Ατομική Διπλωματική Εργασία

ΑΝΙΧΝΕΥΣΗ ΤΗΣ ΠΙΘΑΝΟΤΗΤΑΣ ΜΟΛΥΝΣΗΣ ΑΠΟ ΚΟΡΟΝΟΙΟ ΣΕ ΕΝΑ ΔΩΜΑΤΙΟ, ΜΕ ΤΗΝ ΧΡΗΣΗ RASPBERRY PI

Χριστίνα Χατζηζωρζή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Δεκέμβριος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΙΧΝΕΥΣΗ ΤΗΣ ΠΙΘΑΝΟΤΗΤΑΣ ΜΟΛΥΝΣΗΣ ΑΠΟ ΚΟΡΟΝΟΙΟ ΣΕ ΕΝΑ ΔΩΜΑΤΙΟ, ΜΕ ΤΗΝ ΧΡΗΣΗ RASPBERRY PI

Χριστίνα Χατζηζωρζή

Επιβλέπων Καθηγητής

Μάριος Δικαιάκος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Δεκέμβριος 2021

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Δρ. Μάριο Δικαιάκο για την καθοδήγηση του, τις συμβουλές και την άψογη συνεργασία που είχαμε καθόλη τη διάρκεια του χρόνου.

Θα ήθελα να ευχαριστήσω όλους του καθηγητές μου στο Πανεπιστήμιο Κύπρου που μου πρόσφεραν τις γνώσεις και τις δεξιότητες που απέκτησα σε αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την έμπρακτη υποστήριξη που μου πρόσφερε στα φοιτητικά μου χρόνια, καθώς και του φίλους μου για την υπομονή και την συμπαράσταση τους.

Περίληψη

Ένα από τα σημαντικότερα προβλήματα που προβληματίζουν την κοινωνία ανά το παγκόσμιο είναι η ταχεία μετάδοση του Κορωνιού. Ο ιός και η ασθένεια πρωτοεμφανίστηκαν στα τέλη του 2019, στην πόλη Γουχάν της Κίνας, όπου και έγιναν γνωστοί στο Παγκόσμιο Οργανισμό Υγείας στις 31 Δεκεμβρίου 2019.

Με αφορμή το πιο πάνω, σκοπός της εργασίας είναι η δημιουργία μιας συσκευής, η οποία με την χρήση αισθητήρων και άλλων εξαρτημάτων, να παίρνει τιμές από το περιβάλλον στο οποίο βρίσκεται και να τις περνά από ένα αλγόριθμο, παρουσιάζοντας κατά πόσο στην αίθουσα που βρίσκεται η συσκευή μπορεί να μεταδοθεί ο ιός ή όχι.

Η συσκευή είναι ένα raspberry pi, και οι αισθητήρες από τους οποίους παίρνουμε τις μετρήσεις είναι οι DHT22, PIR, CCS811, KY037. Άλλα εξαρτήματα είναι ένα ψηφιακό μικρόφωνο και το Bluetooth detection το οποίο βρίσκεται ήδη πάνω στο raspberry pi.

Αρχικά έχει αναπτυχθεί ένα πρόγραμμα συλλογής δεδομένων και στην συνέχεια, μόλις τα δεδομένα παρθούν, τότε περνούν από δύο αλγόριθμους, τον αλγόριθμο ανάλυσης δεδομένων και τον αλγόριθμο ανάλυσης ρίσκου.

Στην συνέχεια αναπτύχθηκε ένα πρόγραμμα API για την παρουσίαση των δεδομένων, με βάση τις παραμέτρους που δίνει ο χρήστης.

Τέλος παρουσιάζονται τα αποτελέσματα που έχουμε πάρει από τους παραπάνω αλγόριθμους και τα συμπεράσματα που προκύπτουν, ανάλογα με τις παραμέτρους που έχουμε πάρει, για το πότε ένα δωμάτιο μολύνεται πιο γρήγορα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Το κεφάλαιο αυτό περιέχει τα εισαγωγικά θέματα της διπλωματικής εργασίας, οι στόχοι που τέθηκαν για την εργασία και μια σύντομη περιγραφή της δομής αυτής της αναφοράς και το περιεχόμενων των κεφαλαίων της.

1.1 Υποκίνηση της Εργασίας

Ένα από τα σημαντικότερα κοινωνικά προβλήματα που απασχολεί την ανθρωπότητα τα τελευταία 2 χρόνια είναι η ραγδαία μετάδοση της οξείας αναπνευστικής νόσου 2019-nCoV, γνωστής ως COVID-19.

Είναι μια μολυσματική ασθένεια, που προκαλείται από τον κορονοϊό SARS-CoV-2. Ο ιός και η ασθένεια εντοπίστηκαν για πρώτη φορά στην πόλη Γουχάν της Κίνας στα τέλη του 2019 και έγιναν γνωστοί στον Παγκόσμιο Οργανισμό Υγείας στις 31 Δεκεμβρίου του 2019. Από τότε έχει διασπαρεί σε όλο τον πλανήτη και έχει εξελιχθεί σε πανδημία.

Υπάρχουν πολλά συμπτώματα που εμφανίζονται με την έξαρση του ιού. Τα πιο συχνά από αυτά είναι ο πυρετός, ο ξηρός βήχας και σωματική εξάντληση. Λιγότερα συχνά, είναι η απώλεια γεύσης ή μυρωδιάς, η ρινική συμφόρηση, η επιπεφυκίτιδα, ο πονόλαιμος, ο πονοκέφαλος, ο πόνος στους μύες ή στις αρθρώσεις, τα δερματικά εξανθήματα, η ναυτία ή ο εμετός, η διάρροια, τα ρίγη και η ζάλη. [12]

Ο τρόπος διάδοσης μεταξύ των ανθρώπων, είναι κυρίως όταν ένα μολυσμένο άτομο βρίσκεται σε στενή επαφή με ένα άλλο. Ο ιός μπορεί να εξαπλωθεί από τη μύτη ή το στόμα ενός μολυσμένου ατόμου με υγρά σωματίδια τα οποία ονομάζονται «αναπνευστικά σταγονίδια» τα μεγαλύτερα, ενώ τα μικρότερα, «αερολύματα». Η μετάδοση, από τα μολυσμένα άτομα, συμβαίνει όταν τα άτομα βήχουν, φτερνίζονται, μιλούν, τραγουδούν ή αναπνέουν έντονα. Επίσης μπορεί να μεταδοθεί και μέσω της αφής από αντικείμενα ή επιφάνειες. Οι άνθρωποι μπορούν να προσβληθούν από τον ιό, όταν αυτός εισέλθει στο στόμα, τη μύτη ή τα μάτια τους, που είναι πιο πιθανό να συμβεί όταν οι άνθρωποι βρίσκονται σε άμεση ή στενή επαφή, δηλαδή σε απόσταση μικρότερη από 2 μέτρα[12], αυτό το βλέπουμε και με βάση τον όγκο του δωματίου, ότι ο αριθμός των ατόμων είναι συγκεκριμένος, με βάση τον όγκο.

Το ποσοστό μετάδοσης κυμαίνεται με βάση διαφορετικές παραμέτρους. Αν, για παράδειγμα, το μολυσμένο άτομο δεν φορά μάσκα και συναναστραφεί με ένα άλλο υγιές άτομο που ούτε αυτό φορά μάσκα, τότε το ποσοστό μετάδοσης εκτιμάται 90%. Όμως, αν το υγιές άτομο φορά μάσκα τότε, το ποσοστό μειώνεται στο 70%. Επίσης, αν το μολυσμένο άτομο φορά μάσκα και το υγιές δεν φορά, τότε το ποσοστό μετάδοσης πέφτει στο 5% και τέλος αν και το μολυσμένο και το υγιές φορούν μάσκα αντίστοιχα, τότε το ποσοστό μειώνεται στο 1.5%. [14]

Για αυτούς τους λόγους κρίθηκε σημαντικό ο σχεδιασμός και η υλοποίηση ενός κυβερνοφυσικού συστήματος για την εκτίμηση του ρίσκου μετάδοσης κορονοϊού σε αίθουσες.

1.2 Σκοπός της εργασίας

Στόχοι της της διπλωματικής εργασίας είναι η υλοποίηση αλγορίθμου εκτίμησης ρίσκου, βασιζόμενη σε ροή δεδομένων που εξάγονται από αισθητήρες εγκαταστημένους σε ενσωματωμένο ηλεκτρονικό υπολογιστή χαμηλού κόστους, τοποθετημένος σε μια αίθουσα.
Πιο συγκεκριμένα, κάνουμε χρήση ορισμένων αισθητήρων, όπως τον DHT22[15], οποίος παίρνει την θερμοκρασία και την υγρασία του δωματίου. Επίσης του αισθητήρα PIR[16], ο οποίος ανιχνεύει την κίνηση και μετρά τα άτομα που βρίσκονται μέσα σε ένα δωμάτιο, τον αισθητήρα CCS811[17], ο οποίος ανιχνεύει το διοξείδιο του άνθρακα. Ο αισθητήρας KY037[18] και το μικρόφωνο ESPERANZA EH179 MICROPHONE USB SCREAM[20], τα οποία παίρνουν τον ήχο. Επιπρόσθετα με την χρήση της βιβλιοθήκης PyBluez[23], μετρά πόσα άτομα βρίσκονται μέσα σε ένα δωμάτιο, ανάλογα με το αν έχουν ανοικτό το Bluetooth του κινητού τους, όπου η μέθοδος αυτή είναι συμπληρωματική με την μέθοδο του αισθητήρα PIR.
Μέσω των μετρήσεων που παίρνουμε, τα δεδομένα περνούν από ένα αλγόριθμο και με βάση τις μετρήσεις, ο αλγόριθμος εκτιμά κατά πόσο είναι ένας χώρος σε υψηλό ρίσκο μετάδοσης.
Αυτά πραγματοποιούνται σε πραγματικό χρόνο, συνεπώς και ο αλγόριθμος ανάλυσης δεδομένων θα πρέπει να είναι γρήγορος, έτσι ώστε να υπάρχει άμεσο αποτέλεσμα, με σκοπό να γνωρίζουμε μετά από ποιο χρονικό διάστημα ένα δωμάτιο γίνεται τοξικό (μεταδοτικό) για τα άτομα που βρίσκονται μέσα σε αυτό.
Το πρόγραμμα θα μπορέσει να πραγματοποιήσει όλα τα πιο πάνω, χωρίς να υπάρχει επίβλεψη από κάποιο χρήστη. Με άλλα λόγια όλα θα πρέπει να λειτουργούν αυτόματα.

2.1 Περίγραμμα της εργασίας

Η ατομική διπλωματική εργασία αποτελείται από 6 κεφάλαια:

- **Κεφάλαιο 1:** Γίνεται μια εισαγωγή για τον Κορονοϊό, της τρόπους μετάδοσης του, καθώς και τα ποσοστά μεταδοτικότητας του.

- **Κεφάλαιο 2:** Περιγραφή του συστήματος και των εξαρτημάτων που το απαρτίζουν, καθώς και οι ολικές συνδέσεις που έχουν γίνει.

- **Κεφάλαιο 3:** Γίνεται λεπτομερής περιγραφή της μεθοδολογίας που έχει ακολουθηθεί, καθώς και το πειραματικό περιβάλλον που έχει χρησιμοποιηθεί.

- **Κεφαλαίο 4:** Παρουσίαση πειραματικών αποτελεσμάτων.

- **Κεφάλαιο 5:** Παρουσίαση των τελικών συμπερασμάτων και της μελλοντικής δουλειάς που μπορεί να γίνει.

Κεφάλαιο 2

Περιγραφή – Ανάλυση Υλικού Συστήματος

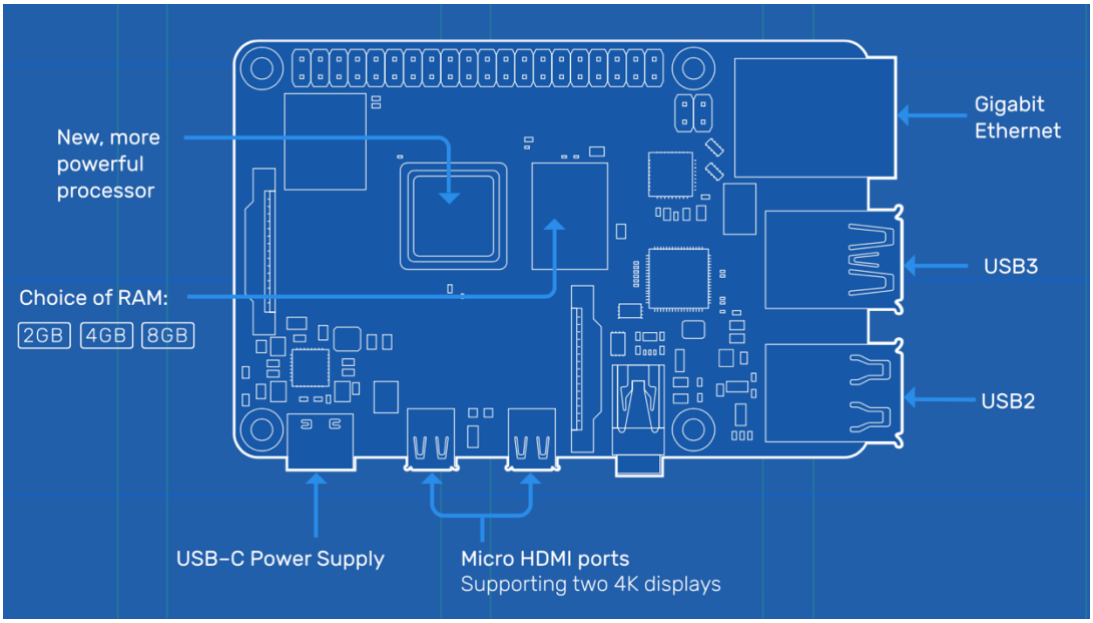
Στο κεφάλαιο αυτό, γίνεται μια περιγραφή του υλικού συστήματος και των συστατικών που το απαρτίζουν, καθώς και η χρήση της Αρχιτεκτονικού Διαγράμματος Υλικού, το οποίο δείχνει της της συνδέσεις που έχουν γίνει και το τελικό αποτέλεσμα.

2.1 Raspberry Pi 4

Το raspberry pi είναι μια σειρά μικρών υπολογιστών με μία πλακέτα, τα οποία αναπτύχθηκαν στο Ηνωμένο Βασίλειο από το ίδρυμα Raspberry Pi σε συνεργασία με την Broadcom. Το έργο Raspberry pi, ξεκίνησε με την προϋπόθεση να προωθήσει τη διδασκαλία της βασικής επιστήμης των υπολογιστών στα σχολεία και τις αναπτυσσόμενες χώρες. Το αρχικό μοντέλο, της, έγινε πιο δημοφιλές από ότι αναμενόταν, για χρήσεις όπως η ρομποτική. Πλέον, χρησιμοποιείται ευρέως σε πολλούς τομείς, λόγω τον πλεονεκτημάτων που προσφέρει, το χαμηλό του κόστος, την ευελιξία και τον ανοικτό σχεδιασμό. [21]

Λόγω αυτών των πλεονεκτημάτων, το raspberry pi είναι το πλέον κατάλληλο για τη χρήση σε συστήματα συλλογής δεδομένων. Ο ανοικτός του σχεδιασμός δίνει ευελιξία στην ένωση διαφόρων αισθητήρων πάνω στο raspberry pi, από της οποίους μπορούμε να πάρουμε ενδείξεις από το περιβάλλον στο οποίο βρίσκεται σε πραγματικό χρόνο.
Για το σύστημα που υλοποιήθηκε, έγινε χρήση του τελευταίου μοντέλου της σειράς Raspberry Pi, το Raspberry Pi 4.

Κάτοψη του συστήματος:



Εικόνα 2.1.1: Raspberry pi 4 Tech Specs [22]

Προδιαγραφές: [22]

Το raspberry pi 4 αποτελείται από τον επεξεργαστή Broadcom BCM2711, Quad core Cortex-A72 (ARM v8)64-bit SoC @ 1.5GHz, από μια 4GB LPDDR4-3200 SDRAM. Έχει ενσωματωμένα 2.4 GHz και 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet. Επίσης 2 θύρες USB 3.0, 2 θύρες USB 2.0, 40 pin GPIO header (συμβατά με προηγούμενες πλακέτες), 2 θύρες micro-HDMI, θύρα οθόνης MIPI DSI 2 λωρίδων, θύρα κάμερας MIPI CSI 2 λωρίδων, 4-πολική θύρα στερεοφωνικού ήχου και σύνθετου βίντεο. H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode), OpenGL ES 3.1, Vulkan 1.0, μια υποδοχή κάρτας Micro-SD για φόρτωση λειτουργικού συστήματος και αποθήκευσης δεδομένων, 5V DC μέσω υποδοχής USB-C (ελάχιστο 3A*), 5V DC μέσω κεφαλίδας GPIO (ελάχιστο 3A*), Ενεργοποιημένη τροφοδοσία μέσω Ethernet (PoE) (απαιτείται ξεχωριστό PoE HAT) και η θερμοκρασία λειτουργίας κυμαίνεται από 0-50 βαθμοί C στο περιβάλλον.

*Μιας καλής ποιότητας τροφοδοτικό 2.5 A μπορεί να χρησιμοποιηθεί εάν το περιφερειακά USB καταναλώνουν λιγότερο από 500 mA συνολικά.

2.2 Αισθητήρες

Για την υλοποίηση του συστήματος, έγινε χρήση 4 αισθητήρων, οι οποίοι παίρνουν τιμές από το περιβάλλον στο οποίο βρίσκονται, οι οποίες τιμές είναι απαραίτητες για την ανάλυση ρίσκου, εφόσον ο κορονοϊός μεταδίδεται μέσω του αέρα. Επίσης γίνεται και η χρήση ενός Μικροφώνου και ενός Analog2Digital Converter.

2.2.1 DHT22[15]

Ο αισθητήρας DHT22, είναι ένας βασικός ψηφιακός αισθητήρας θερμοκρασίας και υγρασίας, χαμηλού κόστους.

Εικόνα 2.2.1 DHT22[15]

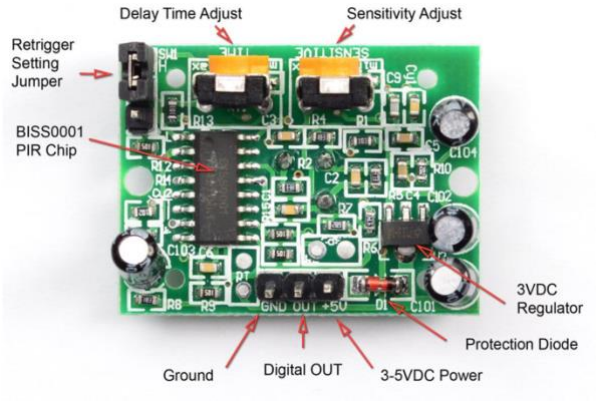
1	Vcc	Power supply 3.5V to 5.5V.
2	Data	Outputs both Temperature and Humidity through serial Data.
3	NC	No Connection and hence not used.
4	Ground	Connected to the ground of the circuit.

Πίνακας 2.2.2 DHT22 Specifications [15]

Ο αισθητήρας αποτελείται από 4 ποδαράκια. Το πρώτο ενώνεται με την πηγή τροφοδοσίας, όπου οι τιμές που παίρνει κυμαίνονται από 3.5-5.5V. Το δεύτερο ενώνεται σε κάποιο pin του raspberry pi και δίνει τις τιμές που παίρνει ο αισθητήρας. Το τρίτο δεν έχει κάποια χρήση. Το τέταρτο ενώνεται με την γείωση στο raspberry pi για να κλείσει το σύστημα.

2.2.2 PIR Motion Sensor[16]

Ο αισθητήρας PIR, ανιχνεύει οποιαδήποτε κίνηση συμβεί, είτε αυτή είναι από άνθρωπο, είτε από ζώο. Όταν κάτι περνά μπροστά από τον αισθητήρα, τότε αυτός ανιχνεύει την αλλαγή στη θερμοκρασία του φόντου και στη συνέχεια ενεργοποιεί μια τάση εξόδου. Αυτή τάση διαβάζεται από το raspberry pi στο οποίο είναι ενωμένο.
Είναι μικρός σε μέγεθος αισθητήρας, φθηνός, χαμηλής ισχύος και εύκολος στη χρήση.



Εικόνα 2.2.3 [a] Pir Top View [16]

Εικόνα 2.2.3 [b] Pir Camera View [16]

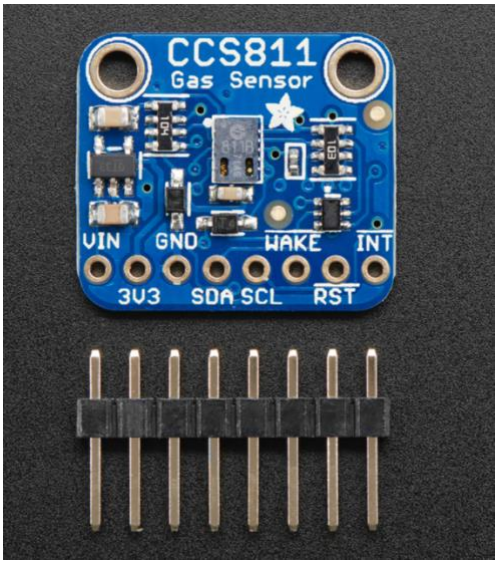
1	Ground	Connected to the ground of the circuit.
2	Data	Digital Output .
3	Vcc	Power supply 3-5V.

Πίνακας 2.2.4 PIR Specifications [16]

Ο αισθητήρας αποτελείται από 3 ποδαράκια. Το πρώτο ενώνεται στην γείωση του raspberry pi, το δεύτερο δίνει το output (δηλαδή 0 ή 1 ανάλογα με την είσοδο) και ενώνεται σε κάποιο GPIO pin του raspberry pi. Το τρίτο ενώνεται με την πηγή της τροφοδοσία και παίρνει τιμές από 3-5V.

2.2.3 CCS811[17]

Ο CCS811 είναι ένας ψηφιακός αισθητήρας χαμηλής-ισχύος, ο οποίος ενσωματώνει μια λύση αισθητήρα αερίου για την ανίχνευση χαμηλών επιπέδων VOCs (οργανικές χημικές ουσίες που έχουν υψηλή τάση ατμών σε θερμοκρασία δωματίου) [13] που βρίσκονται συνήθως σε εσωτερικούς χώρους. Με μονάδα μικροελεγκτή (MCU) και μετατροπέα Analog-to-Digital, γίνεται παρακολούθηση του τοπικού περιβάλλοντος και παροχή ένδειξης της ποιότητας του εσωτερικού αέρα μέσω ισοδύναμης εξόδου CO2 ή TVOc (Η συνολική μάζα, συνήθως σε χιλιοστόγραμμα ανά κυβικό μέτρο, των οργανικών ενώσεων που συλλέγονται στον αέρα.), μέσω τυπικής ψηφιακής διεπαφής I2C. Στην παρούσα εργασία μας ενδιαφέρει μόνο το CO2 που υπάρχει σε μια αίθουσα.



