

Self balancing Robot

Projet LABVIEW

*Filière : Informatique industrielle et
automatique*

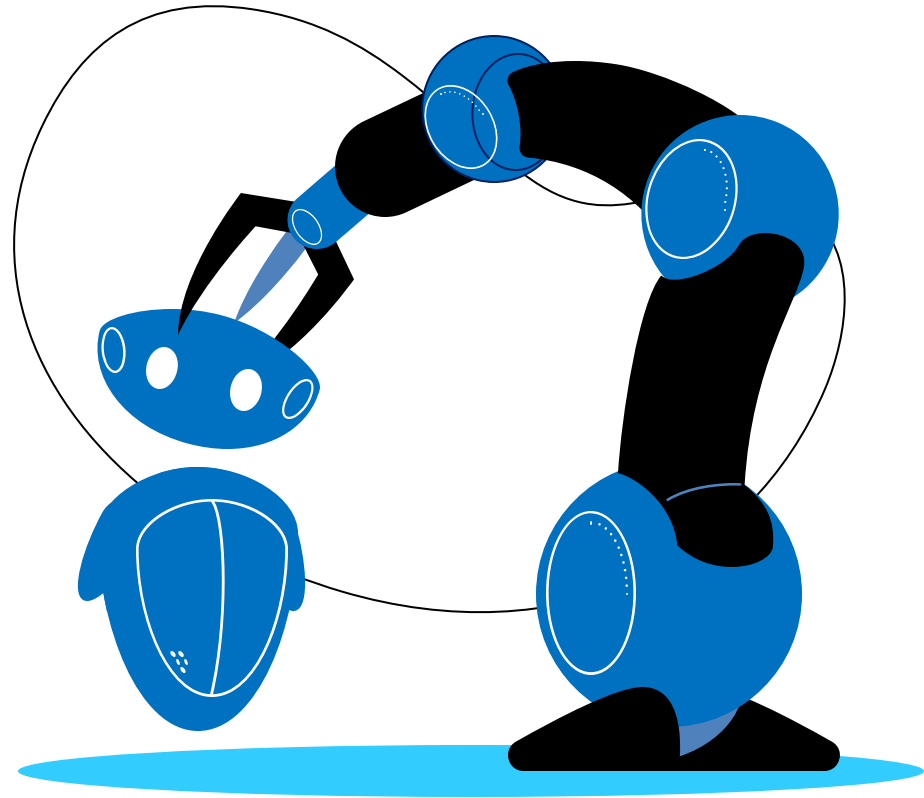
présentée par :

BARGAOUI NOURCHENE

NASRI MOHAMED YOUSSEF

KOUBAA MOHAMED YASSINE

FERSI YOSRI





INTRODUCTION

ETUDE THEORIQUE

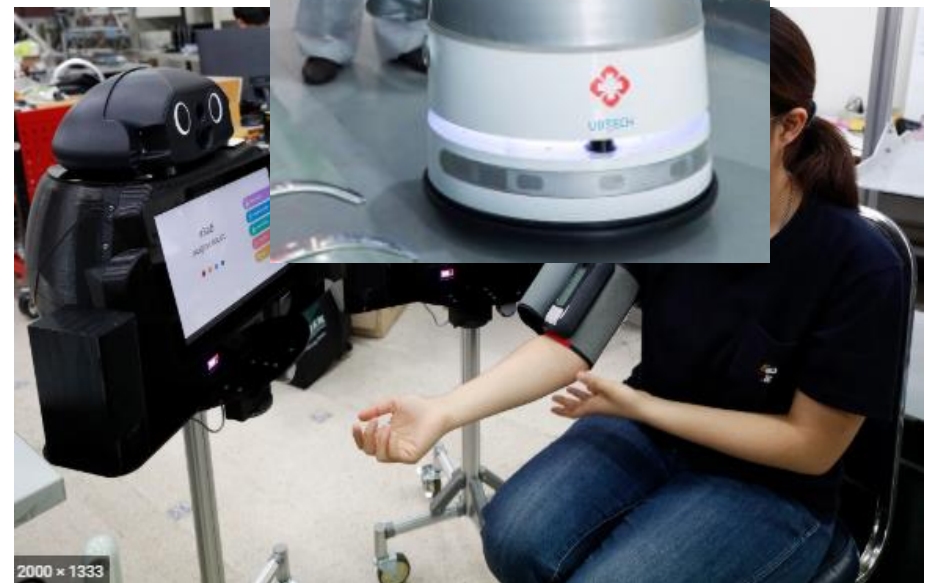
CONCEPTION

ETUDE SUR LABVIEW

SIMULATION

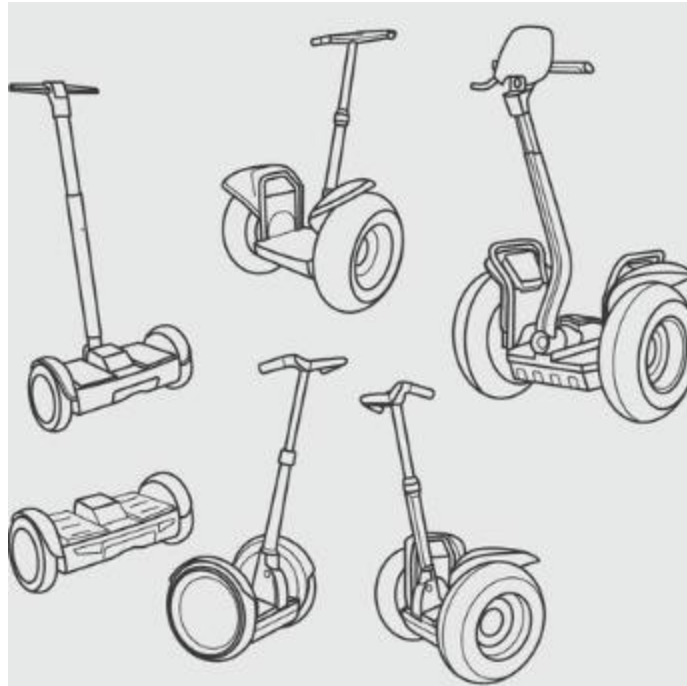
CONCLUSION

“ Le domaine de la robotique est le terrain de jeu des esprits créatifs de l'ère moderne. Les rêves sont devenus réalité avec le développement Des robots dans ce domaine qui ont été mis en service pour effectuer plusieurs taches ”





