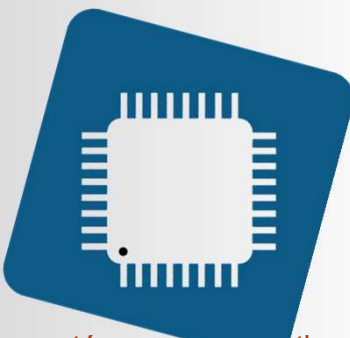


UNIVERSITÉ


La Rochelle



C5-160551-INFO - Objets connectés : programmation microcontrôleur

Généralités
Définitions et introduction microcontrôleurs
Bernard Besserer

1



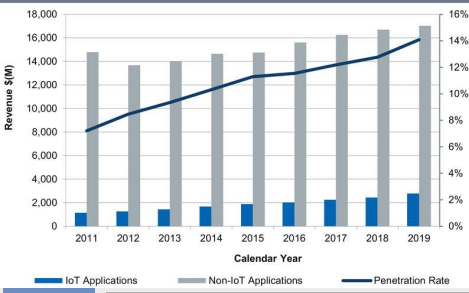
UNIVERSITÉ

La Rochelle

Le marché des microcontrôleurs

- Aller vers l'utilisation des microcontrôleurs dans leur utilisation pour les objets connectés...
- 98% des processeurs pour les « systèmes enfouis »

MCU market in IoT applications compared to markets outside of IoT



Calendar Year	IoT Applications Revenue (\$M)	Non-IoT Applications Revenue (\$M)	Penetration Rate (%)
2011	1,000	14,000	6.7%
2012	1,200	13,500	8.1%
2013	1,500	13,000	9.6%
2014	1,800	12,500	11.2%
2015	2,000	12,000	12.5%
2016	2,200	11,500	13.5%
2017	2,500	11,000	14.5%
2018	2,800	10,500	15.5%
2019	3,000	10,000	16.5%

Contrôle industriel Militaire Automobile : Calculateur de bord Airbags ABS Sécurité Transmission Climatisation GPS Injection...	Applications domestiques : Téléphone/Fax Téléphone mobile Sécurité Portes garage TV, set top box Jeux vidéo Caméscopes Lecteurs DVD, BR-Disk Instruments musique Jouets Réfrigérateurs/Fours...	Bureautique : Téléphones PC Fax Sécurité Imprimantes Photocopieurs PDA...
--	---	---

Aujourd'hui, un aspirateur exige un microcontrôleur 16 bits !
 Un téléphone portable intègre au minimum un microcontrôleurs 32/64 bits multi-cœurs, un GPU (Graphic Processing Unit), un DSP (Digital Signal Processor) et d'autres microcontrôleurs

2

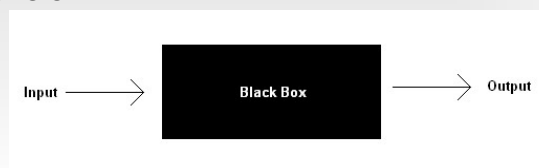
Microcontrôleurs vs microprocesseur

- Unité centrale simple (ALU), manipulant des nombres binaires de 8, 16 ou 32bits, avec des fonctionnalités étendues au niveau des entrées/sorties
Un processeur performant, manipulant des nombres codés sur 64 bits et en format « virgule flottante » sur 64 ou 128 bits.
- Un contrôle facile des E/S à partir du logiciel
Il faut passer par des couches de firmware et pilotes (drivers)
- Peu de mémoire de données (1Ko ... 512 Ko RAM), souvent toute la mémoire se trouve « on-chip »
Beaucoup de mémoire (4GB mini), généralement ajoutée sous forme de barrettes mémoires
- Peu de mémoire pour le programme (4Ko...128Ko), souvent sous forme de PROM, EPROM, EEPROM, ...)
Pas de différence entre mémoire programme et mémoire donnée, mais il y a une mémoire « de masse » en plus (Disque Dur, SSD)
- Donc, pour un microcontrôleur :
 - Peu de mémoire, puissance de calcul limitée !
 - Mais faible consommation énergétique, forte intégration !

