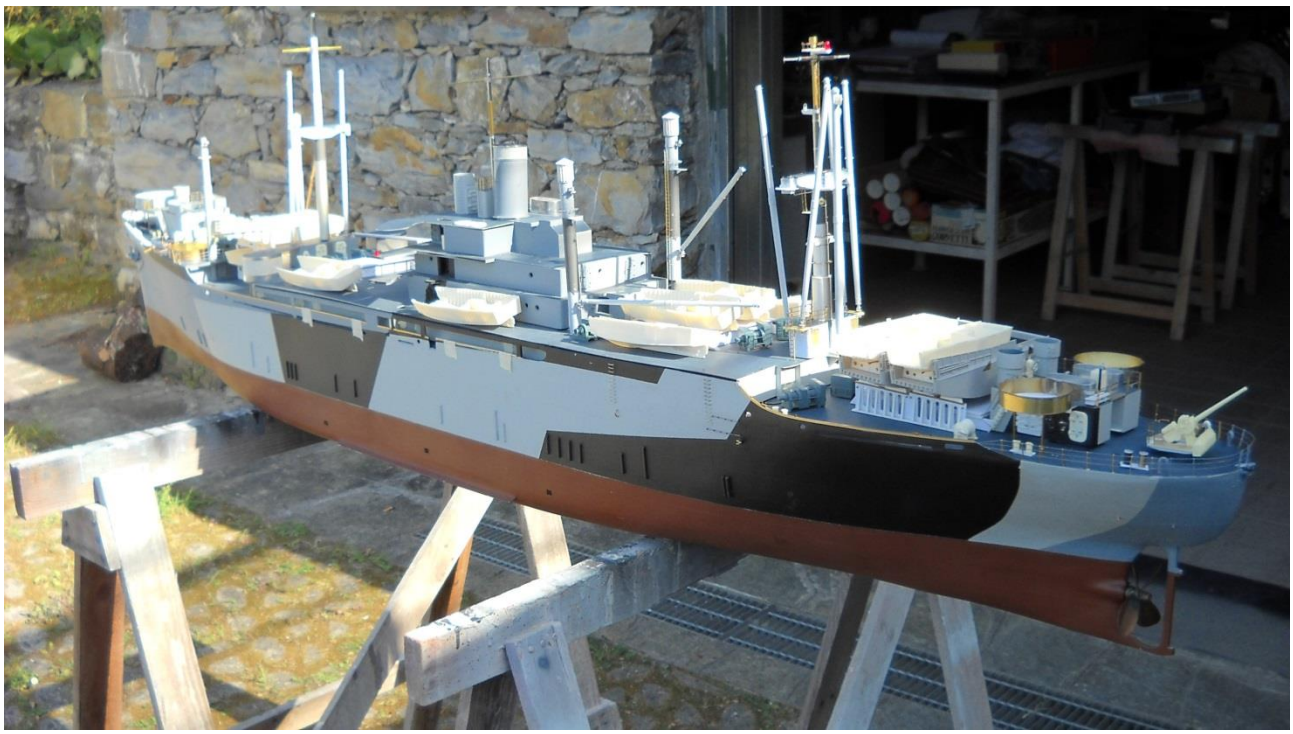
Arduino per l'automazione dei modelli navali

di

Federico Davanteri
e
Albino Benedetto

2015



Introduzione

Quando si pensa ad Arduino e al suo possibile utilizzo in ambito modellistico navale, viene subito in mente l'impiego più classico che si possa fare di questa versatilissima piattaforma: l'automazione di tutti gli accessori che solitamente compaiono sui modelli più sofisticati.
Mi riferisco a luci, radar, argani, gru, sirene ecc.
In poche parole tutto ciò che rende viva una riproduzione navale.
Arduino è lo strumento ideale per questo tipo di applicazione: esso permette infatti di gestire innumerevoli accessori, di varia natura, per creare effetti realistici e all'occorrenza anche complessi.
Il software descritto in questo articolo si presta ad essere facilmente adattato a diverse applicazioni dove sia necessario, in generale, azionare più dispositivi in contemporanea assegnando loro comportamenti diversi.
Per esempio l'automazione di un plastico ferroviario, di un presepio, o la gestione delle luci dell'albero di Natale.

Definizioni

Nell'articolo faremo riferimento a due definizioni ricorrenti, Evento e Sequenza, che per maggior chiarezza sono descritte di seguito:

Evento: Per evento si intende un determinato comportamento di un componente del nostro impianto. Per esempio un evento è l'accensione di una luce oppure la rotazione di un radar o l'emissione di una segnalazione morse.

