Δ Ι Ε Θ Ν Ε Σ Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Τ Η Σ Ε Λ Λ Α Δ Ο Σ
Σ Χ Ο Λ Η Μ Η Χ Α Ν Ι Κ Ω Ν
Τ Μ Η Μ Α Μ Η Χ Α Ν Ι Κ Ω Ν Π Λ Η Ρ Ο Φ Ο Ρ Ι Κ Η Σ ,
Υ Π Ο Λ Ο Γ Ι Σ Τ Ω Ν Κ Α Ι Τ Η Λ Ε Π Ι Κ Ο Ι Ν Ω Ν Ι Ω Ν
Π Ρ Ο Γ Ρ Α Μ Μ Α Μ Ε Τ Α Π Τ Ι Χ Ι Α Κ Ω Ν Σ Π Ο Υ Δ Ω Ν
Σ Τ Η Ρ Ο Μ Π Ο Τ Ι Κ Η

ΚΑΤΑΣΚΕΥΗ ΣΤΑΘΜΟΥ ΕΝΤΟΠΙΣΜΟΥ ΠΥΡΚΑΓΙΩΝ

Πτυχιακή Εργασία του

Κατσάνη Κωνσταντίνου (44)

Επιβλέπων: Στ. Βολογιαννίδης, Επίκουρος Καθηγητής

ΣΕΡΡΕΣ, ΣΕΠΤΕΜΒΡΙΟΣ 2023

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

Περίληψη

Ο σκοπός της διπλωματικής εργασίαςπου εκπονείται στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών του Διεθνούς Πανεπιστημίου Ελλάδας , σχετίζεται με την δημιουργία ενός αυτόνομου σταθμού που θα τροφοδοτεί μια απομακρυσμένη ηλεκτρονική πλατφόρμα με πληροφορίες μικροκλίματος αλλά και εικόνας μιας περιοχής ούτως ώστε να μπορεί αυτή να παρακολουθείται για εκδηλώσεις πυρκαγιών. Επιπλέον έχει δημιουργηθεί όλο το υπόβαθρο ώστε να μπορεί να εκτελείται ένα νευρωνικό δίκτυο εξ` αποστάσεως συνδεδεμένο με το σταθμό για αυτοματοποιημένο εντοπισμό πυρκαγιάς θέτοντας στη διάθεση του όλα τα στοιχεία που συλλέγονται. Πιο συγκεκριμένα για την ολοκλήρωση της ανατιθέμενης εργασίας χρησιμοποιήθηκαν οι γλώσσες προγραμματισμού python3.9, ipython, c++(Arduino). Επίσης οι μικροελεγκτές και μικροϋπολογιστές που χρησιμοποιήθηκαν είναι οι Raspberry Pi 4b και Arduino Uno.

English Summary

The purpose of our Diploma thesis that has been created for Interinstitutional Postgraduate Program (MSc) in Robotics for International Hellenic University, is about creating an autonomous station that will deliver to aranged electronic platform microclimate information, but also a picture of an area in order to monitor for fire events. In addition, all the background has been created so a neutral network can be executed from distance, connected to the autonomous station and have available all the collected data in order to achieve an automated fire detection.In detail the programming languages that we have used are python3.9, ipython, c++(Arduino). In addition the microcontrollers and microcomputers that we have used are a RaspberryPi 4b and a Arduino Uno.

Περιεχόμενα

Πίνακας εικόνων

Πίνακας πινάκων

Εισαγωγή

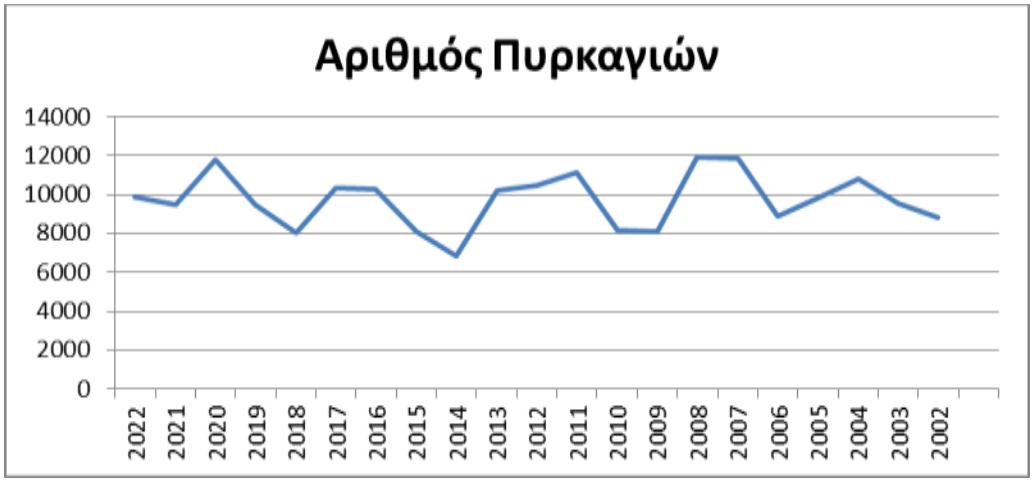
Αναλογιζόμενοι τον ρυθμό εξέλιξης της τεχνολογίας, την μείωση της ταχύτητας απόκρισης συστημάτων, την δυνατότητα αυτοματισμού λήψης αποφάσεων και εκτίμησης κατάστασης, θα έπρεπε προοδευτικά να μπορούσαμε να αντιμετωπίζουμε το πρόβλημα της κατάσβεσης των δασικών πυρκαγιών ολοένα και πιο αποτελεσματικά μειώνοντας τις απώλειες με το πέρασμα των χρόνων.

Δυστυχώς κάτι τέτοιο δεν συμβαίνει και τα στατιστικά στοιχεία μας δείχνουν μια σταθερότητα στην 'αποτελεσματικότητα' των δασικών πυρκαγιών. Πιο συγκεκριμένα χρησιμοποιώντας τα στατιστικά στοιχεία που μας παρέχει το Πυροσβεστικό Σώμα Ελλάδας [17] έχοντας συγκεντρώσει το σύνολο των καμένων δασικών εκτάσεων ανά έτος είναι φανερή αυτή σταθερότητα για τα τελευταία 20 χρόνια όπως περιγράφεται και στον παρακάτω πίνακα και των αντίστοιχων γραφημάτων

Έτος	Αριθμός Πυρκαγιών	ΚΑΜΕΝΗ ΕΚΤΑΣΗ (Σε στρέμματα)
2022	9856	284963
2021	9514	1332140
2020	11799	222154
2019	9500	162759
2018	8006	193815
2017	10356	231322
2016	10263	420012
2015	8118	170858
2014	6834	193192
2013	10196	270919
2012	10458	491381
2011	11144	346802
2010	8179	150798
2009	8124	449161
2008	11908	386814
2007	11868	2644083
2006	8925	153785

2005	9827	88734
2004	10798	134572
2003	9539	84765
2002	8835	81744

Πίνακας 1Ετήσιος αριθμός πυρκαγιών και καμένων εκτάσεων σε στρέμματα



Εικόνα 1 Ετήσιος αριθμός πυρκαγιών

Εικόνα 2 Ετήσιος αριθμός καμένων εκτάσεων σε στρέμματα

Δυστυχώς το 2023 είχαμε μια τεράστια καταστροφή που αν και είναι λάθος να αναφέρουμε από στατιστικής άποψης μεμονωμένα ένα έτος γίνεται φανερή η αδυναμία των ποσοτικών και ποιοτικών μέσων κατάσβεσης που έχουμε διαθέσιμα.

Για το έτος 2023 και συγκρινόμενο με τον μέσο όρο καμένων εκτάσεων των ετών 2002 – 2022 είχαμε αύξηση 195% με σύνολο 1.281.480 στρέμματα καμένων εκτάσεωνόπως αυτά προκύπτουν από την ανάλυση των δεδομένων του Ευρωπαϊκού Συστήματος Πληροφόρησης για Δασικές Πυρκαγιές [18] που επεξεργάστηκε η Πυρομετεωρολογική Ομάδα FLAME της μονάδας ΜΕΤΕΟ του Εθνικού Αστεροσκοπείου Αθηνών (ΕΑΑ). Είναι φανερό ότι σε ακραίες συνθήκες η αποτελεσματικότητα του πυροσβεστικού σώματος και των εθελοντών δεν είναι αρκετή ώστε να περιορίσουν-κατασβήσουν την πυρκαγιά πράγμα που γεννά την ανάγκη ενός διαφορετικού τρόπου αντιμετώπισης που θα συμμετέχει σε όλα τα στάδια της, από τον έγκαιρο εντοπισμό της , την ενημέρωση των μονάδων κατάσβεσης, την πληροφόρηση στοιχείων για την αντιμετώπιση της και έλεγχο για αναζωπυρώσεις.

Το πρόβλημα το οποίο καλούμαστε να εξετάσουμε είναι η πρόληψη μιας πυρκαγιάς αλλά και η συνεισφορά μας στην κατάσβεση της, παρέχοντας πληροφορίες για την οργάνωση της στρατηγικής της κατάσβεσης της. Επιπλέον έγινε προσπάθεια δημιουργίας μια ηλεκτρονικής πλατφόρμας ούτως ώστε να ενδυναμωθεί η αποτελεσματικότητα και ο έγκυρος εντοπισμός μιας πυρκαγιάς από ένα νευρωνικό δίκτυο που θα μπορεί να εκτελείται σε αυτή τη πλατφόρμα σε έναν απομακρυσμένο υπολογιστή, συνδυάζοντας τα ευρήματα των αισθητήρων με την σάρωση της εικόνας.

Το κύριο όμως πρόβλημα που έχει δημιουργήσει την ανάγκη για μια τέτοια κατασκευή είναι :

- η κλιματική αλλαγή που ολοένα αντιμετωπίζουμε ραγδαίες μεταβολές της θερμοκρασίας και αύξησης του εύρους τιμών της σε συνδυασμόμε την διάρκεια αυτών των μεταβολών.
- Η αδυναμία επαρκούς επάνδρωσης του πυροσβεστικού σώματος προσαρμοσμένη στις σύγχρονες ανάγκες.
- Η μη προμήθεια σύγχρονων και επαρκών πυροσβεστικών μέσων.
- Η μη δημιουργία ζωνών πυρόσβεσης για τον περιορισμό της πυρκαγιάς που έχει εκδηλωθεί.

Όλα τα παραπάνω δημιουργούν την ανάγκη ενός γρήγορου εντοπισμού ώστε η αντιμετώπιση να μπορεί να γίνει με μικρότερο σε αριθμό προσωπικό και μέσα αλλά

και να περιορίζεται άμεσα. Επιπλέον η παρακολούθηση των περιοχών να μπορεί να γίνει από μία βάση ελέγχου που θα μπορεί να σπεύσει σε όλες τις περιοχές που παρακολουθεί.

Η δομή της εργασίας χωρίζεται σε τρία μέρη, την δομή και τα τεχνικά χαρακτηριστικά του συστήματος, το πώς κατασκευάστηκε δηλαδή ο σκελετός στήριξης και προστασίας των ηλεκτρονικών συστημάτων μας αλλά και τα τεχνικά χαρακτηριστικά των υλικών που χρησιμοποιήσαμε. Στο δεύτερο μέρος περιγράφεται η συνδεσμολογία των ηλεκτρονικών συστημάτων και τέλος στο τρίτο μέρος ο κώδικας που αναπτύξαμε και οι ρυθμίσεις και αυτοματισμοί που κάναμε ώστε να λειτουργεί και να επαναφέρεται σε λειτουργία το σύστημά μας χωρίς επέμβαση σε αυτό από τον άνθρωπο.

