Ultrasonic with Color sensor

By Thinking Team

MEMBERS

ดรณ์ ชัยสิริมั่นคง	2011310170
ณัชพล ศรีวิทะ	2011310279
ชญาณดา ศรีวิศรุตชน	2011310733
ชินวัตร ชินณรงค์	2011310592
อรนรินทร์ ปรีดีเพชรพงศ์	2011370017

จารย์กัน แห่ง CE13

Objective and Requirements

- เพื่อศึกษาเกี่ยวกับโปรแกรม
- เพื่อศึกษาเกี่ยวกับ sensor
- เพื่อวัดระยะห่างระหว่างสิ่งของ

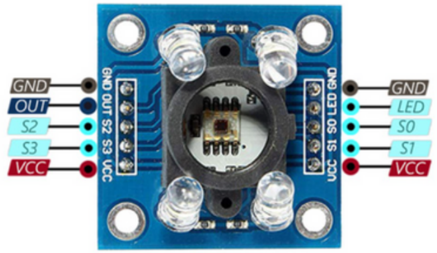
- ให้ Ultrasonic ทำงานเมื่อมีวัตถุเข้ามาใกล้ เมื่อใกล้ระยะที่กำหนด Speaker จะส่งเสียงออกมา พร้อมกับ Color sensor ทำงาน

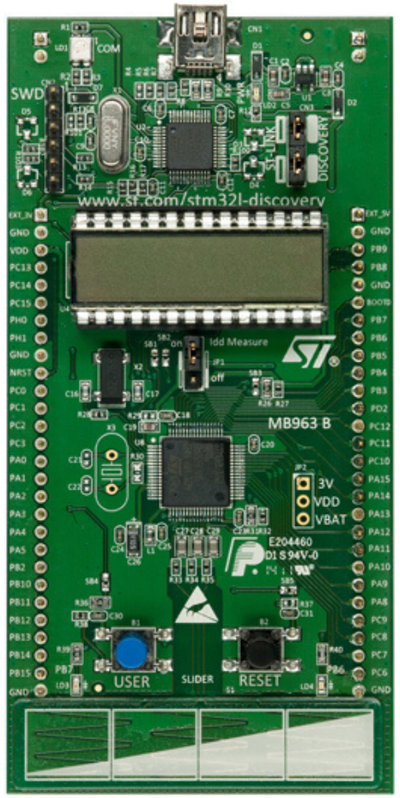
GANTT CHART

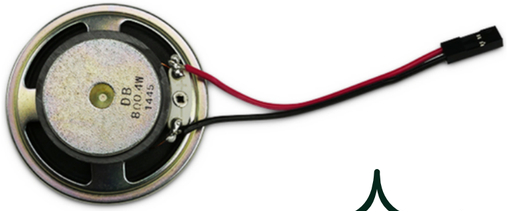
Activity / Duration(week)	Role	W1	W2	W3	W4	W5	W6	W7	W8	W9
1.Planning Project	Team									
1.1. กำหนดความต้องการ										
1.2. กำหนดระบบ										
1.3. กำหนดอุปกรณ์										
1.4. กำหนดแผนการทำงานในทีม										
2. Prepare equipments	Team									
2.1. เตรียมอุปกรณ์	Ornarin/Don									
2.2. ออกแบบอุปกรณ์										
2.3. ออกแบบฐานข้อมูล										
3. Start Project	Team									
3.1. เขียนโค้ด	Nutchapon/Chinnawat/Chayanada									
3.2. ดีบัค	Ornarin/Don									
3.3. ทดสอบ	Nutchapon/Ornarin									

HARDWARE COMPONENT LISTS

Hardware component lists	
Component list	Qty.
STM32L152RB board	1
USART	1
Ultrasonic senser	1
Color senser	1
Speaker	1





