Université de Bretagne Occidentale

UFR Sciences et Techniques

Département Informatique

M2 SIC/SIAM

Projet Exploratoire

Thème

MISE EN PLACE D'UN DÉMONSTRATEUR CHAR À VOILE

Proposé par :

Bernt Weber (SplashElec)

Réalisé par :

Youcef HADJ KACI

Encadré par :

Goulven Guillou

Jean-Philippe Babau

Février 2013

Introduction :

Le Char à voile est un sport de vitesse qui se pratique en général sur de grandes plages de sable. La force de traction du véhicule est le vent, capté par une voile. Le char à voile présente des problématiques de pilotage proches de celle d'un navire à voilier.

Objectifs :

Le but du projet et de réaliser un petit modèle d'un char à voile et développer un pilote qui sera embarqué sur le véhicule afin de le contrôler.

On peut imaginer plusieurs applications pour ce pilote. Je me suis fixé les deux objectifs suivants :

- **Assurer la sécurité :** En faisant des virements, le pilote agit pour éviter la chute du char à voile sur l'un des cotés à cause de la force du vent.
- **Maintenir le cap :** En fixant un angle donné, le pilote contrôlera le char d'une manière à maintenir ce cap.

Materiel :

- **Carte Arduino Mega 2560 :** C'est une carte microcontrôleur basée sur l'ATmega1280. Voir le lien : http://arduino.cc/en/Main/ArduinoBoardMega

- **Carte UM6-LT :** C'est la centrale inertielle qui fournira toutes les mesures. Voir le lien : http://www.chrobotics.com/shop/orientation-sensor-um6-lt

- **GPS :** C'est un oem-em-406. Voir le lien : http://www.lextronic.fr/P1400-recepteur-gps-oem-em-406a.html

- **Servomotor :** Voir le lien : http://www.hobbyking.com/hobbyking/store/__9549__Turnigy_TG9e_9g_1_5kg_0_10sec_Eco_Micro_Servo.html

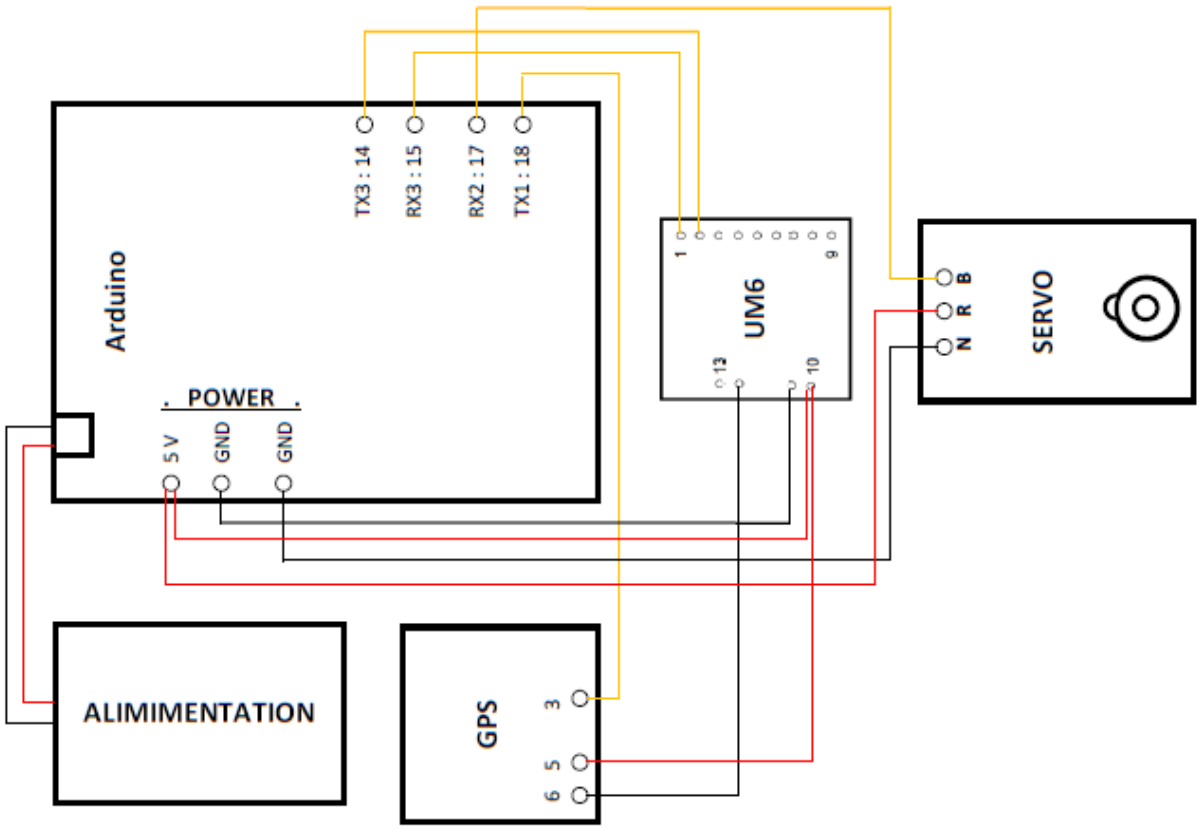
- **Alimentation :** Batterie 9V / 200 mA

