1,2,3...SOLEIL

ABDELLAH Imene - SAUVAGE Maïlys - VIXAMAR Clarah

SOMMAIRE

I - MOTIVATIONS DU PROJET

- Conception d'un projet en lien avec notre futur métier

↓

Création d'un jeu interactif pour les enfants

↓

Modernisation d'un jeu traditionnel avec une touche technologique

Réflexion autour d'outils innovants, inclusifs et ludiques

- Stimulation de l'autonomie des enfants ⟶ jeu accessible même pour un enfant seul
- Conception d'un dispositif mobile, utilisable dans divers contextes

Combinaison de plusieurs éléments dans la création technologique

II - PROBLÉMATIQUE ET OBJECTIFS

Problématique : Comment permettre à un enfant de jouer à "1,2,3...Soleil" de manière autonome grâce à un dispositif électronique ?

🎯 OBJECTIFS 🎯

1. Un dispositif **simple d'utilisation** pour les enfants

   Interface intuitive : leds, son facile à comprendre, bouton simple

2. Un système **évolutif** et facilement **modifiable**

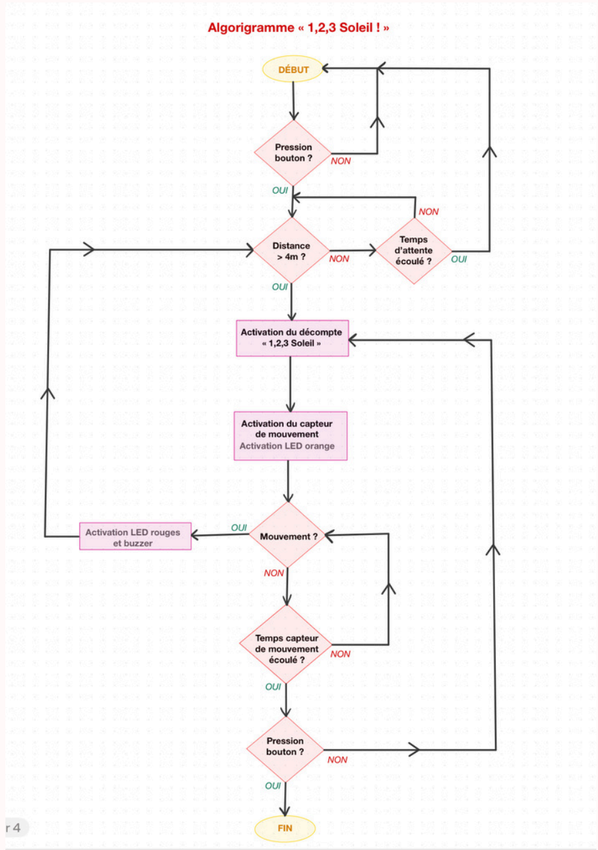
   Changer la distance de détection, ajouter des étapes, modifier les sons, les LED sans tout refaire