Επιπλέον προστέθηκε ένα ακόμη μέρος που επισημαίνονται βελτιώσεις που θα μπορούσαν να γίνουν, προσθήκες και διαφορετικές επιλογές ηλεκτρονικών συστημάτων, που όμως όσο και αν έγινε προσπάθεια βελτίωσης της υπάρχουσας κατασκευής δεν ήταν δυνατόν να συμβαδίσει αυτή η προσπάθεια με τον όγκο και το εύρος των νέων επιλογών.

Φυσικά μελετώντας τη διεθνή βιβλιογραφία και προσπαθώντας να κάνουμε την αναζήτησή μας πιο συγκεκριμένη και ταυτόσημη με την δική μας προοπτική,καταφέραμε να βρούμε μεγάλο όγκο αποτελεσμάτων σε αναζητήσεις που αφορούσαν μετρολογικούς σταθμούς [15], πράγμα που πρακτικά δεν απέχει από αυτό που κάνουμε, επιπλέον όλα τα αποτελέσματα της Iot Based Weather Station Using Raspberry Pi 3 των P Y Muckκαι M J Homam[16] μεταδίδονται μέσω διαδικτύου επίσης. Η διαφοροποίηση του δικού μας συστήματος είναι η πλατφόρμα του jupyternotebookπου μας δίνει την δυνατότητα πλήρους ελέγχου του RaspberryPi με απλοποιημένη την δυνατότητα συγγραφής και εκτέλεσης κώδικα.

1. ΣΚΟΠΙΜΟΤΗΤΑ & ΠΡΟΔΙΑΓΡΑΦΕΣ

Το συγκεκριμένο σύστημα μπορεί να βρει εφαρμογή σε οποιαδήποτε προσπάθεια πρόληψης πυρκαγιάς σε ορεινή και δασική έκταση, σε παρατηρητήρια της πυροσβεστικής και σε οποιοδήποτε σημείο υπάρχει οπτική επαφή με την περιοχή που θέλουμε να παρακολουθήσουμε – προστατεύσουμε.

1.1. Σκοπός

Ο σκοπός του συστήματος είναι να δώσει τα εφόδια σε ένα νευρωνικό δίκτυο ώστε να γίνει πιο αποδοτικό λαμβάνοντας υπόψη τις συνθήκες μικροκλίματος που θα επηρεάζουν την ευαισθησία του αλλά και να δώσει τις απαραίτητες πληροφορίες, σε περίπτωση πυρκαγιάς, στους κατασβέστες της.

1.2. Προδιαγραφές του συστήματος

Αυτό που πετυχαίνει το τελικό σύστημα είναι να επικοινωνήσει στον server τις τιμές που συλλέγουν οι αισθητήρες μας (θερμοκρασία, υγρασία, βαρομετρική πίεση) μέσω του αισθητήρα BME 280[1], την ταχύτητα του αέρα μέσω του ανεμόμετρου που κατασκευάστηκε και την χρήση του αισθητήρα Infrared Sensor TCRT5000[2], μαζί με μια φωτογραφία όπου μπορούμε να την εισάγουμε σε οποιοδήποτε νευρωνικό δίκτυο μέσα στο Jupyter Notebook[3] προσθέτοντας μια μεταβλητή που αλλάζει την ευαισθησία του νευρωνικού επηρεαζόμενη από τις καιρικές συνθήκες και το πώς αυτές συμβάλλουν στην εκδήλωση μιας πυρκαγιάς.

Η πλειονότητα των σημείων ενδιαφέροντος βρίσκονται σε δυσπρόσιτες και απομακρυσμένες περιοχές που στη πλειοψηφία τους δεν υπάρχει παροχή ρεύματος, πόσο μάλλον και σταθερή σύνδεση στο διαδίκτυο, με αποτέλεσμα η αυτονομία του συστήματος και η όσο το δυνατόν λιγότερη ανάγκη για συντήρηση να είναι μονόδρομος. Έτσι τοποθετήθηκε ένα ηλιακό πάνελ συνδεδεμένο σε έναν controller

και στην συνέχεια με μια μπαταρία όπου φορτίζεται από αυτό ενώ ταυτόχρονα ο controller παρέχει την απαραίτητη ενέργεια στο σύστημά μας.
Λόγω της πληθώρας τέτοιων σημείων έγινε προσπάθεια το συνολικό κόστος να είναι σχετικά χαμηλό και να αγγίζει τα 200€.

Το κύριο πλεονέκτημα του συστήματος, σύμφωνα με τις πιθανότητες πρόκλησης πυρκαγιάς, είναι να εκδηλωθούν τους καλοκαιρινούς μήνες που όμως ταυτόχρονα για ένα σύστημα που η πηγή ενέργειάς του είναι ο ήλιος το καθιστά ιδανικό για αυτόν τον σκοπό, αφού αυτή την περίοδο η διάρκεια της ημέρας, η ένταση της ηλιακής ακτινοβολίας αλλά και η ηλιοφάνεια είναι στις μέγιστες τιμές . Το κύριο υλικό από το οποίο αποτελείται το σύστημα μας είναι το ξύλο το οποίο έχει την ικανότητα να μην μεταδίδει την θερμότητα πετυχαίνοντας την προστασία των ηλεκτρικών μας κυκλωμάτων από την υπερθέρμανσή τους.

Αξιολογώντας την μετάδοση κίνησης στο ηλιακό πάνελ μπορούμε να πούμε με ασφάλεια ότι έχει συρρικνωθεί αρκετά το ενεργειακό κόστος αφού με την χρήση γραναζιών και την αξιοποίηση ενός ελεύθερα περιστρεφόμενου άξονα πετυχαίνεται με μόλις έναν σερβοκινητήρα..

2. ΔΟΜΗ ΚΑΙ ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

2.1. Σκελετός και επιφάνεια στήριξης

Η κατασκευή του σκελετού και των επιφανειών στήριξης είναι πολύ συμαντική για την σωστή λειτουργεία του συστήματός μας αφού είναι αυτά που το προστατεύουν από την θερμοκρασία την υγρασία , τον άνεμο και γενικά οτιδήποτε θα μπορούσε να συμβεί. Επίσης λάβαμε υπόψη τον αντίκτιπο που θα έχει στο περιβάλλον και την αντοχή του στο χρόνο.

Το σύνολο του συστήματος μας βασίζεται σε ένα κουτί που η μπροστινή του πλευρά κατασκευάστηκε από πλεξιγκλάς ούτως ώστε να μπορεί να καταγράφει με την κάμερα αλλά και να έχουμε οπτική επαφή με τα ηλεκτρονικά μέρη του, ενώ οι υπόλοιπες πλευρές του κουτιού από Κόντρα πλακέ σημύδας 6 χιλιοστών. Το είδος του ξύλου δεν επιλέχθηκε τυχαία, το κόντρα πλακέ συμύδας θεωρείτε από τα πάνελ με το λιγότερο αποτύπωμα επιβάρυνσης του περιβάλλοντα χώρου αφού είναι φτιαγμένο από λεπτά φύλλα φυσικού ξύλου ‘καπλαμάδες’ κολλημένα μεταξύ τους με φυτική κόλλα ‘MR’ κάνοντας τα ικανά να χρησιμοποιηθούν ακόμη και για παιδικά παιχνίδια.Ταυτόχρονα η αντοχή που έχει είναι αρκετά μεγαλύτερη από αυτή του φυσικού ξύλου σε όλους τους άξονες, σε αντίθεση με το φυσικό ξύλο που είναι ισχυρό σε δύο.

Η επιφάνεια τύπου plexiglass είναι φτιαγμένη από χυτό ακρυλικό με υψηλή αντοχή στην θερμοκρασία πάχους 3 χιλιοστών, η επιλογή του υλικού έγινε και εδώ με γνώμονα το αποτύπωμα επιβάρυνσης του περιβάλλοντος ενώ το πάχος λόγω του τρόπου της κατασκευής δεν μας περιορίζει από άποψη αντοχής να είναι στα 3 χιλιοστά αφού η στήριξή του γίνεται από το ξύλο.

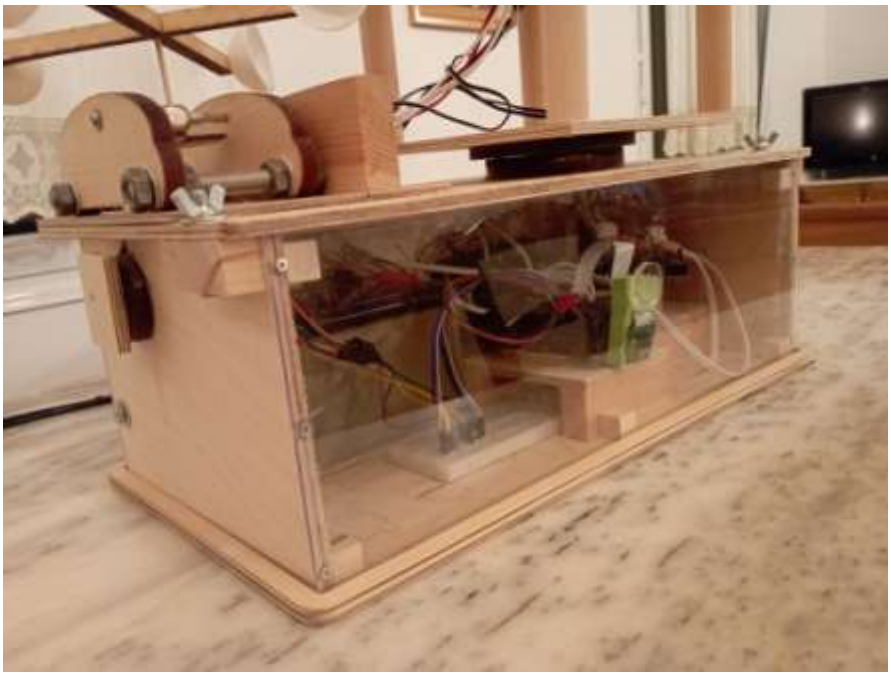
Οι διαστάσεις του κουτιού είναι 45 εκατοστά το πλάτος, 25 εκατοστά το μήκος και 16εκατοστά το ύψος δίνοντάς μας αρκετό χώρο ώστε να τοποθετηθούν τα ηλεκτρονικά στοιχεία του συστήματός μας.

Το πάνω μέρος «καπάκι» του είναι αφαιρούμενο για να έχουμε πρόσβαση και πάνω σε αυτό έχει τοποθετηθεί το ανεμόμετρο με τους αισθητήρες του και μία

περιστρεφόμενη βάση.Στη μία πλαϊνή μεριά έχει γίνει μία τρύπα πού καταλήγει σε ένα μικρό κουτί, στο κάτω μέρος του όποιου ανοίξαμε ακόμη μια τρύπα και τοποθετήσαμε τον αισθητήρα Bme280 για να έχει πρόσβαση στην θερμοκρασία, υγρασία και βαρομετρική πίεση του περιβάλλοντος, ταυτόχρονα όμως και να προστατεύεται από την σώρευση νερού από τις βροχοπτώσεις.

Η στίριξη των ξύλινων πλευρών και της βάσης έγινε μεξύλινους κύβους από φυσικό ξύλο τοποθετημένο έτσι ώστε να αξιοποιήσουμε τις δύο από τις 3 επιφάνειες που έχουμε την δυνατότητα να χρησιμοποιήσουμε βίδες ως μέσω πρόσδεσης. Η ακρυλική επιφάνεια προσδέθηκε με βίδες 2 χιλιοστών στο πλαϊνό μέρος του Κόντρα πλακέ.

Στις Παρακάτω εικόνες μπορούμε να δούμε τα σημεία που έχουμε αναφέρει ως τώρα.



