sablier motorisé en BASIC

avec ESP8266 et Annex WiFi RDS

Peter Neufeld (Allemagne)

Dans le dernier numéro d'Elektor, l'Annex WiFi RDS a été décrite comme une plateforme de programmation en BASIC des microcontrôleurs ESP8266/ESP32. En voici un exemple pratique: le micrologiciel pour un sablier. Pas en C, ni en C#, ni en C++, mais rapidement, simplement et facilement en BASIC.

Figure 1. Le sablier terminé est doté d'une horloge numérique à quatre chiffres sur la gauche et un véritable sablier illuminé est tourné par un servo caché sur la droite.

Voici un projet pratique comme application de l'environnement de développement en BASIC appelé Annex WiFi RDS [2] décrit dans le dernier numéro d'Elektor [1]. Les différents modèles de modules ESP8266, disponibles pour quelques euros, offrent à l'électronicien des possibilités infinies pour des projets de plus en plus ambitieux, non seulement grâce à leur fonction WLAN, mais aussi parce qu'ils peuvent être facilement connectés sans fil à des téléphones tactiles, des tablettes et des PC. Divers capteurs et actionneurs peuvent être facilement connectés au microcontrôleur sans grands frais, de sorte que le bricolage pourrait redevenir un véritable plaisir si seulement il n'y avait pas le fichu obstacle de la programmation plutôt compliquée des SoC modernes.

Le choix est pourtant vaste entre langages de programmation, IDE, bibliothèques, pilotes, ressources et compilateurs ! Ou faudrait-il peut-être contourner l'obstacle en douce et récupérer du code écrit par d'autres ? Celui-ci est souvent difficile à comprendre et le transfert dans le module ne va pas de soi. Et si vous souhaitez commander votre projet avec un téléphone par l'intermédiaire d'une interface web, ça se complique. Sauf si adoptez l'environnement de développement Annex WiFi RDS programmé en BASIC. Vous devriez essayer, la réussite est rapide, avec zéro frustration.

Projet simple

Quelques lignes de BASIC suffisent pour réaliser une horloge numérique avec ESP8266, qui affiche en même temps les prévisions météorologiques. Jugez-en vous-même par les exemples de projets sur le site de l'Annex [3]. J'y ai trouvé l'idée de combiner un affichage numérique et un affichage analogique : le projet *ESP8266 Sand Clock*, renverse un petit sablier toutes les minutes tout en donnant l'heure sous un format numérique conventionnel à quatre chiffres.

Ce n'est pas tout : le sable qui coule est éclairé par quatre LED NeoPixel pour la lumière de fond. La luminosité de l'afficheur doit être adaptée à la luminosité ambiante, grâce à une photorésistance (LDR) utilisée comme capteur de lumière. Si vous ajoutez un capteur de température à 1 fil, vous pouvez même afficher la température ambiante.

Ce sablier devra pouvoir être commandé par un réseau local sans fil (WLAN) grâce à l'interface web intégrée. Imaginez-vous le code qu'il faut pour ça ! Et la ribambelle de bogues qu'il faudra débusquer ? Le mot qui vient à l'esprit quand songe à tout ça commence et finit par *mañana...*

Heureusement il y a le BASIC ! Voyez donc le sablier fini de la **fig. 1** : il ne nécessite pas tant de logiciel qu'on pourrait le penser.

En parcourant le fichier d'aide de l'Annex WiFi RDS, détaillé et orienté vers la pratique, de même que le contenu du site associé, d'autres idées de projets inédits ont surgi. Le BASIC est à la fois puissant facile à maîtriser, même pour des programmeurs occasionnels. Comparé aux autres solutions courantes, l'*Annex Rapid Development Suite* simplifie considérablement le développement de logiciels.

