ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Συστήματος Αυτόματης Αναγνώρισης Αντικειμένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνιος Β. Παπαδημητρίου

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Συστήματος Αυτόματης Αναγνώρισης Αντικειμένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνιος Β. Παπαδημητρίου

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Νοεμβρίου 2023.

.....	.....	.....
Τσανάκας Παναγιώτης Καθηγητής Ε.Μ.Π.	Σούντρης Δημήτριος Καθηγητής Ε.Μ.Π.	Παυλάτος Χρήστος Επίκουρος Καθηγητής, Σχολή Ικάρων

Αθήνα, Νοέμβριος 2023

.....
Αντώνιος Β. Παπαδημητρίου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η τεχνητή νοημοσύνη αποτελεί έναν κλάδο της πληροφορικής, ο οποίος παρουσιάζει ιδιαίτερη ανάπτυξη τα τελευταία χρόνια. Μία υποκατηγορία του κλάδου αυτού είναι η μηχανική μάθηση, η οποία διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Ο σκοπός της διπλωματικής εργασίας είναι, η ανάπτυξη ενός συστήματος αυτόματης αναγνώρισης αντικειμένων, κάνοντας χρήση της βιβλιοθήκης λογισμικού μηχανικής μάθησης OpenCv, του μικροϋπολογιστή Raspberry Pi , δυο σερβομηχανισμών και μιας κάμερας. Κατά τη λειτουργία του συστήματος, ένα αντικείμενο επιλέγεται απο το χρήστη και στη συνέχεια ανιχνεύεται αυτόματα, περιστρέφοντας τη κάμερα ανάλογα με τη κίνηση αυτού. Το σύστημα δύναται να συνδεθεί με αυτόματο πιλότο Μη Επανδρωμένου Αεροσκάφους, τον Pixhawk Cube, επικοινωνώντας μέσω του πρωτοκόλλου MavLink και εξασφαλίζοντας ότι οι κινήσεις του Αεροσκάφους θα ακολουθούν το αντικείμενο που αναγνωρίζεται. Επιπρόσθετα, το σύστημα μπορεί να επικοινωνήσει με τον προσομειωτή SITL(ArduPilot Simulator) έναντι σύνδεσης με τον αυτόματο πιλότο, όπου η πτήση και οι κινήσεις του αεροσκάφους μεταβάλλονται αντίστοιχα του αντικειμένου που αναγνωρίζεται.

Λέξεις Κλειδιά:

Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Αυτόματη Αναγνώριση Αντικειμένων

Abstract

Artificial intelligence is a branch of computer science that has shown significant development in recent years. One of its subfields is machine learning, which explores the study and construction of algorithms that can learn from data and make predictions about it. The purpose of this thesis is to develop an automatic object recognition system using the OpenCV machine learning software library, a Raspberry Pi microcomputer, two servos, and a camera. During the operation of the system, an object is selected by the user and then automatically detected, with the camera rotating accordingly to track its movement. The system can connect to an Unmanned Aerial Vehicle (UAV) autopilot, the Pixhawk Cube, by communicating through the MavLink protocol, ensuring that the aircraft's movements will follow the recognized object. Additionally, the system can communicate with the Software-in-the-Loop simulator (SITL, an ArduPilot simulator) instead of connecting to an autopilot. In this scenario, the flight and movements of the aircraft change in accordance with the recognized object.

Keywords:

Artificial intelligence, Machine learning, OpenCv, Raspberry Pi, UAV, Pixhawk Cube, MavLink

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Παναγιώτη Τσανάκα, ο οποίος εμπιστεύθηκε να μου αναθέσει αυτή τη διπλωματική εργασία. Ακόμη, οφείλω τεράστια ευγνωμοσύνη στον συμμετέχοντα στην επίβλεψη, κ. Χρήστο Παυλάτο για την πολύτιμη βοήθεια που μου προσέφερε καθ'όλη τη διάρκεια της εκπόνησης και της μελέτης της εργασίας μου.

Στη συνέχεια, θα ήθελα να ευχαριστήσω τους γονείς, τα αδέρφια μου, τη γιαγιά μου και τον παππού μου, οι οποίοι από τα παιδικά μου χρόνια μέχρι και σήμερα, μου συμπαραστέκονται και με εμψυχώνουν σε κάθε στίγμη.

Τέλος, θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου, τη σύζυγό μου Ελισάβετ, που όλα αυτά τα χρόνια είναι στο πλευρό μου, στις χαρές και στι λύπες, δίνοντας μου αγάπη, δύναμη και κουράγιο να συνέχισω να πραγματοποιώ τα όνειρά μου.

Αφιερώνω αυτή τη διπλωματική εργασία, στη σύζυγό μου Ελισάβετ και στη κόρη μου Μαρία.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

Η Τεχνητή Νοημοσύνη (ΤΝ) αποτελεί στον 21ο αιώνα την αιχμή της τεχνολογίας στο κλάδο της Πληροφορικής. Η σχεδίαση και η κατασκευή διαφόρων υπολογιστικών συστημάτων που κάνουν χρήση αυτής, έχουν ως στόχο να αντιγράψουν στοιχεία της ανθρώπινης συμπεριφοράς, τα οποία δείχνουν μια στοιχειώδη ευφυΐα. Αυτά τα στοιχεία είναι η μάθηση, η προσαρμοστικότητα, η εξαγωγή συμπερασμάτων, η κατανόηση απο τα συμφραζόμενα και η επίλυση προβλημάτων. Αντίστοιχα η Μηχανική Μάθηση αποτελεί ένα υποπεδίο της επιστήμης των υπολογιστών, που αναπύχθηκε απο τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην ΤΝ. Η Μηχανική Μάθηση διερευνά τη κατασκευή διαφόρων αλγορίθμων, οι οποίοι μπορούν να μαθαίνουν απο τα υπάρχοντα δεδομένα, έτσι ώστε να κάνουν προβλέψεις βασιζόμενοι σε αυτά ή να εξάγουν συμπεράσματα που εκφράζονται ως αποτέλεσμα.

Ο στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος αυτόματης αναγνώρισης αντικειμένων κάνοντας χρήση μιας βιβλιοθήκης λογισμικού Μηχανικής Μάθησης, την OpenCv. Το σύστημα αυτό απαρτίζεται από κατάληλλο υλισμικό (hardware) και λογισμικό (software) όπου με την απαραίτητη διασύνδεση πετυχαίνει το σκοπό του. Η δυνατότητα διασύνδεσής του σε Συστήματα μη Επανδρωμένων Αεροσκαφών (ΣμηΕΑ) διευρύνει τη λειτουργία του και τους τρόπους με τους οποίους μπορέι να χρησιμοποιηθεί.

Η παρούσα διπλωματική εργασία οργανώνεται σε εννέα (9) κεφάλαια όπου αναλύονται τόσο το θεωρητικό όσο και το πρακτικό μέρος της έρευνας για τη δημιουργία του συστήματος αυτόματης αναγνώρισης αντικειμένων και τη χρησιμοποίησή του. Τα κεφάλαια αυτά οργανώνονται όπως παρακάτω.

Στο Κεφάλαιο δύο (2) γίνεται μια εισαγωγή στην Τεχνητή Νοημοσύνη και στην Μηχανική μάθηση που αποτελεί έναν τομέα αυτής. Στην συνέχεια γίνεται μια εκτενής αναφορά στη βιβλιοθήκη λογισμικού μηχανικής μάθησης OpenCv, η οποία αποτελεί πρωταρχικό ρόλο σην αυτόματη αναγνώριση αντικειμένων κάνοντας χρήση των ανιχνευτών της.

Στο Κεφάλαιο τρία (3) γίνεται η παρουσίαση του απαραίτητου υλισμικού που χρησιμοποιείται για την κατασκευή του συτήματος αυτόματης αναγνώρισης αντικειμένων, καθώς και ο τρόπος διασύνδεσής του.

Στο Κεφάλαιο τέσσερα (4) γίνεται μια εισαγωγή στα Συστήματα Αυτομάτου Ελέγχου και στη συνέχεια εκτενής αναφορά στον ελεγκτή PID, ο οποίος χρησιμοποιείται πρακτικά στο συτήμα αυτόματης αναγνώρισης αντικειμένων.

Στο Κεφάλαιο πέντε (5) γίνεται μια εισαγωγή στα Συστήματα μη Επανδρωμένων Αεροσκαφών (ΣμηΕΑ) και εκτενής αναφορά στον αυτόματο πιλότο Pixhawk Cube, στον προσομοιωτή SITL(Ardupilot Simulation) και στο πρωτόκολλο επικοινωνίας MavLink. Το συτήμα αυτόματης αναγνώρισης αντικειμένων δύναται να συνδεθεί με τον αυτόματο πιλότο ή με τον προσομοιωτή και μέσω του πρωτοκόλλου επικοινωνίας να οδηγήσει κατάλληλα ένα ΣμηΕΑ.

Στο Κεφάλαιο έξι (6) γίνεται μια ανάλυση στις διεργασίες των υπολογιστικών συστημάτων, καθώς και στον συγχρονισμό αυτών. Το σύστημα αυτόματης αναγνώρισης αντικειμένων κάνει χρήση διαφορετικών διεργασιών, οι οποίες συγχρονίζονται, έτσι ώστε να γίνει επίτευξη του στόχου του.

Στο Κεφάλαιο εφτά (7) γίνεται η περιγραφή του συστήματος αυτόματης αναγνώρισης αντικειμένων, με την πλήρη ανάλυση της λειτουργίας του, τους τρόπους χρήσης του και γίνεται αναφορά σε μελλοντικές επεκτάσεις.

Στο Κεφάλαιο οχτώ (8) περιγράφεται η συγγραφή κώδικα σε γλώσσα προγραμματισμού Python καθώς και η ανάλυση αυτού. Ο κώδικας αυτός συντάσσει το πλέον ολοκληρωμένο λογισμικό για την αυτόματη αναγνώριση αντικειμένων και την οδήγηδη ενός ΣμηΕΑ.

Στο Κεφάλαιο εννέα (9) η παρούσα διπλωματική εργασία κλείνει με τον επίλογο.

2 Τεχνητή Νοημοσύνη

2.1 Εισαγωγή

Η Τεχνητή Νοημοσύνη (ΤΝ) αποτελεί στον 21ο αιώνα την αιχμή της τεχνολογίας στο κλάδο της Πληροφορικής. Η σχεδίαση και η κατασκευή διαφόρων υπολογιστικών συστημάτων που κάνουν χρήση αυτής, έχουν ως στόχο να αντιγράψουν στοιχεία της ανθρώπινης συμπεριφοράς, τα οποία δείχνουν μια στοιχειώδη ευφυΐα. Αυτά τα στοιχεία είναι η μάθηση, η προσαρμοστικότητα, η εξαγωγή συμπερασμάτων, η κατανόηση απο τα συμφραζόμενα και η επίλυση προβλημάτων. Η εφαρμογή της τα τελευταία χρόνια σε πολλούς ερευνητικούς τομείς ολο'ενα και πληθαίνει. Πολλές επιστήμες στο πέρασμα του χρόνου συνεργάστηκαν για να δημιουργήσουν μία έξυπνη συμπεριφορά ενός υπολογιστικού συστήματος, το οποίο περιείχε στοιχεία συλλογιστικής, μάθησης και προσαρμογής σε ένα περιβάλλον. Αυτές οι επιστήμες είναι η πληροφορική, η ψυχολογία, η φιλοσοφία, η νευρολογία, η επιστήμη των μηχανικών και η γλωσσολογία. Η ΤΝ έχει κατηγοριοποιηθεί, ανάλογα με τον αναγκαίο επιστημονικό στόχο και σε άλλους τομείς. Αυτοί είναι η επίλυση προβλημάτων, η ανακάλυψη γνώσης, τα συστήματα γνώσης και η μηχανική μάθηση. Ακόμη, η επεξεργασία φυσικής γλώσσας, η ρομποτική και η μηχανική όραση, αποτελούν παρόμοια επιστημονικά πεδία, τα οποία μπορούν να θεωρηθούν και ως ανεξάρτητα της αυτής. Τέλος, καθώς η σύγχρονη ΤΝ εξελίσσεται με γρήγορους ρυθμούς, εκμεταλλεύεται ολοένα και περισσότερο εργαλεία εφαρμοσμένων μαθηματικών και επιστημών μηχανικών, σε σύγκριση με παλαιότερα εργαλεία που προερχόταν από τη μαθηματική λογική και τη θεωρητική πληροφορική.

2.2 Βασικές Αρχές της Τεχνητής Νοημοσύνης

Στο πέρασμα του χρόνου πολλοί κλάδοι της ανθρωπότητας συνείσφεραν ιδέες, απόψεις και τεχνικές στην Τεχνητή Νοημοσύνη. Οι κλάδοι αυτοί προσπάθησαν να απαντήσουν σε συγκεκριμένες ερωτήσεις χωρίς να έχουν απαραίτητα την ΤΝ ως τελικό τους στόχο. Ακολοθούν οι ερωτήσεις και η χρονολογική προσέγγιση:

α. Φιλοσοφία (Από το 428 π.Χ. μέχρι σήμερα)

- Δύναται η χρήση τυπικών κανόνων να οδηγούν στην εξαγωγή έγκυρων συμπερασμάτων;
- Πώς από έναν φυσικό εγκέγαλο γίνεται να δημιουργηθεί πνευματική νόηση;
- Ποιά είναι η προέλευση της γνώσης;
- Με ποιό τρόπο η γνώση μπορεί να οδηγήσει σε δράση;

β. Μαθηματικά (Από το 800 μέχρι σήμερα)

- Από ποιούς τυπικούς κανόνες μπορεί να εξαχθεί ένα έγκυρο συμπέρασμα;
- Πώς μπορεί να θεωρηθεί κάτι υπολογίσιμο και πώς μη υπολογίσιμο;
- Με ποιό τρόπο μπορεί ένας άνθρωπος να συλλογιστεί όταν οι πληροφορίες είναι ασαφείς;

γ. Οικονομικά (Από το 1776 μέχρι σήμερα)

- Με ποιό τρόπο πρέπει να γίνεται η λήψη αποφάσεων ώστε η απολαβή να μεγιστοποιείται;
- Με ποιό τρόπο πρέπει να γίνεται η λήψη αποφάσεων όταν άλλοι άνθρωποι δεν συμπεριφέρονατι με ευνοϊκό τρόπο;
- Με ποιό τρόπο πρέπει να γίνεται η λήψη αποφάσεων, όταν η απολαβή μπορεί να καταστεί δυνατή μελλοντικά;

δ. Νευροεπιστήμες (Από το 1861 μέχρι σήμερα)

- Με ποιό τρόπο οι πληροφορίες επεξεργάζονται από τον εγκέφαλο;

ε. Ψυχολογία (Από το 1879 μέχρι σήμερα)

- Με ποιό τρόπο οι άνθρωποι και τα ζώα σκέπτονται και ενεργούν;

στ. Τεχνολογία Υπολογιστών (Από το 1940 μέχρι σήμερα)

- Με ποιό τρόπο δύναται να κατασκευαστεί ένας αποδοτικός υπολογιστής;

ζ. Θεωρία ελέγχου και κυβερνητική (Από το 1948 μέχρι σήμερα)

- Με ποιό τρόπο μπορούν τα κατασκευάσματα να λειτουργούν με έλεγχο που διεξάγεται από αυτά;

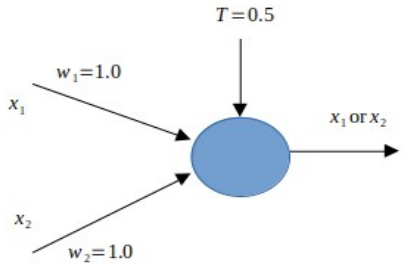
η. Γλωσσολογία (Από το 1957 μέχρι σήμερα)

- Με ποιό τρόπο μπορούν η γλώσσα με τη σκέψη να συσχετισθούν;

2.3 Ιστορική αναδρομή

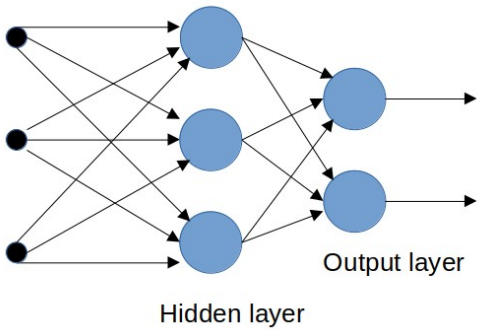
Η πρώτη μαθηματική περιγραφή ενός τεχνητού νευρωνικού δικτύου, του οποίου οι δυνατότητες ήταν αρκετά περιορισμένες, έλαβε χώρα τη δεκαετία του 1940. Η εμφάνιση μηχανών με ευφυΐα άρχιζε να ακμάζει μετά τον Β' Παγκόσμιο Πόλεμο, καθώς οι υπολογιστικές συσκευές που είχαν κατασκευαστεί μέχρι τότε ήταν πολύ διαφορετικές. Η

μεγάλη επανάσταση στην τεχνητή νοημοσύνη ήρθε το 1950, όταν ο μαθηματικός Άλαν Τούρινγκ, που θεωρείται ο πατέρας της θεωρίας υπολογισμού, θέσπισε τη δοκιμασία Τούρινγκ. Αυτή η δοκιμασία είχε ως σκοπό να ελέγξει αν μια μηχανή διαθέτει ευφυΐα. Το πρώτο πρόγραμμα τεχνητής νοημοσύνης εμφανίσθηκε το 1956 με το όνομα Logic Theorist. Αυτό, είχε τη δυνατότητα να αποδεικνύει μαθηματικά θεωρήματα, κάνοντας χρήση ευρετικών αλγορίθμων αναζήτησης και συμπερασματικών κανόνων.



Εικόνα 1. Αποτύπωση και λειτουργία ενός απλού τεχνητού νευρώνα.

Μία επόμενη σημαντική εξέλιξη , χάριν στην οποία δημιουργήθηκαν πολλές εφαρμογές τεχνητής νοημοσύνης, επήλθε το 1958 από τον Μακάρθι, ο οποίος δημιούργησε την γλώσσα συναρτησιακού προγραμματισμού LISP. Επίσης, το 1958 ο Φρίντεμπεργκ παρουσίασε τους γενετικούς αλγορίθμους , ενώ το 1962 ο Ρόσενμπλατ παρουσίασε το βελτιωμένο νευρωνικό δίκτυο perceptron. Όλα τα διαθέσιμα εργαλεία ΤΝ εκείνη την χρονική περίοδο επίλυαν πολύ απλά προβλήματα, με αποτέλεσμα τα ερευνητικά προγράμματα να μην υποστηρίζονται χρηματικά, προκαλώντας καθ'αυτό το τρόπο μια ύφεση στην ανάπτυξη της ΤΝ. Ο τομέας αυτός ξεκίνησε να ενδιαφέρει και πάλι στα μέσα του 1970, όταν συστήματα με συγκεκριμένες μηχανές ΤΝ, χρησιμοποιήθηκαν σε εμπορικές εφαρμογές και εξήγαγαν λογικά συμπεράσματα, αντίστοιχα με αυτά ενός ανθρώπου, ο οποίος εξειδικευόταν στον αντίστοιχο τομέα. Η ανάπτυξη της γλώσσας λογικού προγραμματισμού Prolog, την ίδια χρονική περίοδο, έδωσε μια νέα διάσταση στην ΤΝ, όπως και η ανάπυξη, υλοποίηση πιο σύνθετων νευρωνικών δικτύων στις αρχές της δεκαετίας του 1980. Άξια αναφοράς αποτελούν τα δίκτυα Hopfield και τα πολυεπίπεδα perceptron.



Εικόνα 2. Αποτύπωση ενός απλού τεχνητού νευρωνικού δικτύου δύο επιπέδων.

Η διάδοση του Διαδικτύου, στη δεκαετία του 1990, επέφερε τη χρήση των ευφυών πρακτόρων. Αυτοί αποτελούσαν ένα αυτόνομο λογισμικό ΤΝ, το οποίο ήταν τοποθετημένο σε ένα περιβάλλον αλληλεπίδρασης, παρέχοντας κατάλληλη βοήθεια στους χρήστες του. Επίσης είχαν τη δυνατότητα να συλλέγουν και να αναλύουν μεγάλα σύνολα δεδομένων και

να αυτοματοποιούν επαναλαμβανόμενες εργασίες. Η σχεδίαση και η υλοποίηση ευφυών πρακτόρων, ορίζουν πολλές φορές την ΤΝ.

Στον Πίνακα 1παρουσιάζεται η Ιστορική εξέλιξη της Τεχνητής Νοημοσύνης.

Χρόνος	**Ιστορική Εξέλιξη**
1950	Περιγράφεται η δοκιμή Τούρινγκ από τον Άλαν Τούρινγκ. Η επιδίωξη είναι η συμμετοχή μιας μηχανής σε μια ανθρώπινη συνομιλία.
1951	Ο Κρίστοφερ Στράκλι γράφει ένα πρόγραμμα ΤΝ που παίζει ντάμα. Ο Ντίτριχ Πρίνζ γράφει ένα πρόγραμμα ΤΝ που παίζει σκάκι.
1956	Διάσκεψη του Νταρτμουθ: Ο όρος «Τεχνητή Νοημοσύνη» παρουσιάζεται ως κύριο θέμα από τον Τζον Μακάρθι.
1958	Ανάπτυξη της γλώσσας συναρτησιακού προγραμματισμού LISP από τον Τζον Μακάρθι.
1965	Αναπτύσσεται το έμπειρο σύστημα Dendral από τον Έντουαρτ Φάιγκενμπαουμ για το συμπερασμό της μοριακής δομής οργανικών ενώσεων, κάνοντας χρήση ενδείξεων από επιστημονικά όργανα.
1972	Ο Άλαν Κολμεροέρ αναπτύσσει τη γλώσσα λογικού προγραμματισμού Prolog.
1991	Ο Αμερικάνικος Στρατός χρησιμοποιεί το πρόγραμμα ΤΝ, με όνομα DART, για τη σχεδίαση ενεργειών στον Α’ Πόλεμο του Κόλπου.
1994	Επίδειξη αυτόνομης οδήγησης αυτοκινήτου στο Παρίσι από του Ντίκμαννς και Ντάιμλερ-Μπενζ.
1997	Ο παγκόσμιος πρωταθλητής στο σκάκι Γκάρι Κασπάροφ ητείται από τον υπολογιστή Deep blue της IBM.
1998	Το παιχνίδι ΤΝ Φέρμπι, της Tiger Electronics κυκλοφορεί επιτυχημένα στην αγορά.
1999	Το πρώτο κατοικίδιο ΤΝ, με όνομα AIBO, κυκλοφορεί από την Sony.
2000	Εξερεύνηση της Ανταρκτικής από το ρομπότ Nomad.
2004	Διαγωνισμός δημιουργίας αυτόνομων οχημάτων, με χρηματικό έπαθλο, από την DARPA.
2005	Προσομοίωση εγεφάλου, σε μοριακό επίπεδο, από το πρόγραμμα Blue Brain.
2009	Δημιουργία του πρώτου αυτο – οδηγούμενου αυτοκινήτου από την Google.
2013	Ανάπτυξη συστήματος από την Deep Mind, το οποίο μπορεί να παίξει διάφορα παιχνίδια Atari. Η εκπαίδευση έγινε χρησιμοποιώντας για είσοδο μόνο εικονοστοιχεία/καρέ, χωρίς χρήση κανόνων ή κάποιας γνώσης των παιχνιδιών.
2017	Η Google Brain, μέσω του άρθρου «Attention is all you need» του Ashish Vaswani και άλλων επιστημόνων, παρουσιάζει την αρχιτεκτονική του transformer.
2022	Δημιουργία του συστήματος ΤΝ chatGPT 3 από την OpenAI. Το σύστημα βασίζεται στην αρχιτεκτονική του transformer και εμφανίζει πολλές δεξιότητες όπως γνώσεις σε μεγάλη θεματολογία, συγγραφή κειμένων, συγγραφή κώδικα σε διάφορες γλώσσες προγραμματισμού κ.α.

Πίνακας 1. Ιστορική εξέλιξη της Τεχνητής Νοημοσύνης

2.4 Σύγχρονη Τεχνολογία

Ένα μέγαλο ερώτημα είναι τί μπορεί να κάνει η Τεχνητή Νοημοσύνη σήμερα; Οι απαντήσεις ποικίλουν, καθώς η ΤΝ δραστηριοποιείται σε πολλούς τομείς. Στη συνέχεια, παρουσιάζονται κάποια παραδείγματα:

α. Αυτόνομος σχεδιασμός και χρονοπρογραμματισμός:

Η NASA σχεδίασε το πρόγραμμα Remote Agent, με το οποίο χρονοπρογραμμάτισε τις λειτουργίες ενός διαστημικού σκάφους. Το πρόγραμμα αυτό, ανέλαβε την ορθή λειτουργία του σκάφους, βρίσκοντας και επιδιορθώνοντάς διάφορες βλάβες, μέσω πλάνων που καθοριζόντουσαν από το έδαφος.