3. Garantir la **fiabilité du fonctionnement** même en conditions d'usage répétées

   Eviter les bugs, les déclenchements intempestifs

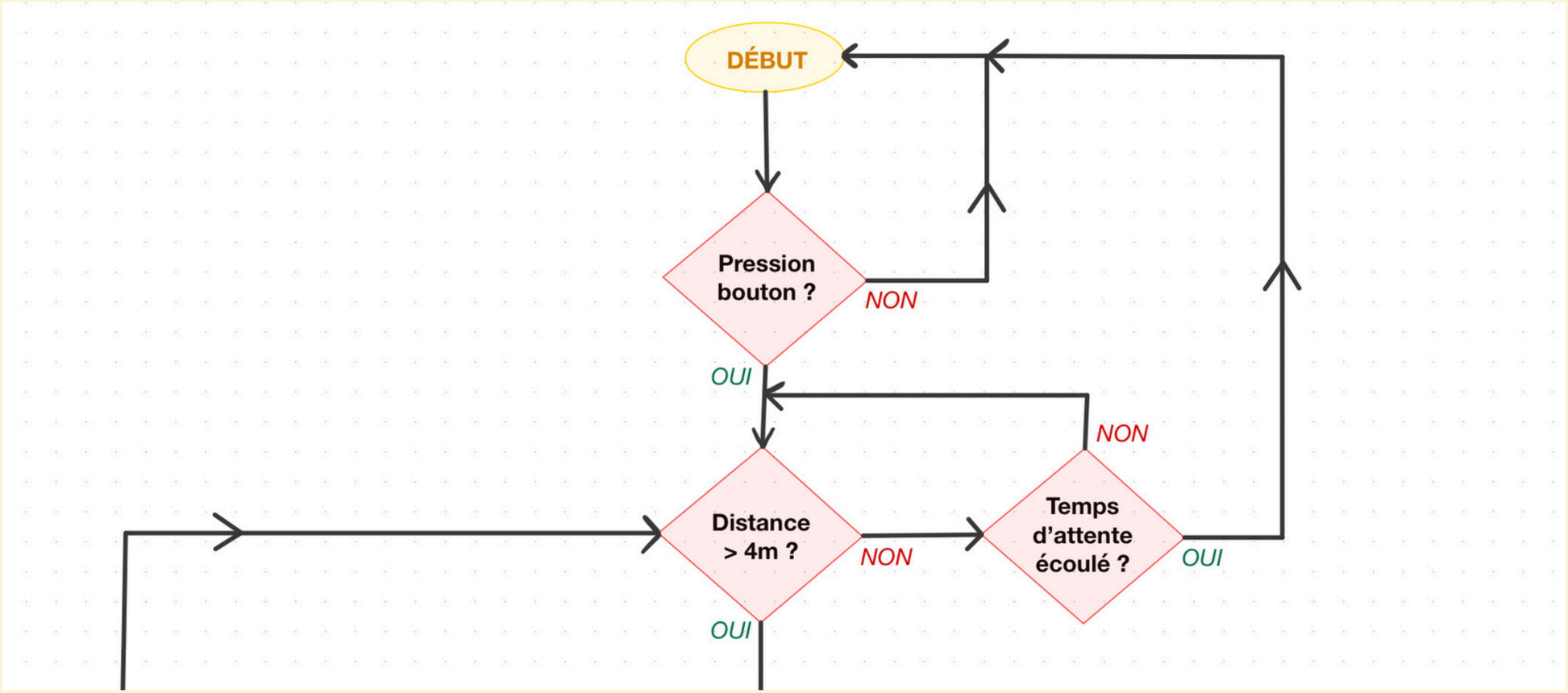
III- FONCTIONNEMENT DU DISPOSITIF

SCHÉMA FONCTIONNEL



1 Lancement du jeu

1



1 Lancement du jeu

FONCTIONNEMENT

- L’enfant appuie sur le bouton
  ↪ Mise en route

- Position de départ requise pour démarrer le jeu