- **Plateforme Lego :** Le char à voile est monté avec des lego (voir les images ci-dessous).

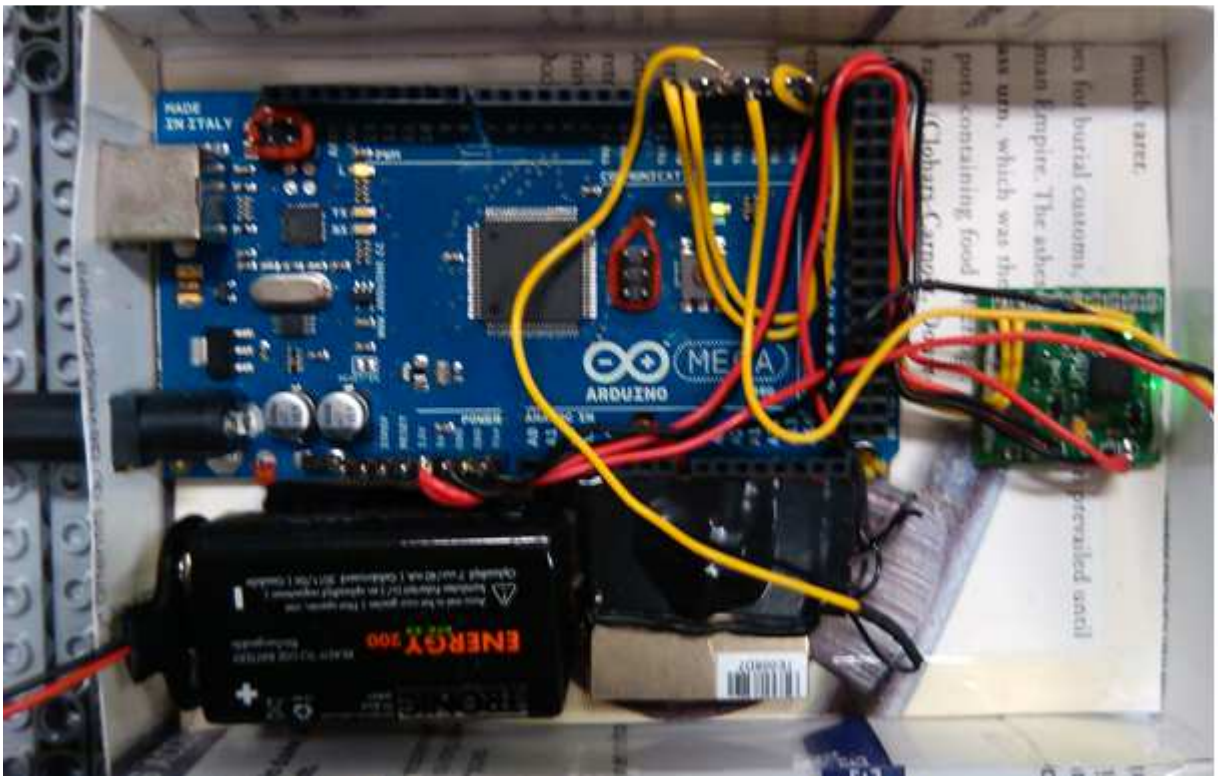
- **Cable USB :** Le cable sert à charger le programme sur la carte Arduino, il peut servir d'alimentation aussi. Voir le lien : http://www.dfrobot.com/index.php?route=product/product&product_id=134#.UQvxQPK0pQQ