Εικόνα 2.2.5 CCS811 [17]

1	VIN	Power Pin. Since the sensor uses 3.3V, we have included an onboard voltage regulator that will take 3-5VDC and safely convert it down.
2	3Vo	3.3V output from the voltage regulator, you can grab up to 100mA from this. No connection.
3	GND	Connected to the ground circuit.
4	SDA	I2C data pin, connected to microcontroller I2C data line.
5	SCL	I2C clock pin, connected to microcontroller I2C clock
6	Wake	Wake up pin for the sensor. No connection.
7	RST	Reset pin. No connection.
8	INT	Interrupt-output pin. No connection.

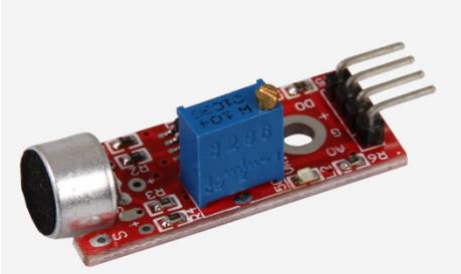
Πίνακας 2.2.6 CCS811 Specifications [17]

Ο αισθητήρας αποτελείται από 8 ποδαράκια. Το πρώτο ενώνεται με την πηγή τροφοδοσίας, όπου οι τιμές που παίρνει κυμαίνονται από 3-5V. Το τρίτο ενώνεται με την γείωση. Το τέταρτο ενώνεται με το SDA1 I2C pin του raspberry pi. Το πέμπτο ενώνεται με το SCL1 I2C του raspberry pi. Το δεύτερο, το έκτο, το έβδομο και το όγδοο δεν ενώνονται κάπου.
(βλ. Πίνακα 2.2.6)

2.2.4 KY037[18] And ADC1115[19]

Ο KY-037 είναι ένας αισθητήρας ανίχνευσης ήχου, ο οποίος ανιχνεύει δυνατούς ήχους (σαν να χτυπάνε πόρτες). Διαθέτει αναλογική και ψηφιακή έξοδο. Η ευαισθησία του αισθητήρα μπορεί να ρυθμιστεί μέσω ενός ελεγκτή. Ο συγκεκριμένος αισθητήρας έχει τρία λειτουργικά εξαρτήματα στην πλακέτα του. Αυτά είναι η μονάδα αισθητήρα στο μπροστινό μέρος, η οποία μετρά το τρέχον περιβάλλον(τους ήχους που παίρνει) και το εξάγει ως αναλογικό σήμα στη δεύτερη μονάδα, που είναι ο ενισχυτής. Αυτό ενισχύει το σήμα ανάλογα με την αντίσταση που έχει ρυθμιστεί στο περιστροφικό ποτενσιόμετρο και το κατευθύνει στην αναλογική έξοδο της μονάδας.

Σημείωση: Το σήμα είναι ανεστραμμένο. Εάν μετρηθεί μια υψηλή τιμή, αυτό έχει ως αποτέλεσμα χαμηλότερη τιμή τάσης στην αναλογική έξοδο.[18]



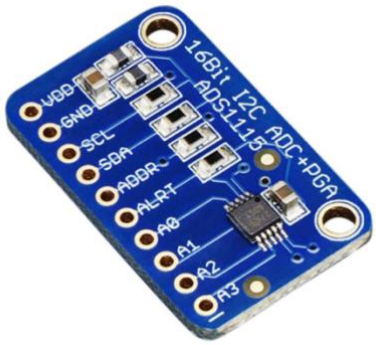
Εικόνα 2.2.7 KY037[18]

1	A0	Analog Output
2	G	Connected to the ground of the circuit.
3	+	Power supply 3.3V to 5.5V.
4	DS0	Digital Output. Not Connected.

Πίνακας 2.2.8 KY037 Specifications[18]

Ο αισθητήρας αποτελείται από 4 ποδαράκια. Το πρώτο δίνει την αναλογικά έξοδο και ενώνεται με ένα Analog-to-Digital Converter. Το δεύτερο ενώνεται με την γείωση. Το τρίτο ενώνεται με την πηγή τροφοδοσίας όπου οι τιμές που παίρνει κυμαίνονται από 3.3-5.5V. Το τέταρτο δίνει την ψηφιακή έξοδο (δηλαδή 0 ή 1) και ενώνεται σε κάποιο GPIO pin του raspberry pi.

Ο ADS1115 είναι ακριβή, χαμηλής κατανάλωσης, 16-bit, συμβατός με I2C, Analog-To-Digital Converter, ο οποίος μετατρέπει την αναλογική είσοδο που παίρνει από το μικρόφωνο σε ψηφιακή. Με κατάλληλες εντολές στο δίαυλο I2C (3ο και 5ο pin στο raspberry pi), οι τιμές αναλογικής τάσης μπορούν να μετρηθούν σε έως και 4 εισόδους (είσοδοι στον ADS1115: 7, 8, 9, 10) με ακρίβεια έως και 16 bit. Το αποτέλεσμα της μέτρησης εξάγεται κωδικοποιημένο στο δίαυλο I2C. [19]



Εικόνα 2.2.9 ADS1115 [19]

1	VDD	Power supply 2V to 5.5V.
2	GND	Connected to the ground of the circuit.
3	SCL	I2C clock pin, connected to microcontroller I2C clock
4	SDA	I2C data pin, connected to microcontroller I2C data line.
5	ADDR	No Connection.
6	ALRT	No Connection.
7	A0	Analog input 1. Connect with KY-037 A0.
8	A1	Analog input 2. No Connection.
9	A2	Analog input 3. No Connection.
10	A3	Analog input 4. No Connection.

Πίνακας 2.2.10 CCS811 Specifications [19]

Ο αισθητήρας αποτελείται από 10 ποδαράκια. Το πρώτο ενώνεται με την πηγή τροφοδοσίας, όπου οι τιμές κυμαίνονται από 2-5.5V. Το δεύτερο ενώνεται με την γείωση. Το τρίτο ενώνεται με το SCL1 I2C του raspberry pi. Το τέταρτο ενώνεται με το SDA1 I2C pin του raspberry pi. Το έβδομο ενώνεται με την αναλογική είσοδο του KY-037. Το πέμπτο, το έκτο, το όγδοο, το ένατο και το δέκατο δεν ενώνονται με το raspberry pi.

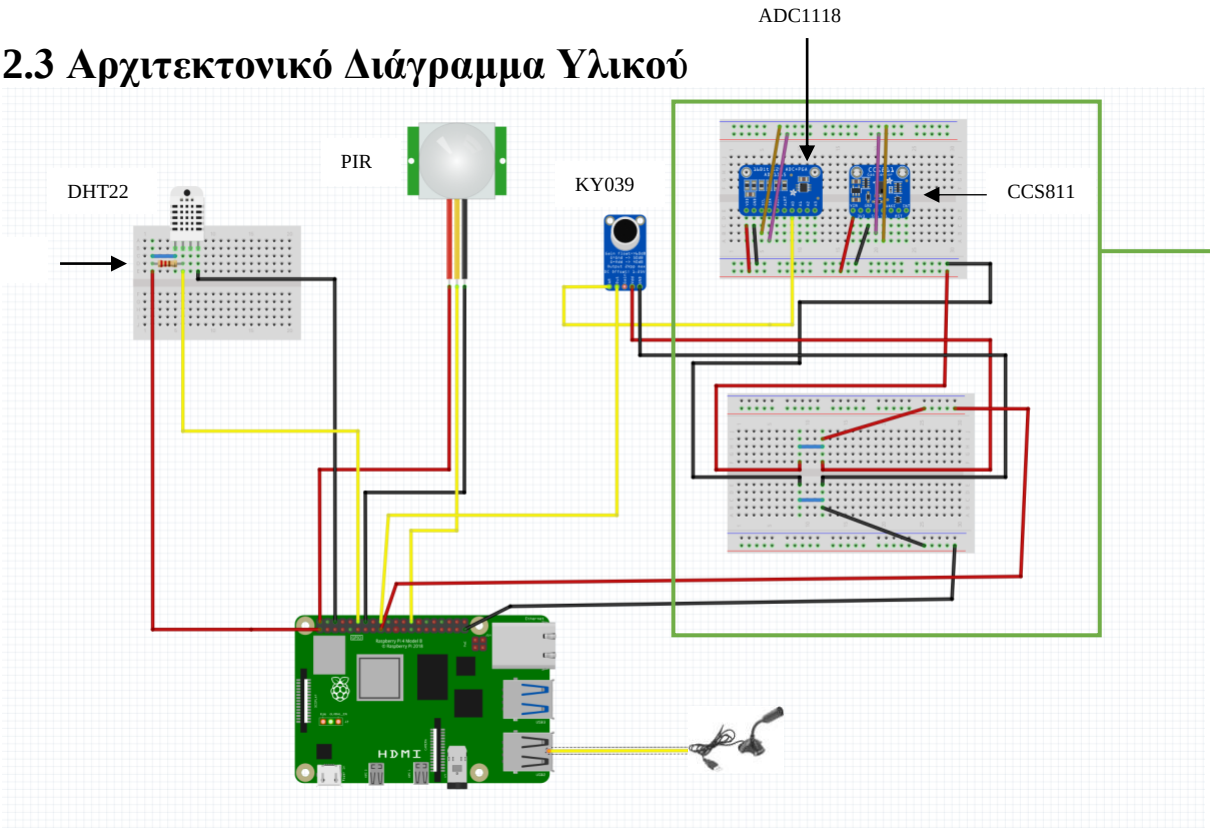
2.3 Ψηφιακό Μικρόφωνο [20]

Το ESPERANZA EH179 MICROPHONE USB SCREAM είναι ένα USB μικρόφωνο, για όλες τις εφαρμογές ήχου, συμπεριλαμβανομένου και του VoIP, με πολύ καθαρό ήχο.

Εικόνα 2.2.11 ESPERANZA EH179 MICROPHONE USB SCREAM[20]

Αποτελείται από USB καλώδιο το οποίο ενώνεται σε κάποια USB θύρα του raspberry pi.

2.3 Αρχιτεκτονικό Διάγραμμα Υλικού



Εικόνα 2.3.1 Διάγραμμα Συσκευής

Πιο πάνω βλέπουμε τις συνδέσεις που έχουν γίνει μεταξύ των αισθητήρων και του raspberry pi.

Ενώσεις:

1. DHT22: Vcc -> Pin 1
   Data -> Pin 12(GPIO18)
   Ground -> Pin 6
2. PIR Motion Sensor: Ground -> Pin 14
   Data -> Pin 26(GPIO7)
   Vcc -> Pin 2
3. Ο αισθητήρας CCS811, KY037 και ADC1118 είναι συνδεδεμένοι παράλληλα και τροφοδοτούνται όλοι από την ίδια πηγή, το pin 17 και γείωση τους, για να κλείσει το κύκλωμα, ενώνεται στο pin 39. Η digital έξοδος του αισθητήρα KY037 ενώνεται με το pin 18(GPIO24). Η analogue έξοδος του αισθητήρα ενώνεται με τον ADC1115 στην είσοδο A0. Οι έξοδοι SDA και SCL των ADC1115 και CCS811 ενώνονται με παράλληλη σύνδεση με τα pin 2(SDA1 I2C) και pin 3 (SCL1 I2C). (βλέπε Εικόνα 2.3.2)
4. Το ψηφιακό μικρόφωνο είναι ενωμένο με μια θύρα USB στο raspberry pi.

Κεφάλαιο 3

Λεπτομέρειες Υλοποίησης

Στο κεφάλαιο αυτό θα γίνει μια περιγραφή των λεπτομερειών υλοποίησης, όπως είναι η συλλογή δεδομένων, η ανάλυση ρίσκου, η ανάλυση δεδομένων, καθώς και ο τρόπος παρουσίασης των δεδομένων μέσω API.

3.1 Συλλογή Δεδομένων

Ο κώδικας περισυλλογής δεδομένων είναι γραμμένος στην γλώσσα προγραμματισμού python 3 και τα δεδομένα αποθηκεύονται σε ένα αρχείο της SQLite[24]. Πιο συγκεκριμένα, καθώς ξεκινά το πρόγραμμα δίνονται από τον χρήστη στοιχεία, όπως το μήκος, το πλάτος και το ύψος του δωματίου, αν τα άτομα που βρίσκονται στην αίθουσα φορούν μάσκα ή όχι, αν βρίσκονται σε κατάσταση ηρεμίας, αν στέκονται, αν τραγουδούν ή αν κάνουν κάποιου είδους άσκηση, είτε ελαφριάς μορφής, είτε μέτριας, είτε βαρείας (ανάλογα με το είδος της αίθουσας που βρίσκεται η συσκευή, δηλαδή αν βρίσκεται σε αίθουσα πανεπιστημίου τότε τα άτομα βρίσκονται σε κατάσταση ηρεμίας). Επίσης ακόμα μια παράμετρος που δίνεται είναι η ηλικία των ατόμων που βρίσκονται στην αίθουσα.
Στην συνέχεια γίνεται σύνδεση με την βάση δεδομένων και δημιουργεί, εφόσον δεν υπάρχει, τον πίνακα στο αρχείο της βάσης στον οποίον θα γράφονται τα δεδομένα.

Μόλις δημιουργηθεί και η βάση, ξεκινά ένα while-loop, το οποίο τερματίζει σε κάποια χρονική στιγμή μέσα στην μέρα, την οποία ορίζει ο χρήστης. Μέσα σε αυτό το while loop ανοίγουν 6 διαφορετικά threads(τα οποία είναι synchronized μεταξύ τους), τα οποία κάθε x δευτερόλεπτα παίρνουν τιμές, οι οποίες αθροίζονται και αποθηκεύονται σε κάποια μεταβλητή, επίσης σε διαφορετική μεταβλητή αποθηκεύεται και το πλήθος των τιμών που παίρνουμε (εξαιρούνται τα threads 2 και 4). Παρακάτω βλέπουμε τα 6 threads και τις ιδιότητες τους:

Thread 1: Ενεργοποιείται ο DHT22[15] με την χρήση της βιβλιοθήκης adafruit_dht[25] και board[26], ο οποίος παίρνει τιμές για την θερμοκρασία και την υγρασία που υπάρχει στην ατμόσφαιρα, κάθε 2 δευτερόλεπτα.

Thread 2: Ενεργοποιείται ο PIR[16] sensor με τη χρήση της βιβλιοθήκης board[26], όπου μετρά τα άτομα που μπαίνουν στην αίθουσα, κάθε 3 δευτερόλεπτα. Υπολογίζουμε ότι ένα άτομο για να μπει σε μια αίθουσα χρειάζεται 3-5 δευτερόλεπτα, με το σενάριο ότι μπορεί ταυτόχρονα να μπουν 2 ή και περισσότερα άτομα την ίδια στιγμή, τότε παίρνουμε το μικρότερο χρονικό διάστημα που είναι τα 3 δευτερόλεπτα. Αν όμως παίρναμε σε διάστημα μικρότερο των 3 δευτερολέπτων, τότε ένα άτομο που θα πέρναγε θα το μετρούσε περισσότερο από μια φορά.

Thread 3: Ενεργοποιείται ο CCS811[17] με την χρήση των βιβλιοθηκών adafruit_ccs811[27], busio[28] και board[26], ο οποίος μετρά το διοξείδιο του άνθρακα που βρίσκεται στην αίθουσα κάθε 2 δευτερόλεπτα, έτσι ώστε να έχουμε παραπάνω μετρήσεις, εφόσον όσο περισσότερες μετρήσεις έχουμε, αντίστοιχα έχουμε και μεγαλύτερη ακρίβεια.

Thread 4: Ενεργοποιείται το Bluetooth του raspberry pi[21], όπου ανιχνεύει όλες τις συσκευές που έχουν ενεργοποιημένο το Bluetooth και βρίσκονται στην αίθουσα. Από αυτές αναγνωρίζει ποιες είναι κινητές συσκευές. Γίνεται χρήση της βιβλιοθήκης bluetooth[23] και παίρνει τιμές κάθε μισό δευτερόλεπτο, εφόσον οι τιμές που παίρνουμε μεταβάλλονται συνεχώς, ως το τελευταίο δευτερόλεπτο και θέλουμε να έχουμε μεγαλύτερη ακρίβεια.

Thread 5: Ενεργοποιούνται οι KY037[18] και ADS1115[19] με την χρήση των βιβλιοθηκών busio[28] και adafruit_ads1x15.ads1115[29]. Ο KY037 παίρνει την τάση του ήχου και ο ADS1115 μετατρέπει την digital τιμή σε analog. Ο αισθητήρας παίρνει τιμές κάθε 0.05 δευτερόλεπτα.

Thread 6: Ενεργοποιεί ένα digital μικρόφωνο, όπου με της εντολής soundmeter[30] παίρνει τον μέσο όρο των RMS [7] (Root Mean Square) (είναι η default τιμή που παίρνει η εντολή soundmeter) values για όλο το χρονικό διάστημα που τρέχει.

Πριν τερματίσει κάθε thread βγαίνει ο μέσος όρος των τιμών που έχουν παρθεί από τους αισθητήρες, κατά το διάστημα που το thread είναι ενεργοποιημένο, και αποθηκεύεται σε μια global μεταβλητή και επιστρέφει. Τα threads μέσα στο χρονικό διάστημα που τρέχουν ανοίγουν και κλείνουν κάθε 15 λεπτά. Ο λόγος που παίρνουμε μετρήσεις κάθε 15 λεπτά είναι για να δούμε πως αντιδρά το δωμάτιο ανά χρονικά διαστήματα. Κάθε φορά που κλείνουν αποθηκεύουν τα δεδομένα που υπάρχουν στις global μεταβλητές στην βάση δεδομένων που δημιουργήθηκε στην αρχή του προγράμματος, μαζί με την χρονική στιγμή που γίνεται η αποθήκευση. Επίσης, για μεγαλύτερη ακρίβεια, για να βρούμε πόσα άτομα βρίσκονται στην αίθουσα χρησιμοποιούμαι δύο τρόπους (Pir και Bluetooth), έτσι το τελικό αποτέλεσμα ισοδυναμεί με τον μέσο όρο των αποτελεσμάτων που έχουμε πάρει από το Pir (thread 2) και το Bluetooth (Thread 4), αν και υπάρχει η περίπτωση, το Bluetooth να μην έχει πάρει κάποια τιμή, λόγω του ότι μπορεί κανένα άτομο να μην έχει ανοικτό το Bluetooth του κινητού του, λόγω του ότι ο αισθητήρας pir παίρνει κάθε κίνηση που γίνεται, οι τιμές του ίσως να είναι περισσότερες από τα άτομα που βρίσκονται στο δωμάτιο, τότε ισοσταθμίζεται και ο μέσος

όρος που βγαίνει είναι πιο κοντά στα άτομα που όντως βρίσκονται στο δωμάτιο. Επιπρόσθετα τόσο για τις τιμές που παίρνουμε από το thread 5, όσο και από το thread 6 μετατρέπουμε τις τιμές τους σε decibel.

Μόλις τερματίσει το πρόγραμμα τα δεδομένα τρέχουν από τον αλγόριθμου εκτίμησης ρίσκου και από τον αλγόριθμο ανάλυσης ρίσκου.

Αν για κάποιο λόγο ο χρήστης θελήσει να τερματίσει το πρόγραμμα πιο νωρίς με την εντολή Control-C τότε γίνεται trap το signal και τα δεδομένα τρέχουν από τον αλγόριθμου εκτίμησης ρίσκου και από τον αλγόριθμο ανάλυσης ρίσκου και τερματίζει.
(βλέπε Παράρτημα Α, για τον κώδικα)

Ψευδοκώδικας Συλλογής Δεδομένων:

INPUT: Handler function. This function take the signal Cntrl-C (signum = 2) from the keyboard. It connects to the database, calculate the average of all metrics and save the result in the database. It closes the database and execute from the run program.
OUTPUT: Execute from the run program.
Begin:
FUNCTION handler (signum, frame):

```
    SET con TO connect_db.get_db(name) #connect to the database

    # check IF there is at least one value and calculate the average of each one

    IF counter_temperature Not Equal 0:
        SET temperature_average TO temperature_total/counter_temperature

    IF counter_humidity Not Equal 0:
        SET humidity_average TO humidity_total/counter_humidity

    IF counter_co2 Not Equal 0:
        SET co2_average TO co2_total/counter_co2

    IF counter_decibel Not Equal 0:
        SET decibel_average TO decibel_total/counter_decibel

    IF counter_micro Not Equal 0:
```

SET micro_average TO micro total / counter_micro

SET micro_average TO (10.0 * logarithm of (micro_average))

 # insert the metrics in the database. The timestamp has declared, when database had been generated, and the insertion of the timestamp is automated with the metrics insertion

SET sql TO ''' INSERT INTO metrics (Area,Volume,BluetoothPeopleCounter,PirPeopleCounter,Temperature,Humidity,Co2,Decibel,Microphone) VALUES(?,?,?,?,?,?,?,?,?) '''

SET val TO (area, volume, bluecounter, peoplecount, temperature_average, humidity_average, co2_average, decibel_average, micro_average)

CALL accurancy.Accurancy(db_name) #find the Risk Assessment

CALL data_analysis(width, length, height, breathing_rate, age, mask, db_name) #find the Risk Analysis

CALL exit(1) #quit from the program

End

INPUT: Temandhum function. This function enables the DHT22 sensor, calculate temperature and humidity average for 15 minutes.

