ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΛΕΙΔΑΡΙΑΣ ΜΕ ΜΙΚΡΟΕΛΕΓΚΤΗ ΓΙΑ ΑΠΟΜΑΚΡΥΣΜΕΝΟ ΕΛΕΓΧΟ ΠΟΡΤΑΣ ΚΑΤΟΙΚΙΑΣ

Βασίλειος Θωμάς Αργυρόπουλος

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ

Αλέξανδρος Καλαράκης

Καθηγητής Εφαρμογών

ΠΑΤΡΑ 2016

ΠΡΟΛΟΓΟΣ

Η καθημερινότητα μας πλέον περικλείεται από τεχνολογία η οποία έχει ως σκοπό να κάνει την ζωή μας πιο εύκολη. Πιο συγκεκριμένα περιλαμβάνει ολοένα και περισσότερες “έξυπνες’’ συσκευές οι οποίες αυτοματοποιούν διάφορες διαδικασίες. Από θερμοστάτες όπου παρακολουθούν την θερμοκρασία του σπιτιού μας και μας ειδοποιούν σε περίπτωση αρκετά χαμηλής θερμοκρασίας ώστε να ενεργοποιηθεί η κεντρική θέρμανση, μέχρι κάμερες ασφαλείας και συναγερμοί που μας ειδοποιούν σε περίπτωση ανίχνευσης κίνησης.

Επιβάλλεται λοιπόν η ανάπτυξη εφαρμογών και μηχανισμών για την αξιοποίηση των δυνατοτήτων της σύγχρονης τεχνολογίας. Στην περίπτωση μας η εφαρμογή αυτή έγινε με την δημιουργία ενός συνόλου για την ευκολότερη είσοδο στην οικία μας αλλά και απομακρυσμένο έλεγχο αυτού μέσω του έξυπνου κινητού μας τηλεφώνου, ή μέσω tablet με την αντίστοιχη εφαρμογή Android.

Υπεύθυνη Δήλωση Σπουδαστή: Ο κάτωθι υπογεγραμμένος σπουδαστής έχω επίγνωση των συνεπειών του Νόμου περί λογοκλοπής και δηλώνω υπεύθυνα ότι είμαι συγγραφέας αυτής της Πτυχιακής Εργασίας, έχω δε αναφέρει στην Βιβλιογραφία μου όλες τις πηγές τις οποίες χρησιμοποίησα και έλαβα ιδέες ή δεδομένα. Δηλώνω επίσης ότι, οποιοδήποτε στοιχείο ή κείμενο το οποίο έχω ενσωματώσει στην εργασία μου προερχόμενο από Βιβλία ή άλλες εργασίες ή το διαδίκτυο, γραμμένο ακριβώς ή παραφρασμένο, το έχω πλήρως αναγνωρίσει ως πνευματικό έργο άλλου συγγραφέα και έχω αναφέρει ανελλιπώς το όνομά του και την πηγή προέλευσης.

Ο σπουδαστής

(Ονοματεπώνυμο)

……

(Υπογραφή)

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη πτυχιακή έχει ως σκοπό την ενσωμάτωση της σύγχρονης τεχνολογίας στην καθημερινή μας ζωή και συγκεκριμένα στην απλοποίηση της διαδικασίας εισόδου στην οικία μας με απουσία του κλασσικού κλειδιού αλλά με ηλεκτρονική υποβοήθηση.

Πιο συγκεκριμένα η εργασία περιλαμβάνει την διαδικασία προκειμένου να κατασκευαστεί ο μηχανισμός ο οποίος θα ενσωματωθεί πάνω στην πόρτα οποιασδήποτε οικίας. Ο μηχανισμός περιλαμβάνει την ικανότητα ελέγχου μέσω εφαρμογής android, χειροκίνητου κουμπιού εσωτερικά της πόρτας αλλά και εξωτερικά μέσω του αναγνώστη RFID και της αντίστοιχης παθητικής ετικέτας (μπρελόκ ή κάρτα).

Σε περίπτωση απώλειας ισχύος του μηχανισμού δίνεται η δυνατότητα εισαγωγής κλασσικού κλειδιού από την εξωτερική πλευρά της πόρτας αλλά και ειδοποίηση του κατόχου μέσω της εφαρμογής android στο κινητό μας τηλέφωνο ή στο tablet όταν η ισχύς της συσκευής πέσει σε χαμηλά επίπεδα. Παράλληλα, οι εσωτερικές μπαταρίες μπορούν να τροφοδοτηθούν από εξωτερικό φορτιστή ώστε να αποκτήσουν την απαραίτητη ισχύ για να επανέλθει ο μηχανισμός στην φυσιολογική του λειτουργία.

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

Wiring = Γλώσσα προγραμματισμού βασισμένη σε C++

USB = Universal Serial Bus – σύστημα διάυλου επικοινωνίας

Kickstarter = Πλατφόρμα συγκέντρωσης χρημάτων για Projects από όλο τον πλανήτη

ABS = Acrylonitrile butadiene styrene – πλαστικό ακρυλονιτρίλιο βουταδιένιο στυρολίου

DDOS = Distributed Denial of Service

Συντελεστής κεφαλής - Coefficient addendum = συντελεστής μετατροπής για μορφοποίηση δοντιών ώστε να εφαρμόζουν καλύτερα.

ΕΙΣΑΓΩΓΗ

Εφόσον η τεχνολογία εξελίσσεται, όχι μόνο διαρκώς ημέρα με την ημέρα, αλλά και με αλματώδη βήματα, παρατηρούμε ότι την χρησιμοποιούμε ολοένα και περισσότερο στην καθημερινή μας ζωή. Είτε στην επικοινωνία, είτε στην μετακίνηση μας είτε σε οποιοδήποτε άλλο παράγοντα της ζωής μας.

Ως αποτέλεσμα έφθασε η στιγμή όπου η τεχνολογία θα εφαρμοσθεί και στον χώρο στον οποίο διαμένουμε. Δηλαδή στο σπίτι μας, όπου θα λάβει χώρα σε διάφορα σημεία, από “έξυπνες” πρίζες και θερμοστάτες μέχρι την ενεργοποίηση και απενεργοποίηση φωτισμού, από την άνεση και ευκολία του κινητού μας τηλεφώνου.

Καταλήξαμε λοιπόν στην πλέον διαδεδομένη έννοια στον χώρο της τεχνολογίας γνωστή και ως IOT(Internet Of Things) η πιο συγκεκριμένα στην διαμόρφωση ενός έξυπνου σπιτιού (Smart Home).

Έτσι στην συγκεκριμένη πτυχιακή θα κατασκευάσουμε άλλο ένα εξάρτημα το οποίο θα μπορούσε εύκολα να προστεθεί στο “οπλοστάσιο” ενός έξυπνου σπιτιού και πιο συγκεκριμένα στον σχεδιασμό και την κατασκευή μίας ηλεκτρονικής κλειδαριάς η οποία προσαρμόζεται στην εσωτερική πλευρά της πόρτας και με ένα μηχανισμό μπορεί να περιστρέψει το κλειδί είτε για να κλειδώσει την πόρτα είτε για να την ξεκλειδώσει. Παράλληλα, από την εξωτερική πλευρά έχει τοποθετηθεί ένας αναγνώστης RFID για να επιτρέπεται η ανάγνωση μιας παθητικής ετικέτας η οποία ενεργοποιεί τον μηχανισμό και με τη σειρά του μας επιτρέπει την είσοδο στον χώρο όπου έχει τοποθετηθεί ο μηχανισμός.

Στη συνέχεια, θα αναλύσουμε οτιδήποτε έχει χρησιμοποιηθεί για την κατασκευή του μηχανισμού αυτού με τη μεγαλύτερη δυνατόν λεπτομέρεια προκειμένου να είναι εύκολη και κατανοητή η ανάγνωση της πτυχιακής εργασίας.

1 Εισαγωγή στο Arduino

Εικόνα 1. 1.1 *: Arduino Logo [1]*

Το Project Arduino ξεκίνησε όταν ο κολομβιανός μαθητής Hernando Barragán δημιούργησε την πλατφόρμα προγραμματισμού Wiring ως διπλωματική για το μάστερ του το 2004 στην πόλη Ivrea στην Ιταλία. Οι επιτηρητές του ήταν ο Massimo Banzi και ο Casey Reas. Ο σκοπός της διπλωματικής του ήταν να δημιουργήσει χαμηλού κόστους εργαλεία για άτομα που δεν ήταν απαραίτητα μηχανικοί για να τα χρησιμοποιήσουν σε διάφορες εργασίες (project).

Η πλατφόρμα Wiring αποτελούνταν από μια πλακέτα pcb με έναν μικροελεγκτή Atmega168, ένα ενσωματωμένο περιβάλλον ανάπτυξης λογισμικού (Integrated Development Environment) βασισμένο στο processing και έτοιμες εντολές «φορτωμένες» από ηλεκτρονικές βιβλιοθήκες για να προγραμματίζεται εύκολα ο μικροελεγκτής. Τελικά το 2005 προστέθηκε υποστήριξη και για τον φθηνότερο μικροελεγκτή Atmega8 στην πλατφόρμα Wiring, αλλά αντί να συνεχιστεί η δουλειά στην πλατφόρμα Wiring, αντέγραψαν όλο τον κώδικα της πλατφόρμας σε ένα ξεχωριστό κομμάτι και άρχισαν να το τρέχουν σαν ξεχωριστό project με το ονομα Arduino.

Πιο συγκεκριμένα το Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες. Το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

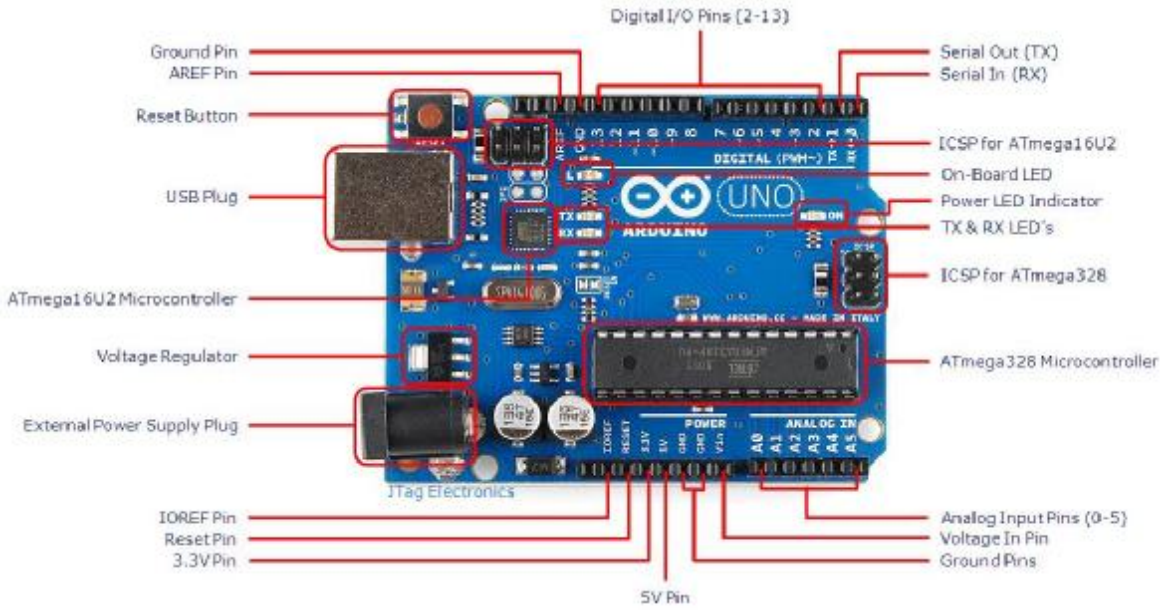
Η πιο κοινή έκδοση του arduino είναι το Arduino Uno, συνήθως όταν κάποιος κάνει αναφορά στο Arduino εννοεί την έκδοση αυτή. Έχει διαδοθεί τόσο πολύ καθώς είναι μια πολύ καλή επιλογή για αρχάριους. Υπάρχουν διάφορες εκδοχές του Arduino Uno, η πιο πρόσφατη είναι η Rev3 ή R3.

Το Arduino Uno είναι ένας μικροελεγκτής βασισμένος στον ATmega328. Έχει 14 ψηφιακά pins εισόδου/εξόδου, από τα οποία τα 6 μπορούν να αξιοποιηθούν ως εξόδοι PWM

(Pulse Width Modulation η Διαμόρφωση εύρους παλμών), 6 αναλογικές εισόδους, ένα κεραμικό resonator, μια usb υποδοχή, μια υποδοχή ρεύματος, ενα ICSP Header και ένα κουμπί reset. Γενικά περιέχει οτιδήποτε χρειάζεται προκειμένου να υποστηριχθεί ο μικροελεγκτής, απλά κάνουμε την σύνδεση στον υπολογιστή με ένα USB καλώδιο και είναι έτοιμο για λειτουργία.

Περιληπτικά Στοιχεία

Μικροελεγκτής	ATmega328
Volt Λειτουργίας	5V
Volt Εισόδου (Προτεινόμενα)	7-12V
Volt Εισόδου (Μέγιστα)	6-20V
Ψηφιακά Pins	14 (Οπου τα 6 παρέχουν PWM εξόδους)
Αναλογικά pin εισόδου	6
DC ρεύμα ανά Pin	40 mA
DC ρεύμα για 3. 3V Pin	50 mA
Μνήμη Flash	32 KB (ATmega328) όπου τα 0. 5 χρησιμοποιούνται απο τον Bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Ταχύτητα Μικροελεγκτή	16 MHz
Μήκος	68. 6 mm
Πλάτος	53. 4 mm



***Εικόνα 1.2**: Arduino R3 Διάγραμμα Υποδοχών [2]*

Επεξήγηση Pins Arduino Uno

- Ground Pin = Pin Γείωσης
- AREF Pin = Pin αναλογικής αναφοράς τάσης
- Reset Button = Κουμπί επανεκκίνησης
- USB Plug = Υποδοχή USB
- Atmega16U2 Microcontroller = Μικροελεγκτής Atmega16U2
- Voltage Regulator = Ρυθμιστής τάσης
- External Power Supply Plug = Υποδοχή εξωτερικής τροφοδοσίας
- IOREF Pin = Voltage Reference Pin
- Reset Pin = Pin Επανεκκίνησης
- 3. 3v Pin = Pin παροχής 3. 3v
- 5v Pin = Pin παροχής 5v
- Voltage In Pin = Pin εισόδου τροφοδοσίας
- Analog Input Pins (0-5) = Αναλογικά pins εισόδου
- ATmega328 Microcontroller = Μικροελεγκτής ATmega328
- ISCP for ATmega328 = Pins Προγραμματισμού ATmega328
- TX & RX LED'S = Λαμπάκια LED σειριακής επικοινωνίας Αποστολής & Λήψης
- Power Led Indicator = Ένδειξη λειτουργίας
- On – Board LED = Λαμπάκι LED μητρικής
- ISCP for Atmega16U2 = Pins Προγραμματισμού Atmega16U2
- Serial In(RX) = Pin λήψης σειριακής επικοινωνίας
- Serial Out (TX) = Pin αποστολής σειριακής επικοινωνίας
- Digital I/O Pins (2-13) = Ψηφιακα pin γενικού σκοπού (2-13)

1.1 Τροφοδοσία Arduino Uno

Το Arduino Uno μπορεί να τροφοδοτηθεί μέσω της USB υποδοχής με ένα καλώδιο το οποίο μπορεί να έρχεται από τον υπολογιστή η από έναν USB φορτιστή ο οποίος είναι συνδεδεμένος στην πρίζα του σπιτιού σας. Η δεύτερη μέθοδος τροφοδοσίας είναι μέσω της υποδοχής ρεύματος (External Power Supply Plug) με έναν μετασχηματιστή 12V, ή μέσω μπαταριών οι οποίες αποδίδουν τουλάχιστον 6V.

1.1.1 Pin Τροφοδοσίας

Voltage In Pin – Μέσω του Pin αυτού μπορούμε να δώσουμε ενέργεια στο Board (7-12V) η να πάρουμε αν η παροχή ενέργειας γίνεται από την USB σύνδεση ή την υποδοχή ρεύματος (External Power Supply Plug).

5V Pin – Το pin αυτό μας δίνει τάση 5V μέσω του Regulator της πλακέτας.

3.3V Pin – Το pin αυτό μας δίνει τάση 3.3V μέσω του Regulator της πλακέτας. Μέγιστο ρεύμα 50mA

Ground Pins – Υπάρχουν διάφορα pins γείωσης πάνω στην πλακέτα. Οποιοδήποτε από αυτά μπορεί να χρησιμοποιηθεί για γείωση.

IOREF Pin – Το pin αυτό μας δίνει την τάση αναφοράς με την οποία δουλεύει ο μικροελεγκτής

1.1.2 Ρυθμιστής τάσης

Ο ρυθμιστής τάσης ρυθμίζει την τάση η οποία εισέρχεται στο Board. Οποιαδήποτε τάση παραπάνω από την επιτρεπόμενη δεν περνάει χάρη στον ρυθμιστή τάσης ώστε να μην βλάψει το κύκλωμα. Βέβαια, έχει τα όρια του και για αυτό δεν πρέπει να ξεπεράσουμε ποτέ τα 20V ως τάση εισόδου.

1.2 Μνήμη και αποθήκευση δεδομένων

Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων :

Flash - 32k bytes (όπου τα 0.5k χρησιμοποιούνται για τον bootloader)

SRAM - 2k bytes

EEPROM - 1k byte

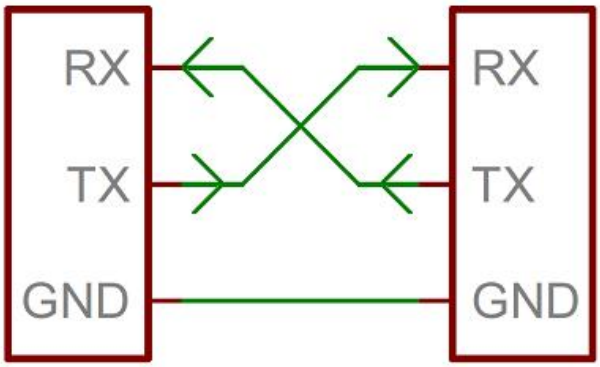
Flash memory (χώρος του προγράμματος), είναι εκεί όπου αποθηκεύεται το πρόγραμμα (sketch) που ανεβάζουμε

SRAM (static random access memory ή ωφέλιμη μνήμη) είναι εκεί όπου το πρόγραμμα αποθηκεύει και τροποποιεί μεταβλητές.

EEPROM είναι μνήμη όπου μπορούμε να αποθηκεύσουμε πληροφορίες για μεγάλο χρονικό διάστημα.

Η μνήμη Flash και η μνήμη EEPROM δεν είναι ασταθής δηλαδή οι πληροφορίες συγκρατώνται ακόμα και όταν το Uno κλείσει. Η μνήμη SRAM όμως είναι ασταθής και όταν υπάρξει απώλεια τροφοδοσίας στο Uno η γίνει reset όλα τα δεδομένα χάνονται.

1.3 Επικοινωνία με τον υπολογιστή και περιφερειακά



***Εικόνα 1.3** : Απεικόνιση σειριακής Επικοινωνίας [3]*

Το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.

Όπως θα δούμε και αργότερα εξαρτήματα και πλακέτες που δεν έχουν υποδοχή USB χρησιμοποιούν τα pins RX και TX για να επικοινωνήσουν με έναν υπολογιστή όπου από την πλευρά του υπολογιστή χρησιμοποιούμε ένα USB-To-Serial Adapter για να συνδεθούν τα pins στο εξάρτημα.

Κάθε ένα από τα 14 ψηφιακά pin στο Uno μπορούν να χρησιμοποιηθούν είτε σαν έξοδος (output) ή σαν είσοδος(input). Μπορούν να αποδώσουν τάση 5V και να παρέχουν ή να λαμβάνουν 40mA ρεύμα. Ως ψηφιακή έξοδος καθορίζεται ότι ένα από αυτά τα pin μπορεί να τεθεί από το πρόγραμμά σε κατάσταση HIGH ή LOW, οπότε το Arduino θα ξέρει αν πρέπει να διοχετεύσει ή όχι ρεύμα στο συγκεκριμένο pin. Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές εισοδους/εξοδους έχουν και δεύτερη λειτουργία.

Serial Out (TX) & Serial In (RX) – χρησιμοποιούνται όπως προαναφέρθηκε για την λήψη (RX) και την αποστολή (TX) σειριακών δεδομένων. Επιπλέον τα Pins αυτά είναι συνδεδεμένα με τα αντίστοιχα ποδαράκια του chip ATmega16U2 USB-to-TTL, που είναι υπεύθυνος για το USB και την σειριακή επικοινωνία.

PWM – Pulse Width Modulation η Διαμόρφωση εύρους παλμών. Τα Pins αυτά μπορούν εκτός από το να λειτουργήσουν ως κανονικά pin αλλά και να προσομοιώσουν αναλογική έξοδο π. χ. για να αναβοσβήνουν ένα LED με διαφορετική ένταση

SPI – Pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). SPI σημαίνει Serial Peripheral Interface – Σειριακό περιφερειακό περιβάλλον. Τα Pins αυτά υποστηρίζουν SPI επικοινωνία χρησιμοποιώντας την βιβλιοθήκη SPI.

Analog Input Pins – Pins αναλογικής εισόδου (A0-A5) όπου έχουν 10 bit διακριτικής ικανότητας (1024 διαφορετικές τιμές). Τα pins αυτά μπορούν να διαβάσουν το σήμα από αναλογικούς αισθητήρες και να το μετατρέψουν σε ψηφιακό. Διαβάζουν απο 0 εως 5 Volt.

Reset Pin – Αν συνδεθεί με GND δηλαδή γίνει LOW θα επανεκκινήσει το Arduino

1.4 Ενδείξεις LED

Power LED Indicator – Το LED αυτό είναι η ένδειξη λειτουργίας του Uno. Όταν είναι ενεργοποιημένο σημαίνει ότι το Uno έχει συνδεθεί με μία παροχή ενέργειας.

On-Board LED – Το LED αυτό είναι συνδεδεμένο με το Pin 13. Οταν το Pin είναι HIGH τότε το LED ανάβει, όταν το Pin είναι LOW τότε σβήνει. Συνήθως χρησιμοποιείται για τσεκάρισμα αν υπάρχει πρόβλημα στο board γιατί μερικά Uno έρχονται σεταρισμένα ήδη με το sketch “Blink”.

TX & RX LEDs – Αυτά τα LED αναβοσβήνουν όταν το Uno δέχεται ή στέλνει δεδομένα. Όπως όταν ανεβάζουμε ένα πρόγραμμα στη μνήμη.

1.5 ATmega328 Microcontroller

Ο ATmega328 είναι ο βασικός μικροελεγκτής του Uno και είναι υπεύθυνος για την λειτουργία όλου του Board.

1.6 ATmega16U2 Microcontroller

Ο ATmega16u2 αναλαμβάνει την USB σύνδεση. Το νέο όμως Arduino Uno R3 χρησιμοποιεί τον ATmega16U2 αντί για τον 8U2 ο οποίος βρίσκεται στις παλιότερες εκδόσεις. Ο καινούργιος επιτρέπει μεγαλύτερες ταχύτητες μεταφοράς δεδομένων και μεγαλύτερη μνήμη.

1.7 ICSP Headers

ICSP σημαίνει In-Circuit Serial Programming – Ενδοκυκλωματικός σειριακός προγραμματισμός. Υπάρχουν 2 ICSP κεφαλές στο Board μία για τον Atmega16U2 και μία για τον ATmega328. Αυτές οι κεφαλές χρησιμοποιούνται για να αναβαθμίσουν η να φορτώσουν firmware στους μικροελεγκτές.

1.8 Reset Button

Το κουπί reset έχει ως σκοπό να επανεκκινεί το Arduino Uno. Πατώντας το κουπί reset προσωρινά συνδέεται το reset Pin με ένα Pin γείωσης και έτσι επανεκκινείται ο κώδικας ο οποίος είναι φορτωμένος στο Arduino.

1.9 Παράγωγα του Arduino

Το Arduino περιλαμβάνει και άλλες εκδόσεις πέρα από το Uno οι οποίες έχουν άλλα πλεονεκτήματα και μειονεκτήματα σε σχέση με το Uno. Όπως διαφορετικό μέγεθος λιγότερα ψηφιακά Pin, περιορισμούς στην τροφοδοσία τάσης κ. τ. λ. . Μερικές από τις διαφορετικές εκδόσεις είναι :

- Arduino Mega
- Arduino Mini
- Arduino Due
- Arduino Nano
- Arduino Yun
- Lilypad Arduino

- Arduino Leonardo

Αλλά και Shields τα οποία ενσωματώνονται πάνω στο Uno για να παράσχουν επιπλέον τεχνολογικές ικανότητες όπως

- GPS
- GPRS/EDGE
- Bluetooth
- WIFI
- Shield Ελέγχου κινητήρων
- Shield αποθήκευσης και καταγραφής δεδομένων με μια υποδοχή κάρτας SD και ρολόι πραγματικού χρόνου (RTC) chip
- LCD

1. 10 Λογισμικό και κώδικας

Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino είναι μία εφαρμογή γραμμένη σε Java, που λειτουργεί σε πολλές πλατφόρμες και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Έχει σχεδιαστεί για να εισαγάγει τον προγραμματισμό τόσο στους ειδικούς, όσο και στους νέους που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα με χαρακτηριστικά όπως είναι η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και είναι επίσης σε θέση να μεταγλωττίζει και να φορτώνει προγράμματα στην πλακέτα με ένα μόνο κλικ. Ένα πρόγραμμα ή κώδικας που γράφτηκε για Arduino ονομάζεται *σκίτσο* (sketch).

Τα Arduino προγράμματα είναι γραμμένα σε C ή C++. Το Arduino IDE έρχεται με μια βιβλιοθήκη λογισμικού που ονομάζεται "Wiring", από το πρωτότυπο σχέδιο Wiring, γεγονός που καθιστά πολλές κοινές λειτουργίες εισόδου/εξόδου πολύ πιο εύκολες. Οι χρήστες πρέπει μόνο να ορίσουν δύο λειτουργίες για να δημιουργησουν ένα πρόγραμμα:

-setup():μία συνάρτηση που τρέχει μία φορά στην αρχή του προγράμματος η οποία αρχικοποιεί τις ρυθμίσεις

-loop():μία συνάρτηση που καλείται συνέχεια μέχρι η πλακέτα να απενεργοποιηθεί

Ένα τυπικό πρώτο πρόγραμμα για έναν μικροελεγκτή αναβοσβήνει απλά ένα LED. Στο περιβάλλον του Arduino, ο χρήστης μπορεί να γράψει ένα πρόγραμμα σαν αυτό:

```
#define LED_PIN 13

void setup () {
 pinMode (LED_PIN, OUTPUT); // enable pin 13 for digital output
}

void loop () {
 digitalWrite (LED_PIN, HIGH); // turn on the LED
 delay (1000); // wait one second (1000 milliseconds)
 digitalWrite (LED_PIN, LOW); // turn off the LED
 delay (1000); // wait one second
}
```

Είναι ένα χαρακτηριστικό των περισσότερων πλακετών Arduino ότι έχουν ένα LED και μία αντίσταση φορτίου που συνδέονται μεταξύ του pin 13 και του εδάφους, ένα βολικό χαρακτηριστικό για πολλά απλά τεστ. Ο προηγούμενος κώδικας δεν θα αναγνωριστεί από ένα κανονικό μεταγλωττιστή C++ ως έγκυρο πρόγραμμα, έτσι ώστε όταν ο χρήστης κάνει κλικ στο κουμπί "Upload to I / O board" στο IDE, ένα αντίγραφο του κώδικα θα γραφτεί σε

ένα προσωρινό αρχείο με ένα παραπάνω include στην κορυφή και μία πολύ απλή συνάρτηση main() στο τέλος, για να φτιάξει ένα έγκυρο C++ πρόγραμμα.

Το IDE του Arduino χρησιμοποιεί το GNU toolchain και το AVR Libc για να μεταγλωττίζει προγράμματα και το avrdude για να φορτώνει προγράμματα στην πλακέτα.

Δεδομένου ότι η πλατφόρμα Arduino χρησιμοποιεί Atmel μικροελεγκτές, το περιβάλλον ανάπτυξης της Atmel, το AVR Studio ή το νεότερη έκδοση του Atmel Studio, μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη λογισμικού για το Arduino.

2 Πρόγραμμα διαχείρισης - Blynk

Εικόνα 2.1 : *Blynk Logo [4]*

Το Blynk είναι ένα project το οποίο δίνει την δυνατότητα να χειριστεί κάποιος το ακόλουθο hardware μέσω ενός “έξυπνου” κινητού τηλεφώνου η Tablet μέσω της εφαρμογής της στο Android.

- Arduino Uno
- Raspberry pi
- Esp8266 (Ολες τις εκδοσεις)
- Esp32
- Duemilanove
- Nano
- Mini
- Pro Mini
- Pro Micro
- Mega
- YÚN
- Due
- Intel Edison
- Και άλλα. Η υποστηρίξη διαρκώς επεκτείνεται και σε άλλα προιόντα

Ξεκίνησε το Φεβρουάριο του 2015 μέσω της πλατφόρμας Kickstarter προκειμένου να συγκεντρώσει την χρηματοδότηση που χρειαζόταν για να γίνει πραγματικότητα και την πήρε τον Απρίλιο του 2015 με το ποσό των 49.235 $ από το αρχικό 10.000 $ που χρειαζόταν.

Το περιβάλλον εργασίας της Android εφαρμογής απεικονίζεται στην παρακάτω εικόνα [2.2]

Εικόνα 2.2 *: Περιβάλλον εργασίας Blynk [4]*

Αποτελείται από διάφορα Widget τα οποία μπορούν να τοποθετηθούν στην αρχική οθόνη για να αποτελέσουν το κέντρο ελέγχου για την Iot συσκευή μας. Όπως ένα Arduino Uno. Μερικά από τα Widget περιλαμβάνουν:

- Button (Κουμπί) 
- Slider (Κουμπί ολίσθησης) 
- Timer (μετρητή) 

2.1 Ξεκινώντας με το Blynk

Δημιουργία λογαριασμού Blynk

Μόλις κατεβάσουμε και εγκαταστήσουμε την εφαρμογή Blynk στην συσκευή μας από το Playstore της Google θα χρειαστεί να δημιουργήσουμε ένα λογαριασμό. **Παρατήρηση**: Ο λογαριασμός αυτός είναι διαφορετικός από τους λογαριασμούς που χρησιμοποιούνται στα Forum του Blynk.

Εικόνα 2.3 *: Δημιουργία Λογαριασμού Blynk [4]*

Δημιουργία νέου project

Μόλις συνδεθούμε στο λογαριασμό που μόλις δημιουργήσαμε ξεκινάμε με το να φτιάξουμε ένα καινούργιο project και του δίνουμε ένα όνομα.

Εικόνα 2.4 *: Δημιουργία νέου Project [4]*

Επιλογή Hardware

Επιλέγουμε το Board που χρησιμοποιούμε

Εικόνα 2.5 *: Επιλογή Board [4]*

Auth Token (κλειδί ασφαλείας)

Auth Token είναι ένα μοναδικό αναγνωριστικό το οποίο χρειάζεται για να συνδέσουμε το hardware που χρησιμοποιούμε με τη συσκευή μας. Κάθε νέο project που δημιουργούμε χρειάζεται το αναγνωριστικό αυτό.

Εικόνα 2.6 *: Auth Token [4]*

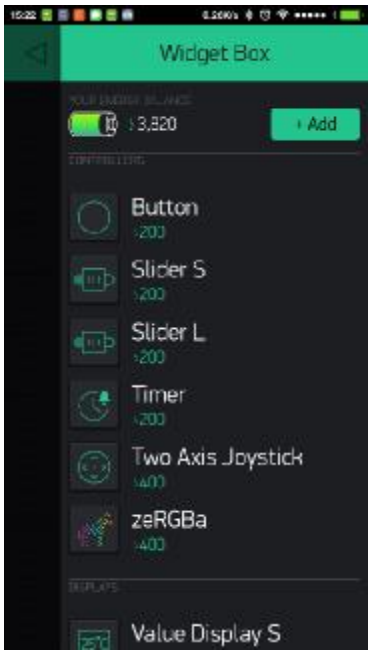
Μπορούμε να το αποστείλουμε και στο Email μας η να πατήσουμε επάνω στο κλειδί και να αντιγραφεί στο clipboard. Έπειτα πατάμε το κουμπί “Create”.

ΠΑΡΑΤΗΡΗΣΗ: Δεν δίνουμε ποτέ το Auth Key σε κανέναν εκτός και αν θέλουμε να μοιραστούμε την είσοδο και τον έλεγχο στο hardware μας.

Εικόνα 2.7 *: Create new project [4]*

Προσθέτοντας το πρώτο widget

Ο καμβάς του project είναι άδειος για να ξεκινήσουμε να τον γεμίζουμε αρχικά θα προσθέσουμε ένα κουμπί για να ελέγχουμε το LED μας. Πατώντας οπουδήποτε στο καμβά θα ανοίξει η πλαινή καρτέλα με τα Widget. Όλα τα διαθέσιμα Widget βρίσκονται εκεί. Επιλέγουμε το κουμπί (button).



Εικόνα 2.8 *: Λίστα Widget [4]*

Εικόνα 2.9 *: Προσθήκη Widget [4]*

Drag-n-Drop – Πατάμε και κρατάμε πατημένο το κουμπί για να το μεταφέρουμε στο καμβά.

Widget ρυθμίσεις – Κάθε widget έχει τις δικιές του ρυθμίσεις. Παταμε επάνω στο widget για να τις εμφανίσουμε.

Εικόνα 2.10 *: Ρυθμίσεις Widget [4]*

Η πιο σημαντική παράμετρος εδώ είναι να ρυθμίσουμε το Pin. Η λίστα των Pin εδώ αντικατοπτρίζει τα φυσικά Pin που έχει το hardware που χρησιμοποιούμε. Πχ. αν το LED που θα χρησιμοποιήσουμε είναι στο ψηφιακό Pin 8 τότε θα επιλέξουμε το 8 και στην εφαρμογή.

Εικόνα 2.11 *: Επιλογή Pin ρυθμίσεις Button Widget [4]*

Τρέχοντας το Project

Όταν τελειώσουμε με τις ρυθμίσεις του widget πατάμε το κουμπί PLAY. Αυτό θα μας μεταφέρει από το EDIT mode στο play mode όπου μπορούμε να επικοινωνήσουμε με το hardware. Όσο είμαστε στο play mode δεν μπορούμε να μεταφέρουμε η να προσθέσουμε καινούργια widget, πρέπει να πατήσουμε το STOP για να πάμε πάλι στο EDIT mode.

Θα λάβουμε ένα μήνυμα που θα μας λέει πως το Arduino Uno είναι εκτός λειτουργίας “Arduino Uno is offline” το οποίο είναι αναμενόμενο καθώς ακόμα δεν έχουν γίνει οι απαραίτητες ρυθμίσεις από την πλευρά του hardware.

