ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ.


# ΤΙΤΛΟΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ :

## *Σχεδιασμός και υλοποίηση διάταξης αναλυτή φάσματος*

Πτυχιακή Εργασία που εκπονήθηκε στο πλαίσιο του Προγράμματος του ΤΕΙ, του Τμήματος Μηχανικών Πληροφορικής Τ.Ε.

ΤΣΟΛΑΚΟΥ ΑΡΧΙΜΗΔΗΣ ΚΕΒΙΝ

ΕΠΙΒΛ. ΚΑΘΗΓΗΤΗΣ : ΒΟΥΡΒΟΥΛΑΚΗΣ ΙΩΑΝΝΗΣ

# Πίνακας περιεχομένων

**Η Πτυχιακή**

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδας.

# Περίληψη

Αρχικά δίνεται μια σύντομη περιγραφή στους αναλυτές φάσματος. Στη συνέχεια αναφερόμαστε στον αναλυτή που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας. Παρουσιάζονται λεπτομέρειες της κατασκευής του, Αναφέρονται όλα τα βήματα που ακολουθήθηκαν για την υλοποίηση του και οι δυσκολίες που αντιμετωπίστηκαν. Επίσης, αναλύουμε τα υλικά κατασκευής και το λογισμικό του αναλυτή φάσματος και περιγράφουμε πως δουλεύει και τι πληροφορία μας παρέχει το καθένα. Δημιουργήθηκε σχηματικό με το κύκλωμα που θα έχει η πλακέτα το όποιο περιλαμβάνει: Την πλακέτα CYBLE_PSoC6_416045 με τον μικροελεγκτή PSoC6, τον σταθεροποιητή τάσης BU33TD3WG-TR και το ολοκληρωμένο κύκλωμα FT234XD-R που πραγματοποιεί μετατροπή σειριακού interface σε USB. Στην συνέχεια σχεδιάστηκε το τυπωμένο κύκλωμα της πλακέτας με το πρόγραμμα kicad 7.0. Έπειτα επιλέχθηκαν όλα τα απαραίτητα στοιχεία, όπως πυκνωτές και αντιστάσεις σύμφωνα με της απαιτήσεις της κάθε μονάδος. Στη συνέχεια στάλθηκαν τα gerber files σε εργοστάσιο για την κατασκευή της πλακέτας. Μετά την κατασκευή της πλακέτας έγινε η συναρμολόγηση πραγματοποιώντας τη συγκόλληση των εξαρτημάτων. Έχοντας το υλικό στη διάθεσή μας ακολούθησε η ανάπτυξη του firmware για τον μικροελεγκτή PSoC6 χρησιμοποιώντας το IDE PSoC Creator. Το firmware υλοποίησε τις εξής λειτουργίες:

- Ανάγνωση αναλογικού σήματος και μετατροπή σε ψηφιακό από τον ADC του μικροελεγκτή μέχρι τη συμπλήρωση 1024 δειγμάτων.
- Εφαρμογή αλγορίθμου CooleyTuckey για τον υπολογισμό του FFT πάνω στα δείγματα του σήματος που διαβάστηκε.
- Αποστολή του FFT στον υπολογιστή μέσω USB χρησιμοποιώντας την UART του μικροελεγκτή και τον εξωτερικό μετατροπέας USB-to-serial (FT234XD-R).
- Για σκοπούς ελέγχου καλής λειτουργίας και αποσφαλμάτωσης υλοποιήθηκε στον μικροελεγκτή η δημιουργία τεσσάρων σημάτων. Ένα ημιτονοειδές σήμα, ένα τετραγωνικό σήμα, ένα σήμα τριγωνικό και ένα σήμα παλμού με DutyCycle 20%, Η δημιουργία των σημάτων πραγματοποιήθηκε φορτώνοντας

τα δείγματα μιας περιόδου σε πίνακα εσωτερικά στον μικροελεγκτή και μεταφορά τους με χρήση DMA σε εσωτερικό DAC ώστε να εμφανίζονται σε έναν ακροδέκτη του μικροελεγκτή ο οποίος μπορεί με εξωτερική σύνδεση να οδηγηθεί στην αναλογική είσοδο που πραγματοποιεί FFT.

Τέλος, αναπτύχθηκε λογισμικό σε Python για τη λήψη των δεδομένων που στέλνονται στον υπολογιστή καθώς επίσης και για την αναπαράστασή τους σε γράφημα.

## Summary of the Bachelor thesis

First a brief description of spectrum analyzers is given. Next, we refer to the spectrum analyzer developed in the context of this thesis. Details of its construction are presented, all the steps followed for its implementation and the difficulties encountered are mentioned. We also analyze the components used for the implementation and the software of the spectrum analyzer, and we describe how it works and what information each component provides us. A schematic design was created for the circuit which includes: The CYBLE_PSoC6_416045 board with the PSoC6 microcontroller, the BU33TD3WG-TR voltage regulator and the FT234XD-R integrated circuit that converts a serial interface to USB. Then the printed circuit of the board was designed with the kicad 7.0 program. Then all the necessary components, such as capacitors and resistors, were selected according to the requirements of each unit. The gerber files were sent to a factory to manufacture the board. After the construction of the board, the assembly was done by soldering the parts. Having the hardware available, we proceeded to develop the firmware for the PSoC6 microcontroller using the PSoC Creator IDE. The firmware implemented the following functions:

- Read analog signal and convert to digital by the ADC until 1024 samples are completed.
- Use of Cooley Tuckey algorithm to calculate the FFT on the samples of the signal.
- Send the FFT to the PC via USB using the microcontroller's UART and the external USB-to-serial converter (FT234XD-R).

- For the purposes of checking functionality and debugging, four signals were implemented inside microcontroller. A sine signal, a square signal, a triangle signal and a pulse signal with a Duty Cycle of 20%, The generation of the signals was done by loading the samples of one period into an array internal to the microcontroller and transferring them using DMA to an internal DAC for display on a pin which can be externally connected to the analog input which performs FFT.

Finally, software was developed in Python to receive the data sent to the computer as well as to present it in a graph.

# Κεφάλαιο 1 – Εισαγωγή

## 1.1 Αντικείμενο της Πτυχιακής

Αντικείμενο της πτυχιακής εργασίας αυτής είναι ο σχεδιασμός και υλοποίηση διάταξης αναλυτή φάσματος.

## 1.2 Σκοπός της Πτυχιακής

Σκοπός της εργασίας αυτής είναι η κατανόηση σχεδιασμός και υλοποίηση διάταξης αναλυτή φάσματος και η ανάπτυξη διάταξης αναλυτή φάσματος για σήματα χαμηλών

συχνοτήτων. Η διάταξη θα βασιστεί στον μικροελεγκτή/System-on-Chip CYBLE_PSoC6_416045. Θα πραγματοποιείται δειγματοληψία από αναλογικά κανάλια εισόδου και στη συνέχεια θα εφαρμόζεται γρήγορος μετασχηματισμός Fourier (FFT). Για την υλοποίηση του FFT θα χρησιμοποιηθεί ο αλγόριθμος Cooley-Tuckey. Το φάσμα συχνοτήτων θα αποστέλλεται σε υπολογιστή για την απεικόνιση και την επαλήθευση της λειτουργίας. Η συνδεσιμότητα με τον υπολογιστή θα υλοποιηθεί ενσύρματα με USB. Θα σχεδιαστεί το σχηματικό και το τυπωμένο κύκλωμα της διάταξης καθώς επίσης θα πραγματοποιηθεί και συναρμολόγηση της πλακέτας.

## 1.3 Τι είναι Αναλυτής Φάσματος

Αναλυτής Φάσματος ονομάζεται η συσκευή η οποία απεικονίζει την ένταση των συχνοτήτων που περιλαμβάνονται σε ένα σήμα.

Ένας αναλυτής φάσματος είναι ένα εργαλείο που χρησιμοποιείται για την ανίχνευση του αρμονικού περιεχομένου που παρατηρείται σε ένα συγκεκριμένο εύρος συχνοτήτων. Ο σχεδιασμός και η υλοποίηση ενός αναλυτή φάσματος απαιτεί κατανόηση των αρχών της επεξεργασίας σημάτων και των αντίστοιχων ηλεκτρονικών κυκλωμάτων.

Οι βασικές λειτουργίες ενός αναλυτή φάσματος περιλαμβάνουν τη λήψη ενός σήματος, τη μετατροπή του σήματος από το πεδίο του χρόνου στο πεδίο της συχνότητας (μετασχηματισμός Fourier) και την αναπαράσταση του φάσματος. Η συχνότητα εμφανίζεται στον οριζόντιο άξονα, ενώ η ένταση του σήματος εμφανίζεται στον κατακόρυφο άξονα με τη μορφή κατακόρυφης γραμμής.

Ο αναλυτής φάσματος βοηθά ώστε να εντοπιστούν σφάλματα-δυσλειτουργίες κτλ για μια συσκευή. Για να μελετήσουμε το φασματικό περιεχόμενο των ηλεκτρικών σημάτων με σκοπό να δούμε τι ακριβώς συμβαίνει σε αυτά καθώς διέρχονται μέσα από μία ηλεκτρική συσκευή, χωρίς να επηρεάζουμε με οποιοδήποτε τρόπο το σήμα, ο μετασχηματισμός Fourier μας δίνει πληροφορίες για το σήμα όπως, πλάτος, ισχύ, περίοδο, πλευρικές μπάντες και συχνότητα, δίνοντάς μας έτσι μια καθαρή και ακριβή εικόνα του φάσματος στο πεδίο της συχνότητας. Ανάλογα με την εφαρμογή, ένα σήμα μπορεί να έχει διαφορετικά χαρακτηριστικά.

## 1.4 Μετασχηματισμός Fourier

Οι μέθοδοι που χρησιμοποιούνται για την συχνοτική ανάλυση είναι ο Μετασχηματισμός Fourier (Fourier transform) και η τεχνική swept-tuned. Η πρώτη μέθοδος χρησιμοποιεί ένα σήμα στο πεδίο του χρόνου, το ψηφιοποιεί με δειγματοληψία, εκτελεί τα μαθηματικά που χρειάζονται για να μετατραπεί στο πεδίο της συχνότητας και τέλος απεικονίζει το αποτέλεσμα. Εν ολίγοις, λαμβάνει την πληροφορία στο πεδίο του χρόνου που περιέχει την απαραίτητη συχνοτική πληροφορία. Με την ικανότητα του για ανάλυση σήματος σε πραγματικό χρόνο, ο αναλυτής Fourier μπορεί να συλλαμβάνει περιοδικά καθώς και τυχαία και παροδικά συμβάντα. Μπορεί ακόμα να αποδώσει σημαντική βελτίωση ταχύτητας σε σύγκριση με παραδοσιακούς αναλυτές σάρωσης (swept analyzer) και μπορεί να μετρήσει το πλάτος. Ωστόσο έχει κάποιους περιορισμούς, ιδιαίτερα στο εύρος συχνότητας.

## 1.5 Μετασχηματισμός Fourier (DFT)

Ο Μετασχηματισμός Fourier, γνωστός και ως Διακριτός Μετασχηματισμός Fourier (Discrete Fourier Transform - DFT), είναι μια μαθηματική τεχνική που χρησιμοποιείται για τον αναλυτικό προσδιορισμό ενός σήματος στο πεδίο της συχνότητας. Αυτός ο μετασχηματισμός είναι εξαιρετικά χρήσιμος για την ανάλυση σημάτων σε εφαρμογές όπως η επεξεργασία σήματος, η αναγνώριση προτύπων, η συμπίεση σημάτων και πολλά άλλα.

Ο DFT μετασχηματίζει ένα διακριτό σήμα από το πεδίο του χρόνου στο πεδίο της συχνότητας. Με άλλα λόγια, αν το αρχικό σήμα είναι μια ακολουθία από διακριτά δείγματα, ο DFT το μετατρέπει σε ένα σύνολο συνιστωσών συχνότητας.

Η μαθηματική του έκφραση είναι η εξής:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn}$$

Όπου:

- $X(k)$ είναι η κ-οστή συνιστώσα της συχνότητας.
- $x(n)$ είναι το δείγμα του σήματος στην n-οστή χρονική θέση.
- $N$ είναι το πλήθος των δειγμάτων στο σήμα.

- $k$ είναι ο δείκτης της συχνότητας.

Ο DFT παρέχει έναν τρόπο για να αναλύσουμε ένα σήμα στις συχνότητες που το αποτελούν. Η αντίστροφη διαδικασία, γνωστή ως Αντίστροφος Διακριτός Μετασχηματισμός Fourier (Inverse DFT - IDFT), επιτρέπει την ανακατασκευή του αρχικού σήματος από τις συχνότητες. Συνήθως, για πρακτικούς υπολογισμούς, χρησιμοποιείται η γρηγορότερη έκδοση του DFT, γνωστή ως FFT (Fast Fourier Transform).

## 1.6 Γρήγορος μετασχηματισμός Fourier

Ο Γρήγορος Μετασχηματισμός Fourier (Fast Fourier Transform - FFT) είναι ένας αλγόριθμος που χρησιμοποιείται για τον υπολογισμό του Διακριτού Μετασχηματισμού Fourier (DFT) με πολύ γρηγορότερο τρόπο από την κανονική μέθοδο. Ο FFT έχει καθοριστική σημασία στον χώρο της επεξεργασίας σημάτων και των επιστημών της πληροφορίας, καθώς επιτρέπει τον αποτελεσματικό υπολογισμό των συνιστωσών συχνότητας ενός σήματος.

Η βασική ιδέα του FFT είναι να εκμεταλλευτεί την συμμετρία και τις επαναλαμβανόμενες ιδιότητες των εκθετικών παραγόντων που εμφανίζονται στην εξίσωση του DFT. Αντί να υπολογίζει τα πάντα από την αρχή, ο FFT διασπάει τον DFT σε διάφορα στάδια, εκμεταλλευόμενος τις συμμετρίες, και μειώνει δραστικά τον αριθμό των υπολογισμών.

Ο FFT χρησιμοποιείται ευρέως σε εφαρμογές όπως η επεξεργασία σημάτων, η ανάλυση φάσματος, η συμπίεση δεδομένων, η επεξεργασία εικόνας, και άλλα πεδία όπου η ανάλυση σημάτων στο πεδίο της συχνότητας είναι σημαντική. Ο FFT έχει διάφορες παραλλαγές, όπως ο Cooley-Tukey FFT, που είναι από τους πιο γνωστούς αλγορίθμους FFT.

# Κεφάλαιο 2 – Υλικά κατασκευής

## 2.1 Πυκνωτές

Ένας πυκνωτής είναι ένα ηλεκτρονικό εξάρτημα που αποθηκεύει ηλεκτρικό φορτίο. Αποτελείται από δύο αγωγούς, τους οποίους διαχωρίζει μονωτικό υλικό, γνωστό και

ως διηλεκτρικό, που βρίσκεται ανάμεσα σε αυτούς. Ο ένας αγωγός είναι θετικός και ονομάζεται "πυκνωτική πλάκα", ενώ ο άλλος είναι αρνητικός και ονομάζεται "αντιπυκνωτική πλάκα".

Ο πυκνωτής λειτουργεί με το να αποθηκεύει φορτίο όταν εφαρμόζεται τάση μεταξύ των δύο πλακών. Όσο μεγαλύτερη είναι η τάση που εφαρμόζεται, τόσο περισσότερο φορτίο αποθηκεύεται. Η χωρητικότητα (η δυνατότητα αποθήκευσης φορτίου) εξαρτάται από την επιφάνεια των πλακών, την απόσταση μεταξύ τους και το υλικό του διηλεκτρικού.

Οι πυκνωτές χρησιμοποιούνται σε πληθώρα εφαρμογών, όπως τροφοδοτικά, φίλτρα, απομονωτές σημάτων, αποθήκευση ενέργειας κ.ά.

## 2.2 Είδη πυκνωτών

Υπάρχουν πολλά είδη πυκνωτών, κάθε ένα από τα οποία έχει διαφορετικές ιδιότητες και χρήσεις. Ορισμένοι από τους κύριους τύπους πυκνωτών:

**Πυκνωτές Κεραμικοί:** Μικροί σε μέγεθος και χαμηλού κόστους, κατάλληλοι για υψηλές συχνότητες.

**Πυκνωτές Ηλεκτρολυτικοί:** Χρησιμοποιούνται για υψηλή χωρητικότητα και συνήθως έχουν μεγαλύτερο μέγεθος. Οι δύο κύριοι τύποι είναι οι Πυκνωτές Αλουμινίου και οι Πυκνωτές Τανταλίου και συνήθως έχουν πολικότητα.

**Πυκνωτές Πολυεστερικοί (Polyester):** Κατάλληλοι για εφαρμογές μεσαίας συχνότητας, μερικές φορές χρησιμοποιούνται για φίλτρα.

**Πυκνωτές Κεραμικοί με Υψηλή Χωρητικότητα (MLCC):** Υψηλής χωρητικότητας πυκνωτές κεραμικοί.

**Πυκνωτές Πολυπροπυλενίου (PP):** Χρησιμοποιούνται για εφαρμογές υψηλής συχνότητας και υψηλής ακρίβειας.

Αυτοί είναι μερικοί μόνο από τους τύπους πυκνωτών που υπάρχουν, και κάθε ένας έχει διαφορετικές χρήσεις, πλεονεκτήματα και περιορισμούς.

## 2.3 Διακόπτης τύπου push-button

Στο πλαίσιο των ηλεκτρονικών κυκλωμάτων, ένας διακόπτης τύπου push-button αναφέρεται συνήθως σε ένα ηλεκτρονικό στοιχείο που χρησιμοποιείται για τη διακοπή ή σύνδεση του ηλεκτρικού ρεύματος (ή τάσης) σε ένα κύκλωμα.

## 2.4 Αντιστάσεις

Ο αντιστάτης (ή αντίσταση) είναι ένα βασικό ηλεκτρονικό εξάρτημα που προσφέρει αντίσταση στην ροή του ηλεκτρικού ρεύματος. Η αντίσταση είναι ένα μέτρο της δυσκολίας που αντιμετωπίζει το ηλεκτρικό ρεύμα κατά τη διάρκεια της διέλευσής του μέσα από τον αντιστάτη. Η μονάδα μέτρησης της αντίστασης είναι το ohm (Ω).

**Τύποι Αντιστάτη:**

Υπάρχουν διάφοροι τύποι αντιστάτη, καθένας με διαφορετικά χαρακτηριστικά και χρήσεις. Οι βασικοί τύποι περιλαμβάνουν:

**Σταθερή Αντίσταση (Fixed Resistor):**

Ο πιο βασικός τύπος, έχει σταθερή αντίσταση και συνήθως χρησιμοποιείται για τον έλεγχο του ρεύματος ή της τάσης.

**ΜεταβλητήΑντίσταση (Variable Resistor - Potentiometer):**

Επιτρέπει τη ρύθμιση της αντίστασης μεταξύ δύο σημείων.

**Αντίσταση με Στρώμα Οξείδωσης (Varistor):**

Αλλάζει την αντίστασή του ανάλογα με την τάση που εφαρμόζεται. Χρησιμοποιείται για προστασία από υπερτάσεις.

**Θερμοστάτης (Thermistor):**

Έχει μεταβλητή αντίσταση ανάλογα με τη θερμοκρασία. Χρησιμοποιείται συχνά σε εφαρμογές που απαιτούν μέτρηση ή έλεγχο θερμοκρασίας.

**Χρήσεις:**

Οι αντιστάτες χρησιμοποιούνται σε πληθώρα εφαρμογών, όπως τα ηλεκτρικά κυκλώματα, τα κυκλώματα ελέγχου, ηλεκτρονικά εξαρτήματα, και πολλά άλλα. Συνδέονται συχνά σε σειρά ή παράλληλα με άλλα εξαρτήματα για τον έλεγχο της ροής του ηλεκτρικού ρεύματος και την προστασία των ηλεκτρονικών κυκλωμάτων.

## 2.5 Σταθεροποιητής τάσης BU33TD3WG

Το BU33TD3WG είναι ένας γραμμικός σταθεροποιητής τάσης (LDO - Low Drop-Out) που παρέχει σταθερή τάση εξόδου 3.3V με ένα ρεύμα εξόδου έως 200mA. Είναι χαρακτηρισμένος για την υψηλή ακρίβεια του (high accuracy), υψηλή σταθερότητα στις διάφορες συνθήκες λειτουργίας και χαμηλή συνολική πτώση τάσης (low drop-out).

Τα βασικά χαρακτηριστικά του BU33TD3WG είναι:

Τάση εξόδου: 3.3V

Ρεύμα εξόδου: 200mA

Χαμηλή πτώση τάσης (Low Drop-Out)

Υψηλή ακρίβεια

Υψηλή αντοχή σε παρεμβολές τροφοδοσίας (High PSRR - Power Supply Rejection Ratio)

Το BU33TD3WG μπορεί να χρησιμοποιηθεί σε διάφορες εφαρμογές, κυρίως όπου απαιτείται σταθερή τάση τροφοδοσίας με χαμηλή πτώση τάσης και υψηλή ακρίβεια, όπως σε ηλεκτρονικές συσκευές, αισθητήρες, κυκλώματα ψηφιακής επεξεργασίας σήματος, κ.ά [i].

## 2.6 Μετατροπέας USB-to-serial FT234XD

Το FT234XD είναι ένα ολοκληρωμένο κύκλωμα που προσφέρεται από την FTDI (Future Technology Devices International) και λειτουργεί ως προσαρμογέας USB σε σειριακό (UART). Συγκεκριμένα, πρόκειται για ένα USB to UART (Serial) IC, που επιτρέπει τη μετατροπή των δεδομένων από μορφή USB σε σειριακή μορφή και αντίστροφα.

Οι βασικές λειτουργίες και χαρακτηριστικά του FT234XD περιλαμβάνουν:

**Μετατροπή USB σε UART:** Το FT234XD μπορεί να χρησιμοποιηθεί για να συνδέσει συσκευές με διεπαφή USB σε συσκευές που χρησιμοποιούν σειριακή επικοινωνία.

**Υψηλή Ταχύτητα Μετάδοσης Δεδομένων:** Υποστηρίζει υψηλές ταχύτητες μετάδοσης δεδομένων μέσω της διεπαφής USB.

**Ευελιξία:** Το IC παρέχει δυνατότητες ευελιξίας όπως ρυθμίσεις για τον ρυθμό μετάδοσης, την παραμετροποίηση UART, κ.ά.

Τα προϊόντα της FTDI, όπως το FT234XD, είναι δημοφιλή σε εφαρμογές όπου χρειάζεται η σύνδεση συσκευών με υποστήριξη USB σε υπολογιστές ή άλλες συσκευές μέσω σειριακής επικοινωνίας[ii].

# Κεφάλαιο 3 – Όργανα και αναλυτής φάσματος

## 3.1 Γεννήτριες σημάτων

Η γεννήτρια σημάτων, γνωστή και ως γεννήτρια συναρτήσεων ή ηλεκτρονική γεννήτρια, είναι μια σημαντική συσκευή στον χώρο της ηλεκτρονικής και της τεχνολογίας γενικότερα. Πρόκειται για μία συσκευή που χρησιμοποιείται για τη δημιουργία ηλεκτρικών σημάτων με διάφορα χαρακτηριστικά, προσφέροντας ευελιξία και ακρίβεια στον έλεγχο των παραμέτρων του σήματος.

Οι γεννήτριες σήματος επιτρέπουν στους χρήστες να δημιουργούν σήματα διαφόρων τύπων, συχνοτήτων και φάσης.

Ο βασικός σκοπός της γεννήτριας σήματος είναι να παρέχει στους μηχανικούς, ηλεκτρονικούς και ερευνητές ένα εργαλείο για τη δοκιμή και τον έλεγχο κυκλωμάτων. Επιπλέον, χρησιμοποιείται για τη διεξαγωγή πειραμάτων σε εκπαιδευτικό περιβάλλον, επιτρέποντας στους φοιτητές να κατανοήσουν τις ιδιότητες των ηλεκτρικών σημάτων και των κυκλωμάτων.

## 3.2 Παλμογράφος

Ένας παλμογράφος (oscilloscope) είναι μία συσκευή μέτρησης που χρησιμοποιείται στην ηλεκτρονική για να παρακολουθεί και να εμφανίζει την κυματομορφή της τάσης κατά μήκος του χρόνου. Είναι ένα εξαιρετικά χρήσιμο εργαλείο για τον έλεγχο και την ανάλυση ηλεκτρικών σημάτων.

Οι πρώτοι παλμογράφοι αποτελούνταν συνήθως από έναν καθολικό σωλήνα εικόνας (CRT) ή μια οθόνη LCD, στην οποία εμφανίζόταν το γράφημα του σήματος. Ο

οριζόντιος άξονας αντιπροσωπεύει τον χρόνο, ενώ ο κατακόρυφος άξονας αντιπροσωπεύει την τάση.

Οι μηχανικοί, ηλεκτρολόγοι, φυσικοί και άλλοι επαγγελματίες χρησιμοποιούν παλμογράφους για να εξετάσουν και να αναλύσουν ηλεκτρικά σήματα, όπως τα κύματα παλμών, τα σήματα ήχου, τα σήματα ραδιοσυχνοτήτων, και πολλά άλλα. Οι παλμογράφοι είναι επίσης απαραίτητα εργαλεία στον τομέα της αναλογικής και ψηφιακής ηλεκτρονικής, καθώς και σε εφαρμογές όπως η συντήρηση και επισκευή ηλεκτρονικών συσκευών.

## 3.3 Αρχή λειτουργίας αναλυτή φάσματος

Ο αναλυτής αποτελείται από ένα φίλτρο διέλευσης συχνοτήτων (bandpass filter) που σαρώνει μια συγκεκριμένη περιοχή του φάσματος που μας ενδιαφέρει. Έστω ότι το σήμα εισόδου είναι 1MHz, όταν το φίλτρο διέλευσης συχνοτήτων σαρώνει την περιοχή του 1MHz, θα “δει” το σήμα εισόδου και θα το απεικονίσει στην οθόνη. Αν και θεωρητικά το παραπάνω σενάριο δουλεύει, είναι πολύ δύσκολο και δαπανηρό να κατασκευαστεί ένα φίλτρο που συντονίζεται σε μια ευρεία περιοχή. Μια ευκολότερη, και συνεπώς λιγότερο δαπανηρή υλοποίηση είναι να χρησιμοποιηθεί ένας συντονισμένος (tunable) τοπικός ταλαντωτής, και να κρατήσουμε σταθερό το φίλτρο διέλευσης συχνοτήτων. Θα δούμε ότι σε αυτό το σενάριο “σκανάρουμε” το σήμα εισόδου πρωτύτερα του σταθερού φίλτρου, και όταν περνάει μέσα από το σταθερό φίλτρο διέλευσης συχνοτήτων, αυτό εμφανίζεται στην οθόνη.

## 3.4 Πλεονεκτήματα και μειονεκτήματα αναλυτή φάσματος

**Α. Πλεονεκτήματα**

**Αναλυτική Προβολή Σημάτων:** Οι αναλυτές φάσματος επιτρέπουν την προβολή των σημάτων σε συχνότητες, διευκολύνοντας τον εντοπισμό και την ανίχνευση συχνοτήτων κορυφής, παρασιτικών σημάτων και παραμορφώσεων. Ανίχνευση συχνοτήτων κορυφής είναι ο εντοπισμός των συχνοτήτων που έχουν τη μεγαλύτερη ένταση ή είναι πιο σημαντικές σε ένα σήμα, αναδεικνύοντας έτσι τα κυρίαρχα χαρακτηριστικά του.

**Μεγάλο Εύρος Συχνοτήτων:** Οι αναλυτές φάσματος καλύπτουν ευρύ φάσμα συχνοτήτων, από χαμηλές συχνότητες (σεισμικές διαταραχές, χαμηλές συχνότητες

ραδιοφωνικών κυμάτων) μέχρι υψηλές συχνότητες (επικοινωνίες μικροκυμάτων, WiFi, ραντάρ).

**Δυνατότητα Ανίχνευσης Κορυφών:** Οι αναλυτές φάσματος μπορούν να εντοπίσουν τις συχνότητες κορυφής σε σύνθετα σήματα, κάτι που είναι ιδιαίτερα χρήσιμο σε επικοινωνίες και ραντάρ.

**Ανίχνευση Συχνοτήτων Παρασίτων:** Μπορούν να ανιχνεύσουν συχνότητες παρασίτων και ανεπιθύμητων σημάτων σε επικοινωνιακά συστήματα, βοηθώντας στον εντοπισμό πιθανών παρεμβολών.

**Αναγνώριση Προτύπων:** Μπορούν να αναγνωρίσουν πρότυπα και τάσεις στα σήματα, βοηθώντας στον εντοπισμό αλλαγών και ανωμαλιών.

**Σύγκριση Σημάτων:** Οι αναλυτές φάσματος μπορούν να συγκρίνουν διαφορετικά σήματα και να εντοπίσουν τις διαφορές μεταξύ τους, βοηθώντας στην ανάλυση και σύγκριση σημάτων.

**Μέτρηση Εντάσεων:** Μπορούν να μετρήσουν την ένταση σήματος σε διάφορες συχνότητες, προσφέροντας πληροφορίες για την ισχύ και το φάσμα του σήματος.

**Εφαρμογές στην Εκπαίδευση:** Χρησιμοποιούνται στην εκπαίδευση για την επίδειξη αρχών σήματος και την κατανόηση της φασματικής ανάλυσης.

Αυτά είναι μερικά από τα πλεονεκτήματα που προσφέρουν οι αναλυτές φάσματος. Οι εφαρμογές τους καλύπτουν πολλούς τομείς, από την ηλεκτρονική και τις τηλεπικοινωνίες έως την ιατρική και την επιστήμη.

**Β. Μειονεκτήματα**

**Πολυπλοκότητα:** Οι αναλυτές φάσματος μπορεί να είναι συχνά πολύπλοκοι και ακριβοί στο σχεδιασμό και την υλοποίηση, ιδιαίτερα όταν απαιτούνται υψηλή ακρίβεια και απόδοση.

**Περιορισμένο Χρονικό Παράθυρο:** Η διαδικασία της σήμανσης και της ανάλυσης Fourier απαιτεί κάποιο χρονικό διάστημα, και αυτό μπορεί να περιορίζει τη δυνατότητα ανίχνευσης σημάτων με γρήγορες μεταβολές.

**Περιορισμένη Ανάλυση Χρόνου:** Οι αναλυτές φάσματος είναι καλοί για την ανάλυση σε συχνότητες, αλλά δεν παρέχουν πληροφορίες για τη χρονική εξέλιξη των σημάτων.

**Παρεμβολές:** Κατά τη λήψη σημάτων, μπορούν να υπάρξουν παρεμβολές από άλλα σήματα ή παράσιτα, καθιστώντας δυσκολότερη την αναγνώριση των επιθυμητών σημάτων.

**Επίπεδο Εκπαίδευσης:** Η χρήση και η κατανόηση των αναλυτών φάσματος απαιτεί μια βαθιά κατανόηση των αρχών σήματος και φασματικής ανάλυσης.

**Κόστος:** Οι εξελίξεις σε τεχνολογίες αναλυτών φάσματος μπορεί να τους καθιστούν ακριβούς για ορισμένες εφαρμογές.

**Παρόλα αυτά τα μειονεκτήματα, οι αναλυτές φάσματος παραμένουν ισχυρά εργαλεία για την ανάλυση και την κατανόηση σημάτων σε ποικίλους τομείς.**

## 3.5 Που και γιατί χρησιμεύουν οι αναλυτές φάσματος

Οι αναλυτές φάσματος είναι χρήσιμοι σε πολλούς τομείς και εφαρμογές λόγω της δυνατότητάς τους να αναλύουν και να προβάλλουν τις συχνότητες των διαφορετικών σημάτων. Ανάλογα με τον τομέα και τον σκοπό χρήσης, οι αναλυτές φάσματος χρησιμοποιούνται για διάφορους λόγους:

**Ηλεκτρονική Επισκευή και Συντήρηση:** Στην ηλεκτρονική, οι αναλυτές φάσματος χρησιμοποιούνται για τη διάγνωση και επισκευή προβλημάτων σε κυκλώματα και εξαρτήματα. Μπορούν να εντοπίσουν αλλοιώσεις, παρεμβολές και παρασιτικές συχνότητες που ενδέχεται να προκαλούν προβλήματα.

