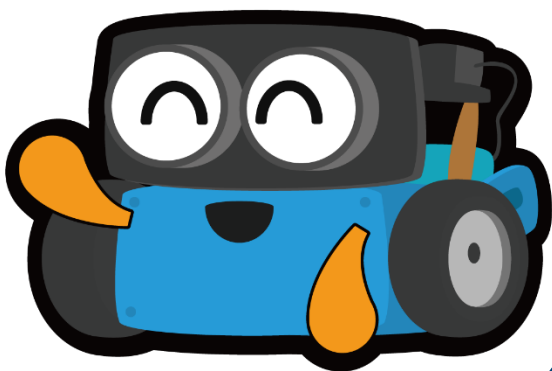


Rapport projet Arduino

Robot Sumo



Sommaire

Sommaire	2
Projet Robot Sumo	3
Objectif.....	3
Matériel	3
Introduction.....	4
Modélisation 3D	5
Chassis.....	6
Roues.....	6
Axe	7
Partie frontale.....	8
Cablage	9
Capteur infrarouge	10
Capteur ultrason.....	11
Moteurs.....	12
Programmation	13
Capteurs	14
Moteurs.....	16
Robot.....	18
Fichier main du robot	20
Fonction setup	20
Fonction Main.....	20
Documentation	21
Installation	21
Liste des PINS.....	21
Capteurs.....	21
Moteurs	21
Robot.....	21

Projet Robot Sumo

Objectif

L'objectif du projet est de concevoir un robot capable de se combattre contre un autre robot dans une arène de 60 cm de diamètre au sol noir et délimitée par une ligne blanche. L'objectif du robot est de rester dans l'arène et d'en éjecter son adversaire

Matériel

Pour réaliser ce projet, nous avons eu à notre disposition un capteur infrarouge permettant de détecter une ligne blanche ainsi qu'un capteur ultrason permettant lui de détecter la présence d'un autre robot. Le robot est également équipé de deux moteurs lui permettant de charger le robot détecté par le capteur ultrason. Et pour finir nous avons eu à disposition une imprimante 3D afin de modéliser les différentes parties du robot.

3



Introduction

Nous avons réalisé ce projet en plusieurs étapes. Etant donné que l'on était 2 sur le projet (projet initialement prévu pour 4), nous avons tenté de nous découper le travail efficacement tout en pouvant continuer à s'aider mutuellement quand l'un était bloqué sur une tâche.


Nous avons également eu le parti pris de ne pas utiliser de bibliothèques pour coder le robot (hormis les bibliothèques natives) bien que ces dernières réduisent le temps de développement. Nous n'avons également décidé de ne pas utiliser de modèle 3D déjà fait et de modéliser nous-même le robot.

C'est un parti pris qui nous aura pris plus de temps mais nous considérons que l'intérêt pédagogique était plus intéressant comme cela.

L'objectif était de comprendre les détails de ce que l'on faisait, et ceux, tant en programmation qu'en modélisation.

Cela nous a permis de ne pas être perdu sur le projet et d'avancer étape par étape.

Nous avons également développé une petite Bibliothèque Arduino afin de mieux segmenter le code pour travailler plus facilement dessus.