Εικόνα 2.12 *: Εισαγωγή Button Widget [4]*

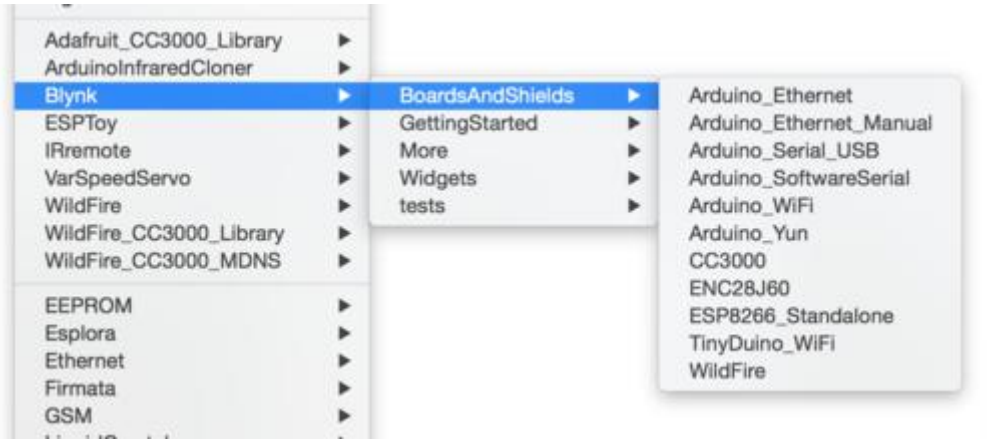
2.2 Blynk στο Hardware

Χρησιμοποιώντας ενα έτοιμο Sketch από την βιβλιοθήκη

Πριν προχωρήσουμε πρέπει να φροντίσουμε ότι έχουμε ήδη εγκαταστήσει την βιβλιοθήκη του Blynk στο Arduino IDE. Για να το κάνουμε μεταβαίνουμε στην ακόλουθη τοποθεσία και την κατεβάζουμε http://docs. blynk. cc/#downloads-blynk-library. Στη συνέχεια την εισάγουμε στο Arduino IDE πηγαίνοντας Sketch>Include Library>add. zip Library. Στο παράθυρο που θα εμφανιστεί επιλέγουμε το αρχείο που κατεβάσαμε πριν από λίγο. Μόλις το επιλέξουμε θα έχουμε εγκαταστήσει την απαραίτητη βιβλιοθήκη.

Υποθέτουμε πως έχουμε το Arduino Uno μαζί με το Ethernet Shield, όπου το Uno είναι συνδεδεμένο στον υπολογιστή μας μέσω καλωδίου USB για να πραγματοποιηθεί η σειριακή σύνδεση.

Για να βρούμε το sketch που θα χρησιμοποιήσουμε για την δοκιμή μας ακολουθούμε την διαδρομή File>Examples>Blynk>Arduino_Ethernet



Εικόνα 2.13 *: Επιλογή Blynk Sketch [4]*

Ανοίγοντας το Sketch θα δούμε τον παρακάτω κώδικα:

```
#define BLYNK_PRINT Serial
#include <SPI. h>
#include <Ethernet. h>
#include <BlynkSimpleEthernet. h>

char auth[] = "YourAuthToken";

void setup()
{
  Serial. begin(9600); // See the connection status in Serial Monitor
  Blynk. begin(auth);  // Here your Arduino connects to the Blynk Cloud.
}

void loop()
{
  Blynk. run(); // All the Blynk Ma
```

Auth Token

Στο παράδειγμα

```
char auth[] = "YourAuthToken";
```

Αυτό είναι το Auth Token που στείλαμε στο mail μας στα προηγούμενα βήματα (εικόνα 2.6 – σελίδα 13)

Θα δείχνει έτσι:

```
char auth[] = "f45626c103a94983b469637978b0c78a";
```

Αφού αντικαταστήσουμε το δικό μας char auth στο sketch μας θα το «ανεβάσουμε» στο Arduino Uno. Επιλέγουμε το board που χρησιμοποιούμε στο tools>Board>Arduino/Genuino Uno και μετά πατάμε «Upload» ή «Ctrl+U».

Αφού «ανέβει», ανοίγουμε το σειριακό μόνιτορ πατώντας Ctrl+Shift+M Και μετά από λίγο θα δούμε

```
Blynk v. X. X. X
Your IP is 192. 168. 0. 11
Connecting. . .
Blynk connected!
```

(Η IP πολύ πιθανόν να διαφέρει)

Μόλις ολοκληρώσαμε επιτυχώς το δοκιμαστικό κομμάτι στο hardware και θα επιστέψουμε στην εφαρμογή του κινητού μας τηλεφώνου.

Πατώντας λοιπόν το κουμπί widget που είχαμε φτιάξει θα κάνει το ενσωματωμένο LED να ανοίξει ή να κλείσει.

***Εικόνα 2.14** : Λειτουργία Κουμπιού Widget [4]*

2.3 Blynk Widgets

Τα Widgets είναι μονάδες διεπαφής. Κάθε ένα από αυτά εκτελεί μια συγκεκριμένη διαδικασία είτε εισόδου, είτε εξόδου όταν επικοινωνεί με το Hardware.

Υπάρχουν 4 Κατηγορίες Widget:

- **Controllers[Χειριστές]** – Στέλνουν εντολές στο Hardware. Τα χρησιμοποιούμε για να ελέγχουμε το Board μας
- **Displays [Οθόνες]** – Χρησιμοποιούνται για απεικόνιση δεδομένων που έρχονται από το hardware στο κινητό μας.
- **Notifications[Ειδοποιήσεις]** – Είναι διάφορα Widget που στέλνουν ειδοποιήσεις ή μηνύματα
- **Interface[Διεπαφής]** – Είναι διάφορα Widget που κάνουν το Interface να δείχνει καλύτερο.
- **Others[Άλλα]** – Widgets που δεν ανήκουν στις υπόλοιπες κατηγορίες.

Κάθε Widget έχει τις δικές του ρυθμίσεις. Ενώ μερικά όπως π. χ. το Bridge Widget δεν έχουν.

2.3.1 Κοινές Ρυθμίσεις Widget

Επιλογή Pin

Μια από τις κυριότερες ρυθμίσεις. Καθορίζει πιο Pin να διαβάζει ή να ελέγχει.

***Εικόνα 2.15** : Επιλογή Pin - Ρυθμίσεις Widget [4]*

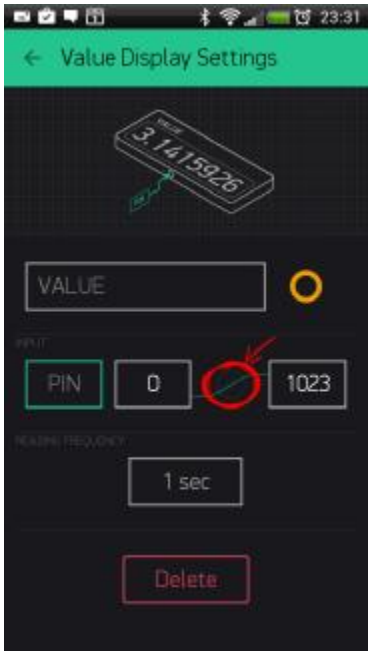
Digital Pins – Αντιπροσωπεύουν τα ψηφιακά pin στο Board που έχουμε επιλέξει.

Analog Pins – Αντιπροσωπεύουν τα αναλογικά pin.

Virtual Pins – Δεν έχουν φυσική αντιπροσώπευση. Χρησιμοποιούνται για την μεταφορά δεδομένων μεταξύ Blynk και του hardware.

Data Mapping

Σε περίπτωση που θέλουμε να καθορίσουμε κάποιες άλλες τιμές αντί για τις εισερχόμενες



Εικόνα 2.16 *: Data Mapping Blynk [4]*

Για παράδειγμα ας υποθέσουμε ότι ο αισθητήρας αποστέλνει τιμές μεταξύ 0 και 1023 αλλά εμείς θέλουμε να να δείχνει τιμές μεταξύ 0 και 100 θα ενεργοποιήσουμε το data mapping και το 1023 θα γίνει 100

Send On Release

Το Send On Release είναι διαθέσιμο για τα περισσότερα widget ελέγχου και επιτρέπει να μειωθεί ο όγκος των δεδομένων που στέλνουμε στο Hardware. Για παράδειγμα όταν μετακινούμε το widget slider, οι εντολές αποστέλνονται διαρκώς στο hardware (κάτι το οποίο μπορεί να θέλουμε για να αποκτήσουμε feedback) με αποτέλεσμα να επιβαρύνεται και τελικά να αναγκάζει το Hardware να επανεκκινεί. Το send on release όταν ενεργοποιηθεί στην ουσία αποστέλνει μόνο μια εντολή όταν καταλήξει στην τελική του θέση όταν αφήσαμε το slider, μειώνοντας έτσι τον όγκο δεδομένων που αποστέλλεται.

2.3.2 Control Widgets [Widgets Ελέγχου]

Button

Λειτουργει με την επιλογη push (κρατώντας πατημένο) ή switch (διακόπτης). Στέλνει 0/1(LOW/HIGH) τιμές. Το widget αυτό στέλνει 1 όταν το πατάμε και 0 όταν το αφήνουμε.

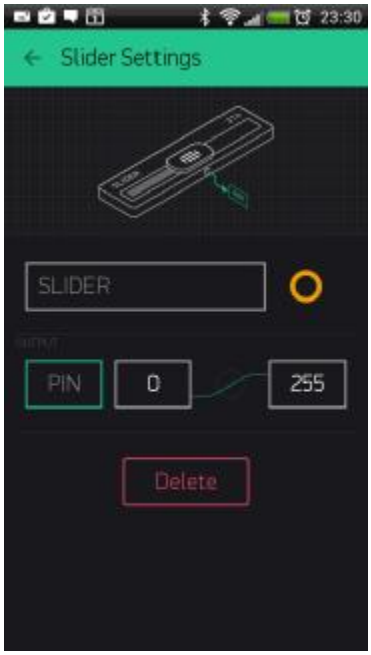
Εικόνα 2.17 *: Button Widget [4]*

Εικόνα 2.18 *: Ρυθμίσεις Button Widget [4]*

Slider

Παρόμοιο με ένα ποτενσιόμετρο. Μπορούν να δοθούν τιμές μεταξύ του Min και του Max που έχει ορισθεί.

Εικόνα 2.19 *: Slider Widget*

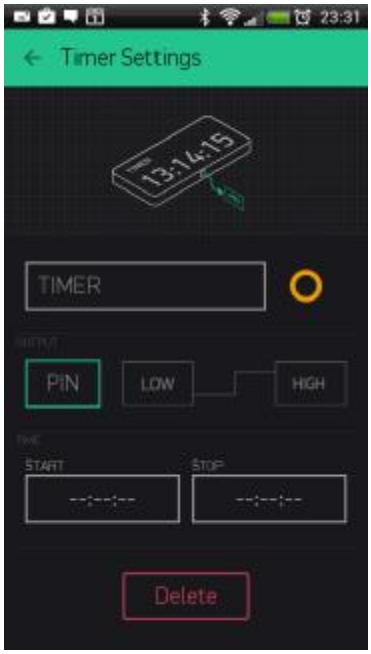


Εικόνα 2.20 *: Slider Widget Ρυθμίσεις [4]*

Timer

Το Widget Timer εκτελεί ενέργειες σε συγκεκριμένο χρόνο ακόμα και αν το Smartphone είναι απενεργοποιημένο. Το Start time στέλνει 1(HIGH) και το Stop Time στέλνει 0(LOW).

Εικόνα 2.21 *: Timer Widget [4]*



Εικόνα 2.22 *: Timer Widget Ρυθμίσεις [4]*

Joystick [Μοχλός]

Μέσω του Widget αυτού μπορούμε να χειριστούμε τις κινήσεις ενός σερβοκινητήρα σε τέσσερις διαφορετικές κατευθύνσεις.

Ρυθμισεις:

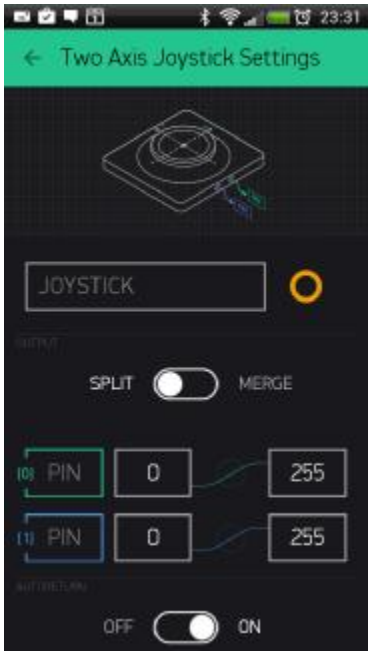
- **SPLIT/MERGE**

Επιλογή για αποστολή δεδομένων ταυτόχρονα μαζί στο φυσικό Pin ή χωριστά ως συστοιχία τιμών

- **Rotate on Tilt**

Rotate on Tilt σημαίνει όταν γυρίσουμε το κινητό μας κάθετα ο μοχλός θα γυρίσει και αυτός. Για να επιστρέψει στην αρχική του θέση πρέπει να έχουμε ενεργοποιήσει την επιλογή **Auto Return.**

Εικόνα 2.23 *: Joystick Widget [4]*

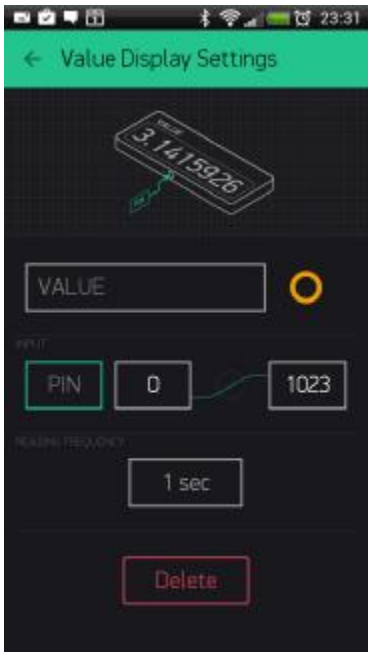


Εικόνα 2.24 *: Joystick Widget Ρυθμίσεις [4]*

2.3.3 Displays [Οθόνες]

Value Display [Απεικόνιση Τιμών]Δείχνει δεδομένα εισερχόμενα από αισθητήρες η Εικονικά Pin

Εικόνα 2.25 *: Value Display [4]*



Εικόνα 2.26 *: Value Display Widget Ρυθμίσεις [4]*

Labeled Value

Είναι μια πιο ισχυρή εκδοχή του προηγούμενου Widget καθώς περιλαμβάνει την ικανότητα μορφοποίησης των δεδομένων που λαμβάνει.

Εικόνα 2.27 *: Labeled Value Widget [4]*

Εικόνα 2.28 *: Labeled Value Widget Ρυθμίσεις [4]*

Επιλογές Μορφοποίησης

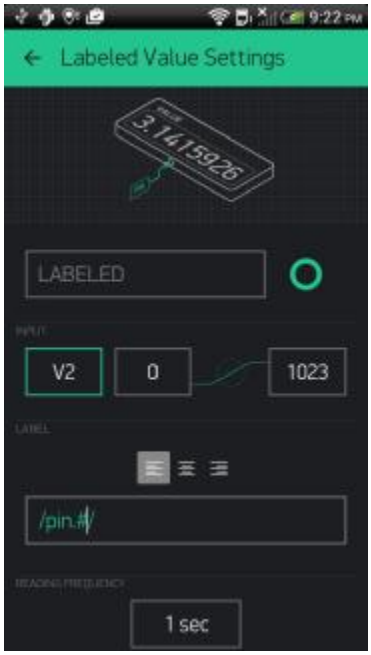
Ας υποθέσουμε ότι ο αισθητήρας αποστέλνει ένα αριθμό π. χ. 12. 6789 στην εφαρμογή Blynk. Οι επιλογές μορφοποίησης είναι

/pin/ - Δείχνει την τιμή χωρίς μορφοποίηση (12. 6789)

/pin. / - Δείχνει την τιμή χωρίς δεκαδικούς (13)

/pin. #/ - Δείχνει την τιμή με ένα δεκαδικό (12. 7)

/pin. ##/ - Δείχνει την τιμή με δυο δεκαδικούς (12. 68)



Εικόνα 2.29 *: Labeled Value Widget Ρυθμίσεις μορφοποίησης [4]*

LED

Το LED widget είναι ένα απλό LED που λειτουργεί απλά σαν ένδειξη. Για να το απενεργοποιήσουμε πρέπει να στείλουμε 0 και για να το ενεργοποιησουμε 255, ή μπορούμε να χρησιμοποιήσουμε το Blynk API ως εξής:

```
WidgetLED led1(V1); //register to virtual pin 1
```

```
led1. off();
```

```
led1. on();
```

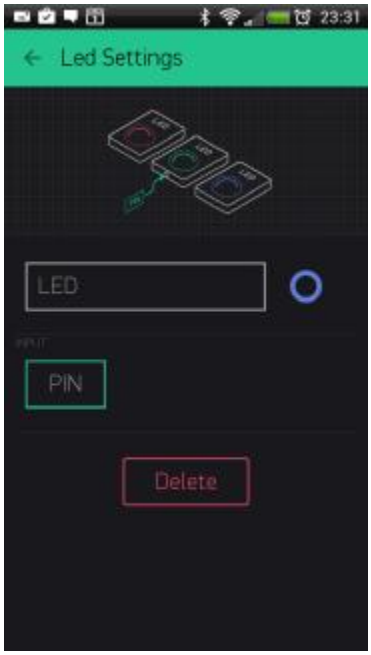
ή

[Όλες οι επιλογές μεταξύ 0 και 255 αλλάζουν την φωτεινότητα του led]

```
WidgetLED led2(2);
```

```
led2. setValue(127); //set brightness of LED to 50%.
```

Εικόνα 2.30 *: LED Widget [4]*

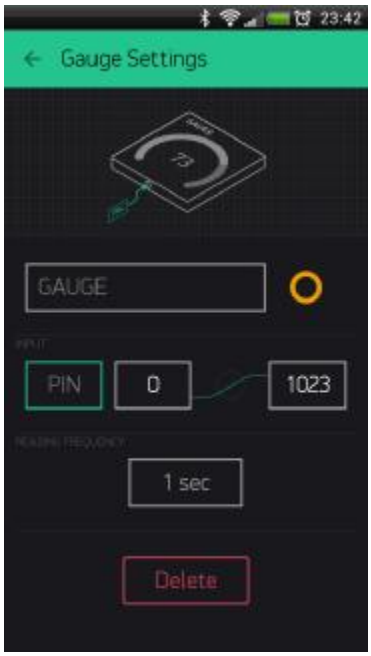


Εικόνα 2.31 : *Led Widget Ρυθμίσεις [4]*

Gauge [Μετρητής]

Το Widget Gauge ή μετρητής είναι ένας τρόπος απεικόνισης αριθμητικών δεδομένων

Εικόνα 2.32 *: Gauge Widget [4]*



Εικόνα 2.33 *: Gauge Widget Ρυθμίσεις [4]*

LCD

Η οθόνη LCD είναι μια συνηθισμένη οθόνη μεγέθους 16px x 2 px

Εντολές

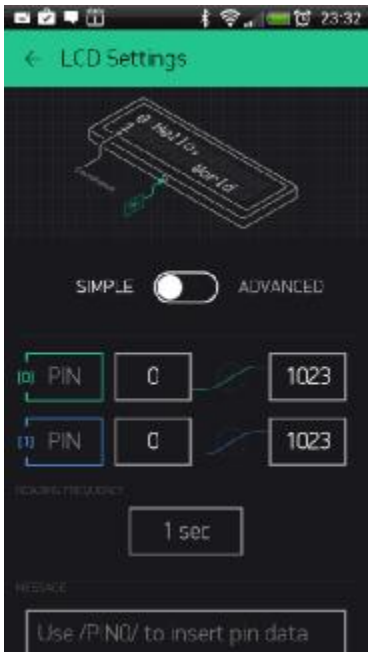
Για να χρησιμοποιήσουμε το Widget αυτό θέλουμε συγκεκριμένες εντολές:

```
lcd. print(x, y, "Your Message");
```

Όπου x είναι μια συμβολική θέση μεταξύ 0-15 και y ο αριθμός γραμμής (0 η 1)

```
lcd. clear();
```

Εικόνα 2.34 *: LCD Widget [4]*



Εικόνα 2.35 *: LCD Widget Ρυθμίσεις [4]*

Ρυθμίσεις μορφοποίησης

Ίδιες με την πιο ισχυρή εκδοχή του widget **Labeled Value**

/pin/ - Δείχνει την τιμή χωρίς μορφοποίηση (12. 6789)

/pin. / - Δείχνει την τιμή χωρίς δεκαδικούς (13)

/pin. #/ - Δείχνει την τιμή με ένα δεκαδικό (12. 7)

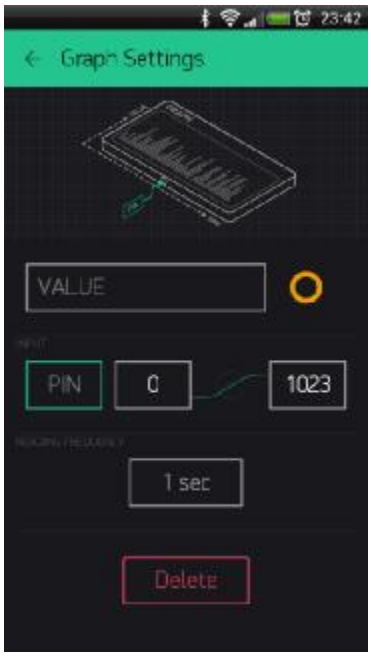
/pin. ##/ - Δείχνει την τιμή με δυο δεκαδικούς (12. 68)

Εικόνα 2.36 *: LCD Widget Ρυθμίσεις Μορφοποίησης [4]*

Graph [Γράφημα]

Το widget αυτό φτιάχνει διαγράμματα βάσει των εισερχομένων δεδομένων.

Εικόνα 2.37 *: Graph Widget [4]*



Εικόνα 2.38 *: Graph Widget Ρυθμίσεις [4]*

History Graph [Ιστορικό]

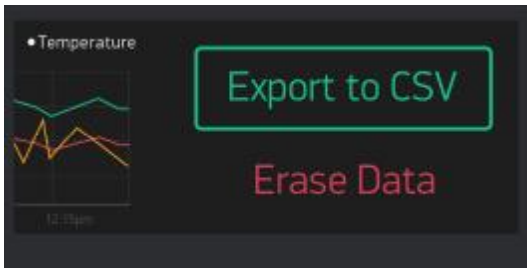
Δημιουργεί ένα γράφημα με δεδομένα που έχει στείλει στο παρελθόν το hardware στο server του Blynk. Το γράφημα έχει 3 διακριτότητες:

- Για χρονικά περιθώρια 1 ώρας η 6 ωρών η ελάχιστη διακριτότητα του γραφήματος είναι στο λεπτό
- Για χρονικά περιθώρια μίας ημέρας ή μίας εβδομάδας η ελάχιστη διακριτότητα είναι στην ώρα
- Για χρονικά περιθώρια ενός μήνα ή 3 μηνών η ελάχιστη διακριτότητα είναι στην μέρα

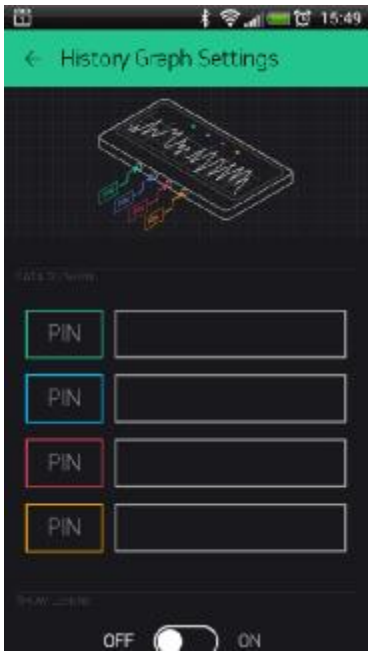
Για να δούμε δεδομένα στο γράφημα πρέπει είτε να χρησιμοποιούμε widgets με την επιλογή “Frequency reading” ή από την πλευρά του hardware να χρησιμοποιούμε την εντολή `Blynk. virtualWrite.` Έτσι στην περίπτωση αυτή τα δεδομένα από το hardware αποθηκεύονται στο server και δεν χρειάζεται να έχουμε ανοιχτή την εφαρμογή.

Για παράδειγμα έστω ότι έχουμε ένα εξάρτημα το οποίο έχει ρυθμιστεί από τον κώδικα να στέλνει δεδομένα στο server του Blynk (όπως ένας αισθητήρας θερμοκρασίας) ανά 30 δευτερόλεπτα. Οι επιλογές εμφάνισης στο γράφημα είναι η ανάδειξη των πληροφοριών ανά ένα λεπτό για το χρονικό περιθώριο 1 ώρας ή αν συμπυκνωθεί το γράφημα θα μας δείξει τα δεδομένα ανά ένα λεπτό σε ένα χρονικό περιθώριο 6 ωρών. Με τον ίδιο τρόπο λειτουργούν και οι άλλοι τρόποι εμφάνισης.

Επίσης μπορούμε να διαγράψουμε όλα τα δεδομένα που έχει συλλέξει το γράφημα πατώντας το κουμπί “Erase Data”



Εικόνα 2.39 *: Επιλογή διαγραφής δεδομένων Graph History Widget [4]*



Εικόνα 2.40 *: Ρυθμίσεις Graph History Widget [4]*

Terminal [Τερματικό]

Η οθόνη τερματικού στο Blynk δείχνει και αυτή δεδομένα που στέλνει το hardware. Η διαφορά είναι ότι μπορούμε και εμείς να στείλουμε προς το Hardware αλφαριθμητικά δεδομένα (string) και να προκαλέσουμε κάποια αντίδραση, π. χ. γράφοντας ένα όνομα θα μπορούσε το Hardware μέσω του Terminal να μας απαντήσει με ένα άλλο όνομα ή να εκτελέσει κάποια ενέργεια.

Το Terminal συγκρατεί τα τελευταία 25 μηνύματα που έχει λάβει ο server του Blynk. Το όριο αυτό μπορεί να αυξηθεί αν στήσουμε δικό μας τοπικό Blynk Server.

Οι εντολές που πρέπει να χρησιμοποιήσουμε είναι:

```
terminal. print();   // Εκτυπώνει τιμές οπως το Serial. print() στο Arduino
terminal. println(); // Εκτυπώνει τιμές αλλά κάθε φορά σε μία γραμμη οπως
και το Serial. println()
terminal. write();   // Εκτυπώνει μη επεξεργασμένα δεδομένα
terminal. flush();   // Βεβαιωνει πως τα δεδομενα εχουν σταλει
```

Εικόνα 2.41 *: Terminal Widget [4]*

Εικόνα 2.42 *: Terminal Widget Ρυθμίσεις [4]*

2.3.4 Notifications [Ειδοποιησεις]
Twitter

Το Twitter Widget συνδέει το λογαριασμό μας στο Twitter με το Blynk και δίνει την άδεια στο Hardware να στέλνει tweets.

Εικόνα 2.43 *: Twitter Widget [4]*

Εικόνα 2.44 *: Twitter Widget Ρυθμίσεις [4]*

Παράδειγμα:

```
Blynk. tweet("Hello! My Arduino can tweet now!");
```

Περιορισμοί :

- Δεν μπορούμε να στείλουμε 2 tweets με το ίδιο μήνυμα (Πολιτική Twitter).
- Μόνο ένα tweet ανά 15 δευτερόλεπτα επιτρέπεται.

Email

Το Email Widget δίνει την δυνατότητα στο Hardware να στείλει Email σε οποιαδήποτε ταχυδρομική ηλεκτρονική διεύθυνση (Email Address).

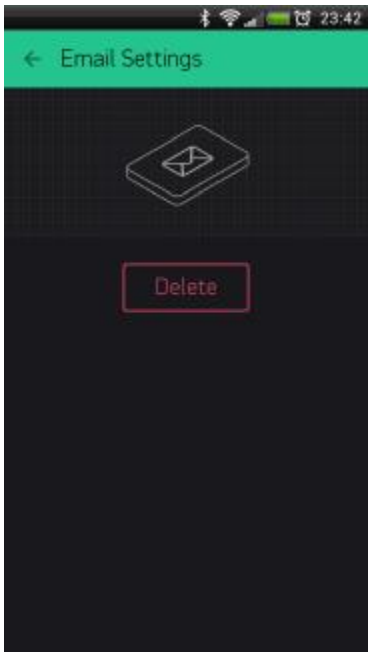
Παράδειγμα:

```
Blynk. email("my_email@example. com", "Subject", "Your message goes here");
```

Επίσης το Widget περιλαμβάνει και ένα πεδίο “προς – το” άρα δεν χρειάζεται να καθορίσουμε στο sketch του Hardware την διεύθυνση email αλλά μόνο στο widget με τον ακόλουθο τόπο.

```
Blynk. email("Subject", "Your message goes here");
```

Εικόνα 2.45 *: Email Widget [4]*



Εικόνα 2.46 *: Widget Email Ρυθμίσεις [4]*

Περιορισμοί:

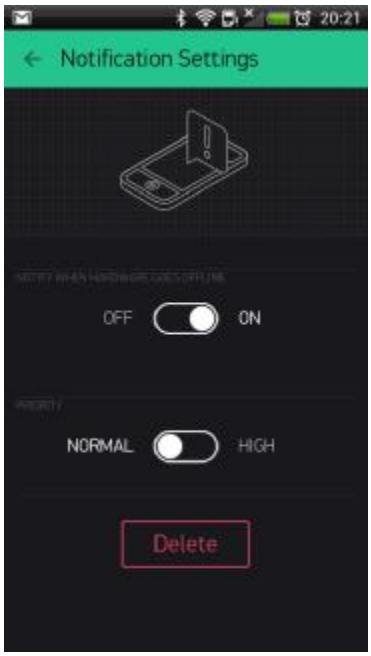
- Το μέγιστο μήκος που μπορεί να έχει ένα Email είναι 120 χαρακτήρες.
- Μόνο ένα Email επιτρέπεται ανά 15 δευτερόλεπτα.
- Στην περίπτωση που χρησιμοποιούμε Gmail περιοριζόμαστε στα 500 Emails την ημέρα (Πολιτική Google).

Push Notifications [Push Ειδοποιήσεις]

Το Widget Push Notifications μας επιτρέπει να στέλνουμε ειδοποιήσεις Push από το hardware στη συσκευή μας. Το widget στην παρούσα φάση περιλαμβάνει τις ρυθμίσεις:

- “Notify when hardware offline” – Δηλαδή, ενημέρωση όταν το hardware βγει εκτός λειτουργίας.
- “Priority” [Normal ή High] – Αν θα είναι η ειδοποίηση υψηλής προτεραιότητας ή όχι, αυξάνει δηλαδή τις πιθανότητες να παραδοθεί το μήνυμα χωρίς καθυστερήσεις.

Εικόνα 2.47 *: Push Notifications Widget [4]*



Εικόνα 2.48 *: Push Notifications Widget Ρυθμίσεις [4]*

Παράδειγμα:

```
Blynk. notify("Hardware can use push notifications now!");
```

Περιορισμοί :

- Μέγιστο μήκος ειδοποίησης είναι οι 120 χαρακτήρες.
- Μόνο μια ειδοποίηση επιτρέπεται ανά 15 δευτερόλεπτα.

2.3.5 Interface [Διεπαφή]
Tabs [Καρτελες]

Ο μόνος σκοπός του συγκεκριμένου widget είναι να αυξήσει το χώρο εργασίας μας. Μπορούμε να έχουμε μέχρι και 4 καρτέλες. Επίσης μπορούμε να μεταφέρουμε widgets μεταξύ καρτελών, απλά σέρνουμε το widget στην ετικέτα της καρτέλας που θέλουμε να την προσθέσουμε.

Εικόνα 2.49 *: Tabs Widget Ρυθμίσεις [4]*

Menu

Το Widget Menu επιτρέπει να στέλνουμε εντολές στο hardware βασισμένες στην επιλογή που κάνουμε μέσα από ένα UI (User interface – Περιβάλλον εργασίας).

Εικόνα 2.50 *: Menu Widget Ρυθμίσεις [4]*

Παράδειγμα:

```
switch (param. asInt())
  {
    case 1: { // Item 1
      Serial. println("Item 1 selected");
      break;
    }
    case 2: { // Item 2
      Serial. println("Item 2 selected");
      break;
    }
  }
```

Time Input [Εισαγωγή ώρας]

Το Widget Time Input επιτρέπει να επιλέξουμε τον χρόνο εκκίνησης και παύσης λειτουργιών του Hardware (Start/Stop). Ο χρόνος αυτός μπορεί να αξιοποιηθεί ώστε το Hardware να εκτελεί ενέργειες βασισμένες στο χρόνο που θέτει ο χρήστης. Οι ρυθμίσεις περιλαμβάνουν την ημέρα της εβδομάδας, την ζώνη ώρας, ώρα που δύει η ανατέλλει ο ήλιος σε συγκεκριμένες μορφοποιήσεις και να αποστείλουμε το χρόνο αυτό στο Hardware που χρησιμοποιούμε.

***Εικόνα 2.51** : Time Input Widget Ρυθμίσεις [4]*

Table [Κελιά]

Το Widget Table είναι χρήσιμο όταν παρόμοια δεδομένα πρέπει να έχουν μια συγκεκριμένη δομή μέσα στο ίδιο γραφικό περιβάλλον. Λειτουργούν αρόμοια με κελιά στο excel.

Προσθέτουμε μια γραμμή με την ακόλουθη εντολή:

```
Blynk. virtualWrite(V1, "add", id, "Name", "Value");
```

Παρατήρηση : Ο μέγιστος αριθμός γραμμών είναι 100.

Αν θέλουμε να δώσουμε έμφαση σε ένα αντικείμενο σε συγκεκριμένο κελί. Το κάνουμε με το ακόλουθο παράδειγμα π. χ. το κελί No 0 που περιεχει την λεξη “pick”

```
Blynk. virtualWrite(V1, "pick", 0);
```

Για να καθαρίσουμε το περιεχόμενο των κελιών χρησιμοποιούμε την ακόλουθη εντολή:

```
Blynk. virtualWrite(V1, "clr");
```

Εικόνα 2.52 *: Table Widget Ρυθμίσεις [4]*

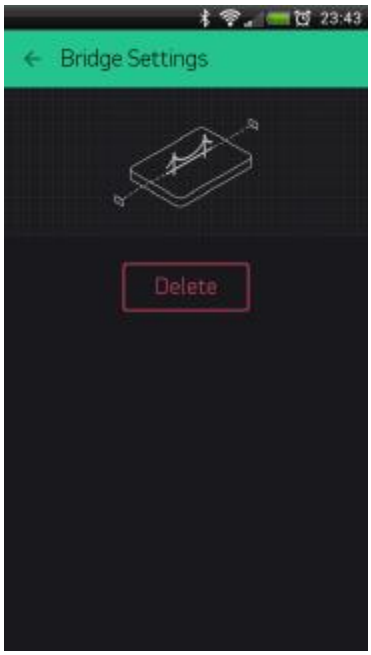
Μπορούμε να εκτελέσουμε και άλλες εντολές. Για παράδειγμα να χρησιμοποιήσουμε την γραμμή σαν κουμπί εναλλαγής γραμμών με τον ακόλουθο κώδικα :

```
BLYNK_WRITE(V1) {
   String cmd = param[0]. asStr();
   if (cmd == "select") {
       //row in table was selected.
       int rowId = param[1]. asInt();
   }
   if (cmd == "deselect") {
       //row in table was deselected.
       int rowId = param[1]. asInt();
   }
   if (cmd == "order") {
       //rows in table where reodered
       int oldRowIndex = param[1]. asInt();
       int newRowIndex = param[2]. asInt();
   }
}
```

2.3.6 Other [Άλλα]
Bridge [Γέφυρα]

Η γέφυρα μπορεί να χρησιμοποιηθεί για την επικοινωνία μεταξύ ενός Hardware π. χ. Arduino Uno με άλλο Hardware π. χ. Arduino Mini, χωρίς να εμπλέκεται η εφαρμογή Blynk ενδιάμεσα. Μπορούμε να στείλουμε ψηφιακά ή αναλογικά δεδομένα, ή και εικονικές εντολές από μια συσκευή σε μια άλλη ξέροντας μόνο το Auth token [κλειδί ασφαλείας]. Ένα βασικό παράδειγμα είναι ότι θα μπορούσαμε να πατάμε ένα φυσικό κουμπί στο Arduino Uno και να ανάβει ένα LED στο Arduino Mini. Μπορούμε να χρησιμοποιήσουμε πολλαπλές γέφυρες για να ελέγξουμε πολλαπλές συσκευές.

