heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Elisa-3 à La Maison d'Ailleurs

Projet réalisé dans le cadre du cours SFT 2015

A. Desclouds, A.Hug, S. Righitto, F. Saam

15/06/2015

Sommaire

heig-vd

- Le robot Elisa-3
- Concept du projet
- Implémentation
- Difficultés rencontrées
- Bilan final

Elisa-3

- Lumière colorée
- Petites leds vertes
- Emetteurs infrarouge
- Possède accéléromètre
- Autonomie de 3 heures

heig-vd

Concept du projet

heig-vd

- Nouveau concept : les danseurs se réveillent
- Concept : rechargement automatique
- Arène
- Raspberry Pi et Phidget Touchrotation

Implémentation

heig-vd

- Reprise du projet déjà existant
- Nouvelle bibliothèque
- Beaucoup de modifications à apporter
- Plusieurs difficultés observées

Démarrage du spectacle

heig-vd

- Le robots sont à l'écoute de:
  - L'antenne commandée par le Rapsberry Pi
  - La proximité avec d'autres robots
  - Son propre niveau de batterie
- Le spectacle se termine après 2 minutes
- Les robots sont à nouveau en attente du démarrage

Démarrage

- L'antenne envoie un code (setFullRed) vers un robot qui devient le « danseur fou »
- Les autres robots attendent l'approche du danseur pour démarrer
- Si le niveau de batterie est trop faible, le robot se met à la recherche du chargeur.

```
void checkStart() {
  //to be able to start with the pc antenna
  handleRFCommands();
  turnOnGreenLeds();

  if (pwm_red == 0 && pwm_green == 255 && pwm_blue == 255) {
    robotState = DANCE_1;
    startDance = 0;
    robotStartedTime = getTime100MicroSec();
    turnOffGreenLeds();
  }
  else if (checkNearbyObjects()) {
    robotState = BASE_MODE;
    robotStartedTime = getTime100MicroSec();
    startDance = 0;
    turnOffGreenLeds();
  } else if (robotState == LOW_BATTERY) {
    robotStartedTime = getTime100MicroSec();
    turnOffGreenLeds();
  }
}
```

heig-vd

Mode de base

heig-vd

- Le robot se déplace de manière aléatoire
- Il change de couleur aléatoirement toutes les 5 secondes
- Il effectue une danse toutes les 40 secondes

Suivi de ligne et rechargement

heig-vd

- D'abord: Lignes noires sur blanc
- Puis: Lignes blanches (dégradées) sur noir

- Idée : quand le niveau de batterie du robot est trop bas, il cherche des lignes qui le guideront à la station de rechargement.

Suivi de ligne
(1) lignes noires

- Reprise de l'ancien code
- Problèmes de fonctionnement
  - Nouvelle librairie ?
- Calcul des valeurs blanc et noir
- Repositionnement du robot (aimantation)

```
// Dans le setup
unsigned int line_in = 0, line_out = 0;
line_out = proximityResult[8];
line_in = line_out-50;

// Dans la boucle
if (proximityResult[1] >= 700 || proximityResult[7] >= 700) {
    if ((getTime100MicroSec()-counterPerso) >= PAUSE_1_SEC) {
        counterPerso = getTime100MicroSec();
        if (testBool == 1) {testBool = 0;}
        else {testBool = 1;}
    }
    if (testBool == 1) {
        setRightSpeed(15);
        setLeftSpeed(-5);
    } else {
        setRightSpeed(-5);
        setLeftSpeed(15);
    }
}
```

