**COMSATS University**

**Islamabad**



**Lab Report # 04**

**Real Time Embedded Systems**

**(EEE-446)**

|  |
| --- |
| **Introduction and implementation of ADC and DAC with Arduino** |

**Submitted By:**

**Arwa Aamir (FA16-EEE-002)**

**Submitted To:**

**Dr. Ahsen Malik**

**Lab # 04**

**Introduction and implementation of ADC and DAC with Arduino**

## Objectives

* Understand Analog to Digital and Digital to Analog Converters.
* Implementing Analog to Digital Converter and Digital to Analog Converters.

**Tools**

* Arduino
* Proteus ISIS

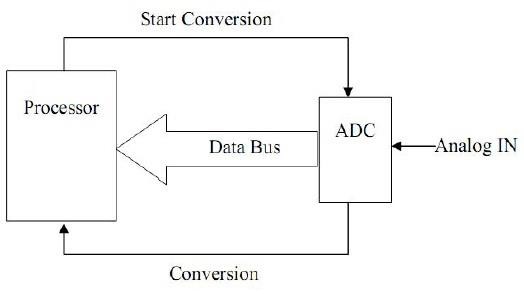
## Pre Lab

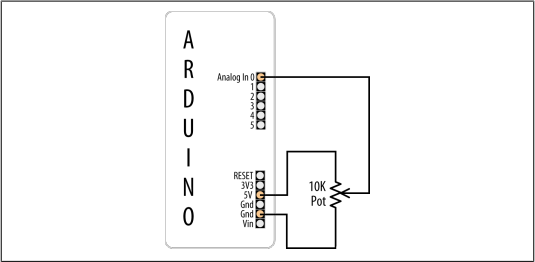
Please read the theoretical background of the using DAC in our lab.

## Analog to Digital Converter

In almost all digital systems there is a frequent need to convert analog signals generated by analog devices such as microphone, sensors and potentiometers into digital values that can be stored and processed. To serve this purpose analog to digital converters are used.

There are various analog to digital convertors: high speed ones, precise ones and economical ones and so on. Most of them can be directly connected to microcontroller as shown below.





Connecting a potentiometer to Arduino

**In-Lab Task 1:**

**Interface variable resistor at analog pin of Arduino, on virtual terminal please show it’s equivalent digital value, percentage out of max value and equivalent analog value received.**

|  |  |
| --- | --- |
| **ARDUINO IDE CODE** | **PROTEUS SCHEMATIC** |
| void setup() {  Serial.begin(9600);  pinMode(A0,INPUT);  pinMode(2,OUTPUT);  }  void loop() {  int a=analogRead(A0);  float vol=a\*0.004875;  float per=map(a,0,1023,0,100);  Serial.print("Volatge=");  Serial.println(vol);  Serial.println("Percentage=");  Serial.println(per);  Serial.println("Digital Value");  Serial.println(a);  if(per>=80)  {  digitalWrite(2,HIGH);  }  else  {  digitalWrite(2,LOW);    }  delay(1000);  } |  |

**In-Lab Task 2:**

**Interface 24Volts DC source with Arduino analog pin, use voltage divider circuit to calculate the identified voltage when varied from 0V-24V.**

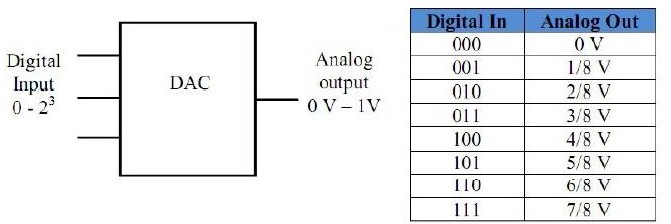
|  |  |
| --- | --- |
| **ARDUINO IDE CODE** | **PROTEUS SCHEMATIC** |
| void setup() {  Serial.begin(9600);  pinMode(A0,INPUT);  pinMode(2,OUTPUT);  }  void loop() {  // put your main code here, to run repeatedly:  int a=analogRead(A0);  float vol=a\*0.004875;  float volvol=4.8\*vol;  float per=map(a,0,1023,0,100);  Serial.print("Voltage=");  Serial.println(volvol);  Serial.println("Percentage=");  Serial.println(per);  Serial.println("Digital Value");  Serial.println(a);  if(per>=80)  {  digitalWrite(2,HIGH);    }  else  {  digitalWrite(2,LOW);    }  delay(1000);    } |  |

## Digital to Analog Convertors

In digital systems data can be processed only if it is represented in digital format (0,1) while the majority of real world applications require analog signals. This leads to the conversion between analog and digital formats. For example most audio signals are stored in digital form and in order to be heard through signal they must be converted to analog signal.

