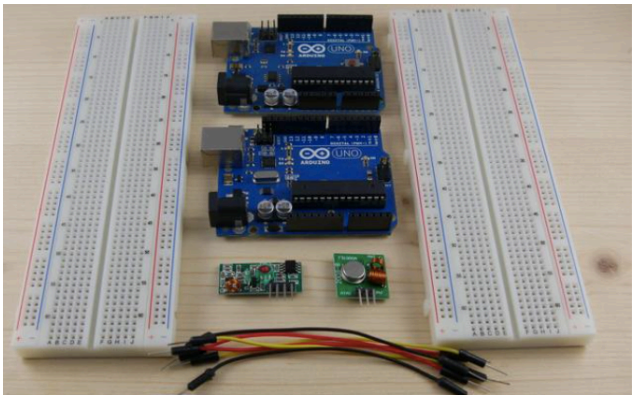
Projet Arduino
Voiture robotisée & contrôlée par radio fréquence

Bernard Anwar (en collaboration avec Chillat Quentin)

Rapport Semaine 2 (le 22/12/2023)

Matériels utilisés

- Deux cartes Arduino UNO (et deux câbles USB),
- Deux modules radio 433MHz (un émetteur et un récepteur),
- Deux plaques d'essai et des fils pour câbler notre montage,
- Accéléromètre



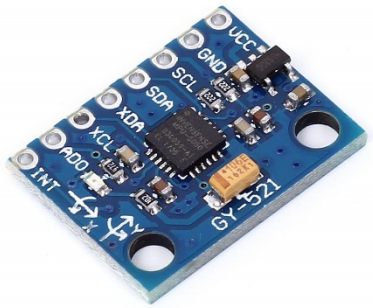
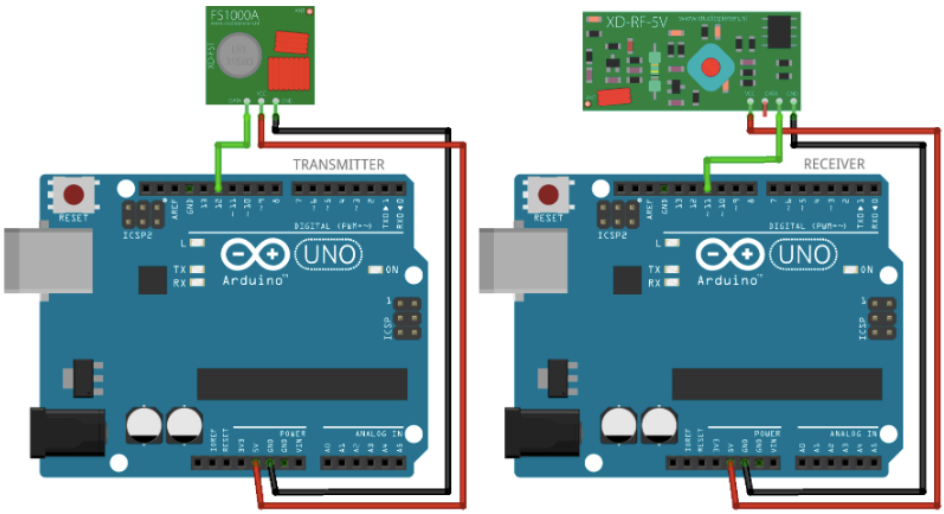


Schéma du câblage



Commencer par câbler la broche VCC du module radio à l'alimentation 5V de la carte Arduino au moyen d'un fil.
Reliera ensuite la broche GND du module radio à la broche GND de la carte Arduino.
Câbler les broches de communication :
- le récepteur, la sortie de communication doit être reliée à la broche D11
- l'émetteur, la sortie de communication doit être reliée à la broche D12

Programme de communication entre émetteur et récepteur

Utilisation de la librairie **VirtualWire**

Codage côté émission

```
#include <VirtualWire.h>
#include <VirtualWire_Config.h>
#define TxPin 12

void setup() {
  // put your setup code here, to run once:
  pinMode(3,OUTPUT);
  Serial.begin(9600);
  vw_set_tx_pin(TxPin);
  vw_setup(2000);

}

void loop() {
  // put your main code here, to run repeatedly:
  const char *msg="Bonsoir tous le monde";
  digitalWrite(3,LOW);
  vw_send((uint8_t *)msg,strlen(msg));
  vw_wait_tx();
  digitalWrite(3,HIGH);
  delay(200);
}
```

Codage côté réception

```
#include <VirtualWire.h>
#include <VirtualWire_Config.h>
#define RxPin 11

void setup() {
  // put your setup code here, to run once:
  pinMode(3,OUTPUT);
  Serial.begin(9600);
  vw_set_rx_pin(RxPin);
  vw_setup(2000);
  vw_rx_start();
}

void loop() {
  // put your main code here, to run repeatedly:
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  if(vw_get_message(buf,&buflen)){
    int i;
    digitalWrite(3,LOW);
    Serial.print("From transistter : ");
    for(i=0;i<buflen;i++){
      Serial.print(char(buf[i]));
    }

    Serial.println("");
    digitalWrite(3,HIGH);
  }

}
```

Adaptation des programmes de communication entre émetteur et récepteur pour prendre en compte les données de l'accéléromètre

Utilisation des librairies **VirtualWire, Wire et MPU6050** (pour simplification du programme).