heig-vd

A. Desclouds, A.Hug, S. Righitto, F. Saam

Suivi de ligne
(2) lignes blanches

- Lignes normales → lignes floutées
- Injecte la différence des senseurs dans la vitesse
- Idée et code de l'assistant (merci beaucoup Hector ! ☺)
- braitenbergLineFollower()

```
void braitenbergLineFollower() {
  front_diff = ((int)proximityResult[9] - (int)proximityResult[10]) >> 5;
  //-------------------------------------------------------------------------------------
  if (inLine) {
    if ( (proximityResult[9] < LINE_OUT_THR_BK) && (proximityResult[10] < LINE_OUT_THR_BK) ) {
      inLine = false;
      setGreenLed(0, 1);
      enableObstacleAvoidance();
    }
  }
  else {
    if ( (proximityResult[9] > LINE_IN_THR_BK) || (proximityResult[10] > LINE_IN_THR_BK) ) {
      inLine = true;
      setGreenLed(0, 0);
      disableObstacleAvoidance();
    }
  }
  //-------------------------------------------------------------------------------------
  if ( !inLine && (proximityResult[8] > LINE_IN_THR_BK) && (proximityResult[11] > LINE_IN_THR_BK) ) {
    if (accY > 0) {       // check the slope and turn the robot to go down
      setLeftSpeed(15);
      setRightSpeed(-15);
    }
    else {
      setLeftSpeed(-15);
      setRightSpeed(15);
    }
    startTempAction(PAUSE_500_MSEC);
  }
  else {
    setLeftSpeed( CONSTANT_SPEED_FOLLOW - front_diff);
    setRightSpeed(CONSTANT_SPEED_FOLLOW + front_diff);
  }
}
```

heig-vd

Rechargement

- Logique du rechargement

```
    case LOW_BATTERY:
      lowBattery();
      break;
    case IN_CHARGER:
      setLEDcolor(255, 0, 255);
      setLeftSpeed(0);
      setRightSpeed(0);
      if (!charging()) {
        if (batteryLevel < BATTERY_LEVEL_MORE_CHARGING) { //we need more charging
          setLEDcolor(255, 255, 0);
          previouslyCharging = true;
          robotState = LOW_BATTERY;
        }
        else {
          previouslyCharging = false;
          robotState = GOING_OFF_CHARGER;
        }
      }
      break;
    case GOING_OFF_CHARGER:
      goingOffCharger();
      break;
  }
}
```

heig-vd

Rechargement

- Contrôle du contact avec le chargeur

```
unsigned int chargeContactDetected = 0;
boolean charging() {
  if (batteryLevel > BATTERY_LEVEL_STOP_CHARGING) {
    return false;
  }
  else if (CHARGE_ON) {
    chargeContactDetected++;
    if (chargeContactDetected > 20) {
      return true;
    }
    return false;
  }
  else {
    chargeContactDetected = 0;
    return false;
  }
}
```

heig-vd

Rechargement

- Séparation de la station de rechargement

```
void goingOffCharger() {

  switch (goingOff_state) {
    case 0:
      setLEDcolor(0, 250, 250);
      setLeftSpeed(0);
      setRightSpeed(0);
      startGoingOff = getTime100MicroSec();
      goingOff_state = 1;
      break;
    case 1:
      currentTime = getTime100MicroSec();
      if (currentTime - startGoingOff > PAUSE_2_SEC) {
        setLeftSpeed(-NORMAL_SPEED);
        setRightSpeed(- NORMAL_SPEED);
        goingOff_state = 2;
        startGoingOff = currentTime;
      }
      break;
    case 2:
      currentTime = getTime100MicroSec();
      if (currentTime - startGoingOff > PAUSE_2_SEC) {
        rotate(1, NORMAL_SPEED);
        goingOff_state = 3;
        startGoingOff = currentTime;
      }
      break;
    case 3:
      currentTime = getTime100MicroSec();
      if (currentTime - startGoingOff > PAUSE_1_SEC) {
        enableObstacleAvoidance();
        setLeftSpeed(0);
        setRightSpeed(0);
        robotState = BASE_MODE;
        startDance = currentTime;
```

A. Desclouds, A.Hug, S. Righitto, F. Saam

heig-vd

Danses

- Clignotement des leds
- Jeu avec les couleurs
- Jeu avec les déplacements
- Fonction « rotate » et « turn »
- Difficultés
  - Vitesse/durée de l'action
  - Détection d'obstacles

heig-vd

Danses

- Choix aléatoire d'une des 2 danses

```
if ((currentTime - startDance) >= (PAUSE_40_SEC)) {
  //startDance = currentTime; this will be updated at the end
  //of the dance in order to let the base mode continue PAUSE_40_SEC
  randDance = rand() % 2;
  switch (randDance) {
    case 0:
      robotState = DANCE_1;
      break;
    case 1:
      robotState = DANCE_2;
      break;
    default:
      robotState = DANCE_1;
  }

}
```

