المدرسة المحمدية للمهندسين
Ecole Mohammadia d'Ingénieurs

TRAVAUX D'INITIATIVE
PERSONNELLE ENCADRÉS T.I.P.E.
2023

Royaume du Maroc
Ministère de l'Education Nationale,
du Préscolaire et des Sports

LA VILLE

Sujet :

Smart-Trottinette

préparé par :

AMRANI Hamza

PLAN

Introduction

Analyse et solution

- Diagramme Sys ML
- Choix de moteur
- Asservissement et réglage de la vitesse
- Mesure les grandeur Vitesse, Poids , Lumière et la Température
- Communication et Application Mobile

Conclusion

Introduction

Les trottinettes électriques sont maintenant très populaires en raison du coût élevé du carburant, en plus raison de leur facilité d'utilisation et le respect de l'environnement.

Le Smart-Trottinette est tout à fait en ligne avec le thème de l'année (la ville). Car il garantit la mobilité des utilisateurs dans la ville afin d'offrir une bonne expérience en toute sécurité. Cela réduit les embouteillages et la pollution atmosphérique en ville.

Introduction

1 Problématique

- Comment garantir la sécurité des utilisateurs de Smart-Trottinette tout en améliorant leur expérience de mobilité ?
- Comment favoriser l'adoption des Smart-Trottinettes en tant que mode de transport durable et respectueux de l'environnement, tout en garantissant la qualité de l'expérience pour les utilisateurs ?

2 Objectifs

1. Asservissement et réglage de la vitesse selon le degré de la ponte

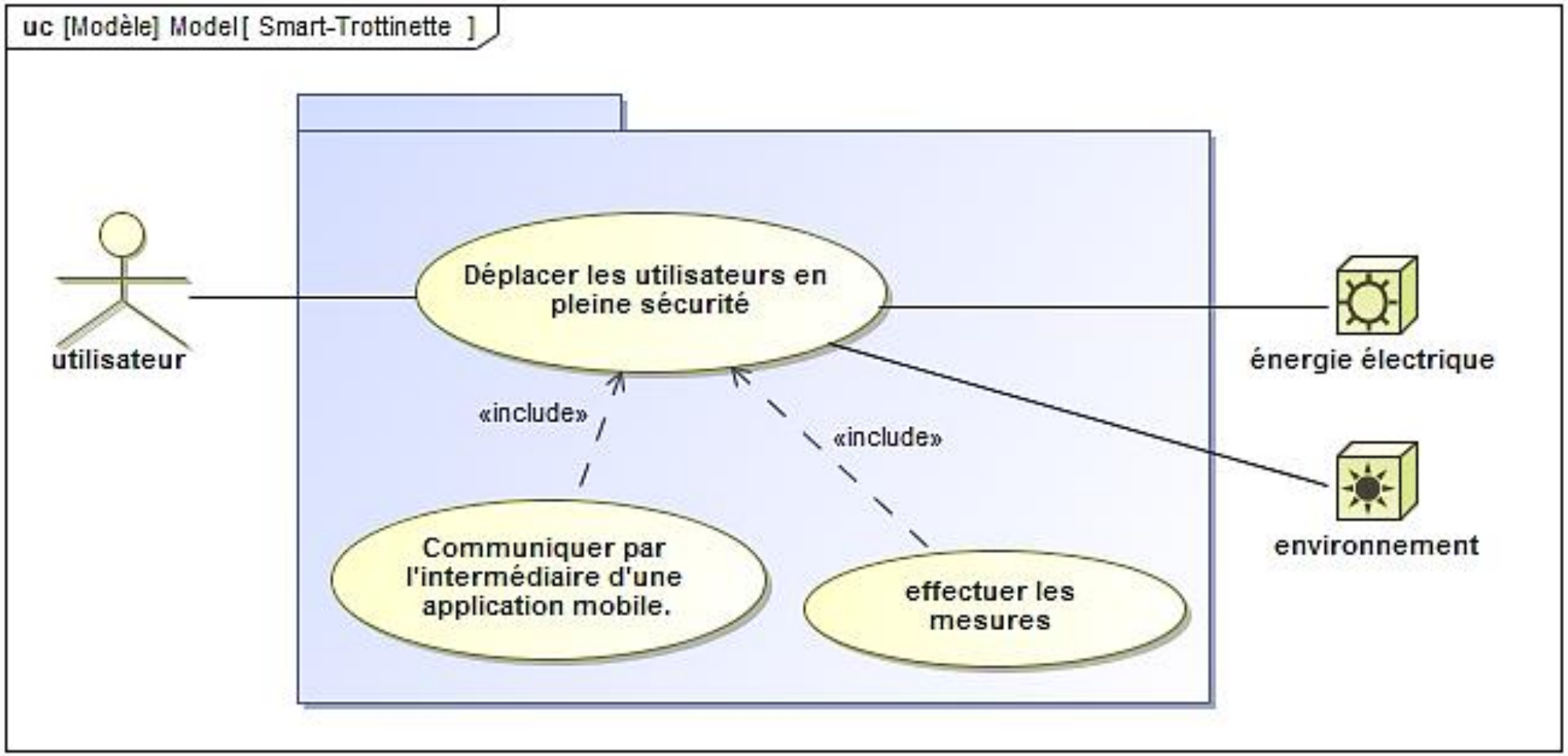
2. Mesure les grandeur vitesse, poids , lumière et la température

3. Localisation du smart-trottinette en cas de perte.

4. Création d'une application mobile qui gère les différents organes du Smart-Trottinette

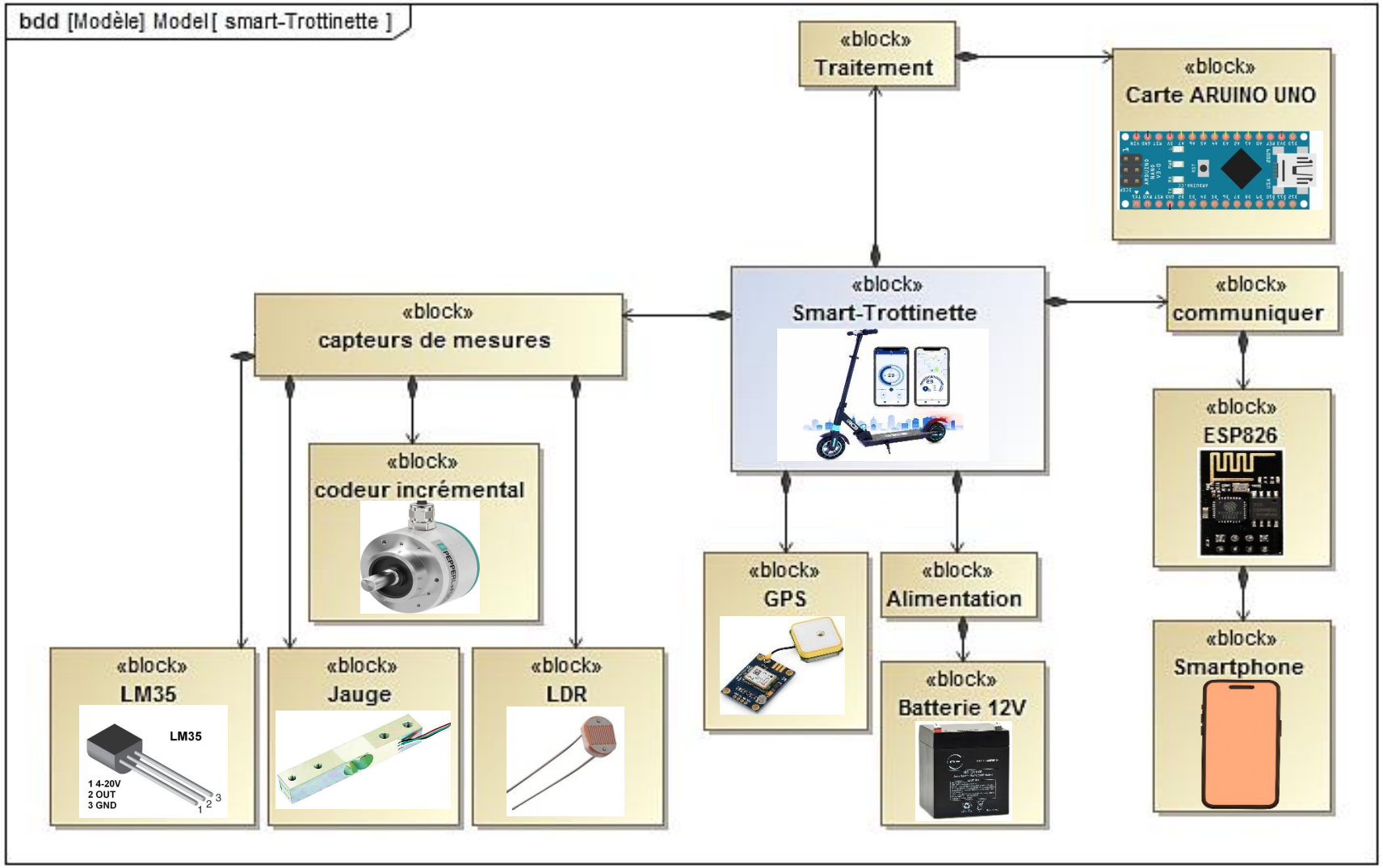
Présentation fonctionnelle du système

1 Diagramme Sys ML : Cas d'utilisation



Présentation fonctionnelle du système

3 Diagramme Sys ML : BDD



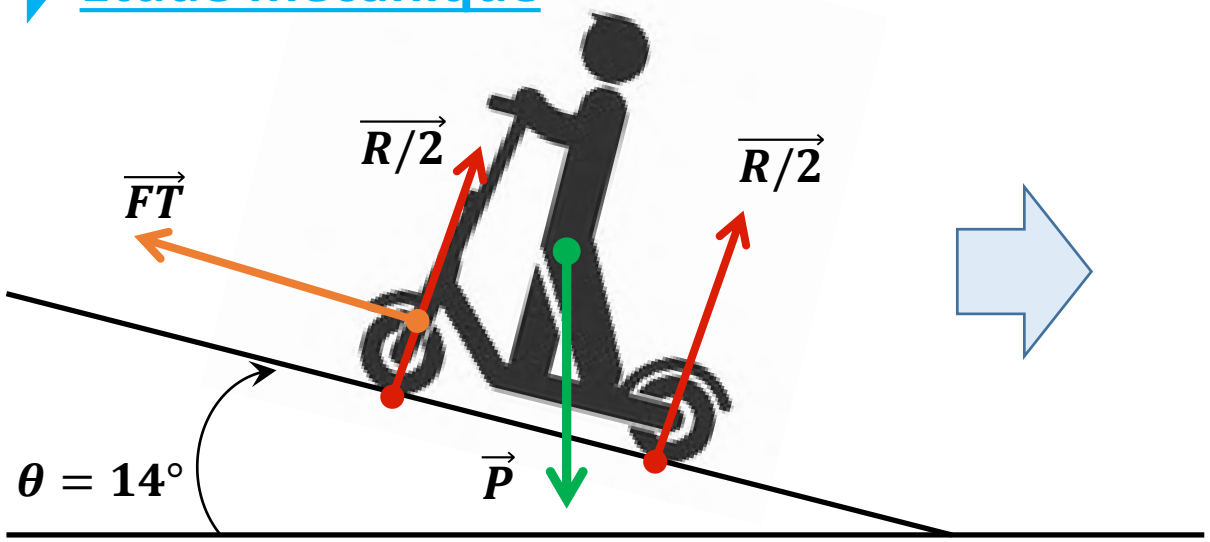
Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Les spécifications nécessaires

La masse portée	70 kg
La vitesse maximale	25 km/h
Diamètre des roues	21.6 cm
La ponte maximale grimpée	14°

Étude mécanique



❑ **Bilan de travail des forces**

- $W(\vec{p}) = -m.g.\sin(\theta).d$
- $W(\overrightarrow{FT}) = FT.d$
- $W(\vec{R}) = 0$

❑ **Théorème de l'énergie cinétique**

$\Delta Ec = \sum W(\overrightarrow{Fex})$ et $Ec(t) = \frac{1}{2}\,\mathrm{m}\,v(t)^2$

Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Calcul de la force de traction FT

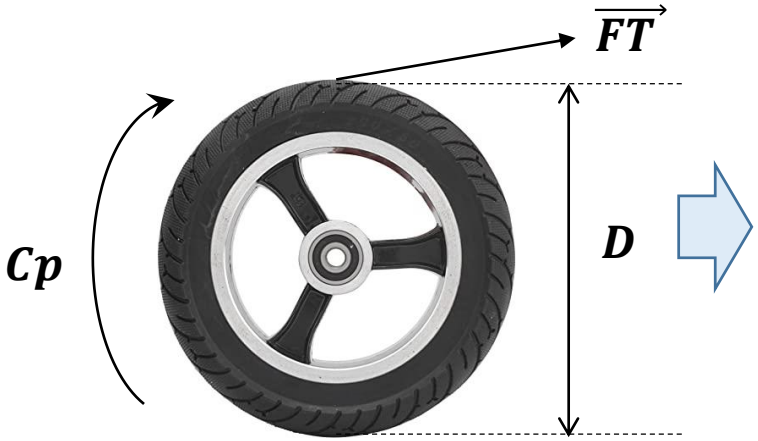
On a : $\Delta Ec = \sum W(\overrightarrow{Fex})$

$$\Rightarrow \frac{1}{2} \text{m} (vf^2 - vi^2) = FT.d - m.g.\sin(\theta).d = 0$$

Donc la force de traction est :

$$FT = m.g.\sin(\theta) \Rightarrow FT = 167\ N$$

Calcul de couple à l'arbre de la roue Cp



$$Cp = FT.\frac{D}{2}$$

$$Cp = 7.56\ Nm$$

Choix de moteur

machine à courant continu MY1020 de caractéristiques suivantes :

Type	Speed (tr/mn)	I (A)	Speed (tr/mn)	I (A)	Torque (Nm)	Power (W)	K (N.m/A)	R (Ω)
	No load		Load rate					
24V	3150	1	2500	26.7	1.9	500	0.072	0.23
36V	3150	1	2500	17.8	1.9	500	0.11	0.49

- Puissance nominale : Pm=500W
- Vitesse nominale : Nn=2500 tr/min
- moment d'inertie : Jm = 0,0071 kgm²
- Coefficient de couple : 0.072 Nm/A
- Résistance d'induit : R=0.23 Ω
- Couple nominal : Cm=1.9 Nm

Analyse et solution

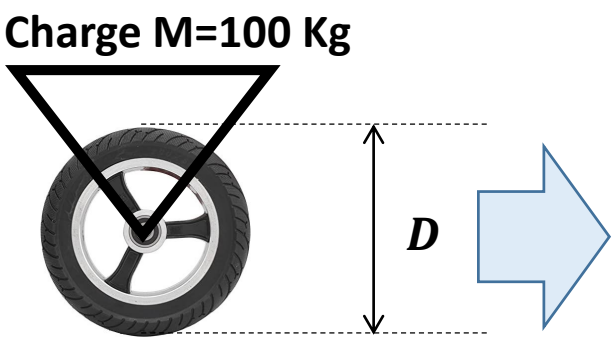
I. Asservissement et réglage de la vitesse selon le degré de la ponte

