```c
/* file: PWM.c */
/* Author - Onisokien Ayonoadu
            Priscilla Chua, Mar 10 2020 */

#include "stm32f10x.h"
#include "clocks.h"
#include "PWM.h"
#include "LCDlab.h"
#include "IR_Sensor.h"

//initialization of PWM
void PWM_clocks(void)
{
  RCC->APB1ENR |=  RCC_APB1ENR_TIM4EN ; //TIM 4

  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_IOPBEN | RCC_APB2ENR_IOPCEN; //port A, B, C

  RCC->APB2ENR |= RCC_APB2ENR_AFIOEN; //AFIO

  GPIOB->CRL |= GPIO_CRL_CNF6_1 | GPIO_CRL_MODE6; //Left forward (Orange)
  GPIOB->CRL &= ~GPIO_CRL_CNF6_0;

  GPIOB->CRL |= GPIO_CRL_CNF7_1 | GPIO_CRL_MODE7; //Left backward (Green)
  GPIOB->CRL &= ~GPIO_CRL_CNF7_0;

  GPIOB->CRH |= GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8; //Right forward
  GPIOB->CRH &= ~GPIO_CRH_CNF8_0;

  GPIOB->CRH |= GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9;  //right backward
  GPIOB->CRH &= ~GPIO_CRH_CNF9_0;

}

//timer initialization for PWM
void Timer4_PWM(void)
{
  TIM4->CR1 |= TIM_CR1_CEN; // Enable Timer4
  TIM4->CR2 |= TIM_CR2_OIS1; // Output Idle State for Channel 1 OC1=1 when MOE=0
  TIM4->CR2 |= TIM_CR2_OIS2;
  TIM4->CR2 |= TIM_CR2_OIS3;
  TIM4->CR2 |= TIM_CR2_OIS4;
  TIM4->EGR |= TIM_EGR_UG; // Reinitialize the counter

  //PWM mode 1, Preload Enable, Fast Enable
  TIM4->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1PE | TIM_CCMR1_OC1FE;
  TIM4->CCMR1 |= TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1 | TIM_CCMR1_OC2PE | TIM_CCMR1_OC2FE;
  TIM4->CCMR2 |= TIM_CCMR2_OC3M_2 | TIM_CCMR2_OC3M_1 | TIM_CCMR2_OC3PE | TIM_CCMR2_OC3FE;
  TIM4->CCMR2 |= TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1 | TIM_CCMR2_OC4PE | TIM_CCMR2_OC4FE;

  TIM4->CCER |= TIM_CCER_CC1E; //Enable CH1 output on PB6
  TIM4->CCER |= TIM_CCER_CC2E; //Enable CH1 output on PB7
  TIM4->CCER |= TIM_CCER_CC3E; //Enable CH1 output on PB8
  TIM4->CCER |= TIM_CCER_CC4E; //Enable CH1 output on PB9

  TIM4->PSC = 0x095F; //Divide 24 MHz by 2400 , PSC_CLK = 10000 Hz, 1 count = 0.1 ms
  TIM4->ARR = 100; // 100 counts = 10 mS
  TIM4->CCR1 = 50; // 50 counts = 5 mS = 50% duty cycle
  TIM4->CCR2 = 50; // 50 counts = 5 mS = 50% duty cycle
  TIM4->CCR3 = 50; // 50 counts = 5 mS = 50% duty cycle
  TIM4->CCR4 = 50; // 50 counts = 5 mS = 50% duty cycle

  TIM4->BDTR |= TIM_BDTR_MOE | TIM_BDTR_OSSI; //Main Output Enable, Force Idle Level First
  TIM4->CR1 |= TIM_CR1_ARPE | TIM_CR1_CEN; // Enable Timer1
}

void move_forward(void)
{
  TIM4->CCR1 = 75;
  TIM4->CCR2 = 0;
  TIM4->CCR3 = 0;
  TIM4->CCR4 = 75;
}
```

```
 73
 74  void move_backward(void)
 75  {
 76    TIM4->CCR1 = 0;
 77    TIM4->CCR2 = 75;
 78    TIM4->CCR3 = 75;
 79    TIM4->CCR4 = 0;
 80  }
 81
 82  void move_right(void)
 83  {
 84    TIM4->CCR1 = 65;
 85    TIM4->CCR2 = 0;
 86    TIM4->CCR3 = 0;
 87    TIM4->CCR4 = 25;
 88  }
 89
 90  void move_left(void)
 91  {
 92    TIM4->CCR1 = 25;
 93    TIM4->CCR2 = 0;
 94    TIM4->CCR3 = 0;
 95    TIM4->CCR4 = 65;
 96  }
 97
 98  void stop(void)
 99  {
100    TIM4->CCR1 = 0;
101    TIM4->CCR2 = 0;
102    TIM4->CCR3 = 0;
103    TIM4->CCR4 = 0;
104  }
105
106  // void tilt_right(void)
107  // {
108  //    TIM4->CCR1 = 50;
109  //    TIM4->CCR2 = 0;
110  //    TIM4->CCR3 = 0;
111  //    TIM4->CCR4 = 20;
112  // }
113  //
114  // void tilt_left(void)
115  // {
116  //    TIM4->CCR1 = 20;
117  //    TIM4->CCR2 = 0;
118  //    TIM4->CCR3 = 0;
119  //    TIM4->CCR4 = 50;
120  // }
121
122  //void findLine(void)
123  //{
124  //    move_backward();
125  //    delay(1000000);
126  //    int i;
127  //    while(1)
128  //    {
129  //        i=500000;
130  //        tilt_right();
131  //        delay(i);
132  //        stop();
133  //
134  //        if(((IR_Left_Sensor()) && (IR_Right_Sensor())) == 0x00)
135  //        {
136  //          return;
137  //        }
138  //        tilt_left();
139  //        delay(i*2);
140  //        stop();
141  //
142  //        if(((IR_Left_Sensor()) && (IR_Right_Sensor())) == 0x00)
143  //          return;
144  //
```

```
145   //
146   //        i=i+(2*500000);
147   //    }
148   //}
149
150   void lineFollowing()
151   {
152       while(1)
153       {
154         if(((IR_Left_Sensor()) && (IR_Right_Sensor())) == 0x00)
155         {
156             StartLCD();
157             move_forward();
158         }
159         if(((IR_Left_Sensor()) && (IR_Right_Sensor())) != 0x00)
160         {
161             BothSensorTriggered();
162             stop();
163             delay(1000000);
164             move_backward();
165             delay(1000000);
166             stop();
167             delay(1000000);
168             move_left();
169             delay(1000000);
170
171             if(((IR_Left_Sensor()) || (IR_Right_Sensor())) == 0x00)
172               return;
173
174             move_right();
175             delay(500000);
176
177             if(((IR_Left_Sensor()) || (IR_Right_Sensor())) == 0x00)
178               return;
179
180
181             return;
182         }
183         if(IR_Left_Sensor() == 0xFFF)
184         {
185             LeftSensorTriggered();
186             stop();
187             delay(1000000);
188             move_backward();
189             delay(1000000);
190             move_right();
191             return;
192         }
193         if(IR_Right_Sensor() == 0xFFF)
194         {
195             RightSensorTriggered();
196             stop();
197             delay(1000000);
198             move_backward();
199             delay(1000000);
200             move_left();
201             return;
202         }
203         if((Flame_Sensor() == 0xFFF))
204         {
205             FlameSensorTrig();
206             stop();
207             delay(4000000);
208
209         }
210
211       }
212   }
213
```