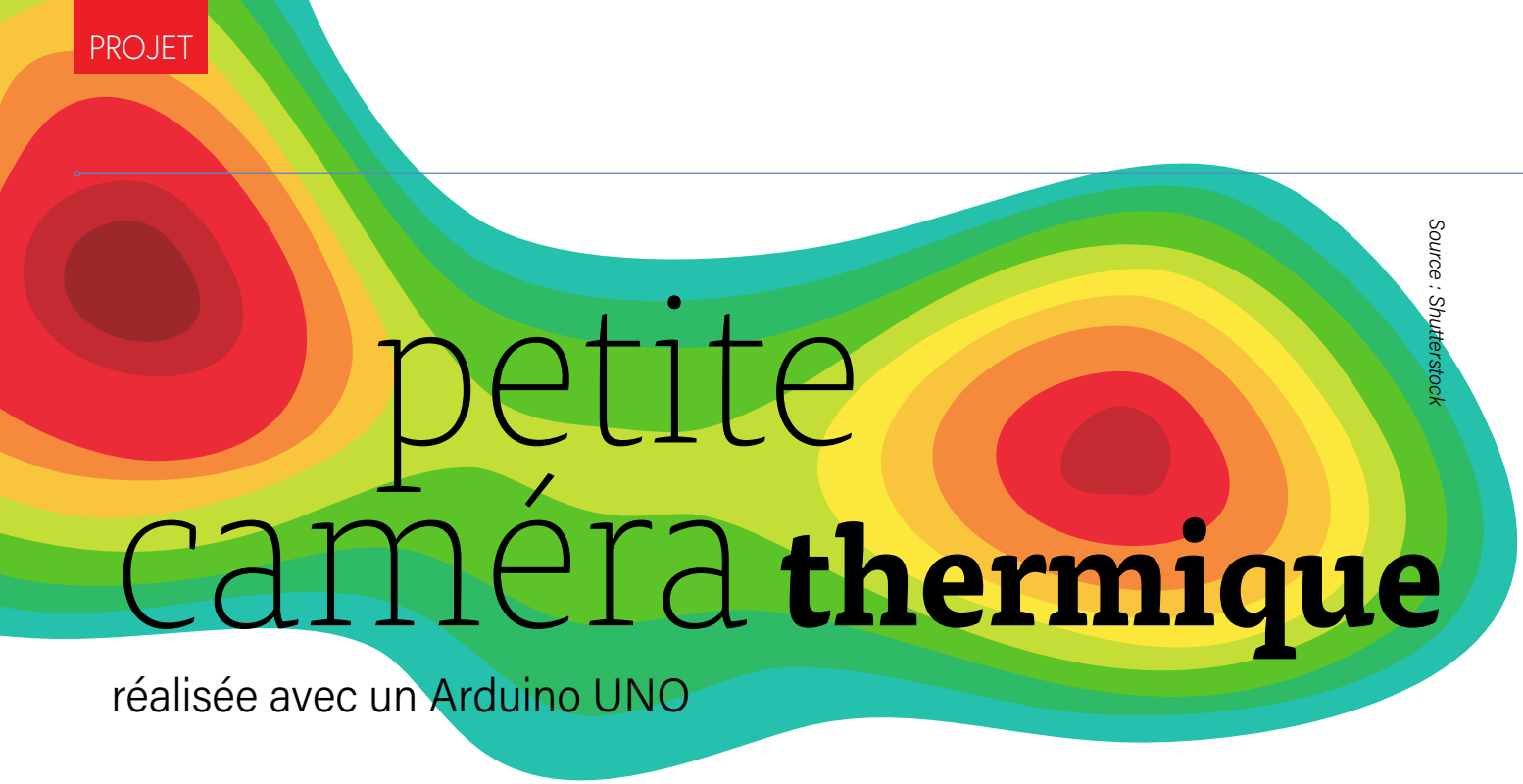


Source : Shutterstock

petite caméra thermique

réalisée avec un Arduino UNO

Roland Stiglmayr (Allemagne)

Quiconque a déjà eu besoin de détecter des points chauds sur un matériel électronique, de mesurer la température d'un composant de puissance ou de détecter des personnes dans une pièce, a probablement souhaité disposer d'une caméra thermique. Dans cet article, nous présentons une solution de bricolage peu coûteuse basée sur un capteur 8×8 pixels de Panasonic.