Sequenza: Una sequenza è un insieme concatenato di eventi che possono avere durate diverse.

Componenti utilizzati

La classica scheda Arduino UNO si presta perfettamente per questo tipo di applicazione.
Può essere direttamente collegata ai carichi esterni (tenendo sempre presente l'assorbimento di corrente, che non deve superare i 20 mA per ogni pin), oppure associata ad una scheda a relè, che permette di pilotare carichi esterni con assorbimenti di corrente più elevati.

Nel nostro caso abbiamo utilizzato una scheda a otto relay molto diffusa.

Questa scheda è caratterizzata dal fatto che, per ragioni di sicurezza, i relay vengono eccitati quando il segnale che arriva al corrispondente pin è LOW, cioè a zero. Viceversa saranno a riposo in presenza di un segnale HIGH da Arduino. La scheda è altresì molto sicura, in quanto l'azionamento dei relay avviene tramite fototransistor. Non c'è nessuna connessione fisica tra i pin collegati ad Arduino e la linea di potenza gestita dai relay.

Il software

Il software che abbiamo scritto gestisce otto diversi canali, ciascuno dei quali è in grado di eseguire ciclicamente una sequenza prestabilita di eventi.
In linea generale, il software gestisce otto timer distinti e tramite questi, a turno, controlla lo stato di ogni sequenza e lo modifica secondo la programmazione fatta.
Ogni sequenza è costituita da una serie di coppie di valori "durata, stato" nelle quali la durata è espressa in secondi e stabilisce appunto la durata dell'evento, mentre lo stato (0/1) stabilisce se l'evento deve manifestarsi (stato=1 es: sirena che suona) oppure no (stato=0).

L'obiettivo che ci siamo posti era quello di scrivere un software che potesse essere utilizzato anche da chi non ha dimestichezza con la programmazione.
Questo particolare software non prevede valori in ingresso, ma solo input in uscita, del tipo on-off.

Per questo abbiamo cercato di rendere il più semplice possibile la personalizzazione delle sequenze di eventi, unica modifica da apportare al codice per poterlo utilizzare adattandolo alle proprie esigenze.

Come personalizzare il software

Le righe seguenti, in giallo, contengono le istruzioni da personalizzare:

```
// ################################################################################

// Impostare la variabile 'modo' in base all'utilizzo o meno della scheda relay
// int modo = 1;  --> Utilizzo scheda 8 releay
// int modo = 0;  --> Utilizzo diretto da Arduino (senza scheda relay)

  int modo = 1;

// definizione sequenze eventi: durata, stato, durata, stato, durata, stato ...
// le sequenze possono avere durate totali diverse
// i valori inseriti nella sequenza devono essere sempre in coppie durata/stato
// lo stato può essere solo 0 oppure 1

  const int ch_1[] = {5,1,5,0,30,1,2,0};  // Esempio luci plancia
  const int ch_2[] = {0,0};  // inserire denominazione del canale
  const int ch_3[] = {0,0};  // inserire denominazione del canale
  const int ch_4[] = {0,0};  // inserire denominazione del canale
  const int ch_5[] = {0,0};  // inserire denominazione del canale
  const int ch_6[] = {0,0};  // inserire denominazione del canale
  const int ch_7[] = {0,0};  // inserire denominazione del canale
  const int ch_8[] = {99,0,30,0};  // Segnalatore MORSE

// inserire nella riga seguente la frase da codificare in MORSE

  char frase[] = {"INSERIRE MESSAGGIO MORSE QUI"};

// ################################################################################
```

La prima riga “int modo = 1” serve per definire se si sta utilizzando la scheda relay o meno. Per utilizzare la scheda relay impostare modo = 1, altrimenti modo = 0 per il collegamento diretto dei carichi su Arduino.

Il programma, così come viene presentato in questo articolo, funziona correttamente se tutti i carichi esterni sono collegati ad Arduino (modo = 0) oppure alla scheda Realy (modo = 1). Non sono ammesse soluzioni ibride, con alcuni carichi su Arduino e altri sui relay. Se si desidera attuare un collegamento di questo tipo occorre modificare il programma invertendo gli stati ON/OFF nei canali interessati.

Vediamo quindi come funziona la programmazione delle sequenze.
Nell’inserimento delle sequenze è necessario porre la massima attenzione al rispetto della sintassi delle righe che si modificano. Errori in questa fase portano a comportamenti del tutto imprevedibili del programma.
Inserire sempre coppie di valori nelle quali la seconda cifra deve essere sempre 0 o 1.

