ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

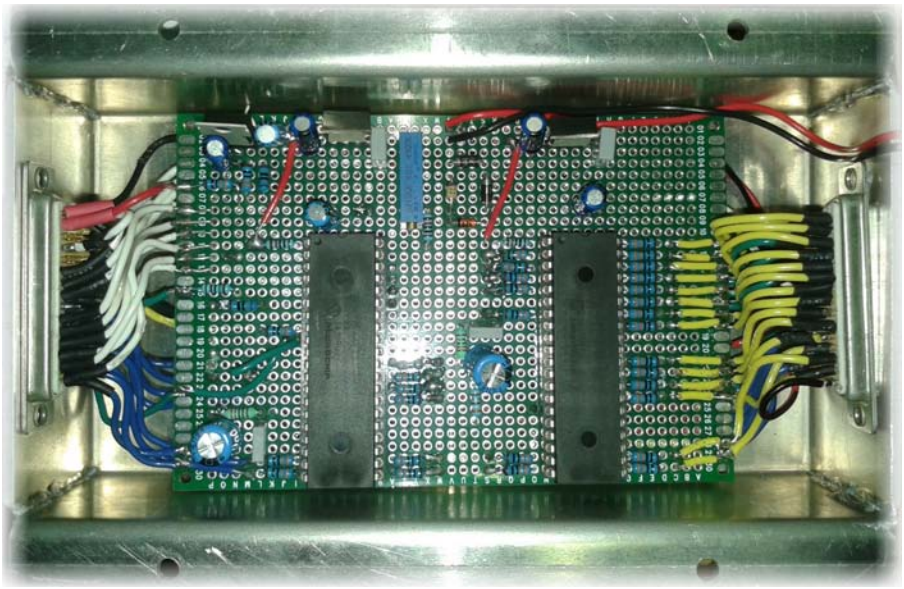
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ:

"Ανάπτυξη και υλοποίηση ηλεκτρονικού και ηλεκτρικού συστήματος αισθητήρων και διεπαφών για έλεγχο κινητήρα υψηλών επιδόσεων"

¨Development and implementation of electronic and electrical system of sensors and interfaces for high performance engine control¨

ΠΑΠΑΣΤΑΜΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ



Επιβλέπων Καθηγητής

Σταμούλης Γεώργιος

Συνεπιβλέπων καθηγητής

Πλέσσας Φώτιος

Βόλος 2014

Στην οικογένεια και στους φίλους μου

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας κ. Γεώργιο Σταμούλη και κ. Πλέσσα Φώτιο για την εμπιστοσύνη που επέδειξαν στο πρόσωπό μου, την άριστη συνεργασία, την συνεχή καθοδήγηση και τις ουσιώδεις υποδείξεις και παρεμβάσεις, τις οξυδερκείς συμβουλές και παρατηρήσεις, την σημαντική συμβολή, τις καίριες παρεμβάσεις, τα κατάλληλα και ουσιώδη κάθε φορά σχόλια, και γενικότερα για την εμψύχωση και υποστήριξή τους που διευκόλυναν την εκπόνηση της πτυχιακής εργασίας.

Θα ήθελα να εκφράσω την εκτίμησή μου σε όλα τα παιδιά με τα οποία συνεργάστηκα όλα αυτά τα χρόνια των προπτυχιακών σπουδών μου. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον προπτυχιακό φοιτητή Δατσογιάννη Δημήτριο για την άψογη συνεργασία που είχαμε στην παρούσα διπλωματική και τον μεταπτυχιακό φοιτητή Ξιφαρά Ιωάννη για τις συμβουλές που μου παρείχε στην διπλωματική μου εργασία.

Επίσης, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου, στην κοπέλα μου και στους φίλους μου για την αμέριστη υποστήριξη και την ανεκτίμητη βοήθεια που μου παρείχαν τόσο κατά την διάρκεια των σπουδών μου όσο και κατά την εκπόνηση της διπλωματικής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω ακόμα πιο πολύ τα μέλη της Centaurus Racing Team του Πανεπιστημίου Θεσσαλίας στην οποία αποτέλεσα μέλος και μας εμπιστεύτηκαν ώστε η διπλωματική μας εργασία να είναι κομμάτι του μονοθεσίου που κατασκευάσαμε. Η άψογη συνεργασία μου μαζί τους οδήγησε στην δημιουργία ενός πολύ ευχάριστου κλίματος με αποτέλεσμα η εκπόνηση της πτυχιακής μου εργασίας να αποτελέσει για μένα μία καταπληκτική διαδικασία και εμπειρία που θα μου μείνει αξέχαστη!

Παπασταμόπουλος Γεώργιος

Βόλος, 2014

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή 8

ΚΕΦΑΛΑΙΟ 2

Επιλογή μικροελεγκτή 9

ΚΕΦΑΛΑΙΟ 3

Θερμοκρασία κινητήρα 11

ΚΕΦΑΛΑΙΟ 4

Στροφές κινητήρα 19

ΚΕΦΑΛΑΙΟ 5

Ταχύτητα μονοθεσίου 26

ΚΕΦΑΛΑΙΟ 6

Υπολογισμός σχέσης κιβωτίου ταχυτήτων 29

ΚΕΦΑΛΑΙΟ 7

Στάθμη καυσίμου 40

ΚΕΦΑΛΑΙΟ 8

Προσομοίωση κυκλωμάτων και κώδικα 43

ΚΕΦΑΛΑΙΟ 9

Κατασκευή της πλακέτας 45

ΚΕΦΑΛΑΙΟ 10

Προοπτικές βελτίωσης του συστήματος 49

ΚΕΦΑΛΑΙΟ 11

Επίλογος 50

ΒΙΒΛΙΟΓΡΑΦΙΑ 51

1. Εισαγωγή

Το περιεχόμενο της παρούσας διπλωματικής εργασίας κινείται γύρω από το μονοθέσιο τύπου φόρμουλα της Centaurus Racing Team του Πανεπιστημίου Θεσσαλίας και συγκεκριμένα ένα ηλεκτρονικό και ηλεκτρικό σύστημα, που αποτελεί κομμάτι του. Η συμμετοχή στην προαναφερθείσα ομάδα αποτέλεσε την αφορμή για την κατασκευή του συστήματος με το οποίο ο οδηγός άλλα και οι μηχανικοί του μονοθεσίου θα μπορούν να ενημερώνονται για κάποιες σημαντικές μεταβλητές του οχήματος.

Το σύστημα αναπτύχθηκε γύρω από μικροελεγκτές PIC της Microchip και για την διεπαφή του οδηγού με το μονοθέσιο γίνεται χρήση μιας οθόνης LCD και μιας αλφαριθμητικής LED, καθώς και διαφόρων άλλων φωτεινών ενδείξεων. Στην LCD οθόνη εμφανίζονται οι στροφές του κινητήρα, η θερμοκρασία του καθώς και η ταχύτητα με την οποία κινείται το μονοθέσιο. Στην LED οθόνη εμφανίζεται η σχέση που είναι επιλεγμένη στο κιβώτιο ταχυτήτων. Επίσης, υπάρχει μια γραμμή από led για την στάθμη του καυσίμου καθώς και άλλη μια για τις στροφές του κινητήρα.

Το σύστημα αυτό είναι αυτοσχέδιο, χειροποίητο και δοκιμασμένο στο μονοθέσιο “NESSUS” και παρουσιάστηκε επιτυχώς στον παγκόσμιο διαγωνισμό Formula Student που συμμετείχε η Centaurus Racing Team τον Αύγουστο του 2014 στην Ουγγαρία.

2. Επιλογή μικροελεγκτή

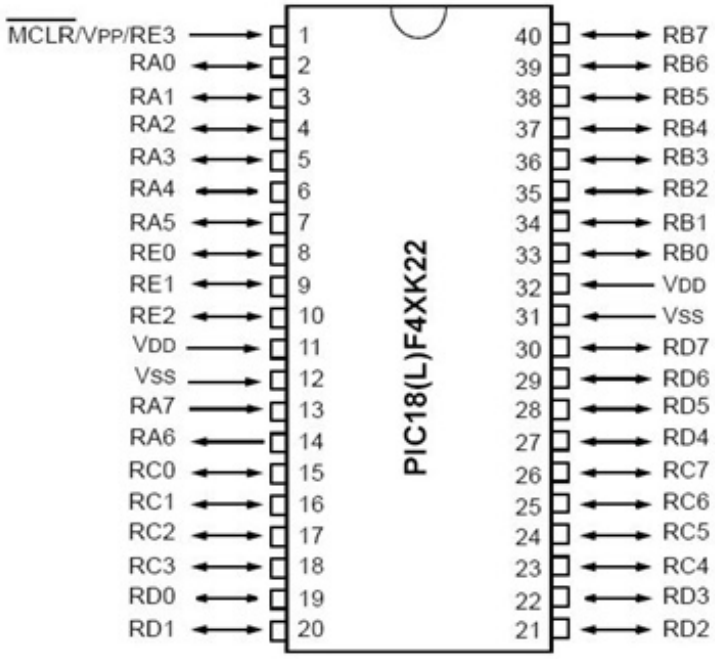
Επειδή στόχος του όλου εγχειρήματος ήταν να παραμείνει το κόστος χαμηλό, το value for money ήταν ένας σημαντικός παράγοντας στην επιλογή των υλικών.

Ο μικροελεγκτής που επιλέχθηκε είναι ο PIC18F46K22 της Microchip. Ο μικροελεγκτής αυτός είναι τεχνολογίας 8-bit RISC με πολύ χαμηλό κόστος και πολλές δυνατότητες, οι οποίες παρουσιάζονται στον παρακάτω πίνακα.

Parameter Name	**Value**
Program Memory Type	Flash
Program Memory (KB)	64
CPU Speed (MIPS)	16
RAM Bytes	3,896
Data EEPROM (bytes)	1024
Digital Communication Peripherals	2-UART, 2-A/E/USART, 2-SPI, 2-I2C2-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	2 CCP, 3 ECCP
Timers	3 x 8-bit, 4 x 16-bit
ADC	28 ch, 10-bit
Comparators	2
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	40
XLP	Yes
Cap Touch Channels	28

Το συγκεκριμένο μοντέλο μικροελεγκτή έχει δυνατότητα λειτουργίας μέχρι 64 MHz με εσωτερικό ρολόι, μέχρι 28 αναλογικές εισόδους, 3896 Bytes RAM και 1024 Bytes EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory). Επίσης, ανήκει στην οικογένεια μικροελεγκτών χαμηλής κατανάλωσης ισχύος όπως επίσης και στην οικογένεια μικροελεγκτών οι οποίοι είναι κατάλληλοι για υψηλές θερμοκρασίες.

Στον παρακάτω πίνακα φαίνεται το διάγραμμα των pin των μικροελεγκτών που χρησιμοποιήθηκαν.



Κάθε pin του μικροελεγκτή μπορεί να έχει μια ή περισσότερες λειτουργίες. Ανάλογα με την δήλωση που θα γίνει μέσα στον κώδικα ένα pin μπορεί να είναι είσοδος ή έξοδος, αναλογικό ή ψηφιακό κλπ. Στη συνέχεια της εργασίας, κάθε pin που χρησιμοποιήθηκε θα αναφέρεται και ο τρόπος που ορίζεται η λειτουργία του.

Ο κώδικας του μικροελεγκτή γράφτηκε σε γλώσσα C και μεταγλωττίστηκε με τον compiler της Mikroelektronika Mikro C Pro for Pic.

3. Θερμοκρασία κινητήρα

Τόσο στους απλούς κινητήρες όσο και σε κινητήρες υψηλών επιδόσεων είναι ζωτικής σημασίας η γνώση της θερμοκρασίας του κινητήρα για την βέλτιστη απόδοση του και σωστή λειτουργία του, καθώς και η παρακολούθησή της για την αποφυγή επικίνδυνων συνθηκών.

Ο κινητήρας είναι εξοπλισμένος με έναν αισθητήρα θερμοκρασίας τεχνολογίας θέρμιστορ NTC (Negative Temperature Coefficient), ο οποίος συνδέεται με τον εγκέφαλο του μονοθεσίου, ώστε να ρυθμίζει την λειτουργία του κινητήρα ανάλογα με την θερμοκρασία του. Το θέρμιστορ είναι μια αντίσταση της οποίας η τιμή μεταβάλλεται σημαντικά σε σχέση με την θερμοκρασία, περισσότερο απ' ότι στις κανονικές αντιστάσεις. Στην παρακάτω εικόνα φαίνονται τα φυσικά χαρακτηριστικά του αισθητήρα.

