Dr. Lei Lei
College of Science and Engineering
Cairns, Queensland, 4878 Australia

Discipline of Electrical & Computer Engineering

ASSESSMENT COVER SHEET

Student details

Surname(s)	Given names	Student's I.D.
Hiyas Jr.	Bienvenido	13824819

Subject details

Subject code	EE3901

Subject name	Sensor Technologies

Title of the work	Laboratory work 1

Name of Coordinator	Dr. Lei Lei

Due Date: 9:00AM Tuesday of Week 5	Date submitted

Student's statement:

I/We certify that :

- I have not plagiarized the work of others;
- I have not participated in unauthorized collusion when preparing this work.

Signature(s) …Bienvenido Hiyas Jr.……
……

EE5901 Sensor Technologies
Lab 1: Potentiometer - Report

Bienvenido Hiyas Jr. (13824819)

I. Introduction

THIS document is a report about the practical experiment of Potentiometer for EE5901 Sensor Technologies using Arduino IDE. The Arduino Integrated Development Environment (IDE) is a cross-platform application that is written in functions from C and C++ and is used to write and upload programs to Arduino compatible boards. Furthermore, Arduino IDE supplies a software library from the Wiring project, which provides many common codes, input and output procedures.

This experiment uses connectors, breadboard, 220 ohm Resistor, LED , a Potentiometer: DFRobot analog slide position sensor, Arduino Board and IDE, multimeter, a laptop and USB connector. The outcome for this practical is to familiarize the students in using Arduino by programming the Arduino board to control the flashing frequency and brightness of an LED using a Potentiometer and display it on the Arduino Serial Monitor using the sample codes provided in the Aurduino software. The main task of this experiment is to familiarize the sample codes of the Arduino software and modify it to meet the desired outcome of this practical experiment.

II. Methods

The goal for Task 1 is to use the potentiometer to control the flashing frequency of internal LED in the Arduino board. The hardware used are Arduino Board (GeekCreit), Potentiometer, USB connector, connectors and Laptop. The 5V pin of the potentiometer is connected to the 5V power pin of the Arduino Board using a connector. The ground pins are also connected and the Signal pin of the potentiometer was connected to the A0 pin on the Analog input of the Arduino board. The setup is shown in Figure 1 below.

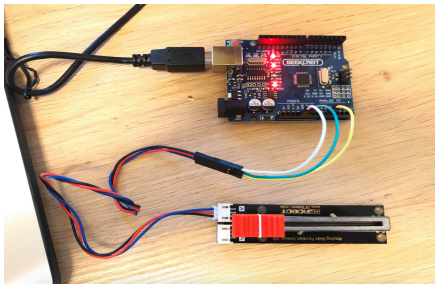

Figure 1: Actual setup of Potentiometer and Arduino Board

Then the Arduino board was connected to the Laptop using the USB connector and opened the Arduino software. The sample codes in "AnalogInput" file in the Arduino software was used to load the program and control the flashing frequency of the Internal LED of the Arduino board. The main task is to observe the flashing frequency as the Potentiometer was slide from left to right or right to left and understand the codes in "AnalogInput" sample code as shown on Figure 2 below.

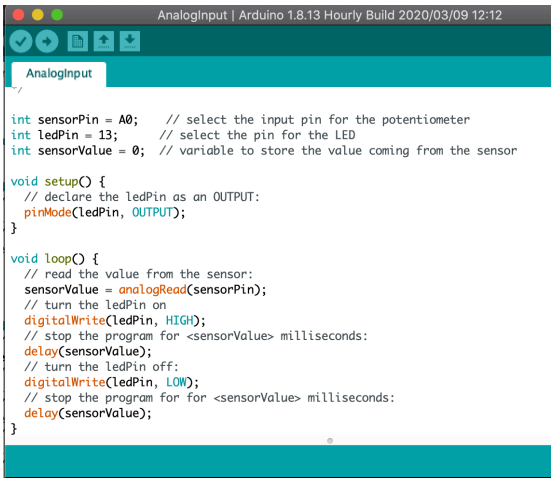

Figure 2: AnalogInput Sample codes

The goal for Task 2 is to build a series resistor and LED circuit and control the flashing frequency of the LED using the Potentiometer. The LED and resistor on the breadboard were just added to the hardware setup in Task 1 as shown in Figure 3 below.

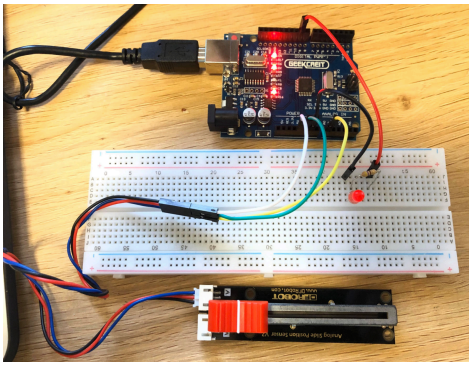

Figure 3: Arduino board and Potentiometer with resistor LED circuit.

