Vittascience

Vittascience FR

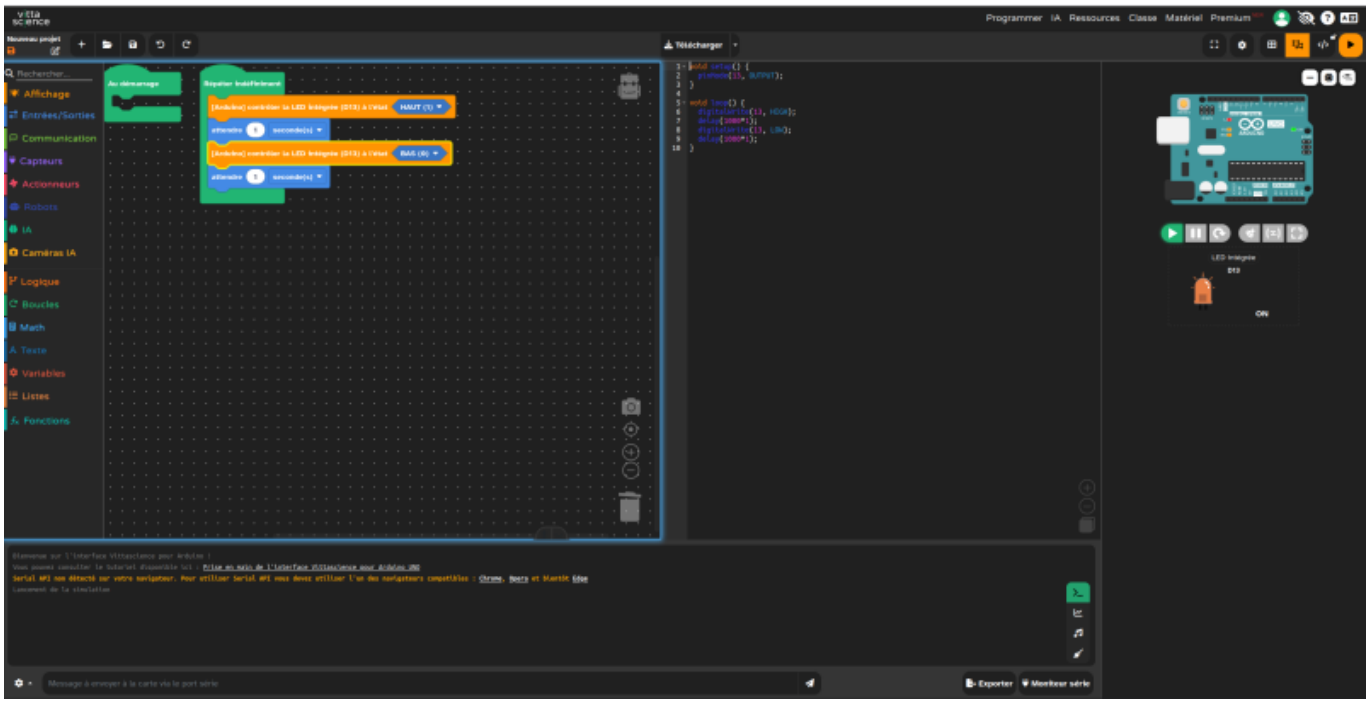
Vittascience : Une plateforme innovante pour enseigner la programmation

Vittascience est une plateforme pédagogique en ligne qui permet d'enseigner la programmation informatique et l'intelligence artificielle aux enfants de manière ludique et interactive. Elle propose une interface intuitive où les enfants peuvent coder en blocs, à la manière de Scratch, ou en code, pour ceux qui souhaitent progresser vers des langages plus complexes. L'interface en mode hybride, comprenant les modes blocs et code juxtaposés, permet de mieux comprendre le lien entre les deux modes de langage. Vittascience offre une transition en douceur entre ces deux modes, permettant aux enfants de développer leurs compétences à leur propre rythme.

