# Πτυχιακή Εργασία

## "Αυτόματο Παράλληλο Παρκάρισμα με Χρήση Μικροελεγκτή και Παράλληλη Απεικόνιση σε Ηλεκτρονικό Υπολογιστή"

Σπουδαστής

Ιωάννου Βασίλειος

Α.Μ.Σ.: 36163

Υπεύθυνος Καθηγητής

Αλατσαθιανός Σταμάτης

*«Δηλώνω υπεύθυνα ότι το παρόν κείμενο αποτελεί προϊόν προσωπικής μελέτης και εργασίας και πως όλες οι πηγές που χρησιμοποιήθηκαν για τη συγγραφή της δηλώνονται σαφώς είτε στις παραπομπές είτε στη βιβλιογραφία. Γνωρίζω πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαι ενήμερος για την επέλευση των νομίμων συνεπειών»*

*Περίληψη*

*Σε αυτήν την εργασία θα σχεδιαστεί και θα αναλυθεί η δημιουργία ενός αυτοκινήτου, το οποίο θα είναι σε θέση να κάνει έλεγχο του περιβάλλοντος του, με σκοπό την εύρεση θέσης παρκινγκ αρκετά μεγάλη για το μέγεθός του. Μετά την εύρεση θα πραγματοποιεί αυτόματα παράλληλο παρκάρισμα. Όλη η διαδικασία θα φαίνεται γραφικά στον υπολογιστή. Θα αναλυθεί ο τρόπος λειτουργίας των μικροελεγκτών PIC της εταιρίας Microchip και συγκεκριμένα ο PIC16F628a που χρησιμοποιήθηκε. Παρατίθεται ο κώδικας που συνοδεύει το κάθε κομμάτι της εργασίας (Αμάξι, Ασύρματος Δέκτης, Ηλεκτρονικός Υπολογιστής) μαζί με τα απαραίτητα σχόλια. Τέλος αναλύονται τα πλεονεκτήματα που υπάρχουν στην χρήση μικροελεγκτών στον χώρο της τεχνολογίας.*

*Abstract*

*In this paper it will be designed and analyzed a way to create a small car that will be able to monitor it's environment with the aim of finding a parking space big enough for its size. After finding it, it will automatically perform parallel parking. The whole process it will be graphically animated on the computer screen. The operation of the PIC microcontrollers by Microchip and particular the PIC16F628a, that was used in this paper, it will be analyzed also. Every material (Car-Wireless Receiver-Computer) is accompanied by the firmware and software that was used along with all the necessary comments. Finally advantages of microcontrollers in the field of technology will be analyzed.*

# ΠΕΡΙΕΧΟΜΕΝΑ

*Κεφάλαιο 1*

## 1.1 ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΜΙΚΡΟΕΛΕΓΚΤΕΣ

Με τον όρο μικροελεγκτή εννοείται ένα ολοκληρωμένο κύκλωμα(IC), το οποίο εμπεριέχει τον μικροεπεξεργαστή και περιφερικά συστήματα. Τα περιφερικά συστήματα μπορεί να είναι μνήμη RAM ή ROM, σειριακή επικοινωνία, πόρτες (I/O) , μετατροπείς αναλογικού συστήματος σε ψηφιακό και αντίστροφα. Το εύρος της χρήσης των μικροελεγκτών είναι μεγάλο, καθώς χρησιμοποιούνται στα περισσότερα συστήματα αυτοματισμού ( ρομποτική , μηχανές εργοστασίων, οικιακές συσκευές κ.α). Επιπλέον χρήση τους γίνεται και σε συστήματα ελέγχου όπως οι ηλεκτρονικοί υπολογιστές, καθώς και σε συστήματα σχετιζόμενα με την ασφάλεια.

Αναλυτικότερα, ο μικροελεγκτής δια μέσω των περιφερικών του συστημάτων καθίσταται ικανός να συνδεθεί με εξωτερικές συσκευές ( Bluetooth, συστήματα υπερύθρων κ.α) και με την κατάλληλη ρύθμιση του, να επιφέρει τις επιθυμητές ενέργειες για τον κατασκευαστή ή να ενεργεί για τη εμφάνιση δεδομένων – αποτελεσμάτων στην οθόνη ενός υπολογιστή.

Η κατασκευάστρια εταιρία Microchip έχει δημιουργήσει μικροελεγκτές, τους επονομαζόμενους PIC (Peripheral Interface Controler)(Εικόνα 1.1). Η εταιρία έχει δημιουργήσει πληθώρα μοντέλων μικροελεγκτών, ώστε να μπορεί να εκπληρώσει τις διαφορετικές απαιτήσεις των καταναλωτών, για την δημιουργία των εκάστοτε εφαρμογών τους. Η Microchip έχει ονομάσει τους μικροελεγκτές της ως PIC, micro MCU's ( Micro-Controller Units).

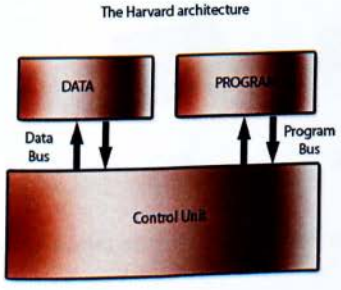
Εικόνα 1.1:Μικροελεγκής PIC

Η Microchip δεν είναι η μοναδική εταιρία κατασκευής μικροελεγκτών. Στην αγορά, υπάρχουν προϊόντα αντίστοιχων ανταγωνιστικών εταιριών. Κάποιες από τις γνωστότερες ανταγωνιστικές εταιρίες είναι η Atmel η οποία κατασκευάζει τους AVR και η Intel που δημιουργεί τους 8051, i860, i960. Πρέπει να σημειωθεί, ότι η Atmel κατασκευάζει μικροελεγκτές με επιπλέον χαρακτηριστικά, έτσι ώστε να είναι συμβατοί με τον γνωστό μικροελεγκτή της Intel 8051.

## 1.2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΚΑΤΗΓΟΡΙΕΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ PIC

Η κατασκευάστρια εταιρία Microchip χρησιμοποιεί συγκεκριμένο τύπο αρχιτεκτονικής για τους μικροελεγκτές της, όσο αναφορά τον αριθμό και την πολυπλοκότητα των εντολών που μπορούν να συμπεριλάβουν και να εκτελέσουν. Ο συγκεκριμένος τύπος αρχιτεκτονικής, σε αγγλική ονομασία λέγεται Reduce Instruction Set Computer (RISC) και σε ελληνική "Υπολογιστής περιορισμένου συνόλου εντολών". Στους μικροελεγκτές PIC, η κεντρική μονάδα επεξεργασίας (ΚΜΕ) και η μνήμη επικοινωνούν με αρχιτεκτονική Harvard, όπου η μνήμη δεδομένων και η μνήμη του προγράμματος δεν σχετίζονται μεταξύ τους, δηλαδή είναι ξεχωριστές και μέσω ξεχωριστών διαύλων γίνεται η διεκπεραίωση της επικοινωνίας (Σχήμα 1.2). Η προαναφερθέντα αρχιτεκτονική, έρχεται σε αντίθεση με την αρχιτεκτονική Von Neumman, όπου υπάρχει μόνο μία μνήμη για τις εκάστοτε εντολές και τα δεδομένα, καθώς επίσης η επικοινωνία επιτυγχάνεται μέσω ενός μόνο διαύλου.

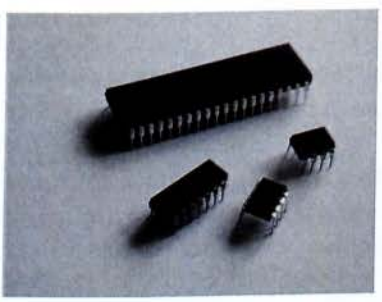
Η αρχιτεκτονική Harvard, επιτρέπει την κίνηση των εντολών και των δεδομένων στο ίδιο χρονικό διάστημα στους διαύλους, όπως επίσης στον χρόνο εκτέλεσης μίας εντολής να καταφθάνει η επόμενη. Σε περίπτωση που ο διάδρομος-δίαυλος ανάμεσα στην κεντρική μονάδα επεξεργασίας και της μνήμης του προγράμματος, έχει μήκος 16 bit, δίνεται η δυνατότητα προσκόμισης εντολών μήκους όμοιων ή μικρότερων, από αυτό του μήκους της ΚΜΕ κατά την διάρκεια ενός κύκλου εντολής.




Σχήμα 1.2: Αρχιτεκτονική Harvard

Οι μικροελεγκτές PIC μπορούν να διαχωριστούν σε διάφορες κατηγορίες , όπου εξ' αυτών τέσσερις είναι οι βασικότερες. Η πρώτη διάκριση σχετίζεται με το μέγεθος της λέξης-εντολής που μπορούν να εκτελέσουν. Αναλυτικότερα, υπάρχουν τρείς διαφορετικοί τύποι. Ο πρώτος είναι η βασική κατηγορία, όπου το μήκος εντολής είναι δώδεκα ψηφία, ο δεύτερος τύπος (μεσαία κατηγορία) με μήκος εντολής ίσου των 14 ψηφίων και ο τρίτος και τελευταίος τύπος, υψηλή κατηγορία, με 16 ψηφία ανά εντολή. Επίσης, αποκαλούνται διαφορετικά ως μήκος λέξης εντολής ή μήκος ψηφιο-λέξης. Τέλος, για την εκτέλεση των προαναφερθέντων, πρέπει να υπάρχει δίαυλος τουλάχιστον 16 bit, καθώς και το μήκος της μνήμης του προγράμματος να είναι αντίστοιχο, με αποτέλεσμα οι εντολές να καταλαμβάνουν μία θέση μνήμης, αφού έχουν κωδικοποιηθεί μέσα σε μόνο μια λέξη. Σε περίπτωση που οι εντολές δεν είναι μόνο μέσα σε μία λέξη, τότε αυξάνεται ο χρόνος που χρειάζεται για τη εκτέλεση τους.

Η επόμενη διάκριση γίνεται με βάση των αριθμό των ακίδων (pins) του ολοκληρωμένου συστήματος. Πιο συγκεκριμένα, στην αγορά υπάρχουν μικροελεγκτές PIC με διαφορετικό αριθμό ποδών (Εικόνα 1.3).



Εικόνα 1.3.: PIC με διαφορετικό αριθμό ακίδων

Η επόμενη διάκριση είναι με βάση την συσκευασία του PIC. Πρέπει να σημειωθεί ότι υπάρχει πλήθος κατηγοριών και υποκατηγοριών. Αναφορικά, τρία διαφορετικά είδη είναι η Dual In Line , Plastic Leaded Chip Carrier ,Plastic Quad Flat Pack (Εικόνα1.4).


Εικόνα1.4:Διαφορετικοί τύποι συσκευασιών PIC

Η τελευταία διάκριση γίνεται με βάση τον τύπο μνήμης. Η Microchip χρησιμοποιεί ανάλογη ονομασία για να υποδηλώσει συγκεκριμένα χαρακτηριστικά του εκάστοτε PIC.

Αναλυτικότερα το πρώτο γράμμα στην συσκευασία ενός προϊόντος δείχνει τον τύπο μνήμης που έχει χρησιμοποιηθεί. Επιπλέον, σε περίπτωση που αναγράφεται C στους μικροελεγκτές η μνήμη τους είναι τύπου EPROM, σε περίπτωση που αναγράφονται τα γράμματα CR έχουν μνήμη προγράμματος ROM και σε περίπτωση που αναγράφεται το F, τότε η μνήμη προγράμματος είναι τύπου FLASH αντιστοίχως.

Αναφορά πρέπει να γίνει, σε επιπλέον χαρακτηριστικά που υπάρχουν στην ονομασία του εκάστοτε μοντέλου PIC. Οι μικροελεγκτές που έχουν τον συμβολισμό 12 στην ονομασία τους , έχουν οχτώ ακίδες, σε περίπτωση όμως που έχουν περισσότερα από 8 πόδια και μήκος εντολής 12 ή 14 ψηφίων τότε συμβολίζονται με τον αριθμό 16 στην ονομασία τους. Επιπλέον μικροελεγκτές με τον αριθμό 16 ή 18 στην ονομασία τους, σημαίνει ότι έχουν 16 ψηφία μήκος εντολής. Τέλος, σε περίπτωση που υπάρχει στο τέλος της ονομασίας τους το γράμμα A ,τότε σημαίνει ότι είναι αναβαθμισμένο μοντέλο PIC από το προηγούμενο μοντέλο που είχε τον αριθμό χωρίς τον συμβολισμό A.

### 1.3 ΣΥΓΚΡΙΣΗ ΜΙΚΡΟΕΛΕΓΚΤΗ ΜΕ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΗ

Στο κεφάλαιο 1.3 θα γίνει σύγκριση της απλούστερης μορφής της δομής του υπολογιστικού συστήματος μεταξύ ενός μικροεπεξεργαστή και ενός μικροελεγκτή. Υπολογιστικό σύστημα θεωρείται αυτό το οποίο περιλαμβάνει υλικό (hardware) και λογισμικό (software). Αναλυτικότερα, το σύστημα αυτό στελεχώνεται από την Κεντρική μονάδα επεξεργασίας, την μνήμη και τις μονάδες εισόδου- εξόδου (IO, Input/ Output). Η επικοινωνία μεταξύ των προαναφερθέντων και άλλων περιφερειακών συστημάτων επιτυγχάνεται μέσω των δίαυλων.

Ο επεξεργαστής, είναι το βασικότερο στοιχείο ενός μικρο-υπολογιστικού συστήματος. Στο εσωτερικό αυτού μπορεί να υπάρχει, η αριθμητική και η λογική ομάδα όπως επίσης και κάποιοι καταχωρητές. Με τον κατάλληλο προγραμματισμό του μικροελεγκτή, αυτός μπορεί να εκτελεί εντολές σταδιακά, έτσι ώστε να γίνει η επεξεργασία δεδομένων. Ο προγραμματισμός του επεξεργαστή αποθηκεύεται στην μνήμη του και από εκεί όταν τεθεί σε λειτουργία εκείνος αρχίζει σταδιακά και ανακαλεί τις εντολές με τις οποίες τον έχουν προγραμματίσει. Η δομή που αναλύθηκε παραπάνω, είναι η βάση για τους επεξεργαστές που κατασκευάζει η Intel για την κατηγορία μοντέλων x86 και για τους αντίστοιχους της AMD. Πρέπει να σημειωθεί ότι τέτοιοι επεξεργαστές έχουν χρησιμοποιηθεί σε παιχνιδο-κονσόλες, σε αριθμομηχανές και σε πολλά αντικείμενα καθημερινής χρήσης. Στις παραπάνω μηχανές ο επεξεργαστής, είναι σε θέση να κάνει ορθούς, γρήγορους υπολογισμούς καθώς επίσης να επικοινωνεί με περιφερειακά συστήματα μέσω πλακέτας (motherboard) στην περίπτωση πχ. Ηλεκτρονικών Υπολογιστών.

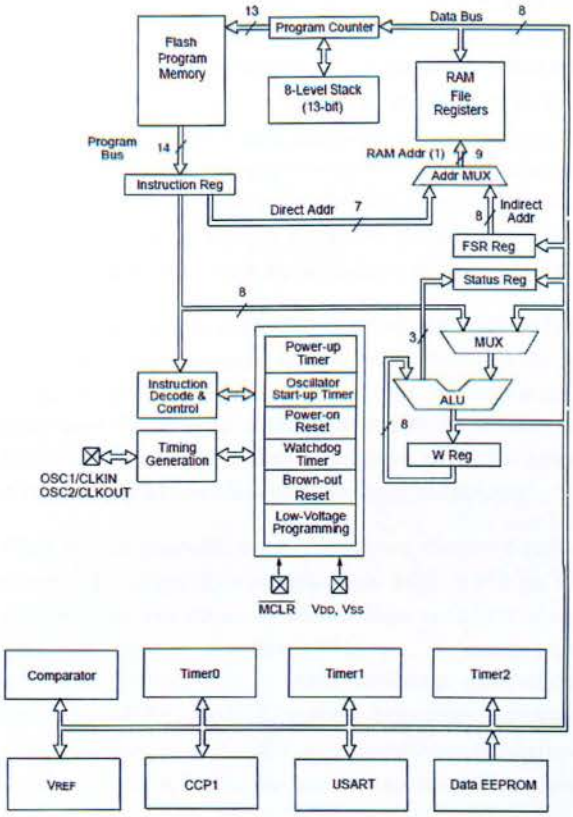
Η δημιουργία ενός μικρο-υπολογιστικού συστήματος για τον έλεγχο μιας κατασκευής-συσκευής χρειάζεται διάφορα περιφερειακά στοιχεία. Για την δημιουργία ενός βιομηχανικού αυτοματισμού θα είναι αναγκαίος ένας μετατροπέας αναλογικού σε ψηφιακού σήματος (ADC). Σε κάτι πιο απλό όπως η δημιουργία μιας αυτόματης ταΐστρας κατοικίδιων θα χρειαστεί ένα ρολόι (real time clock). Σε αυτό το παράδειγμα, μπορεί χρησιμοποιηθεί και ένας κατάλληλα προγραμματισμένος χρονιστής. Συμπερασματικά, ο

σχεδιασμός ενός μικρο-υπολογιστικού συστήματος θα πρέπει να εμπεριέχει, το κύκλωμα, τον επεξεργαστή ο οποίος είναι αναγκαίος καθώς επίσης και τις περιφερειακές συσκευές. Βέβαια, με αυτόν τον τρόπο το επίπεδο δυσκολίας όπως επίσης και της πολυπλοκότητας του συστήματος είναι μεγάλη. Η αποφυγή αυτού το προβλήματος, μπορεί να γίνει με την χρήση ενός ολοκληρωμένου συστήματος (μικροελεγκτής) , το οποίο περιλαμβάνει το κύκλωμα του επεξεργαστή και όλες τις χρήσιμες περιφερειακές μονάδες σε ένα τσιπ.

## 1.4 ΔΟΜΗ ΜΙΚΡΟΕΛΕΓΚΤΗ PIC

Ο μικροελεγκτής PIC έχει συγκεκριμένη δομή. Αρχικά, η κεντρική μονάδα επεξεργασίας (ΚΜΕ) επικοινωνεί με την μνήμη δεδομένων και προγράμματος, όπως επίσης και με τις υπόλοιπες περιφερειακές μονάδες. Οι περιφερειακές μονάδες μπορεί να είναι θύρες, χρονιστές, η σειριακή επικοινωνία, a/d καθώς και η μονάδα διαμόρφωσης μεγέθους παλμού. Η ΚΜΕ εμπεριέχει στο εσωτερικό της την αριθμητική και τη λογική μονάδα και είναι σε θέση να μπορεί να κάνει αριθμητικές πράξεις και λογικές όπως για παράδειγμα AND, OR, NOT κ.α). Ο καταχωρητής W παρέχει βοήθεια, στην εκτέλεση πράξεων δια μέσου της μεσολάβησης του, στον εκάστοτε χρήστη του συστήματος. Ο καταχωρητής STATUS είναι υπεύθυνος για την εύρεση του κατάλληλου τμήματος στην μνήμη δεδομένων, καθώς επίσης και για άλλες λειτουργίες. Πρέπει να σημειωθεί ότι οι δύο προαναφερθέντες καταχωρητές έχουν μεγάλη σημασία για το σύστημα.