Ο αισθητήρας έχει αντίσταση περίπου 2800Ω στους 25 βαθμούς Κελσίου και ως τεχνολογίας NTC thermistor η αντίσταση του μειώνεται όσο αυξάνεται η θερμοκρασία του. Για παράδειγμα, αν η θερμοκρασία φτάσει τους 100 βαθμούς η αντίσταση του μειώνεται στα 177Ω.

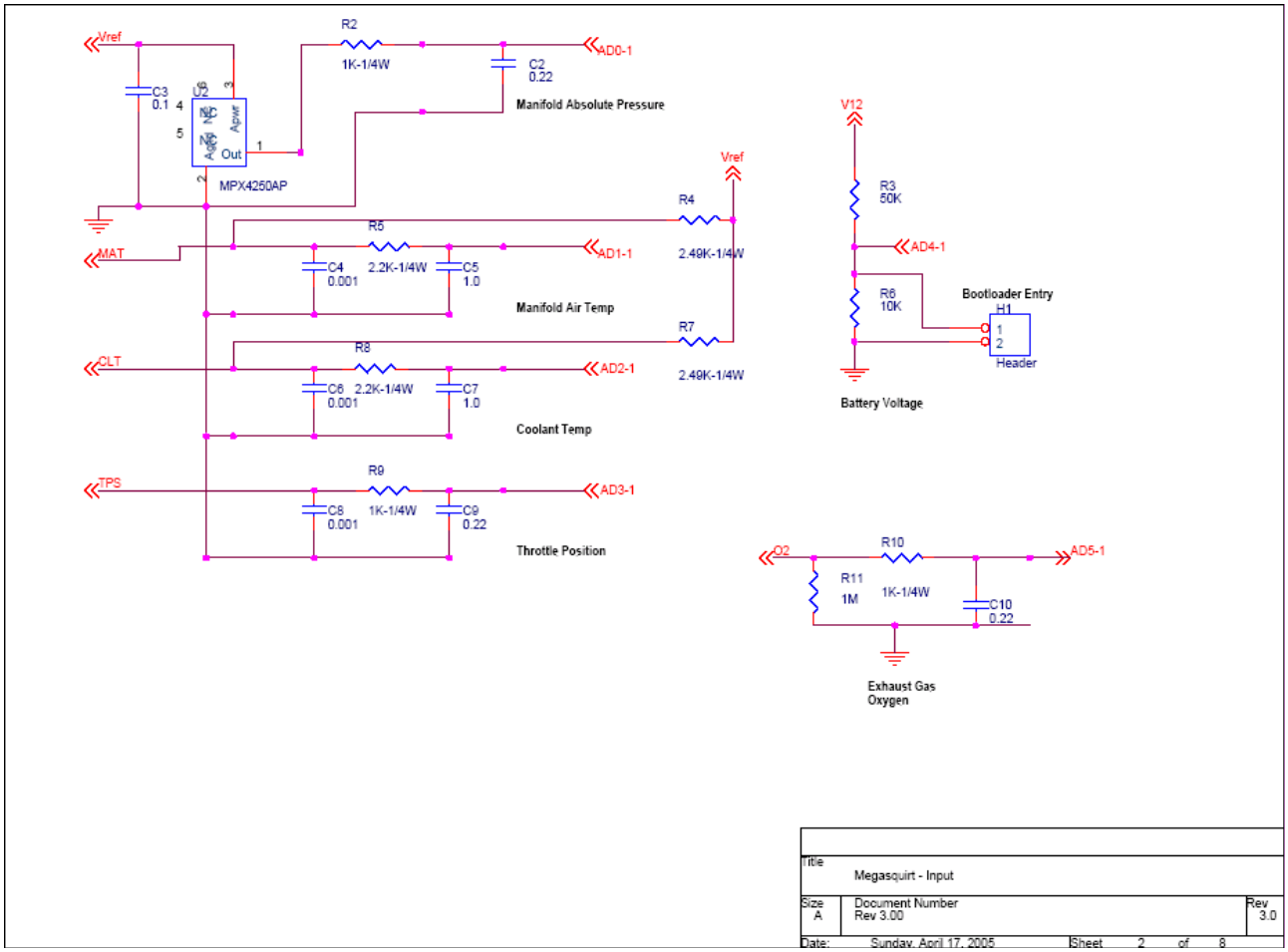
Το εύρος λειτουργίας είναι από -40 έως 150 βαθμούς Κελσίου και αντίσταση από 100ΚΩ έως 48Ω. Στον παρακάτω πίνακα παρουσιάζεται η τιμή της αντίστασης του αισθητήρα σε σχέση με την θερμοκρασία.

Temperature vs Resistance

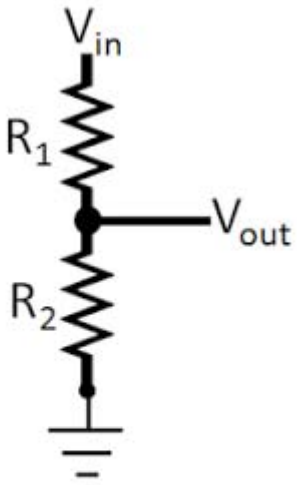
°C	°F	OHMS
Temperature vs Resistance Values (Approximate)		
150	302	47
140	284	60
130	266	77
120	248	100
110	230	132
100	212	177
90	194	241
80	176	332
70	158	467
60	140	667
50	122	973
45	113	1188
40	104	1459
35	95	1802
30	86	2238
25	77	2796
20	68	3520
15	59	4450
10	50	5670
5	41	7280
0	32	9420
−5	23	12300
−10	14	16180
−15	5	21450
−20	−4	28680
−30	−22	52700
−40	−40	100700

Η μεταβολή της αντίστασης του θερμίστορ NTC μετράται με την τοποθέτηση του αισθητήρα μέσα σε ένα βασικό κύκλωμα διαιρέτη τάσης όπου το θερμίστορ έχει συνδεθεί στον ακροδέκτη γείωσης του κυκλώματος. Δεδομένου όμως ότι ο ίδιος αισθητήρας συνδέεται με την ECU, αν ο μικροελεγκτής συνδεθεί στον αισθητήρα θα μπορεί να διαβάσει κάποια τάση.

Λαμβάνοντας υπόψιν τα σχηματικά διαγράμματα της ECU που φαίνονται στο παρακάτω σχήμα, φαίνεται ότι υπάρχει ένας διαιρέτης τάσης που δημιουργείται με την αντίσταση του αισθητήρα και μιας αντίστασης 2.49 ΚΩ.



Γνωρίζοντας λοιπόν την τιμή της pull-up αντίστασης του διαιρέτη τάσης καθώς και την αντίσταση του αισθητήρα σε κάθε θερμοκρασία μπορούμε να υπολογίσουμε την τάση που θα διαβάζει σε κάθε θερμοκρασία η ECU και κατ' επέκταση ο μικροελεγκτής μας.



$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

Vin = 5 V

R1 = 2.49 KΩ

R2 : η αντίσταση του αισθητήρα

Vout : η τάση που θα διαβάσει ο μικροελεγκτής

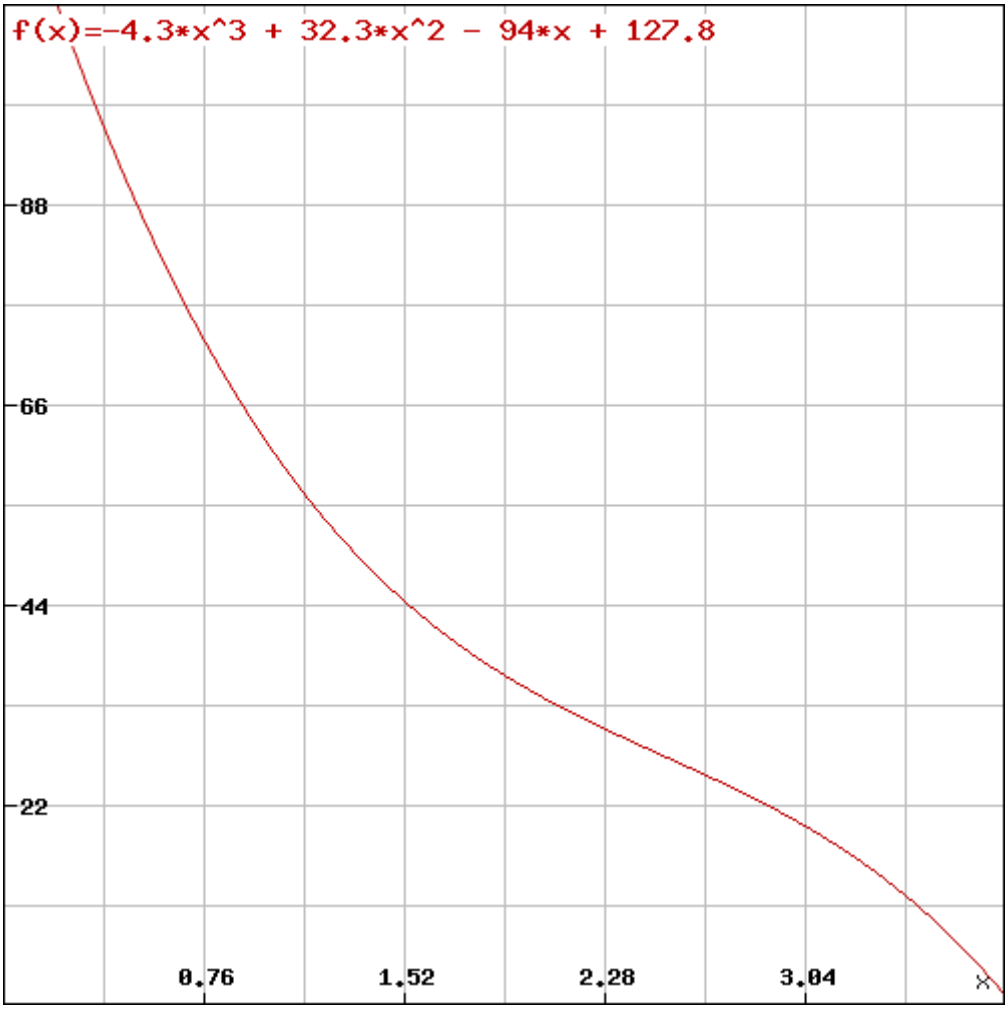
Λαμβάνοντας υπόψιν τα παραπάνω μπορεί να κατασκευαστεί ο παρακάτω πίνακας.

Temp	R2	Vout
5	7280	3,726
10	5670	3,474
20	3520	2,928
25	2796	2,645
35	1802	2,099
50	973	1,405
60	667	1,056
70	467	0,79
80	332	0,588
90	241	0,441
100	177	0,332
110	133	0,587

Με βάση τα στοιχεία του πίνακα πρέπει να κατασκευαστεί μια εξίσωση που να συνδέει τα Vout και Temp. Από το datasheet του αισθητήρα γνωρίζουμε ότι οι τιμές της αντίστασης ακολουθούν μια φθίνουσα καμπύλη. Για την προσέγγιση αυτής της καμπύλης χρησιμοποιήθηκε μια πολυωνυμική εξίσωση τρίτου βαθμού της μορφής :

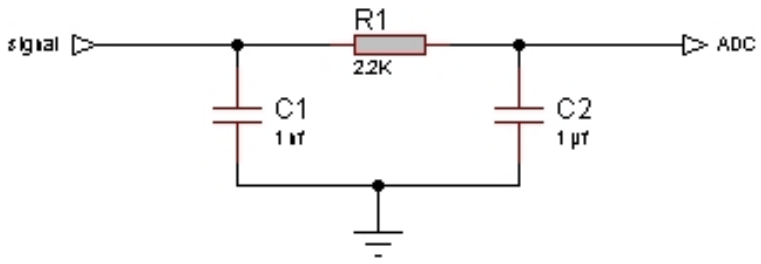
$$Temp = -4.3 \times V^3 + 32.3 \times V^2 - 94 \times V + 127.8$$

Vout	Temperature	Calculated Temperature	Error
3,726	5	3.681472774	1.318527226
3,4743	10	10.89455436	8.945543609·10-1
2,928	20	21.64689165	1.646891649
2,645	25	25.66814202	6.681420244·10-1
2,099	35	33.11606355	1.883936453
1,405	50	47.62814494	2.37185506
1,056	60	59.54713931	4.528606918·10-1
0,79	70	71.62908612	1.629086124
0,588	80	82.86843458	2.86843458
0,441	90	92.3035067	2.303506703
0,332	100	100.0376463	0.037646257
0,254	110	105.9789177	4.021082268



Βάσει της παραπάνω εξίσωσης, ο μικροελεγκτής είναι σε θέση διαβάζοντας την τάση που δίνει το παραπάνω κύκλωμα να μπορεί να υπολογίζει την αντίστοιχη θερμοκρασία και να την εμφανίζει στην οθόνη.

