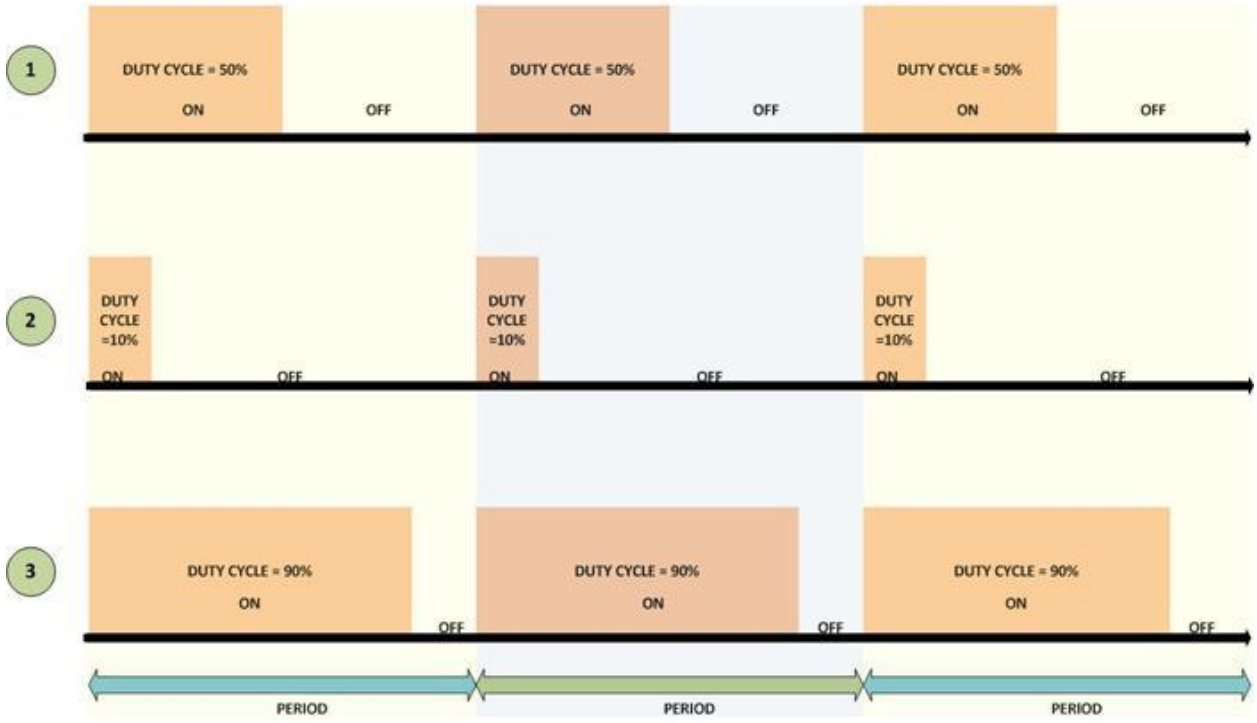
# Sine Wave Generation Using PWM on BBB

**PWM:**

The most common and popular technique of digital pure sine wave generation is pulse width modulation (PWM). The PWM technique involves generation of a digital waveform, for which the duty cycle is modulated such that the average voltage of the waveform corresponds to a pure sine wave. The simplest way of producing the PWM signal is through comparison of a low power reference sine wave with a triangle wave. Using these two signals as input to a comparator, the output will be a 2 level PWM signal. This PWM signal can then be used to control switches connected to a high voltage bus, which will replicate this signal at the appropriate voltage. Put through an LC filter, this PWM signal will clean up into a close approximation of a sine wave .Though this technique produces a much cleaner source of AC power than either the square or modified sine waves, the frequency analysis shows that the primary harmonic is still truncated, and there is a relatively high amount of higher level harmonics in the signal.