Digital to analog converters are an interface between the abstract digital world and analog real life. Usually DACs are used to convert 8 bit digital data into an analog signal. If greater precision is needed, chips with 12-bit, 14-bit or 16-bit data convertibility are available.

Lets assume that we have a 3-bit DAC that has three digital lines (D2,D1,D0) and has one output analog line. Assume that we assign references of analog output to Vref- = 0V and Vref+= 1V, then the input/output relation will be as shown below:



From the table we draw the following points:

* The three bit DAC has 2^3 = 8 possible combinations. If a converter has n input lines then it has 2^n input combinations.
* If the high and low references for analog output are V1 and V2 respectively then the change in the output corresponding to each increment of the n-bit digital input is (V2- V1)/(2^n).This value is known as resolution. In 3-bit case and V1,V2: 0 and 1 volts resolution becomes 1/8 volts.
* When MSB is 1 and other bits are zeros, the output analog is half of the full scale. In 3-bit DAC example it leads to 0.5 volts.
* For the maximum input, the output is equal to the value of full scale minus the value of resolution. In the 3-bit example it gives 1-1/8=7/8 V.

## DAC0808 8-bit Digital to Analog Conversion IC

The DAC0808 is an 8-bit digital-to-analog converter (DAC) featuring a full scale output current settling time of 150 ns while dissipating only 33 mW with ±5V supplies.

|  |  |
| --- | --- |
| DAC0808.jpg | application_cct.jpg |

## R2R Digital to Analog Convertor

A basic R-2R resistor ladder network is shown in Figure below. Bit D7 MSB (most significant bit) to Bit D0 LSB (least significant bit) are driven from digital logic gates. Ideally, the bits are switched between 0 volts (logic 0) and Vref (logic 1). The R-2R network causes the digital bits to be weighted in their contribution to the output voltage Vout.

For a digital value VAL, of a R-2R DAC of N bits of 0 V/Vref, the output voltage Vout is:

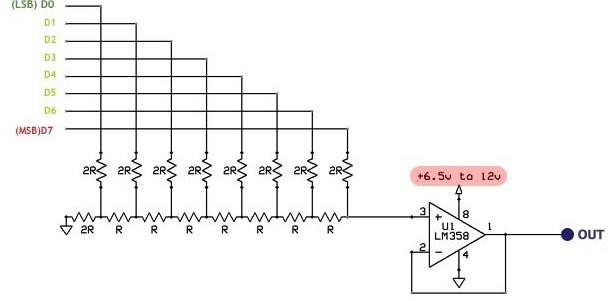
Vout = Vref × VAL / 2N

In the example shown, N = 8 and hence 2N = 256. With Vref = 5 V (typical TTL logic 1 voltage), VAL will vary between 00000000, VAL = 0 and 11111111, VAL = 255.

