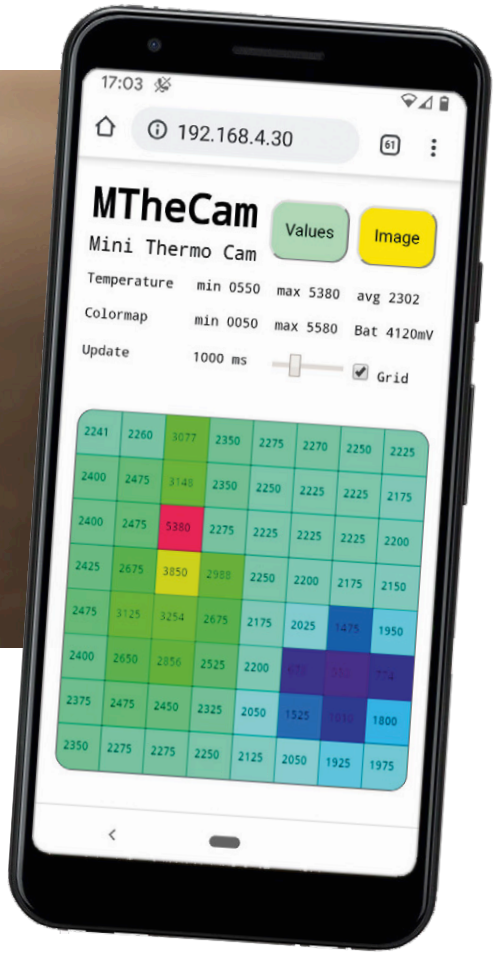
MTheCam
de mini-thermocamera

eenvoudige warmtebeeldcamera voor het lokaliseren van hot- en coldspots

Olaf Mertens (Duitsland)

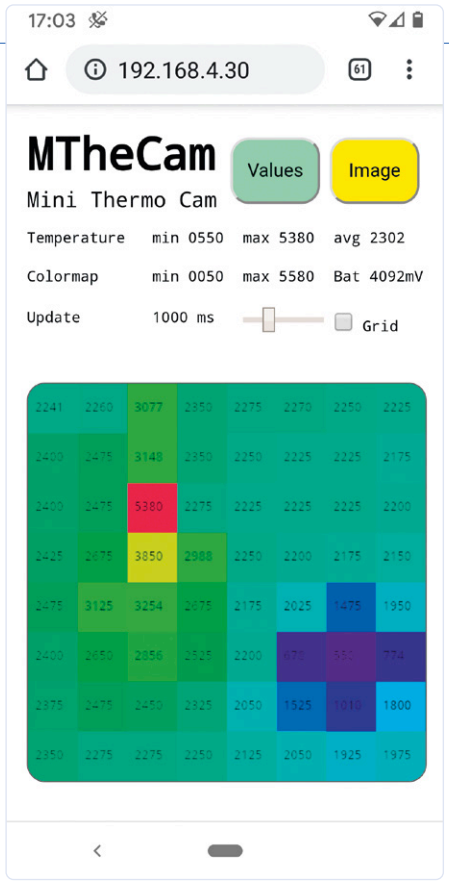
Is de kaars uitgeblazen? En staat de kookplaat uit? Wie heeft bij het verlaten van zijn huis niet ooit het onaangename gevoel gehad dat een of andere warmtebron grote schade kan aanrichten als hij onbeheerd wordt achtergelaten? Voortaan kunt u dat controleren via de MTheCam en uw smartphone. Maar het hier gepresenteerde project, dat met een 8x8-pixel sensor van Panasonic werkt, is natuurlijk geschikt voor veel andere toepassingen.



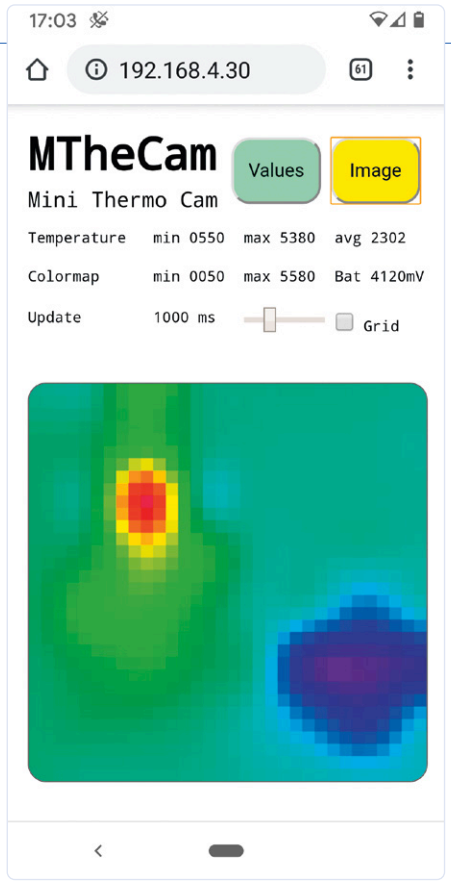
Bij hotspots denken we misschien als eerste aan het openbaar internet, maar eigenlijk wordt die term gebruikt om punten aan te duiden die beduidend warmer (of kouder in het geval van coldspots) zijn dan hun omgeving. Ze wijzen op vlammen, oververhitte plekken en kortsluitingen, maar ook op koudebruggen en lekkages. Als ze niet duidelijk gloeien of opvlammen zijn ze voor ons onzichtbaar. Om ze te detecteren, is een geschikt visueel hulpmiddel nodig: MTheCam.

