

# Ενσωματωμένα Συστήματα Πραγματικού Χρόνου

Ιωάννης Κέλπης AEM:10300

Μάρτιος 2025

## Περίληψη

Στην εργασία αυτή, υλοποιήθηκε ένα σύστημα που λαμβάνει ασύγχρονα δεδομένα συναλλαγών για 8 κρυπτονομίσματα (BTC-USDT, ADA-USDT, ETH-USDT, DOGE-USDT, XRP-USDT, SOL-USDT, LTC-USDT, BNB-USDT) από το δημόσιο κανάλι WebSocket του OKX. Το σύστημα καταγράφει τις συναλλαγές σε αρχεία, υπολογίζει τον κινούμενο μέσο όρο των τιμών και τον συνολικό όγκο συναλλαγών για τα τελευταία 15 λεπτά κάθε λεπτό, και υπολογίζει τον συντελεστή Pearson Correlation μεταξύ των κινούμενων μέσων όρων των διαφορετικών νομισμάτων. Η υλοποίηση γίνεται σε πραγματικό χρόνο και τα δεδομένα καταγράφονται σε αρχεία csv (comma seperated values) για κάθε σύμβολο.

# Κεφάλαιο 1

## Χαρακτηριστικά του συστήματος

Στα πλαίσια της εργασίας χρησιμοποίησα ένα raspberry pi 5, το οποίο λειτουργεί με το OS Debian Bookworm. Παρακάτω δίνεται μια εικόνα των χαρακτηριστικών του συστήματος μέσω του πακέτου neofetch. Η απομακρυσμένη σύνδεση στο pi γινόταν με ssh και το pi ήταν στο δίκτυο με την βοήθεια ethernet cable.

```
kelpis@raspberrypi:~ $ neofetch
,gg_,met$$$$$gg.          kelpis@raspberrypi
,g$$P"      ""Y$$.".
,$$P'           '$$.
,$$P     ,ggs.    '$$:   OS: Debian GNU/Linux 12 (bookworm) aarch64
,$$P" ,$P"'.   .    $$:  Host: Raspberry Pi 5 Model B Rev 1.0
,$$P'     d$'.   ,    $$: Kernel: 6.6.74+rpt-rpi-2712
,$$P     d$'.   ,    $$: Uptime: 1 day, 21 hours, 36 mins
,$$P:    $$-. - ,d$$'   Packages: 786 (dpkg)
,$$;    Y$b._,_d$P'   Shell: bash 5.2.15
Y$$.    `."Y$$$$$P"   Terminal: /dev/pts/0
`$$b    "-___.      CPU: (4) @ 2.400GHz
`Y$$
`Y$$. 
`$$b.
`Y$b.
`"Y$b.
`"Y$b..  Memory: 143MiB / 8052MiB
`"Y$b..
```

Figure 1.1: Neofetch characteristics

## Κεφάλαιο 2

# Υλοποίηση και εξήγηση του κώδικα

Ο κώδικας ξεκινά με την εισαγωγή των απαραίτητων βιβλιοθηκών και τον ορισμό κάποιων παραμέτρων, οι οποίες χρησιμοποιούνται για έγχρωμη εκτύπωση στην κονσόλα, καθώς και για βασικές λειτουργίες του προγράμματος (μέγιστος αριθμός νομισμάτων, ορισμός του μεγέθους του ιστορικού που κρατιέται, μέγεθος buffer κτλ.).

Στην συνέχεια ορίζονται διάφορες δομές δεδομένων για την αποθήκευση των πληροφοριών που χειρίζεται ο κώδικας. Πιο συγκεκριμένα:

- **trade\_t:** Αυτή η δομή αποθηκεύει τα δεδομένα μιας συναλλαγής, συμπεριλαμβανομένης της χρονικής στιγμής, της τιμής, του όγκου και της καθυστέρησης επεξεργασίας.
- **ma\_entry\_t:** Αυτή η δομή αποθηκεύει τα αποτελέσματα των υπολογισμών του κινούμενου μέσου όρου και του συνολικού όγκου για τα τελευταία 15 λεπτά.
- **moving\_avg\_t:** Αυτή η δομή αποθηκεύει όλα τα δεδομένα και τα αρχεία που σχετίζονται με ένα συγκεκριμένο σύμβολο.

Έπειτα αφού δημιουργηθεί directory της μορφής `data/instrument` για κάθε ένα από τα κρυπτονομίσματα, ορίζονται οι ακόλουθες συναρτήσεις, των οποίων τον ρόλο θα εξηγήσουμε συνοπτικά.

### Συνάρτηση `get_instrument`

Χρησιμοποιείται για τη δημιουργία ή ανάκτηση μιας εγγραφής για ένα συγκεκριμένο σύμβολο. Αυτή η συνάρτηση ελέγχει αν το σύμβολο υπάρχει ήδη στον πίνακα instruments. Αν δεν υπάρχει, δημιουργεί μια νέα εγγραφή, ανοίγει τα αντίστοιχα αρχεία καταγραφής και επιστρέφει έναν δείκτη προς αυτήν.

### Συνάρτηση `pearson_corr_vector`

Υπολογίζει τον συντελεστή Pearson συσχέτισης μεταξύ δύο διανυσμάτων τιμών. Η συνάρτηση χρησιμοποιείται για να μετρήσει τη γραμμική συσχέτιση μεταξύ των κινούμενων μέσων όρων δύο συμβόλων. Επιστρέφει μια τιμή μεταξύ -1 και 1, όπου 1 σημαίνει τέλεια θετική συσχέτιση και -1 τέλεια αρνητική.

