ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΚΑΙ ΕΛΕΓΧΟΣ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΜΕ ΠΕΠΙΕΣΜΕΝΟ ΑΕΡΑ

MONITORING AND CONTROL OF A COMPRESSED AIR ENERGY STORAGE SYSTEM (CAESS)

Εισηγητής: Νικίας Τζοβάρας

Επιβλέπων καθηγητής: Ιωάννης Γκιάλας

ΧΙΟΣ

Οκτώβριος 2022

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου εργασίας, τον Καθ. κ. Ι. Γκιάλα για την εμπιστοσύνη του ως προς το πρόσωπο μου με την ανάληψη του θέματος της παρούσας εργασίας, την καθοδήγηση και τις εύστοχες υποδείξεις του κατά την πορεία αυτής.

Επίσης, θα ήθελα να ευχαριστήσω και τα άλλα δύο μέλη της τριμελούς επιτροπής μου, Αναπλ. Καθ. κ. Ν. Αλεξόπουλο και Καθ. κ. Κ. Παπαγεωργίου, για την αποδοχή της πρότασης να την απαρτίσουν.

Συμπληρωματικά, θα ήθελα να ευχαριστήσω και την οικογένειά μου για την ηθική στήριξη που μου προσέφεραν κατά την υλοποίηση και συγγραφή της εργασίας.

ΔΗΛΩΣΗ ΠΕΡΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Έχω διαβάσει και κατανοήσει τους όρους για τη λογοκλοπή και τον τρόπο σωστής αναφοράς των πηγών που περιέχονται στον Οδηγό συγγραφής διπλωματικών εργασιών του ΤΜΟΔ. Δηλώνω ότι, από όσα γνωρίζω, το περιεχόμενο της παρούσας διπλωματικής εργασίας είναι προϊόν δικής μου δουλειάς και υπάρχουν αναφορές σε όλες τις πηγές που χρησιμοποίησα.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, είναι γεγονός πως η ανάγκη για χρήση και κατανάλωση ενέργειας, από όλους τους τελικούς χρήστες, είτε αυτοί είναι επιχειρήσεις και βιομηχανίες, είτε νοικοκυριά, έχει αυξηθεί σημαντικά δημιουργώντας έτσι μια συνεχώς αυξανόμενη ζήτηση στον τομέα αυτό. Παράλληλα, όμως, υπάρχει και η ανάγκη για πιο φιλικές προς το περιβάλλον πρακτικές για την παραγωγή ενέργειας, με τις αντίστοιχες πολιτικές ανά τον κόσμο να δρουν προς αυτή την κατεύθυνση. Έτσι είναι θεμιτό, να περιοριστούν οι παραδοσιακοί τρόποι για την παραγωγή ενέργειας, όπως η καύση ορυκτών καυσίμων, και να αντικατασταθούν από λιγότερο επιβλαβείς για το περιβάλλον αλλά και κατ' επέκταση για τον άνθρωπο.

Η αντικατάσταση των ορυκτών καυσίμων από τις ανανεώσιμες πηγές ενέργειας προϋποθέτει την δυνατότητα αποθήκευσης τους, καθώς αυτές είναι στοχαστικής φύσης και στην πρωτογενή τους μορφή παρουσιάζουν μεγάλη μεταβλητότητα. Οι συνηθισμένοι τρόποι αποθήκευσης είναι η χρήση μπαταριών, αντλησιοταμίευσης, και πρόσφατα έχει αρχίσει να γίνεται συζήτηση για το υδρογόνο.

Στην παρούσα διπλωματική εργασία θα δοθεί έμφαση σε μία άλλη τεχνολογία αποθήκευσης και συγκεκριμένα στο Σύστημα Αποθήκευσης Ενέργειας μέσω Πεπιεσμένου Ατμοσφαιρικού Αέρα(Compressed Air Energy Storage). Πρόκειται για ένα σύστημα, κατά τη λειτουργία του οποίου ατμοσφαιρικός αέρας αρχικά προωθείται σε συμπιεστές(οι οποίοι τροφοδοτούνται αποκλειστικά με ενέργεια από Ανανεώσιμες Πηγές Ενέργειας) και στη συνέχεια συμπιέζεται και αποθηκεύεται σε υπό πίεση αεροφυλάκια, ώστε μελλοντικά με την αποσυμπίεση του όταν υπάρχει μεγάλη ζήτηση, να παραχθεί ηλεκτρισμός αλλά και θερμότητα με σταθερή τάση. Ωστόσο, το σύστημα αυτό χρήζει συνεχούς παρακολούθησης κατά τη λειτουργία του, προκειμένου να μην ξεπεραστούν ορισμένες τιμές σε μεγέθη όπως η πίεση και η θερμοκρασία κατά τα στάδια της συμπίεσης και της αποσυμπίεσης του αέρα.

Σκοπός της συγκεκριμένης διπλωματικής είναι να μελετηθεί και να αποσαφηνιστεί ο τρόπος με τον οποίο θα πραγματοποιηθεί η παρακολούθησή του συστήματος αλλά και ο έλεγχος αυτού, μέσω της παρέμβασης ειδικού λογισμικού, στις περιπτώσεις όπου κρίνεται αναγκαίο.(π.χ. υπέρβαση των επιθυμητών τιμών της πίεσης, της θερμοκρασίας και της υγρασίας). Θα εγκατασταθεί ένας αριθμός αισθητήρων. Τα δεδομένα από αυτούς θα συγκεντρώνονται, θα αναλύονται και θα παρουσιάζονται στον ελεγκτή μέσω μίας πλατφόρμας λογισμικού ανοικτού κώδικα. Θα αναπτυχθεί λογισμικό που θα συνδυάζει τις τιμές από τους αισθητήρες για να αποφασίζει κάποια ενέργεια, πχ. Μεταβολή της ροής, άνοιγμα βαλβίδων, κλπ.

SUMMARY

In recent years, it is a fact that the need for the use and consumption of energy, by all end users, whether they are businesses and industries, or households, has increased significantly, thus creating an ever-increasing demand in this sector. At the same time, however, there is also the need for more environmentally friendly practices for energy production, with the corresponding policies around the world acting in this direction. Hence, it is legitimate to limit the traditional ways of producing energy, such as the burning of fossil fuels, and to replace them with less harmful ones for the environment and, by extension for humans.

The replacement of fossil fuels by renewable energy sources presupposes the ability to store them, as they are stochastic and in their primary form show great variability. Common ways of storage are using batteries, hydro pumped storage, and recently hydrogen has started to be discussed.

In this thesis, emphasis will be placed on another storage technology, namely the Compressed Air Energy Storage System. It is a system, during the operation of which atmospheric air is initially pushed to compressors (which are powered exclusively by energy from Renewable Energy Sources), and then it is compressed and stored at high pressure specially made containers (tanks), so that in the future by expanding it when there is demand, to produce electricity as well as heat. However, this system needs continuous monitoring during its operation, in order not to exceed certain values in quantities as pressure and temperature during the stages of air compression and expansion.

The purpose of the present thesis is to study and clarify how the system will be monitored and controlled, through the intervention of special software, in cases where it is deemed necessary. (e.g., exceeding the desired pressure, temperature, and humidity values). A specific number of sensors will be installed. The data will be collected, analyzed, and presented to the controller through an open-source software platform. The software that will be developed, will combine the values from the sensors to decide some action, e.g., changing the flow, opening valves, etc.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Περιεχόμενα

1. ΕΙΣΑΓΩΓΗ

1.1. Η ΕΝΝΟΙΑ ΤΗΣ ΕΝΕΡΓΕΙΑΣ

Πολύ συχνά στην καθημερινότητα μας συναντάμε την έννοια της ενέργειας, καθώς οι ανθρώπινες δραστηριότητες, είτε πρόκειται σε επίπεδο νοικοκυριού, είτε σε επίπεδο βιομηχανίας, βασίζονται και εξαρτώνται από αυτήν. Ωστόσο, αξίζει να σημειωθεί ότι οι άνθρωποι συνήθως βλέπουν το αποτέλεσμα της μετατροπής μιας μορφής ενέργειας σε μια άλλη, αφού αυτή δε δημιουργείται από το μηδέν, ούτε χάνεται, αλλά αποθηκεύεται στο φυσικό περιβάλλον και μεταβάλλεται σε άλλα είδη και μορφές. Για παράδειγμα, ο λιγνίτης που υπάρχει στη φύση και λογίζεται ως πηγή ενέργειας, μπορεί με την καύση του να μετατραπεί σε μια άλλη μορφή ενέργειας, την ηλεκτρική. Έτσι, για να καταστεί πιο σαφές υπάρχουν πολλές πηγές ενέργειας στη φύση, που αποθηκεύονται με μια συγκεκριμένη μορφή και μπορούν να μετατραπούν αλλά και να αποθηκευτούν ως άλλες για μελλοντική χρήση. Με την εκμετάλλευση και χρήση, λοιπόν, των πηγών ενέργειας καθίσταται δυνατή η παραγωγή έργου, αφού αυτό εκφράζει την ενέργεια που μεταφέρεται από ένα σώμα σε ένα άλλο ή που μετατρέπεται από τη μια μορφή στην άλλη.[1] Συνεπώς, ενέργεια και έργο είναι δύο συνυφασμένες έννοιες, για αυτό και ως ενέργεια ορίζεται η ικανότητα ενός σώματος ή και συστήματος να παράγει έργο.[1] Στο διεθνές σύστημα μονάδων (S.I.) μετριέται σε Joule(J), όπου ουσιαστικά 1J ισούται με το έργο που παράχθηκε κατά την άσκηση σε ένα σώμα ή σύστημα δύναμης 1N και εξαιτίας της εκείνο μετακινείται κατά 1m. (J = N*m)[1]

1.2. ΜΟΡΦΕΣ ΚΑΙ ΠΗΓΕΣ ΕΝΕΡΓΕΙΑΣ

1.2.1. ΜΟΡΦΕΣ ΕΝΕΡΓΕΙΑΣ

Όπως προαναφέρθηκε, η ενέργεια διατηρείται μέσα στο χρόνο και αλλάζει μορφές. Αυτό μπορεί να επιτευχθεί, είτε με φυσικές διαδικασίες (π.χ. ο άνεμος μπορεί να κινήσει ένα σώμα και έτσι αυτό αποκτά κινητική ενέργεια), είτε με διαδικασίες που οφείλονται στον άνθρωπο και τις μηχανές που επινόησε (π.χ. ο καυστήρας του καλοριφέρ με τις κατάλληλες διεργασίες μπορεί να μετατρέψει τη χημική ενέργεια που περιέχεται στο πετρέλαιο σε θερμική).

Στα παραπάνω παραδείγματα, προκειμένου να εξηγηθούν οι τρόποι με τους οποίους η ενέργεια μπορεί να μετατραπεί από μια μορφή σε μια άλλη, αναφέρθηκαν

ήδη τρεις μορφές ενέργειας (κινητική, χημική και θερμική). Στο σημείο αυτό θα παρουσιαστούν σύντομα όλες οι μορφές ενέργειας, που είναι οι εξής[1]:

1. Κινητική ενέργεια: Πρόκειται για τη μορφή ενέργειας που συνδέεται με την κίνηση ενός σώματος ή συστήματος, είτε αυτή είναι οριζόντια ή κάθετη, είτε περιστροφική, είτε ταλάντωση.
2. Δυναμική ενέργεια: Πρόκειται για τη μορφή ενέργειας που σχετίζεται με τη θέση ενός σώματος ή συστήματος στο χώρο και μπορεί να διακριθεί σε βαρυτική και σε ελαστική δυναμική ενέργεια.
3. Μηχανική ενέργεια: Πρόκειται για το άθροισμα της κινητικής και της δυναμικής ενέργειας ενός συστήματος.
4. Ηλεκτρική ενέργεια: Πρόκειται για την ενέργεια που προκύπτει από την κίνηση των ηλεκτρονίων ή άλλων φορτισμένων σωματιδίων (ιόντα).
5. Μαγνητική ενέργεια: Πρόκειται για την ενέργεια που προκύπτει από την έλξη ή την απώθηση, αντίθετων ή ίδιων μαγνητικών πεδίων αντίστοιχα.
6. Πυρηνική ενέργεια: Πρόκειται για την ενέργεια που βρίσκεται αποθηκευμένη στον πυρήνα του ατόμου και απελευθερώνεται με την πυρηνική σχάση ή σύντηξη.
7. Χημική ενέργεια: Πρόκειται για την ενέργεια που βρίσκεται αποθηκευμένη σε μια χημική ένωση και προκύπτει από τους δεσμούς που συγκρατούν τα άτομα στο μόριο της ένωσης αυτής.
8. Θερμική ενέργεια: Πρόκειται για την ενέργεια που σχετίζεται με την κινητική ενέργεια των σωματιδίων της ύλης, καθώς αυτά με την κίνηση τους δημιουργούν θερμότητα που μπορεί να μεταδοθεί από ένα σώμα σε ένα άλλο με αγωγή, συναγωγή και ακτινοβολία.
9. Φωτεινή ενέργεια: Πρόκειται για την ενέργεια που σχετίζεται με την ηλεκτρομαγνητική ακτινοβολία.

Η ενέργεια, λοιπόν, μπορεί να έχει κάποια από τις παραπάνω μορφές και να διατηρηθεί στην αρχική της μορφή ή να μετατραπεί σε κάποια άλλη. Με βάση ότι η ενέργεια δε δημιουργείται από το μηδέν και δε χάνεται, προκύπτει ότι προϋπάρχει αποθηκευμένη με κάποια μορφή στο περιβάλλον. Έτσι, αυτή μπορεί να εμπεριέχεται, για παράδειγμα, στο υπέδαφος της Γης και πιο συγκεκριμένα στα ορυκτά καύσιμα ή ακόμη και στον άνεμο. Όλα αυτά τα μέρη, στα οποία βρίσκεται αποθηκευμένη και από εκείνα μπορεί να αντληθεί και εν συνεχεία να χρησιμοποιηθεί αποτελούν πηγές

ενέργειας. Αυτές διακρίνονται σε δύο κατηγορίες: στις μη Ανανεώσιμες Πηγές Ενέργειας και στις Ανανεώσιμες Πηγές Ενέργειας.

1.2.2. ΜΗ ΑΝΑΝΕΩΣΙΜΕΣ ΠΗΓΕΣ ΕΝΕΡΓΕΙΑΣ

Με τον παραπάνω όρο καλούνται εκείνες οι πηγές ενέργειας, οι οποίες είναι πεπερασμένες και κάποια χρονική στιγμή θα εξαντληθούν. Σε αυτές περιλαμβάνονται φυσικοί πόροι που δεν είναι ανεξάντλητοι και πιο συγκεκριμένα τα ορυκτά καύσιμα, όπως οι γαιάνθρακες, το πετρέλαιο και το φυσικό αέριο. Ωστόσο, το γεγονός ότι οι πόροι αυτοί χαρακτηρίζονται μη ανεξάντλητοι δεν αληθεύει πλήρως, καθώς αναπληρώνονται με έναν συγκεκριμένο ρυθμό μέσω των γεωλογικών διεργασιών, ο οποίος, όμως, είναι μικρότερος συγκριτικά με το ρυθμό εκμετάλλευσης και κατανάλωσης τους από τον άνθρωπο, με αποτέλεσμα την ολοένα και μεγαλύτερη μείωση των αποθεμάτων τους και τη σταδιακή εξάντλησή τους.

1.2.3. ΑΝΑΝΕΩΣΙΜΕΣ ΠΗΓΕΣ ΕΝΕΡΓΕΙΑΣ

Με αυτόν τον όρο λογίζονται εκείνες οι πηγές ενέργειας, οι οποίες βρίσκονται σε αφθονία στο φυσικό περιβάλλον. Συχνά συναντώνται και ως εναλλακτικές πηγές ενέργειας προκειμένου να διαφοροποιηθούν από τις συμβατικές ενεργειακές πηγές, και η χρήση τους αποδεικνύεται φιλικότερη προς το περιβάλλον συγκριτικά με τη χρήση των συμβατικών, αφού δεν προκύπτει έκλυση ρύπων. Τέτοιες πηγές αποτελούν ο ήλιος, ο άνεμος, οι υδατοπτώσεις και η κυματική ενέργεια των ωκεανών, η γεωθερμία, το υδρογόνο και η βιομάζα.

- <u>Ηλιακή ενέργεια</u>

Πρόκειται για τη συνολική ηλεκτρομαγνητική ακτινοβολία με μήκη κύματος από 0,3 έως και 3μm, που εκπέμπεται από τον Ήλιο και η διάθεσή της είναι σταθερή[2]. Η ηλιακή ενέργεια παρέχει όλες τις μορφές ενέργειας στη Γη, εκτός από τη γεωθερμική και την πυρηνική.

Μια πρωτοποριακή ανακάλυψη για την αποθήκευσή και τη χρήση της με τη μορφή θερμότητας έχει ήδη επιτευχθεί από ερευνητική επιστημονική ομάδα στο Πανεπιστήμιο Τεχνολογίας Chalmers του Γκέτεμποργκ της Σουηδίας, χωρίς να έχει κυκλοφορήσει,

όμως, ακόμη στην αγορά και με τη δυνατότητα επέκτασης της τεχνολογίας αυτής μελλοντικά για την παραγωγή και ηλεκτρισμού.[3] Σύμφωνα με την τεχνολογία αυτή, στο αντίστοιχο σύστημα αποθήκευσης περιλαμβάνεται ένα υγρό μόριο από άνθρακα, υδρογόνο και άζωτο, όπου προσπίπτοντας σε αυτό ηλιακή ακτίνα, καταφέρνει να συλλέξει την ηλιακή ενέργεια και να τη συγκρατήσει, έως ότου να ξεκινήσει η απελευθέρωση θερμότητας από αυτό με τη χρήση ενός καταλύτη, με αποτέλεσμα εγκαθιστώντας το μέσω ενός πολυστρωματικού υλικού σε παράθυρα σπιτιών ή και οχήματα, να επιτευχθεί η θέρμανσή τους.[3]

Η εκμετάλλευση και χρήση της ηλιακής ενέργειας επιτυγχάνεται μέσω τριών ειδών συστημάτων: τα παθητικά ηλιακά συστήματα, τα ενεργητικά ηλιακά συστήματα και τα φωτοβολταϊκά συστήματα. Στην κατηγορία των παθητικών ηλιακών συστημάτων, η ηλιακή ακτινοβολία λαμβάνεται και χρησιμοποιείται, χωρίς να έχει προηγηθεί μετατροπή της, με τα συστήματα αυτά να βρίσκουν εφαρμογή στον βιοκλιματικό σχεδιασμό κτιρίων, ενώ σε εκείνη των ενεργητικών ηλιακών συστημάτων, η ηλιακή ακτινοβολία συλλέγεται και μετατρέπεται σε θερμότητα, για την αξιοποίηση της τελευταίας σε θέρμανση νερού σε νοικοκυριά, θέρμανση βιομηχανικών διεργασιών, κ.α.[2] Στην κατηγορία των φωτοβολταϊκών συστημάτων, η ηλιακή ακτινοβολία προσπίπτει στο φωτοβολταϊκό πλαίσιο και με τις κατάλληλες διαδικασίες μετατρέπεται σε ηλεκτρισμό για χρήση σε αυτόνομη ηλεκτροδότηση μεμονωμένης κατοικίας, ηλεκτροπαραγωγή για τροφοδότηση δικτύου, κ.α.[2] Με άλλα λόγια, τα παθητικά και ενεργητικά ηλιακά συστήματα εκμεταλλεύονται τη θερμότητα που προκύπτει από την ηλιακή ακτινοβολία, ενώ τα φωτοβολταϊκά συστήματα την ηλιακή ακτινοβολία κάθε αυτή, ώστε να μετατραπεί, μέσω του φωτοβολταϊκού φαινομένου, σε ηλεκτρισμό.

- <u>Αιολική ενέργεια</u>

Πρόκειται για την ενέργεια που προκύπτει από την κίνηση αερίων μαζών. Αυτή την κινητική ενέργεια του ανέμου, μπορούν να διαχειριστούν και να εκμεταλλευτούν οι αιολικές μηχανές, οι οποίες μετατρέπουν την κινητική ενέργεια του ανέμου σε μια άλλη μορφή ενέργειας. Πιο συχνά στη σημερινή εποχή, για την εκμετάλλευση της αιολικής ενέργειας, χρησιμοποιούνται οι ανεμογεννήτριες, οι οποίες εμπίπτουν στην κατηγορία των αιολικών μηχανών και μετατρέπουν την κινητική ενέργεια πρώτα σε μηχανική και στη συνέχεια την τελευταία σε ηλεκτρική. Διαθέτουν αεροδυναμικά σχεδιασμένα πτερύγια, συστήματα μετάδοσης κίνησης για τη μετατροπή της ενέργεια από κινητική σε μηχανική και μεγάλης απόδοσης ηλεκτρογεννήτριες για τη μετατροπή της ενέργειας από μηχανική σε ηλεκτρική, ενώ συνήθως τοποθετούνται μαζί σε περιοχές υψηλού

αιολικού δυναμικού, δημιουργώντας έτσι τα λεγόμενα αιολικά πάρκα.[4] Για την επιλογή της θέσης των εγκαταστάσεων ενός αιολικού πάρκου κρίνονται αναγκαίες η γνώση του αιολικού δυναμικού μιας περιοχής, δηλαδή η ένταση και οι διακυμάνσεις του ανέμου σε μέτρο και διεύθυνση, καθώς και τα πιθανά ακραία καιρικά φαινόμενα που μπορούν να συμβούν εκεί, μελετώντας το αντίστοιχο ιστορικό της περιοχής. Ακόμη, πρέπει να εξεταστεί η συμβατότητα της λειτουργίας των ανεμογεννητριών που απαρτίζουν το πάρκο με εκείνη του ηλεκτρικού δικτύου και η συμμόρφωση με την κείμενη νομοθεσία για το περιβάλλον, ώστε οι εγκαταστάσεις να μην έχουν αρνητικές επιπτώσεις σε αυτό.

- Υδάτινη ενέργεια

Πρόκειται για την ενέργεια που προσφέρεται από το νερό σε κάθε έκφανση του, είτε είναι το γλυκό νερό μιας λίμνης, είτε το αλμυρό νερό της θάλασσας και των ωκεανών, με αποτέλεσμα αυτή να διαχωρίζεται στην υδροδυναμική ενέργεια, στην ωκεάνια θερμική ενέργεια, στην κυματική ενέργεια, στην παλιρροϊκή ενέργεια και στην ωσμωτική ενέργεια.

Η υδροδυναμική ενέργεια αναφέρεται στη δυναμική ενέργεια του νερού, που συγκεντρώνεται σε λίμνες και ποτάμια, ύστερα από τις υδατοπτώσεις, τα οποία στην περίπτωση που βρίσκονται σε μεγαλύτερο υψόμετρο από κοντινές περιοχές, όπου μπορούν να εγκατασταθούν υδροηλεκτρικά εργοστάσια, μπορούμε να τα εκμεταλλευτούμε για την παραγωγή ηλεκτρισμού.[5] Με τη λειτουργία αυτών, όπου το νερό πέφτει από μεγάλο ύψος και μέσω υδροστροβίλου η δυναμική ενέργεια του νερού μετατρέπεται σε κινητική και εκείνη μέσω γεννήτριας σε ηλεκτρική, επιτυγχάνεται η παραγωγή ηλεκτρισμού.

Ξεκινώντας από ένα σημείο μεγαλύτερου υψομέτρου σε σχέση με το σημείο εγκατάστασης του υδροηλεκτρικού εργοστασίου, η ενέργεια του νερού θεωρείται δυναμική. Όταν αυτό απελευθερώνεται από τον άνω ταμιευτήρα , ρέει προς τα κάτω και περνώντας πρώτα από τον υδροστρόβιλο τον θέτει σε κίνηση (η δυναμική ενέργεια μετατρέπεται σε κινητική) και αυτός όντας συνδεδεμένος με κατάλληλη διάταξη με τη γεννήτρια παράγει ηλεκτρική ενέργεια.[5] Προκύπτει, λοιπόν, ότι η θέση για τις εγκαταστάσεις ενός υδροηλεκτρικού εργοστασίου απαιτεί από τη μια, μια περιοχή με υψηλό υδάτινο δυναμικό, και από την άλλη, μια περιοχή που να υπάρχουν αρκετά επίπεδα σημαντικής υψομετρικής διαφοράς.

Η κυματική ενέργεια αναφέρεται σε εκείνο το είδος ενέργειας που προκύπτει από τη δημιουργία μεγάλων θαλάσσιων και ωκεάνιων κυμάτων λόγω του ανέμου και δυνάμεθα να την εκμεταλλευτούμε για την παραγωγή ηλεκτρικής ισχύος με εγκαταστάσεις σε παράκτιες περιοχές, στις οποίες τα κύματα παρουσιάζουν μεγάλο πλάτος τοπικά και μεγάλη συχνότητα.[5]

Η παλιρροϊκή ενέργεια προκαλείται, τόσο από δυνάμεις της Σελήνης προς τη Γη, όσο και από τις βαρυτικές δυνάμεις της ίδιας της Γης.[55] Αυτές, προκαλούν συγκεκριμένες ροές στο εσωτερικό των θαλασσών και των ωκεανών , που εμφανίζονται με την άνοδο και την πτώση του ύψους των υδάτων και η ενέργεια τους μπούμε να την εκμεταλλευτούμε για να παραχθεί ηλεκτρισμός σε μικρή κλίμακα, λόγω της περιοδικότητας του φαινομένου.[5]

- <u>Γεωθερμική ενέργεια</u>

Η γεωθερμική ενέργεια (ή γεωθερμία) σχετίζεται με την ενέργεια που προκύπτει από τα ρευστά που προέρχονται από το εσωτερικό της Γης. Πρόκειται για τη θερμική ενέργεια που βρίσκεται αποθηκευμένη στο εσωτερικό της Γης και αποδίδεται με την έκλυση ποσοτήτων θερμού αέρα, ατμού ή νερού (γεωθερμικά ρευστά), άλλοτε με φυσικό τρόπο από το υπέδαφος προς την ατμόσφαιρα και άλλοτε με τεχνητό, πραγματοποιώντας γεωτρήσεις.[6] Δεδομένου ότι αυτή βρίσκεται αποθηκευμένη στα έγκατα της Γης σε τεράστια ποσότητα, θεωρείται μη εξαντλήσιμη και άρα αποτελεί Ανανεώσιμη Πηγή Ενέργειας, που μπορούμε να εκμεταλλευτούμε για θέρμανση αλλά και για παραγωγή ηλεκτρισμού.

