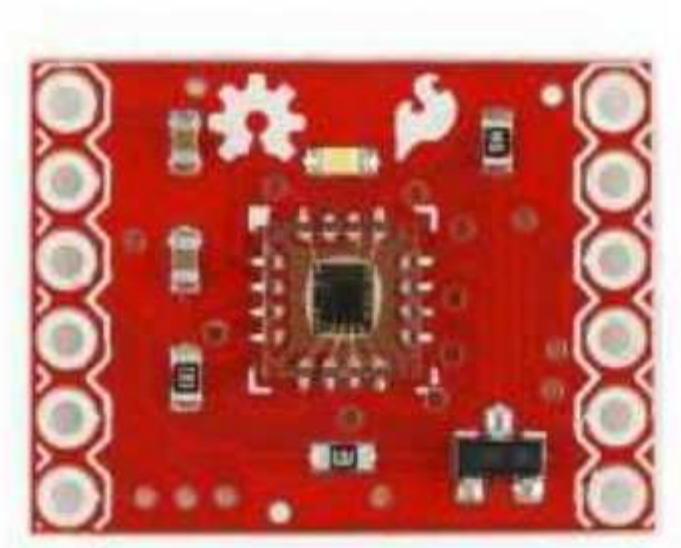


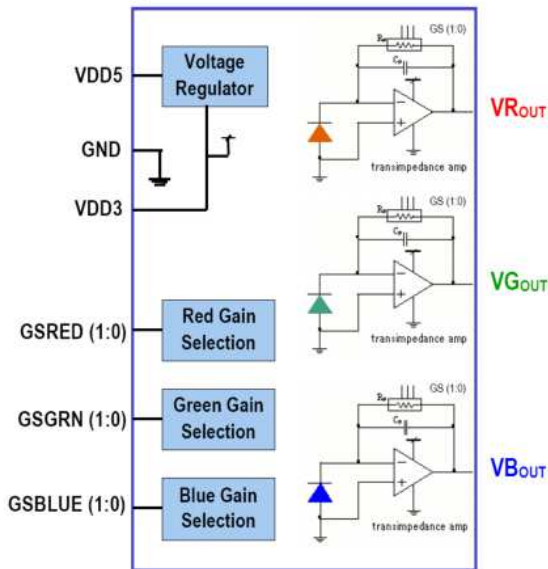
TD Arduino



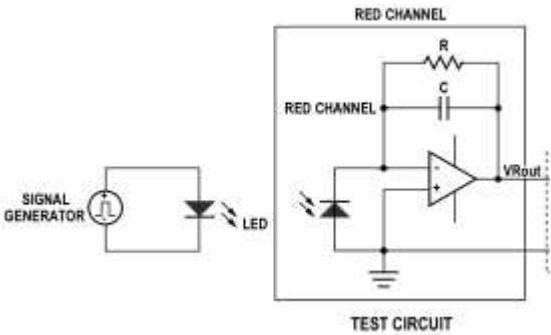
HDJD-S822-QR999
RGB Color Sensor



Sensor IC Block Diagram



Test Circuit



Operating Conditions and Electrical Requirements

Electrical Characteristics at $V_{DD} = 5V$, $T_A = 25^{\circ}C$, $R_L = 68k\Omega$

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Dark voltage	V_D	$E_e = 0$			15	mV
Maximum output voltage swing	$V_{O\ MAX}$			3		V
Supply current	I_{DD}	$E_e = 0$		3		mA
Parameter	Symbol	Remark	Min	Typ	Max	Unit
Irradiance Responsivity	R_e	GS:00 $\lambda_p = 460\text{ nm}$ ^[1] (Blue Channel)		1.54		V/(mW/cm ²)
		GS:00 $\lambda_p = 542\text{ nm}$ ^[2] (Green Channel)		2.05		
		GS:00 $\lambda_p = 645\text{ nm}$ ^[3] (Red Channel)		2.73		