Het lokaliseren van hotspots hoeft zich niet te beperken tot bijvoorbeeld een vergeten kookplaat. In elektronische schakelingen bijvoorbeeld geven overbelaste componenten soms in een vroeg stadium door warmteontwikkeling aan dat ze dreigen te bezwijken. In machines kunnen versleten en slecht gesmeerde lagers oververhit raken; tijdig onderhoud bespaart kostbare reparaties. Zelfs mensen kunnen worden geïdentificeerd als 'hotspots', zodat men ze kan tellen en hun bewegingen kan volgen.

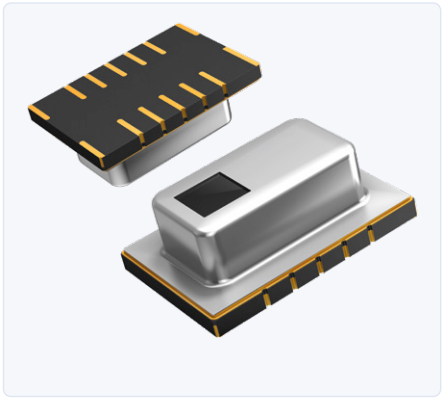
Naast meetapparaten die de temperatuur meten door middel van direct contact, zijn er ook contactloze sensoren die de IR-straling van objec-



*Figuur 1. Screenshot van de temperatuurwaarden in 8*8 pixels.*



*Figuur 2. Screenshot van de naar 32*32 pixels geïnterpoleerde waarden.*



Figuur 3. De AMG88xx-sensor van Panasonic.

ten op enige afstand opvangen en daaruit de gemiddelde temperatuur van het meetgebied berekenen. Dit gebeurt door gebruik te maken van het pyro-elektrisch effect, waarbij de elektrische potentiaal van een gepolariseerd kristal dat in elektroden is ingebed, verandert bij blootstelling aan thermische straling. Dit effect kan natuurlijk elektronisch worden gebruikt [1].

Het project

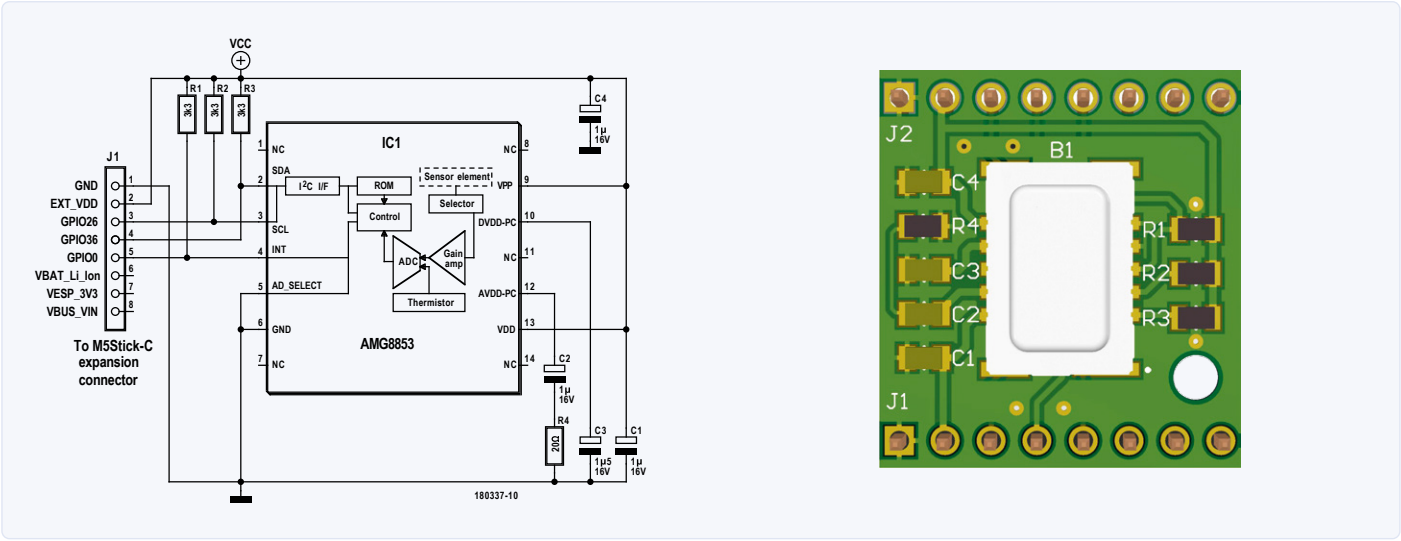
MTheCam meet tegelijk tot vijf maal per seconde 64 punten in acht rijen en acht kolommen, gerangschikt als in een optische camera met een beeldhoek van 60°. Elk punt geeft de temperatuur aan tussen 0 graden en 80 °C (of –20...100 °C). Individuele kleurwaarden van een kleurverloop worden toegewezen aan de individuele gemeten waarden, wat resulteert in een beeld met een vrij geringe resolutie, dat vervolgens via WiFi kan worden weergegeven als een webpagina, bijvoorbeeld op een smartphone. Esthetisch gezien is deze kleurrijke verzameling van reuzenpixels zeker niet wat onze verwende HDTV-ogen gewend zijn, maar het laat wel warme of koude plekken zien met kleurvlakken die sterk verschillen van de omgeving. Naast het kleurenbeeld worden de gemeten waarden per pixel ook weergegeven als temperaturen in graden celsius, wat een nauwkeurigere analyse mogelijk maakt (**figuur 1**).

