Web HMI – A Low-Cost, Modern and Remotely Accessible User Interface for Instruments

www.tismotech.com

Presented at Pittcon 2019, Philadelphia

Anna Mammen, VP of Engineering, presenting
Web HMI: The Winds of Change, at Pittcon 2019, Philadelphia

Introduction

> *"Any application that can be written in JavaScript,*
> *will eventually be written in JavaScript."*
>
> *—Jeff Atwood, Author, Entrepreneur, Cofounder of StackOverflow*

Today's analytical instrument users need fast, responsive, contemporary Human Machine Interfaces (HMI). There is a growing expectation that smartphones, tablets can be used to configure, control and remotely monitor these instruments.

Developing HMI for the instrument and separate apps for each of the different smartphone platforms and desktops while ensuring a consistent user experience is an expensive proposition. There is a need for a low-cost solution that provides rich, intuitive Graphical User Interface (GUI) and ubiquitous connectivity to user devices (mobile/tablet/desktop). Web HMI systems fill this void, easing mobility, monitoring and control of instruments.

This study examines why Web HMI is rapidly gaining grounds within the analytical instrument industry, discusses the technological and economic merits and presents empirical performance data.

Definition

Web HMI refers to a Human Machine Interface (HMI) built using web technologies such as HTML5, CSS and JavaScript.

At a high level, Web HMI can be described as a web app that displays the GUI within any web browser on the local LCD panel and on user devices (mobile/tablet/desktop).

This is achieved through a common code base that is responsive across different user devices.

Architecture

Figure 1: Typical architecture of a Web HMI implementation

The microcontroller on the analytical instrument does the real time control and data acquisition from the underlying hardware. The microcontroller communicates with the application logic of the Web HMI through Ethernet, USB, Modbus, OPC-UA or any serial communication protocol.

The application logic implements functionalities of the instrument, such as data acquisition, calibration and diagnostics. The GUI communicates with the application logic through Web APIs. The GUI displayed on the local LCD is same as the one displayed on any browser on user devices (mobile/tablet/desktop).

The user devices utilize Web APIs to access the instrument over Wi-Fi, Bluetooth, Ethernet. The Web HMI can be hosted on Linux/Windows® on Industrial PC, custom hardware or low-cost single board computer (SBC) such as Raspberry Pi, inside the analytical instrument.

Features

Web standards are open, non-proprietary and defined by the World Wide Web Consortium. These standards are keeping pace with the changing times and will maintain forward compatibility. There is a large choice of higher-level frameworks and tools available, most of them being open source with permissive licensing terms. Additionally, sophisticated, contemporary GUIs can be more easily developed with web technologies than with traditional ones.

An amazing user interface is critical to delivering excellent user experience. This manifests as easy to use, inviting screens with a consistent theme across screens. This can include graphics, animations, uniform layout and fluid transitions between screens. Web HMI opens the door to a GUI that is visually captivating and intuitive.

Web HMI can be accessed via a browser from any user device (mobile/tablet/desktop) and runs the same code base across different mobile and desktop platforms.

Web HMI is obsolescence-proof as web technologies are here to stay. Further, Web HMI is Operating System agnostic, and hence it is easy to port to a different underlying Operating System or hardware. This also makes it easy to accommodate future higher performance requirements, by swapping in a higher performance hardware platform, without any accompanying changes to the application.

Custom client apps can be developed using the Web APIs, to monitor, configure and control the instrument. These apps can be developed using any programming language and tools like LabVIEW™, MATLAB® etc

Cloud connectivity can be implemented using the Web APIs, to remotely monitor instrument clusters and centralize data storage. The Web HMI can be deployed on-premise or on a remote cloud.

Advantages

- Single, Rich GUI Across User Devices

Web HMIs are accessed via browser, with a single code base delivering a responsive, intuitive and rich GUI that can be used across multiple user devices (mobile/tablet/desktop) and Operating Systems.