Choix de réducteur

On choisi un réducteur afin de supporter la charge maximal, on choisi alors un **rapport de transmission de 10:**

$$Cp' \approx 10.Cm \Rightarrow \quad Cp' \approx 19\ Nm > 7.5\ Nm$$

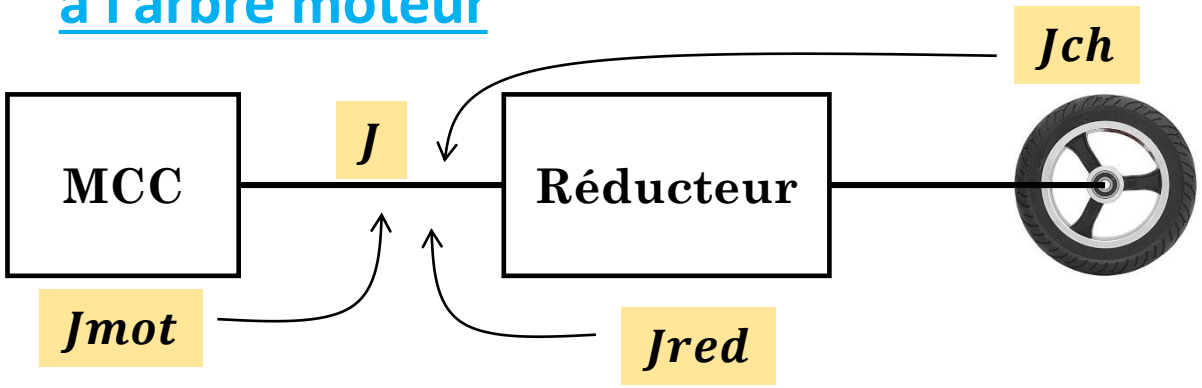
Calcul de moment d'inertie de la charge



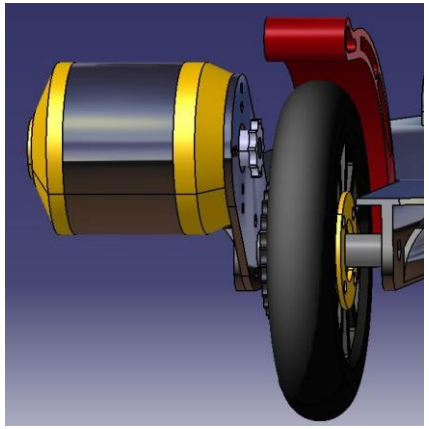
$$Jch = \mathrm{M}.\frac{\mathrm{D}^2}{4}$$

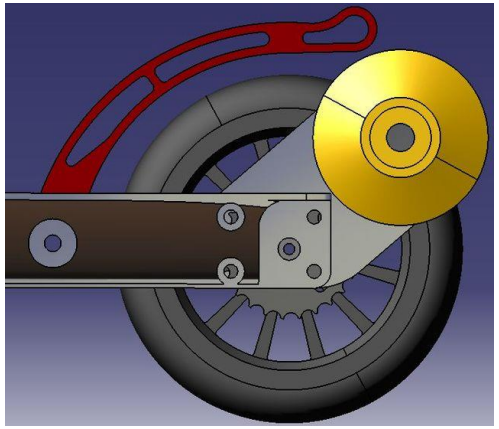
$$\boldsymbol{Jch = 5.4\ Kg.m^2}$$

Calcul de moment d'inertie total ramené à l'arbre moteur



$$\boldsymbol{J = Jmot + Jred + \frac{Jch}{r^2}} \Rightarrow \boldsymbol{J = 0.0611\ Kg.m^2}$$





Analyse et solution

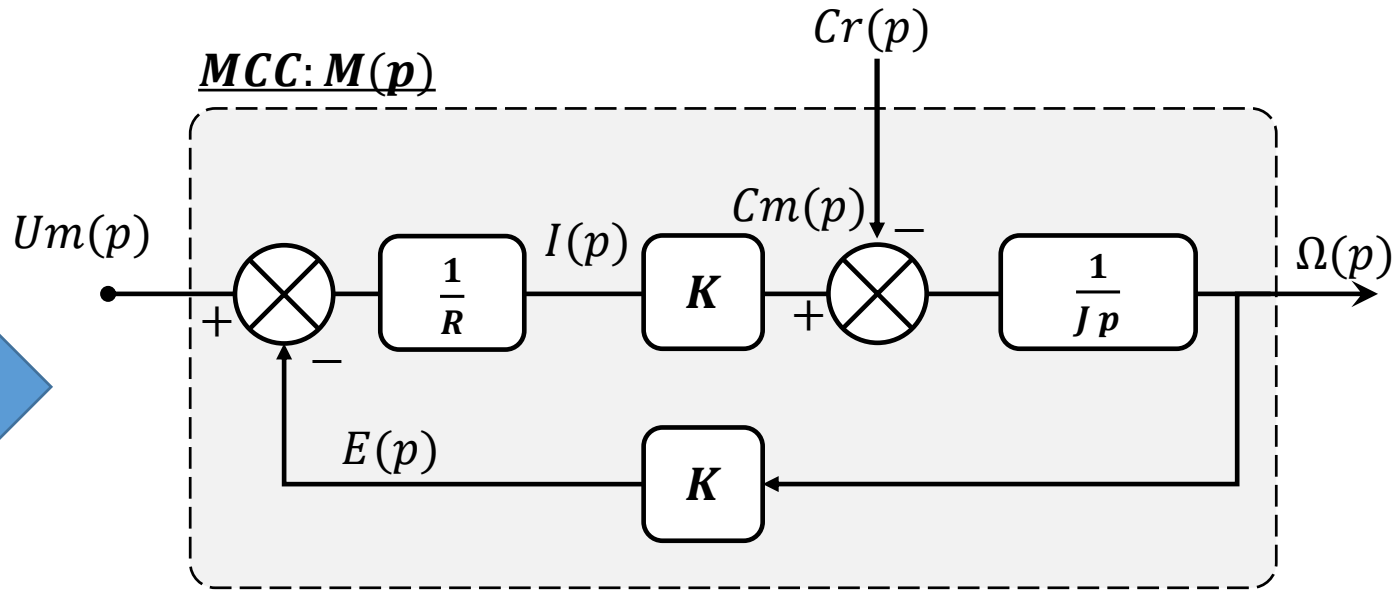
I. Asservissement et réglage de la vitesse selon le degré de la ponte

Modélisation de la MCC

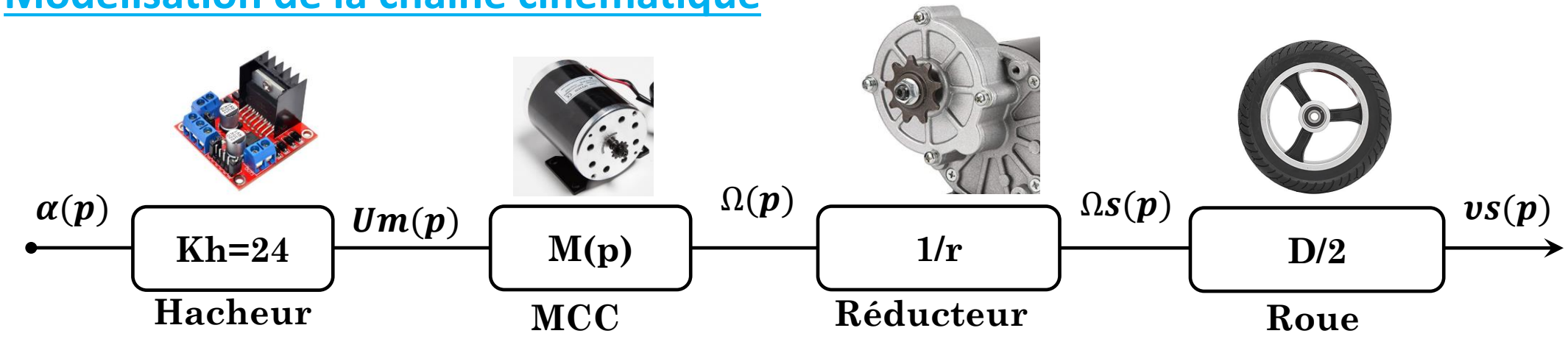
Équations de la machine à courant continu :

E1:	$Um(t) = Ri(t) + e(t)$
E2:	$e(t) = \mathrm{K}\,\Omega(t)$
E3:	$J\frac{d\Omega(t)}{dt} = \mathrm{Cm(t)} - \mathrm{Cr(t)}$
E4:	$\mathrm{Cm(t)} = K\,i(t)$

TL



Modélisation de la chaîne cinématique



Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Schéma fonctionnelle d'asservissement

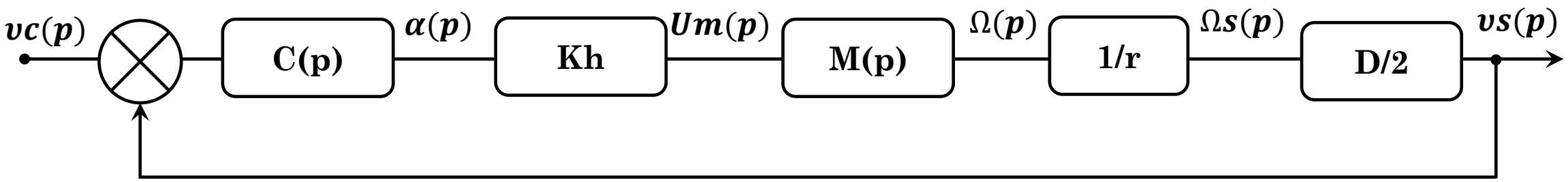
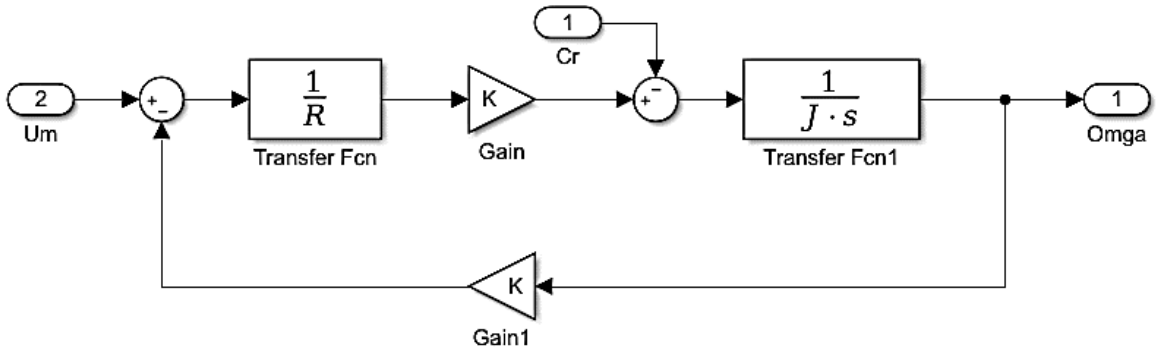
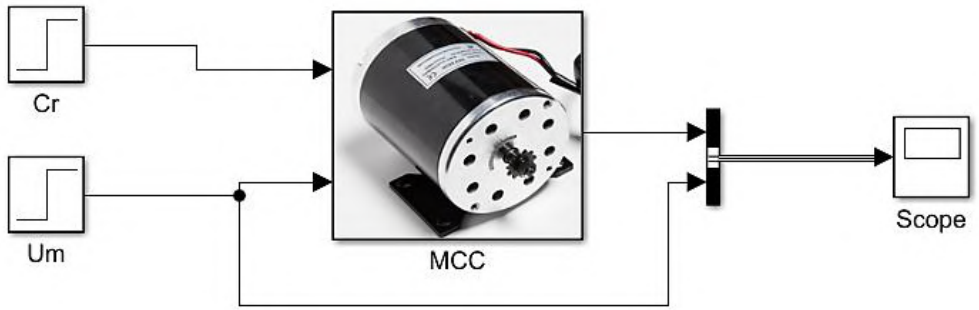


Schéma de simulation

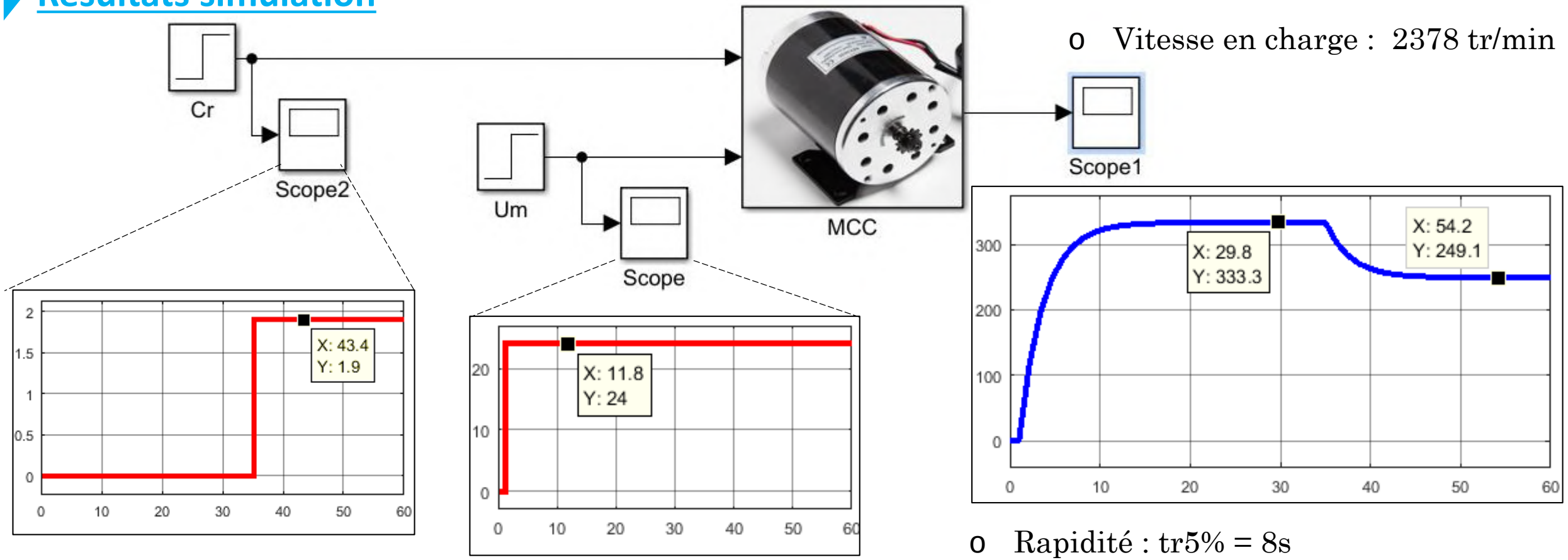




Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Résultats simulation



Couple résistant :

Cr=Cn=1.9 Nm à **t=35s**

Tension nominale:

Um=24V à **t=1s**

- Vitesse à vide : 3182 tr/min
- Vitesse en charge : 2378 tr/min
- Rapidité : tr5% = 8s
- Stabilité : oui
- Précision : ---------

Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Calcul de correcteur pour asservir et régler la vitesse de déplacement

D'après les schéma bloc précédent :

FT de la MCC (Cr=0)	FTBO	FT de correcteur : PI	FTBF
$M(p)=\frac{Km}{1+Tm.p}$	$FTBO(p)=C(p)\frac{Ko}{1+Tm.p}$	$C(p)=Kp.\frac{1+Tip}{Ti\,p}$	$FTBF(p)=\frac{1}{1+Tf.p}$
$Km=\frac{1}{K}$ et $Tm=R.\frac{J}{K^{-2}}$	$Ko=\frac{Kh.Km.D}{2r}$	Ti=Tm	$Tf=\frac{Tm}{Kp.Ko}$

Après application de technique du compensation de pôles afin de réponde au cahier des charge :