Un électronicien ambitieux se demandera certainement s'il est possible de construire soi-même une caméra thermique. La réponse est : « Oui, c'est possible, à condition d'utiliser les bons composants ». Le capteur thermique le plus approprié est le AMG88xx « Grid-EYE » de Panasonic. Grâce à son traitement interne des signaux, qui se charge de tout le calcul des températures des 64 points de mesure, il réduit considérablement la charge de l'ordinateur externe, ce qui fait qu'un UNO avec un écran TFT de 1,8 pouce comme celui de la **figure 1** est tout à fait suffisant pour construire une caméra thermique simple.

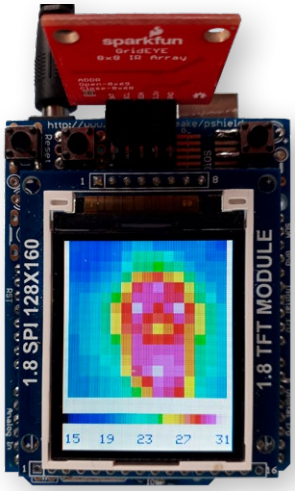


Figure 1. Un cerveau intelligent génère évidemment de la chaleur.

Le capteur à rayons infrarouges

Le capteur AMG88xx [1] est une puce montée sur un support en céramique et fabriquée selon la technologie MEMS (*Micro Electro-Mechanical Systems*) avec 64 points de mesure (*pixels*) sensibles aux infrarouges. Les pixels forment une matrice carrée. Le capteur est accompagné d'un ASIC avec l'électronique de calcul déjà mentionnée et une thermistance pour la saisie de la température de référence. Le support en céramique est recouvert d'un capuchon métallique dans lequel est sertie une lentille, laquelle est en silicium afin de ne pas atténuer le rayonnement dans la gamme de longueurs d'onde de 5 à 12 µm.

La lentille forme l'image de l'objet à mesurer sur le champ du capteur. On peut imaginer que la pointe d'une pyramide carrée est placée sur chaque pixel, qui ne reçoit que le rayonnement contenu à l'intérieur de cette pyramide. L'objet est ainsi représenté par 64 surfaces contiguës formant un carré.

La **figure 2** montre une représentation simplifiée et idéalisée d'un réseau de 4×4 points de mesure. Le champ de vision (*Field of View, FOV*) est la zone visible par le capteur, qui est déterminée exclusivement par les angles d'ouverture vertical et horizontal. Comme le capteur est une matrice carrée, ces angles, notés α, sont égaux. Pour chaque distance d'objet a, il existe une surface visible dans le champ de vision avec des côtés de longueur *S*. La surface est elle-même constituée de 16 surfaces individuelles de côté s_{pix}, chacune associée à un pixel spécifique. La température de l'objet n'est mesurée avec précision que si l'objet couvre entièrement au moins l'une de ces surfaces. Cette condition permet de calculer la plus petite taille d'un objet détectable situé à une distance donnée a. Dans le cas d'une matrice carrée de n×n pixels, on a :

$$s_{pix} = 1/n \; (2a \tan(\alpha/2))$$
$$A_{pix} = s_{pix}^2$$

Dans la figure 2, seule la partie de l'objet ayant les coordonnées (2,2) répond à cette exigence. Les zones du champ de vision ayant les coordonnées (2,1), (1,2), (2,3) ne sont couvertes qu'à moitié. L'écart de température mesuré des pixels associés n'est donc que la moitié de celui de la zone (2,2).

