

Leçon 3 Télécommande Infrarouge



Points clés de la leçon

Le contrôle par infrarouge est une méthode largement répandue. Le robot est équipé d'un récepteur infrarouge et permet donc un tel contrôle.

Sommaire:

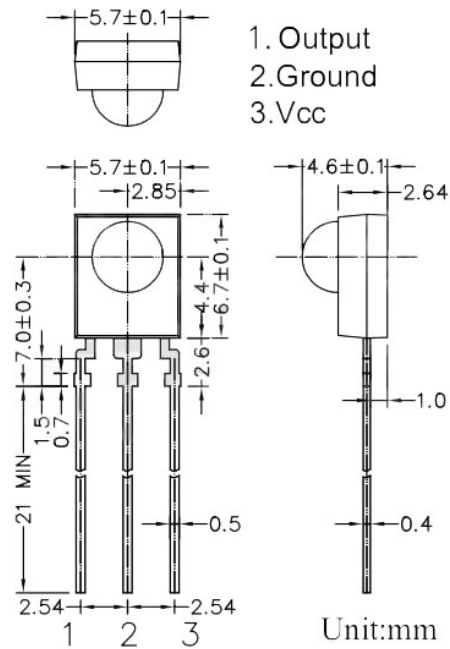
- ◆ Comprendre la télécommande infrarouge et le récepteur
- ◆ Comprendre le principe de fonctionnement

Matériel:

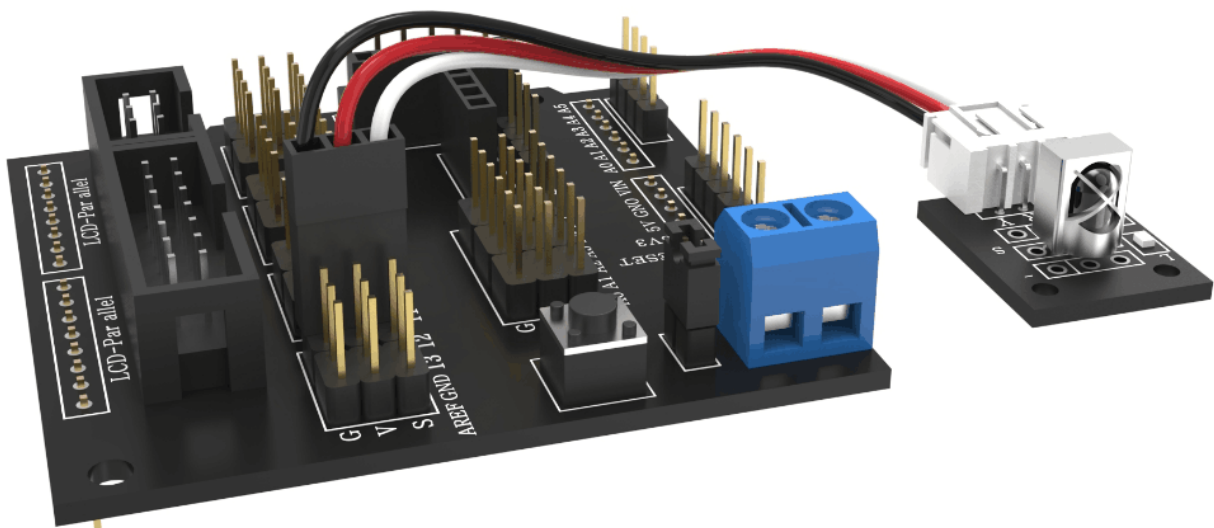
- ◆ Le robot
- ◆ Un câble USB
- ◆ Le module de reception et la télécommande infrarouge

I . Télécommandes et récepteurs infrarouges

Schéma d'un récepteur infrarouge :



Connexion du récepteur sur le robot:



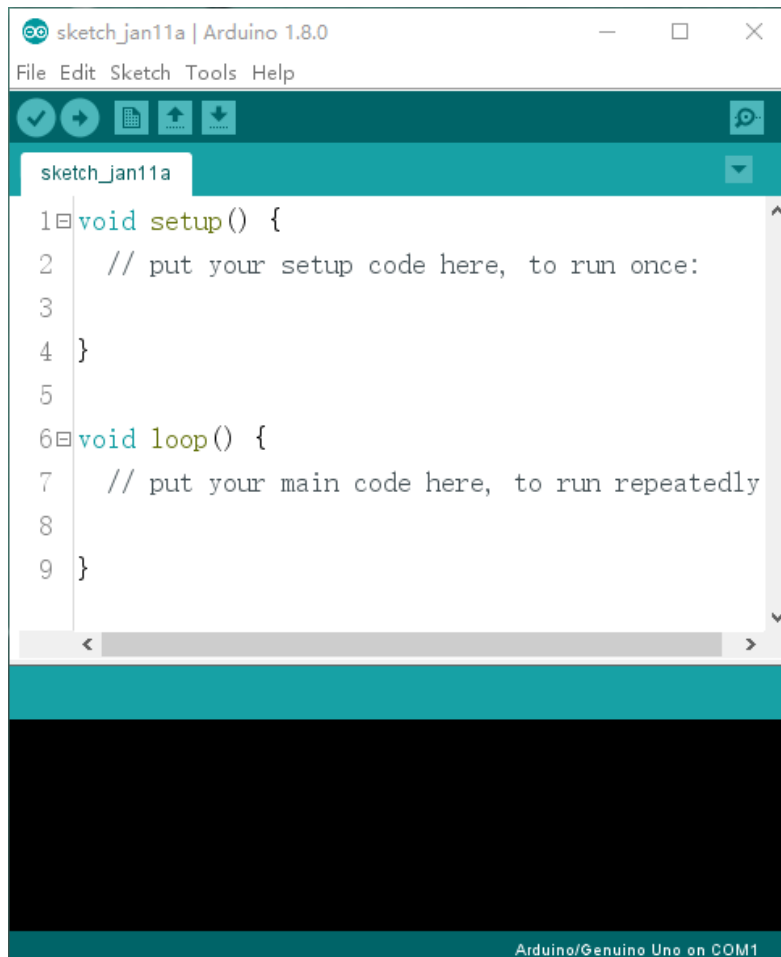
Télécommande IR:



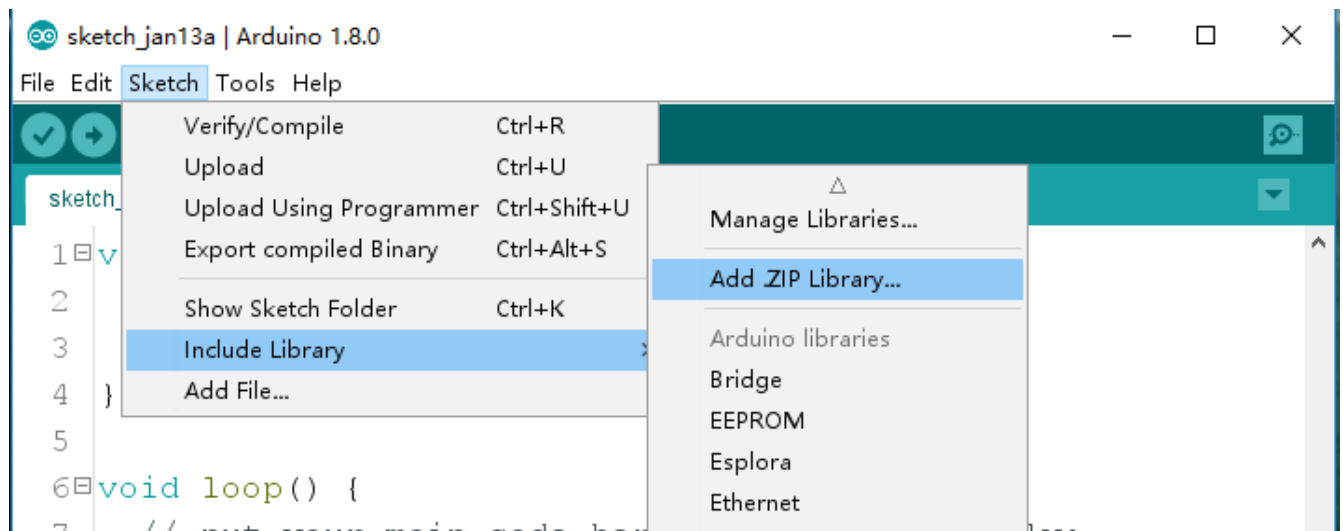
II. Tester le programme

Ce programme utilise une bibliothèque dont il convient de faire l'installation au préalable.

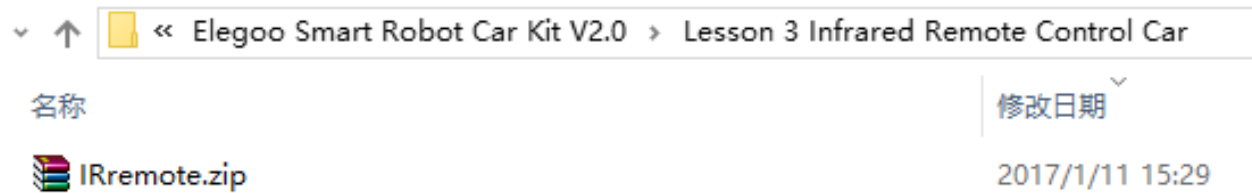
Ouvrez l'IDE Arduino



Cliquez sur Sketch—Include Library—Add .ZIP Library...—et sélectionnez la bibliothèque comme ci-dessous :