cataloguevittascience2025.pdf

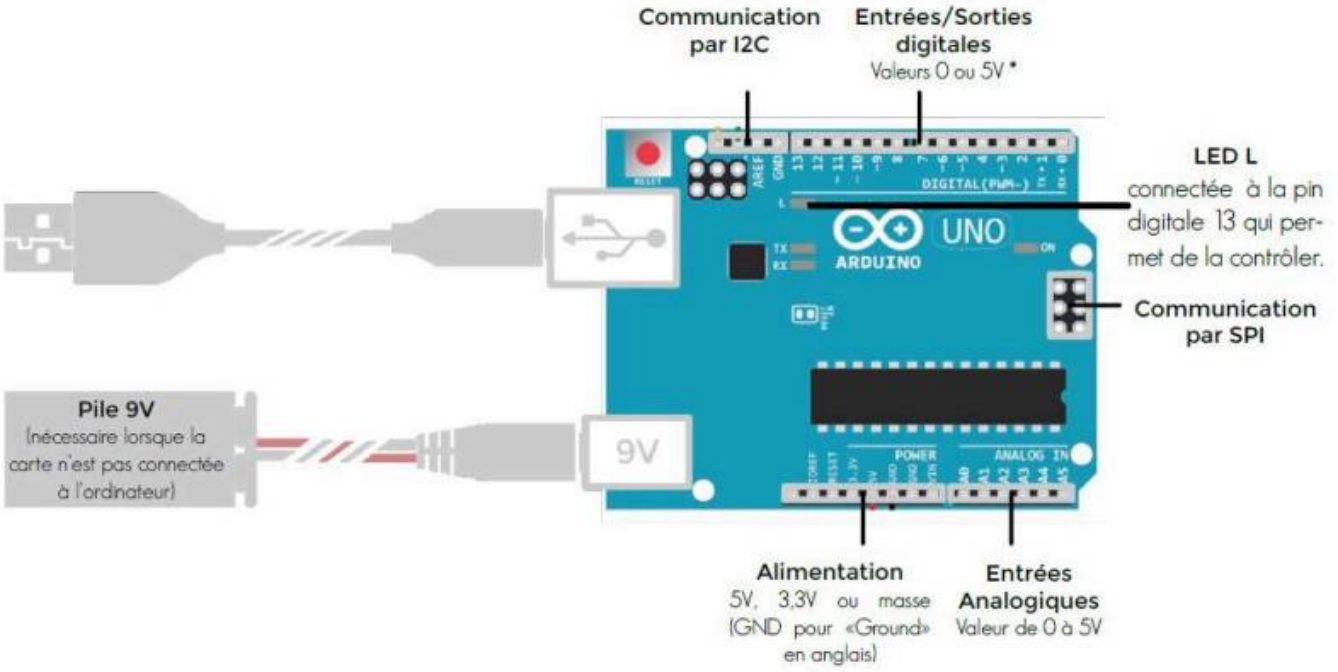
Programmer un arduino

Vittascience et Arduino



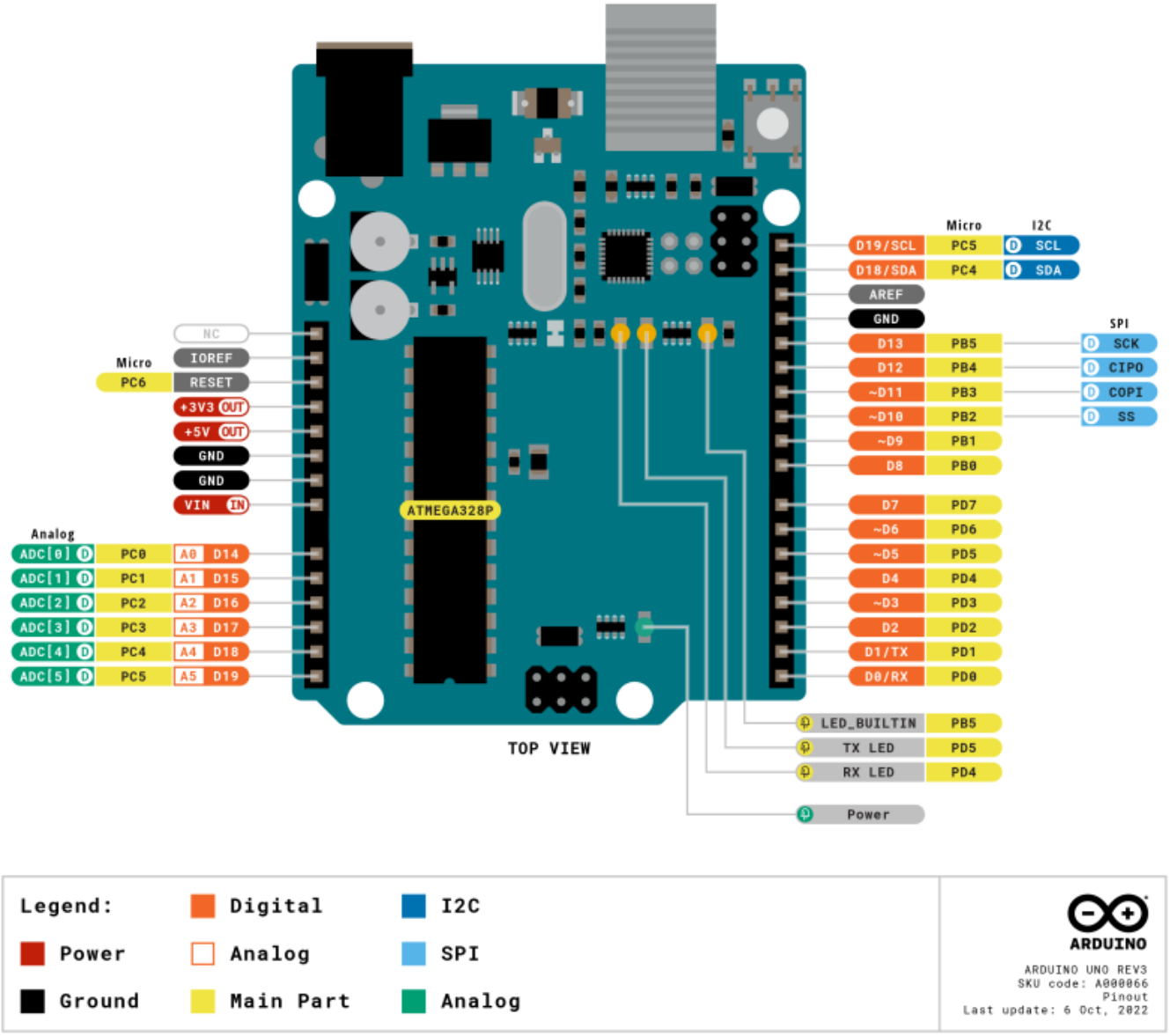
Decouverte Arduino

Manipulation de la carte Arduino Uno et de sa LED intégrée



Decouverte Arduino UNO

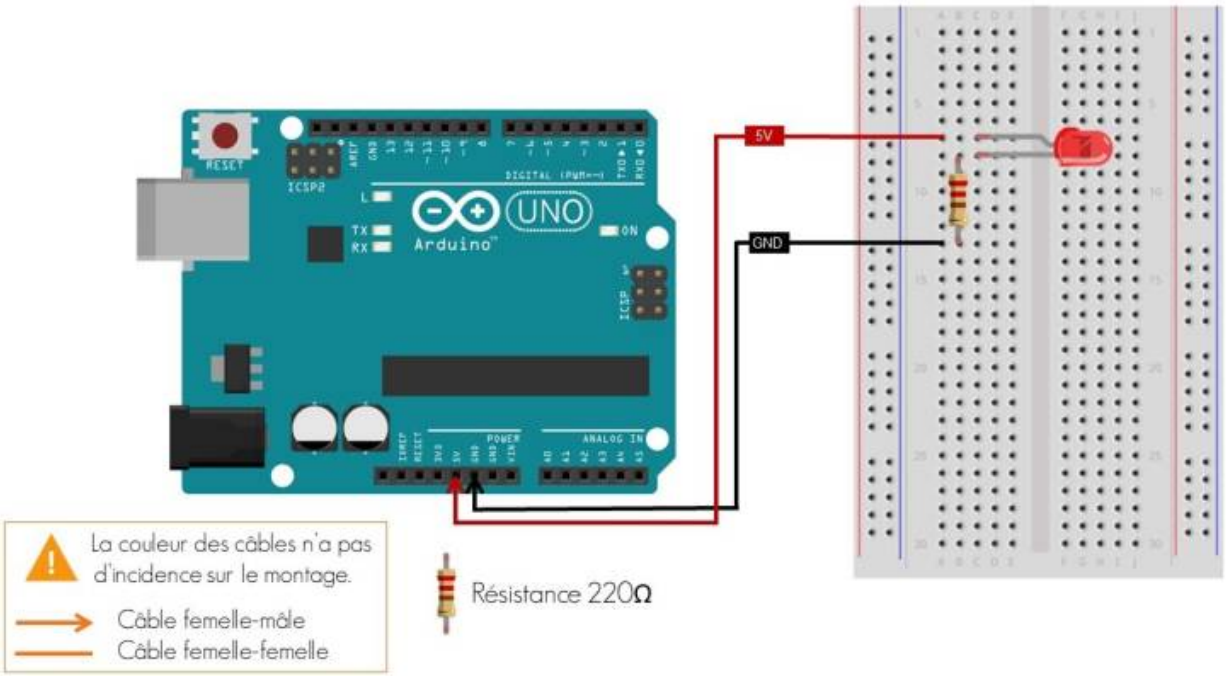
Prise en main de la carte Arduino UNO et de l'interface Vittascience



prise_en_main_de_la_carte_arduino_uno_et_de_l_interface_vittascience.pdf

Decouverte LED et Arduino

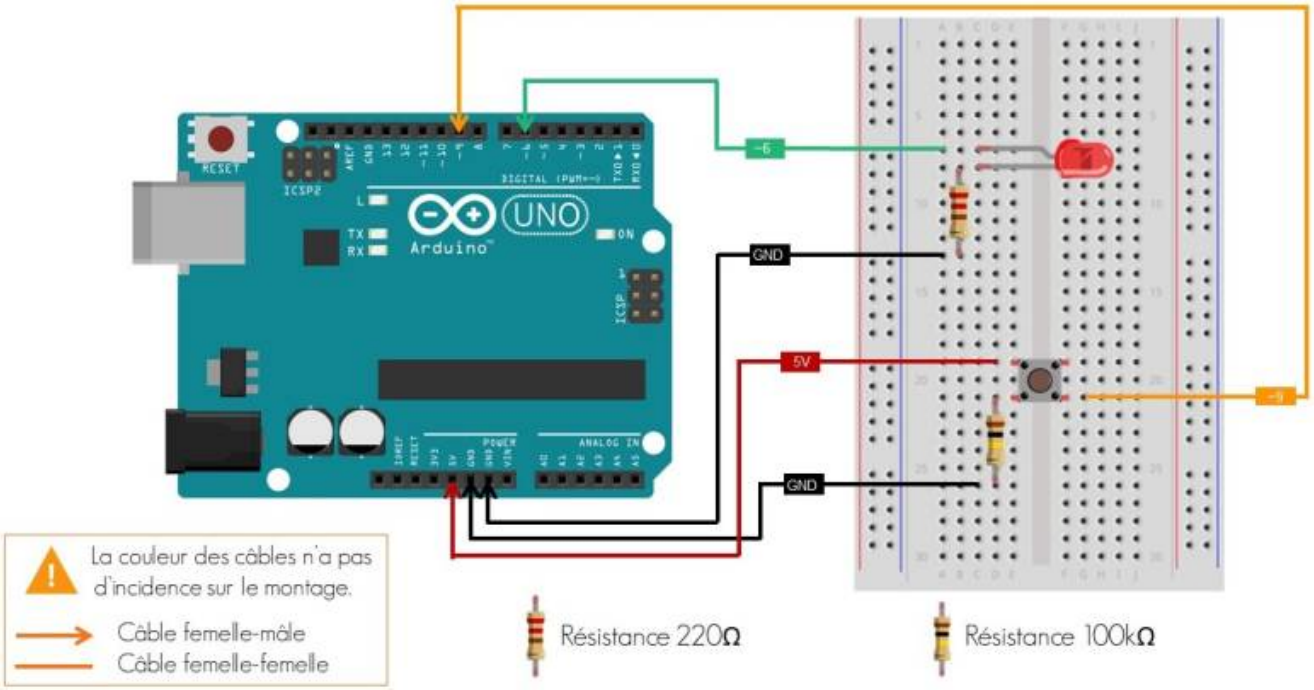
Allumer une LED avec Arduino



allumer_une_led_avec_arduino.pdf

Decouverte Bouton + LED + Arduino

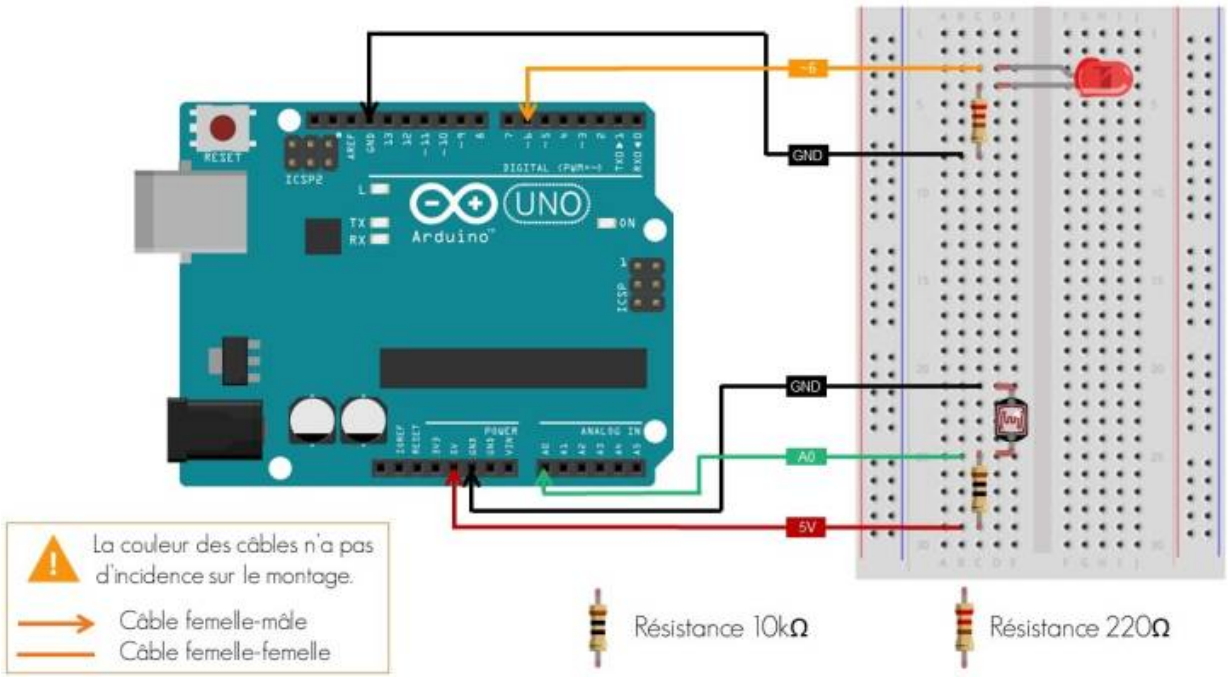
Ajouter un bouton sur Arduino



ajouter_un_bouton_sur_arduino.pdf

Decouverte PhotoResistance + Led + Arduino

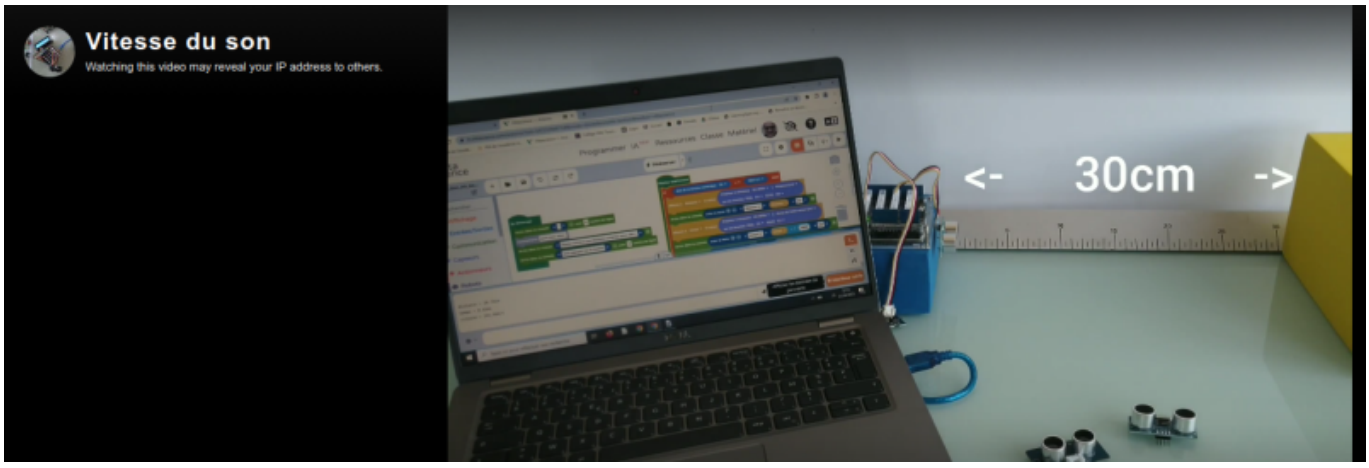
Mesurer la luminosité avec une photorésistance sur Arduino