- Open web standards, licensing

Open web standards are standardised across the industry and not vendor specific. They are free, versatile and robust. Thus, Web HMI can be built with zero licensing cost

- Hardware platform

The CPU/memory resources depend on the HMI requirements. Web HMI requires higher CPU and memory resources but the availability of low-cost hardware options like Raspberry Pi make Web HMI more affordable. It is easy to transition to other hardware platforms without any changes to Web HMI.

- Development, Maintenance Cost

Since the same code base is used across multiple user device types and Operating Systems, there is no need to separately develop apps for different user devices. Hence, the development cost is reduced. It becomes easy to maintain the code in the future. Web HMI is not required to be installed on user devices.

- Development Tool Support

Web technologies enjoy tremendous support for development tools to edit, build, test and deploy web apps. There are hundreds of free as well as paid options, as opposed to expensive, vendor locked embedded tools.

- Future Proof

Web technologies are here to stay and is a foolproof way to guard against obsolescence. Internationally recognized bodies ensure forward compatibility as standards evolve over time. Since the Web HMI is independent of hardware platform, they can be easily ported to any target hardware.

- Cloud (or On-Premise) connectivity

The instrument's features can be easily extended to provide connectivity to a cloud-based application, paving the way for an IoT solution, where analysis data can be stored and tracked. The instrument can fetch configuration data, feature specific licensing information, software/firmware upgrades from the cloud. Additionally, the use of genuine consumables can be ensured. A single dashboard can monitor multiple instruments in different locations. Moreover, diagnostics and test results data can be easily handled. For certain applications, intensive, computational analysis can be done on the cloud

- Plugin and community support

The availability of tried and tested plugins greatly reduces the development challenge. Since web technologies have the largest community support and enjoy continuous contribution from the developer community from across the world, future modifications to meet the evolving needs can easily be carried out.

- Remote Access to HMI

Remote HMI for headless instruments (without local displays) can help lower instrument cost. Remote diagnostics is also possible, allowing service technicians to remotely troubleshoot the instrument

- Extensibility

The architecture of Web HMI makes it easy by design for end users to extend the capabilities of the instruments by using the web API. Advanced users can write their own applications in programming languages of their choice and control the instrument through the web API. This allows users to seamlessly integrate the instrument into their lab work process.