  Env. 2m

- Temps d’attente paramétré 5 min

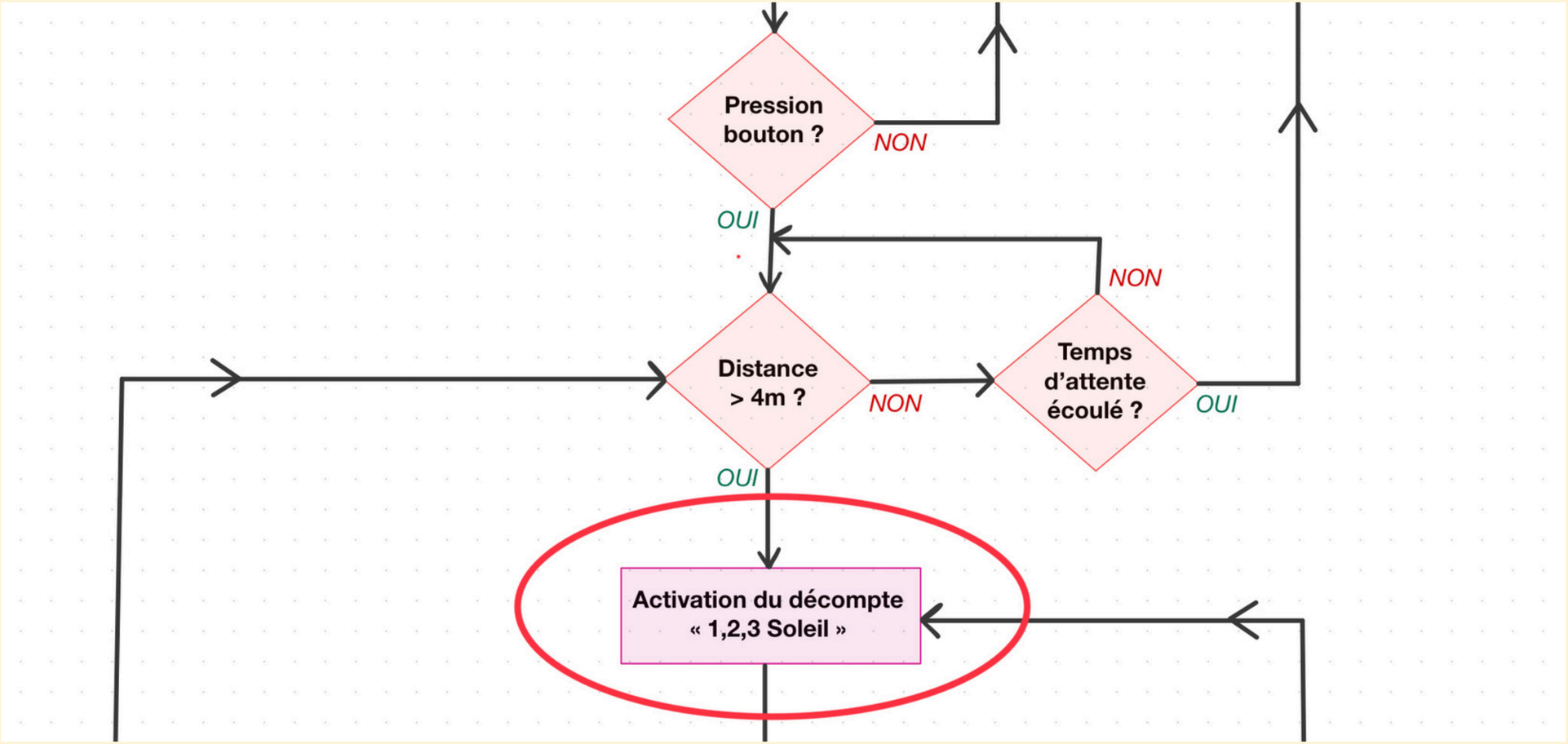
Aucun joueur détecté ? Le dispositif s’éteint
↪ Préservation du matériel électronique

MATÉRIEL

- Bouton
- Capteur ultrason

2 Décompte du “1,2,3 Soleil !”

2



2 Décompte du “1,2,3 Soleil !”

FONCTIONNEMENT

Distance requise OK

- Décompte du "1,2,3... SOLEIL"