3

Ordinateur vs. Microcontrôleur

- | | |
|--|---|
| <ul style="list-style-type: none"> ■ Ordinateur <ul style="list-style-type: none"> • Mémoire et unité de calcul <ul style="list-style-type: none"> • Exécute le code • Entrées (Inputs) : <ul style="list-style-type: none"> • Keyboard • Mouse • Microphone • Sorties (Outputs) : <ul style="list-style-type: none"> • Monitor • Speakers | <ul style="list-style-type: none"> ■ Microcontrôleur <ul style="list-style-type: none"> • Mémoire et unité de calcul <ul style="list-style-type: none"> • Exécute le code • Inputs <ul style="list-style-type: none"> • Voltage on pins • Outputs <ul style="list-style-type: none"> • Voltage on pins |
|--|---|



4

Définition, evolutions

- Un microcontrôleur est un composant programmable, d'une architecture similaire à celle d'un microprocesseur.
- L'avantage d'un microcontrôleur est évidemment d'intégrer d'un grand nombre d'Entrées/Sortie (E/S ou I/O) : Computer on a chip / SoC (System on a Chip)
- Un microcontrôleur doit faciliter le développement d'objets intelligents et permettre de réduire les coûts, généralement liés à l'interfaçage
 - 1980 : Contrôle industriel avec des circuits intégrés et des composants discrets
 - 1990 : Premiers microcontrôleurs, nécessitant énormément de « glue logic »
 - 2000 : Essor des microcontrôleurs « modernes »
 - Intégration de composant autrefois externe (contrôleur LCD)
 - Développement des familles : de centaines de variantes

5

Exemple d'évolution : thermostat



6

Tamagochi breakdown



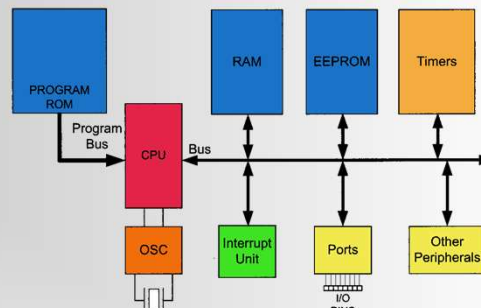
Part	Quantity in Product
Capacitors (0402)	8
Resistors (0402)	5
Crystal Oscillator	1
Electrolytic Capacitor	1
Speaker (Piezoelectric)	1
Battery (3V)	1
LCD Screen	1
Printed Circuit Board	1
Plastic Housing	1
Clear Plastic Screen	1
Soft Plastic Dutton Pallet	1
Integrated Circuit	1
Screws	4
Keychain	1
Plastic Washer	1
Cardboard LCD Screen Backing	1

Fabrication : 20% du cout final
\$1,6 pour un prix de vente de \$7,99

7

Microcontrôleurs : architecture générale

- On retrouve en général les éléments suivants (dans le chip)



- Un processeur (CPU : *Computing Processing Unit* ou *ALU Arithmetic and Logic Unit*) qui travaille généralement sur les mots de largeur 8, 16 ou 32 bits (le support des nombres flottants n'est généralement pas assuré par les « petits » microcontrôleurs.)
L'ALU exécute le code, au rythme de l'horloge (OSC)
- De la mémoire de différents types (RAM, PROM, EEPROM)
- Beaucoup d'interfaces (Ports : parallèles, série), timers, etc... Cette énorme diversité des E/S permet de décliner une famille
- Un bus relie ces composants, généralement de largeur de mot équivalente à la largeur des mots manipulés par l'ALU

8

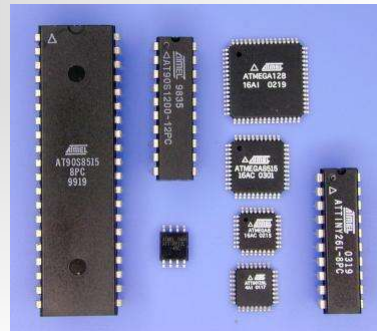
Fabricants et « familles » de microcontrôleurs



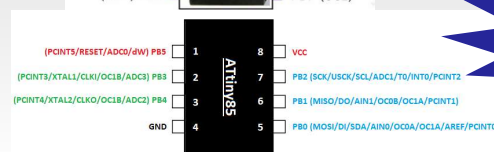
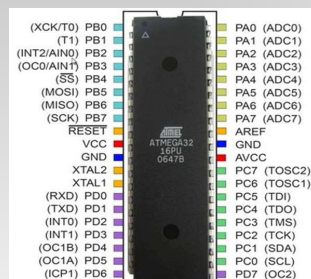
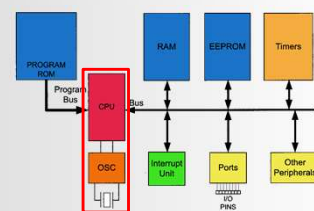
Espressif