Montage du char à voile :

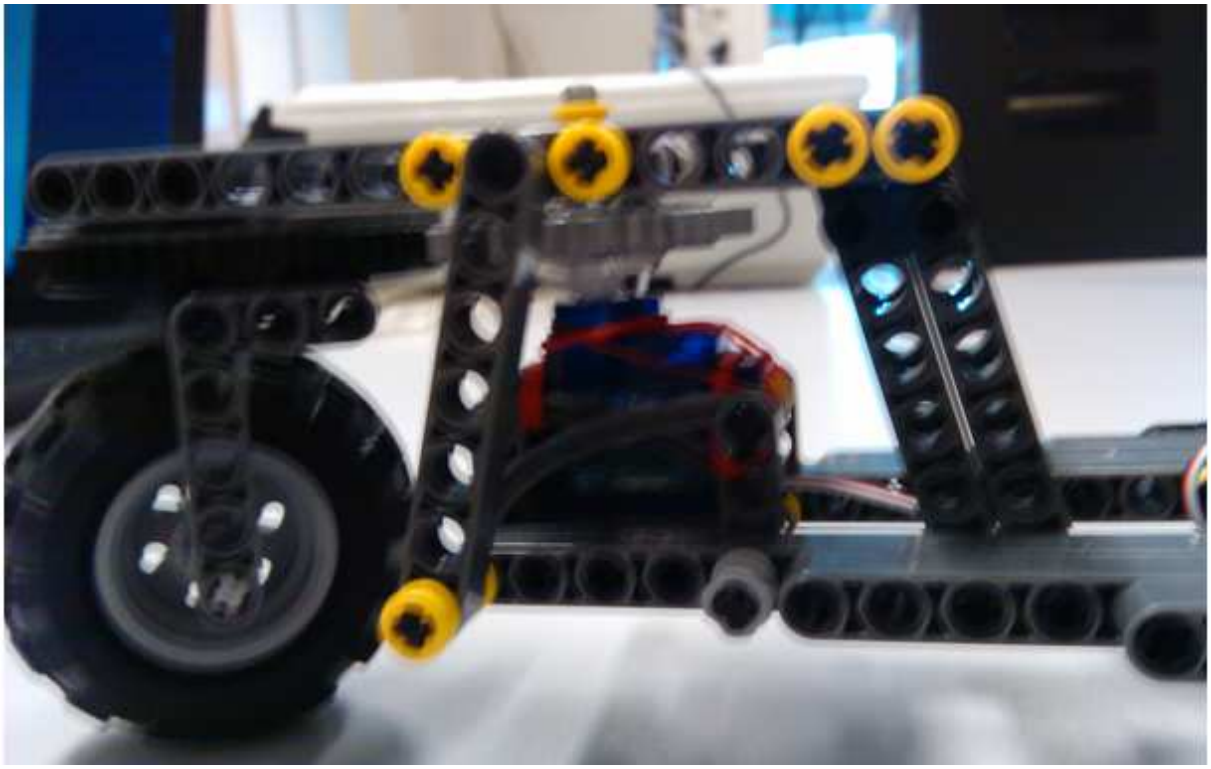
Le raccordement du matériel est donné sur le schéma ci-dessous :



Le tout est mis dans une boite en carton, comme le montre l'image suivante :



Le servomoteur est monté derrière la roue de devant :



La plateforme du char est faite à base de lego et la voile avec un tissu :

Logiciels :

- **Logiciel Arduino :** Un espace de développement intégré (EDI) dédié au langage Arduino et à la programmation des cartes Arduino. Voir le lien : http://arduino.cc/fr/Main/DebuterPresentationLogiciel

