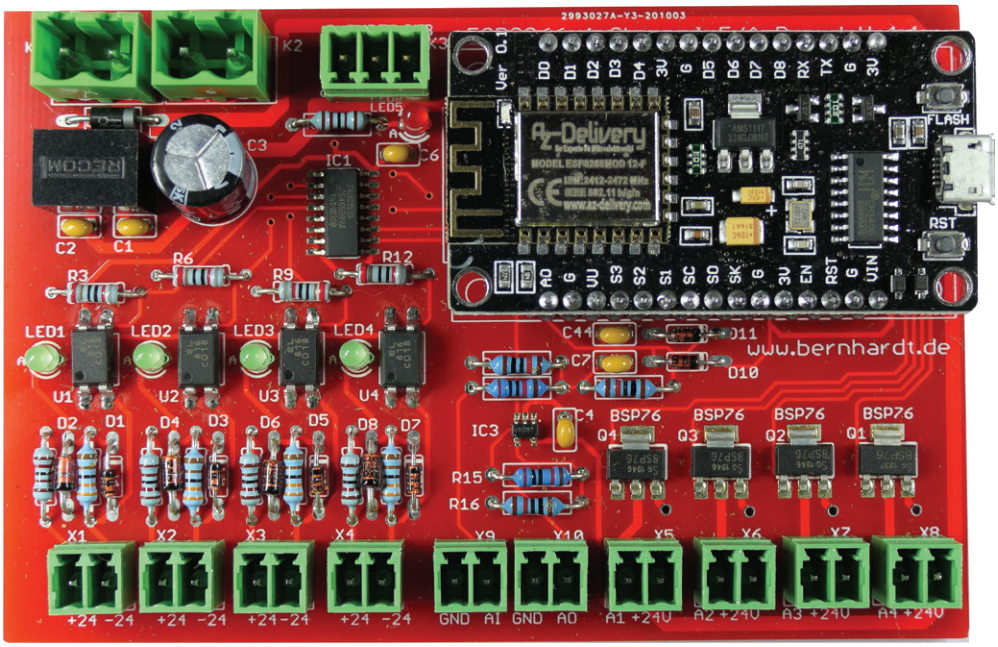
Modbus sans fil (partie 1)

Matériel et programmation

Josef Bernhardt (Allemagne) et
Martin Mohr (Allemagne)



Carte Modbus avec module NodeMCU.

Le protocole Modbus est très utilisé dans l'industrie pour la communication entre systèmes et contrôleurs. En pratique, il met généralement en œuvre des interfaces filaires RS485, éprouvées et fiables. Nous présentons ici un module qui permet d'utiliser le protocole Modbus sur un réseau local sans fil. Le module est construit autour d'une carte Espressif NodeMCU à microcontrôleur ESP8266. Une carte de base Modbus complémentaire permet de travailler avec des signaux de 24 V très courants en environnement industriel. Pour illustrer le fonctionnement, les auteurs ont réalisé la commande d'un jouet, une porte d'ascenseur.

Pour la plupart des lecteurs d'*Elektor*, le module *NodeMCU* d'*Espressif* et l'*EDI Arduino* sont déjà familiers. Si c'est votre cas, vous pouvez sauter l'introduction et passer à la description de la carte *Modbus TCP*. Sinon, voici ce que vous devez savoir en quelques mots.

Ce projet est conçu autour d'un module *NodeMCU* (disponible dans la boutique Elektor). Ce module est équipé d'un microcontrôleur *Espressif ESP8266*, de la taille d'un timbre-poste, il est doté d'une interface réseau local sans fil (*WLAN* ou *Wireless LAN* en anglais). Malgré sa petitesse, sa puissance de traitement est grande. Le **tableau 1** résume les principales caractéristiques du microcontrôleur ESP8266. La carte NodeMCU produit la tension d'alimentation de l'ESP8266 et gère l'interface de programmation du microcontrôleur. La **figure 1** donne le brochage de la carte NodeMCU utilisée dans notre circuit Modbus.