**Τηλεπικοινωνίες:** Στις τηλεπικοινωνίες, οι αναλυτές φάσματος χρησιμοποιούνται για την ανίχνευση και τον έλεγχο των συχνοτήτων σήματος σε δίκτυα και συσκευές. Βοηθούν στην εντοπισμό παρεμβολών, ανίχνευση κορυφών συχνοτήτων, και μελέτη του φάσματος χρήσης.

**Ραδιοερασιτεχνισμός:** Οι ραδιοερασιτέχνες χρησιμοποιούν αναλυτές φάσματος για να εξετάσουν το ραδιοφάσμα, να ανιχνεύσουν σήματα, να μετρήσουν εντάσεις και να αναλύσουν σήματα που λαμβάνουν ή εκπέμπουν.

**Ιατρική και Βιολογία:** Στην ιατρική, οι αναλυτές φάσματος χρησιμοποιούνται για την ανίχνευση και τον έλεγχο ιατρικών σημάτων, όπως ηλεκτροκαρδιογράφημα (ECG) και ηλεκτροεγκεφαλογράφημα (EEG). Επίσης, στη βιολογία, χρησιμοποιούνται για την ανάλυση φασματικών χαρακτηριστικών από δείγματα, όπως στον τομέα της φαρμακολογίας.

**Επιστημονική Έρευνα:** Στην επιστήμη, οι αναλυτές φάσματος χρησιμοποιούνται για τη μελέτη του φάσματος εκπομπής και απορρόφησης από αντικείμενα, παρέχοντας πληροφορίες για τα στοιχεία που αποτελούν τα αντικείμενα.

**Σχεδίαση Κυκλωμάτων:** Στον τομέα της ηλεκτρονικής, οι αναλυτές φάσματος χρησιμοποιούνται για την εκτίμηση των χαρακτηριστικών σημάτων κατά τη σχεδίαση κυκλωμάτων.

## 3.6 Ο Αλγόριθμος Cooley-Tukey στον Υπολογισμό της Μετασχηματισμένης Fourier

Ο αλγόριθμος Cooley-Tukey είναι ένας από τους πιο δημοφιλείς αλγορίθμους για τον υπολογισμό του μετασχηματισμού Fourier (FFT).

### Βασικές Αρχές της FFT

Ο FFT αναλύει μια ακολουθία δειγμάτων σήματος και τη μετατρέπει σε ένα φάσμα συχνοτήτων, δηλαδή τον πίνακα των συχνοτήτων και των συντελεστών τους. Ένας από τους βασικούς τρόπους να υπολογιστεί ο FFT είναι μέσω του αλγορίθμου Cooley-Tukey.

### Αλγόριθμος Cooley-Tukey

Ο αλγόριθμος Cooley-Tukey βασίζεται στην ιδέα του διαχωρισμού μιας ακολουθίας σε υπο-ακολουθίες μικρότερου μεγέθους και την επαναληπτική εφαρμογή του αλγορίθμου σε αυτές. Ο αλγόριθμος εκτελείται σταδιακά, με τον διαχωρισμό της ακολουθίας σε υπο-ακολουθίες με μειωμένο μέγεθος και τη συνένωση των αποτελεσμάτων σε κάθε βήμα.

Ένα απλό παράδειγμα για να κατανόηση της λειτουργίας του αλγόριθμου Cooley-Tukey για τον υπολογισμό της μετασχηματισμένης Fourier (FFT) σε μια ακολουθία αριθμών.

Έστω η ακολουθία:

X= [3, 1, 0, 2]

Βήμα 1 - Χωρίζει την ακολουθία σε υπο-ακολουθίες: Στο πρώτο βήμα, διαιρεί την ακολουθία X σε δύο υπο-ακολουθίες μικρότερου μεγέθους.

Για το παράδειγμά μας, το

N=4 οπού N το μέγεθος του πίνακα, οπότε k=0,1,2,3 και οι δείκτες σε δυαδική μορφή είναι: 00,01,10,11.

Οι δυαδικές αναπαραστάσεις δεικτών μας δείχνει ποια στοιχεία θα πάνε στην υπο-ακολουθία $X_0$ και ποια στο X1. Συγκεκριμένα, τα στοιχεία με δείκτη k που έχουν το πρώτο bit 0 πηγαίνουν στο $X_0$ , ενώ αυτά που έχουν το πρώτο bit 1 πηγαίνουν στο X1.

Έτσι, δημιουργεί τις υπο-ακολουθίες:

$X_0$ = [3, 0] και X1= [1, 2]

Στον αλγόριθμο Cooley-Tukey για τον υπολογισμό του FFT, η διαδικασία bit-reversal (αντιστροφή bit) αφορά την αντιστροφή των bit των δεικτών των ακολουθιών κατά το στάδιο της αναδρομικής εφαρμογής του αλγορίθμου.

Βήμα 2 - Υπολογίζει τους FFT των υπο-ακολουθιών: Στο δεύτερο βήμα, υπολογίζει τους FFT των υπο-ακολουθιών X0 και X1 . Για τον υπολογισμό της FFT των υπο-ακολουθιών, επαναλαμβάνει την ίδια διαδικασία. Έτσι, για το X0:

$X'_0$ = [3,0]

$X_0$ = FFT [$X'_0$] = [3, 3]

Και για το $X'_1$

$X_1$ = FFT [$X'_1$] = [3, -j]

Βήμα 3 - Συνδυάζει τα αποτελέσματα: Στο τρίτο βήμα, συνδυάζει τα αποτελέσματα από τις FFT των υπο-ακολουθιών για να πάρει την τελική FFT της αρχικής ακολουθίας X. Αυτό γίνεται με τη χρήση της εξίσωσης:

$X[k] = X_0[k] + W^i_n \cdot X_1[k]$

όπου $W^i_n$ είναι ο εκθέτης Euler $e^{-i2\pi k/N}$ και k είναι ο δείκτης του στοιχείου.

Στο παράδειγμά μας, το N είναι 4, οπότε έχουμε:

$X[0] = X_0[0] + W^0_4 \cdot X_1[0] = 3 + 3 = 6$

$X[1] = X_0[1] + W^1_4 \cdot X_1[1] = 3 - 3 = 0$

$X[2] = X_0[2] + W^2_4 \cdot X_1[2] = 3 - i = 3 - i$

$X[3] = X_0[3] + W^3_4 \cdot X_1[3] = 3 + i = 3 + i$

Το αποτέλεσμα αυτού του υπολογισμού είναι ο FFT της αρχικής ακολουθίας X:

X= [6 , 0 , 3 -i , 3+i]

Ο αλγόριθμος Cooley-Tukey επιτρέπει τον υπολογισμό του FFT αποδοτικά, διαιρώντας το πρόβλημα σε μικρότερα υπο-προβλήματα και συνδυάζοντας τα αποτελέσματα.

# Κεφάλαιο 4 - Ο μικροελεγκτής PSoC6

## 4.1 Προγραμματιστής Miniprog4

Το Miniprog4 είναι ένα σύστημα προγραμματισμού και αποσφαλμάτωσης που χρησιμοποιείται σε συνδυασμό με τους μικροελεγκτές PSoC 6 της Cypress Semiconductor. Σχεδιάστηκε για να παρέχει αξιόπιστο και αποτελεσματικό προγραμματισμό, ενώ παράλληλα παρέχει δυνατότητες αποσφαλμάτωσης σε εφαρμογές ενσωματωμένων συστημάτων.

Χαρακτηριστικά και Λειτουργίες

**1. Προγραμματισμός Μικροελεγκτών**

Το Miniprog4 επιτρέπει τον προγραμματισμό των μικροελεγκτών PSoC 6. Με αυτόν τον τρόπο, μπορείτε να φορτώνετε το πρόγραμμά σας στον μικροελεγκτή, είτε πρόκειται για firmware ελέγχου, είτε για προγράμματα εφαρμογών.

**2. Αποσφαλμάτωση**

Ένα σημαντικό χαρακτηριστικό του Miniprog4 είναι η δυνατότητα αποσφαλμάτωσης. Μπορεί να χρησιμοποιήσει τη συσκευή για να εκτέλεσει βήμα-βήμα τον κώδικα, να παρακολουθήσει μεταβλητές, και να εντοπίσει σφάλματα στην εφαρμογή.

**3. Συμβατότητα**

Είναι συμβατό με διάφορες συσκευές PSoC, καθιστώντας το ένα πολύ ευέλικτο εργαλείο για ανάπτυξη σε διάφορες πλατφόρμες.

**4. Σύνδεση με το Περιβάλλον Ανάπτυξης**

Το Miniprog4 συνδέεται με το περιβάλλον ανάπτυξης της Infineon και άλλα περιβάλλοντα ανάπτυξης που υποστηρίζουν τους μικροελεγκτές PSoC 6.

**5. Εφαρμογές**

Χρησιμοποιείται για ανάπτυξη σε έργα που απαιτούν προγραμματισμό και αποσφαλμάτωση μικροελεγκτών PSoC 6.

Συνολικά, το Miniprog4 είναι ένα εξειδικευμένο εργαλείο που επιτρέπει στους προγραμματιστές να αναπτύσσουν, να προγραμματίσουν και να αποσφαλματώσουν με αποτελεσματικό τρόπο τους μικροελεγκτές PSoC 6, συμβάλλοντας στην αξιοπιστία και την αποτελεσματικότητα των εφαρμογών ανάπτυξης[iii].

## 4.2 Ο μικροελεγκτής PSoC6

Ο μικροελεγκτής PSoC 6 είναι Ένας διπύρηνος (dual-core) μικροελεγκτής που κατασκευάζεται από την CypressSemiconductor, που πλέον ανήκει στην InfineonTechnologies. Είναι μέρος της σειράς PSoC (ProgrammableSystem-on-Chip) και συνδυάζει τις λειτουργίες ενός μικροελεγκτή με την ευελιξία της προγραμματιζόμενης λογικής και αναλογικού σχεδιασμού (PLD/PSoC) στο ίδιο ενσωματωμένο κύκλωμα.

Το ολοκληρωμένο PSoC 6 περιλαμβάνει δύο πυρήνες:

Πυρήνας ARMCortex-M4F: Αυτός ο πυρήνας παρέχει υψηλή απόδοση και χρησιμοποιείται για την εκτέλεση των εφαρμογών υψηλής απόδοσης. Διαθέτει μονάδα επεξεργασίας για αριθμούς κινητής υποδιαστολής.

Πυρήνας ARMCortex-M0+: Αυτός ο πυρήνας χρησιμοποιείται για χαμηλή κατανάλωση ενέργειας και χαμηλές απαιτήσεις σε πόρους. Συνήθως, αυτός ο πυρήνας αναλαμβάνει τις βασικές λειτουργίες σε καταστάσεις χαμηλής ισχύος.

Σημαντικά χαρακτηριστικά του PSoC 6 περιλαμβάνουν:

**Προγραμματιζόμενη Αναλογική και Ψηφιακή Λογική (PLD):** Ο χρήστης έχει τη δυνατότητα να προγραμματίσει ψηφιακή και αναλογική λογική για να προσαρμόσει τις λειτουργίες του μικροελεγκτή στις ανάγκες της εφαρμογής.

**Υψηλή Ασφάλεια:** Υποστηρίζει χαρακτηριστικά υψηλής ασφάλειας για εφαρμογές που απαιτούν προστασία από απειλές ασφαλείας.

Για παράδειγμα

Ένα χαρακτηριστικό υψηλής ασφάλειας που υποστηρίζει το PSoC6 είναι η "Secure Boot" (Ασφαλής Εκκίνηση). Η ασφαλής εκκίνηση είναι μια διαδικασία που εξασφαλίζει ότι μόνο έγκυρο λογισμικό και επικυρωμένο από τον κατασκευαστή μπορεί να ξεκινήσει τη συσκευή.

Συγκεκριμένα, το χαρακτηριστικό Secure Boot λειτουργεί ως εξής:

**Επαλήθευση Υπογραφής Κώδικα:** Κατά τη διαδικασία εκκίνησης, το σύστημα ελέγχει την υπογραφή του εκτελέσιμου κώδικα για να διασφαλίσει ότι είναι από τον κατασκευαστή.

**Έλεγχος Ακεραιότητας:** Το σύστημα ελέγχει την ακεραιότητα του κώδικα για να βεβαιωθεί ότι δεν έχει τροποποιηθεί.

**Υποστήριξη κρυπτογραφημένων Κλειδιών:** Χρησιμοποιεί κρυπτογραφημένα κλειδιά για την επαλήθευση της υπογραφής και για άλλες διαδικασίες ασφαλούς εκκίνησης.

**Συνδεσιμότητα και Περιφερειακά:** Ποικίλα περιφερειακά, BluetoothLowEnergy (BLE), και άλλα.

**Αισθητήρες**: Υποστηρίζει ενσωματωμένους αισθητήρες, όπως αισθητήρες αφής, που τον καθιστούν κατάλληλο για εφαρμογές αφής και αισθητήρες περιβάλλοντος.

Ο μικροελεγκτής PSoC 6 είναι ένας ευέλικτος και ισχυρός μικροελεγκτής για εφαρμογές που καλύπτουν εύρος απαιτήσεων από υψηλή απόδοση έως χαμηλή κατανάλωση ενέργειας[iv].

## 4.3 Ο μετατροπέας ψηφιακού σήματος σε αναλογικό (DAC)

Η μονάδα DAC (Digital-to-Analog Converter) στο PSoC 6 αποτελεί έναν μετατροπέα ψηφιακού σήματος σε αναλογικό που βρίσκεται ενσωματωμένος στον μικροελεγκτή.

Ο DAC είναι χρήσιμος σε εφαρμογές όπου απαιτείται παραγωγή αναλογικού σήματος, όπως για τον έλεγχο αναλογικών κυκλωμάτων, την αναπαραγωγή ήχου κλπ.

Οι μετατροπείς ψηφιακού σήματος σε αναλογικό επιτρέπουν στον μικροελεγκτή να μετατρέπει ψηφιακά σήματα (συνήθως μια αριθμητική τιμή ή άλλη πηγή ψηφιακού σήματος) σε αντίστοιχα αναλογικά σήματα.

## 4.4 Ο μετατροπέας αναλογικού σήματος σε ψηφιακό ADC

Ο μετατροπέας αναλογικού σήματος σε ψηφιακό (ADC) στο PSoC 6 προσφέρει τη δυνατότητα μετατροπής αναλογικών σημάτων σε ψηφιακή μορφή. Η ADC είναι κρίσιμη για πολλές εφαρμογές  καθώς επιτρέπει την ακριβή ανάγνωση αισθητήρων και άλλων αναλογικών πηγών.

Αναλυτική Εξήγηση του ADC στο PSoC 6:

**Ορισμός ADC:**

Ο ADC είναι ένα κύκλωμα που μετατρέπει αναλογικά σήματα σε ψηφιακά, καθιστώντας δυνατή την ανάγνωση και την επεξεργασία τους από τον μικροελεγκτή.

**Ακρίβεια και Ανάλυση:**

Η ακρίβεια του ADC είναι κρίσιμη για πολλές εφαρμογές. Το PSoC 6 παρέχει ADC με διάφορες δυνατότητες ανάλυσης, επιτρέποντας την προσαρμογή στις απαιτήσεις κάθε εφαρμογής η ανάλυση που χρησιμοποιούμε είναι στα 12-bit.

**Εφαρμογές:**

Οι εφαρμογές του ADC στο PSoC 6 είναι ποικίλες, περιλαμβάνοντας τη μέτρηση θερμοκρασίας, τάσεων μπαταρίας, και την ανάγνωση αισθητήρων περιβάλλοντος.

**Προσαρμογή:**

Το PSoC 6 επιτρέπει την προσαρμογή του ADC στις ανάγκες του σχεδιαστή μέσω του PSoC Creator, παρέχοντας ευελιξία στη σχεδίαση εφαρμογών.

Συνοπτικά, ο ADC στο PSoC 6 παρέχει πολλές ρυθμίσεις στο σχεδιαστή όπως η συχνότητα δειγματοληψίας, το format του αποτελέσματος, δυνατότητα υπολογισμού μέσου όρου μετρήσεων, αυτόματης δειγματοληψίας κλπ.

## 4.5 Η μονάδα απευθείας πρόσβασης στη μνήμη

Η μονάδα απευθείας πρόσβασης στην μνήμη (DMA – DirectMemoryAccess) αναπτύχθηκε για να βελτιώσει την απόδοση της μεταφοράς δεδομένων μεταξύ περιφερειακών συσκευών και της κεντρικής μνήμης, ελαχιστοποιώντας τον φόρτο εργασίας του κεντρικού μικροελεγκτή. Στο PSoC 6, η DMA είναι μια ισχυρή λειτουργία που προσφέρει ευελιξία και απόδοση για διάφορες εφαρμογές.

**Αυτόνομη Λειτουργία:**

Η DMA λειτουργεί αυτόνομα από τον κεντρικό επεξεργαστή, επιτρέποντας στον επεξεργαστή να επικεντρωθεί σε άλλες εργασίες, ενώ η μεταφορά δεδομένων διεξάγεται από την DMA.

**Αποσυμφόρηση εργασιών στον Κεντρικό Επεξεργαστή:**

Η χρήση της DMA μειώνει την επιβάρυνση του κεντρικού επεξεργαστή, καθώς οι μεταφορές δεδομένων διαχειρίζονται από έναν αυτόνομο ελεγκτή.

**Συνδυασμός με Πολλαπλά Κανάλια:**

Το PSoC 6 επιτρέπει τη χρήση πολλαπλών καναλιών DMA, επιτρέποντας τη μεταφορά πολλαπλών ακολουθιών δεδομένων ταυτόχρονα.

**Εφαρμογές:**

Οι εφαρμογές περιλαμβάνουν τη μεταφορά δεδομένων από/προς περιφερειακές συσκευές όπως UART, SPI, I2C, και αισθητήρες.

Συνοπτικά, η μονάδα DMA στο PSoC 6 είναι ένα ισχυρό εργαλείο που ενισχύει την απόδοση του συστήματος, εξασφαλίζοντας αποτελεσματική μεταφορά δεδομένων χωρίς την εμπλοκή του κεντρικού μικροελεγκτή.

## 4.6 Η μονάδα χρονισμού (TCPWM)

Ο Χρονομετρητής/Μετρητής (Timer/Counter) στο PSoC 6 (TCPWM)

Ο Timer/Counter είναι ένα σημαντικό περιφερειακό στο PSoC 6 που προσφέρει λειτουργίες χρονομέτρησης, μέτρησης και απαρίθμησης. Ο TCPWM σχεδιάστηκε για να παρέχει ευελιξία και ακρίβεια στον έλεγχο του χρόνου σε διάφορες εφαρμογές.

Βασικά Χαρακτηριστικά του Timer/Counter (TCPWM) στο PSoC 6:

**Λειτουργίες Χρονομέτρησης:**

Ο TCPWM παρέχει ποικίλες λειτουργίες χρονομέτρησης, συμπεριλαμβανομένων λειτουργιών κανονικού χρονομετρητή, συγκρίσεων και ανίχνευσης συμβάντων.

**Εύρος Εφαρμογών:**

Χρησιμοποιείται για χρονομετρητές, χρονοδιακόπτες, μετρητές και άλλες χρονομετρικές λειτουργίες σε ποικίλες εφαρμογές.

**Συγκρίσεις Καταγραφής:**

Μπορεί να πραγματοποιεί συγκρίσεις μεταξύ της τρέχουσας τιμής του χρονομέτρη και προκαθορισμένων τιμών, εκτελώντας ενέργειες όταν συμβαίνει μια συγκεκριμένη συνθήκη.

**Παραγωγή σημάτων διαμορφωμένων κατά εύρος παλμού (PWM):**

Ο TCPWM μπορεί να παράγει σήματα PWM , ένα σημαντικό χαρακτηριστικό για εφαρμογές όπως ο έλεγχος κινητήρων και η διαμόρφωση σήματος.

**Κανάλια και Λειτουργικότητα:**

Ο TCPWM διαθέτει πολλά κανάλια που μπορούν να παραμετροποιηθούν για διάφορες λειτουργίες, προσφέροντας πολλαπλή λειτουργικότητα.

**Προσαρμογή μέσω PSoCCreator:**

Οι ρυθμίσεις του TCPWM μπορούν να προσαρμοστούν εύκολα μέσω του περιβάλλοντος ανάπτυξης PSoCCreator, παρέχοντας ευελιξία στο σχεδιασμό.

**Εφαρμογές:**

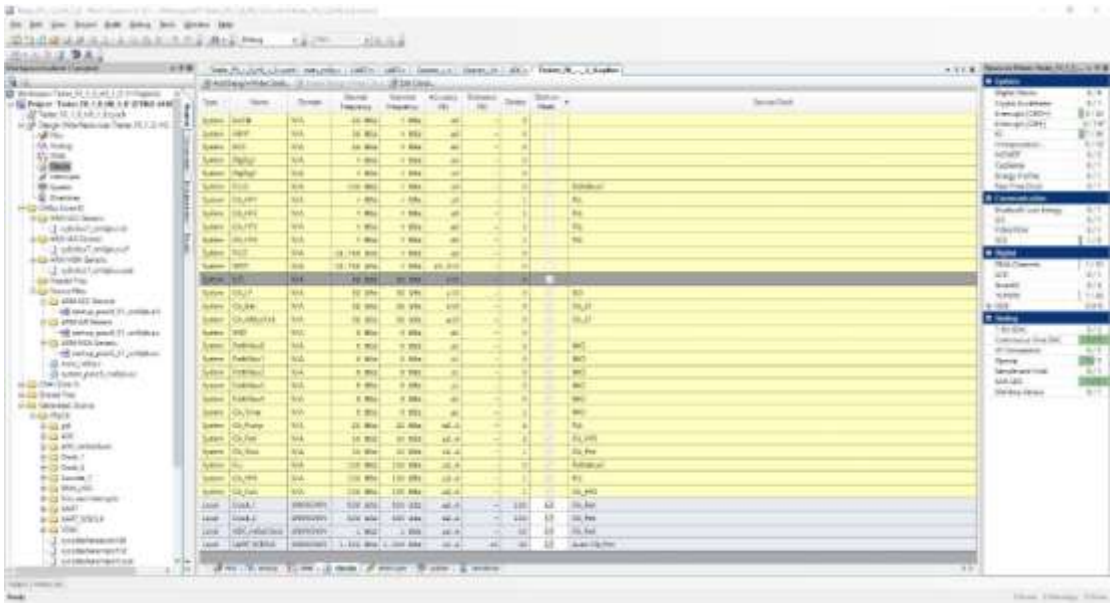
Χρησιμοποιείται σε εφαρμογές όπως ρολόγια πραγματικού χρόνου (RTC), ρομποτική, κινητήρες DC, και όπου απαιτείται η χρονομέτρηση και ο έλεγχος χρονισμού.

## 4.7 Ρολόι (Clock)

Στο PSoC 6, το "clock" αναφέρεται στα ρολόγια που χρησιμοποιούνται για να χρονίσουν τη λειτουργία των διάφορων υποσυστημάτων του μικροελεγκτή. Τα ρολόγια είναι σημαντικά για τον συγχρονισμό των εσωτερικών εργασιών αλλά και για τα κυκλώματα σειριακής επικοινωνίας που διαθέτει το PSoC6.

Υπάρχουν διάφορα ρολόγια στο PSoC 6, και καθένα από αυτά χρησιμοποιείται για τον έλεγχο διαφόρων λειτουργιών του μικροελεγκτή.

Τα ρολόγια που χρησιμοποιήθηκαν βρίσκονται στην παρακάτω εικόνα (Εικ. 1)



Εικ. 1 Ρολόγια στο Psoc Creator

Η ρύθμιση και ο έλεγχος των ρολογιών είναι σημαντικό μέρος του προγραμματισμού για να επιτευχθεί η βέλτιστη λειτουργία του συστήματος.

## 4.8 Η μονάδα ασύγχρονης σειριακής επικοινωνίας (UART)

Η μονάδα Universal Asynchronous Receiver/Transmitter, γνωστή και ως UART, είναι ένα από τα βασικά περιφερειακά κυκλώματα που προσφέρει το PSoC 6. Η UART είναι ένα δημοφιλές interface επικοινωνίας σειριακών δεδομένων, σχεδιασμένο για τη μεταφορά δεδομένων μεταξύ δύο συσκευών μέσω δύο αγωγών επικοινωνίας: ένας για τα δεδομένα που μεταδίδονται (TX - Transmit) και ένας για τα δεδομένα που λαμβάνονται (RX - Receive).

Βασικά Χαρακτηριστικά του UART στο PSoC 6:

**1. Σειριακή Επικοινωνία:**

Η UART χρησιμοποιεί σειριακή μορφή επικοινωνίας, όπου τα δεδομένα μεταδίδονται bit-προς-bit. Αυτό το καθιστά κατάλληλο για εφαρμογές στις οποίες απαιτείται επικοινωνία ανάμεσα σε απομακρυσμένες συσκευές.

**2. Ασύγχρονη Λειτουργία:**

Ο χαρακτήρας "ασύγχρονος" σημαίνει ότι δεν απαιτείται κοινό ρολόι μεταξύ του πομπού και του δέκτη. Η επικοινωνία ξεκινάει και τερματίζεται με την έλευση των Start (λογικό ‘0’) και Stop bits (λογικό ‘1’).

**3. Ευελιξία Ρυθμίσεων:**

Ο UART προσφέρει ευελιξία στις ρυθμίσεις, όπως η ταχύτητα μετάδοσης, το πλάτος, λειτουργίες ελέγχου σφάλματος και άλλες παράμετροι που μπορούν να προσαρμοστούν σύμφωνα με τις απαιτήσεις της εφαρμογής.

**4. Μνήμη προσωρινής αποθήκευσης FIFO:**

Ορισμένες υλοποιήσεις UART περιλαμβάνουν FIFO (First In, First Out) buffers για την αποθήκευση πολλαπλών χαρακτήρων, βελτιώνοντας την απόδοση και την αποτελεσματικότητα.

**5. Εφαρμογές:**

Χρησιμοποιείται για τη μεταφορά δεδομένων ανάμεσα σε μικροελεγκτές και άλλες περιφερειακές συσκευές σε ένα ενσωματωμένο σύστημα.

**6. Προσαρμογή μέσω PSoC Creator:**

Οι ρυθμίσεις της UART μπορούν να προσαρμοστούν εύκολα μέσω του περιβάλλοντος ανάπτυξης PSoC Creator.

## 4.9 Το λογισμικό του PSoC6

Λογισμικό στο PSoC6: Μια Λεπτομερής Επισκόπηση

Το λογισμικό για το PSoC6 (ProgrammableSystem-on-Chip) αποτελεί κρίσιμο κομμάτι για τον σχεδιασμό και τον προγραμματισμό των εφαρμογών.Η πλατφόρμα ανάπτυξης PSoCCreator αποτελεί το κεντρικό εργαλείο, προσφέροντας πληθώρα δυνατοτήτων για την εύκολη και αποτελεσματική ανάπτυξη.

**PSoCCreator:** Το Ενοποιημένο Περιβάλλον Ανάπτυξης

Το PSoCCreator αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης που επιτρέπει τον σχεδιασμό, τον προγραμματισμό και τον έλεγχο του PSoC. Από απλές έως προηγμένες εφαρμογές, το PSoCCreator παρέχει τα εργαλεία που απαιτούνται για τη δημιουργία πολύπλοκων συστημάτων.

**Σχεδιασμός Συστήματος:**

Το interface σχεδίασης συστημάτων επιτρέπει τον εύκολο σχεδιασμό του hardware, ενσωματώνοντας διάφορα περιφερειακά όπως ADCs, DACs, και ψηφιακούς πυρήνες για άλλες περιφερειακές συσκευές.

**Προγραμματισμός Συστήματος:**

Ο προγραμματιστής μικροελεγκτή επιτρέπει τον προγραμματισμό του μικροελεγκτή ο οποίος πραγματοποιείται χρησιμοποιώντας γλώσσες όπως την C ή την Assembly, με τη βοήθεια ενός compiler που μετατρέπει τον πηγαίο κώδικα σε γλώσσα μηχανής.

**Περιφερειακά κυκλώματα:**

Το λογισμικό επιτρέπει τον εύκολο έλεγχο και παραμετροποίηση πολυάριθμων περιφερειακών κυκλωμάτων, όπως ADCs, DACs, Timers, και άλλων, χρησιμοποιώντας γραφικά εργαλεία.

**Διαδικασία Ανάπτυξης:**

1.Σχεδίαση Συστήματος:

Επιλογή και σύνδεση περιφερειακών στην επιφάνεια σχεδίασης του PSoCCreator. Στη συνέχεια πραγματοποιείται ανάπτυξη του κώδικα στον ενσωματωμένο κειμενογράφο του PSoCCreator ή σε άλλον κειμενογράφο της επιλογής μας.

2.Μεταγλώττιση:

Πραγματοποιείται μεταγλώττιση του κώδικα που αναπτύχθηκε από τον compilerGCC 5.4 για επεξεργαστές ARM.

3.Προσομοίωση:

Προεπισκόπηση της συμπεριφοράς μέσω προσομοίωσης.

4.Προγραμματισμός του PSoC με τη βοήθεια του Miniprog4:

Το πρόγραμμα σε γλώσσα μηχανής που έχει χτιστεί μετά τη διαδικασία μεταγλώττισης εγγράφεται στη μνήμη flash του PSoC6.

5.Δοκιμές και αποσφαλμάτωση:

Τέλος, πραγματοποιείται ο έλεγχος λειτουργίας της εφαρμογής και επανάληψη των βημάτων 1-5 μέχρι τη διόρθωση όλων των σφαλμάτων.

Το λογισμικό στο PSoC, με το PSoCCreator ως κύριο εργαλείο, δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν πολύπλοκα ενσωματωμένα συστήματα με ευελιξία και αποδοτικότητα. Η γραφική διεπαφή, οι προσομοιώσεις, και η ευκολία παραμετροποίησης καθιστούν το PSoCCreator ένα ισχυρό εργαλείο.

## 4.10 Γλώσσα προγραμματισμού C

Η γλώσσα C αποτελεί θεμέλιο λίθο στον κόσμο του προγραμματισμού, και στην περίπτωση του PSoC, αναδεικνύεται ως ισχυρό εργαλείο για τη δημιουργία εφαρμογών ενσωματωμένων συστημάτων. Ας εξετάσουμε πώς η γλώσσα C συνδυάζεται με την πλατφόρμα PSoC, επιτρέποντας στους προγραμματιστές να εκμεταλλευτούν πλήρως τις δυνατότητες αυτής της προηγμένης τεχνολογίας.

Πλεονεκτήματα της Γλώσσας C στο PSoC:

**1. Κοινή και Διαδεδομένη:**

Η γλώσσα C είναι διαδεδομένη και κοινή στον χώρο του προγραμματισμού. Αυτό σημαίνει ότι υπάρχει πληθώρα πόρων υλικού και λογισμικού που χρησιμοποιούν αυτήν τη γλώσσα, καθιστώντας την ευρέως προσβάσιμη.

**2. Απόδοση και Αποτελεσματικότητα:**

Η γλώσσα C παρέχει υψηλή απόδοση και αποτελεσματικότητα, καθιστώντας την κατάλληλη για τον προγραμματισμό ενσωματωμένων συστημάτων όπως το PSoC.