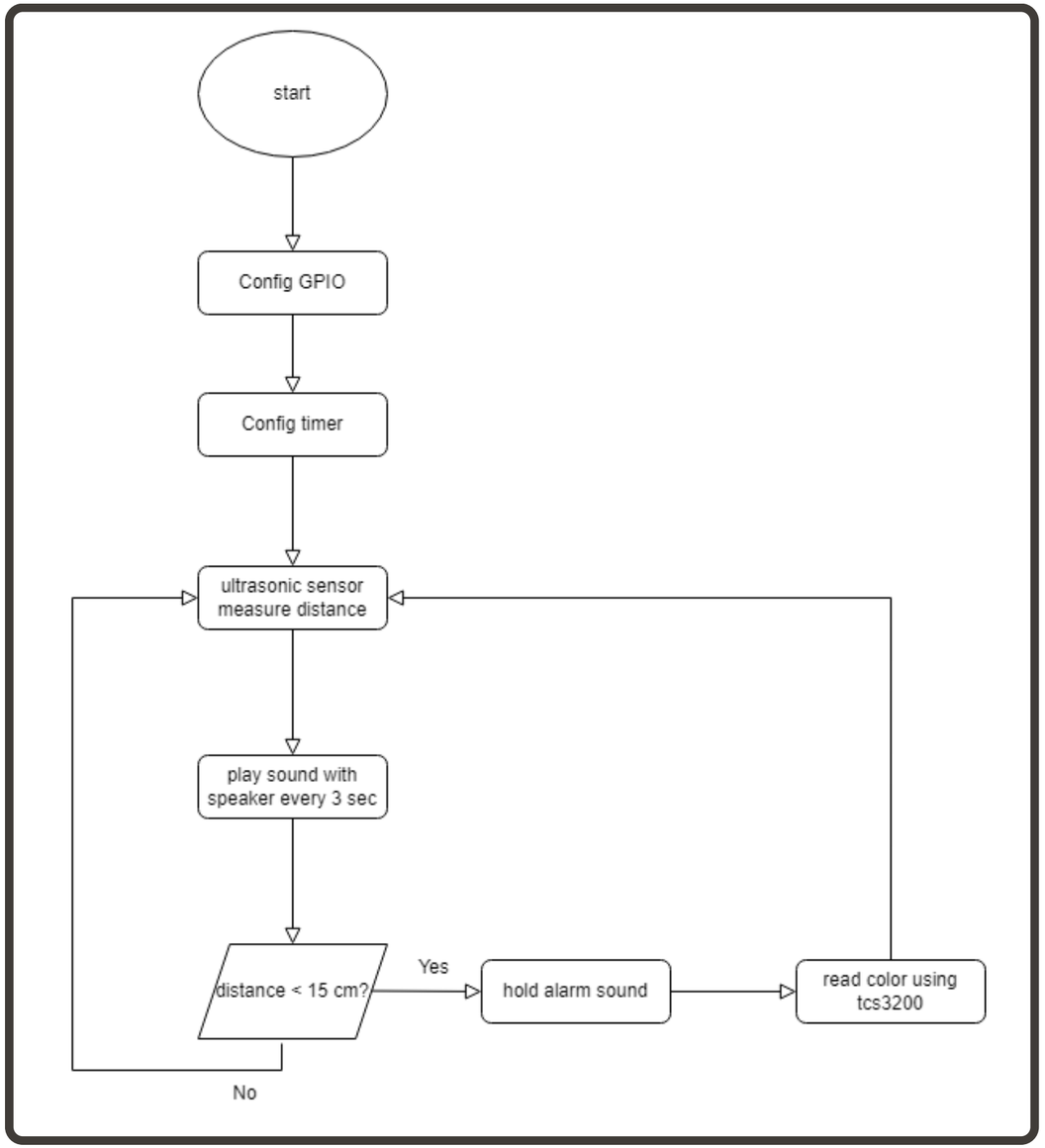


HARDWARE & GPIO SELECTED

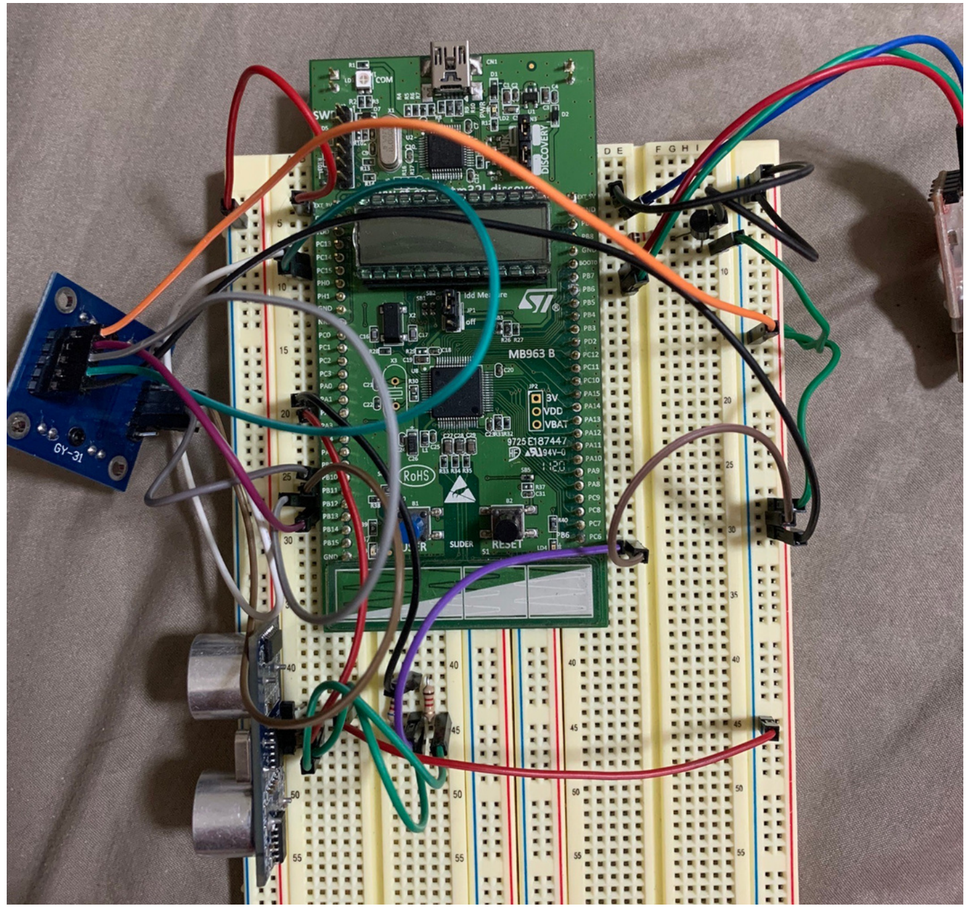
Port/Pin	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A		Trig	Echo													
B							TX	RX	SPK		S0	S1	S2	S3		
C							OUT									
D																
H																