OUTPUT: The average of temperature and humidity

Begin:

FUNCTION temandhum()

#Initial the dht device, with data pin connected to:

SET dhtDevice TO adafruit_dht.DHT22(board.D18)

WHILE True:

#Break when the minutes IN time belong to one of the subcategories

IF minutes are equal to 15 or 30 or 45 or 59 then:

#check IF there is at least one value of temperature or humidity

IF counter_temperature EQUALS 0 or counter_humidity EQUALS 0:

#if there is not exist initialize the average of temperature and humidity with zero and quit

```
                SET temperature_average TO 0
                SET humidity_average TO 0
                break

            #calculate the average of temperature
            SET temperature_average TO temperature_total/counter_total

            #calculate the average of humidity
            SET humidity_average TO humidity_total/counter_humidity

            #killing the process via terminal
            os.system("sudo pkill libgpiod_pulsei")

            break

TRY:

            #calculate the temperature values from the serial port
            temperature_total increase by dhtDevice.temperature and temperature_total

            #calculate the counter of the temperature values
            counter_temperature increase by counter_temperature and 1

            #calculate the humidity values for the serial port
            humidity_total increase by dhtDevice.humidity and humidity_total

            #calculate the counter of the humidity values
            counter_humidity increase by counter_humidity and 1

#if a runtime error occur from devices malfunction, sleep FOR 2 second and continue
  from the beginning
except RuntimeError as error:
            Sleep for 2 seconds
            continue

#if an exception occur, quit from device and raise the error
except Exception as error:
            dhtDevice.exit()
            raise error
```

```
            #sleep FOR 2 seconds
            Sleep for 2 seconds
        end while
End
```

INPUT: PeopleCounter function. Enables the PIR sensor and calculate the people number for 15 minutes.
OUTPUT: The number of the people

```
Begin:
FUNCTION peopleCounter()

        #Initial the pir device, with data pin connected to:
        SET pir TO digitalio.DigitalInOut(board.CE1)

        WHILE True:

            #Break when the minutes IN time belong to one of the subcategories
            IF minutes are equal to 15 or 30 or 45 or 59 then:
                break

            #If pir's value is true then increase the people counter by one
            IF pir's value be EQUALS True:
                SET peoplecount TO increase by peoplecount + 1

                    CALL sys.exit

                #sleep FOR 3 seconds
                Sleep for 3 seconds
        end while
End
```

INPUT: Co2 function. Enable CCS811 sensor and calculate the average of room's co2 for 15 minutes
OUTPUT: The average of Co2 in the room

```
FUNCTION Co2()

        SET i2c TO busio.I2C(board.SCL, board.SDA)  # uses board.SCL and board.SDA
```

#Initial the ccs811 device, with data pin connected to:
SET ccs811 TO adafruit_ccs811.CCS811(i2c)

#pass if the sensor is not ready, because sensor CCS811 wants some seconds to start taking metrics
WHILE not ccs811 sensor's data be ready:
    PASS
end while

WHILE True:

    #Break when the minutes IN time belong to one of the subcategories
    IF minutes are equal to 15 or 30 or 45 or 59 then:

        #check IF there is at least one value of co2
        IF counter_co2 EQUALS 0:

            #if there is not exist initialize the average of co2 with zero
            SET co2_average TO 0
            break

        #calculate the average of co2
        SET co2_average TO co2_total div counter_co2
        break

    #calculate the co2 values from the serial port
    co2_total increase by co2_total and ccs811.eco2

    #calculate the co2 counter
    counter_co2 increase by counter_co2 and 1

    #sleep FOR 2 seconds
    Sleep for 2 seconds

end while
End

INPUT: Bluetoothcounter Function. Enable the bluetooth search on the Rpi device and detect all the mobile devices that have turn on the bluetooth and count them.
OUTPUT: The number of the devices

FUNCTION bluetoothcounter()

WHILE True:

#Break when the minutes IN time belong to one of the subcategories
IF minutes are equal to 15 or 30 or 45 or 59 then:
break

#detect the devices with turned on bluetooth
SET nearby_devices TO bluetooth.discover_devices(duration=8, lookup_names=True, flush_cache=True, lookup_class=False)

#set the number of the mobile devices that are turned on
SET bluecounter TO len(nearby_devices)

Sleep for 0.5 seconds

end while
End

INPUT: Decibel function. Enable KY037 microphone and ADS1115 Analog2Digital Converter and calculate the average of the room's sound level for 15 minutes. The Rpi has not analog GPIO input, because of that we used of the Analog2Digital Converter.
OUTPUT: The average of the decibels in the room.

FUNCTION decibel()

 # Create the I2C bus
SET i2c TO busio.I2C(board. SCL, board. SDA)

 # Create the ADC object using the I2C bus
SET ads TO ADS.ADS1115(i2c, address=0x48)

 # Create single-ended INPUT on channels
SET chan0 TO AnalogIn(ads, ADS.P0)
SET chan1 TO AnalogIn(ads, ADS.P1)
SET chan2 TO AnalogIn(ads, ADS.P2)
SET chan3 TO AnalogIn(ads, ADS.P3)

```
    WHILE True:

        #Break when the minutes IN time belong to one of the subcategories
        IF minutes are equal to 15 or 30 or 45 or 59 then:

            #check IF there is at least one value of decibel
            IF counter_decibel EQUALS 0:
                SET decibel_average TO 0
                break

            #calculate the average of decibel
            SET decibel_average TO decibel_total div counter_decibel
            break

        #take the voltage from the serial port
        SET analog  TO chan0.voltage
        #calculate with the used of a formula the total of decibels
        decibel_total increase by decibel_total and (20.0 * logarithm(3 div analog))
        #caclulate the decibel counter
        counterd increase by counterd and 1

        #sleep for 0.05 seconds
        Sleep for 0.05

    end while
End

--------------------------------------------------------------------------------
INPUT: digMicro function. Enable the digital microphone and calculate the average of the room decibels for 15 minutes.
OUTPUT: The average of the room's decibels.

FUNCTION digMicro()

    WHILE True:

        #Break when the minutes IN time belong to one of the subcategories
        IF minutes are equal to 15 or 30 or 45 or 59 then:
```

#check IF there is at least one value of decibel

IF counter_micro>0:

#calculate the average of decibels

SET micro_average TO round(micro_total div counter_micro)

ELSE:

SET micro TO 0

break

CALL os.system("soundmeter --collect --seconds 10 > test") #collect the sound volume for 10 seconds and send the results in a test.txt file

CALL os.system("cat test | grep avg | tr -s ' ' | cut -d ' ' -f3 > res.txt") #from the test.txt file, apply a formula and extract the sound level number and send the result in a res.txt file

#open the res.txt file

with open('res.txt') as f:

#read the inside of the file

SET metr TO f.readlines()

#close the file

f.close()

##calculate the sound level values

SET micro_total TO micro_total + int(metr[0])

##caclulate the decibel counter

SET counter_micro TO counter_micro + 1

end while

End

INPUT: Main_task function. In this function, the threads are initialized and ran synchronized. When, they are finished return.

OUTPUT: The data from the threads, that are ran.

FUNCTION main_task()

#Initialize the threads

SET t1 TO threading.Thread(target=temandhum)

SET t2 TO threading.Thread(target=peoplecounter)

SET t3 TO threading.Thread(target=Co2)

SET t4 TO threading.Thread(target=bluetoothcounter)

SET t5 TO threading.Thread(target=decibel)

SET t6 TO threading.Thread(target=digMicro)

#Run the threads
CALL t1.start()
CALL t2.start()
CALL t3.start()
CALL t4.start()
CALL t5.start()
CALL t6.start()

Καλεί τις συναρτήσεις που ενεργοποιούν τους αισθητήρες

#Wait all the threads to finished, creating a synchronization

t1.join()

t2.join()

t3.join()

t4.join()

t5.join()

t6.join()

INPUT: Main function. It take some data from the user, do some checks and run the main task function. Take the data and put them in a database. Repeat, until the time being. the time that the user set.
OUTPUT: -

Function main():

#set the project name, input from the user

OUTPUT ("Give the name of the Project")

SET namei TO INPUT()

SET name TO namei + ".db"

create_tables.create_table(name) #connect to the database and create the database table with the project name

#user's inputs and checks

OUTPUT ("Give the length")

SET length TO float(INPUT())

OUTPUT("Give the width")

```
SET width TO float(INPUT())

OUTPUT ("Give the height")
SET height TO float(INPUT())

OUTPUT ("Give the human condition: \n 1.Resting \n 2.Standing \n 3.Singing 4.Ligth_Exercise \n 5.Moderate_Exercise \n 6.Heavy_Exercise \n Give number 1-6")

SET breathing_rate TO int(INPUT())

WHILE breathing_rate<1 and breathing_rate>6:
        OUTPUT ("Wrong Input! Give number from 1 to 6!")
        SET breathing_rate TO int(INPUT())

OUTPUT ("Give the age of the people inside the room.")
SET age TO int(INPUT())

WHILE age<0 and age> 100:
        OUTPUT ("Wrong Input! Give number from 0 to 100")
        SET age TO int(INPUT())

OUTPUT ("Mask wearing. \n 1.No Mask \n 2.Surgical Mask \n 3.Hybrid-Fabric Mask")
SET mask TO int(INPUT())

WHILE mask<1 and mask>3:
        OUTPUT ("Wrong Input! Give number from 1 to 3!")
        SET mask TO int(INPUT())

SET area TO length * width
SET volume TO area * height
SET peoplecount TO 0
```

Main Loop

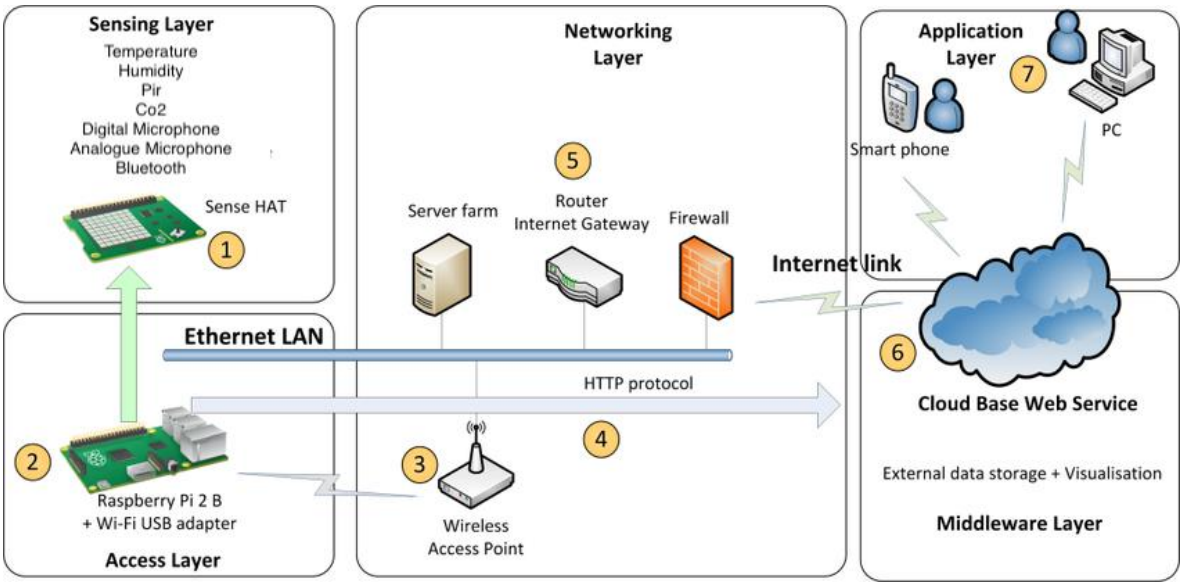
```
WHILE True:
        #Sleep when the minutes IN time belong to one of the subcategories
        IF minutes are equal to 15 or 30 or 45 or 59 then:
                Sleep for 61 seconds, the minutes 15, 30, 45 and 59 the result always is 0, so we
                don't want to call main_task at this time, that the reason for the 61 waiting
                seconds

        CALL main_task()
```

← Καλεί τις επιμέρους συναρτήσεις

```
            #people counter is the average of the counter of the values tha pir sensor and the bluetooth
are take
            SET peoplecount TO round((peoplecount+bluecounter)/2)

            #connect to the database
            SET con TO connect_db.get_db(name)

            #insert the metrics in the database
            SET sql TO ''' INSERT INTO metrics
(Area,Volume,BluetoothPeopleCounter,PirPeopleCounter,Temperature,Humidity,Co2,Decibel,Microph
ne) VALUES(?,?,?,?,?,?,?,?,?) '''

            SET val TO (area, volume, bluecounter, peoplecount, temperature_average,
humidity_average, co2_average, decibel_average, micro_average)

            #Break if the time is equal with the time that user set for ending the program
            IF time are equal to 14:15 then:

                  CALL accurancy.Accurancy(db_name) #find the Risk Assessment
                  CALL data_analysis(width, length, height, breathing_rate, age, mask, db_name)
#find the Risk Analysis
                  break

            #sleep for 61 seconds
            Sleep for 61 seconds
      end while
End
```

Παρακάτω έχουμε την αρχιτεκτονική λογισμικού, όπου έχουμε το sensing layer, που είναι οι αισθητήρες μας, ενωμένοι πάνω στο raspberry pi, το οποίο μπορούμε να τρέξουμε σαν server, και οι χρήστες να πάρουν δεδομένα από την Βάση Δεδομένων με τη χρήση ενός κατάλληλου API μέσω τηλεφώνου ή ηλεκτρονικού υπολογιστή.

Αρχιτεκτονική Λογισμικού:



Εικόνα 3.1.1 Στρώμα αισθητήρων

3.2 Ανάλυση Ρίσκου

Η ανάλυση δεδομένων είναι ένας αλγόριθμος, ο οποίος κάνει χρήση έτοιμων πράξεων, οι οποίες έχουν δημιουργηθεί με βάση στατιστικές μελέτες, με την βοήθεια επιδημιολόγων και μαθηματικών[2][3]. Οι παράμετροι που παίρνει που παίρνει αυτός ο αλγόριθμος είναι στοιχεία για την ανθρώπινη συμπεριφορά, όπως είναι ο ρυθμός αναπνοής, η αναπνευστική δραστηριότητα, ο τύπος ή η αποτελεσματικότητα της μάσκας και αν υπάρχει εφαρμογή ή συμμόρφωση της μάσκας. Επίσης, παίρνει φυσικά χαρακτηριστικά των εσωτερικών χώρων, όπως είναι ο όγκος του δωματίου, ο εξαερισμός, το φιλτράρισμα, οι ρυθμοί ανακυκλοφορίας και η υγρασία. Στην ουσία ο αλγόριθμος αυτός παίρνει τα δεδομένα από την βάση δεδομένων που έχουμε δημιουργήσει και αυτών που εισήγαγε ο χρήστης στην αρχή (βλέπε ενότητα 3.1) και όταν φθάσει στην χρονική στιγμή που ο χρήστης όρισε για να τερματίσει το πρόγραμμα, περνά τις εγγραφές που υπάρχουν στην βάση δεδομένων από την φόρμουλα που εξηγήσαμε πιο πάνω, η οποία περιέχει όλες αυτές τις πράξεις και βγάζει κατά πόσο, εάν υπάρχει έστω και ένα άτομο στο δωμάτιο το οποίο είναι μολυσμένο, εάν το δωμάτιο είναι υψηλού ρίσκου ή όχι. Τα αποτελέσματα που εξάγονται τα αποθηκεύει σε μια άλλη βάση. (βλέπε Παράρτημα Β, για τον κώδικα). Επιπρόσθετα, χρησιμοποιήσαμε δύο διαφορετικές εξισώσεις, όπου η διαφορά των δύο είναι ότι στην μια υπολογίζουμε με βάση τα άτομα και τον όγκο του δωματίου, ενώ στην άλλη υπολογίζουμε με βάση το διοξείδιο του άνθρακα και παίρνουνε το μέγιστο χρόνο που το δωμάτιο είναι ασφαλές και βγάζουμε τον μέσο όρο.

Εξίσωση:

la = Outdoor air exchange rate
Q = volume * la
lr = Recirculation air exchange rate
Qr = volume / lr
QplusQr = Q + Qr
Zp = Q / QplusQr

pf = Aerosol filtration efficiency
lf = pf * Qr / volume
r = Respiratory aerosol radius

humadjrad = $r*((0.4/(1-humidity/100))^{1/3})$

Cq = decibel
RH = Viral deactivation rate

humadjadeact = RH * humidity /50

VSr = (2/9) * 1100 / (1.86 * pow(10,(-5))) * 9.8 * (1/pow(1000,3)) * $humadjrad^2$
tempVSr = VSr * 60 * 60 / 1000
lc = la + humadjadeact + lf + tempVSr / heigth
fd = Qb / (lc * volume)
inflecofroom = Cq * fd

ba = $((Qb * maskpm)^2)$ * Cq * age / (volume * lc)
e = Risk tolerance

Εξίσωση 1: C68 = e / ((Npers-1) * ba)
tmax = C68 * (1 + sqrt(1+4/(lc*C68)))/2

Εξίσωση 2: background_co2 = 350
maxt = e * 38000 * lc / ((Co2-background_co2) * Qb * Cq * age * maskpm * la)

3.3 Παρουσίαση Δεδομένων μέσω API

Η παρουσίαση των δεδομένων στον χρήστη γίνεται μέσω API. Το raspberry pi, μέσω της βιβλιοθήκης flask[31], γίνεται sever και οποιοσδήποτε χρήστης μπορεί να κάνει GET request και να πάρει τα δεδομένα που είναι γραμμένα στη βάση δεδομένων. Για να μπορέσει ο χρήστης να κάνει το GET request πρέπει να έχει την εξής μορφή: http://ip_address: port/user_name:password/query. Πρέπει να δίνεται από τον χρήστη το ip_address που έχει το raspberry pi, το port στην δική μας περίπτωση είναι το 5000, το user_name είναι admin και το password είναι k3p8z5. Στην συνέχεια έχει τις εξής επιλογές:

/metrics : δίνει όλες τις μετρήσεις από την αρχή μέχρι το τέλος της ημέρας ή ως εκεί που έχει γράψει.

/params? : μπορείς να εισάγεις έως 10 παραμέτρους. Οι 9 παράμετροι έχουν την μορφή param[1-9]=any_col, όπου το any_col μπορεί να είναι οποιαδήποτε στήλη από τη βάση, είτε η θερμοκρασία, είτε το διοξείδιο του άνθρακα ή οποιοδήποτε άλλο (area, volume, blueP, pirP, temperature, humidity, co2, decibel, adecibel). Η άλλη παράμετρος που μπορεί να δοθεί έχει τη μορφή records=num, της οποίας της δίνεις κάποιο αριθμό και σου βγάζει τα πρώτα n αποτελέσματα που είναι γραμμένα στη βάση.

/risk : μας δίνει την ημερομηνία και ώρα από την στιγμή που έχει ξεκινήσει ο αλγόριθμος μέχρι την στιγμή που παίρνουμε έχει εγγραφεί μια γραμμή τιμών στη βάση δεδομένων, το διάστημα αυτό σε ώρες, τον μέγιστο αριθμό ωρών για να μην είναι μολυσμένο το δωμάτιο με βάση τον αλγόριθμο και μια τιμή του τύπου «0(όχι) ή 1(ναι)», ανάλογα με το αν το δωμάτιο είναι μεταδοτικό ή όχι, για κάθε χρονικό διάστημα.
(βλέπε Παράρτημα Δ, για τον κώδικα)

Με αντίστοιχο τρόπο μπορούμε να δούμε και τα δεδομένα της ανάλυση ρίσκου.

Κεφάλαιο 4

Αποτελέσματα

Το κεφάλαιο αυτό περιέχει τα αποτελέσματα των πειραμάτων που έχουν γίνει.

4.1 Πειραματικά Αποτελέσματα

Στο κεφάλαιο αυτό, παρουσιάζονται τα αποτελέσματα που έχουμε πάρει από τον αλγόριθμο ανάλυσης δεδομένων. Παρακάτω, βλέπουμε ένα πίνακα, ξεχωριστό για κάθε μέρα, με τις μετρήσεις που έχουμε πάρει, καθώς και το αποτέλεσμα του. Για την στήλη «People in the class», για περισσότερη ακρίβεια, βγάλαμε τον μέσο όρο από τις μετρήσεις που πήραμε από τον αισθητήρα PIR και το Bluetooth(λόγω του ότι μπορεί κανένα άτομο να μην έχει ανοικτό το Bluetooth του κινητού του και λόγω του ότι ο αισθητήρας pir παίρνει κάθε κίνηση που γίνεται, οι τιμές του ίσως να είναι περισσότερες από τα άτομα που βρίσκονται στο δωμάτιο, τότε ισοσταθμίζεται και ο μέσος όρος που βγαίνει είναι πιο κοντά στα άτομα που όντως βρίσκονται στο δωμάτιο) και το στρογγυλοποιήσαμε. Επίσης, όλες οι μετρήσεις έχουν βγει από τον μέσο όρο, μέσα στο χρονικό διάστημα το οποίο τρέχουν (π.χ. η πρώτη γραμμή του Πίνακα 4.1.1 της στήλης Temperature, από το διάστημα 15:00-15:15, έχουμε συλλέξει 450 τιμές για την θερμοκρασία του δωματίου και ο μέσος όρος τους φαίνεται παρακάτω). Στις γραφικές παραστάσεις παρατηρούμε τα αποτελέσματα από την κάθε εξίσωση και τον μέσο όρο τους.

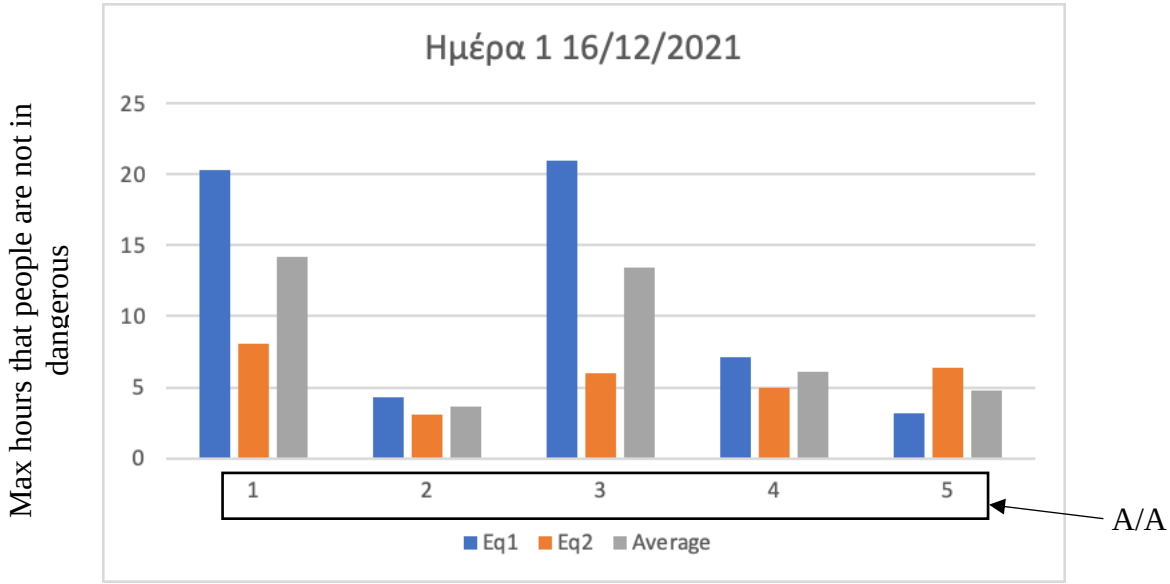
Εξίσωση 1: Τα αποτελέσματα τις βασίζονται κυρίως στον αριθμό των ατόμων που βρίσκονται στην αίθουσα και τον όγκο του δωματίου. (βλέπε Ενότητα 3.3)
Εξίσωση 2: Τα αποτελέσματα τις βασίζονται στο διοξείδιο του άνθρακα που εγκλείεται στην αίθουσα. (βλέπε Ενότητα 3.3)

Τιμές κοινές για όλες τις μετρήσεις: breathing_rate= “resting”, age = 23, mask = “no_mask”
Επίσης όλα τα πειράματα έγιναν σε μια αίθουσα του Πανεπιστημίου.

