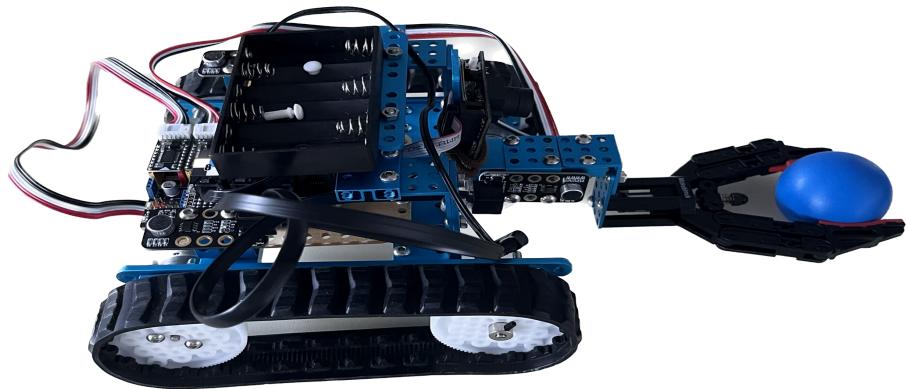
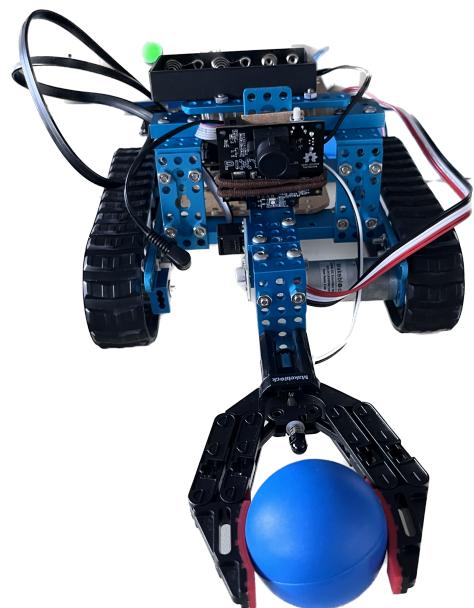


RAPPORT PROJET D'INTEGRATION IE4



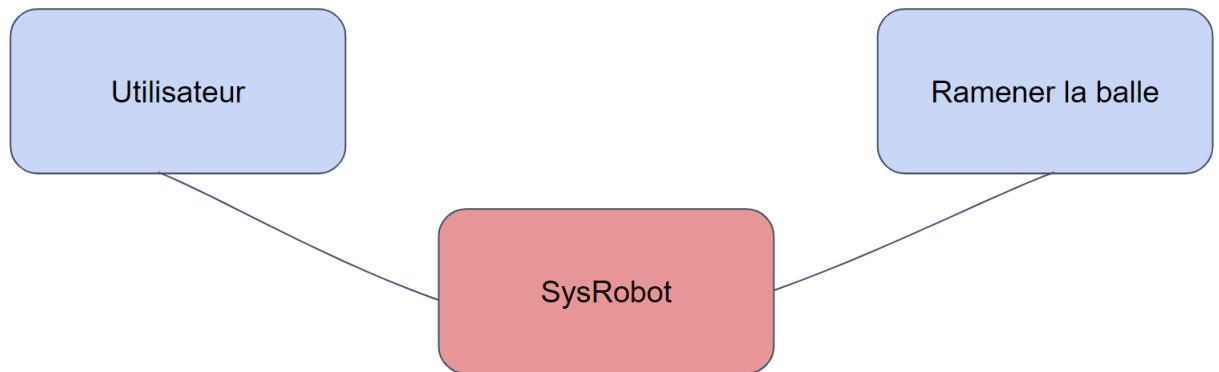
Eugène LECONTE

Table des matières:

Table des matières:	2
Présentation générale	3
Objectif initial	3
Attentes vs Réalités	4
Fonctionnement général	4
Fonctions principales	6
Bilan Personnel et managérial	7
Planning	7
Bilan	8
Démonstration	9
Slides	10
Code	11
Lien vers les vidéos de démonstration	27

Présentation générale

Objectif initial



Synthèse: Permettre à l'utilisateur de ramener une balle grâce au SysRobot.

