

Lezione 3 La macchina controllata ad infrarossi



Gli argomenti di questa sezione

Il controllo remoto ad infrarossi e' un metodo molto usato per il controllo remoto. La macchina e' equipaggiata con un ricevitore ad infrarossi e quindi ci permette di controllarla attraverso un telecomando ad infrarossi.

Parte di apprendimento:

- Comprendere il controllore remoto ad infrarossi e il ricevitore

- Comprendere I principi del controllo remoto

Componenti necessari:

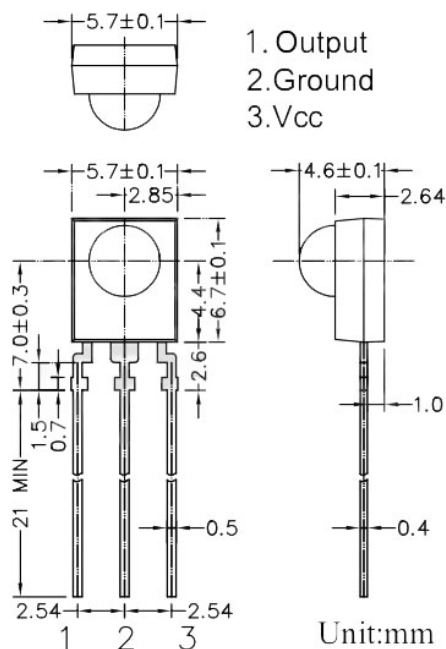
- Una macchina (con batteria)

- Un cavo USB

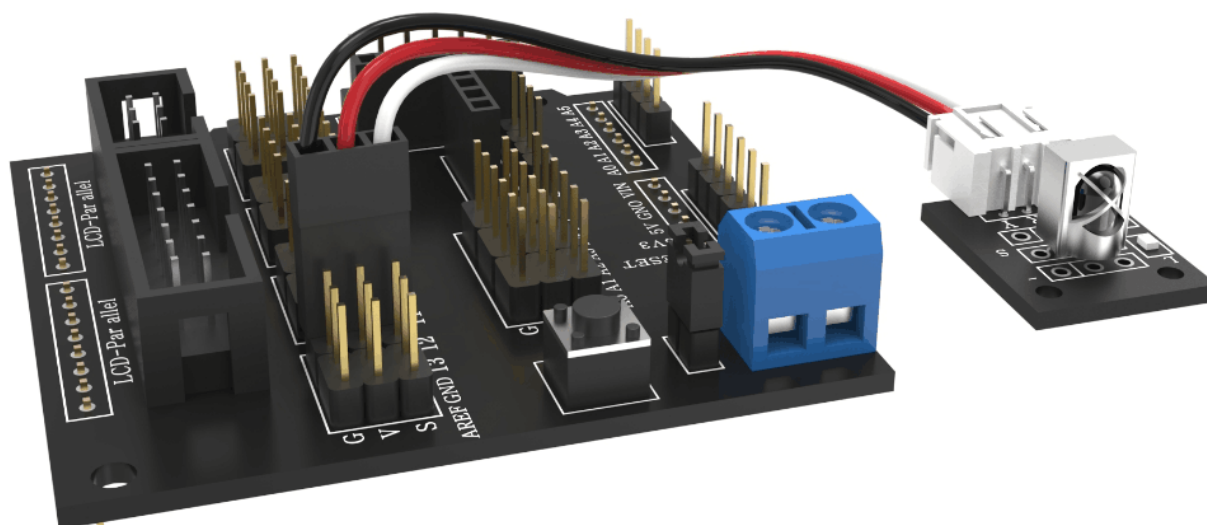
- un modulo ricevitore IR e un telecomando IR

I . Modulo ricevitore IR e telecomando IR

I dati del sensore ricevitore IR sono questi:



E questo e' il modo di connetterlo:



