1. Código para o daemon

```
use rppal::gpio::Gpio;
use std::fs;
use std::thread;
use std::time::Duration;
use syslog::{Facility, Formatter3164};
use std::process;
use nix::unistd::{fork, ForkResult};

// Constantes de configuração
const GPIO_FAN_PIN: u8 = 17; // Pino GPIO para controlar o fan
const TEMP_MIN: f32 = 40.0;  // Temperatura mínima para desligar o fan
const TEMP_MAX: f32 = 50.0;  // Temperatura máxima para ligar o fan
const POLL_INTERVAL: u64 = 10; // Intervalo de verificação da temperatura em segundos

fn get_cpu_temperature() -> f32 {
    // Lê a temperatura da CPU do arquivo do sistema
    let temp_str = fs::read_to_string("/sys/class/thermal/thermal_zone0/temp")
        .expect("Falha ao ler a temperatura da CPU");
    let temp_millicelsius: f32 = temp_str.trim().parse().expect("Temperatura inválida");
    temp_millicelsius / 1000.0
}

fn log_to_syslog(message: &str) {
    // Configura o logger para syslog
    let formatter = Formatter3164 {
        facility: Facility::LOG_USER,
        hostname: None,
        process: "rackbox-fancontroller".into(),
        pid: 0,
    };

    match syslog::unix(formatter) {
        Ok(mut logger) => {
            logger.err(message).expect("Falha ao escrever no syslog");
        }
        Err(e) => eprintln!("Falha ao configurar o syslog: {}", e),
    }
}

fn daemonize() {
    match fork() {
        Ok(ForkResult::Parent { .. }) => {
            process::exit(0);
        }
        Ok(ForkResult::Child) => {
            // O processo filho continua a execução
        }
```

```
        Err(_) => {
            eprintln!("Falha ao criar o daemon");
            process::exit(1);
        }
    }
}

fn main() {
    daemonize();

    let gpio = Gpio::new().expect("Falha ao inicializar o GPIO");
    let mut fan_pin = gpio.get(GPIO_FAN_PIN).unwrap().into_output();

    log_to_syslog("Serviço de controle do fan iniciado.");

    loop {
        let temp = get_cpu_temperature();
        println!("Temperatura da CPU: {:.1}°C", temp);

        if temp >= TEMP_MAX && !fan_pin.is_set_high() {
            fan_pin.set_high(); // Liga o fan
            log_to_syslog(&format!("Fan ligado. Temperatura: {:.1}°C", temp));
        } else if temp <= TEMP_MIN && fan_pin.is_set_high() {
            fan_pin.set_low(); // Desliga o fan
            log_to_syslog(&format!("Fan desligado. Temperatura: {:.1}°C", temp));
        }

        thread::sleep(Duration::from_secs(POLL_INTERVAL));
    }
}
```

2. Compilação do Código

Compile o código Rust para gerar o binário:

```
cargo build --release
```

O binário será gerado em `target/release/nome_do_projeto`.

3. Configuração do Serviço no Systemd

Para gerenciar o daemon como um serviço, você pode criar um arquivo de serviço do systemd.

Crie um arquivo de serviço em `/etc/systemd/system/rackbox-fancontroller.service`:

ini

[Unit]
Description=Rackbox Fan Controller Service
After=network.target

[Service]
ExecStart=/caminho/para/o/binario/rackbox-fancontroller
Restart=always
User=root
Group=root

[Install]
WantedBy=multi-user.target

Substitua `/caminho/para/o/binario/rackbox-fancontroller` pelo caminho completo para o binário que você compilou.

4. Habilitar e Iniciar o Serviço

Depois de criar o arquivo de serviço, habilite e inicie o serviço:

sudo systemctl daemon-reload
sudo systemctl enable rackbox-fancontroller
sudo systemctl start rackbox-fancontroller

5. Verificação do Serviço

Você pode verificar o status do serviço com:

sudo systemctl status rackbox-fancontroller

E verificar os logs com:

journalctl -u rackbox-fancontroller

Conclusão

Agora, o aplicativo está configurado para ser executado como um daemon e gerenciado como um serviço pelo systemd. Ele será iniciado automaticamente na inicialização do sistema e será reiniciado automaticamente em caso de falha.