ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΤΕΦ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ

2017

Έξυπνη λειτουργία φωτεινών σηματοδοτών με δυνατότητα απομακρυσμένου ελέγχου.

Αργυριάδης Κων/νος

ΑΜ : 37430

6/6/2017

Περιεχόμενα

1. Εισαγωγή
   - 1.1. Ευχαριστήριο Σημείωμα *(σελ . 4)*
   - 1.2. Περίληψη *(σελ . 5 – 6)*
   - 1.3. Ιστορικά στοιχεία αυτοματισμών *(σελ . 7 – 8)*
   - 1.4. Αυτοματισμοί δρόμων σήμερα *(σελ . 9 – 10)*

2. Βασικές Γνώσεις Αυτοματισμών
   - 2.1. Ορολογία – Ορισμοί *(σελ . 11 – 14)*
   - 2.2. Ταξινόμηση των Σ.Α.Ε. *(σελ . 15 – 20)*
   - 2.3. Ηλεκτρολογικό/ Ηλεκτρονικό κύκλωμα *(σελ . 21)*

3. Το Project
   - 3.1. Συνοπτική περιγραφή λειτουργίας *(σελ . 23 - 25)*
   - 3.2. Arduino
     - 3.2.1. Τι είναι το Arduino? *(σελ . 26 – 27)*
     - 3.2.2. Κυκλώματα Πλακετών Arduino *(σελ . 28 – 31)*
     - 3.2.3. Γλώσσα Προγραμματισμού και Εντολές του Arduino *(σελ . 32 – 46)*
   - 3.3. I^2c Bus *(σελ . 47 – 48)*
   - 3.4. Blynk
     - 3.4.1. Τι είναι το Blynk? *(σελ . 49)*
     - 3.4.2. Πώς λειτουργεί? *(σελ . 49)*
     - 3.4.3. Εγκατάσταση Blynk – Σύνδεση με Arduino *(σελ . 50 - 59)*

4. Το πρόγραμμα (Software) *(σελ . 60 - 186)*
5. Το κύκλωμα (Hardware)
   - 5.1. Το κύκλωμα της πτυχιακής *(σελ . 187 – 193)*
   - 5.2. Το κύκλωμα μιας πραγματικής κατανάλωσης *(σελ . 194)*
   - 5.3. Ανακεφαλαίωση – Επιπλέον δυνατότητες λειτουργίας αυτοματισμού *(σελ . 195 – 197)*

6. Κοστολόγιο Αυτοματισμού *(σελ . 198)*

7. Πηγές *(σελ . 199 – 200)*

1.1 Ευχαριστήριο Σημείωμα

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Κύριο Βαρσάμη Χρήστο, για την εμπιστοσύνη και το ενδιαφέρον που μου έδειξε κατά την ανάθεση της εργασίας, την πολύτιμη βοήθεια του , την συνεργασία του αλλά και τη δυνατότητα που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, με προεκτάσεις στην καθημερινή ζωή που παίζει κυρίαρχο ρόλο στον επαγγελματικό κλάδο που επέλεξα. Τον ευχαριστώ επίσης καθώς με την συγκεκριμένη ανάθεση, πιστεύω ότι θα με βοηθήσει στην μετέπειτα πορεία μου στον τομέα των αυτοματισμών.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη καθ' όλη τη διάρκεια των ακαδημαϊκών μου σπουδών αλλά και την ηθική συμπαράσταση κατά τη διάρκεια της συγγραφής της πτυχιακής αυτής εργασίας.

1.2 Περίληψη

Γεγονός είναι ότι σε καθημερινή βάση λίγο πολύ σε όλο τον κόσμο όλοι μας έχουμε ταλαιπωρηθεί από την κίνηση που επικρατεί στους δρόμους. Αυτό συνήθως οφείλεται στον αυξημένο όγκο αυτοκινήτων και μέσων μαζικής μεταφοράς κυρίως στα αστικά κέντρα καθώς επίσης και στην έλλειψη εξελιγμένων μέσων ελέγχου της κυκλοφορίας.

Στην συγκεκριμένη πτυχιακή εργασία θα προσπαθήσουμε να κάνουμε την καθημερινότητα μας στους δρόμους λίγο πιο ευχάριστη. Όλοι έχουμε ευχηθεί να κρατούσε λίγο παραπάνω αυτό το φανάρι! Θα δούμε ότι κάτι τέτοιο εάν είναι αναγκαίο μπορεί να γίνει. Θα φροντίσουμε επίσης η λειτουργία των μέσων μαζικής μεταφοράς που κινούνται παράλληλα με το αστικό δίκτυο κυκλοφορίας να παραμένει ανεπηρέαστη από τους φωτεινούς σηματοδότες των αυτοκίνητων, κάτι το οποίο θα εκτόξευε την αξιοπιστία αλλά και την αποδοτικότητα τους.

Δεν θα μπορούσαμε όμως να χαρακτηρίσουμε μια τέτοια προσπάθεια τεχνολογικά εξελιγμένη εάν στο έτος που βρισκόμαστε δεν είχαμε χειρισμό μέσω του διαδικτύου. Οπότε θα δούμε πως είναι αυτό εφικτό και όλες τις υπέροχες δυνατότητες που μπορεί κάτι τέτοιο να προσφέρει!

Με λίγα λόγια, θα φτιάξουμε μια έξυπνη διασταύρωση. Η οποία θα έχει την δυνατότητα εύκολου απομακρυσμένου ελέγχου μέσω οποιασδήποτε android συσκευής. Το κάθε της φανάρι θα λειτουργεί αυτόνομα και τελικά σαν μέρος ενός μεγαλύτερου συνόλου της διασταύρωσης, καθώς επίσης θα δούμε ότι ένα τέτοιο σύστημα μπορεί να επικοινωνεί και να συνεργάζεται με παρόμοια τέτοια συστήματα ελέγχοντας τελικά όχι μόνο μια διασταύρωση αλλά ένα πολύ μεγαλύτερο σύνολο.

Τέλος, θα τα κάνουμε όλα αυτά με πολύ χαμηλό κόστος , αφού θα χρησιμοποιήσουμε το **Arduino** μια πολύ οικονομική πλατφόρμα κατασκευής ηλεκτρονικών ανοιχτού κώδικα σε συνδυασμό με την επίσης ελεύθερη προς χρήση εφαρμογή για android **Blynk** η οποία έχει σχεδιαστεί για να ελέγχει πλατφόρμες σαν το Arduino.

1.2 Abstract

It's a fact that on a daily basis all over the world we have all suffered from traffic during the rush hours in the streets. This is usually due to the excessive amounts of cars and public transportation vehicles mainly in the urban centers in combination with the lack of advanced traffic control systems.

In this thesis we will try to make our daily street routine a little more pleasant. We have all wished for that one traffic light to be kept on for a little bit more! We will see that this is possible if necessary. We will also provide a clear passage to the public transportation vehicles which move along with the urban network. Something like that would launch their reliability and efficiency.

However, we wouldn't be able to call an attempt like that technologically advanced in the year 2017, if we didn't have handling over the internet. So we will see how something like that possible is and all the benefits that we can get from an act like this.

In a few words, we will build a smart crossroad which will have the ability of easy remote control through every android device. Every single traffic light will work autonomously and in the end they will all work together as a part of the crossroad. We will also see that a system like this one could cooperate and interact with several similar systems so they can all together create a much bigger plan.

Finally, we will do all this in very low cost. We will use the Arduino, which is a relatively economical open-source electronics platform combined with the Blynk App, which is also free to use and is designed to control the outputs of hardware of our choice.

1.3 Ιστορικά Στοιχεία Αυτοματισμών

Τα συστήματα του αυτοματισμού φαίνεται ότι είχαν εφαρμογή στις ανάγκες του ανθρώπου από τους αρχαιοτάτους χρόνους. Σαν πρώτες εφαρμογές εμφανίζονται στα χρόνια των αρχαίων Ελλήνων με τη χρήση του αυτοματισμού στη ρύθμιση της στάθμης του νερού σε δεξαμενές.

Στη συνέχεια βλέπουμε συστήματα αυτοματισμού κατά την Αλεξανδρινή και Ρωμαϊκή περίοδο , γύρω στον 3ο αιώνα π.Χ. όπου ο Αλεξανδρεύς χρησιμοποίησε τον μηχανισμό αυτό για την κατασκευή του ξακουστού τότε υδάτινου ωρολογίου.

Αργότερα, γύρω στο 10 μ.Χ. την ίδια μέθοδο χρησιμοποίησε και ο Φίλων για την αυτόματη ρύθμιση της στάθμης μιας λάμπας λαδιού. Ακολούθησε ο Ήρωνας ο Αλεξανδρινός που έζησε τον 1ο αιώνα μ.Χ. χρησιμοποιώντας διάφορα είδη αυτοματισμού, η λειτουργία των οποίων βασίζονταν στην υδραυλική πίεση. Το πιο γνωστό σύστημα ελέγχου ήταν ο ρυθμιστής ή ο μηχανισμός ανοίγματος και κλεισίματος στις πύλες ενός ιερού ναού.

Το πιο γνωστό σύστημα αυτοματισμού των νεωτέρων χρόνων, που λειτούργησε με μεγάλη επιτυχία ήταν ο ρυθμιστής θερμοκρασίας. Σε αυτό το σύστημα χρησιμοποιήθηκε ανάδραση από τον Ολλανδό Κορνήλιο Ντρεμπέλ (1572 - 1633).

Μετά από 50 χρόνια περίπου επινοήθηκε από τον Ντένις Πάπεν η αυτόματη ρύθμιση πίεσης για λέβητες.

Ο Ιβάν Πουλζούνοφ το 1765 είναι ο πρώτος Σοβιετικός που εφεύρε το σύστημα ανατροφοδότησης. Με το σύστημα αυτό ρύθμιζε τη στάθμη του νερού σε ατμολέβητα

Ο James Watt (1769) είναι ο πρώτος επιστήμονας που κατάφερε να ρυθμίσει αυτόματα την ταχύτητα της ατμομηχανής χρησιμοποιώντας το ρυθμιστή με τις σφαίρες (flyball governor) ή φυγοκεντρικό ρυθμιστή.

Μέχρι το 1868 τους ερευνητές δεν τους απασχόλησε καθόλου η μαθηματική περιγραφή της λειτουργίας των κατασκευών τους. Έτσι προσπαθώντας να τελειοποιήσουν το σύστημά τους αντιμετώπιζαν και προβλήματα αστάθειας, με αποτέλεσμα να καταστρέφεται η μηχανή όταν δεν επενέβαινε ο χειριστής. Έτσι στη περίοδο αυτή αναπτύχθηκε η θεωρία των συστημάτων αυτομάτου

ελέγχου. Ο Maxwell και Vyshnegradski ανέπτυξαν τη μαθηματική θεωρία για τους αυτόματους ρυθμιστές.

Τα συστήματα αυτομάτου ελέγχου (Σ.Α.Ε) αναπτύχθηκαν ραγδαία τα τελευταία 50 χρόνια. Συγκεκριμένα το 1934 ο Hazen δημοσίευσε το άρθρο του με τίτλο "Theory of servomechanisms". Την ίδια περίοδο ο Bode (Μπόντ), ο Nyquist (Νάικουιστ) και ο Black (Μπλάκ), επιστήμονες των εργαστηρίων Bell Telephone, ασχολήθηκαν με εργασίες επί των ενισχυτών ανατροφοδότησης και οι εργασίες τους ολοκληρώθηκαν γύρω το 1957. Η θεωρία αυτή των παραπάνω ερευνητών, και κυρίως του Black, χαρακτηρίζεται ως κλασσική θεωρία των συστημάτων αυτομάτου ελέγχου. Κατά τη διάρκεια του Β' Παγκοσμίου πολέμου έγιναν εφαρμογές των συστημάτων αυτών στις πολεμικές μηχανές, π.χ. ο αυτόματος πιλότος του αεροπλάνου, ο αυτόματος σκοπευτής στα πυροβόλα, το αυτόματο σύστημα ανίχνευσης κινητών στόχων με το ραντάρ κ.λπ.

Βλέπουμε λοιπόν ότι οι αυτοματισμοί δεν είναι κάτι καινούργιο στις ημέρες μας, η ιστορία τους ξεκινά απ τους αρχαίους χρόνους π.Χ. και συνεχίζονται να εξελίσσονται ακόμα και σήμερα.

Σκοπός ενός συστήματος αυτόματου ελέγχου ήταν, και παραμένει, η διευκόλυνση της καθημερινότητας και η καλύτερη εξυπηρέτηση των ανθρωπίνων αναγκών. Άρα ως φυσικό επακόλουθο ο τομέας χρήσης συστημάτων αυτόματου ελέγχου είναι αχανής. Οι πρακτικές εφαρμογές είναι απεριόριστες. Οι δυνατότητες τους είναι ατελείωτες.

Λίγο πολύ οτιδήποτε χρησιμοποιούμε στις ημέρες μας από τις βασικές ηλεκτρικές συσκευές του σπιτιού μας μέχρι και τα αυτοκίνητα μας εμπερικλείει ένα σύστημα αυτόματου ελέγχου.

Το ίδιο ισχύει και για τους φωτεινούς σηματοδότες ελέγχου της κυκλοφορίας! Είναι ένα τρανό παράδειγμά πλήρως αυτοματοποιημένου συστήματος ελέγχου.

1.4 Αυτοματισμοί δρόμων σήμερα

Ολοένα και περισσότερες πόλεις ανά τον κόσμο σήμερα έχουν αρχίσει να αναζητούν λύσεις για τα κυκλοφοριακά τους προβλήματα , κάτι το οποίο έχει κινήσει σε δράση πολλές εταιρείες μεγάλες και μικρές να βρούνε λύσεις. Συνεισφέροντας έτσι όχι μόνο στο κυκλοφοριακό τους πρόβλημα και στην προστασία του περιβάλλοντος από αχρείαστους ρύπους αλλά και στην προστασία των "απρόσεκτων" πεζών .Ας δούμε μερικά παραδείγματα!

Στην Θεσσαλονίκη από το 2012 με την βοήθεια του προγράμματος Compass4D , λειτουργεί εδώ και 5 σχεδόν χρόνια ένα συνεργατικό σύστημα μεταφορών. Τι είναι τα συνεργατικά συστήματα μεταφορών ; Είναι έξυπνοι φωτεινοί σηματοδότες που ειδοποιούν τον οδηγό σε οθόνη ή στο κινητό ή σε τάμπλετ για την ταχύτητα που πρέπει να κινείται, ώστε να συναντά πάντα «πράσινο» ή που διατηρούν το πράσινο όταν πλησιάζουν οχήματα έκτακτης ανάγκης (ασθενοφόρα, πυροσβεστικά) Έτσι κερδίζουν χρόνο και μειώνουν τις εκπομπές ρύπων. Βρίσκονται σε εφαρμογή κατά μήκος της Τσιμισκή , καθώς και σε επτά κόμβους της περιφερειακής οδού.

Το ίδιο πρόγραμμα χρησιμοποιείται επίσης στην Μπορντώ της Γαλλίας, στην Κοπεγχάγη στην Δανία , στο Χέλμοντ στην Ολλανδία , στο Νιούκαστλ στο Ηνωμένο Βασίλειο, στην Βερόνα στην Ιταλία και στο Βίγκο στην Ισπανία.

Στο Ηράκλειο της Κρήτης έκαναν την εμφάνιση τους τα πρώτα έξυπνα φανάρια με τεχνολογία LED και αντίστροφη μέτρηση για τους πεζούς . Τα οποί αποσκοπούν να μειώσουν τις απερίσκεπτες διαβάσεις των πεζών με κόκκινο.

Στην Ολλανδία στο Μπούτενγκράβεν αλλά και στην Γερμανία στο Αουγκσμουργκ η Ολλανδική εταιρεία HIG Traffic Systems ανέπτυξε και εγκατέστησε ένα σύστημα με την ονομασία " Light Line" το οποίο προειδοποιεί τους αφηρημένους από την χρήση του κινητού πεζούς και ποδηλάτες.

Στην Ολλανδία έχει επίσης αναπτυχθεί και τεθεί σε λειτουργία ένα σύστημα το οποίο δίνει προτεραιότητα σε ποδήλατα και λεωφορεία πρώτα, και μετά στα αυτοκίνητα. Μερικά από τα φανάρια έχουν επίσης ανιχνευτές βροχής, και αλλάζουν σε πράσινο πιο συχνά για τα ποδήλατα όταν βρέχει.

Στις Η.Π.Α. στο Μίσιγκαν, στην πόλη Αν Αρμπορ. Το Αν Αρμπορ έτσι και αλλιώς χρησιμοποιούσε ένα σύστημα συντονισμού και λειτουργίας των φαναριών παρόμοιο με αυτό που χρησιμοποιούν μεγαλουπόλεις όπως το Λονδίνο, το Σαντιάγκο και το Τορόντο. Το νέο όμως σύστημα είναι προϊόν συνεργασίας της Siemens και της βρετανικής εταιρείας Imtec και θεωρείται εξαιρετικά πρωτοποριακό!

Χρησιμοποιεί κάμερες και αισθητήρες που παρακολουθούν τα οχήματα που πλησιάζουν ή βρίσκονται μπροστά στα φανάρια.

Τα φανάρια αυτά διασυνδέονται μεταξύ τους και όλα τα δεδομένα που συγκεντρώνονται αποστέλλονται στον κεντρικό υπολογιστή, στο κέντρο ελέγχου κίνησης. Ο υπολογιστής επεξεργάζεται τα δεδομένα και έτσι μπορεί να αναλύει την κίνηση των οχημάτων σε όλα τα σημεία που λειτουργεί το νέο σύστημα ώστε να αντιλαμβάνεται πού και πότε πρέπει να δίνει προτεραιότητα. Μέχρι στιγμής το νέο σύστημα έχει καταφέρει να μειώσει την κίνηση στους πολυσύχναστους δρόμους του Αν Αρμπορ κατά 12 % τις καθημερινές και κατά 21% τα Σαββατοκύριακα.

Στις Η.Π.Α. στο Τορόντο, η Samah El-Tantawy χρησιμοποιεί τεχνητή νοημοσύνη και βασίζεται στην θεωρία των παιγνίων έτσι ώστε τα φανάρια της να επικοινωνούν μεταξύ τους και βάσει στρατηγικών αποφάσεων να βρίσκουν κάθε στιγμή τι πρέπει να κάνουν για να βελτιστοποιηθεί η ροή της κυκλοφορίας. «Όπως οι παίκτες μιας ομάδας ποδοσφαίρου, που ο καθένας κοιτάει να σκοράρει, σκέφτεται όμως και τον βασικό στόχο όλης της ομάδας, που είναι η νίκη».

Στην προσομοίωση που έγινε για την πόλη του Τορόντο σε υπολογιστή, η καθυστέρηση των αυτοκινήτων σε 60 διασταυρώσεις μειώθηκε κατά 40%, ενώ ο χρόνος της μετακίνησης των αυτοκινήτων συνολικά μειώθηκε κατά 13-26%.

Στις Η.Π.Α. στο Πίτσμπουργκ, Η Surtrac έχει ξεκινήσει τα τελευταία χρόνια να αναβαθμίζει τα φανάρια χρησιμοποιώντας δεδομένα για το μέγεθος της κίνησης από κάμερες και ραντάρ και με βάση αυτά, προσπαθούν να κάνουν την διάβαση των οχημάτων από διασταυρώσεις όσο πιο γρήγορη γίνεται. Το AI σύστημα της Surtrac ξεκίνησε με 9 διασταυρώσεις το 2012, σε μια γειτονιά του Πίτσμπουργκ, και τώρα έχει επεκταθεί σε 50 διασταυρώσεις.

2.1 Ορολογία – Ορισμοί

Παρακάτω θα δούμε μερικές βασικές έννοιες και ορισμούς οι οποίοι θα μας βοηθήσουν να κατανοήσουμε καλύτερα από τι αποτελείται και πως λειτούργει ένα σύστημα αυτόματου ελέγχου. Είναι πολύ βασικό καθώς θα διαβάζουμε την συνέχεια αυτής της εργασίας να έχουμε πλήρη κατανόηση της παρακάτω ορολογίας που θα χρησιμοποιηθεί.

ΣΥΣΤΗΜΑ (SYSTEM)

Η βασική έννοια η οποία πρέπει να διευκρινιστεί από την αρχή είναι η έννοια του συστήματος. Η λέξη σύστημα έχει πολλές έννοιες, γι' αυτό δεν θα δώσουμε έναν γενικό ορισμό αλλά θα δούμε έναν πιο συγκεκριμένο σε σχέση με το αντικείμενο που θα μας απασχολήσει.

Σύστημα ονομάζουμε (από τη πλευρά του μηχανικού) ένα οργανωμένο σύνολο αλληλεπιδρώντων στοιχείων σχεδιασμένο με τέτοιο τρόπο ώστε να εκπληρώνει ειδικούς σκοπούς.

Στο σύνολο αυτό των στοιχείων είναι δυνατόν να λαμβάνουν μέρος μηχανές και άνθρωποι. Ο μηχανικός συστημάτων συντονίζει τις λειτουργίες των στοιχείων και έτσι το αντικείμενο του περιορίζεται στο σύστημα (σαν σύνολο) και στην λειτουργία του.

ΕΛΕΓΧΟΣ (CONTROL)

Η λέξη έλεγχος έχει την έννοια του ρυθμίζω, κανονίζω, δίδω εντολή.

Έτσι μπορούμε να πούμε ότι ο έλεγχος είναι η ρύθμιση της λειτουργίας ενός εξοπλισμού, π.χ. ο έλεγχος ήχου τόνου φωνής, έλεγχος φωτεινότητας κ.λπ.

ΣΥΣΤΗΜΑ ΕΛΕΓΧΟΥ (CONTROLS SYSTEM)

Σύστημα ελέγχου (από τη πλευρά του μηχανικού) ονομάζουμε ένα οργανωμένο σύνολο αλληλεπιδρώντων στοιχείων σχεδιασμένο με τέτοιο τρόπο ώστε να εκπληρώνει ειδικούς σκοπούς.

Στο σύνολο αυτό των στοιχείων είναι δυνατόν να λαμβάνουν μέρος μηχανές και άνθρωποι ώστε να μπορούν να ελέγχουν μια διεργασία ή ορισμένα μεταβλητά μεγέθη όπως:

• θέση (x,y,z)

• ταχύτητα

• πίεση

• ηλεκτρική τάση

• θερμοκρασία κ.λπ.

ΣΥΣΤΗΜΑΤΑ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ Σ . Α . Ε. (Automatic Control Systems).

Τα συστήματα ελέγχου, που λειτουργούν χωρίς την ανάμειξη του ανθρώπου, ονομάζονται συστήματα αυτομάτου ελέγχου.

Η Είσοδος (Input)

Είσοδος είναι η διέγερση ή ο ερεθισμός, ο οποίος προκαλείται σ' ένα σύστημα ελέγχου από μια εξωτερική πηγή ενέργειας, για να παραχθεί μια ειδική ανταπόκριση από το σύστημα ελέγχου.

Η Έξοδος (Output)

Είναι η πραγματική ανταπόκριση η οποία επιτυγχάνεται από το σύστημα ελέγχου. Μπορεί να είναι ή να μην είναι ίση με την οριζόμενη ανταπόκριση η οποία εφαρμόζεται από την είσοδο.

Πορεία ή Διαδικασία (Process)

Πορεία ή διαδικασία ονομάζουμε τη φυσική ή τεχνική ή εκούσια συνεχή αναπτυξιακή λειτουργία στοχεύοντας σε μία σειρά από βαθμιαίες αλλαγές ή ελεγχόμενες λειτουργίες που μας οδηγούν σ' ένα πρακτικό και χρήσιμο αποτέλεσμα.

Συγκρότημα (Plant) -Σύστημα τύπου ακολουθητή (follow-up) :

Τα συστήματα, των οποίων η έξοδος θα πρέπει να μεταβάλλεται σε συνάρτηση των μεταβολών του σήματος εισόδου (π.χ. σύστημα ελέγχου θερμοκρασίας χώρου).

Σύστημα τύπου σταθεροποιητή (regulator).

Τα συστήματα, των οποίων η έξοδος θα πρέπει να παραμένει σταθερή ακόμα και όταν υπάρχουν μεταβολές του σήματος εισόδου (π.χ. σταθεροποιητής τάσεως DC).

Ο Μετατροπέας (Transducer)

Μετατρέπει μια μορφή ενέργειας σε μια άλλη π.χ. ηλεκτρική σε μηχανική.

Ο Αθροιστής (Summation Point)

Είναι συσκευή που αθροίζει αλγεβρικά τα εισερχόμενα σήματα για να παράγει ένα σήμα εξόδου. Συνήθως αναφέρεται και σαν συγκριτής ή ανιχνευτής σφάλματος.

Ο Ελεγκτής (Controller)

Σε όλα σχεδόν τα συστήματα ελέγχου η είσοδος του ελεγκτή είναι το σφάλμα που παράγεται από τον αθροιστή στα συστήματα ελέγχου κλειστού βρόγχου ή την ίδια την είσοδο στα συστήματα ελέγχου ανοικτού βρόγχου.

Είναι μηχανισμός ελέγχου που παράγει μια έξοδο που οδηγεί την ελεγχόμενη διεργασία με σκοπό τον μηδενισμό του σφάλματος και γενικά την βελτιστοποίηση των χαρακτηριστικών του συστήματος.

Η Ελεγχόμενη διεργασία (Control process)

Κάθε φυσική ποσότητα, όπως θερμοκρασία, πίεση ή στάθμη υγρού μπορεί να ελεγχθεί μέσω διεργασίας που περιλαμβάνει κάθε τι που επηρεάζει τις φυσικές μεταβλητές.

Μ' άλλα λόγια, η ελεγχόμενη διεργασία περιλαμβάνει κάθε τι που απαιτείται για τον έλεγχο της φυσικής ποσότητας.

Το Επενεργούν στοιχείο (Actuator)

Το Επενεργούν Στοιχείο είναι η συσκευή που αποδίδει την απαιτούμενη ενέργεια (π.χ. κινητική) στην διεργασία (π.χ. η συσκευή που αναγκάζει την διεργασία να εξασφαλίσει την έξοδο).

Ανάδραση (Feedback)

Ανάδραση ή ανατροφοδότηση, ονομάζουμε την επιστροφή του σήματος από την έξοδο προς την είσοδο σε μία γραμμή μεταφοράς.
Η χρήση της ανάδρασης συνήθως επιφέρει ευστάθεια και ακρίβεια στο σύστημα. Ένα σύστημα μπορεί να χρησιμοποιεί πολλές αναδράσεις. Πάντως πρωτεύουσα ανάδραση είναι εκείνη όπου το σήμα εξόδου επιστρέφει και συγκρίνεται με την είσοδο.

Αν δεν υπάρχει καμία επικοινωνία μεταξύ εισόδου και εξόδου έχουμε σύστημα ανοιχτού βρόγχου, ενώ αν κάθε φορά παίρνουμε την έξοδο την ελέγχουμε και την οδηγούμε σε μια είσοδο αναφοράς έχουμε σύστημα κλειστού βρόγχου.

Ο κλάδος (δρόμος) που οδηγεί την έξοδο στην είσοδο λέγεται κλάδος ανάδρασης. Αν το σήμα εξόδου προστίθεται στην είσοδο έχουμε θετική ανάδραση και αν αφαιρείται αρνητική ανάδραση.

2.2 ΤΑΞΙΝΟΜΗΣΗ ΤΩΝ Σ.Α.Ε

Οι τομείς, στους οποίους βρίσκουν εφαρμογή τα Σ.Α.Ε, είναι πρακτικά απεριόριστοι. Από το σύστημα αυτόματου πιλότου των αεροσκαφών μέχρι το αυτόματο σύστημα ελέγχου ταχύτητας οχημάτων (cruise control system) και από τον τρόπο λειτουργίας μιας φωτογραφικής μηχανής μέχρι τις γραμμές παραγωγής των μεγάλων βιομηχανιών. Είναι δύσκολο να σκεφτούμε το σύγχρονο κόσμο χωρίς την παρουσία Συστημάτων Αυτόματου Ελέγχου.

Τα Σ.Α.Ε., ανάλογα με το πώς θέλουμε να τα μελετήσουμε, μπορούμε να τα κατατάξουμε σε διάφορες κατηγορίες. Ας δούμε δύο βασικές από αυτές .

1. Ανάλογα με τη φύση του μέσου ελέγχου

Ηλεκτρικά - Ηλεκτρονικά συστήματα

Το μέσο ελέγχου είναι ένα ηλεκτρικό σήμα, που μπορεί να ενισχυθεί κατάλληλα με ηλεκτρικούς ενισχυτές και να διεγείρει κάποιον ηλεκτρικό κινητήρα, που με την σειρά του θα κάνει ανάλογη διόρθωση στον μηχανισμό, με τον οποίο είναι συνδεδεμένος.

Πλεονεκτήματα :

1. Έχουν μεγάλο βαθμό απόδοσης.
2. Δεν χρειάζονται αεροσυμπιεστές, αντλίες.
3. Είναι καθαρά συστήματα και τυχόν διαρροές δεν προκαλούν βλάβες σε διπλανές συσκευές.

Μειονεκτήματα :

1. Μικρή συγκέντρωση ισχύος (δηλαδή ωφέλιμος .ισχύς/ βάρος .εγκατάστασης λόγος μικρός)
2. Προκαλούν σπινθήρες και δεν μπορούν να χρησιμοποιηθούν σε εύφλεκτο περιβάλλον.
3. Η συντήρηση τους είναι πολύπλοκη.
4. Συνοδεύονται με ακριβές ηλεκτρονικές μονάδες.

Πνευματικά συστήματα

Το μέσο ελέγχου είναι αέρας με πίεση που προέρχεται από κάποιο συμπιεστή (compressor) και διεγείρει κάποια βαλβίδα που μέσω κυλίνδρου με έμβολο κάνει την αναγκαία διόρθωση.