Η κεντρική μονάδα επεξεργασίας, ουσιαστικά ανακαλεί και στη συνέχεια εκτελεί τις εντολές που της έχουν δοθεί, κατά τον προγραμματισμό από τον χρήστη. Στην ανάκληση και εκτέλεση των εντολών βοηθάει και ο μετρητής προγράμματος, ο οποίος ουσιαστικά είναι καταχωρητής (program counter), αφού αυτός περιλαμβάνει στην μνήμη του την εκτελεσθείσα εντολή.




**Σχήμα 1.5:Δομή PIC16F628a**

## 1.5 ΟΡΓΑΝΩΣΗ ΜΝΗΜΗΣ ΜΙΚΡΟΕΛΕΓΚΤΗ

Το πρόγραμμα και τα δεδομένα ενός μικροελεγκτή βρίσκονται αποθηκευμένα σε ξεχωριστές μνήμες ( αρχιτεκτονική Harvard). Οι μνήμες RAM και ROM υπάρχουν σε έναν μικροελεγκτή. Η καταχώρηση του προγράμματος του χρήστη εισάγεται και αποθηκεύεται στη μνήμη ROM, ενώ στην RAM βρίσκονται τα δεδομένα, ουσιαστικά στελεχώνεται από τους καταχωρητές, όπου από αυτούς οι περισσότεροι είναι υπεύθυνοι για τον έλεγχο των περιφερειακών συσκευών. Επιπλέον, οι περισσότερες εντολές του προγράμματος του χρήστη απευθύνονται σε αυτούς. Αναλυτικότερα σε περίπτωση αλλαγής των bit των καταχωρητών , διεξάγεται το κατάλληλο αποτέλεσμα. Πρέπει να αναφερθεί πως η μνήμη, η οποία συνηθίζεται να είναι Flash, μπορεί να έχει μέγεθος από 2-8 Kbytes, σε αντίθεση με την μνήμη δεδομένων που μπορεί το μέγεθος της να είναι 512 bytes στο σύνολο. Επιπροσθέτως, στην μνήμη ROM τα περιεχόμενά της δεν μπορούν να αλλοιωθούν, ούτε να μεταβληθούν ακόμα και όταν δεν υπάρχει τροφοδοσία στο σύστημα. Τα χαρακτηριστικά αυτά έρχονται σε αντίθεση με αυτά της μνήμης RAM,η οποία στην περίπτωση που δεν υπάρχει τροφοδοσία δεν συγκρατεί τις αλλαγές που έγιναν στους καταχωρητές.

Η κατασκευάστρια εταιρία Microchip παράγει ολοκληρωμένα με τρείς διαφορετικούς τύπους ROM. Ο πρώτος τύπος είναι ο EPROM (γράμμα C), στον οποίο η διαγραφή της μνήμης γίνεται με υπεριώδη ακτινοβολία. Οι παράγοντες που επηρεάζουν το απαιτούμενο χρόνο για την διαγραφή είναι, το μήκος κύματος της ακτινοβολίας, καθώς επίσης η ένταση και η απόσταση από την πηγή παραγωγής της. Ο επόμενος τύπος λέγεται ROM (γράμματα CR) και η μνήμη γράφεται από τον κατασκευαστή, για αυτόν το λόγο το κόστος της τιμής της είναι μειωμένο. Ο τρίτος και τελευταίος τύπος ονομάζεται FLASH ROM (γράμμα F). Στο συγκεκριμένο τύπο μνήμης η διαγραφή γίνεται ηλεκτρονικά και είναι δυνατόν ο προγραμματισμός της να γίνει χωρίς την απομάκρυνσή της από το κύκλωμα.

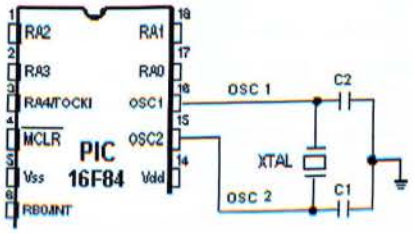
Η μνήμη RAM χωρίζεται σε δύο μέρη από τα οποία και αποτελείται. Το πρώτο μέρος είναι το κομμάτι που εμπεριέχει τους καταχωρητές ειδικής λειτουργίας, Special Function Registers (SFR), οι οποίοι χρησιμοποιούνται για τον προγραμματισμό της λειτουργίας του μικροελεγκτή. Το δεύτερο μέρος είναι οι γενικής χρήσης καταχωρητές , General Purpose Registers (GPR), οι οποίοι είναι η ευρύτερη χρήση της μνήμης RAM και χρησιμοποιούνται για την προσωρινή αποθήκευση δεδομένων του χρήστη.

Το επόμενο θέμα που θα αναλυθεί είναι η άμεση και έμμεση διευθυνσιοδότηση. Κατά την άμεση διευθυνσιοδότηση χρειάζεται η χρήση των ψηφίων PR0 και RP1 έτσι ώστε να επιτευχθεί η προσπέλαση από ένα τμήμα σε κάποιο άλλο. Η "GOTO" είναι μια εντολή τέτοιου τύπου. Κατά την έμμεση διευθυνσιοδότηση , ο καταχωρητής έμμεσης προσπέλασης που ανήκει στην μνήμη FSR, η οποία υπόκειται στον καταχωρητή FSR, ευθύνεται για την έμμεση προσπέλαση της INDEF μνήμης, όπως επίσης χρησιμοποιείται και το ψηφίο IRP του STATUS καταχωρητή. Επιτυγχάνεται η προσπέλαση κομματιών μνήμης για κυκλώματα που χρησιμοποιούν μνήμη δεδομένων και των τεσσάρων κομματιών.

## 1.6 ΤΑΣΗ ΛΕΙΤΟΥΡΓΙΑΣ ΚΑΙ ΧΡΟΝΙΣΜΟΣ

Η τροφοδοσία του PIC επιτυγχάνεται με τροφοδοτικά τα οποία βρίσκονται στην ίδια πλακέτα που είναι ενσωματωμένος ο μικροελεγκτής και προσδίδουν συνεχή τάση ρεύματος της τάξεως από 2- 6 VDC. Το πρώτο είδος τροφοδοτικού που χρησιμοποιείται έχει 5V σταθερή τάση ρεύματος και χρειάζεται τον σταθεροποιητή τάσης 7805. Το δεύτερο είδος τροφοδοτικού είναι αυτό της ρυθμιζόμενης τάσης ρεύματος και είναι αναγκαίος ο ρυθμιστής LM317T. Τέλος για την τροφοδοσία του PIC μπορεί να χρησιμοποιηθεί τροφοδοτικό χωρίς μετασχηματιστή , των οποίων η κατασκευή γίνεται με κύκλωμα και για να λειτουργήσουν πρέπει να συνδυαστούν με αντιστάσεις, πυκνωτές, φίλτρα, διόδους κτλ..

Όπως ειπώθηκε παραπάνω στους μικροελεγκτές υπάρχουν ενσωματωμένοι χρονιστές καθώς και μονάδες οι οποίες παράγουν παλμούς. Ο μικροελεγκτής μπορεί να έχει στο εσωτερικό του, ένα κύκλωμα ταλάντωσης ή να συνδεθεί με κρύσταλλο, έτσι ώστε οι χρονιστές και η μονάδα παραγωγής παλμών να συγχρονιστούν και να υπάρξει παραγωγή παλμών ρολογιού. Αυτό είναι αναγκαίο για δύο λόγους. Αρχικά για την τέλεση εντολών που ζητά ο χρήστης αλλά και για την σωστή λειτουργία των περιφερειακών του. Στην περίπτωση που ο χρήστης θέλει να συνδέσει κρύσταλλο, τότε αυτός συνδέεται με τον μικροελεγκτή στις ακίδες OSC1 και OSC2. Πρέπει να σημειωθεί ότι με την χρήση εσωτερικού κυκλώματος ταλάντωσης συνήθως επιτυγχάνεται μικρότερη ακρίβεια και κατανάλωση, ενώ με την χρήση εξωτερικού κρυστάλλου μπορούν να επιτευχθούν συχνότητες από 8-20+ Mhz.

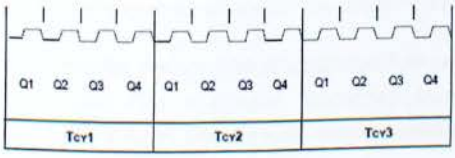



Σχήμα 1.6: Σύνδεση PIC με εξωτερικό κρύσταλλο

## 1.7 ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΕΝΤΟΛΩΝ

Η σταδιακή ανάκληση και η εκτέλεση μιας εντολής από την κεντρική ομάδα επεξεργασίας είναι ένας κύκλος εντολής. Η επίτευξη ενός τέτοιου κύκλου, απαιτεί μία ή ένα σύνολο λειτουργιών όπως η αποκωδικοποίηση της εντολής, η ανάγνωση και εγγραφή της στην μνήμη, η επεξεργασία των δεδομένων κοκ. Για την προαναφερθείσα διαδικασία απαραίτητος είναι ένας κύκλος μηχανής. Χρειάζεται δηλαδή για κάθε μία εντολή που καλείται. Η αναγκαιότητα αυτού του κύκλου υπάρχει για την γνωστή αλληλοεπικάλυψη εντολών ή αλλιώς συνεχής διοχέτευση εντολών. Αναλυτικότερα, σε περίπτωση το σύστημα εκτελεί μία συγκεκριμένη εντολή σε ένα χρονικό διάστημα, τότε μπορεί την ίδια στιγμή να προσκομίζεται μια νέα διαφορετική εντολή.

Η τέλεση ενός κύκλου εντολής επιτυγχάνεται μέσα σε τέσσερις χρονικές περιόδους του ρολογιού, οι οποίες εμπεριέχουν την αποκωδικοποίηση της εντολής, την ανάγνωση και αποθήκευσής της στην μνήμη και την επεξεργασία δεδομένων. Η διαδικασία αυτή γίνεται περισσότερο κατανοητή στο σχήμα 1.7. Στο σχήμα που ακολουθεί, οι χρονικοί περίοδοι παρουσιάζονται με Q1, Q2, Q3, Q4 και με Tcy1 Tcy2 Tcy3 οι κύκλοι τον εντολών.




Σχήμα 1.7: Ένας κύκλος εντολής ( TCY ) = 4 περιόδους ρολογιού (Q)

Πρέπει να αναφερθεί ότι για όλες τις προαναφερθείσες εντολές απαιτείται ένας κύκλος εντολής, σε αντίθεση για τις εντολές CALL, GO TO, RETURN, RETFIE και RETLW που απαιτούνται δύο κύκλοι. Στην περίπτωση των εντολών CALL και GOTO οι δύο κύκλοι χρειάζονται, διότι αυτές οι δύο εντολές αλλάζουν τον απαριθμητή προγράμματος. Όταν

υπάρχει γνώση για το ποσό των κύκλων εκτέλεσης μιας εντολής τότε καθίσταται δυνατός ο υπολογισμός των συνολικών κύκλων εντολών και που χρειάζεται για την εκτέλεση του προγράμματος. Με βάση λοιπόν τους τέσσερις κύκλους ρολογιού που χρειάζονται για την διεκπεραίωση μιας εντολής, και γνωρίζοντας την συχνότητα που λειτουργεί ο μικροελεγκτής με τον τύπο «***Διάρκεια κύκλου εντολής = 4/ (συχνότητα λειτουργίας μικροελεγκτή)***» είναι δυνατή η εύρεση της διάρκειας του κύκλου εντολής. Το σύνολο της διάρκειας του έκαστου κύκλου αποτελεί το χρόνο εκτέλεσης του προγράμματος.

## 1.8 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Ο προγραμματισμός των μικροελεγκτών PIC μπορεί να γίνει με τρείς τρόπους, των οποίων η ανάλυση θα παρατεθεί .

Ο πρώτος τρόπος είναι ο σειριακός προγραμματισμός επί τόπου ICSP (In Circuit Serial Programming). Με αυτό το είδος προγραμματισμού ο μικροελεγκτής μπορεί να προγραμματιστεί στην πλακέτα που βρίσκεται, χωρίς να χρειάζεται η διεξαγωγή του τσιπ από την πλακέτα εφαρμογής. Το προηγούμενο έχει σαν αποτέλεσμα οι αλλαγές και οι διορθώσεις του κώδικα να μπορούν να γίνονται χωρίς ο χρήστης να πρέπει να τοποθετεί το ολοκληρωμένο σε άλλη πλακέτα. Η κατασκευάστρια εταιρία Microchip έχει εκδώσει ειδικό εγχειρίδιο για τις ειδικές συνθήκες που χρειάζονται για τον προγραμματισμό κάθε μοντέλου PIC. Ο προγραμματισμός ICSP μπορεί να γίνει με δύο τρόπους.

Ο πρώτος τρόπος είναι ο επονομαζόμενος χαμηλής τάσης LVP (Low Voltage Programming) των +5 Volt. Σε αυτή την περίπτωση η ακίδα RB3 είναι υπεύθυνη για τον προγραμματισμό LVP καθώς και αυτή συνδέεται με τα +5V. Ο δεύτερος τρόπος προγραμματισμού ICSP είναι αυτός με υψηλή τάση HVP (High Voltage Programming) των 13V±0,5V. Στην περίπτωση αυτή η ακίδα RB6 χρησιμοποιείται ως γραμμή παλμών και μεταφοράς δεδομένων και τα 13V συνδέονται με την ακίδα Master Clear.

Ο δεύτερος τρόπος προγραμματισμού μικροελεγκτών είναι η δυνατότητα αυτό-προγραμματισμού. Ο τρόπος αυτός μπορεί να γίνει διότι στην περιοχή της μνήμης του μικροελεγκτή υπάρχει ένα πρόγραμμα που ονομάζεται bootloader το οποίο υπάρχει για την φόρτωση ενός άλλου κομματιού μνήμης, όπου εκεί υπάρχει το πρόγραμμα που πρέπει να εκτελεστεί. Ο υπολογιστής μεταδίδει το πρόγραμμα στον μικροελεγκτή (σειριακά) μέσω της μονάδας USART του μικροελεγκτή, η οποία επιτρέπει την σειριακή επικοινωνία.

Ο τρίτος και τελευταίος τρόπος για τον προγραμματισμό του μικροελεγκτή είναι η χρήση του προγραμματιστή Pic Star Plus. Η εταιρία Microchip δημιουργεί αυτόν τον προγραμματιστή, ο οποίος χρησιμοποιεί την μέθοδο ICSP που αναλύθηκε πρωτύτερα.

## 1.9 ΑΡΧΙΚΗ ΜΟΡΦΟΠΟΙΗΣΗ

Η αρχική μορφοποίηση περιέχει ψηφία τα οποία είναι υπεύθυνα για την ρύθμιση λειτουργιών του hardware και για τον τρόπο που θα επιλεχθεί να λειτουργήσει ο μικροελεγκτής. Τα ψηφία αυτά τα γράφει και τα αποθηκεύει η εκάστοτε εταιρία παραγωγής μικροελεγκτών κατά τον προγραμματισμό και ο χρήστης δεν μπορεί να τα

διαβάσει με κανένα πρόγραμμα. Η εταιρία τα καταχωρεί στην διεύθυνση 2007H της μνήμης του προγράμματος και υπάρχει η περίπτωση τα ψηφία αυτά να είναι διαφορετικά στον κάθε επεξεργαστή. Τα ψηφία προσφέρουν προστασία του κώδικα από αντιγραφή. Στην συνέχεια ακολουθεί σε πίνακα κάποια βασικά ψηφία της λέξης αρχικής μορφοποίησης που συνήθως εμφανίζονται σε έναν μικροελεγκτή.

| CP1 | CP0 | - | - | - | CPD | LVP | BODEN | - | - | PWRTE | WDTE | FOSC1 | FOSC2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | | | | | | | | | | | | | 0 |

Σχήμα 1.8.: Κοινά ψηφία αρχικής μορφοποίησης

| | |
|---|---|
| Bit 14:13 (CP1:CP0): | Με παραλλαγή αυτών των ψηφίων ενεργοποιείται, ή όχι η προστασία του κώδικα της εφαρμογής από αντιγραφή. |
| Bit 8 (CPD) : | Με αυτό το ψηφίο ενεργοποιείται, η όχι η προστασία της μνήμης EEPROM |
| Bit 7 (LVP) | Ο τρόπος προγραμματισμού χαμηλής τάσης LVP (Low Voltage Programming) που περιγράφεται παραπάνω ενεργοποιείται από αυτό εδώ το ψηφίο. |
| Bit 6 (BODEN): | Με αυτό το ψηφίο ενεργοποιείται, εάν θα συμβεί reset ύστερα από πτώση τάσης. Reset προκαλείται εάν η τάση τροφοδοσίας (VDD) πέσει κάτω από το κατώφλι τάσης VBOR (Brown Out Reset - περίπου 4V) για ένα συγκεκριμένο χρονικό διάστημα περίπου 100 μsec. |
| Bit 3 (PWRTE): | Με αυτό το ψηφίο ενεργοποιείται, ο χρονιστής PWRT (Powerup Timer), ο οποίος επιτρέπει στην τάση τροφοδοσίας (VDD) να φτάσει την ονομαστική της τιμή ύστερα από την αρχική τροφοδοσία, ή ύστερα από πτώση τάσης .Ο χρόνος που παρέχεται από τον χρονιστή αυτόν είναι περίπου 72msec και για όλο αυτό το χρονικό διάστημα το τσιπ παραμένει σε κατάσταση Reset. Εάν το προηγούμενο ψηφίο (BODEN) είναι ενεργοποιημένο αυτό το ψηφίο ενεργοποιείται αυτόματα. |
| Bit 2 (WDTE): | Με αυτό το ψηφίο ενεργοποιείται, ο χρονιστής επιτήρησης WDT (Watchdog Timer). Αυτός ο χρονιστής είναι ανεξάρτητος από το εάν υπάρχει, ή όχι συνδεδεμένο εξωτερικό ρολόι και παραμένει σε λειτουργία ακόμα και αν έχει εκτελεστεί η εντολή SLEEP(χαμηλή κατανάλωση). Πρόκειται για ένα κύκλωμα μέτρησης χρόνου περίπου 18msec) μετά το πέρας του οποίου προκαλείται Reset στον μικροελεγκτή. Μπορεί να χρησιμοποιηθεί σε περιπτώσεις όπου υπάρχουν εξωτερικές παρεμβολές. Τότε προκαλείται Reset και ο μικροελεγκτής επανέρχεται σε κανονική λειτουργία. |
| Bit 1:0 (FOSC1:FOSC0) | Με την παραλλαγή αυτών των ψηφίων επιλέγεται ο τύπος ταλαντωτή (εσωτερικός ή εξωτερικός) και η συχνότητα λειτουργίας. |

