DAC in an Arduino Uno using the R-2R Ladder Architecture

Shubhankar Pandit: 18-13-104

June 21, 2020

Synopsis

The primary objective of this project was to generate analog signals through the use of an *Arduino Uno* and by utilizing the concept of the *R-2R Ladder*. The *Arduino Uno* is used to supply appropriate input signals by making use of its digital pins to the voltage divider circuit of *R-2R Ladder*. The specifications of 4-bit *R-2R Ladder* are discussed in depth, and it's 8-bit variation is used in tandem with the *Arduino Uno*. The viability of generating analog signals internally through the Arduino is also discussed.

Contents

1 Introduction

1.1 Motivation

Arduino Uno does have a built in function to output analog signals known as *analogWrite()* which employs the use of **Pulse Width Modulation** but it is limited in its capabilities. Owing to the fact that PWM produces square waves, the signals generated are not purely analog. Although, a few modern microcontrollers do come with an internal *DAC*, it is relatively inexpensive to buy an external one. Hence, an attempt is made here to create one of the cheapest and simplest *DAC* architectures, the *R-2R Ladder*.

1.2 About DACs

Digital to Analog Converters have a plethora of uses in multiple fields, but are primarily employed in audio conversions. From converting digital data streams into analog audio signals, to converting digital video data into analog video data to make video calls possible, *DACs* are found in almost every aspect of technology.

DACs are of various types :

1. Summing Amplifier
   This is an *OP-AMP* with multiple resistors that combines multiple inputs to a single output that is the weighted sum of the applied inputs.

2. R-2R Ladder
   This is a type of voltage divider which only needs two kinds of resistor values for it to work.

3. Pulse Width Modulation DAC
   The kind of *DAC* that *Arduino Uno* has. It uses *PWM* to output analog signals.

1.3 About the Arduino

The *Arduino Uno* is an open-source microcontroller board based on the Microchip *ATmega328P* microcontroller and developed by Arduino.cc. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the *Arduino IDE* (Integrated Development Environment).
It provides a beginner friendly environment for novices to sharpen their *Embedded* skills. Despite being considered a microcontroller for the inexperienced, the possibilites of projects involving it are endless.

2 The R-2R Ladder

2.1 About the R-2R Ladder

The *R-2R Ladder* is a type of voltage divider circuit which consists of repeating units of resistors arranged in a ladder-like configuration.

The popularity of *R-2R Ladder* is due to several reasons

- It's sheer simplicity
- Only utilizes resistors of two values
- Can be extended to n number of bits
- Output impedance is always equal to R, hence simplifying filtering

However, *R-2R Ladder* can only produce as many voltage steps as the binary number would allow. Hence, despite being able to create signals with magnitude times better resolution than *PWM*, producing a truly continuous signal through this method is also almost impossible.

2.2 Analysis of 4-bit R-2R Ladder

Several observations about the circuit given in *Fig. 1* can be made

- The equivalent resistance is always R when looked from the left of each resistor R.
- The equivalent resistance is always $2R$ when looked from the right terminal of each resistor R.

Keeping these observations in mind can simplify our task greatly.

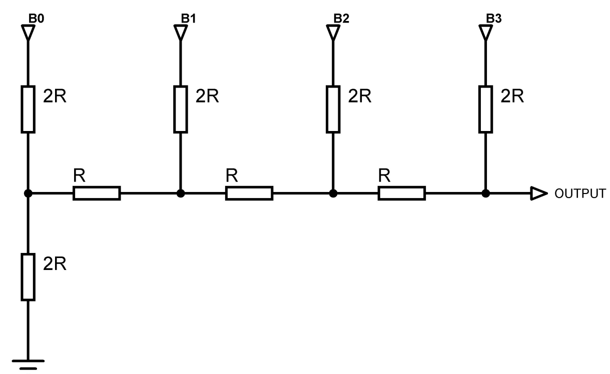
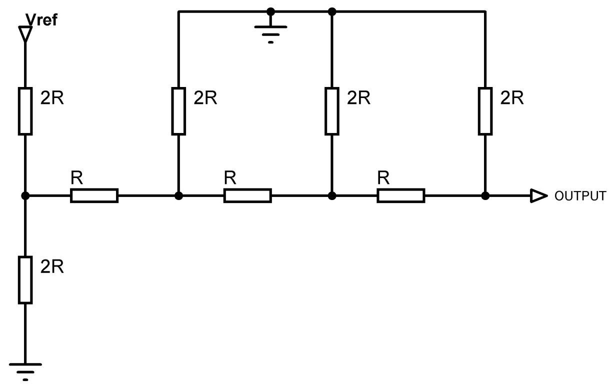