Εικόνα 3Βάσηστήριξης συστήματος

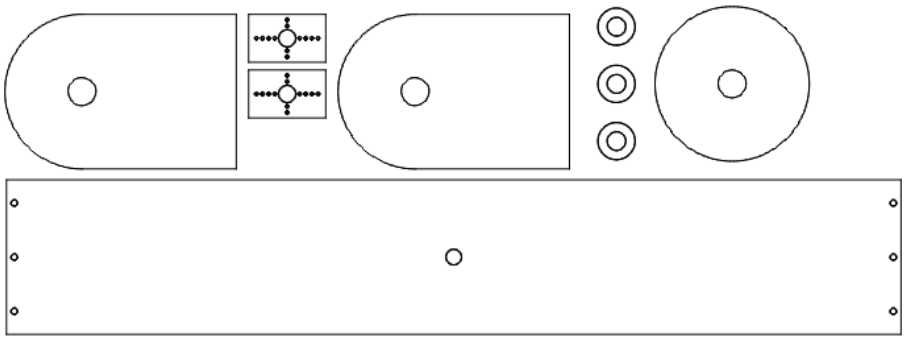
Εικόνα 4 ακρυλική πρόσοψη

Εικόνα 5προσθήκη τοποθέτησης BME280

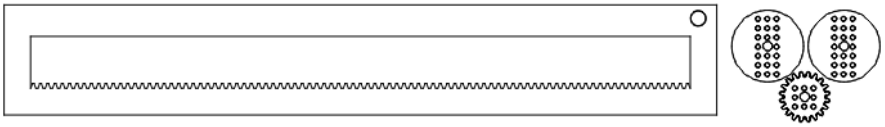
Η περιστρεφόμενη βάση που τοποθετήθηκε στην κορυφή της βάσης του συστήματός μας, σκοπό έχει να τοποθετηθεί πάνω της το ηλιακό πάνελ καθώς και οι σερβοκινητήρες και οι αισθητήρες ανίχνευσης φωτός. Έχει την δυνατότητα να περιστρέφεται 90 μοίρες ώστε το πάνελ να μπορεί να αξιοποιεί την ηλιακή ενέργεια από την ανατολή ως και την δύση του ηλίου ανεξάρτητα την φορά που θα τοποθετηθεί η βάση ώστε να καταγράφει η κάμερα την περιοχή ενδιαφέροντος.

Επάνω στην βαση έχει γίνει μία σταθερή κατασκευή από ξύλινους άξονες οξιάς (λόγω της πυκνότητας της θεωρείται από τους πιο ισχυρούς ξύλινους άξονες) διαμέτρου 25χιλιοστών τοποθετημένες πάνω σε δύο επιφάνειες κόντρα πλακέ σημύδας 6 χιλιοστών (εικόνα 7). Αυτή η κατασκευή μας δίνει την κατάλληλη απόσταση ώστε να μην εμποδίζεται η κίνηση του πάνελ από την βάση.

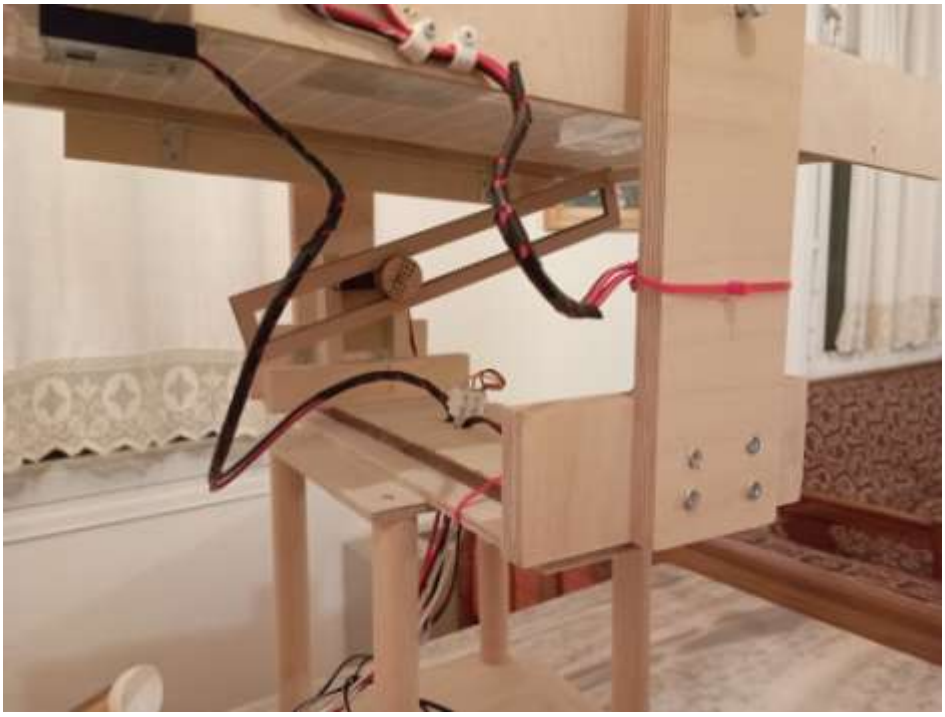
Στο επόμενο επίπεδο έχουμε δημιουργήσει την βάση στήριξης του πάνελ που είναι μία παραλληλόγραμμη επιφάνεια που καταλήγει σε δύο προεκτάσεις που βιδώσαμε το πάνελ με μηχανικές βίδες που δεν έχουν σπείρωμα σε όλη τους την επιφάνεια ώστε να μην προσκολλάται η περιστροφή του πάνελ (εικόνα 4). Στην επιφάνεια αυτή τοποθετήσαμε τον σερβοκινητήρα και στο κινούμενο μέρος του προσδέσαμε ένα γρανάζι που ακολουθεί μια διαδρομή ενός ευθύγραμμου γραναζιού που η άκρη του τοποθετήθηκε στην άκρη του πάνελ με την βοήθεια ενός ξύλινου άξονα 6χιλιοστών με την δυνατότητα ελεύθερης κίνησης (εικόνα 5,6).



Εικόνα 6 Βάση ηλιακού πάνελ



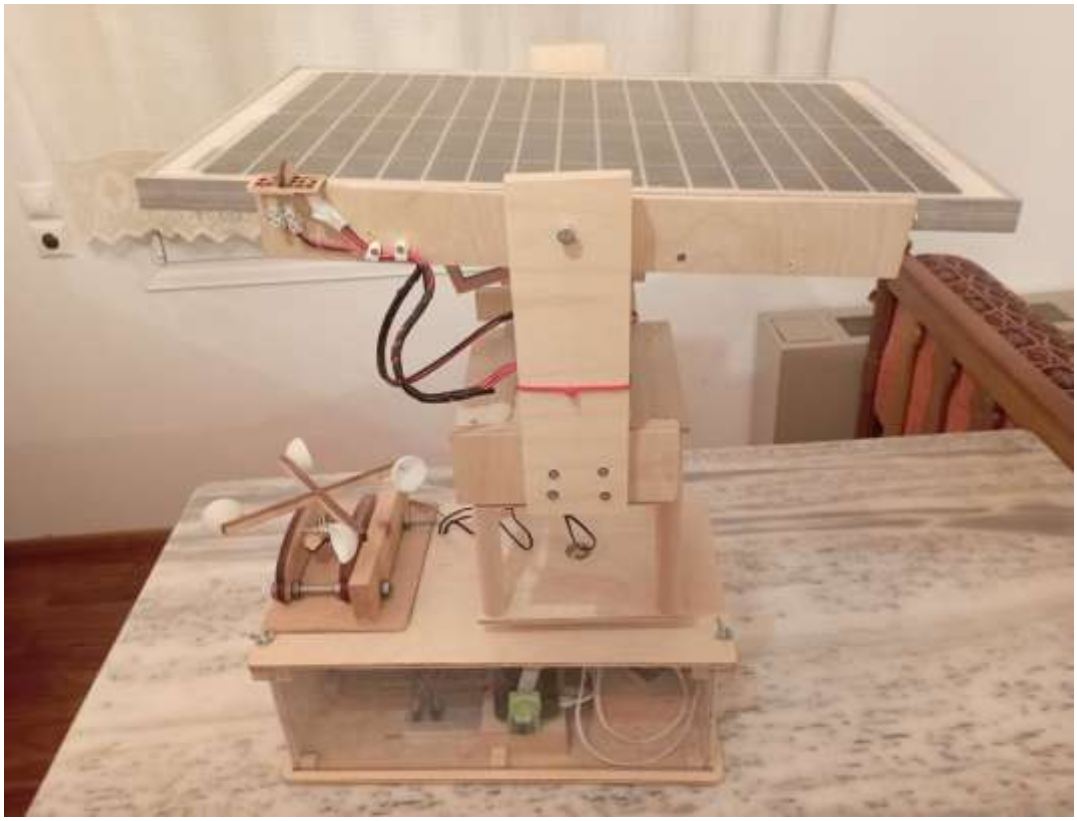
Εικόνα 7 βάση μετάδοσης κίνησης σερβοκινητήρα



Εικόνα 8βάση μετάδοσης κίνησης σερβοκινητήρα 2

Εικόνα 9 Περιστρεφόμενη βάση στήριξης ηλιακού πάνελ

Με την τοποθέτηση του πάνελ και την χρήση των διαδρομών για τα καλώδια που οδηγούν προς την βάση του κουτιού,χωρίς να εμποδίζει την ελεύθερη κίνηση των κινούμενων μερών του συστήματός μας, ολοκληρώθηκε η κατασκευή του σκελετού και των βάσεων στήριξης (εικόνα 8).



Εικόνα 10 Γενική εικόνα της κατασκευής

2.2. Σύστημα τροφοδοσίας

Το σύστημα τροφοδοσίας χωρίζεται σε 3 μέρη.

- Το ηλιακό πάνελ που έχει τα παρακάτω χαρακτηριστικά :
  - solar module type DG-M20W
  - Maximum Power(Pmax) 20W
  - Open-Circuit Current(Voc) 21.24V
  - Short- Circuit Current(Isc) 1.22A
  - Viltage at Pmax(Vmp) 18V
  - Current at Pmax(Imp) 1.12A
  - Power Tolerance +/- 3%
  - Maximum System Voltage 1000V

- Μια μπαταρία μολύβδου
  - 12V

- 12Ah
- 12packs
- 8cells

Στα οποία το ηλιακό πάνελ χρειάζεται 3.6 ώρες αιχμής του ηλίου για να φορτίσει πλήρως την μπαταρία μας, που αυτό μεταφράζεται για την περιοχή της Ανατολικής Μακεδονίας και Θράκης σύμφωνα με τα ιστορικά δεδομένα σε λιγότερο από μία ημέρα περίπου τους καλοκαιρινούς μήνες ενώ σε 2 ημέρες τους μήνες Μάρτιο, Απρίλιο, Μάιο, Σεπτέμβριο και Οκτώβριο με μέσο όρο έτους 4,28 ώρες αιχμής ανά ημέρα δηλαδή λιγότερο από 1ημέρα για να φτάσουμε σε πλήρη φόρτιση.

Ενώ η μπαταρία μας από πλήρη φόρτιση μπορεί να τροφοδοτήσει το σύστημά μας για 2 ημέρες. Πράγμα που σημαίνει ότι τον μήνα Δεκέμβριο που οι ώρες αιχμής του ηλίου φτάνουν τα 1,49 ανά ημέρα , που δηλαδή χρειάζονται περισσότερο από 2 ημέρες για πλήρη φόρτιση δεν είναι αρκετές για να τροφοδοτηθεί το σύστημά μας χωρίς πρόβλημα πράγμα που σημαίνει ότι αν θέλουμε ασφαλή όλο το έτος θα χρειαστούμε μια μεγαλύτερη μπαταρία.

