ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ:ΗΛΕΚΤΡΟΝΙΚΟΣ ΠΡΟΣΟΜΟΙΩΤΗΣ ΑΥΤΟΚΙΝΗΤΟΥ

(CAR ELECTRONIC SIMULATOR)

ΤΟΥ ΖΑΒΙΤΣΑΝΟΥ ΑΠΟΣΤΟΛΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΔΡ. ΑΝΑΣΤΑΣΙΟΣ ΔΡΟΣΟΠΟΥΛΟΣ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του πτυχίου

Ιανουάριος 2014

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ 3

ΕΙΣΑΓΩΓΗ 4

ΚΕ**ΦΑΛΑΙΟ** 1-Arduino 5

1.1 Γενικά 5

1.2 Υλικό 5

1.3 Λογισμικό 6

1.4 Κώδικας-Δομη 6

*1.4.*1 Κώδικας για λειτουργιά των στροφών 8

1.4.2 Κώδικας για την λειτουργιά του RPM SHIFTER 10

1.4.3 Λειτουργιά του 7 segment display 11

1.4.4 Κώδικας για την λειτουργιά οδόμετρου (Speed) 12

1.4.5 Κώδικας για την λειτουργιά δείκτη βενζίνης 14

ΚΕΦΑΛΑΙΟ 2 X-SIM SIMULATION 15

2.1 ΕΦΑΡΜΟΓΗ X-SIM 15

2.2 OUTSIM ΚΑΙ CFG FILE 16

2.3 X-SIM CONVERTER 17

2.4 CURRENT GEAR 21

2.5 CURRENT FUEL 22

ΚΕΦΑΛΑΙΟ 3 - **ΗΛΕΚΤΡΟΝΙΚΟ ΜΕΡΟΣ** 24

3.1Λειτουργιά στροφόμετρου 24

3.2 Λειτουργιά οδομέτρου 25

3.3 Λειτουργιά δείκτη βενζίνης 26

3.4Λειτουργιά δείκτη Turbo 26

3.5 ΛΕΙΤΟΥΡΓΙΑ RPM SHIFTER LIGHTS 27

3.6 ΛΕΙΤΟΥΡΓΙΑ 7 SEGMENT DISPLAY 28

3.7 ΛΕΙΤΟΥΡΓΙΑ ΛΑΜΠΤΗΡΩΝ ΚΑΙ ΠΕΡΙΦΕΡΙΑΚΩΝ 29

ΕΠΙΛΟΓΟΣ 31

ΠΕΡΙΛΗΨΗ

Το θέμα της συγκεκριμένης μελέτης βρίσκετε στην άμεση επικοινωνία του υπολογιστή με τον χρηστή (περιβάλλον). Με τα κατάλληλα υλικά και το κατάλληλο software μπορούμε να έχουμε οπτικό αποτέλεσμα σε όποια μορφή εμείς επιθυμούμε. Η συγκεκριμένη μελέτη αφόρα την επικοινωνία αυτή που προανέφερα και ποιο συγκεκριμένα την κατασκευή ηλεκτρονικού προσομοιωτή αυτοκίνητου μέσω υπολογιστή. Η σωστή αλληλουχία προγραμματισμού και ηλεκτρονικών συνδέσεων πετυχαίνει το αποτέλεσμα. Στην μελέτη αυτή χρησιμοποιώ μια πλακέτα η όποια διαβάζει μια μορφή γλώσσας προγραμματισμού πανόμοια με την C (Arduino mega R32) , το x-sim που περιέχει το x-sim converter και το x-sim extractor, το εγκεκριμένο για simulation παιχνίδι live for speed και τα απαραίτητα ηλεκτρονικά στοιχειά για την τροφοδοσία και την μεταφορά παλμών και σήματος. Επίσης υπάρχει σύστημα air-condition και διακόπτες χειρισμού αυτού του συστήματος , διακόπτες για τον χειρισμό των προβολέων του αυτοκίνητου , άμεση λειτουργιά χειρόφρενου e-brake μέσω του παιχνιδιού και του περιβάλλοντος και λειτουργιά των φλας και της ένδειξης alarm. Η κατασκευή αποτελείτε από πραγματικά υλικά αυτοκίνητου (ταμπλό-καντράν-κάθισμα-χειρόφρενο)

