```c
//Kunal Mukherjee;
//3/26/2019
//Project 2: HI2C

#include "stm32l432.h"
#include <stdint.h>

void GPIO_Init(void);
void ADC_Init(void);
void DMA_Init(void);

int ADC_Results[2] = {0};

#define ADC_value_max 4096

int main()
{
  int value1 = 0, value2 = 0, i = 0;
  GPIO_Init();
  ADC_Init();
  DMA_Init();

  while(1)
  {
    ADC_CR |= (1 << 2); //start adc regular conversion
    while((ADC_ISR & (1 << 3)) == 0);

    value1 = ADC_Results[0] & 0xFFF; //only look at 12 bit
    value2 = ADC_Results[1] & 0xFFF; //only look at 12 bit

//    ADC_ISR |= (1 << 3);

    TIM2_CCR2 = (int)((((float)(value1 + value2))/((float)(2 * ADC_value_max))) * 8096); //scale the
value

    for (i =0; i<10000; i++); //have a delay
  }
  return 0;
}

void GPIO_Init(void)
{
  RCC_AHB2ENR |= (1 << 1); //set the GPIOB clk
  RCC_APB1ENR1 |= (1 << 0); //IO port B clock enable

  GPIOB_MODER &= ~(3 << (2 * 3)); //clear the GPIOB mode bits
  GPIOB_MODER |= (2 << (2 * 3));  //set port b3 is alternate 10
  GPIOB_AFRL |= (1 << (4 * 3));   //alt func 1, port pin 3,
                                  //control bit are 4 bit wide

  TIM2_CR1 |= (1 << 7); //ARPE: Auto-reload preload enable
  TIM2_PSC = 0;         //PSC set to 0
  TIM2_ARR = 8192;     //4MHz/8192 = 488 Hz = 2.05 ms

  TIM2_CCMR1 |= 0x6800; //Channel 2;
                        //bit 11: OC2PE: Output compare 2 preload enable
                        //0110: PWM mode 1 - In upcounting,
                        //channel 1 is active as long as
                         //TIMx_CNT<TIMx_CCR1else inactive.

  TIM2_CCER |= (1 << 4); //CC2E: Capture/Compare 2 output enable.

  TIM2_CCR2 |= 0;     //CCR2 is the value to be loaded in the actual
                         //capture/compare 2 register (preload value).

  TIM2_EGR |= (1 << 0); //UG: update event

  TIM2_CR1 |= (1 << 0); //CEN: counter enabled
}

void ADC_Init(void)
{
  int i;
  RCC_AHB2ENR |= (1 << 0); //set the GPIOA clk
  RCC_AHB2ENR |= (1 << 13); //set ADC clk

  //mode default to analog
  GPIOA_PUPDR &= ~(1  << 1); //PA1 to no pull or down
  GPIOA_PUPDR &= ~(1  << 2); //PA2 to no pull or down

  ADC_CR &= ~(1 << 29); //deep power mode cleared
  ADC_CR |= (1 << 28); //set voltage reg

  for (i=0; i <10000; i++);//wait for .5us

  ADC_CCR |= (1 << 22); //VREF ENAB
  ADC_CCR |= (1 << 16); //HCLK/1 (Synchronous clock mode) enb

  ADC_ISR |= (1 << 0); //ADC ready
  ADC_CR |= (1 << 0); //ENB ADC
  while ((ADC_ISR & (1 << 0)) == 0); //wait till ADC is ready

  ADC_SQR1 |= (1 << 0);//Channel length 2
  ADC_SQR1 |= (6 << 6);//CH6
  ADC_SQR1 |= (7 << 12);//CH7

  ADC_CFGR |= (1 << 0); //DMAEN
  ADC_CFGR |= (1 << 1); //DMACFG: Direct memory access configuration
  ADC_CFGR |= (1 << 16); //DISCEN: Discontinuous mode for regular channels

  ADC_CR |= (1 << 2); //start adc regular conversion
}

void DMA_Init(void)
{ //enable DMA1 clocl
  RCC_AHB1ENR |= RCC_AHB1ENR_DMA1EN;

  //DMA channel 1 configuration for ADC6
  DMA1_Channel1_CCR &= ~DMA_CCR_MEM2MEM; //diabale memory to memory mode

  //Channel priority level
  //00-low, 11-very high
  DMA1_Channel1_CCR &= ~DMA_CCR_PL;
  DMA1_Channel1_CCR |= DMA_CCR_PL_1; //high priority

  //peripheral size:01 = 16 bits
  DMA1_Channel1_CCR &= ~DMA_CCR_PSIZE;
  DMA1_Channel1_CCR |= DMA_CCR_PSIZE_0;

  //memory size:01 = 16 bits
  DMA1_Channel1_CCR &= ~DMA_CCR_MSIZE;
  DMA1_Channel1_CCR |= DMA_CCR_MSIZE_0;

  //peripheral increment mode disable
  DMA1_Channel1_CCR &= ~DMA_CCR_PINC;

  //memory increment mode able
  DMA1_Channel1_CCR |= DMA_CCR_MINC;

  //circular mode enable
  DMA1_Channel1_CCR |= DMA_CCR_CIRC;

  //Data transfer direction - read
  DMA1_Channel1_CCR &= ~DMA_CCR_DIR;

  //Number of data to trnasfer
  DMA1_Channel1_CNDTR = 3; //length of ADC sequence 2
                           // I know it is supposed to be 2
                           //but 2 was not working, ADC_DR was
                           //not being copied to result[], but
                           //making the sequence 3, it is working

  //Peripheral address regsiter
  DMA1_Channel1_CPAR = (uint32_t)&(ADC_DR);

  //memeory address register
  DMA1_Channel1_CMAR = (uint32_t) ADC_Results;

  //DMA Channel Selection
  //MAP DMA channel 1 to ADC1
  DMA1_CSELR &= ~DMA_CSELR_C1S; //0000 CH1 mapped to ADC1

  //enable DMA channel
  DMA1_Channel1_CCR |= DMA_CCR_EN;
}
```