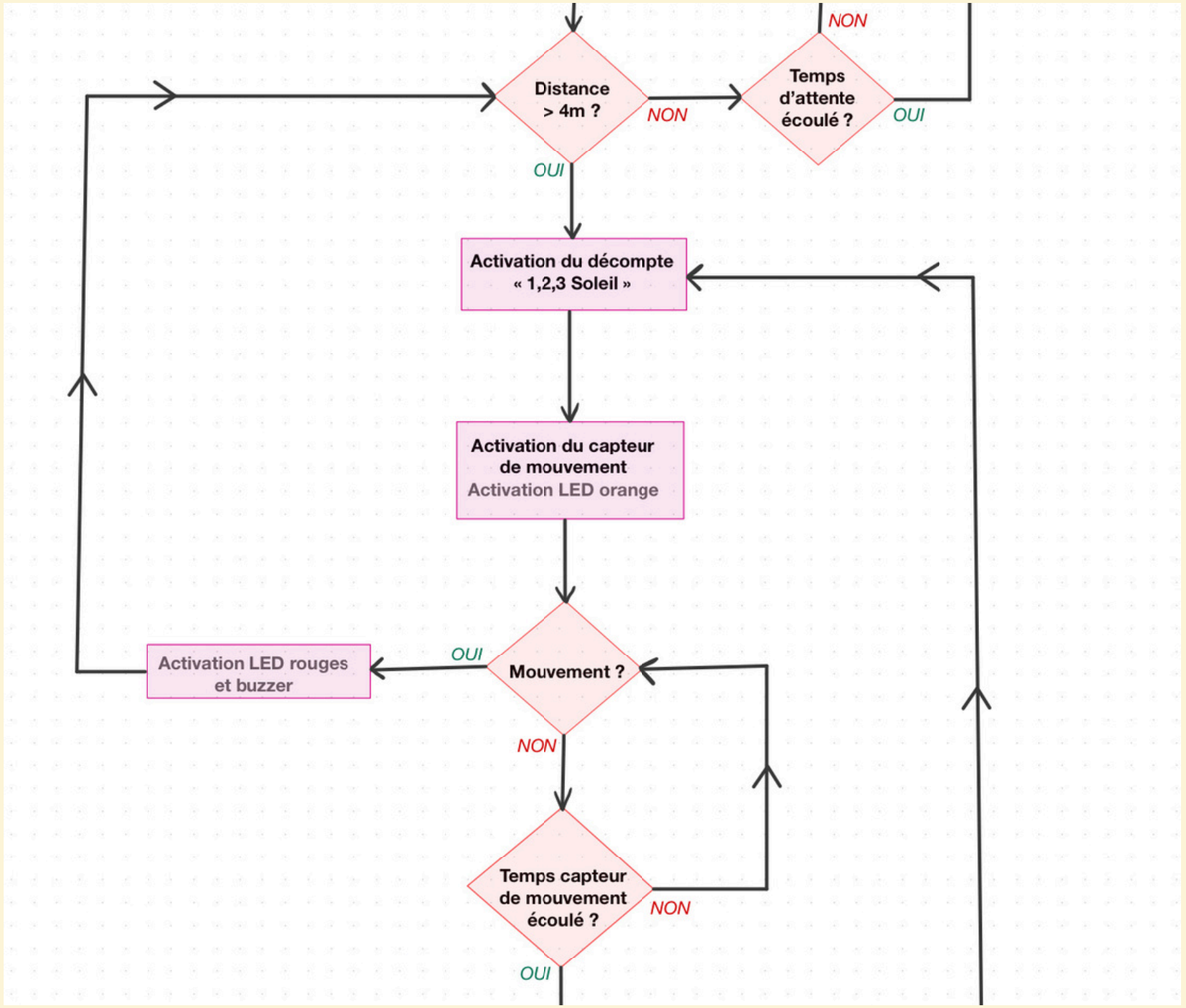
4 bips sonores successifs

Allumage progressif de 4 LEDs

MATÉRIEL

- Buzzer (sonore)
- Leds

3 Détection de mouvement



3 Détection de mouvement

FONCTIONNEMENT

Décompte terminé

↓

Activation capteur de mouvement

