SimHub

Informações Gerais

Link documentação: https://github.com/SHWotever/SimHub/wiki

Compatibilidade com jogos

O SimHub é compatível com uma ampla variedade de jogos de corrida, incluindo títulos populares como Assetto Corsa, iRacing, F1, Forza e outros. Para jogos de console que suportam telemetria via UDP, como Forza Motorsport 7 e F1, é possível configurar o SimHub para receber esses dados através da rede local, substituindo o endereço 127.0.0.1 pelo IP do seu computador e ajustando as configurações de firewall conforme necessário .

Integração com arduino

Uma das funcionalidades mais poderosas do SimHub é a integração com placas Arduino, permitindo a criação de dispositivos personalizados como painéis, indicadores de LED, motores de vibração e muito mais. O processo envolve:

1. **Configuração da Placa**: Utilize a ferramenta de configuração do Arduino fornecida pelo SimHub para selecionar os módulos desejados e gerar o firmware correspondente .GitHub+1GitHub+1

2. **Upload do Firmware**: Carregue o firmware gerado na placa Arduino usando a IDE do Arduino.GitHub+3GitHub+3GitHub+3

3. **Adição no SimHub**: No SimHub, adicione o dispositivo e configure os parâmetros necessários, como PID/VID, para que o software reconheça automaticamente a porta serial .GitHub

Dash Studio

O Dash Studio é uma ferramenta integrada ao SimHub que permite criar e editar dashboards personalizados para exibir informações em tempo real durante as corridas. Você pode:GitHub+1GitHub+1

- **Criar Novos Dashboards**: Inicie um novo projeto e personalize-o conforme suas preferências .GitHub

- **Editar Componentes**: Adicione e configure widgets como indicadores de velocidade, RPM, marchas, entre outros .GitHub+1GitHub+1

- **Ajustar Desempenho**: Otimize a performance dos dashboards para garantir que não haja impacto negativo na experiência de jogo .GitHub

Feedback tátil e simulação de vento

O módulo "ShakeIt" do SimHub permite adicionar feedback tátil ao seu setup, utilizando motores de vibração e ventiladores para simular sensações como trepidações e fluxo de ar. Para configurar:GitHub

1. **Ativar o Plugin**: No SimHub, vá até as configurações de plugins e habilite o "ShakeIt".

2. **Configurar o Arduino**: Utilize a ferramenta de configuração do Arduino para habilitar os motores e ventiladores desejados.GitHub

3. **Testar os Dispositivos**: Após a configuração, teste os motores e ventiladores para garantir que estão funcionando corretamente .GitHub+10GitHub+10

Informações do sistema

O SimHub pode exibir informações detalhadas do sistema, como uso de CPU, GPU, temperatura e FPS, integrando dados de ferramentas como AIDA64 e HWiNFO. Para configurar:GitHub

1. **Ativar Provedores**: Nas configurações do SimHub, vá até "System Infos" e habilite os provedores desejados.GitHub

2. **Selecionar Métricas**: Escolha quais métricas deseja monitorar e associe-as aos provedores correspondentes .GitHub

Dispositivos

- Displays
    - TM1638 8 Digits 7 segment display with optionnal leds
        - Led editor guide

        - TM1637 4 Digits 7 segment display
        - TM1637 6 Digits 7 segment display (RobotDyn)
        - MAX7219/MAX7221 8 Digits 7 segment display
- RGB LEDS
    - WS2812B RGB Leds
        - Led editor guide
    - PL9823 smart leds
- 8x8 Single color LED Matrix
    - MAX7219/MAX7221 7 segment matrix
- RGB 8x8 Matrix :
    - DM163/Iflag/sunfounder 8x8 RGB matrix shield
    - WS2812B 8x8 RGB Matrix
    - How to configure RGB matrix rotation
    - How to configure separate RGB matrix content
- Text LCD or graphic LCD
    - 20x4 or 16x2 LCD wiring and configuration
    - Arduino SSD1306 0.96" Oled I2C
- Inputs
    - Press buttons
    - Button matrix
    - Rotary encoders
- Tactile Feedback and Wind Simulation
    - ShakeIt for vibration motors
    - ShakeIt for wind simulation
- Gauges
    - After Market Tach Gauge
    - After Market Boost Gauge
    - After Market Speedo
    - BMW E36 Water temperature, Fuel gauge
- Custom hardware support
    - Custom Arduino hardware support
- Simple LEDS ?
    - Why simple LEDs won't be supported