Consideriamo la prima riga, che gestisce le luci della plancia

Es: const int ch_1[] = {5,1,5,0,30,1,2,0};   // Luci plancia

Con i valori inseriti nella riga, l'effetto sulle luci sarà il seguente:

- Luci accese per 5 sec. (5,1)
- Luci spente per 5 sec. (5,0)
- Luci accese per 30 sec. (30,1)
- Luci spente per 2 sec. (2,0)
- Da capo

La riga di definizione del MORSE ha invece una codifica diversa.
L'evento MORSE viene identificato dal codice 99 mentre lo stato deve essere sempre zero.

Es: const int ch_8[] = {99,0,30,0}; // Segnalatore MORSE

Con i valori inseriti nella riga si avrà quanto segue:

- Segnalazione morse (99,0) (La durata dipende dalla lunghezza della frase inserita)
- Pausa (30,0) (vedi nota sotto)
- Da capo

Questa versione del software prevede specificatamente l'utilizzo del canale "8" per il codice Morse.
La possibilità di fare una segnalazione in Morse può non essere interessante.
In questo caso per ometterla, è sufficiente apporre "0" (zero) al posto del "99"

NOTA: Per quanto riguarda il morse va fatta una considerazione importante, da tenere presente nella stesura delle sequenze.

La segnalazione morse introduce inevitabilmente un ritardo nell'esecuzione del programma, nel senso che durante il segnale morse non possono avvenire cambi di stato in eventi di altre sequenze. Inoltre, la durata della segnalazione occupa parte o tutta la pausa che viene posta di seguito.

Per chiarire, se il nostro segnale morse dura 20 secondi, questi venti secondi saranno parte dei trenta secondi di pausa posti a seguire. Quindi, all'atto pratico, la codifica precedente avrà questo risultato:

- Segnalazione morse (99,0) (durata di 20 secondi)
- Seguirebbe una pausa di 30 secondi (30,0) ma i primi 20 sono trascorsi durante la segnalazione morse, per cui la pausa effettiva dopo il morse sarà di soli 10 secondi (30 secondi meno la durata del morse. 30 – 20 = 10)
- Da capo

Naturalmente, se la pausa impostata è più breve del segnale morse, questa verrà allungata fino al termine del morse.

Per calcolare la durata del messaggio Morse (che ovviamente dipende dal messaggio stesso) si deve sapere che:

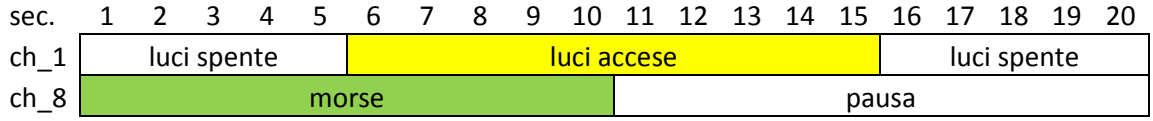
- l'unità “base” del linguaggio Morse è la durata del “punto” (.)
- nel nostro programma:
- il “punto “ dura 1 decimo di secondo
- la “linea” dura come 3 punti (3 decimi di secondo)
- l'intervallo tra due segni (linea o punto) della stessa lettera dura come un punto (1 decimo di secondo).
- Per es.: la lettera “A” si scrive **. -** (punto e linea) e dura 1+1+3 decimi di secondo
- l'intervallo tra due lettere della stessa parola dura come due punti (2 decimi di secondo)
- l'intervallo tra due parole dura come quattro punti (4 decimi di secondo).
- Il programma prevede il segnale di inizio messaggio (- . - . -) e quello di fine messaggio (. - . - .) che durano rispettivamente: il primo 15 decimi di secondo ed il secondo 11 decimi di secondo.
- Il programma prevede solo le lettere dell'alfabeto internazionale (NO numeri, NO segni di interpunzione, NO caratteri speciali).

