Crypto Monitor on a Real Time Embedded System

Epameinondas Bakoulas

July 2025

1 Objective

Our goal in this project is to implement a crypto monitor that will run on a raspberry-pi continuously. It will collect data via a websocket for a total of 8 different symbols, perform some calculations for each symbol and save the results on .txt files.

Later on, we will expand on this idea by performing other calculations out of the scope of this project, in order to better predict the market and make better decisions on when to buy or sell.

2 Analysis

2.1 Overview

Our analysis will be performed on 8 different cryptocurrency symbols, which are BTC-USDT, ADA-USDT, ETH-USDT, DOGE-USDT, XRP-USDT, SOL-USDT, LTC-USDT, BNB-USDT, and we will use the the OKX Websocket public channel to collect the exchange data.

We're going to perform a total of 2 calculations for each symbol.

- Calculate the moving average price and volume of the symbol for the last 15 minutes.
- Calculate the pearson correlation with all other symbols (and itself).

These calculations are going to be performed every minute and the results will be saved on a .txt file. We are using a total of 5 threads, one for the websocket connection, one for the main function that checks if the connection is active, and the other 3 are for the scheduler, moving average and pearson correlation calculations.

2.2 Collecting data and fault tolerance

We will use the library `libwebsockets` to connect to the OKX websocket public channel and collect the exchange data.

In order to ensure that the connection can be re-established in case of a disconnection, we will implement a reconnection mechanism that will attempt to reconnect every 5 seconds until successful. This is done in 2 parts. First, we check if the connection is lost, and then we check if the subscriptions to the symbols are active. If not, we assume that the connection isn't established and we attempt to reconnect.

If the connection is lost, the moving average and pearson correlation calculations are still performed, using only the latest valid values.

2.3 Storing measurements

The measurements (exchanges) that arrive via the websocket will be saved in 2 places:

- In-memory for quick retrieval and calculations (last 15 minutes only), on a double-ended queue.
- In .txt files for long-term storage and analysis, separate for each symbol.

2.4 Scheduler

The calculations should be performed every minute on the minute mark. In order to achieve this, we will spawn a thread called `threadScheduler` that will run continuously and signal the threads `threadAverage` and `threadPearson` on the minute mark to perform the calculations.

In order to signal the threads to perform the calculations, we will use a combination of mutexes and condition variables, as well as use a boolean called `workReady`. When the time comes, the `threadScheduler` will lock the mutex, set `workReady` to true, and notify the condition variable. This way, we can safely signal the other threads to perform the calculations.

The scheduler is also responsible for cleaning up the old in-memory data right before we perform the calculations.

2.5 Moving Average price calculation

We're going to calculate the 15-minute moving average of both the price and volume using the following formula:

$$\text{MA} = \frac{1}{15} \sum_{i=1}^{15} p_{\text{close},i}$$

where $p_{\text{close},i}$ represents the closing price/volume at minute i within the last 15-minute window.

2.6 Pearson Correlation calculation

In order to calculate the pearson correlation between a symbol and all other symbols we will do the following:

1. Get the latest 8 moving averages of the 1st symbol
2. Create a sliding 8-element window of the moving averages of the 2nd symbol, using the last hour values
3. Calculate the pearson correlations ($60 - 8 + 1 = 53$ total) between the 1st symbol and the 2nd symbol
4. Repeat this process by changing the 2nd symbol
5. Find the maximum correlation value and its index
6. Save the results in a .txt file

In case the 1st and 2nd symbols are the same, the sliding window will stop at the 9th latest MA.
The formula for the pearson correlation is:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x_i and y_i are the moving averages of the 1st and 2nd symbols respectively, $\bar{x}$ and $\bar{y}$ are their means, and n is the number of elements in the sliding window (8).

3 Results

3.1 Delays for moving average calculations

The results are produced on a **Raspberry Pi Zero 2W** with a 1GHz quad-core CPU and 512MB of RAM. Figures 1 and 2 show the distribution of the calculation delays for the moving average using box plots, for the symbols `BTC-USDT` (first symbol out of the 8) and `BNB-USDT` (last symbol, highest delays). The delays are measured in milliseconds.

