disque de stationnement avec affichage e-papier

une version informatisée innovante

Antonello Della Pia (Italie)

Un disque de stationnement est un accessoire présent dans toutes les voitures. Ce projet en présente une version numérique utilisant un afficheur e-papier, avec des caractéristiques particulières, telles qu'un bouton unique pour le réglage de l'heure d'arrivée, un message en quatre langues au choix et l'affichage à la demande de l'heure et de la date courantes, de la température ambiante et du niveau de la batterie.

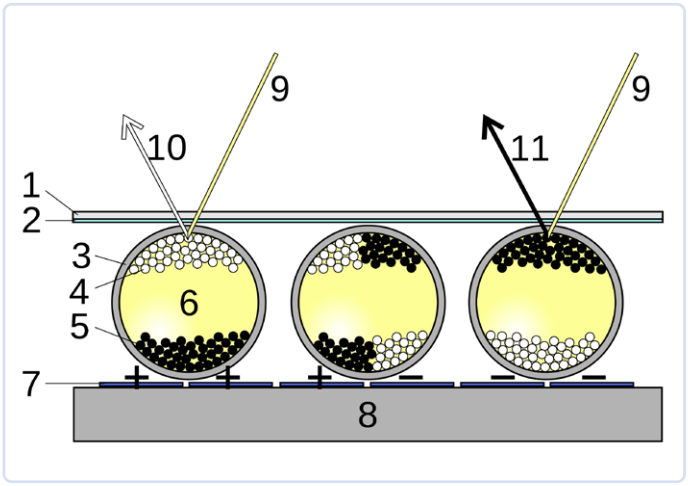

Figure 1. Principe de base de fonctionnement de l'afficheur e-papier. (1 couche supérieure, 2 couche transparente d'électrodes, 3 micro-capsules transparentes, 4 pigments blancs chargés (+), 5 pigments noirs chargés (-) , 6 fluide transparent, 7 couche d'électrode de pixel, 8 couche inférieure, 9 lumière, 10 blanc, 11 noir). (Source : https://en.wikipedia.org/wiki/E_Ink)

L'intégration de la technologie électronique dans les véhicules a atteint des niveaux extrêmement élevés. Un nombre toujours croissant de capteurs performants, d'interfaces numériques, de microprocesseurs et de logiciels associés assurent désormais une gestion optimale de pratiquement tous les processus physiques. Dans l'habitacle de ce que l'on peut désormais considérer comme des « ordinateurs sur roues », il reste quand même un instrument indispensable, souvent en carton et à commande manuelle : le disque horaire, obligatoire pour indiquer l'heure de début du stationnement dans les zones réglementées. Au fil des décennies, cet accessoire est resté pratiquement inchangé – en carton, en plastique ou autre matériau plus noble – et ce n'est que récemment que des modèles numériques sont apparus sur le marché. Celui proposé dans l'article utilise un afficheur moderne e-papier et présente quelques caractéristiques particulières, comme le réglage de l'heure d'arrivée à l'aide d'un seul bouton, un message en quatre langues au choix, l'affichage à la demande de l'heure et de la date courantes, de la température ambiante et du niveau de la batterie.

L'afficheur e-papier

Invention relativement récente (1996), la technologie e-ink (encre électrophorétique), généralement appelée e-papier [1], doit son succès avant tout à son utilisation dans les liseuses de livres électroniques, ces dispositifs portables qui offrent une alternative électronique aux livres traditionnels, grâce à une expérience de lecture semblable à celle du papier et à une visibilité parfaite, même par forte luminosité ambiante. Toutefois, la caractéristique unique qui a conduit à la diffusion de cette technologie dans d'autres domaines est qu'il n'y a pas de consommation d'énergie pour maintenir l'affichage des informations, même pendant une longue période, mais seulement pour la mise à jour de l'afficheur. Les applications typiques, de plus en plus populaires dans les points de vente, sont

