Conception de solutions embarquées temps-réel

Animé par Sylvain Labasse

INTRODUCTION

En fin de module, vous saurez…

Concevoir l'électronique d'un système embarqué simple

Interagir logiciellement avec des E/S numériques ou analogiques

Respecter des contraintes de temps par programmation

Sélectionner le système d'exploitation temps-réel adéquat

Paralléliser des traitements sur un système temps réel

Pre-Requis

Public

I2 dev.

Nécessaire

C ou langage dérivé du C (js, java, c#, php, …)

Support

Copie des slides

My Learning Box

Notes de cours

Correction des ateliers

L'ENVIRONNEMENT

Matériel

Mac ou PC sous Windows ou Linux

Logiciels

Compte TinkerCAD

FreeRTOS https://www.freertos.org/index.html

EVALUATION

Ateliers

A rendre sur MyLearningBox

QCM

Contenu

Vue d'ensemble

- Systèmes embarqués
- Chiffres
- Plateformes de prototypage
- Arduino

Rappels de C

- Processus de compilation
- Principales syntaxes
- Mécanisme de callback

Contenu (suite)

Bases d'électronique

- Lois de Kirchhoff
- Loi de Ohm
- Diviseur de tension

Entrées/sorties numériques

- Sorties numériques
- Machines à états
- Entrées numériques
- Circuits ouverts

Contenu (suite)

Entrées sorties analogiques

- Théorème de l'échantillonnage
- Entrées analogiques
- Sorties analogiques PWM

Périphériques et Shields

- Capteurs
- Actionneurs
- Bus I2C

CONTENU (SUITE ET FIN)

Système temps-réel

- Définition
- Contraintes
- FreeRTOS

FreeRTOS

- Hello world !
- Entrées/Sorties
- Tâches et timers
- Files, sémaphores

VUE D'ENSEMBLE

OBJECTIFS

Distinction d'un système embarqué

Aperçu du marché

Découverte des systèmes de prototypage

VUE D'ENSEMBLE

➔ Systèmes embarqués

Chiffres

Plateformes de prototypage

Arduino

Synthèse

SYSTEME EMBARQUE

Caractéristiques

Sous-système informatique

Partie intégrante d'un système électrique/mécanique

88% des processeurs produits

Souvent temps réel

Contraintes

Consommation/Autonomie

Fiabilité/Résistance

Coût, Taille

VUE D'ENSEMBLE

- ✓ Systèmes embarqués
- ➔ Chiffres
- Plateformes de prototypage
- Arduino
- Synthèse

Chiffres

Global

2019 – 250,72 Md $

2027 – 1463,19 Md $[1]

Tendances

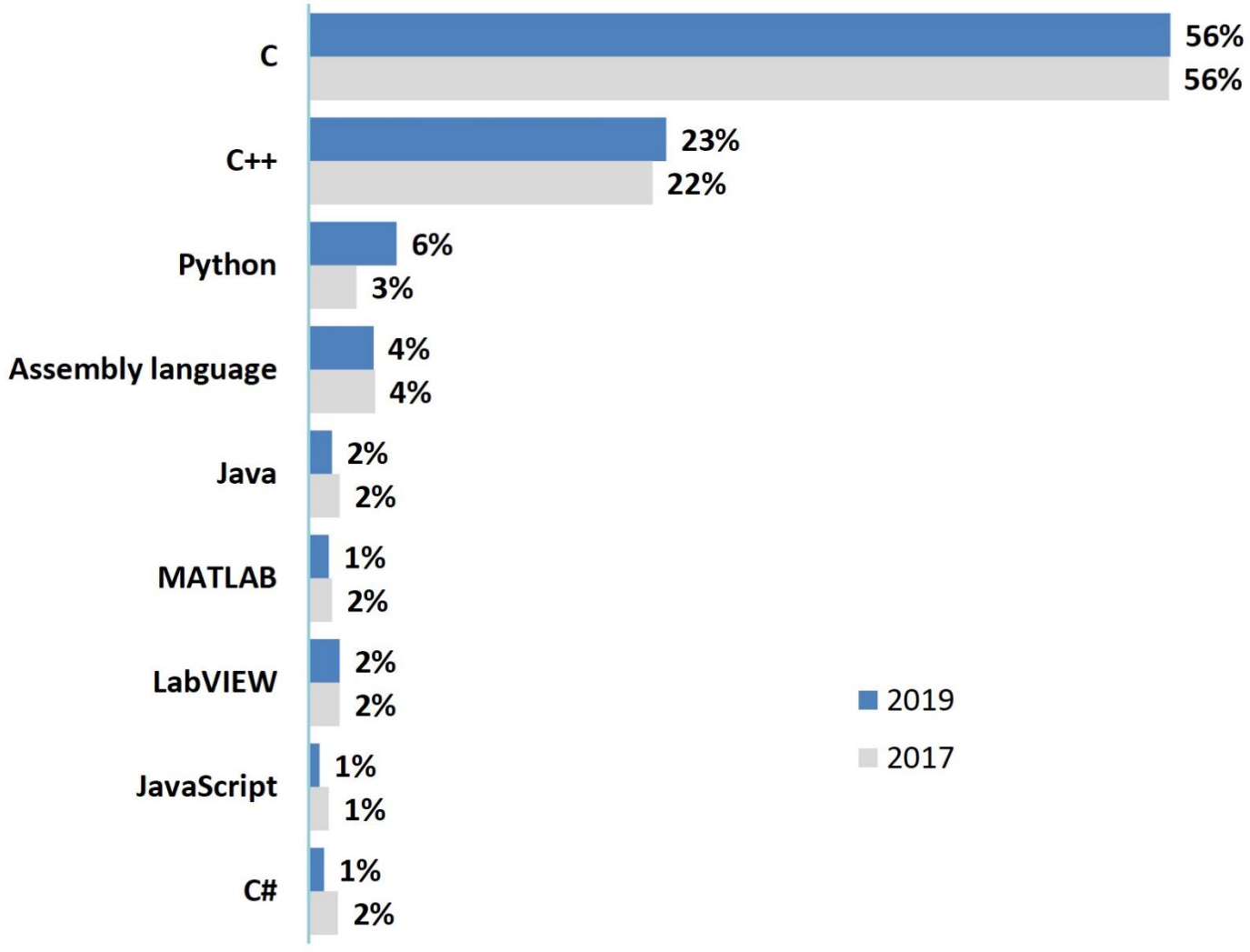
Augmentation ventes électronique grand public

Offre/Demande de produits connectés

Automatisation industrielle

Domotique et optimisation énergétique des bâtiments.

