```
// CONFIG

#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON       // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)

#pragma config LVP = ON         // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; lo
w-voltage programming enabled)

#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by E
ECON control)

#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)

#define _XTAL_FREQ 20000000

#define TMR2PRESCALE 4

#include <xc.h>

long PWM_freq = 5000;

PWM_Initialize()

{

  PR2 = (_XTAL_FREQ/(PWM_freq*4*TMR2PRESCALE)) - 1; //Setting the PR2 formulae using Datasheet // Makes the PWM work in 5KHZ

    CCP1M3 = 1; CCP1M2 = 1;  //Configure the CCP1 module

    T2CKPS0 = 1;T2CKPS1 = 0; TMR2ON = 1; //Configure the Timer module

    TRISC2 = 0; // make port pin on C as output

}

PWM_Duty(unsigned int duty)

{

      if(duty<1023)

  {

    duty = ((float)duty/1023)*(_XTAL_FREQ/(PWM_freq*TMR2PRESCALE)); // On reducing //duty = (((float)duty/1023)*(1/PWM_freq)) / ((1
/_XTAL_FREQ) * TMR2PRESCALE);

    CCP1X = duty & 1; //Store the 1st bit

    CCP1Y = duty & 2; //Store the 0th bit

    CCPR1L = duty>>2;// Store the remining 8 bit

  }

}

void ADC_Initialize()

{

  ADCON0 = 0b01000001; //ADC ON and Fosc/16 is selected
  ADCON1 = 0b11000000; // Internal reference voltage is selected

}

unsigned int ADC_Read(unsigned char channel)

{

  ADCON0 &= 0x11000101; //Clearing the Channel Selection Bits

  ADCON0 |= channel<<3; //Setting the required Bits

  __delay_ms(2); //Acquisition time to charge hold capacitor

  GO_nDONE = 1; //Initializes A/D Conversion

  while(GO_nDONE); //Wait for A/D Conversion to complete

  return ((ADRESH<<8)+ADRESL); //Returns Result

}

void main()

{

    int adc_value;

  TRISC = 0x00; //PORTC as output

  TRISA = 0xFF; //PORTA as input

  TRISD = 0x00;

  ADC_Initialize(); //Initializes ADC Module

  PWM_Initialize();  //This sets the PWM frequency of PWM1

  do

  {

    adc_value = ADC_Read(4); //Reading Analog Channel 0

    PWM_Duty(adc_value);


      __delay_ms(50);


  }while(1); //Infinite Loop


}
```