Arthur **_Rapport de séance n°11_** Projet Vitis

31/01/23

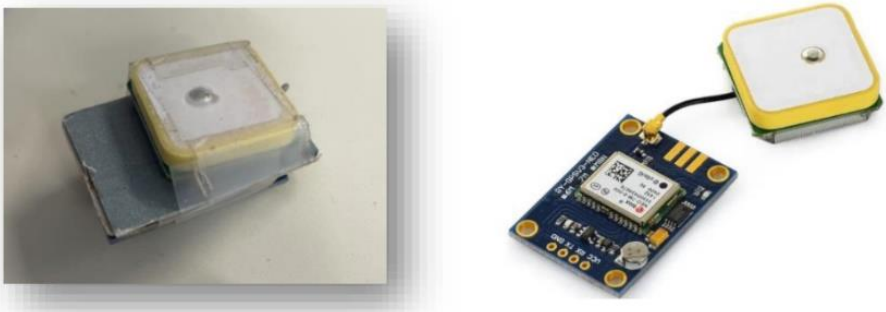
Objectifs de la séance :

Séance spéciale car à distance donc que du code :

- Obtenir ma position actuelle avec le nouveau GPS

Réalisations :

- Nous avons reçu un nouveau GPS Ublox GPS Module suite au dernier GPS NEO 6-M (cf : Rapport de séance n°3), le GPS est sur le papier plus précis mais il y a moins de documentation le concernant.

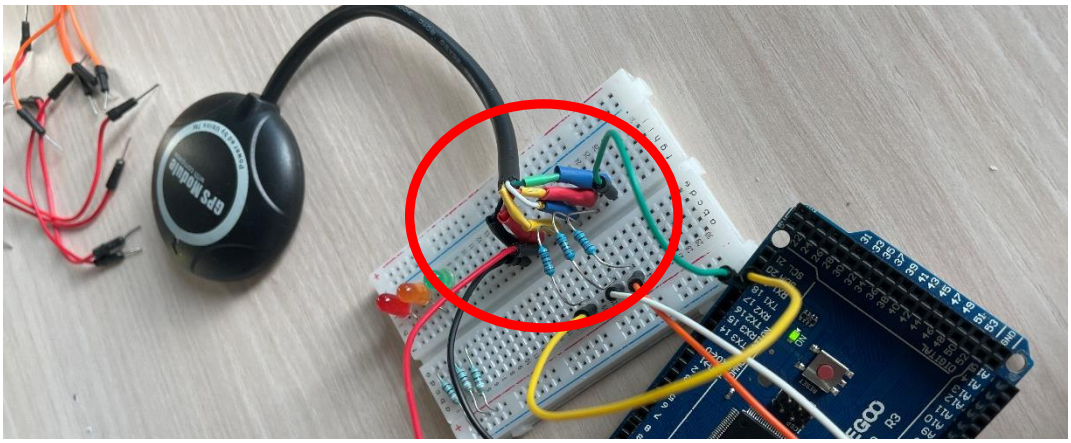


NEO 6-M

- La première étape fut de connecter chaque fil du GPS à un connecteur Arduino afin de le brancher à la carte.

Ublox Module Compass

- Après lecture de la datasheet, je constate que le GPS nécessite que l'on communique en 3.3V donc je dois placer des résistances sur chaque pin d'informations.



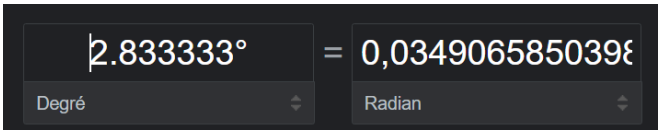
- Avant de rentrer le code dans l'Arduino il faut calibrer le GPS en fonction de notre position géographique. Sur terre il existe le Nord géographique et le Nord magnétique, et en fonction de notre position il peut il y avoir un écart entre les deux suivant notre position sur la planète. Pour augmenter la précision du GPS il est donc important de bien calibrer dans notre code.