**3. Ευελιξία και Φορητότητα:**

Η γλώσσα C προσφέρει ευελιξία και φορητότητα. Ο κώδικας που γράφεται σε C μπορεί να μεταφερθεί εύκολα σε διάφορες πλατφόρμες, επιτρέποντας την επαναχρησιμοποίηση κώδικα.

**Χρήση της Γλώσσας C στο PSoC Creator:**

Το PSoC Creator ενσωματώνει τη γλώσσα C στη διαδικασία ανάπτυξης. Οι προγραμματιστές μπορούν να χρησιμοποιούν τη γλώσσα C για τον προγραμματισμό των μικροελεγκτών του PSoC, συνδυάζοντας την ισχύ της γλώσσας με τις προηγμένες δυνατότητες του περιβάλλοντος ανάπτυξης εφαρμογών.

## 4.11 Γλώσσα προγραμματισμού Python

Η Python αποτελεί μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, χάρη στην ευανάγνωστη σύνταξή της, την ευελιξία και την ικανότητά της να καλύπτει ένα ευρύ φάσμα εφαρμογών. Ας εξετάσουμε πώς η Python έχει επηρεάσει τον προγραμματισμό και πώς μπορεί να χρησιμοποιηθεί για τη δημιουργία ποικίλων εφαρμογών.

Γενική Επισκόπηση:

Η Python είναι μια υψηλού επιπέδου, διερμηνευόμενη γλώσσα προγραμματισμού που δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991. Η σύνταξη της Python είναι ευανάγνωστη και προσφέρει δυνατότητες υψηλού επιπέδου, καθιστώντας την ιδανική για αρχάριους και έμπειρους προγραμματιστές.

Βασικά Χαρακτηριστικά:

**1. Διερμηνευόμενη Γλώσσα:**

Η Python είναι διερμηνευόμενη, που σημαίνει ότι ο κώδικας εκτελείται γραμμικά από ένα διερμηνευτή, χωρίς την ανάγκη για μεταγλώττιση. Αυτό διευκολύνει την ανάπτυξη και τον έλεγχο του κώδικα.

**2. Ευανάγνωστη Σύνταξη:**

Η σύνταξη της Python είναι ευανάγνωστη και καθορισμένη από τον οδηγό PEP 8 (Python Enhancement Proposal). Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν κώδικα που είναι ευανάγνωστος και ευαίσθητος στη μορφοποίηση.

**3. Πλούσια Βιβλιοθήκη:**

Η Python διαθέτει μια πλούσια συλλογή από βιβλιοθήκες και πακέτα που καλύπτουν πολλούς τομείς, όπως την επιστημονική υπολογιστική, τον ιστό, τις βάσεις δεδομένων και πολλούς άλλους.

**4. Δυνατότητες Επεκτασιμότητας:**

Η Python επιτρέπει την επέκταση των δυνατοτήτων της μέσω της χρήσης modules και πακέτων, επιτρέποντας τη δημιουργία επαναχρησιμοποιήσιμου κώδικα.

**Χρήση της Python σε Εφαρμογές:**

Η Python χρησιμοποιείται ευρέως σε διάφορους τομείς, συμπεριλαμβανομένων:

**1. Ανάπτυξη Ιστοσελίδων:**

Η Flask και η Django είναι δημοφιλή πλαίσια που χρησιμοποιούνται για την ανάπτυξη ιστοσελίδων.

**2. Επιστημονική Υπολογιστική:**

Η Python χρησιμοποιείται στον τομέα της επιστημονικής υπολογιστικής με βιβλιοθήκες όπως η NumPy και η SciPy.

**3. Τεχνητή Νοημοσύνη και Μηχανική Μάθηση:**

Η Python χρησιμοποιείται ευρέως για την ανάπτυξη αλγορίθμων μηχανικής μάθησης με βιβλιοθήκες όπως η TensorFlow και η PyTorch.

**4. Αυτοματοποίηση και Διαχείριση Δεδομένων:**

Η Python χρησιμοποιείται για την αυτοματοποίηση και τη διαχείριση δεδομένων με βιβλιοθήκες όπως η Pandas.

Η Python αντιπροσωπεύει μια ισχυρή επιλογή για προγραμματιστές που αναζητούν μια ευέλικτη, ευανάγνωστη και ισχυρή γλώσσα προγραμματισμού. Με τις δυνατότητές της επιτρέπει τη δημιουργία εφαρμογών από απλές έως πολύπλοκες, καλύπτοντας ένα ευρύ φάσμα εφαρμογών.

## 4.12 Μεταφορά δεδομένων του μικροεπεξεργαστή στον υπολογιστή μέσο της Python

Εισαγωγή

Η επικοινωνία μεταξύ ενός υπολογιστή και ενός μικροεπεξεργαστή μπορεί να επιτευχθεί με διάφορους τρόπους. Στο σημείο αυτό, θα εξετάσουμε πώς μπορούμε να επικοινωνήσουμε με τον μικροεπεξεργαστή PSoC 6 χρησιμοποιώντας τη γλώσσα προγραμματισμού Python και τη σειριακή επικοινωνία μέσω UART.

Σύνδεση με τον Μικροεπεξεργαστή PSoC 6

Πριν ξεκινήσουμε την επικοινωνία, πρέπει να εξασφαλίσουμε ότι ο PSoC 6 συνδέεται σωστά με τον υπολογιστή μας. Η σύνδεση μπορεί να γίνει μέσω θύρας UART. Επίσης, πρέπει να απαιτούνται οι αναγκαίες βιβλιοθήκες Python.

Επικοινωνία μέσω Σειριακής Επικοινωνίας UART

Η σειριακή επικοινωνία UART είναι μια συνήθης μέθοδος επικοινωνίας μεταξύ υπολογιστή και ενσωματωμένου συστήματος. Η Python, χρησιμοποιεί τη βιβλιοθήκη PySerial για να διευκολύνει τη σειριακή επικοινωνία η μετατροπή και μεταφορά του σειριακού σήματος της UART, αυτή γίνεται μέσο του μετατροπέα USB-to-serial FT234XD οπού αφού ολοκληρωθεί αποστέλλεται στον υπολογιστή μέσο USB.

# Κεφάλαιο 5 - Κατασκευή PCB

## 5.1 Τι είναι ένα PCB

Το PCB είναι τη συντομογραφία του "Printed Circuit Board," που σημαίνει "Πλακέτα Εκτυπωμένου Κυκλώματος." Ένα PCB είναι μία πλακέτα που χρησιμοποιείται για να στηρίξει και να συνδέσει ηλεκτρονικά εξαρτήματα σε ένα ηλεκτρονικό κύκλωμα. Το PCB αποτελείται από μια επίπεδη βάση με στρώσεις χαλκού που αποτελεί καλό αγωγό του ηλεκτρισμού.

Το μονωτικό υλικό σε ένα PCB συνήθως είναι το FR4 (flame retardant).

Η χρησιμότητα ενός PCB περιλαμβάνει τα παρακάτω:

**Στήριξη Εξαρτημάτων:** Τα ηλεκτρονικά εξαρτήματα, όπως αντιστάσεις, πυκνωτές, μικροεπεξεργαστές, και τρανζίστορ, τοποθετούνται στο PCB.

**Σύνδεση Εξαρτημάτων:** Ο χαλκός πάνω στο PCB χρησιμοποιείται για τη σύνδεση των εξαρτημάτων μεταξύ τους, δημιουργώντας τους ηλεκτρικούς δρόμους που επιτρέπουν την ροή των ηλεκτρονίων.

**Προστασία:** Το PCB παρέχει προστασία από πολλούς παράγοντες, συμπεριλαμβανομένης της σκόνης, της υγρασίας και της μηχανικής φθοράς. Επίσης, μπορεί να προστατεύσει τα ηλεκτρονικά εξαρτήματα από ηλεκτροστατική εκφόρτιση και άλλες φθορές.

**Οργάνωση:** Πάνω σε ένα PCB τα εξαρτήματα τοποθετούνται με έναν λογικό και δομημένο τρόπο, καθιστώντας ευκολότερη τη συναρμολόγηση και τη συντήρηση του ηλεκτρονικού κυκλώματος.

Τα PCB έχουν αντικαταστήσει παντού οποιαδήποτε διάτρητη πλακέτα αποτελώντας θεμελιακό στοιχείο της σύγχρονης τεχνολογίας.

## 5.2 Γιατί να επιλέξω σε μια πλακέτα να χρησιμοποιήσω τυπωμένο κύκλωμα (PCB)

Η κατασκευή πλακέτας εκτυπωμένου κυκλώματος (PCB) μπορεί να προσφέρει πολλά οφέλη, ειδικά όταν πρόκειται για την ανάπτυξη ηλεκτρονικών συσκευών. Ορισμένοι λόγοι για τους οποίους ενδείκνυται η κατασκευή μιας PCB πλακέτας περιλαμβάνουν:

**Επαγγελματική Εμφάνιση και Σταθερότητα:** Η χρήση μιας προσαρμοσμένης PCB πλακέτας παρέχει μια επαγγελματική και σταθερή εμφάνιση στο τελικό προϊόν, καθώς τα εξαρτήματα είναι τοποθετημένα με ακρίβεια.

**Βελτιωμένη Απόδοση Κυκλώματος:** Η χρήση PCB επιτρέπει τη συγκεκριμένη διάταξη και σύνδεση των εξαρτημάτων, βελτιώνοντας την απόδοση του κυκλώματος και μειώνοντας τις αποστάσεις των συνδέσεων.

**Συνολική Οικονομία Χώρου:** Μια πλακέτα PCB επιτρέπει τη συγκέντρωση των εξαρτημάτων σε μικρότερο χώρο, βοηθώντας στη δημιουργία φορητών και πιο συμπαγών συσκευών.

**Ευκολία στην Συναρμολόγηση και Συντήρηση:** Η χρήση PCB καθιστά ευκολότερη τη συναρμολόγηση, επισκευή και συντήρηση του ηλεκτρονικού συστήματος, καθώς οι συνδέσεις δε δημιουργούν ένα χαοτικό κύκλωμα.

**Προσαρμοσμένη Σχεδίαση:** Ένα PCB μπορεί να προσαρμοστεί στις ακριβείς ανάγκες μιας εφαρμογής, συμπεριλαμβανομένων των διαστάσεων της διάταξης, και των χαρακτηριστικών.

**Διευκόλυνση Σειράς Παραγωγής:** Όταν έχει σχεδιαστεί μια τυπωμένη πλακέτα που λειτουργεί σωστά, μπορεί να αναπαραχθεί εύκολα πολλές φορές για μαζική παραγωγή.

Ο σχεδιασμός του PCB πραγματοποιήθηκε μέσο του λογισμικού KiCad

## 5.3 Το λογισμικό σχεδίασης τυπωμένων κυκλωμάτων KiCad

Το KiCad είναι ένα δωρεάν και ανοιχτού κώδικα (open-source) πακέτο λογισμικού για τον σχεδιασμό ηλεκτρονικών κυκλωμάτων (EDA - Electronic Design Automation). Παρέχει ένα ολοκληρωμένο σύνολο εργαλείων για τον σχεδιασμό σχηματικών, το τυπωμένο κύκλωμα (PCB), την προσομοίωση κυκλωμάτων με τη βοήθεια του SPICEκαι τη δημιουργία βιβλιοθηκών εξαρτημάτων.

Ορισμένα βασικά χαρακτηριστικά του KiCad περιλαμβάνουν:

**Σχεδιασμός σχηματικού κυκλώματος:** Η δημιουργία σχηματικών διαγραμμάτων για τον σχεδιασμό της λογικής δομής του κυκλώματος.

**Σχεδίαση PCB:** Η δημιουργία της φυσικής διάταξης των εξαρτημάτων σε μια πλακέτα εκτυπωμένου κυκλώματος.

**Δημιουργία Βιβλιοθηκών Εξαρτημάτων:** Η δημιουργία προσαρμοσμένων βιβλιοθηκών εξαρτημάτων για χρήση σε σχηματικά και πλακέτες PCB.

**Προσομοίωση Κυκλωμάτων:** Η προσομοίωση της συμπεριφοράς των κυκλωμάτων πριν από την πραγματική κατασκευή.

**Εξαγωγή Αρχείων Παραγωγής:** Η δημιουργία αρχείων που χρειάζονται για την παραγωγή των πλακετών, όπως τα αρχεία Gerber και Drill.

**Κοινότητα Χρηστών και Υποστήριξη:** Η ύπαρξη μιας ενεργής κοινότητας χρηστών και φόρουμ που παρέχουν υποστήριξη και ανταλλαγή ιδεών.

Οι χρήστες μπορούν να χρησιμοποιήσουν το KiCad για τη δημιουργία και τον σχεδιασμό ηλεκτρονικών πλακετών, από απλά κυκλώματα έως και πολύπλοκες εφαρμογές. Λόγω του ότι είναι δωρεάν και ανοιχτού κώδικα, το KiCad έχει γίνει δημοφιλές σε ένα ευρύ φάσμα χρηστών, από ερασιτέχνες μέχρι επαγγελματίες σχεδιαστές.

## 5.4 Symbols στο kiCad

Στο KiCad, τα "symbols" (σύμβολα) αναφέρονται στα γραφικά σχήματα που χρησιμοποιούνται για την αναπαράσταση ενός ηλεκτρονικού εξαρτήματος σε ένα σχηματικό διάγραμμα. Τα symbols καθορίζουν τη λογική συμπεριφορά του εξαρτήματος και πώς αυτό συνδέεται με άλλα εξαρτήματα στο κύκλωμα.

Κάθε ηλεκτρονικό εξάρτημα έχει ένα αντίστοιχο symbol που το αναπαριστά στο σχηματικό διάγραμμα. Για παράδειγμα, ένας απλός αντιστάτης μπορεί να έχει ένα σύμβολο που δείχνει μια γραμμή με μια αγκύλη, ενώ ένας μικροεπεξεργαστής μπορεί να έχει ένα πολύπλοκο σύμβολο που αντιπροσωπεύει τις διάφορες ακίδες και λειτουργίες του.

Η χρήση σωστών symbols είναι σημαντική για την ακριβή αναπαράσταση του κυκλώματος στο σχηματικό διάγραμμα, καθώς και για τη σωστή λογική σύνδεση με άλλα εξαρτήματα.

Το KiCad περιλαμβάνει μια εκτενή βιβλιοθήκη με προκαθορισμένα symbols για πολλά διαδεδομένα ηλεκτρονικά εξαρτήματα. Επιπλέον μπορούν να δημιουργηθούν

προσαρμοσμένα symbols για εξαρτήματα που δεν υπάρχουν στις προκαθορισμένες βιβλιοθήκες ή για να προσαρμοστούν τα υπάρχοντα symbols στις ανάγκες του έργου.

## 5.5 Τι είναι τα footprints στο kiCad

Στο KiCad, τα "footprints" αποτελούν το αποτύπωμα των εξαρτημάτων πάνω σε ένα PCB. Τα footprints περιλαμβάνουν τη γεωμετρία, τις διατάξεις των ακίδων, και άλλες πληροφορίες που καθορίζουν πώς το εξάρτημα θα τοποθετηθεί στην πλακέτα PCB.

Κάθε symbol μπορεί να έχει ένα ή περισσότερα διαθέσιμα footprints.

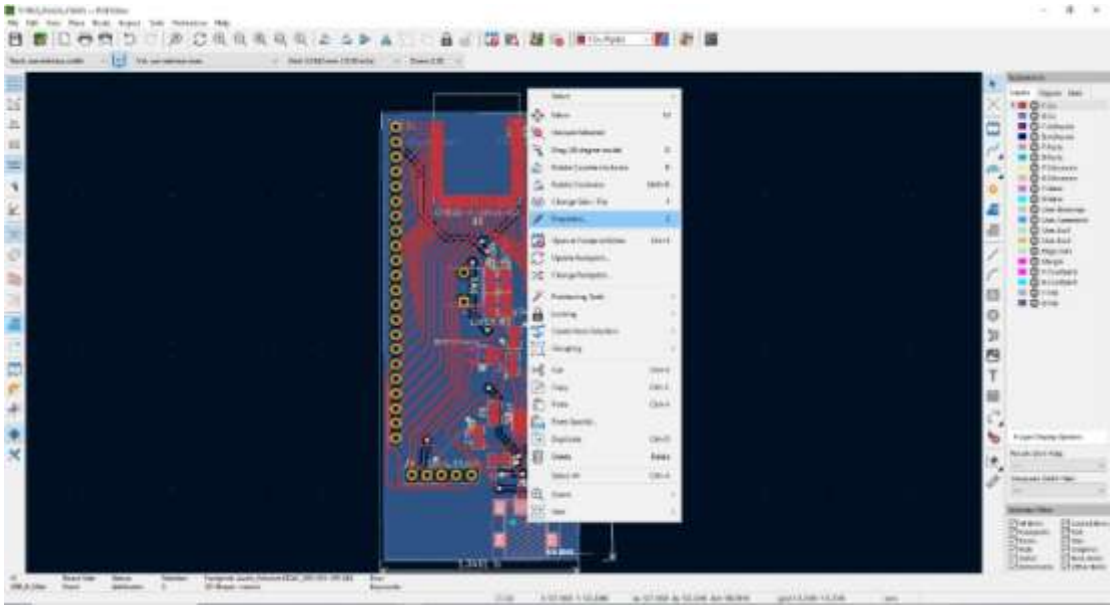
Το KiCad περιλαμβάνει μια εκτενή βιβλιοθήκη με footprints που καλύπτει μεγάλο φάσμα ηλεκτρονικών εξαρτημάτων, αλλά μπορεί επίσης ο χρήστης να δημιουργήσει τα δικά του custom footprints για προσαρμογή στις ανάγκες του εκάστοτε έργου.

**Πως να επεξεργαστείτε ένα αποτύπωμα εξαρτήματος:**

Ένα παράδειγμα για την επεξεργασία των αποτυπωμάτων.

Η θύρα USB έχει πέντε ακροδέκτες οι οποίοι έχουν πλάτος 0.5mm και μήκος 2.3mm. έστω ότι πρέπει να αυξηθεί το μήκος τον ακροδεκτών σε 2.5mm ώστε να εχει πιο εύκολη συγκόλληση με την πλακέτα PCB.

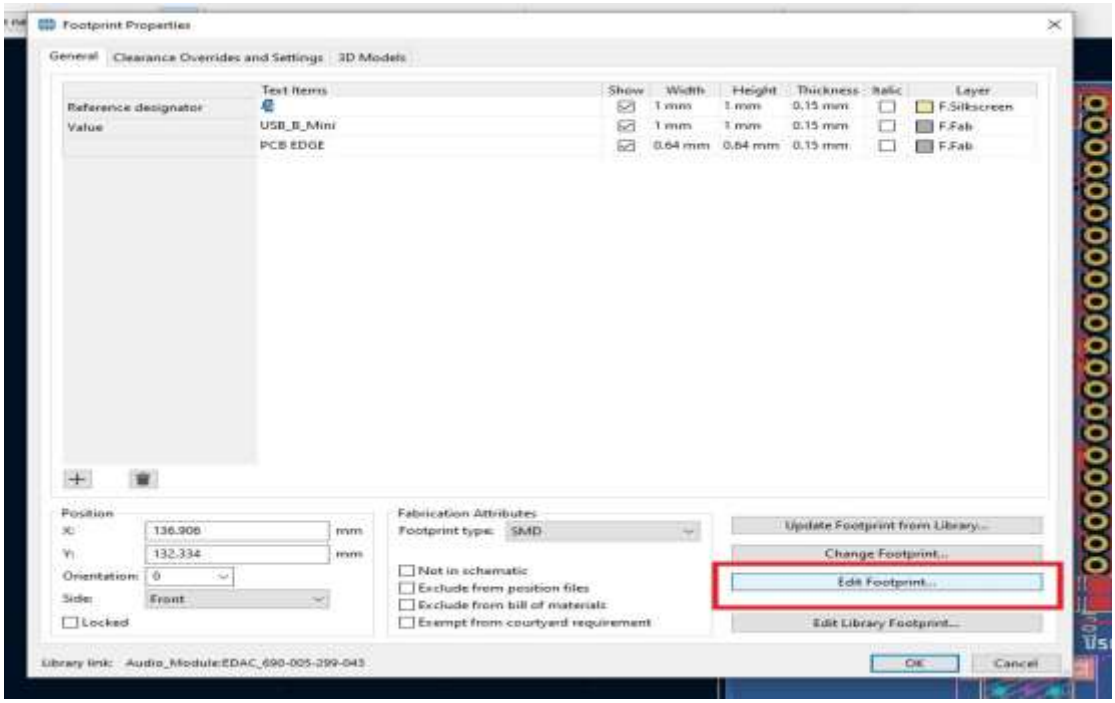
Αφού εντοπίσει την θύρα USB, κάνει δεξί κλικ  (Εικ. 2) οπού έπειτα επιλέγει το «Properties»



Εικ. 2 Επεξεργασία αποτυπώματος

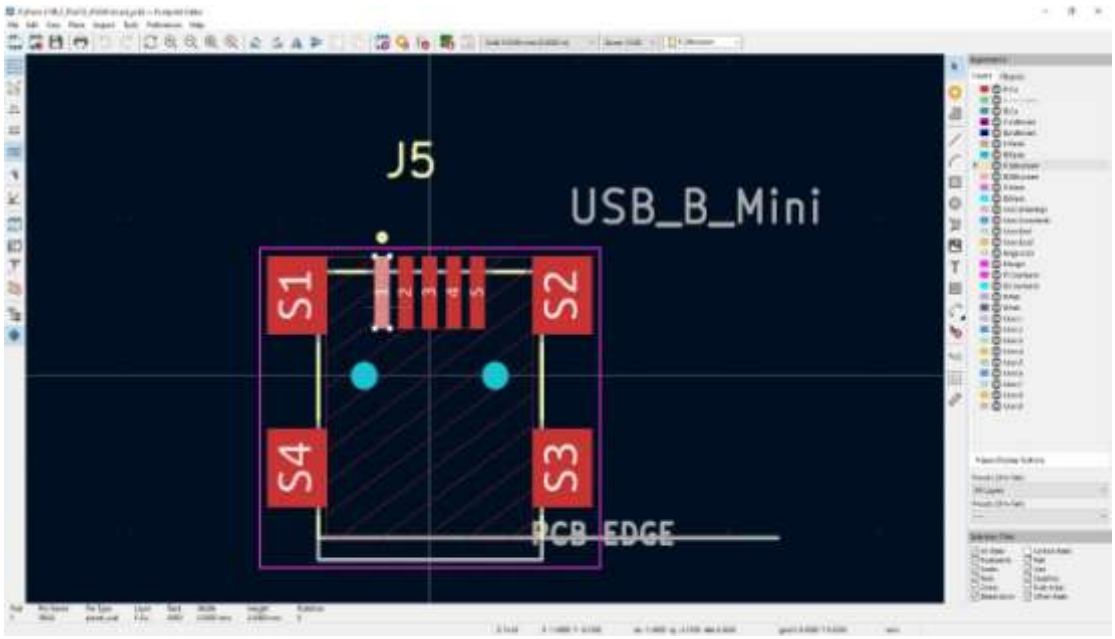
Αφού ανοίξει η καρτέλα «Footprint properties» κάνει κλικ στο «Edit footprint»

(Εικ. 3).




Εικ. 3 Ρυθμίσεις αποτυπώματος

Έπειτα πατώντας πάνω στους ακροδέκτες νούμερο 1,2,3,4,5 μπορεί να αλλάξει το μέγεθος του ακροδέκτη που έχει επιλέξει (Εικ. 4).




Εικ. 4 Τροποποίηση αποτυπώματος

## 5.6 Τι είναι τα models στο kiCad

Στο KiCad, τα "models" αναφέρονται στα μοντέλα ηλεκτρονικών εξαρτημάτων που χρησιμοποιούνται για την προσομοίωση κυκλωμάτων. Τα μοντέλα αυτά παρέχουν λεπτομερείς πληροφορίες σχετικά με τη συμπεριφορά των εξαρτημάτων όταν αυτά είναι συνδεδεμένα σε ένα κύκλωμα.

Τα μοντέλα εξαρτημάτων μπορούν να περιλαμβάνουν πληροφορίες όπως η αντίσταση, η χωρητικότητα, η τάση, το ρεύμα και άλλα χαρακτηριστικά που επηρεάζουν τη συμπεριφορά του εξαρτήματος στο κύκλωμα.

Τα μοντέλα αυτά είναι κρίσιμα για την προσομοίωση κυκλωμάτων, καθώς επιτρέπουν στους σχεδιαστές να εκτιμήσουν το πώς θα συμπεριφερθεί το κύκλωμα υπό διάφορες συνθήκες και φορτία.

## 5.7 Διαθέσιμα Footprints

- Στο KiCad Library GitHub Repository μπορεί κάποιος να βρει και να κατεβάσει επιπλέον βιβλιοθήκες (https://github.com/kicad/kicad-library).
- Στο Snapeda μπορεί επίσης κάποιος να κατεβάσει επιπλέον βιβλιοθήκες (https://www.snapeda.com/kicad/)

Εάν απαιτούνται προσαρμοσμένα footprints, μπορεί επίσης κάποιος να δημιουργήσει τα δικά του footprints, χρησιμοποιώντας το εργαλείο "FootprintEditor" που παρέχει το KiCad.

## 5.8 3DViewer στο KiCad

Το "3D View" στο KiCad είναι μια λειτουργία που επιτρέπει να προβληθεί το σχέδιο όχι μόνο σε διάφορα σχηματικά και λειτουργικά περιβάλλοντα, αλλά και σε ένα τρισδιάστατο περιβάλλον. Αυτό σημαίνει ότι μπορεί κάποιος να δει πώς θα φαίνεται η πλακέτα PCB και τα εξαρτήματά της στην πραγματικότητα.

Ορισμένα από τα βασικά χαρακτηριστικά του 3D View στο KiCad περιλαμβάνουν:

**Προβολή της Πλακέτας PCB σε 3D:**

Στην προβολή σε 3D μπορεί να δει πως θα φαίνεται στον πραγματικό κόσμο η πλακέτα, συμπεριλαμβανομένων των χαλκών, των ιχνών και άλλων χαρακτηριστικών.

**Προβολή Εξαρτημάτων σε 3D:**

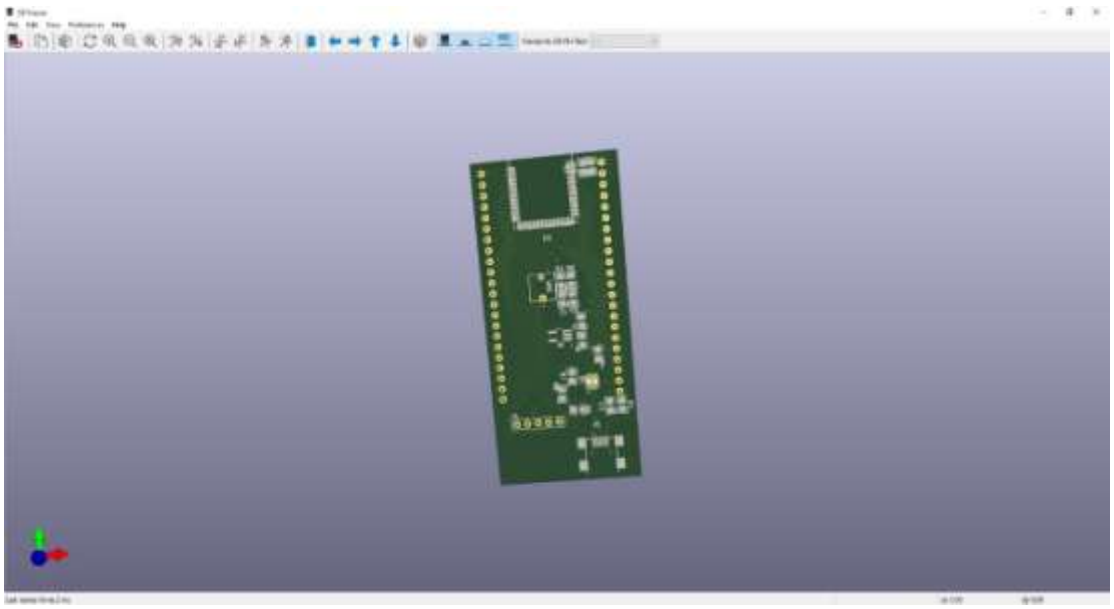
Τα εξαρτήματα στο σχηματικό διάγραμμα μπορούν να εμφανιστούν ως 3D μοντέλα, επιτρέποντας να προβάλλεται πώς θα είναι τοποθετημένα στην πλακέτα PCB.

**Εξαρτήματα με οπές:**

Αν τα εξαρτήματα έχουν οπές (π.χ., οπές για βίδες ή εξαρτήματα συμβατικής τεχνολογίας – ThroughHoleTechnology), αυτές μπορούν επίσης να φαίνονται στο 3D View.

**Ρύθμιση Προεπισκόπησης Υλικών:**

Μπορούν να ρυθμιστούν χρώματα, τα υλικά και άλλες παραμέτρους που επηρεάζουν την εμφάνιση των αντικειμένων στο 3D View. Η λειτουργία 3D View είναι χρήσιμη για να παρει μια πιο ρεαλιστική εικόνα το πρότζεκτ και να ελεγχθούν ενδεχόμενα προβλήματα πριν από την πραγματική κατασκευή της πλακέτας PCB (Εικ. 5).



Εικ. 5 3D View KiCad 7.0

## 5.9 Schematic στοKicad

Σχεδίαση PCB: Η Σημασία του Schematic

Κατά την κατασκευή ενός PCB, το schematic αποτελεί το θεμέλιο για τη δημιουργία ενός λειτουργικού κυκλώματος. Σε αυτό το τμήμα, θα εξετάσουμε τη σημασία του Schematic και πώς καθορίζει τη δομή του PCB.

Τι είναι το Schematic;

Το Schematic σε ένα πρόγραμμα σχεδίασης PCB είναι το σχηματικό διάγραμμα του κυκλώματος. Παρουσιάζει γραφικά τα ηλεκτρονικά συστατικά, όπως αντιστάσεις, πυκνωτές, ενδείκτες και συνδέσεις μεταξύ τους.

Σχεδίαση της Δομής του Κυκλώματος

Ένα καλό Schematic προσφέρει μια οπτική αναπαράσταση της δομής του κυκλώματος. Ο σχεδιαστής επιλέγει συμβολισμούς για τα εξαρτήματα και καθορίζει τις απαραίτητες συνδέσεις.

Η Σημασία του για την Κατασκευή του PCB

Το schematic αποτελεί τον καθοριστικό παράγοντα για το **routing**, δηλαδή τη σύνδεση των ηλεκτρονικών εξαρτημάτων με αγωγούς στην επιφάνεια του PCB. Αποτελεί τον οδηγό για τη δημιουργία των αγωγών που εξασφαλίζουν τη σωστή λειτουργία του κυκλώματος.

Με την κατανόηση της σημασίας του schematic, ο σχεδιαστής PCB (KiCad) μπορεί να προχωρήσει σε ένα αξιόπιστο και αποτελεσματικό σχεδιασμό, δημιουργώντας PCB που ανταποκρίνονται στις απαιτήσεις του.