The LED was connected to the digital pin 9 hence, the value in code ***int ledPin*** was changed from **13** to **9**.

In order to have a voltage value equivalent to the sensorValue, the codes below were added to the AnalogInput sample codes.

// sensorValue in Voltage

float voltageValue = (sensorValue / 1024)/0.2;

In order to display the sensorValue and the voltageValue to the Arduino Serial Monitor, the codes below were added.

// setting the baud rate to 9600
Serial.begin(9600);
// print the results to the Serial Monitor:
Serial.print("sensor = ");
Serial.print(sensorValue, 0);
Serial.print("\t voltage = ");
Serial.print(voltageValue);
Serial.println("V");

Figure 4: Revised AnalogInput codes.
Note: File name AnalogInput was renamed to AnalogInputBien

The goal of Task 3 is to control the intensity of brightness of the LED using the Potentiometer using the setup and circuit built in the breadboard in Task 2. The serial port on the Arduino was used to communicate with a computer via USB, and send the value contained in the variable "sensorValue" serially via the USB plug. Then the voltage drop across the resistor was measured using a multimeter.

III. Discussion

Task 1 is very simple. You just have to properly connect the potentiometer pins to the Arduino board as shown in Figure 1 and load the sample code "AnalogInput". As you slide the potentiometer from left to right or right to left, you can observe that the frequency of the flashing of the internal LED of the Arduino board varies. However, you cannot see the output in Serial Monitor because there are no print codes available.

Task 2 have the same setup as Task 1, except that in Task 2, an external LED was used and observed its flashing frequency. As we slide the Potentiometer, a sensorValue is generated. A sensorValue of numerical 0 corresponds to 0 Volts and 1023 to 5Volts. Therefore, a value of 4V corresponds to a numeric value of: 4/5 = x/1024, x = 4*1024/5 = **819**.

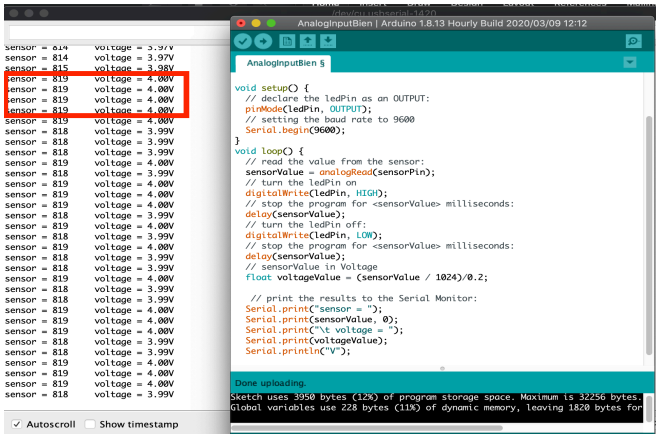

Figure 5: Serial Monitor showing voltageValue 4V equivalent to sensorValue 819

As the sensorValue or the voltageValue increases, the flashing frequency decreases. This is because of the codes in the "AnalogInputBien" program. The program starts in a loop with the code: **sensorValue = analogRead(sensorPin);** declaring that the sensorValue is equal to the value from the sensorPin (**A0**) which is varied by the Potentiometer as we turn the knob left to right. Then, the code: **digitalWrite(ledPin, HIGH)** turns the LED ON. The code: **delay(sensorValue);** delays the program equivalent to the sensorValue. The higher the value, the longer the LED is ON. Then the code: **digitalWrite(ledPin, LOW);** turns the LED OFF. The code: **delay(sensorValue);** delays the program equivalent to the sensorValue. The higher the value, the longer LED is OFF. Finally, the codes **Serial.print()** prints the values in the Serial Monitor. Since these codes are inside a loop (**void loop()**), these codes will run in a cycle and constantly printing result in the Serial Monitor.

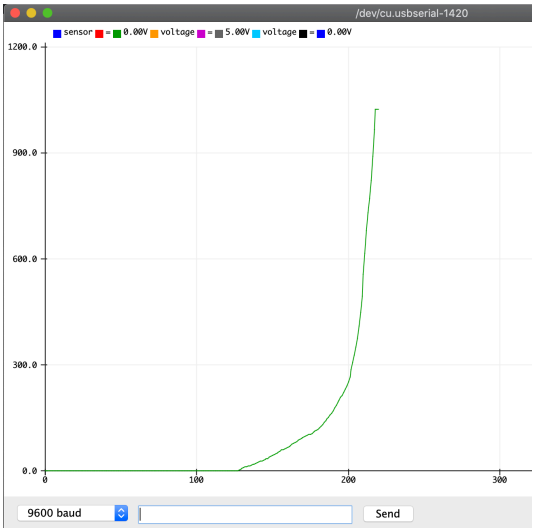

Figure 6: Arduino Serial Plotter result when sensor is increased from 0-1024 or 0-5V

As seen in the graph of Figure 6, when the sensorValue is 0, the duration of turning the LED ON and OFF is very fast, resulting more value fed to the serial plotter relative to time. As the sensorValue increases, the duration of ON and OFF decreases, resulting to less value fed to the serial plotter.