Το σήμα του αισθητήρα πριν φτάσει στον μικροελεγκτή περνάει πρώτα από ένα βαθυπερατό φίλτρο, έτσι ώστε το σήμα να φιλτράρεται πριν μπει στον μικροελεγκτή για αποκοπή θορύβου. Το φίλτρο αποτελείται από δύο πυκνωτές και μια αντίσταση και φαίνεται στο παρακάτω σχήμα.



Το σήμα συνδέεται στο pin 9 του μικροελεγκτή, το οποίο αντιστοιχεί στο RE1/AN6. Το pin ορίζεται μέσα στον κώδικα ως αναλογικό και ως είσοδος με τις εντολές:

```
ANSELE = 0x02;              // Configure RE1 pin as analog
TRISE1_bit = 1;             // Configure RE1 pin as input
```

Στη συνέχεια πρέπει να καλεστεί η συνάρτηση ADC_Init() η οποία αρχικοποιεί την λειτουργία για το ADC και πλέον ο μικροελεγκτής μπορεί να διαβάζει το σήμα που δέχεται σαν είσοδο με τη συνάρτηση ADC_Read(). Παρακάτω παρατίθεται το κομμάτι του κώδικα που εκτελείται ώστε να διαβάζεται κάθε φορά το σήμα.

```
/*
....
*/
unsigned long temp_res;
unsigned long p0,p1,p2,p3 ;
unsigned long temp_volt;
unsigned long temperature;
char temp[15];
```

```
/*
....
*/

void main(){
        ANSELB = 0;                 // set PORT B pins as digital
        TRISE0_bit = 0;             //Configure RE0 pin as output for the warning led
        ANSELE = 0x02;               // Configure RE1 pin as analog
        TRISE1_bit = 1;              // Configure RE1 pin as input
        /*
        ....
        */
        ADC_Init();                  // Initialize ADC
        /*
        ....
        */
        while(1){
                temp_res=0;
                temp_volt=0;
                temp_res = ADC_Read(6);    // Get 10-bit results of AD conversion
                calculate_temp(temp_res);
                /*
                ....
                */
 }
```

Η calculate_temp(temp_res) είναι μια συνάρτηση στην οποία γίνονται οι πράξεις της προηγούμενης εξίσωσης που αναφέρθηκε, ώστε να υπολογίζεται η θερμοκρασία. Επίσης, μετατρέπει τα τρία πρώτα νούμερα του αποτελέσματος σε κωδικούς ASCII τα αποθηκεύει σε έναν πίνακα χαρακτήρων, ώστε να μπορούν να εμφανιστούν στην οθόνη και τέλος, αν η θερμοκρασία ξεπεράσει τους 100 βαθμούς κάνει το pin 8 (RE0) ενεργό, ώστε να ανάψει ένα προειδοποιητικό led στο ταμπλό και να προειδοποιήσει τον οδηγό για υπερθέρμανση του κινητήρα. Ο κώδικας αυτής της συνάρτησης είναι ο ακόλουθος.

```
void calculate_temp(unsigned long temp_res) {

        temp_volt = temp_res*500/1023;          // digital result to volts
        // calculations for the equation
        p0 = 1278*1000000;
        p1 = 940*temp_volt*10000;
        p2 = 323*temp_volt*temp_volt*100;
        p3 = 43*temp_volt*temp_volt*temp_volt;
        temperature =p2-p3-p1+p0;
        temperature = temperature/10000000;
        temp[0] = (temperature/100)%10 + 48;    // first digit to ASCII
        temp[1] = (temperature/10)%10 + 48;     // second digit to ASCII
        temp[2] = temperature%10 + 48;          // third digit to ASCII
        Lcd_Out(1,16,temp);                     // output on LCD
        if (temperature>100){
                LATE0_bit = 1;                  // LED on
        }
        else{
                LATE0_bit = 0;                  // LED off
        }
}
```

4. Στροφές κινητήρα

Η ταχύτητα λειτουργίας ενός κινητήρα μετριέται σε στροφές ανά λεπτό (revolutions per minute ή RPM) και είναι ακόμα μια σημαντική μεταβλητή για την σωστή λειτουργία ενός αγωνιστικού οχήματος. Η μεταβλητή αυτή πρέπει να είναι γνωστή τόσο για την σωστή ρύθμιση του κινητήρα όσο και για την ενημέρωση που παρέχει στον οδηγό για την σωστή αλλαγή ταχυτήτων.

Η διαχείριση του κινητήρα του μονοθεσίου γίνεται από μια ηλεκτρονική κεντρική μονάδα (ECU) που ονομάζεται Megasquirt 3. Η ECU διαθέτει έξοδο για σύνδεση εξωτερικού στροφόμετρου. Η έξοδος αυτή είναι ένας τετραγωνικός παλμός, ο οποίος είναι 12V για υψηλό σήμα και 0V για χαμηλό και παράγει παλμούς για κάθε περιστροφή του στροφάλου. Μετρώντας λοιπόν αυτούς τους παλμούς σε συγκεκριμένο χρονικό διάστημα, μπορούν να υπολογιστούν οι στροφές του κινητήρα.

Ο μικροελεγκτής διαθέτει τον Timer0 που μπορεί να χρησιμοποιηθεί σαν timer ή counter. Μέσω του λογισμικού μπορεί να δηλωθεί η λειτουργία του είτε ως 8-bit είτε ως 16-bit. Η λειτουργία του ελέγχεται από τον καταχωρητή T0CON και η λειτουργικότητα κάθε bit του καταχωρητή αυτού φαίνεται στον παρακάτω πίνακα. Στην παρούσα εργασία ο Timer0 ορίσθηκε ως 16-bit counter ώστε να μετράει τον αριθμό των παλμών που φτάνουν στο pin 6 (RA4-T0CKI). Επίσης, επειδή ο μικροελεγκτής μπορεί να έχει σαν είσοδο σήμα μέχρι 5V, χρησιμοποιήθηκε ένας διαιρέτης τάσης ώστε να μειωθούν τα 12V σε 5V.

T0CON register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Για να ορισθεί το διάστημα που διαβάζεται ο Timer0 και να υπολογίζονται οι στροφές, χρησιμοποιήθηκε ο Timer1, ο οποίος με τον κατάλληλο ορισμό των καταχωρητών T1CON και INTCON μπορεί να δημιουργεί interrupts σε συγκεκριμένο χρόνο. Τα interrupts ειδοποιούν τον επεξεργαστή να σταματήσει την όποια λειτουργία του και να εκτελέσει ένα συγκεκριμένο κομμάτι κώδικα, όταν συμβεί κάποιο γεγονός. Για τον σωστό υπολογισμό των στροφών του κινητήρα επιλέχθηκε ένα χρονικό διάστημα 0.25 sec. Αυτό το νούμερο επιλέχθηκε διότι αν η δειγματοληψία γίνονται πιο γρήγορα, το αποτέλεσμα θα μπορούσε να αμφισβητηθεί, καθώς το δείγμα θα έπρεπε να πολλαπλασιαστεί με μεγαλύτερο αριθμό ώστε να υπολογιστούν οι στροφές ανά λεπτό, ενώ αν γινόταν σε μεγαλύτερο χρονικό διάστημα, ο οδηγός ίσως έβλεπε λιγότερες στροφές από τις κανονικές, αφού πιθανόν το αποτέλεσμα δεν θα μπορούσε να ανανεωθεί τόσο γρήγορα όσο επιταχύνει ο κινητήρας.

Ο χρόνος που συμβαίνει κάθε interrupt υπολογίζεται με τον παρακάτω τύπο.

$$t = \frac{1}{F_{osc}/4} \cdot prescaler \cdot (2^n)$$

Ο ορισμός αυτών των μεταβλητών γίνεται με την κατάλληλη ανάθεση των bit του καταχωρητή T1CON. Ο πίνακας από το datasheet του μικροελεγκτή που δείχνει την χρήση κάθε bit είναι ο παρακάτω.

12.13 Register Definitions: Timer1/3/5 Control

REGISTER 12-1: TXCON: TIMER1/3/5 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/0	R/W-0/u
TMRxCS<1:0>		TxCKPS<1:0>		TxSOSCEN	$\overline{\text{TxSYNC}}$	TxRD16	TMRxON
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **TMRxCS<1:0>**: Timer1/3/5 Clock Source Select bits
- 11 = Reserved. Do not use.
- 10 = Timer1/3/5 clock source is pin or oscillator:
  - If TxSOSCEN = 0:
  - External clock from TxCKI pin (on the rising edge)
  - If TxSOSCEN = 1:
  - Crystal oscillator on SOSCI/SOSCO pins
- 01 = Timer1/3/5 clock source is system clock (FOSC)
- 00 = Timer1/3/5 clock source is instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>**: Timer1/3/5 Input Clock Prescale Select bits
- 11 = 1:8 Prescale value
- 10 = 1:4 Prescale value
- 01 = 1:2 Prescale value
- 00 = 1:1 Prescale value

bit 3 **TxSOSCEN**: Secondary Oscillator Enable Control bit
- 1 = Dedicated Secondary oscillator circuit enabled
- 0 = Dedicated Secondary oscillator circuit disabled

bit 2 **$\overline{\text{TxSYNC}}$**: Timer1/3/5 External Clock Input Synchronization Control bit

TMRxCS<1:0> = 1X
- 1 = Do not synchronize external clock input
- 0 = Synchronize external clock input with system clock (FOSC)

TMRxCS<1:0> = 0X

This bit is ignored. Timer1/3/5 uses the internal clock when TMRxCS<1:0> = 1X.

bit 1 **TxRD16**: 16-Bit Read/Write Mode Enable bit
- 1 = Enables register read/write of Timer1/3/5 in one 16-bit operation
- 0 = Enables register read/write of Timer1/3/5 in two 8-bit operation

bit 0 **TMRxON**: Timer1/3/5 On bit
- 1 = Enables Timer1/3/5
- 0 = Stops Timer1/3/5
  Clears Timer1/3/5 Gate flip-flop

Ο καταχωρητής αρχικοποιήθηκε με την τιμή T1CON = 0b00001111 και έτσι ορίζεται σε λειτουργία 16-bit, με prescaler 1 και clock source = instruction clock (FOSC/4).

Έτσι λοιπόν και με βάση τον παραπάνω τύπο έχουμε: $t = \frac{1}{1\text{Mhz}/4} \times 1 \times 2^{16} = 0.25\,sec$

Επίσης, ο TIMER1 πρέπει να προφορτωθεί με κάποια συγκεκριμένη τιμή ώστε να πετύχουμε αυτόν τον ακριβή χρόνο. Στην προκειμένη περίπτωση είναι η τιμή 0xBDC.

Έχοντας ορίσει όλα τα παραπάνω ο μικροελεγκτής είναι σε θέση πλέον να υπολογίσει τις στροφές του κινητήρα. Κάθε 0.25 sec γίνεται δημιουργείται ένα interrupt και γίνεται ανάγνωση της τιμής του TIMER0. Οι στροφές ανά λεπτό θα είναι: $RPM_{Value} = \frac{T0 \times 60}{0.25}$

όπου T0 είναι η πραγματική μέτρηση του TIMER0 (revolution per cycle). Το T0 πολλαπλασιάζεται με 60 (sec/min) και διαιρείται με το χρόνο δειγματοληψίας 0.25 sec. Μόλις ολοκληρωθούν οι υπολογισμοί, ο TIMER0 μηδενίζεται ώστε να αρχίσει να μετράει ξανά.