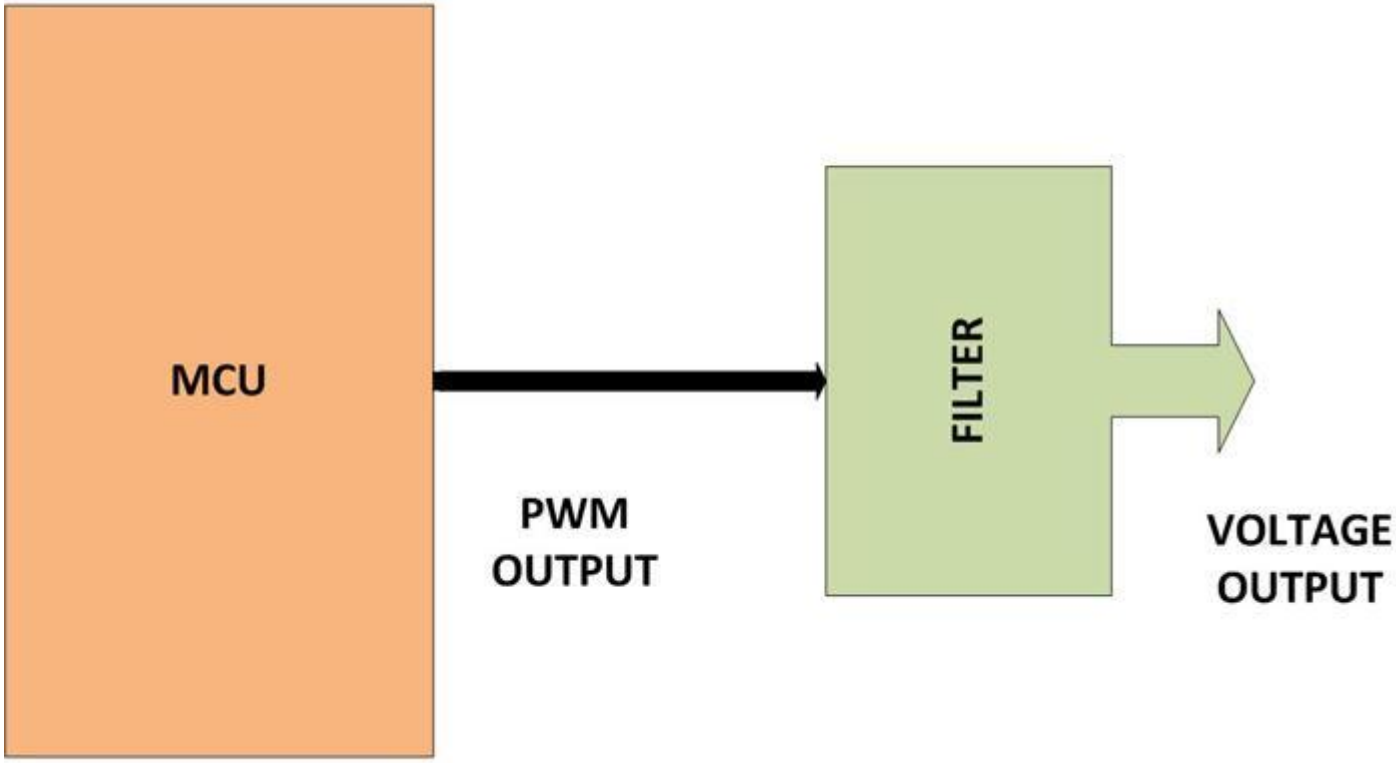

**period of the wave is the sum of the 'ON time + OFF time'. Duty-cycle is the percentage of time period for which the logic1 voltage exists in a cycle (ON time), starting from the beginning of the cycle.**

**The PWM is that kind of a wave in which the ON time and OFF time can vary in a cycle but the sum of 'ON time + OFF time' remains constant for every cycle.**

The period and the duty cycle for a PWM wave can be calculated generally using the following equations;