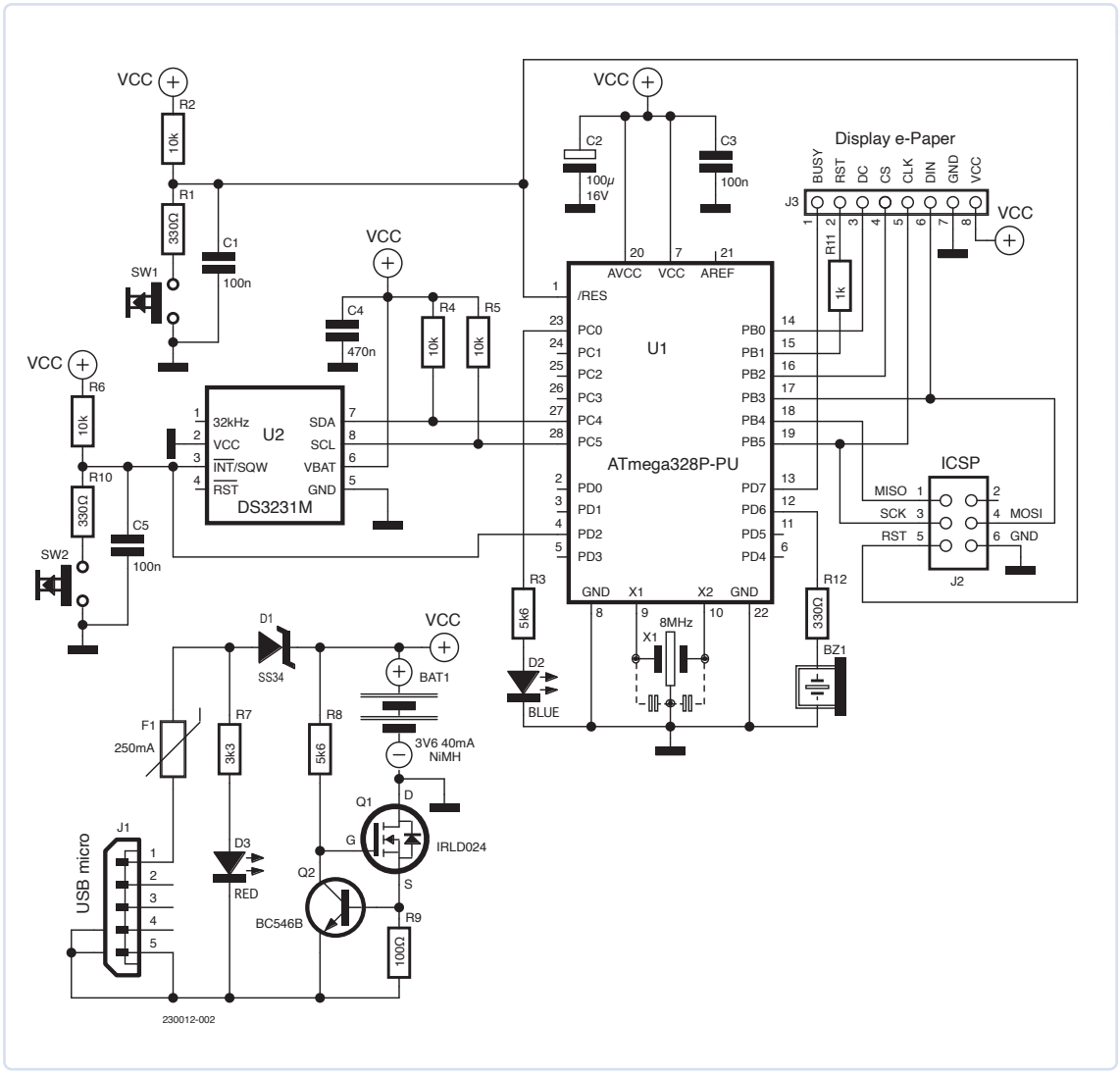

Figure 2. Schéma du circuit.

les étiquettes électroniques d'articles et de prix, souvent difficiles à distinguer des étiquettes en papier, faciles à mettre à jour quand c'est nécessaire, même à distance grâce aux technologies sans fil. La **figure 1** va nous aider à mieux comprendre le fonctionnement de l'encre électronique.

Dans la version la plus simple, l'affichage en noir et blanc, des pigments chargés positivement (blanc) et négativement (noir) sont en suspension dans un liquide contenu dans des microsphères représentant des pixels. Grâce à la polarisation créée par un champ électrique approprié, les pigments, attirés par la charge de signe opposé (électrophorèse), se positionnent pour créer des pixels noirs ou blancs, composant l'image désirée. À ce stade, même si le champ électrique est supprimé, les pigments restent en position jusqu'à ce qu'une nouvelle charge soit appliquée. La visibilité, sous un angle particulièrement large, est obtenue par réflexion de la lumière ambiante, ou, en son absence, d'une source lumineuse. Notons toutefois que la lumière directe du soleil peut perturber la mise à jour correcte de l'afficheur. Sur la base de ce principe de fonctionnement, de nombreux types d'afficheurs ont été réalisés, même de grande taille et en couleur, qui restent cependant très chers. Parallèlement, l'offre abordable de modèles plus petits, en noir et blanc, en niveaux de gris ou à gamme de couleurs limitée, proposée par des détaillants spécialisés, a augmenté. En revanche, l'intérêt des bricoleurs pour ces composants n'a pas suivi cette augmentation, d'après ce que j'ai pu voir sur Internet. À mon avis, cela est dû à différents facteurs critiques, devenus apparents au cours du développement de ce projet, tels que la pléthore de modèles, de versions, de tailles, de pilotes et de gammes de couleurs qui ont envahi le marché, le manque de bibliothèques bien documentées et faciles à mettre en œuvre sur les différentes plates-formes de développement, et des informations et un soutien souvent fragmentaire et insuffisant de la part des fabricants eux-mêmes. Malgré cela, en choisissant parmi les produits les mieux supportés et en appliquant les méthodes des faiseurs, il a été possible d'obtenir un résultat que je crois intéressant. Poursuivons donc avec l'analyse du schéma du circuit (**figure 2**).

