Résumé Détaillé
Développement des Applications Embarquées
2025-2026

ISSIC - ISITD

Table des matières

1 Informations Générales du Cours

1.1 Modalités d'Apprentissage

- **Volume horaire :** 24h (cours magistraux + ateliers en présentiel)
- **Évaluation :**
  - Contrôle continu (Projet en binômes) : 40%
  - Examen Final : 60%
- **Méthodologie :** Ateliers pratiques en groupe, participation active

1.2 Objectifs Pédagogiques

Pour ISITD

Les systèmes embarqués sont le cœur battant de la transformation numérique. Maîtriser leur conception et programmation permet de manipuler des objets intelligents et connectés.

Pour ISSIC

La sécurité est critique dans un monde hyperconnecté. Comprendre les systèmes embarqués permet d'identifier les vulnérabilités et concevoir des systèmes résilients.

2 Fondements des Systèmes Embarqués

2.1 Définitions

Système : Ensemble organisé d'éléments interagissant entre eux et avec l'extérieur pour réaliser une fonction définie.

Système Embarqué (SE) :

— Système électronique et informatique autonome

— Ne possède pas d'entrées/sorties standards (clavier, écran d'ordinateur)

— Ordinateur intégré dans un appareil pour fonctionnement autonome

— Intègre logiciel et matériel conçus ensemble

— Ressources limitées (puissance de calcul, mémoire, stockage)

— Fonctionne en temps réel

2.2 Caractéristiques Principales

1. **Spécialisation :** Une seule mission dédiée
2. **Taille réduite :** Composants compacts et légers
3. **Fonctionnement temps réel :** Opérations en réponse à événements extérieurs
4. **Ressources contraintes :** Limitations physiques volontaires
5. **Autonomie :** Pas d'interaction humaine directe nécessaire

2.3 Exemples d'Applications

— Téléphones et agendas électroniques

— Distributeurs automatiques

— Régulateur de vitesse automobile

— Direction assistée

— Système de contrôle de trajectoire

— Pilotage automatique d'avion

— Contrôle de drones

— Dispositifs médicaux

— Équipements industriels

Exemple concret : Le Ford F-150 fonctionne avec 150 millions de lignes de code et peut comporter 100 microprocesseurs (40 pour les voitures bon marché).

3 Composants Matériels

3.1 Microcontrôleur vs Microprocesseur

Microcontrôleur :

— Mini-ordinateur sur une puce électronique

— Intègre : microprocesseur + ROM + RAM + interfaces E/S

— Tout-en-un sur une seule puce

Microprocesseur :

— Exécute des instructions stockées en mémoire

— Ne contient pas de mémoire ni de périphériques E/S intégrés

— Nécessite composants externes

3.2 Classification des Microcontrôleurs

3.2.1 Par largeur de bus

— **8 bits :** Instructions arithmétiques plus petites (Atmel 8031, 8051)

— **16 bits :** Exécution avec précision (8096 / 8xC196 family)

— **32 bits :** Contrôle automatique, robotique, haute durabilité

— **64 bits :** Applications nécessitant grande puissance de calcul

3.2.2 Par jeu d'instructions

— **CISC (Complex Instruction Set Computing) :** Une instruction remplace plusieurs

— **RISC (Reduced Instruction Set Computing) :** Réduit le temps d'exécution en diminuant le cycle d'horloge

3.2.3 Par type de mémoire

— **Mémoire intégrée :** Plus économique, empreinte plus petite

— **Mémoire externe :** Plus de flexibilité, capacités de stockage importantes

3.2.4 Par constructeur

Famille AVR (Atmel) :

— Architecture RISC

— Facilité de programmation

— Faible consommation d'énergie

— Grande polyvalence

4 Architectures et Paradigmes de Conception

4.1 IoT vs CPS

Internet des Objets (IoT) :

— Ensemble de capteurs, actionneurs et unités de calcul connectés par réseau
— Généralement des liaisons sans fil (peut inclure filaires)
— Surveille, analyse, évalue et agit
— Distribution physique (Physically distributed)
— Échelle massive (milliards d'objets)
— Formule : IoT = capteurs + réseaux sans fil + base de données

Systèmes Cyber-Physiques (CPS) :

— Combinaison de composante logicielle et entités mécaniques/électroniques
— Contrôle, monitoring, transfert en temps réel via Internet
— Couplage fort (More tightly coupled)
— Applications spécialisées
— Lié à l'Industrie 4.0 (quatrième révolution industrielle)

4.2 Edge Computing vs Cloud Computing

Edge Computing :

— Traitement local des données
— Près de la source
— Sur appareils/systèmes embarqués
— Faible latence

Cloud Computing :

— Stockage et gestion centralisée
— Centres de données distants
— Grande capacité de calcul

4.3 TinyML (Tiny Machine Learning)

Définition

TinyML relie les systèmes embarqués (hardware et software) et le machine learning pour réaliser des inférences ultra-basse consommation, bas coût, avec efficacité et confidentialité sur dispositifs à batterie.

Avantages :

— Traitement direct sur dispositifs à ressources limitées
— Ultra-basse consommation
— Faible coût
— Préservation de la confidentialité

5 Processus d'Ingénierie - Cycle en V

5.1 Principe Général

L'ingénierie d'un SE est décomposée en étapes successives représentées par un grand V.

Équation : Ingénierie = Produit + Processus

— Produit = Système embarqué

— Processus = Ingénierie des SE

5.2 Phases Descendantes (Spécification)

5.2.1 1. Cahier des Charges Fonctionnel

— Définir toutes les exigences du système

— Critères quantifiables

— Exemple : climatisation automobile (45°C → 25°C en moins de 10 min)

— Stratégies de réutilisation :

  — **Carry-Over :** Réutiliser composant d'un système précédent

  — **Carry-Across :** Réutiliser composant d'un système voisin

Exigences techniques (Normes IEC 61508) :

— **Fiabilité :** Aptitude à accomplir fonction requise

— **Disponibilité :** Aptitude à être opérationnel à instant donné

— **Maintenabilité :** Aptitude à être maintenu/rétabli

— **Sécurité :** Aptitude à éviter événements critiques

5.2.2 2. Définition Architecture Système

— Synoptique général : représentation visuelle abstraite

— Identifier composants logiciels et matériels

— Mettre en évidence interfaces et flux de données

— Clarifier structure globale

— Spécifier volumes physiques alloués

— Plans et modèles CAO

— Définir interfaces : mécaniques, électriques, communications

Outils : Diagramme états-transitions (formalisme UML) pour spécifier comportement non-ambigu.

5.2.3 3. Spécification des Composants

Capteurs : Acquisition de grandeur physique → signal électrique

— Température : DS18B20, DHT11/22, MAX31855, TMP36, MCP9808

— Lumière : Photorésistance, TSL2561, BH1750

— Distance : Capteur ultrason (HC-SR04)

— Tout ou Rien : Bouton poussoir
— Température : Thermistance

Actionneurs : Signal électrique → action physique

— LED : Signal lumineux
— Buzzer : Signal sonore
— Moteur DC : Mouvement rotatif
— Électrovanne : Contrôle de débit liquide

5.2.4 4. Conception et Réalisation

— Développement de composants spécifiques
— Hardware et software
— Respect des normes (ex : Euro pour émissions automobiles)

5.3 Phases Ascendantes (Validation)

5.3.1 5. Tests Unitaires Software

— Outils de simulation/émulation
— Automatisation des tests
— Intégration continue
— Bancs de tests matériels

5.3.2 6. Qualification Composants Hardware

— Tests de validation selon plan défini
— Vérification conformité spécifications

5.3.3 7. Tests d'Intégration

— Activités géographiquement dispersées
— Simulateurs de communication
— Vérification temps de réponse capteurs
— Transmission sans erreurs
— Respect des délais

5.3.4 8. Validation/Qualification Système Complet

— Plateforme d'intégration unique (locaux constructeur)
— Utilisation faisceau câblage réel
— Assemblage fonctions essentielles puis annexes
— Garantir robustesse et fonctionnalité globale
— Représentation fidèle conditions réelles

6 Plateformes de Développement

6.1 Arduino Uno R3

6.1.1 Microcontrôleur

ATmega328P :

— Microcontrôleur 8 bits famille AVR
— Fabriqué par Microchip Technology
— Mémoire Flash : 32 Ko
— SRAM : 2 Ko
— EEPROM : 1 Ko

6.1.2 Broches (28 au total)

Alimentation :

— VIN : 7-20V alimentation externe
— 5V : Alimentation régulée 5V
— 3.3V : Alimentation régulée 3.3V
— GND : Masse (plusieurs broches)
— IOREF : Tension de référence
— AREF : Référence analogique

Numériques (0-13) :

— INPUT : Lire signaux numériques (HIGH/LOW)
— OUTPUT : Contrôler dispositifs (LED, moteur)
— PWM (3, 5, 6, 9, 10, 11) : Contrôle luminosité/vitesse
— RX (0) / TX (1) : Communication série UART

Analogiques (A0-A5) :

— Lecture tensions analogiques uniquement (INPUT)
— Mesure signaux continus et variables
— Convertisseur ADC 10 bits (0-1023)

Autres :

— Bouton Reset : Redémarrer microcontrôleur
— Broche Reset : Contrôle programmable du reset

6.1.3 Cartes Arduino

— **Arduino Uno :** Polyvalente, idéale débutants
— **Arduino Mega :** Plus de broches et mémoire
— **Arduino Nano :** Compacte, faible encombrement
— **Arduino Micro :** Minuscule, économique

6.2 Raspberry Pi

6.2.1 Raspberry Pi 5 (2023)

— Processeur quadricœur Cortex-A76 @ 2.4 GHz
— GPU VideoCore VII
— Support PCIe
— RAM : 2, 4, 8 ou 16 Go

6.2.2 Raspberry Pi Pico / Pico W

— Microcontrôleur RP2040
— Pico W : Intègre connectivité WiFi
— Programmation MicroPython ou C
— Compact et économique

MicroPython :

— Version Python adaptée aux microcontrôleurs
— Langage de haut niveau sur microcontrôleur
— Puissance et simplicité de Python
— Optimisé pour ressources limitées

6.2.3 GPIO Raspberry Pi

40 broches avec fonctions multiples :

— Alimentation 3.3V et 5V
— Ground (GND)
— GPIO configurables (INPUT/OUTPUT)
— PWM
— I2C (GPIO 2/3 - SDA/SCL)
— SPI (MOSI, MISO, SCLK, CE)
— UART (TX/RX)

6.3 ESP32

6.3.1 Avantages par rapport à Arduino

— 15× plus rapide (240 MHz vs 16 MHz)
— 250× plus de RAM (520 Ko vs 2 Ko)
— Connectivité native (WiFi + Bluetooth)
— Microcontrôleur 32 bits dual-core
— Coût très faible

6.3.2 Spécifications Techniques

— Chip : ESP32 (Espressif Systems, Chine)
— Processeur : Xtensa 32-bit (jusqu'à 240 MHz)
— RAM : 520 Ko SRAM
— WiFi : 802.11 b/g/n
— Bluetooth : Classic & BLE
— GPIO : 34 broches programmables
— Interfaces : SPI, I^2C, UART, ADC, DAC, PWM, Touch

6.3.3 Variantes

— **ESP32-WROOM-32 :** Chip ESP32-D0WDQ6
— **ESP32-WROOM-32D :** Chip ESP32-D0WD
— **ESP32-WROVER-IB :** ESP32-D0WD + PSRAM

6.3.4 Broches Spéciales ESP32

Read Only (36, 39, 34, 35) :
— Uniquement entrées
— Ne peuvent pas être configurées en sortie
— digitalRead() et analogRead() seulement

ADC (Convertisseur Analogique-Numérique) :
— ADC1 et ADC2
— 15 canaux au total
— Résolution 12 bits (0-4095)
— Lecture capteurs analogiques

DAC (Convertisseur Numérique-Analogique) :
— DAC1 : GPIO25
— DAC2 : GPIO26
— Génération signaux analogiques
— Sortie audio

Touch :
— Broches tactiles capacitives
— Détection sans contact physique
— Projets éco-conçus

LED intégrée :
— Généralement GPIO2
— Utilisée pour tests rapides

7 Protocoles de Communication

7.1 Pourquoi des Protocoles ?

Les GPIO simples (0V ou 3.3V) ne suffisent pas pour communiquer efficacement avec composants intelligents (capteurs, écrans, GPS, etc.).

7.2 I2C (Inter-Integrated Circuit)

Caractéristiques :

— Type : Communication série synchrone
— Fils : 2 (SDA, SCL)
— SDA : Transmet données
— SCL : Signal d'horloge (synchronisation)
— Avantages : Peu de fils, facile à chaîner plusieurs périphériques
— Limites : Moins rapide que SPI

Exemples de périphériques :

— BMP280 (Température et pression)
— MPU6050 (Accéléromètre et gyroscope)
— BH1750 (Capteur de luminosité)
— PCA9685 (Contrôleur servomoteurs)

Raspberry Pi :

— GPIO 2 & 3 : Bus I2C principal
— GPIO 0 & 1 : I2C pour identification HAT

7.3 SPI (Serial Peripheral Interface)

Caractéristiques :

— Type : Communication série synchrone
— Fils : 4 (MOSI, MISO, SCLK, CS/CE)
— MOSI : Master Out Slave In (Pi → périphérique)
— MISO : Master In Slave Out (périphérique → Pi)
— SCLK : Horloge de synchronisation
— CS/CE : Chip Select (sélection périphérique)
— Avantages : Très rapide, stable
— Limites : Plus de fils, une broche CS par périphérique

Exemples de périphériques :

— MAX31855 (Thermocouple)
— NRF24L01 (Module radio)
— MCP3008 (Convertisseur ADC)
— OLED SPI (Afficheur graphique)

ESP32 :

— VSPI (par défaut)
— HSPI (secondaire)

7.4 UART (Universal Asynchronous Receiver/Transmitter)

Caractéristiques :

— Type : Communication série asynchrone
— Fils : 2 (TX, RX)
— TX : Transmit (émission)
— RX : Receive (réception)
— Pas d'horloge partagée
— Accord sur vitesse (baud rate : 9600, 115200, etc.)
— Avantages : Simple à mettre en place, très courant
— Limites : Un seul périphérique à la fois

Signaux supplémentaires (optionnels) :

— RTS (Request To Send) : "Je suis prêt à envoyer"
— CTS (Clear To Send) : "OK, tu peux envoyer"

Exemples de périphériques :

— GPS NEO-6M
— Module Bluetooth HC-05/HC-06
— Capteur CO2 MH-Z19
— Module GSM SIM800L
— Écrans série
— Lecteurs RFID

7.5 Comparaison Communication Synchrone vs Asynchrone

Synchrone (SPI, I^2C) :

— Ligne d'horloge partagée
— Émetteur et récepteur synchronisés
— Lecture au bon moment (front horloge)
— Analogie : "Claquement de mains rythmé"

Asynchrone (UART) :

— Pas d'horloge partagée
— Accord préalable sur vitesse (baud rate)
— 2 fils seulement (TX, RX)
— Analogie : "Un mot toutes les 2 secondes"

8 Outils et Environnements de Développement

8.1 Wokwi

— Plateforme de simulation en ligne
— URL : `https://wokwi.com/`
— Connexion via compte GitHub recommandée
— Conservation historique des projets
— Support Arduino, ESP32, Raspberry Pi Pico

8.2 Breadboard (Platine d'Expérimentation)

Structure :

— Rangées horizontales : 5 ports interconnectés
— Colonnes verticales : Rails d'alimentation (+/-)
— Séparation centrale : Isolation des deux moitiés
— Pas de lien électrique par la gouttière

Avantages :

— Prototypage rapide (pas de soudure)
— Flexibilité (connexions réversibles)
— Accessibilité (abordable)
— Applications variées

Utilisation :

— Rail rouge (+) : VCC (5V, 3.3V)
— Rail noir/bleu (-) : GND (masse)
— Rangées : Connexion composants
— Colonnes : Distribution alimentation

8.3 Bibliothèques Courantes

Arduino/ESP32 :

— DHT.h : Capteurs température/humidité DHT11/DHT22
— LiquidCrystal_I2C.h : Écrans LCD I2C
— Wire.h : Communication I2C
— SPI.h : Communication SPI

9 Exercices Pratiques

9.1 TP1 : Découverte Arduino Uno

— Localisation des composants sur la carte
— Configuration broches (pinMode, digitalWrite, digitalRead)
— Contrôle LED avec bouton-poussoir
— Utilisation interrupteur coulissant
— Contrôle luminosité LED avec potentiomètre (PWM)

9.2 TP2 : Système Multi-Composants

Objectif : Contrôler LED RGB selon température (DHT22)
— Température $>$ 38°C : LED rouge
— Température $<$ 0°C : LED bleue
— Température normale : LED verte
— Buzzer si température 50°C
— Déclenchement par bouton-poussoir
— Affichage LCD I2C

Composants :
— Arduino Uno
— Breadboard
— Capteur DHT22 (4 broches : VCC, SDA, NC, GND)
— LED RGB (3 résistances 200)
— Buzzer
— Bouton-poussoir
— LCD I2C 16×2
— Fils de connexion

Capteur DHT22 :
— Plage température : -40°C à +80°C
— Plage humidité : 0% à 100%
— Alimentation : 3.3V-6V (5V recommandé)
— Communication numérique (pas analogique)
— Protocol propriétaire

9.3 TP3 : Raspberry Pi Pico et MicroPython

— Découverte du microcontrôleur RP2040
— Configuration WiFi (Pico W)
— Programmation MicroPython
— Comparaison avec Arduino

9.4 TP4 : ESP32

— Configuration broches spéciales
— Utilisation ADC/DAC
— Capteurs tactiles
— Connectivité WiFi et Bluetooth
— Serveur web embarqué

10 Concepts Avancés

10.1 Temps Réel

Définition : Les opérations de calcul sont faites en réponse à un événement extérieur (interruption matérielle). La validité et pertinence d'un résultat dépendent du moment où il est délivré.

Exemple : Moniteur cardiaque surveillant en temps réel l'activité électrique du cœur d'un patient.

10.2 Contraintes de Conception

Matérielles :

— Espace limité

— Consommation électrique réduite

— Fiabilité élevée requise

Logicielles :

— Systèmes d'exploitation dédiés (RTOS)

— Programmation temps réel

— Sécurité des données

Intégration :

— Coordination capteurs/actionneurs

— Contrôle central

— Interfaces multiples

Développement :

— Conception sur-mesure

— Tests approfondis

— Maintenance long terme

— Coûts élevés

10.3 Criticité et Environnements Contraignants

Applications critiques :

— Dispositifs médicaux

— Systèmes de contrôle industriels

— Avionique

— Véhicules automobiles

Contraintes environnementales :

— Température extrême

— Vibrations

— Humidité

— Conditions difficiles

Nécessité d'approche rigoureuse :

— Minimiser les erreurs (coûteuses à corriger)

— Éviter conséquences désastreuses

— Processus de vérification et validation

11 Bonnes Pratiques

11.1 Développement

1. Suivre le cycle en V rigoureusement
2. Documenter toutes les étapes
3. Réutiliser composants validés (Carry-Over/Across)
4. Tester unitairement avant intégration
5. Automatiser les tests
6. Respecter les normes (IEC 61508, Euro, etc.)

11.2 Circuit Électronique

1. Toujours utiliser résistances avec LED
2. Respecter polarité des composants
3. Vérifier limites tension/courant
4. Documenter connexions (schémas)
5. Tester progressivement

11.3 Programmation

1. Code structuré (setup/loop)
2. Commentaires explicites
3. Noms de variables descriptifs
4. Gestion d'erreurs (isnan, vérifications)
5. Delays appropriés
6. Serial.begin() pour debugging

11.4 Sécurité

1. Valider toutes les entrées
2. Gérer cas d'erreur
3. Prévoir modes de secours
4. Tester cas limites
5. Documentation complète

12 Exemples de Code

12.1 Arduino : Clignotement LED

```
const int LEDPin = 13;

void setup() {
    pinMode(LEDPin, OUTPUT);
}

void loop() {
    digitalWrite(LEDPin, HIGH);
    delay(250);
    digitalWrite(LEDPin, LOW);
    delay(250);
}
```

12.2 Arduino : Lecture Capteur DHT22

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(9600);
    dht.begin();
}

void loop() {
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {
        Serial.println("Erreur lecture!");
        return;
    }

    Serial.print("Humidité: ");
    Serial.print(h);
    Serial.print("%\tTempérature: ");
    Serial.print(t);
    Serial.println("°C");

    delay(2000);
}
```

12.3 Arduino : Potentiomètre et PWM

```
const int LED_pin = 11;

void setup() {
    pinMode(LED_pin, OUTPUT);
}

void loop() {
    int value = analogRead(A0);
    int brightness = map(value, 0, 1023, 0, 255);
    analogWrite(LED_pin, brightness);
    delay(20);
}
```

13 Ressources et Références

13.1 Liens Utiles

— Wokwi (Simulation) : `https://wokwi.com/`

— Arduino Documentation : `https://www.arduino.cc/`

— ESP32 Documentation : `https://docs.espressif.com/`

— Raspberry Pi : `https://www.raspberrypi.org/`

— MicroPython : `https://micropython.org/`

13.2 Normes

— IEC 61508 : Sécurité fonctionnelle systèmes embarqués

— Normes Euro : Émissions véhicules automobiles

— ISO 26262 : Sécurité fonctionnelle automobile

13.3 Bibliographie

— Roberto Saracco, "Guess what requires 150 million lines of code", IEEE Future Directions, 2016

— Baudouin Dafflon et al., "The challenges of CPS for manufacturing in Industry 4.0", 2021

— H. Han & J. Siebert, "TinyML : A Systematic Review", ICAIIC 2022

14 Glossaire

ADC Analog-to-Digital Converter (Convertisseur Analogique-Numérique)

CAN Convertisseur Analogique-Numérique

CAO Conception Assistée par Ordinateur

CISC Complex Instruction Set Computing

CPS Cyber-Physical System

DAC Digital-to-Analog Converter

EEPROM Electrically Erasable Programmable Read-Only Memory

GPIO General Purpose Input/Output

I2C Inter-Integrated Circuit

IoT Internet of Things (Internet des Objets)

LCD Liquid Crystal Display

LED Light Emitting Diode

PWM Pulse Width Modulation

RAM Random Access Memory

RISC Reduced Instruction Set Computing

ROM Read-Only Memory

RTOS Real-Time Operating System

SE Système Embarqué

SPI Serial Peripheral Interface

SRAM Static Random Access Memory

TinyML Tiny Machine Learning

TR Temps Réel

UART Universal Asynchronous Receiver/Transmitter

UML Unified Modeling Language

15 Annexes

15.1 Formules Utiles

Loi d'Ohm :

$$I = \frac{V}{R} \tag{1}$$

où I = courant (A), V = tension (V), R = résistance (Ω)

Mapping (Arduino) :

$$y = \frac{(x - in_{min}) \times (out_{max} - out_{min})}{in_{max} - in_{min}} + out_{min} \tag{2}$$

Résolution ADC :

— Arduino : 10 bits $\rightarrow 2^{10} = 1024$ valeurs (0-1023)

— ESP32 : 12 bits $\rightarrow 2^{12} = 4096$ valeurs (0-4095)

Rapport cyclique PWM :

$$Rapport\ cyclique\ (\%) = \frac{Temps\ HIGH}{Priode\ totale} \times 100 \tag{3}$$

15.2 Tableau Comparatif Plateformes

Caractéristique	Arduino Uno	ESP32	Raspberry Pi 5
Processeur	8-bit 16MHz	32-bit 240MHz	64-bit 2.4GHz
RAM	2 Ko	520 Ko	2-16 Go
Flash	32 Ko	4 Mo	-
WiFi	Non (module)	Oui	Oui
Bluetooth	Non (module)	Oui	Oui
Prix	~ 25	~ 10	$\sim 60 - 120$
Consommation	Faible	Très faible	Moyenne
Usage	Débutants	IoT, embarqué	SBC, projets

15.3 Tableau Comparatif Protocoles

Caractéristique	I2C	SPI	UART
Fils	2	4	2
Type	Synchrone	Synchrone	Asynchrone
Vitesse	Moyenne	Rapide	Lente-Moyenne
Périphériques	Multiple	Multiple	Un seul
Distance	Courte	Courte	Moyenne
Complexité	Moyenne	Haute	Faible

15.4 Valeurs Standards

Résistances courantes :

— LED : 220 - 1k

— Pull-up/down : 10k

— Limitation courant : calculer selon besoin

Baud rates UART :

— 9600 (standard débutant)

— 115200 (rapide, courant)

— 57600, 38400, 19200 (intermédiaires)

Tensions courantes :

— 5V : Arduino, alimentation USB

— 3.3V : ESP32, Raspberry Pi GPIO, logique moderne

— 12V : Moteurs, LED strips

Conclusion

Ce cours couvre les fondamentaux des systèmes embarqués, de la théorie à la pratique. Les compétences acquises permettent de :

- Comprendre l'architecture des systèmes embarqués
- Suivre un processus d'ingénierie rigoureux (Cycle en V)
- Choisir et utiliser les composants appropriés (capteurs, actionneurs)
- Programmer différentes plateformes (Arduino, ESP32, Raspberry Pi)
- Implémenter des protocoles de communication (I2C, SPI, UART)
- Développer des applications IoT et temps réel
- Respecter les normes et contraintes de sécurité

Point Clé

"Les systèmes embarqués sont le cœur battant de la transformation numérique. Leur maîtrise ouvre la porte à l'innovation dans tous les domaines : santé, transport, industrie 4.0, agriculture intelligente, et bien d'autres."

Projet Final : Le projet en binômes permettra d'appliquer l'ensemble de ces connaissances sur un cas réel, en suivant toutes les étapes du cycle de développement.

Bonne continuation dans votre apprentissage des systèmes embarqués !