β. Παιχνίδια:

Ο παγκόσμιος πρωταθλητής Garry Kasparov ηττήθηκε από το πρόγραμμα Deep Blue της IBM (έτος 1997), σε έναν μεγάλο αγώνα, με σκορ 3,5 – 2,5. Μετά από τον αγώνα αυτό, ο πρωταθλητής δήλωσε ότι για πρώτη φορά στην απέναντι σκακιέρα υπήρξε ένα «νέο είδος ευφυΐας». Η εταιρία κάταφερε με αυτόν το τρόπο να αυξήσει δραματικά τις μετοχές της (18 δις. Δολάρια).

Εικόνα 3. Σκακιστικός αγώνας μεταξύ του Garry Kasparov και του προγράμματος Deep Blue.

γ. Δημιουργία οχήματος αυτόνομου ελέγχου:

Ένα όχημα NAVLAB, που ανήκε στο πανεπιστήμιο Carnegie Mellon, κατάφερε μέσω του αυτόνομου ελέγχου να διασχίσει μια απόσταση 2850 μιλίων στις Ηνωμένες Πολιτείες Αμερικής. Για την επιτυχία αυτή χρησιμοποιήθηκε το σύστημα ALVINN, το οποίο μέσω της υπολογιστικής όρασης, έπαιρνε εικόνες του δρόμου, από κάμερες του οχήματος, και υπολόγιζε την καλύτερη κατεύθυνση του τιμονιού βάση της εκπαίδευσής του. Το αποτέλεσμα ήταν, το όχημα να έχει αυτόνομο έλεγχο στο 98% της διαδρομής και ένας άνθρωπος – οδηγός να αναλαμβάνει το υπόλοιπο 2% , κυρίως σε ράμπες εξόδου.

δ. Επίτευξη διάγνωσης στην ιατρική:

Σε διάφορους τομείς της ιατρικής επιστήμης τον ρόλο ενός πεπειραμένου ιατρού έχουν αναλάβει τα προγράμματα ιατρικής διάγνωσης, τα οποία αναλύουν πιθανότητες. Ένα παράδειγμα, όπως περιγράφεται από τον Heckerman το 1991,

αποτελεί η διάγνωση από ένα πρόγραμμα για ένα δύσκολο περιστατικό που ανήκει στη παθολογία των λεμφαδένων. Το πρόγραμμα ανέλυσε διάφορα συμπτώματα του περιστατικού και κατέληξε σε ένα συμπέρασμα διάγνωσης τεκμηριώνοντας τις αποφάσεις του, με τον ειδικό ιατρό να συμφωνεί στο τέλος.

ε. Σχεδιασμός εφοδιασμού:

Ένα πρόγραμμα κατασκευασμένο για σχεδιάζει τους εφοδιασμούς και να χρονοπρογραμματίζει τις μεταφορές, χρησιμοποιήθηκε από τις αμερικάνικες δυνάμεις, το 1991, στον πόλεμο του Περσικού Κόλπου. Το πρόγραμμα αυτό με το όνομα DART (Dynamic Analysis and Replanning Tool, Cross and Walker, 1994), χρησιμοποιούσε δυναμική ανάλυση και επασχεδιασμό για πολλές παραμέτρους όπως τον αριθμό των οχημάτων, τα φορτία, τους ανθρώπους, τα σημεία εκκίνησης, τους προορισμούς και τα δρομολόγια. Με την επίλυση όλων αυτών των παραμέτρων, ταυτόχρονα, μπορούσε να παράγει ένα πλάνο μέσα σε λίγες ώρες, για το οποίο θα χρειαζόταν πολλές εβδομάδες διαφορετικά. Αυτή η δυνατότητα αποζημίωσε τα 30 χρόνια έρευνας που αφιέρωσε η αμερικάνικη υπηρεσία DARPA (Defence Advanced Research Project Agency) στην ΤΝ.

στ. Χρήση της ρομποτικής:

Ένα ρομποτικό σύστημα μικροχειρουργικής, με το όνομα Hip Nav, το 1996, είχε την ικανότητα καθοδήγησης της εισαγωγής ενός προσθετικού γοφού . Αυτό, δημιουργούσε ένα τρισδιάστατο μοντέλο της εσωτερικής ανατομίας ενός ασθενή με χρήση τεχνικών υπολογιστικής όρασης.

ζ. Επίλυση προβλημάτων μέσω κατανόησης της γλώσσας:

Το 1999 κατασκευάστηκε ένα πρόγραμμα, με το όνομα PROVERB, το οποίο είχε τη δυνατότητα επίλυσης σταυρολέξων σε πολύ υψηλό επίπεδο. Αυτό χρησιμοποιούσε δίαφορους περιορισμούς συμπλήρωσης λέξεων, προηγούμενες επιλύσεις σταυρολέξων αποθηκευμένες σε μια τεράστια βάση δεδομένων, καθώς και διάφορες πηγές πληροφοριών.

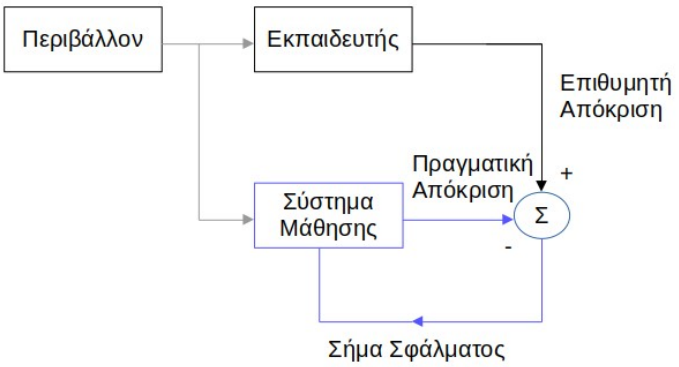
Τα παραπάνω παραδείγματα αποτελούν ένα μικρό δείγμα εφαρμογής της τεχνητής νοημοσύνης σε διάφορους τομείς στο πέρασμα του χρόνου, τα οποία βασίστηκαν στην επιστήμη, τη τεχνολογία και τα μαθηματικά.

2.5 Μηχανική Μάθηση

Τα νευρωνικά δίκτυα έχουν την ικανότητα να μαθαίνουν από το περιβάλλον τους, με τον ίδιο τρόπο που κάνουν και οι άνθρωποι. Οι διαδικασίες μάθησης αυτών μπορούν να κατηγοριοποιηθούν ως εξής: μάθηση με εκπαιδευτή και μάθηση χωρίς εκπαιδευτή. Αντίστοιχα, για την μάθηση χωρίς εκπαιδευτή υφίσταται ο διαχωρισμός της σε δύο κατηγορίες, την επιβλεπόμενη μάθηση και την ενισχυτική μάθηση. Με αυτούς τους τρόπους, η ανθρώπινη μάθηση μπορεί να προσομοιωθεί από τα νευρωνικά δίκτυα.

2.5.1 Μάθηση με Εκπαιδευτή

Μία εναλλακτική ονομασία της μάθησης με εκπαιδευτή είναι η επιβλεπόμενη μάθηση. Στην Εικόνα 4 παρουσιάζεται ένα διάγραμμα που περιγράφει αυτό το είδος μάθησης. Κάνοντας μια ανάλυση του διαγράμματος, θεωρείται ότι ο εκπαιδευτής γνωρίζει το περιβάλλον και τα παραδείγματα εισόδου – εξόδου αντιπροσωπεύουν αυτή τη γνώση. Το νευρωνικό δίκτυο δεν γνωρίζει το περιβάλλον. Σε ένα παράδειγμα, το περιβάλλον μπορεί να αντλήσει ένα διάνυσμα εκπαίδευσης στον εκπαιδευτή και στο νευρωνικό δίκτυο, και λόγω της γνώσης του ο εκπαιδευτής μπορεί να παρέχει μια επιθυμητή απόκριση για το συγκεκριμένο διάνυσμα εκπαίδευσης.



Εικόνα 4. Διάγραμμα μάθησης με εκπαιδευτή. Ο βρόχος ανάδρασης αποτυπώνεται έγχρωμα.

Το νευρωνικό δίκτυο πρέπει να εκτελεί τη «βέλτιστη» ενέργεια, η οποία αντιπροσωπεύεται από την επιθυμητή απόκριση. Το σήμα σφάλματος και το διάνυσμα εκπαίδευσης επηρεάζουν τους παραμέτρους του δικτύου. Η διαφορά μεταξύ της επιθυμητής απόκρισης και της πραγματικής απόκρισης στο δίκτυο, ορίζουν το σήμα σφάλματος. Η προσαρμογή αυτή πρέπει να εκτελείται επαναληπτικά εώς ότου η συμπεριφορά του εκπαιδευτή να μπορεί να προσομοιωθεί από το νευρωνικό δίκτυο. Μέσω μιας στατιστικής έννοιας η προσομοίωση αυτή μπορεί να χαρακτηριστεί ως βέλτιστη. Με αυτό το τρόπο, η γνώση που διαθέτει ο εκπαιδευτής για το περιβάλλον μπορεί να μεταφερθεί στο νευρωνικό δίκτυο μέσω της εκαπαίδευσης. Τελικά, το νευρωνικό δίκτυο αποκτά μια μορφή μακροπρόθεσμης μνήμης, αποθηκεύοντας αυτή τη γνώση με χρήση των σταθερών συναπτικών βαρών και έτσι μπορεί να αφεθεί στο περιβάλλον μόνο του, χωρίς να υπάρχει επίβλεψη από τον εκπαιδευτή.

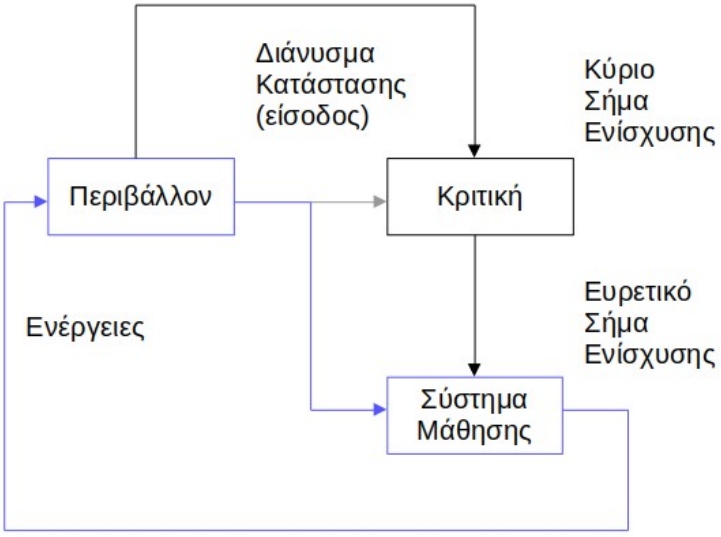
Η περιγραφή που προηγήθηκε για την επιβλεπόμενη μάθηση είναι η βάση της μάθησης μέσω διόρθωσης σφάλματος. Από την Εικόνα 4, φαίνεται ότι ένα σύστημα ανάδρασης κλειστού βρόχου κατασκευάζεται από την επιβλεπόμενη μάθηση, χωρίς όμως να περιέχεται εντός αυτού το άγνωστο περιβάλλον. Το μέσο τετραγωνικό σφάλμα ή το άθροισμα των τετραγώνων των συναπτικών βαρών του δικτύου, χρησιμοποιούνται για τη μέτρηση της απόδοσης του συστήματος.

2.5.2 Μάθηση χωρίς Εκπαιδευτή

Ένας εκπαιδευτής είναι υπεύθυνος να καθοδηγεί την μάθηση στην περίπτωση της επιβλεπόμενης μάθησης. Αντιθέτως, στη περίπτωση της μάθησης χωρίς εκπαιδευτή δεν χρησιμοποιείται κάποιος εκπαιδευτής. Το νευρωνικό δίκτυο στη μάθηση χωρίς εκπαιδευτή δεν έχει κάποια παραδείγματα που χρειάζεται να μάθει. Στη περίπτωση αυτή, μπορούν να ορισθούν δύο υποκατηγορίες μάθησης:

α. Ενισχυτική Μάθηση

Η διαρκής αλληλεπίδραση με το περιβάλλον, έτσι ώστε να ελαχιστοποιηθεί ένας βαθμός δείκτη απόδοσης, αποτελεί τον τρόπο εκμάθησης μιας αντιστοίχισης εισόδου-εξόδου στην ενισχυτική μάθηση. Η Εικόνα 5 παρουσιάζει ένα διάγραμμα, στο οποίο ένα σύστημα ενισχυτικής μάθησης λειτουργεί με έναν μηχανισμό που έχει το ρόλο του κριτή. Ο κριτής αυτός, λαμβάνει το κύριο σήμα από το περιβάλλον και το μετατρέπει σε ευρετικό σήμα ενίσχυσης, του οποίου η ποιότητα είναι πολύ καλύτερη. Ο σχεδιασμός αυτός έχει ως αποτέλεσμα το σύστημα να αποκτά γνώση καθυστερημένα, αφού έχει παρατηρήσει αρκετά ερεθίσματα, τα οποία καταλήγουν να παράξουν το ευρετικό σήμα ενίσχυσης. Η ενισχυτική μάθηση στοχεύει στην ελαχιστοποίηση της συνάρτησης του τρέχοντος κόστους.



Εικόνα 5. Σχηματικό διάγραμμα της ενισχυτικής μάθησης. Ο βρόχος ανάδρασης περιλαμβάνει το σύστημα μάθησης και το περιβάλλον.

Από τις διάφορες ενέργειες που εκτελούνται σε διάφορα χρονικά βήματα κάποιες αποτελούν τις καλύτερες για την συμπεριφορά του συστήματος. Η ανακάλυψη αυτών των ενεργειών και η τροφοδότησή τους αναδρομικά στο περιβάλλον, αποτελεί τη λειτουργία του συστήματος.

Δύο είναι οι βασικοί λόγοι που καθιστούν δύσκολη στην εκτέλεση την ενισχυτική μάθηση με καθυστέρηση:

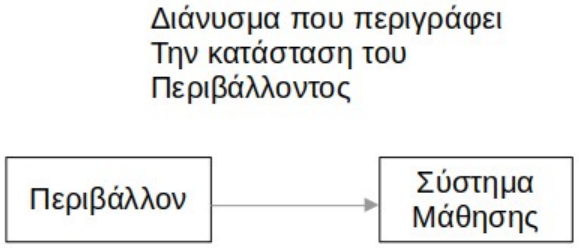
- Σε όλα τα βήματα κατά τη διάρκεια της μάθησης, η απουσία εκπαιδευτή αποκλείει την παροχή επιθυμητών αποκρίσεων.

- Η μηχανή που επιδέχεται την μάθηση οφείλει να επιλύσει ένα χρονικό πρόβλημα ανάθεσης εμπιστοσύνης, λόγω του ότι το κύριο σήμα ενίσχυσης παράγεται με μία καθυστέρηση. Αυτό υπονοεί ότι η μηχανή πρέπει να μπορεί να καθορίσει ένα βαθμό επιτυχίας για κάθε ενέργεια στα διάφορα βήματα που οδηγούν στο τελικό αποτέλεσμα.

Η ενισχυτική μάθηση με καθυστέρηση αποτελεί μια θελκτική μέθοδο, καθώς δίνει τη δυνατότητα επικοινωνίας με το περιβάλλον, στο σύστημα μάθησης, και μέσω αυτής την δυνατότητα εκτέλεσης προκαθορισμένων εργασιών που βασίζονται στην εμπειρία που αποκομίζει από την αλληλεπίδραση με το περιβάλλον.

β. Μη Επιβλεπόμενη Μάθηση

Στην Εικόνα 6 παρουσιάζεται ένα σχηματικό διάγραμμα της μη επιβλεπόμενης μάθησης (λέγεται και αυτο – οργανούμενη μάθηση). Σε αυτή τη περίπτωση η επίβλεψη από εξωτερικό εκπαιδευτή ή από κάποιον κριτή δεν υφίσταται, αλλά υπάρχει ένα μέτρο της ποιότητας αναπαράστασης, το οποίο ανεξαρτητοποιείται από την εργασία και το δίκτυο μαθαίνει από αυτό.Επίσης, οι παράμετροι του δικτύου βελτιώνονται σε σχέση με αυτό. Έτσι, αναπτύσσεται η δυνατότητα σχηματισμού εσωτερικών αναπαραστάσεων που κωδικοποιούν τα χαρακτηρηστικά της εισόδου, και βάσει αυτής δημιουργούνται νέες κλάσεις.



Εικόνα 6. Σχηματικό διάγραμμα μη επιβλεπόμενης μάθησης.

Ένας κανόνας ανταγωνιστικής μάθησης μπορεί να χρησιμοποιηθεί για να εκτελεστεί η μη επιβλεπόμενη μάθηση. Ένα παράδειγμα αποτελεί η χρήση ενός νευρωνικού δικτύου το οποίο αποτελείται από δύο επίπεδα, ένα εισόδου και ένα ανταγωνιστικό. Τα διαθέσιμα δεδομένα λαμβάνονται από το επίπεδο εισόδου. Οι νευρώνες του ανταγωνιστικού επιπέδου ανταγωνίζονται μεταξύ τους (βάσει ενός κανόνα μάθησης) ώστε να επιτευχθεί η απόκριση στα δεδομένα εισόδου. Με το τρόπο αυτό, ένας νευρώνας με τη καλύτερη συνολικά είσοδο κερδίζει και ενεργοποιείται, αφήνοντας τους υπόλοιπους νευρώνες απενεργοποιημένους.

2.6 Η Βιβλιοθήκη λογισμικού OpenCv

Η OpenCV (Open Source Computer Vision Library) είναι μια ανοιχτού κώδικα βιβλιοθήκη λογισμικού όρασης υπολογιστών και μηχανικής μάθησης, η οποία δημιουργήθηκε για να παρέχει μια κοινή υποδομή για εφαρμογές όρασης υπολογιστών και για την επιτάχυνση της χρήσης της μηχανικής αντίληψης σε εμπορικά προϊόντα. Επειδή είναι ένα προϊόν με άδεια Apache 2, το OpenCV καθιστά εύκολη τη χρήση της από διάφορες επιχειρήσεις, καθώς και για τη τροποποίηση του κώδικα. Η βιβλιοθήκη διαθέτει περισσότερα από 2500 βελτιστοποιημένους αλγορίθμους, που περιλαμβάνουν έναν εκτεταμένο σύνολο τόσο κλασικών όσο και τελευταίας τεχνολογίας αλγορίθμων όρασης υπολογιστών και μηχανικής μάθησης. Με τους αλγορίθμους αυτούς μπορούν να επιτευχθούν ενέργειες όπως ανίχνευση, ανίχνευση και αναγνώριση προσώπων, αναγνώριση αντικειμένων, κατηγοριοποίηση ανθρωπίνων ενεργειών σε βίντεο, παρακολούθηση κινήσεων των καμερών, παρακολούθηση κινούμενων αντικειμένων, εξαγωγή τρισδιάστατων (3D) μοντέλων αντικειμένων, τη δημιουργία τρισδιάστατων (3D) σύννεφων σημείων από στερεοκάμερες, τον εύκολο συνδυασμό εικόνων για τη δημιουργία μιας υψηλής ανάλυσης εικόνας ενός ολόκληρου σκηνικού, την εύρεση παρόμοιων εικόνων από μια βάση δεδομένων εικόνων, την αφαίρεση κόκκινων ματιών από εικόνες που τραβήχτηκαν με φλας, την παρακολούθηση της κίνησης των ματιών, την αναγνώριση σκηνικού και την δημιουργία σημάτων για τοποθέτηση επαυξημένης πραγματικότητας, κ.λπ. Η OpenCV διαθέτει περισσότερους από 47 χιλιάδες χρήστες και εκτιμάται ότι έχει ληφθεί πάνω από 18 εκατομμύρια φορές. Η βιβλιοθήκη χρησιμοποιείται εκτενώς από εταιρείες, ερευνητικά ιδρύματα και κυβερνητικούς φορείς. Πέρα από καλά εδραιωμένες εταιρείες όπως η Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda και Toyota που χρησιμοποιούν τη βιβλιοθήκη, υπάρχουν και άλλες πολλές start-ups εταιρίες, όπως η Applied Minds, η VideoSurf και η Zeitera, που χρησιμοποιούν εκτενώς την OpenCV.

Η χρήση της OpenCv καλύπτει ένα ευρύ φάσμα. Παραδείγματα αποτελούν η συρραφή εικόνων της streetview, ο εντοπισμός εισβολών στο Ισραήλ, ο έλεγχος εξοπλισμού ορυχείων στην Κίνα, ο εντοπισμός ατυχημάτων πνίξης σε πισίνες, ο έλεγχος αεροδρομίων, ο έλεγχος ετικετών προϊόντων σε πολλά εργοστάσια ανά τον κόσμο και άμεση ανίχνευση προσώπων. Επίσης, η OpenCV διαθέτει διεπαφές για γλώσσες προγραμματισμού όπως η C++, η Python, η Java και το MATLAB, καθώς και υποστήριξη για τα λειτουργικά συστήματα Windows, Linux, Android και Mac OS. Η OpenCV επικεντρώνται κυρίως σε εφαρμογές πραγματικού χρόνου και εκμεταλλεύεται τις οδηγίες MMX και SSE όταν είναι διαθέσιμες. Υπάρχει επίσης μια πλήρως λειτουργική διεπαφή CUDA και OpenCL που αναπτύσσεται ενεργά. Υπάρχουν πάνω από 500 αλγόριθμοι και περίπου 10 φορές τόσες λειτουργίες που αποτελούν ή υποστηρίζουν αυτούς τους αλγορίθμους. Η OpenCV γράφτηκε αρχικά σε C++ και διαθέτει μια προτύπως εκφρασμένη διεπαφή που λειτουργεί απροβλημάτιστα με τα STL containers (περιοχές αποθήκευσης της C++ Standard Library).

2.6.1 Ανιχνευτές της βιβλιοθήκης OpenCv

Η βιβλιοθήκη λογισμικού OpenCv, διαθέτει οκτώ ανιχνευτές (trackers), ο καθένας με τη δική του υλοποίηση, οι οποίοι μπορούν να χρησιμοποιηθούν για την αναγνώριση αντικειμένων. Οι ανιχνευτές αυτοί είναι οι ακόλουθοι:

1. **BOOSTING Tracker:** Η υλοποίησή του είναι βασισμένη στον ίδιο αλγόριθμο που χρησιμοποιείται για την ενίσχυση Haar Cascades (AdaBoost), αλλά, όπως και ο Haar Cascades, είναι πάνω από δέκα χρόνια παλιός. Αυτός ο ανιχνευτής είναι αργός και δεν λειτουργεί πολύ καλά. Είναι ενδιαφέρον μόνο για ιστορικούς λόγους και για σύγκριση με άλλους αλγορίθμους (απαιτείται τουλάχιστον η έκδοση OpenCV 3.0.0).