heig-vd

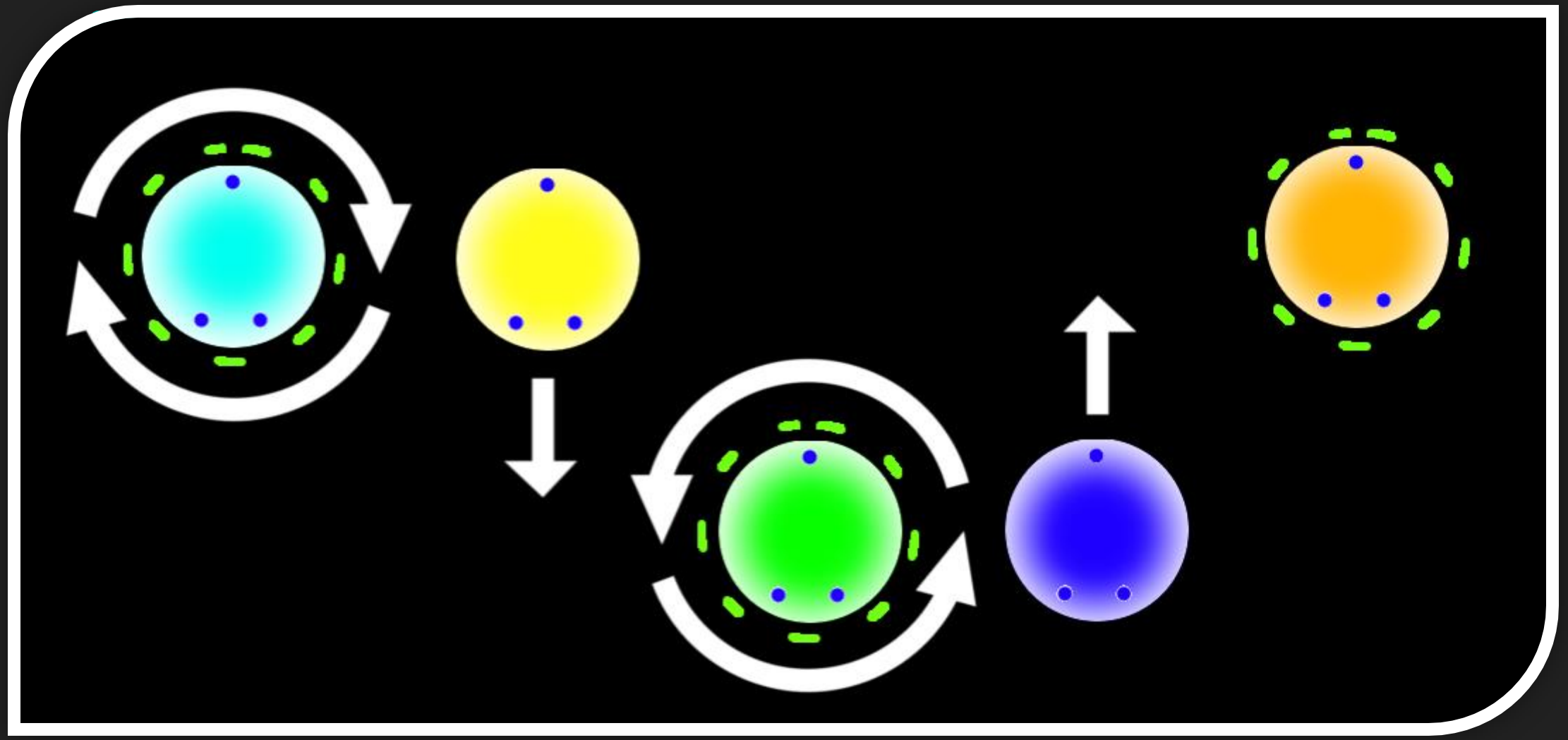
Danses

- dance_1()

```
signed int dance_1_state = 0;
void dance_1() {

  switch (dance_1_state) {
    case 0:
      enableObstacleAvoidance();
      setLEDcolor(255, 0, 57); //turquoise
      rotate(0, HIGH_SPEED);
      start = getTime100MicroSec();
      dance_1_state = 1;
      break;
    case 1:
      currentTime = getTime100MicroSec();
      if (currentTime - start > PAUSE_4_SEC) {
        start = currentTime;
        setLEDcolor(0, 63, 255); //dark yellow