Ο μικροελεγκτής γνωρίζοντας τις στροφές του κινητήρα, αναλαμβάνει να ανάψει τα ανάλογα LED στο ταμπλό του μονοθεσίου καθώς και να παρουσιάσει στην LCD οθόνη την τιμή που υπολόγισε. Σε αυτό το σημείο να σημειωθεί ότι η ανανέωση των LED γίνεται μετά από κάθε υπολογισμό της νέας τιμής των στροφών, δηλαδή κάθε 0.25 sec ενώ η ανανέωση της τιμής της οθόνης γίνεται μέσα στην main συνάρτηση του μικροελεγκτή. Αυτό γίνεται διότι τα LED αποτελούν την κύρια διεπαφή για την ενημέρωση του οδηγού για το σημείο που βρίσκεται ο κινητήρας και έτσι χρειάζονται πιο γρήγορη ανανέωση.

Παρακάτω παρουσιάζεται το κομμάτι του κώδικα του μικροελεγκτή που υλοποιεί τον υπολογισμό των στροφών του κινητήρα και την ανανέωση της οθόνης και των LED.

```
/*...*/
//ορισμός των led για το PORTD
#define LED1 0b00000001
#define LED2 0b00000011
#define LED3 0b00000111
#define LED4 0b00001111
#define LED5 0b00011111
#define LED6 0b00111111
#define LED7 0b01111111
#define LED8 0b11111111
```

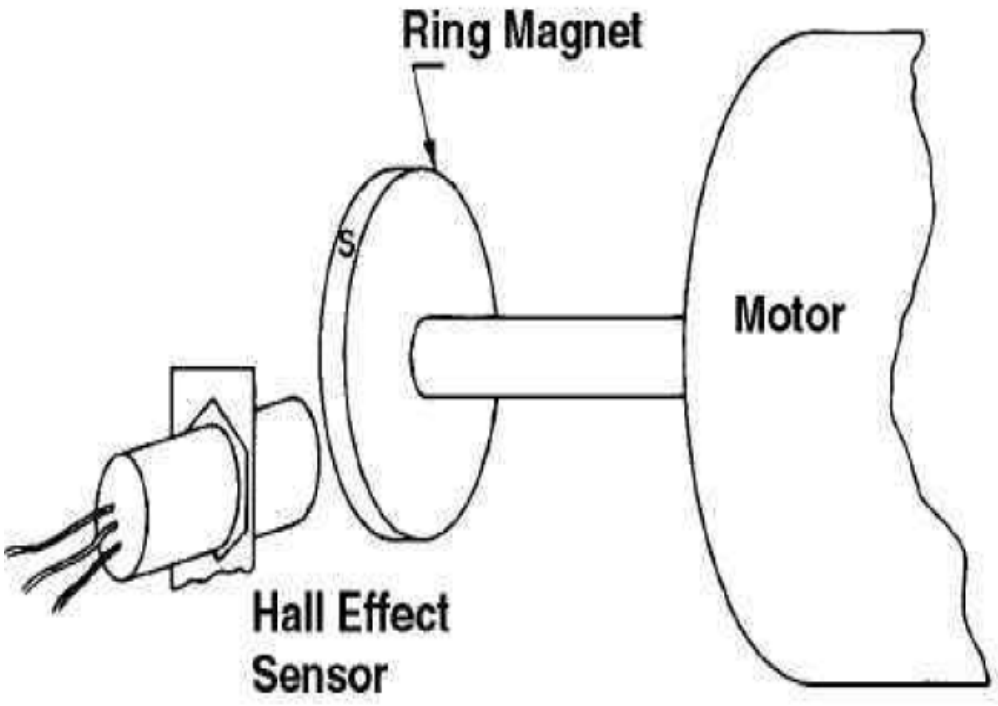
```
/*...*/
unsigned long RPM_Value;
sbit pulses at RA4_bit;
/*...*/
//μετατροπή του αριθμού σε χαρακτήρες ώστε να εμφανιστούν στην οθόνη
char *RPM = "00000 RPM";
void Display_RPM(unsigned long num){

        RPM[0] = num/10000 + 48;
        RPM[1] = (num/1000)%10 + 48;
        RPM[2] = (num/100)%10 + 48;
        RPM[3] = (num/10)%10 + 48;
        RPM[4] = num%10 + 48;

        Lcd_Out(2,12,RPM);
 }

void on_leds (long rpm){ //function to switch on the leds depending the rpm
if(rpm>1000)
{
PORTD=LED1;
}
if(rpm>2500)
{
PORTD=LED2;
}
if(rpm>3500)
{
PORTD=LED3;
}
if(rpm>5000)
{
PORTD=LED4;
```

```
}
if(rpm>6200)
{
PORTD=LED5;
}
if(rpm>7200)
{
PORTD=LED6;
}
if(rpm>9000)
{
PORTD=LED7;
}
if(rpm>11000)                    //led flashing as a warning for the driver
{
PORTD=LED8;
Delay_ms(100);                    // 100 ms delay
PORTD=LED7;
Delay_ms(100);
}
}

/*...*/

void interrupt() {
        if (TMR1IF_bit) {

                 pulses = 0;
                T0CON.TMR0ON = 0;    // Stop the timer
                RPM_Value = ((TMR0H << 8) + TMR0L)*240;

                 on_leds(RPM_Value);
```

```
            TMR0L = 0;
            TMR0H = 0;
            T0CON.TMR0ON = 1;
            pulses = 1;
            TMR1IF_bit = 0;         // clear TMR0IF
            TMR1H = 0x0B;
            TMR1L = 0xDC;
        }
}


void main(){
        /*...*/
         ANSELB = 0;                  // set PORT B pins as digital for the LCD
         TRISD = 0;                   // set direction to be output for the leds
        T0CON = 0b01101000; //configuring TIMER0 for pulse count
        T1CON = 0b00001111; //configuring TIMER1 for time count
        TMR1IF_bit = 0;           // clear TMR1IF
        TMR1H = 0x0B;              // Initialize Timer1 register
        TMR1L = 0xDC;
        TMR1IE_bit = 1;           // enable Timer1 interrupt
        INTCON = 0xC0;             // Set GIE, PEIE
        T0CON.TMR0ON = 1;
        TMR0L = 0;
        TMR0H = 0;
        pulses = 1;
        /*...*/
        while(1){
        /*...*/
        Display_RPM(RPM_Value);
        /*...*/
        }
}
```

5.Ταχύτητα μονοθεσίου

Η ταχύτητα του μονοθεσίου μετράται με έναν αισθητήρα φαινομένου HALL, ο οποίος είναι τοποθετημένος στο κιβώτιο ταχυτήτων και μετράει τις περιστροφές του άξονα που δίνει κίνηση στο εξωτερικό γρανάζι μετάδοσης του κινητήρα. Η λειτουργία του αισθητήρα αυτού βασίζεται σε έναν μαγνήτη που είναι στον άξονα που περιστρέφεται και ένα σταθερό κομμάτι όπως φαίνεται στην παρακάτω εικόνα.



Όταν ο μαγνήτης περνάει μπροστά από τον αισθητήρα, παράγεται ένας τετραγωνικός παλμός. Επομένως, για κάθε περιστροφή του άξονα δημιουργείται ένας παλμός.

Μέσω του παραπάνω άξονα και του εξωτερικού γραναζιού που είναι πάνω σε αυτόν και μέσω αλυσίδας δίνει κίνηση σε ένα μεγαλύτερο γρανάζι που βρίσκεται στο διαφορικό του οχήματος και έτσι περιστρέφονται οι τροχοί. Έτσι, για να υπολογιστεί η ταχύτητα του μονοθεσίου αρκεί να είναι γνωστό οι περιστροφές του άξονα σε κάποιο συγκεκριμένο χρονικό διάστημα, ο λόγος μετάδοσης των δυο γραναζιών, δηλαδή μια περιστροφή του άξονα στο κιβώτιο ταχυτήτων σε πόσες περιστροφές αντιστοιχεί στον άξονα μετάδοσης της κίνησης, και η ακτίνα των τροχών του μονοθεσίου. Ο τύπος που δίνει την ταχύτητα είναι: $$Speed = \frac{pulses/second}{drive\ ratio} \times 2\pi r \times 3.6$$

Το drive ratio στο συγκεκριμένο μονοθέσιο είναι 3.33 και η ακτίνα των τροχών του είναι 0,254 m.

Η μέτρηση των παλμών στον μικροελεγκτή έγινε με τον ίδιο τρόπο όπως περιγράφηκε στο κεφάλαιο για την μέτρηση των στροφών του κινητήρα. Η μόνη διαφορά είναι ο χρόνος που γίνεται η ανάγνωση του περιεχομένου του TIMER0. Λόγω του ότι η ενημέρωση της οθόνης για την την ταχύτητα του μονοθεσίου δεν χρειάζεται να είναι τόσο γρήγορη όσο στην περίπτωση των στροφών, αποφασίστηκε ο χρόνος των interrupts να είναι 0.5 δευτερόλεπτα. Για να επιτευχθεί αυτός ο χρόνος, ο καταχωρητής T1CON ορίζεται όπως στην προηγούμενη περίπτωση με την μόνη διαφορά ότι τα bit5-6 θα έχουν τιμή 01 αντί για 00 ώστε ο prescaler να γίνει 2 και να έχουμε interrupt κάθε 0.5sec.

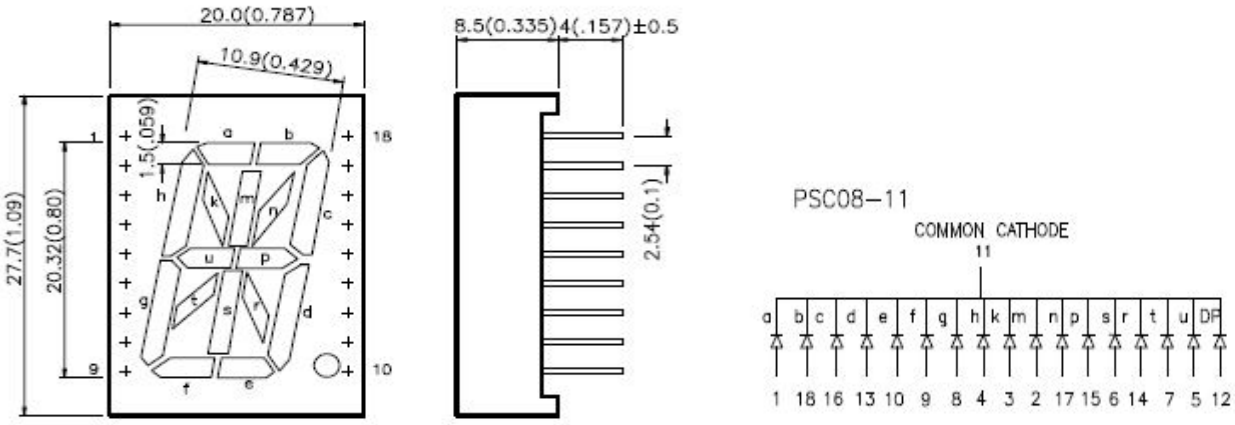
Παρακάτω παρουσιάζεται ο κώδικας για την υλοποίηση του υπολογισμού της ταχύτητας του μονοθεσίου.

```
/*...*/
unsigned long sprocket_rps;
unsigned long KMH_Value;
/*...*/

void interrupt() {
        if (TMR1IF_bit) {

                 pulses = 0;
                T0CON.TMR0ON = 0;    // Stop the timer
                sprocket_rps = ((TMR0H << 8) + TMR0L);
                TMR0L = 0;
                TMR0H = 0;
                T0CON.TMR0ON = 1;
                pulses = 1;
                TMR1IF_bit = 0;          // clear TMR0IF
                TMR1H = 0x0B;
                TMR1L = 0xDC;
        }
}
```

```
void main(){
            /*...*/
             T0CON = 0b01101000; //configuring TIMER0 for pulse count
             T1CON = 0b00011111; //configuring TIMER1 for time count
             TMR1IF_bit = 0;           // clear TMR1IF
             TMR1H = 0x0B;              // Initialize Timer1 register
             TMR1L = 0xDC;
             TMR1IE_bit = 1;           // enable Timer1 interrupt
             INTCON = 0xC0;             // Set GIE, PEIE
             T0CON.TMR0ON = 1;
             TMR0L = 0;
             TMR0H = 0;
             pulses = 1;
             /*...*/
             while(1){
             /*...*/
             KMH_Value = sprocket_rps*3.44;
             Display_KMH(KMH_Value);
             /*...*/
            }
}

char *KMH = "000 km/h";
void Display_KMH(unsigned long num){
          KMH[0] = (num/100)%10 + 48;
          KMH[1] = (num/10)%10 + 48;
          KMH[2] = num%10 + 48;
          Lcd_Out(3,11,KMH);
}
```