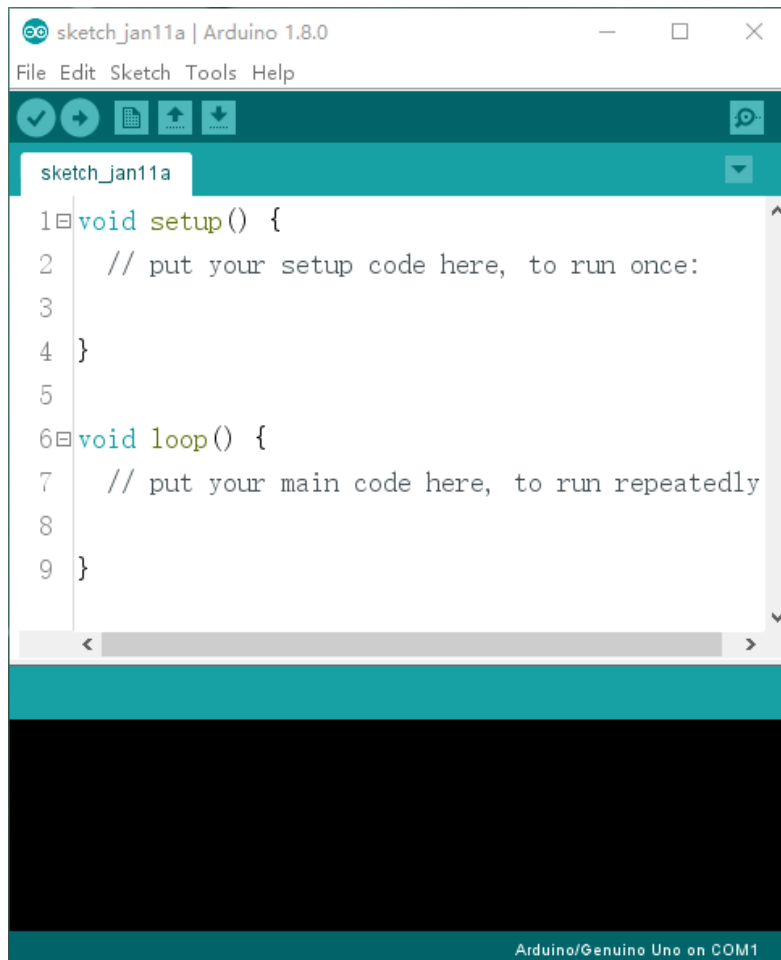
Questo e' il telecomando IR:



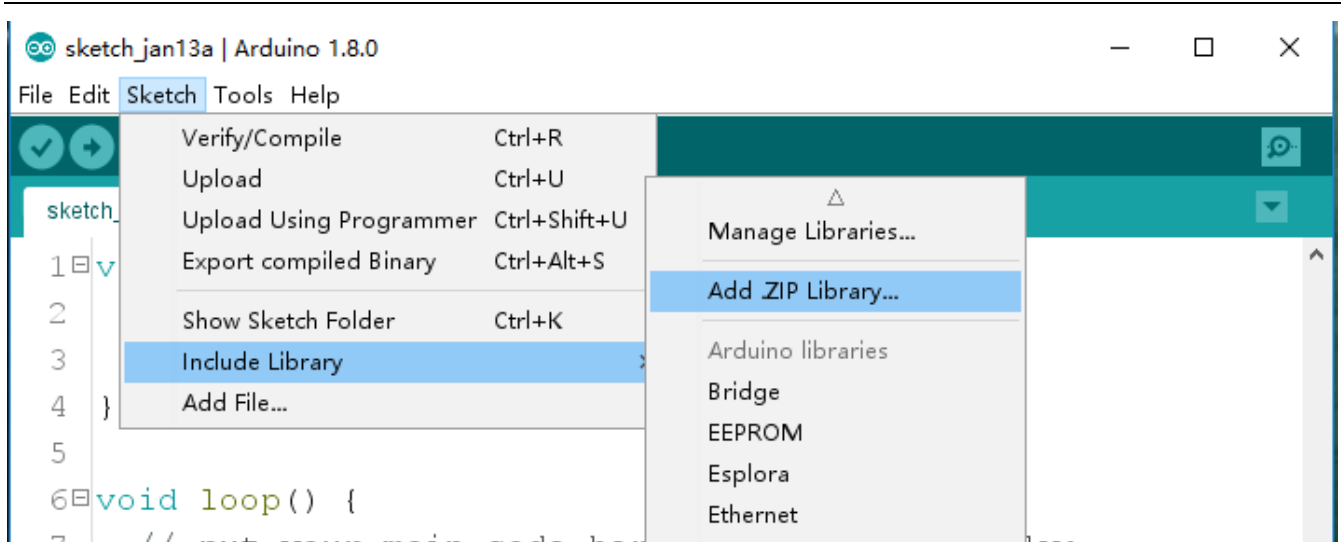
II . Testiamo il programma

Dato che in questo programma useremo delle librerie, sara' necessario aggiungere il file della libreria.