Εικόνα 2.53 *: Bridge Widget [4]*



Εικόνα 2.54 *: Bridge Widget Ρυθμίσεις [4]*

Ο τρόπος λειτουργίας του Bridge Widget είναι ότι παίρνει ένα εικονικό Pin και το μετατρέπει σε κανάλι για να αλληλεπιδράσει με το άλλο Hardware. Αυτό σημαίνει ότι μπορούμε να στείλουμε εικονικά ψηφιακά ή αναλογικά Pins στο Hardware που θέλουμε. Αλλά δεν χρησιμοποιούμε ποτέ Pins όπως A0, A1, A2 διότι μερικές φορές ο πυρήνας του Arduino μπορεί να μπερδευτεί και να τα συνδέσει με άλλα Pins.

Ακολουθεί παράδειγμα επικοινωνίας μεταξύ Hardware A το οποίο στέλνει δεδομένα στο Hardware B:

```
WidgetBridge bridge1(V1); //Initiating Bridge Widget on V1 of Device A

void setup() {

    Blynk. begin(. . . );

    while (Blynk. connect() == false) {

        // Wait until Blynk is connected

    }

    bridge1. digitalWrite(9, HIGH); // will trigger D9 HIGH on Device B. No code on Device B required

    bridge1. analogWrite(10, 123);

    bridge1. virtualWrite(V1, "hello"); // you need to write code on Device B in order to receive this value. See below

    bridge1. virtualWrite(V2, "value1", "value2", "value3");

}

BLYNK_CONNECTED() {

  bridge1. setAuthToken("OtherAuthToken"); // Token of the hardware B

}
```

Σημαντική Παρατήρηση: Όταν αποστέλνουμε δεδομένα από το A στο B με την εντολή **virtualWrite()**, το B θα πρέπει να επεξεργαστεί τα δεδομένα που λαμβάνει, για να γίνει αυτό χρησιμοποιούμε την ακόλουθη εντολή.

```
BLYNK_WRITE(V5){

    int pinData = param. asInt(); //pinData variable will store value that came via Bridge

}
```

Eventor [Γεγονός]

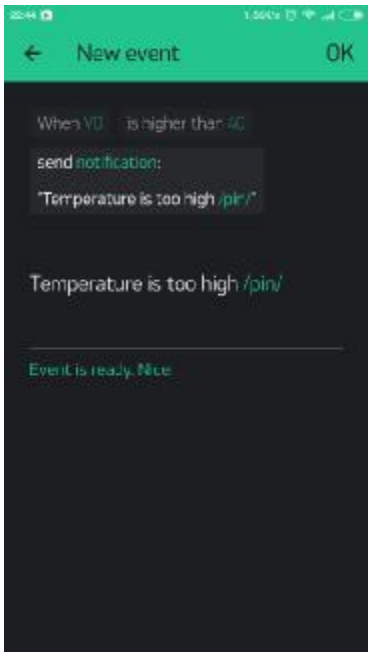
Το Widget Eventor επιτρέπει τη δημιουργία απλών συσχετίσεων μεταξύ γεγονότων όπως όταν χρησιμοποιούμε την εντολή If στον προγραμματισμό. Ένα παράδειγμα είναι η ανάγνωση της θερμοκρασίας από τον ανάλογο αισθητήρα και αποστολή ειδοποίησης push όταν η θερμοκρασία φτάσει ένα συγκεκριμένο όριο όπως :

```
float t = dht. readTemperature();

  if (isnan(t)) {

    return;

  }

  if (t > 40) {

    Blynk. notify(String("Temperature is too high : ") + t);

  }
```

Με τη χρήση του Eventor Widget λοιπόν δεν χρειάζεται καν να γραφεί αυτός ο κώδικας. Το μόνο που χρειάζεται είναι η αποστολή του δεδομένου από τον αισθητήρα προς τον Server. :

```
float t = dht. readTemperature();

Blynk. virtualWrite(V0, t);
```

Στο Widget κάνουμε τις απαραίτητες ρυθμίσεις στο “New Event”:



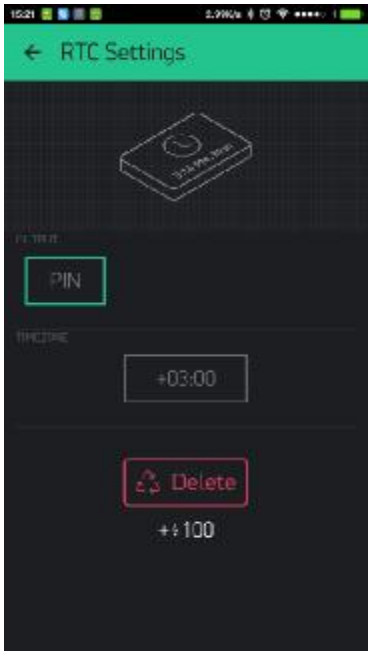
Εικόνα 2.55 *: Eventor Widget "New Event" [4]*

Δεν ξεχνάμε να προσθέσουμε το Widget Push Notifications για το συγκεκριμένο παράδειγμα

Εικόνα 2.56 *: Eventor Widget Ρυθμίσεις [4]*

RTC [Ρολόι Πραγματικού Χρόνου]

Το ρολόι πραγματικού χρόνου επιτρέπει να λάβει ο χρήστης την ώρα από το Server του Blynk. Δίνεται η δυνατότητα επιλογής ζώνης ώρας από τις ρυθμίσεις του Widget για να λάβει το Hardware την σωστή ώρα στην απαραίτητη τοποθεσία.



Εικόνα 2.57 *: RTC Widget Ρυθμίσεις [4]*

Bluetooth

Το Widget αυτό δημιουργήθηκε για να υποστηρίξει Bluetooth εξαρτήματα.

Εικόνα 2.58 *: Bluetooth Widget Ρυθμίσεις [4]*

2. 4 Virtual Pins [Εικονικά Pin]

Το Blynk παρόλο που μπορεί να χειριστεί τα ψηφιακά και αναλογικά Pin του Board που χρησιμοποιούμε, πολλές φορές τα pin αυτά δεν είναι αρκετά. Έτσι μέσω του Blynk μπορούμε να χρησιμοποιήσουμε εικονικά Pin χωρίς να έχουν φυσική ύπαρξη προκειμένου να χειριστούμε διάφορες λειτουργίες και γενικά να αποστείλουμε δεδομένα στο hardware αλλά και να λάβουμε. Για παράδειγμα μπορούμε να ενεργοποιήσουμε λειτουργίες, να μετατρέψουμε τιμές σε άλλη μορφή να ελέγξουμε σερβοκινητήρες και άλλα. Ένα πιο συγκεκριμένο παράδειγμα είναι αν ο σερβοκινητήρας περιστραφεί κατά 20 μοίρες να σταλεί η εντολή στο εικονικό Pin έτσι ώστε να ενεργοποιηθεί το LED Widget το οποίο είναι συνδεδεμένο με το Pin αυτό.

Για να στείλει το hardware δεδομένα σε ένα εικονικό Pin στο LCD Widget γίνεται με αυτό τον τρόπο

```
Blynk. virtualWrite(pin, "abc");
Blynk. virtualWrite(pin, 123);
Blynk. virtualWrite(pin, 12. 34);
Blynk. virtualWrite(pin, "hello", 123, 12. 34);
```

Ενώ για να αποστείλουμε δεδομένα από το App στο Hardware, όπως με το πάτημα του Button Widget να εκτελείται μια λειτουργία έχουμε:

```
BLYNK_WRITE(V1) //Button Widget is writing to pin V1
{
  int pinData = param. asInt();
}
```

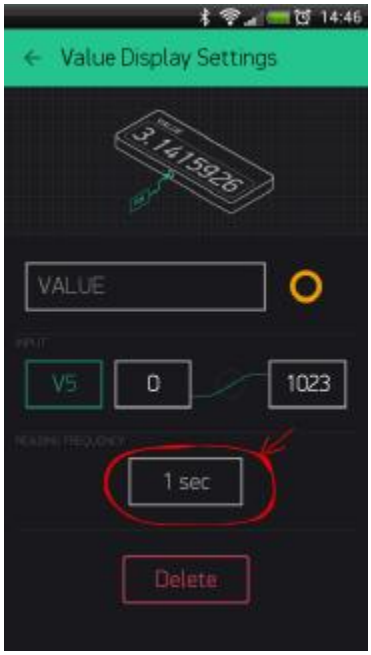
Όταν πατάμε το κουμπί η εφαρμογή στέλνει 1 (ON) στο δεύτερο πάτημα στέλνει 0 (OFF)

Η ρύθμιση του Widget γίνεται έτσι:

Εικόνα 2.59 *: Virtual Pin ρυθμίσεις [4]*

Δημιουργία αιτήσεων ανά χρονικό διάστημα

Χρησιμοποιώντας την ενσωματωμένη επιλογή του Blynk που ονομάζεται ‘Reading Frequency’ μπορούμε να θέσουμε κάθε πότε ένα Widget (έστω το Value Widget) θα διαβάζει την τιμή που του στέλνει το hardware. Έτσι στο ακόλουθο κομμάτι κώδικα αποστέλνουμε την τιμή του χρόνου που το Arduino έχει μείνει ενεργοποιημένο και στη συνέχεια το Value Widget διαβάζει την τιμή ανά 5 δευτερόλεπτα.



***Εικόνα 2.60**: Αιτήσεις ανά χρ. Διάστημα [4]*

```
BLYNK_READ(V5) // Widget in the app READs Virtual Pin V5 with the certain
frequency
{
  // This command writes Arduino's uptime in seconds to Virtual Pin V5
  Blynk. virtualWrite(5, millis() / 1000);
}
```

3 Κατασκευή

3.1 Εξαρτήματα ηλεκτρονικού κυκλώματος

3.1.1 ESP8266 [ESP12-E]

Το ESP8266 είναι ένα χαμηλού κόστους Chip με δυνατότητες Wi-Fi το οποίο κατασκευάστηκε από την κινέζικη εταιρεία **Espressif Systems**, η οποία εδρεύει στην Σαγκάη. Περιλαμβάνει Μικροελεγκτή (MCU) και δυνατότητα TCP/IP συνδέσεων.

Αρχικά ήταν αδύνατη η αυτόνομη αξιοποίηση του ESP8266, γιατί δεν μπορούσε να προγραμματιστεί το chip του και για το λόγο αυτό χρησιμοποιούνταν αρχικά ως εξάρτημα το οποίο έδινε σε Arduino boards την δυνατότητα ασύρματης διασύνδεσης (Wi-Fi). Τέλη Οκτώμβρη 2014 η εταιρεία ανέπτυξε ένα software development kit (SDK), κιτ ανάπτυξης λογισμικού το οποίο έδινε την ευκαιρία να προγραμματιστεί το chip του ESP αφαιρώντας έτσι την εξάρτηση από ξεχωριστό μικροελεγκτή για την αξιοποίηση του.

Άλλα SDK ανοικτού κώδικα είναι τα ακόλουθα:

- **NodeMCU**: Firmware Βασισμένο στην γλώσσα προγραμματισμού LUA.
- **Arduino**: Firmware βασισμένο σε C++. Ο πυρήνας αυτός επιτρέπει στον επεξεργαστή του ESP8266 και τα εξαρτήματα του Wi-Fi να προγραμματιστούν όπως οποιαδήποτε άλλη συσκευή Arduino.
- **MicroPython**: Μια έκδοση της MicroPython (Η έκδοση της γλώσσας προγραμματισμού Python για ενσωματωμένες συσκευές – Embed devices) για την πλατφόρμα του ESP8266.

Το δημοφιλέστερο μοντέλο του ESP8266 ήταν το ESP-01 από τον κατασκευαστή AI-Thinker. Είχε αρκετά μικρές διατάσεις 14.3mm × 24.8 mm αλλά λίγα Pins. το οποίο ήταν απαγορευτικό για μεγάλου σκέλους Projects. Λόγω όμως του μικρού του κόστους διαδόθηκε στην κοινότητα πολύ γρήγορα.

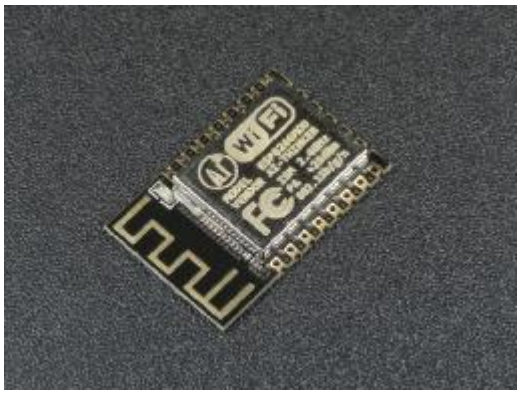
Εικόνα 3.1 *ESP8266 ESP-01 [5]*

Οι διάφορες εκδόσεις που ακολουθήσαν ήταν οι :

Όνομα	Ενεργά Pins	Κεραία	Ηλεκτρομανγητική Θωράκιση	Διαστάσεις (mm)	Παρατηρήσεις
ESP-01	6	Διαδρομή στην πλακέτα	Όχι	14.3 × 24.8	
ESP-02	6	Υποδοχή UFL	Όχι	14.2 × 14.2	
ESP-03	10	Κεραμική	Όχι	17.3 × 12.1	
ESP-04	10	Καμία	Όχι	14.7 × 12.1	
ESP-05	3	Υποδοχή UFL	Όχι	14.2 × 14.2	
ESP-06	11	Καμία	Ναι	14.2 × 14.7	Μη εγκεκριμένο απο FCC
ESP-07	14	Κεραμική + Υποδοχή UFL	Ναι	20.0 × 16.0	Μη εγκεκριμένο απο FCC
ESP-08	10	Καμία	Ναι	17.0 × 16.0	Μη εγκεκριμένο απο FCC
ESP-09	10	Καμία	Όχι	10.0 × 10.0	

ESP-10	3	Καμία	Όχι	14.2 × 10.0	
ESP-11	6	Κεραμική	Όχι	17.3 × 12.1	
ESP-12	14	Διαδρομή στην πλακέτα	Ναι	24.0 × 16.0	Εγκεκριμένο απο FCC και CE
ESP-12E	20	Διαδρομή στην πλακέτα	Ναι	24.0 × 16.0	4 MB μνήμη Flash
ESP-12F	20	Διαδρομή στην πλακέτα	Ναι	24.0 × 16.0	Εγκεκριμένο απο FCC και CE. Αυξημένη απόδοση κεραίας. 4 MB Μνήμη Flash
ESP-13	16	Διαδρομή στην πλακέτα	Ναι	18.0 x 20.0	
ESP-14	22	Διαδρομή στην πλακέτα	Ναι	24.3 x 16.2	

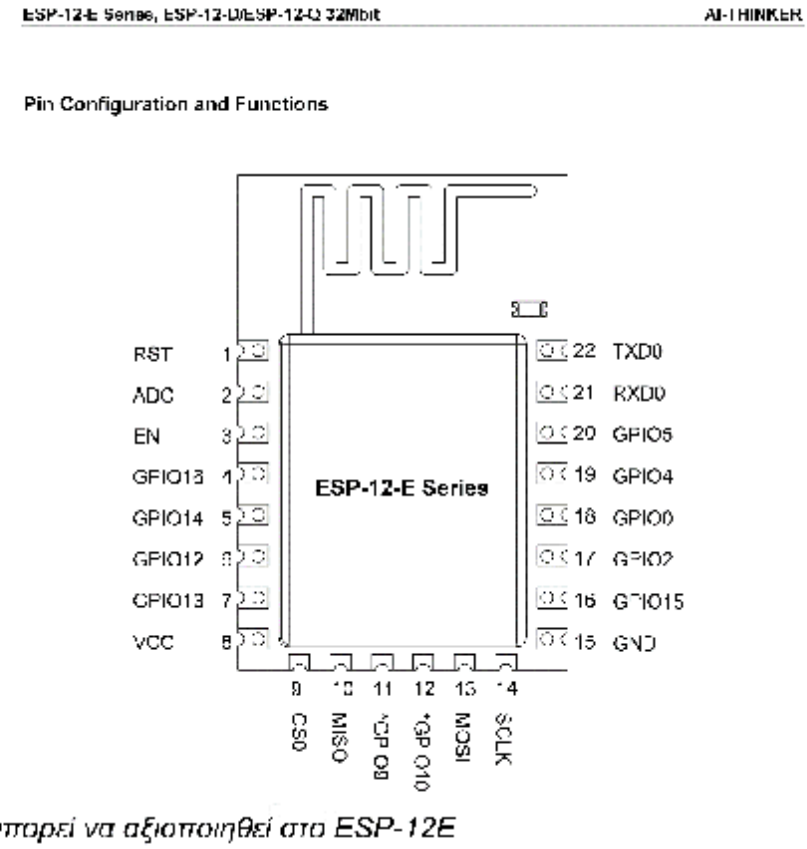
Η έκδοση που χρησιμοποιήθηκε στην κατασκευή ήταν η έκδοση **ESP-12E** με τα ακόλουθα τεχνικά χαρακτηριστικά



Εικόνα 3.2 *: ESP8266 ESP-12E [6]*

Τεχνικά Χαρακτηριστικά ESP8266 ESP-12E		
Κατηγορίες	**Αντικείμενα**	**Τιμές**
Παράμετροι Wi-Fi	Πρωτόκολλο Wi-Fi	802.11 b/g/n
	Εύρος Συχνότητας	2.4G-2.5G(2400M-2483.5M)
Παράμετροι Hardware	Τάση λειτουργίας	3.0~3.6V
	Ρεύμα λειτουργίας	Μέση τιμή : 80mA
	Μέγιστες Θερμοκρ. Λειτουργίας	-40°~125°
	Μνήμη Flash	4MB
	Μέγιστη Θερμοκρ. Συγκόλλησης	260°
	Μικροελεγκτής	Tensilica L106 32-bit
Παράμετροι Λογισμικού	Wi-Fi Mode	Station/SoftAP/SoftAP+Station

	Ασφάλεια	WPA/WPA2
	Κρυπτογράφηση	WEP/TKIP/AES
	Αναβάθμιση Firmware	UART Download / OTA (via network)
	Πρωτόκολλα Δικτύου	IPv4, TCP/UDP/HTTP/FTP



Εικόνα 3.3 *: ESP12-E Διάγραμμα Pin [6]*

Επεξήγηση Pin ESP-12E

RST	Επανεκκίνηση Module
ADC	Μετατροπή Αναλογικού σήματος σε ψηφιακού. Τάση εισαγωγής 0-1V/0-1024mV
EN	Pin ενεργοποίησης του chip
GPIO16	GPIO16; Μπορεί να χρησιμοποιηθεί για να επαναφέρει το chip από "Deep Sleep"
GPIO14	GPIO14
GPIO12	GPIO12
GPIO13	GPIO13

VCC	3. 3V Παροχή ενέργειας
CS0	Επιλογή Chip [Chip Selection]
MISO	Slave Output Main input [Αξιοποίηση στην σειριακή επικοινωνία περιφερειακών]
GPIO9	GPIO9*
GBIO10	GBIO10*
MOSI	Main Output Slave Input[Αξιοποίηση στην σειριακή επικοινωνία περιφερειακών]
SCLK	Serial Clock
GND	GND
GPIO15	GPIO15
GPIO2	GPIO2
GPIO0	GPIO0
GPIO4	GPIO4
GPIO5	GPIO5
RXD	UART0_RXD GPIO3
TXD	UART0_TXD GPIO1

*GPIO9 & GPIO10 Δεν μπορούν να αξιοποιηθούν στην έκδοση ESP-12E

GPIO Σημαίνει General Purpose Input/Output - Pin γενικού σκοπού εισαγωγής/εξαγωγής

3.1.2 ESP8266 ESP-12E Breakout Board

Λόγω του μικρού σχήματος του ESP-12E η πρόσβαση στα Pin είναι αρκετά δυσπρόσιτη έτσι έχουν κατασκευαστεί για ακριβώς αυτό το λόγο τα Breakout Board όπως εμφανίζονται παρακάτω. Η σύνδεση του ESP12-E με το Breakout Board γίνεται με κολλητήρι ώστε να γίνει η ένωση :

Εικόνα 3.4 *: ESP8266 ESP-12E Breakout Board [32]*

Εικόνα 3.5 *: ESP8266 ESP-12E Breakout Board Soldered [32]*

3.1.3 RFID MFRC522 Reader/Writer

Το εξάρτημα MFRC522 βασίζεται πάνω στην τεχνολογία RFID Radio Frequency Identification ή «ταυτοποίηση μέσω ραδιοσυχνοτήτων» και έχει ως σκοπό να διαβάζει τα δεδομένα από RFID ετικέτες (Tags).

Πιο συγκεκριμένα τα συστήματα RFID απαρτίζονται από δύο κύρια μέρη:

Το πρώτο είναι οι πομποδέκτες (transponders) που συχνά αναφέρονται και ως ετικέτες RFID (RFID tags). Οι ετικέτες RFID είναι μικρά chips που αποτελούνται από ένα ολοκληρωμένο κύκλωμα, το οποίο περιλαμβάνει μνήμη ώστε να αποθηκεύει δεδομένα-πληροφορίες, και μία κεραία.

Το δεύτερο μέρος είναι οι αναγνώστες ή αισθητήρες (readers), οι οποίοι ανακτούν τα δεδομένα από τις ετικέτες RFID. Οι αναγνώστες RFID έχουν ενσωματωμένα μια κεραία και μια μονάδα ελέγχου.

Η λειτουργία των συστημάτων RFID είναι απλή και βασίζεται στη δυναμική και αμφίδρομη επικοινωνία των ετικετών και των αναγνωστών. Όταν οι ετικέτες RFID βρεθούν στην εμβέλεια της κεραίας του αναγνώστη, η μονάδα ελέγχου επικοινωνεί με ραδιοκύματα με την κεραία των ετικετών RFID. Οι ετικέτες RFID ενεργοποιούνται με τη σειρά τους και επιστρέφουν τα αναζητούμενα δεδομένα στους αναγνώστες. Στη συνέχεια παρεμβαίνει ένα ενδιάμεσο λογισμικό, το οποίο κατανοεί τις πληροφορίες, οι οποίες αποστέλλονται από τη μονάδα ελέγχου του αναγνώστη. Ο αναγνώστης τις μεταφέρει στο εκάστοτε πληροφοριακό σύστημα.

Οι ετικέτες RFID κατηγοριοποιούνται σε τρεις τύπους ανάλογα με τον τρόπο επικοινωνίας μεταξύ των ετικετών και των αναγνωστών, δηλαδή σε:

- ενεργές ετικέτες [Active Tags]
- παθητικές ετικέτες [Passive Tags]
- ημι-παθητικές ετικέτες [Semi – Passive Tags]

Οι παθητικές ετικέτες δεν έχουν εσωτερική πηγή ενέργειας και τροφοδοτούνται από τον αναγνώστη όταν έρθει σε αρκετά κοντινή επαφή μέσω ηλεκτρομαγνητικής ενέργειας. Τα μέρη από τα οποία αποτελείται είναι το εσωτερικό κύκλωμα με το chip και μια κεραία. Οι ετικέτες αυτές δεν χρησιμοποιούν όλες την ίδια συχνότητα:

- 125 – 134 KHz – Χαμηλής συχνότητας - Low Frequency (LF) – Μεγάλο μήκος κύματος με μικρή απόσταση ανάγνωσης (1-10cm).
- 13. 56 MHz – Υψηλής συχνότητας - High Frequency (HF) & Κοντινής επικοινωνίας Near-Field Communication (NFC) – Τυπική απόσταση ανάγνωσης 1cm με 1m.
- 865 – 960 MHz – Υπερυψηλή Συχνότητα - Ultra High Frequency (UHF) – Χαμηλό μήκος κύματος με απόσταση ανάγνωσης 5-6 m αλλά μεγαλύτερες UHF ετικέτες μπορούν να επιτύχουν αποστάσεις ανάγνωσεις εως και 30 m.

Οι ενεργές ετικέτες βασίζονται σε μπαταριά για την τροφοδοσία τους και για το λόγο αυτό έχουν και μεγάλες αποστάσεις ανάγνωσης αλλά και μεγάλη μνήμη για αποθήκευση δεδομένων. Τα ενεργά συστήματα RFID αποτελούνται από 3 βασικά μέρη :

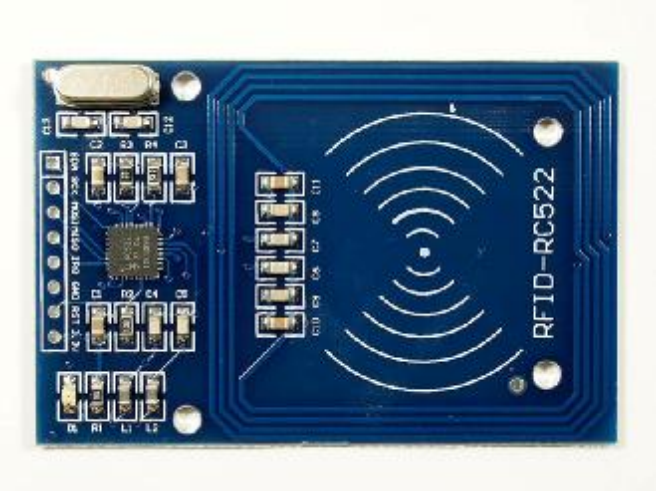
- Τον αναγνώστη
- Την κεραία
- Την ετικέτα

Οι συχνότητες που χρησιμοποιούν είναι οι ακόλουθες :

- 433 MHz - Όπου είναι και η συνηθέστερη επιλογή λόγω του υψηλοτέρου μήκους κύματος επειδή λειτουργεί καλυτέρα σε αντικείμενα εχθρικά σε ραδιοσυχνότητες όπως το νερό και το μέταλλο
- 915 MHz

Τα δεδομένα που αποθηκεύονται στις ετικέτες αποτελούνται από ένα μοναδικό αναγνωριστικό και μπορούν, επίσης, να περιλαμβάνουν ένα λειτουργικό σύστημα, μία αποθήκη δεδομένων και έναν ηλεκτρονικό κώδικα προϊόντων (Electronic Product Code - EPC). Το μέγεθος των δεδομένων, που μια ετικέτα RFID έχει την δυνατότητα να αποθηκεύσει, καθορίζεται από τον εκάστοτε προμηθευτή αλλά και την ίδια την εφαρμογή, με συνήθως το ανώτερο όριο αποθήκευσης να είναι τα 2KB. Χωρητικότητα αρκετή για να αποθηκευτούν τα απαραίτητα δεδομένα του κάθε αντικειμένου. Μια άλλη σημαντική κατηγοριοποίηση που μπορούμε να διακρίνουμε στις ετικέτες RFID σχετίζεται με την κατασκευή και την εφαρμογή τους. Δεδομένου ότι τα συστήματα RFID έχουν εφαρμογή σε διάφορους τομείς στην καθημερινή ζωή του σύγχρονου ανθρώπου, η κατασκευή των ετικετών RFID αλλάζει ανάλογα με τις εφαρμογές και τις ανάγκες που χρειάζεται κάθε φορά, να καλύψει.

Οι αναγνώστες RFID αποτελούνται από μία κεραία, η οποία αναλαμβάνει την επικοινωνία, μέσω ραδιοσυχνοτήτων, με τις ετικέτες. Καθώς και μία μονάδα ελέγχου, που εκτελεί δύο συγκεκριμένα έργα. Πρωτίστως τον καθορισμό των διάφορων ενεργειών (αποστολή/λήψη σημάτων, ανάγνωση/εγγραφή ετικετών κ.ά.), ενέργεια που πραγματοποιείται μέσω του ενδιάμεσου λογισμικού. Και δευτερευόντως την επικοινωνία με το πληροφοριακό σύστημα. Οι αναγνώστες RFID μπορούν να κατηγοριοποιηθούν σε σχέση με τις φυσικές τους διαστάσεις, την εφαρμογή τους και τις τεχνικές ιδιότητες σε "σταθερούς αναγνώστες", "ολοκληρωμένους αναγνώστες", "αναγνώστες χειρός" και σε "ενσωματωμένους αναγνώστες"



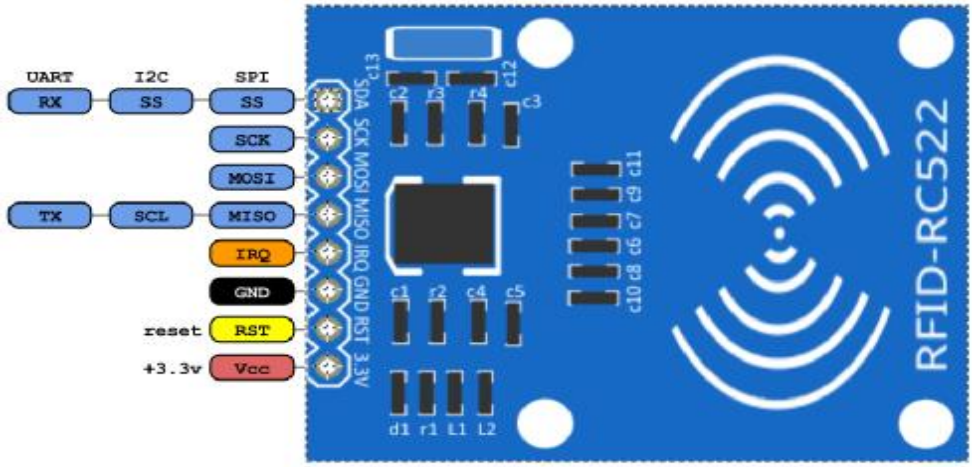
Εικόνα 3.6 *: RFID Reader MFRC522 [33]*

Ο αναγνώστης MFRC522 υποστηρίζει ISO/IEC 14443 A/MIFARE και NTAG. Η ανέπαφη επικοινωνία αγγίζει ταχύτητες των 848kB.

Οι διεπαφές του δέκτη (Host Interface) που υποστηρίζονται είναι οι ακόλουθες:

- Σειριακη Περιφερειακη Επικοινωνια - Serial Peripheral Interface (SPI)
- Σειριακη UART (universal asynchronous receiver/transmitter**)**
- I2C – Διεπαφής διάυλου (Bus interface)

Για την επικοινωνία με το ESP8266 χρησιμοποιήθηκε το πρωτόκολλο SPI ως μέσο για την επικοινωνία μεταξύ του ESP8266 και του αναγνώστη RFID-RC522, διότι η ψηφιακή βιβλιοθήκη που αναπτύχθηκε για τον συγκεκριμένο αναγνώστη χρησιμοποιεί SPI. Ο αναγνώστης υποστηρίζει και UART και I2C αλλά δεν υπάρχει υποστήριξη από πλευράς software.



Εικόνα 3.7 *: RFID Reader MRC522 Διάγραμμα Pin [33]*

Επεξήγηση Pin για μετάδοση με SPI (Serial Peripheral Interface)

SDA	SPI signal input
SCK	SPI serial clock input
MOSI	SPI master out, slave in
MISO	SPI master in, slave out
IRQ	Δεν χρησιμοποιείται
GND	Γείωση
RST	Επανεκκίνηση
3. 3V	Παροχή Τάσης 3. 3V

3.1.4 RFID Passive 13.56 MHz Keychain Tag

Ως παθητική ετικέτα RFID σε συνδυασμό με τον αναγνώστη RC522 χρησιμοποιήθηκε η ακόλουθη η οποία έχει χαμηλό κόστος και περιλαμβάνει την ευκολία του να τοποθετηθεί σε ένα μπρελόκ με τα κλειδιά μας.

Εικόνα 3.8 *: 13.56 Mhz Keychain Passive Tag [34]*

3.1.5 RFID Passive 13. 56 MHz Keycard Tag

Παθητική ετικέτα άλλου τύπου είναι η ακόλουθη στην μορφή κάρτας. Οι διαστάσεις είναι οι ίδιες με μια πιστωτική κάρτα η κάρτα ATM.

Εικόνα 3.9 *: 13.56 Mhz Passive Keycard Tag [34]*

3.1.6 6V Voltage Regulator L7806CV TO-220

Ο γραμμικός ρυθμιστής τάσης L7806CV τύπου TO-220 χρησιμοποιήθηκε για να παράσχει τροφοδοσία στον σερβοκινητήρα AS3103RG που αξιοποιήθηκε όπως θα δούμε αργότερα στο κύκλωμα.

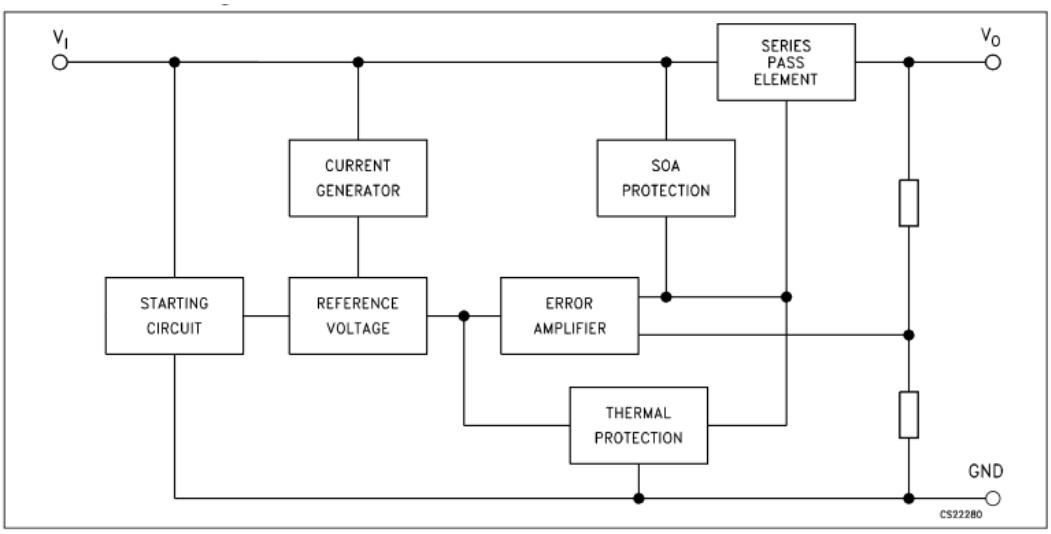
Οι γραμμικοί ρυθμιστές τάσης έχουν ως σκοπό να λαμβάνουν μια τάση εισόδου μεταξύ δύο τιμών και να την εξάγουν ως μια σταθερή τιμή. Στην περίπτωση του L7806CV η τάση εισαγωγής είναι μεταξύ 8-35V και αποδίδει ως τάση εξόδου σταθερά 6V. Για να το καταφέρει αυτό ο γραμμικός ρυθμιστής, η αντίσταση στο κύκλωμα του ρυθμιστή εναλλάσσεται αναλόγως το φορτίο. Ο μηχανισμός αυτός λειτουργεί δηλαδή σαν μια ρυθμιζόμενη αντίσταση όπου διαρκώς ρυθμίζει ένα κύκλωμα διαιρέτη τάσεως για να αποδώσει την σταθερή τάση στην έξοδο ενώ την διαφορά τάσης την μετατρέπει σε θερμότητα. Παλιότερα ως μηχανισμός αντίστασης χρησιμοποιούνταν Vacuum Tubes ενώ πλέον σε σύγχρονα σχέδια ρυθμιστών τρανζίστορ. Επειδή όμως λόγω της ιδιομορφίας του κυκλώματος, η τάση εξόδου πρέπει να είναι πάντα μικρότερη από την τάση εισόδου η απόδοση είναι αρκετά περιορισμένη. Ένας **εναλλασσόμενος ρυθμιστής (Switching Regulator)** όμως χρησιμοποιεί μια διαδικασία ανατροφοδότησης όπου ανοιγοκλείνει ένα κύκλωμα σε σειρά πολλές φορές αναλόγως τις ανάγκες του δικτύου έτσι ώστε να μην σπαταλά την ενέργεια.

