorClassification = 'There is an error';
end
end

function RGB_LED_Handler(redState, greenState, blueState, pauseDuration)
    global a redLEDPin greenLEDPin blueLEDPin;
    writeDigitalPin(a, redLEDPin, redState);
    writeDigitalPin(a, greenLEDPin, greenState);
    writeDigitalPin(a, blueLEDPin, blueState);
    pause(pauseDuration);
end
+
`
```