Ουσιαστικά, η λάβα στο εσωτερικό της Γης θερμαίνει τα πετρώματα που βρίσκονται στον μανδύα της και εκείνα διαδοχικά τα επίπεδα του υπεδάφους, στα οποία περιέχονται τα γεωθερμικά ρευστά και στη συνέχεια αυτά, είτε με φυσικό, είτε με τεχνητό τρόπο αναβλύζουν στην επιφάνεια. Το μέρος που τα ρευστά αυτά παραμένουν αποθηκευμένα κάτω από την επιφάνεια της Γης καλείται γεωθερμικός ταμιευτήρας και αναφέρεται στο γεωθερμικό δυναμικό του υπεδάφους μιας περιοχής, καθώς αυτό μπορεί να διαφέρει από περιοχή σε περιοχή.[6]

- <u>Κυψέλες καυσίμου υδρογόνου</u>

Πρόκειται για την ενέργεια που σχετίζεται με το αέριο υδρογόνο και προκύπτει από τη χημική αντίδραση: $2H_2 + O_2 \rightarrow 2H_2O$ + ενέργεια

Η παραγόμενη ενέργεια δύναται να αποθηκευτεί στις κυψέλες καυσίμου υδρογόνου, ενώ οι υδρατμοί να απελευθερωθούν στο περιβάλλον, δεδομένου ότι δεν είναι επιβλαβείς για αυτό. Για την εκμετάλλευση αυτής ορίζεται ως προϋπόθεση, η λιγότερη περιβαλλοντική επιβάρυνση από τη χρήση των κυψελών καυσίμου αλλά και από τον τρόπο παραγωγής του καυσίμου του υδρογόνου. Ένας τρόπος παραγωγής του θα μπορούσε να είναι ο διαχωρισμός του από το μεθάνιο (CH_4), μέσω χημικής αντίδρασης που περιλαμβάνει άνθρακα, οξυγόνο και ατμό, που όμως θα ήταν επιζήμιος για το περιβάλλον, δεδομένου ότι εκλύονται μεγάλα ποσά διοξειδίου του άνθρακα στην ατμόσφαιρα, ανά μονάδα παραγόμενης θερμότητας.[7] Υπό το πρίσμα της ικανοποίησης της προϋπόθεσης για μικρότερη περιβαλλοντική επιβάρυνση από τον τρόπο παραγωγής του αερίου υδρογόνου, πολλές ερευνητικές ομάδες το διερευνούν και προκύπτουν νέες προτάσεις και ιδέες για την παραγωγή του, όπως αυτή του χημικού Daniel Nocera, σύμφωνα με την οποία ένα τεχνητό φύλλο από πυρίτιο μπορεί να παράγει υδρογόνο και οξυγόνο, αν τοποθετηθεί σε ένα ποτήρι με νερό βρύσης και πέσει πάνω του ηλιακό φως.[7]

Οι κυψέλες καυσίμου υδρογόνου μπορούν να συνδυάσουν με χημικό τρόπο υδρογόνο και οξυγόνο και να παράγουν ηλεκτροχημικά ηλεκτρισμό, θερμότητα και νερό, λειτουργώντας με υψηλή απόδοση της τάξεως του 40-80% ανάλογα τον τύπο τους, που διαχωρίζεται με βάση τον ηλεκτρολύτη που χρησιμοποιείται και τη θερμοκρασία λειτουργίας τους.[7] Τα συστήματα που αποτελούνται από αυτές περιέχουν συστοιχία κυψελών, κατάλληλη διάταξη επεξεργασίας του καυσίμου και διατάξεις μεταφοράς θερμότητας και παραλαβής της ισχύος που παράγεται, ενώ συνοδεύονται από αντίστοιχα συστήματα ελέγχου.[7] Στηρίζουν τη λειτουργία τους στο ότι το υδρογόνο επανενώνεται με το οξυγόνο, προκύπτοντας έτσι ηλεκτρικό ρεύμα, που είναι το αντίθετο της ηλεκτρόλυσης, ενώ τα λειτουργικά τους μέρη αποτελούν τα ηλεκτρόδια άνοδος και κάθοδος και ο ηλεκτρολύτης που παρεμβάλλεται σε αυτά.[7]

- <u>Βιοενέργεια</u>

Πρόκειται για την ενέργεια που απελευθερώνεται μέσω της καύσης των βιοκαυσίμων, τα οποία προκύπτουν ύστερα από επεξεργασία με διάφορες μεθόδους (θερμικές, χημικές ή βιοχημικές μέθοδοι) της βιομάζας.[8] Στη βιομάζα, εμπίπτει το βιολογικό υλικό, προερχόμενο από ζωντανούς οργανισμούς και δύναται να είναι τόσο φυτικό προϊόν, όσο και ζωϊκό απόβλητο ή απόβλητο αστικής φύσης.[8] Εμφανίζει μικρό χρονικό διάστημα αναπλήρωσης και τα συστήματα που τη χρησιμοποιούν ως πρώτη

ύλη για την παραγωγή ισχύος, δίνουν ως αποτέλεσμα ηλεκτρική ισχύ, τη λεγόμενη βιο-ισχύ.

Ως πρώτες ύλες βιομάζας, για τη χρήση από συστήματα που την εκμεταλλεύονται για την παραγωγή ηλεκτρικής ισχύος, μπορούν να είναι υπολείμματα ξύλου, είτε προερχόμενα από βιομηχανίες (π.χ. πριονίδια) και το αστικό περιβάλλον (π.χ. σπασμένες ξύλινες παλέτες και άλλα ξύλινα προϊόντα που καταλήγουν στις χωματερές), είτε προερχόμενα από το φυσικό περιβάλλον (π.χ. μη χρησιμοποιημένα υπολείμματα υλοτομίας από δασικές εκτάσεις), γεωργικά υπολείμματα (π.χ. άχυρα σιτηρών), βιομηχανικά απορρίμματα (π.χ. υπολείμματα φρούτων που δεν ανταποκρίνονται στον ποιοτικό έλεγχο της βιομηχανίας και τελικά απορρίπτονται), αστικά απορρίμματα (π.χ. οικιακά απόβλητα) και ζωϊκά απορρίματα (π.χ. κοπριά από βοοειδή σε εγκατάσταση εκτροφείου).[] Όλα αυτά με τις κατάλληλες μετατροπές έχουν ως αποτέλεσμα την παραγωγή βιοκαυσίμων, τα οποία μπορεί να είναι στερεά, υγρά ή και αέρια και με την καύση τους να παραχθεί ηλεκτρική ενέργεια αλλά και θερμότητα.

Η εκμετάλλευση και χρήση βιομάζας μπορεί να αποδώσει τη βιοενέργεια, που δύναται να καλύπτει μεγάλο μέρος της συνολικής ενεργειακής παραγωγής, αποτελώντας μια ανεξάντλητη πηγή ενέργειας, που εμφανίζει μειωμένες εκπομπές διοξειδίου του άνθρακα και άλλων ενώσεων (π.χ. οξείδια του αζώτου) σε σχέση με συμβατικούς τρόπους παραγωγής ενέργειας και αποτρέπει τη συσσώρευση υπολειμμάτων και απορριμμάτων κάθε είδους, δεδομένου ότι αυτά αξιοποιούνται.

1.3. ΑΠΟΘΗΚΕΥΣΗ ΕΝΕΡΓΕΙΑΣ

Είναι γεγονός ότι τα τελευταία χρόνια, η ζήτηση για κατανάλωση ενέργειας είναι διαρκώς αυξανόμενη, το οποίο προϋποθέτει και την αντίστοιχη αύξηση στην παραγωγή της. Πράγματι, η παραγωγή της έχει αυξηθεί σημαντικά τις τελευταίες δεκαετίες, με δεδομένες τις ολοένα μεγαλύτερες ανάγκες των νοικοκυριών αλλά και των βιομηχανιών για ενέργεια. Ενδεικτικά, η παγκόσμια ζήτηση για ενέργεια αυξήθηκε κατά 4,5% το 2021, σύμφωνα με έρευνα του Διεθνούς Οργανισμού Ενέργειας.[9] Ωστόσο, ακόμη και σήμερα, παρόλο που έχει γίνει μια αξιοσημείωτη στροφή στις Ανανεώσιμες Πηγές Ενέργειας, το μερίδιο συνεισφοράς στην παγκόσμια παραγωγή ενέργειας από συμβατικές ενεργειακές πηγές παραμένει υψηλό. Έτσι, λοιπόν, υπάρχει περιθώριο για περαιτέρω διείσδυση των Ανανεώσιμων Πηγών Ενέργειας στο μερίδιο αυτό, προκειμένου να περιοριστούν οι παραδοσιακοί τρόποι για την παραγωγή της και να ικανοποιηθούν παράλληλα οι διεθνείς συμφωνίες για το περιβάλλον και το κλίμα,

όπως η Συμφωνία του Παρισιού που υπογράφηκε το 2016 και προβλέπει η μέση παγκόσμια αύξηση της θερμοκρασίας να περιοριστεί κάτω από τους 2^{o}C. Για να επιτευχθεί αυτό, η συνεισφορά των ΑΠΕ στην παραγωγή ενέργειας θα πρέπει να αγγίζει το 57% παγκοσμίως, βάση του Διεθνούς Οργανισμού Ενέργειας.[9]

Όπως γίνεται αντιληπτό, οι ΑΠΕ και οι αντίστοιχες εγκαταστάσεις εκμετάλλευσης τους, πρόκειται να διαδραματίσουν καθοριστικό ρόλο στο ζήτημα της ενέργειας, καθώς εκτός του ότι είναι φιλικές προς το περιβάλλον, έχουν μικρό λειτουργικό κόστος και δύνανται να δώσουν λύση και στο θέμα της ενεργειακής εξάρτησης μιας χώρας από μια άλλη, από τη στιγμή που η κάθε χώρα μπορεί να χρησιμοποιήσει τις ΑΠΕ που υπάρχουν στην επικράτειά της.[10] Όμως, το γεγονός ότι η γεωγραφική τους κατανομή είναι διάσπαρτη συνδυαστικά με το ότι έχουν στοχαστική φύση και παρουσιάζουν μεγάλη μεταβλητότητα, καθιστά δύσκολη τη συγκέντρωσή τους σε μεγάλα μεγέθη ισχύος.[10] Παρόλα αυτά, είναι εφικτή αλλά και απαραίτητη η αποθήκευση ενέργειας προερχόμενη από ΑΠΕ, για να εγκαταλειφθούν σταδιακά οι συμβατικοί τρόποι παραγωγής ενέργειας και να αντικατασταθούν από λιγότερο επιβλαβείς για το περιβάλλον και τον άνθρωπο. Η αναγκαιότητα της αποθήκευσης ενέργειας προερχόμενης από ΑΠΕ συνίσταται στο γεγονός ότι οι ΑΠΕ δύνανται να εξυπηρετήσουν με μεγαλύτερη αποδοτικότητα από τους συμβατικούς τρόπους παραγωγής ενέργειας τη διαρκώς αυξανόμενη ζήτηση για ενέργεια, καθώς η παραγόμενη ενέργεια από αυτές δε χάνεται όπως στην περίπτωση παραγωγής από συμβατικούς τρόπους (π.χ. απώλεια θερμότητας κατά την παραγωγή ηλεκτρισμού από έναν ατμοηλεκτρικό σταθμό με την καύση ορυκτών καυσίμων). Με άλλα λόγια, υπάρχει ανάγκη για αποθήκευση ενέργειας προερχόμενης από ΑΠΕ για το λόγο ότι με την επίτευξη της μπορούν να τροφοδοτηθούν τα διάφορα δίκτυα, όποτε υπάρχει ανάγκη για ενέργεια. Επιπλέον, ένας άλλος λόγος είναι ότι οι ΑΠΕ παρουσιάζουν μεγάλη μεταβλητότητα και στοχαστικότητα, δηλαδή υπάρχει μεγάλο εύρος και τυχαία εξέλιξη στις τιμές της ενέργειας που αποδίδουν, και πρέπει με κάποιον τρόπο αυτά τα χαρακτηριστικά να τεθούν υπό έλεγχο. Αυτά τα χαρακτηριστικά είναι ταυτόσημα με την έννοια της αστάθειας των ΑΠΕ, η οποία μπορεί να προκαλέσει δυσκολίες μέχρι και διακοπή στην τροφοδοσία της ενέργειας (black out), στην περίπτωση που αυτές διοχετευτούν απευθείας στα εκάστοτε δίκτυα μεταφοράς ηλεκτρικής ενέργειας. Έτσι, συλλέγοντας και αποθηκεύοντας την ενέργεια από τις ΑΠΕ, αυτή μπορεί να προωθηθεί στα δίκτυα με έναν πιο ελεγχόμενο και ομαλό τρόπο, με τη βοήθεια πάντα του αντίστοιχου συστήματος αποθήκευσης και τις λειτουργικές του δυνατότητες.

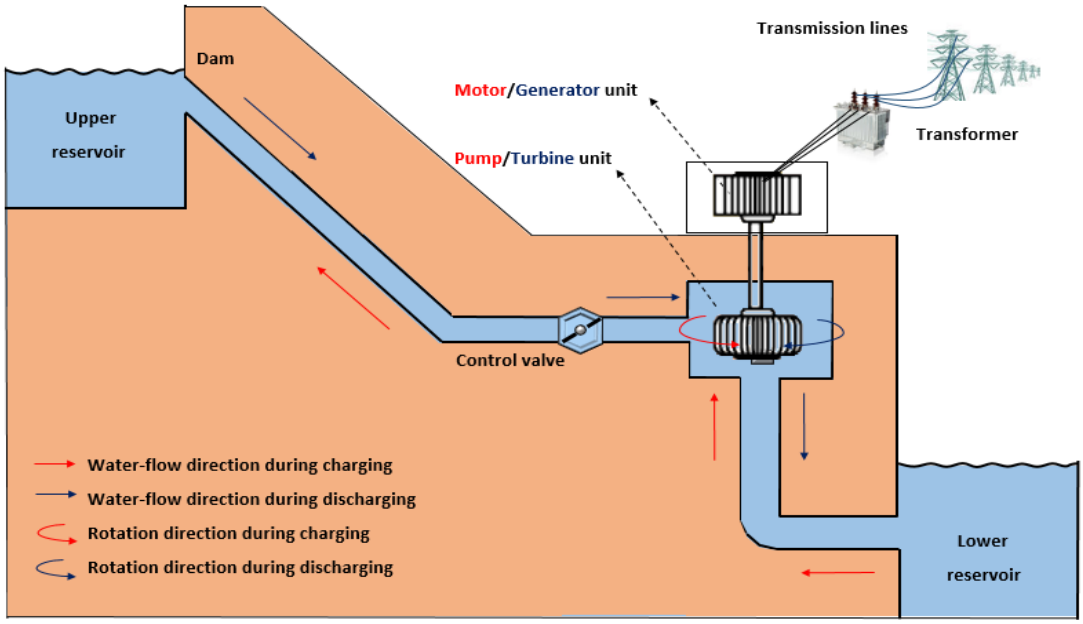
Ήδη υπάρχουν αρκετές τεχνολογίες αποθήκευσης ενέργειας που συμβάλλουν σε αυτή την κατεύθυνση, όπως το σύστημα αποθήκευσης ενέργειας μέσω

αντλησιοταμίευσης, το σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα και οι μπαταρίες.

- Σύστημα Αποθήκευσης Ενέργειας μέσω Αντλησιοταμίευσης (Pumped Hydro Energy Storage System)

Πρόκειται για την πιο γνωστή τεχνολογία αποθήκευσης ενέργειας και αξίζει να σημειωθεί ότι εγκαταστάσεις τέτοιου τύπου, σε πρώιμο στάδιο βέβαια, είχαν αναπτυχθεί στις Άλπεις ήδη από το 1890.[9] Αν και η ύπαρξη εγκαταστάσεων τέτοιων συστημάτων προϋποθέτει ιδιαίτερες γεωμορφολογικές συνθήκες, γεγονός που δεν επιτρέπει τη δημιουργία τους σε οποιοδήποτε μέρος, προσφέρουν υψηλή αποτελεσματικότητα, μεγάλη χωρητικότητα ενέργειας και μακρά περίοδο αποθήκευσής της.

Η λειτουργία ενός τέτοιου συστήματος αποθήκευσης ενέργειας αποτελείται από δύο επιμέρους διαδικασίες: αυτή της φόρτισης και εκείνη της εκφόρτισης. Προκειμένου να υλοποιηθεί κάθε μια από αυτές, το εν λόγω σύστημα στην πιο απλή μορφή του απαρτίζεται από δύο μεγάλους ταμιευτήρες νερού, οι οποίοι θα πρέπει να έχουν αρκετή υψομετρική διαφορά, μια αντλία, ώστε να αντλείται το νερό από τον χαμηλότερο ταμιευτήρα προς τον υψηλότερο κατά τη φόρτιση, μια ηλεκτρική μηχανή με διπλή λειτουργία, είτε ως ηλεκτρικός κινητήρας για τη μετατροπή της ηλεκτρικής ενέργειας του δικτύου σε μηχανική, είτε ως γεννήτρια για την αντίστροφη μετατροπή και μια τουρμπίνα, ώστε να παράγεται ηλεκτρικό ρεύμα όταν το νερό διοχετεύεται από τον υψηλότερο ταμιευτήρα προς τον χαμηλότερο κατά την εκφόρτιση. Ως διαδικασία φόρτισης του συστήματος νοείται η διαδικασία κατά την οποία η ηλεκτρική ενέργεια, η οποία προέρχεται τόσο από το δίκτυο παροχής ηλεκτρικού ρεύματος, όταν αυτό δεν είναι υπερφορτωμένο, δηλαδή σε ώρες μη αιχμής όπου υπάρχει χαμηλή ζήτηση, όσο και από ΑΠΕ κατά τις ίδιες ώρες, μετατρέπεται σε μηχανική μέσω του ηλεκτρικού κινητήρα (μοτέρ) και εν συνεχεία εκείνη σε δυναμική, αφού με την αντλία το νερό αντλείται από τον χαμηλό ταμιευτήρα, οδηγείται και αποθηκεύεται σε εκείνον του μεγαλύτερου ύψους.[9] Ως διαδικασία εκφόρτισης θεωρείται η ανάποδη διαδικασία, που υλοποιείται σε ώρες αιχμής και επομένως όταν υπάρχει μεγάλη ανάγκη και ζήτηση για ενέργεια και κατά την οποία το αποθηκευμένο νερό στον υψηλότερο ταμιευτήρα διοχετεύεται προς τον χαμηλότερο, θέτοντας σε λειτουργία την τουρμπίνα, ώστε η δυναμική ενέργεια να μετατραπεί σε κινητική και αυτή με τη σειρά της μέσω της γεννήτριας σε ηλεκτρική, ώστε να τροφοδοτήσει το δίκτυο με ηλεκτρικό ρεύμα.[9]



Εικόνα 1: Μέρη και τρόπος λειτουργίας ενός συστήματος αποθήκευσης ενέργειας με αντλησιοταμίευση

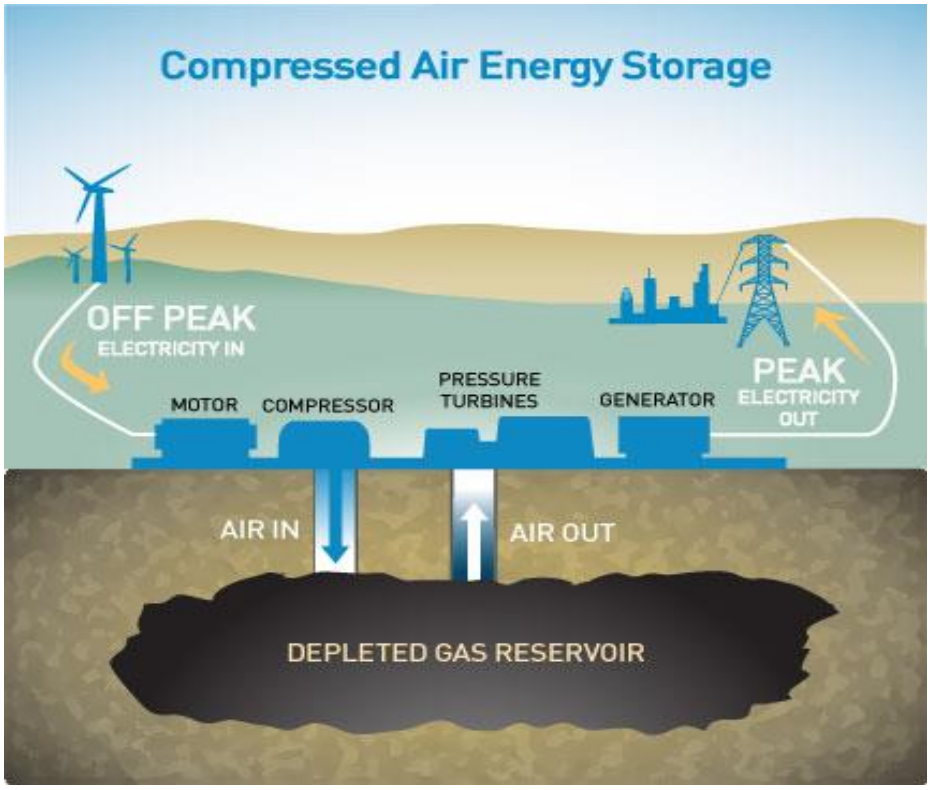
Πηγή: *https://www.researchgate.net/figure/Schematic-diagram-of-pumped-hydro-storage-plant_fig3_320755664*

- Σύστημα Αποθήκευσης Ενέργειας μέσω Πεπιεσμένου Ατμοσφαιρικού Αέρα (Compressed Air Energy Storage System)

Πρόκειται για μια τεχνολογία αποθήκευσης ενέργειας, μέσω συμπίεσης του ατμοσφαιρικού αέρα σε ειδική δεξαμενή, το ρόλο της οποίας μπορεί να διαδραματίσει, είτε κάποιο υπόγειο σπήλαιο, είτε κάποιο μεγάλου μεγέθους δοχείο. Παράγοντες που καθορίζουν την ποσότητα της αποθηκευμένης ενέργειας στη δεξαμενή είναι, τόσο ο όγκος αυτής, όσο και η θερμοκρασία και η πίεση, υπό τις οποίες αποθηκεύεται ο ατμοσφαιρικός αέρας. Τα λειτουργικά μέρη ενός τέτοιου συστήματος αποτελούν ένας ηλεκτρικός κινητήρας για την εκκίνηση του συμπιεστή, ο οποίος συμπιέζει τον αέρα σε πολλαπλά στάδια κατά τη διαδικασία φόρτισης του συστήματος στην αεροδεξαμενή, όπου και αποθηκεύεται, ένας συμπιεστής, μια αεροδεξαμενή, μια συστοιχία τουρμπινών καθώς και μια γεννήτρια για την παραγωγή ηλεκτρικού ρεύματος. Κατά τη φόρτιση, η ηλεκτρική ενέργεια από το δίκτυο, τροφοδοτεί τον κινητήρα, ο οποίος τη μετατρέπει σε μηχανική και με τη λειτουργία του αεροσυμπιεστή αυτή αποθηκεύεται με τη μορφή ατμοσφαιρικού αέρα στην αεροδεξαμενή.[9] Κατά την εκφόρτιση, ο αέρας που περιέχεται στη δεξαμενή διοχετεύεται στους αεροστρόβιλους για να μετατραπεί η

ενέργεια που περιέχεται σε αυτόν σε μηχανική και έπειτα με τη λειτουργία της γεννήτριας η μηχανική σε ηλεκτρική για την τροφοδοσία του δικτύου.[9]

Ανάλογα με τη διαχείριση της θερμότητας που εκλύεται ύστερα από την συμπίεση του αέρα στη δεξαμενή τα συστήματα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, χωρίζονται στα διαβατικά και στα αδιαβατικά συστήματα. Σε ένα διαβατικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, η θερμότητα που προκύπτει κατά τη συμπίεση δε συγκρατείται αλλά απομακρύνεται μέσω μιας ψυκτικής συσκευής και πριν ο αέρας αποσυμπιεστεί θερμαίνεται περνώντας από έναν θερμό θάλαμο και στη συνέχεια να προωθηθεί στις τουρμπίνες. Αντίθετα, σε ένα αδιαβατικό σύστημα, η θερμότητα που προκύπτει από το στάδιο συμπίεσης του αέρα αποθηκεύεται συνήθως σε ειδικά δοχεία, από τα οποία κατά το στάδιο της αποσυμπίεσης διοχετεύεται για να θερμάνει τον αέρα πριν αυτός εισέλθει στις τουρμπίνες.



Εικόνα 2: Τυπικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, που χρησιμοποιεί ως αεροδεξαμενή ένα υπόγειο σπήλαιο

Πηγή: https://www.pge.com/en_US/about-pge/environment/what-we-are-doing/compressed-air-energy-storage/compressed-air-energy-storage.page

Ενδεικτικό σύστημα αποθήκευσης ενέργειας με πεπιεσμένο αέρα της παραπάνω εικόνας είναι αυτό που λειτουργεί στο Huntorf της Γερμανίας από το 1978 και χρησιμοποιεί αλατωρυχείο ως χώρο αποθήκευσης του αέρα. Κατά τις περιόδους υψηλής ζήτησης ενέργειας τίθεται σε λειτουργία, ώστε ο πεπιεσμένος αέρας να χρησιμοποιηθεί για να γίνει καύση φυσικού αερίου σε αντίστοιχο θάλαμο και το προϊόν της καύσης να προωθηθεί σε αεροστρόβιλο για να τον θέσει σε κίνηση και να παραχθεί ηλεκτρισμός. Ουσιαστικά, σε αυτό το συμβατικό σύστημα αποθήκευσης ενέργειας με πεπιεσμένο αέρα, ο αέρας χρησιμοποιείται για να βελτιώσει και να ενισχύσει τη μονάδα καύσης.

- Αποθήκευση Ενέργειας μέσω Μπαταριών (Batteries)

Μέσω των μπαταριών η χημική ενέργεια που βρίσκεται αποθηκευμένη σε αυτές μετατρέπεται σε ηλεκτρική, όταν τεθούν σε λειτουργία και τροφοδοτήσουν κάποιο σύστημα. Αποτελούνται από πολλές διαφορετικές κυψέλες, μέσα στις οποίες υπάρχουν δύο ηλεκτρόδια, η άνοδος και η κάθοδος, καθώς και ένας ηλεκτρολύτης. Μπορεί να είναι είτε επαναφορτιζόμενες είτε όχι, με εκείνες που έχουν τη δυνατότητα να επαναφορτίζουν να διακρίνονται με βάση το υλικό των ηλεκτροδίων και του ηλεκτρολύτη, όπως μπαταρίες ιόντων λιθίου, νικελίου – κάδμιου, κ.α.

Οι μπαταρίες ιόντων λιθίου είναι επαναφορτιζόμενες και χρησιμοποιούνται στα ηλεκτρονικά αλλά και σε όλα τα ηλεκτρικά οχήματα. Σήμερα βρίσκονται σε ευρεία χρήση σε αυτοκίνητα και υποβρύχια αλλά και σε μικρά μη επανδρωμένα ιπτάμενα και θαλάσσια οχήματα. Στο εγγύς μέλλον μπορούν να χρησιμοποιηθούν και σε μεγάλης κλίμακας αεροπλάνα, πλοία, κλπ. . Διαθέτουν διάλυμα άλατος λιθίου ως ηλεκτρολύτη, ενώ το ηλεκτρόδιο της ανόδου αποτελείται από γραφίτη άνθρακα και το ηλεκτρόδιο της καθόδου από οξείδιο λιθίου. Κατά τη διαδικασία της φόρτισης, όπου ρεύμα από εξωτερική πηγή ισχύος διαρρέει τη μπαταρία, τα ιόντα του λιθίου μετακινούνται από το ηλεκτρόδιο της καθόδου προς το ηλεκτρόδιο της ανόδου και έτσι η μπαταρία λειτουργεί. Κατά τη διαδικασία της εκφόρτισης, η παραπάνω διαδικασία είναι αντίστροφη.