6.Υπολογισμός σχέσης κιβωτίου ταχυτήτων

Στο επόμενο στάδιο περιγράφεται η αναπαράσταση της σχέση του κιβωτίου ταχυτήτων. Για το σκοπό αυτό χρησιμοποιήθηκε μια οθόνη LED-16 ψηφίων η οποία έχει τις ενδείξεις «1, 2, 3, 4, 5, 6, N» , όπου ¨N¨ συμβολίζεται η Νεκρά(Neutral). Στόχος είναι η οθόνη να απεικονίζει κάθε φορά τις αλλαγές που γίνονται στο κιβώτιο ταχυτήτων για να βοηθάει τον οδηγό ώστε να έχει καλύτερο έλεγχο του αυτοκινήτου και βελτίωση των επιδόσεων κατά την αγωνιστική οδήγηση. Βεβαίως, η χρήση αυτής της λειτουργίας μπορεί να είναι πολλαπλή καθώς επιτρέπει την ικανότητα να αναπαρίσταται η 'Νεκρά' όπου επιτρέπει την ελεύθερη κίνηση των τροχών από αυτήν του κινητήρα καθώς και τη δυνατότητα να παραμένει σε στάση το όχημα με την λειτουργία του κινητήρα ενεργή χωρίς να χρειάζεται η χρήση του συμπλέκτη.



Το σύστημα αναπαράστασης της σχέσης του κιβωτίου συνδυάζεται με ένα ηλεκτρομαγνητικό μηχανισμό εναλλαγής ταχυτήτων της Kliktronic ο οποίος επιτρέπει οι αλλαγές να γίνονται με κουμπιά(Push On) στο τιμόνι του αυτοκινήτου. Ένα κουμπί προκαλεί την κίνηση προς τη μία κατεύθυνση(ανέβασμα- up shift) και το άλλο προς την άλλη(κατέβασμα- down shift)

Ο μηχανισμός KLIKTRONIC:

Η συνδεσμολογία μεταξύ του μηχανισμού και των κουμπιών είναι σχετικά απλή καθώς ο μηχανισμός τροφοδοτείται με ρεύμα από τα relays των αντίστοιχων κουμπιών. Τα κουμπιά είναι συνδεδεμένα στην είσοδο της γείωσης των αντίστοιχων relays. Έτσι όταν το κουμπί πατηθεί, κάνει επαφή με τη γείωση στο relay και αυτό σαν έξοδο τροφοδοτεί με ρεύμα τον μηχανισμό ανάλογα με την κίνηση που θέλουμε να κάνει.

Στο σύστημα μας χρησιμοποιείτε μία εργοστασιακή έξοδο του κινητήρα, όπου όταν στον κιβώτιο δεν υπάρχει κάποια σχέση τότε στέλνεται ένα σήμα 0V, το οποίο μπορεί να συμπεριληφθεί σαν είσοδο στον μικροελεγκτή για την αναπαράσταση της ΄Νεκράς- Neutral΄.

Το πιο σημαντικό κομμάτι του συστήματος της αναπαράστασης της σχέσης του κιβωτίου είναι ένα περιστροφικό ποτενσιόμετρο τύπου 21176-0002 το οποίο λειτουργεί με τάση αναφοράς τα 5V και η έξοδος που επιστρέφει τη διαβάζεται σαν είσοδος στον μικροελεγκτή. Οι τιμές που επιστρέφει με αυτήν την τάση αναφοράς σαν είσοδο, κυμαίνονται από 3,27V έως 2,43V και αντιστοιχίζονται μέσω μία συνάρτησης της βιβλιοθήκης, την ADC_Read(), σε αναγνώριση κίνησης προς την μία ή την άλλη κατεύθυνση από τον κώδικα.

.

Το ποτενσιόμετρο αυτό είναι συνδεδεμένο με μία προέκταση που έχει τοποθετηθεί στον άξονα όπου συνδέεται το πεντάλ αλλαγής ταχυτήτων.

Είναι αριστερόστροφο για αυτό το λόγο όταν τοποθετείται, βρίσκεται σε μία προφορτισμένη κατάσταση όπου μπορεί να οριστεί κατάσταση ηρεμίας και έχοντας σαν είσοδο τα 5V η τάση που επιστρέφει στην κατάσταση αυτή είναι τα 2,87V. Με αυτό τον τρόπο είναι εφικτό να διαβάζονται οι αλλαγές στην επιστρεφόμενη τάση και να αντιλαμβάνεται το σύστημα προς τα ποια κατεύθυνση κινήθηκε ο άξονας, δηλαδή αν έγινε ανέβασμα ή κατέβασμα. Επιπλέον βοηθάει στην αναγνώριση εγκαίρως κάποιου σφάλματος στην διαδικασία, όπως για παράδειγμα να μην ολοκληρωθεί η αλλαγή ταχύτητας παρόλο που έχει πατηθεί το κουμπί.

Έτσι αρχικά ορίζονται οι έξοδοι που θα χρησιμοποιηθούν από τον μικροελεγκτή για αυτήν την αναπαράσταση και στην συνέχεια τις αντιστοιχίζονται με τα Pins της οθόνης . Στη συνέχεια ορίζονται τα σύμβολα που θέλουμε να αναπαρασταθούν και αυτά όπως αναφέρθηκαν και προηγουμένως είναι τα « 1, 2, 3, 4, 5, 6, N, --». Παρατηρούμε ότι θα χρειάζονται 8 σύμβολα αναπαράστασης. Το σύμβολο « -- » χρησιμοποιείται όταν ενεργοποιείται το σύστημα μας. Αυτό γίνεται διότι πρέπει να γίνει αρχικοποίηση καθώς είναι επιθυμητό να αποφευχθούν σφάλματα τα οποία μπορεί να προέρχονται από αλλαγές στο κιβώτιο ενώ αυτό ήταν ανενεργό. Έτσι όταν τροφοδοτηθεί το σύστημα με ρεύμα, στην οθόνη υπάρχει αυτή η ένδειξη μέχρι να δεχθεί σαν είσοδο, το σήμα από τον κινητήρα 0V και να γίνει με αυτό τον τρόπο η αρχικοποίηση στον μετρητή.

Στον κώδικα έχει οριστεί ότι τα PortB kai PortD ως εξόδους και συνδεδεμένα με τα pins της οθόνης. Έτσι αρχικά ορίζονται τα σύμβολα που θέλουμε να αναπαρασταθούν:

```
#define D0  0b00010001
#define B0  0b00000000                    //   «  - - »

#define B1  0b00000000
#define D1  0b01000100

#define B2  0b11101110
#define D2  0b00010001

#define B3  0b11111100
#define D3  0b00010001

#define B4  0b00110001
#define D4  0b00010001

#define B5  0b11011101
#define D5  0b00010001

#define B6  0b11011111
#define D6  0b00010001

#define BN  0b00110011
#define DN  0b10001000


/*
…
*/
```

Έπειτα γίνεται η αρχικοποίηση του προγράμματος και ορίζονται τα Pins του μικροελεκτή για το ποια θα είναι η λειτουργία τους (είσοδος/έξοδος,αναλογική/ψηφιακή,)

```
void initialization (void)
{

  TRISA=0;
  TRISB=0;
  TRISD=0;
  ANSELA=0;
  ANSELB=0;
  ANSELD=0;
  LATA=0;
  LATB=0;
  LATD=0;
  ANSELC = 0b11111100;
  TRISC  = 0xff;
  ANSELE = 0;
  TRISE  = 0xff;
   tempmax=-1;
   gpt=-1;
 } }
/*
…
*/
```

Στη συνέχεια το πρόγραμμα προχωρά στην κλήση της main() και ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν:

```
void main (void)
{
 initialization();
 tempN=10000;
 gpt=-1;
/*
…
*/
```

Έπειτα στο κυρίως κομμάτι του κώδικα όπου εδώ γίνεται ο υπολογισμός της σχέσης του κιβωτίου του κινητήρα λαμβάνοντας υπ' όψιν όλα τα σήματα και το ποτενσιόμετρο το οποίο έχουν τοποθετηθεί.

Αρχικά έχουν ληφθεί τα κουμπιά (Push on) ως είσοδοι στο κύκλωμα μας. Έτσι όταν πατηθεί το κουμπί για ανέβασμα ταχύτητας (up-shift) το σήμα των 0V στέλνεται και στις εισόδους του μικροελεγκτή που βρίσκονται στο PortC και μέσω της συνθήκης (Button(&PORTC, 0,15,0)) ελέγχεται αν έχει πατηθεί το κουμπί για τουλάχιστον 15ms που είναι αρκετό για να βεβαιωθούμε ότι ο οδηγός έχει δώσει εντολή για αλλαγή ταχύτητας. Φυσικά όπως αναφέρθηκε και πιο πριν για να υπάρξει ένδειξη θα πρέπει να έχει πάρει είσοδο πρώτα ο μικροελεγκτή ότι έχει αρχικοποιηθεί το σύστημα με τα 0V από το σήμα του κινητήρα, της Νεκράς. Για το σκοπό αυτό έχουν τοποθετηθεί όπως διακρίνεται και στον κώδικα ορισμένες μεταβλητές που έχουν το ρόλο της σήμανσης(flags). Έτσι παρατίθεται ο κώδικας:

```
/*
…
*/


if ((Button(&PORTC, 0,15,0))&&(gpt!=-1))

        flagup=0;

         while(tempPot2<tempPot1){
         tempPot1= ADC_Read(15);
         tempPot2= ADC_Read(15);
                        PORTB=BN;
                         PORTD=DN;
          if(tempPot1<tempPot2 && flagup==1) break;
          if(tempPot1<512){tempmin=tempPot1;}break;
          if(tempPot2<512){tempmin=tempPot2;}break;
            flagup=1;
          } ;
```

```
      if(tempmin<512) {
         if(gpt==0){
            gpt=2;
         }
         if(tempN<4){
         gpt==2;
         }
         else{  gpt++;
           if(gpt>6){
           gpt=6;
           }
         }
      }


          Delay_ms(20);

 }

     if (Button(&PORTC, 1,15,0)&&(gpt!=-1)){
 tempN = ADC_Read(16);

flagdn=0;

 while(tempPot2>tempPot1);{
 tempPot1= ADC_Read(15);
 tempPot2= ADC_Read(15);
       break;
  if(tempPot2<tempPot1 && flagdn==1)break;
  if(tempPot1>673){tempmax=tempPot1;}break;
  if(tempPot2>673){tempmax=tempPot2;}break;
    flagdn=1;
  }
```

```
}

        if(tempmax>673){
            if(gpt==0){
               gpt=1;
            }
            if(tempN<4){
            gpt==1;
            }
            else  gpt--;
              if(gpt<1){
              gpt=1;
              }
            }
```

Και έπειτα στην αναπαράσταση της αποτίμησης στην οθόνη:

```
/*
…
 */
if(gpt==-1){
            PORTB=B0;
            PORTD=D0;

    }
    if(gpt==0){
            PORTB=BN;
            PORTD=DN;

    }

    if(gpt==1){
            PORTB=B1;
            PORTD=D1;
```

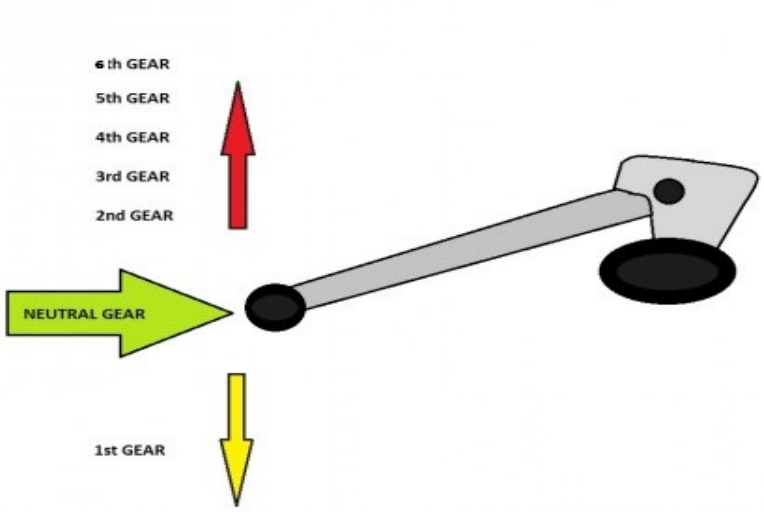
```
}

if(gpt==2){
        PORTB=B2;
        PORTD=D2;

}

if(gpt==3){
        PORTB=B3;
        PORTD=D3;

}

if(gpt==4){
        PORTB=B4;
        PORTD=D4;

}

if(gpt==5){

        PORTB=B5;
        PORTD=D5;

}

if(gpt==6){
        PORTB=B6;
        PORTD=D6;
}
```

Κατά την ανάγνωση της τιμής της τάσης από το ποτενσιόμετρο διαπιστώθηκε ότι δεν μπορούσε να υπολογιστεί ακριβώς ο χρόνος όπου ο μικροελεγκτής θα διάβαζε στιγμιότυπο από την τιμή, με αποτέλεσμα παρόλο που η επιστρεφόμενη τάση ήταν αρκετή έτσι ώστε να επιβεβαιωθεί ότι έγινε αλλαγή ταχύτητας ο μικροελεγκτής μπορεί να τύχαινε να διαβάσει μία μικρότερη ή μία μεγαλύτερη στην αντίστροφη διαδικασία.