mesurer_la_luminosite_avec_une_photoresistance_sur_arduino.pdf

Decouverte Vitesse du son avec un HC-SR04 + Arduino

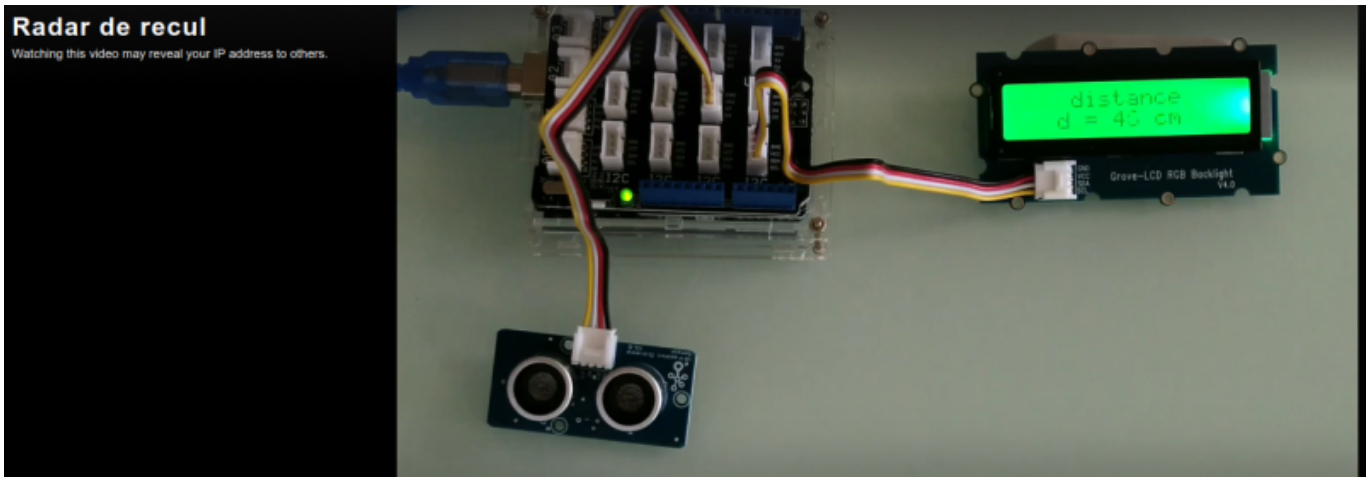
Vitesse du son : Capteur ultrason HC-SR04



vitesse_du_son.pdf

Decouverte Radar de recul avec HC-SR04 + Arduino

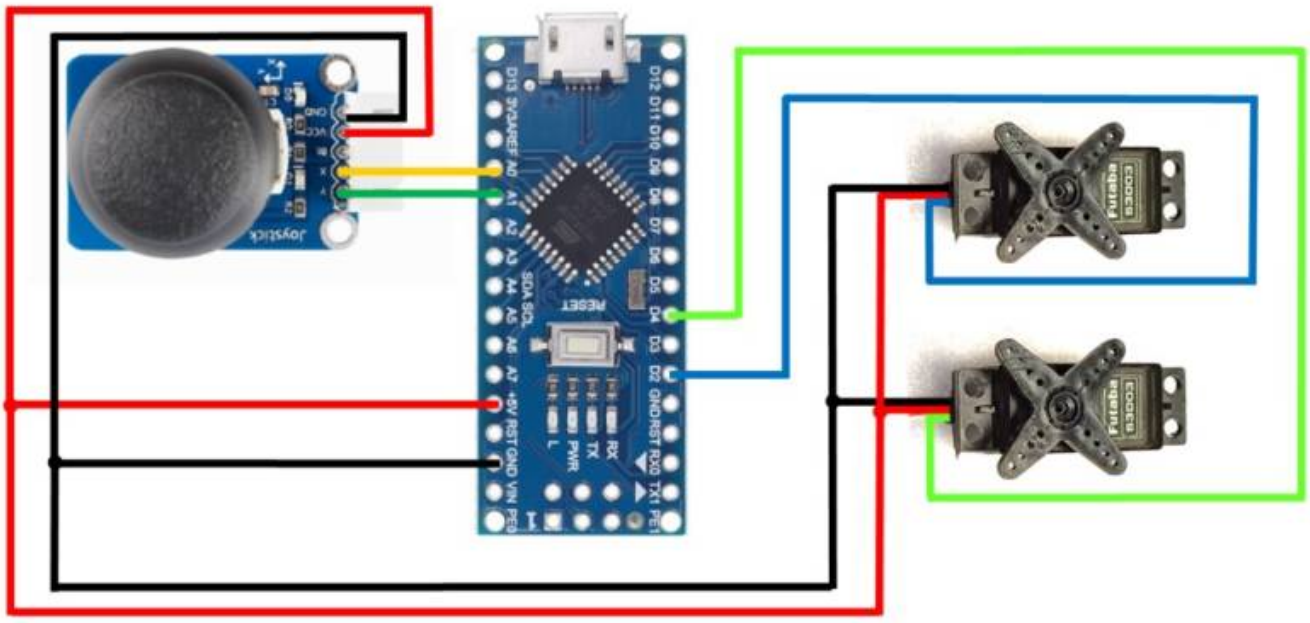
Radar de recul avec un écran LCD



radar_de_recul_avec_un_ecran_lcd_colore.pdf

Decouverte Chariot+ servo Moteur 360° + Arduino

Chariot téléguidé par un joystick



chariot_teleguide_par_un_joystick.pdf

Programmer un Robot Mbot

Televerser dans le robot mBot1 à partir de l'interface Vittascience

Pour Téléverser le programme robot sur mBot1 à partir de l'interface Vittascience (si “Televerser” ne fonctionne pas , passer par les étapes 1 et 2 ...) :

1- Vider le cache de Google Chrome

2-Passer en navigation privée sur Google chrome

3- cliquer sur “Téléverser” (connecter sur USB et allumer le mBot1 avant ...)

4- choisir le port COM du mBot1 et faire connexion (exemple: **“COM3”** sous Windows 11 ou **“USB Serial (ttyUSB0)“** sous Linux)

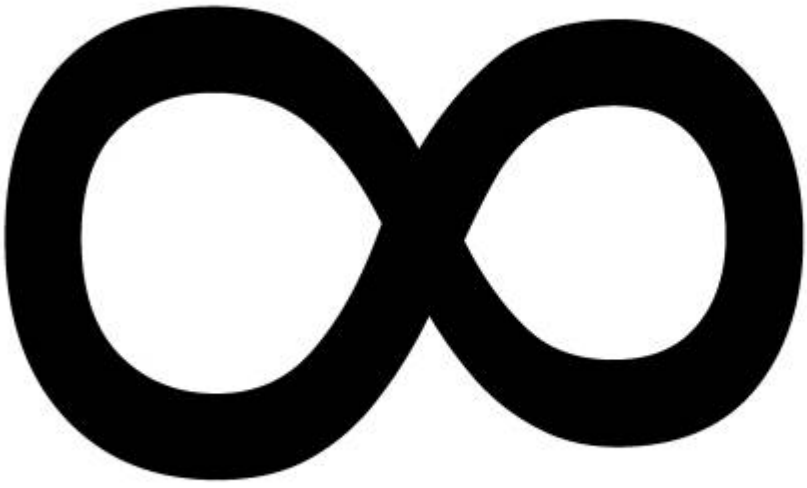
et c'est tout , dans la console série de vittascience tu dois voir :

```
Le programme utilise 8469 octets d'espace de stockage. (28%)
Compilation réussie.
Téléversement du programme ...
Carte détectée: Arduino Nano w/ ATmega328
Programme Arduino envoyé avec succès dans la carte !"
```