Schéma

Un microcontrôleur (MCU) ATmega328P [2] a été choisi pour gérer l'affichage, le même que celui utilisé dans la carte Arduino UNO, étant donné la disponibilité d'une bibliothèque polyvalente spécifique à cette plateforme. Le MCU fonctionne à une fréquence d'horloge de 8 MHz, grâce à un résonateur céramique externe, et se trouve normalement en mode sommeil (*SLEEP_MODE_PWR_DOWN*), pour ne se réveiller que si l'affichage doit être mis à jour. La puce DS3231M [3], l'horloge en temps réel (RTC), quant à elle, reste active en permanence, capable de maintenir, tant qu'elle est alimentée, la date et l'heure avec une précision de ±5 ppm (±0,432 seconde/jour). Elle communique avec le microcontrôleur via une interface

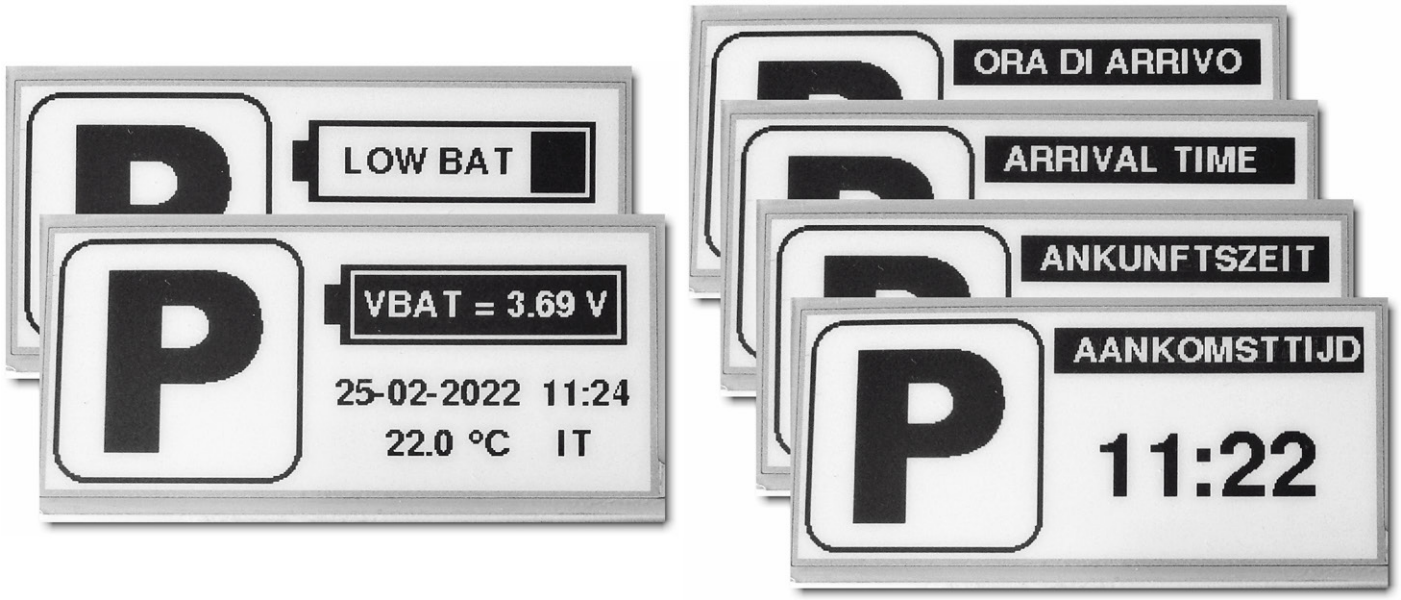

Figure 3. Exemples d'affichages info et parking.

I²C et une bibliothèque dédiée. Parmi ses autres caractéristiques notables, citons un capteur de température, une double alarme et une consommation de courant extrêmement faible, de l'ordre de quelques microampères (µA) dans la configuration utilisée, à source d'alimentation unique (VBAT). Le bouton SW2 (bleu dans le prototype) « réveille » le MCU, qui peut ainsi recevoir l'heure courante et la transmettre à l'afficheur, ainsi que le message « TIME OF ARRIVAL » (HEURE D'ARRIVÉE) et le logo « P », avant de retourner en mode sommeil. Un court bip du bipeur confirme l'action du bouton tandis que la LED bleue D2 reste allumée pendant la durée de l'opération. Le bouton SW1 (rouge dans le prototype) produit une réinitialisation matérielle de l'ATmega328P, suivie immédiatement par l'affichage du logo « P », le symbole et la tension de la batterie, la température ambiante, l'heure et la date courantes et la langue sélectionnée pour le message (**figure 3**).

Toutes les 24 heures (à l'heure souhaitée, grâce à la fonction d'alarme de la RTC), selon la recommandation du fabricant, un rafraîchissement complet de l'affichage est effectué, qui permet aussi de vérifier l'état de la batterie et le bon fonctionnement de l'ensemble du système. Par conséquent, nous pouvons appeler SW2 le « bouton parking » et SW1 le « bouton info ». Les résistances R1 et R10, avec les condensateurs C1 et C5, contribuent à supprimer tout bruit dû au rebondissement des contacts des boutons-poussoirs, tandis que R2, R4, R5, R6 sont des résistances de rappel. C2, C3 et C4 sont les condensateurs de découplage habituels de l'alimentation des circuits intégrés. R11 limite le courant tiré par la broche de réinitialisation de l'afficheur utilisé. JP2 est le connecteur de l'afficheur, tandis que JP1 permet de connecter un programmateur USBasp pour télécharger le micrologiciel. En raison de la faible consommation de courant, une petite batterie rechargeable Ni-MH de 3,6 V avec une capacité de 40 mAh suffit pour l'alimentation. Le circuit formé par Q1, Q2, R8, R9 est un régulateur de courant constant à faible chute de tension, qui limite le courant de charge à environ 6 mA, une valeur que ce type de batterie peut supporter sans inconvénient même si les 14…16 heures de charge prévues sont dépassées.

