gros_bouton.zip

chrono.zip

PROGRAMME POUR LA BARRIERE

```
#include <Wire.h>
#include <Servo.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Servo myservo;
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

unsigned long prochainDeclenchement = 0;
int pos = 0;
// Capteurs IR
const byte CAPT1 = 2;
const byte CAPT2 = 3;
const byte FEUVERT = 12;
const byte FEUROUGE = 11;
// Distance entre capteurs
const float DISTANCE_M = 20;

// Variables ISR
volatile unsigned long t1 = 0;
volatile unsigned long t2 = 0;
volatile bool capteur1Declenche = false;
volatile bool mesurePrete = false;

// Anti-rebond
volatile unsigned long lastTrig1 = 0;
volatile unsigned long lastTrig2 = 0;
const unsigned long antiRebondUs = 5000;  // 5 ms

// Timeout de mesure
const unsigned long timeoutMesureUs = 3000000UL; // 3 s

// Filtrage dt
const unsigned long dtMinUs = 5000UL;      // 5 ms
const unsigned long dtMaxUs = 2000000UL;   // 2 s

// Résultats
float vitesse_ms = 0.0;
float vitesse_kmh = 0.0;
```

```
// Gestion affichage temporisé
unsigned long affichageJusqua = 0;
bool afficherResultat = false;

void ISR_capteur1() {
  unsigned long now = micros();

  if (now - lastTrig1 < antiRebondUs) return;
  lastTrig1 = now;

  t1 = now;
  capteur1Declenche = true;
  mesurePrete = false;
}

void ISR_capteur2() {
  unsigned long now = micros();

  if (now - lastTrig2 < antiRebondUs) return;
  lastTrig2 = now;

  if (capteur1Declenche && now > t1) {
    t2 = now;
    mesurePrete = true;
    capteur1Declenche = false;
  }
}

void afficheAttente() {
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);

  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Mesure vitesse");

  display.setTextSize(2);
  display.setCursor(10, 22);
  display.println("ATTENTE");

  display.display();
}

void afficheVitesse(float vms, float t) {
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);

  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Vitesse mesuree");
```

```
  display.setTextSize(2);
  display.setCursor(0, 18);
  display.print(vms, 2);
  display.println(" cm/s");

  display.setTextSize(2);
  display.setCursor(0, 42);
  display.print(t, 3);
  display.println(" sec");

  display.display();
}

void barriere() {
  digitalWrite(FEUVERT,LOW);
  digitalWrite(FEUROUGE,HIGH);
  Serial.println("Test servo setup -> 90");
  for (int i=0;i<90;i++) {
    myservo.write(i);
    delay(15);
  }

  Serial.println("Test servo setup -> 0");
   for (int i=90;i>0;i--) {
    myservo.write(i);
    delay(15);
  }
  digitalWrite(FEUROUGE,LOW);
  digitalWrite(FEUVERT,HIGH);
}

void setup() {
  randomSeed(analogRead(A0));  // initialise l'aléatoire
  prochainDeclenchement = millis() + random(4000, 8000);
  Serial.begin(9600);
  myservo.attach(6);
  myservo.write(0);
  delay(500);


  pinMode(CAPT1, INPUT_PULLUP);
  pinMode(CAPT2, INPUT_PULLUP);
  pinMode(FEUVERT, OUTPUT);
  pinMode(FEUROUGE, OUTPUT);
  digitalWrite(FEUVERT,HIGH);
  digitalWrite(FEUROUGE,LOW);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    while (1) {}
  }
```

```
  afficheAttente();

  attachInterrupt(digitalPinToInterrupt(CAPT1), ISR_capteur1, FALLING);
  attachInterrupt(digitalPinToInterrupt(CAPT2), ISR_capteur2, FALLING);
}

void loop() {
  bool calculer = false;
  unsigned long local_t1 = 0;
  unsigned long local_t2 = 0;
  bool local_capteur1Declenche = false;

   noInterrupts();
  if (mesurePrete) {
    local_t1 = t1;
    local_t2 = t2;
    mesurePrete = false;
    calculer = true;
  }
  local_capteur1Declenche = capteur1Declenche;
  unsigned long local_t1_en_cours = t1;
  interrupts();



  // Timeout si capteur 2 ne vient jamais
  if (local_capteur1Declenche) {
    if (micros() - local_t1_en_cours > timeoutMesureUs) {
      noInterrupts();
      capteur1Declenche = false;
      interrupts();

      Serial.println("Timeout mesure");
      afficheAttente();
    }
  }

  if (calculer) {
    unsigned long dt_us = local_t2 - local_t1;

    if (dt_us >= dtMinUs && dt_us <= dtMaxUs) {
      float dt_s = dt_us / 1000000.0;
      vitesse_ms = DISTANCE_M / dt_s;
      vitesse_kmh = vitesse_ms * 3.6;

      Serial.print("t1 = ");
      Serial.print(local_t1);
      Serial.print(" us, t2 = ");
      Serial.print(local_t2);
      Serial.print(" us, dt = ");
      Serial.print(dt_us);
```

```
      Serial.println(" us");

      Serial.print("Vitesse = ");
      Serial.print(vitesse_ms, 3);
      Serial.print(" m/s ; ");


      afficheVitesse(vitesse_ms, dt_s);
      afficherResultat = true;
      affichageJusqua = millis() + 2000;
    } else {
      Serial.print("Mesure rejetee, dt_us = ");
      Serial.println(dt_us);
      afficheAttente();

    }
  }

  if (afficherResultat && millis() >= affichageJusqua) {
    afficherResultat = false;
    afficheAttente();
    if (millis() >= prochainDeclenchement) {

      barriere();   // ta fonction
      prochainDeclenchement = millis() + random(4000, 8000);
    }
  }
}
```