BASIC voor ESP32/ESP8266 (2)

zandloper met ESP8266 en Annex WiFi RDS

Peter Neufeld (Duitsland)

In het vorige nummer van Elektor beschreven we Annex WiFi RDS als een platform voor het programmeren van ESP8266/ ESP32-controllers in BASIC. Nu is het tijd voor een praktische toepassing: de firmware voor een zandloper. We doen dat niet in C, C# of C++, maar snel, simpel en eenvoudig in BASIC.

Figuur 1. De zandloper heeft links een viercijferige digitale klok en rechts een echte, verlichte zandloper die precies op tijd wordt gedraaid door een verborgen servo.

Dit is een praktisch en voor een eerste kennismaking geschikt project voor de BASIC-ontwikkelomgeving Annex WiFi RDS [2] die we beschreven in Elektor maart/april 2020 [1]. De ESP8266-modules, die in verschillende uitvoeringen verkrijgbaar zijn voor slechts enkele euro's, bieden de elektronicus eindeloze mogelijkheden, niet in het minst dankzij de ondersteuning van WLAN, waarmee projecten gemakkelijk 'draadloos' kunnen worden verbonden met smartphones, tablets en PC's. Omdat er ook verschillende sensoren en actuatoren bestaan die voor weinig geld eenvoudig op de microcontroller kunnen worden aangesloten, zouden we weer heerlijk kunnen gaan knutselen… Als het programmeren van deze moderne SoC's niet zo vreselijk ingewikkeld was.

We moeten kiezen tussen verschillende programmeertalen, diverse IDE's, bibliotheken, drivers, resources en compilers. Of moeten we 'valsspelen' en een stuk moeilijk te begrijpen code downloaden en dat op de een of andere manier overbrengen naar de module? Het wordt echt ingewikkeld als u uw project via een webinterface met een smartphone wilt besturen. De ontwikkelomgeving Annex WiFi RDS met de programmeertaal BASIC maakt zo'n project veel eenvoudiger. Daarmee kunt u snel tot een bevredigend resultaat komen.

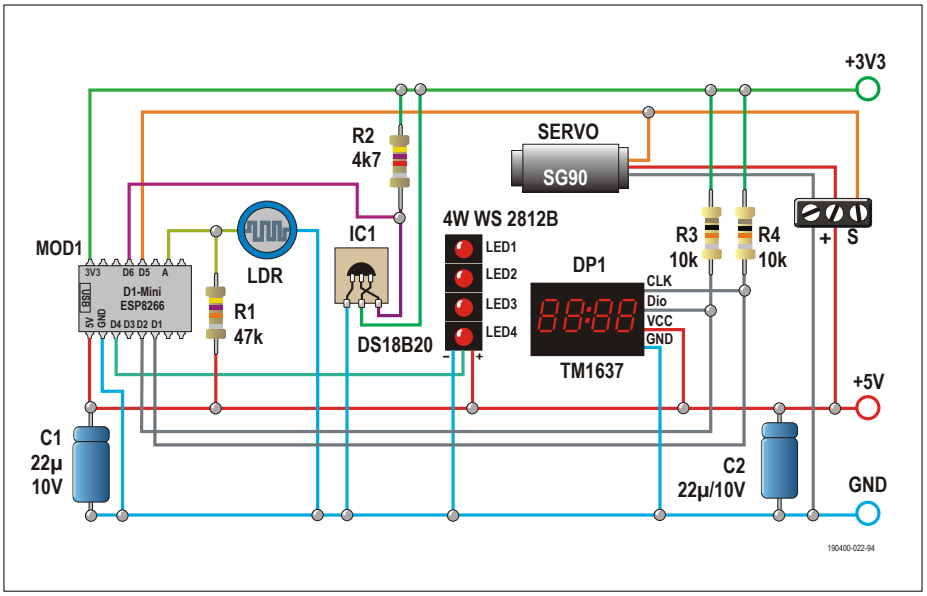
Een eenvoudig project

Met een paar regels BASIC kunt u met de ESP8266 een digitale klok realiseren, die ook nog eens het weerbericht toont. De voorbeeldprojecten op de Annex-website [3] bewijzen dat. Ik had een idee om er een bijzonder chique uitstraling aan te geven: de klok zou de tijd digitaal én analoog weergeven. Bij het project 'ESP8266-zandloper' bouwen we een conventionele, viercijferige digitale klok met daarnaast een kleine zandloper die elke minuut wordt omgedraaid.