Une prise micro-USB permet de connecter un chargeur standard de 5 V, via un fusible de protection réarmable, tandis que la diode D1 empêche l'inversion du courant de la batterie. La LED D3 indique que l'appareil est en charge. Nous reviendrons plus en détail sur l'alimentation électrique un peu plus tard.

Réalisation pratique

Le prototype du disque de stationnement a été réalisé comme d'habitude sur une carte de prototypage (**figure 4**). Bien sûr, ce n'est qu'une des nombreuses possibilités ; en utilisant des CMS et un circuit imprimé spécial, on pourrait probablement obtenir

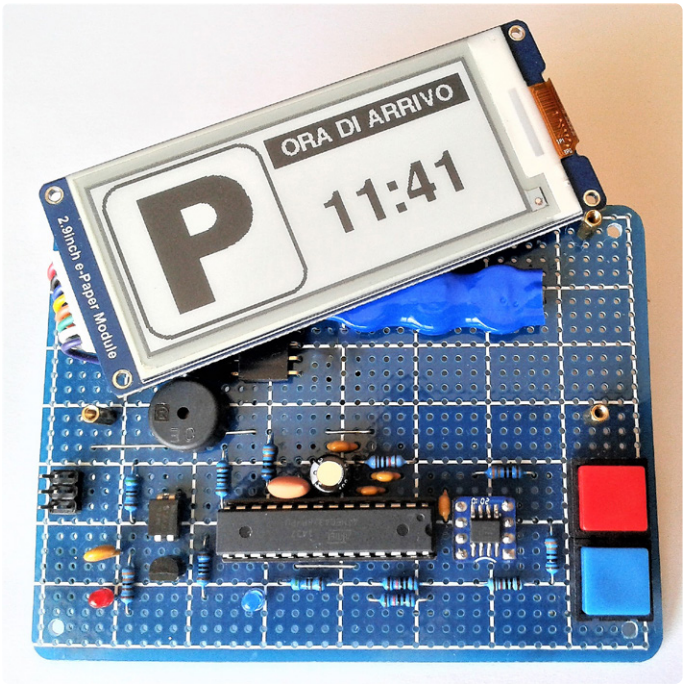

Figure 4. Le prototype, côté composants.

Figure 5. Le module e-papier utilisé dans le projet avec son connecteur « personnalisé ». (Source : Waveshare)

une taille plus petite, légèrement plus grande que l'afficheur. La batterie et le bipeur sont situés sous le module e-papier, tandis que le connecteur à huit broches de ce dernier est réalisé à partir de barrettes mâle et femelle (**figure 6**). Les circuits intégrés sont montés sur embase et un adaptateur SO-8 facile à trouver a été utilisé pour la puce DS3231M. Le choix du boîtier est libre, pourvu qu'il possède une fenêtre transparente !

En ce qui concerne le module e-papier, comme déjà mentionné, il est parfois difficile de s'y retrouver dans l'abondante offre de composants apparemment similaires. Celui utilisé dans le prototype (**figure 5**) est un *Waveshare* de 2,9 pouces de diagonale, noir et blanc, de résolution 296 x 128 [4]. Il possède un convertisseur de niveau logique intégré et supporte le rafraîchissement partiel, une fonctionnalité indispensable dans ce projet. D'autres modèles peuvent ne pas arriver aux résultats que j'ai obtenus, ou nécessiter des modifications du micrologiciel ou du schéma de câblage. Il faut aussi tenir compte de la plage de température de fonctionnement autorisée, qui, dans ce cas, s'étend de 0 à 50 °C.

La **figure 6** montre le côté soudure du prototype, sur lequel se trouvent également la diode D1 (type SMD) et le connecteur de rechargement micro-USB.

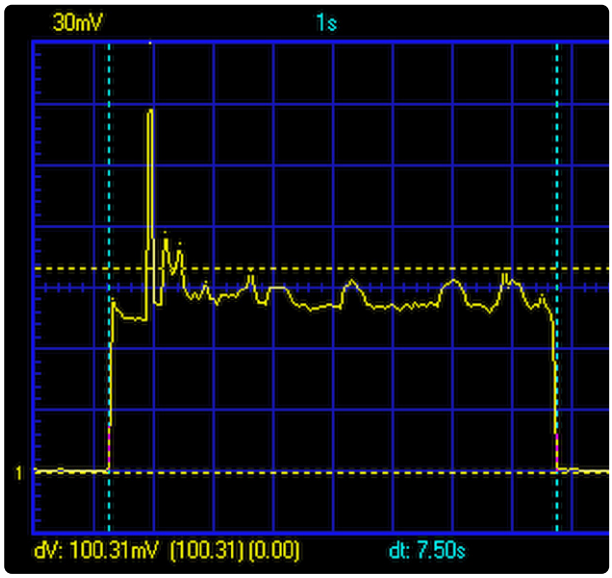

Figure 7. Oscillogramme du courant consommé par le circuit.