Πλεονεκτήματα

1. Είναι πολύ απλά στην κατασκευή και τη λειτουργία τους.
2. Χρησιμοποιούν αέρα που είναι ελεύθερος στην φύση και δεν εγκυμονεί κινδύνους πυρκαγιάς. (Η τυχόν διαρροή αέρα δεν προκαλεί βλάβες σε διπλανά συστήματα).
3. Οι μεταβολές της θερμοκρασίας του περιβάλλοντος δεν έχουν επίδραση στο ιξώδες του μέσου λειτουργίας.
4. Η συντήρηση τους είναι εύκολη και γίνεται σε αραιά χρονικά διαστήματα.
5. Αποθηκεύεται ποσότητα αέρα στις αεροφιάλες και έτσι λειτουργούν για λίγο, και μετά την διακοπή του αεροσυμπιεστή.

Μειονεκτήματα

1. Είναι αναγκαία η ύπαρξη τουλάχιστον δύο αεροσυμπιεστών, πολλών αεροφιαλών και κατάλληλου δικτύου σωλήνων.
2. Παρουσιάζουν δυσκολία στην λίπανση των κινούμενων μερών.
3. Λόγω της συμπιεστότητας του αέρα δεν έχουμε ταχεία ανάπτυξη δύναμης στην έξοδο του συστήματος ούτε ακρίβεια εξόδου.

Υδραυλικά συστήματα

Το μέσο ελέγχου είναι συνήθως λάδι που έρχεται από κάποια αντλία, ελέγχεται από βαλβίδα και κινεί υδραυλικό κινητήρα ή έμβολο κυλίνδρου.

Πλεονεκτήματα

1. Υψηλή απόδοση ακόμα και σε μεγάλη ισχύ και ταχύτητα.
2. Μπορούμε να επιτύχουμε μεγάλη ισχύ εξόδου.
3. Το λάδι είναι ασυμπίεστο και έχουμε ταχύτητα απόκρισης.
4. Αυτολίπανση με το κυκλοφορούν λάδι.
5. Σε περίπτωση ανωμαλίας ο επενεργητής (υδραυλικός κινητήρας ή κύλινδρος με έμβολο) παραμένει στην θέση του, γιατί υπάρχουν ειδικές ανεπίστροφες βαλβίδες στις γραμμές παροχής και επιστροφής.

Μειονεκτήματα

1. Συχνή συντήρηση των δικτύων υψηλής πίεσης για να μην υπάρχουν διαρροές.
2. Η τυχόν διαρροή λαδιού εγκυμονεί κινδύνους βλάβης των διπλανών συσκευών.
3. Με την αύξηση της θερμοκρασίας μεταβάλλεται το ιξώδες του λαδιού και φυσικά η απόδοση του συστήματος.
4. Η ύπαρξη λαδιού δημιουργεί ένα επιπλέον κόστος στην συντήρηση.
5. Αν στο υδραυλικό δίκτυο υπάρχει θυλάκιο αέρα δεν έχουμε ακρίβεια στην έξοδο (φαινόμενο αερισμού).

Ηλεκτροϋδραυλικά συστήματα

Η βασική διαφορά τους με τα προηγούμενα είναι ότι η ρυθμιστική βαλβίδα δεν είναι μηχανική αλλά ηλεκτρική δηλαδή λειτουργεί ανάλογα με κάποιο ηλεκτρικό σήμα που θα της στείλουμε από μακριά και καθορίζει έτσι το μέγεθος και τη φορά της ροής λαδιού προς τον υδραυλικό κινητήρα (ή έμβολο).

ΗλεκτροΠνευματικά συστήματα

Η βασική διαφορά τους με τα πνευματικά συστήματα είναι στο ότι διαθέτουν ηλεκτροπνευματική βαλβίδα που μπορεί να ρυθμιστεί από μακριά με κάποιο ηλεκτρικό σήμα.

2. Ανάλογα με το εάν χρησιμοποιείται ή όχι ανάδραση

Συστήματα Ελέγχου ανοιχτού βρόχου (χωρίς ανάδραση)

Ένα σύστημα ανοιχτού βρόγχου χρησιμοποιεί μία ενεργό συσκευή (που παράγει το σήμα εισόδου), για να ελέγξει απευθείας τη διεργασία χωρίς τη χρήση ανατροφοδότησης.

Τα χαρακτηριστικά του είναι :

1. Είναι πολύ απλής κατασκευής.
2. Η ακρίβεια εξαρτάται από τη ρύθμιση διαφόρων στοιχείων.
3. Δεν παρουσιάζουν συνήθως προβλήματα αστάθειας.

Ένα γενικό λειτουργικό διάγραμμα ενός συστήματος ελέγχου ανοικτού βρόγχου δίδεται παρακάτω στο Σχήμα 1 .

Σύστημα ανοικτού βρόγχου. Σχήμα 1 .

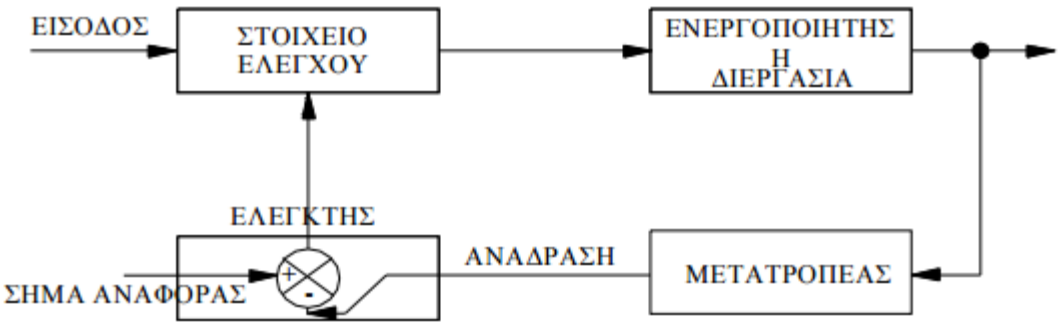
Σύστημα Ελέγχου κλειστού βρόχου (με ανάδραση)

Ένα σύστημα κλειστού βρόγχου χρησιμοποιεί τη μέτρηση του σήματος εξόδου και την ανατροφοδοτεί για να συγκριθεί με το σήμα αναφοράς (είσοδος - επιθυμητή έξοδος).

Τα χαρακτηριστικά του είναι :

1. Μεγάλη ακρίβεια.
2. Πολυπλοκότερα συστήματα.
3. Παρουσιάζουν προβλήματα αστάθειας.
4. Έχουν εύρος λειτουργίας.

Στα συστήματα κλειστού βρόγχου υπάρχουν διάφορες παραλλαγές. Εμείς θα παρουσιάσουμε ένα απλό διάγραμμα, όπως φαίνεται στο Σχήμα 2.



Σύστημα κλειστού βρόγχου (με ανάδραση). Σχήμα 2 .

Με βάση τις κατηγορίες και τα χαρακτηριστικά των συστημάτων αυτομάτου ελέγχου μέσα από όλα όσα έχουμε διαβάσει έως τώρα είδαμε λοιπόν ότι το σύστημα το οποίο θα φτιάξουμε εμείς,

Είναι :

- Εν μέρει ένα σύστημα αυτόματου ελέγχου.
  Καθώς σε πολλές φάσεις τις λειτουργίας του θα λειτουργεί αυτόνομα χωρίς την ανάμειξη του ανθρώπου.
- Ένα ηλεκτρικό - ηλεκτρονικό σύστημα (ανάλογα με την φύση του μέσου έλεγχου)
- Ένα σύστημα με αρκετές εισόδους (Inputs) και εξόδους (Outputs), εκ των οποίον θα λαβαίνουμε πληροφορίες και θα εκτελούμε μεταβολές στο σύστημα μας .
- Αποτελείται από μια Πορεία – Διαδικασία (Process)
- Ο έλεγχος και οι αποφάσεις θα γίνονται βάση ενός ελεγκτή (Controler).
- Ένα σύστημα τύπου ακολουθητή (follow-up).
- Επειδή η έξοδος του συστήματος θα μεταβάλλεται σε συνάρτηση της μεταβολής μιας εκ των εισόδων.
- Επιπλέον είναι ένα σύστημα αυτομάτου ελέγχου ανοιχτού βρόγχου χωρίς ανάδραση.
  Διότι , δεν συλλέγει πληροφορίες απ τις εξόδους του για να συγκρίνει με τις εισόδους , αλλά αντίθετα , κάθε φόρα ανάλογα με τα στοιχεία που έχει ως εισόδους θα ορίζει τις εξόδους.

Έχοντας υπόψη πλέον όλα τα παραπάνω και αφού έχουμε κατανοήσει πλήρως την γενική θεωρία των Συστημάτων Αυτομάτου Ελέγχου αλλά και πώς τελικά συνδέεται με το σύστημα μας , πάμε να δούμε από τι αποτελείται πρακτικά ένα πλήρως αυτοματοποιημένο σύστημα και πιο συγκεκριμένα ένα έξυπνο σύστημα λειτουργίας φωτεινών σηματοδοτών.

2.3 Ηλεκτρονικό / Ηλεκτρολογικό Κύκλωμα

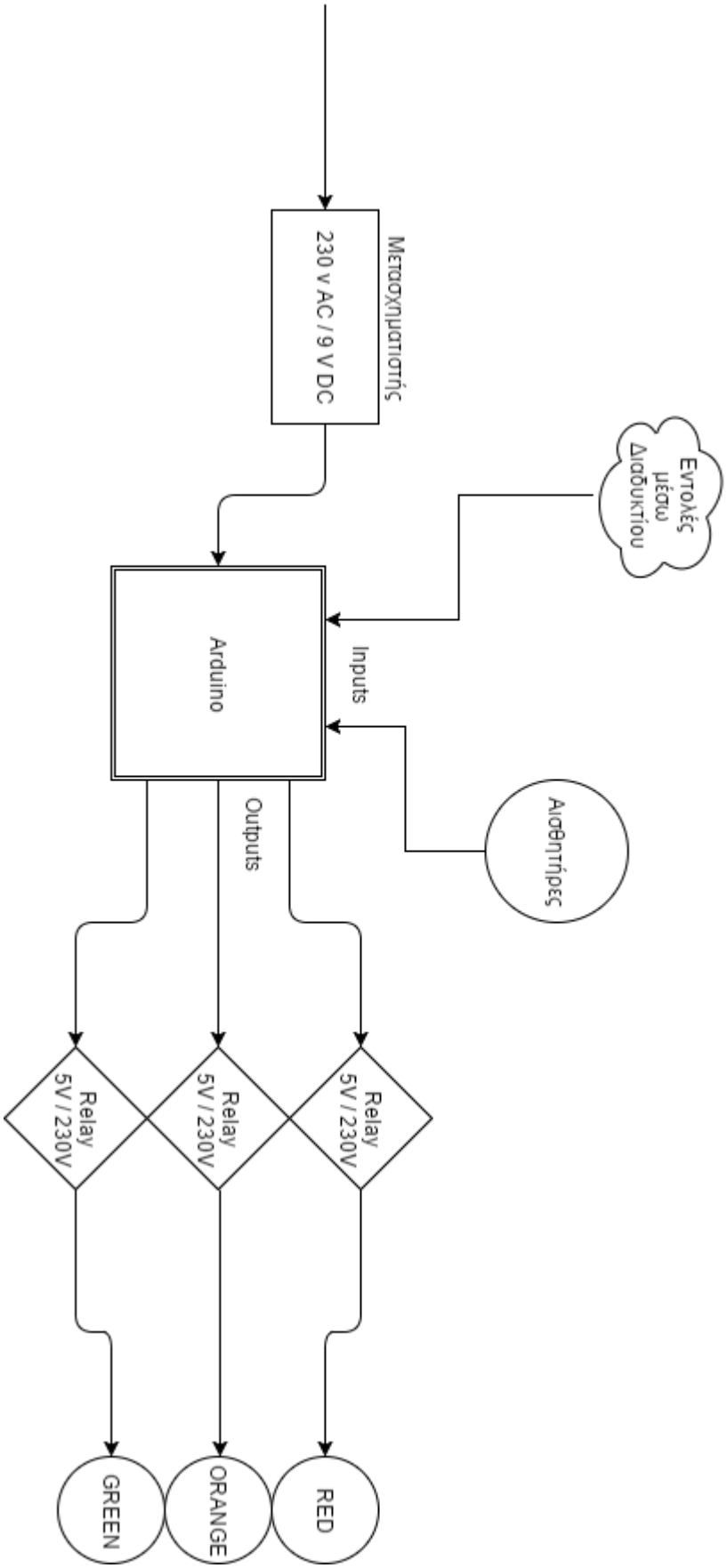
Όταν μιλάμε για αυτοματισμούς σίγουρα έρχονται στο μυαλό μας πλακέτες αντιστάσεις , πυκνωτές , ολοκληρωμένα, Μίκρο-ελεγκτές και σίγουρα πίσω απ όλα αυτά ένα πρόγραμμα! Αυτό είναι όμως το ένα μέρος ενός ολοκληρωμένου συστήματος αυτομάτου ελέγχου.

Για να υλοποιήσουμε, ένα σύστημα το οποίο θα ελέγχει ηλεκτρικές καταναλώσεις της καθημερινότητας μας θα πρέπει να φτάσουμε να ελέγχουμε τάσεις αντίστοιχες με αυτές του δικτύου οι οποίες είναι αρκετά μεγαλύτερες απ τα 5 ή και 3,3 Volt που χρησιμοποιούν τα περισσότερα ηλεκτρονικά αισθητήρια. Γιατί όμως δεν χρησιμοποιούμε απ' ευθείας στα αισθητήρια μας τάσεις ίσες με αυτές του δικτύου ;

Ας μην ξεχνάμε ότι ένα σύστημα αυτομάτου ελέγχου επηρεάζεται και αντιδρά ανάλογα με τα ερεθίσματα που δέχεται απ τα διάφορα αισθητήρια του, τα οποία πολλές φορές είναι εκτεθειμένα στο περιβάλλον αλλά και προς την ανθρώπινη επαφή! Κάτι τέτοιο καθιστά άμεσα αδύνατο να διαρρέεται από "μεγάλα" ρεύματα και τάσεις διότι υπάρχει πολύ μεγάλη πιθανότητα ηλεκτροπληξίας.
Επίσης ένα κύκλωμα με τόσο χαμηλές τάσεις είναι πολύ πιο οικονομικό ως προς την κατασκευή του αλλά και πολύ πιο ακίνδυνο ως προς την συντήρηση και την χρήση του.
Έτσι λοιπόν θα ξεχωρίσουμε το Ηλεκτρονικό απ το Ηλεκτρολογικό κομμάτι αυτής της Εργασίας.

Στο Ηλεκτρονικό Μέρος ανήκουν όλα τα αισθητήρια , οι μικροεπεξεργαστές και οι μεταξύ τους καλωδιώσεις (Hardware) καθώς και το πρόγραμμα (Software) με βάση το οποίο θα λειτουργεί ο αυτοματισμός μας. Ενώ στο Ηλεκτρολογικό θα ανήκουν όλες οι ελεγχόμενες καταναλώσεις, στην περίπτωση μας οι λαμπτήρες των φωτεινών σηματοδοτών και ίσως κάποια κάμερα ελέγχου της κυκλοφορίας καθώς επίσης και η δομή της τροφοδοσίας του ηλεκτρονικού μας συστήματος.

Στο Σχήμα 3. Μπορούμε να δούμε πολύ απλοϊκά ένα μπλοκ διάγραμμα στο οποίο απεικονίζεται παραστατικά η τροφοδοσία αλλά και η λειτουργία ενός έξυπνου φωτεινού σηματοδότη με την χρήση μικροεπεξεργαστή.


230 v AC / 9 V DC
Μετασχηματιστής
Εντολές μέσω Διαδυκτίου
Arduino
Inputs
Outputs
Αισθητήρες
Relay 5V / 230V
Relay 5V / 230V
Relay 5V / 230V
GREEN
ORANGE
RED

Σχήμα 3.

3.1 Συνοπτική Περιγραφή του Project

Στο συγκεκριμένο Project, σκοπός μας είναι να βελτιώσουμε λιγάκι την καθημερινότητα μας όσον αφόρα την κίνηση στους δρόμους. Είναι γεγονός ότι ειδικά στην Αθήνα τις ώρες αιχμής ο όγκος των αυτοκινήτων ξεπερνά κάθε προϋπάρχουσα μελέτη του κυκλοφοριακού δικτύου. Έτσι λοιπόν, αξιοποιώντας τα τεχνολογικά επιτεύγματα τις εποχής μας θα προσπαθήσουμε να δώσουμε έστω μια μικρή ανακούφιση στο ήδη υπάρχον σύστημα.

Στο παράδειγμα μας, θα χρησιμοποιήσουμε μια πραγματική διασταύρωση. Αυτήν μεταξύ των οδών Αρδήττου /Λεωφ. Βασιλέως Κωνσταντίνου και Λεωφ. Βασσιλίσης Όλγας , στο κέντρο της Αθήνας πολύ κοντά στο Καλλιμάρμαρο.
Ο λόγος που επέλεξα την συγκεκριμένη διασταύρωση είναι γιατί αποτελεί ένα εξαιρετικό παράδειγμα καθώς συνδυάζει φανάρια αυτοκινήτων , τραμ και ένα φανάρι το οποίο θα μπορεί να μένει μόνιμα ανοιχτό εκτός εάν το έξυπνο σύστημα μας του δώσει διαφορετική εντολή.

Για τον αυτοματισμό και την αυτόνομη λειτουργία του κάθε φαναριού θα χρησιμοποιηθεί μια πλακέτα τύπου Arduino Uno. Αυτό σημαίνει ότι τα φανάρια μας θα έχουν την δυνατότητα να λειτουργούν και να προγραμματιστούν ξεχωριστά το καθένα από τα υπόλοιπα , έτσι ώστε να μπορούν να ανταποκριθούν σε οποιαδήποτε λειτουργία - ανάγκη του συνολικού συστήματός μας.

Σαν σύνολο όμως, όλα τα φανάρια θα ελέγχονται από μια ξεχωριστή πλακέτα τύπου Arduino Mega η οποία θα φέρει το ρόλο του μητρικού επεξεργαστή.
Θα είναι αυτή πάνω στην οποία θα καταλήγουν όλοι οι αισθητήρες της διασταύρωσης. Θα επεξεργάζεται τα δεδομένα, θα λαβαίνει τις κατάλληλες αποφάσεις και θα μεταβιβάζει τελικά τις απαραίτητες εντολές σε όλες τις υπόλοιπες δευτερεύουσες συσκευές, για την ομαλή λειτουργία του συστήματος.
Αυτό θα γίνει με την χρήση ενός δικτύου επικοινωνίας μεταξύ των μικροεπεξεργαστών που ονομάζεται I^2C-Bus.

Επιπλέον, δεν θα μπορούσαμε να χαρακτηρίσουμε ως έξυπνο το σύστημα μας εάν δεν είχαμε την δυνατότητα να το ελέγχουμε μέσω του διαδικτύου. Για να το πετύχουμε αυτό καταρχήν θα συνδέσουμε την μητρική μας πλακέτα στο διαδίκτυο έτσι ώστε να μπορεί να δεχτεί εξωτερικές εντολές.

Κατόπιν θα χρησιμοποιήσουμε μία εφαρμογή για Android και IOs, την Blynk η οποία είναι σχεδιασμένη για να ελέγχει τα σήματα εξόδων από διάφορες συσκευές παρόμοιες με το Arduino.

Έτσι με τις ελεγχόμενες μεταβολές μέσω του διαδικτύου από την εφαρμογή Blynk στις εξόδους της μητρικής πλακέτας και με το κατάλληλα σχεδιασμένο πρόγραμμα θα μπορούμε τελικά να ελέγχουμε την μία (τον Master)! Αλλά εφόσον η μία έχει σχεδιαστεί να επικοινωνεί και να ελέγχει τις υπόλοιπες (τα Slaves) τελικά θα τις ελέγχουμε όλες.

Βασικές Λειτουργίες του συστήματος :

Αρχικά θα υπάρχον δύο βασικές λειτουργίες.

Η ομαλή κυκλική λειτουργία των φαναριών κατά την οποία θα λειτουργεί κανονικά σαν ένα κοινό σύστημα φωτεινών σηματοδοτών και δεύτερο το "sleeping mode" κατά το οποίο το σύστημα θα μπαίνει σε μια διαδικασία αναμονής για περεταίρω εντολές.

Έξυπνη λειτουργία του συστήματος – Λήψη αποφάσεων :

Τα κριτήρια με τα οποία ο αυτοματισμός μας θα παίρνει αποφάσεις και θα εκτελεί εντολές είναι τα εξής.

Αρχικά, το σύστημα μας έχει σχεδιαστεί για να **δίνει προτεραιότητα στα μέσα μαζικής μεταφοράς.** Άρα πρωταρχικό ρόλο στην διασταύρωση μας θα έχει το τραμ. Όταν θα "αισθάνεται" την παρουσία του τραμ στην διασταύρωση θα κάνει τις απαραίτητες ενέργειες – προετοιμασίες στα υπόλοιπα φανάρια έτσι ώστε πριν φτάσει το τραμ στην διασταύρωση να βρει τα φανάρια του ανοιχτά και να περάσει με ασφάλεια. Επιπλέον μετά την διέλευση του Τραμ θα επαναφέρει αυτόματα την λειτουργία των φαναριών των αυτοκινήτων.

Σε δεύτερο ρόλο έρχονται οι κεντρικές αρτηρίες, Πέραν του συνηθισμένου κύκλου λειτουργίας μίας διασταύρωσης το σύστημα μας με

την απομακρυσμένη βοήθεια ενός χειριστή θα προσπαθήσει για την καλύτερη δυνατή αποσυμφόρηση της κίνησης εάν αυτό κρίνεται απαραίτητο. Πολύ σημαντικό είναι το γεγονός ότι η χρήση του συστήματος μπορεί να γίνει απ τον οποιονδήποτε πολύ απλά χωρίς καμία ιδιαίτερη γνώση του προγραμματισμού πίσω από την κάθε λειτουργία .

Παρ ότι υπάρχουν πάνω από 12 πιθανοί συνδυασμοί λειτουργιών της διασταύρωσης με συνολικά 36 διαφορετικές φάσεις μεταξύ τους κάτι το οποίο μεταφράζεται σε ένα σύνολο περίπου 1260 εντολών, ο χειριστής απλά διαλέγει εάν θέλει να λειτουργεί η άνοδος και η κάθοδος, το δεξιά και αριστερά της διασταύρωσης ή αν έρχεται κάποιο από τα τραμ χωρίς να τον απασχολεί σε τι κατάσταση βρισκόντουσαν προηγουμένως τα φανάρια.

Επειδή το παράδειγμά μας αφόρα μία πολύ κεντρική οδό , δεν θα μπορούσε να δοθεί προτεραιότητα στους πεζούς σε κάθε φανάρι.

Παρ όλα αυτά επέλεξα την συγκεκριμένη διασταύρωσή διότι υπάρχει η δυνατότητα να γίνει και αυτό. Έτσι λοιπόν στην έξοδο της καθόδου της Λεωφ. Βασιλέως Κωνσταντίνου προς την είσοδο της Λεωφ. Βασιλίσσης Όλγας το ένα φανάρι των αυτοκινήτων θα μένει πάντα ανοιχτό εκτός εάν πατηθεί το κουμπί – αισθητήρας των πεζών ή περνά το τραμ και πρέπει ούτως η άλλως να διακοπεί η κυκλοφορία στον συγκεκριμένο δρόμο.

3.2 Arduino

3.2.1 Τι είναι το Arduino ;

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοιχτού κώδικα. Η ιδέα του Arduino γεννήθηκε στο Ινστιτούτο Ivrea σαν ένα ευκολόχρηστο εργαλείο για τον γρήγορο σχεδιασμό πρωτοτύπων. Από τότε μέχρι σήμερα η ιδέα του Arduino έχει μεγαλώσει πολύ.

Ολόκληρη η νοοτροπία του Arduino βασίζεται στην ελεύθερη γνώση. Τα σχέδια της κάθε πλατφόρμας καθώς επίσης η γλώσσα προγραμματισμού αλλά και το απαραίτητο λογισμικό IDE (Integrated Development Environment) για τον προγραμματισμό και την χρήση του διατίθενται δωρεάν στο διαδίκτυο. Αυτό σημαίνει ότι ο καθένας θα μπορούσε να φτιάξει μόνος του μια πλατφόρμα τύπου Arduino ή ακόμα και να την προσαρμόσει στις δικές του ανάγκες.

Όλες οι πλατφόρμες του Arduino αποτελούνται από έναν μικροεπεξεργαστή, αναλογικές και ψηφιακές εισόδους και εξόδους καθώς διάφορα άλλα ηλεκτρονικά στοιχεία απαραίτητα για την σύνδεση τους και τον προγραμματισμό τους από έναν οποιονδήποτε ηλεκτρονικό υπολογιστή , και την μετέπειτα τροφοδοσία του.
Απ εκεί και πέρα σε κάθε πλακέτα μπορεί να υπάρχουν διαφορές ανάλογα με τις ανάγκες του καθενός πχ ενσωματωμένο chip για σύνδεση στο internet.

Γενικότερα, θα μπορούσαμε να πούμε ότι το Arduino είναι ένα εργαλείο με το οποίο μπορούμε να αναπτύξουμε διάφορους αυτοματισμούς. Μας δίνει την δυνατότητα να παίρνουμε στοιχεία από το περιβάλλον μέσω των εισόδων του σε συνδυασμό με τους διάφορους αισθητήρες που θα μπορούσαμε να χρησιμοποιήσουμε και τελικά μέσω των εξόδων του να ελέγχουμε συσκευές του φυσικού κόσμου.

Απ εκεί και έπειτα οι δυνατότητες είναι απεριόριστες. Με τον κατάλληλο σχεδιασμό ,την αξιοποίηση των δυνατοτήτων της κάθε πλατφόρμας αλλά

και με τον συνδυασμό των ήδη υπαρχόντων ηλεκτρονικών στοιχείων μπορούμε να υλοποιήσουμε από απλά καθημερινά projects μέχρι πολύπλοκα επιστημονικά όργανα.

Πλεονεκτήματα της χρήσης του Arduino :

- **Οικονομικό.**
  Οι πλατφόρμες Arduino είναι σχετικά πολύ πιο οικονομικές από αντίστοιχες πλατφόρμες μικροεπεξεργαστών. Επίσης επειδή ο σχεδιασμός τους είναι ανοιχτός προς το κοινό μπορείς ακόμα και να την φτιάξεις μόνος σου.
- **Ευέλικτο.**
  Το Software (IDE) του Arduino μπορεί να χρησιμοποιηθεί σε λογισμικό Windows , Macintosh OSX και Linux σε αντίθεση με τα περισσότερα συστήματα Μίκρο-ελεγκτών τα οποία λειτουργούν μόνο σε Windows.
- **Απλό και εύκολο προγραμματιστικό περιβάλλον**
  Το Software (IDE) του Arduino είναι εύκολο στην χρήση ακόμα και από αρχάριους, αλλά παρ' όλα αυτά αρκετά εξελιγμένο για χρήση από έμπειρους χρήστες.
- **Ανοιχτό και Επεκτάσιμο Λογισμικό**
  Το λογισμικό αλλά και διάφορες έτοιμες βιβλιοθήκες κώδικα του Arduino υπάρχουν διαθέσιμες στο διαδίκτυο δωρεάν. Επιπλέον επειδή είναι σχεδιασμένο με βάση την γλώσσα προγραμματισμού C αυτό σημαίνει ότι μπορείς να προσθέσεις απευθείας κομμάτια κώδικα από C, C++ στο πρόγραμμα σου.
- **Διαθέσιμα σχέδια της πλακέτας**
  Είναι εξαιρετικά σπάνιο από μια εταιρία να διαθέτει τα σχέδια μιας πλακέτας της στο κοινό. Το γεγονός ότι υπάρχουν ελεύθερα τα σχέδια της κάθε πλακέτας Arduino είναι πολύ θετικό διότι έτσι μπορείς να την κατασκευάσεις μόνος σου με το ελάχιστο δυνατό κόστος, αλλά επίσης, εάν είσαι αρκετά εξοικειωμένος με τα ηλεκτρονικά ακόμα και να την βελτιώσεις με βάση τις δικές σου ανάγκες.

3.2.2 Κυκλώματα πλακετών Arduino

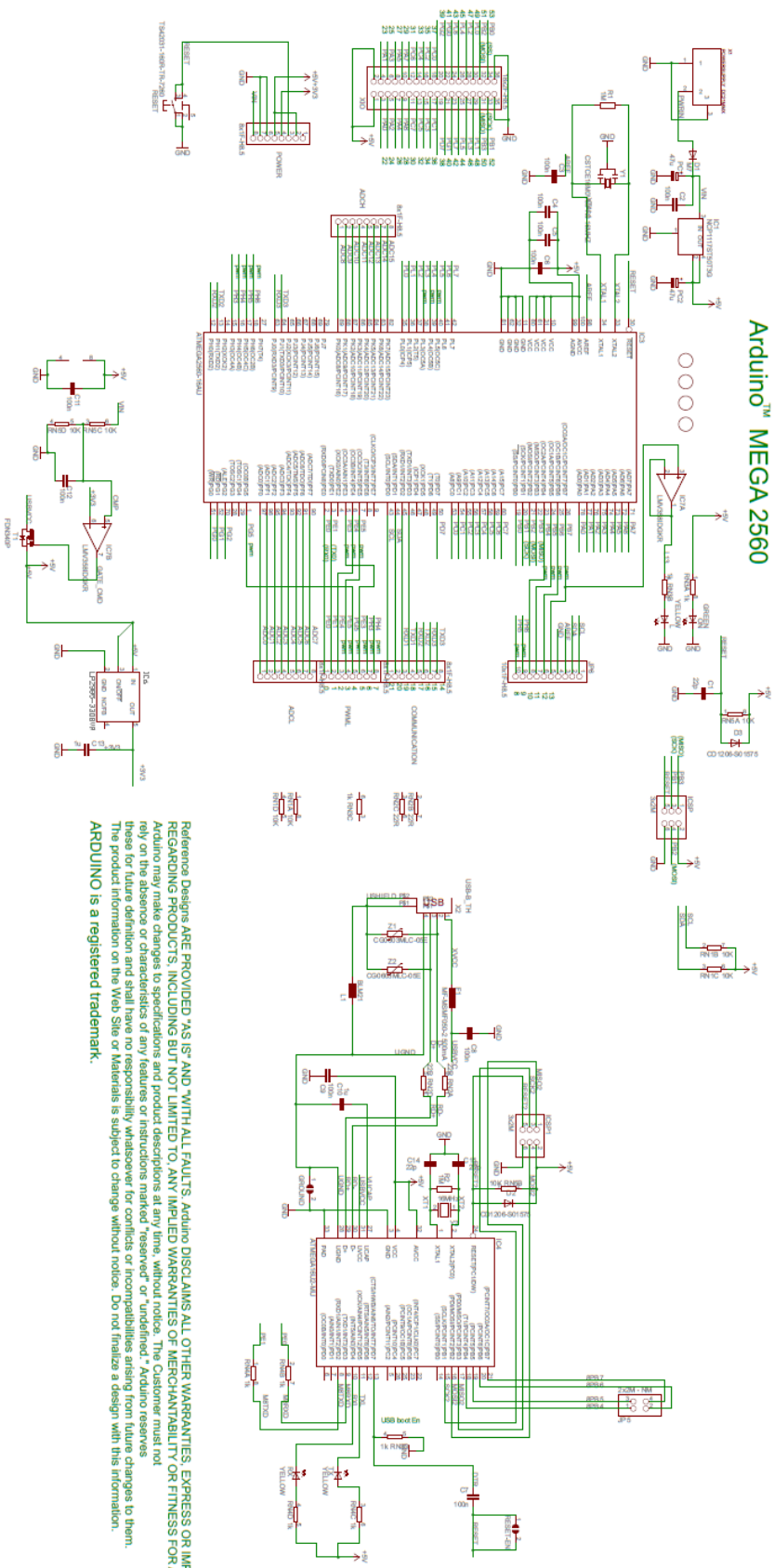
Το Arduino έχει διαδοθεί και εξελιχθεί πολύ τα τελευταία χρόνια, με αποτέλεσμα να κυκλοφορούν πλέον περισσότερες από 25 παραλλαγές του. Για τις ανάγκες της εφαρμογής μας θα χρησιμοποιήσουμε δύο τύπους της Πλατφόρμας Arduino.