Display OLED com dados de RPM e Velocidade via SimHub e Arduino

🎯 Objetivo:

Montar um pequeno display (usando uma tela OLED e Arduino) que mostre em tempo real o RPM e a velocidade do carro durante uma corrida em um jogo como *Assetto Corsa*, *F1 23* ou *iRacing*.

🛠️ Materiais:

- Arduino Pro Micro (ou Uno, Nano etc.)
- Tela OLED (por exemplo, SSD1306, 128x64, via I2C)
- Cabos jumper
- PC com SimHub instalado
- Jogo compatível com SimHub

🔧 Etapas de configuração

1. Conectar a Tela OLED ao Arduino

- VCC → 5V ou 3.3V (depende do seu Arduino)
- GND → GND
- SDA → Pino 2 (pode variar conforme o sketch)
- SCL → Pino 3 (idem)

---

2. Configurar o Arduino no SimHub

- Abra o SimHub > Vá em **Arduino** > **My Hardware**
- Clique em **"Show Arduino Setup Tool"**
- Escolha o tipo de display: **“OLED Display - I2C”**
- Selecione as informações a serem exibidas: por exemplo, **[Speed]** e **[RPM]**
- Clique em **"Generate Sketch Files"**
- O SimHub irá gerar o código (firmware)

---

3. Carregar o código no Arduino

- Abra o Arduino IDE

- Cole o código gerado ou abra o arquivo `.ino` que o SimHub criou
- Selecione a placa e porta correta (Ferramentas > Placa/Porta)
- Clique em **"Upload"**

---

4. Testar com o jogo

- Inicie o jogo compatível (ex: Assetto Corsa)
- O SimHub detectará automaticamente e começará a enviar os dados para o Arduino
- A tela OLED irá exibir os dados em tempo real: velocidade (km/h), rotação do motor (RPM), marcha, etc.

Pré-requisitos (antes de carregar o código):

1. Instale a biblioteca **U8g2** na IDE do Arduino:
    - Vá em: **Sketch > Incluir Biblioteca > Gerenciar Bibliotecas**
    - Procure por **"U8g2"** e instale
2. Verifique se sua tela usa o controlador **SSD1306 128x64 I2C**