On s'est inspiré de : Segway Robot

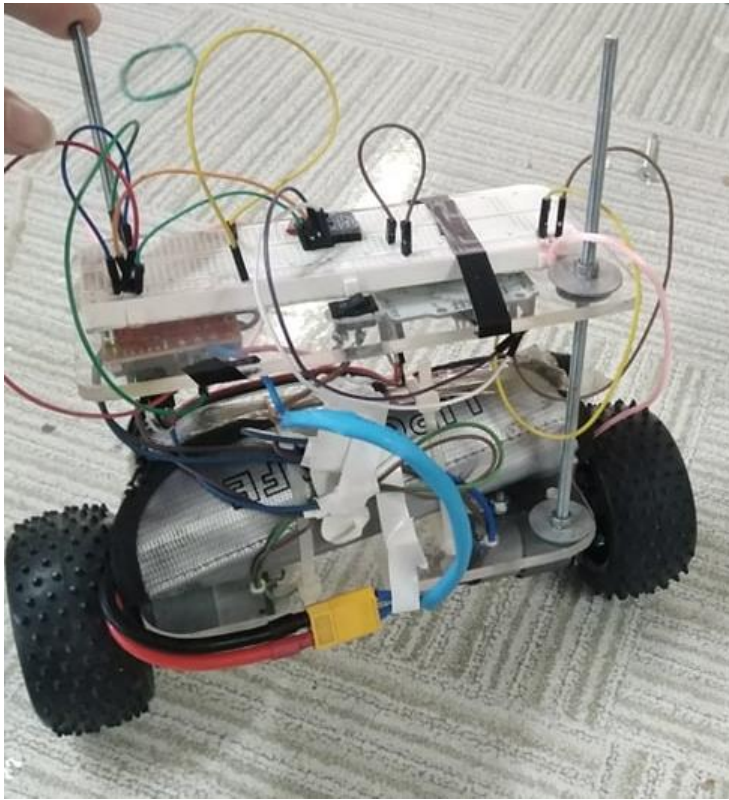




Vidéo :



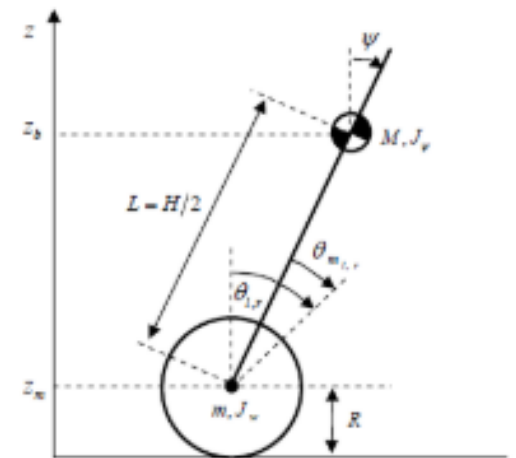
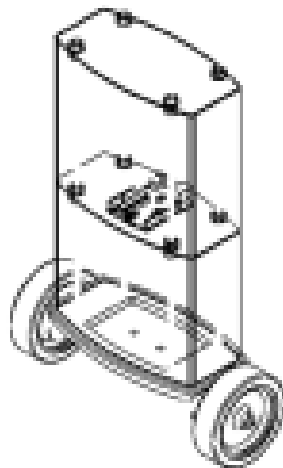
Notre projet :





Buts

- Equilibrer un robot balanceur sur deux roues
- Concevoir un système de contrôle numérique complet avec Labview



Le robot auto-équilibrant
est basé sur le problème
du pendule inversé



Valeur
théorique et
stabilité

Introduction

ETUDE
THEORIQUE

CONCEPTION

ETUDE SUR
LABVIEW

SIMULATION

CONCLUSION

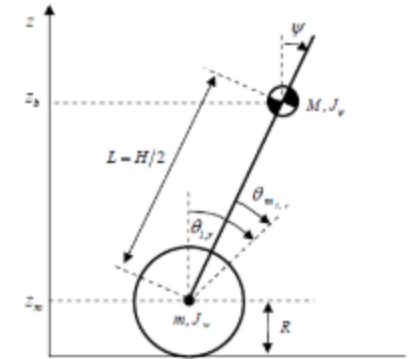
- En supposant le système de pendule inversé où une masse **M** est reliée par une tige sans masse de longueur **L** à un point sans frottement. La vitesse angulaire **w** ainsi que sa variation au cours du temps **dw / dt** sont données par la formule suivante :

$$\frac{d\theta}{dt} = \omega$$

$$\frac{d\omega}{dt} = -\frac{MgL}{J} \sin \theta \quad *$$

J : moment d'inertie
g : constante gravitationnelle

- A l'équilibre on a : **w = 0** and $\frac{dw}{dt} = 0$



➡ La resolution de l'équation (*) donne : $\dot{\theta} = 0$, and $\theta_1 = 0$ or $\theta_2 = 180^\circ$.

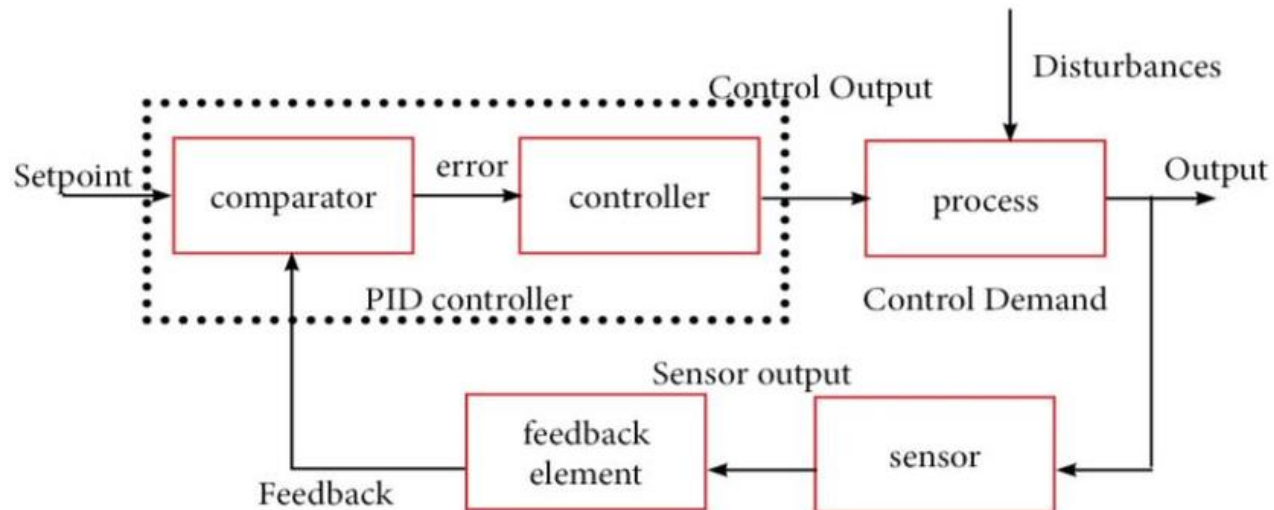
➡ θ_1 est stable mais ne s'applique pas aux robots auto-équilibrés,
 θ_2 est la position cible pour le pendule inversé.