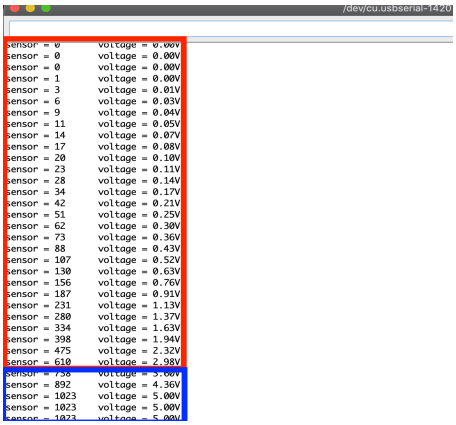

Figure 7: Serial Monitor showing the sensor and voltage value as knob of Potentiometer was slide from 0-1024 or 0-5V

As seen in Figure 7, there are many results or values fed to the Serial Monitor approximately 0-3V compared to the higher value or 4-5V. Each results or values represents every time the program runs or looped turning the LED On and OFF. This means that as we slide the Potentiometer approaching to 5V, the flashing of LED becomes very slow.

In Task 3, the same hardware and setup in Task 2 was used. However, a different sample sketch or code was loaded. The program used was "AnalogInOutSerial". This sketch reads the analog input pin, maps the result to a range from 0 to 255 and used the result to set the pulsewidth modulation (PWM) of an output pin. Also, it prints the results to the serial monitor.

As the sensor (sensorValue) increases, the output(outputValue) also increases the same as the outputVoltage, thus making the LED brighter. Arduino uses 10 bit digital converter which is equivalent to 1023 and 8-bit PWM timer which is 255. Therefore, we need to convert our sensorValues with 0-1023 to 0-255 in order to have a 0-5V output that changes the brightness of the LED. This is done using the codes in "AnalogInOutSerialBien" program.

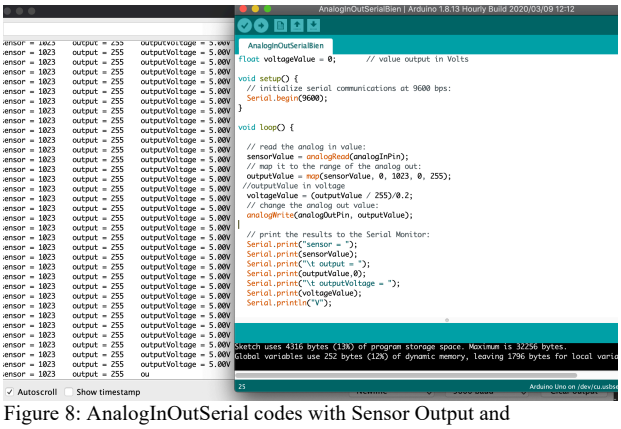

Figure 8: AnalogInOutSerial codes with Sensor Output and outputVoltage shown in Serial Monitor
Note: AnalogInOutSerial was renamed to AnalogInOutSerialBien

The program starts in a loop with the code: **sensorValue = analogRead(sensorPin);** declaring the sensorValue is equal to the value of the sensorPin (**A0**) which is varied by the Potentiometer. The code: **outputValue = map(sensorValue, 0, 1023, 0, 255)** map or scale the sensorValue from 0-1023 to 0-255 and store in the variable outputValue. The code: **voltageValue = (outputValue / 255)/0.2** just converts 0-255 to 0-5V range. The outputValue and voltageValue were declared as a "float" so that they can have decimal places. Then the code: **analogWrite(analogOutPin, outputValue);** writes the outputValue to analogOutPin which is set 9 to power the circuit and the LED. Finally, the codes **Serial.print()** prints the sensorValue, outputValue and voltageValue. Since these codes are in a loop, the program will run in cycle thus, continuously printing results in Serial Monitor.

As the Potentiometer was slide from left to right, the sensorValue increases the same as the outputValue and voltageValue. In result, it increased the brightness of the LED. As shown in Figure 8, the sensorValue 1023 was converted to outputValue of 255 which is equivalent to 5V and made the highest brightness of the LED.