Exemples Vittascience Robot mBot1

Image fond ecran pour virtualisation mBot1 sur Vittascience

tracerobot004.jpg.zip

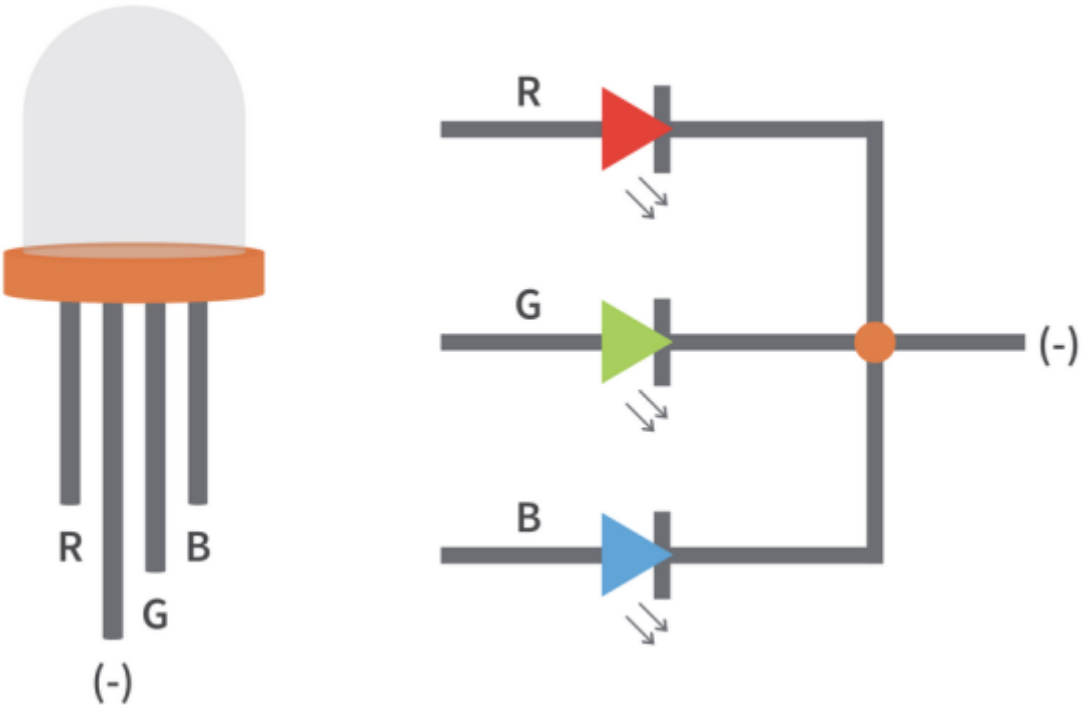


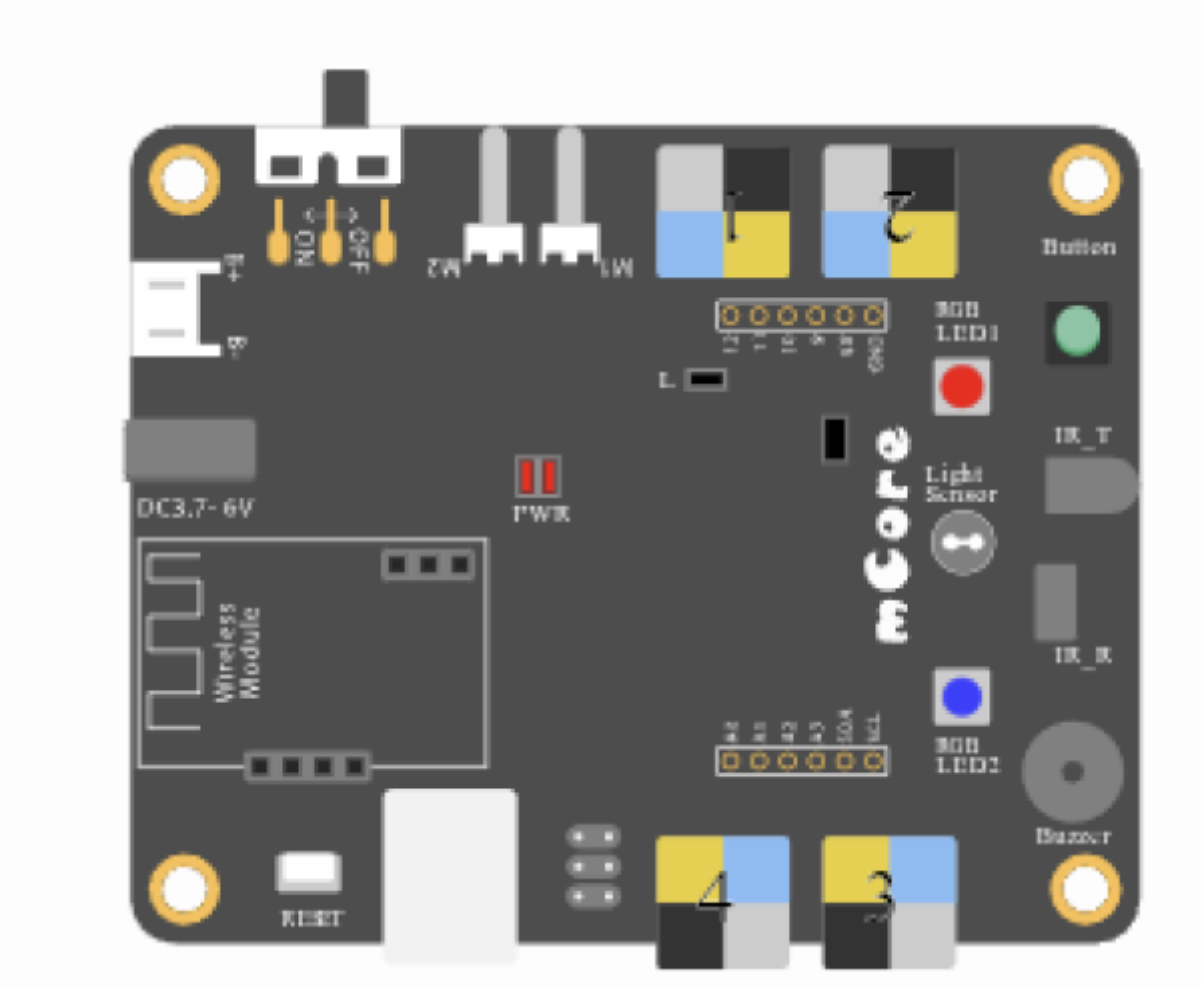
Programmes mBot1 Vittascience

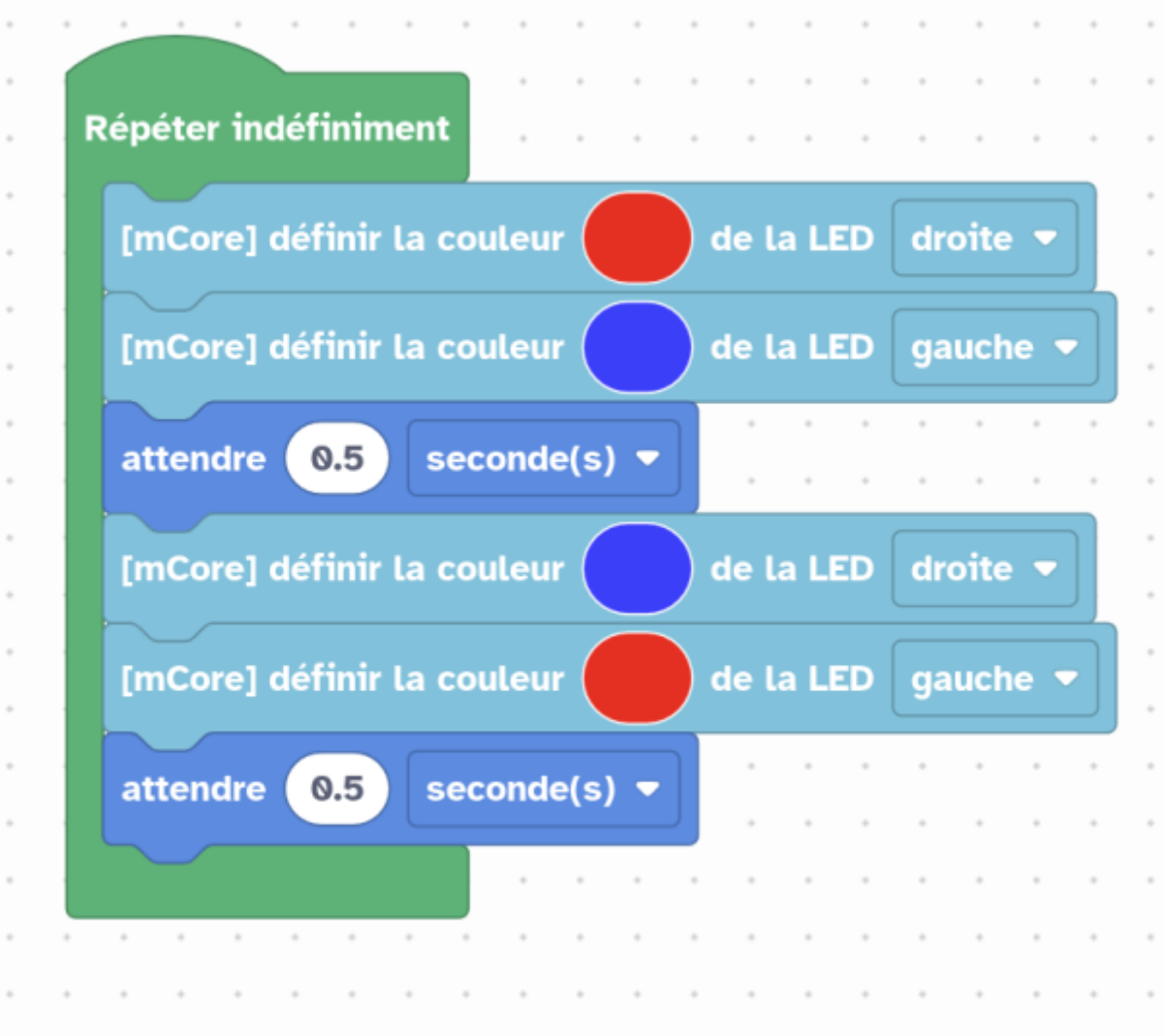
robot-mbot-utiliser-les-led-rgb

robot-mbot-utiliser-les-led-rgb

robot_mbot_-_utiliser_les_led_rgb.pdf




ON
OFF
M2
M1
Button
RGB LED1
IR_T
Light Sensor
mCore
PWR
DC3.7- 6V
Wireless Module
IR_R
RGB LED2
Buzzer
RESET
1
2
3
4



rgb.ino

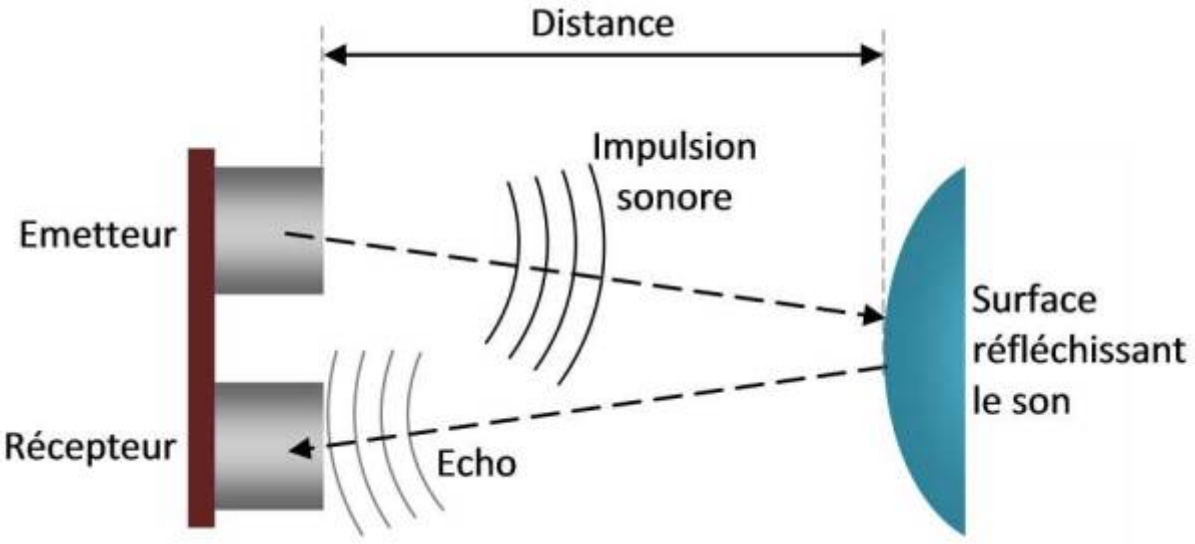
```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeRGBLed rgbled_board(7, 2);

void setup() {
}

void loop() {
  rgbled_board.setColor(1, 255, 0, 0);
  rgbled_board.show();
  rgbled_board.setColor(2, 51, 51, 255);
  rgbled_board.show();
  delay(1000*0.5);
  rgbled_board.setColor(1, 51, 51, 255);
  rgbled_board.show();
  rgbled_board.setColor(2, 255, 0, 0);
```