Instabilité du système :

Toute perturbation externe fera que le pendule s'éloignera indéfiniment de ce point d'équilibre spécifique, d'où la nécessité qu'il soit activement équilibré.

➔ Bloc PID





3. Conception

1. Conception
mécanique

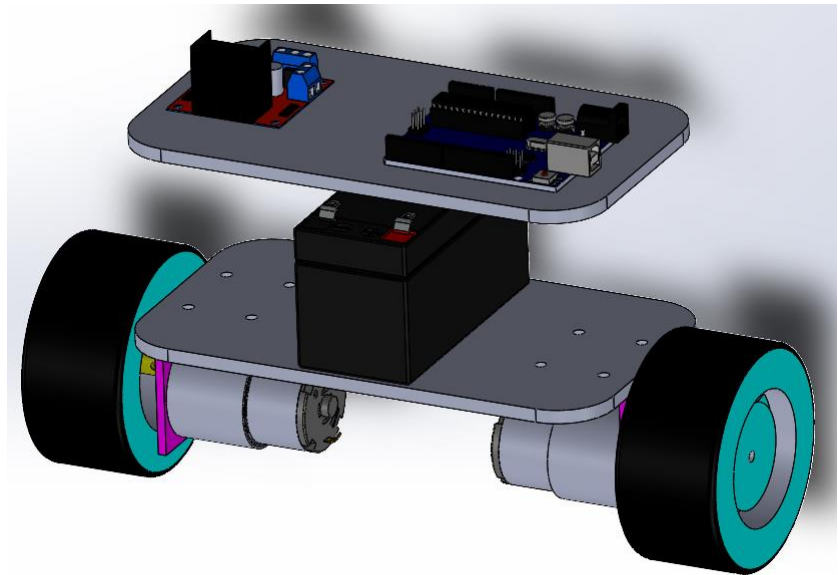
2. Conception électrique

3. Programmation










1. Conception mécanique

- Arduino UNO
- Geared DC motors
- L298N Motor Driver Module
- MPU6050 Gyroscope
- Une paire de Roue
- Batterie
- Fils de connexion
- Châssis





Composants existant		Composant choisi	Pourquoi ?
 Moteur avec encodeur	 Encodeur	 Moteur jaune	<ul style="list-style-type: none"> • Le moteur jaune est moins cher • Le moteur jaune prend moins de place qu'un moteur avec encodeur • Le PID avec moteur et encodeur est plus compliqué
 Arduino UNO	 STM32 Discovery	 Arduino MEGA	
 Moteur jaune			<ul style="list-style-type: none"> • Côté performance, la carte STM32 semble la meilleure mais selon notre application et le prix de chaque carte, l'arduino UNO semble la plus optimale (Arduino UNO coûte 22\$, Arduino MEGA coûte 35\$ alors que STM32 coûte 30\$) • De plus la carte Arduino UNO nous l'avons déjà



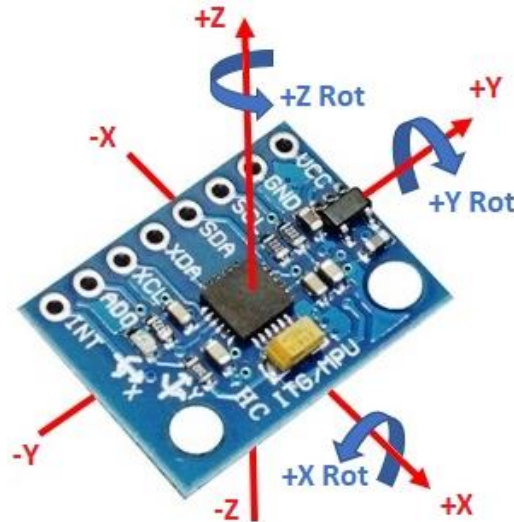
 <p>Gyroscope avec accéléromètre</p>	 <p>Gyroscope</p>	 <p>Gyroscope avec accéléromètre</p>	<ul style="list-style-type: none"> • Gyroscope avec accéléromètre nous permet de jouer sur la vitesse pour mieux améliorer le contrôle du robot • Quand on compare le rapport prix / performances le MPU 6050 est le meilleur gyroscope (il ne coûte que 3\$)
 <p>Batterie Lion *2</p>	 <p>Batterie plomb-acide</p>	 <p>Batterie Lion *2</p>	<ul style="list-style-type: none"> • La batterie plomb-acide est plus lourde que la batterie Lion, elle peut donc endommager l'équilibre de notre robot • La batterie Lion est moins chère que la batterie plomb-acide
 <p>L298 Driver</p>	 <p>Arduino Shield</p>	 <p>L298 Driver</p>	<ul style="list-style-type: none"> • L298 ne coûte que 4\$ et nous donne beaucoup de flexibilité pour câbler notre composant • Nous l'avons déjà



The **MPU6050** module

Accéléromètre:

Mesure l'accélération sur les 3 axes

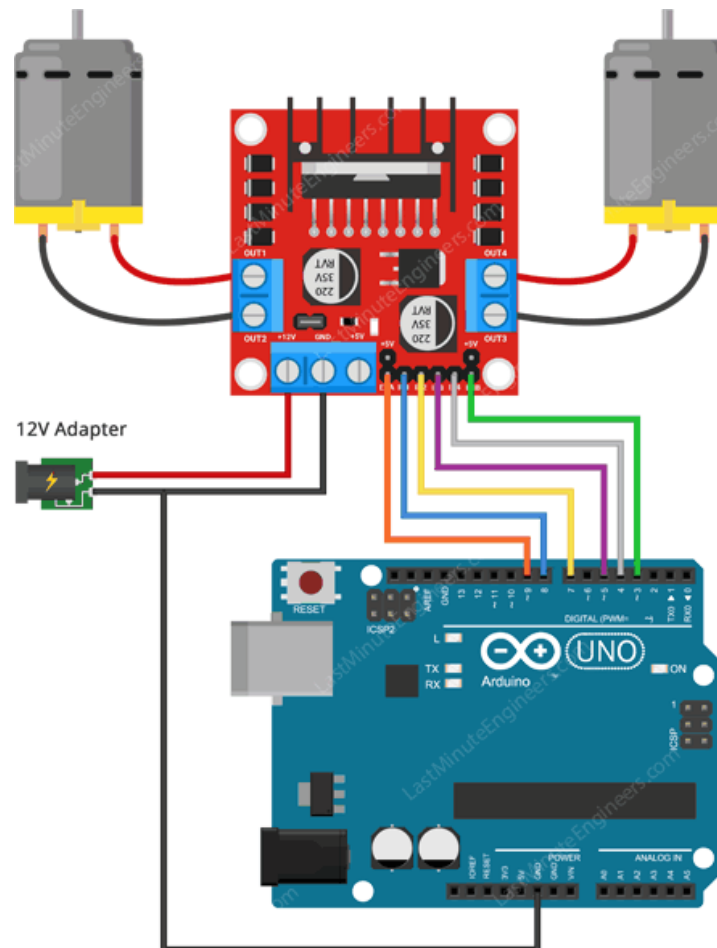


The **gyroscope**: mesure la vitesse angulaire sur les 3 axes

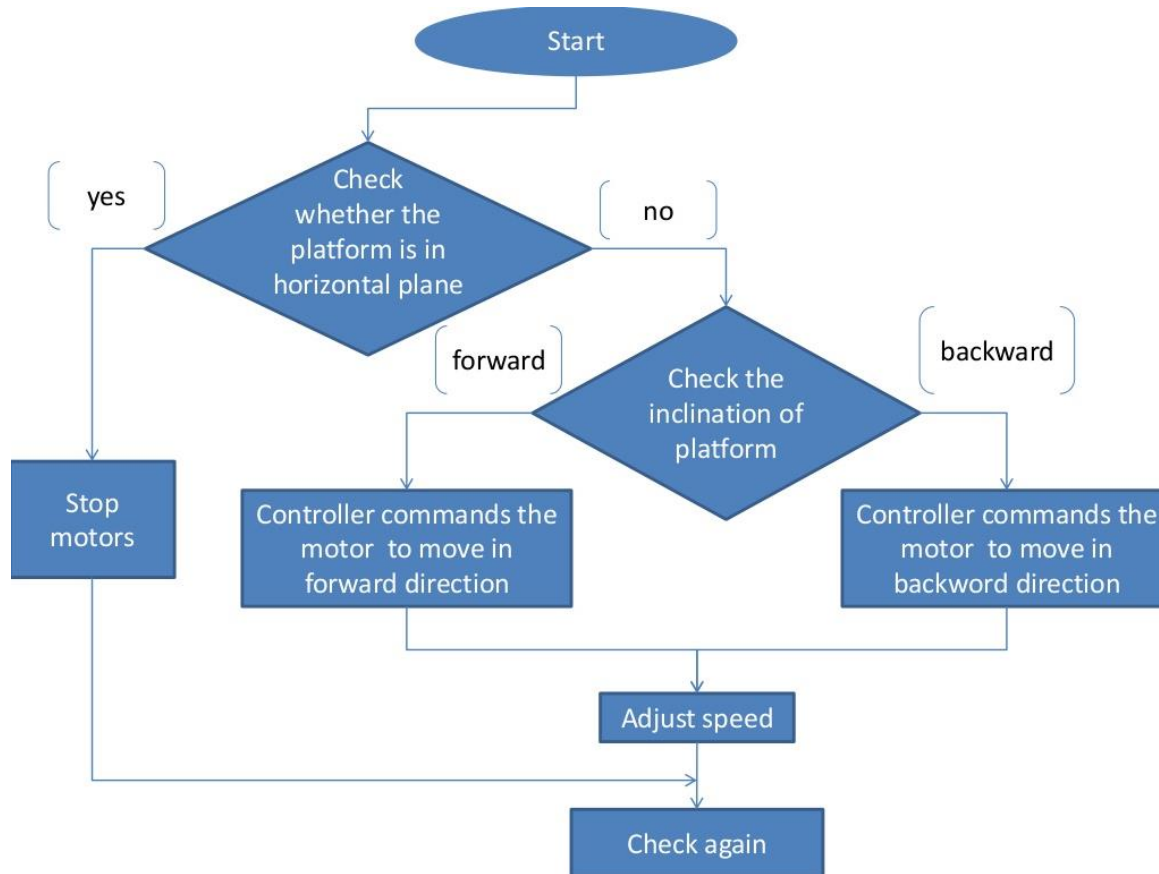
MPU6050 est une petite carte électronique équipée d'un gyroscope et d'un accéléromètre



2. Conception électrique



Organigramme





2. Programmation

1. Mesurer l'angle de l'inclinaison à l'aide du gyroscope

2. Calcul de l'erreur

3. Fonction PID



The MPU6050 library
sert à lire les données
depuis le MPU6050.

WIRE est utilisé pour la
communication avec
MPU6050

```
#include <MPU6050_tockn.h>  
#include <Wire.h>
```



1. Mesurer l'angle d'inclinaison à l'aide du gyroscope

```
void Calc_Error() {  
    mpu6050.update();  
    errorp=kp*(mpu6050.getAngleY()-Error0);  
    errord=kd*(errorp-previousError);  
    if(errorp*previousError<0) errori=0;  
    else errori+=ki*errorp;  
    previousError=errorp;  
    Serial.println((int) ((mpu6050.getAngleY()-Error0)*100));  
}
```




2. Calcul de l'erreur

```
void Calc_Error() {  
    mpu6050.update();  
    errorp=kp*(mpu6050.getAngleY()-Error0);  
    errord=kd*(errorp-previousError);  
    if(errorp*previousError<0) errori=0;  
    else errori+=ki*errorp;  
    previousError=errorp;  
    Serial.println((int)((mpu6050.getAngleY()-Error0)*100));  
}
```

```
void loop() {  
    if ( millis() % 10 < 2 ) Calc_Error();  
    Pid_control(errorp,errori,0);  
    //Calc_Error();  
    //Serial.println(mpu6050.getAngleY());  
    //delay(100);  
  
}
```



