CPGE

Royaume du Maroc
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation

Hockey sur glace : combinaison intelligente

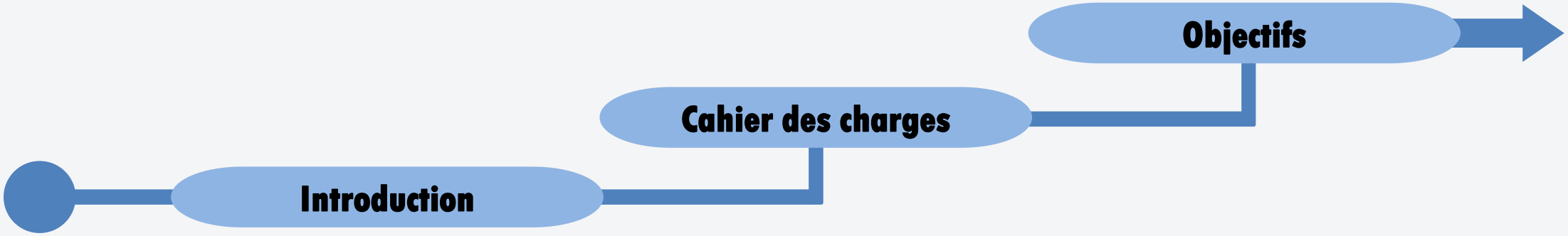
Réalisé par :

Youssef Mazoir

Encadré par :

- **Mr. Abderrahman Ouaanabi**
- **Mr. Youssef Rahou**

Plan de la présentation

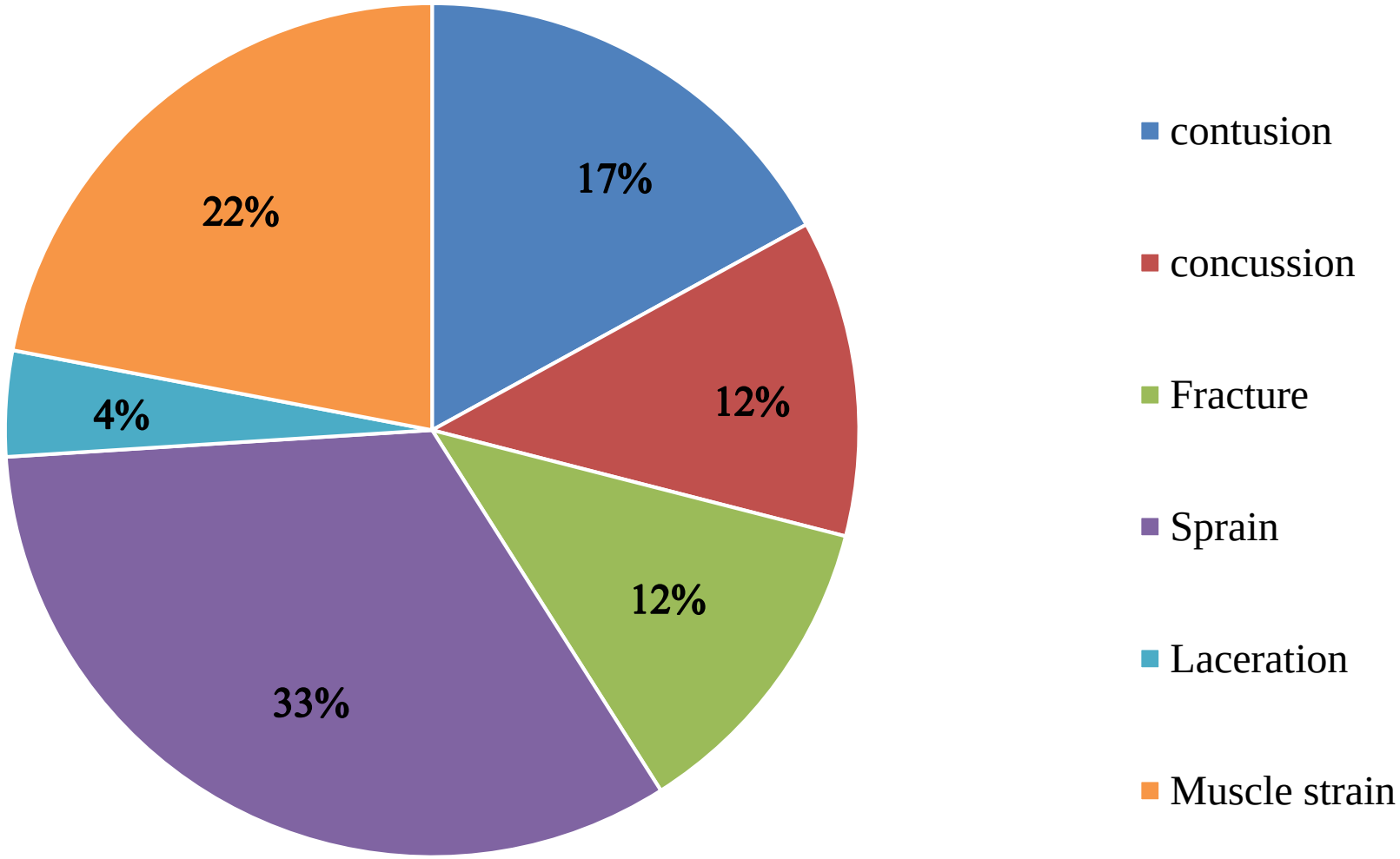


Contextualisation

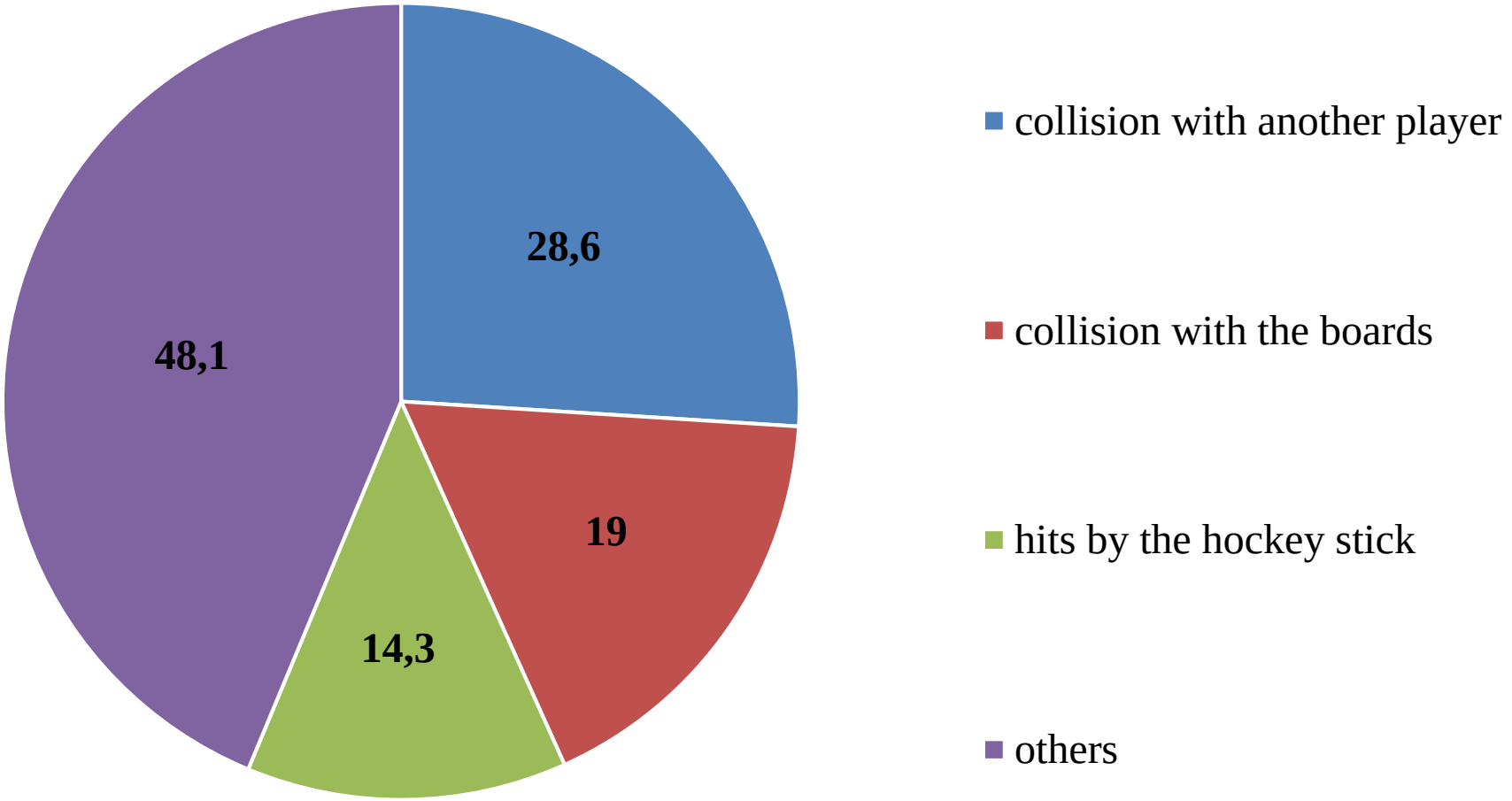
Figure 1 : Plaquage de Max Pacioretty par Zdeno Chara

Figure 2 : transporter Max Pacioretty sur une civière

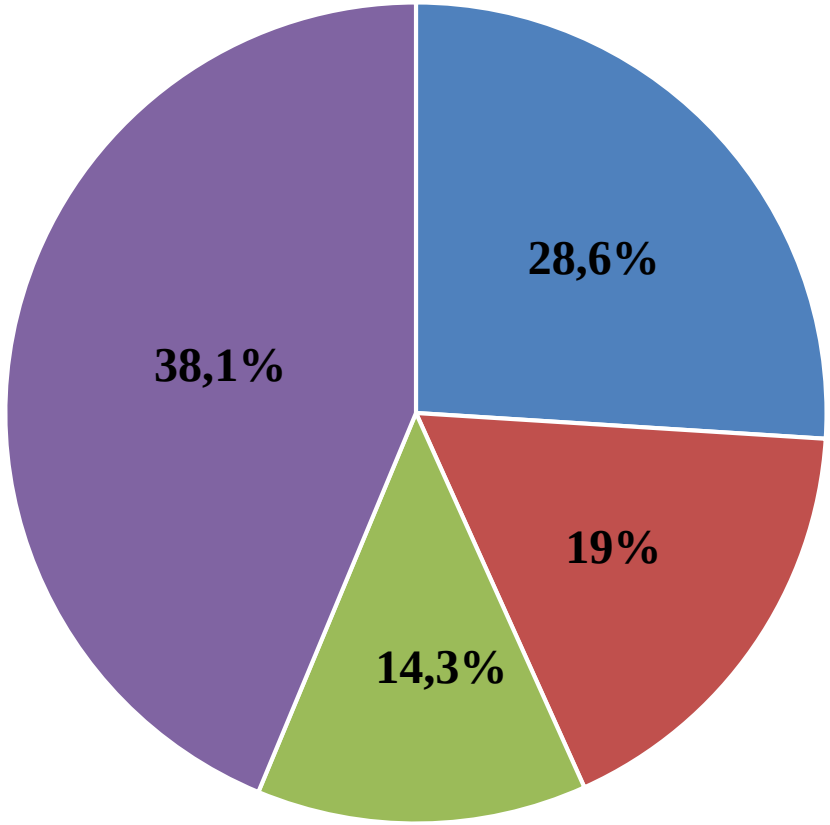
Injuries type distribution leading to TL



the cause of severe injuries (> 28 days off)



the cause of severe injuries (> 28 days off)



- collision with another player
- collision with the boards
- hits by the hockey stick
- others

47,6 %

Collisions ⇊ → **50 % off severe injuries**

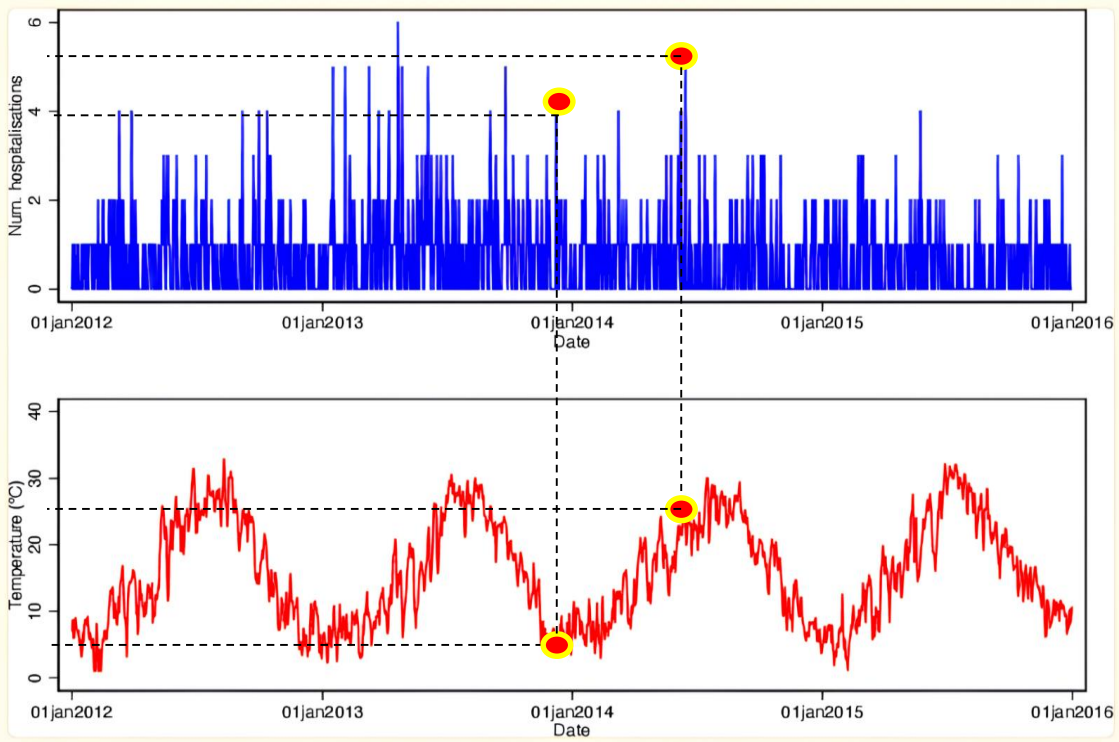


Fig 1 : Daily distribution of emergency visits by sports injuries (top panel) and daily mean ambient temperature (bottom panel) in Madrid, 2012–2015

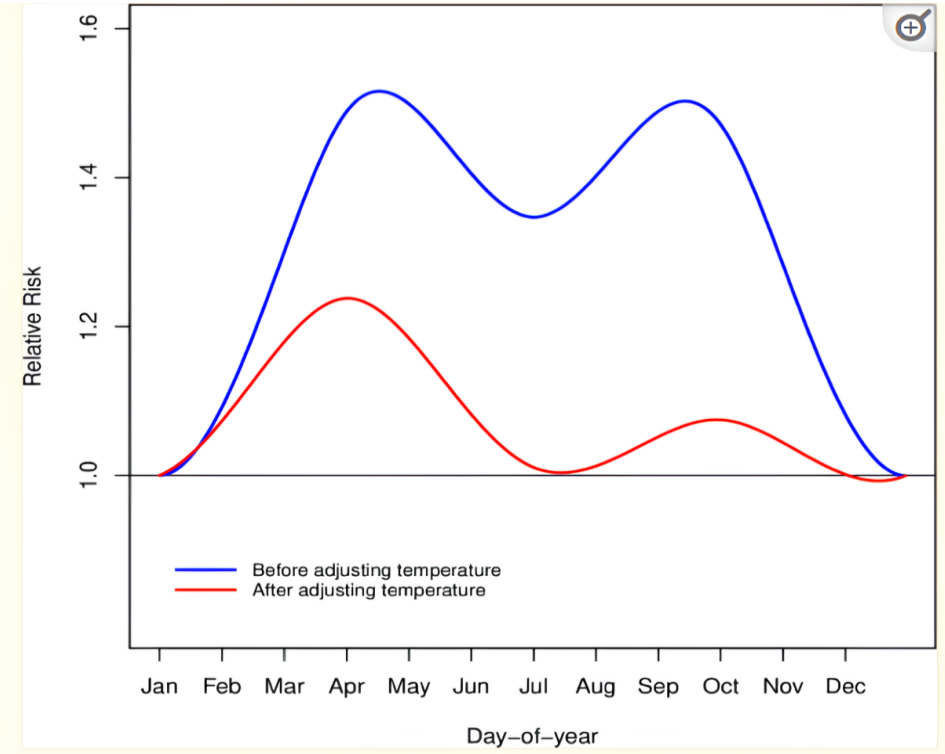
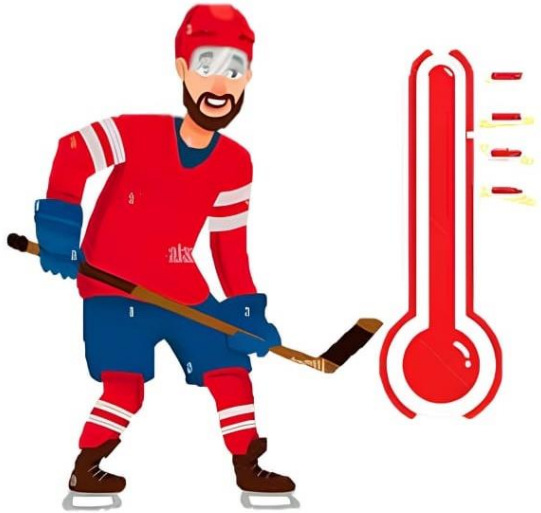
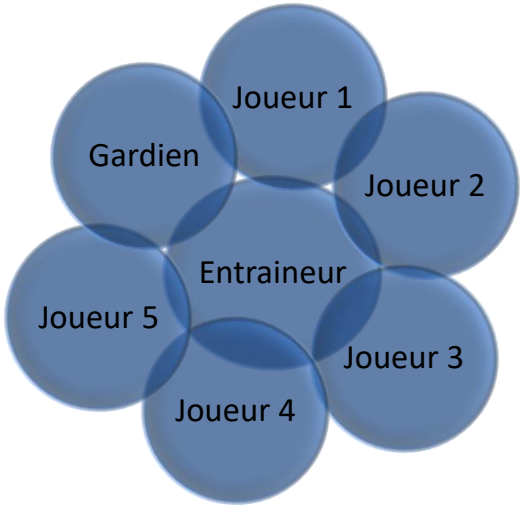


Fig 2 : Relative risk of emergency visits by sports injuries for seasonal changes before and after adjusting for daily ambient temperature

