Generazione di un'Onda Sinusoidale con ESP32

Questo documento descrive come generare un'onda sinusoidale utilizzando un ESP32. Include dettagli sul codice, i parametri configurabili e i comandi disponibili.

Parametri Configurabili

Parametro	Descrizione	Valore Predefinito
`N_SAMPLES`	Numero di campioni per ciclo	100
`ampV`	Ampiezza dell'onda (Volt)	1.0
`offsetV`	Offset dell'onda (Volt)	1.5
`freqHz`	Frequenza dell'onda (Hz)	2.0
`Vref`	Tensione massima DAC (logica)	5.0

Funzionalità Principali

1. Setup del Sistema

Inizializza la comunicazione seriale e calcola l'intervallo tra i campioni.

```
void setup() {
  Serial.begin(115200);
  delay(1000);  // Attesa per stabilità seriale
  recalculateTiming();
  inputString.reserve(20);  // Ottimizzazione memoria per stringa
  Serial.println("Sinusoide pronta. Comandi: A=amp O=offset F=freq N=n°campioni");
}
```

2. Generazione dell'Onda Sinusoidale

Calcola i valori dell'onda sinusoidale e li invia via seriale.

```
void loop() {
  // Gestione dei comandi ricevuti
  if (stringComplete) {
    parseCommand(inputString);
    inputString = "";  // Reset stringa
    stringComplete = false;
  }

  // Generazione del segnale sinusoidale
  unsigned long now = micros();
  if (now - lastSampleTime >= sampleIntervalMicros) {
    lastSampleTime = now;

    // Calcolo del valore della sinusoide
    float angle = 2.0 * PI * i / N_SAMPLES;
    float voltage = ampV * sin(angle) + offsetV;
    voltage = constrain(voltage, 0.0, Vref);  // Limita il range analogico

    Serial.println(voltage * 1000.0, 3);  // Output in millivolt con 3 decimali

    i = (i + 1) % N_SAMPLES;

    // Misura della frequenza effettiva ogni ciclo completo
    if (i == 0) {
      unsigned long cycleDuration = micros() - lastCycleTime;
      lastCycleTime = micros();
      float actualFreq = 1000000.0 / cycleDuration;
      Serial.print("#FreqEffettiva: ");
      Serial.print(actualFreq, 3);
      Serial.println(" Hz");
    }
  }
}
```

3. Ricalcolo dell'Intervallo tra i Campioni

Aggiorna l'intervallo di campionamento in base alla frequenza e al numero di campioni.

```
void recalculateTiming() {
  sampleIntervalMicros = (1.0 / (freqHz * N_SAMPLES)) * 1e6;
}
```

4. Parsing dei Comandi Seriali

Permette di configurare i parametri tramite comandi inviati via seriale.

Comando	Descrizione	Formato	Esempio
A	Imposta ampiezza	A<valore>	A2.5
O	Imposta offset	O<valore>	O1.0
F	Imposta frequenza	F<valore>	F50.0
N	Imposta numero di campioni	N<valore>	N200

```
void parseCommand(String cmd) {
  char type = cmd.charAt(0);
  float value = cmd.substring(1).toFloat();

  switch (type) {
    case 'A':  // Imposta ampiezza
      ampV = constrain(value, 0.0, MAX_AMP_V);
      Serial.print("#Ampiezza: "); Serial.println(ampV);
      break;
    case 'O':  // Imposta offset
      offsetV = constrain(value, 0.0, Vref);
      Serial.print("#Offset: "); Serial.println(offsetV);
      break;
    case 'F':  // Imposta frequenza
      freqHz = constrain(value, 0.1, MAX_FREQ_HZ);
      recalculateTiming();
      Serial.print("#Frequenza: "); Serial.println(freqHz);
      break;
    case 'N':  // Imposta numero di campioni
      if (value >= 10 && value <= 2000) {
        N_SAMPLES = (unsigned int)value;
        recalculateTiming();
        Serial.print("#Campioni: "); Serial.println(N_SAMPLES);
      }
      break;
  }
}
```

5. Gestione degli Eventi Seriali

Raccoglie i dati inviati via seriale e li processa.

```
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      stringComplete = true;
    } else {
      inputString += inChar;
    }
  }
}
```

Note

- Assicurarsi che i valori dei parametri siano entro i limiti specificati.
- La frequenza effettiva viene calcolata e stampata ogni ciclo completo.