2. **MIL Tracker:** Αυτός ο ανιχνευτής είναι παρόμοιος με τον BOOSTING tracker που περιγράφηκε παραπάνω. Η μεγάλη διαφορά είναι ότι, αντί να εξετάζει μόνο την τρέχουσα θέση του αντικειμένου ως θετικό παράδειγμα, αναζητά σε ένα μικρό περιβάλλον γύρω από την τρέχουσα θέση για να δημιουργήσει αρκετά δυνητικά θετικά παραδείγματα. Ο ανιχνευτής αυτός, προσφέρει καλύτερη ακρίβεια από τον BOOSTING tracker, αλλά δεν επιδεικνύει καλή απόδοση στην αναφορά αποτυχιών (απαιτείται τουλάχιστον η έκδοση OpenCV 3.0.0)

3. **KCF Tracker:** Βασίζεται στα Kernelized Correlation Filters. Είναι ταχύτερος από τον BOOSTING tracker και τον MIL tracker. Είναι παρόμοιος με τον MIL, αλλά δεν διαχειρίζεται καλά τον πλήρη εμπλοκισμό. (απαιτείται τουλάχιστον η έκδοση OpenCV 3.1.0)

4. **CSRT Tracker:** Στον αλγόριθμο του Διακριτικού Φίλτρου Συσχέτισης με Κανάλι και Χωρική Αξιοπιστία (DCF-CSR), χρησιμοποιούμε τον χάρτη χωρικής αξιοπιστίας για τη ρύθμιση της υποστήριξης του φίλτρου στο τμήμα της επιλεγμένης περιοχής από το καρέ για την παρακολούθηση. Αυτό εξασφαλίζει τη διεύρυνση και τοποθέτηση της επιλεγμένης περιοχής και βελτιωμένη παρακολούθηση των μη-ορθογωνίων περιοχών ή αντικειμένων. Χρησιμοποιεί μόνο δύο βασικά χαρακτηριστικά (HoGs και Colornames). Λειτουργεί επίσης σε σχετικά χαμηλότερα καρέ ανά δευτερόλεπτο (fps) (25 fps), αλλά προσφέρει υψηλότερη ακρίβεια στην παρακολούθηση αντικειμένων. Τείνει να είναι πιο ακριβής από τον KCF tracker, αλλά είναι ελαφρώς πιο αργός. (απαιτείται τουλάχιστον η έκδοση OpenCV 3.4.2)

5. **MedianFlow Tracker:** Αυτός ο ανιχνευτής παρακολουθεί το αντικείμενο τόσο προς τα εμπρός όσο και προς τα πίσω στον χρόνο και μετρά τις ασυμφωνίες μεταξύ αυτών των δύο πορειών. Η ελαχιστοποίηση αυτού του σφάλματος μεταξύ του προς τα εμπρός και προς τα πίσω κινήσεων επιτρέπει την αξιόπιστη ανίχνευση αποτυχιών στην παρακολούθηση και την επιλογή αξιόπιστων πορειών σε ακολουθίες βίντεο. Ωστόσο, εάν υπάρχει υπερβολικά μεγάλη αλλαγή στην κίνηση, όπως γρήγορα κινούμενα αντικείμενα ή αντικείμενα που αλλάζουν γρήγορα την εμφάνισή τους, τότε το μοντέλο θα αποτύχει. (απαιτείται τουλάχιστον η έκδοση OpenCV 3.0.0)

6. **TLD Tracker:** Ο TLD tracker αναφέρεται στον όρο "Tracking, Learning, and Detection" (Παρακολούθηση, Μάθηση και Ανίχνευση). Όπως υποδηλώνει το όνομά του, αυτός ο ανιχνευτής διαχωρίζει την μακροχρόνια εργασία παρακολούθησης σε τρία συστατικά: την (σύντομο χρονική) παρακολούθηση, τη μάθηση και την ανίχνευση. Σύμφωνα με το άρθρο του συγγραφέα, "Ο ανιχνευτής ακολουθεί το αντικείμενο από καρέ σε καρέ. Ο ανιχνευτής εντοπίζει όλες τις εμφανίσεις που έχουν παρατηρηθεί μέχρι τώρα και τον διορθώνει εάν απαιτείται."(απαιτείται τουλάχιστον η έκδοση OpenCV 3.0.0)

7. **MOSSE Tracker:** Ο αλγόριθμος Minimum Output Sum of Squared Error (MOSSE) χρησιμοποιεί μια προσαρμοζόμενη συσχέτιση για την παρακολούθηση αντικειμένων, η οποία παράγει σταθερά φίλτρα συσχέτισης όταν αρχικοποιείται χρησιμοποιώντας ένα μόνο καρέ. Ο MOSSE tracker είναι ανθεκτικός σε ποικίλες παραλλαγές στον φωτισμό, την κλίμακα, τη θέση και τις μη-σταθερές παραμορφώσεις. Επιπλέον, ανιχνεύει την εμπόδιση με βάση το λόγο κορυφής προς πλευρικού λοβού, πράγμα που επιτρέπει στον tracker να παύσει και να συνεχίσει από το σημείο όπου σταμάτησε όταν το αντικείμενο επανεμφανίζεται. Ο MOSSE tracker λειτουργεί επίσης σε υψηλότερα καρέ ανά δευτερόλεπτο (fps), φθάνοντας τα 450 fps και ακόμη περισσότερα. Για να προστεθούν στα θετικά του, είναι επίσης πολύ εύκολο στην υλοποίηση, είναι τόσο ακριβής όσο και άλλοι πολύπλοκοι ανιχνευτές και πολύ πιο γρήγορος. Ωστόσο, όσον αφορά την απόδοση, υστερεί σε σχέση με τους ανιχνευτές που βασίζονται σε βαθιά μάθηση (deep learning-based trackers).(απαιτείται τουλάχιστον η έκδοση OpenCV 3.4.1)

8. **GOTURN Tracker:** Από όλους τους αλγορίθμους παρακολούθησης στην κατηγορία των ανιχνευτών, αυτός είναι ο μόνος που βασίζεται σε ένα Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network - CNN). Σύμφωνα με την τεκμηρίωση του OpenCV, γνωρίζουμε ότι είναι "ανθεκτικός στις αλλαγές γωνίας προβολής, στις αλλαγές φωτισμού και στις παραμορφώσεις". Ωστόσο, δεν διαχειρίζεται πολύ καλά την εμπόδιση (occlusion).(απαιτείται τουλάχιστον η έκδοση OpenCV 3.2.0)

Στον Πίνακα 2 παρατίθενται οι ανιχνευτές αντικειμένων της OpenCv και οι εκδόσεις στις οποίες εμφανίζονται.

Tracker	OpenCv 3.0	OpenCv 3.1	OpenCv 3.2	OpenCv 3.3	OpenCv 3.4+
CSRT	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Όχι	Όχι	Ναί
KCF	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Ναί	Ναί	Ναί
BOOSTING	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Ναί	Ναί	Ναί
MIL	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Ναί	Ναί	Ναί
TLD	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Ναί	Ναί	Ναί

MedianFlow	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Ναί	Ναί	Ναί
MOSSE	Δεν υποστηρίζεται	Δεν Μεταγλωτίζεται	Όχι	Όχι	Ναί

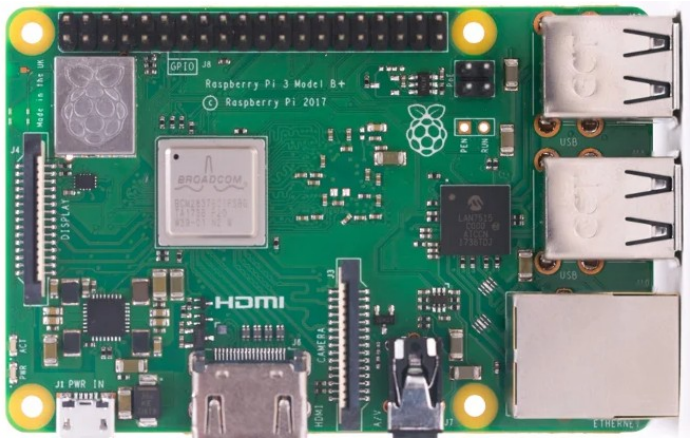
Πίνακας 2. Ανιχνευτές αντικειμένων της OpenCv και οι εκδόσεις που εμφανίζονται.

3 Υλισμικό συστήματος αυτόματης αναγνώρισης αντικειμένων

3.1 Ο μικροϋπολογιστής Raspberry Pi

Το Raspberry Pi είναι το όνομα μιας σειράς υπολογιστών μιας πλακέτας (single-board), που δημιουργήθηκαν από το Ίδρυμα Raspberry Pi, μια βρετανική φιλανθρωπική οργάνωση που στοχεύει στην εκπαίδευση των ανθρώπων στον τομέα της πληροφορικής και στη δημιουργία ευκολότερης πρόσβασης στην εκπαίδευση της πληροφορικής. Η πρώτη κυκλοφορία του Raspberry Pi έγινε το 2012 και από τότε μέχρι σήμερα έχουν κυκλοφορήσει διάφορες εκδόσεις. Το αρχικό Raspberry Pi διέθετε μια μονοπύρηνη CPU στα 700MHz και μόνο 256MB RAM, ενώ η τελευταία εκδοχή διαθέτει μια τετραπύρηνη CPU με ταχύτητα άνω των 1.5GHz και 4GB RAM. Το κόστος για ένα Raspberry Pi πάντα ήταν κάτω από 100 δολάρια (συνήθως γύρω στα 35 δολάρια USD), με το Pi Zero να ξεχωρίζει με κόστος μόλις 5 δολάρια. Η χρήση του Raspberry Pi ανά τον κόσμο αφορά στην εκμάθηση δεξιοτήτων προγραμματισμού, τη δημιουργία έργων υλικού, την υλοποίηση συστημάτων αυτοματισμού για το σπίτι, τη δημιουργία συστημάτων Kubernetes και Edge computing, καθώς και τη χρησιμοποίησή του σε βιομηχανικές εφαρμογές.

Το Raspberry Pi είναι ένα πολύ οικονομικό υπολογιστικό σύστημα που λειτουργεί με το λειτουργικό σύστημα Linux, αλλά παρέχει επίσης ένα σύνολο από ακροδέκτες GPIO (γενικής χρήσης είσοδου/εξόδου), επιτρέποντάς τον έλεγχο ηλεκτρονικών εξαρτημάτων για φυσικό υπολογισμό και την εξερεύνηση του Διαδικτύου των Πραγμάτων (Internet of Things).



Εικόνα 7. Ο μικροϋπολογιστής Raspberry Pi

Υπήρχαν πολλές γενιές στη σειρά του Raspberry Pi: από το Pi 1 έως το 4, και ακόμη ένα Pi 400. Γενικά, υπήρχε μια έκδοση Model A και μια έκδοση Model B για τις περισσότερες γενιές. Το Model A ήταν μια πιο οικονομική παραλλαγή και είχε συνήθως μειωμένη μνήμη RAM και λιγότερες θύρες (όπως USB και Ethernet). Το Pi Zero αποτελεί

μια παραλλαγή της αρχικής γενιάς (Pi 1), και έγινε ακόμη μικρότερο και πιο οικονομικό. Στον Πίνακα 3 παρατίθενται τα μοντέλα Raspberry Pi που έχουν εκδοθεί στο πέρας του χρόνου.

Μοντέλο	Χρονολογία έκδοσης
Pi 1 Model B	2012
Pi 1 Model A	2013
Pi 1 Model B+	2014
Pi 1 Model A+	2014
Pi 2 Model B	2015
Pi Zero	2015
Pi 3 Model B	2016
Pi Zero W	2017
Pi 3 Model B+	2018
Pi 3 Model A+	2019
Pi 4 Model A	2019
Pi 4 Model B	2020
Pi 400	2021

Πίνακας 3. Εκδόσεις του Raspberry Pi

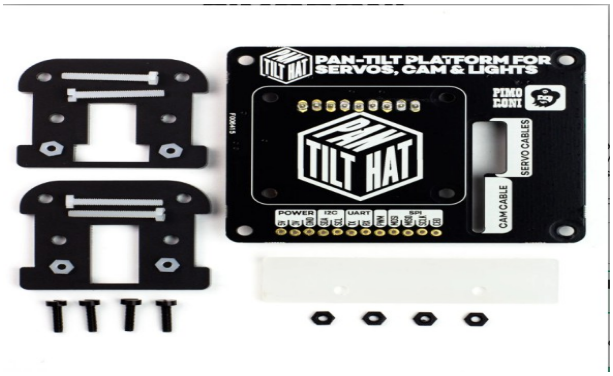
Το Raspberry Pi λειτουργεί στο οικοσύστημα του ανοικτού κώδικα: χρησιμοποιεί το Linux (διάφορες διανομές), και το κύριο υποστηριζόμενο λειτουργικό σύστημά του, το Pi OS, είναι ανοικτού κώδικα και τρέχει ένα σύνολο λογισμικού ανοικτού κώδικα. Το Ίδρυμα Raspberry Pi συνεισφέρει στον πυρήνα του Linux και σε διάφορα άλλα ανοικτού κώδικα έργα, ενώ δημοσιεύει πολύ από το δικό του λογισμικό ως ανοικτού κώδικα. Τα σχηματικά διαγράμματα του Raspberry Pi δημοσιεύονται τακτικά ως τεκμηρίωση, αλλά η πλακέτα δεν είναι ανοικτού υλικού. Το Ίδρυμα Raspberry Pi βασίζεται στα έσοδα από την πώληση των μονάδων Raspberry Pi για το φιλανθρωπικό του έργο στον τομέα της εκπαίδευσης.

3.2 Pimoroni Pan-tilt-hat

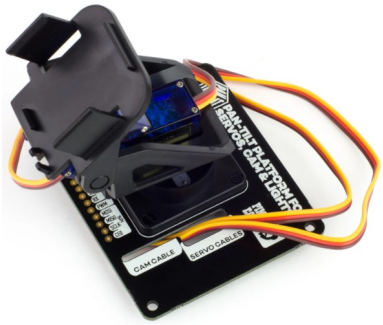
Το Pan-Tilt HAT είναι ένας συνδυασμός υλισμικού και λογισμικού, όπου τοποθετείται πάνω από τον μικροϋπολογιστή Raspberry Pi, δίνοντας τη δυνατότητα να ελεγχθεί η κίνηση μιας κάμερας που θα τοποθετηθεί στους άξονες x και y (pan – tilt). Το HAT και ο ενσωματωμένος μικροελεγκτής, επιτρέπει την οδήγηση των δύο σερβομηχανισμών ανεξάρτητα , καθώς και την οδήγηση έως 24 κανονικών λαμπτήρων LED (με έλεγχο PWM) ή NeoPixel RGB (ή RGBW) LEDs. Υπάρχει επίσης μια χρήσιμη υποδοχή μέσω της οποίας μπορούν να δρομολογηθούν τα καλώδια των σερβομηχανισμών, των λαμπτήρων LED και της κάμερας. Το πρόσθετο αυτό μπορεί να προσανατολίσει το σύστημα πάνω-κάτω και αριστερά-δεξιά κατά 180 μοίρες σε κάθε άξονα.
Το πρόσθετο αυτό έχει τα εξής χαρακτηριστικά:

- Ικανότητα κίνησης 180 μοιρών σε κάθε άξονα χρησιμοποιώντας τους δυο σερβομηχανισμούς.
- Ένα HAT με δύο κανάλια για τους σερβομηχανισμούς, ένα κανάλι PWM ή NeoPixel RGB (ή RGBW) για τους λαμπτήρες LED.
- Προεξοχή μεγάλης γωνίας, που έχει ήδη συγκολληθεί, στο κάτω μέρος του HAT για τα κανάλια των σερβομηχανισμών και των λαμπτήρων LED.
- Εγκοπή για την δρομολόγηση των καλωδίων των σερβομηχανισμών, των λαμπτήρων LED και της κάμερας.
- Ακρυλική βάση για την τοποθέτηση της κάμερας Raspberry Pi v1 ή v2 και τις λωρίδας NeoPixel (με διαχυτή) στη θέση τους.
- Πρόσθετη έξοδο pin (pinout) για σύνδεση.
- Είναι συμβατό με όλα τα μοντέλα Raspberry Pi που διαθέτουν κεφαλή των 40 pin.
- Περιλαμβάνει βιβλιοθήκη λογισμικού της γλώσσας προγραμματισμού Python για εύκολο προγραμματισμό και έλεγχο.

Με τη χρήση του πρόσθετου αυτού η κάμερα Raspberry Pi μπορεί να χρησιμοποιηθεί σε διάφορες εφαρμογές.



Εικόνα 8. Η πλατφόρμα του πρόσθετου Pan Tilt HAT.



Εικόνα 9. Ολοκληρωμένο πρόσθετο Pan Tilt HAT.

3.3 Raspberry Pi Camera

Η κάμερα Raspberry Pi Module 2 έχει έναν αισθητήρα Sony IMX219 8-megapixel και μπορεί να χρησιμοποιηθεί τόσο για την λήψη βίντεο υψηλής ανάλυσης (high definition) όσο και για την λήψη φωτογραφιών. Η κάμερα αυτή μπορεί να υποστηρίξει εικόνα σε πραγματικό χρόνο, σε ανάλυση 1080p στα 30 καρέ ανά δευτερόλετο [frames per second (fps)] είτε σε ανάλυση 720p στα 60 καρέ ανά δευτερόλετο. Είναι προσαρμοσμένη σε μία ταινία καλωδίων των 15 εκατοστών και τοποθετείται στη θύρα CSI του μικρουπολογιστή Raspberry Pi.

Η κάμερα είναι συμβατή με όλα τα μοντέλα του Raspberry Pi 1, 2, 3 και 4. Μπορεί να προσπεράσεται μέσω των διεπαφών MMAL και V4L, και υπάρχουν πολλές βιβλιοθήκες άλλων κατασκευαστών που έχουν δημιουργηθεί για αυτή, συμπεριλαμβανομένης της βιβλιοθήκης Picamera για τη γλώσσα προγραμματισμού Python.

Εικόνα 10. Η κάμερα Raspberry Pi Module 2.

3.4 Σύνδεση υλισμικού

Η σύνδεση όλου του απαιτούμενου υλισμικού, για την ανάπτυξη του συστήματος αυτόματης αναγνώρισης αντικειμένων, είναι πολύ απλή και επιτυγχάνεται με τρία βήματα:

1. Τοποθέτηση καλωδίων των σερβομηχανισμών στη πλακέτα του Pan – Tilt - HAT.

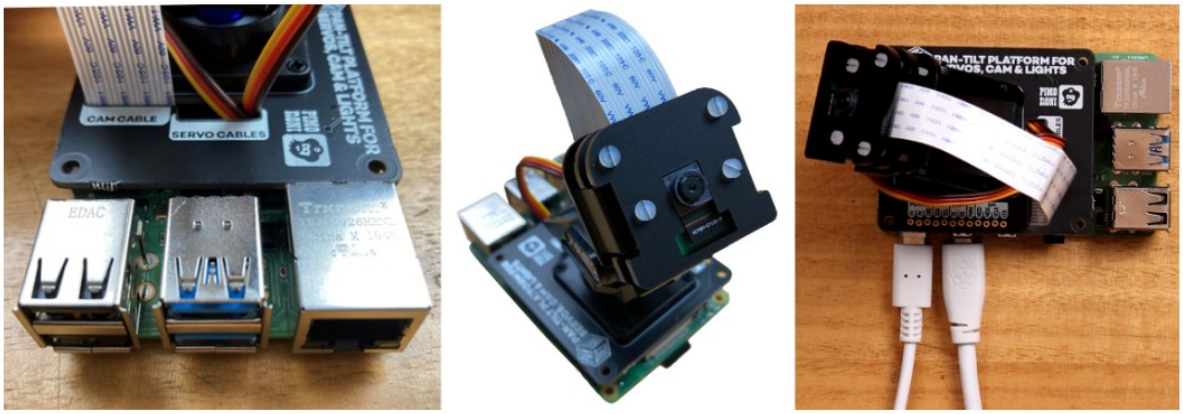
2. Τοποθέτηση της ταινίας καλωδίων της, κάμερας, στη θύρα CSI του Raspberry Pi.

3. Τοποθέτηση της πλακέτας του Pan – Tilt – HAT στη κεφαλή των σαράντα pin του Raspberry Pi.

Η Εικόνα 11 δείχνει τα τρία βήματα σύνδεσης.

Εικόνα 11. Σύνδεση υλισμικού σε τρία βήματα.

Η Εικόνα 12 δείχνει το ολοκλήρωμενο, συνδεδεμένο υλισμικό που χρησιμοποιείται για την ανάπυξη του συστήματος.



Εικόνα 12. Ολοκληρωμένο υλισμικό για το σύστημα αυτόματης αναγνώρισης αντικειμένων.

4 Συστήματα Αυτομάτου Ελέγχου (ΣΑΕ)

4.1 Εισαγωγή στα ΣΑΕ

Τα σύγχρονα συστήματα αυτομάτου ελέγχου περιλαμβάνουν αισθητήρες, μετρητικές διατάξεις, ενεργοποιητές και ένα πλήθος ελεγκτών, οι οποίοι μπορούν να συντονιστούν από τον έλεγχο διεργασιών που βρίσκεται σε ένα υπολογιστικό σύστημα. Ο αριθμός των διεργασιών που εκτελούνται στη μονάδα του χρόνου με την πρόοδο της τεχνολογίας συνεχώς αυξάνεται. Η αύξηση αυτή επιφέρει κρίσιμους μηχανισμούς διεργασιών πραγματικού χρόνου για τις οποίες το λογισμικό των συστημάτων είναι υπεύθυνο ώστε να μην υπάρξει αστοχία στο σύστημα. Χαρακτηριστικό παράδειγμα αποτελούν τα μαχητικά αεροπλάνα που φέρουν ένα σύνολο διατάξεων διεπαφής του συστήματός τους με τον φυσικό κόσμο και λογισμικό ελέγχου, στο οποίο ένα σφάλμα διεργασιών μπορεί να οδηγήσει στην πτώση και την καταστροφή του αεροσκάφους .

Στο πέρασμα των χρόνων και με την εμφάνιση των μικροεπεξεργαστών, όλες οι συσκευές διαθέτουν ελεγκτές με ενσωματωμένα υπολογιστικά συστήματα (embedded systems), τα οποία παρέχουν το πλεονέκτημα της αυξημένης επεξεργαστικής ισχύος, με αποτέλεσμα να εκτε λούνται από τα συστήματα όλο και πιο πολύπλοκες διεργασίες.

Κάθε στοιχείο σε ένα σύστημα μετρήσεων και ελέγχου μετατρέπει ενέργεια από μια μορφή σε άλλη. Προκειμένου λοιπόν να περιγραφεί η απόδοση ολόκληρου του συστήματος μέτρησης και ελέγχου, χρησιμοποιείται η συνάρτηση μεταφοράς. Συνάρτηση μεταφοράς καλείται η σχέση μεταξύ της εισόδου και της εξόδου του συστήματος ελέγχου. Πιο συγκεκριμένα, η συνάρτηση μεταφοράς ορίζεται ως το πηλίκο εξόδου προς την είσοδο.

Κάθε σύστημα ελέγχου διαθέτει τουλάχιστον έναν ελεγκτή και έναν ενργοποιητή. Ο ελεγκτής είναι αυτός που παρέχει τη βελτιστοποίηση στο σύστημα, που συνήθως είναι ένας μικροϋπολογιστής.

Η είσοδος που λαμβάνει ο ελεγκτής καλείται σημείο αναφοράς ή επιθυμητή τιμή (set point). Αυτό το σήμα εισόδου αντιπροσωπεύει την επιθυμητή έξοδο του συστήματος. Για παράδειγμα, επιθυμητή τιμή της θερμοκρασίας του χώρου που ενεργεί ως είσοδο στον ελεγκτή μπορεί να είναι η τιμή των 28°C. Ο ενεργοποιητής (actuator) αποτελεί μια ηλεκτρομηχανική συσκευή η οποία λαμβάνει το σήμα από τον ελεγκτή και το μετατρέπει σε ενέργεια προς τον φυσικό κόσμο. Για το πιο πάνω παράδειγμα ενεργοποιητή αποτελεί ένας ανεμιστήρας ή ένα κλιματιστικό σύστημα. Η διασύνδεση ενός ενεργοποιητή στο σύστημα ελέγχου γίνεται μέσω κατάλλη λης μονάδας διασύνδεσης ισχύος. Η μονάδα ισχύος παρέχει την κατάλληλη ηλεκτρική ενέργεια στον ενεργοποιητή για να μπορέσει να λειτουργήσει. Πολλές φορές η μονάδα ισχύος του συστήματος ενσωματώνει δικό της ελεγκτή, ώστε να παρέχει αδιάλειπτη λειτουργία στο σύστημα καθώς και προστασία από φαινόμενα υπερεντάσεων ή υπερτάσεων. Για τη διασύνδεση των βαθμίδων σε ένα σύστημα απαιτούνται κυκλώματα υποστήριξης, τα οποία μετατρέπουν το επίπεδο των σημάτων ελέγχου σύμφωνα με τις προδιαγραφές εισόδου κάθε επόμενης βαθμίδας.

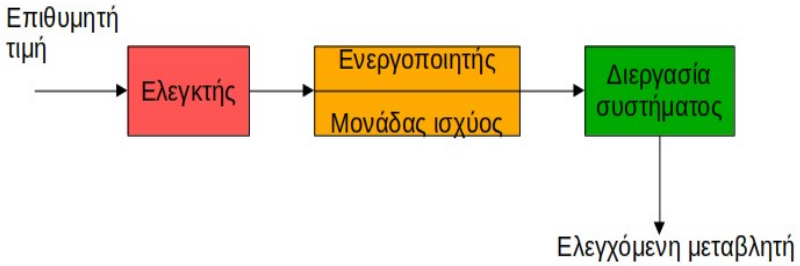
Στην κατηγορία των συστημάτων μετρήσεων και ελέγχου διακρίνουμε δυο είδη:

- συστήματα ελέγχου ανοιχτού βρόχου (open loop control systems).

- συστήματα ελέγχου κλειστού βρόχου (closed loop control systems).