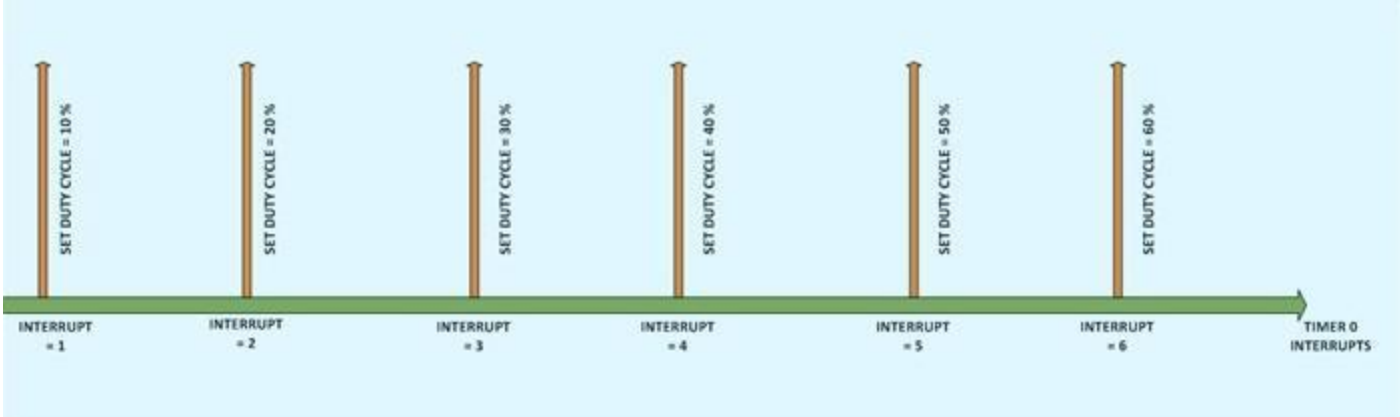
***Period = ON time + OFF time***

***Duty-cycle = ON time / (ON time + OFF time) = ON time / Period***
As shown in the figure, the first PWM wave has a Duty-cycle of 50% which means the on-time is exactly half of that the period of the wave. The second wave has 10% Duty-cycle and the third wave has 90% Duty-cycle.
According to the modulation of the width of a pulse in a period of the wave, the PWM can generate any required voltage with the help of a proper filter circuits. The filter circuits are used for generating the voltage corresponding to a modulated wave. Hence the PWM wave is always associated with a filter circuit.
When applied to a proper filter the 50% duty cycle can produce half of the maximum voltage of the pulse. If the maximum voltage (logic 1 voltage) of the pulse is 5V then the 50% duty cycle wave can produce a continuous 2.5V. The 10% duty cycle wave can produce nearly 0 voltages and the 90% duty cycle wave can produce nearly 5V.Thus the wave when applied to a filter can continuously produce a voltage which is the average of the voltage in a single period.
**Increasing the Duty-cycle will increase the voltage at the filter device's output and decreasing the Duty-cycle will decrease the voltage as well**

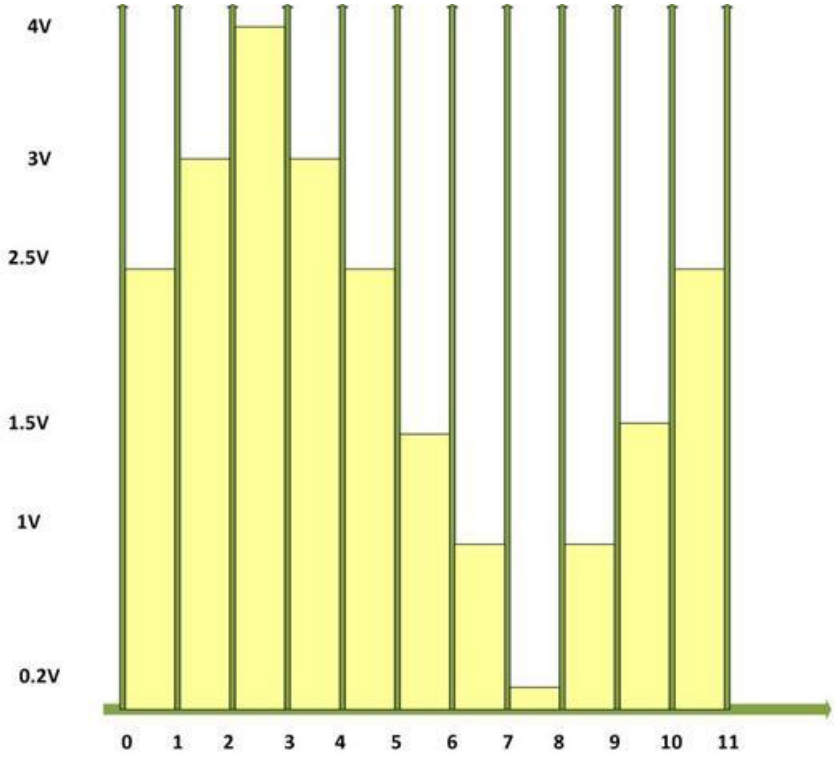



The filter could be any device which can generate the equivalent voltage of a PWM wave. It normally consists of Low-pass filters, amplifiers, load drivers etc. They generate the voltage corresponding to the pulse width modulation in the wave and can fed that voltage to the load device. The registers associated with the PWM modules like CPR1, PR2 and TMR2 etc. can be used to set the required Period and Duty-cycle in a PWM wave.

