



Projet Tutoré - Grand Oral

Automatisation du positionnement d'une parabole

2ème année DUT GEII – Groupe Robotic1

Lise Chauvin — ATS1

Gilles Devillers — ATS1

Alexandre Minot — MOP2

Thomas Grageon — MOP3

Groupe Robotic1

Coach : Vincent Grimaud
Référent professionnel : Christophe Tailliez

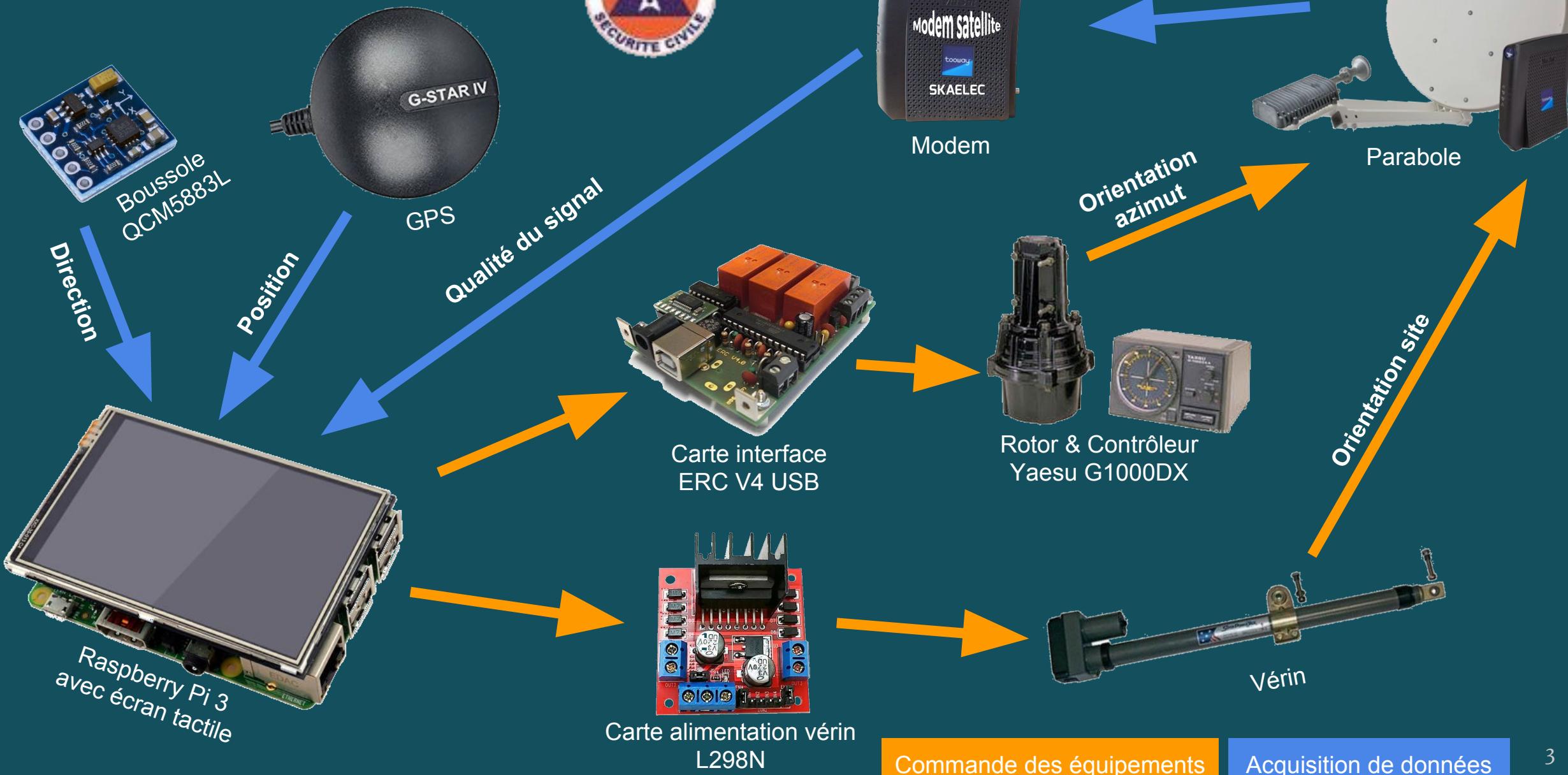
Jury communication : Sylvie Aubert
Jury technique : Jean-Luc Padiolleau

Sommaire

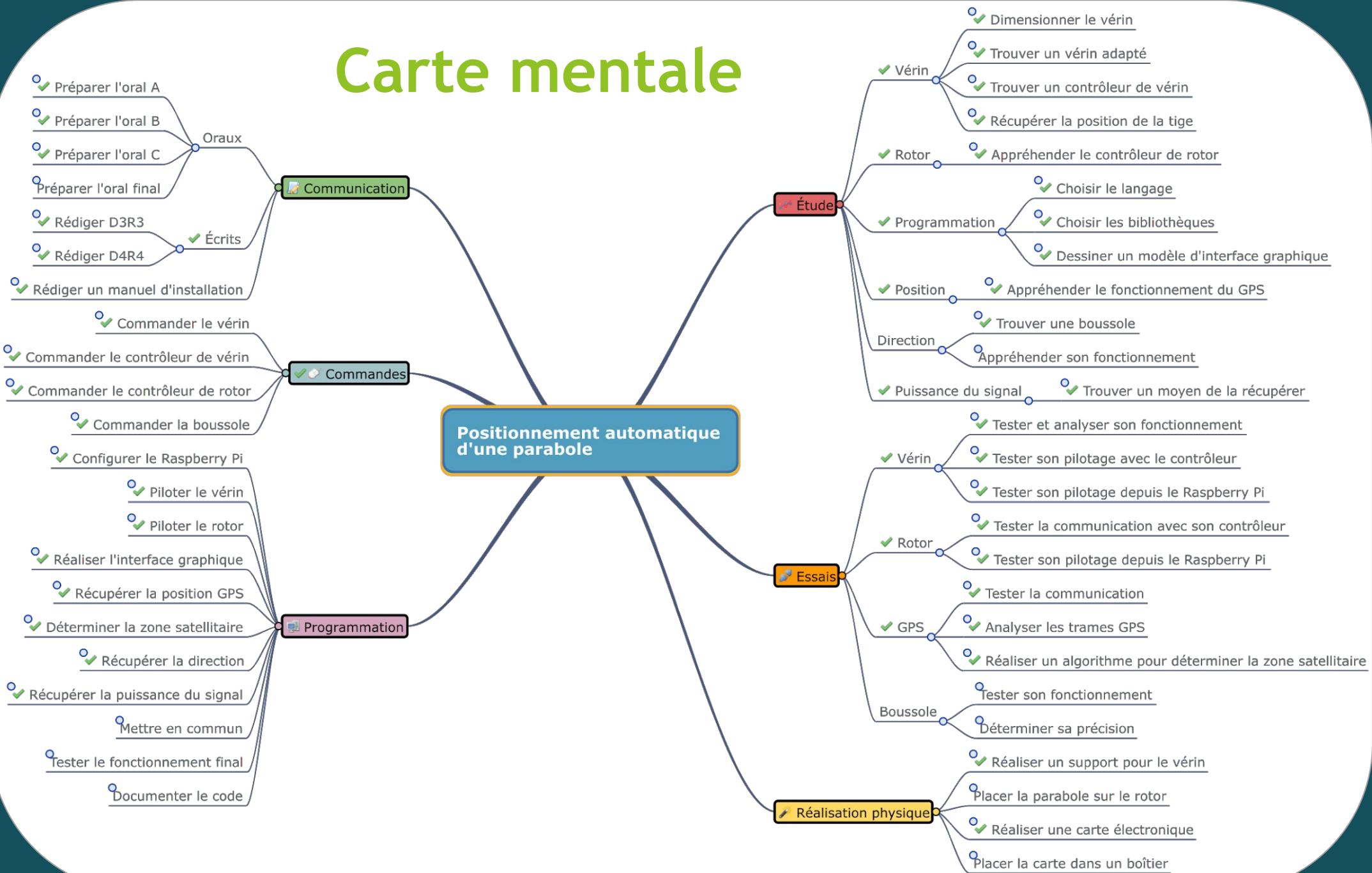
- Introduction
- Organisation
- Acquisition
- Commande
- Coordination
- Démonstration
- Conclusion



Introduction



Carte mentale



Répartition des tâches

Tâches	Lise	Gilles	Thomas	Alexandre
Choix des solutions techniques	X	X	X	X
Acquisition de la position et zone de réception à l'aide du GPS	X	X		
Acquisition du cap à l'aide de la boussole	X		X	
Acquisition de la puissance du signal satellite		X		
Rotor (recherches, essais, programmation)				X
Vérin (commandes, recherches, essais, programmation)			X	X
Réalisation du circuit électronique	X	X		
Réalisation du boîtier	X	X		
Configuration du Raspberry Pi et rédaction du mode d'emploi	X			
Conception et développement de l'interface graphique	X			
Intégration globale des programmes	X			
Positionnement automatique (algorithmes, programmation, essais)	X			

Budget

Matériel fourni



Matériel choisi
et commandé

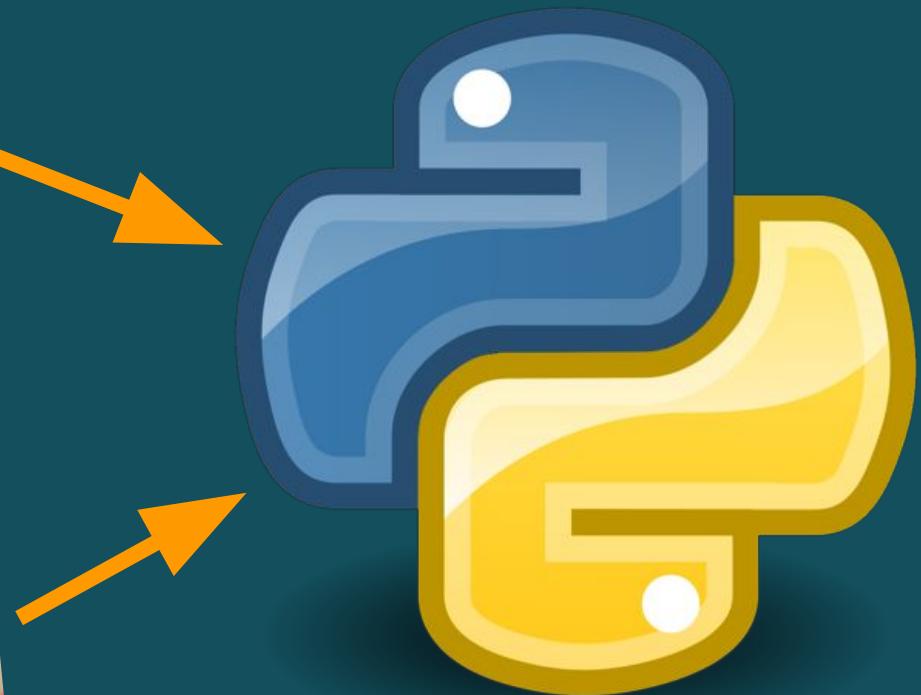
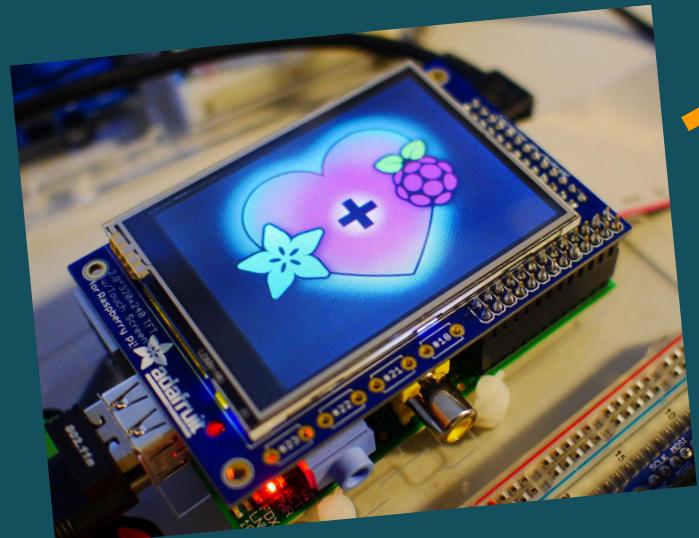