Il codice Morse è il seguente:

A: ·—	B: —···	C: —·—·
D: —··	E: ·	F: ··—·
G: ——·	H: ····	I: ··
J: ·———	K: —·—	L: ·—··
M: ——	N: —·	O: ———
P: ·——·	Q: ——·—	R: ·—·
S: ···	T: —	U: ··—
V: ···—	W: ·——	X: —··—
Y: —·——	Z: ——··	

Alla luce di quanto descritto sopra, vediamo come funzioneranno due differenti programmazioni, una sbagliata e una corretta:

SEQUENZA SBAGLIATA SU CH_1

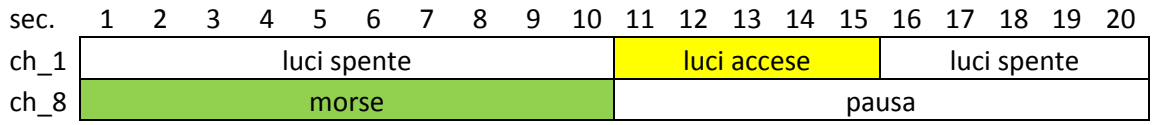


La codifica da inserire nel programma sarà la seguente:

```
const int ch_1[] = {5,0,10,1,5,0};  // Luci
const int ch_2[] = {0,0};  // xxxx
const int ch_3[] = {0,0};  // xxxx
const int ch_4[] = {0,0};  // xxxx
const int ch_5[] = {0,0};  // xxxx
const int ch_6[] = {0,0};  // xxxx
const int ch_7[] = {0,0};  // xxxx
const int ch_8[] = {99,0,20,0};  // Segnalatore MORSE
```

In questo caso le luci si accenderanno solo dopo dieci secondi dall'inizio (e non dopo 5), al termine del messaggio morse, in quanto durante i dieci secondi del morse nessun'altro evento può cambiare stato.

SEQUENZA CORRETTA SU CH_1



La codifica da inserire nel programma sarà la seguente:

```
const int ch_1[] = {10,0,5,1,5,0};  // Luci
const int ch_2[] = {0,0};  // xxxx
const int ch_3[] = {0,0};  // xxxx
const int ch_4[] = {0,0};  // xxxx
const int ch_5[] = {0,0};  // xxxx
const int ch_6[] = {0,0};  // xxxx
const int ch_7[] = {0,0};  // xxxx
const int ch_8[] = {99,0,20,0};  // Segnalatore MORSE
```

Questa sequenza è formalmente corretta in quanto durante la segnalazione morse l'altro canale (ch_1) non fa nulla. Le luci si possono accendere solo dopo il morse.

Per la definizione delle sequenze può essere utile organizzare un foglio Excel, come nell'esempio seguente, dove sono state definite otto colonne (una per ogni canale). Numerando le celle a partire da 1 ogni volta che si ha una variazione in un evento, sarà facile ricavare le durate dei singoli eventi da riportare nel codice di Arduino. La prima colonna "tempo" numerata in modo univoco, permette di tenere conto della durata totale delle sequenze.
Per esempio, la colonna 7 eseguirà un segnale della tromba di 6 secondi, poi pausa per 15 secondi e di nuovo un segnale di 9 secondi. Tutta la sequenza di eventi durerà 30 secondi.

tempo	1 radar	2 cannone	3 Luci #1	4 Luci #2	5 luci rosse	6 sirena	7 tromba	8 MORSE
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	**1**	5	5	5	5	5
6	**1**	6	2	6	6	6	6	6
7	2	7	3	7	7	7	1	7
8	3	8	4	8	8	8	2	8
9	4	9	5	9	9	9	3	9
10	5	10	6	10	10	10	4	10
11	6	11	7	11	11	11	5	11
12	7	12	8	12	12	12	6	12
13	8	13	9	13	13	13	7	13
14	9	14	10	14	14	14	8	14
15	10	15	11	15	15	15	9	15
16	11	16	12	**1**	16	16	10	16
17	12	17	13	2	17	17	11	17
18	13	18	14	3	18	18	12	18
19	14	19	15	4	19	19	13	19
20	15	20	16	5	20	20	14	20
21	16	21	17	6	21	21	15	21
22	17	22	18	7	22	22	**1**	morse
23	18	23	19	8	23	23	2	morse
24	19	24	20	9	24	24	3	morse
25	20	25	21	10	25	25	4	morse
26	21	26	22	11	26	26	5	morse
27	22	27	23	12	27	27	6	morse
28	23	28	24	13	28	28	7	morse
29	24	29	25	14	29	29	8	morse
30	25	30	26	15	30	30	9	morse

In conclusione, questo tipo di codifica va bene per tutti quei componenti che possono funzionare solo con due stati definiti. Luci, sirene o trombe, radar in rotazione ecc.
Accessori diversi, come per esempio l'utilizzo di motori per movimentare argani, gru ecc, che per loro natura rispondono ad una logica diversa, dovranno essere gestiti con routine di controllo espressamente scritte allo scopo e opportunamente inserite nel programma.