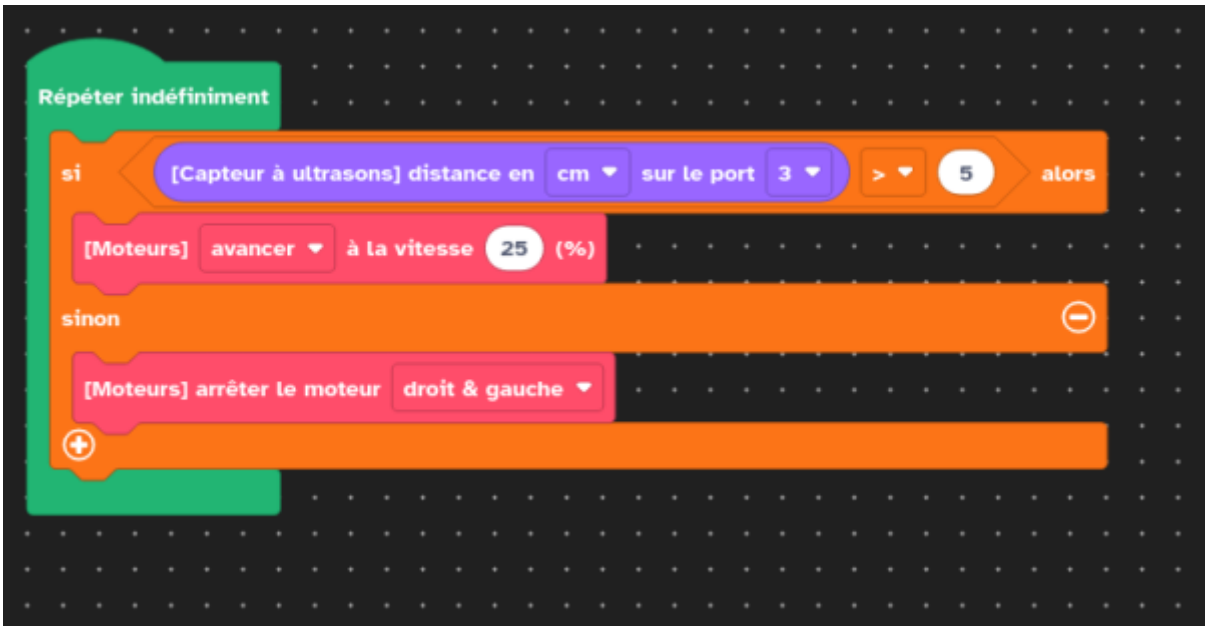
```
  rgbled_board.show();
  delay(1000*0.5);
}
```

Robot mBot - Utiliser le capteur de distance

robot-mbot-utiliser-le-capteur-de-distance

robot_mbot_-_utiliser_le_capteur_de_distance.pdf





robot_mbot_-_utiliser_le_capteur_de_distance_-_partie_3_20251027_123737.ino.tar (Enlever .tar)

distancearduino.ino

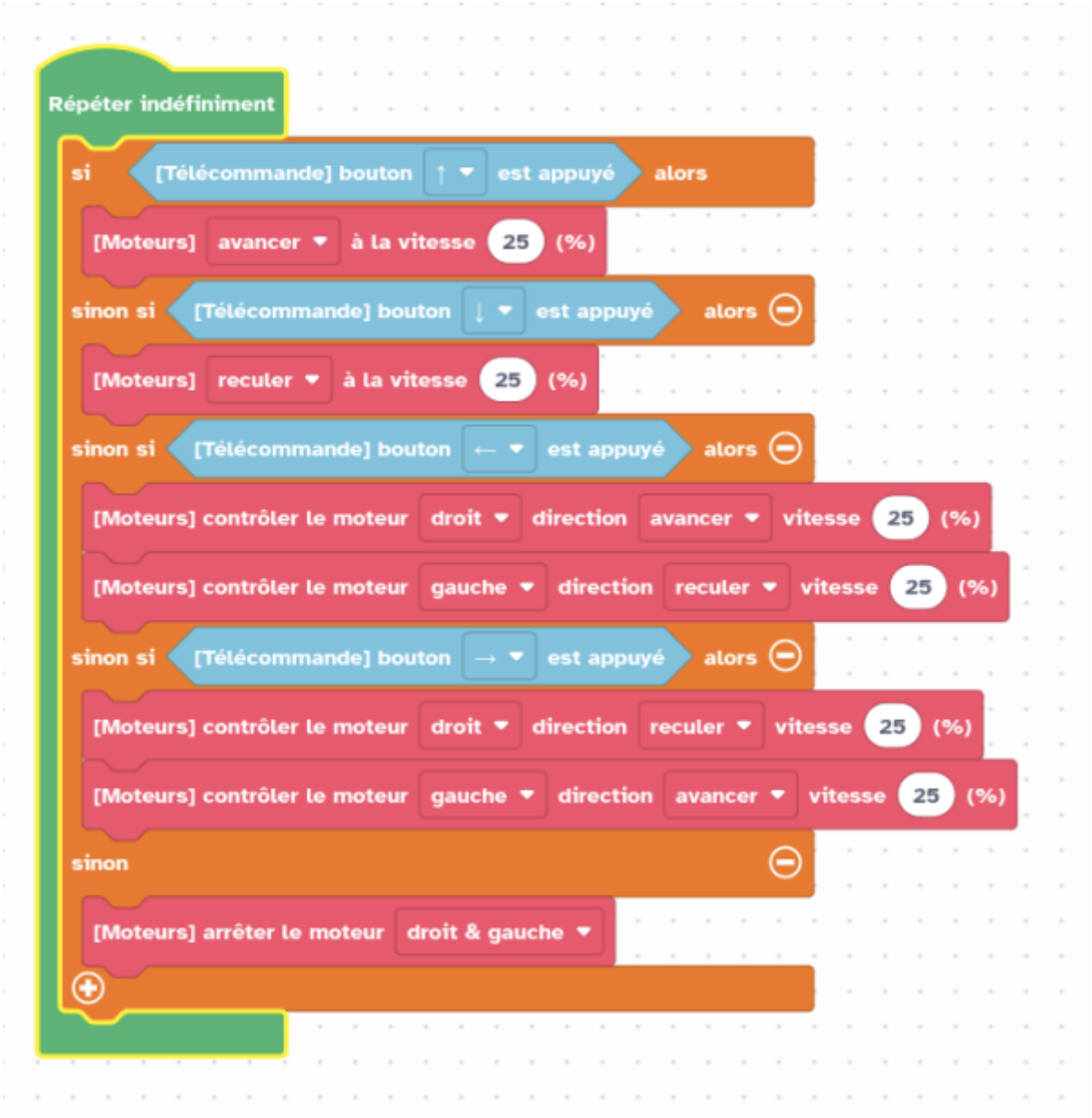
```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

// Ultrasonic on PORT_3
MeUltrasonicSensor ultrasonic_3(PORT_3);
MeDCMotor motor_L(9);
MeDCMotor motor_R(10);

void mBot_setMotorLeft(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_L.run((9) == M1 ? -(dir*speed) : (dir*speed));
}

void mBot_setMotorRight(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_R.run((10) == M1 ? -(dir*speed) : (dir*speed));
}


void setup() {
}

void loop() {
  if (ultrasonic_3.distanceCm() > 5) {
    mBot_setMotorRight(1, 25);
```

```
    mBot_setMotorLeft(1, 25);
  } else {
    mBot_setMotorRight(0, 0);
    mBot_setMotorLeft(0, 0);
  }
}
```

robot-mbot-controler-le-robot-avec-une-telecommande

robot-mbot-controler-le-robot-avec-une-telecommande

robot_mbot_-_controler_le_robot_avec_une_telecommande.pdf



robot_mbot_-_controler_le_robot_avec_une_telecommande_-_partie_3_20251027_144323.ino.tar (enlever .tar)

telecommandemBot.ino

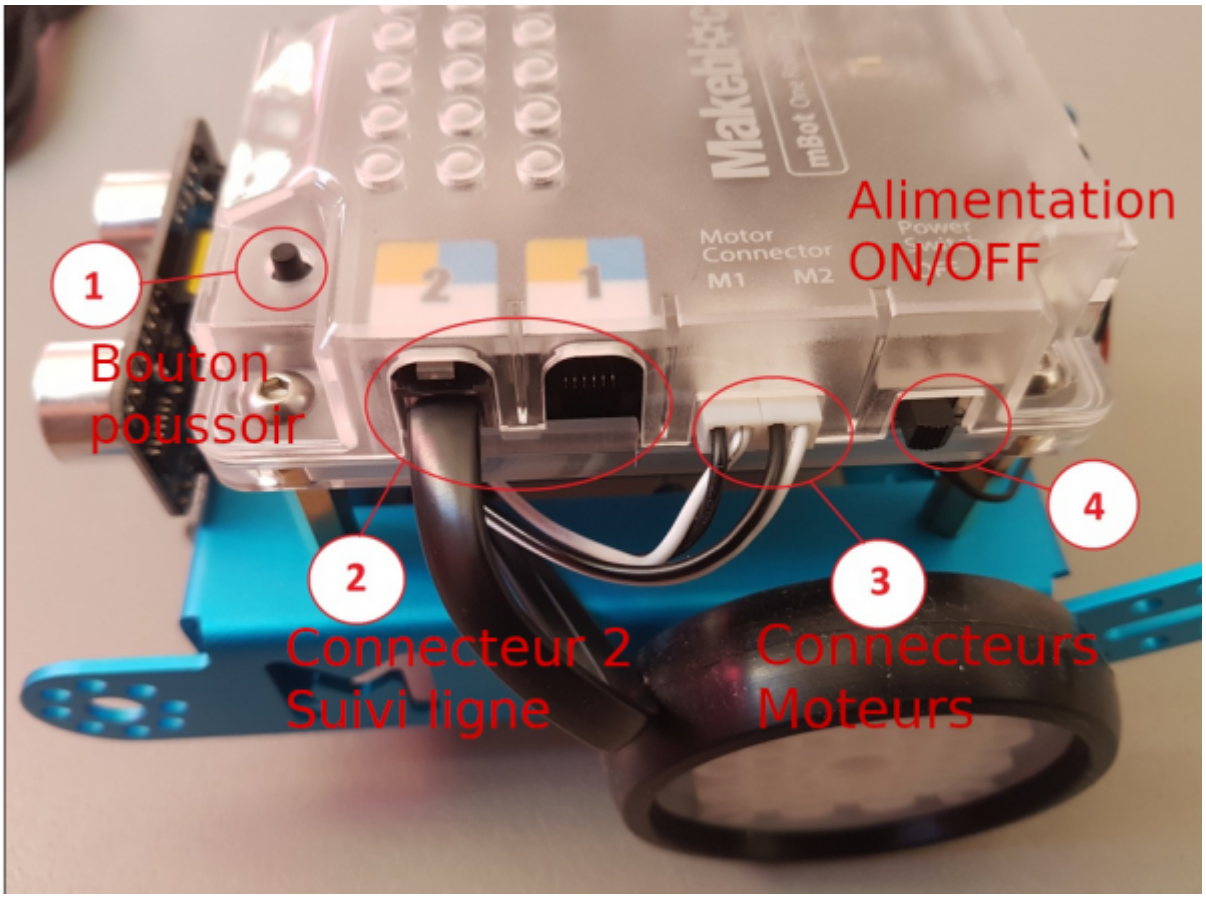
```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeIR ir;
MeDCMotor motor_L(9);
MeDCMotor motor_R(10);

void mBot_setMotorLeft(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
```

