9980

```
#ifndef _ENV_H
#define _ENV_H

#include <stdint.h>

// Definir níveis de log
#define LOG_LEVEL_NONE  0  // Sem logs
#define LOG_LEVEL_ERROR 1  // Apenas erros
#define LOG_LEVEL_WARN  2  // Avisos e erros
#define LOG_LEVEL_INFO  3  // Informações, avisos e erros
#define LOG_LEVEL_DEBUG 4  // Mensagens de debug, informações, avisos e erros

// Definir qual o nível de log ativo
#define CURRENT_LOG_LEVEL LOG_LEVEL_ERROR

// Funções de log condicional com base no nível
#define LOG_LEVEL_ERROR 1
#define LOG_LEVEL_WARN  2
#define LOG_LEVEL_INFO  3
#define LOG_LEVEL_DEBUG 4

// Defina o nível de log atual
#define CURRENT_LOG_LEVEL LOG_LEVEL_DEBUG

// Macros de log condicional com suporte para mensagens simples e formatadas
#if CURRENT_LOG_LEVEL >= LOG_LEVEL_ERROR
#define LOG_ERROR(fmt, ...)                                 \
    do                                                      \
    {                                                       \
        Serial.printf("[ERROR] " fmt "\n", ##__VA_ARGS__); \
        Serial.flush();                                     \
    } while (0)
#else
#define LOG_ERROR(fmt, ...)
#endif
```

```
#if CURRENT_LOG_LEVEL >= LOG_LEVEL_WARN
#define LOG_WARN(fmt, ...)                                  \
    do                                                      \
    {                                                       \
        Serial.printf("[WARN] " fmt "\n", ##__VA_ARGS__);  \
        Serial.flush();                                     \
    } while (0)
#else
#define LOG_WARN(fmt, ...)
#endif

#if CURRENT_LOG_LEVEL >= LOG_LEVEL_INFO
#define LOG_INFO(fmt, ...)                                  \
    do                                                      \
    {                                                       \
        Serial.printf("[INFO] " fmt "\n", ##__VA_ARGS__);  \
        Serial.flush();                                     \
    } while (0)
#else
#define LOG_INFO(fmt, ...)
#endif

#if CURRENT_LOG_LEVEL >= LOG_LEVEL_DEBUG
#define LOG_DEBUG(fmt, ...)                                  \
    do                                                       \
    {                                                        \
        Serial.printf("[DEBUG] " fmt "\n", ##__VA_ARGS__);  \
        Serial.flush();                                      \
    } while (0)
#else
#define LOG_DEBUG(fmt, ...)
#endif

// WiFi Network Credentials
constexpr char NETWORK_CLIENT[] = "SmartVac_Telemetria";
constexpr char PASSW[] = "br@skem#2023";

constexpr uint32_t DELAY_MS_READER_TASK = 300; //300ms para que os envios sejam feitos a cada
20s
// Define o atraso em milissegundos para tarefas de leitura (10.000 parece ser um número
```

```
mágico)

// Equipment Identifiers
namespace Equipment
{
    constexpr char TAG[] = "9980";
    constexpr char TOPIC[] = "v4/matr0570";
    constexpr char SENDING_SERVER[] = "web.smartvac.app";
    constexpr uint16_t SENDING_PORT = 1883;
    constexpr unsigned long SENDING_VELOCITY = 115200;
    constexpr uint16_t SENDING_KEEPALIVE = 200;
    constexpr uint32_t DELAY_MS = 500; // Delay time in milliseconds for tasks
}

// Voltage Calibration Constants
struct VoltageCalibration
{
    static constexpr float CAL_R =  79.02;
    static constexpr float CAL_S = 71.36;
    static constexpr float CAL_TT =  69.67;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 15.49;
    static constexpr int CAL_S = 17.5;
    static constexpr int CAL_TT =  16.55;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 39;
    static constexpr int CURRENT_S = 33;
    static constexpr int CURRENT_TT = 35;
    static constexpr int TEMPERATURE = 16;
    static constexpr int VOLTAGE_R = 32;
    static constexpr int VOLTAGE_S = 34;
    static constexpr int VOLTAGE_TT = 36;
};
```

```
// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s100000261";
    constexpr char RET[] = "s100000262";
    constexpr char SUC[] = "s100000265";
    constexpr char LL[] = "s100000266";
    constexpr char ENT_CONDES[] = "s100000286";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s100000263";
    constexpr char VOLT_S[] = "s100000269";
    constexpr char VOLT_TT[] = "s100000270";

    constexpr char CURR_R[] = "s100000264";
    constexpr char CURR_S[] = "s100000271";
    constexpr char CURR_TT[] = "s100000272";

    constexpr char BAT_INS[] = "s100000279";
    constexpr char BAT_RET[] = "s100000280";
    constexpr char BAT_SUC[] = "s100000281";
    constexpr char BAT_LL[] = "s100000282";

    constexpr char VIBR_X_SUC[] = "s100000273";
    constexpr char VIBR_Y_SUC[] = "s100000274";
    constexpr char VIBR_Z_SUC[] = "s100000275";

}

// BLE Sensor MAC Addresses
namespace BLEAddresses
{
    constexpr char TEMP_INSU[] = "bc:57:29:13:fd:07"; //Ble2
    constexpr char TEMP_RET[] = "bc:57:29:13:fd:0b";  //Ble1
    constexpr char TEMP_SUC[] = "bc:57:29:13:fe:f4";  //Ble4
    constexpr char TEMP_LL[] = "bc:57:29:13:fe:f7";   //Ble3
    constexpr char TEMP_ENT_CONDES[] = "bc:57:29:13:fd:0c";  //Ble4
    constexpr char TEMP_SAI_CONDES[] = "";  //Ble4
};
```

```
#endif
```

Revision #1
Created 9 June 2025 12:07:03 by Patrick Leal
Updated 9 June 2025 12:07:43 by Patrick Leal