Βιβλιογραφία Κεφαλαίου

<<Α>> "Εισαγωγή στους μικροελεγκτές PICmicro" Σταμάτης Αλατσαθιανός
<<Β>> "Ανάπτυξη Συστημάτων με Μικροελεγκτές 8051" Σταμάτης Αλατσαθιανός
<<Γ>> "Microcontroller Programming" Julio Sanchez, Maria P. Canton
http://www.pi-schools.gr/lessons/tee/electronic/biblia.php
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf
http://www.hlektronika.gr/
http://www.gooligum.com.au/
http://www.electronics-lab.com/pic-in-greek/
http://en.wikipedia.org/wiki/PIC_microcontroller
http://el.wikipedia.org/wiki/Reduced_instruction_set_computer
http://el.wikipedia.org/wiki/CISC

Εικόνες Κεφαλαίου

Εικόνα 1.1: Μικροελεγκτής PIC
http://www.ps3hax.net/wp-content/uploads/2010/10/pic18f2550.jpg

Εικόνα 1.2: Αρχιτεκτονική Harvard
http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_3/vonn_neuman/miniweb/images/harvard_architecture.jpg

Εικόνα 1.3: PIC με διαφορετικό αριθμό ακίδων
http://socialbarrel.com/wp-content/uploads/2011/09/microchip-ships-10-billionth-pic-mcu-to-samsung-electronics.jpg

Εικόνα1.4:Διαφορετικοί τύποι συσκευασιών PIC
http://octopart.com/ds26ls31cn-national+semiconductor-21289
http://octopart.com/pic16f877-20/l-microchip-104362
http://theartificialintelligence.blogspot.com/2009/11/beyond-microchips.html
http://theartificialintelligence.blogspot.com/2009/11/beyond-microchips.html

Σχήμα 1.5: Δομή μικροελεγκτή PIC16F628a
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf

Σχήμα 1.6: Σύνδεση PIC με εξωτερικό κρύσταλλο
http://www.8051projects.net/pic_tutorial/2_6.gif

Σχήμα 1.7: Ένας κύκλος εντολής ( TCY ) = 4 περιόδους ρολογιού(Q)
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf

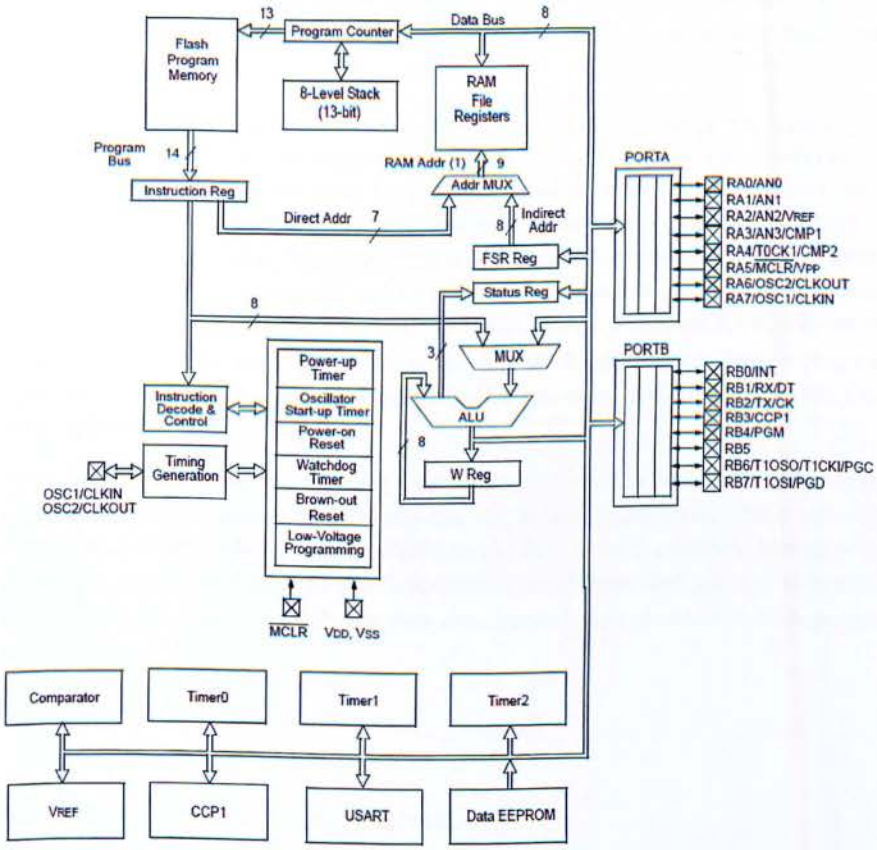
Κεφάλαιο 2

## 2.1 ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα ακολουθήσει η περιγραφή του μικροελεγκτή της Microchip PIC16F628a, ο οποίος χρησιμοποιήθηκε στην εργασία αυτή, τα χαρακτηριστικά του, καθώς και οι περιφερειακές μονάδες που τον συνοδεύουν. Ακόμα, θα γίνει ανάλυση του μπλοκ διαγράμματος καθώς και της λειτουργίας της κάθε ακίδας του. Θα εξηγηθεί η λειτουργία των εσωτερικών κυκλωμάτων επανεκκίνησης (reset) και διακοπών (interrupts). Ιδιαίτερη σημασία θα δοθεί στις περιφερειακές μονάδες σειριακής επικοινωνίας (serial communication), θύρες εισόδου/εξόδου (I/O ports) αλλά και στην μονάδα Σύγκρισης/Σύλληψης/Παραγωγής Παλμών (Compare/Capture/PWM), οι οποίες θα χρησιμοποιηθούν στην εργασία αυτή. Τέλος, θα αναφερθούν δύο πολύ βασικές περιφερειακές μονάδες, οι οποίες δεν χρησιμοποιήθηκαν. Αυτές είναι, η μονάδα Χρονιστών/Απαριθμητών (Timers/Counters) και ο Μετατροπέας αναλογικού σήματος σε ψηφιακό ( ADC).

## 2.2 ΜΙΚΡΟΕΛΕΓΚΤΗΣ PIC16F628A

Ο PIC16F628a είναι ένας μικροελεγκτής με μνήμη τύπου Flash και ανήκει στην οικογένεια των χαμηλής τιμής, υψηλής απόδοσης 8-bit μικροεπεξεργαστών. Περιέχει 2048 λέξεις μνήμης για προγραμματισμό και 2024-bytes μνήμη τύπου SRAM μαζί με 128 bytes EEPROM για αποθήκευση δεδομένων. Η τάση λειτουργίας του είναι ανάμεσα στα -0.3V και 6.5V. Ο μικροελεγκτής PIC16F628a είναι, όπως αναφέρεται και από την ονομασία του, η ανανεωμένη έκδοση του PIC16F628. Περιβάλλεται από 18 pins εκ των οποίων 16 είναι λειτουργίες εισόδου-εξόδου. Ένα από τα pins είναι για λειτουργίες τύπου CCP. Μέσα στο ολοκληρωμένο κύκλωμα του PIC16F628a περιέχονται δύο συγκριτές αλλά και 3 χρονιστές, δύο των 8-bit και ένας των 16. Παρακάτω φαίνεται το μπλοκ διάγραμμα του μικροελεγκτή αυτού.




Σχήμα:2.1: Μπλοκ διάγραμμα μικροελεγκτή PIC16F628a

Από το παραπάνω σχήμα μπορεί κανείς να παρατηρήσει τα περιφερειακά που προαναφέρθηκαν και που περιγράφονται στα επόμενα υποκεφάλαια. Τον καταχωρητή εργασίας (W Register), την αριθμητική και λογική μονάδα (ALU), τους πολυπλέκτες (MUX), τις μνήμες, τον μετρητή προγράμματος (Program Counter), κτλ.. Ορισμένα ακόμα από τα στοιχεία που υπάρχουν στο μπλοκ-διάγραμμα μικροελεγκτή και δεν έχουν αναφερθεί, εξηγούνται παρακάτω:

- Αριθμητική και Λογική Μονάδα (ALU): Η ALU είναι ένα θεμελιώδη στοιχείο μιας ΚΜΕ. Είναι ένα ψηφιακό κύκλωμα που κάνει όλες τις αριθμητικές και λογικές πράξεις.

- Καταχωρητής Εντολής (Instruction Register): Σε αυτόν τον καταχωρητή αποθηκεύεται η εντολή που δείχνει ο μετρητής προγράμματος. (Program Counter)

- Πολυπλέκτες (MUX): Μια συσκευή που επιλέγει ένα από τα πολλά αναλογικά ή ψηφιακά σήματα εισόδου και τα προωθεί σε μια ενιαία γραμμή.

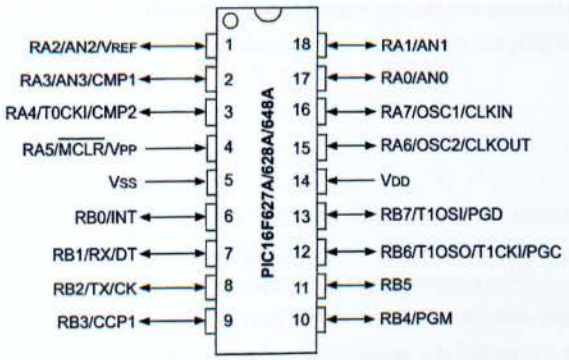
- Καταχωρητής W: Ο καταχωρητής στον οποίο αποθηκεύονται όλα τα αποτελέσματα από την ALU.

- Αποκωδικοποίηση και έλεγχος εντολής (Instruction Decode and Control): Ο έλεγχος και η εκτέλεση της εντολή που αποθηκεύτηκε στον καταχωρητή εντολής (Instruction Register)

- Καταχωρητής σωρού 8 επιπέδων ( 8- Level Stack): Σε αυτόν τον καταχωρητή αποθηκεύεται η διεύθυνση της επόμενης εντολής που πρόκειται να εκτελεστεί από την CPU μετά την αλλαγή της κανονικής ροής προγράμματος από πρόκληση διακοπής ή από κλήση της εντολής CALL, η οποία καλεί μία υπορουτίνα και ο μετρητής προγράμματος συνεχίζει με τις εντολές τις υπορουτίνας. Όταν τελειώσει η εκτέλεση της υπορουτίνας τότε ο μετρητής προγράμματος συνεχίζει κανονικά από το σημείο που διακόπηκε η εκτέλεση του καθώς αντιγράφεται σε αυτόν η τιμή του καταχωρητή σωρού (Stack). Ο αριθμός 8, υποδηλώνει ότι ο καταχωρητής σωρού μπορεί να αποθηκεύσει έως και 8 διαφορετικές διευθύνσεις, που σημαίνει ότι, η υπορουτίνα που κλήθηκε μπορεί να έχει άλλες 7 κλήσεις τις εντολής CALL (υπορουτίνες)

- Αναλογικοί Συγκριτές ( Analog Comparators): Είναι μία από τις περιφερειακές μονάδες που υποστηρίζουν οι μικροελεγκτές. Σε πολλές περιπτώσεις (όπως και στην περίπτωση του PIC16F628a) συμπεριλαμβάνονται δύο τέτοιες μονάδες. Μπορούν να συγκρίνουν αν μία τάση εισόδου είναι μικρότερη ή μεγαλύτερη από μία τιμή κατωφλίου και για αυτόν τον λόγο είναι χρήσιμοι όταν στον μικροελεγκτή συνδέονται διάφορα είδη αισθητήρων.

Ακολουθεί το γράφημα ακίδων του PIC16F628a.




Σχήμα 2.2: Γράφημα ακίδων PIC16F628a

Εύκολα παρατηρείται από το παραπάνω γράφημα , ότι για την σωστή λειτουργία κάποιων περιφερειακών μονάδων , χρειάζονται μία ή περισσότερες ακίδες. Παρακάτω ακολουθεί ο σχολιασμός της λειτουργίας όλων των ακίδων.

- Τα pin με τον χαρακτηρισμό RA0-RA7 χρησιμοποιούνται για την PORTA.

- Τα pin με τον χαρακτηρισμό RB0-RB7 χρησιμοποιούνται για την PORTB.
- Τα pin με τον χαρακτηρισμό AN0-AN3 χρησιμοποιούνται για τα τέσσερα κανάλια του A/D μετατροπέα.
- Το pin με τον χαρακτηρισμό OSC1/CLKIN χρησιμοποιείται για την σύνδεση εξωτερικού κυκλώματος ρολογιού (κρύσταλλος).
- Το pin με τον χαρακτηρισμό OSC2/CLKOUT χρησιμοποιείται για την σύνδεση εξωτερικού κυκλώματος ρολογιού (κρύσταλλος).
- Τα pin με τον χαρακτηρισμό T0CKI και T1CKI χρησιμοποιούνται για την λειτουργία TIMER0 και TIMER1.
- Τα pin τον χαρακτηρισμό CNP1 και CNP2 χρησιμοποιούνται για τις λειτουργίες που αφορούν τους δύο αναλογικούς συγκριτές.
- Το pin με τα χαρακτηρισμό MCLR χρησιμοποιείται για την πρόκληση reset από τον χρήστη κατά την διάρκεια λειτουργίας.
- Το pin INT χρησιμοποιείται για την πρόκληση διακοπής ( interrupt) από εξωτερική πηγή.
- Τα pin RX- DT- TX-CK χρησιμοποιούνται για τις λειτουργίες που αφορούν τον συγχρονισμό και μετάδοση δεδομένων που αφορούν την σειριακή επικοινωνία.
- Το pin με τον χαρακτηρισμό CCP1 χρησιμοποιείται στις λειτουργίες που αφορούν την περιφερειακή μονάδα Σύγκρισης/ Σύλληψης/ Παραγωγής μεταβαλλόμενων παλμών.
- Τα pin με τον χαρακτηρισμό PGM/PGC/PGC χρησιμοποιούνται στη λειτουργία του ICSP προγραμματισμού.
- Το pin με τον χαρακτηρισμό $V_{PP}$ εφαρμόζεται η τάση προγραμματισμού.
- Το pin με τον χαρακτηρισμό $V_{REF}$ εφαρμόζεται η εξωτερική τάση αναφοράς του A/D μετατροπέα.
- Το pin με τον χαρακτηρισμό $V_{SS}$ εφαρμόζεται η γείωση του μικροελεγκτή.
- Το pin με τον χαρακτηρισμό $V_{DD}$ εφαρμόζεται η τάση λειτουργίας του μικροελεγκτή.

## 2.3 ΤΟ RESET ΚΑΙ ΟΙ ΔΙΑΚΟΠΕΣ

Υπάρχουν διάφοροι τρόποι με τους οποίους μπορεί να προκληθεί reset στον PIC16F628a. Ο λόγος για τον οποίο γίνεται ένα reset, είναι για να προστατευτεί ο μικροελεγκτής από απροσδιόριστες καταστάσεις. Μετά το reset οι τιμές των μεταβλητών χάνονται και ο καταχωρητής PC μηδενίζεται. Μερικοί από αυτούς αναφέροντα παρακάτω.

- Κατά την παροχή της αρχικής τροφοδοσίας και λόγου του ότι πολλά από τα κυκλώματα που περιέχει ένας μικροελεγκτής δεν έχουν την ίδια τάση τροφοδοσίας, γίνεται ένα reset τύπου POR(Power On Reset) μέχρι να σταθεροποιηθεί η τάση και όλα τα κυκλώματα να μπορούν να λειτουργήσουν κανονικά.

- Σε περίπτωση μεγάλης πτώσης τάσης γίνεται ένα BOR(Brown Out Reset).

- Μετά από λήξη του χρονιστή WDT(Watchdog Timer)

- Σε περίπτωση που ο χρήστης θελήσει να κάνει ο ίδιος Reset, αυτό μπορεί να το πετύχει με την ακίδα MCLR.

Σε έναν μικροελεγκτή όπως και σε έναν επεξεργαστή υπάρχει η δυνατότητα να προκληθούν διακοπές. Κατά την διάρκεια μιας διακοπής η διεύθυνση 0004h φορτώνεται στον PC και εκτελείται ο κώδικας που ο χρήσης έχει ορίσει. Σε περίπτωσή που ο χρήστης θέλει, μπορεί να εκτελεστεί και εντολή τύπου GOTO δίνοντας έτσι στον μικροελεγκτή την δυνατότητα να εκτελέσει περισσότερες εντολές από αυτές που το επιτρέπει ο πίνακας διακοπών. Μια διακοπή μπορεί να προκληθεί είτε από την ακίδα INT του μικροελεγκτή είτε και από τις περιφερειακές του συσκευές. Οι καταχωρητές INTCON - PIE1 - PIR1 - PIE2 - PIR2 είναι αυτοί που κάνουν των χειρισμό των διακοπών και ελέγχουν το κατά πόσο μια διακοπή θα ακολουθηθεί ή θα παραληφθεί.