4.2 Συστήματα ελέγχου ανοιχτού βρόχου

Στην Εικόνα 13 παρουσιάζεται ένα διάγραμμα συστήματος ανοιχτού βρόχου. Στο σύστημα αυτό, ένας ελεγκτής μπορεί να ενεργεί χωρίς το σήμα εξόδου να ανατροφοδοτείται (feedback) στην είσοδο. Με το τρόπο αυτό ο ελεγκτής δεν έχει την ικανότητα να επιβεβαιώσει ότι η εντολή που έδωσε στον ενεργοποιητή εκτελέστηκε σωστα. Η επιβεβαίωση παρέχεται από τη γνώση για τη φύση του συστήματος από τον ίδιο τον ελεγκτή. Τα ρελέ και οι βηματικοί κινητήρες (stepper motors) είναι συσκευές που διαθέτουν αξιόπιστα χαρακτηριστικά και συνήθως αποτελούν τμήματα συστημάτων ανοιχτού βρόχου. Συνεπώς, για να ενεργεί ορθά ένα σύστημα ανοικτού βρόχου θα πρέπει ο ελεγκτής να γνωρίζει τον τρόπο λειτουργίας του ενεργοποιητή και του συστήματος ώστε να ενεργεί κατάλληλα.

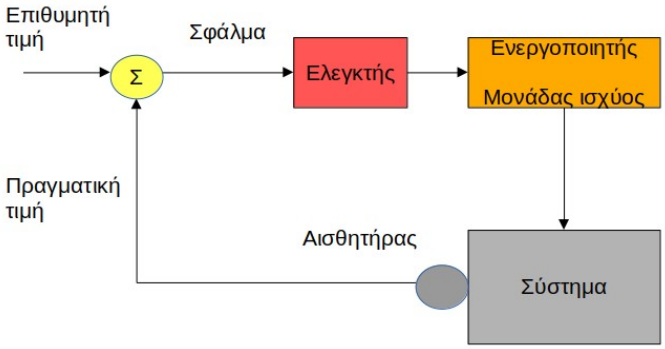


Εικόνα 13. Διάγραμμα συστήματος ανοιχτού βρόχου.

Ένα παράδειγμα συστήματος ανοιχτού βρόχου αποτελεί ο έλεγχος ενός βηματικού κινητήρα. Το σύστημα δεν είναι εφοδιασμένο με αισθητήρα θέσης, επειδή μέσω του ανοιχτού βρόχου η θέση μετακίνησης του ρότορα απο το σημείο αναφοράς είναι προκαθορισμένη μέσω ηλεκτρικών παλμών που αποστέλλει η μονάδα του ελεγκτή.

4.3 Συστήματα ελέγχου κλειστού βρόχου

Σε ένα σύστημα ελέγχου κλειστού βρόχου, το αποτέλεσμα της επεξεργασίας (ελεγχόμενη μεταβλητή) παρακολουθείται μονίμως από έναν ή και περισσότερους αισθητήρες, όπως απεικονίζεται στην Εικόνα 14.



Εικόνα 14. Διάγραμμα συστήματος κλειστού βρόχου.

Ο αισθητήρας λαμβάνει δείγματα από την τιμή της ελεγχόμενης μεταβλητής και μετατρέπει τις μετρήσεις αυτές σε ηλεκτρικά σήματα προς τον ελεγκτή (πραγματική τιμή). Με τον τρόπο αυτό, ο ελεγκτής γνωρίζει την κατάσταση του συστήματος. Έτσι, μπορεί να πραγματοποιεί τις κατάλληλες ρυθμίσεις προκειμένου να διατηρήσει την έξοδο του συστήματος στην επιθυμητή τιμή. Με την αφαίρεση που πραγματοποιεί ο συγκριτής (Σ), της πραγματικής τιμής από την επιθυμητή τιμή, λαμβάνεται το σφάλμα που έχει το σύστημα. Ο σκοπός του ελεγκτή είναι να μηδενίσει το σφάλμα, πράγμα που σημαίνει ότι η έξοδος του συστήματος είναι ίδια με την επιθυμητή τιμή.

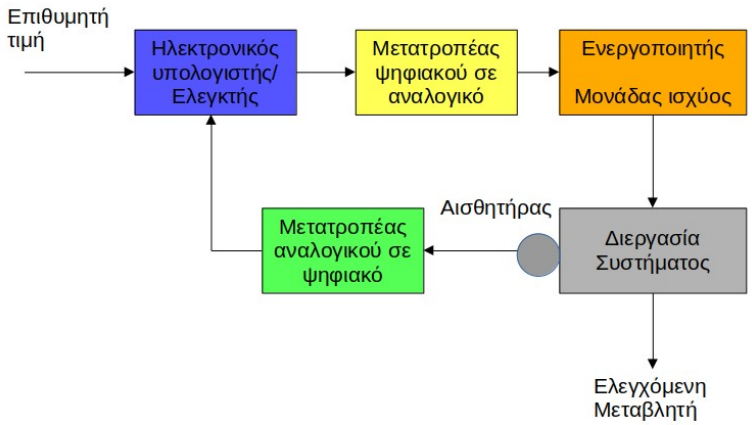
Ένα παράδειγμα συστήματος κλειστού βρόχου αποτελεί ο έλεγχος ενός σερβομηχανισμού. Ο σερβομηχανισμός είναι ένας όρος για την περιγραφή ενός ηλεκτρομηχανικού συστήματος ελέγχου κλειστού βρόχου που κατευθύνει την ακριβή κίνηση ενός αντικειμένου (π.χ. ένα ρομποτικό χέρι). Πιο συγκεκριμένα, ελέγχεται η τελική θέση όπως επίσης και η τελική ταχύτητα του αντικειμένου.

Η μέθοδος ελέγχου όπως: PID, on/ off, Fuzzy, κ.α., που εφαρμόζει ο ελεγκτής προκειμένου να μηδενίσει το σφάλμα, εξαρτάται από τη φύση του συστήματος και των παραμέτρων ελέγχου σε αυτό. Για παράδειγμα, διαφορετικός τύπος ελεγκτή θα χρησιμοποιηθεί στην περίπτωση ελέγχου της θερμοκρασίας σε έναν κλίβανο, από ότι στον έλεγχο θέσης ενός αντικειμέννου και διαφορετική μέθοδος στον έλεγχο φαινομένων υψηλών συχνοτήτων, όπως στην περίπτωση μιας μηχανής τζετ. Πολλές φορές απαιτείται εφαρμογή δύο ή και περισσοτέρων μεθόδων ελέγχου σε ένα σύστημα προκειμένου αυτό να ανταπεξέλθει στις προδιαγραφές της σχεδίασής του.

4.4 Ψηφιακά συστήματα ελέγχου

Καθώς η τεχνολογία των υπολογιστικών συστημάτων προοδεύει με ταχύ ρυθμό, η χρήση τους ως μονάδα ελέγχου ενσωματώνεται όλο και περισσότερα στους σύγχρονους ελεγκτές, οδηγώντας στην ύπαρξη των ψηφιακών συστημάτων αυτομάτου ελέγχου.

Στα ψηφιακά συστήματα ελέγχου, ο ελεγκτής χρησιμοποιεί ψηφιακά κυκλώματα. Αυτό σημαίνει ότι τα ψηφιακά συστήματα ελέγχου θα πρέπει πριν την επεξεργασία των δεδομένων να μετατρέψουν το αναλογικό σήμα εισόδου που λαμβάνουν από τους αισθητήρες, σε ψηφιακά δεδομένα μέσω ενός μετατροπέα αναλογικού σε ψηφιακό. Με την είσοδο των ψηφιακών συστημάτων και στον τομέα των αισθητήρων υπάρχουν αισθητήριοι μεταδότες οι οποίοι αποστέλουν απευθείας ψηφιακά δεδομένα στον ελεγκτή του συστήματος, δίχως αυτός να χρειαστεί να τα μετατρέψει. Ομοίως, στην περίπτωση που το σύστημα φέρει αναλογικό ενεργοποιητή, το ψηφιακό αποτέλεσμα της εξόδου του συστήματος θα πρέπει να μετατραπεί πάλι σε αναλογική μορφή μέσω ενός μετατροπέα ψηφιακού σε αναλογικό και να ενισχυθεί από μια ηλεκτρονική μονάδα ισχύος. Στα ψηφιακά συστήματα, λόγω του διακριτού χρόνου επεξεργασίας, το ψηφιακό κύκλωμα εξετάζει τις εισόδους σε κάθε επανάληψη του προγράμματος σε συγκεκριμένες χρονικές στιγμές και παράγει έπειτα την ανανεωμένη νέα (updated) τιμή της εξόδου. Η Εικόνα 15 παρουσιάζει το διάγραμμα ενός ψηφιακού συστήματος κλειστού βρόχου. Στο διάγραμμα δομικών τμημάτων παρατηρείται η ύπαρξη των μονάδων μετατροπής αναλογικού / ψηφιακού και το αντίστροφο, για την επικοινωνία των μονάδων με τον ηλεκτρονικό υπολογιστή.



Εικόνα 15. Μονάδες μετατροπής αναλογικών και ψηφιακών δεδομένων.

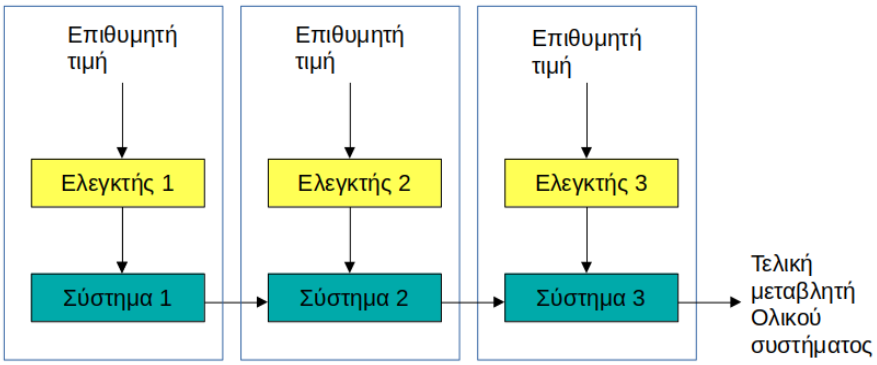
Η διαδικασία ελέγχου σε ένα κλειστό σύστημα αναφέρεται στη διατήρηση του σήματος εξόδου στην επιθυμητή τιμή. Ο τρόπος με τον οποίο ο ελεγκτής το επιτυγχάνει, είναι εξετάζοντας και ρυθμίζοντας την ελεγχόμενη μεταβλητή, προκειμένου να διασφαλίσει το σωστό αποτέλεσμα στην έξοδο του συστήματος. Κλασικό παράδειγμα διαδικασίας ελέγχου αποτελεί ο φούρνος, ο οποίος περιέχει ένα σύστημα κλειστού βρόχου το οποίο διατηρεί σταθερή τη θερμοκρασία στο εσωτερικό του.

Σε αυτή τη περίπτωση, ο ενεργοποιητής είναι το εξάρτημα παραγωγής θερμότητας (π.χ. ένας ηλεκτρικός αντιστάτης), ελεγχόμενη μεταβλητή είναι η θερμοκρασία στο εσωτερικό του φούρνου, και ο αισθητήρας είναι ένα θερμοζεύγος thermocouple) που μετατρέπει τη θερμοκρασία σε ηλεκτρικό δυναμικό διαφορικής τάσης. Ο ελεγκτής ρυθμίζει την ισχύ της ηλεκτρικής αντίστασης με τρόπο ώστε να διατηρεί τη θερμοκρασία του φούρνου στην τιμή που έχει οριστεί από το σημείο αναφοράς ή την επιθυμητή τιμή.

Η διαδικασία ελέγχου μπορεί να γίνει:

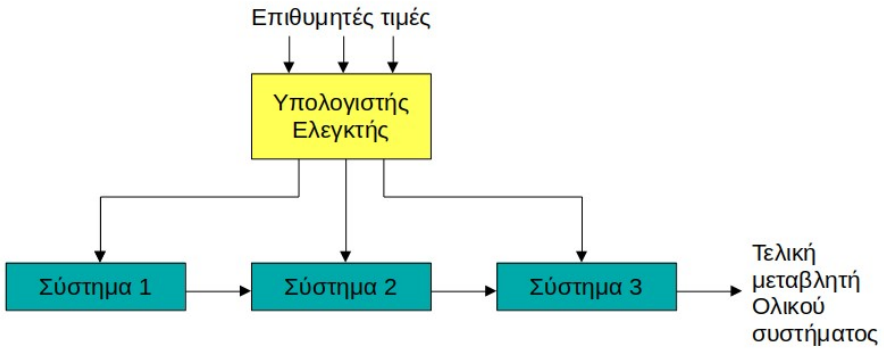
- κατά τμήματα (μη συνεχής (batch)), και
- συνεχόμενα.

Στην περίπτωση της συνεχούς διαδικασίας υπάρχει μια συνεχής παραγωγή αποτελεσμάτων εξόδου. Στην μη συνεχή διαδικασία υπάρχει αρχή και τέλος με εσωτερικές επαναλήψεις. Παράδειγμα μη συνεχούς διαδικασίας αποτελεί η τακτική επικοινωνία σήματος κινητού τηλεφώνου και σταθμού. Συνήθως, ένα σύστημα αποτελείται από ένα σύνολο υποσυστημάτων τα οποία θα πρέπει να ελέγχονται από ένα σύστημα ελεγκτών κάτω από έναν ενιαίο συγχρονισμό. Στο παρελθόν γινόταν χρήσση ανεξάρτητων ελεγκτών για κάθε διαδικασία (Εικόνα 16). Στις περιπτώσεις αυτές κατά την αλλαγή διεργασιών σε μια γραμμή παραγωγής για την οποία ένας ελεγκτής δεν ανταποκρινόταν στις προδιαγραφές του συστήματος, θα έπρεπε να αλλάζεται ή να επαναρυθμιζόταν χειροκίνητα.



Εικόνα 16. Σύστημα ελέγχου απομονωμένων ελεγκτών.

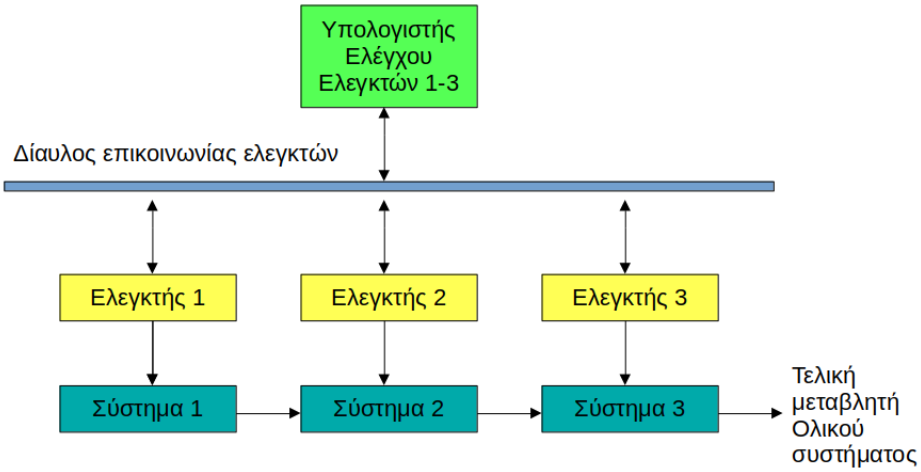
Η λύση του προβλήματος καλείται άμεσος ψηφιακός έλεγχος (Direct Digital Control, DDC). Στον άμεσο ψηφιακό έλεγχο όλοι οι ανεξάρτητοι ελεγκτές αντικαταστήθηκαν από έναν κύριο μεγάλο υπολογιστή (Εικόνα 17) κάτω από ένα ενιαίο σύστημα. Αυτό δίνει τη δυνατότητα όλες οι διεργασίες να μπορούν να ελεχθούν από έναν χρήστη μπροστά από την οθόνη ενός ψηφιακού ελεγκτή ή του υπολογιστή.



Εικόνα 17. Σύστημα άμεσου ψηφιακού ελέγχου.

Μειονέκτημα όμως του άμεσου ψηφιακού ελέγχου αποτελεί η εξάρτηση ολόκληρης της εγκατάστασης από έναν υπολογιστή. Αυτό σημαίνει ότι ο υπολογιστής θα πρέπει να φέρει συστήματα υποστήριξης παροχής αδιάλειπτης λειτουργίας, διότι σε περίπτωση που τεθεί εκτός λειτουργίας, όλο το σύστημα βρίσκεται εκτός διεργασιών ελέγχου. Επίσης, σε περίπτωση συντήρησης του υπολογιστή το σύστημα ελέγχου θα έπρεπε να σταματήσει.

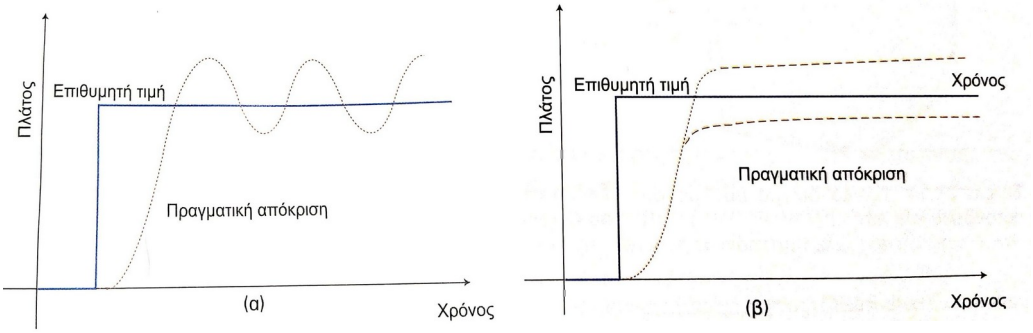
Η λύση του παραπάνω προβλήματος καλείται κατανεμημένος υπολογιστικός έλεγχος (Distributed Computer Control, DCC) (Εικόνα 18). Ένα σύστημα κατανεμημένου ελέγχου διαμοιράζει τις διεργασίες μεταξύ μικρών ελεγκτών που βασίζονται σε μικροεπεξεργαστές (microprocessor-based controllers). Οι μικροί ελεγκτές που βασίζονται σε μικροεπεξεργαστές (microprocessor-based controllers) αποτελούν αυτόνομους ελεγκτές οι οποίοι μέσω διαύλων όπως PROFIBUS και GPIB ελέγχονται και επαναπρογραμματίζονται από έναν και μόνο υπολογιστή, που έχει τον ρόλο του επόπτη. Η επικοινωνία μέσω του διαδικτύου ενός επιμέρους ελεγκτή του συστήματος με τον κύριο υπολογιστή δίνει τη δυνατότητα του απομακρυσμένου ελέγχου και εκτός ζώνης του συστήματος ή της τοπολογίας.



Εικόνα 18. Σύστημα κατανεμημένου ελέγχου.

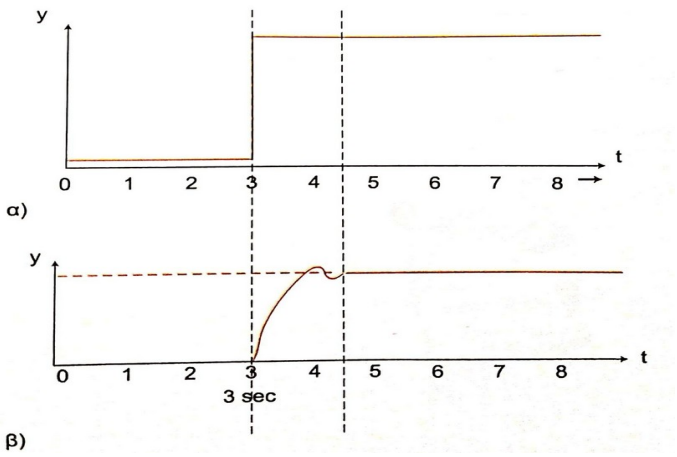
4.5 Ο ελεγκτής PID

Διάφορα συστήματα ελέγχου στη βιομηχανία χρησιμοποιούν των ελεγκτή τριών όρων PID. Η χρήση του ελεγκτή αυτού αποσκοπεί στην ταχεία διόρθωση των παραγόμενων σφαλμάτων καθώς μεταβάλεται η επιθυμητή τιμή του συστήματος, στην ελαχιστοποίηση φαινομένων σφαλμάτων που μεταβάλουν την επιθυμητή τιμή και μπορεί να επιφέρουν αστοχία στον έλεγχο, στην ελαχιστοποίηση φαινομένων που προκαλούν αμείωτες ταλαντώσεις στο σύστημα, μόνιμο σφάλμα κ.α. Όπως φαίνεται στην Εικόνα 19 (α), ο ελεγκτής δεν καταφέρνει να διορθώσει το σφάλμα που προήλθε από τη μεταβολή του σήματος εισόδου στο σύστημα και το σύστημα οδηγείται σε συμπεριφορά αμείωτης ταλάντωσης, ενώ στην Εικόνα 19 (β) το σύστημα παρουσιάζει μόνιμο σφάλμα εξόδου.



Εικόνα 19. Αστοχία ελέγχου που οδηγεί α) σε αμείωτη ταλάντωση και β) σε μόνιμο σφάλμα εξόδου.

Στην Εικόνα 20 (α) παρουσιάζεται η ιδανική απόκριση του συστήματος σε βηματική μεταβολή του σήματος εισόδου με έναν ελεγκτή PID. Όπως παρατηρούμε στο γράφημα της Εικόνας 20 (β), το σύστημα αποκτά την επιθυμητή τιμή μετά από διόρθωση του σφάλματος σε χρόνο 1,5 sec.

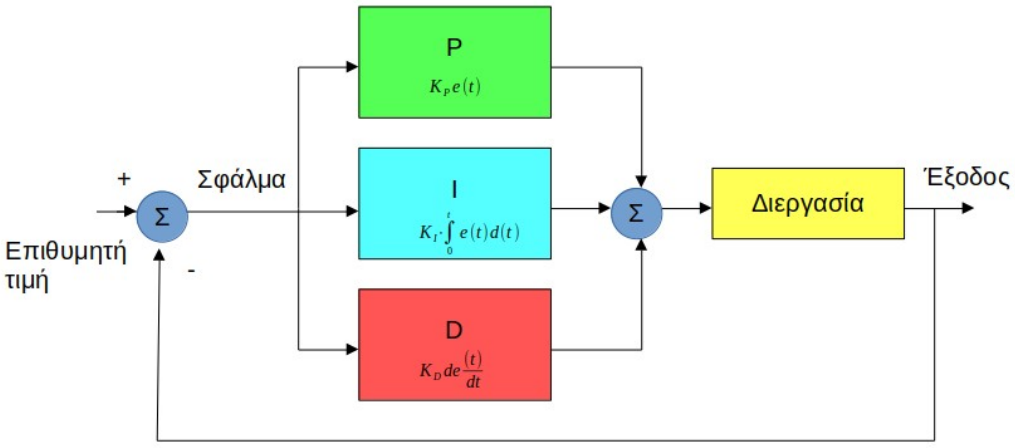


Εικόνα 20. Απόκριση συστήματος με διόρθωση του σφάλματος από τον ελεγκτή PID.

Για να διορθώνεται το παραγόμενο σφάλμα κατά τη διάρκεια μεταβολής της επιθυμητής τιμής του συστήματος, ο ελεγκτής PID παράγει ένα σήμα διόρθωσης που αποτελείται από το άθροισμα ενός αναλογικού (Proportional), ενός ολοκληρωτικού (Integral) και ενός παραγωγικού (Derivative) όρου. Η συνάρτηση μεταφοράς του ελεγκτή στο πεδίο του χρόνου, που παρουσιάζεται στην Εικόνα 21, δίνεται από τη σχέση:

$$u(t) = K_P e + K_I \cdot \int_0^t e\, d(t) + K_D \frac{de}{dt}$$

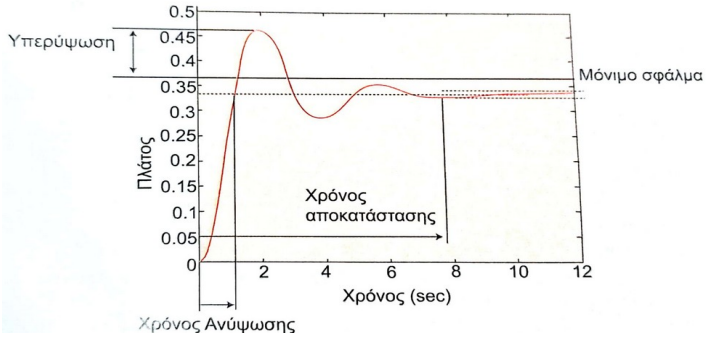
όπου K_P, K_I και K_D, σταθερές παραμέτρων του ελεγκτή.



Εικόνα 21. Διάγραμμα ενός ελεγκτή PID.

Η ρύθμιση των τριών παραμέτρων του ελεγκτή PID επιτυγχάνεται μέσω κανόνων ρύθμισης προσδιορισμού σε βηματικές αποκρίσεις του συστήματος. Οι πιο γνωστοί κανόνες ρύθμισης των παραμέτρων K_P, K_I και K_D ενός ελεγκτή PID καταγράφονται από τους Ziegler και Nichols. Όπως παρουσιάζεται στο γράφημα της Εικόνας 22, το φαινόμενο του χρόνου ανόδου (rise time), του πλάτους υπερύψωσης (overshoot), του χρόνου αποκατάστασης (settling time) και του μόνιμου σφάλματος (error band) που εμφανίζονται κατά την επίδραση του ελεγκτή PID σε σήμα εισόδου μοναδιαίας βηματικής αλλαγής,

μπορούν να περιοριστούν σύμφωνα με τους κανόνες ρύθμισης των παραμέτρων K_P, K_I και K_D.