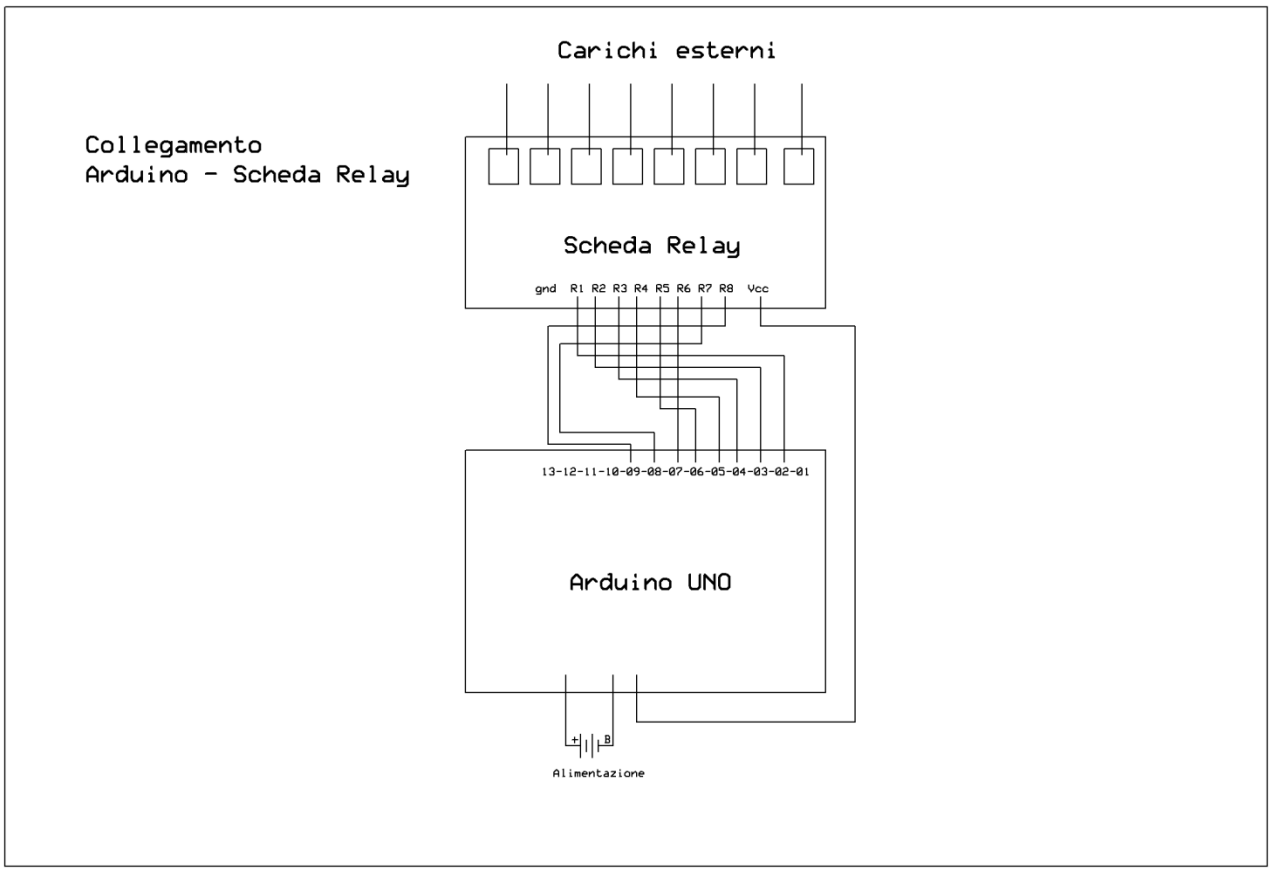
Il circuito

Innanzitutto vediamo come sono configurati i pin di Arduino. Quelli utilizzati vanno dal 2 al 9 e corrispondono ai relay della scheda da 1 a 8 (i pin 0 e 1 sono stati esclusi in quanto servono per la comunicazione seriale ed è buona norma non utilizzarli per la gestione di eventi)

Dato che il software è utilizzabile con o senza la scheda relay, la tabella seguente riporta le corrispondenze tra i canali (sequenze), i pin di Arduino e i pin della scheda relay

	pin Arduino	relay
canale 1	2	1
canale 2	3	2
canale 3	4	3
canale 4	5	4
canale 5	6	5
canale 6	7	6
canale 7	8	7
canale 8	9	8