## 2.4 ΘΥΡΕΣ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ (I/O PORTS)

Ένα βασικό κομμάτι ενός μικροελεγκτή είναι η επικοινωνία του με άλλες συσκευές και κατ' επέκταση με το φυσικό περιβάλλον. Η επικοινωνία αυτή, επιτυγχάνεται δια μέσω των ακίδων του. Αναλυτικότερα, η επικοινωνία του PIC γίνεται με τις θύρες ( δια μέσω των ακίδων), όπου το ποσό ακίδων για κάθε πόρτα στην προκειμένη περίπτωση είναι οχτώ. Επιπλέον πρέπει να σημειωθεί ότι ένας PIC μπορεί να διαθέτει έως και πέντε θύρες. Ο PIC που χρησιμοποιήθηκε για την υλοποίηση της εργασίας αυτής (PIC16F628a), διαθέτει δύο πόρτες την PORTA και PORTB. Στην PORTA οι ακίδες που αντιστοιχούν είναι η RA0, RA1, RA2, RA3, RA4, RA5, RA6 και η RA7. Στην PORTB οι ακίδες που αντιστοιχούν είναι RB0, RB1, RB2, RB3, RB4, RB5, RB6 και η RB7 αντίστοιχα. Οι ακροδέκτες επειδή είναι δυνατόν να χρησιμοποιούνται και από άλλα περιφερειακά συστήματα δεν μπορούν να χρησιμοποιηθούν με σκοπό εισόδου/ εξόδου γενικής χρήσεως. Στην εκάστοτε θύρα ενός μικροελεγκτή αντιστοιχεί ένας καταχωρητής , ο οποίος έχει την ίδια ονομασία με την πόρτα στην οποία ανήκει. Για παράδειγμα για την θύρα PORTA ο καταχωρητής που της ανήκει ονομάζεται και αυτός PORTA. Όσο αφορά τον PIC16F628a που οι θύρες του ονομάζονται PORTA και PORTB, ισχύει το ίδιο και ως αποτέλεσμα οι καταχωρητές ονομάζονται PORTA και PORTB για την κάθε θύρα αντίστοιχα. Η κατάλληλη τροποποίηση του αντίστοιχου καταχωρητή TRIS δίνει την δυνατότητα στον χειριστή να επιλέγει την θύρα που θα λειτουργεί σαν είσοδος ή έξοδος. Για την εκάστοτε PORTx υπάρχει και ο αντίστοιχος καταχωρητής (TRISx). Η επιλογή μιας πόρτας για είσοδο, γίνεται βάζοντας ένα ψηφίο στον καταχωρητή TRIS. Σε περίπτωση που ο χειριστής θέλει να χρησιμοποιήσει μία πόρτα σαν έξοδο, τότε πρέπει να τοποθετήσει ένα ψηφίο ίσο του μηδενός, έτσι ώστε η πόρτα που αντιστοιχεί σε αυτόν να γίνει πλέον έξοδος. Η ακίδα μίας θύρας μπορεί να οριστεί ανεξάρτητα από τις άλλες, με αποτέλεσμα κάποιες από τις ακίδες μιας θύρας να λειτουργούν ως είσοδοι και κάποιες από τις υπόλοιπες σαν έξοδοι. Τα δεδομένα τα οποία πρέπει να σταλούν στην έξοδο μιας θύρας, αναγράφονται στον PORTx της θύρας , ο οποίος όμως εξυπηρετεί και στο διάβασμα των δεδομένων που προέρχονται από την είσοδο. Όλα τα δεδομένα τα οποία στέλνονται προς την έξοδο διασφαλίζονται και παραμένουν εκεί μέχρι να αλλάξει μέσω του Data Latch , το οποίο είναι ένα ψηφιακό κύκλωμα. Πρέπει να

αναφερθεί ότι, σε περίπτωση που μια θύρα διαβάζει δεδομένα τότε ουσιαστικά οι πληροφορίες διαβάζονται από την αντίστοιχη ακίδα και όχι από το ψηφιακό κύκλωμα Data Latch.

## 2.5 ΧΡΟΝΙΣΤΕΣ/ ΑΠΑΡΙΘΜΗΤΕΣ

Οι Χρονιστές/ Απαριθμητές είναι μια περιφερειακή μονάδα, η οποία είτε αυξάνει την τιμή της κάθε ένα συγκεκριμένο χρονικό διάστημα, είτε μετράει την ποσότητα των παλμών που δέχεται ένα pin του μικροελεγκτή. Χρησιμοποιούνται συνήθως είτε όταν απαιτείται ο μικροελεγκτής να ενεργήσει με κάποιον τρόπο μετά το πέρας κάποιου χρόνου (π.χ αυτόματο πότισμα), είτε για τη καταμέτρηση παραδείγματος χάριν των φορών που άνοιξε η πόρτα ενός μαγαζιού. Ο μικροελεγκτής που χρησιμοποιήθηκε στην εργασία διαθέτει τρία κυκλώματα Χρονιστή/Απαριθμητή τα οποία είναι ανεξάρτητα μεταξύ τους ( TIMER0, TIMER1, TIMER2). Ο TIMER0 διαθέτει μετρητή μήκους 8bits. Περιέχει έναν προγραμματιζόμενο προμετρητή, ικανό να διαιρέσει με τις δυνάμεις του 2 έως 256. Αναλυτικότερα, μπορεί να προσαυξηθεί 2,4,8,...,256 κύκλους μηχανής, χρησιμοποιώντας είτε εσωτερικό είτε εξωτερικό ρολόι. Σε περίπτωση που γίνει υπερχείλιση μπορεί να προκληθεί διακοπή. Ο TIMER1 σε αντίθεση με τον TIMER0 διαθέτει μετρητή μήκους 16bits. Αποτελείται από δύο 8-ψήφιους καταχωρητές (TMR1H και TMRIL), σαν απαριθμητής προσαυξάνει από την ακίδα T1CKI, διαθέτει και αυτός ένα προμετρητή. Ο TIMER2 διαθέτει έναν μετρητή των 8bits. Ο προαναφερθείς μετρητής έχει 2 προμετρητές, η διαφορά μεταξύ αυτών των δύο είναι, ότι ο ένας διαιρεί τους παλμούς της εισόδου, ενώ ο δεύτερος το αποτέλεσμα της μέτρησης. Ο TIMER2, είναι ένα από τα βασικότερα στοιχεία της περιφερειακής μονάδας CCP.

## 2.6 ΠΕΡΙΦΕΡΕΙΑΚΗ ΜΟΝΑΔΑ CAPTURE /COMPARE/PULSE WITH MODULATION (CCP)

Εύκολα μπορεί να κατανοήσει κάποιος ότι αυτή η περιφερειακή μονάδα, μπορεί να λειτουργήσει με τρεις διαφορετικούς τρόπους, οι οποίοι είναι η μονάδα σύλληψης, σύγκρισης και παραγωγής παλμού. Τις περισσότερες φορές ένας PIC διαθέτει δύο τέτοιες μονάδες. Ο PIC16F628a έχει μόνο μία, την CCP1, η οποία ελέγχεται από τους καταχωρητές: CCP1CON, PWM1CON, CCPR1L, CCPR1H, ECCPAS, PSTRCON, ενώ για επικοινωνία χρησιμοποιεί το pin CCP1. Σε κάθε λειτουργία της η μονάδα CCP χρησιμοποίει και έναν από τους χρονιστές που παρουσιάστηκαν στο προηγούμενο υποκεφάλαιο. Παρακάτω παρουσιάζονται με λίγα λόγια οι τρεις λειτουργίες της CCP:

Σύλληψη:

Αυτή η λειτουργία χρησιμοποιείται για να "διαβάσει" την τιμή του καταχωρητή TMR1 του Timer1, όταν η μονάδα CCP δέχεται σήμα στην ακίδα CCP1 και αποθηκεύει το αποτέλεσμα στους καταχωρητές CCPR1L, CCPR1H. Η λειτουργία αυτή μπορεί να ενεργοποιηθεί είτε στην αρχή είτε στο τέλος του παλμού.

Σύγκρισης:

Σε αυτήν την λειτουργία, η μονάδα CCP συγκρίνει την τιμή των καταχωρητών CCPR1L, CCPR1H με την τιμή του καταχωρητή TMR1 και θέτει την ακίδα CCP1 ανάλογα με την ρύθμιση που της έχει γίνει σε λογικό 0 ή σε λογικό 1.

Παραγωγή παλμών:

Στην λειτουργία αυτή παράγεται μια σειρά από παλμούς διαμορφωμένη σε σχέση με την συχνότητα, και το duty cycle. Το duty cycle μιας παλμοσειράς ορίζεται ως λόγος της χρονικής διάρκειας ενός παλμού ως προς την συνολική περίοδο της παλμοσειράς και μετράται με επί τοις εκατό %. Με την γρήγορη εναλλαγή μεταξύ λογικού 0 και λογικού 1 μπορεί να δοθεί η δυνατότητα σε ένα ψηφιακό σύστημα απόδοση μια ψευδοαναλογική τιμή ικανή για τον έλεγχο οδήγησης κινητήρων, καταγραφή γεγονότων (events), μετρήσεις χρονικών περιόδων κτλ.

## 2.7 Η ΠΕΡΙΦΕΡΕΙΑΚΗ ΜΟΝΑΔΑ A/D CONVERTER

Ο A/D Converter (Analog-to-digital Converter - Μετατροπέας Αναλογικού σήματος σε Ψηφιακό) μετατρέπει ένα αναλογικό σήμα σε ψηφιακό. Ο A/D Converter λειτουργεί με τον ίδιο τρόπο σε σχεδόν όλους τους μικροελεγκτές. Η ρυθμίσεις του γίνονται στους καταχωρητές ADCON0, ADCON1, ADRES (ADRESH, ADRESL). Ο A/D Converter αποτελείται από έναν πολυπλέκτη, ο οποίος χρησιμοποιείται για να επιλεχθεί ένα από τα αναλογικά κανάλια μικροελεγκτή. Τα τέσσερα ψηφία CHS<3:0> του καταχωρητή ADCON0 κάνουν την επιλογή του καναλιού. Είναι σημαντικό ο χρήστης να προνοήσει και να αφήσει στον μετατροπέα λίγο χρόνο μετά τον έλεγχο μιας τιμής για την σταθεροποίηση του κυκλώματος. Ο μετατροπέας, μετά την ολοκλήρωση της μετατροπής αποθηκεύει την ψηφιακή τιμή μεγέθους 10-bit στον καταχωρητή ADRESH και ADRESL. Στον ADRESH αποθηκεύονται τα 8 πιο σημαντικά (MSB) bit του αποτελέσματος και στον ADRESL τα 8 λιγότερο σημαντικά (LSB) bit.

## Βιβλιογραφία Κεφαλαίου

<<Α>> "Εισαγωγή στους μικροελεγκτές PICmicro" Σταμάτης Αλατσαθιανός
<<Β>> "Ανάπτυξη Συστημάτων με Μικροελεγκτές 8051" Σταμάτης Αλατσαθιανός
<<Γ>> "Microcontroller Programming" Julio Sanchez, Maria P. Canton
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf
http://www.pi-schools.gr/lessons/tee/electronic/biblia.php
http://www.gooligum.com.au/tutorials/baseline/PIC_Base_C_5.pdf
http://www.gooligum.com.au/tutorials/baseline/PIC_Base_A_9.pdf
http://ww1.microchip.com/downloads/en/AppNotes/00879D.pdf
http://www.best-microcontroller-projects.com/pic-microcontroller.html

## Εικόνες Κεφαλαίου

Σχήμα 2.1: Μπλοκ διάγραμμα μικροελεγκτή PIC16F628a
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf
Σχήμα 2.2: Γράφημα ακίδων PIC16F628a
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf

Κεφάλαιο 3

## 3.1 ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό, θα αναλυθεί η λειτουργία του αυτοκινήτου. Στην αρχή, θα παρατεθεί η συνδεσμολογία του μικροελεγκτή με τα περιφερειακά του, ο λόγος που χρησιμοποιήθηκαν, καθώς και ο τρόπος λειτουργίας των περιφερειακών. Επιπλέον, θα γίνει αναφορά στον τρόπο με τον οποίο επιτυγχάνεται η ασύρματη επικοινωνία αλλά και ο συγχρονισμός μεταξύ αμαξιού και υπολογιστή. Τέλος θα αναφερθεί ο κώδικας που θα συνοδέψει τον μικροελεγκτή, για τον σωστό χειρισμό των περιφερειακών μονάδων, καθώς και η ανάλυση αυτού.

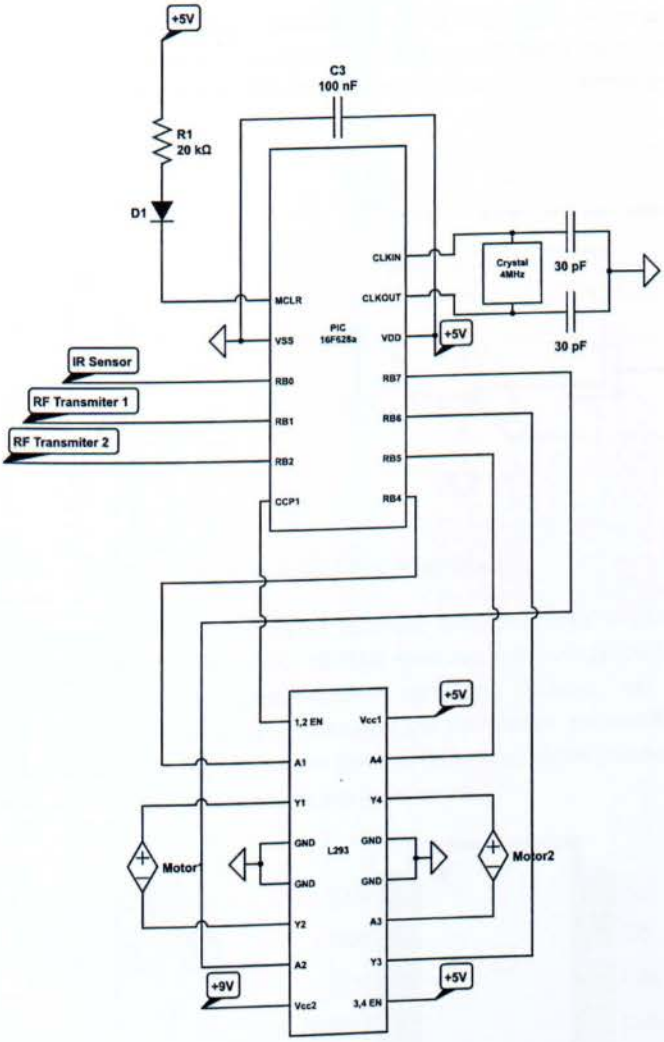
### 3.1.2 ΛΕΙΤΟΥΡΓΙΑ ΕΡΓΑΣΙΑΣ

Παρακάτω θα περιγραφεί ο τρόπος με τον οποίο επιτυγχάνει το αμάξι να γίνει το παράλληλο παρκάρισμα αλλά και ο συγχρονισμός του ηλεκτρονικού υπολογιστή με τις κινήσεις του αμαξιού. Ένα από τα πολύ βασικά εξαρτήματα για την επίτευξη αυτού, είναι ο αισθητήρας υπερύθρων που σαν εργασία έχει να ενημερώνει το αμάξι για την κατάσταση τις πλαϊνής του θέσης. Ο τρόπος με τον οποίο είναι σε θέση να το κάνει θα αναλυθεί στο κεφάλαιο 3.3. Έχοντας ως δεδομένο, ότι ο μικροελεγκτής είναι σε θέση να "διαβάσει" αν υπάρχει εμπόδιο ή εάν η θέση είναι ελεύθερη, θα προβεί σε διάφορές λειτουργίες για να επιβεβαιώσει το μέγεθος της. Με μια συγκεκριμένη ταχύτητα προς τα εμπρός και με το έλεγχο για την κατάσταση της ελεύθερης θέσης στο πλάι του, να επαναλαμβάνεται έλεγχος κάθε περίπου 50msec δηλαδή περίπου 20 φορές το δευτερόλεπτο, κατά πόσο η ελεύθερη αυτή θέση μπορεί να χωρέσει το μήκος του αυτοκινήτου. Καθ' όλη την διάρκεια του έλεγχου αυτού, ο μικροελεγκτής στέλνει μέσω του ασύρματου πομπού ένα σήμα που μετά θα κωδικοποιηθεί στο γράμμα 'Y'. Η πληροφορία αυτή καθιστά ικανό τον υπολογιστή να γνωρίζει την κατάσταση της θέσης. Όταν το αμάξι "διαβάσει", ότι ο αισθητήρας δεν ανιχνεύει κάποιο εμπόδιο για ένα συνεχόμενο αριθμό φορών (21) αναγνωρίζει, ότι η θέση είναι αρκετά μεγάλη και ξεκινάει την διαδικασία παρκαρίσματος.

Ένα παρόμοιο σήμα με αυτό που στέλνει ο μικροελεγκτής για την κατάσταση της θέσης παρκινγκ, στέλνει και στο ξεκίνημα της διαδικασίας παρκαρίσματος. Αυτή την φορά θα κωδικοποιηθεί στο γράμμα 'P'. Ο υπολογιστής την στιγμή που θα διαβάσει αυτήν την πληροφορία θα ξεκινήσει και αυτός την διαδικασία παρκαρίσματος. Δεδομένου, ότι τα σήματα 'Y' και 'P' δεν θα χρησιμοποιηθούν ξανά από την λειτουργία του προγράμματος και πλέον η χρήση τους δεν θα ήταν αναγκαία, θα αξιοποιηθούν στον συγχρονισμό των κινήσεων υπολογιστή και αμαξιού. Το αμάξι μετά το ολοκλήρωμα μίας από της προκαθορισμένες κινήσεις του παρκαρίσματος, θα αποστέλλει ένα από αυτά τα σήματα στον υπολογιστή επιτρέποντας του έτσι να προχωρήσει στην επόμενη κίνηση. Η αποστολή αυτών των σημάτων γίνεται εναλλάξ για την αποφυγή παρερμηνείας του σήματος ολοκλήρωσης κίνησης. Το πρώτο από τα δύο σήματα είναι το 'Y' μιας και εφόσον το 'P' είχε μόλις χρησιμοποιηθεί για την ένδειξη έναρξης της λειτουργίας παρκαρίσματος και ενδέχεται η παρερμηνεία του σε σήμα ολοκλήρωσης κίνησης.

## 3.2 Ο ΜΙΚΡΟΕΛΕΓΚΤΗΣ ΚΑΙ Η ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΤΟΥ ΜΕ ΤΗΣ ΠΕΡΙΦΕΡΕΙΑΚΕΣ ΜΟΝΑΔΕΣ ΤΟΥ ΑΜΑΞΙΟΥ

Ο βασικός σκοπός του μικροελεγκτή σε αυτό το κομμάτι της εργασίας είναι, να μπορεί να αναγνωρίζει αν υπάρχει κάποιο εμπόδιο στο πλαϊνό του μέρος και παράλληλα να μπορεί να οδηγήσει τους κινητήρες του αμαξιού με την σωστή φορά. Αυτό επιτυγχάνεται με την τοποθέτηση ενός κυκλώματος υπερύθρων, με το οποίο θα μπορεί να ανιχνεύει την ύπαρξή ή όχι εμποδίου στο πλαϊνό του μέρος. Το άλλο μεγάλο εμπόδιο στην ολοκλήρωση του κυκλώματος είναι η οδήγησή των κινητήρων του αυτοκινήτου. Λόγω του χαμηλού ρεύματος εξόδου που μπορεί να παρέχει ο μικροελεγκτής, το οποίο δεν έχει την ένταση για την οδήγηση των μοτέρ, θα τοποθετηθούν τέσσερα τρανζίστορ, ένα σε κάθε είσοδο των μοτέρ, για την ελεγχόμενη βραχυκύκλωση της μπαταρίας με τα μοτέρ και την οδήγησή τους. Την θέση αυτών των τεσσάρων τρανζίστορ θα πάρει το ολοκληρωμένο L293DN, της Texas Instrument, το οποίο προσφέρει μια πιο ολοκληρωμένη λύση. Τέλος, θα τοποθετηθεί ο αποστολέας της ασύρματης επικοινωνίας για την αποστολή πληροφοριών στον ηλεκτρονικό υπολογιστή. Η σύνδεση αυτή φαίνεται στο παρακάτω σχήμα (σχήμα 3.2.α).

