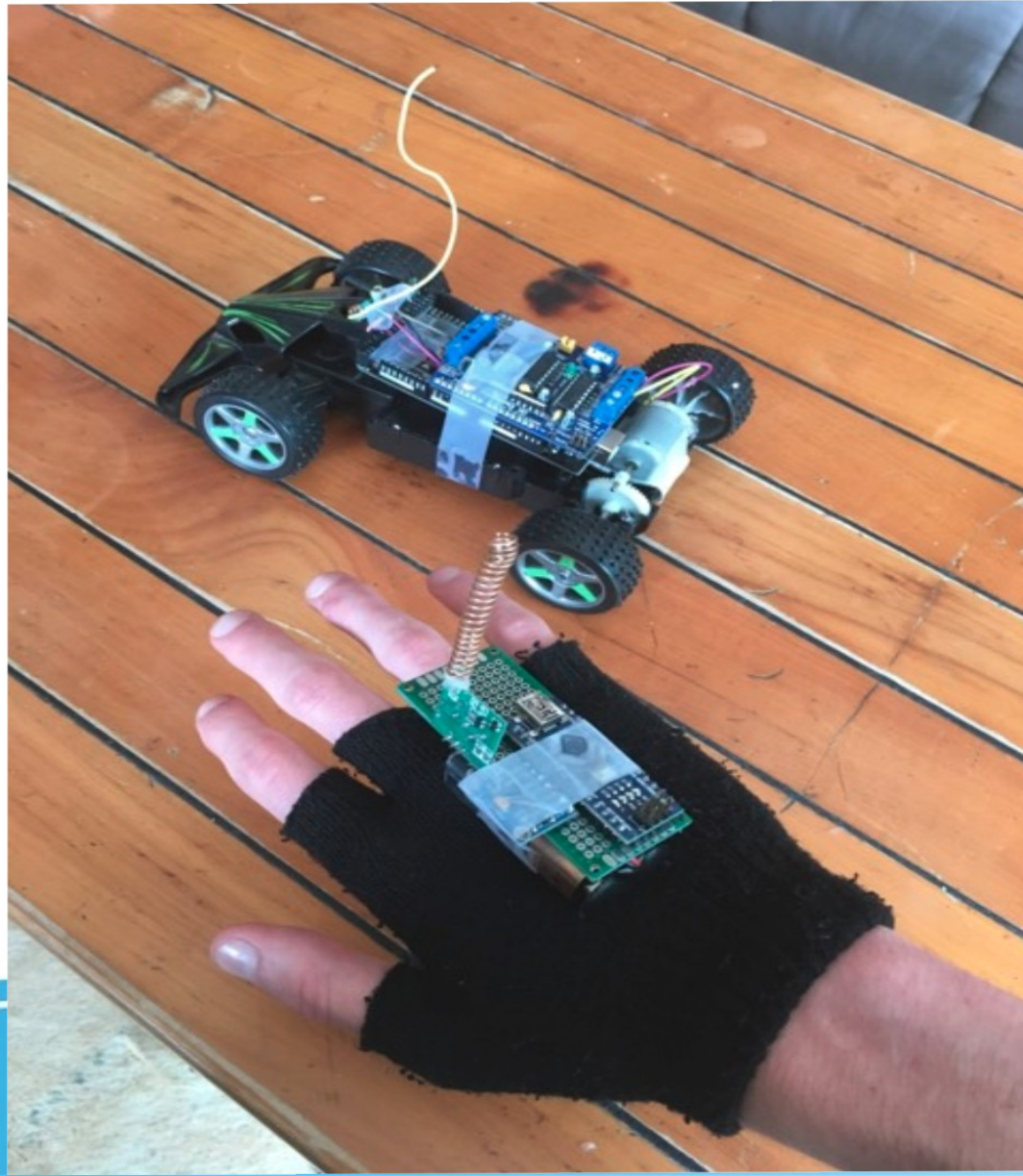


LA VOITURE TOUT EN MAIN



SOMMAIRE

I) Explication générale

- Matériel utilisé
- Schéma du dispositif

II) Cahier des charges

III) Informatique

- Partie émetteur
- Partie récepteur

IV) Problèmes rencontrés

Le but de ce TIPE est de contrôler une voiture grâce aux mouvements de la main selon 2 rotations:

Le but de ce TIPE est de contrôler une voiture grâce aux mouvements de la main selon 2 rotations:

- Le tangage permet la rotation du moteur et donc de faire avancer/ reculer la voiture

Le but de ce TIPE est de contrôler une voiture grâce aux mouvements de la main selon 2 rotations:

- Le tangage permet la rotation du moteur et donc de faire avancer/ reculer la voiture
- Le roulis actionne le servo moteur et qui fait tourner la voiture

Rapport au thème

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Etant une voiture radiocommandée, notre TIPE rentre parfaitement dans le thème du transport

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur
- Un Gyroscope

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur
- Un Gyroscope
- Un émetteur et un récepteur 433MHz

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur
- Un Gyroscope
- Un émetteur et un récepteur 433MHz
- Un moteur 5V à courant continu

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur
- Un Gyroscope
- Un émetteur et un récepteur 433MHz
- Un moteur 5V à courant continu
- Un servo moteur

Matériel utilisé

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- 3 piles 9V
- Une carte Arduino Nano + une carte Arduino Mega
- Un motor shield pour contrôler la vitesse de rotation du moteur
- Un Gyroscope
- Un émetteur et un récepteur 433MHz
- Un moteur 5V à courant continu
- Un servo moteur
- Un châssis de voiture réutilisé et un gant

Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

Pile 9V



Arduino Nano

Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

Pile 9V

Gyroscope

Arduino Nano

Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

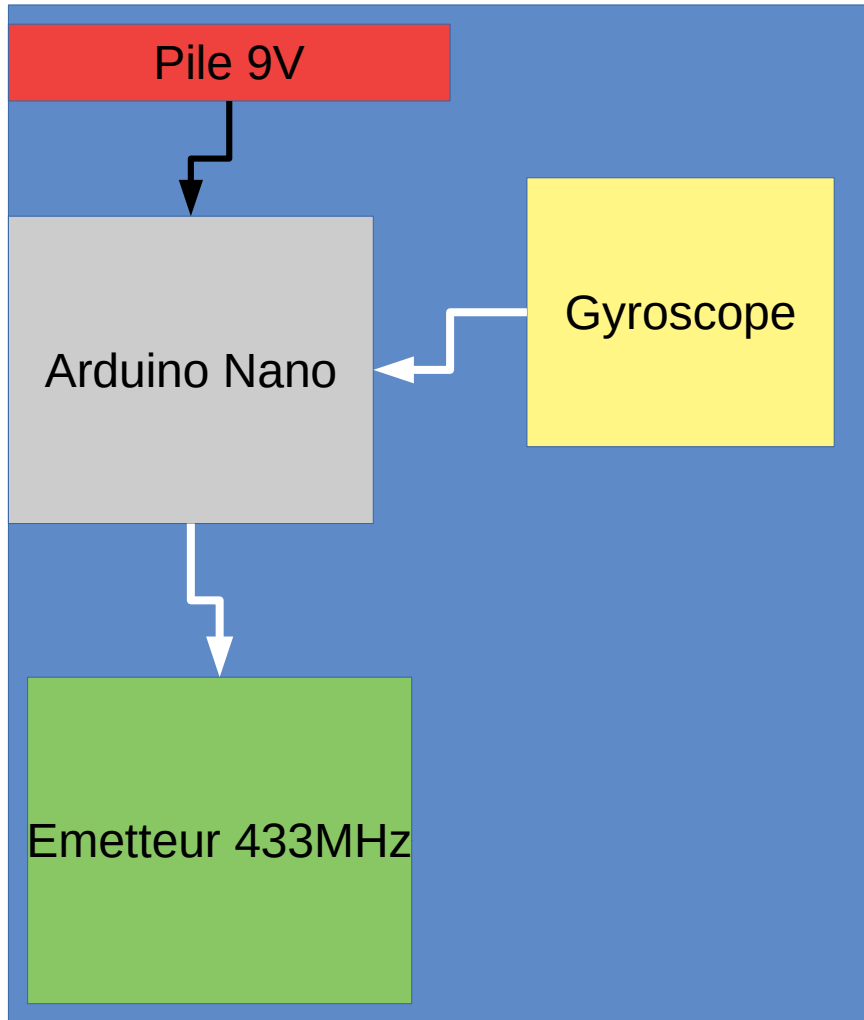


Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