Πιο αναλυτικά ο υπολογισμός πλήρους φόρτισης της μπαταρίας μας από το ηλιακό πάνελ έγινε με τον παρακάτω τρόπο:

Τάση ηλιακού πάνελ = 20W ÷ 12V = 1,667A

Χρόνος φόρτισης = 12Ah ÷ 1,667A = 7,2 ώρες

Λόγω του ότι χρησιμοποιούμε μπαταρία μολύβδου πολλαπλασιάζουμε τον χρόνο φόρτισης με το 50%

Χρόνος φόρτισης μπαταρίας μολύβδου = 7,2 x 50% = 3.6h[4]

Το αποτέλεσμα φυσικά ανάγεται σε ώρες αιχμής του ηλίου που σύμφωνα με το Εθνικό εργαστήριο ανανεώσιμων πηγών ενέργειας των Ηνωμένων πολιτειών της

Αμερικής [5] για την περιοχή μας (Ανατολική Μακεδονία και Θράκη) έχει τις εξής τιμές ανά μήνα σύμφωνα με τα ιστορικά στοιχεία :

Ετήσιο μέσο όρο:

4.28 ώρες αιχμής ηλίου ανά ημέρα

Μέσω όρο ανά μήνα:

Ιαν	Φεβ	Μαρ	Απρ	Μαϊ	Ιουν	Ιουλ	Αυγ	Σεπ	Οκτ	Νοε	Δεκ
1.75	2.92	3.97	4.9	6.02	7.1	7.07	6.24	4.77	3.18	1.96	1.49

Πίνακας 2Μέσος όρος ωρών αιχμής ηλίου ανά μήνα

Σε σχέση με την μπαταρία μας και την διάρκεια τροφοδοσίας του συστήματός μας θα πρέπει πρώτα να υπολογίσουμε την κατανάλωση των συσκευών μας.

Το arduino στο οποίο έχουμε συνδεδεμένο τον σερβοκινητήρας και τους ηλιακούς αισθητήρες βέβαια τις περισσότερες ώρες της ημέρας είναι σε αδράνεια οπότε υπολογίζεται ως μια μέση χρήσηπου σύμφωνα με την voltaic systems [6]

Όπως βλέπουμε στον παρακάτω πίνακαη μέγιστη κατανάλωση του Arduino είναι στα 25mAh

Arduino Runtime from V1S, V50 Batteries

#	Arduino Current Draw (mA)	Arduino Power Consumption (Watts)	Arduino Power Consumption per Day (Watt Hours)	Arduino + Battery Power Consumption Per Day (Watt Hours)	Days Runtime V1S	Days Runtime V50
Arduino - Sleep	5	.025	0.6	1.2	10.1	33.4
Arduino - Normal	25	.125	3	3.6	3.4	11.1
Arduino - High Power	300	1.5	36	36.6	0.3	1.1

Εικόνα 11Πίνακας κατανάλωσης ενέργειας Arduino

Στην συνέχεια στο Raspberry pi 4b η κατανάλωση ανέρχεται στα 540mAh σε μια χαμηλή χρήση όπως και στην δική μας περίπτωση που οι αισθητήρες χρησιμοποιούνται μερικές φορές την ημέρα [7], [8].

Ενώ το κινητό τηλέφωνο που χρησιμοποιήθηκε σε αντικατάσταση της GSM Shield υπολογίζεται ότι αφού χρειάζεται 5V / 2 A για την φόρτιση του δηλαδή 10 W και φορτίζει σε 2 ώρες , για την αποφόρτιση του χρειάζεται 3 ημέρες με την χρήση που κάνει το σύστημά μας αφού βρίσκεται σε μερική αδράνεια χρησιμοποιώντας μόνο την πρόσβαση στο διαδίκτυο. Πράγμα που σημαίνει ότι το ενεργειακό κόστος του είναι 10w / 48 h = 0.20W/h δηλαδή 40mAh Στο σύνολο δηλαδή 605 mAh.

Όλα τα παραπάνω σε σχέση με την μπαταρία μας που είναι στα 12 V 12Ah μετατρέποντας τη σε 5V μέσω του controller με απόδοση 90% έχουμε

12000mAh x (12V/5V) x 0.9 = 25920mAh στα 5V

Κάνοντας τους παρακάτω υπολογισμούς:

25920mAh / 605mAh = 42.84 h

Δηλαδή περίπου 2 ημέρες αφού τις βραδινές ώρες δεν χρησιμοποιείται ο σερβοκινητήρας και οι μετρήσεις μας συμβαίνουν περιοδικά μέσα στην ημέρα.

- Και ο Ελεγκτής του ηλιακού πάνελ με τα εξής χαρακτηριστικά
  - Rated Voltage 12v/24V
  - Rated Current 10A
  - PWM battery Charging

Ο συγκεκριμένος ελεγκτής πέρα από τις κλασικές εισόδους για το πάνελ και είσοδο για την μπαταρία έχει και δύο Usb-A εξόδους των 5v πράγμα πολύ πρακτικό για να συνδέσουμε το Raspberry Pi και το Arduino.

2.3. Σύστημα συλλογής δεδομένων

Τα δεδομένα που έχουμε επιλέξει να συλλέγουμε είναι τα κύρια που συμβάλλουν στην πρόκληση πυρκαγιάς είτε αυτόματης είτε λόγω εμπρησμού (βάσει στατιστικών δεδομένων και λογικής).

Όλοι μας οι αισθητήρες συνδέονται με το RPI που τροφοδοτείται από τον controller.

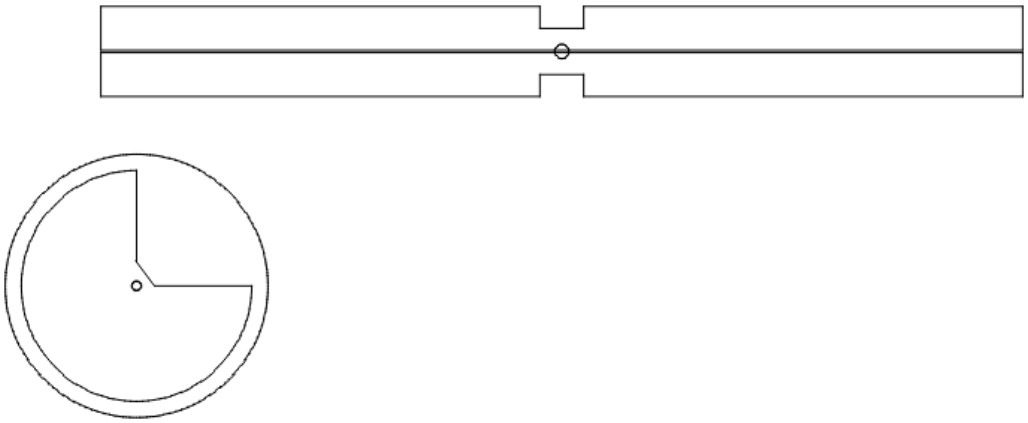
Η συλλογή δεδομένων γίνεται περιοδικά, ανά τριάντα λεπτά.

2.3.1. Ταχύτητα αέρα

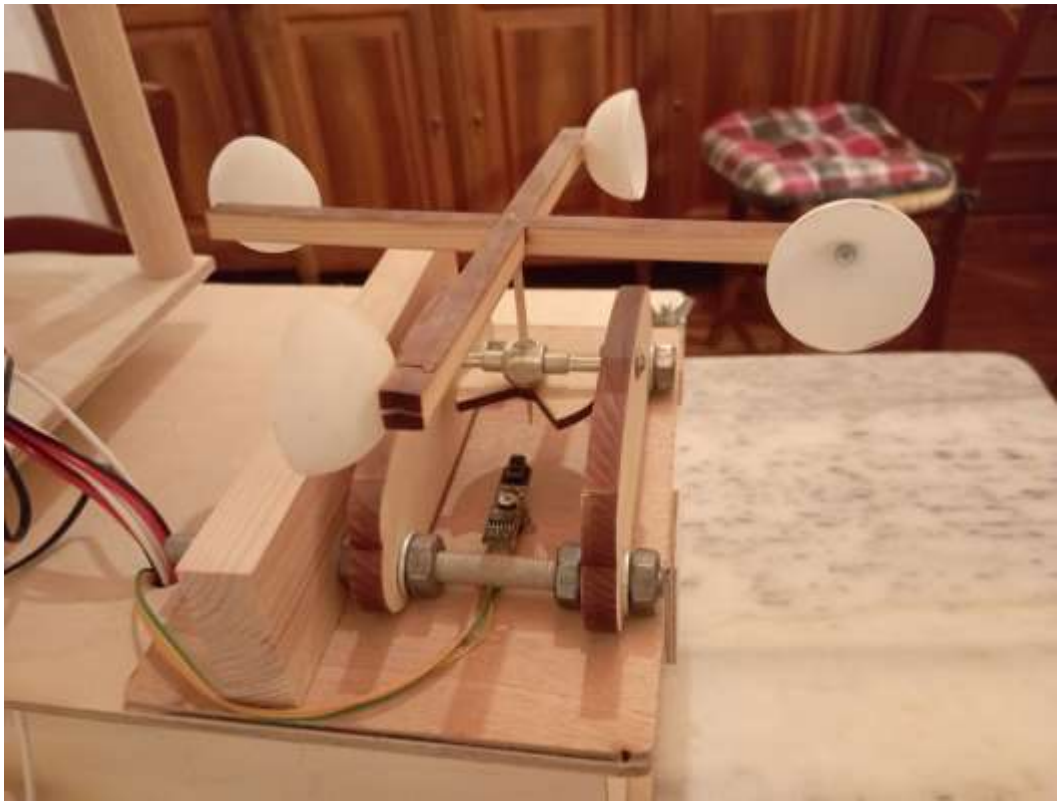
Το ανεμόμετρο που κατασκευάστηκε βασίζεται σε έναν αισθητήρα εμποδίων και έναν περιστρεφόμενο, κυκλικό δίσκο με κενό στο ¼ της επιφάνειάς του, τοποθετημένος σε έναν ξύλινο άξονα που στερεώνεται σε ένα ρουλεμάν και

καταλήγει σε έναν ξύλινο σταυρό με κενά ημισφαίρια στην εσωτερική μεριά των άκρων του με την ίδια φορά μεταξύ τους. Αυτό επιτρέπει να εγκλωβίζεται ο αέρας από την μία μεριά και να ‘γλιστρά’ από την άλλη δίνοντας έτσι κίνηση στον άξονα και κατ` επέκταση στον δίσκο, σχετική με την ταχύτητα του αέρα. Ο αισθητήρας εμποδίων μετράει σε ένα συγκεκριμένο χρονικό διάστημα πόσες αλλαγές στην κατάσταση εμποδίου / μη εμποδίου γίνονται και με αυτόν τον τρόπο υπολογίζεται η ταχύτητα του αέρα βάσει της απόστασης του άξονα από το κέντρο του ημισφαιρίδιου.

Παρακάτω μπορούμε να δούμε το βασικό σύστημα μετάδοσης κίνησης καθώς και το ανεμόμετρο στο σύνολό του.