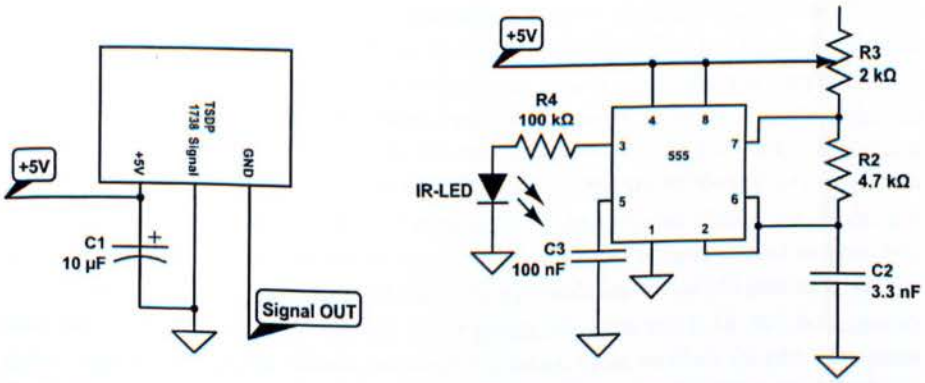



Σχήμα 3.2.a: Σχήμα σύνδεσης περιφερειακών μονάδων στον 16F628a

## 3.3 ΑΙΣΘΗΤΗΡΑΣ ΥΠΕΡΥΘΡΩΝ (IR)

Ο έλεγχος τον εμποδίων επιτυγχάνεται με την χρήση ενός LED υπερύθρων και την αναζήτηση της αντανάκλασης του πάνω στο εμπόδιο, εάν αυτό υπάρχει, από τον δέκτη. Αναλυτικότερα στο πάνω μέρος το αμαξιού θα τοποθετηθεί ένα υπέρυθρο LED με κατεύθυνση προς το πλάι του αυτοκινήτου για να "φωτίζει" όποιο εμπόδιο τυχόν βρίσκεται εκεί. Επίσης πάνω στο αυτοκίνητο θα τοποθετηθεί και ένας δέκτης-IR για την τυχόν εύρεση της αντανάκλασης. Σε περίπτωση που ο δέκτης δεχθεί φως σημάνει ότι υπάρχει κάτι για να αντανακλαστεί το IR φως πάνω του, το οποίο με την σειρά σημαίνει, ότι υπάρχει εμπόδιο. Τότε ο αισθητήρας θα επιστέψει +5V ή αλλιώς το λογικό "1" πίσω στον μικροελεγκτή. Σε περίπτωση που δεν υπάρχει αντανάκλαση η επιστροφή θα είναι 0V ή αλλιώς λογικό "0". Για

την επιβεβαίωση, ότι ο δέκτης δεν εκλαμβάνει σαν αντανάκλαση και ουσιαστικά εμπόδιο, το υπέρυθρο φως που υπάρχει στο περιβάλλον από άλλες πηγές εκτός του LED υπερύθρων, το IR-LED ρυθμίζεται σε συχνότητα 40Hz-συχνότητα κατά την οποία δέχεται σαν είσοδο ο δέκτης.




Σχήμα 3.3a: Αισθητήρας Υπερύθρων

Για την συχνότητα με την οποία λειτουργεί το IR-LED είναι υπεύθυνο το ολοκληρωμένο NE555N. Το ολοκληρωμένο κύκλωμα (IC) NE555N είναι ένα τσιπ που χρησιμοποιείται σε μια ποικιλία λειτουργιών, από χρονοδιακόπτη, γεννήτρια παλμού, και εφαρμογές ταλαντωτή. Το NE555N μπορεί να χρησιμοποιηθεί για να παρέχει χρονοκαθυστερήσεις, όπως ένας ταλαντωτής, αλλά και ως στοιχείο flip-flop. Παράγωγα ολοκληρωμένα παρέχουν μέχρι και τέσσερα κυκλώματα χρονισμού σε ένα μόνο πακέτο.

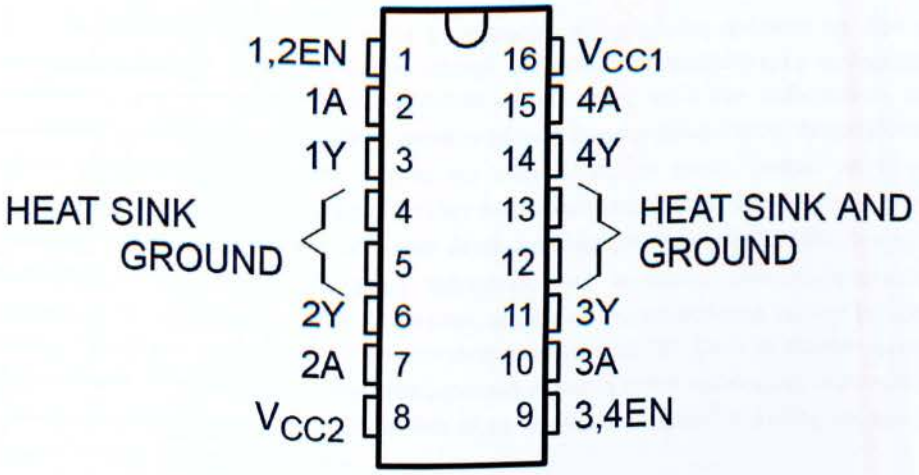

Εικόνα 3.3.b: NE555N IC


Εικόνα 3.3.c: NE555N Pin-out

Την συχνότητα λειτουργίας, την οποία θα θέσει το NE555N το IR-LED, ορίζεται βάση του λόγου ανάμεσα στην αντίσταση R3 και R2 (σχήμα 3.3.a ). Ένα ποτενσιόμετρο-τρίμερ 2ΚΩ στην θέση της αντίστασης R2 μας εγγυάται την εύκολη εύρεση του λόγου αυτού.

### 3.2.4 ΟΔΗΓΗΣΗ ΚΙΝΙΤΗΡΩΝ ΜΕ ΤΟ L293DN

Η κίνηση των κινητήρων σε όλες τις εφαρμογές είναι δύσκολο να υλοποιηθεί χωρίς την χρήση κάποιας άλλης περιφερειακής συσκευής. Ο λόγος είναι, ότι όλοι οι μικροελεγκτές είναι σχεδιασμένοι με τέτοιον τρόπο ώστε να καταναλώνουν όσο τον δυνατόν λιγότερο ρεύμα. Ένα μοτέρ, για να μπορέσει να λειτουργήσει σωστά, εκτός από την διαφορά τάσης την οποία πρέπει να έχει στους ακροδέκτες του, θα πρέπει να τροφοδοτείται και με αρκετό ρεύμα. Ανάλογα το μέγεθος του μοτέρ και τον τρόπο λειτουργίας του, αυτά τα στοιχεία διαφέρουν αλλά σχεδόν πάντα όταν ένας μικροελεγκτής πρέπει να οδηγήσει ένα κινητήρα χρησιμοποιεί κάποιο είδος οδηγού. Ο οδηγός αυτός δέχεται σαν είσοδο την έξοδο του μικροελεγκτή και πράττει ανάλογα με αυτήν. Το κύκλωμα του οδηγού μπορεί να είναι δύο τρανζίστορ με την βάση τους στον μικροελεγκτή και συνδεδεμένα με μία μπαταρία και τον έναν ακροδέκτη του μοτέρ. Ο τρόπος αυτός χρησιμοποιείται πιο πολύ από βιομηχανικό τρόπο παραγωγής μιας και είναι ο πιο φθηνός. Μπορεί όμως να είναι και κάτι άλλο πολύ πιο περίπλοκο. Σε αυτήν την εργασία τον ρόλο των τρανζίστορ τον έχει πάρει το L293DN της Texas Instrument. Εκτός από την οδήγηση των δύο μοτέρ, δίνει την δυνατότητα για εύκολο έλεγχο της ταχύτητάς τους. Το ίδιο πράγμα θα μπορούσε να υλοποιηθεί με την χρήση τεσσάρων πυλών τύπου AND.




Σχήμα 3.2.4.a: Pin-Out του L293DN

Όπως φαίνεται και στο παραπάνω σχήμα, το L293DN έχει δύο εισόδους Vcc. Το Vcc1, αφορά την τροφοδοσία του ολοκληρωμένου με +5V. Το Vcc2 αναφέρεται στην τάση τροφοδοσίας που θα παρέχεται στα μοτέρ. Για απλούστευση το Vcc2 θα αναφέρεται ως εξής ως Vmotor. Τα ποδαράκια(pin) 2, 7, 10 και 15 αναφέρονται στις εισόδους του L293DN. Τα 3, 6, 11 και 14 είναι οι αντίστοιχες έξοδοί του. Για παράδειγμα, εάν στο pin-2(1A) ανιχνευθεί είσοδος +5V, το L293DN θα βγάλει στο pin-3(1Y) το Vmotor με το οποίο έχει τροφοδοτηθεί. Τα pin 4, 5, 12 και 13 είναι εκείνα στα οποία συνδέεται η γείωση για το ολοκληρωμένο αλλά είναι και με τέτοιον τρόπο σχεδιασμένο, ώστε να γίνεται η μετάδοση από την αρκετά μεγάλη θερμότητα την οποία αναπτύσσει. Μία heat-sink θα ήταν καλό να

εφαρμοστεί σε αυτά τα σημεία ώστε να αποφευχθεί η καταστροφή του ολοκληρωμένου ή μετάβαση του σε συνθήκες απροσδιόριστης κατάστασης. Τέλος, τα pin 1 και 9 είναι αυτά με τα οποία μπορεί να γίνει ο έλεγχος της ταχύτητας των κινητήρων. Ο καθένας από τους δύο ακροδέκτες ελέγχει δύο εξόδους παράλληλα. Οι έξοδοι λειτουργούν σαν μια πύλη AND. Εφόσον υπάρχει είσοδος, αλλά και ο ακροδέκτης ENx,y είναι ενεργός, τότε θα υπάρξει και σήμα εξόδου. Ο τρόπος με τον οποίο γίνεται ο έλεγχος της ταχύτητας είναι εφαρμόζοντας ένα τετραγωνικό παλμό και εναλλάσσοντας το ENx,y από κατάσταση λειτουργίας σε κατάσταση αποκοπής. Ο πίνακας με τις καταστάσεις λειτουργίας φαίνεται παρακάτω.

| INPUTS† | | OUTPUT |
|---|---|---|
| A | EN | Y |
| H | H | H |
| L | H | L |
| X | L | L |

Σχήμα 3.2.4.b: Πίνακας καταστάσεων λειτουργίας του L293DN

### 3.2.5 ΑΣΥΡΜΑΤΗ ΕΠΙΚΟΙΝΩΝΙΑ

Η ασύρματη επικοινωνία είναι η μεταφορά πληροφορίας ανάμεσα σε δύο ή περισσότερα σημεία χωρίς την χρήση κοινού καλωδίου. Οι περισσότερες ασύρματες τεχνολογίες χρησιμοποιούν ηλεκτρομαγνητικά κύματα όπως αυτά του ραδιοφώνου. Οι αποστάσεις μπορεί να είναι της τάξης μερικών μέτρων έως και χιλιομέτρων. Απαρτίζονται πάντα από δύο κομμάτια, έναν πομπό και έναν δέκτη, οι οποίοι μπορεί να είναι συνδυασμένοι μεταξύ τους (πομποδέκτης) για αμφίδρομη επικοινωνία. Στην εργασία αυτή χρησιμοποιήθηκε ένα ζευγάρι πομπού και δέκτη (μονόδρομη επικοινωνία), που περιέχει κωδικοποιητή για ασφαλή μεταφορά δεδομένων από παράσιτα ηλεκτρομαγνητικών κυμάτων του περιβάλλοντα χώρου. Ο πομπός εκτός από την τροφοδοσία και την γείωση παρέχει τέσσερεις εισόδους με κατάσταση αναμονής λογικού "1". Όταν οι είσοδοι αυτοί γίνουν λογικό "0" από κάποια άλλη περιφερειακή συσκευή (στην προκειμένη περίπτωση από τον μικροελεγκτή) αναμεταδίδεται ένα σήμα, το οποίο λαμβάνει ο δέκτης και κάνει λογικό "1" την αντίστοιχη έξοδό του.

## 3.3 FIRMWARE ΜΙΚΡΟΕΛΕΓΚΤΗ ΑΜΑΞΙΟΥ

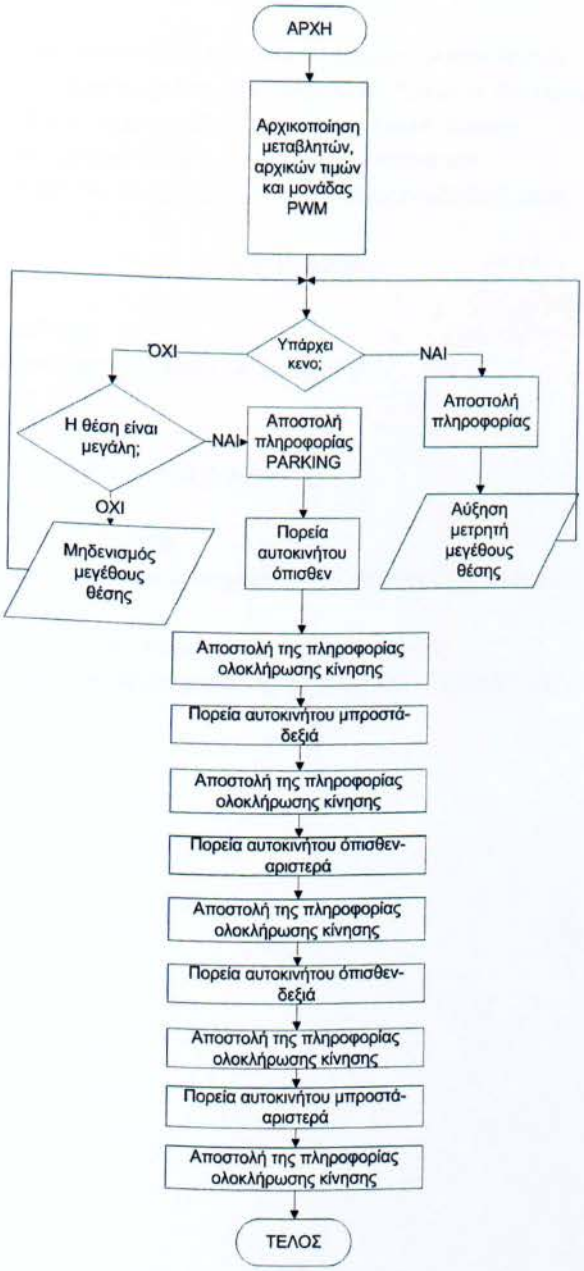
Ακολουθεί ο κώδικας που εκτελεί ο μικροελεγκτής, για να μπορέσει να αξιοποιήσει σωστά όλες τις περιφερειακές μονάδες που τον απαρτίζουν.

```
program CarParkingo                 Έναρξη του προγράμματος
dim k as byte                       Αρχικοποίηση μεταβλητής τύπου Byte με όνομα k
dim flag as byte                    Αρχικοποίηση μεταβλητής τύπου Byte με όνομα flag


main:
      delay_ms(5000)                Χρονοκαθυστέρηση 5sec στην αρχή του προγράμματος

      k=0                           Τίθεται αρχική τιμή στο k=0
      TRISB = %00000001             Τίθενται τα δύο πρώτα bit τις πόρτας B ως εισόδους ενώ
                                    τα υπόλοιπα  6 σαν εξόδους
      RB1_bit = 1                   Αρχικοποίηση τιμών στης πόρτας B
      RB2_bit = 1
      RB3_bit = 0
      RB4_bit = 0
      RB5_bit = 0
      RB6_bit = 0
      RB7_bit = 0

      PWM1_Init(4480)               Αρχικοποίηση του Pin 8(CCP1) του 16F628a σαν μονάδα
                                    γεννήτριας παλμού με συχνότητα στα 60Hz
      PWM1_Start()                  Ενεργοποίηση της μονάδας γεννήτριας παλμού
      PWM1_Set_Duty(180)            Τίθεται το Duty Cycle στο 70%
      While (True)                  Αιώνια εκτελέσιμη λούπα
  if (flag=0) then                  Εκτέλεση αν δεν έχει βρεθεί θέση parking
    RB4_bit = 0                     {Τίθενται οι κινητήρες στην ευθεία πορεία
    RB5_bit = 0
    RB6_bit = 0
    RB7_bit = 1
  end if
  While (RB0_bit = 1)               Εφόσον κομμάτι θέσης parking είναι διαθέσιμο
    RB2_bit = 0                     {Αποστολή πληροφορίας μέσο ασύρματου πομπού
    k = k + 1                       Αύξηση μεταβλητής k (μέγεθος θέσης parking)
    flag = 1                        Έγερση σημαίας διαθέσιμης  θέσης parking
    Delay_ms(50)                    Χρονοκαθυστέρηση για την αποστολή της πληροφορίας
                                    μέσο του ασύρματου πομπού}
  Wend

  RB2_bit = 1                       Διακοπή αποστολής διαθέσιμης θέσης parking


  if (k > 21) then                  Έλεγχος μεγέθους διαθέσιμης θέσης parking

    RB1_bit = 0                     Αποστολή πληροφορίας ότι η διαθέσιμη θέση parking
                                    είναι αρκετά μεγάλη. Εκκίνηση παρκαρίσματος.
    PWM1_Set_Duty(210)              Τίθεται το Duty Cycle στο 82%

    RB4_bit = 1                     Φρενάρισμα του μοτέρ πορείας
    RB7_bit = 1
```

```
Delay_ms(1000)          Χρονοκαθυστέρηση 1sec
RB1_bit = 1             Διακοπή αποστολής εκκίνησης παρκαρίσματος

RB4_bit = 1             Πορεία αυτοκινήτου όπισθεν
RB5_bit = 0
RB6_bit = 0
RB7_bit = 0
Delay_ms(510)           Χρονοκαθυστέρηση 510msec για την κίνηση του
                        αυτοκινήτου με προαναφερθέντα πορεία
RB4_bit = 1             Φρενάρισμα του μοτέρ πορείας
RB7_bit = 1
Delay_ms(1000)          Χρονοκαθυστέρηση 1sec
RB4_bit = 0
RB5_bit = 0
RB6_bit = 0
RB7_bit = 0

RB2_bit = 0             Χρονοκαθυστέρηση 1sec για την αποστολή της
Delay_ms(1000)          πληροφορίας ολοκλήρωσης κίνησης μέσο του
RB2_bit = 1             ασύρματου πομπού

RB4_bit = 0             Πορεία αυτοκινήτου μπροστά-δεξιά
RB5_bit = 1
RB6_bit = 0
RB7_bit = 1
Delay_ms(1000)          Χρονοκαθυστέρηση 1sec για την κίνηση του αυτοκινήτου
                        με προαναφερθέντα πορεία
RB4_bit = 0
RB5_bit = 0
RB6_bit = 0
RB7_bit = 0

RB1_bit = 0             Χρονοκαθυστέρηση 1sec για την αποστολή της
Delay_ms(1000)          πληροφορίας ολοκλήρωσης κίνησης μέσο του
RB1_bit = 1             ασύρματου πομπού

RB4_bit = 1             Πορεία αυτοκινήτου όπισθεν-αριστερά
RB5_bit = 0
RB6_bit = 1
RB7_bit = 0
Delay_ms(1350)          Χρονοκαθυστέρηση 1,35sec για την κίνηση του
                        αυτοκινήτου με προαναφερθέντα πορεία
RB4_bit = 0
RB5_bit = 0
RB6_bit = 0
RB7_bit = 0

RB2_bit = 0             Χρονοκαθυστέρηση 1sec για την αποστολή της
Delay_ms(1000)          πληροφορίας ολοκλήρωσης κίνησης μέσο του
```

```
    RB2_bit = 1                 ασύρματου πομπού

    RB4_bit = 1                 Πορεία αυτοκινήτου όπισθεν-δεξιά
    RB5_bit = 1
    RB6_bit = 0
    RB7_bit = 0
    Delay_ms(1350)              Χρονοκαθυστέρηση 1,35sec για την κίνηση του
                                αυτοκινήτου με προαναφερθέντα πορεία
    RB4_bit = 0
    RB5_bit = 0
    RB6_bit = 0
    RB7_bit = 0

    RB1_bit = 0                 Χρονοκαθυστέρηση 1sec για την αποστολή της
    Delay_ms(1000)              πληροφορίας ολοκλήρωσης κίνησης μέσο του
    RB1_bit = 1                 ασύρματου πομπού

    RB4_bit = 0                 Πορεία αυτοκινήτου μπροστά-αριστερά
    RB5_bit = 0
    RB6_bit = 1
    RB7_bit = 1
    Delay_ms(1000)              Χρονοκαθυστέρηση 1sec για την κίνηση του αυτοκινήτου
    RB4_bit = 0                 με προαναφερθέντα πορεία
    RB5_bit = 0
    RB6_bit = 0
    RB7_bit = 0
    Delay_ms(1000)              Χρονοκαθυστέρηση 1sec

    PORTB = %00000110           Όλες οι έξοδοι τίθενται στο λογικό "0"
    PWM1_stop()                 Απενεργοποίηση της μονάδας γεννήτριας παλμού
    Delay_ms(20000)             Χρονοκαθυστέρηση 20sec

  else

    k = 0                       Μηδενισμός μεταβλητής k (μέγεθος θέσης parking)
    flag = 0                    Υποβολή σημαίας διαθέσιμης θέσης parking
  end if

      wend                      Τέλος αιώνιας λούπας
end.                            Τέλος προγράμματος CarParkingo
```