De 64 meetwaarden kunnen ook als JSON-datarecord worden opgeroepen, zodat koppeling met andere toepassingen mogelijk is.

Om de illusie van een hogere resolutie dan de 8x8 puntmatrix te creëren, wordt een 'vals' beeld met 32x32 kleurstippen gecreëerd met behulp van een wiskundige methode die bicubische interpolatie heet [2]. Het spectrum van de kleurschaal wordt ook aangepast aan het bereik van de gemeten minimum- en maximumtemperaturen, inclusief een beetje 'speling'; het effect is als van een loep voor de gemeten temperaturen (**figuur 2**).

De thermosensor AMG88xx

Panasonic heeft een krachtige MEMS (**figuur 3**) ontwikkeld voor het opnemen van warmtebeelden, die optiek, thermo-elektrische conversie, analoge meting en andere gegevensverwerkingsfuncties combineert in een relatief kleine behuizing [3]. Er zijn twee versies met verschillende meetbereiken. De AMG8853 werkt in het bereik 0...80 °C en de AMG8854 bij -20...100 °C. De maximale fout is ±2,5 K (±3,0) en de gegevens worden uitgevoerd via een I²C-interface. De absolute nauwkeurigheid is weliswaar niet geweldig, maar de sensor is toch goed geschikt voor de kwalitatieve beoordeling van relatieve waarden. Met een paar regels software kan ook een kalibratie worden uitgevoerd, wat de nauwkeurigheid vergroot en de ruis vermindert. De Panasonic-sensor is ondergebracht in een compacte SMD-behuizing die ontworpen is voor reflow-solderen. Voor onze toepassing plaatsen we hem op een klein zelfgemaakt breakout-board, dat naast de sensor zelf een paar vereiste passieve componenten (in het gemakkelijk te hanteren 0805-formaat) voor het ontkoppelen van de voedingsspanning en als pullups voor de I²C-bus bevat. Naast de voeding (+5 V en GND) zijn er drie belangrijke verbindingen (INT en de I²C-bus) beschikbaar op de pads aan de rand van de printplaat, waar een (rechte of haakse)



Figuur 4. Interne en perifere schakeling van de AMG88xx-sensor op het breakout-board.

pinheader kan worden geplaatst. **Figuur 4** toont het schema van dit board en geeft ook wat informatie over de ingewanden van de sensor. De gratis projectdownload [9] bevat layouts van beide printlayers en de componentenopstelling. Als u het uzelf gemakkelijk wilt maken, kunt u de volledig gemonteerde module ook bij de auteur bestellen.

De M5StickC: een echte duizendpoot

Elke smartphone is WLAN-compatibel en heeft een aanraakscherm voor in- en uitvoer, waardoor hij een ideale gebruikersinterface is voor onze MTheCam. Zo kunnen we via WiFi met de sensor communiceren. Maar daarvoor heeft de sensor een slimme transceiver nodig: een SoC (System-on-Chip) met controller en WLAN-module.

We hebben gekozen voor de ESP32 van Espressif [4]. Deze module heeft alles wat een ontwikkelaar zich maar kan wensen, is goedkoop en energiezuinig en is bovendien eenvoudig te programmeren met de Arduino IDE. Naast de sensor en ESP-module is alleen een 5V-voeding nodig voor de elektronica, die via USB of, in mobiele toepassingen, uit een oplaadbare batterij kan worden gevoed. In de Elektor Shop vindt u een product in een fel oranje outfit, dat een paar interessante dingen biedt bovenop de ESP32: de M5StickC [5] past perfect!

Dit compacte apparaatje met een afmeting van ongeveer 50x26x14 mm heeft behoorlijk wat in zijn mars. Naast de ESP-32Pico met 4 MB flashgeheugen en 520 kB SRAM heeft hij een 0,96"-display met 80x160 pixels, een 6-assige bewegingssensor, een real-time klok, een power management unit, een rode LED, een IR-LED voor de afstandsbediening, een MEMS-microfoon, een 80mAh-accu, USB-C, Grove-connectoren (PWR, GND en twee I²C-poorten) en een 8-polige vrouwelijke connector met drie poorten plus voeding. Daar zal zelfs de meest geharde elektronicus van onder de indruk zijn!

Door het kleine formaat, het display en de mobiele accuvoeding kan de M5StickC ook als een chique smartwatch worden gebruikt: een houder en een fel oranje polsbandje zijn al inbegrepen. Hiermee trekt u gegarandeerd de aandacht, zoals moge blijken uit **figuur 5**! Wie wil er dan nog zo'n saaie armband van een bepaalde Californische appelproducent, waar je niet eens iets in kunt stoppen?

