```
// Active Noise Cancellation (ANC) using NLMS Algorithm on
Raspberry Pi Pico W with Two Microphones
#define SAMPLE_RATE 20000    // 20 kHz sampling rate
#define FILTER_ORDER 37      // Adaptive filter order
#define NLMS_STEP_SIZE 0.000001 // Learning rate

#define MIC1_PIN A0          // GP26 (ADC0) for primary
microphone (speech + noise)
#define MIC2_PIN A1          // GP27 (ADC1) for reference
microphone (noise only)
#define SPEAKER_PIN 15       // GP15 for PWM speaker output

// NLMS Filter Variables
float w[FILTER_ORDER] = {0};  // Adaptive filter weights
float x[FILTER_ORDER] = {0};  // Delay line for noise referen

// Introduce a small delay for better noise cancellation
#define DELAY_SAMPLES 5
float delay_buffer[DELAY_SAMPLES] = {0};
int delay_index = 0;

// Function to apply NLMS algorithm
float nlms_adaptive_filter(float reference, float
error_signal) {
    static float mu = NLMS_STEP_SIZE; // Step size

    // Shift delay line
    for (int i = FILTER_ORDER - 1; i > 0; i--) {
        x[i] = x[i - 1];
    }
    x[0] = reference;

    // Compute filter output (estimated noise)
    float y = 0;
    for (int i = 0; i < FILTER_ORDER; i++) {
        y += w[i] * x[i];
```

```
    }

    // Compute error (desired signal - estimated noise)
    float e = error_signal - y;

    // Update filter weights (NLMS)
    float norm_factor = 0.0001; // Prevent division by zero
    for (int i = 0; i < FILTER_ORDER; i++) {
        norm_factor += x[i] * x[i];
    }
    float step = mu / norm_factor;

    for (int i = 0; i < FILTER_ORDER; i++) {
        w[i] += step * e * x[i];
    }

    return e;  // Return anti-noise signal
}

void setup() {
    Serial.begin(115200);

    // Initialize ADC for both microphones
    analogReadResolution(12);  // 12-bit ADC (0-4095)
    pinMode(MIC1_PIN, INPUT);
    pinMode(MIC2_PIN, INPUT);

    // Initialize PWM for speaker output
    pinMode(SPEAKER_PIN, OUTPUT);
    analogWriteResolution(12); // 12-bit PWM output (0-4095)
}

void loop() {
    // Read ADC inputs from both microphones
    int raw_mic1 = analogRead(MIC1_PIN);
    int raw_mic2 = analogRead(MIC2_PIN);
```

```
    // Normalize ADC readings to [-1,1] range
    float input_signal = ((float)raw_mic1 - 2048) / 2048.0;
    float noise_reference = ((float)raw_mic2 - 2048) / 2048.0

    // Introduce a small delay in noise reference for better
ANC
    float delayed_noise = delay_buffer[delay_index];
    delay_buffer[delay_index] = noise_reference;
    delay_index = (delay_index + 1) % DELAY_SAMPLES;

    // Apply ANC using NLMS
    float anti_noise_signal =
nlms_adaptive_filter(delayed_noise, input_signal);

    // Convert anti-noise signal back to PWM (0 - 4095)
    int pwm_output = (int)((anti_noise_signal + 1.0) * 2047.5
    analogWrite(SPEAKER_PIN, pwm_output);

    // Display the error signal (should approach zero if ANC
works well)
    Serial.println(anti_noise_signal - input_signal);
    //Serial.println(input_signal);

    // Maintain 20 kHz sampling rate
    delayMicroseconds(50);
}
```