Ausdauern Infotafel

Drahtloses Türschild mit E-Paper-Display

Ein Belegungsplan für den Konferenzraum, die Anwesenheit von Kollegen im Büro oder der Putzplan für die WG: Ein digitales Türschild ersetzt Zettelwirtschaft und Papieraushänge.

Von Jan Mahn

Ein digitales Türschild mit einem LCD ist schnell gebaut: eine Wandhalterung mit einem alten Tablet, darauf eine App oder eine Webseite im Vollbild. Das Problem ist der Stromverbrauch, denn selbst mit einem neuen Akku wird die Lösung nur wenige Tage durchhalten. Schließlich dürstet der Bildschirm ununterbrochen nach Strom. Wer ein Türschild plant, das mehrere Monate mit einer Akkuladung hält, kommt an E-Paper-Displays nicht vorbei. Diese benötigen nur dann Energie, wenn der Bildinhalt wechselt. Kombiniert mit dem Tiefschlafmodus des ESP32 entsteht ein digitales Türschild mit langer Akkulaufzeit.

Für das E-Paper-Türschild haben wir ein 7,5-Zoll großes Display verwendet, das der Hersteller Waveshare mit einem Aufsteckmodul für den Raspberry Pi anbietet. Das Panel kostet beim deutschen Händler 58 Euro, hat eine Auflösung von 640 × 384 Pixel und läuft, wie der ESP32, mit 3,3 V. Vom selben Hersteller gibt es auch Varianten mit 2,9 und 4,2 Zoll.

Um Strom zu sparen, wollen wir dem Mikrocontroller möglichst wenig Arbeit überlassen: Es soll einmal stündlich aus dem Tiefschlaf aufwachen, sich mit dem WLAN verbinden, per HTTP-Anfrage von einem Webserver das anzuzeigende Bild als Rohdaten herunterladen, dieses auf dem Display anzeigen und sich wieder in den Tiefschlaf begeben. Da die Gestaltung des Bildes auf dem Webserver erledigt wird, können Sie die Inhalte jederzeit austauschen, ohne neuen Code auf den ESP32 zu flashen. Ein Webserver kann auch mehrere Schilder mit Bildern versorgen und beispielsweise für jedes Konferenzraumschild ein Bild mit der aktuellen Belegung zusammenbauen.

Prototyp

Den ersten Aufbau sollten Sie mit einem ESP32-Entwicklerboard mit integriertem Progammieradapter zusammenstecken. Waveshare liefert eine Kabelpeitsche mit Jumper-Kabeln mit, die Belegung finden Sie in der Tabelle rechts oben.

Ist die Arduino-IDE mit der ESP32-Erweiterung und unserer Bibliothek Basecamp eingerichtet (siehe S. 64), kann die Arbeit am Code für das Türschild beginnen. Installieren Sie zu Beginn die beiden externen Bibliotheken „GxEPD“ und „Adafruit_GFX“. Die Links dafür finden Sie unter ct.de/yrzv. Laden Sie den Inhalt der Repositories als zip-Dateien herunter und entpacken sie im Ordner „libraries“ im Arduino-Verzeichnis. Starten Sie anschließend die IDE neu. Die Bibliotheken erledigen die Kommunikation mit dem Display. Laden Sie zunächst ein Beispielprojekt, das die Entwickler von „GxEPD“ mitgeliefert haben, um die Verkabelung zu prüfen. Sie finden es in der Arduino-IDE unter „Datei/Beispiele/GxEPD/GxEPD_SPI_TestExample“. In den Zeilen 46 bis 56 (Zeilennummern aktivieren Sie unter „Datei/Voreinstellungen“) binden Sie die für das Display passende Bibliothek ein und kommentieren Sie alle anderen mit „//“ aus. Für das 7,5-Zoll-Display entfernen Sie die Kommentarzeichen für `#include <GxGDEW075T8/GxGDEW075T8.cpp>`.