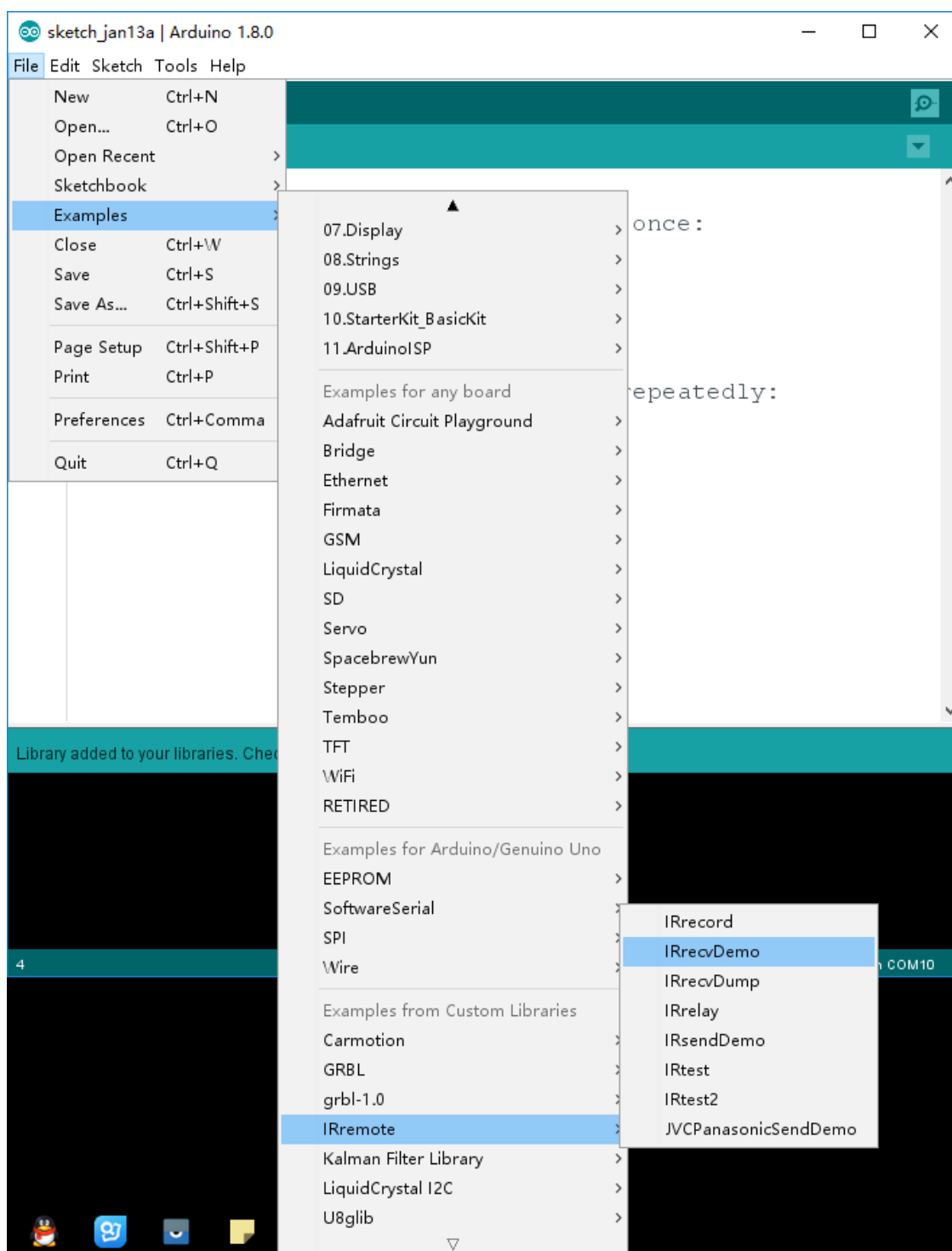


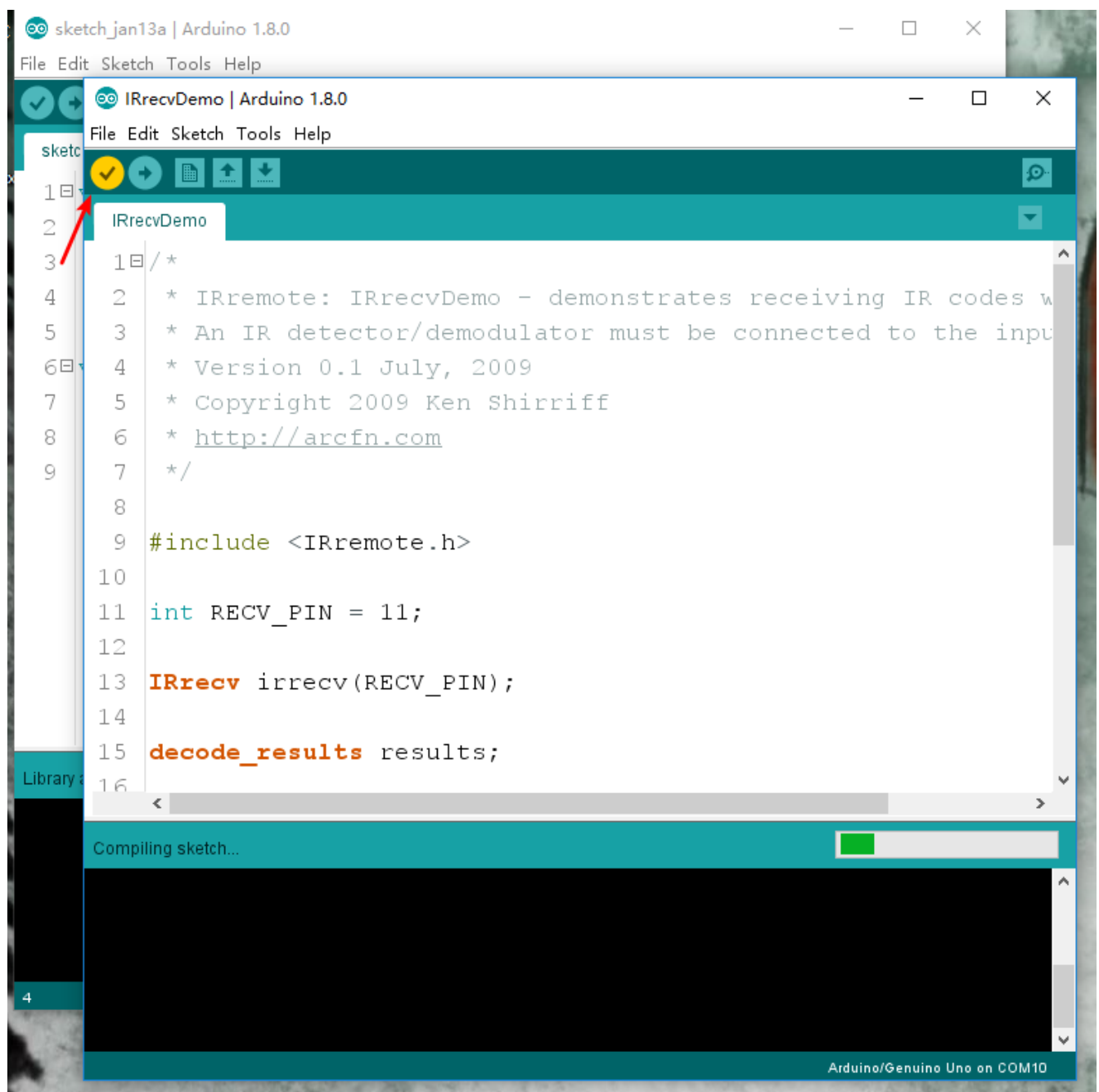
Le nom de la bibliothèque au format zip doit être IRremote.zip



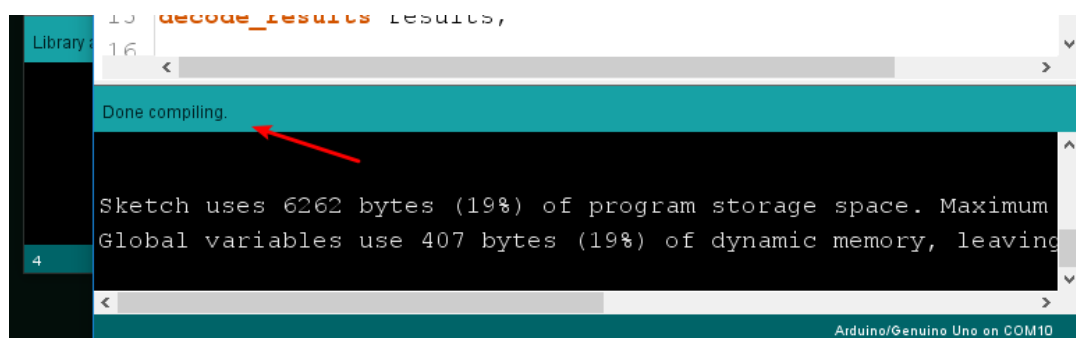
Doit être compilé avec cette bibliothèque qui est spécialement modifiée.

Choisissez un exemple IRremote :






A l'ouverture, vous devriez voir apparaître le message "done compiling". Si ce n'est pas le cas, cela signifie que la bibliothèque IRremote n'est pas correctement installée. Répétez la procédure d'installation en ajoutant à nouveau la bibliothèque IRremote.