Di seguito lo schema di collegamento



Il software

```
// definizione pin dei canali
// Inserire un commento per ogni canale

#define RELAY_1 2 // Luci plancia
#define RELAY_2 3 // xxxxx
#define RELAY_3 4 // xxxxx
#define RELAY_4 5 // xxxxx
#define RELAY_5 6 // xxxxx
#define RELAY_6 7 // xxxxx
#define RELAY_7 8 // xxxxx
#define RELAY_8 9 // segnalatore MORSE

// ################################################################################
// Impostare la variabile 'modo' in base all'utilizzo o meno della scheda relay
// int modo = 1;  --> Utilizzo scheda 8 releay
// int modo = 0;  --> Utilizzo diretto da Arduino (senza scheda relay)

  int modo = 1;

// definizione sequene eventi: durata, stato, durata, stato, durata, stato ...
// le sequenze possono avere durate totali diverse
// i valori inseriti nella sequenza devono essere sempre in coppie durata/stato

  const int ch_1[] = {0,0};  // Luci plancia
  const int ch_2[] = {0,0};  // xxxx
  const int ch_3[] = {0,0};  // xxxx
  const int ch_4[] = {0,0};  // xxxx
  const int ch_5[] = {0,0};  // xxxx
  const int ch_6[] = {0,0};  // xxxx
  const int ch_7[] = {0,0};  // xxxx
  const int ch_8[] = {99,0,10,0};  // Segnalatore MORSE

// inserire nella riga seguente la frase da codificare in MORSE

   char frase[] = {"INSERIRE MESSAGGIO MORSE QUI"};
 //char frase[] = {"SOS"};

// ################################################################################

int l_frase = sizeof(frase)-2;

const int dotlength = 100; // punto
const int dashlength = dotlength * 3;  // linea
const int inter = dotlength;    // interspazio punti
const int lgap = dotlength * 2; // interspazio lettere
const int wgap = dotlength * 4; // interspazio parole
```

```
// calcolo numero elementi array
int ch_1_count = sizeof(ch_1)/2;
int ch_2_count = sizeof(ch_2)/2;
int ch_3_count = sizeof(ch_3)/2;
int ch_4_count = sizeof(ch_4)/2;
int ch_5_count = sizeof(ch_5)/2;
int ch_6_count = sizeof(ch_6)/2;
int ch_7_count = sizeof(ch_7)/2;
int ch_8_count = sizeof(ch_8)/2;

// durata eventi (millisecondi)
unsigned long t_event_1;
unsigned long t_event_2;
unsigned long t_event_3;
unsigned long t_event_4;
unsigned long t_event_5;
unsigned long t_event_6;
unsigned long t_event_7;
unsigned long t_event_8;

// timer
unsigned long start_1;
unsigned long start_2;
unsigned long start_3;
unsigned long start_4;
unsigned long start_5;
unsigned long start_6;
unsigned long start_7;
unsigned long start_8;

unsigned long convtime = 1000;
int ON;
int OFF;

// stato eventi on/off
int t_stat_1;
int t_stat_2;
int t_stat_3;
int t_stat_4;
int t_stat_5;
int t_stat_6;
int t_stat_7;
int t_stat_8;

// Indici degli array
int trig_1 = 0;
int trig_2 = 0;
```

```
int trig_3 = 0;
int trig_4 = 0;
int trig_5 = 0;
int trig_6 = 0;
int trig_7 = 0;
int trig_8 = 0;

void dot()
{
  // play a dot
  //tone(buzzerPin, tonefreq);
  // LED
  digitalWrite(RELAY_8, ON);
  delay(dotlength);
  //noTone(buzzerPin);
  // LED
  digitalWrite(RELAY_8, OFF);
  delay(inter);
}

void dash()
{
  // play a dash
  //tone(buzzerPin, tonefreq);
  // LED
  digitalWrite(RELAY_8, ON);
  delay(dashlength);
  //noTone(buzzerPin);
  // LED
  digitalWrite(RELAY_8, OFF);
  delay(inter);
}

void soundLetter(char letter)
{
  // letters are in order of frequency
  switch(letter)
  {
  case 'E':
    dot();
    return;
  case 'T':
    dash();
    return;
  case 'A':
    dot();
    dash();
    return;
```

```
case 'O':
  dash();
  dash();
  dash();