Pinbelegung E-Paper-Display

Display	ESP32
BUSY	4
RST	16
DC	17
CS	5
CLK	18
DIN	23
GND	GND
3.3V	3.3V

Schließen Sie das ESP-Entwicklerboard per USB an und flashen Sie das Beispiel auf den Mikrocontroller – stimmt die Verkabelung, zeigt das Display ein Testprogramm mit Bildern und Texten an. Die Funktion zur Bildausgabe, die in der Bibliothek mitgeliefert wird, verlangt als Bildformat ein Array mit Hexadezimalwerten – solche Bilder können Sie zwar problemlos auf den ESP32 flashen. Um es per HTTP zu übertragen, ist es aber ungeeignet.

Pixelstrom

Das E-Paper-Display stellt bauartbedingt nur monochrome Bilder dar, kennt also nur zwei Zustände für einen Pixel – ein oder aus. Das Display hat 640 Pixel in der Breite und 384 in der Höhe, insgesamt also 245.760 Bildpunkte. Unser Bildformat ist auf das Wesentliche reduziert: Beginnend in der linken oberen Ecke werden die Schaltzustände der Pixel zu einem String zusammengefügt – je acht Bits werden zu einem Byte zusammengefasst. Aus Pixelfolge `00101011` wird beispielsweise das ASCII-Zeichen +. Am Ende der Zeile folgen kommentarlos die Pixel der nächsten Zeile.

Bevor die Hardware Bilddaten empfangen kann, müssen diese erst einmal in ein geeignetes Datenformat umgewandelt werden. Auf dem Webserver haben wir das mit PHP realisiert (siehe Listing auf S. 70).

Eingelesen wird das Bild „bild.png“, das von der Funktion `createMonochromeImage()` Zeile für Zeile zerlegt wird. Für jedes Pixel prüft das Programm, ob in einem der Farbkanäle Rot, Grün oder Blau ein Farbwert gesetzt ist und hängt das Zeichen 0 oder 1 an den String an. Dieser Code ersetzt bei aufwendigen farbigen Bildern keine Umwandlung durch ein Bildbearbeitungsprogramm – jeder nicht weiße Pixel wird auf dem Display schwarz. Ist die Länge des Strings durch acht teilbar, wird das nächste Byte erzeugt und an den Ausgabe-String angehängt.

Aus diesem Material muss der ESP32 jetzt wieder ein Bild zusammensetzen. Nachdem er sich den String vom Webserver geholt hat, liest er ihn Byte für Byte ein, zerlegt die Bytes wieder in Bits und schaltet die Pixel des E-Paper-Displays einzeln ein oder aus. Innerhalb der Schleife ist das aktuelle Byte in der Variable `byte`:

```
for (int b = 7; b >= 0; b--) {
 int bit = bitRead(byte, b);
 if (bit == 1) {
  display.drawPixel(x, y,GxEPD_BLACK);
 } else {
  display.drawPixel(x, y,GxEPD_WHITE);
 }
 x++;
 if(x == 640) {
  y++;
  x = 0;
 }
}
```

Einkaufsliste Steckdose

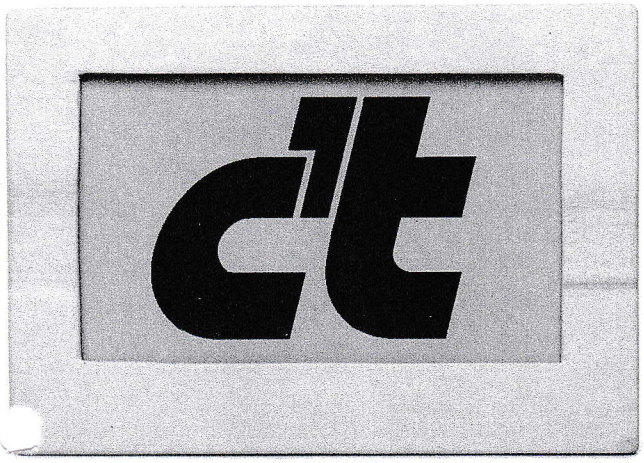
- 4 Stunden (mit Gehäusebau)
- 80 €
- ESP32
- Waveshare 7.5inch E-Ink display HAT for Raspberry Pi
- Multiplex
- Lötkolben
- Stichsäge

