Automating a Day Trading Strategy Based on Fair Value Gaps: Technical Implementation, Challenges, and Best Practices

Executive Summary

Fair Value Gaps (FVGs) represent a compelling price action concept for identifying market inefficiencies where price has moved too quickly, leaving behind untraded levels. While FVGs have gained significant popularity among discretionary traders, automating an FVG-based day trading strategy presents substantial technical, operational, and risk management challenges. This essay explores the architectural requirements, implementation strategies, critical limitations, and best practices for developing a robust automated FVG trading system.[1][2]

Understanding Fair Value Gaps and Their Characteristics

The Core Concept

A Fair Value Gap is identified through a specific three-candle pattern: the price action creates an imbalance where the middle candle's body causes a gap that remains uncovered by the third candle. For bullish FVGs, the high of the first candle remains above the low of the third candle, creating an uncovered space. Conversely, bearish FVGs occur when the low of the first candle remains below the high of the third candle. These imbalances represent areas where institutional orders may have been placed but not fully executed, theoretically creating zones where price is likely to return.[3][1]

The rationale underlying FVG trading stems from market microstructure. When algorithms execute large trades at institutional speed or when liquidity suddenly contracts, price can gap over areas where limit orders exist. The market, driven by both algorithmic trading and human participation, tends to "fill" these gaps as traders seek liquidity or as the imbalance attracts new trading activity. FVGs become particularly pronounced during low liquidity periods, news-driven volatility, or when rapid algorithmic execution creates inefficiencies.[4]

Pattern Validation and Parameters

For effective automation, FVG identification requires precise mathematical definitions. The most sophisticated automated systems incorporate gap size thresholds to eliminate noise. A bullish FVG becomes statistically meaningful only if the gap size exceeds a certain volatility measure, typically calculated as a multiple of the average candlestick body size over the preceding N periods. Traders often require the middle candle's body to be at least 1.5 times the average body size to confirm a genuine FVG rather than minor intrabar noise. This filtering significantly reduces false signals and computational overhead while maintaining genuine trading opportunities.[5]

Architectural Components of an Automated FVG System

1. Data Acquisition and Feed Management

Any automated FVG system requires real-time market data with sufficient granularity and accuracy. For day trading, tick-level or minute-bar data is essential, with latency measured in milliseconds. The data pipeline must handle multiple assets simultaneously and manage connection resilience across market hours.[6]

Critical Requirements:

- Real-time tick data or high-frequency bar data with minimal latency
- Historical data repository for backtesting and walk-forward analysis
- Redundant connections to prevent data gaps
- Accurate timestamp synchronization across distributed systems
- Volume and liquidity information for order book depth

Modern implementations typically leverage API connections to brokerage platforms or specialized data providers. CCXT libraries enable cryptocurrency trading bots to access unified APIs across multiple exchanges, while forex and equity traders often connect directly to MetaTrader 5 via MQL5 or through direct broker APIs. Cloud-based infrastructure (AWS, GCP, DigitalOcean) provides scalability for handling large datasets and parallel computations.[7][8][9]

2. Signal Generation Engine: FVG Detection

The signal generation engine represents the core logic for identifying valid FVGs. This component executes the pattern recognition algorithm on each new candle close, maintaining historical context and calculating real-time indicators.

Implementation Strategy:

For each new candle, the system retrieves the preceding three candles and performs pattern matching against defined FVG criteria. It calculates the average body size over a rolling window (typically 10-20 candles) and validates whether the middle candle's body exceeds the minimum threshold. For bullish FVGs, the algorithm verifies that the high of the first candle exceeds the low of the third candle, creating an uncovered gap. The system then records the gap range boundaries for later entry and exit calculations.[5]

An enhanced system incorporates multi-timeframe analysis, where FVGs on shorter timeframes (1-minute or 5-minute for day trading) must align with higher timeframe structure for increased reliability. The system maintains a registry of active FVGs, tracking whether each gap has been fully or partially filled, which influences trade management decisions.[10]

