GSC

- 2120
- 23715
- 23716
- 23717
- 30145

2120

```
#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[] = "GSC";
constexpr char PASSW[] = "#2021alta!";

#define MDASH_APP_NAME "SEDE_2120"    //nome de usuario do madash para esse equipamento
```

```
#define DEVICE_PASSWORD "aOmyn290Dz1SMlXk499qX4aw"  //senha do mdash

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[] = "2120";
    constexpr char TOPIC[] = "v4/matr0026";
    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 =  52;
    static constexpr float CAL_S = 84.7;
    static constexpr float CAL_TT = 82.3;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 6.15;
    static constexpr int CAL_S = 13.75;
    static constexpr int CAL_TT = 15.9;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 39;
    static constexpr int CURRENT_S = 35;
```

```
    static constexpr int CURRENT_TT = 33;
    static constexpr int TEMPERATURE = 14;
    static constexpr int VOLTAGE_R = 34;
    static constexpr int VOLTAGE_S = 36;
    static constexpr int VOLTAGE_TT = 32;
};

// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s572";
    constexpr char RET[] = "s571";
    constexpr char SUC[] = "s597";
    constexpr char LL[] = "s573";
    constexpr char ENT_CONDES[] = "s598";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s574";
    constexpr char VOLT_S[] = "xxxx";
    constexpr char VOLT_TT[] = "xxxx";

    constexpr char CURR_R[] = "s575";
    constexpr char CURR_S[] = "xxx";
    constexpr char CURR_TT[] = "xxx";

    constexpr char BAT_INS[] = "XXX";
    constexpr char BAT_RET[] = "XXX";
    constexpr char BAT_SUC[] = "XXX";
    constexpr char BAT_LL[] = "XXX";

    constexpr char VIBR_X_SUC[] = "XXX";
    constexpr char VIBR_Y_SUC[] = "XXX";
    constexpr char VIBR_Z_SUC[] = "XXX";

}


#define TEMP_RET {0x28, 0x75, 0xF7, 0x95, 0xF0, 0xFF, 0x3C, 0xEB}  //endereço para a leitura
do r de temperatura de retorno
```

```
#define TEMP_INSU {0x28, 0x49, 0x37, 0x95, 0xF0, 0x01, 0x3C, 0x1C}  //endereço para a leitura do r de temperatura de insuflamento

#define TEMP_EXT {0x28, 0xEE, 0xD8, 0x95, 0xF0, 0x01, 0x3C, 0x70}  //endereço para a leitura do r de temperatura de externa

#define TEMP_LL {0x28, 0xCF, 0x11, 0x95, 0xF0, 0x01, 0x3C, 0x20}

#define TEMP_SUC {0x28, 0xD7, 0x4F, 0x95, 0xF0, 0x01, 0x3C, 0xED}  //endereço para a leitura do sensor de temperatura de sucção
#endif
```

23715

```
#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[] = "GSC";
constexpr char PASSW[] = "#2021alta!";

#define MDASH_APP_NAME "SEDE_23715"    //nome de usuario do madash para esse equipamento
```

```
#define DEVICE_PASSWORD "99B4BwvEulFImu09190qsWP7w"  //senha do mdash

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[] = "23715";
    constexpr char TOPIC[] = "v4/matr0016";
    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 =  50;
    static constexpr float CAL_S = 84.7;
    static constexpr float CAL_TT = 82.3;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 5.7;
    static constexpr int CAL_S = 13.75;
    static constexpr int CAL_TT = 15.9;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 39;
    static constexpr int CURRENT_S = 35;
```

```
    static constexpr int CURRENT_TT = 33;
    static constexpr int TEMPERATURE = 14;
    static constexpr int VOLTAGE_R = 34;
    static constexpr int VOLTAGE_S = 36;
    static constexpr int VOLTAGE_TT = 32;
};

// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s1030";
    constexpr char RET[] = "s1031";
    constexpr char SUC[] = "s1032";
    constexpr char LL[] = "s1033";
    constexpr char ENT_CONDES[] = "s1034";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s1035";
    constexpr char VOLT_S[] = "xxxx";
    constexpr char VOLT_TT[] = "xxxx";

    constexpr char CURR_R[] = "s1036";
    constexpr char CURR_S[] = "xxx";
    constexpr char CURR_TT[] = "xxx";

    constexpr char BAT_INS[] = "XXX";
    constexpr char BAT_RET[] = "XXX";
    constexpr char BAT_SUC[] = "XXX";
    constexpr char BAT_LL[] = "XXX";

    constexpr char VIBR_X_SUC[] = "XXX";
    constexpr char VIBR_Y_SUC[] = "XXX";
    constexpr char VIBR_Z_SUC[] = "XXX";

}


#define TEMP_RET {0x28, 0x37, 0x5B, 0x56, 0xB5, 0x01, 0x3C, 0x12}  //endereço para a leitura
do r de temperatura de retorno
```

```
#define TEMP_INSU {0x28, 0x97, 0x25, 0x56, 0xB5, 0x01, 0x3C, 0x81}  //endereço para a leitura do r de temperatura de insuflamento

#define TEMP_EXT {0x28, 0x4B, 0x47, 0x56, 0xB5, 0x01, 0x3C, 0xAA}  //endereço para a leitura do r de temperatura de externa

#define TEMP_LL {0x28, 0xB9, 0x1B, 0x56, 0xB5, 0x01, 0x3C, 0x02}

#define TEMP_SUC {0x28, 0xDC, 0x20, 0x56, 0xB5, 0x01, 0x3C, 0xDE}  //endereço para a leitura do sensor de temperatura de sucção
#endif
```