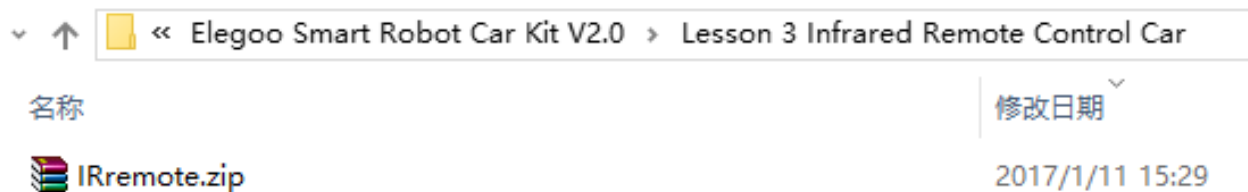
Apriamo l'IDE di Arduino



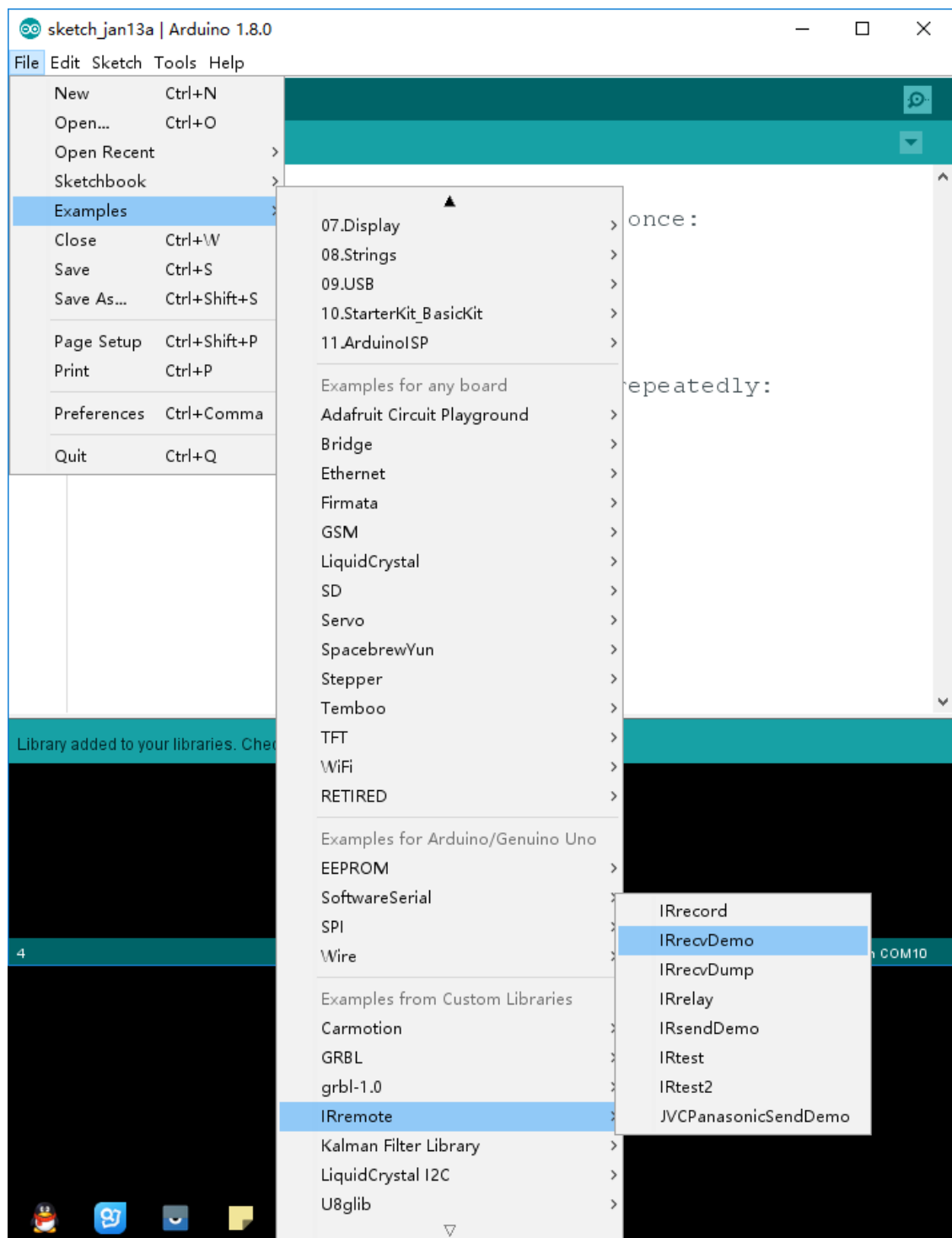
Clicchiamo Sketch—Include Library—Add .ZIP Library...—selezioniamo poi la libreria come mostrato qui sotto.

