Polished title

OrderViz: Bridging MetaTrader's (MT5) MetaQuotes Language's (MQL5) API with ZeroMQ for Real-Time Microsecond Bid/Ask Orderflow Data Extraction and Visualization Across Rust, Go, C++, Python, Java, C#, and NodeJS

Authors

Albeos, Rembrant Oyangoren
Independent Researcher

Draft. Overview Title

Binding MT5's MQL5's API: Exploiting Real-time Microsecond live trading data from MetaTrader (MT5) using ZeroMQ bound to MetaQuotes (MQL5) Expert Advisor, bridging 3 combinations to programming languages of either such as Rust, Go, C++, Python, Java, C#, NodeJS to extract and visualize developing Bid/Ask Orderflows in an external UI chart application.

MT5Flow

Exploiting Real-time Microsecond Bid/Ask Orderflows data from MetaTrader 5 Using Rust, ZeroMQ, and MetaQuotes Language (MQL5)

At the time of writing, there is no publicly known approach using Rust, ZeroMQ, and MetaQuotes Language (MQL5) combination to exploit real-time microsecond bid/ask orderflows data from MetaTrader 5, before this implementation. This study will show a new approach to extract and visualize the developing microsecond bid/ask formation with a low-latency phase. We used MT5's API to fetch real-time live trading data, bound to ZeroMQ & Expert Advisor MQL5 integration, and Rust as an external processor to generate fast-phase developing bid-ask orderflow visualization through a simple chart.

At the time of writing, it's really hard to find benchmark research papers since no one is interested in binding MT5's API using a ZMQ bridge to other programming languages such as Rust, Go, C++, and Python (creating a combination of 3). We search across the internet using 9 AI leading platforms, namely Grok, Gemini, Claude, ChatGPT, MSCopilot, Perplexity, Baidu (Ernie Bot), DeepSeek, and Qwen (1, 2, 3, 4, 5, 6, 7, 8, 9). Some of the AI's responses end up hallucinating, suggesting a combination of 2, and not the strictly 3, but later been clarified and confirmed that there are no public projects yet to implement before this. We also searched across four scholarly libraries, such as Arxiv, Google Scholar, SSRN, and Baidu 学术 (10, 11, 12, 13). The lack of literature suggests that this study's approach (specifically using MT5's API to fetch live-trading datafeed from its connected Broker) is bound to ZMQ, bringing the other programming languages are not that popular and not well explored.

Detailed ZMQ (The DLLs require that you have the latest Visual C++ runtime (2015).) binding to MQL - https://github.com/dingmaotu/mql-zmq (14)
ZMQ binding to Rust - https://github.com/zeromq/zmq.rs (15)
ZMQ binding between MQL and JavaScript - https://github.com/EricSchles/bindings-mql4-5 (16)

We used this repository to train (be-aware) locally our Gemini Pro inside Google Antigravity to analyze and have a benchmark of what has been proven to work so far. This approach is similar to training LLMs using open-source repositories to generate source code. Such as Proposes *RepoMark*, a framework to audit whether a code repository has been used in training a code large language model, addressing transparency and license compliance concerns in training on open-source projects (17). Investigates whether a given model has *actually used* specific code from public repositories in its training data via membership inference, providing methods for detecting code inclusion (18). By being mindful of regulatory concerns (19) and following practices (20). With that being said, Gemini Pro already has access to the internet.

MetaTrader 5 (MT5)

MetaTrader 5 (MT5) is a multi-asset trading platform developed by MetaQuotes Software and launched in 2010. It is designed to facilitate trading in various markets, including Forex, stocks, futures, and CFDs. (21). It supports connecting to a regulated broker (22)

MetaQuotes Language 5 (MQL5) MQL5 Reference – How to use algorithmic/automated trading language for MetaTrader 5 (23)

MQL5 API Common APIs - MQL5 Programming for Traders (24)

Methodology
Combination of

A	Rust, ZeroMQ, and MQL5	RuZM
B	Go, ZeroMQ, and MQL5	GuZM
C	C++, ZeroMQ, and MQL5	C+ZM
D	Python, ZeroMQ, and MQL5	PyZM
E	Java, ZeroMQ, and MQL5	JaZM
F	C# , ZeroMQ, and MQL5	C#ZM
G	Node.js, ZeroMQ, and MQL5	NoZM

Make a software application for those combinations and make a simple bid/ask exploit. Download CSV files, run all software at once using the same ZMQ socket. Compare using a correlation matrix to see how well (in %) does the developing bid/ask formation correlate to each other.