23716

```
#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[] = "GSC";
constexpr char PASSW[] = "#2021alta!";

#define MDASH_APP_NAME "SEDE_23716"    //nome de usuario do madash para esse equipamento
```

```
#define DEVICE_PASSWORD "cBBKNv90KmemEUL90bgODi2Q"  //senha do mdash

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[] = "23716";
    constexpr char TOPIC[] = "v4/matr0017";
    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 =  50;
    static constexpr float CAL_S = 84.7;
    static constexpr float CAL_TT = 82.3;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 5.7;
    static constexpr int CAL_S = 13.75;
    static constexpr int CAL_TT = 15.9;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 39;
    static constexpr int CURRENT_S = 35;
```

```
    static constexpr int CURRENT_TT = 33;
    static constexpr int TEMPERATURE = 14;
    static constexpr int VOLTAGE_R = 34;
    static constexpr int VOLTAGE_S = 36;
    static constexpr int VOLTAGE_TT = 32;
};

// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s1037";
    constexpr char RET[] = "s1038";
    constexpr char SUC[] = "s1039";
    constexpr char LL[] = "s1040";
    constexpr char ENT_CONDES[] = "s1041";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s1042";
    constexpr char VOLT_S[] = "xxxx";
    constexpr char VOLT_TT[] = "xxxx";

    constexpr char CURR_R[] = "s1043";
    constexpr char CURR_S[] = "xxx";
    constexpr char CURR_TT[] = "xxx";

    constexpr char BAT_INS[] = "XXX";
    constexpr char BAT_RET[] = "XXX";
    constexpr char BAT_SUC[] = "XXX";
    constexpr char BAT_LL[] = "XXX";

    constexpr char VIBR_X_SUC[] = "XXX";
    constexpr char VIBR_Y_SUC[] = "XXX";
    constexpr char VIBR_Z_SUC[] = "XXX";

}


#define TEMP_RET {0x28, 0x1E, 0x96, 0x56, 0xB5, 0x01, 0x3C, 0xFC}  //endereço para a leitura do r de temperatura de retorno
```

```
#define TEMP_INSU {0x28, 0xC2, 0x55, 0x56, 0xB5, 0x01, 0x3C, 0x29}  //endereço para a leitura do r de temperatura de insuflamento

#define TEMP_EXT {0x28, 0x79, 0x74, 0x56, 0xB5, 0x01, 0x3C, 0xE3}  //endereço para a leitura do r de temperatura de externa

#define TEMP_LL {0x28, 0xA7, 0x42, 0x56, 0xB5, 0x01, 0x3C, 0x2C}

#define TEMP_SUC {0x28, 0x8F, 0x6A, 0x56, 0xB5, 0x01, 0x3C, 0xFD}  //endereço para a leitura do sensor de temperatura de sucção
#endif
```