Minimum (single step) VAL = 1, we have

Vout = 5 × 1 / 256 = 0.0195 volts Maximum output (11111111 VAL = 255, we have

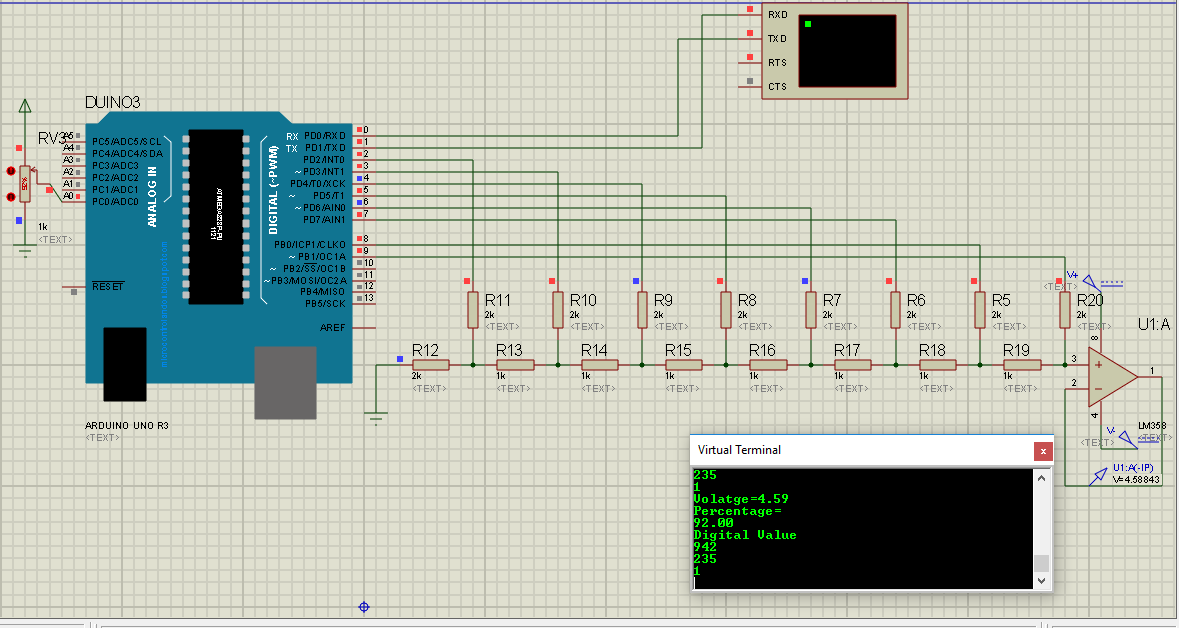
Vout = 5 × 255 / 256 = 4.980 volts

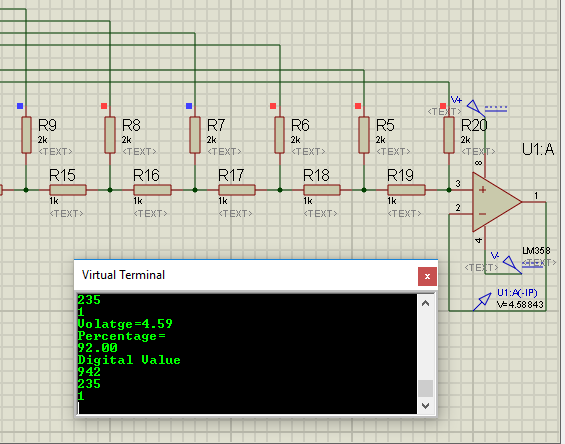
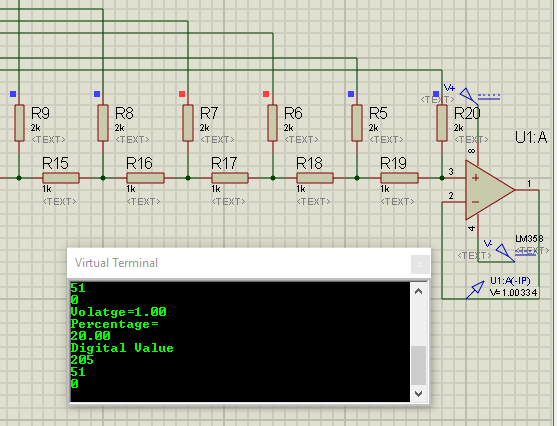


**In-Lab Task 3:**

**Use the same principle as before to identify 0V-5V from output of variable resistor, but the output of Arduino is digital 8-bit digital value, use R2R ladder as shown above to calculate the digital to analog output, which will eventually be the same as given at input. Please code and show circuit.**

|  |
| --- |
| **ARDUINO IDE CODE** |
| const int OUT[8]={2,3,4,5,6,7,8,9};  void setup() {  Serial.begin(9600);  pinMode(A0,INPUT);  pinMode(2,OUTPUT);  pinMode(3,OUTPUT);  pinMode(4,OUTPUT);  pinMode(5,OUTPUT);  pinMode(6,OUTPUT);  pinMode(7,OUTPUT);  pinMode(8,OUTPUT);  pinMode(9,OUTPUT);  }  void loop() {  int a=analogRead(A0);  float vol=a\*0.004875;  float per=map(a,0,1023,0,100);  Serial.print("Volatge=");  Serial.println(vol);  Serial.println("Percentage=");  Serial.println(per);  Serial.println("Digital Value");  Serial.println(a);  // converson of analgueRead value to 8-bit digital value for DAC  int nn=map(a,0,1023,0,256);  Serial.println(nn);  int dig\_bits[8];  int k=nn;  // counter for binary array  int i = 0;  while (k > 0) {    // storing remainder in binary array  dig\_bits[i] = k % 2;  k = k / 2;  i++;  }  Serial.println(dig\_bits[7]);      int bit\_status;  for (int j=0; j<8; j++)  {  bit\_status=dig\_bits[j];  digitalWrite(OUT[j],bit\_status);  }  delay(500);  } |

