

Internet of Things architectures et technologies

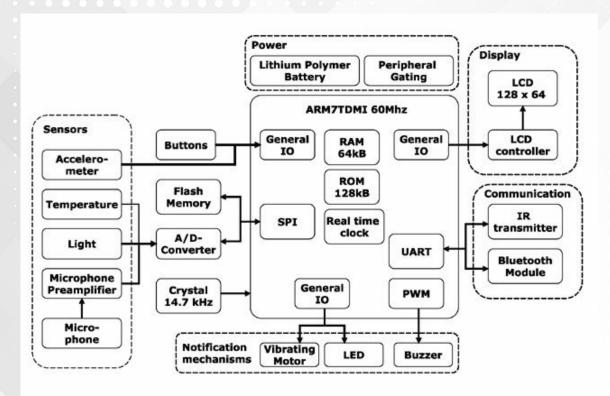
Chapitre #1 Hardware

?

Qu'est-ce qu'un "objet connecté"? Comment le concevoir?

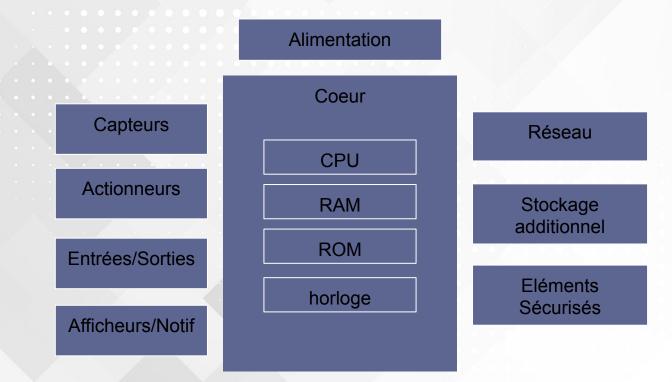


Anatomie d'un équipement connecté (exemple=

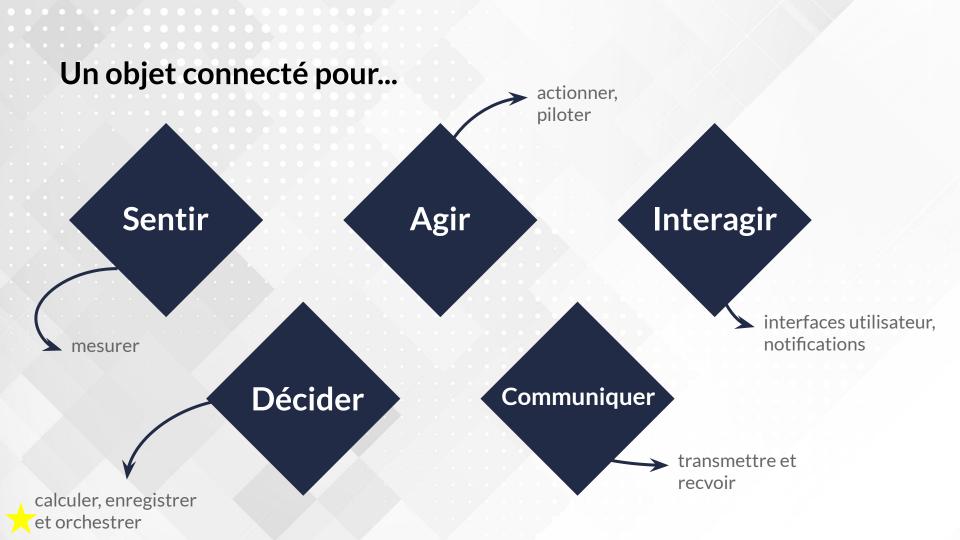


source: https://users.ece.cmu.edu/~agr/old/old-projects/ewatch/hardware.html

Plus schématiquement







"Sentir" > Capteurs

Comment mesurer le réel?

accéléromètre:

accélération par axe (X,Y,Z) (unité: m/s^2)

gyroscope:

rotation sur différents axe (Y,Y,Z) (unité: rad/sec)

magnétomètre:

"boussole", indique champ magnétique par axe (X,Y,Z) (unité: uTesla)