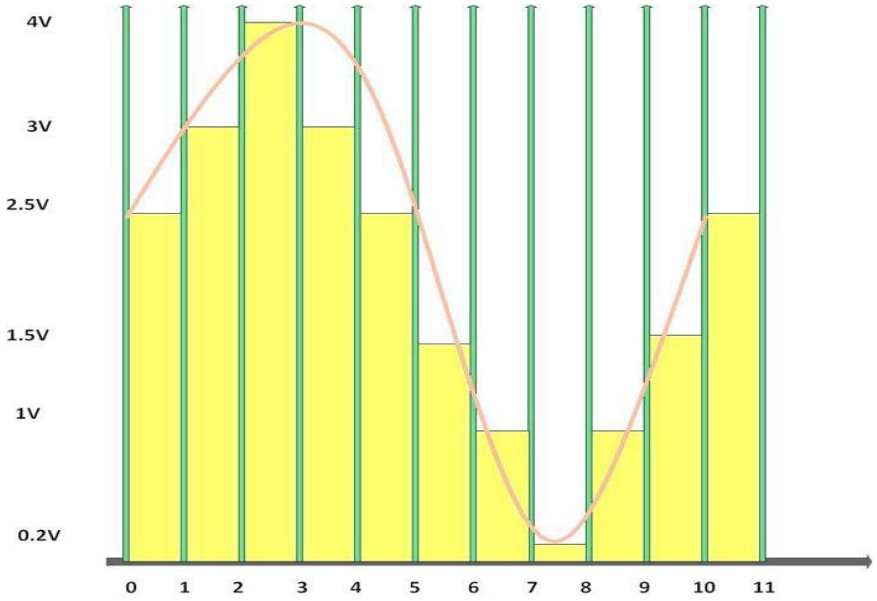
The voltage generated by the PWM wave at a filter device can vary by simply varying the Duty-cycle of the PWM wave. Increasing the Duty-cycle will increase the voltage at the filter device's output and decreasing the Duty-cycle will decrease the voltage as well. In this particular project the sine wave samples are generated periodically by re-writing the value of the CCPR1 register to vary the Duty-cycle. It is done by generating interrupts periodically with another timer module timer0 and changing the CCPR1 value when the code is inside the timer0's ISR. It is done as shown in the following figure;




**The voltage generated by the PWM wave in the interval between two interrupts will be a constant value and this time period can be called 'sampling period'.**
**The values of voltage that appears at each sampling period are simply called 'samples'.**
A sampling time can have 10 to 500 PWM cycles normally and depends on the particular output device requirement.
**The more the number of PWM cycles in a Sampling time, more stable the output voltage will be**



These values when applied to a filter circuit can generate the sine wave at its output by smoothing the step size. The output of the filter which is fed with the sine wave samples is shown in the following figure. The brown line shows the actual sine wave constructed by the filter circuit.