Désignation	Prix TTC
Raspberry Pi 3	35,00 €
Alimentation Raspberry Pi	10,00 €
Carte MicroSD Kingston 16 Go	8,00 €
ERC V4 USB	110,00 €
Écran tactile TFT 3,5"	37,00 €
Rotor Yaesu G-1000DX & Contrôleur	490,00 €
Câble Ethernet RJ45	5,00 €
Kit satellite Tooway (parabole, modem, support, activation)	375,00 €
Abonnement Tooway 25	120 €/mois
Vérin SuperJack III	45,00 €
Contrôleur vérin L298N	6,00 €
Boussole HMC5883L	4,00 €
Total	1545 €

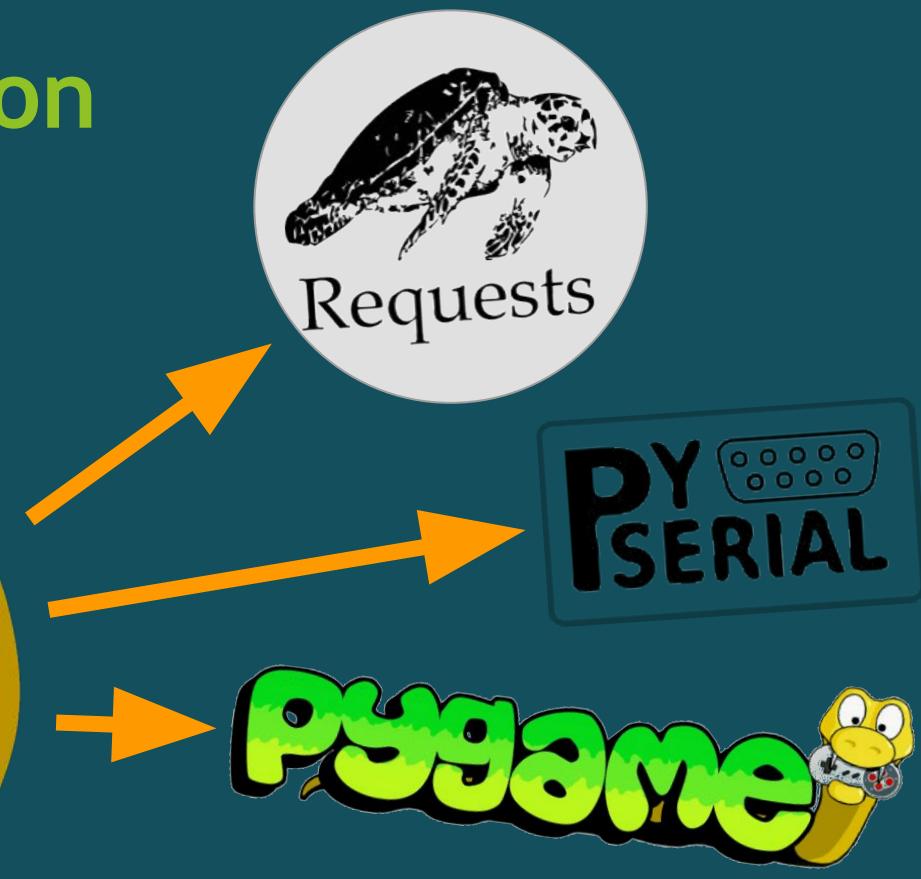
Choix du langage de programmation



Le cerveau du système :
Raspberry Pi



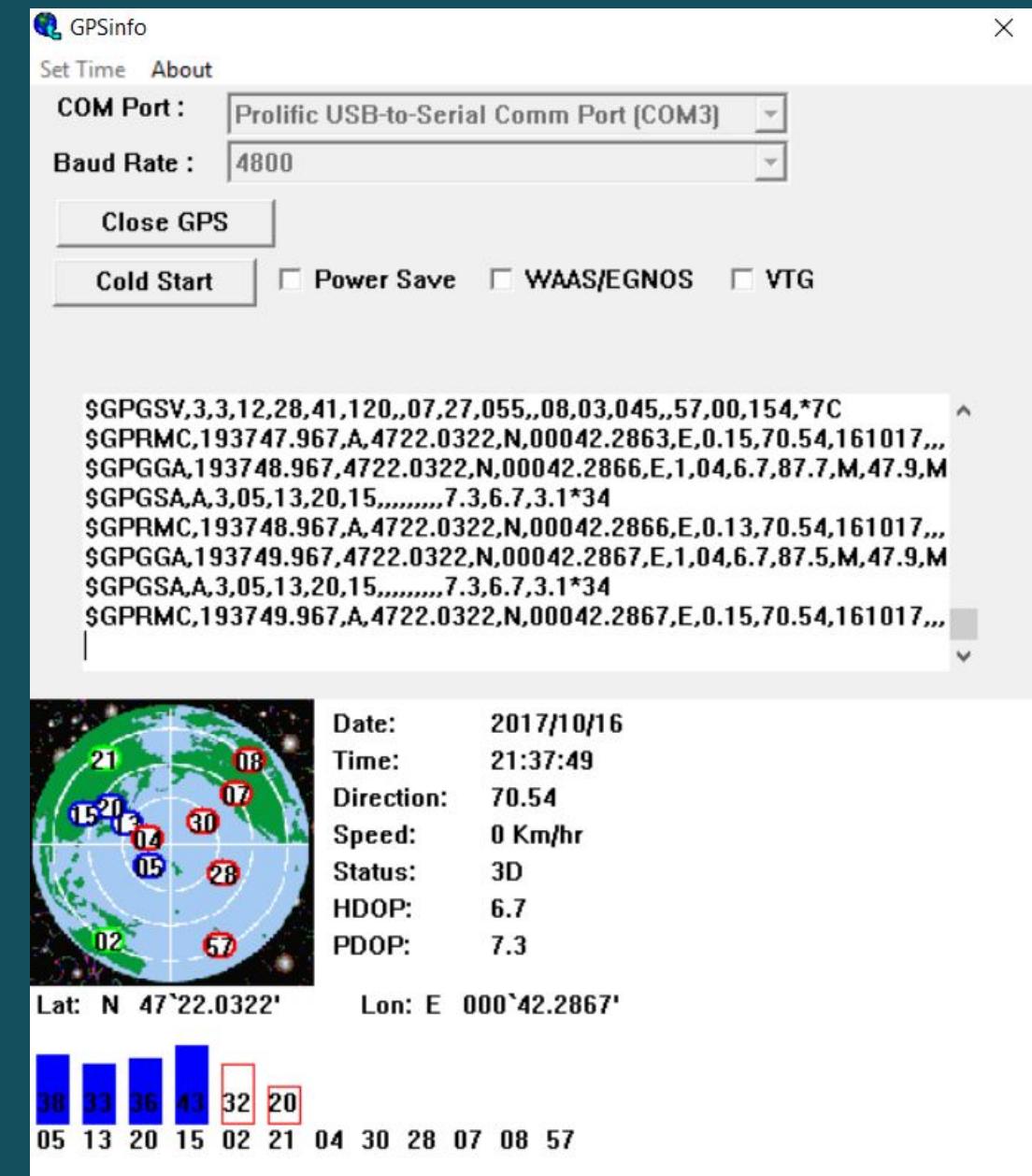
Python
Un langage commun



- Interprété
- Haut niveau
- POO (Programmation Orientée Objet)
- Communauté très importante
- Nombreuses bibliothèques

Acquisition

Analyse des trames GPS NMEA



Acquisition

Récupération des coordonnées GPS

- Connexion du GPS par USB
- Communication série 4800 baud
- Bibliothèque Pyserial pour communiquer avec le GPS
- Bibliothèque Pynmea2 pour décoder et exploiter les trames NMEA



```
1 class GPS(object):
2     def __init__(self, path):
3         try:
4             self.gps = serial.Serial(path, 4800)
5             self.gps.readline() # On se place au début d'une trame
6         except serial.SerialException: # Le GPS n'est pas connecté
7             self.gps = None
8
9         self.data = None
10
11    def __del__(self):
12        if self.gps is not None:
13            self.gps.close()
14
15    def getInfos(self):
16        if self.gps is not None:
17            while self.gps.in_waiting > 0:
18                serialBuffer = self.gps.readline()
19                if serialBuffer[0:6] == "$GPGGA":
20                    self.data = pynmea2.parse(serialBuffer)
21
22        return self.data
23
24        else:
25            return None
```

```
3 import serial
4 import pynmea2
5 import string
6
7 gps = serial.Serial('/dev/ttyUSB0', 4800)
8 print(gps.name)
9
10 while True:
11     nmea = gps.readline()
12
13     if nmea[0:6] == "$GPGGA":
14         gpsData = pynmea2.parse(nmea)
15         print("Latitude : " + "%02d°%02d'%.4f'" % (gpsData.latitude,
16         print("Longitude : " + "%02d°%02d'%.4f'" % (gpsData.longitude,
17         print("Fix" if gpsData.gps_qual else "No fix")
18         print("Sats : " + str(gpsData.num_sats))
19         print("")
20
21 gps.close()
22
```