L'EDI Arduino est très bien adapté à la programmation de la carte NodeMCU. Le site web d'Arduino [2] permet de télécharger gratuitement la version de l'EDI Arduino

adaptée au système d'exploitation de votre ordinateur et de l'installer en quelques instructions. Si vous exécutez l'EDI pour la 1re fois, vous voyez une fenêtre comme celle de la **figure 2**. Le volet du code du programme contient deux fonctions prédéfinies : la fonction `setup()` qui est exécutée une seule fois au démarrage du programme. Elle gère – entre autres – l'initialisation des interfaces du microcontrôleur. Puis vient la fonction `loop()` qui accueille normalement le code source du programme.

La fonction `loop()` s'exécute à la suite de la fonction `setup()`. Si le programme atteint la fin de la fonction `loop()`, il recommence depuis le début. L'ESP8266 gère l'interface WLAN **entre la fin et le redémarrage** de la fonction `loop()`. Cela signifie qu'il faut éviter de créer des boucles infinies dans la fonction `loop()`, sinon l'ESP8266 produira une erreur. Par conséquent, le code de la fonction `loop()` doit être conçu pour s'exécuter de manière cyclique.

Un grand nombre d'erreurs mystérieuses de l'ESP8266 s'expliquent par le fait que le processeur n'obtient pas assez de temps CPU pour gérer l'interface WLAN. Si l'exécution du programme peut prendre un temps excessif, par ex. en raison de grandes boucles, mieux vaut utiliser la fonction `yield()` ou la fonction `delay()` pour accorder à l'ESP8266 le temps requis pour gérer le WLAN.

Pour brancher le module NodeMCU sur la carte Modbus, il faut le connecter à un port USB de l'ordinateur, mais au préalable, quelques opérations préparatoires sont indispensables. L'EDI Arduino standard ne prend pas l'ESP8266 en charge, il faut donc d'abord le mettre à jour. Pour ce faire, sélectionner *Fichier* -> *Préférences* et dans la zone URL du gestionnaire de cartes supplémentaires, saisissez :

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Cliquez sur *OK*, puis sélectionnez *Outils* -> *Type de carte …* -> pour ouvrir la fenêtre *Gestionnaire de cartes*, et recherchez *ESP8266* puis installez les cartes de la *Communauté ESP8266*.

Après cette installation, la carte *NodeMCU 1.0 (Module ESP-E12)* figure sous *Outils* -> *Type de carte* > et sous *Outils* -> *Port* > figure le port auquel la carte NodeMCU est connectée (*/dev/ttyUSBx* sous Linux ou *COMx* sous Windows).

Dès lors, le premier programme simple d'essai du module NodeMCU peut être exécuté. À cet effet, ouvrez le programme d'exemple *Blink* sous *Fichier* -> *Exemples* -> ESP8266 -> *Blink*. Le programme du **listage 1** fait clignoter la LED installée sur la carte ESP8266. La fonction `loop()` accueille les instructions d'allumage/extinction de la LED. La fonction `setup()` configure la broche GPIO de la LED en sortie.

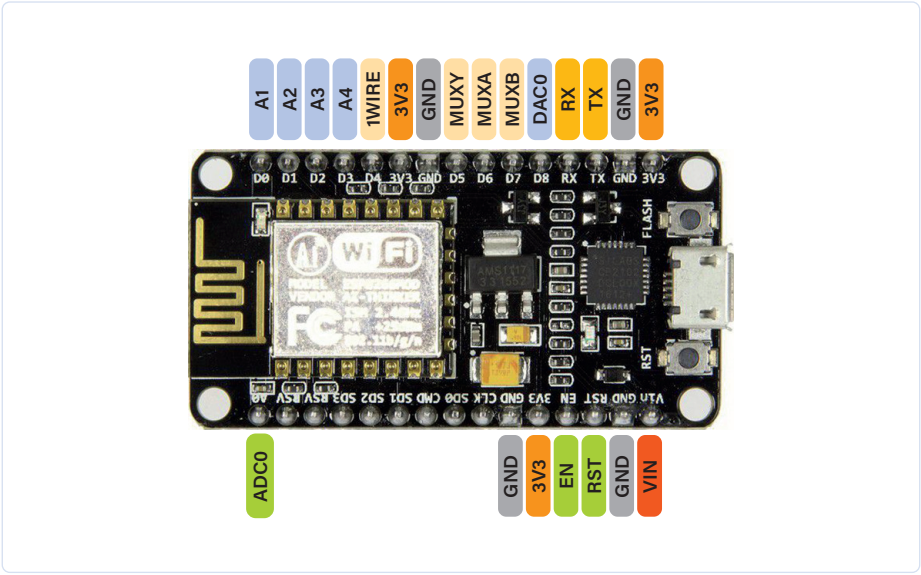


Figure 1. Brochage du module NodeMCU pour le projet Modbus.