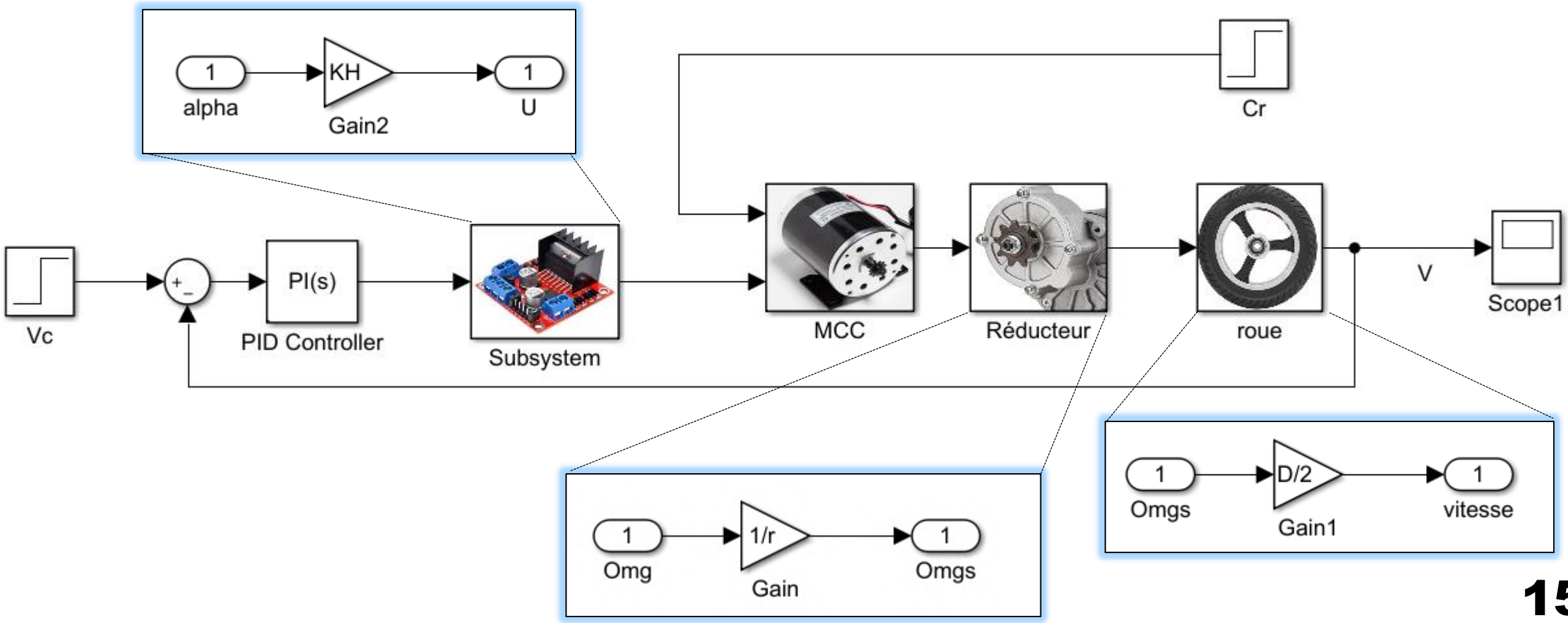
$FTBO'(p)=\frac{KoKp}{Tm\,p}$ ⇒ $FTBF(p)=\frac{1}{1+Tf.p}$ avec $Tf=\frac{Tm}{Kp.Ko}$ ⇒ $Tr5\%=\frac{3\,Tm}{Kp.Ko}$

$C(p)=Kp.\frac{1+Tip}{Ti\,p}$	**Kp**	**Ti**
	2.77	2.71s

Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

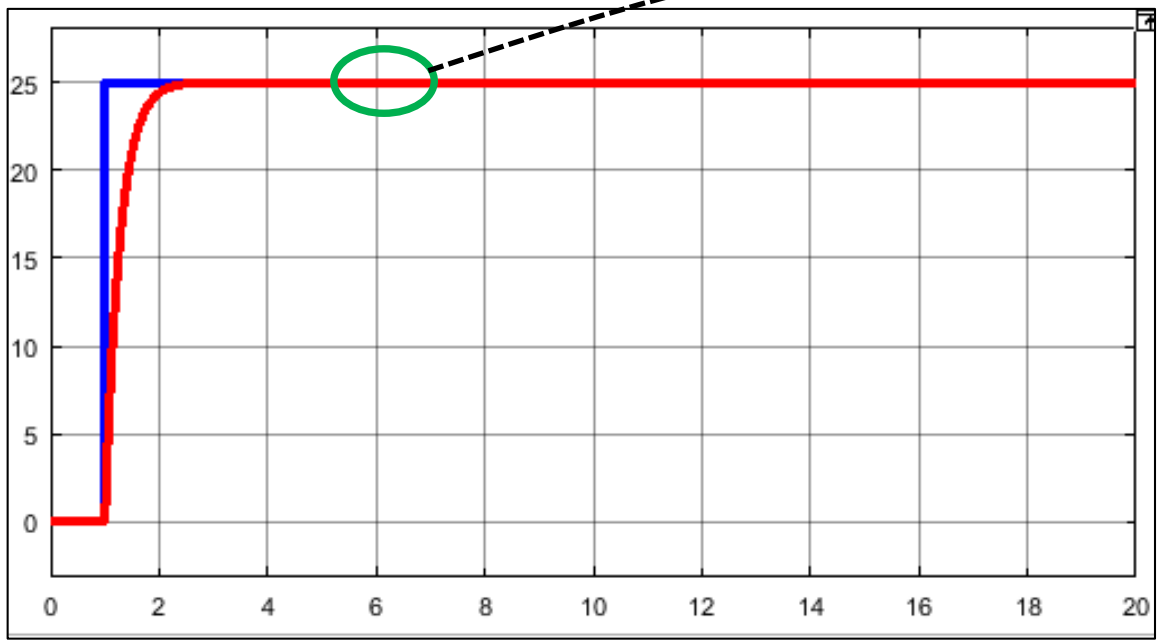
simulation

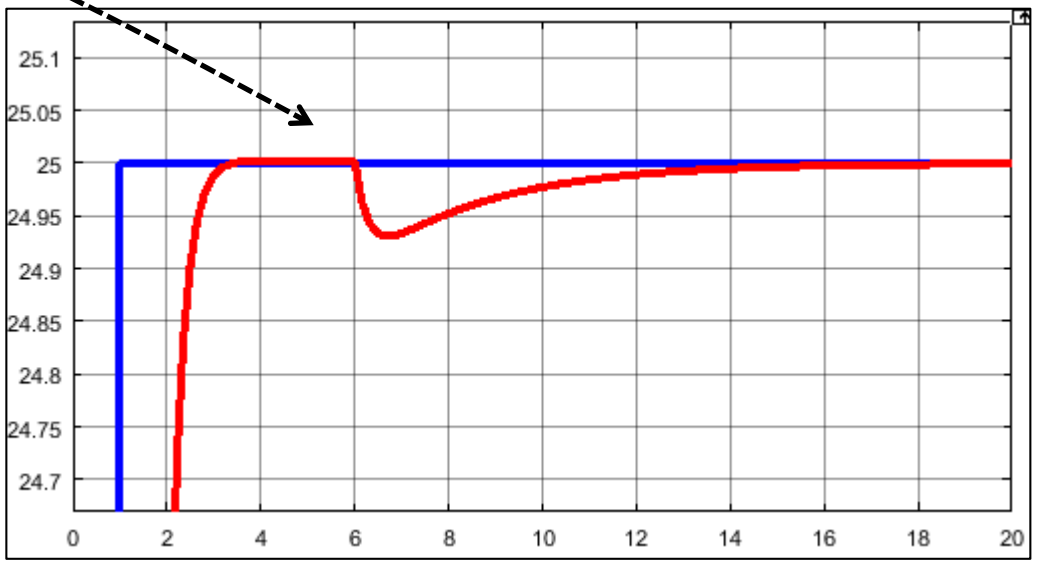


Analyse et solution

I. Asservissement et réglage de la vitesse selon le degré de la ponte

Résultats de simulation





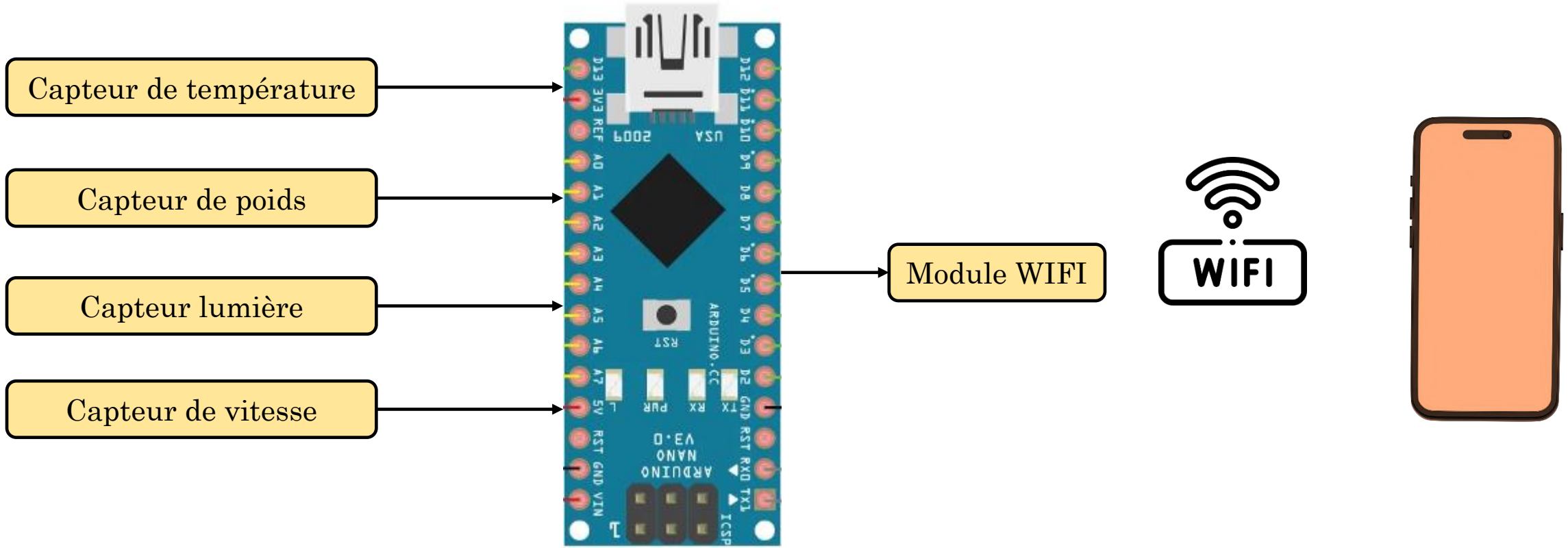
- **Rapidité :** temps de réponse tr=0.8 s
- **Précision :** erreur statique nulle
- **Stabilité :** le système est stable

- **Effet de régulation :** le système supprime la perturbation après leur application

Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Prototype de système.



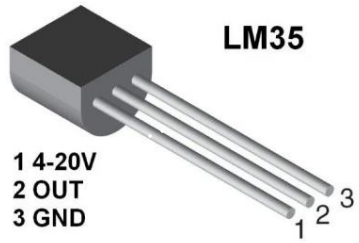
Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

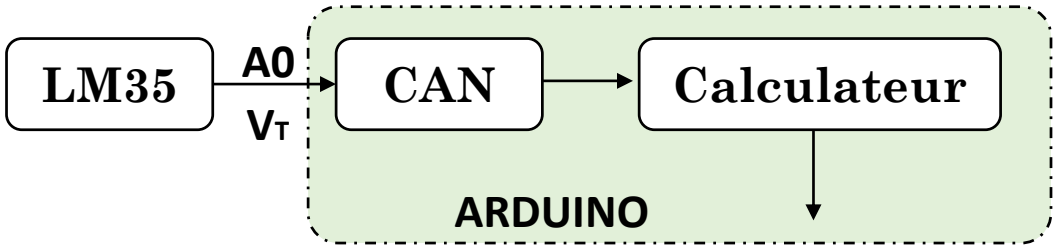
Capteur de température LM35.

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range



❑ Chaîne d'acquisition.



❑ Calcul de la température

- Capteur : $V_T = s \, . \, T$ avec s=0.01 V/°C
- Convertisseur : $V_T = q \, . N_T$
- Calculateur : $T = K \, . N_T$ avec K= 0.48875855

❑ Programme Arduino

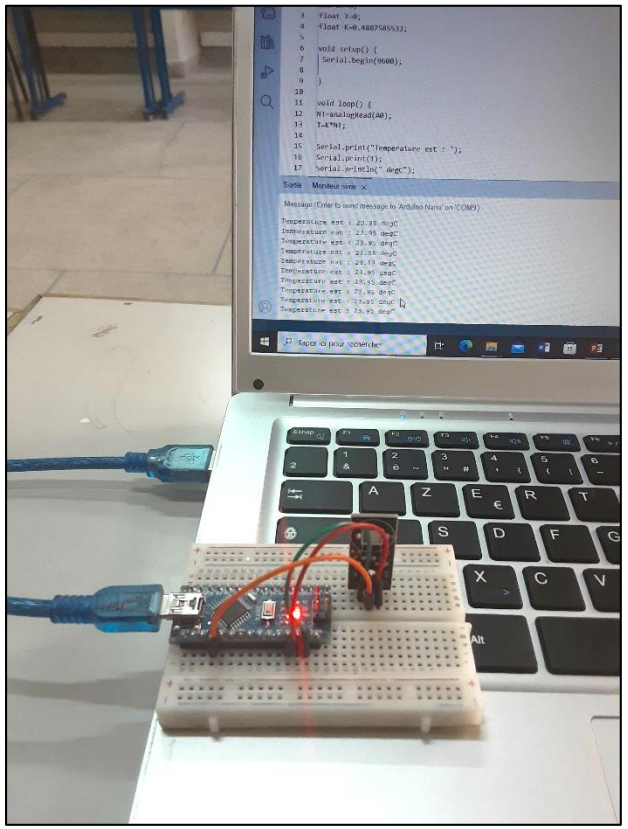

```
11  void loop() {
12  NT=analogRead(A0);
13  T=K*NT;
14
15  Serial.print("Temperature est : ");
16  Serial.print(T);
17  Serial.println(" degC");
18  delay(250);
19  }
```

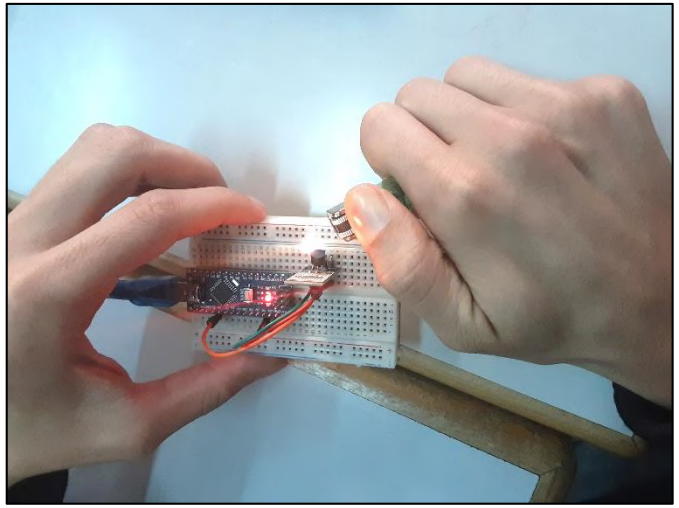
Analyse et solution

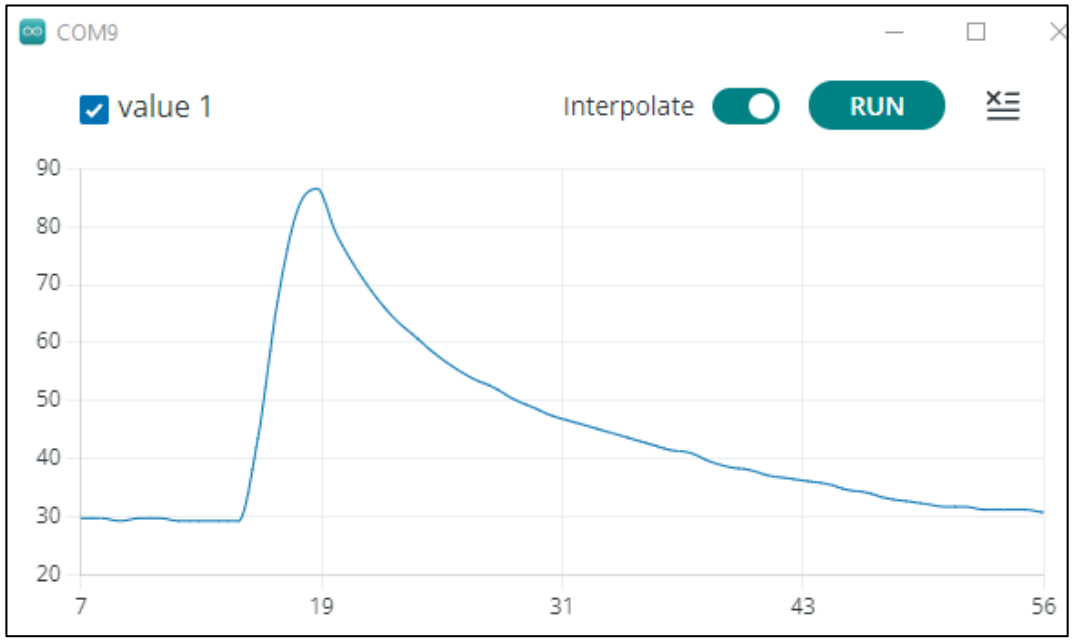
II- Mesure les grandeur vitesse, poids , lumière et la température

Capteur de température LM35.