- **CH Robotics Serial Interface :** Pour visualiser les mesures fournis par la centrale UM6-LT. A télécharger sur le site : http://www.chrobotics.com/shop/orientation-sensor-um6-lt (Rousources/Software Downloads)

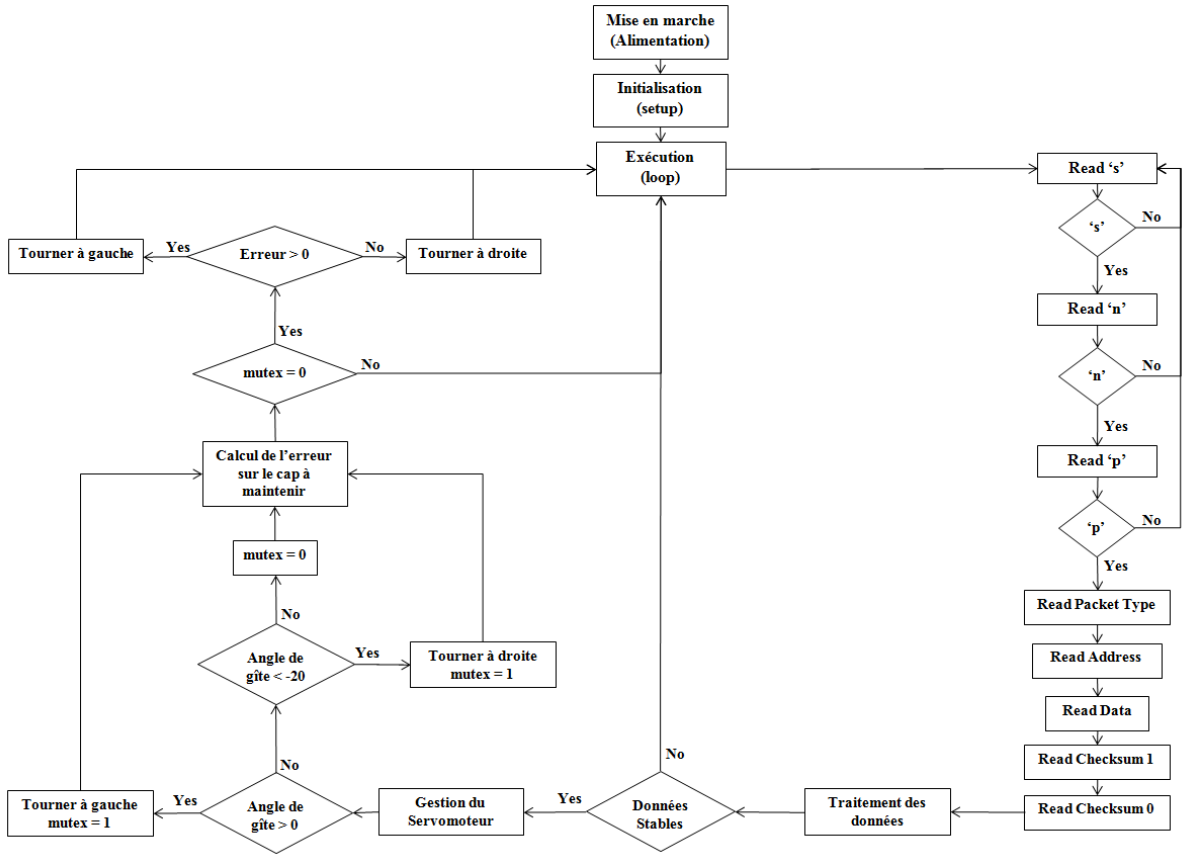
Pour utiliser le logiciel CH Robotics afin de visualiser les données, il faut charger le programme ci-dessous sur la carte Arduino.

```
void setup() {
  // initialize both serial ports:
  Serial.begin(115200);
  Serial3.begin(115200);
}

void loop() {
  // read from port 1, send to port 0:
  if (Serial3.available()) {
    int inByte = Serial3.read();
    Serial.write(inByte);
  }

  // read from port 0, send to port 1:
  if (Serial.available()) {
    int inByte = Serial.read();
    Serial3.write(inByte);
  }
}
```