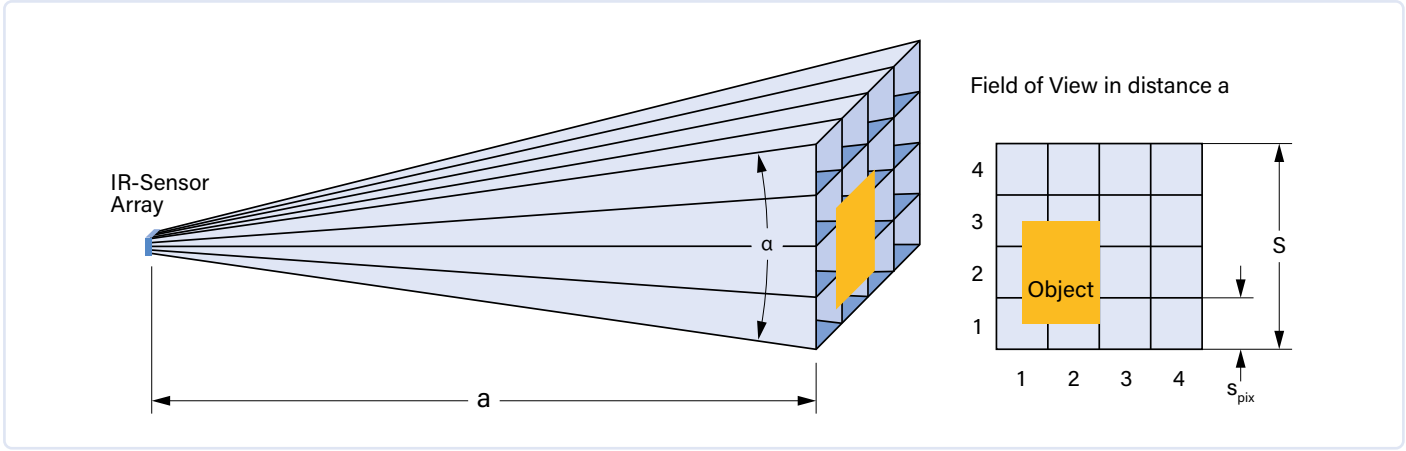


*Figure 2. Zone visible d'un capteur carré 4×4 avec un champ de vision α à une distance **a**.*

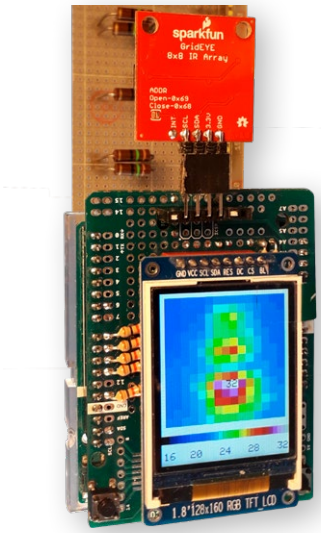


Figure 3. Mesure d'un objet à une distance de 50 mm, α = 60°.

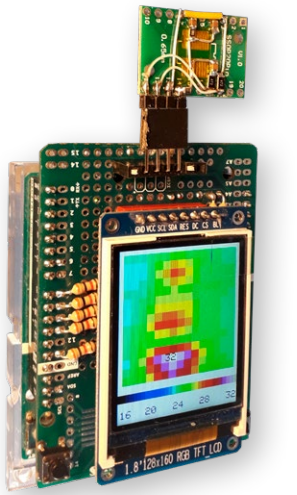


Figure 4. Mesure d'un objet à une distance de 100 mm, α = 30°.

Autrement dit, cela signifie que, pour que la zone projetée sur un pixel soit entièrement couverte, un objet doit être d'autant plus grand qu'il est plus éloigné. Cette condition doit toujours être respectée pour éviter des erreurs de mesure et d'interprétation. Les **figures 3** et **4** illustrent la relation entre le champ de vision et la distance de l'objet à mesurer.

La précision de la mesure dépend également de l'émissivité ε de l'objet. Elle dépend de la composition de la surface et caractérise sa capacité à émettre un rayonnement thermique (c'est le rapport entre sa luminance et celle du corps noir). Le capteur Grid-EYE suppose une émissivité constante de 0,93 dans son calcul interne de la température.

Thermopile

Chaque point de mesure individuel du champ du capteur est ce que l'on appelle une thermopile [2], qui se compose de nombreux thermocouples connectés électriquement en série et thermiquement en parallèle. Le support en céramique fournit la température de référence, la surface irradiée donne la température de la thermopile à mesurer.

Les thermocouples génèrent une tension proportionnelle à l'écart de température. Pour déterminer la température absolue, il faut mesurer très précisément la température de référence et l'ajouter à cet écart. La courbe caractéristique d'un thermocouple n'est pas linéaire, mais dépend de la température de référence et de la tension mesurée. Les valeurs mesurées sont linéarisées à l'aide de courbes stockées dans la mémoire morte du capteur. La dispersion des différentes thermopiles est prise en compte par des données d'étalonnage individuelles stockées en interne. Il est facile d'imaginer à quel point la linéarisation de la courbe caractéristique consommerait de ressources en temps et en calcul si l'ASIC interne ne se chargeait pas de cette tâche. Sur son interface I²C, l'ASIC fournit les valeurs mesurées linéarisées et parfaitement traitées en degrés Celsius. Différents types de capteurs sont disponibles en fonction de l'application. Le **tableau 1** en donne un aperçu.

Tableau 1. Valeurs typiques des capteurs infrarouges de la série AMG88xx.