We can clearly see that the delays are generally below 20ms, and we never calculate the moving averages before the minute mark (i.e. the delay is always positive). There are some edge cases where the delay reaches above 1000ms, but this is very rare.

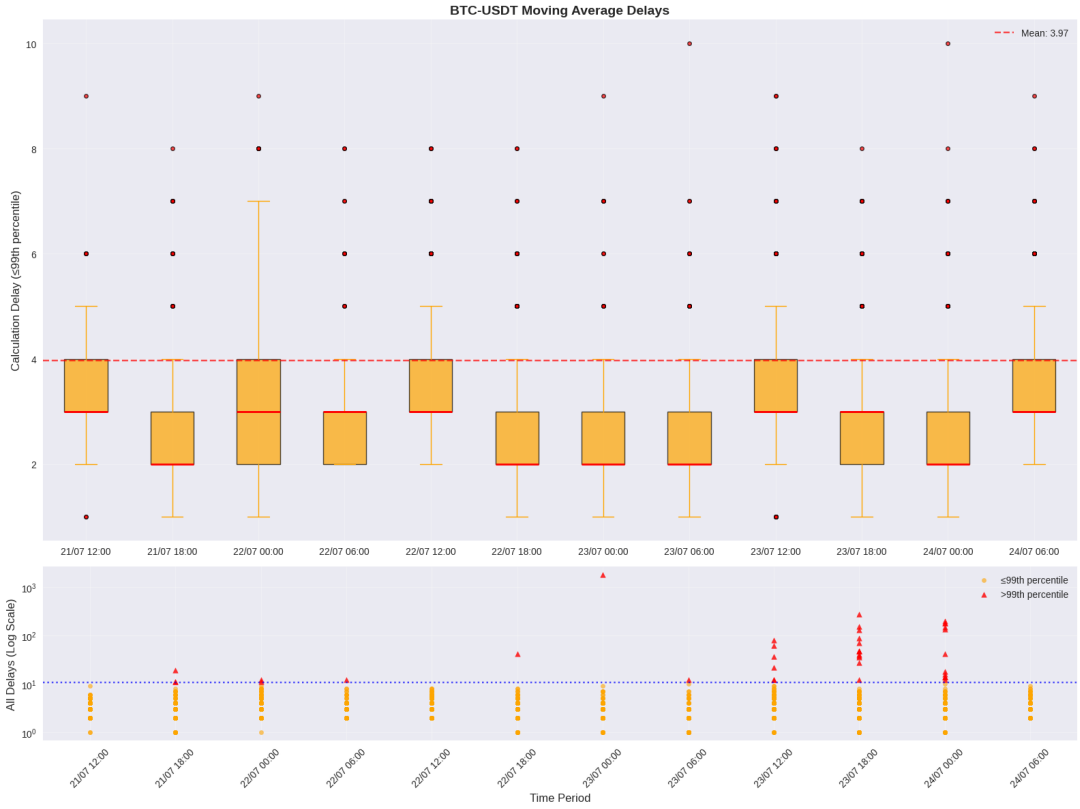

Figure 1: Moving average calculation delays for BTC-USDT (in ms)