3. Fonction PID

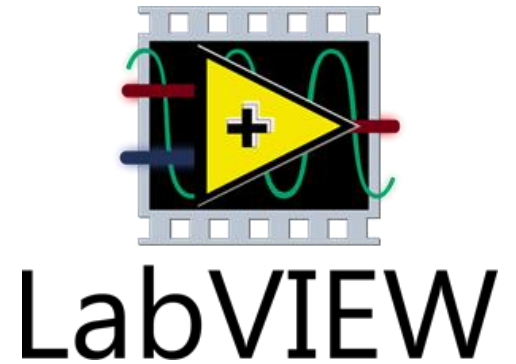
```
void Pid_control(float P,float i,float d){  
  if(P>0){  
    analogWrite(in1,0);  
    analogWrite(in2,constrain(abs(P+i-d),0,255));  
    analogWrite(in3,constrain(abs(P+i-d),0,255));  
    analogWrite(in4,0);  
  }  
  else {  
    analogWrite(in1,constrain(abs(P+i-d),0,255));  
    analogWrite(in2,0);  
    analogWrite(in3,0);  
    analogWrite(in4,constrain(abs(P+i-d),0,255));  
  }
```

```
void loop(){  
  if ( millis() % 10 <2 ) Calc_Error();  
  Pid_control(errorp,errori,0);  
  //Calc_Error();  
  //Serial.println(mpu6050.getAngleY());  
  //delay(100);  
}
```