## 5.10 Περιγραφή του schematic

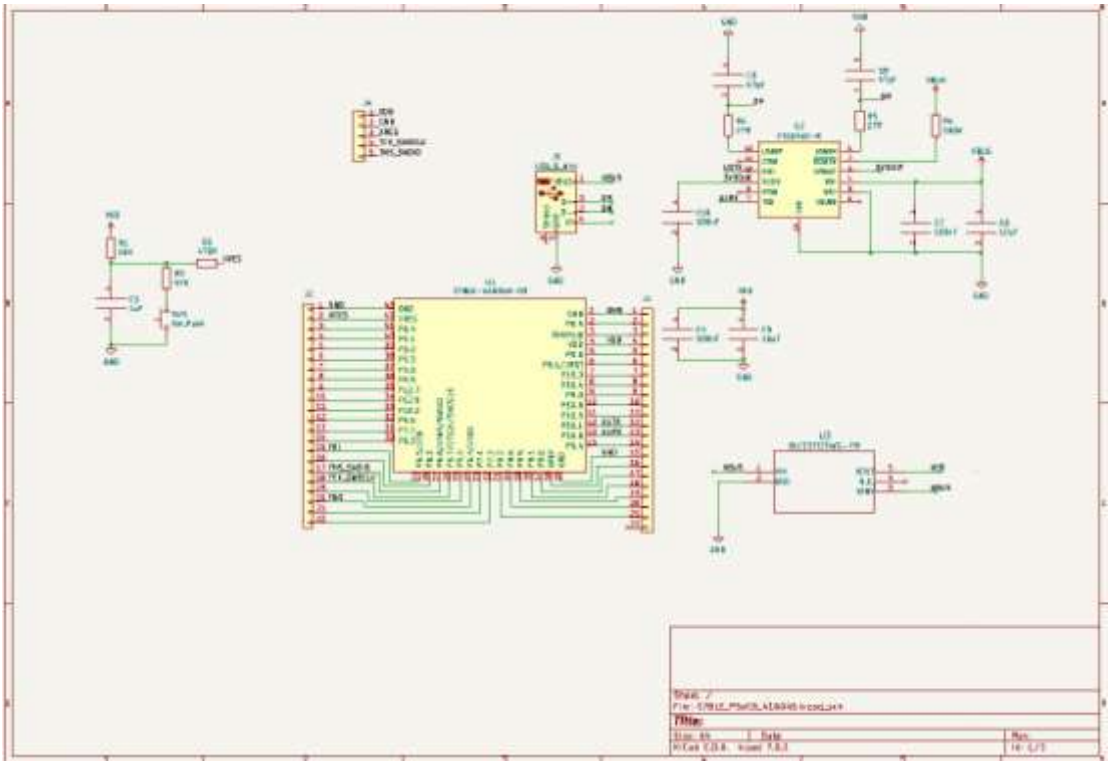
Η παρουσίαση του παρακάτω σχηματικού (Εικ. 6) αποσκοπεί στην αναπαράσταση ενός ηλεκτρικού κυκλώματος που σχεδιάστηκε για τη δημιουργία ενός αναλογικού σήματος. Ακολούθως, το κύκλωμα υπόκειται σε ανάλυση του προκειμένου να εξαχθεί το επιθυμητό ψηφιακό σήμα. Στη συνέχεια, το παραγόμενο σήμα μεταφέρεται στον υπολογιστή για την απεικόνισή του, προσφέροντας μια οπτική αναπαράσταση που διευκολύνει την κατανόηση και την παρακολούθηση του κυκλώματος.

**Περιγραφή των συνδέσεων:**

Ξεκινά από το PSoC6 το οποίο στον ακροδέκτη νούμερο 42 (XRES) συνδέεται ένας διακόπτης τύπου push-button οπού η χρήση του είναι για να κάνει επανεκκίνηση το PSoc6, για την ασφαλής λειτουργία του PSoc6 απαιτείται το push-button να συνδεθεί σε κύκλωμα με τρεις αντιστάσεις και ένα πυκνωτή(Εικ. 6).

Στο σημείο αυτό η θύρα USB, η οποία συνδέεται με τον μετατροπέα USB-to-serial FT234XD. Αυτό είναι απαραίτητο για την επεξεργασία των πληροφοριών που εξάγονται από το PSoc6 προτού αποσταλούν σε έναν υπολογιστή. Ο μετατροπέας συνδέεται με το PSoC6 στον ακροδέκτη νούμερο 12 (P10.1) για την αποστολή πληροφοριών, καθώς και στον ακροδέκτη νούμερο 13 (P10.0) για την είσοδο πληροφοριών από το FT234XD.

Επιπλέον στο κύκλωμα υπάρχει ο σταθεροποιητής τάσης BU33TD3WG. Ο ρόλος του οποίου είναι να μειώνει την τάση τροφοδοσίας από τα 5 Volts σε 3.3 Volts. Αφού ολοκληρωθεί η μείωση της τάσης, συνδέεται με τον ακροδέκτη νούμερο 4 (VDD) του PSoC6. Ο BU33TD3WG αποτελεί κρίσιμο στοιχείο για τη διατήρηση της σωστής τάσης και εξασφαλίζει ασφαλή λειτουργία του PSoc6.



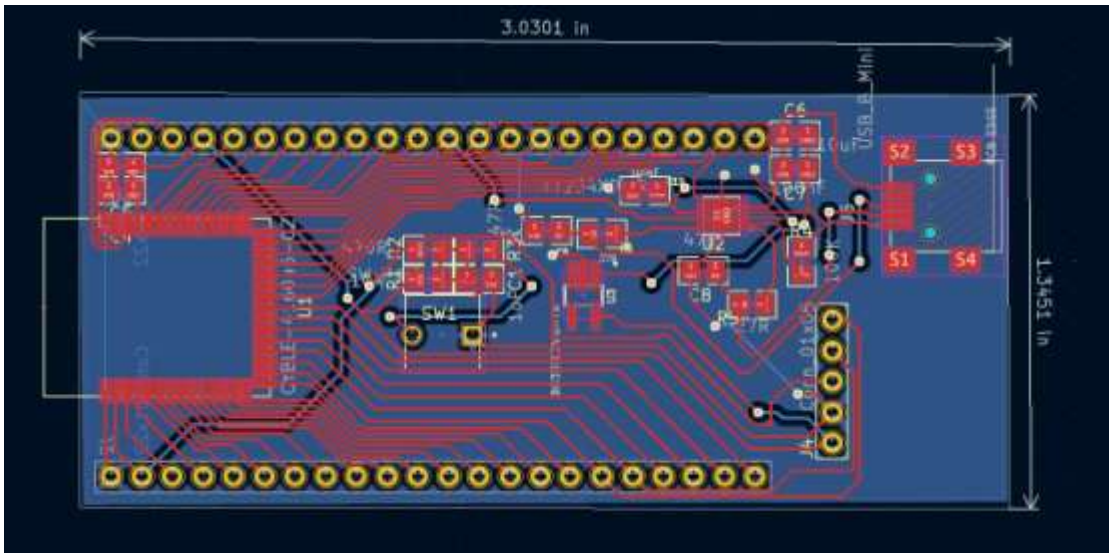
Εικ. 6 KiCad 7.0 Schematic

## 5.11 Δρομολόγηση συνδέσεων ανάμεσα στα υλικά του PCB

Στον χώρο των κυκλωμάτων, το "routing" αναφέρεται στη διαδικασία σύνδεσης των ηλεκτρονικών εξαρτημάτων (όπως αντιστάσεις, πυκνωτές, κ.λπ.) μεταξύ τους μέσω αγωγών που χαράσσονται στην επιφάνεια ενός PCB (PrintedCircuitBoard).

Κατά το routing, ο σχεδιαστής πρέπει να λαμβάνει υπόψη πολλούς παράγοντες όπως το μήκος των αγωγών (για να αποφευχθούν προβλήματα χρονισμού), αποφυγή εμποδίων, και ελαχιστοποίηση ηλεκτρομαγνητικών παρεμβολών.

Οι παρακάτω εικόνα μας δείχνει το routing για της ανάγκες της πτυχιακής εργασίας (Εικ. 7)

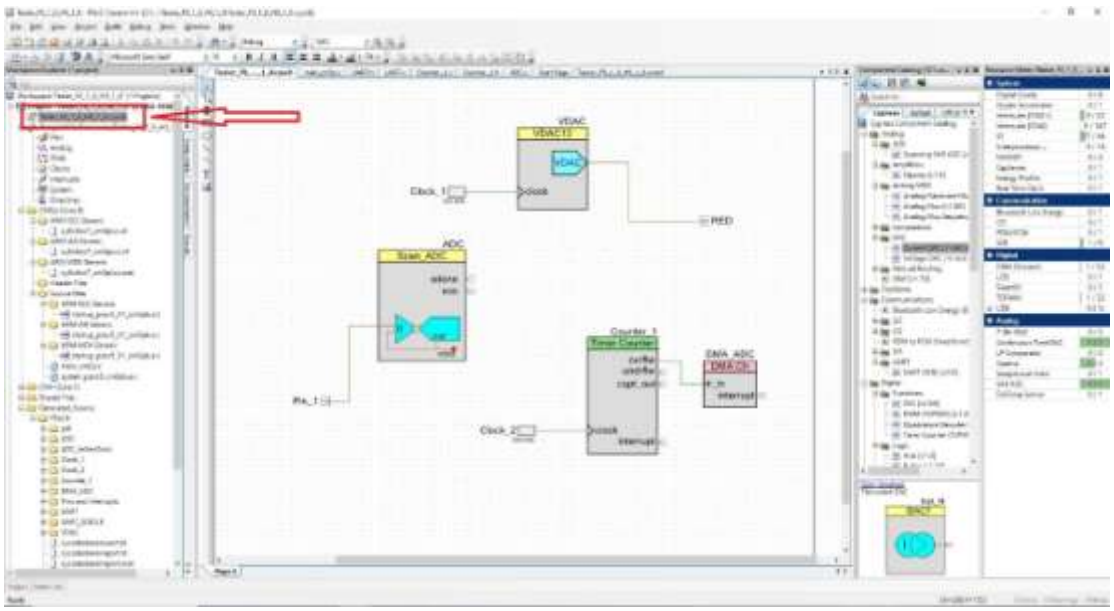



Εικ. 7 Routing στο KiCad 7.0

# Κεφάλαιο 6 – Software

## 6.1 Ανάλυση των Components

Για την υλοποίηση του firmware στο PSoC6 χρησιμοποιήθηκαν κάποια components όπως τον ADC, τον TimerCounter, τον DAC και μία μονάδα DMA. Μπορείτε να δείτε όλα τα components που χρησιμοποιήθηκαν στην καρτέλα Tester_F0_1_0_H0_1_0.cysch βλέπε (Εικ. 8)




Εικ. 8 Components που χρησιμοποιούνται

Ας εξετάσουμε ένα ένα τα components και της ρυθμίσεις που έχουν οριστεί. Ξεκινώντας από τον DAC ο οποίος είναι υπεύθυνος για την δημιουργία του αναλογικού σήματος για σκοπούς αποσφαλμάτωσης και επίδειξης λειτουργίας του αναλυτή φάσματος. Με διπλό κλικ στο εικονίδιο της VDAC θα ανοίξει το παράθυρο που μπορεί να επεξεργαστεί μερικές λειτουργίες που μας προσφέρει το συγκεκριμένο component.

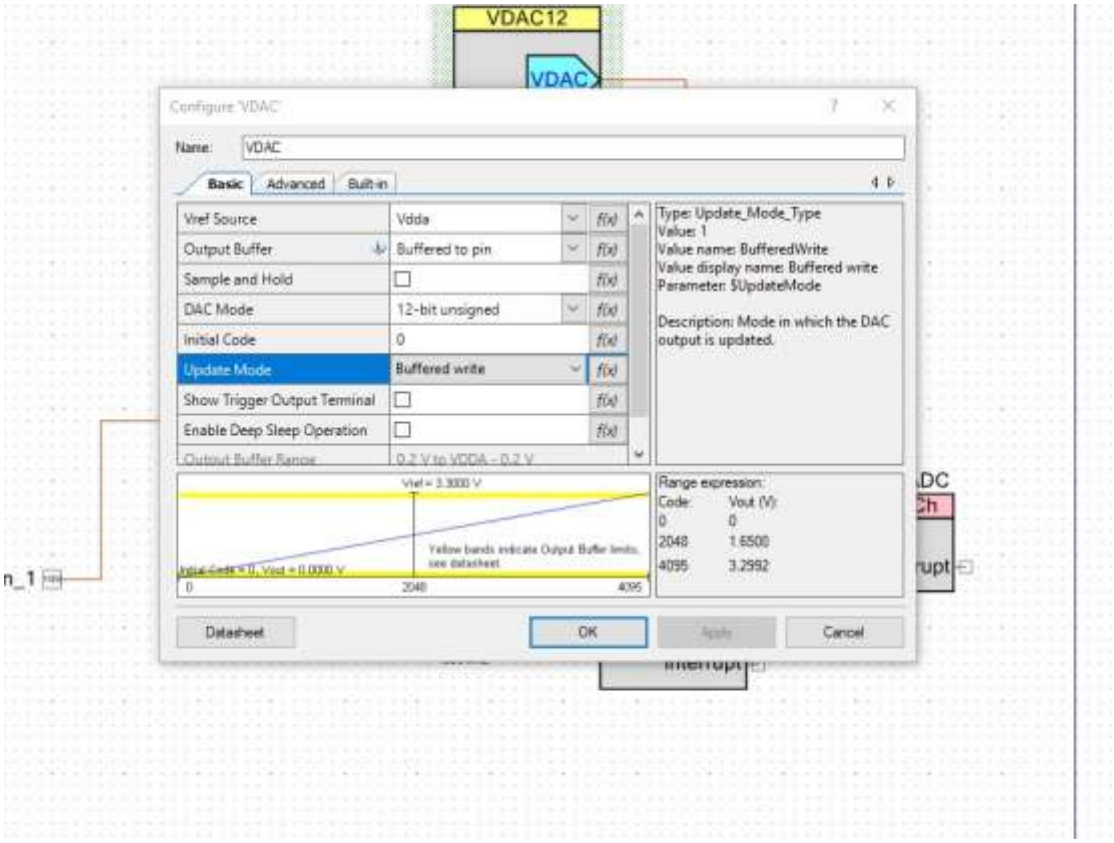
Οι πιο σημαντικές ρυθμίσεις είναι να οριστεί το VrefSource σε Vdda το οποίο καθορίζει τη θετική τάση αναφοράς για τη λειτουργία του DAC.

Μια άλλη σημαντική ρύθμιση είναι να οριστεί το OutputBuffer σε Bufferedtopin το οποίο συμβολίζει η έξοδος της DAC ( δηλαδή το αναλογικό σήμα ) να κατευθύνεται σε κάποιο συγκεκριμένο pin, στην δική μας περίπτωση έχουμε το pin 9[6] του PSoC6.

Μια ακόμη σημαντική ρύθμιση είναι το Update mode το οποίο είναι επιλεγμένο το Buffered write το οποίο δίνει την δυνατότητα να αλλάξει η DAC το μέσο εξόδου.

Και τέλος για να λειτουργήσει ο DAC θα πρέπει να συνδεθεί ένα ρολόι το οποίο θα έχει συχνότητα την μέγιστη δυνατή συχνότητα που επιτρέπεται από τον κατασκευαστή και η συχνότητα αυτή είναι τα 500khz.

Όλες οι ρύθμισης του DAC βρίσκονται στην εικόνα 9.



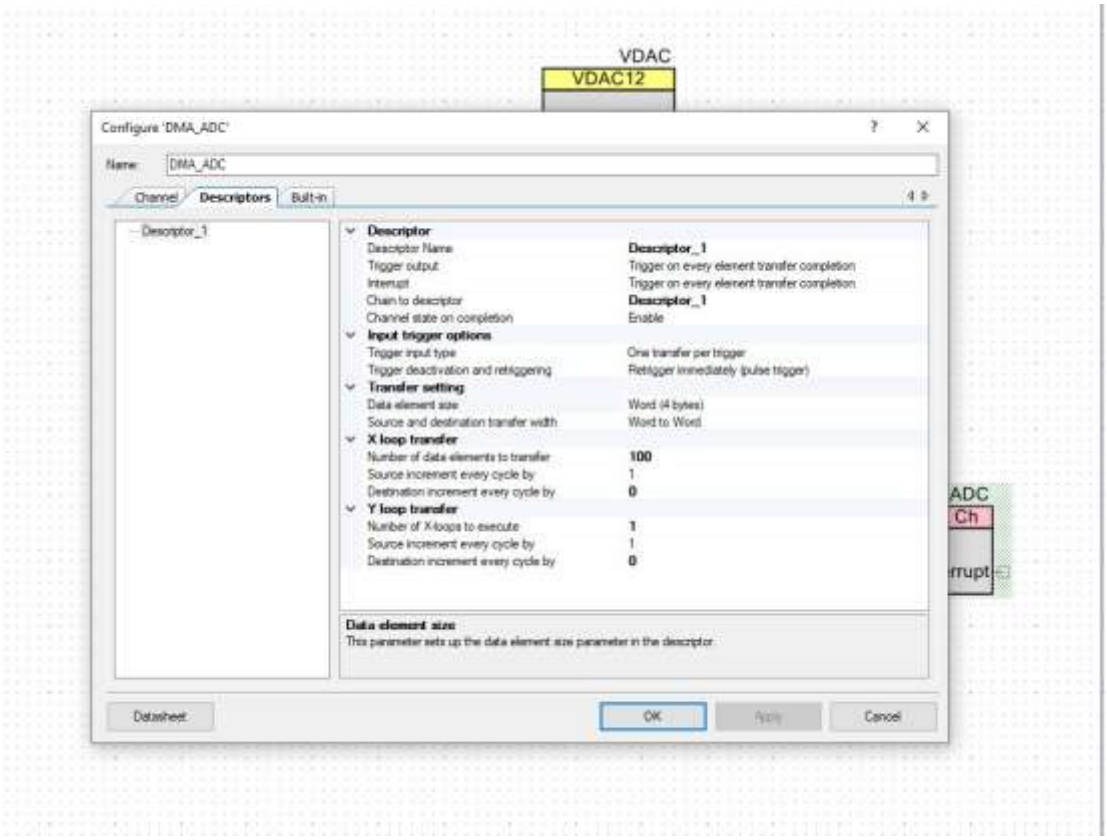

Εικ. 9 Οι ρυθμίσεις του DAC

Συνεχίζοντας με την ανάλυση της DMA η οποία η λειτουργία της είναι να πάρει έναν πίνακα 1024 θέσεων με ακέραιες τιμές που συμβολίζουν ένα σήμα και να δώσει αυτές τις τιμές μια μια στον DAC ώστε να δημιουργήσει το αναλογικό σήμα.

Στο Descriptor Name ορίζετε το όνομα του descriptor δηλαδή (Descriptor_1) και στο Chain to Descriptor μπαίνει το όνομα που δόθηκε στο descriptor (Descriptor_1) , αυτό γίνεται διότι η DMA μας δίνει την δυνατότητα για πολλαπλούς Descriptor και με την διαφορετική ονομασία είναι ξεκάθαρο ποιες οδηγίες έχει ο κάθε Descriptor

- έπειτα ορίζει στο Transfer setting την επιλογή word 4 bytes,
- έπειτα ορίζει πόσες μεταβλητές θα είναι η είσοδος του πίνακα στην συγκεκριμένη περίπτωση ο πίνακας θα αποτελείται από εκατό μεταβλητές οπού αυτές οι εκατό μεταβλητές θα απεικονίζουν έναν παλμό,
- έπειτα στο X loop transfer ορίζουμε το Βήμα σε ένα από την επιλογή Source increment every cyrcle, αυτό το γίνεται για να δώσει η DMA την οδηγία ότι τις εκατό μεταβλητές να της στείλει μια-μια.
- Έπειτα στο Y loop transfer ορίζει πόσες επαναλήψεις θα κάνει το X loop και ορίζει το Source increment every cyrcle και το Number of X-loops to execute σε ένα

Όλες οι παραπάνω ρύθμισης της DMA βρίσκονται στην εικόνα 10.




Εικ. 10 Οι ρυθμίσεις της DMA

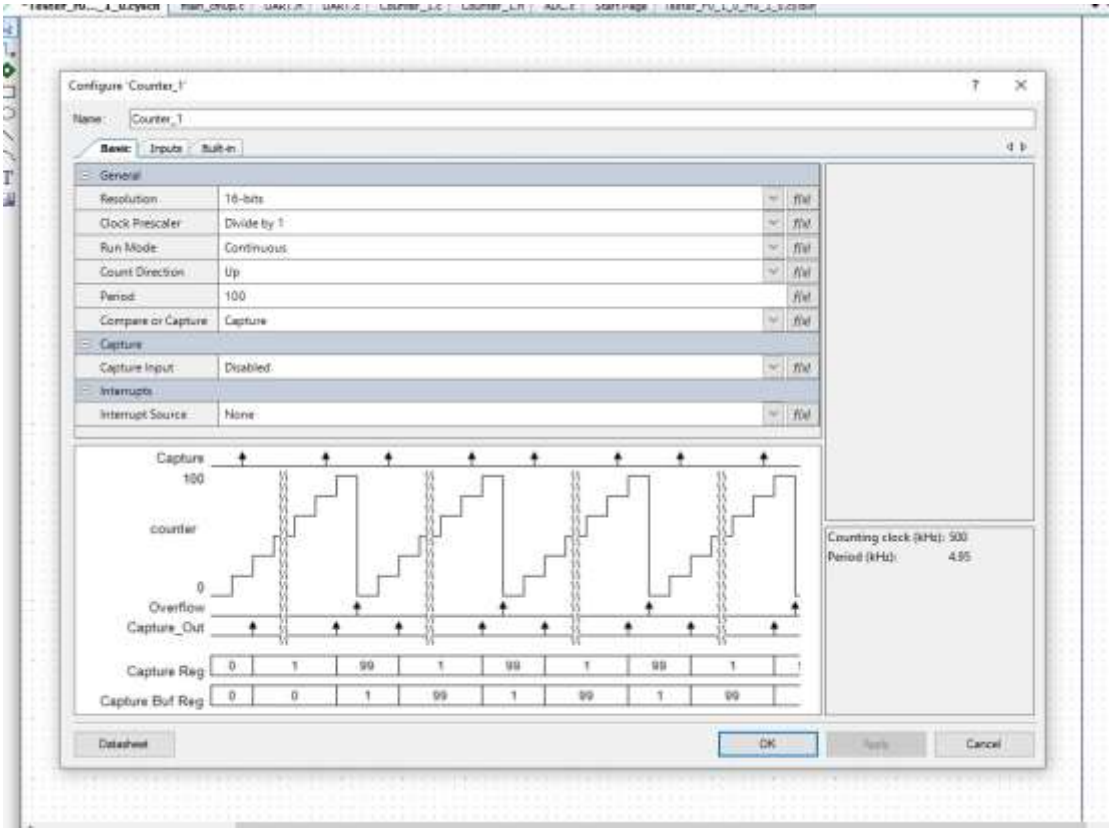
Συνεχίζουμε με την ανάλυση του Timer Counter όπου η λειτουργία του είναι να ελέγχει την συχνότητα με την οποία η μονάδα DMA θα μεταφέρει τις τιμές στον DAC άρα συνεπώς μέσω του Timer counter ελέγχει την συχνότητα την οποία τα σήματα θα δημιουργούνται από τον DAC.

Η πιο σημαντική ρύθμιση είναι το Period το οποίο ορίζεται στην αρχή και επίσης τροποποιείται μέσω του κώδικα ώστε να λαμβάνει τα σήματα στις επιθυμητές συχνότητες.

Μια ακόμη σημαντική μεταβλητή είναι το Run Mode το οποίο ορίζεται σε continuous γιατί χρειάζεται ο timer να τρέχει πάντα και να μην τρέξει απλώς μια φορά και να σταματήσει.

Στην δεξιά κάτω μεριά μας δείχνει το ρολόι οπού έχει συχνότητα στα 500khz και διαιρείται με το 100 το οποίο δίνει 5khz η οποία είναι η συχνότητα που ενεργοποιεί την μονάδα DMA ώστε να στείλει την επομένη τιμή.

Όλες οι παραπάνω ρυθμίσεις του Timer Counter βρίσκονται στην εικόνα 11.




Εικ. 11 Οι ρυθμίσεις του TimerCounter

Συνεχίζουμε με την ανάλυση της ADC η οποία είναι η πιο σημαντική λειτουργία για την πτυχιακή εργασία καθώς αυτή η λειτουργία διαβάζει το σήμα εισόδου που δημιουργείται από τον DAC είτε από μια εξωτερική πηγή

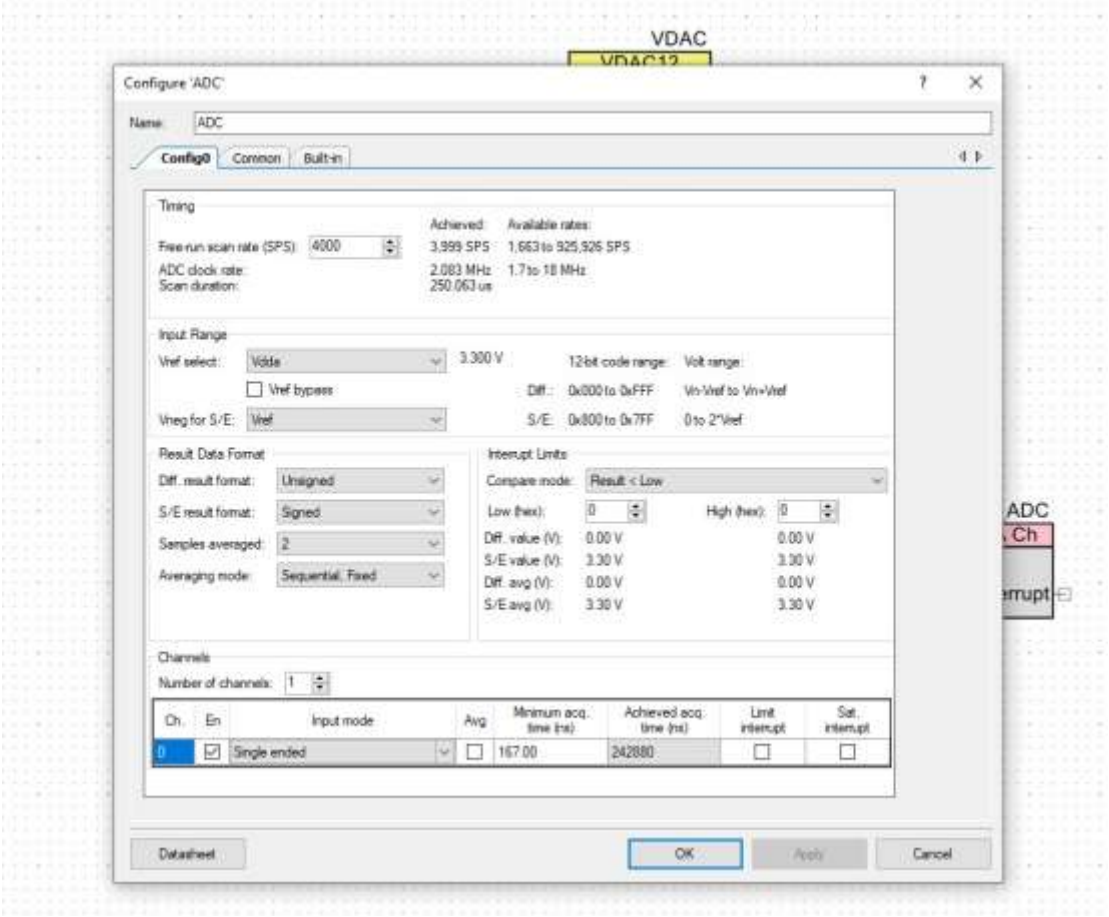
Στο InputRange στην επιλογή Vref ορίζουμε το Vdda,

Στο SPS δηλαδή Samplespersecond ορίζεται στα 4000 δείγματα το δευτερόλεπτο,

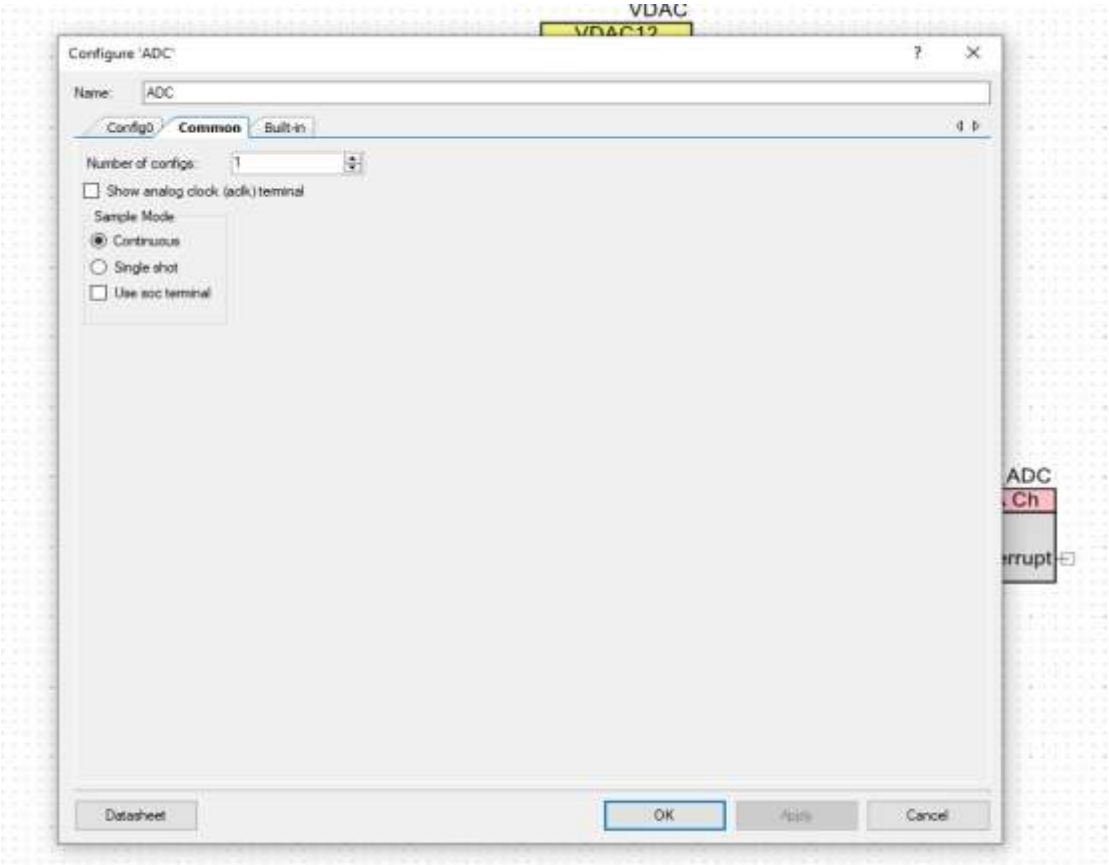
Καιτοchanelσε Single Ended,

Τέλος στην καρτέλα Common στο Samplemode είναι επιλεγμένο το Continuous.

Όλες οι παραπάνω ρυθμίσεις του ADC βρίσκονται στην εικόνα 12 και 13.