Σαν "Master" θα χρησιμοποιήσουμε μια πλακέτα τύπου Arduino Mega 2560 , και σαν "Slaves" θα χρησιμοποιήσουμε πλακέτες τύπου Arduino Uno.

Βασικά Χαρακτηριστικά και ηλεκτρονικό κύκλωμα , Arduino Mega 2560 :

Μίκρο-ελεγκτής	ATmega2560
Ψηφιακές Είσοδοι - Έξοδοι	54
Αναλογικές Είσοδοι - Έξοδοι	16
Σειριακές Πύλες	4
Τάση Λειτουργίας	5 V
Τάση Εισόδου	7 – 12 V
Όρια Τάσης Εισόδου	6 – 20 V
Flash Memory	256 KB
SRAM	8 KB
Συνεχές Ρεύμα ανά Pin	20mA
Θύρα USB	
Τροφοδοσία μέσω Jack	
Reset Button	

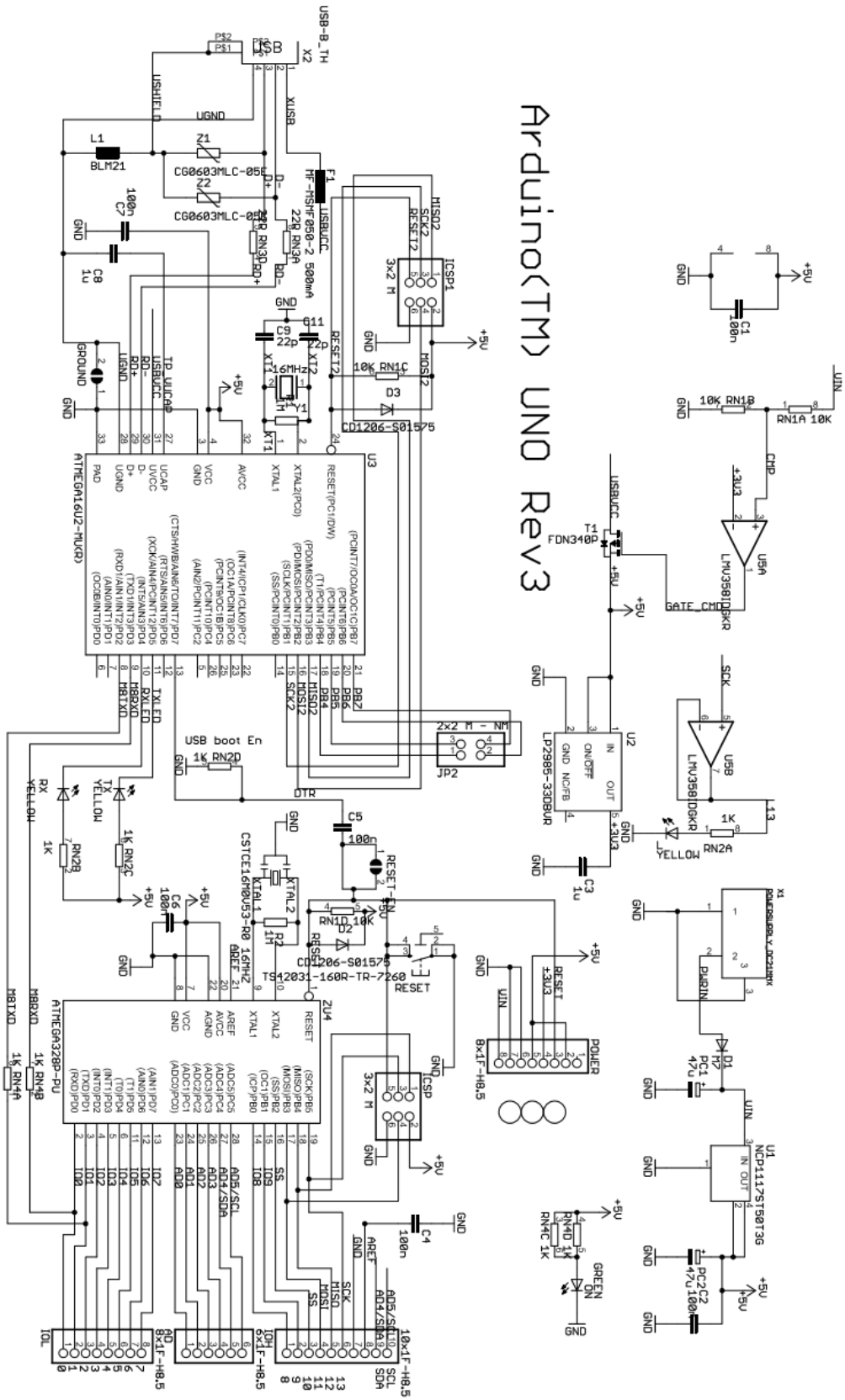
3. To Project


Arduino™ MEGA 2560
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
ARDUINO is a registered trademark.

Βασικά Χαρακτηριστικά και ηλεκτρονικό κύκλωμα Arduino UNO

Μίκρο-ελεγκτής	ATmega328P
Ψηφιακές Είσοδοι - Έξοδοι	13
Αναλογικές Είσοδοι - Έξοδοι	6
Σειριακές Πύλες	2
Τάση Λειτουργίας	5 V
Τάση Εισόδου	7 – 12 V
Όρια Τάσης Εισόδου	6 – 20 V
Flash Memory	32 KB
SRAM	2 KB
Συνεχές Ρεύμα ανά Pin	20mA
Θύρα USB	
Τροφοδοσία μέσω Jack	
Reset Button	

3. To Project


Arduino(TM) UNO Rev3
ATMEGA16U2-MUR
ATMEGA328P-PU
USB-B_TH
USB
ICSP1
ICSP
POWER
JP2
USB boot En
RESET-EN
RESET
TS12031-160R-TR-7260
CD1206-S01575
FDN340P
LP2985-33DBVR
LMV358IDGKR
NCP1117ST50T3G
GATE_CMD
YELLOW
GREEN
ON
RX
TX
MBRXD
MBTXD
RXLED
TXLED
DTR
VIN
+5V
+3V3
GND

3.2.3 Γλώσσα Προγραμματισμού και Εντολές του Arduino

Η γλώσσα προγραμματισμού του Arduino έχει βασιστεί στην *Wiring*. Η *Wiring* είναι ένα ελεύθερο πλαίσιο προγραμματισμού για Μίκρο-ελεγκτές. Βασίζεται στην *C/C ++*, δημιουργήθηκε το 2003 απ τον *Hernando Barragán* και συνεχίζει να εξελίσσεται ακόμη και σήμερα μέσα από τις συνεισφορές, την εμπειρία και την συλλογική γνώση των χρηστών της. Βασικός της σκοπός ήταν και είναι να καλύψει την ανάγκη των γρήγορων ηλεκτρονικών πειραμάτων με τον συνδυασμό μερικών σειρών κώδικα και λίγων ηλεκτρονικών στοιχείων.

Ένα στοιχείο που την κάνει να ξεχωρίζει απ τις υπόλοιπες γλώσσες προγραμματισμού είναι ότι μπορεί να χρησιμοποιηθεί σε διάφορα λογισμικά συστήματα (GNU/Linux , Macintosh OSX) και όχι μόνο σε Windows. Επιπλέον συνδυάζεται με την βιβλιοθήκη AVR Libc και επιτρέπει την χρήση κάθε μίας από τις λειτουργίες της.

Για να γράψουμε όμως το πρόγραμμα μας, χρησιμοποιώντας τις εντολές που μάθαμε μέσα από την *wiring* και να το ανεβάσουμε στην πλακέτα μας, θα χρειαστούμε το *Arduino Software (IDE)*. Το λογισμικό (IDE) (Integrated Development Environment) του Arduino έχει γραφτεί σε *Java* και έχει βασιστεί στην *Processing*.

Ας δούμε αρχικά μέσα από μερικά απλά βήματα, πώς μπορούμε να κατεβάσουμε και να εγκαταστήσουμε το Arduino Software IDE.

Βήμα 1°

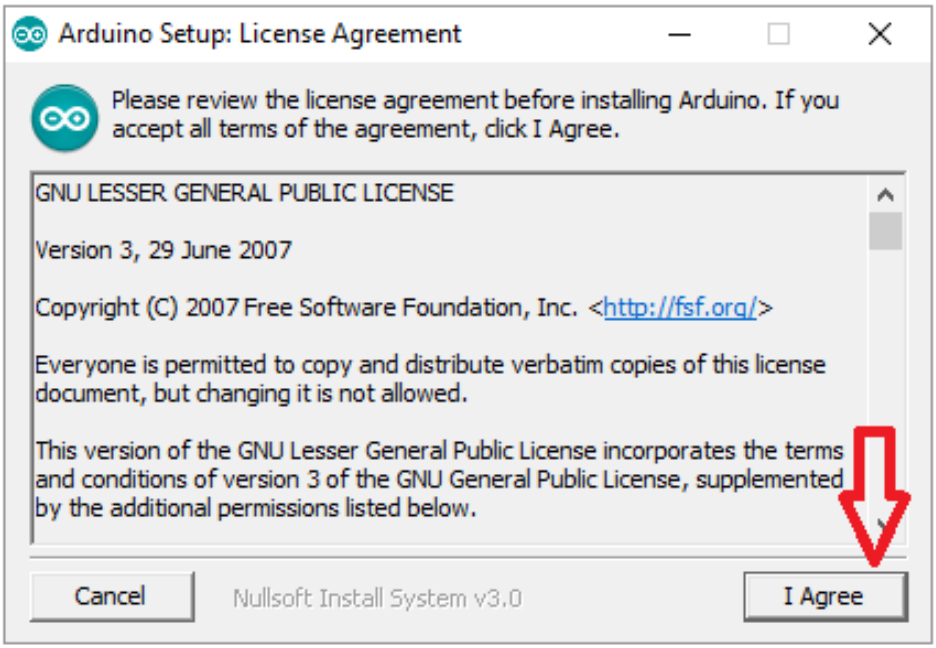
Για να κατεβάσουμε το λογισμικό το μόνο που χρειάζεται να κάνουμε είναι να επισκεφτούμε την ιστοσελίδα του arduino και να επιλέξουμε την καρτέλα *SOFTWARE.*

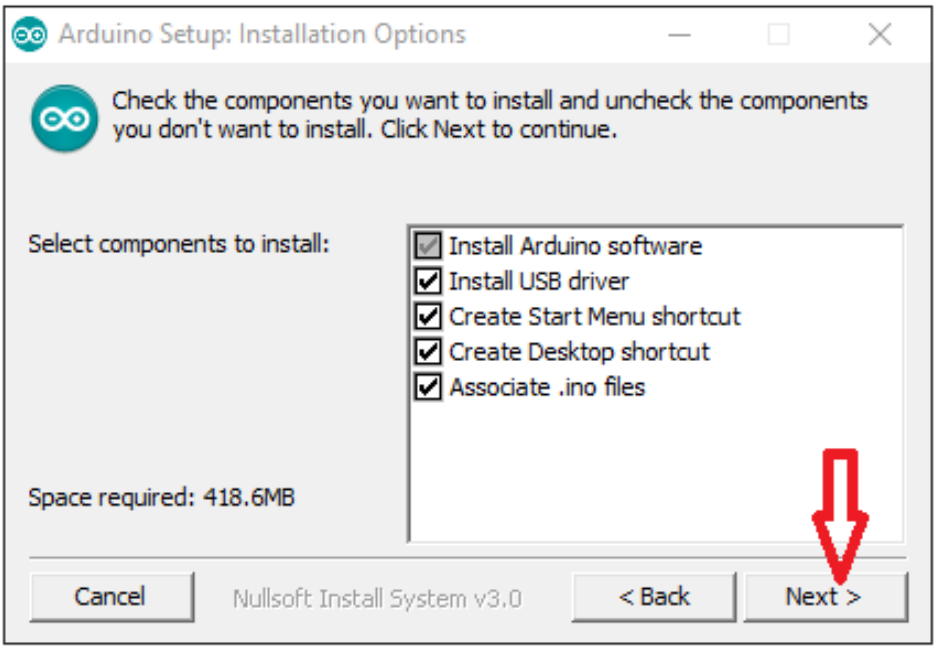
https://www.arduino.cc/en/Main/Software

Κατόπιν λίγο πιο κάτω στην ίδια σελίδα μπορούμε να διαλέξουμε την κατάλληλη έκδοση για το λογισμικό που χρησιμοποιούμε.

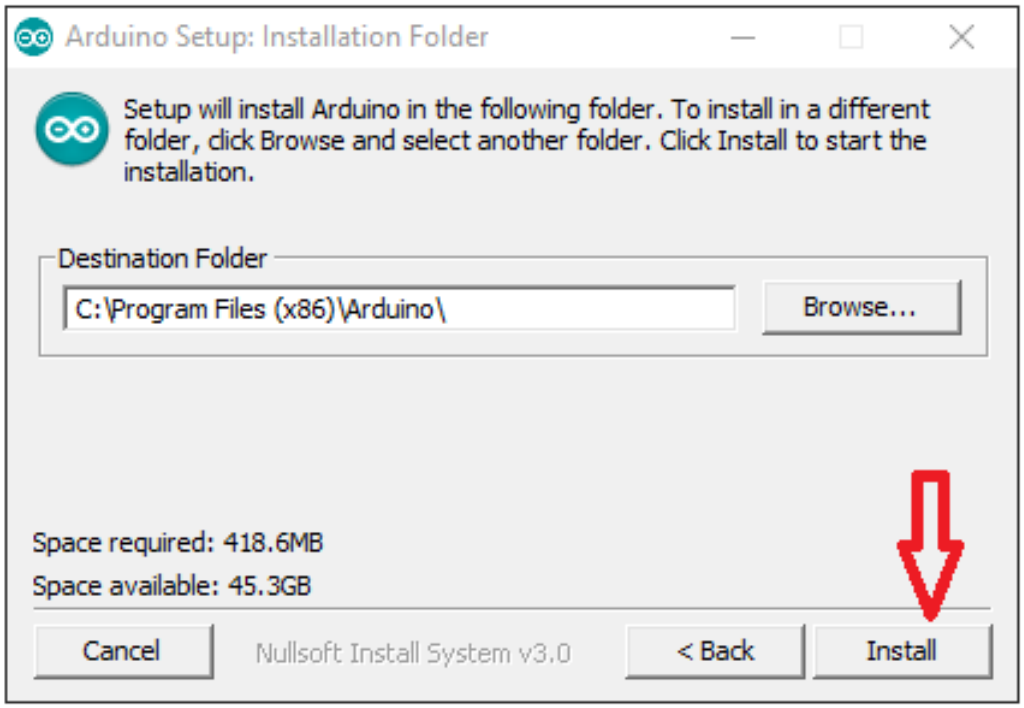
Βήμα 2ο

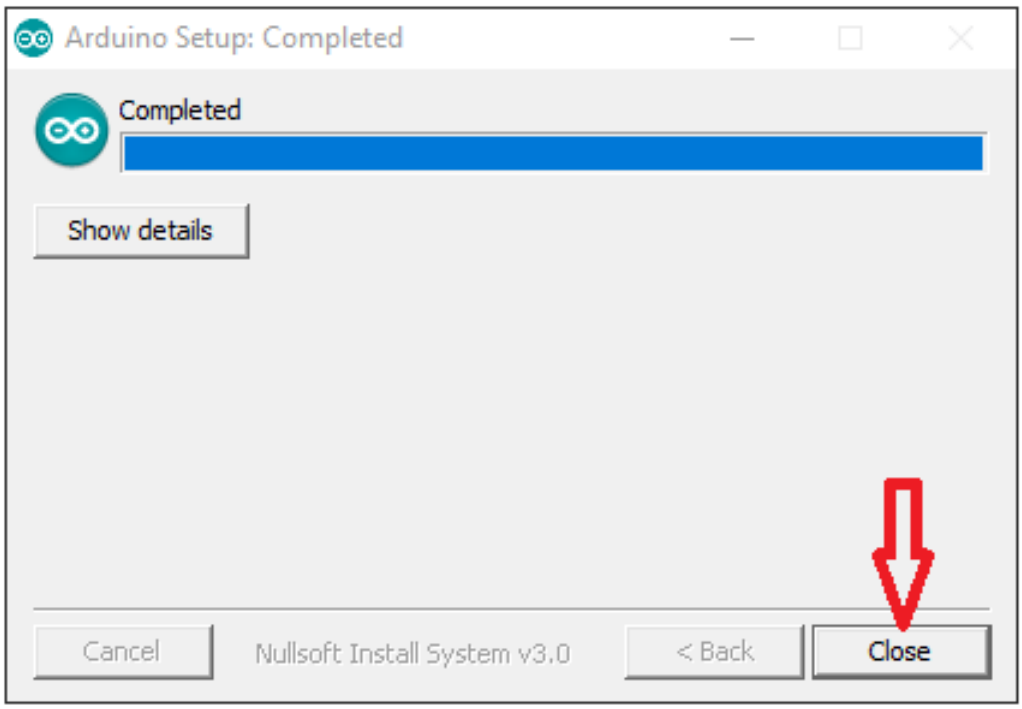
Ανοίγουμε το αρχείο που κατεβάσαμε και ακολουθούμε τα παρακάτω βήματα :





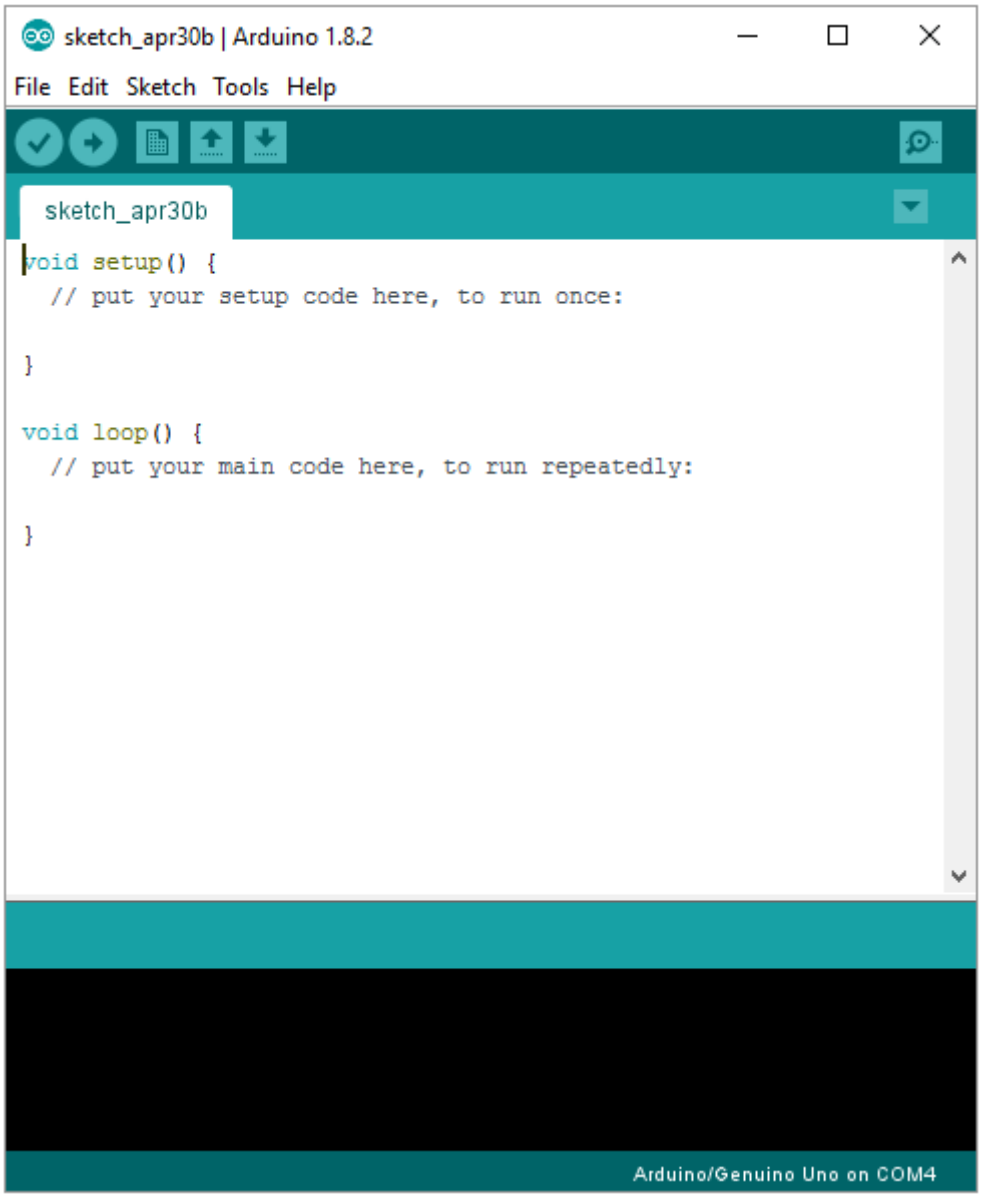
3. To Project


Arduino Setup: Installation Folder
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files (x86)\Arduino\
Browse...
Space required: 418.6MB
Space available: 45.3GB
Cancel
Nullsoft Install System v3.0
< Back
Install


Arduino Setup: Completed
Completed
Show details
Cancel
Nullsoft Install System v3.0
< Back
Close

Βήμα 3°

Στην επιφάνεια εργασίας μας θα έχει δημιουργηθεί μια συντόμευση για το Arduino software IDE κάνουμε διπλό κλικ και το παρακάτω παράθυρο θα μας εμφανιστεί .



Θα δούμε ότι μέσα στο πρόγραμμα έχουμε 5 γρήγορα κουμπιά.

1. (Κόκκινος κύκλος) *Verify* , Επιβεβαιώνει ότι οτιδήποτε έχουμε γράψει έως τώρα είναι συντακτικά σωστά και ότι στέκει σαν πρόγραμμα, εάν υπάρχει κάποιο λάθος θα μας το ορίσει στο κάτω μέρος του παραθύρου μας.

2. (Κίτρινος κύκλος) *Upload* , Με αυτό το κουμπί θα ανεβάσουμε το πρόγραμμα μας στην επιλεγμένη πλακέτα

3. (Μαύρο τετράγωνο) *New* , Ανοίγει νέο παράθυρο για την συγγραφή νέου σχεδίου.

4. (Πορτοκαλί τετράγωνο) *Open* , το χρησιμοποιούμε για να εισάγουμε κάποιο προϋπάρχον σχέδιο.

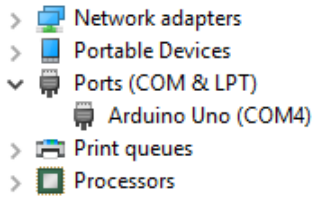
5. (Μωβ τετράγωνο) *Save* ,το χρησιμοποιούμε για να σώσουμε το σχέδιο μας.

Βήμα 4°

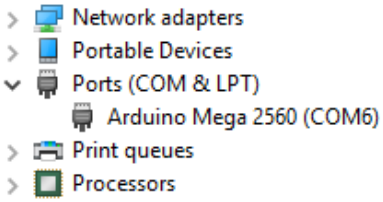
Πλέον είμαστε έτοιμοι να αρχίσουμε να γράφουμε τον κώδικα μας . Δεν είμαστε όμως έτοιμοι να ανεβάσουμε τον κώδικα μας στην πλακέτα μας. Θα πρέπει να δημιουργήσουμε μια σύνδεση μεταξύ τους πρώτα.

Για να δημιουργήσουμε μια σύνδεση μεταξύ του υπολογιστή και της πλακέτας μας μένει ένα τελευταίο βήμα. Από τον υπολογιστή μας πρέπει να ορίσουμε μια πύλη-δίαυλο επικοινωνίας με την πλακέτα μας. (*COM port*).

Για να το κάνουμε αυτό εύκολα και γρήγορα από την αναζήτηση των Windows βρίσουμε την Διαχείριση Συσκευών *(Device Manager)* και έχοντας συνδεδεμένη την πλακέτα βρίσκουμε την υποκατηγορία *Ports* (*COM & LPT*). Λογικά το σύστημα μας θα έχει ορίσει από μόνο του μια ελεύθερη πύλη , εάν όχι μπορούμε να το κάνουμε εμείς επί τόπου. Βλέπουμε πια πύλη έχει οριστεί τελικά :



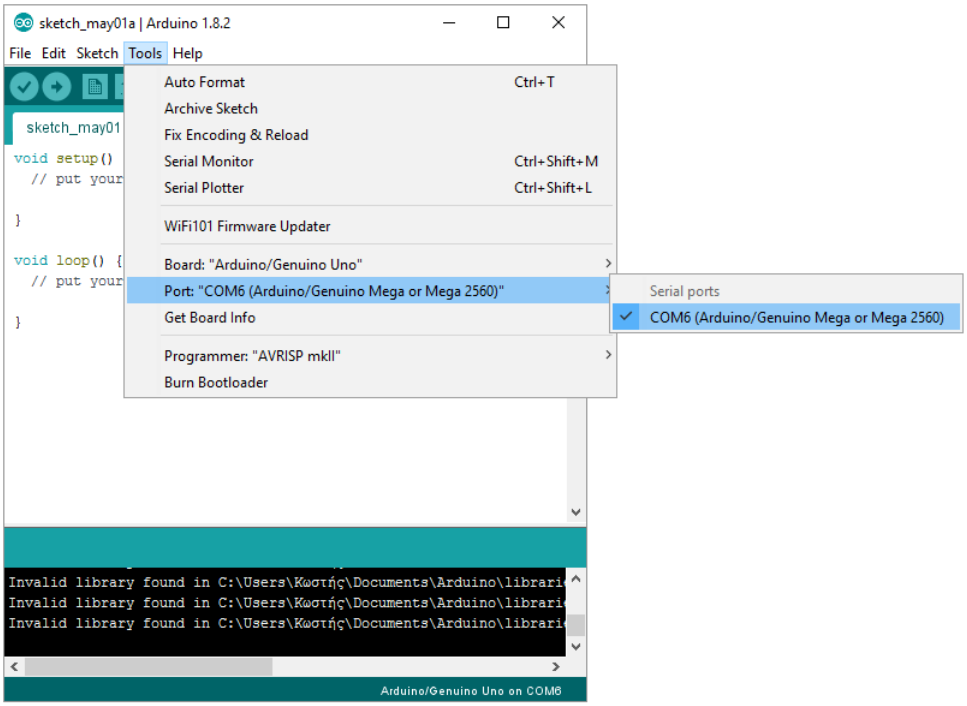
Στο παράδειγμα μας η πύλη *COM4* έχει οριστεί για το *Arduino UNO*. ***ΠΡΟΣΟΧΗ εάν συνδέσουμε διαφορετικές πλακέτες στο υπολογιστή μας, πχ μια *Arduino MEGA 2560* θα χρειαστεί να ξανακάνουμε την παραπάνω διαδικασία καθώς η κάθε πλακέτα θα χρησιμοποιεί την δική της πύλη *(COM)*



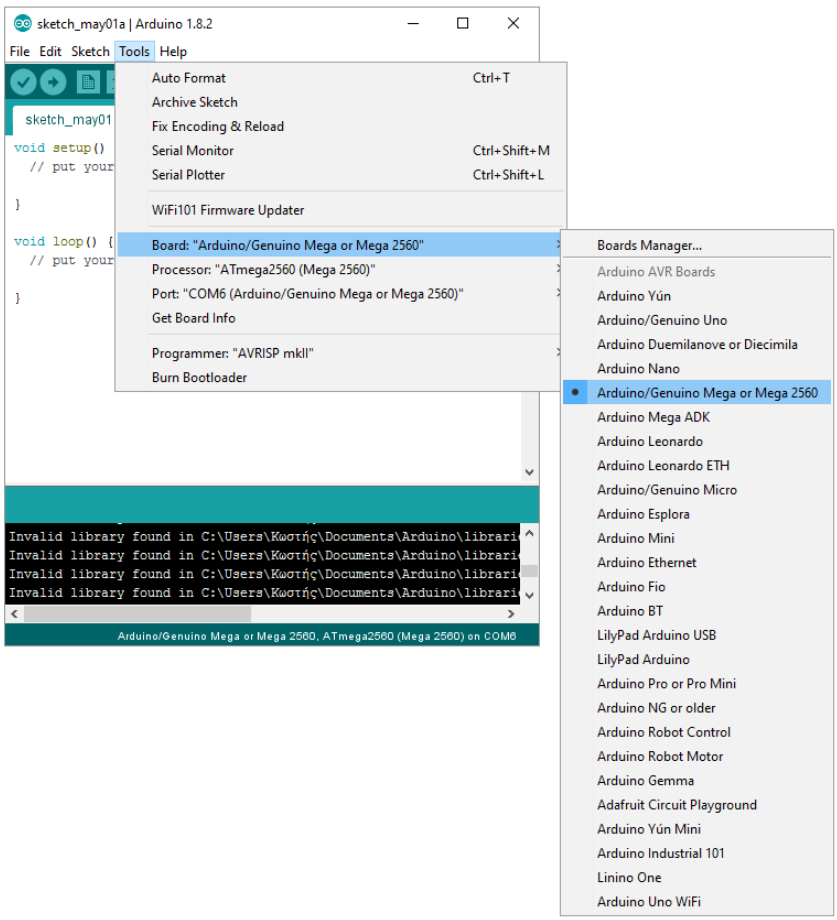
Αφού ξέρουμε πλέον ποιά πύλη χρησιμοποιεί η πλακέτα μας για να επικοινωνεί με τον υπολογιστή μας, μένει να ανοίξουμε το *Arduino Software IDE* και μέσα και από αυτό πλέον να ορίσουμε την πύλη μας ανάλογα με τν πλακέτα που θα χρησιμοποιήσουμε.