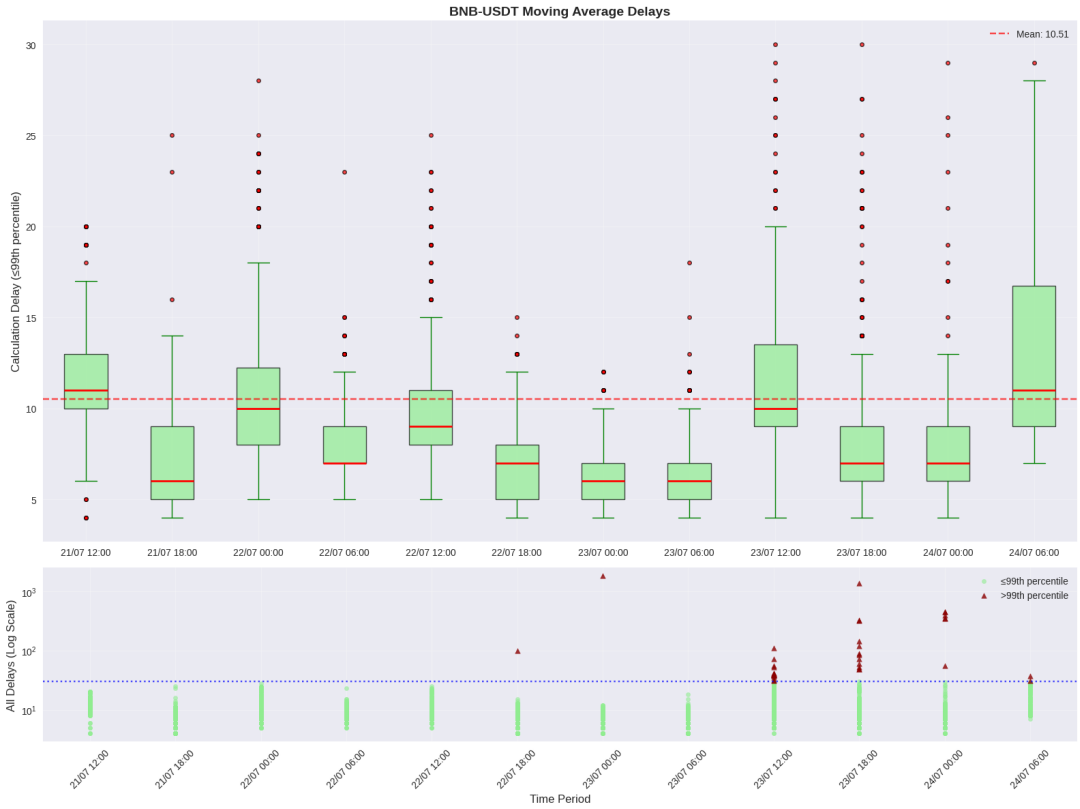

Figure 2: Moving average calculation delays for BNB-USDT (in ms)

3.2 Delays for pearson correlation calculations

Similarly, figures 3 and 4 show the distribution of the calculation delays for the pearson correlation. The delays are generally below 40ms, while on the edge cases we reach above 1000ms.

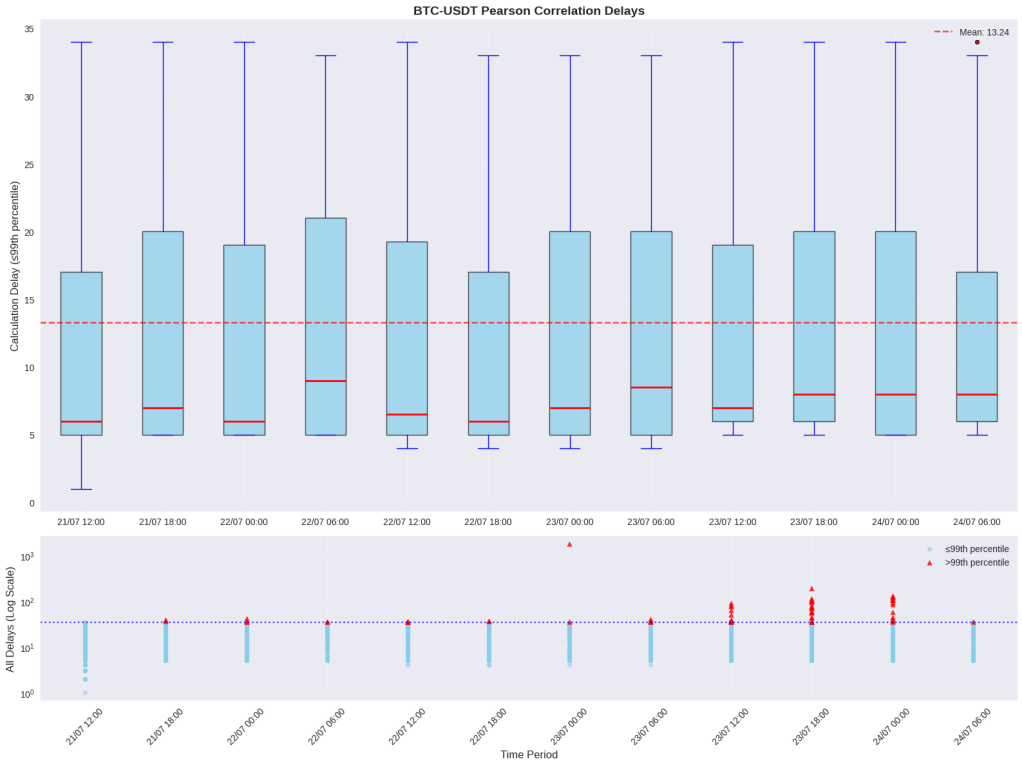

Figure 3: Pearson correlation calculation delays for BTC-USDT (in ms)