Typical Characteristics

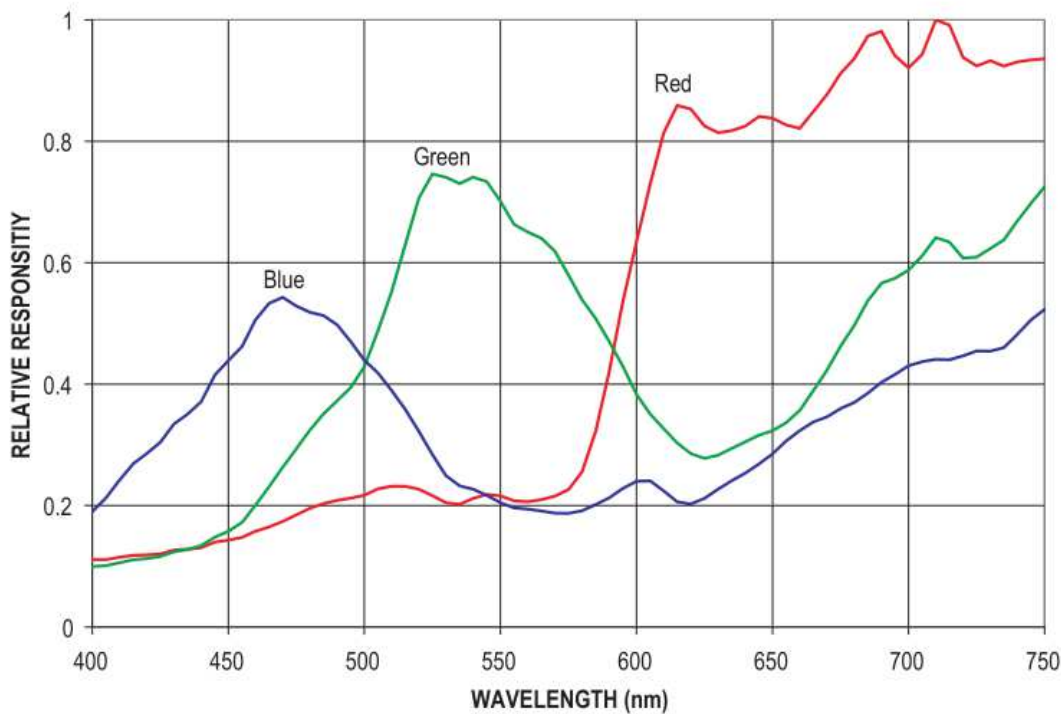


Figure 1. Spectral Responsivity

```

1
2 /* Colorimétrie : Mesure de couleur RVB
3 Mesures cumulées
4 Correction selon feuilles de données
5 Normalisation à 255
6 */
7
8 // Gestion des entrées du capteur de couleurs SparkFun Color Sensor Breakout - HDJD-S822
9 // 0 Bleu entrée sur A0, 1 Vert entrée sur A1, 2 Rouge entrée sur A2
10 #define ledPin 5 // Commande LED sur D5 Led blanche de test
11
12 // Calcul des correction sur la base des sensibilité relatives (graphe p7)
13 // Bleu 460 nm Sb=0.48 48% Vert 542 nm Sg=0.75 Rouge 645 nm Sr=0.85
14 const float CoefCorrection[3]={0.48 , 0.75, 0.85}
15
16 int ColorRdg[3]; // triplet lecture brute
17 float ColorC[3]; // triplet corrigé selon feuilles de données p7 du capteur
18 float ColorN[3]; // triplet corrigé normalisé
19
20 float ColorMax;
21 // -----
22
23 void setup()
24 {
25   Serial.begin(9600);
26   analogReference(EXTERNAL); // AREF relié à 3.3V sur le shield (donc 1023 correspond à 3.3V)
27   pinMode(ledPin, OUTPUT);
28 } // fin de setup
29
30 void loop(void)
31 {
32   // Initialisation d'un cycle de lecture
33   ColorRdg[0]=ColorRdg[1]=ColorRdg[2]=0;
34   digitalWrite(ledPin, HIGH); // éclairage de l'objet coloré
35   delay(500);
36
37   byte n=0; byte i;
38   while (n<10) { // cumul de 10 lectures successives pour effet de lissage
39     for (i=0; i<3;i++) {
40       ColorRdg[i] = analogRead(i) + ColorRdg[i] ;
41     }
42     //delay(100);
43     n=n+1;
44   } // Fin des 3*10 lectures
45