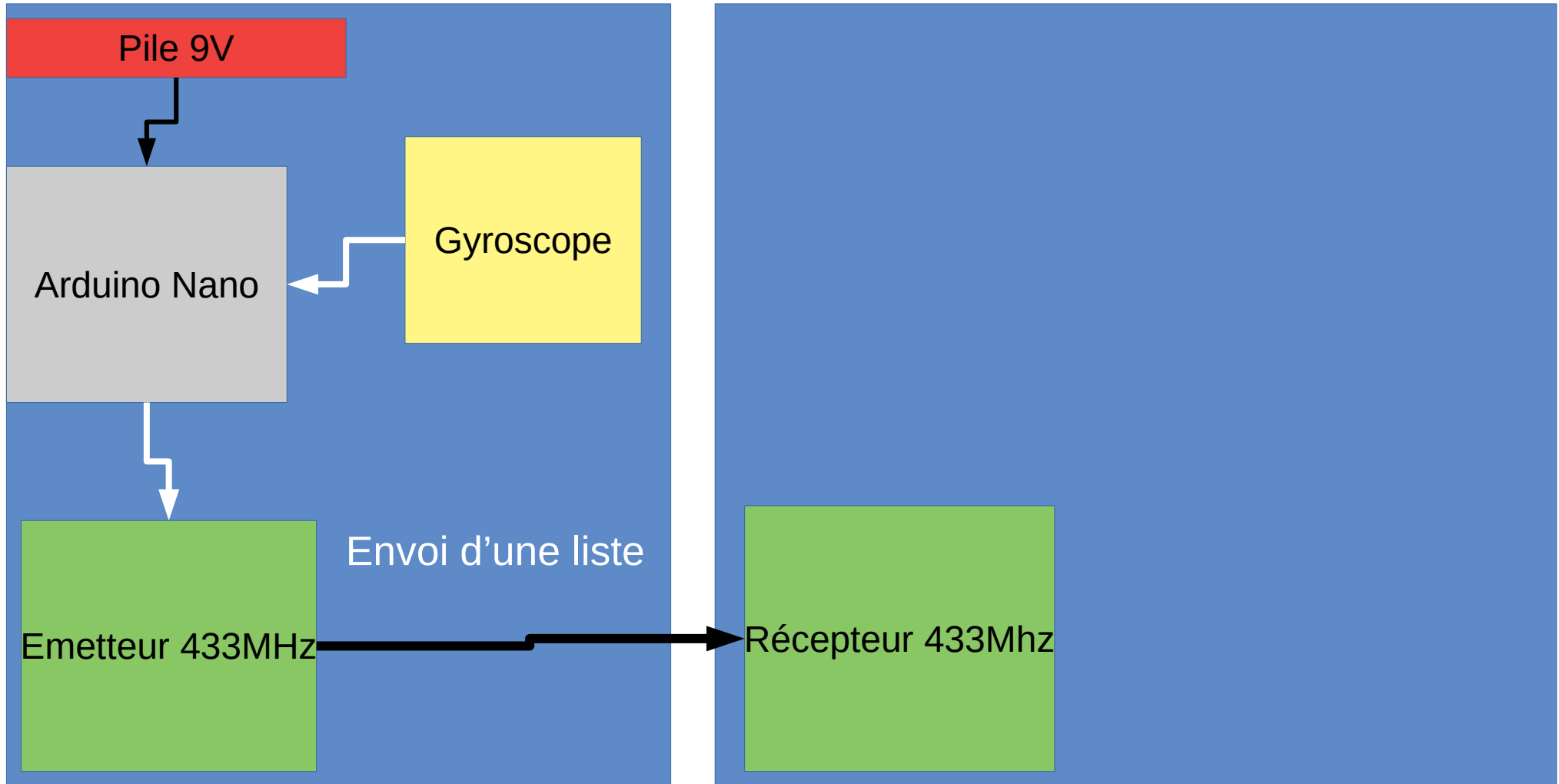


Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

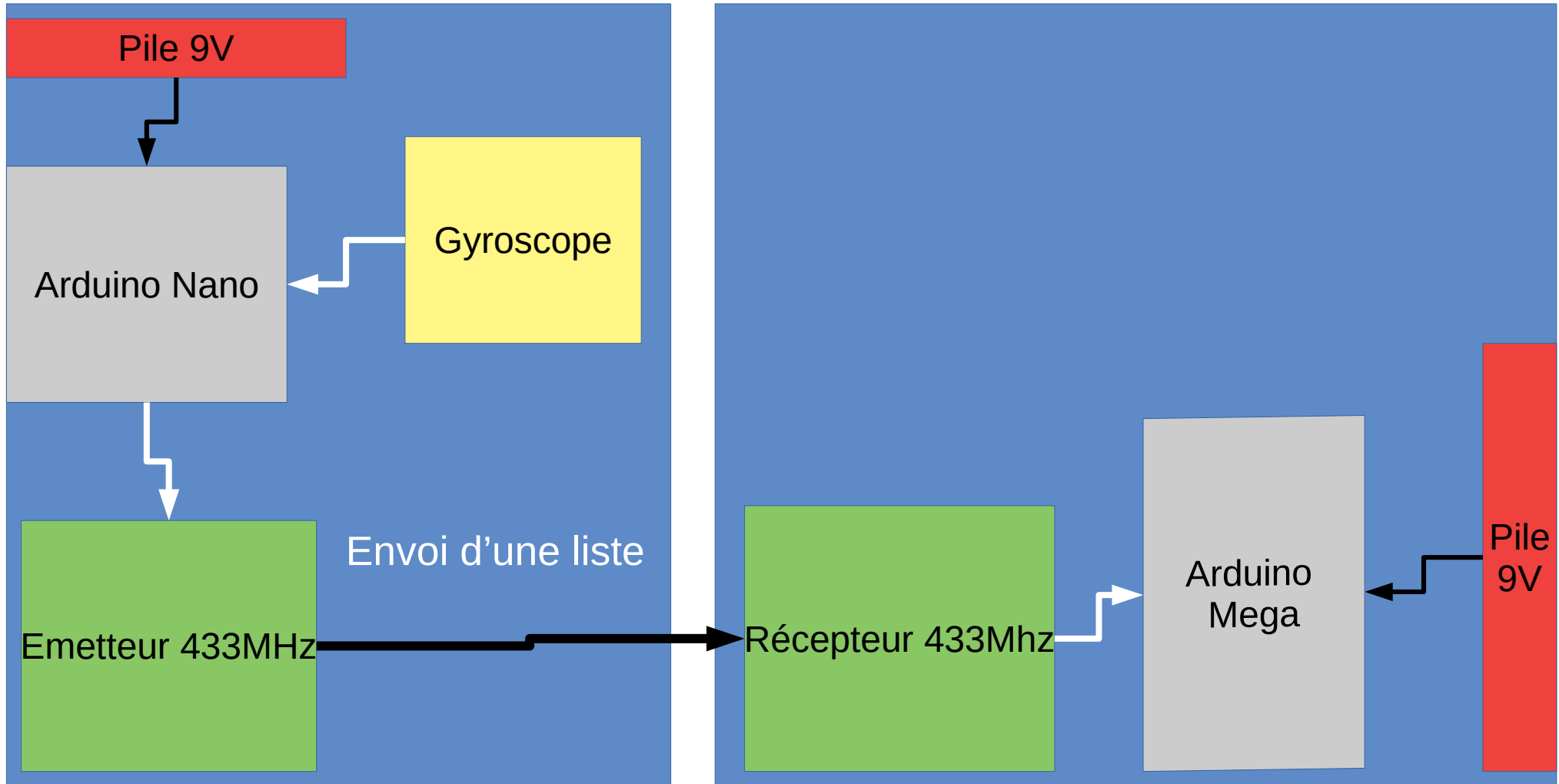


Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

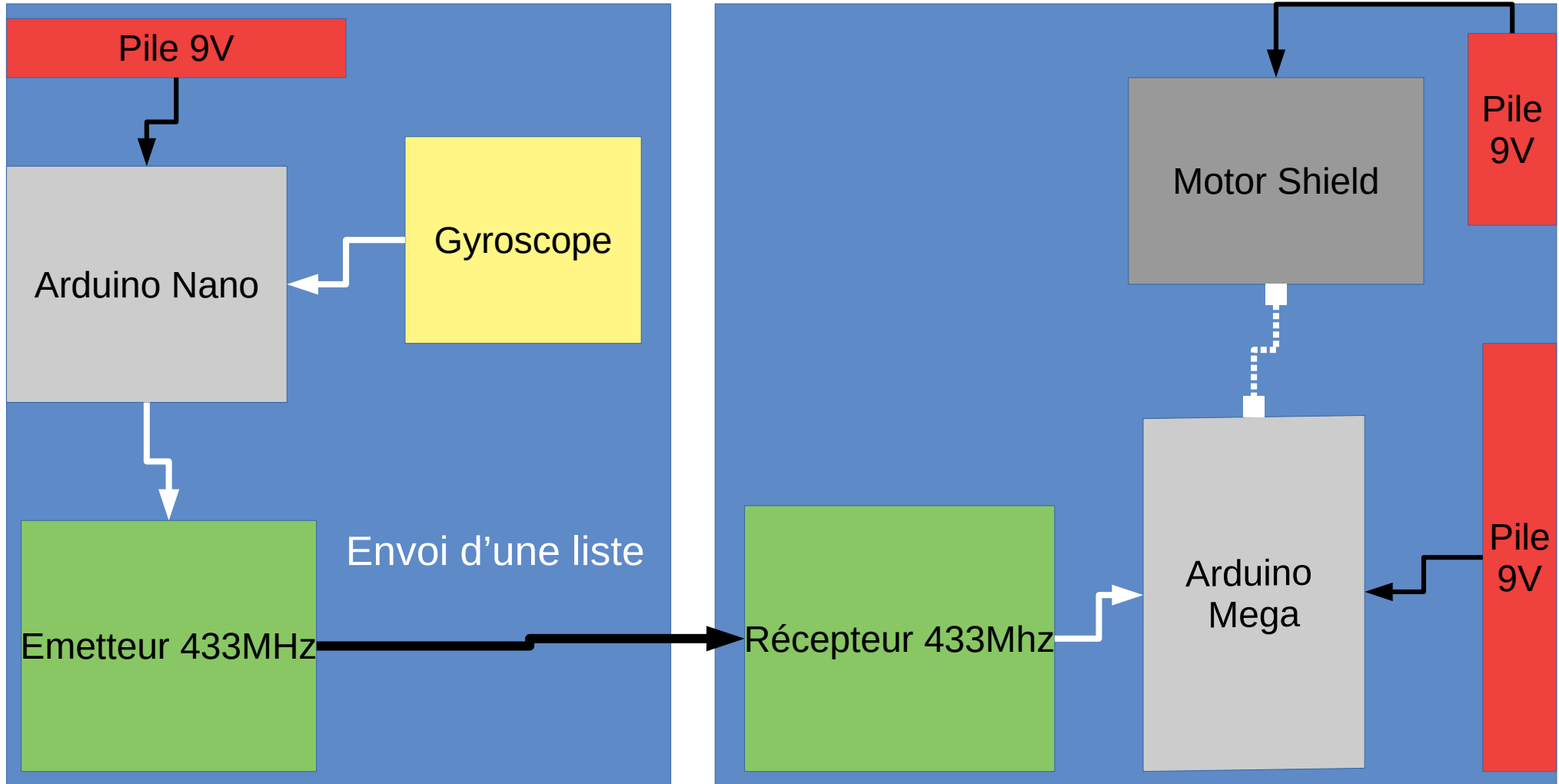


Schéma du dispositif

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)