Εικ. 12 Οι ρυθμίσεις του ADC




Εικ. 13 Οι ρυθμίσεις του ADC

## 6.2 Ανάλυση του προγράμματος στο Psoc Creator

Ο κώδικας περιέχει μια σειρά από δηλώσεις περιλαμβανομένων αρχείων και βιβλιοθηκών, καθώς και μια συνάρτηση με την ονομασία cooley_tukey_fft. Ας αναλύσουμε κάθε γραμμή του κώδικα:

| |
|---|
| #include "project.h": Αυτή η γραμμή περιλαμβάνει το αρχείο "project.h", το οποίο περιέχει δηλώσεις και ρυθμίσεις που αφορούν το πρόγραμμα ή το σχέδιο του έργου. |
| #include <stdio.h>: Δηλώνει τη χρήση βιβλιοθήκης εισόδου/εξόδου προτύπου C, που παρέχει λειτουργίες εισόδου/εξόδου όπως printf και scanf. |
| #include "ADC.h": Περιλαμβάνει ένα αρχείο με το όνομα "ADC.h", περιέχει λειτουργίες που σχετίζονται με τον αναλογικό μετατροπέα (ADC). |
| #include <stdlib.h>: Δηλώνει τη χρήση της βιβλιοθήκης C Standard Library που περιέχει συναρτήσεις για δυναμική δέσμευση μνήμης και άλλες χρήσιμες |

| |
|---|
| λειτουργίες. |
| #include <math.h>: Δηλώνει τη χρήση της βιβλιοθήκης C για μαθηματικούς υπολογισμούς, περιλαμβάνοντας τη συνάρτηση pow για ύψωση σε δύναμη και άλλες. |
| #include "ctdac/cy_ctdac.h": Περιλαμβάνει ένα αρχείο που σχετίζεται με τον ψηφιακό αναλογικό μετατροπέα (DAC). |
| #include "dma/cy_dma.h": Περιλαμβάνει ένα αρχείο που σχετίζεται με τη διαχείριση μνήμης (DMA). |
| #include <stdint.h>: Δηλώνει τη χρήση της βιβλιοθήκης C για ακέραιους τύπους μεγέθους συγκεκριμένου. |
| #include <string.h>: Δηλώνει τη χρήση της βιβλιοθήκης C για συναρτήσεις σχετικές με τη διαχείριση αλφαριθμητικών. |
| #include <complex.h>: Δηλώνει τη χρήση της βιβλιοθήκης C για υποστήριξη σύνθετων αριθμών. |
| #define PI 3.14159265358979323846: Ορίζει τη σταθερά PI. |
| void cooley_tukey_fft(complex double* x, int n): Δηλώνει τη συνάρτηση cooley_tukey_fft που παίρνει έναν πίνακα σύνθετων διπλών αριθμών x και έναν ακέραιο n. Η συνάρτηση αυτή εκτελεί τον αλγόριθμο Cooley-Tukey για τον υπολογισμό του γρήγορου μετασχηματισμού Fourier (FFT). |
| sineWaveLUT: Περιέχει τιμές που αντιστοιχούν σε ένα ημίτονο. Ουσιαστικά, παριστάνει μια περίοδο του κύκλου της συνάρτησης ημίτονου, χρησιμοποιώντας ακεραίους αριθμούς. |
| squareWave: Είναι ένας πίνακας που περιέχει μηδενικές τιμές μέχρι τη μέση του πίνακα, και μετά έχει τιμή 4092 για το υπόλοιπο του πίνακα. Αυτό προσομοιώνει ένα τετραγωνικό κύμα. |
| triangleWave: Περιέχει τιμές που προσομοιώνουν ένα τριγωνικό κύμα. Οι τιμές αυξάνονται γραμμικά μέχρι τη μέση του πίνακα και μετά μειώνονται γραμμικά. |

| |
|---|
| pulseWave: Είναι παρόμοιος με τον squareWave αλλά έχει μηδενικές τιμές πριν τη μέση του πίνακα. |
| #define ARRAY_SIZE 1024: περιέχει μια καθολική μεταβλητή ARRAY_SIZE που ορίζει το μέγεθος των παραπάνω πινάκων, το οποίο είναι 1024. |
| __enable_irq();: Ενεργοποιεί τις διακοπές (interrupts) στο επίπεδο του λειτουργικού συστήματος. Αυτό επιτρέπει στο πρόγραμμα να ανταποκρίνεται σε διακοπές. |
| VDAC_Start();: Ξεκινά τη λειτουργία του Ψηφιακού-Αναλογικού Μετατροπέα (VDAC). Ο VDAC χρησιμοποιείται για τη μετατροπή ψηφιακών τιμών σε αναλογικά σήματα. |
| UART_Start();: Ξεκινά τη λειτουργία της UART, το οποίο είναι ένας τρόπος επικοινωνίας σειριακών δεδομένων με άλλες συσκευές. |
| ADC_Start();: Ξεκινά τη λειτουργία του Αναλογικού-Ψηφιακού Μετατροπέα (ADC). Ο ADC χρησιμοποιείται για τη μετατροπή αναλογικών σημάτων σε ψηφιακή μορφή. |
| Counter_1_Start();: Ξεκινά τη λειτουργία ενός αριθμητή (counter) ώστε να ορίζει την συχνότητα της DMA. |
| UART_PutString("Debug UART is up and running\n\r");: Στέλνειμιασυμβολοσειράμέσωτου UART. Αυτό χρησιμεύει για να εμφανίσει ένα μήνυμα ώστε να γνωρίζουμε ότι η πλακέτα έχει ξεκινήσει να λειτουργεί. |
| int16_t result;: Δημιουργεί μια μεταβλητή τύπου int16_t με όνομα result. Η μεταβλητή αυτή θα χρησιμοποιηθεί για την αποθήκευση του αποτελέσματος της ανάγνωσης της ADC. |
| int16_t resultmV;: Δημιουργεί μια ακόμη μεταβλητή int16_t με όνομα resultmV. Η μεταβλητή αυτη θα χρησιμοποιηθεί για την αποθήκευση του αποτελέσματος της ADC σε milivolts γιατί η ADC φέρνει τα αποτελέσματα της σε Volts |
| uint32_t chan = 0UL;: Δημιουργεί μια μεταβλητή uint32_t με όνομα chan και αρχικοποιηται σε 0. |

| |
|---|
| complex double valuesArray[ARRAY_SIZE];: Δημιουργεί έναν πίνακας valuesArray με στοιχεία τύπου complex double (πολύπλοκος αριθμός διπλής ακρίβειας) και μέγεθος ARRAY_SIZE οπού θα αποθηκευτούνε τα αποτελέσματα του FFT. |
| int signalArray[ARRAY_SIZE];: Δημιουργεί έναν πίνακας signalArray με στοιχεία τύπου int και μέγεθος ARRAY_SIZE όπου θα αποθηκευτούνε όλες οι τιμές που θα διαβάσει η ADC. |
| uint16_t index = 0;: Δημιουργεί μια μεταβλητή uint16_t με όνομα index και αρχικοποιείται σε 0 όπου αυτή η μεταβλητή χρησιμοποιείται για να κρατάει το Step για της θέσεις των πινάκων ,valuesArray ,signalArray . |
| char pulse_name_from_serial_port = '0';: Δημιουργεί μια μεταβλητή χαρακτήρα pulse_name_from_serial_port και αρχικοποιείται στον χαρακτήρα '0' χρησιμοποιείται όταν ο χρήστης μεταβεί στο βήμα 1 και θέλει να ορίσει τι είδους σήμα θέλει να δημιουργήσει η DAC. |
| char type_of_wave[20] = "";: Δημιουργεί έναν πίνακας χαρακτήρων type_of_wave με μέγεθος 20 και αρχικοποιείται με ένα κενό string χρησιμοποιείται όταν ο χρήστης μεταβεί στο βήμα 1 και θέλει να ορίσει τι είδους σήμα θέλει να παραξει η DAC. |
| int stage = 0;: Δημιουργεί μια μεταβλητή int με όνομα stage και παίρνει την τιμή 0 η χρησιμότητα αυτής της μεταβλητής είναι για να ελέγχει σε πιο στάδιο βρίσκεται ώστε να τρέξει ο ανάλογος κώδικας με την σωστή σειρά |
| bool has_receive_order = false;: Δημιουργεί μια μεταβλητή boolean has_receive_order και αρχικοποιείται σε false χρησιμοποιείται για να ελέγχει εάν ο χρήστης έχει στείλει μέσο της UART κάποια οδηγία |
| bool has_receive_mode = false;: Δημιουργεί μια ακόμη μεταβλητή boolean has_receive_mode και αρχικοποιείται σε false χρησιμοποιείται για να ελέγχει εάν ο χρήστης έχει στείλει μέσο της UART κάποια οδηγία για το ποιο mode θέλει να χρησιμοποιήσει |
| int mode = -1;: Δημιουργεί μια μεταβλητή int με όνομα mode και αρχικοποιείται σε -1 χρησιμοποιείται για να ελέγχει πιο από τα τέσσερα modes θέλει να δουλέψει |

| |
|---|
| ο χρήστης |
| char period_string[9] = "";: Δημιουργεί έναν πίνακας χαρακτήρων period_string με μέγεθος 9 και αρχικοποιείται με ένα κενό string, χρησιμοποιείται στο mode ένα οπού ο χρήστης στης οδηγίες του δίνει ένα νούμερο το οποίο αυτό το νούμερο πρέπει να είναι η συχνότητα της περίοδου του Counter_timer(). |
| int period_integer = -1;: Δημιουργεί μια μεταβλητή int με όνομα period_integer και αρχικοποιείται σε -1, λόγο του ότι η UART στέλνει την πληροφορία με τύπου μεταβλητής String και χρειάζεται να είναι Integer χρειαζόμαστε αυτήν την μεταβλητή οπού θα αποθηκευτεί η περίοδο που ζήτησε ο χρήστης. |
| char myString[9] = "";: Δημιουργεί έναν ακόμη πίνακας χαρακτήρων myString με μέγεθος 9 και αρχικοποιείται με ένα κενό string , χρησιμοποιείται στο mode 1 οπού στον πίνακα αυτό αποθηκεύεται το String με της οδηγίες που στέλνει ο χρήστης σχετικά με το ποιο σήμα και σε τι περίοδο θέλει να εμφανιστεί. |
| char myMode[3] = "" ;: Δημιουργεί ακόμη ένας πίνακας χαρακτήρων myMode με μέγεθος 3 και αρχικοποιείται με ένα κενό string, χρησιμοποιείται για να αποθηκεύσει από την UART το μήνυμα του χρήστη σχετικά με το πιο mode θέλει να χρησιμοποιήσει. |
| ADC_StartConvert();: Ξεκινά τη διαδικασία μετατροπής αναλογικού σήματος σε ψηφιακό μέσω του Αναλογικού-Ψηφιακού Μετατροπέα (ADC). |
| int timesSkipedLoop = 0;: Δημιουργεί μια μεταβλητή int με όνομα timesSkipedLoop και αρχικοποιείται σε 0 χρησιμοποιείται στο mode τέσσερα οπού θέλουμε να δούμε εάν υπάρχουν αναγνώσεις από την ADC οπού διαβάσαμε τον ίδιο αριθμό 2 φορές. |
| int mode_3_fake_signal_variable = 0;: Δημιουργεί μια ακόμη μεταβλητή int με όνομα mode_3_fake_signal_variable και αρχικοποιείται σε 0 χρησιμοποιείται στο mode τρία για να προσομοιώσουμε της μεταβλητές που θα διάβαζε η ADC . |
| int mode_3_direction = 1;: Δημιουργεί μια μεταβλητή int με όνομα mode_3_direction και αρχικοποιείται σε 1. |
| int mode_3_step = 50;: Δημιουργεί μια ακόμη μεταβλητή int με όνομα |

| |
|---|
| mode_3_step και αρχικοποιείται σε 50. |
| CyDelay(100);: Προκαλεί έναν χρονικό καθυστέρησης 100 ms χρησιμοποιώντας την συνάρτηση CyDelay του PSoC. |
| for(;;): ατέλειωτος βρόγχος που θα εκτελείται συνεχώς. |
| if(!has_receive_mode): Έλεγχοςανημεταβλητήhas_receive_modeείναι false. Αυτό σημαίνει ότι δεν έχουν ληφθεί ακόμη πληροφορίες για τον τύπο λειτουργίας. |
| UART_GetArray(myMode , 2);: Διαβάζει έναν πίνακα χαρακτήρων myMode μήκους 2 από το UART. Αυτό αναμένεται να περιέχει πληροφορίες για τον τύπο λειτουργίας. |
| printf("myMode[0] = %c \n\r" , myMode[0]);: Εκτυπώνει τον πρώτο χαρακτήρα του myMode. |
| printf("myMode[1] = %c \n\r" , myMode[1]);: Εκτυπώνει τον δεύτερο χαρακτήρα του myMode. |
| if (myMode[0] == '1') { ... }: Έλεγχος αν ο πρώτος χαρακτήρας του myMode είναι '1'. Αν ναι, τότε ορίζει τον τύπο λειτουργίας (mode) σε 1 και θέτει τη μεταβλητή has_receive_mode σε true. |
| else if(myMode[0] == '2'){ ... }: Εάν ο πρώτος χαρακτήρας είναι '2', τότε ορίζει το mode σε 2, την stage σε 1 και το period_integer. Στη συνέχεια, θέτει την has_receive_mode σε true. |
| else if(myMode[0] == '3'){ ... }: Εάν ο πρώτος χαρακτήρας είναι '3', τότε ορίζει το mode σε 3 και την stage σε 1. Στη συνέχεια, θέτει την has_receive_mode σε true. |
| else if(myMode[0] == '4'){ ... }: Εάν ο πρώτος χαρακτήρας είναι '4', τότε ορίζει το mode σε 4, την stage σε 1 και ξεκινά μια διαδικασία που συνδέεται με την DMA (Direct Memory Access) και άλλες ρυθμίσεις. Στη συνέχεια, θέτει την has_receive_mode σε true. |
| CyDelay(100);: Καθυστερεί την εκτέλεση για 100 ms χρησιμοποιώντας τη συνάρτηση CyDelay. Αυτό μπορεί να είναι χρήσιμο για να δοθεί χρόνος στο σύστημα να εκτελέσει ορισμένες εργασίες πριν συνεχιστεί ο κύκλος του βρόγχου. |

| |
|---|
| Ο κώδικας περιμένει είσοδο από το UART για να καθορίσει τον τύπο λειτουργίας (mode) και να προχωρήσει στην επόμενη φάση (stage). Αυτός ο βρόγχος είναι κομβικό κομμάτι για τον έλεγχο της λειτουργίας του προγράμματος. |
| if(mode == 1) { ... }: Ελέγχει αν ο τύπος λειτουργίας είναι 1. Αν αυτό ισχύει, τότε εκτελούνται οι εντολές μέσα στο μπλοκ. |
| |
| if(!has_receive_order) { ... }: Ελέγχει αν η μεταβλητή has_receive_order είναι false, υποδηλώνοντας ότι δεν έχει ληφθεί ακόμη εντολή. |
| UART_GetArray(myString , 8);: Διαβάζει έναν πίνακα χαρακτήρων myString από το UART, περιμένοντας 8 χαρακτήρες. |
| Οι επόμενες 9 γραμμές (από το printf("myString[0] = %c \n\r" , myString[0]); έως printf("myString[8] = %c \n\r" , myString[8]);) χρησιμοποιούνται για εκτύπωση του περιεχομένου του πίνακα myString. Αυτό μπορεί να βοηθήσει στον έλεγχο ανάλυσης των ληφθέντων δεδομένων. |
| if (myString[0] != '\0') { ... }: Έλεγχος αν ο πρώτος χαρακτήρας του myString δεν είναι το κενό (null character). Αν αυτό ισχύει, τότε εκτελούνται οι εντολές μέσα στο μπλοκ. |
| pulse_name_from_serial_port = myString[0];: Ορίζειτημεταβλητήpulse_name_from_serial_portίσημετονπρώτοχαρακτήρατουmyString. |
| strcpy(period_string, myString + 2);: Κάνει αντιγραφή των χαρακτήρων από το δεύτερο χαρακτήρα του myString και μετά στον πίνακα period_string. |
| period_integer = atoi(period_string);: Μετατρέπει το period_string σε ακέραιο χρησιμοποιώντας τη συνάρτηση atoi και τον αποθηκεύει στη μεταβλητή period_integer. |
| printf("period int :%d \n\r", period_integer);: Εκτυπώνειτηντιμήτηςμεταβλητήςperiod_integer. |
| Counter_1_SetPeriod(period_integer);: Ορίζειτηνπερίοδοτουχρονομέτρου |

| |
|---|
| (counter) σeperiod_integer. |
| CyDelay(1000);: Καθυστερεί την εκτέλεση για 1000 ms (1 δευτερόλεπτο). stage = 1;:<br><br>Ορίζει τη μεταβλητή stage ίση με 1 has_receive_order = true;: Θέτει τη μεταβλητή has_receive_order σε true, υποδηλώνοντας ότι έχει ληφθεί μια εντολή.<br>Αυτό το τμήμα κώδικα διαχειρίζεται την επεξεργασία εντολών όταν ο τύπος λειτουργίας είναι 1. Αναμένει να λάβει μια εντολή μέσω του UART, εξάγει πληροφορίες όπως το όνομα του παλμού και την περίοδο, και εκτελεί συγκεκριμένες ενέργειες ανάλογα με τα δεδομένα αυτά. |
| Εφόσον έχει επιλεγεί ο τύπος κύματος, ελέγχει τον τύπο του κύματος και εκκινεί το αντίστοιχο DMA (Direct Memory Access) με τον πίνακα που περιέχει τα δεδομένα.<br>Αν ο τύπος κύματος είναι "pulse", τότε εκκινεί το DMA με τα δεδομένα από τον πίνακα pulseWave. |
| Αν ο τύπος κύματος είναι "square", τότε εκκινεί το DMA με τα δεδομένα από τον πίνακα squareWave. |
| Αν ο τύπος κύματος είναι "sin", τότε εκκινεί το DMA με τα δεδομένα από τον πίνακα sineWaveLUT. |
| Αν ο τύπος κύματος είναι "triangle", τότε εκκινεί το DMA με τα δεδομένα από τον πίνακα triangleWave. |
| stage = 2;: Ορίζει τη μεταβλητή stage ίση με 2, υποδηλώνοντας ότι το πρόγραμμα έχει μεταβεί στο δεύτερο στάδιο. |
| if (stage == 2) {: Ελέγχει εάν η μεταβλητή stage έχει την τιμή 2. |
| Cy_GPIO_Inv(P9_4_PORT, P9_4_PIN);: Αντιστρέφει την κατάσταση του GPIO pin που είναι συνδεδεμένος στο Pin_10_5 (P9_4_PORT, P9_4_PIN). Αυτή η ενέργεια μπορεί να χρησιμοποιηθεί για τον έλεγχο της κατάστασης ενός pin. |
| result = Cy_SAR_GetResult16(SAR, chan);: Λαμβάνει την τρέχουσα τιμή από τον SAR ADC (Analog-to-Digital Converter) για τον κανάλι που ορίζεται από τη |

| |
|---|
| μεταβλητή chan και αποθηκεύει το αποτέλεσμα στη μεταβλητή result. |
| resultmV = Cy_SAR_CountsTo_mVolts(SAR, chan, result);: Μετατρέπει την τιμή που λήφθηκε από τον SAR ADC σε mV (millivolts) χρησιμοποιώντας τη συνάρτηση Cy_SAR_CountsTo_mVolts. |
| signalArray[index] = resultmV; valuesArray[index] = resultmV;: Αποθηκεύει τη μετατραπείσα τιμή στους πίνακες signalArray και valuesArray στον δείκτη index. |
| index = (index + 1) % ARRAY_SIZE;: Αυξάνει τον δείκτη index κατά ένα και εφαρμόζει τον τελεστή % (modulo) για να διατηρήσει τον δείκτη στο εύρος [0, ARRAY_SIZE-1]. |
| printf("Loop counter :%d \r\n", index);: Εκτυπώνει την τρέχουσα τιμή του δείκτη index. |
| if (index % ARRAY_SIZE == 0) { printf("you can see the FFT now.\n\r"); stage = 3; }: Εάν ο δείκτης index είναι μηδέν μετά την αύξησή του, τότε εκτυπώνει ένα μήνυμα και ορίζει τη μεταβλητή stage σε 3. |
| Συνολικά, αυτό το τμήμα κώδικα εκτελεί την ανάγνωση μιας τιμής από έναν ADC, τη μετατροπή της σε mV, και την αποθήκευση της σε έναν πίνακα, ενώ παράλληλα παρακολουθεί την τιμή του δείκτη index και αλλάζει την κατάσταση stage όταν ο δείκτης φτάσει σε μια συγκεκριμένη τιμή. |
| if (stage == 3) { cooley_tukey_fft(valuesArray, ARRAY_SIZE); stage = 4; }: Εάν η μεταβλητή stage έχει την τιμή 3, τότε καλεί τη συνάρτηση cooley_tukey_fft για να υπολογίσει τη διακριτή μετασχηματισμένη Fourier (FFT) του πίνακα valuesArray με μέγεθος ARRAY_SIZE. Έπειτα, η μεταβλητή stage αλλάζει σε 4. |
| if (stage == 4) { for (int i = 0; i < ARRAY_SIZE; i++) { printf("Signal[%d] = %d \r\n", i, signalArray[i]); } stage = 5; }: Εάν η μεταβλητή stage έχει την τιμή 4, τότε εκτυπώνει τις τιμές του πίνακα signalArray (ο οποίος φαίνεται να περιέχει τις τιμές του αρχικού σήματος) και στη συνέχεια η μεταβλητή stage αλλάζει σε 5. |
| if (stage == 5) { for (int i = 0; i < ARRAY_SIZE; i++) { printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), cimag(valuesArray[i])); } stage = 6; }: Εάν η μεταβλητή stage έχει την τιμή 5, τότε εκτυπώνει τις τιμές του πίνακα valuesArray |

| |
|---|
| με τη μορφή πραγματικού και φανταστικού μέρους (πραγματικό + φανταστικό i) και στη συνέχεια η μεταβλητή stage αλλάζει σε 6. |
| if (stage == 6) { UART_PutString("Ended.\n\r"); stage = 7; }: Εάν η μεταβλητή stage έχει την τιμή 6, τότε εκτυπώνει το μήνυμα "Ended" μέσω του UART και η μεταβλητή stage αλλάζει σε 7. |
| Συνολικά, αυτό το τμήμα του κώδικα εκτελεί την FFT, εκτυπώνει τις τιμές του αρχικού σήματος (signalArray) και τις μετασχηματισμένες τιμές (FFT) του σήματος (valuesArray), και εκτυπώνει ένα μήνυμα κατάληξης ("Ended"). Η μεταβλητή stage χρησιμοποιείται για τον έλεγχο της εκτέλεσης των διαφόρων σταδίων του προγράμματος. |
| if (stage == 7) { UART_PutString("Finished.\n\r"); ... }: Εάν η μεταβλητή stage έχει την τιμή 7, τότε εκτυπώνει το μήνυμα "Finished" μέσω του UART. |
| memset(valuesArray, 0, sizeof(valuesArray));: Κάνει μηδενισμό όλων των στοιχείων του πίνακα valuesArray. |
| memset(signalArray, 0, sizeof(signalArray));: Κάνει μηδενισμό όλων των στοιχείων του πίνακα signalArray. |
| index = 0;: Θέτει τη μεταβλητή index σε 0. |
| pulse_name_from_serial_port = '0';: Θέτειτημεταβλητήpulse_name_from_serial_portσετιμή '0'. |
| memset(type_of_wave, 0, sizeof(type_of_wave));: Κάνειμηδενισμόόλωντωνστοιχείωντουπίνακαtype_of_wave. |
| stage = 0;: Θέτει τη μεταβλητή stage σε 0. |
| has_receive_order = false;: Θέτειτημεταβλητήhas_receive_orderσε false. |
| memset(period_string, 0, sizeof(period_string));: Κάνειμηδενισμόόλωντωνστοιχείωντουπίνακαperiod_string. |
| strcpy(period_string, "");: Αντιγράφειένακενό string στονπίνακαperiod_string. |
| period_integer = -1;: Θέτειτημεταβλητήperiod_integerσε -1. |

| |
|---|
| memset(myString, 0, sizeof(myString));: ΚάνειμηδενισμόόλωντωνστοιχείωντουπίνακαmyString. |
| myString[0] = '\0'; ... myString[8] = '\0';: Θέτει κάθε στοιχείο του πίνακα myString σε '\0'. |
| memset(myMode, 0, sizeof(myMode));: Κάνει μηδενισμό όλων των στοιχείων του πίνακα myMode. |
| myMode[0] = '\0'; myMode[1] = '\0';: Θέτει τα πρώτα δύο στοιχεία του πίνακα myMode σε '\0'. |
| DMA_ADC_Stop();: Σταματά τη λειτουργία του DMA που χρησιμοποιείται για τον μετασχηματισμό του σήματος. |
| CyDelay(500);: Προκαλεί έναν χρονικό καθυστέρηση για 500 ms. |
| Η συνθήκη if (mode == 2) ελέγχει αν η μεταβλητή mode έχει την τιμή 2. Εάν η συνθήκη αυτή είναι αληθής, τότε ο κώδικας που βρίσκεται μέσα στην επόμενη παράγραφο θα εκτελεστεί. |
| if(stage == 1) { DMA_ADC_Start(sineWaveLUT, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT)); stage = 2; CyDelay(100); }: Εάν η μεταβλητή stage είναι 1, τότε ξεκινά την DMA. Στη συνέχεια, η μεταβλητή stage ορίζεται σε 2 και υπάρχει μια καθυστέρηση 100ms. |
| if(stage == 2) { Counter_1_SetPeriod(period_integer); result = Cy_SAR_GetResult16(SAR, chan); ... }: Αν η μεταβλητή stage είναι 2, τότε ορίζεται η περίοδος του Timer/Counter 1 στην τιμή period_integer. Στη συνέχεια, διαβάζεται η τιμή της τάσης από το SAR (Αναλογικός-Ψηφιακός Μετατροπέας) και αποθηκεύεται στα arrays signalArray και valuesArray. Η μεταβλητή index αυξάνεται και εκτυπώνεται ο αριθμός επανάληψης (loop counter). Εάν έχει συμπληρωθεί ο πίνακας valuesArray, η μεταβλητή stage ορίζεται σε 3. |
| if(stage == 3) { cooley_tukey_fft(valuesArray, ARRAY_SIZE); stage = 4; }: Αν η μεταβλητή stage είναι 3, τότε εκτελείται ο αλγόριθμος FFT στον πίνακα valuesArray και η μεταβλητή stage ορίζεται σε 4. |
| if(stage == 4) { for (int i = 0; i < ARRAY_SIZE; i++) { printf("Signal[%d] = %d |

| |
|---|
| \r\n", i, signalArray[i]); } stage = 5; }: Αν η μεταβλητή stage είναι 4, τότε εκτυπώνονται τα σήματα από τον πίνακα signalArray και η μεταβλητή stage ορίζεται σε 5. |
| if(stage == 5) { for (int i = 0; i < ARRAY_SIZE; i++) { printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), cimag(valuesArray[i])); } stage = 6; }: Αν η μεταβλητή stage είναι 5, τότε εκτυπώνονται τα αποτελέσματα του FFT από τον πίνακα valuesArray και η μεταβλητή stage ορίζεται σε 6. |
| if(stage == 6) { UART_PutString("Ended.\n\r"); stage = 7; }: Αν η μεταβλητή stage είναι 6, τότε εκτυπώνεται το μήνυμα "Ended" μέσω UART και η μεταβλητή stage ορίζεται σε 7. |
| if(stage == 7) { UART_PutString("Finished.\n\r"); ... }: Αν η μεταβλητή stage είναι 7, τότε εκτυπώνεται το μήνυμα "Finished" μέσω UART και επαναφέρονται οι μεταβλητές και οι πίνακες σε αρχικές τιμές. Το stage ορίζεται σε 2 και το period_integer αυξάνεται κατά κάποιον τρόπο, ανάλογα με την τρέχουσα τιμή της. |
| if (mode == 3): Ελέγχει αν η μεταβλητή mode έχει την τιμή 3. Αν αυτή η συνθήκη είναι αληθής, τότε ο ενταχθείς κώδικας θα εκτελεστεί. |
| if (stage == 1): Αν βρισκόμαστε στο στάδιο 1 της λειτουργίας με την τιμή 3 για το mode, τότε εκτελούμε τα παρακάτω βήματα. |
| b. Έλεγχοι για τα όρια της mode_3_fake_signal_variable: |
| Αν η mode_3_fake_signal_variable είναι μεγαλύτερη ή ίση με 3300, τότε θέτει την mode_3_direction σε -1 και την mode_3_fake_signal_variable στη μέγιστη τιμή 3300. |
| Αν η mode_3_fake_signal_variable είναι μικρότερη από 50, τότε θέτει την mode_3_direction σε 1 και την mode_3_fake_signal_variable στην ελάχιστη τιμή 0. |
| c. Εκχώρηση τιμής στις μεταβλητές αποτελέσματος resultmV, signalArray και valuesArray: |
| Η τιμή της resultmV γίνεται ίση με την τρέχουσα τιμή της mode_3_fake_signal_variable. |

| |
|---|
| Οι τιμές προστίθενται στα arrays signalArray και valuesArray. |
| Ο δείκτης index αυξάνεται κατά 1 και υπολογίζεται το ARRAY_SIZE. |
| d. Έλεγχος για ολοκλήρωση του σταδίου: |
| Επανάληψη για τα υπόλοιπα στάδια: |
| Στάδιο 2: Εκτέλεση του FFT. |
| Στάδιο 3: Εκτύπωση των τιμών του σήματος. |
| Στάδιο 4: Εκτύπωση των τιμών του FFT. |
| Στάδιο 5: Εκτύπωση "Ended". |
| Στάδιο 6: Εκτύπωση "Finished" και επαναφορά των μεταβλητών σε αρχικές τιμές. |
| if (stage == 6): Αφού ολοκληρωθεί η διαδικασία, ορίζεται το στάδιο σε 1, η μεταβλητή mode_3_step διπλασιάζεται και χρησιμοποιείται για τον υπολογισμό της ποσότητας που προστίθεται στο επόμενο κύκλο επανάληψης. |
| if (mode == 4): Ελέγχει αν η μεταβλητή mode έχει την τιμή 4. Αν αυτή η συνθήκη είναι αληθής, τότε ο ενταχθείς κώδικας θα εκτελεστεί. |
| if (stage == 1): Αν βρίσκεται στο στάδιο 1 της λειτουργίας με την τιμή 4 για το mode, τότε εκτελεί τα παρακάτω βήματα. |
| a. Εναλλαγή της κατάστασης του pin P9_4: |
| Η συνάρτηση Cy_GPIO_Inv(P9_4_PORT, P9_4_PIN) αντιστρέφει την τρέχουσα κατάσταση του pin P9_4. |
| b. Απόκτηση της τρέχουσας τιμής του SAR ADC και μετατροπή των αποτελεσμάτων σε mV: |
| Με τη χρήση των συναρτήσεων Cy_SAR_GetResult16 και Cy_SAR_CountsTo_mVolts, παίρνει την τιμή που μετράει το SAR ADC στο κανάλι chan και τη μετατρέπει σε mV. |
| c. Έλεγχος για την ολοκλήρωση του σταδίου: |
| Αν η τρέχουσα τιμή διαφέρει από την προηγούμενη τιμή στο signalArray, τότε η |

| |
|---|
| τρέχουσα τιμή προστίθεται στα arrays signalArray και valuesArray, ο δείκτης index αυξάνεται και υπολογίζεται το modulo με το ARRAY_SIZE. |
| Αν η τρέχουσα τιμή είναι ίδια με την προηγούμενη, τότε αυξάνεται ο μετρητής timesSkipedLoop. |
| d. Έλεγχος για την ολοκλήρωση του σταδίου: |
| Επανάληψη για τα υπόλοιπα στάδια: |
| Στάδιο 2: Εκτέλεση του FFT. |
| Στάδιο 3: Εκτύπωση των τιμών του σήματος. |
| Στάδιο 4: Εκτύπωση των τιμών του FFT και του αριθμού επαναλήψεων που παραλείφθηκαν (timesSkipedLoop). |
| Στάδιο 5: Εκτύπωση "Ended". |
| Στάδιο 6: Εκτύπωση "Finished" και επαναφορά των μεταβλητών σε αρχικές τιμές. |
| if (stage == 6): Αφού ολοκληρωθεί η διαδικασία, ορίζεται το στάδιο σε 1, ο μετρητής timesSkipedLoop μηδενίζεται και επαναφέρονται οι μεταβλητές σε αρχικές τιμές. |

Η συνάρτηση cooley_tukey_fft ξεκινάει με έναν έλεγχο για το αν το μήκος της ακολουθίας n είναι μικρότερο ή ίσο με 1. Σε αυτήν την περίπτωση, δεν χρειάζεται να γίνει κανένας υπολογισμός, και η συνάρτηση επιστρέφει.

Δημιουργούνται δύο πίνακες even και odd για να διαχωρίσουν τα ζευγάρια και τις μονές θέσεις αντίστοιχα.