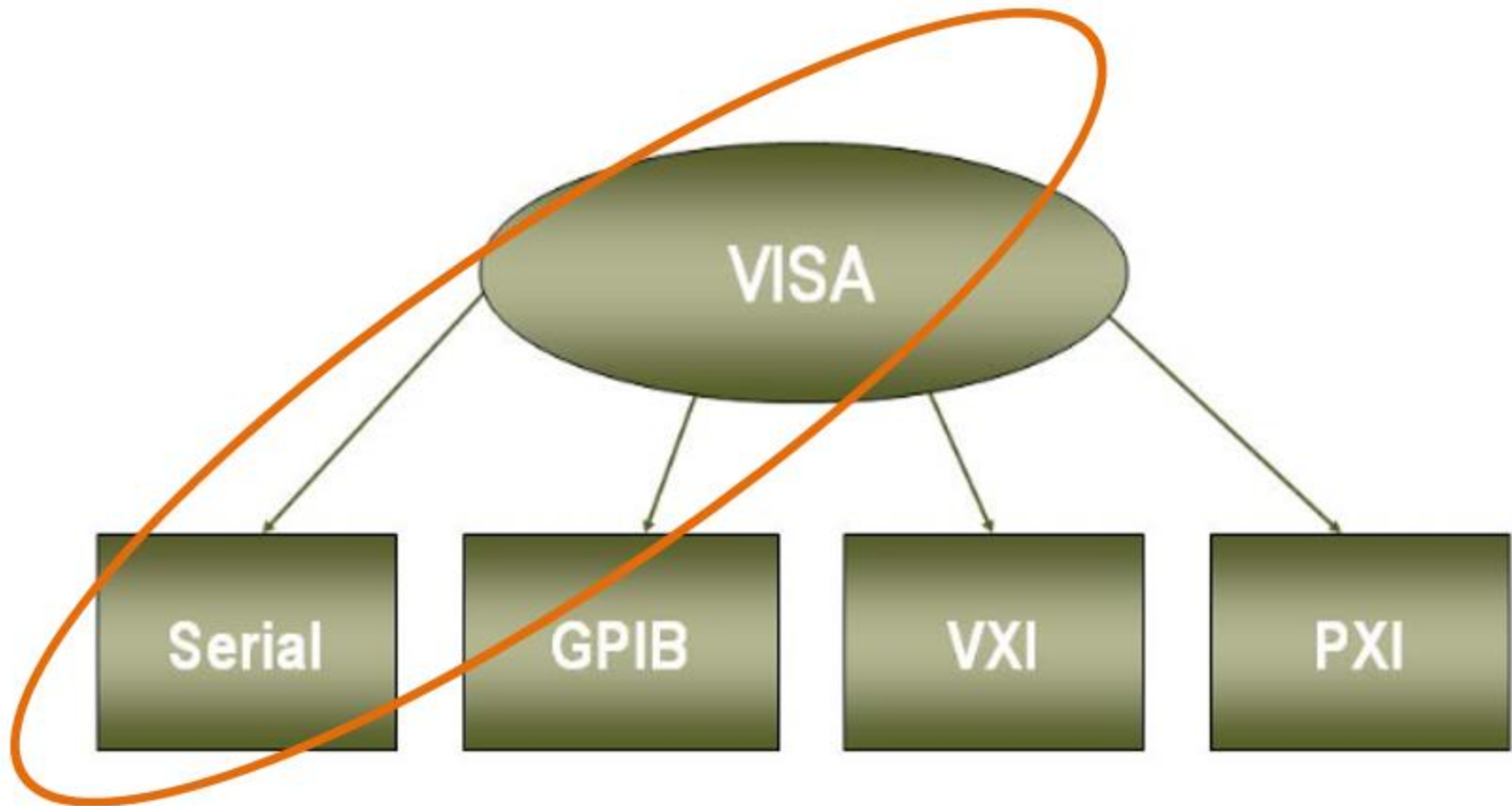
Process de communication

Nous allons simuler le robot par communication entre Arduino UNO et LABVIEW





Labview commandes dashboard





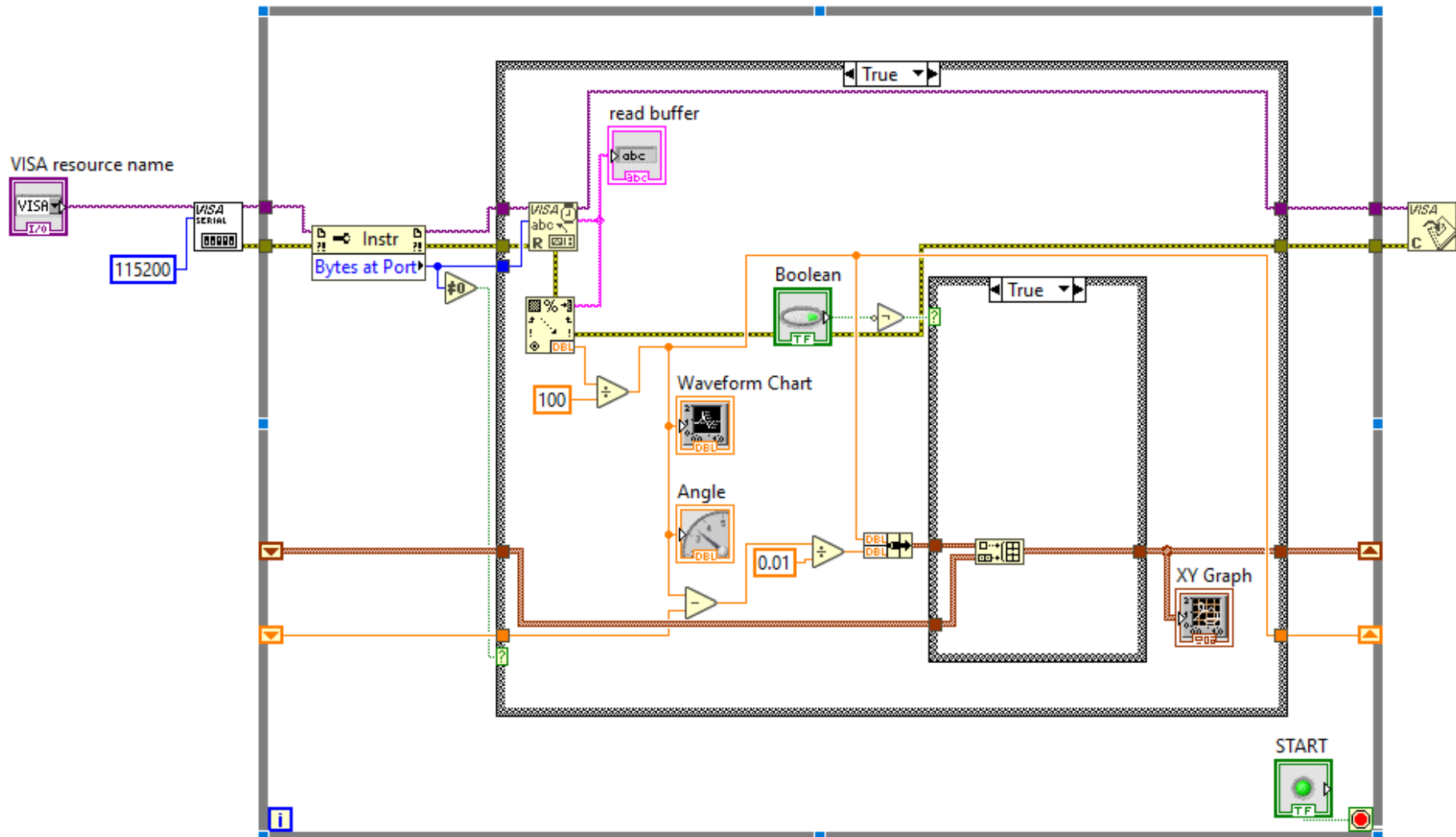
initiating the communication with the **VISA resource name(COM1)** witch is connected to the serial port of ARDUINO and returns a session identifier that can be used to call any other operations of that device.



Reads the "distance " from the port specified by **VISA resource name** and returns the data in **read buffer**.



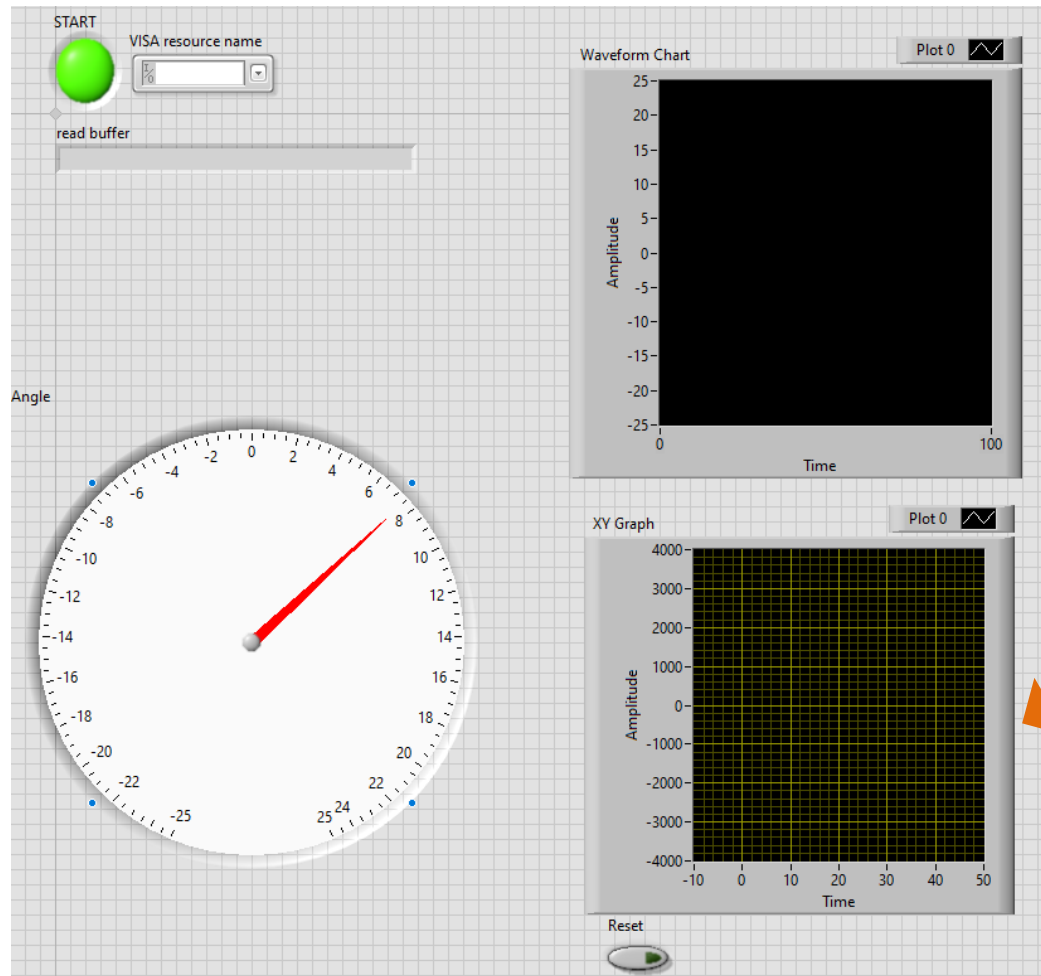
Writes an alphabetic character from **write buffer** to the Serial port of the ARDUINO board to initiate one of the robot's programmed movements