In dit project worden in eerste instantie alleen de ESP32, het display en de accu gebruikt. Maar alle andere functies worden niet verspild, daar zullen we in toekomstige projecten vast en zeker nog nuttige toepassingen voor vinden.

Het display van de M5StickC is vrij klein, maar het toont een scherp kleurenbeeld. Een mooie uitdaging voor ontwerpers van gebruikersinterfaces om de informatie op een zo klein mogelijk oppervlak onder te brengen. Het warmtebeeld past goed en er zijn ook nog een paar regels voor de meetwaarden beschikbaar. De displaybibliotheek biedt veel mogelijkheden voor grafische effecten, animaties en kleurweergave; dat geeft makers volop de gelegenheid om zich creatief uit te leven.

De Li-Ion accu van 80 mAh kan de M5StickC bijna een uur lang volledig van stoom voorzien. Hij wordt opgeladen via een USB-C aansluiting met 5 V/500 mA, die hij via de meegeleverde kabel krijgt uit een normale USB-A-poort. Een kleine knop aan de zijkant schakelt de M5StickC in en door die zes seconden lang gedrukt te houden schakelt hij weer uit.

Samenwerking

Het breakout-board van de sensor wordt in de busstrip voor externe uitbreidingen geprikt. Met een haakse pinheader komt het als een rugzak tegen de achterkant van de behuizing te liggen, met een rechte pinheader meet hij in de richting van de lange kant van de behuizing, zodat we de sensor kunnen richten op het te onderzoeken gebied (**figuur 6**). Met behulp van een 3D-printer kan voor beide varianten gemakkelijk een mooie behuizing worden getoverd.

De uitbreidingsconnector van de M5StickC heeft acht pinnen, waarvan naast 5V-Out en GND de twee I²C-lijnen voor de data- en kloksignalen (SDA op GPIO26 en SCL op GPIO0) en het interruptsignaal INT (op GPIO36) toegankelijk zijn. De aansluitingen BAT, 3V3 en 5V-In worden niet gebruikt. Overigens kan GPIO36 alleen als ingang worden gebruikt; deze kennis bespaart veel onnodig debuggen.

De mogelijkheid voor interrupts wordt in de software vooralsnog niet benut. Daemee zou men de sensorchip bijvoorbeeld kunnen programmeren om een INT-signaal te geven wanneer een drempelwaarde wordt overschreden of niet wordt gehaald, en dat voor elk afzonderlijk pixel! Dat betekent dat de chip een bepaalde positie binnen zijn gezichtsveld zelf kan bewaken en pas bij overschrijding van een drempelwaarde de ESP32 kan wekken. Dat spaart energie! Uiteraard

kan het monitoringproces later door de app voor de smartphone zelf worden afgehandeld door het gehele beeld te evalueren.

Een kwestie van software

Naast de eigen ontwikkelomgeving van Espressif kan ook het Arduino-ecosysteem worden gebruikt voor het maken van de firmware, wat erg handig is vanwege de zeer effectieve wereldwijde ondersteuning door de community. De originele Arduino IDE is goed geschikt voor het begin, maar wordt bij veeleisende taken al snel overbelast. De auteur beveelt daarom de gratis Visual Studio code-editor van Microsoft aan, die ook Arduino-ondersteuning bevat als extensie [6]. *M5Stick-C (esp32) wordt* hier geïntegreerd via de board manager.

De firmware voor MTheCam is met deze middelen door de auteur geprogrammeerd in C++. Naast het bronbestand *MTheCam_LT.ino* worden enkele .h- en .cpp-bestanden (*Mxxx.cpp/h*) meegenomen, plus de sterk aanbevolen *ArduinoJson*-bibliotheek (minimaal V6, V5 is niet geschikt!) voor JSON-afhandeling [7]. De *M5StickC*-bibliotheek [8] maakt het gebruik van de speciale mogelijkheden van de hardware gemakkelijk. Deze moeten met de bibliotheekmanager worden toegevoegd (*F1 - Arduino: Library Manager*).

In tegenstelling tot de eersteklas documentatie van de *ArduinoJson*-Lib laat de beschrijving van de *M5StickC*-bibliotheek veel te wensen over. Om die goed te kunnen gebruiken is het noodzakelijk om de broncode zorgvuldig en aandachtig te lezen. Daar valt zoveel over te vertellen, dat het wel een apart artikel waard is!

Laten we nu eens kijken naar wat softwarefragmenten voor de afzonderlijke functies, in hun logische volgorde.

Sensor uitlezen

De actuele meetwaarden worden door de sensor tien keer per seconde aangeleverd in registers van één byte, die door de software continu worden uitgelezen via de *Wire* I²C-interface. Per pixel moeten twee bytes worden opgevraagd (vanwege de 12-bits resolutie). *Wire* doet dat in `MTC_readReg()` als volgt:

```
#define BUF_LENRX 128
int reg = 0x80;
byte _rxBuf[BUF_LENRX];
. . .
Wire.beginTransmission(devAddr); // Chip-Address @
   datasheet
Wire.write(reg); // 0x80 -> read 128bytes of 64
   pixels @12bit
Wire.endTransmission();
if (Wire.readTransmission(devAddr, rxBuf, BUF_
   LENRX) == I2C_ERROR_OK) { success = true; }
Wire.endTransmission();
```

`Wire.readTransmission()` kan foutcodes retourneren die, als de uitlezing niet werkt, nuttig kunnen zijn voor het debuggen.