Classe du GPS

Programme de test 9

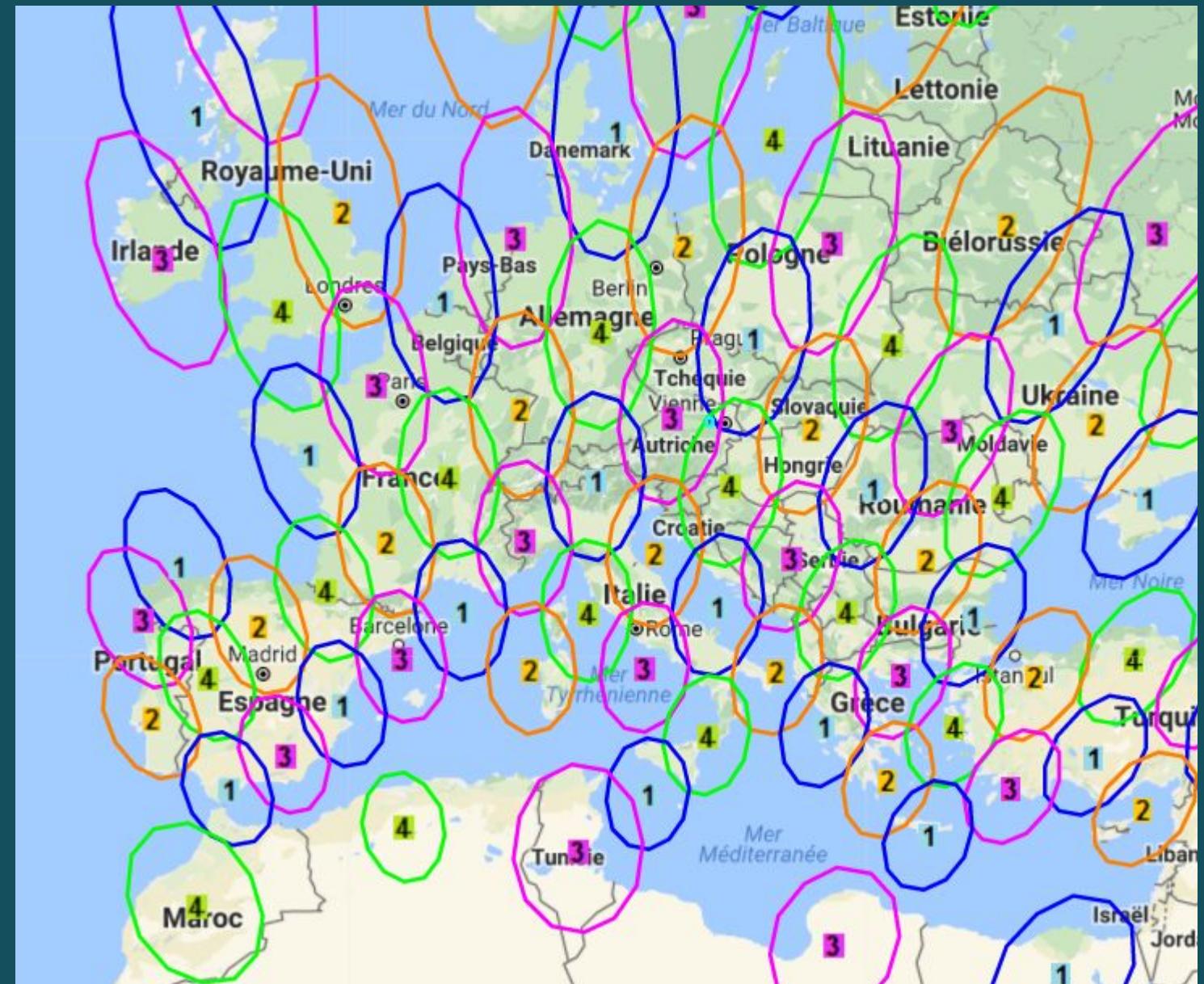
Acquisition

Détermination de la zone

Carte des satellites Ka-Sat en Europe

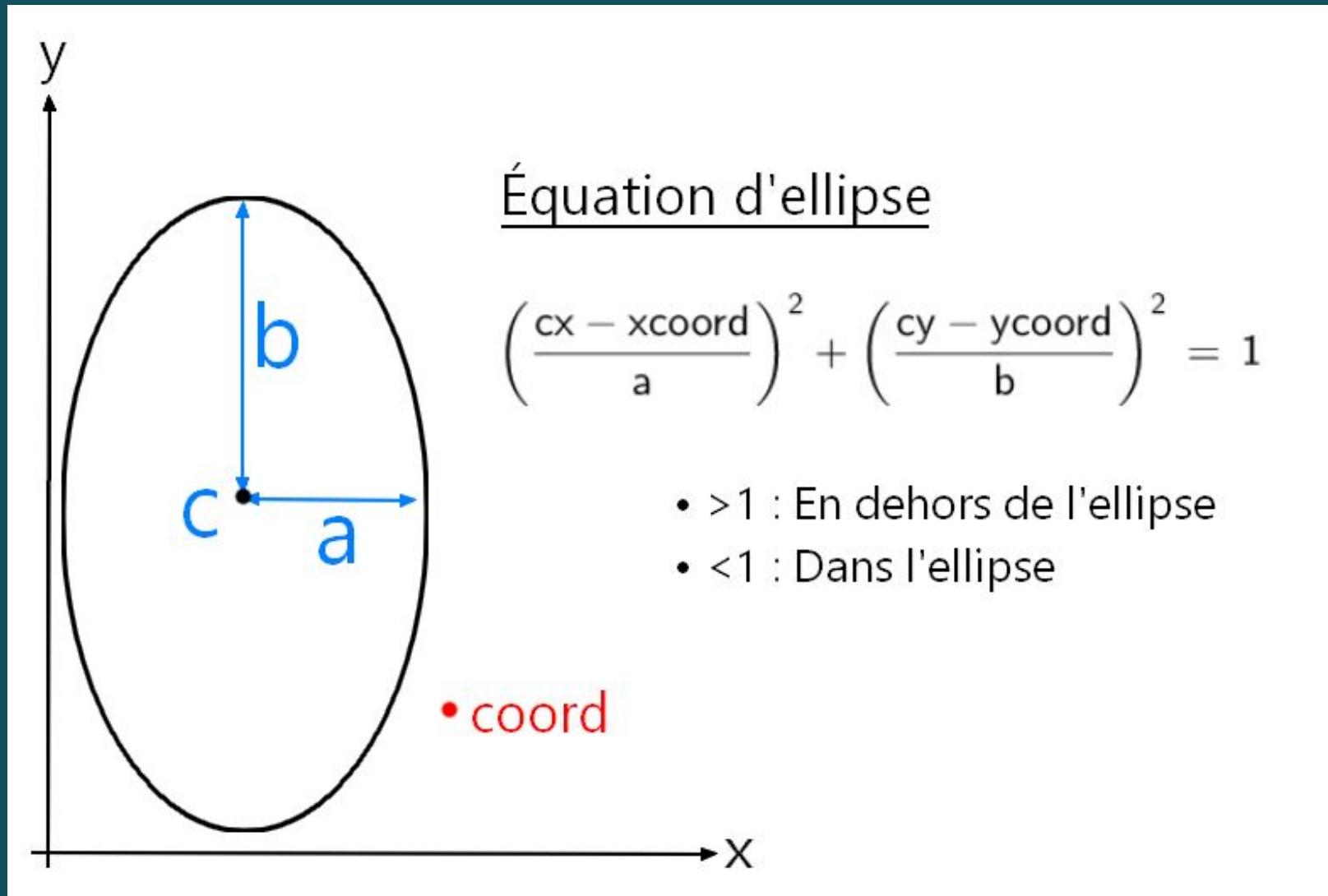
- 4 Types de zones
- 82 Spots
- Identifier le spot et la zone

Capture d'écran du site satsig.net



Acquisition

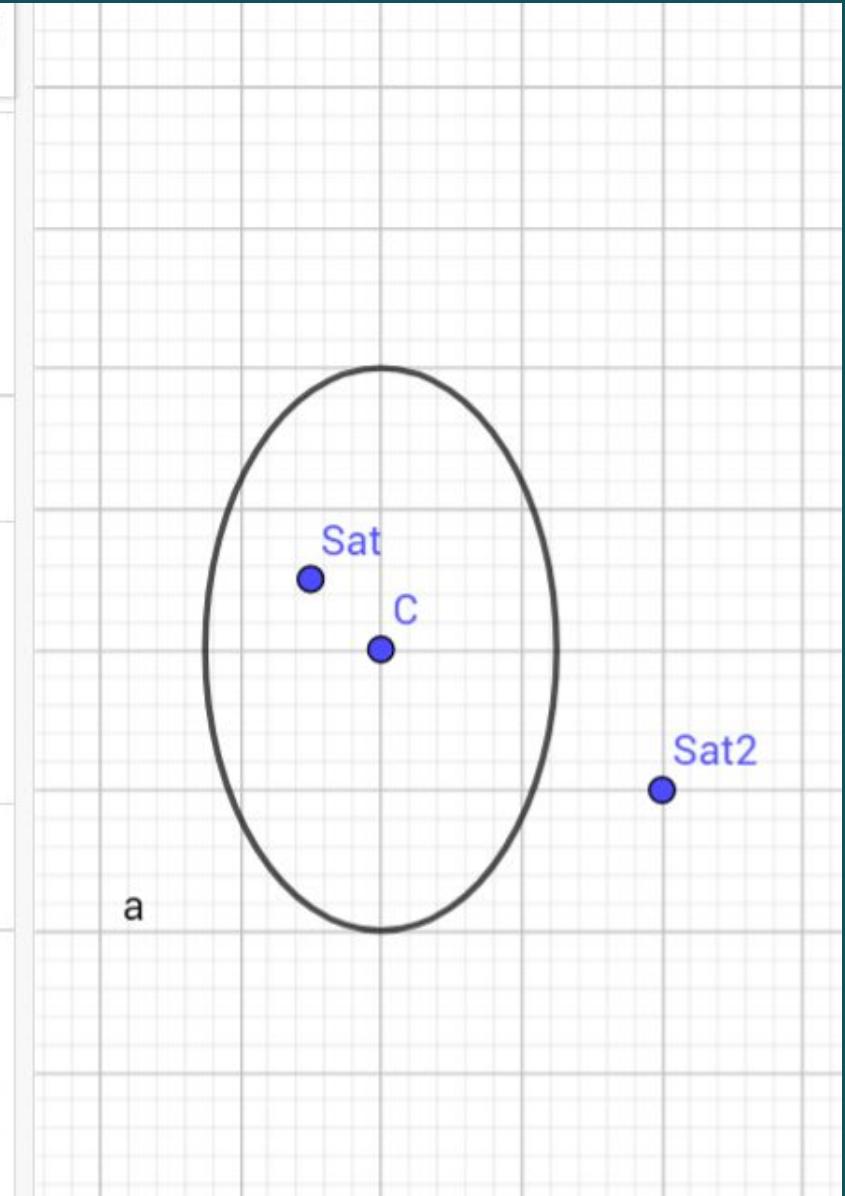
*Modèle géométrique
de l'ellipse*



Acquisition

Démonstration visuelle

●	$C = (2, 40)$	EN
●	$a : \left(\frac{2-x}{2.5}\right)^2 + \left(\frac{40-y}{4}\right)^2 = 1$	⋮
●	$Sat = (1, 41)$	⋮
	$b = \left(\frac{2-1}{2.5}\right)^2 + \left(\frac{40-41}{4}\right)^2$	⋮
	$\rightarrow 0.22 < 1$ donc dans l'ellipse	
●	$Sat2 = (6, 38)$	⋮
	$c = \left(\frac{2-6}{2.5}\right)^2 + \left(\frac{40-38}{4}\right)^2$	⋮
	$\approx 2.81 > 1$ donc en dehors de l'ellipse	



Acquisition

Extrait de code

```
for i in range(83):
    resultat=((cx-alng[i])/a)**2+((cy-alat[i])/b)**2
    if resultat<1:
        affichage=1          # si un satellite est détecté, on
        affiche les résultats à la fin
    if resultat<res_mem:    # car plus le résultat est petit, plus
        on est proche du centre de l'ellipse
    res_mem=resultat       # si résultat plus proche
    i_mem=i                # si résultat plus proche
```