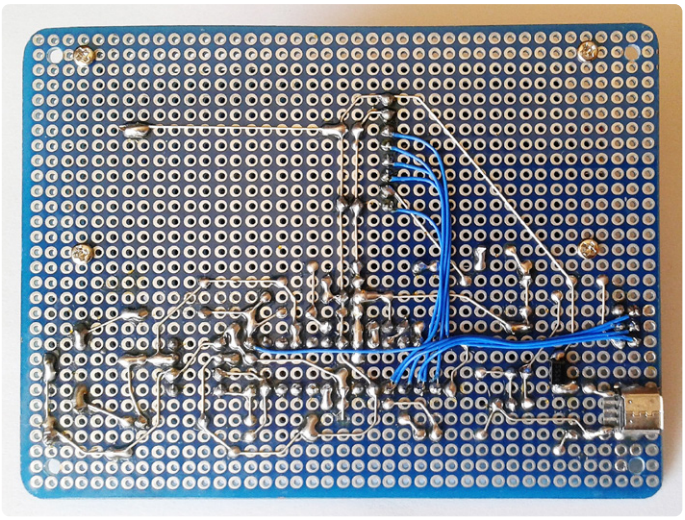

Figure 6. Le prototype côté soudures.

Pour en revenir à l'alimentation électrique, le choix d'une batterie Ni-MH rechargeable, une technologie qui date déjà un peu, est principalement motivé par la recherche d'une sécurité et d'une fiabilité accrues dans les conditions pas toujours optimales d'utilisation dans un véhicule. Par rapport, par exemple, aux batteries Li-Ion, le risque d'explosion, même lointain, est nul et même les performances à basse température sont meilleures. La charge normale est certes plus lente, mais plus facile à gérer. Quoi qu'il en soit, compte tenu des courants minimes en jeu et de la tension de fonctionnement du circuit, qui peut varier sans problème entre 3,3 V et 5,0 V, une solution utilisant trois piles alcalines AAA en série peut également être envisagée, éliminant ainsi la nécessité du circuit de charge et assurant une durée de vie qui peut dépasser celle de la voiture ! Quantifier avec précision la « consommation » des appareils à piles, de plus en plus populaires, qui alternent périodes d'activité et pauses en mode d'économie d'énergie (appelé aussi mode veille, hibernation ou basse énergie) n'est pas simple ; malheureusement, il ne suffit pas d'insérer un multimètre en série avec l'alimentation et de lire le courant. Le calcul de base à effectuer pour obtenir la valeur moyenne du courant consommé, et donc une estimation de l'espérance de vie de la batterie, peut être représenté par la pseudo-formule suivante :

$$I_{AVG} = \frac{[(I_{ON} \cdot T_{ON}) + (I_{SBY} \cdot T_{SBY})]}{(T_{ON} + T_{SBY})}$$

où I_{AVG} représente le courant moyen, I_{ON} le courant en mode actif, I_{SBY} le courant de veille, T_{ON} le temps d'activité et T_{SBY} le temps de veille.

J'ai observé les fluctuations du courant actif à l'oscilloscope (**figure 7**) en détectant la chute de tension aux extrémités d'une résistance de précision de faible valeur en série avec l'alimentation, obtenant, avec une approximation acceptable, la valeur de 10 mA pendant un temps de 7,5 secondes, qui représente la durée d'un cycle d'affichage. Le courant de repos, mesuré à l'aide d'un