        turnOnGreenLeds();
        setRightSpeed(NORMAL_SPEED);
        setLeftSpeed(NORMAL_SPEED);
        dance_1_state = 2;
      }
      break;
    case 2:
      currentTime = getTime100MicroSec();
      if (currentTime - start > PAUSE_2_SEC) {
        turnOffGreenLeds();
        start = currentTime;
        setLEDcolor(171, 0, 255); //green
        rotate(1, HIGH_SPEED);
        dance_1_state = 3;
      }
      break;
```

heig-vd



heig-vd

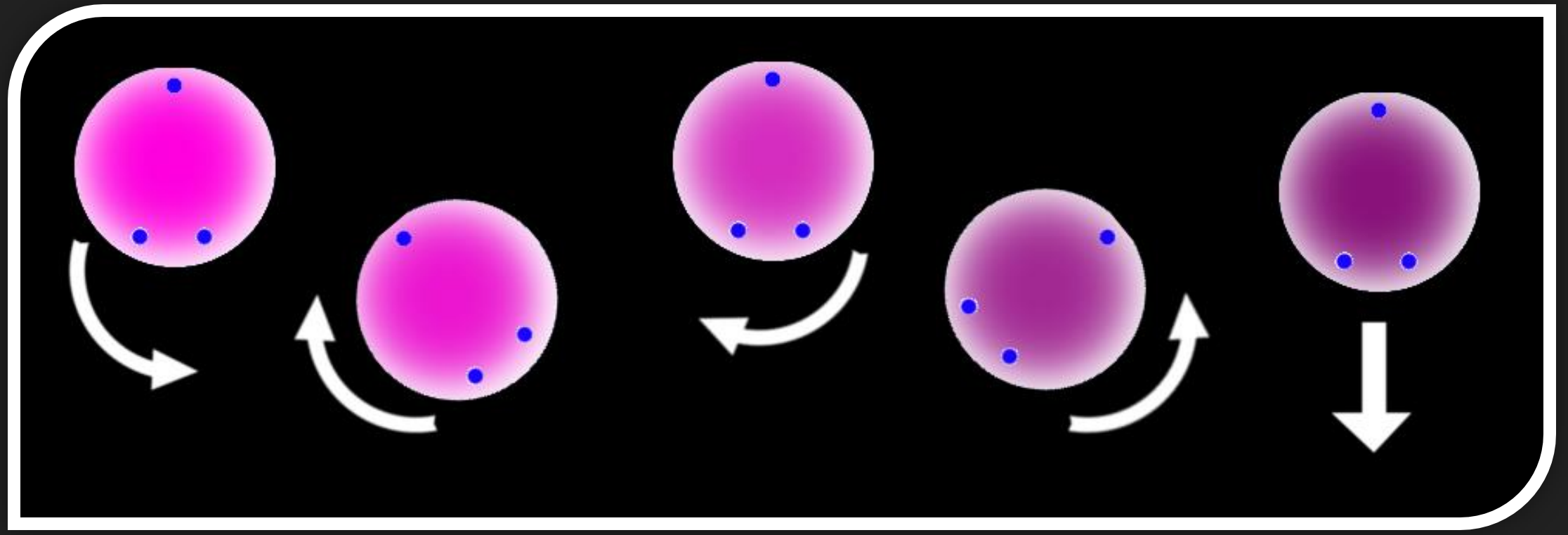
A. Desclouds, A.Hug, S. Righitto, F. Saam