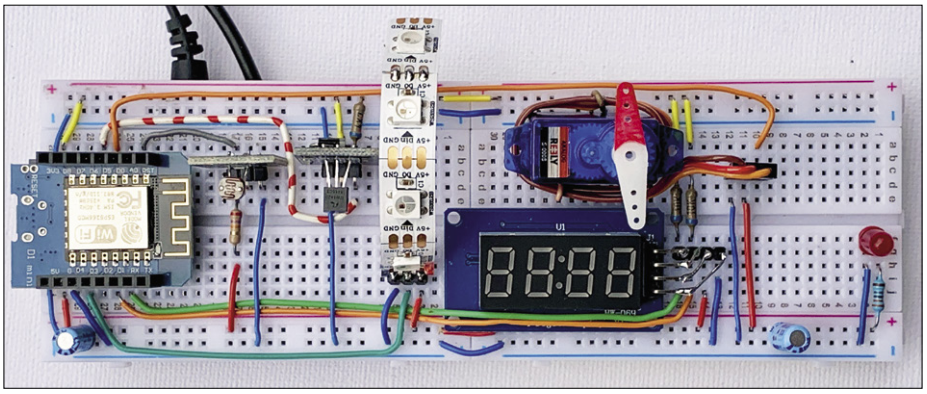
En dat is nog niet alles: we kunnen het bewegende zand ook nog verlichten: vier NeoPixel-LED's dienen als achtergrondverlichting. Natuurlijk moet de helderheid van het display worden aangepast aan het omgevingslicht, dus voegen we een LDR toe als lichtsensor. Als u een 1-wire-temperatuursensor installeert, kunt u zelfs de kamertemperatuur op het display weergeven.

Natuurlijk moeten we de zandloper ook via WLAN kunnen bedienen met behulp van de geïntegreerde webinterface. Als we dat allemaal willen realiseren, zal er wel zoveel code voor nodig zijn, met

allerlei potentiële bugs – genoeg om u de moed in de schoenen te doen zinken ;-) Maar omdat BASIC het in praktijk veel eenvoudiger maakt, was voor de zandloper in **figuur 1** lang niet zoveel software nodig als u wellicht vreest.
Bij het bladeren door het uitgebreide, praktijkgerichte helpbestand van Annex WiFi RDS en de bijbehorende website, kreeg ik allerlei ideeën voor nieuwe projecten. Ondanks zijn kracht is BASIC eenvoudig te gebruiken, zelfs voor mensen die niet vaak programmeren. Dankzij de Annex **R**apid **D**evelopment **S**uite is het schrijven van software een stuk gemakkelijker dan met andere ontwikkelomgevingen.



Figuur 2. Het schema van de elektronische zandloper is heel eenvoudig: naast de ESP8266-module zijn er vier weerstanden, twee condensatoren, vier LED's, een temperatuursensor, een display en een servo nodig. Een onderdelenlijst is overbodig.



Figuur 3. De testopstelling van de zandloper volgens het schema van figuur 2 op breadboard.

