

**Implementation of a Free and Lightweight HMI Solution for Embedded Systems**

**MANGLESH PATIDAR**

**TVS Motor Company Ltd.**

**Co-coordinated by:**

###### Mr. Nabakrushna Pradhan

*Sr. Engineering Manager*

*SW & Connected Systems CoC,*

*TVS Motor Company*

###### Mr. Vamsi Krishna Garla

*Lead SW Engineer*

*SW & Connected Systems CoC,*

*TVS Motor Company*

**Sponsored by:**

*Mr. Parag Ashok Inamdar*

*Head, SW & Connected Systems CoC,*

*TVS Motor Company*

**Acknowledgement**

I would like to express my sincere gratitude to all those who supported me throughout the completion of this project.

First and foremost, I am thankful to my project guide **Mr. Vamsi Krishna Garla** whose guidance, encouragement, and constructive feedback were invaluable throughout the development process. I am also grateful to **TVS MOTOR COMPANY Ltd** for providing the necessary infrastructure and support for carrying out this work.

A special thanks to the open-source community behind **LVGL (Light and Versatile Graphics Library)** for offering such a powerful and accessible HMI development framework. Their continuous efforts in maintaining comprehensive documentation and tools such as the LVGL UI Editor made this project possible.

Finally, I would like to thank Mr. Vamsi Krishna for his encouragement, patience, and support during this project.

2025

1 Year, M.Tech.

Department of Computer

Science & Engineering.

IIT Madras.

**CONTENTS**

ABSTRACT...................................................................................................................4

PROBLEM STATEMENT ............................................................................................5

TOP 10 HMI TOOLS …………………...…................................................................6

WHY LVGL CHOSEN? ............................................................................................. 7

LVGL UI EDITOR & INSTALLATION.....................................................................8

TYPES OF HARDWARE LVGL CAN RUN ON........................................................10

LVGL PC SIMULATOR SETUP (VS 2022) ................................................................11

ARCHITECUTRE OVERVIEW……………………………………………........……12

LVGL UI EDITOR DEMO ...........................................................................................14

VISUAL STUDIO 2022 SIMULATOR .........................................................................15

VS CONFIGURATION.............................................................................................17

DEMO PROJECT........................................................................................................19

EBIKE RUN ON PC SIMULATOR………………………………………………...21

FIGMA TO LVGL EDITOR...............................................………………………….22

PROJECT TVS HMI EBIKE ………………………………………………........…...23

FUTURE WORK & NEXT STEPS ...............................................................................24

REFERENCES.............................................................................................................25

**Abstract**

Human-Machine Interfaces (HMIs) are essential in modern embedded systems, often developed using frameworks like Qt, Kanzi, CGI Studio, EB Guide etc. While these tools offer robust features and cross-platform support, its commercial licensing model impose cost and legal limitations. This report explores the need for an open-source, cross-platform HMI development tool that supports C/C++ and offers modern UI capabilities without licensing fees. LVGL (Light and Versatile Graphics Library) is investigated as a viable alternative for creating cost-effective and scalable HMI solutions.

**Problem Statement**

Many widely used Human-Machine Interface (HMI) development tools offer rich features and cross-platform support, making them attractive choices for building interactive user interfaces in embedded systems. However, most of these tools follow a commercial licensing model that can introduce significant costs for proprietary or commercial use. Additionally, some licensing terms (e.g., GPL/LGPL) may require developers to share their source code, posing legal and operational challenges. These constraints create barriers for individual developers, startups, and small organizations who seek to develop custom, proprietary HMIs without incurring licensing fees or exposing their codebase. There is a strong need for a free, open-source, and license-friendly HMI solution that supports C/C++, runs cross-platform, and is suitable for both prototyping and production

There is a need for a free, open-source HMI framework that:

* Supports cross-platform development (Windows, Linux, MCUs/MPUs)
* Works with C and C++, common in embedded systems
* Does not mandate open-source code disclosure
* Offers modern, customizable UI capabilities with a small memory footprint

TOP 10 HMI TOOLS  
(Open Source)