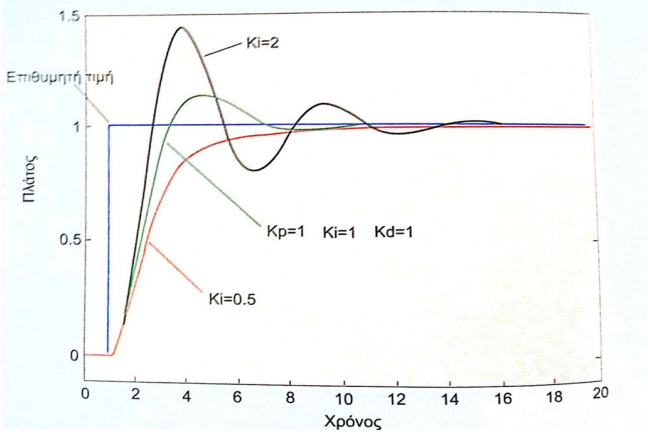
Εικόνα 22. Απόκριση ενός συστήματος ελέγχου PID, χρησιμοποιώντας για είσοδο μια βηματική συνάρτηση.

Στον Πίνακα 4 που ακολουθεί, παρουσιάζεται η επίδραση του κάθε όρου (K_P, K_I και K_D) στη διόρθωση της απόκρισης του ελεγκτή για τον περιορισμό των παραπάνω φαινομένων.

	Χρόνος ανύψωσης	Πλάτος υπερύψωσης	Χρόνος αποκατάστασης	Μόνιμο σφάλμα
K_P	-	+	Ελάχιστη μεταβολή	-
K_I	-	+	+	Σχεδόν μηδενισμός
K_D	Ελάχιστη μεταβολή	-	-	Ελάχιστη μεταβολή

Πίνακας 4. Επίδραση των όρων K_P, K_I και K_D στη διόρθωση της απόκρισης του ελεγκτή.

Στην Εικόνα 23 παρουσιάζεται η επίδραση επιλογής των παραμέτρων ρύθμισης σε έναν ελεγκτή PID με σήμα εισόδου μοναδιαίας βηματικής συνάρτησης. Από το γράφημα φαίνεται ότι ο βέλτιστος έλεγχος επιτυγχάνεται (στην εξεταζόμενη περίπτωση με τις παραμέτρους K_P και K_D σταθερές) όταν η τιμή της παραμέτρου ολοκλήρωσης γίνει $K_I = 1$. Στην περίπτωση που η παράμετρος ολοκλήρωσης έχει τιμή $K_I = 2$ παρατηρούμε στο σήμα εξόδου α) μεγάλη ταλάντωση και β) να μην λαμβάνει την επιθυμητή τιμή (μόνιμο σφάλμα), ενώ στην περίπτωση επιλογής $K_I = 0,5$, η απόκριση θα παρουσιάζει α) μεγάλο χρόνο καθυστέρησης και β) μικρό μόνιμο σφάλμα. Πολλές φορές η επιλογή των παραμέτρων γινεται εμπειρικά από τον μηχανικό.



Εικόνα 23. Απόκριση ενός συστήματος ελέγχου PID, χρησιμοποιώντας για είσοδο μια βηματική συνάρτηση. Ο όρος ολοκλήρωσης (K_I) μεταβάλλεται ενώ οι όροι K_P και K_D είναι σταθεροί.

4.6 Χρήση του ελεγκτή PID στο Σύστημα Αυτόματης Αναγνώρισης Αντικειμένων

Το σύστημα αυτόματης αναγνώρισης αντικειμένων, όπως παρουσιάσθηκε το υλισμικό του στο Κεφάλαιο 3, περιέχει στο πρόσθετο Pan-Tilt-HAT, δύο σερβομηχανισμούς που είναι υπεύθυνοι για την κίνηση της κάμερας σε κάθε κατεύθυνση. Για την ενργοποίηση του συστήματος, εκτελείται πηγαίος κώδικας, ο οποίος περιλαμβάνει και τον ελεγκτή PID. Ο ελεγκτής PID αποτελεί μέρος του λογισμικού του συστήματος και είναι υπεύθυνος να οδηγεί σωστά τους σερβομηχανισμούς ώστε η κάμερα να μην χάνει το αντικείμενο που αναγνωρίζεται. Για την ακρίβεια, το σύστημα χρησιμοποιεί δύο ελεγκτές PID, έναν για την οριζόντια κίνηση (panning) και έναν για την κάθετη (tilting), έτσι ώστε ο καθένας από αυτούς να ελέγχει μια συγκεκριμένη κίνηση. Περισσότερες λεπτομέρειες για το λογισμικό του συστήματος θα γίνουν γνωστές στο Κεφάλαιο 7.

5 Συστήματα μη Επανδρωμένων Αεροσκαφών (ΣμηΕΑ)

5.1 Εισαγωγή στα ΣμηΕΑ

Στο πέρασμα του χρόνου πολλοί όροι χρησιμοποιήθηκαν για να περιγράψουν τα συστήματα των μη επανδρομένων αεροσκαφών. Οι παρακάτω όροι έχουν χρησιμοποιηθεί κατά καιρούς, για να περιγράψουν όλα τα ιπτάμενα μέσα που η άτρακτός τους δεν περιέχει χειριστή και μπορούν να πραγματοποιήσουν πτήσεις με αυτόνομο τρόπο ή με τηλεκατεύθυνση:

- Μη επανδρωμένα αεροχήματα-MEA(unmanned aerial vehicle-UAV)
- Μη επανδρωμένα αεροσυστήματα (unmanned aerial system-UAS)
- Σύστημα αεροπορίας με απομακρυσμένο πιλότο (remotely piloted aircraft system-RPAS ή drones)

Για να περιγραφούν αυτού του είδους οχήματα, στο πέρασμα του χρόνου δόθηκαν οι παραπάνω ονομασίες. Ένα αεροσκάφος που δεν έχει χειριστή περιγράφεται από τον όρο UAV, ενώ με τον όρο UAS περιγράφεται το ολοκληρωμένο σύστημα του μη επανδρωμένου αεροσκάφους, το οποίο αποτελείται από το προσωπικό, τις συσκευές και τις διαδικασίες που χρησιμοποιούνται. Η ανάγκη της νομοθεσίας για την ύπαρξη τουλάχιστον ενός επιβλέποντα πιλότου, που βρίσκεται στο έδαφος, σε κάθε πτήση ενός μη επανδρωμένου αεροσκάφους, καθιέρωσε τον όρο RPAS. Η μορφή των μη επανδρωμένων ιπτάμενων οχημάτων είναι τις περισσότερες φορές αυτή ενός μικρού αεροπλάνου ή ενός ελικοπτέρου, περιλαμβάνοντας έναν ή περισσότερους κινητήρες και έλικες αντίστοιχα. Ο χειρισμός αυτών γίνεται συνήθως από ειδικά προγράμματα ή από χειριστήρια εδάφους.

Ο διαχωρισμός των μη επανδρωμένων αεροσκαφών γίνεται σε δύο κατηγορίες:

1. Χρήση σταθερών πτερύγων, όπως τα αεροπλάνα

2. Χρήση ελίκων, όπως τα ελικόπτερα.

Οι όροι multicopter και multirotor χρησιμοποιούνται για να προσδιορίσουν τον αριθμό των κινητήρων ή των ελίκων αντίστοιχα. Ένα παράδειγμα για τα ελικόπτερα αποτελεί ο διαχωρισμός σε τετρακόπτερα (με τέσσερις έλικες - quadcopters), εξακόπτερα (hexacopters), οκτακόπτερα (octacopters) κλπ.

Στην αεροναυτική, η πιλοτική ενός αεροσκάφους σημαίνει βασικά τον έλεγχο της πτήσης του. Έχει έναν πολύ συγκεκριμένο νόημα που σχετίζεται με τη δυνατότητα να ελέγχεται η στάση του οχήματος ως προς το κέντρο βάρους του. Αν και τα περισσότερα UAVs ελέγχονται απομακρυσμένα, σχεδόν όλα έχουν έναν ενσωματωμένο αυτόματο πιλότο που είναι υπεύθυνος για την πτήση του. Συνεπώς, δεν πρόκειται για ένα αεροσκάφος που ελέγχεται απομακρυσμένα, αλλά μόνο για ένα αεροσκάφος που λειτουργεί

απομακρυσμένα, όπου του αποστέλλονται οδηγίες. Επιπλέον, οι οδηγίες πλοήγησης, όπως σημεία διέλευσης, διαδρομές και αλγόριθμοι απόφασης, μπορεί ακόμη να συμπεριληφθούν στον ενσωματωμένο υπολογιστή για την ολοκλήρωση της αποστολής χωρίς ανθρώπινη επέμβαση. Με αυτό τον τρόπο, η ανθρώπινη κρίση επικεντρώνεται σε ενέργειες που αφορούν υψηλότερα επίπεδα, όπως η λήψη αποφάσεων ή ο καθορισμός στρατηγικής. Ο όρος «σύστημα αεροσκάφους που λειτουργεί απομακρυσμένα» (ROAS) θα είχε, επομένως, περισσότερο νόημα για τη σύγχρονη επιστημονική κοινότητα.

Εικόνα 24. Ένα μαχητικό μη επανδρωμένο αεροσκάφος.

5.2 Ο αυτόματος πιλότος Pixhawk Cube

Το Pixhawk Cube είναι ένα δημοφιλές και ευέλικτο σύστημα αυτόνομου ελέγχου που σχεδιάστηκε για μη επανδρωμένα αεροσκάφη (UAVs) και άλλα αυτόνομα οχήματα. Είναι γνωστό τόσο για την αξιοπιστία του, όσο και για την φύση του ως ανοιχτού κώδικα, καθιστώντας έτσι, τη προτιμόμενη επιλογή για ερασιτέχνες, ερευνητές και επαγγελματίες στον τομέα της ρομποτικής και των UAVs.

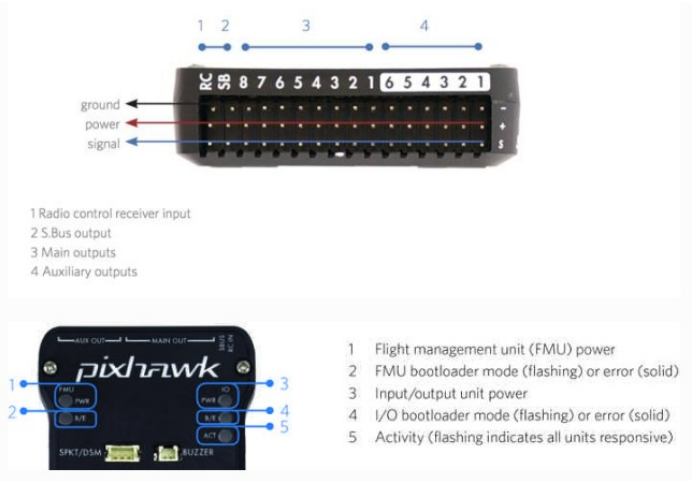
Στον πυρήνα του Pixhawk Cube βρίσκεται ένας ελεγκτής πτήσης που εκτελεί το ανοικτού κώδικα λογισμικό αυτόνομου ελέγχου ArduPilot. Αυτό το λογισμικό παρέχει μια ευρεία γκάμα χαρακτηριστικών και δυνατοτήτων, συμπεριλαμβανομένων της πλοήγησης με σημεία, της σταθεροποίησης, των λειτουργιών ασφαλείας σε περίπτωση αποτυχίας και της υποστήριξης διάφορων αισθητήρων και περιφερειακών. Το Cube παρουσιάζεται σε διάφορα μοντέλα, όπως το Pixhawk 2.1 και το Pixhawk 2.1 "Mini", τα οποία προσαρμόζονται σε συγκεκριμένες ανάγκες και τύπους οχημάτων.

Ένα από τα μεγαλύτερα πλεονεκτήματα του Pixhawk Cube αποτελεί η ενδυνάμωση και η δυνατότητα επέκτασης. Οι χρήστες μπορούν εύκολα να συνδέσουν επιπλέον αισθητήρες και ενότητες για να ενισχύσουν τη λειτουργικότητα των αυτόνομων οχημάτων τους. Η συμβατότητά του με διάφορα πρωτόκολλα επικοινωνίας και διεπαφές, όπως UART, I2C και CAN, επιτρέπει την ολοκλήρωση με μια ευρεία γκάμα αισθητήρων, καμερών και άλλων αντικειμένων. Αυτή η ευελιξία καθιστά το Pixhawk Cube μια δημοφιλή επιλογή για διάφορες εφαρμογές, από αεροφωτογραφία και γεωγραφικές μετρήσεις μέχρι αγώνες UAVs και ερευνητικά έργα. Έχει διαδραματίσει σημαντικό ρόλο στην προώθηση των δυνατοτήτων και της προσβασιμότητας των αυτόνομων συστημάτων για διάφορους χρήστες.

Στις Εικόνες 25 (α), (β) και (γ) παρουσιάζεται το σύστημα αυτόματου ελέγχου Pixhawk cube, καθώς και οι θύρες που έχει διαθέσιμες για σύνδεση διαφόρων αντικειμένων.

Εικόνα 25 (α). το σύστημα αυτόματου ελέγχου Pixhawk cube και οι διαθέσιμες θύρες.

Εικόνα 25 (β). το σύστημα αυτόματου ελέγχου Pixhawk cube και οι διαθέσιμες θύρες.



Εικόνα 25 (γ). το σύστημα αυτόματου ελέγχου Pixhawk cube και οι διαθέσιμες θύρες.

5.3 Ο προσομοιωτής SITL(Ardupilot Simulator)

Το Software-in-the-Loop (SITL) αποτελεί ένα κρίσιμο συστατικό του οικοσυστήματος του ArduPilot και λειτουργεί ως ένα ισχυρό προσομοιωτή που επιτρέπει στους προγραμματιστές να δοκιμάσουν και να αναπτύξουν λογισμικό αυτόνομων οχημάτων σε ένα εικονικό περιβάλλον. Αυτό το εργαλείο διαδραματίζει έναν καίριο ρόλο στην ανάπτυξη, τον έλεγχο και τη λεπτομέρεια των αυτόνομων συστημάτων, ιδιαίτερα για τα μη επανδρωμένα αεροσκάφη (drones) και άλλα ρομποτικά οχήματα.

Ένα από τα κύρια χαρακτηριστικά του SITL είναι η δυνατότητά του να αντιγράφει τη συμπεριφορά του υλικού και των αισθητήρων του πραγματικού κόσμου, προσφέροντας ένα ρεαλιστικό περιβάλλον δοκιμής για το λογισμικό αυτόνομης οδήγησης του ArduPilot. Με τον τρόπο αυτό, επιτρέπει στους χρήστες να αξιολογούν πώς αντιδρά το λογισμικό του οχήματος σε διάφορες σενάρια και εισόδους αισθητήρων. Αυτή η δυνατότητα είναι ανεκτίμητη για την επίλυση σφαλμάτων στον κώδικα, τη βελτιστοποίηση των αλγορίθμων ελέγχου και τον έλεγχο της απόδοσης νέων χαρακτηριστικών, χωρίς τα πραγματικό υλικό ή τα οχήματα να τίθενται σε κίνδυνο καταστροφής ή στη δημιουργία βλαβών.

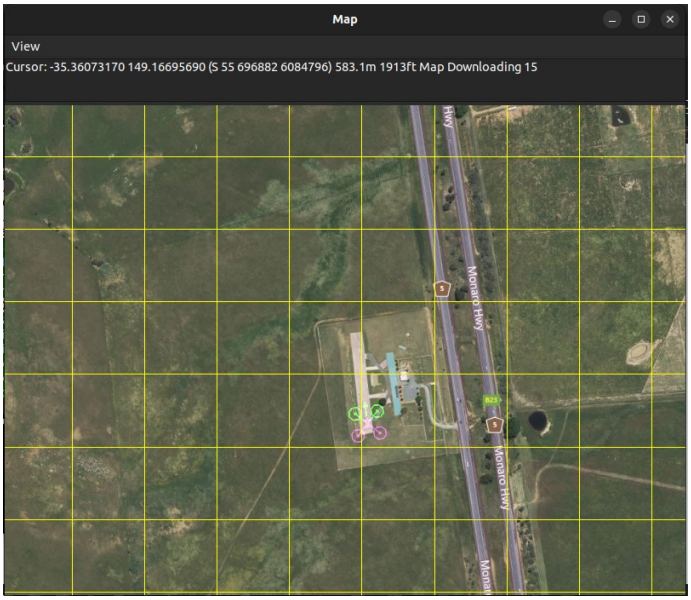
Ο προσομοιωτής μπορεί να προσομοιώσει τα παρακάτω:

- Πολυκόπτερα αεροσκάφη
- Σταθερά φτερωτά αεροσκάφη
- Επίγεια οχήματα
- Υποβρύχια οχήματα
- Σταθεροποιητές κάμερας
- Κατευθυντικοί παρακολουθητές κεραίας
- Μια μεγάλη ποικιλία προαιρετικών αισθητήρων, όπως τα Lidars και οπτικούς αισθητήρες ροής

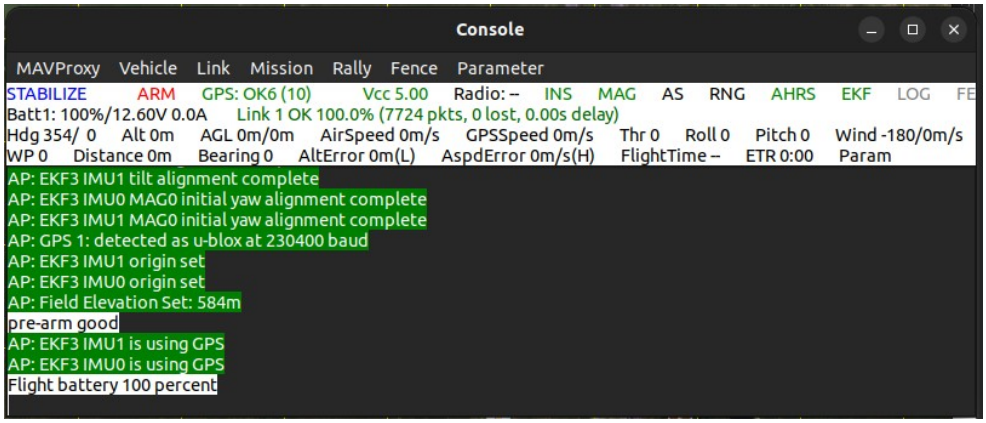
Η ευελιξία και η επεκτεινόμενη χρησιμότητα αυτού του προσομοιωτή τον καθιστούν ένα αναντικατάστατο εργαλείο για τους προγραμματιστές του ArduPilot, καθώς μπορεί να ενσωματωθεί σε διάφορα περιβάλλοντα ανάπτυξης και να χρησιμοποιηθεί συνδυαστικά με άλλες πλατφόρμες προσομοίωσης για τη βελτίωση των δοκιμών και την ανάπτυξη αυτόνομων συστημάτων.

Στη παρούσα εργασία διενεργήθηκε η προσομοίωση ενός τετρακόπτερου μη επανδρωμένου αεροσκάφους όπου ο προσομοιωτής SITL εκτελέσθηκε σε λειτουργικό σύστημα Linux.

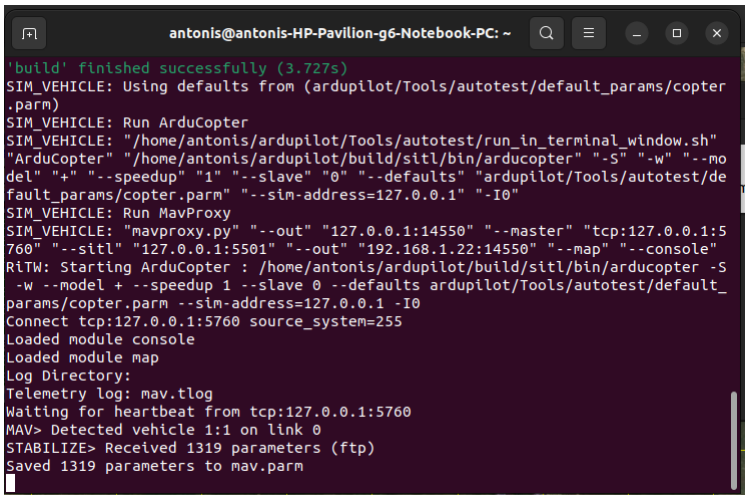
Στις Εικόνες 26,27 και 28 παρατηρείται η προσομοίωση του τετρακόπτερου, ο επίγειος σταθμός ελέγχου (Ground Control Station - GCS) και το τερματικό γραμμής εντολών του αντίστοιχα.



Εικόνα 26. Ο προσομοιωτής ενός τετρακόπτερου (ArduCopter).



Εικόνα 27. Ο επίγειος σταθμός ελέγχου (GCS).



Εικόνα 28. Το τερματικό γραμμής εντολών του τετρακόπτερου.

5.4 Το πρωτόκολλο επικοινωνίας MavLink

Το πρωτόκολλο επικοινωνίας MavLink είναι ένα ανοικτού κώδικα πρωτόκολλο που έχει σχεδιαστεί για την ανταλλαγή δεδομένων μεταξύ αυτόνομων οχημάτων, όπως μη επανδρωμένων αεροσκαφών (drones) και ρομποτικά αυτοκίνητα. Αναπτύχθηκε αρχικά από την ανοικτή κοινότητα του ArduPilot και του PX4, και έχει ανελιχθεί σε ένα από τα πιο δημοφιλή πρωτόκολλα επικοινωνίας για αυτόνομα οχήματα. Το MavLink χρησιμοποιεί μια απλή, ελαφριά δομή μηνυμάτων, τα οποία είναι ευκολά συμπιεσμένα, ώστε να καθιστά εύκολη την επικοινωνία και να δίνει τη δυνατότητα χρήσης σε συσκευές με περιορισμένους πόρους. Έχει υποστήριξη για μια εκτεταμένη γκάμα μηνυμάτων, συμπεριλαμβανομένων πληροφοριών τροχιάς, κατάστασης μπαταρίας, δεδομένων αισθητήρων και εντολών ελέγχου.

Επίσης, είναι ανεξάρτητο από την πλατφόρμα και το λειτουργικό σύστημα, κάτι που το καθιστά ιδιαίτερα ευέλικτο και κατάλληλο για εφαρμογές σε διάφορες πλατφόρμες και περιβάλλοντα. Αυτό το πρωτόκολλο επικοινωνίας έχει συμβάλει σημαντικά στην ανάπτυξη και την εξέλιξη της τεχνολογίας αυτόνομων οχημάτων, καθώς επιτρέπει την αξιοποίηση των πλεονεκτήμάτων της ανοικτής πηγής κώδικα και την εύκολη επέκταση και προσαρμογή του σε διάφορες ανάγκες εφαρμογών.

Το MavLink προσφέρει υποστήριξη για πολλές γλώσσες προγραμματισμού, καθιστώντας το προσβάσιμο και ευέλικτο σε προγραμματιστές από διάφορες κοινότητες. Οι γλώσσες προγραμματισμού που υποστηρίζονται περιλαμβάνουν την C/C++, Python, Java, JavaScript, Swift, και πολλές άλλες. Αυτή η ποικιλία επιλογών επιτρέπει στους προγραμματιστές να εργαστούν στο πλαίσιο που τους είναι πιο άνετο και να αναπτύξουν εφαρμογές που εξυπηρετούν τις ανάγκες τους.

Στη παρούσα εργασία, αυτό το πρωτόκολλο επικοινωνίας χρησιμοποιείται μέσω της γλώσσας προγραμματισμού Python και της βιβλιοθήκης λογισμικού pymavlink, και ο σκοπός της χρήσης του είναι να μεταδωθούν μηνύματα από το Raspberry Pi, που εκτελεί το σενάριο της αυτόματης αναγνώρισης αντικειμένων, στον προσομοιωτή SITL.