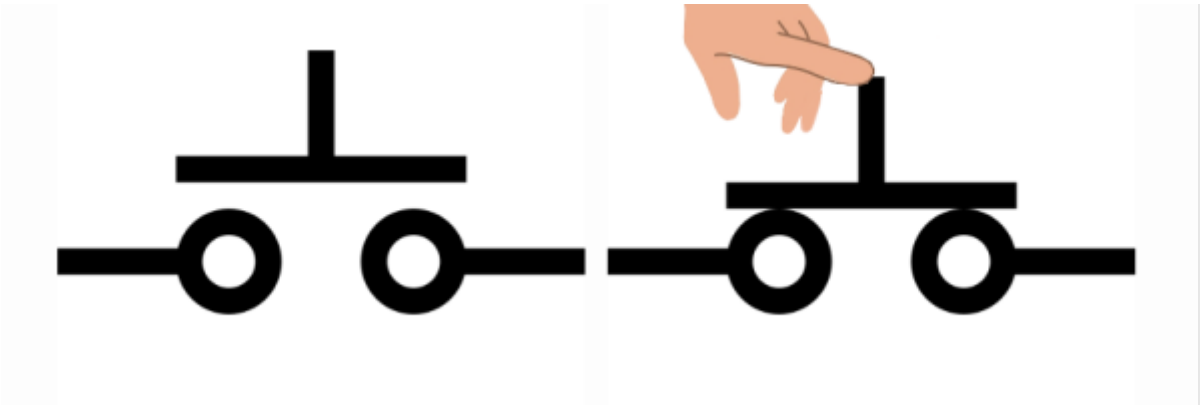
```
  motor_L.run((9) == M1 ? -(dir*speed) : (dir*speed));
}

void mBot_setMotorRight(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_R.run((10) == M1 ? -(dir*speed) : (dir*speed));
}


void setup() {
  ir.begin();
}

void loop() {
  if (ir.keyPressed(64)) {
    mBot_setMotorRight(1, 25);
    mBot_setMotorLeft(1, 25);
  } else if (ir.keyPressed(25)) {
    mBot_setMotorRight(-1, 25);
    mBot_setMotorLeft(-1, 25);
  } else if (ir.keyPressed(7)) {
    mBot_setMotorRight(1, 25);
    mBot_setMotorLeft(-1, 25);
  } else if (ir.keyPressed(9)) {
    mBot_setMotorRight(-1, 25);
    mBot_setMotorLeft(1, 25);
  } else {
    mBot_setMotorRight(0, 0);
    mBot_setMotorLeft(0, 0);
  }
}
```

robot-mbot-utiliser-le-bouton

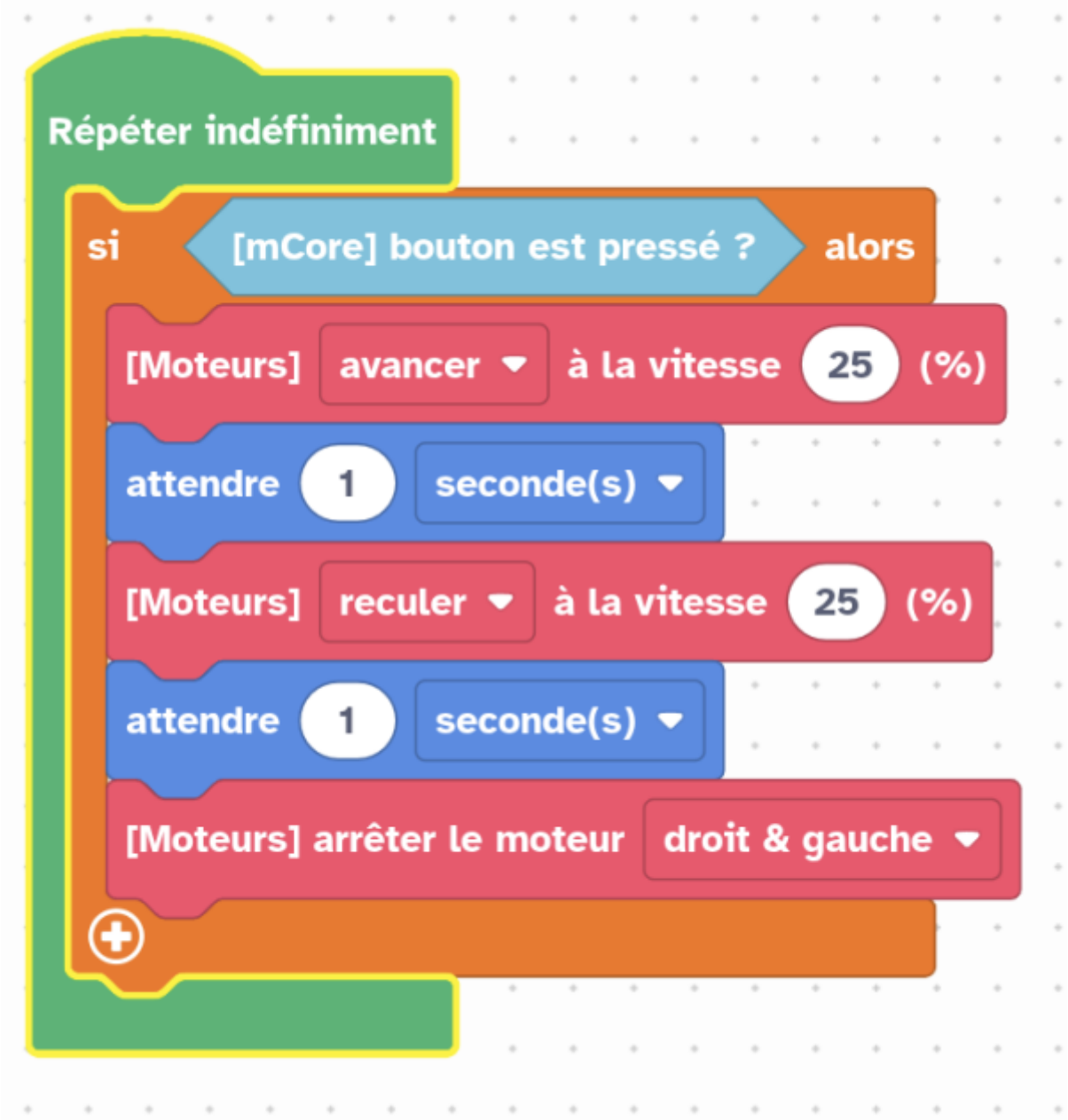
robot-mbot-utiliser-le-bouton

robot_mbot_-_utiliser_le_bouton.pdf





robot_mbot_-_utiliser_le_bouton_-_partie_3_20251027_15143.ino.tar
(enlever .tar)



boutonpoussoir.ino

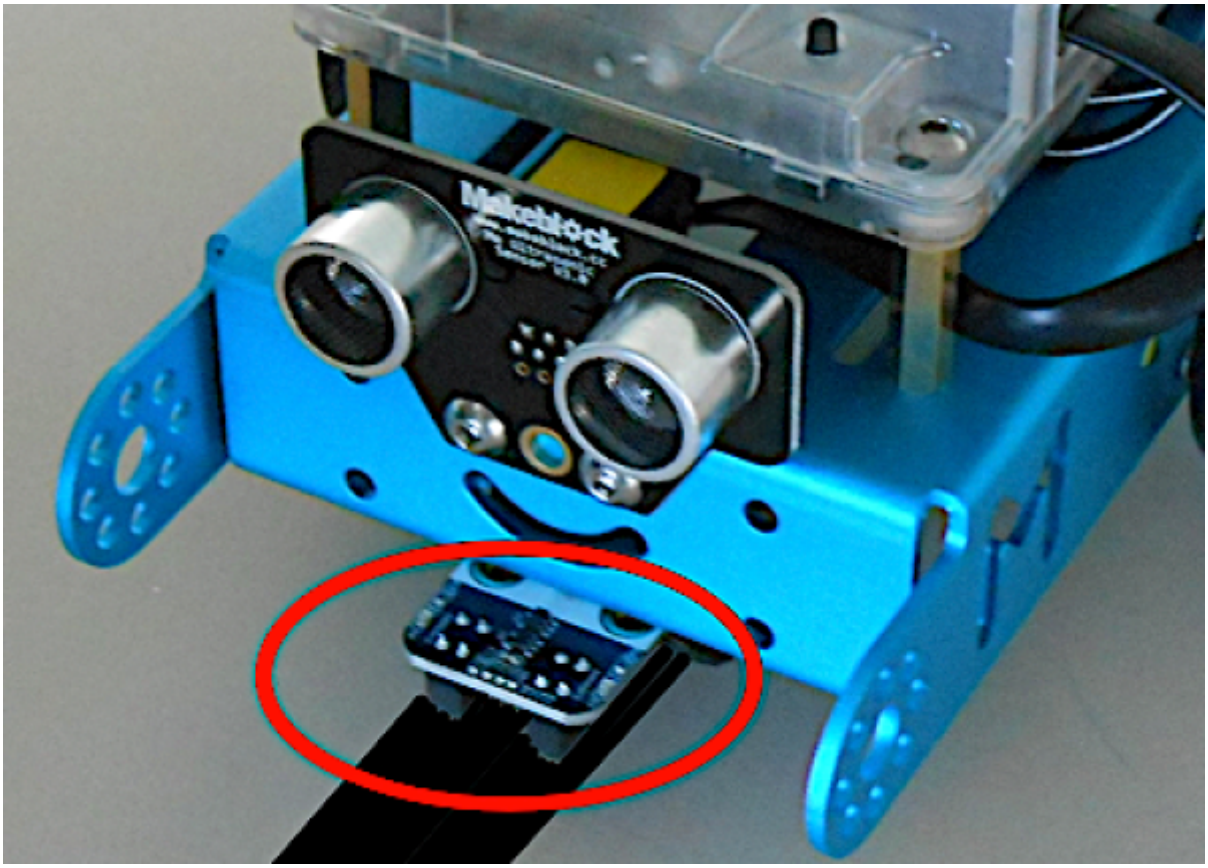
```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeDCMotor motor_L(9);
MeDCMotor motor_R(10);

int buttonPressed() {
  return analogRead(A7) <= 10 ? 1 : 0;
}

void mBot_setMotorLeft(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_L.run((9) == M1 ? -(dir*speed) : (dir*speed));
```