Στις μπαταρίες μολύβδου – οξέος ως ηλεκτρολύτης χρησιμοποιείται διάλυμα θειικού οξέος και ως ηλεκτρόδια το διοξείδιο του μολύβδου και ο σπογγώδης μόλυβδος. Όταν διαρρεύσει ρεύμα την μπαταρία, αυτή φορτίζεται και τα ηλεκτρόδια και ο ηλεκτρολύτης που κατά τη διαδικασία της εκφόρτισης έχουν αναχθεί σε νερό, επιστρέφουν στην αρχική τους κατάσταση για να λειτουργήσει. Αν και διαθέτουν μικρή

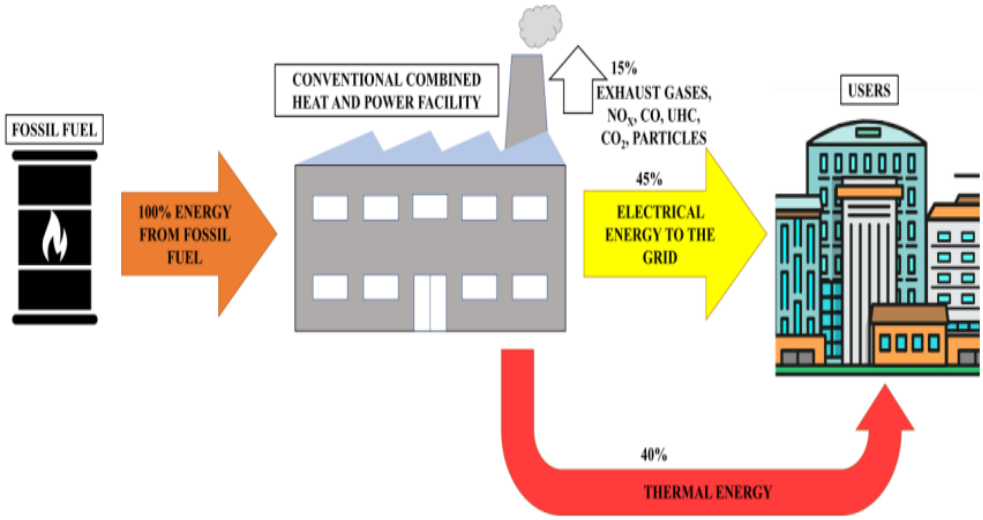
διάρκεια ζωής, χρησιμοποιούνται ιδιαίτερα στον κλάδο της βιομηχανίας, είτε ως σταθερές εγκαταστάσεις, είτε ως φορητές.

1.4.ΣΥΜΠΑΡΑΓΩΓΗ ΕΝΕΡΓΕΙΑΣ

Κατά τη λειτουργία ενός ατμοηλεκτρικού σταθμού, υπάρχει καύση ορυκτών καυσίμων, με τη θερμότητα που προκύπτει από αυτήν να θερμαίνει νερό για τη δημιουργία ατμού υψηλής πίεσης. Ο ατμός αυτός προωθείται σε μια τουρμπίνα για να τη θέση σε κίνηση και εκείνη συνδεδεμένη με μια γεννήτρια μεταφέρει την αντίστοιχη ενέργεια στην τελευταία για την παραγωγή ηλεκτρικής ενέργειας. Ωστόσο, τα ποσά της θερμότητας που προκύπτουν κατά τη λειτουργία του με την καύση των ορυκτών καυσίμων, αποβάλλονται στο περιβάλλον, τόσο ως καυσαέρια απευθείας, όσο και μέσω ψυκτικών κυκλωμάτων ύστερα από την επεξεργασία τους από αυτά. Με αυτόν τον τρόπο, όμως, καλύπτεται μόνο η ανάγκη για ηλεκτρική ενέργεια και όχι για παράδειγμα η ανάγκη για θέρμανση, η οποία καλύπτεται ξεχωριστά με την καύση εκ νέου κάποιου ορυκτού καυσίμου σε κάποιο άλλο μέσο (π.χ. καύση πετρελαίου στον αντίστοιχο καυστήρα).

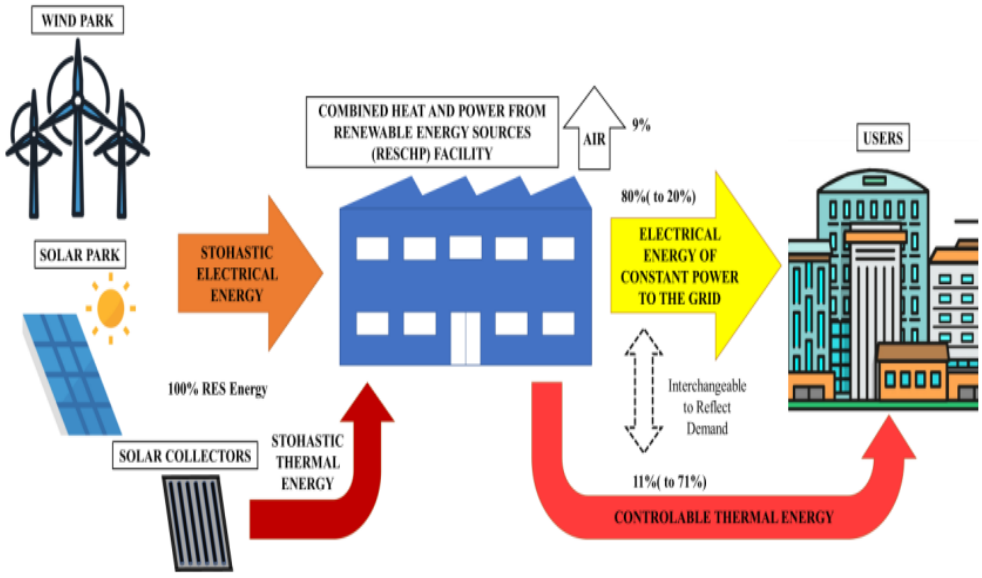
Έτσι, γίνεται σαφές πως είναι προτιμότερη παραγωγή ενέργειας που να εξυπηρετεί ταυτόχρονα και τις δυο ανάγκες, που έχει και ως αποτέλεσμα τη μείωση στην κατανάλωση των ορυκτών καυσίμων, αφού αυτά θα υποστούν μια φορά καύση και όχι δυο ξεχωριστές. Αυτό επιτυγχάνεται με τη συμπαραγωγή ηλεκτρισμού και θέρμανσης (ΣΗΘ ή CHP), κατά την οποία επιτυγχάνεται παραγωγή ηλεκτρικής και θερμικής ενέργειας από ίδια αρχική πηγή ενέργειας σε ένα ενιαίο σύστημα.[11] Αυτή η τεχνολογία παραγωγής ενέργειας εκμεταλλεύεται τη θερμότητα, που κατά τους συμβατικούς και ξεχωριστούς τρόπους ενέργειας θα αποβαλλόταν και άρα θα ήταν χαμένη, και τη χρησιμοποιεί με ωφέλιμο τρόπο, με αποτέλεσμα η απόδοση ενός συστήματος που στηρίζεται σε αυτήν την τεχνολογία να είναι μεγαλύτερη έναντι εκείνης των συστημάτων που στηρίζονται στις άλλες τεχνολογίες. Παράλληλα, αυτή η αυξημένη απόδοση των συστημάτων ΣΗΘ, έχει ως αποτέλεσμα τη μικρότερη κατανάλωση ορυκτών καυσίμων και συνεπώς έκλυση ρύπων σε μικρότερο βαθμό από την εξόρυξη, επεξεργασία και μεταφορά τους, σε σχέση με την παραγωγή ηλεκτρικής και θερμικής ενέργειας ξεχωριστά.[11] Παραδείγματα συστημάτων ΣΗΘ αποτελούν η τηλεθέρμανση που προκύπτει από τις λιγνιτικές μονάδες παραγωγής ισχύος στην Πτολεμαΐδα, καθώς και η κάλυψη των αναγκών ηλεκτρισμού και θέρμανσης σε γεωργική βιομηχανία και συγκεκριμένα σε εκκοκκιστήριο στην περιοχή της Βοιωτίας.

Όπως προκύπτει από τα παραπάνω και η λειτουργία ενός συστήματος που βασίζεται στη συμπαραγωγή ηλεκτρικής ενέργειας και θέρμανσης, έχει, αν και μικρότερη από τα συμβατικά συστήματα παραγωγής ενέργειας, χρήση και κατανάλωση καυσίμων. Συνεπώς, επιφέρει και τα παράγωγα της καύσης αυτών, όπως το διοξείδιο του άνθρακα και άλλα επικίνδυνα αέρια, γνωστά και ως αέρια του θερμοκηπίου, τα οποία έχουν σημαντικές περιβαλλοντικές επιπτώσεις. Παρόλο, που τέτοια συστήματα έχουν υψηλό βαθμό απόδοσης, η λειτουργία τους δε συμβαδίζει με τις ολοένα και πιο αυστηρές πολιτικές που ακολουθούνται για την προστασία του περιβάλλοντος αλλά και τις αντίστοιχες συμφωνίες που έχουν υπογραφεί. Είναι αναγκαία, λοιπόν, η επίτευξη νέων πιο «πράσινων» και φιλικών προς το περιβάλλον πρακτικών, που μπορούν να υλοποιηθούν με τη συμβολή των Ανανεώσιμων Πηγών Ενέργειας. Έτσι, μπορούν να υπάρξουν συστήματα ΣΗΘ, τα οποία να τροφοδοτούνται αποκλειστικά από ΑΠΕ, ώστε να υπάρχει μείωση των επικίνδυνων αερίων, έχοντας πρώτα επιλύσει το πρόβλημα της αστάθειας, που προκύπτει από την απευθείας διοχέτευση της παραγόμενης από ΑΠΕ ενέργειας στο δίκτυο, μέσω της αποθήκευσής της σε προηγούμενο στάδιο. Παρακάτω αποτυπώνονται η απόδοση της ηλεκτρικής και θερμικής ενέργειας, παραγόμενη από ένα συμβατικό σύστημα ΣΗΘ και από ένα αντίστοιχο τροφοδοτούμενο από ΑΠΕ, καθώς και η μείωση των εκπομπών αερίων του θερμοκηπίου από τη λειτουργία τους.



Εικόνα 3: Απόδοση και εκπομπή αερίων του θερμοκηπίου σε ποσοστά κατά τη λειτουργία συμβατικού συστήματος συμπαραγωγής ηλεκτρισμού και θέρμανσης (ΣΗΘ)

Πηγή: Renewable Energy Source for Combined Heat and Power (RESCHP) (2021)



Εικόνα 4: Απόδοση και εκπομπή ατμοσφαιρικού αέρα σε ποσοστά κατά τη λειτουργία συστήματος συμπαραγωγής ηλεκτρισμού και θέρμανσης (ΣΗΘ) τροφοδοτούμενο αποκλειστικά από ΑΠΕ

Πηγή: Renewable Energy Source for Combined Heat and Power (RESCHP) (2021)

1.5.ΑΔΙΑΒΑΤΙΚΟ ΣΥΣΤΗΜΑ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΜΕΣΩ ΠΕΠΙΕΣΜΕΝΟΥ ΑΤΜΟΣΦΑΙΡΙΚΟΥ ΑΕΡΑ

Όπως προαναφέρθηκε, δύναται και είναι αναγκαίο τα συμβατικά συστήματα ΣΗΘ να αντικατασταθούν από αντίστοιχα συστήματα, τα οποία, όμως, θα έχουν αποκλειστικά ως αρχική πηγή ενέργειας όχι ορυκτά καύσιμα αλλά ανανεώσιμες πηγές. Τέτοια συστήματα μπορούν να στηριχθούν στην τεχνολογία αποθήκευσης πεπιεσμένου ατμοσφαιρικού αέρα, όπως εκείνη εξηγήθηκε παραπάνω, να προσφέρουν απεξάρτηση από την κατανάλωση και χρήση των ορυκτών πόρων και να αποτελέσουν λύση στην ταυτόχρονη παραγωγή ηλεκτρικής και θερμικής ενέργειας. Το αδιαβατικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα εμπίπτει στην κατηγορία τέτοιων συστημάτων και μπορεί να υλοποιηθεί βάση της αντίστοιχης πρότασης των κυρίων Πίττα, Γκιάλα, Γεωργίου και Μούτσιου.

1.5.1.ΠΕΙΡΑΜΑΤΙΚΗ ΔΟΚΙΜΗ ΤΟΥ ΑΔΙΑΒΑΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΜΕΣΩ ΠΕΠΙΕΣΜΕΝΟΥ ΑΤΟΣΦΑΙΡΙΚΟΥ ΑΕΡΑ

Το σύστημα της πρότασης υλοποιήθηκε πειραματικά σε μικρή κλίμακα, χρησιμοποιώντας ως πηγή ενέργειας την ηλιακή, συλλεγμένη από 120 φωτοβολταϊκά πάνελ, ισχύος 240W το καθένα.[12] Η παραγόμενη ηλεκτρική ενέργεια χρησιμοποιήθηκε, ΄ώστε να τροφοδοτήσει τους δύο συμπιεστές του αέρα ισχύος 5,5kW ο καθένας, που με τη σειρά τους συμπίεσαν τον αέρα στα 200bar σε μια αεροδεξαμενή όγκου περίπου 1,8m^3, με χωρητικότητα αποθήκευσης ενέργειας 40kWh. Η απόδοση του συστήματος, ύστερα από έναν μήνα δοκιμών μετρήθηκε σε 92% και συγκεκριμένα σε 65% για την παραγωγή ηλεκτρικής ενέργειας και 27% για την παραγωγή θερμικής ενέργειας.[12]

1.5.2.ΛΕΙΤΟΥΡΓΙΑ ΚΑΙ ΘΕΡΜΟΔΥΝΑΜΙΚΟΣ ΚΥΚΛΟΣ ΤΟΥ ΑΔΙΑΒΑΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΜΕΣΩ ΠΕΠΙΕΣΜΕΝΟΥ ΑΤΜΟΣΦΑΡΙΚΟΥ ΑΕΡΑ

Όπως προαναφέρθηκε, στο αδιαβατικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, η θερμότητα που παράγεται κατά τη συμπίεση δεν απελευθερώνεται στο περιβάλλον, αλλά αποθηκεύεται σε ειδικά δοχεία για να επαναχρησιμοποιηθεί κατά την εκτόνωση του αέρα, δηλαδή αυτό στηρίζεται στη μη ανταλλαγή θερμότητας με το περιβάλλον (Αδιαβατική μεταβολή). Σε αυτό το σημείο, διευκρινίζεται ότι το σύστημα του οποίου η λειτουργία θα αναφερθεί και πάνω σε αυτό θα υλοποιηθεί η παρακολούθηση και ο έλεγχος είναι εκείνο που αναφέρεται στην αντίστοιχη πρόταση υλοποίησης τέτοιου συστήματος των κυρίων Πίττα, Γκιάλα, Γεωργίου και Μούτσιου.

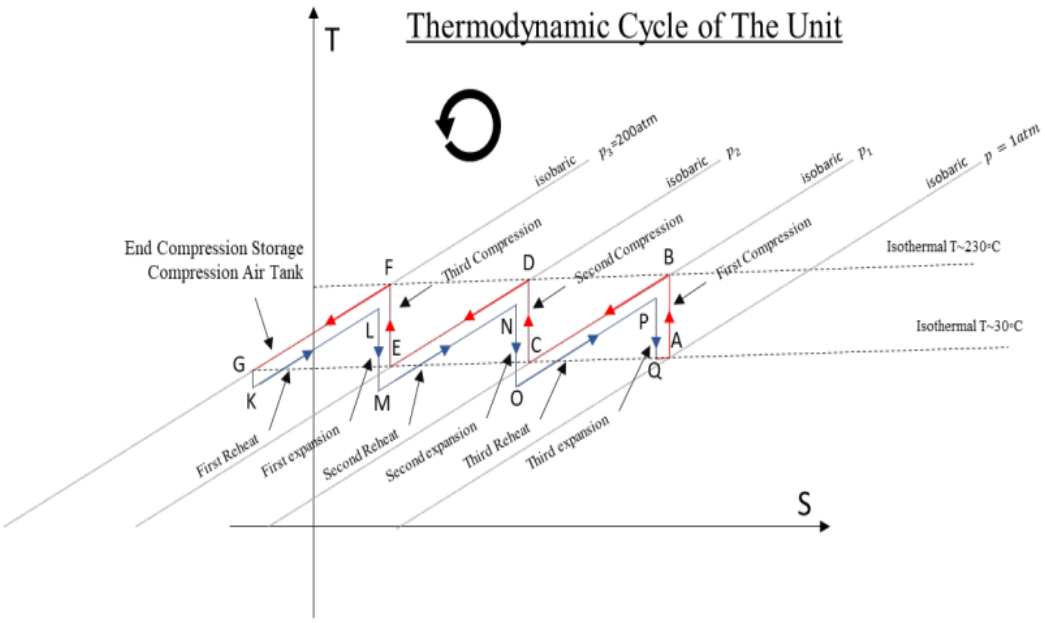
Κατά τη λειτουργία ενός τέτοιου συστήματος, ενέργεια από ανανεώσιμες πηγές χρησιμοποιείται, ώστε να τεθούν σε λειτουργία δυο συμπιεστές, οι οποίοι θα λειτουργούν με 3 στάδια ο καθένας, με τις τιμές εισόδου για την πίεση του αέρα στο πρώτο στάδιο να είναι 1 bar, στο δεύτερο 5,85 bar, στο τρίτο 34,2 bar και οι τιμές εξόδου 5,85 , 34,2 και 200 bar αντίστοιχα.[13] Η τιμή της θερμοκρασίας του αέρα κατά την περάτωση του κάθε σταδίου συμπίεσης είναι 270^oC.[13] Οι συμπιεστές αυτοί χρησιμοποιούν ένα σύστημα εσωτερικής ψύξης και θέρμανσης με ειδικό θερμικό λάδι ο ένας και νερό ο άλλος, ώστε η θερμότητα που παράγεται να μεταφέρεται και να αποθηκεύεται μέσω αυτών σε ειδικά ψυχρά και θερμά δοχεία αποθήκευσης και να

χρησιμοποιείται για τη ψύξη του αέρα κατά τη συμπίεση και τη θέρμανσή του κατά την εκτόνωση. Ο αέρας που συμπιέζεται από τον πρώτο συμπιεστή ψύχεται μέσω του ειδικού θερμικού λαδιού, ύστερα από την είσοδο του τελευταίου σε εναλλάκτη θερμότητας για να επιτευχθεί η επιθυμητή θερμοκρασία του και στη συνέχεια η θερμότητα που προκύπτει από το στάδιο της συμπίεσης αποθηκεύεται σε ειδικά δοχεία λαδιού, που είναι ικανά να αποθηκεύσουν το θερμικό λάδι σε υψηλές θερμοκρασίες. Η ίδια διαδικασία επιτελείται και από τον δεύτερο συμπιεστή, όπου εκεί ο αέρας ψύχεται μέσω του νερού, που έχει αντληθεί από το ψυχρό δοχείο αποθήκευσης νερού και αυτό ύστερα από την εισαγωγή του σε εναλλάκτη θερμότητας αποθηκεύεται στο θερμό δοχείο αποθήκευσης νερού, για να χρησιμοποιηθεί αργότερα κατά το τελευταίο στάδιο της εκτόνωσης.

Η αποθήκευση του αέρα γίνεται σε συνδεδεμένους μεταξύ τους συμβατικούς βιομηχανικούς αγωγούς, σχεδιασμένους έτσι ώστε να αντέχουν τον αέρα αποθηκευμένο σε υψηλή πίεση. Οι αγωγοί αυτοί είναι επικαλυμμένοι με ένα λεπτό κέλυφος από ατσάλι, ενώ ανάμεσα στο κέλυφος και στα τοιχώματα των αγωγών ρέει το θερμικό λάδι μεταξύ των δοχείων από τα οποία εισέρχεται και εξέρχεται το λάδι στους εναλλάκτες θερμότητας.[12]

΄Υστερα από την αποθήκευση του αέρα στους αγωγούς, αυτός προωθείται στον αεροστρόβιλο (ο οποίος είναι συνδεδεμένος με γεννήτρια για την παραγωγή ενέργειας) για την εκτόνωσή του, που περιλαμβάνει έξι στάδια.[13] Πριν από το πρώτο στάδιο της εκτόνωσης, γίνεται θέρμανση του αέρα μέσω του θερμικού λαδιού που βρίσκεται στα ειδικά δοχεία αποθήκευσής του σε θερμοκρασία 240^{o}C και αυτό έπειτα από την εισαγωγή του σε εναλλάκτη θερμότητας επιστρέφει στο ψυχρό δοχείο αποθήκευσης λαδιού, που βρισκόταν αρχικά πριν ακόμα το στάδιο της συμπίεσης.[13] Με αυτή τη διαδικασία θέρμανσης αέρα επιτυγχάνεται τόσο η αύξηση της απόδοσης του συστήματος, όσο και η αποφυγή θερμοκρασιών στις οποίες μπορεί να σχηματιστεί πάγος. Η διαδικασία αυτή επαναλαμβάνεται και πριν από το τέταρτο και έκτο στάδιο της εκτόνωσης του αέρα. ΄Υστερα από τις πρώτες τρεις εκτονώσεις η θερμοκρασία του αέρα καταλήγει στους 20^{o}C και σε αυτό το σημείο επαναλαμβάνεται η θέρμανση του αέρα για να ακολουθήσουν το τέταρτο και πέμπτο στάδιο της εκτόνωσης όπου πάλι η θερμοκρασία πέφτει στους 20^{o}C.[13] Ο αέρας επαναθερμένεται έπειτα από την ολοκλήρωση αυτών των σταδίων για να πραγματοποιηθεί και το έκτο και τελευταίο στάδιο, πριν όμως από το οποίο ατμός από το ζεστό δοχείο αποθήκευσης νερού αναμιγνύεται με τον αέρα για να επιτευχθεί μεγαλύτερη παραγωγή ενέργειας.[13] Κατά την έξοδο από τον αεροστρόβιλο, το νερό και ο αέρας διαχωρίζονται, όπου το νερό αποθηκεύεται στο ψυχρό δοχείο νερού, ενώ ο αέρας που έχει πλέον θερμοκρασία 65

βαθμών Κελσίου μπορεί να χρησιμοποιηθεί για θέρμανση.[13] Έτσι, λοιπόν, με τη σύνδεση του αεροστρόβιλου με τη γεννήτρια και την παραγωγή ηλεκτρικής ενέργειας για άμεση τροφοδοσία του δικτύου αλλά και δυνατότητα θέρμανσης μέσω του εξερχόμενου αέρα επιτυγχάνεται συμπαραγωγή ηλεκτρισμού – θέρμανσης. Τονίζεται ότι με τη μείωση εισαγωγής ατμού, η θερμοκρασία του αέρα μπορεί να φτάσει και τους 20°C, με αποτέλεσμα να μπορεί να χρησιμοποιηθεί και για ψύξη.[13]



Εικόνα 5: Θερμοδυναμικός κύκλος του αδιαβατικού συστήματος αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα

Πηγή: Energy balance analysis between energy consumed for the compression of atmospheric air up to 200bar and the energy produced from its expansion down to atmospheric pressure on Nicholas Pittas' patent about storage and continuous production energy system (2021)

1.5.3.ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΟΥ ΑΔΙΑΒΑΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΚΑΙ ΛΟΙΠΩΝ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ

Η αποθήκευση ενέργειας στα αντίστοιχα συστήματα αποθήκευσης ποικίλει ανάλογα με το μέσο αποθήκευσης, όμως έχει πάντα τον ίδιο σκοπό, που είναι η παραγωγή ενέργειας όταν υπάρχει μεγάλη ανάγκη και ζήτηση για αυτήν. Αυτά τα συστήματα αποθήκευσης θέτουν όρια και περιορισμούς αναφορικά με τη διάρκεια και

τη χωρητικότητα αποθήκευσης, τη διάρκεια ζωής, την αποδοτικότητα και τα κόστη υλοποίησης και λειτουργίας τους.

Οι μπαταρίες, είτε είναι σε μικρή, είτε σε μεγάλη κλίμακα ισχύος, έχουν υψηλή πυκνότητα ενέργειας, σχετικά μικρούς σε αριθμό κύκλους λειτουργίας και υψηλή τιμή υλοποίησης και λειτουργίας. Ενδεικτικά, για ένα σύστημα αποθήκευσης ενέργειας μπαταριών μεγάλης κλίμακας με χωρητικότητα αποθήκευσης ενέργειας 45,28MWh το κόστος υλοποίησης αγγίζει τα 28 εκατομμύρια ευρώ, ενώ το σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα για τη συμπαραγωγή ηλεκτρισμού και θέρμανσης με αντίστοιχη χωρητικότητα αποθήκευσης ενέργειας έχει κόστος σχεδόν 4 φορές χαμηλότερο και συγκεκριμένα περί τα 6 εκατομμύρια ευρώ.[12] Ως αρνητικός παράγοντας στη χρήση των μπαταριών για την αποθήκευση ενέργειας, προστίθεται η επιβάρυνση του περιβάλλοντος με τοξικές ουσίες, μετά το πέρας της λειτουργίας τους και αν δεν ανακυκλωθούν.

Στα θετικά της αποθήκευσης ενέργειας σε κυψέλες καυσίμου συγκαταλέγονται η υψηλή πυκνότητα ενέργειας χάρη στη χρήση των καυσίμων και κυρίως του υδρογόνου, με αυτήν να μπορεί να φτάσει τα 1100 kWh/m^3, καθώς και η διάρκεια αποθήκευσης που είναι απεριόριστη.[12] Ωστόσο, το κόστος λειτουργίας είναι ασύμφορο οικονομικά, δεδομένου ότι η υλοποίηση της τεχνολογίας αυτής βρίσκεται ακόμα υπό εξέλιξη.

Ο σφόνδυλος ως σύστημα αποθήκευσης ενέργειας παρουσιάζει υψηλή πυκνότητα ενέργειας, αλλά έχει χαμηλή χωρητικότητα αποθήκευσης ενέργειας και υπάρχει δυσκολία και πολυπλοκότητα στην κατασκευή του, με αποτέλεσμα το κόστος της να είναι πολύ υψηλό.

Το σύστημα αποθήκευσης ενέργειας μέσω αντλησιοταμίευσης, αν και έχει μεγάλη διάρκεια ζωής παρουσιάζει χαμηλή πυκνότητα ενέργειας. Επίσης, χρειάζεται ιδιαίτερες γεωμορφολογικές συνθήκες, ενώ στην περίπτωση που κατασκευαστούν τεχνητοί ταμιευτήρες νερού υπάρχει αντίκτυπο στο φυσικό περιβάλλον.

Τα συστήματα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα που χρησιμοποιούν ορυχεία ή υπόγεια σπήλαια ως μέσο για την αποθήκευση του αέρα μπορούν να αντέξουν χαμηλά όρια πίεσης του αποθηκευμένου αέρα, με αυτόν τον παράγοντα να είναι περιοριστικός και η αποδοτικότητά τους είναι χαμηλή λόγω της μη συγκράτησης και απόρριψης της παραγόμενης θερμότητας, σε αντίθεση με το αδιαβατικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα της πρότασης που την ανακτά και επιτυγχάνει απόδοση έως και 80%.[12] Στο τελευταίο δεν γίνεται καύση ορυκτού καυσίμου για την θέρμανση του ατμοσφαιρικού

αέρα στο απαραίτητο στάδιο και τα κατασκευαστικά του υλικά είναι ανακυκλώσιμα μετά το πέρας της διάρκειας ζωής του, για αυτό και είναι 100% «πράσινη λύση». Επιπλέον, σε αυτό η πυκνότητα ενέργειας αγγίζει τις 30 KWh/m^3 αέρα και η διάρκεια ζωής του υπερβαίνει τα 40 έτη και μπορεί να φτάσει έως και 60, με το κόστος της παραγόμενης ενέργειας να δύναται να είναι μικρότερο από 150€/ KWh.[12]

Παρακάτω, παρουσιάζεται η σύγκριση των τεχνολογιών αποθήκευσης ενέργειας, καταδεικνύοντας το αδιαβατικό σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, ως την ιδανικότερη λύση μεταξύ αυτών.