Από την καρτέλα *Tools* , πάμε στο *Port* και από εκεί επιλέγουμε την πύλη που αντιστοιχεί στην πλακέτα μας όπως είδαμε παραπάνω

3. Το Project



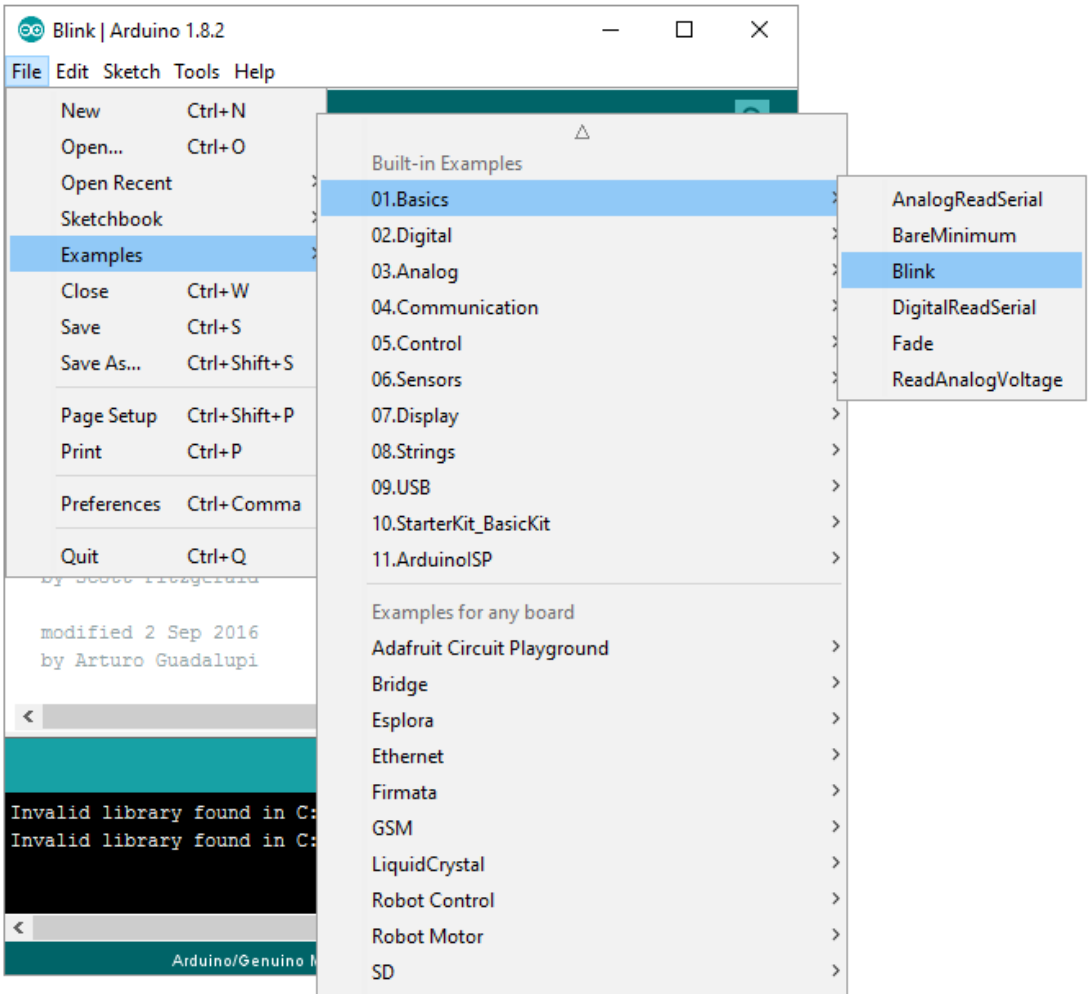
Επίσης θα πρέπει να ορίσουμε στο πρόγραμμα ποία πλακέτα θα χρησιμοποιήσουμε. Ξανά λοιπόν απ την καρτέλα *Tools* , πάμε στο *Board* και από εκεί επιλέγουμε την πλακέτα που θα χρησιμοποιήσουμε.



Βήμα 5ο

Για να επιβεβαιώσουμε ότι η εγκατάσταση του προγράμματος μας κύλησε ομαλά και ότι ο υπολογιστής μας βρίσκεται σε επικοινωνία με την πλακέτα μας , καλό Θα ήταν να τρέξουμε ένα από τα έτοιμα δοκιμαστικά έτοιμα *Scripts* που έχει το *Arduino*.

Απ την καρτέλα *File -> Examples -> 01. Basics ->* επιλέγουμε το *Blink*. Το *Blink* είναι ένα δοκιμαστικό πρόγραμμα το οποίο ανάβει και σβήνει ανά 1 sec ένα ενσωματωμένο *LED* στην πλακέτα μας.



Αφού το έχουμε ανεβάσει και είδαμε ότι όλα λειτουργούν όπως πρέπει, είμαστε πλέον έτοιμοι όχι μόνο να γράψουμε τον δικό μας κώδικα αλλά να τον ανεβάσουμε στην πλακέτα μας και να κάνουμε τις δοκιμές μας.

Εφόσον είμαστε πλέον συνδεδεμένοι και έτοιμοι ας πάμε να δούμε τις εντολές ή αλλιώς την γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε κατά την συγγραφή ενός προγράμματος στο Arduino.

Δομικά Στοιχεία Προγράμματος

setup()	Εντολή ορισμού αρχικών τιμών
loop()	Εντολή έναρξης επαναλαμβανόμενου βρόγχου
void()	Εντολή εκκίνησης υπό-ρουτίνας

Δομές ελέγχου ροής

if	Δομή ελέγχου μίας συνθήκης
if ... else	Δομή ελέγχου πολλαπλών συνθηκών
for	Δομή επαναληπτικού ελέγχου συνθήκης
while	Δομή επαναληπτικού ελέγχου συνθήκης
do ... while	Δομή επαναληπτικού ελέγχου συνθήκης
switch ... case	Δομή ελέγχου περιπτώσεων
break	Εντολή διακοπής μιας επαναληπτικής δομής
continue	Εντολή παράλειψης της τρέχουσας επανάληψης
return	Εντολή επιστροφής από μία συνάρτηση
goto	Εντολή μετάβασης σε κάποιο σημείο του κώδικα

Επιπλέον Συντακτικά Στοιχεία

;	Ολοκλήρωση μιας δήλωσης
{}	Αρχή και τέλος μιας δήλωσης
//	Σχόλιο μίας σειράς
/**/	Σχόλιο πολλών σειρών
#define	Ορίζει ένα όνομα σε μια σταθερή τιμή
#include	Συμπεριλαμβάνει εξωτερικές βιβλιοθήκες

Αριθμητικοί τελεστές

=	Τελεστής εκχώρησης
+	Τελεστής πρόσθεσης
-	Τελεστής αφαίρεσης
*	Τελεστής πολλαπλασιασμού
/	Τελεστής διαίρεσης
%	Τελεστής υπόλοιπου ακεραίας διαίρεσης

Λογικοί τελεστές

&&	Τελεστής εκχώρησης
||	Τελεστής πρόσθεσης
!	Τελεστής αφαίρεσης

Δυαδικοί τελεστές

&	Δυαδική σύζευξη
|	Δυαδική διάζευξη
^	Δυαδική αποκλειστική διάζευξη
~	Δυαδική άρνηση
<<	Δυαδική αριστερή ολίσθηση
>>	Δυαδική δεξιά ολίσθηση

Τελεστές Σύνθετου χειρισμού

++	Αύξηση κατά μία ακέραιη μονάδα
--	Μείωση κατά μία ακέραιη μονάδα
+=	Σύνθετη αφαίρεση
-=	Σύνθετη μείωση
*=	Σύνθετός πολλαπλασιασμός
/=	Σύνθετη διαίρεση
%=	Σύνθετη ακέραια διαίρεση
&=	Σύνθετη δυαδική σύζευξη

Σταθερές

HIGH	Τιμή υψηλής στάθμης
LOW	Τιμή χαμηλής στάθμης
false	Λογικό επίπεδο ψεύδους σε μία συνθήκη)
true	Λογικό επίπεδο αλήθειας σε μία συνθήκη)
INPUT	Ορισμός μίας επαφής ως είσοδο
OUTPUT	Ορισμός μίας επαφής ως έξοδο
LED_BUILTIN	Ενσωματωμένο LED στην αναλογική θέση 13

Τύποι Δεδομένων

boolean	Λογική δυαδική τιμή
char	Προσημασμένος χαρακτήρας 8 ψηφίων
unsigned char	Μη προσημασμένος χαρακτήρας 8 ψηφίων
byte	Μη προσημασμένος χαρακτήρας 8 ψηφίων
int	Προσημασμένος ακέραιος αριθμός 16 ψηφίων
unsigned int	Μη προσημασμένος ακέραιος αριθμός 16 ψηφίων
word	Μη προσημασμένος ακέραιος αριθμός 16 ψηφίων
long	προσημασμένος ακέραιος αριθμός 32 ψηφίων
unsigned long	Μη προσημασμένος ακέραιος αριθμός 32 ψηφίων
float	Αριθμός κινητής υποδιαστολής απλής ακρίβειας
String	Αντικείμενο αλφαριθμητικού με χρήσιμες μεθόδους

Συναρτήσεις μετατροπής τύπων

char()	Μετατρέπει μια τιμή σε τύπο δεδομένων char
byte()	Μετατρέπει μια τιμή σε τύπο δεδομένων byte
int()	Μετατρέπει μια τιμή σε τύπο δεδομένων int
word()	Μετατρέπει μια τιμή σε τύπο δεδομένων word
long()	Μετατρέπει μια τιμή σε τύπο δεδομένων long
float()	Μετατρέπει μια τιμή σε τύπο δεδομένων float

Συναρτήσεις εισόδου και εξόδου

pinMode()	Ορίζει μια επαφή ως είσοδο ή έξοδο

Συναρτήσεις ψηφιακής εισόδου και εξόδου

digitalWrite()	Γράφει σε μια ψηφιακή επαφή
digitalRead()	Διαβάζει μια ψηφιακή επαφή

Συναρτήσεις αναλογικής εισόδου και εξόδου

analogReference()	Ορίζει την τάση αναλογικής αναφοράς
analogWrite()	Γράφει σε μια αναλογική επαφή
analogRead()	Διαβάζει μια αναλογική επαφή

Προηγημένες συναρτήσεις εισόδου και εξόδου

tone()	Παράγει ένα τετραγωνικό σήμα ορισμένης συχνότητας
noTone()	Διακόπτει την παραγωγή τετραγωνικών σημάτων
shiftOut()	Ολισθαίνει τα ψηφία μιας τιμής σε μία επαφή εξόδου
pulseIn()	Επιστρέφει την διάρκεια σε μs ενός παλμού HIGH ή LOW

Συναρτήσεις αναλογικής εισόδου και εξόδου

millis()	Διάρκεια εκτέλεσης του προγράμματος σε ms
micros()	Διάρκεια εκτέλεσης του προγράμματος σε μs
delay()	Παύση προγράμματος - η διάρκεια δίδεται σε ms
delayMicroseconds()	Παύση προγράμματος - η διάρκεια δίδεται σε μs

Μαθηματικές κ Τριγωνομετρικές Συναρτήσεις

max()	Βρίσκει τον μεγαλύτερο ανάμεσα σε δύο αριθμούς
min()	Βρίσκει τον μικρότερο ανάμεσα σε δύο αριθμούς
abs()	Επιστρέφει την απόλυτη τιμή ενός αριθμού
constrain()	Ελέγχει για υπερχείλιση ή υποχείλιση ορίων
map()	Πραγματοποιεί γραμμικό μετασχηματισμό ορίων
pow()	Επιστρέφει το αποτέλεσμα μίας δύναμης
sqrt()	Επιστρέφει την ρίζα ενός αριθμού
sin()	Υπολογίζει το ημίτονο ενός αριθμού
cos()	Υπολογίζει το συνημίτονο ενός αριθμού
tan()	Υπολογίζει την εφαπτομένη ενός αριθμού

Συναρτήσεις γεννήτριας ψευδοτυχαίων αριθμών

random()	Δίδεται ένας νέος αριθμός από την γεννήτρια
randomSeed()	Θέτει τον σπόρο της γεννήτριας παραγωγής

Συναρτήσεις επεξεργασίας δυαδικών αριθμών

lowByte()	Επιστρέφει το δεξιότερο byte μίας μεταβλητής
highByte()	Επιστρέφει το αριστερότερο byte μίας μεταβλητής
bitRead()	Διαβάζει ένα συγκεκριμένο ψηφίο μίας μεταβλητής
bitWrite()	Γράφει σε ένα συγκεκριμένο ψηφίο μιας μεταβλητής
bitSet()	Γράφει την τιμή 1 σε κάποιο ψηφίο μίας μεταβλητής
bitClean()	Γράφει την τιμή 0 σε κάποιο ψηφίο μιας μεταβλητής
bit()	Υπολογίζει μία συγκεκριμένη δύναμη με βάση το 2

Συναρτήσεις χρήσης ρουτινών εξυπηρέτησης διακοπών

attachInterrupt()	Ενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής
detachInterrupt()	Απενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής

Συναρτήσεις ψηφιακής εισόδου και εξόδου

interrupts()	Ενεργοποιεί τα σήματα διακοπής
noIntrerrupts()	Απενεργοποιεί τα σήματα διακοπής

Υποστήριξη σειριακής επικοινωνίας

serial ()	Ενεργοποιεί την σειριακή επικοινωνία μεταξύ Η/Υ και πλακέτας

3.3 I^2c Bus

Inter-Integrated Circuit ή αλλιώς I2C ονομάζουμε έναν σειριακό δίαυλο μεταξύ πολλαπλών Υπολογιστών. Εφευρέθηκε και αναπτύχθηκε αρχικά από την Philips Semiconductors η οποία είναι γνωστή σήμερα ως NXP Semiconductors. Αργότερα έγινε γνωστό και ως TWI Two Wire Interface καθώς ονομάστηκε έτσι από την Atmel για την αποφυγή νομικών κυρώσεων για την χρήση του ονόματος.

Η τεχνολογία του I2C έχει πλέον εδραιωθεί στην παγκόσμια αγορά καθώς πάνω από 50 εταιρίες έχουν φτιάξει πάνω από 1000 προϊόντα συμβατά με αυτή. Μερικές από αυτές είναι η Siemens AG, η Intel mobile communications ,η NEC, η Texas Instruments, η STMicroelectronics, η Motorola η Intersil και άλλες.

Από τις 10 Οκτώβριου του 2006 και έπειτα η υλοποίηση μιας εφαρμογής η οποία χρησιμοποιεί το πρωτόκολλο του I2C είναι ελεύθερη για το κοινό και δεν απαιτείτε πλέον καμία ειδική άδεια.

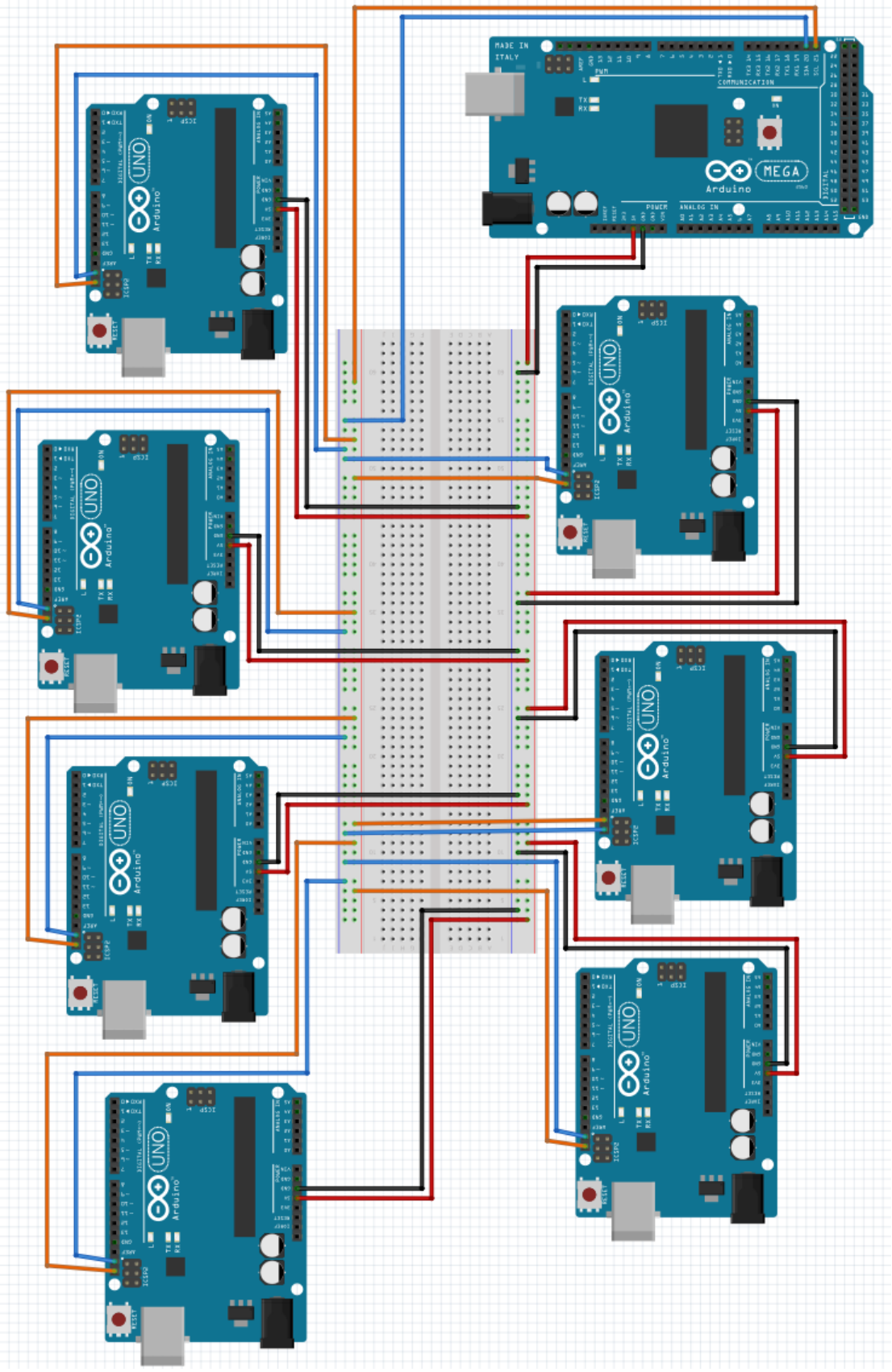
Είναι μια πολύ απλή μέθοδος πολλαπλής σειριακής επικοινωνίας διότι χρησιμοποιεί μόνο δύο αγωγούς. Αυτό είναι πολύ πρακτικό στην εξοικονόμηση των PINs του Μίκρο-επεξεργαστή το οποίο είναι βασικό πρόβλημα των μηχανικών ηλεκτρονικών. Ο ένας αγωγός ονομάζεται SDA (Serial Data Line) και ο άλλος SCL (Serial Clock Line).

Κάθε συνδεδεμένη συσκευή στον δίαυλο έχει μια μοναδική διεύθυνση μέσω της οποίας μπορούμε ανά πάσα στιγμή να την επικαλεστούμε σε κάποια ενέργεια. Βασικός τρόπος λειτουργίας του I2c είναι οι σχέσεις Master/Slaves με δύο διαφορετικά σενάρια όπως Master Transmitter(Writer)/Slave Receiver ή Master Receiver/Slave Sender.

Στο παράδειγμά μας θα χρησιμοποιήσουμε των Master ως Transmitter-Writer και τους επτά Slaves ως receivers.

Στο παρακάτω σχήμα θα δούμε πώς πρέπει να συνδέσουμε τα Arduino μας όσον αφορά την τροφοδοσία, την Γή και τα SDA, SCL Lines.

3. To Project



3.4 Blynk

3.4.1 Τι είναι το Blynk?

Το *Blynk* είναι μια πλατφόρμα με *iOS* και *Android* εφαρμογές σχεδιασμένη για τον έλεγχο πλακετών όπως το Arduino, το Raspberry Pi αλλά και όλων των υπόλοιπων παρόμοιων συσκευών.

Είναι ένα ψηφιακό ταμπλό στο οποίο μπορείς να κατασκευάσεις και να σχεδιάσεις τα γραφικά της εφαρμογής σου για τον έλεγχο της συσκευής σου. Είναι πολύ απλό στην εγκατάσταση αλλά και στην λειτουργία του είναι επίσης δωρεάν έως ένα βαθμό.

Το *Blynk* δεν περιορίζεται στην χρήση μίας μόνο συσκευής σαν το Arduino, αλλά μπορεί να ελέγξει όλες σχεδόν τις διαθέσιμες στην αγορά πλακέτες είτε μέσω *Wi Fi* , μέσω *Ethernet*, μέσω του νέου τσιπ ESP8266 κλπ. Αυτή την στιγμή υποστηρίζει πάνω από τετρακόσιες διαφορετικές συσκευές.

3.4.2 Πώς λειτουργεί?

Θα μπορούσαμε να χωρίσουμε το *Blynk* σε δύο κομμάτια.
Στο πρώτο μέρος του θα βάζαμε την εφαρμογή που θα χτίσουμε και θα χρησιμοποιήσουμε για να ελέγχουμε την πλακέτα μας , και στο δεύτερο μέρος την διαδικασία που θα πρέπει να ακολουθήσουμε για να συμπεριλάβουμε την βιβλιοθήκη του *Blynk* στο πρόγραμμα μας. Αλλά και για να ανοίξουμε ένα δίαυλο μεταξύ του ηλεκτρονικού Υπολογιστή και της πλακέτας έτσι ώστε να είναι συνδεδεμένη στο διαδίκτυο .

Για να κατεβάσουμε την εφαρμογή του *Blynk* σε οποιαδήποτε *Android* ή *iOS* συσκευή αρκεί να μπούμε στο *Play Store*. Μπορούμε να χρησιμοποιήσουμε όμως το Blynk και σε οποιονδήποτε ηλεκτρονικό υπολογιστή , μέσω του *Android* εξομοιωτή *Bluestacks*.

3.4.3 Εγκατάσταση Blynk – Σύνδεση με Arduino

Αρχικά πρέπει να εγκαταστήσουμε την εφαρμογή στην Android συσκευή μας. Μέσω του Play Store κάνοντας αναζήτηση το Blynk μπορούμε πολύ εύκολα να το βρούμε.

Κατόπιν με την ολοκλήρωση την εγκατάστασης , θα μας ζητηθεί να δημιουργήσουμε έναν λογαριασμό. Μπορούμε να το κάνουμε με δύο τρόπους είτε κάνουμε εγγραφή μέσω email είτε μέσω Facebook. Αφού θα μας γίνει μια γρήγορη – αυτόματη περιήγηση ως προς τον τρόπο λειτουργίας της εφαρμογής μπορούμε να ξεκινήσουμε να χτίσουμε το δικό μας Project.

Επιλέγουμε λοιπόν την δημιουργία ενός νέου project . Δίνουμε όνομα στο project μας και μέσα από μια λίστα διαλέγουμε την συσκευή την οποία θα χρησιμοποιήσουμε ή αλλιώς ελέγχουμε μέσω της εφαρμογής.

Έπειτα θα προσδιορίσουμε τον τρόπο σύνδεσης με τον οποίο θα συνδεθεί η συσκευή μας στο διαδίκτυο (Ethernet/usb/ wifi / Bluetooth /gsm / ble) , καθώς και το background της οθόνης μας μέσα στην εφαρμογή (άσπρο η μαύρο.)

Όταν ολοκληρώσουμε τις επιλογές μας ένα email θα αποσταλεί στο προσωπικό μας ηλεκτρονικό ταχυδρομείο, με τον μοναδικό κωδικό πρόσβασης μας στην εφαρμογή (Auth. Token) Κρατάμε πρόχειρο τον κωδικό αυτόν θα τον χρειαστούμε σύντομα!

Τώρα πλέον είμαστε σχεδόν έτοιμοι όσον αφορά την Android συσκευή μας και μπορούμε να προσθέσουμε διάφορα widgets (Controllers , Displays , Notifications κλπ.) και να χτίσουμε την εφαρμογή μας.

Εάν πχ επιλέξουμε να το τοποθετήσουμε ένα Button που θα χειρίζεται μια έξοδο του Arduino θα πρέπει και να ορίσουμε κάποια πράγματα.

Όπως :

- Το όνομα του button
- Το χρώμα του
- Το PIN OUTPUT (number) της συσκευής το οποίο θα ανοίγει-κλείνει.
- Εάν θα είναι διακόπτης on/off ή push button .
- Ποια θα είναι η αρχική θέση του διακόπτη (ανοιχτό η κλειστό) .
- Την ταμπέλα - αναγραφή στον διακόπτη και στις δύο καταστάσεις.

Αφού έχουμε ολοκληρώσει πλέον την δομή της εφαρμογής μας στην μένουν ακόμη 2 βήματα.

1) Να εγκαταστήσουμε την κατάλληλη βιβλιοθήκη (Blynk library) στο λογισμικό του Arduino
2) Να τρέξουμε το Arduino IDE και να επιβεβαιώσουμε ότι η βιβλιοθήκη μας προστέθηκε με επιτυχία.

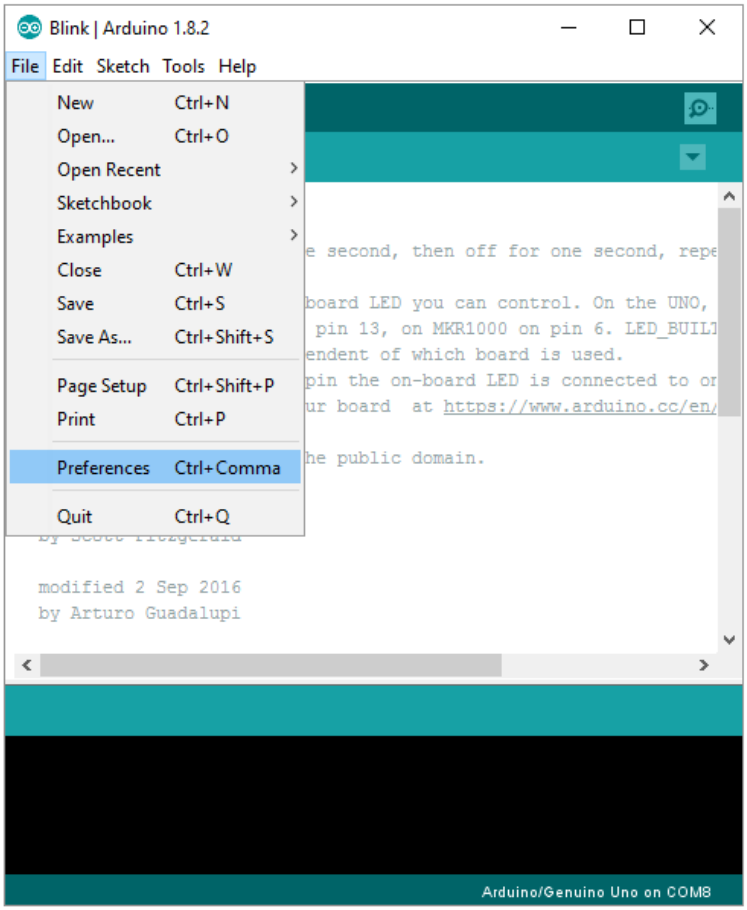
1) Εγκατάσταση της βιβλιοθήκης (Blynk library) στο λογισμικό του Arduino.

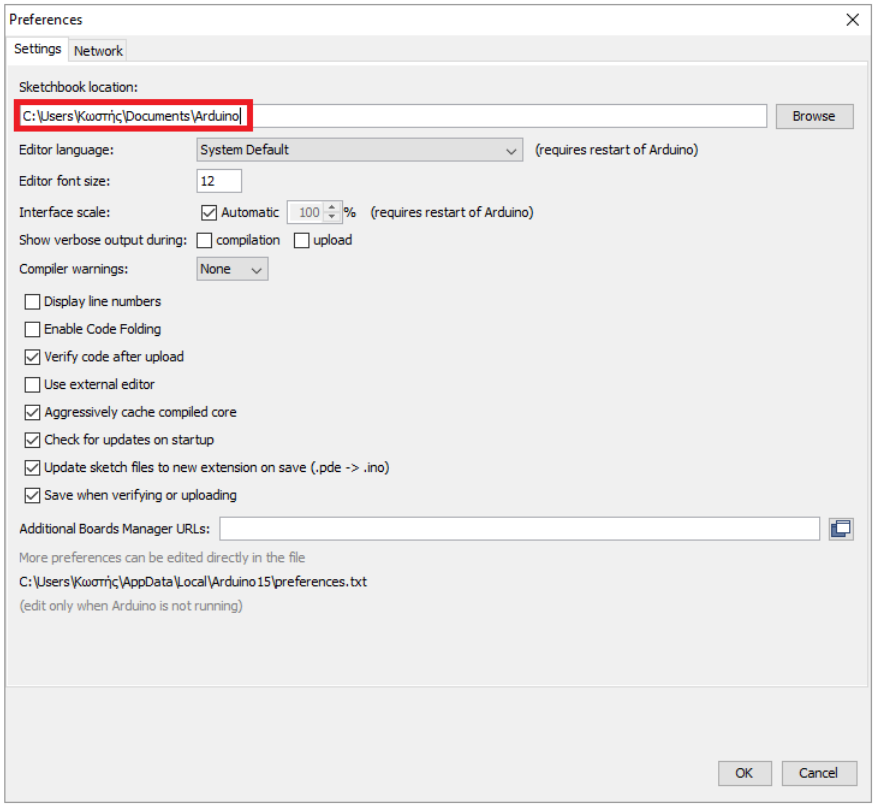
Η διαδικασία που πρέπει να ακολουθήσουμε είναι η εξής :

Αρχικά θα πρέπει να επισκεφτούμε την ιστοσελίδα των δημιουργών του Blynk για να κατεβάσουμε την βιβλιοθήκη (http://www.blynk.cc/getting-started/) .

