10243

```
#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[] = "10243";
    constexpr char TOPIC[] = "v4/matr0564";
    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 =  491.29;
    static constexpr float CAL_S = 103.06;
    static constexpr float CAL_TT = 104.82;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 13;
    static constexpr int CAL_S =  13.17;
    static constexpr int CAL_TT = 13.5;
};

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

```
// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s100000160";
    constexpr char RET[] = "s100000161";
    constexpr char SUC[] = "s100000162";
    constexpr char LL[] = "s100000163";
    constexpr char ENT_CONDES[] = "s100000177";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s100000164";
    constexpr char VOLT_S[] = "s100000173";
    constexpr char VOLT_TT[] = "s100000174";

    constexpr char CURR_R[] = "s100000165";
    constexpr char CURR_S[] = "s100000175";
    constexpr char CURR_TT[] = "s100000176";

    constexpr char BAT_INS[] = "s100000169";
    constexpr char BAT_RET[] = "s100000170";
    constexpr char BAT_SUC[] = "s100000171";
    constexpr char BAT_LL[] = "s100000172";

    constexpr char VIBR_X_SUC[] = "s100000166";
    constexpr char VIBR_Y_SUC[] = "s100000167";
    constexpr char VIBR_Z_SUC[] = "s100000168";

}

// BLE Sensor MAC Addresses
namespace BLEAddresses
{
    constexpr char TEMP_INSU[] = "bc:57:29:13:fc:f4"; //Ble2
    constexpr char TEMP_RET[] = "bc:57:29:13:fc:f7";  //Ble1
    constexpr char TEMP_SUC[] = "bc:57:29:13:ff:6a";  //Ble4
    constexpr char TEMP_LL[] = "bc:57:29:13:ff:85";   //Ble3
    constexpr char TEMP_ENT_CONDES[] = "bc:57:29:0e:19:d7";  //Ble4
    constexpr char TEMP_SAI_CONDES[] = "";  //Ble4
};
```

```
#endif
```

Revision #1
Created 23 May 2025 12:20:15 by Patrick Leal
Updated 23 May 2025 12:20:59 by Patrick Leal