### Συνάρτηση `compute_corr_thread`

Αυτή η συνάρτηση εκτελείται σε ξεχωριστό νήμα και υπολογίζει τον συντελεστή Pearson συσχέτισης μεταξύ των κινουμένων μέσων όρων ενός συμβόλου και όλων των άλλων συμβόλων. Βρίσκει τη μέγιστη συσχέτιση και καταγράφει το σύμβολο και τη χρονική στιγμή που επιτυγχάνεται αυτή η συσχέτιση. Τα αποτελέσματα αποθηκεύονται σε αρχεία CSV για κάθε σύμβολο ξεχωριστά.

### Συνάρτηση `save_trade`

Η συνάρτηση `save_trade` είναι υπεύθυνη για την ανάλυση των δεδομένων συναλλαγών που λαμβάνονται από το WebSocket του OKX και την αποθήκευσή τους στον πίνακα συναλλαγών του αντίστοιχου συμβόλου. Όταν το WebSocket λαμβάνει ένα μήνυμα σε μορφή JSON, η συνάρτηση χρησιμοποιεί τη βιβλιοθήκη jansson για να εξάγει τα πεδία της συναλλαγής, όπως την τιμή (last), τον όγκο (vol ή lastSz) και το σύμβολο (instId). Στη συνέχεια, η χρονική στιγμή της συναλλαγής καταγράφεται με ακρίβεια χρησιμοποιώντας την `clock_gettime`, και τα δεδομένα αποθηκεύονται στον πίνακα trades του αντίστοιχου

συμβόλου. Η καθυστέρηση επεξεργασίας υπολογίζεται ως η διαφορά μεταξύ της χρονικής στιγμής λήψης και της στιγμής καταγραφής της συναλλαγής. Τέλος, η συναλλαγή καταγράφεται σε ένα αρχείο CSV για το συγκεκριμένο σύμβολο, ώστε να διατηρείται ένα ιστορικό των συναλλαγών όπως ζητείται στην εκφώνηση, το οποίο θα φανεί χρήσιμο και στην δημιουργία των διαγραμμάτων. Η χρήση mutexes εξασφαλίζει ότι η πρόσβαση στα κοινόχρηστα δεδομένα γίνεται με ασφάλεια, χωρίς race conditions.

### Συνάρτηση compute\_moving\_avg\_and\_volume

Τυπολογίζει τον κινούμενο μέσο όρο των τιμών και τον συνολικό όγκο συναλλαγών για τα τελευταία 15 λεπτά. Η συνάρτηση διατρέχει τον πίνακα συναλλαγών, αιθροίζει τις τιμές και τους όγκους, και υπολογίζει τον μέσο όρο και τη μέση καθυστέρηση. Τα αποτελέσματα αποθηκεύονται σε μια δομή ma\_entry\_t.

### Συνάρτηση per\_minute\_worker

Εκτελείται κάθε λεπτό και είναι υπεύθυνη για τον υπολογισμό των κινουμένων μέσων όρων, τον υπολογισμό των συσχετίσεων και την καταγραφή των αποτελεσμάτων. Χρησιμοποιεί ακριβή χρονισμό για να εξασφαλιστεί ότι οι εργασίες εκτελούνται ακριβώς κάθε λεπτό, χωρίς ολίσθηση χρόνου.

### Συνάρτηση cput\_idle\_monitor

Παρακολουθεί το ποσοστό αδράνειας της CPU κάθε δευτερόλεπτο, διαβάζοντας τα δεδομένα από το αρχείο /proc/stat. Τα δεδομένα καταγράφονται σε ένα αρχείο CSV, ώστε να μπορεί να παρακολουθείται η φόρτωση της CPU κατά τη λειτουργία του προγράμματος. Τα δεδομένα που αποθηκεύονται θα χρησιμοποιηθούν για το διάγραμμα που ζητείται στην εκφώνηση.

### Συναρτήσεις websocket\_write\_back, ws\_service\_callback

Η συνάρτηση websocket\_write\_back χρησιμοποιείται για την αποστολή μηνυμάτων μέσω του WebSocket. Είναι υπεύθυνη για την κωδικοποίηση και την αποστολή των δεδομένων, όπως τα μηνύματα συνδρομής στα σύμβολα. Η ws\_service\_callback είναι η κύρια callback συνάρτηση του WebSocket, η οποία χειρίζεται συμβάντα όπως τη σύνδεση, τη λήψη δεδομένων και την αποσύνδεση. Όταν λαμβάνονται δεδομένα συναλλαγών, καλεί τη συνάρτηση save\_trade για την επεξεργασία και την αποθήκευσή τους.

### Συνάρτηση main

Η κύρια συνάρτηση του προγράμματος, όπου γίνεται η αρχικοποίηση του συστήματος. Δημιουργεί τα απαραίτητα αρχεία, συνδέεται με το WebSocket του OKX, και ξεκινά τα νήματα για την επεξεργασία δεδομένων και την παρακολούθηση της CPU. Η κύρια επανάληψη διαχειρίζεται τη σύνδεση και την επανασύνδεση στο WebSocket, ενώ διασφαλίζει την ομαλή λειτουργία του συστήματος μέχρι να ληφθεί σήμα τερματισμού. Άξιο αναφοράς είναι και το γεγονός ότι η main ενσωματώνει και την λειτουργία επανασύνδεσης στο WebSocket. Αν η σύνδεση χαθεί λόγω προβλημάτων στο δίκτυο, το σύστημα προσπαθεί να επανασυνδεθεί κάθε 10 δευτερόλεπτα, χωρίς να διακόπτεται η λειτουργία του. Αυτό εξασφαλίζει ότι τα δεδομένα συνεχίζουν να καταγράφονται ακόμη και μετά από προσωρινές διακοπές (πειραματικά, διαπίστωσα πως χωρίς αυτή την λειτουργία, κάθε μερικές ώρες το πρόγραμμα αποσυνδεόταν και σταματούσε η απόκτηση των δεδομένων μέσω του websocket). Με αυτόν τον τρόπο το πρόγραμμα είναι σε θέση να τρέχει για πολλές μέρες χωρίς να αντιμετωπίσει κάποιο πρόβλημα, όπως φαίνεται και από τα αποτελέσματα που έχουμε από την εκτέλεση του προγράμματος για πάνω από 48 ώρες!