Exemple : AT Mega

ALU 8 bits, RISC
32 KB flash, 1KB EEPROM,
2 KB SRAM, 32 registres
Jusqu'à 54 GPIOs
3 timers et compteurs
Interface série (USI et SPI)
CAN 10bits

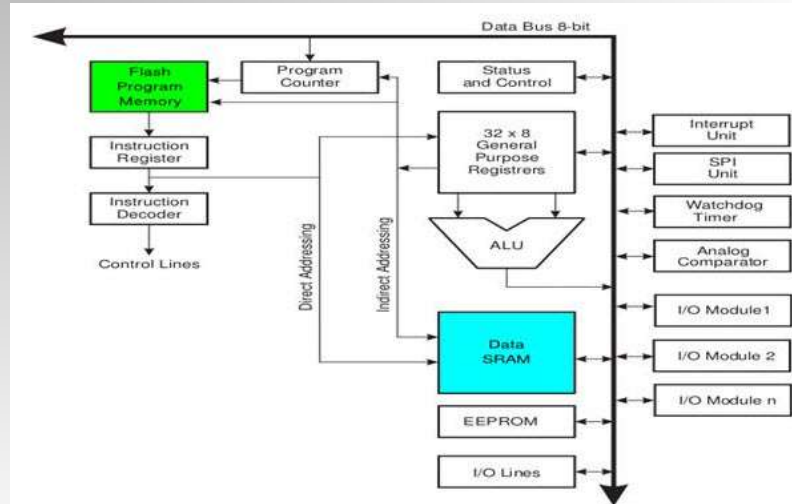


- Le nombre de fonctionnalités ramenés vers l'extérieur varie
GPIO (general purpose input/output)
=pin=broche



Multiplexage des fonctions

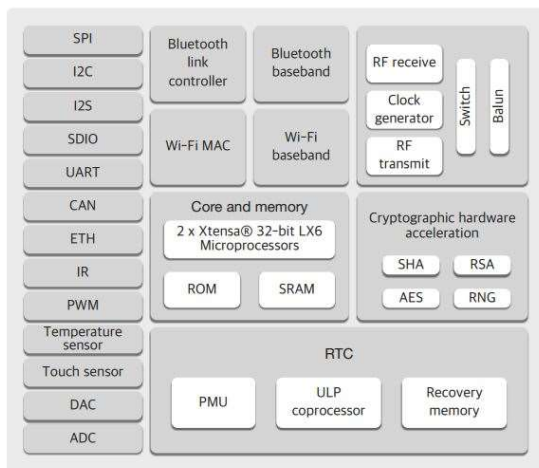
Exemple : ATMEL AVR Mega 32



11

Exemple : MSP32

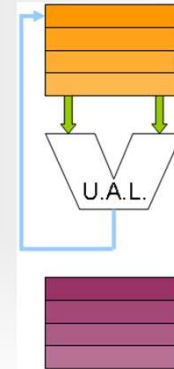
- 2 ALU 32bits
- Communication Radio : Wifi et BT
- Accélération cryptage et décryptage (sécurisation des liaisons sans fil)
- Liaisons séries selon divers protocoles



12

Rappel sur l'architecture interne

- Arithmetic and Logic Unit (Unité Arithmétique et Logique) = Circuits électroniques (transistors, ...) assurant :
 - Les opérations de bases : additions / soustractions et opération logiques (ET, OU, NOT, ...)
 - Tests & branchements : généralement une opération qui affecte un indicateur et influe au final sur le déroulement du code
 - Le transfert des données de case mémoire vers case mémoire.
- Remarques :
 - Multiplication et division...
 - Opération sur des nombres « à virgule »



13

Architecture RISC

- Reduced Instruction Set Computer
- Étude statistique ayant conduit au RISC :
 - Les opérations les plus usitées sont :
 - les opérations d'échange entre l'ALU et la mémoire.
 - les appels à des sous-programmes.
 - L'instruction d'appel d'une procédure est la plus gourmande en temps : sauvegarde et restitution du contexte et passage des paramètres.
 - Maintenir les structures de données complexes comme des variables globales : éviter de les passer en paramètre et éviter de saturer la pile
- On ne programme plus les microcontrôleurs en ASSEMBLEUR (transcription du jeu d'instruction dudit microcontrôleur) mais dans un langage comme C ou JAVA -> Le compilateur joue son rôle !