USART_2
ULTRASONIC
COLOR
SPEAKER

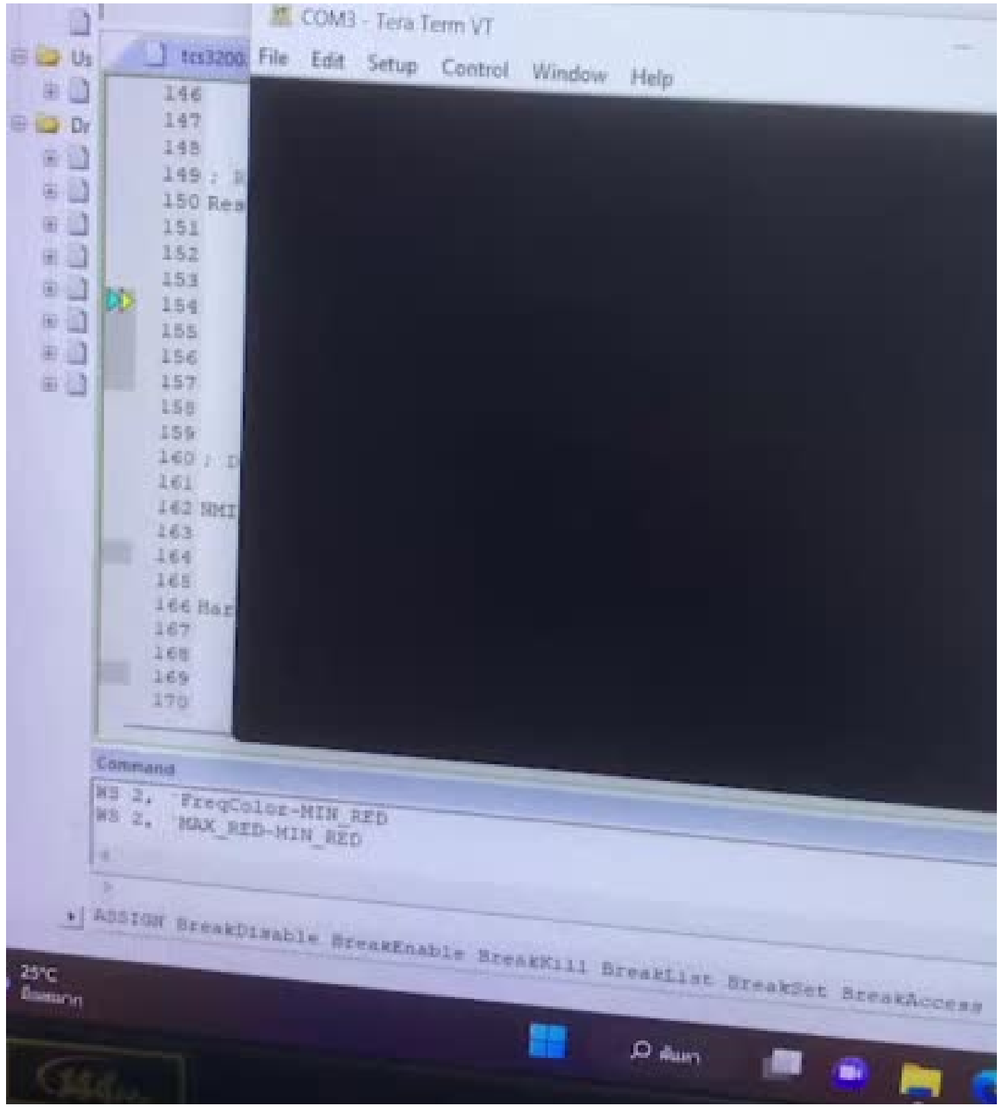
FLOAT CHART


start
Config GPIO
Config timer
ultrasonic sensor measure distance
play sound with speaker every 3 sec
distance < 15 cm?
Yes
hold alarm sound
read color using tcs3200
No

CIRCUIT



PREVIEW


COM3 - Tera Term VT
File Edit Setup Control Window Help
Command
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

CODE

```c
#include <stdbool.h>
#include "stm32l1xx.h"
#include "tcs3200.h"
#include "stm32l1xx_ll_gpio.h"
#include "stm32l1xx_ll_tim.h"
#include "stm32l1xx_ll_bus.h"

#include "stm32l1xx_ll_utils.h"

bool IC_ColorMode = false;
uint8_t calibrate_number;

uint16_t TimeColor_H=0, TimeColor_L=0;
uint16_t TimeColor;
uint16_t FreqColor = 0;

void Captura_TCS3200_Init(void)
{
        LL_GPIO_InitTypeDef GPIO_InitStructure;
        LL_TIM_InitTypeDef TIM_TimeBaseInitStructure;
        LL_TIM_IC_InitTypeDef TIM_ICInitStructure;
        //NVIC_InitTypeDef NVIC_InitStructure;

        /* GPIOB clock enable */
        LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
        /*-------------------------- GPIO Configuration ----------------------------*/
        /* GPIOB Configuration: PB0 as input for capture */
        GPIO_InitStructure.Pin = LL_GPIO_PIN_3;
        GPIO_InitStructure.Mode = LL_GPIO_MODE_ALTERNATE;
        GPIO_InitStructure.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
        GPIO_InitStructure.Pull = LL_GPIO_PULL_UP;

        GPIO_InitStructure.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStructure.Alternate = LL_GPIO_AF_3;
        LL_GPIO_Init(GPIOA, &GPIO_InitStructure);
        /* Connect TIM4 pins to AF2 */
        //GPIO_PinAFConfig(GPIOB, GPIO_PinSource0, GPIO_AF_TIM3); //TIM3 CC3 -> PB0