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | License | Features | Cons |
| **1. LVGL (Light and Versatile Graphics Library)** | MIT | - Very lightweight- Designed for embedded systems- Excellent performance on MCUs- C++ compatible | - Lacks advanced widgets like Qt- No official GUI designer tool |
| **2. Flutter with Embedder API (C++ engine)** | BSD | - High-quality UI- Skia rendering engine- Can embed with custom C++ runtimes | - Heavyweight for small devices- Complex build system |
| **3. µGFX** | Custom Permissive (like MIT) | - Designed for resource-constrained systems- C/C++ compatible- Modular | - Not as active as others- Commercial support better than community |
| **4. Crank Storyboard (Community Edition)** | GPL (with permissive evaluation) | - Purpose-built for automotive HMIs- Great animation and embedded support | - Community version limited- Full version is commercial |
| **5. Nuklear** | Public Domain / MIT | - Single header GUI- Extremely lightweight- C-based (C++ friendly) | - Basic look- No layout system |
| **6. Dear ImGui** | MIT | - Fast for prototyping- Highly efficient- Easy to embed in C++ | - Meant for dev tools, not consumer HMIs- Minimal animation/custom style |
| **7. LittlevGL (LVGL’s predecessor)** | MIT | - Similar benefits to LVGL- Works on 32-bit MCUs- Easier for small EV dashboards | - Obsolete compared to LVGL |
| **8. Embedded Wizard (FreeRTOS / open parts)** | Free Edition with permissive parts | - Great for animations and modern HMI- Supports C++ indirectly | - Free edition is limited- Mostly commercial |
| **9. EFL (Enlightenment Foundation Libraries)** | BSD 2-Clause | - Efficient rendering- Used in Tizen (Samsung)- Supports embedded use | - Older documentation- C-focused, less modern UI feel |
| **10. CEGUI (Crazy Eddie's GUI)** | MIT | - Full-featured GUI system- Skinnable and scriptable- Used in games and simulations | - Complex setup- Larger footprint than LVGL |

Why LVGL is Chosen

LVGL (Light and Versatile Graphics Library) is an open-source embedded GUI library that provides a modern, customizable, and resource-efficient solution for HMI development. It is designed specifically for microcontrollers and low-resource environments, making it ideal for embedded systems. LVGL is selected for this project due to its flexibility, free licensing model, and robust community support, especially for applications where performance, cost-efficiency, and source code privacy are essential.

Advantages of LVGL

* **Open Source and Free for Commercial Use**: Licensed under MIT, LVGL allows full commercial usage without licensing fees or code disclosure.
* **Cross-Platform Support**: Can run on Windows, Linux, RTOS, and bare-metal MCUs/MPUs.
* **Lightweight and Efficient**: Optimized for memory- and CPU-constrained embedded systems.
* **Modular and Customizable**: Developers can extend or remove components as needed.
* **Supports C and C++**: Ideal for embedded applications traditionally written in these languages.
* **Runtime XML Support**: Enables dynamic UI updates without firmware reflashing.
* **Figma Integration and UI Editor**: Speeds up UI design and collaboration between developers and designers.
* **Strong Community and Documentation**: Active development, regular updates, and detailed documentation.

Disadvantages of LVGL

* **No Native Drag-and-Drop Editor for Beginners**: Requires some familiarity with XML and C/C++ to use effectively.
* **Limited Desktop Application Use**: Primarily designed for embedded targets, not full-scale desktop applications.
* **Steeper Learning Curve for Complex UIs**: Advanced animations or complex effects may require manual handling.
* **Fewer Out-of-the-Box Components Compared to Qt**: Developers may need to build more custom widgets.

## **LVGL UI Editor and Installation Process**

The **LVGL UI Editor** is a visual design tool used to create Human-Machine Interfaces (HMI) using the LVGL graphics library. It simplifies the process of designing graphical components, editing layouts, managing fonts and images, and integrating with C/C++ application logic. The editor generates LVGL-compliant XML and C code, making it suitable for embedded GUI development.