Ouvrez le fichier "infrared_Blink\infrared_Blink.ino"

v ↑ « Lesson 3 Infrared Remote Control Car » infrared_Blink		
名称	修改日期	类型
 infrared_Blink.ino	2017/1/5 11:57	Ardui

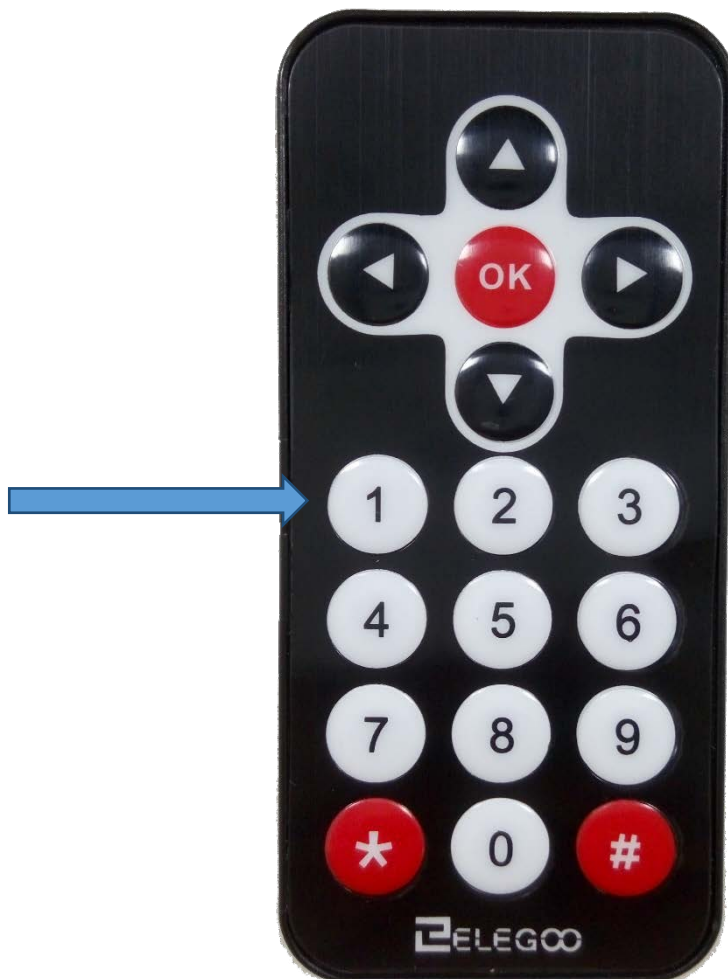
Le code est le suivant:

```
#include <IRremote.h> //Infrared Library
int receiverpin = 12; //Infrared signal receiving pin
int LED=13; //define LED pin
volatile int state = LOW; //define default input mode
unsigned long RED;
#define L 16738455
IRrecv irrecv(receiverpin); //initialization
decode_results results; //Define structure type
void setup() {
    pinMode(LED, OUTPUT); //initialize LED as an output
    Serial.begin(9600); // debug output at 9600 baud
    irrecv.enableIRIn(); // Start receiving
}
void stateChange()
{
    state = !state;
    digitalWrite(LED, state);
}
void loop() {
    if (irrecv.decode(&results))
    {
        RED=results.value;
        Serial.println(RED);
        irrecv.resume(); // Receive the next value
        delay(150);
        if(RED==L)
```

```
{  
  stateChange();  
}  
}  
}
```

Transférez le code dans la carte UNO, débranchez le câble USB, posez le robot au sol. Allumez-le, vous pouvez maintenant contrôler les mouvements du robot avec la télécommande.

Appuyez sur la touche 1 de la télécommande en pointant le robot. Observez la voiture, et vous verrez led s'éteindre.



III. Principes

1. Principe de fonctionnement

Le contrôle universel par infrarouge se divise en deux parties : l'émission et la réception d'un signal.

Les signaux qui sont émis par la télécommande sont une série d'impulsions codées en binaire. Le signal est modulé en fréquence pour ne pas interférer avec les signaux d'autres systèmes infrarouges.

Le récepteur décode le signal pour en extraire le codage binaire.

La faiblesse du signal, le gain, l'amplification, le filtrage ... sont gérés par des systèmes intégrés aux émetteurs et récepteurs et ce de manière automatique, sans besoin de réglage par l'utilisateur.

2. Protocole d'échange de données

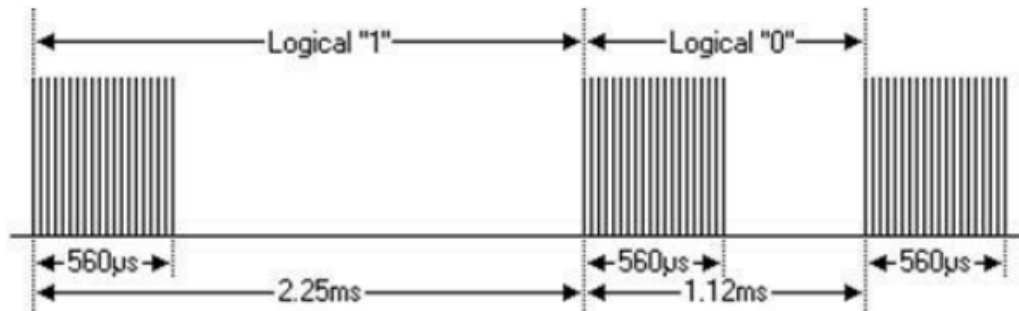
L'échange de données suit un protocole nommé NEC.

Next, let's learn what NEC protocol is.