ΕΙΣΑΓΩΓΗ

Το αντικείμενο και η ιδέα της συγκεκριμένης πτυχιακής είναι η λειτουργιά ενός κοντέρ αυτοκίνητου συνδεδεμένο με τα κατάλληλα στοιχειά έτσι ώστε να λειτούργει παράλληλα με ένα συγκεκριμένο παιχνίδι εξομοίωσης το ***live for speed***. Το συγκεκριμένο παιχνίδι μου δίνει την δυνατότητα να επέμβω στα αρχεία του και να μπορέσω να κάνω αναγνωρίσιμες στο χρηστή κάποιες εξόδους όπως στροφές και χιλιόμετρα. Το κάθε ένα στοιχειό παρουσίασε διαφορετική συμπεριφορά και διαφορετική συνδεσμολογία. Για την επίτευξη της συγκεκριμένης μελέτης χρειάστηκα κάποια ενδιάμεσα στοιχειά και προγράμματα. Τα στοιχειά που χρησιμοποίησα είναι ένας επεξεργαστής που αναγνωρίζει την εύκολη μορφή της γλώσσας προγραμματισμού C# και ονομάζετε ARDUINO MEGA R32. Το επόμενο απαραίτητο στοιχειό είναι ένα πρόγραμμα προσομοίωσης εξομοιωτών που συνεργάζεται με τον επεξεργαστή ανταλλάσοντας πληροφορίες και ονομάζετε X-SIM SIMULATION και τέλος το ηλεκτρονικό κομμάτι που αποτελείτε από ολοκληρωμένα τσιπάκια των 8 bit που βοηθούν στην οδήγηση κάθε μοτέρ που περιέχει το κοντέρ.

ΚΕΦΑΛΑΙΟ 1-ARDUINO

1.1 ΓΕΝΙΚΑ

Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες· το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

1.2 ΥΛΙΚΟ

Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό αντηχητή σε κάποιες παραλλαγές). Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Η πλακέτα του Arduino έχει εκτεθειμένες τις περισσότερες επαφές εισόδου/εξόδου για χρήση με άλλα κυκλώματα. Το Diecimila, για παράδειγμα, παρέχει 14 ψηφιακές επαφές εισόδου/εξόδου, από τις οποίες οι 6 μπορούν να παράξουν σήματα PWM (παλμούς), και 6 αναλογικές εισόδους. Αυτές οι επαφές είναι διαθέσιμες στην κορυφή της πλακέτας μέσω θηλυκών συνδέσεων μεγέθους 0,1 ιντσών. Διάφορες plug-in πλακέτες εφαρμογών γνωστές σαν “shields” είναι, επίσης, διαθέσιμες στο εμπόριο. Οι συμβατές με το Arduino πλακέτες Barebones και Boarduino διαθέτουν αρσενικές επαφές στην κάτω πλευρά της πλακέτας για να μπορούν να συνδεθούν με πλακέτες που δεν χρειάζονται συγκολλήσεις

1.3 Λογισμικο

Το IDE του Arduino είναι γραμμένο σε Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστής και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα. Το περιβάλλον ανάπτυξης είναι βασισμένο στην Processing, ένα περιβάλλον ανάπτυξης σχεδιασμένο να εισαγάγει στον προγραμματισμό μη εξοικειωμένους χρήστες με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring, μια γλώσσα που μοιάζει με την C η οποία παρέχει παρόμοια λειτουργικότητα για μια πιο περιορισμένης σχεδίασης πλακέτα, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στην Processing.