De elektronica

Het hier toegepaste D1-Mini-controllerboard bevat een ESP8266-module; de plaatsing van de pennen maakt het geschikt voor breadboard-experimenten. Alle relevante aansluitingen van de SoC zijn naar buiten gevoerd. Via de USB-interface en de autoflash-mogelijkheid kan de module heel gemakkelijk en snel, zonder extra hardware, met firmware worden geladen met de Annex Toolkit [4]. Vergeet niet de juiste tijdzone in te stellen op de CONFIG-pagina en voer daar ook het BASIC-script in op de autostart-regel (het pad van mijn actieve script is altijd *"/program/default.bas"*).
Ik heb voor de tests de schakeling van **figuur 2** opgebouwd op breadboard. Het resultaat is te bewonderen in **figuur 3**.
Als de voedingsspanning wordt aangesloten via de micro-USB-bus, worden ook +5 V (via een Schottky-diode) en +3,3 V (via een spanningsregelaar) naar buiten gevoerd. We kunnen deze spanningen (binnen zekere grenzen) gebruiken om externe hardware te voeden. De stromen die de hier toegepaste periferie gebruikt kan de module gemakkelijk leveren. Als u helemaal op safe wilt spelen, kunt u de servo apart voeden met +5 V. De twee condensatoren tussen +5 V en massa bufferen de stroompieken van de servo.
Houd er rekening mee dat de ingangen van een ESP8266 maximaal 3,3 V kunnen verdragen. Daarom zijn bijna alle externe pullup-weerstanden aangesloten op +3V3. De interne spanningsdeler van de analoge ingang zorgt ervoor dat spanningen van 0...3,3 V kunnen worden gemeten en dat er zelfs geen problemen ontstaan als hier 5 V wordt aangesloten. De spanningsdeler met de LDR moet zodanig worden gedimensioneerd, dat bij maximale verlichting niet meer dan 500 mV op de ingang komt te staan. Als de analoge ingang niet wordt aangesloten, staat hier 0 V en wordt het display dus ingesteld op volle lichtsterkte.
Let op bij het aansluiten van de DS18B20-temperatuursensor! Het type op een moduleprintje heeft een andere pinbezetting dan de sensor in de TO-behuizing.

BASIC-software

Als u met behulp van de Annex Toolkit [4] de firmware hebt geïnstalleerd op de ESP8266 en de instructies in [5] heeft opgevolgd, kunt u met een webbrowser de webserver van de module benaderen. Nieuwe code kan snel worden geschreven in de editor of met copy & paste in *default.bas* worden geplakt.
Het Annex helpbestand (F2-toets in de editor) bevat uitleg, pintoewijzingen en scriptfragmenten voor het TM1637 LED-display, de analoge servo, de NeoPixel-LED's en diverse in de handel verkrijgbare actuatoren en sensoren, alsmede voor verschillende ESP-modules. De online-help bevat ook een nuttig hoofdstuk over de grondbeginselen van de programmeertaal.
In **listing 1** ziet u het begin van het gebruikte BASIC-script. **Listing 2** bevat de code van de opzettelijk eenvoudige web-interface die te zien is in **figuur 4**.
De software voor dit project bestaat uit ongeveer 140 regels BASIC. Omdat het script van voldoende commentaar is voorzien, is het goed te begrijpen, ook als u het niet zelf hebt geschreven of als u het na een paar weken wilt uitbreiden

Listing 1. Begin van “sanduhr.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:
```

of verbeteren. De volledige code kan worden gedownload van de projectpagina bij dit artikel [6].
Het werken aan de software betekent telkens weer het openen van het bestand *default.bas*, bewerken, opslaan (!), starten, debuggen, uitbreiden, opslaan, starten – en zo vervolgens. Daarbij worden fouten in de code door de interpreter gesignaleerd en gemarkeerd.

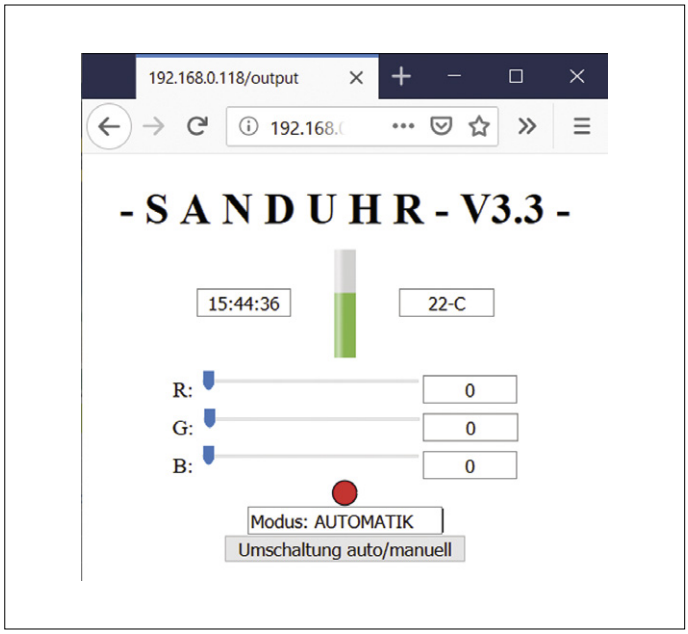
Listing 2. Subroutine “WEB_PAGE”.