Caractéristiques :

- (1) 8 bits pour l'adresse bits, 8 bits pour la commande
- (2) Bits d'adresses et bits de commandes sont envoyés deux fois afin de garantir la fiabilité de l'échange.
- (3) Modulation de la position de la pulsation
- (4) Fréquence de la porteuse 38kHz
- (5) Temporisation des bits 1.125ms or 2.25ms

Définition des 0 et 1 logiques :



Protocole :

Déclenchement instantané de la pulsation de transmission :



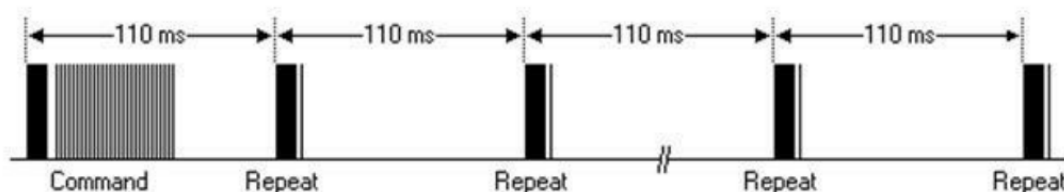
Le protocole utilisé est dit "protocole par envoi de bits non significatifs" en premier : protocole LSB (least-significant bit) .

L'adresse de l'instruction ci-dessus est 0x59

L'ordre de l'instruction ci-dessus est 0x16

Une instruction débute par un état haut de 9ms puis un état bas de 4,5ms. Suivent l'adresse et l'ordre. L'adresse et l'ordre sont répétés 2 fois. La deuxième fois, les bits sont inversés et peuvent servir à confirmer la bonne réception de l'adresse et de l'ordre. Le temps total de l'impulsion reste fixé. Il est possible de ne pas faire de répétition et/ou d'inversion et donc de passer à 16bits.

Une commande est envoyée à la fois et répétée à intervalle régulier :

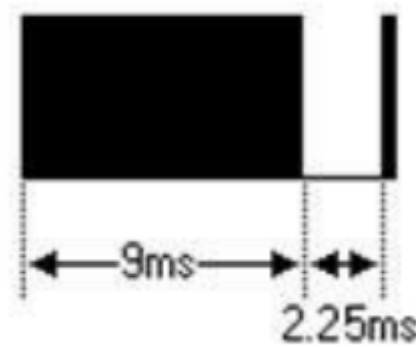


La commande est envoyée dès que l'appui sur le bouton.

Si le bouton de la télécommande est maintenu appuyé, l'impulsion des premières 110ms est différente de ci-dessus, le code est dupliqué et transmis après chaque

110ms. Le code dupliqué est transmit à une impulsion de niveau haut de 9ms et un niveau bas de 2.25ms puis un niveau haut de 560µs.

Impulsion répétée:



Note: après l'implusion, la vague d'onde entre dans le récepteur infrarouge pour y être décodée. Lorsqu'il n'y a pas d'onde reçue, le récepteur renvoi un signal haut, et un signal bas lorsqu'une réception est en cours. Le signal reçu est donc opposé à celui émis.

Il est possible d'observer ces signaux sur des oscilloscopes ou des programmes informatiques dédiés.

3. Programmer un robot contrôlé par infrarouge

Comme exposé dans le chapitre précédent, le protocole NEC expose comment coder un ordre en 4 parties :

- pulsion de tête (9ms + 4,5ms)
- code d'adresse (8bits code d'adresse / 8bits code de récupération)
16 bits code d'adresse (8bits code d'adresse / 8bits code de récupération)
- 16 bits code d'ordre (8bits ordre / 8 bits récupération)

Le code est répété en impulsion 9ms + 2,25ms + 560µs

Le temps est exploité pour tester l'état haut (1 logique), l'état bas (0 logique), la pulsion de tête, la pulsion répétée et savoir si le signal est correct ou incorrect.

Pour notre expérimentation, nous devons juste permettre au robot d'aller en avant, en

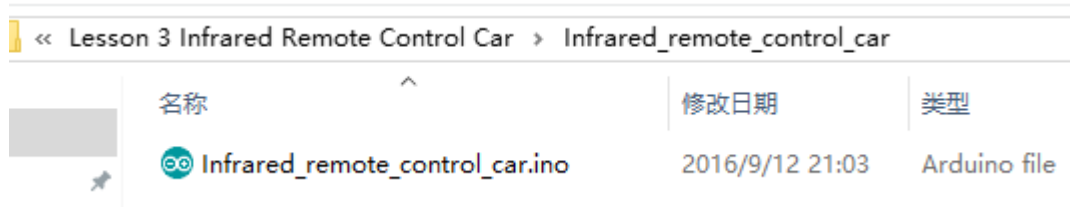
arrière, à droite, à gauche, s'arrêter: soit 5 commandes.