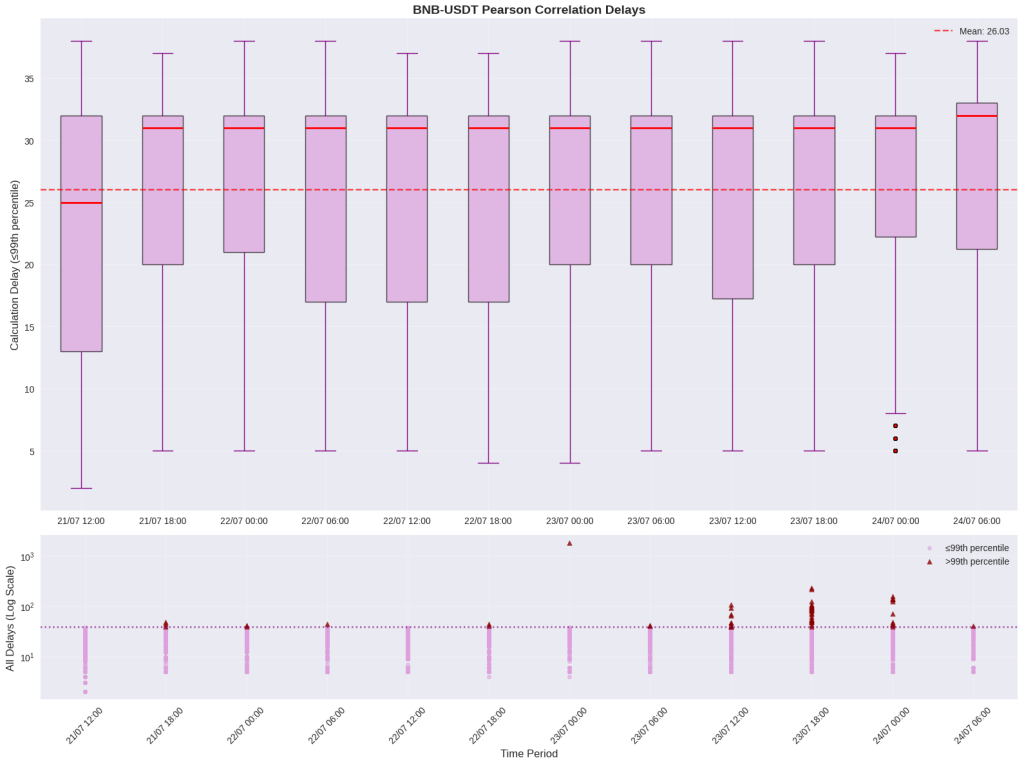

Figure 4: Pearson correlation calculation delays for BNB-USDT (in ms)

3.3 CPU Utilization

Lastly, figure 5 shows the distribution of the CPU idle percentage throughout the calculations. The CPU idle percentage is generally above 96%, while on the edge cases it can drop to 90%. Generally, the CPU is not heavily utilized, which is a good thing in case we have a limited supply of power.