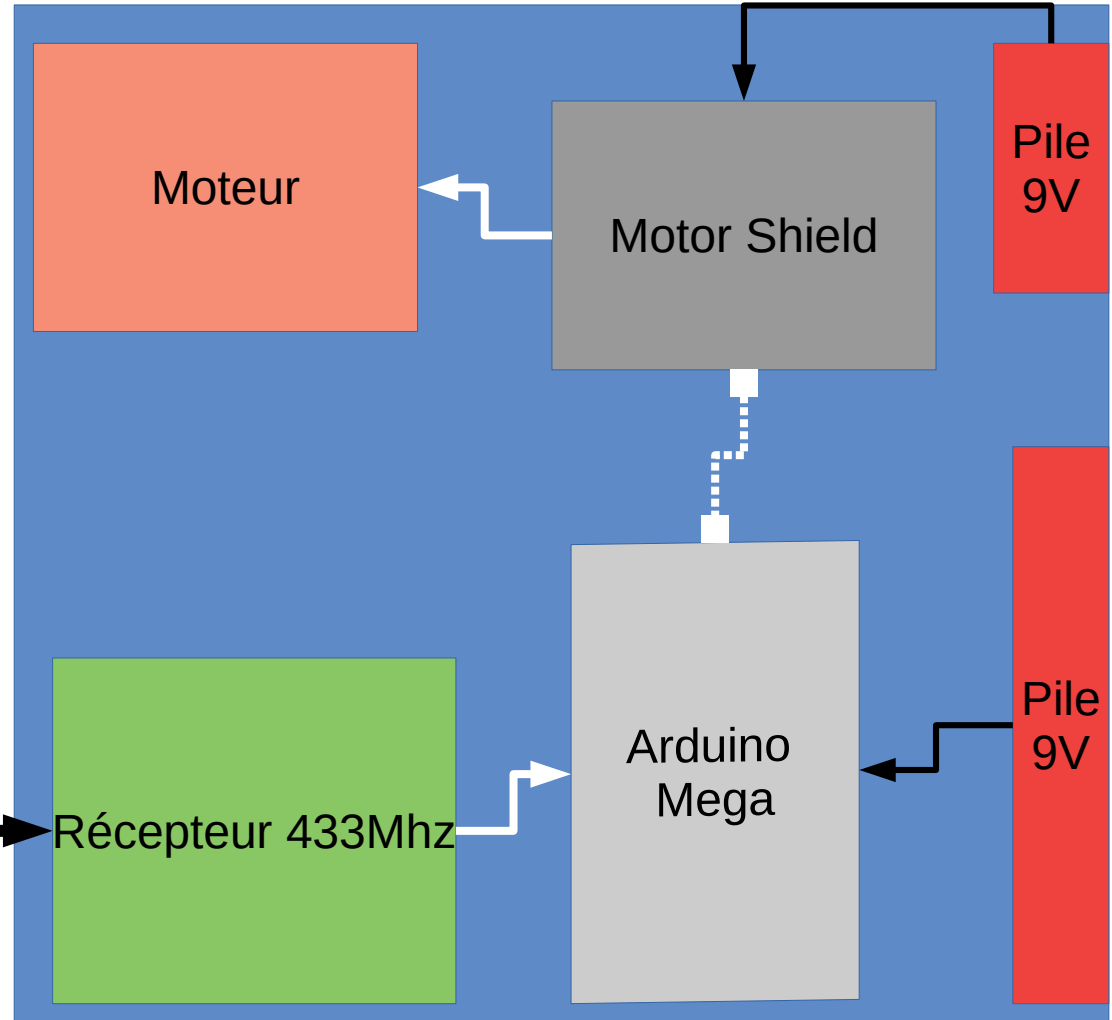
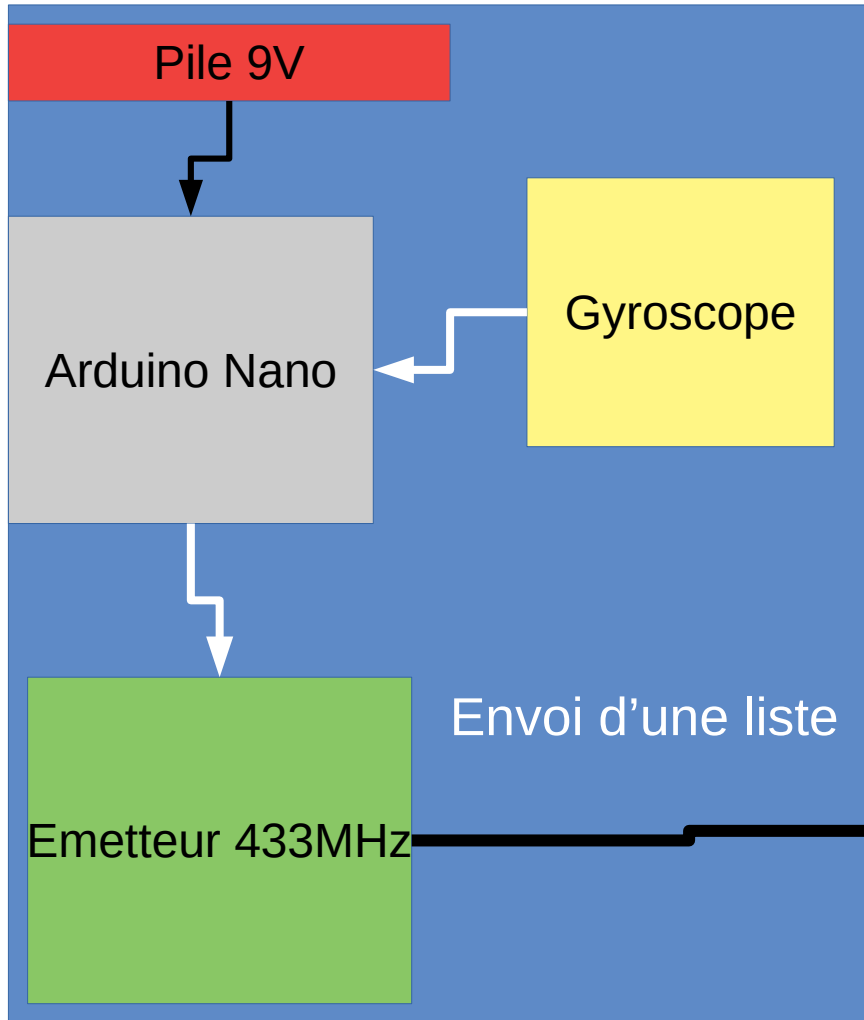