Step-by-Step Installation Guide: (Approach 1)

1. Install Prerequisites

* **Install Docker** (Required for code compilation and integration within the editor)
  + Download Docker from: <https://www.docker.com/products/docker-desktop>
  + On **Windows**, make sure **Docker Desktop (daemon) is running**

2. Set Up the LVGL UI Editor

* **Download the Editor**: https://github.com/lvgl/lvgl\_editor
* **Clone the Repository** using Git: git clone https://github.com/lvgl/lvgl\_editor.git

3. Load and Prepare the Project

* Open the **LVGL UI Editor**
* Load the sample project by selecting the example/ folder
* Click on **Generate Code** button.
* Then click on **Compile Project** to refresh the editor preview with the generated C code

4. Edit UI Elements

* **Edit Components**:
  + Open components/button\_default.xml
  + Modify as needed and press Ctrl+S to auto-update the preview
* **Edit Widgets**:
  + Open widgets/slider\_box.xml
  + Click **Compile Code** to regenerate both XML and C logic
* **Manage Fonts and Images**:
  + Open globals.xml to add, remove, or update fonts and image assets.

5. Integrate with Figma (**Figma to LVGL Plugin)**: Export styles from Figma directly to XML:

* + Update your components with auto-generated styles for design consistency

Step-by-Step Installation Guide: (Approach 2) Dt:16-07-25

**1. Install Dependencies**

To enable full functionality (including code compilation and live preview), the following tools must be installed:

* **Windows**:  
   Open Command Prompt and run: wsl –install
* **Linux**:  
   Install Podman using: sudo apt-get install podman
* **macOS**:  
   No additional installation is required; works out of the box.

2. Set Up the LVGL UI Editor

* **Download the Editor**: https://github.com/lvgl/lvgl\_editor
* **Clone the Repository** using Git: git clone https://github.com/lvgl/lvgl\_editor.git

3. Load and Prepare the Project

* Open the **LVGL UI Editor**
* Load the sample project by selecting the example/ folder
* Click on **Generate Code** button.
* Then click on **Compile Project** to refresh the editor preview with the generated C code

4. Edit UI Elements

* **Edit Components**:
  + Open components/button\_default.xml
  + Modify as needed and press Ctrl+S to auto-update the preview
* **Edit Widgets**:
  + Open widgets/slider\_box.xml
  + Click **Compile Code** to regenerate both XML and C logic
* **Manage Fonts and Images**:
  + Open globals.xml to add, remove, or update fonts and image assets.

5. Integrate with Figma (**Figma to LVGL Plugin)**:

* + Export styles from Figma directly to XML:
  + Update your components with auto-generated styles for design consistency

Types of Hardware LVGL Can Run On:

#### **1. Microcontrollers (MCUs)**

* **STM32** (STMicroelectronics)
* **ESP32 / ESP8266** (Espressif)
* **NXP i.MX RT** series
* **GD32** (GigaDevice)
* **Renesas RA and RX series**
* **Microchip SAM series**
* **TI MSP432 / Tiva-C**

#### **2. Microprocessors (MPUs)**

* **NXP i.MX6 / i.MX8**
* **Allwinner A33, A64**
* **Rockchip RK3399**
* **Raspberry Pi (as an MPU platform)**

#### **3. Single-Board Computers (SBCs)**

* **Raspberry Pi (Zero, 3, 4, 5)**
* **BeagleBone Black**
* **Orange Pi**
* **NanoPi**

#### **4. PC/Desktops (Simulation/Development)**

* **Windows**
* **Linux**
* **macOS**  
   *(Used with SDL2 for simulation and testing without hardware)*

#### **5. RTOS / OS Platforms**

* Bare-metal (no OS)
* **FreeRTOS**
* **Zephyr**
* **RT-Thread**
* **Linux framebuffer**
* **Little Kernel / U-Boot for bootloader UI**

LVGL PC Simulator Setup (Windows + Visual Studio 2022)

The **LVGL PC Simulator** allows developers to run and test graphical user interfaces directly on a personal computer without the need for physical hardware. This environment is particularly useful during the early development and debugging phases, enabling fast iteration and issue tracking.