Ήμερα 1, 16/12/2021:

A/A	Hours	Time range (in hours)	Area m^2	Volume m^3	People in the class	Temperature °C	Humidity %	Co2 ppm	Decibel db	Max Hours, that people are not in dangerous	Risk
1	15:00-15:15	0.25	40	140	1	26.57	55.94	460.88	54.66	14.18	0
2	15:00-15:30	0.50	40	140	6	26.55	57.75	640.41	54.97	3.67	0
3	15:00-15:45	0.75	40	140	1	26.71	62.83	502.99	53.81	13.46	0
4	15:00-15:59	0.98	40	140	4	26.62	64.71	536.20	54.82	6.05	0
5	15:00-16:15	1.25	40	140	8	26.61	66.75	495.18	54.83	4.79	0

Πίνακας 4.1.1 Μετρήσεις ημέρας 1



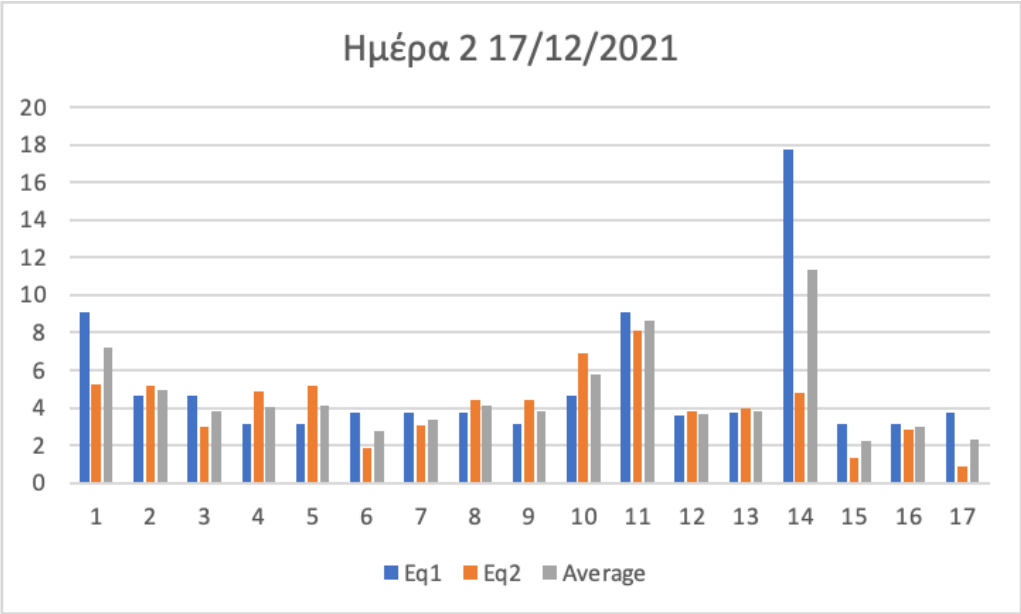
Γραφική Παράσταση 4.1.1 Αποτελέσματα των δύο εξισώσεων και ο μέσος όρος τους.

Για την ημέρα 1 βλέπουμε ότι για όλες τις ώρες, αν υπάρχει μέσα στην αίθουσα ένα άτομο το οποίο είναι μολυσμένο, δεν υπάρχει κάποιο ρίσκο στο να κολλήσει κάποιο άλλο άτομο. Αυτό οφείλεται στο χαμηλό αριθμό ατόμων που βρίσκονται στην αίθουσα, στο χαμηλό διοξείδιο του άνθρακα, καθώς και στο ότι το χρονικό διάστημα μέσα στην αίθουσα είναι σχετικά μικρό. Επίσης, ακόμα και την στιγμή που στο δωμάτιο υπάρχει μόνο ένα άτομο, δεν κινδυνεύει να κολλήσει από άλλο άτομο, αλλά μπορεί να κολλήσει από τα υπολείμματα των προηγούμενων.

Ήμερα 2, 17/12/2021:

Hours	Time range (in hours)	Area m^2	Volume m^3	People in the class	Temperature °C	Humidity %	Co2 ppm	Decibel db	Max Hours, that people are not in dangerous	Risk
11:00-11:15	0.25	40	140	3	26.56	35.46	499.61	54.81	7.18	0
11:00-11:30	0.5	40	140	5	26.68	35.06	500.10	54.84	4.93	0
11:00-11:45	0.75	40	140	5	26.77	34.64	613.33	54.81	3.8	0
11:00-11:59	0.98	40	140	7	26.75	34.02	509.24	54.84	4.02	0
11:00-12:15	1.25	40	140	7	26.74	33.87	501.36	54.83	4.15	0
11:00-12:30	1.5	40	140	6	26.69	34.09	775.16	54.83	2.79	0
11:00-12:45	1.75	40	140	6	26.71	34.27	606.53	54.83	3.4	0
11:00-12:59	1.98	40	140	6	26.66	34.64	527.72	54.83	4.09	0
11:00-13:15	2.25	40	140	7	26.74	34.38	527.72	54.83	3.79	0
11:00-13:59	2.98	40	140	5	26.61	34.43	463.23	55.42	5.77	0
11:00-14:06	3.11	40	140	3	26.64	45.26	446.95	55.85	8.62	0
11:00-14:15	3.25	40	140	3	26.64	45.26	446.95	55.78	3.69	0
11:00-14:30	3.5	40	140	6	26.47	33.15	664.25	55.64	3.84	1
11:00-14:45	3.75	40	140	2	27.13	32.20	510.92	55.27	11.31	0
11:00-14:59	3.98	40	140	7	27.37	31.87	932.07	55.28	2.25	1
11:00-15:15	4.25	40	140	7	27.35	32.66	623.76	55.05	3.02	1
11:00-15:30	4.5	40	140	6	27.29	33.39	1214.51	55.05	2.33	1

4.1.2 Μετρήσεις ημέρας 2



Γραφική Παράσταση 4.1.2 Αποτελέσματα των δύο εξισώσεων και ο μέσος όρος τους

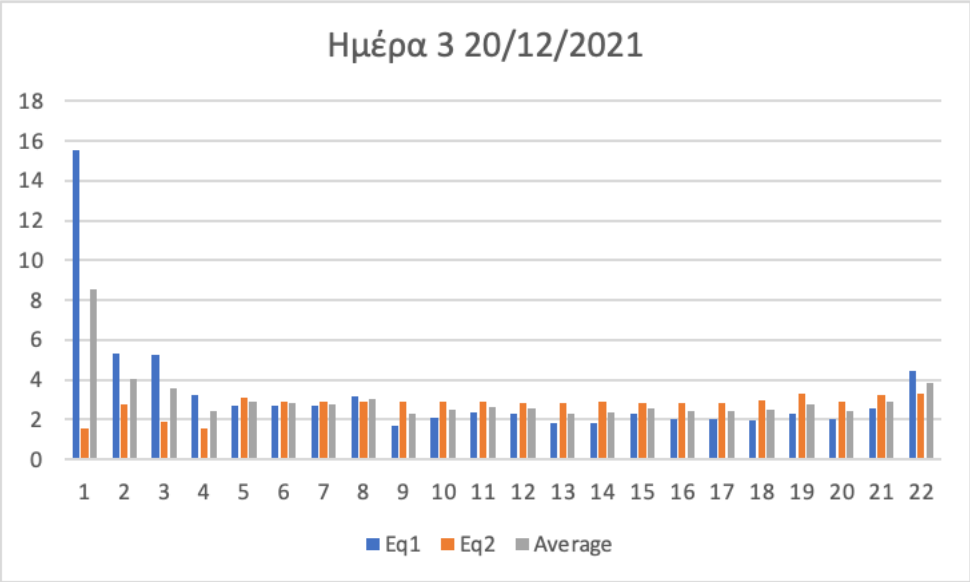
Για την ημέρα 2, παρατηρούμε ότι αν υπάρχει ένα μολυσμένο άτομο μέσα στην αίθουσα, στις 3.5 ώρες, η αίθουσα θεωρείται μολυσμένη και έτσι οποιοδήποτε άτομο βρίσκεται μέσα σε αυτή μπορεί να μολυνθεί και το ίδιο. Αυτό οφείλεται στη μείωση της υγρασίας, στο αριθμό των ατόμων που βρίσκονται μέσα στην αίθουσα, το διοξείδιο του άνθρακα. Όμως παρατηρούμε στις $3^{3/4}$ ώρες το δωμάτιο γίνεται ξανά ασφαλές, αυτό οφείλεται στο ότι τ διοξείδιο του άνθρακα μειώνεται καθώς και ο αριθμός των ατόμων που βρίσκονται στην αίθουσα. Στις 4 ώρες και μετά η αίθουσα, θεωρείται ξανά μολυσμένη για τους ίδιους λόγους που ήταν και πριν, καθώς και στο μεγάλο χρονικό διάστημα, το οποίο βρίσκονται μέσα στο δωμάτιο

Ήμερα 3, 20/12/2021:

Hours	Time range	Area m^2	Volume m^3	People in the class	Temperature °C	Humidity %	Co2 ppm	Decibel db	Max Hours, that people are not in dangerous	Risk
9:44-9:59	0.25	40	140	2	27.16	36.65	786.39	54.65	8.53	0
9:44-10:15	0.5	40	140	4	27.39	35.29	590.97	54.64	4.04	0
9:44-10:30	0.77	40	140	4	27.37	34.73	706.30	54.65	3.56	0
9:44-10:45	1.02	40	140	6	27.29	34.49	770.55	54.64	2.40	0
9:44-10:59	1.25	40	140	7	27.34	34.11	562.75	54.63	2.92	0
9:44-11:15	1.5	40	140	7	27.36	33.99	578.43	54.63	2.81	0
9:44-11:30	1.77	40	140	7	27.34	33.97	578.43	54.58	2.80	0
9:44-11:45	2.01	40	140	6	27.58	33.12	578.43	54.58	3.03	0

9:44-11:59	2.25	40	140	11	27.55	32.79	578.43	54.58	2.29	0
9:44-12:15	2.52	40	140	9	27.62	32.38	578.43	54.61	2.48	1
9:44-12:30	2.77	40	140	8	27.44	32.06	578.43	54.61	2.61	1
9:44-12:45	3.02	40	140	8	27.45	32.18	578.43	54.61	2.58	1
9:44-12:59	3.25	40	140	10	27.38	32.62	578.43	54.65	2.33	1
9:44-13:15	3.51	40	140	10	27.34	32.66	578.43	54.65	2.35	1
9:44-13:45	4.01	40	140	8	27.39	32.68	578.43	54.61	2.59	1
9:44-13:59	4.25	40	140	9	27.37	32.67	578.43	54.60	2.46	1
9:44-14:15	4.51	40	140	9	27.32	32.61	578.43	54.60	2.44	1
9:44-14:30	4.76	40	140	10	27.38	27.37	578.43	54.66	2.47	1
9:44-14:45	5.01	40	140	9	27.44	35.21	578.43	54.65	2.80	1
9:44-14:59	5.25	40	140	9	27.47	34.96	578.43	54.66	2.46	1
9:44-15:15	5.5	40	140	8	26.62	58.41	578.43	54.66	2.92	1
9:44-15:30	5.77	40	140	5	26.47	57.98	578.43	54.65	3.85	1

Πίνακας 4.1.3 Μετρήσεις ημέρας 3

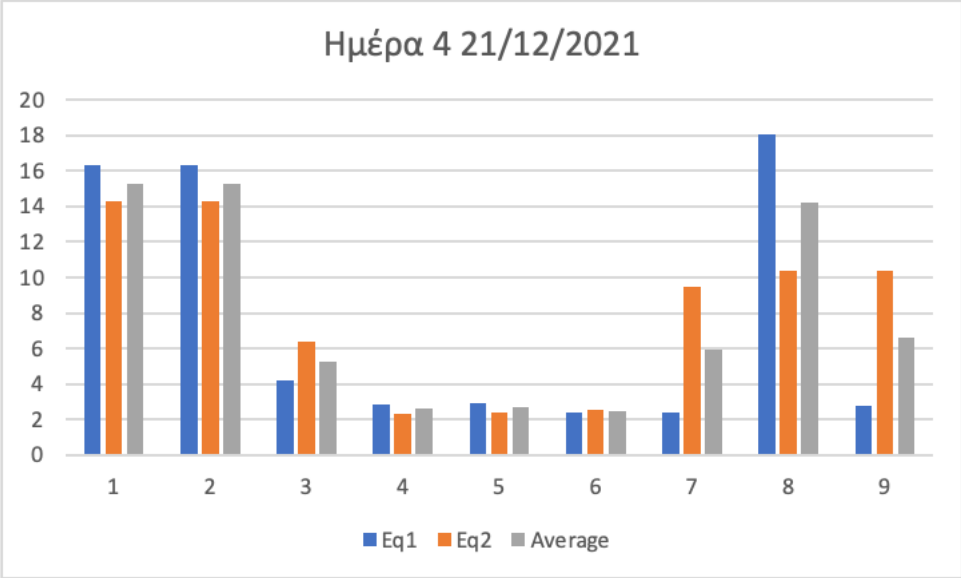


Γραφική Παράσταση 4.1.3 Αποτελέσματα των δύο εξισώσεων και ο μέσος όρος τους.

Για την ημέρα 3, παρατηρούμε ότι μετά τις 2 ώρες και 30 λεπτά, η αίθουσα θεωρείται μεταδοτική. Η διαφορά σε σχέση με την ημέρα 2 είναι ο αριθμός των ατόμων που βρίσκονται μέσα στην αίθουσα. Την 3η ημέρα τα άτομα μέσα στην αίθουσα είναι περισσότερα από την 2η ήμερα. Επίσης αν και στις τελευταίες 2 μετρήσεις η υγρασία είναι αρκετά υψηλή, στη προτελευταία γραμμή ο αριθμός των ατόμων είναι αρκετά υψηλός, για αυτό και ο μέγιστος χρόνος είναι σχεδόν 3 ώρες και στην τελευταία γραμμή, αν και ο μέγιστος χρόνος αυξάνεται, λόγω του ότι ο αριθμός των ατόμων μειώθηκε, καθοριστικό ρόλο παίζει το διάστημα που τα άτομα βρίσκονται μέσα στο δωμάτιο, όπου αντιστοιχεί σχεδόν σε 6 ώρες.

Ήμερα 4, 21/2/2021:

Hours	Time range	Area m^2	Volume m^3	People in the class	Temperature °C	Humidity %	Co2 ppm	Decibel db	Max Hours, that people are not in dangerous	Risk
9:49-10:04	0.25	40	140	0	24.1	37.6	400.0	54.65	15.28	0
9:49-10:04	0.25	40	140	0	24.1	37.6	400.0	54.65	15.28	0
9:49-10:15	0.43	40	140	5	24.91	35.20	460.63	54.65	5.28	0
9:49-10:30	0.68	40	140	7	25.42	36.18	650.52	54.07	2.62	0
9:49-10:45	0.93	40	140	7	25.53	39.95	650.52	54.30	2.67	0
9:49-14:30	4.68	40	140	9	27.13	49.97	654.66	54.32	2.49	1
9:49-14:45	4.93	40	140	9	27.24	49.08	433.24	54.22	5.93	0
9:49-14:59:06	5.16	40	140	0	27.24	49.79	426.06	54.13	14.24	0
9:49-14:59:13	5.17	40	140	8	27.24	49.80	426.06	54.15	6.58	0



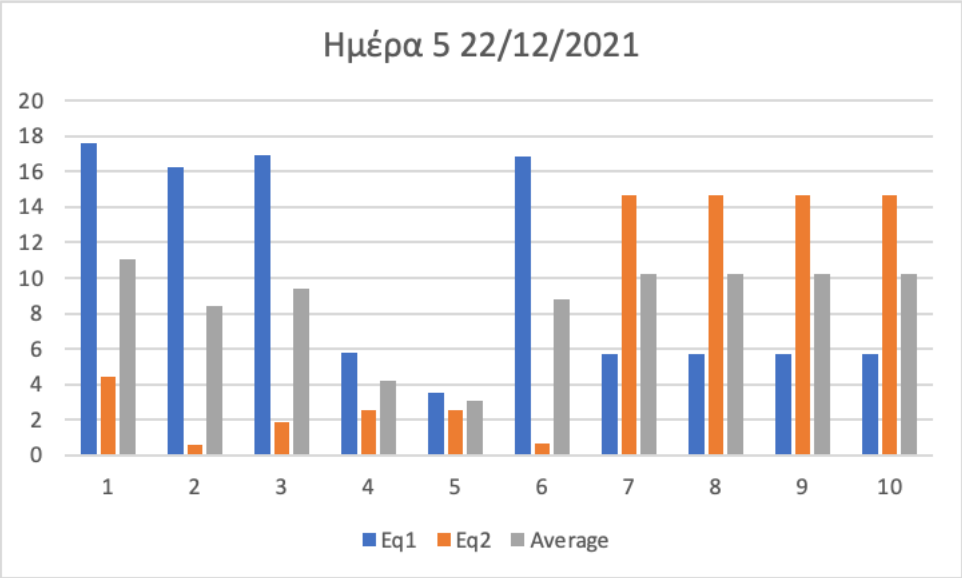
Γραφική Παράσταση 4.1.4 Αποτελέσματα των δύο εξισώσεων και ο μέσος όρος τους.

Για την ημέρα 4 παρατηρούμε ότι η αίθουσα αρχίζει και μολύνεται μετά από περίπου 4 ώρες και τριάντα λεπτά. Αυτό οφείλεται στον αυξημένο αριθμό ατόμων στην αίθουσα, καθώς και μεγάλο χρονικό διάστημα που βρίσκονται τα άτομα στην αίθουσα. Ξαναγίνεται ασφαλές στις 5 ώρες περίπου, επειδή πέφτει απότομα το διοξείδιο του άνθρακα και η υγρασία είναι σχετικά υψηλή καθώς και στο ότι για ένα χρονικό διάστημα ο αριθμός των ατόμων είναι μηδενικός.

Ήμερα 5, 22/12/2021:

Hours	Time range	Area m^2	Volume m^3	People in the class	Temperature °C	Humidity %	Co2 ppm	Decibel db	Max Hours, that people are not in dangerous	Risk
9:00-9:15	0.25	40	140	2	25.58	41.11	523.55	50.73	11.03	0
9:00-9:30	0.5	40	140	1	25.28	41.81	1590.61	50.26	8.43	0
9:00-9:45	0.75	40	140	1	25.53	42.11	746.36	50.26	9.41	0
9:00-9:59	0.98	40	140	4	25.58	41.64	638.13	50.73	4.18	0
9:00-10:15	1.25	40	140	6	25.50	41.39	638.13	50.53	3.06	0
9:00-13:59	4.98	40	140	2	25.62	37.1	1472.07	51.53	8.76	0
9:00-14:07:41	5.127	40	140	4	25.90	30.59	400.0	51.48	10.20	0
9:00-14:07:42	5.1280	40	140	4	25.90	30.59	400.0	51.48	10.20	0
9:00-14:07:43	5.1283	40	140	4	25.90	30.59	400.0	51.48	10.20	0
9:00-14:07:48	5.129	40	140	4	25.90	30.59	400.0	51.48	10.20	0

Πίνακας 4.1.5 Μετρήσεις Ημέρας 5



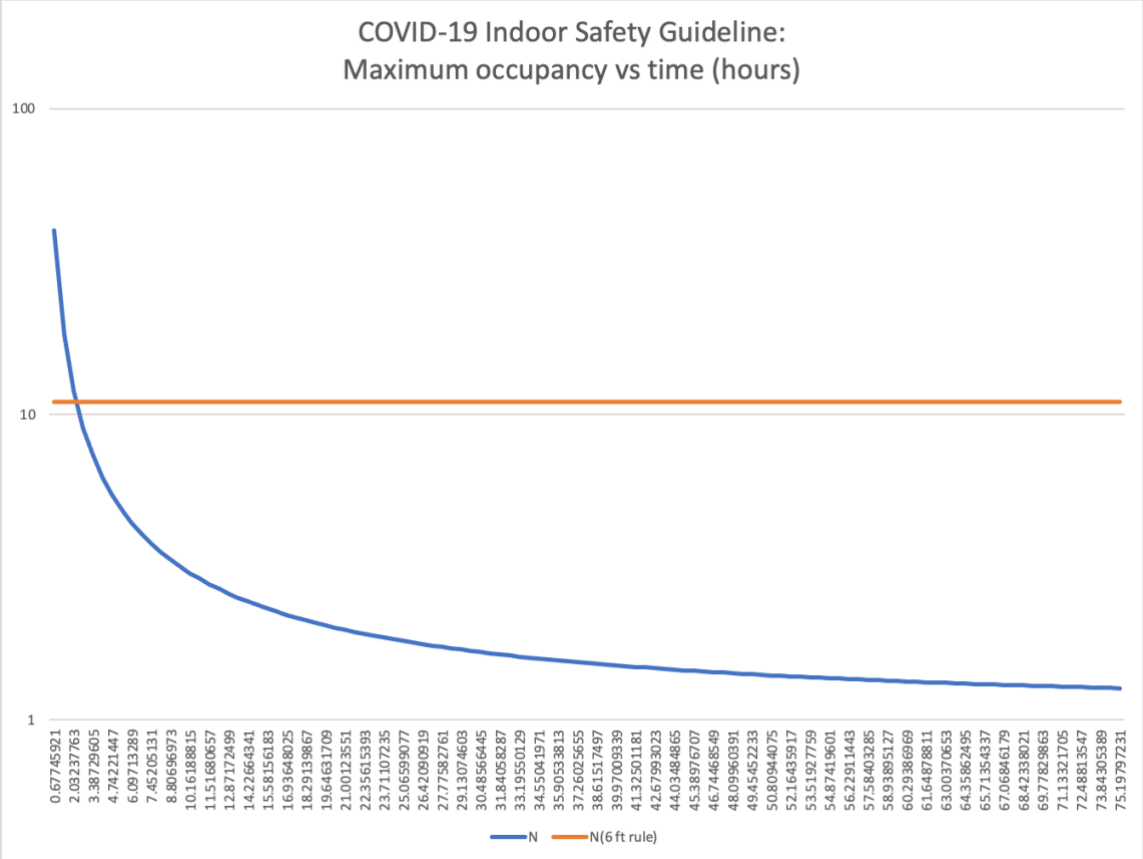
Γραφική Παράσταση 4.1.5 Αποτελέσματα των δύο εξισώσεων και ο μέσος όρος τους.