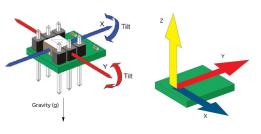
thermomètre:

température ambiante (unité: °C/°F/°K)

luxmètre:

illumination (unité: lux)





Diagrams illustrating the axes of 2-axis (left) and 3-axis accelerometers. This particular 2-axis sensor is also capable of tilt measurements.

capteur de distance:

distance à l'obstacle le plus proche (unité: m) (ex: infra-rouge, ultra-sons)

manomètre:

mesure la pression (unité: hPa)

humidité:

(unité: %)











détecteurs de gaz:

capteurs électrochimiques (méthane, butane, alcohol, ethanol, monoxide, hydrogen, ozone...) (unit: ppm)







source: http://playground.arduino.cc/Main/MQGasSensors

capteur optique:

détection de coupure de faisceau

capteur de position:

via codeur rotatif, interrupteur à lame souple

capteur de contact:

microrupteur / bouton poussoir

microphone:

(onde acoustique)

caméra:

image / flux vidéo









Capteurs - critères

- plage de mesure
- précision / sensibilité
- réactivité / fréquence de rafraîchissement
- conditions de fonctionnement
- encombrement
- consommation électrique

Exemple:

Accuracy for temperature:

```
-< +/-0.5°C from 0° to +65°C

-< +/-1°C from -30°C to 0°C and

from +65°C to +90°C

-< +/-2°C below -30°C and above

+90°C
```

Accuracy for relative humidity:

-< +/- 3% from 20% to 80% -< +/- 5% below 20% and above 80%

- Measurement range: -40 to 120°C and 0 to 100% for relative humidity
- Resolutions: 1/100°C for temperature and 4/100% for relative humidity.
- Luminosity indicator: in % of the luminosity level.



Un "sens" particulier: la géo-localisation

Géolocalisation

GPS (ex-"Navstar")

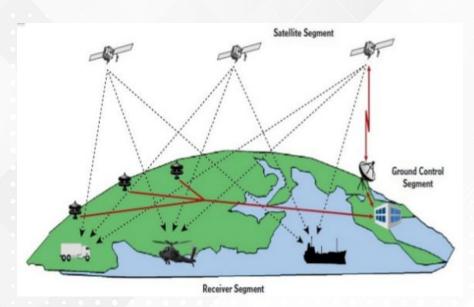
(Global Positioning System)

USA (DoD), 24 satellites

disponible depuis 1994

1575 MHz

précision: ~20m



Améliorable via SBAS (Satellite based augmentation system), en Europe: EGNOS



Géolocalisation

GLONASS

Russie,

à nouveau disponible depuis 2010 même propriétés que GPS

Galileo:

Europe,

gratuit: même précision que GPS,

commercial: ~10cm

fin de déploiement 2020

Beidou / COMPASS:

Chine, disponibilité 2020





Géolocalisation

Ce principe de "triangulation" peut s'appliquer avec un référentiel fixe:

- Réseaux mobile: utilisation des identité des relais (BTS) + information de niveau de signal (RSSI)
- Réseaux Wifi: identifications via les SSIDs (cf. https://wigle.net/)
- En intérieur: déploiement de "beacons" (ex: bluetooth/BLE, RFID, ...) qui émettent en boucle leur identité

"Agir" > Actionneurs

Comment agir sur le réel?

Actionneurs - examples

relais:

contrôler un courant important





moteur / moteur pas-à-pas / servo-moteur







source: http://playground.arduino.cc/Main/MQGasSensors

"Interagir" > Comment informer/avertir un utilisateur ou lui offrir les Sees: —Interfaces

Signaler à l'utilisateur

voyants / LEDs







buzzer / audio

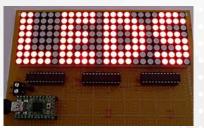




Signaler à l'utilisateur

afficheurs (LCD, LED, e-paper)









vibreur



Contrôles

boutons poussoir / interrupteurs









Potentiomètres:







Contrôles

écran tactile



autre...