Step-by-Step Setup Guide:

1. Install Visual Studio 2022

* Download Visual Studio: https://visualstudio.microsoft.com/downloads/
* During installation, select:
  + **"Desktop development with C++"** workload

2. Download LVGL Visual Studio Simulator

* Clone: git clone <https://github.com/lvgl/lv_port_pc_visual_studio.git>

3. Install SDL2 Development Libraries (Download SDL2 Not SDL3+):

* Download: https://github.com/libsdl-org/SDL/releases/tag/release-2.x.x  
   **SDL2-devel-2.0.x-VC.zip** *(VC = Visual C++ version for Visual Studio)*

4. Configure SDL2 in Visual Studio

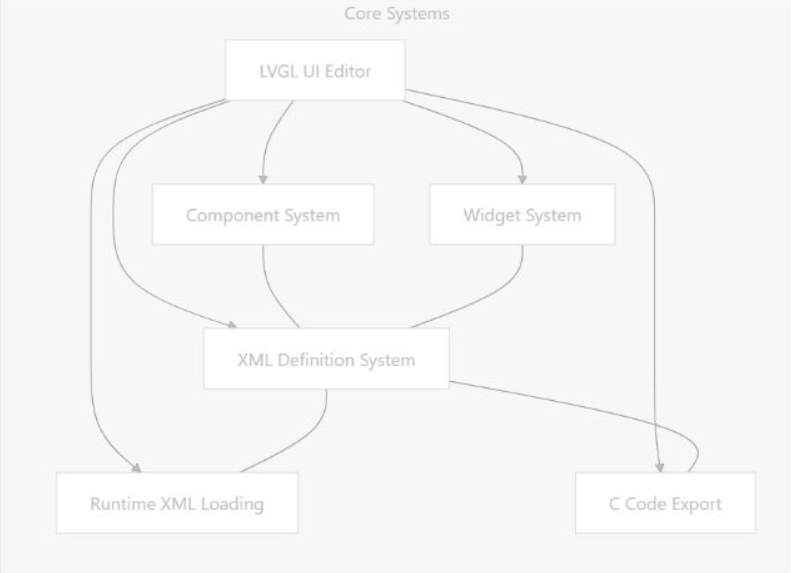
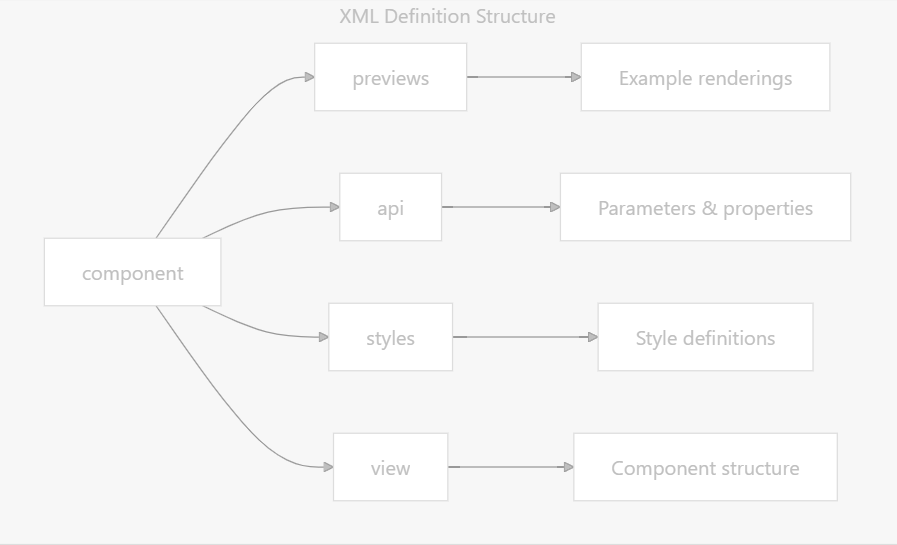
* In the **LVGL project**, go to:
  + **Project ➝ Properties ➝ VC++ Directories**
    - **Include Directories**: Add the SDL2/include path and add your project path
    - **Library Directories**: Add the SDL2/lib/x64 path and also your project path
* Then go to:
  + **Linker ➝ Input ➝ Additional Dependencies**  
     Add: SDL2.lib and SDL2main.lib
* Finally, copy SDL2.dll from SDL2/lib/x64/ into your project’s build output folder (e.g., Debug/)