Algorithme :



- Algorithme du pilote -

Code :

```
#include <Servo.h>

#define STATE_ZERO         0
#define STATE_S            1
#define STATE_SN           2
#define STATE_SNP          3
#define STATE_PT           4
#define STATE_READ_DATA    5
#define STATE_CHK1         6
#define STATE_CHK0         7

#define PT_HAS_DATA  0b10000000
#define PT_IS_BATCH  0b01000000
#define PT_COMM_FAIL 0b00000001

#define UM6_EULER_PHI_THETA 0x62      // Data register for angles
#define UM6_GYRO_PROC_XY 0x5c         // Data register for angular rates
#define GYRO_SCALE_FACTOR  0.0610352  // Scale factor to convert register data to Degrees per
Second
#define EULER_SCALE_FACTOR  0.0109863 // Scale factor to convert register data to Degrees

#define DATA_BUFF_LEN  16
#define BAUD 115200

byte aPacketData[DATA_BUFF_LEN];
byte c = 0;
int nState = 0;
int n = 0;
int nDataByteCount = 0;

struct IMU{
  int in;
}phi, theta, psi; //Roll, pitch and yaw, respectively.

struct GYRO{
  int in;
}girX, girY, girZ; // angular rates relative to the axes X, Y and Z, respectively.

typedef struct {
  boolean HasData;
  boolean IsBatch;
  byte BatchLength;
  boolean CommFail;
  byte Address;
  byte Checksum1;
  byte Checksum0;
  byte DataLength;
} UM6_PacketStruct ;

UM6_PacketStruct UM6_Packet;

Servo myservo;   // create servo object to control a servo motor
int pin = 18;    // arduino pin
int pos = 80;    // servo position : 80° corresponds to the central axis of sand yacht
                 // (the wheel turns between 0° and 160°)
int pos_new;
int desired_angle = 40; // between 0° and 180°
int error_angle = 0;
int mutex = 0;   // priority management
int WaitCounter; // wait counter for stable data
float P = 2.5;   // proportional controller
float D = 0.1;   // derivative controller

void setup(){
  myservo.attach(pin);   // attaches the servo on pin 18 to the servo object
  myservo.write(pos);    // center the wheel at the axis
  Serial.begin(BAUD);    // initialize serial port 0
  Serial3.begin(BAUD);   // initialize serial port 3
  WaitCounter = 0;
}

void loop(){
  n = Serial3.available();
  if (n > 0){
```

```
    c = Serial3.read();
    switch(nState){
      case STATE_ZERO : // Begin. Look for 's'.
        Reset();
        if (c == 's'){ //0x73 = 's'
          nState = STATE_S;
        } else {
          nState = STATE_ZERO;
        }
        break;
      case STATE_S : // Have 's'. Look for 'n'.
        if (c == 'n'){ //0x6E = 'n'
          nState = STATE_SN;
        } else {
          nState = STATE_ZERO;
        }
        break;
      case STATE_SN : // Have 'sn'. Look for 'p'.
        if (c == 'p'){ //0x70 = 'p'
          nState = STATE_SNP;
        } else {
          nState = STATE_ZERO;
        }
        break;
      case STATE_SNP : // Have 'snp'. Read PacketType and calculate DataLength.
        UM6_Packet.HasData = 1 && (c & PT_HAS_DATA);
        UM6_Packet.IsBatch = 1 && (c & PT_IS_BATCH);
        UM6_Packet.BatchLength = ((c >> 2) & 0b00001111);
        UM6_Packet.CommFail = 1 && (c & PT_COMM_FAIL);
        nState = STATE_PT;
        if (UM6_Packet.IsBatch){
          UM6_Packet.DataLength = UM6_Packet.BatchLength * 4;
        } else {
          UM6_Packet.DataLength = 4;
        }
        break;
      case STATE_PT : // Have PacketType. Read Address.