Αφού κατεβάσουμε το τελευταίο και πιο ενημερωμένο – πρόσφατο zip αρχείο θα πρέπει να το κάνουμε Unzip .
Θα δούμε ότι το αρχείο αυτό περιέχει πάνω από μια βιβλιοθήκη και φακέλους. Θα τα αντιγράψουμε όλα και να τα τοποθετήσουμε στον φάκελο που αποθηκεύει τις βιβλιοθήκες του το Arduino IDE .

Για να βρούμε εύκολα το φάκελο στον οποίο αποθηκεύει το Arduino της βιβλιοθήκες του στην δική μας συσκευή αρκεί να ανοίξουμε το Arduino IDE File -> Preferences όπως φαίνεται στην παρακάτω εικόνα.





Περιμένουμε να βρούμε ένα τέτοιο αποτέλεσμα :

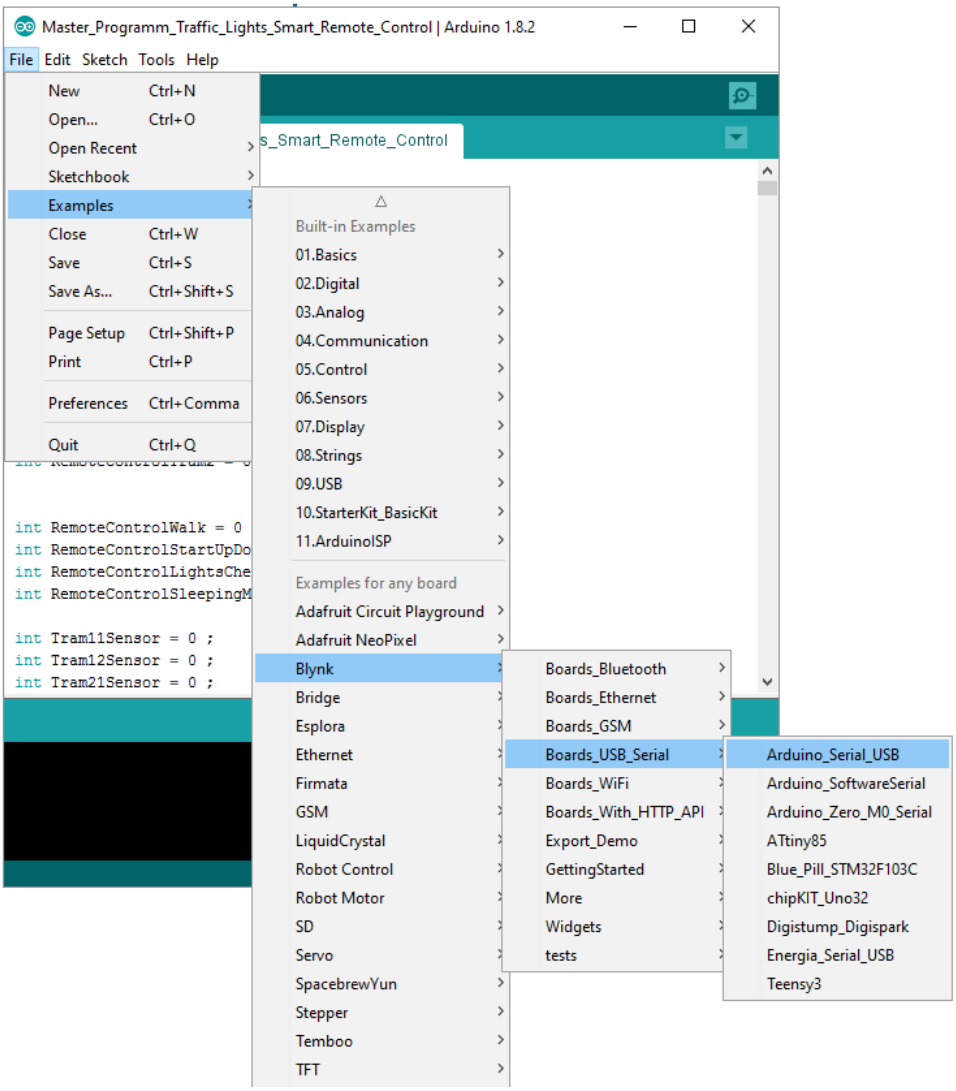
C:\Program Files (x86)\Arduino\libraries

Πηγαίνουμε λοιπόν στην συγκεκριμένη τοποθεσία και κάνουμε επικόλληση τα αρχεία που είχαμε αντιγράψει.

ΠΡΟΣΟΧΗ!!! Οι βιβλιοθήκες στις βιβλιοθήκες (libraries) και τα εργαλεία στα εργαλεία (Tools) !

2) Επιβεβαίωση Σωστής Εγκατάστασης

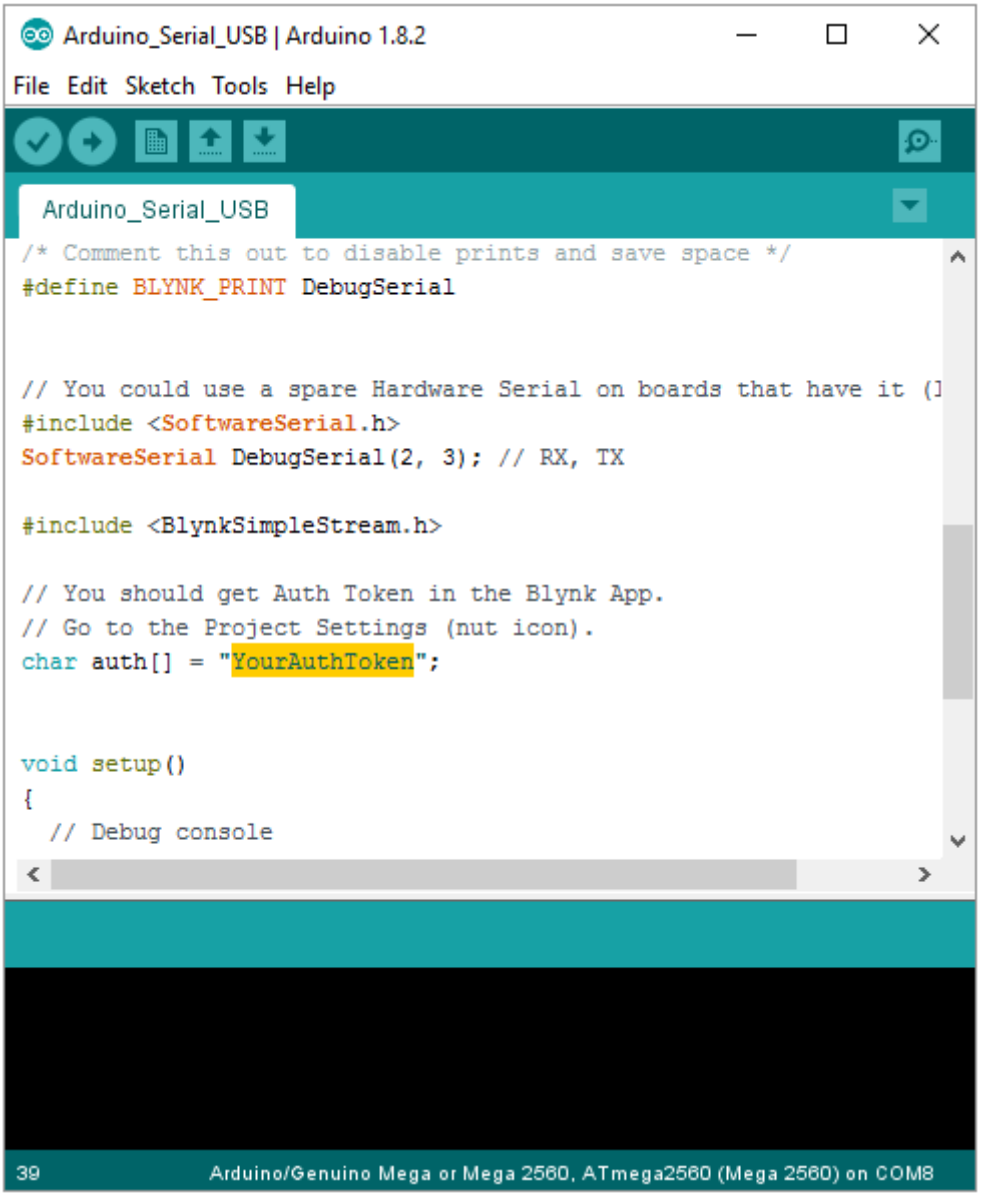
Καλό θα ήταν στην πρώτη μας προσπάθεια να δοκιμάσουμε με ένα έτοιμο παράδειγμα. Ανοίγουμε λοιπόν το Software του Arduino και πάμε στο : File -> examples -> BLYNK -> BOARDS USB SERIAL -> ARDUINO_SERIAL_USB



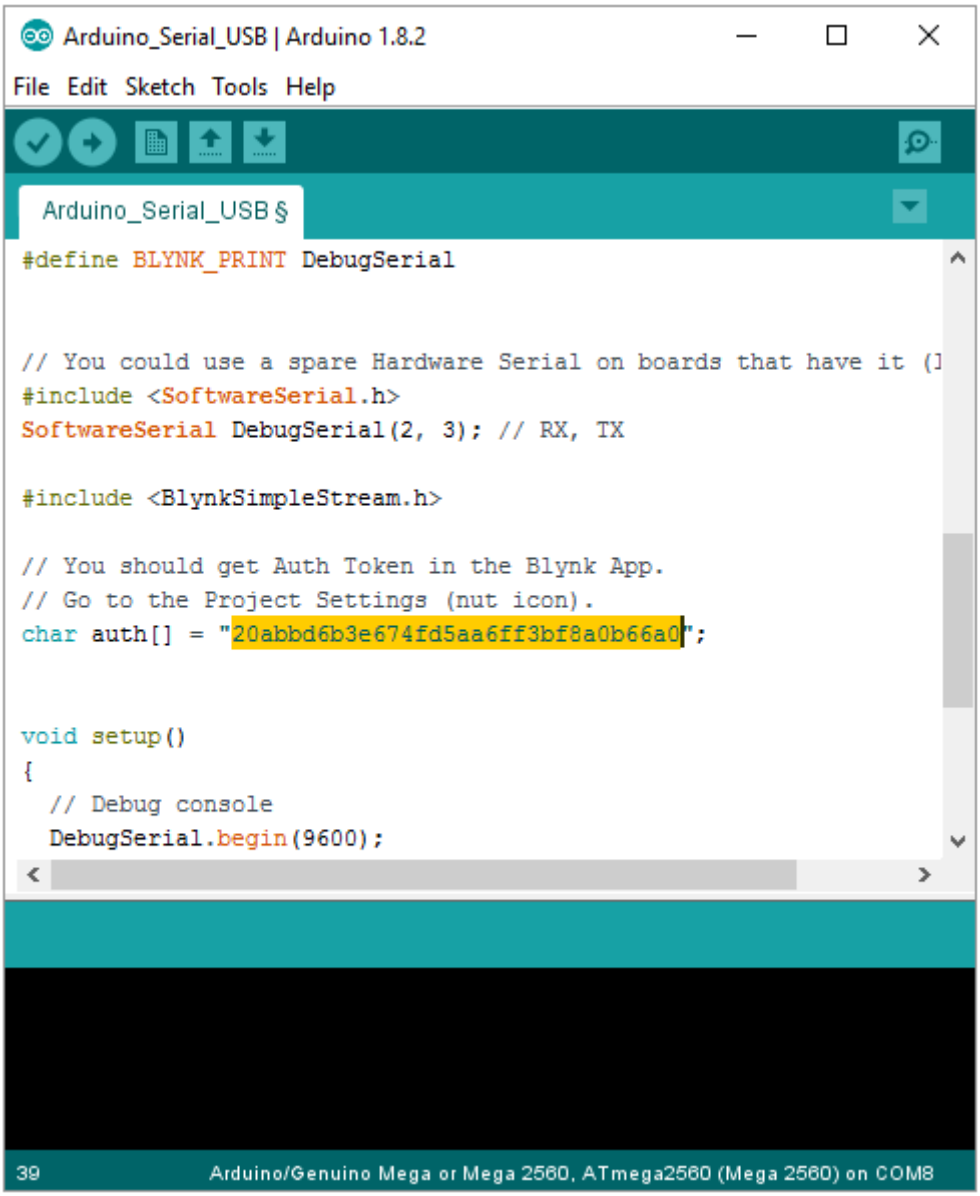
Εάν πρόκειται να χρησιμοποιήσουμε την σύνδεση του Blynk με την πλακέτα μας μέσω θύρας USB όπως έγινε και κατά την πτυχιακή αυτή εργασία.

3. To Project

Κατόπιν Θα χρειαστεί α αλλάξω το auth token και να βάλω αυτό που μου εστάλη μέσω email.



3. Το Project

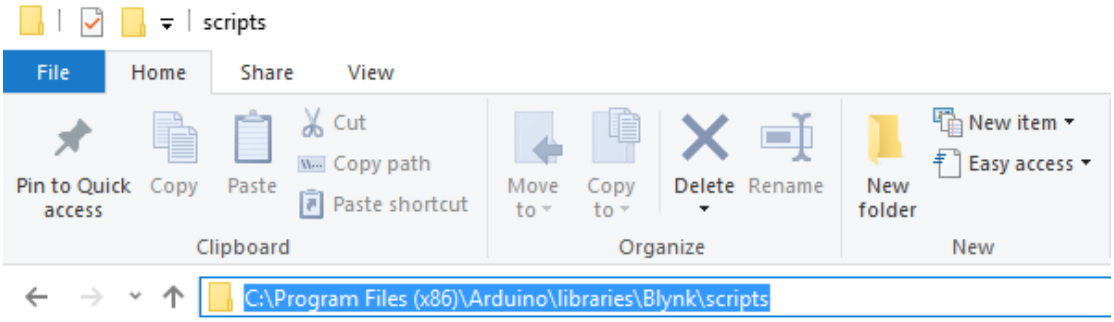


Αφού έχω αλλάξει το token και έχω επιλέξει την σωστή πλακέτα Arduino (από το Tools) αλλά και πύλη (Port) (πχ COM8) ανεβάζω το πρόγραμμα στην πλακέτα μου .

Θέλω τώρα να ανοίξω έναν σειριακό δίαυλο επικοινωνίας μεταξύ της πλακέτα και του υπολογιστή μου μέσω της θύρας USB που έχω επιλέξει.

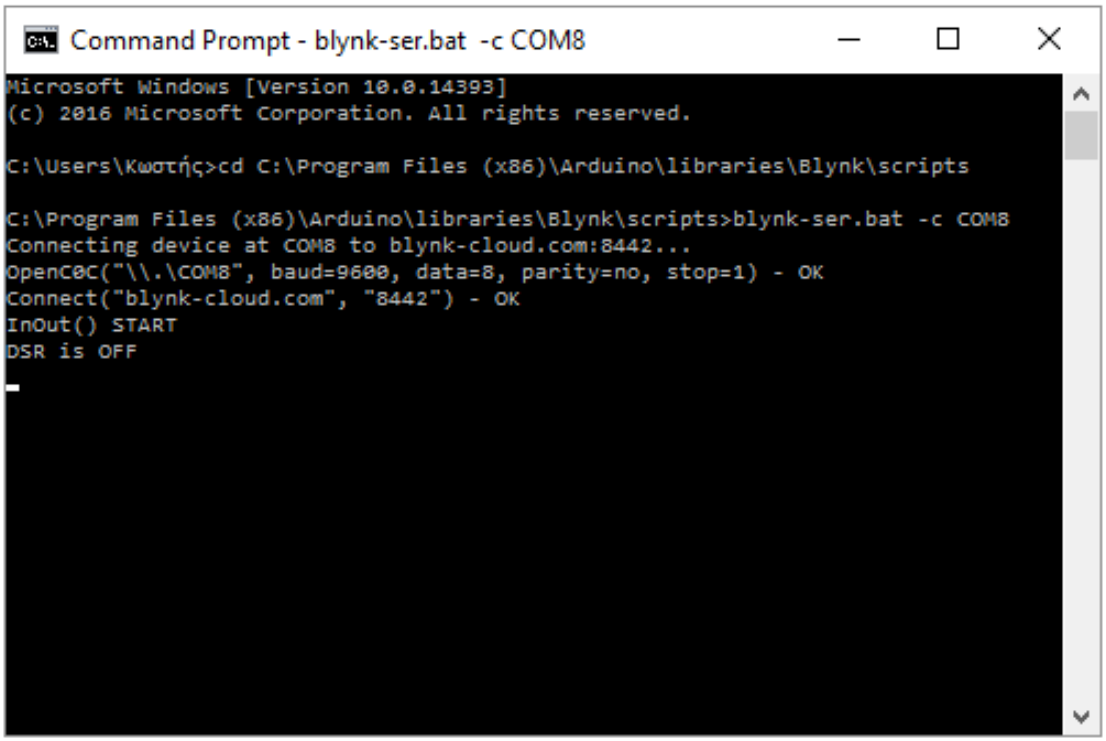
Πρέπει να κάνω τα εξής.

1) Πρέπει να βρω την τοποθεσία των Scripts



2) Να την αντιγράψω
3) Να ανοίξω το command prompt (cmd.exe αναζήτηση στα Windows) σαν διαχειριστής
4) Να πληκτρολογήσω :
   cd (κενό) και να κάνω επικόλληση την τοποθεσία των scripts. *(cd C:\Program Files (x86)\Arduino\libraries\Blynk\scripts)*
5) Να πατήσω enter και μετά από κάτω να πληκτρολογήσω
6) blynk-ser.bat -c COM8
   (εάν το COM8 είναι αυτό που χρησιμοποιώ)
7) και ξανά enter.

Εφόσον έχω ολοκληρώσει τα παραπάνω βήματα αυτό θα πρέπει να βλέπω.



Πλέον η πλακέτα μου είναι συνδεδεμένη στο διαδίκτυο και μπορώ να την χειριστώ μέσω του Blynk. Στην Android εφαρμογή θα πρέπει να βλέπω πλέον ότι η συσκευή μου είναι "ONLINE"

Για να δω εάν λειτουργεί όπως θα πρέπει χωρίς να κάνω κάποια αλλαγή στο έτοιμο παράδειγμα Arduino_Serial_USB που έχω χρησιμοποιήσει, μπορώ πολύ απλά στην εφαρμογή μου να εκμεταλλευτώ το ενσωματωμένο LED που έχουν όλα τα Arduino στην ψηφιακή έξοδο 13 . Επιλέγοντας την, μέσα στην εφαρμογή σε ένα από τα button που δημιούργησα και αλλάζοντας την κατάσταση της από ON σε OFF μπορώ να δω το ενσωματωμένο LED στην πλακέτα να ανάβει ή να σβήνει.

Στο Project μας η Android εφαρμογή Blynk θα έχει οχτώ διακόπτες switches οι οποίοι θα χρησιμοποιηθούν κατάλληλα για τον έλεγχο και την ομαλή λειτουργία των φωτεινών σηματοδοτών.

Τα ονόματα τους καθώς και οι ψηφιακές έξοδοι που θα ελέγχουν θα είναι οι εξής :

Όνομα	Ψηφιακή Είσοδος	Πλακέτα
UpDown	2	Master Transmitter Arduino Mega
LeftRight	3	
Tram1	4	
Tram2	5	
Walk	6	
LightsCheck	7	
StartUpDown	8	
SleepingMode	9	

Όπως αναγράφεται και παραπάνω κάθε ένας θα ελέγχει μια ψηφιακή είσοδο του κεντρικού υπολογιστή Master (Arduino Mega) και έτσι θα ενεργοποιείται κάθε φορά μια διαφορετική υπό-ρουτίνα του προγράμματος .

Μέσα σε κάθε μια από αυτές τις υπό-ρουτίνες ο Master θα κληθεί να μεταβιβάσει μια συγκεκριμένη πληροφορία στους Slaves.

Έχοντας τώρα ο κάθε δέκτης (Slave) μια συγκεκριμένη πληροφορία από την επικοινωνία του με τον Master διαλέγει μέσα από τις υπό-ρουτίνες του προγράμματος του πιο σημείο του κώδικα του θα εκπληρώσει μέχρι να ολοκληρωθεί ο κύκλος και να λάβει μια νέα εντολή.

4. Το πρόγραμμα (Software)

Θα χωρίσουμε το συνολικό πρόγραμμα της πτυχιακής αυτής άσκησης σε οχτώ τμήματα . Όπως έχουμε δει μέχρι τώρα το σύστημα μας θα αποτελείται από οχτώ διαφορετικούς Μίκρο-επεξεργαστές. Έναν Master Transmitter ο οποίος θα μπορεί να δέχεται τα διάφορα σήματα (εισόδους –ερεθίσματα) της διασταύρωσης μας (κουμπιά πεζών , αισθητήρες τραμ κλπ.) ή ακόμα και από άλλες διασταυρώσεις και επτά Slaves Receivers τα οποία θα δέχονται εντολές από τον Master, οπότε προφανώς ο κάθε Μίκρο-επεξεργαστής θα έχει το δικό του πρόγραμμα.

Έτσι λοιπόν θα έχουμε τα εξής οχτώ μέρη του συνολικού προγράμματος :

Master Software

Slave_1 Software

Slave_2 Software

Slave_3 Software

Slave_4 Software

Slave_5 Software

Slave_6 Software

Slave_7 Software

Μέχρι τώρα έχουμε δει ότι η επικοινωνία μεταξύ των Μίκρο – επεξεργαστών θα γίνει μέσω του I2C Bus ή αλλιώς TWI (Two Wire Interface) κεφάλαιο 3.3 . Οπότε βασικό στοιχείο και των 8 οχτώ προγραμμάτων θα είναι η εισαγωγή της απαραίτητης βιβλιοθήκης στον κώδικα για την επίτευξη της σειριακής επικοινωνίας. Η έναρξη της αναμετάδοσης σήματος από την πλευρά του Master προς τους Slaves αλλά επίσης και ο συνεχής έλεγχος από την πλευρά των Slaves για νέα σήματα από τον Master.

Ένα δεύτερο στοιχείο το οποίο θα συναντήσουμε μόνο στο πρόγραμμα του Master είναι η βιβλιοθήκη του Blynk. Μέσα από την λειτουργία του Blynk και τον μοναδικό κωδικό του (Auth. Token) όπως είδαμε στο κεφάλαιο 3.4 θα έχουμε την δυνατότητα να ελέγχουμε μερικές από τις εξόδους του Arduino μέσω οποιουδήποτε Smartphone, Tablet ή Η/Υ. Στο παράδειγμά μας θα ελέγχουμε οχτώ (8).

Οι οχτώ αυτές έξοδοι θα αντιστοιχούν στις επιθυμητές λειτουργίες της διασταύρωσης μας , έτσι λοιπόν, το πρόγραμμα μας όταν θα εντοπίζει μία μεταβολή σε κάποια από αυτές τις εξόδους θα βρίσκει σε ποία από τις παραπάνω λειτουργίες βρίσκεται ήδη και θα ενεργεί ανάλογα. Έτσι ώστε να επιτευχθεί η μετάβαση στην νέα επιθυμητή λειτουργία με τον κατάλληλο τρόπο.

Οι επιθυμητές λειτουργίες την διασταύρωσης μας είναι η εξής :

StartUpDown

UpDown

LeftRight

Tram 1

Tram 2

Walk

SleepingMode

Lights Check

Τα πιθανά σενάρια (Operations) που προκύπτουν από τον συνδυασμό των παραπάνω λειτουργιών και τελικά τον πλήρη έλεγχο της διασταύρωσης μας σε πραγματικό χρόνο είναι τα εξής :

OPERATIONS

***Όπου **(Φ x)** αντιστοιχεί **Φανάρι Αυτοκινήτων με αριθμό (x)**
Π.χ. Φ3 είναι το φανάρι των Αυτοκινήτων με αριθμό 3 .

**** Όπου **(Π x)** αντιστοιχεί **Φανάρι Πεζών με αριθμό (x)**
Π.χ. Π1 είναι το φανάρι των Πεζών με αριθμό 1.

***** Όπου **(Τ x)** αντιστοιχεί **Φανάρι Τραμ με αριθμό (x)**
Π.χ. Τ2 είναι το φανάρι του Τραμ με αριθμό 2

4 . Το Πρόγραμμα (Software)

Left Right to **(Up Down Operation)**	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π1 , Π2 , Π3 , Π6**
Ανάβει τα Κόκκινα	**Π1 , Π2 , Π3 , Π6**
DELAY 7 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ5 , Φ7**
Ανάβει τα Πορτοκαλί	**Φ5 , Φ7**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ5 , Φ7**
Ανάβει τα Κόκκινα	**Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Π5 , Π7**
Ανάβει τα Πράσινα	**Π5 , Π7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3**
Ανάβει τα Πράσινα	**Φ1 , Φ3**

*Τα φανάρια **Φ4 , Π4** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ1 και Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Up Down to Tram 1 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π7**
Ανάβει τα Κόκκινα	**Π7**
DELAY 5 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ3 , Φ4**
Ανάβει τα Πορτοκαλί	**Φ3 , Φ4**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ3 , Φ4**
Ανάβει τα Κόκκινα	**Φ3 , Φ4**
Σβήνει τα Κόκκινα	**Π3 , Π4 , Π6**
Ανάβει τα Πράσινα	**Π3 , Π4 , Π6**
Σβήνει τα Πορτοκαλί	**Τ1**
Ανάβει τα Μπλε	**Τ1**
Σβήνει τα Κόκκινα	**Φ7**
Ανάβει τα Πράσινα	**Φ7**

*Τα φανάρια **Φ1 , Π1 , Π2 , Φ5** και **Π5** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Up Down to Tram 2 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π5 , Π7**
Ανάβει τα Κόκκινα	**Π5 , Π7**
DELAY 5 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ1 , Φ3**
Ανάβει τα Πορτοκαλί	**Φ1 , Φ3**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ1 , Φ3**
Ανάβει τα Κόκκινα	**Φ1 , Φ3**
Σβήνει τα Κόκκινα	**Π1 , Π2 , Π3 , Π6**
Ανάβει τα Πράσινα	**Π1 , Π2 , Π3 , Π6**
Σβήνει τα Πορτοκαλί	**Τ2**
Ανάβει τα Μπλε	**Τ2**
Σβήνει τα Κόκκινα	**Φ5 , Φ7**
Ανάβει τα Πράσινα	**Φ5 , Φ7**

*Τα φανάρια **Φ4 , Π4** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ1** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Up Down to **(Left Right Operation)**	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π5 , Π7**
Ανάβει τα Κόκκινα	**Π5 , Π7**
DELAY 5 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ1 , Φ3**
Ανάβει τα Πορτοκαλί	**Φ1 , Φ3**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ1 , Φ3**
Ανάβει τα Κόκκινα	**Φ1 , Φ3**
Σβήνει τα Κόκκινα	**Π1 , Π2 , Π3 , Π6**
Ανάβει τα Πράσινα	**Π1 , Π2 , Π3 , Π6**
Σβήνει τα Κόκκινα	**Φ5 , Φ7**
Ανάβει τα Πράσινα	**Φ5 , Φ7**

*Τα φανάρια **Φ4 , Π4** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ1 , Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Left Right to Tram 1 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π1 , Π2**
Ανάβει τα Κόκκινα	**Π1 , Π2**
DELAY 7 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ4 , Φ5**
Ανάβει τα Πορτοκαλί	**Φ4 , Φ5**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ4 , Φ5**
Ανάβει τα Κόκκινα	**Φ4 , Φ5**
Σβήνει τα Κόκκινα	**Π4 , Π5**
Ανάβει τα Πράσινα	**Π4 , Π5**
Σβήνει τα Πορτοκαλί	**Τ1**
Ανάβει τα Μπλε	**Τ1**
Σβήνει τα Κόκκινα	**Φ1**
Ανάβει τα Πράσινα	**Φ1**

*Τα φανάρια **Φ3 , Π3 , Π6 , Φ7** και **Π7** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Left Right to Tram 2 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
-	-
-	-
DELAY 2 sec / PHASE 2	
Σβήνει τα Πορτοκαλί	**T2**
Ανάβει τα Μπλε	**T2**
DELAY 0 sec / PHASE 3	
-	-
-	-
-	-
-	-
-	-
-	-
-	-
-	-

Κανένα από τα φανάρια των πεζών ή των αυτοκινήτων δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **T1** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 1 to Up Down Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π3 , Π4 , Π6**
Ανάβει τα Κόκκινα	**Π3 , Π4 , Π6**
DELAY 7 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ5 , Φ7**
Ανάβει τα Πορτοκαλί	**Φ5 , Φ7**
Σβήνει τα Μπλε	**Τ1**
Ανάβει τα Πορτοκαλί	**Τ1**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ5 , Φ7**
Ανάβει τα Κόκκινα	**Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Π5 , Π7**
Ανάβει τα Πράσινα	**Π5 , Π7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3 , Φ4**
Ανάβει τα Πράσινα	**Φ1 , Φ3 , Φ4**

*Τα φανάρια **Φ1 , Π1 , Π2 , Φ5** και **Π5** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 1 to Left Right Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π4 , Π5**
Ανάβει τα Κόκκινα	**Π4 , Π5**
DELAY 5 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ1**
Ανάβει τα Πορτοκαλί	**Φ1**
Σβήνει τα Μπλε	**Τ1**
Ανάβει τα Πορτοκαλί	**Τ1**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ1**
Ανάβει τα Κόκκινα	**Φ1**
Σβήνει τα Κόκκινα	**Π1 , Π2**
Ανάβει τα Πράσινα	**Π1 , Π2**
Σβήνει τα Κόκκινα	**Φ4 , Φ5**
Ανάβει τα Πράσινα	**Φ4 , Φ5**

*Τα φανάρια **Φ3 , Π3 , Π6 , Φ7** και **Π7** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ2** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 1 to Tram 2 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π4 , Π5**
Ανάβει τα Κόκκινα	**Π4 , Π5**
DELAY 5 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ1**
Ανάβει τα Πορτοκαλί	**Φ1**
Σβήνει τα Μπλε	**Τ1**
Ανάβει τα Πορτοκαλί	**Τ1**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ1**
Ανάβει τα Κόκκινα	**Φ1**
Σβήνει τα Κόκκινα	**Π1 , Π2**
Ανάβει τα Πράσινα	**Π1 , Π2**
Σβήνει τα Πορτοκαλί	**Τ2**
Ανάβει τα Μπλε	**Τ2**
Σβήνει τα Κόκκινα	**Φ4 , Φ5**
Ανάβει τα Πράσινα	**Φ4 , Φ5**

*Τα φανάρια **Φ3 , Π3 , Π6 , Φ7** και **Π7** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 2 to Up Down Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π1 , Π2 , Π3 , Π6**
Ανάβει τα Κόκκινα	**Π1 , Π2 , Π3 , Π6**
DELAY 7 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ5 , Φ7**
Ανάβει τα Πορτοκαλί	**Φ5 , Φ7**
Σβήνει τα Μπλε	**Τ2**
Ανάβει τα Πορτοκαλί	**Τ2**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ5 , Φ7**
Ανάβει τα Κόκκινα	**Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Π5 , Π7**
Ανάβει τα Πράσινα	**Π5 , Π7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3**
Ανάβει τα Πράσινα	**Φ1 , Φ3**

*Τα φανάρια **Φ4** και **Π4** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **Τ1** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 2 to Left Right Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
-	-
-	-
DELAY 2 sec / PHASE 2	
Σβήνει τα Μπλε	**T2**
Ανάβει τα Πορτοκαλί	**T2**
DELAY 4 sec / PHASE 3	
-	-
-	-
-	-
-	-
-	-
-	-

Κανένα από τα φανάρια των πεζών ή των αυτοκινήτων δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.