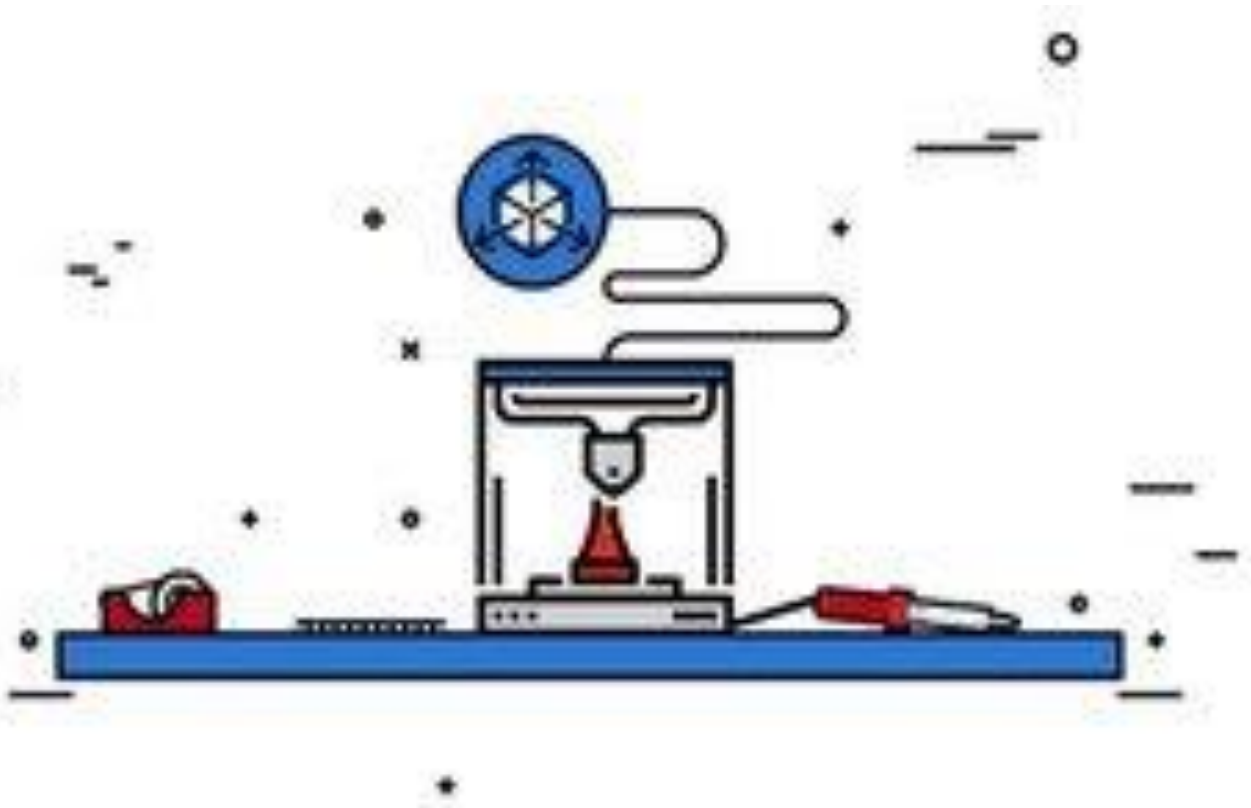
```
this example of
Single::ToString( ),
Single::ToString( String* ),
Single::ToString( IFormatProvider* ), and
Single::ToString( String*, IFormatProvider* )
generates the following output when run in
a Single number is formatted with various
strings and IFormatProvider.

IFormatProvider is not used; the default c
No format string: 11876.54
'N5' format string: 11,876.540
'E' format string: 1.187654E+
'ES' format string: 1.187654E+

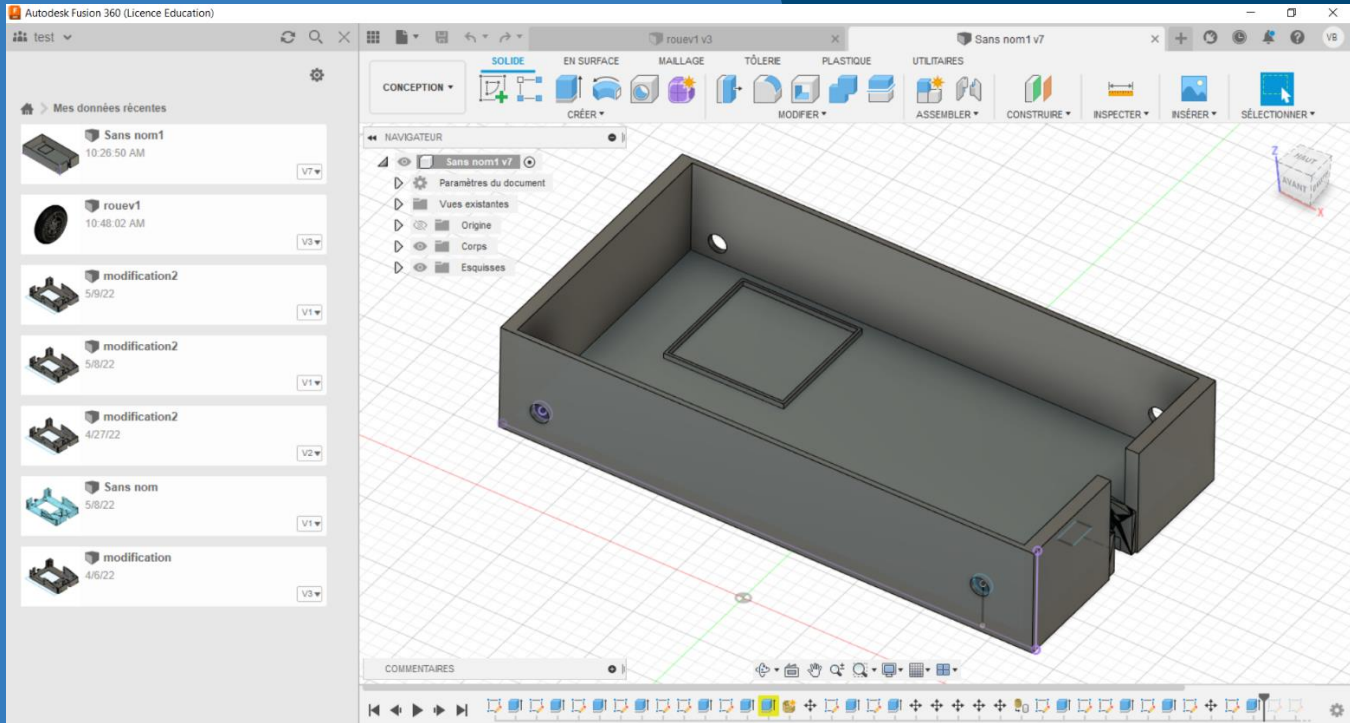
A CultureInfo object for Inl-NL is used f
No format string: 11876.54
'N5' format string: 11,876.540
'E' format string: 1.187654E+

A NumberFormatInfo object with digit group
digit separator = ',' is used for the IFor
'N' format string: 1,1876,54
'E' format string: 1.187654E+
Press any key to continue . . .
```