Équation d'ellipse

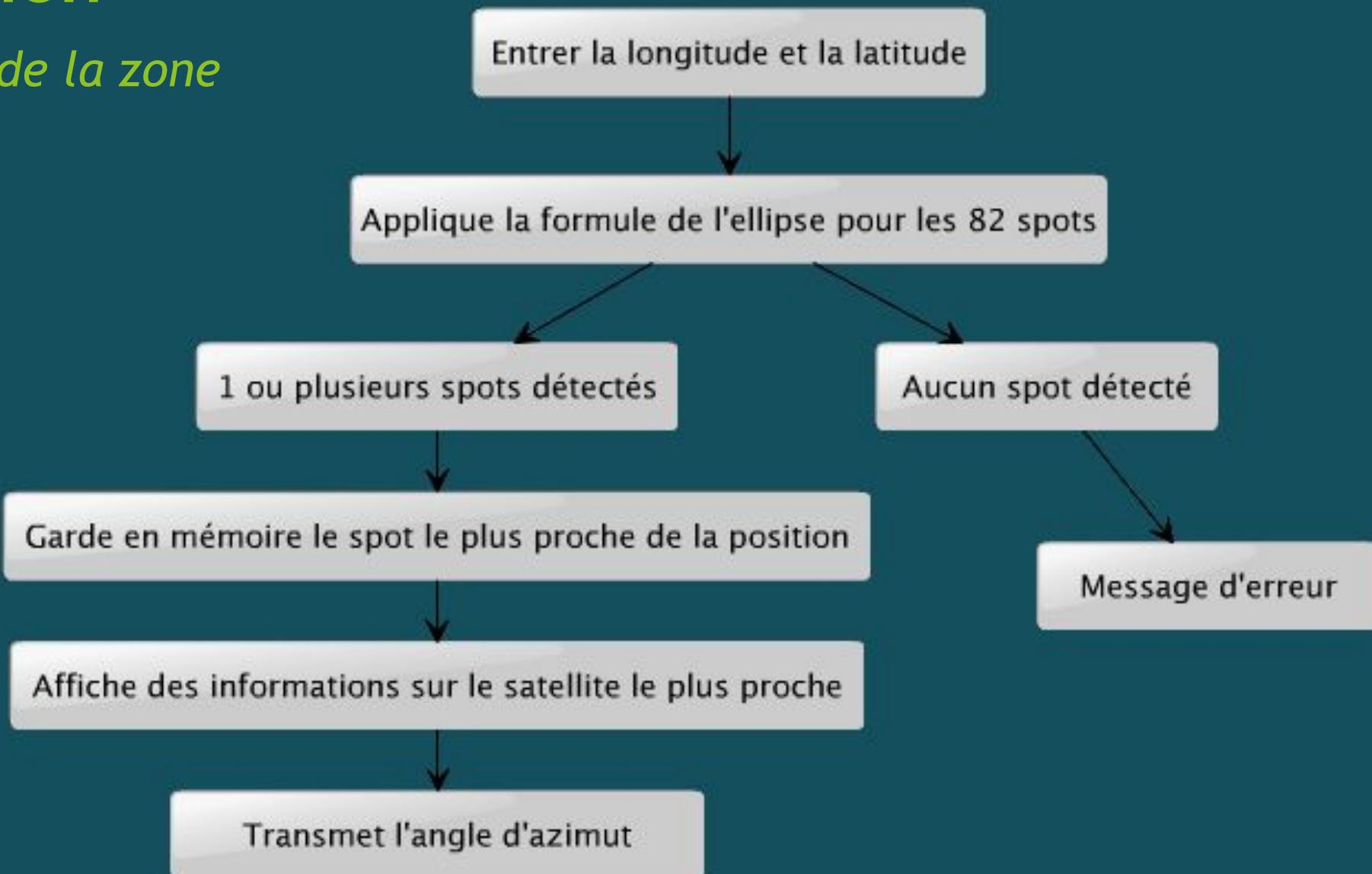
$$\left(\frac{cx - x_{\text{coord}}}{a}\right)^2 + \left(\frac{cy - y_{\text{coord}}}{b}\right)^2 = 1$$

- >1 : En dehors de l'ellipse
- <1 : Dans l'ellipse

```
Entrer la lattitude : 49.4
Entrer la longitude : 14.7
i: 28
type : 3
Lattitude du centre de l'ellipse : 48.36
Longitude du centre de l'ellipse : 14.06
Azimuth : 186.75688915875756
Elévation : 34.259828553744434
```

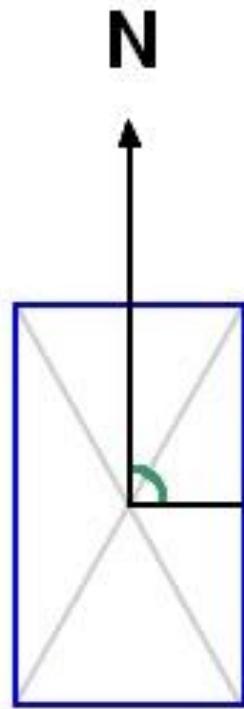
Acquisition

Détection de la zone



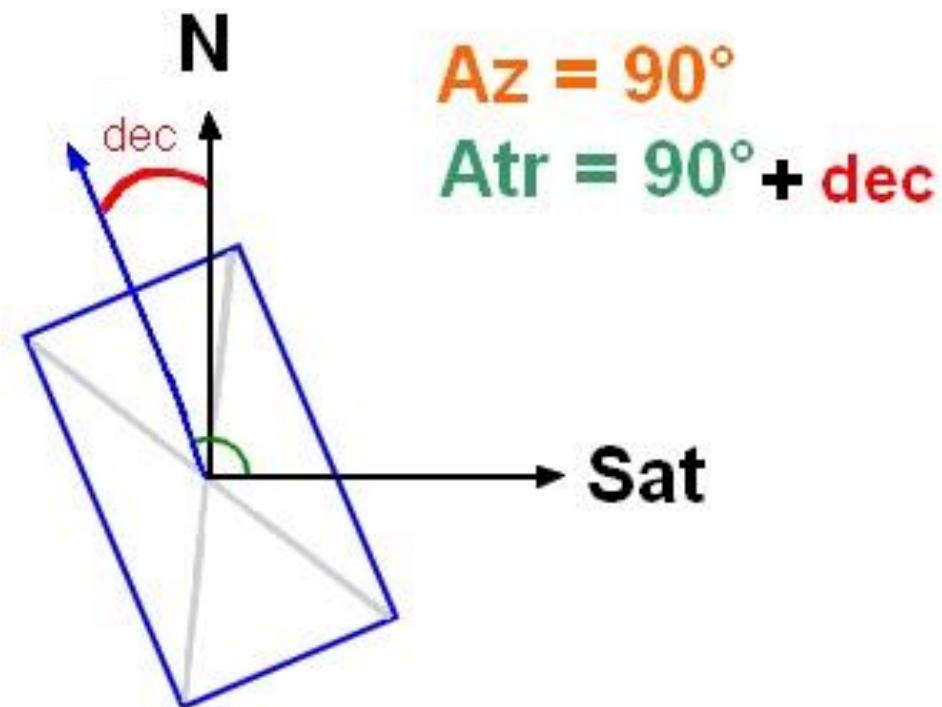
Acquisition

Capteur boussole



$$\begin{aligned} \text{Az} &= 90^\circ \\ \text{Atr} &= 90^\circ \end{aligned}$$

Cas idéal



$$\begin{aligned} \text{Az} &= 90^\circ \\ \text{Atr} &= 90^\circ + \text{dec} \end{aligned}$$

Cas réel

Acquisition

Capteur boussole QMC5883L

Fonctionnement

- Bibliothèque I2C smbus
 - write_byte_data() pour écrire dans les registres
 - read_byte_data() pour lire les valeurs d'axes
- Renvoie un nombre signé sur deux Octets
- Calcul de l'angle grâce à la fonction atan2()



```
import smbus #Librairie pour communication I2C
import time
from math import atan2 #permet d'utiliser la fonction atan2
pi=3.14159265359
bus = smbus.SMBus(1)
boussole = 0x0d #adresse de la boussole

#Initialisation de la boussole
bus.write_byte_data(boussole,0x0B,0x01)
time.sleep(0.010)
bus.write_byte_data(boussole,0x09,0x11)
time.sleep(1)

while 1:

    xlo=bus.read_byte_data(boussole, 0x00)#LSB X
    xhi=bus.read_byte_data(boussole, 0x01)#MSB X
    x=(xhi << 8) + xlo
    if(x>(2**15)-1):
        x=x-2**16      #transformer en non signé

    ylo=bus.read_byte_data(boussole, 0x02)#LSB Y
    yhi=bus.read_byte_data(boussole, 0x03)#MSB Y
    y=(yhi << 8) + ylo
    if (y>(2**15-1)):
        y=y-2**16      #transformer en non signé

    #Calcul de l'angle en fonction des valeurs d'axes
    angle=atan2(x,y)*180/pi+180
    #Afficher les valeurs
    print("x=",x)
    print("y=",y)
    print("angle :",angle)
    time.sleep(1)
```

Acquisition

Capteur boussole QMC5883l

Résultats obtenus

```
x= -1201  
y= -160  
angle 82.41160651189338
```

```
x= -931  
y= 326  
angle 109.29826970677156
```

```
x= -361  
y= 756  
angle 154.47490748334408
```

Résultats négatifs

```
[4255, 4788, 2156]  
('angle=', 221.62685388807296)  
[4247, 4802, 2156]  
('angle=', 221.4902890528038)  
[4255, 4791, 2166]  
('angle=', 221.60903460466778)  
[4241, 4786, 2161]  
('angle=', 221.54499368406385)
```

Résultats positifs

```
[-455, -943, -1296]  
('angle=', 25.75743242457341)  
[-7, -148, -1398]  
('angle=', 2.707917485877857)  
[-488, -363, -1393]  
('angle=', 53.35619435859445)  
[52, -743, -1355]  
('angle=', 355.99659444087536)  
[315, -467, -1357]
```

Acquisition

Récupération de la puissance du signal du satellite

- Bibliothèque Requests pour les requêtes HTTP
- Puissance du signal reçu et bruit accessibles depuis une page web hébergée sur le routeur

```
1 import requests
2 from requests_toolbelt.utils import dump
3
4 data = []
5 def getData():
6     global data
7
8     r    = requests.get("http://192.168.100.1/index.cgi?page=modemStatusData")
9     data = r.text.split("##")
10
11 getData()
12 rxPower = data[14]
13 rxSNR   = data[11]
14
15 print("Rx Power : " + rxPower + " dBm\tRx SNR : " + rxSNR + " dB")
```



Commande

Choix du vérin

Vérin SuperJack III
de la marque JAEGER

- 12 Pouces = 30,48 cm
- Charge statique: 225 kg
- Charge dynamique: 135 kg
- 76 impulsions par pouces
- Alimenté en 36 volts

Poids de la parabole:
~15 kg



Vérin JAEGER SuperJack III



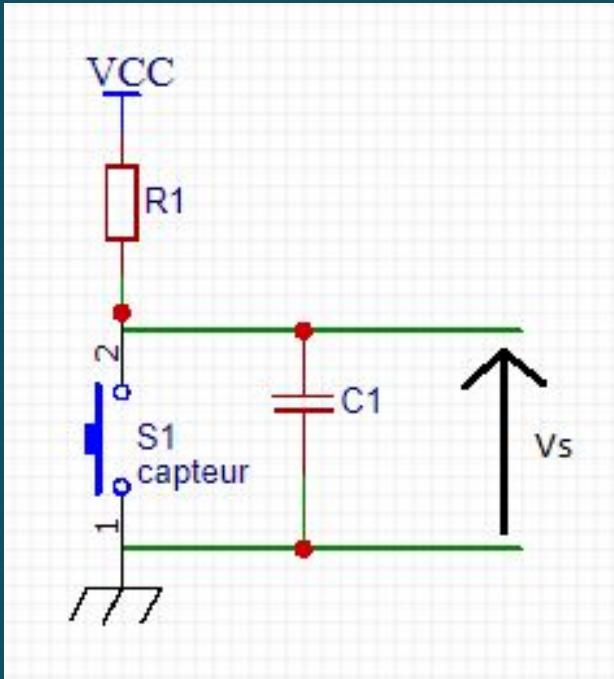
*Vérin installé sur la parabole
grâce au support*