❑ Résultats



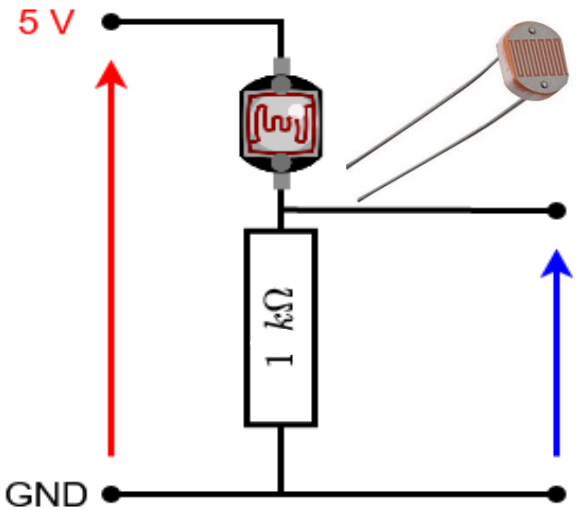




Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

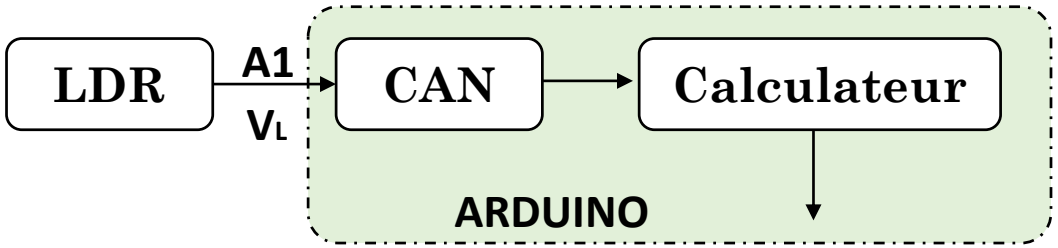
Capteur de lumière .



Photorésistance (LDR ou Light Dependent Resistor)

Exemple de schéma de branchement de LDR

❑ Chaîne d'acquisition.



❑ Calcul de la température

- Convertisseur : $V_L = q \,.\, N_L$
- Calculateur : L = $K1 \,.\, N_L$ avec K= 0.09775171

Si la lumière au dessous de 3% on doit allumé les lampe de trottinette

❑ Programme Arduino

sketch_apr13a.ino

```
float q=0.0048875855;
void setup() {
  Serial.begin(9600);
  pinMode(2,OUTPUT);
}
```

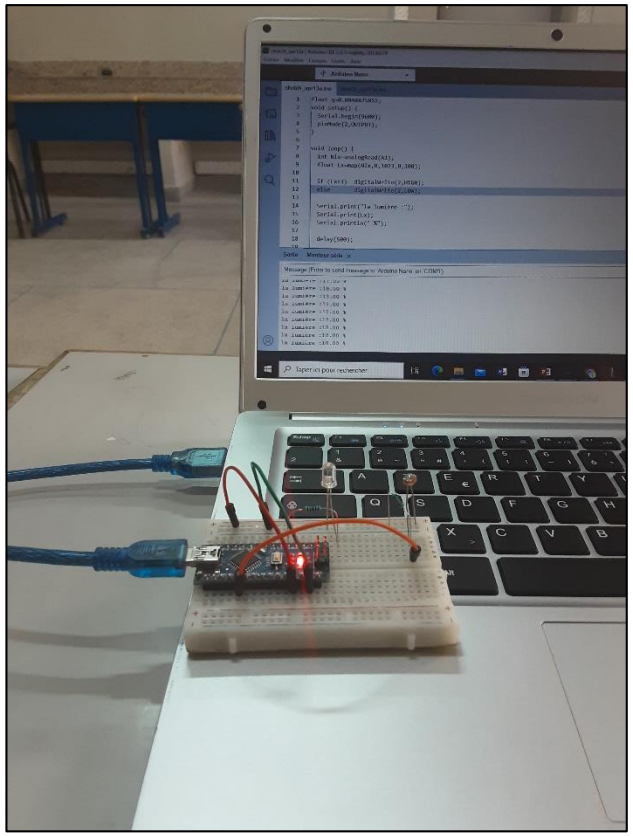
```
void loop() {
  int Nlx=analogRead(A1);
  float Lx=map(Nlx,0,1023,0,100);

  if (Lx<3)  digitalWrite(2,HIGH);
  else       digitalWrite(2,LOW);

  Serial.print("la lumière :");
  Serial.print(Lx);
  Serial.println(" %");

  delay(500);

}
```

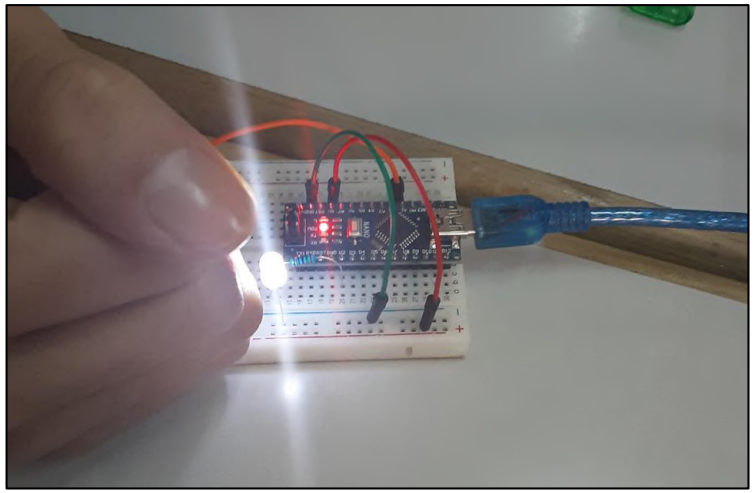
Analyse et solution

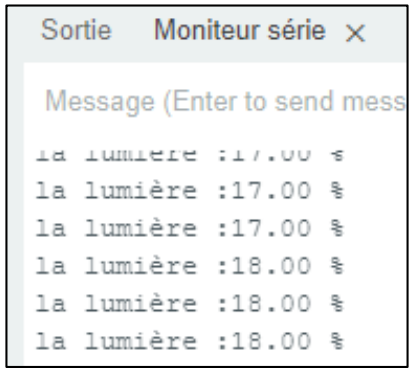
II- Mesure les grandeur vitesse, poids , lumière et la température

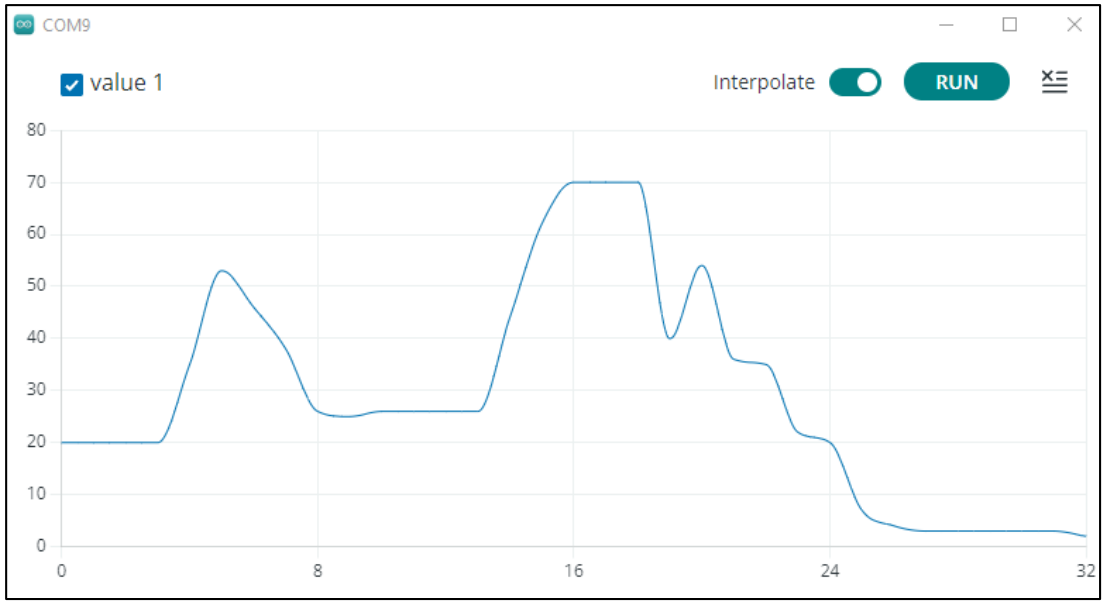
Capteur de lumière .

❑ Résultats







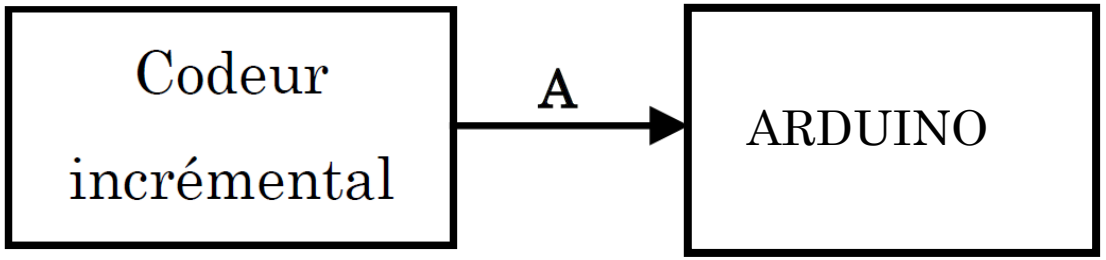


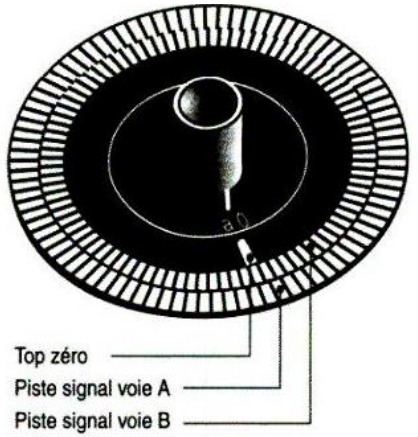
Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Capteur de vitesse : codeur incrémental .

Le codeur incrémental est consacré à des applications où l'information de position est obtenue par mesure du déplacement de l'objet. Le codeur délivre un train d'impulsions dont le nombre permet de déduire la valeur du déplacement ainsi que la vitesse de rotation.





La fréquence l'un des deux signaux A et B permet de récupérer la vitesse de rotation en tour par minute :

$$F(hz) = R.\frac{N((\mathrm{tr/min})}{60}$$

Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Capteur de vitesse : codeur incrémental .

- **Mesure de la période par Arduino**

$$T = 2\,Nt\,\frac{5.10^{-3}}{4969}$$

- **Calcul de la fréquence**

$$f = \frac{1}{T} = \frac{496\,910^{3}}{Nt}$$

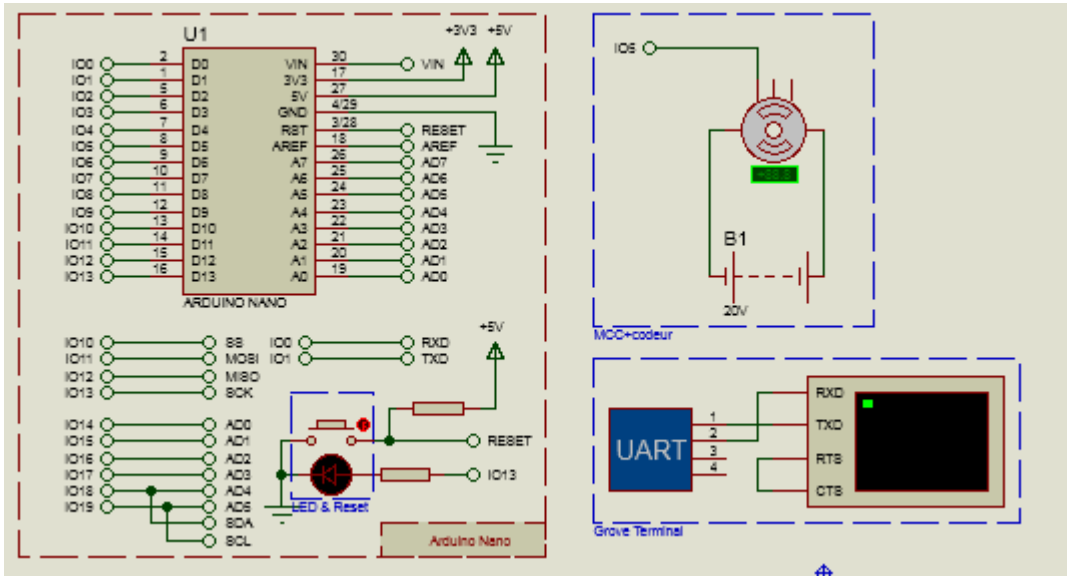
- **Calcul de la vitesse en tr/min**

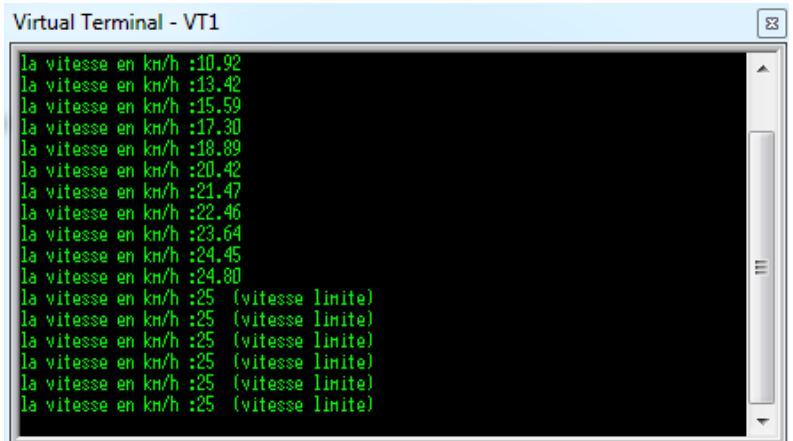
$$N = 60.\frac{f}{R}$$ **Avec R=100**

- **Calcul de la vitesse en km/h**

$$v = 3.6.r\,.\frac{2\pi N}{60}$$

$$v = 0.024408\,f$$





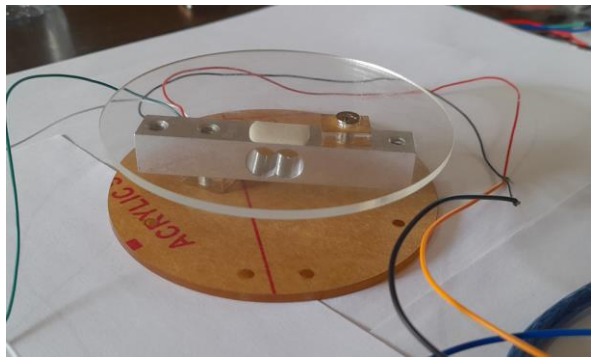
Programme voir annexe 1

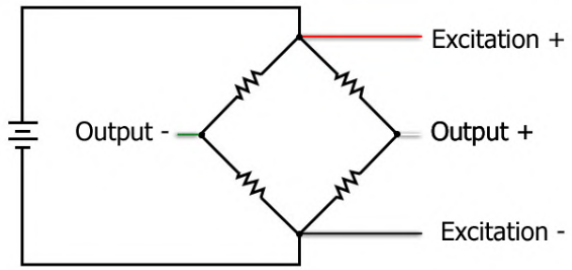
Analyse et solution

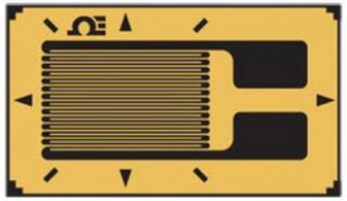
II- Mesure les grandeur vitesse, poids , lumière et la température