1. On lance une balle d'une couleur et/ou forme significativement reconnaissable par une caméra.
2. Après un signal sonore, le robot tourne autour de lui même jusqu'à ce qu'il détecte la balle.
3. Il se dirige vers la balle, la ramasse avec une pince et signale avec un buzzer qu'il l'a récupérée.
4. On signale à nouveau notre présence pour qu'il puisse retrouver où l'utilisateur est (il peut avoir bougé depuis le début de l'application).
5. Le robot se dirige alors vers l'utilisateur, s'arrête à une dizaine de centimètre de lui et lâche la balle à ses pieds.

Avant de pouvoir mettre en place le processus de récupération de la balle par le robot, plusieurs équipements doivent être rassemblés. Les besoins initiaux comprennent une caméra couleur de type Pixy, trois capteurs sonores, un buzzer, deux moteurs et une pince.

La caméra couleur de type Pixy sera utilisée pour repérer la balle et transmettre les informations nécessaires au robot pour qu'il puisse la localiser. Les capteurs sonores sont nécessaires pour fournir des signaux sonores qui aideront le robot à localiser l'utilisateur. Le buzzer sera utilisé pour signaler la récupération de la balle par le robot.

Le robot nécessite également deux moteurs pour se déplacer vers la balle et l'utilisateur, ainsi que pour tourner sur lui-même. Enfin, une pince est nécessaire pour saisir et transporter la balle.

En somme, les besoins initiaux nécessaires pour la mise en place du processus de récupération de la balle par le robot comprennent une caméra couleur, des capteurs sonores, un buzzer, des moteurs et une pince.

Attentes vs Réalités

1. Ajustement de la caméra
2. Le paramétrage des capteurs sonores
3. Utiliser un buzzer pour prévenir que le robot a récupéré la balle n'apporte pas de plus-value au projet.
4. La triangulation des capteurs sonores

Plusieurs ajustements et paramétrages sont nécessaires pour que le robot soit en mesure de récupérer la balle et de la remettre à son utilisateur.

Tout d'abord, il est nécessaire d'ajuster la caméra pour qu'elle soit en mesure de repérer la balle. Le réglage de la caméra est crucial car elle est la principale source d'information pour le robot. Elle permet de détecter la balle et de suivre sa trajectoire jusqu'à ce qu'elle soit récupérée par le robot.