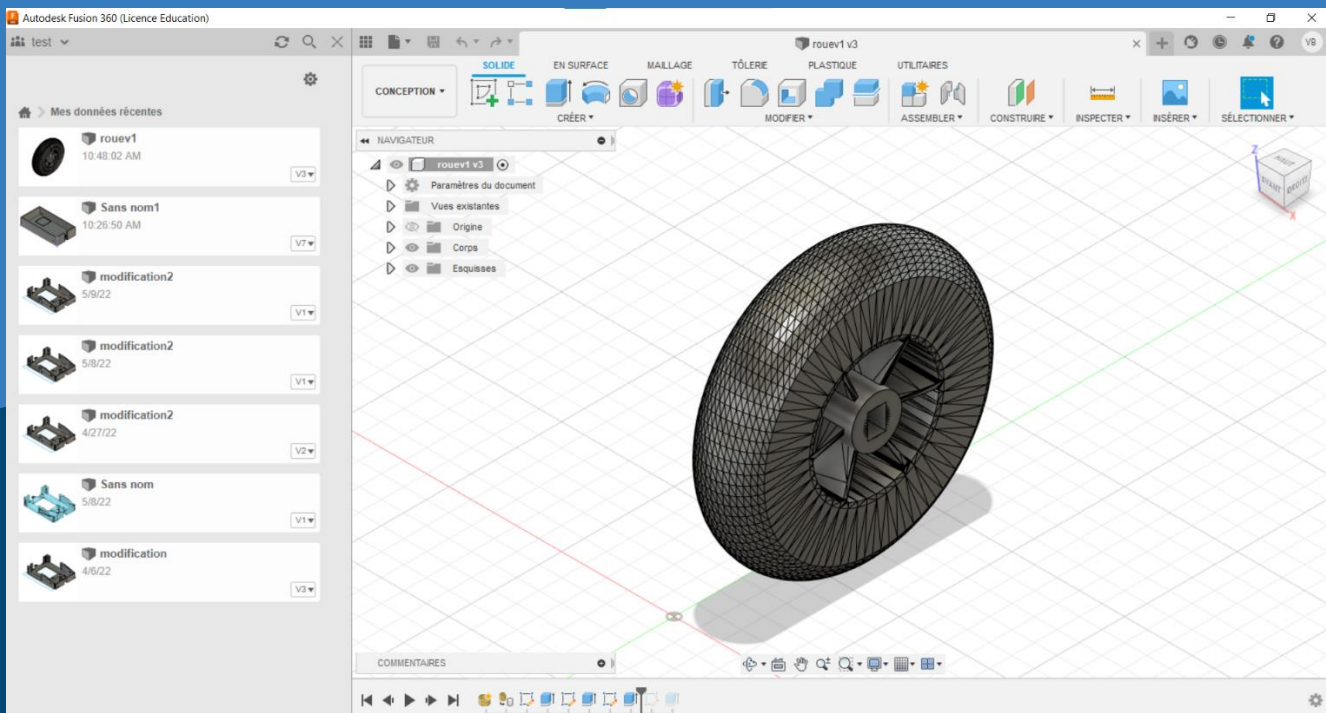
Modélisation 3D

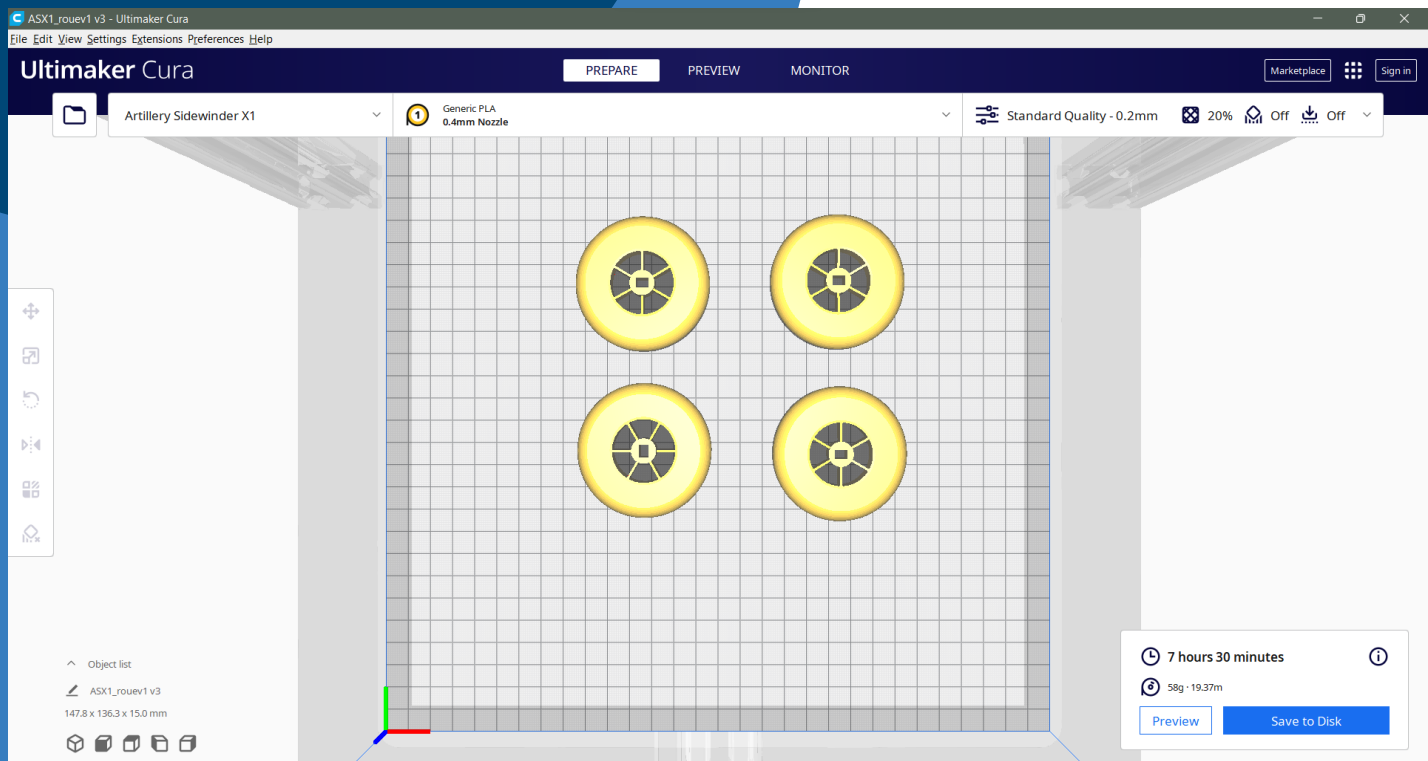


Chassis