Για το σκοπό αυτό προστέθηκε ένας βρόγχος όπου ο ρόλος του είναι να διαβάζει συνεχώς στιγμιότυπα όταν πατηθεί ένα από τα δύο κουμπιά για αλλαγή ταχύτητας. Έτσι λοιπόν για κατέβασμα ο βρόγχος αυτός διαβάζει και τις τιμές της τάσης μέχρι να βρει μία μικρότερη. Με αυτόν τον τρόπο επιτυγχάνεται η εξασφάλιση ότι θα αναγνωσθεί η μεγαλύτερη επιστρεφόμενη τάση και στη συνέχεια αυτή θα ελεγχθεί εάν είναι αρκετή για να αποφασιστεί αλλαγή ή όχι και έπειτα εμφάνιση στην οθόνη. Εφόσον το ποτενσιόμετρο μας είναι περιστροφικό η αντίστροφη διαδικασία γίνεται για αλλαγή ταχύτητας ανέβασμα- up shift. Δηλαδή θα διαβάζει τιμές και θα κρατάει την μικρότερη έως ότου βρει μία μεγαλύτερη όπου και τερματίζει.
Για την ανάγνωση της τάσης από τον μικροελεγκτή χρησιμοποιείτε όπως προαναφέρθηκε η συνάρτηση ADC_Read().

	Accept Down	**No valid down**	**Normal**	**No valid up**	**Accept Up**
Vout	3,33V	3,27V	2,87V	2,60V	2,43V
ADC_Read	682	670	588	532	498

Επειδή το κιβώτιο ταχυτήτων στις μοτοσυκλέτες έχει λίγο διαφορετικό τρόπο λειτουργίας, παρακάτω παραθέτεται ένα σχήμα για το πώς γίνονται οι αλλαγές:

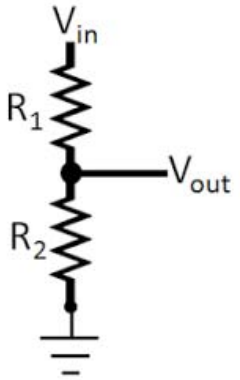


Φαίνεται λοιπόν ότι υπάρχει η Νεκρά ανάμεσα στην πρώτη και δευτέρα ταχύτητα όπου σε εκείνο το σημείο ο κινητήρας στέλνει ένα σήμα 0V. Έτσι έχει γραφτεί ένας έλεγχος στον κώδικα όπου κάθε φορά όπου βρίσκεται στην θέση δύο και έχει πατηθεί το κουμπί για κατέβασμα, περιμένει ο μικροελεγκτής να δεχθεί ένα σήμα 0V για να επιβεβαιώσει ότι βρίσκεται στο σωστό πεδίο υπολογισμού. Αν όμως ο μικροελεγκτής δεχθεί σήμα 0V ενώ βρίσκεται σε κατέβασμα από τρίτη σε δευτέρα ταχύτητα τότε αναγνωρίζει ότι έχει υπολογίσει λάθος και λειτουργεί σύμφωνα με το σήμα των 0V του κινητήρα. Το ίδιο ισχύει και για το ανέβασμα από πρώτη σε δευτέρα και για τα ανεβάσματα. Με αυτόν τον τρόπο το σύστημα μπορεί και διορθώνει πιθανά σφάλματα αλλά και επιβεβαιώνει το σωστό τρόπο λειτουργίας.

7. Στάθμη καυσίμου

Για την ένδειξη της στάθμης του καυσίμου, χρησιμοποιείται ένα γραμμικό μαγνητικό φλοτέρ το οποίο έχει 10 επίπεδα αντιστάσεων. Αυτά είναι από 10 έως 180 ohm.

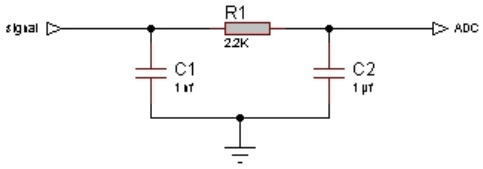
Δημιουργώντας ένα διαιρέτη τάσης με R1 μια αντίσταση 39ohm και την εκάστοτε αντίσταση στην είσοδο του μικροελεγκτή αποτιμάται το επίπεδο της στάθμης του καυσίμου. Η τάση αναφοράς είναι στα 5V και μέσω της συνάρτησης ADC_Read() τη διαβάζει ο μικροελεγκτής.



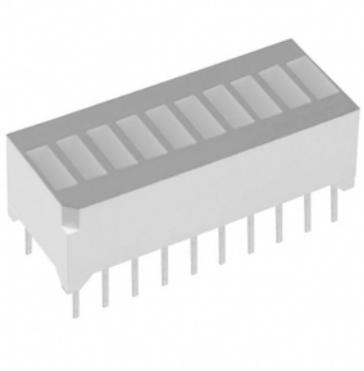
$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

Vin	5	5	5	5	5	5	5	5	5	5
R1	39	39	39	39	39	39	39	39	39	39
R2	10	28,88	47,76	66,64	85,52	104,4	123,28	142,16	161,04	179,92
Vout	1,02	2,13	2,75	3,15	3,43	3,64	3,80	3,92	4,03	4,11

Επίσης για την αποκοπή του θορύβου το σήμα πριν εισαχθεί στον μικροελεγκτή περνάει και αυτό από ένα βαθυπερατό φίλτρο όπως και στην περίπτωση του αισθητήρα θερμοκρασίας αποτελούμενο από δύο πυκνωτές και μία αντίσταση.



Ο μικροελεγκτής βγάζει 5 εξόδους όπου συνδέονται σε δύο γραμμές Led ο καθένας για την αναπαράσταση της στάθμης. Όλες οι έξοδοι του μικροελεγκτή που οδηγούν σε Led περνάνε από αντίσταση 330ohm για την ομαλή λειτουργία και την κατάλληλη φωτεινότητα.



Προχωρώντας στο κομμάτι του κώδικα στο οποίο έχουμε ορίσει τις έννοιες LED1, LED2,LED3,LED4 και LED5 διαβάζουμε από το pin 14, όπου έχουμε ορίσει την έξοδο του διαιρέτη,

Έτσι λοιπόν:

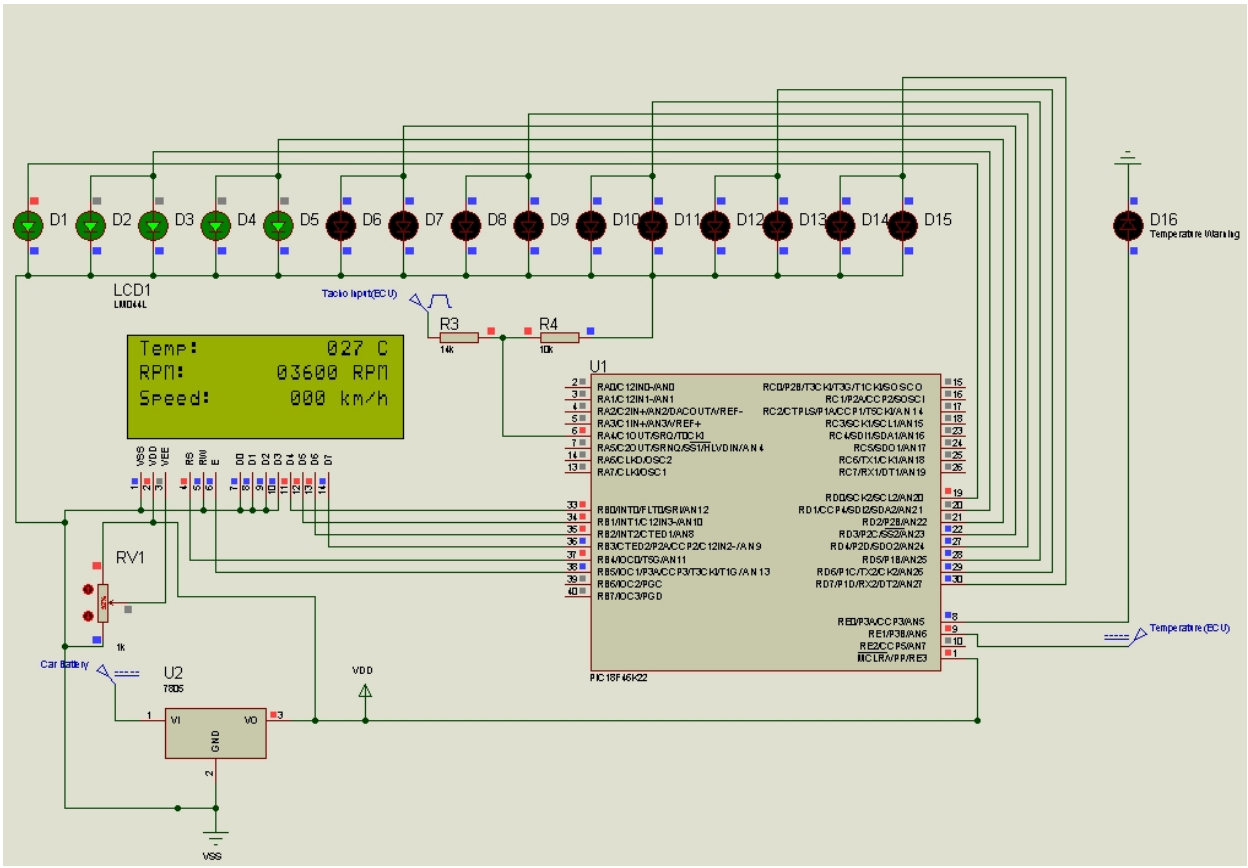
```
void floter(void) {

     temp_res = ADC_Read(14);

 if(temp_res<295){
        PORTA=LED1;
  }

 if (temp_res>295) {
        PORTA=LED2;
        }

 if (temp_res>563) {
        PORTA=LED3;
        }

 if (temp_res>777) {
        PORTA=LED4;
        }

 if (temp_res>803){
        PORTA=LED5;

        }
```