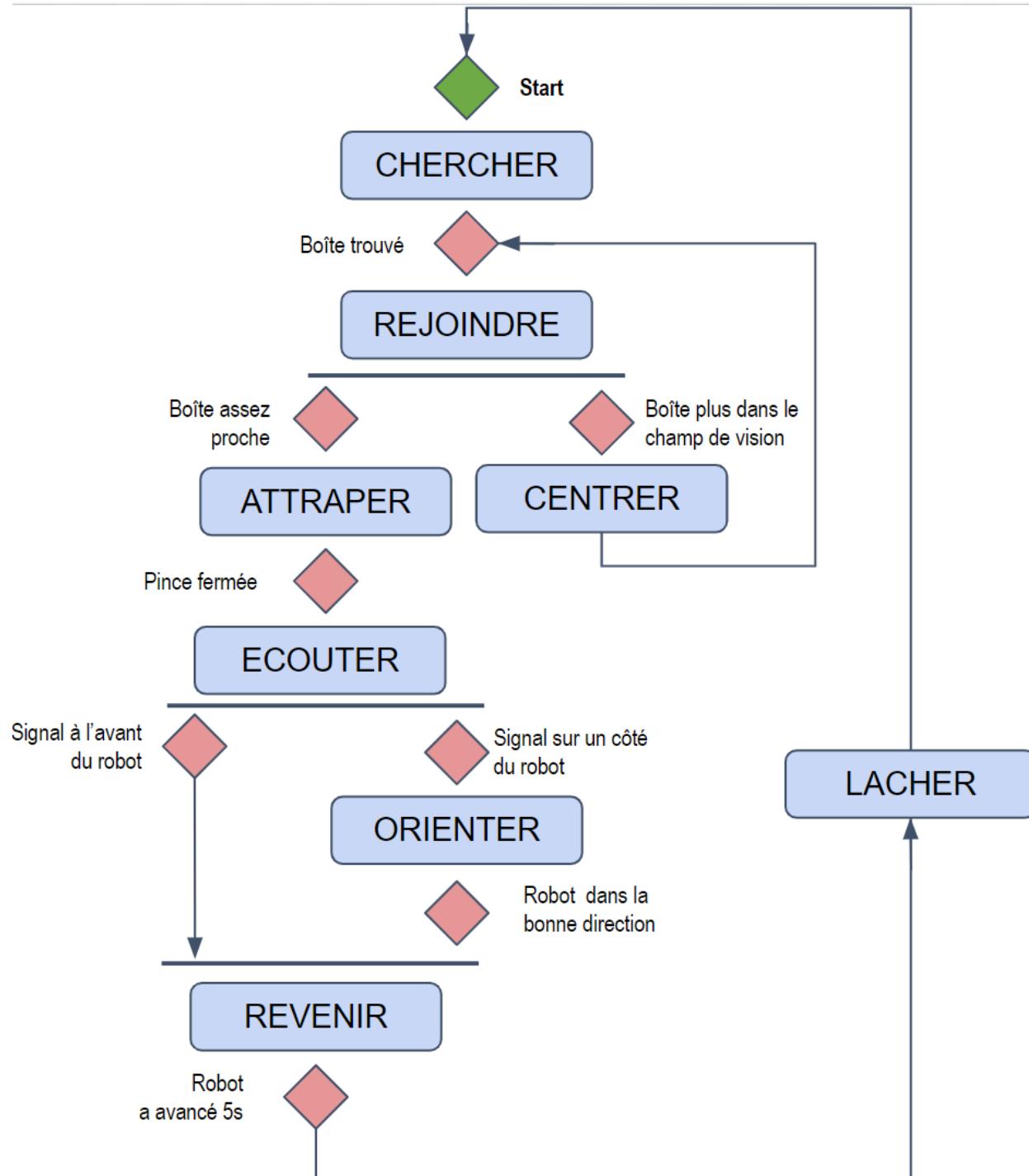
Ensuite, les capteurs sonores doivent être paramétrés pour fournir des signaux sonores appropriés qui aideront le robot à localiser l'utilisateur. Il est important de s'assurer que les signaux sonores soient suffisamment forts et distincts pour que le robot puisse les identifier et les suivre.

Cependant, l'utilisation d'un buzzer pour signaler la récupération de la balle n'apporte pas de plus-value au projet et peut être considérée comme superflue. Il est donc possible de ne pas utiliser cet équipement dans le processus de récupération de la balle par le robot.

Enfin, la triangulation des capteurs sonores est un élément important du processus de récupération de la balle. Cette technique permet de déterminer la position de l'utilisateur par rapport au robot, en utilisant les signaux sonores émis par les capteurs sonores. Cette information est ensuite utilisée pour permettre au robot de se diriger vers l'utilisateur et de lâcher la balle à ses pieds.

Fonctionnement général

Modélisation graphique des différentes séquences du robot:



Fonctions principales

Ci dessous, les cinq fonctions principales qui nous permettent de faire fonctionner tous les capteurs entre eux:

1. **machine_a_etat**

Gestion des différents états du programme

2. **gestion_pince**

Gestion de l'ouverture et de la fermeture de la pince

3. **deplacement**

Gestion de toutes les fonctions de déplacement en adaptant les directions et vitesses

4. **camera**

Récupération des informations reçues par la caméra

5. **Orientation**

Renvoi la direction à prendre en fonction du son perçu

Plusieurs éléments clés doivent être pris en compte pour la mise en place de la récupération de la balle par le robot, notamment la gestion des différents états du programme, la gestion de la pince, les déplacements, la caméra et l'orientation.

La machine à états est un élément crucial pour la gestion des différents états du programme. Elle permet de définir les différentes étapes du processus de récupération de la balle, en s'assurant que chaque étape est correctement exécutée avant de passer à l'étape suivante.

La gestion de la pince est également importante pour que le robot puisse saisir et transporter la balle. Il est nécessaire de mettre en place un système de gestion de l'ouverture et de la fermeture de la pince pour que le robot puisse manipuler la balle de manière efficace.