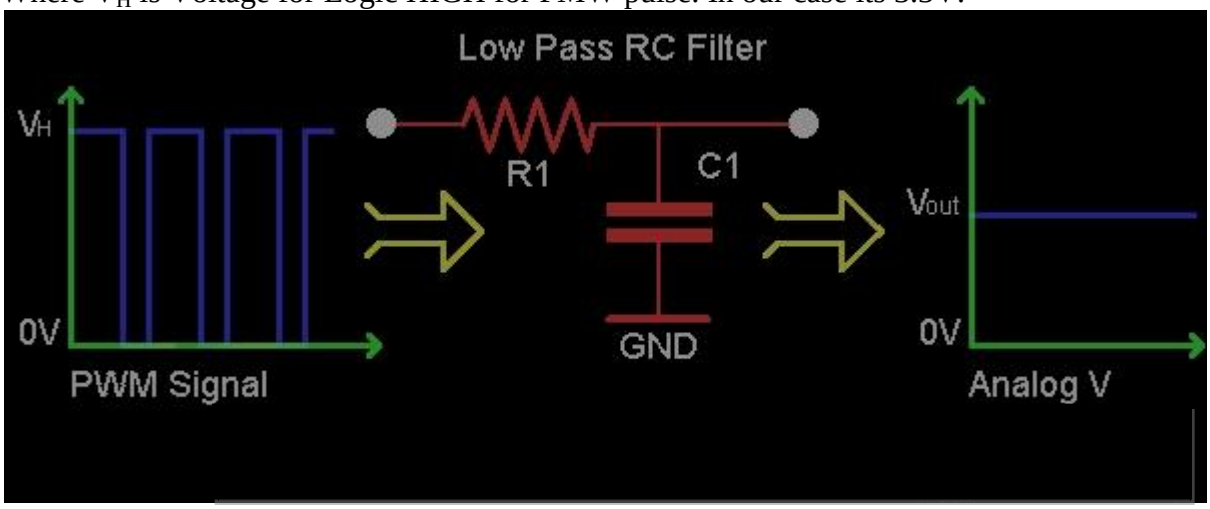



**The Basics:**

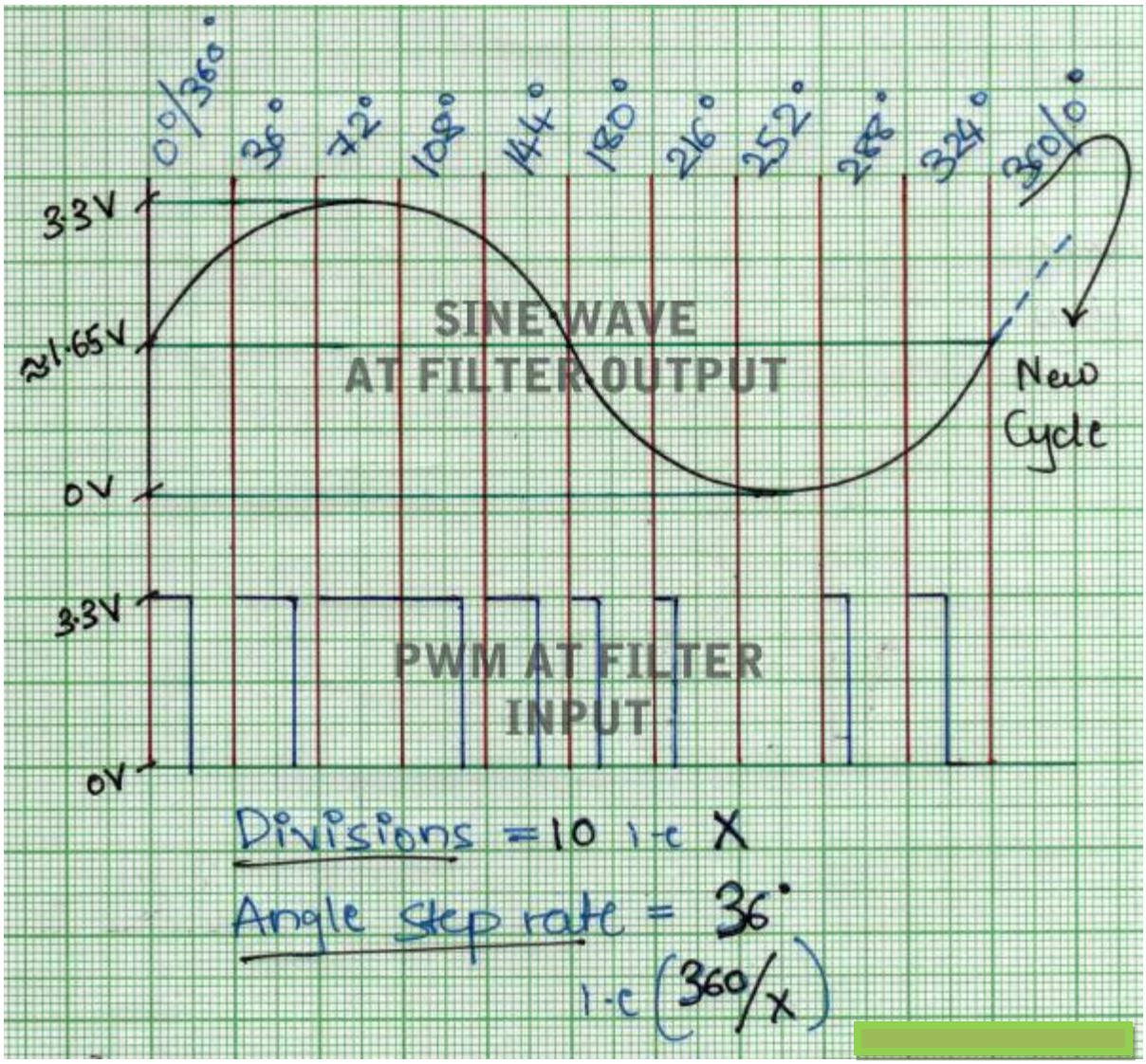
When we pass a PWM signal through a corresponding low pass filter then we get an analog voltage(or say Signal) at the output which is proportional to the dutycycle and given by:

**$V_{out}$ = $V_H$ x Dutycycle**

Where $V_H$ is Voltage for Logic HIGH for PMW pulse. In our case its 3.3V.




Now, when we change the dutcycle i.e the T-ON time of PWM in sine fashion we get a sine waveform at the filter output. The basic idea here is to divide the waveform we want , Sine wave in our case , into ‘x’ number of divisions. For each division we have a single PWM cycle. The T-ON time or the duty cycle directly corresponds to the amplitude of the waveform in that division which is calculated using sin() function. Consider the diagram shown below :




Here I've divided the Sine wave into 10 divisions. So here we will require 10 different PWM pulses increasing & decreasing in sinusoidal manner. A PWM pulse with 0% Duty cycle will represent the min amplitude(0V) , the one with 100% duty cycle will represent max amplitude(3.3V). Since out PWM pulse has voltage swing between 0V to 3.3V , our sine wave too will swing between 0V to 3.3V.
It takes 360 degrees for a sine wave to complete one cycle. Hence for 10 divisions we will need to increase the angle in steps of 36 degrees. This is called the **Angle Step Rate or Angle Resolution.**

Now we can increase the number of divisions to get more accurate waveform. But as divisions increase we also need to increase the resolution .. but we cant increase the resolution after we reach the max resolution which depends on the Processor(& Peripheral) clock.
**Resolution(PWM)** : Its the minimum increment required to increase or decrease the pwm T-ON time i.e Duty Cycle. For e.g if we have a resolution of 1us then valid T-ON values would be 1,2,3,.. and if we have a resolution of 0.5us then valid T-ON values will be 1,1.5,2,2.5,.. (i.e now we get double the values)

**Note :** A high(i.e more Fine) Resolution in our case means that the actual value is low and vice-verse. For eg. 0.25us is a higher Resolution as compared to 1us. – This is my convention others might use it in reverse.

**50 Hz Sinewave using PWM**

The above was just a simple example to explain the basic stuff. Now lets the actually example that I'll be using. **Here we will be generating a 50Hz Sine wave with 200 divisions.**

To get the output waveform as smooth as possible we need to calculate the best possible divisions and resolution .. given we know the frequency of the output waveform. In our case will be generating a 50Hz sine wave using PWM signal generated by lpc2148 microcontroller.

Now for a 50Hz sine wave we get a period time 1/50 = 20 milliseconds which is the time required for the sine wave to complete 1 full cycle. Now our job is divide an interval of 20ms into 'X' divisions. Hence the number of division i.e X will decide the period of PWM signal and hence the best possible resolution can be chosen.

**Note :** The first thing that we must keep in mind is that we need the resolution to be a rational number like 0.25 and not 0.1666666… to keep things more accurate and calculations straight forward. Given that maximum speed of lpc2148 is 60mhz it would take 1us(micro-second) for 60 ticks of CPU clock(assuming CPU and Peripheral Clocks are at same speed) , 0.1 us for 6 ticks , 0.05us for 3 ticks and 0.0166666… for 1 tick. Hence the maximum resolution will be 0.016666..us i.e 16.6666..ns(nano-seconds). But since its an irrational number the best maximum resolution we can use is 0.05 us or 50ns. But in our case we wont be using such fine resolution since our number or divisions are less. If our number of division would have been more than we could have used 50ns as our resolution. For our case 0.5us resolution is enough.