*Τα Πορτοκαλί και Μπλε φανάρια του Τραμ , **T1** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Tram 2 to Tram 1 Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Πράσινα	**Π1 , Π2**
Ανάβει τα Κόκκινα	**Π1 , Π2**
DELAY 7 sec / PHASE 2	
Σβήνει τα Πράσινα	**Φ4 , Φ5**
Ανάβει τα Πορτοκαλί	**Φ4 , Φ5**
Σβήνει τα Μπλε	**Τ2**
Ανάβει τα Πορτοκαλί	**Τ2**
DELAY 4 sec / PHASE 3	
Σβήνει τα Πορτοκαλί	**Φ4 , Φ5**
Ανάβει τα Κόκκινα	**Φ4 , Φ5**
Σβήνει τα Κόκκινα	**Π4 , Π5**
Ανάβει τα Πράσινα	**Π4 , Π5**
Σβήνει τα Πορτοκαλί	**Τ1**
Ανάβει τα Μπλε	**Τ1**
Σβήνει τα Κόκκινα	**Φ1**
Ανάβει τα Πράσινα	**Φ1**

*Τα φανάρια **Φ3 , Π3 , Π6 , Φ7** και **Π7** δεν θα υποστούν κάποια αλλαγή σε αυτήν την μετάβαση τους συστήματος.*

Sleeping Mode Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Σβήνει τα Μπλε	**Τ1 , Τ2**
Ανάβει τα Πορτοκαλί	**Τ1 , Τ2 , Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
Σβήνει τα Κόκκινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
DELAY 1 sec / PHASE 2	
Σβήνει τα Μπλε	**Τ1 , Τ2**
Σβήνει τα Πορτοκαλί	**Τ1 , Τ2 , Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
Σβήνει τα Κόκκινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
DELAY 1 sec	

Lights Check Operation	
Περιγραφή Λειτουργίας	***Επηρεαζόμενα Φανάρια***
PHASE 1	
Ανάβει τα Μπλε	**Τ1 , Τ2**
Ανάβει τα Πορτοκαλί	**Τ1 , Τ2 , Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Ανάβει τα Πράσινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Ανάβει τα Κόκκινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Ανάβει τα Πράσινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
Ανάβει τα Κόκκινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
DELAY 1 sec / PHASE 2	
Σβήνει τα Μπλε	**Τ1 , Τ2**
Σβήνει τα Πορτοκαλί	**Τ1 , Τ2 , Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Κόκκινα	**Φ1 , Φ3 , Φ4 , Φ5 , Φ7**
Σβήνει τα Πράσινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
Σβήνει τα Κόκκινα	**Π1 , Π2 , Π3 , Π4 , Π5 , Π6 , Π7**
DELAY 1 sec	

MASTER SOFTWARE

```
/*  TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 */

// Ορίζει την Θέση του κάθε Slave
const byte slave1 = 1;
const byte slave2 = 2;
const byte slave3 = 3;
const byte slave4 = 4;
const byte slave5 = 5;
const byte slave6 = 6;
const byte slave7 = 7;

 // Θέτω τις παραμέτρους που θα χρησιμοποιήσω ίσες με μηδέν
int RemoteControlUpDown = 0 ;
int RemoteControlLeftRight = 0 ;
int RemoteControlTram1 = 0 ;
int RemoteControlTram2 = 0 ;
int RemoteControlWalk = 0 ;
int RemoteControlStartUpDown = 0 ;
int RemoteControlLightsCheck = 0 ;
int RemoteControlSleepingMode = 0 ;

int Tram11Sensor = 0 ;
int Tram12Sensor = 0 ;
int Tram21Sensor = 0 ;
int Tram22Sensor = 0 ;
int WalkSensor = 0 ;

int UpDownFlag = 0 ;
int LeftRightFlag = 0 ;
int Tram1Flag = 0 ;
int Tram2Flag = 0 ;
```

```
// Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

// Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την λειτουργία του Blynk
#define BLYNK_PRINT Serial1
#include <BlynkSimpleStream.h>

//  Auth Token
char auth[] = "20abbd6b3e674fd5aa6ff3bf8a0b66a0";


void setup(){

  Wire.begin();

 // Debug console
  Serial1.begin(9600);

  // Blynk will work through Serial
  Serial.begin(9600);

  Blynk.begin(Serial, auth);

// Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

    // Remote Control Inputs ( Κενά PINs )
    pinMode(2,INPUT);   // RemoteControlUpDown
    pinMode(3,INPUT);   // RemoteControlLeftRight
    pinMode(4,INPUT);   // RemoteControlTram1
    pinMode(5,INPUT);   // RemoteControlTram2
    pinMode(6,INPUT);   // RemoteControlWalk
    pinMode(7,INPUT);   // RemoteControl_Lights_Check_Operation
    pinMode(8,INPUT);   // RemoteControl_Start_UpDown_Operation
    pinMode(9,INPUT);   // RemoteControl_Sleeping_Mode
    pinMode(10,INPUT);  // RemoteControl_
```

```
// Buttons
    pinMode(26,INPUT);   // Tram11Sensor
    pinMode(28,INPUT);   // Tram12Sensor
    pinMode(30,INPUT);   // Tram21Sensor
    pinMode(32,INPUT);   // Tram22Sensor
    pinMode(34,INPUT);   // WalkSensor


   // Tram Lights
   pinMode(36,OUTPUT); // Tram 1 Μπλε
   pinMode(38,OUTPUT); // Tram 1 Πορτοκαλί
   pinMode(40,OUTPUT); // Tram 2 Μπλε
   pinMode(42,OUTPUT); // Tram 1 Πορτοκαλί
}



void loop(){

// Ενεργοποιεί το Blynk
    Blynk.run();

 // Δίνω κατάλληλα ονόματα στις ψηφιακές εισόδους που χρησιμοποιώ
  RemoteControlUpDown = digitalRead(2);
  RemoteControlLeftRight = digitalRead(3);
  RemoteControlTram1 = digitalRead(4);
  RemoteControlTram2 = digitalRead(5);
  RemoteControlWalk = digitalRead(6);
  RemoteControlLightsCheck = digitalRead(7);
  RemoteControlStartUpDown = digitalRead(8);
  RemoteControlSleepingMode = digitalRead(9);

  Tram11Sensor = digitalRead(26);
  Tram12Sensor = digitalRead(28);
  Tram21Sensor = digitalRead(30);
  Tram22Sensor = digitalRead(32);
```

```
  WalkSensor = digitalRead(34);

 // Ξεκινάει να ελέγχει τα μεταβλητά στοιχεία του προγράμματος

  //Εάν έχει πατηθεί από τον χρήστη το κουμπί StartUpDown
if (RemoteControlStartUpDown == HIGH){
  UpDownFlag = 1 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  StartUpDownTransmition();
  }


//Εάν έχει πατηθεί από τον χρήστη το κουμπί UpDown
if (RemoteControlUpDown == HIGH){
    // Εάν είναι ήδη στην λειτουργία αυτή θα αγνοήσει την εντολή, δεν θα κάνει τίποτα.
    if (UpDownFlag == 1){
    }
    // Εάν είναι στην λειτουργία LeftRight θα εκτελέσει την υπό-ρουτίνα, UpDownTransmition.
    if (LeftRightFlag == 1){
        UpDownTransmition();
    }
    // Εάν είναι στην λειτουργία Tram1 θα εκτελέσει την υπό-ρουτίνα,
Tram1toUpDownTransmition.
    if (Tram1Flag == 1){
        Tram1toUpDownTransmition();
    }
    // Εάν είναι στην λειτουργία Tram2 θα εκτελέσει την υπό-ρουτίνα,
Tram2toUpDownTransmition.
    if (Tram2Flag == 1){
        Tram2toUpDownTransmition();
    }
}
```

```
 //Εάν έχει πατηθεί από τον χρήστη το κουμπί LeftRight
if (RemoteControlLeftRight == HIGH){
   // Εάν είναι ήδη στην λειτουργία αυτή θα αγνοήσει την εντολή, δεν θα κάνει τίποτα.
   if (LeftRightFlag == 1){
   }
   if (UpDownFlag == 1){
      LeftRightTransmition();
   }
   if (Tram1Flag == 1){
      Tram1toLeftRightTransmition();
   }
   if (Tram2Flag == 1){
       Tram2toLeftRightTransmition();
   }
}




  // Εάν έχει πατηθεί από τον χρήστη το κουμπί Tram1
  //     ή
  //    έχει ενεργοποιηθεί ο αισθητήρας εισόδου του Τραμ 1

if (RemoteControlTram1 == HIGH  || Tram11Sensor == HIGH ){
   // Εάν είναι ήδη στην λειτουργία αυτή θα αγνοήσει την εντολή, δεν θα κάνει τίποτα.
   if (Tram1Flag == 1){
   }
   if (UpDownFlag == 1){
      UpDowntoTram1Transmition();
   }
   if (LeftRightFlag == 1){
      LeftRighttoTram1Transmition();
   }
   if (Tram2Flag == 1){
      Tram2toTram1Transmition();
   }
}
```

```
// Εάν έχει πατηθεί από τον χρήστη το κουμπί Tram2
  //      ή
  //     έχει ενεργοποιηθεί ο αισθητήρας εισόδου του Τραμ 2

if (RemoteControlTram2 == HIGH  || Tram21Sensor == HIGH ){
    // Εάν είναι ήδη στην λειτουργία αυτή θα αγνοήσει την εντολή, δεν θα κάνει τίποτα.
    if (Tram2Flag == 1){
    }
    if (UpDownFlag == 1){
        UpDowntoTram2Transmition();
    }
    if (LeftRightFlag == 1){
        LeftRighttoTram2Transmition();
    }
    if (Tram1Flag == 1){
        Tram1toTram2Transmition();
    }
}




  // Εάν έχει πατηθεί από τον χρήστη το κουμπί Walk
  //      ή
  //     έχει ενεργοποιηθεί ο αισθητήρας εισόδου Walk

if (RemoteControlWalk == HIGH || WalkSensor == HIGH ){
    // Εάν βρίσκεται σε οποιαδήποτε λειτουργία εκτός της Tram1 θα εκτελέσει την υπό-ρουτίνα,
WalkTransmition.
    if (Tram1Flag == 0){
        WalkTransmition();
       }
  }
```

```
  // Εάν έχει ενεργοποιηθεί ο αισθητήρας εξόδου του Τραμ 2
if ( Tram22Sensor == HIGH ){
   // Εάν είναι στην λειτουργία Tram2 θα εκτελέσει την υπό-ρουτίνα,
      Tram2toUpDownTransmition.
   if (Tram2Flag == 1){
      Tram2toUpDownTransmition();
   }
}


  // Εάν έχει ενεργοποιηθεί ο αισθητήρας εξόδου του Τραμ 1
if ( Tram12Sensor == HIGH ){
   // Εάν είναι στην λειτουργία Tram1 θα εκτελέσει την υπό-ρουτίνα,
       Tram1toUpDownTransmition.
   if (Tram1Flag == 1){
      Tram1toUpDownTransmition();
   }
}


  // Εάν έχει πατηθεί από τον χρήστη το κουμπί LightsCheck θα εκτελέσει την υπό-ρουτίνα,
LightsCheck.
if (RemoteControlLightsCheck == HIGH){
   LightsCheck();
  }

  // Εάν έχει πατηθεί από τον χρήστη το κουμπί SleepingMode θα εκτελέσει την υπό-ρουτίνα,
SleepingModeTransmition.
if (RemoteControlSleepingMode == HIGH){
   SleepingModeTransmition();
  }

}
```

```
// Υπό-ρουτίνα για την μετάβαση των φαναριών από το δεξιά - αριστερά
   // στο πάνω - κάτω
void UpDownTransmition(){
  UpDownFlag = 1 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  UpDownTransmition_Phase_1();
  delay(7000);
  UpDownTransmition_Phase_2();
  delay(4000);
  UpDownTransmition_Phase_3();
}

   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το πάνω - κάτω
   // στο δεξιά - αριστερά
void LeftRightTransmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 1 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  LeftRightTransmition_Phase_1();
  delay(5000);
  LeftRightTransmition_Phase_2();
  delay(4000);
  LeftRightTransmition_Phase_3();
}
```

```
// Υπό-ρουτίνα για την μετάβαση των φαναριών από το πάνω – κάτω στο Τραμ 1
void UpDowntoTram1Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 1 ;
  Tram2Flag = 0 ;
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  UpDowntoTram1Transmition_Phase_1();
  delay(5000);
  UpDowntoTram1Transmition_Phase_2();
  delay(4000);
  digitalWrite(38,LOW);  // ( Τραμ 1 Πορτοκαλί = OFF )
  digitalWrite(36,HIGH);  // ( Τραμ 1 Μπλε = ON )
  UpDowntoTram1Transmition_Phase_3();
}

   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το πάνω - κάτω στο Τραμ 2
void UpDowntoTram2Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 1 ;
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  UpDowntoTram2Transmition_Phase_1();
  delay(5000);
  UpDowntoTram2Transmition_Phase_2();
  delay(4000);
  digitalWrite(42,LOW);  // ( Τραμ 2 Πορτοκαλί = OFF )
  digitalWrite(40,HIGH);  // ( Τραμ 2 Μπλε = ON )
  UpDowntoTram2Transmition_Phase_3();
```

```
}

   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το δεξιά - αριστερά
   // στο Τραμ 1
void LeftRighttoTram1Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 1 ;
  Tram2Flag = 0 ;
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  LeftRighttoTram1Transmition_Phase_1();
  delay(7000);
  LeftRighttoTram1Transmition_Phase_2();
  delay(4000);
  digitalWrite(38,LOW);  // ( Τραμ 1 Πορτοκαλί = OFF )
  digitalWrite(36,HIGH);  // ( Τραμ 1 Μπλε = ON )
  LeftRighttoTram1Transmition_Phase_3();
}


   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το δεξιά - αριστερά
   // στο Τραμ 2
void LeftRighttoTram2Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 1 ;
  LeftRighttoTram2Transmition_Phase_1();
  delay(2000);
  LeftRighttoTram2Transmition_Phase_2();
  LeftRighttoTram2Transmition_Phase_3();
  digitalWrite(42,LOW);  // ( Τραμ 2 Πορτοκαλί = OFF )
  digitalWrite(40,HIGH);  // ( Τραμ 2 Μπλε = ON )
}
```

```
 // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 1
   // στο πάνω - κάτω
void Tram1toUpDownTransmition(){
  UpDownFlag = 1 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  Tram1toUpDownTransmition_Phase_1();
  delay(7000);
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  Tram1toUpDownTransmition_Phase_2();
  delay(4000);
  Tram1toUpDownTransmition_Phase_3();
}

   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 2
   // στο πάνω - κάτω
void Tram2toUpDownTransmition(){
  UpDownFlag = 1 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  UpDownTransmition_Phase_1();
  delay(5000);
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  UpDownTransmition_Phase_2();
  delay(4000);
  UpDownTransmition_Phase_3();
}
```

```
   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 1
   // στο δεξιά - αριστερά

void Tram1toLeftRightTransmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 1 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  Tram1toLeftRightTransmition_Phase_1();
  delay(5000);
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  Tram1toLeftRightTransmition_Phase_2();
  delay(4000);
  Tram1toLeftRightTransmition_Phase_3();
}


   // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 2
   // στο δεξιά - αριστερά
void Tram2toLeftRightTransmition(){
  delay(2000);
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
}
```

```
  // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 1
  // στο Τραμ 2
void Tram1toTram2Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 1 ;
  Tram1toLeftRightTransmition_Phase_1();
  delay(5000);
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  Tram1toLeftRightTransmition_Phase_2();
  delay(4000);
  digitalWrite(42,LOW);  // ( Τραμ 2 Πορτοκαλί = OFF )
  digitalWrite(40,HIGH);  // ( Τραμ 2 Μπλε = ON )
  Tram1toLeftRightTransmition_Phase_3();
}


  // Υπό-ρουτίνα για την μετάβαση των φαναριών από το Τραμ 2
  // στο Τραμ 1
void Tram2toTram1Transmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 1 ;
  Tram2Flag = 0 ;
  LeftRighttoTram1Transmition_Phase_1();
  delay(7000);
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  LeftRighttoTram1Transmition_Phase_2();
  delay(4000);
  digitalWrite(38,LOW);  // ( Τραμ 1 Πορτοκαλί = OFF )
  digitalWrite(36,HIGH);  // ( Τραμ 1 Μπλε = ON )
  LeftRighttoTram1Transmition_Phase_3();
}
```

```
   // Υπό-ρουτίνα λειτουργίας φαναριού πεζών
void WalkTransmition(){
  WalkTransmition_Phase_1();
  delay(3000);
  WalkTransmition_Phase_2();
  delay(3000);
  WalkTransmition_Phase_3();
  delay(3000);
  WalkTransmition_Phase_4();
}


   // Υπό-ρουτίνα λειτουργίας SleepingMode (  Παλλόμενο πορτοκαλί )
void SleepingModeTransmition(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  digitalWrite(36,LOW);  // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW);  // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(38,HIGH);  // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH);  // ( Τραμ 2 Πορτοκαλί = ON )
  SleepingModeTransmition_Phase_1();
  delay(1000);
  digitalWrite(38,LOW);  // ( Τραμ 1 Πορτοκαλί = OFF )
  digitalWrite(42,LOW);  // ( Τραμ 2 Πορτοκαλί = OFF )
  SleepingModeTransmition_Phase_2();
  delay(1000);
}
```

```
   // Υπό-ρουτίνα Ελέγχου Καμένων Φαναριών
void LightsCheck(){
  UpDownFlag = 0 ;
  LeftRightFlag = 0 ;
  Tram1Flag = 0 ;
  Tram2Flag = 0 ;
  digitalWrite(36,HIGH); // ( Τραμ 1 Μπλε = ON )
  digitalWrite(40,HIGH); // ( Τραμ 2 Μπλε = ON )
  digitalWrite(38,HIGH); // ( Τραμ 1 Πορτοκαλί = ON )
  digitalWrite(42,HIGH); // ( Τραμ 2 Πορτοκαλί = ON )
  LightsCheckTransmition_Phase_1();
  delay(1000);
  digitalWrite(36,LOW); // ( Τραμ 1 Μπλε = OFF )
  digitalWrite(40,LOW); // ( Τραμ 2 Μπλε = OFF )
  digitalWrite(38,LOW); // ( Τραμ 1 Πορτοκαλί = OFF )
  digitalWrite(42,LOW); // ( Τραμ 2 Πορτοκαλί = OFF )
  LightsCheckTransmition_Phase_2();
  delay(1000);
}




  /* Wire Coding
   * R = RemoteControlStartUpDown_Operation
   * r = RemoteControlStartLeftRight_Operation
   *
   * Q = RemoteControlUpDown_Operation_Phase_1
   * q = RemoteControlUpDown_Operation_Phase_2
   * W = RemoteControlUpDown_Operation_Phase_3
   * w = RemoteControlUpDown_Operation_Phase_4
   *
   * A = RemoteControlLeftRight_Operation_Phase_1
   * a = RemoteControlLeftRight_Operation_Phase_2
   * Z = RemoteControlLeftRight_Operation_Phase_3
   * z = RemoteControlLeftRight_Operation_Phase_4
```

```
 *


 * T = RemoteControlUpDowntoTram1_Operation_Phase_1
 * t = RemoteControlUpDowntoTram1_Operation_Phase_2
 * Y = RemoteControlUpDowntoTram1_Operation_Phase_3
 *
 * F = RemoteControlUpDowntoTram2_Operation_Phase_1
 * f = RemoteControlUpDowntoTram2_Operation_Phase_2
 * G = RemoteControlUpDowntoTram2_Operation_Phase_3
 *
 * E = RemoteControlLeftRighttoTram1_Operation_Phase_1
 * e = RemoteControlLeftRighttoTram1_Operation_Phase_2
 * D = RemoteControlLeftRighttoTram1_Operation_Phase_3
 *
 * U = RemoteControlLeftRighttoTram2_Operation_Phase_1
 * u = RemoteControlLeftRighttoTram2_Operation_Phase_2
 * I = RemoteControlLeftRighttoTram2_Operation_Phase_3
 *
 * O = RemoteControlTram1toUpDownTransmition_Phase_1
 * o = RemoteControlTram1toUpDownTransmition_Phase_2
 * P = RemoteControlTram1toUpDownTransmition_Phase_3
 *
 * K = RemoteControlTram1toLeftRightTransmition_Phase_1
 * k = RemoteControlTram1toLeftRightTransmition_Phase_2
 * L = RemoteControlTram1toLeftRightTransmition_Phase_3
 *
 * C = RemoteControl_Lights_Check_Operation_Phase_1
 * c = RemoteControl_Lights_Check_Operation_Phase_2
 *
 * S = RemoteControl_Sleeping_Mode_Phase_1
 * s = RemoteControl_Sleeping_Mode_Phase_2
 *
 * H = RemoteControlWalk_Operation_Phase_1
 * h = RemoteControlWalk_Operation_Phase_2
 * J = RemoteControlWalk_Operation_Phase_3
 * j = RemoteControlWalk_Operation_Phase_4
 */
```

```
// Υπό-ρουτίνες Διαβίβασης εντολών στα 7 Slaves

void Tram1toUpDownTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('O');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('O');
  Wire.endTransmission();
}

void Tram1toUpDownTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('o');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('o');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('o');
  Wire.endTransmission();
```

```
  Wire.beginTransmission(slave4);

  Wire.write('o');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('o');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('o');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('o');
  Wire.endTransmission();
}

void Tram1toUpDownTransmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('P');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('P');
  Wire.endTransmission();
}
```

```
void Tram1toLeftRightTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('K');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('K');
  Wire.endTransmission();
}

void Tram1toLeftRightTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('k');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('k');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('k');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('k');
```

```
  Wire.endTransmission();

  Wire.beginTransmission(slave5);
  Wire.write('k');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('k');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('k');
  Wire.endTransmission();
}

void Tram1toLeftRightTransmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('L');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('L');
  Wire.endTransmission();
}
```

```
void UpDownTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('Q');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('Q');
  Wire.endTransmission();
}

void UpDownTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('q');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('q');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('q');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('q');
```

```
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('q');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('q');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('q');
  Wire.endTransmission();
}
void UpDownTransmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('W');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('W');
  Wire.endTransmission();
}
```

```
void UpDownTransmition_Phase_4(){
  Wire.beginTransmission(slave1);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('w');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('w');
  Wire.endTransmission();
}

void LeftRightTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('A');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('A');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('A');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('A');
```

```
  Wire.endTransmission();

  Wire.beginTransmission(slave5);
  Wire.write('A');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('A');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('A');
  Wire.endTransmission();
}

void LeftRightTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('a');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('a');
  Wire.endTransmission();
}
```

```
void LeftRightTransmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('Z');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('Z');
  Wire.endTransmission();
}

void LeftRightTransmition_Phase_4(){
  Wire.beginTransmission(slave1);
  Wire.write('z');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('z');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('z');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('z');
```

```
  Wire.endTransmission();

  Wire.beginTransmission(slave5);
  Wire.write('z');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('z');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('z');
  Wire.endTransmission();
}

void UpDowntoTram1Transmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('T');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('T');
  Wire.endTransmission();
}
```

```
void UpDowntoTram1Transmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('t');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('t');
  Wire.endTransmission();
}

void UpDowntoTram1Transmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('y');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('y');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('y');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('y');
```

```
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('Y');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('Y');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('Y');
  Wire.endTransmission();
}

void LeftRighttoTram1Transmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('E');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('E');
  Wire.endTransmission();
}
```

```
void LeftRighttoTram1Transmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('e');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('e');
  Wire.endTransmission();
}

void LeftRighttoTram1Transmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('D');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('D');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('D');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('D');
```

```
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('D');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('D');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('D');
  Wire.endTransmission();
}

void UpDowntoTram2Transmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('F');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('F');
  Wire.endTransmission();
}
```

```
void UpDowntoTram2Transmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('f');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('f');
  Wire.endTransmission();
}

void UpDowntoTram2Transmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('G');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('G');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('G');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('G');
```

```
  Wire.endTransmission();

  Wire.beginTransmission(slave5);
  Wire.write('G');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('G');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('G');
  Wire.endTransmission();
}

void LeftRighttoTram2Transmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('U');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('U');
  Wire.endTransmission();
}
```

```
void LeftRighttoTram2Transmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('u');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('u');
  Wire.endTransmission();
}

void LeftRighttoTram2Transmition_Phase_3(){
  Wire.beginTransmission(slave1);
  Wire.write('I');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('I');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('I');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('I');
```

```
  Wire.endTransmission();

  Wire.beginTransmission(slave5);
  Wire.write('I');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('I');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('I');
  Wire.endTransmission();
}

void SleepingModeTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('S');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('S');
  Wire.endTransmission();
}
```

```
void SleepingModeTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('s');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('s');
  Wire.endTransmission();
}

void LightsCheckTransmition_Phase_1(){
  Wire.beginTransmission(slave1);
  Wire.write('C');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('C');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('C');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('C');
```

```
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('C');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('C');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('C');
  Wire.endTransmission();
}

void LightsCheckTransmition_Phase_2(){
  Wire.beginTransmission(slave1);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('c');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('c');
  Wire.endTransmission();
}
```

```
void WalkTransmition_Phase_1(){
  Wire.beginTransmission(slave4);
  Wire.write('H');
  Wire.endTransmission();
}

void WalkTransmition_Phase_2(){
  Wire.beginTransmission(slave4);
  Wire.write('h');
  Wire.endTransmission();
}

void WalkTransmition_Phase_3(){
  Wire.beginTransmission(slave4);
  Wire.write('J');
  Wire.endTransmission();
}

void WalkTransmition_Phase_4(){
  Wire.beginTransmission(slave4);
  Wire.write('j');
  Wire.endTransmission();
}


void StartUpDownTransmition(){
  Wire.beginTransmission(slave1);
  Wire.write('R');
  Wire.endTransmission();
  Wire.beginTransmission(slave2);
  Wire.write('R');
  Wire.endTransmission();
  Wire.beginTransmission(slave3);
  Wire.write('R');
  Wire.endTransmission();
  Wire.beginTransmission(slave4);
```

```
  Wire.write('R');

  Wire.endTransmission();
  Wire.beginTransmission(slave5);
  Wire.write('R');
  Wire.endTransmission();
  Wire.beginTransmission(slave6);
  Wire.write('R');
  Wire.endTransmission();
  Wire.beginTransmission(slave7);
  Wire.write('R');
  Wire.endTransmission();
}
```

SLAVE_1 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_1 Software
 */

  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 1
const byte slave1 = 1;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave1);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ1 Πράσινο ( Φανάρι Αυτοκινήτων 1 Πράσινο)
  pinMode(3,OUTPUT); // Φ1 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ1 Κόκκινο
  pinMode(5,OUTPUT); // Π1 Πράσινο ( Φανάρι Πεζών 1 Πράσινο)
  pinMode(6,OUTPUT); // Π1 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...