Module	V_{DD}	Plage de mesure	Angle total de détection	Angle de détection d'un pixel	Taille de l'objet relative à un pixel à une distance de 30 cm
AMG8832	3,3 V	-20°C ... 100°C	60°	7.5°	4 cm
AMG8833	3,3 V	0°C ... 80°C	60°	7,5°	4 cm
AMG8834	3,3 V	-20°C ... 100°C	60°	7,5°	4 cm
AMG8853	5 V	0°C ... 80°C	60°	7,5°	4 cm
AMG8854	5 V	-20°C ... 100°C	60°	7,5°	4 cm
AMG883642	3,3 V	-20°C ... 100°C	32° ... 34°	4°	2 cm
AMG883543	3,3 V	0°C ... 80°C	90°	11,25°	6 cm

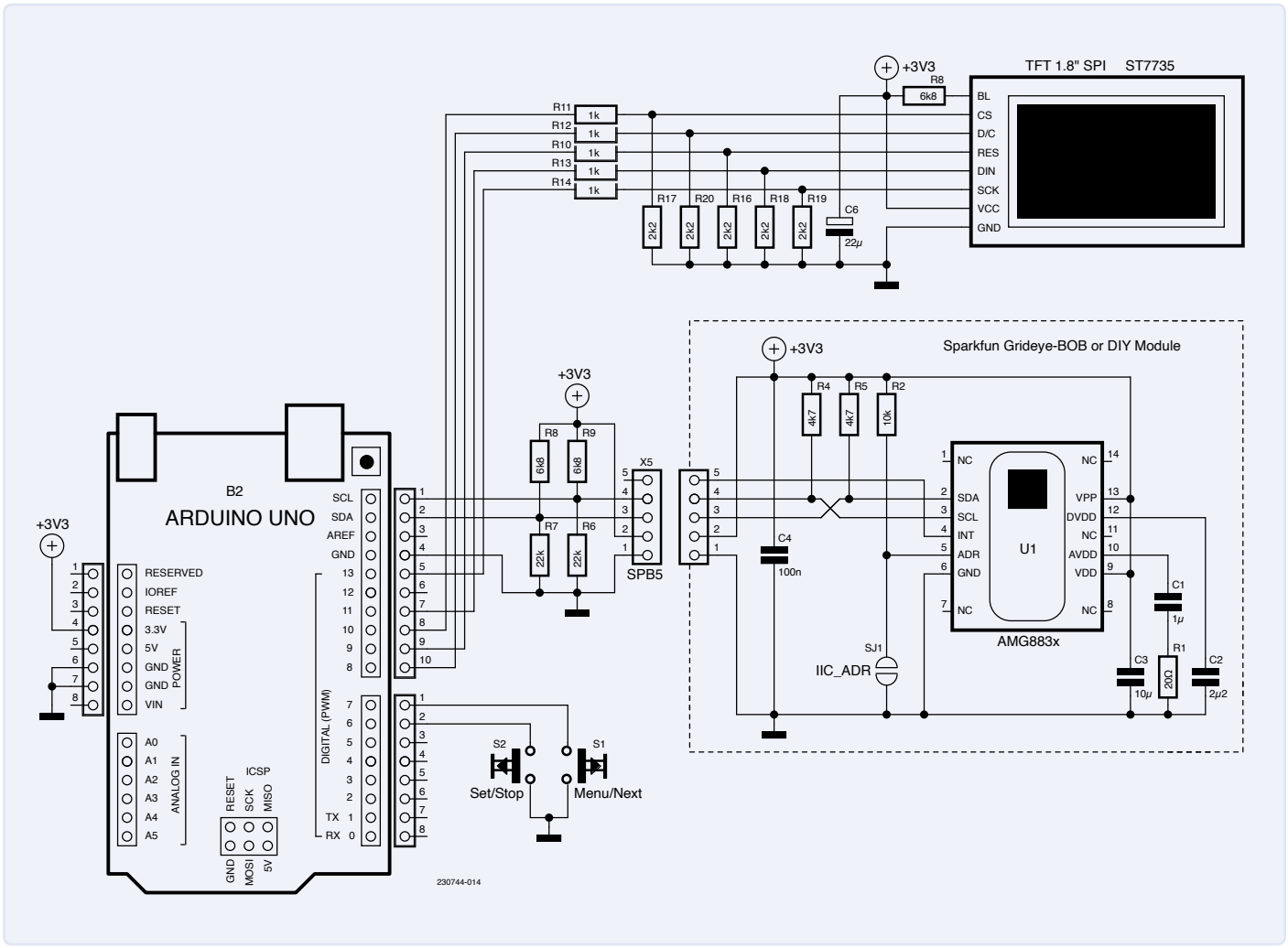


Figure 5. Schéma de la caméra.