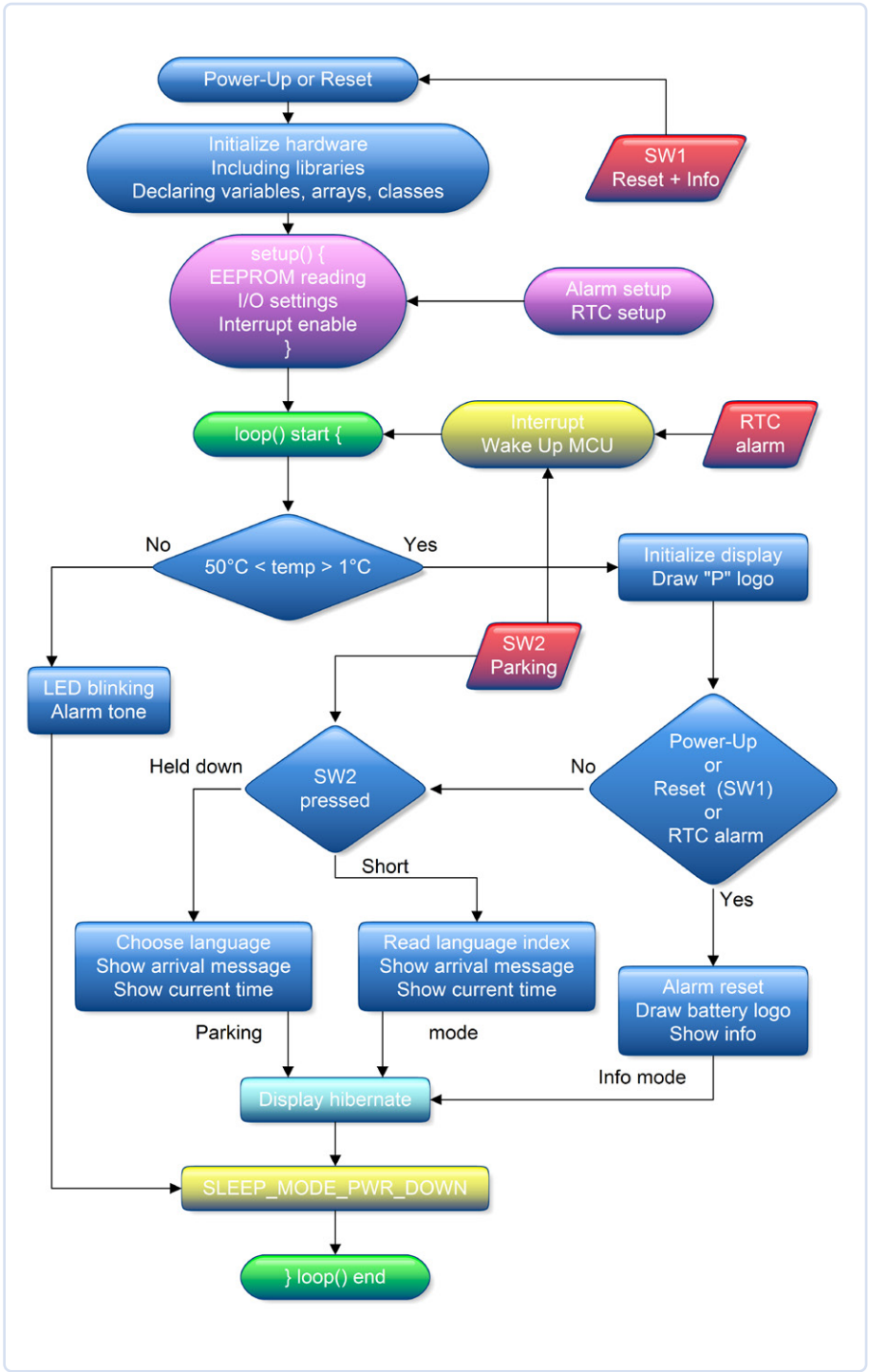

Figure 8. Organigramme du micrologiciel.

multimètre numérique, n'est que de 3 µA. En supposant une utilisation du disque de stationnement quatre fois par jour, plus le cycle de rafraîchissement, nous obtenons un temps de fonctionnement total de 7,5 x 5 = 37,5 secondes et un temps de repos de (24 x 3 600) - 37,5 = 86 362,5 secondes sur 24 heures. Nous obtenons donc :

$$I_{AVG} \approx \frac{[(10\,mA \cdot 3,75\,s) + (0,003\,mA \cdot 86362,5\,s)]}{(37,5\,s + 86362,5\,s)}$$

$$\approx 0,008\,mA$$

En négligeant pour simplifier, la réduction de capacité due à l'autodécharge, à une baisse de la tension nominale, aux variations de température et à d'autres facteurs, une batterie de 40 mAh, à un courant moyen de 0,008 mA, pourrait théoriquement alimenter le circuit pendant environ 40/0,008 = 5 000 h (c'est-à-dire plus de 200 jours ou au moins six mois) ! De manière plus réaliste, même la moitié de cette durée serait un bon résultat. Des applications en ligne sont disponibles pour ce type de calcul. En voici une [5] que j'ai trouvée efficace et pratique.

Le micrologiciel et sa fonctionnalité

Dans ce projet, le code source [9] est écrit avec l'EDI Arduino 1.8.19 et nécessite, pour une compilation correcte, l'installation du noyau *Arduino MiniCore* v2.1.3 et de quelques bibliothèques spécifiques. Le noyau utilisé permet une gestion plus souple et efficace du microcontrôleur ATmega328P et surtout optimise la taille mémoire du code compilé, qui occupe 31,264 des 32,768 octets de mémoire programme (Flash) et 1,501 des 2,048 octets de mémoire dynamique (SRAM), presque à la limite des capacités de ce MCU.

Il faut noter que ce projet n'est pas réalisable, même en tâtonnant, avec une carte Arduino Uno, car une partie de la mémoire de cette dernière est utilisée par le chargeur pour permettre une programmation directe, alors que pour le microcontrôleur nu, nous utilisons un programmateur USBasp externe. À propos de mémoire, le mot clé `PROGMEM` apparaît plusieurs fois dans le listage, faisant référence aux tableaux d'octets des chaînes de caractères bitmap et texte, qui sont des données en lecture seule. En déclarant ces tableaux comme `PROGMEM`, les fonctions peuvent accéder à ces données en les lisant directement depuis la mémoire Flash, sans les copier au préalable dans la SRAM, beaucoup plus petite, qui reste alors disponible pour l'exécution « dynamique » du programme. La bibliothèque *DS3231M 1.0.6* est utilisée pour la communication avec l'horloge en temps réel (RTC), tandis que la bibliothèque *GxEPD2 1.3.6*, supportée par la bibliothèque graphique *GFX_Root 2.0.0*, a été choisie pour la gestion de base de l'afficheur e-papier.