Temps paramétré

Pas de mouvement

- Lancement d'un nouveau décompte

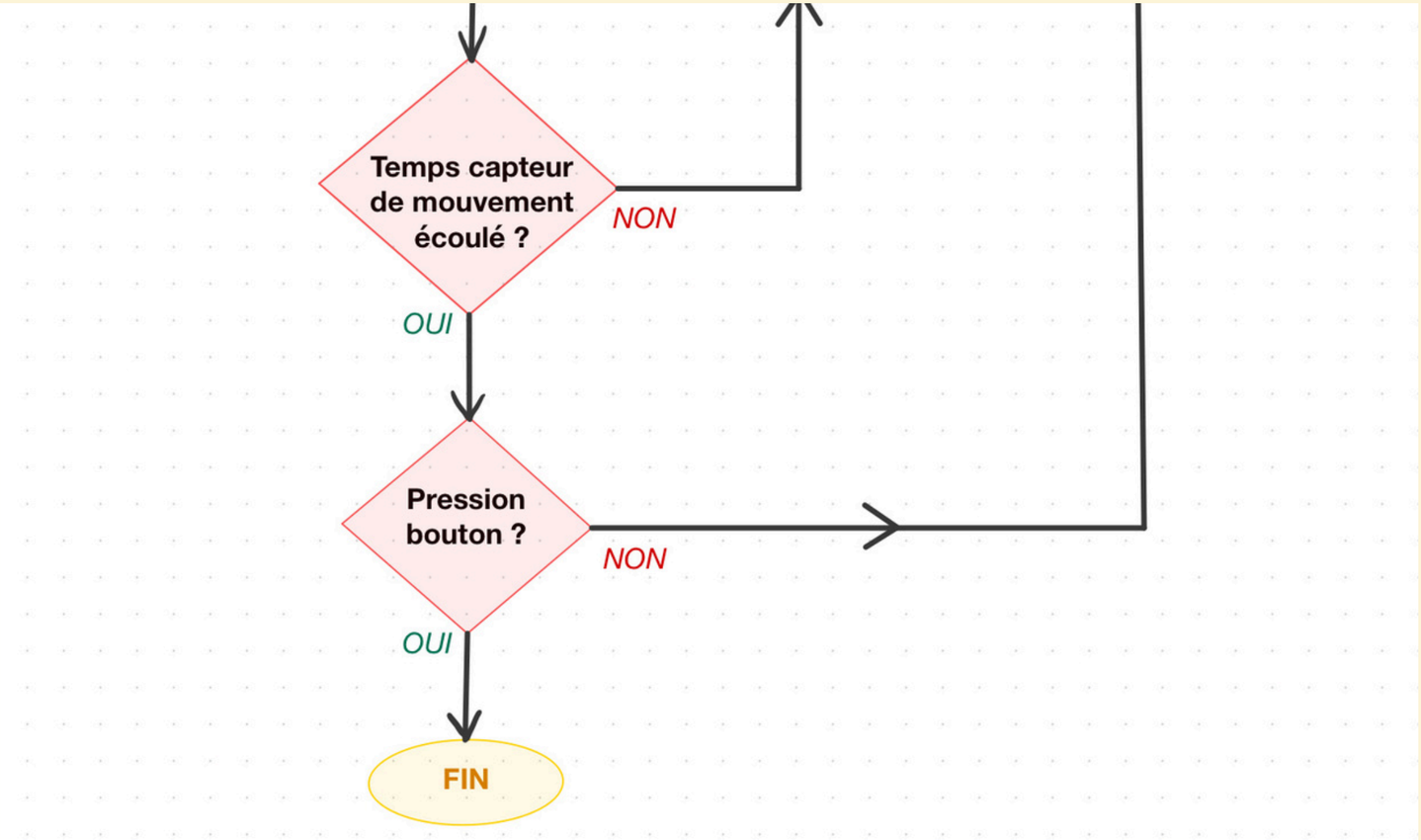
Mouvement

- Allumage des Leds en rouge
- Bip d'alerte
- Retour à la position de départ requise pour relancer un tour