Την 5η ημέρα παρατηρούμε ότι ο ιός δεν μπορεί να μεταδοθεί. Αυτό οφείλεται στο χαμηλό αριθμό ατόμων που βρίσκονται μέσα στην αίθουσα και το χαμηλό διοξείδιο του άνθρακα. Αν και οι πρώτες τιμές του διοξειδίου του άνθρακα είναι υψηλές, το δωμάτιο παραμένει ασφαλές, εφόσον το χρονικό διάστημα που τα άτομα βρίσκονται μέσα στο δωμάτιο είναι αρκετά μικρό.

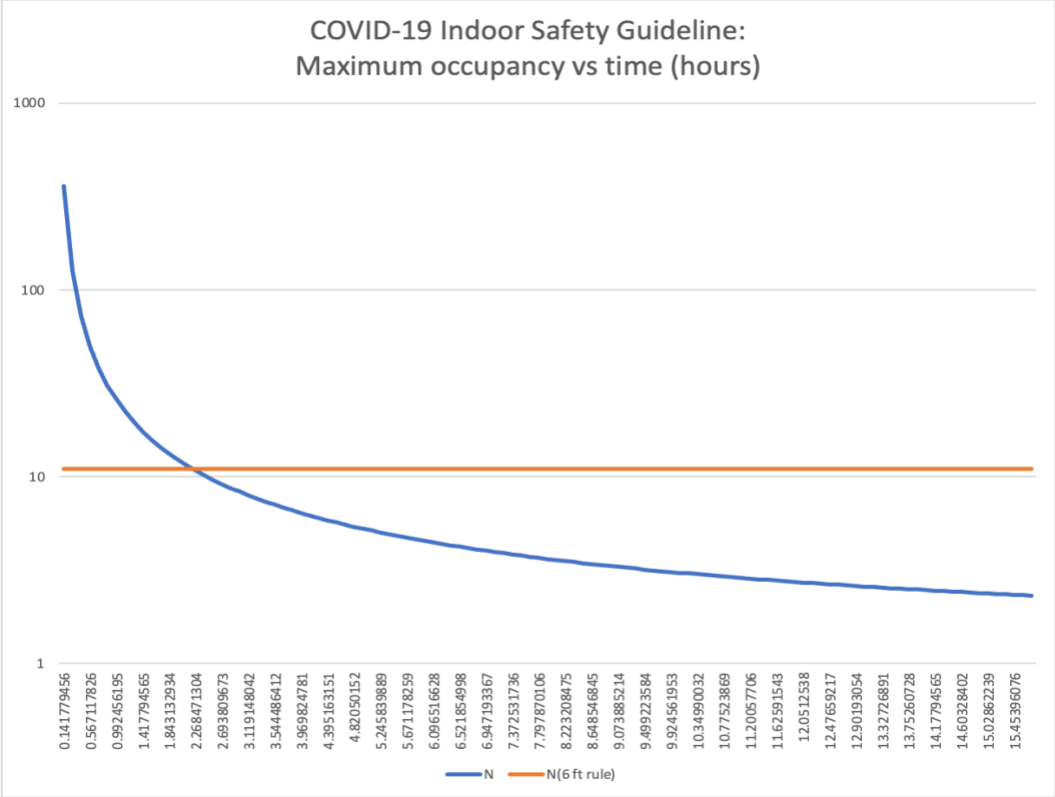
Γραφικές Παραστάσεις Υπολογισμού Ασφαλής Συγκέντρωσης Co2:

Οι παρακάτω λογαριθμικές γραφικές παραστάσεις έχουν στον κάθετο άξονα τους τη μέγιστη χωρητικότητα και στον οριζόντιο άξονα τους τον μέγιστο χρόνο έκθεσης. Οι τιμές των γραφικών παραστάσεων είναι οι μετρήσεις που έχουμε πάρει πιο πάνω. Οι γραφικές αυτές παραστάσεις έχουν δημιουργηθεί από μετρήσεις μιας χρονικής στιγμής αν θεωρητικά οι μετρήσεις αυτής της χρονικής στιγμής δεν μεταβάλλονταν, δηλαδή έμεναν σταθερές, τότε πως θα υπολογίζαμε την χωρητικότητα του δωματίου τα απόμενα χρονικά διαστήματα.

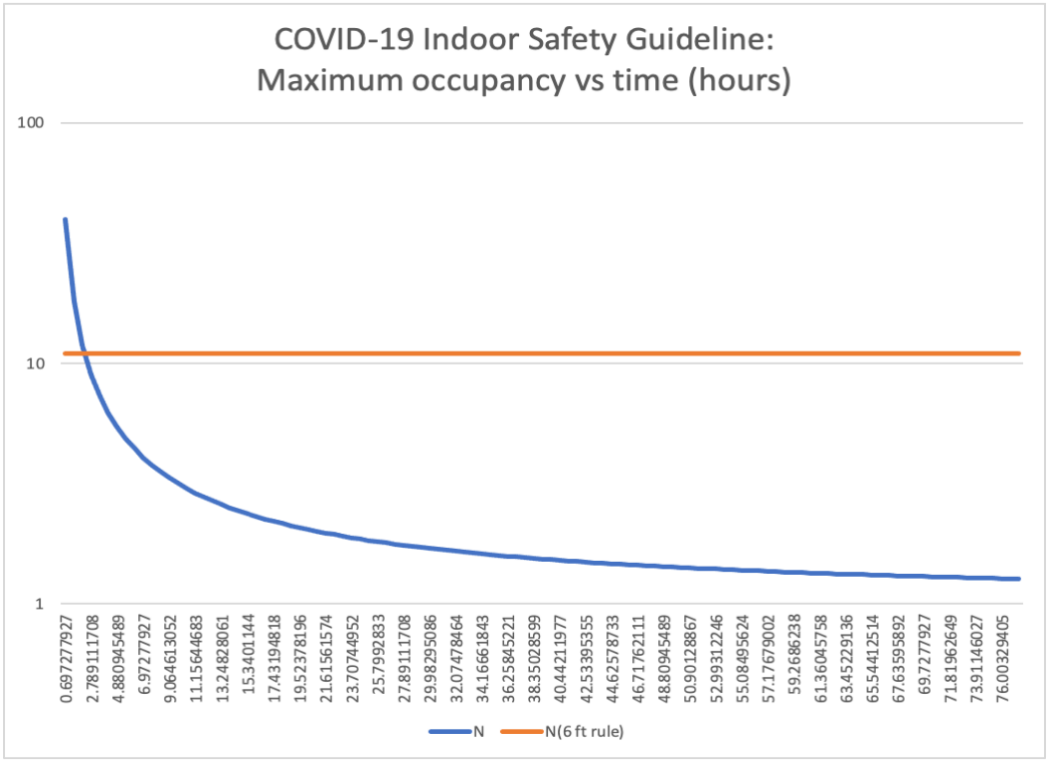
Ημέρα 1:



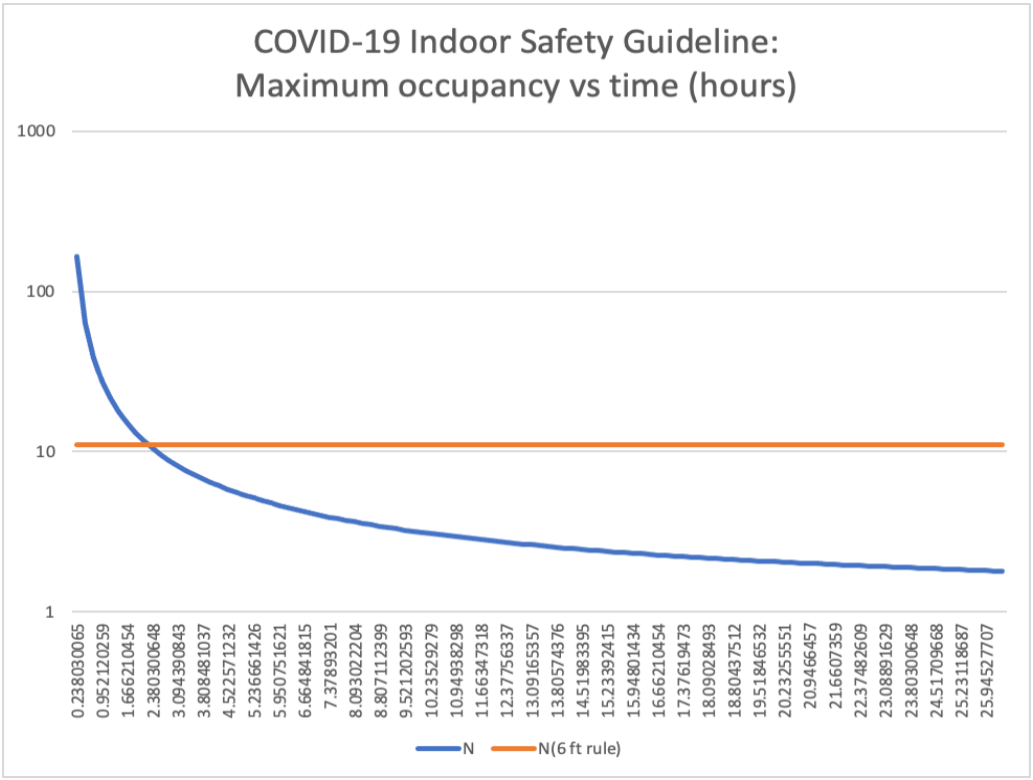
Εικόνα 4.1.6 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.25



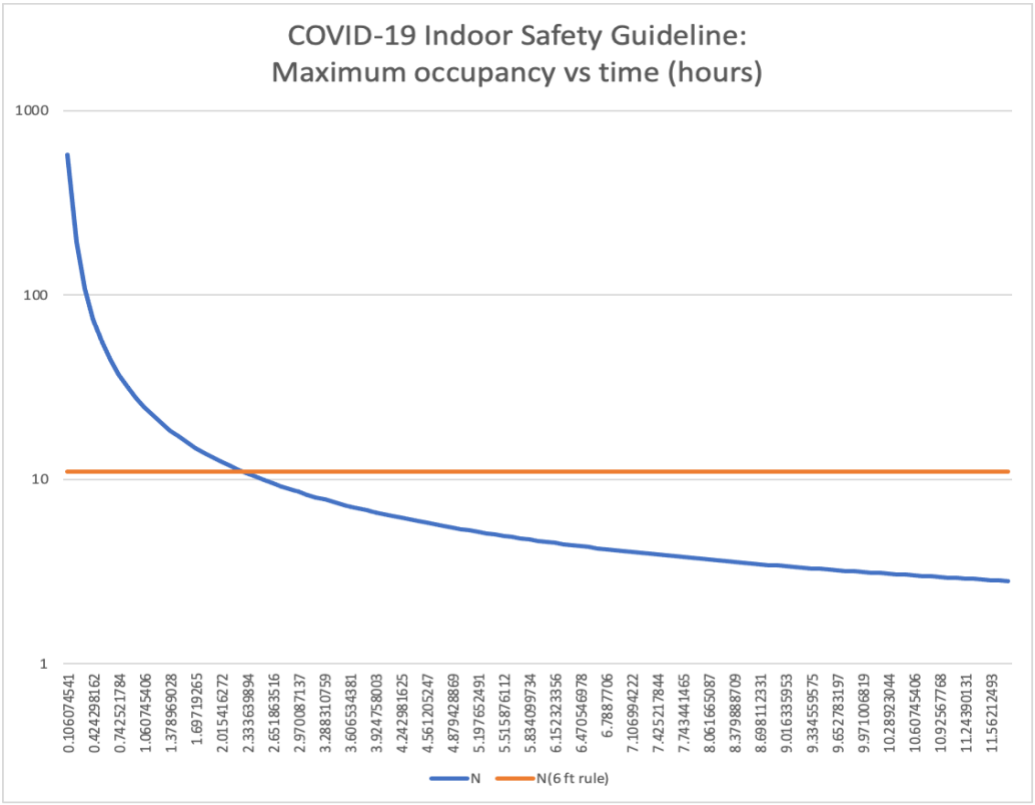
Εικόνα 4.1.7 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.50



Εικόνα 4.1.8 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.75

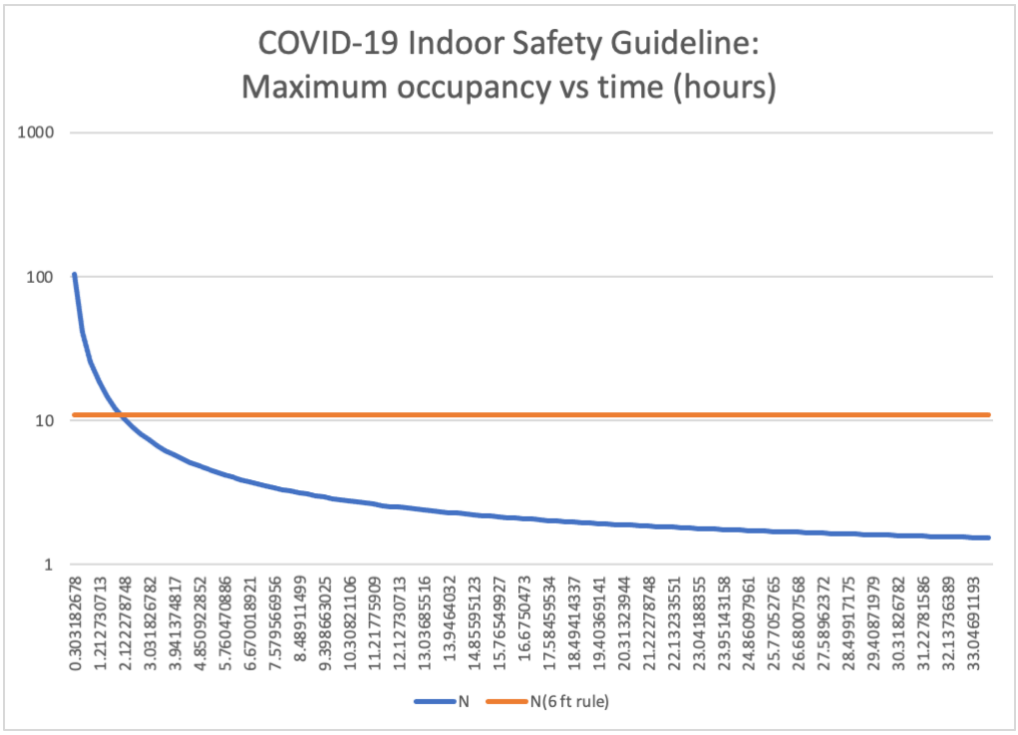


Εικόνα 4.1.9 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.98

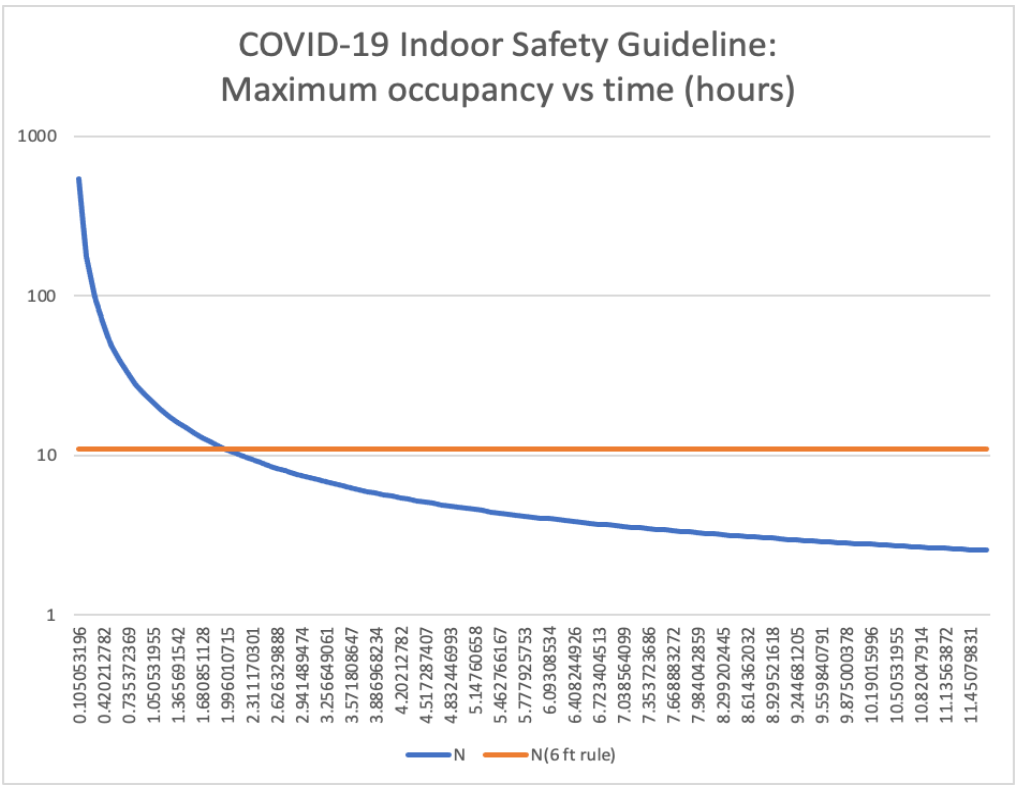


Εικόνα 4.1.9 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 1.25

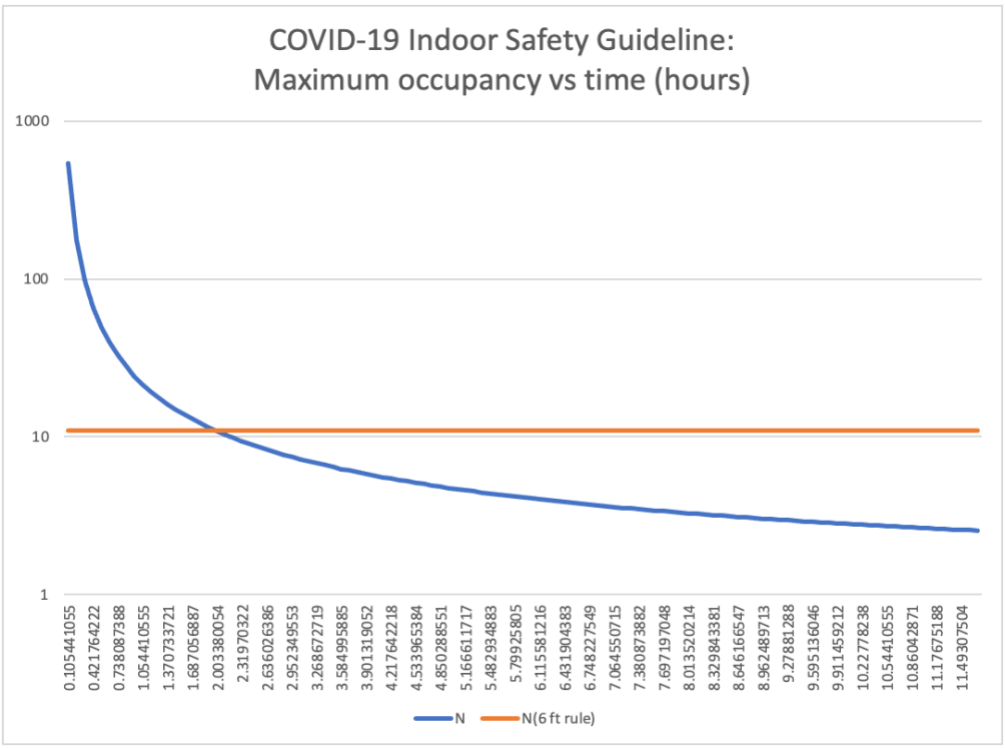
Ημέρα 2:



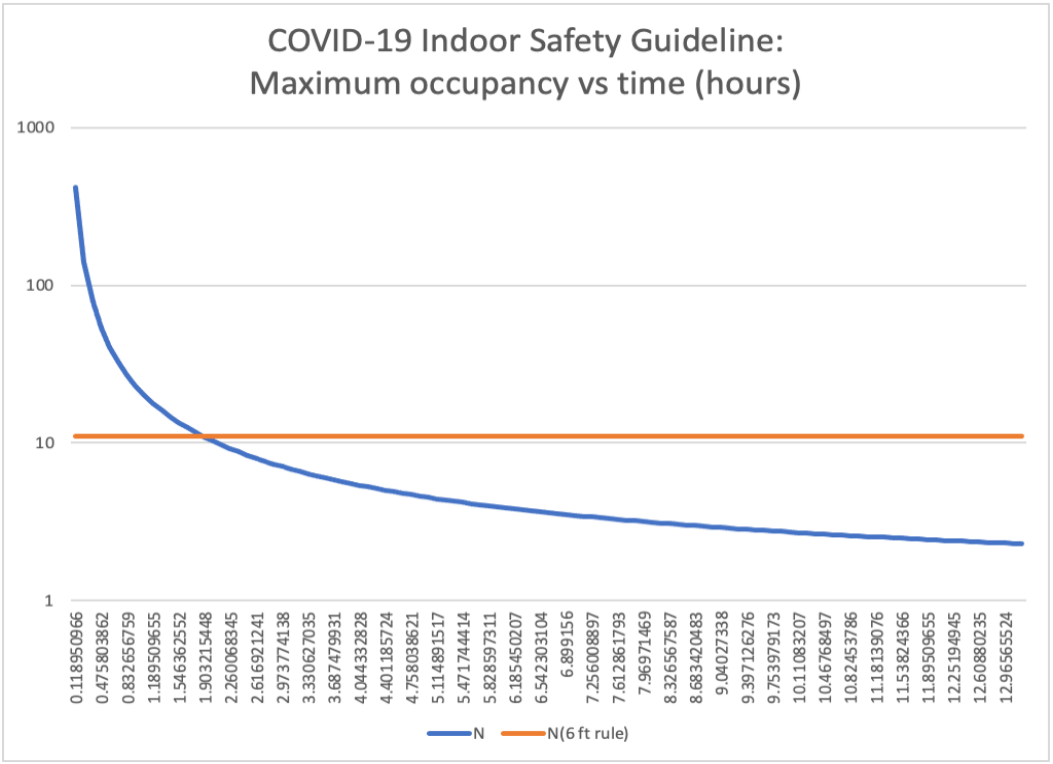
Εικόνα 4.1.10 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.25