14

Concepts du RISC

- Jeu réduit d'instructions
- Exécuter si possible 1 instruction par cycle
- 2 instructions seulement pour l'accès mémoire (load/store) ou 1 seule instruction (move)
- Beaucoup de registres
- Certaines instructions plus complexes sont des séquences d'instruction plus simples
 - Instructions dites émulées
 - Exemple : CLR R4 (Effacer le registre R4)
 - en fait, l'instruction CLR peut ne pas exister, et l'action effectuée est en fait :

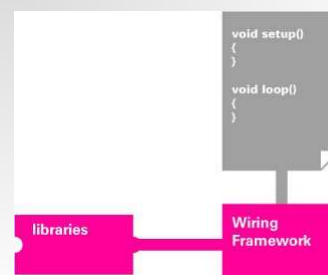
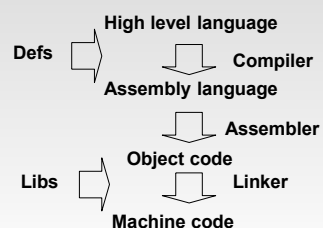

```
XOR R4, R4
```
 - Séquences créés par des compilateurs
- Designs « anciens » réutilisés pour des microcontrôleurs

15

Fonctionnement de l'ALU

- Comment ça marche...
 - Cours EPFL 2.1 : Architecture d'un microcontrôleur
 - <https://www.youtube.com/watch?v=SFbMPX7u9KI>
- Programmation d'un microcontrôleur
 - Assembleur
 - Langage de haut niveau

Langage de haut niveau
+ framework



16

Langages de programmation

ASSEMBLEUR

- L'assembleur est spécifique à chaque famille de microcontrôleur (le code n'est pas portable)
- Proche du matériel, permet facilement l'accès aux Entrées/Sorties, gestion des interruptions, des modes de veille.
- Il n'y a pas de typage de données, pas de structures de données complexes, pas de structure de contrôle d'exécution (for, while, if, switch)
- Maîtrise de la localisation en mémoire (RAM, ROM, ...) des variables, constantes et instructions du code exécutable
- Opérations arithmétiques complexes non implémentées : pas d'instruction assembleur pour cela (types long long, arithmétique en virgule flottante) -> routines
- Difficile à documenter
- Programmation « compacte »
- Exécution très rapide

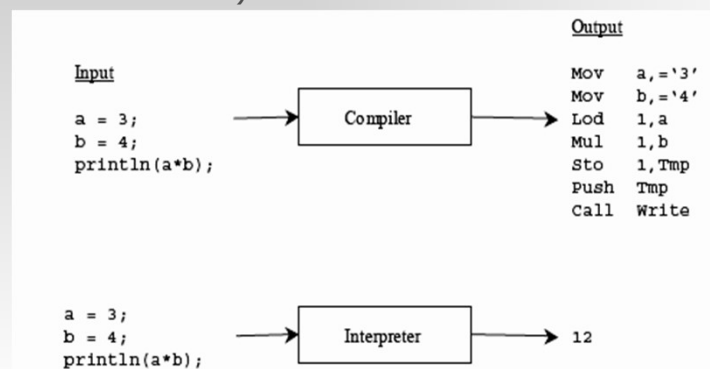
LANGAGE HAUT NIVEAU

- Indépendant de la plateforme
- Le typage des données évite bien des erreurs
- Le type de donnée long long ou float ou double fait partie du langage, ainsi que les opérateurs associés.
- Structure de données évolués possibles, et plus encore avec :
 - C++
 - Utilisation des bibliothèques (STL)
- Placement en mémoire des variables et constantes est délicat, dépend du typage des données, nécessite des mots clés particuliers
- L'accès aux E/S dont la localisation est fixe n'est guère aisée, et nécessite une syntaxe spéciale.
- Accès aux modes de veille ? Au routines d'interruption ?

17

Rappel : Phase de compilation

- Le compilateur analyse le code « de haut niveau » et converti celui-ci en code machine
- Cross-compilation : le compilateur génère du code pour l'architecture de la cible (le microcontrôleur).