        /*Activo Clock para el periferico del timer*/
        LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_TIM9);
        // RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

        /*Configuro la base de tiempos del timer*/
        TIM_TimeBaseInitStructure.CounterMode = LL_TIM_COUNTERMODE_UP;
        TIM_TimeBaseInitStructure.Autoreload = 0xFFFF;
    TIM_TimeBaseInitStructure.Prescaler = 32-1; //Resolución de 0.001ms = 1       us
        // TIM_TimeBaseInitStructure.ClockDivision = 0;

        LL_TIM_Init(TIM9, &TIM_TimeBaseInitStructure);

        TIM_ICInitStructure.ICActiveInput = LL_TIM_ACTIVEINPUT_DIRECTTI;
        TIM_ICInitStructure.ICFilter = LL_TIM_IC_FILTER_FDIV1_N2;

        //CHANNEL 3 -> SUBIDA
        //TIM_ICInitStructure.Channel = TIM_Channel_3;
        TIM_ICInitStructure.ICPolarity = LL_TIM_IC_POLARITY_RISING;
        //TIM_ICInitStructure.ICSelection = TIM_ICSelection_DirectTI;
        TIM_ICInitStructure.ICPrescaler = LL_TIM_ICPSC_DIV2;
        //TIM_ICInitStructure.ICFilter = 5;
    TIM_ICInitStructure.ICFilter = LL_TIM_IC_FILTER_FDIV1_N2;

        LL_TIM_IC_Init(TIM9,LL_TIM_CHANNEL_CH2,&TIM_ICInitStructure);

        LL_TIM_EnableIT_CC2(TIM9);

        //Configuro interrupcion en el TIM3 CC3
        //TIM_ITConfig(TIM3, TIM_IT_CC3, ENABLE);

        NVIC_SetPriority(TIM9_IRQn, 0);
        NVIC_EnableIRQ(TIM9_IRQn);
         LL_TIM_CC_EnableChannel(TIM9, LL_TIM_CHANNEL_CH2);
        //LL_TIM_EnableIT_CC2(TIM3);
        LL_TIM_EnableCounter(TIM9);
        /*
        TIM_Cmd(TIM3, ENABLE);

        //Configurar interrupción del Channel 4 (BAJADA) del TIM3 -> NVIC
        NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;

        NVIC_Init(&NVIC_InitStructure);*/
}
```