Remote control character	key value
Middle red button	16712445
Above triangle	16736925
Below triangle	16754775
Left triangle	16720605
Right triangle	16761405

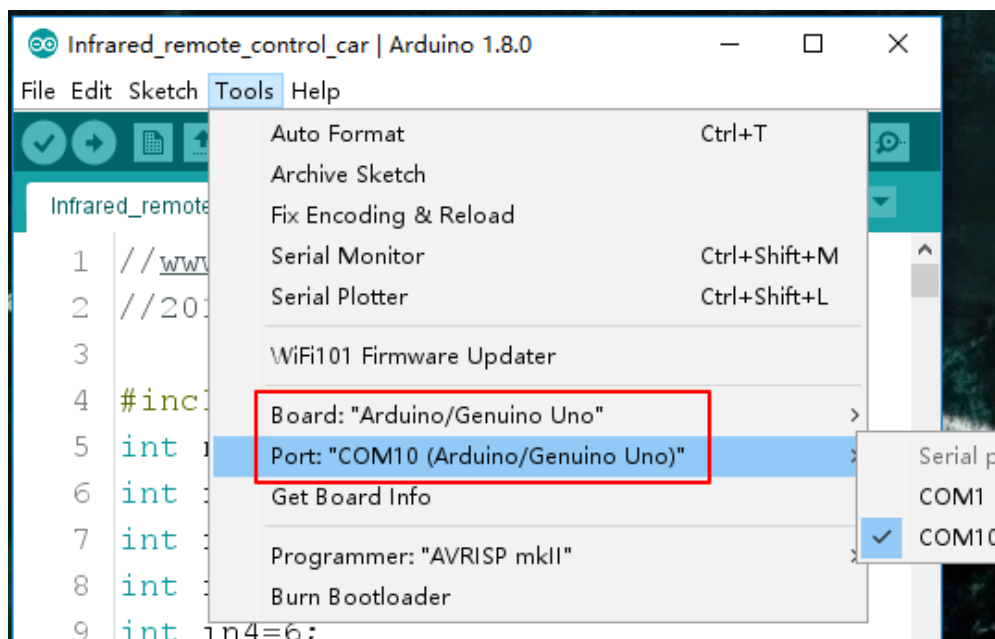


IV. Un robot commandé par infrarouge

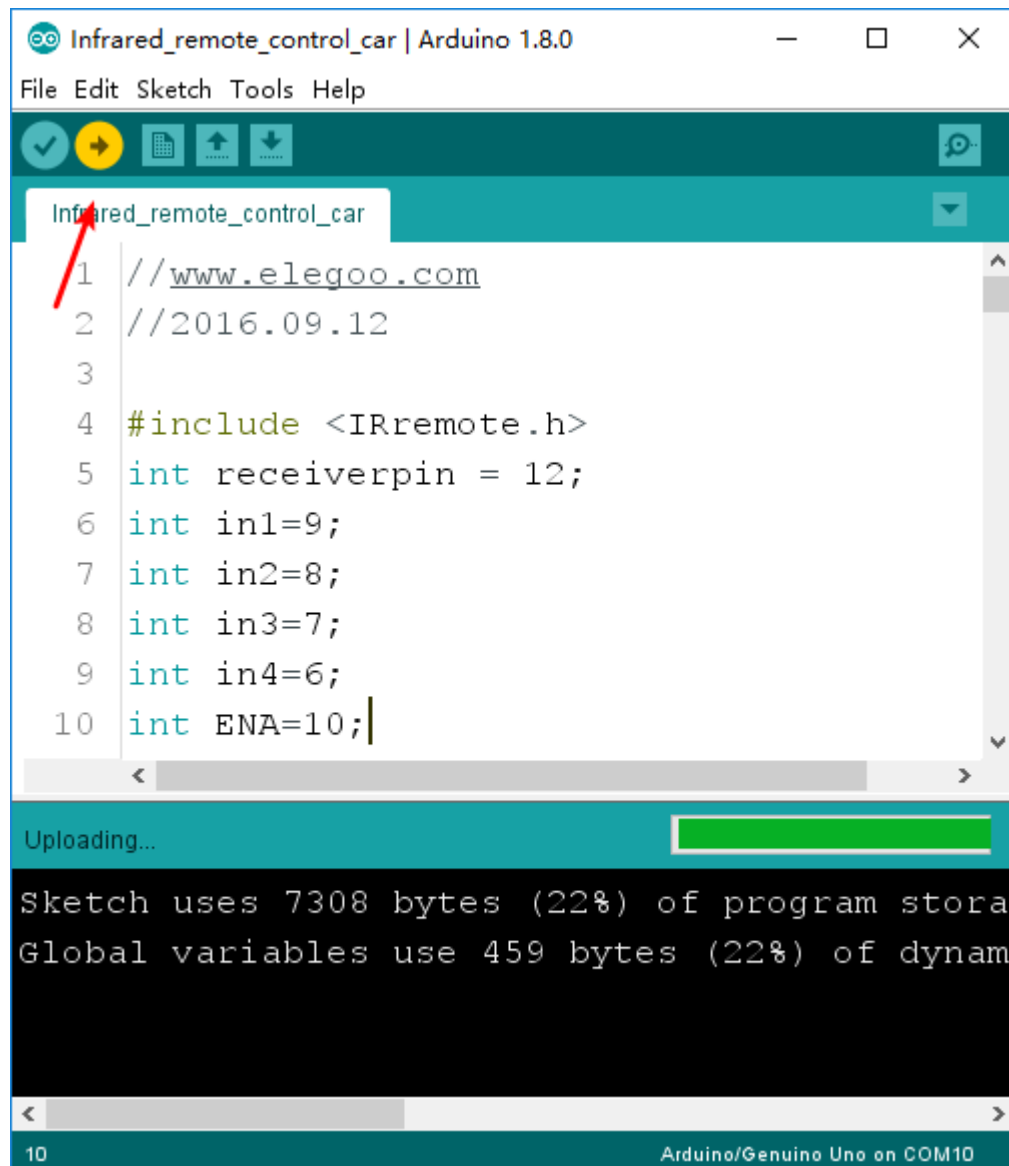
Ouvrez le fichier "Infrared_remote_control_car\ Infrared_remote_control_car.ino"