Danses

- dance_2()

```
signed int dance_2_state = 0;
void dance_2() {

  switch (dance_2_state) {
    case 0:
      enableObstacleAvoidance();
      setLEDcolor(0, 255, 240);
      turn(0, HIGH_SPEED);
      start = getTime100MicroSec();
      dance_2_state = 1;
      break;
    case 1:
      currentTime = getTime100MicroSec();
      if (currentTime - start > PAUSE_1_SEC) {
        start = currentTime;
        setLEDcolor(20, 255, 220);
        turn(0, -HIGH_SPEED);
        dance_2_state = 2;
      }
      break;
    case 2:
      currentTime = getTime100MicroSec();
      if (currentTime - start > PAUSE_1_SEC) {
        start = currentTime;
        setLEDcolor(40, 255, 200);
        turn(1, HIGH_SPEED);
        dance_2_state = 3;
      }
      break;
    case 3:
      currentTime = getTime100MicroSec();
      if (currentTime - start > PAUSE_1_SEC) {
        start = currentTime;
        setLEDcolor(60, 255, 180);
        turn(1, -HIGH_SPEED);
        dance_2_state = 4;
```



heig-vd

A. Desclouds, A.Hug, S. Righitto, F. Saam

Danses
rotate()

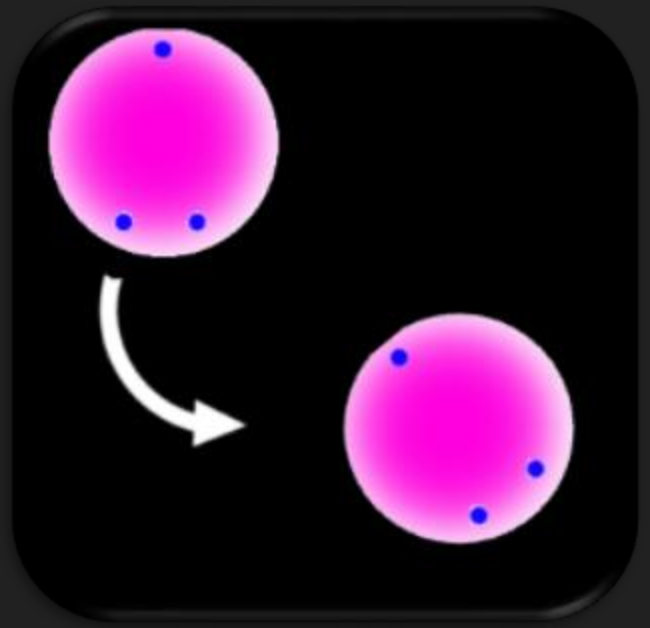
```
//rotationDirection : 0 --> clockwise ; 1 --> cunterclokwise
void rotate(int rotationDirection, unsigned int rotationSpeed) {
  switch (rotationDirection) {

    case 0:
      setLeftSpeed(rotationSpeed);
      setRightSpeed(- rotationSpeed);
      break;
    case 1:
      setLeftSpeed(- rotationSpeed);
      setRightSpeed(rotationSpeed);
      break;
  }
}
```

heig-vd

A. Desclouds, A.Hug, S. Righitto, F. Saam

Danses
turn()



```
//turnDirection : 0 --> clockwise ; 1 --> cunterclokwise
void turn(int turnDirection, unsigned int turnSpeed) {
  switch (turnDirection) {

    case 0:
      setLeftSpeed(turnSpeed);
      setRightSpeed(0);
      break;
    case 1:
      setLeftSpeed(0);
      setRightSpeed(turnSpeed);
      break;
  }
}
```

heig-vd

A. Desclouds, A.Hug, S. Righitto, F. Saam

Rapsberry Pi et Phidget TouchRotation 1016

heig-vd

- Démarrage du spectacle
  - Un visiteur passe la main sur un détecteur
- Code C
  - Permet de mettre les robots en bleu et un en rouge
- Python
  - Appelle le code C quand un mouvement est repéré par l'interface TouchRotation

Difficultés rencontrées

heig-vd

- Changement de librairie (librairie non stable et contenant des bugs)
- Impossibilité de débuguer
- Sensibilité des robots: chaque robot réagit de manière différente
- Manque de documentation
- Difficulté de reproduire un environnement constant (lumière, reflets, dégradation de l'arène du à l'usage)

Bilan final

heig-vd

- Evolution constante du concept
- Résultat obtenu ☺
- Jouer avec les robots ☺
- Expérience unique d'un projet pour un musée
- Améliorations possibles
  - Récrire la librairie

Merci pour votre attention !

heig-vd

https://github.com/SimoneRighitto/SFT15_Elisa

Démonstration!

heig-vd