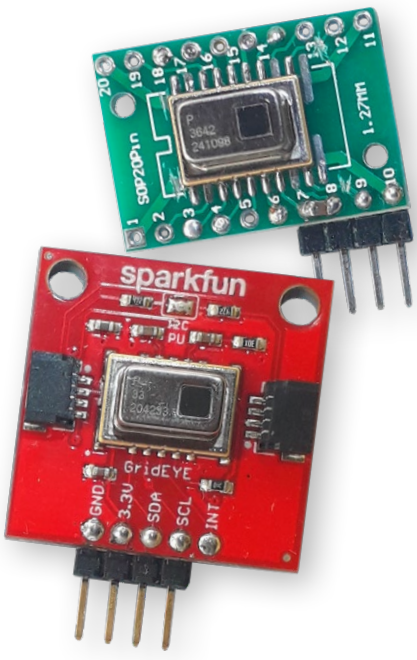


Figure 6. Module de capteur maison à côté d'un BoB de Sparkfun.

La structure de la caméra

Le circuit complet de la caméra thermique se compose d'un Arduino UNO, d'un bouclier avec un petit écran couleur TFT, de deux boutons et du module de la caméra. La **figure 5** montre le schéma complet de la caméra

Module de caméra

Si vous décidez d'utiliser un BoB (*Breakout Board*) [4] (**figure 6**) entièrement assemblé avec un AMG8833, vous avez déjà terminé. Il ne vous reste plus qu'à déterminer si le câblage doit être réalisé avec des Qwiic (la connectique de *SparkFun*) ou des barrettes à broches. Cependant, vous pouvez aussi construire le module vous-même assez facilement, car le capteur peut être soudé sans problème à l'aide d'un fer à souder fin. L'avantage de la version maison est que vous pouvez choisir un capteur approprié. Par exemple, le AMG883642 [3] est très intéressant pour de nombreuses applications en raison de son petit champ de vision et de sa large plage de température.
L'auteur a soudé le capteur sur un module adaptateur SSOP20 vers DIL (1,27 mm par 2,54 mm) en compagnie de quelques autres composants CMS (**figure 7**). L'adresse I²C de l'appareil n'a pas d'importance, car le logiciel détermine l'adresse correcte au démarrage, de sorte que la broche ADDR peut être connectée à la masse.

Bouclier TFT

Le circuit de la figure 5 étant très simple, il peut facilement être construit sur un bouclier Arduino Mega Proto Shield. Il y a suffisamment

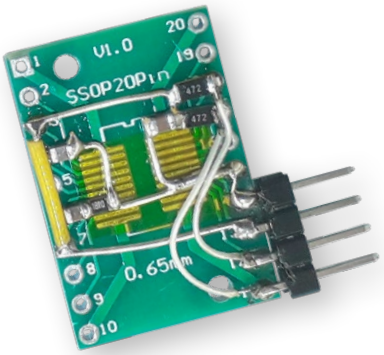


Figure 7. Module de capteur maison, côté arrière.

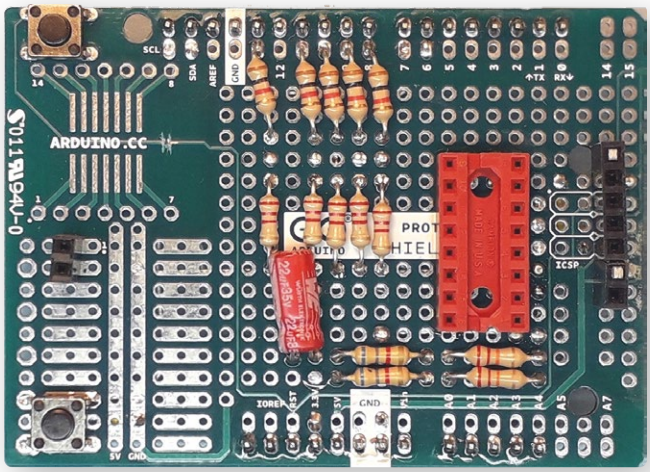


Figure 8. Bouclier maison, côté assemblage.