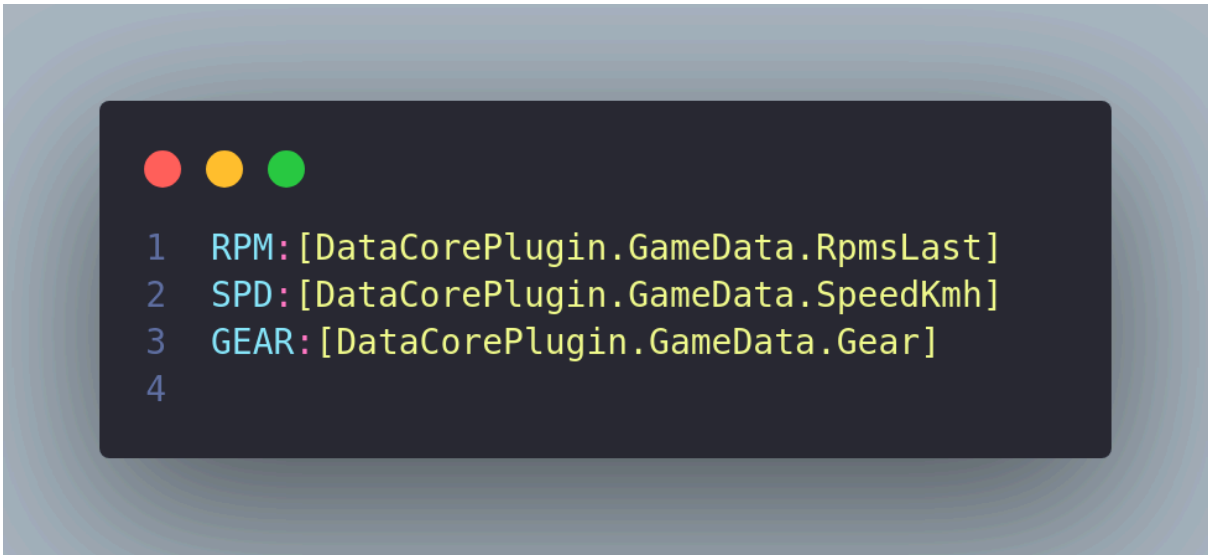
💻 Código Arduino:

```ino
#include <U8g2lib.h>
#include <Wire.h>

// Display SSD1306 128x64 I2C - Pode ajustar para sua tela
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0);

// Variáveis para receber dados do SimHub
int rpm = 0;
int speed = 0;
int gear = 0;

void setup() {
  Wire.begin();
  u8g2.begin();

  Serial.begin(115200);
  Serial.println("SimHub Arduino Display");
}

void loop() {
  // Lê os dados vindos do SimHub via porta serial
  if (Serial.available()) {
    String data = Serial.readStringUntil('\n');

    if (data.startsWith("RPM:")) {
      rpm = data.substring(4).toInt();
    } else if (data.startsWith("SPD:")) {
      speed = data.substring(4).toInt();
    } else if (data.startsWith("GEAR:")) {
      gear = data.substring(5).toInt();
    }
  }

  // Atualiza o display
  u8g2.clearBuffer();

  u8g2.setFont(u8g2_font_ncenB08_tr);
  u8g2.drawStr(0, 12, "== SimHub Display ==");

  char buffer[20];
  sprintf(buffer, "Velocidade: = km/h", speed);
  u8g2.drawStr(0, 28, buffer);

  sprintf(buffer, "RPM:       M", rpm);
  u8g2.drawStr(0, 44, buffer);

  sprintf(buffer, "Marcha:    %d", gear);
  u8g2.drawStr(0, 60, buffer);

  u8g2.sendBuffer();

  delay(100);
}
```

Arquivo .cpp

⚙️ Como enviar os dados do SimHub

Você pode configurar no SimHub para enviar os dados via serial como:



Arquivo Makefile

Isso é feito em:

- **Arduino > Serial Output > Custom Serial Output**
- Ative e insira a saída serial acima, formatando como mostrado.