Cette dernière doit être remplacée par celle fournie avec le projet, qui comporte des polices modifiées. La bibliothèque *GxEPD2* est un monument qui ne dispose malheureusement pas d'une documentation structurée, qu'il faut aller chercher dans le code

des exemples disponibles, qui sont très nombreux, mais à première vue déconcertants en raison de leur complexité apparente, et dont on découvre par la suite qu'elle est due à la tentative d'être compatibles avec autant de modèles d'afficheur que possible. J'ai donc tenté de simplifier, en ne conservant que les fonctions et les définitions nécessaires au type d'afficheur utilisé dans le projet. Celles-ci se trouvent dans le fichier *Waveshare_29_BW_avr.h*, tandis que le fichier *ParkBitmap128x128.h* contient le tableau d'octets, obtenu au moyen d'un convertisseur spécial [6], représentant l'image bitmap du logo du parking (P majuscule, noir et blanc, inscrit dans un carré aux coins arrondis de taille 128 x128 pixels). Ces fichiers, disponibles au téléchargement, se trouvent dans le dossier du croquis, avec le fichier principal du code source *Disco_Orario_e-Paper.ino*, dans lequel on trouve également des liens vers les sites du noyau et de la bibliothèque, des commentaires approfondis sur le code, et d'autres indications que j'ai trouvées utiles. Je conseille aux lecteurs intéressés par les détails du listage de l'examiner en l'ouvrant avec l'EDI Arduino (ou leur éditeur préféré). Au lieu de cela, je voudrais illustrer ici le fonctionnement du programme d'une manière plus descriptive, à l'aide de l'organigramme de la **figure 8**.

En tenant compte du fait que le circuit est en permanence connecté à la batterie, lors de la première mise sous tension (*Power-Up*), les opérations d'initialisation et la fonction `setup()` sont exécutées, avec l'activation d'une interruption, puis la `loop()` est exécutée.

Au début de la boucle, la température ambiante est vérifiée. Si elle n'est pas dans la plage prévue, un signal d'alarme sonore et lumineux est émis, puis le microcontrôleur est placé en mode d'économie d'énergie maximale (*SLEEP_MODE_PWR_DOWN*). Sinon, la `loop()` continue en initialisant l'affichage avec le logo « P », l'état de la batterie, la date et l'heure courantes, la température et la langue sélectionnée pour le message d'arrivée, montrant ce que nous pouvons appeler le mode info, après quoi l'affichage passe en hibernation (économie d'énergie), le MCU est mis en sommeil et la `loop()` interrompue. À partir de cet état, le microcontrôleur peut être « réveillé » (*Wake Up*) par une réinitialisation matérielle (avec le bouton SW1) ou par une interruption, un événement généré par la fonction d'alarme quotidienne du RTC ou par le bouton SW2. Si la reprise est provoquée par SW1 ou l'alarme, la `loop()` redémarre depuis le début et se termine toujours en mode *info*. Par contre, si le microcontrôleur est réactivé via SW2, la `loop()` redémarre, initialise l'affichage avec le logo et, si le bouton a été pressé et immédiatement relâché, affiche le message d'arrivée et l'heure courante (mode *parking*). En revanche, en maintenant SW2 enfoncé, les messages dans les quatre langues défilent successivement. En relâchant le bouton lorsque la langue désirée apparaît, ce choix est mémorisé dans l'EEPROM jusqu'au prochain changement. L'heure courante apparaît alors, et le cycle se termine comme précédemment en mode parking. La **figure 3** montre un exemple des captures d'écran. Nous avons vu comment la date et l'heure courantes sont gérées et fournies par le circuit intégré RTC DS3231M, qui doit encore être programmé après la première mise sous tension et en cas de défaillance de la tension d'alimentation. Afin de simplifier le micrologiciel et le circuit, en évitant l'ajout de boutons supplémentaires, la programmation de la date et de l'heure est effectuée en même temps que le chargement du croquis, au moyen d'une ligne de code spéciale insérée dans la fonction `setup()` :

```
DS3231M.adjust(DateTime(2022, 03, 02, 19, 10, 00));
```

Liste des composants

Résistances (0,25 W, 1%)

R1, R10, R12 = 330 Ω
R2, R4, R5, R6 = 10 k
R3, R8 = 5,6 k
R7 = 3,3 k
R9 = 100 Ω
R11 = 1 kΩ

Condensateurs

C1, C3, C5 = 100 nF, 50 V condensateur en polyester ou céramique multicouche
C2 = 100 µF, 10 V condensateur électrolytique
C4 = 470 nF, 50 V condensateur en polyester ou céramique multicouche