3. Entry Logic and Execution Strategy

Automating FVG entries requires sophisticated decision rules accounting for slippage, liquidity, and false breakouts.[10]

Entry Approaches:

Retracement Entry: The algorithm waits for price to return to the FVG edge before entering. Upon price approaching the gap boundary within a defined tolerance (e.g., within 2 pips for forex), the system generates an entry signal. This approach minimizes premature entries but risks missing rapid fills.[2]

Confirmation Entry: After price enters the FVG zone, the algorithm waits for confirmation through secondary signals—candlestick patterns, volume spikes, or momentum indicators—before committing capital. This reduces whipsaw risk but introduces timing lag and execution delays.[2]

Aggressive Entry: The system places limit orders within the FVG in anticipation of it being filled. This maximizes execution at optimal prices but introduces fill uncertainty and potential non-execution if price rapidly moves away.[2]

For day trading, the retracement entry typically outperforms due to reduced false breakout risk. The system should implement conditional logic preventing entries during low-volatility sideways markets and emphasizing entries within confirmed trends. Entry rules must also incorporate time-of-day filters, avoiding entries near market close or during known high-spread periods.[11]

4. Risk Management and Position Sizing

Robust automation requires deterministic risk controls that execute without human discretion.[12]

Position Sizing Logic:

The most reliable method is fixed percentage-based sizing: risking no more than 1% of account equity per trade. For an FVG trade with a known stop-loss distance, the position size calculates proportionally to the risk parameters. More sophisticated systems employ volatility-adjusted sizing using Average True Range (ATR), increasing position size during low-volatility periods and decreasing exposure during high-volatility environments. However, this introduces complexity that may degrade system robustness.[13][14]

Stop-Loss Placement:

For bullish FVGs, the stop-loss automatically places below the gap's lower boundary plus a safety buffer (typically 5-10 pips depending on asset volatility). For bearish FVGs, stops place above the gap's upper boundary. The system enforces these stops through hard-coded order management, preventing emotional or discretionary overrides during adverse price movements.[2]

Take-Profit Targets:

A disciplined profit target framework improves consistency. Common approaches include fixed multiple targets capturing 2:1, 3:1, and 4:1 reward-to-risk ratios, primary targets at complete FVG fill with trailing stops capturing additional moves, or ATR-based targets using market volatility to scale profit targets dynamically. For day trading, trailing stops prove particularly valuable, allowing the system to lock in gains as momentum builds while capturing extended moves.[15][1][2]

5. Order Management and Execution

Automated execution requires sophisticated order handling accounting for partial fills, rejections, and market volatility.[16][7]

Order Types:

- Entry orders through limit orders for retracement entries; market orders for confirmation or aggressive entries with maximum slippage parameters
- Stop-loss orders executed immediately upon trigger, using stop-limit orders to control execution price within acceptable slippage
- Profit-taking orders across multiple levels, with remaining quantity subject to trailing stops

Execution Safeguards:

- Maximum order duration limits preventing stale orders from execution during gaps
- Automatic order cancellation if positions reach maximum drawdown thresholds
- Position deduplication logic preventing accidental double entries
- Error handling and retry logic for failed submissions with exponential backoff

The system must handle partial fills intelligently, maintaining accurate position tracking and adjusting stop-loss and profit-target sizes to reflect actual executed quantity. This becomes critical in lower-liquidity instruments where fills across multiple price levels are common.[7][16]

Backtesting, Validation, and the Reality Gap

Backtesting Methodology

Developing an FVG automation system requires extensive historical validation before live deployment. The backtesting framework must simulate realistic trading conditions including slippage, commissions, spreads, and execution delays.[17][18]

Critical Backtesting Parameters:

- **Slippage Assumptions:** 0.5-3% performance reduction annually from realistic slippage is typical. Day trading in forex might assume 1-2 pip slippage; equities might assume 5-10% of bid-ask spread[19]
- **Commission and Fees:** Including per-trade commission, platform fees, and any borrowing costs for short positions

- **Liquidity Constraints:** Limiting order sizes to realistic market depth, with price impact models for larger orders[19]
- **Data Quality:** Using tick-level data rather than OHLC bars where possible to capture intrabar volatility more accurately

Backtesting software ranges from platform-specific solutions (MetaTrader's Strategy Tester for MQL5, TradingView's Pine Script Backtester) to dedicated frameworks (AmiBroker with Monte Carlo simulation, Python-based tools using historical data libraries).[17][7]

Walk-Forward Analysis and Overfitting Prevention

The most critical backtesting weakness is overfitting—tuning parameters until historical performance looks artificially perfect, only to fail in live trading. Walk-forward analysis systematically addresses this by simulating real-time parameter optimization.[20][21][22]

The methodology divides historical data into overlapping windows. Initial optimization occurs on the first period using standard optimization techniques, then the strategy is tested on subsequent out-of-sample data. The window then shifts forward, and the process repeats. If a strategy shows consistent profitability across multiple windows while maintaining stable parameter values, it has demonstrated genuine edge rather than curve-fit luck.[21][20]

For FVG strategies, walk-forward analysis particularly matters because parameter sensitivity is high. Small adjustments to gap-size thresholds, timeframe selections, or confirmation filters dramatically affect trade frequency and win rates. A strategy showing 60% win rate with specific parameters in one time period but only 40% in the next represents a red flag for overfitting.[20]

Monte Carlo Simulation

Beyond walk-forward analysis, Monte Carlo simulations evaluate strategy robustness by randomizing historical trade sequences. By reordering actual trades and recalculating drawdowns and returns, this method reveals whether a strategy's success depends on a specific sequence of fortunate trades or represents genuine, order-independent edge.[23][24]

Monte Carlo analysis typically runs hundreds of simulations, each generating a different trade sequence. The resulting distribution of outcomes indicates probability ranges for maximum drawdown, returns, and recovery factors. A robust strategy shows narrow probability

distributions; a fragile strategy shows wide variation in outcomes, indicating sequence dependency.[23]

The Backtesting-to-Live Gap

Despite rigorous backtesting, live trading performance inevitably diverges from simulated results. Multiple factors contribute:[18][25][19]

Execution Slippage: Backtests assume idealized fills; live markets often gap through projected entry/exit levels, particularly during news events or rapid moves. Realistic slippage can reduce annual returns by 0.5-3%, though it varies dramatically across instruments and market conditions.[19]

Latency and Non-Execution: Even limit orders fail to fill as anticipated. In fast-moving markets, a backtested entry at a specific price might never execute, causing missed trades. Backtests cannot fully simulate order queue positions or their impact on fill probability.[26]

Regime Changes: Market structure evolves. Low-volatility regimes transition to high-volatility environments; trending markets become range-bound; liquidity patterns shift. Strategies optimized during quiet periods often deteriorate during volatility spikes because FVG patterns behave differently under stress.[18]

Psychological Factors: Backtests impose perfect discipline; live trading does not. When actual capital is at risk and drawdowns accumulate, emotional trading can override system rules. Automation mitigates this through enforced algorithm adherence, yet system monitoring challenges can tempt manual intervention.[18]

The wise approach involves starting live trading at reduced scale (micro-contracts, minimal position sizes) on paper trading or with simulated funds, allowing real-time performance comparison against backtests. Only after demonstrating consistent real-world results should traders scale to target position sizes.[27][18]

Implementation Platforms and Technology Stack

MQL5 for Metatrader 5

MQL5 remains the industry standard for automating forex and CFD trading strategies. The platform offers Expert Advisors (EAs) as self-contained trading robots implementing custom

logic, built-in backtesting engines with optimization capabilities, and the MQL5 Wizard for code generation creating EAs from modular components without manual programming.[8][28]