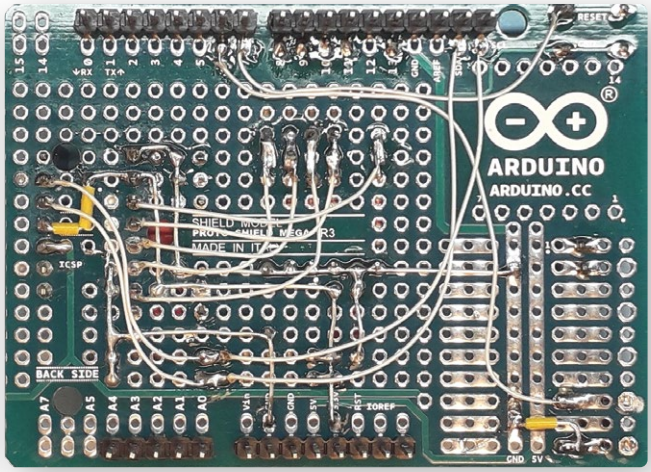


Figure 9. Bouclier maison, côté câblage.

d'espace pour utiliser des composants traversants (**figure 8**, **figure 9**). L'écran TFT, d'une diagonale de 1,8 pouce et de 128×160 pixels, doit être équipé d'un contrôleur ST7735 et doit pouvoir fonctionner sous 3,3 V. Un produit bon marché d'origine chinoise convient parfaitement. Mais attention : ces cartes ont des brochages variés. Ce n'est pas un problème, mais il faut y prêter attention. Une embase DIL est un très bon connecteur pour l'écran.

Alimentation et bus

Le module caméra est alimenté par l'alimentation 3,3 V de l'UNO. C'est pourquoi les résistances de rappel des lignes I² C sont connectées au 3,3 V. En plus des résistances sur le module de la caméra, il y a également sur le bouclier de l'auteur deux rappels vers le 3,3 V et deux rappels vers le 0 V pour compenser le courant interne de 140 µA provenant des ports. Bien que l'UNO fonctionne à 5 V, il lit les signaux de 3,3 V sans aucune erreur.

L'écran TFT est contrôlé via l'interface SPI. Son contrôleur ST7735 travaille exclusivement avec des signaux de 3,3 V. Les signaux du port 5 V de l'UNO sont donc réduits à 3,3 V à l'aide des diviseurs de tension R10 et R14 ainsi que R16 et R20. Comme l'horloge SPI est de 10 MHz, la valeur de ces résistances ne doit pas être trop élevée pour limiter l'effet des parasites. Les valeurs spécifiées représentent un bon compromis entre la consommation de courant et la transmission de signaux sans distorsion. L'emplacement des deux boutons est illustré à la **figure 10**.

Mise en service

Commencez par installer les deux bibliothèques, *Adafruit_ST7735_and_ST7789_Library* et *Adafruit_GFX_Library* en utilisant le gestionnaire de bibliothèque de l'IDE Arduino.

Ensuite, compilez le croquis *Grideye_V2x.ino* à partir de la page du projet Elektor [5] et chargez le programme dans l'UNO.

L'alimentation étant éteinte, enfichez le bouclier, toujours sans le module caméra, et mettez l'UNO sous tension. Le TFT doit s'allumer et afficher le message d'erreur « *no valid device found* ». Vérifiez maintenant la polarité de la tension d'alimentation au niveau du connecteur du module caméra. Les signaux I² C peuvent être mesurés à l'aide d'un oscilloscope. Ensuite, éteignez à nouveau l'appareil, connectez le module de caméra au bouclier et remettez l'ensemble en service.

Si tout va bien, on obtient une image significative. Selon la version du contrôleur ST7735, la position et les couleurs peuvent être incorrectes, ce qui peut être corrigé dans le programme. Le croquis propose quatre routines d'initialisation différentes à cette fin. Quatre valeurs `TFT_TYPE_n` sont définies à cet effet, dont l'une doit être assignée à la constante `TFT_TYPE`. Avec la valeur correcte, l'image est cadrée en haut, à gauche et à droite. Si les couleurs rouge et bleue sont inversées, la valeur de `TFT_CHANGE_COLR` doit être fixée à `true`. Si l'image est à l'envers, il faut attribuer la valeur `02` à `TFT_ORIENTATION`. Enfin, la plage de température du capteur utilisé est sélectionnée en assignant la valeur `true` ou `false` à `AMG88x3`.

```
#define AMG88x3 false
//select sensor type, true if AMG88x3
#define PERMANENT_AUTO false
//if true autoranging always active
#define T_OFFSET 0
//offset for correcting T [°C]

#define TFT_ORIENTATION 00
//portrait format
//#define TFT_ORIENTATION 02
//portrait format overhead

#define TFT_TYPE_1 00
//module type 1 (w/o SD)
#define TFT_TYPE_2 01
//module type 2
#define TFT_TYPE_3 02
//module type 3 (with SD)
#define TFT_TYPE_4 03
//module type 4
const byte TFT_TYPE= TFT_TYPE_1;
//set used module type

#define TFT_CHANGE_COLR true
//change red-blue depending on tft
```