Τέλος, για τη μεταγλώττιση του προγράμματος για το Raspberry Pi 5 (ARM64 αρχιτεκτονική) από x86\_64 σύστημα, χρησιμοποιήθηκε cross-compilation με το εργαλείο aarch64-linux-gnu-gcc μέσα από το repository που μας δόθηκε στις οδηγίες του μαθήματος (συγκεκριμένα χρησιμοποιήθηκε το tarball cross-gcc-14.2.0-pi\_64cross-gcc-14.2.0-pi\_64.tar.gz. Η διαδικασία απαιτεί την εγκατάσταση ενός sysroot που περιέχει τις βιβλιοθήκες και τις κεφαλίδες της ARM64, καθώς και την τροποποίηση των flags μεταγλώτισης για να επιτευχθεί συμβατότητα.

Με την βοήθεια του cross-compilation επιτυγχάνεται η ανάπτυξη σε έναν ισχυρό x86\_64 υπολογιστή, ενώ το εκτελέσιμο αρχείο τρέχει στο Raspberry Pi χωρίς απαίτηση για local compilation. Αυτό εξουκονομεί χρόνο και πόρους, ειδικά σε embedded συστήματα με περιορισμένη υπολογιστική ισχύ. Για να λειτουργήσει σωστά, το sysroot πρέπει να ταιριάζει ακριβώς με τις εκδόσεις των βιβλιοθηκών στο Raspberry Pi (π.χ., Debian Bookworm). Παρακάτω φαίνεται η εντολή που χρησιμοποιήθηκε προκειμένου να παραχθεί το executable αρχείο που τρέχει στο Raspberry Pi.

```
aarch64-linux-gnu-gcc -sysroot=/opt/arm64-sysroot \
-B/opt/arm64-sysroot/usr/lib/aarch64-linux-gnu \
-I/opt/arm64-sysroot/usr/include \
-I/opt/arm64-sysroot/usr/include/aarch64-linux-gnu \
-L/opt/arm64-sysroot/usr/lib/aarch64-linux-gnu \
-Wl,-rpath-link=/opt/arm64-sysroot/usr/lib/aarch64-linux-gnu \
-o okx_client okx.c -lwebsockets -lz -lssl -lcrypto -ljansson -lpthread -lm -lcap -ldl
```

- **-sysroot=/opt/arm64-sysroot:** Καθορίζει τη ρίζα του συστήματος (sysroot) που περιλαμβάνει τις απαραίτητες βιβλιοθήκες (π.χ., libwebsockets, jansson) και κεφαλίδες για ARM64.
- **-B, -I, -L:** Ορίζουν τα paths για τις βιβλιοθήκες (-L), τις κεφαλίδες (-I), και τα binaries (-B) της ARM64.
- **-Wl,-rpath-link:** Διασφαλίζει ότι ο linker θα βρει τις κοινόχροντες βιβλιοθήκες κατά την εκτέλεση.
- **Linking:** Συνδέονται οι βιβλιοθήκες -lwebsockets, -ljansson, -lpthread, κ.λπ., οι οποίες πρέπει να είναι compiled για ARM64 στο sysroot.

Στην εικόνα φαίνεται μέσα από το terminal το executable αρχείο που παράγεται σε αρχιτεκτονική ARM64.

```
john@john:~/Documents/SXOLI/8o_EJAMINO/Embedded$ file okx_client
okx_client: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter
/lib/ld-linux-aarch64.so.1, for GNU/Linux 3.7.0, with debug_info, not stripped
```

Figure 2.1: Arm executable

# Κεφάλαιο 3

## Διαγράμματα

Τα παρακάτω διαγράμματα αποτελούν την γραφική αναπαράσταση των τιμών που συλλέχθηκαν στα αρχεία .csv μέσα σε ένα διάστημα περίπου 50 ωρών. Τα αρχεία python που χρησιμοποιήθηκαν για την παραγωγή τους βρίσκονται στο repository του github. Προφανώς, προκειμένου να είναι τα διαγράμματα ευανάγνωστα, τα δεδομένα που είχαν αποθηκευτεί έχουν down sampling, το οποίο μπορεί εύκολα να ρυθμιστεί μέσα από τον κώδικα python. Στα αρχεία .csv παραμένει ωστόσο όλη η πληροφορία. Το αρχείο κώδικα `diagrams_instruments.py` δημιουργεί έναν φάκελο `plots`, μέσα στον οποίο τοποθετούνται σε υποφακέλους (ανάλογα με το όνομα του `instrument`) τα διαγράμματα που αφορούν το κάθε κρυπτονόμισμα. Ενδεικτικά δίνονται παρακάτω τα διαγράμματα για το Bitcoin και το Ethereum. Τα διαγράμματα για τον επεξεργαστή και για τις χρονικές διαφορές δίνονται από τα αρχεία `diagram_cpu.py` και `diagram_timing.py` αντίστοιχα.