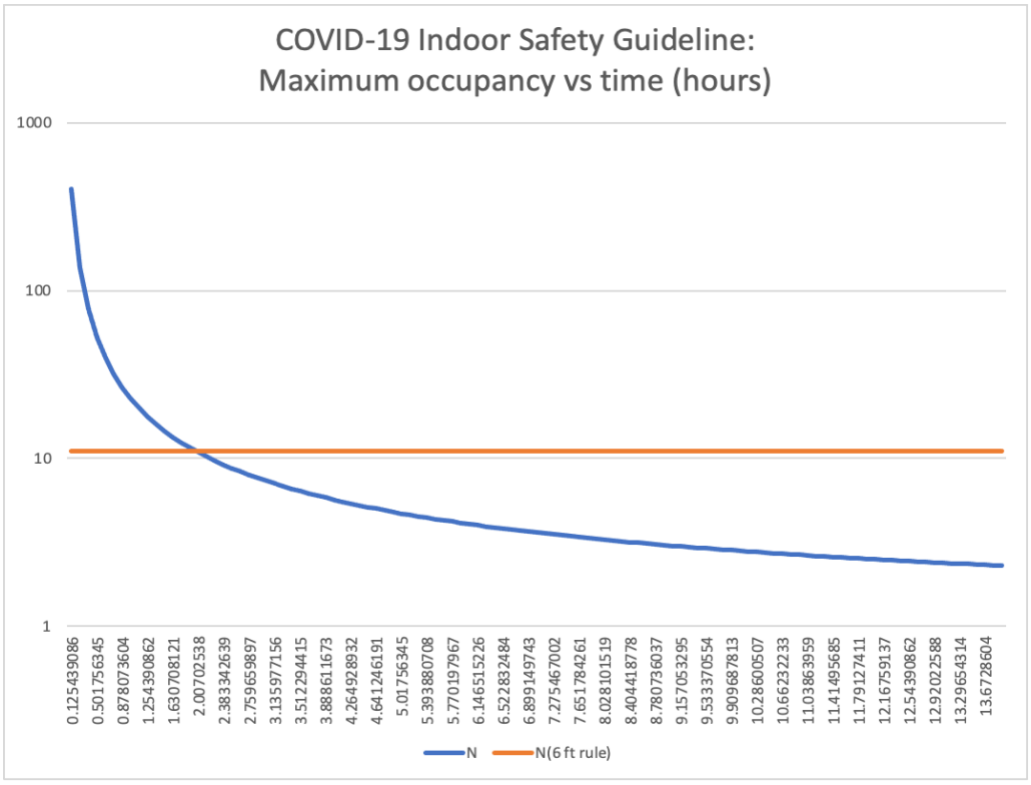
Εικόνα 4.1.11 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 1.25



Εικόνα 4.1.12 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 2.25

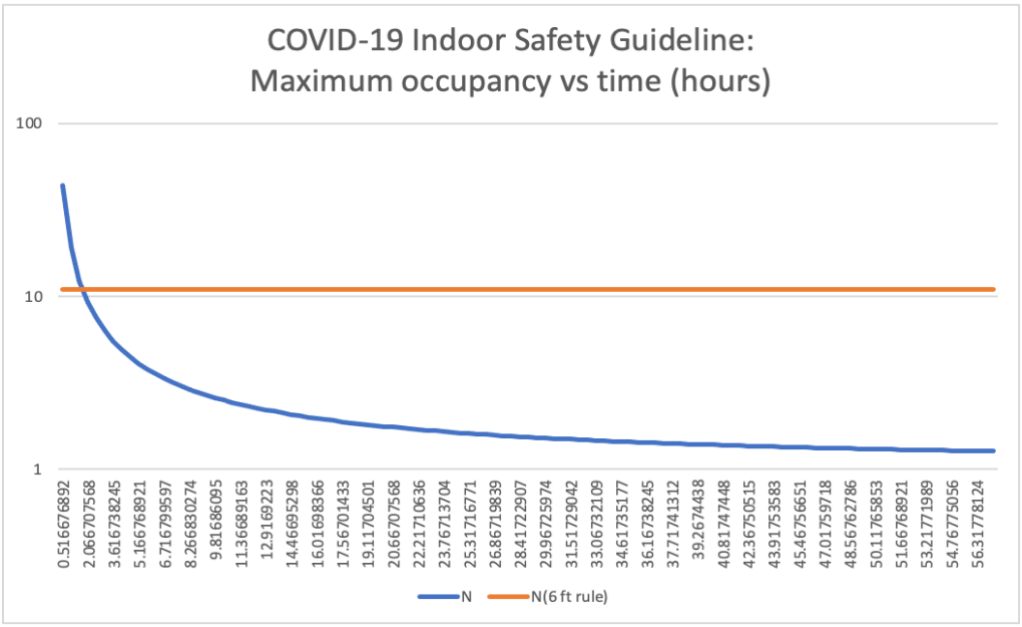


Εικόνα 4.1.13 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 3.25

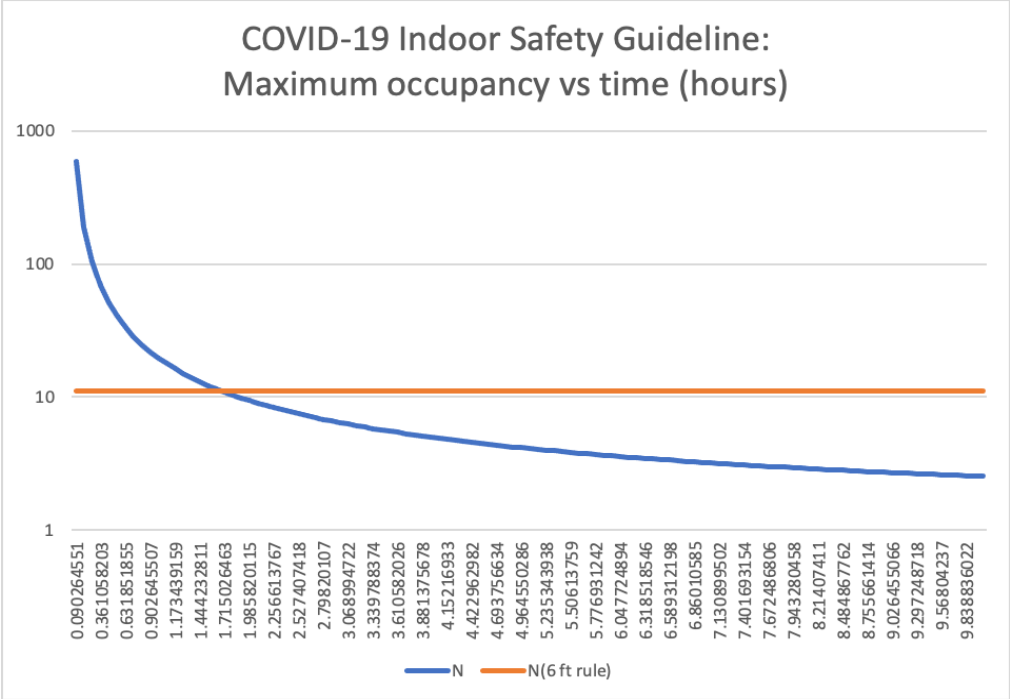


Εικόνα 4.1.14 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 4.5

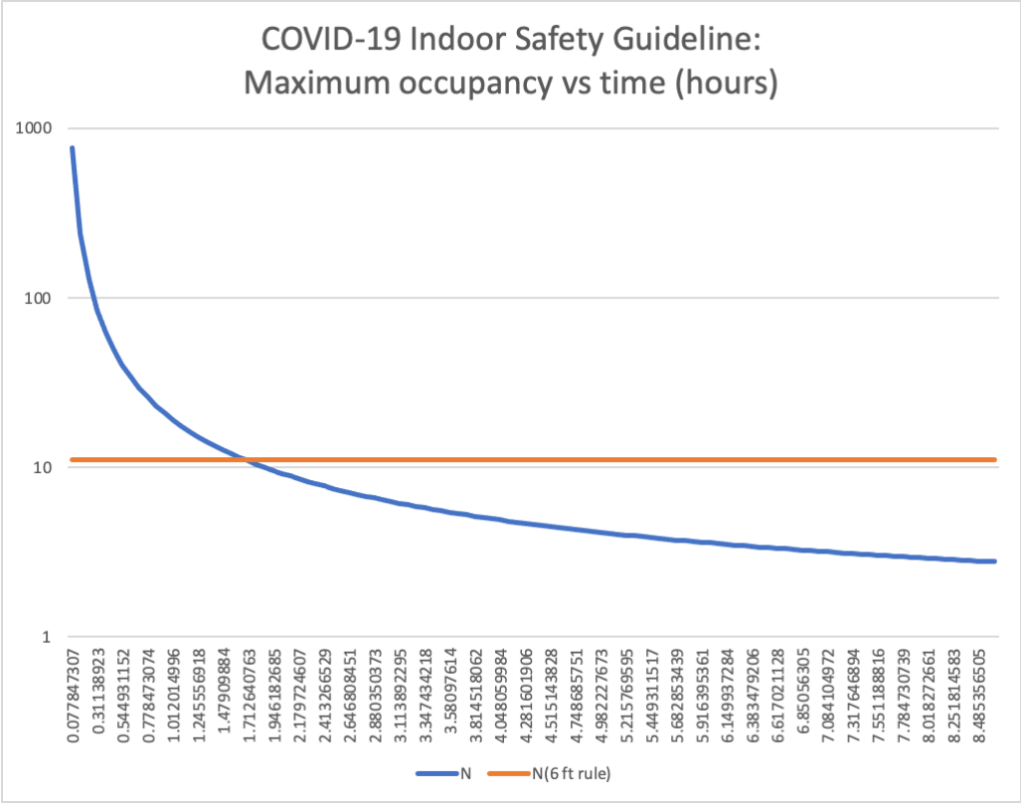
Ημέρα 3:



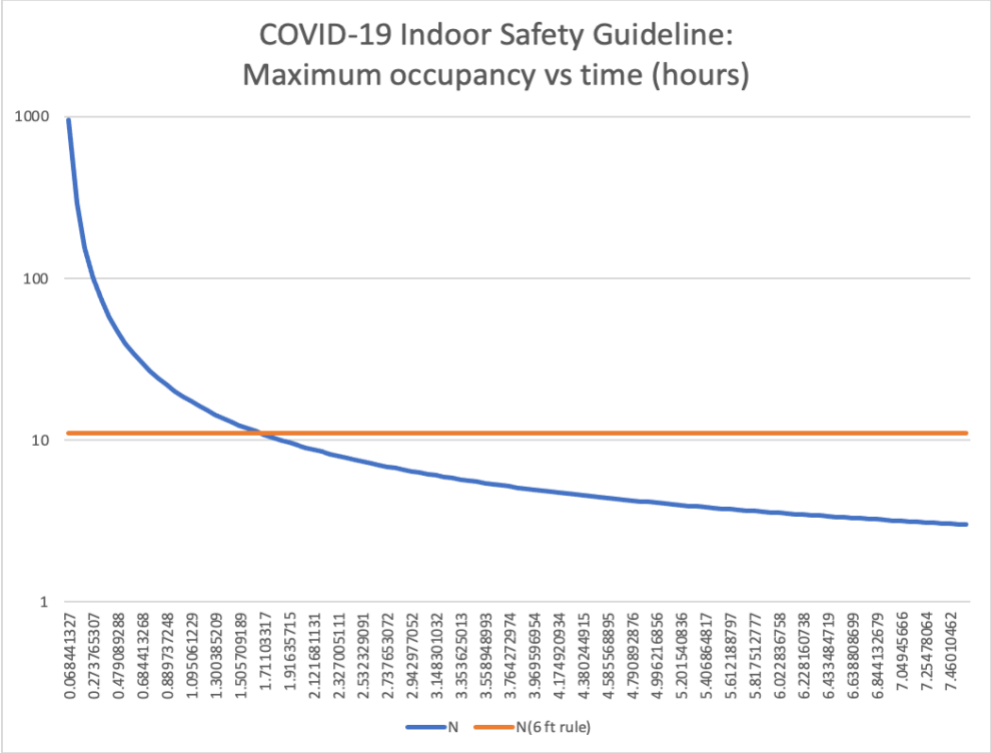
Εικόνα 4.1.15 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.25



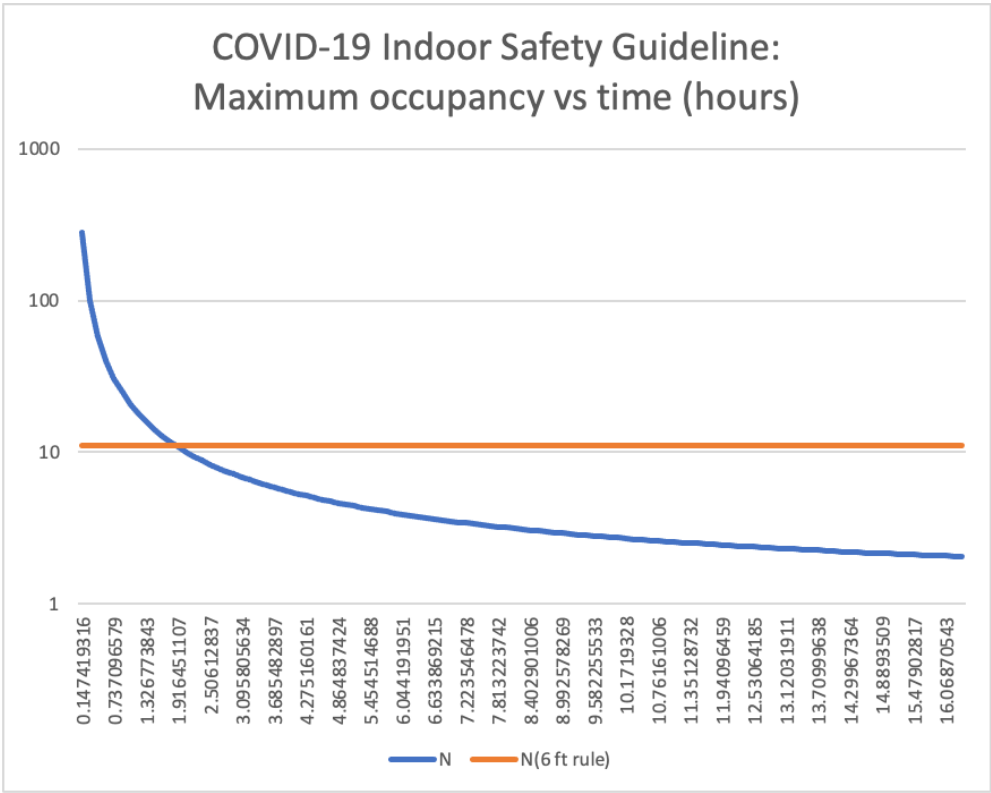
Εικόνα 4.1.16 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 1.25



Εικόνα 4.1.17 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 2.77

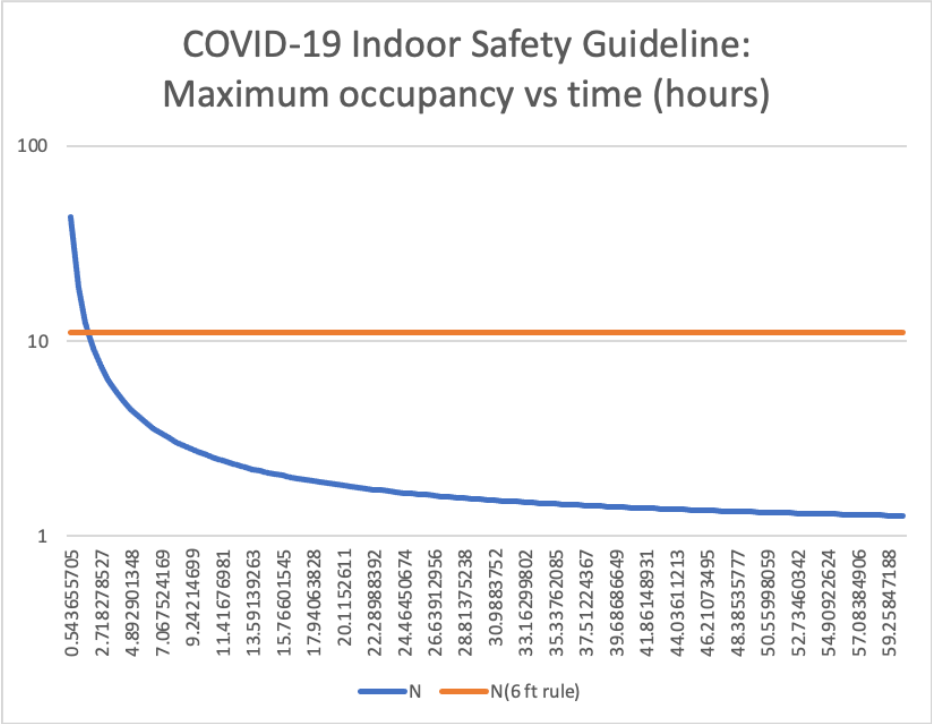


Εικόνα 4.1.18 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 4.25

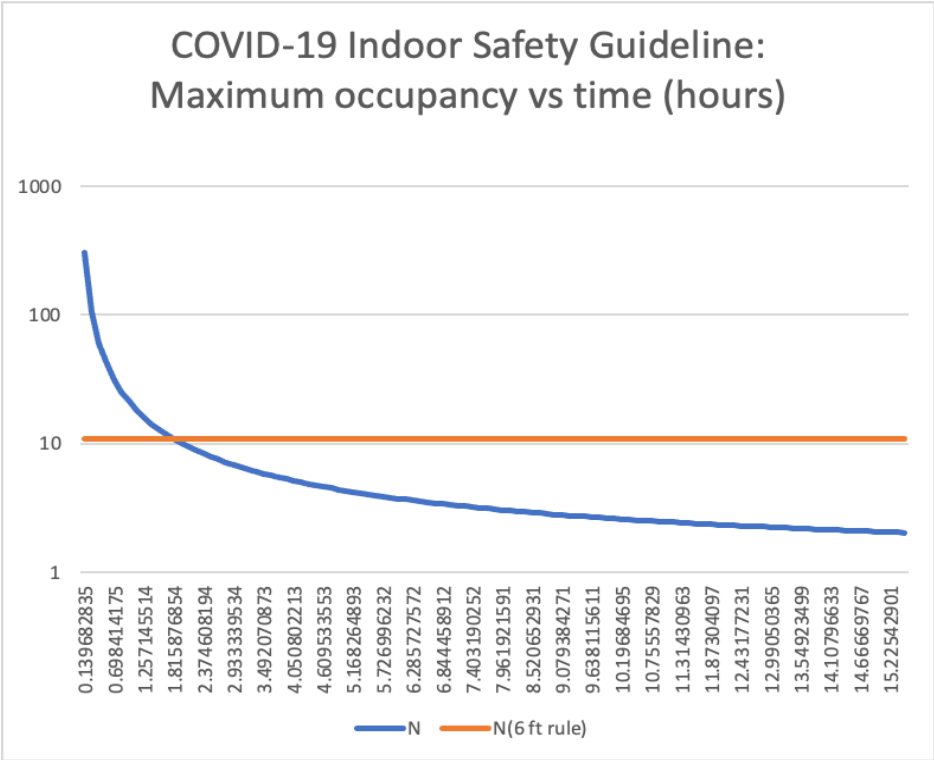


Εικόνα 4.1.19 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 5.77

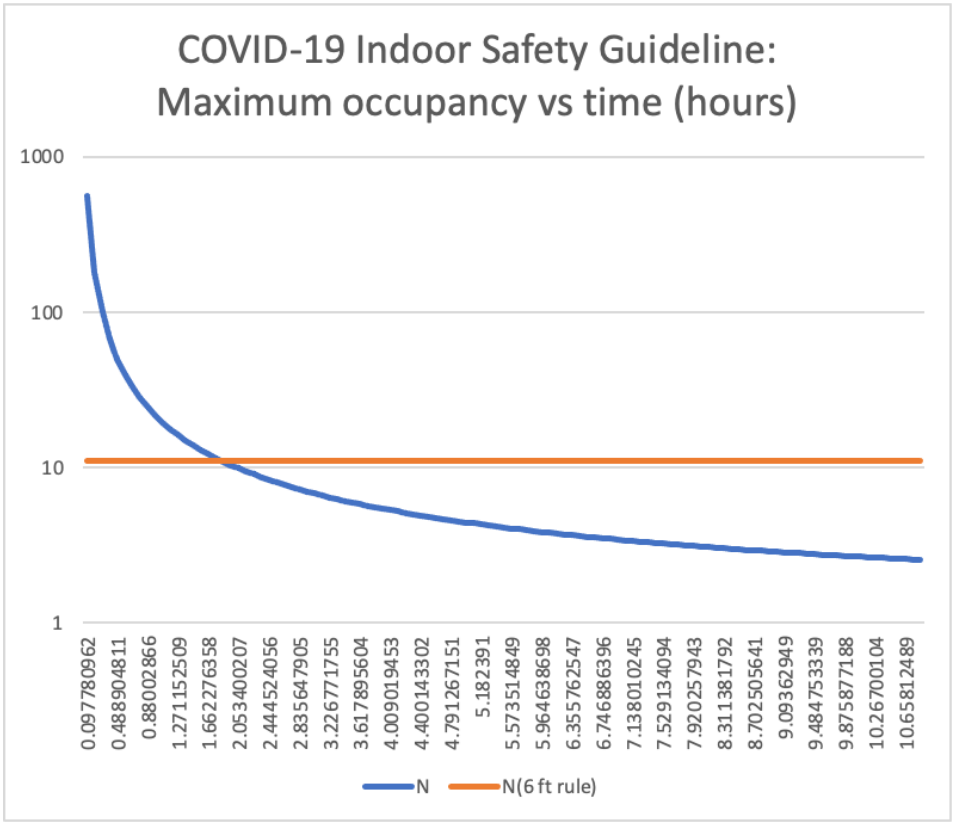
Ημέρα 4:



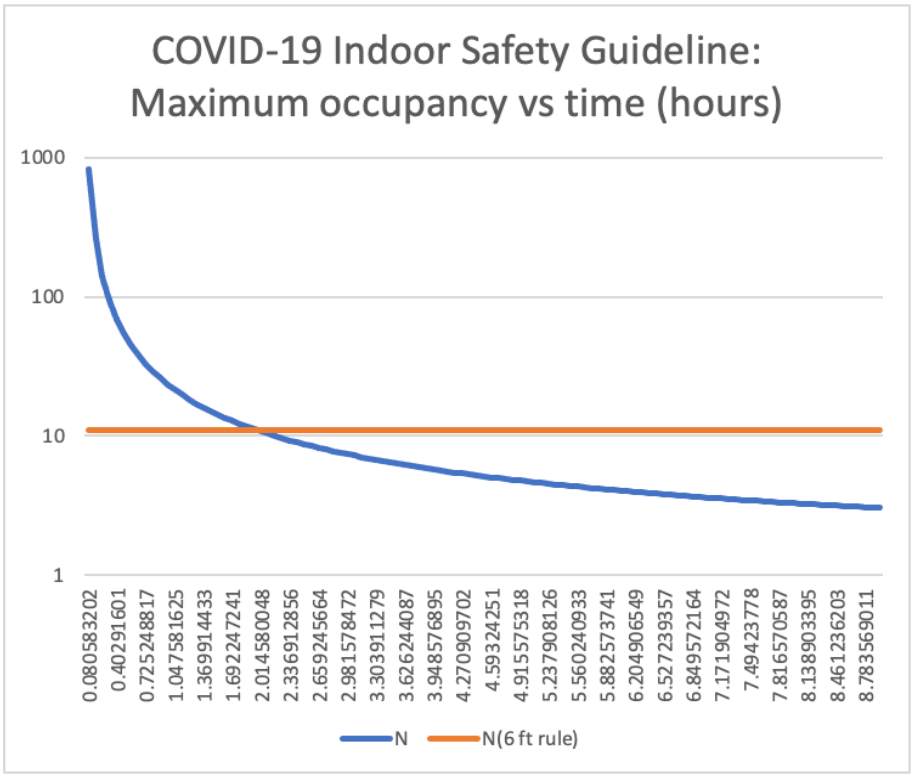
Εικόνα 4.1.20 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.25



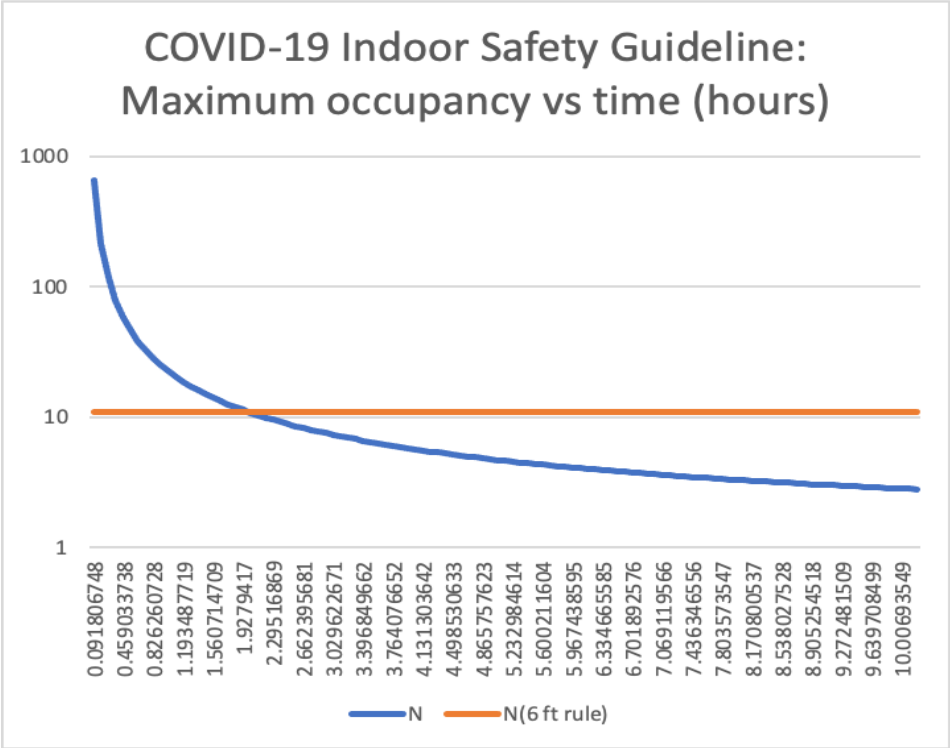
Εικόνα 4.1.21 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.43



Εικόνα 4.1.22 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.93

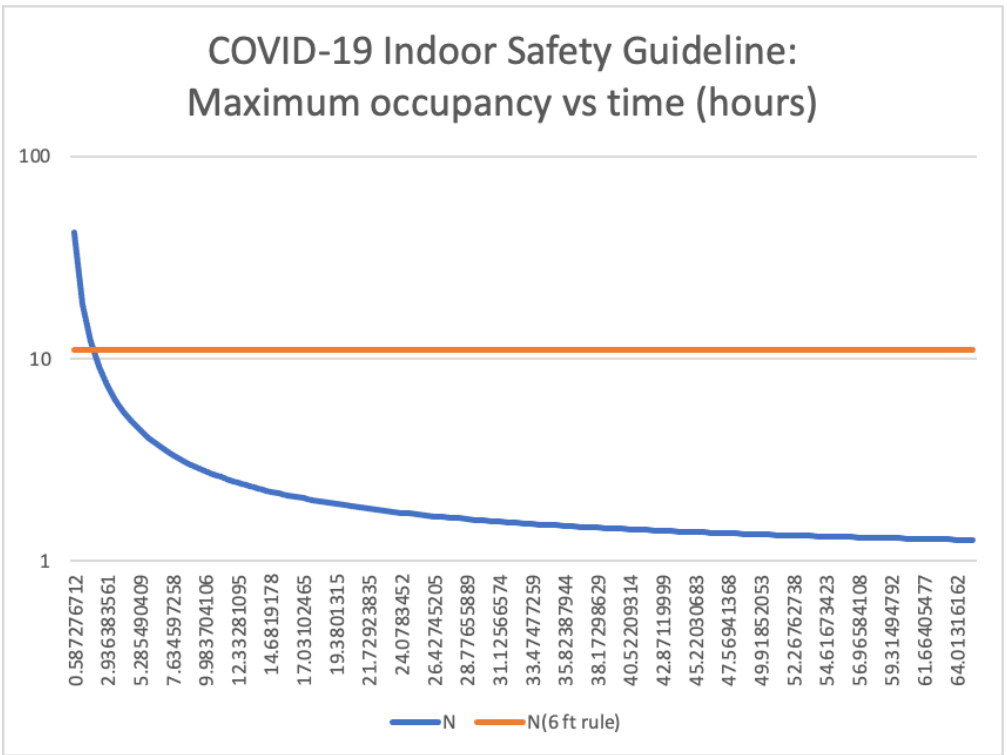


Εικόνα 4.1.23 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 4.93

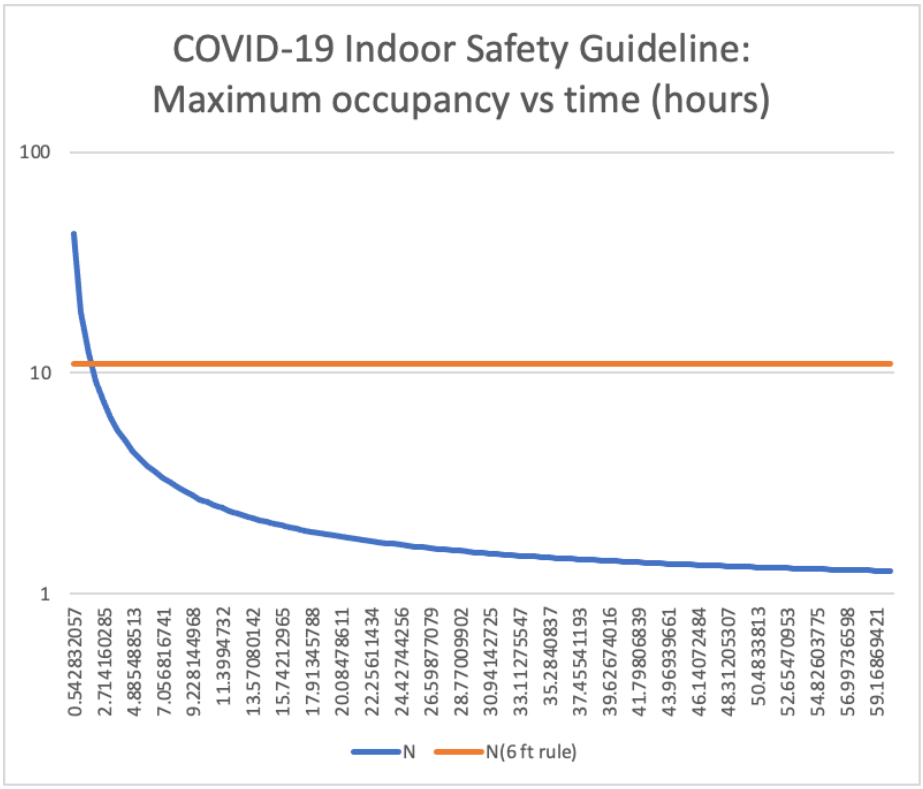


Εικόνα 4.1.24 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 5.17

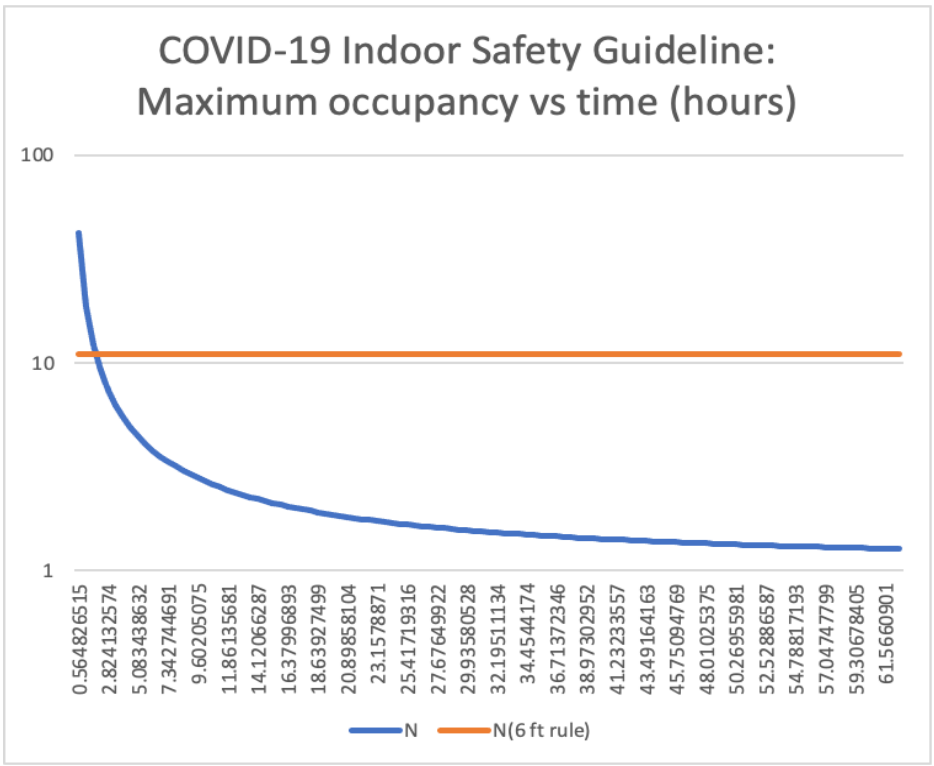
Ήμερα 5:



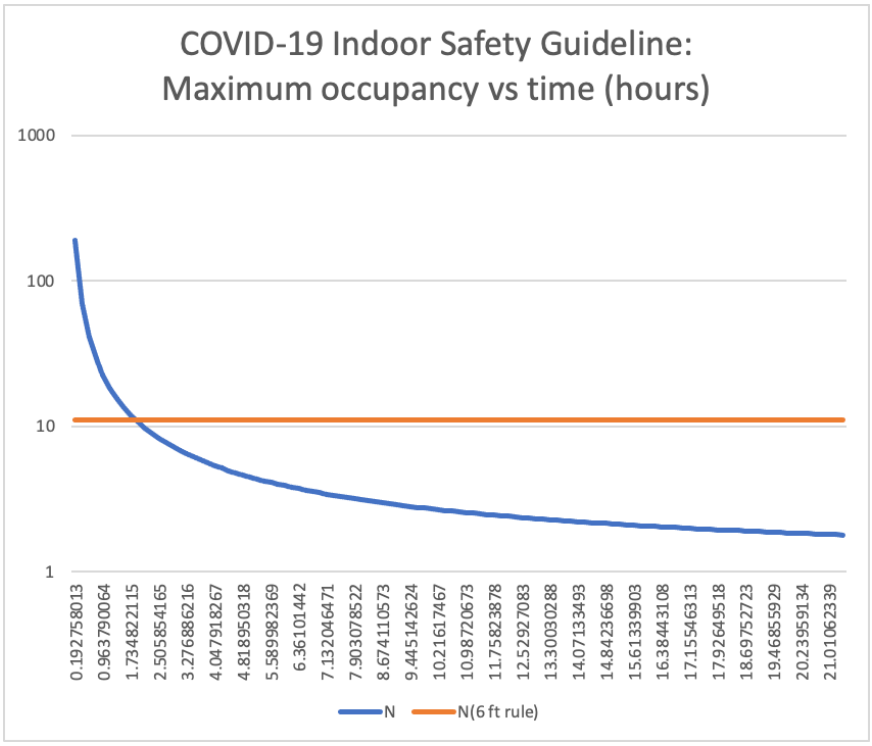
Εικόνα 4.1.25 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.25



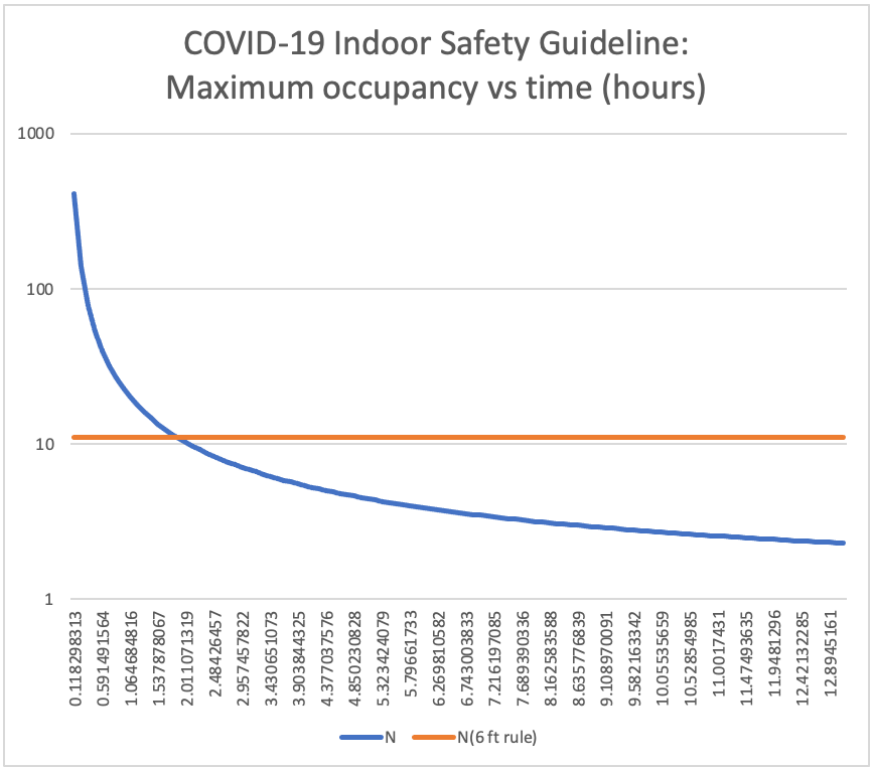
Εικόνα 4.1.26 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.50



Εικόνα 4.1.27 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.75

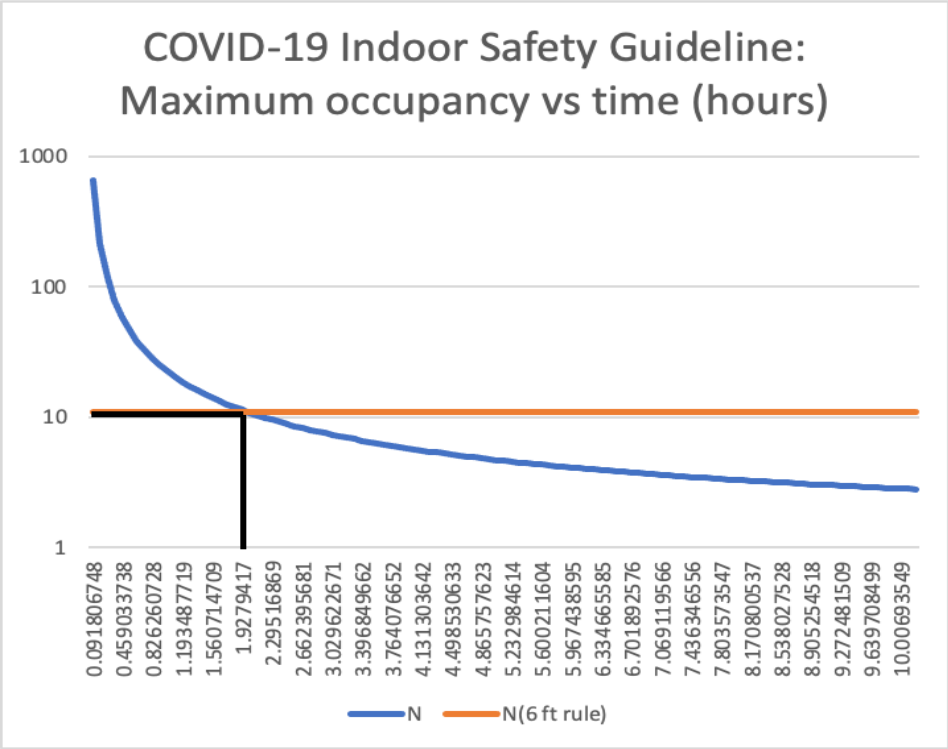


Εικόνα 4.1.28 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 0.98



Εικόνα 4.1.29 Κατευθυντήριες γραμμές για την ασφάλεια εσωτερικών χώρων για το χρονικό διάστημα 1.25

Στις παραπάνω γραφικές παραστάσεις, παρατηρούμε την μέγιστη χωρητικότητα ανά μέγιστο χρονικό διάστημα, με βάση το 6 foot rule . Παραδείγματος χάρη:



Για το χρονικό διάστημα 1.927 η μέγιστη χωρητικότητα είναι 10.

Κεφάλαιο 5

Συμπεράσματα

Στο κεφάλαιο αυτό βλέπουμε τα συμπεράσματα με βάση των αποτελεσμάτων και τι μπορεί να γίνει σαν μελλοντική δουλειά για να έχουμε καλύτερη απόδοση.

5.1 Τελικά Συμπεράσματα

Η υλοποίηση ενός χαμηλού κόστους συστήματος ανίχνευσης ρίσκου για την μετάδοση του Κορωνιού μέσα σε μια αίθουσα είναι αρκετά προκλητική αφού έχουμε να κάνουμε με συσκευές με περιορισμένους πόρους, οι οποίες μπορούν να τοποθετηθούν σε οποιονδήποτε περιβάλλον. Στη παρούσα εργασία, έχει γίνει η υλοποίηση ενός προγράμματος ανίχνευσης ρίσκου, το οποίο έχει χρησιμοποιηθεί για την συλλογή δεδομένων από ένα δίκτυο αισθητήρων. Μετά την συλλογή δεδομένων, περάσαμε τα δεδομένα από ένα αλγόριθμο ανάλυσης ρίσκου, ο οποίος μας έδωσε κάποια αποτελέσματα. Από τα αποτελέσματα που έχουμε πάρει, παρατηρούμε ότι η επικινδυνότητα μετάδοσης του ιού σε μια αίθουσα εξαρτάται κυρίως, από είδος του φιλτραρίσματος και εξαερισμού του αέρα[8][1], τον αριθμό των ατόμων που βρίσκονται μέσα στην αίθουσα, την χρήση μάσκας[1] ή όχι, καθώς και το ποσοστό αναπνοής τους ανάλογα με την άσκηση που κάνουν (ξεκουράζονται, μιλούν, κάνουν ελαφριά μορφής γυμναστική, κ.τ.λ.)[1], καθώς και με την υγρασία[10] που υπάρχει στην αίθουσα. Με βάση και το [3], από το οποίο έχουμε πάρει και τον αλγόριθμο ανάλυσης δεδομένων, παρατηρούμε ότι κυρίως χωρίς τη χρήση της μάσκας, οι ώρες που μια αίθουσα παραμένει ασφαλές, αν εισέλθει κάποιο μολυσμένο με τον ίο άτομο, μειώνονται ραγδαία.

5.2 Μελλοντική Δουλειά

Η δουλεία που έχει γίνει, μας έχει βοηθήσει στο μπορούμε να γνωρίσουμε πότε ένα δωμάτιο αρχίζει να γίνεται μη ασφαλές για τα άτομα που βρίσκονται μέσα σε αυτό.

Σαν επέκταση του προγράμματος θα μπορούσαν να χρησιμοποιηθούν διαφορετικές εξισώσεις για την ανάλυση ρίσκου για περισσότερη ακρίβεια. Επιπρόσθετα, ο αισθητήρας KY037 θα μπορούσε να μην χρησιμοποιηθεί καθόλου, λόγω της μηδαμινής του ακρίβειας [9] και αντίθετα να χρησιμοποιηθεί ένα digital microphone, το οποίο μας δίνει πιο ακριβές τιμές. Επίσης, το μοντέλο εκτίμησης ρίσκου, θα μπορούσε, με βάση τα πειραματικά αποτελέσματα, να γίνει πιο ακριβές, για να μπορούμε με περισσότερη ακρίβεια να βρούμε την απόκλιση που υπάρχει.

Τέλος, θα ήταν αρκετά χρήσιμο, αν μπορούσαμε να χρησιμοποιήσουμε και άλλους αλγόριθμους για διάφορες άλλες μεταδοτικές(με τον αέρα) ασθένειες , και ο χρήστης να μπορούσε να επιλέξει για ποια ασθένεια θα ήθελε να παρακολουθήσει την επικινδυνότητα της σε μια αίθουσα.