Εικόνα 12 Βάση μετάδοσης κίνησης και δίσκος' εμποδίου'



Εικόνα 13 Ανεμόμετρο

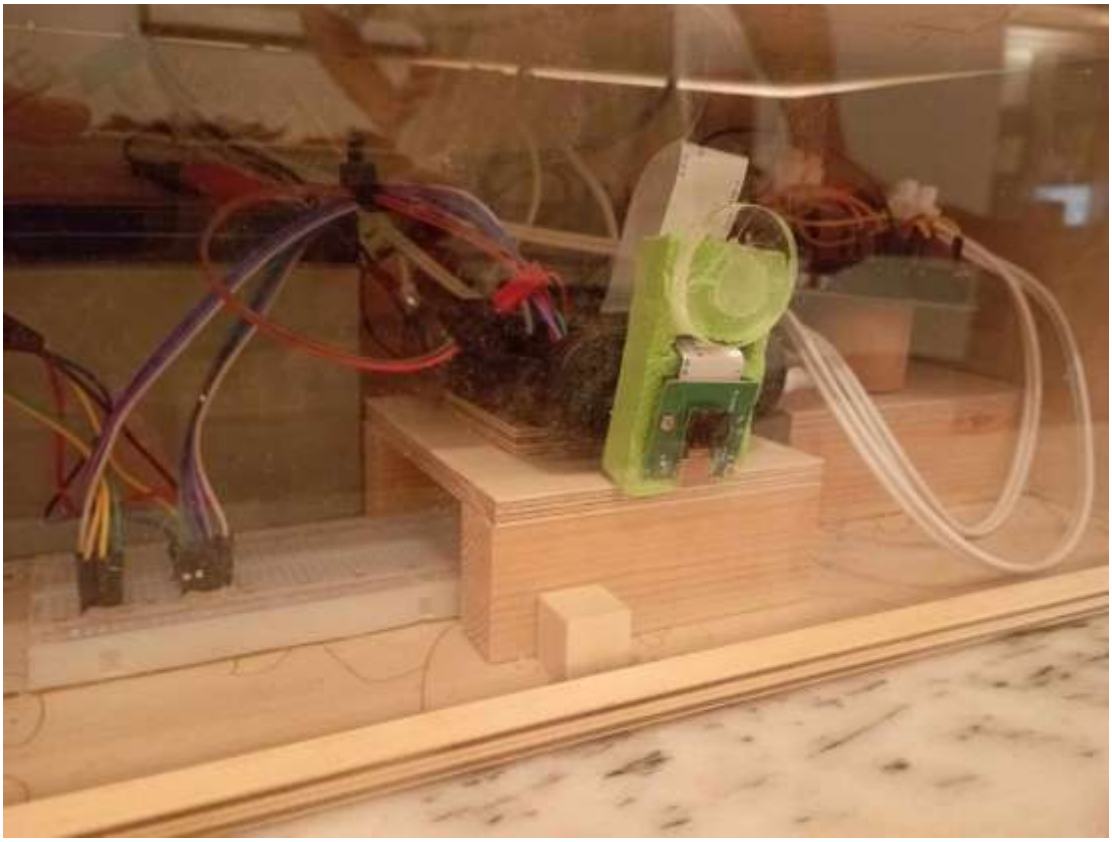
2.3.2. Θερμοκρασία – Υγρασία – Ατμοσφαιρική πίεση

Για την μέτρηση της Θερμοκρασίας,Υγρασίας και Ατμοσφαιρικής πίεσης χρησιμοποιήθηκε ο αισθητήρας Bme280 που έχει την δυνατότητα να συλλέγει και τις τρείς τιμές ταυτόχρονα με τον κώδικα που παρέχεται από τον κατασκευαστή. Η δυσκολία όμως για να μπορέσουμε να έχουμε αξιόπιστες μετρήσεις και να προφυλάσσεται ταυτόχρονα ο αισθητήρας, είναι στην τοποθέτησή του. Αυτό το πετύχαμε ανοίγοντας μια τρύπα στο πλαϊνό μέρος του κουτιού μεγέθους τριών καλωδίων που χρειάζεται ο αισθητήρας ώστε η θερμοκρασία από τα υπόλοιπα συστήματα να μην μεταδίδεται σε αυτόν, και αφού τοποθετήσαμε τον αισθητήρα τον καλύψαμε με ένα ξύλινο κουτί ώστε να προστατεύεται από τις καιρικές συνθήκες όπως η βροχή ή η έκθεση στον ήλιο (όπως έχουμε αναφέρει και προηγουμένως το ξύλο είναι κακός αγωγός της θερμότητας), ενώ στο κάτω μέρος του κουτιού ανοίχτηκε μια τρύπα ώστε να πετυχαίνονται σωστές μετρήσεις όμως να μην

σωρεύεται υγρασία ή / και νερό πράγμα που θα μπορούσε να συμβεί αν η πρόσβαση ήταν σε κάποιο άλλο σημείο του κουτιού.

2.3.3. Φωτογραφία

Για την λήψη της φωτογραφίας χρησιμοποιήθηκε μια pi-camera 1.3v τοποθετημένη στο κέντρο της πρόσοψης της κατασκευής. Η ακρυλική βάση με 90% φωτοδιαπερατότητα μας επιτρέπει να μην εμποδίζεται η λήψη φωτογραφίας. Η τοποθέτηση έγινε με CAκόλλα αφού προσδέθηκε η κάμερα σε μια μικρή επιφάνεια (εικόνα 11).



Εικόνα 14Pi Camera

3. ΣΥΝΔΕΣΜΟΛΟΓΊΑ ΚΑΙ ΔΥΝΑΤΌΤΗΤΕΣ

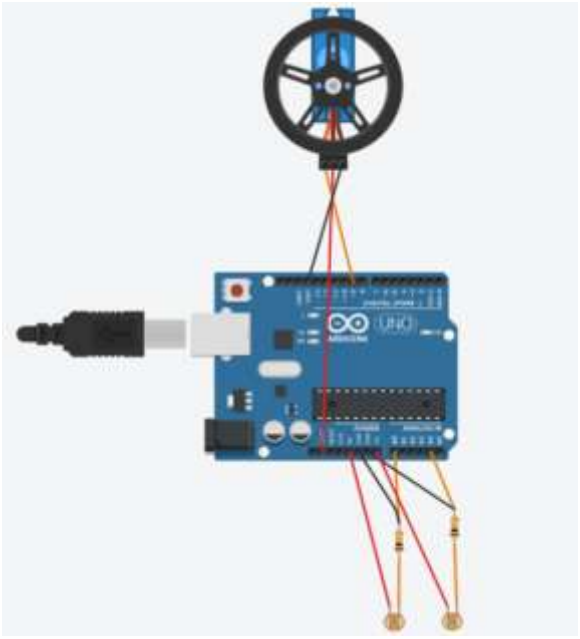
3.1. Τροφοδοσία – Μπαταρία – Controller – Ηλιακό Panel

Η βασική συνδεσμολογία ξεκινά από τον ελεγκτή ο οποίος διαχειρίζεται την αποθήκευση της παραγόμενης ενέργειας από το ηλιακό πάνελ στην μπαταρία και την τροφοδοσία του Arduino[9] και του Raspberry Pi[10]. Η συνδεσμολογία είναι τυποποιημένη, τα δύο καλώδια του πάνελ και της μπαταρίας έχουν συγκεκριμένη θέση ενώ έχει 3 εξόδους οι δύο από τις οποίες είναι Usb 5v/3A όπου και έχουμε συνδέσει το Arduino και το Raspberry Pi (εικόνα 12).

Εικόνα 15Τροφοδοσία – Μπαταρία – Controller – Ηλιακό Panel

3.2. Arduino – Σερβοκινητήρας - αισθητήρες LDR

Συνεχίζοντας, η συνδεσμολογία στο Arduino περιλαμβάνει ένα σερβοκινητήρα που εκτός της παροχής ενέργειας και της γείωσης του, συνδέεται στο pin 9. Και οι δύο LDR[11] αισθητήρες μας που ο ένας ακροδέκτης τους συνδέεται στη παροχή ενέργειας ενώ ο δεύτερος σε μια αντίσταση 10kohm και αυτή με την σειρά της διακλαδώνεται στην γείωση και στις αναλογικές θήρες A0 και A4 αντίστοιχα (εικόνα 13).



Εικόνα 16Arduino – Σερβοκινητήρας - αισθητήρες LDR

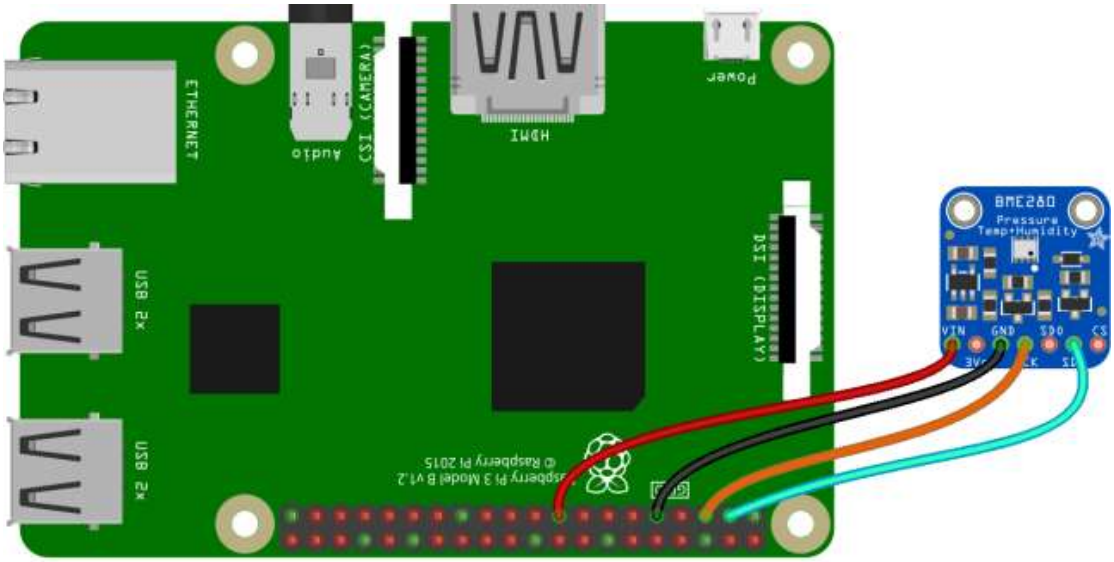
1.1. Raspberry Pi - Picamera - αισθητήρας Bme280 - Mobile-Ανεμόμετρο

Στο Raspberry Pi έχουμε συνδέσει τις 3 βασικές λειτουργίες του συστήματός μας

- Την Pi camera [12]που μας επιτρέπει να έχουμε οπτική επαφή με την περιοχή που θέλουμε να ελέγχουμε για πρόκληση πυρκαγιάς. Η συνδεσμολογία της είναι τυποποιημένη όπως φαίνεται παρακάτω (εικόνα 14).

Εικόνα 17 Συνδεσμολογία PiCamera

• Τον αισθητήρα Bme280 όπου μας παρέχει την θερμοκρασία , την υγρασία και την βαρομετρική πίεση. Η συνδεσμολογία του περιγράφεται στον παρακάτω πίνακα και οπτικοποιήται παρακάτω (εικόνα 15, Πίνακας 2)[12].



Εικόνα 18 Συνδεσμολογία αισθητήρα Bme280

Πίνακας 3Συνδεσμολογία αισθητήρα BME280

Pi GPIO	**BME280**
17 (3V3)	Vin
6 (Gnd)	Gnd
3 (SDA)	SDA (SDI)
5 (SCL)	SCL (SCK)

- Και τέλος το ανεμόμετρό μας που βασίζεται σε έναν αισθητήρα Infrared TCRT5000 συνδεόμενο στην GPIO 12 και στην παροχή ενέργειας και τη γείωση.