Ακολουθεί το λογικό διάγραμμα του Firmware που εκτελεί ο PIC16F628a που βρίσκεται στο αμάξι(Σχήμα 3.3a).




Σχήμα 3.3α: Λογικό διάγραμμα Firmware αμαξιού

## Βιβλιογραφία Κεφαλαίου

<<Α>> "Εισαγωγή στους μικροελεγκτές PICmicro" Σταμάτης Αλατσαθιανός
<<Β>> "Ανάπτυξη Συστημάτων με Μικροελεγκτές 8051" Σταμάτης Αλατσαθιανός
<<Γ>> "Microcontroller Programming" Julio Sanchez, Maria P. Canton
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf
http://pdf1.alldatasheet.net/datasheet-pdf/view/26889/TI/L293DNE.html

## Εικόνες Κεφαλαίου

Εικόνα 3.3.b: NE555N IC
http://en.wikipedia.org/wiki/File:Signetics_NE555N.JPG

Εικόνα 3.3.c: NE555N Pin-out
http://en.wikipedia.org/wiki/File:555_Pinout.svg

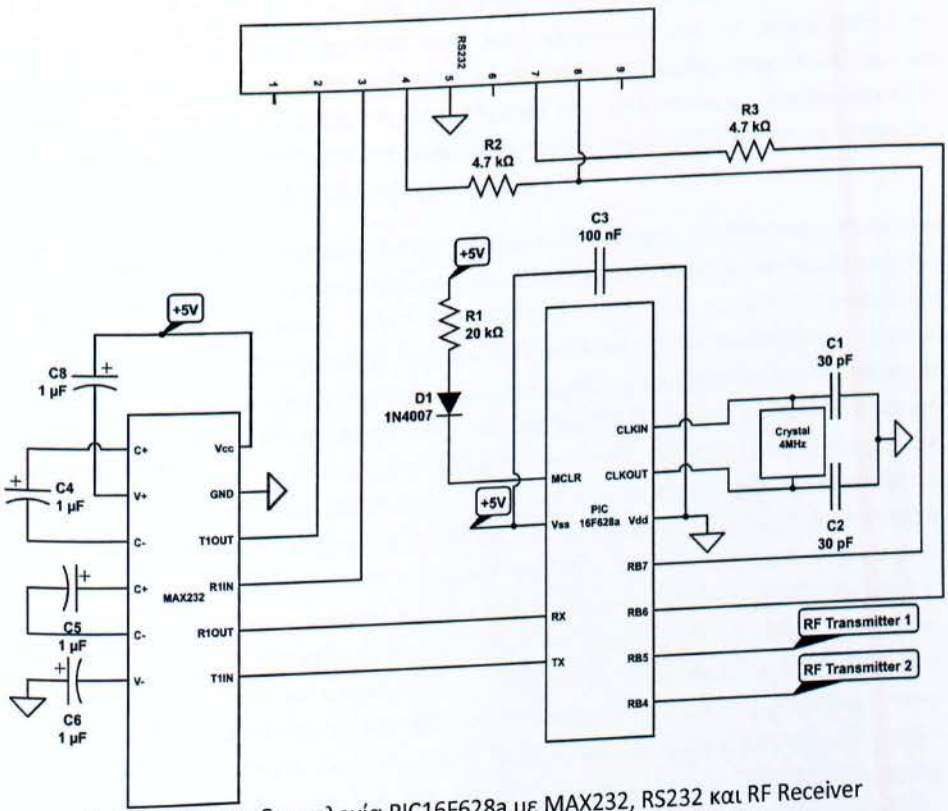
Σχήμα 3.2.4.a: Pin-Out του L293DN
http://pdf1.alldatasheet.net/datasheet-pdf/view/26889/TI/L293DNE.html

Σχήμα 3.2.4.b: Πίνακας καταστάσεων λειτουργίας του L293DN
http://pdf1.alldatasheet.net/datasheet-pdf/view/26889/TI/L293DNE.html

Κεφάλαιο 4

## 4.1 ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα αναλυθεί η λειτουργία και η συνδεσμολογία του περιφερειακού συστήματος το οποίο θα συνδεθεί στον ηλεκτρονικό υπολογιστή. Το σύστημα αυτό, θα είναι υπεύθυνο για την αναγνώριση των ασύρματων σημάτων που στέλνει ο μικροελεγκτής του αυτοκινήτου και την αναμετάδοση τους στον ηλεκτρονικό υπολογιστή μέσω της σειριακής θύρας RS232. Αρχικά, θα παρατεθεί ο τρόπος με τον οποίο επιτυγχάνεται η επικοινωνία ανάμεσα στον μικροελεγκτή και τον υπολογιστή με την χρήση του ολοκληρωμένου MAX232 και η συνδεσμολογία του μικροελεγκτή με τα περιφερειακά του. Τέλος θα αναφερθεί ο κώδικας που θα συνοδέψει τον μικροελεγκτή, καθώς και η ανάλυση αυτού.




Σχήμα 4.1.a: Συνδεσμολογία PIC16F628a με MAX232, RS232 και RF Receiver

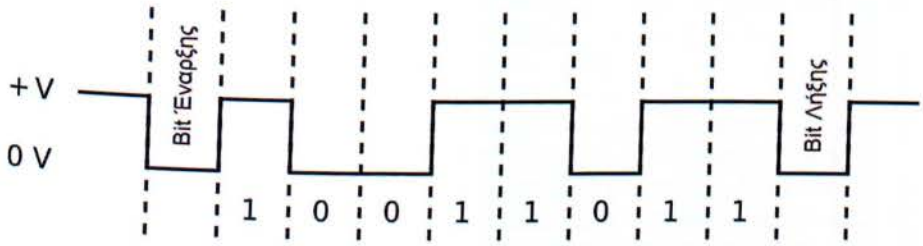
### 4.2.1 ΣΕΙΡΙΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ

Πολλές είναι οι εφαρμογές οι οποίες χρήζουν επικοινωνίας ανάμεσα σε δύο ή παραπάνω συσκευές. Ο πιο δημοφιλής και εύκολος τρόπος για την επίτευξη τέτοιας

επικοινωνίας είναι ο σειριακός. Η σειριακή επικοινωνία είναι η διαδικασία στην οποία η αποστολή της πληροφορίας γίνεται σταδιακά ένα bit την φορά μέσα από ένα δίαυλο. Αυτό έρχεται σε αντίθεση με την παράλληλη επικοινωνία, όπου διαφορετικά bits στέλνονται ως σύνολο σε δίαυλο με διάφορα παράλληλα κανάλια. Η σειριακή επικοινωνία χρησιμοποιείται για όλες τις μεγάλων αποστάσεων επικοινωνίες, όπως και στα περισσότερα δίκτυα υπολογιστών, όπου το κόστος του καλωδίου και ο συγχρονισμός κάνουν την παράλληλη επικοινωνία ανέφικτη. Οι σειριακοί δίαυλοι υπολογιστών χρησιμοποιούνται όλο και πιο συχνά, ακόμη και σε μικρές αποστάσεις. Η βελτιωμένη ακεραιότητα του σήματος και οι μεγαλύτερες ταχύτητες μετάδοσης νεότερης τεχνολογίας έχουν αρχίσει να αντισταθμίζουν την υπεροχή της παράλληλης επικοινωνίας. Η σειριακή επικοινωνία χωρίζεται σε δύο κατηγορίες. Την σύγχρονη (synchronous) και την ασύγχρονη (asynchronous) σειριακή επικοινωνία.

Η σύγχρονη σειριακή επικοινωνία λειτουργεί ανάμεσα σε έναν πομπό και έναν δέκτη που λειτουργούν με το ίδιο ρολόι. Σε ένα σύγχρονο σύστημα και τα δύο κομμάτια του ζεύγους έχουν μια "συνομιλία" πριν γίνει η μετάδοση των δεδομένων. Σε αυτήν στην συνομιλία συγχρονίζουν τα ρολόγια τους και συμφωνούν για τις παραμέτρους της μεταφοράς δεδομένων, συμπεριλαμβανομένου του χρονικού διαστήματος μεταξύ των bits δεδομένων. Οποιαδήποτε δεδομένα που πέφτουν έξω από αυτές τις παραμέτρους θα θεωρούνται λάθος. Ο δίαυλος σε μια σύγχρονη σειριακή επικοινωνία πρέπει να παραμένει συνέχεια ενεργός για να μην χαθεί ο συγχρονισμός.

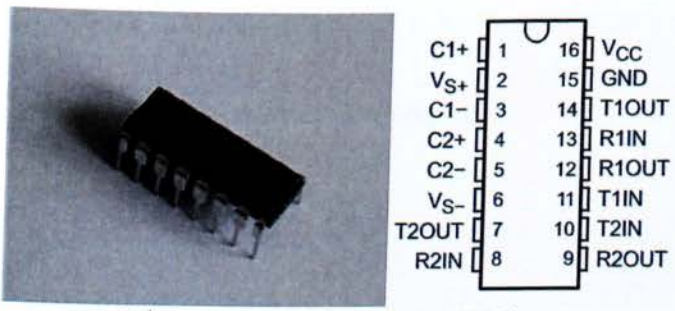
Οι περισσότερες σειριακές συσκευές όπως πληκτρολόγια, ποντίκια και μόντεμ είναι ασύγχρονες. Η ασύγχρονη επικοινωνία απαιτεί έναν πομπό, ένα δέκτη και ένα σύρμα που να τους συνδέει. Επομένως, είναι η απλούστερη από τα σειριακά πρωτόκολλα επικοινωνίας, και λιγότερο δαπανηρή για την εφαρμογή της. Όπως υποδηλώνει το όνομα, ασύγχρονη επικοινωνία γίνεται μεταξύ δύο (ή περισσοτέρων) συσκευών που λειτουργούν με ανεξάρτητα ρολόγια. Ως εκ τούτου, ακόμη και αν τα δύο ρολόγια συμφωνήσουν για μια στιγμή, δεν υπάρχει καμία εγγύηση ότι θα συνεχίσουν να συμφωνούν για μεγάλα χρονικά διαστήματα, και ως εκ τούτου δεν υπάρχει καμία εγγύηση ότι όταν αρχίζει να μεταδίδει το σημείο Α, το σημείο Β, θα αρχίσει να λαμβάνει. Για να καταπολεμηθεί αυτό το πρόβλημα συγχρονισμού, η ασύγχρονη επικοινωνία απαιτεί επιπλέον bits να προστεθούν γύρω από τα πραγματικά δεδομένα προκειμένου να διατηρηθεί η ακεραιότητα του σήματος. Προηγείται ένα bit έναρξης που δείχνει προς το δέκτη ότι μια λέξη (ένα κομμάτι των δεδομένων) είναι έτοιμο να ξεκινήσει. Για να αποφευχθεί η σύγχυση με άλλα bits, το δυαδικό ψηφίο εκκίνησης είναι διπλάσιο από το μέγεθος οποιουδήποτε άλλου bit στη μετάδοση. Για να διασφαλιστεί η ακεραιότητα των δεδομένων, ένα bit ισοτιμίας συχνά προστίθεται μεταξύ του τελευταίου bit δεδομένων και του stop bit. Το bit ισοτιμίας εξασφαλίζει ότι τα δεδομένα που λήφθηκαν αποτελούνται από τον ίδιο αριθμό των bits με την ίδια σειρά με την οποία είχαν αποσταλεί.




Σχήμα 4.2.a: Ασύγχρονη σειριακή επικοινωνία

### 4.2.2 ΤΟ MAX232

Το MAX232 είναι ένα ολοκληρωμένο κύκλωμα που δημιουργήθηκε για πρώτη φορά από την Atmel. Μετατρέπει τα σήματα από την RS-232 σειριακή θύρα, σε σήματα κατάλληλα για χρήση σε TTL συμβατά ψηφιακά λογικά κυκλώματα όπως ένας μικροελεγκτής. Είναι ένας διπλός πομπός και δέκτης και συνήθως μετατρέπει τα RX, TX, CTS και RTS σήματα. Το MAX232 παρέχει εξόδους τάσης (περίπου ± 7,5 V) από μία ενιαία +5V τροφοδοσία μέσω αντλιών φορτίου και εξωτερικούς πυκνωτές. Αυτό το καθιστά χρήσιμο για την υλοποίηση RS-232 σε συσκευές που δεν λειτουργούν με τάσεις παραπάνω από +5 V.




Σχήμα 4.2.2.a: Max 232 και Max 232 Pin out

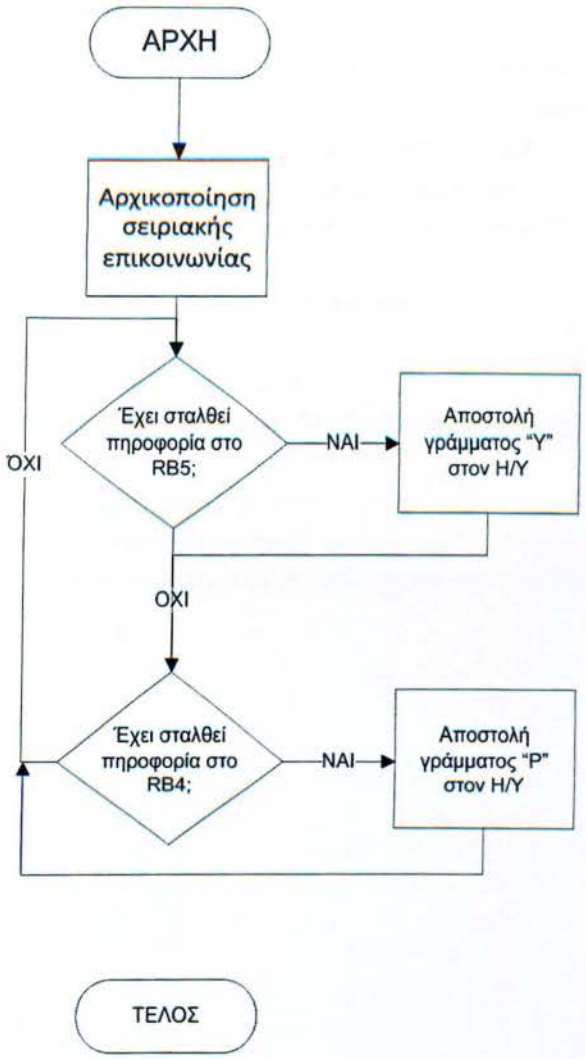
### 4.3 FIRMWARE 16F628A ΓΙΑ ΤΗΝ ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΟΝ ΗΛΕΚΤΡΟΝΙΚΟ ΥΠΟΛΟΓΙΣΤΗ

Ακολουθεί ο κώδικας που "τρέχει" ο μικροελεγκτής για να μπορέσει να επικοινωνήσει με τον υπολογιστή καθώς και το λογικό διάγραμμα.

| | |
|---|---|
| program wireless<br>dim uart_rd as byte | Έναρξη του προγράμματος<br>Αρχικοποίηση μεταβλητής τύπου Byte με όνομα uart_rd |
| main:<br>TRISB = %00011000 | Τίθενται τα bit 5 και bit4 τις πόρτας B ως είσοδοι ενώ τα υπόλοιπα 6 σαν έξοδοι |
| UART1_Init(9600) | Αρχικοποίηση της περιφερειακής μονάδας ασύγχρονης σειριακής επικοινωνίας με ταχύτητα 9600bps |
| Delay_ms(100) | Χρονοκαθυστέρηση 100msec για |

```
                                        σταθεροποίηση της  περιφερειακής
                                        μονάδας  ασύγχρονης σειριακής
                                        επικοινωνίας
  while(TRUE)                           Αιώνια λούπα
      if(RB5_bit = 1) then              Έλεγχος εάν έχει γίνει αποστολή
                                        πληροφορίας στο bit 5 της πόρτας B
        uart_rd="Y"                     Αποθήκευση του ASCII κωδικού του
                                        γράμματος "Y" στην μεταβλητή uart_rd
        UART1_Write(uart_rd)            Αποστολή προς τον υπολογιστή του
                                        προαναφερθέντα κωδικού
        Delay_ms(100)                   Χρονοκαθυστέρηση 100msec για την
      end if                            αποστολή της πληροφορίας
      if (RB4_bit = 1) then             Έλεγχος εάν έχει γίνει αποστολή
                                        πληροφορίας στο bit 4 της πόρτας B
        uart_rd = "P"                   Αποθήκευση του ASCII κωδικού του
                                        γράμματος "P" στην μεταβλητή uart_rd
        UART1_Write(uart_rd)            Αποστολή προς τον υπολογιστή του
                                        προαναφερθέντα κωδικού
        Delay_ms(100)                   Χρονοκαθυστέρηση 100msec για την
      end if                            αποστολή της πληροφορίας
      Delay_ms(200)                     Χρονοκαθυστέρηση 200msec
  wend
end.
```

Ακολουθεί το λογικό διάγραμμα του προγράμματος.