}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



     // Ελέγχει για νέα σήματα από τον Master και εκτελεί την επιθυμητή υπό-ρουτίνα
   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }



   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
   RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
   RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
   RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
   RemoteControlLeftRightOperation_Phase_4();
  }



  if (inChar == 'C'){
   RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
   RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
   //S = RemoteControl_Sleeping_Mode
   RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
   //s = RemoteControl_Sleeping_Mode
   RemoteControlSleepingModeOperation_Phase_2();
  }

 }
```

```
 /*
  * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 1 ).
  * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
  * έχουν τουλάχιστον 3 φάσεις.
  *
  * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
  *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
  *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
  *
  * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
  *
  * Η 3η φάση θα  :
  *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
  *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
  *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
  *
  * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
  * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
  * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
  * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
  */

void RemoteControlStartUpDownOperation(){
    digitalWrite(2,HIGH); //( Φ1 πράσινο = ON )
    digitalWrite(3,LOW); //( Φ1 πορτοκαλί = OFF )
    digitalWrite(4,LOW); //( Φ1 κόκκινο = OFF )
    digitalWrite(5,LOW); //( Π1 πράσινο = OFF )
    digitalWrite(6,HIGH); //( Π1 κόκκινο = ON )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
     digitalWrite(5,LOW); //( Π1 πράσινο = OFF )
     digitalWrite(6,HIGH); //( Π1 κόκκινο = ON )
   }
void RemoteControlUpDownOperation_Phase_2(){
 }
void RemoteControlUpDownOperation_Phase_3(){
    digitalWrite(4,LOW); //( Φ1 κόκκινο = OFF )
    digitalWrite(2,HIGH); //( Φ1 πράσινο = ON )
   }
void RemoteControlUpDownOperation_Phase_4(){
   }




void RemoteControlLeftRightOperation_Phase_1(){
   }
void RemoteControlLeftRightOperation_Phase_2(){
   digitalWrite(2,LOW); //( Φ1 πράσινο = OFF )
   digitalWrite(3,HIGH); //( Φ1 πορτοκαλί = ON )
   }
void RemoteControlLeftRightOperation_Phase_3(){
   digitalWrite(3,LOW); //( Φ1 πορτοκαλί = OFF )
   digitalWrite(4,HIGH); //( Φ1 κόκκινο = ON )
   digitalWrite(6,LOW); //( Π1 κόκκινο = OFF )
   digitalWrite(5,HIGH); //( Π1 πράσινο = ON )
   }
void RemoteControlLeftRightOperation_Phase_4(){
   }
```

```
void RemoteControlUpDowntoTram1_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
}




void RemoteControlUpDowntoTram2_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
     digitalWrite(2,LOW); //( Φ1 πράσινο = OFF )
     digitalWrite(3,HIGH); //( Φ1 πορτοκαλί = ON )
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
   digitalWrite(3,LOW); //( Φ1 πορτοκαλί = OFF )
   digitalWrite(4,HIGH); //( Φ1 κόκκινο = ON )
   digitalWrite(6,LOW); //( Π1 κόκκινο = OFF )
   digitalWrite(5,HIGH); //( Π1 πράσινο = ON )
}




void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
   digitalWrite(5,LOW); //( Π1 πράσινο = OFF )
   digitalWrite(6,HIGH); //( Π1 κόκκινο = ON )
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
     digitalWrite(4,LOW); //( Φ1 κόκκινο = OFF )
     digitalWrite(2,HIGH); //( Φ1 πράσινο = ON )
}
```

```
void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}


void RemoteControlTram1toUpDown_Operation_Phase_1(){
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
}



void RemoteControlTram1toLeftRightTransmition_Phase_1(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
    digitalWrite(2,LOW); //( Φ1 πράσινο = OFF )
    digitalWrite(3,HIGH); //( Φ1 πορτοκαλί = ON )
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
    digitalWrite(3,LOW); //( Φ1 πορτοκαλί = OFF )
    digitalWrite(4,HIGH); //( Φ1 κόκκινο = ON )
    digitalWrite(6,LOW); //( Π1 κόκκινο = OFF )
    digitalWrite(5,HIGH); //( Π1 πράσινο = ON )
}
```

```
void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}



void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_2 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_2 Software
 */

  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 2
const byte slave2 = 2;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave2);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ2 Πράσινο ( Φανάρι Αυτοκινήτων 2 Πράσινο)
  pinMode(3,OUTPUT); // Φ2 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ2 Κόκκινο
  pinMode(5,OUTPUT); // Π2 Πράσινο ( Φανάρι Πεζών 2 Πράσινο)
  pinMode(6,OUTPUT); // Π2 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...

}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
   RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
   RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
   RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
   RemoteControlLeftRightOperation_Phase_4();
  }




  if (inChar == 'C'){
   RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
   RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
   RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
   RemoteControlSleepingModeOperation_Phase_2();
  }

}
```

```
/*
 * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 2 ).
 * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
 * έχουν τουλάχιστον 3 φάσεις.
 *
 * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
 *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
 *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
 *
 * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
 *
 * Η 3η φάση θα  :
 *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
 *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
 *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
 *
 * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
 * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
 * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
 * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
 */


void RemoteControlStartUpDownOperation(){
    digitalWrite(5,LOW); //( Π2 πράσινο = OFF )
    digitalWrite(6,HIGH); //( Π2 κόκκινο = ON )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
     digitalWrite(5,LOW); // ( Π2 πράσινο = OFF )
     digitalWrite(6,HIGH); // ( Π2 κόκκινο = ON )
   }
void RemoteControlUpDownOperation_Phase_2(){
   }
void RemoteControlUpDownOperation_Phase_3(){
   }
void RemoteControlUpDownOperation_Phase_4(){
   }



void RemoteControlLeftRightOperation_Phase_1(){
   }
void RemoteControlLeftRightOperation_Phase_2(){
   }
void RemoteControlLeftRightOperation_Phase_3(){
     digitalWrite(6,LOW); // ( Π2 κόκκινο = OFF )
     digitalWrite(5,HIGH); // ( Π2  πράσινο = ON )
   }
void RemoteControlLeftRightOperation_Phase_4(){
   }



void RemoteControlUpDowntoTram1_Operation_Phase_1(){
      digitalWrite(5,LOW); // ( Π2  πράσινο = OFF )
      digitalWrite(6,HIGH); //( Π2  κόκκινο = ON )
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
}
```

```
void RemoteControlUpDowntoTram2_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
      digitalWrite(6,LOW); //( Π2  κόκκινο = OFF )
      digitalWrite(5,HIGH); // ( Π2  πράσινο = ON )
}



void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
      digitalWrite(5,LOW); // ( Π2  πράσινο = OFF )
      digitalWrite(6,HIGH); //( Π2  κόκκινο = ON )
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
}



void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}


void RemoteControlTram1toUpDown_Operation_Phase_1(){
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
}
```

```
void RemoteControlTram1toLeftRightTransmition_Phase_1(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
      digitalWrite(6,LOW); //( Π2  κόκκινο = OFF )
      digitalWrite(5,HIGH); // ( Π2  πράσινο = ON )
}



void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}



void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί
σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
```

```
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_3 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_3 Software
 */

  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 3
const byte slave3 = 3;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave3);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ3 Πράσινο ( Φανάρι Αυτοκινήτων 3 Πράσινο)
  pinMode(3,OUTPUT); // Φ3 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ3 Κόκκινο
  pinMode(5,OUTPUT); // Π3 Πράσινο ( Φανάρι Πεζών 3 Πράσινο)
  pinMode(6,OUTPUT); // Π3 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...
}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



     // Ελέγχει για νέα σήματα από τον Master και εκτελεί την επιθυμητή υπό-ρουτίνα
   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
    RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
    RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
    RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
    RemoteControlLeftRightOperation_Phase_4();
  }




  if (inChar == 'C'){
    RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
    RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
    //S = RemoteControl_Sleeping_Mode
    RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
    //s = RemoteControl_Sleeping_Mode
    RemoteControlSleepingModeOperation_Phase_2();
  }

}
```

```
 /*
  * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 3 ).
  * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
  * έχουν τουλάχιστον 3 φάσεις.
  *
  * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
  *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
  *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
  *
  * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
  *
  * Η 3η φάση θα  :
  *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
  *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
  *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
  *
  * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
  * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
  * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
  * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
  */

void RemoteControlStartUpDownOperation(){
    digitalWrite(2,HIGH); //( Φ3 πράσινο = ON )
    digitalWrite(3,LOW); //( Φ3 πορτοκαλί = OFF )
    digitalWrite(4,LOW); //( Φ3 κόκκινο = OFF )
    digitalWrite(5,LOW); //( Π3 πράσινο = OFF )
    digitalWrite(6,HIGH); //( Π3 κόκκινο = ON )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
    digitalWrite(5,LOW); // ( Π3 πράσινο = OFF )
    digitalWrite(6,HIGH); // ( Π3 κόκκινο = ON )
  }
void RemoteControlUpDownOperation_Phase_2(){
  }
void RemoteControlUpDownOperation_Phase_3(){
     digitalWrite(4,LOW); // ( Φ3 κόκκινο = OFF )
     digitalWrite(2,HIGH); //  ( Φ3 πράσινο = ON )
  }
void RemoteControlUpDownOperation_Phase_4(){
  }




void RemoteControlLeftRightOperation_Phase_1(){
  }
void RemoteControlLeftRightOperation_Phase_2(){
      digitalWrite(2,LOW); // ( Φ3 πράσινο = OFF )
      digitalWrite(3,HIGH); // ( Φ3 πορτοκαλί = ON )
  }
void RemoteControlLeftRightOperation_Phase_3(){
      digitalWrite(3,LOW); // ( Φ3 πορτοκαλί = OFF )
      digitalWrite(4,HIGH); // ( Φ3 κόκκινο = ON )
      digitalWrite(6,LOW); // ( Π3 κόκκινο = OFF )
      digitalWrite(5,HIGH); // ( Π3  πράσινο = ON )
  }
void RemoteControlLeftRightOperation_Phase_4(){
  }
```

```
void RemoteControlUpDowntoTram1_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
        digitalWrite(2,LOW); //(  Φ3 πράσινο = OFF )
        digitalWrite(3,HIGH); // ( Φ3  πορτοκαλί = ON )
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
        digitalWrite(3,LOW); // ( Φ3  πορτοκαλί = OFF )
        digitalWrite(4,HIGH); //( Φ3  κόκκινο = ON )
        digitalWrite(6,LOW); //(  Π3 κόκκινο = OFF )
        digitalWrite(5,HIGH); // ( Π3  πράσινο = ON )
}




void RemoteControlUpDowntoTram2_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
        digitalWrite(2,LOW); //(  Φ3 πράσινο = OFF )
        digitalWrite(3,HIGH); // ( Φ3  πορτοκαλί = ON )
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
        digitalWrite(3,LOW); // ( Φ3  πορτοκαλί = OFF )
        digitalWrite(4,HIGH); //( Φ3  κόκκινο = ON )
        digitalWrite(6,LOW); //(  Π3 κόκκινο = OFF )
        digitalWrite(5,HIGH); // ( Π3  πράσινο = ON )
}




void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
}
```

```
void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}


void RemoteControlTram1toUpDown_Operation_Phase_1(){
        digitalWrite(5,LOW); // ( Π3  πράσινο = OFF )
        digitalWrite(6,HIGH); //(  Π3 κόκκινο = ON )
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
        digitalWrite(4,LOW); // ( Φ3  κόκκινο = OFF )
        digitalWrite(2,HIGH); //( Φ3  πράσινο = ON )
}



void RemoteControlTram1toLeftRightTransmition_Phase_1(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
}
```

```
void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}



void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_4 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_4 Software
 */

  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 4
const byte slave4 = 4;

void setup()
{
    // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave4);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ4 Πράσινο ( Φανάρι Αυτοκινήτων 4 Πράσινο)
  pinMode(3,OUTPUT); // Φ4 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ4 Κόκκινο
  pinMode(5,OUTPUT); // Π4 Πράσινο ( Φανάρι Πεζών 4 Πράσινο)
  pinMode(6,OUTPUT); // Π4 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...
}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();


     // Ελέγχει για νέα σήματα από τον Master και εκτελεί την επιθυμητή υπό-ρουτίνα
   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
    RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
    RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
    RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
    RemoteControlLeftRightOperation_Phase_4();
  }



  if (inChar == 'H'){
    RemoteControlWalkOperation_Phase_1();
  }
  if (inChar == 'h'){
    RemoteControlWalkOperation_Phase_2();
  }
  if (inChar == 'J'){
    RemoteControlWalkOperation_Phase_3();
  }
  if (inChar == 'j'){
    RemoteControlWalkOperation_Phase_4();
  }



  if (inChar == 'C'){
    RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
    RemoteControlLightsCheckOperation_Phase_2();
  }
```

```
  if (inChar == 'S'){
    RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
    RemoteControlSleepingModeOperation_Phase_2();
  }
}
```

```
/*
 * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 4 ).
 * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
 * έχουν τουλάχιστον 3 φάσεις.
 *
 * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
 *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
 *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
 *
 * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
 *
 * Η 3η φάση θα  :
 *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
 *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
 *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
 *
 * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
 * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
 * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
 * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
 */
```

```
void RemoteControlStartUpDownOperation(){
     digitalWrite(2,HIGH); //( Φ4 πράσινο = ON )
     digitalWrite(3,LOW); //( Φ4 πορτοκαλί = OFF )
     digitalWrite(4,LOW); //( Φ4 κόκκινο = OFF )
     digitalWrite(5,LOW); //( Π4 πράσινο = OFF )
     digitalWrite(6,HIGH); //( Π4 κόκκινο = ON )
}
void RemoteControlStartLeftRightOperation(){
}

void RemoteControlUpDownOperation_Phase_1(){
      digitalWrite(5,LOW); // ( Π4 πράσινο = OFF )
      digitalWrite(6,HIGH); // ( Π4 κόκκινο = ON )
   }
void RemoteControlUpDownOperation_Phase_2(){
   }
void RemoteControlUpDownOperation_Phase_3(){
     digitalWrite(4,LOW); //  ( Φ4 κόκκινο = OFF )
     digitalWrite(2,HIGH); //  ( Φ4 πράσινο = ON )
   }
void RemoteControlUpDownOperation_Phase_4(){
   }


void RemoteControlLeftRightOperation_Phase_1(){
      digitalWrite(5,LOW); // ( Π4 πράσινο = OFF )
      digitalWrite(6,HIGH); // ( Π4 κόκκινο = ON )
     digitalWrite(4,LOW); //  ( Φ4 κόκκινο = OFF )
     digitalWrite(2,HIGH); //  ( Φ4 πράσινο = ON )
     digitalWrite(3,LOW); //  ( Φ4 πορτοκαλί = OFF )
   }
void RemoteControlLeftRightOperation_Phase_2(){
   }
void RemoteControlLeftRightOperation_Phase_3(){
   }
void RemoteControlLeftRightOperation_Phase_4(){
   }
```

```
void RemoteControlUpDowntoTram1_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
        digitalWrite(2,LOW); //( Φ4  πράσινο = OFF )
        digitalWrite(3,HIGH); //( Φ4  πορτοκαλί = ON )
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
        digitalWrite(3,LOW); // ( Φ4  πορτοκαλί = OFF )
        digitalWrite(4,HIGH); //( Φ4  κόκκινο = ON )
        digitalWrite(5,HIGH); //(  Π4  πράσινο = ON )
        digitalWrite(6,LOW); //( Π4  κόκκινο = OFF )
}



void RemoteControlUpDowntoTram2_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
}



void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
        digitalWrite(2,LOW); //( Φ4  πράσινο = OFF )
        digitalWrite(3,HIGH); //( Φ4  πορτοκαλί = ON )
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
        digitalWrite(3,LOW); // ( Φ4  πορτοκαλί = OFF )
        digitalWrite(4,HIGH); //( Φ4  κόκκινο = ON )
        digitalWrite(5,HIGH); //( Π4  πράσινο = ON )
        digitalWrite(6,LOW); //( Π4  κόκκινο = OFF )
}
```

```
void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}




void RemoteControlTram1toUpDown_Operation_Phase_1(){
        digitalWrite(5,LOW); //(  Π4  πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π4  κόκκινο = ON )
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
        digitalWrite(4,LOW); //( Φ4  κόκκινο = OFF )
        digitalWrite(2,HIGH); // ( Φ4  πράσινο = ON )
}




void RemoteControlTram1toLeftRightTransmition_Phase_1(){
        digitalWrite(5,LOW); //(  Π4  πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π4  κόκκινο = ON )
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
        digitalWrite(4,LOW); //( Φ4  κόκκινο = OFF )
        digitalWrite(2,HIGH); // ( Φ4  πράσινο = ON )
}
```

```
void RemoteControlWalkOperation_Phase_1(){
        digitalWrite(2,LOW); //( Φ4  πράσινο = OFF )
        digitalWrite(3,HIGH); // ( Φ4  πορτοκαλί = ON )
   }
void RemoteControlWalkOperation_Phase_2(){
        digitalWrite(3,LOW); // ( Φ4  πορτοκαλί = OFF )
        digitalWrite(4,HIGH); //( Φ4  κόκκινο = ON )
        digitalWrite(6,LOW); //( Π4  κόκκινο = OFF )
        digitalWrite(5,HIGH); //(  Π4  πράσινο = ON )
   }
void RemoteControlWalkOperation_Phase_3(){
        digitalWrite(5,LOW); //(  Π4  πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π4  κόκκινο = ON )
   }
void RemoteControlWalkOperation_Phase_4(){
        digitalWrite(4,LOW); //( Φ4  κόκκινο = OFF )
        digitalWrite(2,HIGH); // ( Φ4  πράσινο = ON )
   }




void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}
```

```
void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_5 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_5 Software
 */


  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 5
const byte slave5 = 5;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave5);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ5 Πράσινο ( Φανάρι Αυτοκινήτων 5 Πράσινο)
  pinMode(3,OUTPUT); // Φ5 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ5 Κόκκινο
  pinMode(5,OUTPUT); // Π5 Πράσινο ( Φανάρι Πεζών 5 Πράσινο)
  pinMode(6,OUTPUT); // Π5 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...
}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
    RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
    RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
    RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
    RemoteControlLeftRightOperation_Phase_4();
  }









  if (inChar == 'C'){
    RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
    RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
    RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
    RemoteControlSleepingModeOperation_Phase_2();
  }

}
```

```
 /*
  * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 5 ).
  * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
  * έχουν τουλάχιστον 3 φάσεις.
  *
  * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
  *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
  *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
  *
  * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
  *
  * Η 3η φάση θα  :
  *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
  *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
  *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
  *
  * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
  * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
  * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
  * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
  */

void RemoteControlStartUpDownOperation(){
    digitalWrite(2,LOW); //( Φ5 πράσινο = OFF )
    digitalWrite(3,LOW); //( Φ5 πορτοκαλί = OFF )
    digitalWrite(4,HIGH); //( Φ5 κόκκινο = ON )
    digitalWrite(5,HIGH); //( Π5 πράσινο = ON )
    digitalWrite(6,LOW); //( Π5 κόκκινο = OFF )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
   }
void RemoteControlUpDownOperation_Phase_2(){
    digitalWrite(2,LOW);  // ( Φ5 πράσινο = OFF )
    digitalWrite(3,HIGH); // ( Φ5 πορτοκαλί = ON )
   }
void RemoteControlUpDownOperation_Phase_3(){
      digitalWrite(3,LOW); // ( Φ5 πορτοκαλί = OFF )
      digitalWrite(4,HIGH); // ( Φ5 κόκκινο = ON )
      digitalWrite(6,LOW); // ( Π5  κόκκινο = OFF )
      digitalWrite(5,HIGH); // ( Π5  πράσινο = ON )
   }
void RemoteControlUpDownOperation_Phase_4(){
   }


void RemoteControlLeftRightOperation_Phase_1(){
      digitalWrite(5,LOW); // ( Π5 πράσινο = OFF )
      digitalWrite(6,HIGH); // ( Π5 κόκκινο = ON )
   }
void RemoteControlLeftRightOperation_Phase_2(){
   }
void RemoteControlLeftRightOperation_Phase_3(){
       digitalWrite(4,LOW); // ( Φ5 κόκκινο = OFF )
       digitalWrite(2,HIGH); // ( Φ5  πράσινο = ON )
   }
void RemoteControlLeftRightOperation_Phase_4(){
   }


void RemoteControlUpDowntoTram1_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
}
```

```
void RemoteControlUpDowntoTram2_Operation_Phase_1(){
        digitalWrite(5,LOW); //( Π5 πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π5 κόκκινο = ON )
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
        digitalWrite(4,LOW); //( Φ5 κόκκινο = OFF )
        digitalWrite(2,HIGH); //( Φ5 πράσινο = ON )
}




void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
    digitalWrite(2,LOW);  // ( Φ5 πράσινο = OFF )
    digitalWrite(3,HIGH); // ( Φ5 πορτοκαλί = ON )
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
     digitalWrite(3,LOW); // ( Φ5 πορτοκαλί = OFF )
     digitalWrite(4,HIGH); // ( Φ5 κόκκινο = ON )
     digitalWrite(6,LOW); // ( Π5  κόκκινο = OFF )
     digitalWrite(5,HIGH); // ( Π5  πράσινο = ON )
}




void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}
```

```
void RemoteControlTram1toUpDown_Operation_Phase_1(){
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
}



void RemoteControlTram1toLeftRightTransmition_Phase_1(){
        digitalWrite(5,LOW); //( Π5 πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π5 κόκκινο = ON )
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
        digitalWrite(4,LOW); //( Φ5 κόκκινο = OFF )
        digitalWrite(2,HIGH); //( Φ5 πράσινο = ON )
}



void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}
```

```
void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_6 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_6 Software
 */

  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 6
const byte slave6 = 6;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave6);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ6 Πράσινο ( Φανάρι Αυτοκινήτων 6 Πράσινο)
  pinMode(3,OUTPUT); // Φ6 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ6 Κόκκινο
  pinMode(5,OUTPUT); // Π6 Πράσινο ( Φανάρι Πεζών 6 Πράσινο)
  pinMode(6,OUTPUT); // Π6 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...
}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



     // Ελέγχει για νέα σήματα από τον Master και εκτελεί την επιθυμητή υπό-ρουτίνα
   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}



if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
    RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
    RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
    RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
    RemoteControlLeftRightOperation_Phase_4();
  }




  if (inChar == 'C'){
    RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
    RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
    RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
    RemoteControlSleepingModeOperation_Phase_2();
  }

}
```

```
 /*
  * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 6 ).
  * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
  * έχουν τουλάχιστον 3 φάσεις.
  *
  * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
  *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
  *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
  *
  * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
  *
  * Η 3η φάση θα  :
  *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
  *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
  *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
  *
  * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
  * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
  * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
  * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
  */


void RemoteControlStartUpDownOperation(){
    digitalWrite(5,LOW); //( Π6 πράσινο = OFF )
    digitalWrite(6,HIGH); //( Π6 κόκκινο = ON )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
      digitalWrite(5,LOW); // ( Π6 πράσινο = OFF )
      digitalWrite(6,HIGH); // ( Π6 κόκκινο = ON )
   }
void RemoteControlUpDownOperation_Phase_2(){
   }
void RemoteControlUpDownOperation_Phase_3(){
   }
void RemoteControlUpDownOperation_Phase_4(){
   }



void RemoteControlLeftRightOperation_Phase_1(){
   }
void RemoteControlLeftRightOperation_Phase_2(){
   }
void RemoteControlLeftRightOperation_Phase_3(){
      digitalWrite(48,LOW); // ( Π6 κόκκινο = OFF )
      digitalWrite(47,HIGH); // ( Π6  πράσινο = ON )
   }
void RemoteControlLeftRightOperation_Phase_4(){
   }



void RemoteControlUpDowntoTram1_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
        digitalWrite(6,LOW); //( Π6  κόκκινο = OFF )
        digitalWrite(5,HIGH); //( Π6  πράσινο = ON )
}
```

```
void RemoteControlUpDowntoTram2_Operation_Phase_1(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
        digitalWrite(6,LOW); //( Π6  κόκκινο = OFF )
        digitalWrite(5,HIGH); //( Π6  πράσινο = ON )
}


void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
}


void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}



void RemoteControlTram1toUpDown_Operation_Phase_1(){
      digitalWrite(5,LOW); // ( Π6 πράσινο = OFF )
      digitalWrite(6,HIGH); // ( Π6 κόκκινο = ON )
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
}
```

```
void RemoteControlTram1toLeftRightTransmition_Phase_1(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
}

void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}

void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί
σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
}
```

SLAVE_7 SOFTWARE

```
/*
 * TRAFIC LIGHTS REMOTE CONTROL Project by Kostas Argyriadis
 *  Student Of PIRAEUS UNIVERSITY OF APPLIED SCIENCES
 *  Slave_7 Software
 */



  // Συμπεριλαμβάνει την Απαραίτητη Βιβλιοθήκη για την επικοινωνία μεταξύ Master - Slave
#include <Wire.h>

  // Ορίζει την Θέση του Slave 7
const byte slave7 = 7;

void setup()
{
  // Ξεκινά την επικοινωνία με τον Master
  Wire.begin(slave7);
  Wire.onReceive(receiveEvent);

    // Ορίζω τα INPUT και τα OUTPUT της συσκευής μου.

  pinMode(2,OUTPUT); // Φ7 Πράσινο ( Φανάρι Αυτοκινήτων 7 Πράσινο)
  pinMode(3,OUTPUT); // Φ7 Πορτοκαλί
  pinMode(4,OUTPUT); // Φ7 Κόκκινο
  pinMode(5,OUTPUT); // Π7 Πράσινο ( Φανάρι Πεζών 7 Πράσινο)
  pinMode(6,OUTPUT); // Π7 Κόκκινο
  pinMode(7,OUTPUT); //...
  pinMode(8,OUTPUT); //...
  pinMode(9,OUTPUT); //...
  pinMode(10,OUTPUT); //...  Κενές Θέσεις
  pinMode(11,OUTPUT); //...
  pinMode(12,OUTPUT); //...
  pinMode(13,OUTPUT); //...
}
```

```
void loop()
{
}

   // Υπό-ρουτίνα για την συνεχή αναγνώριση των σημάτων μεταξύ των συσκευών.
void receiveEvent(int howMany)
{
  char inChar;

   inChar = Wire.read();



     // Ελέγχει για νέα σήματα από τον Master και εκτελεί την επιθυμητή υπό-ρουτίνα
   if (inChar == 'K'){
    RemoteControlTram1toLeftRightTransmition_Phase_1();
   }
   if (inChar == 'k'){
    RemoteControlTram1toLeftRightTransmition_Phase_2();
   }
   if (inChar == 'L'){
    RemoteControlTram1toLeftRightTransmition_Phase_3();
   }




   if (inChar == 'O'){
    RemoteControlTram1toUpDown_Operation_Phase_1();
   }
   if (inChar == 'o'){
    RemoteControlTram1toUpDown_Operation_Phase_2();
   }
   if (inChar == 'P'){
    RemoteControlTram1toUpDown_Operation_Phase_3();
   }
```

```
if (inChar == 'T'){
  RemoteControlUpDowntoTram1_Operation_Phase_1();
}
if (inChar == 't'){
  RemoteControlUpDowntoTram1_Operation_Phase_2();
}
if (inChar == 'Y'){
  RemoteControlUpDowntoTram1_Operation_Phase_3();
}



if (inChar == 'F'){
  RemoteControlUpDowntoTram2_Operation_Phase_1();
}
if (inChar == 'f'){
  RemoteControlUpDowntoTram2_Operation_Phase_2();
}
if (inChar == 'G'){
  RemoteControlUpDowntoTram2_Operation_Phase_3();
}



if (inChar == 'E'){
  RemoteControlLeftRighttoTram1_Operation_Phase_1();
}
if (inChar == 'e'){
  RemoteControlLeftRighttoTram1_Operation_Phase_2();
}
if (inChar == 'D'){
  RemoteControlLeftRighttoTram1_Operation_Phase_3();
}
```

```
if (inChar == 'U'){
  RemoteControlLeftRighttoTram2_Operation_Phase_1();
}
if (inChar == 'u'){
  RemoteControlLeftRighttoTram2_Operation_Phase_2();
}
if (inChar == 'I'){
  RemoteControlLeftRighttoTram2_Operation_Phase_3();
}



if (inChar == 'R'){
  RemoteControlStartUpDownOperation();
}
if (inChar == 'r'){
  RemoteControlStartLeftRightOperation();
}


if (inChar == 'Q'){
  RemoteControlUpDownOperation_Phase_1();
}
if (inChar == 'q'){
  RemoteControlUpDownOperation_Phase_2();
}
if (inChar == 'W'){
  RemoteControlUpDownOperation_Phase_3();
}
if (inChar == 'w'){
  RemoteControlUpDownOperation_Phase_4();
}
```

```
  if (inChar == 'A'){
    RemoteControlLeftRightOperation_Phase_1();
  }
  if (inChar == 'a'){
    RemoteControlLeftRightOperation_Phase_2();
  }
  if (inChar == 'Z'){
    RemoteControlLeftRightOperation_Phase_3();
  }
  if (inChar == 'z'){
    RemoteControlLeftRightOperation_Phase_4();
  }



  if (inChar == 'C'){
    RemoteControlLightsCheckOperation_Phase_1();
  }
  if (inChar == 'c'){
    RemoteControlLightsCheckOperation_Phase_2();
  }



  if (inChar == 'S'){
    RemoteControlSleepingModeOperation_Phase_1();
  }
  if (inChar == 's'){
    RemoteControlSleepingModeOperation_Phase_2();
  }