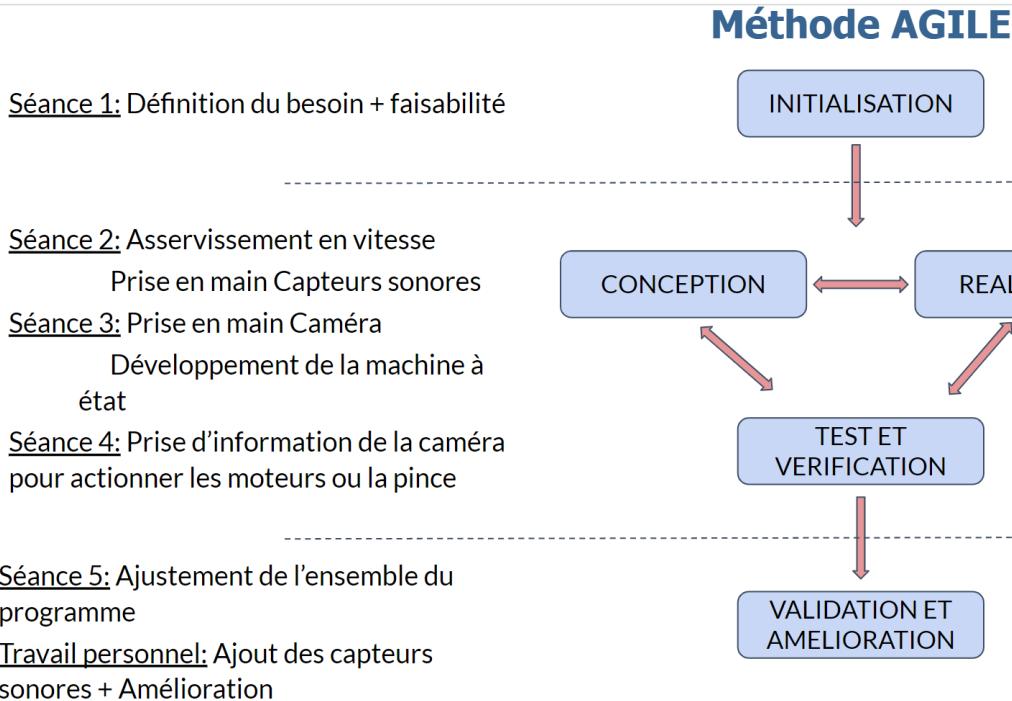
La gestion des déplacements est également un élément crucial pour le processus de récupération de la balle. Il est important d'adapter les directions et les vitesses de déplacement en fonction de la position de la balle et de l'utilisateur. Le robot doit être en mesure de se déplacer de manière précise et efficace pour récupérer la balle et la remettre à l'utilisateur.

La caméra est également un élément important de la récupération de la balle par le robot. Elle doit être en mesure de récupérer les informations nécessaires pour localiser la balle et suivre sa trajectoire jusqu'à ce qu'elle soit récupérée par le robot.

Enfin, l'orientation est un élément clé pour permettre au robot de se diriger vers l'utilisateur et de lâcher la balle à ses pieds. Il est nécessaire de mettre en place un système de gestion de l'orientation, qui renvoie la direction à prendre en fonction du son perçu par les capteurs sonores. Cela permettra au robot de se déplacer efficacement vers l'utilisateur et de lâcher la balle à ses pieds.

Bilan Personnel et managérial

Planning



Le projet a été découpé en plusieurs séances pour faciliter son développement et sa mise en œuvre.

La séance 1 consistait à définir le besoin et la faisabilité du projet. Cette séance a permis de définir les objectifs du projet et de vérifier la faisabilité de la récupération de la balle par un robot en utilisant les différents composants disponibles.

La séance 2 a été consacrée à l'asservissement en vitesse et à la prise en main des capteurs sonores. Cette séance a permis de mettre en place un système d'asservissement en vitesse pour contrôler les moteurs du robot et de prendre en main les capteurs sonores nécessaires pour la détection de la balle.

La séance 3 était dédiée à la prise en main de la caméra et au développement de la machine à état. Cette séance a permis de mettre en place la récupération d'informations par la caméra pour permettre au robot de localiser la balle et de suivre sa trajectoire, ainsi que de développer la machine à état pour gérer les différents états du programme.