Ajuster la température → Moins de blessures

Problématique





prédire et prévenir ? **T °C interne = constante ?** **communication ?**

Diagramme cas d'utilisation

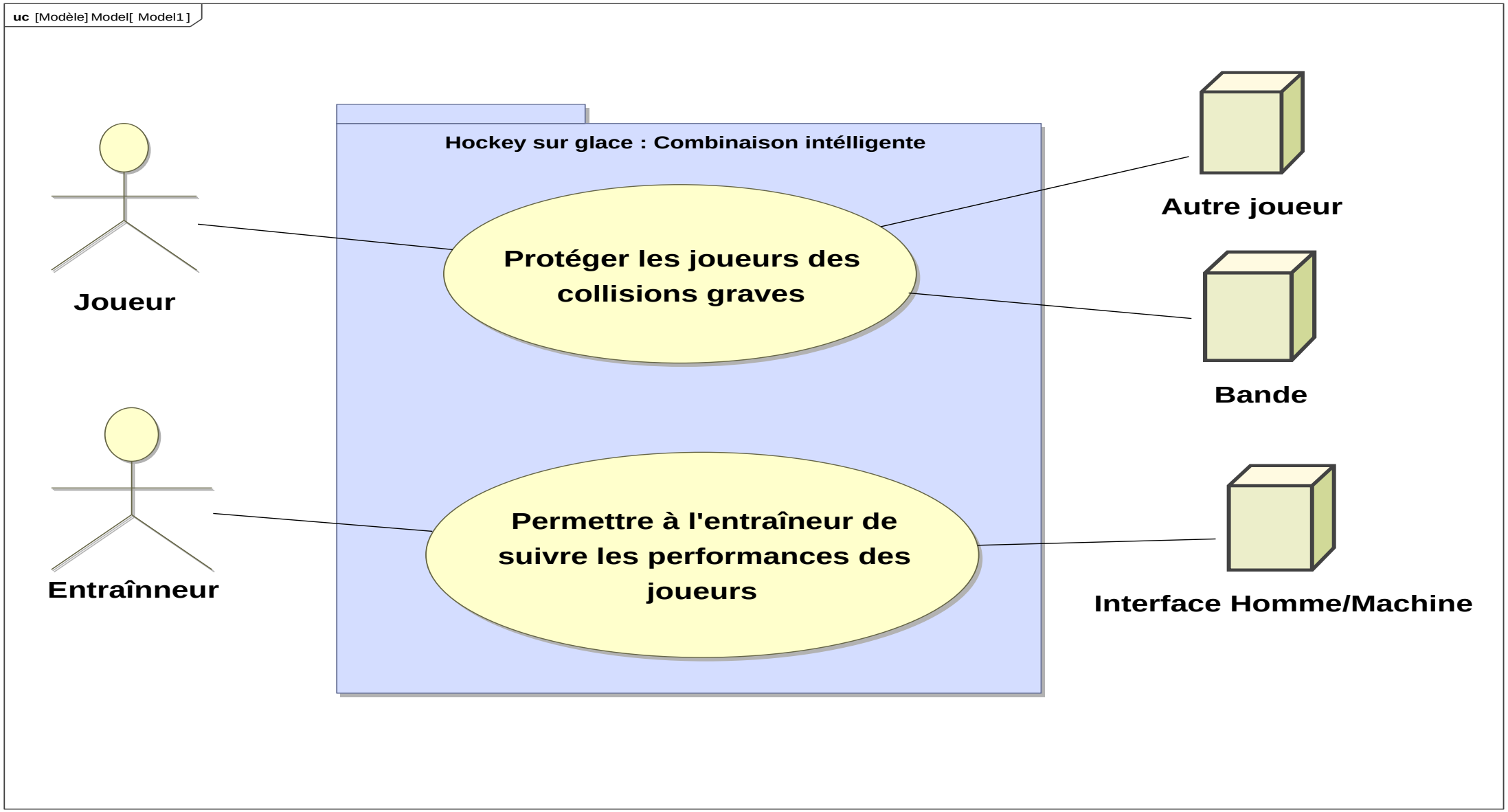


Diagramme des exigences

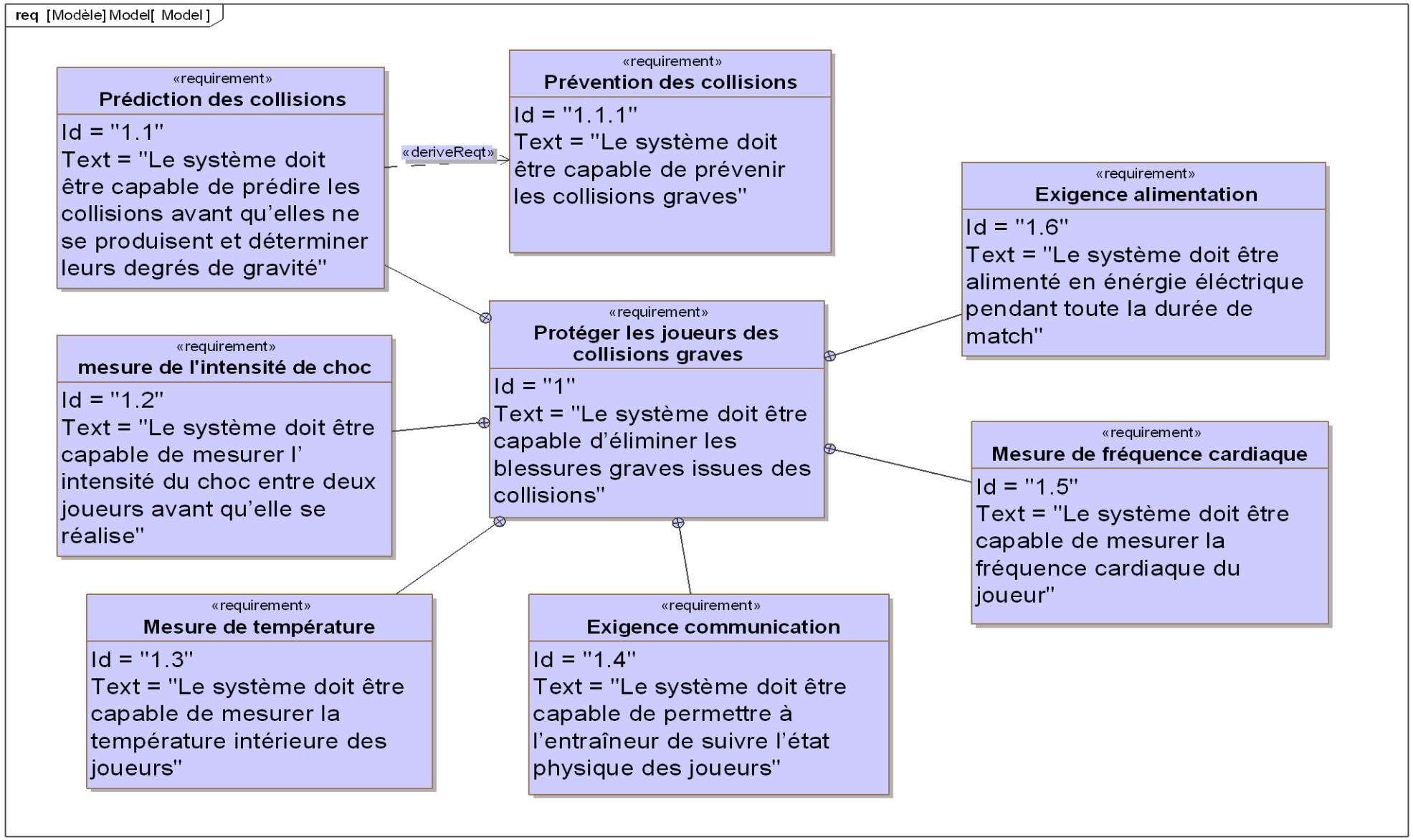
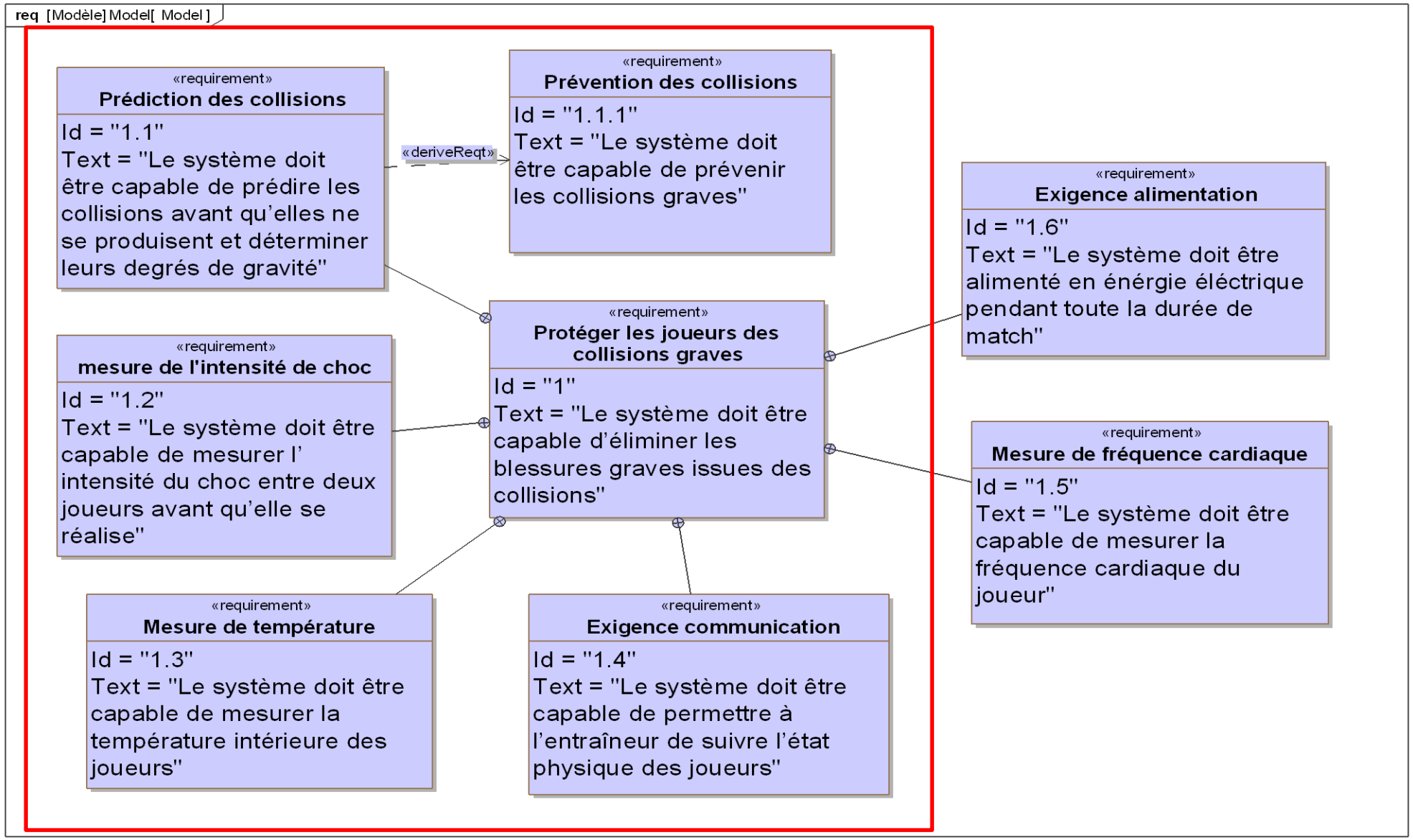


Diagramme des exigences



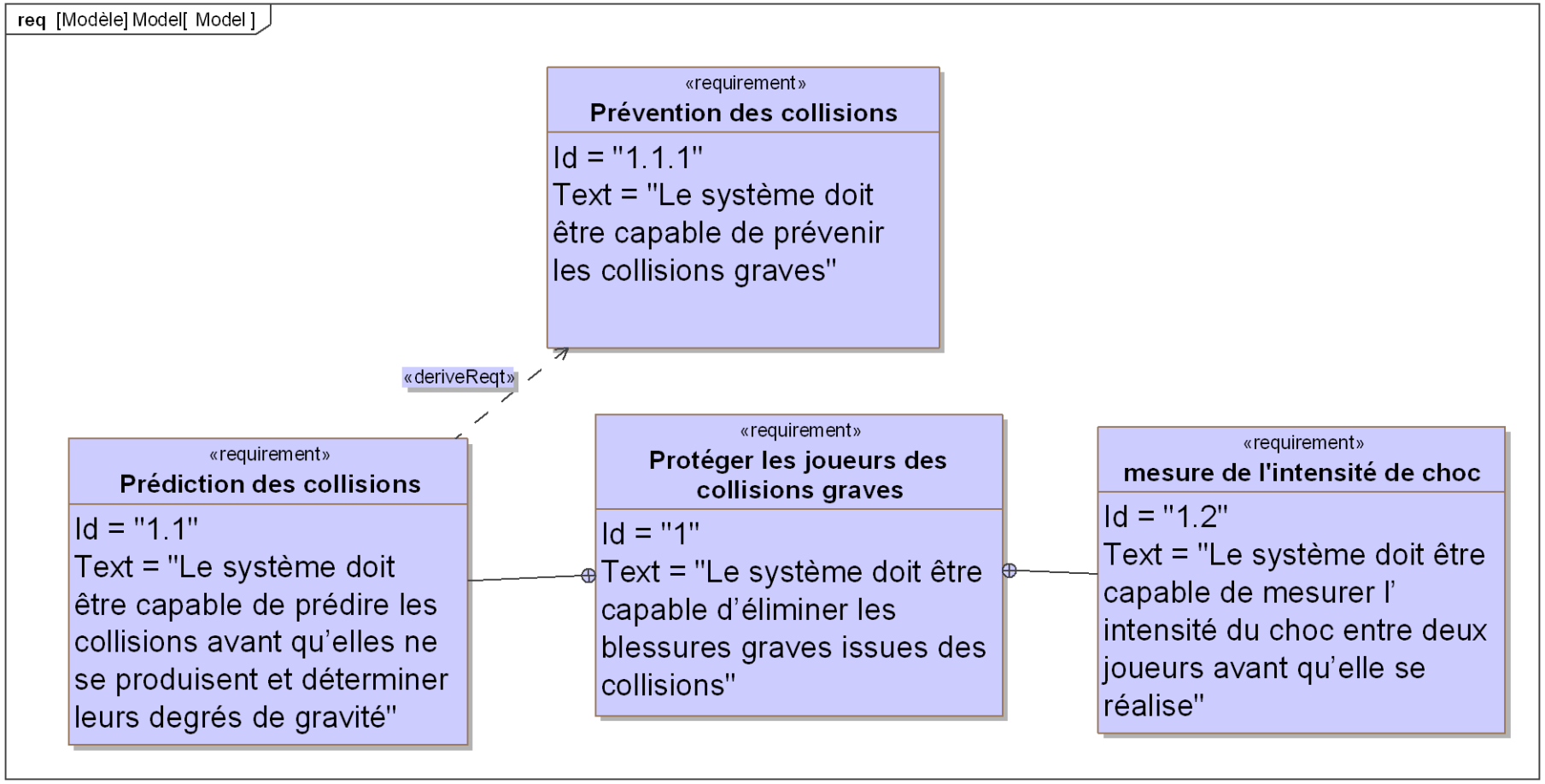
Objectif 1 : Prédire les collisions entre les joueurs et les prévenir si elles présentent un danger

Objectif 2 : Assurer une température interne constante des joueurs

Objectif 3 : Assurer une communication avec l'entraineur

Objectif 1 :

Prédire les collisions entre les joueurs et les prévenir si elles présentent un danger



Objectif 1 : Prédire les collisions entre les joueurs et les prévenir si elle présentent un danger

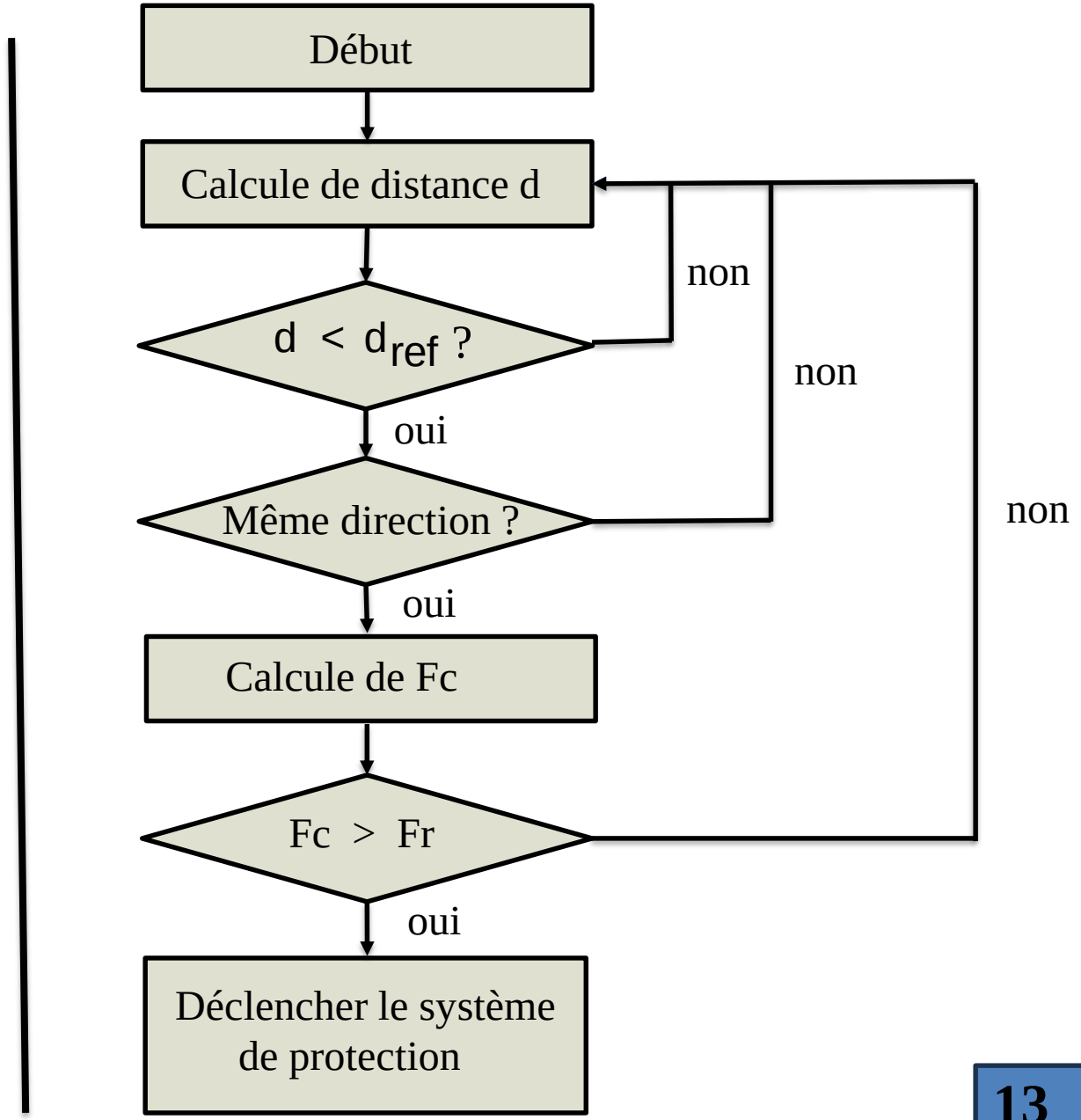
Première partie

1. Termes de référence et Organigramme de fonctionnement

1

Termes de référence

- **Intensité de choc de référence F_r**
- **Distance de référence d_{ref}**



Objectif 1 : Prédire les collisions entre les joueurs et les prévenir si elle présentent un danger