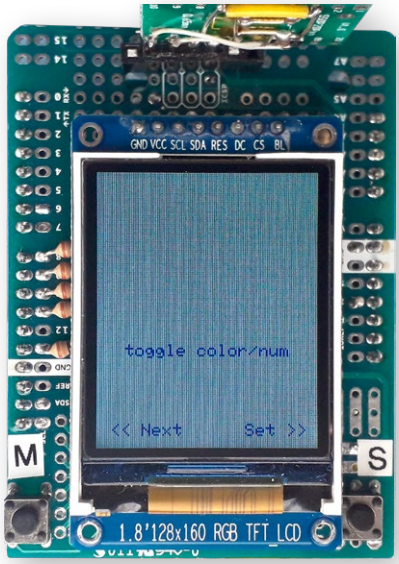


Figure 10. Positionnement des boutons, commutation entre l'affichage graphique et l'affichage numérique.

Listage 1. La fonction read_GridEYE.

```
if (acquire==true)
     (read_GridEYE (&lowestT, &highestT,              //read all pixels and
                    &highestTpix));                   //get lowest and highest T

...

  void read_GridEYE (int *pl, int *ph, int *phnum)
     {
       *ph= -512; *pl= 1024;                          //init values highest, lowest
       int T;                                         //temporary
       byte pixnum;

        //  read data of 64 pixels
        //  ----------------------

       for(byte y = 0; y <8; y++)                     //line address
         {
           for(byte x=0; x<8; x++)                    //column address
             {
               pixnum= (y+56)-(x*8);                  //due to orientation, start with pixel 56
               T= GridEye_getT (pixnum);              //read this pixel
               pixel_Table[y*2][x*2]= T;              //T to each second column, line position

                 if (T>*ph)                           //find highest temperature
                   { *ph=T;
                     *phnum= y*256 + x;               //save position of this pixel
                   }
                 if (T<*pl) *pl=T;                    //find lowest temperature
             }
         }

        //  interpolate between the horizontal pixels of each line
       //  ------------------------------------------------------

       for (byte y=0; y<8; y++)                       //line address counter
         {
           for (byte x=0; x<7; x++)                   //column address counter
             {
               T= (pixel_Table [2*y][2*x] +  pixel_Table [2*y][(2*x)+2])/2;
               pixel_Table [2*y][(2*x)+1]= T;
             }
         }

      //  interpolate between the vertical pixels of each column
      //  ------------------------------------------------------

       for (byte x=0; x<15; x++)                      //column address counter
         {
           for (byte y=0; y<7; y++)                   //line address counter
             {
               T= (pixel_Table [2*y][x] + pixel_Table [(2*y)+2][x])/2;
               pixel_Table [(2*y)+1][x]= T;
             }
         }
     }
```

Logiciel

Le croquis a été créé avec l'EDI Arduino. Aucune bibliothèque externe n'est utilisée pour le capteur Grid-EYE ; toutes les fonctions nécessaires sont présentes dans le croquis. Il est donc facile d'apporter des modifications ou des extensions en fonction de vos propres idées.

Le programme principal appelle la routine de mesure `read_GridEYE` (**listage 1**) de manière cyclique (c'est-à-dire que le processus est déterministe). La routine de mesure lit les 64 températures via la fonction `GridEye_getT` (les températures la plus basse et la plus élevée sont également relevées lors de la lecture et enregistrées avec la position de la température la plus élevée).

Les températures ont une résolution de 0,25 K. Pour éviter les nombres fractionnaires, elles sont multipliées par un facteur 4 par le capteur. Le capteur fournit ensuite les valeurs sur 12 bits, le 12e bit étant le signe. La routine de lecture les convertit en un type `int`, c'est-à-dire 16 bits en complément à deux.