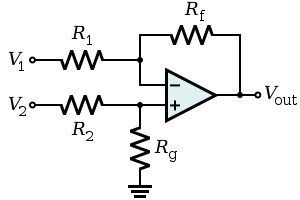


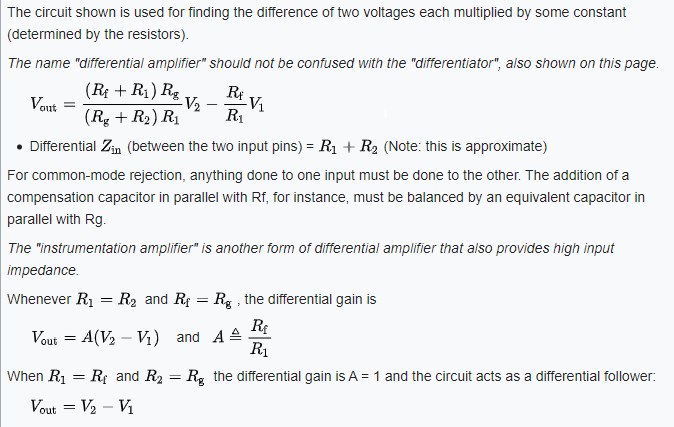


**Post-Lab Task 1:**

**Let’s suppose the input signal you want to monitor using circuit that uses Arduino is -5V to 5V peak to peak sinusoidal wave with 100Hz frequency, please identify the circuit and Arduino code which takes this input sinusoid and regenerates this signal again at the out which can be verified on oscilloscope.**

**Design of a Level Shifter using Op-Amp in Differential Amplifier Configuration:**





So, to make the ±5Vp-p sine wave purely positive we add -5VDC to it using differential op amp.

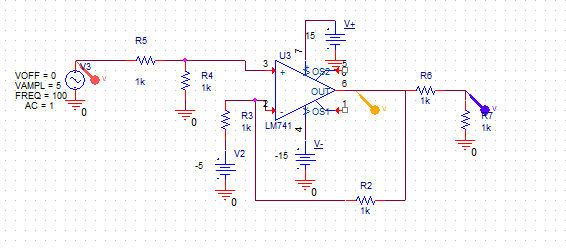
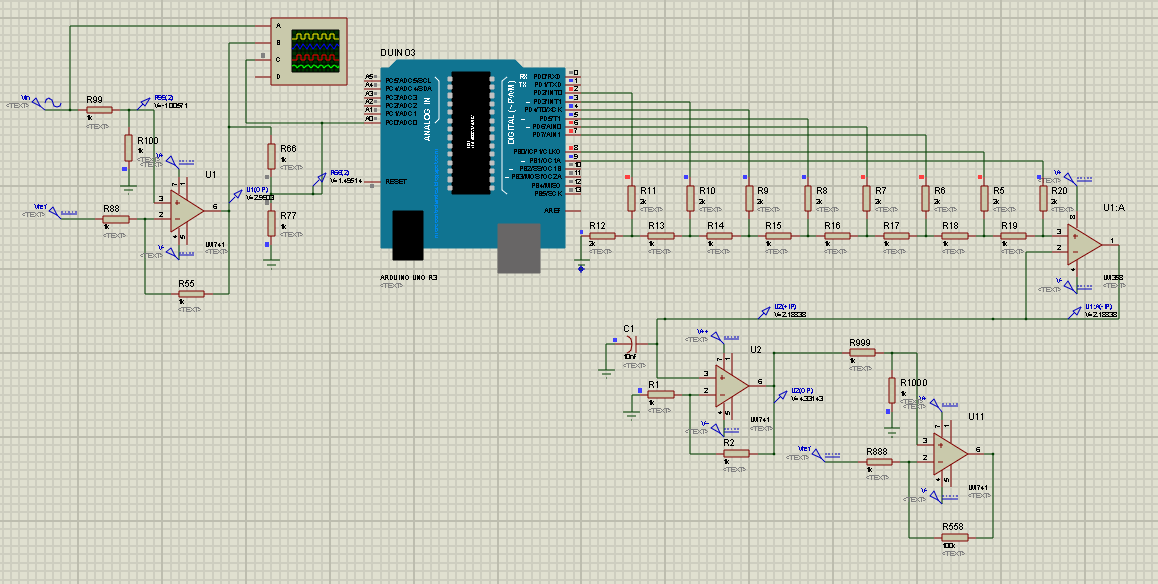