Het codefragment in **listing 1** is verantwoordelijk voor het uitlezen van de sensor en het berekenen van de temperatuurwaarden. Uit twee `rxBuf[]` array-items kan de temperatuurwaarde worden berekend, hier als een integer met een resolutie van 1/100 graad celsius: 21,35 °C geeft dus de waarde 2135. De sensor wordt zo geïnitialiseerd dat hij intern het voortschrijdend gemiddelde van twee beelden vormt en zo de ruis wat vermindert. Alle registers worden in één keer uitgelezen, zodat alle waarden gegarandeerd tot één frame behoren. De verdere berekeningen van de pixel- en chiptemperatuur (die laatste wordt hier niet gebruikt) verschillen een factor (zie datasheet [3]). Voor een grotere nauwkeurigheid van de berekening wordt eerst met 10000 vermenigvuldigd en later gedeeld door 100, zodat we een resolutie van 1/100 graad krijgen.

Figuur 5. De M5StickC als een fel oranje smartwatch (bron: m5stack.com).

Figuur 6. Haaks of recht: twee manieren om de sensor aan de M5StickC te bevestigen.

Berekening van de kleurwaarden

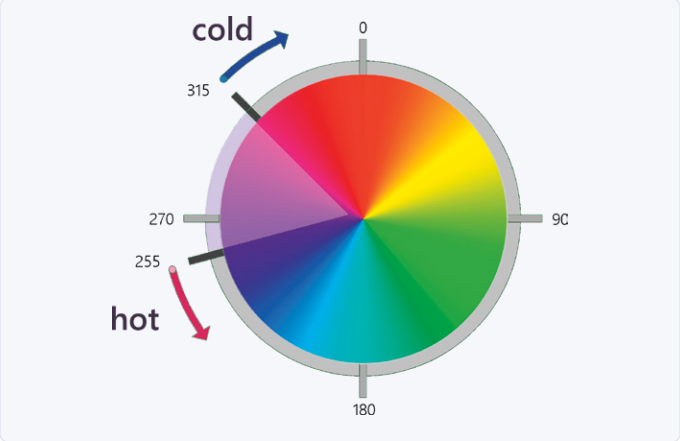
Voor de kleurweergave wordt gebruik gemaakt van het HSL-kleurenschema [10] (**figuur 7**). In de M5Stick-C-verzameling is daarvoor een TFT-bibliotheek aanwezig. Van de parameters kleurhoek (hue, H), verzadiging (saturation, S) en helderheid (lightness, L) wordt alleen de H-parameter gebruikt, S en L blijven constant. De

Listing 1. Sensor uitlezen en temperatuur berekenen.

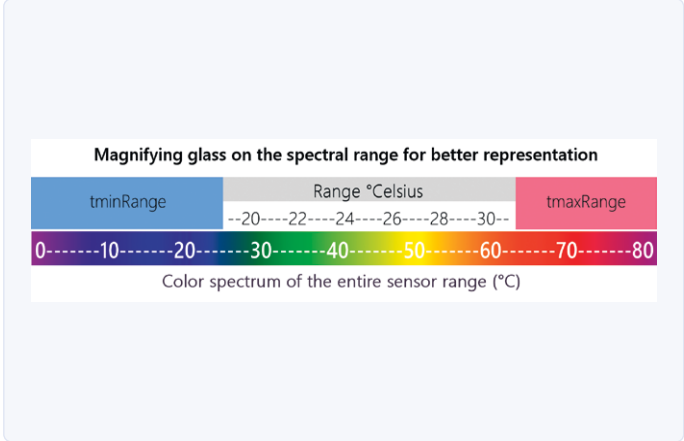
```
. . .
 // read one frame with 64 temp-values à 12bit -> join 2 bytes to 1 int
 // result is int[] with calculated values -> _frame[]
 int  _frame[FRAMESIZE];          // filled by MTC_getFrame()
. . .
 int* MTC_getFrame()
  {
    byte _rxBuf[BUF_LENRX];
    byte dL[FRAME_SIZE];  // low byte
    byte dH[FRAME_SIZE];  // high byte
    int reg = 0x80;       // start with this reg for next 128 bytes to read
    . . .
    // sensor-read 128 bytes - low + high
    boolean ok = MTC_readReg(MTCADDRESS, reg, _rxBuf, 128);  // readings now in _rxBuf
    if (ok)
    {
        int k = 0;
        for (int i = 0; i < FRAME_SIZE; i++)
        {
            int index   = i;
            int d1 = _rxBuf[k++];                // low byte here
            int d2 = _rxBuf[k++];                // high byte next: includes sign!
            // sensor on PCB is upside down, start from the end
            index  = (FRAME_SIZE - i - 1);
            _frame[index] = MTC_calcTemp(d1, d2); // 2 bytes calc'd to one int
        }
     }
    return _frame;
 }
. . .
 // make one int from low/hight bytes d1/d2
 int MTC_calcTemp(uint8_t d1, uint8_t d2)        // return temp in deg Celsius * 100
  {
    int factor = (int) (0.25 * 10000);           // pixel-temp, for device-temp use 0.0625
    if (d2 & 0b00001000) d2 |= 0b11111000;       // handle sign-bit #4 in high-byte
    int valRaw = (d2 << 8) | d1;                 // merge to one int …
    int valCalc = (int)(valRaw * factor / 100);  // … with this factor
    return valCalc;
 }
. . .
```