For FVG automation, developers code pattern recognition algorithms identifying three-candle gaps, then integrate entry/exit logic with risk management modules. The MQL5 ecosystem provides extensive libraries for technical indicators, order management, and risk controls, significantly accelerating development.[28]

Python with CCXT for Cryptocurrency

Python combined with the CCXT library enables cryptocurrency trading bot development with access to 100+ exchanges through a unified API interface. The architecture typically involves data acquisition through WebSocket connections for real-time price feeds and REST API for historical data, signal engines using custom Python functions detecting FVG patterns with NumPy for vectorized calculations, execution layers submitting orders through exchange APIs with state management tracking active positions, and risk modules handling position sizing, stop-loss placement, and portfolio-level exposure monitoring.[9][29][7]

Python development offers flexibility and rapid iteration but requires careful attention to latency, thread management, and error handling. Dependencies on external libraries (Pandas, NumPy, TensorFlow for advanced variations) introduce complexity but enable sophisticated analysis.[30][7]

Hybrid Approaches with TradingView Webhooks

Many traders implement hybrid systems using TradingView's Pine Script for signal generation combined with execution through brokers. TradingView alerts trigger webhooks that call backend services (Lambda functions, Node.js servers) executing trades through broker APIs. This separation of concerns simplifies development but introduces latency through the webhook layer—typically 100-500ms, meaningful for day trading but acceptable for swing strategies.[31][12]

Critical Challenges and Limitations

1. Whipsaw Risk and False Breakouts

FVGs inherently suffer from false breakout risk. Market makers and algorithmic traders intentionally create fakeouts, triggering stop-losses on small traders before reversing direction. Approximately 70% of breakouts fail in certain market conditions, and FVGs exhibit similar vulnerability.[32]

Automation addresses this through multi-layer confirmation (requiring additional indicators, volume validation, trend alignment) and wider stops accepting larger per-trade losses to reduce whipsaw casualty rate. However, broader stops directly conflict with position sizing discipline and portfolio-level risk constraints. The optimal solution involves acknowledging inherent whipsaw rates and designing profit targets and risk ratios that remain profitable despite losing 40-50% of trades.[32][10]

2. Liquidity and Order Execution Certainty

Backtests assume orders fill at specified prices; reality introduces execution uncertainty. During rapid price moves, limit orders might never execute; during slow moves, market orders execute with wide slippage. Automated systems must accommodate this through timeout logic canceling unfilled orders after specified periods, alternative execution strategies (e.g., if limit order doesn't fill within 2 seconds, submit market order), slippage toleration and acceptance of suboptimal fills, and volume analysis preventing oversized orders in illiquid instruments.[33][34][19]

3. Latency and Technology Infrastructure

Even with excellent infrastructure, latency matters. Order submission latency (250-500ms for retail brokers) means entry prices have already moved substantially by order execution. High-frequency trading systems achieve microsecond execution through co-location and direct exchange connections—unavailable to most retail traders.[35][26]

Automated systems running on cloud infrastructure introduce additional latency through network transmission. A strategy requiring sub-second reaction times to FVGs will struggle with cloud-based execution. Dedicated servers in data centers closer to exchanges reduce latency but increase operational complexity and cost.[36][37]

4. Overfitting and Parameter Optimization Fragility

Parameter optimization easily produces false positives. Different gap-size thresholds or body multipliers can show dramatically different results on historical data, yet which is "correct"?

