```c
#include "stm32f10x.h"

/*
    SysTick, belirli beklemeleri gerçeklestirmek için kullanilir. Asagi sayicidir.
    Her clock darbesinde bulundugu degeri bir asagiya sayar ve sifir oldugunda
    basta yüklenen degerden tekrar saymaya baslar.

    SysTick Register'lari;

    SysTick_CTRL
    SysTick kontrol register'idir. 32 bit uzunlugundadir [0-31].
      0.bit "Enable" bitidir, bu sayiciyi aktif hale getirmek için kullanilir. 1 iken aktifdir.
      1.bit "Tickint" bitidir, interrupt fonksiyonunu devreye almak için kullanilir.
      2.bit "Clock Source" bitidir. 0 iken 72mHz/8, 1 iken 72mHz clock darbesi üretilir.
      16.bit "CountFlag" bitidir, sayici sifira ulastiginda 1 degerini alir.
      *Clock Source, default 0 degerini alir saniyede 9 milyon sayar.
      **CountFlag ve Tickint bitleri 1 iken "SysTick Interrupt" Fonksiyonu çagrilir.

    SysTick_LOAD
    Kaçtan itibaren asagi dogru saymak istedigimiz degeri tutan register'dir.
      Ilk 24 biti bu degeri tutmak için ayrilmistir[0-23]. Max: 16.777.215 degerini alabilir.
      Tanimlanan deger SysTick calistigi sürece degismez.

    SysTick_VAL
    Baslangic degerinden sonra asagi dogru sayarkenki degerleri tutar.
      SysTick calistigi sürece sifira kadar birer azalir.
      Sifira ulastiginda "SysTick_LOAD" register'indaki deger tekrar yüklenir.
      Sifira ulastiginda "CountFlag" register'i 1 degerini alir.

    Iki SysTick interrupt arasi geçen süre; T = (SysTick_LOAD + 1) * Clock Source Period
    */

static uint32_t a1,pwm=100;
  /* uint32_t yazmamizdaki amac bitwise islemlerde en soldaki bitin isaret biti olmasidir.
     1 olmasi durumunda sayinin negatif oldugu anlamina gelir bu yüzden compiler bizi uyarir,
     eger signed(isaretli) bir degisken tanimlamamiz halinde ilk bitinde 1 olursa
     bize negatif bir sayi döndürecektir. Negatif sayi ile egrasmadigimizda unsigned degisken
     tanimlamak olasi hatalari önler. */

static void delay(uint32_t a) {
    /* ORNEK-1 */
  SysTick -> LOAD = 9000*a; // a degeri milisaniye olarak yazilmali. 1000ms = 1sn
    /* Baslangiç degeri olarak 9000*a degerini atadik */
  SysTick -> CTRL |= 1;
    /* Enable bitini 1 yaparak SysTick aktif hale getirdik */
  while(!(SysTick -> CTRL &(1 << 16))){}
    /* CountFlag degeri 1 oldugunda yani sayim islemi bittiginde döngüden cikilacak */
  SysTick -> CTRL &= ~(uint32_t)1;
    /* Enable bitini 0 yaparak SysTick pasif hale getirdik. */
}

int main(){
  /*
  // ORNEK-1
  RCC -> APB2ENR |= (1 << 4);
  GPIOC -> CRH |= (1 << 21);
  GPIOC -> CRH &= ~(uint32_t)(1 << 22);

  while(1){
    GPIOC -> ODR |= (1 << 13);
    delay(1000);
    GPIOC -> ODR &= ~(uint32_t)(1 << 13);
    delay(1000);
  }
  */


  // ORNEK-2
  RCC -> APB2ENR |= (1 << 4);
  GPIOC -> CRH |= (1 << 21);
  GPIOC -> CRH &= ~(uint32_t)(1 << 22);

  SysTick -> LOAD = 9000000; //saniyede 1 kez
  SysTick -> CTRL |= 3;
   // Register'in Tickint ve Enable bitlerini 1 yaptik.
  while(1){ }

```

```
78
79      /*
80      //  ORNEK-3
81      RCC -> APB2ENR |= (1 << 4);
82      GPIOC -> CRH |= (1 << 21);
83      GPIOC -> CRH &= ~(uint32_t)(1 << 22);
84
85      SysTick -> LOAD = 9000; //1 milisaniye
86      SysTick -> CTRL |= 3;
87
88      while(1){
89      }
90      */
91    }
92
93
94    void SysTick_Handler(){
95      /* Bu fonksiyon SysTick_CTRL register'inin
96      Tickint ve CountFlag bitleri 1 oldugunda calisir. */
97
98
99      // ORNEK-2
100     if(GPIOC ->ODR & (1<<13))
101       GPIOC ->ODR &= ~(uint32_t)(1<<13);
102     else
103       GPIOC -> ODR |= (1<<13);
104
105
106     /*
107     // ORNEK-3
108     //  pwm degiskenini degistirerek ledin yanik kalma süresini
109     //  artirabilir veya azaltabilirsiniz.
110     a1++;
111     if(a1<pwm)
112       GPIOC ->ODR &= ~(uint32_t)(1<<13);
113     else
114       GPIOC -> ODR |= (1<<13);
115
116     if (a1>1000)
117       a1=0;
118     */
119   }
120
121
```