MATÉRIEL

- Détecteur mouvement
- Buzzer
- Leds
- Capteur Ultrason

4 Fin de la partie



4 Fin de la partie

FONCTIONNEMENT

La partie est remportée lorsque l'enfant appuie sur le bouton

⬇ Allumage des Leds en mutlicolores

⬇ Bip de victoire

↪ **Fin du jeu**

MATÉRIEL

- Bouton
- Buzzer
- Leds

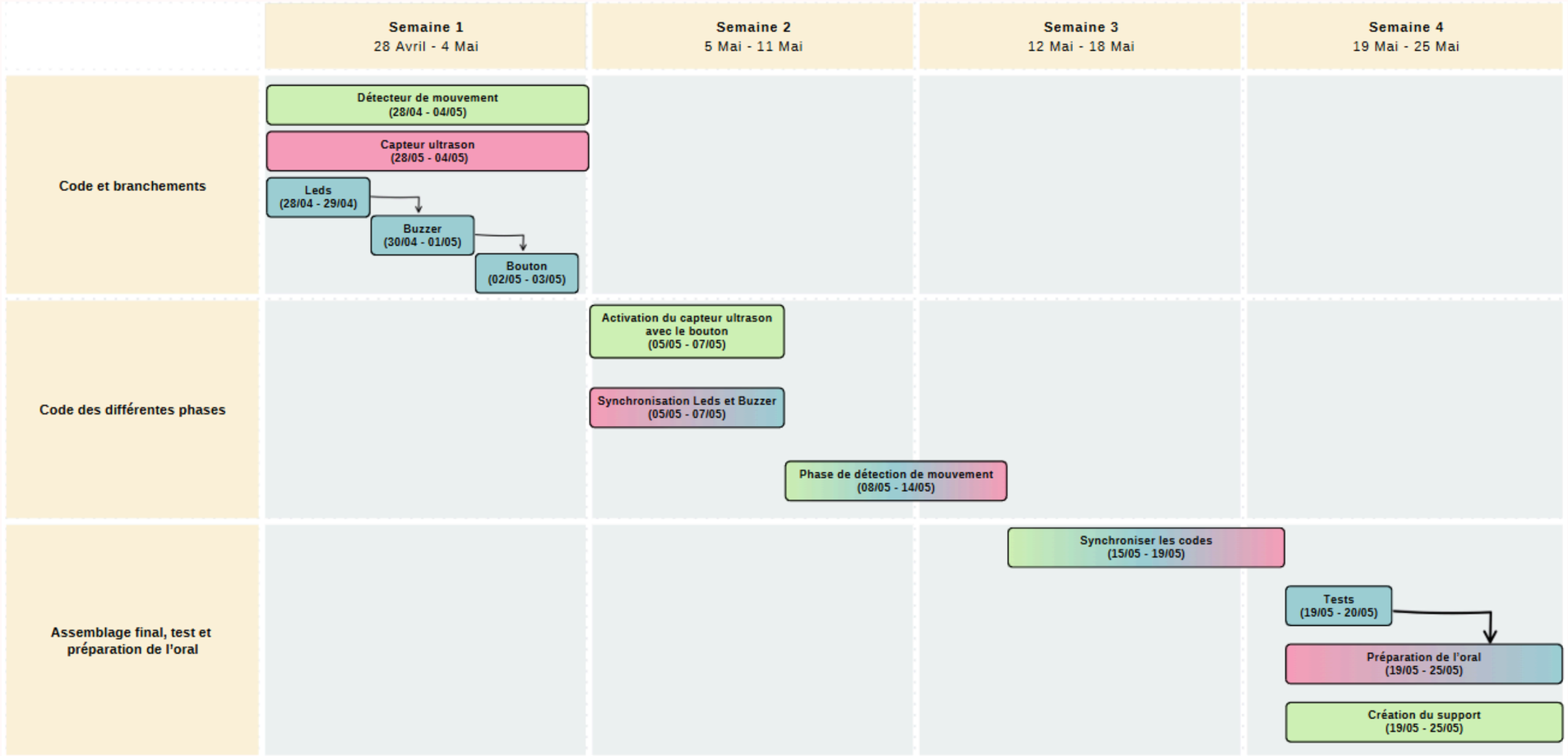
IV - PLANNING DE RÉALISATION

DIAGRAMME DE GANTT

Maïlys

Clarah

Imene

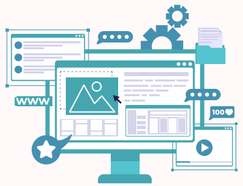


IV - PLANNING DE RÉALISATION

Obstacles rencontrés

Programmation : combien de temps faut-il pour coder telle ou telle chose

Organisation du codage : par où commencer ?



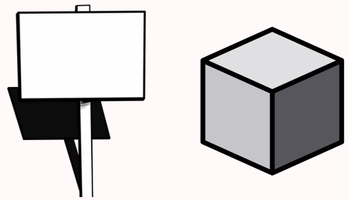
Contraintes techniques : quel matériel choisir ?

V - CONCLUSION ET PERSPECTIVES

Ce qu’il nous reste à faire

- Finaliser la programmation complète du jeu (code)
- Tester le fonctionnement en conditions réelles
- Trouver un support adapté pour installer notre dispositif

boîte, panneau, socle ?