Capteur de poids

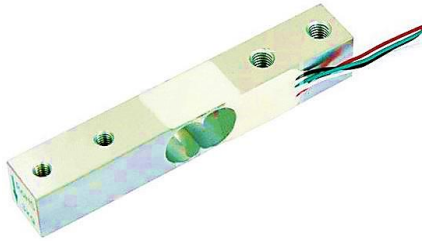
❖ Constituions





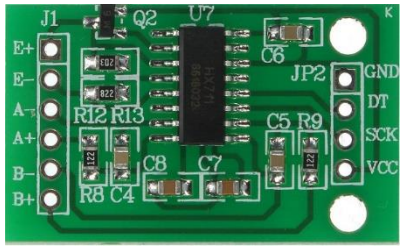


4 jauges



structure déformable

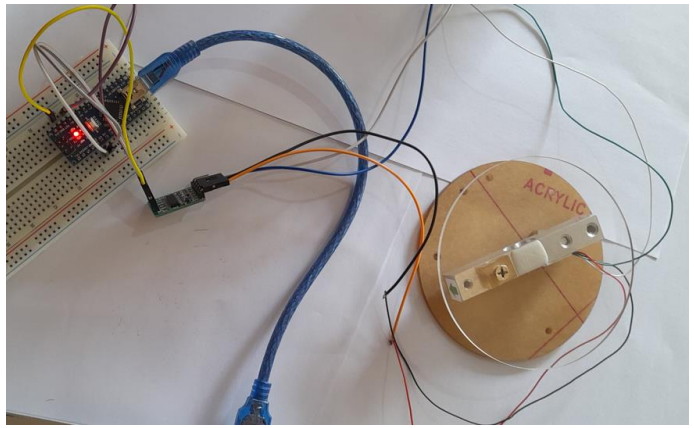
10 Kg



HX711

- AOP
- CAN 12 bits

❖ Calibration



Coefficient de Calibration:

20780.0

❖ Etalonnage

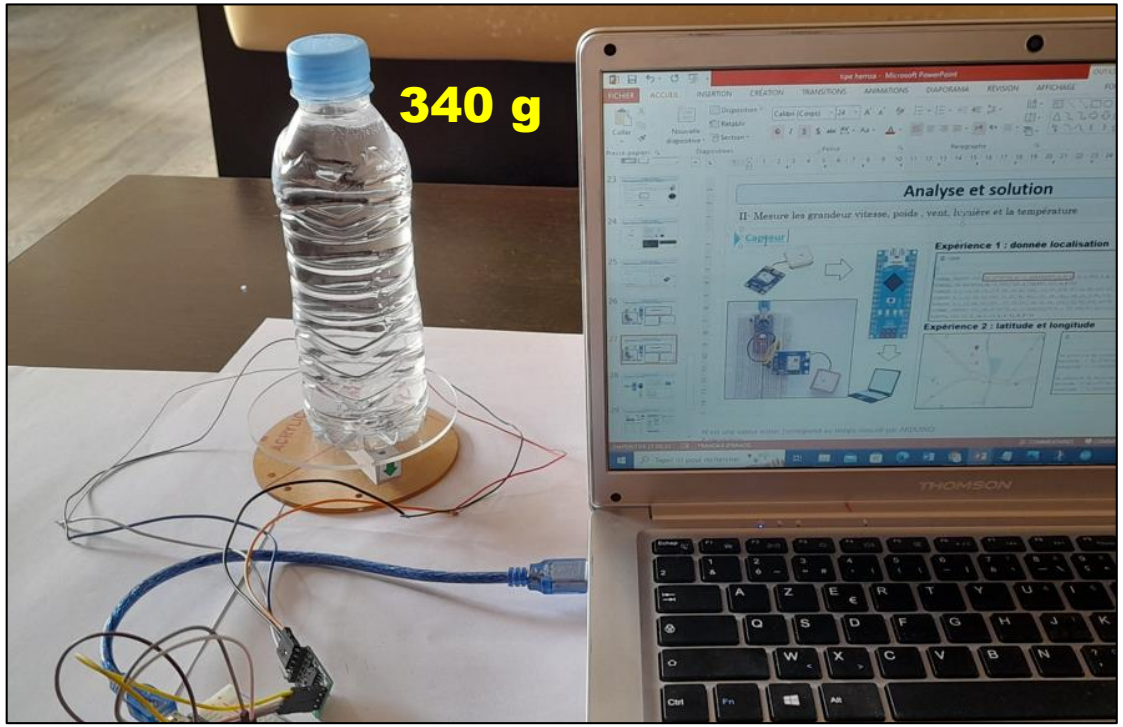
Expression du masse :

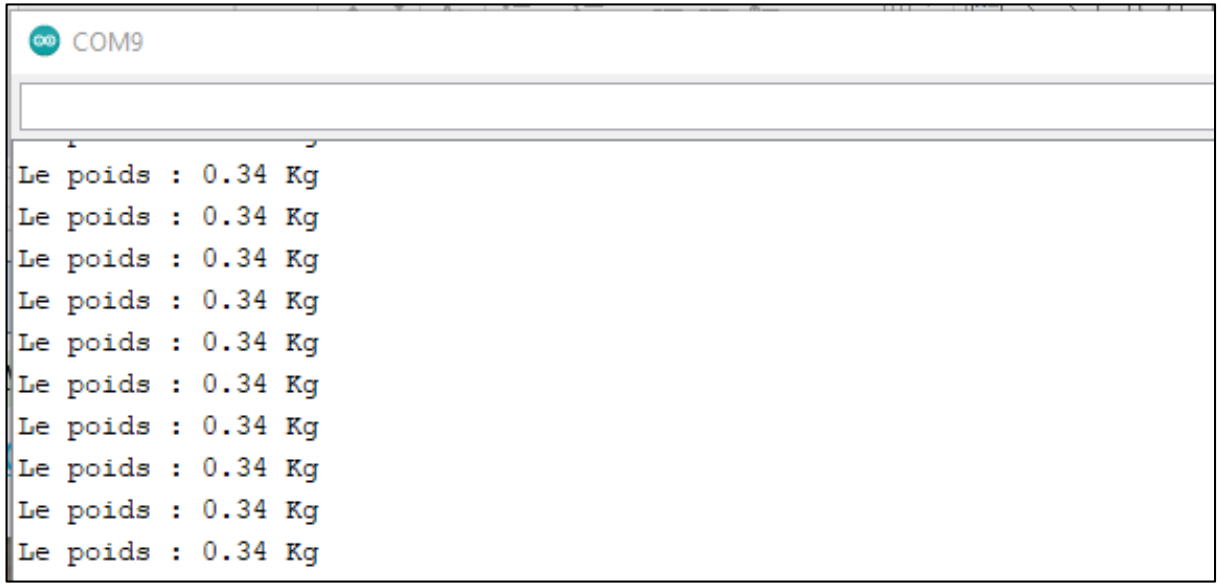
$$M(kg) = \frac{0.336}{-4.4} \; N = 0.076363 \text{ N}$$

Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Capteur de poids





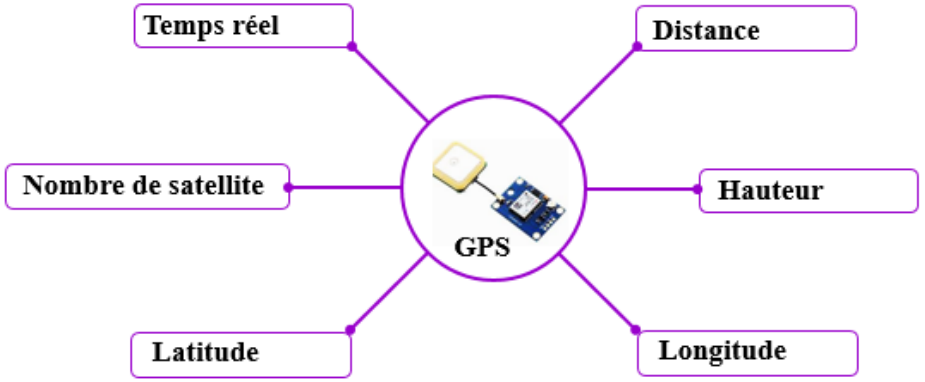
Le programme Arduino en Annexe 2

Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Localisation de trottinette par GPS.

Le GPS est un système de positionnement par satellite qui fournit des informations de localisation et de temps précises partout sur Terre, opéré par l'US Air Force. i



❑ **Caractéristique de GPS NEO-6M-0-001**

Le GPS NEO-6M-0-001 est un module GPS compact et facile à intégrer, doté des caractéristiques suivantes :

- Alimentation : 3 à 5V
- Interface série UART

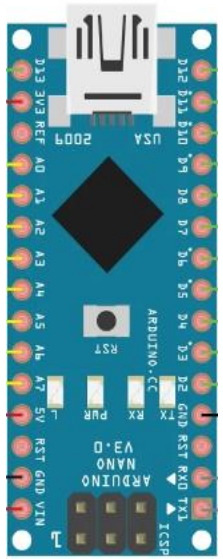
❑ **Trame de communication**

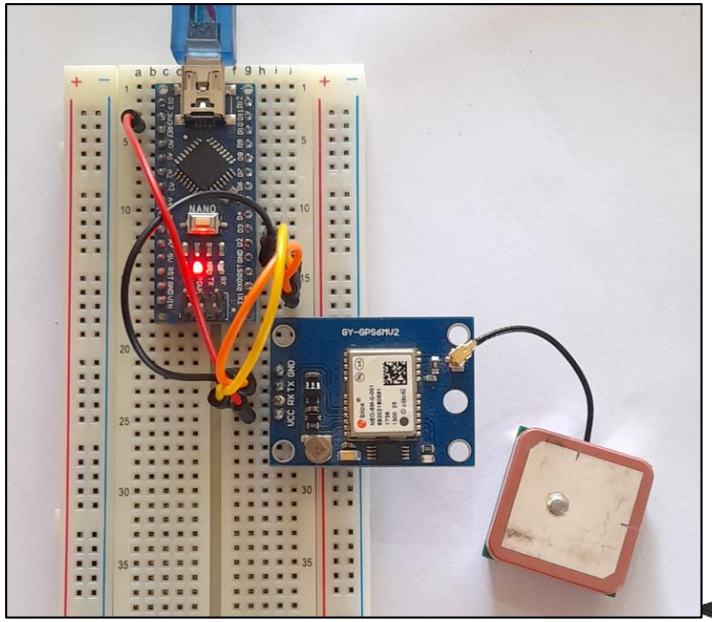
entête	heures	Latitude	Longitude	N satellite	Altitude	…	…	Checksum

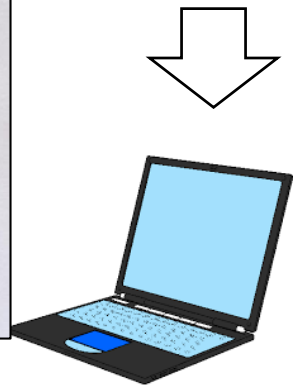
Analyse et solution

II- Mesure les grandeur vitesse, poids , lumière et la température

Localisation de trottinette par GPS.





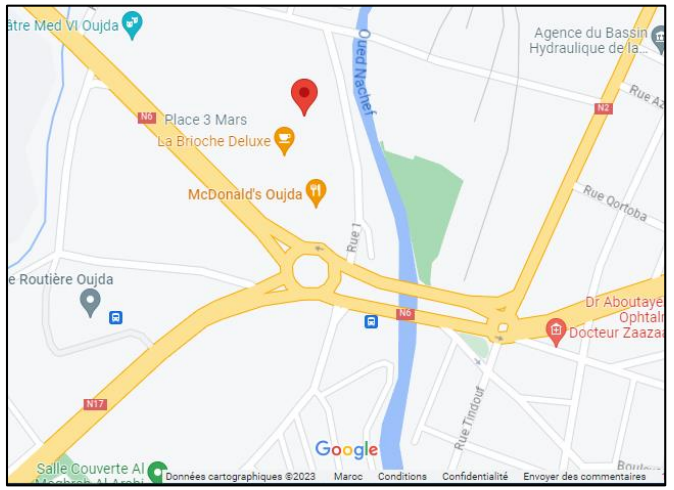


Expérience 1 : donnée localisation

COM9

```
$GPGA,092457.000,34.6778739,N,-1.929305277,O,6,0[illegible],4.0,453.2,M,0.0,,*6D
$GPGLL,34.6639824,N,-1.9363725,E,092457.000,A,E*59
$GPGSV,3,1,12,08,16,273,28,10,46,042,,11,07,315,37,14,56,315,37,14,56,315,41*7E
$GPGSV,3,1,12,08,16,230,41,20,36,081,,21,29,152,,25,10,0198,*72
$GPGSV,3,1,12,26,06,185,,27,23,241,26,31,54,182,,32,50,353,34*7D
$CPRMC,092457.000,34.6778739,N,-1.929305277,O,2.33,332.91,3110118,,,E*6C
$GPVTG,332.51,T,,M,2.33,N,4.32,K,E*34
```

Expérience 2 : latitude et longitude



COM9

```
La position du trottinette :
Latitude  : 34.6778738 N
Longitude : -1.929305273 O

La position du trottinette :
Latitude  : 34.6778741 N
Longitude : -1.929305275 O
```