### 3.1 Επεξεργαστής, αρχείο `diagrams_cpu.py`

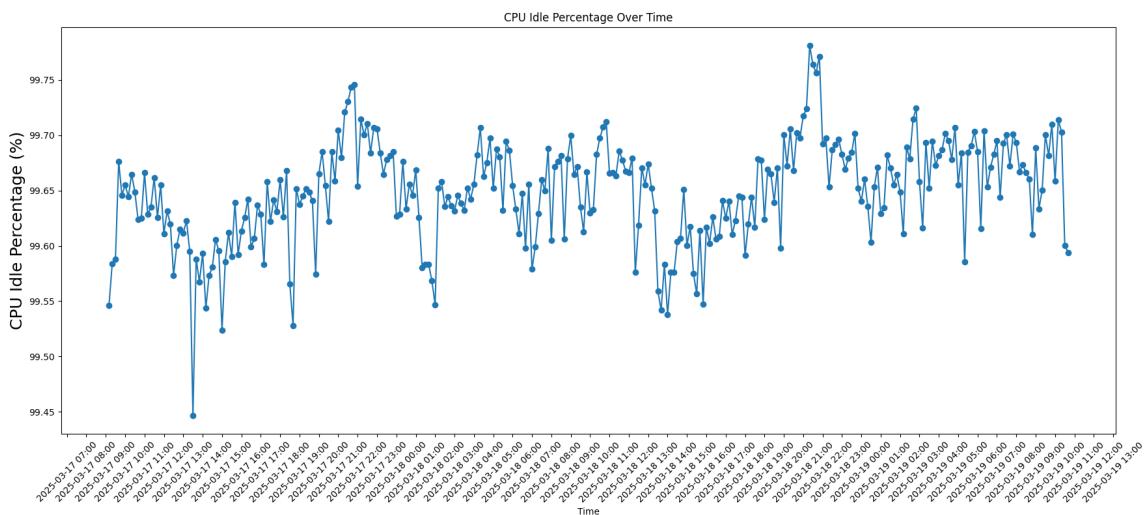


Figure 3.1: CPU Idle percentage over 48 hour span

Όπως φαίνεται από το διάγραμμα, ο επεξεργαστής είναι για ένα πολύ μεγάλο ποσοστό του χρόνου σε κατάσταση `idle`, γεγονός που σημαίνει ότι το πρόγραμμα δεν καταναλώνει σημαντικούς πόρους και μπορεί να τρέξει για μεγάλο χρονικό διάστημα χωρίς να επηρεάζει την απόδοση του Raspberry Pi.