TO-220

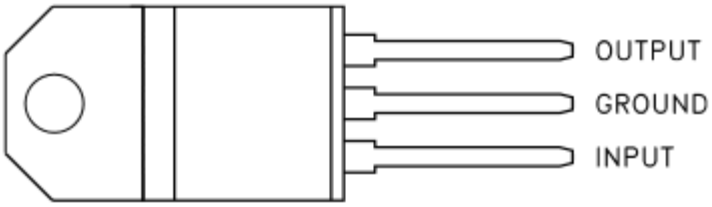
Εικόνα 3.10 *6V Voltage Regulator L7806CV TO-220 [35]*

Χαρακτηριστικά L7806CV τύπου TO-220

- Προστασία από βραχυκύκλωση
- Προστασία από υπερθέρμανση
- Σταθερό ρεύμα εξόδου έως 1.5A
- Σταθερή Τάση εξόδου 6V
- Μέγιστη Τάση εισόδου 35V
- Ελάχιστη Τάση εισόδου 8V
- Μέγιστη Θερμοκρασία λειτουργίας 125C



Εικόνα 3.11 *Διασύνδεση εσωτερικού κυκλώματος L7806CV TO-220 [35]*



Εικόνα 3.12 *Διάγραμμα Pin L7806CV TO-220 [35]*

INPUT = Pin Τάσης εισόδου (8-35V)

GROUND = Pin Γείωσης

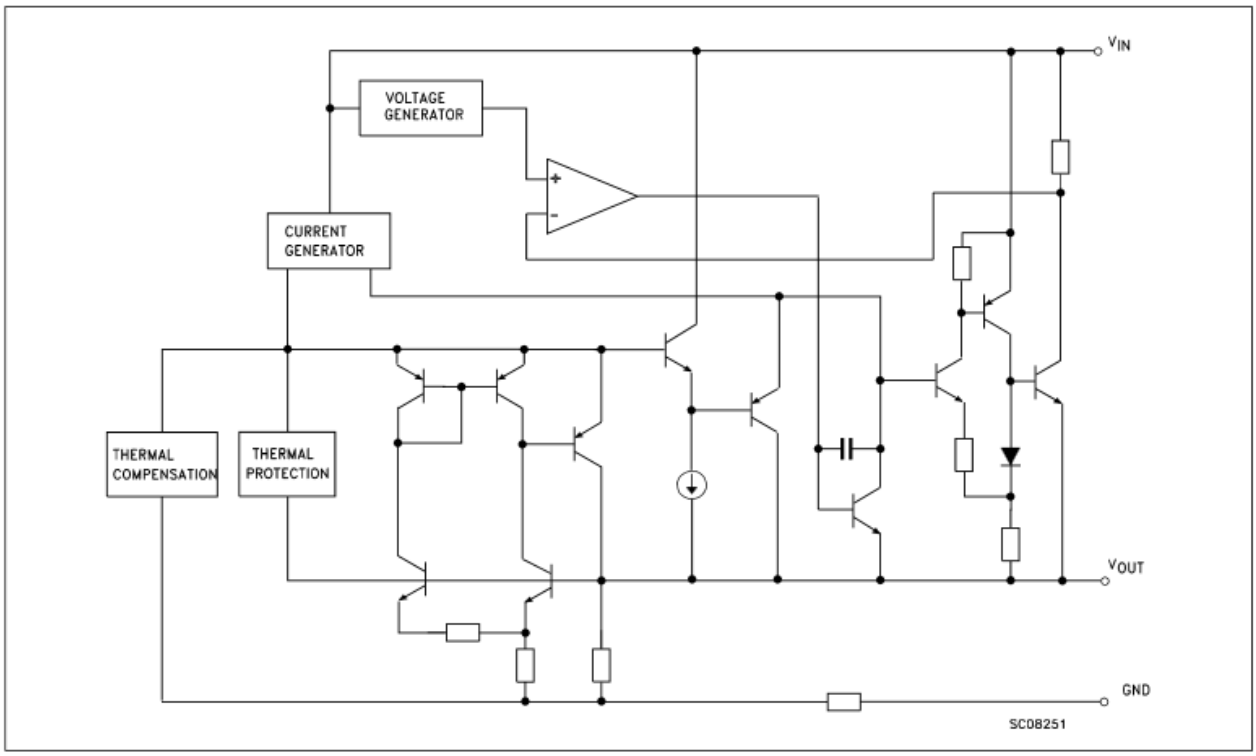
OUTPUT = Pin Τάσης εξόδου (6V)

3.1.7 3. 3V Linear Voltage Regulator LD1117 TO-220

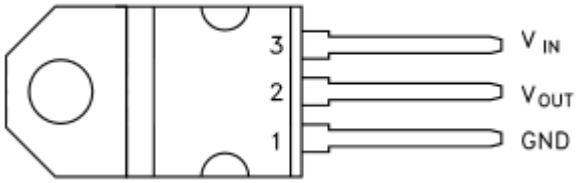
Ο γραμμικός ρυθμιστής τάσης LD33V3 είναι και αυτός τύπου TO-220 και στην κατασκευή χρησιμοποιήθηκε για να παράσχει την απαραίτητη τάση για να τροφοδοτηθεί το ESP8266 αλλά και ο RFID Reader MFRC522.

Χαρακτηριστικά LD1117

- Προστασία από υπερθέρμανση
- Προστασία από βραχυκύκλωμα
- Ρεύμα εξόδου 800mA
- Τάση εξόδου 3. 3V
- Ελάχιστη τάση εισόδου 4. 3V
- Μέγιστη τάση εισόδου 15V
- Μέγιστη θερμοκρασία λειτουργίας 125C



Εικόνα 3.13 *Διάγραμμα εσωτερικής διασύνδεσης LD1117 TO-220 [35]*



Εικόνα 3.14 *Διάγραμμα Pin LD1117 TO-220 [35]*

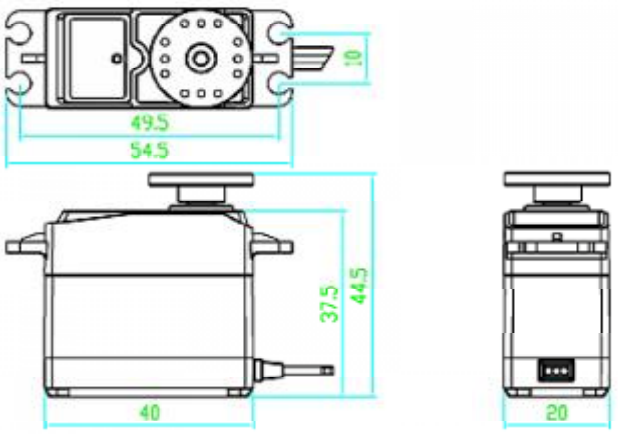
Vin = Pin Τάσης εισόδου (8-35V)

GND = Pin γείωσης

Vout = Pin Τάσης εξόδου (6V)

3.1.8 AS3103 Αναλογικός σερβοκινητήρας ελεύθερης περιστροφής

Εικόνα 3.15 *: AS3103 σερβοκινητήρας [36]*



Εικόνα 3.16 *: Διαστάσεις AS3103 σερβοκινητήρα [37]*

Για την κίνηση των γραναζιών της κατασκευής χρησιμοποιήθηκε ο ακόλουθος σερβοκινητήρας με δυνατότητα ελεύθερης περιστροφής 360 μοιρών καθώς έχει υποστεί τροποποίηση για να μην έχει τον περιορισμό των 180 μοιρών ενός συνηθισμένου σερβοκινητήρα, αφαιρώντας το εσωτερικό ποτενσιόμετρο και την δυνατότητα ανατροφοδότησης και την ακίδα στον τελικό δίσκο μετάδοσης για να κάνει ελεύθερη περιστροφή για να μην υπάρχει το εμπόδιο στην περιστροφική κίνηση.

Χαρακτηριστικά AS3103 Analog Servomotor

Βάρος	38g
Ταχύτητα	0.22sec/60° στα 4.8v & 0.19sec/60° στα 6v
Ροπή	5kg/cm στα 4. 8v - 5.5kg/cm στα 6v
Τάση εισόδου	4.8V-6V
Μοίρες Περιστροφής	360°

Οι συμβατικοί σερβοκινητήρες αποτελούνται από 5 βασικά στοιχεία:

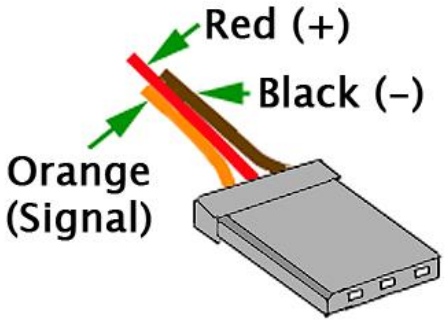
1. Ηλεκτρικό κινητήρα
2. Ποτενσιόμετρο αναφοράς θέσης
3. Ηλεκτρονικό κύκλωμα ανατροφοδότησης
4. Γρανάζια μειωτήρων (πλαστικά η μεταλλικά)
5. Εξάρτημα τελικού γραναζιού μετάδοσης κίνησης (Ενεργοποιητής)

Εικόνα 3.17 *Εξαρτήματα σερβοκινητήρα [38]*

Η θέση της εξόδου μετριέται από το ποτενσιόμετρο του οποίου το αναλογικό σήμα συγκρίνεται διαρκώς με την θέση που δίνει το σήμα ελέγχου (έστω το σήμα από το ESP8266). Οποιαδήποτε διαφορά των σημάτων αυτών αυξάνει το σήμα “λάθους” το οποίο καθοδηγεί τον ηλεκτρικό κινητήρα είτε αριστερόστροφα, είτε δεξιόστροφα το οποίο με τη σειρά του γυρίζει τα γρανάζια των μειωτήρων προς τη σωστή κατεύθυνση. Όταν λοιπόν ο σερβοκινητήρας φτάσει την σωστή θέση, το σήμα λάθους μηδενίζεται και ο σερβοκινητήρας σταματάει να κινείται.

Στην περίπτωση που η θέση του σερβοκινητήρα είναι άλλη από αυτή που έχει οριστεί μέσω του σήματος, είτε γιατί το σήμα ελέγχου άλλαξε η γιατί το γρανάζι γυρίστηκε προς άλλη κατεύθυνση χειροκίνητα τότε το σήμα λάθους θα ξαναεμφανιστεί και ο σερβοκινητήρας θα επιστρέψει στην σωστή θέση.

Όσον αφορά το σήμα ελέγχου του σερβοκινητήρα. Τα Pins που περιλαμβάνει είναι τρία:



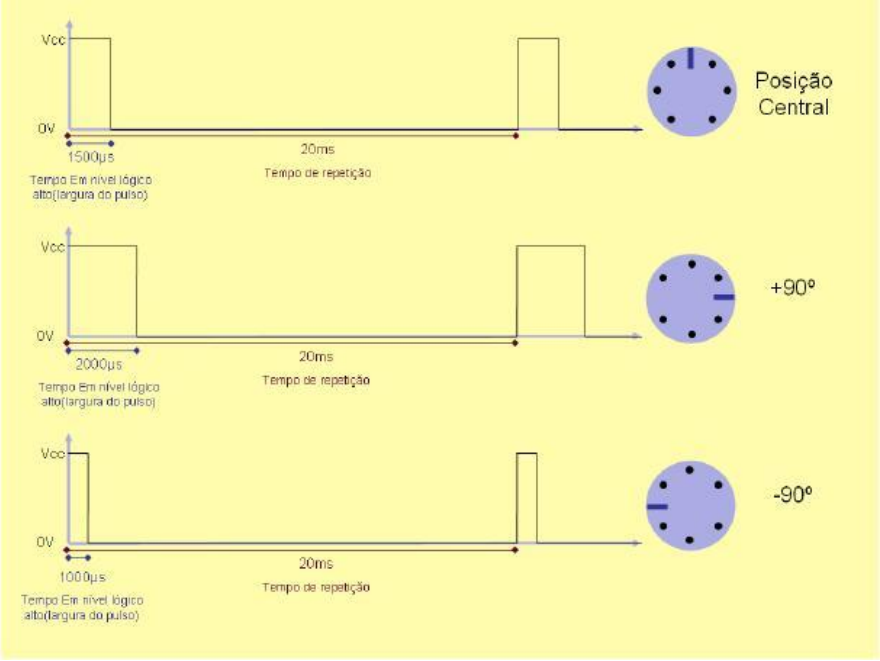
Εικόνα 3.18 *: Διάγραμμα Pin σερβοκινητήρα [38]*

- Το μαύρο (ή καφέ) καλώδιο αντιπροσωπεύει την γείωση
- Το κόκκινο καλώδιο αντιπροσωπεύει το θετικό πόλο για την τάση (4. 8 – 6v)
- Το πορτοκαλί καλώδιο αντιπροσωπεύει το σήμα ελέγχου που συνδέεται σε ένα pin του μικροελεγκτή. Στην περίπτωση μας στο ESP8266

Το σήμα ελέγχου που λαμβάνει ένας σερβοκινητήρας είναι τύπου PWM (Pulse Width Modulation – διαμόρφωση εύρους παλμών). Κάθε 20ms ένας ενεργός ψηφιακός παλμός ελέγχει την τοποθεσία. Ο παλμός αυτός έχει εμβέλεια από 1.0ms μέχρι 2.0ms ενώ πάντα το 1.5ms είναι πάντα το κέντρο της εμβέλειας. Παλμοί πέρα από τους αναγραφόμενους κάνουν το σερβοκινητήρα να γυρίσει πέρα από το επιτρεπόμενο όριο με μεγάλη πιθανότητα να προκαλέσει ζημιά σε όλο το σύστημα. Όπως αναφέρθηκε, ο σερβοκινητήρας περιμένει ενα σήμα ανά 20ms για να αποκτήσει την σωστή πληροφορία που αφορά την γωνία του σερβοκινητήρα, το πλάτος λοιπόν του παλμού καθορίζει κατά πόσες μοίρες θα γυρίσει ένας σερβοκινητήρας.

Συνηθέστερα δεδομένα γωνιών(μεταβάλλονται ανά μοντέλο)

- 1.5ms - 45°
- 1ms - 0°
- 2ms - 180°



Εικόνα 3.19 *: Απεικόνιση συσχέτισης PWM με θέση σερβοκινητήρα [18]*

Εικόνα 3.20 *: Απεικόνιση συχνότητας σήματος PWM [18]*

Στην περίπτωση του σερβοκινητήρα που χρησιμοποιήσαμε όπου είναι ελεύθερης περιστροφής ισχύουν τα ακόλουθα:

- 1ms – Μέγιστη ταχύτητα δεξιόστροφα
- 1ms-1. 49ms – Μεταβλητή ταχύτητα δεξιόστροφα
- 1. 5ms – Ακινησία
- 1. 51ms-1. 99ms – Μεταβλητή ταχύτητα αριστερόστροφα
- 2ms – Μέγιστη ταχύτητα αριστερόστροφα

Για τον έλεγχο ενός συμβατικού σερβοκινητήρα θα χρησιμοποιούσαμε τις ακόλουθες εντολές όπου 0 και 180 οι μοίρες περιστροφής. Έστω ένα βασικό sketch για μετακίνηση από 0 σε 180 μοίρες με καθυστέρηση 5 δευτερολέπτων

```
#include <Servo. h>
Servo myservo;  // create servo object to control a servo

void setup() {
myservo. attach(9);  // attaches the servo on pin 9 to the servo object
myservo. write(0);
delay(5000) //5s Delay
myservo. write(180);
}

void loop() {

}
```

Στην περίπτωση μας 0°=1ms(Δεξιόστροφα) & 180° = 2ms(Αριστερόστροφα) & 90° = 1. 5ms (Ακινησία).

```
#include <Servo. h>
Servo myservo;  // create servo object to control a servo

void setup() {
myservo. attach(9);  // attaches the servo on pin 9 to the servo object
myservo. write(0);  //Start turning at full speed clockwise
delay(5000) //5s Delay //Keep turning for 5s
myservo. write(90);  //stop turning
myservo. detach(); //Detach the pin to cut power
myservo. attach(9);  //reattach on pin 9
myservo. write(180); //Start turning at full speed counterclockwise
delay(3000) //keep turning for 3s
myservo. write(90);  //stop turning
myservo. detach() //
}

void loop() {

}
```

3.1.9 FDC-2S-2 Charge Protection Board

Εικόνα 3.21 *: FDC-2S-2 Charge protection board [39]*

Το συγκεκριμένο Board εχει ως σκοπό την προστασία των 2 μπαταριών λιθίου 18650 κατά την φόρτιση τους όταν είναι συνδεδεμένες σε σειρά. Ως προστασία αντί να ισορροπεί την τάση ανάμεσα στις δύο μπαταρίες κόβει απευθείας την παροχή προς τις μπαταρίες αν

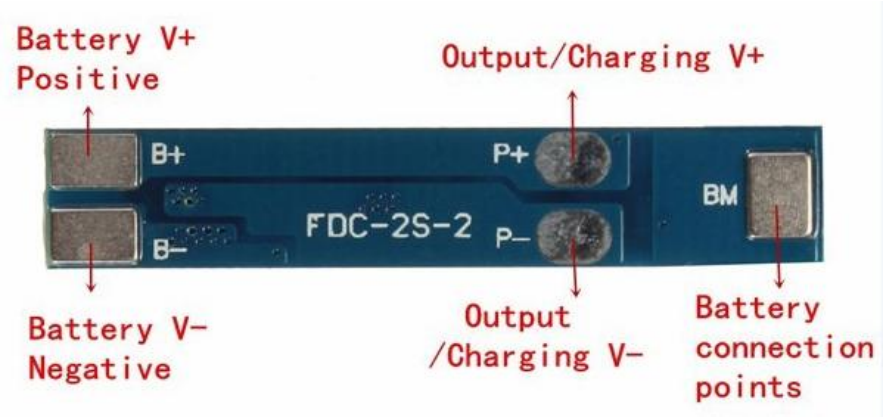
μία από τις δύο ξεπεράσει το όριο των 4.25V. Με τον ίδιο τρόπο κόβει την παροχή σε περίπτωση που αποφορτιστεί μία από τις δύο κάτω από την τάση των 2.3V.

Περισσότερες πληροφορίες σχετικά με την λειτουργία, ενδοκυκλωματική συνδεσμολογία και ποια εξαρτήματα χρησιμοποιεί το Board δεν φαίνεται να υπάρχουν στο διαδίκτυο και οι πωλητές δεν γνωρίζουν περισσότερα πέρα από τα αναγραφόμενα στις περιγραφές του προιόντος.

Υποθετικά το Board χρησιμοποιεί κάποιου είδος τρανζίστορ για να σταματά την παροχή προς τις μπαταρίες όταν ανιχνεύσουν την τάση είτε υπερφόρτωσης είτε αποφόρτισης εφόσον μπορούν να λειτουργήσουν και ως διακόπτες.

Χαρακτηριστικά FDC-2S-2

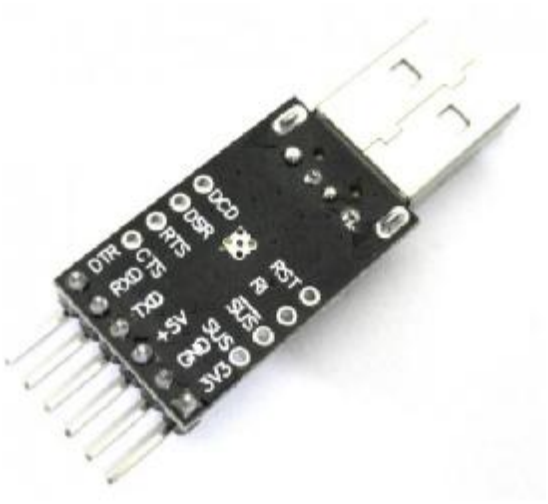
- Προστασία απο βραχυκύκλωση
- Προστασία απο υπερφόρτιση
- Προστασία απο αποφόρτιση
- Προστασία απο παραπάνω από επιτρεπτό ρεύμα
- Τάση ανίχνευσης υπερφόρτιστης : 4.25-4.35V±0.05V
- Τάση ανίχνευσης αποφόρτισης : 2.3-3. 0V±0.05V
- Μέγιστο ρεύμα λειτουργίας: 3A
- Επιτρεπτά όρια θερμοκρασίας : Από -40 έως 50℃
- Εσωτερική αντίσταση : 45 Ω
- Τάση τροφοδοσίας : 8.4V
- Μέση θερμοκρασία λειτουργίας : 25℃



Εικόνα 3.22 *: Τρόπος σύνδεσης FDC-2S-2 σε κύκλωμα [26]*

3.1.10 CP2102 USB 2. 0 to TTL UART Module 6pin Serial Converter

Εικόνα 3.23 *: CP2102 USB-To-TTL UART Module [40]*



Εικόνα 3.24 *: CP2102 Pins [40]*

Το CP2102 χρησιμοποιήθηκε για την σειριακή επικοινωνία με πρωτόκολλο UART μεταξύ υπολογιστή και ESP8266 με σκοπό τον προγραμματισμό και ρύθμιση του ESP8266.

Τα pins του CP2102 όπου φαίνονται και στην εικόνα 3.23 (CP2102 Pins) είναι τα ακόλουθα :

DTR ή RST Γραμμή Reset για επανεκκίνηση μετά την μεταφορά δεδομένων

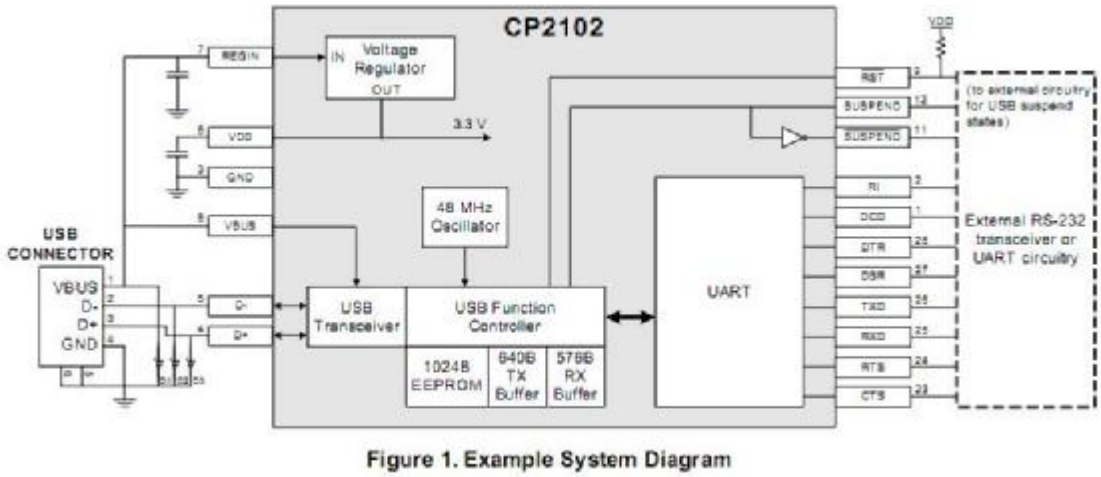
RXD UART0_RXD [Pin Μεταφοράς δεδομένων UART – Receive Line]

TXD UART0_TXD [Pin Μεταφοράς δεδομένων UART – Transmit Line]

+5V Παροχή τάσης 5V +

GND Γείωση -

3V3 Παροχή τάσης 3.3V +



Εικόνα 3.25 *: Διάγραμμα κυκλώματος CP2102 [40]*

3.1.11 18650 3. 7V 2650mAh Samsung Rechargeable Battery

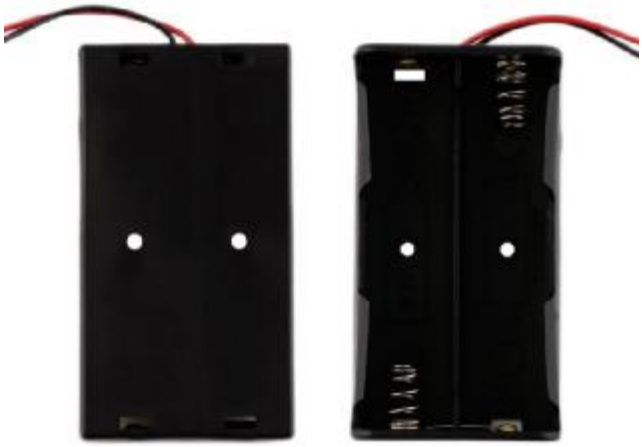
Η παροχή ενέργειας του ηλεκτρονικού κυκλώματος έγινε με δύο επαναφορτιζόμενες μπαταρίες λιθίου (li-on) τύπου 18650. Συγκεκριμένα χρησιμοποιήθηκαν για να δώσουν την απαραίτητη ενέργεια για να τροφοδοτηθούν το ESP8266, το RFID MRC522 και ο σερβοκινητήρας AS3103.

Εικόνα 3.26 *: 18650 Li-on Samsung batteries [41]*

Οι μπαταρίες λιθίου (li-on) είναι τύποι επαναφορτιζόμενων μπαταριών όπου τα ιόντα λιθίου μετακινούνται από το αρνητικό ηλεκτρόδιο στο θετικό ηλεκτρόδιο κατά τη διαδικασία αποφόρτισης και αντίθετα κατά την φόρτιση. Οι μπαταρίες λιθίου χρησιμοποιούν μια μίξη στρωμάτων λιθίου ως το υλικό ηλεκτροδίων σε σχέση με το μεταλλικό λίθιο που χρησιμοποιείται στις μη επαναφορτιζόμενες μπαταρίες λιθίου. Ο ηλεκτρολύτης που επιτρέπει την κίνηση των ιόντων και τα δύο ηλεκτρόδια είναι τα βασικά συστατικά μίας μπαταρίας λιθίου

3.1.12 18650 Dual Battery Series Case

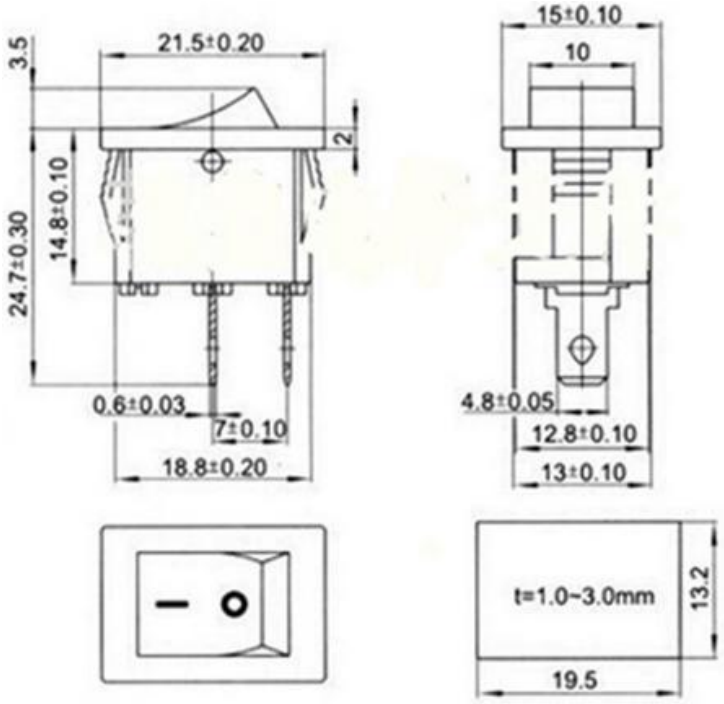
Η θήκη για τις δύο 18650 μπαταρίες αποθηκεύει και συνδέει τις δύο μπαταρίες σε σειρά. Είναι κατασκευασμένη από ABS πλαστικό έχει διαστάσεις 75mm x 40mm x 20mm και βάρος 14g



Εικόνα 3.27 *: 18650 Dual Battery Series Case [42]*

3.1.13 Black On/Off Switch KCD1-101

Εικόνα 3.28 *: KCD1-101 On/Off Switch [43]*



Εικόνα 3.29 *Διαστάσεις KCD1-101 [43]*

Το εξάρτημα KCD1-101 είναι ένας κλασικός διακόπτης διακοπής τροφοδοσίας ON/OFF. Ο διακόπτης περιλήφθηκε για να σταματά την παροχή ενέργειας στο κύκλωμα. Με τον τρόπο αυτό αποκτούμε δύο χρήσιμες λειτουργίες :

1)Ως κουμπί επανεκκίνησης (Reset)

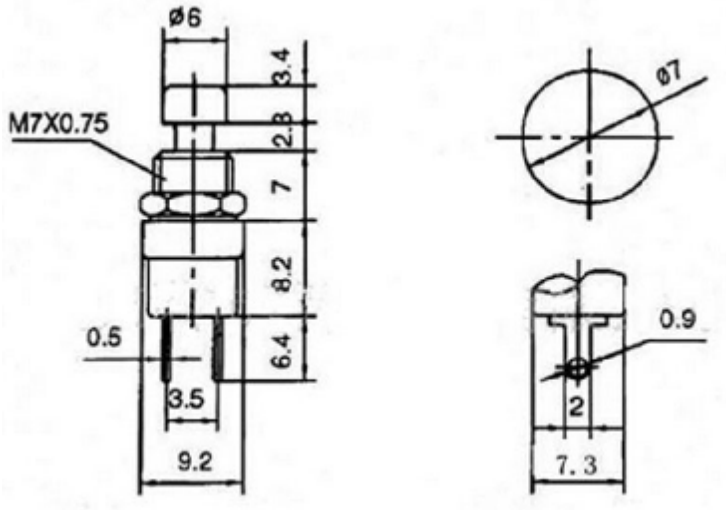
2)Να κλείνει η τροφοδοσία αν πρέπει να γίνουν ρυθμίσεις/τροποποιήσεις (αλλαγές στο κλειδί/αλλαγή μπαταριών κ.τ.λ.).

Τεχνικά χαρακτηριστικά KCD1-101

Αντίσταση επαφής	≤ 20 MΩ
Μονωτική Αντίσταση	≥ 100MΩ
Διηλεκτρική ένταση	≥ 1800 VAC/3S
Χρόνος ζωής	≥10000 φορές

3.1.14 Momentary Push Button PBS-110

Εικόνα 3.30 *: PBS-110 Momentary Push Button [44]*



Εικόνα 3.31 *: Διαστάσεις PBS-110 [44]*

Ο διακόπτης PBS-110 χρησιμοποιήθηκε ως κουμπί για να κλειδώνει και να ξεκλειδώνει η κλειδαριά από την εσωτερική πλευρά της πόρτας χωρίς να χρειάζεται να έχουμε πρόσβαση στο κλειδί η στο κινητό μας τηλέφωνο μέσω της εφαρμογής ελέγχου. Ο διακόπτης αυτός περιλαμβάνει ένα ελατήριο για να επιστρέφει στην αρχική του θέση όταν πιεστεί. Όταν το κουμπί πιέζεται η σύνδεση μεταξύ των Pin στο κάτω μέρος του PBS-110 διακόπτεται εσωτερικά και η πόρτα κλειδώνει η ξεκλειδώνει αναλόγως την κατάσταση στην οποία βρίσκεται. Το ESP8266 αναλαμβάνει τον ρόλο του να ελέγχει, αν και πότε διακόπτεται η σύνδεση αυτή, έτσι μέσω του προγραμματισμού του, όταν ανιχνεύσει διακοπή της σύνδεσης στέλνει το σήμα για να κλειδώσει η να ξεκλειδώσει η κλειδαριά πόρτα μέσω του σερβοκινητήρα.

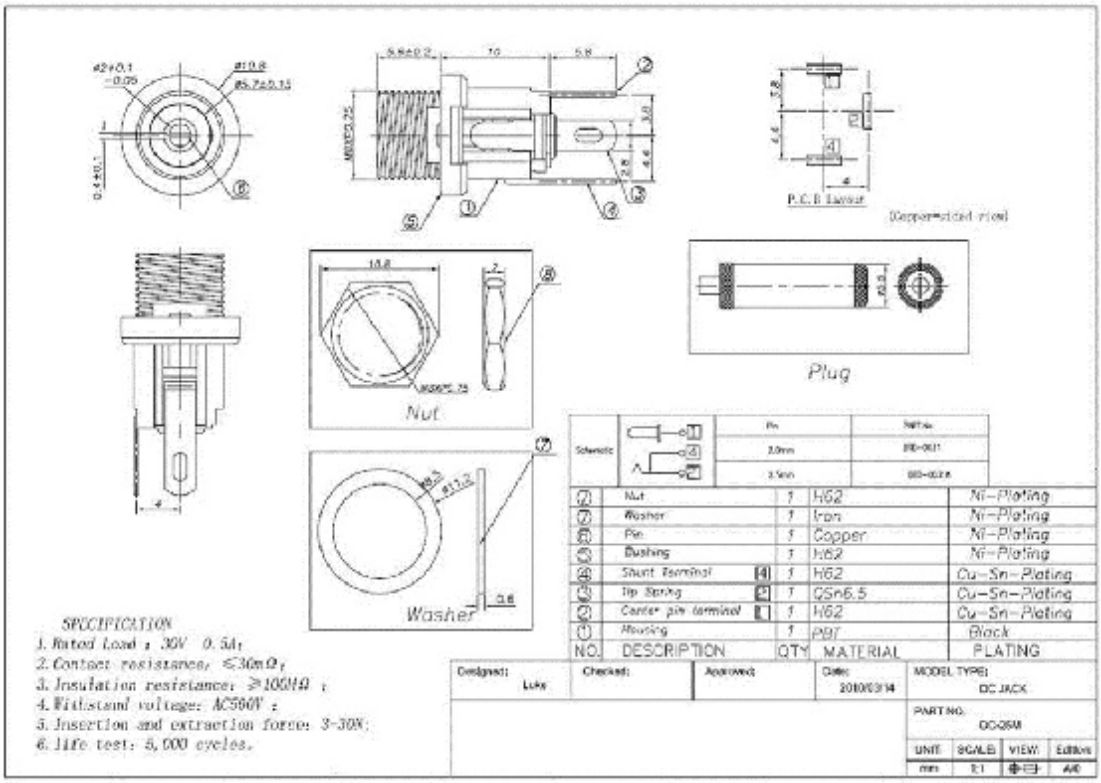
Τεχνικά Χαρακτηριστικά PBS-110

- Μέγιστη επιτρεπόμενη τάση 12V
- Μέγιστο επιτρεπόμενο ρεύμα 3A

3.1.15 DC Power Jack Socket Female DC-25M

Εικόνα 3.32 *: DC Power Jack Socket Female DC-25M [46]*

Ο DC-25M είναι ένας υποδοχέας ρεύματος ο οποίος αξιοποιήθηκε για να παρέχει την απαραίτητη τροφοδοσία στις μπαταρίες λιθίου 18650 από τον φορτιστή EU-9V-2A-2507 μέσω του FDC-2S-2 ώστε να τις επαναφορτίσει σε περίπτωση όπου η τάση μειωθεί σε μη αποδεκτά επίπεδα.



Εικόνα 3.33 *: Διαστάσεις & Προδιαγραφές DC-25M [46]*

Τεχνικά Χαρακτηριστικά

- **Μέγιστο φορτίο** : 30V 0.5A
- **Μέγιστη τάση** : AC 500V

3.1.16 9V 2A Charger EU-9V-2A-2507

Εικόνα 3.34 *: Φορτιστής EU-9V-2A-2507 [45]*

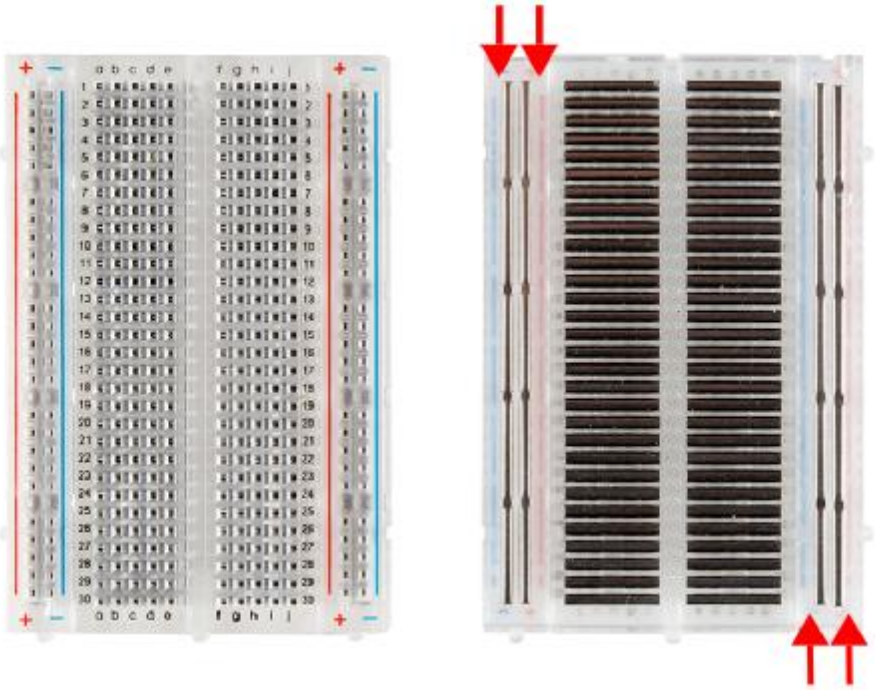
Ο φορτιστής EU-9V-2A-2507 χρησιμοποιήθηκε στην κατασκευή για να παρέχει φόρτιση στις μπαταρίες λιθίου 18650 μέσω του υποδοχέα DC-25M στη συνέχεια στο board

προστασίας υπερφόρτισης/αποφόρτισης FDC-2S-2 και έπειτα στις μπαταρίες που αναφέρθηκαν.