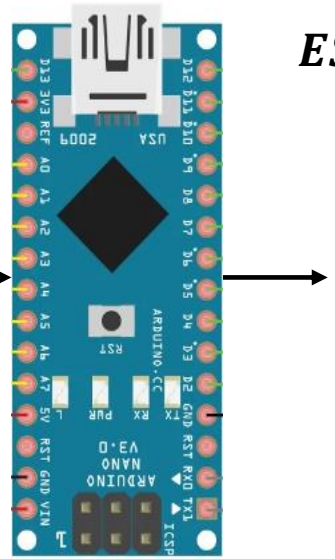
Programme voir annexe 3

Analyse et solution

III- communication et application mobile

❑ Schéma de principe

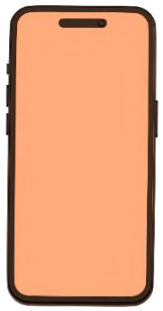
capteurs

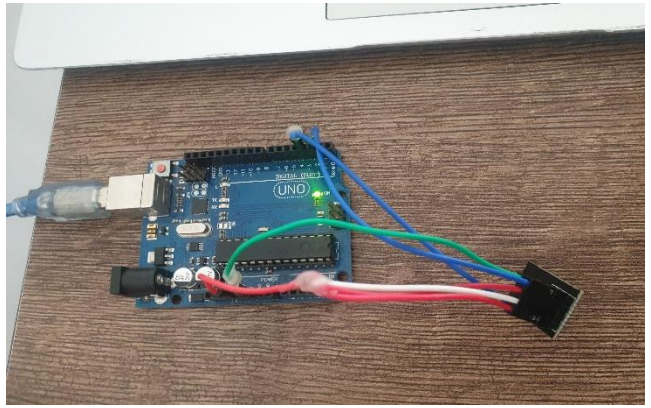


ARDUINO NANO

ESP8266 E01

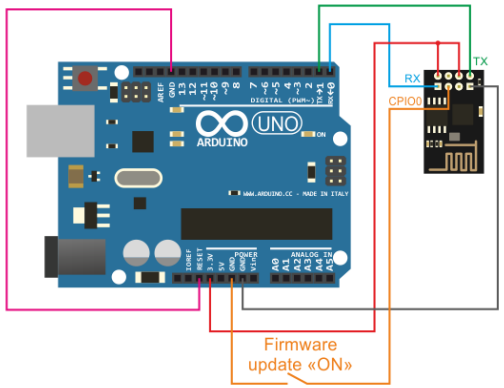
WIFI





❑ Configuration de ESP8266

- Réinitialisation de module
- Définir la vitesse de communication
- Définir le module en mode serveur
- Définir les cordonnées de wifi



Les commande AT

AT

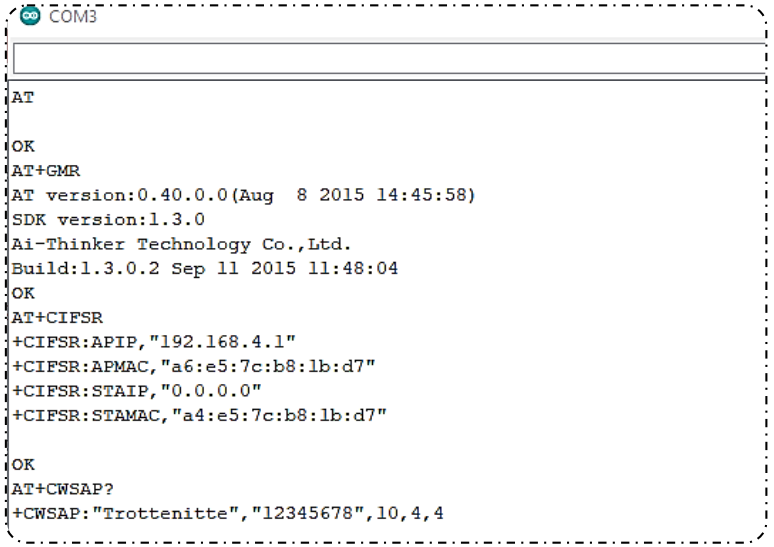
AT+RST

AT+GMR

AT+CWMODE

AT+CWSAP

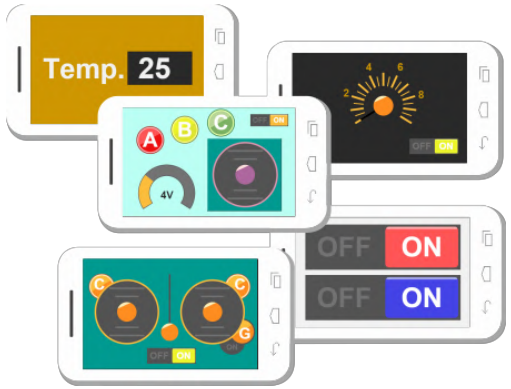
AT+UART_DEF



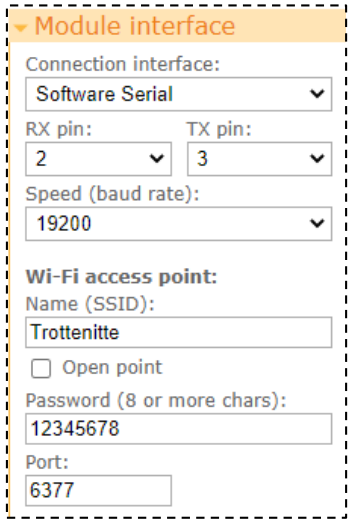
Analyse et solution

III- communication et application mobile

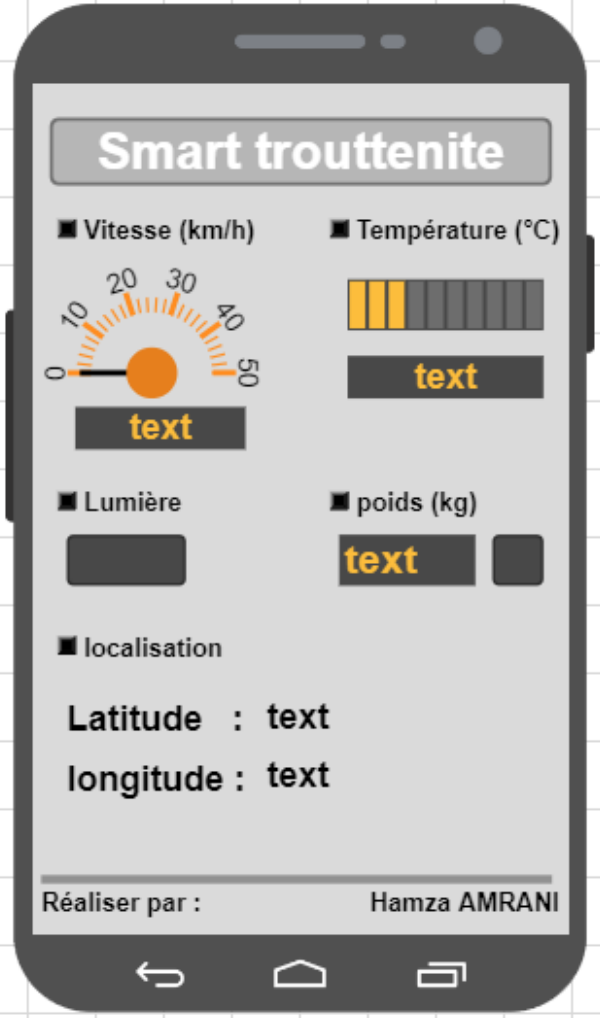
❑ Remotexy?



❑ Configuration au niveau de site



❑ Conception de l'application



Description

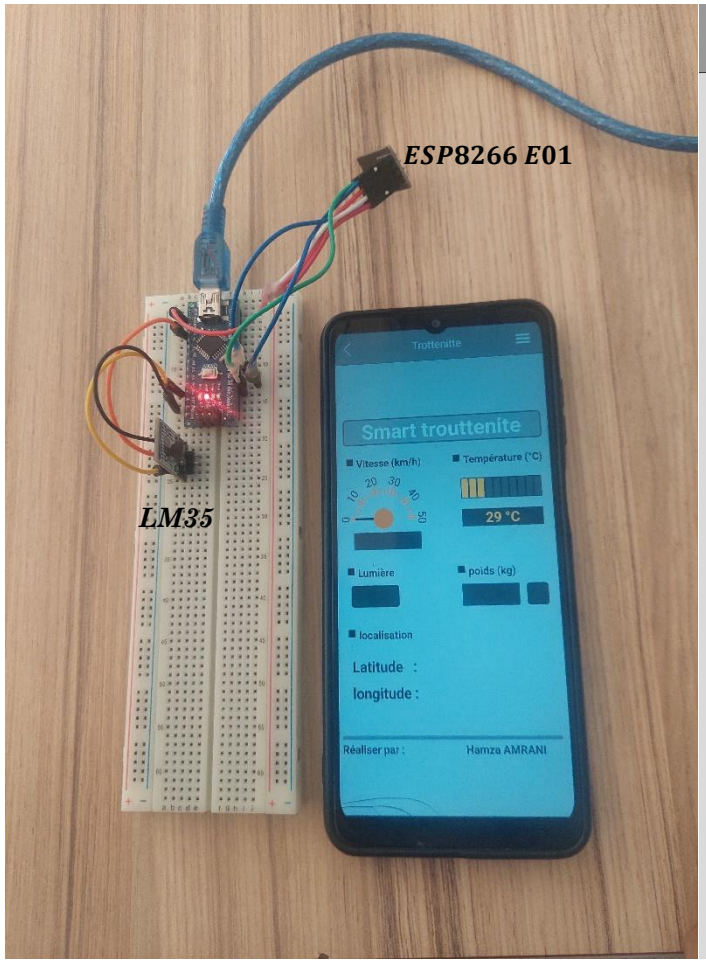
- Affichage de température
- Affichage de vitesse
- Affichage de l'état de lumière
- Affichage le poids porté
- Affichages des cordonnées de localisation par GPS

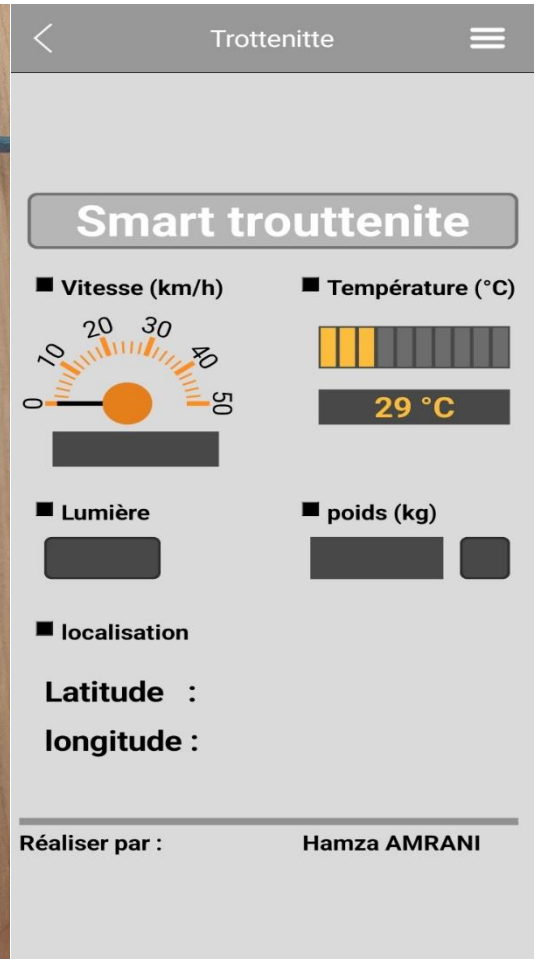
Analyse et solution

III- communication et application mobile

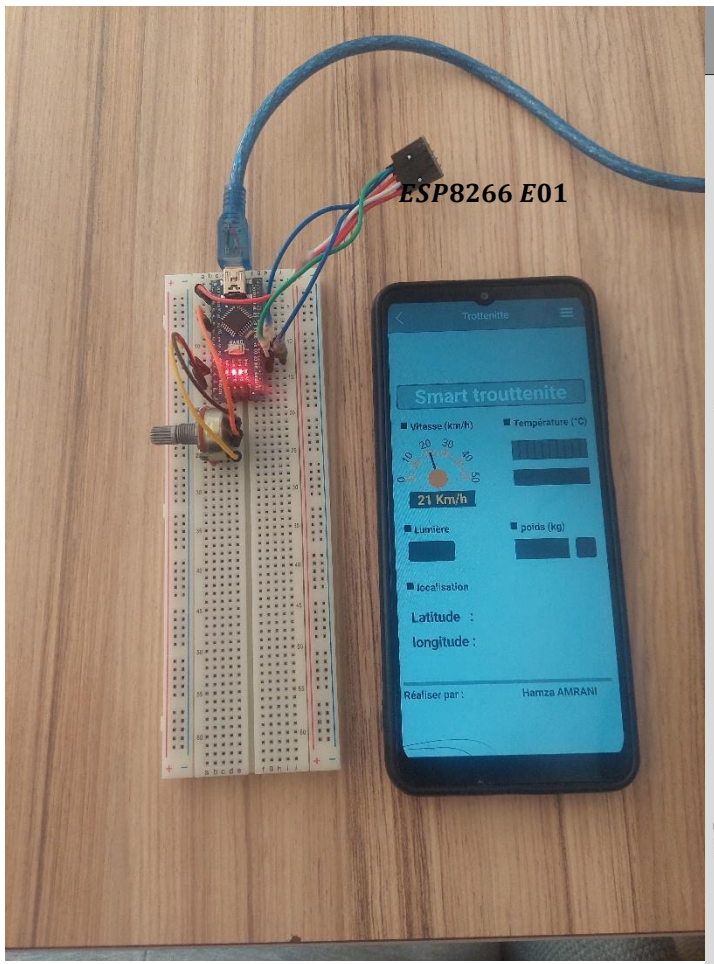
❑ Résultats de chaque capteur.