Ακολουθούν αναδρομικές κλήσεις της cooley_tukey_fft για τους πίνακες even και odd.

Στο τμήμα συγχώνευσης των αποτελεσμάτων, υπολογίζεται για κάθε k η τιμή t βάσει του τύπου του ριζών της μονάδας, και στη συνέχεια ενώνονται τα αποτελέσματα των even και odd με βάση αυτήν την τιμή.

Οι πίνακες even και odd ελευθερώνονται από τη μνήμη.

Το αποτέλεσμα είναι η αναδρομική εφαρμογή του αλγορίθμου Cooley-Tukey για τον υπολογισμό του FFT στην ακολουθία εισόδου x.

Ο κώδικας υλοποιεί τον αλγόριθμο Cooley-Tukey για τον γρήγορο μετασχηματισμό Fourier (FFT), ο οποίος χρησιμοποιείται για τον υπολογισμό της μετασχηματισμένης Fourier μιας ακολουθίας σημάτων.

Κατά την ανάλυση του κώδικα στο PSoC Creator, εξερευνήσαμε μια σειρά από λειτουργίες και δομές που χρησιμοποιήθηκαν σε ένα πρόγραμμα εφαρμογής για τον υπολογισμό του FFT (Fast Fourier Transform). Ο κώδικας περιλάμβανε τη χρήση ποικίλων περιφερειακών κυκλωμάτων, όπως ADC (Analog-to-Digital Converter), DMA (Direct Memory Access), και τον χειρισμό διαφόρων λειτουργιών, όπως η επικοινωνία μέσω UART, η αρχικοποίηση εξαρτήσεων οπού ορίστικε ποιο είδος σήματος θα στέλνει η DMA και η διαχείριση λειτουργικών καταστάσεων οπού μπορεί να επιλέξει εάν θέλει να χρησιμοποιήσει το mode 4 (Live analyzer) η mode 1 (οπού ελέγχεται η συχνότητα της κυματομορφής και το είδος αυτής) είτε να επιλέξει το mode 2 ή 3 οπού μας παρουσιάζονται δυο έτοιμες προσομοιώσεις.

Το πρόγραμμα υποστηρίζει διάφορα modes of operation, όπως οι λειτουργίες για τον υπολογισμό του FFT για σήματα παλμικής, τετραγωνικής, ημιτονοειδούς κυματομορφής και τριγωνικής. Η αλληλεπίδραση με το πρόγραμμα γίνεται μέσω της σειριακής θύρας UART, όπου μπορούμε να εισάγουμε λειτουργικές εντολές και παραμέτρους.

Επιπλέον, προσδιορίσαμε τον τρόπο λειτουργίας του προγράμματος στα διάφορα modes, εξηγώντας την ακριβή ακολουθία ενεργειών που λαμβάνονται για κάθε mode, συμπεριλαμβανομένης της αρχικοποίησης, της λήψης εντολών μέσω UART, του υπολογισμού του FFT, και της αποστολής δεδομένων πίσω μέσω UART.

Κατά τη διαδικασία ανάλυσης, επισημάναμε τη χρήση ποικίλων βιβλιοθηκών και συναρτήσεων, καθώς και τη σημασία των αρχείων ρυθμίσεων του PSoC Creator για τη σωστή ρύθμιση του περιβάλλοντος ανάπτυξης. Με αυτόν τον τρόπο, κατανοήσαμε τον τρόπο λειτουργίας του προγράμματος και τον τρόπο που εκμεταλλεύεται τις δυνατότητες του PSoC για τον υπολογισμό του FFT σε πραγματικό χρόνο.

## 6.3 Ανάλυση του προγράμματος python

Το πρόγραμμα σε Python ελέγχει και αναλύει δεδομένα που προέρχονται από το PSoC6 με τη χρήση της σειριακής θύρας UART. Η εφαρμογή παρέχει διάφορες λειτουργίες, όπως ο υπολογισμός FFT για διάφορα σήματα, εμφάνιση δεδομένων σε γραφική μορφή με χρήση του matplotlib, και αλληλεπίδραση με τον χρήστη για την επιλογή της λειτουργίας.

Το πρόγραμμα Python χρησιμοποιεί το serial module για την επικοινωνία με το PSoC6, το matplotlib για τη δημιουργία γραφημάτων, και το numpy για τις αναλυτικές διαδικασίες, όπως ο υπολογισμός FFT.

Το πρόγραμμα αναλύει τα δεδομένα που λαμβάνει από το PSoC6 και ενημερώνει τα γραφήματα σε πραγματικό χρόνο. Οι λειτουργίες που παρέχονται περιλαμβάνουν τον υπολογισμό του FFT, την εμφάνιση του σήματος και των αποτελεσμάτων FFT, καθώς και την αλληλεπίδραση με τον χρήστη για την επιλογή διαφορετικών κυμάτων.

Σημείωση: Αν ο διερμηνευτής της Python αναφέρει σφάλματα κατά την εκτέλεση, βεβαιωθείτε ότι οι απαραίτητες βιβλιοθήκες (serial, matplotlib, numpy) είναι εγκατεστημένες στο περιβάλλον εκτέλεσης της Python.

Οι απαραίτητες βιβλιοθήκες εισάγονται, συμπεριλαμβανομένων των serial, matplotlib, numpy, time, threading και Queue.

Δημιουργείται μια λίστα με τα διαθέσιμα COM ports.

Δημιουργείται ένα αντικείμενο Serial με την ονομασία serialInst και ρυθμίζονται τα παράμετροι της σειριακής θύρας (όπως το baud rate, port name) και εν συνεχεία ανοίγεται η σύνδεση.

Εμφανίζεται ένα μήνυμα για τη λήψη σήματος από το Psoc6.

Ορίζονται μεταβλητές για την αρχικοποίηση δεδομένων, όπως οι μεταβλητές timesSkipedLoop, max_data_points, sampling_rate, frequencies, time_values, signalArray, realArray, imaginaryArray, fftArray, numbersArray, readingSignnal, fft_magnitudes, modeOfAnalyzer, fft_magnitudes_queue, και plotReadyToGetUpdated.

Η συνάρτηση update χρησιμοποιείται ως callback για την ενημέρωση γραφήματος σε πραγματικό χρόνο. Ας αναλύσουμε τη συνάρτηση γραμμή προς γραμμή:

Η συνάρτηση δηλώνεται με το όνομα update και παίρνει ένα όρισμα frame. Το όρισμα αυτό χρησιμοποιείται για την ενημέρωση του γραφήματος σε περιπτώσεις που απαιτούν διαφορετικά frames (π.χ., σε animations).

Χρησιμοποιεί τις μεταβλητές fft_magnitudes_queue, plotReadyToGetUpdated, frequencies, signalArray και timesSkipedLoop, οι οποίες έχουν προανακαθοριστεί ως global μεταβλητές.

Ελέγχει αν η μεταβλητή plotReadyToGetUpdated είναι True, προκειμένου να αποφασίσει αν πρέπει να ενημερώσει το γράφημα.

Καθαρίζει το τρέχον γράφημα με την εντολή plt.clf().

Ανακτάένανέο FFT (Fast Fourier Transform) απότηνουράfft_magnitudes_queue.

Εκτυπώνει τη λίστα signalArray.

Πραγματοποιεί το plotting του νέου FFT, χρησιμοποιώντας τις συχνότητες και τις νεές τιμές του FFT.

Καθαρίζει τις λίστες signalArray, fftArray, και fft_magnitudes.

Ορίζει τη μεταβλητή plotReadyToGetUpdated ως False.

Ορίζει τη μεταβλητή timesSkipedLoop ως 0.

Ο κώδικας gotoStepTwo είναι μια συνάρτηση

Καλεί τη μέθοδο plt.plot(numbersArray, signalArray) για να σχεδιάσει ένα γράφημα με τις τιμές του signalArray σε σχέση με τις τιμές του numbersArray.

Εκτυπώνει τη λίστα signalArray στο τερματικό.

Προσθέτει ετικέτες στους άξονες (Number για τον άξονα x και Volts για τον άξονα y).

Ορίζει έναν τίτλο για το γράφημα με τη μέθοδο plt.title('Received Signal').

Ενημερώνει το γράφημα με τη μέθοδο plt.draw().

Περιμένει 5 δευτερόλεπτα χρησιμοποιώντας τη μέθοδο plt.pause(5).

Κλείνει το τρέχον γράφημα με τη μέθοδο plt.close().

Καλεί τη συνάρτηση gotoStepThree().

Συνολικά, η συνάρτηση gotoStepTwo() δημιουργεί ένα γράφημα με τις τιμές του σήματος που περιέχονται στο signalArray, το εμφανίζει για 5 δευτερόλεπτα, και στη συνέχεια κλείνει το γράφημα και καλεί τη συνάρτηση gotoStepThree().

gotoStepThree:
Δημιουργεί ένα γράφημα (plt.figure(figsize=(10, 6))) με δεδομένα που βρίσκονται στη λίστα fft_magnitudes.

Χρησιμοποιεί τη μέθοδο plt.plot(fft_magnitudes) για να σχεδιάσει το γράφημα με τις τιμές του fft_magnitudes.

Εκτυπώνει τη λίστα fft_magnitudes στο τερματικό.

Ορίζει τίτλο για το γράφημα (plt.title("Magnitude of FFT Output")).

Ορίζει ετικέτες για τους άξονες x και y.

Ενεργοποιεί το grid (plt.grid(True)).

Εμφανίζει το γράφημα (plt.show()).

Καθυστερεί για 0.1 δευτερόλεπτα (plt.pause(0.1)).

Απενεργοποιεί την αλληλεπίδραση με τον χρήστη (plt.ioff()).

Εναλλακτικά, κλείνει το γράφημα (plt.close()).

Καλεί τη συνάρτηση gotoStepFour().

gotoStepFour:

Ορίζει τη μεταβλητή readingSignnal σε 1.

Καθαρίζειτιςλίστεςtime_values, signalArray, fftArray, καιfft_magnitudes.

Διαβάζει από τον χρήστη το όνομα του κύματος (nameOfWave) μέσω της συνάρτησης input.

Στέλνει το nameOfWave μέσω της σειράς εντολών προς το Psoc6 με τη μέθοδο serialInst.write(nameOfWave.encode("utf-8")).

Ο κώδικας chooseType() περιέχει μια συνάρτηση που επιτρέπει στον χρήστη να εισάγει μια λειτουργία. Ας αναλύσουμε τον κώδικα γραμμή προς γραμμή:

Ζητά από τον χρήστη να εισάγει έναν αριθμό (mode) από τους αριθμούς 1, 2, 3, 4, αντιστοιχώντας τους σε συγκεκριμένες λειτουργίες.

Μετατρέπει τον εισαγμένο αριθμό σε ακέραιο με τη χρήση της int(mode).

Ορίζει τη μεταβλητή modeOfAnalyzer ως μια καθολική μεταβλητή για να μπορεί να χρησιμοποιηθεί σε όλο το πρόγραμμα.

Κάνει έλεγχο χρησιμοποιώντας το if-elif-else για να εκτελέσει διάφορες ενέργειες ανάλογα με την τιμή του mode.

Αν mode είναι 1, τότε ορίζει τη modeOfAnalyzer σε 1 και στέλνει το "1" στον Psoc6 μέσω της σειράς εντολών.

Αν mode είναι 2, τότε παρόμοια ορίζει τη modeOfAnalyzer σε 2 και στέλνει το "2".

Αν mode είναι 3, τότε παρόμοια ορίζει τη modeOfAnalyzer σε 3 και στέλνει το "3".

Αν mode είναι 4, τότε παρόμοια ορίζει τη modeOfAnalyzer σε 4 και στέλνει το "4".

Σε άλλη περίπτωση (αν ο αριθμός δεν είναι από 1 έως 4), τότε ορίζει τη modeOfAnalyzer σε -2.

Αυτός ο κώδικας επιτρέπει στον χρήστη να επιλέξει μια λειτουργία και καθορίζει τη μεταβλητή modeOfAnalyzer ανάλογα.

Ο κώδικας read_loop() είναι μια συνάρτηση που διαβάζει συνεχώς από τη σειριακή θύρα και επεξεργάζεται τα δεδομένα που λαμβάνει. Ας αναλύσουμε τον κώδικα

Έναρξη ενός ατέρμονα βρόχου (while True).

Έλεγχος εάν υπάρχουν δεδομένα προς ανάγνωση στη σειριακή θύρα (if serialInst.in_waiting).

Εάν υπάρχουν δεδομένα, διαβάζει μια γραμμή από τη σειριακή θύρα (packet = serialInst.readline()).

Κωδικοποίηση της γραμμής σε Unicode χρησιμοποιώντας UTF-8 και κατάργηση χαρακτήρων νέας γραμμής και tab (response = packet.decode('utf-8').strip('\r\n\t ')).

Έλεγχος του περιεχομένου της απάντησης για συγκεκριμένα πρότυπα:

Αν η απάντηση ξεκινά με τη λέξη "Signal", τότε διαχωρίζει την απάντηση για να εξαχθεί η τιμή του σήματος και την προσθέτει στον πίνακα signalArray.

Αν η απάντηση ξεκινά με τη λέξη "FFT", τότε προσθέτει την απάντηση στον πίνακα fftArray.

Αν η απάντηση ξεκινά με τη λέξη "LOOPSKIPED", τότε διαχωρίζει την απάντηση για να εξαχθεί η τιμή του αριθμού που δηλώνει τον αριθμό των επαναλήψεων που παραλείπονται και τον αποθηκεύει στη μεταβλητή timesSkipedLoop.

Αν η απάντηση ξεκινά με τη λέξη "Ended", τότε επεξεργάζεται τα δεδομένα FFT από τον πίνακα fftArray, μετατρέπει τα δεδομένα σε αριθμητικά, και τις τοποθετεί στην ουρά fft_magnitudes_queue για ενημέρωση του γραφήματος.

Αυτό το τμήμα κώδικα ελέγχει την τιμή της μεταβλητής modeOfAnalyzer και εκτελεί διαφορετική λειτουργία ανάλογα με την τιμή της. Ας το αναλύσουμε γραμμή προς γραμμή:

Αν η τιμή της μεταβλητής modeOfAnalyzer είναι 1, τότε το πρόγραμμα περιμένει τον χρήστη να εισάγει το όνομα μιας κυματομορφής (παλμός, τετράγωνο, sine, τρίγωνο) και την στέλνει στον μικροελεγκτή μέσω της σειριακής θύρας. Έπειτα, διαβάζει συνεχώς δεδομένα από τη σειριακή θύρα και ενημερώνει τα γραφήματα.

Αν η τιμή της μεταβλητής modeOfAnalyzer είναι 2, 3 ή 4, τότε ξεκινά ένα νέο νήμα εκτέλεσης (thread) που εκτελεί τη συνάρτηση read_loop. Το thread αυτό αναλαμβάνει να διαβάζει συνεχώς δεδομένα από τη σειριακή θύρα. Συγχρόνως, δημιουργεί ένα καινούργιο παράθυρο Matplotlib, ρυθμίζει μια συνάρτηση (update) για ανανέωση του γραφήματος, και ξεκινά μια animation (FuncAnimation) για ανανέωση του γραφήματος σε τακτά χρονικά διαστήματα.

Το else του τελευταίου block εκτελείται εάν η τιμή της modeOfAnalyzer δεν είναι ούτε 1, ούτε 2, ούτε 3, ούτε 4, εμφανίζοντας ένα μήνυμα σφάλματος.

Ο σκοπός αυτού του τμήματος κώδικα είναι να διαχειριστεί τον έλεγχο ροής του προγράμματος ανάλογα με τον τρόπο λειτουργίας που επιλέγει ο χρήστης.

Αυτό το τμήμα κώδικα ελέγχει την τιμή της μεταβλητής modeOfAnalyzer και εκτελεί διαφορετική λειτουργία ανάλογα με την τιμή της.

Αν η τιμή της μεταβλητής modeOfAnalyzer είναι 1, τότε το πρόγραμμα περιμένει τον χρήστη να εισάγει το όνομα μιας κυματομορφής (παλμός, τετράγωνο, ημίτονο, τρίγωνο) και την στέλνει στον μικροελεγκτή μέσω της σειριακής θύρας. Έπειτα, διαβάζει συνεχώς δεδομένα από τη σειριακή θύρα και ενημερώνει τα γραφήματα.

Αν η τιμή της μεταβλητής modeOfAnalyzer είναι 2, 3 ή 4, τότε ξεκινά ένα νέο νήμα εκτέλεσης (thread) που εκτελεί τη συνάρτηση read_loop. Το thread αυτό αναλαμβάνει να διαβάζει συνεχώς δεδομένα από τη σειριακή θύρα. Συγχρόνως, δημιουργεί ένα καινούργιο παράθυρο Matplotlib, ρυθμίζει μια συνάρτηση (update) για ανανέωση του γραφήματος, και ξεκινά μια animation (FuncAnimation) για ανανέωση του γραφήματος σε τακτά χρονικά διαστήματα.

Το else του τελευταίου block εκτελείται εάν η τιμή της modeOfAnalyzer δεν είναι ούτε 1, ούτε 2, ούτε 3, ούτε 4, εμφανίζοντας ένα μήνυμα σφάλματος.

Ο σκοπός αυτού του τμήματος κώδικα είναι να διαχειριστεί τον έλεγχο ροής του προγράμματος ανάλογα με τον τρόπο λειτουργίας που επιλέγει ο χρήστης.

Συνοψίζοντας παρουσιάσαμε μια πλήρη ανάλυση της αλυσίδας εργασιών, από την ανάπτυξη του κώδικα στο PSoC Creator μέχρι την απεικόνιση και ανάλυση των δεδομένων στο πρόγραμμα Python. Ο κώδικας του PSoC προγραμματίστηκε με βάση τις λειτουργικές απαιτήσεις, ενώ το πρόγραμμα Python επιτέλεσε τη συλλογή και την αναπαράσταση των δεδομένων με χρήση γραφημάτων και αναλύσεων FFT.

## 6.4 Η πλακέτα

1) Ο μικροελεγκτής CYBLE_PSoC6_416045

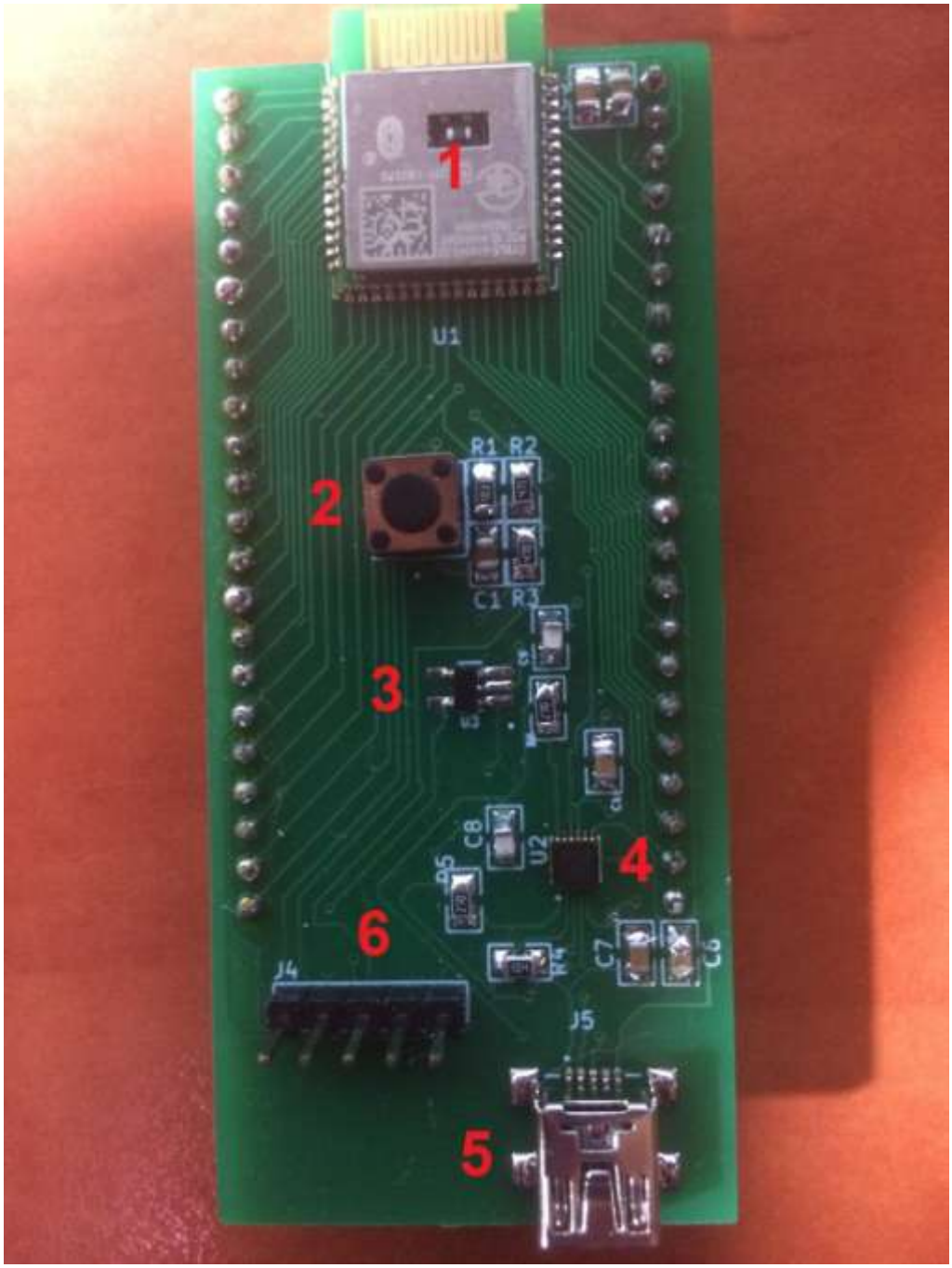
2) Ο διακόπτης τύπου push-button

3) Ο  Σταθεροποιητής τάσης BU33TD3WG

4) ΟΜετατροπέας USB-to-serial FT234XD

5) Η θύρα USB

6) Οι ακροδέκτες  που θα  συνδεθεί  το miniprog4 για τον προγραμματισμό του PSoC6



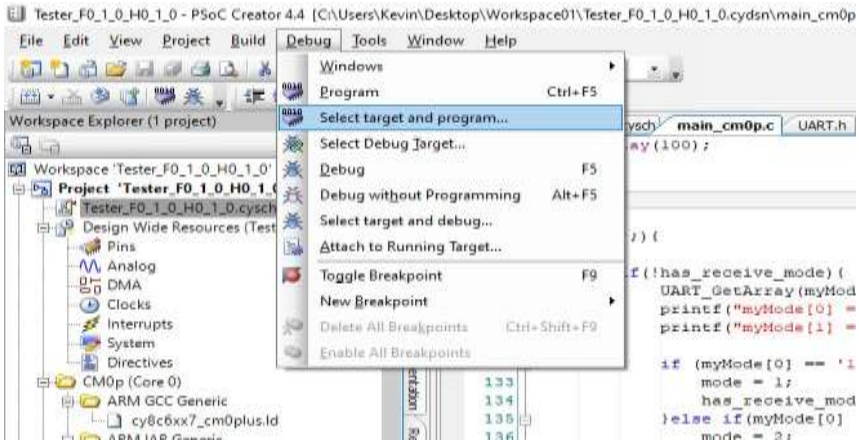

Εικ. 14 Η πλακέτα

# Κεφαλαιο 7 - TheRun

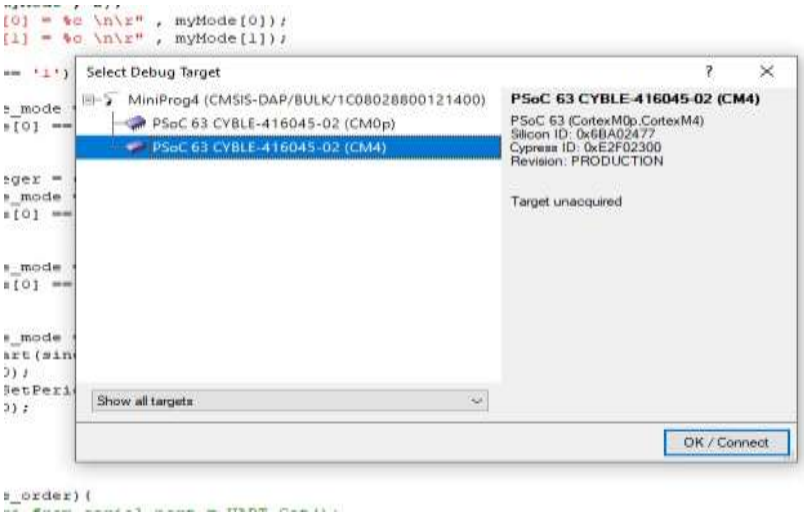
## 7.1 Setup

Για τη λειτουργία της εφαρμογής απαιτείται πρώτα η εκτέλεση του προγράμματος του PSoC6 και στη συνέχεια το πρόγραμμα στην Python (Εικ. 15 , Εικ. 16).

Μετά την εκκίνηση του προγράμματος Python, ορίζεται το επιθυμητό σήμα για ανάλυση (παλμός, τετράγωνο, ημίτονο, τρίγωνο). Στη συνέχεια, εμφανίζονται τα γραφήματα και οι αναλύσεις καθώς αυτά αποτελούν την αντανάκλαση της λειτουργίας του ενσωματωμένου συστήματος.




Εικ. 15 Εκτέλεση κώδικα Psoccreator




Εικ. 16 ΕκτέλεσηκώδικαPsoc creator

**Psoc Creator**

Άνοιγμα του PSoCCreator:

Άνοιγμα του Έργου:

Ανοίξτε το έργο που περιλαμβάνει το πρόγραμμά σας. Αυτό μπορεί να γίνει είτε επιλέγοντας το έργο από το "RecentProjects" είτε χρησιμοποιώντας την επιλογή "OpenProject..." και περιηγηθείτε στον κατάλογο του έργου.

Σύνδεση του Συστήματος:

Συνδέστε τον προγραμματιζόμενο ελεγκτή PSoC στον υπολογιστή σας μέσω USB.

Φόρτωση του Προγράμματος στον Ελεγκτή:

Στο PSoCCreator, μπορείτε να χρησιμοποιήσετε την επιλογή "Program" για να φορτώσετε το πρόγραμμα στον PSoC. Το κουμπί αυτό συνήθως βρίσκεται δίπλα στο κουμπί "Build".

Εκτέλεση του Προγράμματος:

Μετά τη φόρτωση, μπορείτε να εκτελέσετε το πρόγραμμα πατώντας το αντίστοιχο κουμπί (Run). Αυτό θα εκτελέσει το πρόγραμμα στον ελεγκτή PSoC.

**Python**

Εισαγωγή στο Mode 1:

Το Mode 1 αντιπροσωπεύει την αρχική κατάσταση του συστήματος, Κατανοώντας πλήρως τη λειτουργία του Mode 1, μπορούμε να αναδείξουμε τις βασικές αρχές και τις παραμέτρους που καθορίζουν τη συμπεριφορά του συστήματος.

Ακόμα, θα εξεταστούν οι πιθανές παράμετροι εισόδου που επηρεάζουν τη μετάβαση στο Mode 1 και οι συνέπειες αυτών των μεταβάσεων στη συνολική απόδοση του συστήματος.

Εκτέλεση Αρχείου Python μέσω του Command Prompt

Για να εκτελέσετε ένα αρχείο Python μέσω του Command Prompt, ακολουθήστε τα παρακάτω βήματα:

1. Άνοιγμα του Command Prompt:

Πατήστε Win + R για να ανοίξετε το παράθυρο "Run".

Εισάγετε cmd και πατήστε Enter.

2. Πλοήγηση στον Κατάλογο του Αρχείου Python:

Χρησιμοποιήστε την εντολή cd για να πλοηγηθείτε στον κατάλογο όπου βρίσκεται το αρχείο Python.

3. Εκτέλεση του Αρχείου Python:

Χρησιμοποιήστε την εντολή python ακολουθούμενη από το όνομα του αρχείου Python.

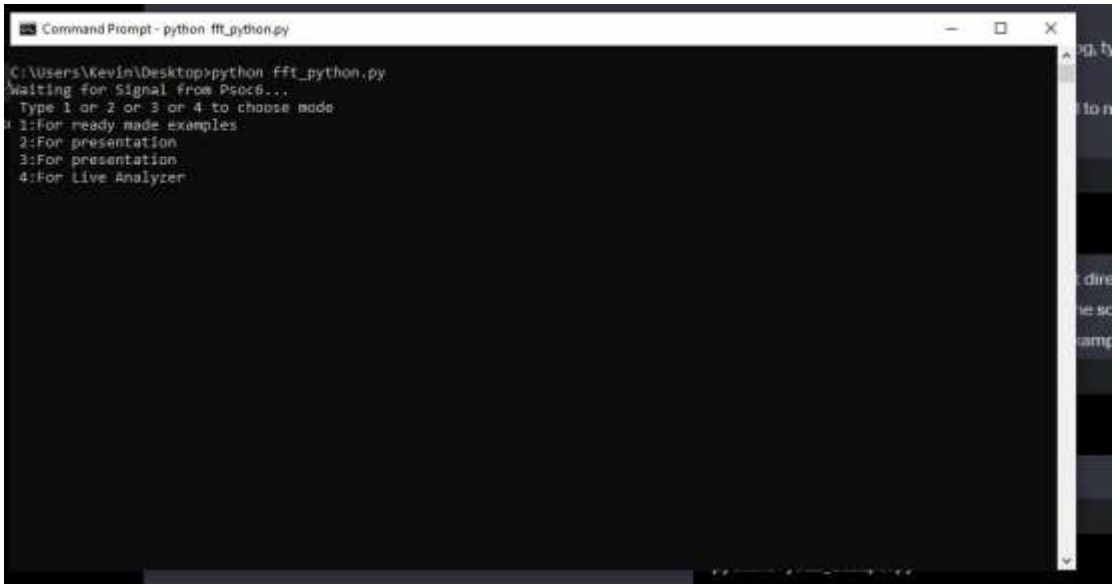
pythonfft_python.py

**Σημείωση:**

Βεβαιωθείτε ότι έχει εγκατασταθεί η Python στον υπολογιστή.

Βεβαιωθείτε ότι η διαδρομή προς τον εκτελέσιμο αρχείο Python προστέθηκε στο μεταβλητή περιβάλλοντος PATH.

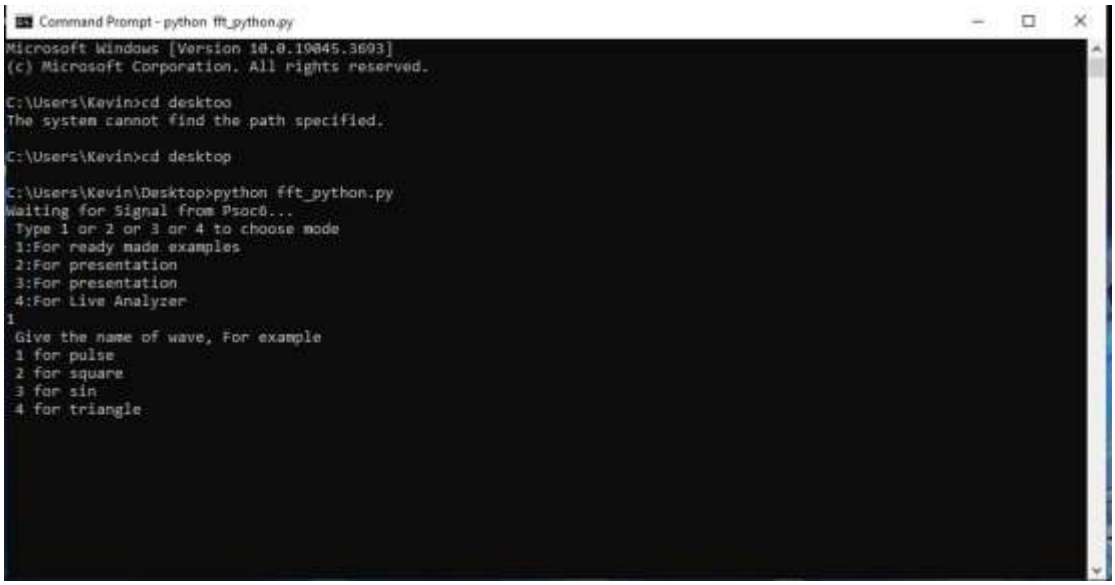
Με αυτόν τον τρόπο, εκτελείται επιτυχώς το Python αρχείο σας μέσω του Command Prompt.




