```
#include "stm32g4xx.h"

#include <math.h>

 #define PI 3.14159265359

#define SAMPLE_RATE 10000 // Sample rate in Hz

 #define SIN_FREQ 1000 // Sinusfrequenz in Hz

#define TIMER_CLOCK_FREQ 170000000 // Timer clock frequency in Hz

volatile uint16_t lookup_table[256]; // Lookup table for sin values

void configure_pins() {

RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN; // Enable GPIOB clock

GPIOB->MODER &= ~GPIO_MODER_MODE0; // Clear mode for PB0

GPIOB->MODER |= GPIO_MODER_MODE0_0; // Set PB0 as output

GPIOB->OSPEEDR |= GPIO_OSPEEDR_OSPEED0; // Set PB0 as high speed

 }

 void configure_timer() {

 RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN; // Enable TIM2 clock

TIM2->CR1 = 0; // Set timer to default values

TIM2->PSC = (TIMER_CLOCK_FREQ / SAMPLE_RATE) - 1; // Set prescaler

TIM2->ARR = (TIMER_CLOCK_FREQ / SAMPLE_RATE / SIN_FREQ) - 1; // Set auto-
reload value

 TIM2->DIER |= TIM_DIER_UIE; // Enable update interrupt

 NVIC_EnableIRQ(TIM2_IRQn); // Enable TIM2 IRQ

 }

void generate_sin_wave() {

 for (int i = 0; i < 256; i++) {

 lookup_table[i] = 2047 * sin(2 * PI * i / 256) + 2048; // Fill sine lookup table

 }
} void TIM2_IRQHandler() {
if (TIM2->SR & TIM_SR_UIF) {
static uint16_t phase = 0; // Phase accumulator
phase += 256 * SIN_FREQ / SAMPLE_RATE;
 GPIOB->ODR = lookup_table[phase >> 8]; // Output sin value to GPIO
TIM2->SR &= ~TIM_SR_UIF; // Clear timer interrupt flag
}
}
 int main() {
 configure_pins();
configure_timer();
 generate_sin_wave();
TIM2->CR1 |= TIM_CR1_CEN; // Enable timer
 while (1) {
 // Main loop
 }
}
```