Τεχνικά χαρακτηριστικά

Είσοδος : AC 100-240V 50/60Hz
Έξοδος : 9V 2A

3.1.17 Breadboard



Εικόνα 3.35 *: Breadboard [48]*

Ένα Breadboard έχει ως σκοπό να λειτουργεί ως η βάση για την περίοδο όπου το ηλεκτρονικό κύκλωμα δεν έχει καταλέξει στο τελικό του στάδιο. Το Breadboard είναι φτιαγμένο από διάτρητο πλαστικό και οι επαφές είναι συνήθως από ασημί νικέλιο το οποίο είναι ένα κράμα χαλκού. Ένα Breadboard αποτελείται από τις πλαϊνές λωρίδες που προορίζονται για την παροχή τροφοδοσίας στο κύκλωμα και τις εσωτερικές για τα διάφορα εξαρτήματα που χρησιμοποιεί το κύκλωμα όπως αντιστάσεις, πυκνωτές, καλώδια επαφών-δοκιμών (Jumper Wires) και άλλα. Η σύνδεση των λωρίδων και των εξαρτημάτων που έχουν τοποθετηθεί σε αυτές γίνεται με την βοήθεια των Jumper Wires συμπληρώνοντας ένα δοκιμαστικό κλειστό κύκλωμα.

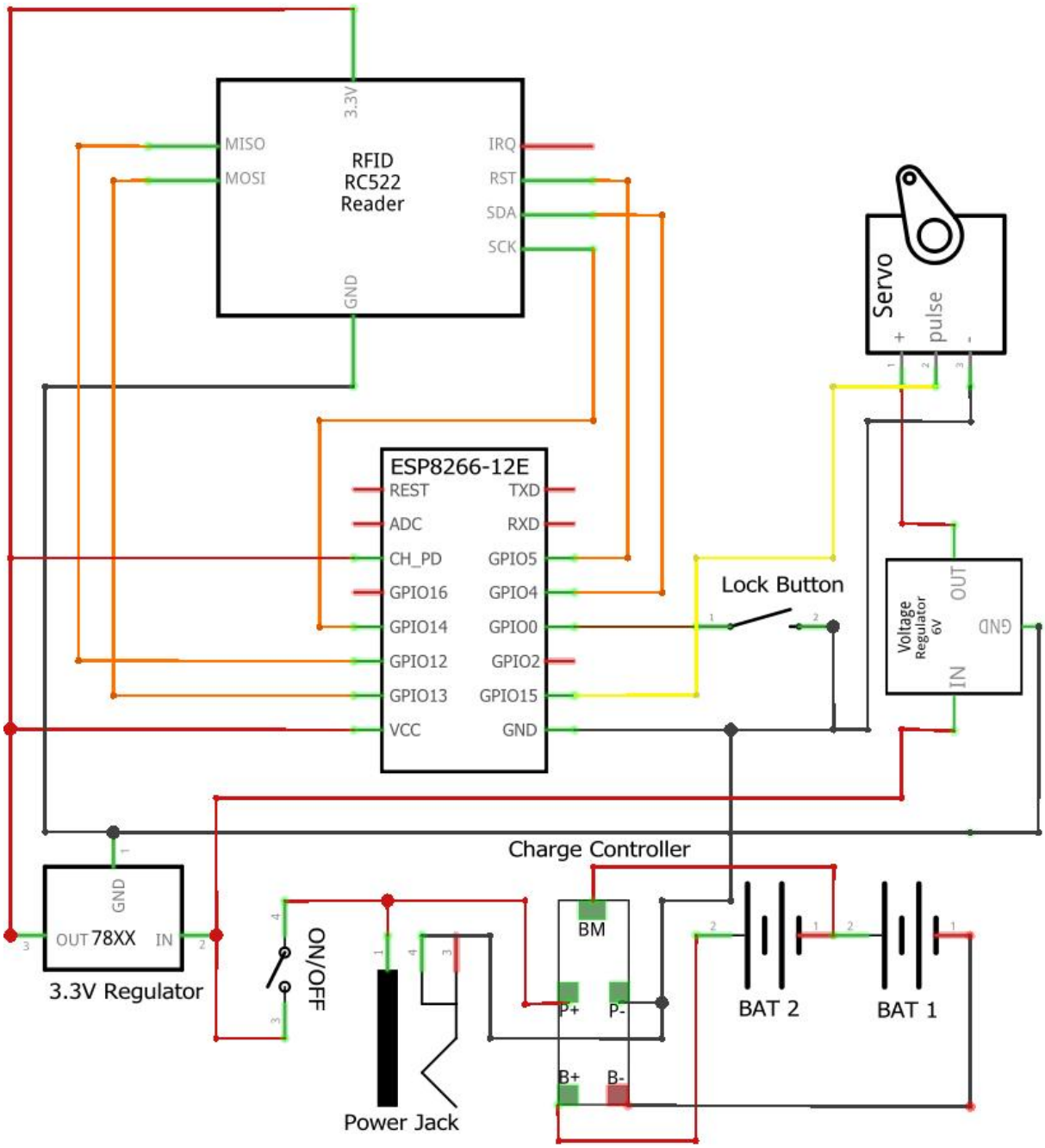
3.1.18 Jumper Wires

Εικόνα 3.36 *: Jumper wires [47]*

Τα Jumper wires η καλώδια επαφών/δοκιμών είναι τα καλώδια τα οποία χρησιμοποιούνται κατά την δοκιμαστική περίοδο για την δημιουργία επαφών μεταξύ Pin και άλλων σημείων για την δημιουργία κυκλώματος σε συνδυασμό με ένα Breadboard, που αναφέρθηκε προηγουμένως.

3.2 Συνδεσμολογία Ηλεκτρονικού Κυκλώματος

Στο υποκεφάλαιο αυτό θα αναλυθεί η συνδεσμολογία του ηλεκτρονικού κυκλώματος που χρησιμοποιήθηκε για την δημιουργία της τελικής κατασκευής. Στην ακόλουθη εικόνα φαίνεται το κύκλωμα το οποίο σχεδιάστηκε με την βοήθεια του προγράμματος Fritzing.



Εικόνα 3.2.1 *: Συνδεσμολογία ηλεκτρονικού κυκλώματος κατασκευής*

3.3V Regulator	3.3V Linear Voltage Regulator LD1117 TO-220
ON/OFF	Black On/Off Switch KCD1-101
Power Jack	DC Power Jack Socket Female DC-25M
Charge Controller	FDC-2S-2 Charge Protection Board
Bat1	18650 3.7V 2650mAh Samsung Rechargeable Battery
Bat2	18650 3.7V 2650mAh Samsung Rechargeable Battery

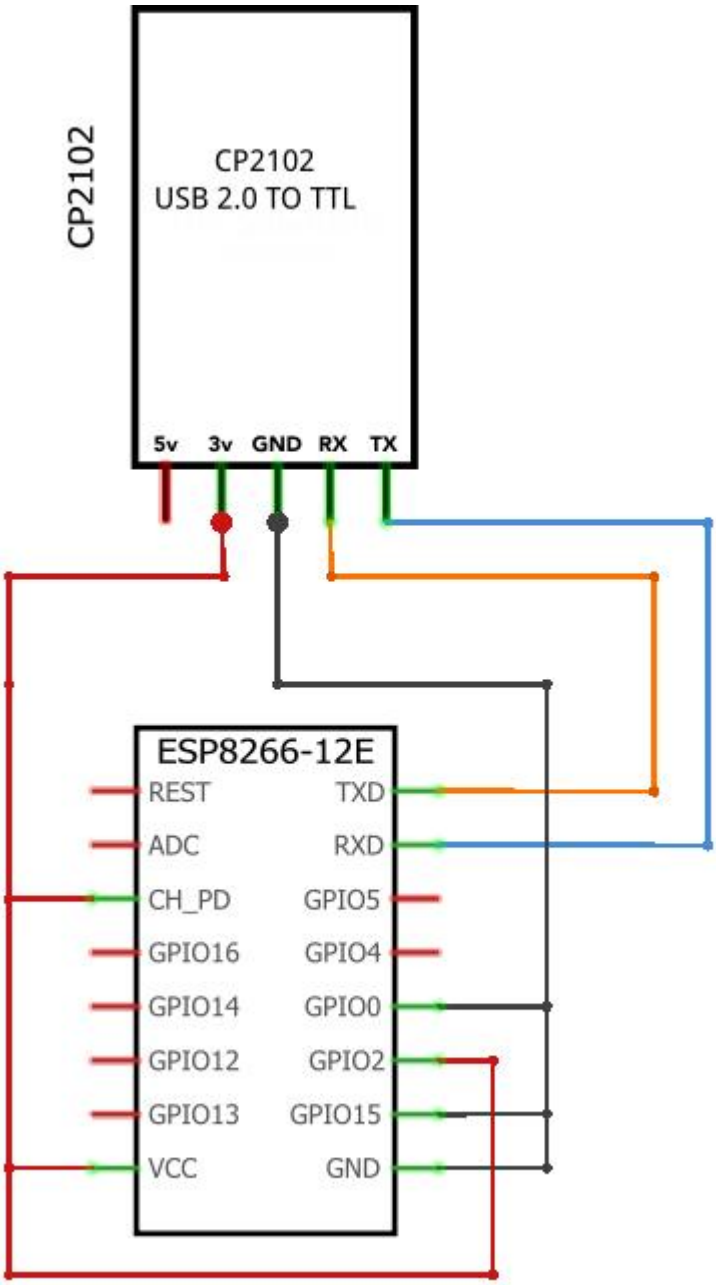
Voltage Regulator 6V	6V Voltage Regulator L7806CV TO-220
Lock Button	Momentary Push Button PBS-110
ESP8266-12E	ESP8266 – 12E
Servo	AS3103 Analog Servo Continuous Rotation
RFID RC522 Reader	RFID MFRC522 Reader/Writer

Επεξηγήσεις Συνδεσμολογίας Κυκλώματος (Εικόνα 3.2.1):

- Οι διαγραμμίσεις με κόκκινο χρώμα σηματοδοτούν την τάση, είτε είναι 3.3V, 6V, 9V, 8.4V
- Οι διαγραμμίσεις με μαύρο χρώμα σηματοδοτούν την γείωση.
- Οι διαγραμμίσεις με πορτοκαλί χρώμα σηματοδοτούν τις ενώσεις του RFID για την επικοινωνία με το ESP8266.
- Η διαγράμμιση με κίτρινο χρώμα σηματοδοτεί το σήμα που δέχεται ο σερβοκινητήρας.
- Η διαγράμμιση με καφέ χρώμα σηματοδοτεί τη σύνδεση του κουμπιού κλειδώματος/ξεκλειδώματος της πόρτας όπου όταν το κουμπί «πατιέται» κόβεται η σύνδεση.
- Η παροχή τάσης 3.3V από την έξοδο του 3.3V LD1117 Regulator δίνει την απαραίτητη τροφοδοσία παράλληλα στα δυο pins του ESP8266 αλλά και στο RFID RC522 καθώς απαιτούν τάση 3.3V
- Η σύνδεση του κουμπιού ON/OFF έγινε με τέτοιο τρόπο ώστε να σταματά οποιαδήποτε μορφή παροχής τροφοδοσίας ενέργειας στο κύκλωμα. Όπως φαίνεται και στο σχεδιάγραμμα τοποθετήθηκε μετά από την τροφοδοσία της μπαταρίας αλλά και της παροχής του φορτιστή 9V μέσω της υποδοχής DC.
- Οι δύο μπαταρίες είναι συνδεδεμένες σε σειρά για να δίνουν την απαραίτητη τάση για την τροφοδοσία του συστήματος. Ο θετικός πόλος καταλήγει B+ του FDC-2S-2 και ο αρνητικός στο B-. Ενώ η ενδιάμεση σύνδεση των 2 μπαταριών καταλήγει στο BM του FDC-2S-2.
- Η παροχή τάσης 6V επιτυγχάνεται από την έξοδο του 6V L7806CV Regulator και απευθείας στον σερβοκινητήρα ο οποίος δέχεται τάση 4.8V έως 6V με τα 6V να αποδίδουν την μέγιστη ταχύτητα του σερβοκινητήρα.

Οι υπόλοιπες συνδέσεις αφορούν την επικοινωνία μεταξύ RFID και ESP8266.

Η συνδεσμολογία για την οποία έγινε η περιγραφή αφορά καθαρά και μόνο το τελικό σκέλος για την σωστή λειτουργία της κατασκευής. Όμως προκειμένου να προγραμματίσουμε το ESP8266 η συνδεσμολογία αλλάζει και γίνεται ως εξής στην εικόνα 3.2.2.



Εικόνα 3.2.2 *: Συνδεσμολογία CP2102 με ESP8266-12E*

Το CP2102 μας δίνει την επιλογή ανάμεσα σε 5V και 3.3V. Επιλέγουμε ως παροχή τα 3.3V και συνδέουμε τα Pins – CH_PD, VCC και GPIO2 στο pin 3.3V του CP2102. Αναλόγως συνδέουμε τα Pins – GPIO0, GPIO15 και GND με το Pin GND του CP2102. Αφού κάνουμε τις απαραίτητες συνδέσεις συνδέουμε το CP2102 στον υπολογιστή μας και μπορούμε να “ανεβάσουμε” το Sketch που θέλουμε.

3.3 Κώδικας Προγράμματος (Sketch) & Blynk Interface

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο η γλώσσα προγραμματισμού που χρησιμοποιεί το Arduino είναι η Wiring* και εφόσον το ESP8266 προγραμματίστηκε με Arduino SDK αυτή χρησιμοποιήθηκε και εδώ.

Το ακόλουθο Sketch είναι αυτό που χρησιμοποιήθηκε για τον έλεγχο του ESP8266 και της τελικής κατασκευής. Παρόλο που στο διαδίκτυο είναι αναρτημένα διάφορα παραδείγματα για την σωστή αξιοποίηση της κάθε λειτουργίας τόσο για την επικοινωνία του Blynk με το Hardware, όσο και για τα επιμέρους φυσικά εξαρτήματα, ο κώδικας που χρησιμοποιήθηκε στο πρόγραμμα τροποποιήθηκε σε ποσοστό 99% για την σωστή λειτουργία του μηχανισμού, καθώς δεν υπάρχει τίποτα έτοιμο που να συνδυάζει τις συσκευές που χρησιμοποιήθηκαν στην πτυχιακή εργασία αλλά και πολλές φορές οι πληροφορίες του διαδικτύου είναι ελλιπής και λάθος. Τελικώς, τα μόνα κομμάτια που πάρθηκαν αυτοτελή ήταν τα «`Look for new cards`» & «`Select one of the cards`». Ο παρακάτω κώδικας περιέχει σχολιασμούς για επεξήγηση σε κάθε σκέλος.

```
/* GPIO Pins To ESP8266 (ESP-12E)
2 -
16 Lock State
4 SDA(SS)
5 RST
12 MISO
13 MOSI
14 SCK
15 ServoMotor (used to be 0)
0 Physical Button (Connect from GPIO_0 to Ground)
*/
#include <EEPROM. h>     // We are going to read and write PICC's UIDs
from/to EEPROM
#include <MFRC522. h>  // Library for Mifare RC522 Devices
#include <Servo. h>
#include <ESP8266WiFi. h>
#include <SPI. h>
#include <BlynkSimpleEsp8266. h>
#include <SimpleTimer. h>

//Blynk Connection Related
char auth[] = "2019cckee9634001a7213141fc142db0"; //Changed for Security
Reasons
char ssid[] = "SSID_NAME_HERE"; //Changed for Security Reasons
char pass[] = "SSID_PASSWORD_HERE"; //Changed for Security Reasons
bool Connected2Blynk;
bool Connect2Blynk;

//ESP Read Battery voltage to get VCC//
ADC_MODE(ADC_VCC);
float espvolt = ESP. getVcc();
float espcritical = 2950; //Lowest Possible Voltage for Normal operation
float esplastvoltreading = 0;

//SimpleTimer Define
SimpleTimer timer;

//Virtual Pins Declaration
WidgetLED virtualgreenled(V4);
WidgetLED virtualredled(V5);
```

```
// Servo Declaration
int servopin = 15;
Servo servo;

//Lock - Unlock physical Button
int pushbutton = 0;

//Lock Status Physical Pin
int lockpin = 16;

//Virtual Serial Terminal to Pin V8
WidgetTerminal terminal(V8);

//Push Notifications Starting Variables
int doorstate;
int lastdoorstate;

//MFRC522 PIN SETTINGS
#define SS_PIN  4  // SDA-PIN for RC522 - RFID - SPI - Modul GPIO4
#define RST_PIN  5 // RST-PIN for RC522 - RFID - SPI - Modul GPIO5
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

////////////////////////// INITIAL SETUP //////////////////////////

void setup() {

//DOOR MUST BE UNLOCKED AND KEY IN PROPER POSITION WHEN INSTALLING THE LOCK

  //Pin Setup
  digitalWrite(servopin, LOW); //GPIO15 Servopin Pulled Low
  delay(1100); //Delay to make sure the 250ms limit is through for GPIO_0
to work normally
  pinMode(lockpin, OUTPUT);  //GPIO_16 LOCK_STATUS PIN SET OUTPUT
  pinMode(pushbutton, INPUT_PULLUP); //GPIO_0 LOCK/UNLOCK PHYSICAL BUTTON

  //Startup Lock Status Pin Config To Store Locked/Unlocked Status
  digitalWrite(lockpin, LOW); //GPIO_16 Set Low

  //Routine Call To connect to WIFI & BLYNK
  FirstConnection();

  //RFID RC522 SETUP AND PRINT TO BLYNK TERMRINAL
  Serial. begin(115200);  // Serial Comm Start in data rate in bits per
second(BAUD)
  terminal. println(F("Booting"));
  terminal. println(F("Blynk v" BLYNK_VERSION ": Device started"));
  terminal. println("-------------");
  terminal. println("BlynkStatus : Connected");
  terminal. flush();
  SPI. begin(); // Init SPI bus
  mfrc522. PCD_Init(); // Init MFRC522
  terminal. println(F("========================"));
  terminal. println(F("Ready To Scan For Cards"));
  terminal. flush();

  //Timer Subroutine Call in ms
  timer. setInterval(30000, CheckConnection); // check if still connected
every 30 seconds
  timer. setInterval(2000, checkPin);
  timer. setInterval(25000, voltageread);
  digitalWrite(servopin, HIGH); //Servopin Pulled High Again
```

```
 }


//////////////////////////// Main Loop //////////////////////
void loop() {
  if(Blynk. connected()){
    Blynk. run();  // only process Blyk. run() function if connected to
Blynk server
  }
  //Timer Function runs constantly
  timer. run();
  //PHYSICAL BUTTON FUNCTION CALL
  physicalbutton();
  //Blynk Notification FUNCTION CALL
  pushblynk();
  //POWER WARNING FUNCTION CALL
  powerwarning();

  // Look for new cards
  if ( ! mfrc522. PICC_IsNewCardPresent()) {
    delay(50);
    return;
  }
  // Select one of the cards
  if ( ! mfrc522. PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  if (mfrc522. uid. uidByte[0] == 0x55 &&
     mfrc522. uid. uidByte[1] == 0x4F &&
     mfrc522. uid. uidByte[2] == 0x1C &&
     mfrc522. uid. uidByte[3] == 0x53) {
     terminal. println(F("UID Match - Access Granted"));
     terminal. flush();
     if (digitalRead(lockpin) == LOW){
// GPIO_2 LOCK STATUS READ - LOW means unlocked
          tempUnlocking();
          }else{
          permUnlocking();
        }

  }else if(mfrc522. uid. uidByte[0] != 0x55 &&
     mfrc522. uid. uidByte[1] != 0x4F &&
     mfrc522. uid. uidByte[2] != 0x1C &&
     mfrc522. uid. uidByte[3] != 0x53) {
    terminal. println(F("Access Denied - UID does Not Match"));
    terminal. flush();
  }
}
//////////////////////// FIRST CONNECTION ROUTINE ///////////////////////
void FirstConnection(){
  Serial. println("FirstConnectionRoutine");
  Serial. println(WiFi. status());
  WiFi. begin(ssid, pass);
  Serial. println(WiFi. status());
  Serial. println(WiFi. localIP());
  if (WiFi. status() == 6){
    Serial. println("Wifi Connecting");
    Serial. println("Blynk Next");
    Serial. println(WiFi. localIP());
```

```
    Blynk.config(auth);
    Connect2Blynk = Blynk.connect(1000);
    if (Connect2Blynk == true){
      Serial.println("Blynk Connected");
    }else{
      Serial.println("Blynk Failed");
    }
  }else if (WiFi.status() == 1){
    Serial.println("Wifi Unavailable");
  }
 ////////////////// CHECK WIFI AND BLYNK CONNECTION ////////////////////
}void CheckConnection(){
  Serial.println("Checking Connection");
  Connected2Blynk = Blynk.connected();
  if (WiFi.status() == 6){
    WiFi.begin(ssid, pass);
    Serial.println("Connecting to WIFI");
    Serial.println(WiFi.status());
    if (!Connected2Blynk){
      Blynk.config(auth);
      Blynk.connect(1000);
      Serial.println(WiFi.status());
    }else{
      Serial.println(WiFi.status());
    }
  }else if(WiFi.status() == 1){
    Serial.println("Wifi not available. Try Later");
    Serial.println("WIFI Status CODE:");
  }else if(WiFi.status() == 3) {
    Serial.println("Still connected to WIFI");
    Serial.println(WiFi.status());
    if (!Connected2Blynk){
      Blynk.config(auth);
      Blynk.connect(1000);
    }else{
      Serial.println(WiFi.status());
    }
  }else{
    Serial.println("Last Case");
  }
}

//Battery Voltage Info Send To LCD
void voltageread(){
  Blynk.virtualWrite(V7, espvolt);
}

//Lock Status - GPIO_2 Read
void checkPin()
{
  if (digitalRead(lockpin) == LOW) // GPIO_2 Read lock status
  {
    //Means Door Is unlocked
    virtualgreenled.off(); //Turn Virtual Green Led Off
    virtualredled.on(); //Turn Virtual Red Led On
  }else{
    //Means Door Is locked
    virtualgreenled.on(); //Turn Virtual Green Led On
    virtualredled.off(); //Turn Virtual Led Off
  }
}
```

```
//V1_TEMPORARY 3 Second UNLOCK when door is unlocked-FreeRotatingServo
//LOW means unlocked
//HIGH means locked
void tempUnlocking(){
    terminal. println(F("V1Door TempUnlocks - Wait 3 Seconds"));
    terminal. flush();
    servo. attach(servopin);
    servo. write(180);
    delay(1000); // Time for Latch Bolt [Glossa] to get pulled in if door
was unlocked


    servo. detach();
    delay(3000);//Latch bolt stays back for 3s
    servo. attach(servopin);
    servo. write(0);
    delay(1000);
    servo. detach();
    terminal. println(F("V1TempUnlock Worked"));
    terminal. flush();
}
//V2 TEMPORARY 3 Second UNLOCK when door is locked-FreeRotatingServo
//LOW means unlocked
//HIGH means locked
void permUnlocking(){
    terminal. println(F("V2Door TempUnlocks - Wait 6 Seconds"));
    terminal. flush();
    servo. attach(servopin);
    servo. write(180);
    delay(3000); // Time for Latch Bolt [Glossa] to get pulled in if door
was locked
    servo. detach();
    digitalWrite(lockpin, LOW); //Write GPIO_16 Low To Show door unlocked
in checkpin Subroutine
    delay(3000);// Latch Bolt [Glossa] stays back for 3s
    servo. attach(servopin);
    servo. write(0);
    delay(1000);
    servo. detach();
    terminal. println(F("V2-TempUnlock Worked"));
    terminal. flush();
}
// UNLOCKING Routine-FreeRotatingServo
//LOW means unlocked
//HIGH means locked
void unlocking(){
  terminal. println(F("Door Unlocks - Wait 2 Seconds"));
  terminal. flush();
  servo. attach(servopin);
  servo. write(180);
  delay(2000);
  servo. detach();
  digitalWrite(lockpin, LOW); //Write GPIO_2 Low To Show door unlocked in
checkpin Subroutine
  terminal. println(F("Door Has Been Unlocked"));
  terminal. flush();
}
// LOCKING Routine-FreeRotatingServo
//LOW means unlocked
//HIGH means locked
void locking(){
```

```
  terminal. println(F("Door locks - Wait 4 Seconds"));
  terminal. flush();
  servo. attach(servopin);
  servo. write(0);
  delay(2000);
  servo. detach();
  digitalWrite(lockpin, HIGH); //Write GPIO_2 HIGH To Show door locked in
checkpin Subroutine
  terminal. println(F("Door Has Been Locked"));
  terminal. flush();
}
// Blynk Button LOCKING/UNLOCKING
BLYNK_WRITE(V3) //WIDGET BUTTON NEEDS TO BE SWITCH NOT PUSH
{
  if (param. asInt() == 0)
  {
    if (digitalRead(lockpin) == LOW){
      locking();
    }
    else{
      unlocking();
    }
  }
  else
  {
    if (digitalRead(lockpin) == LOW){
      locking();
    }
    else{
      unlocking();
    }
  }
}
//Blynk Button TEMPLOCKING/TEMPUNLOCKING
BLYNK_WRITE(V6)
{
  if (param. asInt() == 0)
  {
    if (digitalRead(lockpin) == LOW){
      tempUnlocking();
    }
    else{
      permUnlocking();
    }
  }
  else
  {
    if (digitalRead(lockpin) == LOW){
      tempUnlocking();
    }
    else{
      permUnlocking();
    }
  }
}
//LOCK - UNLOCK PHYSICAL BUTTON
//LOW means unlocked
//HIGH means locked
void physicalbutton() {
if (digitalRead(pushbutton) == LOW){
  if (digitalRead(lockpin) == LOW){
```

```
      locking();
  }else{
      unlocking();
    }
  }else{
    //Do nothing
  }
}
//PUSH NOTIFICATION BLYNK LOCK STATUS
void pushblynk(){
doorstate = digitalRead(lockpin);
if (doorstate != lastdoorstate){
  if (digitalRead(lockpin) == LOW) {
    Blynk. notify("Door is Unlocked");
  }else{
    Blynk. notify("Door is Locked");
  }
}
lastdoorstate = doorstate;
}
//PUSH NOTIFICATION BLYNK VOLTAGE WARNING
void powerwarning(){
if (espvolt <= espcritical){
   if ((espvolt != esplastvoltreading) && (espvolt <= espcritical)){
    Blynk. notify("Warning - Power Is Running Low");
    }else{
     //do nothing
    }
  }
esplastvoltreading = espvolt;
}
```

Ανάλυση και επεξήγηση του Sketch

Ο κώδικας ξεκινά με τη βασική δήλωση των μεταβλητών μέχρι το σημείο του void setup.

Η ρουτίνα setup περιλαμβάνει όλες εκείνες τις εντολές που θα τρέξουν όταν ενεργοποιηθεί το ESP8266. Χαρακτηρίζεται λοιπόν ως το σετάρισμα ή η αρχική ρύθμιση του συστήματος. Οι εντολές στο σημείο τρέχουν μόνο μια φορά.

Στη συνέχεια έρχεται η βασική ρουτίνα του βρόχου («loop»), που όπως υποδεικνύει και η λέξη οι εντολές μέσα σε αυτή θα τρέχουν ξανά και ξανά, όπως ο συνεχής έλεγχος για ύπαρξη παθητικών ετικετών στο RFID Reader, την υπορουτίνα timer κ.τ.λ.

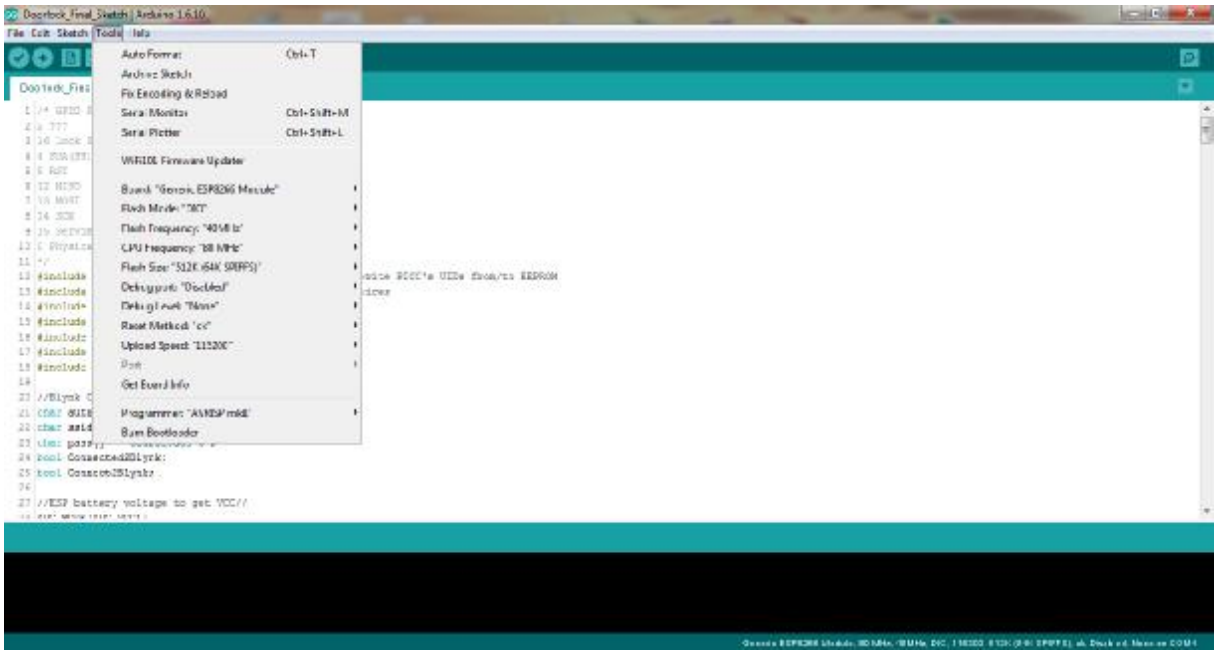
Και ακολουθούν οι υπορουτίνες οι οποίες είναι για

- Την πρώτη προσπάθεια σύνδεσης με το δίκτυο - FirstConnection
- Επανειλημμένο έλεγχο για σύνδεση σε ασύρματο δικτυο Wi-Fi και στο server του blynk – CheckConnection
- Ανάγνωση και αποστολή στάθμης της μπαταρίας σε millivolt σε widget του Blynk – Voltageread
- Έλεγχο σε τι κατάσταση βρίσκεται το lock pin και άνοιγμα του κόκκινου η του πράσινου LED widget στο Blynk – checkPin
- Περιστροφή του κλειδιού για να γυρίσει και η γλώσσα όταν η πόρτα είναι ξεκλείδωτη – tempUnlocking

- Περιστροφή του κλειδιού για να γυρίσει και η γλώσσα όταν η πόρτα είναι κλειδωμένη permUnlocking
- Ξεκλείδωμα της πόρτας - unlocking
- Κλείδωμα της πόρτας - locking
- Το κουμπί widget στο Blynk για κλείδωμα ξεκλείδωμα της πόρτας - BLYNK_WRITE(V3)
- Το Κουμπί widget στο Blynk για να ξεκλειδώνει η πόρτα μέχρι και τη γλώσσα - BLYNK_WRITE(V6)
- Το φυσικό κουμπί για να κλειδώνει/ξεκλειδώνει η πόρτα – physicalbutton
- Ειδοποιήσεις PUSH στην συσκευή μας(Έξυπνο κινητό τηλέφωνο ή Tablet)– pushblynk
- Ειδοποιήσεις PUSH στο κινητό για χαμηλά επίπεδα στην μπαταρία - powerwarning

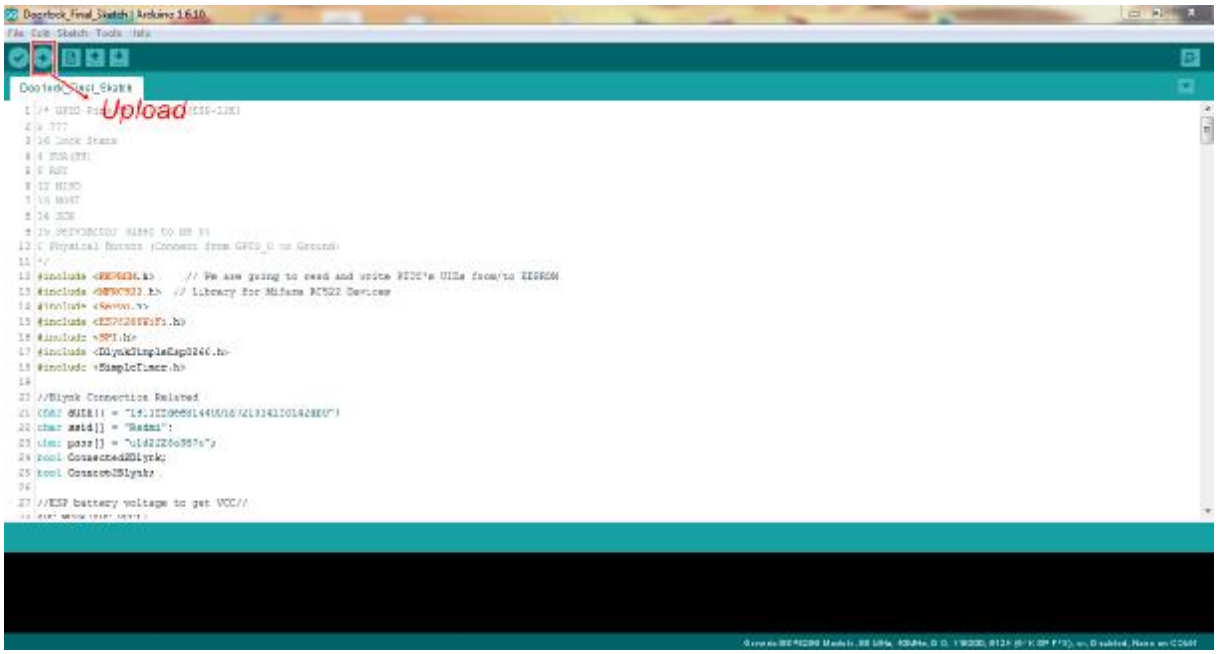
Για να ανεβάσουμε τον κώδικα στο ESP8266 αφού συνδεθεί στον υπολογιστή με τον τρόπο που δείξαμε προηγουμένως ακολουθούμε τα παρακάτω βήματα:

Ανοίγουμε το Arduino IDE και στην καρτέλα Tools [Εικόνα 3.3.1] επιλέγουμε τις παρακάτω ρυθμίσεις.