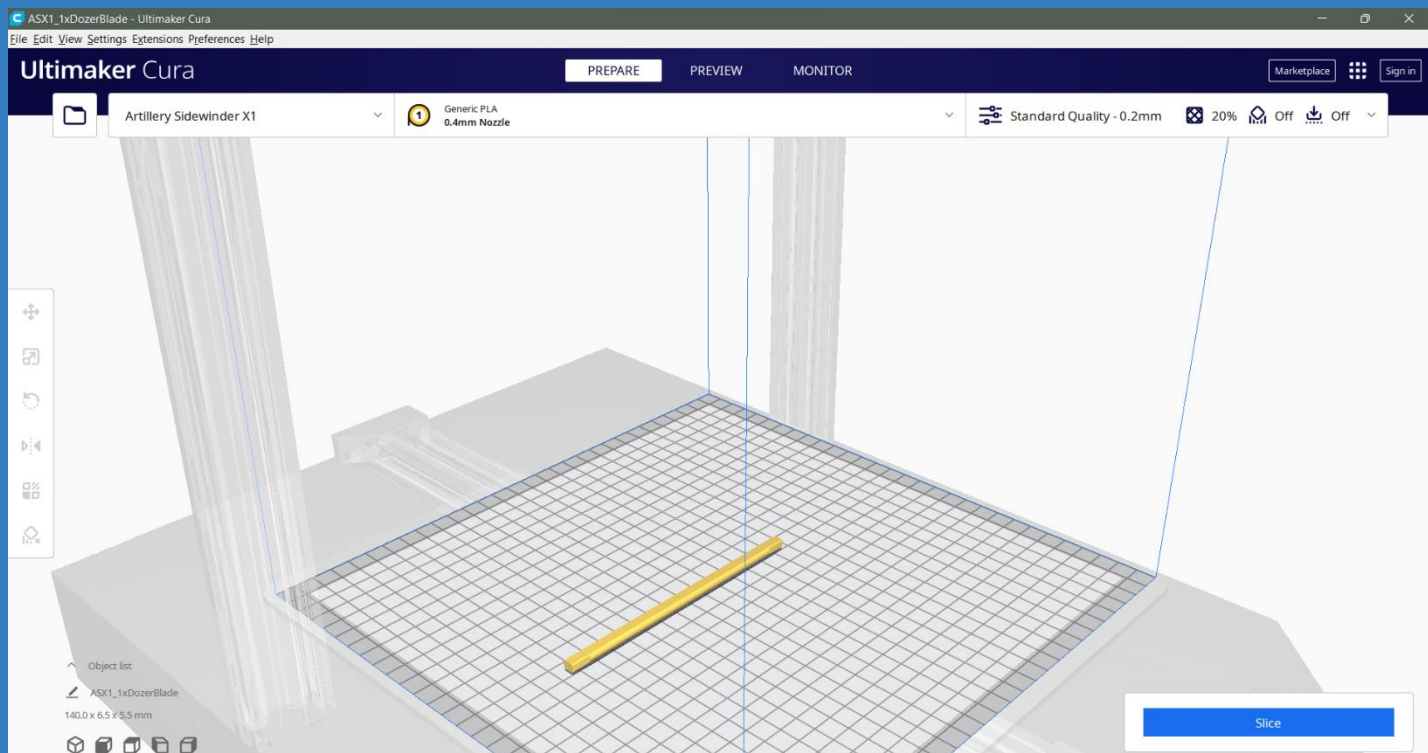


Roues

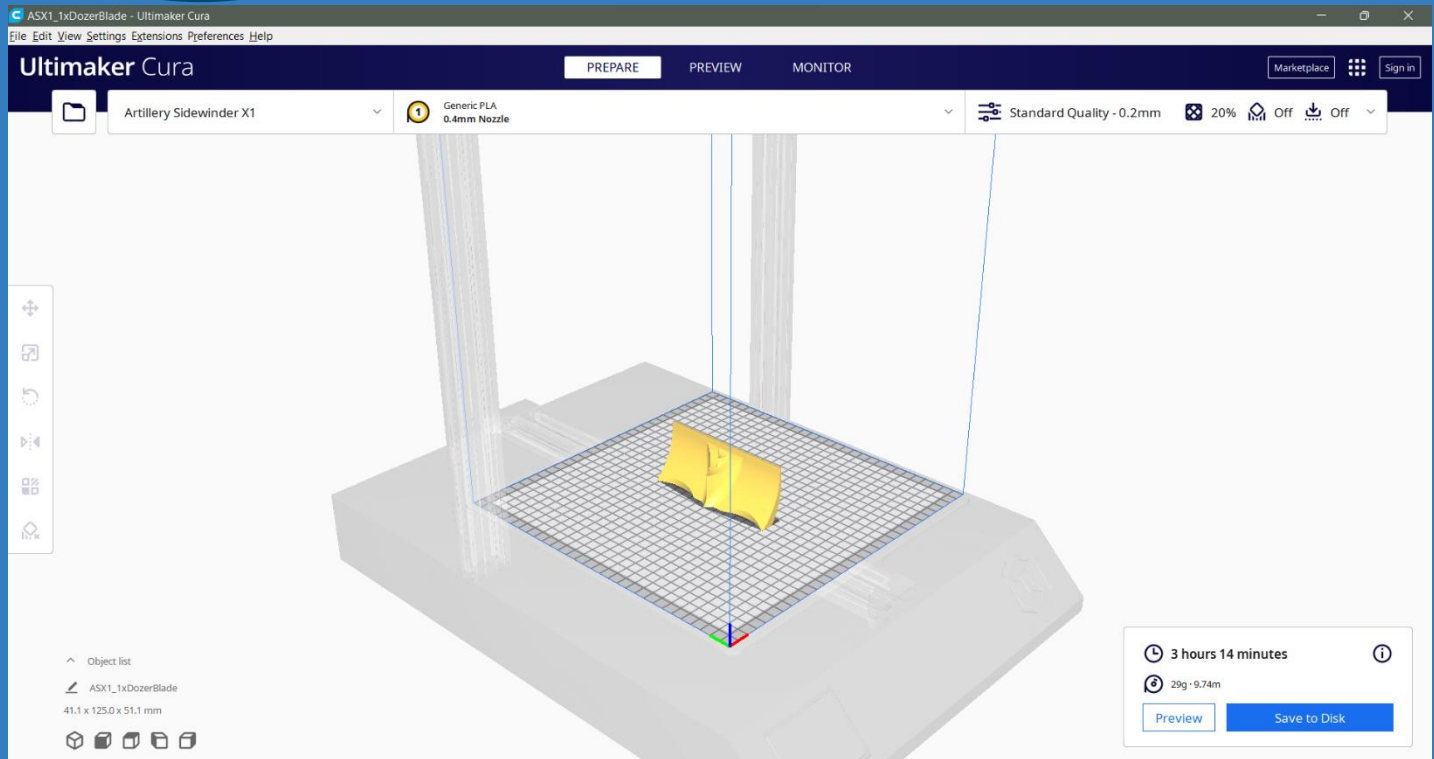




Axe



Partie frontale





Cablâge