Schéma du dispositif

Explication générale

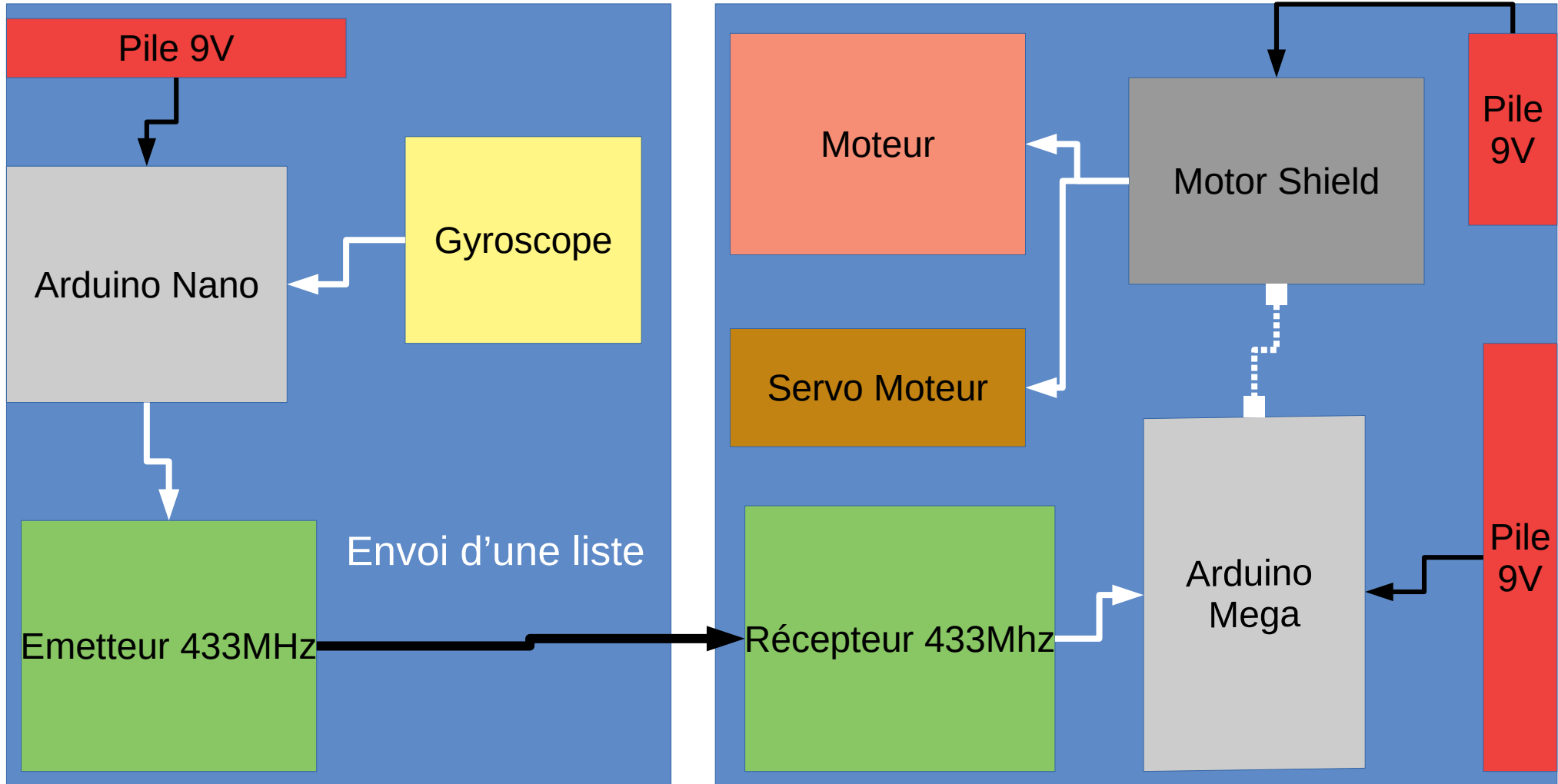
Cahier des charges

Informatique

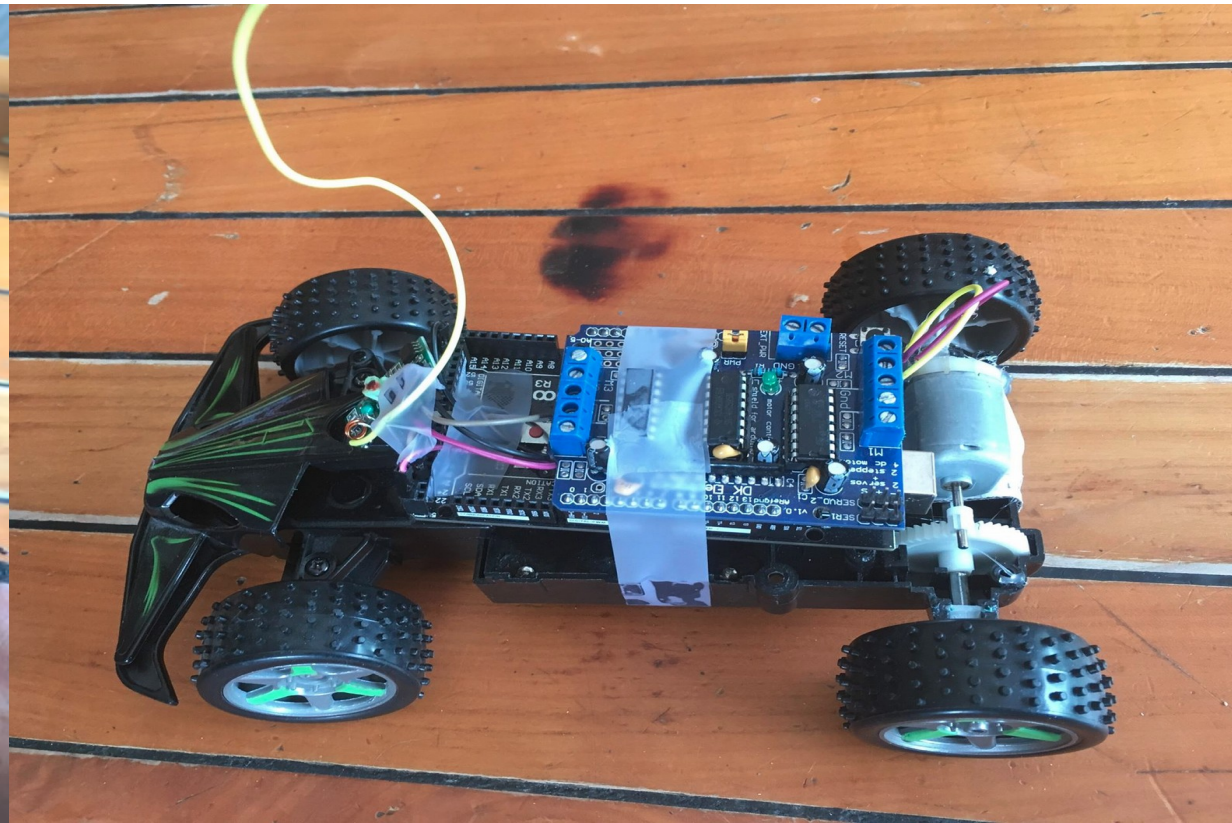
Problèmes rencontrés

Partie émetteur (sur la main)

Partie récepteur (sur la voiture)



Photographies du dispositif:



Cahier des charges

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

- Budget limité (inférieur à 100 €)
- Réutiliser des matériaux (comme le châssis de la voiture)
- Motorisation: moteur 5V à courant continu
- Type d'entraînement des roues: propulsion

Informatique

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Cette partie est séparée en deux: la partie émetteur et la partie récepteur.

Rémi, mon partenaire pour ce TIPE, était plutôt au point sur l'électronique tandis que moi sur l'informatique. On s'est donc naturellement réparti les rôles.

Partie émetteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

Partie émetteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Récupérer les données envoyées par le gyroscope selon la rotation autour de deux axes grâce à la carte Arduino

Partie émetteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Récupérer les données envoyées par le gyroscope selon la rotation autour de deux axes grâce à la carte Arduino
- Envoyer ces données via l'émetteur 433MHz

Partie émetteur

Explication générale

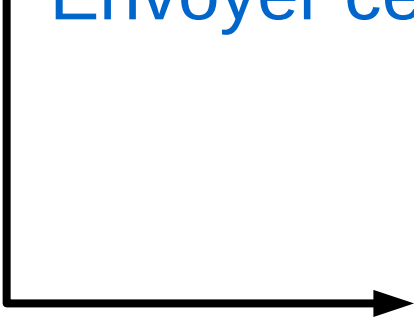
Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Récupérer les données envoyées par le gyroscope selon la rotation autour de deux axes grâce à la carte Arduino
- Envoyer ces données via l'émetteur 433MHz



Pour ce faire, on a envoyé une liste de taille 2 dans laquelle le premier élément est la valeur à récupérer pour le moteur et la deuxième, celle pour le servo moteur.

