```c
/*ENEL 387*/
/*Authors: Daniel Takyi & Dwijen Kapadia*/
/*Filename: pwm.c*/



#include "stm32f10x.h"
#include "pwm.h"


void pwmInit(void)
{
  RCC->APB2ENR |= RCC_APB2ENR_TIM1EN | RCC_APB2ENR_IOPAEN | RCC_APB2ENR_AFIOEN; //ENABLE clocks for Timer 1, Port A, and alternate function I/O


  GPIOA->CRH &= 0xFFFFFF00;
  GPIOA->CRH |= 0x000000BB; //Alternate function output Push-pull, max speed 50 MHz on PA8 and PA9



  TIM1->CR1 |= TIM_CR1_CEN; // Enable Timer1
  TIM1->CR2 |= TIM_CR2_OIS2 | TIM_CR2_OIS1; // Output Idle State for Channel 1 & 2 OC1=1 when MOE=0
  TIM1->EGR |= TIM_EGR_UG; // Reinitialize the counter
  TIM1->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1PE | TIM_CCMR1_OC1FE; //PWM mode 1, Preload Enable, Fast Enable on CH1
  TIM1->CCMR1 |= TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1 | TIM_CCMR1_OC2PE | TIM_CCMR1_OC2FE; //PWM mode 1, Preload Enable, Fast Enable on CH1
  TIM1->CCER |= TIM_CCER_CC1E | TIM_CCER_CC2E; //Enable CH1 output on PA8 & CH2 output on PA9
  TIM1->PSC = 0x095F; //Divide 24 MHz by 2400 , PSC_CLK = 10000 Hz, 1 count = 0.1 ms
  TIM1->ARR = 200; // period = 20 ms <== is required for the motor controller(from datasheet)
  TIM1->CCR1 = 0; // 10 counts = 1 ms
  TIM1->CCR2 = 0; // 10 counts = 1 ms
  TIM1->BDTR |= TIM_BDTR_MOE | TIM_BDTR_OSSI; //Main Output Enable, Force Idle Level First
  TIM1->CR1 |= TIM_CR1_ARPE | TIM_CR1_CEN; // Enable Timer1
}


/*****************************************************************
**From the motorcontroller datasheet- the motor controller uses **
**pulsewidth to determine speed and direction of the motors'    **
**rotation.                                                     **
**Pulsewidth > 1520us ==> Forward revolution                    **
**Pulsewidth < 1480us ==> Backward revolution                   **
**1480us > Pulsewidth > 1520us ==> No revolution                **
*****************************************************************/

void PWMForward(void)
{
  TIM1->CCR1 = 17;
  TIM1->CCR2 = 17;
}
void PWMLeft(void)
{
  TIM1->CCR1 = 16;
  TIM1->CCR2 = 15;
}
void PWMRight(void)
{
  TIM1->CCR1 = 15;
  TIM1->CCR2 = 16;
}
void PWMSharpLeft(void)
{
  TIM1->CCR1 = 16;
  TIM1->CCR2 = 14;
}
void PWMSharpRight(void)
{
  TIM1->CCR1 = 14;
  TIM1->CCR2 = 16;
}
void PWMStop(void)
```

```c
{
  TIM1->CCR1 = 15;
  TIM1->CCR2 = 15;
}
void PWMBack(void)
{
  TIM1->CCR1 = 14;
  TIM1->CCR2 = 14;
}





```