Capteur infrarouge

Schéma Arduino

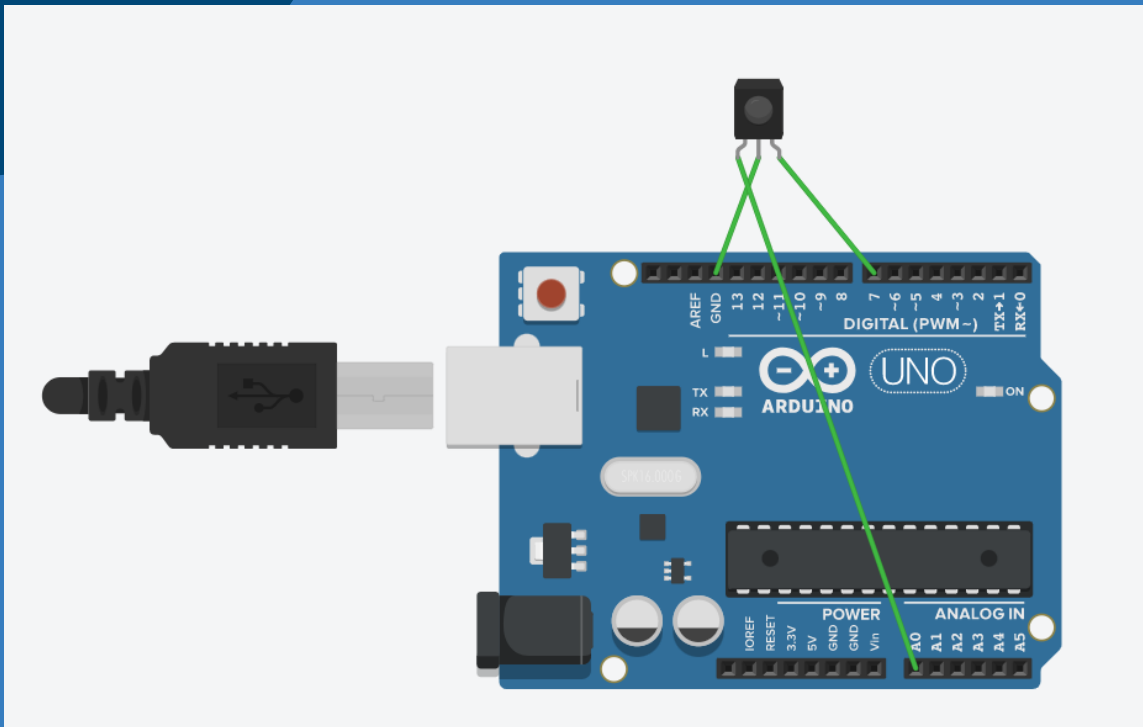
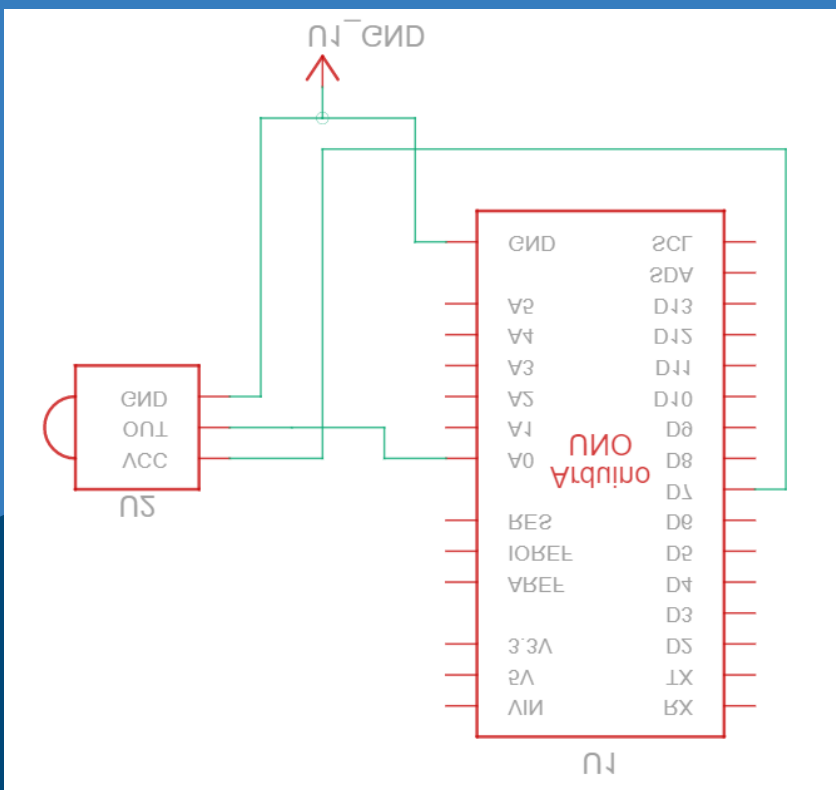