```

```

46 // Correction
47 for (i=0; i<3;i++) {
48     ColorC[i] = ColorRdg[i]/CoefCorrection[3]; // reel = entier/reel
49 }
50
51 // Recherche de la valeur max dans le triplet corrigé (au moins 6 lignes avec des if)
52 ColorMax = max(max(ColorC[0],ColorC[1]), ColorC[2]);
53
54 // Normalisation à 255 pour chaque triplé mesuré
55 for (i=0; i<3;i++) {
56     ColorN[i] = 255.0*ColorC[i]/ColorMax;
57 }
58
59 digitalWrite(ledPin, LOW); // Eteindre la LED
60
61 // affichage des résultats etc.
62 Serial.print("Bleu : "); Serial.print(ColorN[0]);
63 Serial.print(" Vert : "); Serial.print(ColorN[1]);
64 Serial.print(" Rouge : "); Serial.println(ColorN[2]);
65 Serial.println("");
66
67 delay (2000) // pause
68 } // Fin de loop
69

```

De l'intérêt d'une fonction ad hoc :

```

// Recherche de la valeur max dans le triplet corrigé
ColorMax = max(max(ColorC[0],ColorC[1]),ColorC[2]);

71 // Variante Choix du max avec des if
72
73 if(ColorC[0]<=ColorC[1]){ // test logique <= fig12 p 17
74     if(ColorC[1]<=ColorC[2])
75         {ColorMax=ColorC[2];}
76     else
77         {ColorMax=ColorC[1];} // affectation de valeur
78 }
79 else {
80     if(ColorC[0]<=ColorC[2])
81         {ColorMax=ColorC[2];}
82     else
83         {ColorMax=ColorC[0];}
84 }

```

Réalisation d'un luxmètre basé sur BH1750

```
33 // Variables globales *****
34 const byte nMax = 10; // nombre de mesures pour la moyenne glissée sur les nMax dernières mesures
35 int tabLux[nMax]; // pour le stockage des nMax dernières valeurs
36 byte n; // indice du tableau de valeurs tabLux[n], donc compteur cyclique de la boucle principale
37 int luxMoyen; // pour la boucle For de calcul de la moyenne

85 //
86 // Initialisation du tableau tabLux à des valeurs nulles
87   for (byte i=0; i<nMax; i=i+1) //
88       {tabLux[i]=0;}
89 // Initialisation du compteur de boucle n
90   n=0;
91

97 void loop()
98 {
99 // Mise à jour cyclique des nMax valeurs du tableaux
100   tabLux[n] = luxMeter.readLightLevel();
101   Serial.print("tabLux["); Serial.print(n); Serial.print("] : ");
102   Serial.print(tabLux[n]); Serial.println(" lx");
103
104 // Calcul de la moyenne glissée sur les nMax dernières valeurs du tableau tabLux
105   luxMoyen=luxMoyenne(nMax, tabLux);
106   Serial.print("luxMoyen : "); Serial.println(luxMoyen);
107
108 // Affichage OLED sur deux lignes de la valeur instantanée et de la moyenne glissée
109   displayResult(tabLux[n], luxMoyen);
110
111 // Gestion du compteur de boucle principale
112   n=n+1; // pour élément suivant
113   if (n==nMax){ n=0;}
114   // après le nMaxième, on rafraichit le premier élément du tableau etc.
115   // dans le test logique : égalité numérique == et pas affectation de valeur =
116
117
118 } // Fin de Loop()
119 //*****

39 // Fonctions et procédures *****
40 // Calcul de la moyenne glissée sur les nMax
41 // dernières valeurs du tableau local _tabLux
42 int luxMoyenne(byte _nMax, int _tabLux[])
43 {
44   int _cumulLux = 0;
45   for (byte i=0; i<nMax;i=i+1)
46   {
47     _cumulLux=_cumulLux+_tabLux[i];
48   }
49   return round(1.0*_cumulLux/_nMax);
50 }
```