Commande

Essais du vérin

Tests réalisés :

- Sur le capteur de position
 - Capteur de type interrupteur à lame souple
- Sur le vérin
 - Montée en 24V en ~ 2 min
 - Montée en 36V en ~ 1 min



Schémas de test du capteur de position



Vérin JAEGER SuperJack III



Vérin installé sur la parabole grâce au support

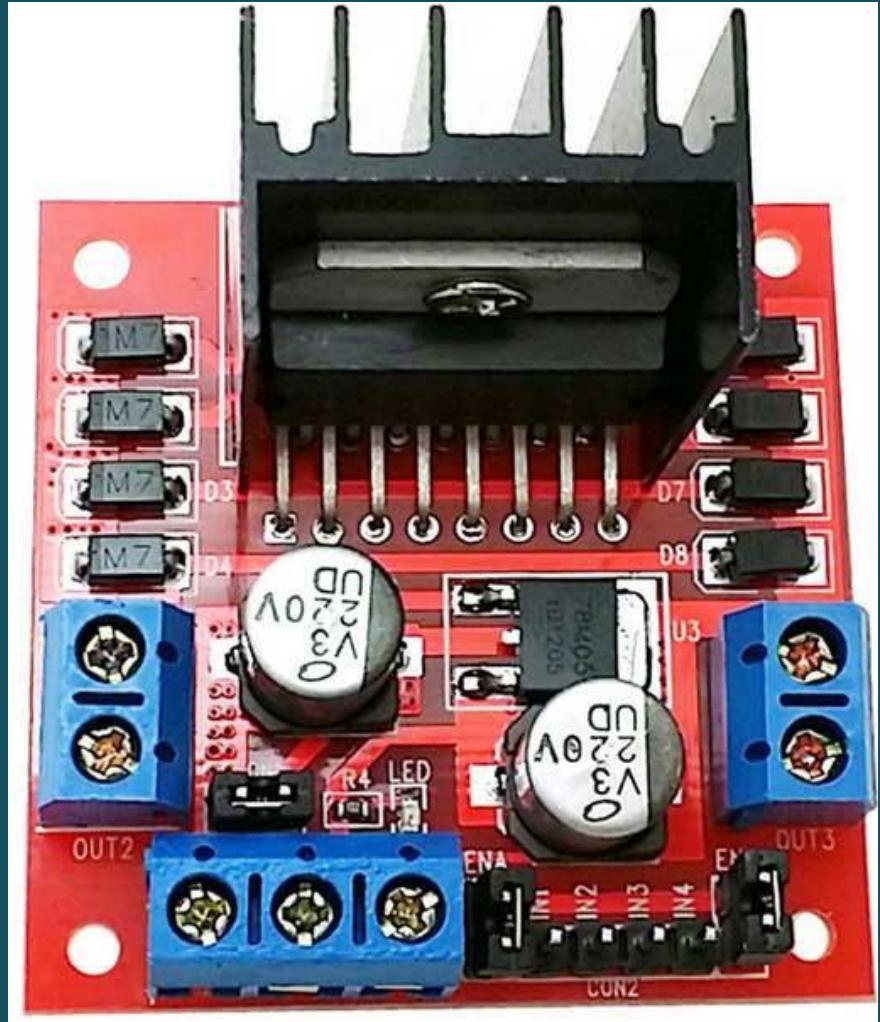
Commande

Essais du vérin

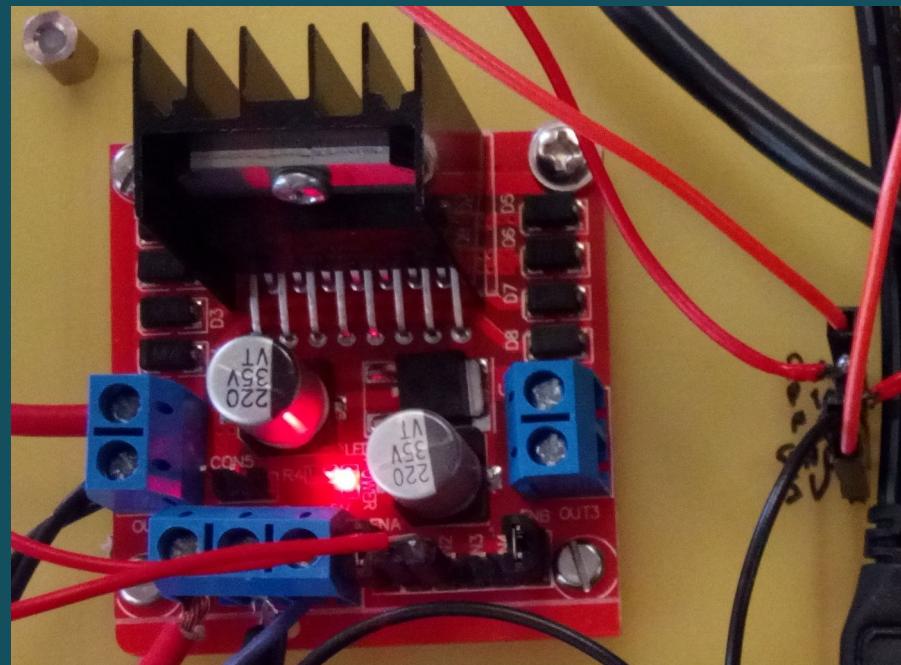


Commande

Carte interface vérin: L298N



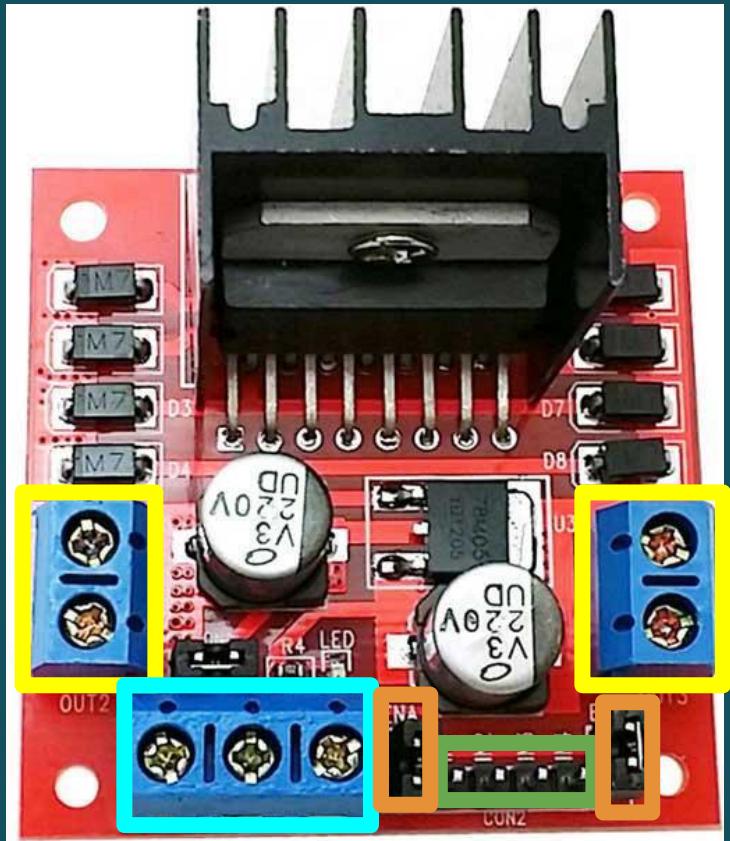
- Caractéristiques:
 - Tension de conduction: 5V à 35V
 - Tension de commande : 5V
 - Fournit jusqu'à 2A
 - Peut contrôler jusqu'à 2 moteur



L298N câblé et en fonctionnement

Commande

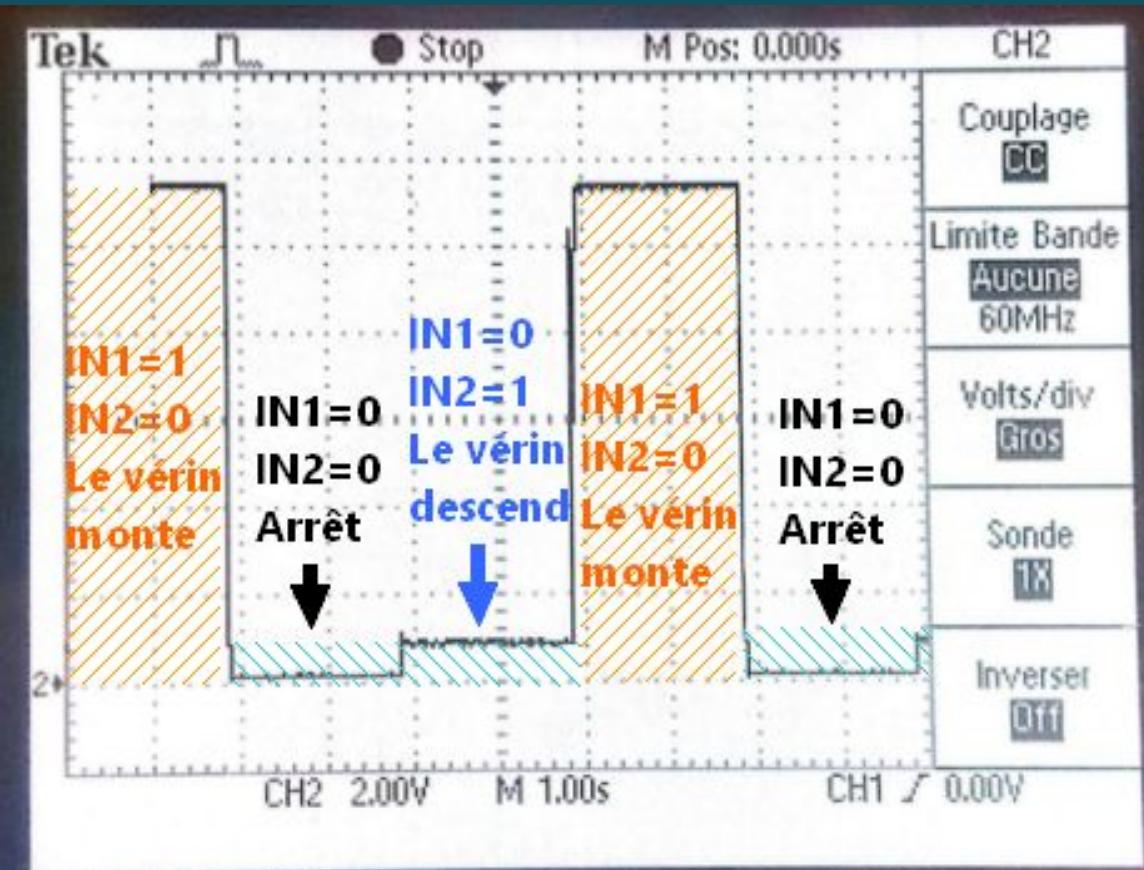
Étude de la carte de commande du vérin l298n