Deuxième partie

1. Calcule de la distance de référence
2. Calcule de l'intensité de choc de référence
3. Mesure de la vitesse

1 Calcule de la distance de référence

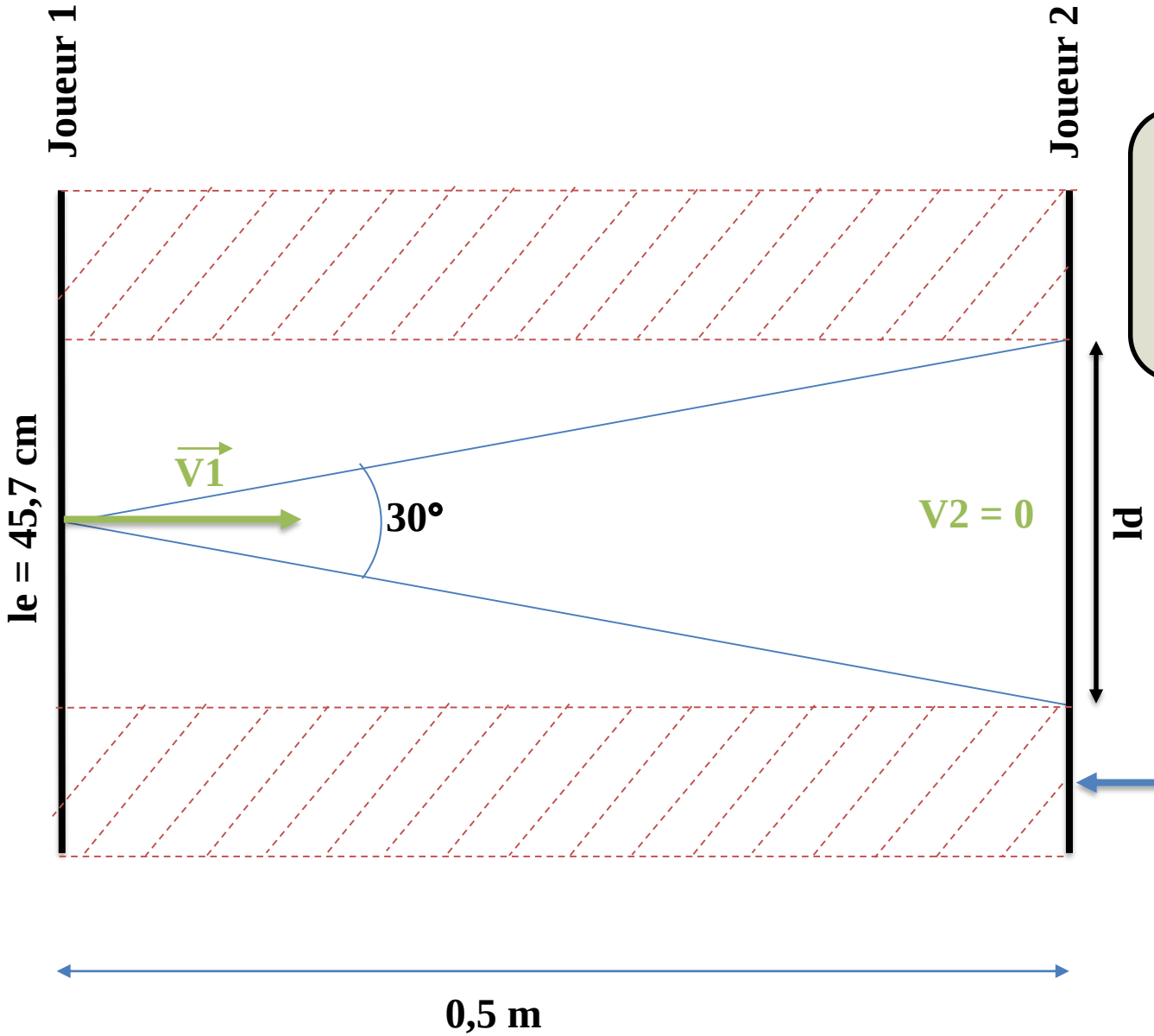
La vitesse maximale atteinte par un joueur de Hockey est : v_m = 44 km/h

Un exemple d'élément de protection une airbag de temps de réaction t_r = 0,1 s

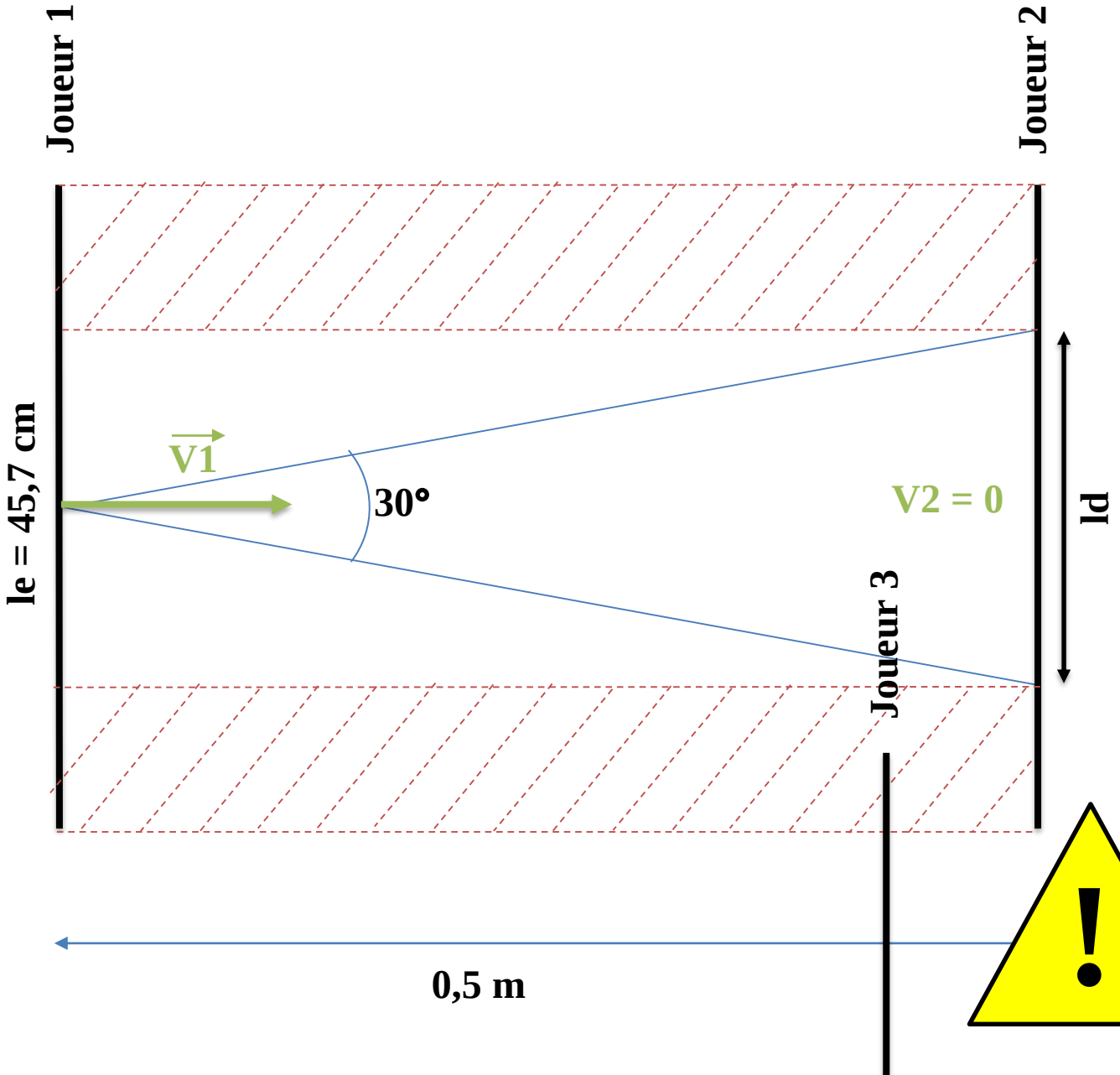
Donc pour donner au système de protéction le temps de réagir, il faut que :

$$d_{ref} >= v_m \times t_r = (44 / 3,6) \times 0,1 = 0,48 \text{ m}$$

$$d_{ref} >= 0,5 \text{ m}$$



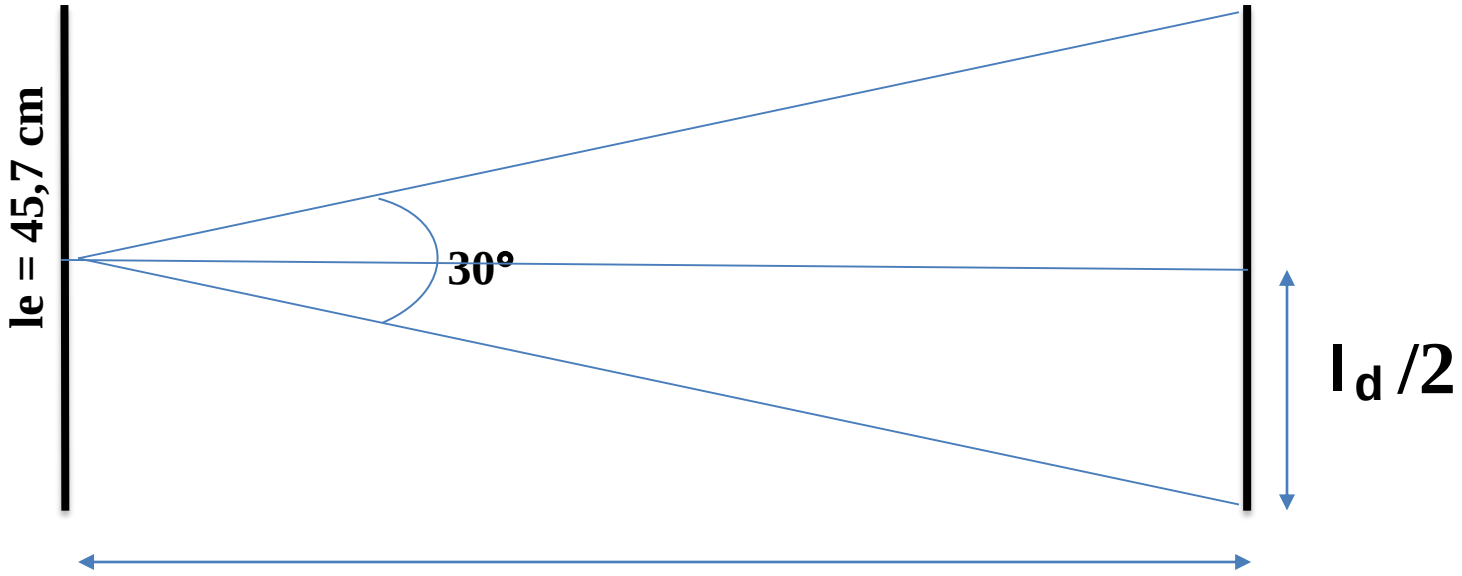
- La largeur moyenne des épaules pour les hommes est le = 45,7cm
- L'angle de détection d'un capteur ultrason est 30°



Donc, il faut assurer une distance de référence tel que:

ld = le

Possibilité de collision sans détection



$$\tan(15°) = \frac{l_d/2}{d_{ref}} \quad \text{Alors} \quad d_{ref} = \frac{l_d/2}{\tan(15°)}$$

$$d_{ref} = 86cm > 50\ cm$$

- Si V est très grande, il est difficile que la direction de joueur va se changer

La condition que les deux joueurs doivent avoir la même direction est automatiquement assuré

2 Calcule de l'intensité de choc de référence

La masse moyenne des joueurs :

$$m_{moy} = 198.9\ lb = 90\ Kg$$

L'intensité de choc de référence :

$$F_r = 12\ kN$$

Selon des études militaires sur des parachutistes : le corps humain d'un sportif pourrait accepter une décélération maxi d'environ 15 G, soit 12 kN pour une masse de 90 kg

Équipes	Moyenne de Poids
Islanders	206,4
Washington	205,4
Vegas	203,5
Dallas	203,5
Toronto	203,4
St-Louis	202,7
Tampa	201,6
Detroit	201,1
Minnesota	200,8
Anaheim	199,9
Floride	199,9
Vancouver	199,7
Winnipeg	199,1
Los Angeles	198,9
Caroline	198,5
Nashville	198,5
Columbus	198,4
Philadelphie	198,2
Buffalo	198,0
Calgary	198,0
Colorado	198,0
San Jose	196,9
Boston	196,9
Rangers	196,4
Edmonton	196,1
Ottawa	195,6
Montréal	194,9
New Jersey	194,4
Phoenix	194,0
Pittsburgh	193,7
Chicago	191,8
Moyenne	198,9

2 Calcule de l'intensité de choc de référence

Relation entre intensité de choc et vitesse :

$$F = m_{moy} \times \frac{\triangle V}{\triangle t}$$

Avec : $\triangle V = Vi - Vf = Vi$

$\triangle t = 0{,}08$ s (La durée moyenne de choc entre deux joueurs)

Pour simplifier les calcules, on prend :

$$V_{iref} = \frac{F_{ref} \times t_r}{m_{moy}} = 10{,}66 \text{ m/s}$$

3 Mesure de la vitesse

$d > d_{ref}$ $d_{ref} = 0,86$ m

- Si $V > V_{iref}$  Danger ➡ Protection

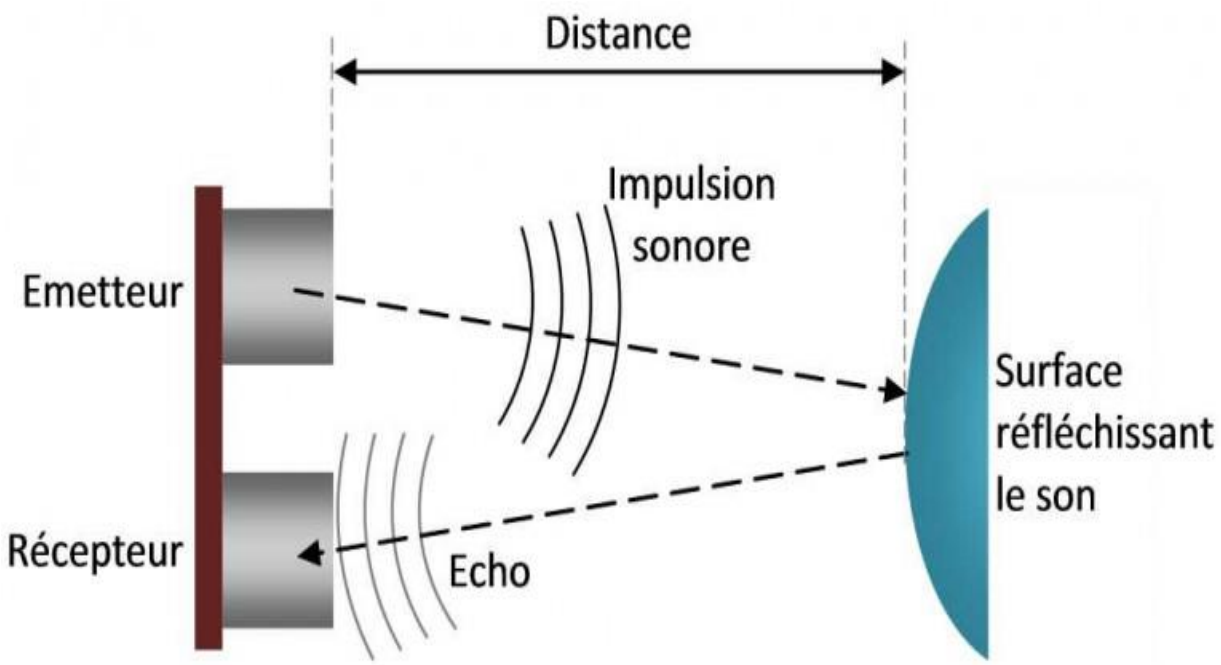
- Si $V < V_{iref}$  Pas de danger ➡ Pas de protection

Objectif 1 : Prédire les collisions entre les joueurs et les prévenir si elles présentent un danger

Troisième partie (solutions techniques)

1. Mesure de la distance
2. Mesure de la vitesse

1 Capteur ultrason



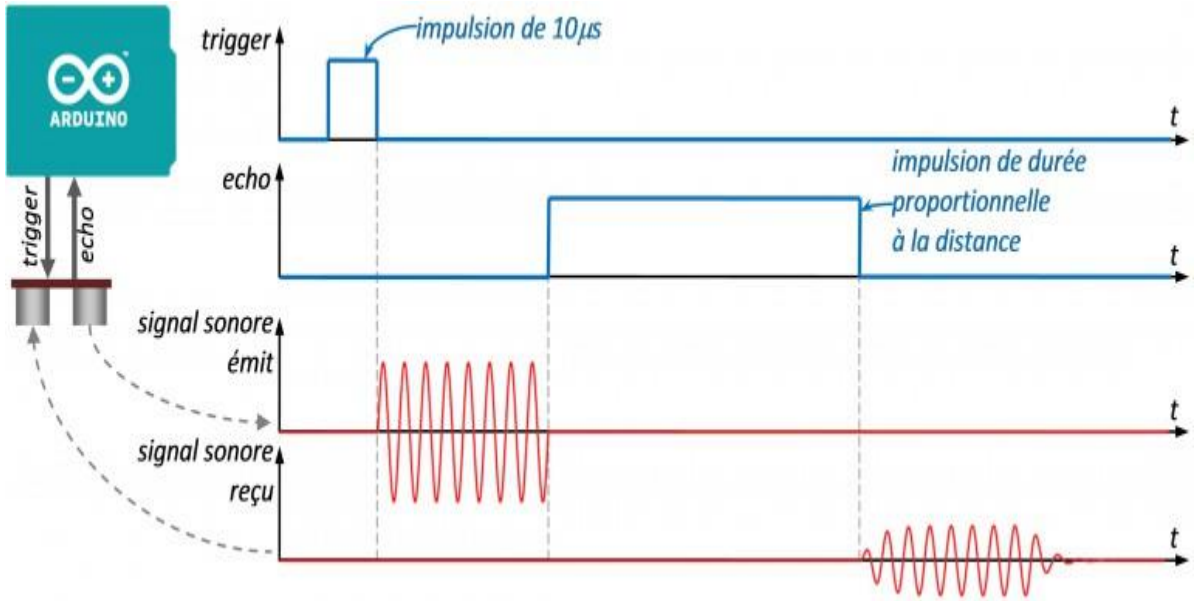
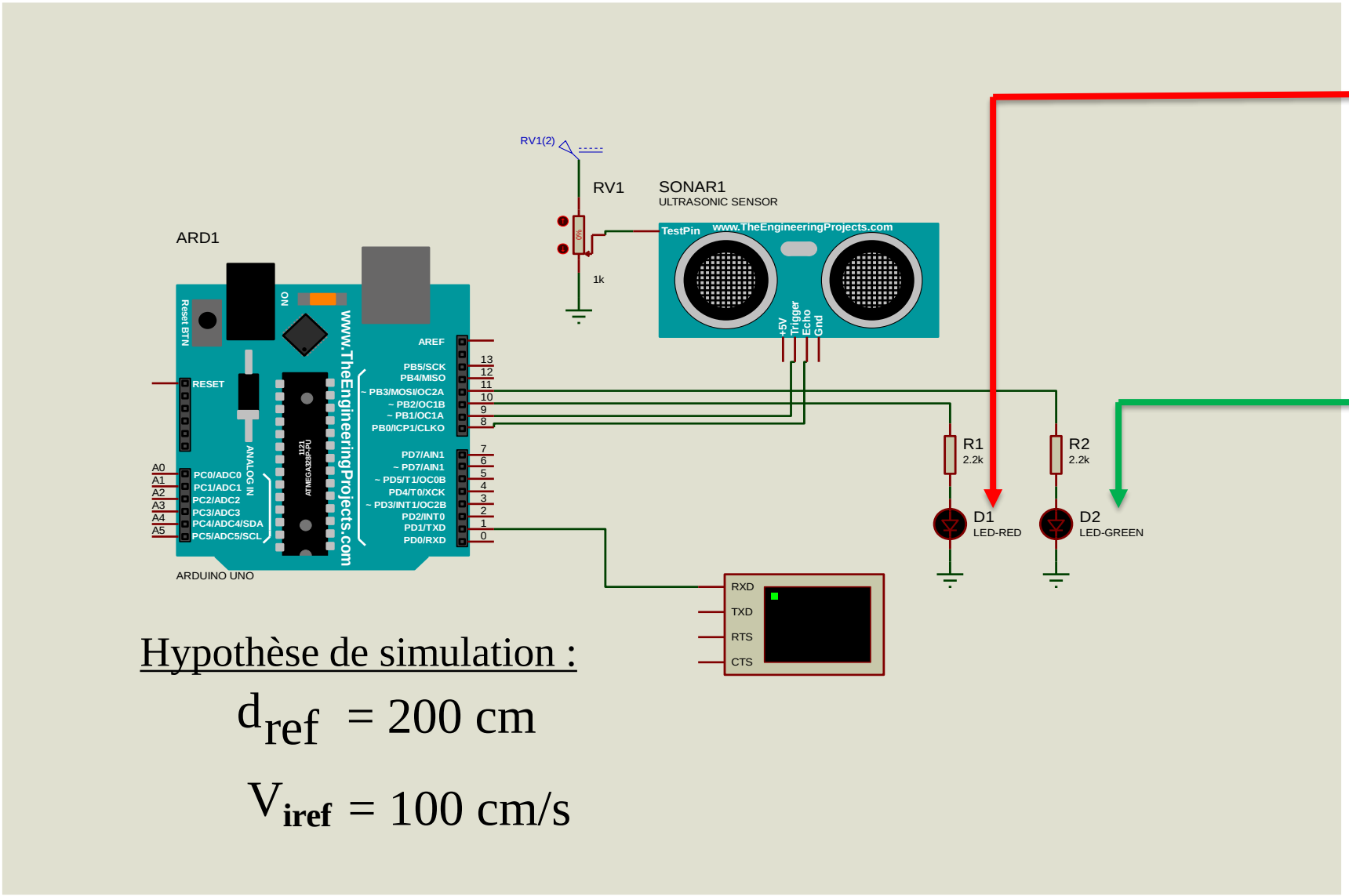