        UM6_Packet.Address = c;
        nDataByteCount = 0;
        nState = STATE_READ_DATA;
        break;
      case STATE_READ_DATA : // Read Data. (UM6_PT.BatchLength * 4) bytes.
        aPacketData[nDataByteCount] = c;
        nDataByteCount++;
        if (nDataByteCount >= UM6_Packet.DataLength){
          nState = STATE_CHK1;
        }
        break;
      case STATE_CHK1 : // Read Checksum 1
        UM6_Packet.Checksum1 = c;
        nState = STATE_CHK0;
        break;
      case STATE_CHK0 : // Read Checksum 0
        UM6_Packet.Checksum0 = c;
        nState = STATE_ZERO;
        ProcessPacket(); // Entire packet consumed. Process packet
        break;
    }
  }
}

void ProcessPacket(){
  switch(UM6_Packet.Address){ // Get the angles values (phi, theta and psi)
    case UM6_EULER_PHI_THETA :
      phi.in = ((aPacketData[0] << 8) | aPacketData[1]) * EULER_SCALE_FACTOR;
      theta.in = ((aPacketData[2] << 8) | aPacketData[3]) * EULER_SCALE_FACTOR;
      psi.in = ((aPacketData[4] << 8) | aPacketData[5]) * EULER_SCALE_FACTOR;
      ManageServo();
      break;
    case UM6_GYRO_PROC_XY :  // Get the angular rates
      if (UM6_Packet.HasData && !UM6_Packet.CommFail){
        girX.in = ((aPacketData[0] << 8) | aPacketData[1]) * GYRO_SCALE_FACTOR;
        girY.in = ((aPacketData[2] << 8) | aPacketData[3]) * GYRO_SCALE_FACTOR;
        if (UM6_Packet.DataLength > 4){
          girZ.in = ((aPacketData[4] << 8) | aPacketData[5]) * GYRO_SCALE_FACTOR;
        }
      }
```

```
      break;
  }
  PrintDebug();
}

void ManageServo(){
  if (WaitCounter < 800){ // wait to obtain stable values of data
    WaitCounter++;
    return;
  }

  // Manage the servo motor to prevent the fall of sand yacht
  if (theta.in > 0){ // Turn to left
    mutex = 1;
    myservo.write(myservo.read() - 45);
  } else if (theta.in < -20){ // Turn to right
    mutex = 1;
    myservo.write(myservo.read() + 45);
  } else {
    mutex = 0;
  }

  // Manage servo motor to obtain desired angle
  error_angle = psi.in - desired_angle;
  if (error_angle < - 180){
    error_angle = error_angle + 360;
  }

  if (error_angle > 0){ // Turn to left
    if (girZ.in > 0){
      pos_new = pos - P * error_angle + D * girZ.in;
    } else{
      pos_new = pos - P * error_angle - D * girZ.in;
    }
  } else{ // Turn to right
    if (girZ.in > 0){
      pos_new = pos - P * error_angle + D * girZ.in;
    } else{
      pos_new = pos - P * error_angle - D * girZ.in;
    }
  }

  if (mutex == 0){
    myservo.write(pos_new);
  }
}

void Reset(){
  UM6_Packet.HasData = false;
  UM6_Packet.IsBatch = false;
  UM6_Packet.BatchLength = 0;
  UM6_Packet.CommFail = false;
  UM6_Packet.Address = 0;
  UM6_Packet.Checksum1 = 0;
  UM6_Packet.Checksum0 = 0;
  UM6_Packet.DataLength = 0;
}

void PrintDebug(){
  Serial.print("PHI = ");
  Serial.print(phi.in);
  Serial.print(" THETA = ");
  Serial.print(theta.in);
  Serial.print(" PSI = ");
  Serial.print(psi.in);
  Serial.print(" girX = ");
  Serial.print(girX.in);
  Serial.print(" girY = ");
  Serial.print(girY.in);
  Serial.print(" girZ = ");
  Serial.print(girZ.in);
  Serial.print(" ERROR_ANGLE = ");
  Serial.print(error_angle);
  Serial.print(" POS_NEW = ");
  Serial.print(pos_new);
  Serial.println(".");
}
```