Listing 2. Berekening van de kleurwaarden.

```
#define MC_COLORWHEEL_LOW  315     // degrees on colorwheel
#define MC_COLORWHEEL_HIGH  15
#define MC_DIFFCW           60     // diff in degrees high to low
. . .
int MC_getColorHSL(int value, int vmin, int vmax, int vavg)
{
    // do not use 255 - 315 deg = 60 deg -> it's not clearly 'felt' as cold or hot
    // vmin/vmax for minimum spread of spectrum -> less color flicker if no spots seen
    if((vavg - vmin) < MC_TMIN_SPREAD2AVG)  { vmin = vavg-MC_TMIN_SPREAD2AVG; }
    if((vmax - vavg) < MC_TMIN_SPREAD2AVG)  { vmax = vavg+MC_TMIN_SPREAD2AVG; }
    // map range vmin -> 315, vmax -> 15 degree 'left turn' - exclude 315...15 = 60 deg
    int colInd = map(value, vmin, vmax, MC_COLORWHEEL_LOW, MC_COLORWHEEL_HIGH); // 315/15
    colInd = colInd - MC_DIFFCW;   // turn - 60: icecold->DEEPBLUE=255, hot->DARKRED=315
    if (colInd < 0) colInd += 360; // no neg. values!
    return colInd;
}
```



Figuur 7. HSL-model: kleurenwiel en het gebruikte gebied.



Figuur 8. Het meetbereik wordt over het gewenste deel van het spectrum gelegd.

HSL-waarden worden omgerekend naar RGB-waarden om ze later op het display te kunnen weergeven.
Omdat we het bereik 0…80 °C verdelen in 8000 meetwaarden (van 1/100 kelvin) voor de numerieke weergave, moeten we die eerst schalen naar de mogelijke kleurwaarden 0…359. We associëren de kleur blauw met koud en rood met warm, zodat het cirkelsegment vanaf 255 graden (tMin = blauwgroen) linksom draaiend tot 315 graden (tMax = donkerrood) optimaal lijkt te zijn voor de weergave. Omdat we violet meestal niet associëren met een temperatuursensatie, worden de violette kleurwaarden niet gebruikt. Zo worden 300 van de mogelijke 360 kleurwaarden gebruikt en 60 niet.
De handige functie `map()` in **listing 2** schaalt de temperatuurwaarden naar bereik van 300 graden tegen de klok in, van 315° tot 15°. Daarna draait hij alles weer 60° linksom (kleurenhoek –60) om uiteindelijk uit te komen bij de gewenste waarden 255…315. Om de tamelijk sterke kleurruis in gelijkmatig warme gebieden te verminderen, wordt een minimale afstand van 500 eenheden vastgelegd tussen tMin en tMax en de gemiddelde temperatuur, dat wil zeggen dat het weergavebereik (het 'vergrootglas') een minimale breedte heeft van 1000 eenheden, wat overeenkomt met 10 graden. Zo wordt niet elk ruispixel onnodig weergegeven als een volledig andere kleur.
De kleurenbalk met het bruikbare spectrale bereik ligt dus vast: de minimale temperatuur komt overeen met 255 graden op het kleurenwiel en de maximale met 315 graden, waarbij we tegen de klok in gaan!

Weergave op het display

De waarden `tminRange` en `tmaxRange` worden gebruikt als parameters voor de kleurbepalingsfunctie (**figuur 8**) om een soort loepfunctie te creëren die het werkelijke weergavebereik comprimeert naar het beschikbare spectrale bereik. Dat is maar één, uiterst eenvoudige, manier om een aantrekkelijk kleurendisplay te krijgen. Het zou bijvoorbeeld nog slimmer zijn om de grenzen van het bereik adaptief te bepalen over meerdere metingen, of om de dynamiek voor de hotspot en de achtergrond apart te behandelen om vlekken duidelijker te benadrukken. Dit kan een interessant terrein zijn voor deskundigen op het gebied van machinaal leren voor patroonherkenning, ruisonderdrukking en gegevensanalyse.
Het TFT-display van de M5StickC geeft in de bovenste helft het originele warmtebeeld met het gebruikte kleurenspectrum weer zoals in

Figuur 9. Het temperatuurveld op het eerste prototype.

figuur 9, en daaronder zien we de gemeten waarden voor de minimale en maximale temperatuur van alle pixels van het beeld (tMin en tMax) en de gemiddelde waarde die hieruit wordt berekend (tAvg). Deze waarden worden gegeven in °C * 100, dus zonder decimale punt.