Schéma de simulation sur Proteus



Hypothèse de simulation :

$d_{ref} = 200$ cm

$V_{iref} = 100$ cm/s

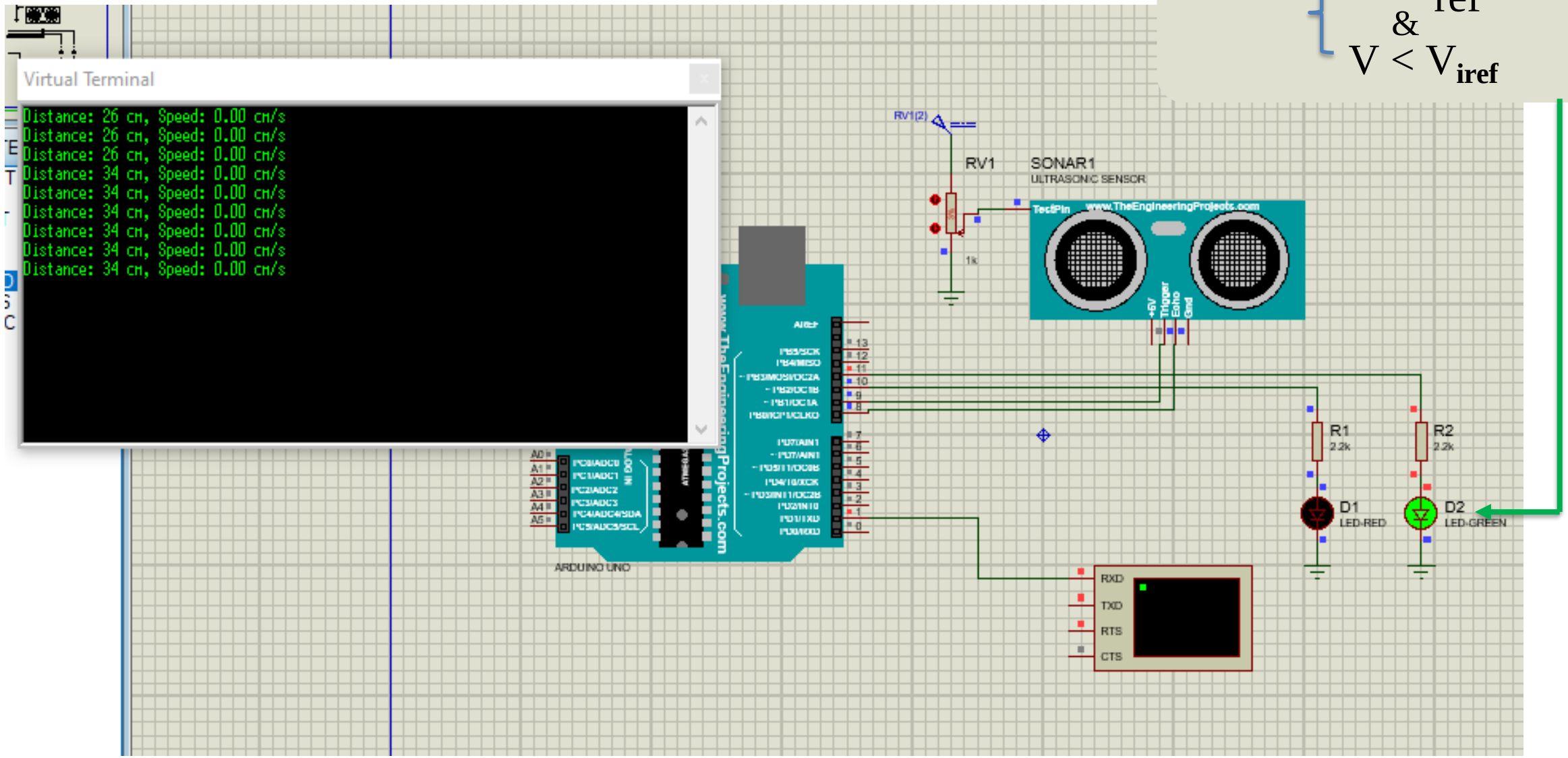
DANGER

Si $d < d_{ref}$ & $V > V_{iref}$

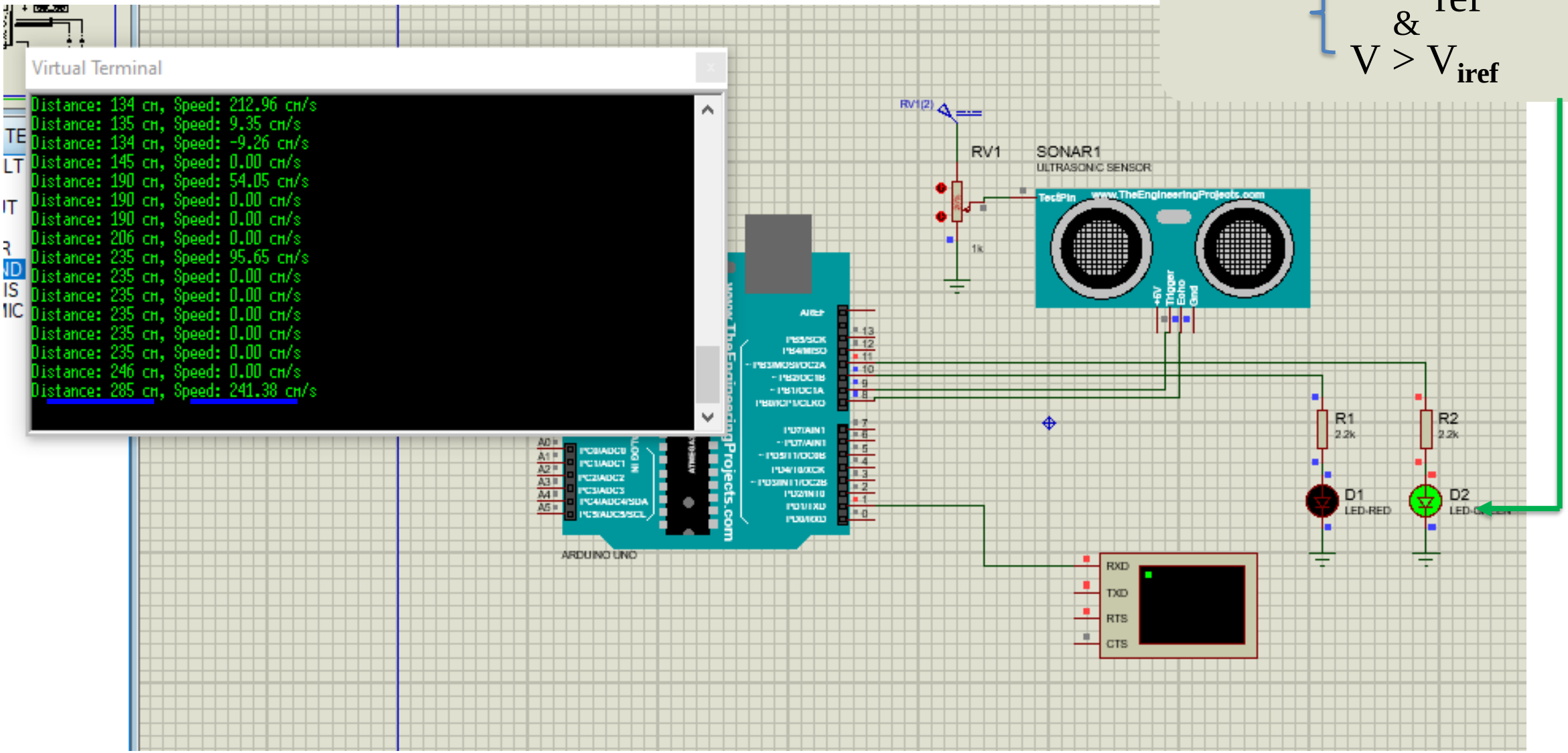
Si $d > d_{ref}$ Or $V < V_{iref}$

PAS DE DANGER

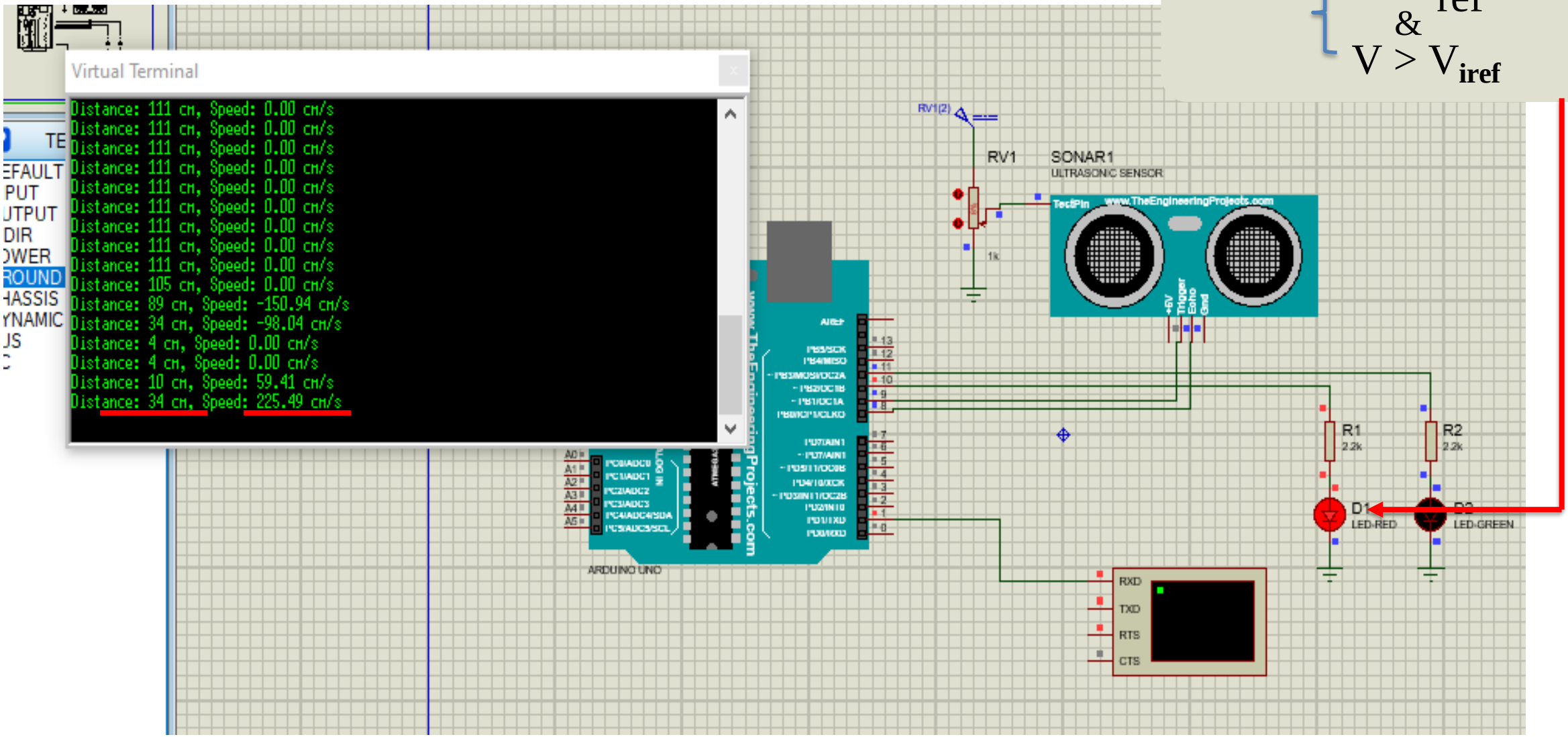
$$\begin{cases} d < d_{ref} \\ \& \\ V < V_{iref} \end{cases}$$



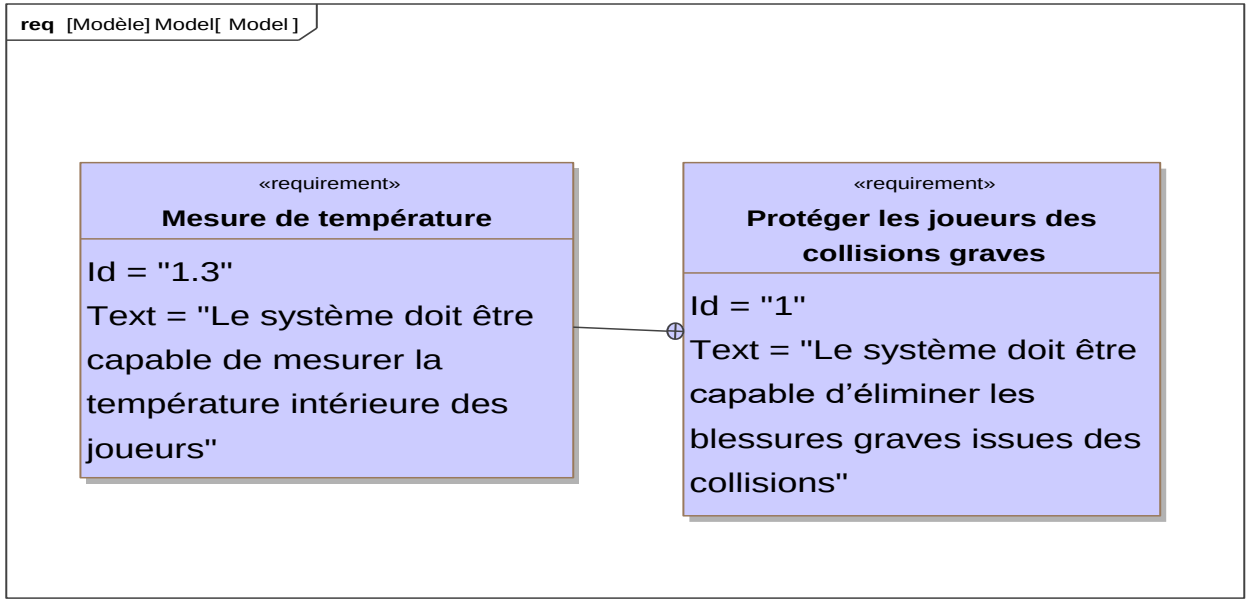
$$\begin{cases} d > d_{ref} \\ \& \\ V > V_{iref} \end{cases}$$



$$\begin{cases} d < d_{ref} \\ \& \\ V > V_{iref} \end{cases}$$



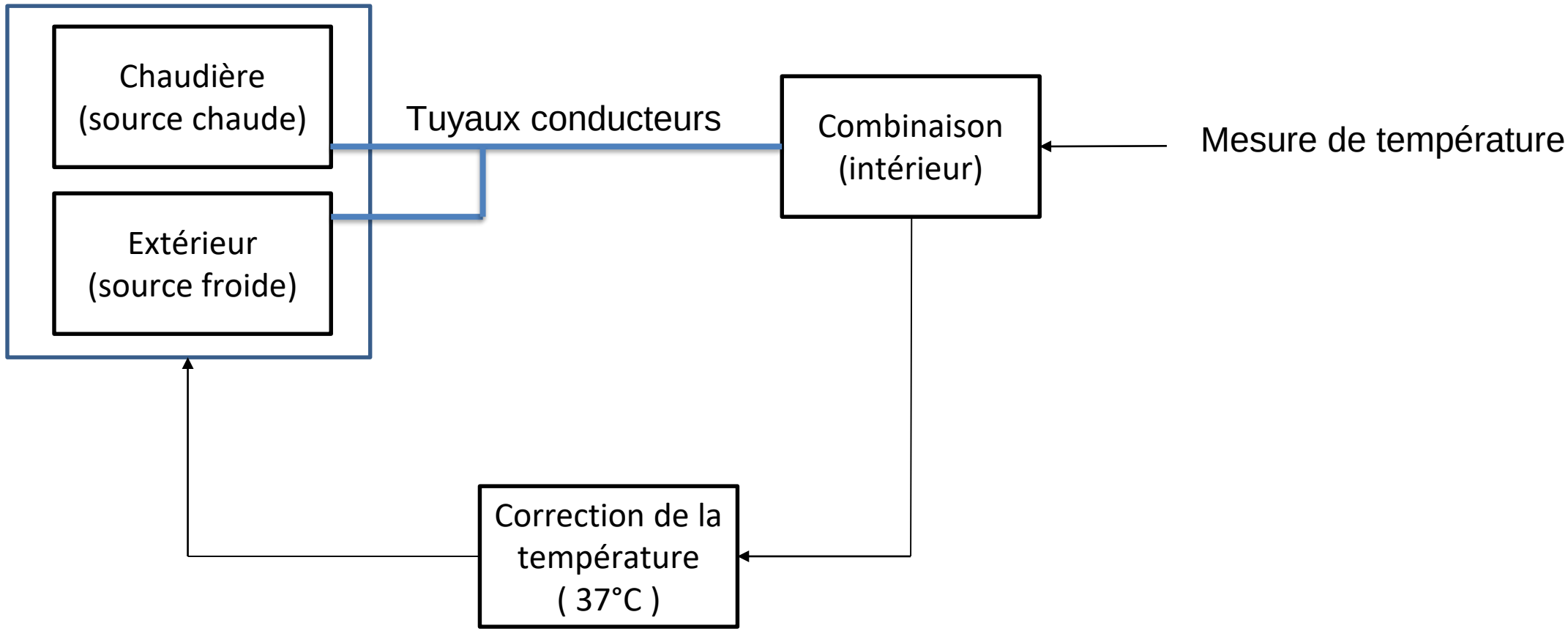
Objectif 2 : Assurer une température interne constante des joueurs