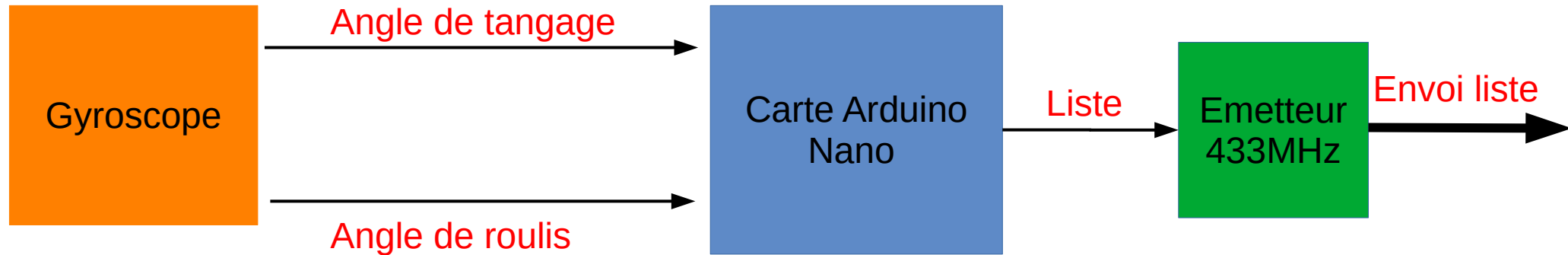
Organigramme émetteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés



Programme récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie on inclut les bibliothèques nécessaires:

```
Servo_moteur.ino x
#include <VirtualWire.h>
#include <MPU6050.h>
#include <I2Cdev.h>

MPU6050 mpu ;

int16_t ax, ay, az ;
int16_t gx, gy ,gz ;
```

Programme récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie on inclut les bibliothèques nécessaires:

- VirtualWire pour communiquer avec les émetteurs/récepteurs 433MHz

```
Servo_moteur.ino x
#include <VirtualWire.h>
#include <MPU6050.h>
#include <I2Cdev.h>

MPU6050 mpu ;

int16_t ax, ay, az ;
int16_t gx, gy ,gz ;
```


Programme récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie on inclut les bibliothèques nécessaires:

- VirtualWire pour communiquer avec les émetteurs/récepteurs 433MHz
- *I2Cdev et MPU6050 pour le gyroscope*

```
Servo_moteur.ino x
#include <VirtualWire.h>
#include <MPU6050.h>
#include <I2Cdev.h>

MPU6050 mpu ;

int16_t ax, ay, az ;
int16_t gx, gy ,gz ;
```

Initialisation des axes de rotation et d'accélération, de la vitesse de transfert et du gyroscope avec vérification de la connexion de ce dernier.

```
int16_t ax, ay, az ;
int16_t gx, gy ,gz ;

void setup()
{
  Serial.begin(9600);
  vw_setup(2000);
  mpu.initialize();
  Serial.println(mpu.testConnection()? " YES " : " NO " );
  delay(1000);
  Serial.println("Capt values from sensor");
  delay(1000);
}
```

Association des axes de rotation et d'accélération avec le gyroscope.

```
22 delay(1000);  
23 }  
24  
25 void loop() {  
26  
27     mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  
28     float a = map(ax, -17000, 17000, -90, 90);  
29     float b = map(az, -1700, 1700, 750, 2250);  
30     float tab[2];  
31     tab[0] = a;  
32     tab[1] = b;  
33  
34     vw_send((byte *) &tab, sizeof(tab));    ///  
35     vw_wait_tx();  
36 }
```

Association des axes de rotation et d'accélération avec le gyroscope.

Utilisation de la fonction `map` pour remettre à l'échelle voulue les valeurs prises par le gyroscope.

```
22 delay(1000);  
23 }  
24  
25 void loop() {  
26  
27     mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  
28     float a = map(ax, -17000, 17000, -90, 90);  
29     float b = map(az, -1700, 1700, 750, 2250);  
30     float tab[2];  
31     tab[0] = a;  
32     tab[1] = b;  
33  
34     vw_send((byte *) &tab, sizeof(tab));    ///  
35     vw_wait_tx();  
36 }
```

Association des axes de rotation et d'accélération avec le gyroscope.

Utilisation de la fonction `map` pour remettre à l'échelle voulue les valeurs prises par le gyroscope.

Création et envoi de la liste vers le récepteur.

```
22 delay(1000);  
23 }  
24  
25 void loop() {  
26  
27     mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  
28     float a = map(ax, -17000, 17000, -90, 90);  
29     float b = map(az, -1700, 1700, 750, 2250);  
30     float tab[2];  
31     tab[0] = a;  
32     tab[1] = b;  
33  
34     vw_send((byte *) &tab, sizeof(tab));    ///  
35     vw_wait_tx();  
36 }
```

Le Gyroscope prend des valeurs entre -1700 et 1700 ce qui n'est pas pratique à utiliser

→ On a utilisé la fonction map qui permet de changer cette échelle. Pour le moteur (tab[0]) on étalonne entre -90 et 90 les valeurs prises par le moteur tandis que pour le servo moteur (tab[1]) on étalonne entre 750 et 2250.

Le code complet de l'émetteur

```
1
2 #include <VirtualWire.h>
3 #include <MPU6050.h>
4 #include <I2Cdev.h>
5
6 MPU6050 mpu ;
7
8 int16_t ax, ay, az ;
9 int16_t gx, gy ,gz ;
10
11 void setup()
12 {
13
14   Serial.begin(9600);
15   vw_setup(2000);
16   mpu.initialize();
17   Serial.println(mpu.testConnection()? " YES " : " NO " );
18   delay(1000);
19   Serial.println("Capt values from sensor");
20   delay(1000);
21 }
22
23 void loop() {
24
25   mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
26   float a = map(ax, -17000, 17000, -90, 90);
27   float b = map(az, -1700, 1700, 750, 2250);
28   float tab[2];
29   tab[0] = a;
30   tab[1] = b;
31
32   vw_send((byte *) &tab, sizeof(tab));    ///On envoie le message
33   vw_wait_tx();                            ///On attend la fin de l'envoi
34 }
35
36
```

Partie récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

Partie récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Recevoir la liste envoyée par l'émetteur

Partie récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Recevoir la liste envoyée par l'émetteur
- Etalonner de nouveau pour que les valeurs du moteur soient entre 0 et 255

Partie récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Recevoir la liste envoyée par l'émetteur
- Etalonner de nouveau pour que les valeurs du moteur soient entre 0 et 255
- Faire tourner le moteur à la vitesse correspondant à l'angle d'inclinaison de la main

Partie récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés

Dans cette partie il fallait:

- Recevoir la liste envoyée par l'émetteur
- Etalonner de nouveau pour que les valeurs du moteur soient entre 0 et 255
- Faire tourner le moteur à la vitesse correspondant à l'angle d'inclinaison de la main
- Faire tourner le servo moteur d'un angle correspondant au second angle d'inclinaison de la main