Tableau 1. Caractéristiques techniques de l'ESP8266.

CPU RISC 32 bits	Tensilica Xtensa LX106 cadencé à 80 MHz
Mémoire de programme	64 Ko
Mémoire de données	96 Ko
Mémoire flash externe (en option)	512 Ko à 4 Mo
GPIO	10 broches ; interfaces prises en charge : SPI, I²C, I²S, UART. DMA et IRQ sont aussi prises en charge.
Wi-Fi	- IEEE 802.11b/g/n - Authentification WEP ou WPA/WPA2 - Wi-Fi direct (P2P), Soft AP - Pile de protocole TCP/IP intégrée
CA/N	1× convertisseur 10 bits
Antenne	intégrée
Puissance de sortie	+19,5 dBm en mode 802.11b
Taille	17,2 × 12,3 mm²
Courant de veille	< 1,0 mA

Listage 1. Programme de LED clignotante pour l'ESP8266.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(2000);
}
```

Figure 2. Fenêtre initiale de l'EDI Arduino.

Pour charger le programme dans la carte NodeMCU, cliquez sur l'icône *Téléverser* (flèche vers la droite). Au bout de quelques secondes, le programme est transféré puis la LED se met à clignoter. Ce test valide l'installation et la configuration de l'EDI.

Carte Modbus TCP

Toutes les entrées/sorties sont réalisées avec des borniers à vis enfichables ce qui confère un aspect industriel (v. photo en tête d'article) à la carte Modbus TCP qui accueille le module NodeMCU. Cela simplifie l'assemblage et permet de remplacer facilement les bornes à vis en cas de détérioration. La principale tâche de la carte Modbus TCP est de convertir le niveau de signal de 3,3 V des entrées/sorties du microcontrôleur en 24 V, compatible avec

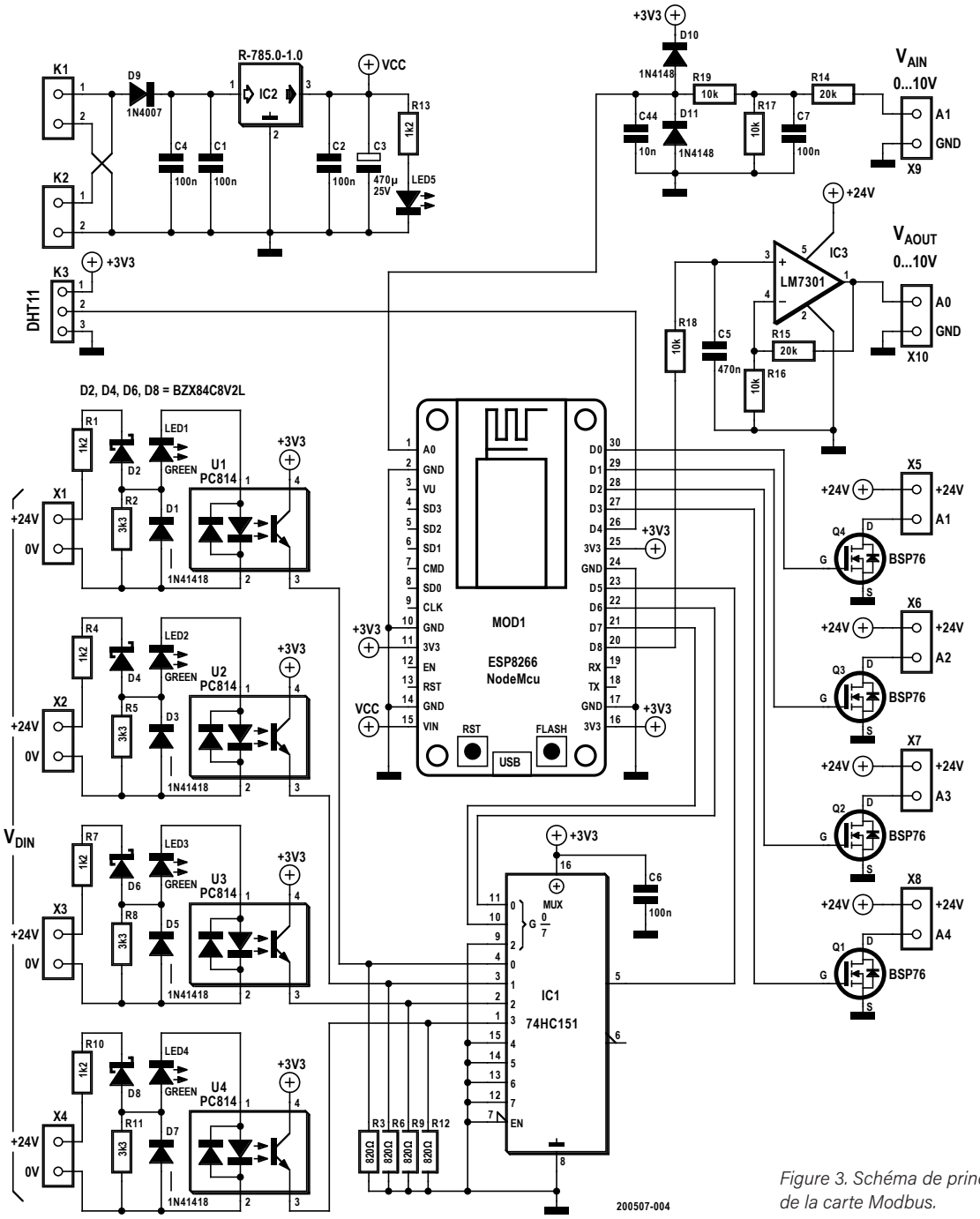


Figure 3. Schéma de principe de la carte Modbus.

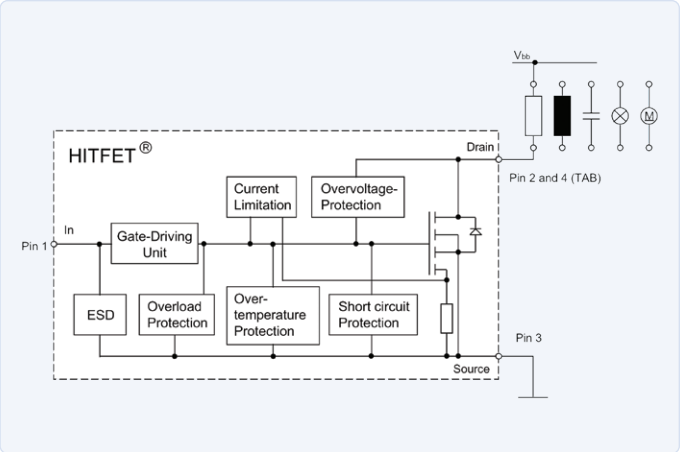


Figure 4. Structure interne du BSP76 (source : fiche technique Infineon).

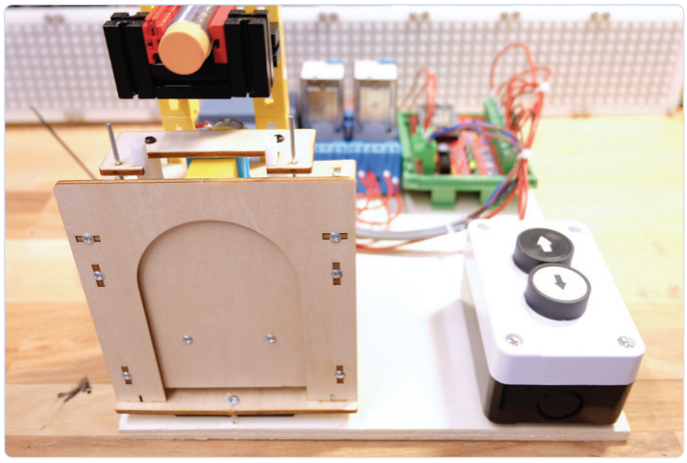


Figure 5. Porte d'ascenseur de maquette, en bois.