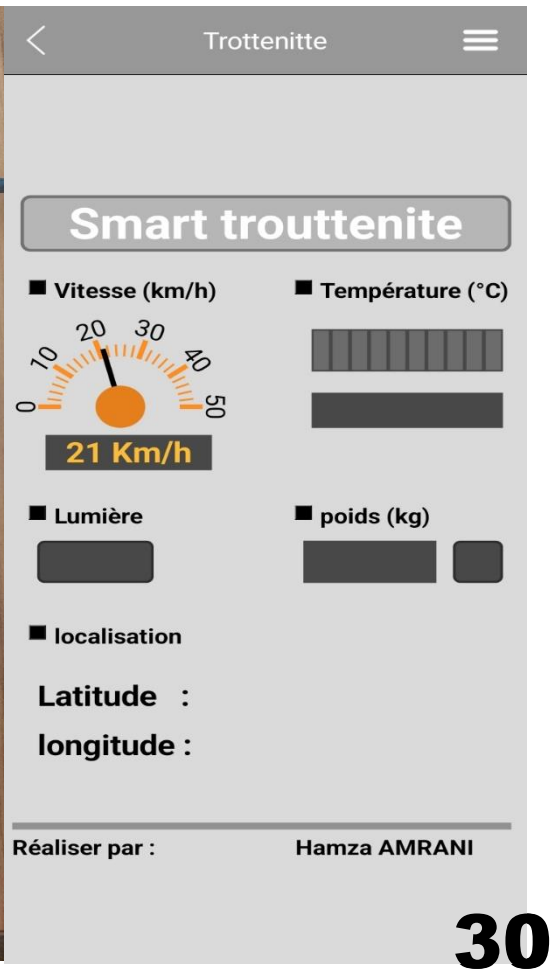
Capteur de* température*: LM35





potentiomètre



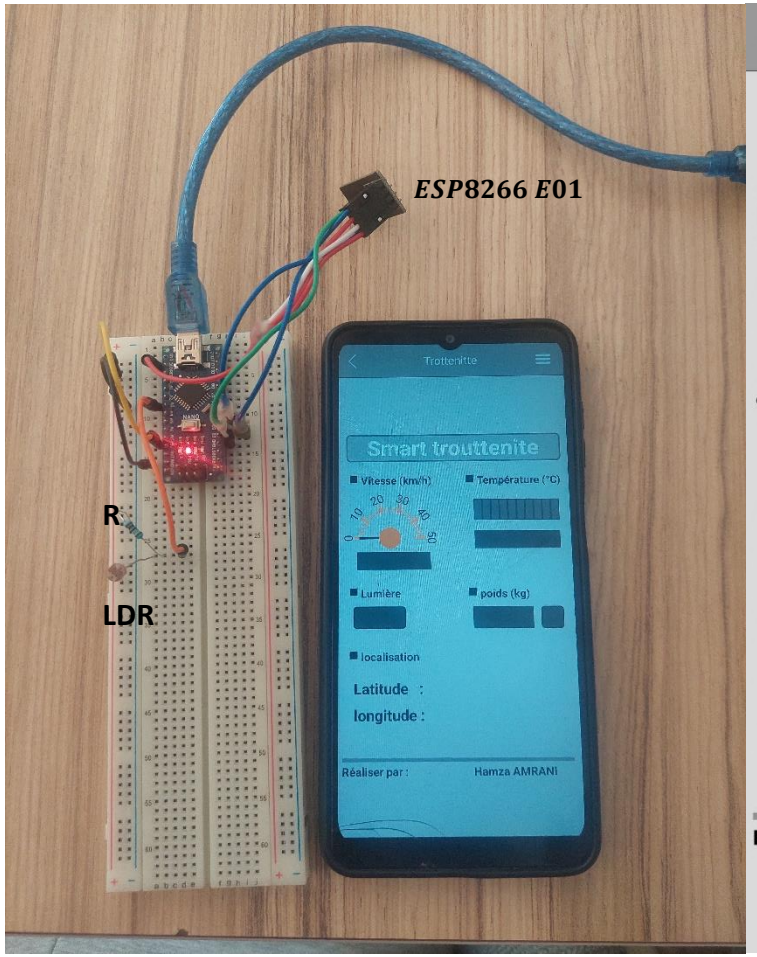


Analyse et solution

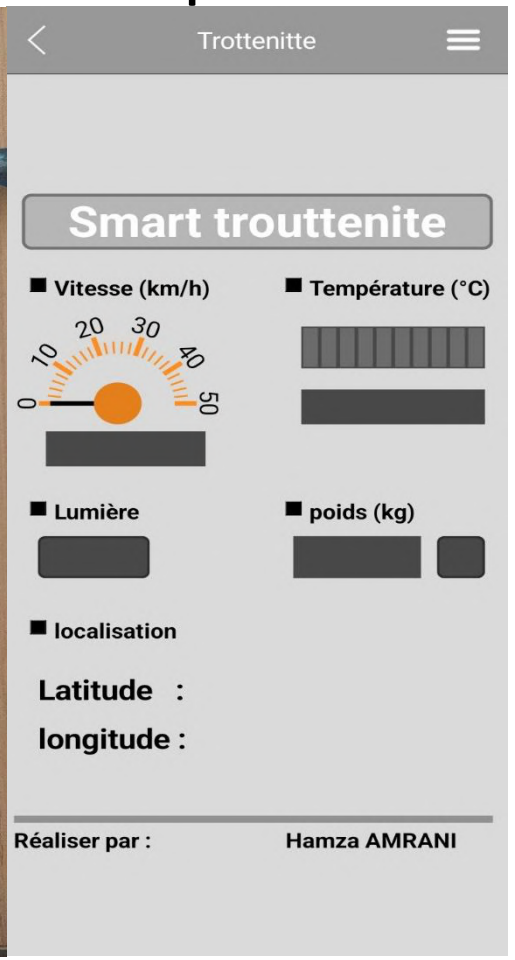
III- communication et application mobile

❑ Résultats de chaque capteur.

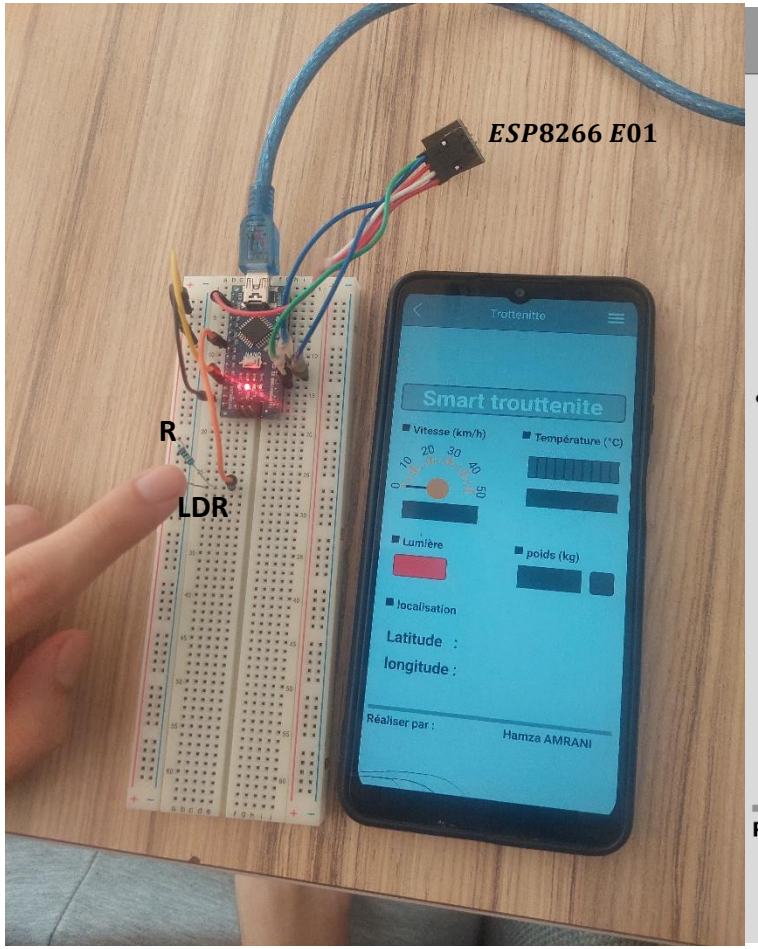
Capteur de lumière

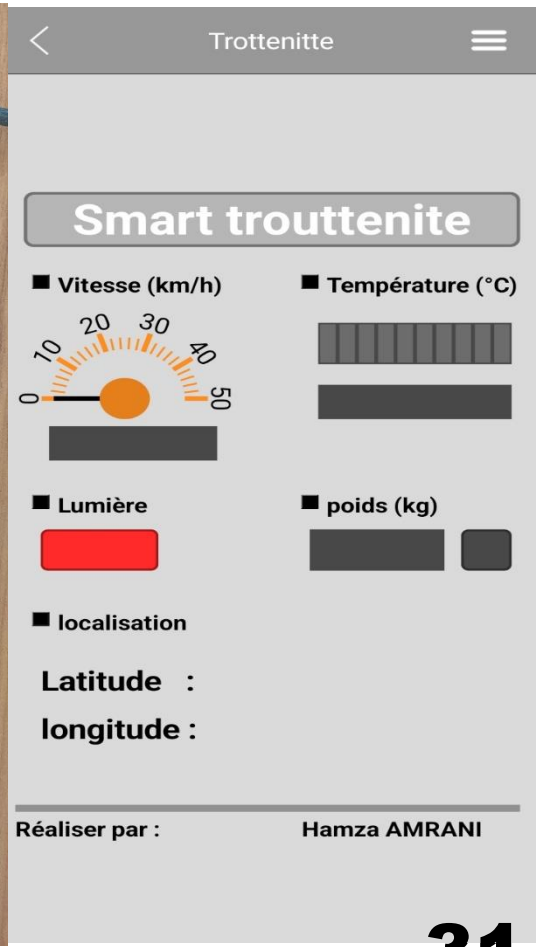


Lampe éteinte



Lampe allumée





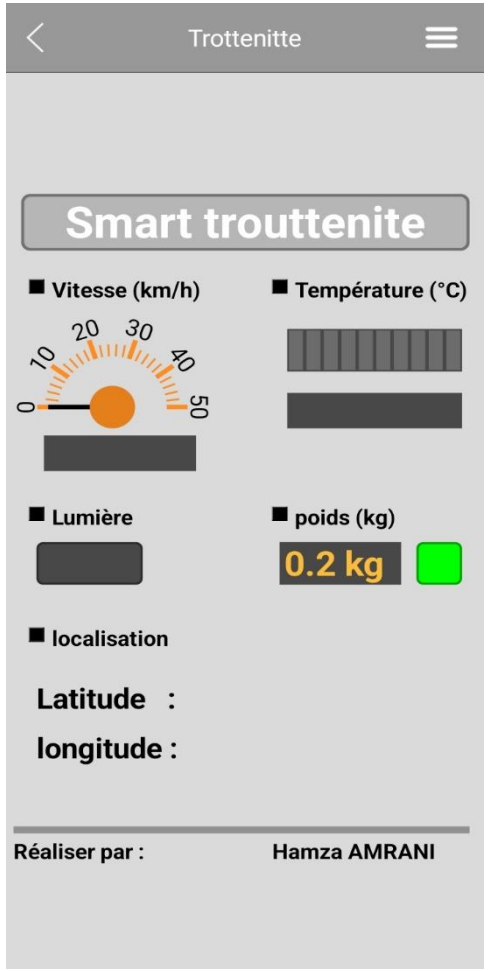
Analyse et solution

III- communication et application mobile

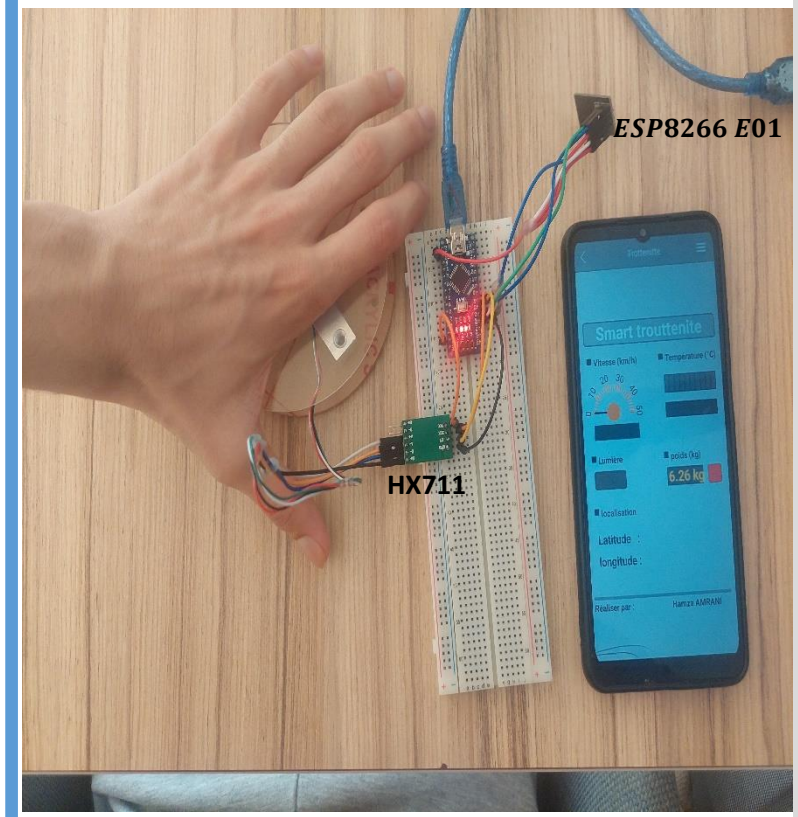
❑ Résultats de chaque capteur.

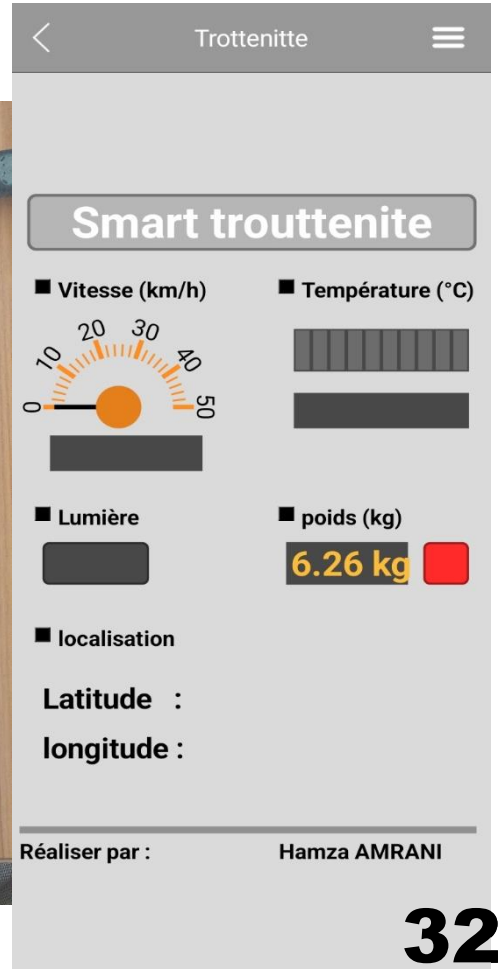
Capteur de poids

Le poids autorisé



Le poids non autorisé



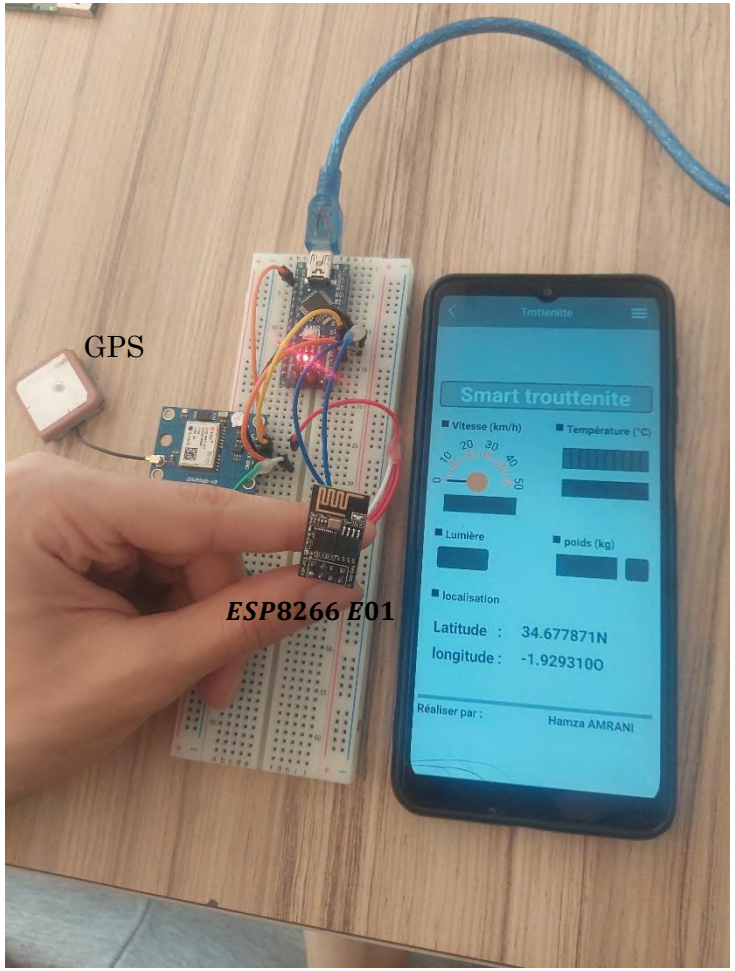


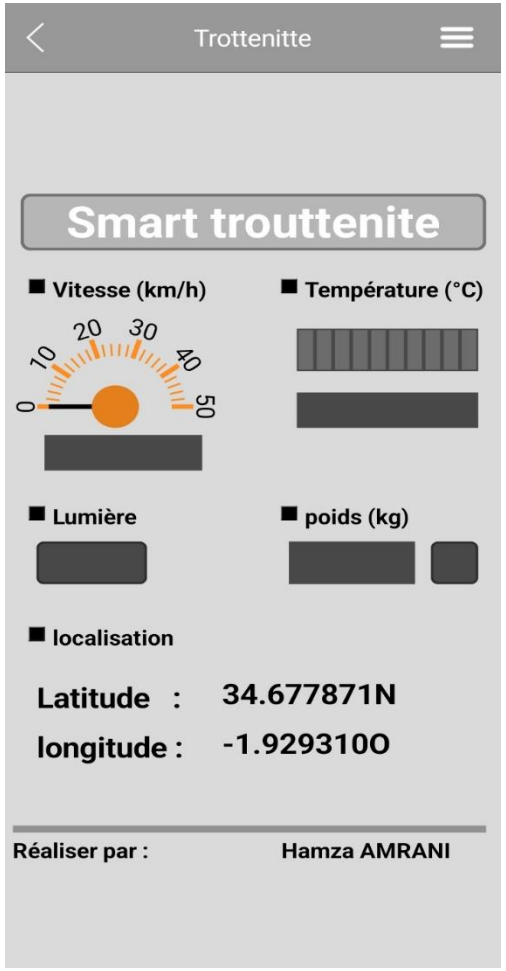
Analyse et solution

III- communication et application mobile

❑ Résultats de chaque capteur.