  return;
case 'I':
  dot();
  dot();
  return;
case 'N':
  dash();
  dot();
  return;
case 'S':
  dot();
  dot();
  dot();
  return;
case 'H':
  dot();
  dot();
  dot();
  dot();
  return;
case 'R':
  dot();
  dash();
  dot();
  return;
case 'D':
  dash();
  dot();
  dot();
  return;
case 'L':
  dot();
  dash();
  dot();
  dot();
  return;
case 'C':
  dash();
  dot();
  dash();
  dot();
  return;
case 'U':
  dot();
```

```
  dot();
  dash();
  return;
case 'M':
  dash();
  dash();
  return;
case 'W':
  dot();
  dash();
  dash();
  return;
case 'F':
  dot();
  dot();
  dash();
  dot();
  return;
case 'G':
  dash();
  dash();
  dot();
  return;
case 'Y':
  dash();
  dot();
  dash();
  dash();
  return;
case 'P':
  dot();
  dash();
  dash();
  dot();
  return;
case 'B':
  dash();
  dot();
  dot();
  dot();
  return;
case 'V':
  dot();
  dot();
  dot();
  dash();
  return;
case 'K':
```

```
    dash();
    dot();
    dash();
    return;
  case 'J':
    dot();
    dash();
    dash();
    dash();
    return;
  case 'X':
    dash();
    dot();
    dot();
    dash();
    return;
  case 'Q':
    dash();
    dash();
    dot();
    dash();
    return;
  case 'Z':
    dash();
    dash();
    dot();
    dot();
    return;
  case ' ':
    delay(wgap);
    return;
  }
}

void setup() {

  // Serial.begin(9600);
  if(modo == 0){ON = 1; OFF = 0;}  // Diretto
  if(modo == 1){ON = 0; OFF = 1;}  // Con sheda relay

  pinMode(RELAY_1,OUTPUT);
  pinMode(RELAY_2,OUTPUT);
  pinMode(RELAY_3,OUTPUT);
  pinMode(RELAY_4,OUTPUT);
  pinMode(RELAY_5,OUTPUT);
  pinMode(RELAY_6,OUTPUT);
  pinMode(RELAY_7,OUTPUT);
  pinMode(RELAY_8,OUTPUT);
```

```
  digitalWrite(RELAY_1, OFF);
  digitalWrite(RELAY_2, OFF);
  digitalWrite(RELAY_3, OFF);
  digitalWrite(RELAY_4, OFF);
  digitalWrite(RELAY_5, OFF);
  digitalWrite(RELAY_6, OFF);
  digitalWrite(RELAY_7, OFF);
  digitalWrite(RELAY_8, OFF);

// Inizializzazione timer canali
start_1 = millis();
start_2 = millis();
start_3 = millis();
start_4 = millis();
start_5 = millis();
start_6 = millis();
start_7 = millis();
start_8 = millis();

}  // end setup


void loop ()
{

// temporizzazione canale 1
#####################################################################
if(trig_1 == 0){
t_event_1 = ch_1[trig_1]*convtime;
t_stat_1 = ch_1[trig_1+1];
if (t_stat_1 == 0){ digitalWrite(RELAY_1, OFF); } else {digitalWrite(RELAY_1, ON);}
}
if((millis() - start_1) >= t_event_1 ) {
  trig_1 = trig_1+2;
  if(trig_1 == ch_1_count) { trig_1 = 0;}
  start_1 = millis();
  t_event_1 = ch_1[trig_1]*convtime;
  t_stat_1 = ch_1[trig_1+1];
  if (t_stat_1 == 0){ digitalWrite(RELAY_1, OFF); } else {digitalWrite(RELAY_1, ON);}
}

// temporizzazione canale 2
#####################################################################
if(trig_2 == 0){
t_event_2 = ch_2[trig_2]*convtime;
t_stat_2 = ch_2[trig_2+1];
if (t_stat_2 == 0){ digitalWrite(RELAY_2, OFF); } else {digitalWrite(RELAY_2, ON);}
```

```
}
if((millis() - start_2) >= t_event_2 ) {
  trig_2 = trig_2+2;
  if(trig_2 == ch_2_count) { trig_2 = 0;}
  start_2 = millis();
  t_event_2 = ch_2[trig_2]*convtime;
  t_stat_2 = ch_2[trig_2+1];
  if (t_stat_2 == 0){ digitalWrite(RELAY_2, OFF); } else {digitalWrite(RELAY_2, ON);}
}

// temporizzazione canale 3
######################################################################
if(trig_3 == 0){
t_event_3 = ch_3[trig_3]*convtime;
t_stat_3 = ch_3[trig_3+1];
if (t_stat_3 == 0){ digitalWrite(RELAY_3, OFF); } else {digitalWrite(RELAY_3, ON);}