Figure 1: A 4-bit R-2R Network

This circuit will be analysed bit by bit, keeping all outputs at Logic 0 except one.
Then, after calculating the individual weightage of each input through *Thevenin's Theorem*, we'll add them to get the net output voltage by using the *Superposition Theorem*.

Now, according to *Thevenin's Theorem*, "Any linear circuit containing several voltages and resistances can be replaced by just one single voltage in series with a single resistance connected across the load".

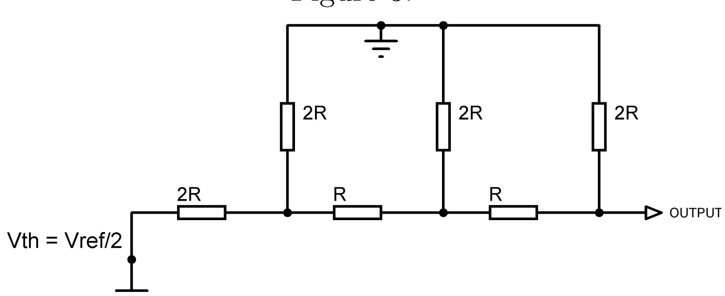
Figure 2:



Therefore, applying *Thevenin's Theorem* to the first branch in the above circuit (Fig. 2) assuming B_0 is V_{ref} and all other inputs are at Logic 0, and replacing all voltage sources with short circuits and all current sources with open circuits we get,

$$V_{th} = V_{ref}/2$$
$$R_{th} = R$$

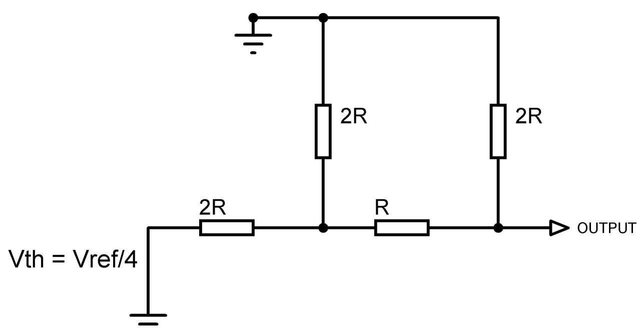
Figure 3:



Similarly, applying Thevenin's Theorem to the next branch, we get,

$$V_{th} = V_{ref}/4$$
$$R_{th} = R$$

Figure 4:

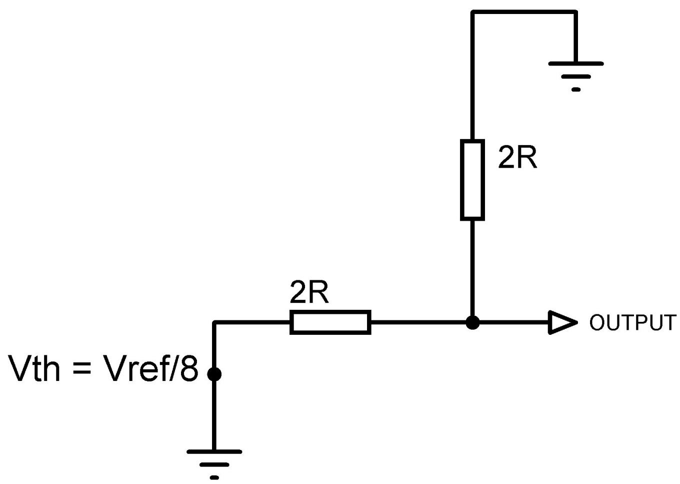


A repeated pattern is observed. V_{th} gets reduced by a factor of 2, and R_{th} stays the same.