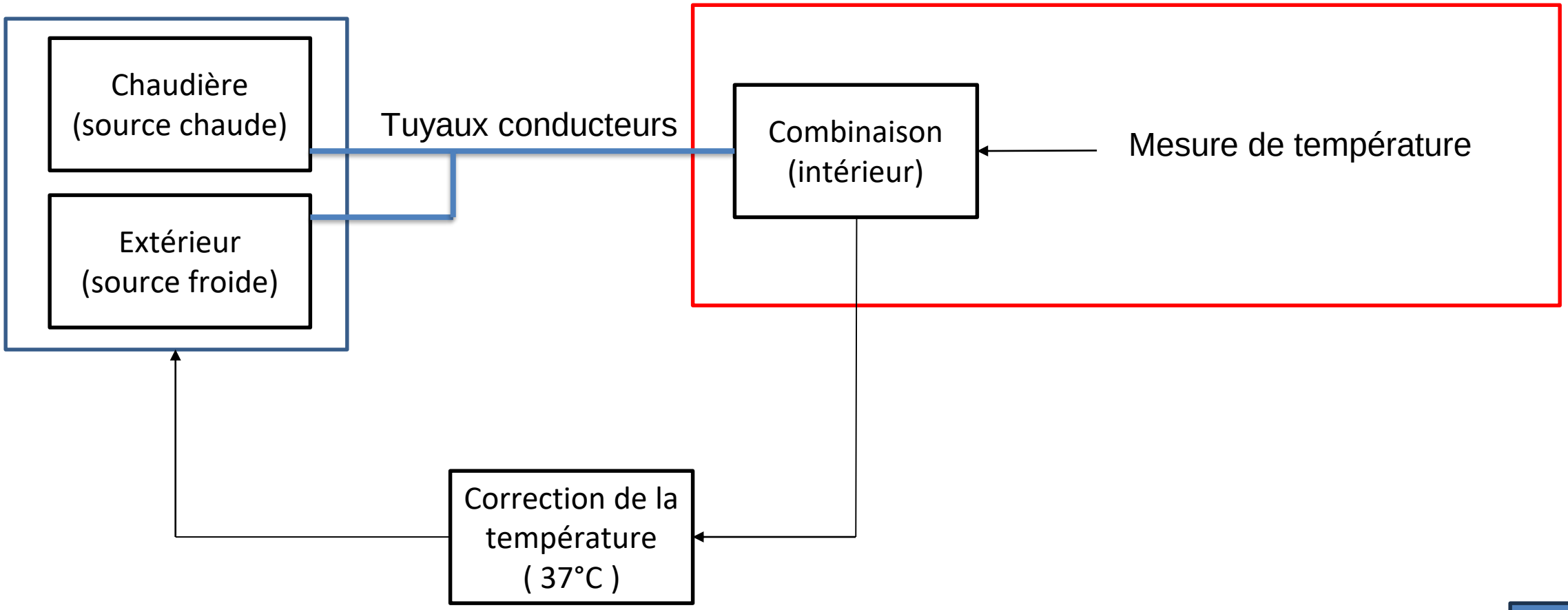
Objectif 2 : Assurer une température interne constante des joueurs

1. Proposé une solution pour la régulation de température
2. Mesure de Temperature

1 Proposé une solution pour la régulation de température



A cause d'insuffisance d'informations, il est difficile de dimensionner un nouveau système de régulation de température. Mais après plusieurs recherches, on a proposer un système qui fonctionne comme le suivant :



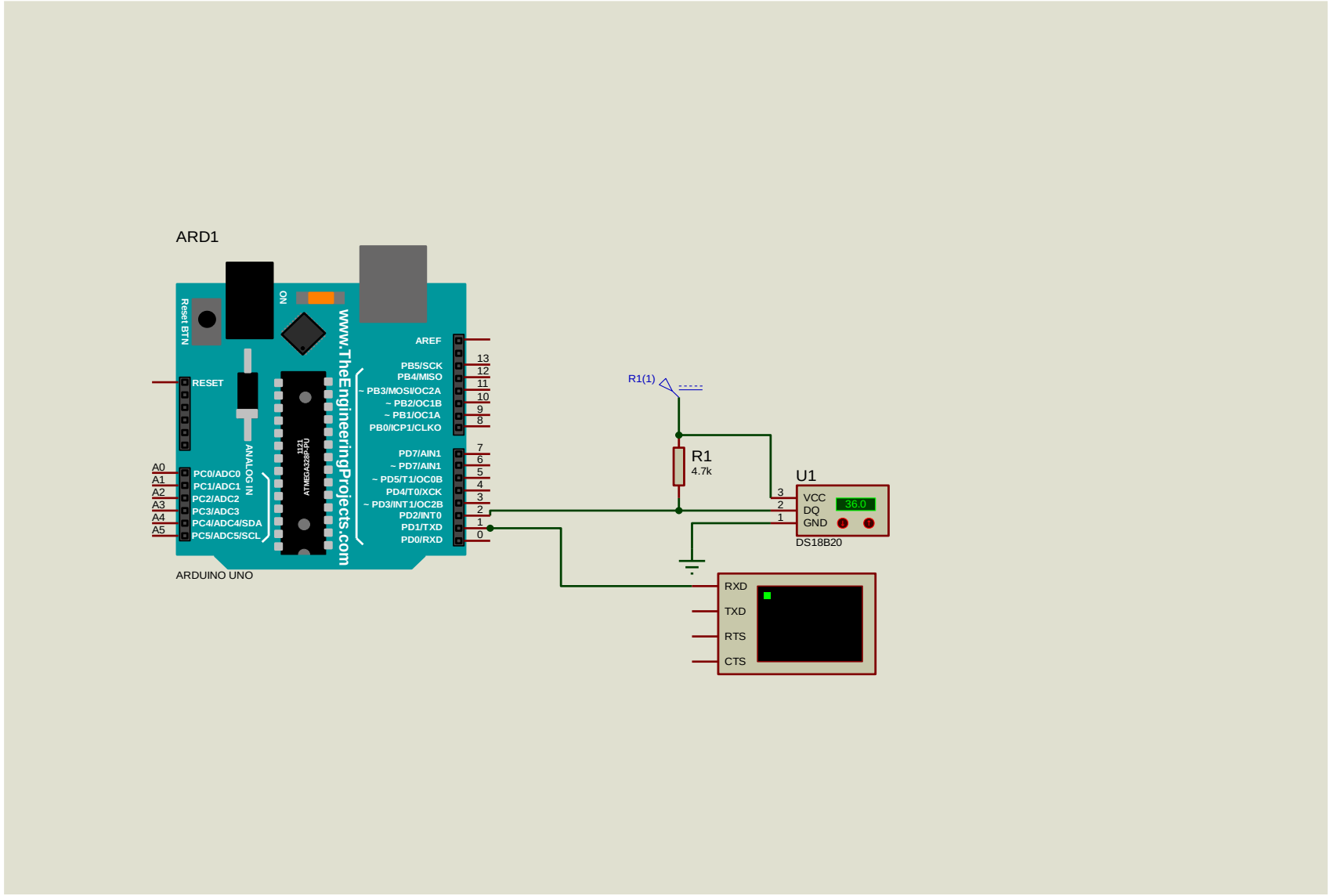
Capteur DS18B20 (sonde étanche)

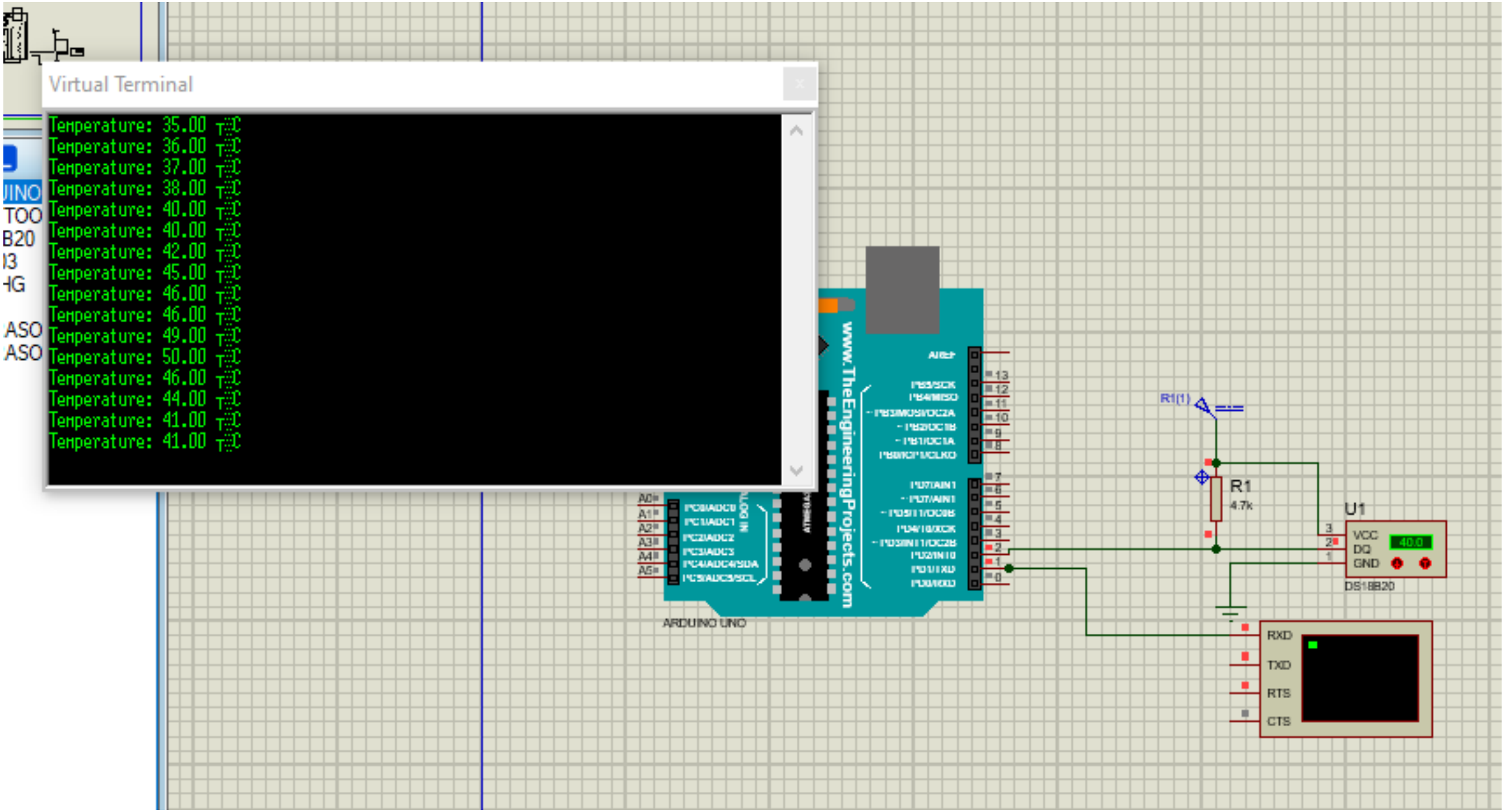
- **Précision** : Le DS18B20 offre une **précision élevée** dans la plage de température spécifiée (de **-55°C à +125°C**)

- **Étanchéité** : Il est conçu pour résister à l'humidité, ce qui est important lorsque les joueurs transpirent abondamment.

- **Facilité d'utilisation** : Son interface 1-wire est simple à connecter à des microcontrôleurs comme Arduino.

Schéma de simulation sur Proteus




Virtual Terminal
Temperature: 35.00 °C
Temperature: 36.00 °C
Temperature: 37.00 °C
Temperature: 38.00 °C
Temperature: 40.00 °C
Temperature: 40.00 °C
Temperature: 42.00 °C
Temperature: 45.00 °C
Temperature: 46.00 °C
Temperature: 46.00 °C
Temperature: 49.00 °C
Temperature: 50.00 °C
Temperature: 46.00 °C
Temperature: 44.00 °C
Temperature: 41.00 °C
Temperature: 41.00 °C
www.TheEngineeringProjects.com
ARDUINO UNO
R1
4.7k
U1
VCC
DQ
GND
40.0
DS18B20
RXD
TXD
RTS
CTS

Objectif 1 + Objectif 2

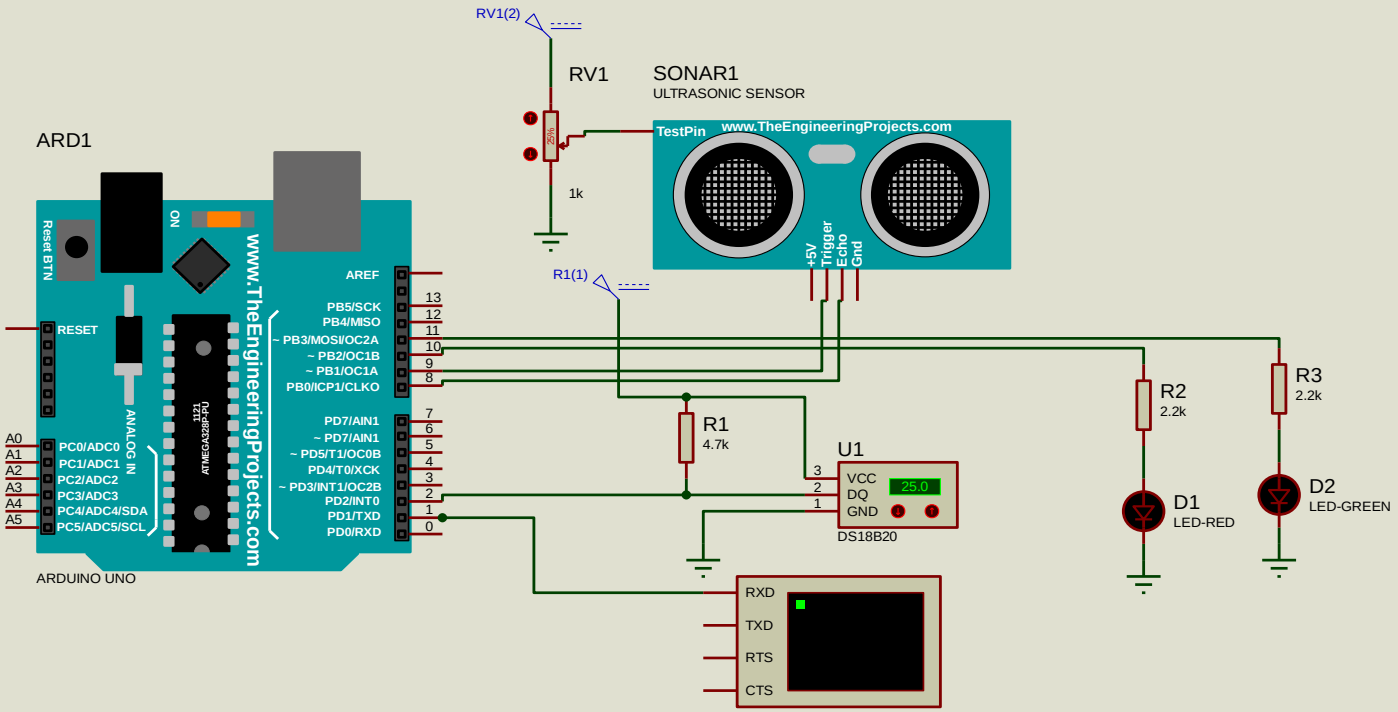
Joueur loin d'un autre avec une distance

Température intérieur du joueur

→ Calculer la température, la vitesse et la distance →

- Valeur de vitesse
- Valeur de température
- Valeur de distance / autre joueur

Schéma et simulation sur Proteus

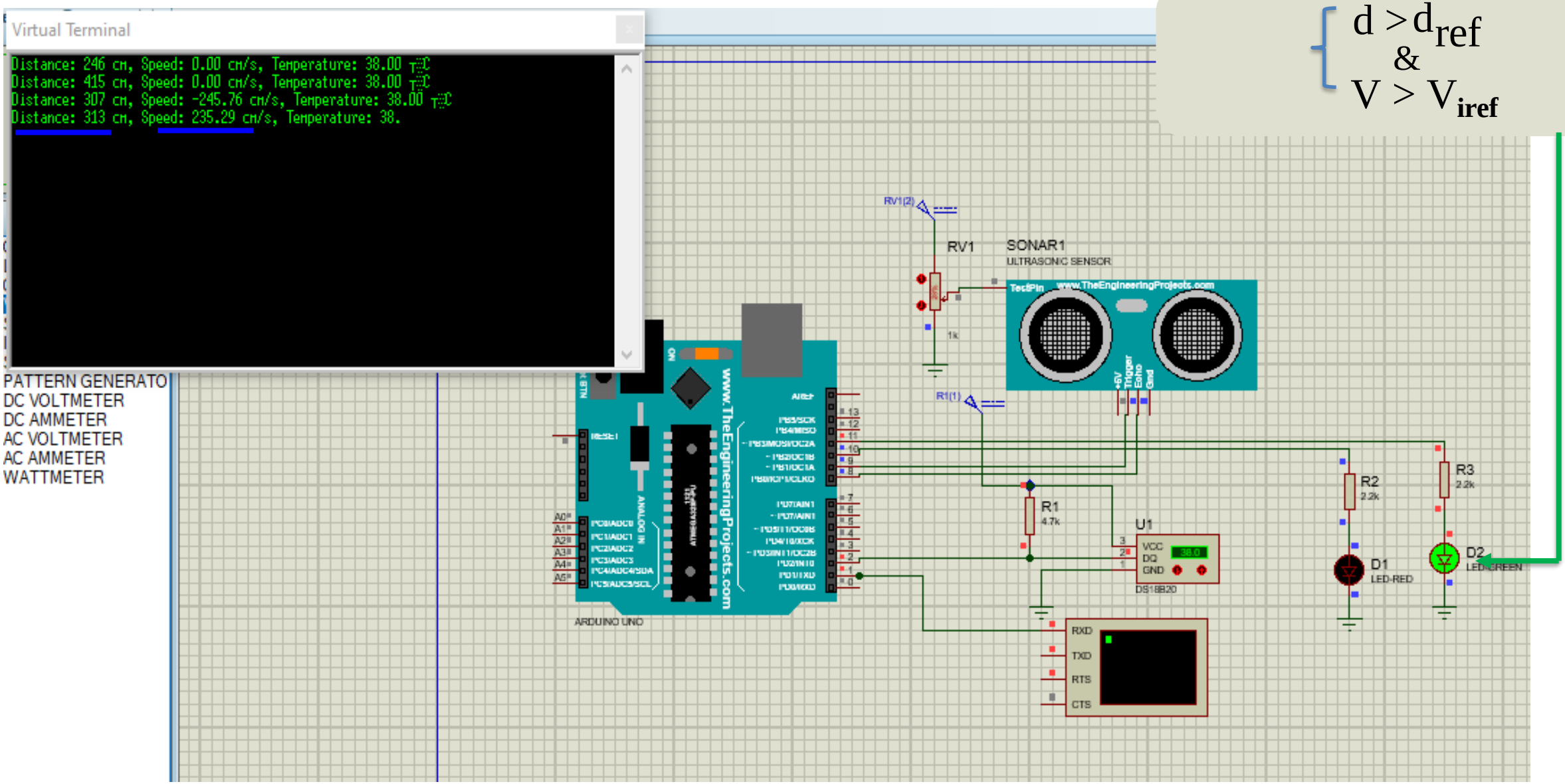


Hypothèse de simulation :