Task 3					
Sensor	output	outputvoltage(V)	Vresistor(V)	VLED(V)	CurrentLED(ma)
0	0	0	0	0	0
211	52	1.02	0.608	0.412	2.76
440	109	2.15	1.27	0.88	5.77
615	153	3	1.79	1.21	8.14
822	204	4	2.3	1.7	10.45
1023	255	5	2.99	2.01	13.6

Figure 9: Table showing the relationship between the sensor, output, outputVoltage, the Voltage of the Resistor and the Voltge and Current of the LED

As shown in the table in Figure 9 above, as the sensor increases, outputvoltage also increases the same as the voltage in the resistor. Given outputvoltage(V) as the supply voltage and measuring the voltage across the Resistor (Vresistor(V)), We can compute the Voltage drop across the LED by using the formula $V_{LED} = V_s - V_R$. Where V_{LED} is the voltage across LED, Vs is the outputvoltage(V) and V_R is the Vresistor(V). Notice that as the supply voltage increases, the Voltage across the resistor and LED also increases. The Resistor limited the current to the LED ensuring that the LED had the Maximum voltage of only 2V and 13.6mA.

IV. Conclusion

In summary, the resistor in circuit plays an important role to limit and keep the LED from bursting because of the excessive current it may have. In addition, Arduino is very useful because it can control the output of the digital pin from the analog data from the potentiometer or any sensors that has variable resistance. It uses 8-bit Pulse Width Modulation timer so it is important to map the input to 255 for getting analog results with digital means. We can also use Arduino to control the flashing frequency or the brightness of the LED and other devices. Lastly, we can use Arduino to process data from sensors that can be used in IoT systems.

V. Skill Questions

1) What is the purpose of the resistor in series with the LED? How will it affect the brightness of the LED? How to derive a proper value for the resistance?

Ans: The purpose of the resistor in series with the LED is to regulate a proper flow of current through the LED so that, it will not blow off because the Arduino supplies 5V and most LED perates in 2-3V. When the potentiometer reaches to its lowest resistance the current will be its maximum or very high for the LED to handle and the fixed resistor 220 ohms is necessary so that the lowest resistance in the circuit is still 220 ohms. The brightness of the LED only varies with the potentiometer. The resistor 220 ohms only ensures that there is no too much current produces to the LED . The formula for the value of the resistor is: $R= (V_s-V_{LED})/I_{LED}$ where: R is the Resistance in Ohms, V_s is the supply voltage in (V), V_{LED} is the voltage across the LED in (V) and I_{LED} is the current across the LED. Referring to Figure 9 data, the value of the resistance is calculated below:

$R= (V_s-V_{LED})/I_{LED}$ = (5V – 2.01V)/13.6mA

R = 219.85 = **220 ohms**

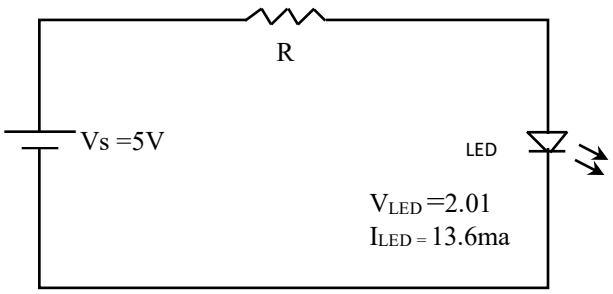

Figure 10: Resistor LED Circuit Diagram

2) Write a line of code that is equivalent to outputValue = map(sensorValue, 0, 1023, 0, 255).

Ans: By using ratio and proportion, we can have the outputValue equivalent to the map() function using the code below. Hint: You need to declare sensorValue as a float variable in order to print the outputValue in the Serial Monitor.

outputValue=(sensorValue/1023)*255

VI. Appendix

A. Task 1 and Task 2

```
//Replacing ledPin from Arduino pin 13 to 9
int ledPin = 9;
 // setting the baud rate to 9600
 Serial.begin(9600);

//Declaring vaiable float voltageValue to have vale with decimal places.
float voltageValue = 0;
// sensorValue in Voltage
 voltageValue = (sensorValue / 1024)/0.2;

 // print the results to the Serial Monitor:
 Serial.print("sensor = ");
 Serial.print(sensorValue, 0);
 Serial.print("\t voltage = ");
 Serial.print(voltageValue);
 Serial.println("V");
```

B. Task 3

```
//Declaring voltageValue as float to have value with decimal places.
float voltageValue = 0;      // value output in Volts
//outputValue in voltage
 voltageValue = (outputValue / 255)/0.2;

//print voltageValue to the Serial Monitor
 Serial.print(voltageValue);
 Serial.println("V");
```

VII. References

[1] SM, “Analog Input” Online Tutorial, Sep. 28, 2015. [Online]. Available: https://www.arduino.cc/en/Tutorial/AnalogInput

[2] SM, “Analog In, Out Serial” Online Tutorial, Sep. 28, 2015. [Online]. Available: https://www.arduino.cc/en/Tutorial/AnalogInOutSerial