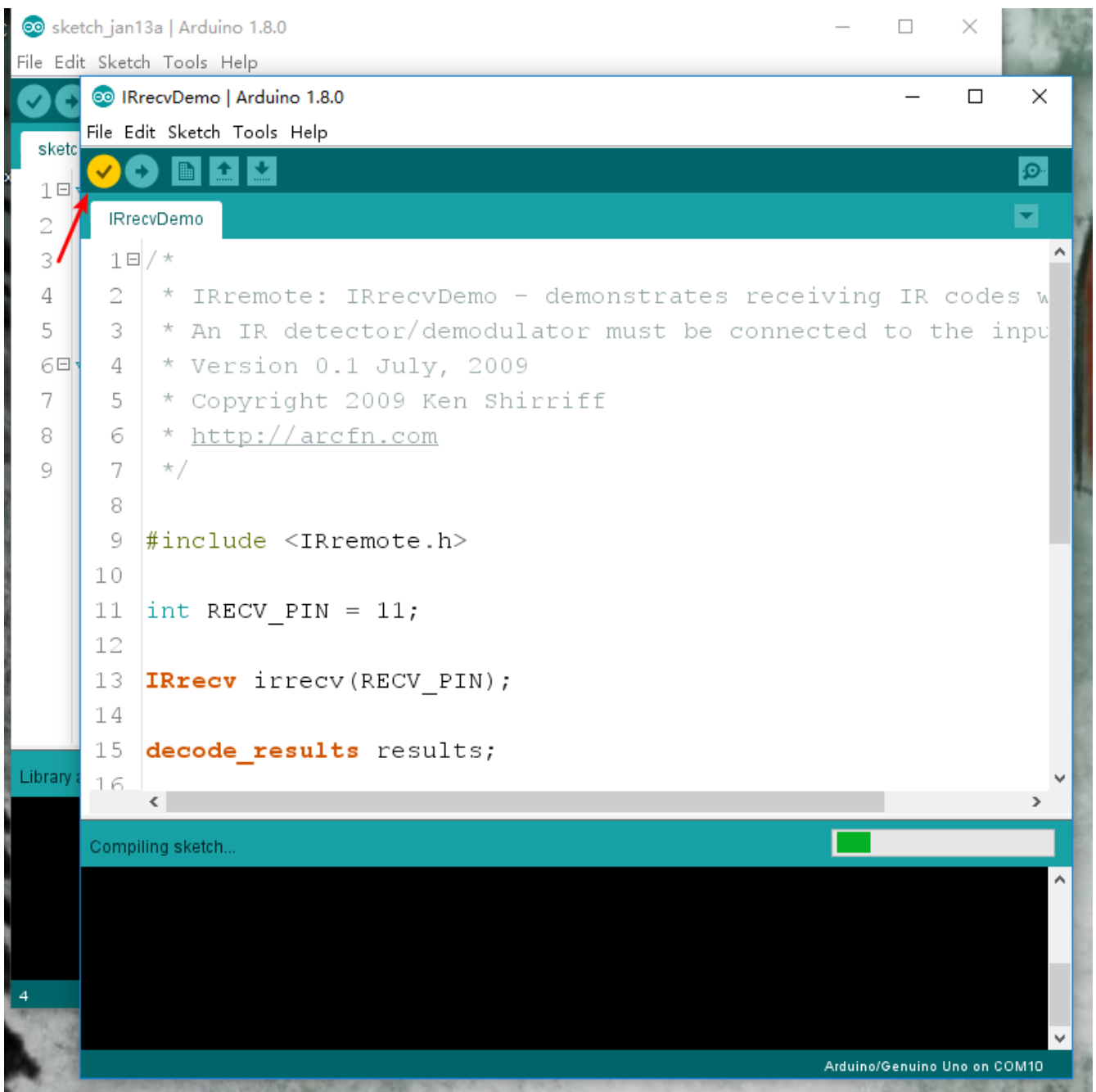


Il nome della libreria ZIP deve essere IRremote.zip.