5. Build and Run

* Click **Build ➝ Build Solution** (or press Ctrl+Shift+B)
* After a successful build, press **F5** to run
* A **simulated LVGL UI window** should appear

**ARCHITECTURE OVERVIEW:**

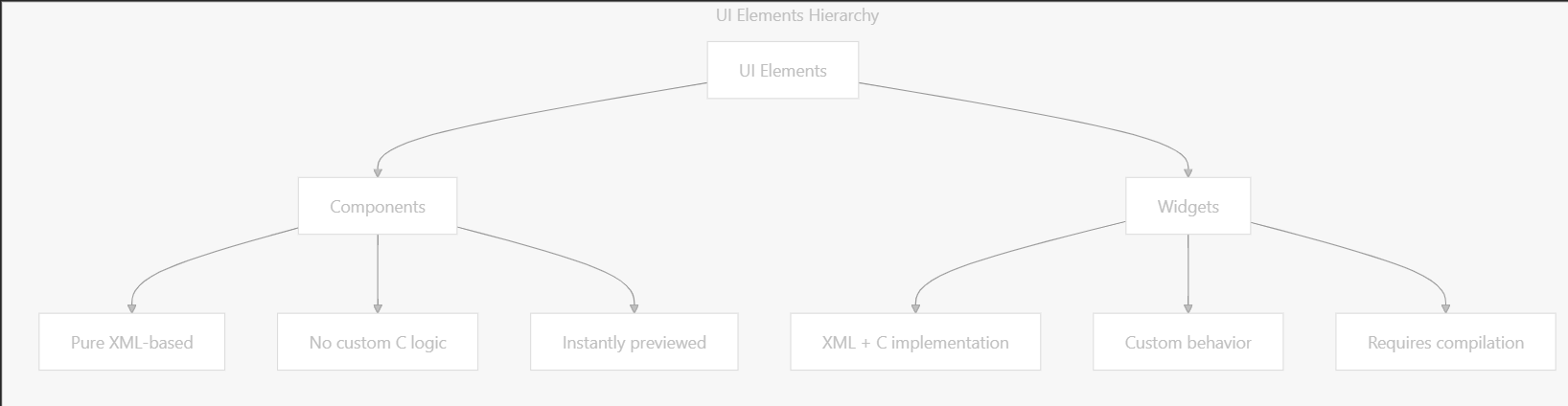
**Design: High-Level Architecture**



**Code Generation Pipeline:**

A screenshot of a diagram

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.



