GUIDA RAPIDA

SYNTH DIY

DANIELE MURGIA © 2019-20

1. STRUTTURA BASE DEL CODICE

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>
#include <Artboard.h>

Artboard artboard;

// GUItool: begin automatically generated code
// Qui vanno gli oggetti audio (oscillatori, filtri, mixer...)
AudioSynthWaveformSine sine1;
AudioOutputAnalog dac0;
AudioConnection patchCord1(sine1, 0, dac0, 0);
// GUItool: end automatically generated code

void setup() {
  Serial.begin(9600);
  AudioMemory(12);  // Memoria per l'audio

  // Configurazione iniziale (eseguita UNA SOLA VOLTA)
  sine1.frequency(440);
  sine1.amplitude(0.5);
}

void loop() {
  // Codice che si ripete continuamente
  int potValue = artboard.pot(0);
  // ...
}
```

2. DIFFERENZA TRA SETUP() E LOOP()

SETUP()

- ESEGUITO UNA SOLA VOLTA ALL'AVVIO
- USALO PER:
- INIZIALIZZARE IL SERIAL MONITOR
- ALLOCARE MEMORIA AUDIO CON `AudioMemory()`
- IMPOSTARE VALORI INIZIALI (FREQUENZE, AMPIEZZE)
- CONFIGURARE I MIXER

LOOP()

- ESEGUITO CONTINUAMENTE (CIRCA 100 VOLTE AL SECONDO)
- USALO PER:
- LEGGERE I SENSORI (TOUCH, POT, BUTTON)
- CONTROLLARE GLI INVILUPPI
- AGGIORNARE I PARAMETRI IN TEMPO REALE

ESEMPIO:

```
void setup() {
  sine1.frequency(440);  // Imposta SOLO all'inizio
}

void loop() {
  int freq = map(artboard.pot(0), 0, 1023, 100, 1000);
  sine1.frequency(freq);  // Aggiorna CONTINUAMENTE
}
```

---

3. LIBRERIA ARTBOARD - API REFERENCE

INPUT CAPACITIVI (TOUCH)

```
int valore = artboard.touch(pin);  // pin: 0-11
// Ritorna: 0-65535 (più alto = più tocco)
// Soglia tipica: > 6000 per rilevare tocco
```

PULSANTI

```
int stato = artboard.button(pin);  // pin: 0-7
// Ritorna: LOW (premuto) o HIGH (rilasciato)
```

POTENZIOMETRI

```
int valore = artboard.pot(pin);  // pin: 0-7
// Ritorna: 0-1023
```

---

4. SERIAL MONITOR

APRIRE IL SERIAL MONITOR

1. CARICA LO SKETCH SU TEENSY
2. CLICCA L'ICONA LENTE DI INGRANDIMENTO (IN ALTO A DESTRA)
3. IMPOSTA VELOCITÀ: 9600 BAUD

STAMPARE MESSAGGI

```
void setup() {
  Serial.begin(9600);  // Inizializza SEMPRE nel setup
  Serial.println("Programma avviato!");
}

void loop() {
  int pot = artboard.pot(0);

  // Stampa semplice
  Serial.println(pot);

  // Stampa con etichetta
  Serial.print("Potenziometro: ");
  Serial.println(pot);

  // Stampa formattata
  Serial.print("Pot 0: ");
  Serial.print(pot);
  Serial.println(" (0-1023)");
}
```

ATTENZIONE: TROPPI `Serial.println()` NEL LOOP RALLENTANO IL PROGRAMMA!

---

5. CONTROLLO DI STATO (STATE MANAGEMENT)

PROBLEMA

SE AGGIORNI QUALCOSA AD OGNI LOOP, ANCHE QUANDO NON È CAMBIATO, SPRECHI RISORSE:

```
// ❌ SBAGLIATO - aggiorna sempre, anche se non serve
void loop() {
  int ottava = 4;
  sine1.frequency(440 * ottava);  // Chiamato 100 volte/sec inutilmente!
}
```

SOLUZIONE: VARIABILE DI STATO

SALVA IL VALORE PRECEDENTE E AGGIORNA SOLO QUANDO CAMBIA:

```
int lastOctave = -1;  // Variabile globale (fuori da setup/loop)

void loop() {
  int potOctave = artboard.pot(2);
  int currentOctave;

  if (potOctave < 205) currentOctave = 1;
  else if (potOctave < 410) currentOctave = 2;
  else if (potOctave < 615) currentOctave = 4;
  else if (potOctave < 820) currentOctave = 8;
  else currentOctave = 16;

  // ✅ CORRETTO - aggiorna solo se è cambiato
  if (currentOctave != lastOctave) {
    Serial.print("Nuova ottava: ");
    Serial.println(currentOctave);

    sine1.frequency(440 * currentOctave);

    lastOctave = currentOctave;  // Salva per il prossimo confronto
  }
}
```

VANTAGGI: - RIDUCE IL CARICO DEL PROCESSORE - EVITA MESSAGGI SERIAL DUPLICATI - PREVIENE GLITCH AUDIO

---

6. MAPPATURA VALORI CON MAP()

LA FUNZIONE `map()` TRASFORMA UN VALORE DA UN RANGE A UN ALTRO:

```
int nuovoValore = map(vecchioValore, fromLow, fromHigh, toLow, toHigh);
```

ESEMPI COMUNI

POTENZIOMETRO → FREQUENZA:

```
int pot = artboard.pot(0);  // 0-1023
int freq = map(pot, 0, 1023, 100, 1000);  // 100-1000 Hz
sine1.frequency(freq);
```

POTENZIOMETRO → TEMPO (MILLISECONDI):

```
int attackTime = map(artboard.pot(0), 0, 1023, 5, 1000);  // 5-1000ms
envelope.attack(attackTime);
```

POTENZIOMETRO → VALORI MIDI (0-127):

```
int midiValue = map(artboard.pot(0), 0, 1023, 0, 127);
```

---

7. CONTROLLO TOUCH - LOGICA SEMPLIFICATA

```
void loop() {
  int attackTime = map(artboard.pot(0), 0, 1023, 5, 1000);
  int releaseTime = map(artboard.pot(1), 0, 1023, 5, 1000);

  // Logica semplice: la libreria Audio ignora comandi duplicati
  if (artboard.touch(0) > 6000) {
    envelope.amplitude(1.0, attackTime);  // Nota ON
  }
  else {
    envelope.amplitude(0.0, releaseTime);  // Nota OFF
  }
}
```

PERCHÉ FUNZIONA SENZA EDGE DETECTION?

LA LIBRERIA AUDIO È "INTELLIGENTE": SE CONTINUI A DIRE `amplitude(1.0, 100)` MENTRE È GIÀ A 1.0, IGNORA I COMANDI DUPLICATI. REAGISCE SOLO QUANDO CAMBIA IL TARGET (DA 1.0 A 0.0 O VICEVERSA).

---

8. DIVISIONE POTENZIOMETRO IN ZONE

PER CREARE "STEPS" DISCRETI INVECE DI VALORI CONTINUI:

```
int potValue = artboard.pot(2);  // 0-1023
int zona;

if (potValue < 205) zona = 1;        // Zona 1: 0-204
else if (potValue < 410) zona = 2;   // Zona 2: 205-409
else if (potValue < 615) zona = 4;   // Zona 3: 410-614
else if (potValue < 820) zona = 8;   // Zona 4: 615-819
else zona = 16;                      // Zona 5: 820-1023

Serial.print("Zona attiva: ");
Serial.println(zona);
```

CALCOLO DELLE SOGLIE:

PER N ZONE: `1024 / N = dimensione zona`
PER 5 ZONE: `1024 / 5 ≈ 205` PER ZONA

---

9. GESTIONE BOTTONI (EDGE DETECTION)

I BOTTONI RICHIEDONO EDGE DETECTION PERCHÉ VOGLIAMO REAGIRE SOLO AL CLICK, NON MENTRE TIENI PREMUTO:

```
bool btn0_last = false;  // Variabile globale

void loop() {
  bool btn0_now = (artboard.button(0) == LOW);  // LOW = premuto

  // Rileva il FRONTE di salita (momento del click)
  if (btn0_now && !btn0_last) {
    Serial.println("Bottone cliccato!");
    // Esegui azione UNA SOLA VOLTA
  }

  btn0_last = btn0_now;  // Salva per il prossimo loop
}
```

---

10. TROUBLESHOOTING COMUNE

IL SUONO NON SI SENTE

- ✅ HAI CHIAMATO `AudioMemory(12)` NEL SETUP?

- ✅ HAI IMPOSTATO L'AMPLITUDE > 0?
- ✅ LE AUDIOCONNECTION SONO CORRETTE?
- ✅ IL VOLUME DEL COMPUTER È ALZATO?

IL SERIAL MONITOR NON FUNZIONA

- ✅ HAI CHIAMATO `Serial.begin(9600)` NEL SETUP?
- ✅ IL SERIAL MONITOR È IMPOSTATO A 9600 BAUD?
- ✅ NEL MENU TOOLS → USB TYPE HAI SELEZIONATO "SERIAL"?

IL TOUCH NON RISPONDE

- ✅ LA SOGLIA (6000) POTREBBE ESSERE TROPPO ALTA, PROVA 4000-5000
- ✅ STAMPA IL VALORE RAW CON `Serial.println(artboard.touch(0))`

IL PROGRAMMA È LENTO

- ✅ RIDUCI I `Serial.println()` NEL LOOP
- ✅ AUMENTA IL `delay()` ALLA FINE DEL LOOP (ES. DA 5 A 10MS)
- ✅ USA IL CONTROLLO DI STATO PER EVITARE AGGIORNAMENTI INUTILI

---

11. ESEMPI PRATICI

OSCILLATORE CONTROLLATO DA POT

```
void loop() {
  int freq = map(artboard.pot(0), 0, 1023, 100, 2000);
  sine1.frequency(freq);
  delay(5);
}
```

NOTA ON/OFF CON TOUCH

```
void loop() {
  if (artboard.touch(0) > 6000) {
    sine1.amplitude(0.8);
  } else {
    sine1.amplitude(0.0);
  }
  delay(5);
}
```

TOGGLE LED CON BOTTONE

```
bool ledState = false;
bool btn_last = false;

void loop() {
  bool btn_now = (artboard.button(0) == LOW);

  if (btn_now && !btn_last) {
    ledState = !ledState;  // Inverte lo stato
    digitalWrite(LED_PIN, ledState ? HIGH : LOW);
  }

  btn_last = btn_now;
  delay(5);
}
```

PER DOMANDE E SUPPORTO:
SGTMURGIA@GMAIL.COM

REPOSITORY E DOCUMENTAZIONE:
HTTPS://GITHUB.COM/FRMURGIA/ARTBOARD_LIBRARY