We repeat the above steps again and reduce the next branch.
We get,

$$V_{th} = V_{ref}/8$$
$$R_{th} = R$$

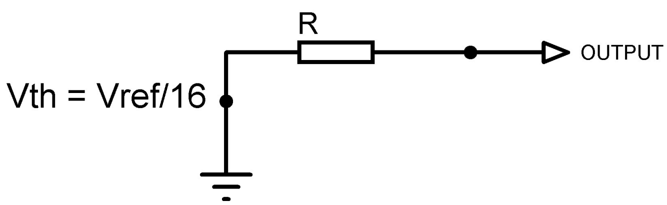
Figure 5:



As we can see, the pattern is repeated again. This process continues methodically by substituting the next branch with its *Thevenin* equivalent. We repeat this method for the last branch, and we get,

$$V_{th} = V_{ref}/16$$
$$R_{th} = R$$

Figure 6:



As we can see, V_{ref} is reduced by a factor of 2 and R_{th} stays the same in every iteration.

Each bit stage the voltage contributed by each branch gets affected by a factor of 2. Therefore the contribution of each bit to the output is their input voltage divided by their respective factors, *i.e.*

$$V_{out} = V_{B0}/16 + V_{B1}/8 + V_{B2}/4 + V_{B3}/2$$

And this forms the very principle of *DAC* using *R-2R Ladder*, wherein we can create voltage steps of $V_{B0}/16$, $V_{B1}/8$, $V_{B2}/4$ and $V_{B3}/2$. These voltage steps are the binary weighted functions of each bit, and thus can be used to convert any digital input to its analog counterpart, by providing appropriate voltages to each bit.

This formula can also be extended to n-bits,

$$V_{out} = \frac{V_{B0}}{2^n} + \frac{V_{B1}}{2^{n-1}} + \frac{V_{B2}}{2^{n-2}} + ... + \frac{V_{Bn}}{2^1}$$

2.3 The 8-bit R-2R Ladder

In the previous section the 4-bit *R-2R Ladder* was discussed and deconstructed. As it was discussed, that the *R-2R Ladder* can be extended to n number of bits, the 8-bit version of this circuit will be put to use in this project.

The 8-bit variant of this circuit provides more resolution and is suitable for getting input from *Arduino Uno's* PORT D which comprises of 8 Digital *I/O* pins, and is therefore better suited for converting digital signals to analog signals.

The equation for the output voltage in the case of 8-bit *R-2R Ladder* will be,

$$V_{out} = \frac{V_{B0}}{2^8} + \frac{V_{B1}}{2^7} + \frac{V_{B2}}{2^6} + \frac{V_{B3}}{2^5} + \frac{V_{B4}}{2^4} + \frac{V_{B5}}{2^3} + \frac{V_{B6}}{2^2} + \frac{V_{B7}}{2^1}$$

3 Details of Hardware Implementation

3.1 Circuit Description

This project is based on the *Arduino Uno*, and makes use of the 0 - 7 Digital pins, collectively defined as *PORT D* for **Port Manipulation**.

Two kinds of resistors are used for making the 8-bit *R-2R Ladder*, one of resistance $1Kohm$ and the other of $2Kohm$. An AC Voltmeter is also used across the circuit to measure the net output voltage.