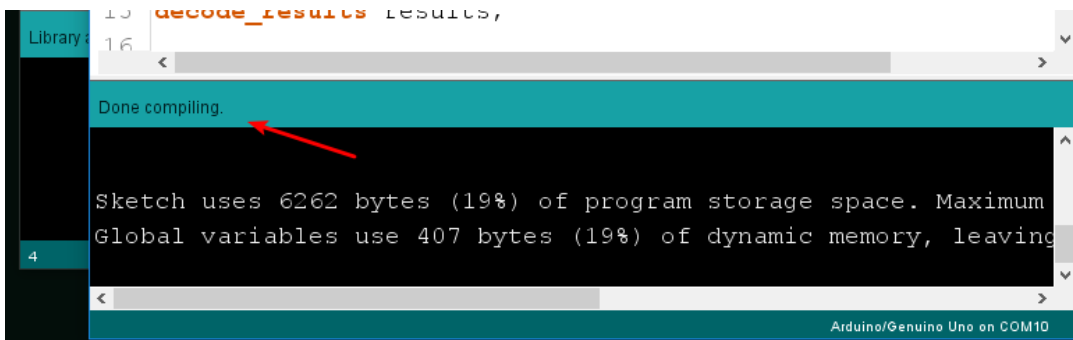


La compilazione va fatta con questo file di libreria, che e' stato modificato allo scopo. Selezioniamo un esempio IRremote






Effettuate la compilazione, se da errore, sara' necessario aggiungere il file della libreria di nuovo, perche' non si e' caricato correttamente.



Aprire il file infrared_Blink\infrared_Blink.ino

名称	修改日期	类型
 infrared_Blink.ino	2017/1/5 11:57	Ardui

Ecco il codice:

```
#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();
    }
  }
}
```



Fate l'Upload del programma sulla scheda di controllo ROMEO. Dopo averlo disconnesso dal computer, poggiate la macchina a terra e accendetela.

Premete il bottone "1" puntando il telecomando verso la macchina, osservate la macchina, vedrete spegnersi il led della macchina stessa.



III. Introduzione ai principi

1. Principio di funzionamento

Il sistema universale di controllo remoto consiste di due parti: inviare e ricevere, la parte di invio consiste in un telecomando IR, la parte ricevente consiste di un tubo ricevente ad infrarossi. I segnali inviati dal telecomando IR sono una serie di impulsi di codice binario. Per evitare l'interferenza di altri segnali infrarossi durante l'invio senza fili, vengono generalmente modulati su una frequenza portante, e quindi lanciati attraverso un fototransistor emettitore. Il tubo ricevente ad infrarossi, filtra a sua volta altre onde di disturbo, ricevendo unicamente i segnali con una data frequenza ritrasformandoli in impulsi di codice binario, operazione detta demodulazione. Il tubo ricevente trasforma i segnali di luce inviati dal diodo emettitore a infrarossi in deboli segnali elettrici, questi segnali vengono amplificati tramite un amplificatore interno IC, e attraverso un controllo automatico del guadagno, filtraggio passa-banda, demodulazione, riformazione dell'onda, vengono riportati alla codifica originale inviata dal telecomando, che viene riconosciuta dal circuito tramite la codifica che si trasforma nell'input dell'apparato elettrico passando per i pin di output del modulo ricevitore ad infrarossi.

2. Protocollo del controllo remoto ad infrarossi

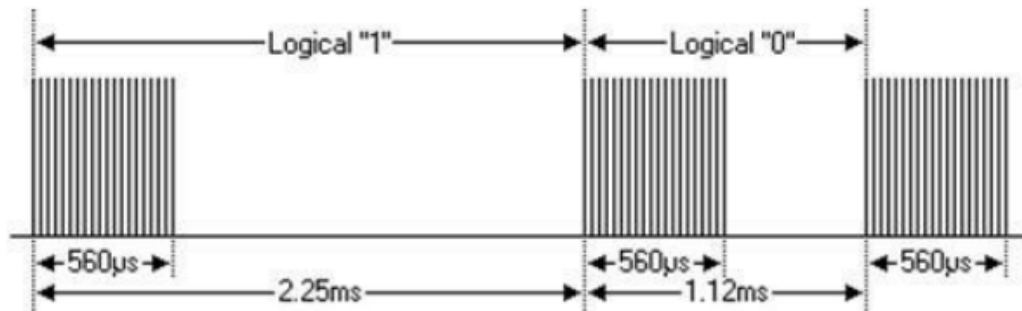
Lo schema di codifica degli apparati accoppiati tramite IR e': NEC protocol.