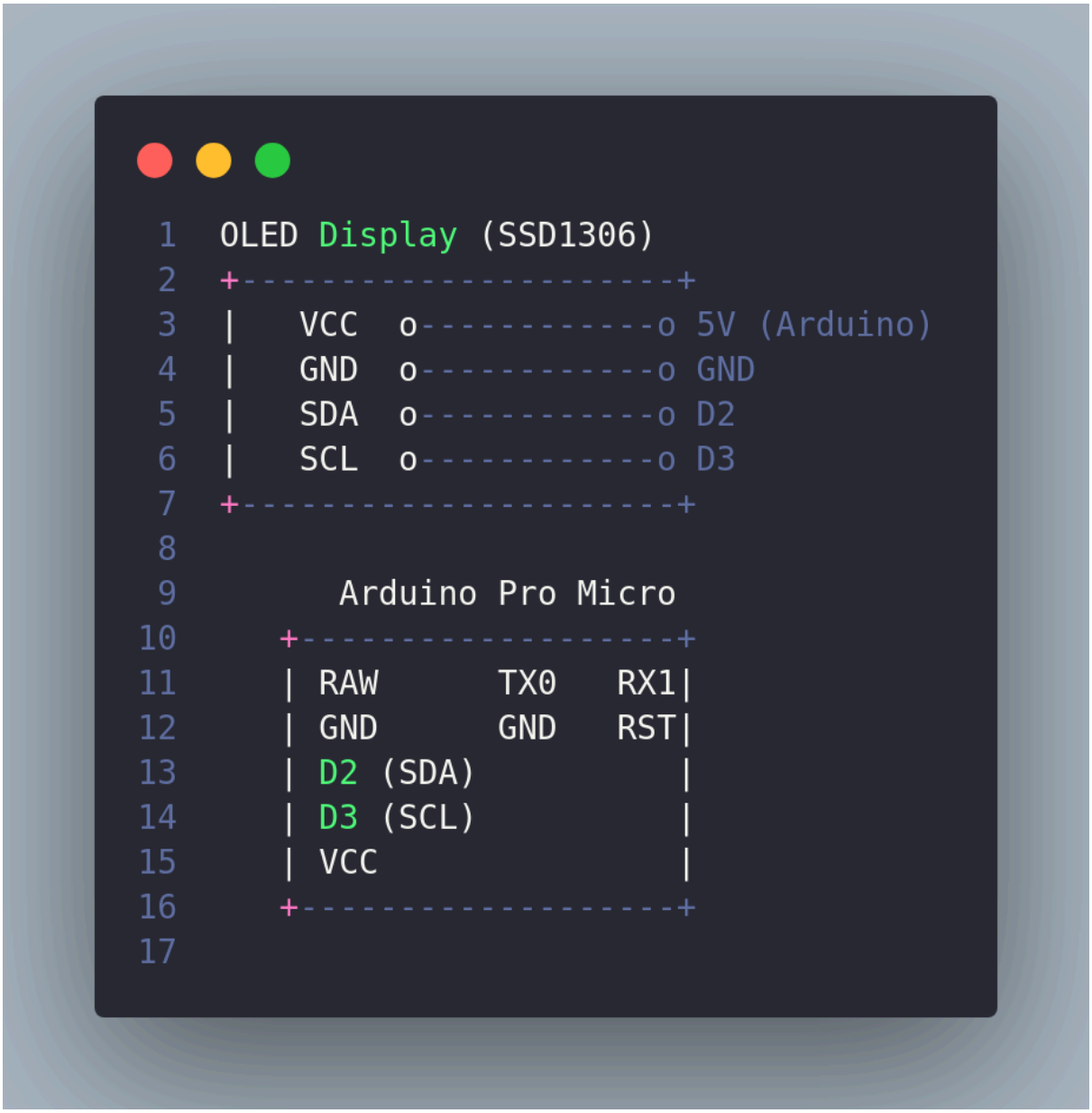
🔌 Ligações entre o Arduino e o Display OLED:

OLED Pin	Conecta ao Arduino	Função
VCC	VCC (5V ou 3.3V)	Alimentação
GND	GND	Terra
SDA	Pino 2	Comunicação I2C (dados)
SCL	Pino 3	Comunicação I2C (clock)

> **Observação**: o código acima usa `Wire.begin()` padrão. Em alguns Arduinos, os pinos SDA/SCL podem variar:

- **Uno/Nano**: SDA = A4, SCL = A5
- **Mega**: SDA = 20, SCL = 21
- **Pro Micro**: SDA = 2, SCL = 3 (como usado no código)

📷 Diagrama (ASCII-style para referência rápida):



Arquivo .lua

📘 Alimentação:

- Se estiver usando o Arduino conectado via USB ao PC (recomendado com o SimHub), **não precisa de alimentação externa**.

🛠️ Dica para testes:

Se o display OLED não ligar:

- Verifique se ele é 3.3V ou 5V (alguns só funcionam com 3.3V)
- Troque os pinos SDA/SCL conforme seu modelo de Arduino
- Verifique o endereço I2C com o scanner como este aqui

Referências

Criando uma segunda tela com Dash:
https://www.youtube.com/watch?v=DhHThWRhGP4
Resumo: No vídeo é apresentado como utilizar o SimHub com uma segunda tela. É utilizado uma tela principal ligada em um computador onde foi instalado o jogo F1 e o SimHub.Na segunda tela é instalado apenas o SimHub. Após isso, é necessário acessar o Dash Studio no SimHub e escolher um modelo com informações de RPM, etc. Ao clicar será disponibilizado um IP que deverá ser colocado no Dashboard selecionado na segunda tela. Após selecionar o Dashboard, é necessário abrir o jogo na tela principal, abrir configurações do jogo e opções de telemetria, habilitar a telemetria por UDP e colocar o IP que tinha sido gerado no SimHub.