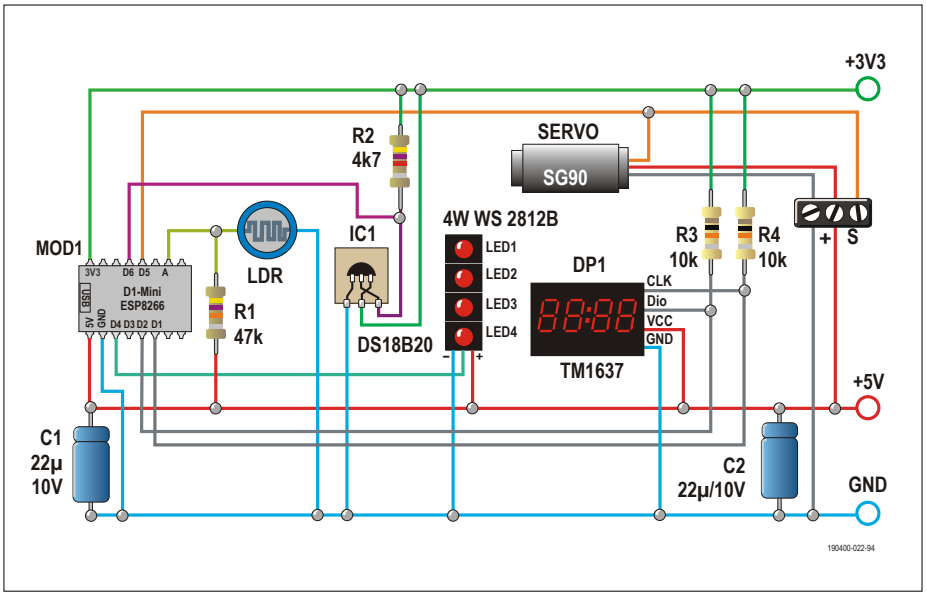

Figure 2. Le circuit du sablier n'est pas plus compliqué que ça : en plus du module ESP8266, quatre résistances, deux condensateurs, quatre LED, un capteur de température, un écran et un servo, c'est tout !

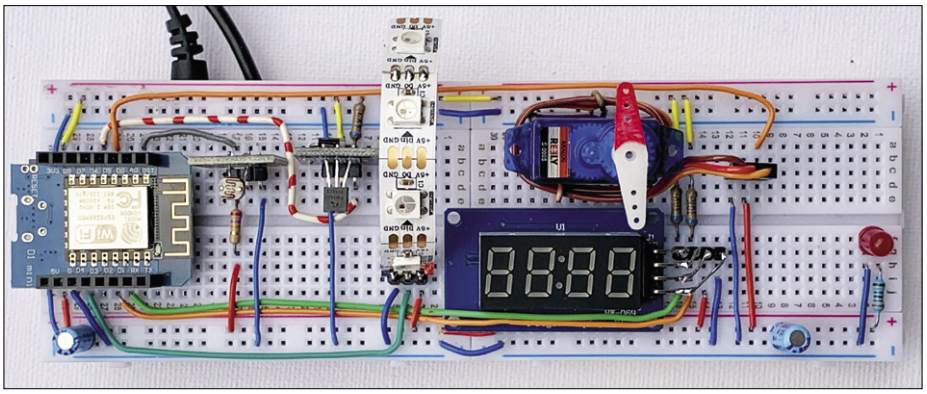

Figure 3. Le prototype du sablier sur une plaque d'expérimentation le circuit de la figure 2.

L'électronique

La carte à µC D1-Mini utilisée offre un module ESP8266 et, grâce à ses broches, convient à l'expérimentation sur une plaque à trous, car toutes les connexions SoC pertinentes sont accessibles. Grâce à son interface USB et à son mode auto-flash, le micrologiciel peut être chargé très facilement et rapidement, sans intervention matérielle, à l'aide de l'Annex Toolkit [4]. N'oubliez pas de définir le fuseau horaire correct sur la page CONFIG et d'y saisir le script BASIC dans la ligne de démarrage automatique (le chemin de mon script actif est toujours «*/program/default.bas*»).

Pour mes tests, j'ai assemblé le circuit de la **fig. 2** sur une plaque d'expérimentation (**fig. 3**).

Si la tension de service est fournie par la prise micro-USB, Les tensions de +5 V et +3,3 V sont également accessibles de l'extérieur et peuvent être utilisées dans certaines limites du fait de la présence d'une diode Schottky et d'un régulateur de tension de 3,3 V. Le courant consommé par les périphériques utilisés ici ne dépasse pas ce que l'on peut attendre de ces lignes d'alimentation. Si vous y tenez, vous pouvez alimenter le servomoteur séparément sous +5 V. Les deux condensateurs électrolytiques entre +5 V et la masse lissent les pics de courant du servo.