l'environnement industriel. Le schéma de la **figure 3** montre que les quatre entrées sont isolées galvaniquement du microcontrôleur par des opto-isolateurs PC814. En outre, ces entrées ne sont pas reliées directement à la carte ESP8266, mais à travers un multiplexeur 74HC151. Cette astuce permet d'économiser une broche GPIO et d'équiper le circuit d'un capteur DHT11 pour mesurer la température et l'humidité. La carte Modbus est donc capable d'exécuter des tâches simples de contrôle CVAC (Chauffage Ventilation Air Conditionné).

Les étages de sortie font appel à des *BSP76* de la série *HITFET* d'*Infineon* (**fig. 4**). Ces composants de commutation électronique ne sont pas de simples MOSFET de puissance, mais des circuits intégrés dotés de protections contre les surtensions et la surchauffe et qui limitent le courant. Ils sont donc capables de commuter indifféremment des charges résistives, inductives ou capacitives.

En plus des entrées et sorties numériques, la carte Modbus TCP a une entrée et une sortie analogiques. Ces deux ports fonctionnent sur la gamme de tension standard de l'industrie, soit 0 à 10 V. Le diviseur de tension R14/R17 réduit la tension à un maximum de 3,3 V sur l'entrée analogique. Cette entrée n'est pas isolée galvaniquement, mais C7, D10 et D11 assurent une protection notable contre les crêtes et les surtensions. Les résistances série R14 et R19 limitent le courant d'entrée pour que les diodes ne partent pas en fumée en cas de problème sur l'entrée.

Sur la sortie analogique, l'*AOP IC3* à usage général (*LM7301* avec capacité rail à rail en entrée comme en sortie) augmente la tension de sortie du *CA/N* du NodeMCU. Les résistances R15 et R16 confèrent un gain de 3 à cet AOP par ailleurs alimenté par le rail 24 V.