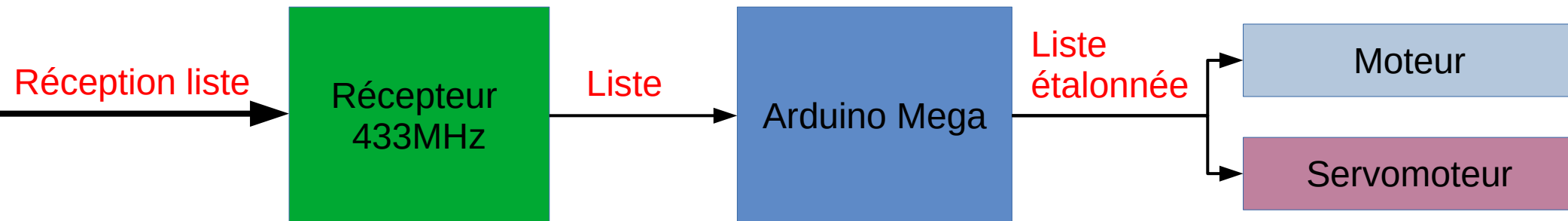
Organigramme récepteur

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés



Dans cette partie, on inclut les bibliothèques nécessaires:

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur/récepteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit le servo
10
11
12 float valeur[2];                   /// on définit le tableau
```

Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit
10
11
12 float valeur[2];                   /// on définit le
```

Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire
- ServoTimer2 pour contrôler le servo moteur

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit
10
11
12 float valeur[2];                   /// on définit le
```


Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire
- ServoTimer2 pour contrôler le servo moteur
- AFMotor pour contrôler le moteur avec le Motor Shield

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit le servo
10
11
12 float valeur[2];                   /// on définit le tableau
```

Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire
- ServoTimer2 pour contrôler le servo moteur
- AFMotor pour contrôler le moteur avec le Motor Shield

Ensuite on associe le moteur au pin 2

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit
10
11
12 float valeur[2];                   /// on définit le
```

Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire
- ServoTimer2 pour contrôler le servo moteur
- AFMotor pour contrôler le moteur avec le Motor Shield

Ensuite on associe le moteur au pin 2

On crée un objet 'myservo' qui sera le servo moteur

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit
10
11
12 float valeur[2];                   /// on définit le
```

Dans cette partie, on inclut les bibliothèques nécessaires:

- VirtualWire
- ServoTimer2 pour contrôler le servo moteur
- AFMotor pour contrôler le moteur avec le Motor Shield

Ensuite on associe le moteur au pin 2

On crée un objet 'myservo' qui sera le servo moteur

```
1 #include <AFMotor.h>
2
3           /// motor shield
4 #include "ServoTimer2.h"           /// servomoteur
5 #include <VirtualWire.h>           /// emetteur
6
7
8 AF_DCMotor motor(2);               /// le moteur
9 ServoTimer2 myservo;               /// on définit
10
11
12 float valeur[2];                   /// on définit le
```

Pour finir, on crée une liste de taille 2 pour recevoir les informations concernant le moteur et le servo moteur

- Initialisation de la vitesse de transfert, du pin du servo moteur ainsi que le pin sur lequel est attaché le récepteur.

```
13
14 void setup()
15 {
16
17     Serial.begin(9600);
18     myservo.attach(31);    /// Servo attaché a
19     vw_set_rx_pin(23);    /// Modification du
20
21     vw_setup(2000);
22     vw_rx_start();
23     Serial.println("Recepteur");
24
25 }
26
```

Initialisation de la vitesse de transfert, du pin du servo moteur ainsi que le pin sur lequel est attaché le récepteur.

En effet, le motor shield utilise le pin par défaut du récepteur. Il a fallu modifier son pin en lui en attribuant un libre (ici le 23)

```
13
14 void setup()
15 {
16
17     Serial.begin(9600);
18     myservo.attach(31);    /// Servo attaché a
19     vw_set_rx_pin(23);    /// Modification du
20
21     vw_setup(2000);
22     vw_rx_start();
23     Serial.println("Recepteur");
24
25 }
26
```

Attente de la réception d'un message

```
22  vw_rx_start();
23  Serial.println("Recepteur");
24
25  }
26
27  void loop()
28  {
29      byte taille_message = sizeof(valeur);
30
31      vw_wait_rx();
32      if (vw_get_message((byte *) &valeur, &taille_message))
33      {
34          Serial.print("valeur[0]=");
35          Serial.println(valeur[0]);
36          Serial.print("valeur[1]=");
37          Serial.println(valeur[1]);
38          moteur();
39          servo();
40      }
```


Attente de la réception
d'un message

Affichage des valeurs de
la liste

```
22  vw_rx_start();  
23  Serial.println("Recepteur");  
24  
25  }  
26  
27  void loop()  
28  {  
29      byte taille_message = sizeof(valeur);  
30  
31      vw_wait_rx();  
32      if (vw_get_message((byte *) &valeur, &taille_message))  
33      {  
34          Serial.print("valeur[0]=");  
35          Serial.println(valeur[0]);  
36          Serial.print("valeur[1]=");  
37          Serial.println(valeur[1]);  
38          moteur();  
39          servo();  
40      }
```


Attente de la réception
d'un message

Affichage des valeurs de
la liste

Lancement des fonctions
moteur() et servo()

```
22  vw_rx_start();  
23  Serial.println("Recepteur");  
24  
25  }  
26  
27  void loop()  
28  {  
29      byte taille_message = sizeof(valeur);  
30  
31      vw_wait_rx();  
32      if (vw_get_message((byte *) &valeur, &taille_message))  
33      {  
34          Serial.print("valeur[0]=");  
35          Serial.println(valeur[0]);  
36          Serial.print("valeur[1]=");  
37          Serial.println(valeur[1]);  
38          moteur();  
39          servo();  
40      }
```

Définition de la fonction `moteur()`:

```
46  
47 void moteur()  
48 {  
49     if (valeur[0] > 0)  
50     {  
51         motor.run(FORWARD);  
52         int a = map(valeur[0], 1, 90, 1, 255);  
53         motor.setSpeed(a);  
54     }  
55  
56     else  
57     {  
58         motor.run(BACKWARD);  
59         int a = map(valeur[0], -1, -90, 1, 255);  
60         motor.setSpeed(a);  
61     }  
62 }  
63
```

Définition de la fonction moteur():

- Si `valeur[0] > 0` alors le moteur tourne dans un sens à la valeur 'a' qui est étalonnée entre 1 et 255 (vitesse de rotation maximale du moteur)

```
46
47 void moteur()
48 {
49     if (valeur[0] > 0)
50     {
51         motor.run(FORWARD);
52         int a = map(valeur[0], 1, 90, 1, 255);
53         motor.setSpeed(a);
54     }
55
56     else
57     {
58         motor.run(BACKWARD);
59         int a = map(valeur[0], -1, -90, 1, 255);
60         motor.setSpeed(a);
61     }
62 }
63
```

Définition de la fonction moteur():

- Si `valeur[0] > 0` alors le moteur tourne dans un sens à la valeur 'a' qui est étalonnée entre 1 et 255 (vitesse de rotation maximale du moteur)
- Sinon, le moteur tourne dans l'autre sens à la vitesse 'a' étalonnée de la même manière

```
46
47 void moteur()
48 {
49     if (valeur[0] > 0)
50     {
51         motor.run(FORWARD);
52         int a = map(valeur[0], 1, 90, 1, 255);
53         motor.setSpeed(a);
54     }
55
56     else
57     {
58         motor.run(BACKWARD);
59         int a = map(valeur[0], -1, -90, 1, 255);
60         motor.setSpeed(a);
61     }
62 }
63
```