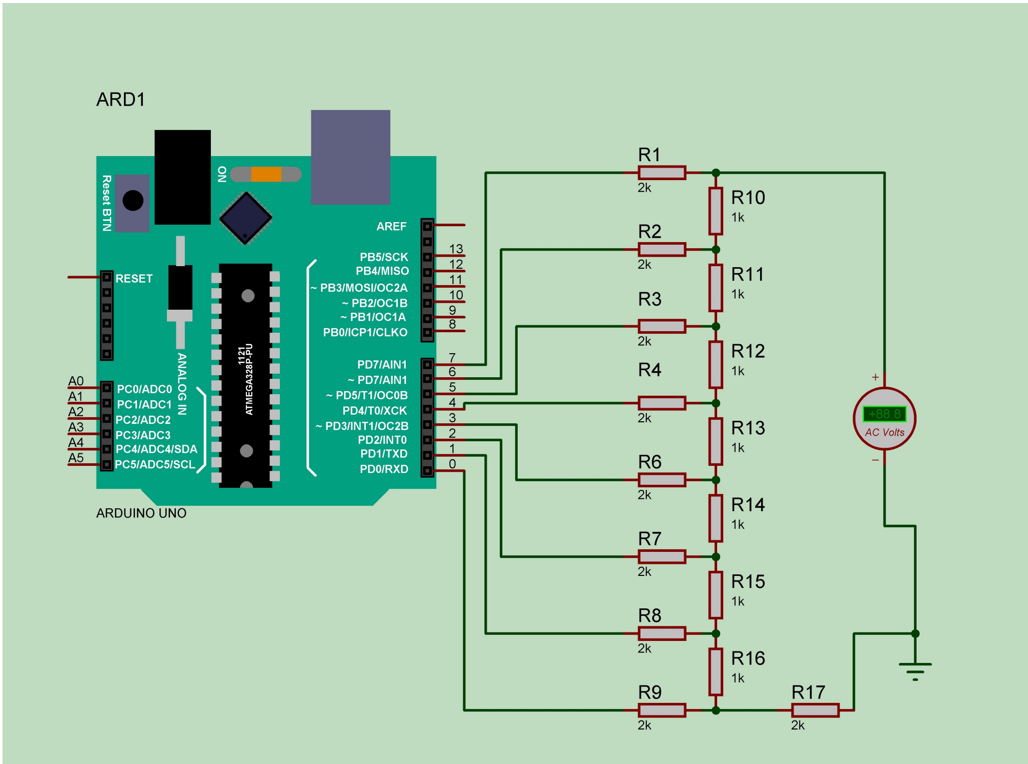


Figure 7: A comprehensive schematic of the circuit

As we can see in the figure above, each resistor of resistance $2Kohm$ ranging from R_1 to R_9 is connected to the digital pins $0-7$ of the *Arduino Uno*. The $1Kohm$ resistors from R_{10} to R_{16} are connected in tandem with the $2Kohm$ resistors to form the ladder. The R_{17} resistor is connected to the last branch and grounded to complete the ladder.

3.2 Bill of Materials

Part	Specification	Quantity
Arduino Uno		1
Resistors	1k ohm	7
Resistors	2k ohm	9
Oscilloscope		1

4 Details of Software Implementation

4.1 Theory

With the knowledge of the 8-bit *R-2R Ladder* equation we derived in the previous sections,

$$V_{out} = \frac{V_{B0}}{2^8} + \frac{V_{B1}}{2^7} + \frac{V_{B2}}{2^6} + \frac{V_{B3}}{2^5} + \frac{V_{B4}}{2^4} + \frac{V_{B5}}{2^3} + \frac{V_{B6}}{2^2} + \frac{V_{B7}}{2^1}$$

We know that each bit makes a certain contribution to the overall output voltage. Keeping this in mind, we can deduce that 00000000 or 0 in decimal will produce an output voltage of 0. And, 11111111 or 255 in decimal will produce the maximum output voltage (which will be dependent on the input high logic voltage of the pins, which in this case is $5V$). Therefore, adjusting the bits inputted into the ladder, we can produce its analog counterpart according to our requirements.