## 3.2 Timing Difference

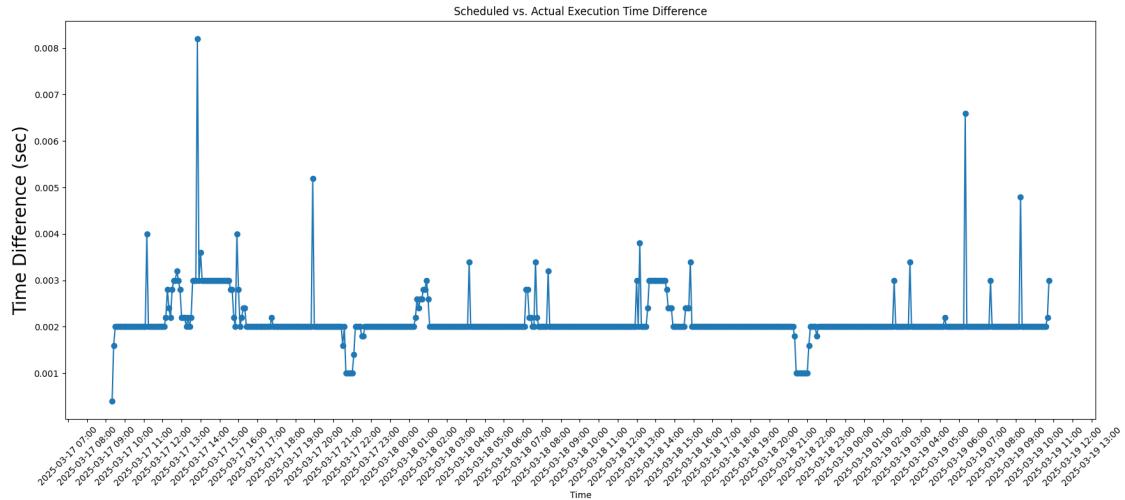


Figure 3.2: Timing differences over 48 hour span

## 3.3 Moving Average of Instruments

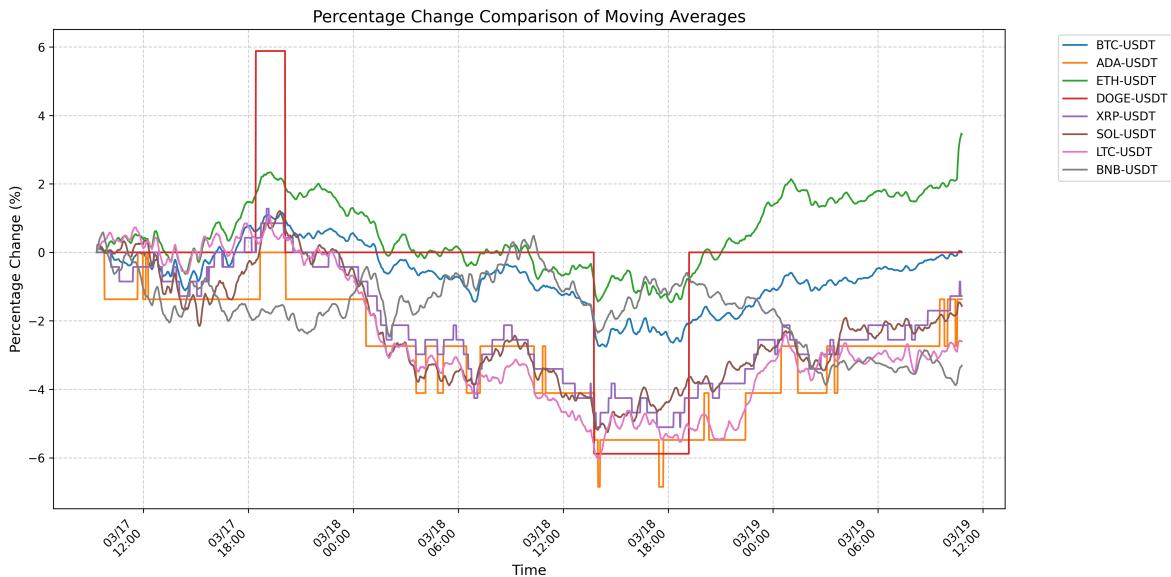


Figure 3.3: Enter Caption

Οι καμπυλές μερικών από των νομισμάτων εμφανίζονται σε τετραγωνική μορφή, διοτι δεν είχα αρκετά μεγάλη ακρίβεια δεκαδικών ψηφίων όταν έγινε η συλλογή δεδομένων, με αποτέλεσμα να μην παρατηρείται αλλαγή για μικρές μεταβολές.

Το αρχείο `timings.csv` καταγράφει την διαφορά μεταξύ προγραμματισμένων και πραγματικών χρόνων εκτέλεσης των εργασιών (π.χ., υπολογισμοί ανά λεπτό). Στο διάγραμμα φαίνεται από τις πολύ μικρές τιμές των καθυστερήσεων το γεγονός πως το σύστημα είναι αξιόπιστα συγχρονισμένο και δεν συσσωρεύονται καθυστερήσεις, κάτι απαραίτητο για συνεχή λειτουργία.

### 3.4 Total volume of instruments

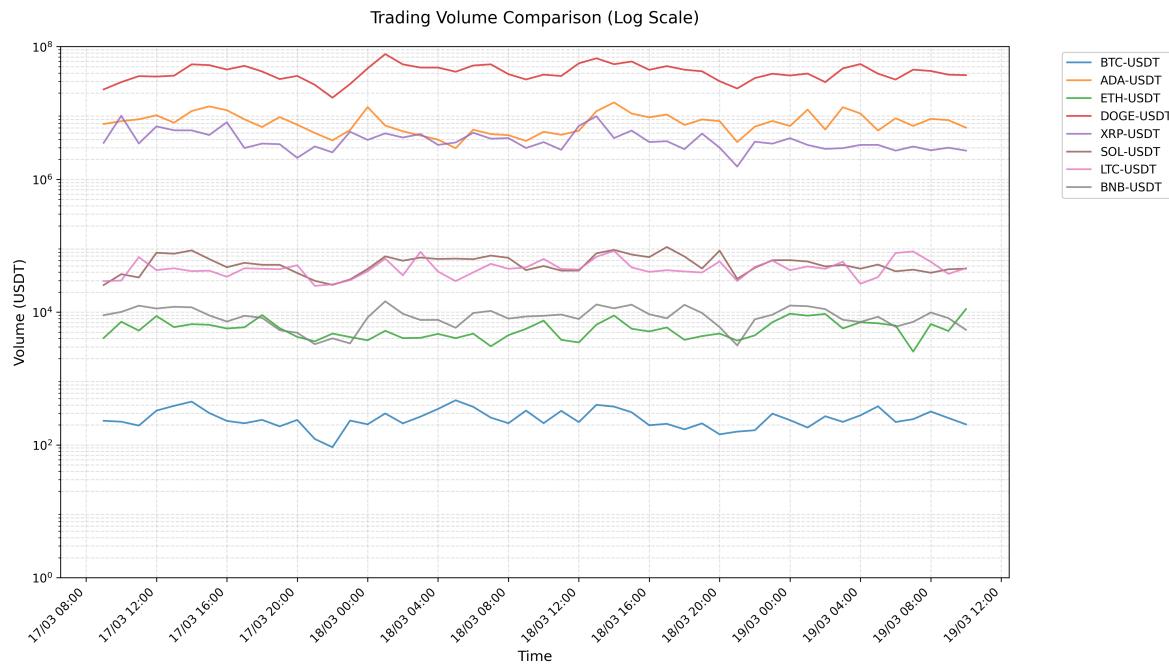


Figure 3.4: Volume Comparison

### 3.5 Diagrams per Instrument

Δίνονται ενδεικτικά όπως σημειώθηκε νωρίτερα, τα 3 διαγράμματα για τα instruments BTC και ETH. Τα διαγράμματα αυτά για κάθε σύμβολο παρέχουν μια ολοκληρωμένη ανάλυση της συμπεριφοράς και των σχέσεων του, πιο συγκεκριμένα:

1. Διάγραμμα Transactions: Καταγράφει τις τιμές συναλλαγών σε πραγματικό χρόνο, δείχνοντας τις διακυμάνσεις της αγοράς. Αποκαλύπτει πότε έγιναν οι πιο έντονες κινήσεις και σε ποιο εύρος τιμών.
2. Διάγραμμα Moving Average: Εμφανίζει τον 15-λεπτο κινούμενο μέσο όρο τιμών, εξομαλύνοντας τις βραχυπρόθεσμες διακυμάνσεις. Αυτό βοηθά στον εντοπισμό μακροπρόθεσμων τάσεων (π.χ., ανοδική/πτωτική πορεία).
3. Διάγραμμα Correlation: Δείχνει τον συντελεστή Pearson συσχέτισης με άλλα νομίσματα. Τιμές κοντά στο 1 ή -1 υποδηλώνουν ισχυρή θετική ή αρνητική συσχέτιση, αντίστοιχα, αποκαλύπτοντας πώς το νόμισμα αντιδρά σε αλλαγές άλλων.

# BITCOIN

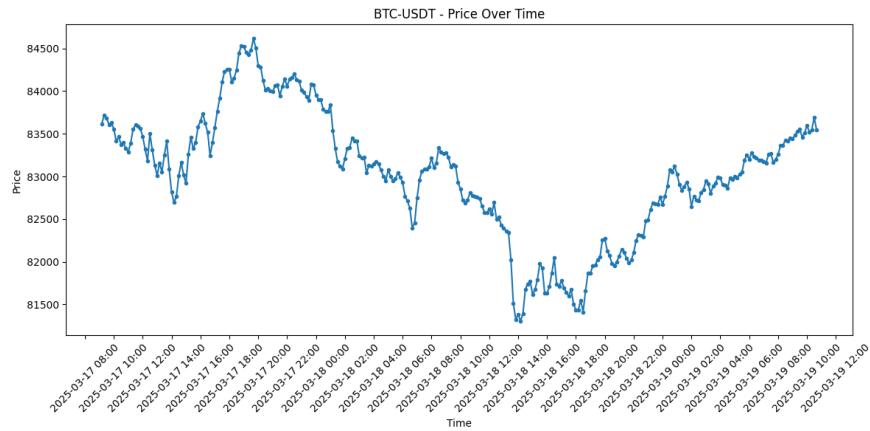


Figure 3.5: Transactions of BTC

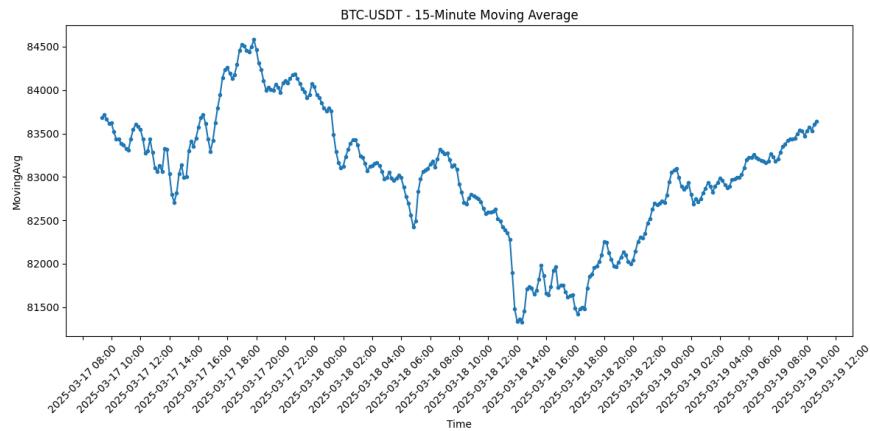


Figure 3.6: Moving Average of BTC

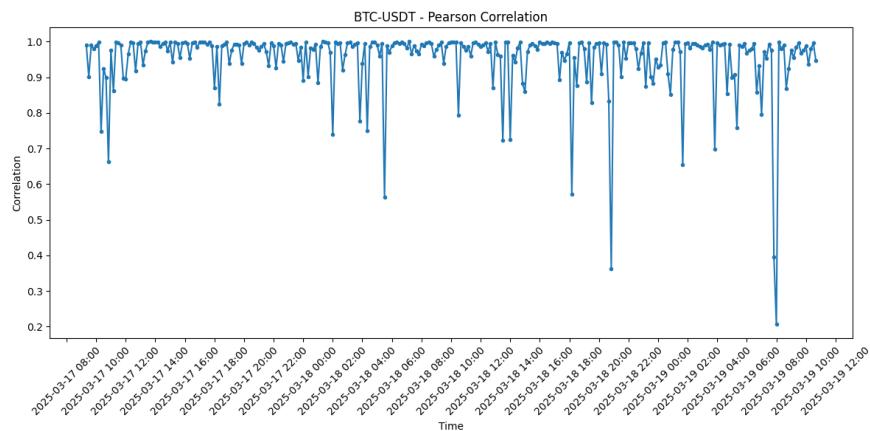