La séance 4 était consacrée à la récupération d'informations de la caméra pour actionner les moteurs ou la pince. Cette séance a permis de mettre en place un système pour récupérer les informations de la caméra et les utiliser pour actionner les moteurs ou la pince.

La séance 5 était dédiée à l'ajustement de l'ensemble du programme. Cette séance a permis de tester et d'ajuster l'ensemble du programme pour s'assurer qu'il fonctionne correctement.

Enfin, le travail personnel consistait à ajouter les capteurs sonores et à améliorer le programme en fonction des résultats obtenus lors des différentes séances.

Chaque séance était consacrée à un aspect spécifique du projet, ce qui a permis de faciliter le développement et la mise en place de la récupération de la balle par le robot.

Bilan

La gestion de projet est une étape importante dans la réussite d'un projet. Dans le cas de la récupération de balle par un robot, il s'agit d'un projet ambitieux qui nécessite une bonne organisation et une gestion rigoureuse pour atteindre les objectifs fixés.

L'un des axes d'amélioration possibles pour ce projet serait l'amélioration de la précision des capteurs sonores. En effet, les capteurs sonores utilisés peuvent ne pas être suffisamment précis pour détecter avec précision la position de la balle. Il est donc possible d'explorer d'autres options pour améliorer la précision de la détection, comme l'utilisation de capteurs plus performants ou l'ajout d'autres capteurs.

Il est également important de considérer les autres utilisations potentielles de ce robot, en dehors de la récupération de balle. Le robot pourrait être utilisé dans d'autres contextes où la détection et la récupération d'objets sont nécessaires, comme dans l'industrie ou la logistique.

L'accompagnement et la pédagogie sont également des éléments importants de la gestion de projet. Il est important de s'assurer que tous les membres de l'équipe sont bien informés et formés sur les différentes composantes du projet, afin de garantir une bonne compréhension et une bonne utilisation de chaque composante.

Enfin, il est essentiel de s'assurer que le projet est bien planifié et que les délais sont respectés. Il est important de fixer des objectifs clairs et des échéances précises pour chaque étape du projet, afin de garantir une avancée régulière et une gestion efficace des ressources.

Démonstration

Sénaire

1. Sur le sol, on pose une boîte rectangulaire et jaune. On place le robot à moins d'un mètre de l'objet, peu importe la direction.
2. Le robot tourne sur lui-même jusqu'à avoir la totalité de l'objet visualisable.
3. Une fois chose faite, il s'avance vers l'objet en se recentrant si besoin. Par exemple: Si, en avançant, l'objet est trop à gauche, il se recentre vers la gauche.
4. Une fois que le robot est assez proche de l'objet, il ferme la pince pour prendre l'objet et s'arrête. On va souffler sur l'un des capteurs sonores pour simuler un bruit.
5. Le robot distingue sur quel capteur la valeur est la plus forte. Il avance dans l'une de ces trois directions et s'arrête au bout de 5 secondes.
6. Il lâche la boîte.

Ce cycle est réalisable plusieurs fois.

Slides

Présentation de notre objectif initial réalisé en séance 1:

<https://docs.google.com/presentation> 


Présentation finale de notre projet:

<https://docs.google.com/presentation> 


Code

```

///#include <SPI.h> and #include <Pixy.h>: These are the header files that
allow the code to interface with the Pixy camera via SPI communication.

#include <SPI.h>

#include <Pixy.h>

#include <Arduino.h>

#include <MeMegaPi.h>

#include <Wire.h>

// This is the main Pixy object

Pixy pixy; //This initializes a Pixy object named pixy, which will be
used to communicate with the Pixy camera.

//def variable vitesse et deplacement

#define BELT_PITCH 2.032                                // distance between teeth

#define NTEETH 90.0                                     // number of teeth

#define RPM_2_MMS BELT_PITCH* NTEETH / 60.0 // conversion of the speed
from rpm to mm/s

#define VOLTS_2_PWM 255.0 / 12.0                      // conversion voltage to pwm
[-255 255]

#define MAX_VOLTAGE 9                                  // max allowed voltage

//Variable de déplacement

#define STOP 0 //Le robot est a l'arret

#define AVANCER 1 //le robot avance

#define RECULER 2 //le robot recule

#define GAUCHE 3 //le robot tourne à gauche

#define DROITE 4 //le robot tourne à droite

//Variable de situation