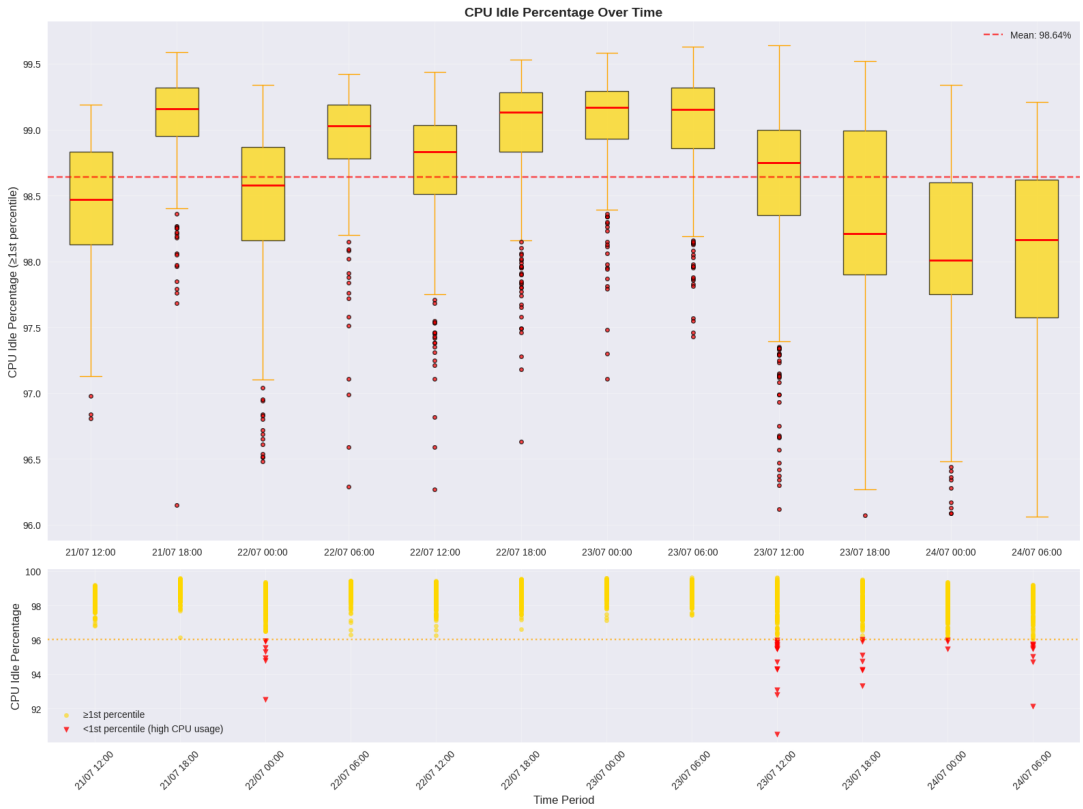

Figure 5: CPU idle percentage

4 Predicting the market

On this section, we are going to make some modifications/additions to the code in order to predict the market bettter. The steps we're going to follow are the following:

1. Instead of calculating only the `MA` (or `Simple Moving Average`, `SMA`) of the price, we will also calculate the `EMA` (`Exponential Moving Average`) of the price.
2. We will also calculate the `MACD` (`Moving Average Convergence Divergence`) of the price as well as the `signal line` of the `MACD` and the `Distance` between the `MACD` and the `signal line`.
3. We'll use the above indicators to make predictions on when to buy/sell stocks.
4. Lastly, we will deploy a web server and display all of these lines on a web browser as well as the timestamps on when to buy/sell stocks.

These metrics are of high importance if we want to make better financial decisions.

4.1 Exponential Moving Average (EMA)

The `EMA` is a type of moving average that places a greater weight and significance on the most recent data points. This means that it reacts more significantly to recent price changes compared to the `SMA`, which treats all data points equally.

We will calculate two different `EMA` values, one will be the `Short term EMA` (12 hours), and the other will be the `Long term EMA` (26 hours). The formula for the `EMA` is as follows:

$$\text{EMA} = \begin{cases} P_{\text{current}} & \text{if } \text{EMA}_{\text{previous}} = 0 \\ (P_{\text{current}} - \text{EMA}_{\text{previous}}) \cdot \alpha + \text{EMA}_{\text{previous}} & \text{otherwise} \end{cases}$$

where:

$$n = 12 \text{ or } 26$$
$$\alpha = \frac{2}{n+1}$$
$$P_{\text{current}} = \text{current price}$$

This formula assumes that we're calculating the `EMA` every hour, but since we're going to be calculating it every minute some modifications are needed (i.e. `n` will be much larger and alpha will be much smaller). The results will be the same. By using large windows (12/26 hours, instead of minutes), we can achieve better results with more sporadic buy/sell signals.