WiFi en webserver

Tot nu toe werkt alles lokaal op de M5StickC en zonder WiFi. Maar als de meetwaarden op afstand moeten worden weergegeven of worden verwerkt op een mobiel apparaat, moet de M5StickC verbinding kunnen maken met een WiFi-netwerk en een webpagina met de meetwaarden kunnen aanbieden als webserver.
De bibliotheken `WiFi.h` en `WiFiClient.h` in de Arduino-software creëren zo'n webserver, die als parameters alleen de toegangsgegevens van uw netwerk (`ssid` en `password`) nodig heeft. Na een reset of PowerOn toont het display dat wordt geprobeerd om verbinding te maken met het netwerk. Als dit lukt, wordt het ontvangen IP-adres weergegeven. Als dat niet het geval is, moet u het volgende contro-

Listing 3. WLAN en webserver.

```
. . .
    const char *ssid     = "YourSSIDHere";      // change…
    const char *password = "YourPasswordHere";  // … to your WLAN
    WebServer webServer(80);                    // listening on port 80 = standard

    WiFi.mode(WIFI_STA);                        // switch to ESP station mode
    WiFi.begin(ssid, password);                 // look for known WLAN & try to connect
    while (WiFi.status() != WL_CONNECTED)       // wait for connection
    {
       delay(500);                              // 500ms delay
       Serial.print(".");                       // still alive
    }
    Serial.println("\n[Setup] WIFI connected!");
    // what to do if browser request is coming in
    webServer.on("/", MW_index);                // call MW_index() for / or /index.html
    webServer.on("/index.html", MW_index);
    webServer.on("/frame", MW_frameJS);         // /frame - output as JSON-Data
    webServer.onNotFound(MW_notFound);
    webServer.begin();                          // webserver starts here
. . .
// Webserver callback
void MW_index()     // send HTML-Doc from literally defined PROGMEM var
 {
    webServer.send_P(200, "text/html", _uidoc); // send_P: use PROGMEM-Var
 }
. . .
// main loop
void loop()
 {
                    // give webserver a chance to handle requests. No delay() in LOOP!
      webServer.handleClient();
     . . .
 }
. . .
```

leren: zijn de toegangsgegevens correct? Is er een toegangspunt in de buurt? Overigens is het bereik verrassend goed voor zo'n klein apparaatje: het doet niet onder voor dat van een smartphone.
De webserver luistert dan, op het lokale IP-adres dat door de router is toegewezen, bijvoorbeeld http://192.168.10.1/, naar requests voor `/`, `/index.html` en `/frame`.

Nadat de WLAN-verbinding tot stand is gebracht, krijgt de webserver te horen wat hij moet doen als hij een specifiek verzoek van de browserclient ontvangt. De toegewezen functie wordt dan aangeroepen, bijvoorbeeld bij `/` (hetzelfde als `/index.html`) levert de functie `MW_index()`, het document aan de client: `webServer.send_P(200, "text/html", _uidoc);`
`Send_P` wordt gebruikt omdat het HTML-document `_uidoc` als `PROGMEM` in het flashgeheugen wordt bewaard om RAM-geheugen te besparen. Daardoor kan MTheCam kleurrijke warmtebeelden weergeven op elke smartphone. De programmagedeelten voor WLAN- en webserver zijn te bekijken in **listing 3.**

HTML en JSON

De requests `/` en `/index.html` genereren een webpagina met 8x8 kleurvlakken waarin de temperatuurwaarden worden weergegeven. Daarnaast worden de minimum (tMin), maximum (tMax) en gemiddelde temperatuur (tAvg) van de meting weergegeven. Elke kleurtoon komt overeen met een temperatuur, waarbij het kleurverloop dynamisch wordt aangepast binnen het MIN- en MAX-waardenbereik (`tminrange`/`tmaxrange`). De knop ***image*** schakelt het warmtebeeld om naar een false-color beeld van geïnterpoleerde waarden met 'hoge resolutie', de knop ***values*** schakelt weer terug. Elke seconde wordt een nieuwe meting weergegeven, dit interval kan met een schuifregelaar worden gewijzigd. Het selectievakje ***grid*** legt een raster over het beeld. Het request `/frame` retourneert de meetwaarden als een JSON-dataset voor verdere verwerking door andere programma's. Laat na elke aanroep minstens 200 ms pauze toe zodat de sensorchip opnieuw kan meten, wat ongeveer 100 ms duurt. Zo'n dataset kan indien nodig met strings worden opgebouwd, maar de handige bibliotheek `ArduinoJson.h` maakt dat veel gemakkelijker en kan ook dat soort data inlezen. Het resultaat ziet er zo uit:

```
{"device": "MTheCam","ver": "1.0","rowsize":
   8,"batvolt": 3.39,
"tavg": 2317,"tmin": 2050,"tmax": 2750,"tminrange":
   1817,"tmaxrange": 2817,
"frame": [2075, 2050, 2100, 2275, 2600, 2475, 2100,
   2050, , ..... ]} // 64 values °C * 100 // + 2
   arrays with 'HSL'-colors of 8*8 and 32*32 tiles
```

Listing 4 toont de interessante gedeelten van de HTML-tekst. Het document met CSS- en HTML-tags wordt gedefinieerd als een vaste string en naar de browser-client gestuurd. Het thermische beeld wordt opgebouwd uit een CANVAS met 8*8 (values) of 32*32 (image) pixels. Een script haalt dan onder timerbesturing de sensorgegevens (*/frame*)