Notez que les entrées d'un ESP8266 ne tolèrent pas de niveaux de plus 3,3 V. C'est pourquoi presque toutes les résistances de polarisation externes sont connectées à +3,3 V. Grâce au diviseur de tension interne de l'entrée analogique, toute tension de 0 à 3,3 V est acceptée et même l'application de 5 V ne pose pas de problème. Le diviseur de tension de la photorésistance doit être dimensionné de manière à ce que, en pleine lumière, il règne au maximum 500 mV à l'entrée. Si l'entrée analogique reste déconnectée, la tension sera nulle ici et la luminosité de l'écran sera à son maximum.

Attention lors du branchement du capteur de température DS18B20 ! Le brochage du capteur sur la carte est différent de celui du capteur en boîtier TO.

Logiciel BASIC

Une fois le micrologiciel installé sur l'ESP8266 en utilisant le Toolkit d'Annex [4] et une fois les instructions [5] suivies, il suffit d'un navigateur pour accéder au serveur du module. Le nouveau code peut être rapidement écrit dans l'éditeur ou inséré dans *default.bas* par copier-coller.

Le fichier d'aide d'Annex (touche F2 dans l'éditeur) contient des explications, les brochages et des bribes de script pour l'afficheur à LED TM1637, le servo analogique, les LED NeoPixel et divers actionneurs et capteurs disponibles dans le commerce ainsi que pour les différents modules ESP. L'aide en ligne contient également une section utile sur les bases du langage.

Le **listage 1** présente le début du script BASIC utilisé. Le **listage 2** contient le

Listage 1. Début de hourglass.bas

```
'#################### HOURGLASS ########################
' 4Digit digital clock combined with 10 minutes hourglass
  VERSION$      = "V3.3"
' 10/2019  Peter.Neufeld@gmx.de   min:ANNEX_1.39b6
' - ESP8266-Module: WEMOS-D1-Mini
' - Display: 4digit 7segment TM1637
' - Servo: analog servo (SG90)
' - Background light: 4 NeoPixel-LEDs
' - Temperature sensor: DS18B20
' - Light sensor: LDR @ analogue input
' GPIO-Mapping (PIN-Label) for WEMOS and NodeMCU boards
D0=16:D1= 5:D2= 4:D3= 0:D4=2
D5=14:D6=12:D7=13:D8=15:D9=3:D10=1
SERVO_PIN   = D5  'Servo @ GPIO14=D5
TM_DATA     = D2  'TM1637 Data @ GPIO4=D2
TM_CLOCK    = D1  'TM1637 Clock @ GPIO5=D1
TM_BRIGHT   = 7   'TM1637 brightness
TURN_TIME   = 10  'Turn hourglass every xx minutes
SERVO_LEFT  = 180 '1. Position of hourglass
SERVO_RIGHT = 0   '2. Position of hourglass
SERVO_POS   = 0   'actual position of hourglass
BLINK       = 0   '255= visible colon @ TM1637
ADC_DARK    = 450 'max. ADC value for dark LDR,
'            depends on LDR and Pull-up
LED1_STATUS = 0 'Background LEDs: 0=automatic, 1=manually
STATUS$     = "Modus: AUTOMATIK"
ADC_IN = 0:
```

Listage 2. Sous-programme pour une page web.

```
WEB_PAGE:
cls
a$ = "<center><h1> - S A B L I E R - "+ VERSION$ +" - </h1>"
a$ = a$ + "<br>" + textbox$(t$,"cssTB")
a$ = a$ + METER$(SS,0,60,"cssMET")
a$ = a$ + textbox$(TEMP$,"cssTB")+"<br><br>"
a$ = a$ + "<br>R: "+ slider$(R, 0,255)+ textbox$(R,"cssTB")
a$ = a$ + "<br>G: "+ slider$(G, 0,255)+ textbox$(G,"cssTB")
a$ = a$ + "<br>B: "+ slider$(B, 0,255)+ textbox$(B,"cssTB")
a$ = a$ + "<br>"+ LED$(LED1_STATUS)
a$ = a$ + "<br>"+ textbox$(STATUS$)
a$ = a$ + "<br>"+ BUTTON$("Mode auto/manuel",MAN_AUTO)
a$ = a$ + cssid$("cssTB"," width:70px;text-align:center")
a$ = a$ + cssid$("cssMET"," transform:rotate(-90deg);")
html a$
return
```

code de l'interface web délibérément simple (**fig. 4**).
Ce projet a rapidement compté environ 140 lignes BASIC. Comme il est amplement commenté, vous comprendrez un tel script même si vous ne l'avez pas écrit vous-même. Si vous le souhaitez, vous le prolongerez ou l'améliorerez dans quelque temps. Le code complet de cet article peut être téléchargé sur le site d'Elektor [6].