Figure 3.7: Correlation of BTC

# ETHEREUM

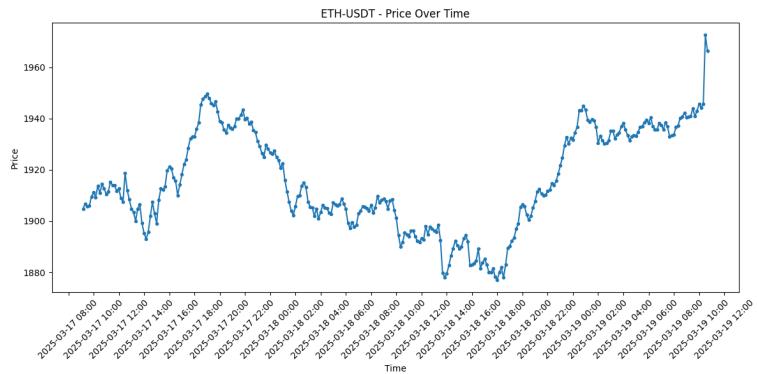


Figure 3.8: Transactions of ETH

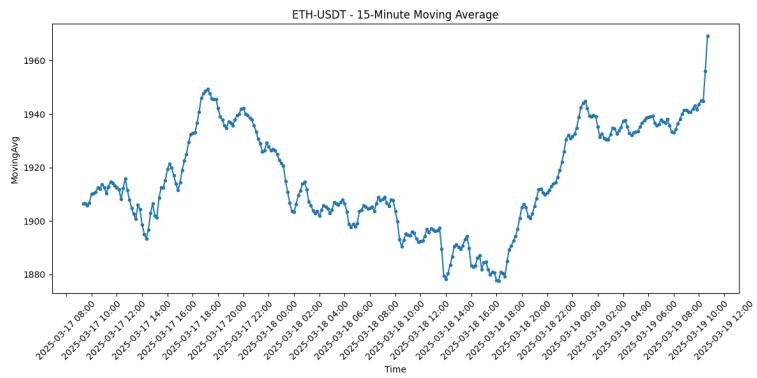


Figure 3.9: Moving Average of ETH

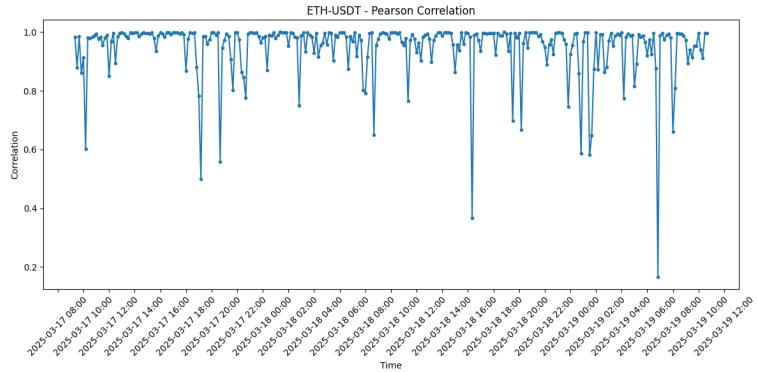


Figure 3.10: Correlation of ETH

## Διαγράμματα Συσχέτισης

Τα παρακάτω διαγράμματα δείχνουν τη συσχέτιση και τη χρονική υστέρηση (time lag) μεταξύ ζευγών κρυπτονομισμάτων. Φαίνεται δηλαδή πως κινούνται ”σε συντονισμό” δύο κρυπτονομίσματα (π.χ. BTC-ETH) με τον χρόνο, αλλά με πιθανή χρονική καθυστέρηση (ένα νόμισμα να ακολουθεί το άλλο μετά από λίγα λεπτά). Η χρωματική κλίμακα δείχνει το βέλτιστο χρονικό κενό (lag) σε λεπτά που μεγιστοποιεί τη συσχέτιση. Δίνονται ενδεικτικά τρία διαγράμματα (στο repository υπάρχουν τα plots για όλα τα πιθανά ζευγάρια νομισμάτων).