Broche	Fonction
OUT1, OUT2	Alimentation moteur 1
OUT3, OUT4	Alimentation moteur 2
IN1, IN2	Choix polarité du moteur 1
IN3, IN4	Choix polarité du moteur 2
ENA	Signal PWM de commande de vitesse moteur 1
ENB	Signal PWM de commande de vitesse moteur 2
+ et GND	Alimentation des moteur
5V	Alimentation partie commande 5V

Commande

Étude de la carte de commande du vérin l298n



- Quand :
 - IN1=1 et IN2 = 0 : le vérin monte
 - IN1=0 et IN2 = 1 : le vérin descend
 - IN1=0 et IN2 = 0 : le vérin s'arrête

Oscillogramme de la tension en sortie du L298N

Commande

Programme test de commande du vérin

```
from RPi import GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

mont = 16
desc = 18
GPIO.setup(mont,GPIO.OUT)
GPIO.setup(desc,GPIO.OUT)
#GPIO.output(mont,GPIO.LOW)
#GPIO.output(desc,GPIO.LOW)

def monter():
    GPIO.output(mont,GPIO.HIGH)
    GPIO.output(desc,GPIO.LOW)

def descendre():
    GPIO.output(mont,GPIO.LOW)
    GPIO.output(desc,GPIO.HIGH)

def arreter():
    GPIO.output(mont,GPIO.LOW)
    GPIO.output(desc,GPIO.LOW)|
```

Bibliothèque RPi.GPIO

- Permet la gestion de port GPIO

Méthode

- setmode (BOARD/GPIO)
Précise si l'on travaille sur les numéros de broche ou de GPIO
- setup (int, OUT/IN) Définit le mode
- output (int, HIGH/LOW)
Définit l'état de la sortie

Commande

Classe du vérin

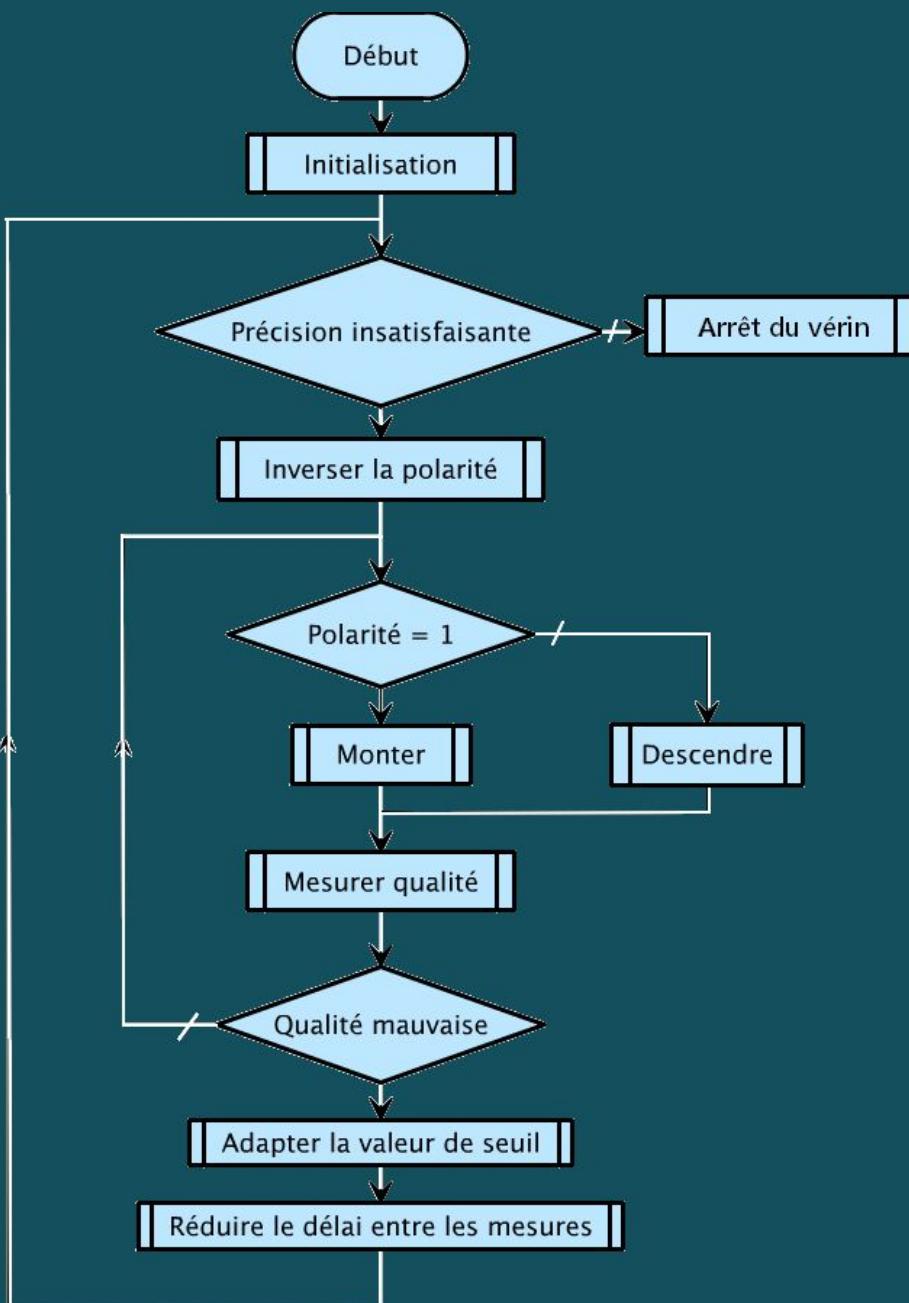
Bibliothèque RPi.GPIO

- Méthodes de contrôle
 - monter()
 - descendre()
 - arreter()
- Méthode d'acquisition
 - impulsion()
(sur événement)

```
1  class Verin:  
2      def __init__(self, pinPush, pinPull, pinInterrupt):  
3          self.pinPush = pinPush  
4          self.pinPull = pinPull  
5          self.pinInterrupt = pinInterrupt  
6          GPIO.setup(pinPush, GPIO.OUT)  
7          GPIO.setup(pinPull, GPIO.OUT)  
8          GPIO.setup(pinInterrupt, GPIO.IN, pull_up_down=GPIO.PUD_UP)  
9          GPIO.output(pinPush, GPIO.LOW)  
10         GPIO.output(pinPull, GPIO.LOW)  
11         self.compteur = 100  
12         GPIO.add_event_detect(pinInterrupt,  
13                               GPIO.FALLING,  
14                               callback=self.impulsion,  
15                               bouncetime=50)  
16  
17     def impulsion(self, channel):  
18         self.compteur += self.increment  
19         print("Impulsion {}".format(self.compteur))  
20  
21     def monter(self):  
22         self.increment = 1  
23         GPIO.output(self.pinPush, GPIO.HIGH)  
24         GPIO.output(self.pinPull, GPIO.LOW)  
25  
26     def descendre(self):  
27         self.increment = -1  
28         GPIO.output(self.pinPush, GPIO.LOW)  
29         GPIO.output(self.pinPull, GPIO.HIGH)  
30  
31     def arreter(self):  
32         GPIO.output(self.pinPush, GPIO.LOW)  
33         GPIO.output(self.pinPull, GPIO.LOW)
```

Commande

Algorigramme de positionnement de la parabole en site



```
sens :True  
Délai entre les mesures : 0.417 secondes  
Entrer la qualité : 1  
Entrer la qualité : 3  
Entrer la qualité : 5  
Entrer la qualité : 7  
Entrer la qualité : 5  
Entrer la qualité : 3  
Valeur seuil : 6.25  
sens :False  
Délai entre les mesures : 0.347 secondes  
Entrer la qualité : 3  
Entrer la qualité : 4  
Entrer la qualité : 5  
Entrer la qualité : 6  
Entrer la qualité : 7  
Entrer la qualité : 6  
Entrer la qualité : 5  
Valeur seuil : 6.625  
sens :True  
Délai entre les mesures : 0.289 secondes  
Entrer la qualité : 5  
Entrer la qualité : 6  
Entrer la qualité : 6.5  
Entrer la qualité : 7  
Entrer la qualité : 7.1  
Entrer la qualité : 7  
Entrer la qualité : 6.7  
Entrer la qualité : 5.9  
Entrer la qualité : 5.4  
Valeur seuil : 6.8625  
sens :False  
Délai entre les mesures : 0.241 secondes  
Entrer la qualité :
```

Commande

Rotor



Commande manuelle du rotor

rotor

- Permet de contrôler le rotor manuellement

ERC Mini V2

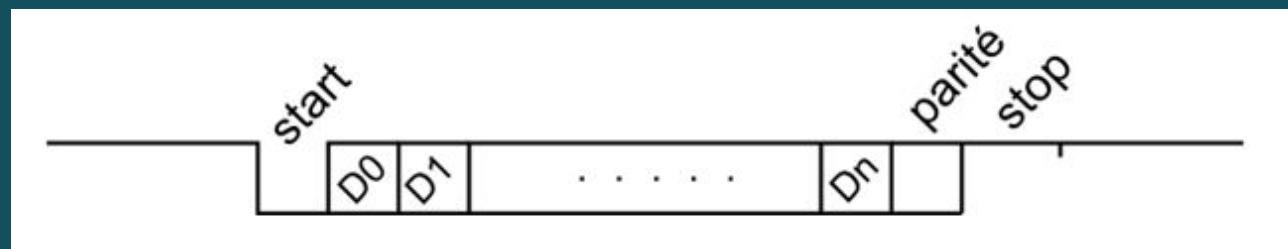


Le rotor

Commande

Rotor

- Bibliothèque *Serial*
 - Permet la communication série
- `ser = serial.Serial(port='/dev/ttyUSB1')` -> Crée un objet de type *serial*
- Méthodes :
 - `write()` permet d'envoyer un message
 - `readline()` permet de lire un message



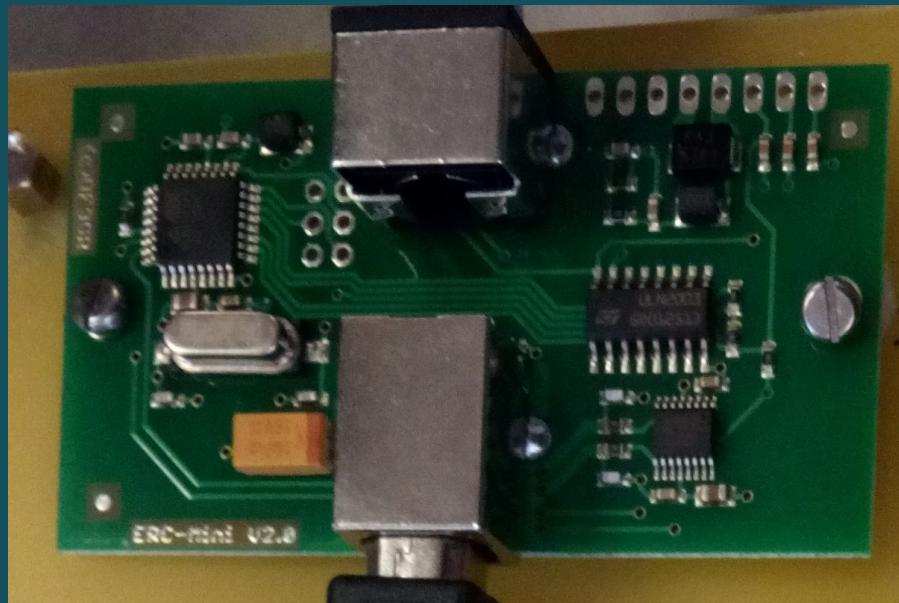
Commande

Rotor

- Configuration :
 - “sPRO0000” protocole
Yaesu GS232A
- Calibration :
 - rotor à zéro : “sCAL000”
 - rotor à 360° : “sCAR360”
- Positionner le rotor :
 - Rotor a xxx : “Mxxx”
 - Rotation à droite : “R”
 - Rotation à gauche : “L”
 - Stopper la rotation : “S”
- Toutes les commandes doivent être finies par un retour chariot
= \r