leicht schwer

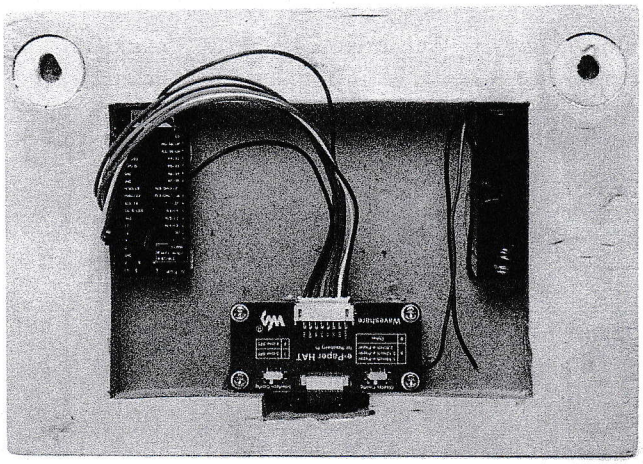
Aus den Bytes werden stückweise wieder acht Bildpunkte, die direkt aufs Display geschrieben werden. Nach jeweils 640 Spalten beginnt die nächste Zeile. Sollten Sie ein kleineres Modell einsetzen, müssen Sie die Breite aus dem jeweiligen Datenblatt des Herstellers entnehmen und den Wert eintragen. Sind alle Bytes eingelesen, folgt der Befehl `display.update()`, damit das Display die neuen Bildpunkte anzeigt.

Tiefschlaf

Die HTTP-Anfrage und der Bildaufbau haben nur wenige Sekunden gedauert. Jetzt ist es für den ESP32 Zeit, sich wieder schlafen zu legen. Soll er das nächste Bild in 60 Minuten abrufen, bekommt er eine Wartezeit von 3.600.000.000 Mikrosekunden (3600 Sekunden): `esp_sleep_`



Damit das digitale Türschild an der Wand seiner Arbeit nachgehen kann, bekommt es ein Gehäuse aus Holz.



Im Hohlraum auf der Rückseite finden ESP32 und Batteriefach Platz.

```
$im = imagecreatefrompng("bild.png");
echo createMonochromeImage($im);
function createMonochromeImage($im) {
 $bits = "";
 $bytes = "";
 $pixelcount = 0;
 for ($y = 0;$y<imagesy($im); $y++){
   for ($x = 0;$x<imagesx($im); $x++){
    $rgb = imagecolorat($im, $x, $y);
    $r = ($rgb >> 16) & 0xFF;
    $g = ($rgb >> 8 ) & 0xFF;
    $b = $rgb & 0xFF;
    $gray = ($r + $g + $b) / 3;
    if ($gray < 0xFF) {
     $bits .= "1";
    }else {
         $bits .= "0";
    }
    $pixelcount++;
    if ($pixelcount % 8 == 0) {
     $bytes .= pack('H*',str_pad(base_convert($bits,2,16),2,"0",STR_PAD_LEFT));
     $bits = "";
    }
   }
  }
  return $bytes;
}
```

Für das E-Paper-Display berechnet das PHP-Skript aus einer PNG-Datei einen String aus Bytes mit je acht monochromen Pixeln.

`enable_timer_wakeup(3600000000)` und verabschiedet sich anschließend mit `esp_deep_sleep_start()` in den Tiefschlaf.