Le convertisseur DC/DC de *Recom* mérite une mention spéciale. Il abaisse la tension d'alimentation de 24 V aux 5 V nécessaires à l'alimentation du module NodeMCU. Le rendement de ce convertisseur DC/DC est largement supérieur à 90 %, et son utilisation est très polyvalente. Il peut par ex. être utilisé pour produire une tension d'alimentation négative pour les AOP. Cela vaut la peine de jeter un coup d'œil à la fiche technique du convertisseur DC/DC [4]. La diode *1N4007* montée en série sur l'entrée 24 V protège le circuit contre l'inversion de polarité de connexion.

Les fichiers *Gerber* de la carte imprimée peuvent être téléchargés depuis la page du projet [8]. La carte est disponible auprès de l'auteur [7] soit nue, soit entièrement assemblée et prête à l'emploi.

Démonstration à l'aide d'une maquette en bois

Pour montrer la capacité du module Modbus à gérer une tâche industrielle type, les auteurs ont utilisé une porte d'ascenseur de maquette provenant du magasin en ligne chinois bien connu [5] (**fig. 5**). Un bouton-poussoir ouvre la porte, l'autre la ferme. Deux capteurs de proximité capacitifs détectent les positions extrêmes de la porte. Le moteur qui ouvre et ferme la porte est alimenté par un pont en H basé sur deux relais de puissance.

Le **listage 2** donne le programme de commande. Celui-ci définit d'abord des mnémoniques pour les entrées et sorties du

LIENS

[1] Carte NodeMCU : www.elektor.fr/17952
[2] Téléchargement de l'EDI Arduino : www.arduino.cc/en/software
[3] Fiche technique BSP76 : https://bit.ly/3r3GP99
[4] Fiche technique du convertisseur DC/DC : https://recom-power.com/pdf/Innoline/R-78-0.5.pdf
[5] Porte d'ascenseur de maquette, en bois : www.aliexpress.com/item/33008243573.html
[6] Porte d'ascenseur de maquette, en bois sur YouTube : https://youtu.be/VHIBQswdA0E
[7] Josef Bernhardt : www.bernhardt.de
[8] Page du projet : www.elektormagazine.fr/200507-04

Listage 2. Commande de porte d'ascenseur.

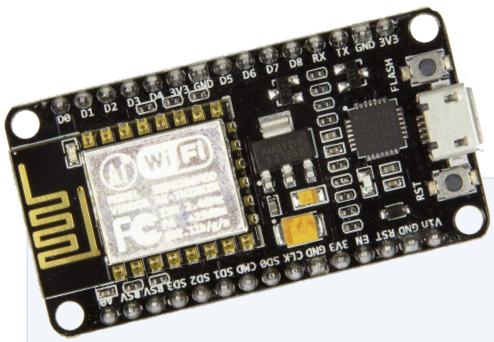
```
#define A1 16
#define A2 5
#define A3 19
#define A4 0
#define MUXA 12
#define MUXB 13
#define MUXY 14
#define E1 0
#define E2 1
#define E3 2
#define E4 3