Semi-conducteurs

Q1 = IRLD024 MOSFET à grille de commande par niveau logique
Q2 = BC546B
D1 = Diode Schottky à faible chute
D2 = LED bleue 3 mm, forte luminosité
D3 = LED rouge 3 mm, forte luminosité
U1 = ATmega328P-PU
U2 = DS3231M RTC, boîtier SO-8

Divers

Y1 = Résonateur céramique 8 MHz
BUZZER = Bipeur piézo-électrique
SW1, SW2 = N.O. Bouton-poussoir
F1 = Fusible réarmable (polyfusible) 250 mA
X1 = Prise Micro-USB type B
JP1 = Connecteur PCB ICSP 6 broches, M
JP2 = Connecteur PCB 8 broches M/F
AFFICHAGE = Afficheur e-papier 2,9 pouces N/B, (voir texte)
BAT1 = Batterie Ni-MH 3.6V, 40 mAh (3 x V40H)
Embase DIP 28 broches
Embase DIP 8 broches
Adaptateur CMS — DIP8 pour boîtier SO-8
Carte de prototypage

Le format à utiliser est du type « *YYYYY,MM,DD,hh,mm,ss* ». Une fois que vous avez entré les données correctes, vous pouvez charger le croquis, vérifier la date et l'heure, changer la ligne en commentaire (en ajoutant une double barre oblique en tête) et recharger le croquis, ceci afin d'éviter que l'horloge ne revienne à ses paramètres initiaux à chaque réinitialisation. Pour obtenir une précision suffisante, il suffit de mesurer le temps qu'il faut pour télécharger le code, disons 30 secondes, puis de lancer le premier téléchargement 30 secondes à l'avance. Après quelques essais, une synchronisation à la seconde près peut être obtenue gratuitement ! L'heure de rafraîchissement quotidien est également effectuée à partir du code, en entrant simplement l'heure souhaitée.

Enfin, une note sur la méthode utilisée pour mesurer la tension de la batterie. On la trouve souvent sur Internet, avec quelques variantes, et Microchip elle-même la documente dans sa propre note [7]. L'astuce consiste à définir, via les registres appropriés, la tension de référence interne (1,1 V) comme valeur d'entrée de l'ADC et la tension à mesurer (VCC) comme référence. Tout changement de V_{CC} modifiera la lecture de l'ADC, ce qui permettra de calculer la valeur de la tension avec suffisamment de précision.

Conclusion

Bien qu'offrant des caractéristiques d'un intérêt indéniable, qui la rendent particulièrement adaptée au projet présenté, la technologie e-papier présente également certaines limites, dont la plus évidente est la faible fréquence de rafraîchissement. Le modèle utilisé ici effectue le cycle de rafraîchissement complet en 2 secondes et le rafraîchissement partiel en 0,3 seconde. Ces valeurs ne peuvent donc pas rivaliser avec d'autres types d'affichage dans la visualisation d'images, de graphiques et de textes qui changent rapidement. Enfin, comme je le dis souvent, au-delà de l'utilité réelle de l'objet proposé, j'espère que vous avez trouvé quelques idées intéressantes à méditer et à réutiliser avec l'esprit d'un faiseur, et que cet article a éveillé votre curiosité et une envie d'expérimenter avec les afficheurs e-papier !

VF : Helmut Müller — 230012-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (a.dellapia@elettronicaemaker.it) ou contactez (redaction@elektor.fr).

À propos de l'auteur

Depuis son enfance, Antonello Della Pia est attiré par l'électricité et les appareils électroniques. Il est titulaire d'un diplôme d'études supérieures de « Technicien en génie électrique ». Antonello a toujours cultivé et développé sa passion pour l'électronique analogique et numérique. Actuellement, il s'occupe avec les microcontrôleurs et la programmation, en essayant d'améliorer ses compétences en informatique. Antonello aime développer et proposer des projets aussi originaux que possible et, comme il l'espère, aussi intéressants.

Produits

- **Waveshare Module d'affichage E-Ink/E-Paper de 2,9 pouces (SKU 18776)**
  www.elektor.fr/18776
- **Ynvisible Segment kit d'afficheur e-papier (SKU 20143)**
  www.elektor.fr/20143

LIENS

[1] Papier électronique (Wikipédia) : https://fr.wikipedia.org/wiki/Papier_%C3%A9lectronique
[2] Fiche technique de l'ATmega328P : https://www.elektormagazine.com/atmega328p-datasheet
[3] Fiche technique du DS3231M : https://datasheets.maximintegrated.com/en/ds/DS3231M.pdf
[4] Waveshare - 296x128, module d'affichage e-Ink de 2,9 pouces : https://www.elektormagazine.com/waveshareeink
[5] Calculateur de durée de la batterie en ligne : https://www.omnicalculator.com/other/battery-life
[6] BitmapToByteArrayConverter : https://www.briandorey.com/post/bitmap-byte-converter-for-e-ink-display
[7] Mesurer la tension VCC/Batterie sans utiliser les broches d'E/S, Microchip : https://www.elektormagazine.com/vcc/battery
[8] Principe de fonctionnement de base de l'afficheur e-papier : https://www.elektormagazine.com/bwcapsules
[9] Code source : http://www.elektormagazine.fr/230012-04