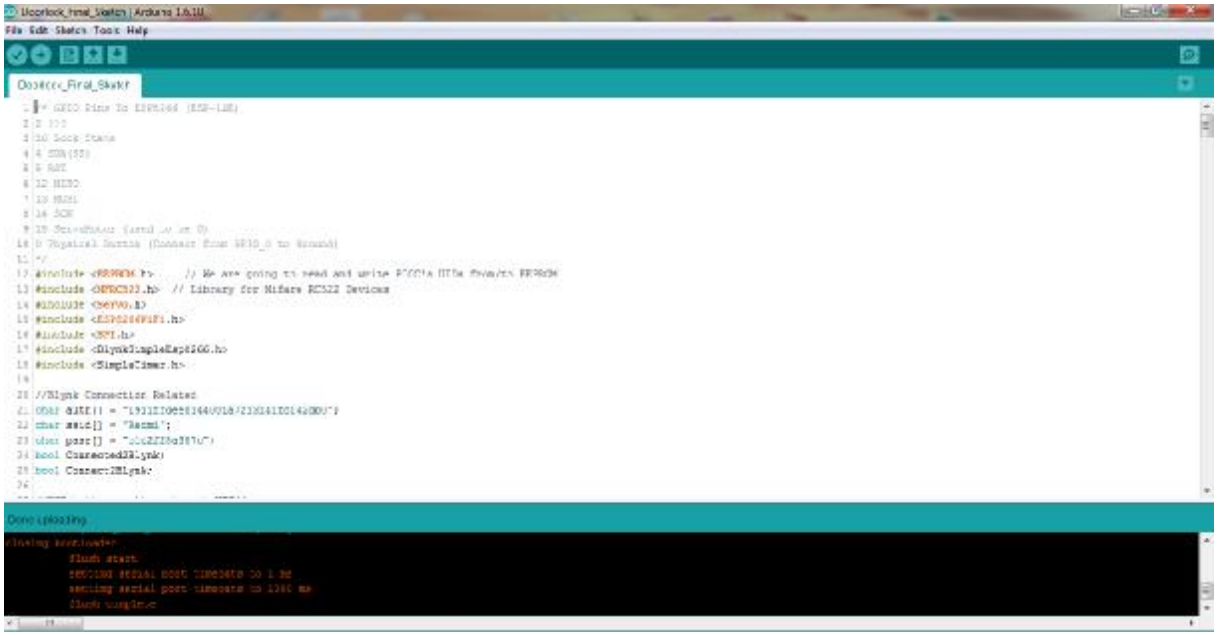
Εικόνα 3.3.1 *: Arduino IDE καρτέλα "Tools"*

Στην συνέχεια επιλέγουμε Upload όπως φαίνεται στην παρακάτω εικόνα 3.3.2



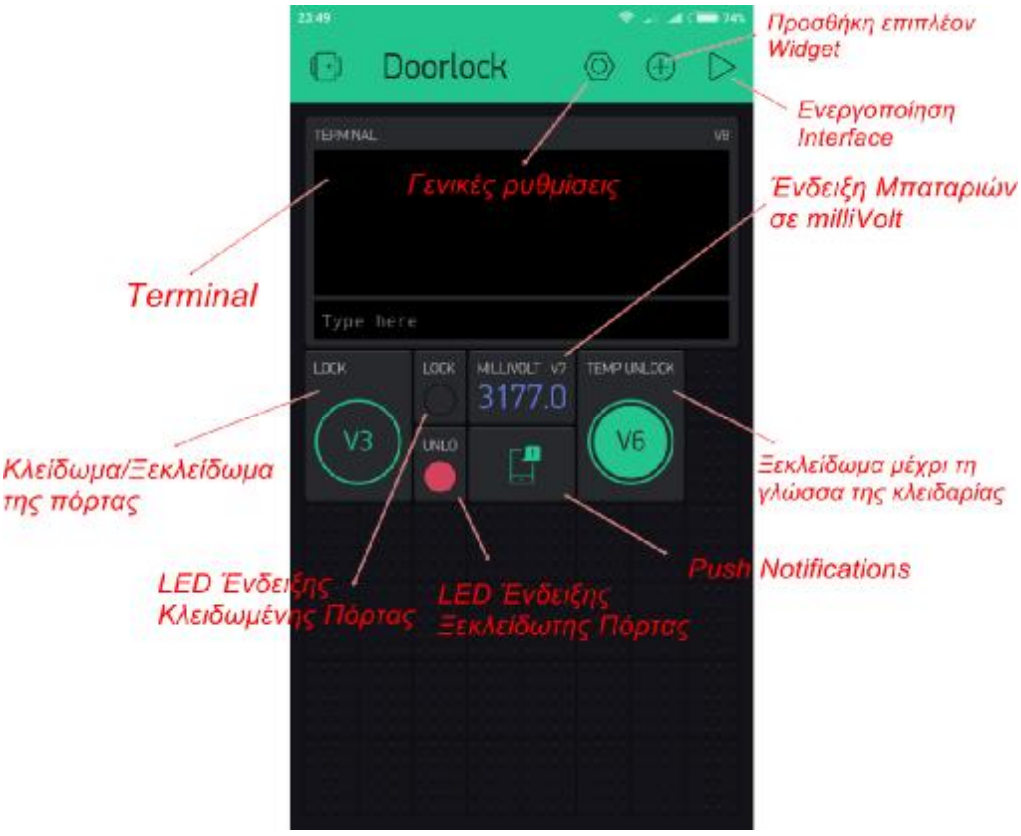
Εικόνα 3.3.2 *: Arduino IDE "Upload"*

Όταν η αποστολή των δεδομένων ολοκληρωθεί θα δούμε το μήνυμα “Done Uploading” [Εικόνα 3.3.3].



Εικόνα 3.3.3 *: Arduino IDE "Done Uploading"*

Το Menu που έχει στηθεί στην εφαρμογή Android του Blynk φαίνεται και επεξηγείται στην εικόνα 3.3.4



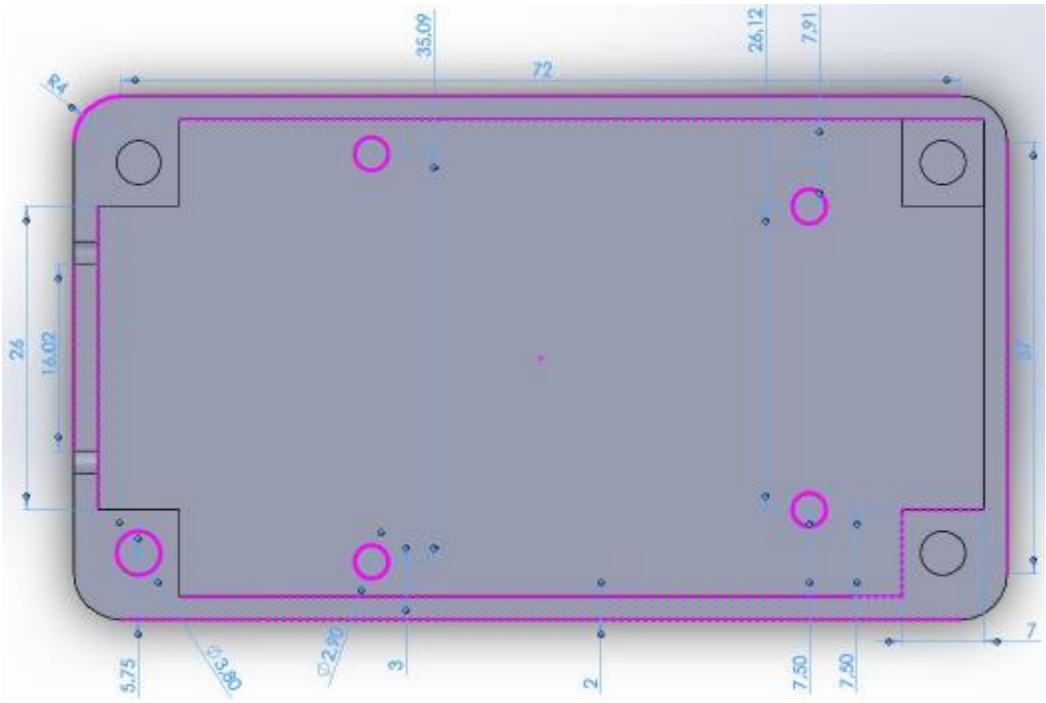
Εικόνα 3.3.4 *: Blynk App menu*

3.4 Σχέδιο Κατασκευής

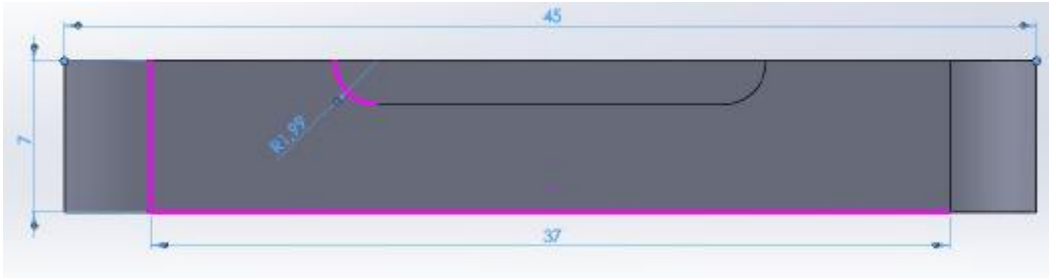
Για την σχεδίαση της κατασκευής και των επιμέρου εξαρτημάτων χρησιμοποιήθηκε το σχεδιαστικό πρόγραμμα Solidworks.

Βάση RFID Reader MFRC522

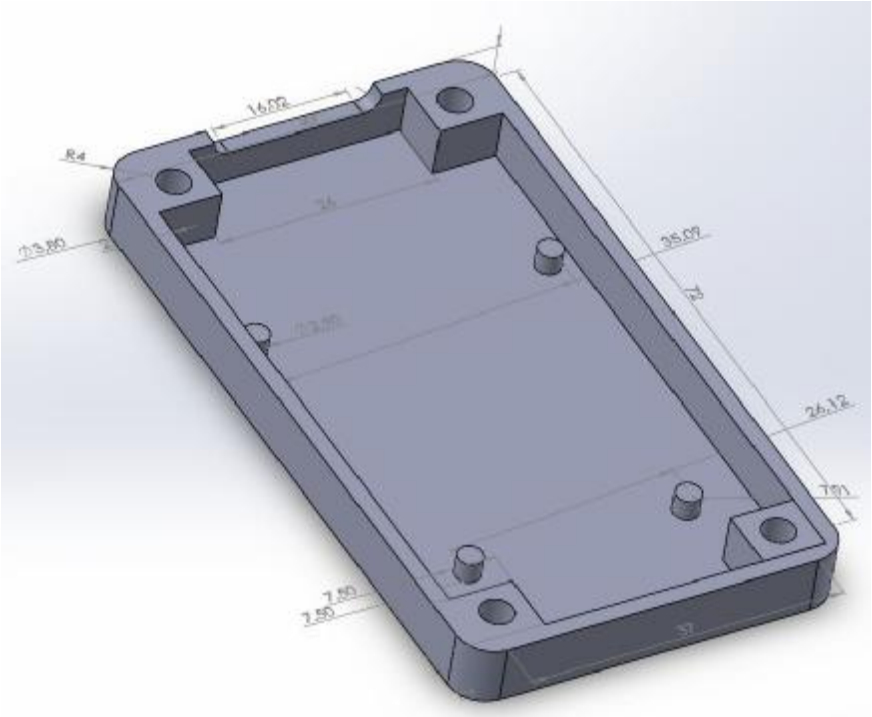
Η συγκεκριμένη βάση έχεις τις απαραίτητες διαστάσεις προκειμένου να φιλοξενήσει το εξάρτημα MFRC-522 Rfid Reader. Είναι κατάλληλο για την εξωτερική τοποθέτηση καθώς έχει μικρές διαστάσεις και προφυλλάσει το εξάρτημα από καιρικά φαινόμενα σε συνδυασμό με το καπάκι. Η βάση χρησιμοποιεί 4 βίδες στις τέσσερις γωνίες για να κλείσει μαζί με το καπάκι.



Εικόνα 3.4.1 *: Rfid Reader MFRC522 βάση επάνω όψη*

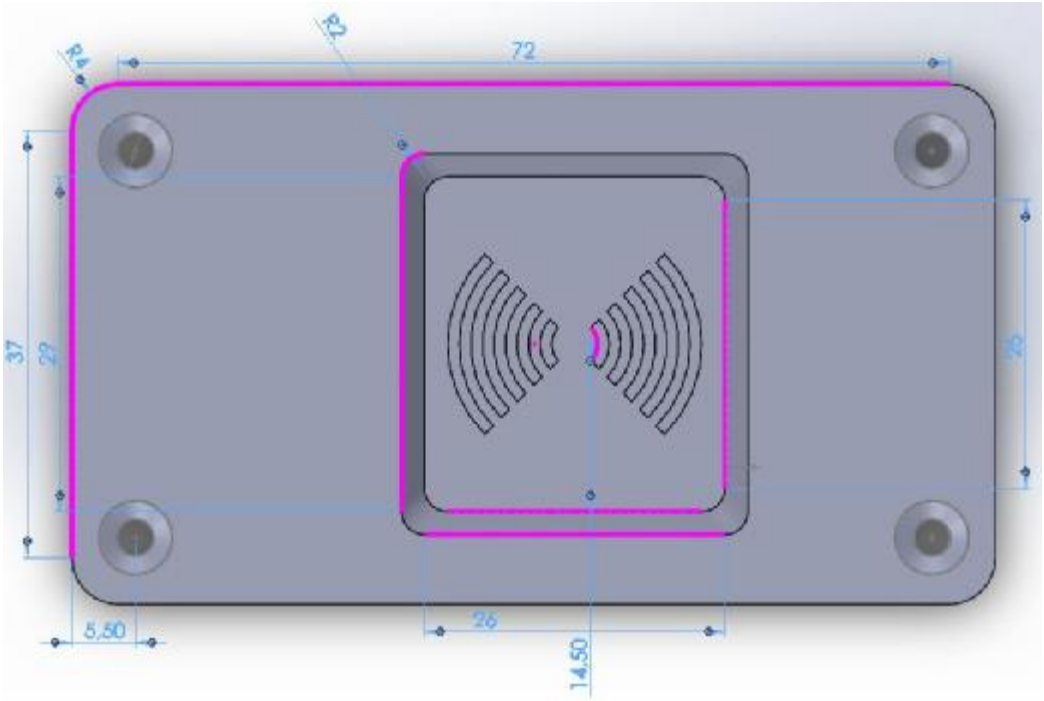


Εικόνα 3.4.2 *: Rfid Reader MFRC522 βάση μπροστινή όψη*

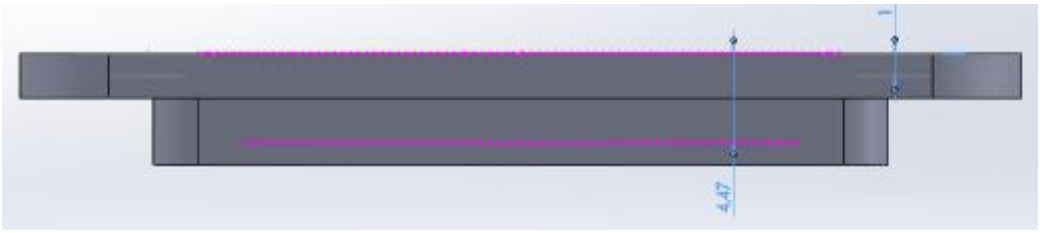


Εικόνα 3.4.3 *: Rfid Reader MFRC522 βάση τρισδιάστατη όψη*

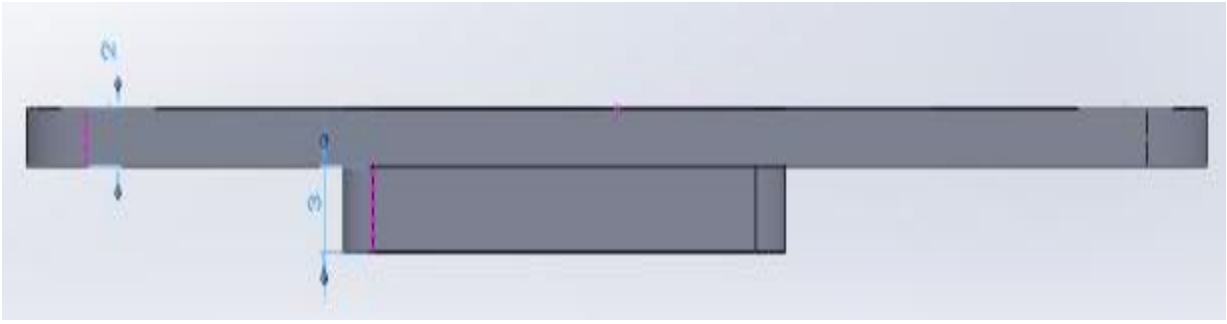
Καπάκι RFID Reader MFRC522



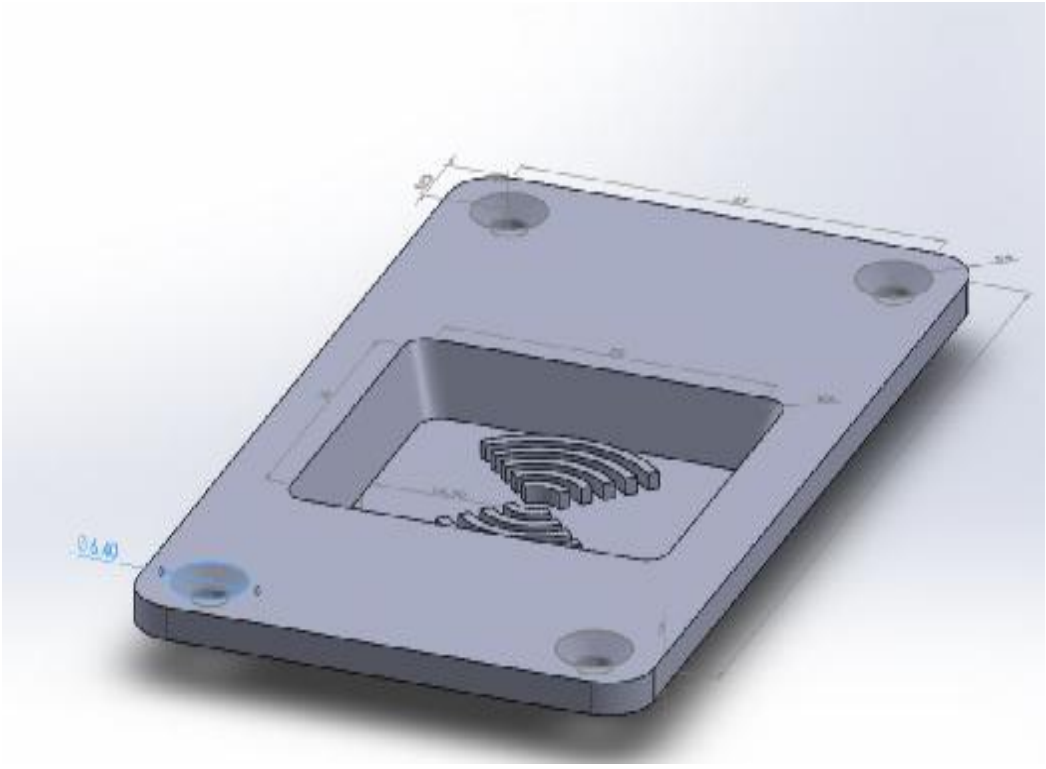
Εικόνα 3.4.4 *: Καπάκι RFID Reader MFRC522 Επάνω όψη*



Εικόνα 3.4.5 *: Καπάκι RFID Reader MFRC522 Επάνω όψη*



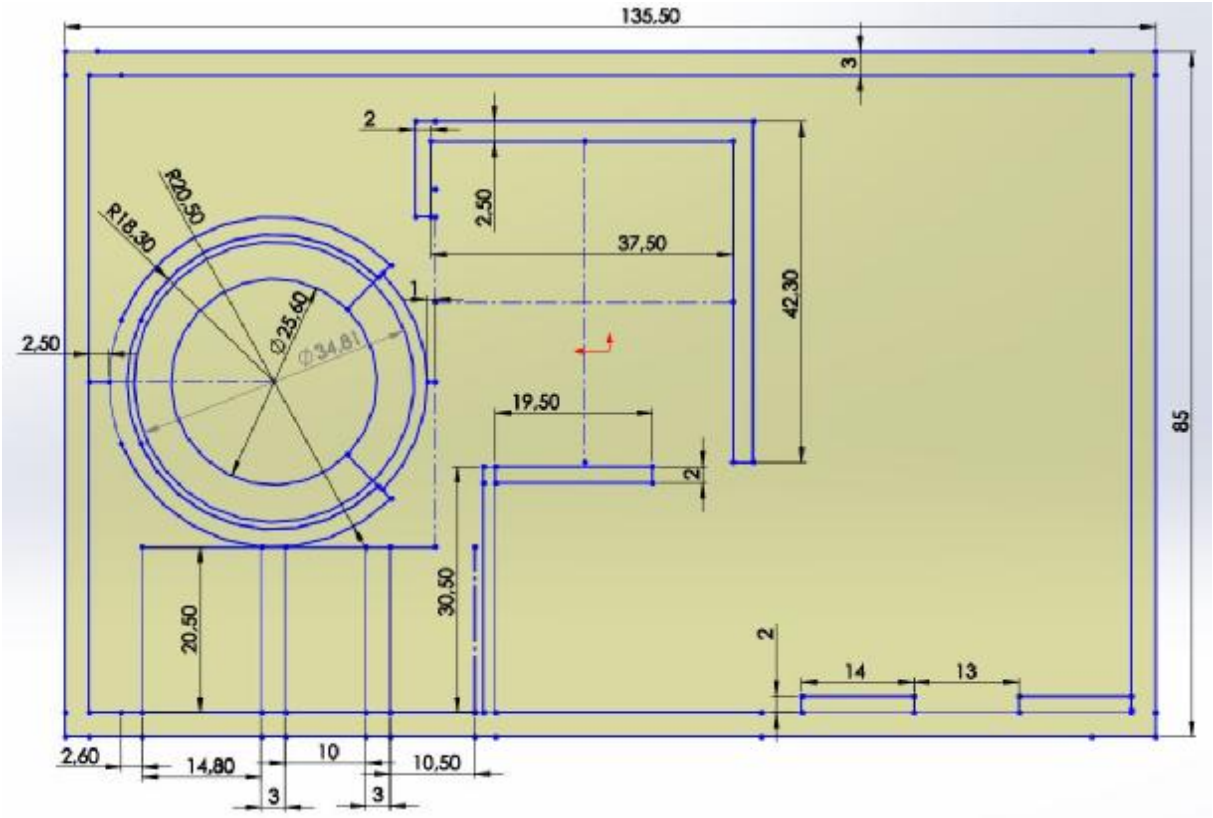
Εικόνα 3.4.6 *: Καπάκι RFID Reader MFRC522 πλάγια όψη*



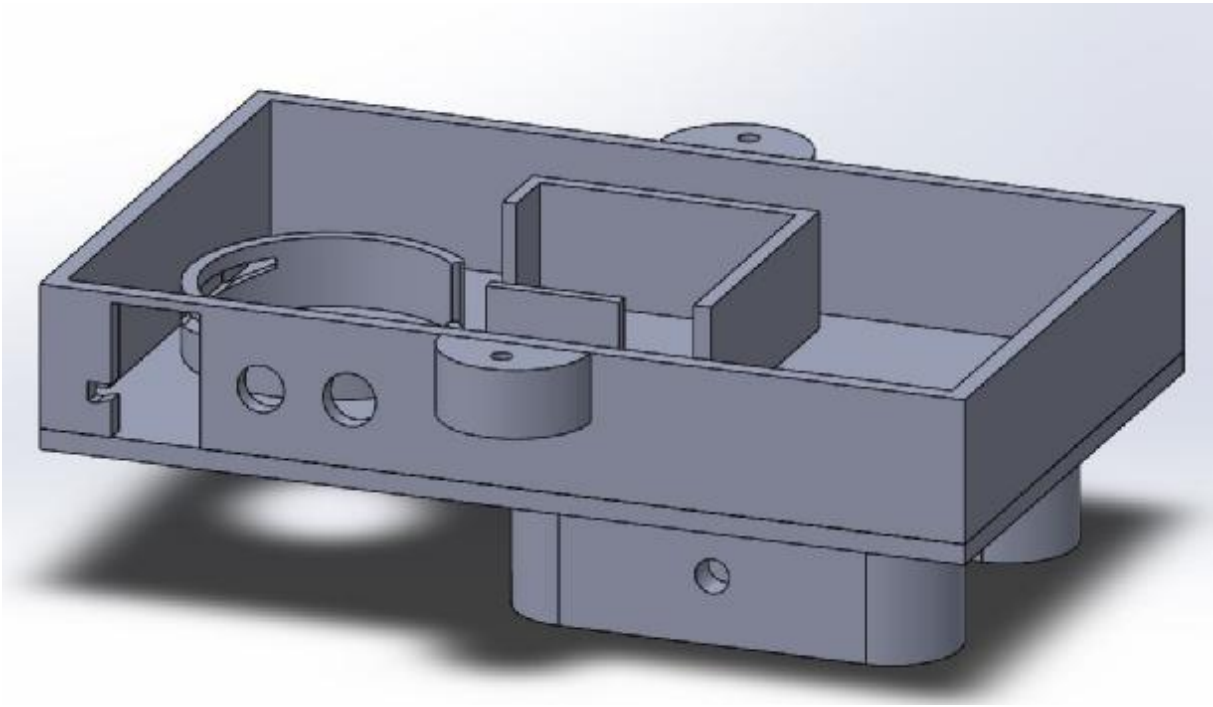
Εικόνα 3.4.7 *: Καπάκι RFID Reader MFRC522 τρισδιάστατη όψη*

Βάση θήκης Μηχανισμού

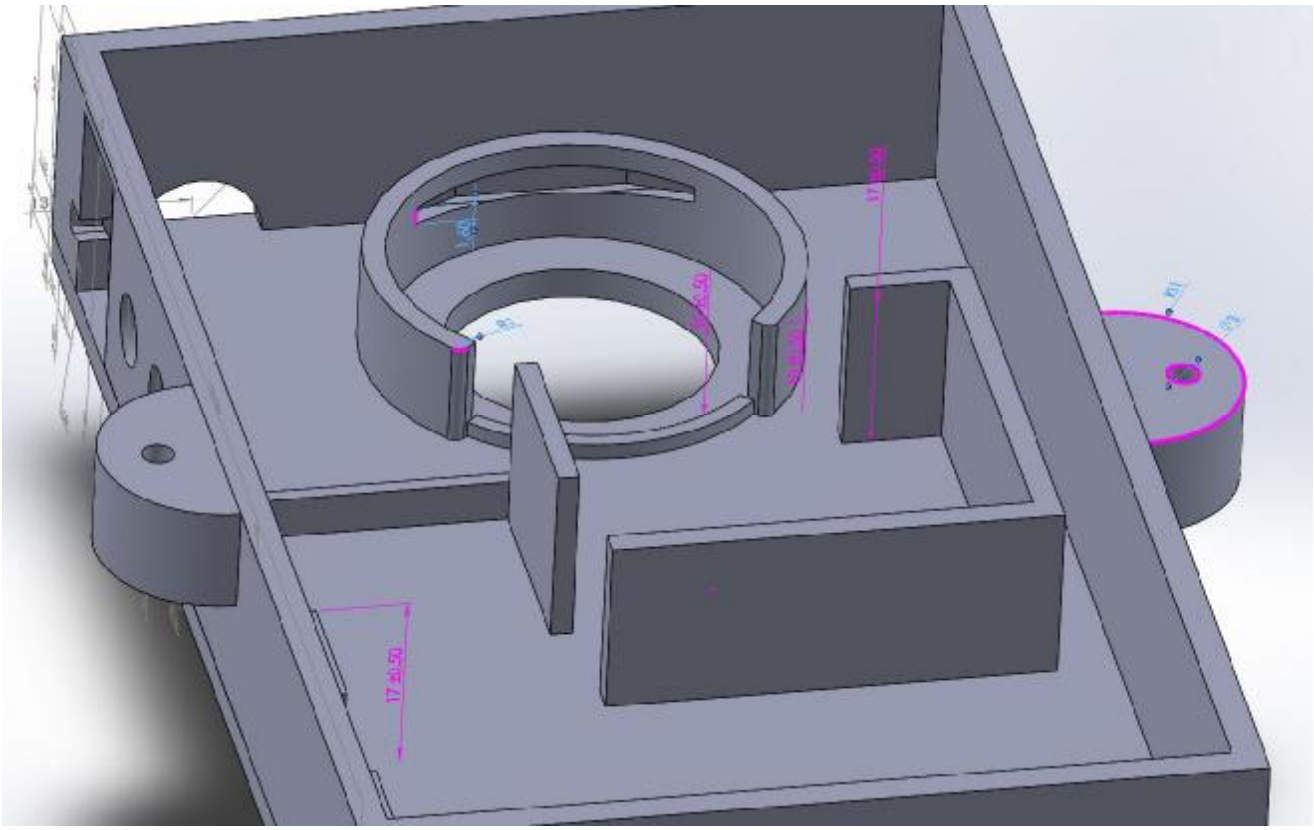
Η θήκη του μηχανισμού είναι ο πυρήνας της κατασκευής και αποτελείται από διάφορα εξαρτήματα τα οποία αναλύονται στην συνέχεια.



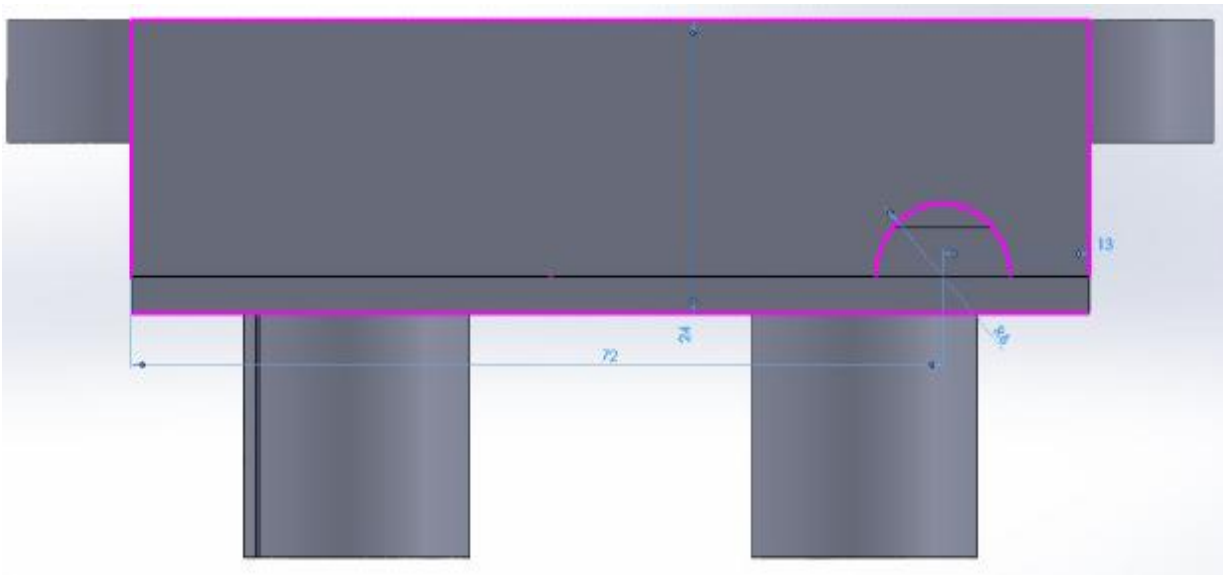
Εικόνα 3.4.8 *: Βάση θήκης μηχανισμού δισδιάστατη όψη*



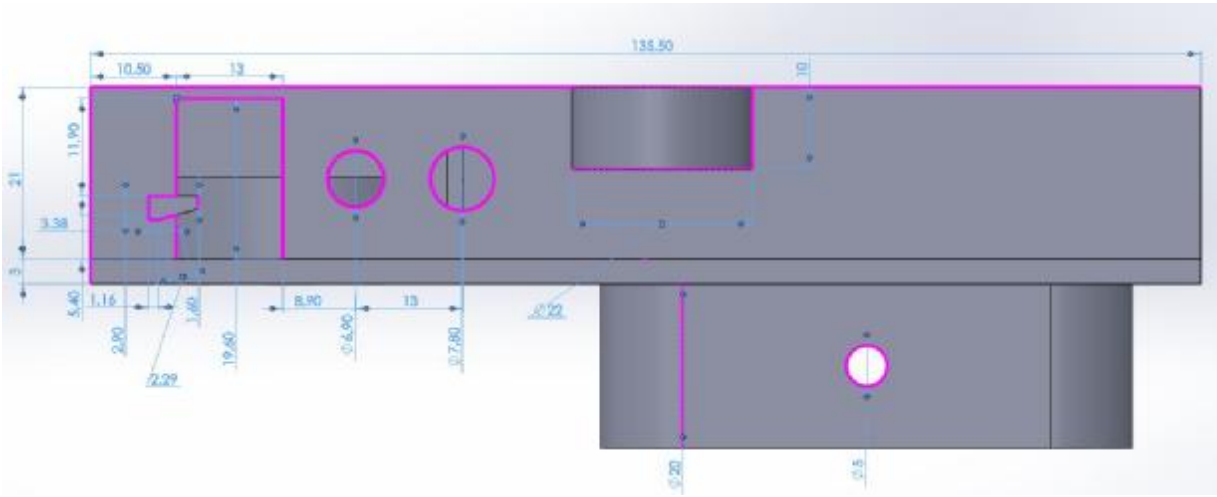
Εικόνα 3.4.9 *: Βάση θήκης μηχανισμού τρισδιάστατη όψη*



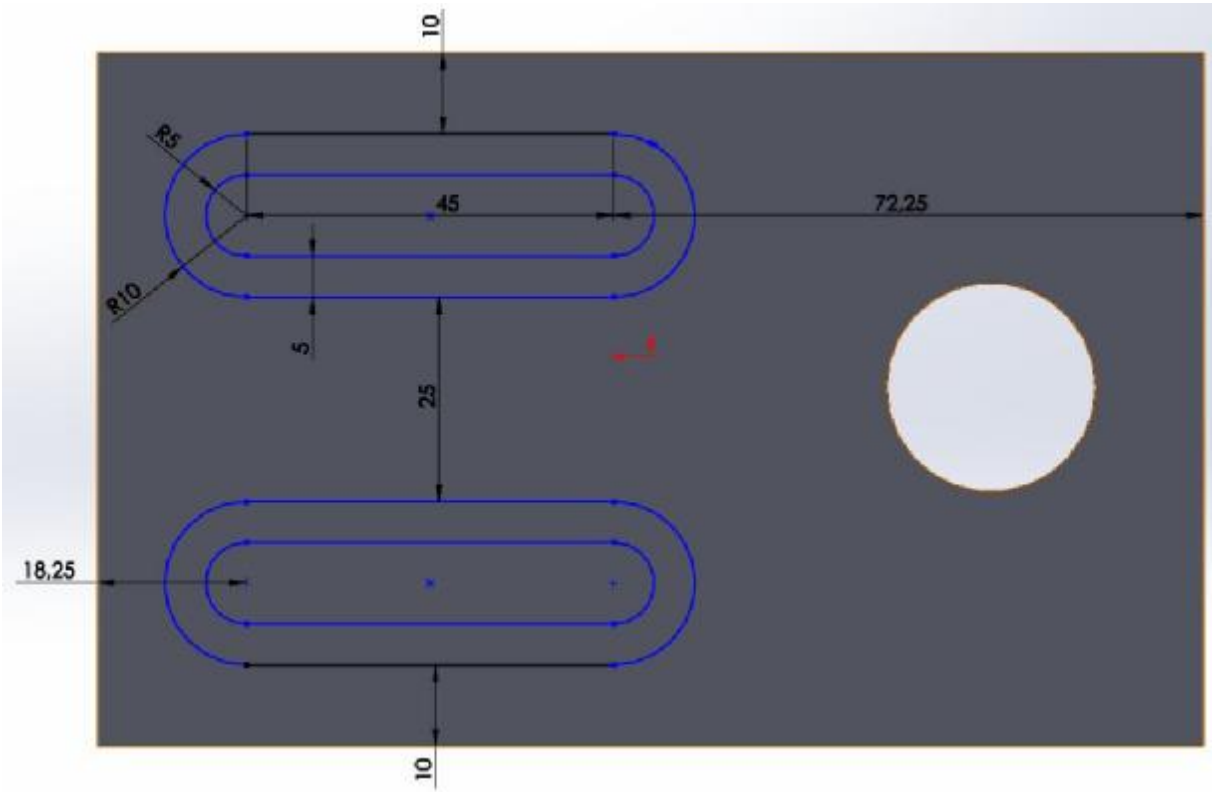
Εικόνα 3.4.10 *: Βάση θήκης μηχανισμού - επάνω τρισδιάστατη όψη*