Définition de la fonction servo():

```
63  
64  
65 void servo()  
66 {  
67     int b = valeur[1];  
68     myservo.write(b);  
69 }  
70  
71
```

Définition de la fonction servo():

- Le servo moteur tourne de l'angle 'valeur[1]'

```
63  
64  
65 void servo()  
66 {  
67     int b = valeur[1];  
68     myservo.write(b);  
69 }  
70  
71
```

Définition de la fonction servo():

- Le servo moteur tourne de l'angle 'valeur[1]'

L'utilisation de VirtualWire empêche l'utilisation de la bibliothèque Servo c'est pourquoi on a utilisé ServoTimer2.

```
63  
64  
65 void servo()  
66 {  
67     int b = valeur[1];  
68     myservo.write(b);  
69 }  
70  
71
```

Définition de la fonction servo():

- Le servo moteur tourne de l'angle 'valeur[1]'

L'utilisation de VirtualWire empêche l'utilisation de la bibliothèque Servo c'est pourquoi on a utilisé ServoTimer2.

```
63  
64  
65 void servo()  
66 {  
67     int b = valeur[1];  
68     myservo.write(b);  
69 }  
70  
71
```

Autant les valeurs prises par le servo moteur avec Servo sont dans l'intervalle [0,180], autant avec ServoTimer2 les valeurs prises sont dans l'intervalle [750,2250]

Le code complet du récepteur (1)

```
1 #include <AFMotor.h>
2
3 // motor shield
4 #include "ServoTimer2.h" // servomotor
5 #include <VirtualWire.h> // emetteur recepteur 433Mhz
6
7
8 AF_DCMotor motor(2); // le moteur est en 2
9 ServoTimer2 myservo; // on définit le servo "myservo"
10
11
12 float valeur[2]; // on définit le tableau "valeur" pour tout le programme
13
14 void setup()
15 {
16
17   Serial.begin(9600);
18   myservo.attach(31); // Servo attaché au pin 31
19   vw_set_rx_pin(23); // Modification du pin par défaut du récepteur car déjà occupé par le motorshield
20
21   vw_setup(2000);
22   vw_rx_start();
23   Serial.println("Recepteur");
24 }
25
26
27 void loop()
28 {
29   byte taille_message = sizeof(valeur);
30
31   vw_wait_rx(); // On attend la reception d'un message
32   if (vw_get_message((byte *) &valeur, &taille_message))
33   {
34     Serial.print("valeur[0]=");
35     Serial.println(valeur[0]);
36     Serial.print("valeur[1]=");
37     Serial.println(valeur[1]);
38     moteur();
39     servo();
40   }
```

Le code complet du récepteur (2)

```
46
47 void moteur()
48 {
49     if (valeur[0] > 0)
50     {
51         motor.run(FORWARD);
52         int a = map(valeur[0], 1, 90, 1, 255);
53         motor.setSpeed(a);
54     }
55
56     else
57     {
58         motor.run(BACKWARD);
59         int a = map(valeur[0], -1, -90, 1, 255);
60         motor.setSpeed(a);
61     }
62 }
63
64
65 void servo()
66 {
67     int b = valeur[1];
68     myservo.write(b);
69 }
70
71
72
```

Problèmes rencontrés

Explication générale

Cahier des charges

Informatique

Problèmes rencontrés


Bien évidemment, comme dans tout projet, tout ne se passe pas exactement comme prévu et nous avons rencontré certains problèmes durant ce TIPE. Cette partie est consacrée à la façon dont ces problèmes ont été traités.

Au début de ce TIPE, on a réussi à contrôler la vitesse du moteur grâce au motor shield en envoyant des valeurs.

Cependant, en branchant le motor shield sur une carte Arduino Uno, tous les pins étaient occupés il était alors impossible de brancher le récepteur dessus.

Au début de ce TIPE, on a réussi à contrôler la vitesse du moteur grâce au motor shield en envoyant des valeurs.

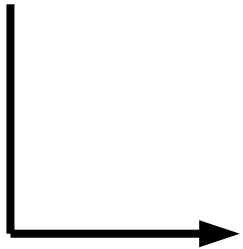
Cependant, en branchant le motor shield sur une carte Arduino Uno, tous les pins étaient occupés il était alors impossible de brancher le récepteur dessus.



Pour résoudre ce problème, on a dû acheter une carte Arduino Mega sur laquelle plus de pins sont disponibles. Ensuite on a modifié le pin par défaut du récepteur (vu [page 51](#)) car celui-ci était occupé par le motor shield.

Ensuite, pour envoyer des valeurs à la fois au moteur et au servo moteur et que ces derniers utilisent les bonnes valeurs, j'ai pensé à envoyer des valeurs pour le moteur et pour le servo très distinctes. Dans le but que le moteur identifie ses valeurs et de même pour le servo moteur. Cependant cette méthode ne fonctionnait pas très bien.

Ensuite, pour envoyer des valeurs à la fois au moteur et au servo moteur et que ces derniers utilisent les bonnes valeurs, j'ai pensé à envoyer des valeurs pour le moteur et pour le servo très distinctes. Dans le but que le moteur identifie ses valeurs et de même pour le servo moteur. Cependant cette méthode ne fonctionnait pas très bien.

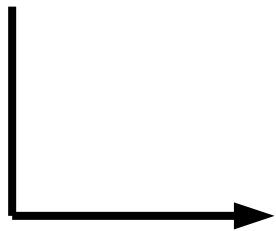


Pour régler ce problème, j'ai ensuite pensé à envoyer une liste comprenant deux termes: le premier correspondant aux valeurs que prendront le moteur et le second à celles que prendront le servo moteur.

Après quelques séances passées sur la maquette, on a décidé de tester notre programme avec la maquette.

Après quelques séances passées sur la maquette, on a décidé
de tester notre programme avec la maquette.
Hélas plus rien ne fonctionnait!

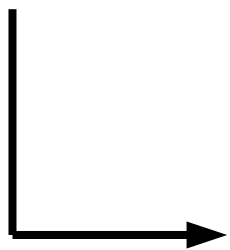
Après quelques séances passées sur la maquette, on a décidé de tester notre programme avec la maquette.
Hélas plus rien ne fonctionnait!



Après quelques vérifications, il s'est avéré que le problème venait de l'émetteur. Après une petite soudure faite par nos soins, tout fonctionnait parfaitement.

Le TIPE suivait son cours, nous continuions le programme lorsque tout à coup, le récepteur ne recevait qu'une partie des valeurs. Ce problème est sans doute celui qui nous a coûté le plus de temps.

Le TIPE suivait son cours, nous continuions le programme lorsque tout à coup, le récepteur ne recevait qu'une partie des valeurs. Ce problème est sans doute celui qui nous a coûté le plus de temps.



Après avoir vérifié, modifié plusieurs fois le programme, les branchements, l'émetteur/le récepteur, le problème n'était toujours pas résolu. On a alors décidé de changer de carte Arduino Mega. Cependant aucune Arduino Mega n'était disponible pour tester si le problème était bien là.

Nous avons donc commandé une nouvelle carte arduino Mega et pu vérifier que le problème était ainsi résolu.

Ma présentation est terminée, merci de m'avoir écouté avec attention.

Avez-vous des questions?