23717

```
#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[] = "GSC";
constexpr char PASSW[] = "#2021alta!";

#define MDASH_APP_NAME "SEDE_23717"    //nome de usuario do madash para esse equipamento
```

```
#define DEVICE_PASSWORD "X9950At1n7s4GB5fuP90Wf90A"  //senha do mdash

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[] = "23717";
    constexpr char TOPIC[] = "v4/matr0018";
    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 =  53;
    static constexpr float CAL_S = 84.7;
    static constexpr float CAL_TT = 82.3;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 3;
    static constexpr int CAL_S = 13.75;
    static constexpr int CAL_TT = 15.9;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 33;
    static constexpr int CURRENT_S = 35;
```

```
    static constexpr int CURRENT_TT = 39;
    static constexpr int TEMPERATURE = 4;
    static constexpr int VOLTAGE_R = 32;
    static constexpr int VOLTAGE_S = 36;
    static constexpr int VOLTAGE_TT = 34;
};

// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s1044";
    constexpr char RET[] = "s1045";
    constexpr char SUC[] = "s1046";
    constexpr char LL[] = "s1047";
    constexpr char ENT_CONDES[] = "s1048";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s1049";
    constexpr char VOLT_S[] = "xxxx";
    constexpr char VOLT_TT[] = "xxxx";

    constexpr char CURR_R[] = "s1050";
    constexpr char CURR_S[] = "xxx";
    constexpr char CURR_TT[] = "xxx";

    constexpr char BAT_INS[] = "XXX";
    constexpr char BAT_RET[] = "XXX";
    constexpr char BAT_SUC[] = "XXX";
    constexpr char BAT_LL[] = "XXX";

    constexpr char VIBR_X_SUC[] = "XXX";
    constexpr char VIBR_Y_SUC[] = "XXX";
    constexpr char VIBR_Z_SUC[] = "XXX";

}


#define TEMP_RET {0x28, 0xC5, 0x7A, 0x56, 0xB5, 0x01, 0x3C, 0x3B}  //endereço para a leitura
do r de temperatura de retorno
```

```
#define TEMP_INSU {0x28, 0x5E, 0xB9, 0x56, 0xB5, 0x01, 0x3C, 0x1E}  //endereço para a leitura do r de temperatura de insuflamento

#define TEMP_EXT {0x28, 0x45, 0x5C, 0x56, 0xB5, 0x01, 0x3C, 0xB5}  //endereço para a leitura do r de temperatura de externa

#define TEMP_LL {0x28, 0xF9, 0xD3, 0x56, 0xB5, 0x01, 0x3C, 0x6B}

#define TEMP_SUC {0x28, 0xA5, 0xF7, 0x56, 0xB5, 0x01, 0x3C, 0xDF}  //endereço para a leitura do sensor de temperatura de sucção
#endif
```

30145

```
#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[] = "GSC";
constexpr char PASSW[] = "#2021alta!";

#define MDASH_APP_NAME "SEDE_30145"    //nome de usuario do madash para esse equipamento

#define DEVICE_PASSWORD "oSrsVWSXowuqKQpPJAnvsQ"  //senha do mdash
```

```
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[] = "30145";
    constexpr char TOPIC[] = "v4/matr0068";
    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 =  49.1;
    static constexpr float CAL_S = 84.7;
    static constexpr float CAL_TT = 82.3;
};

// Current Calibration Constants
struct CurrentCalibration
{
    static constexpr int CAL_R = 7.1;
    static constexpr int CAL_S = 13.75;
    static constexpr int CAL_TT = 15.9;
};

// Pin Definitions
struct Pins
{
    static constexpr int CURRENT_R = 39;
    static constexpr int CURRENT_S = 35;
    static constexpr int CURRENT_TT = 33;
    static constexpr int TEMPERATURE = 14;
```

```
    static constexpr int VOLTAGE_R = 34;
    static constexpr int VOLTAGE_S = 36;
    static constexpr int VOLTAGE_TT = 32;
};

// Sensor Addresses
namespace Sensors
{
    constexpr char INS[] = "s3515";
    constexpr char RET[] = "s3514";
    constexpr char SUC[] = "s3519";
    constexpr char LL[] = "s3516";
    constexpr char ENT_CONDES[] = "s3520";
    constexpr char SAD_CONDES[] = "xxxxx";// essa temp não ta sendo enviada, ignorar ela ou
adicionar nos envios se necessario

    constexpr char VOLT_R[] = "s3517";
    constexpr char VOLT_S[] = "xxxx";
    constexpr char VOLT_TT[] = "xxxx";

    constexpr char CURR_R[] = "s3518";
    constexpr char CURR_S[] = "xxx";
    constexpr char CURR_TT[] = "xxx";

    constexpr char BAT_INS[] = "XXX";
    constexpr char BAT_RET[] = "XXX";
    constexpr char BAT_SUC[] = "XXX";
    constexpr char BAT_LL[] = "XXX";

    constexpr char VIBR_X_SUC[] = "XXX";
    constexpr char VIBR_Y_SUC[] = "XXX";
    constexpr char VIBR_Z_SUC[] = "XXX";

}


#define TEMP_RET {0x28, 0x0B, 0x86, 0x95, 0xF0, 0x01, 0x3C, 0x85}  //endereço para a leitura
do r de temperatura de retorno

#define TEMP_INSU {0x28, 0x06, 0xDF, 0x95, 0xF0, 0x01, 0x3C, 0xA9}  //endereço para a leitura
do r de temperatura de insuflamento
```

```
#define TEMP_EXT {0x28, 0x04, 0x8E, 0x95, 0xF0, 0x01, 0x3C, 0x9F}  //endereço para a leitura do r de temperatura de externa

#define TEMP_LL {0x28, 0x46, 0x2E, 0x95, 0xF0, 0x01, 0x3C, 0xB7}

#define TEMP_SUC {0x28, 0x0D, 0x06, 0x95, 0xF0, 0x01, 0x3C, 0xFC}  //endereço para a leitura do sensor de temperatura de sucção
#endif
```