Εικόνα 3.4.11 *: Βάση θήκης μηχανισμού μπροστινή όψη*



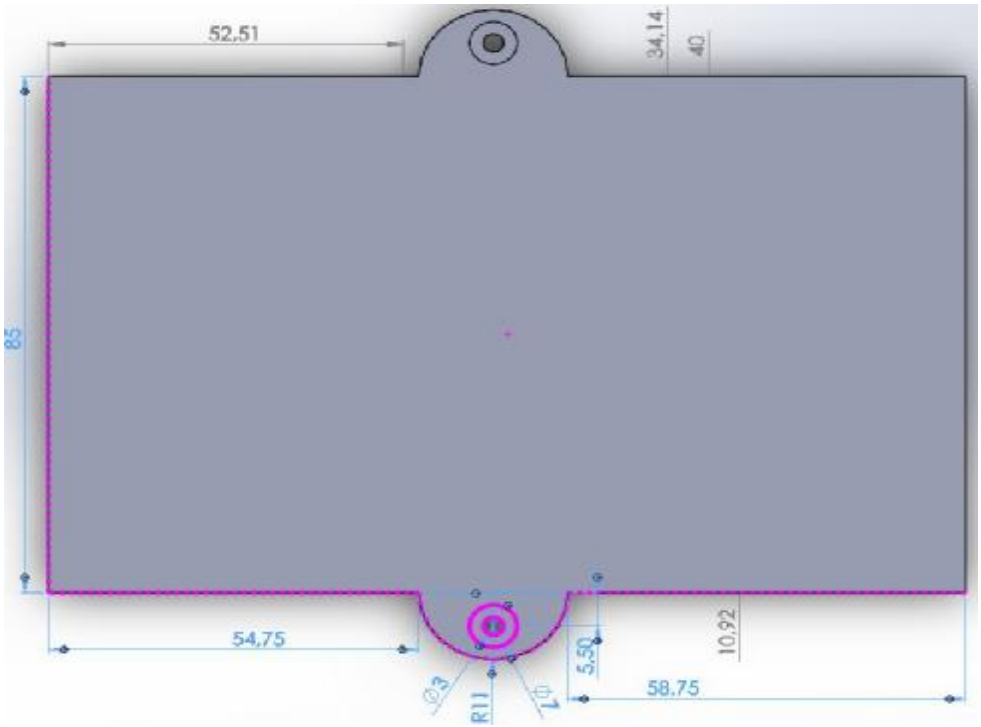
Εικόνα 3.4.12 *: Βάση θήκης μηχανισμού πλαινή όψη*



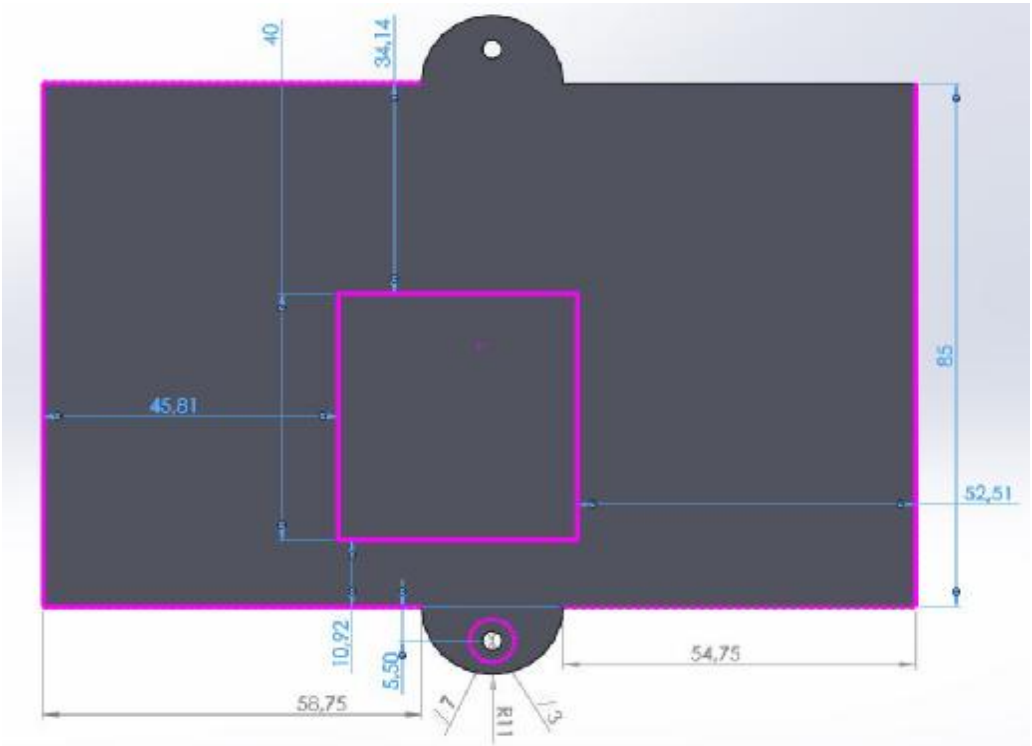
Εικόνα 3.4.13 *: Βάση θήκης μηχανισμού κάτω όψη*

Καπάκι θήκης μηχανισμού

Το καπάκι κλείνει την βάση της θήκης του μηχανισμού και περιλαμβάνει μια επεκτάση στο σημείο που βρίσκεται ο σερβοκινητήρας κατά 1mm πάχος επιπλέον ώστε να πιέζει το σερβοκινητήρα και αυτός με τη σειρά του το γρανάζι των 36 δοντιών για να μην μετακινείται πέρα από την προβλεπόμενη θέση του.



Εικόνα 3.4.14 *: Καπάκι θήκης μηχανισμού επάνω όψη*

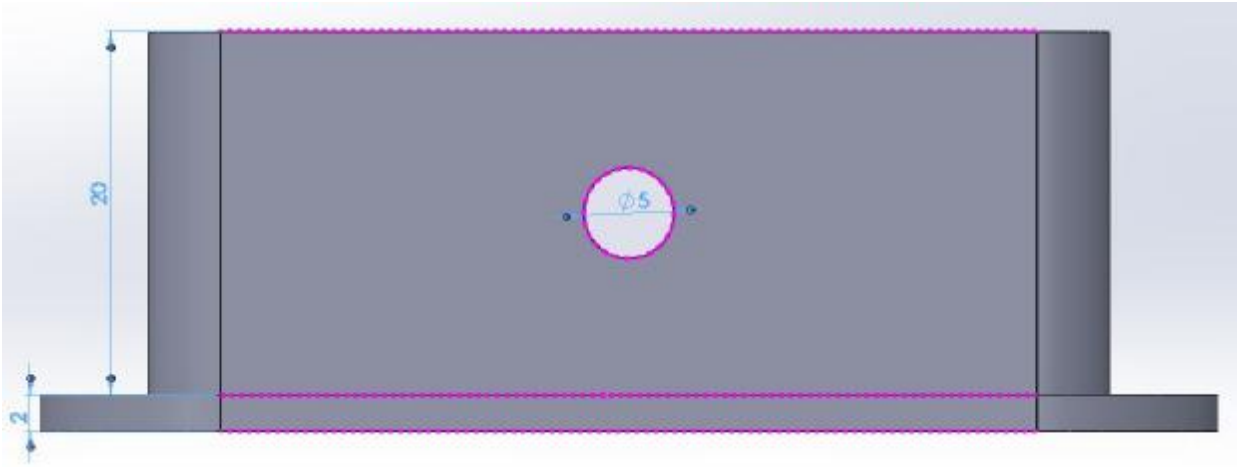


Εικόνα 3.4.15 *: Καπάκι θήκης μηχανισμού κάτω όψη*

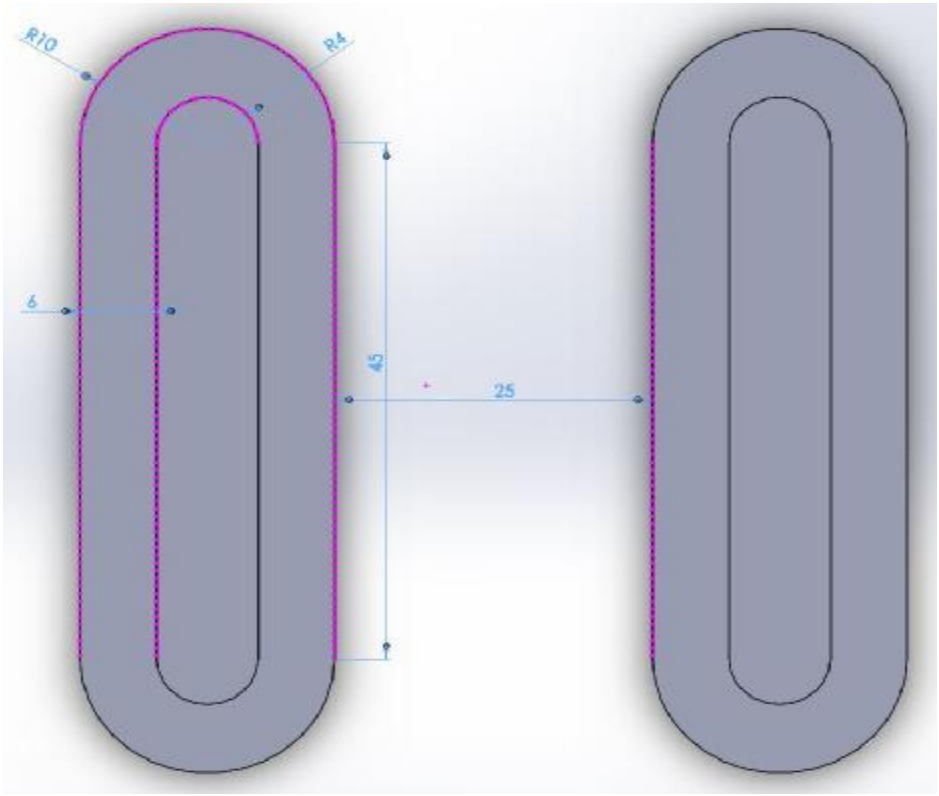
Εικόνα 3.4.16 *: Καπάκι θήκης μηχανισμού πλάγια όψη*

Πόδια βάσης θήκης μηχανισμού

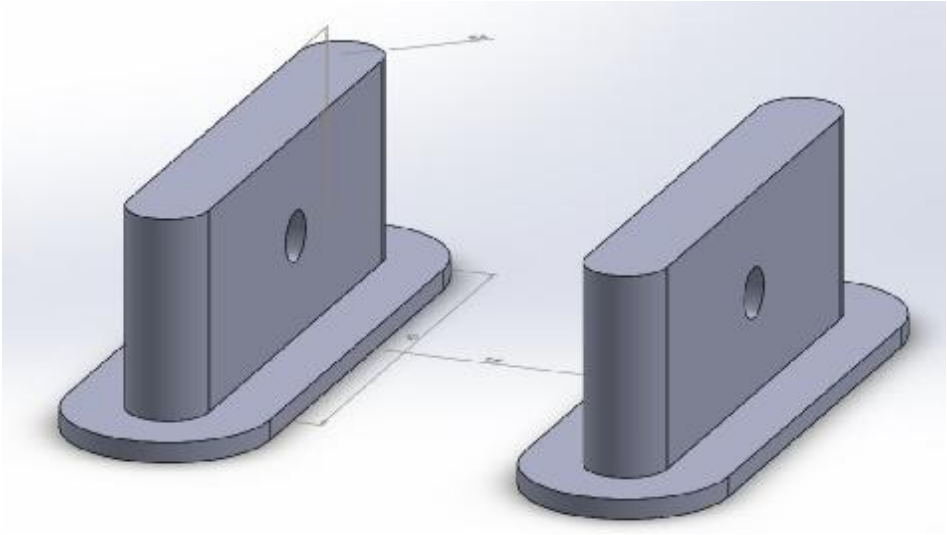
Τα πόδια είναι αποσπώμενα και τοποθετούνται μέσα στις θήκες της βάσης και μένουν στις προβλεπόμενες θέσεις τους με τη βοήθεια μιας ράβδου.



Εικόνα 3.4.17 *: Πόδια βάσης θήκης μηχανισμού πλάγια όψη*

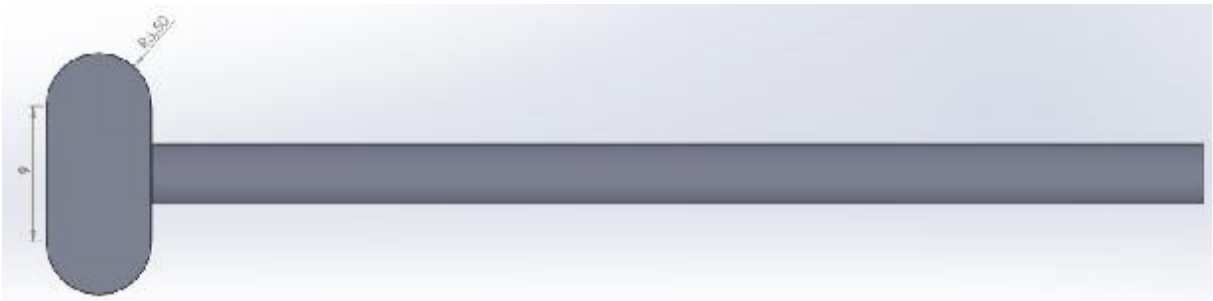


Εικόνα 3.4.18 *: Πόδια βάσης θήκης μηχανισμού επάνω όψη*



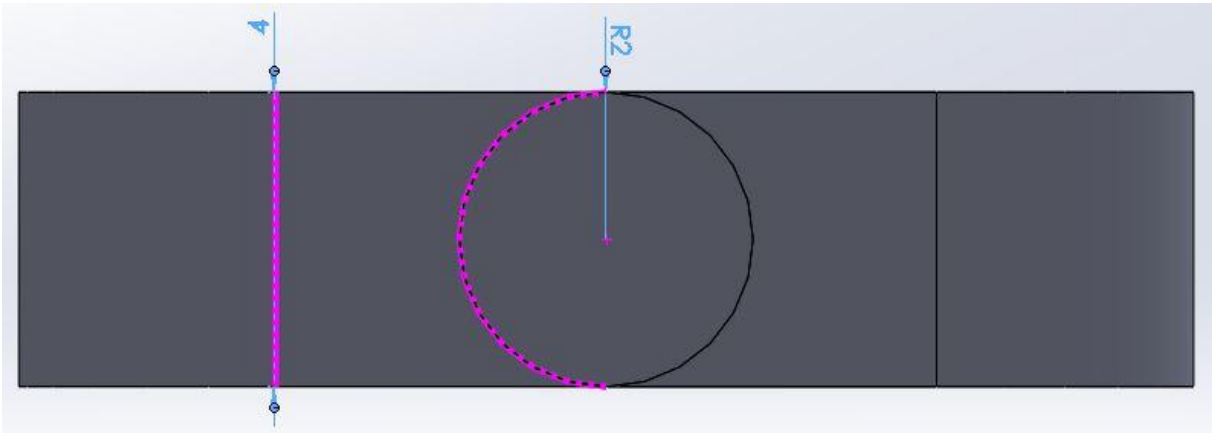
Εικόνα 3.4.19 *: Πόδια βάσης θήκης μηχανισμού τρισδιάστατη όψη*

Ράβδος βάσης ποδιών θήκης μηχανισμού



Εικόνα 3.4.20 *: Ράβδος βάσης ποδιών θήκης μηχανισμού επάνω όψη*

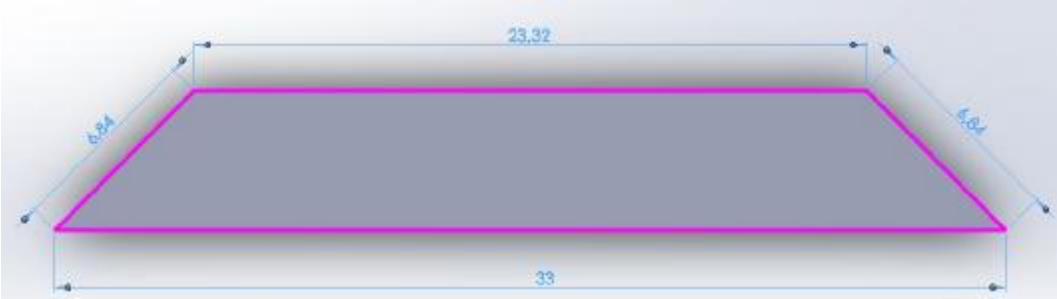
Εικόνα 3.4.21 *: Ράβδος βάσης ποδιών θήκης μηχανισμού πλάγια όψη*



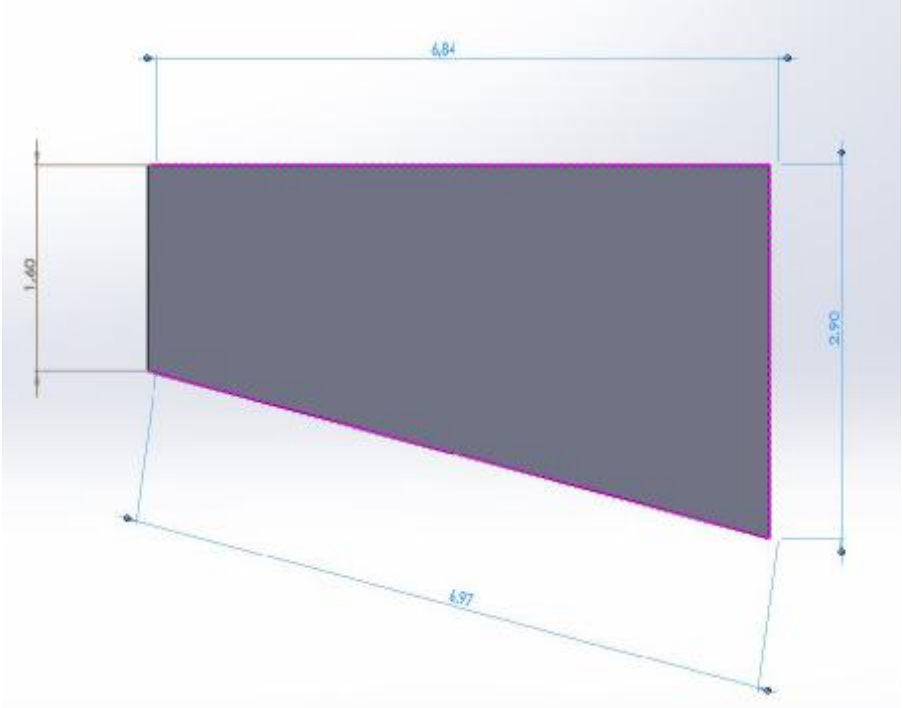
Εικόνα 3.4.22 *: Ράβδος βάσης ποδιών θήκης μηχανισμού μπροστινή όψη*

Ράβδος υποστήριξης κωνικού γραναζιού 36 δοντιών

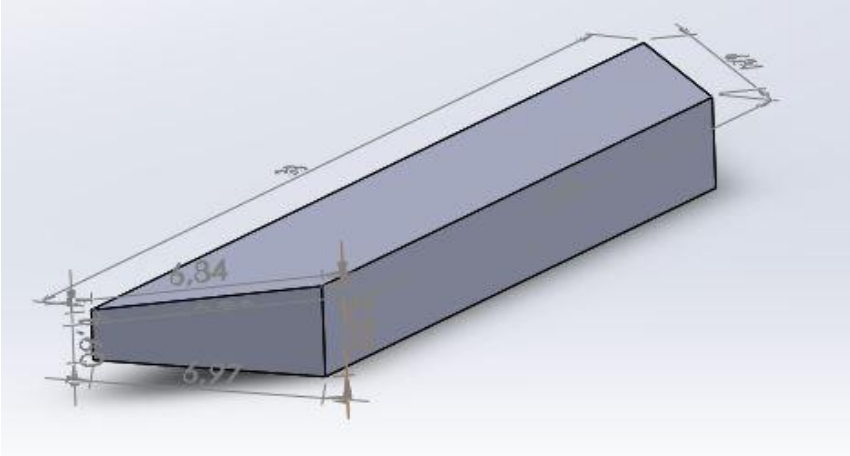
Η ράβδος υποστήριξης κωνικού γραναζιού είναι τοποθετημένη αντιδιαμετρικά του κωνικού γραναζιού και σκοπός της είναι να κράτα το γρανάζι στην θέση που πρέπει όταν περιστρέφεται.



Εικόνα 3.4.23 *: Ράβδος υποστήριξης κωνικού γραναζιού 36 δοντιών επάνω όψη*



Εικόνα 3.4.24 *: Ράβδος υποστήριξης κωνικού γραναζιού 36 δοντιών πλάγια όψη*



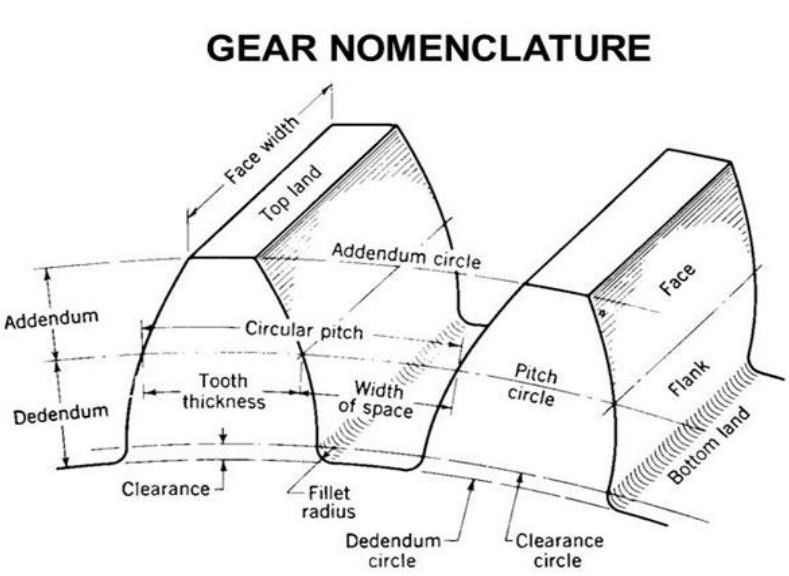
Εικόνα 3.4.25 *: Ράβδος υποστήριξης κωνικού γραναζιού 36 δοντιών τρισδιάστατη όψη*

Κωνικό σετ γραναζιών 36 δοντιών & 11 δοντιών μετάδοσης κίνησης & περιστροφής κλειδιού

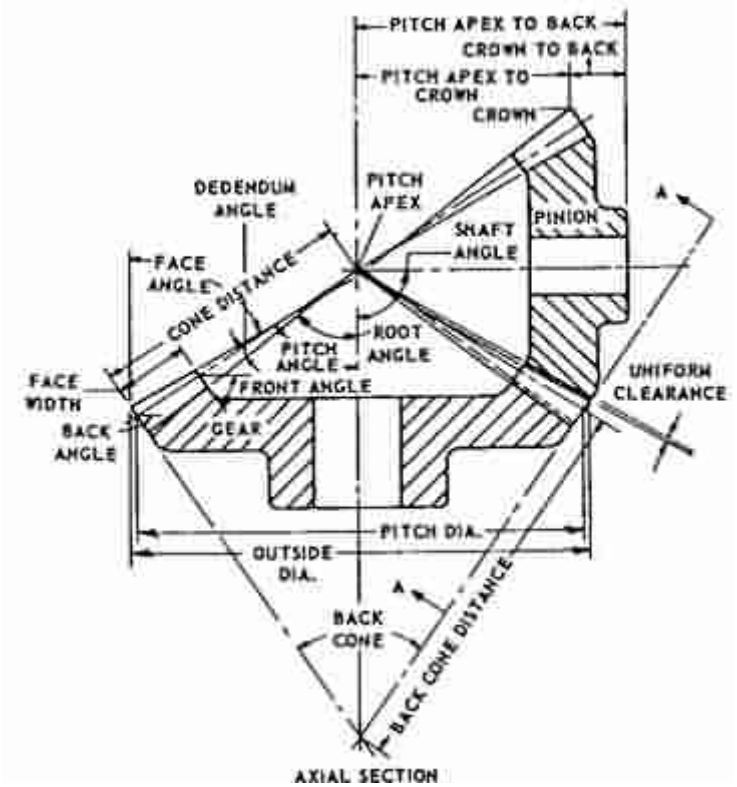
Το σετ των δύο γραναζιών αυτών έχει ως σκοπό την μετάδοση της κίνησης από τον σερβοκινητήρα προς το κωνικό γρανάζι 11 δοντιών, το οποίο στη συνέχεια περιστρέφει το κωνικό γρανάζι των 36 δοντιών όπου με τη σειρά του περιστρέφει το κλειδί. Το σετ των κωνικών γραναζιών σχεδιάστηκε με την βοήθεια του προγράμματος Geartrax. Οι λεπτομέρειες είναι οι ακόλουθες.

Ιδιότητες κωνικών γραναζιών μετάδοσης κίνησης		
Τύπος κωνικού γραναζιού [Bevel Type]		Ευθύγραμμο
Διαμετρικό βήμα [Diametral Pitch]		25.4
Μοντούλ [Module]		1.00
Κυκλικό βήμα [Circular Pitch]		**3.142 mm**
Γωνία επαφής μετάδοσης κίνησης ή γωνία εξειλγμένης [Pressure Angle]		20°
Συντελεστής κεφαλής [Coefficient Addendum]		1.00
Συντελεστής διάκενου [Coefficient Clearance]		0.2
Συντελεστής Fillet [Coefficient Fillet]		0.25
Σχέση Μετάδοσης [Gear Ratio]		1:3.273
Στοιχεία Κώνου [Bevel Data]		
	Pinion	**Gear**
Αριθμός δοντιών [Number of teeth]	11	36
Θεωρητική διάμετρος γραναζιού [Pitch diameter]	11.0mm	36.0mm
Εξωτερική διάμετρος [Outside Diameter]	13.735mm	36.333mm
Ύψος κεφαλής[Addendum]	1.430mm	0.570mm
Ύψος ποδός [Dedendum]	0.770	1.630
Βάθος λειτουργίας δοντιού[Whole Depth]	2.2mm	2.2mm
Πάχος δοντιού [Tooth Thickness]	1.884mm	1.258mm
Ακτίνα Fillet [Fillet Radius]	0.25mm	0.25mm
Γωνίες [Angles]		
Γωνία μετακύ κεφαλής & ποδός [Pitch Angle]	16.9908°	73.0092°
Γωνία προσώπου[Face Angle]	21.9405°	75.3519°
Γωνία ποδός [Root Angle]	14.6481°	68.0595°
Γενική γεωμετρία [Blank Geometry]		
Πλάτος προσώπου[Face Width]	5mm	10mm
Απόσταση βάσης[Mounting Distance]	19	7.455
Απόσταση κορώνας έως πλάτης[Crown to Back]	1.418mm	2.5mm
Απόσταση προσώπου έως πλάτης[Face to Back]	6.073mm	5.031mm
Απόσταση κορώνας έως την κορυφή [Crown to Apex]	17.582mm	4.955mm

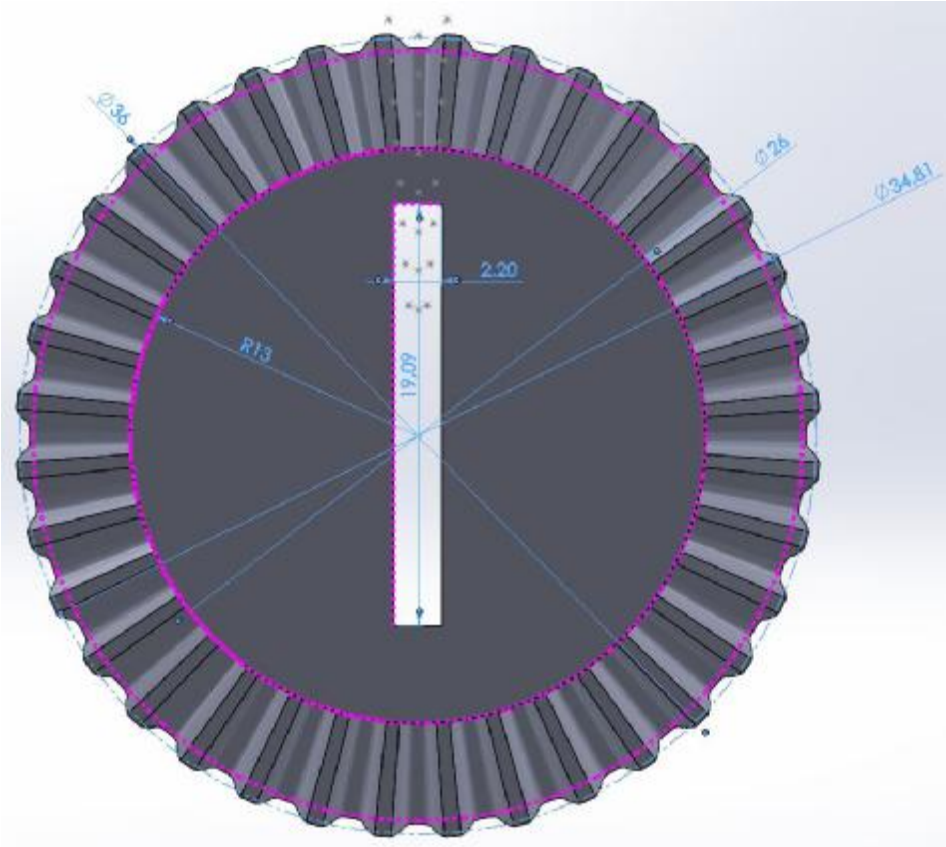
Τα στοιχεία του πίνακα απεικονίζονται στις εικόνες 3.4.26 & 3.4.27



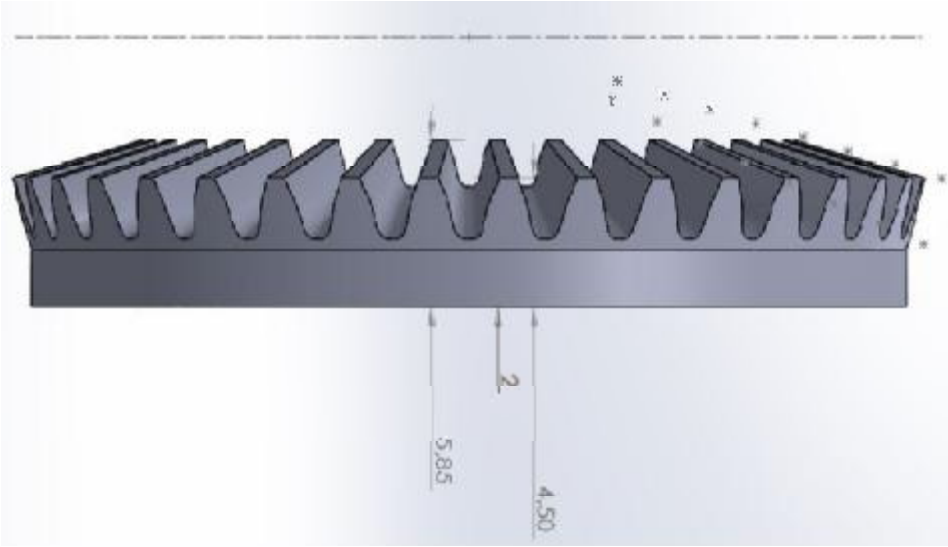
Εικόνα 3.4.26 *: Ιδιότητες και στοιχεία γραναζιού*



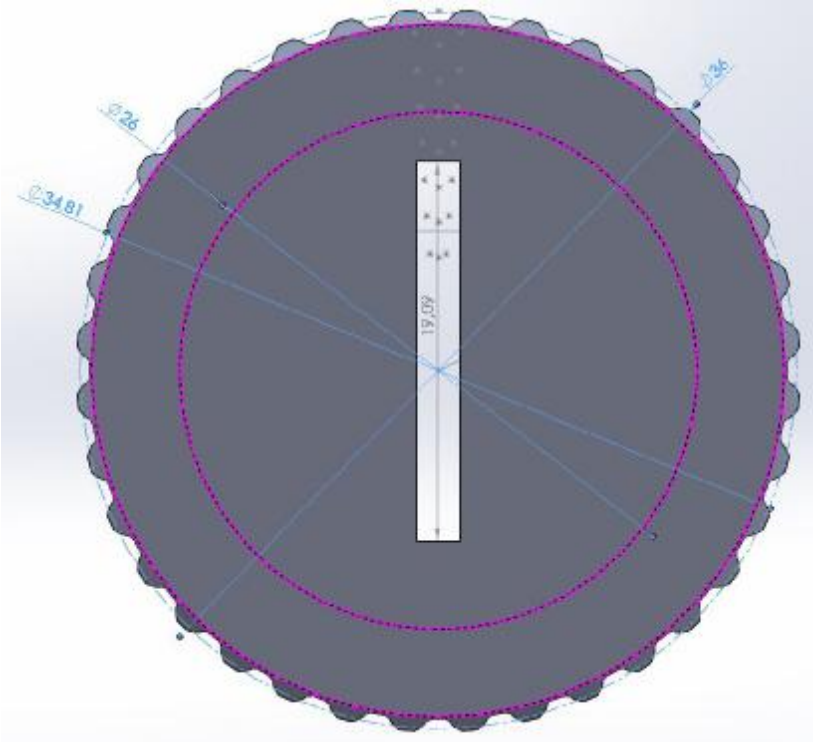
Εικόνα 3.4.27 *: Ιδιότητες κωνικών γραναζιών*



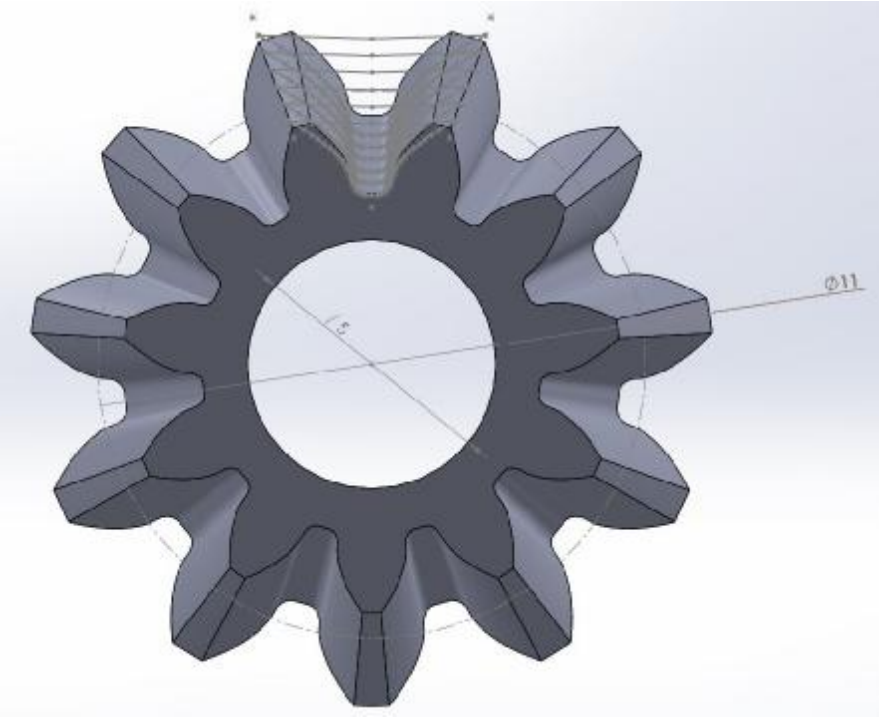
Εικόνα 3.4.28 *: Κωνικό γρανάζι 36 δοντιών επάνω όψη*