Πίνακας 1: Σύγκριση των τεχνολογιών αποθήκευσης ενέργειας

Επιμέρους Βαθμολογία Καλή Μέτρια Κακή	Κόστη, σε €/kWh	Διάρκεια εκφόρτισης, σε ώρες	Κύκλοι λειτουργίας*10^3, Διάρκεια ζωής σε χρόνια	Πυκνότητα ενέργειας σε kWh/m^3	Αποδοτικότητα σε %	Φιλικότητα προς το περιβάλλον	Πυκνότητα ενέργειας (θετικά ή αρνητικά κρινόμενη)
Μπαταρίες Ιόντων Λιθίου	300…600	0.25…8	0.4…6 15	190…375	90…97	--	+
Κυψέλες καυσίμου	340…420	>12	10…15 20	350	50…80	+	+
Σφόνδυλος	650……2625	0.004…0.25	>1000 -	210	83…93	+	+

Αντλησιοταμίευση	40…180	12	12.8…33 40…100	0.35…1.1	70…82	-	--
Συμβατικό Σύστημα Αποθήκευσης Ενέργειας με Πεπιεσμένο Αέρα	40…80	2…24	8.6…17.1 40	2…7	40…55	-	--
Αδιαβατικό Σύστημα Αποθήκευσης Ενέργειας με Πεπιεσμένο Αέρα	<150	>24	>30 >40	30	70…80	+	+

Πηγή: Renewable Energy Source for Combined Heat and Power (RESCHP) (2021)

2. ΣΥΣΤΗΜΑΤΑ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ

Με την εμφάνιση των πρώτων πολιτισμών στον κόσμο και τα τότε τεχνολογικά επιτεύγματά τους, εμφανίζεται και η έννοια του ελέγχου, σε απλούστερη, συγκριτικά με τα σημερινά δεδομένα, μορφή. Η ύπαρξη των επιτευγμάτων της εποχής εκείνης συνοδευόταν από κάποιας μορφής έλεγχο, που πήγαζε από την επιθυμία αλλά και την ανάγκη του ανθρώπου να μπορεί να τα διαχειρίζεται. Αργότερα, με την εξέλιξη του ανθρώπου και των μηχανών που επινόησε κυρίως κατά την περίοδο της βιομηχανικής επανάστασης, ο έλεγχος σε αυτές αποτέλεσε αναπόσπαστο κομμάτι για την ομαλή λειτουργία τους. Έτσι, στη σημερινή εποχή υπάρχει μια ολόκληρη θεωρία, η θεωρία ελέγχου, η οποία εφαρμόζεται σε μια ευρεία γκάμα συστημάτων και εγκαταστάσεων, είτε αυτά είναι στο πλαίσιο λειτουργίας οικιακών συσκευών (π.χ. συσκευή θέρμανσης – θερμοστάτης), είτε στο πλαίσιο λειτουργίας βιομηχανιών (π.χ. κατεργασίες παραγωγής – τόρνος).

2.1. ΘΕΩΡΙΑ ΕΛΕΓΧΟΥ

Βασική έννοια της θεωρίας ελέγχου αποτελεί το σύστημα ελέγχου, με το οποίο η μεταβλητή εξόδου (από το προς έλεγχο σύστημα ή εγκατάσταση) ή αλλιώς απόκριση ελέγχεται με τη χρήση ενός σήματος ελέγχου, το οποίο λέγεται είσοδος ελέγχου.[14] Η κυριότερη διάκριση των συστημάτων ελέγχου γίνεται βάσει του αν ενέχουν την ιδιότητα της ανάδρασης, δηλαδή αν υπάρχει επανατροφοδότηση της μεταβλητής εξόδου και σύγκρισή της με το σήμα ελέγχου. Εκείνα τα συστήματα που δεν έχουν την ιδιότητα της ανάδρασης καλούνται Συστήματα Ανοιχτού Βρόχου, ενώ εκείνα που την έχουν Συστήματα Κλειστού Βρόχου.

- <u>Συστήματα Ανοιχτού Βρόχου</u>

Κατά τη λειτουργία ενός τέτοιου συστήματος υπάρχει ένας κατευθυντής, με σκοπό να επιτευχθεί το επιθυμητό αποτέλεσμα, η είσοδος του οποίου λέγεται είσοδος αναφοράς και η έξοδός του, που εισάγεται στο προς έλεγχο σύστημα και επηρεάζει την ελεγχόμενη μεταβλητή, ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα, λέγεται σήμα ελέγχου. Παράλληλα, μπορεί να υπάρχει και μια διαταραχή, η οποία συνήθως έχει αρνητική επίδραση στην ελεγχόμενη μεταβλητή και στο κατά πόσο αυτή θα ακολουθήσει την επιθυμητή συμπεριφορά. Σε αυτά τα συστήματα ελέγχου, η διορθωτική δράση καθορίζεται από την τιμή της εισόδου αναφοράς και όχι από αυτήν της ελεγχόμενης μεταβλητής.[14]

- Συστήματα Κλειστού Βρόχου

Κατά τη λειτουργία ενός τέτοιου συστήματος η ελεγχόμενη μεταβλητή συγκρίνεται με την είσοδο αναφοράς και έτσι προκύπτει το σφάλμα, που είναι η διαφορά των τιμών τους. Ύστερα από τη διαδικασία της ανάδρασης της ελεγχόμενης μεταβλητής και την εξαγωγή του σφάλματος, αυτό προωθείται στον κατευθυντή για τη διόρθωσή του. Σε αυτά τα συστήματα, η διορθωτική δράση καθορίζεται, τόσο από την ελεγχόμενη μεταβλητή, όσο και από την είσοδο αναφοράς.[14]

Τα συστήματα ελέγχου μπορεί να είναι χρονικά αμετάβλητα ή μεταβλητά, στατικά ή δυναμικά και γραμμικά ή μη γραμμικά. Όταν οι σχέσεις εισόδου – εξόδου δε μεταβάλλονται με την πάροδο του χρόνου, το σύστημα καλείται χρονικά αμετάβλητο, ενώ όταν η μορφή της εξόδου (απόκριση) δεν εξαρτάται μόνο από τη μορφή της εισόδου αλλά και από το χρόνο εφαρμογής της καλείται χρονικά μεταβλητό.[14] Όταν οι τιμές της εξόδου σε κάθε χρονική στιγμή εξαρτώνται από τις τιμές της εισόδου στην ίδια χρονική στιγμή, το σύστημα καλείται στατικό, ενώ όταν αυτές εξαρτώνται από προηγούμενες τιμές εισόδου, καλείται δυναμικό.[14] Όταν για την περιγραφή του συστήματος χρησιμοποιούνται γραμμικές διαφορικές εξισώσεις, το σύστημα καλείται γραμμικό και μη γραμμικό σε αντίθετη περίπτωση.[14]

Σύμφωνα με την κλασική θεωρία ελέγχου, η ανάλυση ενός συστήματος ελέγχου απαιτεί την ύπαρξη ενός μαθηματικού μοντέλου. Αυτό επιτυγχάνεται με την επίλυση διαφορικών εξισώσεων, οι οποίες όμως είναι συνήθως μη γραμμικές και για αυτό αρχικά αγνοούνται ορισμένες παράμετροι που αφορούν στη φύση του συστήματος και εμφανίζουν μη γραμμικότητες, ώστε να προκύψει ένα μαθηματικό μοντέλο με γραμμικές διαφορικές εξισώσεις και στη συνέχεια μέσω του μετασχηματισμού Laplace να υλοποιηθεί η λύση αυτών, η οποία θα προσδιορίσει και τη συμπεριφορά του συστήματος. Ωστόσο, αυτή εφαρμόζεται μόνο σε γραμμικά χρονικά αμετάβλητα συστήματα που έχουν μια είσοδο και μια έξοδο. Στην περίπτωση πολυμεταβλητών συστημάτων, δηλαδή όταν ελέγχονται συγχρόνως περισσότερες από μια μεταβλητές, και υπάρχουν πολλές έξοδοι ως αποτέλεσμα των αντίστοιχα πολλών εισόδων και τα οποία μπορεί να είναι γραμμικά ή μη γραμμικά αλλά και χρονικά αμετάβλητα ή μεταβλητά, χρησιμοποιείται η σύγχρονη θεωρία ελέγχου. Για τον προσδιορισμό της συμπεριφοράς του συστήματος ελέγχου χρησιμοποιείται η ίδια διαδικασία με την κλασική θεωρία, δηλαδή η κατάστρωση ενός μαθηματικού μοντέλου και η επίλυση του, πλέον όμως υπάρχουν οι έννοιες της κατάστασης, των μεταβλητών κατάστασης, των διανυσμάτων κατάστασης και του χώρου κατάστασης. Η μορφή των εξισώσεων, βάση των οποίων περιγράφεται το σύστημα με τη σύγχρονη θεωρία

ελέγχου είναι διαφορικές εξισώσεις πρώτης τάξης τέτοιου αριθμού όσες και οι μεταβλητές κατάστασης.[14] Όταν οι διαφορικές εξισώσεις πρώτης τάξης που περιγράφουν το σύστημα είναι γραμμικές και οι συντελεστές των μεταβλητών σταθεροί το σύστημα είναι γραμμικό, χρονικά αμετάβλητο.[14] Όταν μερικοί από τους συντελεστές είναι συναρτήσεις του χρόνου το σύστημα είναι γραμμικό, χρονικά μεταβλητό.[14]

Η επίλυση των διαφορικών εξισώσεων και η εύρεση ενός αποτελέσματος προσδιορίζει και δίνει στοιχεία για τη συμπεριφορά του συστήματος ελέγχου, προκειμένου να υπάρχει γνώση ότι επιτελούν σωστά και με αποδοτικό τρόπο το έργο τους. Έτσι, λοιπόν, πρέπει να διαθέτουν ορισμένα χαρακτηριστικά, όπως να παρουσιάζουν ευστάθεια, δηλαδή η έξοδος να κυμαίνεται σε ορισμένα επιθυμητά όρια, σχετική ευστάθεια, δηλαδή η απόκριση να έχει μια λογική ταχύτητα και απόσβεση, καθώς και ακρίβεια, δηλαδή να μην υπάρχουν ή να υπάρχουν στον ελάχιστο βαθμό σφάλματα.[14]

2.2. ΚΕΝΤΡΑ ΕΛΕΓΧΟΥ ΕΝΕΡΓΕΙΑΣ

Τα κέντρα ελέγχου ενέργειας αποσκοπούν στον έλεγχο της ομαλής και ασφαλούς λειτουργίας ενός ενεργειακού συστήματος σε πραγματικό χρόνο. Τέτοια ενεργειακά συστήματα μπορούν να είναι μια μονάδα παραγωγής ηλεκτρικής ενέργειας, τόσο με συμβατικούς τρόπους, όσο και μέσω ανανεώσιμων πηγών ενέργειας. Τα κέντρα ελέγχου ενέργειας βασίζονται κυρίως σε συστήματα που συλλέγουν δεδομένα από το προς έλεγχο σύστημα, εποπτεύουν και ελέγχουν τη λειτουργία του, γνωστά και ως συστήματα SCADA (Supervision, Control and Data Acquisition). Τα λειτουργικά μέρη ενός τέτοιου συστήματος αποτελούνται από το ελεγχόμενο σύστημα, τις απομακρυσμένες τερματικές μονάδες ή απομακρυσμένες μονάδες τηλεμέτρησης (RTU – Remote Telemetry Unit), τις γραμμές επικοινωνίας των μονάδων αυτών με τους κεντρικούς υπολογιστές και το σύστημα των κεντρικών υπολογιστών που συνοδεύεται από οθόνες και πίνακες ελέγχου τοποθετημένα στο κέντρο ελέγχου ενέργειας.[15]

Η λειτουργία του συστήματος SCADA προϋποθέτει την τοποθέτηση αισθητήρων και λοιπών οργάνων μέτρησης φυσικών μεγεθών σε σημεία του προς έλεγχο συστήματος. Ύστερα από τη συλλογή των δεδομένων, αυτά μεταφέρονται μέσω των τερματικών μονάδων/μονάδων τηλεμετρίας στο κέντρο ελέγχου. Αυτές οι μονάδες βρίσκονται σε απομακρυσμένα σημεία με σκοπό την αποστολή και τη λήψη δεδομένων και εντολών.[15] Ουσιαστικά, μετατρέπουν τις τιμές φυσικών μεγεθών που

λαμβάνουν σε σήματα που να μπορούν να αποσταλούν ενσύρματα ή ασύρματα, καθώς και σήματα από μια άλλη αντίστοιχη μονάδα ή ηλεκτρονικό υπολογιστή του κέντρου ελέγχου σε σήματα εξόδου, ώστε να καθοριστεί κάποια ενέργεια (π.χ. κλείσιμο βαλβίδας).[15] Για να υλοποιηθούν όλες αυτές οι μεταφορές, συνδέσεις και επικοινωνίες μεταξύ των τερματικών μονάδων και των υπολογιστών χρειάζονται γραμμές επικοινωνίας που μπορούν να είναι καλωδιακές, τηλεφωνικές και ασύρματες.[8] Οι τερματικές μονάδες διαθέτουν ενσωματωμένο διαμορφωτή – αποδιαμορφωτή σήματος (modem) και έτσι συνδέονται με τις γραμμές επικοινωνίας αλλά και επικοινωνούν μεταξύ τους, χρησιμοποιώντας πρωτόκολλα επικοινωνίας, όπως το ASCII (American Standard Code for Information Interchange), ενώ με τους αισθητήρες μέσω κυκλωμάτων εισόδου – εξόδου.[15] Στον υπολογιστικό σταθμό του κέντρου ελέγχου υπάρχει συστοιχία από ηλεκτρονικούς υπολογιστές με κατάλληλες διασυνδέσεις, ώστε να υπάρχει εφεδρεία σε περίπτωση βλάβης κάποιου, και συνήθως μέσω ενός πρωτεύοντος υπολογιστή δίνονται εντολές για τις απαραίτητες ενέργειες αφού πρώτα τα συλλεχθέντα δεδομένα έχουν αποτυπωθεί στις οθόνες του κέντρου.

Έτσι, λοιπόν, γίνεται συγκέντρωση των απαραίτητων πληροφοριών, μεταφορά τους στον υπολογιστικό σταθμό του κέντρου ελέγχου, απεικόνιση των δεδομένων και εκτίμηση της κατάστασης μέσω ειδικών προγραμμάτων που φέρουν οι υπολογιστές και έπειτα δρομολογούνται μέσω αντίστοιχων εντολών ο προληπτικός έλεγχος σε περίπτωση κανονικής λειτουργίας του συστήματος, ο έλεγχος ανάγκης σε περίπτωση κατάστασης ανάγκης και ο έλεγχος επαναφοράς όταν αυτό κριθεί απαραίτητο.

3. ΥΛΟΠΟΙΗΣΗ ΕΠΟΠΤΕΙΑΣ ΤΟΥ ΑΔΙΑΒΑΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΕΝΕΡΓΕΙΑΣ ΜΕΣΩ ΠΕΠΙΕΣΜΕΝΟΥ ΑΤΜΟΣΦΑΙΡΙΚΟΥ ΑΕΡΑ

Όπως σε κάθε σύστημα απαιτείται έλεγχος στα διάφορα στάδια λειτουργίας του, ώστε αυτή να είναι ασφαλής, έτσι οφείλει να γίνει και στο εν λόγω σύστημα. Λόγω των ιδιαίτερων συνθηκών που επικρατούν κατά τη λειτουργία του και οφείλονται στις μεταβολές, όπως αυτές παρουσιάστηκαν στον θερμοδυναμικό κύκλο του συστήματος παραπάνω, αυτό χρήζει συνεχούς εποπτείας, προκειμένου οι τιμές συγκεκριμένων μεγεθών, όπως η θερμοκρασία, η πίεση και η υγρασία να μην υπερβαίνουν κάποια όρια και να συνεχίζεται η απρόσκοπτη και ασφαλής λειτουργία του, χωρίς να τίθεται θέμα κατάστασης ανάγκης.

Στο παρόν σημείο, αποφασίστηκε να παρακολουθούνται και να ελέγχονται η θερμοκρασία και η υγρασία κατά τα στάδια της συμπίεσης και εκτόνωσης του αέρα, αλλά και κατά το στάδιο που αυτός βρίσκεται αποθηκευμένος, δηλαδή οι συνθήκες αυτές μέσα στην αεροδεξαμενή. Η παρακολούθηση και ο έλεγχος των τιμών αυτών και κατ' επέκταση του συστήματος στα προαναφερθέντα στάδια, πρόκειται να επιτευχθεί με την τοποθέτηση ειδικών αισθητήρων στον συμπιεστή, στην αεροδεξαμενή και στον αεροστρόβιλο. Τα δεδομένα των μετρήσεων από τους αισθητήρες, πρόκειται να συγκεντρωθούν και να παρουσιαστούν στην οθόνη ελέγχου, με τη χρήση εφαρμογής λογισμικού, ενώ παράλληλα μέσω σύγκρισης και ελέγχου των τιμών στα μεγέθη αυτά με τη βοήθεια του λογισμικού, πρόκειται να αποφασιστεί κάποια ενέργεια.

3.1. ΑΠΑΙΤΟΥΜΕΝΟ ΥΛΙΣΜΙΚΟ ΚΑΙ ΛΟΓΙΣΜΙΚΟ

Όπως επισημάνθηκε παραπάνω, κατά την εποπτεία του συστήματος είναι αναγκαία τρία στάδια: η εγκατάσταση συγκεκριμένων αισθητήρων, η σαφής αποτύπωση και παρουσίαση των δεδομένων έπειτα από τη συλλογή τους και ο έλεγχος σε αυτά, προκειμένου να υλοποιείται κάποια ενέργεια. Αναφορικά με τις μετρήσεις της θερμοκρασίας και της υγρασίας στα σημεία που αναφέρθηκαν προηγουμένως, αυτές κρίνονται αναγκαίες, τόσο για να υπάρχει γνώση των τιμών των μεγεθών αυτών και να βεβαιώνεται ότι όλα κυλούν ομαλά και με τον επιθυμητό τρόπο στο σύστημα, όσο και σε αντίθετη περίπτωση να είναι εφικτό να γίνει οποιαδήποτε παρέμβαση σε αυτό. Το μέσο για να γίνουν αυτές οι μετρήσεις είναι οι αισθητήρες, ενώ για να γίνει σαφής αποτύπωση και παρουσίαση των δεδομένων από αυτές στην οθόνη της εφαρμογής χρειάζεται συνεχής επικοινωνία με εκείνη, που επιτυγχάνεται με τη χρήση πλακέτας μικροηλεκτρονικών και αντίστοιχου λογισμικού. Τα τελευταία

χρησιμοποιούνται και κατά το στάδιο ελέγχου των τιμών των μετρήσεων, για να αποφασιστεί κάποια ενέργεια παρέμβασης στο σύστημα. Παρακάτω αναφέρονται ποιες είναι οι απαιτήσεις σε υλισμικό και εξαρτήματα, καθώς και σε λογισμικό για την υλοποίηση της εποπτείας.

3.1.1. ΥΛΙΣΜΙΚΟ

- ΠΛΑΚΕΤΑ ARDUINO UNO

Πρόκειται για μια πλακέτα μικροηλεκτρονικών με ενσωματωμένο μικροελεγκτή και κύκλωμα εισόδων/εξόδων που διευκολύνουν το χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Σε αυτή υπάρχουν 14 ψηφιακοί είσοδοι/έξοδοι και 6 είσοδοι αναλογικού σήματος, με την τάση λειτουργίας να είναι στα 5V και την τάση εισόδου να κυμαίνεται από 7 έως 20V. Μπορεί να τροφοδοτηθεί μέσω καλωδίου ενιαίου σειριακού διαύλου με το οποίο παράλληλα συνδέεται και με το αντίστοιχο λογισμικό (Arduino IDE), εξωτερικής μπαταρίας, καλωδίου με έξοδο τύπου Jack ή μέσω καλωδίου συνδεδεμένου στην είσοδο τάσης.[16]

- ΑΙΣΘΗΤΗΡΑΣ DHT11

Πρόκειται για έναν αισθητήρα που μετράει τη σχετική υγρασία και τη θερμοκρασία του αέρα σε % και ^{O}C αντίστοιχα. Μερικά από τα χαρακτηριστικά του σχετικά με τις τιμές που αποδίδει (D-Robotics 2010) παρατίθενται παρακάτω[17]:

  - Εύρος τιμών υγρασίας: 20%-90%
  - Ακρίβεια τιμών υγρασίας: ±5%
  - Εύρος τιμών θερμοκρασίας: 0-50^{O}C
  - Ακρίβεια τιμών θερμοκρασίας: ±2^{O}C

- ΚΑΛΩΔΙΑ ΜΕΤΑΓΩΓΗΣ ΤΥΠΟΥ ΑΡΣΕΝΙΚΟ ΣΕ ΘΥΛΗΚΟ ΚΑΙ ΑΡΣΕΝΙΚΟ ΣΕ ΑΡΣΕΝΙΚΟ
- ΚΑΛΩΔΙΟ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ ΤΥΠΟΥ ΕΝΙΑΙΟΥ ΣΕΙΡΙΑΚΟΥ ΔΙΑΥΛΟΥ A/B
- ΠΕΙΡΑΜΑΤΙΚΗ ΠΛΑΚΕΤΑ
- ΛΑΜΠΑΚΙΑ ΤΥΠΟΥ LED
- ΑΝΤΙΣΤΑΣΕΙΣ 220Ω
- ΠΛΑΚΕΤΑ ESP8266 NodeMCU

Πρόκειται για μια πλακέτα μικροηλεκτρονικών, η οποία επιτρέπει τη σύνδεση στο διαδίκτυο για τη μεταφορά δεδομένων σε πραγματικό χρόνο. Για αυτό το σκοπό,

μπορεί να χρησιμοποιηθεί, είτε αυτόνομα, είτε συνδεόμενη κατάλληλα με άλλες πλακέτες κάνοντας τον διαμεσολαβητή, χρησιμοποιώντας και στις δύο περιπτώσεις την ασύρματη τεχνολογία δικτύωσης (Wi-Fi). Χρησιμοποιήθηκε για την υλοποίηση των εναλλακτικών λύσεων.

3.1.2. ΛΟΓΙΣΜΙΚΟ

- ARDUINO IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

Πρόκειται για το λογισμικό προγραμματισμού των ομώνυμων πλακετών. Ουσιαστικά, το υλισμικό μέσω του λογισμικού αυτού προγραμματίζεται για να εκτελέσει κάποια λειτουργία. Στο περιβάλλον εργασίας του, μπορεί να συνταχθεί και να εκτελεστεί ο κώδικας σε γλώσσα προγραμματισμού C ή C++ για τη λειτουργία της πλακέτας, συμπεριλαμβάνοντας τις κατάλληλες βιβλιοθήκες.

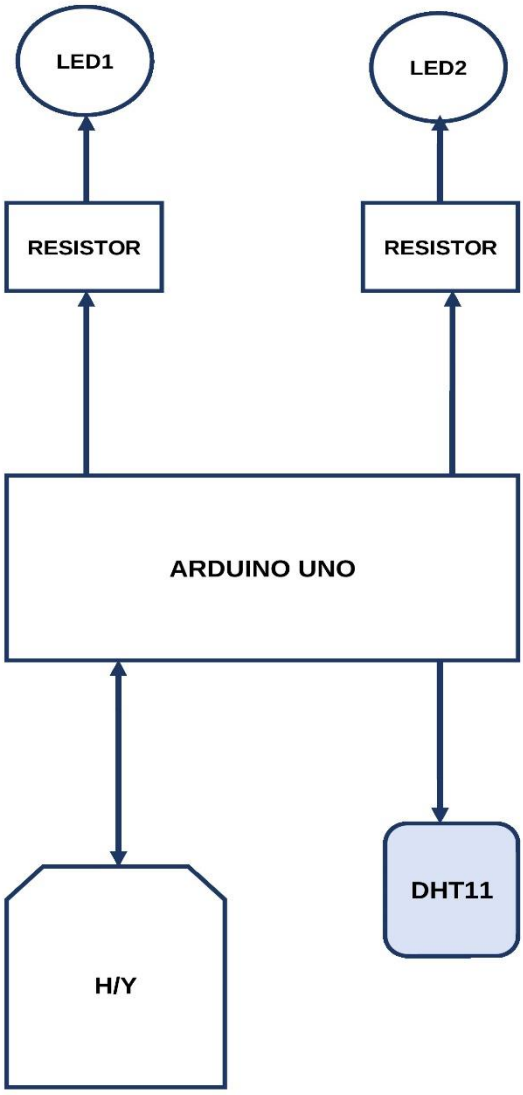
- VISUINO

Πρόκειται για την εφαρμογή οπτικοποίησης της λειτουργίας των πλακετών μικροηλεκτρονικών. Λειτουργεί με την εναπόθεση στο περιβάλλον εργασίας της των κατάλληλων στοιχείων από μια αντίστοιχη λίστα και τη σύνδεση μεταξύ τους, ώστε να δημιουργηθεί ψηφιακά το πραγματικό κύκλωμα. Με την εναπόθεση και σύνδεση των στοιχείων που εμπίπτουν στο πραγματικό κύκλωμα, δημιουργείται αυτόματα ο κώδικας που το περιγράφει στο παρασκήνιο και εκτελείται στη συνέχεια μέσω του Arduino IDE.

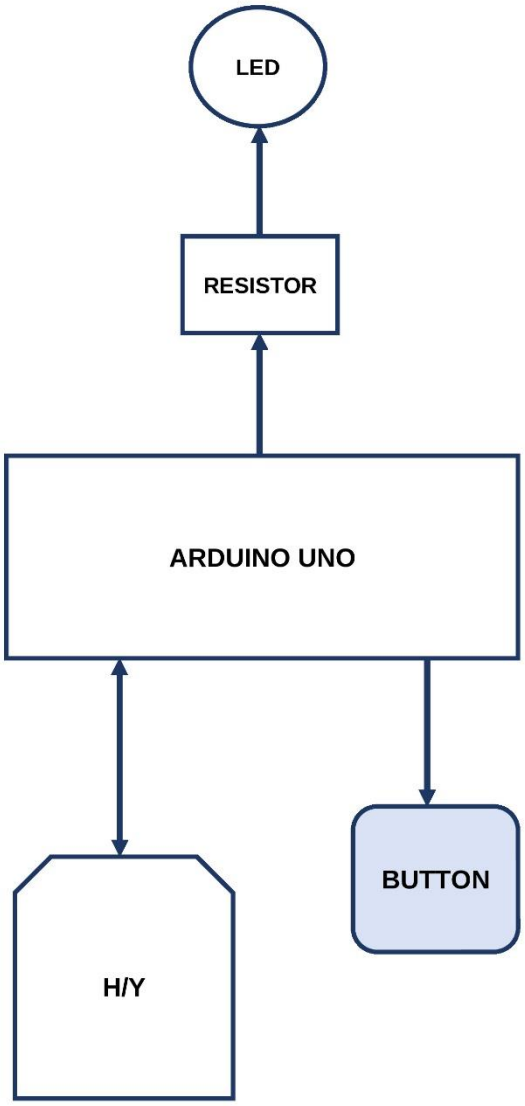
3.2. ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΥΚΛΩΜΑΤΩΝ

Λόγω μη επαρκών εισόδων γείωσης στην πλακέτα μικροηλεκτρονικών (διαθέτει τρεις σε αριθμό), δεν μπορεί να δημιουργηθεί ένα ενιαίο κύκλωμα που θα περιλαμβάνει τον αισθητήρα και τρεις λαμπτήρες, καθώς απαιτούνται μια είσοδος γείωσης της πλακέτας για τη σύνδεση με την ακίδα της γείωσης του αισθητήρα, μια είσοδος γείωσης της πλακέτας για τη σύνδεση με την κάθοδο του πρώτου λαμπτήρα, μια είσοδος γείωσης της πλακέτας για τη σύνδεση με την κάθοδο του δεύτερου λαμπτήρα και μια είσοδος γείωσης της πλακέτας για τη σύνδεση με την κάθοδο του τρίτου λαμπτήρα, δηλαδή 4 σε αριθμό που υπερβαίνουν τις 3 διαθέσιμες εισόδους γείωσης από την πλευρά της πλακέτας. Για το λόγο αυτό, το ενιαίο κύκλωμα διαχωρίζεται σε δυο επιμέρους κυκλώματα, το πρωτεύον κύκλωμα για τη μέτρηση των τιμών των μεγεθών της υγρασίας και της θερμοκρασίας, καθώς και για το συγκριτικό

έλεγχο στις τιμές της τελευταίας και το δευτερεύον κύκλωμα για την παρέμβαση στο σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα με τη χρήση ενός κουμπιού, όταν η θερμοκρασία υπερβεί το επιθυμητό όριο.