Web HMI vs Traditional HMI

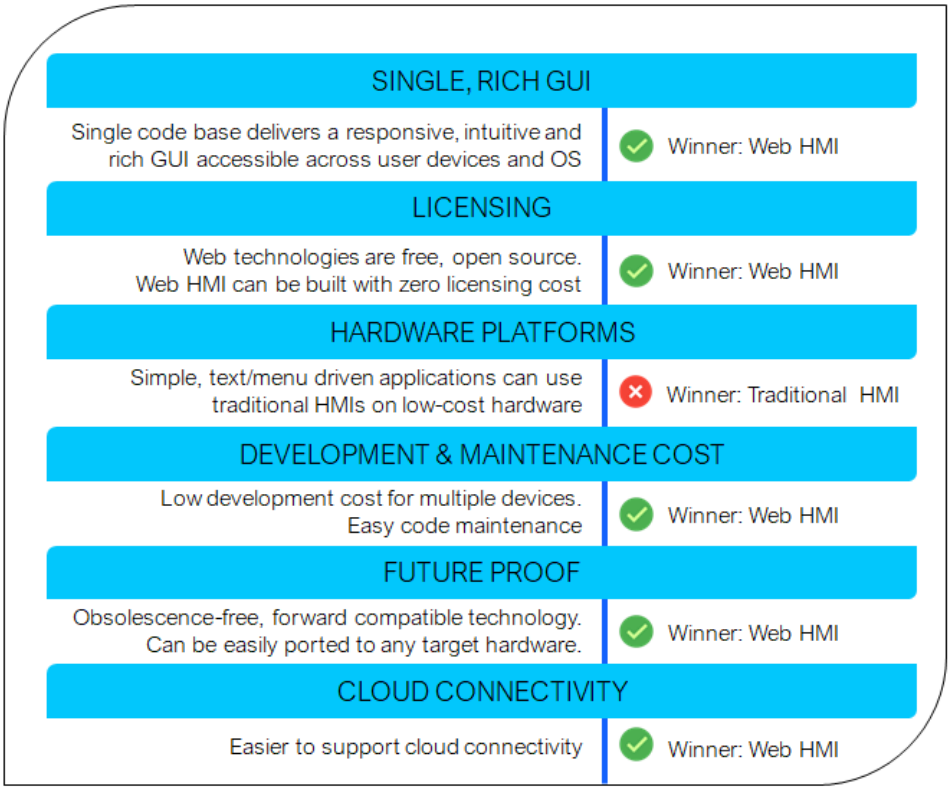

Figure 2: Comparison of Web HMI and Traditional HMI

Technology

The Web HMI can be hosted on a web server, embedded in the instrument. It will comprise of two modules, the GUI and the application logic

- GUI Programming Languages, Frameworks and Tools
  - HTML: Primary Markup language used to display the HMI
  - CSS: Styling and theming the HMI
  - JavaScript/TypeScript: GUI functionality including navigation, animation
  - Popular Frameworks/Libraries: Angular, Vue.js, React.js, Bootstrap.js

- Application logic Programming languages, Frameworks and Tools

  - The application logic can be implemented in languages like C#/.NET Core, C++, Java, Python, JavaScript/TypeScript, DART, RUST etc.
  - Node.js, ASP .NET Core, Django etc.
  - This application logic will communicate with the microcontroller, via Ethernet/USB, responsible for the control of peripherals like relays, heater coils etc and for data acquisition from the sensors.

- Web HMI is deployed on Web servers like Apache®, NGINX, Kestrel etc.

- The communication between the GUI and application logic will be through Web API

- The communication to the different user devices can be through Ethernet, Bluetooth or Wi-Fi.

- Security
  Addressing security concerns is imperative while enabling remote access to any instrument.
  - HTTPS protocol along with strong authentication ensures that only identified users or applications have access to the instrument.
  - Role based authorization guarantees that only those users with the proper clearance can view or interact with the instrument. It also provides fine grained control over the instrument's functionalities.
  - Authentication and Authorization features can be re-used to comply with FDA's 21 CFR Part 11.
  - Security hardening of the Operating System is another standard security measure. It includes secure configuration, timely updates, creation of rules/policies and removal of unnecessary applications/services to minimize exposure to threats and to mitigate possible risk.

Experiment

Objective

To get empirical comparative data for Web HMI and Traditional HMI on a Bioreactor.

Methodology

For the experiment, the 'Home' screen would display real time data from the instrument for parameters such as dissolved oxygen and temperature. The real time status of the agitator and sparger would be represented with 2D/3D animations. The 'Trends' screen would display 3 graphs, plotting the temperature, RPM and pH readings of the Bioreactor.

The selected options were

- C++/TouchGFX on Arm® board
- C++/Qt® on Raspberry Pi
- Web HMI on Raspberry Pi
- Web HMI on Industrial PC

The same code would be used for both Web HMI implementations.

To measure performance, it was decided that the data processing rate would be increased until each option reached its respective borderline unusable state. The CPU usage and memory overhead would also be measured at these upper limits of data processing.

Setup	Processor	Memory	Operating System	GUI	Application Logic
C++/TouchGFX App on Arm® Board	STM32F429NIH6 (Core: Arm® Cortex® M4) CPU Frequency: 180 MHz (Using external clock source of 25 KHz)	Flash: 2 MB SRAM: 256 KB	FreeRTOS™ v7.3.0	TouchGFX 4.10.0	C++
C++/Qt® App on Raspberry Pi 3 Model B	Broadcom BCM2387 chipset. 1.2GHz Quad-Core Arm® Cortex®-A53 (64Bit)	1 GB LPDDR2 SDRAM	Raspbian	Qt® v 5.11.3 Qt® WebGL	C++
Web HMI on Raspberry Pi 3 Model B	Broadcom BCM2387 chipset. 1.2GHz Quad-Core Arm® Cortex®-A53 (64Bit)	1 GB LPDDR2 SDRAM	Raspbian	HTML5, CSS, TypeScript on Angular 2.0. Three.js, Bootstrap.js	C# .NET Core
Web HMI on Industrial PC	64 bit Intel® Celeron® CPU J3455 @ 1.5 GHz, 4 Cores	Transcend 4GB DDR3 @ 1600 MHz	Ubuntu 16.04 Kernel Version: 4.15.0-45-generic	HTML5, CSS, TypeScript on Angular 2.0. Three.js, Bootstrap.js	C# .NET Core