2. ΚΩΔΙΚΑΣ ΚΑΙ ΕΠΙΛΟΓΕΣ ΛΕΙΤΟΥΡΓΙΑΣ

Το τρίτο μέρος της εργασίας μας επικεντρώνεται στον κώδικα και τις επιλογές λειτουργίας που έχουμε κάνει ώστε το σύστημά μας να είναι πιο αποδοτικό, όμως και να δέχεται διορθώσεις γρήγορα και απομακρυσμένα.

Βλέποντάς το σαν σύνολο έχουμε να κάνουμε με δύο προγράμματα, το ένα για να ελέγξουμε τους αισθητήρες και τους σερβοκινητήρες (που δίνουν κίνηση και ευθυγραμμίζουν το ηλιακό πάνελ με τον ήλιο) του Arduino και το άλλο για να πάρουμε τις μετρήσεις από τους υπόλοιπους αισθητήρες και την φωτογραφία από την Picamera, που χρειαζόμαστε αλλά και να τα αποτυπώσουμε χρησιμοποιώντας το rpi και τις γλώσσες python 3.9 και ipython.

Επιπλέον επειδή ο έλεγχος του συστήματος θα γίνεται μέσω του jupyter notebook έχουμε προσθέσει κατά την εκκίνηση του rpi να ενεργοποιεί έναν jupyter notebook server με συγκεκριμένη διεύθυνση και θύρα, ούτως ώστε να μπορούμε να ελέγχουμε το notebook μέσω του τοπικού δικτύου μας. Για να μπορούμε να το ελέγχουμε και μέσω διαδικτύου έχουμε εγγράψει την συσκευή (rpi) στον λογαριασμό που δημιουργήσαμε στο remote.it όπου καθιστά διαθέσιμες τις επικοινωνίες από τη συσκευή αυτή αλλά και πιο συγκεκριμένα επιλέξαμε να μπορούμε να συνδεθούμε με την διεύθυνση και θύρα που επικοινωνεί το jupyter notebook με το πρωτόκολλο https.

2.1. Arduino

Ο κώδικας που ελέγχει την κίνηση του ηλιακού πάνελ μέσω των σερβοκινητήρων αλλά και οι μετρήσεις της έντασης φωταγώγησης των αισθητήρων LDR και ο υπολογισμός της διαφοράς τους γίνεται μέσω ενός Arduino UNO . Παρακάτω παραθέτεται ο κώδικας και η επεξήγησή του.

Αρχικά ορίζουμε τις θήρες σύνδεσης των LDR και αρχικοποιούμε τις μεταβλητές που καταγράφουν τις μετρήσεις τους.

```
intlsu = A0;
int lsuv = 0;
intlsd = A4;
intlsdv = 0;
```

Συνεχίζοντας, αρχικοποιούμε την μεταβλητή που θα αποθηκεύουμε την διαφορά της έντασης του φωτός και ορίζουμε την μέγιστη αποδεκτή διαφορά.

```
int d = 0;
int l = 90;
```

Ορίζουμε τον χρόνο εκτέλεσης κίνησης του Servo και αρχικοποιούμε την μεταβλητή μέτρησης κίνησης του πάνελ.(ορίζουμε επίσης την ένταση του φωτός όταν είναι σκοτάδι σε περίπτωση που θελήσουμε να τροποποιηθεί ο κώδικας περεταίρω για περισσότερη οικονομία ενέργειας θέτοντας το σύστημα σε αναμονή για την διάρκεια της νύχτας ξεκινώντας όταν τα επίπεδα φωτός φτάσουν το όριο που θέτουμε ως σκοτάδι)

```
intsp = 1000;
int counter = 10;
int dark = 850;
```

Περιλαμβάνουμε την βιβλιοθήκη Servo.

```
#include <Servo.h>
Servo servo;
```

Ορίζουμε την ταχύτητα μετάδωσης δεδομένων και τον ακροδέκτη πρόσδεσης του Servo.

```
void setup(){
Serial.begin(9600);
servo.attach(9, 600, 2300);
}
```

Ξεκινώντας τον βρόγχο επανάληψης συνδέουμε τα ευρήματα των αναλογικών θυρών που συνδέσαμε τους αισθητήρες LDR με τις μεταβλητές που τα αναθέσαμε, υπολογίζουμε την διαφορά τους και την μετατρέπουμε σε απόλυτη τιμή, καθώς επίσης ακινητοποιούμε το Servo.

```
void loop(){
lsuv = analogRead(lsu);
lsdv = analogRead(lsd);
d = lsuv - lsdv ;
d = abs(d);
servo.write(90);
delay(sp);
```

Ξεκινάμε ορίζοντας και θέτοντας λειτουργίες στα ενδεχόμενα τιμών και θέσης του πάνελ.

Στο πρώτο ενδεχόμενο που η απόλυτη διαφορά είναι μεγαλύτερη της αποδεκτής και ο lsuv είναι μεγαλύτερος του lsdv και ο μετρητής κλήσης είναι ίσος με την μέγιστη κλίση προς την κατεύθυνση που θέλουμε να κινήσουμε το πάνελ προκειμένου να ακολουθήσει την φορά του φωτός τότε θέτουμε το Servo σε ακινησία. Σε περίπτωση που ισχύουν όλα τα παραπάνω όμως ο μετρητής δεν είναι στην μέγιστη κλήση τότε θέτουμε σε λειτουργία το Servo προς την κατεύθυνση του φωτός για τον ορισμένο χρόνο κίνησης του και αυξάνουμε τον μετρητή κλίσης κατά μια μονάδα.

```
if (d > l && lsuv >lsdv)
{
if (counter == 15)
  {servo.write(90);
delay(sp);
  }
else
```

```
{
servo.write(0);
delay(sp);
counter = counter + 1;
}
}
```

Εδώ περιγράφεται το ακριβώς αντίστροφο ενδεχόμενο με διαφορετική μέγιστη κλίση, φορά του Servo , μείωση κατά μια μονάδα στον μετρητή και τέλος χρόνος εκτέλεσης της κίνησης του Servo όπως αυτή επηρεάζεται από τα στερεωμένα καλώδια λόγω του βάρους που προσθέτουν από αυτή τη μεριά.

```
else if (d > l && lsuv <lsdv)
{
if (counter == 3)
{
servo.write(90);
delay(sp);
}
else
{
servo.write(180);
delay(sp/1.8);
counter = counter - 1;
}
}
```

Τέλος σε όλα τα υπόλοιπα ενδεχόμενα θέτουμε το Servo σε ακινησία.

```
else
{
servo.write(90);
delay(sp);
}
Serial.println(d);
}
```

2.2. Raspberry - ανεμόμετρο - Bme280 – Picamera

Αρχικά εισαγάγουμε στο πρόγραμμα τις απαιτούμενες βιβλιοθήκες για να μπορέσουμε να θέσουμε σε λειτουργία τους αισθητήρες

- για τον bme280 την ομώνυμη βιβλιοθήκη και την smbus2
- για την PiCamera την picamera2, και το Preview ώστε να μπορούμε να κάνουμε προεπισκόπηση της φωτογραφίας, την libcamera και το Transform ώστε να μπορέσουμε να αλλάξουμε την φωτογραφία οπτική γωνία
- για να μπορέσουμε να αποθηκεύουμε τις φωτογραφίες με ονομασία την στιγμή που έγινε η λήψη αλλά και για τον υπολογισμό της επανάληψης της λήψης και γενικότερα όλου του προγράμματος εισαγάγουμε την time και την datetime.
- Για να μπορέσουμε να εκτελούμε το ίδιο πρόγραμμα και στο Jupyter notebook εισάγουμε το display και το clear_output από την IPython
- Για να μπορέσουμε να διαγράφουμε φωτογραφίες εισαγάγουμε την os

Τέλος ορίζουμε τον τρόπο απαρίθμησης των ακροδεκτών του Raspberry Pi και θέτουμε τον ακροδέκτη 12 ως εισροή δεδομένων.

```
import bme280
import smbus2
importRPi.GPIO as GPIO
from picamera2 import Picamera2, Preview
import time, libcamera
fromdatetime import datetime
fromIPython import display
fromlibcamera import Transform
fromIPython.display import clear_output
importos

# Set up GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12,GPIO.IN)
```

Ορίζουμε την θήρα και την διεύθυνση επικοινωνίας του αισθητήρα bme280.

```
port = 1
address = 0x77 # Adafruit BME280 address.
bus = smbus2.SMBus(port)
bme280.load_calibration_params(bus,address)
```

Ορίζουμε την διάμετρο από ημισφαιρίδιο σε ημισφαιρίδιο του έλικα και στην συνέχεια υπολογίζουμε την περίμετρο του κύκλου που σχηματίζουν τα ημισφαιρίδια σε μέτρα.

Δημιουργούμε μια μεταβλητή για να μπορούμε να προσαρμόζουμε το αποτέλεσμα σε περίπτωση αποκλίσεων που δεν μπορούμε να υπολογίσουμε.

Ορίζουμε την μεταβλητές ενεργοποίησης και περιστροφής.

Ορίζουμε τον χρόνο που θα γίνεται η μέτρηση των περιστροφών.

Τέλος ορίζουμε τον ακροδέκτη 12 ως αυτόν που θα ελέγχει την κίνηση του ανεμόμετρου.

```
vane_diameter = float(106)
vane_circ = float (vane_diameter/1000)*3.1415
afactor = float(5.5)
rotations = float(0)
trigger = 0
endtime = time.time() + 10
sensorstart = GPIO.input(12)
```

Ορίζουμε το μονοπάτι που θα διαγραφούμε τις χρονικά ξεπερασμένες φωτογραφίες.

Τον αριθμό των ημερών που όταν παρέλθει θεωρείται ξεπερασμένη.

Και ορίζουμε τον αριθμό των δευτερολέπτων που περιέχει μια ημέρα.

```
folder = "/home/katsanhs/Desktop/fire_detection_img"
N = 3
day = 86400
```

Ξεκινάμε έναν βρόγχο επανάληψης while Trueκαι συλλέγουμε τα δεδομένα από τον bme280 στις αντίστοιχες μεταβλητές.

```
while True:
    bme280_data = bme280.sample(bus,address)
humidity  = bme280_data.humidity
pressure  = bme280_data.pressure
ambient_temperature = bme280_data.temperature
```

Ξεκινάμε την μέτρηση της ταχύτητας του ανέμου για 10 δευτερόλεπτα. Σε αυτό το διάστημα όταν ο αισθητήρας μας ενεργοποιείται τότε προσθέτουμε στις περιστροφές μία μονάδα ενώ όταν είναι απενεργοποιημένος δεν κάνουμε τίποτα.

Δημιουργούμε μια καθυστέρηση στο πρόγραμμα ώστε να ομαλοποιηθεί η εκτέλεση του προγράμματος.

Σε περίπτωση άπνοιας και παραμονής του αισθητήρα σε κατάσταση ενεργοποίησης θέτουμε τις περιστροφές στο μηδέν.

Υπολογίζουμε τις περιστροφές ανά δευτερόλεπτο και την ταχύτητα του ανέμου πολλαπλασιάζοντας με την περίμετρο και τον συντελεστή απόκλισης.

```
whiletime.time() <endtime:
ifGPIO.input(12)==1 and trigger==0:
rotations = rotations + 1
trigger=1
ifGPIO.input(12)==0:
trigger = 0
time.sleep(0.001)
if rotations==1 and sensorstart==1:
rotations = 0
rots_per_second = float(rotations/10)
windspeed = float((rots_per_second)*vane_circ*afactor)
```

Ορίζουμε την κάμερα, την ανάλυση της και την μετατροπή για να είναι ορθή.

```
   picam2 = Picamera2()
camera_config = picam2.create_still_configuration(main={"size": (640, 480)}, //
//lores={"size": (640, 480)}, display="lores")
camera_config["transform"] = libcamera.Transform// //(hflip=1, vflip=1)
picam2.configure(camera_config)
```

Πριν ξεκινήσουμε την αποθήκευση των μετρήσεων από τον αισθητήρα bme280 και την λήψη από την PiCamera ορίζουμε ως timestamp την ώρα και την ημέρα.