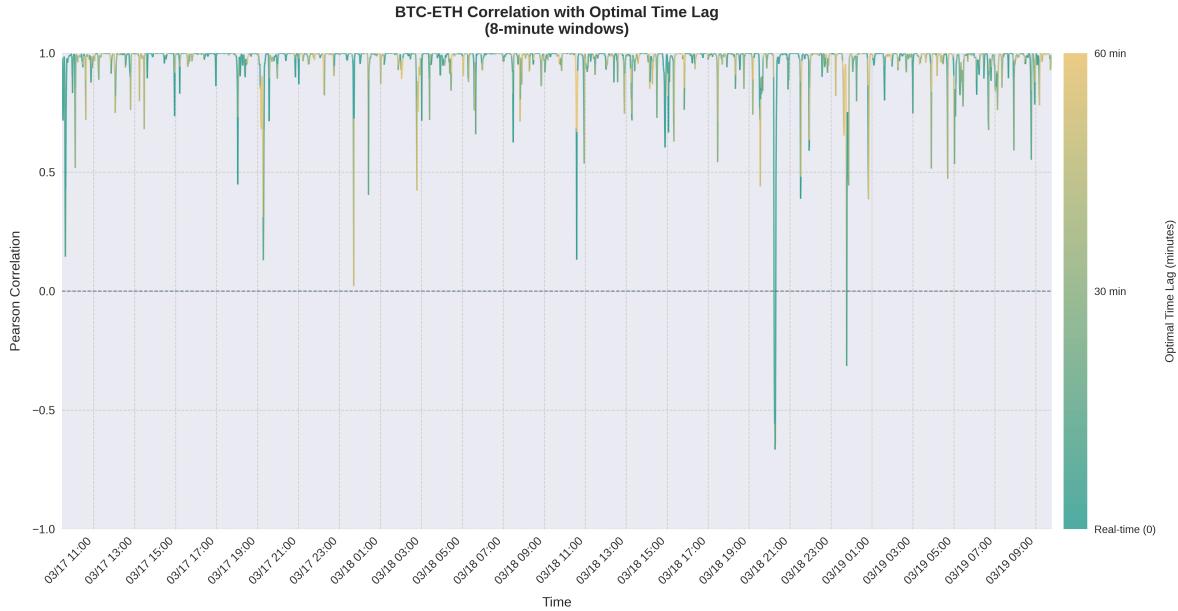


Figure 3.11: BTC - ETH correlation

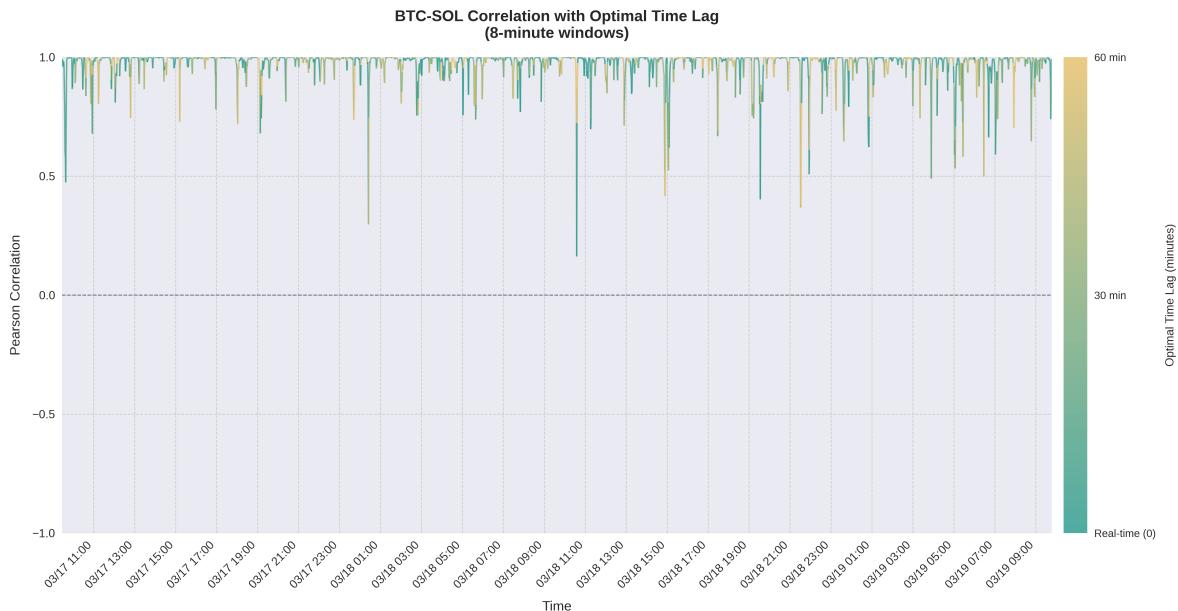


Figure 3.12: BTC - SOL correlation

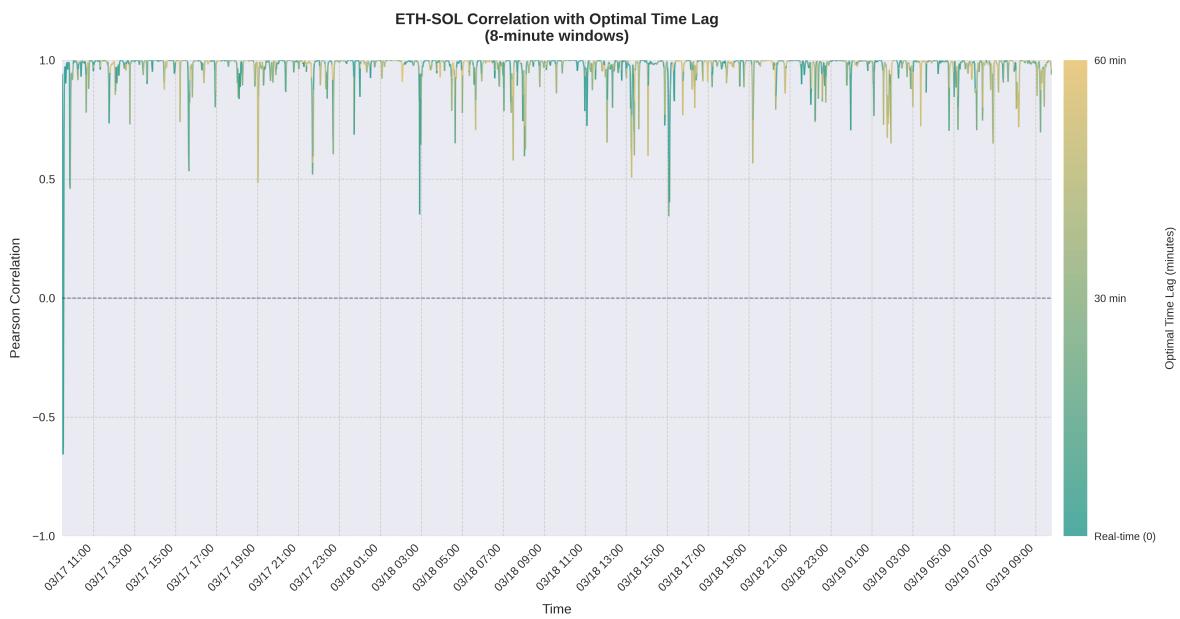


Figure 3.13: ETH - SOL correlation

**Σημείωση:** Όλοι οι κώδικες και τα αρχεία που χρησιμοποιήθηκαν και παράχθηκαν κατά τη διάρκεια της εργασίας μπορούν να βρεθούν στο παρακάτω repository στο github: [https://github.com/gianini10/Embedded\\_ece](https://github.com/gianini10/Embedded_ece)