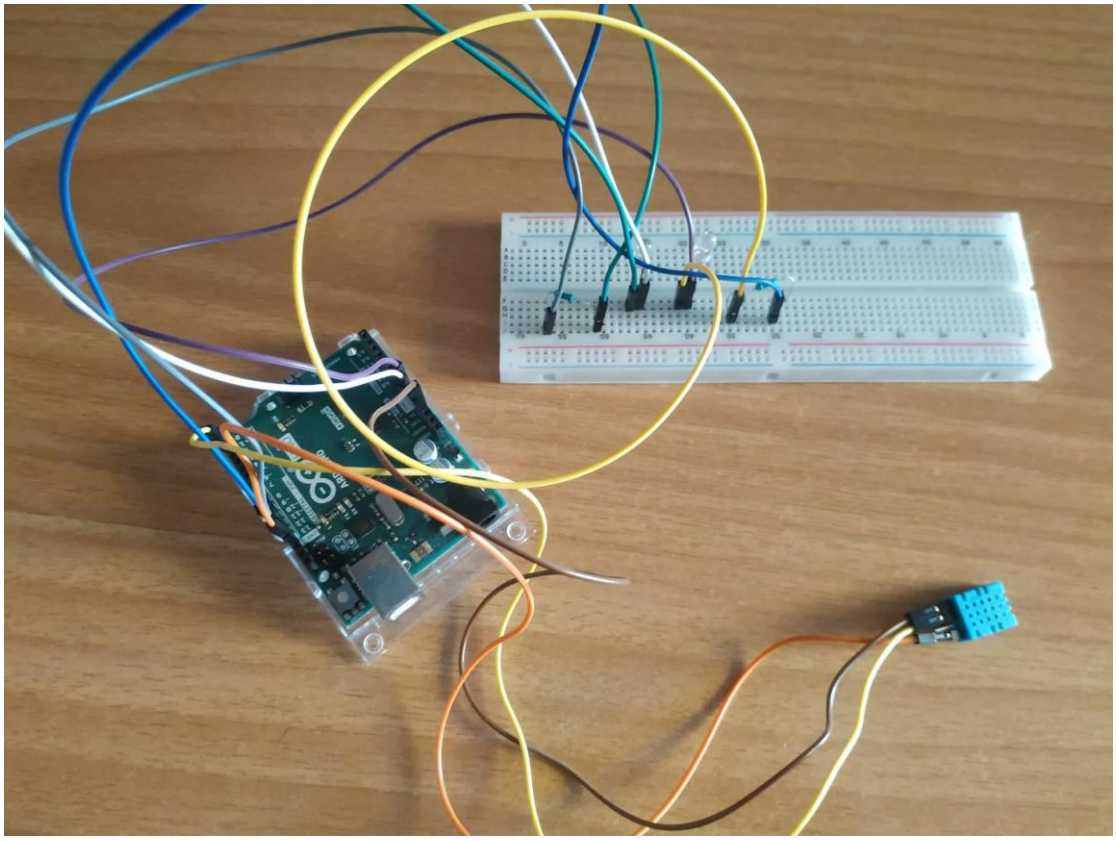


Σχήμα 1: Συνδεσμολογία πρωτεύοντος κυκλώματος

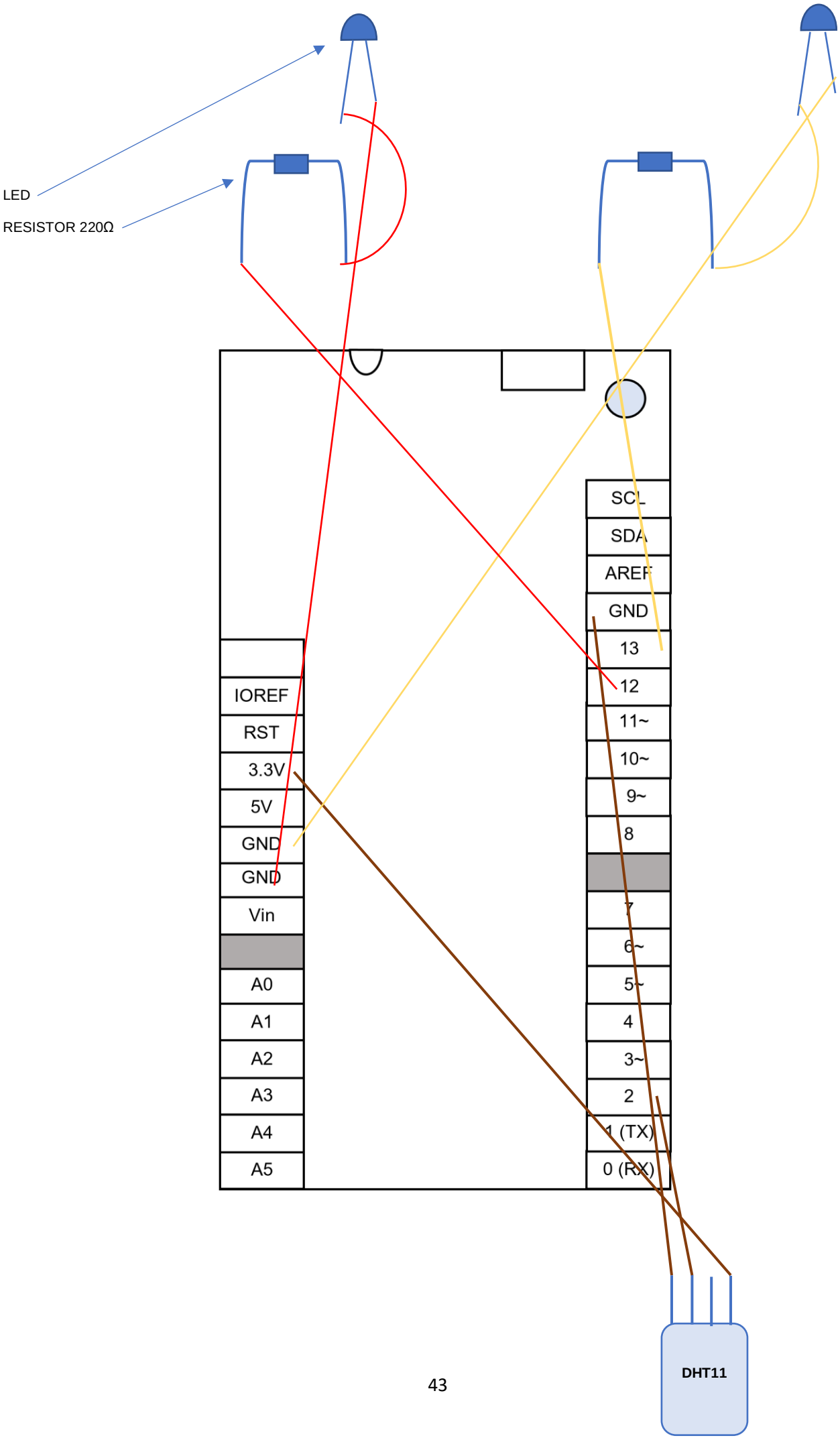


Σχήμα 2: Συνδεσμολογία δευτερεύοντος κυκλώματος

Το πρωτεύον κύκλωμα χρησιμοποιείται για τη μέτρηση των τιμών των μεγεθών της υγρασίας και της θερμοκρασίας, καθώς και για το συγκριτικό έλεγχο των τιμών της θερμοκρασίας, με ένδειξη του αντίστοιχου λαμπτήρα. Για την επίτευξη της συνδεσμολογίας του πρωτεύοντος πραγματικού κυκλώματος, πρέπει τα στοιχεία αυτού, δηλαδή η πλακέτα μικροηλεκτρονικών, ο αισθητήρας υγρασίας και θερμοκρασίας, οι λαμπτήρες LED, οι αντιστάσεις και η πειραματική πλακέτα (breadboard) να ενωθούν κατάλληλα μεταξύ τους με τα καλώδια μεταγωγής (τύπου αρσενικό σε θηλυκό και αρσενικό σε αρσενικό) και έπειτα με το καλώδιο ενιαίου σειριακού διαύλου αυτό να συνδεθεί με τον ηλεκτρονικό υπολογιστή, ώστε να τροφοδοτηθεί με ρεύμα. Αρχικά, για τη σύνδεση της πλακέτας με τον αισθητήρα, συνδέεται η δεύτερη ψηφιακή έξοδος/είσοδος της πλακέτας μέσω καλωδίου αρσενικό σε θηλυκό με την ακίδα εξόδου των δεδομένων του αισθητήρα, η είσοδος της γείωσης της πλακέτας μέσω καλωδίου αρσενικό σε θηλυκό με την ακίδα της γείωσης του αισθητήρα και η έξοδος της τάσης ρεύματος της πλακέτας και πιο συγκεκριμένα εκείνη των 3,3V, μέσω καλωδίου αρσενικό σε θηλυκό με την ακίδα της εισερχόμενης τάσης του αισθητήρα. Σημειώνεται ότι στον αισθητήρα, κρατώντας τον με τις ακίδες προς τα κάτω και φορά από τα αριστερά προς τα δεξιά, η πρώτη είναι αυτή της τάσης, η δεύτερη αυτή της εξόδου των δεδομένων, η τρίτη δε χρησιμοποιείται και η τέταρτη αυτή της γείωσης. Στη συνέχεια, για τη σύνδεση της πλακέτας με τους λαμπτήρες, χρησιμοποιείται η πειραματική πλακέτα, πάνω στην οποία τοποθετούνται εκείνοι, καθώς και οι αντιστάσεις. Η κάθοδος του λαμπτήρα (το πιο κοντό σύρμα) συνδέεται στην εσοχή της γείωσης της πλακέτας μέσω καλωδίου αρσενικό με αρσενικό, ενώ η άνοδος του (το πιο μακρύ σύρμα) συνδέεται με κάποια ψηφιακή έξοδο της πλακέτας μέσω καλωδίου αρσενικό σε αρσενικό, παρεμβάλλοντας μια αντίσταση. Σημειώνεται ότι η αντίσταση που χρησιμοποιείται είναι 220Ω, για περιορισμό του ρεύματος που μπορεί να διαρρεύσει το κύκλωμα και να αποτραπεί το ενδεχόμενο εκείνος να καεί, καθώς και ότι η διαδικασία σύνδεσης που αναφέρθηκε παραπάνω είναι η ίδια και για τους δυο λαμπτήρες με τη ψηφιακή έξοδο της πλακέτας για τον έναν να είναι η δωδέκατη και για τον άλλον η δέκατη τρίτη.



Εικόνα 6: Συνδεσμολογία πρωτεύοντος πραγματικού κυκλώματος


LED
RESISTOR 220Ω
SCL
SDA
AREF
GND
13
12
11~
10~
9~
8
7
6~
5~
4
3~
2
1 (TX)
0 (RX)
IOREF
RST
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
DHT11

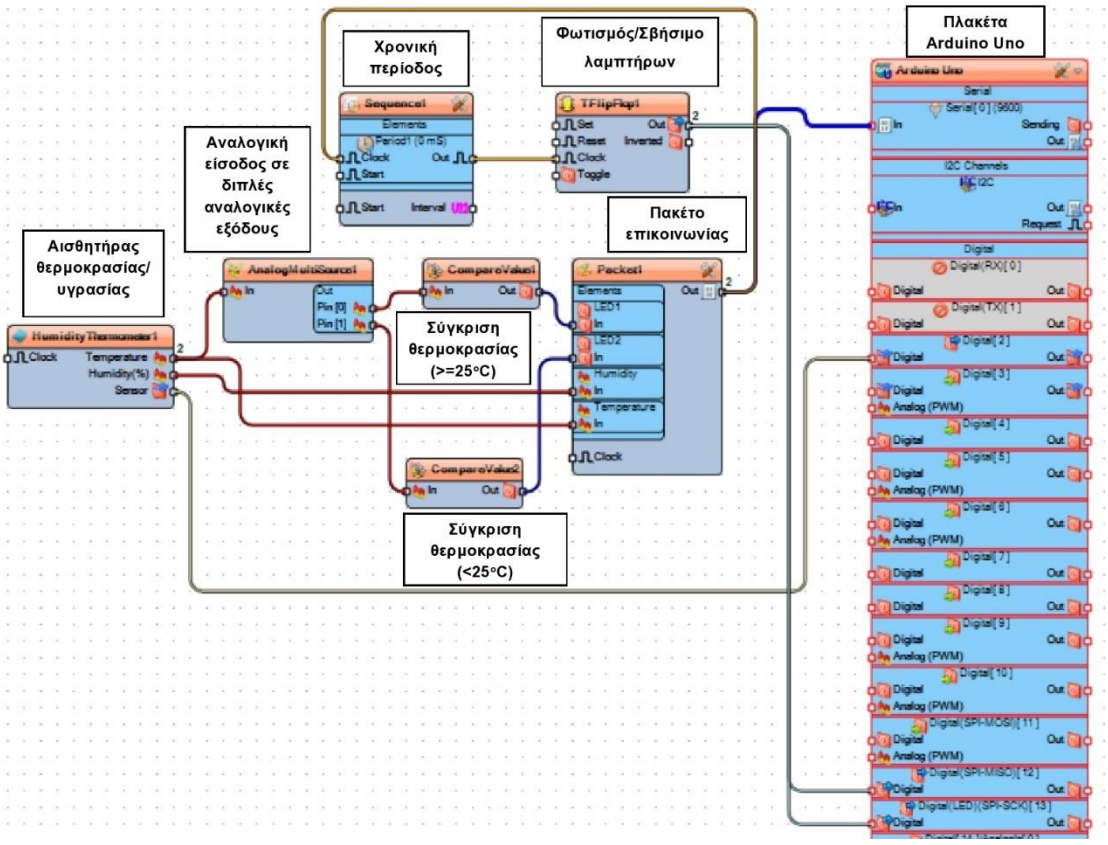
Σχήμα 3: Συνδεσμολογία πρωτεύοντος πραγματικού κυκλώματος

Για την αποτύπωση του πρωτεύοντος πραγματικού κυκλώματος με ψηφιακό τρόπο χρησιμοποιείται η εφαρμογή Visuino. Ανοίγοντας την εφαρμογή, φαίνεται ήδη στο περιβάλλον εργασίας, οπτικοποιημένη η πλακέτα, μέσω ενός μπλε σχήματος γραφικών που την αντιπροσωπεύει, έχοντας όλες τις εισόδους/εξόδους, τόσο τις ψηφιακές και αναλογικές, όσο και εκείνες της τάσης και της γείωσης, καθώς και οτιδήποτε άλλο υπάρχει και στην πραγματική πλακέτα. Πατώντας στις ρυθμίσεις η πλακέτα μπορεί να αλλάξει ανάλογα με τον τύπο της πραγματικής και να εμφανιστούν αντίστοιχα όλα τα πεδία και στοιχεία που περιέχονται σε εκείνην. Στη συνέχεια, δεξιά εμφανίζονται τα στοιχεία που είναι διαθέσιμα προς επιλογή και εναπόθεση στο περιβάλλον εργασίας και μια μπάρα αναζήτησης των στοιχείων αυτών, ενώ αριστερά εμφανίζονται το περιβάλλον εργασίας σε σμίκρυνση και το πεδίο των ιδιοτήτων των στοιχείων, που πρόκειται να εισαχθούν. Κάτω από το χώρο εργασίας βρίσκεται το πεδίο των οπτικοποιήσεων που θα προκύψουν από την εισαγωγή των κατάλληλων στοιχείων και τη λειτουργία του συστήματος.

Αρχικά, επιλέγεται η πλακέτα Arduino Uno και στη συνέχεια από το πεδίο των στοιχείων ο αισθητήρας υγρασίας και θερμοκρασίας. Από τις εξόδους του στοιχείου του αισθητήρα, επιλέγεται η τελευταία στη σειρά που φέρει την ένδειξη αισθητήρας («sensor») και συνδέεται με την δεύτερη ψηφιακή είσοδο της πλακέτας, σύροντας με το ποντίκι του ηλεκτρονικού υπολογιστή από τη μια κατεύθυνση προς την άλλη. Στη συνέχεια, επιλέγεται το στοιχείο του πακέτου επικοινωνίας, ώστε οι μετρήσεις του αισθητήρα να μπορούν μέσω αυτού να επικοινωνήσουν με την πλακέτα, με τέτοιο τρόπο που αυτή μπορεί να αντιληφθεί. Στις ιδιότητες του στοιχείου του πακέτου επικοινωνίας, προσθέτονται τα αναλογικά στοιχεία της υγρασίας και της θερμοκρασίας, τα οποία ενώνονται με τις αντίστοιχες εξόδους του στοιχείου του αισθητήρα, προκειμένου να επιτευχθεί η επικοινωνία που προαναφέρθηκε. Επίσης από το πεδίο των ιδιοτήτων, επιλέγεται το κοντέρ και το θερμόμετρο ως οπτικοποιήσεις των τιμών της υγρασίας και της θερμοκρασίας αντίστοιχα, με αρχική τιμή το 0, τελική το 100 και βήμα 10, ενώ καθορίζονται και τα bytes για να διευκρινιστεί το μέγεθος της χωρητικότητας της πληροφορίας. Τέλος, ενώνεται η έξοδος του στοιχείου του πακέτου επικοινωνίας με τη σειριακή θύρα εισόδου της πλακέτας.

Για την οπτικοποίηση των λαμπτήρων, προσθέτονται πάλι από τις ιδιότητες του στοιχείου του πακέτου επικοινωνίας τα ψηφιακά στοιχεία των LED, δύο σε αριθμό, τα οποία θα ανάβουν πότε το ένα και πότε το άλλο, ανάλογα με τις τιμές της θερμοκρασίας. Συγκεκριμένα, η ένδειξη του πρώτου λαμπτήρα θα ανάβει όταν η

θερμοκρασία υπερβαίνει την τιμή των 25°C, ενώ όταν αυτή είναι ίση ή μικρότερη από 25°C θα ανάβει η ένδειξη του δεύτερου λαμπτήρα. Για να επιτευχθεί αυτό, εισάγεται το στοιχείο των αναλογικών πολλαπλών πηγών, η είσοδος του οποίου ενώνεται με την έξοδο της θερμοκρασίας του αισθητήρα και οι δύο έξοδοι του σε δυο ξεχωριστά στοιχεία σύγκρισης τιμών. Η πρώτη έξοδος του συνδέεται με το πρώτο στοιχείο σύγκρισης τιμών και η δεύτερη με το δεύτερο. Μέσα από τις ιδιότητες των στοιχείων της σύγκρισης τιμών είναι το σημείο όπου θέτουμε τη τιμή και τα κριτήρια για τα οποία θα ελεγχθεί η θερμοκρασία και για την ακρίβεια είναι αυτά που αναφέρθηκαν προηγουμένως. Οι έξοδοι των στοιχείων σύγκρισης τιμών ενώνονται με τις εισόδους των ψηφιακών στοιχείων των λαμπτήρων στο πακέτο επικοινωνίας, με αυτή του πρώτου στοιχείου να ενώνεται με το ψηφιακό στοιχείο LED1 και του δεύτερου με το LED2, ώστε να ανάψουν και οι αντίστοιχες ενδείξεις.



Εικόνα 7: Συνδεσμολογία πρωτεύοντος κυκλώματος στην εφαρμογή Visuino, με εμφανή όλα τα εμπλεκόμενα στοιχεία

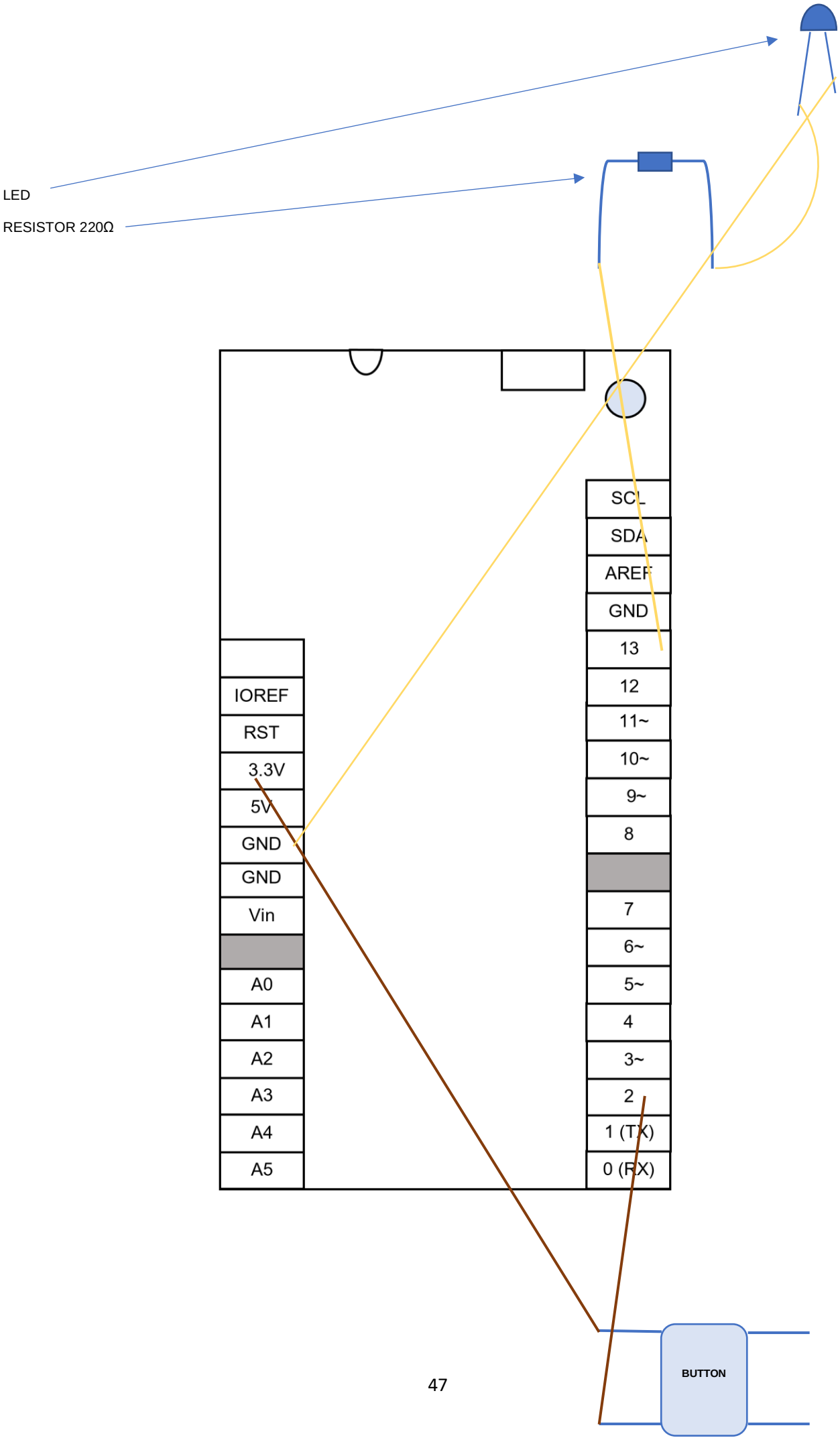
Πηγή: Στιγμιότυπο από εφαρμογή Visuino

Το δευτερεύον κύκλωμα χρησιμοποιείται για τον έλεγχο και την υλοποίηση συγκεκριμένων ενεργειών, με την προϋπόθεση ότι η τιμή της θερμοκρασίας έχει

υπερβεί το επιθυμητό όριο. Με άλλα λόγια, θα υπάρχει κάποια παρεμβατική ενέργεια στο σύστημα αποθήκευσης ενέργειας μέσω πεπιεσμένου ατμοσφαιρικού αέρα, όπως άνοιγμα ή κλείσιμο βαλβίδων της αεροδεξαμενής, όταν υπάρχει υπέρβαση του επιθυμητού ορίου στην τιμή της θερμοκρασίας.

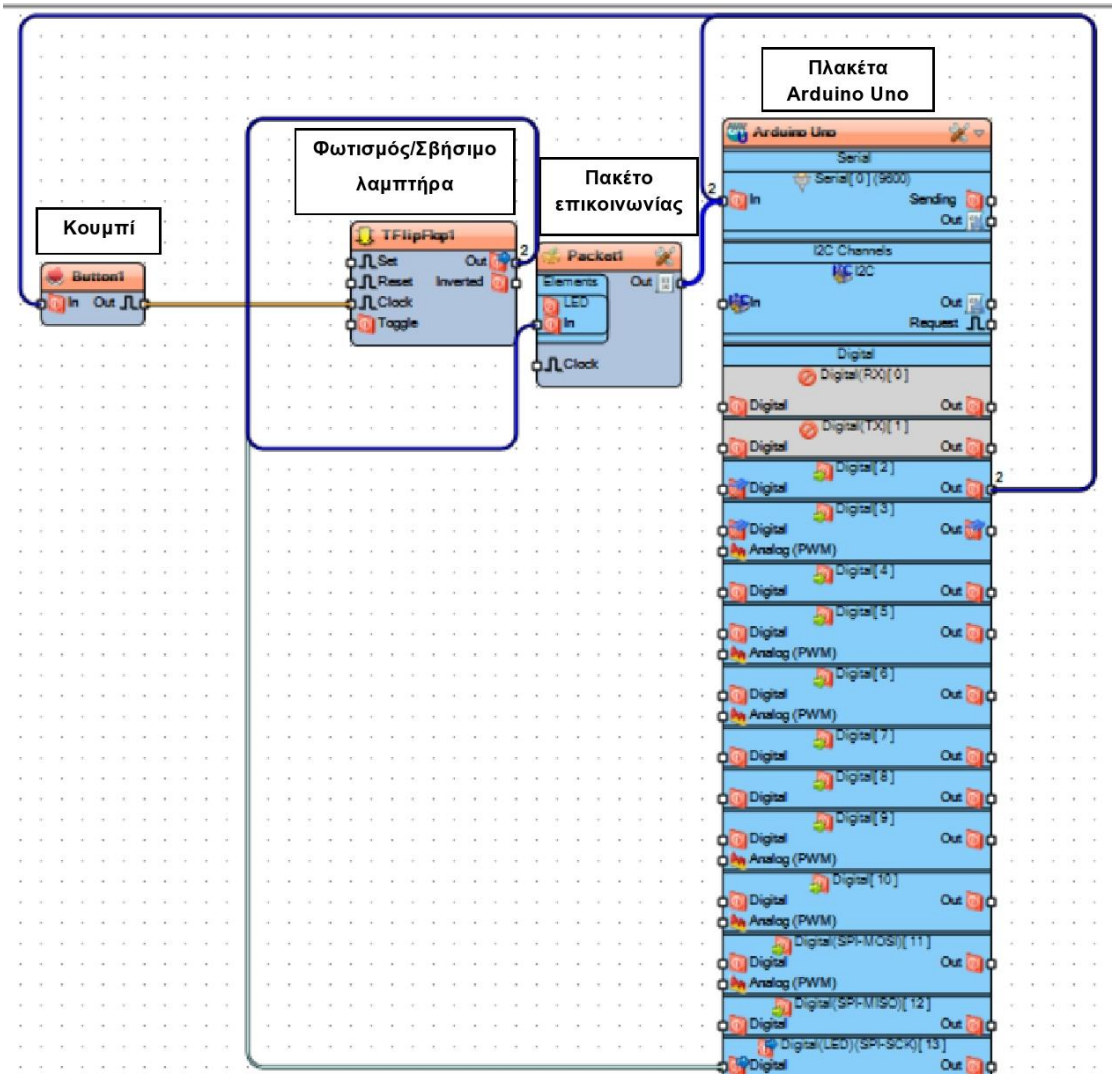
Λόγω αυξημένου κόστους της βαλβίδας και δυσκολίας στην πειραματική διαδικασία σχετικά με τη χρήση αυτής, θα επιτευχθεί ο φωτισμός και το σβήσιμο ενός λαμπτήρα LED μέσω ενός κουμπιού, με την παραδοχή ότι όπως μπορεί να ελεγχθεί αυτός, έτσι δύναται να ελεγχθεί και το άνοιγμα ή κλείσιμο της βαλβίδας.