```
WEB_PAGE:
cls
a$ = "<center><h1> - S A N D U H 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$("Umschaltung auto/manuell",MAN_AUTO)
a$ = a$ + cssid$("cssTB"," width:70px;text-align:center")
a$ = a$ + cssid$("cssMET"," transform:rotate(-90deg);")
html a$
return
```

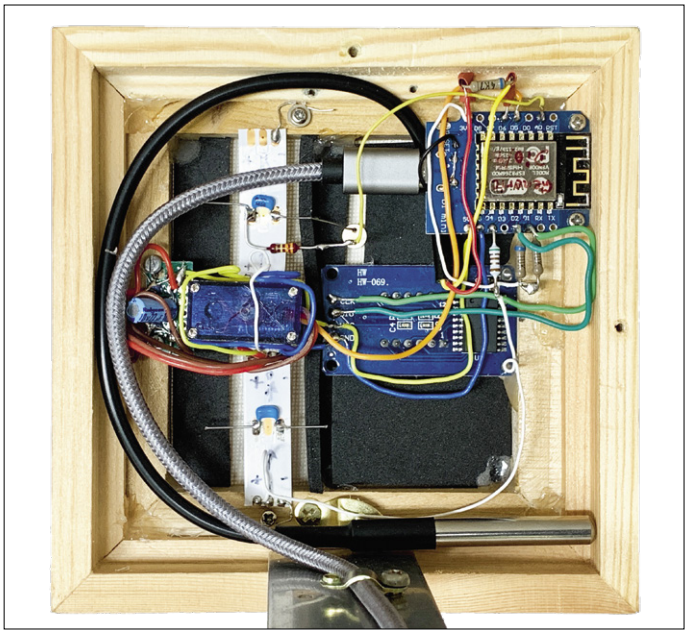
Meestal zijn ze dan snel te verhelpen. De BASIC-code is gemakkelijk te begrijpen, zelfs voor mensen die niet dagelijks programmeren, als de namen van variabelen en subroutines duidelijk maken waar ze voor dienen. BASIC is gemakkelijk te lezen. De automatische *code highlighting* verbetert de leesbaarheid en vergemakkelijkt de controle van de syntax. Als u niet weet hoe u een statement precies moet gebruiken, helpt de F2-toets bij het contextgevoelig zoeken in het online-helpbestand. Er is ook een klein groen vinkje in het bewerkingsvenster, waarmee u de syntaxcontrole kunt starten.
Het BASIC-script is vrij overzichtelijk. Na het initialiseren van enkele variabelen en een korte beweging van de zandloperservo wordt het IP-adres van de ANNEX-webserver ‘beetje bij beetje’ op het display gezet. Dat helpt enorm bij het benaderen van de site met een webbrowser, vooral als de router een ander IP-adres heeft toegewezen.
De hoofdroutine wordt één keer per seconde verwerkt door middel van een timer. De weergave van de uren en minuten wordt voorbereid, en de subroutines voor de lichtsensor, achtergrondverlichting, de weergave van tijd en temperatuur worden achtereenvolgens aangeroepen. Om de tien minuten wordt ook de servobesturing aangeroepen.
Ik ben afgestapt van het oorspronkelijke plan om de zandloper één keer per minuut te laten draaien, toen ik merkte hoeveel lawaai een modelbouwservo maakt als hij de zandloper omdraait in de woonkamer. Daarom heb ik besloten een zandloper van tien minuten te gebruiken, en het glas langzaam om te draaien in ongeveer vier seconden. Dat maakt minder herrie en gebeurt minder vaak. Als u dat leuk vindt, kunt u ook een ritme van één of vijf minuten realiseren met een snellere zandloper en de variabele `TURN_TIME`.

Webinterface

Als een webbrowser voor het eerst of opnieuw verbinding maakt, of wanneer de waarden van de variabelen veranderen, wordt de HTML-inhoud van de ESP8266-webpagina voorbereid en bijgewerkt in de subroutine `WEB_PAGE` (zie listing 2). De webinterface toont de huidige tijd en temperatuur. De eenvoudige LED-animatie achter de zandloper kan via een druktoets worden vervangen door handmatige RGB-kleurenselectie.



