```
#define BLYNK_TEMPLATE_ID "TMPL2K8DtjleI"
#define BLYNK_TEMPLATE_NAME "Sistema de monitoramento de enchentes"
#define BLYNK_AUTH_TOKEN "KeIiTvBOYxH8HtWIKBjHkV64ORIPdVEF"

#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// Auth Token do Blynk
char auth[] = "KeIiTvBOYxH8HtWIKBjHkV64ORIPdVEF";  // Substitua com seu
Auth Token

// Credenciais da rede WiFi
char ssid[] = "Iphone de gabi";   // Substitua com o nome da sua rede
WiFi
char pass[] = "3380212900gg";  // Substitua com a senha da sua rede
WiFi
BlynkTimer timer; // Cria um objeto timer para gerenciar intervalos de
tempo no Blynk

void EnviarDadosBlink{
  // Envia os dados para o Blynk
  Blynk.virtualWrite(V1, INSIRA VARIAVEL AQUI);  // Envia a distância
para o Virtual Pin V1
  Blynk.virtualWrite(V2, INSIRA VARIAVEL AQUI);  // Envia a altura da
água para o Virtual Pin V2
  Blynk.virtualWrite(V3, INSIRA VARIAVEL AQUI);  // Envia o volume de
água para o Virtual Pin V3
}
void setup() {
  // Inicia a comunicação serial
  Serial.begin(9600);
  // Conecta ao Blynk usando o Auth Token e as credenciais WiFi
  Blynk.begin(auth, ssid, pass);
}
void loop() {
  // Executa a função principal do Blynk
  Blynk.run();
}

#define D22 22
#define D23 23
```

```
#define D27 27
#define D26 26
#define D25 25

const int trigPin = D22;
const int echoPin = D23;
const int ledVD = D27;
const int ledAM = D26;
const int ledVM = D25;

// Definindo a velocidade do som em cm/µs
#define SOUND_SPEED 0.034

long duration;
float distanceCm;

void setup() {
Serial.begin(115200); // Inicializa a comunicação serial
pinMode(trigPin, OUTPUT); // Configura o trigPin como saída
pinMode(echoPin, INPUT); // Configura o echoPin como entrada
pinMode(ledVD, OUTPUT); // Configura o ledVD como saída
pinMode(ledAM, OUTPUT); // Configura o ledAM como saída
pinMode(ledVM, OUTPUT); // Configura o ledVM como saída
}

void loop() {
// Limpa o trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Envia o pulso de 10 microsegundos
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Lê o tempo do sinal de eco (em microsegundos)
duration = pulseIn(echoPin, HIGH);

// Calcula a distância em centímetros
distanceCm = (duration * SOUND_SPEED) / 2;

// Exibe a distância no Monitor Serial
Serial.print("Distância (cm): ");
```

```
Serial.println(distanceCm);

// Controle dos LEDs baseado na distância
if (distanceCm <= 3.5) {
digitalWrite(ledVM, HIGH); // LED vermelho acende
digitalWrite(ledAM, LOW); // LED amarelo apaga
digitalWrite(ledVD, LOW); // LED verde apaga
} else if (distanceCm > 3.5 && distanceCm <= 9) {
digitalWrite(ledVM, LOW); // LED vermelho apaga
digitalWrite(ledAM, HIGH); // LED amarelo acende
digitalWrite(ledVD, LOW); // LED verde apaga
} else if (distanceCm > 9 && distanceCm <= 12.5) {
digitalWrite(ledVM, LOW); // LED vermelho apaga
digitalWrite(ledAM, LOW); // LED amarelo apaga
digitalWrite(ledVD, HIGH); // LED verde acende
} else {
digitalWrite(ledVM, LOW); // LED vermelho apaga
digitalWrite(ledAM, LOW); // LED amarelo apaga
digitalWrite(ledVD, LOW); // LED verde apaga
}

delay(1000); // Atraso de 1 segundo antes de realizar a próxima leitura
}
```