Les valeurs du capteur sont écrites dans le tableau bidimensionnel `pixel_Table`. Ce tableau est de taille 15×15 `int` pour contenir aussi des valeurs additionnelles interpolées. Après avoir lu les valeurs du capteur, la routine calcule des valeurs intermédiaires entre deux valeurs de ligne voisines par interpolation, puis répète le calcul colonne par colonne, en incluant les valeurs déjà interpolées. Les 15×15 = 225 valeurs obtenues sont maintenant disponibles dans le tableau `pixel_Table`.

La fonction `showT_as_Color` affiche les 225 valeurs en fausses couleurs. Les couleurs sont attribuées aux valeurs via la table de correspondance de couleur `color_Scale`. Pour qu'elles soient bien distinctes, la résolution est limitée à 32 couleurs.

Les valeurs mesurées doivent donc être redimensionnées de manière à être identifiables à un ensemble de 32 valeurs. C'est ce que fait la fonction `do_Auto_Ranging`. En fonction de la température la plus basse mesurée, la fonction sélectionne la valeur la plus basse de la plage de mesure parmi six valeurs prédéfinies et la transmet à la variable `Tcoloffset`. La différence entre la température mesurée la plus élevée et `Tcoloffset` est utilisée pour sélectionner la résolution `Tcolrange` de la plage de mesure parmi cinq valeurs prédéfinies. Par soustraction de `Tcoloffset` et multiplication par `Tcolrange`, chaque valeur mesurée est transformée en un nombre représentant l'une des 32 couleurs.

La représentation numérique des valeurs mesurées est gérée par `showT_as_Numeric` et leur sortie série est gérée par la fonction `Serial_send_T`. La routine `Set_menu` permet de sélectionner divers réglages et de les appliquer lorsqu'elle est activée. Ces réglages sont enregistrés dans l'EEPROM et restaurés à chaque redémarrage.

Utilisation

Le bouton *Menu* permet d'accéder aux réglages. En appuyant plusieurs fois sur ce bouton, vous pouvez faire défiler les éléments sélectionnables. Les réglages actuellement actifs sont affichés en vert. Appuyez sur le bouton *Set* pour appliquer l'entrée affichée.

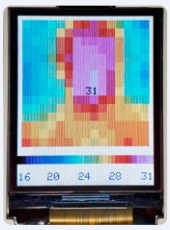
Le premier paramètre est le *calibre automatique*, qui permet généralement d'obtenir un affichage optimal. Avec l'*option Couleur/Num* (voir figure 10), les résultats peuvent être affichés numériquement ou en fausses couleurs.

Si l'on appuie sur le bouton *Set* pendant le processus de mesure normal, la mesure est arrêtée et figée. L'affichage peut alors être modifié dans le menu sans changer les valeurs mesurées. Appuyez à nouveau sur ce bouton pour poursuivre la mesure.

Dans l'affichage en fausses couleurs, la température du point le plus chaud est affichée sous forme de valeur numérique dans le graphique. Cette fonction peut être supprimée lors du redémarrage en maintenant le bouton *Menu* enfoncé pendant la réinitialisation. Les résultats sont transmis en continu via l'interface série avec une résolution de 0,25 K.

En conclusion, on ne peut que s'émerveiller de ce qu'un si petit processeur ATmega est capable de faire !

Vf : Helmut Müller — 230744-04



À propos de l'auteur

Roland Stiglmayr a étudié les technologies de l'information dans les années 1970 et possède plus de 40 ans d'expérience en recherche et développement. Ses travaux ont porté sur le développement d'ordinateurs centraux, de systèmes de transmission de données par fibre optique, de RRH (Remote Radio Head) pour les communications mobiles et de systèmes de transmission d'énergie sans fil. Aujourd'hui, il travaille en tant que conseiller. Il est particulièrement attaché au transfert de connaissances.

Questions ou commentaires ?

Envoyez un courriel à l'auteur (1134-715@online.de) ou contactez Elektor (redaction@elektor.fr).

Produits

- **SparkFun Grid-EYE Infrared Array Breakout – AMG8833**
  www.elektor.fr/19605

LIENS

[1] Documentation du Grid-EYE : https://industrial.panasonic.com/ww/products/pt/grid-eye
[2] Thermopile (Wikipedia) : https://fr.wikipedia.org/wiki/Thermopile
[3] AMG883462 data sheet: https://industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ast-ind-139049.pdf
[4] Grid-EYE-BoB : https://sparkfun.com/products/14607
[5] Page du projet : https://elektormagazine.fr/230744-04