```

```
#define CHERCHER 0
#define CENTRER 1
#define REJOINDRE 2
#define ATTRAPER 3
#define ECOUTER 4
#define ORIENTER 5
#define REVENIR 6
#define LACHER 7
#define TOURNER 8

//Pour les conditions de vivion de l'objet
#define VISIBLE 140
#define ATTEINT 60

//Pour la pince
#define OPEN 1
#define CLOSE -1

#define MIN_SOUND 480 // Minumun son écouté

//moteur camera
int ETAT = CHERCHER;
int j;
uint16_t blocks;

//moteur pince
uint8_t motorSpeed_pince = 100;
```

```
int etat_pince = 0;

int arrive = 0;
int oriente = 0;
int tourner = 0;
int chercher = 0;

//Moteur roues

float mesure_moteur1;
float mesure_moteur2;
//float consigne_vitesse1 = 50; // Vitesse en mm/s
//float consigne_vitesse2 = 50; // Vitesse en mm/s
float commande_moteur1;
float commande_moteur2;

// PID parameters Initialisation

long prevT = 0;
float eprev1 = 0;
float eprev2 = 0;
float eintegral1 = 0;
float eintegral2 = 0;
int etat_moteur = 0;
int temps = 0;

// Gestion du so

int countF = 0;
int countBR = 0;
```

```
int countBL = 0;

float sommeF = 0;
float sommeBR = 0;
float sommeBL = 0;

float moyF;
float moyBL;
float moyBR;

unsigned long start_time;
unsigned long current_time;
int duration_rotation = 5000;
// Port de connexion

MeEncoderOnBoard Moteur_1(SLOT1); // Port 1 = Moteur droit || motor with
encoder object instanciation

MeEncoderOnBoard Moteur_2(SLOT3); // Port 2 = Moteur gauche

MeMegaPiDCMotor motorPince(PORT4B);

MeSoundSensor Sound_F(PORT_6);
MeSoundSensor Sound_BR(PORT_7);
MeSoundSensor Sound_BL(PORT_8);

void setup() {
    Serial.begin(9600);
    Serial.println("Starting...");
    gestion_pince(OPEN);
    pixy.init();
    //machine_a_etat(); //If blocks are detected, it prints out the number
```

```

of detected blocks and their coordinates through the Serial monitor.

attachInterrupt(Moteur_1.getIntNum(), isr_process_encoder1, RISING);
attachInterrupt(Moteur_2.getIntNum(), isr_process_encoder2, RISING);

start_time = millis();

}

void loop() {

machine_a_etat();

define_etat();

}

// definition des etats

void define_etat() {

//ETAT 0, je

if (camera() == 1) {

if (pixy.blocks[j].x < 60 && etat_pince == 0 && chercher == 0) {

ETAT = CHERCHER;

}

//ETAT 1, centrer la balle

if ((pixy.blocks[j].x < 120 || pixy.blocks[j].x > 180) &&
(pixy.blocks[j].width < ATTEINT || pixy.blocks[j].height < ATTEINT) &&
etat_pince == 0 && chercher == 0) {

ETAT = CENTRER;

}

//ETAT 2, j'avance

if ((pixy.blocks[j].x >= VISIBLE && pixy.blocks[j].x < 320 - VISIBLE) &&
(pixy.blocks[j].width < ATTEINT || pixy.blocks[j].height < ATTEINT) &&
etat_pince == 0 && chercher == 0) {

ETAT = REJOINDRE;

}
}
}