"Décider" > CPU et mémoires

Histoire de l'électronique en (très) bref

- 1800 pile électrique (Volta)
- 1827 loi d'Ohm (Ohm), début électromagnétisme (Ampère)
- 1846 induction électromagnétique (Faraday)
- 1873 équations de Maxwell
- 1876 téléphone (Bell)
- 1879 ampoule (Joseph Swan)
- 1890 cohéreur de Branly (détection onde)
- 1895 transmission à distance sans fil (Marconi et Popov)
- 1904 premier tube électronique (Fleming)
- 1907 tube triode (Lee De Forest)
- 1918 premier additionneur binaire (Bloch et Abraham)
- 1947 transistor à pointes
- 1954 premiers ordinateurs à transistors
- 1959 procédé de fabrication planar (Noyce)
- 1959 circuits intégrés (Kilby)
- 1969 mémoire 64 bits (Intel)
- 1971 premier microprocesseur (intel: 4004)
- 1975 RISC (IBM)



John Bardeen, William Shockley et Walter Brattain (Laboratoires Bell)

mémoires

RAM = Random Access Memory / "mémoire vive":

mémoire volatile, très performante (DRAM, SRAM, MRAM)

ROM = Read Only Memory / "mémoire morte":

mémoire non volatile, lecture seul, sauf opération de "programmation" (EEPROM)

Flash:

mémoire réinscriptible, sorte d'EEPROM éditable/réinitialisable par zone

(NOR, NAND)











microprocesseur VS microcontrôleur

microprocesseur =

unité de calcul générique dans un seul boîtier





micro-contrôleur =

une seule puce comprenant microprocesseur + mémoire + I/O + timer...





Exemple:

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - **Fully Static Operation**
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 64K/128K/256KBytes of In-System Self-Programmable Flash
 - 4Kbytes EEPROM
 - 8Kbytes Internal SRAM
 - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program

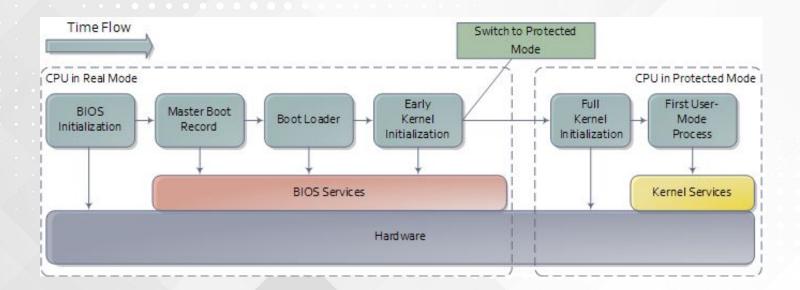
 - . True Read-While-Write Operation
 - Programming Lock for Software Security
- Endurance: Up to 64Kbytes Optional External Memory Space
 Atmel[®] QTouch[®] library support
- - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix acquisition
 - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

Exemple

- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
- Real Time Counter with Separate Oscillator
- Four 8-bit PWM Channels Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits
- (ATmega1281/2561, ATmega640/1280/2560)
 - Output Compare Modulator
 - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
- Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
- Master/Slave SPI Serial Interface
- Byte Oriented 2-wire Serial Interface Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby,
- and Extended Standby
 - I/O and Packages 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
 - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
 - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
 - RoHS/Fully Green Temperature Range:

 - 40°C to 85°C Industrial
 - Ultra-Low Power Consumption
 - Active Mode: 1MHz, 1.8V: 500µA
 - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
- ATmega640V/ATmega1280V/ATmega1281V:
- 0 4MHz @ 1.8V 5.5V, 0 8MHz @ 2.7V 5.5V
- ATmega2560V/ATmega2561V:
- 0 2MHz @ 1.8V 5.5V, 0 8MHz @ 2.7V 5.5V ATmega640/ATmega1280/ATmega1281:
- O 8MHz @ 2.7V 5.5V, O 16MHz @ 4.5V 5.5V ATmega2560/ATmega2561:
 - 0 16MHz @ 4.5V 5.5V

Séquence de boot d'un ordinateur



source: http://duartes.org/qustavo/blog/post/how-computers-boot-up/

Élément Sécurisé (SE)

"Puce" = circuit intégré avec contacts exposés

"Elément sécurisé" = plate-forme matérielle inviolable, peut contenir mémoire, processeur (typiquement processeur cryptographique + registre de secrets)

Exemple: carte SIM







Alimentation

Alimentation externe (secteur, générateur...):

pas de problématique d'autonomie

Alimentation autonome:

contrôler la consommation et le niveau d'énergie disponible devient un enjeu critique.