Σχήμα 4.3.a: Λογικό διάγραμμα Firmware περιφερειακού υπολογιστή

## Βιβλιογραφία Κεφαλαίου

<<Α>> "Εισαγωγή στους μικροελεγκτές PICmicro" Σταμάτης Αλατσαθιανός
<<Β>> "Ανάπτυξη Συστημάτων με Μικροελεγκτές 8051" Σταμάτης Αλατσαθιανός
<<Γ>> "Microcontroller Programming" Julio Sanchez, Maria P. Canton
http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf
http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf

## Εικόνες Κεφαλαίου

Σχήμα 4.2.a: Ασύγχρονη σειριακή επικοινωνία
http://www.fiz-ix.com/wp-content/uploads/2013/02/SerialCommunication.png

Σχήμα 4.2.2.a: Max 232 και Max 232 Pin out
http://en.wikipedia.org/wiki/File:Max232.jpg
http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf

Κεφάλαιο 5

## 5.1 Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

Η Python είναι μια γενικής χρήσης, υψηλού επιπέδου γλώσσα προγραμματισμού, της οποίας ο σχεδιασμός και η φιλοσοφία τονίζει την αναγνωσιμότητα στον κώδικα. Η σύνταξη της είναι πολύ σαφής και εκφραστική και συνοδεύεται από ένα μεγάλο και ολοκληρωμένο πρότυπο βιβλιοθήκης. Η Python υποστηρίζει πολλαπλά πρότυπα προγραμματισμού, συμπεριλαμβανομένων του αντικειμενοστραφή (object-oriented) προγραμματισμού. Διαθέτει ένα πλήρως δυναμικό σύστημα τύπου και αυτόματη διαχείριση μνήμης, παρόμοια με των Ruby και Perl. Όπως και άλλες δυναμικές γλώσσες προγραμματισμού, η Python χρησιμοποιείται συχνά ως μια γλώσσα προγραμματισμού, αλλά επίσης και σε ένα ευρύ φάσμα περιβάλλοντος χωρίς κώδικα. Χρησιμοποιώντας εργαλεία τρίτων κατασκευαστών, ο Python κώδικας μπορεί να συσκευαστεί σε αυτόνομα εκτελέσιμα προγράμματα που μπορούν να εκτελεστούν σε σχεδόν οποιοδήποτε λειτουργικό σύστημα. Η πιο ενημερωμένη έκδοση της Python αλλά και ο πηγαίος κώδικας, δυαδικά αρχεία όπως και έγγραφα τεκμηρίωσης, νέα κ.τ.λ. βρίσκονται στην επίσημη ιστοσελίδα της Python. http://www.python.org/

## 5.2 PYGAME

Η pygame είναι μία από τις πολλές βιβλιοθήκες που συνοδεύουν την Python. Είναι σχεδιασμένη να μπορεί να λειτουργήσει σε σχεδόν όλα τα λειτουργικά συστήματα και παρέχει ενότητες για τον σχεδιασμό και την δημιουργία βιντεοπαιχνιδιών. Περιλαμβάνει γραφικά υπολογιστών και βιβλιοθήκες ήχου που έχουν σχεδιαστεί για να χρησιμοποιηθούν με τη γλώσσα Python. Η pygame είναι χτισμένη πάνω από τη απλή βιβλιοθήκη DirectMedia Layer (SDL) , με την πρόθεση να επιτρέπει την ανάπτυξη παιχνιδιών σε πραγματικό χρόνο στον υπολογιστή, χωρίς να χρησιμοποιούνται χαμηλού επιπέδου γλώσσες όπως η C και τα παράγωγά της. Αυτό βασίζεται στη θεωρία ότι οι πιο απαιτητικές λειτουργίες σε ένα παιχνίδι (κυρίως το τμήμα γραφικών) μπορεί να αφαιρεθεί από τη λογική του παιχνιδιού. Αυτό καθιστά δυνατή την χρήση μιας υψηλού επιπέδου γλώσσας προγραμματισμού, όπως η Python, για την δόμηση του παιχνιδιού. Η Pygame γράφτηκε αρχικά από τον Pete Shinners και απελευθερώθηκε στο πλαίσιο της ανοικτής πηγής ελεύθερο λογισμικό GNU (Lesser General Public License).

## 5.3 SOFTWARE ΗΛΕΚΤΡΟΝΙΚΟΥ ΥΠΟΛΟΓΙΣΤΗ

Το πρόγραμμα που θα εκτελεί ο Η/Υ για να μπορέσει να λάβει τα δεδομένα μέσω της σειριακής θύρας, για να εκτελέσει τις διάφορες λειτουργίες και χειρισμό των δεδομένων του είναι γραμμένο στην γλώσσα προγραμματισμού Python. Ο σχεδιασμός των γραφικών αλλά και η μετάδοση τους στον οδηγό της οθόνης του Η/Υ γα την απεικόνισή τους θα γίνει μέσο της Python και της επιπρόσθετης βιβλιοθήκης Pygame. Ακολουθούν κομμάτια του κώδικα με επεξήγηση της χρησιμότητας τους.

```
bif="C:/Users/Billos/Desktop/road.jpg"
mif="C:/Users/Billos/Desktop/jeep2.png"
```

Ορίζονται οι μεταβλητές bif και mif, μεταβλητές στις οποίες αποθηκεύεται το μονοπάτι στο οποίο είναι αποθηκευμένες οι εικόνες που θα χρησιμοποιηθούν. Από το όνομα της κάθε εικόνας μπορεί να διακριθεί ποια αναφέρεται στον δρόμο και ποια στην εικόνα του αμαξιού.

```
import pygame, sys, time, serial
from pygame.locals import *
pygame.init()
```

Κάνουμε εισαγωγή της βιβλιοθήκης pygame, sys, time και serial. Από την βιβλιοθήκη pygame, κάνουμε εισαγωγή όλων των λειτουργιών της, καθώς και μετά την κάνουμε initialize. Την βιβλιοθήκη sys την χρησιμοποιούμε για την αυτόματη έξοδο του προγράμματος κατά τον τερματισμό του αλλά και για την ομαλή έξοδο, σε περίπτωση που το θελήσει ο χρήστης. Η time βιβλιοθήκη είναι απαραίτητη για την διεκπεραίωση της χρονοκαθυστέρησης σε διάφορα σημεία του προγράμματος. Τέλος χρησιμοποιείται και η βιβλιοθήκη serial για να μπορεί να έχει το πρόγραμμα επικοινωνία μέσω της σειριακής θύρας με τον δέκτη.

```
global speed
speed = 2

carspeedx=5
carspeedy=0
parkstage=1
allowmove=True
i=0
ii=-45
back = 10
end=False
```

Η χρήση του χαρακτηριστικού global πριν το όνομα κάποιας μεταβλητής δίνει στην python την πληροφορία ότι στην από κάτω γραμμή θα αναφερθεί η global μεταβλητή με αυτό το όνομα και όχι η εκάστοτε local, ανάλογα στο σημείο που βρίσκεται η εκτέλεση του κώδικα. Στην συγκεκριμένη περίπτωση η μεταβλητή speed δηλώνει την ταχύτητα με την οποία θα οπισθοχωρούν ο δρόμος και τα εκάστοτε εμπόδια, δίνοντας έτσι τη ψευδαίσθηση ότι το αμάξι θα κινείται προς τα μπροστά. Οι μεταβλητές carspeedx και carspeedy αναφέρονται στην ταχύτητα μετακίνησης του αμαξιού στους άξονες x και y αντίστοιχα κατά τη διαδικασία του παρκαρίσματος. Οι μεταβλητές parkstage και allowmove χρησιμοποιούνται αποκλειστικά στην διαδικασία παρκαρίσματος. Η parkstage αναφέρεται στην κίνηση του παρκαρίσματος που βρίσκεται αυτή την στιγμή το αμάξι και η allowmove είναι μια μεταβλητή την οποία αλλάζει ο δέκτης για να πετύχει συγχρονισμό κινήσεων με το πραγματικό και εικονικό αμάξι κατά την διάρκεια παρκαρίσματος. Οι μεταβλητές i και ii χρησιμοποιούνται για την περιστροφή του εικονικού αμαξιού στην οθόνη κατά την διάρκεια των κινήσεων του παρκαρίσματος. Η μεταβλητή back αναφέρεται στην ταχύτητα με την οποία το αμάξι θα οπισθοχωρεί κατά την διαδικασία του παρκαρίσματος. Τέλος η μεταβλητή end δηλώνει την επιτυχημένη ολοκλήρωση του παρκαρίσματος και επιτρέπει στο πρόγραμμα να μπει στην διαδικασία τερματισμού.

```
serial = serial.Serial("COM13", 9600)
```

Η προαναφερθείσα εντολή αναφέρεται στην δημιουργία διαύλου σειριακής επικοινωνίας ανάμεσα στον υπολογιστή και στον δέκτη χρησιμοποιώντας την βιβλιοθήκη serial αναφέροντας ως θύρα επικοινωνίας την com13 και με ταχύτητα στα 9600 baud.

```
class Road:
  def __init__(self):
    self.image=pygame.image.load("C:/Users/Billos/Desktop/road.jpg").convert()
    self.pos = [0,0]

  def blit(self,surface):
    surface.blit(self.image,self.pos)

  def tick(self):
    self.pos[0] += -speed
    if self.pos[0]<-800:
      self.pos[0]=0
```

Σε αυτό το κομμάτι του κώδικα αναφέρεται η κλάση Road ο τρόπος δημιουργίας και οι λειτουργίες που εμπεριέχει. Κατά την δημιουργία ενός αντικειμένου της κλάσης Road εκτελείται η λειτουργία init (initialize) η οποία δίνει στο αντικείμενο μια εικόνα που το αντιπροσωπεύει αλλά και την αρχική του θέση πάνω στον καμβά της οθόνης. Η εικόνα περνάει από την διαδικασία convert της βιβλιοθήκης pygame για τον σωστό χειρισμό της από τον υπολογιστή. Η λειτουργία blit της κλάσης αυτής χρησιμοποιείται για τον σχεδιασμό του εκάστοτε αντικειμένου στον καμβά της οθόνης παίρνοντας σαν όρισμα το αντικείμενο (μαζί με την εικόνα και την τοποθεσία του) αλλά και τον καμβά απεικόνισης στην οθόνη. Η λειτουργία tick χρησιμοποιείται για την νοητή μετακίνηση των συντεταγμένων της τοποθεσίας του αντικειμένου, με σκοπό την ένδειξη κίνησης του αντικειμένου. Η ταχύτητα μετακίνησης αναφέρεται από την μεταβλητή speed . Ένα σημαντικό στοιχείο που πρέπει να αναφέρουμε είναι ότι εάν η μετακίνηση ξεπεράσει τα -800pixel οι συντεταγμένες μηδενίζονται, αυτό γίνεται για την μείωση των διαστάσεων της εικόνας που αντιπροσωπεύει τον δρόμο.

```
class Car:
  def __init__(self):
    self.image=pygame.image.load("C:/Users/Billos/Desktop/jeep2.png").convert_alpha()

self.imagerot=pygame.image.load("C:/Users/Billos/Desktop/jeep2.png").convert_alpha()
```

```
    self.pos = [350,100]

  def blit(self,surface):
    surface.blit(self.image,self.pos)
```

Το παραπάνω κομμάτι αναφέρεται στην κλάση Car η οποία εμπεριέχει τις λειτουργίες init και blit οι οποίες λειτουργούν με τον ίδιο τρόπο όπως της κλάσης Road. Σημαντικό είναι να αναφερθεί ότι τα αντικείμενα τύπου Car θα έχουν δύο εικόνες που θα τα αντιπροσωπεύουν. Η πρώτη θα χρησιμοποιείται για την απεικόνιση του αυτοκινήτου σε κατάσταση κίνησης στον άξονα x , ενώ η δεύτερη θα χρησιμοποιηθεί εάν στην κίνηση εμπεριέχεται και περιστροφή του αυτοκινήτου(κατά τη διάρκεια παρκαρίσματος). Τέλος στην περίπτωση του αμαξιού, επειδή η εικόνα του εμπεριέχει σημεία διαφάνειας, θα χρησιμοποιηθεί η λειτουργία convert_alpha() για τον σωστό χειρισμό της από την Python.

```
  def move(self):

    if allowmove:
      self.pos[0] += carspeedx
      self.pos[1] += carspeedy
      time.sleep(0.1)

    if parkstage == 1:
      if i<18 :
        global i
        i+=2
      else:
        global parkstage
        parkstage = 2
        global speed
        speed = 0
        global carspeedx
        carspeedx = -5
        global carspeedy
        carspeedy = 0
        global i
        i=0



    if parkstage == 2:
      if i<20 :
        global i
        i+=2

      else:
        global parkstage
        parkstage = 25
        global speed
```

```
        speed = 0
        global carspeedx
        carspeedx = 1
        global carspeedy
        carspeedy = 2
        global i
        i=0
        time.sleep(0.5)
        global allowmove
        allowmove=False
        time.sleep(0.5)



if parkstage == 3:
    if i<23 :
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i+=2

    else:
        global parkstage
        parkstage = 35
        global carspeedx
        carspeedx = -8
        global carspeedy
        carspeedy = -4
        global allowmove
        allowmove=False
        time.sleep(0.5)


if parkstage == 4:
    if i<46:
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i+=2

    else:
        global parkstage
        parkstage = 45
        global carspeedx
        carspeedx = -8
        global carspeedy
        carspeedy = -4
        global allowmove
        allowmove=False
```

```
if parkstage == 5:
    if i>23:
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i-=2

    else:
        global parkstage
        parkstage = 55
        global carspeedx
        carspeedx = 3
        global carspeedy
        carspeedy = 0

        global allowmove
        allowmove=False


if parkstage == 6:
    if i>0:
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i-=2

    else:
        global end
        end=True
```

Εδώ αναφέρεται μια από της λειτουργίες της κλάσης car, η λειτουργία move. Η λειτουργία move είναι υπεύθυνη για την μετακίνηση του εικονικού αυτοκινήτου μέσα στον καμβά της οθόνης . Κάθε φορά που εκτελείτε και εφόσον ο δέκτης του έχει επιτρέψει την κίνηση μέσω της μεταβλητής allowmove ανανεώνει της συντεταγμένες στον άξονα x και y του αντικειμένου. Τέλος υπάρχει μια πολύ μικρή χρονοκαθυστέριση της τάξης του 1msec για την πιο αληθοφανή κίνηση του αντικειμένου.

Μέσα στην λειτουργία move της κλάσης Car υπάρχουν και πολλά διαφορετικά κομμάτια κώδικα(διαφοροποιημένα με την εντολή ελέγχου if ) για την κίνηση του αυτοκινήτου κατά την διάρκεια του παρκαρίσματος. Αυτό που κάνουν είναι να αλλάζουν την ταχύτητα με την οποία ο αυτοκίνητο θα μετακινείται πάνω στον καμβά. Στο τέλος κάθε κομματιού του κώδικα υπάρχει μία εντολή else για με την οποία θα προετοιμαστεί η λειτουργία move για να μην έχει επιρροή στην μετατόπιση του αυτοκινήτου μέχρι ο δέκτης να επιτρέψει την επόμενη κίνηση του αυτοκινήτου.

Κατά την διάρκεια της κίνησης 3, 4, 5 και 6 υπάρχει και η εντολή self.image = pygame.transform.rotozoom(self.imagerot, x, 1) η οποία περιστρέφει την εικόνα του αυτοκινήτου. Κάθε φορά που εκτελείται περιστρέφει την εικόνα imagerot κατά x μοίρες είτε δεξιόστροφα είτε αριστερόστροφα . Η περιστρεμμένη εικόνα αποθηκεύεται στο image του αντικειμένου. Είναι σημαντικό η εικόνα imagerot να μην τροποποιηθεί καθόλου για να μην αλλοιωθεί η ποιότητα της εικόνας μετά πολλές αποθηκεύσεις και πάντα η python να έχει ένα αντίγραφο της εικόνας του αμαξιού με την καλύτερη δυνατή ποιότητα. Αυτό

επιτυγχάνεται με την περιστροφή της εικόνας αρχικά λίγων μοιρών και κατά την πάροδο του χρόνου αυξάνοντας τον αριθμό των μοιρών, δίνοντας την ψευδαίσθηση ότι η εικόνα κάνει μια γραμμική περιστροφή.