Schéma Technique



Capteur ultrason

Schéma Arduino

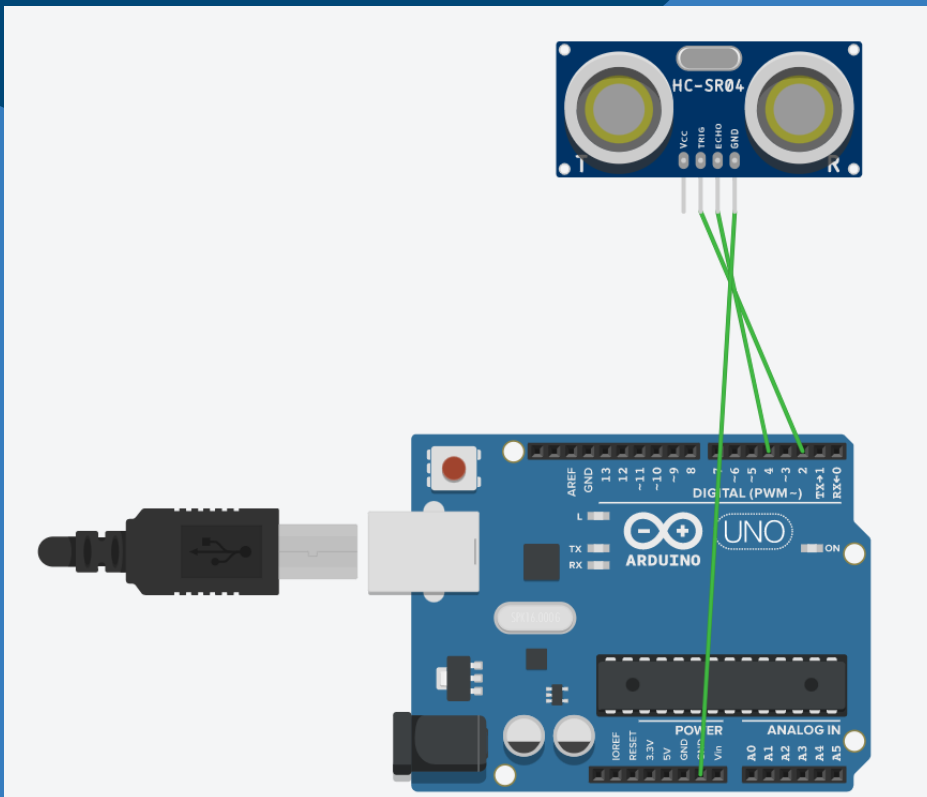
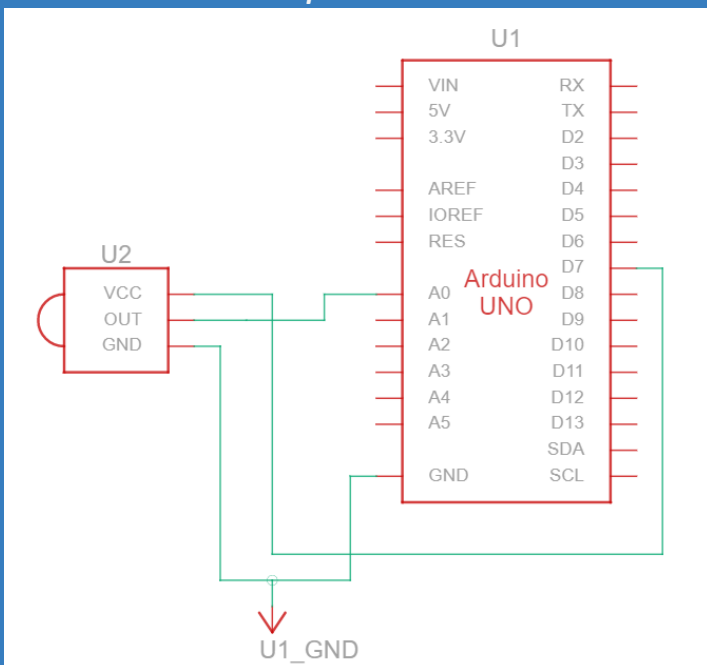


Schéma Technique



Moteurs

Schémas à intégrer

Programmation



Capteurs

Fichier ".h"

```
1  /*
2    Capteurs.cpp - Librairie pour projet Robot Sumo.
3  */
4
5  #ifndef Captors_h
6  #define Captors_h
7  #include "Arduino.h"
8
9  class Captors {
10     private :
11         // PINS CAPTEUR ULTRA SON
12         const byte PIN_TRIG = 2;
13         const byte PIN_ECHO = 4;
14         // PINS CAPTEUR INFRA ROUGE
15         const byte PIN_IR_DIGITAL = 7;
16         const int PIN_IR_ANALOG = A0;
17         // SETUP
18         void setUpCapteurUltraSon();
19         void setUpCapteurInfraRouge();
20
21     public :
22         Captors();
23         // Fonctions
24         long getEnemyDistance();
25         int getIRValue();
26 };
27 #endif
```