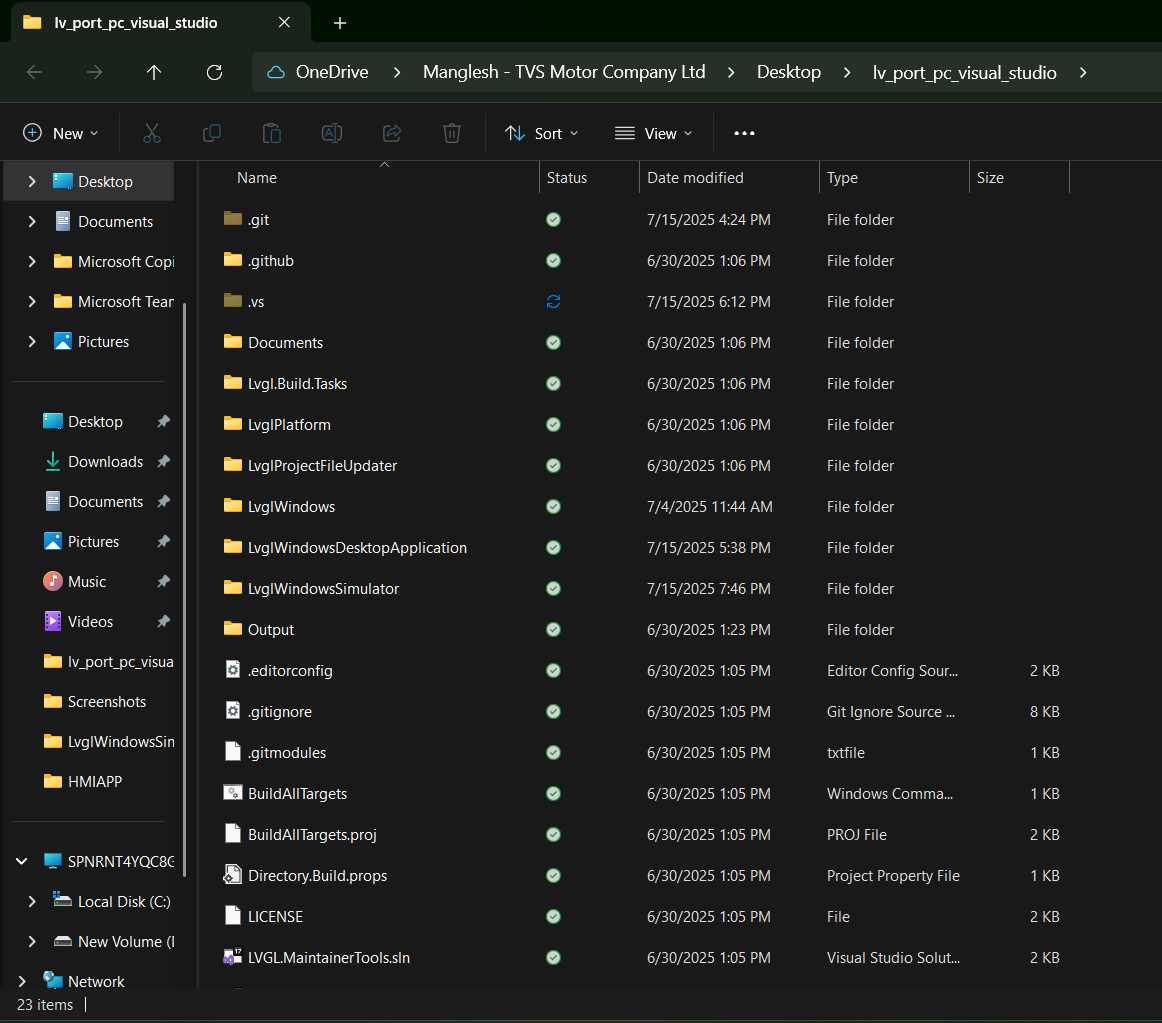
LVGL UI EDITOR DEMO

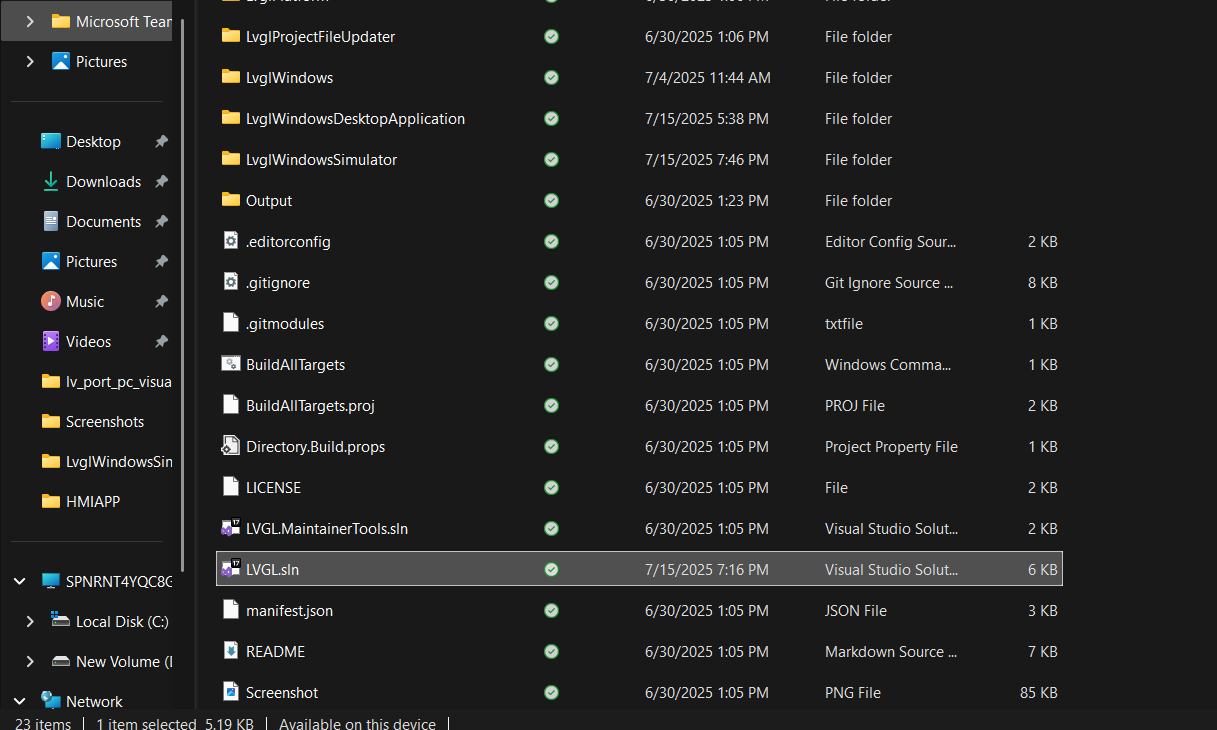
A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**VISUAL STUDIO 2022 (SIMULATOR)**