Figure :PSPICE Simulation of Differential Op-Amp

|  |
| --- |
| **ARDUINO IDE CODE** |
| const int OUT[8]={2,3,4,5,6,7,8,9};  void setup() {  Serial.begin(9600);  pinMode(A0,INPUT);  pinMode(2,OUTPUT);  pinMode(3,OUTPUT);  pinMode(4,OUTPUT);  pinMode(5,OUTPUT);  pinMode(6,OUTPUT);  pinMode(7,OUTPUT);  pinMode(8,OUTPUT);  pinMode(9,OUTPUT);  }  void loop() {  int a=analogRead(A0);  float vol=a\*0.004875;  float per=map(a,0,1023,0,100);  Serial.print("Volatge=");  Serial.println(vol);  Serial.println("Percentage=");  Serial.println(per);  Serial.println("Digital Value");  Serial.println(a);  // conversion of analogeRead value to 8-bit digital value for DAC  int nn=map(a,0,1023,0,256);  Serial.println(nn);  int dig\_bits[8];  int k=nn;  // counter for binary array  int i = 0;  while (k > 0) {    // storing remainder in binary array  dig\_bits[i] = k % 2;  k = k / 2;  i++;  }  Serial.println(dig\_bits[7]);  int bit\_status;  for (int j=0; j<8; j++)  {  bit\_status=dig\_bits[j];  digitalWrite(OUT[j],bit\_status);  }  delay(500);  } |

**PROTEUS SCHEMATIC**



**RESULTS:**

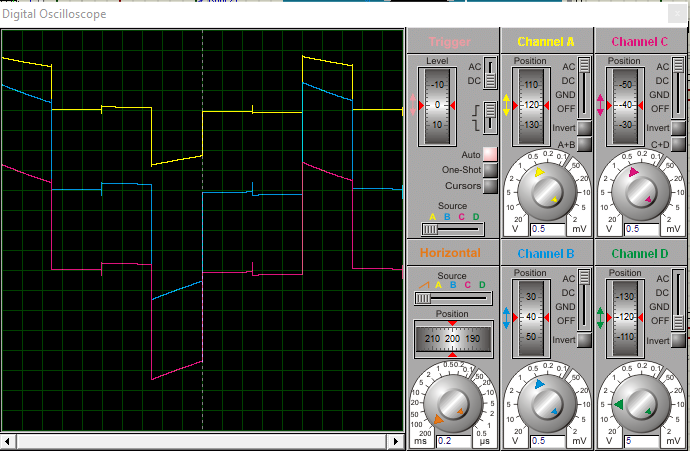


Figure : Before smoothing capacitor

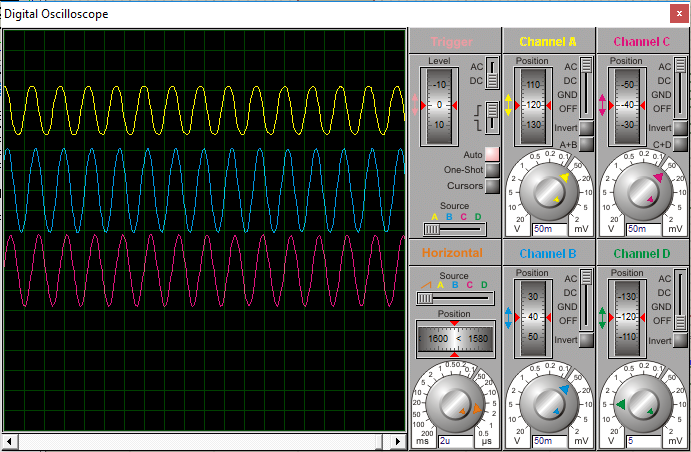


Figure : After smoothing capacitor

**Critical Analysis/Conclusion:**

|  |
| --- |
| In almost all digital systems there is a frequent need to convert analog signals generated by analog devices such as microphone, sensors and potentiometers into digital values that can be stored and processed. To serve this purpose analog to digital converters are used.  Arduino UNO has a 6 bit ADC that can detect 0-5V input and the map function can be used to find the digital value of the analogue input.  In this lab we used a potentiometers to vary different DC input values and analyzed them using the ADC of Arduino by displaying the equivalent percentages and equivalent analogue and binary values on Virtual Terminal and LCD.  We also regenerated the detected analogue input through the 8 digital pins of Arduino and an R2R Ladder using LM358.  These Analogue-Digital-Analogue techniques are widely used in sound systems and other high precision instrumentation applications like ECG etc. |