1.4 Κωδικας-Δομη

Δήλωση μεταβλητών

Int rpm; (δήλωση αρχικής τιμής στροφών)

Int Speed; (δήλωση αρχικής τιμής χιλιόμετρων)

Int fuel; (δήλωση αρχικής τιμής βενζίνης)

Char kind_of_data; (ανάγνωση των δεδομένων)

Δήλωση αρχικών και τελικών τιμών

int fuel_LOW=110;

int fuel_HIGH=255;

int speed_LOW=0;

int speed_HIGH=230;

Int rpm_LOW=10;

Int rpm_HIGH=255;

*LOW-**à** αρχική τιμή*

HIGH- **à** *τελική τιμή*

***Δήλωση Pin εξόδων από** Arduino*

```
Void   setup () {
 Serial. Begin (115200);
  PinMode (3, OUTPUT);
  PinMode (5, OUTPUT);
  PinMode (6, OUTPUT);
  PinMode (11, OUTPUT);
  PinMode (36, OUTPUT);
  PinMode (37, OUTPUT);
  PinMode (38, OUTPUT);
  PinMode (39, OUTPUT);
  //calibration();
}
```

Τιμή σειριακής εκκίνησης Από X-Sim

επαναλαμβανομένη λούπα

Αναγνώριση του είδους των δεδομένων

```
Void loop ()
{
  While (Serial. Available () > 0)
  {
    kind_of_data = Serial. Read ();
    if (kind_of_data == 'R' ) Read_Rpm();
    if (kind_of_data == 'S' ) Read_Speed();
```

έλεγχος για θετικό σειριακό αριθμό

επαναλαμβανομένη λούπα

Είδος δεδομένων= με τον σειριακό αριθμό

Αναγνώριση από X-sim converter

```
  if(kind_of_data == 'F' ) Read_fuel();    }
```

1.4.1 Κωδικας για λειτουργια των στροφων

```
void  Read_Rpm(){
 delay(1);
  int Rpm100 = Serial.read()- '0';

  delay(1);

 int Rpm10 = Serial.read()- '0';

delay(1);

 int Rpm1 = Serial.read()- '0';

 int rpm = 100*Rpm100 + 10*Rpm10 + Rpm1;
```

Είναι μια επαναλαμβανομένη λούπα που φτιάχνει τιμές σε μορφή χιλιάδας εκατοντάδας και δεκάδας. Εσωτερικά διαβάζετε σε 8 bit. **Ποιο συγκεκριμένα είναι ένα κομμάτι κώδικα για την συριακή επικοινωνία της** UBS Serial comport. **Η τελική τιμή χρειάζεται για τον υπολογισμό του** map **στο** set Air core

setAircore(map(rpm,127,255,63,245)); ⟶ ανάλογα με τις τιμές που θα δώσει το map γίνονται ο πράξεις με όρια αναγνώρισης του x-sim και της arduino 127-255 και δυο ενδιάμεσες τιμές.

Η εντολή: setAircore(map(rpm,127,255,63,245)); μας παραπέμπει

εδώ :

```
void setAircore(float pos){
float sinCoilValue = 255*sin(pos/35);
 float cosCoilValue = 255*cos(pos/35);
 if (sinCoilValue<=0) {
  digitalWrite(41, LOW);
  digitalWrite(40, HIGH);
 }
 else {
  digitalWrite(41, HIGH);
  digitalWrite(40, LOW);
 }
 if (cosCoilValue<=0) {
  digitalWrite(42, LOW);
  digitalWrite(43, HIGH);
 else {
  digitalWrite(42, HIGH);
  digitalWrite(43, LOW);
 }
 sinCoilValue = abs(sinCoilValue);
```

Για τις στροφές έχουμε ένα μοτέρ air core **με** 4 pins . **Για την οδήγηση του χρειαζόμαστε κάποιες τιμές συνημίτονου και ημιτόνου . Κάνοντας τις πράξεις και βάζοντας και τις κατάλληλες μοίρες στην διαίρεση έχω μια τιμή για το** sinCoilValue **και το** cosCoilValue

σε έναν βρόγχο ελέγχω τις τιμές των cosCoilValue sin Coil Value. **Η διαδικασία αύτη γίνετε χρησιμοποιώντας** 4 **ψηφιακές εξόδους**