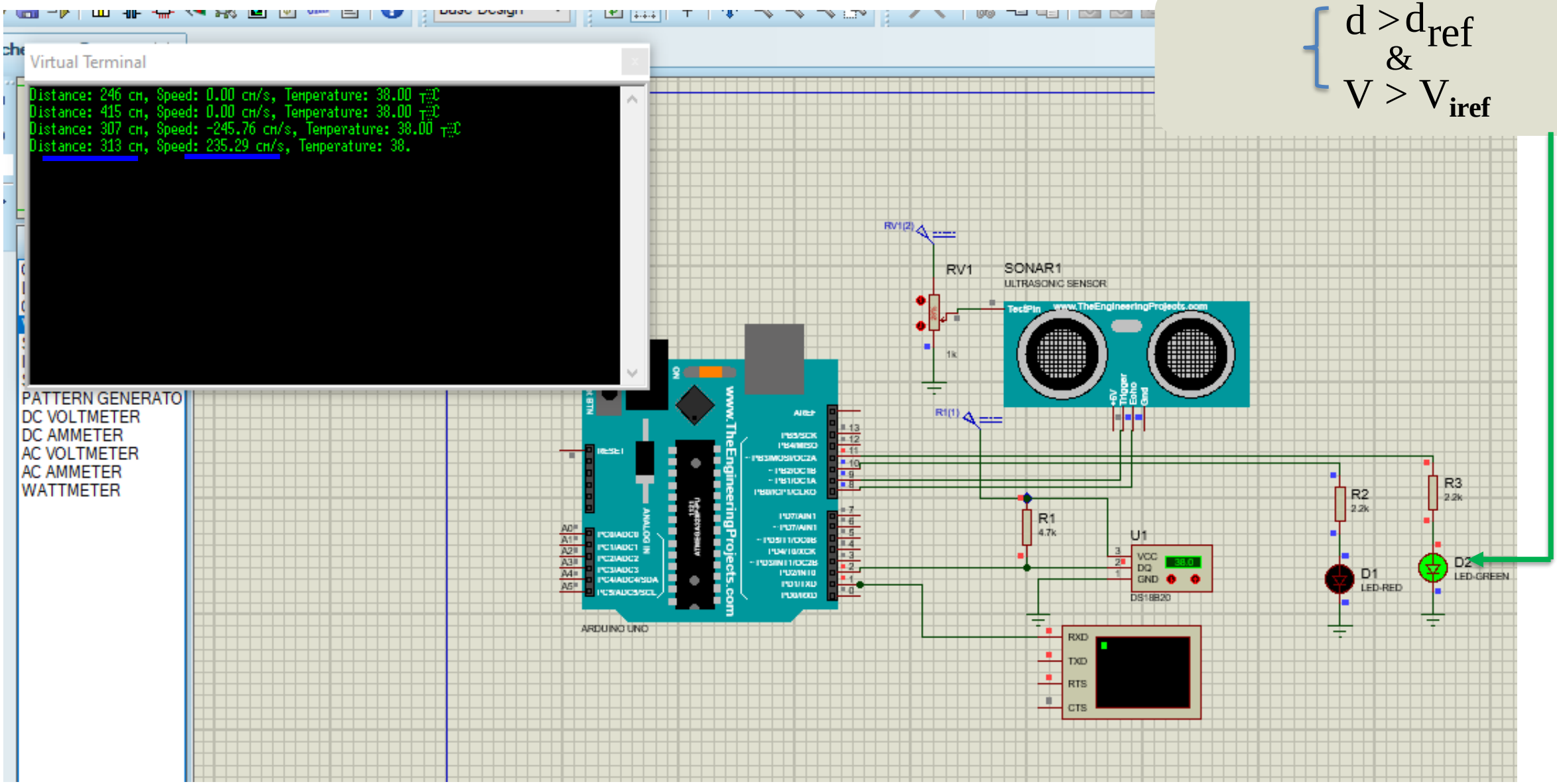
$d_{ref} = 200$ cm

$V_{iref} = 100$ cm/s

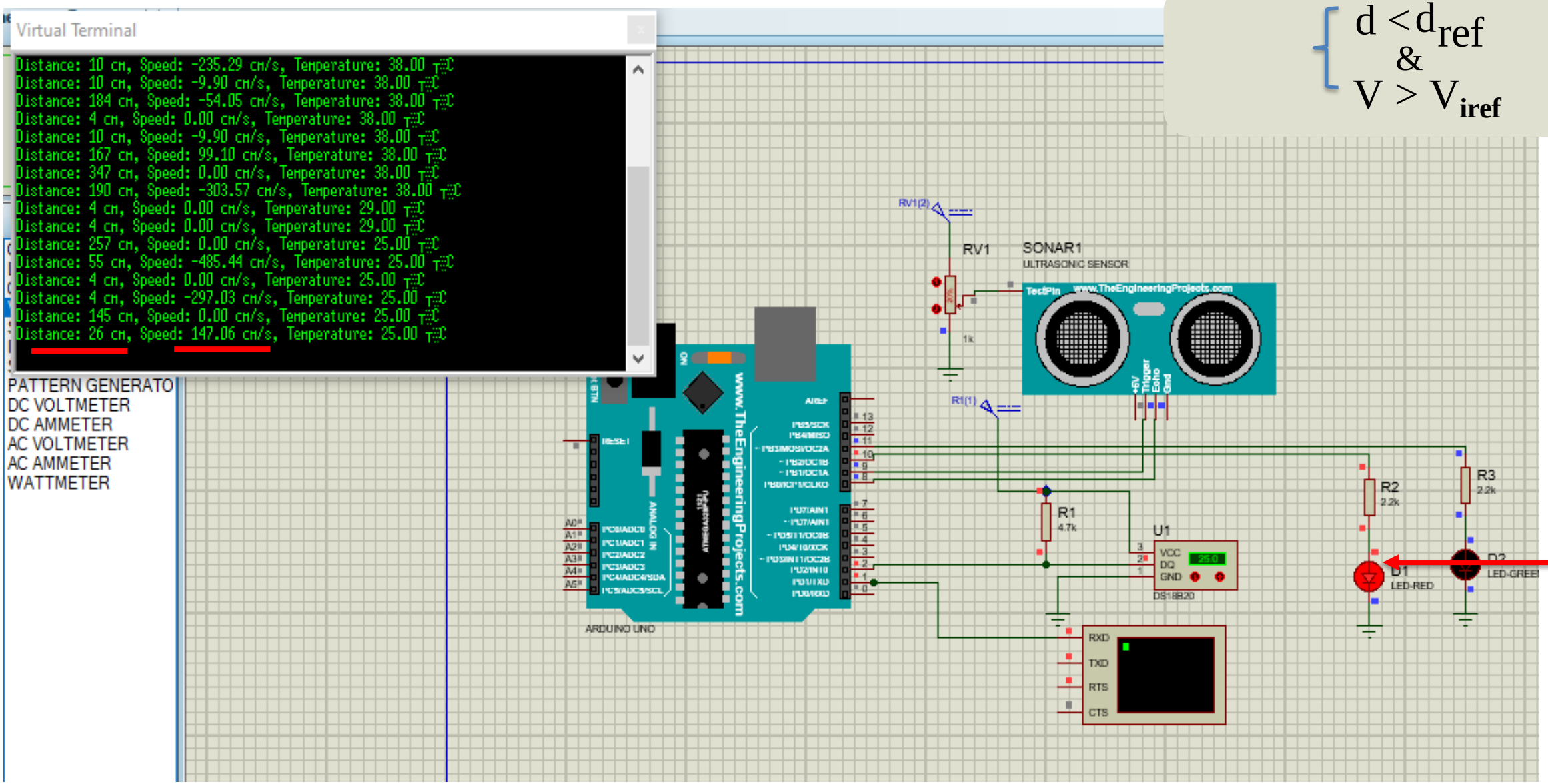
$$\begin{cases} d > d_{ref} \\ \& \\ V > V_{iref} \end{cases}$$



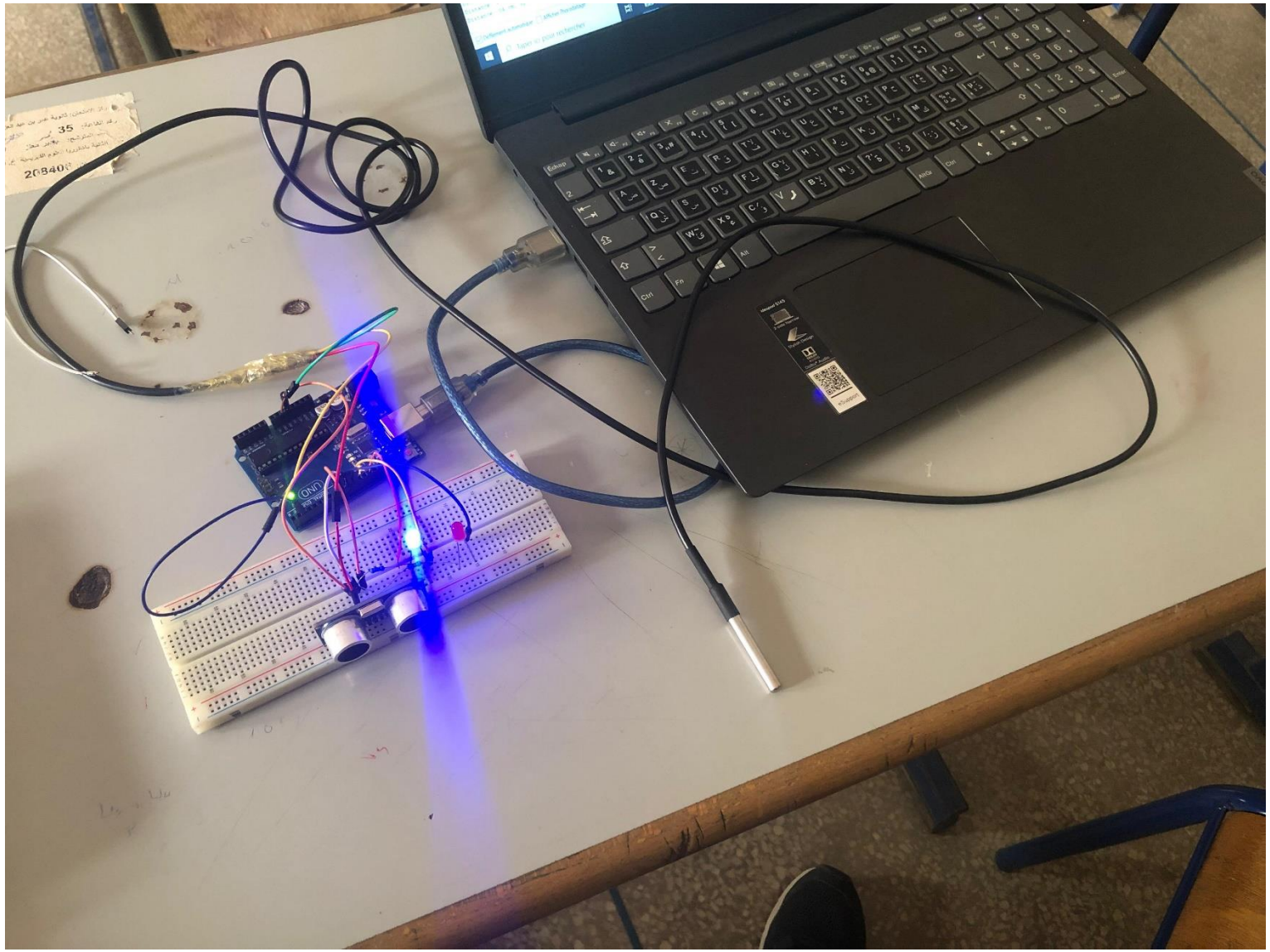
$$\begin{cases} d > d_{ref} \\ \& \\ V > V_{iref} \end{cases}$$



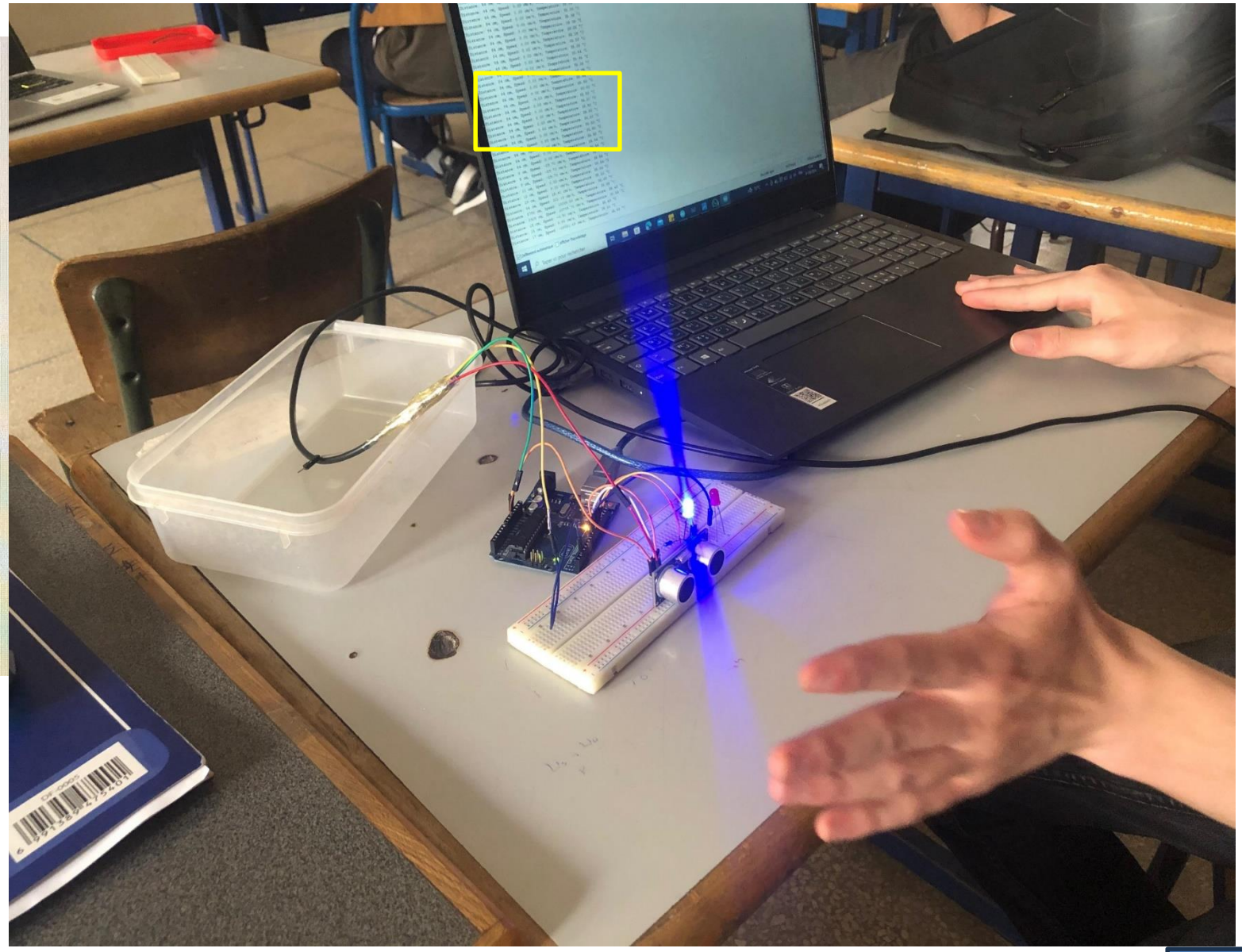
$$\begin{cases} d < d_{ref} \\ \& \\ V > V_{iref} \end{cases}$$