Batterie ("accu"):

chimique (lithium, nickel - NiMH, alcaline), capacitors... varient en densité, tenue de charge, comportement en haute/b température

Énergie ambiante ("harvesting"): soleil, chaleur, mouvement, radio (ex: WiFi)...







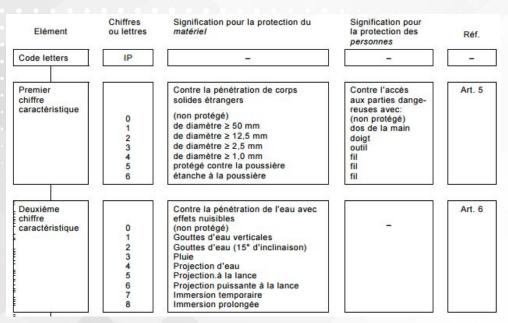




Environnement

Indices de Protection (IP)

établis par "Commission électrotechnique Internationale" (CEI / IEC)







Conditions opérationnelles

Equipements "Durcis" - capables de fonctionner dans des conditions extrêmes:

- Vibrations
- Ondes EM / particules (nucléaire, espace)
- Basses/hautes températures

- Commercial: 0 °C to 85 °C
- Industrial: -40 °C to 100 °C
- Automotive: -40 °C to 125 °C
- Extended: -40 °C to 125 °C
- Military: -55 °C to 125 °C







source: https://en.wikipedia.org/wiki/Operating temperature

Software "embarqué"

Software "embarqué" - ("firmware")

La complexité du logiciel exécuté par un équipement embarqué est très variable: d'une simple "boucle for" qui joue une séquence figée ... à un *operating system* (OS) complet faisant tourner "drivers" et multiples processus en parallèle.

Il existe des OS spécifiques aux contextes "embarqués:

OS temps-réel (Contiki, FreeRTOS, QNX...) ou non (Linux - Yocto, Android - Brillio, W10 for IoT...).

Plateformes \ Arduino (Genuino)

Architecture matérielle libre, à base de microcontrôleur Atmel,

programmation: langage dédié (simili C, basé sur <u>Wiring</u>) via des outils (IDE) open-source et cross-platform,

coût: < 50\$

plusieurs dizaines de modèles (différents form-factors et fonctions)









ref: https://www.arduino.cc/

Plateformes \ Raspberry Pl

"nano-ordinateur" (taille carte de crédit) à processeur ARM, à très bas coût. Support audio/video

lancement: 2012

cible: éducation / bricolage (DIY) / multi-media,

coût: 20\$ - 40\$



Raspberry Pi 2 Model B v1.2 Raspberry Pi 3 Model B Processor Chipset Broadcom BCM2837 64Bit Quad Core Processor Broadcom BCM2837 64Bit Quad Core Processor powered Single Board Computer running at 1.2GHz powered Single Board Computer running at 900MHz Processor Speed QUAD Core @1.2 GHz QUAD Core @900 MHz RAM 1GB SDRAM @ 400 MHz 1GB SDRAM @ 400 MHz Storage MicroSD MicroSD USB 2.0 4x USB Ports 4x USB Ports Max Power Draw/voltage 2.5A @ 5V 1.8A @ 5V **GPIO** 40 pin 40 pin Ethernet Port Built in No Built in Bluetooth LE No

ref: https://www.raspberrypi.org/

Plateformes \ BeagleBone (Texas Instruments)

mini-ordinateur, hardware open-source (CC) basse consommation

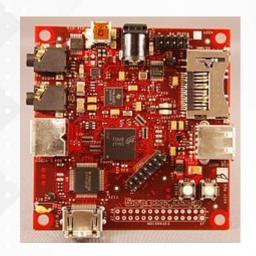
lancement: 2008

cible: éducation / bricolage (DIY)