4.2 Code Description

The following analog signals were recreated through the help of *R-2R DAC* in this project

- Sine Curve
- Cosine Curve
- Rectangular Curve
- Sawtooth Curve
- Triangular Curve

Each curve entails a particular set of inputs which have to be given in the source code of the *Arduino*. The source code was written in *Embedded C* and compiled by the *Arduino Genuino* Windows Application.

```
//THE SOURCE CODE USED IN THE ARDUINO
uint8_t itr = 0, sin_itr ; //declaring unsigned 8-bit numbers to
    hold values
void setup() {
DDRD = B11111111; //configuring all Digital Pins as output pins
}
void loop() {
//For a sin graph
for(int i = 0; i <= 360; i++){ //iterating from 0 to 360 degrees
    for all possible values of sin
   sin_itr = ((sin(i * DEG_TO_RAD)) * 255) / 2; //converting to
       radians by multiplying i with constant DEG_TO_RAD
   PORTD = sin_itr;
   delay(5);
   }
//For a rectangular graph
PORTD = 255; //11111111 in binary
delay(500);
PORTD = 0; //00000000 in binary
delay(500);
//For a triangular graph
for(int i = -255; i < 255; i++){
   PORTD = abs(i);
   delay(1);
}
//For a sawtooth graph
itr %= 255;
PORTD = itr; //increasing the value of itr till 255, and then
    resetting it again
itr++;
delay(5);
//For a cosine graph
for(int i = 0; i <= 360; i++){ //iterating from 0 to 360
    degrees for all possible values of cosine
   sin_itr = ((cos(i * DEG_TO_RAD)) * 255) / 2; //converting to
       radians by multiplying i with constant DEG_TO_RAD
   PORTD = sin_itr;
   delay(5);
   }
}
```

`uint8_t` is used to declare two unsigned 8-bit integers `itr` and `sin_itr`. The reason being that since the code uses **Port Manipulation** we can directly equate the unsigned 8-bit value without doing any further calculations.

`DDRD` is used to configure the input/output settings of the 0 to 7 Digital pins of the *Arduino*. In this case we have set all the pins in output configuration.

Inside `loop` we have defined all the code segments to produce their respective graphs.

In the case of *sin* graph, a `for` loop iterates from the value 0 to 360 which are all the possible values of *sin* in degrees. The iterated value is converted to radians by using `DEG_TO_RAD` constant, and then multiplied by 255 since it is the highest value in 8-bit. This value is fed to `sin_itr` which in turn feeds it to `PORTD` of the *Arduino*. A delay is provided so that the value is retained by the pins for some time.
The same steps are repeated for the *cosine* graph, except the `sin` function is replaced by the `cos` function.

The triangular graph is produced by a `for` loop iterating from the lowest 8-bit value (-255) to the highest 8-bit value (255). The positive absolute value of this iterator is passed to `PORTD` by utilizing the `abs()` function.

For the rectangular graph, first `PORTD` is given the value of the highest output, *i.e.* 255 which is 11111111 in binary. And after a small `delay`, the smallest output (0 or 00000000 in binary) is fed. This creates a curve oscillating between the two extreme values.

To produce a sawtooth graph, the input should gradually increase till the maximum value has been reached, and then start from the minima again. First, the *modulus* of the `itr` variable with 255, which is the highest possible 8-bit value, is calculated. The resultant value is fed to `PORTD` and the `itr` variable is incremented. Once `itr` reaches 255, the value will reset to 0 once again (due to the *modulus* operation) and this process will continue indefinitely.

4.3 The Simulation

The project was simulated with the help of *Proteus 8* Windows Application. The *Arduino Uno* Library for *Proteus* was downloaded seperately.