Αποθηκεύουμε τις μετρήσεις από τον bme280.

```
timestamp = datetime.now().isoformat()
   l= ('temperature {:.1f}*C pressure {}hPa humidity {}%
windspeed {:.2f}m/s at {}'.format(ambient_temperature, pressure,
humidity, windspeed, timestamp))
```

Θέτουμε σε λειτουργία την κάμερα και εισάγουμε μια καθυστέρηση για την ομαλή λειτουργία της.

Πραγματοποιούμε την λήψη και αποθηκεύουμε στον αντίστοιχο φάκελο την φωτογραφία με την ονομασία του timestamp.

Σταματάμε την κάμερα και στην συνέχεια την απενεργοποιούμε (η απενεργοποίηση γίνεται για να μπορούμε οποιαδήποτε στιγμή θέλουμε να πραγματοποιήσουμε λήψη φωτογραφίας ή και προβολή βίντεο μέσω του Jupyter notebook).

```
picam2.start()
time.sleep(2)
picam2.capture_file('/home/katsanhs/Desktop/fire_detection_img/%s.jpg' % timestamp)
picam2.stop()
picam2.close()
```

Προβάλουμε τα αποτελέσματα στο Jupyter notebook

```
print(l)
   img2 = display.Image("/home/katsanhs/Desktop/fire_detection_img/%s.jpg" % timestamp)
display.display_jpeg(img2)
```

Ξεκινάμε την διαδικασία ελέγχου για φωτογραφίες οι οποίες έχει παρέλθει η χρησιμότητά τους ώστε να προχωρήσουμε στην διαγραφή τους, αλλάζοντας πρώτα την διεύθυνση εργασίας που βρισκόμαστε σε αυτή που αποθηκεύονται και στη συνέχεια φορτώνουμε την λίστα των παρόντων αρχείων, και της τρέχουσας ώρας.

Δημιουργούμε έναν βρόγχο που αντιστοιχίζει την τελευταία τροποποίηση του κάθε αρχείου.

Όταν η διαφορά ημερομηνίας ξεπερνά το όριο που έχουμε θέσει και το αρχείο είναι στην κωδικοποίηση που αποθηκεύουμε τις φωτογραφίες τότε τις διαγράφει και μας ενημερώνει για την διαγραφή τους εκτυπώνοντας την ενέργεια αυτή και το όνομα του αρχείου (η επιλογή κωδικοποίησης για την πραγματοποίηση της διαγραφής έγινε γιατί το Jupyter notebook αποθηκεύει προσωρινά αρχεία στον φάκελο κρίσιμα για την σωστή του λειτουργία)

```
os.chdir(os.path.join(os.getcwd(), folder))
list_of_files = os.listdir()
current_time = time.time()
for i in list_of_files:
file_location = os.path.join(os.getcwd(), i)
file_time = os.stat(file_location).st_mtime
if(file_time<current_time - day*N):
iffile_location.endswith('.jpg'):
print(f" Delete : {i}")
os.remove(file_location)
```

Θέτουμε σε αδράνεια το πρόγραμμα μέχρι την επόμενη προγραμματισμένη συλλογή δεδομένων.

```
time.sleep(1800)
```

Ένα κομμάτι του κώδικα που χρησημοποιήθηκε για το ανεμόμετρο βασίστηκε στον κώδικα του ChristopherBarnatt λέκτορα του NottinghamUniversityBusinessSchool[13].

2.3. Jupyter notebook

Η χρήση του Jupyter notebook μας δίνει την δυνατότητα να χειριζόμαστε εξολοκλήρου το Raspberry Pi είτε ξεκινώντας ένα τερματικό(terminal) είτε εκτελώντας προγράμματα.

Ο server που ξεκινάει αυτόματα με την εκκίνηση του συστήματος μας παραπέμπει στον home folder και από εκεί περιηγούμαστε στην επιφάνεια εργασίας που είναι αποθηκευμένο το notebook. Σε αυτό είναι αποθηκευμένο ολόκληρο το πρόγραμμα που εκτελείται κατά την εκκίνηση όμως εκτελώντας το ξανά από το notebook, παίρνει την θέση του πρώτου και για αυτό έχει προστεθεί μια γραμμή κώδικα που επανεκκινεί το Raspberry Pi.

Φυσικά οι δυνατότητες που δίνονται είναι πολλές και πάνω σε αυτό μπορεί να βασιστεί οποιοδήποτε νευρωνικό δίκτυο θελήσουμε να εκτελεί ανίχνευση φωτιάς ή/και καπνού.

2.4. Android

Η σύνδεση του Raspberry Pi με το διαδίκτυο γίνεται μέσω ενός κινητού Android, ωστόσο υπάρχουν πιο συνήθεις εναλλακτικές όπως ένα gsm hat.

Για να μπορέσουμε να χρησιμοποιήσουμε με αξιοπιστία ένα κινητό android θα πρέπει να το προσδέσουμε με μια θύρα Usb του raspberry Pi και να

αυτοματοποιήσουμε το Usb tethering, δηλαδή να μοιράζεται τη σύνδεση του διαδικτύου με την συνδεδεμένη συσκευή, αυτό επιτυγχάνεται ενεργοποιώντας το μενού προγραμματιστών και επιλέγοντας καθορισμένη λειτουργία κατά την σύνδεση Usb, στην προκειμένη περίπτωση να μοιράζεται την σύνδεση διαδικτύου.

Επιπλέον προγραμματίζουμε την αυτόματη ενεργοποίηση και απενεργοποίηση του κινητού μία φορά την ημέρα ώστε στην χειρότερη των περιπτώσεων το σύστημά μας να μείνει αποσυνδεδεμένο 24 ώρες σε περίπτωση έλλειψης ενέργειας η δυσλειτουργίας. Και φυσικά απενεργοποιούμε την εισαγωγή κωδικού πρόσβασης κατά την ενεργοποίηση του.

2.5. Remote.it (server)

Το Remote.it μας παρέχει την δυνατότητα να μοιραζόμαστε το τοπικό δίκτυο μας μέσω του server του. Δεδομένου ότι το jupyter notebook κατασκευάστηκε κυρίως για χρήση σε τοπικό δίκτυο, είναι μια αρκετά βολική επιλογή ώστε να μπορέσουμε να το αξιοποιήσουμε πλήρως και με ασφάλεια και σταθερότητα μέσω του διαδικτύου.

Δημιουργίήσαμε έναν λογαριασμό, όπου σε αυτόν συνδεόμαστε για την απομακρυσμένη σύνδεση, και εγγράψαμε το Raspberry Pi σε αυτό το λογαριασμό , αφού εγκαταστήσαμε το απαραίτητο λογισμικο. Τέλος συμπληρώσαμε τα στοιχεία που χρειάζονταν για να μπορούμε να συνδεόμαστε συγκεκριμένα στο Jupyter notebook (ip, port, https)[14].

3. ΣΥΜΠΕΡΑΣΜΑΤΑ-ΒΕΛΤΙΩΣΕΙΣ

Το βασικό συμπέρασμα, κρίνοντας σε σχέση με την καθημερινότητα που βιώνουμε μετά από τις φετινές πυρκαγιές είναι ότι υπάρχει ανάγκη για ένα τέτοιο σύστημα με εγκατεστημένο νευρωνικό δύκτιο, και το γεγονός ότι δεν έχει υιοθετηθεί σαν ένα στρατηγικό σχέδιο πρόληψης πυρκαγιών αποδεικνύει ότι υπάρχουν πολλά περιθώρια βελτίωσης ώστε να εμπιστευθούμε ένα τέτοιο σύστημα. Αυτό απορρέει από την δυσκολία στον εντοπισμό, την αναγώριση και την αντιμετώπιση ‘θορύβου’ από το περιβάλλον. Παρόλα αυτά θα μπορούσε να εγκατασταθεί σαν παρατηρητήριο έτσι ώστε να αξιοποιηθεί πιο σωστά το πυροσβεστικό σώμα.

Το κομμάτι των βελτιώσεων της εργασίας θα μπορούσε να είναι το πιο μακροσκελές αφου οι επιλογές και οι τρόποι κατασκευής ποικίλουν . Ταυτόχρονα όταν ολοκληρώνεται μία κατασκευή μπορούμε να δούμε λάθη και αστοχίες που προηγουμένως δεν μπορούσαμε να υπολογίσουμε ή δεν κατείχαμε την γνώση για να το κάνουμε.

- Αρχικά όσων αφορά το κατασκευαστικό κομμάτι της βάσης θα μπορούσε να γίνει πιο μικρό καταργώντας το πάνω κινούμενο μέρος και αξιοποιώντας αποκλειστικά τις ώρες αιχμής του ηλίου χωρίς αυτό να κινείται. Αυτό θα μας έδινε την δυνατότητα να καταργήσουμε το Arduinoεξ`ολοκλήρου και μαζί με αυτό τον υψηλής κατανάλωσης σε ενέργεια σερβοκινητήρα.
    - Μια ακόμη εκδοχή θα μπορούσε να είναι η χρήση σερβοκινητήρων 180 μοιρών για να έχουμε μεγαλύτερη ακρίβεια και επανάληψη στην κίνηση του πάνελ.
    - Μια πολύ χρήσιμη αναβάθμιση θα ήταν να προσθέταμε έναν αισθητήρα άνθρακα που θα έκανε το σύστημά μας πιο αποδοτικό στον εντοπισμό πυρκαγιάς σε μια ακτίνα γύρω του αλλά και σε σημεία που δεν θα είχαμε εικόνα.
    - Όπως είπαμε και προηγουμένως μια GSMShield για να αντικαταστούσαμε το κινητό τηλέφωνο.

- Θα μπορούσε να χρησιμοποιηθεί μια μεγαλύτερης χωρητικότητας μπαταρία ώστε να μην κινούμαστε στα όρια κατά τους χειμερινούς μήνες.

Φυσικά η λίστα δεν έχει τέλος και με το πέρασμα του χρόνου θα αυξάνεται ακόμη περισσότερο, όμως οι παραπάνω βελτιώσεις θα μπορούσαν να γίνουν χωρίς ιδιαίτερο κοπο και σε μερικές περιπτώσεις ενδεχομένως να έκαναν το σύστημάμας πιο αποδοτικό.

Φυσικά δεν θα μπορούσαμε να παραλήψουμε τον τελικό σκοπό που θα ολοκλήρωνε το σύστημα μας, που αν και αναφερθήκαμε σε αυτό θεωρουμε ότι υπάρχει ανάγκη αποσαφήνισης για τον τρόπο που θα μπορούσαμε να προσθέσουμε ένα νευρωνικό δίκτυο που θα μπορεί να αξιοποεί όλα τα χαρακτηριστικά του. Αρχικά αν κατηγοριοποιήσουμε τα νευρωνικά δίκτυα θα μπορούσαμε να τα χωρίσουμε σε δύο κατηγορίες αυτά που ανιχνεύουν φωτιά και αυτά που ανιχνεύουν και καπνό και φωτιά. Λόγω του ότι ο προορισμός του συστήματός μας αφορά δασικές εκτάσεις όπου η κάυσιμη ύλη είναι με υψηλή υγρασία θα ήταν πιο σωστό να επιλέξουμε κάποιο νευρωνικό δίκτυο με ανίχνευση φωτιάς αλλά και καπνού. Στην συνέχεια θα πρέπει να αναφέρουμε ένα από τα χαρακτηριστικά των νευρωνικών δικτύων (αυτό του ορισμολυ της ευαισθησίας) που θα χρησιμοποιήσουμε ως μέσο για την αξιοποίηση των μετρήσεών μας και την ερμηνεία τους από οποιοδήποτε νευρωνικό δίκτυο.