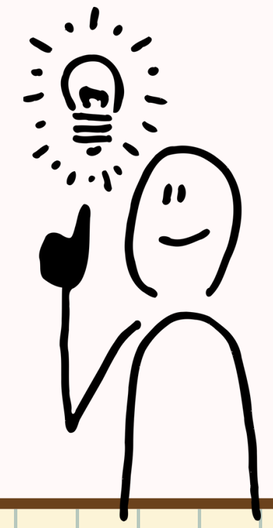
V - CONCLUSION ET PERSPECTIVES

Ce qu’on a appris

Nécessité de **structurer nos idées**, s’organiser avec un schéma fonctionnel

Adapter nos ambitions à la réalité technique

Apprendre à organiser un projet et **résoudre les problèmes** ensemble



Commencer à concevoir un premier projet électronique

- Travail en équipe et répartition des tâches
- Utilisation du diagramme de Gantt

Merci pour votre écoute

PRÉSENTATION FINALE ET DÉMONSTRATION

1,2,3...SOLEIL

ABDELLAH Imene - SAUVAGE Maïlys - VIXAMAR Clarah

SOMMAIRE

CAPTEUR ULTRASONS

Fonction lireDistanceUltrason : mesure la distance entre le capteur et l'enfant placé devant (envoie d'une impulsion sonore puis mesure du temps de retour)

CODE

- S'assure que la broche TRIG (émission) et à l'état low pendant 2 micros. pour "réinitialiser" le capteur.

- Envoi d'une brève impulsion à l'état high sur TRIG pour déclencher l'émission de l'ultrason.
  ↳ Emission d'une onde sonore (haute fréquence)

- Mesure du temps de retour de l'onde jusqu'au capteur (réception sur la broche ECHO)

- pulseIn : quand la broche passe à HIGH → mesure la durée jusqu'à ce qu'elle redevienne LOW, et retourne cette durée.

```
1  float lireDistanceUltrason() {
2    digitalWrite(TRIG_PIN, LOW);
3    delayMicroseconds(2);
4    digitalWrite(TRIG_PIN, HIGH);
5    delayMicroseconds(10);
6    digitalWrite(TRIG_PIN, LOW);
7
8    long duration = pulseIn(ECHO_PIN, HIGH);
9    float distance = duration * 0.034 / 2;
10   return distance;
11 }
```

Distance calculée à partir du temps mesuré :
- Le son se déplace à environ 0,034 cm/µs.
- On divise par 2 car le signal a fait un aller-retour.

DÉTECTION MOUVEMENTS

```
1  bool detecterMouvement() {
2    int etat = digitalRead(PIR_PIN);
3    if (etat == HIGH) {
4      return true;
5    } else {
6      return false;
7    }
8  }
```

- Lecture de l'état du capteur PIR connecté à la broche PIR_PIN

- Si le capteur détecte un mouvement (= changement de rayonnement IR dû à un déplacement humain)

→ il renvoie HIGH (1), retourne *true*

- S'il ne détecte rien

→ il renvoie LOW (0), retourne *false*

BUZZER

- Fonction sonAllumageLED: joue un court son aigu (80 ms), synchrone à l'allumage des LEDs lors du décompte
- Fonction sonPerdu : joue un son grave (500 ms), signale la défaite (le mouvement)
- Fonction sonVictoire: joue une petite mélodie de victoire de 4 notes de 150 ms espacées de 50 ms
- noTone : coupe le son à la fin si elle reçoit true :

```
1   void sonAllumageLED() {
2     tone(A4, 1000, 80);
3   }
4
5   void sonPerdu() {
6     tone(A4, 200, 500);
7   }
8
9   void sonVictoire() {
10    int notes[] = { 523, 659, 784, 988 };  // Do - Mi - Sol - Si aigu
11    int duration = 150;
12
13    for (int i = 0; i < 4; i++) {
14      tone(BUZZER_PIN, notes[i], duration);
15      delay(duration + 50);  // petit espace entre les notes
16    }
17
18    noTone(BUZZER_PIN);
19  }
```

BOUTON

```
1       extern bool jeuEnCours; // variable de loop.ino
2
3       bool gererBouton() {
4         int etat = digitalRead(BUTTON_PIN);
5
6         if (etat == LOW) {
7           return true;
8         } else {
9           return false;
10        }
11      }
12
13      // Vérifie bouton, interrompt le jeu si appuyé
14      bool verifierFin() {
15        if (gererBouton()){
16            sonVictoire();
17            Serial.println("VICTOIRE !!!");
18            allumerTout(CRGB(255, 215, 0));
19            jeuEnCours = false;
20            return true;
21        }
22        return false;
23      }
```