Au cours des manipulations du fichier *default.bas*, édition, sauvegarde (!), lancement, débogage, modification, extension, sauvegarde, nouveau lancement, etc., les erreurs courantes du code sont identifiées et marquées par l'interpréteur. Cela en facilite la correction. Le code BASIC est à la portée des programmeurs occasionnels, surtout si on prend soin d'attribuer aux variables des noms qui parlent d'eux-mêmes et si on a recours à des sous-programmes. Le code BASIC est facile à lire. Le codage graphique de la syntaxe améliore la lisibilité du code et en facilite la vérification. En cas de problème de syntaxe, la touche F2 permet d'effectuer une recherche contextuelle dans le fichier d'aide en ligne. Une petite coche verte dans la fenêtre de l'éditeur mène aussi au contrôle syntaxique.
Pour commencer, le scénario BASIC est assez linéaire. Après initialisation de certaines variables et un court mouvement du servo du sablier, l'adresse IP du serveur Annex est affichée par portions. C'est commode lorsqu'on accède au serveur avec un navigateur, surtout si le routeur a attribué une adresse IP différente.
La routine principale est exécutée une fois par seconde au moyen d'un temporisateur. Heures et minutes sont préparées pour l'affichage, puis sont exécutés successivement les sous-programmes de commande de lumière et du rétroéclairage, l'affichage de l'heure et de la température, et enfin, toutes les dix minutes, la servocommande.

J'ai abandonné l'idée initiale de tourner le sablier une fois par minute à cause du bruit que fait un servo de modèle réduit dans un salon silencieux. J'ai opté pour un sablier de dix minutes, qui tourne le verre lentement pendant environ quatre secondes et donc plus silencieusement et moins fréquemment. Si vous voulez, vous pouvez utiliser un sablier plus rapide pour

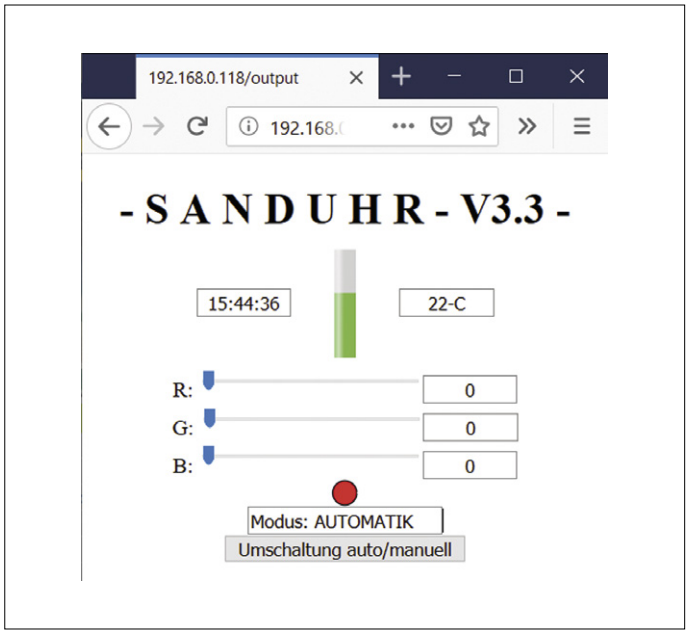

Figure 4. Un peu de code BASIC suffit pour créer une interface web simple.