```

```
// ETAT 3, je suis assez proche, j'attrape

if (pixy.blocks[j].x >= 145 && (pixy.blocks[j].width >= ATTEINT || pixy.blocks[j].height >= ATTEINT) && etat_pince == 0 && oriente == 0 && chercher == 1) {

    ETAT = ATTRAPER;

}

}

// ETAT 4, j'ai la balle, j'attends d'entendre la direction

if (etat_pince == 1 && oriente == 0) {

    ETAT = ECOUTER;

}

// ETAT 5, Je m'oriente dans la bonne direction

if (etat_pince == 1 && (oriente == 2 || oriente == 3)) {

    ETAT = ORIENTER;

}

// ETAT 6, je rejoins mon maître

if (etat_pince == 1 && oriente == 1) {

    ETAT = REVENIR;

}

// ETAT 7, je dépose la balle

if (etat_pince == 1 && arrive == 1) {

    ETAT = LACHER;

}
```

```
// ETAT 7, je dépose la balle

if (etat_pince == 1 && arrive == 1 && tourner == 1) {
    ETAT = TOURNER;
}

}

//SWITCH CASE MACHINE A ETAT

void machine_a_etat() {
    switch (ETAT) // Machine d'état (Comme un Grafset)
    {
        case CHERCHER:
            Serial.println("Case Chercher !");
            chercher = 0;
            Serial.println(oriente);
            deplacement(GAUCHE, 20);
            break;

        case CENTRER:
            Serial.println("Case Centrer !");
            if (pixy.blocks[j].x < 155) {
                deplacement(GAUCHE, 20);
            } else {
                deplacement(DROITE, 20);
            }
            break;
    }
}
```

```
case REJOINDRE:

    Serial.println("Case Rejoindre !");
    deplacement(AVANCER, 40);

    if (pixy.blocks[j].x >= 145 && (pixy.blocks[j].width >= ATTEINT || pixy.blocks[j].height >= ATTEINT)) {

        chercher = 1;

    }

    break;

case ATTRAPER:

    Serial.println("Case Attraper !");
    deplacement(AVANCER, 30);
    gestion_pince(CLOSE);
    etat_pince = 1;
    start_time = millis();
    break;

case ECOUTER:

    //Serial.println("Case Ecouter !");
    current_time = millis() - start_time;
    if (current_time >= 2000) {

        orientation();
        Serial.println(orientee);
        start_time = millis();
        break;

    } else {

        deplacement(STOP, 0);
    }
}
```

```
        }

        break;

case ORIENTER:

    Serial.println("Case Orienter !");
    current_time = millis() - start_time;
    if (current_time >= duration_rotation) {

        oriente = 1;

        start_time = millis();

    } else {

        if (oriente == 2) {

            deplacement(DROITE, 20);

        }

        if (oriente == 3) {

            deplacement(GAUCHE, 20);

        }

    }

}

break;

case REVENIR:

    Serial.println("Case Revenir !");
    current_time = millis() - start_time;
    if (current_time >= 4000) {

        //arrêter les moteurs

        arrive = 1;

    } else {
```

```
//faire pivoter les moteurs dans des directions opposées

deplacement(AVANCER, 30);

}

break;

case LACHER:

Serial.println("Case Lacher !");

deplacement(RECULER, 30);

//gestion_pince(OPEN);

tourner = 1;

start_time = millis();

break;

case TOURNER:

Serial.println("Case TOURNER !");

current_time = millis() - start_time;

if (current_time >= 1500) {

etat_pince = 0;

arrive = 0;

orientee = 0;

tourner = 0;

chercher = 0;

moyF = 0;

moyBR = 0;

moyBL = 0;

gestion_pince(OPEN);

ETAT = CHERCHER;
```

```
    Serial.println(etat_pince);

    Serial.println(orienter);
    Serial.println(chercher);

    break;

} else {

    deplacement(RECULER, 20);

}

break;

}

//-----Gestion de la pince-----

//ouverture ou fermeture de la pince

void gestion_pince(float sens) {

    Serial.println("Pince");

    unsigned long current_time = millis();

    motorPince.run(sens * motorSpeed_pince);

    delay(1000);

    motorPince.stop();

}

//-----Gestion de la pince-----

//-----Gestion des déplacement-----

//fonction de deplacement

void deplacement(float direction, float vitesse) {

    float C_V1, C_V2;

    if (direction == STOP) {

        C_V1 = vitesse * 0;
        C_V2 = vitesse * 0;
```

```
}

if (direction == AVANCER) {

    C_V1 = vitesse;
    C_V2 = vitesse;

}

if (direction == RECULER) {

    C_V1 = -vitesse;
    C_V2 = -vitesse;

}

if (direction == GAUCHE) {

    C_V1 = vitesse;
    C_V2 = -vitesse;

}

if (direction == DROITE) {

    C_V1 = -vitesse;
    C_V2 = vitesse;

}

/* Lecture des encodeurs */

mesure_moteur1 = Moteur_1.getCurrentSpeed();

mesure_moteur2 = Moteur_2.getCurrentSpeed();

/* Régulation PID */

// PID constants

float kp = 0.01;

float kd = 0;

float ki = 0.5;

long currT = micros();

float deltaT = ((float)(currT - prevT)) / (1.0e6);
```

```

prevT = currT;

// error

int e1 = C_V1 - mesure_moteur1;
int e2 = C_V2 + mesure_moteur2;

// derivative

float dedt1 = (e1 - eprev1) / (deltaT);
float dedt2 = (e2 - eprev2) / (deltaT);

// integral

eintegral1 = eintegral1 + e1 * deltaT;
eintegral2 = eintegral2 + e2 * deltaT;

// control signal

float u1 = kp * e1 + kd * dedt1 + ki * eintegral1;
float u2 = kp * e2 + kd * dedt2 + ki * eintegral2;

setMotorsVoltage(u1, -u2);

Moteur_1.updateSpeed();

Moteur_2.updateSpeed();

}

//fonction des roues

void isr_process_encoder1(void) {
    if (digitalRead(Moteur_1.getPortB()) == 0) {
        Moteur_1.pulsePosMinus();
    } else {
        Moteur_1.pulsePosPlus();
    }
}

void isr_process_encoder2(void) {
}