Εικ. 17 Στιγμιότυπο της έξοδού της μονάδας UART με το που ξεκινήσει η εφαρμογή

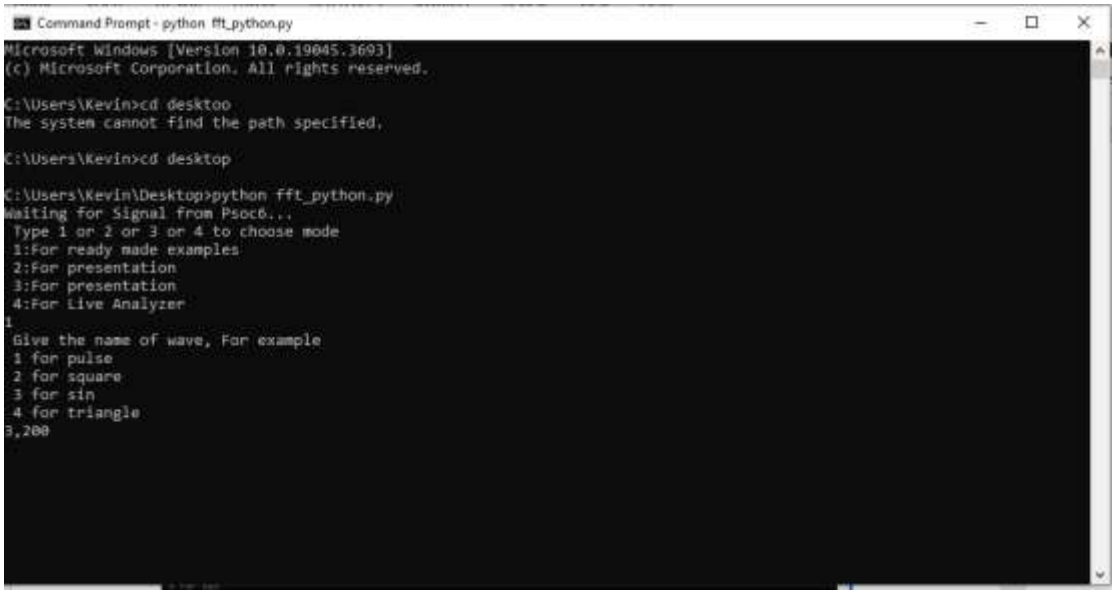
Εφόσον έχουν ολοκληρώσει τα παραπάνω βήματα στην οθόνη θα σας εμφανιστεί το κεντρικό μενού και ο χρήστης καλείται να διαλέξει ποιο mode θέλει να τρέξει,

Επιλέγει το νούμερο ένα , γράφοντας στην κονσόλα το νούμερο "1"  και πάτησε Enter (Εικ. 17),




**Εικ. 18 Στιγμιότυπο της έξοδού της μονάδας UART αφού έχουμε επιλέξει το mode  1**

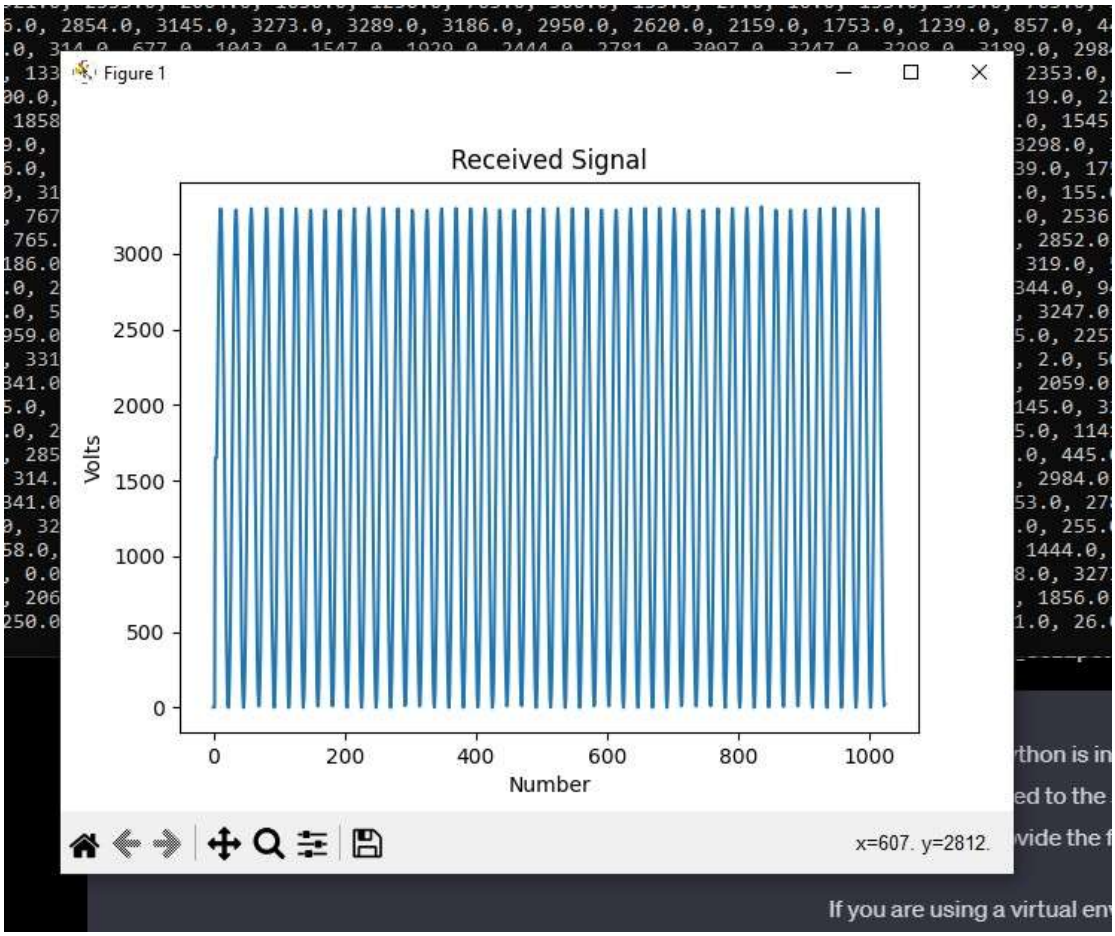
Στην συνέχεια διαλέγει τι είδους σήμα θέλει να στείλει και του δίνει μια περίοδο που θα την  χρησιμοποιήσει ο Timer_counter ώστε να ελέγχει την συχνότητα της κυματομορφής (Εικ. 18),



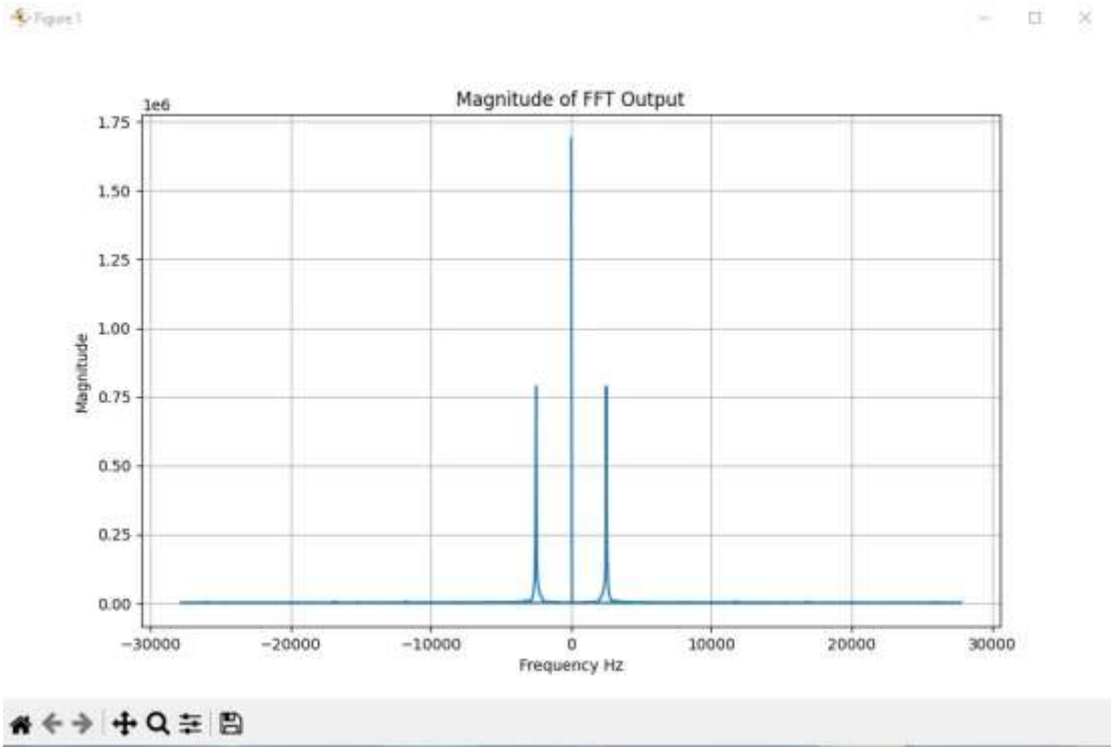

**Εικ. 19 Στιγμιότυπο της έξοδού της μονάδας UART πριν στείλουμε της οδηγίες στο mode 1**

Έστω ότι θέλει να στείλει ένα ημίτονο με περίοδο 200 ο τρόπος για να δώσει αυτήν την οδηγία είναι να γράψει "3,200" (Εικ. 19), αφού έδωσε την οδηγία περιμένει να

τελειώσει η μεταφορά τον δεδομένων για να εμφανιστεί το αρχικό σήμα (Εικ. 20), και έπειτα το αποτέλεσμα του μετασχηματισμού Fourier (Εικ. 21),




Εικ. 20 προβολή αρχικού σήματος.

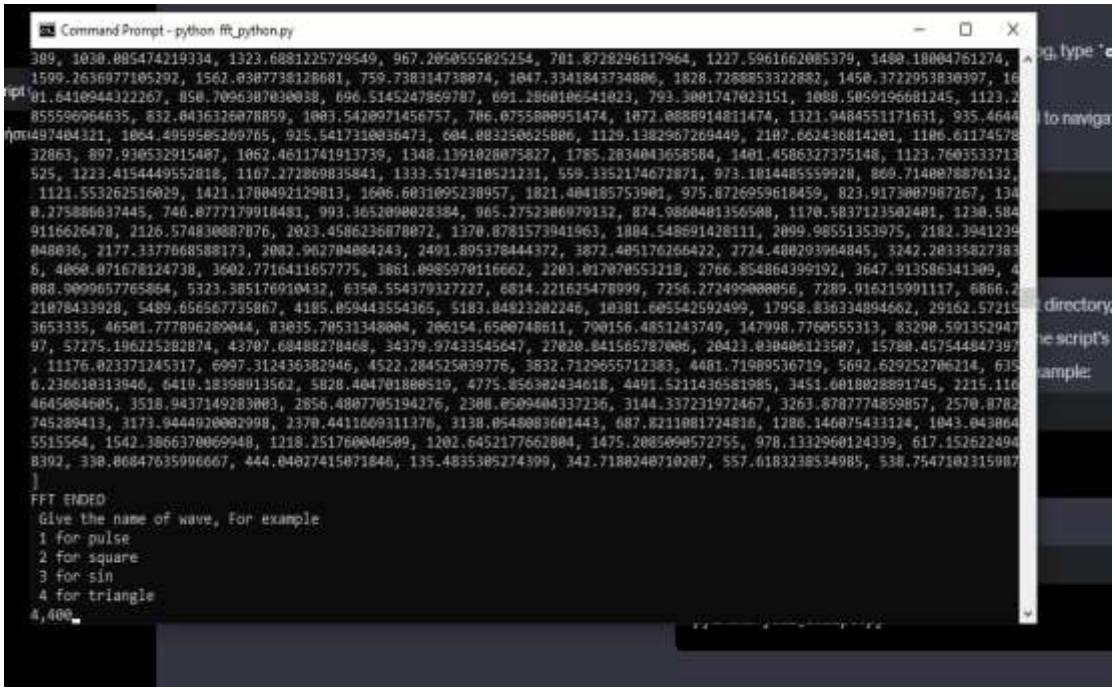



Εικ. 21 προβολή FFT του αρχικού σήματος.

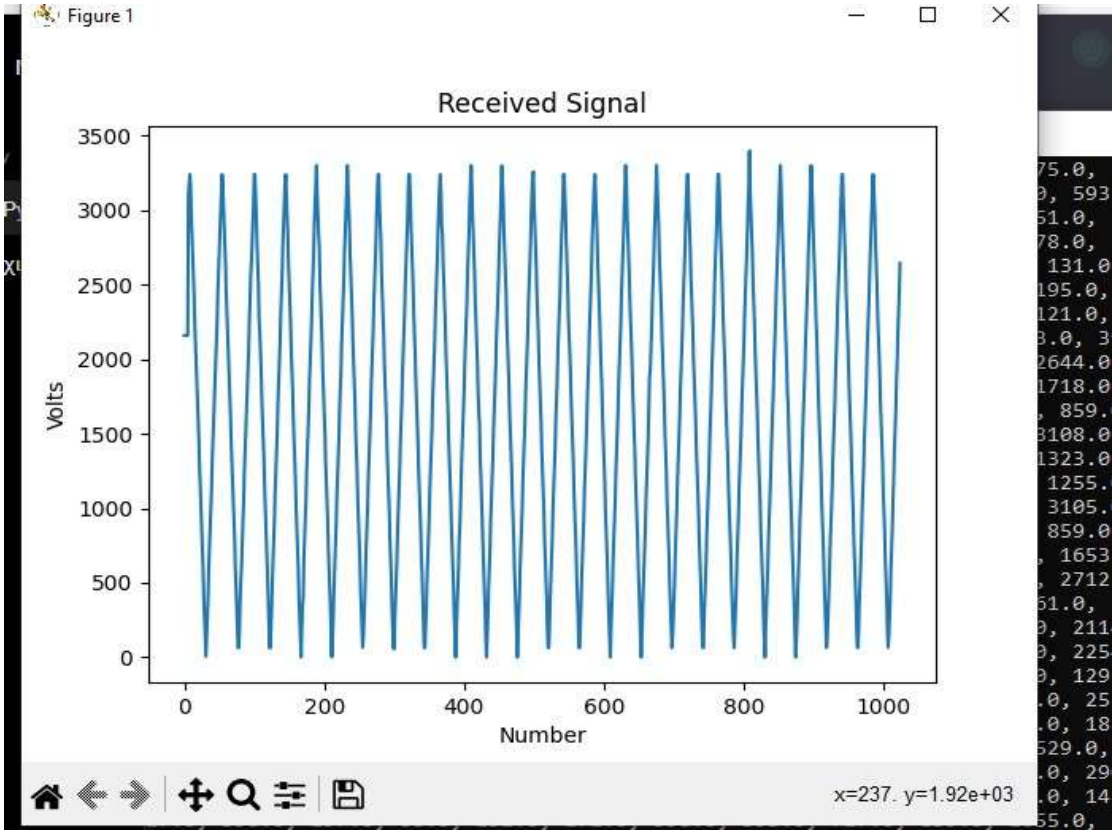
Αφού εμφανιστεί το αποτέλεσμα κλείνει το παράθυρου και δίνει εκ νέου να νέα οδηγία

έστω ότι θέλει έναν τριγωνικό παλμό με περίοδο 400 (Εικ. 22 , 23 ,24).

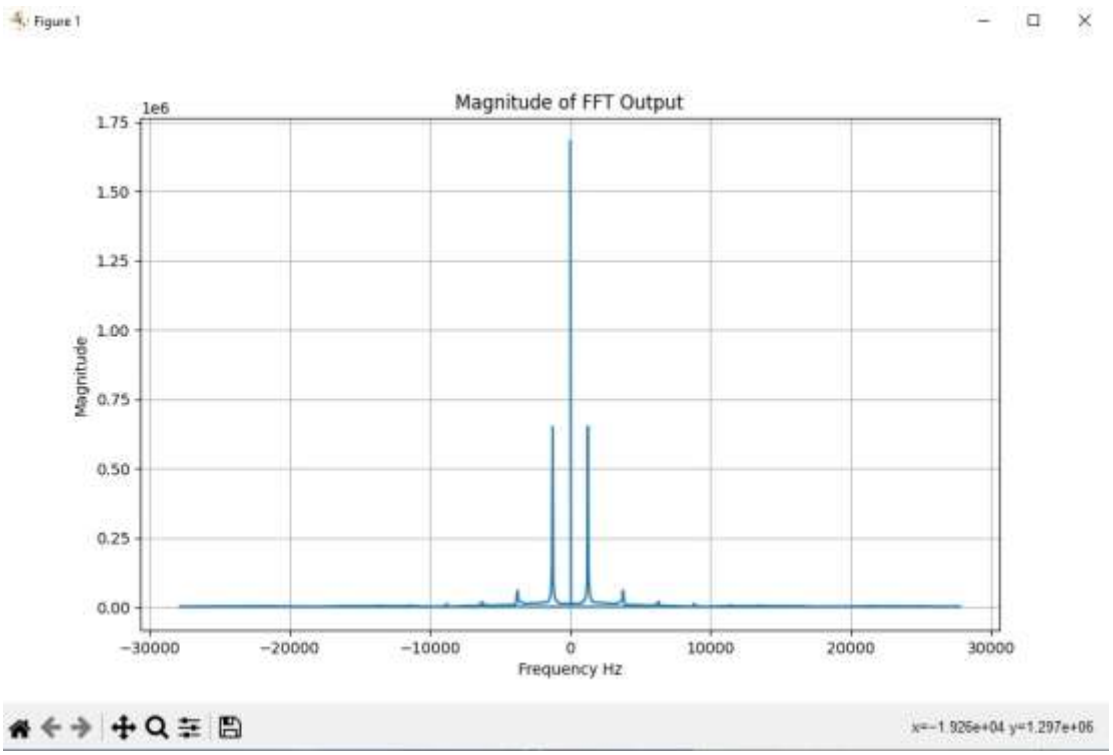
Στον οριζόντιο άξονα έχει τρεις γραμμές, η γραμμή στην συχνότητα μηδέν είναιη DCσυνιστώσα, το αποτέλεσμα του μετασχηματισμού φούριε βρίσκεται στην θετική μεριά του γραφήματος, και στην αρνητική μεριά του γραφήματος είναι το (mirroring).




Εικ. 22 Στιγμιότυπο της έξοδού της μονάδας UART οπού δίνουμε ξανά οδηγία για το νέο σήμα.

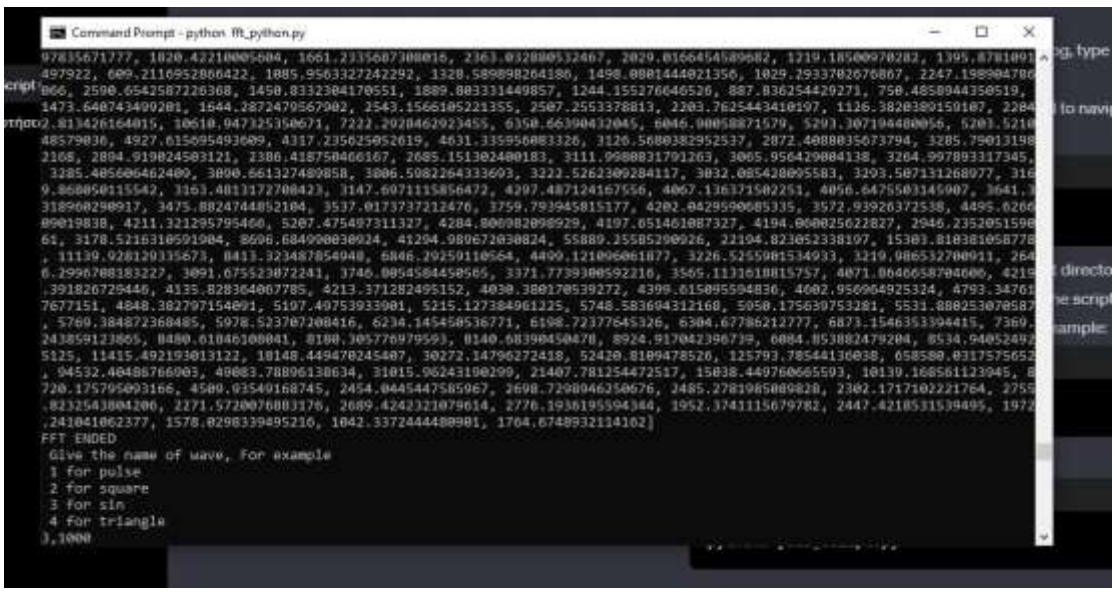



Εικ. 23 Προβολή τριγωνικού σήματος.

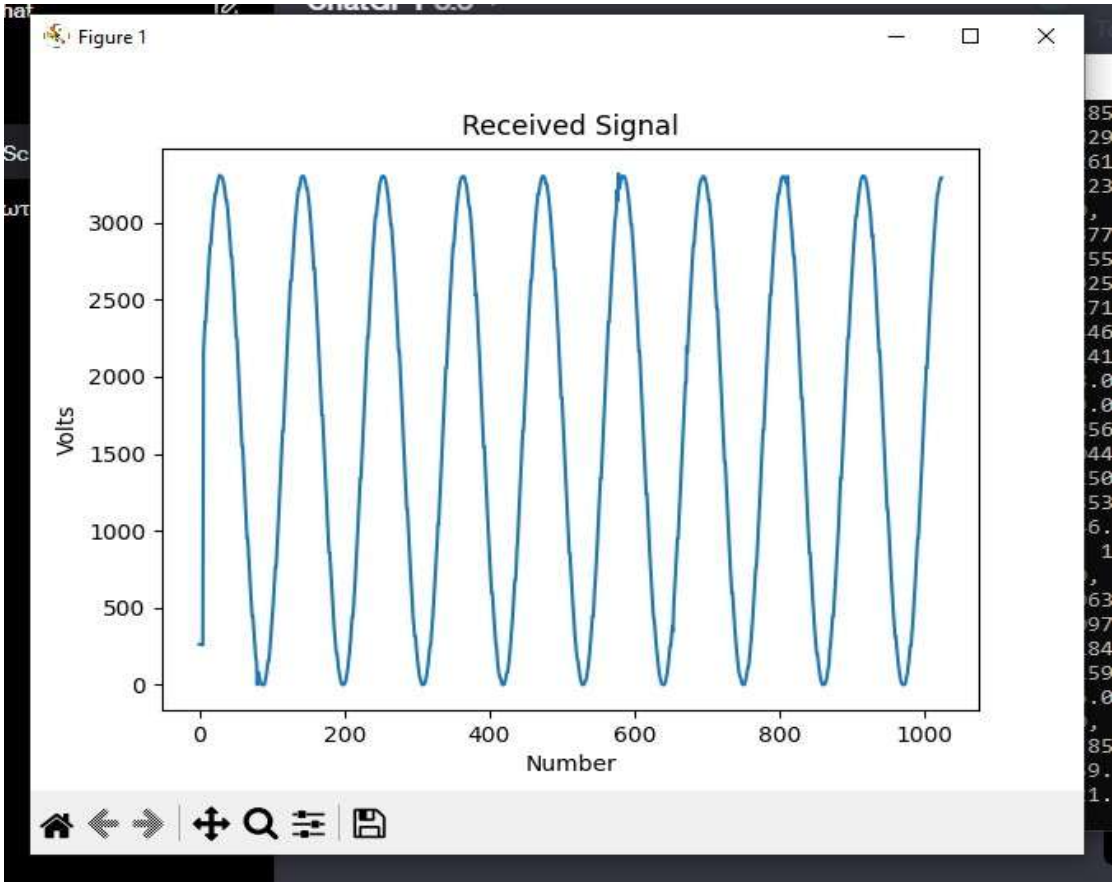



Εικ. 24 FFTτριγωνικού σήματος.

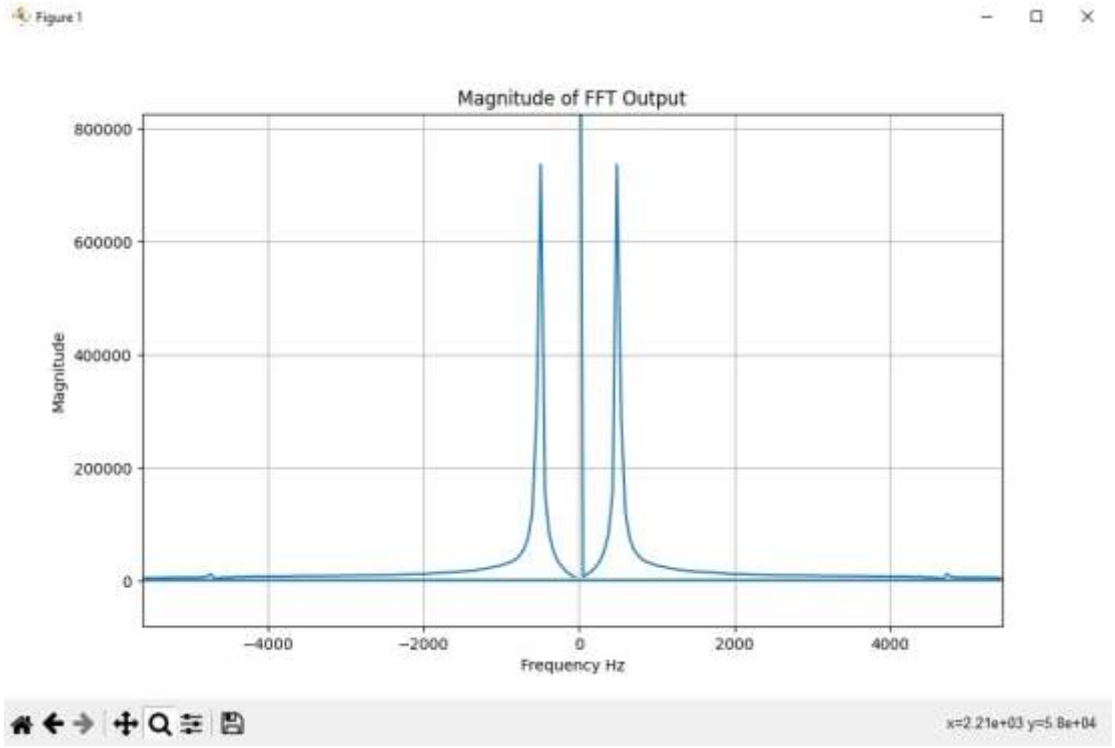
Τέλος ορίζει ένα ημίτονο ξανά άλλα με μεγαλύτερη περίοδο (1000) (Εικ. 25) οπού είναι ξεκάθαρο πως άλλαξε η συχνότητα του σήματος αλλά και το αποτέλεσμα του μετασχηματισμού Fourier.




Εικ. 25 Στιγμιότυπο της έξοδού της μονάδας UART οδηγία σήματος ημίτονου με περίοδο 1000.




Εικ. 26 Προβολή ημίτονου με περίοδο 1000.



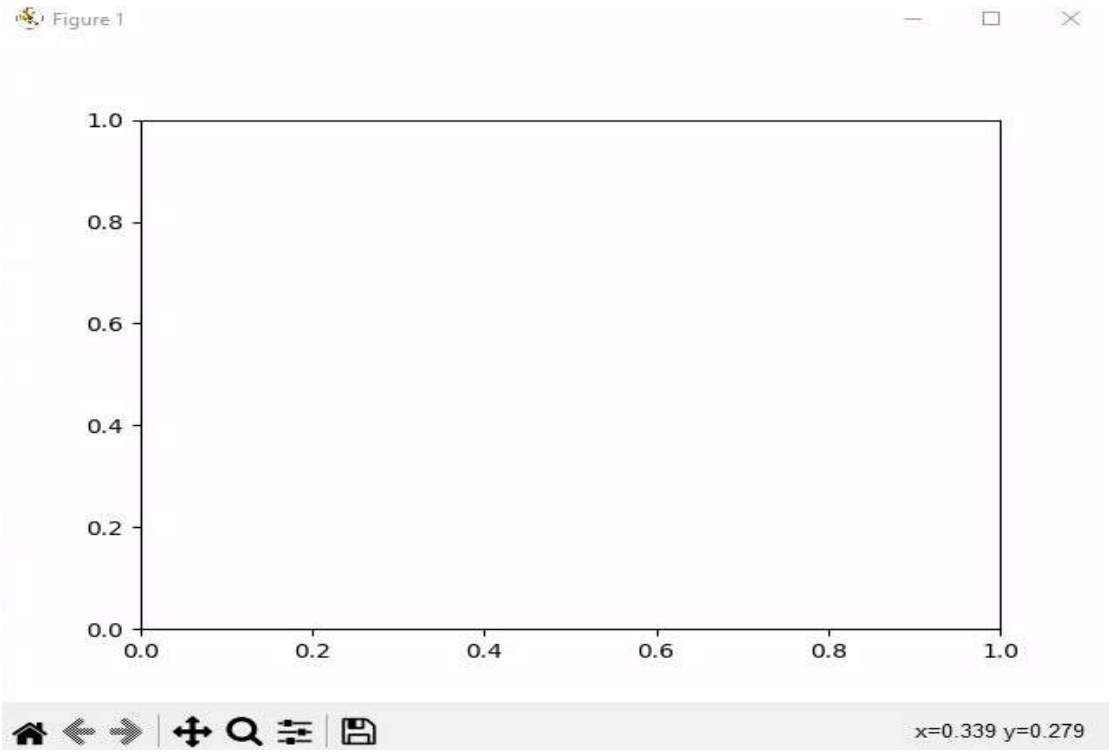
Εικ. 27 FFT ημίτονου με περίοδο 1000.

Εισαγωγή στο Mode 2:

Απαιτείται επανεκκίνηση του python αρχείου καθώς και κάνουμε επανεκκίνηση και το PSoc6, Έπειτα από το μενού στην την επιλογή 2 θα δείξει την (Εικ. 28),


Εικ. 28 Στιγμιότυπο της έξοδού της μονάδας UART κατά την έναρξη του προγράμματος.