CODE

```
94  void TCS3200_Config(void)
95  {
96          //Configuración de los pines de entrada y salida para configurar el filtro y el preescaler
97          //GPIOB
98          //S0 -> GPIO_Pin_1 10
99          //S1 -> GPIO_Pin_2 11
100         //S2 -> GPIO_Pin_3 12
101         //S3 -> GPIO_Pin_4 13
102
103         LL_GPIO_InitTypeDef GPIO_InitStructure;
104
105         /* GPIOB clock enable */
106         LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);
107         /*------------------------- GPIO Configuration ----------------------------*/
108         /* GPIOB Configuration: PB0 como entrada para captura */
109         GPIO_InitStructure.Pin = LL_GPIO_PIN_10 | LL_GPIO_PIN_11 | LL_GPIO_PIN_12 | LL_GPIO_PIN_13;
110         GPIO_InitStructure.Mode = LL_GPIO_MODE_OUTPUT;
111         GPIO_InitStructure.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
112         GPIO_InitStructure.Pull = LL_GPIO_PULL_NO;
113         GPIO_InitStructure.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
114         LL_GPIO_Init(GPIOB, &GPIO_InitStructure);
115 }
116
117 void Set_Filter (uint8_t mode) //Mode es de tipo enum Filtro
118 {
119         switch (mode){
120                 case(Red):
121                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_12 | LL_GPIO_PIN_13);
122                         break;
123                 case(Blue):
124                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_12);
125                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_13);
126                         break;
127                 case(Clear):
128                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_13);
129                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_12);
130                         break;
131                 case(Green):
132                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_12 | LL_GPIO_PIN_13);
133                         break;
134         }
135 }
136
```

```
137 void Set_Scaling (uint8_t mode) //Mode es de tipo enum Filtro
138 {
139         switch (mode){
140                 case(Scl0):
141                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_10 | LL_GPIO_PIN_11);
142                         break;
143                 case(Scl2):
144                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_10);
145                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_11);
146                         break;
147                 case(Scl20):
148                         LL_GPIO_ResetOutputPin(GPIOB, LL_GPIO_PIN_11);
149                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_10);
150                         break;
151                 case(Scl100):
152                         LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_11 | LL_GPIO_PIN_11);
153                         break;
154         }
155 }
```