}
if((millis() - start_3) >= t_event_3 ) {
  trig_3 = trig_3+2;
  if(trig_3 == ch_3_count) { trig_3 = 0;}
  start_3 = millis();
  t_event_3 = ch_3[trig_3]*convtime;
  t_stat_3 = ch_3[trig_3+1];
  if (t_stat_3 == 0){ digitalWrite(RELAY_3, OFF); } else {digitalWrite(RELAY_3, ON);}
}

// temporizzazione canale 4
######################################################################
if(trig_4 == 0){
t_event_4 = ch_4[trig_4]*convtime;
t_stat_4 = ch_4[trig_4+1];
if (t_stat_4 == 0){ digitalWrite(RELAY_4, OFF); } else {digitalWrite(RELAY_4, ON);}
}
if((millis() - start_4) >= t_event_4 ) {
  trig_4 = trig_4+2;
  if(trig_4 == ch_4_count) { trig_4 = 0;}
  start_4 = millis();
  t_event_4 = ch_4[trig_4]*convtime;
  t_stat_4 = ch_4[trig_4+1];
  if (t_stat_4 == 0){ digitalWrite(RELAY_4, OFF); } else {digitalWrite(RELAY_4, ON);}
}

// temporizzazione canale 5
######################################################################
if(trig_5 == 0){
t_event_5 = ch_5[trig_5]*convtime;
t_stat_5 = ch_5[trig_5+1];
if (t_stat_5 == 0){ digitalWrite(RELAY_5, OFF); } else {digitalWrite(RELAY_5, ON);}
```

```
}
if((millis() - start_5) >= t_event_5 ) {
  trig_5 = trig_5+2;
  if(trig_5 == ch_5_count) { trig_5 = 0;}
  start_5 = millis();
  t_event_5 = ch_5[trig_5]*convtime;
  t_stat_5 = ch_5[trig_5+1];
  if (t_stat_5 == 0){ digitalWrite(RELAY_5, OFF); } else {digitalWrite(RELAY_5, ON);}
}

// temporizzazione canale 6
#####################################################################
if(trig_6 == 0){
t_event_6 = ch_6[trig_6]*convtime;
t_stat_6 = ch_6[trig_6+1];
if (t_stat_6 == 0){ digitalWrite(RELAY_6, OFF); } else {digitalWrite(RELAY_6, ON);}
}
if((millis() - start_6) >= t_event_6 ) {
  trig_6 = trig_6+2;
  if(trig_6 == ch_6_count) { trig_6 = 0;}
  start_6 = millis();
  t_event_6 = ch_6[trig_6]*convtime;
  t_stat_6 = ch_6[trig_6+1];
  if (t_stat_6 == 0){ digitalWrite(RELAY_6, OFF); } else {digitalWrite(RELAY_6, ON);}
}

// temporizzazione canale 7
#####################################################################
if(trig_7 == 0){
t_event_7 = ch_7[trig_7]*convtime;
t_stat_7 = ch_7[trig_7+1];
if (t_stat_7 == 0){ digitalWrite(RELAY_7, OFF); } else {digitalWrite(RELAY_7, ON);}
}
if((millis() - start_7) >= t_event_7 ) {
  trig_7 = trig_7+2;
  if(trig_7 == ch_7_count) { trig_7 = 0;}
  start_7 = millis();
  t_event_7 = ch_7[trig_7]*convtime;
  t_stat_7 = ch_7[trig_7+1];
  if (t_stat_7 == 0){ digitalWrite(RELAY_7, OFF); } else {digitalWrite(RELAY_7, ON);}
}

// temporizzazione canale 8 - MORSE
#####################################################

  if(trig_8 == 0){
        t_event_8 = ch_8[trig_8]*convtime;
        digitalWrite(RELAY_8, OFF);
```

```
        }

  if(ch_8[trig_8] == 99) {

        // carattere inizio messaggio
        dash();
        dot();
        dash();
        dot();
        dash();

        for(int k=0; k<=l_frase; k++){
        soundLetter(frase[k]);
        delay(lgap);
        }

        // carattere fine messaggio
        dot();
        dash();
        dot();
        dash();
        dot();

        trig_8 = trig_8+2;
        if(trig_8 == ch_8_count) { trig_8 = 0;}
        t_event_8 = ch_8[trig_8]*convtime;
        }

  if((millis() - start_8) >= t_event_8) {
        trig_8 = trig_8+2;
        if(trig_8 == ch_8_count) { trig_8 = 0;}
        t_event_8 = ch_8[trig_8]*convtime;
        start_8 = millis();
        digitalWrite(RELAY_8, OFF);
        }

} // end loop
```