Στο δευτερεύον κύκλωμα λοιπόν, συνδέονται κατάλληλα η πλακέτα μικροηλεκτρονικών, το κουμπί, ο λαμπτήρας LED, η αντίσταση και η πειραματική πλακέτα μεταξύ τους με τα καλώδια μεταγωγής (τύπου αρσενικό σε αρσενικό και αρσενικό σε θηλυκό) και έπειτα με το καλώδιο ενιαίου σειριακού διαύλου αυτό να συνδεθεί με τον ηλεκτρονικό υπολογιστή, ώστε να τροφοδοτηθεί με ρεύμα. Αρχικά, για τη σύνδεση της πλακέτας με τον λαμπτήρα, χρησιμοποιείται η πειραματική πλακέτα, πάνω στην οποία τοποθετούνται ο λαμπτήρας και η αντίσταση. Η κάθοδος του λαμπτήρα (το πιο κοντό σύρμα) συνδέεται στην εσοχή της γείωσης της πλακέτας μέσω καλωδίου αρσενικό με αρσενικό, ενώ η άνοδος του (το πιο μακρύ σύρμα) συνδέεται με κάποια ψηφιακή έξοδο της πλακέτας μέσω καλωδίου αρσενικό σε αρσενικό, παρεμβάλλοντας μια αντίσταση. Σημειώνεται ότι η ψηφιακή έξοδος της πλακέτας στην οποία ενώνεται ο λαμπτήρας είναι η δέκατη τρίτη. Έπειτα, για τη σύνδεση του κουμπιού με την πλακέτα συνδέονται η γείωση της πλακέτας, μέσω καλωδίου αρσενικό σε θηλυκό, και η μια ακίδα του κουμπιού, ενώ η δεύτερη ψηφιακή είσοδος/έξοδος της πλακέτας με την άλλη ακίδα του κουμπιού.


LED
RESISTOR 220Ω
SCL
SDA
AREF
GND
13
12
11~
10~
9~
8
7
6~
5~
4
3~
2
1 (TX)
0 (RX)
IOREF
RST
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
BUTTON

Σχήμα 4: Συνδεσμολογία δευτρεύοντος πραγματικού κυκλώματος

Για την αποτύπωση του δευτερεύοντος πραγματικού κυκλώματος με ψηφιακό τρόπο, χρησιμοποιείται η εφαρμογή Visuino. Εκεί επιλέγεται και πάλι η πλακέτα Arduino Uno, καθώς και το στοιχείου του κουμπιού. Στη συνέχεια, επιλέγεται το στοιχείο της μεταβολής κατάστασης, που χρησιμεύει στο πάτημα του κουμπιού, ώστε αυτό όταν πατιέται μια φορά να έχει κατάσταση ‘ανοιχτό’ και όταν πατιέται και δεύτερη να έχει κατάσταση ‘κλειστό’. Η είσοδος του στοιχείου του κουμπιού ενώνεται με τη δεύτερη ψηφιακή έξοδο της πλακέτας , ενώ η έξοδος του με την είσοδο του στοιχείου της μεταβολής κατάστασης. Παράλληλα, η δεύτερη έξοδος της πλακέτας ενώνεται με τη σειριακή θύρα της πλακέτας. Παρακάτω, επιλέγεται το στοιχείο του πακέτου επικοινωνίας, εισάγοντας μέσα ένα ψηφιακό στοιχείο για την οπτικοποίηση του λαμπτήρα, και η είσοδος του ενώνεται με την έξοδο του στοιχείου της μεταβολής κατάστασης, από την οποία υπάρχει και άλλη μια σύνδεση με τη δωδέκατη είσοδο της πλακέτας για να φωτιστεί ο λαμπτήρας. Τέλος, η έξοδος του στοιχείου επικοινωνίας ενώνεται με τη σειριακή θύρα της πλακέτας.

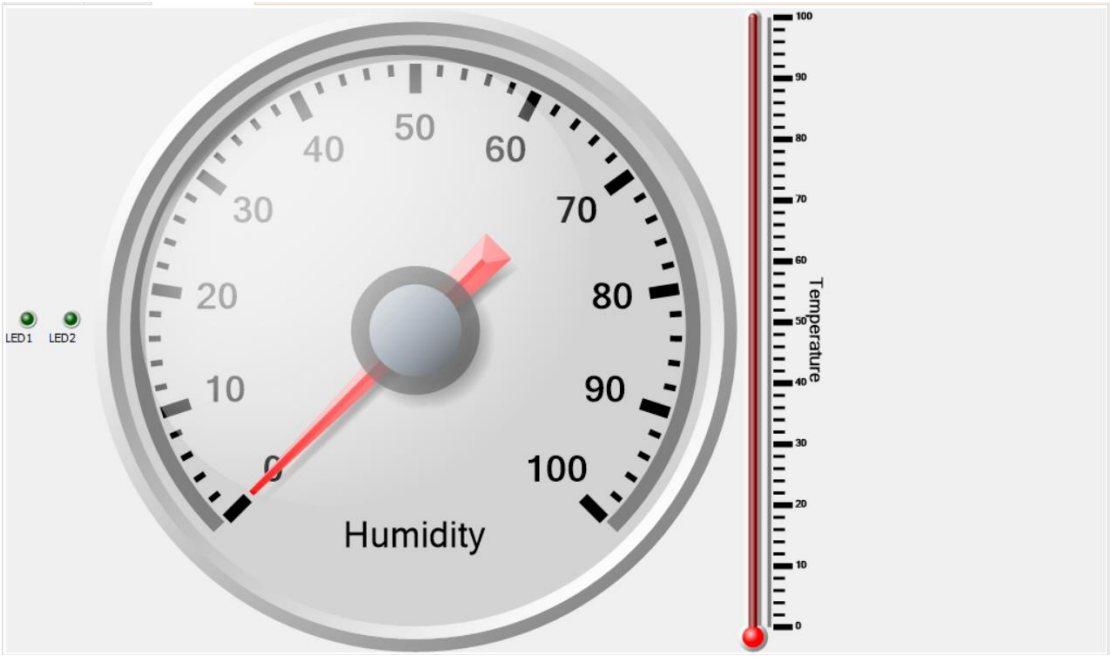


Εικόνα 8: Συνδεσμολογία δευτερεύοντος κυκλώματος στην εφαρμογή Visuino, με εμφανή όλα τα εμπλεκόμενα στοιχεία

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

3.3. ΟΠΤΙΚΟΠΟΙΗΣΕΙΣ ΜΕΤΡΗΣΕΩΝ ΚΑΙ ΕΛΕΓΧΟΣ

Με την υλοποίηση του κυκλώματος στο περιβάλλον εργασίας της εφαρμογής, έχει δημιουργηθεί ο αντίστοιχος κώδικας, ο οποίος επαληθεύεται και εκτελείται μέσω του λογισμικού Arduino IDE. Πριν από την εκτέλεση του το πεδίο των οπτικοποιήσεων, περιέχει τα όργανα μέτρησης, δηλαδή το κοντέρ και το θερμόμετρο, αλλά και τους λαμπτήρες, χωρίς κάποια ένδειξη.



Εικόνα 9: Όργανα μέτρησης υγρασίας και θερμοκρασίας και οπτικοποιημένοι λαμπτήρες LED , χωρίς κάποια ένδειξη

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

Με την εκτέλεση του κώδικα λαμβάνονται και αποτυπώνονται στα όργανα μέτρησης οι τιμές για τα μεγέθη της υγρασίας και θερμοκρασίας, ενώ παράλληλα η τιμή της θερμοκρασίας ελέγχεται αν υπερβαίνει μια συγκεκριμένη τιμή, αυτή των 25°C. Στην περίπτωση που η θερμοκρασία είναι μεγαλύτερη από αυτή την τιμή, ανάβει η ένδειξη του πρώτου λαμπτήρα, ενώ όταν είναι μικρότερη ή ίση με αυτή την τιμή, ανάβει η ένδειξη του δεύτερου λαμπτήρα. Σημειώνεται ότι η τιμή αυτή είναι ενδεικτική και χρησιμοποιείται για την περάτωση της πειραματικής διαδικασίας, καθώς στο Σύστημα Αποθήκευσης Ενέργειας μέσω Πεπιεσμένου Ατμοσφαιρικού Αέρα οι τιμές στα μεγέθη λόγω της συμπίεσης και της εκτόνωσης του αέρα θα διαφέρουν.

Παρακάτω παρουσιάζεται η οθόνη ελέγχου σε μια από τις δοκιμές της πειραματικής διαδικασίας, με την τιμή της υγρασίας να είναι 46% και της θερμοκρασίας 24°C, με αποτέλεσμα βάση του ελέγχου να ανάβει η ένδειξη του δεύτερου λαμπτήρα.

Εικόνα 10: Ενδείξεις οργάνων μέτρησης και λαμπτήρων

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

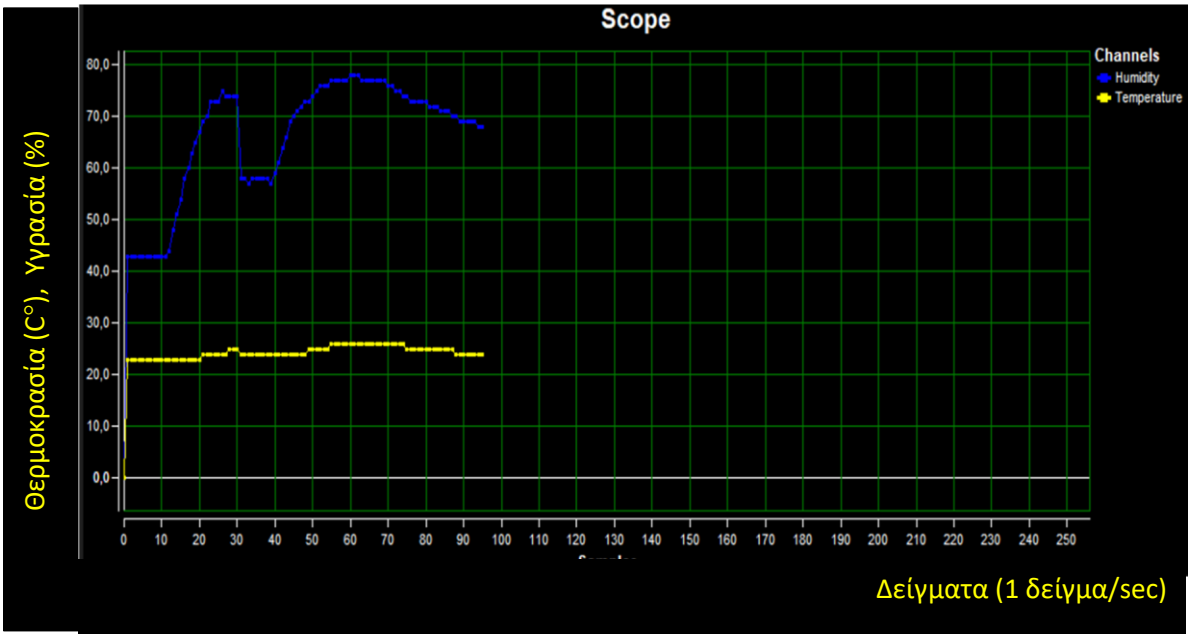
Εικόνα 11: Ενδείξεις οργάνων μέτρησης και λαμπτήρων

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

Στην ανωτέρω εικόνα παρουσιάζεται η οθόνη ελέγχου σε δοκιμή της πειραματικής διαδικασίας με άλλες επικρατούσες συνθήκες, όπου η τιμή της υγρασίας

είναι 67% και της θερμοκρασίας 32°C, με αποτέλεσμα βάση του ελέγχου να ανάβει η ένδειξη του πρώτου λαμπτήρα.

Μέσω της εφαρμογής δίνεται η δυνατότητα να αποτυπωθούν οι μετρήσεις των τιμών των μεγεθών που μετριούνται και σε διάγραμμα. Παρακάτω ακολουθεί το διάγραμμα όπως αυτό αποτυπώθηκε σε μια άλλη δοκιμή του πειράματος



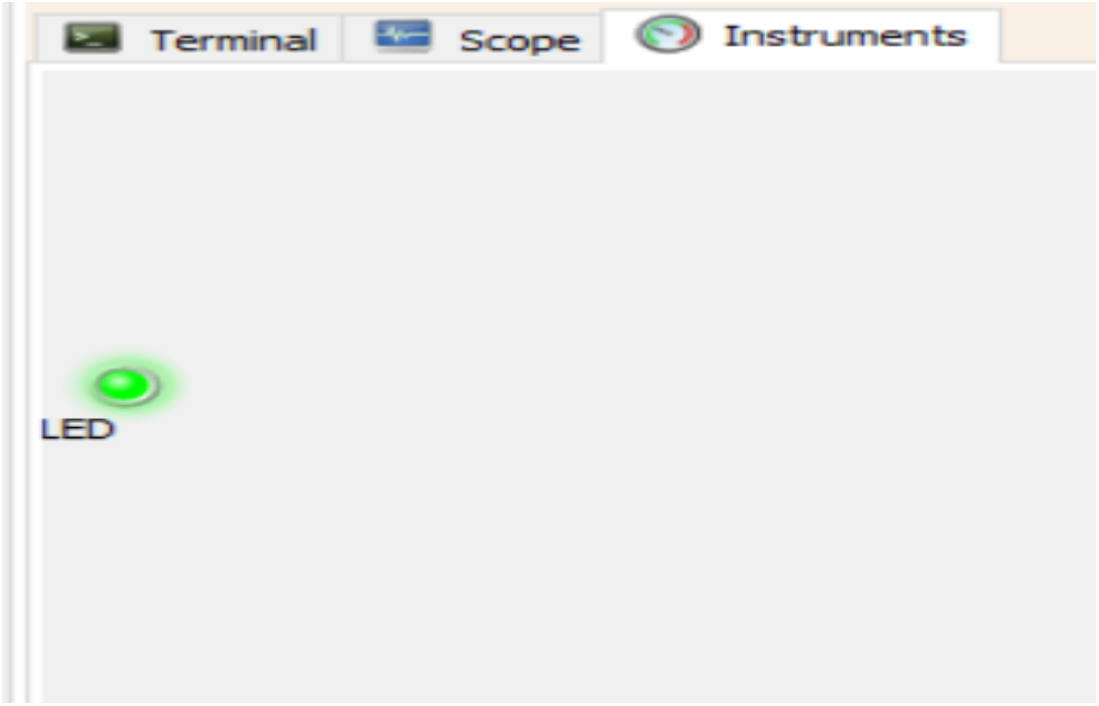
Εικόνα 12: Διάγραμμα απεικόνισης τιμών υγρασίας και θερμοκρασίας

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

Όταν η θερμοκρασία υπερβαίνει το επιθυμητό όριο, όπως αυτό φαίνεται στην εικόνα 11, κρίνεται απαραίτητη η παρέμβαση στο σύστημα. Με τη χρήση του δευτερεύοντος κυκλώματος επιτυγχάνεται διορθωτική ενέργεια και συγκεκριμένα ο φωτισμός και το σβήσιμο ενός λαμπτήρα, μέσω ενός πραγματικού κουμπιού. Το κουμπί αυτό δε δύναται να εμφανίζεται στην οθόνη ελέγχου της συγκεκριμένης πλατφόρμας, ώστε με το πάτημα του μέσω του ποντικιού του ηλεκτρονικού υπολογιστή, να επιτυγχάνεται η παρέμβαση και εν προκειμένω ο φωτισμός και το σβήσιμο του λαμπτήρα. Ως μοναδική λύση στο πρόβλημα αυτό καταδεικνύεται η χρήση ενός άλλου λογισμικού, ονόματι Nextion, σε συνδυασμό με την υπάρχουσα πλατφόρμα για τη σχεδίαση ενός κουμπιού στο περιβάλλον εργασίας του. Ύστερα από τη σχεδίαση του και τη σύνδεση του με μια οθόνη συμβατή με το Nextion (Nextion

display), αυτό οπτικοποιείται σε αυτήν και μπορεί να πατηθεί από εκεί, όπου με την κατάλληλη συνδεσμολογία της οθόνης με το κύκλωμα στην πλατφόρμα Visuino, επιτυγχάνεται ο έλεγχος του λαμπτήρα. Λόγω του αυξημένου κόστους της οθόνης Nextion, αυτή δε χρησιμοποιήθηκε αλλά επιλέχθηκε η περίπτωση υλοποίησης του ελέγχου, μέσω πραγματικού και όχι οπτικοποιημένου κουμπιού.

Παρακάτω, παρουσιάζεται η οθόνη ελέγχου σε μια από τις δοκιμές της πειραματικής διαδικασίας, με την ένδειξη του λαμπτήρα να ανάβει, καθώς έχει δοθεί η αντίστοιχη εντολή ελέγχου μέσω ενός πραγματικού κουμπιού.



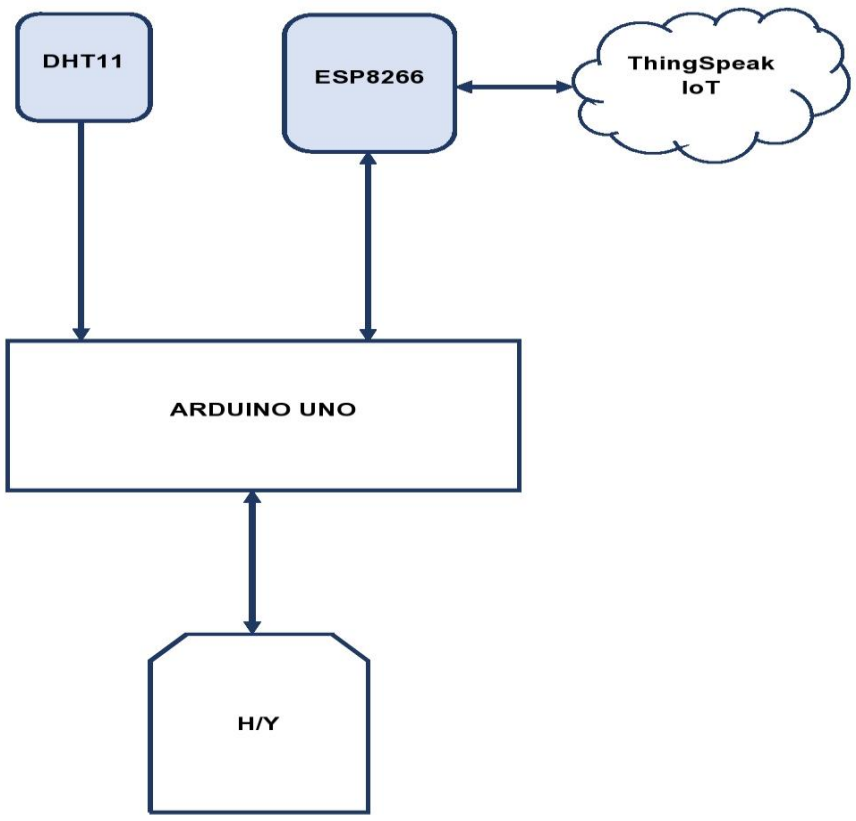
Εικόνα 13: Ένδειξη λαμπτήρα, ύστερα από έλεγχο του μέσω πραγματικού κουμπιού

Πηγή: Στιγμιότυπο από εφαρμογή Visuino

4. ΕΝΑΛΛΑΚΤΙΚΕΣ ΛΥΣΕΙΣ

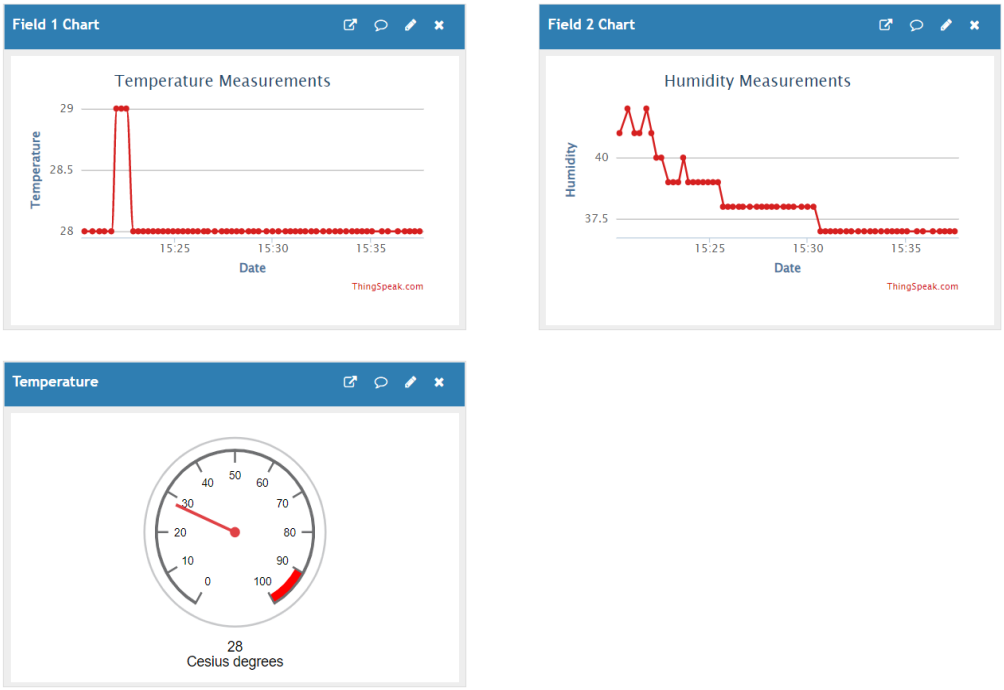
Οι εναλλακτικές λύσεις που παρουσιάζονται παρακάτω, αφορούν στην οπτικοποίηση των μετρήσεων των μεγεθών της υγρασίας και της θερμοκρασίας μέσω του αισθητήρα και διαφοροποιούνται, τόσο ως προς τη χρήση υλισμικού, όσο και λογισμικού. Συγκεκριμένα, για την πρώτη εναλλακτική λύση χρησιμοποιείται η πλακέτα μικροηλεκτρονικών Arduino Uno και ESP8266 Node MCU και η πλατφόρμα ThingSpeak, ενώ για τη δεύτερη η πλακέτα ESPP 8266 Node MCU και η πλατφόρμα Arduino Cloud.

4.1. ΠΡΩΤΗ ΕΝΑΛΛΑΚΤΙΚΗ



Σχήμα 5: Συνδεσμολογία πρώτης εναλλακτικής

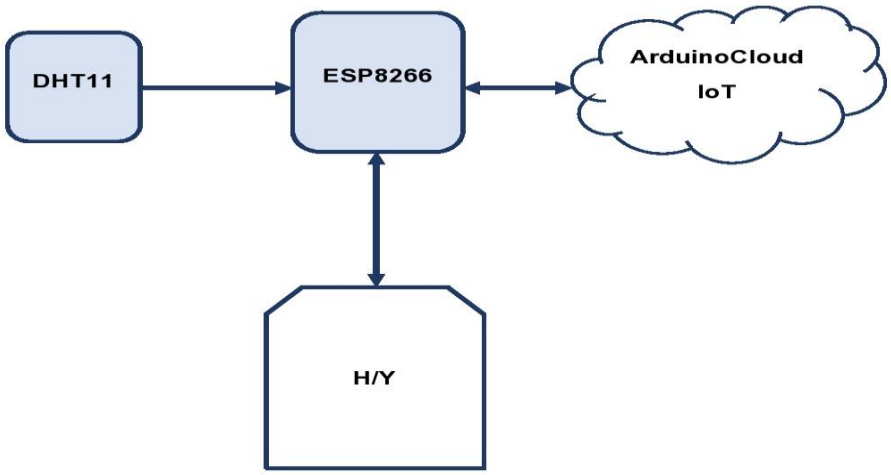
Κατά την υλοποίηση της πρώτης εναλλακτικής, συνδέονται κατάλληλα οι δυο πλακέτες και ο αισθητήρας και συντάσσεται και εκτελείται κώδικας στο χώρο εργασίας του Arduino IDE. Στον κώδικα αυτόν, πριν από τις εντολές που αντιστοιχούν στη λήψη των δεδομένων από τον αισθητήρα και εκείνων στη μεταφορά τους στην πλατφόρμα οπτικοποίησης, εισάγονται στα κατάλληλα πεδία το όνομα δικτύου που χρησιμοποιείται, καθώς και ο κωδικός του, ώστε η πλακέτα ESP8266 να συνδεθεί στο διαδίκτυο. Ουσιαστικά, αυτή η πλακέτα επιτρέπει τη μεταφορά δεδομένων στο διαδίκτυο σε πραγματικό χρόνο και στη συγκεκριμένη περίπτωση αποτελεί το διαμεσολαβητή μεταξύ της πλακέτας Arduino Uno και της πλατφόρμας ThingSpeak, χρησιμοποιώντας την ασύρματη τεχνολογία δικτύωσης (Wi-Fi). Εν συνεχεία, στην πλατφόρμα δημιουργείται το κανάλι στο οποίο θα μεταφερθούν τα δεδομένα και ορίζονται τα κλειδιά διεπαφής προγραμματισμού εφαρμογών (API keys), τα οποία μπορούν να έχουν τη δυνατότητα ανάγνωσης δεδομένων από το κανάλι και γραφής δεδομένων στο κανάλι. Δεδομένου, ότι στη συγκεκριμένη περίπτωση πρόκειται να επιτευχθεί μεταφορά δεδομένων στο κανάλι επιλέγεται το κλειδί διεπαφής προγραμματισμού εφαρμογών που αντιστοιχεί στη γραφή δεδομένων και εισάγεται στην εντολή του κώδικα για τη μεταφορά τους στην πλατφόρμα καθώς και ορίζονται, τόσο στον κώδικα, όσο και στην πλατφόρμα, τα δύο πεδία που αντιστοιχούν στις μεταβλητές (υγρασία και θερμοκρασία). Σε αυτήν επιλέγονται ως μέσα οπτικοποίησης και για τα δύο πεδία στα οποία θα εισαχθούν δεδομένα, διαγράμματα και κοντέρ μέτρησης. Έτσι, με τη συμπλήρωση των απαραίτητων στοιχείων του κώδικα, την εκτέλεση του και τη μορφοποίηση της πλατφόρμας, απεικονίζονται τα δεδομένα που έχουν συλλεχθεί από τον αισθητήρα.