A screenshot of a computer program

AI-generated content may be incorrect., Picture

Put your Project inside LVGLWIndowsSImulator folder after successfully build YOur Project on LVGL Editor..

To Run On VS 2022, Your Project Click on LVGL.sln (above shown)  
Then In Solution Explorer Right Click LvglWindowsSimulator ->Add->Existing Project :  
 Select all the neccessary files (all .c , .h, .xml , .ttf , .png ) not folder bring all files any type it here...  
Modify the lv\_conf.h where you need to set 1 according to your project requirement ...

Before Build (Ctrl + shift + B) you need to Configuration in VS code

In Soltuion Explorer Right Click LvglWindowsSimulator->Properties

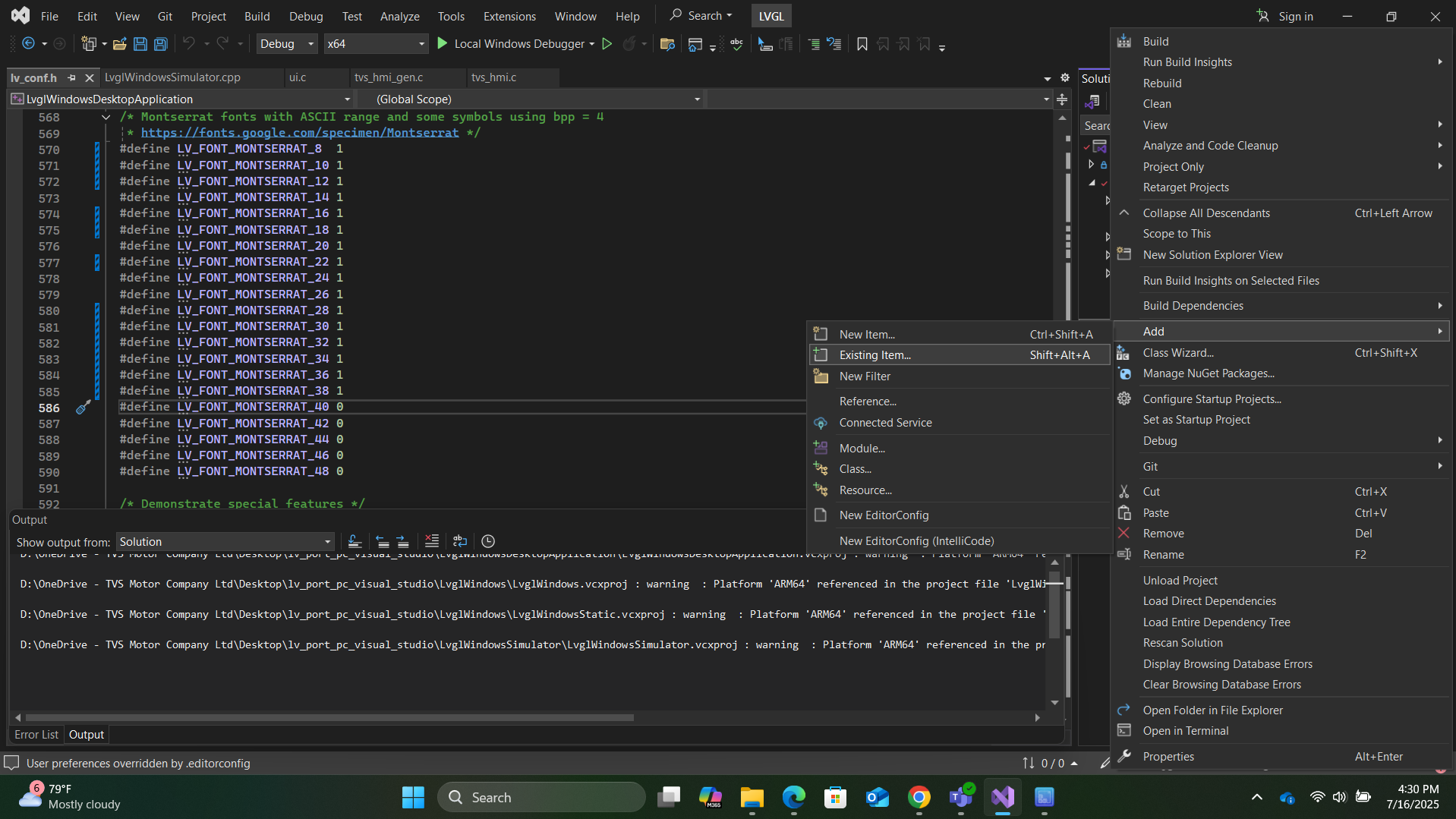
VC++ Directories --> Include Directories -> Set a project Path