Without rigorous out-of-sample testing, chosen parameters likely represent historical fitting rather than genuine edge.[22][20]

FVG strategies typically have 5-15 key parameters: gap size threshold, body multiplier, timeframe selection, confirmation period, profit target multiples, stop-loss buffer distance, and trading hours restrictions. The interaction effects of parameter combinations create an enormous search space, making overfitting almost inevitable without disciplined methodology.[22]

5. Market Regime Dependency

FVG effectiveness depends on market structure. During trending periods with high volatility, FVGs perform well as price rapidly moves away then returns to fill gaps. During choppy, range-bound markets with frequent reversals, FVGs generate false signals as price oscillates without filling gaps. Automated systems adapting to regime changes face increased complexity and additional overfitting risk.[10]

Best Practices for Robust Automation

1. Comprehensive Backtesting Protocol

Before any live trading, subject the strategy to rigorous testing:

Step 1: Historical Backtesting (3-5 years minimum)
Test across multiple market cycles including bull runs, bear markets, and sideways consolidations. Include periods of known stress (March 2020 COVID crash, December 2018 correction). Verify strategy survives 30%+ drawdowns without catastrophic losses.

Step 2: Walk-Forward Analysis
Divide historical data into 8-12 overlapping windows. Optimize parameters only on in-sample portion (typically 70% of data). Test results on out-of-sample portion (remaining 30%). Verify consistent profitability and stable parameters across windows.[21][20]

Step 3: Monte Carlo Simulation
Run 500-1000 simulations with randomized trade sequences. Verify returns and drawdowns remain acceptable across simulated scenarios. Identify parameter sensitivity (does small parameter variance produce large outcome variance?).[24][23]

Step 4: Out-of-Sample Testing

Reserve recent market data for testing strategy on data the optimization never encountered. Treat this as closest approximation to live trading before deploying real capital. Verify results remain positive despite randomness of unseen data.

2. Graduated Live Deployment

Begin live trading conservatively:

Phase 1: Paper Trading (1-2 weeks)

Execute all signals via broker's demo account using real market data. Compare paper trading results against backtested expectations. Identify any execution issues, data feed delays, or system errors.

Phase 2: Micro-Position Live Trading (1-4 weeks)

Deploy on live account with minimal position sizes (0.1% risk per trade or smaller). Monitor actual slippage, fees, and fill rates versus backtested assumptions. Verify stop-loss and profit-target execution reliability. Collect live performance data for comparison.[27]

Phase 3: Proportional Scaling (ongoing)

Gradually increase position sizes while monitoring performance consistency. Establish confidence checkpoints (e.g., 20 profitable trades, positive Sharpe ratio). Implement circuit breakers stopping all trading if daily losses exceed threshold.

3. Continuous Performance Monitoring

Live trading requires ongoing analysis comparing actual performance against expectations:[25]

Daily Metrics:

Trade count and win rate comparison to backtest projections, average profit per trade and risk-reward ratio validation, maximum consecutive losses identifying regime changes.

Weekly Metrics:

Cumulative returns and Sharpe ratio trajectory, maximum drawdown realization against tolerance levels, comparison of actual slippage and fees to backtested assumptions.

Monthly Analysis:

Correlation analysis between live performance and market regimes, identification of specific

types of trades (certain timeframes, instruments, or market conditions) driving over/under-performance, confidence intervals for forward projections based on sample size and variance.

4. Risk Controls and Emergency Stops

Even well-designed systems require safety mechanisms:[13]

Hard Position Limits:
Maximum number of simultaneously open positions, maximum aggregate exposure across all positions, maximum position size in any single instrument.

Daily/Monthly Loss Limits:
Stop all trading if daily losses exceed specified threshold (e.g., 5% account equity), cease trading if monthly losses exceed threshold (e.g., 10% account equity), implement mandatory review before resuming trading after breaches.

Execution Safeguards:
Kill-switch enabling instant closure of all positions, automatic position closure during news events or volatility spikes exceeding thresholds, order timeout preventing indefinite pending orders, maximum slippage tolerance preventing market orders from executing at unacceptable prices.