ERC mini V2 avec et sans boitier



Commande

Classe du rotor

- Méthodes de configuration
 - calibrationDroite()
 - calibrationGauche()
 - config()
- Méthodes de contrôle
 - tourner()
 - tournerHoraire()
 - tournerAntiHoraire()
 - stop()
- Méthode d'acquisition
 - angle()

```
1 class Rotor:  
2     def __init__(self, chemin):  
3         try:  
4             self.ser = serial.Serial(chemin)  
5         except serial.SerialException:  
6             self.ser = None  
7  
8     def tourner(self, angle):  
9         if self.ser is not None:  
10            angle = int(angle)  
11            commande = "M" + repr(angle) + "\r\n"  
12            self.ser.write(commande.encode('latin-1'))  
13  
14     def tournerHoraire(self):  
15         if self.ser is not None:  
16             self.ser.write(b'R\r\n')  
17  
18     def tournerAntiHoraire(self):  
19         if self.ser is not None:  
20             self.ser.write(b'L\r\n')  
21  
22     def stop(self):  
23         if self.ser is not None:  
24             self.ser.write(b'S\r\n')  
25  
26     def calibrationDroite(self):  
27         if self.ser is not None:  
28             self.ser.write(b'sCAR0360\r\n')  
29  
30     def calibrationGauche(self):  
31         if self.ser is not None:  
32             self.ser.write(b'sCAL0000\r\n')  
33  
34     def calibrationConfig(self):  
35         if self.ser is not None:  
36             self.ser.write(b'sPRO0000')  
37             self.ser.write(b'sSPA0001')  
38             self.ser.write(b'sSPL0001')  
39             self.ser.write(b'sSPH0003')
```

Coordination

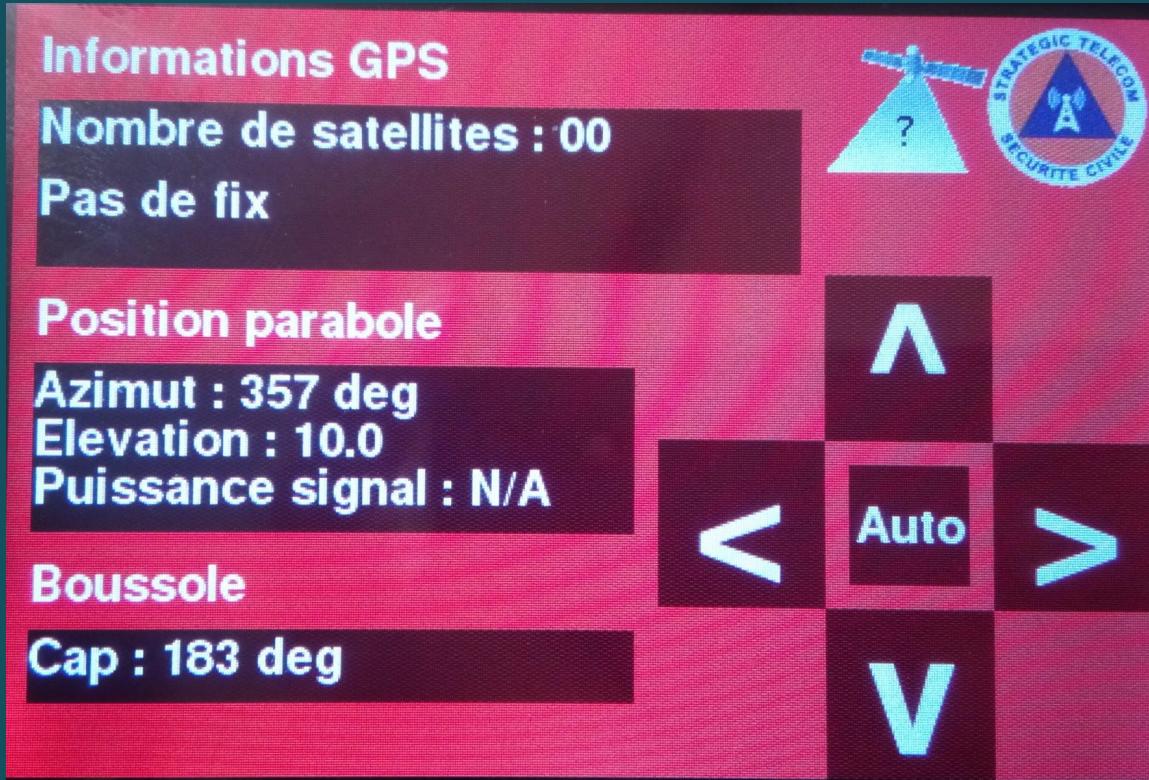
Configuration du Raspberry Pi

```
1 Installation
2 =====
3
4 Sur un ordinateur *nix
5 -----
6
7 Flasher le système d'exploitation sur la carte microSD du Raspberry Pi (Raspbian Jessie Lite pour PiTFT 3,5" https://learn.adafruit.com/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi/easy-install) :
8 sudo dd if=2016-11-08-pitft-35r.img of=/dev/diskX bs=1M
9
10 Pour utiliser la connexion SSH, créer un fichier « ssh » à la racine de la partition « boot » créée :
11 touch ssh
12
13 Alimenter le Raspberry Pi
14
15 Mettre les fichiers « dependances.sh » et « parabole.py » sur le Raspberry Pi, par exemple en utilisant la connexion SSH avec l'adresse IP de Raspberry Pi 192.168.2.3 :
16 scp dependances.sh parabole.py pi@192.168.2.3:~
17
18 Mot de passe par défaut : raspberry
19
20
21 Sur le Raspberry Pi (en SSH ou en direct)
22 -----
23 sudo raspi-config
24 « Expand filesystem »
25
26 sudo chmod +x dependances.sh
27 sudo ./dependances.sh
28
29 Éditer le fichier "/etc/rc.local"
30 nano /etc/rc.local
31 Ajouter les lignes suivantes à la fin du fichier
32 cd /home/pi
33 sudo python /home/pi/parabole.py &
```

```
1 #!/bin/bash
2
3 # Mises a jour initiales
4 sudo apt-get update
5 sudo apt-get -y --force-yes upgrade
6
7 # Installation de Python et Pip
8 sudo apt-get -y --force-yes install python-pip
9 sudo apt-get -y --force-yes install python-pygame
10
11 # Dependances pour le bon fonctionnement de l'écran tactile avec Pygame
12 # Activation des sources de paquets Wheezy
13 echo "deb http://archive.raspbian.org/raspbian wheezy main
14 " > /etc/apt/sources.list.d/wheezy.list
15
16 # Réglage de la source des paquets par défaut comme wheezy
17 echo "APT::Default-release \"oldstable\";
18 " > /etc/apt/apt.conf.d/10defaultRelease
19
20 # Réglage de la priorité de libsdl wheezy supérieure à celle du paquet jessie
21 echo "Package: libsdl1.2debian
22 Pin: release n=jessie
23 Pin-Priority: -10
24 Package: libsdl1.2debian
25 Pin: release n=wheezy
26 Pin-Priority: 900
27 " > /etc/apt/preferences.d/libsdl
28
29 # Installation de SDL 1.2
30 apt-get update
31 apt-get -y --force-yes install libsdl1.2debian/wheezy
32
33 # Installation de la bibliothèque Python PySerial (communication avec les ports série)
34 sudo python -m pip install pyserial
35 # Installation de la bibliothèque Python Pynmea2 (exploitation des tracés GPS)
36 sudo pip install pynmea2
37 # Installation de la bibliothèque Python bitstring (conversion des valeurs binaires)
38 sudo pip install bitstring
39 # Installation de la bibliothèque Python smbus2 (communication avec les périphériques I2C)
40 sudo pip install smbus2
```

Coordination

Réalisation de l'interface graphique



Informations affichées :

- Position (latitude & longitude)
- Nombre de satellites connectés
- Zone dans laquelle l'utilisateur est situé (1 à 4)
- Qualité de la communication avec le satellite
- Orientation du camion

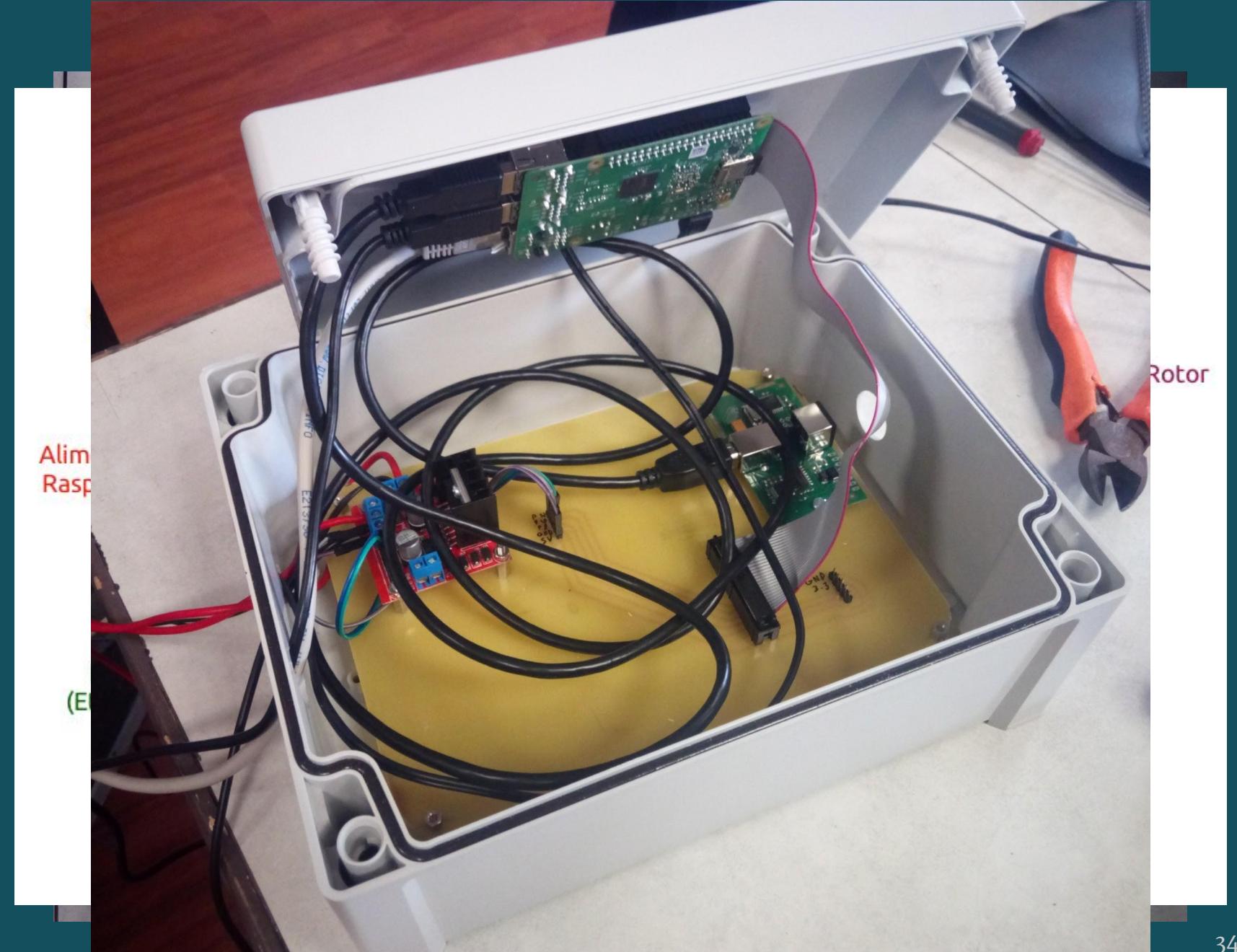
Contrôles :