[1] https://www.fortunebusinessinsights.com/industry-reports/infographics/internet-of-things-iot-market-100307


C
56%
56%
C++
23%
22%
Python
6%
3%
Assembly language
4%
4%
Java
2%
2%
MATLAB
1%
2%
LabVIEW
2%
2%
JavaScript
1%
1%
C#
1%
2%
2019
2017

VUE D'ENSEMBLE

- ✓ Systèmes embarqués
- ✓ Chiffres
- ➔ Plateformes de prototypage
  - Arduino
  - Synthèse

PLATEFORMES DE PROTOTYPAGE

OpenSource

Arduino : 17 versions officielles

https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems

RaspberryPI : Modèles A/B, 2, 3, 3B+, 4B

Equivalents

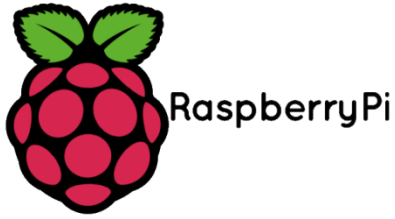
Texas Instrument : BeagleBone, LaunchPad

STMicroelectronics : STM32

Microchip / PIC

Nombreuses versions

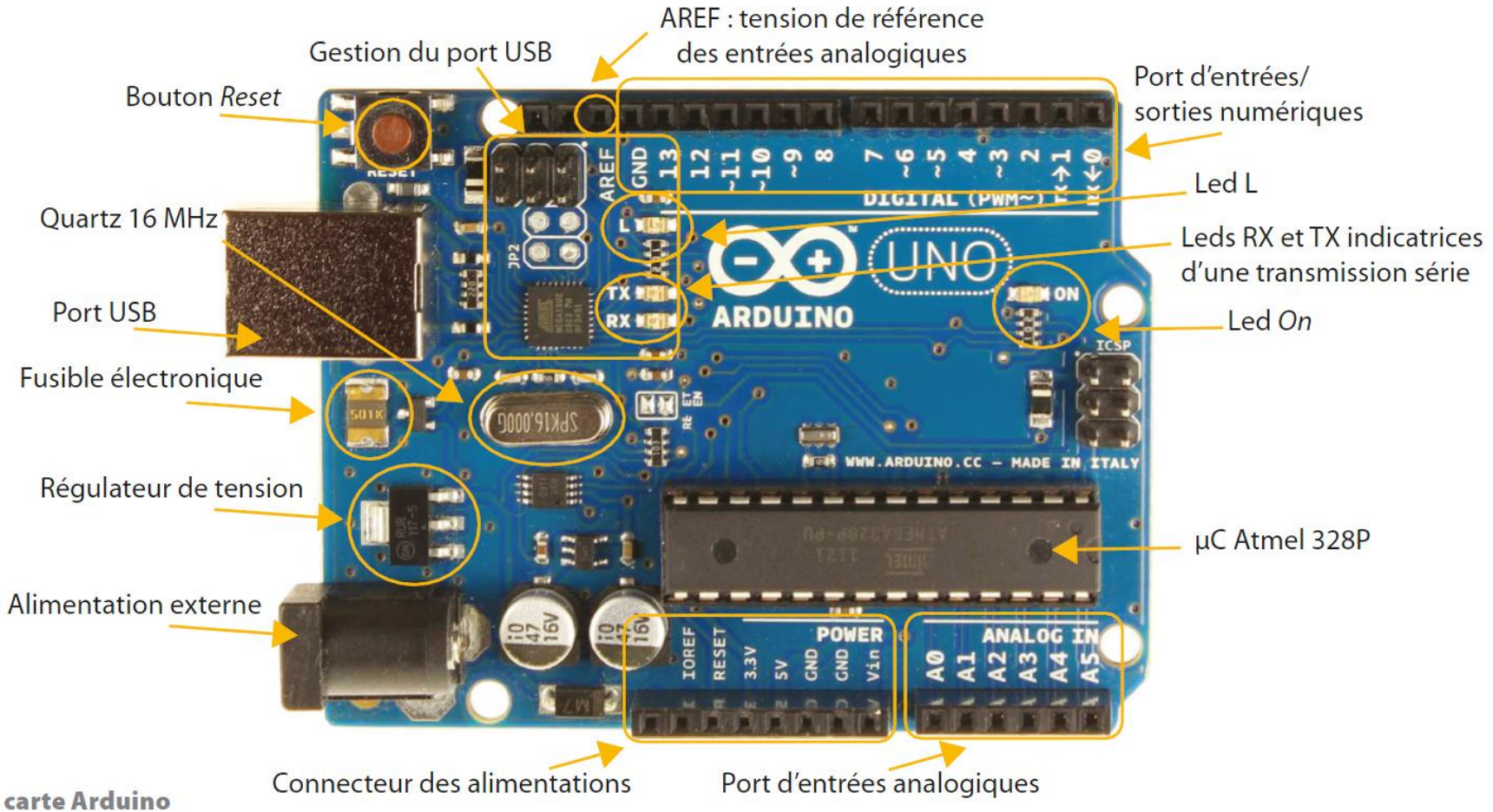
ARDUINO VS RASPBERRYPI

		
OS	Aucun	Linux => Boot
Alimentation	5V élémentaire = piles	5V alimentation spéc.
Réseau	Shield(s) nécessaire(s)	Natif
E/S analogiques	Natif	Librairies Arduino comme contrôleur

VUE D'ENSEMBLE

- ✓ Systèmes embarqués
- ✓ Chiffres
- ✓ Plateformes de prototypage
- ➔ Arduino
- Synthèse

ARDUINO



1 La carte Arduino

Arduino (suite)

Objectif

Plateforme matérielle opensource

Orchestration E/S numériques et analogiques

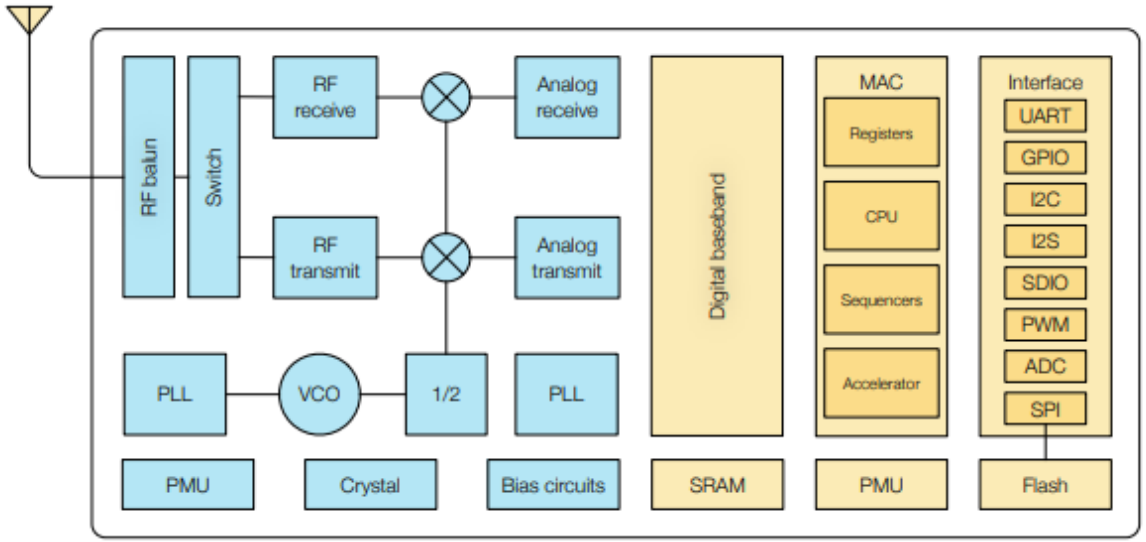
Shields : PCB pour Arduino

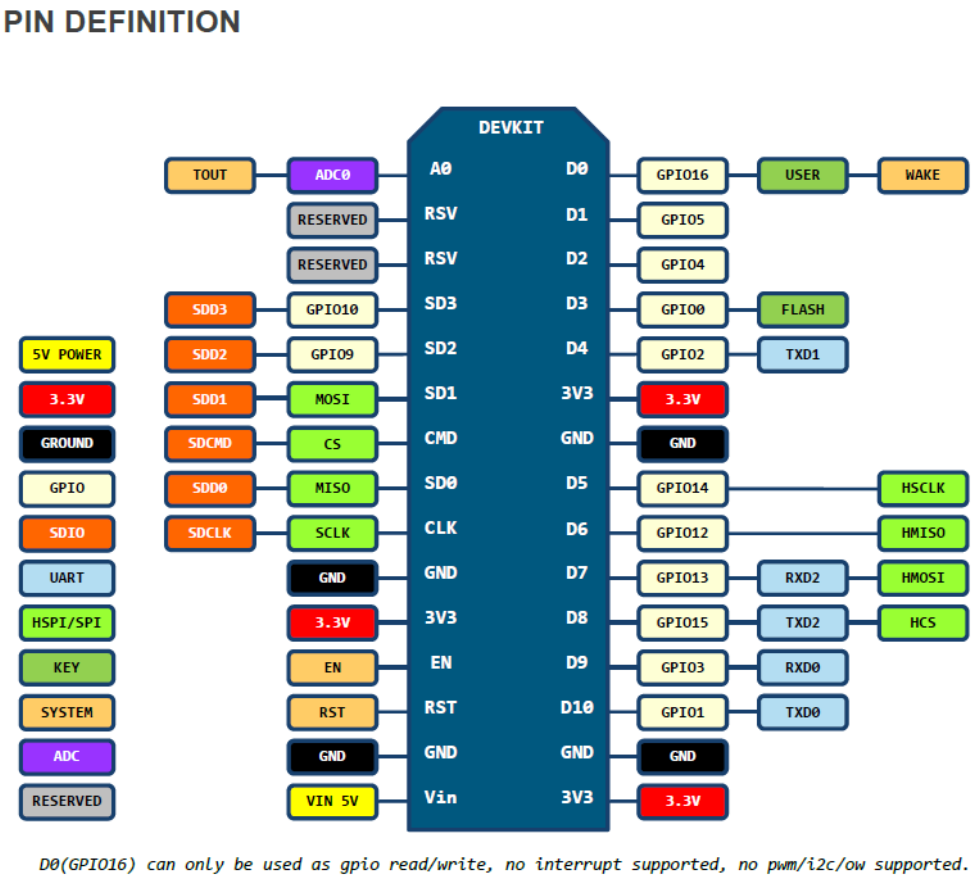
Structure d'un programme

setup() : Initialisation du programme, Configuration E/S

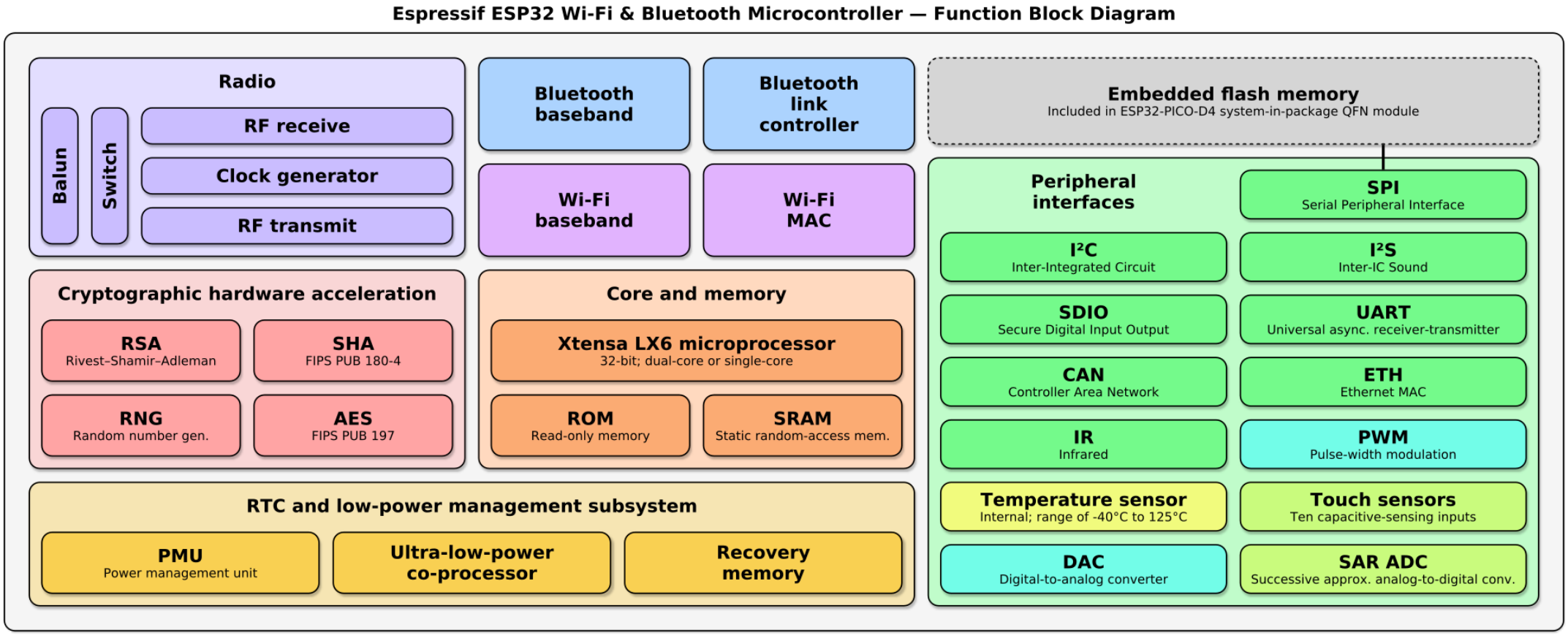
loop() : Opérations répétées indéfiniment

MICROCONTROLEURS ESP 8266





ESPRESSIF 32



RESUME

Distinction d'un système embarqué

Aperçu du marché

Découverte des systèmes de prototypage

RAPPELS DE C

OBJECTIFS

Subtilités de la compilation en C

Syntaxes utiles au projet

Fonctions et leurs références

RAPPELS DE C

➔ Processus de compilation

Principales syntaxes

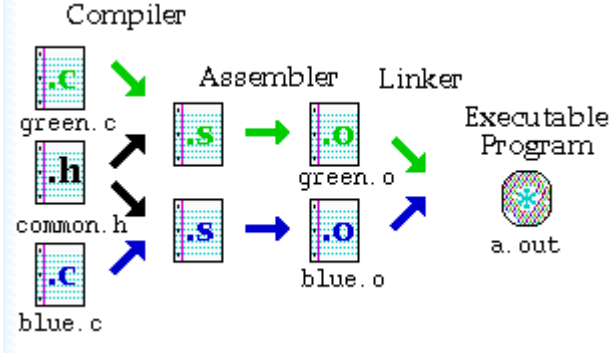
Mécanisme de callback

Synthèse

PROCESSUS DE COMPILATION

Précompilation

#include, #define, #ifdef #else #endif



Compilation

Validation syntaxique, Traduction de chaque fichier en .o

Edition des liens

Résolution des références entre fichiers .o

RAPPELS DE C

- ✓ Processus de compilation
- ➔ Principales syntaxes
- Mécanisme de callback
- Synthèse

PRINCIPALES SYNTAXES

Types et variables

char, short, int, long, bool

ou int8/uint8, int16/uint16, int32/uint32

Opérateurs

+, -, *, /, %, <, >, <=, >=, ==, !=, &&, ||, !, +=, ++

Structures

struct Type { int champ1 ; float champ2 ; } ;

PRINCIPALES SYNTAXES

Tests

```
if(condition) { … } else { … }
switch(expr) { case … : … break ; default : … break ; }
```

Boucles

```
for(int i=0 ; i<10 ; i++) { … }
while(condition) { … }
do { … } while(condition) ;
```

RAPPELS DE C

- ✓ Processus de compilation
- ✓ Principales syntaxes
- ➔ Mécanisme de callback
- Synthèse

Fonctions

Fonctions

type nomFonction(int param1, float param2)

Suvi de ; (déclaration) ou { … } (définition)

Si « type * param » alors mettre & devant l'argument

Modules

MonModule.h contient les déclarations et types

MonModule.c contient les définitions

CALLBACKS

Principe

Passage d'une fonction en paramètre

Equivalent aux lambdas, closures, delegués, …

Syntaxe

Nom de la fonction sans ()

Ex : xTaskCreate(**pointEntreeTache**, "nom", 1000, NULL, 1, NULL) ;

RAPPELS DE C

- ✓ Processus de compilation
- ✓ Principales syntaxes
- ✓ Mécanisme de callback
- ➔ Synthèse

Atelier 1 – Signal Morse

Sujet

Proposer un programme qui émet un signal morse au travers de la led 13.

Compétence

Interagir logiciellement avec des E/S numériques ou analogiques.

Resume

Subtilités de la compilation en C

Syntaxes utiles au projet

Fonctions et leurs références

BASES D'ELECTRONIQUE

OBJECTIFS

Lois fondamentales

Tensions et intensités dans un circuit

Maîtrise de la répartition des flux

BASES D'ELECTRONIQUE

➔ Loi de Kirchhoff

Loi d'ohm

Diviseur de tension

Résumé

Lois de Kirchhoff – Mailles

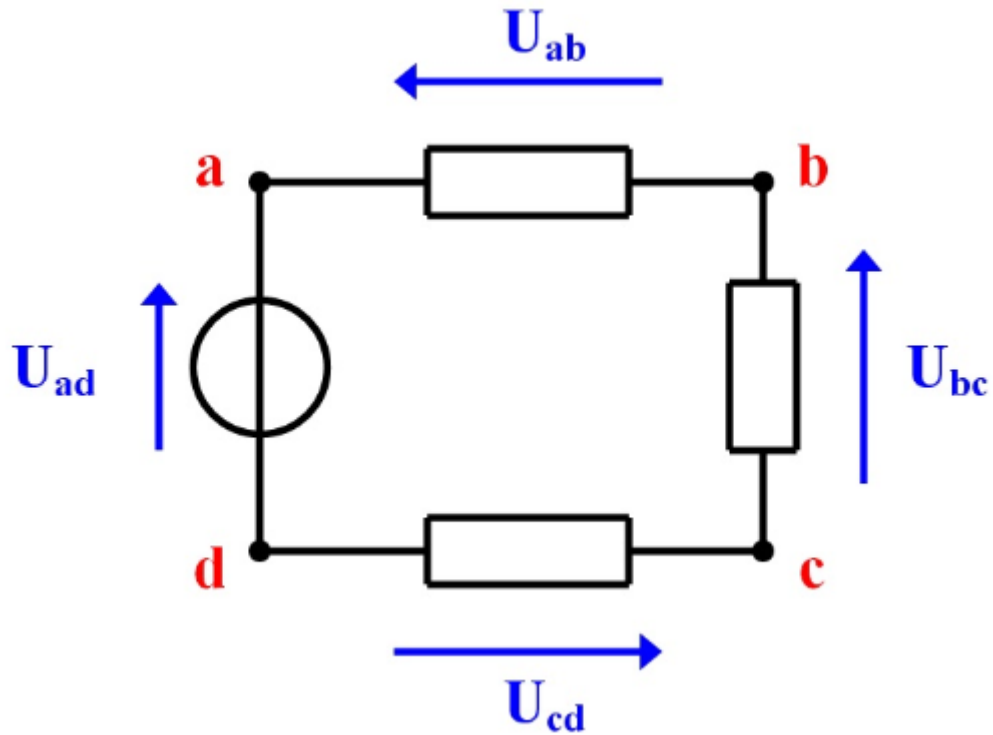
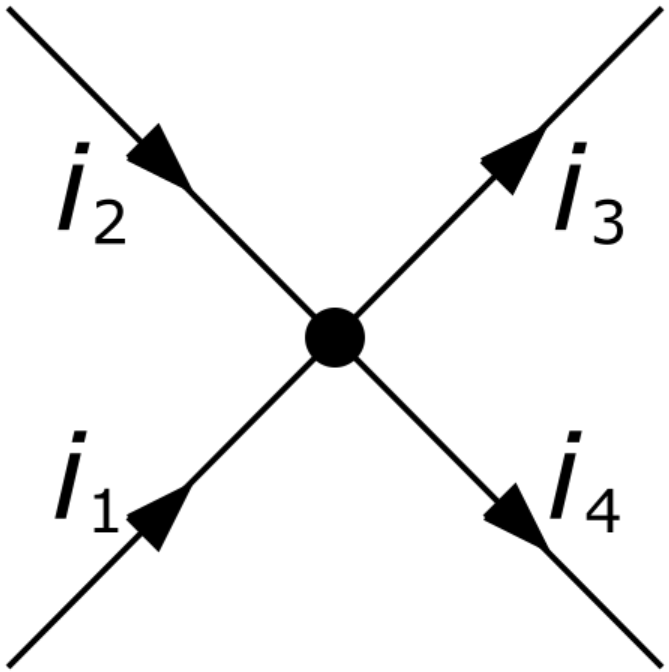


Figure 1 - source Wikimedia

Lois de Kirchhoff – Nœuds



$$i_1 + i_2 = i_3 + i_4$$

Figure 2 - source wikipedia

BASES D'ELECTRONIQUE

- ✓ Loi de Kirchhoff
- ➔ Loi d'ohm
- Diviseur de tension
- Résumé

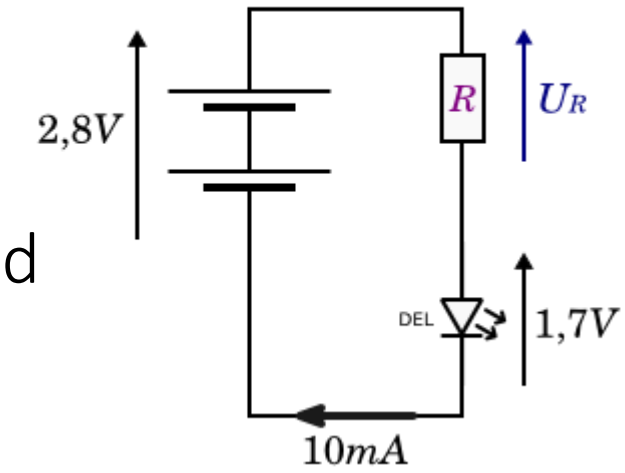
Loi d'Ohm

Formule

U = R.I

Utilité

Tension dans une led



$$U_R = 2,8V - 1,7V$$
$$= 1,1V$$

$$R = \frac{1,1V}{10mA} = 110\Omega$$

Figure 3 - Calcul d'une tension dans une LED (src: chicoree.fr)

BASES D'ELECTRONIQUE

- ✓ Loi de Kirchhoff
- ✓ Loi d'ohm
- ➔ Diviseur de tension
- Résumé

DIVISEUR DE TENSION

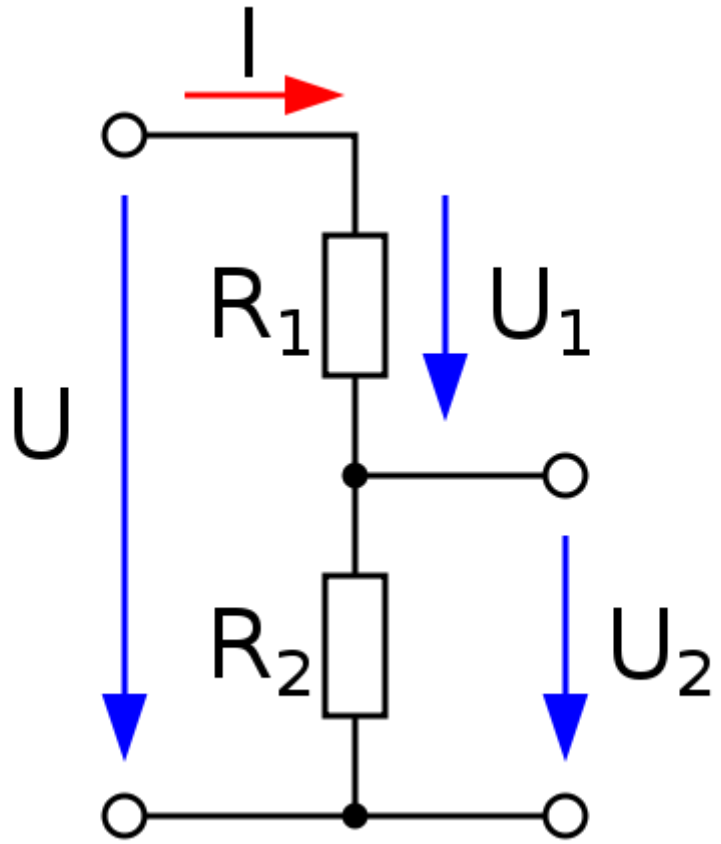


Figure 4 - source Wikipedia

BASES D'ELECTRONIQUE

- ✓ Loi de Kirchhoff
- ✓ Loi d'ohm
- ✓ Diviseur de tension
- ➔ Résumé

RESUME

Lois fondamentales

Tensions et intensités dans un circuit

Maîtrise de la répartition des flux

ENTREES/SORTIES NUMERIQUES

OBJECTIFS

Interaction avec des signaux numériques

Commandes numériques et leds

Intégration de capteurs et boutons poussoirs

ENTREES/SORTIES NUMERIQUES

➔ Sorties numériques

Machines à états

Entrées numériques

Circuits ouverts

Synthèse

Sorties numeriques

Configuration

pinMode(broche, OUTPUT)

Ecriture en sortie

digitalWrite(broche, HIGH/LOW) ;

tone(broche, freq[, durée]) / noTone(broche) exclusif

ENTREES/SORTIES NUMERIQUES

- ✓ Sorties numériques
- ➔ Machines à états
- Entrées numériques
- Circuits ouverts
- Synthèse

MACHINE A ETAT

Principe

Etats du programme avec comportement spécifiques

Implémentation

Procédural : Entier/enum contenant l'état + switch(etat) { }

Objet : Design pattern Etat

ENTREES/SORTIES NUMERIQUES

- ✓ Sorties numériques
- ✓ Machines à états
- ➔ Entrées numériques
- Circuits ouverts
- Synthèse

ENTREES NUMERIQUES

Configuration

pinMode(broche, INPUT/INPUT_PULLUP)

Lecture en entrée

int val = digitalRead(broche);

HIGH si > 3V, LOW si < 0.5, indéterminé sinon

INPUT_PULLUP « inverse » le signal

Sensible : Résistance 10K si reliée au 5V

ENTREES/SORTIES NUMERIQUES

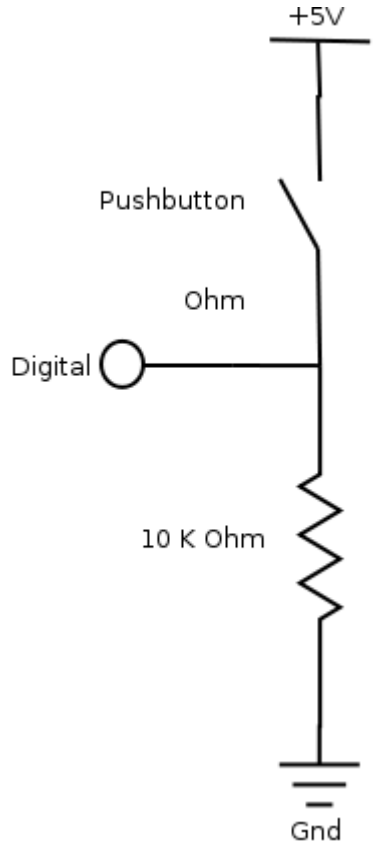
- ✓ Sorties numériques
- ✓ Machines à états
- ✓ Entrées numériques
- ➔ Circuits ouverts
- Synthèse

Circuits ouverts

Montage

Bouton poussoir relâché = circuit ouvert

Nécessité de fermer le circuit avec résistance



Alternative Arduino

pinMode(broche, INPUT_PULLUP)

INPUT_PULLUP « inverse » le signal

Sensible : Résistance 10K si reliée au 5V

ENTREES/SORTIES NUMERIQUES

- ✓ Sorties numériques
- ✓ Machines à états
- ✓ Entrées numériques
- ✓ Circuits ouverts
- ➔ Synthèse

Atelier 2 – Compteur binaire

Sujet

…

Compétences

Concevoir l'électronique d'un système embarqué simple
Interagir logiciellement avec des E/S numériques
Respecter des contraintes de temps par programmation

RESUME

Interaction avec des signaux numériques

Commandes numériques et leds

Intégration de capteurs et boutons poussoirs

ENTREES/SORTIES ANALOGIQUES

OBJECTIFS

Nécessités liées aux signaux échantillonnés

Interprétation des signaux analogiques

Production d'un signal analogique

ENTREES/SORTIES ANALOGIQUES

➔ Théorème de l'échantillonnage

Entrées analogiques

Sorties analogiques PWM

Résumé

ECHANTILLONNAGE

Principe

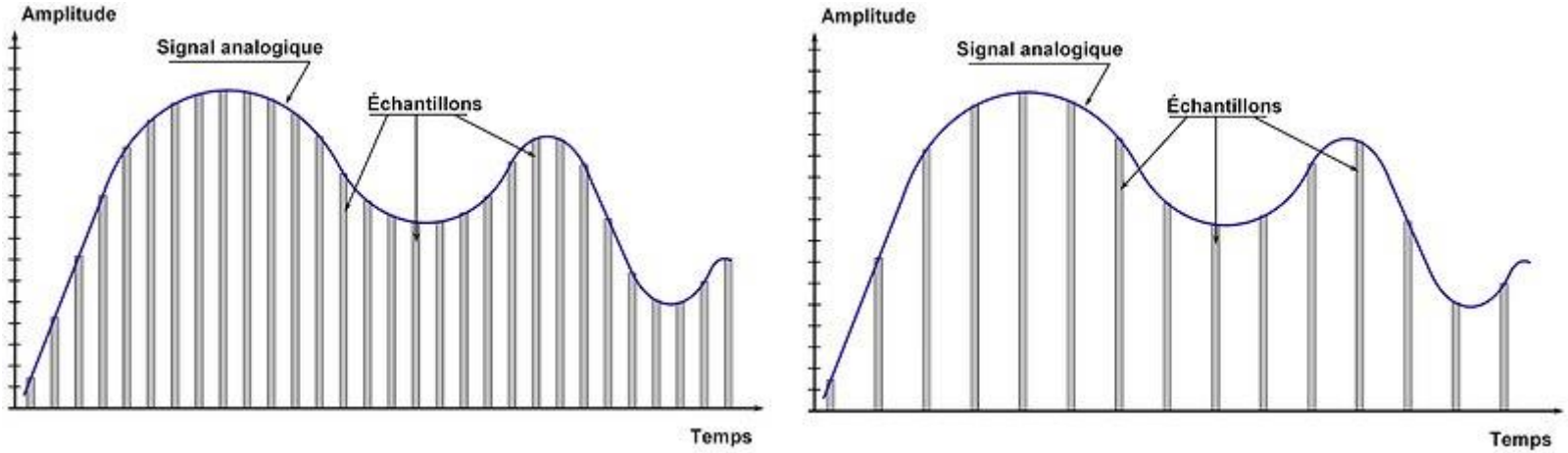


Figure 5 - src : https://www.cinenow.fr/tutoriels-guides/2094-la-notion-de-frequence-dechantillonnage

Théorème de Nyquist-Shannon

Fréquence d'échantillonnage min. = 2 x fréq. Max du signal

ENTREES/SORTIES ANALOGIQUES

- ✓ Théorème de l'échantillonnage
- ➔ Entrées analogiques
- Sorties analogiques PWM
- Résumé

Entrees analogiques

Principe

Broches A0 à A5 (A15 sur AtMega)

500µs pour une lecture

Lecture

Valeur 0 (0v) à 2^{10}-1 (5V)

analogRead(A0)

Fonction map(val, 0, 1023, min, max)

ENTREES/SORTIES ANALOGIQUES

- ✓ Théorème de l'échantillonnage
- ✓ Entrées analogiques
- ➔ Sorties analogiques PWM
- Résumé

Sorties analogiques PWM

Principe

Pas de DAC sur Arduino

Pulse Width Modulation

En pratique

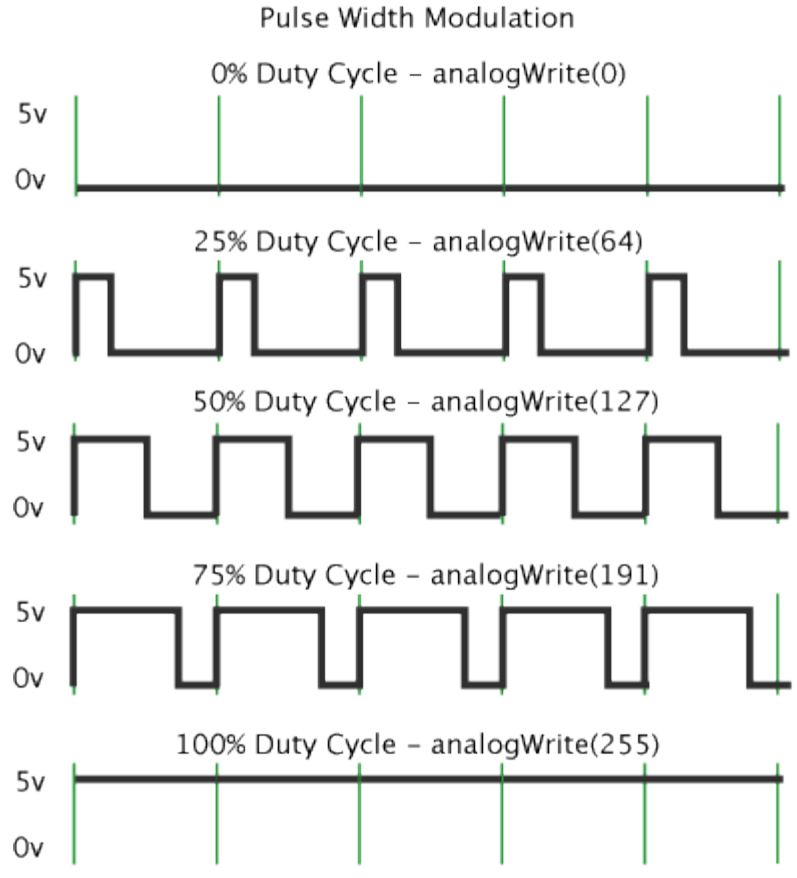
Broches 5, 6 (980hz)

Broches 3, 9, 10, 11 (490Hz)

0 (0%) – 255 (100%)

pinMode(5, OUTPUT) ;

analogWrite(5, 128);



ENTREES/SORTIES ANALOGIQUES

- ✓ Théorème de l'échantillonnage
- ✓ Entrées analogiques
- ✓ Sorties analogiques PWM
- ➔ Résumé

RESUME

Nécessités liées aux signaux échantillonnés

Interprétation des signaux analogiques

Production d'un signal analogique

PERIPHERIQUES

OBJECTIFS

Classification des types de capteurs/actionneurs

Bases de traitement du signal

Principe des bus séries et notamment I2C

PERIPHERIQUES

➔ Capteurs

Actionneurs

Bus I2C

Synthèse

Capteurs

Valeur unique

Potentiomètre, photosensible, température

Dérivés : Joystick analogique, …

Valeur unique ondulatoire

Microphone, détection de vibration

Théorème de Nyquist-Shannon : Fréq. d'échantillonage

Transformée de Fourier : Spectre de fréquences

Valeurs multiplexées = Sérialisation

1 voie : bits de synchro / 2 voies : données+horloges (cf I2C)

PERIPHERIQUES

- ✓ Capteurs
- ➔ Actionneurs
- Bus I2C
- Synthèse

Actionneurs

Numériques simples

HIGH/LOW : ex: led

PWM : Ex: Led RGB classique (3 valeurs)

Fréquence (tone) : Buzzer

Numériques complexes

Plusieurs valeurs numériques, parallèle/série

Ex : Servo-moteur, Led RGB chainable, …

Analogiques

Ex : Haut-parleur

PERIPHERIQUES

- ✓ Capteurs
- ✓ Actionneurs
- ➔ Bus I2C
- Synthèse

Bus I2C

Bus

Canal de communication auquel se greffent des/les composants d'un système informatique.

Caractéristiques

1982, V1 en 1992, v6 en 2014

3 fils : Masse, SDA (données), SDC (horloge)

100kbps à 5Mbps

Arduino : wire.h

PERIPHERIQUES

- ✓ Capteurs
- ✓ Actionneurs
- ✓ Bus I2C
- ➔ Synthèse

Atelier 3 – Fade In / Fade Out

Sujet

…

Compétences

Concevoir l'électronique d'un système embarqué simple
Interagir logiciellement avec des E/S analogiques
Respecter des contraintes de temps par programmation

RESUME

Classification des types de capteurs/actionneurs

Bases de traitement du signal

Principe des bus séries et notamment I2C

SYSTEME TEMPS-REEL

Objectifs

Vision des exigences temps réel

Classification selon contraintes

Implications pour le développeur

Plateforme utilisée dans ce cours

SYSTEME TEMPS-RÉEL

➔ Définition

Contraintes

FreeRTOS

Synthèse

Definition

Critères

Respect de **contrainte temps** aussi important que **résultat**

Domaines

Loisir : Jeu vidéo, Gyropode, Réalité augmentée, …

Communication : VoIP, Visioconférence, Réseau mobile

Industrie/Energie : Automatisation, Régulation, …

Transport : Aéronautique DO-178C, Rail, automobile

Médical : ISO62304

Taxonomie

Temps-réel dur

Tolérance d'aucune variation des contraintes de temps

Non-respect de contrainte de temps = dysfonctionnement

Conséquences critiques/vitales : Transport, Centrale, …

Temps-réel souple

Non respect des contraintes temps = inutilisable / pénible

Conséquences fonctionnelles : Audio/vidéo, jeux, …

SYSTEME TEMPS-RÉEL

- ✓ Définition, Taxonomie
- ➔ Contraintes
- FreeRTOS
- Synthèse

CONTRAINTES DE PROGRAMMATION

Performance

Multitâches => Scheduler, exigences + priorités

Mécanismes de synchronisation

Mémoire

Statique vs dynamique

Garbage collector

SYSTEME TEMPS-RÉEL

- ✓ Définition, Taxonomie
- ✓ Contraintes
- ➔ FreeRTOS
- Synthèse

FreeRTOS

Système d'exploitation temps-réel dur

- Tâche = Thread
- Ouvert / License MIT

Fonctionnalités

- Faible encombrement cpu/mémoire
- Ordonnanceur préemptif/coopératif, priorité/échéance
- Gestion avancée des interruptions
- Mode faible consommation

FreeRTOS

Atouts

Communauté et ressources

a:FreeRTOS (Amazon) pour IoT AWS

Utilisation commerciale libre

Personnalisation du noyau : FreeRTOSConfig.h

Marché

20% du marché

Contre 22% embedded linux

19% os maison, 13% androïd

Most Used	World	Americas	EMEA	APAC
Embedded Linux	31%	32%	31%	**26%**
FreeRTOS	27%	25%	24%	**37%**
Android	14%	12%	**10%**	**26%**

SYSTEME TEMPS-RÉEL

- ✓ Définition
- ✓ Contraintes
- ✓ FreeRTOS
- ➔ Synthèse

Resume

Vision des exigences temps réel

Classification selon contraintes

Implications pour le développeur

Plateforme utilisée dans ce cours

FREE RT OS

Objectifs

Compilation et test d'un programme

Utilisation des E/S numériques et analogiques

Réaction aux évènements matériels

Lancement et synchronisation de tâches et timers

FREE RT OS

➔ Hello world !

Entrées/sorties

Tâches et timers

Files, sémaphores

Synthèse

HELLO WORLD !

Code

Via putty :

cp -R ~/esp-rt/ESP8266_RTOS_SDK/examples/get-started/hello_world/ ~/mon_projet

Via VSCode + extension SSH FS :

Dans main/app_main.c, ajouter printf("Hello world !") ;

Construction/Test

cd ~/mon_projet

make

make flash monitor

Ctrl+$

FREE RT OS

- ✓ Hello world !
- ➔ Entrées/sorties
- Tâches et timers
- Files, sémaphores
- Synthèse

Entrees/Sorties "driver/gpio.h"

Constantes

GPIO_Pin_0..16/All combinables avec |

GPIO_NUM_0..16 pour les valeurs de broche

Configuration

```
gpio_config_t c ;
c.mode=GPIO_MODE_DISABLE/INPUT/OUTPUT ;
c.pin_bit_mask = GPIO_Pin_4 | GPIO_Pin_5 ;
c.intr_type = GPIO_INTR_DISABLE/POSEDGE/NEGEDGE/ANYEDGE ;
c.pull_down_en = c.pull_up_en = 0/1 ;
gpio_config(&c) ;
```

Entrees/Sorties

Lecture/Ecriture

```
int n = gpio_get_level(GPIO_NUM_x) ;
gpio_set_level(GPIO_NUM_x, 0/1) ;
uint16_t val; adc_read(&val); // tout broche 6 "driver/adc.h"
```

PWM "driver/pwm.h"

```
pwm_init(period, tab_pwm, taille, tab_broches) ;
pwm_start() ; pwm_stop(mask) ;
pwm_set_channel_invert(mask) ;
pwm_set_duties(tab); pwm_set_phases(tab) ; // -180..180
```

INTERRUPTIONS

Gestionnaires "driver/gpio.h"

gpio_install_isr_service(0) ;
gpio_isr_handler_add(GPIO_NUM_x, fun, NULL);
gpio_isr_handler_remove(GPIO_NUM_x) ;

Données

Mot clé volatile pour les globales
Appeler les méthodes « FromISR »
ex : xQueueSendFromISR(…)

FREE RT OS

- ✓ Hello world !
- ✓ Entrées/Sorties
- ➔ Tâches et timers
- Files et Sémaphores
- Synthèse

TACHES "freertos/task.h"

Création

```
void fun(void *) { … }
xTaskCreate(fun, 1024, NULL, 1, &task)==pdPass/pdFail
vTaskDelete();
```

Temps Ticks/ms

```
duration = pdMS_TO_TICKS(100) ;
vTaskDelay(duration) ;
TickType_t t = xTaskGetTickCount() ;
vTaskDelayUntil(&t, duration)
```

TIMERS "freertos/timers.h"

Création

```
void fun(TimerHandler_t t) { … }
TimerHandler_t t = xTimerCreate("nom", period, pdTrue/pdFalse, 0, fun) ;
xTimerDelete(t) ;
```

Démarrage/Arrêt

xTimerStart(t) ; xTimerStartFromISR(t) ;

xTimerStop(t) ; xTimerStopFromISR(t) ;

FREE RT OS

- ✓ Hello world !
- ✓ Entrées/Sorties
- ✓ Tâches et timers
- ➔ Files et Sémaphores
- Synthèse

FILES "freertos/queue.h"

Création

```
QueueHandle_t q = xQueueCreate(nItems, itemSize) ;
xQueueReset(q) ;
xQueueDelete(q) ;
```

Envoi/Réception

```
xQueueSend(q, &data, portMAX_DELAY) == pdPass/errQUEUE_FULL
xQueueReceive(q, &data, portMAX_DELAY) == pdPass/errQUEUE_EMPTY
UBaseType_t n = uxQueueMessagesWaiting(q)
```

SEMAPHORES "freertos/semphr.h"

Création

```
SemaphoreHandle_t s=xSemaphoreCreateMutex() ;
SemaphoreHandle_t s=xSemaphoreCreateCount(5, 0);
vSemaphoreDelete(s) ;
```

Prendre/Laisser

```
xSemaphoreTake(s, timeout) ; xSemaphoreTakeFromISR(s, timeout) ;
xSemaphoreGive(s) ; xSemaphoreGiveFromISR(s) ;
```

FREE RT OS

- ✓ Hello world !
- ✓ Entrées/sorties
- ✓ Tâches et times
- ✓ Files, sémaphores
- ➔ Synthèse

Atelier 4 – Decouverte FreeRTOS

Sujet

Implémenter un tri par pivot (aka quicksort) pour un tableau de 2^{20} éléments en parallélisant grâce à des tâches FreeRTOS.
Optionnel : Faire varier le nombre maximum de tâches et mesurer quel nombre apporte les meilleures performances.

Compétence

Sélectionner le système d'exploitation temps-réel adéquat
Paralléliser des traitements sur un système temps réel

Resume

Compilation et test d'un programme

Utilisation des E/S numériques et analogiques

Réaction aux évènements matériels

Lancement et synchronisation de tâches et timers

BILAN

MAINTENANT, VOUS POUVEZ…

Concevoir l'électronique d'un système embarqué simple

Interagir logiciellement avec des E/S numériques ou analogiques

Respecter des contraintes de temps par programmation

Sélectionner le système d'exploitation temps-réel adéquat

Paralléliser des traitements sur un système temps réel