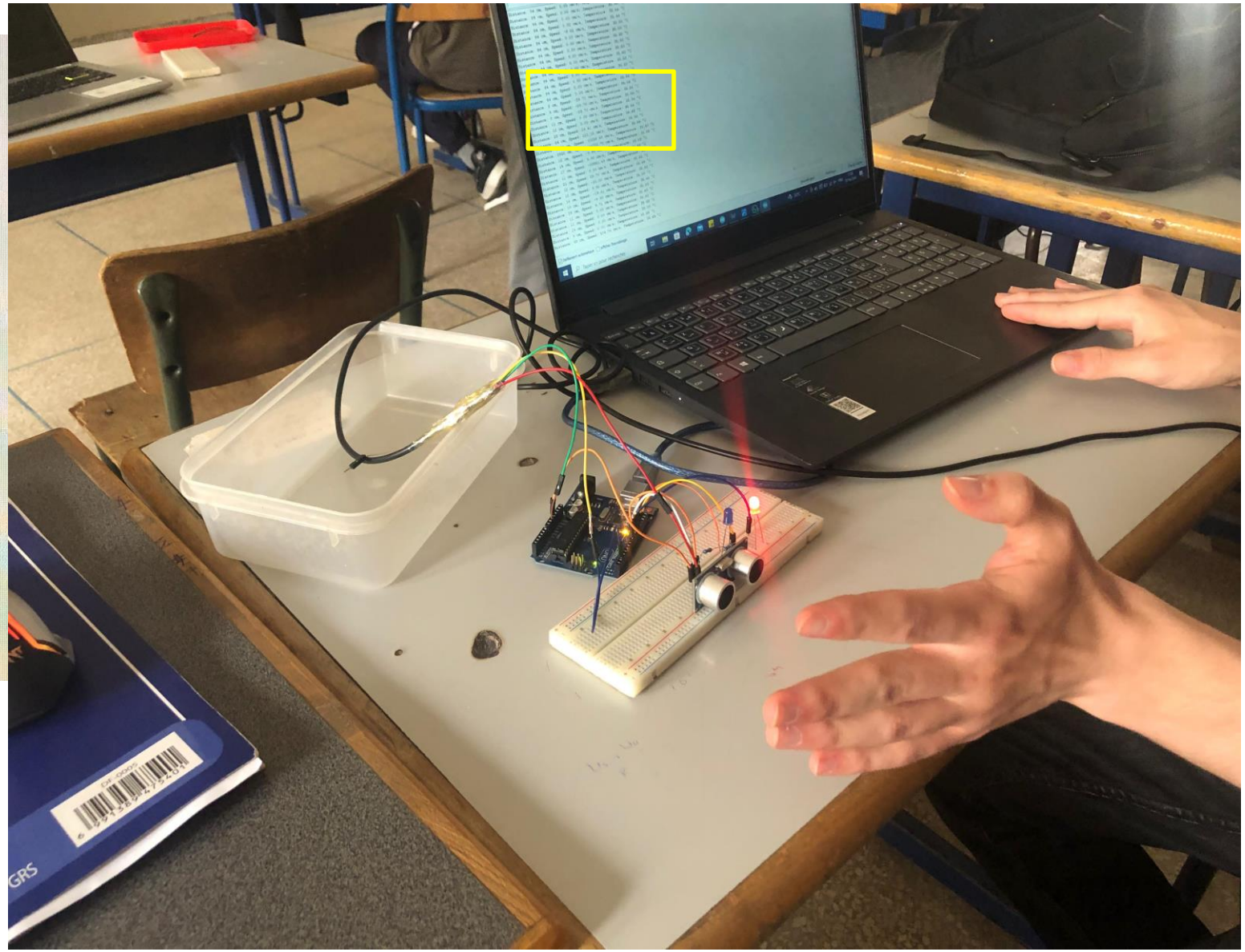
Expérience



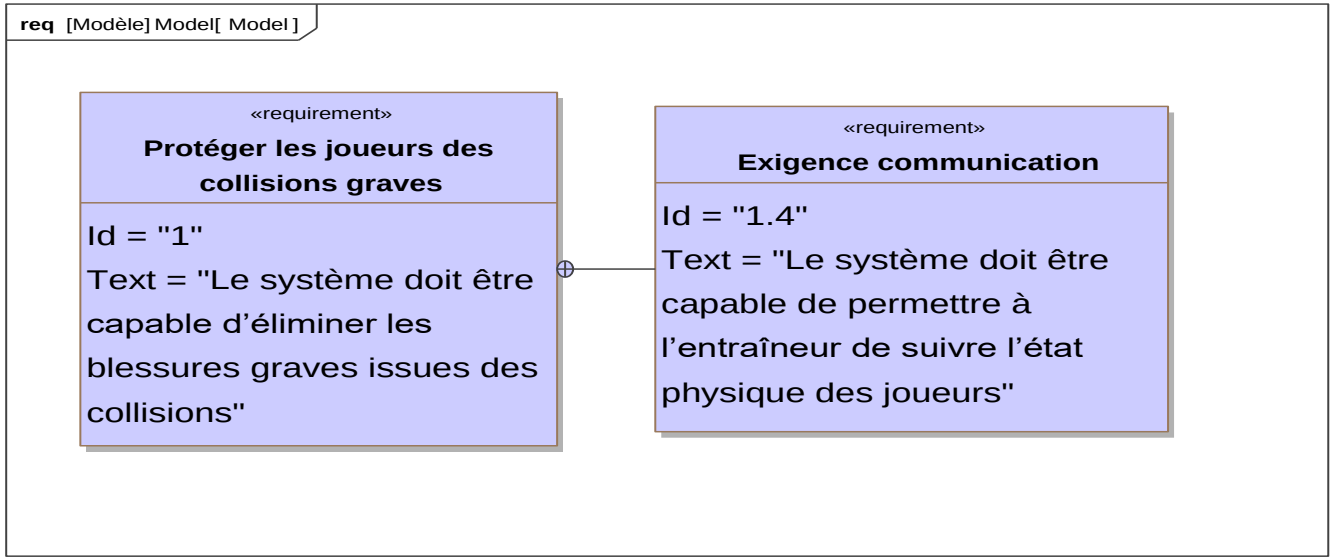
```
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.63 °C
Distance: 21 cm, Speed: 29.70 cm/s, Temperature: 35.69 °C
Distance: 12 cm, Speed: -20.00 cm/s, Temperature: 35.63 °C
Distance: 12 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 16 cm, Speed: -19.61 cm/s, Temperature: 35.63 °C
Distance: 10 cm, Speed: -9.80 cm/s, Temperature: 35.63 °C
Distance: 29 cm, Speed: 9.71 cm/s, Temperature: 35.69 °C
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.63 °C
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 29 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 9 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 85 cm, Speed: 504.76 cm/s, Temperature: 35.69 °C
Distance: 13 cm, Speed: 9.90 cm/s, Temperature: 35.63 °C
Distance: 12 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 55 cm, Speed: -278.85 cm/s, Temperature: 35.69 °C
Distance: 27 cm, Speed: -9.71 cm/s, Temperature: 35.69 °C
Distance: 27 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 29 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
```



```
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.63 °C
Distance: 21 cm, Speed: 29.70 cm/s, Temperature: 35.69 °C
Distance: 12 cm, Speed: -20.00 cm/s, Temperature: 35.63 °C
Distance: 12 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 16 cm, Speed: -19.61 cm/s, Temperature: 35.63 °C
Distance: 10 cm, Speed: -9.80 cm/s, Temperature: 35.63 °C
Distance: 29 cm, Speed: 9.71 cm/s, Temperature: 35.69 °C
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.63 °C
Distance: 11 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 29 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 9 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
Distance: 85 cm, Speed: 504.76 cm/s, Temperature: 35.69 °C
Distance: 13 cm, Speed: 9.90 cm/s, Temperature: 35.63 °C
Distance: 12 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 55 cm, Speed: -278.85 cm/s, Temperature: 35.69 °C
Distance: 27 cm, Speed: -9.71 cm/s, Temperature: 35.69 °C
Distance: 27 cm, Speed: 9.90 cm/s, Temperature: 35.69 °C
Distance: 29 cm, Speed: 0.00 cm/s, Temperature: 35.69 °C
```



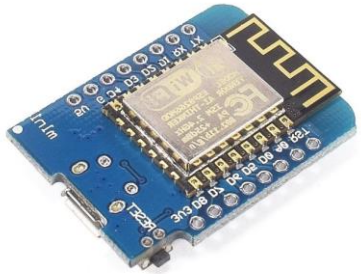
Objectif 3 : Assurer une communication avec l'entraineur



Comparaison entre module WIFI et module Bluetooth :

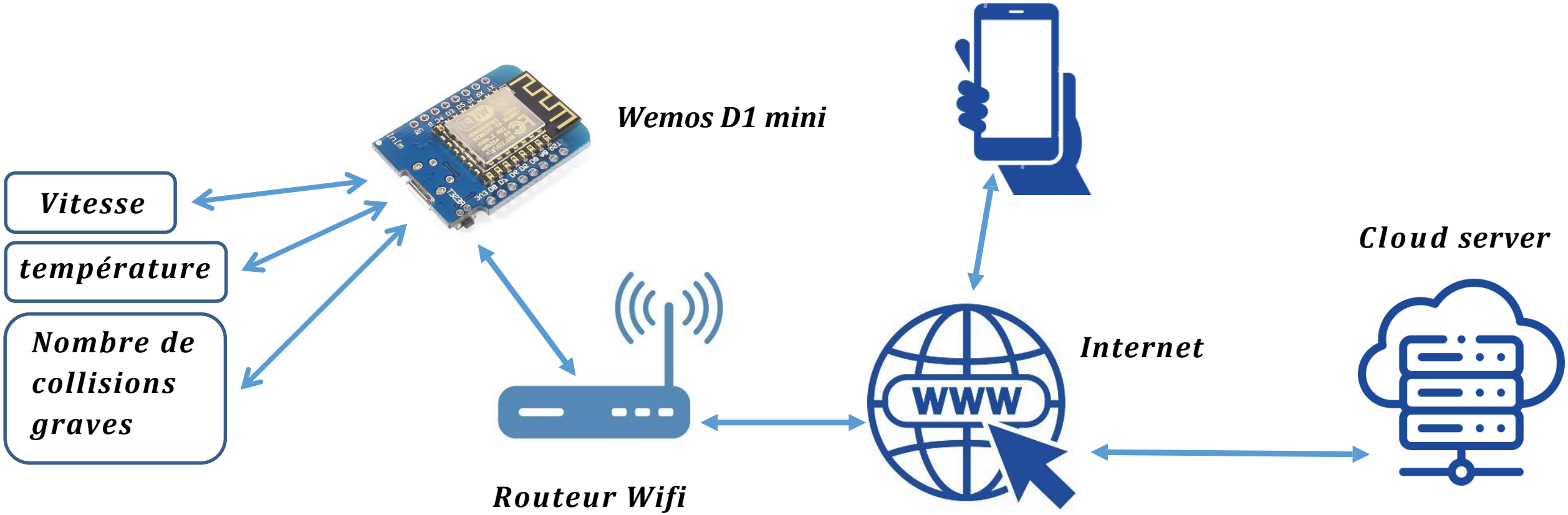
	Vitesse	Portée
Module WIFI	Module WIFI plus rapide que le module Bluetooth	Jusqu'à 100 m
Module Bluetooth		Environ 10 m

Wemos D1 mini :



- Contient un module WIFI EPS8266
- Compatible avec l'IDE Arduino
- Occupe moins d'espace

Principe de fonctionnement:



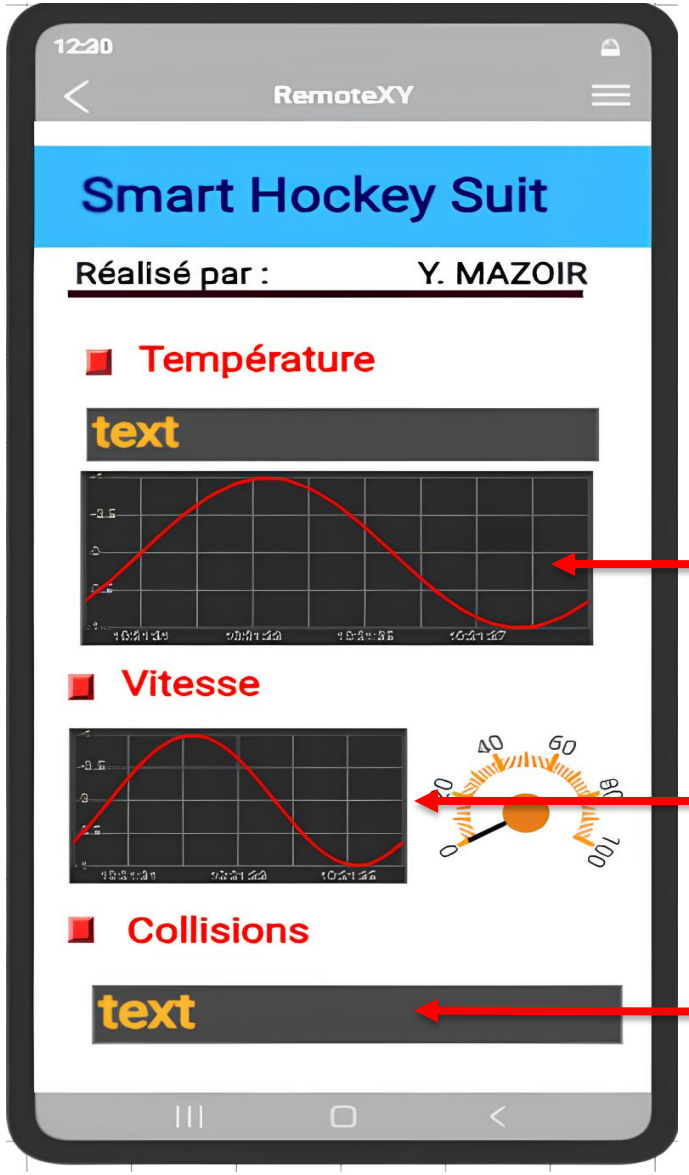
Configuration

Wi-Fi access point

WeMos D1 mini

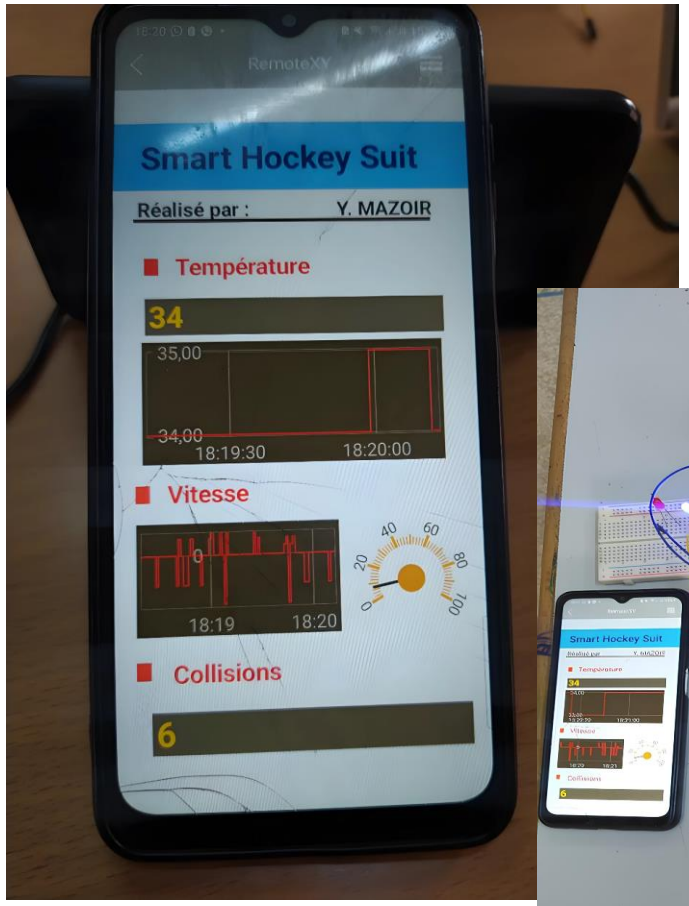
Integrated WiFi

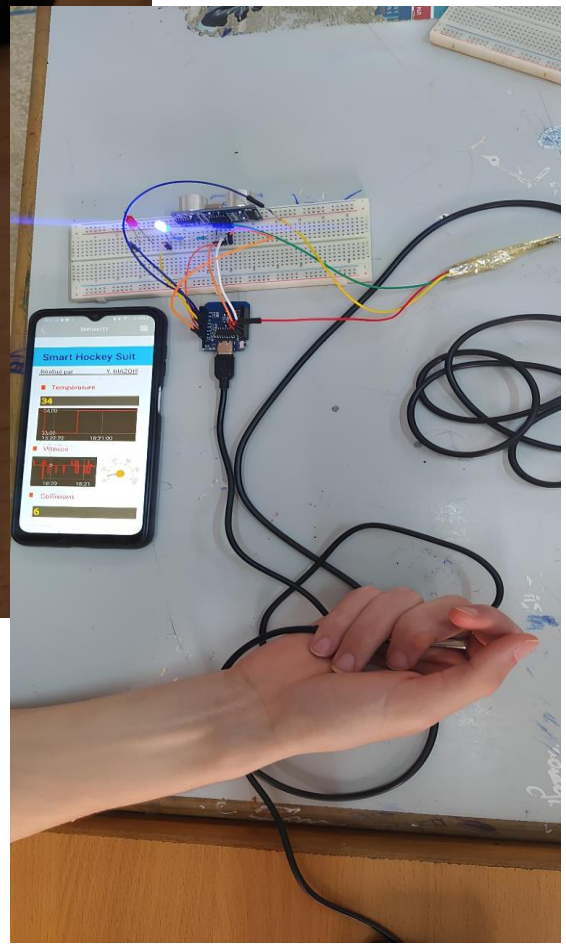
Arduino IDE



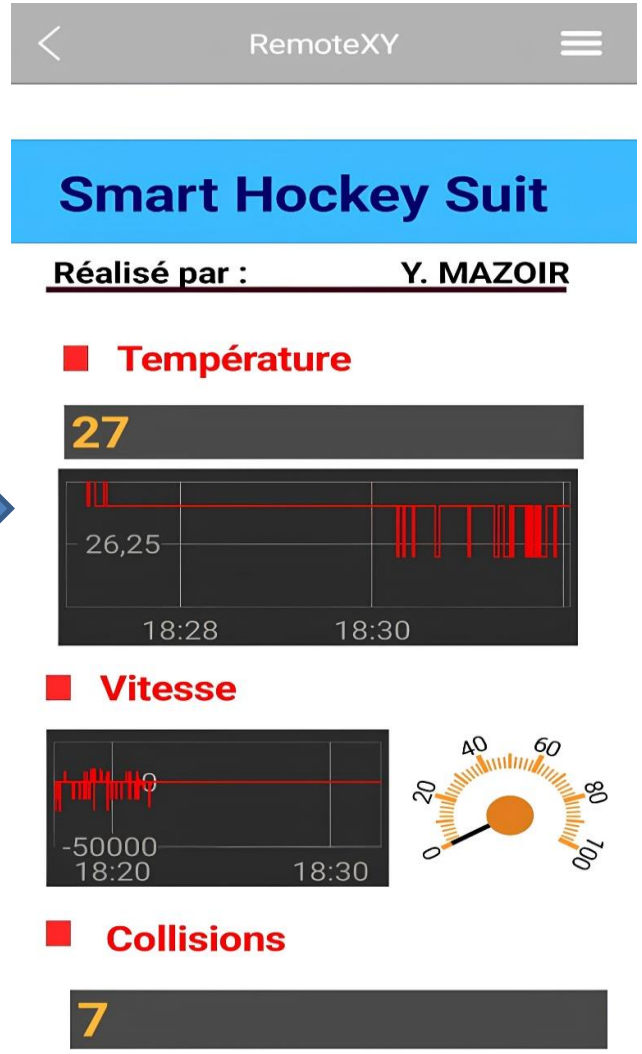
Courbes des variations de vitesse et de température durant un match