Βιβλιογραφία

[1] Lindsley, W. G., Derk, R. C., Coyle, J. P., Martin Jr, S. B., Mead, K. R., Blachere, F. M., ... & Noti, J. D. (2021). Efficacy of portable air cleaners and masking for reducing indoor exposure to simulated exhaled SARS-CoV-2 aerosols—United States, 2021. *Morbidity and Mortality Weekly Report*, *70*(27), 972. Available at: https://www.cdc.gov/mmwr/volumes/70/wr/mm7027e1.htm

[2] Martin Z. Bazant, John W. M. Bush, "This model of MIT can predict risk of covid-19 spreading indoors.", Journal PNAS, WORLD ECONOMIC FORUM, 2021. Available at: https://www.weforum.org/agenda/2021/04/method-to-assess-covid19-transmission-risks-indoor-settings

[3] Kasim Khan, Martin Z. Bazant, John W. M. Bush, "COVID-19 Indoor Safety Guideline", Journal PNAS, 2021. Available at: https://indoor-covid-safety.herokuapp.com/apps/advanced?units=metric

[4] Bazant, M. Z., & Bush, J. W. (2021). A guideline to limit indoor airborne transmission of COVID-19. *Proceedings of the National Academy of Sciences*, *118*(17). Available at: https://www.pnas.org/content/118/17/e2018995118?utm_source=www.diariodapalhoca.com.br&utm_medium=referral&utm_content=portal_primenews&utm_campaign=hotfixpress

[5] Mecenas, P., Bastos, R. T. D. R. M., Vallinoto, A. C. R., & Normando, D. (2020). Effects of temperature and humidity on the spread of COVID-19: A systematic review. *PLoS one*, *15*(9), e0238339. Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0238339

[6] Jingyuan Wang, Kai Feng, Weifeng Lv, Ke Tang, "New study says 'high temperature and high relative humidity significantly reduce' spread of COVID-19", AccuWeather, 2020. Available at:

https://www.accuweather.com/en/health-wellness/new-study-says-high-temperature-and-high-relative-humidity-significantly-reduce-spread-of-covid-19/703418

[7] Daqarta Help System, "Data AcQuisition And Real-Time Analysis" Available at:

https://www.daqarta.com/dw_gg0u.htm

[8] Lee, J. H., Rounds, M., McGain, F., Schofield, R., Skidmore, G., Wadlow, I., ... & Monty, J. (2021). Effectiveness of portable air filtration on reducing indoor aerosol transmission: preclinical observational trials. *Journal of Hospital Infection*. Available at: https://www.sciencedirect.com/science/article/pii/S0195670121003364

[9] KY-037 microphone sensitivity, https://forum.arduino.cc/t/ky-037-microphone-sensitivity/193967

[10] Homeland Security, "Estimated Surface Decay of SARS-CoV2-2 (virus that causes COVID-19)". Available at:

https://www.dhs.gov/science-and-technology/sars-calculator

[11] SAFEAIRSPACES COVID-19 Aerosol Relative Risk Estimator. Available at: https://safeairspaces.com/safeairspaces-estimator?fbclid=IwAR1DdOeJkpP7bLApPH-NnkJZ0P0JWGzzFcQa4boyYIhQldzJMfmD6u9Yyhw

[12] Wikipedia, "COVID-19", 2021. Available at: https://el.wikipedia.org/wiki/COVID-19

[13] Dr. Wikipedia, "Volatile organic compound". Available at:

https://en.wikipedia.org/wiki/Volatile_organic_compound

[14] Panagiotis Hadjiloizou, "ΓΕΝΙΚΗ ΕΝΗΜΕΡΩΣΗ ΓΙΑ ΤΗ ΝΟΣΟ COVID-19 (ΙΟΣ SARS-COVID-2)". Available at:

http://www.mscyprus.org.cy/files/BASIC-COVID19-v3-F-pdf.pdf

[15] "Your specialist in innovating humidity & temperature sensors". Available at:

https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

[16] Lady Ada, "PIR Motion Sensor". Available at:

https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf

[17] Dean Miller, "Adafruit CCS811 Air Quality Sensor". Available at:

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ccs811-air-quality-sensor.pdf

[18] KY-037 Microphone Sensor Module (High Sensitivity). Available at:

https://sensorkit.joy-it.net/en/sensors/ky-037

[19] KY-053 ANALOG DIGITAL CONVERTER. Available at:

https://sensorkit.joy-it.net/en/sensors/ky-053

[20] ESPERANZA EH179 MICROPHONE USB SCREAM. Available at:

https://www.e-shop.cy/esperanza-eh179-microphone-usb-scream-p-PER.580453

[21] Wikipedia, “Raspberry Pi”. Available:

https://en.wikipedia.org/wiki/Raspberry_Pi

[22] Raspberry Pi, Available at:

https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[23] pybluez.. Available at:

https://github.com/pybluez/pybluez

[24] SQLite Available at:

https://www.sqlite.org/index.html

[25] adafruit/Adafruit_Python_DHT, GitHub. Available at:

https://github.com/adafruit/Adafruit_Python_DHT

[26] board 1.0 Python Library. Available at:

https://pypi.org/project/board/

[27] adafruit/Adafruit_CCS811, GitHub. Available at:

https://github.com/adafruit/Adafruit_CCS811

[28] Adafruit CircuitPython, busio. Available at:

https://circuitpython.readthedocs.io/en/3.x/shared-bindings/busio/__init__.html

[29] adafruit/Adafruit_ADS1X15, GitHub. Available at:

https://github.com/adafruit/Adafruit_ADS1X15

[30] soundmeter 0.1.5. Available at:

https://pypi.org/project/soundmeter/

[31] Flask python. Available at:

https://www.fullstackpython.com/flask.html

Παράρτημα Α: Κώδικας Python της Main

```
import os
import subprocess
import threading
import time
import board
import adafruit_dht
import sys
import datetime
import digitalio
import busio
import adafruit_ccs811
import bluetooth
import math
import  adafruit_ads1x15.ads1115 as  ADS
from  adafruit_ads1x15.analog_in import AnalogIn
import sqlite3
import connect_db
import create_tables
import accurancy
import signal
from board import *
from board import SCL, SDA
import data_analysis

global temp
global countert
global hum
global counterh
global co2total
global counter
global cdecibel
global counterd
global summicro
global countmicro

def handler(signum, frame):
    print ("Ctrl-c was pressed.")
    con = connect_db.get_db(name)
    cursor = con.cursor()
    temperature_c = 0
    humiditys = 0
    co2 = 0
    decibels = 0
    micro = 0
    if countert != 0:
        temperature_c = temp/countert
    if counterh != 0:
        humiditys = hum/counterh
    if counter != 0:
        co2 = co2total/counter
    if counterd != 0:
        decibels = cdecibel/counterd
```

```
    if countmicro != 0:
        avgmicro = summicro / countmicro
        micro = (10.0 * math.log(avgmicro))


    sql = ''' INSERT INTO metrics (Area,Volume,BluetoothPeopleCounter,PirPeopleCounter,Temperature,Humidity,Co2,Decibel,Microphone) VALUES(?,?,?,?,?,?,?,?,?) '''
    val = (area, volume, bluecounter, peoplecount, temperature_c, humiditys, co2, decibels,micro)

    cursor.execute(sql,val)
    con.commit()
    con.close()

    accurancy.Accurancy(name)
    exit(1)

signal.signal(signal.SIGINT, handler)

def temandhum():

    global temperature_c
    global humiditys
    global temp
    global countert
    global hum
    global counterh
    temp = 0
    countert = 0
    hum = 0
    counterh = 0

    dhtDevice = adafruit_dht.DHT22(board.D18)

    while True:

        if time.strftime("%M") in ("15", "30", "45", "59"):
            if countert == 0 or counterh == 0:
                temperature_c = 0
                humiditys = 0
                break
            temperature_c = temp/countert
            humiditys = hum/counterh
            os.system("sudo pkill libgpiod_pulsei")
            break


        try:
            temp += dhtDevice.temperature
            countert += 1
            hum += dhtDevice.humidity
            counterh += 1

        except RuntimeError as error:
```

```
            dhtDevice.exit()
            raise error

        time.sleep(2.0)


def peoplecounter():

    pir = digitalio.DigitalInOut(board.CE1)
    global peoplecount
    #peoplecount = 0

    while True:

        if time.strftime("%M") in ("15", "30", "45", "59"):
            break

        if pir.value == True:
            peoplecount = peoplecount + 1
            sys.exit

            time.sleep(3)

def Co2():

    i2c = busio.I2C(board.SCL, board.SDA)  # uses board.SCL and board.SDA
    ccs811 = adafruit_ccs811.CCS811(i2c)

    global co2total
    global counter
    global co2
    co2total = 0
    counter = 0

    while not ccs811.data_ready:
        pass

    while True:
        if time.strftime("%M") in ("15", "30", "45", "59"):
            if counter == 0:
                co2 = 0
                break
            co2 = co2total/counter
            break

        co2total += ccs811.eco2
        counter += 1
        time.sleep(2)

def bluetoothcounter():
    global bluecounter
    bluecounter = 0
```

```
    while True:
        if time.strftime("%M") in ("15", "30", "45", "59"):
            break
        nearby_devices = bluetooth.discover_devices(duration=8, lookup_names=True, flush_cache=True, lookup_class=False)
        bluecounter = len(nearby_devices)

        time.sleep(0.5)

def decibel():

    #  Create the I2C bus
    i2c  =  busio.I2C(board. SCL, board. SDA)

        #  Create the  ADC object  using the  I2C  bus
    ads  =  ADS.ADS1115(i2c, address=0x48)

    global decibels
    global cdecibel
    global counterd

    cdecibel = 0
    counterd = 0

    #  Create single-ended input on channels
    chan0  =  AnalogIn(ads, ADS.P0)
    chan1 = AnalogIn(ads, ADS.P1)
    chan2 =  AnalogIn(ads, ADS.P2)
    chan3 = AnalogIn(ads, ADS.P3)

    while True:

        if time.strftime("%M") in ("15", "30", "45", "59"):
            if counterd == 0:
                decibels = 0
                break
            decibels = cdecibel/counterd
            break

        analog  = chan0.voltage
        cdecibel += (20.0 * math.log(3/analog))
        counterd += 1

        #button_pressed  =  False
        time.sleep(0.05)

def digMicro():

    global micro
    global summicro
    global countmicro

    summicro = 0
```

```
    countmicro = 0

    while True:

        if time.strftime("%M") in ("14", "29", "44", "58"):
#           metr = os.system("cat test | grep avg | tr -s ' ' | cut -d ' ' -f3")
            if countmicro>0:
                micro = round(summicro / countmicro)
                #print (avgmicro)
                #micro = (10.0 * math.log((avgmicro*avgmicro)))
                #print (micro)
            else:
                micro = 0

            break

        os.system("soundmeter --collect --seconds 10 > test")
        os.system("cat test | grep avg | tr -s ' ' | cut -d ' ' -f3 > res.txt")
        with open('res.txt') as f:
            metr = f.readlines()
        f.close()
        #print (metr[0])
        summicro = summicro + int(metr[0])
        countmicro = countmicro + 1


def main_task():

    global humiditys
    global temperature_c
    global peoplecount
    global co2
    global bluecounter
    global decibels
    global micro

    t1 = threading.Thread(target=temandhum)
    t2 = threading.Thread(target=peoplecounter)
    t3 = threading.Thread(target=Co2)
    t4 = threading.Thread(target=bluetoothcounter)
    t5 = threading.Thread(target=decibel)
    t6 = threading.Thread(target=digMicro)

    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t5.start()
    t6.start()
```

```
    t1.join()
    t2.join()
    t3.join()
    t4.join()
    t5.join()
    t6.join()

if __name__ == "__main__":

    print ("Give the name of the Project")
    namei = input()
    name = namei + ".db"
    #name = "Day5.db"

    create_tables.create_table(name)

    print ("Give the length")
    length = float(input())
    #length = 5

    print("Give the width")
    width = float(input())
    #width = 8

    print ("Give the height")
    height = float(input())
    #height = 3.5

    print ("Give the human condition: \n 1.Resting \n 2.Standing \n 3.Singing 4.Ligth_Exercise \n 5.Moderate_Exercise \n 6.Heavy_Exercise \n Give number 1-6")
    breathing_rate = int(input())

    while breathing_rate<1 and breathing_rate>6:
        print ("Wrong Input! Give number from 1 to 6!")
        breathing_rate = int(input())

    print ("Give the age of the people inside the room.")
    age = int(input())

    while age<0 and age> 100:
        print ("Wrong Input! Give number from 0 to 100")
        age = int(input())

    print ("Mask wearing. \n 1.No Mask \n 2.Surgical Mask \n 3.Hybrid-Fabric Mask")
    mask = int(input())

    while mask<1 and mask>3:
        print ("Wrong Input! Give number from 1 to 3!")
        mask = int(input())

    area = length * width
    volume = area * height
```

```
    #with open(testname, mode='w') as csv_file:
    #   fieldnames = ['Timestamp', 'Area', 'Volume', 'Bluetooth People Counter', 'Pir People Counter', 'Temperature', 'Humidity', 'CO2', 'Decimbel']
    #   writer = csv.DictWriter(csv_file, fieldnames = fieldnames)

    #   writer.writeheader()

    #counttime = 0
    #countertime = 0
    peoplecount = 0

    while True:

        if time.strftime("%M") in ("15", "30", "45", "59"):
            time.sleep(61)

        main_task()
        #with open(testname , mode='a') as csv_file:
        #   writer = csv.DictWriter(csv_file, fieldnames = fieldnames)
        #   writer.writerow({'Timestamp': time.strftime("%H:%M:%S"), 'Area': area, 'Volume' : volume, 'Bluetooth People Counter': bluecounter, 'Pir People Counter': p

        con = connect_db.get_db(name)
        cursor = con.cursor()

        sql = ''' INSERT INTO metrics (Area,Volume,BluetoothPeopleCounter,PirPeopleCounter,Temperature,Humidity,Co2,Decibel,Microphone) VALUES(?,?,?,?,?,?,?,?,?) '''
        val = (area, volume, bluecounter, peoplecount, temperature_c, humiditys, co2, decibels, micro)

        cursor.execute(sql,val)
        con.commit()
        con.close()

        peoplecount = round((peoplecount+bluecounter)/2)
        #if counttime == 2:
            #counttime = 0
            #peoplecount = round((peoplecount+bluecounter)/2)
            #countertime = countertime + 1
            #if countertime == 2:
                #accurancy.Accurancy(name)
                #break
            #time.sleep(1800)
        #counttime = counttime + 1
        if time.strftime("%H:%M") in ("14:15"):
            accurancy.Accurancy(name)
            data_analysis(width, length, heigth, breathing_rate, age, mask, name)
            break
        time.sleep(61)
```

Παράρτημα Β: Κώδικας Python Ανάλυσης Δεδομένων

```python
import connect_db
import math
import re
from datetime import datetime
from datetime import timedelta
import PlotData
import create_risk_table


def data_analysis(width, length, heigth, breathing_rate, age, mask, database):

    age = age / 100

    database_name = "Risk_" + database
    create_risk_table.create_table(database_name)

    db = connect_db.get_db(database)
    cursor = db.cursor()
    query = "SELECT Timestamp FROM metrics LIMIT 1"
    cursor.execute(query)
    firsthours = cursor.fetchall()
    firsthours = firsthours[0][0]

    n = 15

    date_format_str = '%Y-%m-%d %H:%M:%S'

    giv_time = datetime.strptime(firsthours, date_format_str)

    a = giv_time - timedelta(minutes=n)

    temp_hour = a + timedelta(hours=2)
    time = str(temp_hour)

    q1 = cursor.execute("SELECT Timestamp FROM metrics")
    allhours = q1.fetchall()

    q2 = cursor.execute("SELECT PirPeopleCounter FROM metrics")
    allpeople = q2.fetchall()

    q3 = cursor.execute("SELECT Humidity FROM metrics")
    allhumidity = q3.fetchall()

    q4 = cursor.execute("SELECT Co2 FROM metrics")
    allco2 = q4.fetchall()

    q5 = cursor.execute("SELECT Decibel FROM metrics")
    alldec = q5.fetchall()

    i = 0

    #a = datetime.strptime(firsthour, date_format_str)
```

```python
    for record in allhours:
        hour = allhours[i][0]
        Npers = allpeople[i][0]
        humidity = allhumidity[i][0]
        Co2 = allco2[i][0]
        decibel = alldec[i][0]

        if Npers<2:
            Npers=2

        decibel = (10.0 * math.log((decibel*decibel)))

        b = datetime.strptime(hour, date_format_str)
        diff_hours = b-a
        hours = diff_hours.total_seconds() / 3600

        temp_hour2 = b + timedelta(hours=2)
        times = time + " - " + str(temp_hour2)

        if breathing_rate == 1:
            Qb = 0.49
        elif breathing_rate == 2:
            Qb = 0.54
        elif breathing_rate == 3:
            Qb = 1
        elif breathing_rate == 4:
            Qb = 1.38
        elif breathing_rate == 5:
            Qb = 2.35
        elif breathing_rate == 6:
            Qb = 3.30

        if mask == 1:
            maskpm = 1
        elif mask == 2:
            maskpm = 0.055
        elif mask == 3:
            maskpm = 0.25


        area = width * length
        volume = area * heigth

        la = 3
        Q = volume * la
        lr = 1
        Qr = volume / lr
        QplusQr = Q + Qr
        Zp = Q / QplusQr
        pf = 0.5

        lf = pf * Qr / volume
```

```
        r = 2
        humadjrad = r*pow((0.4/(1-humidity/100)),(1/3))

        Cq = decibel
        RH = 0.6
        humadjadeact = RH * humidity /50

        VSr = (2/9) * 1100 / (1.86 * pow(10,(-5))) * 9.8 * (1/pow(1000,3)) * pow(humadjrad,2)
        tempVSr = VSr * 60 * 60 / 1000
        lc = la + humadjadeact + lf + tempVSr / heigth
        fd = Qb / (lc * volume)
        inflecofroom = Cq * fd

        ba = pow((Qb * maskpm),2) * Cq * age / (volume * lc)
        e = 0.1

        t = 10
        persons = 1

        WHO = math.floor(area/1)
        minimum_outdoor_airflow = 15
        maximum_occupancy_outdoor_air = Q/minimum_outdoor_airflow

        #Npers = pircounter
        C68 = e / ((Npers-1) * ba)
        tmax = C68 * (1 + math.sqrt(1+4/(lc*C68)))/2

        average_co2 = 700
        maxt = e * 38000 * lc / ((average_co2-Co2) * Qb * Cq * age * maskpm * la)
        safe_co2_concentration = Co2 + e * 38000 * lc / (t * Qb * Cq * age * maskpm * maskpm * la)

        if hours > tmax:
            risk = 1
        else:
            risk = 0

        con = connect_db.get_db(database_name)
        cursor = con.cursor()

        sql = ''' INSERT INTO risk (Times, Hours, MaxHours, Risk) VALUES(?,?,?,?)'''
        val = (times, hours, tmax, risk)

        cursor.execute(sql, val)
        con.commit()
        con.close()

        PlotData.Plot(e, ba, lc, width, length, tmax, database, i)
        times = ""
        i = i+1
```

```
import math
import csv

def Plot(epsilon, beta_a, lamba_c, width, length, dtau, name, i):

    N6foot = (width*3.2808399) * (length * 3.2808399)
    N6foot = math.floor(N6foot/36)

    dtau = dtau / 30

    plotdata = []

    t0 = 0
    t1 = dtau
    N = 1+epsilon*(1+1/(lamba_c*t1))/(beta_a*t1)


    name = name[:-3]
    name = name + str(i) +".csv"

    with open(name, mode='w') as csv_file:
        fieldnames = ['t(hours)', 'N', 'epsilon', 'beta_a', 'lambda_c', 'N(6 ft rule)']
        writer = csv.DictWriter(csv_file, fieldnames = fieldnames)

        writer.writeheader()


    with open(name, mode='a') as csv_file:
        writer = csv.DictWriter(csv_file, fieldnames = fieldnames)
        writer.writerow({'t(hours)':t1, 'N':N, 'epsilon':epsilon, 'beta_a':beta_a, 'lambda_c':lamba_c, 'N(6 ft rule)':N6foot})

    for x in range(110):

        t = t1+(t1-t0)
        t0 = t1
        t1 = t
        N = 1+epsilon*(1+1/(lamba_c*t1))/(beta_a*t1)

        with open(name, mode='a') as csv_file:
            writer = csv.DictWriter(csv_file, fieldnames = fieldnames)
            writer.writerow({'t(hours)':t1, 'N':N, 'epsilon':epsilon, 'beta_a':beta_a, 'lambda_c':lamba_c, 'N(6 ft rule)':N6foot})
```

Παράρτημα Γ: Κώδικας Python API

```python
from flask import Flask, request, jsonify

import get_metrics
import sys

global name
app = Flask(__name__)

@app.route('/admin:k3p8z5/metrics')
def all_metrics():
    metrics = get_metrics.get_metrics(name)
    return jsonify(metrics)

@app.route('/admin:k3p8z5/risk')
def risk():
    risks = get_metrics.get_risk(name2)
    return jsonify(risks)


def switch_demo(param):

    if param == "area":
        return get_metrics.get_area(name)
    elif param == "volume":
        return get_metrics.get_volume(name)
    elif param == "blueP":
        return get_metrics.get_blueP(name)
    elif param ==  "pirP":
        return get_metrics.get_PirP(name)
    elif param == "temperature":
        return get_metrics.get_Temperature(name)
    elif param == "humidity":
        return get_metrics.get_Humidity(name)
    elif param == "co2":
        return get_metrics.get_Co2(name)
    elif param == "decibel":
        return get_metrics.get_Decibel(name)
    else:
        return []

def get_rec(records):

    print (records)

    rec = 0
    if records==None:
        return []
    else:
        rec = int(records)
```

```python
    if rec > 0:
        return get_metrics.get_records(rec, name)
    else:
        return []

@app.route('/admin:k3p8z5/params')
def parameters():

    param1 = request.args.get('param1')
    param2 = request.args.get('param2')
    param3 = request.args.get('param3')
    param4 = request.args.get('param4')
    param5 = request.args.get('param5')
    param6 = request.args.get('param6')
    param7 = request.args.get('param7')
    param8 = request.args.get('param8')
    records = request.args.get('records')

    ans = switch_demo(param1)
    ans2 = switch_demo(param2)
    ans3 = switch_demo(param3)
    ans4 = switch_demo(param4)
    ans5 = switch_demo(param5)
    ans6 = switch_demo(param6)
    ans7 = switch_demo(param7)
    ans8 = switch_demo(param8)
    arec = get_rec(records)

    fans = ans + ans2 + ans3 + ans4 + ans5 + ans6 + ans7 + ans8 + arec
    return jsonify(fans)


if __name__ == '__main__':

    name = sys.argv[1]
    name2 = sys.argv[2]
    app.run(debug=True,host='192.168.10.44', port = 5000)
```

```
import connect_db

def get_metrics(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT * FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_risk(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT * FROM risk"
    cursor.execute(query)
    return cursor.fetchall()

def get_area(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Area FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_volume(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Volume FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_blueP(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT BluetoothPeopleCounter FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_PirP(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT PirPeopleCounter FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_Temperature(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Temperature FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()
```

```
def get_Humidity(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Humidity FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_Co2(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Co2 FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_Decibel(name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    query = "SELECT Decibel FROM metrics"
    cursor.execute(query)
    return cursor.fetchall()

def get_records(rec, name):
    db = connect_db.get_db(name)
    cursor = db.cursor()
    recc = str(rec)
    exeq = "SELECT * FROM metrics ORDER BY Timestamp DESC LIMIT " + recc
    query = exeq
    cursor.execute(query)
    return cursor.fetchall()
```