Εικόνα 14: Ενδεικτική οπτικοποίηση δεδομένων στο ThingSpeak

Πηγή: Στιγμιότυπο από πλατφόρμα ThingSpeak

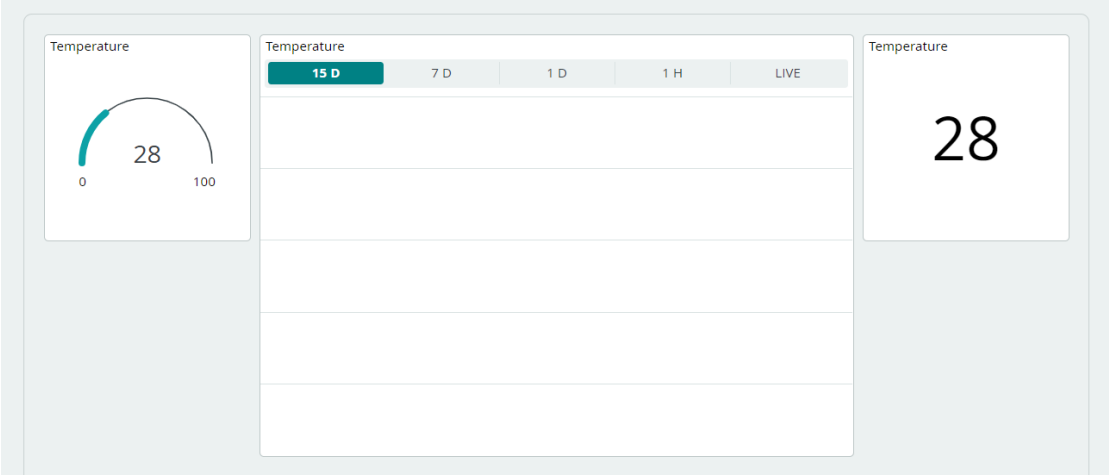
4.2. ΔΕΥΤΕΡΗ ΕΝΑΛΛΑΚΤΙΚΗ



Σχήμα 6: Συνδεσμολογία δεύτερης εναλλακτικής

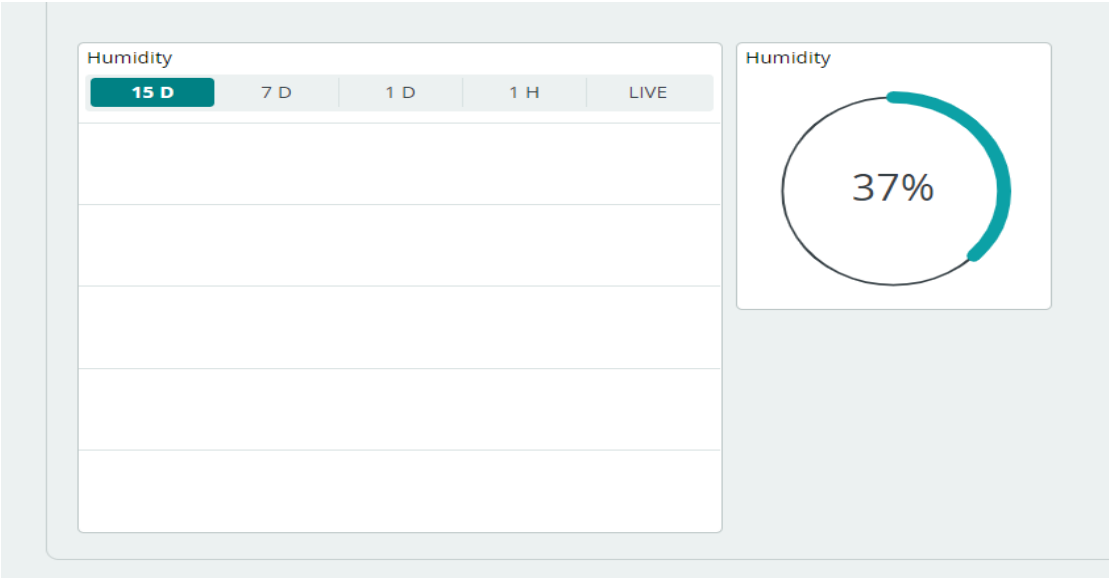
Κατά την υλοποίηση της δεύτερης εναλλακτικής, συνδέονται η πλακέτα ESP8266 με τον αισθητήρα. Στο περιβάλλον εργασίας της πλατφόρμας δημιουργείται το προς υλοποίηση σχέδιο, επιλέγοντας αρχικά τον τύπο της πλακέτας και συνδέοντας την με το διαδίκτυο, χρησιμοποιώντας και γράφοντας στα αντίστοιχα πεδία το όνομα και τον κωδικό του δικτύου, καθώς και το προσωπικό κλειδί της πλακέτας (secret key). Εν συνεχεία, εισάγονται στα πεδία των μεταβλητών του περιβάλλοντος εργασίας οι μεταβλητές που θα μετρηθούν, καθορίζοντας τον τύπο της μεταβλητής, τη λειτουργία ανάγνωσης των δεδομένων από τον αισθητήρα, ώστε να μπορούν μέσω της πλακέτας να μεταφερθούν στην πλατφόρμα και ο χρόνος μεταξύ των μετρήσεων. Με αυτόν τον

τρόπο δημιουργείται στο παρασκήνιο ο κώδικας που περιλαμβάνει τις αντίστοιχες εντολές και βιβλιοθήκες, και συμπληρώνοντας εντολές για την απόκτηση (ενδεικτικά: dht_sensor_getdata) και την εκτύπωση των δεδομένων (ενδεικτικά: Serial.print(“Temperature”)), αυτός είναι έτοιμος για μεταφόρτωση στην πλακέτα ESP8266 NodeMCU και εκτέλεση. Στο πεδίο των οπτικοποιήσεων επιλέγεται με ποια μορφή θα εμφανιστούν τα δεδομένα στην οθόνη.



Εικόνα 15: Ενδεικτική οπτικοποίηση δεδομένων θερμοκρασίας στο Arduino Cloud

Πηγή: Στιγμιότυπο από πλατφόρμα Arduino Cloud



Εικόνα 16: Ενδεικτική οπτικοποίηση δεδομένων υγρασίας στο Arduino Cloud

Πηγή: Στιγμιότυπο από πλατφόρμα Arduino Cloud

4.3. ΕΠΙΛΟΓΗ ΚΥΡΙΑΣ ΛΥΣΗΣ

Ωστόσο, τονίζεται ότι αν και οι δυο παραπάνω λύσεις διερευνήθηκαν και δοκιμάστηκαν επιτυχώς, τελικώς απορρίφθηκαν λόγω του γεγονότος ότι η λειτουργία τους απαιτεί συνεχή σύνδεση στο διαδίκτυο και η πρόσβαση σε αυτές γίνεται μέσω περιηγητή και όχι με τη μορφή εφαρμογής, όπως η λύση που επιλέχθηκε. Η εφαρμογή Visuino είναι και αυτή που επιλέγεται, διότι λειτουργεί τόσο με σύνδεση στο διαδίκτυο, όσο και χωρίς.

5.ΣΥΜΠΕΡΑΣΜΑΤΑ

Συμπερασματικά, γίνεται αντιληπτό ότι με τη χρήση όλων των απαραίτητων μέσων, τόσο αυτών του υλισμικού και εξαρτημάτων, όσο και εκείνων του λογισμικού, καθίσταται δυνατό να μετρηθούν και να αποτυπωθούν οπτικά τα μεγέθη της υγρασίας και θερμοκρασίας, καθώς και να υπάρχει συγκριτικός έλεγχος στην τιμή της θερμοκρασίας, που έχει ως αποτέλεσμα αντίστοιχες ενδείξεις και προειδοποίηση για τυχόν υπέρβαση συγκεκριμένου επιθυμητού ορίου. Σε συνέχεια της προειδοποιητικής ένδειξης για υπέρβαση του ορίου, καθίσταται δυνατός ο έλεγχος στον φωτισμό ή το σβήσιμο ενός λαμπτήρα. Η όλη διαδικασία εκτελείται πειραματικά και καθιστά σαφές το γεγονός ότι μπορεί να υπάρχει παρακολούθηση και έλεγχος στις επικρατούσες συνθήκες. Επιπρόσθετα, καταδεικνύει ότι με την τοποθέτηση ειδικών αισθητήρων μεγαλύτερης ακρίβειας και εύρους τιμών από αυτόν της πειραματικής διαδικασίας, στα σημεία τα οποία χρήζουν ελέγχου (συμπιεστής, αεροδεξαμενή, αεροστρόβιλος), μπορεί να επιτευχθεί η παρακολούθηση και ο έλεγχος ενός Συστήματος Αποθήκευσης Ενέργειας μέσω Πεπιεσμένου Ατμοσφαιρικού Αέρα (CAESS), όταν αυτό υλοποιηθεί.

6.ΠΑΡΑΡΤΗΜΑ

Παρακάτω παρατίθεται ο κώδικας για την υλοποίηση της πειραματικής διαδικασίας, χρησιμοποιώντας το πρωτεύον κύκλωμα.

```
#define VISUINO_ARDUINO_UNO

#include <OpenWire.h>
#include <Mitov.h>
#include <Mitov_StandardSerial.h>
#include <Mitov_DHT_Sensor.h>
#include <Mitov_Packet.h>
#include <Mitov_Math.h>
#include <Mitov_Sequence.h>
#include <Mitov_LogicFlipFlops.h>



// Shared Component Member Variables

namespace ComponentVariables
{
class
{
public:
  bool Value1 : 1;
  volatile uint32_t Value2 : 3;
  volatile uint32_t Value3 : 3;
  volatile uint32_t Value4 : 3;
  volatile uint32_t Value5 : 3;
  bool Value6 : 1;
  bool Value7 : 1;
  bool Value8 : 1;
  bool Value9 : 1;
  bool Value10 : 1;
  bool Value11 : 1;
  bool Value12 : 1;
  bool Value13 : 1;
  bool Value14 : 1;
  bool Value15 : 1;
  bool Value16 : 1;
  uint32_t Value17 : 2;
  bool Value18 : 1;
  bool Value19 : 1;
  bool Value20 : 1;

} BitFields;

class Variable1
{
```

```
public:
  inline static bool GetValue() { return BitFields.Value1; }
  inline static void SetValue( bool AValue ) { BitFields.Value1 = AValu
e; }

};

class Variable2
{
public:
  inline static uint32_t GetValue() { return BitFields.Value2; }
  inline static void SetValue( uint32_t AValue ) { BitFields.Value2 = A
Value; }

};

class Variable3
{
public:
  inline static uint32_t GetValue() { return BitFields.Value3; }
  inline static void SetValue( uint32_t AValue ) { BitFields.Value3 = A
Value; }

};

class Variable4
{
public:
  inline static uint32_t GetValue() { return BitFields.Value4; }
  inline static void SetValue( uint32_t AValue ) { BitFields.Value4 = A
Value; }

};

class Variable5
{
public:
  inline static uint32_t GetValue() { return BitFields.Value5; }
  inline static void SetValue( uint32_t AValue ) { BitFields.Value5 = A
Value; }

};

class Variable6
{
public:
  inline static bool GetValue() { return BitFields.Value6; }
```

```
  inline static void SetValue( bool AValue ) { BitFields.Value6 = AValu
e; }

};

class Variable7
{
public:
  inline static bool GetValue() { return BitFields.Value7; }
  inline static void SetValue( bool AValue ) { BitFields.Value7 = AValu
e; }

};

class Variable8
{
public:
  inline static bool GetValue() { return BitFields.Value8; }
  inline static void SetValue( bool AValue ) { BitFields.Value8 = AValu
e; }

};

class Variable9
{
public:
  inline static bool GetValue() { return BitFields.Value9; }
  inline static void SetValue( bool AValue ) { BitFields.Value9 = AValu
e; }

};

class Variable10
{
public:
  inline static bool GetValue() { return BitFields.Value10; }
  inline static void SetValue( bool AValue ) { BitFields.Value10 = AVal
ue; }

};

class Variable11
{
public:
  inline static bool GetValue() { return BitFields.Value11; }
  inline static void SetValue( bool AValue ) { BitFields.Value11 = AVal
ue; }
```

```
};

class Variable12
{
public:
  inline static bool GetValue() { return BitFields.Value12; }
  inline static void SetValue( bool AValue ) { BitFields.Value12 = AVal
ue; }

};

class Variable13
{
public:
  inline static bool GetValue() { return BitFields.Value13; }
  inline static void SetValue( bool AValue ) { BitFields.Value13 = AVal
ue; }

};

class Variable14
{
public:
  inline static bool GetValue() { return BitFields.Value14; }
  inline static void SetValue( bool AValue ) { BitFields.Value14 = AVal
ue; }

};

class Variable15
{
public:
  inline static bool GetValue() { return BitFields.Value15; }
  inline static void SetValue( bool AValue ) { BitFields.Value15 = AVal
ue; }

};

class Variable16
{
public:
  inline static bool GetValue() { return BitFields.Value16; }
  inline static void SetValue( bool AValue ) { BitFields.Value16 = AVal
ue; }

};

class Variable17
```

```
{
public:
  inline static uint32_t GetValue() { return BitFields.Value17; }
  inline static void SetValue( uint32_t AValue ) { BitFields.Value17 =
AValue; }

};

class Variable18
{
public:
  inline static bool GetValue() { return BitFields.Value18; }
  inline static void SetValue( bool AValue ) { BitFields.Value18 = AVal
ue; }

};

class Variable19
{
public:
  inline static bool GetValue() { return BitFields.Value19; }
  inline static void SetValue( bool AValue ) { BitFields.Value19 = AVal
ue; }

};

class Variable20
{
public:
  inline static bool GetValue() { return BitFields.Value20; }
  inline static void SetValue( bool AValue ) { BitFields.Value20 = AVal
ue; }

};

} // ComponentVariables

// Arduino Constant Declarations

namespace VisuinoConstants
{
class FloatValue0
{
public:
    inline static constexpr float GetValue() { return 24; }
};

  constexpr PROGMEM const uint8_t ArrayValue0[] = { 85,85,85,85 };
```

```
} // VisuinoConstants

// Pin Call Declarations

namespace PinCalls
{
class PinCallerReceive0
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive1
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive2
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive3
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive4
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive5
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive6
{
public:
  void Notify( void *_Data );

};
```

```
} // PinCalls

// Call Chains

namespace CallChains
{
class InterruptFalling1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call();

};
class CheckPopulated1
{
public:
  inline static uint32_t Count() { return 4; }
  static void Call( bool & AIsPopulated );

};
class GetData1
{
public:
  inline static uint32_t Count() { return 4; }
  static void Call( uint8_t *& ADataStart, uint32_t & ADataIndex, uint8
_t & AOffset );

};
class GetSize1
{
public:
  inline static uint32_t Count() { return 4; }
  static void Call( int32_t AIndex, bool & AAlligned, int & AResult );

};
class Expand1
{
public:
  inline static uint32_t Count() { return 4; }
  static void Call( int32_t AIndex, uint8_t * const & AInBuffer, uint8_
t * const & AOutBuffer, int & ASize, bool & AResult );

};
class GetValue1
{
public:
  inline static uint32_t Count() { return 0; }
  static void Call( int32_t AIndex, uint8_t & AValue );
```

```
};
class ApplyValues1
{
public:
  inline static uint32_t Count() { return 0; }
  static void Call( uint8_t * AValue );

};
class MoveToNextIndex1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( int32_t *& ACurrentIndex );

};
class SetElementValue1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( int32_t AIndex, bool AValue );

};
class InitElementValue1
{
public:
  inline static uint32_t Count() { return 0; }
  static void Call();

};
} // CallChains

// Arduino Board Declarations

namespace BoardDeclarations
{
namespace Types
{
typedef Mitov::ArduinoDigitalInputChannel<
  Mitov::ConstantProperty<51, bool, false>, // InitialValue
  Mitov::ConstantProperty<25, bool, false>, // IsOpenDrain
  Mitov::ConstantProperty<25, bool, true>, // IsOutput
  Mitov::ConstantProperty<20, bool, false>, // IsPullDown
  Mitov::ConstantProperty<5, bool, false >, // IsPullUp
  Mitov::DigitalPin_NoImplementation<6 >, // OutputPin
  2 // PIN
  > ArduinoDigitalChannel_2;
} // Types
```

```
namespace Instances
{
Types::ArduinoDigitalChannel_2 ArduinoDigitalChannel_2;
} // Instances

namespace Types
{
typedef Mitov::ArduinoDigitalChannel<
  Mitov::ConstantProperty<51, bool, false>, // InitialValue
  Mitov::ConstantProperty<22, bool, false>, // IsAnalog
  Mitov::ConstantProperty<24, bool, false>, // IsCombinedInOut
  Mitov::ConstantProperty<21, bool, false>, // IsOpenDrain
  Mitov::ConstantProperty<25, bool, true>, // IsOutput
  Mitov::ConstantProperty<20, bool, false>, // IsPullDown
  Mitov::ConstantProperty<5, bool, false >, // IsPullUp
  Mitov::ConstantProperty<23, bool, true>, // IsRawInput
  Mitov::DigitalPin_NoImplementation<6 >, // OutputPin
  12 // PIN
  > ArduinoDigitalChannel_12;
} // Types

namespace Instances
{
Types::ArduinoDigitalChannel_12 ArduinoDigitalChannel_12;
} // Instances

namespace Types
{
typedef Mitov::ArduinoDigitalChannel<
  Mitov::ConstantProperty<51, bool, false>, // InitialValue
  Mitov::ConstantProperty<22, bool, false>, // IsAnalog
  Mitov::ConstantProperty<24, bool, false>, // IsCombinedInOut
  Mitov::ConstantProperty<21, bool, false>, // IsOpenDrain
  Mitov::ConstantProperty<25, bool, true>, // IsOutput
  Mitov::ConstantProperty<20, bool, false>, // IsPullDown
  Mitov::ConstantProperty<5, bool, false >, // IsPullUp
  Mitov::ConstantProperty<23, bool, true>, // IsRawInput
  Mitov::DigitalPin_NoImplementation<6 >, // OutputPin
  13 // PIN
  > ArduinoDigitalChannel_13;
} // Types

namespace Instances
{
Types::ArduinoDigitalChannel_13 ArduinoDigitalChannel_13;
} // Instances
```

```
namespace Types
{
typedef Mitov::SerialPort<
  SERIAL_TYPE, // 0_T_TYPE
  Serial, // 1_C_OBJECT
  Mitov::ConstantProperty<4, uint32_t, 0 >, // AfterSendingDelay
  Mitov::ConstantProperty<7, uint32_t, 8 >, // DataBits
  Mitov::ConstantProperty<2, bool, true >, // Enabled
  Mitov::ConstantProperty<12, uint32_t, 0 >, // FEndTime
  Mitov::ConstantProperty<10, bool, false >, // FSending
  Mitov::GenericPin_NoImplementation<5 >, // OutputPin
  Mitov::ConstantProperty<9, Mitov::TArduinoStandardSerialParity, Mitov
::spNone >, // Parity
  Mitov::DigitalPin_NoImplementation<3 >, // SendingOutputPin
  Mitov::ConstantProperty<6, uint32_t, 9600 >, // Speed
  Mitov::ConstantProperty<8, uint32_t, 1 > // StopBits
   > SerialPort0;
} // Types

namespace Instances
{
Types::SerialPort0 SerialPort0;
} // Instances

namespace Types
{
typedef Mitov::ArduinoSerialInput_Binary<BoardDeclarations::Types::Seri
alPort0, BoardDeclarations::Instances::SerialPort0, Mitov::TArray<uint8
_t>> SerialPort0_Input_IOWByteStream_1;
} // Types

namespace Instances
{
Types::SerialPort0_Input_IOWByteStream_1 SerialPort0_Input_IOWByteStrea
m_1;
} // Instances

} // BoardDeclarations

// Interrupts

namespace Interrupts
{
void __ICACHE_RAM_ATTR__ Handler2();

OpenWire::PinFallingInterrupt<2, ::CallChains::InterruptFalling1> Pin2;

void __ICACHE_RAM_ATTR__ Handler2()
```

```
{
    Pin2.InterruptHandler();
}

} // Interrupts

// Declarations

namespace Declarations
{
namespace Types
{
typedef Mitov::DHTSensor_Interrupt<
  Mitov::ConstantProperty<8, bool, false>, // ClockInputPin_o_IsConnect
ed
  Mitov::ConstantProperty<3, bool, true >, // Enabled
  Mitov::TypedVariable<19, uint32_t, ::ComponentVariables::Variable5 >,
 // FBitIndex
  Mitov::TypedVariable<11, bool, ::ComponentVariables::Variable1 >, //
FClocked
  Mitov::TypedVariable<17, uint32_t, ::ComponentVariables::Variable4 >,
 // FIndex
  Mitov::TypedVariable<13, uint32_t, ::ComponentVariables::Variable2 >,
 // FState
  Mitov::TypedVariable<15, uint32_t, ::ComponentVariables::Variable3 >,
 // FStatus
  Mitov::AnalogPin_EmbeddedPinImplementation<5, ::PinCalls::PinCallerRe
ceive1 >, // HumidityOutputPin
  Interrupts::Handler2, // INTERRUPT
  Mitov::ConstantProperty<6, bool, false >, // InFahrenheit
  2, // SensorOutputPin
  Mitov::AnalogPin_EmbeddedPinImplementation<4, ::PinCalls::PinCallerRe
ceive0 > // TemperatureOutputPin
   > HumidityThermometer1;
} // Types

namespace Instances
{
Types::HumidityThermometer1 HumidityThermometer1;
} // Instances

namespace Types
{
typedef Mitov::Packet<
  4, // COUNT_Elements
  Mitov::NestedProperty<12, Mitov::PacketChecksumElement<
    Mitov::ConstantProperty<11, bool, true > // Enabled
     > >, // Checksum
```

```
  Mitov::EmbeddedCallChain<CallChains::CheckPopulated1 >, // Elements_C
heckPopulated
  Mitov::EmbeddedCallChain<CallChains::Expand1 >, // Elements_Expand
  Mitov::EmbeddedCallChain<CallChains::GetData1 >, // Elements_GetData
  Mitov::EmbeddedCallChain<CallChains::GetSize1 >, // Elements_GetSize
  Mitov::TypedVariable<15, bool, ::ComponentVariables::Variable7 >, //
FModified
  Mitov::TypedVariable<13, bool, ::ComponentVariables::Variable6 >, //
FNeedsNewSize
  Mitov::TypedVariable<17, bool, ::ComponentVariables::Variable8 >, //
FRefreshed
  Mitov::NestedProperty<9, Mitov::TArduinoHeadMarkerBinaryPacketElement
<
    Mitov::ConstantPropertyArray<8, uint8_t, uint8_t, ::VisuinoConstant
s::ArrayValue0, 4 >, // Bytes
    Mitov::EmbeddedCallChain<CallChains::ApplyValues1 >, // Bytes_Apply
Values
    Mitov::EmbeddedCallChain<CallChains::GetValue1 > // Bytes_GetValue
     > >, // HeadMarker
  Mitov::ConstantProperty<5, bool, false >, // OnlyModified
  Mitov::TypedPin_EmbeddedPinImplementation<3, ::PinCalls::PinCallerRec
eive2, Mitov::TArray<uint8_t> > // OutputPin
   > Packet1;
} // Types

namespace Instances
{
Types::Packet1 Packet1;
} // Instances

namespace Types
{
typedef Mitov::PacketDigitalBinaryElement<
  Declarations::Types::Packet1, // 0_TYPE_OWNER
  Declarations::Instances::Packet1, // 1_NAME_OWNER
  Mitov::TypedVariable<5, bool, ::ComponentVariables::Variable10 >, //
FBoolValue
  Mitov::TypedVariable<3, bool, ::ComponentVariables::Variable9 > // FP
opulated
   > TArduinoDigitalBinaryPacketElement1;
} // Types

namespace Instances
{
Types::TArduinoDigitalBinaryPacketElement1 TArduinoDigitalBinaryPacketE
lement1 = Types::TArduinoDigitalBinaryPacketElement1( false );
} // Instances
```

```
namespace Types
{
typedef Mitov::PacketDigitalBinaryElement<
  Declarations::Types::Packet1, // 0_TYPE_OWNER
  Declarations::Instances::Packet1, // 1_NAME_OWNER
  Mitov::TypedVariable<5, bool, ::ComponentVariables::Variable12 >, //
FBoolValue
  Mitov::TypedVariable<3, bool, ::ComponentVariables::Variable11 > // F
Populated
   > TArduinoDigitalBinaryPacketElement2;
} // Types

namespace Instances
{
Types::TArduinoDigitalBinaryPacketElement2 TArduinoDigitalBinaryPacketE
lement2 = Types::TArduinoDigitalBinaryPacketElement2( false );
} // Instances

namespace Types
{
typedef Mitov::BasicTypedPacketSinkElement<
  Declarations::Types::Packet1, // 0_TYPE_OWNER
  Declarations::Instances::Packet1, // 1_NAME_OWNER
  Mitov::TypedVariable<4, bool, ::ComponentVariables::Variable13 >, //
FPopulated
  float, // PIN
  float // TYPE
   > TArduinoAnalogBinaryPacketElement1;
} // Types

namespace Instances
{
Types::TArduinoAnalogBinaryPacketElement1 TArduinoAnalogBinaryPacketEle
ment1 = Types::TArduinoAnalogBinaryPacketElement1( false );
} // Instances

namespace Types
{
typedef Mitov::BasicTypedPacketSinkElement<
  Declarations::Types::Packet1, // 0_TYPE_OWNER
  Declarations::Instances::Packet1, // 1_NAME_OWNER
  Mitov::TypedVariable<4, bool, ::ComponentVariables::Variable14 >, //
FPopulated
  float, // PIN
  float // TYPE
   > TArduinoAnalogBinaryPacketElement2;
} // Types
```

```
namespace Instances
{
Types::TArduinoAnalogBinaryPacketElement2 TArduinoAnalogBinaryPacketEle
ment2 = Types::TArduinoAnalogBinaryPacketElement2( false );
} // Instances

namespace Types
{
typedef Mitov::CompareValue<
  Mitov::CompareType_Implementation_ctBigger, // CompareType
  Mitov::ConstantProperty<4, bool, true >, // Enabled
  Mitov::TypedVariable<9, bool, ::ComponentVariables::Variable15 >, //
FStarted
  Mitov::ConstantProperty<8, bool, false >, // OnlyChanged
  Mitov::DigitalPin_EmbeddedPinImplementation<5, ::PinCalls::PinCallerR
eceive3 >, // OutputPin
  float, // TYPE
  Mitov::ConstantPropertyFloat<6, float, ::VisuinoConstants::FloatValue
0 > // Value
   > CompareValue1;
} // Types

namespace Instances
{
Types::CompareValue1 CompareValue1 = Types::CompareValue1( 0.0f );
} // Instances

namespace Types
{
typedef Mitov::CompareValue<
  Mitov::CompareType_Implementation_ctSmallerOrEqual, // CompareType
  Mitov::ConstantProperty<4, bool, true >, // Enabled
  Mitov::TypedVariable<9, bool, ::ComponentVariables::Variable16 >, //
FStarted
  Mitov::ConstantProperty<8, bool, true >, // OnlyChanged
  Mitov::DigitalPin_EmbeddedPinImplementation<5, ::PinCalls::PinCallerR
eceive4 >, // OutputPin
  float, // TYPE
  Mitov::ConstantPropertyFloat<6, float, ::VisuinoConstants::FloatValue
0 > // Value
   > CompareValue2;
} // Types

namespace Instances
{
Types::CompareValue2 CompareValue2 = Types::CompareValue2( 0.0f );
} // Instances
```

```
namespace Types
{
typedef Mitov::Sequence<
  1, // COUNT_Elements
  Mitov::ConstantProperty<9, bool, true >, // CanRestart
  Mitov::EmbeddedCallChain<CallChains::InitElementValue1 >, // Elements
_InitElementValue
  Mitov::EmbeddedCallChain<CallChains::MoveToNextIndex1 >, // Elements_
MoveToNextIndex
  Mitov::EmbeddedCallChain<CallChains::SetElementValue1 >, // Elements_
SetElementValue
  Mitov::ConstantProperty<3, bool, true >, // Enabled
  Mitov::TypedVariable<6, uint32_t, ::ComponentVariables::Variable17 >,
 // FCurrentIndex
  Mitov::TypedPin_NoImplementation<5, uint32_t >, // IntervalOutputPin
  Mitov::ConstantProperty<8, bool, false > // Repeat
   > Sequence1;
} // Types

namespace Instances
{
Types::Sequence1 Sequence1;
} // Instances

namespace Types
{
typedef Mitov::SequenceElement<
  Declarations::Types::Sequence1, // 0_TYPE_OWNER
  Declarations::Instances::Sequence1, // 1_NAME_OWNER
  Mitov::ConstantProperty<6, uint32_t, 0 >, // Delay
  Mitov::ConstantProperty<4, bool, true >, // Enabled
  0, // INDEX
  Mitov::ConstantProperty<5, bool, false >, // InMicroSeconds
  Mitov::ClockPin_EmbeddedPinImplementation<7, ::PinCalls::PinCallerRec
eive5 > // OutputPin
   > TArduinoSequenceClockElement1;
} // Types

namespace Instances
{
Types::TArduinoSequenceClockElement1 TArduinoSequenceClockElement1;
} // Instances

namespace Types
{
typedef Mitov::TFlipFlop<
  Mitov::TypedVariable<10, bool, ::ComponentVariables::Variable20 >, //
 FToggleValue
```

```
  Mitov::TypedVariableValue<5, bool, ::ComponentVariables::Variable19,
false >, // InitialValue
  Mitov::DigitalPin_NoImplementation<4 >, // InvertedOutputPin
  Mitov::DigitalPin_EmbeddedPinImplementation_ChangeOnly<3, ::PinCalls:
:PinCallerReceive6, Mitov::TypedVariable<0, bool, ::ComponentVariables:
:Variable18 > > // OutputPin
   > TFlipFlop1;
} // Types

namespace Instances
{
Types::TFlipFlop1 TFlipFlop1;
} // Instances

} // Declarations

// Call Chains

namespace CallChains
{
void InterruptFalling1::Call()
{
  Declarations::Instances::HumidityThermometer1.InterruptHandler( false
 );
}

void CheckPopulated1::Call( bool & AIsPopulated )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.CheckPop
ulated( AIsPopulated );
  Declarations::Instances::TArduinoDigitalBinaryPacketElement2.CheckPop
ulated( AIsPopulated );
  Declarations::Instances::TArduinoAnalogBinaryPacketElement1.CheckPopu
lated( AIsPopulated );
  Declarations::Instances::TArduinoAnalogBinaryPacketElement2.CheckPopu
lated( AIsPopulated );
}

void GetData1::Call( uint8_t *& ADataStart, uint32_t & ADataIndex, uint
8_t & AOffset )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.GetData(
 ADataStart, ADataIndex, AOffset );
  Declarations::Instances::TArduinoDigitalBinaryPacketElement2.GetData(
 ADataStart, ADataIndex, AOffset );
  Declarations::Instances::TArduinoAnalogBinaryPacketElement1.GetData(
ADataStart, ADataIndex, AOffset );
```

```
  Declarations::Instances::TArduinoAnalogBinaryPacketElement2.GetData(
ADataStart, ADataIndex, AOffset );
}

void GetSize1::Call( int32_t AIndex, bool & AAlligned, int & AResult )
{
  switch( AIndex )
  {
    case 0: Declarations::Instances::TArduinoDigitalBinaryPacketElement
1.GetSize( AAlligned, AResult ); break;
    case 1: Declarations::Instances::TArduinoDigitalBinaryPacketElement
2.GetSize( AAlligned, AResult ); break;
    case 2: Declarations::Instances::TArduinoAnalogBinaryPacketElement1
.GetSize( AAlligned, AResult ); break;
    case 3: Declarations::Instances::TArduinoAnalogBinaryPacketElement2
.GetSize( AAlligned, AResult ); break;
  }
}

void Expand1::Call( int32_t AIndex, uint8_t * const & AInBuffer, uint8_
t * const & AOutBuffer, int & ASize, bool & AResult )
{
  switch( AIndex )
  {
    case 0: Declarations::Instances::TArduinoDigitalBinaryPacketElement
1.Expand( AInBuffer, AOutBuffer, ASize, AResult ); break;
    case 1: Declarations::Instances::TArduinoDigitalBinaryPacketElement
2.Expand( AInBuffer, AOutBuffer, ASize, AResult ); break;
    case 2: Declarations::Instances::TArduinoAnalogBinaryPacketElement1
.Expand( AInBuffer, AOutBuffer, ASize, AResult ); break;
    case 3: Declarations::Instances::TArduinoAnalogBinaryPacketElement2
.Expand( AInBuffer, AOutBuffer, ASize, AResult ); break;
  }
}

void GetValue1::Call( int32_t AIndex, uint8_t & AValue )
{
}

void ApplyValues1::Call( uint8_t * AValue )
{
}

void MoveToNextIndex1::Call( int32_t *& ACurrentIndex )
{
  Declarations::Instances::TArduinoSequenceClockElement1.MoveToNextInde
x( ACurrentIndex );
}
```

```
void SetElementValue1::Call( int32_t AIndex, bool AValue )
{
  switch( AIndex )
  {
    case 0: Declarations::Instances::TArduinoSequenceClockElement1.SetE
lementValue( AValue ); break;
  }
}

void InitElementValue1::Call()
{
}

} // CallChains

// Pin Call Implementations

namespace PinCalls
{
void PinCallerReceive0::Notify( void *_Data )
{
  Declarations::Instances::CompareValue1.InputPin_o_Receive( _Data );
  Declarations::Instances::CompareValue2.InputPin_o_Receive( _Data );
  Declarations::Instances::TArduinoAnalogBinaryPacketElement2.InputPin_
o_Receive( _Data );
}

void PinCallerReceive1::Notify( void *_Data )
{
  Declarations::Instances::TArduinoAnalogBinaryPacketElement1.InputPin_
o_Receive( _Data );
}

void PinCallerReceive2::Notify( void *_Data )
{
  BoardDeclarations::Instances::SerialPort0_Input_IOWByteStream_1.Input
Pin_o_Receive( _Data );
  Declarations::Instances::TArduinoSequenceClockElement1.ClockInputPin_
o_Receive( _Data );
}

void PinCallerReceive3::Notify( void *_Data )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.InputPin
_o_Receive( _Data );
}
```

```
void PinCallerReceive4::Notify( void *_Data )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement2.InputPin
_o_Receive( _Data );
}

void PinCallerReceive5::Notify( void *_Data )
{
  Declarations::Instances::TFlipFlop1.ClockInputPin_o_Receive( _Data );
}

void PinCallerReceive6::Notify( void *_Data )
{
  BoardDeclarations::Instances::ArduinoDigitalChannel_12.DigitalInputPi
n_o_Receive( _Data );
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.DigitalInputPi
n_o_Receive( _Data );
}

} // PinCalls

namespace ComponentsHardware
{
void SystemUpdateHardware()
{
}
} // ComponentsHardware

//The setup function is called once at startup of the sketch
void setup()
{
  BoardDeclarations::Instances::SerialPort0.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_12.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemInit();
  Declarations::Instances::HumidityThermometer1.SystemInit();

  Interrupts::Pin2.Init( Interrupts::Handler2 );

  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemStart();
  BoardDeclarations::Instances::ArduinoDigitalChannel_12.SystemStart();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemStart();
  Declarations::Instances::HumidityThermometer1.SystemStart();
  Declarations::Instances::Packet1.SystemStart();
  Declarations::Instances::CompareValue1.SystemStart();
  Declarations::Instances::CompareValue2.SystemStart();
  Declarations::Instances::Sequence1.SystemStart();
  Declarations::Instances::TFlipFlop1.SystemStart();
```

```
}

// The loop function is called in an endless loop
void loop()
{
  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemLoopBegin
();
  BoardDeclarations::Instances::ArduinoDigitalChannel_12.SystemLoopBegi
n();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemLoopBegi
n();
  Declarations::Instances::HumidityThermometer1.SystemLoopBegin();
  Declarations::Instances::Packet1.SystemLoopBegin();
  Declarations::Instances::TArduinoSequenceClockElement1.SystemLoopBegi
n();
```