Nombre de collisions graves pendant un match

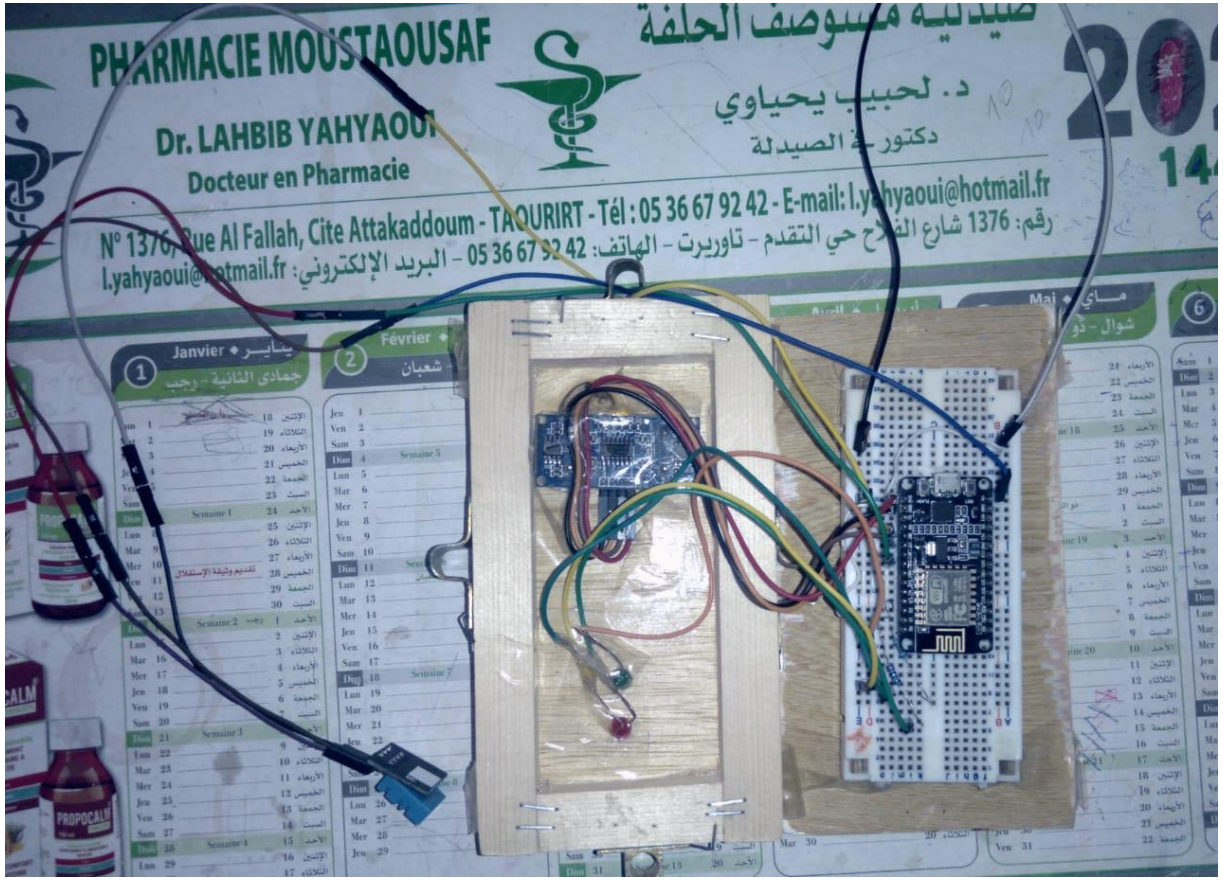


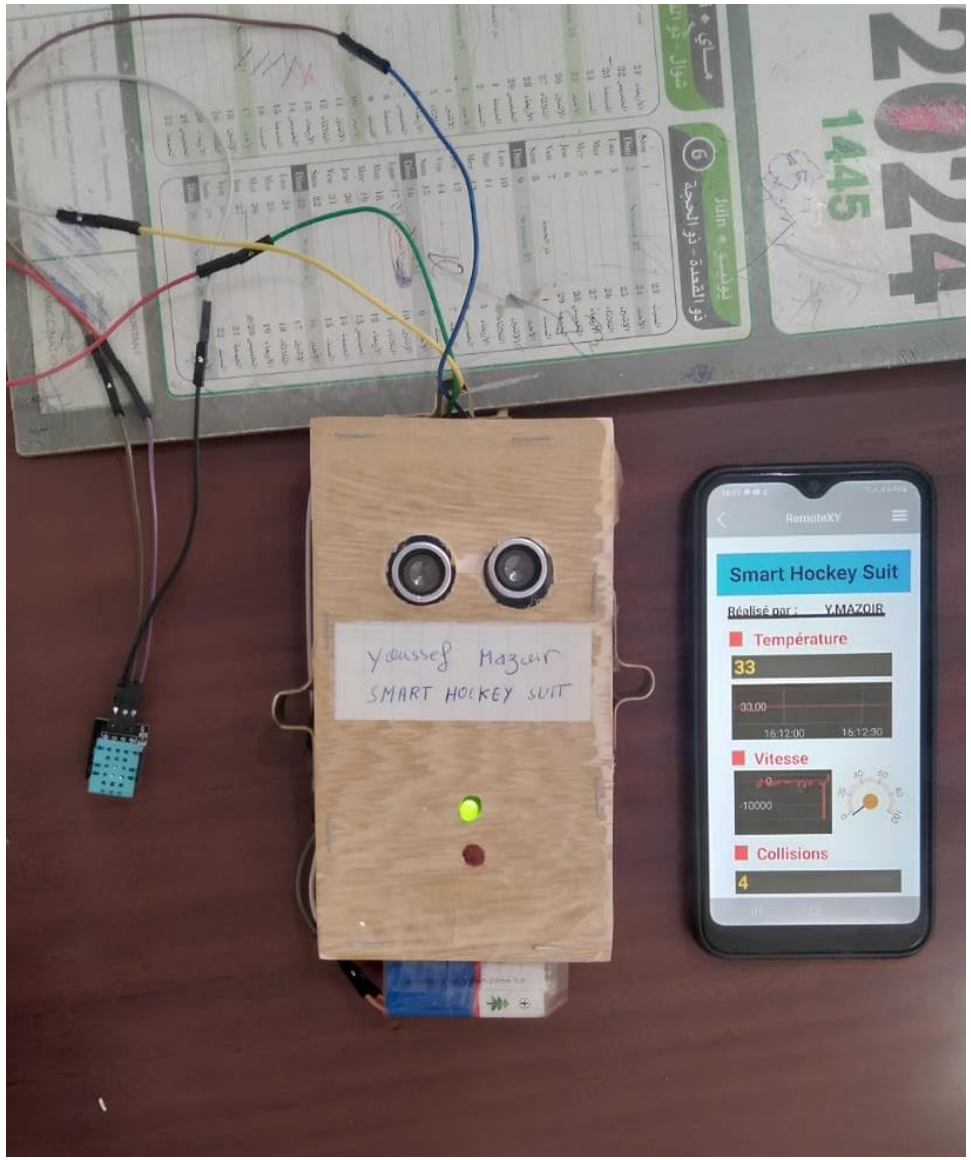


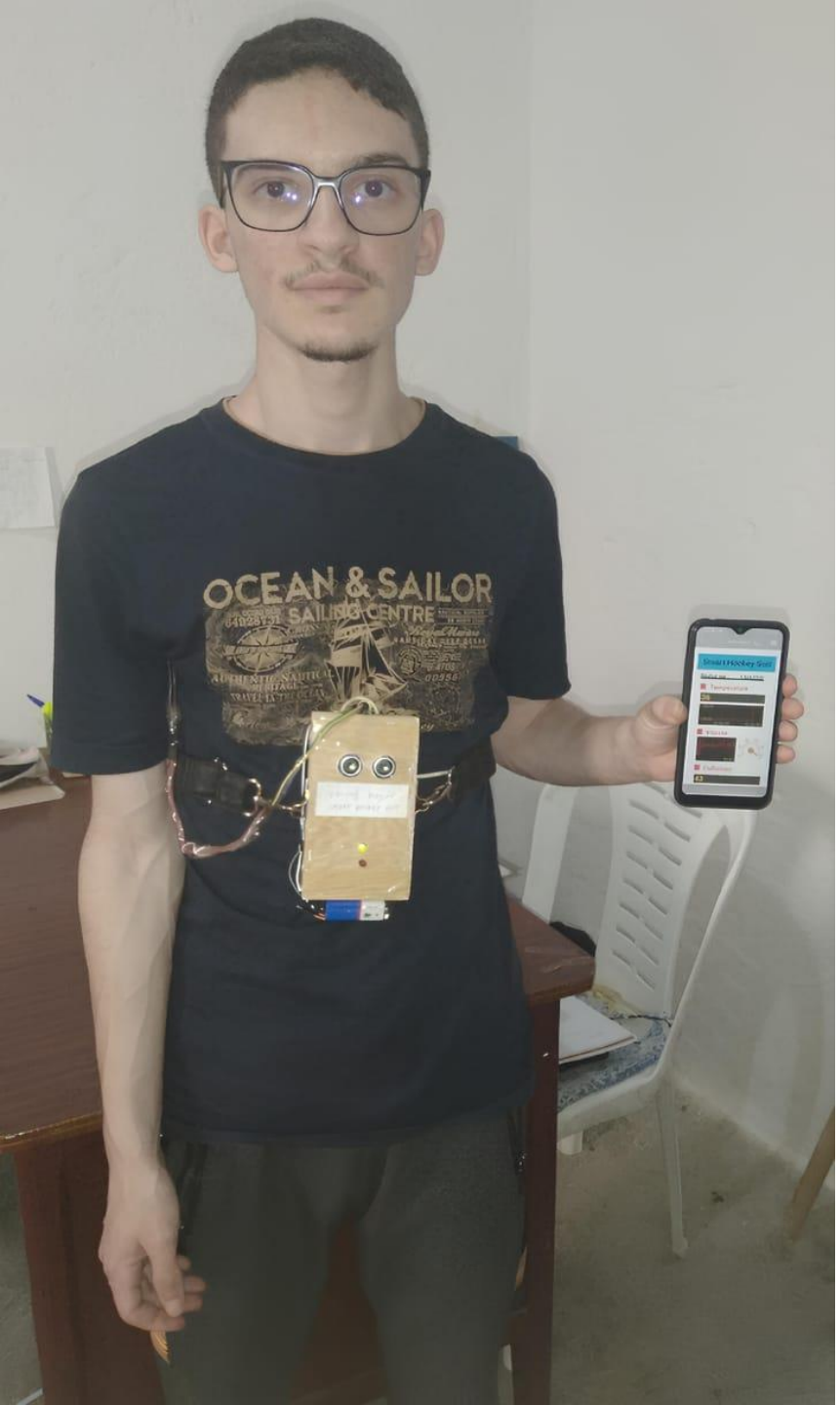
À la fin du match



Prototype:


PHARMACIE MOUSTAOUSAF
Dr. LAHBIB YAHYAOUI
Docteur en Pharmacie
د. لحبيب يحياوي
دكتورة الصيدلة


Youssef Mazouz
SMART HOCKEY SUIT
RemoteXY
Smart Hockey Suit
Réalisé par : Y.MAZOUZ
Température
33
Vitesse
Collisions
4



conclusion

MERCI POUR VOTRE ATTENTION

Références

Annexes

1. Épidémiologie des blessures dans le hockey professionnel sur glace : une étude prospective sur sept ans :
   Epidemiology of injuries in professional ice hockey: a prospective study over seven years | Journal of Experimental Orthopaedics | Full Text (springeropen.com)

2. Impacts de la température ambiante et des variations saisonnières sur les blessures sportives à Madrid :
   Impacts of ambient temperature and seasonal changes on sports injuries in Madrid, Spain: a time-series regression analysis (bmj.com)

3. La grandeur et le poids des joueurs de hockey :
   Hockey | Poids et grandeur | Tendances (yuccait.com)

Code de mesure de vitesse

mesure_vitesse | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_vitesse §

```
const int echoPin = 8;
const int pingPin = 9;
const int redLedPin = 10;
const int greenLedPin = 11;

long duration, cm1, cm2;
unsigned long time1, time2;
float speed;

void setup()
{
  Serial.begin(9600);
  pinMode(pingPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
}

void loop()
{
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm1 = microsecondsToCentimeters(duration);
  time1 = millis();
```

Compilation terminée.

```
Le croquis utilise 4540 octets (14%) de l'espa
Les variables globales utilisent 246 octets (1
```

20

mesure_vitesse | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_vitesse §

```
  delay(100);

  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm2 = microsecondsToCentimeters(duration);
  time2 = millis();

  speed = (float)(cm2 - cm1) / (float)(time2 - time1) * 1000;

  Serial.print("Distance: ");
  Serial.print(cm2);
  Serial.print(" cm, Speed: ");
  Serial.print(speed);
  Serial.print(" cm/s");
  Serial.println();

  if (cm2 < 200 && speed > 100) {
    digitalWrite(redLedPin, HIGH);
    digitalWrite(greenLedPin, LOW);
  } else {
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
  }
```

Compilation terminée.

```
Le croquis utilise 4540 octets (14%) de l'espace de stockage d
Les variables globales utilisent 246 octets (12%) de mémoire d
```

20

```
  delay(100);
}

long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

Compilation terminée.

```
Le croquis utilise 4540 octets (14%) de l'espace de
Les variables globales utilisent 246 octets (12%) d
```

20

Code de mesure de température

mesure_de_temperature | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_de_temperature

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int ds18b20Pin = 2;

// DS18B20 settings
OneWire oneWire(ds18b20Pin);
DallasTemperature sensors(&oneWire);

void setup()
{
  sensors.begin();
  Serial.begin(9600);
}

void loop()
{
  sensors.requestTemperatures();
  delay(500);
  float temp = sensors.getTempCByIndex(0);


  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" °C");


  delay(500);
}
```

Compilation terminée.

```
Le croquis utilise 5538 octets (17%) de l'espace de stockage de programmes. Le maximum est de 32256 octets.
Les variables globales utilisent 261 octets (12%) de mémoire dynamique, ce qui laisse 1787 octets pour les variables locales. Le maximum est de 2048 octets.
```


28 - 29

Code de mesure de vitesse & température

mesure_vitesse_temp | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_vitesse_temp §

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int echoPin = 8;
const int pingPin = 9;
const int ds18b20Pin = 2;
const int redLedPin = 10;
const int greenLedPin = 11;

long duration, cm1, cm2;
unsigned long time1, time2;
float speed;


OneWire oneWire(ds18b20Pin);
DallasTemperature sensors(&oneWire);
void setup()
{
  sensors.begin();
  Serial.begin(9600);
  pinMode(pingPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
}

void loop()
```

Enregistrement terminé.

```
Le nom de croquis a dû être modifié.
Les noms de croquis doivent commencer pa
chiffres, tirets, points et traits de so
```

81

mesure_vitesse_temp | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_vitesse_temp §

```
void loop()
{
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm1 = microsecondsToCentimeters(duration);
  time1 = millis();

  delay(100);

  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm2 = microsecondsToCentimeters(duration);
  time2 = millis();
```

Enregistrement terminé.

```
Le nom de croquis a dû être modifié.
Les noms de croquis doivent commencer par une
chiffres, tirets, points et traits de soulign
```

81

mesure_vitesse_temp | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

mesure_vitesse_temp §

```
  time2 = millis();

  speed = (float)(cm2 - cm1) / (float)(time2 - tim

  sensors.requestTemperatures();
  delay(500);
  float temp = sensors.getTempCByIndex(0);

  Serial.print("Distance: ");
  Serial.print(cm2);
  Serial.print(" cm, Speed: ");
  Serial.print(speed);
  Serial.print(" cm/s, Temperature: ");
  Serial.print(temp);
  Serial.println(" °C");

  if (cm2 < 200 && speed > 100) {
    digitalWrite(redLedPin, HIGH);
    digitalWrite(greenLedPin, LOW);
  } else {
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
  }

  delay(500);
}

long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

Compilation terminée.

```
Le croquis utilise 6746 octets (20%) de l'espace de
Les variables globales utilisent 309 octets (15%) d
```

51

application | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

application

```
#define REMOTEXY_MODE__WIFI_POINT

#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define REMOTEXY_WIFI_SSID "Hockey sur glace"
#define REMOTEXY_WIFI_PASSWORD "12345678"
#define REMOTEXY_SERVER_PORT 6377

#include <RemoteXY.h>

#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
  { 255,0,0,34,0,230,0,17,0,0,0,31,1,106,200,1,1,16,0,130,
  244,5,135,21,0,178,129,8,11,85,10,201,83,109,97,114,116,32,72,111,
  99,107,101,121,32,83,117,105,116,32,0,129,7,29,34,6,24,82,195,169
```

Compilation terminée.

```
. Variables and constants in RAM (global, static), used 28420 / 80192 bytes (35%)
|   SEGMENT  BYTES    DESCRIPTION
|-- DATA     1496     initialized variables
|-- RODATA   1004     constants
`-- BSS      25920    zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 60363 / 65536 bytes (92%)
|   SEGMENT  BYTES    DESCRIPTION
|-- ICACHE   32768    reserved space for flash instruction cache
`-- IRAM     27595    code in IRAM
. Code in flash (default, ICACHE_FLASH_ATTR), used 255220 / 1048576 bytes (24%)
|   SEGMENT  BYTES    DESCRIPTION
`-- IROM     255220   code in flash
```

R2 & mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32K

application | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

application

```
#define REMOTEXY_MODE__WIFI_POINT

#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define REMOTEXY_WIFI_SSID "Hockey sur glace"
#define REMOTEXY_WIFI_PASSWORD "12345678"
#define REMOTEXY_SERVER_PORT 6377

#include <RemoteXY.h>

#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
  { 255,0,0,34,0,230,0,17,0,0,0,31,1,106,200,1,1,16,0,130,
  244,5,135,21,0,178,129,8,11,85,10,201,83,109,97,114,116,32,72,111,
  99,107,101,121,32,83,117,105,116,32,0,129,7,29,34,6,24,82,195,169,
  97,108,105,115,195,169,32,112,97,114,32,58,32,0,129,18,46,42,7,36,
  84,101,109,112,195,169,114,97,116,117,114,101,32,0,130,6,35,88,1,0,
  242,130,9,47,4,5,0,37,129,66,29,29,6,24,89,46,32,77,65,90,
  79,73,82,0,67,9,59,88,11,4,2,26,11,68,8,72,88,36,1,8,
  36,129,15,113,25,7,36,86,105,116,101,115,115,101,32,0,130,6,114,4,
  5,0,37,71,67,125,35,35,56,0,2,24,135,0,0,0,0,0,0,200,
  66,0,0,160,65,0,0,32,65,0,0,0,64,24,0,68,6,125,58,32,
  1,8,36,129,16,164,33,7,36,67,111,108,108,105,115,105,111,110,115,32,
  0,130,6,164,4,5,0,37,67,10,178,91,12,4,2,26,11 };

struct {

  char text_01[11];
  float onlineGraph_01_var1;
```

Compilation terminée.

```
. Code in flash (default, ICACHE_FLASH_ATTR), used 255220 / 1048576 byt
|   SEGMENT  BYTES    DESCRIPTION
`-- IROM     255220   code in flash
```

R2 & mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32

application | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

application

```
struct {

  char text_01[11];
  float onlineGraph_01_var1;
  float instrument_01;
  float onlineGraph_02_var1;
  char text_02[11];

    // other variable
  uint8_t connect_flag;

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//           END RemoteXY include          //
/////////////////////////////////////////////

const int echoPin = D8;
const int pingPin = D7;
const int ds18b20Pin = D2;
const int redLedPin = D1;
const int greenLedPin = D3;

long duration, cm1, cm2;
unsigned long time1, time2;
float speed;

OneWire oneWire(ds18b20Pin);
DallasTemperature sensors(&oneWire);
```

Compilation terminée.

```
. Code in flash (default, ICACHE_FLASH_ATTR),
‖   SEGMENT  BYTES    DESCRIPTION
└── IROM     255220   code in flash
```

R28& mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, A

application | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

application

```
int c=0;
void setup()
{
  RemoteXY_Init ();

  sensors.begin();
  Serial.begin(9600);
  pinMode(pingPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);

}

void loop()
{
  RemoteXY_Handler ();

  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm1 = microsecondsToCentimeters(duration);
  time1 = millis();

  delay(100);
```

Compilation terminée.

```
. Code in flash (default, ICACHE_FLASH_ATTR),
‖   SEGMENT  BYTES    DESCRIPTION
└── IROM     255220   code in flash
```

R28& mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, A

application | Arduino 1.8.19

Fichier Édition Croquis Outils Aide

application

```
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  cm2 = microsecondsToCentimeters(duration);
  time2 = millis();

  speed = (float)(cm2 - cm1) / (float)(time2 - time1) * 1000;
  RemoteXY.instrument_01 = speed;
  RemoteXY.onlineGraph_02_var1 = speed;


  sensors.requestTemperatures();
  delay(500);
  float temp = sensors.getTempCByIndex(0);
  int Te=temp;
  sprintf (RemoteXY.text_01, "%d", Te);
  RemoteXY.onlineGraph_01_var1 = Te;


  Serial.print("Distance: ");
  Serial.print(cm2);
  Serial.print(" cm, Speed: ");
  Serial.print(speed);
  Serial.print(" cm/s, Temperature: ");
  Serial.print(temp);
```

Compilation terminée.

```
. Code in flash (default, ICACHE_FLASH_ATTR), used 255220 / 10
‖   SEGMENT  BYTES    DESCRIPTION
└── IROM     255220   code in flash
```

R28& mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most co

```
  Serial.print(temp);
  Serial.println(" °C");

  if (cm2 < 200 && speed > 100) {
    digitalWrite(redLedPin, HIGH);
    digitalWrite(greenLedPin, LOW);
    c++;
    sprintf (RemoteXY.text_02, "%d", c);
  } else {
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
  }

  delay(100);

}

long microsecondsToCentimeters(long microsec
{
  return microseconds / 29 / 2;
}
```

Compilation terminée.

```
. Code in flash (default, ICACHE_FLASH_ATTR),
    SEGMENT  BYTES    DESCRIPTION
└─── IROM     255220   code in flash
```

R296 mini, 80 MHz, Flash, Disabled (new aborts on oom), Disabled,