Qui di seguito apprenderemo cosa e' il protocollo NEC .

Caratteristiche:

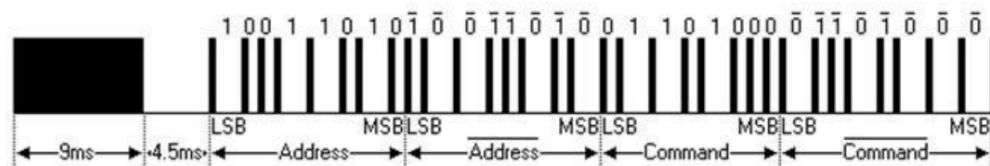
- (1) 8 bit di indirizzo, 8 bit di comando
- (2) i bit di indirizzamento e di comando vengono trasmessi due volte per garantire l'affidabilita'
- (3) Modulazione della posizione d'impulso
- (4) La frequenza portante e' 38kHz
- (5) Il tempo di ogni bit e' 1.125ms o 2.25ms

La definizione dello 0 e 1 logici e questa:



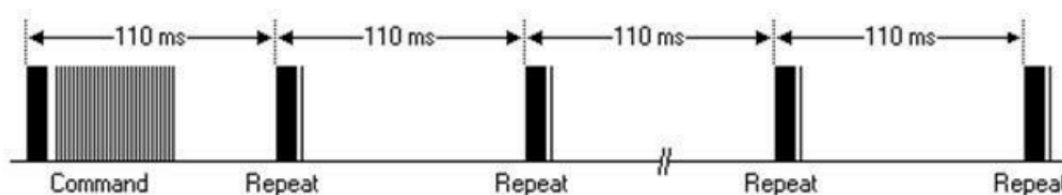
Il protocollo e' come mostrato sotto:

Stampa dell'istante dell'impulso di trasmissione:



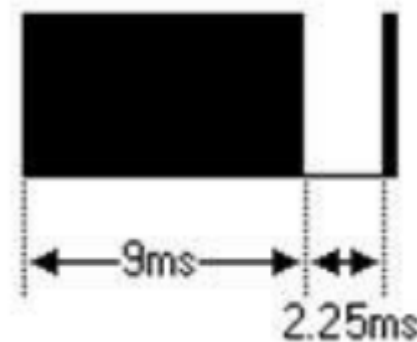
Nota: Questo protocollo invia prima l'LSB (bit meno significativo) . L'indirizzo di trasporto dell'impulso qui sopra e' 0x59, il comando e' 0x16. Un messaggio parte da un livello alto di 9ms, il seguente ha un livello basso di 4.5ms, (codice con schema di guida a due livelli) e attraverso il codice di indirizzo e quello di comando. L'indirizzo e il comando vengono trasmessi due volte. In seconda battuta tutti i bit convertiti negli opposti, possono essere usati per validare i messaggi ricevuti e dichiararli utilizzabili. Il tempo totale di invio e' fisso, Se non e' importante per voi potete trascurare il sistema di affidabilita' tramite inversione ed espandere a 16 bit gli indirizzi ed i comandi! In base al fatto che la lunghezza si raddoppia se ogni bit viene inviato anche invertito.

Stampa dell'istante dell'impulso di trasmissione:



Il comando viene inviato una volta, anche se il pulsante del telecomando rimane premuto. Quando il pulsante rimane premuto, l'impulso dei primi 110ms differisce da quello mostrato sopra, il codice duplicato viene trasmesso dopo ogni 110ms. Il codice duplicato è composto da un impulso di livello alto di 9ms e di uno di livello basso di 2.25 e di uno di livello alto di 560μs.

Impulso ripetuto:



Nota: Dopo che la forma d'onda dell'impulso viene integrata dal sensore, in base al fatto che l'integrazione del sensore va decodificata, il segnale amplificato e formato, va notato che nel tempo in cui non ci sono segnali infrarossi, il suo terminale di uscita ha un livello alto, ha un livello basso quando invece sono presenti segnali. Quindi il livello del segnale di uscita e' l'opposto di quello del terminale trasmittente. Chiunque puo' osservare l'impulso del ricevitore tramite un oscilloscopio, e comprendere il programma osservando la forma d'onda.