For comparison, we change the window for the `SMA` to 15 hours (results displayed on the frontend).

4.2 EMA Strategy

The strategy on when to buy/sell is as follows, when there's a crossover between the two EMAs:

- When the short-term EMA crosses above the long-term EMA, this indicates a bullish signal and we should buy
- When the short-term EMA crosses below the long-term EMA, this indicates a bearish signal and we should sell

4.3 MACD Indicator

The `MACD` is a trend-following momentum indicator that shows the relationship between two EMAs. In order to calculate it, all we have to do is subtract the long-term EMA from the short-term EMA:

$$\text{MACD} = \text{EMA}_{\text{short}} - \text{EMA}_{\text{long}}$$

4.4 Signal Line

The `Signal Line` is a 9-hour EMA of the `MACD`. It is used to generate buy/sell signals based on the MACD line.

4.5 Distance between MACD and Signal Line

The `Distance` between the `MACD` and the `Signal Line` is calculated as follows:

$$\text{Distance} = \text{MACD} - \text{Signal Line}$$

4.6 MACD, Signal and Distance Strategy

The strategy on when to buy/sell is as follows, when there's a crossover between the MACD and the Signal Line:

- When the MACD crosses above the Signal Line, this indicates a bullish signal and we should buy
- When the MACD crosses below the Signal Line, this indicates a bearish signal and we should sell

This is a very strong indicator, and it is used by many traders to make decisions on when to buy/sell stocks. In our case, we used the MACD(12,26,9) which is used by many traders and produces good results.

5 Web Server

In order to display the results, we will attach a REST web server to the `C++` application. This web server will have some endpoints (for example `/ema?window=100&symbol=BTC-USDT`) that will server the data. In order to safely serve this data (without exposing home IP addresses), the server is hosted on a VPS by DigitalOcean.

The frontend is created using `React` with the help of `Vite`, `Tailwind` and `Shadcn UI`. Utilizing Cloudflare's Workers (serverless platform), we can cheaply and securely host the frontend.