```

```

if (digitalRead(Moteur_2.getPortB()) == 0) {

    Moteur_2.pulsePosMinus();

} else {

    Moteur_2.pulsePosPlus();

}

}

void setMotorsVoltage(float voltage1, float voltage2) {

    Moteur_1.setMotorPwm(constrain(voltage1, -MAX_VOLTAGE, MAX_VOLTAGE) * VOLTS_2_PWM);

    Moteur_2.setMotorPwm(constrain(voltage2, -MAX_VOLTAGE, MAX_VOLTAGE) * VOLTS_2_PWM);

}

//-----Gestion des déplacement-----
//-----Gestion caméra-----

int camera() {

    static int i = 0;

    int j;

    uint16_t blocks;

    char buf[32];

    // grab blocks!

    blocks = pixy.getBlocks();

    // If there are detect blocks, print them!

    if (blocks) {

        i++;

        // do this (print) every 50 frames because printing every

        // frame would bog down the Arduino

        if (i % 50 == 0) {

            sprintf(buf, "Detected %d:\n", blocks);

```

```
    Serial.print(buf);

    for (j = 0; j < blocks; j++) {

        sprintf(buf, " block %d: ", j);

        Serial.print(buf);

        pixy.blocks[j].print();

        return 1;

    }

}

} else {

    return 0;

}

}

//-----Gestion caméra-----

//-----Gestion SoundSensor-----

void orientation() {

    Serial.println(Sound_F.strength() - 250);

    Serial.println(Sound_BL.strength() - 280);

    Serial.println(Sound_BR.strength() - 450);

    if (Sound_F.strength() > MIN_SOUND) {

        moyF = Sound_F.strength() - 250;

    }

    if (Sound_BL.strength() > MIN_SOUND) {

        moyBL = Sound_BL.strength() - 280;

    }

    if (Sound_BR.strength() > MIN_SOUND) {

        moyBR = Sound_BR.strength() - 450;

    }

}
```

```
if (moyF > moyBR && moyF > moyBL) {  
    oriente = 1;  
}  
  
if (moyBR > moyF && moyBR > moyBL) {  
    oriente = 2;  
}  
  
if (moyBL > moyF && moyBL > moyBR) {  
    oriente = 3;  
}  
}  
  
//-----Gestion SoundSensor-----
```

Lien vers les vidéos de démonstration

Vous trouverez sur ce drive tous les liens qui permettront de visualiser des démonstrations que nous avons effectués. Ce dossier contient également notre présentation et ce rapport finale.

[https://drive.google.com/drive/\[REDACTED\]](https://drive.google.com/drive/[REDACTED])