18

Phase de compilation

- Exemple pour ce code en 'C' :

```
while (x < (a-b))
    x = 2*x;
```
- Sortie du compilateur, code assembleur "generique"


```
L1:  LOD R1,A      // Load A into reg. 1
     SUB R1,B      // Subtract B from reg. 1
     STO R1,Temp1  // Temp1 = A - B
     CMP X,Temp1   // Test for while condition
     BL L2         // Continue with loop if X<Temp1
     B L3          // Terminate loop
L2:  LOD R1,'2'    // Load immediate
     MUL R1,X      // X = 2*X
     STO R1,X      // X = 2*X
     JMP L1        // Repeat loop
L3:
```

Phase de compilation

- Exemple pour ce code en 'C' :

```
while (x < (a-b))
    x = 2*x;
```
- Sortie du compilateur,code assembleur "RISC"
 - Pas de LOAD et STORE, mais MOVE
 - Pas d'instruction SUB ni MUL


```
L1: MOV R1,B      // Load B into reg. 1
     NOT R1       // B complement's to 1
     ADD R1,'1'   // B complement's to 2 (-B)
     ADD R1,A     // Add A to reg. 1
     MOV Temp1,R1 // Temp1 = A - B (can be discarded)
     NOT R1       // Temp1 complement's to 1
     ADD R1,'1'   // Temp1 complement's to 2 (-Temp1)
     ADD R1,X     // Test for while condition
     BN L3        // stop if x >= a+b
     MOV R1,'2'   // no multiply instruction !
     CALL mult
     MOV X,R1     // X = 2*X
     JMP L1       // Repeat loop
L3:
```



Microcontrôleur : Recap...

- Processeurs basés sur un design généralement ancien, manipulant des mots de taille réduite, avec un jeu d'instructions réduit.
- Disposant de peu de mémoire (registres, quelques KB de RAM, plusieurs dizaines à quelques MB de flash/ROM)
- Conceptions et outils de développement (compilateurs, émulateurs, etc...) amortis donc économiques, et bibliothèques de code existants car long historique d'utilisation
- A savoir :
 - Leur architecture n'aime ni les multiplications, ni les divisions (ou opérateur modulo...)
 - Leur architecture n'aime pas manipuler les nombres réels
 - Comme la RAM est réduite, la pile est réduite :
 - Ne pas effectuer des appels de fonction avec beaucoup de paramètres
 - Ne pas passer en paramètre des données complexes (tableau, structure, ...)
 - **Ne pas implémenter de fonctions récursives**

21

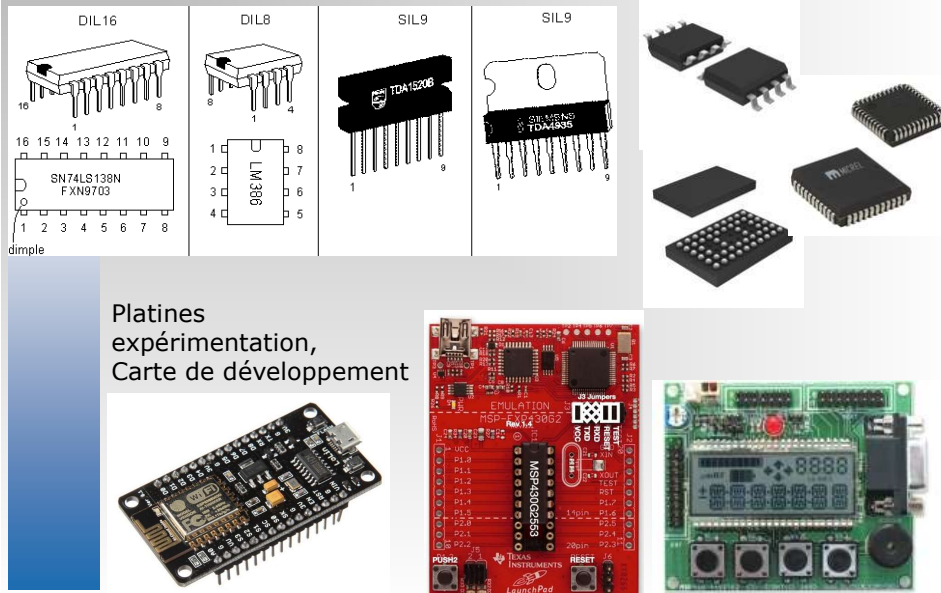


Microcontrôleurs : les E/S

- Les microcontrôleurs intègrent des périphériques et se distinguent par leur grand nombre d'Entrées/Sorties (E/S)
 - Entrées/Sorties digitales (port parallèle, port série)
 - Entrées/sorties analogiques
 - Timer(s) interne(s) avec E/S (comptage, génération d'impulsions)
 - Sorties de pilotage LCD
 - Interfaces avec des bus de périphérique (I2C, CAN, ...)
 - Eventuellement, interface USB (host)
- L'usage étant orientés « temps réel », Le E/S doivent pouvoir fonctionner en mode interruptif
- Contraintes fortes :
 - *footprint* et nombre de broches (= multiplexage des fonctions)
 - Consommation électrique