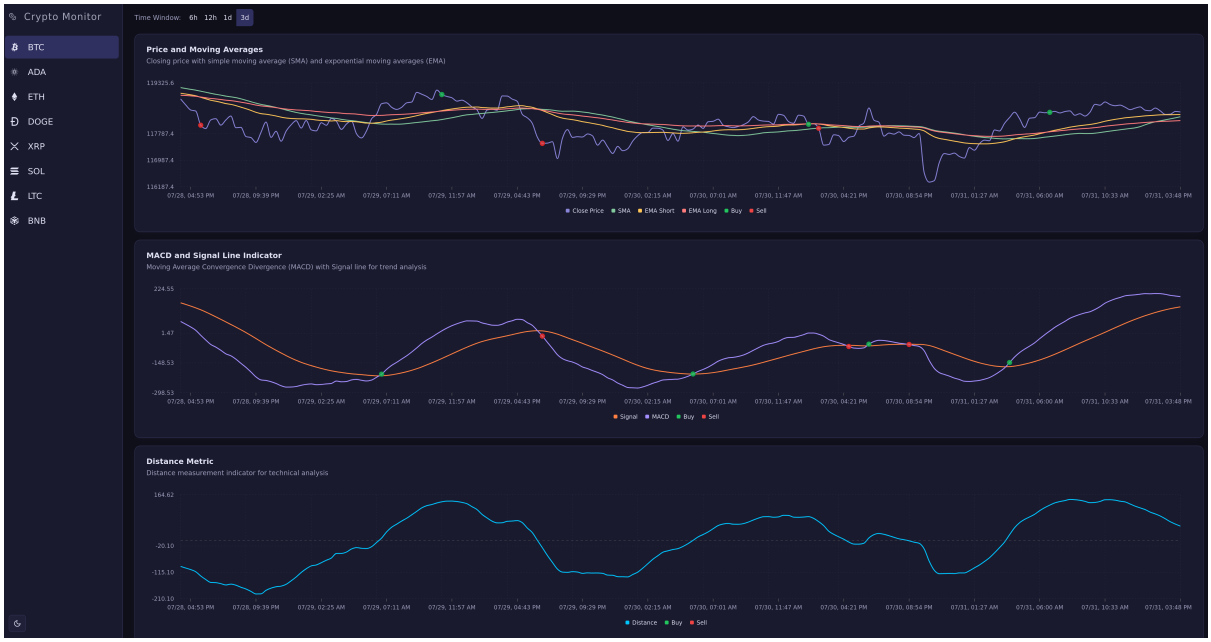

Figure 6: The frontend UI of the Crypto Monitor

6 Analyzing the predictions

6.1 Scenario

By creating a fake scenario we can make an analysis on the results of the trading strategies. For the period 28th July - 31st July at 19:00 (3 days total), we will assume that we start with **80k** USD capital, **10k** for each symbol.

- When there's a signal to `buy`, we will buy 2k worth of the symbol.
- When there is a signal to `sell`, we will sell all holdings of the symbol (if we have invested any).

6.2 Strategies Recap

There are 3 strategies in total:

- **EMA Strategy**: Buy/sell based on the EMA crossover.
- **Distance Strategy**: Buy/sell based on the Distance line crossover with $y = 0$.
- **Buy & Hold**: Buy at the start of the 3-day period (80k worth) and sell at the end of it.

6.3 Results

Table 1 shows the overall performance comparison of the three strategies over the 3-day period.

Strategy	Final Value	Total Return	Return (%)	Total Trades	Avg Trades/Symbol
Distance	$78,963.10	$-1,036.90	-1.30%	62	7.8
EMA	$79,535.80	$-464.20	-0.58%	21	2.6
Buy & Hold	$77,638.99	$-2,361.01	-2.95%	0	0.0

Table 1: Overall Strategy Performance Comparison

Even though we lost money with all strategies, the results show that the **EMA Crossover strategy performed the best** with the smallest loss of -0.58%, followed by the Distance-Based strategy at -1.30%. The passive Buy & Hold strategy had the worst performance with a -2.95% loss over the 3-day period.

The per-symbol best-strategy comparison is shown in table 2.

Symbol	Better Strategy	Difference	Distance Trades	EMA Trades
BTC-USDT	EMA	$16.99	8	5
ADA-USDT	EMA	$121.92	8	2
ETH-USDT	Distance	$3.48	8	5
DOGE-USDT	EMA	$102.73	8	2
XRP-USDT	EMA	$137.22	10	1
SOL-USDT	EMA	$141.79	8	2
LTC-USDT	Distance	$13.22	6	3
BNB-USDT	EMA	$68.74	6	1

Table 2: Per-symbol Strategy Performance Comparison

As we can see, the EMA strategy outperformed the Distance strategy in most symbols, with the exception of ETH-USDT and LTC-USDT. This indicates that we should take into account both strategies when making decisions on when to buy/sell stocks.

7 Conclusion

In this project, we successfully implemented an embedded system crypto monitor that can run on a raspberry pi continuously no matter the internet conditions, with a high efficiency. By utilizing mutexes and condition variables, we can ensure that the program is fault-tolerant and behaves correctly, as specified by the producer-consumer problem.

The trading strategies we implemented, such as the EMA and Distance strategies, show promising results in predicting the market. By utilizing them, we can make better financial decisions. Other strategies that utilize the SMA or the Pearson correlation can be used, but they might show worse results in high volatile markets like the cryptocurrency market.

8 Sources

- The web application can be accessed at `https://crypto-monitor.nontasbak.com/`
- The API for the data can be accessed at `https://api-crypto-monitor.nontasbak.com/` (the REST endpoints are `/sma`, `/ema`, `/close`, `/macd`, `/signal`, `/distance`)
- The source code can be found on the Github repository.

References

[1] Ahmet Soydemir. *Stock Price Prediction with Technical Analysis Tutorial.* Medium, 2023. `https://medium.com/@ahmettsdmr1312/stock-price-prediction-with-technical-analysis-tutorial-f0ad103cc35f`

[2] *Exponential smoothing.* Wikipedia. `https://en.wikipedia.org/wiki/Exponential_smoothing`

[3] *MACD Trading Strategy*. YouTube. https://www.youtube.com/watch?v=rf_EQvubKlk
Thanks to Github Copilot for helping with the final data analysis.