Codage côté émission

```
#include <VirtualWire.h>
#include <Wire.h>
#include <MPU6050.h>
#define TxPin 12
MPU6050 mpu;
void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
  pinMode(3, OUTPUT);

  vw_set_tx_pin(TxPin);
  vw_setup(2000);
}

void loop() {
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);

  // Construction du message en format texte avec trois entiers signés sur 16 bits
  String message = String(ax) + "," + String(ay) + "," + String(az);
  Serial.println(ax < -8000&& ay < 1000);
  Serial.println(ax > 8000&& ay < 1000);
  // Affichage du message sur le moniteur série
  Serial.print("Sending: ");
  Serial.println(message);
  digitalWrite(3, LOW);
  // Obtention du pointeur de chaîne constante pour le message
  const char *messageChar = message.c_str();
  // Envoi du message via la communication RF
  vw_send((uint8_t *)messageChar, strlen(messageChar) + 1);
  vw_wait_tx();
  digitalWrite(3, HIGH);
  // Attente avant d'envoyer le prochain message
  delay(1000);
}
```

Codage côté réception

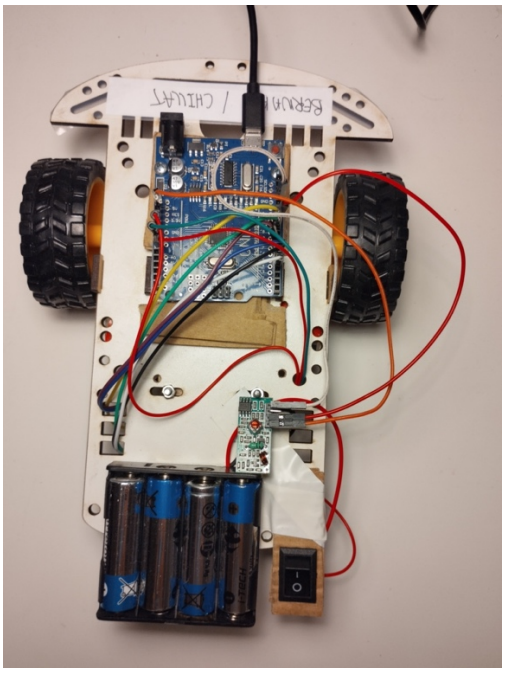
```
#include <VirtualWire.h>
#include <VirtualWire_Config.h>

#define RxPin 11

const int ENA=9;
const int IN1=4;
const int IN2=5;

const int ENB=10;
const int IN3=6;
const int IN4=7;


int16_t accelerometer_x, accelerometer_y, accelerometer_z;

void setup() {
  Serial.begin(9600);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);

  analogWrite(ENA,255);
  analogWrite(ENB,255);
  //cli(); // Désactiver les interruptions
  vw_set_rx_pin(RxPin);
  vw_setup(2000);
  vw_rx_start();
  //sei(); // Réactiver les interruptions
}

void loop() {
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  if (vw_get_message(buf, &buflen)) {
    buf[buflen] = '\0'; // Ajouter la terminaison de chaîne
    String receivedMessage = String((char *)buf);

    Serial.print("Transmetteur: ");
    Serial.println(receivedMessage);

    // Analyser les données reçues (assumant qu'elles sont au format "x,y,z")
    sscanf(receivedMessage.c_str(), "%d,%d,%d", &accelerometer_x, &accelerometer_y,
&accelerometer_z);

    if (accelerometer_x < 1000 && accelerometer_y < -8000) {
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
    } else if (accelerometer_x < 1000 && accelerometer_y > 8000) {
```

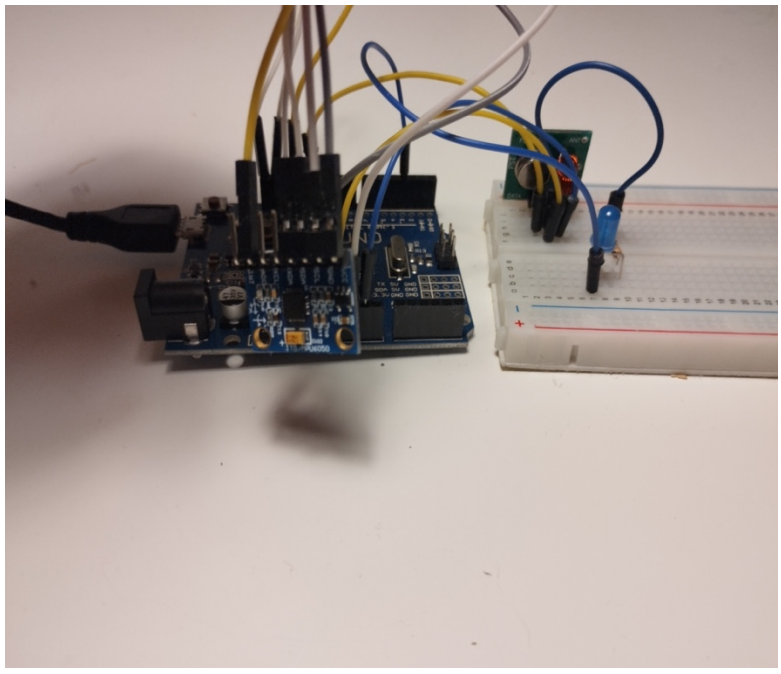
```
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
   } else if (accelerometer_x > 8000 && accelerometer_y < 1000) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
   } else if (accelerometer_x < -8000 && accelerometer_y < 1000) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
   } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
   }
  }
}
```

Tests effectués :

Vérification que les formats des données émissent par l'accéléromètre soient conformes lors de leur réception ➔ tests réussis

Vérification que les directives données par l'accéléromètre soient correctement réalisées par le système embarqué de la voiture ➔ seul le moteur B réagi correctement. (la roue du moteur A ne tourne pas)





Réalisation à prévoir pour la semaine 3

Correction du problème accéléromètre sur le moteur A.
Ajout d'une antenne.
Alimentation de la carte Arduino côté transmetteur.

Fixation du transmetteur sur un gant.