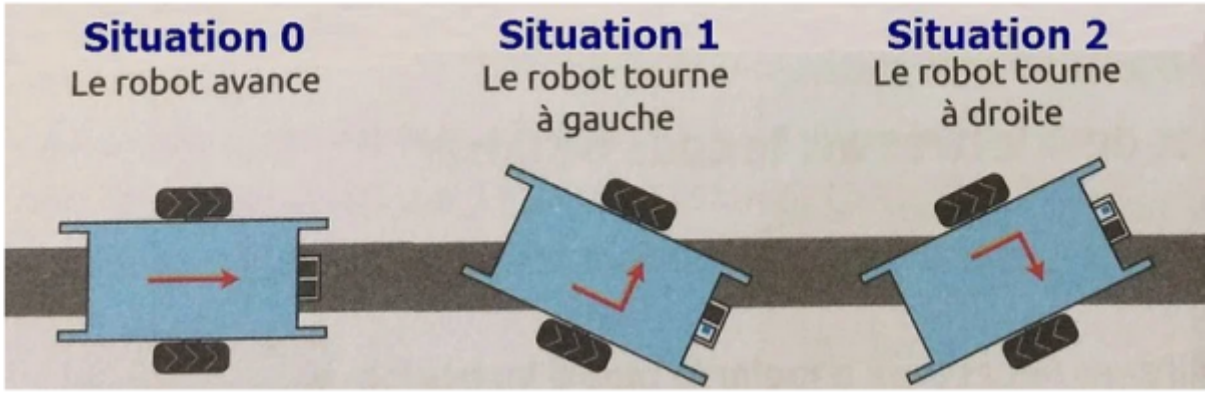
```
}

void mBot_setMotorRight(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_R.run((10) == M1 ? -(dir*speed) : (dir*speed));
}


void setup() {
  pinMode(A7,INPUT);
}

void loop() {
  if (buttonPressed()) {
    mBot_setMotorRight(1, 25);
    mBot_setMotorLeft(1, 25);
    delay(1000*1);
    mBot_setMotorRight(-1, 25);
    mBot_setMotorLeft(-1, 25);
    delay(1000*1);
    mBot_setMotorRight(0, 0);
    mBot_setMotorLeft(0, 0);
  }
}
```

Robot suivi Ligne

robot-mbot-utiliser-les-suiveurs-de-ligne

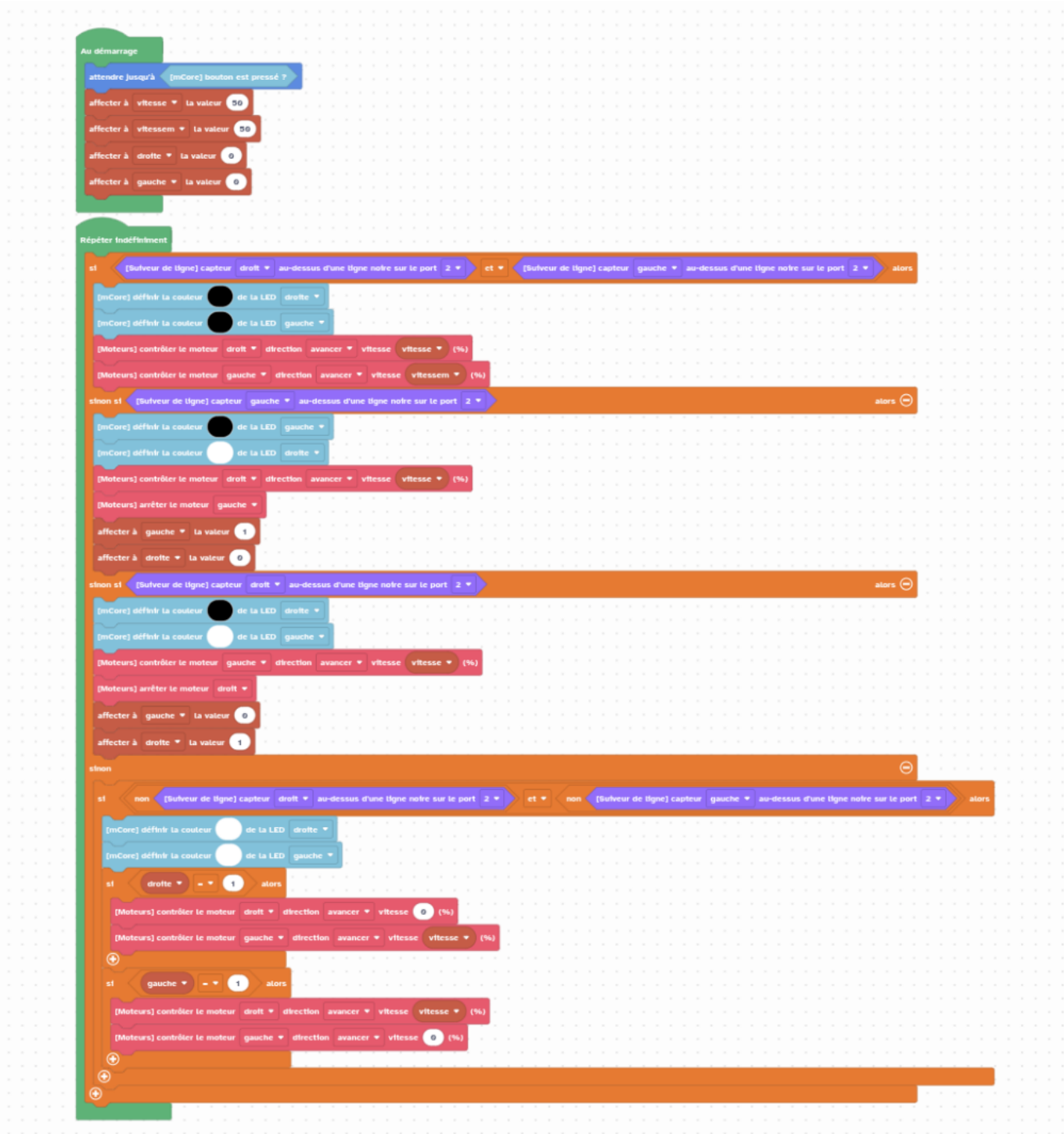
robot_mbot_-_utiliser_les_suiveurs_de_ligne.pdf





correctionsuiviligneexo4_vittascience_20251020_193057.ino.zip

Robot circuit Vittascience test gauchedroite variables 20251121



suivilignevariable.ino

```
#include <MeMCore.h>
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

// Line Finder on PORT_2
```

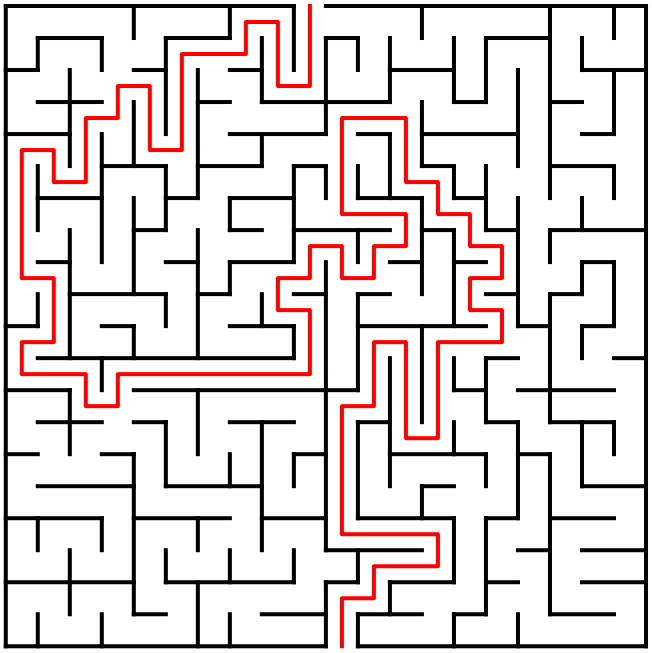
```
MeLineFollower lineFinder_2(PORT_2);
MeRGBLed rgbled_board(7, 2);
MeDCMotor motor_L(9);
MeDCMotor motor_R(10);

int vitesse;
int vitessem;
int droite;
int gauche;

int buttonPressed() {
  return analogRead(A7) <= 10 ? 1 : 0;
}

void mBot_setMotorLeft(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_L.run((9) == M1 ? -(dir*speed) : (dir*speed));
}

void mBot_setMotorRight(int8_t dir, int16_t speed) {
  speed = speed/100.0*255;
  motor_R.run((10) == M1 ? -(dir*speed) : (dir*speed));
}


void setup() {
  pinMode(A7,INPUT);
  while (!buttonPressed() ) {}
  vitesse = 50;
  vitessem = 50;
  droite = 0;
  gauche = 0;
}

void loop() {
  if (!lineFinder_2.readSensor2() && !lineFinder_2.readSensor1()) {
    rgbled_board.setColor(1, 0, 0, 0);
    rgbled_board.show();
    rgbled_board.setColor(2, 0, 0, 0);
    rgbled_board.show();
    mBot_setMotorRight(1, vitesse);
    mBot_setMotorLeft(1, vitessem);
  } else if (!lineFinder_2.readSensor1()) {
    rgbled_board.setColor(2, 0, 0, 0);
    rgbled_board.show();
    rgbled_board.setColor(1, 255, 255, 255);
    rgbled_board.show();
    mBot_setMotorRight(1, vitesse);
    mBot_setMotorLeft(0, 0);
    gauche = 1;
    droite = 0;
```

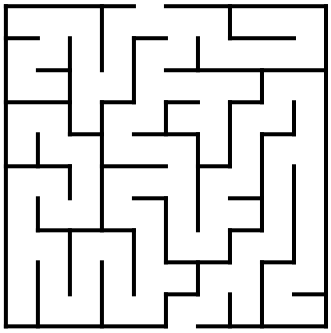
```
  } else if (!lineFinder_2.readSensor2()) {
    rgbled_board.setColor(1, 0, 0, 0);
    rgbled_board.show();
    rgbled_board.setColor(2, 255, 255, 255);
    rgbled_board.show();
    mBot_setMotorLeft(1, vitesse);
    mBot_setMotorRight(0, 0);
    gauche = 0;
    droite = 1;
  } else {
    if (!!lineFinder_2.readSensor2() && !!lineFinder_2.readSensor1()) {
      rgbled_board.setColor(1, 255, 255, 255);
      rgbled_board.show();
      rgbled_board.setColor(2, 255, 255, 255);
      rgbled_board.show();
      if (droite == 1) {
        mBot_setMotorRight(1, 0);
        mBot_setMotorLeft(1, vitesse);
      }
      if (gauche == 1) {
        mBot_setMotorRight(1, vitesse);
        mBot_setMotorLeft(1, 0);
      }
    }
  }
}
```