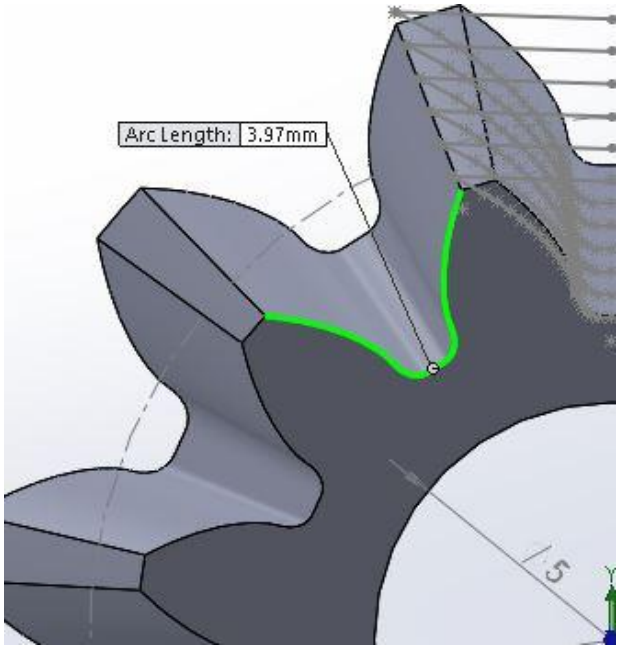
Εικόνα 3.4.29 *: Κωνικό γρανάζι 36 δοντιών πλάγια όψη*



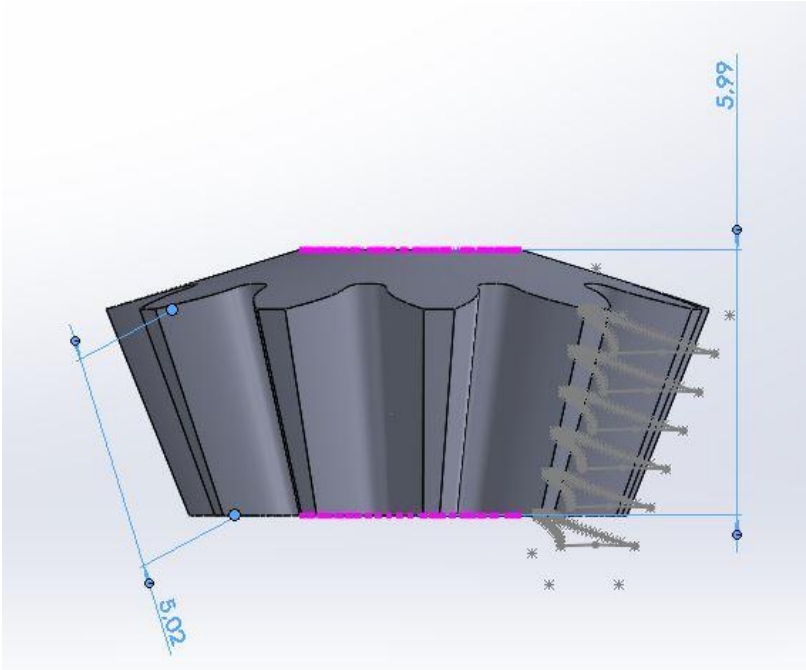
Εικόνα 3.4.30 *: Κωνικό γρανάζι 36 δοντιών κάτω όψη*



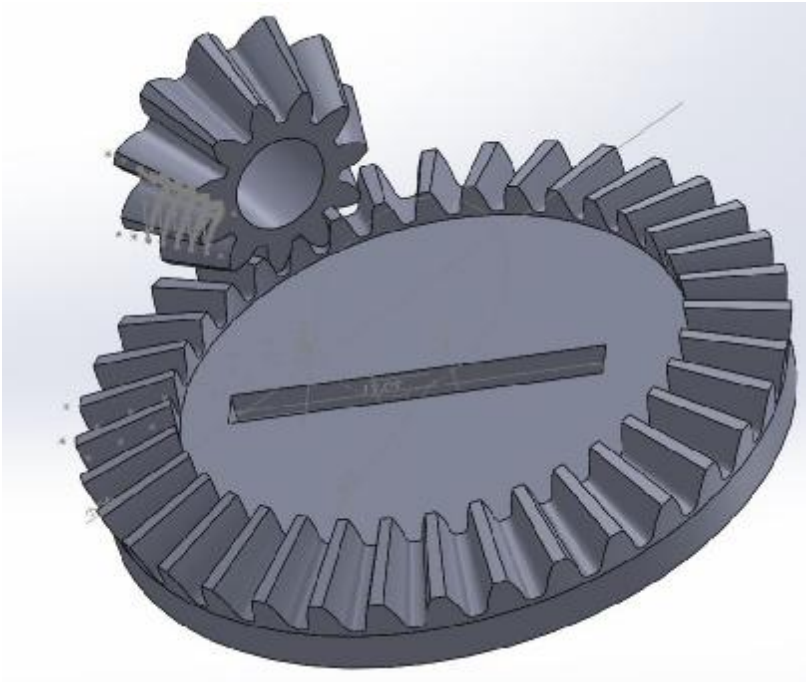
Εικόνα 3.4.31 *: Κωνικό γρανάζι 11 δοντιών επάνω όψη*



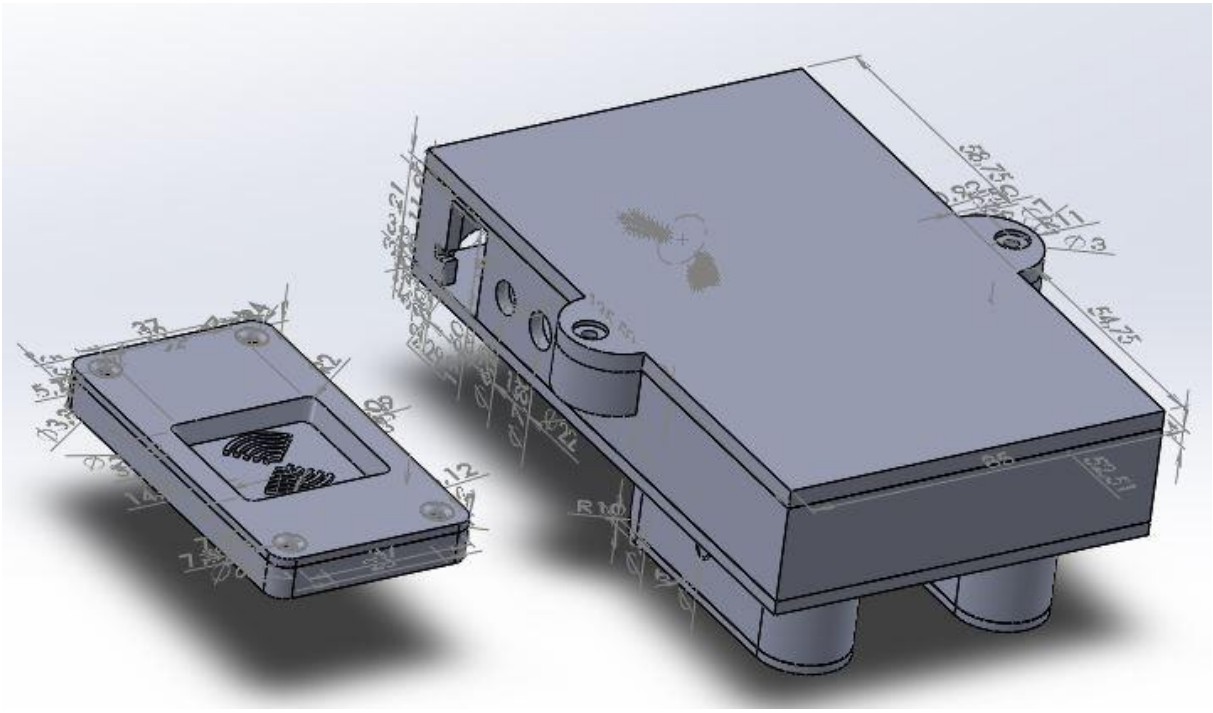
Εικόνα 3.4.32 *: Κωνικό γρανάζι 11 δοντιών διάσταση αψίδας*



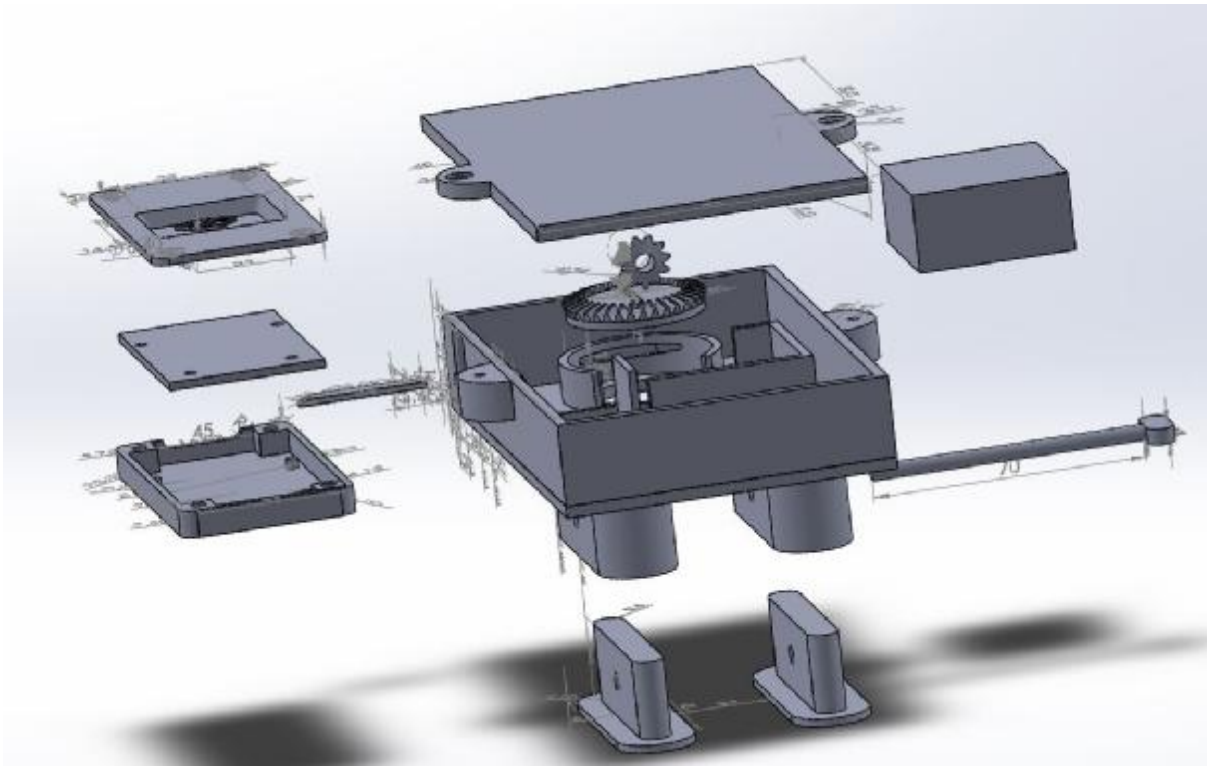
Εικόνα 3.4.33 *: Κωνικό γρανάζι 11 δοντιών πλάγια όψη*



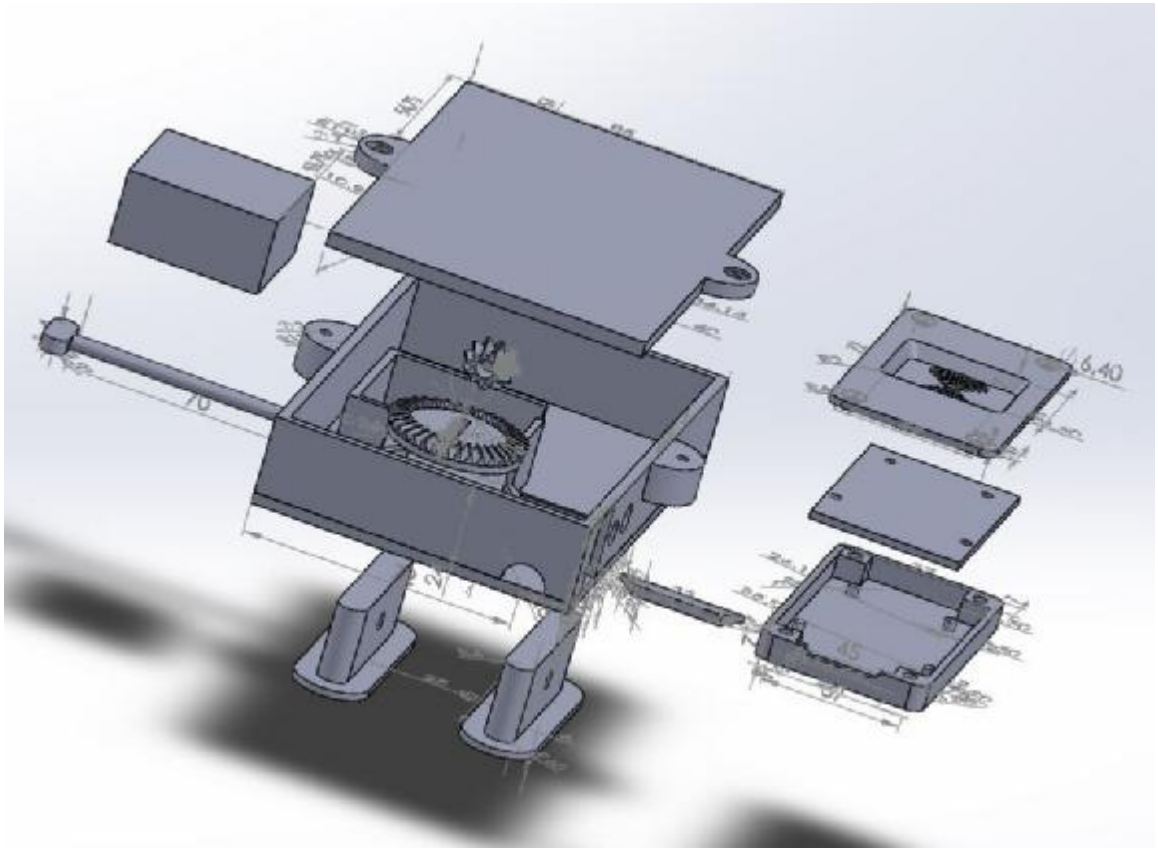
Εικόνα 3.4.34 *: Τρισδιάστατη όψη συναρμογής κωνικών γραναζιών 36 & 11 δοντιών*



Εικόνα 3.4.35 *: Συναρμογή τελικής κατασκευής τρισδιάστατη όψη*



Εικόνα 3.4.36 *: Συναρμογή τελικής κατασκευής τρισδιάστατη όψη – exploded (1)*



***Εικόνα 3.4.37** : Συναρμογή τελικής κατασκευής τρισδιάστατη όψη – exploded (2)*

3.5 Φυσικό Μοντέλο

Η μεταφορά από το σχέδιο στο φυσικό μοντέλο έγινε με την βοήθεια της τεχνολογίας τρισδιάστατης εκτύπωσης [3D Printing].

Η τρισδιάστατη εκτύπωση (3D Printing) είναι μια μέθοδος προσθετικής κατασκευής στην οποία κατασκευάζονται αντικείμενα μέσω της διαδοχικής πρόσθεσης επάλληλων στρώσεων υλικού. Στη τρισδιάστατη εκτύπωση μπορούν να χρησιμοποιηθούν διάφοροι τύποι υλικού, κυρίως κεραμικά και πολυμερή. Σε σύγκριση με άλλες τεχνολογίες και εξοπλισμό προσθετικής κατασκευής, οι τρισδιάστατοι εκτυπωτές είναι συνήθως ταχύτεροι, φθηνότεροι και ευκολότεροι στη χρήση. Οι τρισδιάστατοι εκτυπωτές χρησιμοποιούνται και προτιμούνται κυρίως για την κατασκευή φυσικών μοντέλων και πρωτοτύπων από σχεδιαστές, μηχανικούς και ομάδες ανάπτυξης νέων προϊόντων, έχουν τη δυνατότητα να εκτυπώνουν μέρη και εξαρτήματα από διάφορα υλικά, με διαφορετικές μηχανικές και φυσικές ιδιότητες και συχνά σε μια ενιαία διαδικασία κατασκευής.

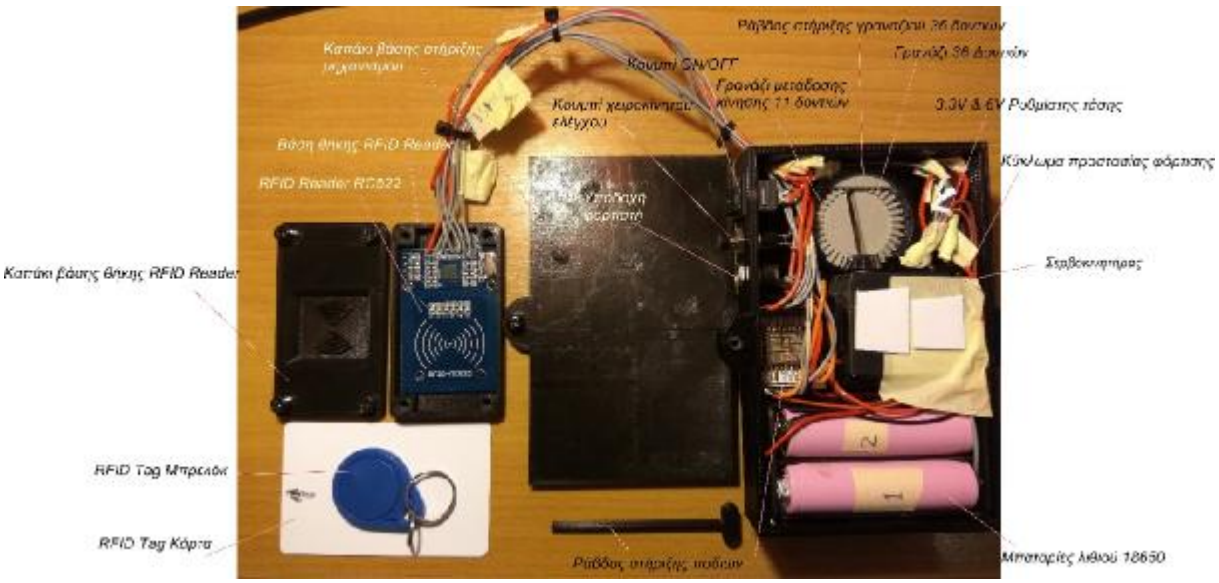
Ο τύπος της 3D Printing τεχνολογίας που χρησιμοποιήθηκε ήταν fused depositional modeling (FDM) και λειτουργεί ως εξής. Ένα θερμοπλαστικό υλικό όπως Polylactic acid (PLA) η Acrylonitrile butadiene styrene (ABS) βγαίνει από έναν εξωθητή (Extruder) που στην άκρη του έχει ένα καυτό σημείο προκειμένου να λιώσει το πλαστικό. Στο σημείο αυτό το στερεό πλαστικό γίνεται υγρό. Στη συνέχεια ο εκτυπωτής χρησιμοποιώντας G-Code μεταφέρει τον Extruder στα σημεία στα οποία έχει πάρει από το σχέδιο που κατασκευάσαμε και στρώμα-στρώμα από κάτω προς τα πάνω δημιουργεί το φυσικό υλικό. Ο Extruder δηλαδή είναι σαν ένα θερμοπίστολο, το υλικό όταν βγει στερεοποείται στην θερμοκρασία περιβάλλοντος. Η διαδικασία εκτύπωσης μπορεί να πάρει από λεπτά μέχρι και μέρες, ο

χρόνος εξαρτάται από το μέγεθος του αντικειμένου και την ανάλυση που έχουμε επιλέξει στον εκτυπωτή.

Το υλικό που χρησιμοποιήθηκε στην κατασκευή ήταν PLA καθώς προσφέρει μεγαλύτερη ακρίβεια και δεν λυγίζει δημιουργώντας κάμψεις, αν το αντικείμενο είναι αρκετά ψηλά από το τραπέζι σε σχέση με το ABS. Στα αρνητικά του όμως είναι λιγότερο ανθεκτικό και μπορεί να σπάσει εύκολα. Γενικά η εκτύπωση σε PLA ή ABS είναι αρκετά οικονομική, ενώ ανεβαίνει, όσο μεγαλύτερη ακρίβεια αναζητάται, ή αν η εκτύπωση γίνει με άλλου είδους υλικού άλλων ιδιοτήτων.

3.6 Τελική Κατασκευή

Η τελική κατασκευή απεικονίζεται στις επόμενες φωτογραφίες (3.6.1,3.6.2,3.6.3,3.6.4)



Εικόνα 3.6.1 *: Τελική κατασκευή*

Εικόνα 3.6.2 : *Μηχανισμός εγκατεστημένος πάνω σε πόρτα (1)*

Εικόνα 3.6.3 : Μηχανισμός εγκατεστημένος πάνω σε πόρτα (2)

Εικόνα 3.6.4 : *Μηχανισμός εγκατεστημένος πάνω σε πόρτα (3)*

3. 7 Δυσκολίες Υλοποίησης και βελτίωση

3. 7. 1 Δυσκολίες Υλοποίησης Κατασκευής

Κώδικας

Μια από τις κυριότερες δυσκολίες ήταν η ενσωμάτωση του Blynk στο ESP8266 και η γραφή του κώδικα γύρω από τις ιδιοτροπίες του, όπως:

Επανειλημμένα Request στο server αναγκάζουν το ESP8266 να αποσυνδεθεί από το server του Blynk με αποτέλεσμα να χάνεται η ασύρματη συνδεσιμότητα και δικτυώση. Ο λόγος είναι γιατί επανειλημμένα request στο server της εταιρείας μπορούν να διακοψούν την λειτουργία του ως μια μορφή DDOS. Για παράδειγμα οι ειδοποιήσεις Push δεν επιτρέπονται να αποστέλνονται παραπάνω από μία φορές το λεπτό όπως και πολλές άλλες λειτουργίες με ίδιους η ελάχιστα διαφορετικούς χρονικούς περιορισμούς.

Η δυσκολία της υπορουτίνας powerwarning ήταν ότι, όχι μόνο έπρεπε να αποστέλνεται το πολύ μία φορά μόνο ανά λεπτό η ειδοποίηση, αλλά και μόνο όταν αλλάζει η τιμή κάτω από το όριο επίσης μία μόνο φορά, ενώ παράλληλα ο έλεγχος της τιμής της τάσης έπρεπε να γίνεται συνέχεια κάτι το οποίο επιτεύχθηκε με αρκετές συνθήκες IF και δημιουργία ξεχωριστών πολλαπλών μεταβλητών.

Οι εντολές delay στον κώδικα δεν έπρεπε να ξεπερνούν συνολικά τα 10 δευτερόλεπτα από εντολή σε εντολή blynk καθώς τότε το ESP8266 έκανε timeout και έχανε την σύνδεση με το server.

Το μεγαλύτερο μακράν πρόβλημα ήταν η αδυναμία του συστήματος να λειτουργήσει χωρίς συνδεσιμότητα στο internet. Η ρουτίνα που χρησιμοποιεί το Blynk για να συνδεθεί σε ένα ασύρματο δίκτυο και στην συνέχεια στο Internet γινόταν με το «Blynk.begin(auth)», η οποία αντλεί τις εντολές από την βιβλιοθήκη του Blynk. Η επίλυση έγινε με την διάσπαση σε κομμάτια και ανάλυση των εντολών αυτών μία προς μία, έτσι ώστε να χρησιμοποιηθούν μεμονωμένα σε ρουτίνες που δημιουργήθηκαν με συνθήκες IF και επιτρέπουν το «blynk.run» να τρέξει μόνο όταν υπάρχει σύνδεση στο διαδίκτυο. Ο λόγος λοιπόν για τον οποίο κατάρρευσε (“κράσαρε”) το ESP8266, ήταν επειδή το «blynk.run» προσπαθεί να ανταλλάξει δεδομένα με το server ξανά και ξανά χωρίς επιτυχία με το αναμενόμενο αποτέλεσμα, έτσι μόλις το hardware φτάσει στην συγκεκριμένη γραμμή κώδικα κολλάει και καταρρέει όλο το σύστημα. Χωρίς το «Blynk.run()» δεν υποστηρίζεται καμία λειτουργία, δηλαδή είναι σαν η ενσωμάτωση του Blynk στο Hardware να μην έγινε ποτέ.

ESP8266 & Hardware

Η αρχική ιδέα ήταν να ενσωματωθούν δύο φυσικά LED και ένα Buzzer. Ένα κόκκινο και ένα πράσινο για την ένδειξη του αν είναι ή όχι, η πόρτα κλειδωμένη και ένα Buzzer να ηχεί μόλις ανοίγει η πόρτα με την ρουτίνα tempUnlocking. Λόγω όμως του περιορισμού των pins και των ιδιοτροπιών μερικών από αυτών οι επιλογές περιορίστηκαν σημαντικά. Πιο συγκεκριμένα κατά την εκκίνηση του ESP8266 στο διάστημα των 200ms το GPIO0 δεν παραμένει σταθερό και πρέπει να παρέλθει το χρονικό αυτό διάστημα, επιπλέον, όταν αρχικά είχε συνδεθεί ο σερβοκινητήρας στο GPIO15 δεν επικοινωνούσε με το σύστημα εκτός και αν αποσυνδεόταν το Pin του σήματος και επανασυνδεόταν μετά. Η λύση ήταν ότι έπρεπε να ρυθμιστεί ως LOW το Pin αυτό κατά την εκκίνηση και αργότερα HIGH όπως φαίνεται και στο Sketch. Μάλιστα προτείνεται από την κοινότητα του ESP8266 το GPIO0, GPIO15 & GPIO2 να μην χρησιμοποιούνται λόγω των ανωμαλιών αυτών.

Μία άλλη επιλογή για να ελευθερωνόταν ένα Pin για το Lock Status ήταν να χρησιμοποιηθεί ένα εικονικό pin του blynk αλλά σε περίπτωση που δεν υπήρχε σύνδεση στο δίκτυο ο μηχανισμός δεν θα μπορούσε να διαβάσει σε τι κατάσταση είναι η κλειδαριά και έτσι το σκεπτικό αυτό εγκαταλείφθηκε άμεσα. Για τους λόγους που αναφέρθηκαν το GPIO2 δεν χρησιμοποιήθηκε καν, καθώς η σύνδεση του για οποιοδήποτε λόγο ανάγκαζε το ESP8266 να κάνει επανεκκινήσεις επανειλημμένα χωρίς τέλος.

Όπως αναφέρθηκε στον κώδικα για να διαβαστεί η τάση του ESP8266 χρησιμοποιούμε την εντολή «ADC_MODE(ADC_VCC)» μαζί με «ESP.getVcc()». Αν για τον οποιοδήποτε λόγο είχαμε συνδέσει το ADC Pin με κάποιο εξάρτημα τότε η εντολή αυτή δεν θα λειτουργούσε.

Οι μπαταρίες λιθίου γενικά είναι επικίνδυνες αν δεν επέλθει προσοχή κατά την φόρτιση τους η γίνει λάθος σχεδιασμός κυκλώματος που να καταλήγει σε υπερφόρτιση.

Η τεχνολογία FDM - 3D Printing παρόλο που είναι αρκετά φθηνή για κατασκευή πρωτοτύπων δεν είναι τέλεια. Δεν έχει μεγάλη ακρίβεια και δεν πετυχαίνει ακριβώς αυτό που δίνεται σε ένα σχέδιο αφήνοντας αρκετές ατέλειες οι οποίες πρέπει να τακτοποιηθούν χειροκίνητα με διάφορα εργαλεία.

Άλλα

Στο διαδίκτυο παρόλο που υπάρχουν πολλές πήγες για διάφορα θέματα η κατάσταση έχει φτάσει στο σημείο όπου υπάρχει περισσότερη ποσότητα παρά ποιότητα. Το διδακτικό

υλικό που είναι αναρτημένο συνήθως ήταν λάθος και επιβαλλόταν η διασταύρωση με τουλάχιστον τρεις διαφορετικές πηγές.

Τα περισσότερα ηλεκτρονικά εξαρτήματα παραγγέλθηκαν από την Κίνα καθώς δεν υπήρχε διαθεσιμότητα στην Ελλάδα. Ως αποτέλεσμα ο χρόνος παράδοσης των παραγγελιών ήταν τουλάχιστον τρεις εβδομάδες την φορά καταλήγοντας στην εκ νέου επιμήκυνση της πτυχιακής εργασίας.

Όταν σχεδιάστηκε η συνδεσμολογία του ηλεκτρονικού κυκλώματος στο πρόγραμμα Fritzing, αρκετά από τα εξαρτήματα δεν υπήρχαν για να τοποθετηθούν στο σχεδιάγραμμα. Έτσι αρκετά από αυτά έπρεπε να σχεδιαστούν αυτοτελώς από την αρχή στο Inkscape ως. svg(**Scalable Vector Graphics)** αρχεία, ακολούθως να φορτωθούν στο Fritzing, να υποστούν την απαραίτητη εργασία για τα σημεία των Pin και τελικώς να εισαχθούν στο κύκλωμα.

3. 7. 2 Περιθώρια Βελτίωσης

Μερικές προτάσεις βελτίωσης της κατασκευής είναι οι ακόλουθες:

Θα ήταν προτιμότερο να προστεθεί μία ρυθμιζόμενη πλατφόρμα στη βάση για να προσαρμόζεται η απόσταση της βάσης από το κλειδί. Έτσι ο μηχανισμός θα μπορούσε να τοποθετηθεί εύκολα σε οποιαδήποτε πόρτα χωρίς να υποστεί τροποποιήσεις

Οι γραμμικοί ρυθμιστές του κυκλώματος δεν είναι ιδιαίτερα αποδοτικοί λογω της κατασκευης τους. Θα ήταν συνετό να αντικατασταθούν με εναλλασσόμενους ρυθμιστές τάσης (switching Regulator), οι οποίοι έχουν μακράν καλύτερη απόδοση για μεγαλύτερη αυτονομία του συστήματος.

Εφόσον ο χώρος για τις μπαταρίες έχει δημιουργηθεί θα ήταν προτιμότερο να γίνουν κάποιες αλλαγές στο κύκλωμα, είτε στο τμήμα της συνδεσμολογίας, είτε στην προθήκη άλλων ηλεκτρονικών εξαρτημάτων ώστε να συνδεθούν οι 2 μπαταρίες παράλληλα για ακόμη μεγαλύτερη αυτονομία

Τελικά, θα μπορούσε να προστεθεί και η δυνατότητα ενημέρωσης του sketch χωρίς την σύνδεση του ESP8266 σε υπολογιστή αλλά απευθείας από το διαδίκτυο (OTA – Over The Air) με αρκετές αλλαγές στο υπάρχον sketch και προθήκη επιπλέον βιβλιοθηκών σχετικές με την λειτουργία αυτή.

Το συνολικό κόστος κατασκευής ανήλθε στα 59,05€ και παρόλο που το κόστος είναι σημαντικά χαμηλότερο από αντίστοιχες λύσεις του εμπορίου που ξεκινούν από 120€ και άνω θα μπορούσε το κόστος να μειωθεί περαιτέρω σε περίπτωση που υπήρχε κάποια οικονομικότερη επιλογή στην τρισδιάστατη εκτύπωση των εξαρτήματων καθώς από τα 59,05€ τα 30€ ήταν το κόστος των εξαρτημάτων του εκτυπωτή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://en.wikipedia.org/wiki/Arduino
[2] https://github .com/ellak-monades-aristeias/CruiseCabinDigitalSignage/blob/master/Hardware/ArduinoUNO.md
[3] https://learn.sparkfun.com/tutorials/serial-communication
[4] http://docs.blynk.cc/
[5] https://en.wikipedia.org/wiki/ESP8266
[6] http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family
[7] http://www.kloppenborg.net/images/blog/esp8266/esp8266-esp12e-specs.pdf
[8] https://el.wikipedia.org/wiki/RFID
[9] https://github.com/Jorgen-VikingGod/ESP8266-MFRC522/issues/6
[10] https://github.com/Jorgen-VikingGod/ESP8266-MFRC522
[11] http://www.forward.com.au/pfod/ESP8266/GPIOpins/index.html
[12] http://bbs.espressif.com/viewtopic.php?t=774
[13] https://zoetrope.io/tech-blog/esp8266-bootloader-modes-and-gpio-state-startup
[14] http://www.makeuseof.com/tag/technology-explained-how-do-rfid-tags-work/
[15] http://news.thomasnet.com/imt/2014/03/04/passive-vs-semi-passive-vs-active-tags-in-rfid
[16] http://stackoverflow.com/questions/32839396/how-to-get-the-uid-of-rfid-in-arduino
[17] https://en.wikipedia.org/wiki/Pulse-width_modulation
[18] https://en.wikipedia.org/wiki/Servo_control
[19] http://arduino.stackexchange.com/questions/1321/servo-wont-stop-rotating
[20] https://www.arduino.cc/en/Tutorial/PWM
[21] http://www.esp8266.com/viewtopic.php?f=33&t=2485&start=20
[22] http://community.blynk.cc/t/esp8266-freeze/1520/9
[23] https://en.wikipedia.org/wiki/Breadboard
[24] http://www.egr.msu.edu/classes/ece480/capstone/fall14/group07/PDFs/PWM_Application_Note.pdf
[25] http://www.silabs.com/Support%20Documents/TechnicalDocs/CP2102-9.pdf
[26] http://www.banggood.com/2S-Li-ion-Lithium-Battery-18650-Charger-Protection-Module-Board-3A-7_4V-8_4V-p-1044554.html
[27] https://en.wikipedia.org/wiki/Linear_regulator
[28] http://forum.arduino.cc/index.php?topic=162674.0
[29] http://community.blynk.cc/t/continuing-loop-while-reconnecting/620
[30] https://en.wikipedia.org/wiki/3D_printing
[31] https://el.wikipedia.org/wiki/%CE%A4%CF%81%CE%B9%CF%83%CE%B4%CE%B9%CE%AC%CF%83%CF%84%CE%B1%CF%84%CE%B7_%CE%B5%CE%BA%CF%84%CF%8D%CF%80%CF%89%CF%83%CE%B7
[32] http://www.dx.com/p/esp-12e-wi-fi-module-adapter-board-esp8266-wireless-transceiver-module-w-pcb-antenna-for-arduino-386920
[33] http://www.theengineeringprojects.com/2015/08/interfacing-rfid-rc522-arduino.html
[34] https://ae01.alicdn.com/kf/HTB12E9bKFXXXXaLXVXXq6xXFXXXm/100PCS-13-56MHz-font-b-Small-b-font-Smart-font-b-RFID-b-font-IC-Key.jpg
[35] http://grobotronics.com/images/companies/1/L78xx.pdf
[36] http://www.terapeak.com/worth/36g-as3103-servo-for-trex-trex-align-600-rc-helicopter-plane-boat-car-i/351500016832/
[37] http://www.autobotic.com.my/as3103-continuous-servo-360deg
[38] https://en.wikipedia.org/wiki/Servo_(radio_control)
[39] http://dilium.es/circuitos-bmspcbpcm/710-circuitos-bmspcbpcm-74v25afdc-2s.html
[40] http://www.dx.com/p/cp2102-usb-to-ttl-stc-promini-download-module-for-arduino-265730
[41] https://www.aliexpress.com/item/1PCS-New-18650-lithium-battery-2600mAh-3-7v-Rechargeable-Li-ion-Battery/32567982965.html
[42] https://www.aliexpress.com/item/Battery-Storage-Case-Plastic-for-2-x-18650-Box-Holder-Black-With-Wire-Lead-hot-new/32514280834.html
[43] https://www.aliexpress.com/item/New-Black-Push-Button-Mini-Switch-6A-10A-110V-250V-KCD1-101-2Pin-Snap-in-On/32438204147.html

[44] https://www.aliexpress.com/item/10pcs-Lockless-Momentary-ON-OFF-Push-button-Black-Mini-Push-button-Switch/32622558592.html
[45] https://www.aliexpress.com/popular/jack-conector.html
[46] https://www.aliexpress.com/item/20pcs-DC25M-metal-DC-Power-Jack-Socket-2-1mm-Jacks-Connectors/32714989337.html
[47] https://solarbotics.com/product/45040/
[48] https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard