CERGY PARIS UNIVERSITÉ

Morse Code Communication System

Microcontroller-Based Signal Encoding and Decoding

Élèves :
Julien Raad
Sarah Ecard
Jassim Mohamed Halim
Eliarisoa Andriantsitohaina
Adam Desiles

Enseignants :
Prof. Bouallagui Hamza
Prof. Lemhaouri Zakaria
Prof. Van kets Maelle

2 avril 2025

Table des matières

1 Introduction

1.1 Contexte du projet

Ce projet s'inscrit dans le cadre du cours "Introduction aux microcontrôleurs" de la Licence 3 Informatique. L'objectif principal est de développer un système de communication sans fil entre deux microcontrôleurs STM32F4, utilisant le son comme médium de transmission.

1.2 Objectifs

L'objectif de se projet est de faire la communication entre 2 cartes stm32 à l'aide du code morse. Cela nous aidera a nous familiariser avec les principaux fondamentaux du développement embarqué. Nous avons pu également tester différentes techniques de communication comme le convertisseur analogique/ numérique, la modulation de la largeur d'impulsion, les communication asynchrones et synchrones, les interruptions et le traitement de signale.

1.3 Cahier des charges

Les spécifications du projet incluent :

- Conversion de texte en signaux Morse
- Décodage de signaux Morse en texte
- Utilisation d'un buzzer pour la génération de signaux
- Réception des signaux via un microphone (GPIO)
- Gestion précise des durées des signaux (points, tirets, espacements)

2 Conception du Système

2.1 Architecture matérielle

- Microcontrôleur : STM32 (basé sur l'architecture ARM Cortex-M)
- Composants principaux :
  - Buzzer connecté à GPIO_PIN_0 de GPIOA
  - Entrée GPIO pour la réception de signaux
  - Horloge système configurée avec PLL
- Paramètres système :
  - Fréquence système : 84 MHz
  - Tension de régulation : Échelle 3

2.2 Conception logicielle

- Langage de programmation : C
- Structure du code :
  - Modules de conversion Morse
  - Gestion des durées de signaux
  - Décodage et encodage
- Algorithmes principaux :
  - Conversion de caractères en signaux Morse
  - Détection et décodage des signaux
  - Gestion des temporisations

3 Mise en Œuvre

3.1 Développement du Matériel

Configuration du GPIO et du système :

- Initialisation du buzzer sur GPIOA
- Configuration des broches GPIO
- Mise en place de l'horloge système

3.2 Branchement du matérielle

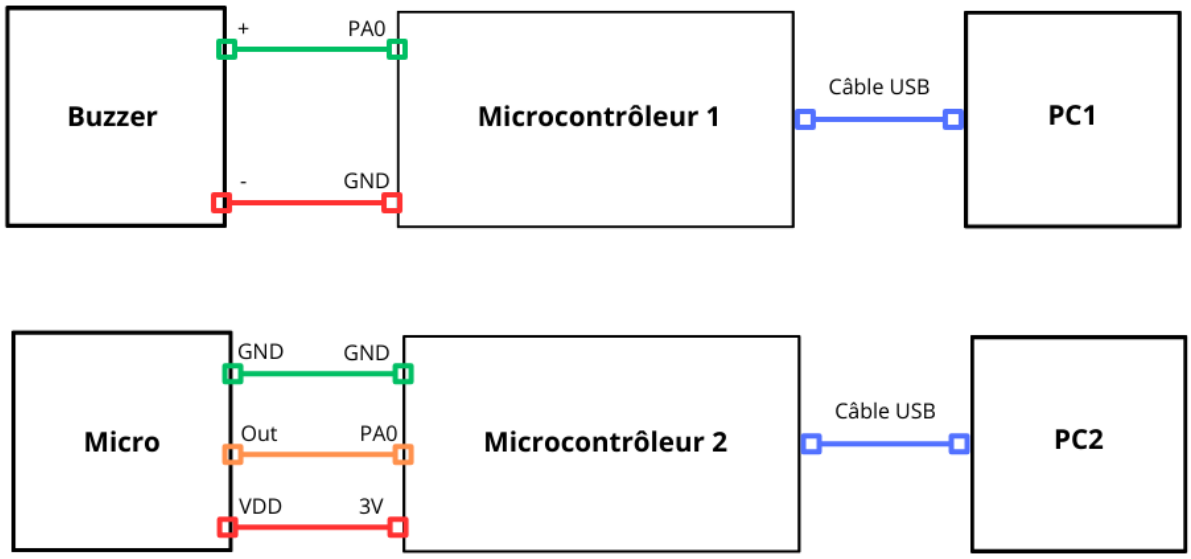

FIGURE 1 – Architecture du système de communication Morse

3.2.1 Configuration des broches

Le Buzzer est branché sur le port "+" relié à une broche (PA0) du microcontrôleurs 1. Ensuite le port "-" est branché au "GND" du microcontrôleur 1. Le microphone est branché sur le port "GND" qui est connecté au "GND" du microcontôleur 2, la borne "VDD" est branché à l'alimentation de 3V sur le microcontôleur 2. Et enfin, la broche "PA0" est relié à la sortie "Out". Les microcontrôleurs connecté à 2 PC différents.

Émetteur (PB3 - Buzzer) : La broche PB3 est configurée en mode sortie (GPIO Output) afin de piloter le buzzer. Ce dernier est activé ou désactivé selon les signaux à transmettre.

Récepteur sonore (PA0 - ADC1) : La broche PA0 est configurée en mode analogique (GPIO Analog) pour permettre la lecture du signal provenant du microphone via l'ADC1.

Communication UART (PA2/PA3 - USART2) : Les broches PA2 (TX) et PA3 (RX) sont configurées pour assurer la communication série via USART2, notamment pour envoyer ou recevoir des messages depuis un terminal comme PuTTY.

3.2.2 Configuration du Timer et des Interruptions

Timer TIM2 – Mesure des signaux Le timer TIM2 est utilisé pour mesurer la durée des signaux sonores (émis par le buzzer) et des silences. Il démarre dès qu'un son est détecté et s'arrête à la fin du signal. La durée mesurée permet de distinguer un point (signal court) d'un tiret (signal long). Interruption liée au Timer Des interruptions sont générées par le TIM2 à chaque changement d'état du signal sonore (début ou fin). Ces interruptions déclenchent :

- L'enregistrement de la durée du son ou du silence
- La mise à jour des variables internes
- Le déclenchement du décodage du signal Morse

3.3 Émetteur

Implémentation des fonctionnalités de l'émetteur :

- Conversion d'une phrase en morse

Code 1 : Fonction de conversion Morse C

```c
void convertirEtJouerMorse(const char *phrase) {
    int i = 0;
    while (phrase[i] != '\0') {
        playMorseChar(phrase[i]);
        i++;
```

```
    }
}
```

Dans cette implémentation, le message à transmettre est déjà défini statiquement dans le code via la variable test, qui contient la chaîne "sos". Il ne s'agit donc pas d'une réception dynamique via UART, mais plutôt d'un message prédéfini qui sera ensuite encodé en Morse.

- Conversion d'un caractère et jouer le son

Code 2 : Fonction pour jouer un caractère en Morse C

```c
void playMorseChar(char c) {
    c = toupper(c);

    // Trouver le code Morse correspondant
    if (c >= 'A' && c <= 'Z') {
        const char *code = morse[c - 'A'];

        // Jouer le code Morse du caractère
        for (int i = 0; code[i] != '\0'; i++) {
            if (code[i] == '.') {
                playDot();
            } else if (code[i] == '-') {
                playDash();
            }
             // Espace entre les lettres (moins l'espace entre
             ↪  symboles déjà ajouté)
             Timer_Wait(LETTER_SPACE - SYMBOL_SPACE);
        }
    else if (c >= '0' && c <= '9') {
        const char *code = morse[c - '0' + 26];

        // Jouer le code Morse du chiffre
        for (int i = 0; code[i] != '\0'; i++) {
            if (code[i] == '.') {
                playDot();
            } else if (code[i] == '-') {
                playDash();
            }
        // Espace entre les lettres (moins l'espace entre symboles
        ↪  déjà ajouté)
        Timer_Wait(LETTER_SPACE - SYMBOL_SPACE);
    }
```

```
    else if (c == ' ') {
        // Espace entre les mots (moins l'espace entre lettres déjà
        ↪  ajouté)
        Timer_Wait(WORD_SPACE - LETTER_SPACE);
    }
}
```

Dans notre implémentation, nous utilisons un buzzer actif, ce qui simplifie considérablement la génération des signaux sonores nécessaires pour transmettre le code Morse. Contrairement à un buzzer passif, qui nécessite un signal PWM (Pulse Width Modulation) à une fréquence donnée pour produire un son, le buzzer actif intègre déjà un circuit d'oscillation. Ainsi, il suffit d'alimenter la broche de contrôle en tension haute (niveau logique 1) pour qu'il émette automatiquement un bip sonore.

- Fonction principale

Code 3 : Fonction principale C

```c
// Activation/désactivation du buzzer
void buzzerOn();
void buzzerOff();

// Génération des composants du code Morse
void playDot();
void playDash();
void playMorseChar(char c);

// Conversion et émission du message en Morse
void convertirEtJouerMorse(const char *phrase);
```

Dans le code, l'allumage du buzzer est réalisé par la fonction buzzerOn(), qui utilise l'instruction HAL GPIO WritePin() pour mettre la broche à l'état haut. L'extinction se fait via buzzerOff(), en mettant la broche à l'état bas. Pour jouer les différents symboles du code Morse, la fonction playDot() génère un signal sonore de 300 ms pour représenter un point, tandis que playDash() produit un son de 900 ms pour un tiret. Après chaque signal, un délai est appliqué pour marquer une pause entre les symboles.

- Dictionnaire

Code 4 : Dictionnaire Morse - Caractères C

```
Lettres (A-Z):
A: .-    B: -...  C: -.-.  D: -..   E: .
F: ..-.  G: --.   H: ....  I: ..    J: .---
K: -.-   L: .-..  M: --    N: -.    O: ---
P: .--.  Q: --.-  R: .-.   S: ...   T: -
U: ..-   V: ...-  W: .--   X: -..-  Y: -.--
Z: --..

Chiffres (0-9):
0: ----- 1: .---- 2: ..--- 3: ...-- 4: ....-
5: ..... 6: -.... 7: --... 8: ---.. 9: ----.
```

Ce choix d'un buzzer actif permet une implémentation simple, efficace et stable, sans nécessiter de configuration de timers en mode PWM. Il est donc parfaitement adapté à un système de transmission Morse basé sur la gestion temporelle précise via des timers standards.

3.4 Récepteur

Implémentation des fonctionnalités du récepteur :

- Réception du son
- Conversion d'un son en caractère
- Fonction principale

Code 5 : Fonction principale C

```c
// Lecture des données du capteur sonore
uint16_t readSoundSensor();

// Détection des signaux Morse
void detectMorseSignal();

// Décodage du Morse en caractères
char decodeMorse(const char* morseCode);

// Traitement des signaux et décodage du message
void processSignal(uint16_t signalValue);
```

- Dictionnaire

1. Détection des signaux sonores et acquisition avec ADC ou Timer. La détection des signaux sonores est effectuée à l'aide de l'ADC (convertisseur analogique-numérique), qui

échantillonne la valeur du signal analogique provenant du capteur sonore. L'ADC est configuré pour fonctionner en mode de conversion continue, et les valeurs sont lues à intervalle régulier (toutes les millisecondes) par l'intermédiaire de la fonction HAL ADC PollForConversion(). Si une valeur d'ADC dépasse un seuil défini (THRESHOLD), cela indique qu'un son a été détecté. Pour mesurer la durée du son, un timer (TIM2) est utilisé pour obtenir des informations sur le temps. Lorsqu'un son est détecté, le moment du début du son est marqué grâce à un compteur TIM, et un état is sound detected est mis à 1. Ensuite, dès que le son s'arrête (c'est-à-dire lorsque la valeur ADC redescend en dessous du seuil), la durée du son est calculée en fonction de l'écart entre les valeurs du compteur du timer au début et à la fin du son. Cela permet de déterminer si le son est un "point" (durée courte) ou un "tiret" (durée plus longue), conformément à la logique du code Morse.

2. Décodage en morse et conversion en texte Le code Morse est une représentation binaire des lettres et chiffres, où un "point" est un "0" et un "tiret" est un "1". Lorsque la durée du son détecté est inférieure à un seuil (DOT DURATION), cela est interprété comme un "point", et sinon, c'est un "tiret". Ces valeurs sont stockées dans un tableau morse code pour former une séquence. Une fois que les silences entre les signaux sont suffisamment longs, la séquence accumulée dans morse code est comparée aux séquences Morse connues pour chaque caractère de l'alphabet et des chiffres (contenus dans le tableau codeMorse). Cette comparaison est effectuée par la fonction recherchemorse(). Si une correspondance est trouvée, la lettre ou le chiffre correspondant est ajouté à un tableau texte. En cas d'absence de correspondance, un caractère de remplacement (comme " ?" ou "") est utilisé. De plus, un espace est ajouté à chaque fois qu'une séquence complète de caractères est décodée, ou lorsqu'un long silence est détecté.

3. Envoi du message décodé vers le PC via UART. Le message décodé est envoyé en temps réel au PC via une interface série UART (USART2). Cela permet à l'utilisateur de suivre la progression du décodage en temps réel. Après chaque ajout d'un nouveau caractère décodé, le tableau texte est mis à jour et envoyé via la fonction UART SendString(). Cette fonction transmet le contenu de la chaîne de caractères au PC, qui peut l'afficher ou l'utiliser pour d'autres traitements.

3.5 Gestion de la communication UART

La communication entre les microcontrôleurs et les PCs est gérée par l'UART2, configuré en mode asynchrone avec les paramètres suivants :

- Vitesse : 115200 bauds
- Fonctions de génération de signaux Morse
- Format : 8 bits de données, pas de parité, 1 bit de stop
- Contrôle de flux : aucun

Code 6 : Fonction principale C

```c
// Réception de données
HAL_UART_Receive(&huart2, buffer, size, timeout);

// Transmission de données
HAL_UART_Transmit(&huart2, buffer, size, timeout);
```

4 Tests et Validation

Cette section présente les différents tests réalisés afin d'évaluer la fiabilité de la communication sonore entre l'émetteur (buzzer actif) et le récepteur (capteur sonore via ADC). L'objectif est d'analyser la portée effective, les conditions optimales de fonctionnement et les sources d'erreurs potentielles pouvant altérer la transmission et le décodage du message.

4.1 Protocole de test

Méthodes de validation :

- Tests de conversion texte-Morse
- Tests de décodage de signaux Morse
- Vérification de la précision temporelle
- Tests de réception et transmission

4.1.1 Portée de transmission

La portée de transmission dépend principalement de la sensibilité du capteur à la fréquence sonore émise par le buzzer. En pratique, la détection du signal sonore devient imprécise au-delà d'une certaine distance. Pour identifier cette limite de fonctionnement, nous avons mené une série de tests en éloignant progressivement le buzzer du capteur. Les résultats montrent qu'au-delà d'une certaine distance (environ [à préciser : ex. 60 cm]), la qualité du signal diminue fortement, ce qui compromet la détection correcte des points et des tirets. Une distance raisonnable et constante entre les deux dispositifs est donc recommandée.

4.1.2 Précision de transmission

La précision de la transmission sonore est fortement influencée par l'environnement sonore. En présence de bruits parasites (conversation, bruit ambiant, objets en mouvement), le capteur peut enregistrer ces interférences comme des signaux valides, provoquant ainsi une interprétation erronée du message Morse.

Les tests montrent que pour assurer une communication fiable :

- L'expérience doit être menée dans un environnement calme.

- Le buzzer doit être orienté directement vers le capteur, à une distance stable.
- Les délais de transmission restent acceptables, avec une latence très faible, la détection et l'interprétation s'effectuant quasiment en temps réel.

L'interface UART permet de suivre en temps réel la réception et l'analyse du signal sonore. Les données sont affichées via un terminal série (comme PuTTY), selon deux phases :

1.Phase de décodage brut : les signaux détectés sont affichés sous forme de séquences de 0 et 1, correspondant aux points et tirets du code Morse.

2.Phase de traduction : à la fin de chaque caractère, la séquence complète est convertie en lettre ou chiffre et affichée sous forme de texte lisible. Cette double étape d'affichage permet de vérifier la cohérence du décodage, d'identifier les éventuelles erreurs, et de faciliter le débogage en comparant la séquence Morse détectée au caractère interprété.

4.2 Résultats des Tests

Critères de performance :

- Plage de détection des signaux :
  - Point : ≤ 110 ms
  - Tiret : ≥ 290 ms
- Capacité de décodage : Lettres A-Z, Chiffres 0-9

5 Analyse et Discussion

5.1 Interprétation des Résultats

Points forts :

- Système compact et efficace
- Algorithmes de décodage robustes
- Flexibilité de configuration

Limitations :

- Sensibilité aux variations de timing
- Limitation du jeu de caractères
- Dépendance à la précision matérielle

5.2 Perspectives d'Amélioration

- Ajout de la prise en charge des caractères spéciaux
- Amélioration de l'algorithme de décodage
- Interface utilisateur plus avancée
- Support de la transmission sans fil

Bien que le système de communication basé sur des signaux sonores ait démontré sa fonctionnalité et sa pertinence, plusieurs axes d'amélioration peuvent être envisagés afin de le rendre plus fiable, performant et adaptable à des environnements variés.

5.2.1 Améliorations techniques envisagées

Optimisation du filtrage du signal : L'intégration d'un filtre numérique permettrait de réduire les bruits parasites captés par le capteur sonore, améliorant ainsi la qualité de la détection et la robustesse du décodage.

Meilleure gestion des interruptions : En optimisant la gestion des interruptions liées aux changements de signal, il serait possible de réduire les erreurs de détection, notamment dans des environnements soumis à des variations rapides de bruit.

Extension de la portée de communication : Une amélioration de l'intensité du signal émis et un calibrage précis des seuils du capteur permettraient d'augmenter significativement la distance de fonctionnement fiable entre émetteur et récepteur.

5.2.2 Extensions fonctionnelles et évolutions possibles

Renforcement du décodage automatique : Une amélioration des algorithmes de reconnaissance des séquences Morse pourrait permettre un décodage plus rapide et plus précis, même en conditions perturbées.

Interface utilisateur interactive : La conception d'une interface graphique permettrait de visualiser en temps réel les signaux détectés, le message Morse et le texte décodé, rendant le système plus accessible pour un usage pédagogique ou grand public.

Ajout d'un canal de communication infrarouge : En complément du canal sonore, l'intégration d'un module infrarouge offrirait une alternative plus stable dans les environnements bruyants, améliorant la polyvalence et l'adaptabilité du système à différents contextes d'utilisation.

6 Conclusion

En résumé, les apports les plus significatifs de ce projet résident dans la mise en œuvre réussie de la transmission sonore et infrarouge, ainsi que dans le décodage automatique des messages, démontrant à la fois la faisabilité technique et la pertinence du concept. Les résultats obtenus aboutissent à un prototype fonctionnel, capable de transmettre des messages avec une précision globalement satisfaisante.

Cependant, certaines améliorations restent envisageables, notamment en ce qui concerne la robustesse du décodage face aux interférences, ainsi que l'ajout d'une interface utilisateur graphique ou physique pour faciliter l'interaction avec le système.

Ce projet nous a permis d'acquérir de solides compétences pratiques en programmation embarquée, en communication série, ainsi qu'en traitement de signaux analogiques. Il a également ouvert des perspectives intéressantes quant à l'utilisation du code Morse dans des contextes contraints, comme les environnements à faible connectivité, illustrant le potentiel de systèmes de communication simples mais efficaces.

7 Annexes

7.1 Guide d'utilisation

1. Configurer le matériel selon le schéma de connexion (Figure 1)
2. Programmer le microcontrôleur avec le code fourni
3. Connecter l'alimentation
4. Utiliser le terminal série (115200 bauds) pour saisir les messages à transmettre
5. Observer la réception des messages sur le terminal série du récepteur

7.2 Tableau de correspondance Morse complet

Char	Code	Char	Code	Char	Code
A	.-	N	-.	0	——
B	-...	O	—	1	.—-
C	-.-.	P	.–.	2	..—
D	-..	Q	–.-	3	...–
E	.	R	.-.	4	....-
F	..-.	S	...	5	.....
G	–.	T	-	6	-....
H	....	U	..-	7	–...
I	..	V	...-	8	—..
J	.—	W	.–	9	—-.
K	-.-	X	-..-		
L	.-..	Y	-.–		
M	–	Z	–..		

Table 1 – Table de correspondance complète des caractères en code Morse

7.3 Affichage du résultat

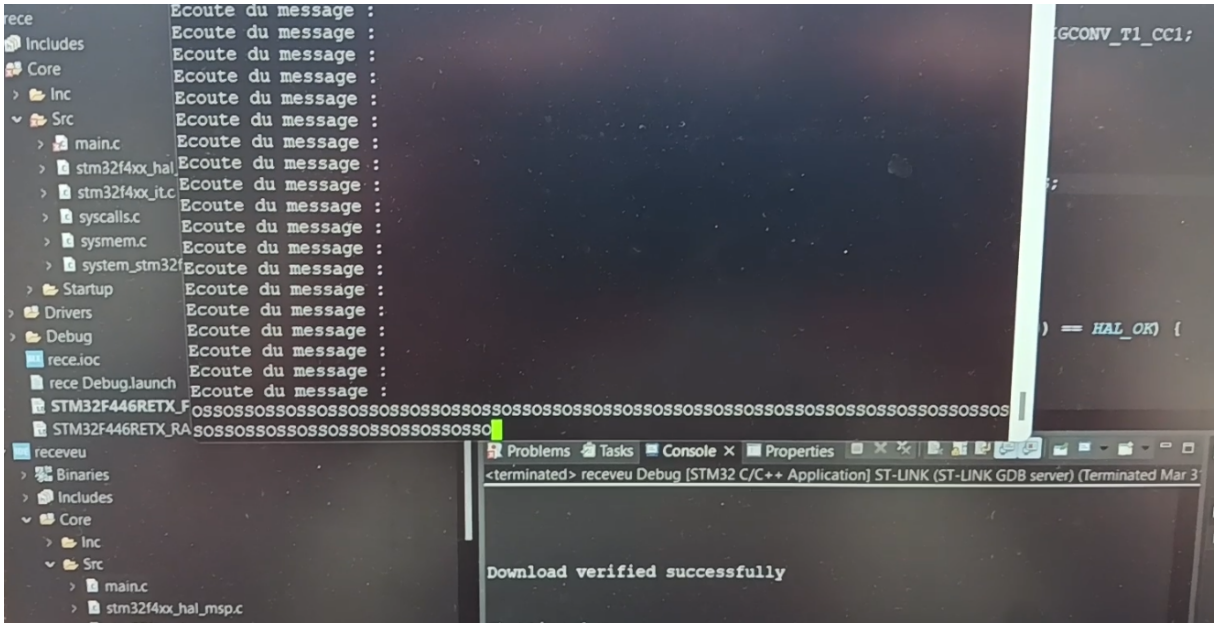

FIGURE 2 – Résultat sur la console externe PUTTY