Sortie d un Labyrinthe avec un mBot

Generateur de labyrinthe

Exemples





l_algorithme_de_pledge_-_interstices_-_interstices.pdf

grain_d_usage_labyrinthe_sams_0.pdf

resolution_de_labyrinthe.pdf

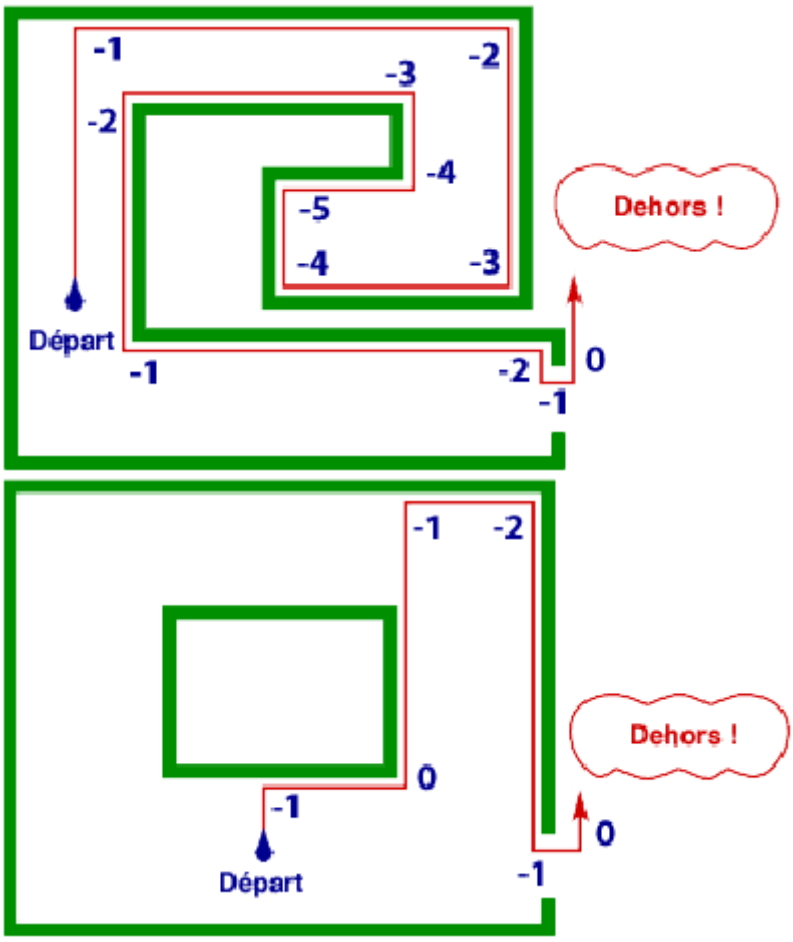
idées d'algorythme pour sortir d un labytrinthe

algolaby001.txt

```
Initialisation :
       *1 Démarrage du robot MBot.
       *2 Activation des capteurs de distance (Yeux).
       *3 Configuration des paramètres de vitesse et de détection.
   *Détection d'obstacle :
       *4 Si la distance est supérieure à 20 cm, le robot accélère.
       *5 Si la distance est entre 20 cm et 10 cm, le robot maintient
une vitesse constante.
       *6 Si la distance est inférieure à 10 cm mais supérieure à 5 cm,
le robot ralentit.
```

```
       *7 Si la distance est inférieure à 5 cm, le robot s'arrête.
   *Orientation pour trouver la sortie :
       *8 Si le robot est arrêté à cause d'un obstacle à 5 cm, il
effectue une séquence d'orientations pour trouver la sortie.
       *9 Si après un virage à droite de 90°, il n'y a pas d'obstacle
proche, le robot avance.
       *10 Sinon, s'il effectue un demi-tour (180°) et ne détecte pas
d'obstacle proche, le robot avance.
       *11 Sinon, après un virage à gauche de 90°, le robot avance (car
il aura effectué un demi-tour).
   *Répétition du processus :
       *12 Le robot continue à avancer tout en détectant les obstacles
et ajustant sa vitesse en conséquence.
       *13 En cas d'obstacle à 5 cm, il effectue les étapes
d'orientation pour trouver la sortie.
   *Fin :
       *14 Arrêt du robot lorsque la sortie du labyrinthe est détectée.
```

lalgorithme-de-pledge



algolaby00.txt

```
il ne suffit pas de marcher en ligne droite. Il faut compter les
```

```
changements de direction.

Supposons que, comme dans les exemples précédents, tous les angles
soient droits. On n’a alors que deux possibilités,
tourner à droite ou à gauche selon un angle de 90°. On compte les
changements de direction en augmentant d’un point
lorsque l’on tourne à gauche et en diminuant d’un point lorsque l’on
tourne à droite (y compris la première fois
que l’on tourne à droite quand on atteint un mur). Au début, le
décompte est à zéro. Les deux instructions sont
alors les suivantes :

    *1 Aller tout droit jusqu’au mur, passer à l’instruction 2 ;
    *2 Longer le mur par la droite (ou par la gauche, mais toujours
dans le même sens) jusqu’à
ce que le décompte des changements de direction atteigne zéro, passer à
l’instruction 1 ;

Il faut répéter ces actions jusqu’à ce que l’on revienne à la lumière
du jour.
```

Algorithmes de sortie d un labyrinthe

Voici une **présentation claire des principaux algorithmes de résolution de labyrinthe pour un robot**, du plus simple au plus avancé. J’indique à chaque fois **le principe**, **les avantages** et **les limites**, avec un peu de pseudo-code quand c’est utile.

---

1. Algorithme de la (ou main gauche)

Principe

Le robot garde **toujours la main droite en contact avec un mur**. Tant que le labyrinthe est **simplement connexe**, il finira par trouver la sortie.

Avantages

- Très simple à implémenter
- Peu de mémoire
- Adapté aux robots simples (Arduino, capteurs IR)

Limites

- Ne fonctionne pas si le labyrinthe a des îlots (murs isolés)
- Pas forcément le chemin le plus court

Pseudo-code

```
tant que sortie non trouvée :
    si mur à droite absent :
        tourner à droite
        avancer
    sinon si devant libre :
        avancer
    sinon :
        tourner à gauche
```

---

2. Algorithme de (marquage des chemins)

Principe

Le robot **marque les passages déjà visités** :

- 1 marque → déjà visité
- 2 marques → cul-de-sac

Il évite de repasser inutilement au même endroit.

Avantages

- Fonctionne dans tous les labyrinthes
- Garantit de trouver la sortie

Limites

- Besoin de mémoire (ou marquage physique)
- Plus complexe que la main droite

3.

Principe

Le robot explore un chemin **jusqu'au bout**, puis revient en arrière quand il est bloqué.

Avantages

- Simple conceptuellement
- Garantie de trouver la sortie

Limites

- Peut être très long
- Pas le plus court chemin

Pseudo-code simplifié

```
fonction DFS(case):
    marquer case visitée
    si case = sortie :
        fin
    pour chaque voisin libre :
        si non visité :
            DFS(voisin)
```

4.

Principe

Le robot explore **niveau par niveau** toutes les possibilités.

Avantages

- Trouve **le plus court chemin**
- Très fiable

Limites

- Utilise beaucoup de mémoire
- Moins adapté aux petits robots

---

5. * (A-star)

Principe

Algorithme intelligent utilisant :

- le coût déjà parcouru
- une estimation de la distance jusqu’à la sortie

Avantages

- Très rapide
- Chemin optimal
- Utilisé en robotique avancée

Limites

- Nécessite une carte du labyrinthe
- Plus complexe à programmer

---

6. Comparatif rapide

Main droite	Très faible	Très simple
Trémaux	Faible	Simple
DFS	Moyenne	Moyenne
BFS	Élevée	Moyenne
A*	Élevée	Complexe

Recommandation selon ton robot

- **Robot simple (capteurs, pas de carte)** → Main droite ou Trémaux
- **Robot avec mémoire et carte** → BFS ou A*
- **Projet scolaire** → DFS ou Trémaux

Voici **l'ajout de la gestion des angles et des virages précis** pour un robot de labyrinthe avec **capteurs IR / ultrasons**, adapté à un **robot à deux roues motrices (différentiel)**.

⚙ Principe de base

Pour tourner **proprement à 90°**, le robot ne se contente pas d'un délai (`delay`) : il utilise soit :

- **le temps** (solution simple)
- **les encodeurs de roues** (précis)
- **un gyroscope (IMU)** (très précis)

Je te montre les **3 niveaux**, du plus simple au plus avancé.

Schéma de rotation du robot

Virage à 90° à droite

```
Vue de dessus

     AVANT
       ↑
       |
   ┌───┴───┐
   │ ROBOT │   → rotation
   └───┬───┘
       |
       ↓
```

```
Roue gauche : AVANCE
Roue droite : RECULE (ou arrêt)
```

NIVEAU 1 – Rotation par le TEMPS (simple)

Schéma logique

```
Tourner à droite :
- Moteur gauche : ON
- Moteur droit : OFF
- Temps ≈ 400 ms → 90°
```

Pseudo-code

```
void tournerDroite90() {
    moteurGauche(AVANCE);
    moteurDroit(ARRET);
    delay(400);   // à calibrer
    arret();
}
```

Avantages

- Très simple
- Suffisant pour projets scolaires

Limites

- Dépend de la batterie et du sol

NIVEAU 2 – Rotation avec ENCODEURS (précis)

Principe

Chaque roue a un encodeur → nombre de ticks = angle.

Schéma

```
Angle = (ticks mesurés / ticks par tour) × 360°
```

Pseudo-code

```
void tournerDroite90() {
    resetEncodeurs();
    moteurGauche(AVANCE);
    moteurDroit(RECULE);
```

```
    while (encodeurGauche < TICKS_90) {
        // attendre
    }
    arret();
}
```

✅ Avantages

- Bonne précision
- Indépendant du sol

🧭 NIVEAU 3 – Rotation avec GYROSCOPE (IMU)

Principe

Le robot mesure directement l'angle de rotation (yaw).

Schéma

```
Angle cible = angle actuel ± 90°
```

Pseudo-code

```
void tournerDroite90() {
    float angleInitial = lireYaw();

    moteurGauche(AVANCE);
    moteurDroit(RECULE);

    while (lireYaw() < angleInitial + 90) {
        // rotation
    }
    arret();
}
```

✅ Avantages

- Très grande précision
- Utilisé en robotique avancée

🧠 Correction de trajectoire (ligne droite)

Schéma

```
Mur à droite trop proche → corriger à gauche
```

```
Mur à droite trop loin  → corriger à droite
```

Pseudo-code

```
erreur = distanceD - distanceConsigne;

vG = vBase - Kp * erreur;
vD = vBase + Kp * erreur;
```

 C’est un **contrôle proportionnel (P)**.

Schéma global mis à jour

```
Lire capteurs
     ↓
Décision (droite / avant / gauche)
     ↓
Si virage :
   → rotation 90° précise
Sinon :
   → avance + correction mur
     ↓
Répéter
```

Conseils pratiques

- Calibrer les **90°** une seule fois
- Utiliser **encodeurs OU gyroscope** si possible
- Toujours arrêter les moteurs après rotation
- Ajouter un petit délai de stabilisation (50 ms)

Résumé

Temps		
Encodeurs		
Gyroscope		

From:
https://www.magenealogie.chanterie37.fr/www/fablab37110/ - **Castel'Lab le Fablab MJC de Château-Renault**

Permanent link:
https://www.magenealogie.chanterie37.fr/www/fablab37110/doku.php?id=start:arduino:cours:vittascience&rev=1765666564

Last update: **2025/12/13 23:56**