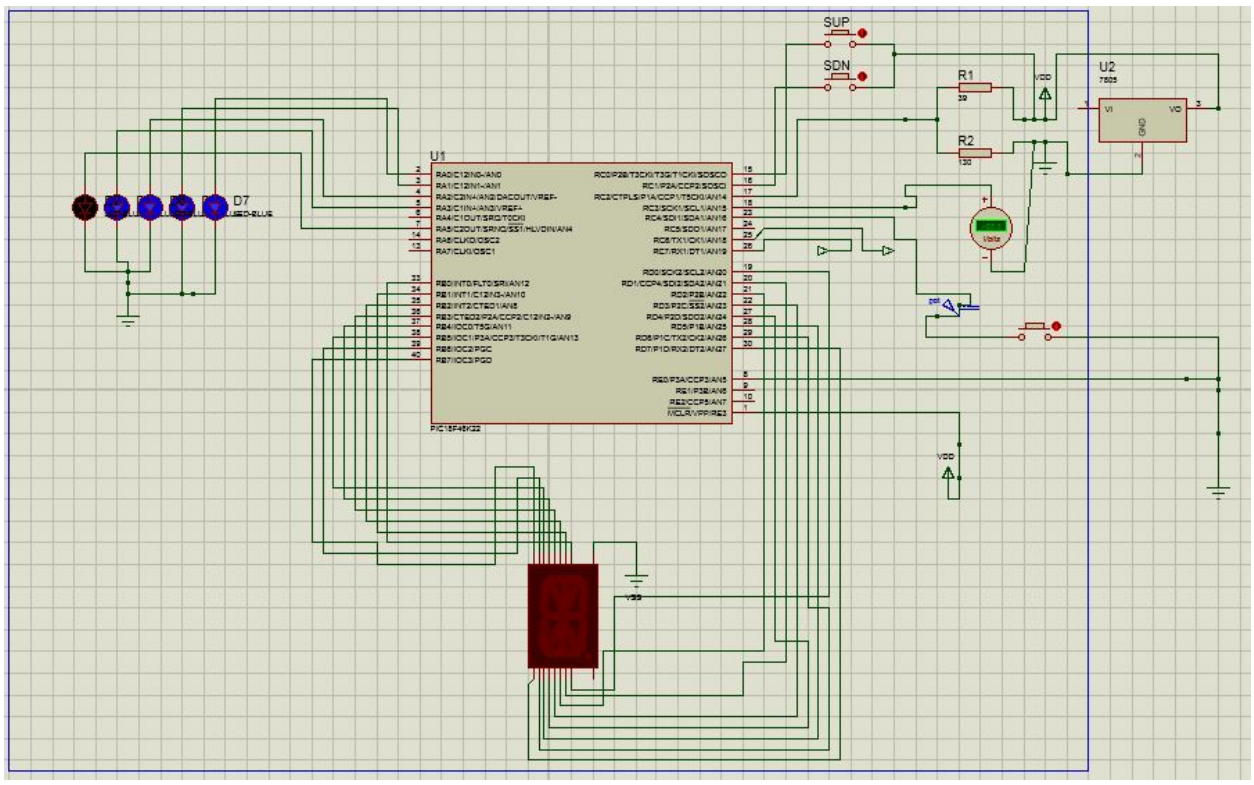
8. Προσομοίωση κυκλωμάτων και κώδικα

Τα κυκλώματα του συστήματος σχεδιάστηκαν στο Proteus πριν κατασκευαστούν. Το συγκεκριμένο πρόγραμμα παρέχει επίσης την δυνατότητα προσομοίωσης τόσο τον κυκλωμάτων όσο και του κώδικα του μικροελεγκτή.

Για την προσομοίωση των σημάτων για τις στροφές του κινητήρα και για τον αισθητήρα Hall χρησιμοποιήθηκαν γεννήτριες παλμών, οι οποίες αλλάζοντας την συχνότητά των παλμών γινόταν προσομοίωση των μεταβλητών. Για τον αισθητήρα θερμοκρασίας μια πηγή τάσης όπου αλλάζοντας την τιμή της άλλαζε η τιμή της LCD οθόνης και γινόταν έλεγχος του αποτελέσματος. Για την προσομοίωση του αισθητήρα στάθμης καυσίμου σχεδιάστηκε ο διαιρέτης τάσης και με κατάλληλη επιλογή αντιστάσεων προσομοιώθηκε η λειτουργία του. Τέλος, για τον υπολογισμό της σχέσης στο κιβώτιο ταχυτήτων τοποθετήθηκαν buttons και το ποτενσιόμετρο και κατά την προσομοίωση του συστήματος αλλάζοντας τις καταστάσεις τους γινόταν έλεγχος για την σωστή λειτουργία του.

Παρακάτω υπάρχουν στιγμιότυπα από την προσομοίωση του συστήματος.


D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
Temperature Warning
LCD1
LM044L
Tacho Input(ECU)
R3
14k
R4
10k
Temp: 027 C
RPM: 03600 RPM
Speed: 000 km/h
U1
PIC18F45K22
RV1
1k
Car Battery
U2
7805
VDD
VSS
Temperature (ECU)