Παρακάτω παρατίθεται ο κώδικας για την υλοποίηση της πειραματικής διαδικασίας, χρησιμοποιώντας το δευτερεύον κύκλωμα.

```
#define VISUINO_ARDUINO_UNO

#include <OpenWire.h>
#include <Mitov.h>
#include <Mitov_StandardSerial.h>
#include <Mitov_Button.h>
#include <Mitov_LogicFlipFlops.h>
#include <Mitov_Packet.h>



// Shared Component Member Variables

namespace ComponentVariables
{
class
{
public:
  bool Value1 : 1;
  bool Value2 : 1;
  bool Value3 : 1;
  bool Value4 : 1;
  bool Value5 : 1;
  bool Value6 : 1;
  bool Value7 : 1;
  bool Value8 : 1;
  bool Value9 : 1;
  bool Value10 : 1;
  bool Value11 : 1;
```

```
  bool Value12 : 1;

} BitFields;

class Variable1
{
public:
  inline static bool GetValue() { return BitFields.Value1; }
  inline static void SetValue( bool AValue ) { BitFields.Value1 = AValu
e; }

};

class Variable2
{
public:
  inline static bool GetValue() { return BitFields.Value2; }
  inline static void SetValue( bool AValue ) { BitFields.Value2 = AValu
e; }

};

class Variable3
{
public:
  inline static bool GetValue() { return BitFields.Value3; }
  inline static void SetValue( bool AValue ) { BitFields.Value3 = AValu
e; }

};

class Variable4
{
public:
  inline static bool GetValue() { return BitFields.Value4; }
  inline static void SetValue( bool AValue ) { BitFields.Value4 = AValu
e; }

};

class Variable5
{
public:
  inline static bool GetValue() { return BitFields.Value5; }
  inline static void SetValue( bool AValue ) { BitFields.Value5 = AValu
e; }

};
```

```
class Variable6
{
public:
  inline static bool GetValue() { return BitFields.Value6; }
  inline static void SetValue( bool AValue ) { BitFields.Value6 = AValu
e; }

};

class Variable7
{
public:
  inline static bool GetValue() { return BitFields.Value7; }
  inline static void SetValue( bool AValue ) { BitFields.Value7 = AValu
e; }

};

class Variable8
{
public:
  inline static bool GetValue() { return BitFields.Value8; }
  inline static void SetValue( bool AValue ) { BitFields.Value8 = AValu
e; }

};

class Variable9
{
public:
  inline static bool GetValue() { return BitFields.Value9; }
  inline static void SetValue( bool AValue ) { BitFields.Value9 = AValu
e; }

};

class Variable10
{
public:
  inline static bool GetValue() { return BitFields.Value10; }
  inline static void SetValue( bool AValue ) { BitFields.Value10 = AVal
ue; }

};

class Variable11
{
```

```
public:
  inline static bool GetValue() { return BitFields.Value11; }
  inline static void SetValue( bool AValue ) { BitFields.Value11 = AVal
ue; }

};

class Variable12
{
public:
  inline static bool GetValue() { return BitFields.Value12; }
  inline static void SetValue( bool AValue ) { BitFields.Value12 = AVal
ue; }

};

} // ComponentVariables

// Arduino Constant Declarations

namespace VisuinoConstants
{
  constexpr PROGMEM const uint8_t ArrayValue0[] = { 51,51,51 };
} // VisuinoConstants

// Pin Call Declarations

namespace PinCalls
{
class PinCallerReceive0
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive1
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive2
{
public:
  void Notify( void *_Data );

};
class PinCallerReceive3
```

```
{
public:
  void Notify( void *_Data );

};
} // PinCalls

// Call Chains

namespace CallChains
{
class CheckPopulated1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( bool & AIsPopulated );

};
class GetData1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( uint8_t *& ADataStart, uint32_t & ADataIndex, uint8
_t & AOffset );

};
class GetSize1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( int32_t AIndex, bool & AAlligned, int & AResult );

};
class Expand1
{
public:
  inline static uint32_t Count() { return 1; }
  static void Call( int32_t AIndex, uint8_t * const & AInBuffer, uint8_
t * const & AOutBuffer, int & ASize, bool & AResult );

};
class GetValue1
{
public:
  inline static uint32_t Count() { return 0; }
  static void Call( int32_t AIndex, uint8_t & AValue );

};
```

```
class ApplyValues1
{
public:
  inline static uint32_t Count() { return 0; }
  static void Call( uint8_t * AValue );

};
} // CallChains

// Arduino Board Declarations

namespace BoardDeclarations
{
namespace Types
{
typedef Mitov::ArduinoDigitalOutputChannel<
  Mitov::ConstantProperty<22, bool, false>, // IsAnalog
  Mitov::ConstantProperty<24, bool, false>, // IsCombinedInOut
  Mitov::ConstantProperty<21, bool, false>, // IsOpenDrain
  Mitov::ConstantProperty<26, bool, false>, // IsOutput
  Mitov::ConstantProperty<20, bool, false>, // IsPullDown
  Mitov::ConstantProperty<5, bool, false >, // IsPullUp
  Mitov::ConstantProperty<23, bool, false>, // IsRawInput
  Mitov::DigitalPin_EmbeddedPinImplementation<6, ::PinCalls::PinCallerR
eceive0 >, // OutputPin
  2 // PIN
  > ArduinoDigitalChannel_2;
} // Types

namespace Instances
{
Types::ArduinoDigitalChannel_2 ArduinoDigitalChannel_2;
} // Instances

namespace Types
{
typedef Mitov::ArduinoDigitalChannel<
  Mitov::ConstantProperty<51, bool, false>, // InitialValue
  Mitov::ConstantProperty<22, bool, false>, // IsAnalog
  Mitov::ConstantProperty<24, bool, false>, // IsCombinedInOut
  Mitov::ConstantProperty<21, bool, false>, // IsOpenDrain
  Mitov::ConstantProperty<25, bool, true>, // IsOutput
  Mitov::ConstantProperty<20, bool, false>, // IsPullDown
  Mitov::ConstantProperty<5, bool, false >, // IsPullUp
  Mitov::ConstantProperty<23, bool, true>, // IsRawInput
  Mitov::DigitalPin_NoImplementation<6 >, // OutputPin
  13 // PIN
  > ArduinoDigitalChannel_13;
```

```
} // Types

namespace Instances
{
Types::ArduinoDigitalChannel_13 ArduinoDigitalChannel_13;
} // Instances

namespace Types
{
typedef Mitov::SerialPort<
  SERIAL_TYPE, // 0_T_TYPE
  Serial, // 1_C_OBJECT
  Mitov::ConstantProperty<4, uint32_t, 0 >, // AfterSendingDelay
  Mitov::ConstantProperty<7, uint32_t, 8 >, // DataBits
  Mitov::ConstantProperty<2, bool, true >, // Enabled
  Mitov::ConstantProperty<12, uint32_t, 0 >, // FEndTime
  Mitov::ConstantProperty<10, bool, false >, // FSending
  Mitov::GenericPin_NoImplementation<5 >, // OutputPin
  Mitov::ConstantProperty<9, Mitov::TArduinoStandardSerialParity, Mitov
::spNone >, // Parity
  Mitov::DigitalPin_NoImplementation<3 >, // SendingOutputPin
  Mitov::ConstantProperty<6, uint32_t, 9600 >, // Speed
  Mitov::ConstantProperty<8, uint32_t, 1 > // StopBits
   > SerialPort0;
} // Types

namespace Instances
{
Types::SerialPort0 SerialPort0;
} // Instances

namespace Types
{
typedef Mitov::ArduinoSerialInput<BoardDeclarations::Types::SerialPort0
, BoardDeclarations::Instances::SerialPort0, bool> SerialPort0_Input_IO
WBoolStream_1;
} // Types

namespace Instances
{
Types::SerialPort0_Input_IOWBoolStream_1 SerialPort0_Input_IOWBoolStrea
m_1;
} // Instances

namespace Types
{
```

```
typedef Mitov::ArduinoSerialInput_Binary<BoardDeclarations::Types::Seri
alPort0, BoardDeclarations::Instances::SerialPort0, Mitov::TArray<uint8
_t>> SerialPort0_Input_IOWByteStream_1;
} // Types

namespace Instances
{
Types::SerialPort0_Input_IOWByteStream_1 SerialPort0_Input_IOWByteStrea
m_1;
} // Instances

} // BoardDeclarations

// Declarations

namespace Declarations
{
namespace Types
{
typedef Mitov::Button<
  Mitov::ConstantProperty<5, uint32_t, 50 >, // DebounceInterval
  Mitov::TypedVariable<6, bool, ::ComponentVariables::Variable2 >, // F
LastValue
  Mitov::TypedVariable<8, bool, ::ComponentVariables::Variable3 >, // F
Value
  Mitov::DigitalPin_EmbeddedPinImplementation_ChangeOnly<3, ::PinCalls:
:PinCallerReceive1, Mitov::TypedVariable<0, bool, ::ComponentVariables:
:Variable1 > > // OutputPin
   > Button1;
} // Types

namespace Instances
{
Types::Button1 Button1;
} // Instances

namespace Types
{
typedef Mitov::TFlipFlop<
  Mitov::TypedVariable<10, bool, ::ComponentVariables::Variable6 >, //
FToggleValue
  Mitov::TypedVariableValue<5, bool, ::ComponentVariables::Variable5, f
alse >, // InitialValue
  Mitov::DigitalPin_NoImplementation<4 >, // InvertedOutputPin
  Mitov::DigitalPin_EmbeddedPinImplementation_ChangeOnly<3, ::PinCalls:
:PinCallerReceive2, Mitov::TypedVariable<0, bool, ::ComponentVariables:
:Variable4 > > // OutputPin
   > TFlipFlop1;
```

```
} // Types

namespace Instances
{
Types::TFlipFlop1 TFlipFlop1;
} // Instances

namespace Types
{
typedef Mitov::Packet<
  1, // COUNT_Elements
  Mitov::NestedProperty<12, Mitov::PacketChecksumElement<
    Mitov::ConstantProperty<11, bool, true > // Enabled
     > >, // Checksum
  Mitov::EmbeddedCallChain<CallChains::CheckPopulated1 >, // Elements_C
heckPopulated
  Mitov::EmbeddedCallChain<CallChains::Expand1 >, // Elements_Expand
  Mitov::EmbeddedCallChain<CallChains::GetData1 >, // Elements_GetData
  Mitov::EmbeddedCallChain<CallChains::GetSize1 >, // Elements_GetSize
  Mitov::TypedVariable<15, bool, ::ComponentVariables::Variable8 >, //
FModified
  Mitov::TypedVariable<13, bool, ::ComponentVariables::Variable7 >, //
FNeedsNewSize
  Mitov::TypedVariable<17, bool, ::ComponentVariables::Variable9 >, //
FRefreshed
  Mitov::NestedProperty<9, Mitov::TArduinoHeadMarkerBinaryPacketElement
<
    Mitov::ConstantPropertyArray<8, uint8_t, uint8_t, ::VisuinoConstant
s::ArrayValue0, 3 >, // Bytes
    Mitov::EmbeddedCallChain<CallChains::ApplyValues1 >, // Bytes_Apply
Values
    Mitov::EmbeddedCallChain<CallChains::GetValue1 > // Bytes_GetValue
     > >, // HeadMarker
  Mitov::ConstantProperty<5, bool, false >, // OnlyModified
  Mitov::TypedPin_EmbeddedPinImplementation<3, ::PinCalls::PinCallerRec
eive3, Mitov::TArray<uint8_t> > // OutputPin
   > Packet1;
} // Types

namespace Instances
{
Types::Packet1 Packet1;
} // Instances

namespace Types
{
typedef Mitov::PacketDigitalBinaryElement<
  Declarations::Types::Packet1, // 0_TYPE_OWNER
```

```
  Declarations::Instances::Packet1, // 1_NAME_OWNER
  Mitov::TypedVariable<5, bool, ::ComponentVariables::Variable11 >, //
FBoolValue
  Mitov::TypedVariable<3, bool, ::ComponentVariables::Variable10 > // F
Populated
   > TArduinoDigitalBinaryPacketElement1;
} // Types

namespace Instances
{
Types::TArduinoDigitalBinaryPacketElement1 TArduinoDigitalBinaryPacketE
lement1 = Types::TArduinoDigitalBinaryPacketElement1( false );
} // Instances

} // Declarations

// Type Converters

namespace TypeConverters
{
Mitov::Convert_BinaryToClock<Mitov::TypedVariable<0, bool, ::ComponentV
ariables::Variable12 >> Converter0;
} // TypeConverters

// Call Chains

namespace CallChains
{
void CheckPopulated1::Call( bool & AIsPopulated )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.CheckPop
ulated( AIsPopulated );
}

void GetData1::Call( uint8_t *& ADataStart, uint32_t & ADataIndex, uint
8_t & AOffset )
{
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.GetData(
 ADataStart, ADataIndex, AOffset );
}

void GetSize1::Call( int32_t AIndex, bool & AAlligned, int & AResult )
{
  switch( AIndex )
  {
    case 0: Declarations::Instances::TArduinoDigitalBinaryPacketElement
1.GetSize( AAlligned, AResult ); break;
  }
```

```
}

void Expand1::Call( int32_t AIndex, uint8_t * const & AInBuffer, uint8_
t * const & AOutBuffer, int & ASize, bool & AResult )
{
  switch( AIndex )
  {
    case 0: Declarations::Instances::TArduinoDigitalBinaryPacketElement
1.Expand( AInBuffer, AOutBuffer, ASize, AResult ); break;
  }
}

void GetValue1::Call( int32_t AIndex, uint8_t & AValue )
{
}

void ApplyValues1::Call( uint8_t * AValue )
{
}

} // CallChains

// Pin Call Declarations

namespace PinCalls
{
void PinCallerConverterReceive1( void *_Data );
} // PinCalls

// Pin Call Implementations

namespace PinCalls
{
void PinCallerReceive0::Notify( void *_Data )
{
  BoardDeclarations::Instances::SerialPort0_Input_IOWBoolStream_1.Input
Pin_o_Receive( _Data );
  Declarations::Instances::Button1.InputPin_o_Receive( _Data );
}

void PinCallerReceive1::Notify( void *_Data )
{
  TypeConverters::Converter0.Convert( _Data, PinCallerConverterReceive1
 );
}

void PinCallerConverterReceive1( void *_Data )
{
```

```
  Declarations::Instances::TFlipFlop1.ClockInputPin_o_Receive( _Data );
}
void PinCallerReceive2::Notify( void *_Data )
{
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.DigitalInputPi
n_o_Receive( _Data );
  Declarations::Instances::TArduinoDigitalBinaryPacketElement1.InputPin
_o_Receive( _Data );
}

void PinCallerReceive3::Notify( void *_Data )
{
  BoardDeclarations::Instances::SerialPort0_Input_IOWByteStream_1.Input
Pin_o_Receive( _Data );
}

} // PinCalls

namespace ComponentsHardware
{
void SystemUpdateHardware()
{
}
} // ComponentsHardware

//The setup function is called once at startup of the sketch
void setup()
{
  BoardDeclarations::Instances::SerialPort0.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemInit();
  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemStart();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemStart();
  Declarations::Instances::Button1.SystemStart();
  Declarations::Instances::TFlipFlop1.SystemStart();
  Declarations::Instances::Packet1.SystemStart();
}

// The loop function is called in an endless loop
void loop()
{
  BoardDeclarations::Instances::ArduinoDigitalChannel_2.SystemLoopBegin
();
  BoardDeclarations::Instances::ArduinoDigitalChannel_13.SystemLoopBegi
n();
  Declarations::Instances::Button1.SystemLoopBegin();
  Declarations::Instances::Packet1.SystemLoopBegin();
}
```

7.ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Πολυζάκης Α., 2019, 'Στοιχεία εφαρμοσμένης θερμοδυναμικής' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ. 1-24.

[2] Πολυζάκης Α., 2019, 'Ηλιακή ενέργεια' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.422-472

[3] *Επαναστατική εφεύρεση για αποθήκευση ηλιακής ενέργειας* (10 Νοεμβρίου 2019) [Online], Διαθέσιμο από: https://www.businessdaily.gr/tehnologia/4350_epanastatiki-efeyresi-gia-apothikeysi-iliakis-energeias [21 Οκτωβρίου 2022]

[4] Πολυζάκης Α.,2019, 'Αιολική ενέργεια' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ. 473-514.

[5] Πολυζάκης Α., 2019, 'Υδάτινη ενέργεια' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.515-586.

[6] Πολυζάκης Α., 2019, 'Γεωθερμική ενέργεια' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.587-622.

[7] Πολυζάκης Α., 2019, 'Κυψέλες καυσίμου' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.623-649

[8] Πολυζάκης Α., 2019, 'Βιοενέργεια' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.650-668.

[9] Mitali J., Dhinakaran S., Mohama A.A., 2022, 'Energy storage systems: A review', Energy Storage and Saving

[10] Πολυζάκης Α., 2019, 'Εισαγωγή' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ. 415-421.

[11] Πολυζάκης Α., 2019, 'Συμπαραγώγή – τριπαραγωγή' στο *Ενέργεια, Περιβάλλον και Αειφόρος Ανάπτυξη,* 1η έκδοση, Ρ.Η.C. , Πτολεμαΐδα, σ.157-180.

[12] 2021, 'Renewable Energy Source for Combined Heat and Power (RESCHP)', Research and Innovation actions

[13] Pittas N., Gkialas I., Georgiou D. & Moutsios V., 2021, 'Energy balance analysis between energy consumed for the compression of atmospheric air up to 200bar and the energy produced from its expansion down to atmospheric pressure on Nicholas Pittas' patent about storage and continuous production energy system', International Journal of Engineering Technologies and Management Research, p. 77-89

[14] Κρικέλης Ν.Ι., 2014, 'Εισαγωγή στα συστήματα ελέγχου' στο *Εισαγωγή στον Αυτόματο Έλεγχο,* 4η έκδοση, Εκδόσεις Συμμετρία, Αθήνα, σ. 9-33.

[15] Κορρές, Γ., 2010, 'Συστήματα Τηλεμετρήσεων – SCADA', σημειώσεις μαθήματος που διανεμήθηκαν στην Εποπτεία και Διαχείριση Ενεργειακών Συστημάτων στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου, Αθήνα, Οκτώβριος 2010.

[16] *Arduino Uno* (27 Σεπτεμβρίου 2022) [Online]. Διαθέσιμο από: https://en.wikipedia.org/wiki/Arduino_Uno [10 Οκτωβρίου 2022]

[17] *DHT11 Humidity & Temperature Sensor* (2014) [Online]. Διαθέσιμο από: https://datasheetspdf.com/pdf/785590/D-Robotics/DHT11/1 [10 Οκτωβρίου 2022]