Fichier ".cpp"

```
#include "Captors.h"
#include "Arduino.h"

// Constructeur
Captors::Captors() {
    void setUpCapteurUltraSon();
    void setUpCapteurInfraRouge();
}

// Setup les capteurs
void Captors::setUpCapteurUltraSon() {
    pinMode(PIN_TRIG, OUTPUT);
    pinMode(PIN_ECHO, INPUT);
    digitalWrite(PIN_TRIG, LOW);
}

void Captors::setUpCapteurInfraRouge() {
    pinMode(PIN_IR_DIGITAL, INPUT);
    pinMode(PIN_IR_ANALOG, INPUT);
}

// La fonction retourne la distance d'un obstacle en cm
long int Captors::getEnemyDistance() {
    digitalWrite(PIN_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);
    long echoValue = pulseIn(PIN_ECHO, HIGH);
    return echoValue / 58;
}

// La fonction retourne la valeur de l'intensité lumineuse
int Captors::getIRValue() {
    int analogValue = analogRead(PIN_IR_ANALOG);
    int digitalValue = digitalRead(PIN_IR_DIGITAL);
    return digitalValue;
}
```

Moteurs

Fichier ".h"

```
1  /*
2    Capteurs.cpp - Librairie pour projet Robot Sumo.
3  */
4
5  #ifndef Motors_h
6  #define Motors_h
7  #include "Arduino.h"
8
9  class Motors {
10     private:
11         void setUpMotors();
12         const byte PIN_LEFT_MOTOR_FORWARD = 6;
13         const byte PIN_LEFT_MOTOR_BACKWARD = 3;
14         const byte PIN_RIGHT_MOTOR_FORWARD = 9;
15         const byte PIN_RIGHT_MOTOR_BACKWARD = 10;
16         const byte PIN_RIGHT_MOTOR_SPEED = 5;
17         const byte PIN_LEFT_MOTOR_SPEED = 11;
18
19     public:
20         Motors();
21         // Fonctions
22         void rightForward(byte speed);
23         void leftForward(byte speed);
24         void rightBackward(byte speed);
25         void leftBackward(byte speed);
26         void stopRight();
27         void stopLeft();
28     };
29 #endif
```

Fichier ".cpp"

```
1  #include "Motors.h"
2  #include "Arduino.h"
3
4  Motors::Motors() {
5      void setUpMotors();
6  }
7
8  // Setup les moteurs
9  void Motors::setUpMotors() {
10     pinMode(PIN_LEFT_MOTOR_FORWARD, OUTPUT);
11     pinMode(PIN_LEFT_MOTOR_BACKWARD, OUTPUT);
12     pinMode(PIN_RIGHT_MOTOR_FORWARD, OUTPUT);
13     pinMode(PIN_RIGHT_MOTOR_BACKWARD, OUTPUT);
14     pinMode(PIN_RIGHT_MOTOR_SPEED, OUTPUT);
15     pinMode(PIN_LEFT_MOTOR_SPEED, OUTPUT);
16 }
17
18 void Motors::rightForward(byte speed) {
19     digitalWrite(PIN_RIGHT_MOTOR_BACKWARD, LOW);
20     digitalWrite(PIN_RIGHT_MOTOR_FORWARD, HIGH);
21     analogWrite(PIN_RIGHT_MOTOR_SPEED, speed);
22 }
23
24 void Motors::leftForward(byte speed) {
25     digitalWrite(PIN_LEFT_MOTOR_BACKWARD, HIGH);
26     analogWrite(PIN_LEFT_MOTOR_SPEED, speed);
27     digitalWrite(PIN_LEFT_MOTOR_FORWARD, LOW);
28 }
29
30 void Motors::rightBackward(byte speed) {
31     digitalWrite(PIN_LEFT_MOTOR_FORWARD, LOW);
32     analogWrite(PIN_LEFT_MOTOR_SPEED, speed);
33     digitalWrite(PIN_LEFT_MOTOR_BACKWARD, HIGH);
34 }
35
36 void Motors::leftBackward(byte speed) {
37     digitalWrite(PIN_LEFT_MOTOR_BACKWARD, LOW);
38     analogWrite(PIN_LEFT_MOTOR_SPEED, speed);
39     digitalWrite(PIN_LEFT_MOTOR_BACKWARD, HIGH);
40 }
41
42 void Motors::stopRight() {
43     digitalWrite(PIN_RIGHT_MOTOR_BACKWARD, LOW);
44     analogWrite(PIN_RIGHT_MOTOR_SPEED, 0);
45 }
46
47 void Motors::stopLeft() {
48     digitalWrite(PIN_LEFT_MOTOR_BACKWARD, LOW);
49     analogWrite(PIN_LEFT_MOTOR_SPEED, 0);
50 }
```

Robot

Fichier ".h"

```
1  /*
2      Robot.cpp - Librairie pour projet Robot Sumo.
3  */
4
5  #ifndef ROBOT_H
6  #define ROBOT_H
7
8  #include "Arduino.h"
9  #include "Captors.h"
10 #include "Motors.h"
11
12 class Robot {
13     private:
14         Captors* captors;
15
16     public:
17         // Constructeur et destructeur
18         Robot();
19         ~Robot();
20         Motors* motors;
21         // Actions de mouvements des moteurs
22         void moveRight(byte speed);
23         void moveLeft(byte speed);
24         void moveForward(byte speed);
25         void moveBackward(byte speed);
26         void stopMove();
27         // Calculs infos capteurs
28         bool checkBorder();
29         bool checkEnemy();
30         // Actions avancées
31         void safeMoving(int timeInMs);
32         void dodge(double enemyDistance);
33 };
34 #endif
```