Table 1: Hardware specifications of the experimental setup

Implementation

C++/TouchGFX App on Arm® Board

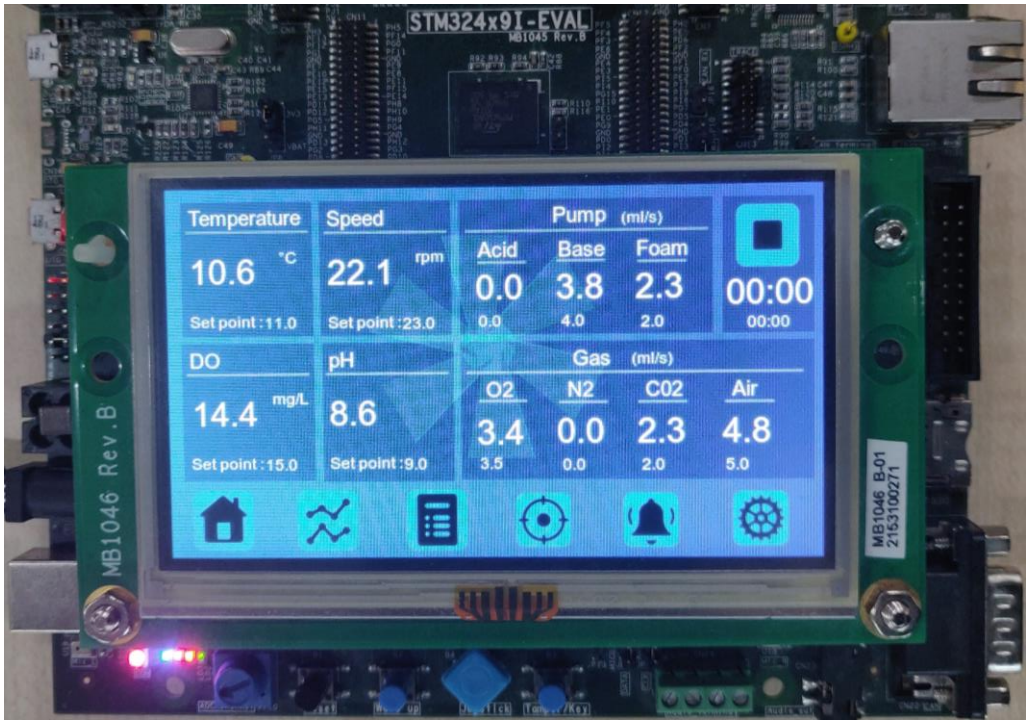

Figure 3: Home Screen of C++/TouchGFX App on Arm® Board

C++/Qt® App on RPi 3 Model B

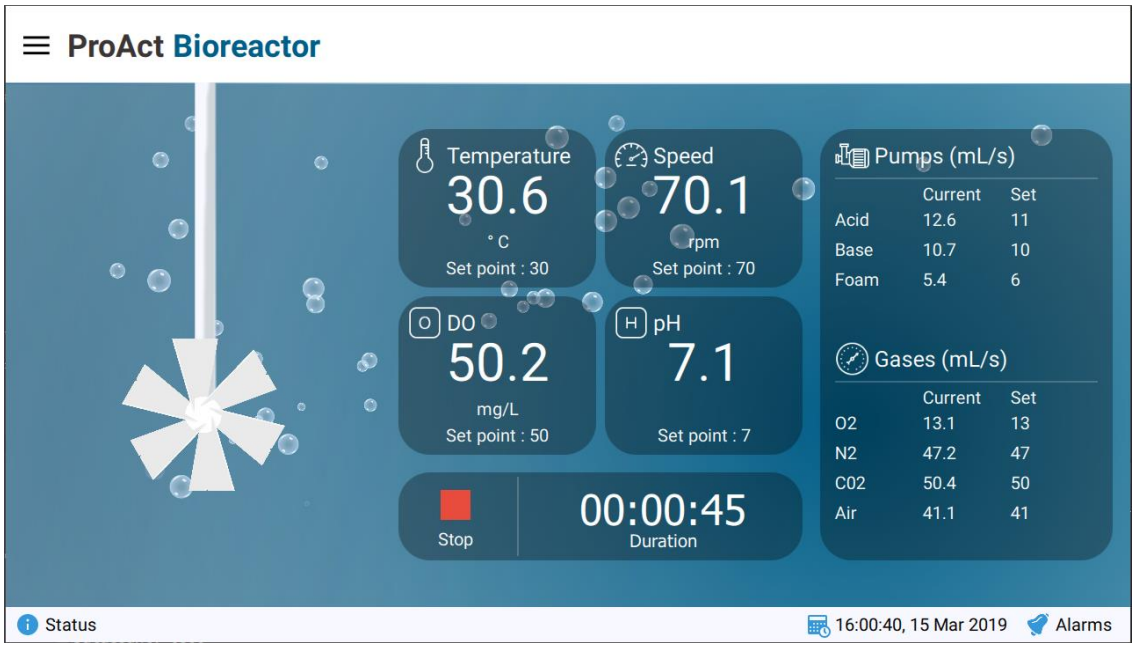

Figure 4: Home Screen of C++/Qt® App on RPi 3 Model B

*App icons by icons8 (https://icons8.com/)