- variable *jeuEnCours* définie dans un autre fichier

- Fonction gererBouton : lit l'état du bouton connecté à la broche BUTTON_PIN

LOW (0) : bouton appuyé → fonction retourne true
HIGH (1) : pas d'appui → fonction retourne false

- Fonction verifierFin : vérifie si le bouton a été pressé pour mettre fin au jeu

Appel gererBouton, si elle reçoit true :

- déclenche sonVictoire
- affiche "VICTOIRE !!!" dans le moniteur série
- allume toutes les LEDs en dorée

LEDS

```
1   extern bool jeuEnCours; // variable de loop.ino
2
3   void allumerLeds(CRGB couleur, int delai) {
4     eteindreLeds();
5
6     for (int i = 1; i < NUM_LEDS; i++) {
7       leds[i] = couleur;
8
9       // Afficher toutes les 5 LEDs ou à la fin
10      if ((i - 1) % 5 == 4 || i == NUM_LEDS - 1) {
11        FastLED.show();
12        sonAllumageLED();
13
14        // ⏳ Attente divisée en petits pas pour pouvoir interro
15        int pas = 10;  // 10 ms d'attente
16        int attente = 0;
17        while (attente < delai) {
18          if (verifierFin()){
19            return;  // 👈 interrompt l'allumage proprement
20          }
21          delay(pas);
22          attente += pas;
23        }
24        delay(250);
25      }
26    }
27  }
28
29
30  void allumerTout(CRGB couleur) {
31    for (int i = 1; i < NUM_LEDS; i++) {
32      leds[i] = couleur;
33    }
34    FastLED.show();
35    delay(1000);
36  }
37
38  void eteindreLeds() {
39    for (int i = 1; i < NUM_LEDS; i++) {
40      leds[i] = CRGB::Black;
41    }
42    FastLED.show();
43  }
```

- variable *jeuEnCours* définie dans un autre fichier
- Fonction allumerLeds

Allume progressivement les LEDs de 1 par groupes de 5

Pour chaque groupe de 5 LEDs :

-Affiche l'état des LEDs

- Joue un son d'allumage (sonAllumageLED)

-Attend un délai donné, découpé en petits pas permettant d'interrompre l'attente.

Si la fonction verifierFin() détecte la fin du jeu pendant l'attente, interrompt l'allumage proprement

Éteint toutes les LEDs avant de commencer

LEDS

```
1   extern bool jeuEnCours; // variable de loop.ino
2
3   void allumerLeds(CRGB couleur, int delai) {
4     eteindreLeds();
5
6     for (int i = 1; i < NUM_LEDS; i++) {
7       leds[i] = couleur;
8
9       // Afficher toutes les 5 LEDs ou à la fin
10      if ((i - 1) % 5 == 4 || i == NUM_LEDS - 1) {
11        FastLED.show();
12        sonAllumageLED();
13
14        // ⏳ Attente divisée en petits pas pour pouvoir interro
15        int pas = 10;  // 10 ms d'attente
16        int attente = 0;
17        while (attente < delai) {
18          if (verifierFin()){
19            return;  // 👈 interrompt l'allumage proprement
20          }
21          delay(pas);
22          attente += pas;
23        }
24        delay(250);
25      }
26    }
27  }
28
29
30  void allumerTout(CRGB couleur) {
31    for (int i = 1; i < NUM_LEDS; i++) {
32      leds[i] = couleur;
33    }
34    FastLED.show();
35    delay(1000);
36  }
37
38  void eteindreLeds() {
39    for (int i = 1; i < NUM_LEDS; i++) {
40      leds[i] = CRGB::Black;
41    }
42    FastLED.show();
43  }
```

- Fonction **allumerTout**

Allume toutes les LEDs en même temps avec une couleur donnée.
Met à jour l'affichage et attend 1 seconde.

- Fonction **eteindreLeds**

Éteint toutes les LEDs en les mettant en noir.
Met à jour l'affichage.