3. Programmare una macchina con controllo remoto

In accordo con le caratteristiche del codice NEC e l'onda della parte ricevente, questo esperimento divide le onde della parte ricevente in quattro parti: codice di intestazione (impulso di 9ms e 4.5ms), codice di indirizzo (includendo il codice di indirizzo di 8-bit e gli 8-bit dell'indirizzo ricevuto), codice di indirizzo a 16-bit (includendo il codice di indirizzo di 8-bit e gli 8-bit dell'indirizzo ricevuto), 16-bit del codice di comando (includendo il codice di indirizzo di 8-bit e gli 8-bit dell'indirizzo ricevuto), il codice ripetuto (puo' essere composto di un impulso di 9ms, 2.25ms, 560us).

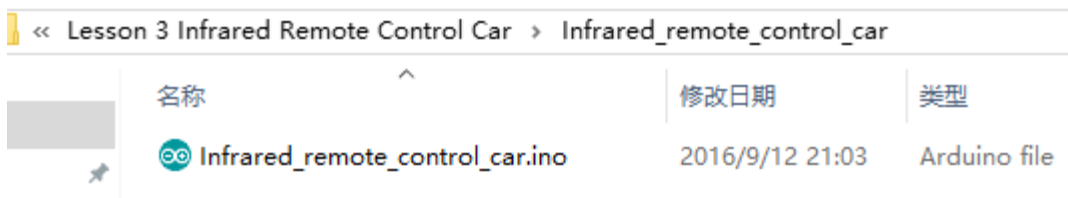
Utilizzando il timer per testare le onde ricevute di alto livello e di basso livello, in accordo con il tempo testato logical"01", logical"1", leading pulse, repeat pulse. Leading code e address code vengono analizzati sulla correttezza ma non memorizzati, per il fatto che il codice di comando di ogni tasto e' differente, l'azione e' veicolata dal codice di comando. In questo esperimento vogliamo unicamente che la macchina possa andare avanti, indietro, girare a destra, a sinistra e fermarsi, cio' significa che abbiamo necessita' di cinque comandi in tutto; I pulsanti utilizzati e il loro valore sono nella tabella qui sotto:

Simbolo sul telecomando	Valore del tasto
Bottone rosso mediano	16712445
Triangolo superiore	16736925
Triangolo inferiore	16754775
Triangolo sinistro	16720605
Triangolo destro	16761405