Rahmenbedingungen

Den Code zum Bildaufbau haben wir zusammen mit unserer Bibliothek Basecamp bereitgestellt, sodass Ihr erstes Türschild schnell einsatzbereit ist. Das Repository enthält im Ordner „Server" Anregungen, was der Webserver darstellen könnte. Das gesamte Projekt finden Sie unter ct.de/yrzv. Flashen Sie das Programm auf den ESP32, verbinden Sie sich mit dem WLAN, das dieser bereitstellt, und öffnen Sie die Konfigurationsoberfläche (siehe S. 67). In der Weboberfläche geben Sie jetzt die Adresse des Servers und den Pfad zum Bild sowie die gewünschte Schlafzeit ein. Anschließend startet der Mikrocontroller neu und beginnt mit der Arbeit.

Sandwich-Gehäuse

Beim Bau des Gehäuses für das 7,5 Zoll große Display haben wir ein Design entwickelt, das Sie ohne 3D-Drucker und CNC-Fräse nachbauen können. Es ist schichtweise zusammengeleimt und besteht aus einer Schicht Sperrholz (3 mm), zwei Lagen Hartfaserplatte (4 mm) und einer Schicht Multiplex (18 mm). Die Zeichnung mit allen Maßen finden Sie im Projekt-Repository im Ordner „case", die Verarbeitung funktioniert mit Stich- und Kreissäge (oder Zuschnitten aus dem Baumarkt) auch mit wenig Werkzeug. Legen Sie das Display in seine Aussparung in der zweiten Ebene und verbinden Sie die Schichten mit Holzleim. Das Flachbandkabel führen Sie bis in die letzte Ebene durch. Dort finden ein ESP32 und ein Batteriefach Platz. Mit eingelassenen Schlüssellochblechen befestigen Sie das Schild elegant an der Wand. Um das Projekt über einen Akku mit Strom zu versorgen, finden Sie zwei Ansätze im Projekt „Türsensor" auf Seite 76.

Luftpost-Update

Das Überspielen der Firmware auf den ESP32 ist kein Problem, solange Sie ein ESP-Entwicklerboard mit integriertem USB-Adapter verwenden und die Hardware neben dem Computer liegt. Hängt das Schild erst einmal fertig montiert an der Wand, sind Veränderungen am Code aber sehr aufwendig. Wenn Sie das Entwicklerboard durch einen einzelnen ESP32 ersetzt haben, müssten Sie für jedes Update einen Programmieradapter anklemmen. Dankenswerterweise haben die Entwickler die Möglichkeit integriert, Updates per WLAN einzuspielen. Im Programm müssen Sie dafür nur wenige Zeilen Code ergänzen:

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

void setup() {
 // [...] WLAN-Verbindung herstellen

  ArduinoOTA.setPassword("secret");
  ArduinoOTA.begin();
}
void loop() {
  ArduinoOTA.handle();
}
```

Überspielen Sie dieses Programm auf klassischem Weg per USB. Ist der ESP32 mit dem gleichen WLAN verbunden wie der Computer mit der Arduino-IDE, finden Sie ihn im Menü unter „Werkzeuge/Ports" unterhalb der seriellen Ports – zu erkennen an seiner IP-Adresse. Achten Sie darauf, dass in jeder veränderten Version des Programms die Zeilen für das OTA-Update erhalten bleiben. Fehlen diese, haben Sie sich ausgesperrt und müssen wieder den USB-Programmer bemühen.

In unserer Bibliothek Basecamp sind OTA-Updates bereits integriert und aktiviert. Um ein Kennwort zu setzen, verwenden Sie die Zeile `configuration.set("OTAPassword", "IHR KENNWORT")`. Zum Deaktivieren der Funktion `configuration.set("OTAActive", "false")`.

Weiterdenken

Mit den veröffentlichten Beispielen und dem Gehäuse haben wir einige Anregungen zusammengestellt – wenn Sie eigene Ideen für Inhalte oder Gehäuse-Baupläne für die kleineren Displays haben, lassen Sie es uns wissen oder erstellen Sie einen Pull-Request bei GitHub. *(jam@ct.de)* c't

Repository und Downloads: ct.de/yrzv