Generatore di Onde con ESP32

Questo documento descrive come configurare e utilizzare un generatore di onde basato su ESP32. Il codice supporta diverse forme d'onda e parametri configurabili tramite comandi seriali.

Configurazione Iniziale

Definire i parametri di default e le costanti principali:

```
#define DEFAULT_SAMPLES 100
#define MAX_AMP_V 10.0
#define MAX_FREQ_HZ 2000.0

unsigned int N_SAMPLES = DEFAULT_SAMPLES;
float ampV = 1.0;
float offsetV = 1.5;
float freqHz = 2.0;

const float Vref = 3.3;
unsigned long sampleIntervalMicros;
unsigned long lastSampleTime = 0;

String inputString = "";
bool stringComplete = false;

int i = 0;
unsigned long lastCycleTime = 0;
```

Tipi di Onde Supportate

Il generatore supporta i seguenti tipi di onde:

```
enum WaveType { SINE, SQUARE, TRIANGLE, SAWTOOTH };
WaveType currentWave = SINE;
```

Funzioni Principali

Setup

Configurare la comunicazione seriale e calcolare il timing iniziale:

```
void setup() {
  Serial.begin(115200);
  delay(1000);
  recalculateTiming();
  inputString.reserve(20);
  Serial.println("Pronto. Comandi: A=amp O=offset F=freq N=campioni W=onda (sin, tri, saw
}
```

Loop

Gestire i comandi seriali e generare i campioni dell'onda:

```
void loop() {
  if (stringComplete) {
    parseCommand(inputString);
    inputString = "";
    stringComplete = false;
  }

  unsigned long now = micros();
  if (now - lastSampleTime >= sampleIntervalMicros) {
    lastSampleTime = now;

    float voltage = generateWaveSample(i);
    voltage = constrain(voltage, 0.0, Vref);
    Serial.println(voltage * 1000.0, 3); // millivolt

    i = (i + 1) % N_SAMPLES;

    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");
    }
  }
}
```

Generazione dell'Onda

Calcolare il valore del campione in base al tipo di onda selezionato:

```
float generateWaveSample(int index) {
  float t = (float)index / N_SAMPLES;  // tempo normalizzato [0,1)

  switch (currentWave) {
    case SINE:
      return ampV * sin(2 * PI * t) + offsetV;

    case SQUARE:
      return (t < 0.5 ? ampV : -ampV) + offsetV;

    case TRIANGLE:
      return (2 * ampV * abs(2 * t - 1) - ampV) + offsetV;

    case SAWTOOTH:
      return (2 * ampV * t - ampV) + offsetV;

    default:
      return offsetV; // fallback
  }
}
```

Ricalcolo del Timing

Aggiornare l'intervallo tra i campioni in base alla frequenza e al numero di campioni:

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

Comandi Seriali

Analisi dei Comandi

Interpretare i comandi inviati tramite seriale per modificare i parametri:

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

  switch (type) {
    case 'A':
      ampV = constrain(arg.toFloat(), 0.0, MAX_AMP_V);
      Serial.print("#Ampiezza: "); Serial.println(ampV);
      break;
    case 'O':
      offsetV = constrain(arg.toFloat(), 0.0, Vref);
      Serial.print("#Offset: "); Serial.println(offsetV);
      break;
    case 'F':
      freqHz = constrain(arg.toFloat(), 0.1, MAX_FREQ_HZ);
      recalculateTiming();
      Serial.print("#Frequenza: "); Serial.println(freqHz);
      break;
    case 'N':
      if (arg.toInt() >= 10 && arg.toInt() <= 2000) {
        N_SAMPLES = arg.toInt();
        recalculateTiming();
        Serial.print("#Campioni: "); Serial.println(N_SAMPLES);
      }
      break;
    case 'W': // Selezione forma d'onda
      arg.toLowerCase();
      if (arg.startsWith("sin")) {
        currentWave = SINE;
        Serial.println("#Onda: sinusoide");
      } else if (arg.startsWith("tri")) {
        currentWave = TRIANGLE;
        Serial.println("#Onda: triangolare");
      } else if (arg.startsWith("saw")) {
        currentWave = SAWTOOTH;
        Serial.println("#Onda: dente di sega");
      } else if (arg.startsWith("sq")) {
        currentWave = SQUARE;
        Serial.println("#Onda: quadra");
      } else {
        Serial.println("#Onda non riconosciuta");
      }
      break;
```

```
  }
}
```

Lettura Serial

Gestire l'input seriale:

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