}
```

```
 /*
  * Οι παρακάτω υπό-ρουτίνες αντιστοιχούν στις διάφορες φάσεις που μπορεί να χρειαστεί να βρεθεί το συγκεκριμένο φανάρι ( 7 ).
  * Όλες οι αλλαγές του συστήματος εκτός από την εκκίνηση (StartOperation) το  Sleeping Mode και το Lights Check
  * έχουν τουλάχιστον 3 φάσεις.
  *
  * Η 1η φάση θα κλείνει πάντα τα φανάρια των πεζών που είναι πράσινα και θα τα κάνει κόκκινα.
  *  (αυτό θα γίνεται γιατί πρέπει να δημιουργηθεί χρόνος και για τον τελευταίο πεζό να διασχίσει την διάβαση με ασφάλεια,
  *   προτού ανοίξει το φανάρι των αυτοκινήτων πράσινο.)
  *
  * Η 2η φάση θα κλείνει τα πράσινα φανάρια των αυτοκινήτων και θα ανάβει τα πορτοκαλί.
  *
  * Η 3η φάση θα  :
  *   Σβήνει τα πορτοκαλί φανάρια των αυτοκινήτων και θα ανάβει τα κόκκινα.
  *   Σβήνει τα κόκκινα φανάρια των πεζών και θα τα κάνει πράσινα.
  *   Σβήνει τα κόκκινα φανάρια των αυτοκινήτων και θα τα κάνει πράσινα.
  *
  * Κάθε φανάρι(κάθε λαμπτήρας) είναι αυτόνομο ως προς την λειτουργία του.
  * Αυτό σημαίνει πως ανάλογα την μεταβολή των φαναριών στην διασταύρωση, υπάρχει περίπτωση
  * σε κάποια φάση ή ακόμη και σε κάποια ολόκληρη λειτουργία, το συγκεκριμένο φανάρι να μην χρειάζεται
  * να κάνει κάποια αλλαγή στην κατάσταση του οπότε δεν θα έχει να εκτελέσει και κάποια εντολή.
  */

void RemoteControlStartUpDownOperation(){
    digitalWrite(2,LOW); //( Φ7 πράσινο = OFF )
    digitalWrite(3,LOW); //( Φ7 πορτοκαλί = OFF )
    digitalWrite(4,HIGH); //( Φ7 κόκκινο = ON )
    digitalWrite(5,HIGH); //( Π7 πράσινο = ON )
    digitalWrite(6,LOW); //( Π7 κόκκινο = OFF )
}

void RemoteControlStartLeftRightOperation(){
}
```

```
void RemoteControlUpDownOperation_Phase_1(){
  }
void RemoteControlUpDownOperation_Phase_2(){
    digitalWrite(2,LOW);  // ( Φ7 πράσινο = OFF )
    digitalWrite(3,HIGH); // ( Φ7 πορτοκαλί = ON )
  }
void RemoteControlUpDownOperation_Phase_3(){
    digitalWrite(3,LOW); // ( Φ7 πορτοκαλί = OFF )
    digitalWrite(4,HIGH); // ( Φ7 κόκκινο = ON )
    digitalWrite(6,LOW); // ( Π7  κόκκινο = OFF )
    digitalWrite(5,HIGH); // ( Π7  πράσινο = ON )
  }
void RemoteControlUpDownOperation_Phase_4(){
  }



void RemoteControlLeftRightOperation_Phase_1(){
    digitalWrite(5,LOW); // ( Π7 πράσινο = OFF )
    digitalWrite(6,HIGH); // ( Π7 κόκκινο = ON )
  }
void RemoteControlLeftRightOperation_Phase_2(){
  }
void RemoteControlLeftRightOperation_Phase_3(){
    digitalWrite(4,LOW); // ( Φ7 κόκκινο = OFF )
    digitalWrite(2,HIGH); // ( Φ7  πράσινο = ON )
  }
void RemoteControlLeftRightOperation_Phase_4(){
  }
```

```
void RemoteControlUpDowntoTram1_Operation_Phase_1(){
        digitalWrite(5,LOW); //( Π7 πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π7 κόκκινο = ON )
}
void RemoteControlUpDowntoTram1_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram1_Operation_Phase_3(){
        digitalWrite(4,LOW); //( Φ7  κόκκινο = OFF )
        digitalWrite(2,HIGH); //( Φ7  πράσινο = ON )
}




void RemoteControlUpDowntoTram2_Operation_Phase_1(){
        digitalWrite(5,LOW); //( Π7 πράσινο = OFF )
        digitalWrite(6,HIGH); //( Π7 κόκκινο = ON )
}
void RemoteControlUpDowntoTram2_Operation_Phase_2(){
}
void RemoteControlUpDowntoTram2_Operation_Phase_3(){
        digitalWrite(4,LOW); //( Φ7  κόκκινο = OFF )
        digitalWrite(2,HIGH); //( Φ7  πράσινο = ON )
}



void RemoteControlLeftRighttoTram1_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram1_Operation_Phase_3(){
}
```

```
void RemoteControlLeftRighttoTram2_Operation_Phase_1(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_2(){
}
void RemoteControlLeftRighttoTram2_Operation_Phase_3(){
}



void RemoteControlTram1toUpDown_Operation_Phase_1(){
}
void RemoteControlTram1toUpDown_Operation_Phase_2(){
      digitalWrite(2,LOW);  // ( Φ7 πράσινο = OFF )
      digitalWrite(3,HIGH); // ( Φ7 πορτοκαλί = ON )
}
void RemoteControlTram1toUpDown_Operation_Phase_3(){
      digitalWrite(3,LOW); // ( Φ7 πορτοκαλί = OFF )
      digitalWrite(4,HIGH); // ( Φ7 κόκκινο = ON )
      digitalWrite(6,LOW); // ( Π7  κόκκινο = OFF )
      digitalWrite(5,HIGH); // ( Π7  πράσινο = ON )
}



void RemoteControlTram1toLeftRightTransmition_Phase_1(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_2(){
}
void RemoteControlTram1toLeftRightTransmition_Phase_3(){
}
```

```
void RemoteControlSleepingModeOperation_Phase_1(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(2,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);

  digitalWrite(3,HIGH);
}
void RemoteControlSleepingModeOperation_Phase_2(){
  // Λειτουργεί με μόνιμα παλλόμενο πορτοκαλί φανάρι.
  digitalWrite(3,LOW);
}



void RemoteControlLightsCheckOperation_Phase_1(){
  // Ανάβει όλα τα φανάρια που διαχειρίζεται για να δει εάν είναι κάτι καμένο η δεν λειτουργεί σωστά.
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
}
void RemoteControlLightsCheckOperation_Phase_2(){
  // Σβήνει όλα τα φανάρια που διαχειρίζεται.
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW)
```

5 . Το Κύκλωμα (Hardware)

Όσον αφορά το κομμάτι του κυκλώματος η πτυχιακή αυτή άσκηση είναι αρκετά απλή . Έχει μεν πολλά φανάρια (LEDs) διακόπτες (Buttons) αντιστάσεις, τερματικά και μίκρο-επεξεργαστές αλλά ο τρόπος σύνδεσης του είναι παρόμοιος σε όλους τους μικροεπεξεργαστές και τα φανάρια τους.

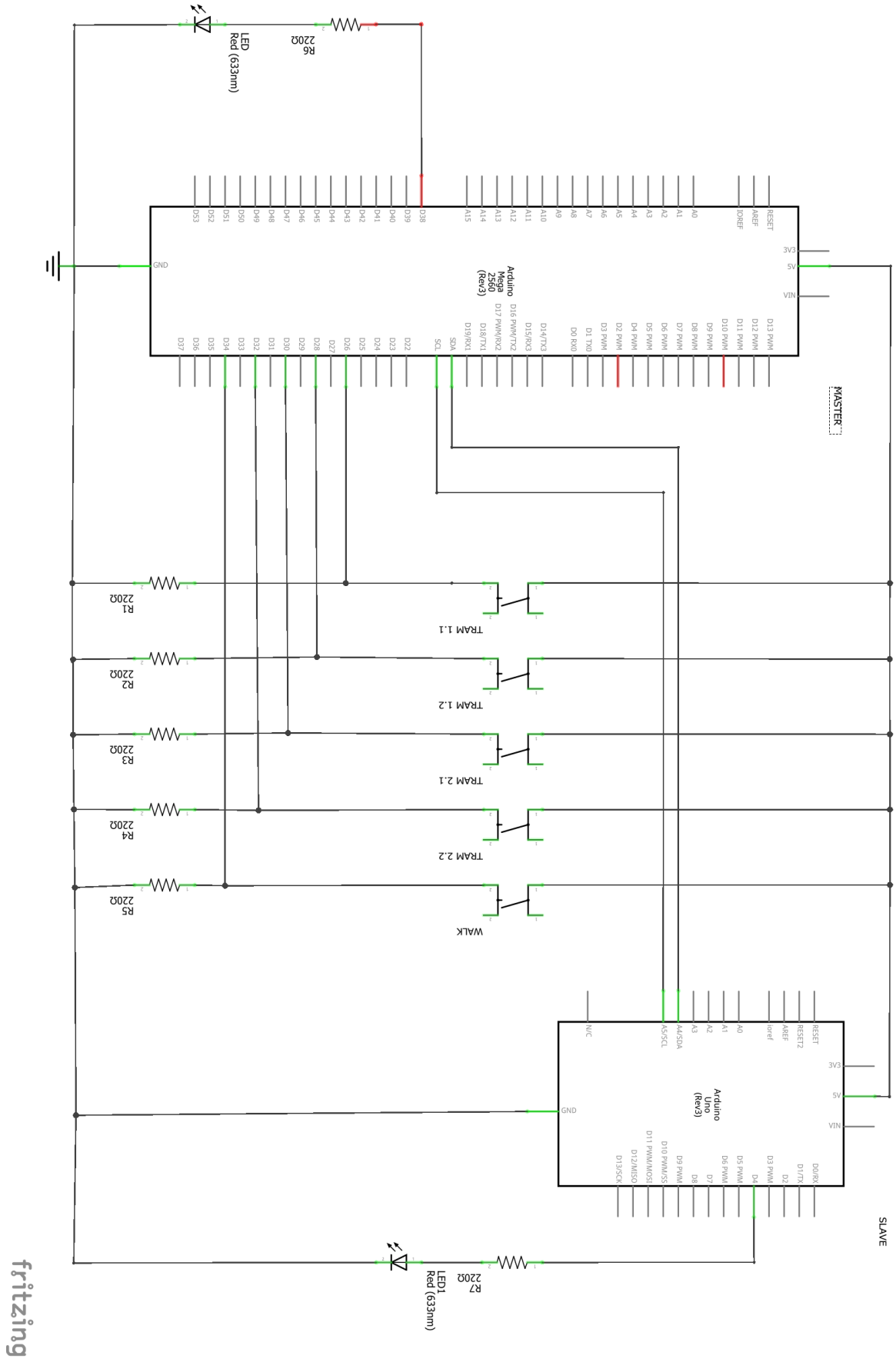
Οπότε στα παρακάτω σχέδια θα δούμε :

1. Τον τρόπο σύνδεσης για την σειριακή επικοινωνία μεταξύ δύο επεξεργαστών
2. Την κοινή τροφοδοσία τους
3. Τις απαραίτητες 10kΩ Pull Down αντιστάσεις στα Button
4. Τις αντιστάσεις 220Ω στα LEDs
5. Τον χρωματικό κώδικα που χρησιμοποιήθηκε για την αναγνώριση των γραμμών
   και
6. Όλα τα pins των μίκρο-επεξεργαστών που χρησιμοποιήθηκαν και την αντιστοίχηση τους με τα φανάρια , αισθητήρια της διασταύρωσης.

Επίσης θα δούμε πώς μπορούμε πολύ απλά με ένα relay να φέρουμε τις χαμηλές τάσεις λειτουργίας των 5V των μίκρο-επεξεργαστών πιο κοντά στις πραγματικές καταναλώσεις των 230V.

Για λόγους εξοικονόμησης χώρου αλλά και χαρτιού δεν θα αποτυπώσουμε τα κυκλώματα όλων των Led και όλων των μικροεπεξεργαστών αλλά θα δούμε παραδειγματικά τον Master με έναν Slave.

5 . Το Κύκλωμα (Hardware)



WIRES COLOR CODE	
Color	Meaning
TOP	
White	Tram Lights
Red	Car Trafic Lights
Green	CrossWalk Lights
Brown	Sensors
BOTTOM	
White	1
Yellow	2
Green	3
Blue	4
Red	5
White - Yellow	6
White - Green	7

Η κάθε γραμμή θα σηματοδοτείτε από χρωματιστά καλώδια και "δαχτυλίδια" . Π.χ. Εάν ψάχνω το πορτοκαλί φανάρι αυτοκινήτων Φ4 , περιμένω να δω πορτοκαλί καλώδιο με κόκκινο μεγάλο επάνω δαχτυλίδι και μπλε μικρό κάτω δαχτυλίδι.

5 . Το Κύκλωμα (Hardware)

Το κύκλωμα της πτυχιακής

ARDUINO MEGA "MASTER" PINS				
Inputs	PIN number	Wire Ending Collor		
		TOP	END	
RemoteControlUpDown	2	-	-	-
RemoteControlLeftRight	3	-	-	-
RemoteControlTram1	4	-	-	-
RemoteControlTram2	5	-	-	-
RemoteControlWalk1	6	-	-	-
RemoteControlLightsCheck	7	-	-	-
RemoteControlStartUpDownCheck	8	-	-	-
RemoteControlSleepingMode	9	-	-	-
-	10	-	-	-

Sensors	PIN number	Wire Ending Collor		
		TOP	END	
Z1	22			-
Z2	24			-
Tram1.1Sensor	26			-
Tram1.2Sensor	28			-
Tram2.1Sensor	30			
Tram2.2Sensor	32			
WalkSensor	34			-

Outputs LED	PIN number	Wire Ending Collor		
		TOP	END	
Tram1 (Φανάρι Τράμ 1 Μπλέ)	36			-
Tram1 (Φανάρι Τράμ 1 Πορτοκαλί)	38			-
Tram2 (Φανάρι Τράμ 2 Μπλέ)	40			-
Tram2 (Φανάρι Τράμ 2 Πορτοκαλί)	42			-

ARDUINO UNO "SLAVE 1" PINS				
Outputs LED	PIN number	Wire Ending Collor		
		TOP	END	
Φ1 (Φανάρι Αυτοκινήτων Πράσσινο)	2			-
Φ1 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			-
Φ1 (Φανάρι Αυτοκινήτων Κόκκινο)	4			-
Π1 (Φανάρι Πεζών Πράσινο)	5			-
Π1 (Φανάρι Πεζών Κόκκινο)	6			-

ARDUINO UNO "SLAVE 2" PINS				
Outputs LED	PIN number	Wire Ending Collor		
		TOP	END	
Φ2 (Φανάρι Αυτοκινήτων Πράσσινο)	2			-
Φ2 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			-
Φ2 (Φανάρι Αυτοκινήτων Κόκκινο)	4			-
Π2 (Φανάρι Πεζών Πράσινο)	5			-
Π2 (Φανάρι Πεζών Κόκκινο)	6			-

ARDUINO UNO "SLAVE 3" PINS				
Outputs LED	PIN number	Wire Ending Collor		
		TOP	END	
Φ3 (Φανάρι Αυτοκινήτων Πράσσινο)	2			-
Φ3 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			-
Φ3 (Φανάρι Αυτοκινήτων Κόκκινο)	4			-
Π3 (Φανάρι Πεζών Πράσινο)	5			-
Π3 (Φανάρι Πεζών Κόκκινο)	6			-

ARDUINO UNO "SLAVE 4" PINS				
Outputs LED	PIN number	Wire Collor		
		TOP	END	
Φ4 (Φανάρι Αυτοκινήτων Πράσσινο)	2			-
Φ4 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			-
Φ4 (Φανάρι Αυτοκινήτων Κόκκινο)	4			-
Π4 (Φανάρι Πεζών Πράσινο)	5			-
Π4 (Φανάρι Πεζών Κόκκινο)	6			-

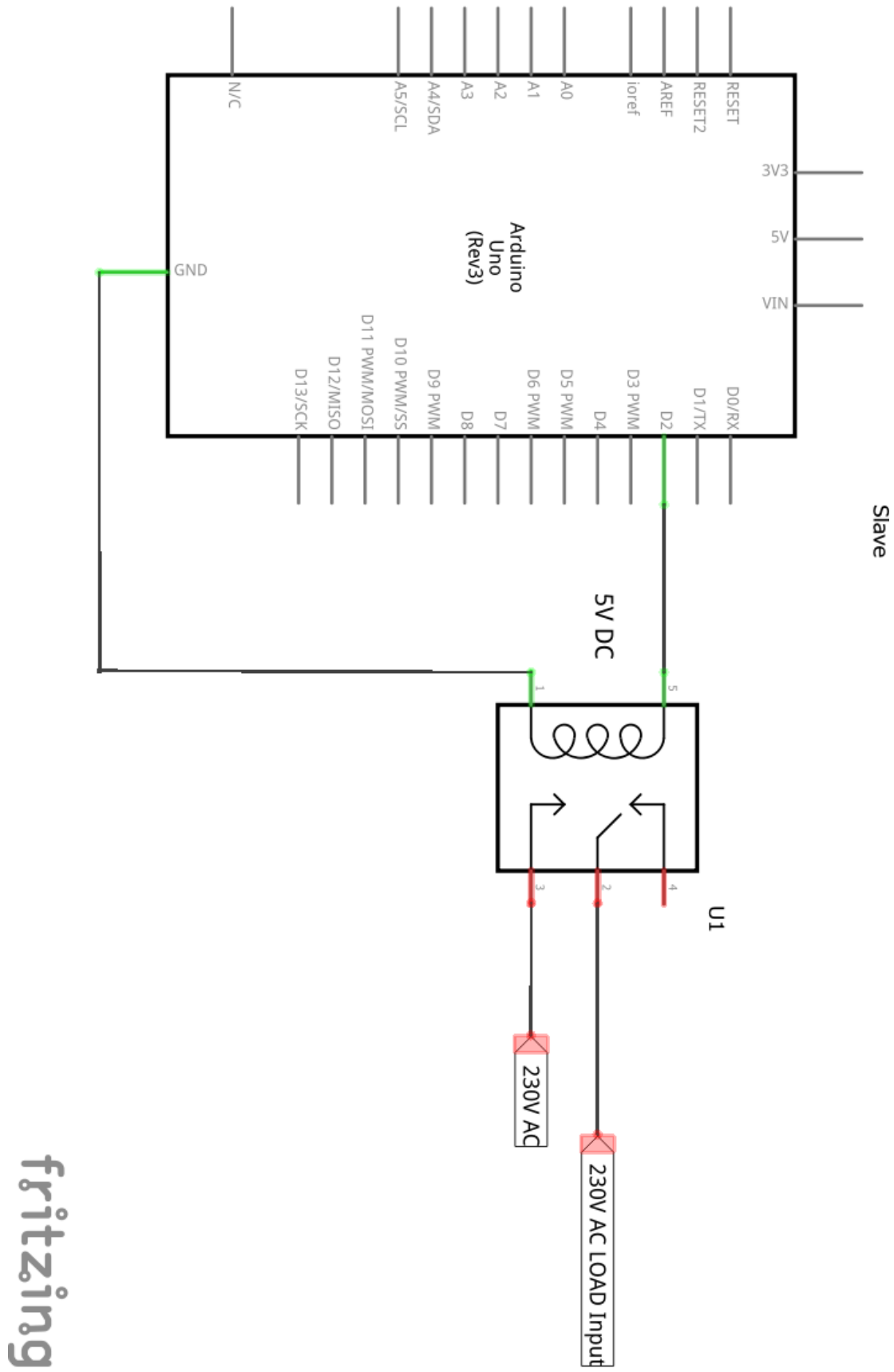
ARDUINO UNO "SLAVE 5" PINS				
Outputs LED	PIN number	Wire Collor		
		TOP	END	
Φ5 (Φανάρι Αυτοκινήτων Πράσσινο)	2			-
Φ5 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			-
Φ5 (Φανάρι Αυτοκινήτων Κόκκινο)	4			-
Π5 (Φανάρι Πεζών Πράσινο)	5			-
Π5 (Φανάρι Πεζών Κόκκινο)	6			-

ARDUINO UNO "SLAVE 6" PINS				
Outputs LED	PIN number	Wire Collor		
		TOP	END	
Φ6 (Φανάρι Αυτοκινήτων Πράσσινο)	2			
Φ6 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			
Φ6 (Φανάρι Αυτοκινήτων Κόκκινο)	4			
Π6 (Φανάρι Πεζών Πράσινο)	5			
Π6 (Φανάρι Πεζών Κόκκινο)	6			

ARDUINO UNO "SLAVE 7" PINS				
Outputs LED	PIN number	Wire Collor		
		TOP	END	
Φ7 (Φανάρι Αυτοκινήτων Πράσσινο)	2			
Φ7 (Φανάρι Αυτοκινήτων Πορτοκαλί)	3			
Φ7 (Φανάρι Αυτοκινήτων Κόκκινο)	4			
Π7 (Φανάρι Πεζών Πράσινο)	5			
Π7 (Φανάρι Πεζών Κόκκινο)	6			

5 . Το Κύκλωμα (Hardware)

Το κύκλωμα μίας πραγματικής κατανάλωσης



5 . 3 Ανακεφαλαίωση – Επιπλέον δυνατότητες λειτουργίας αυτοματισμού

Ανακεφαλαιώνοντας,
Είδαμε μέχρι τώρα :

- πως θα χαρακτηρίζαμε το σύστημα μας, είδαμε ότι είναι ένα ηλεκτρικό - ηλεκτρονικό σύστημα .
- τί είναι το Arduino , πώς συνδέεται με τον υπολογιστή μας, τι γλώσσα χρησιμοποιεί και από τι αποτελείται .
- τον τρόπο σύνδεσης και επικοινωνίας δυο ή περισσοτέρων πλακετών Arduino
- τί είναι το Blynk , πώς γίνεται η εγκατάσταση του, πως συνδέεται με τον υπολογιστή μας και πως τελικά χρησιμοποιείται στην εφαρμογή μας για τον έλεγχο της έξυπνης διασταύρωσης μας.
- ολόκληρο το πρόγραμμα μας και τα οχτώ δηλαδή προγράμματα του κάθε Μίκρο-επεξεργαστή.
- κομμάτι του κυκλώματος της μακέτας παρουσίασης αλλά και παράδειγμα σύνδεσης μιας πραγματικής κατανάλωσης.

Ας δούμε λοιπόν γιατί μπήκαμε σε τόσο κόπο . Ποιους θα βοηθήσει πραγματικά μια τέτοια εφαρμογή ; Αλλά και τι προοπτικές εξέλιξης έχει.

Η εφαρμογή μας έχει την δυνατότητα να ελέγχει σε πραγματικό χρόνο , άμεσα, πανεύκολα και χωρίς καμία καθυστέρηση μια υπαρκτή διασταύρωση από οπουδήποτε.

Κάτι τέτοιο καταλαβαίνουμε ότι θα ήταν ένα πανίσχυρο εργαλείο για τους τροχονόμους και για το κέντρο έλεγχου της τροχαίας. Καθώς εάν το συνδυάσεις με μία κάμερα στην διασταύρωση μπορείς να δουλέψεις απ το γραφείο χωρίς να καθυστερήσεις καθόλου, κάτι που θα γινόταν διαφορετικά, μέχρι να πας στην διασταύρωση και να κάνεις νοήματα με τα χέρια.

Επιπλέον είναι πολύ πιο ασφαλές για τους οδηγούς για όλους εμάς διότι πολλές φορές μπορεί να μην δουν τον τροχονόμο και να προκληθεί κάποιο ατύχημα μεταξύ των μέσων μεταφοράς. Αυτό ισχύει και για τους τροχονόμους και επιπρόσθετα οι συνθήκες εργασίας τους θα βελτιωθούν εντυπωσιακά

διότι δεν θα χρειάζεται να στέκονται στην μέση του δρόμου με όλο το καυσαέριο και εκτεθειμένοι στις ακραίες πολλές φορές για το ανθρώπινο σώμα καιρικές συνθήκες που επικρατούν (καύσωνας και ήλιος, βροχές τον χειμώνα κλπ.)

Ο τρόπος με τον οποίο συνδέονται και επικοινωνούν οι πλακέτες μας μας δίνει την δυνατότητα να ελέγχουμε ταυτόχρονα στο ίδιο σύστημα έως και 128 συσκευές όπως το Arduino. Αυτό πρακτικά σημαίνει ότι θα μπορούσαν τελικά να ελέγχουμε με έναν μάστερ η και μια διαφορετική κατανομή Masters και Slaves έως και 128 διαφορετικά φανάρια . Δηλαδή εάν κατά μέσο όρο μια διασταύρωση έχει 4 φανάρια θα μπορούσαμε τελικά να ελέγχουμε έως και 32 διασταυρώσεις.

Βέβαιά 32 διασταυρώσεις είναι πάρα πολλές και για να τις συντονίσεις όλες μαζί, θα είναι μια δύσκολη πρόκληση. Αλλά αν σκεφτούμε ότι η Λεωφ. Κηφισίας από την κηφισιά έως και το Hilton έχει περίπου 20 διασταυρώσεις δεν χρειάζεται κιόλας.

Επιπλέον το ότι δεν απαιτεί φυσική παρουσία προσώπου στην διασταύρωση εξοικονομεί ένα σημαντικό ποσό καθώς ένας υπάλληλος με την κατάλληλα σχεδιασμένη διάταξη θα μπορούσε πολύ εύκολα να χειριστεί από το κέντρο ελέγχου 5 ή και παραπάνω διασταυρώσεις καθώς ο σχεδιασμός του συστήματος δεν σου επιτρέπει να κάνεις λάθη προκαλώντας έτσι μεγαλύτερα προβλήματα.

Ένα ακόμη στοιχείο που θα μπορούσε να προστεθεί είναι η ειδοποίηση μέσω email ή ακόμα και στο κινητό ότι η διασταύρωση « έχει αρχίσει να έχει κίνηση » για την πρόληψη του προβλήματος. Αυτό θα μπορούσε να γίνει με επιδαπέδιους αισθητήρες οι οποίοι θα κλείνουν κύκλωμα όταν κάτι περνάει από πάνω τους. Και με το κατάλληλο πρόγραμμα το Arduino με την εντολή PulseIn θα εντοπίζει για πόσο χρόνο είναι κάτι σταματημένο (κλείνει κύκλωμα) και τροφοδοτεί την είσοδο του .

Επίσης, με λίγο διαφορετικό σχεδιασμό της επικοινωνίας μεταξύ των Μίκρο-επεξεργαστών , δηλαδή με μικρές αλλαγές στις εντολές του I^2C bus Communication , το σύστημα θα μπορούσε να λειτουργεί με ή χωρίς έλεγχο

από άνθρωπο και να παίρνει ανά πάση στιγμή μόνο του όλες τις αποφάσεις ενός τροχονόμου.

Κλείνοντας , με λίγα λόγια το σύστημά μας με τον τρόπο με το οποιό έχει σχεδιαστεί να λειτουργεί μας δίνει την δυνατότητα να κάνουμε ότι μπορούμε να φανταστούμε στην διασταύρωση μας , έχει πολλές ελεύθερες εισόδους και εξόδους ακόμα που μας δίνουν την δυνατότητα να προσθέσουμε φανάρια , αισθητήρες κάθε είδους , ακόμα και κάποιο μοτέρ το οποίο θα περιστρέφει την κάμερα της διασταύρωσης. Έχει την βάση να κάνει τα πάντα!

6 . Κοστολόγιο Αυτοματισμού

Ίσως το πιο βασικό πλεονέκτημα της χρήσης του Arduino είναι το μικρό του κόστος. Αυτή την στιγμή μπορεί να βρει κανείς την πλακέτα Arduino Uno από 25 ευρώ ενώ το Arduino Mega 2560 από 45 ευρώ . Βέβαια υπάρχουν και αρκετά πιο φτηνοί κλώνοι του Arduino όπως το GeekCreit UNO στα 4 ευρώ και το GeekCreit MEGA 2560 στα 12 ευρώ . Συνολικά λοιπόν για την συγκεκριμένη διασταύρωση διαθέσαμε 102 ευρώ για τους Μίκρο-επεξεργαστές.

Περίπου 20 ευρώ σε τερματικά, breadboards, αντιστάσεις και μπουτόν.

Εάν θέλαμε να την κάνουμε πραγματικότητα θα χρειαζόμασταν επίσης 35 relay modules με συνολικό κόστος 27 ευρώ.

Ένα Wi Fi shield για την πλακέτα μας κόστους 10 ευρώ.

Ένα τροφοδοτικό 9V κόστους 3 ευρώ

Επιπρόσθετο και τελευταίο κόστος είναι το κόστος των αγωγών κάτι που είναι δύσκολο να υπολογιστεί εάν δεν γίνει πραγματικότητα.

Συνολικά χωρίς τους αγωγούς σαν υλικά δεν θα κόστιζε πάνω από 100 ευρώ. (με την χρήση μόνο κλώνων)

7 . Πηγές

Κεφάλαιο 1

http://elearning.teicm.gr/file.php/318/askisi_therm.pdf
http://www.cityportal.gr/articles_det1.asp?article_id=56715
http://www.voria.gr/article/meiosi-sto-kukloforiako-tis-thessalonikis-me-ta-exupna-fanaria
http://www.compass4d.eu/en/about/about-compass4d-greek/
http://www.enet.gr/?i=news.el.article&id=168831
https://www.autonomous.gr/artificial-intelligence-traffic-lights-surtrac-7612/
http://www.ethnos.gr/epistimi/arthro/perissotero_epikindynoi_oi_marathonioi_gia_osous_den_agonizontai-65112466/
http://www.videoman.gr/98803
http://www.newsit.gr/default.php?pname=Article&art_id=251282&catid=43
http://www.drive.gr/news/tehnologia/exypna-fanaria-apofasizoyn-mona-toys-sto-toronto-video
http://www.inewsgr.com/94/to-exypno-fanari-pou-xerei-pote-na-ginei-prasino.htm
http://www.zarpanews.gr/tag/%CE%B5%CE%BE%CF%85%CF%80%CE%BD%CE%B1-%CF%86%CE%B1%CE%BD%CE%B1%CF%81%CE%B9%CE%B1/

Κεφάλαιο 2

http://elearning.teicm.gr/file.php/318/askisi_therm.pdf

http://imm.demokritos.gr/platon/AEOAAUAC_OOIOO_AOOIIAOEOIIOO/aeoaauac_ooioo_aooiiaoeoiioo.html
http://ikaros.teipir.gr/phyche/Subjects/Routoulas/AutomatismoiVIKTE/ViomixanikosAutomatismos.pdf

Κεφάλαιο 3

https://www.arduino.cc/en/Guide/Introduction
https://www.arduino.cc/en/Main/ArduinoBoardMega2560
https://www.arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf
https://www.arduino.cc/en/Main/ArduinoBoardUno
https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf
https://www.arduino.cc/en/Reference/HomePage
http://wiring.org.co/
http://wiring.org.co/about.html
https://www.arduino.cc/en/Main/Software
https://processing.org/
http://www.nongnu.org/avr-libc/user-manual/index.html
http://web.engr.oregonstate.edu/~traylor/ece473/lectures/twi.pdf
nxp%20i2c%20bus.pdf
https://www.arduino.cc/en/Tutorial/MasterWriter
https://www.arduino.cc/en/Tutorial/MasterReader
http://www.blynk.cc/
http://www.blynk.cc/getting-started/
http://www.blynk.cc/getting-started/
https://examples.blynk.cc/?board=Arduino%20Mega%202560&shield=Serial%20or%20USB&example=GettingStarted%2FBlynkBlink&auth=20abbd6b3e674fd5aa6ff3bf8a0b66a0
http://help.blynk.cc/hardware-and-libraries/arduino/usb-serial
https://youtu.be/fgzvoan_3_w
https://youtu.be/I_hgIj2FdPI