6 Διεργασίες και συγχρονισμός διεργασιών

6.1 Η έννοια της διεργασίας

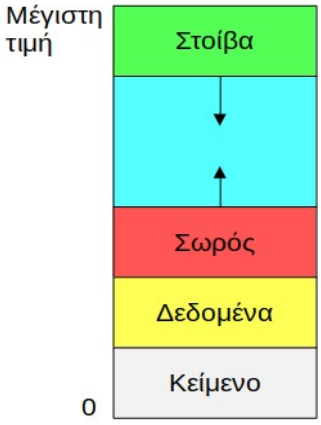
Τα πρώτα υπολογιστικά συστήματα επέτρεπαν να εκτελείται μόνο ένα πρόγραμμα ανά πάσα στιγμή. Αυτό το πρόγραμμα είχε τον πλήρη έλεγχο του συστήματος και είχε πρόσβαση σε όλους τους πόρους του. Αντίθετα, τα σημερινά υπολογιστικά συστήματα επιτρέπουν σε πολλαπλά προγράμματα να φορτώνονται στη μνήμη και να εκτελούνται ταυτόχρονα. Αυτή η εξέλιξη απαιτούσε αυστηρότερο έλεγχο και εκτενέστερη τμηματοποίηση των διαφόρων προγραμμάτων. Οι ανάγκες αυτές οδήγησαν στην ιδέα της διεργασίας (process), η οποία είναι ένα πρόγραμμα που εκτελείται. Μία διεργασία είναι μία μονάδα έργου σε ένα σύγχρονο σύστημα καταμερισμού χρόνου.

Όσο πιο πολύπλοκο είναι ένα λειτουργικό σύστημα, τόσο περισσότερα αναμένεται να κάνει για λογαριασμό των χρηστών του. Αν και ο κύριος στόχος του είναι η εκτέλεση των προγραμμάτων των χρηστών, πρέπει επίσης να διεκπεραιώνει διάφορες ενέργειες του συστήματος που είναι καλύτερα να μένουν εκτός του πυρήνα. Ένα σύστημα λοιπόν, αποτελείται από μία συλλογή διεργασιών: διεργασίες λειτουργικού συστήματος, που εκτελούν κώδικα συστήματος και διεργασίες χρήστη, που εκτελούν τον κώδικα χρήστη. Όλες αυτές οι διεργασίες είναι δυνατόν να εκτελούνται ταυτόχρονα, με πολυπλεξία της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ) ή των ΚΜΕ. Με την εναλλαγή των διεργασιών στην ΚΜΕ, το λειτουργικό σύστημα μπορεί να κάνει τον υπολογιστή πιο παραγωγικό.

Ένα σύστημα μαζικής επεξεργασίας εκτελεί εργασίες (jobs), ενώ ένα σύστημα καταμερισμού χρόνου εκτελεί προγράμματα χρήστη (user programs) ή ενέργειες (tasks). Ακόμα και σε ένα μονοχρηστικό σύστημα, ο χρήστης μπορεί να είναι σε θέση να εκτελεί αρκετά προγράμματα ταυτόχρονα: έναν επεξεργαστή κειμένου, ένα πρόγραμμα πλοήγησης και μια εφαρμογή ηλεκτρονικού ταχυδρομείου. Ακόμα και στην περίπτωση που ο χρήστης μπορεί να εκτελεί μόνο ένα πρόγραμμα τη φορά, όπως συμβαίνει σε μια ενσωματωμένη συσκευή, που δεν υποστηρίζει πολλαπλές εργασίες, το λειτουργικό σύστημα μπορεί να χρειάζεται να υποστηρίζει τις δικές του εσωτερικές προγραμματισμένες ενέργειες, όπως τη διαχείριση μνήμης. Από πολλές απόψεις, όλες αυτές οι δραστηριότητες είναι παρόμοιες, έτσι μπορούνε να ονομαστούνε όλες διεργασίες. Οι όροι εργασία και διεργασία χρησιμοποιούνται συχνά εναλλακτικά, καθώς ένα μεγάλο κομμάτι της θεωρίας και της ορολογίας των λειτουργικών συστημάτων αναπτύχθηκε σε μια χρονική περίοδο, κατά την οποία η κύρια δραστηριότητα των λειτουργικών συστημάτων ήταν η επεξεργασία εργασιών.

Μία διεργασία είναι κάτι περισσότερο από τον κώδικα ενός προγράμματος, ο οποίος μερικές φορές είναι γνωστός ως τμήμα κειμένου (text section). Περιλαμβάνει επίσης την τρέχουσα δραστηριότητα, όπως αυτή αναπαρίσταται από την τιμή του μετρητή προγράμματος (program counter) και τα περιεχόμενα των καταχωρητών του επεξεργαστή. Μια διεργασία, γενικά, περιλαμβάνει ακόμη την στοίβα (stack) διεργασίας, η οποία περιέχει προσωρινά δεδομένα (όπως παραμέτρους συναρτήσεων, διευθύνσεις επιστροφής και τοπικές μεταβλητές) και ένα τμήμα δεδομένων (data section), το οποίο περιέχει καθολικές μεταβλητές. Μπορεί επίσης να περιλαμβάνει ένα σωρό (heap), ο οποίος είναι μνήμη, που

δεσμεύεται δυναμικά κατά την διάρκεια του χρόνου εκτέλεσης της διεργασίας. Η δομή μιας διεργασίας στην μνήμη εμφανίζεται στην Εικόνα 29.



Εικόνα 29. Μία διεργασία στη μνήμη.

‘Ενα πρόγραμμα δεν είναι από μόνο του διεργασία, αλλά μια παθητική οντότητα, όπως ένα αποθηκευμένο στο δίσκο αρχείο που περιέχει μια λίστα εντολών (συχνά καλείται εκτελέσιμο αρχείο - executable file), ενώ μια διεργασία είναι μια ενεργή οντότητα με ένα μετρητή προγράμματος που υποδεικνύει την επόμενη προς εκτέλεση εντολή και ένα σύνολο από συσχετισμένους πόρους. Ένα πρόγραμμα γίνεται διεργασία όταν ένα εκτελέσιμο αρχείο φορτώνεται στη μνήμη. Δύο κοινές τεχνικές για τη φόρτωση των εκτελέσιμων αρχείων είναι το διπλό κλικ στο εικονίδιο που αντιπροσωπεύει το εκτελέσιμο αρχείο και η εισαγωγή του ονόματος του εκτελέσιμου αρχείου στη γραμμή εντολών (όπως prog.exe ή a. out).

Παρόλο που δύο διεργασίες μπορούν να σχετίζονται με το ίδιο πρόγραμμα, θεωρούνται ως δύο ξεχωριστές ακολουθίες εκτέλεσης. Για παράδειγμα, αρκετοί χρήστες μπορούν να εκτελούν διαφορετικά αντίγραφα του προγράμματος ηλεκτρονικού ταχυδρομείου ή ο ίδιος χρήστης μπορεί να καλεί πολλά αντίγραφα του προγράμματος πλοήγησης. Κάθε αντίγραφο είναι μία ξεχωριστή διεργασία και, παρόλο που τα τμήματα κειμένου είναι ίδια, τα τμήματα δεδομένων, σωρού και στοίβας διαφέρουν. Επίσης, συχνά υπάρχει μία διεργασία που γεννά πολλές διεργασίες ενώ εκτελείται.

Μια διεργασία μπορεί να είναι από μόνη της ένα περιβάλλον εκτέλεσης για άλλο κώδικα. Το περιβάλλον προγραμματισμού Java αποτελεί ένα παράδειγμα. Στις περισσότερες περιπτώσεις, ένα εκτελέσιμο πρόγραμμα Java εκτελείται μέσα στην εικονική μηχανή της Java (VM). Το JVM εκτελεί μια διεργασία, η οποία διερμηνεύει τον φορτωμένο κώδικα Java και εκτελεί ενέργειες (μέσω εγγενών εντολών μηχανής) εκ μέρους αυτού του κώδικα. Για παράδει μα, για να εκτελεστεί το μεταγλωττισμένο πρόγραμμα Program.class της Java, θα εισαχθεί στη γραμμή εντολών το παρακάτω:

java Program

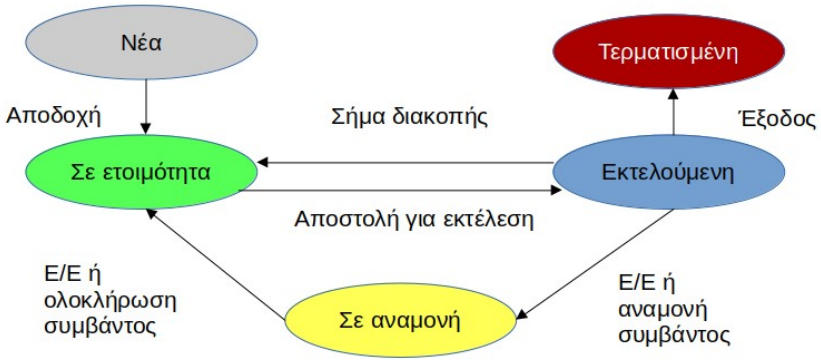
Η εντολή Java εκτελεί το JVM σαν μια κανονική διεργασία, η οποία με την σειρά της εκτελεί το πρόγραμμα Program της Java μέσα στην εικονική μηχανή. Η έννοια είναι ίδια με την έννοια της προσομοίωσης, εκτός του ότι ο κώδικας, αντί να γραφεί για ένα διαφορετικό σύνολο εντολών, γράφεται στη γλώσσα Java.

6.2 Κατάσταση διεργασίας

Καθώς εκτελείται μια διεργασία, η κατάστασή της μπορεί να αλλάξει. Η δραστηριότητα της διεργασίας σε μία χρονική στιγμή ορίζει την κατάστασή της, η οποία μπορεί να είναι μία από τις ακόλουθες:

- **Νέα (New):** Δημιουργία της διεργασίας.
- **Εκτελούμενη (Running):** Στη κατάσταση αυτή οι εντολές της εκτελούνται.
- **Σε αναμονή (Waiting):** Στη κατάσταση αυτή υπάρχει αναμονή κάποιου συμβάντος [όπως η λήψη ενός σήματος ή η ολοκλήρωση ενός γεγονότος Εισόδου/Εξόδου (Ε/Ε)].
- **Σε ετοιμότητα (Ready):** Στη κατάσταση αυτή υπάρχει αναμονή διάθεσης της διεργασίας σε έναν επεξεργαστή.
- **Τερματισμένη (Terminated):** Ολοκλήρωση της διεργασίας και τερματισμός.

Οι ονομασίες αυτές μπορεί να διαφέρουν ανάμεσα στα διάφορα λειτουργικά συστήματα, όμως οι καταστάσεις που αντιπροσωπεύουν συναντώνται σε όλα. Ορισμένα λειτουργικά συστήματα ορίζουν με περισσότερες λεπτομέρειες τις καταστάσεις των διεργασιών, όμως είναι σημαντικό να διευκρινισθεί ότι σε έναν επεξεργαστή, μόνο μία διεργασία μπορεί να βρίσκεται σε κατάσταση εκτέλεσης κάθε στιγμή, και πολλές διεργασίες μπορούν να βρίσκονται σε κατάσταση ετοιμότητας και αναμονής. Το διάγραμμα καταστάσεων που απεικονίζει τις καταστάσεις αυτές, παρουσιάζεται στην Εικόνα 30.



Εικόνα 30. Διάγραμμα καταστάσεων διεργασίας.

6.3 Πίνακας ελέγχου διεργασίας

Κάθε διεργασία παρίσταται μέσα στο λειτουργικό σύστημα από ένα μπλόκ ελέγχου διεργασίας (Process Control Block – PCB ή task control block). Ένα μπλόκ ελέγχου διεργασίας παρουσιάζεται στην Εικόνα 31.

Εικόνα 31. Πίνακας Ελέγχου Διεργασίας (PCB).

Το μπλόκ ελέγχου διεργασίας περιέχει πολλά τμήματα πληροφοριών, που σχετίζονται με μία συγκεκριμένη διεργασία, όπως τα παρακάτω:

- **Κατάσταση διεργασίας:** Η διεργασία μπορεί να είναι νέα, σε κατάσταση ετοιμότητας, εκτέλεσης, αναμονής, τερματισμού κ.ο.κ.

- **Μετρητής Προγράμματος:** Ο μετρητής προγράμματος υποδεικνύει τη διεύθυνση της επόμενης προς εκτέλεση εντολής για αυτή τη διεργασία.

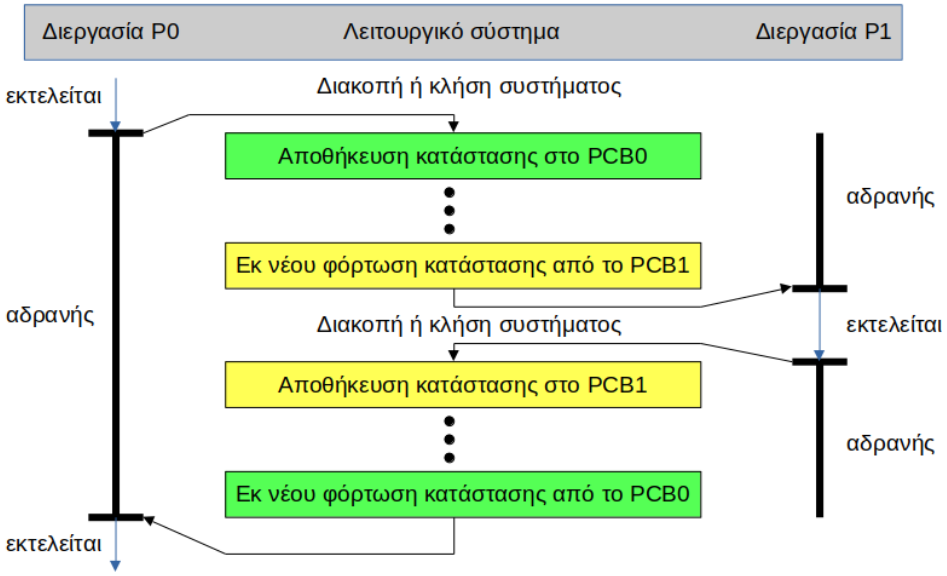
- **Καταχωρητές ΚΜΕ:** Οι καταχωρητές ποικίλλουν σε αριθμό και τύπο ανάλογα με την αρχιτεκτονική του υπολογιστή. Συμπεριλαμβάνουν συσσωρευτές, καταχωρητές δεικτοδότησης, δείκτες στοίβας και καταχωρητές γενικού σκοπού, και επιλέον οποιαδήποτε πληροφορία κατάστασης κώδικα. Αυτές οι πληροφορίες κατάστασης μαζί με το μετρητή προγράμματος πρέπει να αποθηκεύονται, όταν συμβεί μία διακοπή, έτσι ώστε η διεργασία να μπορεί συνεχίσει την εκτέλεσή της σωστά αμέσως μετά (Εικόνα 32).

- **Πληροφορίες χρονοπρογραμματισμού της ΚΜΕ:** Αυτές οι πληροφορίες περιλαμβάνουν την προτεραιότητα της διεργασίας, δείκτες προς ουρές χρονοπρογραμματισμού και κάθε άλλη παράμετρο χρονοπρογραμματισμού.

- **Πληροφορίες διαχείρισης μνήμης:** Αυτές οι πληροφορίες περιλαμβάνουν τις τιμές των καταχωρητών βάσης και ορίου, τους πίνακες σελίδων ή τους πίνακες τμημάτων, ανάλογα με το σύστημα μνήμης που χρησιμοποιείται από το λειτουργικό σύστημα.

- **Λογιστικές Πληροφορίες (Accounting information):** Αυτές οι πληροφορίες περιλαμβάνουν το ποσοστό χρήσης της ΚΜΕ, τον πραγματικό χρόνο χρήσης της ΚΜΕ, τα χρονικά όρια, τους αριθμούς λογαριασμών, τους αριθμούς εργασιών ή διεργασιών κ.ο.κ.

- **Πληροφορίες κατάστασης Ε/Ε (I/O status information):** Αυτές οι πληροφορίες περιλαμβάνουν τη λίστα συσκευών Ε/Ε που έχουν αποδοθεί στη διεργασία, μία λίστα των ανοικτών αρχείων, κ.ο.κ.



Εικόνα 32. Διάγραμμα εναλλαγής της ΚΜΕ από διεργασία σε διεργασία.

Με λίγα λόγια, ο πίνακας ελέγχου διεργασίας απλά χρησιμεύει ως αποθήκη για κάθε πληροφορία, η οποία μπορεί να διαφέρει από διεργασία σε διεργασία.

6.4 Συγχρονισμός διεργασιών

Συνεργαζόμενη διεργασία (cooperating process) ονομάζεται η διεργασία, η οποία μπορεί να επηρεάσει άλλες διεργασίες, που εκτελούνται στο σύστημα και να επηρεαστεί από αυτές. Οι συνεργαζόμενες διεργασίες μπορούν είτε να διαμοιράζονται απευθείας ένα χώρο λογικών διευθύνσεων (δηλ., κώδικα και δεδομένα) είτε να τους επιτρέπεται να διαμοιράζονται δεδομένα μόνο μέσω αρχείων και μηνυμάτων. Η συνδρομική πρόσβαση σε διαμοιραζόμενα δεδομένα μπορεί να οδηγήσει σε ασυνέπεια δεδομένων, για τον λόγο αυτό, υπάρχουν διάφοροι μηχανισμοί, οι οποίοι διασφαλίζουν τη μεθοδική εκτέλεση των συνεργαζόμενων διεργασιών, οι οποίες διαμοιράζονται ένα χώρο λογικών διευθύνσεων, έτσι ώστε να διατηρείται η συνέπεια των δεδομένων.

6.4.1Σημαφόροι

Οι σημαφόροι αποτελούν ένα στιβαρό εργαλείο και μπορούν να παρέχουν περίτεχνους τρόπους ώστε οι διεργασίες να μπορούν να συγχρονίζουν τις

δραστηριότητές τους. Ένας σημαφόρος (semaphore) S είναι μία ακέραιη μεταβλητή η οποία, με εξαίρεση την αρχικοποίησή της, προσπελαύνεται μόνο μέσω δύο πρότυπων ατομικών λειτουργιών: wait() και signal(). Η λειτουργία wait ( ) αρχικά ονομαζόταν P (από το Ολλανδικό proberen, "ελέγχω"). Η signal() αρχικά ονομαζόταν V (από το verhogen, "προσαυξάνω").
Ο ορισμός της wait( ) είναι ο εξής:

```
wait (S) {
        while (S <= 0) ; // αναμονή με έργο
        S--;
}
```

Ο ορισμός της signal( ) είναι ο εξής:

```
signal (S){
        S++;
}
```

Όλες οι τροποποιήσεις της ακέραιης τιμής του σημαφόρου, που περιέχονται στις λειτουργίες wait() και signal(), πρέπει να εκτελούνται αδιαίρετα. Δηλαδή, όταν μία διεργασία τροποποιεί την τιμή του σημαφόρου, καμία άλλη διεργασία δεν μπορεί να τροποποιήσει την τιμή του ίδιου σημαφόρου ταυτόχρονα. Επιπρόσθετα, στην περίπτωση της wait(S), ο έλεγχος της ακέραιας τιμής του S (S<=0) και η πιθανή τροποποίησή της (S--), πρέπει επίσης να εκτελούνται χωρίς διακοπή.
Τα λειτουργικά συστήματα συχνά κάνουν διάκριση μεταξύ των μετρητικών σημαφόρων και των δυαδικών σημαφόρων. Η τιμή ενός μετρητικού σημαφόρου (counting semaphore) μπορεί να κινείται μέσα σε ένα απεριόριστο πεδίο τιμών. Η τιμή ενός δυαδικού σημαφόρου (binary semaphore) μπορεί να παίρνει μόνο τις τιμές 0 και 1.

Οι μετρητικοί σημαφόροι μπορούν να χρησιμοποιηθούν για τον έλεγχο της πρόσβασης σε ένα συγκεκριμένο πόρο, που αποτελείται από ένα πεπερασμένο αριθμό στιγμιότυπων (instances). Ο σημαφόρος αρχικοποιείται στο πλήθος των διαθέσιμων πόρων. Κάθε διεργασία που επιθυμεί να χρησιμοποιήσει έναν πόρο εκτελεί μία λειτουργία wait() πάνω στο σημαφόρο (μειώνοντας έτσι τον μετρητή κατά ένα). Όταν μία διεργασία αποδεσμεύει έναν πόρο, εκτελεί μία λειτουργία signal() (αυξάνοντας το μετρητή κατά ένα). Όταν η τιμή του σημαφόρου φθάσει στο 0, όλοι οι πόροι βρίσκονται σε χρήση. Μετά από αυτό, οι διεργασίες που επιθυμούν να χρησιμοποιήσουν έναν πόρο θα μπλοκαριστούν μέχρι η τιμή του σημαφόρου να γίνει μεγαλύτερη του 0.

Μπορεί επίσης να γίνει χρήση των σημαφόρων, ώστε να λυθούν διάφορα προβλήματα συγχρονισμού. Για παράδειγμα, έστω δύο διεργασίες, που εκτελούνται συνδρομικά: η P_1 με μία πρόταση S_1 και η P_2. με μία πρόταση S_2. Έστω ότι απαιτείται η S_2 να εκτελεστεί μόνο μετά την ολοκλήρωση της S_1. Η υλοποίηση αυτού του σχήματος γίνεται εύκολα, κάνοντας τις διεργασίες P_1 και P_2 να διαμοιράζονται έναν κοινό σημαφόρο synch, αρχικοποιημένο στο 0.

Στην διεργασία P_1, εισάγονται οι προτάσεις:

```
S1;
signal(synch);
```

Στην διεργασία P_2, εισάγονται οι προτάσεις:

```
wait(synch);
S2;
```

Επειδή ο σημαφόρος synch αρχικοποιείται στο 0, η P_2 θα εκτελέσει την S_2 μόνο αφού η P_1 καλέσει την signal(synch), μία ενέργεια που θα γίνει εφικτή μετά την εκτέλεση της πρότασης S_1.

Το σύστημα αυτόματης αναγνώρισης αντικειμένων χρησιμοποιεί παράληλλα τέσσερις (4) διεργασίες, οι οποίες δύναται να τερματιστούν με τη χρήση σημαφόρων, οι οποίοι είναι υλοποιημένοι στη γλώσσα προγραμματισμού Python. Περισσότερες λεπτομέρειες θα γνωστοποιηθούν στα Κεφάλαια 7 και 8.

7 Περιγραφή συστήματος αυτόματης αναγνώρισης αντικειμένων

7.1 Ανάλυση λειτουργίας συστήματος

Το σύστημα αυτόματης αναγνώρισης αντικειμένων αποτελεί έναν συνδυασμό υλισμικού και λογισμικού. Ο χρήστης αυτού, εκτελεί συγκεκριμένες ενέργειες έτσι ώστε ένα αντικείμενο να αναγνωρίζεται, να ακολουθείται και να δίνονται συγκεκριμένες εντολές από το σύστημα στο μη επανδρωμένο αεροσκάφος ή στον προσομοιωτή (SITL simulator). Στο Κεφάλαιο 3 έγινε η ανάλυση του υλισμικού που περιλαμβάνει το σύστημα. Το απαραίτητο λογισμικό που χρήζει εγκατάστασης για τη λειτουργία του συστήματος παρουσιάζεται παρακάτω:

- Το λειτουργικό σύστημα Raspberry Pi OS, που αποτελεί τη βάση του μικροϋπολογιστή Raspberry Pi.
- Η βιβλιοθήκη λογισμικού OpenCV, που απαιτείται για την χρήση των ανιχνευτών (trackers).
- Η βιβλιοθήκη λογισμικού pantilthat, που απαιτείται για τη χρήση των σερβομηχανισμών, οι οποίοι μετακινούν την κάμερα.
- Η βιβλιοθήκη λογισμικού pymavlink, που απαιτείται για τη χρήση του πρωτοκόλλου επικοινωνίας MavLink.
- Ο προσομοιωτής SITL, έτσι ώστε να γίνει εφικτή η προσομοίωση ενός μη επανδρωμένου αεροσκάφους και οι κινήσεις του.

Στη παρούσα εργασία, οι εντολές στο μη επανδρωμένο αεροσκάφος (συγκεκριμένα σε ένα τετρακόπτερο) θα δωθούν στον προσομοιωτή SITL, χωρίς να γίνει χρήση του αυτόματου πιλότου Pixhawk Cube.

Το παραπάνω λογισμικό που αναφέρθηκε εγκαθίσταται στον μικροϋπολογιστή Raspberry Pi, όπου και εκτελείται το σύστημα αυτόματης αναγνώρισης αντικειμένων. Μοναδική εξαίρεση αποτελεί ο προσομοιωτής SITL, όπου εγκαθίσταται σε έναν ξεχωριστό υπολογιστή, που λαμβάνει το ρόλο του επίγειου σταθμού ελέγχου (GCS). Η επικοινωνία μεταξύ του συστήματος και του προσομοιωτή επιτυγχάνεται ασύρματα, στέλνωντας συγκεκριμένες εντολές στη διεύθυνση IP του υπολογιστή.