The Role of Machine Learning and Advanced Optimization

Some traders incorporate machine learning (ML) into FVG automation, attempting to improve signal quality or entry/exit optimization. Supervised learning models can theoretically predict which FVGs lead to profitable fills versus whipsaws, or optimize entry timing through pattern recognition.[38]

However, ML in trading faces substantial challenges:[39][40][38]

Overfitting Risk: ML models easily overfit to historical patterns that don't generalize to new data. Complex neural networks achieve excellent backtested performance yet fail miserably live, having memorized noise rather than learned generalizable patterns.[38]

Data Quality Requirements: Machine learning requires enormous quantities of high-quality labeled data. FVG datasets are relatively sparse (perhaps 10-50 FVGs per day per instrument),

making statistical significance difficult to achieve. Poor data labels (defining "successful" FVG fills when markets contain multiple simultaneous influences) corrupt model training.[39]

Regulatory and Explainability Issues: Complex ML models lack interpretability. Regulators increasingly scrutinize trading algorithms for market manipulation or unfair advantages. Black-box ML models struggle to explain decision rationale, creating regulatory exposure and institutional resistance.[38]

Computational Overhead: Real-time ML inference demands significant computational resources, increasing latency and system complexity. The marginal benefit must justify these costs.[38]

For most traders, straightforward rules-based systems using proven technical analysis outperform complex ML variations. When ML proves valuable, ensemble methods combining traditional technical signals with ML-derived confidence scores typically work better than pure ML approaches.

Conclusion

Automating a Fair Value Gap day trading strategy represents a legitimate but non-trivial undertaking. The core concept has merit: identifying price gaps left behind by rapid market moves, then profiting as price returns to fill those gaps. However, transforming this concept into a profitable automated system requires architectural sophistication, rigorous validation, robust risk management, and realistic expectations about inevitable drawdowns and slippage.

Success requires architectural discipline—choosing appropriate platforms (MQL5 for forex, Python-CCXT for crypto, hybrid webhook approaches for flexibility), implementing multi-layer risk controls, establishing realistic backtesting protocols incorporating walk-forward analysis and Monte Carlo simulation, and maintaining continuous monitoring comparing live performance against expectations.[20][21]

The greatest challenges are not technical but psychological: accepting that backtested results will not perfectly predict live trading, tolerating the substantial drawdowns and losing periods inherent in day trading, and maintaining discipline through periods when the strategy underperforms market baselines. Well-implemented FVG automation may achieve modest positive returns (perhaps 10-30% annually after fees and slippage), but success requires

treating the endeavor as a serious business with appropriate risk controls, capital allocation discipline, and performance monitoring infrastructure rather than as a quick path to wealth.

The traders most likely to succeed are those who treat automation as an enhancement to rigorous methodology—not as a replacement for understanding market microstructure, risk management principles, and the psychological demands of real-money trading.

⁂

---

1. https://ftmo.com/en/boost-your-trading-edge-with-the-fair-value-gap-strategy/
2. https://eplanetbrokers.com/en-US/training/what-is-fair-value-gap
3. https://trendspider.com/learning-center/fair-value-gap-trading-strategy/
4. https://b2broker.com/news/fair-value-gap-what-is-it-and-how-to-use-it-in-trading/
5. https://www.youtube.com/watch?v=cjDgibEkJ_M
6. https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp
7. https://www.troniextechnologies.com/blog/built-a-custom-crypto-trading-bot
8. https://www.mql5.com/en/articles/171
9. https://www.tutorialspoint.com/building-a-cryptocurrency-trading-bot-with-python-and-the-ccxt-library
10. https://www.luxalgo.com/blog/fair-value-gap-market-imbalance-trading-hack/
11. https://www.mstock.com/articles/how-to-enter-and-exit-in-intraday
12. https://obside.com/trading-guides/best-day-trading-platform/
13. https://www.luxalgo.com/blog/risk-management-strategies-for-algo-trading/
14. https://robots4forex.com/smart-trading-with-algorithms/understanding-risk-management-in-automated-trading/
15. https://www.stockgro.club/blogs/intraday-trading/good-exit-strategies-for-day-traders/
16. https://www.biz4group.com/blog/how-to-create-an-ai-trading-bot