22

Microcontrôleurs : packaging



Platines
expérimentation,
Carte de développement

23

Comment choisir ?

- Platine avec interfaces E/S
 - Robuste, testé
 - Pas ou peu de câblage :
 - Peu de risque de casse, courts-circuits
 - Les projets et travaux possible dépendent des E/S disponibles
- Platine sans équipements E/S sur la platine
 - Moins chères mais fragiles
 - Il faut acquérir (et maintenir en état) les capteurs/actionneurs, plaquettes d'essais, fils, ...
- Simulation sans hardware réel possible (Fritzling) mais est-ce bien le but recherché ?
- Le choix du logiciel est déterminant :
 - Programmation
 - Bibliothèques disponibles

24

Raspberry Pi & Co.

■ Ordinateur simplifié avec quelques entrées / sortie

- Processeur 64 bit
- Système d'exploitation (linux, Win10 IOT)
- Sortie vidéo
- 1 GB RAM
- Carte SD
- 27 GPIO, UART, I²C
- Faible consommation



■ Pour :

- Serveur multimédia, centrale domotique, console (retro-) gaming, ...

La platine Arduino

- Platine développée par des enseignants à Ivree, Italie, à destination d'étudiants mais aussi artistes, créateurs, ... souhaitant créer des interactions entre capteurs et actionneurs (à partir de 2005)
- Pensé pour des novices, le framework *Wiring* simplifie la programmation à l'extrême.
- Plateforme open source :
 - Cool, fait du buzz, plébiscité par les hackers & makers
 - Dans la réalité... moins de 1% de platines conçues par les « amateurs ». Les chinois clonent, mais beaucoup d'acheteurs « soutiennent » l'initiative et achètent « genuine arduino »
 - Prix de fabrication : \$1 à \$2, prix de vente \$30
 - Estimation : 700000 platines officielles vendues en 2013

Pour comparer, STM 429 DISCOVERY, env. 2015

\$ 23.52



- STM32F429ZIT6 microcontroller featuring 2 MB of Flash memory, 256 KB of RAM
- CPU 32bits ARM Cortex M4 180MHz / FPU / MCU
- 2.4" QVGA TFT LCD (touchscreen)
- SDRAM 64 Mbits
- L3GD20, ST MEMS motion sensor, 3-axis digital output gyroscope
- Six LEDs
- Two pushbuttons (user and reset)
- USB on board

Coût microcontrôleur : \$0,23 pour un STM8S003F3 (ALU 8bits) ... \$0,40 pour un NXP LPC811 (ALU 32-Bit Cortex M0+)

27

ESP8266

- Circuit intégrant un microcontrôleur et un module de communication WiFi développé par le fabricant chinois Espressif.
- Module pour équiper des imprimantes de la connectivité WiFi -> adopté par les hackers et makers pour des applications IoT basiques
- kit de développement logiciel (SDK) depuis fin 2014
- Rapidement, le Framework Arduino est adapté à ce circuit. Des versions intégrant un interpréteur sont également disponibles (Lua, uPython)

- 32-bit RISC CPU 80 MHz
- 64 KB of instruction Mem, 96 KB of data RAM
- IEEE 802.11 b/g/n Wi-Fi
- 16 GPIO pins
- SPI, I²C, UART
- 10-bit ADC



28

- Sorti en 2016, SoC intégrant un microcontrôleur et les modules de communication WiFi et BT.
 - CPU 32bits single/dual core @ 160 or 240 MHz
 - 520 KB SRAM
 - Communication Radio : Wifi et BT
 - Accélération cryptage et décryptage (sécurisation des liaisons sans fil)
 - Liaisons séries selon divers protocoles
- Cette fois ci conçu comme un microcontrôleur : Documentation, SDK, IDE, debug, ...

Espressif ESP32

Adafruit Huzzah32