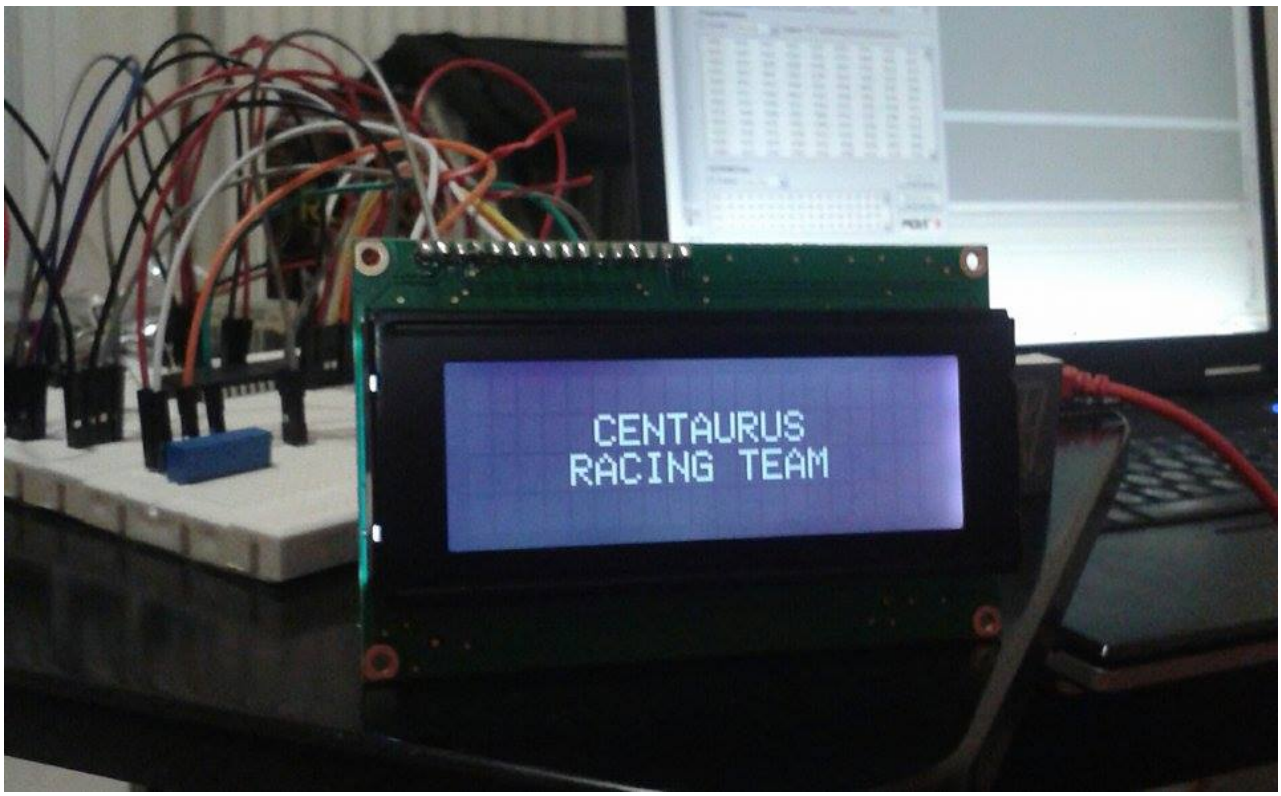

SUP
SDN
R1
R2
U2
7805
U1
PIC18F46K22
D7
LED-BLUE
VDD
VSS

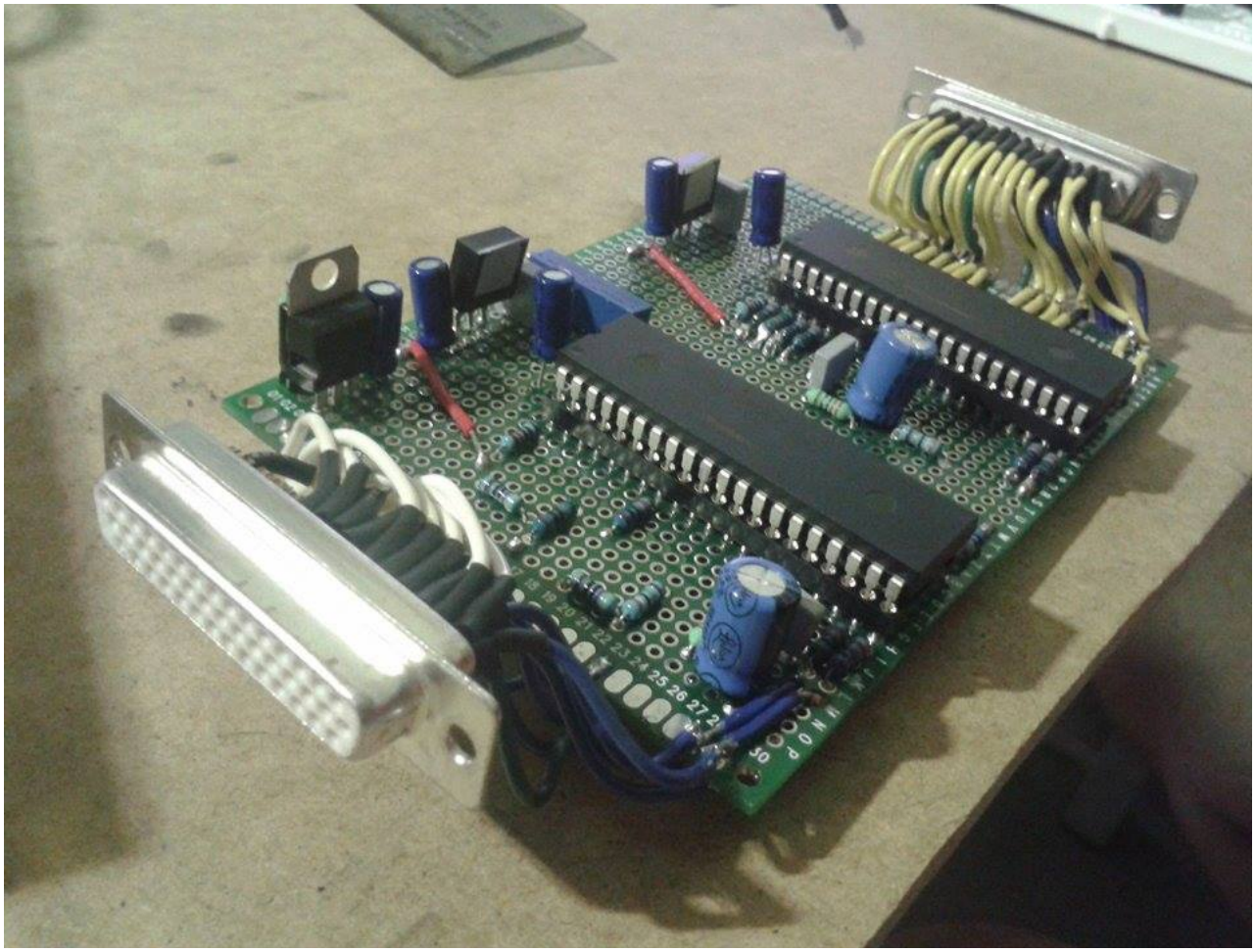
9. Κατασκευή της πλακέτας

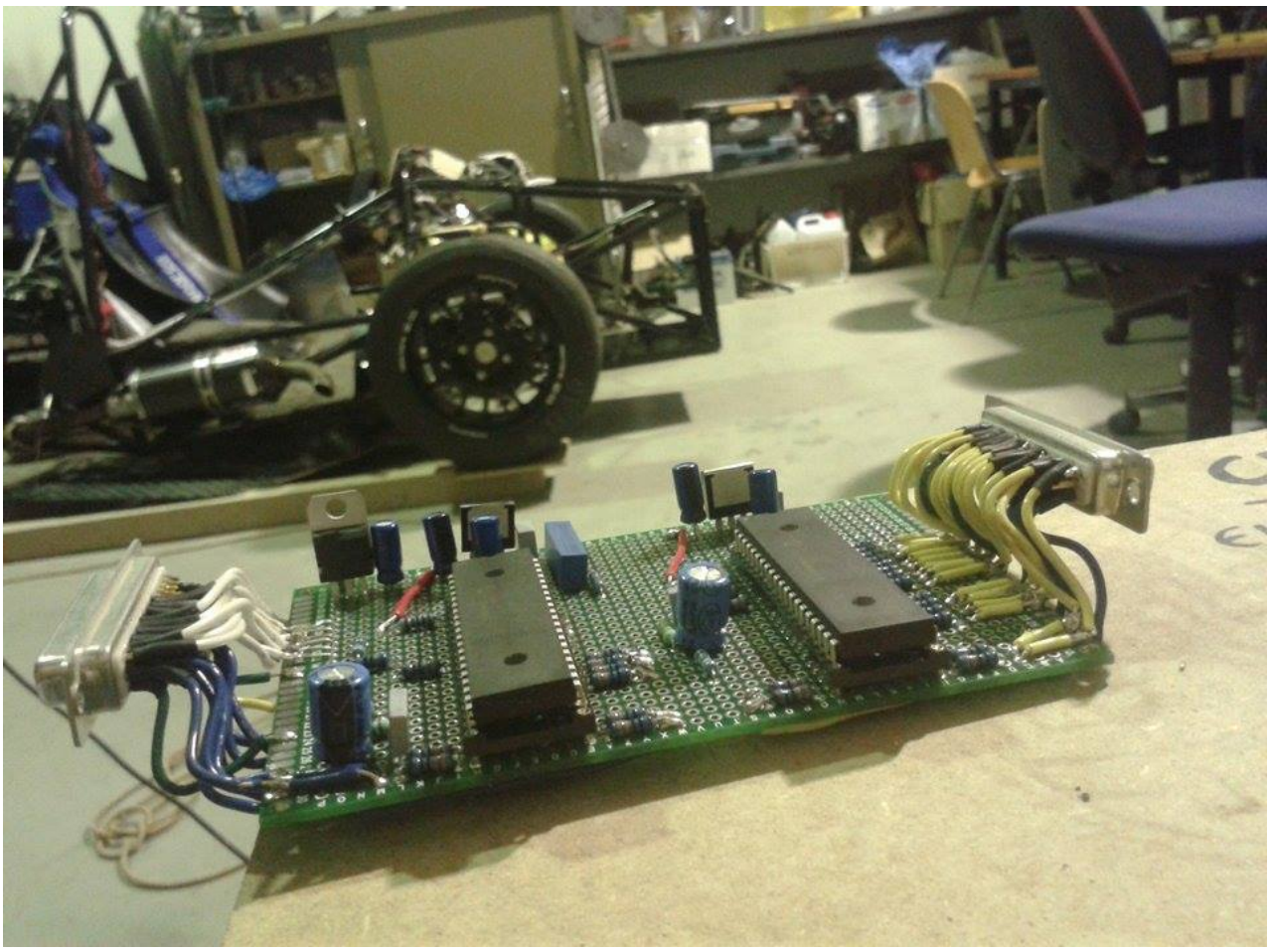
Μετά το πέρας του σχεδιασμού και της προσομοίωσης, όλα τα κυκλώματα δοκιμάστηκαν σε breadboard για την σωστή λειτουργία τους. Έπειτα, προχωρήσαμε στην κατασκευή του σε διάτρητη πλακέτα. Για τις συνδέσεις χρησιμοποιήθηκαν καλώδια 28 AWG διαφορετικών χρωμάτων και οι κολλήσεις έγιναν με κολλητήρι και καλάι.

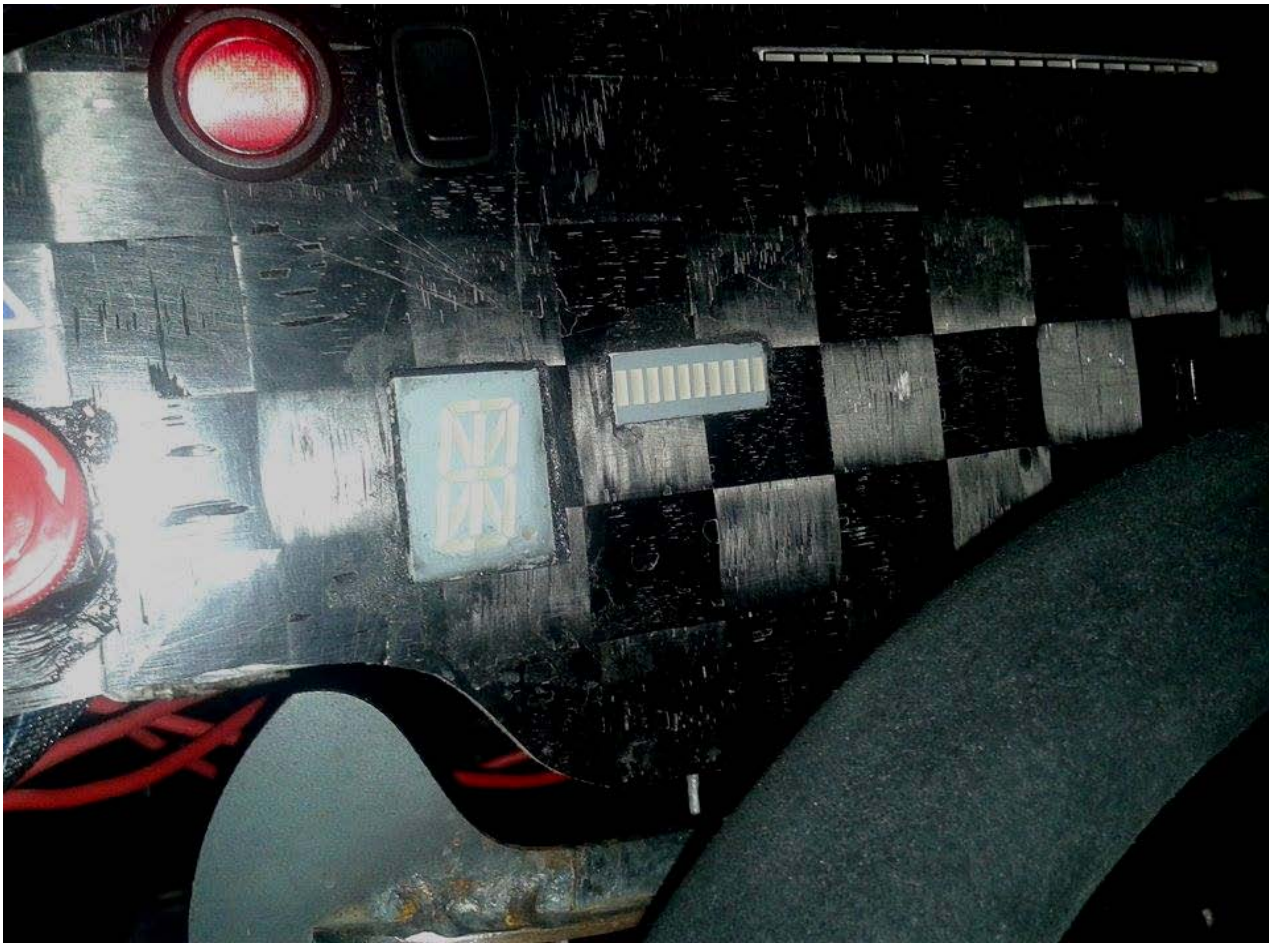
Το κύκλωμα τροφοδοσίας του συστήματος αποτελείται από τρεις σταθεροποιητές τάσης LM7805, οι οποίοι δέχονται σαν είσοδο την τάση της μπαταρίας του μονοθεσίου (12V) και παρέχουν 5V για την τροφοδοσία των δυο μικροελεγκτών και της LCD οθόνης. Οι μικροελεγκτές τοποθετήθηκαν σε ειδικά socket ώστε να μπορούν να αφαιρούνται εύκολα και να προγραμματίζονται. Για τα led χρησιμοποιήθηκαν αντιστάσεις 330Ohm για τον περιορισμό του ρεύματος όπως επίσης και ένα ποτενσιόμετρο 10KOhm για την ρύθμιση της φωτεινότητας της οθόνης. Τέλος, κατασκευάστηκε ένα αλουμινένιο κουτί προστασίας της πλακέτας και τοποθετήθηκαν connectors DB-44 για την είσοδο και την έξοδο των σημάτων.

Ακολουθούν εικόνες από την διαδικασία κατασκευής και την τοποθέτηση του συστήματος στο μονοθέσιο.


CENTAURUS
RACING TEAM








ALY
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΕΠΙΤΡΟΠΗ ΕΡΕΥΝΩΝ
PTC
sparco

10. Προοπτικές βελτίωσης του συστήματος

Το σύστημα που παρουσιάστηκε, όπως κάθε σύστημα, έχει περιθώρια εξέλιξης και βελτίωσης. Κάποιες από αυτές τις βελτιώσεις που θα μπορούσαν να γίνουν είναι οι παρακάτω:

- Τοποθέτηση φίλτρων στην τροφοδοσία του συστήματος για αποφυγή δυσλειτουργιών λόγω θορύβου
- Τοποθέτηση κάρτας SD για καταγραφή όλων των μεταβλητών κάθε χρονική στιγμή για περαιτέρω ανάλυση
- Κατασκευή PCB (printed circuit board) για καλύτερη παρουσίαση του συστήματος

11.Επίλογος

Εν κατακλείδι, η συμμετοχή στην Centaurus Racing Team και η ενασχόληση με το ηλεκτρολογικό και ηλεκτρονικό τμήμα της ομάδας, υπήρξαν έναυσμα για την μελέτη και την κατασκευή του εν λόγω συστήματος. Καθώς η εργασία κινήθηκε στα πλαίσια του Πανεπιστημίου και σε συνεργασία με την ομάδα Centaurus, έγινε προσπάθεια για την επίτευξη υψηλού επιπέδου αποτελέσματος αλλά όσο το δυνατόν χαμηλότερου κόστους κατασκευής. Η υλοποίηση του αποδείχθηκε εξέχουσας σημασίας για την σωστή λειτουργία του μονοθεσίου και την αλληλεπίδρασή του με τον οδηγό, καθώς η προηγούμενη έκδοση του μονοθεσίου δεν διέθετε κάποιο σύστημα πληροφόρησης για τις παραμέτρους του κινητήρα και υστερούσε στην διεπαφή με τον οδηγό. Η γνώση των παραμέτρων αυτών κατά την διάρκεια της οδήγησης βελτίωσε τόσο την τεχνική του οδηγού όσο και τη διαχρονική απόδοση του οχήματος.

Ωστόσο, υπάρχουν ακόμη προοπτικές βελτίωσης όπως αναφέρθηκε εκτενώς παραπάνω και προσδοκούμε στην επίτευξη ενός βέλτιστου δυνατού αποτελέσματος και την εξέλιξη τόσο του συστήματος όσο και των προσωπικών μας γνώσεων, σε συνεργασία πάντα με τους αρμόδιους καθηγητές και τα μέλη της ομάδας.

Βιβλιογραφία

- Microchip PIC18F46K22 datasheet : *http://ww1.microchip.com/downloads/en/DeviceDoc/41412F.pdf*
- Mikroelektronika http://www.mikroe.com/products/view/285/book-pic-microcontrollers-programming-in-c/
- LM 7805 datasheet http://www.tme.eu/en/Document/f1c7df2cf7bbaa88b33acdef6adfcd2e/LM7812.pdf
- Lcd display datasheet http://www.tme.eu/en/Document/1412f1d0dd0505ba30d395b30569dba9/RC2004A-TIW-ESV.pdf
- LED 16-segment display http://www.tme.eu/en/Document/1412f1d0dd0505ba30d395b30569dba9/RC2004A-TIW-ESV.pdf
- Temperature sensor datasheet http://www.pe-ltd.com/joomla/images/downloads/coolant_temp.pdf
- Megasquirt 3 manual http://msextra.com/doc/index.html#ms3
- Embedded-lab.com blog http://embedded-lab.com/