coût: 70-160\$

OS: Linux et plus





ref: http://beagleboard.org/bone

Comparatif

Name	Arduino Uno	Raspberry Pi	BeagleBone	
Model Tested	R3	Model B	Rev A5	
Price	\$29.95	\$35	\$89	
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"	
Processor	ATMega 328	ARM11	ARM Cortex-A8	
Clock Speed	16MHz	700MHz	700MHz	
RAM	2KB	256MB	256MB	
Flash	32KB	(SD Card)	4GB(microSD)	
EEPROM	1KB			
nput Voltage	7-12v	5v	5v	
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)	
Digital GPIO	14	8	66	
Analog Input	6 10-bit	N/A	7 12-bit	
PWM	6		8	
TWI/I2C	2	1	2	
SPI	1	1	1	
JART	1	1	5	
Dev IDE	IDLE, Scrato Arduino Tool Squeak/Linu		Python, Scratch, Squeak, Cloud9/Linux	
Ethernet	N/A	10/100	10/100	
USB Master	N/A	2 USB 2.0	1 USB 2.0	
Video Out	N/A	HDMI, Composite	N/A	
Audio Output	N/A	HDMI, Analog	Analog	

Brand	Models	CPU	RAM	+	Price	Туре	Connectivity
Arduino	20+ and many clones (Spark, Intel, and so on)	ATmega, 8–64 MHz, Intel Curie, Linino	16 KB- 64 MB	Largest community	~30 USD	RTOS, Linux, hobbyists	Pluggable exten- sion boards (Wi- Fi, GPRS, BLE, Zigbee, and so on)
Raspberry Pi	A, A+, B, B+, 2, 3, Zero	ARMv6 or v7, 700 MHz -1.2 GHz	256-1 GB	Full Linux, GPU, large community	~5-35 USD	Linux, hobbyists	Ethernet, extension through USB, BLE (Pi3)
Intel	Edison	Intel Atom 500 MHz	1 GB	X86, full Linux	~50 USD	Linux, hobbyist to industrial	Wi-Fi, BLE
BeagleBoard	BeagleBone Black, X15, and so on	AM335x 1 GHz ARMv7	512 MB- 2 GB	Stability, full Linux, SDK	~50 USD	Linux, hobbyist to industrial	Ethernet, extension through USB and shields
Texas Instru- ments	CC3200, SoC loT, and so on	ARM 80 MHz, etc.	from 256 KB	Cost, Wi-Fi	<10 USD	RTOS, industrial	Wi-Fi, BLE, Zigbee
Marvell	88MC200, SoC loT, and so on	ARM 200 MHz, etc.	from 256 KB	Cost, Wi-Fi, SDK	<10 USD	RTOS, indus- trial	Wi-Fi, BLE, Zigbee
Broadcom	WICED, and so on (also at the heart of the Raspberry PIs)	ARM 120 MHz, and so on	from 256 KB	Cost, Wi-Fi, SDK	<10 USD	RTOS, industrial	Wi-Fi, BLE, Zig- bee, Thread

Source: Building the Web of Things: book.webofthings.io Creative Commons Attribution 4.0

Plateformes \ Android (Google)

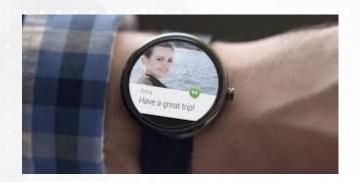
Système d'exploitation basé sur noyau Linux.

cible principale: smartphones, tablettes

OS le plus utilisé dans le monde (>80% des smartphones)

version "Android Wear" (2014) dédiée au wearables et montres connectées





Plateformes \ mbed (ARM)

Système d'exploitation pour équipements basés sur processeur ARM Cortex-M (32bits)

Programmation: C/C++, SDK, IDE en ligne

OS "temps réel"

plateforme cloud / services...





Plateformes \ Contiki (open-source)

Système d'exploitation open-source, pour microcontrôleurs Développement C, programmation événementielle (séquentiel)

Contiki

The Open Source OS for the Internet of Things

Hello world

```
/* Declare the process */
PROCESS(hello_world_process, "Hello world");

/* Make the process start when the module is loaded */
AUTOSTART_PROCESSES(&hello_world_process);

/* Define the process code */
PROCESS_THREAD(hello_world_process, ev, data) {
    PROCESS_BEGIN(); /* Must always come first */
    printf("Hello, world!\n"); /* Initialization code goes here */
    while(1) { /* Loop for ever */
        PROCESS_WAIT_EVENT(); /* Wait for something to happen */
    }
    PROCESS_END(); /* Must always come last */
}
```



ref: www.contiki-os.org / http://fr.slideshare.net/DingxinXu/contiki-introduction-i

Contraintes d'un projet "hardware"

- Cycles de développements plus long
- Réutilisation de composants >> forte dépendance aux fournisseurs
- Tests manuels + besoin de qualifier pour des conditions d'utilisation variées
- Besoin de certification pour usage radio, grand public, etc.
- Besoin d'anticiper sourcing, procédés de fabrication en masse, QA, SAV
- ⇒ Coût et durée beaucoup plus importants que pour du soft.
- ⇒ Faire gain à partir de "plateformes" existantes, et réutiliser des composants maîtrisés et éprouvés.



Conclusion: les grands enjeux de la conception embarquée

Fonctions

Quelles grandeurs à mesurer? Sur quoi agir? Avec quoi s'interfacer?

Autonomie

Source d'énergie externe? Quelle durée de vie sans maintenance/recharge?

Coût (\$)

Le cas d'application est-il rentable?

Conditions opérationnelles

Quel environnement de fonctionnement? Quel contexte légal (certifications...)?

Sécurité

Manipule-t-on des informations personnelles ou sensibles?

Quel est l'impact d'un dysfonctionnement?



Merci.

DES QUESTIONS?

charly@rtone.fr



Differences between Hardware and Software Development

- Software is easier to change than hardware. The cost of change is much higher for hardware than for software.
- Software products evolve through multiple releases by adding new features and re-writing existing logic to support the new features. Hardware products consist of physical components that cannot be "refactored" after manufacturing, and cannot add new capabilities that require hardware changes.
- Designs for new hardware is often based upon earlier-generation products, but commonly rely on next-generation components not yet present.
- Hardware designs are constrained by the need to incorporate standard parts.
- Specialized hardware components can have much longer lead times for acquisition than is true for software.
- Hardware design is driven by architectural decisions. More of the architectural work must be done up front compared to software products.
- The cost of development for software products is relatively flat over time. However, the cost of hardware development rises
 rapidly towards the end of the development cycle. Testing software commonly requires developing thousands of test cases.
 Hardware testing involves far fewer tests.
- Software testing is done by specialized Quality Assurance (QA) engineers, while hardware testing is commonly done by the
 engineers who are creating the product.
- Hardware must be designed and tested to work over a range of time and environmental conditions, which is not the case for software.
- Hardware development incorporates four parallel, synchronized projects:
- 1) The detailed design of the manufacturable product 2) the manufacturing process and tooling 3) the test and inspection process and equipment; and 4) the supply chain for purchased parts. In software development, the detailed design is the product, and production deployment consists of moving the product into a context where it can be used. Due to many of the above factors, it is possible to make major changes in direction for a planned software-product upgrade in mid-development, without massive disruption and waste. Attempts to make such changes in hardware development come at a much higher cost, in terms of sunk costs wasted, and shipping schedules postponed. As a result, major changes must either be deferred to a future product upgrade, or are done when an assessment is made that the impact is justified by the magnitude of the benefits. Ready to learn more about Agile for Hardware development?

SENTIR = capter de l'information

via capteurs de différentes natures, souvent fournissent une information analogique qui doit être convertie en donnée numérique. Parfois s'appuie sur systèmes externes, ex: GPS.

AGIR = agir sur l'environnement

via « actionneurs »,

souvent nécessite de s'appuyer sur des système complémentaires au cœur (électronique basse tension): monde de l' « électronique de puissance ».

INTERAGIR = interface utilisateur

un utilisateur à proximité de l'objet peut être notifié (sens: visuel, audio, vibration),

des « contrôles » peuvent lui être fournis

DECIDER

on trouve au cœur de l'objet un processeur / microcontrôleur qui exécute un programme en charge de l'orchestration des mesures/actions et communications, ce programme s'appuie sur des mémoires (ROM/RAM/Flash) pour stocker un état.

COMMUNIQUER

l'objet transmet de l'information à distance ou bien est télé-opéré



les enjeux de la conception embarquée

- Fonctionnel: choix périphérique, puissance, mode de communication,
- Autonomie: mode alimentation, consommation des différents composants, fréquence de leur utilisation

(ex: limiter les communications)

- Contraintes opérationnelles: compatibilité avec environnement/usage ciblé (cf. Indices de Protection, certification electro-magnetique)
- Sécurité: protéger les secrets
- Coût (\$)



les enjeux de la conception embarquée

Développer un nouveau « hardware » est long et couteux,

De nombreuses solutions « ouvertes » permettent un prototypage à moindre coût,

• La réutilisation de composants déjà pré-intégré (ex: « board » complet CPU+mémoire+comm...) limite la partie « custom » à son stricte nécessaire.



INGENIEUR EN DEVELOPPEMENT SYSTEMES EMBARQUES H/F

Studiel **** 2 avis - Sophia Antipolis (06)

45 000 € - 60 000 € par an - CDI

En étroite collaboration avec la Direction de la société, l'ingénieur participera au développement des solutions logicielles embarquées: - Développement du logiciel embarqué des produits en environnement GNU/Linux, C/C++ - Interface avec le développement électronique

Profil recherché: De formation BAC+5 et une forte expérience de développement en environnement Linux embarqué de 8 ans minimum. Savoir-faire La maitrise de l'environnement technologique suivant: - Algorithmes C++ - La couche physique LTE idéalement ou bien celle d'autres protocoles radio, - HW (Power PC) et en traitement du signal. Savoir-être - Autonome, vue globale des tâches à réaliser, dans le but de piloter les projets. - Bon niveau d'anglais

- De formation ingénieur(e) ou Master dans le domaine des télécoms et/ou du logiciel embarqué, vous avez entre
 5 et 15 ans d'expérience en développement informatique : programmation C/C++, conception objet, test,
 intégration
- Connaissance de Linux et des logiciels temps réels.
- Connaissance des protocoles du monde IP (mac /VLAN/IP V4/IP V6/UDP/TCP/FTP/DHCP/...)
- Connaissance des environnements de développement Linux/Unix, GIT.
- Connaissance des scripts (Shell, Python)
- La connaissance en architectures processeur Intel/ARM serait un plus apprécié.

Profil

Diplômé d'une école d'ingénieur/Master en robotique/mécatronique/aéronautique ou équivalent, vous êtes avant tout un passionné.

Vous avez 2 à 5 ans d'expérience en mécatronique, idéalement dans le domaine des drones.

Vous êtes familier avec les procédés de prototypage rapide et avez idéalement des notions d'industrialisation.

Vous maîtrisez Solidworks

Vous maîtrisez les environnements Linux embarqués temps réel, C++ et python

Vous avez des notions d'asservissement, de traitement du signal et d'intégration de capteurs.



INGENIEUR LOGICIEL EMBA

74300 Cluses, France . Temps complet

Connaissances et expériences souhaitées :

- Maîtrise du cycle de développement Logiciel (Cycle en V et/ou Agile)
- Maîtrise de la programmation en langage C/C++
- Compétence en modélisation d'architecture logicielle et conception orientée objet (UML)
- Compétence en programmation pour plate-forme embarqué 8 ou 32 bits (ST, FREESCALE, ARM...)
- Utilisation d'outils de gestion de configuration (SVN...)
- Utilisation de systèmes d'exploitation (MicroC OS2, FreeRTOS...)
- Connaissance des bus courants (I2C, SPI, UART...)
- Bonne connaissance du monde temps réel, de l'embarqué et de l'automatisme
- Connaissance des réseaux (IPV6, JSON, COAPP…) et des objets connectés.
- Notions sur les contraintes d'un protocole radio (ZigBee, Bluetooth Low Energy)