Ξεκινώντας λοιπόν θα πρέπει να ορίσουμε επίπεδα συνεγερμού ανάλογα των τιμών των αισθητήρων μας όπως περιγράφονται στον παρακάτω πίνακα, φυσικά οι τιμές είναι ενδεικτικές και επηρεάζονται από νευρωνικό σε νευρωνικό δίκτυο

Επίπεδο	Ευαισθησία	Ανίχνευση Καπνού	Θερμοκρασία	Υγρασία	Ταχύτητα ανέμου
1	95-	Ναι	35<	0-10	5<
2	80-95	Ναι	35<	0-10	0-5
3	70-80	Ναι	30-34	10-20	0<
4	65-70	Ναι	20-30	20-50	0<
5	55-65	Ναι	10-20	50-70	0<

6	50-65	Ναι	5-10	70-80	0<
7	50-65	Όχι	5>	80-100	0<

Πίνακας 4 Τρόπος εισαγωγής μετρήσεων σε νευρωνικό δίκτυο

Στον παραπάνω πίνακα βλέπουμε το πώς επηρεάζεται το νευρωνικό δίκτυο ανάλογα των μετρήσεων των αισθητήρων μας , έτσι όταν έχουμε πολύ υψηλή θερμοκρασία , σχεδόν μηδενική υγρασία και υψηλό άνεμο ορίζουμε την ευαισθησια στο υψηλότερο επίπεδο , ενώ όταν έχουμε θερμοκρασία κοντά στο 0 και υγρασία περισσότερο από 80% ορίζουμε το χαμηλότερο επίπεδο ενώ απενεργοποιούμε τον εντοπισμό καπνού αφού είναι πολύ πιθανό ναι σχηματιστεί ομίχλη και θα μπορούσαμε να οδηγηθούμε σε λάθος συνεγερμό. Για να ολοκληρωθεί η σύνδεση χρειάζεται ένα επιπλέον πρόγραμμα που θα έχει ως είσοδο τις τιμές των αισθητήρων και κάνοντας τις σχετικές συγκρίσεις- υπολογισμούς θα ορίζει στο νευρωνικό δίκτυο την ευαισθησία του.

Βιβλιογραφία

[1] BME280 data sheet 2012-11-06
https://nettigo.eu/attachments/442

[2] Infared Sensor TCRT5000
https://www.haoyuelectronics.com/Attachment/TCRT5000/tcrt5000.pdf

[3] Jupyter notebook
https://jupyter.org/

[4] Solar Panel Charge Time Calculator, Alex Beale16-08-2022
https://footprinthero.com/solar-panel-charge-time-calculator

[5] Εθνικό εργαστήριο ανανεώσιμων πηγών ενέργειας των Ηνωμένων πολιτειών της Αμερικής
https://www.nrel.gov/index.html

[6] Voltaic systems
https://voltaicsystems.com/solar-arduino-guide/#faq_1

[7] Power Consumption Benchmarks
https://www.pidramble.com/wiki/benchmarks/power-consumption)

[8] Raspberry Pi Power Consumption Guide August 30, 2022
https://www.ecoenergygeek.com/raspberry-pi-power-consumption/

[9] Arduino Uno
https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[10] Rpi model 4b
https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[11] LDR sensor(Light dependent resistors)
https://components101.com/sites/default/files/component_datasheet/LDR%20Datasheet.pdf

[12] Weather station, Raspberry Pi Fountation
https://projects.raspberrypi.org/en/projects/build-your-own-weather-station/2

[13] Christopher Barnatthttps://www.explainingcomputers.com/pi_weather_video.html

[14] Remote.it https://www.remote.it/

[15] 'Modular Weather and Environment Monitoring Systems using Raspberry Pi' International Journal of Engineering Research & Technology Kuruvadi Praveen, AnkhitBalaVenkata,Rithesh M Nanda,Harshini H Kadakol, Department of Telecommunications BMS College of Engineering Bangalore, India https://www.ijert.org/research/modular-weather-and-environment-monitoring-systems-using-raspberry-pi-IJERTV3IS090619.pdf

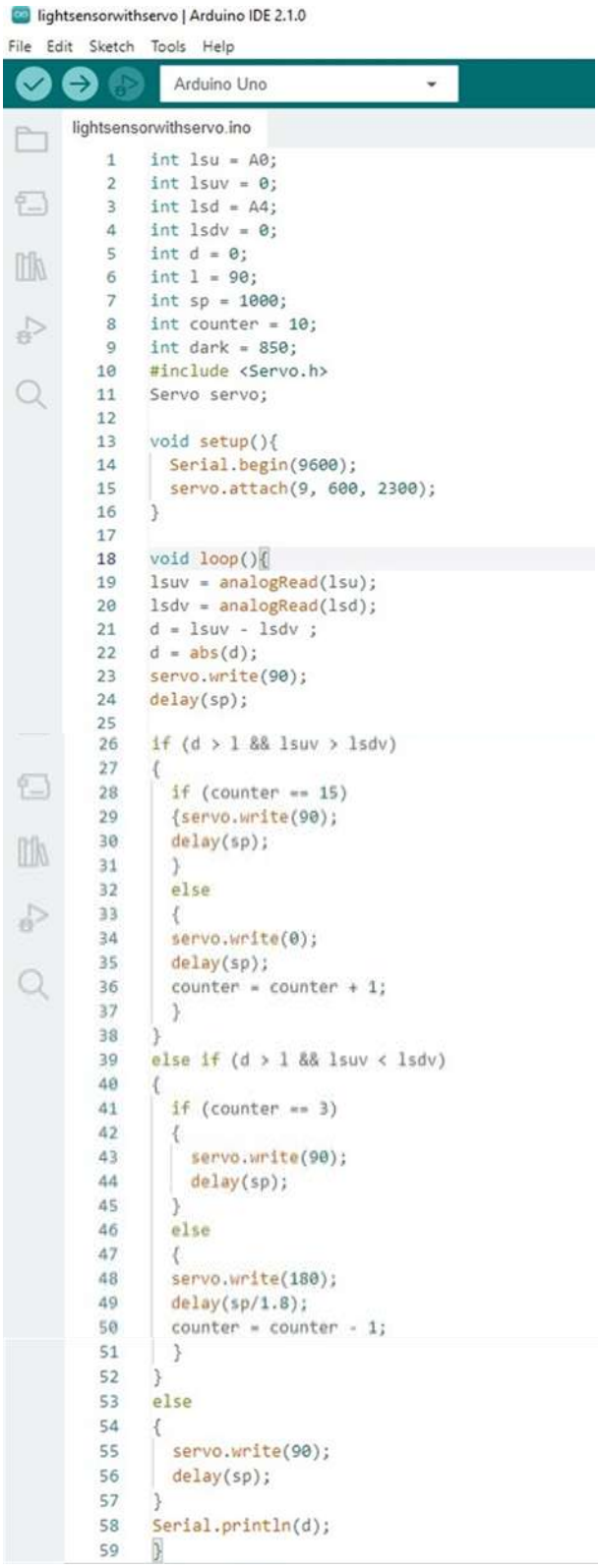
[16] Iot Based Weather Station Using Raspberry Pi 3 P Y Muck1 and M J Homam2 International Journal of Engineering & Technology https://www.researchgate.net/publication/332676356_Iot_Based_Weather_Station_Using_Raspberry_Pi_3

[17] Πυροσβεστικό σώμα Ελλάδας https://www.fireservice.gr

[18] Ευρωπαϊκό Σύστημα Πληροφόρησης για Δασικές Πυρκαγιέςhttps://effis.jrc.ec.europa.eu

Παράρτημακώδικα

Arduino



Raspberry Pi

```
#!/usr/bin/env python

import bme280
import smbus2
importRPi.GPIO as GPIO
from picamera2 import Picamera2, Preview
import time, libcamera
fromdatetime import datetime
fromIPython import display
fromlibcamera import Transform
fromIPython.display import clear_output
importos

# Set up GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12,GPIO.IN)

port = 1
address = 0x77 # Adafruit BME280 address.
bus = smbus2.SMBus(port)

bme280.load_calibration_params(bus,address)

# Anamometer vane diameter (set to the value for your cup-to-cup in mm)
vane_diameter = float(106)

# Calculate vane circumference in metres
vane_circ = float (vane_diameter/1000)*3.1415

# Set an anamometer factor to account for inefficiency (value is a guess)
afactor = float(5.5)
```

```
# Define variables rotations and trigger (trigger = 1 if sensor triggered)
rotations = float(0)
trigger = 0

# Define variable endtime to be current time in seconds plus 10 seconds
endtime = time.time() + 10

# Get initial state of sensor
sensorstart = GPIO.input(12)



##########     auto delete jpgs start    #########

# folder is the name of the folder in which we
# have to perform the delete operation
folder = "/home/katsanhs/Desktop/fire_detection_img"

# N is the number of days for which
# we have to check whether the file
# is older than the specified days or not
N = 3
# "day" is the number of seconds in a day
day = 86400

##########     auto delete jpgs end     #########

while True:



   bme280_data = bme280.sample(bus,address)
humidity  = bme280_data.humidity
pressure  = bme280_data.pressure
ambient_temperature = bme280_data.temperature
```

```
        # Measurement loop to run for 10 seconds
whiletime.time() <endtime:
ifGPIO.input(12)==1 and trigger==0:
rotations = rotations + 1
trigger=1
ifGPIO.input(12)==0:
trigger = 0
      # We seem to need to a little delay to make things work reliably...
time.sleep(0.001)

   # Loop has now finished. But if sensor triggered at start and did not move,
   # rotations value will be 1, which is probably wrong, so . . .
if rotations==1 and sensorstart==1:
rotations = 0

   # Calculate
rots_per_second = float(rotations/10)
windspeed = float((rots_per_second)*vane_circ*afactor)

   ##clear_output(wait=True)
   picam2 = Picamera2()
camera_config = picam2.create_still_configuration(main={"size": (640, 480)},
lores={"size": (640, 480)}, display="lores")
camera_config["transform"] = libcamera.Transform(hflip=1, vflip=1)
picam2.configure(camera_config)

timestamp = datetime.now().isoformat()
   l= ('temperature {:.1f}*C pressure {}hPa humidity {}% windspeed {:.2f}m/s at
{}'.format(ambient_temperature, pressure, humidity, windspeed, timestamp))

picam2.start()
time.sleep(2)
   picam2.capture_file('/home/katsanhs/Desktop/fire_detection_img/%s.jpg'          %
timestamp)
picam2.stop()
picam2.close()
```

```
print(l)
   img2 = display.Image("/home/katsanhs/Desktop/fire_detection_img/%s.jpg" % timestamp)
display.display_jpeg(img2)


   ##########  auto delete jpgs start  #########
   # changing the current working directory
   # to the folder specified
os.chdir(os.path.join(os.getcwd(), folder))
   # get a list of files present in the given folder
list_of_files = os.listdir()
   # get the current time
current_time = time.time()
   # loop over all the files
for i in list_of_files:
      # get the location of the file
file_location = os.path.join(os.getcwd(), i)
      # file_time is the time when the file is modified
file_time = os.stat(file_location).st_mtime

      # if a file is modified before N days then delete it
if(file_time<current_time - day*N):
iffile_location.endswith('.jpg'):
print(f" Delete : {i}")
os.remove(file_location)

   ##########  auto delete jpgs  end  #########

time.sleep(1800)
```