17. https://www.quantinsti.com/articles/algorithmic-trading-strategies/
18. https://www.pineconnector.com/blogs/pico-blog/backtesting-vs-live-trading-bridging-the-gap-between-strategy-and-reality
19. https://www.luxalgo.com/blog/backtesting-limitations-slippage-and-liquidity-explained/
20. https://www.mql5.com/en/blogs/post/756386
21. https://easylanguagemastery.com/products/walk-forward-analysis/
22. https://www.luxalgo.com/blog/what-is-overfitting-in-trading-strategies/
23. https://www.marketcalls.in/amibroker/monte-carlo-analysis-trading-system-validation-to-take-informed-trading-decisions.html
24. https://www.investopedia.com/terms/m/montecarlosimulation.asp
25. https://www.quantifiedstrategies.com/what-can-you-expect-from-a-trading-strategy-backtest-when-you-are-trading-it-live/
26. https://arxiv.org/html/2504.00846v1
27. https://greaterwaves.com/how-to-compare-backtesting-and-live-trading-results-with-gen-ai/
28. https://www.mql5.com/en/book/automation/experts
29. https://github.com/ccxt/ccxt
30. https://www.luxalgo.com/blog/building-your-first-trading-bot-step-by-step-guide/
31. https://www.reddit.com/r/Daytrading/comments/sw0h2t/can_day_trading_be_automated/
32. https://www.reddit.com/r/Daytrading/comments/1fhfiku/i_dont_like_false_outbreaks_how_to_avoid_them/
33. https://quantra.quantinsti.com/glossary/How-to-Avoid-Common-Mistakes-in-Backtesting
34. https://www.daytrading.com/order-execution-strategies
35. https://www.luxalgo.com/blog/latency-standards-in-trading-systems/
36. https://github.com/ebi2kh/Real-Time-Financial-Analysis-Trading-System
37. https://dev.to/sgchris/building-a-stock-trading-system-high-frequency-trading-architecture-e2f

38. https://www.linkedin.com/pulse/integrating-machine-learning-automated-trading-eric-jellerson-qo62c
39. https://algosone.ai/the-challenges-limitations-and-potential-of-ai-trading/
40. https://jisem-journal.com/index.php/journal/article/download/4285/1917/7054
41. https://www.buildalpha.com/automate-trading-with-no-coding/
42. https://www.mql5.com/en/forum/467330
43. https://www.fluxcharts.com/articles/Trading-Concepts/Price-Action/Fair-Value-Gaps
44. https://infotechgroup.com/automated-trading-systems-benefits-and-challenges/
45. https://github.com/AmineAndam04/Algorithmic-trading
46. https://www.reddit.com/r/algotrading/comments/h8kpue/not_all_algorithmic_trading_is_technical_analysis/
47. https://www.youtube.com/watch?v=N83qi1jRVEo
48. https://www.mql5.com/en/book/automation/tester/tester_example_ea
49. https://www.mql5.com/en/forum/ea
50. https://chartalert.in/2023/12/07/backtesting-drawbacks-of-ignoring-brokerage-or-slippages/
51. https://www.dominionmarkets.com/trading-at-the-speed-of-light-why-low-latency-matters/
52. https://www.youtube.com/watch?v=iwRaNYa8yTw
53. https://www.quantifiedstrategies.com/entry-and-exit-rules-trading-plan/
54. https://www.lesformationstrading.fr/le-trading-quantitatif-et-la-methode-de-monte-carlo-des-outils-capables-de-predire-les-tendances-du-marche/
55. https://www.bajajbroking.in/knowledge-center/breakout-strategy
56. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2397805
57. http://www.math-evry.cnrs.fr/_media/members/vlyvath/clvrs_slippage_201507.pdf
58. https://www.linkedin.com/pulse/walk-forward-analysis-smart-way-test-your-trading-yuan-cfa-cqf-exczc