GPS NEO-6M-0-001 **:**

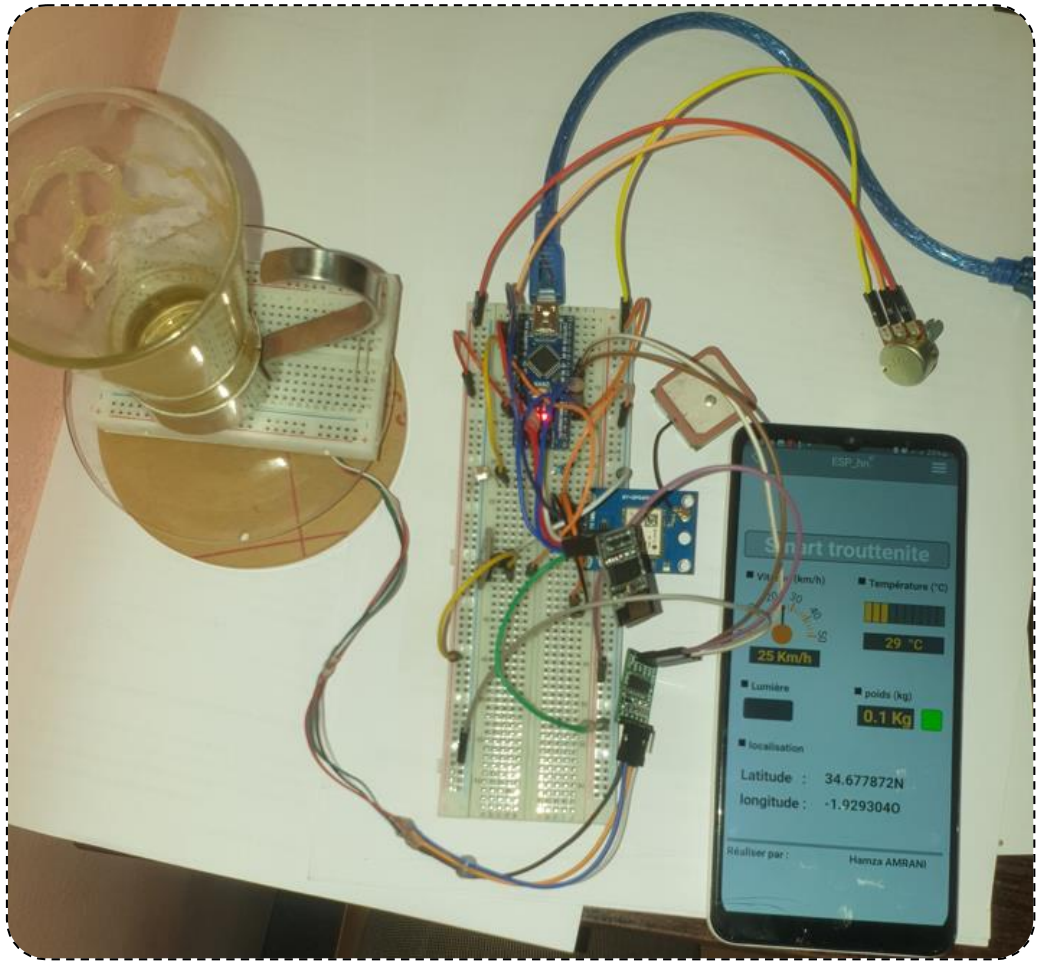


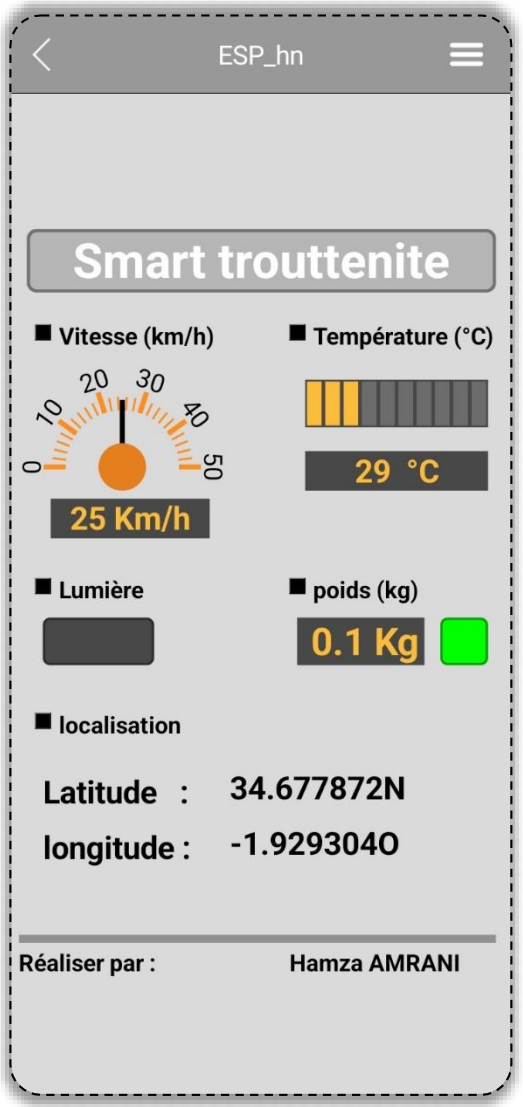


Analyse et solution

III- communication et application mobile

❑ Résultats





Programme voir annexe 34

Conclusion

En conclusion, mon travail sur la "smart trottinette" a exploré les aspects clés du concept. J'ai choisi un moteur approprié, développé un asservissement efficace, utilisé des capteurs et intégré un système GPS. L'ensemble de ces réalisations constitue une avancée majeure vers une trottinette intelligente.

Fin de présentation

Merci pour votre attention

Annexe

ANNEXE 1: codeur incrémental

*main.ino

```
int pin = 5;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseIn(pin, HIGH);

 float F=(993.8*1000)/(2*duration);
 float vitesse=3.6*0.00678*F;

 if(vitesse >25)
 {
  Serial.print("la vitesse en km/h :");
  Serial.print(25);
  Serial.println("  (vitesse limite)");
  }

 else {
  Serial.print("la vitesse en km/h :");
  Serial.println(vitesse);}

  delay(500);
}

```

Annexe

ANNEXE 2: Capteur de poids

poids.ino

```
#include "HX711.h"

#define DEBUG_HX711
#define calibration 20780.0   // le poide M=0 kg


byte pinData = 2;
byte pinClk = 3;

HX711 poids;

void setup() {

#ifdef DEBUG_HX711
  Serial.begin(9600);
  Serial.println("le poids ");
#endif

  poids.begin(pinData, pinClk);
  poids.set_scale(calibration);
  poids.tare();

}

void loop() {
#ifdef DEBUG_HX711
  Serial.print("Le poids : ");
  float N=poids.get_units();
  float M=-0.0763636*N;    //étalonnage (0.336 ---> -4.4)
  Serial.print(M);
  Serial.print(" Kg");
  Serial.println();
#endif
}
```

Annexe

ANNEXE 3: GPS

simple_test.ino

```
#include <SoftwareSerial.h>

#include <TinyGPS.h>

TinyGPS gps;
SoftwareSerial ss(4, 3);

void setup()
{
  Serial.begin(115200);
  ss.begin(4800);

  Serial.print("Simple TinyGPS library v. "); Serial.println(TinyGPS::library_version());
  Serial.println("by Mikal Hart");
  Serial.println();
}

void loop()
{
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;

  // For one second we parse GPS data and report some key values
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }
```

```
  if (newData)
  {
    float flat, flon;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    Serial.print("LAT=");
    Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
    Serial.print(" LON=");
    Serial.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
    Serial.print(" SAT=");
    Serial.print(gps.satellites() == TinyGPS::GPS_INVALID_SATELLITES ? 0 : gps.satellites());
    Serial.print(" PREC=");
    Serial.print(gps.hdop() == TinyGPS::GPS_INVALID_HDOP ? 0 : gps.hdop());
  }

  gps.stats(&chars, &sentences, &failed);
  Serial.print(" CHARS=");
  Serial.print(chars);
  Serial.print(" SENTENCES=");
  Serial.print(sentences);
  Serial.print(" CSUM ERR=");
  Serial.println(failed);
  if (chars == 0)
    Serial.println("** No characters received from GPS: check wiring **");
}
```

Annexe

Programme de l'application

```
1
2   // RemoteXY select connection mode and include library
3   #define REMOTEXY_MODE__ESP8266_SOFTSERIAL_POINT
4   #include <SoftwareSerial.h>
5   #include <RemoteXY.h>
6   #include "HX711.h"
7   #include <TinyGPS.h>
8
9   #define DEBUG_HX711
10  #define calibration 20780.0   // le poide M=0 kg
11
12  // RemoteXY connection settings esp8266
13  #define REMOTEXY_SERIAL_RX 2
14  #define REMOTEXY_SERIAL_TX 3
15  #define REMOTEXY_SERIAL_SPEED 19200
16  #define REMOTEXY_WIFI_SSID "Trottenitte"
17  #define REMOTEXY_WIFI_PASSWORD "12345678"
18  #define REMOTEXY_SERVER_PORT 6377
19  #define REMOTEXY_ACCESS_PASSWORD "987654321"
20
```

```
22  // RemoteXY configurate
23  #pragma pack(push, 1)
24  uint8_t RemoteXY_CONF[] =   // 385 bytes
25    { 255,1,0,62,0,122,1,16,30,1,1,1,2,4,59,8,29,31,83,109,
26    97,114,116,32,116,114,111,117,116,116,101,110,105,116,101,0,130,3,1,93,
27    60,1,28,129,0,1,95,41,3,24,82,195,169,97,108,105,115,101,114,32,
28    112,97,114,32,58,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
29    32,32,32,32,32,32,32,32,32,72,97,109,122,97,32,65,77,82,65,78,
30    73,0,71,56,1,21,26,26,0,2,24,75,0,0,0,0,0,0,72,66,
31    0,0,32,65,0,0,32,65,0,0,0,64,24,0,130,1,3,16,2,2,
32    24,129,0,6,16,20,3,24,86,105,116,101,115,115,101,32,40,107,109,47,
33    104,41,0,129,0,38,16,23,3,24,84,101,109,112,195,169,114,97,116,117,
34    114,101,32,40,194,176,67,41,0,130,1,35,16,2,2,24,66,128,37,23,
35    23,6,2,26,67,5,5,38,20,5,2,26,11,67,5,37,32,23,5,2,
36    26,11,70,17,4,53,14,6,26,37,0,130,1,3,48,2,2,24,129,0,
37    6,48,20,3,24,76,117,109,105,195,168,114,101,32,0,129,0,38,48,12,
38    3,24,112,111,105,100,115,32,40,107,103,41,0,130,1,35,48,2,2,24,
39    67,4,36,53,16,6,2,26,11,70,33,54,53,6,6,26,37,135,0,130,
40    1,3,65,2,2,24,129,0,6,65,12,3,24,108,111,99,97,108,105,115,
41    97,116,105,111,110,32,0,129,0,4,73,18,4,24,76,97,116,105,116,117,
42    100,101,32,32,32,58,0,129,0,4,80,19,4,24,108,111,110,103,105,116,
43    117,100,101,32,58,32,0,67,0,27,72,35,5,24,26,11,67,0,27,79,
44    35,5,24,26,11 };
45
```

Annexe

Programme de l'application

```
// this structure defines all the variables and events of your control interface
struct {

    // input variables
  uint8_t button_1; // =1 if button pressed, else =0

    // output variables
  float instrument_1;  // from 0 to 50
  int8_t level_1; // =0..100 level position
  char text_1[11];  // string UTF8 end zero
  char text_2[11];  // string UTF8 end zero
  uint8_t led_1; // led state 0 .. 1
  char text_3[11];  // string UTF8 end zero
  uint8_t led_2; // led state 0 .. 2
  char text_4[11];  // string UTF8 end zero
  char text_5[11];  // string UTF8 end zero

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

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

```
byte pinData = 4;
byte pinClk = 5;
HX711 poids;

TinyGPS gps;
SoftwareSerial ss(7, 6);


/*-----------------------------------------------------------------------
                    Configuration
-----------------------------------------------------------------------*/
void setup()
{
  RemoteXY_Init ();
#ifdef DEBUG_HX711
  Serial.begin(9600);
  Serial.println("le poids ");
#endif

  poids.begin(pinData, pinClk);
  poids.set_scale(calibration);
  poids.tare();
  ss.begin(4800);

}
/*-----------------------------------------------------------------------
                    Programme principale
-----------------------------------------------------------------------*/
void loop()
{
  RemoteXY_Handler ();
  vitesse();
  lumiere();
  temperature();
  poids();
  position();

  delay(500);
}

```

Annexe

Programme de l'application

```
/*-----------------------------------------------------------------------------
                    les fonctions
-----------------------------------------------------------------------------*/
void      vitesse()
{
  char str[] = " Km/h";
  int N=analogRead(A0);
  int V=map(N,0,1023,0,50);
  RemoteXY.instrument_1 = V;

  if(V<25)  sprintf (RemoteXY.text_1, "%d %s",V ,str);
  else      sprintf (RemoteXY.text_1, "limite !!!");

}
```

```
/*-----------------------------------------------------------------------------*/
void      lumiere()
{
  int N=analogRead(A1);
  int V=map(N,0,1023,0,100);


 if(V<10)  RemoteXY.led_1 = 1;
 else      RemoteXY.led_1 = 0;

}
/*-----------------------------------------------------------------------------*/
```

```
/*-----------------------------------------------------------------------------*/
void   temperature()
{ int T=0,NT=0;
  float K=0.48875855;
  char sr[] = " °C";
  NT=analogRead(A2);
  T=K*NT;
  sprintf (RemoteXY.text_2, "%d %s",T ,sr);
  RemoteXY.level_1 = T;

}
/*-----------------------------------------------------------------------------*/
```

```
/*-----------------------------------------------------------------------------*/
void   poids()
{
  #ifdef DEBUG_HX711
    float N=poids.get_units();
    float M=-0.0763636*N;    //étalonnage (0.336 ---> -4.4)
  #endif
      sprintf (RemoteXY.text_1, "%d %s",M ,"Kg")

  if(M<5)  RemoteXY.led_2=2;
  else     RemoteXY.led_2=1;
}
/*-----------------------------------------------------------------------------*/
```

Annexe

Programme de l'application

```
/*-----------------------------------------------------------------------------*/
void position()
{
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;

  // For one second we parse GPS data and report some key values
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }

  if (newData)
  {
    float flat, flon;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);

    sprintf (RemoteXY.text_4, "%f", flat);
    sprintf (RemoteXY.text_5, "%f", flon);
  }

}
/*-----------------------------------------------------------------------------*/
```