Ο πηγαίος κώδικας για την λειτουργία του συστήματος, έχει χωριστεί σε τρία (3) ξεχωριστά αρχεία, τα οποία συνδέονται μεταξύ τους, και είναι γραμμένος στη γλώσσα προγραμματισμού Python. Τα αρχεία αυτά είναι τα παρακάτω:

1. Αρχείο ***classes.py*****:** Στο αρχείο αυτό είναι γραμμένες η κλάση του ελεγκτή PID και η κλάση του ανιχνευτή αντικειμένου.

2. Αρχείο ***modules.py*****:** Στο αρχείο αυτό είναι γραμμένες διάφοροι μέθοδοι που είναι απαραίτητες για το σύστημα, όπως ο χειριστής σημαφόρων, ο μετακινητής ενός σερβομηχανισμού κ.α.

3. Αρχείο ***pan_tilt_main.py*****:** Στο αρχείο αυτό είναι γραμμένο το κυρίως πρόγραμμα του συστήματος που δημιουργεί τις απαραίτητες διεργασίες για την ολοκληρωμένη λειτουργία του.

Η ανάλυση και η επεξήγηση του κώδικα, όλων των ανωτέρω αρχείων, θα λάβει χώρα στο Κεφάλαιο 8. Με την εκκίνηση την μικροϋπολογιστή Raspberry Pi και πληκτρολογώντας την εντολή *python3 pan_tilt_main.py* σε ένα τερματικό, ξεκινάει η λειτουργία του συστήματος, το οποίο δημιουργεί και εκτελεί τέσσερις (4) ξεχωριστές διεργασίες, οι οποίες εκτελούνται παράληλλα και μοιράζονται τις τιμές συγκεκριμένων μεταβλητών. Οι διεργασίες αυτές είναι οι ακόλουθες:

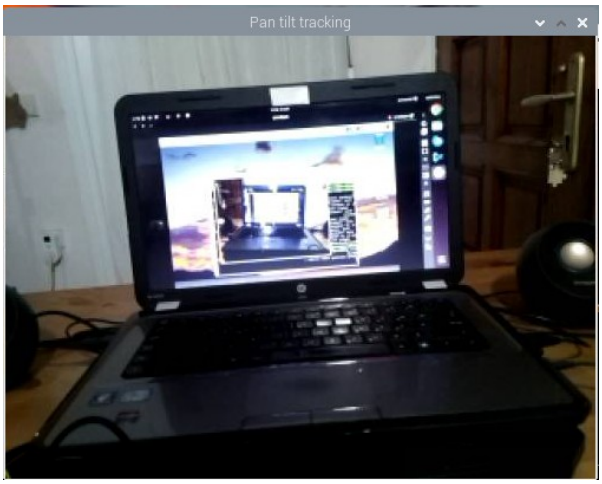
- Διεργασία 1 (processTracker): Η διεργασία αυτή είναι υπεύθυνη για τη λειτουργία του ανιχνευτή αναγνώρισης ενός αντικειμένου.

- Διεργασία 2 (processPanning): Η διεργασία αυτή είναι υπεύθυνη για τον έλεγχο PID που γίνεται στην οριζόντια κατεύθυνση, δίνωντας κατάλληλη τιμή, ώστε η μετακίνηση του σερβομηχανισμού που κινείται οριζόντια (και κατ' επέκταση της κάμερας) να γίνεται σε συνάρτηση με το κέντρο του αντικειμένου που ανιχνευέται.

- Διεργασία 3 (processTilting): Η διεργασία αυτή εργάζεται ακριβώς όπως τη Διεργασία 2 , δίνωντας όμως την κατάλληλη τιμή για τον σερβομηχανισμό που κινείται κάθετα.

- Διεργασία 4 (processSetServos): Η διεργασία αυτή λαμβάνει τις τιμές των διεργασιών 2 και 3, οι οποίες έχουν εφαρμόσει προηγουμένως έλεγχο PID, και μετακινεί τους δύο σερβομηχανισμούς στις κατάλληλες μοίρες ώστε να ανιχνεύεται ορθά το κέντρο του αντικειμένου.

Αρχικά, κατά την εκκίνηση του συστήματος, δεν υπάρχει αντικείμενο προς αναγνώριση και παρέχεται μόνο η εικόνα της κάμερας σε πραγματικό χρόνο (Εικόνα 33).



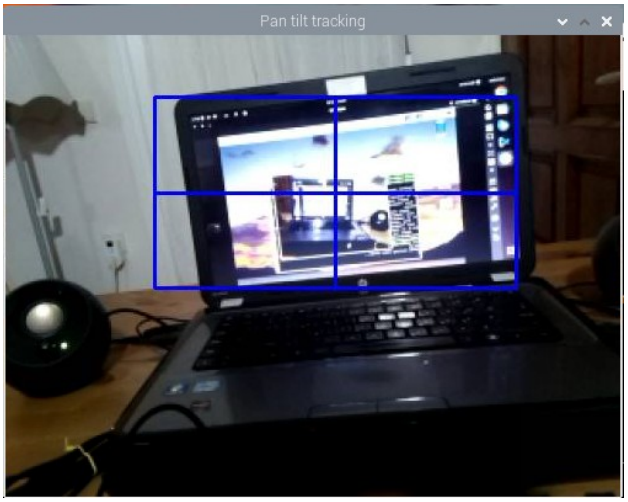
Εικόνα 33. Αρχική εκκίνηση του συστήματος αυτόματης αναγνώρισης αντικειμένων

Ο χρήστης έχει τη δυνατότητα να εκτελέσει τρεις διαφορετικές ενέργειες, κάνοντας χρήση των παρακάτω πλήκτρων:

1. Πλήκτρο «**s**» **(select)**: Με το πλήκτρο αυτό ενργοποιείται ο ανιχνευτής και ο χρήστης καλείται να επιλέξει ένα πλαίσιο (bounding box) που θα περιέχει το αντικείμενο που πρόκειται να ανιχνευτεί στη συνέχεια (Εικόνα 34). Με το πλήκτρο «**Enter**» επιβεβαιώνεται το πλαίσιο και κατ' επέκταση το αντικείμενο.

2. Πλήκτρο «**c**» **(re-select)**: Με το πλήκτρο αυτό ο χρήστης μπορεί να επιλέξει εκ νέου ένα διαφορετικό αντικείμενο μέσα σε ένα καινούργιο πλαίσιο. Η χρήση του πλήκτρου αυτού προϋποθέτει ότι ο ανιχνευτής έχει ήδη ενεργοποιηθεί (έχει ήδη γίνει χρήση του πλήκτρου «**s**»). Με το πλήκτρο «**Enter**» επιβεβαιώνεται το νέο πλαίσιο και κατ' επέκταση το νέο αντικείμενο.

3. Πλήκτρο «**q**» **(quit)**: Με το πλήκτρο αυτό ο χρήστης μπορεί να τερματίσει όλες τις διεργασίες, οι οποίες επικοινωνούν μέσω σημαφόρων, τερματίζοντας με αυτό τον τρόπο όλο το σύστημα.



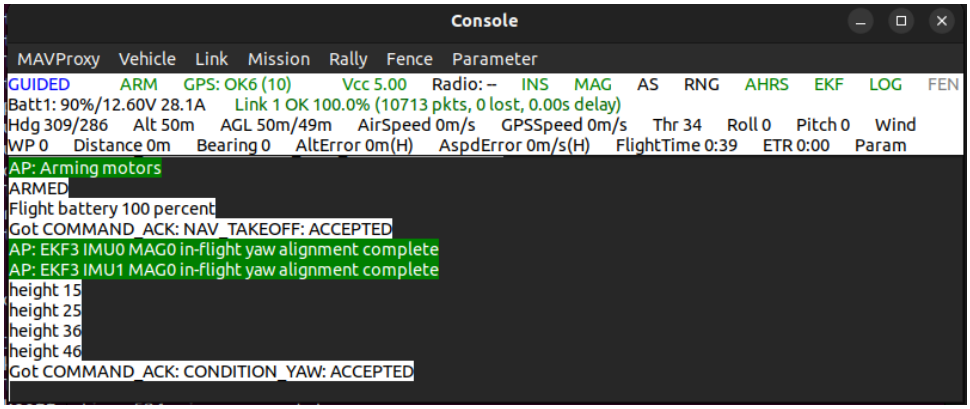
Εικόνα 34. Επιλογή πλαισίου (bounding box) που περιέχει το αντικείμενο που ανιχνεύεται.

Στη συνέχεια, αφού επιλεχθεί το αντικείμενο, οι τέσσερις διεργασίες που αναφέρθηκαν παραπάνω συνεργάζονται, δίνωντας κάθε μία τις δικές τις τιμές (για ορισμένες κοινές μεταβλητές), ως προς τη θέση του αντικειμένου, τη θέση του αντικειμένου σε συνάρτηση με το κέντρο του καρέ (frame), μετά από έλεγχο PID και τις μοίρες που πρέπει να περιστραφεί κάθε σερβομηχανισμός. Το επιθυμητό αποτέλεσμα του συστήματος είναι η επίτευξη της περιστροφή της κάμερας, έτσι ώστε το κέντρο του καρέ να βρίσκεται στο κέντρο του πλαισίου που περιέχει το αντικείμενο που ανιχνεύεται.

Κατά τη διάρκεια λειτουργίας του συστήματος διενεργείται και η επικοινωνία με τον προσομοιωτή SITL μέ εντολές της βιβλιοθήκης pymavlink. Οι εντολές αυτές έχουν ως σκοπό να περιστρέψουν δεξιά-αριστερά ή να ανυψώσουν-καθυψώσουν το τετρακόπτερο που προσομοιώνεται σε συνάρτηση με τη περιστροφή της κάμερας. Για παράδειγμα, όταν το αντικείμενο που ανιχνεύεται μετακινηθεί 200 pixel αριστερά από το κέντρο του καρέ και ο αντίστοιχος σερβομηχανισμός κληθεί να μετακινήσει την κάμερα περί των 30 μοιρών, τότε θα δωθεί εντολή στο τετρακόπτερο να περιστραφεί αριστερά 30 μοίρες. Στην Εικόνα 35 παρουσιάζεται η κατάσταση του τετρακόπτερου πριν την περιστροφή, στην Εικόνα 36 παρουσιάζεται η λήψη της εντολής περιστροφής (Got COMMAND_ACK :

CONDITION_YAW : ACCEPTED) και στην Εικόνα 37 παρουσιάζεται η κατάσταση του τετρακόπτερου μετά την περιστροφή.

Εικόνα 35. Η κατάσταση του τετρακόπτερου πρίν την περιστροφή.



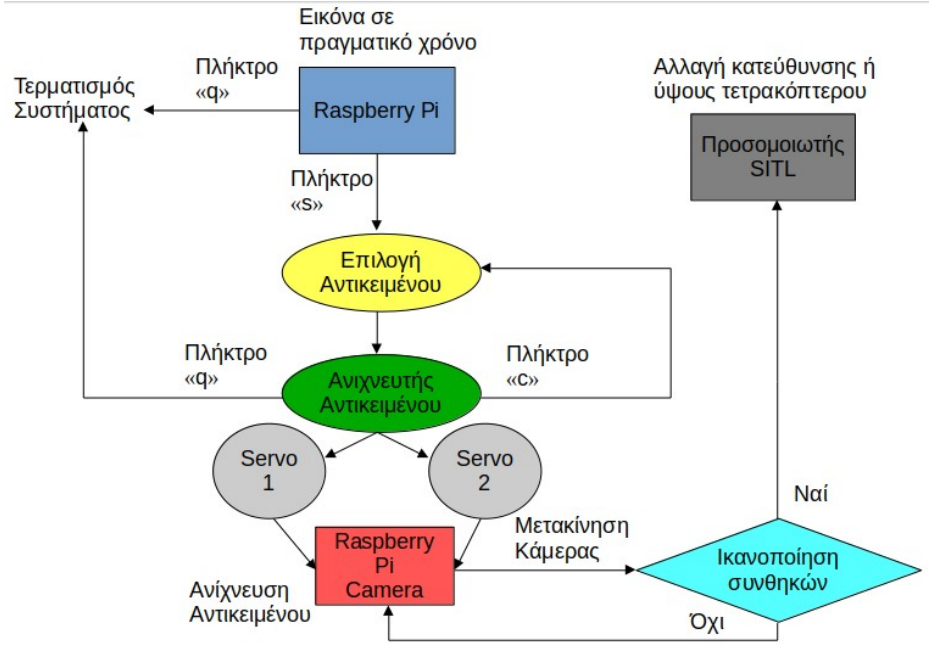
Εικόνα 36. Λήψη της εντολής περιστροφής (Got COMMAND_ACK : CONDITION_YAW : ACCEPTED).

Εικόνα 37. Η κατάσταση του τετρακόπτερου μετά την περιστροφή.

Το επιθυμητό αποτέλεσμα είναι το τετρακόπτερο να μπορεί να περιστραφεί ή να αυξομοιώσει το ύψος του σε συνάρτηση με τη κίνηση των σερβομηχανισμών , έτσι ώστε ο ανιχνευτής να μην χάσει το αντικείμενο από το πεδίο του. Σημειώνεται ότι η κίνηση των

σερβομηχανισμών περιορίζεται συνολικά στις 180 μοίρες (-90 και +90 μοίρες προς κάθε κατεύθυνση αντίστοιχα).

Τέλος, στην Εικόνα 38 παρουσιάζεται το Διάγραμμα ενεργειών του συστήματος αυτόματης αναγνώρισης αντικειμένων, όπως αναλύθηκε παραπάνω.



Εικόνα 38. Διάγραμμα ενεργειών του συστήματος αυτόματης αναγνώρισης αντικειμένων.

7.2 Χρήσεις του συστήματος και μελλοντικές επεκτάσεις

Το ολοκληρώμενο σύστημα αυτόματης αναγνώρισης αντικειμένων, όπως αναλύθηκε προηγουμένως, δίνει την δυνατότητα σε έναν χρήστη να παρακολουθεί ένα αντικείμενο που επιθυμεί από τον ηλεκτονικό υπολογιστή του σε πραγματικό χρόνο. Αυτή η δυνατότητα μπορεί να εφαρμοστεί σε δίαφορους τομείς, όπως παρακάτω:

1. **Ασφάλεια εγκαταστάσεων:** Στον τομέα αυτό, παρατηρείται στις μέρες μας, ότι τα κυριότερα ηλεκτρονικά συστήματα ασφάλειας που χρησιμοποιούνται, είναι οι στατικές κάμερες. Οι κάμερες αυτές καλύπτουν στγκεκριμένους τομείς παρατήρησης με αποτέλεσμα να υπάρχουν συνήθως περιοχές οι οποίες δεν μπορούν να καλυφθούν. Ενα σύστημα αυτόματης αναγνώρισης αντικειμένων, τοποθετημένο σε ένα μη επανδρωμένο αεροσκάφος, θα μπορούσε να χρησιμοποιηθεί από προσωπικό ασφαλείας, ώστε να ακολοθούνται αντικειμένα αυτόματα (π.χ ένας εισβολέας ή ένα όχημα) σε όλο τον χώρο, ακόμα και σε σημεία μη ελεγχόμενα απο στατικές κάμερες.

2. **Κινηματογράφος:** Στον τομέα αυτό, ένα τέτοιο σύστημα θα μπορούσε να παρέχει υλικό βίντεο από συγκεκριμένους ηθοποιούς ή και άλλα αντικείμενα γενικότερα (π.χ οχήματα που κινούνται), αποτελώντας έτσι ένα επιπλέον εργαλείο για το προσωπικό που εργάζεται για το σκοπό αυτό.

3. **Φωτογραφία:** Στον τομέα αυτό, ένα τέτοιο σύστημα θα μπορούσε να δώσει τη δυνατότητα τόσο σε επαγγελματίες, όσο και σε ερασιτέχνες φωτογράφους, να κάνουν δύσκολες λήψεις φωτογραφιών, από απόσταση, σε αντικείμενα που κινούνται, όπως σε ανθρώπους, σε οχήματα, σε άγρια ζώα κ.τ.λ.

Το υπάρχων σύστημα αυτόματης αναγνώρισης αντικειμένων, που αναπτύχθηκε, δύναται να λάβει πολλές μελλοντικές επεκτάσεις. Στη συνέχεια, ακολουθούν κάποιες από αυτές:

1. **Γραφικό περιβάλλον χρήστη (User Interface):** Η ανάπτυξη ενός γραφικού περιβάλλοντος για τους χρήστες, είναι μία επέκταση που μπορεί εύκολα να δημιουργηθεί, δίνωντας τους έτσι την δυνατότητα να επιλέγουν απλά με "κλικ", τον ανιχνευτή που θα χρησιμοποιήσει το σύστημα, τη λήψη φωτογραφιών ή βιντεο κ.τ.λ

2. **Κινήσεις του μη επανδρωμένου αερασάφους:** Στο υπάρχων σύστημα, ένα μη επανδρωμένο αεροσκάφος έχει τη δυνατότητά να αυξομειώσει το ύψος του και να κινηθεί στον οριζόντιο άξονα. Με μια μικρή επέκταση, θα μπορεί να κινείται μπροστά και πίσω ανάλογα με την κίνηση του αντικειμένου (π.χ απομάκρυνση αυτού απο τη κάμερα θα σήμαινε ότι το αεροσκάφος πρέπει να κινηθεί εμπρός).

3. **Εφαρμογή βαθιάς μάθησης (deep learning), με χρήση του Intel Movidius – Vision Processing Units (VPUs):** Το Movidius είναι ένα usb στικ, στο οποίο μπορεί κάποιος να εκπαιδεύσει ένα νευρωνικό δίκτυο και στη συνέχεια να το χρησιμοποιήσει μέσω του στικ σε ένα διαφορετικό σύστημα. Στο σύστημα αυτόματης αναγνώρισης αντικειμένων, τοποθετώντας το στικ στον μικροϋπολογιστή Raspberry Pi, θα ήταν δυνατή η αυτόματη εύρεση αντικειμένων, για τα οποία θα είχε εκπαιδευτεί το νευρωνικό δίκτυο, και η ανίχνευση – παρακολούθηση αυτών από το υφιστάμενο σύστημα. Για παράδειγμα, η εκπαίδευση του νευρωνικού δικτύου στην αναγνώριση οχημάτων, θα έδινε την δυνατότητα στο σύστημα να αναγνωρίζει αυτόματα τα οχήματα και στη συνέχεια το μη επανδρωμένο αεροσκάφος να τα ακολουθεί.

4. **Χρήση του συστήματος σε άλλα μη επανδρωμένα οχήματα:** Το σύστημα όπως ήδη περιγράφηκε, λειτουργεί και δίνει εντολές σε μη επανδρωμένα αεροσκάφη μέσω του αυτόματου πιλότου Pixhawk Cube. Είναι εφικτή όμως και η χρησιμοποίησή του σε άλλα μη επανδρωμένα οχήματα (επίγεια, υποβρύχια κ.τ.λ.) απλά αφαιρώντας τον αυτόματο πιλότο και τροποποιώντας κατάληλλα τον πηγαίο κώδικα.

8 Συγγραφή κώδικα σε γλώσσα προγραμματισμού Python και ανάλυση

Στο Κεφάλαιο 7, κατά την ανάλυση του συστήματος, έγινε αναφορά για τη συγγραφή του πηγαίου κώδικα στη γλώσσα προγραμματισμού Python, καθώς και στο διαχωρισμό του κώδικα σε τρία αρχεία. Στο κεφάλαιο αυτό, παρατίθεται ο κώδικας των αρχείων αυτών και επεξηγείται.

(1) Αρχείο ***classes.py***

Στο αρχείο αυτό έχουν γραφτεί οι κλάσεις του ελεγκτή PID και του ανιχνευτή αντικειμένων. Ακολουθεί ο κώδικας:

```
1.import time
2.import cv2
3.import signal
4.import argparse
5.import modules as m
```

Στις γραμμές κώδικα 1 εως 4 γίνεται η εισαγωγή των απαραίτητων βιβλιοθηκών. Στη γραμμή 5 εισάγεται το δεύτερο αρχείο *modules.py* , ώστε να χρησιμοποιηθούν συγκεκριμένες μέθοδοι.

```
7.class PID:
8.      def __init__(self, kP, kI, kD): #kP=1, kI=0, kD=0
9.              # αρχικοποίηση τιμών
10.             self.kP = kP
11.             self.kI = kI
12.             self.kD = kD
13.     def initialize(self):
14.             # αρχικοποίηση χρόνου
15.             self.currTime = time.time()
16.             self.prevTime = self.currTime
17.             # αρχικοποίηση σφάλματος (error)
18.             self.prevError = 0
19.             # αρχικοποίηση σταθερών
20.             self.cP = 0
21.             self.cI = 0
22.             self.cD = 0
```

Στις γραμμές κώδικα 8-22 έχουμε τις απαραίτητες αρχικοποιήσεις των μεταβλητών.

```
23.     def update(self, error, sleep=0.2):
24.             # μια μικρή παύση
25.             time.sleep(sleep)
26.             #υπολογισμός του deltatime
27.             self.currTime = time.time()
28.             deltaTime = self.currTime - self.prevTime
29.             #υπολογισμός του delta error
30.             deltaError = error - self.prevError
31.             #αναλογικός όρος
32.             self.cP = error
33.             # ολοκληρωτικός όρος
34.             self.cI += error * deltaTime
35.             # παραγωγικός όρος
36.             self.cD = (deltaError / deltaTime) if deltaTime > 0 else 0
37.             # αποθήκευση μεταβλητών previous time και previous error για το
                επόμενο update
38.             self.prevtime = self.currTime
39.             self.prevError = error
40.             # πρόσθεση όρων και επιστροφή αυτών
41.             return sum([
42.                     self.kP * self.cP,
43.                     self.kI * self.cI,
44.                     self.kD * self.cD])
```

Στις γραμμές κώδικα 23 εως 44 ορίζεται η μέθοδος update, όπου λειτουργεί ο έλεγχος PID σε κάθε καρέ του αντικειμένου που ανιχνεύεται.

```
46.class ObjectTracker:
47.     def __init__(self, track_name):
48.             # χειρισμός σήματος τερματισμού ( keyboard interrupt )
49.             signal.signal(signal.SIGINT, m.signal_handler)
50.
51.             ap = argparse.ArgumentParser()
52.
53.             ap.add_argument("-t", "--tracker", type=str, default=track_name,
54.                     help="OpenCV object tracker type")
55.
56.             args = vars(ap.parse_args())
57.             OPENCV_OBJECT_TRACKERS = {
58.                             "csrt": cv2.TrackerCSRT_create,
59.                             "kcf": cv2.TrackerKCF_create,
60.                             "boosting": cv2.TrackerBoosting_create,
61.                             "mil": cv2.TrackerMIL_create,
62.                             "tld": cv2.TrackerTLD_create,
63.                             "medianflow": cv2.TrackerMedianFlow_create,
64.                             "mosse": cv2.TrackerMOSSE_create
65.                     }
66.                     # επιλογή του επιθυμητού ανιχνευτή
67.                     # αρχικοποίηση του ανιχνευτή
68.             (self.tracker) = OPENCV_OBJECT_TRACKERS[args["tracker"]]()
```

```
69.             self.name = args["tracker"]
```

Στις γραμμές κώδικα 47 εώς 69 γίνεται η αρχικοποίηση του ανιχνευτή που έχει επιλέξει ο χρήστης.

```
70.     def init_frame(self,frame,sel_frame):
71.             #αρχικοποίηση καρέ του ανιχνευτή
72.             (self.tracker).init(frame,sel_frame)
```

Στις γραμμές κώδικα 70 εώς 72 γίνεται η αρχικοποίηση του καρέ του ανιχνευτή.

```
73.     def updateFrame(self,frame,fps):
74.
75.             (H, W) = frame.shape[:2]
76.
77.             centerFrameX = W // 2
78.             centerFrameY = H // 2
79.             frameCenter = [centerFrameX,centerFrameY]
80.
81.             # σχεδιασμός του πλαισίου επιλογής (bounding box)
82.             (success, box) = (self.tracker).update(frame)#επιστροφή τύπου boolean
                                                                όταν γίνει επιτυχία (sucess)
83.             # έλεγχος επιτυχίας ανίχνευσης (success)
84.             if success: #τότε σχεδιάζεται το πλαίσιο
85.                     (x, y, w, h) = [int(v) for v in box]
86.                     cv2.rectangle(frame, (x, y), (x + w, y + h),
87.                             (0, 255, 0), 2)
88.                     objectX = int(x + (w / 2.0))
89.                     objectY = int(y + (h / 2.0))
90.                     objectCenter = [objectX,objectY]
91.
92.             # ενημέρωση του μετρητή καρέ (FPS counter)
93.             if fps is not None:
94.                     fps.update()
95.                     fps.stop()
96.
97.             # αρχικοποίηση πληροφοριών που σχεδιάζονται στο καρέ
98.
99.             info = [
100.                    ("Tracker", self.name),
101.                    ("Success", "Yes" if success else "No"),
102.                    ("FPS", "{:.2f}".format(fps.fps())),
103.            ]
104.            # σχεδιασμός πληροφοριών στο καρέ
105.            for (i, (k, v)) in enumerate(info):
106.                    text = "{}: {}".format(k, v)
107.                    cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
108.                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
```

```
109.        if success:
110.            return (objectCenter,frame,fps)
111.        else:
112.            return (frameCenter,frame,fps)
```

Στις γραμμές κώδικα 73 εώς 112 ορίζεται η μέθοδος που ενημερώνει κάθε καρέ κατά τη διάρκεια της ανίχνευσης ενός αντικειμένου. Σε αυτήν την ενημέρωση περιλαμβάνονται πληροφορίες όπως το αντικείμενο που ανιχνεύεται, ο αριθμός των καρέ ανά δευτερόλεπτο (fps), το όνομα του ανιχνευτή που χρησιμοποιείται και αν είναι επιτυχής η ανίχνευση. Στο τέλος, επιστρέφεται το ενημερωμένο καρέ.