void setup() {
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(MUXA, OUTPUT);
  pinMode(MUXB, OUTPUT);
  pinMode(MUXY, INPUT);
}


bool input(int i){
  digitalWrite(MUXA,(i&1));
  digitalWrite(MUXB,(i&2)>>1);
  delay(1);
  return digitalRead(MUXY);
}


void output(int i, int v){
  digitalWrite(i,v);
}


void loop() {
if(input(E4)& !input(E1)){output(A1,HIGH);}
if(input(E3)& !input(E2)){output(A2,HIGH);}
if(input(E1)){output(A1,LOW);}
if(input(E2)){output(A2,LOW);}
}
```

circuit. Les différentes broches d'E/S sont initialisées comme entrées et sorties par la fonction `setup()`. Viennent ensuite les fonctions `input()` et `output()` qui facilitent un peu l'accès aux broches d'entrée/sortie. La fonction `loop()` constitue le programme principal où les sorties de commande du moteur sont activées par les boutons et désactivées par les capteurs « fin de course ». La fonction `loop()` peut être exécutée de manière répétitive. Pour voir la porte d'ascenseur en action, il suffit de regarder cette vidéo *YouTube* [6].

Perspective

Malgré sa valeur démonstrative, la commande de porte d'ascenseur réalisée ne rend pas compte du potentiel de la carte Modbus et du module WLAN. Nous y reviendrons dans la 2e partie de cet article, où nous nous familiariserons avec le protocole Modbus et parlerons des logiciels utilisables pour communiquer sur le bus. Nous montrerons également comment configurer le module Modbus.

200507-04



Contributeurs
Conception et article :
Josef Bernhardt et **Martin Mohr**
Rédaction : **Rolf Gerstendorf**
Traduction : **Yves Georges**
Mise en page : **Giel Dols**

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (josef@bernhardt.de) ou contactez Elektor (redaction@elektor.fr).

À propos des auteurs

Josef Bernhardt s'est intéressé à l'électronique dès son plus jeune âge. Il a construit son premier poste à galène à l'âge de douze ans, puis a continué avec d'autres circuits. Dans les années 1980, il fit ses premières expériences en programmation sur le Commodore VC20, et il connaît également bien les techniques de l'assembleur sur 8088. Il peut se prévaloir de plus de 30 ans d'expérience en électronique à l'Université de Ratisbonne (Allemagne), où il a œuvré dans le développement électronique et logiciel. Grâce à sa propre installation de fabrication CMS, il réalise également des projets électroniques pour des clients.

Martin Mohr vit le jour à l'époque des mémoires à tores magnétiques et des interrupteurs Strowger, ce qui lui permit de vivre personnellement toute l'épopée des techniques informatiques modernes. Sa fascination pour tout ce qui clignote remonte à sa prime jeunesse et a été renforcée par sa formation en électronique. Après des études en informatique, il a surtout travaillé au développement d'applications Java, mais le Raspberry Pi a ravivé sa passion pour l'électronique.

PRODUITS

› **NodeMCU-Modul**
www.elektor.fr/nodemcu-microcontroller-board-with-esp8266-and-lua