Architecture of Web HMI Implementation

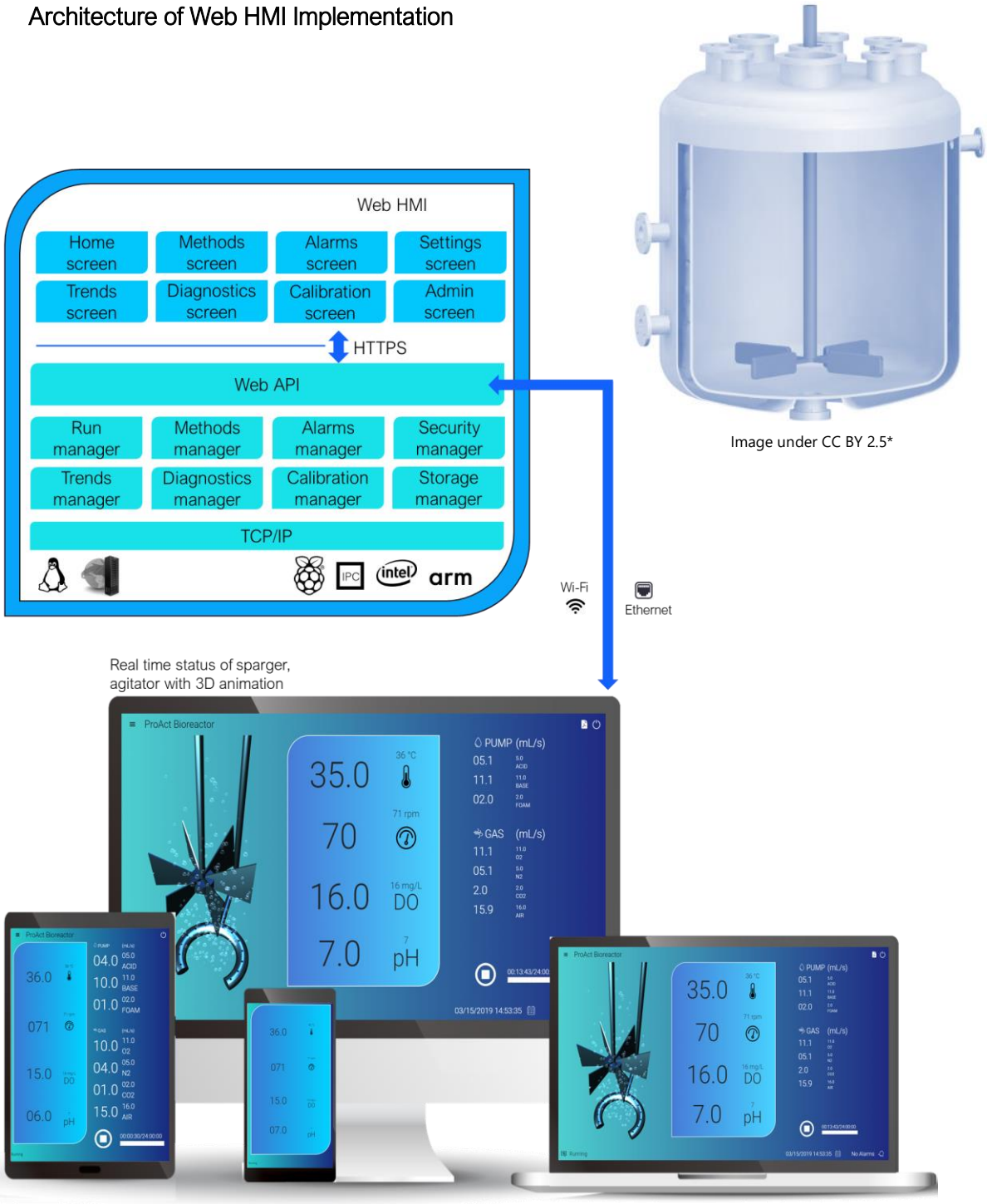

Image under CC BY 2.5*

Figure 5: Architecture of Web HMI for Bioreactor

*"Batch reactor.2" by Echis at English Wikipedia is licensed under CC BY 2.5

The Web HMI was deployed simultaneously on an Industrial PC and a Raspberry Pi, running Linux and Raspbian Operating Systems respectively. It was hosted on a Kestrel web server, a cross-platform web server for ASP.NET Core, and was kept modular. Model View Controller (MVC) design approach was used for the development along with other design patterns like factory, adapter, strategy etc.

For this implementation, TCP/IP was used to enable communication between the application logic of the Web HMI and the microcontroller. Since the communication layer and application logic were mutually abstracted, future additions of other communication protocols become hassle-free and does not require any change to the application logic.

Individual modules were created to manage functionalities such as methods, security, and trends. The 'Run Manager' is responsible for monitoring instrument status during the run. The 'Methods Manager' allows users to create methods for different chemical studies and modify existing ones. The 'Alarms Manager' is designed to alert users to instrument malfunctions, threshold breaches and other unexpected events. The 'Security Manager' is responsible for user authorization and authentication. The 'Trends Manager' acquires