(2) Αρχείο ***modules.py***

Στο αρχείο αυτό, έχουν γραφτεί οι μέθοδοι που αφορούν την επικοινωνία των διεργασιών μέσω σημαφόρων (signal_handler), την ορθή μετακίνηση των σερβομηχανισμών και τον έλεγχο PID, ο οποίος αποφέρει τις ορθές τιμές μετακίνησης στους σερβομηχανισμούς καθώς και την επικοινωνία του συστήματος με τον προσομοιωτή SITL. Ακολουθεί ο κώδικας:

```
1.import pantilthat
2.import time
3.import sys
4.import signal
5.from pymavlink import mavutil
6.import classes as c
```

Στις γραμμές κώδικα 1 εως 5 γίνεται η εισαγωγή των απαραίτητων βιβλιοθηκών. Στη γραμμή 6 εισάγεται το πρώτο αρχείο *classes.py* , ώστε να χρησιμοποιηθεί η κλάση του ελεγκτή PID.

```
7.
8.# μέθοδος για χειρισμό διακοπής (keyboard interupt)
9.def signal_handler(sig, frame):
10.    # εκτύπωση μηνύματος διακοπής
11.    print("[INFO] You pressed `ctrl + c`! Exiting...")
12.    # απενεργοποίηση σερβομηχανισμών
13.    pantilthat.servo_enable(1, False)
14.    pantilthat.servo_enable(2, False)
15.    # έξοδος
16.    sys.exit()
```

Σις γραμμές κώδικα 8 εώς 16 ορίζεται η μέθοδος για τη διαχείρηση του τερματισμού των διεργασιών είτε μέσω των πλήτρων «ctrl + c» είτε μέσω του πλήκτρου «q», όπως περιγράφηκε στο Κεφάλαιο 7.

```
17.#μέθοδος ελέγχου εμβέλειας κίνησης σερβομηχανισμών
18.def in_range(val, start, end):
19.    # έλεγχος εμβέλειας
20.    return (val >= start and val <= end)
```

Στις γραμμές κώδικα 17 εώς 20 ελέγχεται αν η κίνηση που μπορεί να κάνει ένας σερβομηχανισμός, βρίσκεται εντός της εμβέλειας του (+90 μοίρες εώς -90 μοίρες).

```
21.#μέθοδος κίνησης ενός σερβομηχανισμού
22.def set_servos(pan, tlt):
23.    # χειρισμός διακοπής
24.    signal.signal(signal.SIGINT, signal_handler)
25.
26.
27.    # ατέρμων βρόχος
28.    while True:
29.        # οι μοίρες αντιστρέφονται
30.        panAngle = -1 * pan.value
31.        tiltAngle = -1 * tlt.value
32.        # έλεγχος οριζόντιας κίνησης
33.        if in_range(panAngle,-90,90):
34.            pantilthat.pan(panAngle)
35.        # έλεγχος κάθετης κίνησης
36.        if in_range(tiltAngle,-90,90):
37.            pantilthat.tilt(tiltAngle)
```

Στις γραμμές κώδικα 21 εώς 37 εκτελείται η μέθοδος που κινεί τους σερβομηχανισμούς σε συνάρτηση με τη κίνηση του αντικειμένου που ανιχνεύεται, αφού γίνουν αρχικά οι απαραίτητοι έλεγχοι.

```
38.
39.def pid_process(output, p, i, d, objCoord, centerCoord,isPanning):
40.    # χειρισμός διακοπής
41.    signal.signal(signal.SIGINT, signal_handler)
42.    # δημιουργία αντικειμένου PID και αρχικοποίηση
43.    p = c.PID(p.value, i.value, d.value)
44.    p.initialize()
45.
46.    #μεταβλητή για υπολογισμό των μοιρών
47.    previous_angle = 0
48.
49.    """Δημιουργία σύνδεσης Mavlink"""
50.    # Σύνδεση στον προσομοιωτή SITL με χρησιμοποίηση UDP. Η διεύθυνση IP
       μεταβάλεται ανάλογα με τη σύνδεση του υπολογιστή σε έναν δρομολογητή.
51.    connection = mavutil.mavlink_connection('udp:192.168.1.7:14550')
52.    connection.wait_heartbeat()
53.
54.    #μοίρες περιστροφής του τετρακόπτερου
55.    desired_yaw = 25
56.
57.    #αυξομοίωση ύψους τετρακόπτερου σε μέτρα
58.    meters = 0.1
59.    previous_Copter_height = 50 #αρχικό ύψος αιώρησης σε μέτρα
```

```
60.
61.  # ατέρμων βρόχος
62.  while True:
63.      # υπολογισμός σφάλματος
64.      error = centerCoord.value - objCoord.value
65.      # έλεγχος PID και ενημέρωση σφάλματος
66.      output.value = p.update(error) // 100
67.
68       #Υπολογισμός κίνησης σε pixel
69.      previous_angle += (error)
70.
71.      if previous_angle >= 200 and isPanning == False: # Το τετρακόπτερο
         πρέπει να αυξήσει το ύψος του
72.          connection.mav.command_long_send(connection.target_system,
                 connection.target_component,
                 mavutil.mavlink. MAV_CMD_NAV_CONTINUE_
                 AND_CHANGE_ALT,1,0,0,0,0,0,0,
                 (previous_Copter_height+meters))
73.          #Εκτύπωση μηνύματος
74.          print("Copter ascented 0.1 meter")
75.          #Υπολογισμός νέου ύψους
76.          previous_Copter_height += meters
77.          #Μηδενισμός κριτηρίου
78.          previous_angle = 0
79.
80.      elif previous_angle <= -200 and isPanning == True: #Το τετρακόπτερο
         πρέπει να μειώσει το ύψος του
81.          connection.mav.command_long_send(connection.target_system,
                 connection.target_component,
                 mavutil.mavlink. MAV_CMD_NAV_CONTINUE_
                 AND_CHANGE_ALT,2,0,0,0,0,0,0,
                 (previous_Copter_height+meters))
82.          #Εκτύπωση μηνύματος
83.          print("Copter descented 0.1 meter")
84.          #Υπολογισμός νέου ύψους
85.          previous_Copter_height -= meters
86.          #Μηδενισμός κριτηρίου
87.          previous_angle = 0

88.      elif previous_angle >= 200 and isPanning == True: #Το τετρακόπτερο
         πρέπει να περιστραφεί προς τα δεξιά
89.          connection.mav.command_long_send(connection.target_system,
                 connection.target_component,
                 mavutil.mavlink.MAV_CMD_CONDITION_
                 YAW ,0,desired_yaw,25,1,1,0,0,0)
90.          #Εκτύπωση μηνύματος
91.          print("Copter rotated 25 deegrees right ")
92.          #Μηδενισμός κριτηρίου
93.          previous_angle = 0
94.
```

```
95.     elif previous_angle <= -200 and isPanning == True: #Το τετρακόπτερο
        πρέπει να περιστραφεί προς τα αριστερά
96.         connection.mav.command_long_send(connection.target_system,
                connection.target_component,
                mavutil.mavlink.MAV_CMD_CONDITION_
                YAW ,0,desired_yaw,25,-1,1,0,0,0)
97.         #Εκτύπωση μηνύματος
98.         print(" Copter rotated 25 deegrees left ")
99.         #Μηδενισμός κριτηρίου
100.        previous_angle = 0
101.
102.    else:
103.        continue
```

Στις γραμμές κώδικα 39 εώς 103 ορίζεται η μέθοδος, μέσω της οποίας, θα πραγματοποιηθούν δύο λειτουργίες. Η πρώτη λειτουργία αφορά στη δημιουργία ενός αντικειμένου της κλάσης PID, μέσω του οποίου πραγματοποιείται έλεγχος PID (εύρεση σφάλματος μεταξύ κέντρου του καρέ και κέντρου του αντικειμένου που ανιχνεύεται), έτσι ώστε να αποδοθεί η κατάληλλη τιμή, σε μοίρες, κίνησης σε έναν σερβομηχανισμό. Η δεύτερη λειτουργία αφορά την αποστολή σήματος στον προσομειωτή SITL, εφόσων ικανοποιούνται συγκεκριμένες συνθήκες, έτσι ώστε να το τετρακόπτερο που προσομοιώνεται να περιστραφεί κατάληλλα ή να μεταβάλει το ύψος του.

(3) Αρχείο ***pan_tilt_main.py***

Στο αρχείο αυτό, έχουν γραφτεί δύο μέθοδοι. Η πρώτη αφορά την κατασκευή και την λειτουργία του ανιχνευτή αντικειμένου, ο οποίος αποτελεί αντικείμενο της κλάσης ανιχνευτών, τη λειτουργία των πλήκτρων «***s***»,«***q***»,«***c***» και την παροχή εικόνας σε πραγματικό χρόνο. Η δεύτερη λειτουργία αποτελεί το κυρίως πρόγραμμα (main) του συστήματος. Στο πρόγραμμα αυτό και μέσω της βιβλιοθήκης ***multiprocessing***, γίνεται χρήση ενός διαχειριστή (Manager) έτσι ώστε να εκτελεστούν παράληλλα τέσσερις διεργασίες, και να χρησιμοποιηθούν κοινές μεταβλητές μεταξύ αυτών. Οι διεργασίες αυτές είναι οι παρακάτω:

1. **Διεργασία processTracker**: Αφορά τη λειτουργία του ανιχνευτή αντικειμένων, τη παροχή εικόνας σε πραγματικό χρόνο και τη λειτουργία των πλήτρων «s»,«q»,«c».

2. **Διεργασία processPanning:** Αφορά τη λειτουργία του ελεγκτή PID και τη παροχή της κατάληλλης τιμής; στον σερβομηχανισμό που είναι υπεύθυνος για την κινήση σε οριζόντια κατεύθυνση.

3. **Διεργασία processTilting:** Αφορά τη λειτουργία του ελεγκτή PID και τη παροχή της κατάληλλης τιμής; στον σερβομηχανισμό που είναι υπεύθυνος για την κινήση σε κάθετη κατεύθυνση.

4. **Διεργασία processSetServos:** Στη διεργασία αυτή, γίνεται χρήση των τιμών που παράγουν οι διεργασίες (2) και (3), μέσω των ελεγκτών PID, ώστε να κινηθούν οι σερβομηχανισμοί αντίστοιχα.

Ακολυθεί ο κώδικας:

```
1.import classes as c
2.import modules as m
3.import imutils
4.from multiprocessing import Manager
5.from multiprocessing import Process
6.from imutils.video import VideoStream
7.from imutils.video import FPS
8.import pantilthat as pth
9.import signal
10.import time
11.import cv2
```

Στις γραμμές κώδικα 1 και 2 γίνεται η εισαγωγή των αρχείων *classes.py* και *modules.py* ώστε να χρησιμοποιηθούν οι αντίστοιχες και μέθοδοι και η κλάση των ανιχνευτών αντικειμένων. Στις γραμμές 3 εώς και 11 συνεχίζεται η εισαγωγή των απαραίτητων βιβλιοθηκών.

```
12.def obj_tracker(tracker_name,objX, objY, centerX, centerY):
13.     # χειρισμός διακοπής
14.     signal.signal(signal.SIGINT, m.signal_handler)
15.
16.     # Κατασκευή αντικειμένου-ανιχνευτή και αρχικοποίηση
17.     tracker = c.ObjectTracker(tracker_name)
18.
19.     #Αρχικοποίηση του πλαισίου επιλογής αντικειμένου (bounding box)
20.     initBB = None
21.
22.     #Εκτύπωση μηνύματος έναρξης ροής βίντεο
23.     print("[INFO] starting video stream...")
24.     # Έναρξη ροής βίντεο, αναμονή 2 sec για την ορθή λειτουργία της κάμερας.
25.     vs = VideoStream(usePiCamera=True,framerate=30).start()
26.     time.sleep(2.0)
27.
28.     # Αρχικοποίηση μετρητή των καρέ
29.     fps = None
30.
31.     # Ατέρμων βρόχος
32.     while True:
33.             # Αναστροφή καρέ κάμερας κάθετα
34.             # ώστε να μην είναι ανάποδα
35.             frame = vs.read()
36.             frame = cv2.flip(frame, 0)
37.             if frame is None:
38.                     break
39.
40.             # Υπολογισμός κέντρου του καρέ
41.             # Ρύθμιση μεγέθους του καρέ για καλύτερη επεξεργασία
```

```
42.             frame = imutils.resize(frame, width=500)
43.             (H, W) = frame.shape[:2]
44.             centerX.value = W // 2
45.             centerY.value = H // 2
46.
47.             #Ελεγχος για ύπαρξη πλαισίου επιλογής και υπολογισμός κέντρου
48.             #αντικειμένου
49.             if initBB is not None:
50.                     objectLoc = tracker.updateFrame(frame,fps)
51.                     objectCenter,frame,fps = objectLoc
52.                     objX.value = objectCenter[0]
53.                     objY.value = objectCenter[1]
54.             else:
55.                     objX.value = centerX.value
56.                     objY.value = centerY.value
57.
58.             #Εμφάνιση καρέ στην οθόνη
59.             cv2.imshow("Pan tilt tracking", frame)
60.
61.             #Αναμονή επιλογής πλήκτρου
62.             key = cv2.waitKey(1) & 0xFF
63.
64.             # Επιλογή πλαισίου επιλογής με το πλήκτρο «s»
65.             if key == ord("s"):
66.                     initBB = cv2.selectROI("Pan tilt tracking", frame,
67.                             fromCenter=False,showCrosshair=True)
68.                     # Έναρξη ανιχνευτή της βιβλιοθήκης OpenCV και μετρητή καρέ
69.                     tracker.init_frame(frame,initBB)
70.                     fps = FPS().start()
71.
72.             # Επανεπιλογή πλαισίου επιλογής με το πλήκτρο «c»
73.             elif key == ord("c"):
74.                     initBB = cv2.selectROI("Pan tilt tracking", frame,
75.                             fromCenter=False,showCrosshair=True)
76.                     tracker = c.ObjectTracker(tracker_name)
77.                     tracker.init_frame(frame,initBB)
78.
79.             # Τερματισμός διεργασιών με το πλήκτρο «q» και έξοδος
80.             elif key == ord("q"):
81.                     # κλείσιμο των παραθύρων
82.                     cv2.destroyAllWindows()
83.                     signal.signal(signal.SIGINT, m.signal_handler)
84.                     break
```

Στις γραμμές κώδικα 12 εώς και 84 γίνεται η έναρξη της ροής βίντεο σε πραγματικό χρόνο, η έναρξη και η λειτουργία του ανιχνευτή αντικειμένων σε συνάρτηση με τη χρήση των πλήκτρων «s», «c» και «q».

```
85.     #Ελεγχος για το κύριο πρόγραμμα main
86.     if __name__ == "__main__":
```

```
87.     #Εκκίνηση διαχειριστή διεργασιών και κοινών  μεταβήτών
88.     with Manager() as manager:
89.         # Ενεργοποίηση σερβομηχανισμών
90.         pth.servo_enable(1, True)
91.         pth.servo_enable(2, True)
92.         # Τοποθέτηση ακεραίων τιμών για το κέντρο (x,y) του καρέ
93.         #(center) και για το κέντρο του αντικειμένου (obj)
94.         centerX = manager.Value("i", 0)
95.         centerY = manager.Value("i", 0)
96.
97.         objX = manager.Value("i", 0)
98.         objY = manager.Value("i", 0)
99.
100.        #Οι τιμές pan και tilt ενημερώνονται μετά από έλεγχο PID και
101.        #καθορίζουν τη κίνηση των σερβομηχανισμών
102.        pan = manager.Value("i", 0)
103.        tlt = manager.Value("i", 0)
104.
105.        # Αρχικοποίηση όρων KP,KI,KD για τους δύο ελεγκτές
106.        panP = manager.Value("f", 0.09)
107.        panI = manager.Value("f", 0.06)
108.        panD = manager.Value("f", 0.002)
109.
110.        tiltP = manager.Value("f", 0.09)
111.        tiltI = manager.Value("f", 0.06)
112.        tiltD = manager.Value("f", 0.002)
113.
114.        #Ορισμός τεσσάρων διεργασιών όπως περιγράφηκαν αρχικά
115.        processTracker  =  Process(target=obj_tracker,args=("kcf",objX,
116.            objY, centerX, centerY))
117.        processPanning = Process(target=m.pid_process,
118.            args=(pan, panP, panI, panD, objX, centerX,True))
119.        processTiltin g = Process(target=m.pid_process,
120.            args=(tlt, tiltP, tiltI, tiltD, objY, centerY,False))
121.        processSetServos = Process(target=m.set_servos, args=(pan, tlt))
122.
123.        # Έναρξη τεσσάρων διεργασιών
124.        processTracker.start()
125.        processPanning.start()
126.        processTilting.start()
127.        processSetServos.start()
128.
129.        # Ένωση τεσσάρων διεργασιών
130.        processTracker.join()
131.        processPanning.join()
132.        processTilting.join()
133.        processSetServos.join()
```

Στις γραμμές κώδικα 85 εώς και 113 γίνεται η εκτέλεση του κυρίου προγράμματος με τις τέσσερις διεργασίες, μέσω του διαχειριστή (manager). Οι μεταβλητές value είναι κοινόχρηστες από όλες τις διεργασίες.

Η εκτέλεση του αρχείου pan_tilt_main.py σε ένα τερματικό, θέτει σε λειτουργία το σύστημα αυτόματης αναγνώρισης αντικειμένων.

9 Επίλογος

Η τεχνητή νοημοσύνη και κατ'επέκταση η μηχανική μάθηση, αποτελούν την αιχμή της τεχνολογίας στην εποχή μας, βρίσκοντας εφαρμογή σε πολλούς τομείς και σε διάφορα συστήματα. Η εκμετάλλευση της αρχιτεκτονικής του transformer, όπως βλέπουμε στις μέρες μας, από την OpenAI και όχι μόνο, εκπλήσσει καθημερινά την κοινωνία μέσω νέων εφαρμογών που χρησιμοποιούν τα νευρωνικά δίκτυα. Η ραγδαία αυτή εξέλιξη δεν αποκλείει το ενδεχόμενο να δούμε πολύ σύντομα ακόμα και ρομποτικούς βοηθούς στις οικίες των ανθρώπων, χρησιμοποιώντας με αυτόν τον τρόπο την τεχνητή νοημοσύνη σε ατομικό και προσωπικό επίπεδο. Ακόμη, φαίνεται η χρήση των συστηματων αυτομάτου ελέγχου να χρησιμοποιείται ευρέως και να κρίνεται αναγκαία για την δημιουργία «έξυπνων» συστημάτων και εφαρμογών. Το σύστημα αυτόματης αναγνώρισης αντικειμένων που αναπτύχθηκε στην παρούσα διπλωματική εργασία, ανήκει στα προαναφερθέντα συστήματα, κάνοντας χρήση την βιβλιοθήκη λογισμικού OpenCv, και μπορεί να αποτελέσει ένα σημαντικό εργαλείο για χρήση σε πολλές εφαρμογές και σε διάφορους τομείς. Ιδιαίτερα, η τοποθέτησή του επί των μη επανδρωμένων αεροσκαφών δίνει τη δυνατότητα μιας αυτόνομης πτήσης, όπου η ανίχνευση ενός αντικειμένου επιτυγχάνεται διαρκώς, παρέχοντας συνεχή πληροφόρηση επί αυτού, χωρίς να κρίνεται αναγκαία η παρέμβαση από τον χειριστή του συστήματος. Τέλος, οι μελλοντικές του επεκτάσεις μπορούν να αυξήσουν τόσο τη λειτουργία του όσο και τη χρήση του.

10 Βιβλιογραφία

[1] Τεχνητή Νοημοσύνη, μια σύγχρονη προσέγγιση, 2η Έκδοση, 2005, Εκδόσεις Κλειδάριθμος, Stuart Russell & Peter Norvig, ISBN: 960-209-873-2

[2] Τεχνητή Νοημοσύνη - Δ' Έκδοση, 2020, Εκδόσεις Πανεπιστημίου Μακεδονίας, Βλαχάβας, Κεφαλάς, Βασιλειάδης, Κόκκορας, Σακελλαρίου, ISBN 978-618-5196-44-8

[3] Marc Sollinger, January 5, 2018, *Garry Kasparov and the game of artificial intelligence,* Διαθέσιμο στον ιστότοπο: https://theworld.org/stories/2018-01-05/garry-kasparov-and-game-artificial-intelligence

[4] Νευρωνικά Δίκτυα και Μηχανική Μάθηση, 3η Έκδοση, 2010, Εκδόσεις Παπασωτηρίου, Simon Haykin, ISBN: 960-7182-64-2

[5] OpenCv, Διαθέσιμο στον ιστότοπο: https://opencv.org/

[6] Adrian Rosebrock, July 30, 2018, *OpenCv Object Tracking*, Διαθέσιμο στον ιστότοπο: https://pyimagesearch.com/2018/07/30/opencv-object-tracking/

[7] Satya Mallick, February 13, 2017, *Object Tracking using OpenCv (C++/Python)*, Διαθέσιμο στον ιστότοπο: https://learnopencv.com/object-tracking-using-opencv-cpp-python/

[8] Raspberry Pi, Διαθέσιμο στον ιστότοπο: https://opensource.com/resources/raspberry-pi

[9] Pan - Tilt HAT Pimoroni, Διαθέσιμο στον ιστότοπο: https://shop.pimoroni.com/products/pan-tilt-hat?variant=33704345034

[10] Raspberry Pi Camera Module 2, Διαθέσιμο στον ιστότοπο: https://www.raspberrypi.com/products/camera-module-v2/

[11] Tim, February 16, 2023, *Pan-Tilt Hat with Raspberry Pi - Quick Start Guide*, Διαθέσιμο στον ιστότοπο: https://core-electronics.com.au/guides/pan-tilt-hat-raspberry-pi/

[12] ΑΙΣΘΗΤΗΡΕΣ ΜΕΤΡΗΣΗΣ ΚΑΙ ΕΛΕΓΧΟΥ, 3η Έκδοση, 2018, Εκδόσεις ΤΖΙΟΛΑ , Δρ. Κωνσταντίνος Καλαβρέκτης, Δρ. Νικόλαος Κατέβας, ISBN: 978-960-418-758-4

[13] Adrian Rosebrock, April 1, 2019, *Pan/tilt face tracking with a Raspberry Pi and OpenCv,* Διαθέσιμο στον ιστότοπο: https://pyimagesearch.com/2019/04/01/pan-tilt-face-tracking-with-a-raspberry-pi-and-opencv/

[14] Moschetta, Jean-Marc and Namuduri, Kamesh, Introduction to UAV Systems, (2017) In: UAV Networks and Communications. Cambridge University Press, Cambridge, 1-25. ISBN 9781316335765, Διαθέσιμο στον ιστότοπο: https://oatao.univ-toulouse.fr/19225/

[15] Μαγκίρης, Βαγγέλης: «UAV και MAV: Τα ιπτάμενα ρομπότ», Περισκόπιο της Επιστήμης, τεύχος 219 (Ιούλιος-Αύγουστος 1998)

[16] Ardupilot, *Pixhawk Overview,* Διαθέσιμο στον ιστότοπο: https://ardupilot.org/copter/docs/common-pixhawk-overview.html

[17] Ardupilot, *SITL Simulator (Software in the loop)*, Διαθέσιμο στον ιστότοπο: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[18] MAVLINK, *MAVLink Developer Guide*, Διαθέσιμο στον ιστότοπο: https://mavlink.io/en/

[19] ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ, 9η Έκδοση, 2013, Εκδόσεις Μ.Γκιούρδας , Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, ISBN: 978-960-512-651-3

[20] Intel Movidius - Vision Processing Units (VPUs), Διαθέσιμο στον ιστότοπο: https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html

[21] Adrian Rosebrock, September 19, 2018, *pip install OpenCv*, Διαθέσιμο στον ιστότοπο: https://pyimagesearch.com/2018/09/19/pip-install-opencv/?_ga=2.136927609.89142617.1638719522-1545978548.1616802209