- Démarrage du positionnement automatique
- Positionnement manuel de la parabole
- Sélection manuel de la zone

Coordination

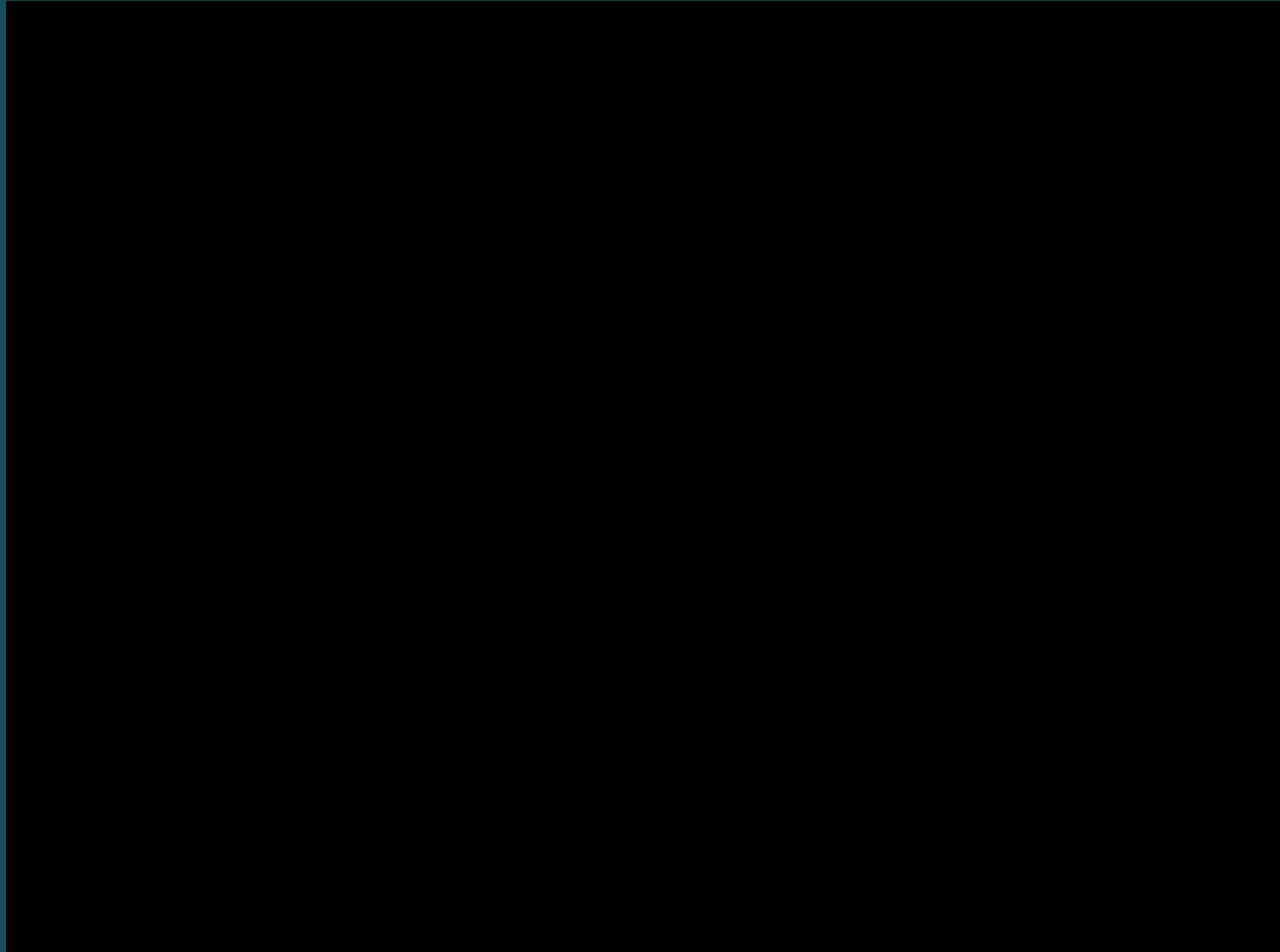
Implantation de la carte électronique et boîtier

- Regroupe tous les composants
- Placement logique des différents modules
- Placée par la suite dans un boîtier



Démonstration

(attention les yeux)



Problèmes rencontrés

- Problèmes de livraison
- Boussole hantée (problèmes d'acquisition des valeurs de la boussole)
- Nombreux essais sur site nécessaires

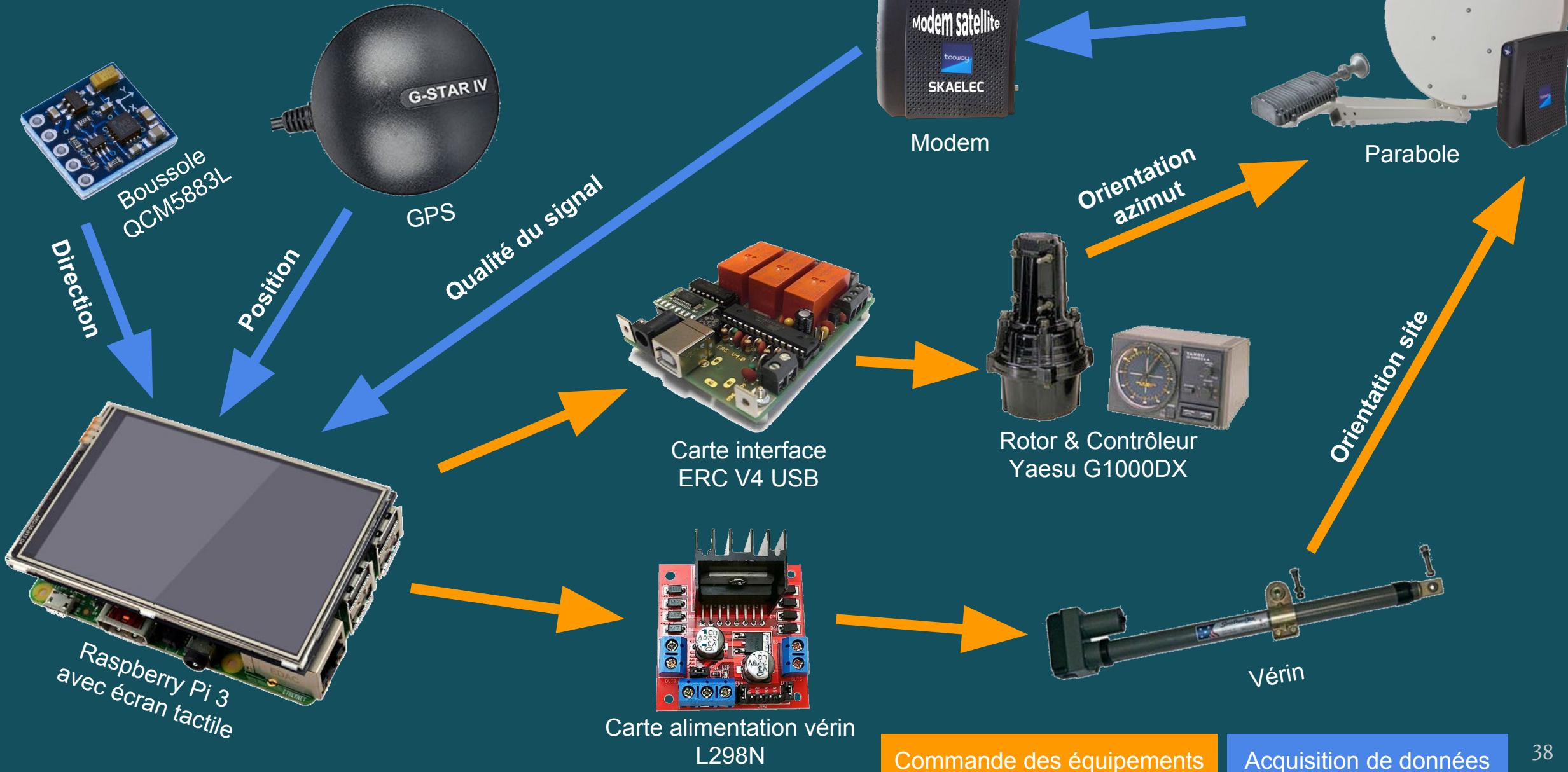


Conclusion

- Projet terminé à 90 %
- Très bonne expérience
- Retards à gérer
- Plusieurs nouvelles technologies



Questions



Travail réalisé

Rotor



Introduction

La camionnette de la **Sécurité Civile** sur laquelle nous allons travailler est équipée d'une parabole **Tooway** qui permet d'accéder à **internet** par **satellite**.



Le **positionnement** de la parabole s'effectue **manuellement** en montant sur le toit de la camionnette et en écoutant les « bips » émis par celle-ci en fonction de la **qualité** de réception du signal satellite.

Notre travail est d'automatiser le positionnement de cette parabole afin d'obtenir **facilement** et **rapidement** le meilleur signal possible.

<i>Diaporama (forme)</i>							
Utilisation des supports	0	1	2	3	4		
- qualité du diaporama : première page réglementaire							
- annonce de la structure de l'exposé							
- équilibre texte, image, discours							
- animations, rythme							
<i>[chorégraphie, costumes ;-) !]</i>							
<i>Expression (forme)</i>							
Débit, fluidité de langue	0	1	2				
Correction de la langue	0	1	2				
Attitude, regard, prise en compte du public, conviction, implication	0	1	2				
<i>Déroulement et technique (fond)</i>							
Analyse et présentation du projet et des tâches	0	1	2	3	4	5	6
Répartition des tâches du projet	0	1	2	3	4		
Présentation des choix opérés	0	1	2				
Justification des choix opérés	0	1	2	3	4	5	6
Budget (maîtrise des coûts, justification)	0	1	2	3			
Comparatif des plannings et analyse des problèmes	0	1	2	3	4		
Bilan circonstancié	0	1	2				
<i>Conclusion</i>							
Rappel des idées principales	0	1					
Interrogations, perspectives, avis personnel	0	1	2				
<i>Pénalités appliquées à la note /20</i>							
Respect du temps imparti : < 20min ou > 30 min = - 2pts	0		-2				
NOTE OBTENUE /40 puis ramenée /20 / 40		 / 20			