and analyses real time data and displays plots to visually represent data from the instrument. The 'Diagnostics Manager' allows authorized users to troubleshoot the instrument. The 'Calibration Manager' allows users to re-calibrate the instrument and check precision. The 'Storage Manager' took care of data storage using an SQLite database.

The GUI was kept decoupled from the application logic and was designed to be accessed over secure HTTPS. User devices could access the remote GUI via any browser, over Wi-Fi or Ethernet.

Performance

Performance was measured by increasing the data processing of each implementation to its respective borderline unusable states. Both data acquisition and display refresh were done every 100 ms, and the data pumped into the system was increased in steps. CPU load and memory usage were measured at the upper limit of each implementation.

Performance	CPU Load	Memory Usage	Data processing rate (3 graphs)
C++/TouchGFX App on Arm® Board	30% with 2D animation	187 KB	800 points/graph/100 ms
C++/Qt® App on RPi 3 Model B	32% with 2D animation	122 MB	9000 points/graph/100 ms
Web HMI on RPi 3 Model B	30% (Google Chrome) with 2D animation	436 MB (Google Chrome)	11000 points/graph/100 ms
Web HMI on Industrial PC	29% (Google Chrome) with 3D animation	340 MB (Google Chrome)	25000 points/graph/100 ms

*CPU Load and Memory Usage were both measured close to the maximum data processing rate (approximately represented here as Data Processing Rate)