CODE

```
int Output_Color;

int GetColor(int set_color) //Funcion que Devuelve RGB de color Rojo (0-255)
{
        //char Output_Color;

        calibrate_number=0;

        TimeColor_H=0;
        TimeColor_L=0;
        TimeColor=0;

        LL_TIM_EnableIT_CC2(TIM9);
        //TIM_ITConfig(TIM3, TIM_IT_CC3, ENABLE); //Enable interruption

        Set_Filter(set_color); //Set filter to Color
        //DelayMillis(100);
        LL_mDelay(100);
        IC_ColorMode = true;

        while(IC_ColorMode == true); //Wait until value is get on the interrupt routine

        Set_Filter(Clear); //Set filter to default


        //TIM_ITConfig(TIM3, TIM_IT_CC3, DISABLE); //Disable interruption
        LL_TIM_DisableIT_CC2(TIM9);

        /*Timing calculation -> Get Color Value*/
        if(TimeColor_H > TimeColor_L) //Avoid overflow
                TimeColor = TimeColor_H - TimeColor_L;
        else
                TimeColor = 0xFFFF - TimeColor_L + TimeColor_H;


        //FreqColor = SystemCoreClock/(TimeColor*84); //Frequency conversion by means of SystemCoreClock
        FreqColor= 10000*SystemCoreClock/(TimeColor*84); //168

        //Freq to Color -> Depending of the filter
        switch (set_color){
                case Red:
                        Output_Color = 50*(255.0/(16400.0-500.0))*(FreqColor-500.0); //MAPEAR FUNCIÓN
                        break;
```

CODE

```
200
201                 case Green:
202                         Output_Color = 50*(255.0/(11000.0-700.0))*(FreqColor-700.0);  //MAPEAR FUNCIÓN
203                         break;
204
205                 case Blue:
206                         Output_Color = 50*(255.0/(10000.0-600.0))*(FreqColor-600.0);  //MAPEAR FUNCIÓN
207                         break;
208         }
209
210         //Constrain Value to MaxRange
211         if (Output_Color > 255) Output_Color = 255;
212         if (Output_Color < 0) Output_Color = 0;
213
214         return Output_Color     ; //Mapeo y retorno valor
215 }
216
```

```
217 void TIM9_IRQHandler(void)
218 {
219         //Manejador de interrupción del TIM3
220         if(LL_TIM_IsActiveFlag_CC2(TIM9) == SET)
221         {
222                 LL_TIM_ClearFlag_CC2(TIM9);
223                 if((IC_ColorMode == true) && (calibrate_number == 0))
224                 {
225                         TimeColor_L = LL_TIM_IC_GetCaptureCH2(TIM9);
226                         //TimeColor_L = TIM_GetCapture3(TIM3);
227                         calibrate_number = 1;
228                 }
229                 else if((IC_ColorMode == true) && (calibrate_number == 1))
230                 {
231                         TimeColor_H = LL_TIM_IC_GetCaptureCH2(TIM9);
232                         //TimeColor_H = TIM_GetCapture3(TIM3);
233                         IC_ColorMode = false;
234                         calibrate_number = 0;
235                 }
236
237         /*if (TIM_GetITStatus(TIM3,TIM_IT_CC3)!= RESET){
238                 //Clear flag is the first statement
239                 TIM_ClearITPendingBit(TIM3, TIM_IT_CC3);
240
241                 if((IC_ColorMode == true) && (calibrate_number == 0))
242                 {
243                         TimeColor_L = TIM_GetCapture3(TIM3);
244                         calibrate_number = 1;
245                 }
246                 else if((IC_ColorMode == true) && (calibrate_number == 1))
247                 {
248                         TimeColor_H = TIM_GetCapture3(TIM3);
249                         IC_ColorMode = false;
250                         calibrate_number = 0;
251                 }*/
252         }
```

Problems and Solutions

ค่าความถี่ที่ใช้อ่านจาก color sensor ไป stm32L152RB	ศึกษาดู data sheet เพื่อนำมาปรับแก้ไข
TCS3200อ่านค่าไม่คงที่	ศึกษาดู data sheet เพื่อนำมาปรับแก้ไข

THANK YOU!

Do you have any questions
before we go?