Η λειτουργία πίσω από το mode δυο έχει περιγραφεί στις προηγούμενες ενότητες ωστόσο αυτό που περιμένουμε να δούμε είναι να αποστέλλεται ένα σήμα το οποίο όσο περνάει ο χρόνος θα μειώνεται η συχνότητα του άρα και συνεπώς η γραμμή που θα μας εμφανίζει το αποτέλεσμα του μετασχηματισμού Fourier θα τείνει προς το μηδέν (Εικ. 28),

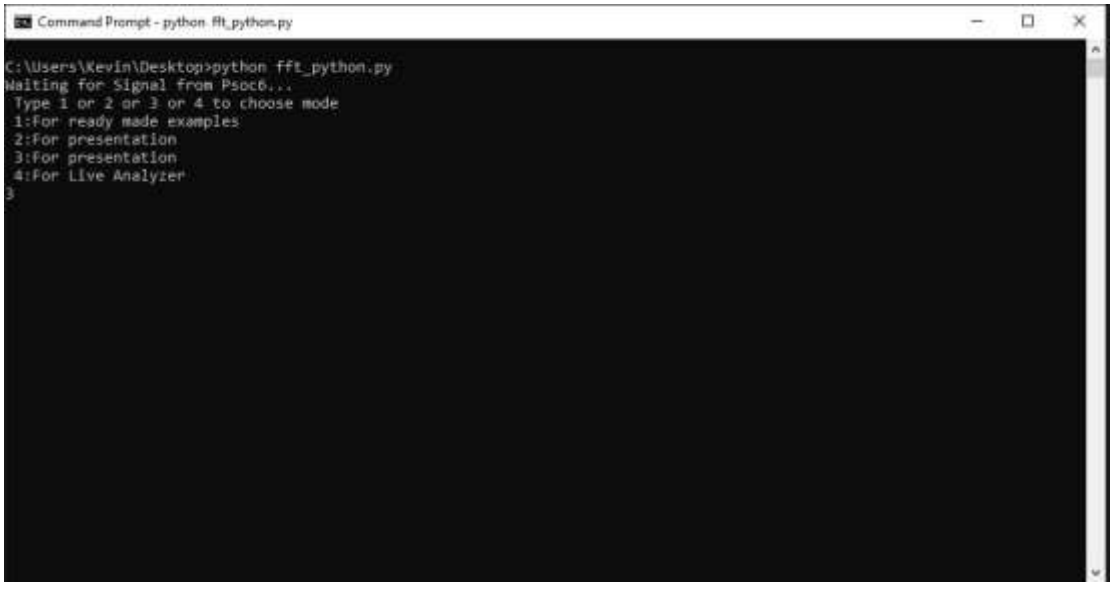



Εικ. 29 GIF με εικόνες οπού βλέπουμε την μετακίνηση της συχνότητας του σήματος

Εισαγωγή στο Mode 3:

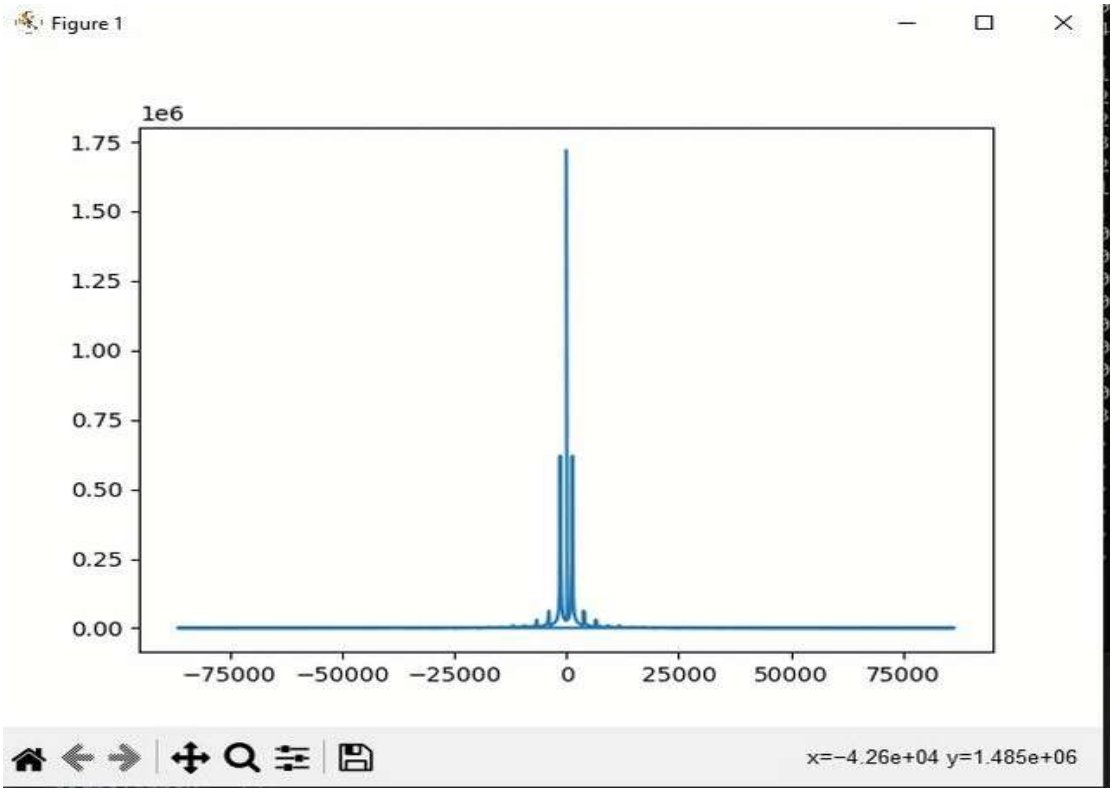
Απαιτείται επανεκκίνηση του python αρχείου καθώς και κάνουμε επανεκκίνηση και το PSoc6,

Έπειτα από το μενού στην την επιλογή 3 θα δείξει την Εικ. 30,




Εικ. 30 Στιγμιότυπο της έξοδού της μονάδας UART κατά την έναρξη του προγράμματος.

Η λειτουργία πίσω από το mode τρία έχει περιγραφεί στης προηγούμενες ενότητες ωστόσο αυτό που δείχνει είναι να δημιουργείται ένα εικονικό σήμα με έναν αλγόριθμο και να αυξάνεται η συχνότητα του όσο περνάει ο χρόνος (Εικ. 31).




Εικ. 31 GIF με εικόνες οπού βλέπουμε την μετακίνηση της συχνότητας του σήματος

Εισαγωγή στο Mode 4:

Απαιτείται επανεκκίνηση του python αρχείου καθώς και κάνουμε επανεκκίνηση και το PSoc6,

Έπειτα από το μενού στην την επιλογή 4 (Εικ. 32),


Εικ. 32 Στιγμιότυπο της έξοδού της μονάδας UART κατά την έναρξη του προγράμματος.

Σε αυτό το mode το πρόγραμμα περιμένει να διαβάσει σήματα από μια εξωτερική πηγή ώστε να υλοποιήσει τον μετασχηματισμό Fourier, η συχνότητα η οποία ανανεώνεται η γραφική παράσταση είναι περίπου στα 6 δευτερόλεπτα, και η δειγματοληψία κυμαίνεται από 120000-140000 khz

Σημείωση:

το πρόγραμμα περιμένει να λάβει σήμα από το Pin 10[5] σε περίπτωση που δεν εμφανίζει κάτι η γραφική παράσταση τότε ελέγξτε ξανά την συνδεσμολογία σας.

## Προτάσεις

Με βάση τις παρακάτω προτάσεις, προτείνεται η προσθήκη νέων λειτουργιών και βελτιώσεων για την αύξηση της λειτουργικότητας και της ακρίβειας του συστήματος.

Οπτικοποίηση Σήματος και FFT στο Ίδιο Plot:

Επιδιώκεται η ενσωμάτωση λειτουργικότητας στο πρόγραμμα Python ώστε το σήμα και το αποτέλεσμα του FFT να εμφανίζονται στο ίδιο γραφικό παράθυρο (plot). Αυτό θα παρέχει ολοκληρωμένη οπτική αναπαράσταση του σήματος και των συχνοτήτων που αντιστοιχούν σε αυτό.

Δυναμική Αλλαγή του Mode:

Επιθυμούμε την υλοποίηση μιας δυναμικής λειτουργίας που θα επιτρέπει την αλλαγή λειτουργικού mode κατά τη διάρκεια της εκτέλεσης του προγράμματος. Αυτό θα δίνει

τη δυνατότητα στον χρήστη να εξερευνά διάφορες ρυθμίσεις και λειτουργίες του συστήματος.

Αυξημένη Ακρίβεια της Συχνότητας Δειγματοληψίας:

Αναζητούμε τρόπους βελτίωσης της ακρίβειας της συχνότητας δειγματοληψίας μέσω της χρήσης πιο εξελιγμένων μεθόδων. Η βελτιωμένη ακρίβεια θα συνεισφέρει στην πιο αξιόπιστη ανάλυση των σημάτων.

Δημιουργία Οπτικής Εφαρμογής με Python:

Προτείνεται η δημιουργία μιας οπτικής εφαρμογής που θα επιτρέπει στους χρήστες να διερευνούν τις δυνατότητες του συστήματος μέσω μιας φιλικής προς τον χρήστη γραφικής διεπαφής. Η εφαρμογή θα παρέχει ευκολία στη χρήση και θα βοηθά στην καλύτερη κατανόηση των διαθέσιμων λειτουργιών.

Αυτές οι προτάσεις αποτελούν τη βάση για την εξέλιξη του υπάρχοντος προγράμματος και τη βελτίωση της εμπειρίας του χρήστη, ενώ παράλληλα προσφέρουν νέες δυνατότητες για την εξατομίκευση και την ανάλυση σημάτων σε πραγματικό χρόνο.

## Βιβλιογραφία / Διαδίκτυο


i. https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/linear_regulator/buxxtd3wg-e.pdf
ii. https://gr.mouser.com/datasheet/2/163/DS_FT234XD-220088.pdf
iii. https://www.infineon.com/cms/en/product/evaluation-boards/cy8ckit-005/
iv. https://www.infineon.com/dgdl/Infineon-CYBLE-416045-02-DataSheet-v02_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee63f136e57

# ΠαράρτημαΑ – ΚώδικαςPsoc Creator

**main_cm0p.c**

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include<stdio.h>
#include "ADC.h"
#include<stdlib.h>
#include<math.h>
#include "ctdac/cy_ctdac.h"
#include "dma/cy_dma.h"
#include<stdint.h>
#include<string.h>
#include<complex.h>

#definePI 3.14159265358979323846
voidcooley_tukey_fft(complexdouble* x, int n);

uint32_tsineWaveLUT[] = {
    0x7FF, 0x880, 0x900, 0x97F, 0x9FC, 0xA78, 0xAF1, 0xB67, 0xBD9,
0xC48,
    0xCB2, 0xD18, 0xD79, 0xDD4, 0xE29, 0xE77, 0xEC0, 0xF01, 0xF3C,
0xF6F,
    0xF9A, 0xFBE, 0xFDA, 0xFEE, 0xFFA, 0xFFF, 0xFFA, 0xFEE, 0xFDA,
0xFBE,
    0xF9A, 0xF6F, 0xF3C, 0xF01, 0xEC0, 0xE77, 0xE29, 0xDD4, 0xD79,
0xD18,
    0xCB2, 0xC48, 0xBD9, 0xB67, 0xAF1, 0xA78, 0x9FC, 0x97F, 0x900,
0x880,
    0x7FF, 0x77E, 0x6FE, 0x67F, 0x602, 0x586, 0x50D, 0x497, 0x425,
0x3B6,
    0x34C, 0x2E6, 0x285, 0x22A, 0x1D5, 0x187, 0x13E, 0x0FD, 0x0C2,
0x08F,
    0x064, 0x040, 0x024, 0x010, 0x004, 0x000, 0x004, 0x010, 0x024,
0x040,
    0x064, 0x08F, 0x0C2, 0x0FD, 0x13E, 0x187, 0x1D5, 0x22A, 0x285,
0x2E6,
    0x34C, 0x3B6, 0x425, 0x497, 0x50D, 0x586, 0x602, 0x67F, 0x6FE,
0x77E
};

uint32_tsquareWave[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092,
```

```
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092
};
uint32_ttriangleWave[] = {
    0,    82,   164,  246,  328,  410,  492,  574,
    656,  738,  820,  902,  984,  1066, 1148, 1230,
    1312, 1394, 1476, 1558, 1640, 1722, 1804, 1886,
    1968, 2050, 2132, 2214, 2296, 2378, 2460, 2542,
    2624, 2706, 2788, 2870, 2952, 3034, 3116, 3198,
    3280, 3362, 3444, 3526, 3608, 3690, 3772, 3854,
3936, 4018, 4092, 4018, 3936, 3854, 3772, 3690,
    3608, 3526, 3444, 3362, 3280, 3198, 3116, 3034,
    2952, 2870, 2788, 2706, 2624, 2542, 2460, 2378,
    2296, 2214, 2132, 2050, 1968, 1886, 1804, 1722,
    1640, 1558, 1476, 1394, 1312, 1230, 1148, 1066,
    984,  902,  820,  738,  656,  574,  492,  410,
    328,  246,  164,  82
};
uint32_tpulseWave[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092,
    4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092, 4092
};

#defineARRAY_SIZE 1024

int main(void){
    __enable_irq(); /* Enable global interrupts. */



VDAC_Start();
UART_Start();
ADC_Start();
    Counter_1_Start();



UART_PutString("Debug UART is up and running\n\r");


int16_t result;
int16_tresultmV;
uint32_tchan = 0UL;
complexdoublevaluesArray[ARRAY_SIZE];
intsignalArray[ARRAY_SIZE];
//DMA_Start(sineWaveLUT, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
uint16_t index = 0;
charpulse_name_from_serial_port = '0';
chartype_of_wave[20] = "";
int stage = 0;
boolhas_receive_order = false;
boolhas_receive_mode = false;
```

```
int mode = -1;
charperiod_string[9] = "";
intperiod_integer = -1;
charmyString[9] = "" ;
charmyMode[3] = "" ;
ADC_StartConvert();
inttimesSkipedLoop = 0;

int mode_3_fake_signal_variable = 0;
int mode_3_direction = 1;
int mode_3_step = 50;

CyDelay(100);




for(;;){

if(!has_receive_mode){
UART_GetArray(myMode , 2);
printf("myMode[0] = %c \n\r" , myMode[0]);
printf("myMode[1] = %c \n\r" , myMode[1]);

if (myMode[0] == '1') {
                mode = 1;
has_receive_mode = true;
            }elseif(myMode[0] == '2'){
                mode = 2;
                stage = 1;
period_integer = 60;
has_receive_mode = true;
            }elseif(myMode[0] == '3'){
                mode = 3;
                stage = 1;
has_receive_mode = true;
            }elseif(myMode[0] == '4'){
                mode = 4;
                stage = 1;
has_receive_mode = true;
//DMA_ADC_Start(sineWaveLUT, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                //CyDelay(100);
                //Counter_1_SetPeriod(200);
CyDelay(100);
            }
        }

if(mode == 1){
if(!has_receive_order){
//pulse_name_from_serial_port = UART_Get();
UART_GetArray(myString , 8);
printf("myString[0] = %c \n\r" , myString[0]);
printf("myString[1] = %c \n\r" , myString[1]);
printf("myString[2] = %c \n\r" , myString[2]);
printf("myString[3] = %c \n\r" , myString[3]);
printf("myString[4] = %c \n\r" , myString[4]);
printf("myString[5] = %c \n\r" , myString[5]);
printf("myString[6] = %c \n\r" , myString[6]);
printf("myString[7] = %c \n\r" , myString[7]);
printf("myString[8] = %c \n\r" , myString[8]);
```

```
if (myString[0] != '\0') {
pulse_name_from_serial_port = myString[0];
strcpy(period_string, myString + 2);
//period_string[7] = '\0';
period_integer = atoi(period_string);
printf("period int :%d \n\r", period_integer);
                    Counter_1_SetPeriod(period_integer);

CyDelay(1000);
stage = 1;
has_receive_order = true;
                }
            } else {
if(stage == 1){
if (pulse_name_from_serial_port == '1') {
strcpy(type_of_wave, "pulse");
                    } elseif(pulse_name_from_serial_port == '2') {
strcpy(type_of_wave, "square");
                    } elseif(pulse_name_from_serial_port == '3') {
strcpy(type_of_wave, "sin");
                    } elseif(pulse_name_from_serial_port == '4') {
strcpy(type_of_wave, "triangle");
}
if(strcmp(type_of_wave, "pulse") == 0){
DMA_ADC_Start(pulseWave, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                    }
if(strcmp(type_of_wave, "square") == 0){
DMA_ADC_Start(squareWave, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                    }
if(strcmp(type_of_wave, "sin") == 0){
DMA_ADC_Start(sineWaveLUT, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                    }
if(strcmp(type_of_wave, "triangle") == 0){
DMA_ADC_Start(triangleWave, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                    }
printf("pulse_name_from_serial_port: %c \n\t" ,
pulse_name_from_serial_port);
                    stage = 2;
                }
if(stage == 2){
//Read the Value of Volt on Pin_10_5

Cy_GPIO_Inv(P9_4_PORT, P9_4_PIN);

                    result = Cy_SAR_GetResult16(SAR, chan);
resultmV = Cy_SAR_CountsTo_mVolts(SAR, chan, result);

//Saves the Value into Array
signalArray[index] = resultmV;
valuesArray[index] = resultmV;

                    index = (index + 1) % ARRAY_SIZE;
//printf("Signal:%d  \r\n", resultmV);
                    //printf("Signal:%d  \r\n",  result);
printf("Loop counter :%d  \r\n", index);
if (index % ARRAY_SIZE == 0){
printf("you  can see the FFT now.\n\r");
                        stage = 3;
                    }
                }
if (stage == 3){
```

```
cooley_tukey_fft(valuesArray, ARRAY_SIZE);
                    stage = 4;
                }
if(stage == 4){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("Signal[%d] = %d \r\n", i, signalArray[i]);
                    }
                    stage = 5;
                }
if(stage == 5){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), ci-
mag(valuesArray[i]));
                    }
                    stage = 6;
                }
if(stage == 6){
UART_PutString("Ended.\n\r");
//printf("Ended");
                    stage = 7;
                }
if(stage == 7){
UART_PutString("Finished.\n\r");

memset(valuesArray, 0, sizeof(valuesArray));
memset(signalArray, 0, sizeof(signalArray));
                    index = 0;
pulse_name_from_serial_port = '0';
memset(type_of_wave, 0, sizeof(type_of_wave));
                    stage = 0;
has_receive_order = false;
memset(period_string, 0, sizeof(period_string));
strcpy(period_string, "");
period_integer = -1;
memset(myString, 0, sizeof(myString));
myString[0] = '\0';
myString[1] = '\0';
myString[2] = '\0';
myString[3] = '\0';
myString[4] = '\0';
myString[5] = '\0';
myString[6] = '\0';
myString[7] = '\0';
myString[8] = '\0';
memset(myMode, 0, sizeof(myMode));
myMode[0] = '\0';
myMode[1] = '\0';
DMA_ADC_Stop();
CyDelay(500);
                }
            }

        }
elseif(mode == 2){
if(stage == 1){
DMA_ADC_Start(sineWaveLUT, (uint32_t *)&(CTDAC0->CTDAC_VAL_NXT));
                stage = 2;
CyDelay(100);
            }
if(stage == 2){
                Counter_1_SetPeriod(period_integer);
```

```
//Read the Value of Volt on Pin_10_5

                result = Cy_SAR_GetResult16(SAR, chan);
resultmV = Cy_SAR_CountsTo_mVolts(SAR, chan, result);

//Saves the Value into Array
signalArray[index] = resultmV;
valuesArray[index] = resultmV;

                index = (index + 1) % ARRAY_SIZE;
//printf("Signal:%d  \r\n", resultmV);
                //printf("Signal:%d  \r\n",  result);
printf("Loop counter :%d  \r\n", index);
if (index % ARRAY_SIZE == 0){
printf("you  can see the FFT now.\n\r");
                    stage = 3;
                }
            }
if (stage == 3){
cooley_tukey_fft(valuesArray, ARRAY_SIZE);
                stage = 4;
            }
if(stage == 4){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("Signal[%d] = %d \r\n", i, signalArray[i]);
                }
                stage = 5;
            }
if(stage == 5){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), ci-
mag(valuesArray[i]));
                }
                stage = 6;
            }
if(stage == 6){
UART_PutString("Ended.\n\r");
//printf("Ended");
                stage = 7;
            }
if(stage == 7){
UART_PutString("Finished.\n\r");

memset(valuesArray, 0, sizeof(valuesArray));
memset(signalArray, 0, sizeof(signalArray));
                index = 0;
                stage = 2;

if(period_integer< 150){
period_integer = period_integer + 15;
                } elseif(period_integer>=  150
&&period_integer<500){
period_integer = period_integer + 100;
                } else{
period_integer = period_integer + 400;
                }
            }
        }
elseif(mode == 3){
if(stage == 1){
                mode_3_fake_signal_variable =
```

```
mode_3_fake_signal_variable + mode_3_step*mode_3_direction;

if(mode_3_fake_signal_variable >= 3300 ){
                    mode_3_direction = -1;
                    mode_3_fake_signal_variable = 3300;
                }
if(mode_3_fake_signal_variable < 50){
                    mode_3_direction = 1;
                    mode_3_fake_signal_variable = 0;
                }
resultmV = mode_3_fake_signal_variable;

//Saves the Value into Array
signalArray[index] = resultmV;
valuesArray[index] = resultmV;

                index = (index + 1) % ARRAY_SIZE;
//printf("Signal:%d  \r\n", resultmV);
                //printf("Signal:%d  \r\n",  result);
printf("Loop counter :%d  \r\n", index);
if (index % ARRAY_SIZE == 0){
printf("you  can see the FFT now.\n\r");
                    stage = 2;
                }
            }
if (stage == 2){
cooley_tukey_fft(valuesArray, ARRAY_SIZE);
                stage = 3;
            }
if(stage == 3){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("Signal[%d] = %d \r\n", i, signalArray[i]);
                }
                stage = 4;
            }
if(stage == 4){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), ci-
mag(valuesArray[i]));
                }
                stage = 5;
            }
if(stage == 5){
UART_PutString("Ended.\n\r");
//printf("Ended");
                stage = 6;
            }
if(stage == 6){
UART_PutString("Finished.\n\r");

memset(valuesArray, 0, sizeof(valuesArray));
memset(signalArray, 0, sizeof(signalArray));
                index = 0;
                stage = 1;

                mode_3_step = mode_3_step * 2;

//CyDelay(1000);
            }
        }
elseif(mode == 4){
```

```
if(stage == 1){

Cy_GPIO_Inv(P9_4_PORT, P9_4_PIN);

                result = Cy_SAR_GetResult16(SAR, chan);
resultmV = Cy_SAR_CountsTo_mVolts(SAR, chan, result);

if(resultmV != signalArray[index - 1]){
signalArray[index] = resultmV;
valuesArray[index] = resultmV;
                    index = (index + 1) % ARRAY_SIZE;
                } else {
timesSkipedLoop++;
                }

//printf("Signal:%d  \r\n", resultmV);
                //printf("Signal:%d  \r\n",  result);

if (index % ARRAY_SIZE == 0){
printf("you  can see the FFT now.\n\r");
                    stage = 2;
                }
            }
if (stage == 2){
cooley_tukey_fft(valuesArray, ARRAY_SIZE);
                stage = 3;
            }
if(stage == 3){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("Signal[%d] = %d \r\n", i, signalArray[i]);
                }
stage = 4;
            }
if(stage == 4){
for (inti = 0; i<ARRAY_SIZE; i++){
printf("FFT[%d] = %f + %fi\r\n ", i , creal(valuesArray[i]), ci-
mag(valuesArray[i]));
             }
printf("LOOPSKIPED = %d \n\r" , timesSkipedLoop);
                stage = 5;
            }
if(stage == 5){
UART_PutString("Ended.\n\r");
//printf("Ended");
                stage = 6;
            }
if(stage == 6){
UART_PutString("Finished.\n\r");

memset(valuesArray, 0, sizeof(valuesArray));
memset(signalArray, 0, sizeof(signalArray));
                index = 0;
                stage = 1;
timesSkipedLoop = 0;
//CyDelay(1000);
            }
        }
//CyDelay(1);
    }
}
```

```
voidcooley_tukey_fft(complexdouble* x, int n){
if (n <= 1)
return;

// Divide the input sequence into even and odd parts
complexdouble* even = (complexdouble*)malloc(n / 2 * si-
zeof(complexdouble));
complexdouble* odd = (complexdouble*)malloc(n / 2 * si-
zeof(complexdouble));

for (inti = 0; i< n / 2; i++)
    {
        even[i] = x[2 * i];
        odd[i] = x[2 * i + 1];
    }

// Recursive calls for FFT on even and odd parts
cooley_tukey_fft(even, n / 2);
cooley_tukey_fft(odd, n / 2);

// Combine the results
for (int k = 0; k < n / 2; k++)
    {
complexdouble t = cexp(-I * 2 * M_PI * k / n) * odd[k];
        x[k] = even[k] + t;
        x[k + n / 2] = even[k] - t;
    }

free(even);
    free(odd);
}


/* [] END OF FILE */
```

## Παράρτημα Β – Κωδικας Python

```
importserial.tools.list_ports
importmatplotlib.pyplotasplt
frommatplotlib.animationimportFuncAnimation
importnumpyas np
import time
import threading
from queue import Queue



# Serial port setup
ports =serial.tools.list_ports.comports()
serialInst=serial.Serial()
portName="COM4"
nameOfWave=""
serialInst.baudrate=115200
```

```
serialInst.port=portName
serialInst.open()

print("Waiting for Signal from Psoc6...")


# Initialize data
timesSkipedLoop=0
max_data_points=1024  # Maximum number of data points to display
sampling_rate=173000
frequencies =np.fft.fftfreq(max_data_points, 1/sampling_rate)
time_values= []
signalArray= []
realArray= []
imaginaryArray= []
fftArray= []
numbersArray= [iforiinrange(0, max_data_points)]
readingSignnal=1
fft_magnitudes= []
modeOfAnalyzer=-1
fft_magnitudes_queue= Queue()
plotReadyToGetUpdated=False

# Sample data for the arrays
arrays = [np.random.rand(100) for _ inrange(500)]  # Replace with
your arrays



# Create a function to update the plot
defupdate(frame):
    print("inside update")
    globalfft_magnitudes_queue
    globalplotReadyToGetUpdated
    global frequencies
    globalsignalArray
    globaltimesSkipedLoop
    if(plotReadyToGetUpdated):
        print("plotReadyToGetUpdated")
        plt.clf()  # Clear the current plot
        new_fft=  fft_magnitudes_queue.get()

        print(signalArray)
        #fft_result = np.fft.fft(signalArray)
        plt.plot(frequencies , new_fft)
        signalArray.clear()
        fftArray.clear()
        fft_magnitudes.clear()
        plotReadyToGetUpdated=False
```

```
        timesSkipedLoop=0

defgotoStepTwo():
    plt.plot(numbersArray, signalArray)
    print(signalArray)
    # Add labels and a title
    plt.xlabel('Number')
    plt.ylabel('Volts')
    plt.title('Received Signal')
    plt.draw()
    plt.pause(5)
    plt.close()
    gotoStepThree()

defgotoStepThree():
    plt.figure(figsize=(10, 6))
    plt.plot(fft_magnitudes)
    print(fft_magnitudes)
    plt.title("Magnitude of FFT Output")
    plt.xlabel("Index")
    plt.ylabel("Magnitude")
    plt.grid(True)
    plt.show()
    print("FFT ENDED")
    #time.sleep(1)
    plt.pause(0.1)
    plt.ioff()  # Turn off interactive mode
    plt.show()  # Display the final plot
    plt.close()
    gotoStepFour()

defgotoStepFour():
    globalreadingSignnal
    readingSignnal=1
    time_values.clear()
    signalArray.clear()
    fftArray.clear()
    fft_magnitudes.clear()

    nameOfWave=input(" Give the name of wave, For example \n 1 for
pulse \n 2 for square \n 3 for sin \n 4 for triangle \n")
    serialInst.write(nameOfWave.encode("utf-8"))

defchooseType():
    mode =input(" Type 1 or 2 or 3 or 4 to choose mode \n 1:For ready
made examples \n 2:For presentation \n 3:For presentation \n 4:For
Live Analyzer \n")
    globalmodeOfAnalyzer
```

```
    mode =int(mode)
    if(mode ==1):
        modeOfAnalyzer=1
        serialInst.write("1".encode("utf-8"))
    elif(mode ==2):
        modeOfAnalyzer=2
        serialInst.write("2".encode("utf-8"))
    elif(mode ==3):
        modeOfAnalyzer=3
        serialInst.write("3".encode("utf-8"))
    elif(mode ==4):
        modeOfAnalyzer=4
        serialInst.write("4".encode("utf-8"))
    else:
        modeOfAnalyzer=-2

defread_loop():
    globalfft_magnitudes_queue
    globalplotReadyToGetUpdated
    globaltimesSkipedLoop
    whileTrue:
       ifserialInst.in_waiting:
                packet =serialInst.readline()
                response =packet.decode('utf-8').strip('\r\n\t ')

                #print(response)
                if(response.startswith("Signal")):
                    signal =response.split('=')
                    signalArray.append(float(signal[1]))

                if(response.startswith("FFT")):
                    fftArray.append(response)
                if(response.startswith("LOOPSKIPED")):
                    loopskiped=response.split('=')
                    timesSkipedLoop=int(loopskiped[1]);
                    print(response)
                if(response.startswith("Ended")):
                    foriinrange(len(fftArray)):
                        parts =fftArray[i].split('=')
                        iflen(parts) ==2:
                            fftArray[i] = parts[1]

                    fft_output_complex= []
                    for s infftArray:
                        real_part, imag_part=s.split('+')
                        real =float(real_part)
                        imag=float(imag_part.replace('i',
'').replace(' ', ''))
```

```
                        fft_output_complex.append(complex(real, im-
ag))
                    fft_magnitudes= [np.abs(c) for c
infft_output_complex]
                    fft_magnitudes_queue.put(fft_magnitudes)
                    plotReadyToGetUpdated=True


chooseType()

if(modeOfAnalyzer==1):
    nameOfWave=input(" Give the name of wave, For example \n 1 for
pulse \n 2 for square \n 3 for sin \n 4 for triangle \n")
    serialInst.write(nameOfWave.encode("utf-8"))
    try:
        whileTrue:
        # Read data from serial port
            ifserialInst.in_waiting:
                packet =serialInst.readline()
                response =packet.decode('utf-8').strip('\r\n\t ')
                current_time=time.time()
                print(response)
                ifreadingSignnal==1:
                    if(response.startswith("Signal")):
                        signal =response.split('=')
                        signalArray.append(float(signal[1]))
                        time_values.append(current_time)

                    if(response.startswith("FFT")):
                        fftArray.append(response)
                    if(response.startswith("Ended")):

                        foriinrange(len(fftArray)):
                            parts =fftArray[i].split('=')
                            iflen(parts) ==2:
                                fftArray[i] = parts[1]

                        fft_output_complex= []
                        for s infftArray:
                            real_part, imag_part=s.split('+')
                            real =float(real_part)
                            imag=float(imag_part.replace('i',
'').replace(' ', ''))
                            realArray.append(float(real))
                            imaginaryArray.append(float(imag))

                            fft_output_complex.append(complex(real,
imag))
```

```
                        fft_magnitudes= [np.abs(c) for c
infft_output_complex]
                        # print("realArray \n \r ")
                        # print(realArray)
                        # print("imaginaryArray \n \r ")
                        # print(imaginaryArray)
                        gotoStepTwo()
    exceptKeyboardInterrupt:
        plt.ioff()  # Turn off interactive mode
        plt.show()  # Display the final plot
elif(modeOfAnalyzer==2):
    main_thread=threading.Thread(target=read_loop)
    main_thread.daemon=True  # Allow the thread to exit when the main
program finishes
    main_thread.start()

    fig, ax =plt.subplots()
    ani =FuncAnimation(fig, update, frames=1000, repeat=False, inter-
val=1000)  # Initialize an empty plot
    plt.show()
elif(modeOfAnalyzer==3):
    main_thread=threading.Thread(target=read_loop)
    main_thread.daemon=True  # Allow the thread to exit when the main
program finishes
    main_thread.start()

    fig, ax =plt.subplots()
    ani =FuncAnimation(fig, update, frames=1000, repeat=False, inter-
val=1000)  # Initialize an empty plot
    plt.show()
elif(modeOfAnalyzer==4):
    main_thread=threading.Thread(target=read_loop)
    main_thread.daemon=True  # Allow the thread to exit when the main
program finishes
    main_thread.start()

    fig, ax =plt.subplots()
    ani =FuncAnimation(fig, update, frames=1000, repeat=False, inter-
val=1000)  # Initialize an empty plot
    plt.show()

else:
    print("Please restart the procedure and type only 1 or 2 or 3 or
4")
```