Figuur 4. Een eenvoudige web-interface is te realiseren met een klein beetje BASIC-code.



Figuur 5. Zo ziet de elektronische zandloper er van achteren uit: niet mooi, maar de zwevende bedrading werkt goed.

In de automatische modus worden de RGB-bedienings- en weergavevelden *live* berekend en geanimeerd.

De website is alleen bedoeld om te laten zien hoe de Annex-firmware op de achtergrond zorgt voor de bidirectionele communicatie met de webbrowser. Annex ondersteunt het gebruik van CSS (en ook Javascript, AJAX en websockets) dus er zijn vrijwel geen grenzen aan de mogelijkheden om te experimenteren met de communicatie met (vrijwel) elke webbrowser. Hier zijn ook veel voorbeelden van te vinden op de Annex-website, vooral in de bestanden die tijdens de firmware-installatie van de Annex-toolkit naar de directory *"/program"* op de ESP-module kunnen worden gekopieerd. Het is de moeite waard om dat nader te bekijken! U kunt een interessant script gemakkelijk met een teksteditor (bijv. *GEANY.exe*) kopiëren & plakken naar *default.bas* in het venster van de Annex-editor om het te testen! Hele stukken van het script kunnen ook snel worden uitgecommentarieerd vanuit het menu en weer worden teruggezet.

Inbouw in de definitieve behuizing

De geteste schakeling is uiteindelijk zonder print of breadboard geïnstalleerd in een klein houten frame, dat aan de voorkant is afgedekt met doorschijnend doek (**figuur 5**). Dat betekent dat alleen de actieve, verlichte LED-displaysegmenten zichtbaar zijn en dat de LDR-lichtsensor ook voldoende omgevingslicht kan absorberen. De servo-as voor de zandloper werd door een klein gaatje in het doek geleid. Met op maat gesneden stukken schuimrubber of zwart papier wordt de lichtdoorlatendheid van het frontmateriaal beperkt waar het nodig is: alleen voor het display, achter de zandloper en voor de LDR maken we openingen in de afdekking.

Alternatief kunnen ook andere frames met een plexiglas front worden gebruikt (bijvoorbeeld uit het assortiment van een groot woonwarenhuis). De transparante voorzijde kan dan aan de binnenkant worden voorzien van doorschijnend papier of een fotoprint. ⏮

190400-B-03

Weblinks

[1] BASIC voor ESP32/ESP8266: www.elektormagazine.nl/190400-04
[2] Annex WiFi RDS: https://sites.google.com/site/annexwifi/home
[3] Voorbeeldprojecten: https://sites.google.com/site/annexwifi/projects
[4] Toolkit: https://sites.google.com/site/annexwifi/downloads
[5] Handleiding: https://sites.google.com/site/annexwifi/home/first-steps
[6] Projectpagina bij dit artikel: www.elektormagazine.nl/190400-B-03