Table 2: Performance of each implementation

Conclusion

Outcome	Remote Access (mobile/tablet/desktop)	Cost	Future Proof	Rich UI
C++/TouchGFX App on Arm® Board	✗	No Licensing cost Low Hardware cost ✓	Code not portable Vendor dependent ✗	Relatively limited UI Constraints on animation Vendor locked libraries −
C++/Qt® App on Raspberry Pi 3 Model B	✗	High Licensing cost Low Hardware cost ✗	Same code portable across OS Vendor dependent −	Modern, intuitive UI 2D animations Vendor locked libraries ✓
Web HMI on Raspberry Pi 3 Model B	✓	No Licensing cost Low Hardware cost Low development cost for multiple user devices ✓	Same code across OS Forward compatible Good community and plugin support ✓	Modern, intuitive UI 2D animations Numerous libraries ✓
Web HMI on Industrial PC	✓	No Licensing cost High Hardware cost Low development cost for multiple user devices −	Same code across OS Forward compatible Good community and plugin support ✓	Modern, intuitive UI 2D, 3D animations Numerous libraries ✓

Table 3: Outcome of the experiment

C++/TouchGFX App on Arm® Board

A simple but modern UI, with 2D animation, was implemented. Remote access from user device would require the development of separate apps. Overall, the implementation was quite inexpensive. The UI design is limited by vendor locked templates and libraries. This is suitable for instruments that do not need remote access and need a simple user interface.

C++/Qt® App on RPi 3 Model B

A sophisticated UI, with 2D animations and satisfactory performance, was implemented. Since Qt ® is supported on multiple platforms, it should be easy to port across Operating Systems. Remote access from user device would require the development of separate apps. The licensing cost and vendor locked libraries proved to be the downsides. 3D animations are not recommended. This is suitable for instruments that do not need remote access and need a more complex user interface.

Web HMI on RPi 3 Model B

The Web HMI implementation on Raspberry Pi resulted in a modern, intuitive GUI. It supports built-in remote access and can run the same code base across different user devices (mobile/tablet/desktop). Moreover, complex animations in the UI are also possible, however, 3D animations are not recommended. This is suitable for instruments that require remote access and where RPi or similar SBCs provide a low-cost option. Future user interface requirements can be accommodated through effortless migration to a high-end SBC.

Web HMI on Industrial PC

The same code base that was run on RPi was deployed and executed on Industrial PC, and resulted in a modern, intuitive GUI. Moreover, complex animations, including 3D, were possible. It supports built-in remote access and can run the same code base across different user devices (mobile/tablet/desktop). This is suitable for instruments that need remote access and require a high-end and high-performance user interface

TISMO
Embedding Excellence

ELECTRONICS

Sensor, Instrument Controller PCB design & development

Control of Motors, Valves, Heaters

Signal conditioning, Energy efficient

Prototyping, Testing & Certifications

FIRMWARE

Measurement, Control & Automation

Real time signal processing & analysis, Web HMI

Modbus, LIMS, Profibus, HL7, ZigBee, BT LE

C, C++ on RTOS / Embedded Linux

SOFTWARE

Device control software

Acquire, Analyze, Visualize & Reporting software

Web, Mobile, Desktop Apps, Cloud deployment

C# / .NET, C , C++, Java, Swift

MECHANICAL

Industrial Design Concept, Development

Solid and Surface modeling

Drafting, Detailing, Engineering analysis

Prototyping, Testing & Certification

CERTIFICATIONS

ISO 9001:2015

ISO 13485:2016

WE OFFER

WORLD CLASS DESIGN AND ENGINEERING SERVICES WITH HIGHEST QUALITY STANDARDS

NEW PRODUCT DESIGN AND ENGINEERING

PRODUCT ENHANCEMENT, SUPPORT & MAINTENANCE

RE-ENGINEERING LEGACY SYSTEMS

BENEFITS

ACCELERATED PRODUCT DEVELOPMENT

INCREASED ENGINEERING BANDWIDTH

HIGH RETURN ON INVESTMENT

Sample Web HMI
https://proact.tismotech.com

Tismo Technology Solutions (P) Ltd
22/2, Palmgrove Road,
Bangalore – 560047, India

E: engineering@tismotech.com
T: +91-80-4091-3741
US: +1-408-786-5648