- Pour cela on commence par trouver l'inclinaison magnétique de notre ville (https://www.magnetic-declination.com/)

- Puis on la convertie en degrés pour que ce soit exploitable

```
2° 50' 0"
= 2° + 50'/60 + 0"/3600
= 2.833333°
```

- Enfin en radian car l'angle est en radian dans le code final



- Maintenant il ne reste plus qu'à aller en extérieur (car le GPS ne traverse pas les murs) et tenter d'obtenir notre position. Malheureusement je n'ai pas réussi à obtenir ma position, je laisse le capteur en suspens.

```ino
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>

/*
   This sample sketch demonstrates the normal use of a TinyGPS++ (TinyGPSPlus) object.
   It requires the use of SoftwareSerial, and assumes that you have a
   9600-baud serial GPS device hooked up on pins 8(rx) and 9(tx) and a HMC5883 Magnetic Compass
   connected to the SCL/SDA pins.
*/

static const int RXPin = 8, TXPin = 9;
static const uint32_t GPSBaud = 9600;

// Assign a Uniquej ID to the HMC5883 Compass Sensor
Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the NEO-6m GPS module
SoftwareSerial ss(RXPin, TXPin);

void displaySensorDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void setup()
{
  Serial.begin(9600);
  ss.begin(GPSBaud);

  Serial.println(F("Simple Test with TinyGPS++ and attached NEO-6M GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. "));
Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println();
  displaySensorDetails();
}

void loop()
{
  // This sketch displays information every time a new sentence is correctly encoded from the GPS Module.
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
```

```
      displayGpsInfo();
  }

  void displayGpsInfo()
  {
    // Prints the location if lat-lng information was recieved
    Serial.print(F("Location: "));
    if (gps.location.isValid())
    {
      Serial.print(gps.location.lat(), 6);
      Serial.print(F(","));
      Serial.print(gps.location.lng(), 6);
    }
    // prints invalid if no information was recieved in regards to location.
    else
    {
      Serial.print(F("INVALID"));
    }

    Serial.print(F("  Date/Time: "));
    // prints the recieved GPS module date if it was decoded in a valid response.
    if (gps.date.isValid())
    {
      Serial.print(gps.date.month());
      Serial.print(F("/"));
      Serial.print(gps.date.day());
      Serial.print(F("/"));
      Serial.print(gps.date.year());
    }
    else
    {
      // prints invalid otherwise.
      Serial.print(F("INVALID"));
    }

    Serial.print(F(" "));
    // prints the recieved GPS module time if it was decoded in a valid response.
    if (gps.time.isValid())
    {
      if (gps.time.hour() < 10) Serial.print(F("0"));
      Serial.print(gps.time.hour());
      Serial.print(F(":"));
      if (gps.time.minute() < 10) Serial.print(F("0"));
      Serial.print(gps.time.minute());
      Serial.print(F(":"));
      if (gps.time.second() < 10) Serial.print(F("0"));
      Serial.print(gps.time.second());
      Serial.print(F("."));
      if (gps.time.centisecond() < 10) Serial.print(F("0"));
      Serial.print(gps.time.centisecond());
    }
    else
    {
      // Print invalid otherwise.
      Serial.print(F("INVALID"));
    }
    Serial.println();
    if(mag.begin())
    {
      displayCompassInfo();
```

```
  }
}

void displayCompassInfo()
{
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("
");Serial.println("uT");

  // Hold the module so that Z is pointing 'up' and you can measure the heading with x&y
  // Calculate heading when the magnetometer is level, then correct for signs of axis.
  float heading = atan2(event.magnetic.y, event.magnetic.x);

  // Once you have your heading, you must then add your 'Declination Angle', which is the
'Error' of the magnetic field in your location.
  // Find yours here: http://www.magnetic-declination.com/
  // Mine is: -13* 2' W, which is ~13 Degrees, or (which we need) 0.22 radians
  // If you cannot find your Declination, comment out these two lines, your compass will be
slightly off.
  float declinationAngle = 0.05;
  heading += declinationAngle;

  // Correct for when signs are reversed.
  if(heading < 0)
    heading += 2*PI;

  // Check for wrap due to addition of declination.
  if(heading > 2*PI)
    heading -= 2*PI;

  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI;

  Serial.print("Heading (degrees): "); Serial.println(headingDegrees);

  delay(500);
}
```