Όταν το αμάξι κάνει όλες τις απαραίτητες κινήσεις η μεταβλητή end θα γίνει αληθείς επιτρέποντας στο πρόγραμμα την έξοδο.

```
class Obstacle:
  def __init__ (self,pos):
    self.pos = [pos[0],pos[1]]
    self.rect = pygame.Rect(self.pos,(2,50))
    self.image = pygame.image.load("C:/Users/Billos/Desktop/obstacle.jpg").convert()

  def blit(self,surface):
    surface.blit(self.image,self.pos)

  def tick(self):
    self.pos[0] += -speed
    self.rect.move_ip((-speed,0))

  def get_rect(self):
    return self.rect
```

Η κλάση Obstacle περιλαμβάνει όλες τις λειτουργίες οι οποίες αναφέρονται στον σχεδιασμό, απεικόνιση και δημιουργία εμποδίων πάνω στον καμβά. Ο τρόπος με τον οποίο λειτουργεί είναι αφαιρετικός. Το πρόγραμμα θεωρεί ότι παντού υπάρχουν εμπόδια αλλά με την δημιουργία ενός τέτοιου αντικειμένου στον εικονικό αυτό χώρο το εμπόδιο εξαφανίζεται. Κάθε αντικείμενο κατά την δημιουργία του από την λειτουργία init του αναθέτονται συντεταγμένες, εικόνα(κομμάτι δρόμου) που θα προβάλλεται στην οθόνη αλλά και ένα αόρατο τετράγωνο το οποίο θα χρησιμοποιηθεί αργότερα για την διαγραφή του αντικειμένου όταν αυτό ξεφύγει από τα ορατά όρια της οθόνης, ελευθερώνοντας μνήμη RAM. Άλλες λειτουργίες της κλάσης αυτής είναι η blit, που όπως και προηγουμένως, είναι υπεύθυνη για την απεικόνιση του αντικειμένου στην οθόνη. Η λειτουργία tick μετακινεί το εμπόδιο πάνω στον καμβά αλλά και η λειτουργία get_rect που επιστέφει σε όποιον την καλέσει το αόρατο τετράγωνο για τον έλεγχο των συντεταγμένων του εμποδίου πάνω στον καμβά.

```
def __main__():
```

```
pygame.display.set_mode((800,480),0,32)
surface = pygame.display.get_surface()
```

Η λειτουργία main είναι αυτή που περιλαμβάνει όλες τις εντολές για των συγχρονισμό όλων των λειτουργιών του προγράμματος και όλα τα υπόλοιπα κομμάτια του προγράμματος. Αρχικά γίνεται η δημιουργία του καμβά απεικόνισης του προγράμματος με όνομα surface. Η ανάλυση που θα χρησιμοποιηθεί είναι 800x480 με χρωματικό βάθος 32bit.

```
park = False
road = Road()
car = Car()
obstacle = []
clock = pygame.time.Clock()
clock.tick()
counter=10
carspeed = 1
global back
back=10
```

Αρχικοποιούμε μια μεταβλητή με όνομα park και της τοποθετούμε την τιμή False. Η μεταβλητή αυτή θα είναι ο δείκτης του προγράμματος για το αν το αμάξι βρίσκετε σε κατάσταση παρκαρίσματος ή εύρεση θέσης παρκαρίσματος.

Ακολουθεί η δημιουργία ενός αντικειμένου με όνομα road τύπου Road αλλά και ονόματος car τύπου Car. Δημιουργούμε μια λίστα obstacle στην οποία θα αποθηκεύονται όλα τα αντικείμενα τύπου Obstacle.

Για να μπορέσουμε να οριοθετήσουμε την ταχύτητα εκτέλεσης του προγράμματος σε έναν πολύ γρήγορο ηλεκτρονικό υπολογιστή δημιουργούμε ένα εσωτερικό ρολόι αναφοράς με όνομα clock. Αρχικοποιούμε την ταχύτητα του αμαξιού προς τα μπροστά και όπισθεν με τις μεταβλητές carspeed και back αντίστοιχα. Η μεταβλητή counter θα ελέγχει πόσο σύντομα θα γίνεται έλεγχος στον δέκτη για νέα δεδομένα (εμποδίου ή παρκαρίσματος). Λόγο της αργής ανταπόκρισης της σειριακής επικοινωνίας θέλουμε να γίνεται ο έλεγχος όσο πιο σπάνια γίνεται χωρίς όμως να επηρεάζεται η εγκυρότητα των δεδομένων.

```
while True:

    clock.tick(120)
```

Μια while με όρισμα True θα κάνει το πρόγραμμα να εκτελείται επ' αόριστων μέχρι ο χρήστης ή το ίδιο το πρόγραμμα να αποφασίσει να τερματιστεί.

Η εντολή clock.tick(120) οριοθετεί το πρόγραμμα να εκτελέσει το πολύ 120 φορές σε ένα δευτερόλεπτο άσχετα με την ταχύτητα την οποία προσφέρει μηχανικά ο υπολογιστής. Η έλλειψη αυτής της εντολής είναι ο λόγος που πολλά παιχνίδια και προγράμματα από την δεκαετία του '80 και '90 δεν είναι σε θέση να εκτελεστούν από του σήμερα ισχυρότερους υπολογιστές.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

Η χρήση των εντολών αυτών δίνει την δυνατότητα στον χρήστη να τερματίζει με ασφάλεια το πρόγραμμα πατώντας το γραφικό κουμπί X στην πάνω δεξιά γωνία του παραθύρου. Η διαδικασία που εκτελεί ο κώδικας είναι ο έλεγχος των δραστηριοτήτων του αφορούν το προγράμματος(π.χ. πάτημα ενός κουμπιού στο πληκτρολόγιο) και εάν κάποια τέτοια δραστηριότητα υπάρχει και είναι το πάτημα του κουμπιού <QUIT> να τερματίσει με ασφαλή τρόπο το πρόγραμμα.

```
if counter == 0:
    serialdata=serial.read()
    if (serialdata == "Y") and (not (park)):
        obstacle.append(Obstacle([430,50]))
    if (serialdata == "P"):
        park = True

    if park:
        if ((serialdata == "Y") and (parkstage == 25)):
            global parkstage
            parkstage = 3
            global allowmove
            allowmove=True
        if ((serialdata == "P") and (parkstage == 35)):
            global parkstage
            parkstage = 4
            global allowmove
            allowmove=True
        if ((serialdata == "Y") and (parkstage == 45)):
            global parkstage
            parkstage = 5
            global allowmove
            allowmove=True
        if ((serialdata == "P") and (parkstage == 55)):
            global parkstage
            parkstage = 6
            global allowmove
            allowmove=True
    counter=10
```

```
counter-=1
```

Σε αυτό το κομμάτι αναφέρεται η λήψη και ο χειρισμός των δεδομένων που λαμβάνει το πρόγραμμα από τον δέκτη. Εάν η μεταβλητή counter είναι 0 σημάνει ότι έχει περάσει αρκετός χρόνος από την τελευταία φορά που το πρόγραμμα έκανε έλεγχο για νέα δεδομένα. Ελέγχει για δεδομένα και εφόσον υπάρχουν τα αποθηκεύει στην μεταβλητή serialdata. Ύστερα γίνεται ο έλεγχος τον δεδομένων. Αν ο δέκτης έχει στείλει το γράμμα "Y" που σηματοδοτεί κενή θέση παρκινγκ και δεν βρίσκεται ηδη σε κατάσταση παρκαρίσματος γίνεται η δημιουργία ενός αντικειμένου τύπου Obstacle και τοποθετείτε στην υπάρχουσα λίστα με όλα τα υπόλοιπα αντικείμενα. Εάν στην serialdata βρίσκεται το γράμμα "P" ενεργοποιείται η κατάσταση παρκαρίσματος αλλάζοντας την τιμή της μεταβλητής park σε True.

Το υπόλοιπο κομμάτι του κώδικα αυτού χρησιμεύει στον συγχρονισμό των κινήσεων του πραγματικού και του εικονικού αμαξιού. Τα γράμματα "Y" και "P" θα αποστέλλονται εναλλάξ από τον δέκτη στο πρόγραμμα για την αποφύγει πιθανών λαθών στον συγχρονισμό. Κάθε φορά που θα λαμβάνει το πρόγραμμα ένα από αυτά τα δύο γράμματα και το αμάξι βρίσκεται ηδη δε κατάσταση παρκαρίσματος θα επιτρέπει στο εικονικό αμάξι να κάνει μια κίνηση ακόμα. Τα γράμματα αυτά στέλνονται στον δέκτη από το πραγματικό αμάξι κάθε φορά που ολοκληρώνει μία από τις προκαθορισμένες κινήσεις του παρκαρίσματος. Τέλος, η μεταβλητή counter επανέρχεται στην τιμή 10(δέκα) για τον και μετέπειτα σωστό έλεγχο πιθανών δεδομένων από τον δέκτη.

```
for j in obstacle:
    j.tick()
    if j.get_rect().top < 0:
        obstacle.remove(j)
road.tick()
```

Για κάθε ένα αντικείμενο τύπου Obstacle στην λίστα obstacle γίνεται η κλίση τις λειτουργίας get_rect() για να επιστραφεί το αόρατο τετράγωνο που περικλείει κάθε αντικείμενο. Οι πάνω συντεταγμένες του τετραγώνου αυτού ελέγχονται για το κατά πόσον βρίσκονται ακόμα σε ορατό κομμάτι του καμβά της οθόνης. Αν δεν βρίσκονται σε ορατό κομμάτι του καμβά της οθόνης, το αντικείμενο σβήνεται από την μνήμη του υπολογιστή. Τέλος εκτελείτε η λειτουργία road.tick(), που είναι υπεύθυνη για την κίνηση του δρόμου πάνω στον καμβά της οθόνης.

```
if park:
    global speed
    speed = 0

    car.move()
```

Πραγματοποιείται έλεγχος της τιμής της μεταβλητής park. Εάν είναι True, μεταβλητή speed μηδενίζεται κάνοντας τον δρόμο ακίνητο και ενεργοποιείται η κίνηση του αμαξιού.

```
    if end:
        font = pygame.font.SysFont("Free Sans",32)
        level_image = font.render("Parking OK",True,(255,255,0))
        surface.blit(level_image,(240,180))
        pygame.display.flip()
        time.sleep(6)
        pygame.quit()
        sys.exit()
```

Ο κώδικας αυτός θα εκτελεστεί στο τέλος του παρκαρίσματος που η μεταβλητή end θα έχει τιμή True και έχει σαν εύθηνη τον σχεδιασμό των λέξεων " Parking OK" στην οθόνη και τη εμφάνισή τους με την εντολή της Pygame pygame.display.flip() . Μετά την εμφάνιση των λέξεων αυτών υπάρχει μια χρονοκαθυστέρηση των 10 δευτερολέπτων πριν την έξοδο του προγράμματος.

```
    surface.fill((0,0,0))
    road.blit(surface)

    for j in obstacle:
        j.blit(surface)
    car.blit(surface)
    pygame.display.flip()
```

Σε αυτό το κομμάτι του προγράμματος γίνεται ο σχεδιασμός του καμβά της οθόνης. Αρχικά μια στρώση μαύρου χρώματος σχεδιάζεται. Ύστερα ο δρόμος και στην συνέχεια ένα-ένα όλα τα κομμάτια που δεν έχουν εμπόδιο. Τέλος γίνεται ο σχεδιασμός του αμαξιού. Η τελευταία εντολή είναι αυτή που ανανεώνει την οθόνη με τον καινούριο καμβά για να μπορεί ο χρήστης να τον δει.

```
__main__()
```

Τέλος, η εντολή __main__() θέτει σε λειτουργία όλο το πρόγραμμα.

Ακολουθεί όλο το πρόγραμμα γραμμένο την σύνταξη που χρειάζεται η Pyton.

```
bif="C:/Users/Billos/Desktop/road.jpg"
mif="C:/Users/Billos/Desktop/jeep2.png"



import pygame, sys, time, serial
from pygame.locals import *
pygame.init()

global speed
speed = 2
```

```
carspeedx=5
carspeedy=0
parkstage=1
allowmove=True
i=0
ii=-45
end=False

back = 10
#
#
serial = serial.Serial("COM13", 9600)
serial.write(2)
#
#


class Road:
  def __init__(self):
    self.image=pygame.image.load("C:/Users/Billos/Desktop/road.jpg").convert()
    self.pos = [0,0]

  def blit(self,surface):
    surface.blit(self.image,self.pos)

  def tick(self):
    self.pos[0] += -speed
    if self.pos[0]<-800:
      self.pos[0]=0


class Car:
  def __init__(self):
    self.image=pygame.image.load("C:/Users/Billos/Desktop/jeep2.png").convert_alpha()
    self.imagerot=pygame.image.load("C:/Users/Billos/Desktop/jeep2.png").convert_alpha()
    self.pos = [350,100]

  def blit(self,surface):
    surface.blit(self.image,self.pos)

  def move(self):

    if allowmove:
      self.pos[0] += carspeedx
      self.pos[1] += carspeedy
      time.sleep(0.1)

    if parkstage == 1:
      if i<18 :
        global i
        i+=2
      else:
        global parkstage
        parkstage = 2
        global speed
        speed = 0
        global carspeedx
        carspeedx = -5
        global carspeedy
        carspeedy = 0
        global i
        i=0
```

```
if parkstage == 2:
    if i<20 :
        global i
        i+=2

    else:
        global parkstage
        parkstage = 25
        global speed
        speed = 0
        global carspeedx
        carspeedx = 1
        global carspeedy
        carspeedy = 2
        global i
        i=0
        time.sleep(0.5)
        global allowmove
        allowmove=False
        time.sleep(0.5)



if parkstage == 3:
    if i<23 :
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i+=2

    else:
        global parkstage
        parkstage = 35
        global carspeedx
        carspeedx = -8
        global carspeedy
        carspeedy = -4
        global allowmove
        allowmove=False
        time.sleep(0.5)


if parkstage == 4:
    if i<46:
        self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
        global i
        i+=2

    else:
        global parkstage
        parkstage = 45
        global carspeedx
        carspeedx = -8
        global carspeedy
        carspeedy = -4
        global allowmove
        allowmove=False
```

```
    if parkstage == 5:
        if i>23:
            self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
            global i
            i-=2

        else:
            global parkstage
            parkstage = 55
            global carspeedx
            carspeedx = 3
            global carspeedy
            carspeedy = 0

            global allowmove
            allowmove=False



    if parkstage == 6:
        if i>0:
            self.image = pygame.transform.rotozoom(self.imagerot, -i, 1)
            global i
            i-=2

        else:
            global end
            end=True


    #if self.pos[1]<30:#==
     #  global carspeedy
      # carspeedy = 0
       # if ii<0:
        #   self.image = pygame.transform.rotozoom(self.imagerot, ii, 1)
        #   global ii
         #  ii+=1



        # if self.pos[0]<200:#==
        #   global end
         #  end=True



class Obstacle:
    def __init__ (self,pos):
        self.pos = [pos[0],pos[1]]
        self.rect = pygame.Rect(self.pos,(2,50))
        self.image = pygame.image.load("C:/Users/Billos/Desktop/obstacle.jpg").convert()

    def blit(self,surface):
        surface.blit(self.image,self.pos)

    def tick(self):
        self.pos[0] += -speed
        self.rect.move_ip((-speed,0))

    def get_rect(self):
```

```
    return self.rect

def __end__(surface):
  font = pygame.font.SysFont("Free Sans",32)
  level_image = font.render("Parking OK",True,(255,255,0))
  surface.blit(level_image,(240,180))
  pygame.display.flip()
  time.sleep(10)


def __main__():
  pygame.display.set_mode((800,480),0,32)
  surface = pygame.display.get_surface()


  road = Road()
  car = Car()
  obstacle = []
  park = False
  add = False
  clock = pygame.time.Clock()
  clock.tick()
  counter=60
  carspeed = 1
  global back
  back=10




  while True:

    clock.tick(120)

    for event in pygame.event.get():
      if event.type == QUIT:
        pygame.quit()
        sys.exit()


      if event.type == KEYUP:
        if event.key==K_UP:
          park = True
        if event.key==K_SPACE:
          add=False
      if event.type == KEYDOWN:
        if event.key==K_SPACE:
          add = True



    if (add == True):
        obstacle.append(Obstacle([430,50]))




    if counter == 0:
      serialdata=serial.read()
```

```
    if (serialdata == "Y") and (not (park)):
        obstacle.append(Obstacle([430,50]))
    if (serialdata == "P"):
        park = True

    if park:
        if ((serialdata == "Y") and (parkstage == 25)):
            global parkstage
            parkstage = 3
            global allowmove
            allowmove=True
        if ((serialdata == "P") and (parkstage == 35)):
            global parkstage
            parkstage = 4
            global allowmove
            allowmove=True
        if ((serialdata == "Y") and (parkstage == 45)):
            global parkstage
            parkstage = 5
            global allowmove
            allowmove=True
        if ((serialdata == "P") and (parkstage == 55)):
            global parkstage
            parkstage = 6
            global allowmove
            allowmove=True
    counter=10

counter-=1




for j in obstacle:
    j.tick()
    if j.get_rect().top < 0:
        obstacle.remove(j)
road.tick()




if park:
    global speed
    speed = 0

    car.move()

if end:
    font = pygame.font.SysFont("Free Sans",32)
    level_image = font.render("Parking OK",True,(255,255,0))
    surface.blit(level_image,(240,180))
    pygame.display.flip()
    time.sleep(6)
    pygame.quit()
    sys.exit()


surface.fill((0,0,0))
```

```
    road.blit(surface)

    for j in obstacle:
        j.blit(surface)
    car.blit(surface)
    pygame.display.flip()



__main__()
```

Βιβλιογραφία Κεφαλαίου

http://www.python.org/doc/
http://wiki.python.org/moin/BeginnersGuide
http://wiki.python.org/moin/
http://www.pygame.org/docs/
http://www.pygame.org/wiki/tutorials

**Επίλογος**

Η διεκπεραίωση του παράλληλου παρκαρίσματος ενός αυτοκινήτου, το οποίο είναι ικανό να αντιλαμβάνεται εάν ο χώρος είναι κατάλληλος για στάθμευση όσο αφορά το μήκος του, είναι μια σύνθετη εργασία, για την οποία χρειάζεται το αυτοκίνητο που είναι κατάλληλα διαμορφωμένο από τον χειριστή. Εκτός της δημιουργίας του αυτοκινήτου σημασία πρέπει να δοθεί και στον προγραμματισμό, ο οποίος ουσιαστικά είναι υπεύθυνος για τις αυτόματες κινήσεις που απαιτούνται για το παρκάρισμα. Τέλος η άρτια γραφική απεικόνιση επιτρέπει στον χειριστή και στο κοινό, να αντιλαμβάνεται σε αληθοφανές περιβάλλον τις κινήσεις και τον προγραμματισμό αυτών για την σωστή επίτευξη της στάθμευσης ενός αυτοκινήτου. Η εργασία αυτή σε περίπτωση που βελτιωθεί και διαμορφωθεί κατάλληλα θα μπορούσε να χρησιμοποιηθεί από αυτοκινητοβιομηχανίες, έτσι ώστε οι χειριστές αυτοκινήτων να μπορούν ευκολότερα να βρίσκουν κατάλληλες θέσεις και να σταθμεύουν ευκολότερα τα οχήματά τους.

Συντελικά πρέπει να σημειωθεί ότι, με την χρήση αυτού το κυκλώματος και με τον κατάλληλο προγραμματισμό, κάποιος ο οποίος είναι γνώστης μπορεί να δημιουργήσει και να προγραμματίσει τέτοιου είδους κυκλώματα και να δημιουργήσει πατέντες φθηνού κόστους που θα μπορούν να εκπληρώσουν πληθώρα αναγκών του καταναλωτικού κοινού. Η δημιουργία αυτής της εργασίας στόχο έχει την απαρχή παρόμοιων κυκλωμάτων, τα οποία αναβαθμίζουν την τεχνολογία και είναι ευκολονόητη η λειτουργία τους όχι μόνο από γνώστες προγραμματισμού και κυκλωμάτων, αλλά και από ανθρώπους οι οποίοι δεν είναι επαγγελματίες του χώρου.