Now , **since we will be using 200 divisions , the period of each division will be 20ms/200 = 0.1ms = 100us**. This means that the period of the pwm signal will be 100us. Hence if we use a resolution of 1us then we will only have a set 100 values(actually 101 values if you consider 0 also) ranging from 1 to 100 to describe a full sine wave cycle. If we use a resolution of 0.5ms then we will have a set of 200 values. These values are the match values i.e the number of ticks of PWM counter. So if the resolution is 0.5us and period is 100 us then we will require 200 ticks to complete 100us. Similarly if resolution is 0.25us we will require 400 ticks for 100us.

**Max Number of PWM counter Ticks = PWM Period / Resolution**

This gives us the Maximum value for the match register i.e 100% Dutycycle.

**NOTE: The higher this count is the more accurate the sine wave will be – given we have correspondingly high number of divisions!**.

**Generating the Sine lookup table**

In our case since our period time is 100us we will require max 200 ticks @ 0.5us resolution. So in a lot of 200 divisions or PWM cycles the match value i.e the no.of ticks must vary from 0 to 200 in a sine fashion. This can done by finding the angle for the corresponding division. So now we must first find the angle step rate or the angle resolution , which can be given by:

**Angle step rate = 360 / No.of divisions**

In our case we get Angle step rate = **1.8 degrees**. Hence the 1st division represents 0 degrees , 2nd division 1.8 degs , 3rd division 3.6 degs and so on.

Now , by multiplying the sine of angle with 50% T-ON value we can get the T-ON values for a complete cycle. But since sine gives negative values for angle between 180 and 360 we need to scale it since PWM Match values cannot be negative. Again we use 50% T-ON and add it so all values are positive. This is given by :

SineLookupTable[Y] = rint( 100 + 100 * sin( Angle_Step_Rate*Y ) );

//where Y=0,1,...X-1 & X=no.of Divisions

**Note** that the function "**rint(..)**" is used to round the computed Match Value to the nearest integer.

Using the above equation the min amplitude will be 0V(match val=0) and max 3.3V(match val=200). In order to limit the amplitude between say 0.05V(match val=1) and 3.25V(match val=199) we can modify the equation by reducing the sine scale factor from 100 to 99. So , the equation in this case will be :

SineLookupTable[Y] = rint( 100 + 99 * sin( Angle_Step_Rate*Y ) );

//where Y=0,1,...X-1 & X=no.of Divisions

**Filter Design**

We need a **Low Pass filter(LPF)** which we will be using as **Digital to Analog Converter i.e DAC** for converting PWM into Sine. This Low Pass Filter will block the high frequency PWM signal and will let the 'encoded' low frequency sine wave to pass through. To design this filter , first we must decide the Cut-Off or Corner frequency. Here we can follow a rule of thumb that the cut-off frequency must be 3 to 4 times the frequency we want at the output. Since we want 50hz sine at the output we will design a simple low pass RC filter with cut-off frequency of around 200hz.

The Cut-off Frequency for a RC Filter is given by:

$$\mathbf{F_c = \frac{1}{2 * PI * R * C} Hertz}$$

Here the Time Constant is $\mathbf{T_c = R * C}$

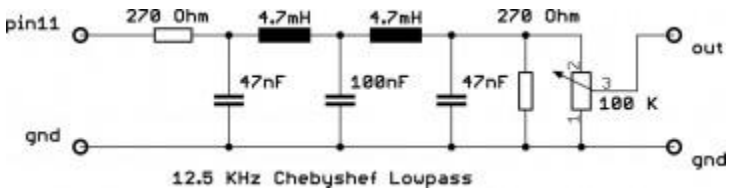
So , from the above equation we get the equation for Time Constant as :