Labview Front panel

ON/OFF
button



Error char

Display the
error



phase diagram



Vidéo :

File Edit View Project Operate Tools Window Help



START



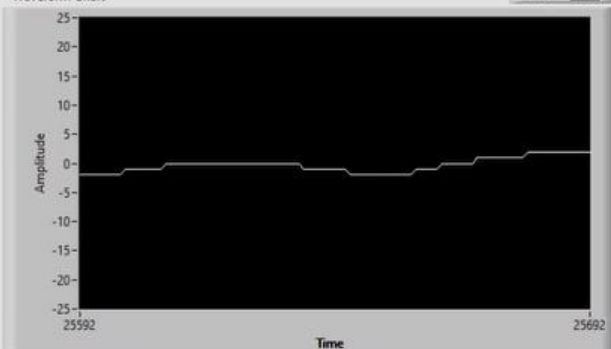
VISA resource name

COM24

read buffer

2

Waveform Chart



Angle



File Edit View Project Operate Tools Window Help



VISA resource name

115200

VISA

INSTR

Bytes at Port

40

True

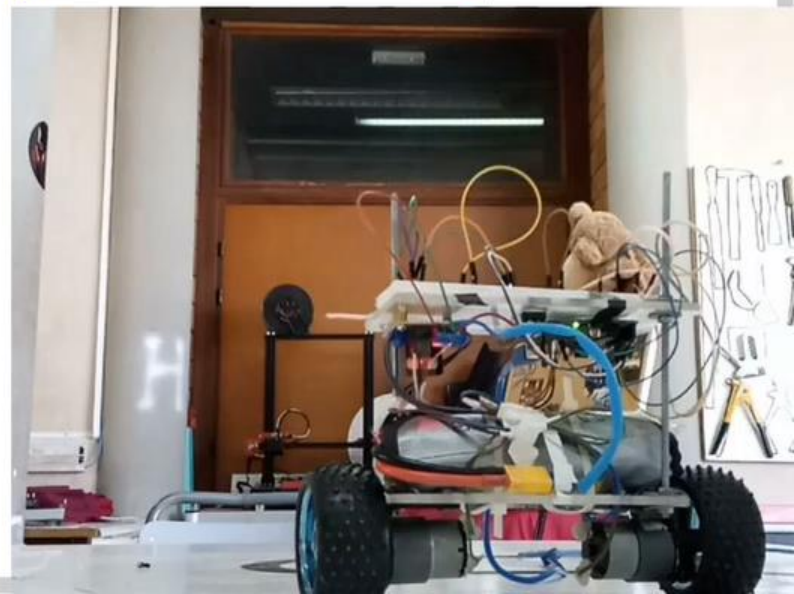
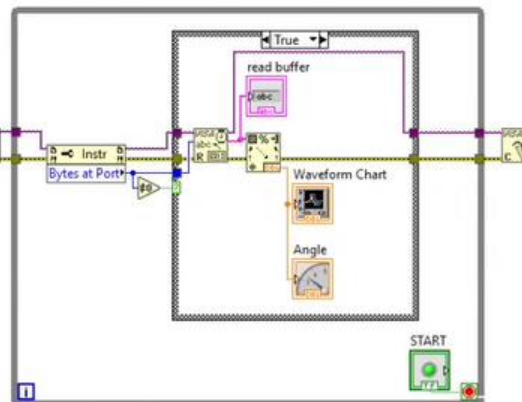
read buffer