Έχοντας τις τιμές των sin Coil Value **και** cosCoilValue **τις βάζουμε σε ισότητα πολλαπλασιάζοντας με την εντολή** abs **που ανήκει στην κατηγορία των μαθηματικών εντολών του** arduino **και υπολογίζει την απολυτή τιμή**

```
cosCoilValue = abs(cosCoilValue);
analog Write(8, sinCoilValue);
analog Write(7, cosCoilValue);
```

Τέλος έχοντας τις απόλυτες τιμές με την εντολή analog write **τις δίνουμε στην** arduino **μέσα από** 2 pin **παλμών**

1.4.2 ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ RPM SHIFTER

```
if (rpm>130) digitalWrite(22,HIGH);
if (rpm<130) digitalWrite(22,LOW);
if (rpm>150) digitalWrite(23,HIGH);
if (rpm<150) digitalWrite(23,LOW);
if (rpm>160) digitalWrite(24,HIGH);
if (rpm<160) digitalWrite(24,LOW);
if (rpm>170) digitalWrite(25,HIGH);
if (rpm<170) digitalWrite(25,LOW);
if (rpm>180) digitalWrite(26,HIGH);
if (rpm<180) digitalWrite(26,LOW);
if (rpm>190) digitalWrite(27,HIGH);
if (rpm<190) digitalWrite(27,LOW);
if (rpm>200) digitalWrite(28,HIGH);
if (rpm<200) digitalWrite(28,LOW);
if (rpm>210) digitalWrite(29,HIGH);
if (rpm<210) digitalWrite(29,LOW) ;
if (rpm>220) digitalWrite(30,HIGH);
if (rpm<220) digitalWrite(30,LOW);
```

Οι αριθμοί από 22 έως 28 είναι τα digital pins της arduino

→ εντολές για την λειτουργιά του rpm shifter

Οι τιμές από το 130 **έως το** 200 **είναι αντιπροσωπευτικές ανάλογα με το κοντέρ που που διαθέτω και τις ρυθμίσεις στο** x-sim high και low ενεργοποίηση ψηφιακής εξόδου όπως **μηδέν** η **ένα**

```
if (rpm>225) digitalWrite(31,HIGH);
```

1.4.3 ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ 7 SEGMENT DISPLAY

```
void  Read_Gear(){
 delay(2);
 int Gear100 = Serial.read()- '0';
 delay(2);
 int Gear10 = Serial.read()- '0';
 delay(2);
 int Gear1 = Serial.read()- '0';

 gear = 100* Gear100 + 10*Gear10 +  Gear1;

 gear = map(gear,127,255,-1,7);

  if ( gear == -1 ){
  //write '0'
 digitalWrite(32, 0);
 digitalWrite(33, 1);
 digitalWrite(34, 1);
 digitalWrite(35, 1);
```

Στην λούπα αύτη φτιάχνουμε το display (gear indication)

Χρησιμοποιώ 7 **ψηφιακές εξόδους κάθε φορά και για** 7 **φορές όσες και ο αριθμός των ταχυτήτων**

Σε κάθε λούπα ανάλογα με τον αριθμό που δίνω 1 **η** 0 **ενεργοποιώ η απενεργοποιώ τις ψηφιακές εξόδους φτιάχνοντας στο** display.

Κάθε παύλα του display **είναι μια ψηφιακή έξοδος**

Έχουμε την αρχική τιμή του gear **η όποια μαζί με το** map **μας δίνουν την τελική τιμή του**.

εντολες για την εμφανιση της οπισθεν

```
digitalWrite(36, 1);

digitalWrite(37, 1);

digitalWrite(38, 0);
```

Για την εμφάνιση των υπολοίπων ταχυτήτων ακολουθειτέ η ιδία διαδικασία ενεργοποιώντας κάθε φορά το σωστό Pin έτσι ώστε να δοθεί η τάση των +5 volt σε κάθε μια απ τις παύλες του display οπού και αν είναι απαραίτητο .