Library Directories --> Set a Project Path

C/C++--> Additional Include Directories --> Set a Project Path

Linker--> Output File --> Set a Project Path , After Apply then Ok

**VS Configuration (Set a Path)**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

DEMO PROJECT A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

EBIKE RUN ON PC SIMULATOR

A screenshot of a computer

AI-generated content may be incorrect.

FIGMA--> LVGL EDITOR

FIGMA:

A screen shot of a device

AI-generated content may be incorrect.

LVGL EDITOR:

A screenshot of a video player

AI-generated content may be incorrect.

PROJECT TVS\_HMI\_EBIKE

A screenshot of a computer

AI-generated content may be incorrect.

### **Future Work and Next Steps**

Due to time constraints, this project was limited to developing and simulating the HMI using the LVGL UI Editor and running it on a PC-based simulator. Given additional time, the next critical milestone would be completing the TVS\_HMI\_EBIKE , porting and deploying the LVGL-based HMI onto actual embedded hardware. This step involves integrating the UI with real-world inputs such as sensors, buttons, and communication protocols used in electric two-wheelers.

Key hardware platforms that can be used to run the HMI include:

* **NXP i.MX RT1050/1060 or RT1170** (high-performance crossover MCUs with display support)
* **STM32F7 / STM32H7 series** (widely used in automotive and HMI applications)
* **ESP32-S3** (low-cost, Wi-Fi/Bluetooth-enabled MCU with LCD support)
* **Raspberry Pi 4 or CM4** (Linux-based option for rapid prototyping)
* **RK3566 / RK3399 SoCs** (for Android/Linux-based rich HMI systems)

The future scope includes interfacing with sensors like speed, battery level, GPS modules, and Bluetooth; optimizing the UI for real-time performance; and deploying the solution on a touch display for a fully functional EV bike dashboard.

REFERENCES

* <https://lvgl.io/>
* <https://lvgl.io/demos>
* <https://docs.lvgl.io/master/intro/introduction/index.html>
* <https://github.com/lvgl>
* <https://github.com/lvgl/lvgl>
* <https://github.com/lvgl/lv_port_pc_visual_studio>
* [XML Structure and Syntax | lvgl/lvgl\_editor | DeepWiki](https://deepwiki.com/lvgl/lvgl_editor/4.1-xml-structure-and-syntax)
* <https://www.youtube.com/watch?v=gCxBAK9EByA>
* [Simulator on PC — LVGL documentation](https://docs.lvgl.io/8.2/get-started/pc-simulator.html)
* https://github.com/libsdl-org/SDL