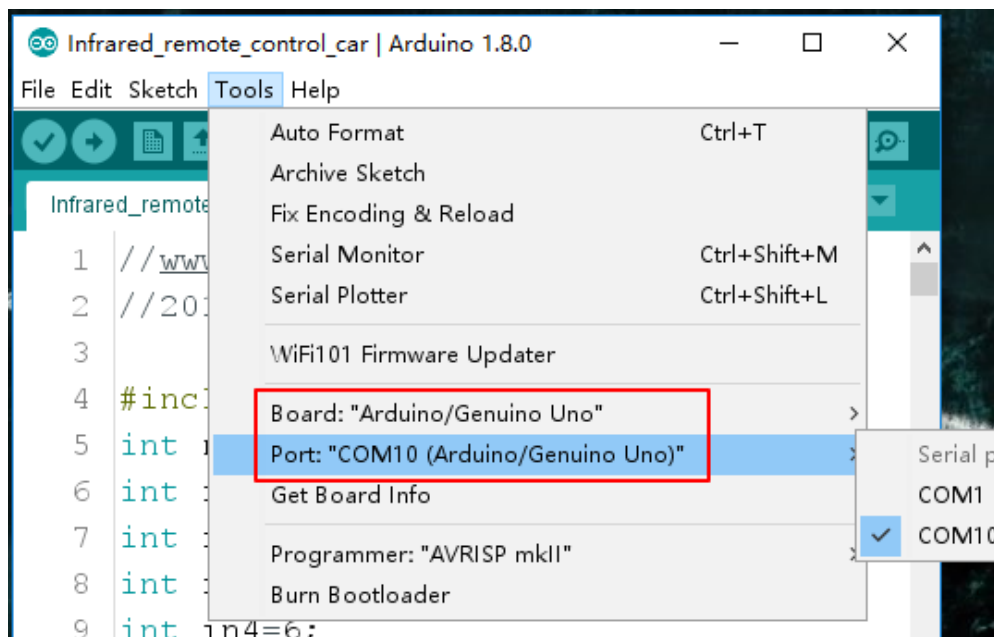


IV. Creiamo una macchina controllata a distanza

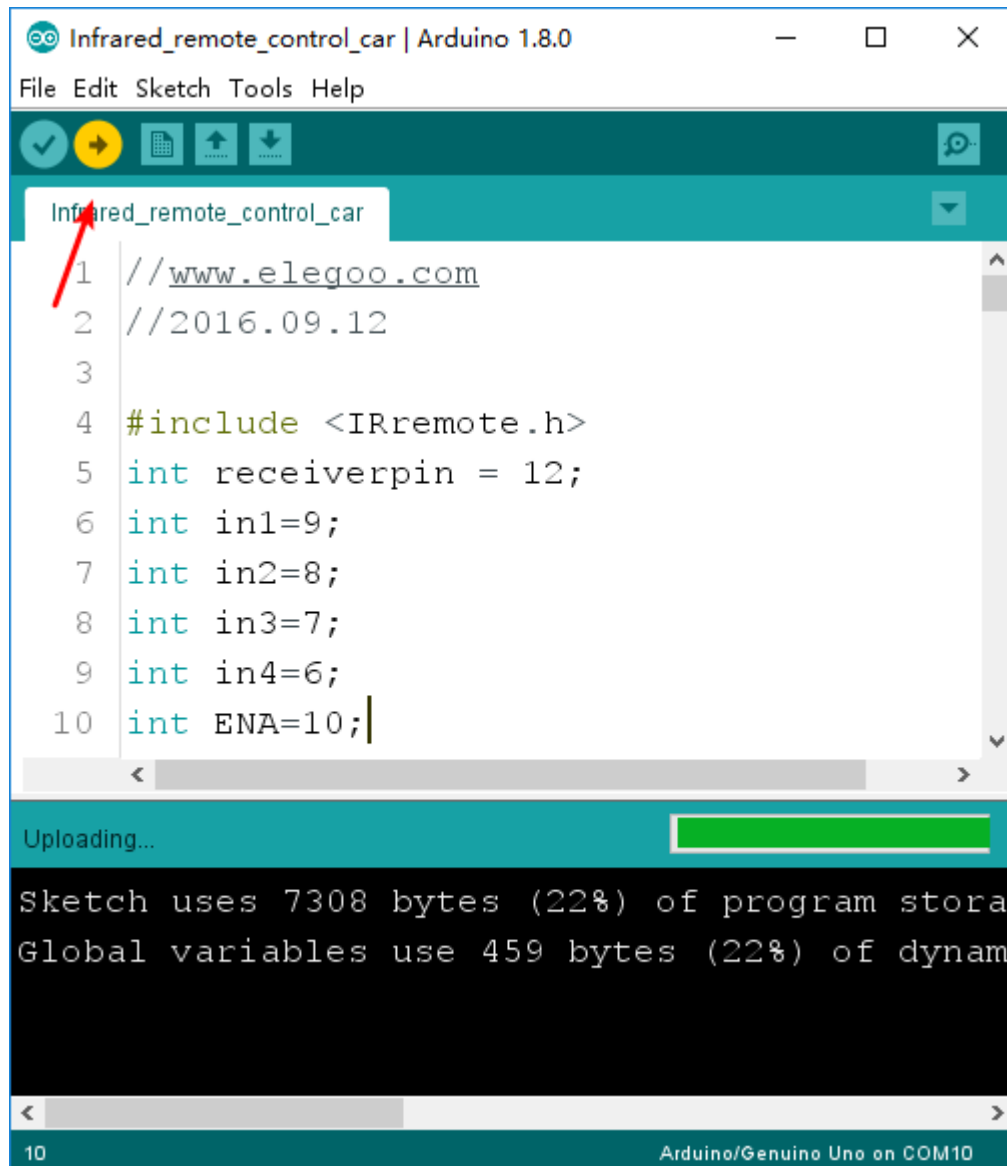
Effettuiamo l'upload del programma sulla macchina , come indicato qui sotto, aprite il file Infrared_remote_control_car\ Infrared_remote_control_car.ino



Selezionate la scheda Arduino Uno e la giusta porta seriale.



Premete il pulsante per effettuare l'upload



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, and saving. The sketch name 'Infrared_remote_control_car' is displayed in the top bar. 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, the 'Uploading...' status bar shows a green progress bar. The output window displays the following message:

```

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

The status bar at the bottom indicates '10' and 'Arduino/Genuino Uno on COM10'.

Dopo aver fatto l'upload, disconnettete la macchina dal computer. Accendete la macchina e poggiatela a terra. Premete I pulsanti sul telecomando e vedrete la macchina eseguire I comandi.



Ecco fatto, potete felicemente giocare con la macchina controllata via IR.

Ecco il codice:

```
#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();
    }

}
```