Listing 4. Aanmaken van een HTML-document met meetwaarden.

```
. . .
/* Canvas-Area 400 px with 8*8/32*32 tiles à 50/12 px */
<div class="row">
      <div class="column">
        <canvas id="canvas"></canvas>    /* size see resizeCanvas() */
      </div>
</div>
. . .
/* resize canvas for value or image mode */
function resizeCanvas () {
 . . .
     canvasSize = rectSize * gridSize    /* gridSize: 8 or 32 @rectSize: 12 or 50 */
     canvas.width  = canvasSize
     canvas.height = canvasSize
. . .
/* update image,  rate = 200…3000 ms */
function refresh (rate = 1000) {
 . . .
refresher = setTimeout(function () {
        getData()
        }, rate)
. . .
/* request temperature-frame from MTC via /frame - call  */
function getData () {
     fetch(url)
        .then(response => response.json())
        .then(jsonData => {
          if (mode == 1) {
            // tempvalues, colors (interpolated) in framecolinter array
            paint(jsonData.frame, jsonData.framecolinter);
          } else {
            // tempvalues, colors (original) in framecol array
            paint(jsonData.frame, jsonData.framecol)
          }
          infoText(jsonData)             /* show min/max etc. values */
        }).catch(err => {
          errorText.innerHTML = 'Error: $'
      }).then(function () {
        refresh()                        /* start timer for next ride */
     })
}
. . .
/* paint tiles on canvas */
const ctx  = canvas.getContext("2d")

function paint(values, colors) {
  . . .
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   colors.foreach(color, index) ….  {
     . . .   /* set x/y of next tile: x+=rectSize … */ . . .
     /* paint tile */
     ctx.beginPath()
     ctx.rect(x, y, rectSize, rectSize)           /* paint square */
     ctx.fillStyle = 'hsl($, 100%, 50%)'   /* set color of square */
     if (grid) { ctx.stroke() }                    /* show grid optional */
     ctx.fill()                                   /* do fill */
     ctx.closePath()
     if (mode == 0) {                             /* values as text over */
         // show tempvalues @mode=0 -> lowres 8*8 only
         drawText(values[index], x + 7, y + rectSize / 2)
       }  }
 . . .
```

op, die vervolgens worden aangeleverd in de vorm van een JSON-packet. Javascript en JSON kunnen goed samenwerken, zodat de gegevens heel gemakkelijk uit het pakket kunnen worden geëxtraheerd: zowel *jsonData.frame* als *jsonData.framecol* voorzien de arrays rechtstreeks van de waarden en kleuren waaruit de oorspronkelijke temperatuur- en geïnterpoleerde gradiëntwaarden worden bepaald, de corresponderende gebieden worden op de juiste manier ingekleurd en de warmtewaarden worden er als getallen overheen gelegd. Het update-interval voor dit scherm kan worden ingesteld van 200 ms tot 3 s.

Een goede basis voor verdere projecten

In een uitgebreide firmwareversie creëert de ESP32 een eigen WiFi-netwerk, dat gegevens levert zolang er geen verbinding met een bestaand draadloos netwerk beschikbaar is en dat een gemakkelijke invoer van toegangsgegevens naar andere WiFi-netwerken mogelijk maakt. De kleurweergave en de kleurenloop kunnen instelbaar worden gemaakt voor een optimale weergave, de filterberekening vermindert de ruis en het flikkeren. Een Android-app zou meerdere MTheCam-apparaten kunnen beheren en configureren en ze tegelijk live weergeven. Het is ook een mogelijkheid om alarmtoestanden en -berichten te definiëren en te laten triggeren, waarbij Node-Red via een MQTT-server het gebruik van de gegevens vergemakkelijkt. Het is dus de moeite waard om af en toe de projectpagina te bekijken [9]. De basis voor slimme praktische toepassingen is gelegd, nu is er ruimte voor ideeën! Hartelijk dank aan Daniel Zelosko voor het ontwikkelen van het prototype.

180337-04

GERELATEERDE PRODUCTEN

› **M5StickC**
www.elektor.nl/m5stack-m5stickc-esp32-pico-mini-iot-development-board

Vragen of opmerkingen?
Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur of naar de redactie van Elektor, via redactie@elektor.com.

Een bijdrage van
Schema, software en tekst: **Olaf Mertens (Micom)**
Redactie: **Rolf Gerstendorf**
Vertaling: **Evelien Snel**
Layout: **Giel Dols**

WEBLINKS

[1] **Pyrometer**: https://nl.wikipedia.org/wiki/Pyrometer
[2] **Interpolatie in foto's**: https://de.wikipedia.org/wiki/Interpolation_(Fotografie)
[3] **AMG88x**: https://eu.industrial.panasonic.com/products/sensors-optical-devices/sensors-automotive-and-industrial-applications/infrared-array
[4] **ESP-documentatie**: www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[5] **M5StickC**: www.elektor.de/m5stack-m5stickc-esp32-pico-mini-iot-development-board
[6] **VisualStudio**: https://code.visualstudio.com/
[7] **Arduino-JSON** : https://arduinojson.org/
[8] **M5StickC-bibliotheek**: https://github.com/m5stack/M5StickC
[9] **Projectpagina van de auteur**: www.micom.de/lab/mthecam
[10] **HSV-kleurruimte**: https://de.wikipedia.org/wiki/HSV-Farbraum