Waveform Chart

Angle

START

115200



Introduction

ETUDE
THEORIQUE

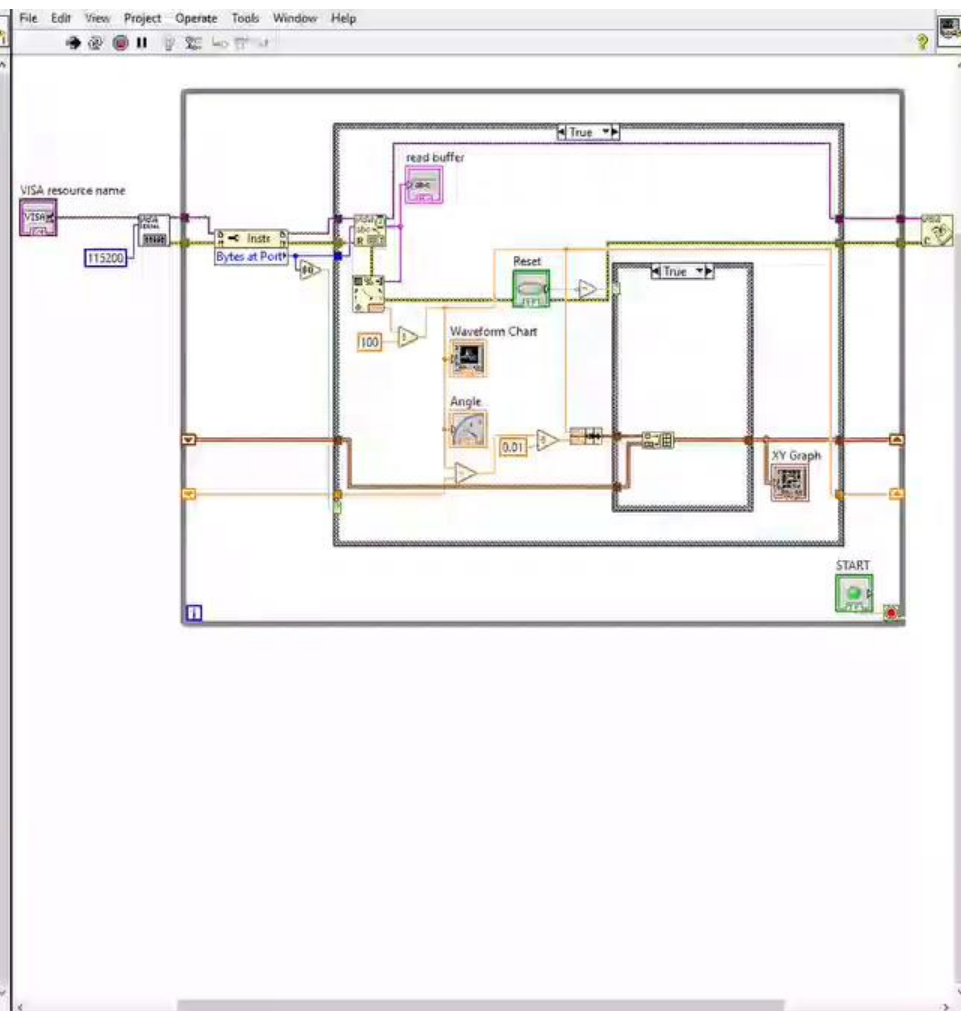
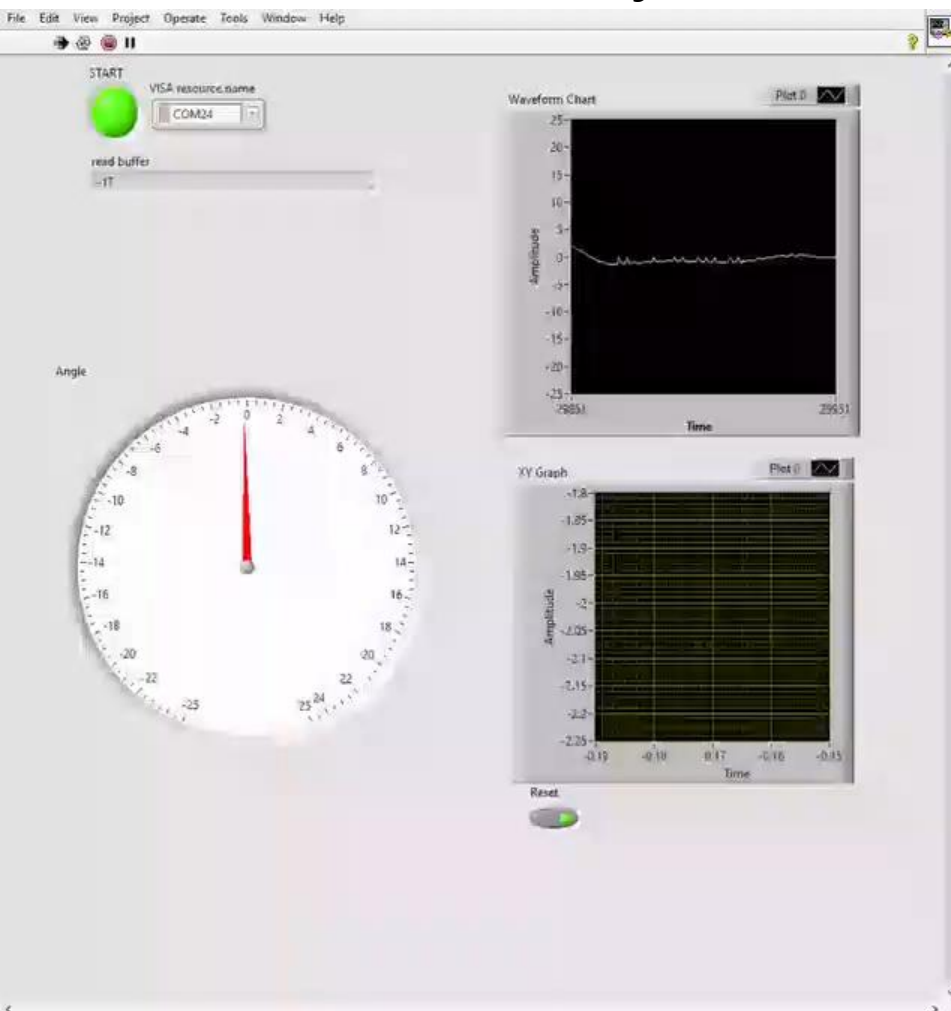
CONCEPTION

ETUDE SUR
LABVIEW

SIMULATION

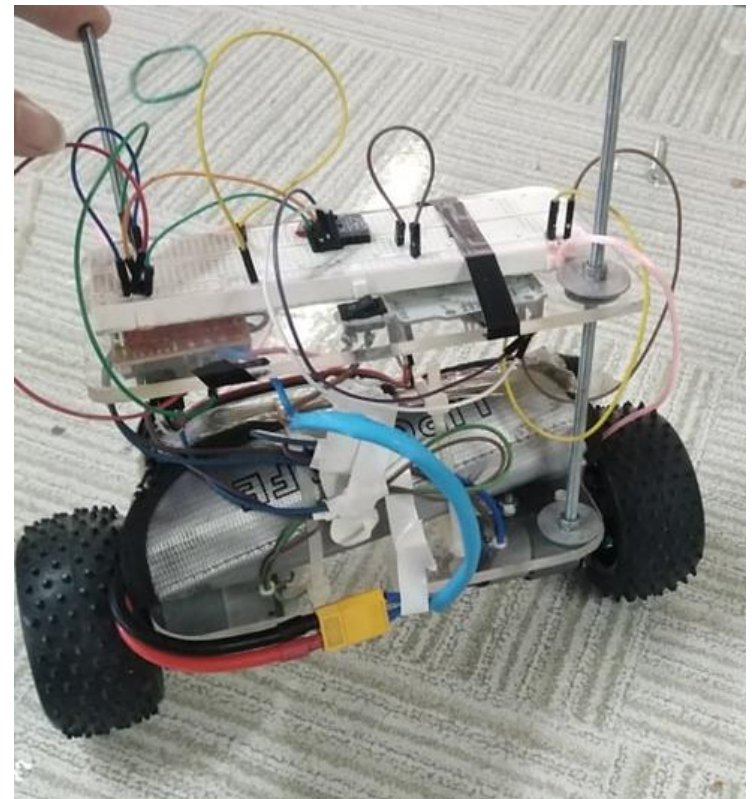
CONCLUSION

Vidéo démonstration cycle limite :





On a travaillé sur un robot balanceur, on a fait l'acquisition de donnée ainsi que travailler sur l'équilibre du système en boucle fermé à l'aide d'un PID





**Merci Pour votre
attention**