Fichier ".cpp"

```
1  #include "Arduino.h"
2  #include "Robot.h"
3  #include "Captors.h"
4  #include "Motors.h"
5
6  // Constructeur
7  Robot::Robot() {
8      captors = new Captors();
9      motors = new Motors();
10 }
11
12 // Destructeur
13 Robot::~Robot() {
14     delete captors;
15     delete motors;
16 }
17
18 // Tourne à droite
19 void Robot::moveRight(byte speed) {
20     // motors->rightBackward(speed);
21     motors->leftForward(speed);
22 }
23
24 // Tourne à gauche
25 void Robot::moveLeft(byte speed) {
26     motors->rightForward(speed);
27     motors->leftBackward(speed);
28 }
29
30 // Avance tout droit
31 void Robot::moveForward(byte speed) {
32     motors->rightForward(speed);
33     motors->leftForward(speed);
34 }
35
36 // Recule
37 void Robot::moveBackward(byte speed) {
38     motors->rightBackward(speed);
39     motors->leftBackward(speed);
40 }
41
42 // Arrête de bouger
43 void Robot::stopMove() {
44     motors->stopRight();
45     motors->stopLeft();
46 }
47
48 // Renvoie true si le capteur détecte une ligne blanche
49 bool Robot::checkBorder() {
50     const byte SEUIL_IR_VALUE = 255;
51     return captors->getIRValue() < SEUIL_IR_VALUE;
52 }
53
54 // Renvoie true si un ennemi se trouve à 50 cm ou moins du capteur
55 bool Robot::checkEnemy() {
56     const byte SEUIL_DISTANCE_ENNEMY = 35;
57     return captors->getEnemyDistance() < SEUIL_DISTANCE_ENNEMY;
58 }
59
60 // Se déplace et vérifie si la ligne est blanche en même temps
61 void Robot::safeMoving(int timeInMs) {
62     for (int i = 0; i < timeInMs; i++) {
63         if (checkBorder()) {
64             break;
65         }
66         else {
67             delay(1);
68         }
69     }
70 }
```

Fichier main du robot

C'est ici que l'on va utiliser le code des fonctions des bibliothèques que nous avons écrites.

Fonction setup

```
#include <Arduino.h>
#include <Robot.h>

Robot* robot = new Robot();

// Les vitesses
const byte MIN_SPEED = 0;
const byte AVERAGE_SPEED = 100;
const byte MAX_SPEED = 255;

// Etats du robot
enum State {
    SEARCHING = 0,
    ENEMY_SPOTTED = 1,
} state;

// Fonction setup
void setup() {
    Serial.begin(9600);
    state = SEARCHING;
}
```

Fonction Main

```
// Fonction Main
void loop() {
    // On verifie l'état du robot à chaque tour de boucle
    switch (state) {
        // Etat de recherche
        case SEARCHING:
            if (robot->checkEnemy()) {
                state = ENEMY_SPOTTED;
            }
            else {
                robot->moveRight(AVERAGE_SPEED);
            }

            // Etat d'attaque
            case ENEMY_SPOTTED:
                robot->moveForward(AVERAGE_SPEED);
                delay(700);

                // Si le robot voit une ligne blanche ou si il ne voit pas d'ennemi il retourne en état de recherche
                if (robot->checkBorder() || !robot->checkEnemy()) {
                    robot->stopMove();
                    state = SEARCHING;
                }
            }
    }
}
```


Documentation

Installation

Installez les librairies "Robot", "Captors" et "Motors" dans le dossier des librairies d'Arduino.
-> Par défaut : Documents/Arduino/librairie

Liste des PINS

PINS MOTEURS :

PIN_LEFT_MOTOR_FORWARD : 6
PIN_LEFT_MOTOR_BACKWARD : 3
PIN_RIGHT_MOTOR_FORWARD : 9
PIN_RIGHT_MOTOR_BACKWARD : 10
PIN_RIGHT_MOTOR_SPEED : 5
PIN_LEFT_MOTOR_SPEED : 11

PINS CAPTEUR ULTRA SON :

PIN_TRIG : 2
PIN_ECHO : 4

PINS CAPTEUR INFRA ROUGE :

PIN_IR_DIGITAL : 7
PIN_IR_ANALOG : A0

Capteurs

Liste des fonctions :

bool checkBorder() -> Renvoie True si le capteur infrarouge détecte du blanc
bool checkEnemy() -> Renvoie True si le capteur ultrason détecte une présence

Moteurs

Robot

Utiliser l'objet robot

Instanciez l'objet robot au debut de votre programme de cette façon :

```
Robot* robot = new Robot();
```

Accédez aux actions du robot en utilisant une flèche pour pointer vers les méthodes du robot :

exemple:

```
robot->moveForward();
```

Actions de mouvements du robot :

Speed est une variable de type byte, elle va de 0 à 255 et définit la vitesse des moteurs du robot.
donc 255 -> vitesse max et 0 -> robot à l'arrêt.

Liste des fonctions :

`void moveRight(speed)` -> Le robot tourne à droite à la Vitesse "speed".

`void moveLeft(speed)` -> Le robot tourne à gauche à la Vitesse "speed".

`void moveForward(speed)` -> Le robot avance à la Vitesse "speed".

`void moveBackward(speed)` -> Le robot recule à la Vitesse "speed".

`void stopMove()` -> Le robot s'arrête.