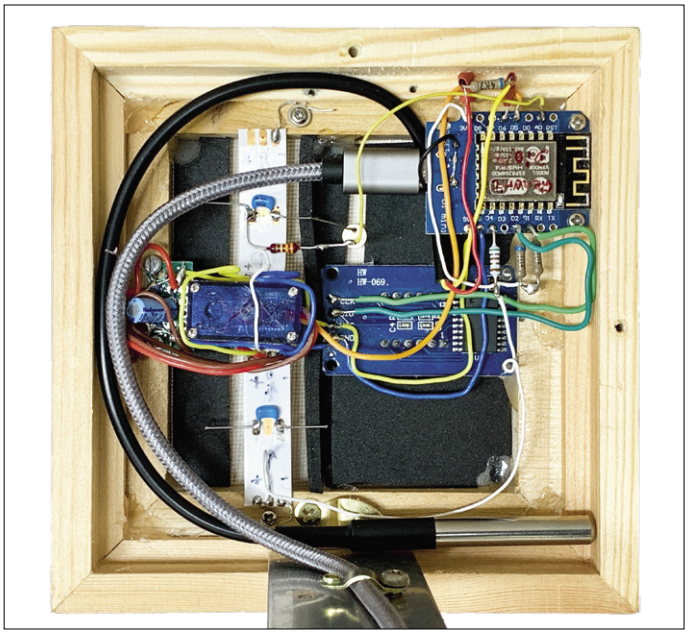

Figure 5. Les coulisses de mon sablier électronique ou les joies du bricolage en montage volant !

obtenir par exemple un rythme d'une ou cinq minutes. Il suffit de modifier la variable `TURN_TIME`.

Interface web

Lorsqu'un navigateur accède pour la première fois ou à nouveau, ou que les valeurs des variables utilisées changent, le contenu HTML de la page web ESP8266 est préparé et mis à jour dans le sous-programme `WEB_PAGE` (listage 2). L'interface affiche l'heure et la température actuelles. La simple animation d'arrière-plan à LED derrière le sablier est remplacée par une sélection manuelle de couleurs RVB par bouton-poussoir. En mode automatique, les contrôleurs RGB et les champs d'affichage suivent en direct les valeurs de l'animation calculées en continu.

Le site vise uniquement à montrer comment le micrologiciel d'Annex assure la communication bidirectionnelle avec le navigateur en arrière-plan. Grâce à l'utilisation par Annex de CSS (Javascript, AJAX et les sockets web sont également pris en charge), il n'y a pratiquement pas de limites aux expériences de communication avec (presque) tous les navigateurs. On en trouve également de nombreux exemples sur le site web d'Annex - notamment dans les fichiers qui peuvent être copiés du Toolkit d'Annex dans le dossier «*/program*» du module ESP lors de l'installation du micrologiciel. Cela vaut la peine de fouiller ! Si vous trouvez un script intéressant, vous pouvez l'insérer grâce à la commande Ctrl+A (= tout sélectionner) suivie d'un copié-collé dans la fenêtre de l'éditeur d'Annex avec un éditeur de texte (p. ex. *GEANY.exe)*, puis le sauvegarder et le lancer rapidement dans *default.bas*) ! Des sections complètes du script peuvent être rapidement converties en lignes de commentaires à partir du menu et vice versa.

Emménager

Mon circuit dûment a finalement emménagé en montage volant dans un petit cadre en bois, recouvert à l'avant d'un écran translucide (**fig. 5**). Seuls les segments actifs et lumineux de l'affichage à LED sont visibles. Le capteur de lumière (LDR) reçoit suffisamment de lumière ambiante. L'axe du servo qui fait tourner le sablier passe par un petit trou dans l'écran. Des plaques de mousse ou du papier noir découpé à la bonne taille permettent de limiter la transparence de l'écran aux seules zones nécessaires, derrière le sablier et sur la LDR.

On peut également utiliser d'autres types de cadres avec façade en plexiglas (comme on en trouve au catalogue d'un célèbre magasin de meubles en kit). La face transparente sera recouverte à l'intérieur par du papier translucide ou une photo adéquate. À vous de bricoler ! ⏮

190400-B-02

Liens

[1] BASIC pour ESP32/ESP8266 : www.elektormagazine.fr/190400-02
[2] Annex WiFi RDS : https://sites.google.com/site/annexwifi/home
[3] Exemples de projets : https://sites.google.com/site/annexwifi/projects
[4] Toolkit : https://sites.google.com/site/annexwifi/downloads
[5] Instructions : https://sites.google.com/site/annexwifi/home/first-steps
[6] Page du présent article : www.elektormagazine.fr/190400-B-02