$$\mathbf{T_c = R * C = \frac{1}{2 * PI * F_c} seconds}$$

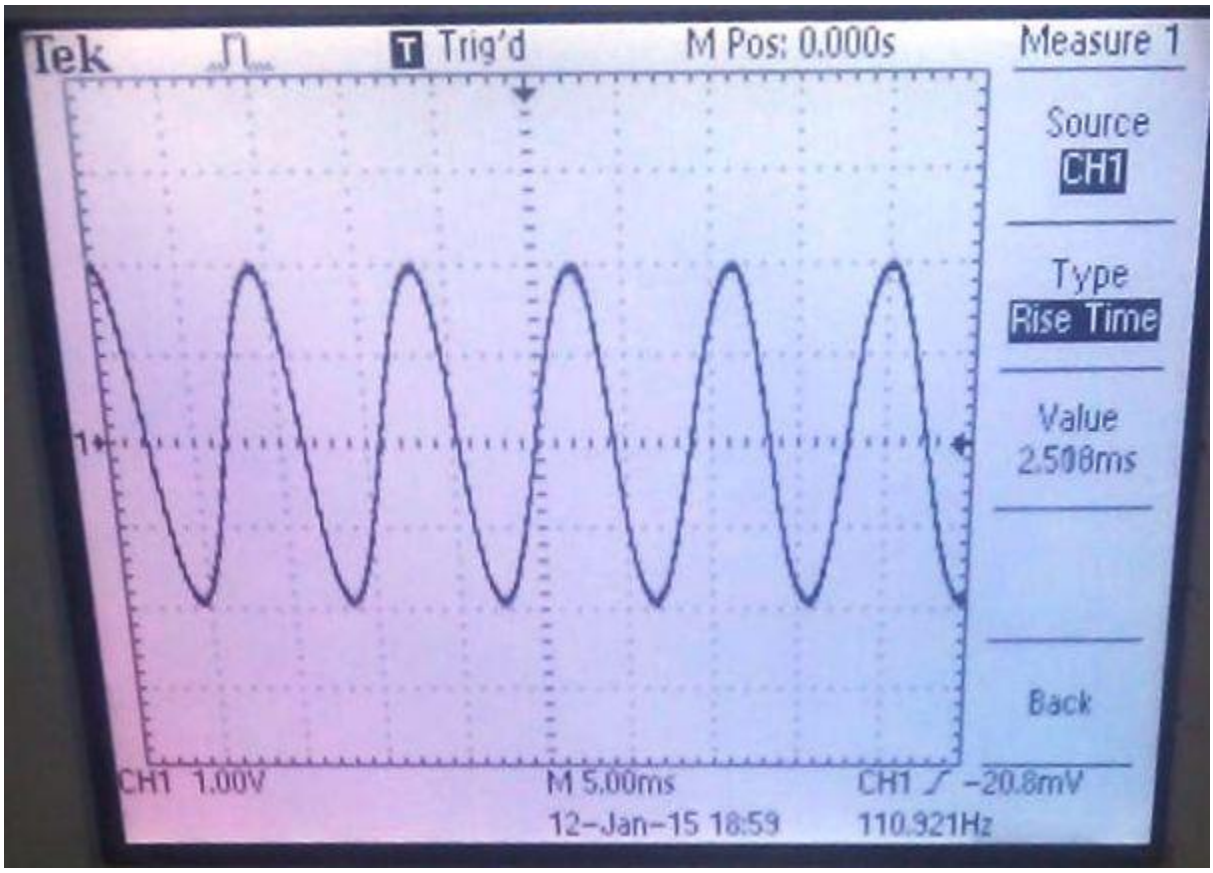
Now we can Find the Time Constant $T_c$ from $F_c$ and then select **appropriate** values for R and C. So , In our case with $\mathbf{F_c=200}$ **, we get** $\mathbf{T_c=796us}$. Hence in our case we can select a **1.8K Resistor and a 0.47uF Capacitor** which gives a Time constant of **846us**.

If you don't have R & C with those values and just want to check the Sine wave output - **A Quick & Dirty method to make an RC filter** for the PWM in **our case** can be as follows – use a resistor between 1.8K and 5.6K and a Capacitor between 0.33uF and 1uF. For example you may use a 4.7K Resistor with a 0.47uF or 1uF Capacitor.

Note : If you need more better Sine Wave you can use a Low-Pass RC or Chebyeshev Filter of Higher order.



**Output in CRO:**



## References:


http://www.engineersgarage.com/embedded/pic-microcontroller-projects/sine-wave-generator-using-pwm-with-pic-microcontroller

http://www.ocfreaks.com/sine-wave-generator-using-pwm-lpc2148-microcontroller-tutorial/

https://www.wpi.edu/Pubs/E-project/Available/E-project-042711-190851/unrestricted/PWM_Techniques_final.pdf

http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-dds-sinewave-generator/

http://aquaticus.info/pwm-sine-wave