Sélectionnez la carte contrôleur Arduino UNO et le port série.



Cliquez sur le bouton de transfert (fleche)



The screenshot shows the Arduino IDE interface. The title bar reads 'Infrared_remote_control_car | Arduino 1.8.0'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for checking, uploading, creating a new sketch, opening a sketch, and saving a sketch. The sketch name 'Infrared_remote_control_car' is displayed in the top left. The code editor shows the following code:

```
1 //www.elegoo.com
2 //2016.09.12
3
4 #include <IRremote.h>
5 int receiverpin = 12;
6 int in1=9;
7 int in2=8;
8 int in3=7;
9 int in4=6;
10 int ENA=10;
```

Below the code editor, a status bar indicates 'Uploading...' with a green progress bar. The serial monitor shows the following output:

```
Sketch uses 7308 bytes (22%) of program storage
Global variables use 459 bytes (22%) of dynam
```

The bottom status bar shows '10' and 'Arduino/Genuino Uno on COM10'.

Quand le transfert est terminé, Débranchez le câble USB, posez le robot sur le sol, mettez-le en marche.



Voila, maintenant vous commandez ses déplacements via la télécommande infrarouge.

.

Le code est le suivant :

```
#include <IRremote.h>
int receiverpin = 12;
int in1=9;
int in2=8;
int in3=7;
int in4=6;
int ENA=5;
int ENB=10;
```

```
int ABS=130;
unsigned long RED;
#define A 16736925

#define B 16754775

#define X 16712445

#define C 16720605

#define D 16761405

IRrecv irrecv(receiverpin);
decode_results results;

void _mForward()
{
    digitalWrite(ENA,HIGH);
    digitalWrite(ENB,HIGH);
    digitalWrite(in1,HIGH);//digital output
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
    Serial.println("go forward!");
}

void _mBack()
{
    digitalWrite(ENA,HIGH);
    digitalWrite(ENB,HIGH);
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
}
```

```
    Serial.println("go back!");  
}  
  
void _mleft()  
{  
    analogWrite(ENA,ABS);  
    analogWrite(ENB,ABS);  
    digitalWrite(in1,HIGH);  
    digitalWrite(in2,LOW);  
    digitalWrite(in3,HIGH);  
    digitalWrite(in4,LOW);  
    Serial.println("go left!");  
}  
  
void _mright()  
{  
    analogWrite(ENA,ABS);  
    analogWrite(ENB,ABS);  
    digitalWrite(in1,LOW);  
    digitalWrite(in2,HIGH);  
    digitalWrite(in3,LOW);  
    digitalWrite(in4,HIGH);  
    Serial.println("go right!");  
}  
  
void _mStop()  
{  
    digitalWrite(ENA,LOW);  
    digitalWrite(ENB,LOW);  
    Serial.println("STOP!");  
}  
  
void setup() {  
    // put your setup code here, to run once:  
    pinMode(in1,OUTPUT);  
    pinMode(in2,OUTPUT);  
    pinMode(in3,OUTPUT);
```

```
pinMode(in4,OUTPUT);
pinMode(ENA,OUTPUT);
pinMode(ENB,OUTPUT);
pinMode(receiverpin,INPUT);
Serial.begin(9600);
_mStop();
irrecv.enableIRIn();
}
```

```
void loop() {
  if (irrecv.decode(&results))
  {
```

```
    RED=results.value;
    Serial.println(RED);
    irrecv.resume();
    delay(150);
    if(RED==A)
    {
      _mForward();
    }
```

```
    else if(RED==B)
    {
      _mBack();
    }
```

```
    else if(RED==C)
    {
      _mleft();
    }
```

```
    else if(RED==D)
```

```
{  
  _mright();  
}  
  
else if(RED==X)  
{  
  _mStop();  
}  
  
}  
}
```