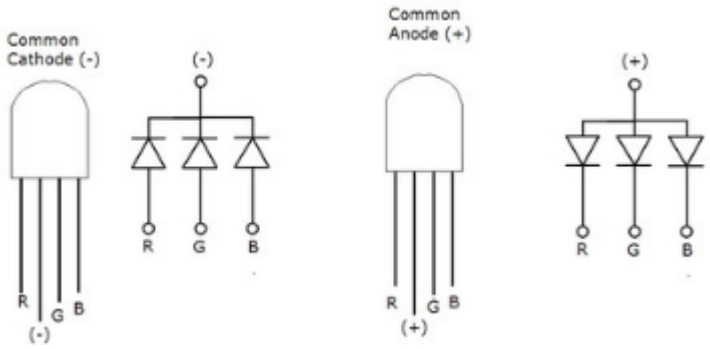
Case Study
RGB, Scheduler

Obiective

- Utilizarea LED-ului RGB
- Prezentarea conceptului de “Task Scheduler”

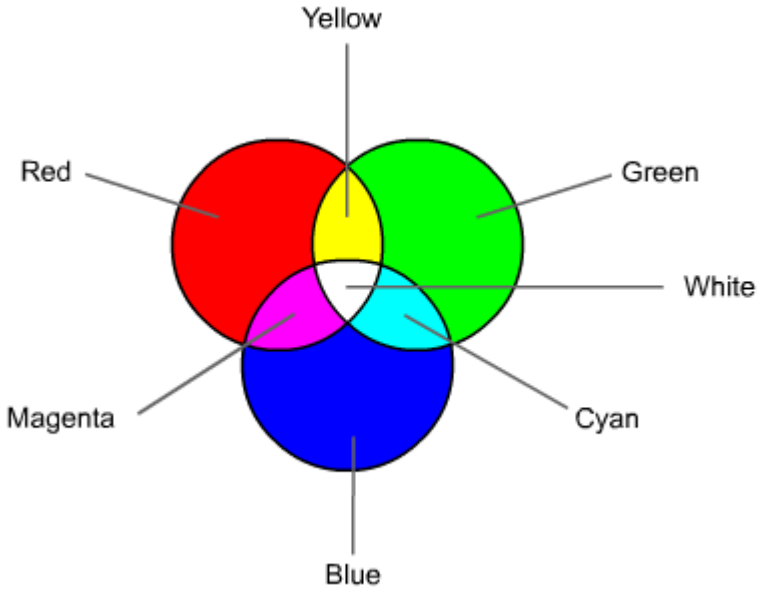
Cuprins

RGB



Un LED RGB este un LED format din 3 diode luminiscente de culori diferite (red, green, blue – „roșu”, „verde”, „albastru”). Acestea pot fi conectate în 2 moduri: catod comun sau anod comun. În conexiunea „catod comun”, toate diodele au catodul conectat în același punct, în timp ce în conexiunea „anod comun”, diodele au anodul conectat în același punct.

Pentru a se aprinde o singură culoare se va deshide o singură diodă, celelalte fiind blocate. Pentru a obține combinații diferite de culori se vor deschide câte 2 diode simultan, iar a 3-a va fi blocată. Pentru a obține culoarea „alb”, se vor deschide toate cele 3 diode.



Cele 3 terminale ale LED-ului RGB se vor conecta la 3 din ieșirile registrului de shiftare prin intermediul unor rezistențe pentru a limita curentul prin diode, iar pinul cel mai lung se va conecta la masă/alimentare (catod comun/anod comun).

Task Scheduler

Dupa cum am vazut in laboratoarele anterioare, proiectul este structurat pe 4 nivele: BSW, RTE, ASW si SRVL. În cadrul BSW întâlnim, de asemenea, 2 ramificații: MCAL și HAL. Fiecare strat are acces la stratul inferior, dar nu poate accesa stratul superior, exceptând stratul RTE care este folosit ca intermediar între BSW și ASW și stratul SRVL în care regăsim „Task Scheduler-ul”.

„Task Scheduler-ul” este mecanismul prin care putem asigura o executare a sarcinilor la intervale precise de timp. În cadrul proiectului nostru vom simula un asemenea „Task Scheduler” contorizand numărul de bucle prin care trece programul, o buclă durand 100ms, obținând astfel executări ale sarcinilor cu o precizie de până la 100ms.

```
107 void SYSTEM_vTaskScheduler(void)
108 {
109     uint16_t u16TickCount = 0;
110 
111     while (1)
112     {
113         if (u16TickCount % TASK_100MS == 0)
114         {
115             vTask100ms();
116         }
117 
118         if (u16TickCount % TASK_200MS == 0)
119         {
120             vTask200ms();
121         }
122 
123         if (u16TickCount % TASK_500MS == 0)
124         {
125             vTask500ms();
126         }
127 
128         if (u16TickCount % TASK_800MS == 0)
129         {
130             vTask800ms();
131         }
132 
133         if (u16TickCount % TASK_1000MS == 0)
134         {
135             vTask1000ms();
136         }
137 
138         if (u16TickCount % TASK_2000MS == 0)
139         {
140             vTask2000ms();
141         }
142 
143         if (u16TickCount % TASK_5000MS == 0)
144         {
145             vTask5000ms();
146         }
147 
148         u16TickCount++;
149         if (u16TickCount >= TASK_5000MS)
150         {
151             u16TickCount = 0;
152         }
153 
154         vTaskDelay(10);
155     }
156 }
```