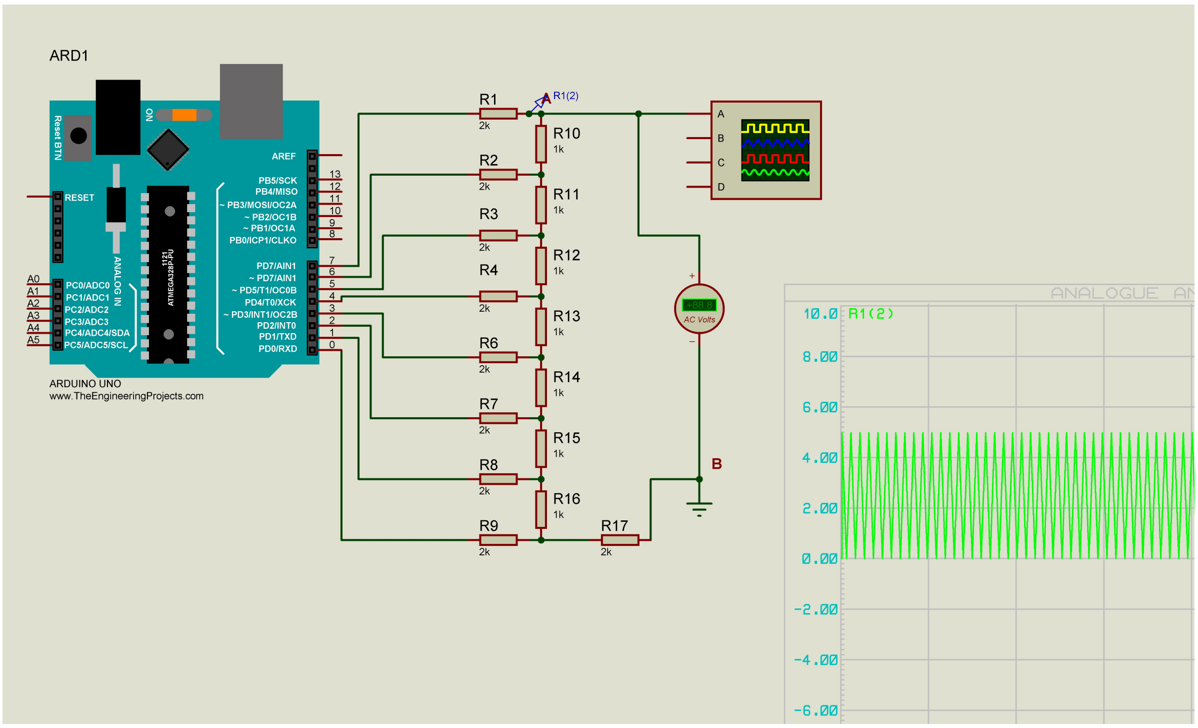


Figure 8: The setup used in Proteus

The *Oscilloscope* and the *Analogue Graph* features of *Proteus* were used to get the output graphs. The *Arduino* source file was exported to hex code and uploaded in the simulated *Arduino* board inside *Proteus*. A voltage probe was placed at point *A* and was plotted in the *Analogue Graph*. An *AC Voltmeter* was also used for reference.

The following graphs were generated in the simulation:

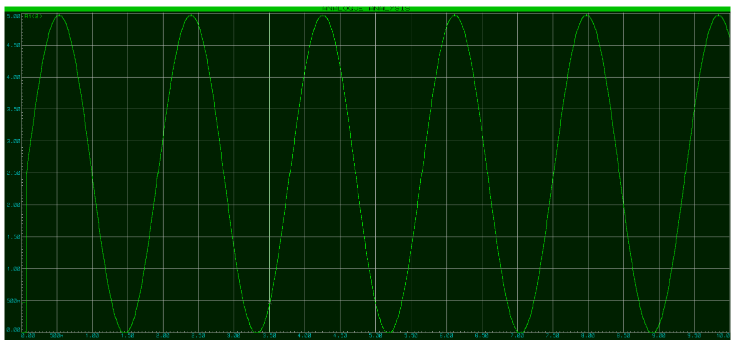


Figure 9: sine Wave Graph Output

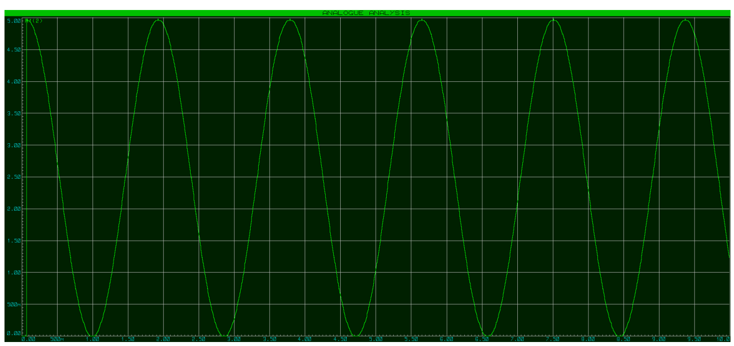


Figure 10: cosine Wave Graph Output

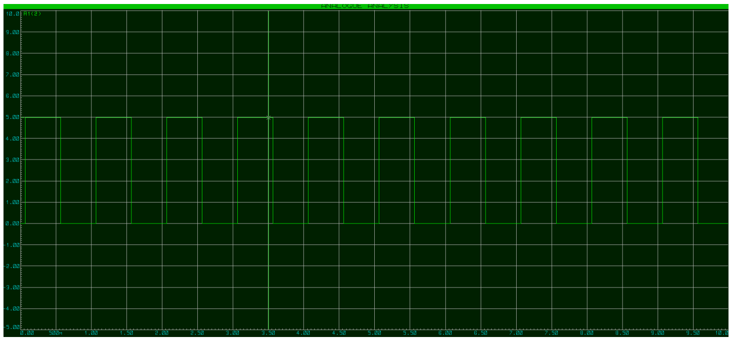


Figure 11: Rectangular Wave Graph Output

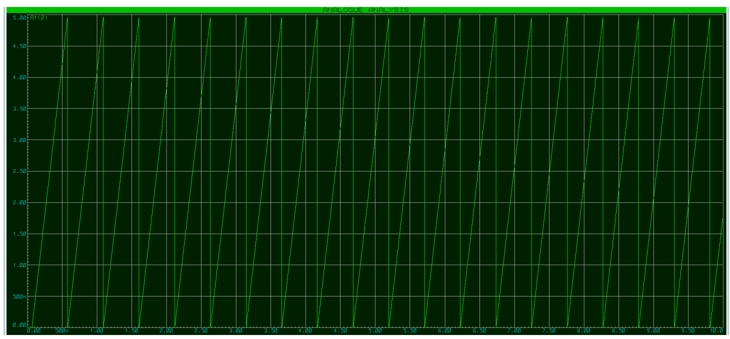


Figure 12: Sawtooth Wave Graph Output

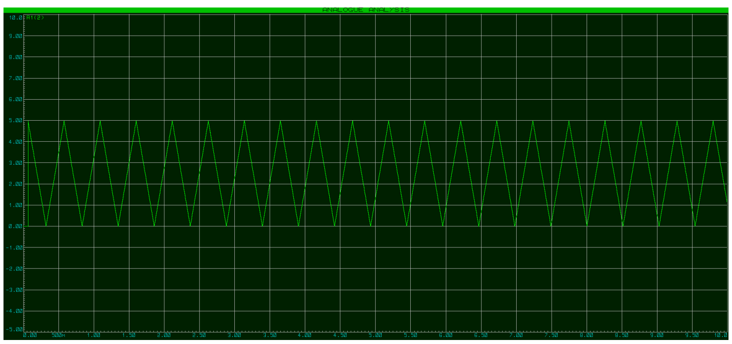


Figure 13: Triangular Wave Graph Output

5 Conclusion

Hence, it can be concluded from this project that *R-2R Ladder* configuration is an effective *Digital to Analog Converter* and can be used to convert Digital signals to Analog signals as an inexpensive replacement to the ICs available in the market and to the inbuilt *PWM* method of *Arduino*. However, the signals produced aren't purely analog and have a limited resolution. To upscale the quality, higher bit configurations of *R-2R* can be used instead.

References

[1] Introduction to Digital to Analog Converters (DAC) https://components101.com/articles/digital-to-analog-converters-dac

[2] The Operation and Characteristics of Voltage-Mode R-2R DACs https://www.allaboutcircuits.com/technical-articles/

[3] "The R-2R Ladder: What it is and Why you need one!" https://www.best-microcontroller-projects.com/R-2R-ladder.html