1.4.4 ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΛΕΙΤΟΥΡΓΙΑ ΟΔΟΜΕΤΡΟΥ (SPEED)

```
void  Read_Speed(){

 delay(1);

 int Speed100 = Serial.read()- '0';

 delay(1);

 int Speed10 = Serial.read()- '0';

 delay(1);

 int Speed1 = Serial.read()- '0';


 int Speed = 100* Speed100 + 10* Speed10 +  Speed1;


 setAircore(map( Speed,127,255,0,245));
```

Όπως και το στροφόμετρο έτσι και το οδόμετρο είναι ένα air core μοτέρ με 4 pin. Ακλουθώντας την ιδία διαδικασία δίνουμε μια αρχική τιμή στο Speed δηλώνοντας το η όποια τιμή μπαίνει στην εντολή setAircore δίνοντας μια τελική τιμη στο map. Γράφοντας τώρα την εντολή setAircore ο κώδικας μας στέλνει σε ένα άλλο κομμάτι προγραμματισμού το οποίο συναντήσαμε και για το στροφόμετρο

Για το οδόμετρο έχουμε ένα μοτέρ air core **με** 4 pins . **Για την οδήγηση του χρειαζόμαστε κάποιες τιμές συνημίτονου και ημιτόνου . Κάνοντας τις πράξεις και βάζοντας και τις κατάλληλες μοίρες στην διαίρεση έχω μια τιμή για το** sinCoilValue **και το** cosCoilValue

```
void setAircoreSp(float pos){
float sinCoilValue = 255*sin(pos/35);
float cosCoilValue = 255*cos(pos/35);


if (sinCoilValue<=0) {
  digitalWrite(45, LOW);
  digitalWrite(44, HIGH);
 }
 else {
  digitalWrite(45, HIGH);
  digitalWrite(44, LOW);
 }
 if (cosCoilValue<=0) {
  digitalWrite(46, LOW);
  digitalWrite(47, HIGH);
 }
 else {
  digitalWrite(46, HIGH);
  digitalWrite(47, LOW);
 }
 sinCoilValue = abs(sinCoilValue);
 cosCoilValue = abs(cosCoilValue);
```

σε έναν βρόγχο ελέγχω τις τιμές των cosCoilValue sin Coil Value. **Η διαδικασία αύτη γίνετε χρησιμοποιώντας** 4 **ψηφιακές εξόδους διαφορετικές αύτη τη φορά**

Έχοντας τις τιμές των sin Coil Value **και** cosCoilValue **τις βάζουμε σε ισότητα πολλαπλασιάζοντας με την εντολή** abs **που ανήκει στην κατηγορία των μαθηματικών εντολών του** arduino **και υπολογίζει την απολυτή τιμή**

Τέλος έχοντας τις απόλυτες τιμές με την εντολή analog write **τις δίνουμε στην** arduino **μέσα από** 2 pin **παλμών**

```
  analog Write(8, sinCoilValue);
  analog Write(7, cosCoilValue);
}
```

1.4.5 ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΛΕΙΤΟΥΡΓΙΑ ΔΕΙΚΤΗ ΒΕΝΖΙΝΗΣ

Σε αυτό το κομμάτι προγραμματισμού παρουσιάζονται διαφορές σε σχέση με το οδόμετρο και το στροφόμετρο. Το μοτέρ της βενζίνης έχει 2 pin Ο παρακάτω κώδικας είναι για την οδήγηση του συγκεκριμένου μοτέρ.

```
Void Read_Fuel(){
  delay(1);
  int Fuel100 = Serial.read()- '0';
  delay(1);
  int Fuel10 = Serial. Read()- '0';
  delay(1);
  int Fuel1 = Serial. Read()- '0';

  fuel = 100*Fuel100 + 10*Fuel10 + Fuel1;
analog Write(10,map(fuel,127,255,0,10));
```

Έχοντας τη τιμή του fuel , **υπολογίζουμε την τελική τιμή του** map **και την στέλνουμε στο** pin 10 **του** arduino

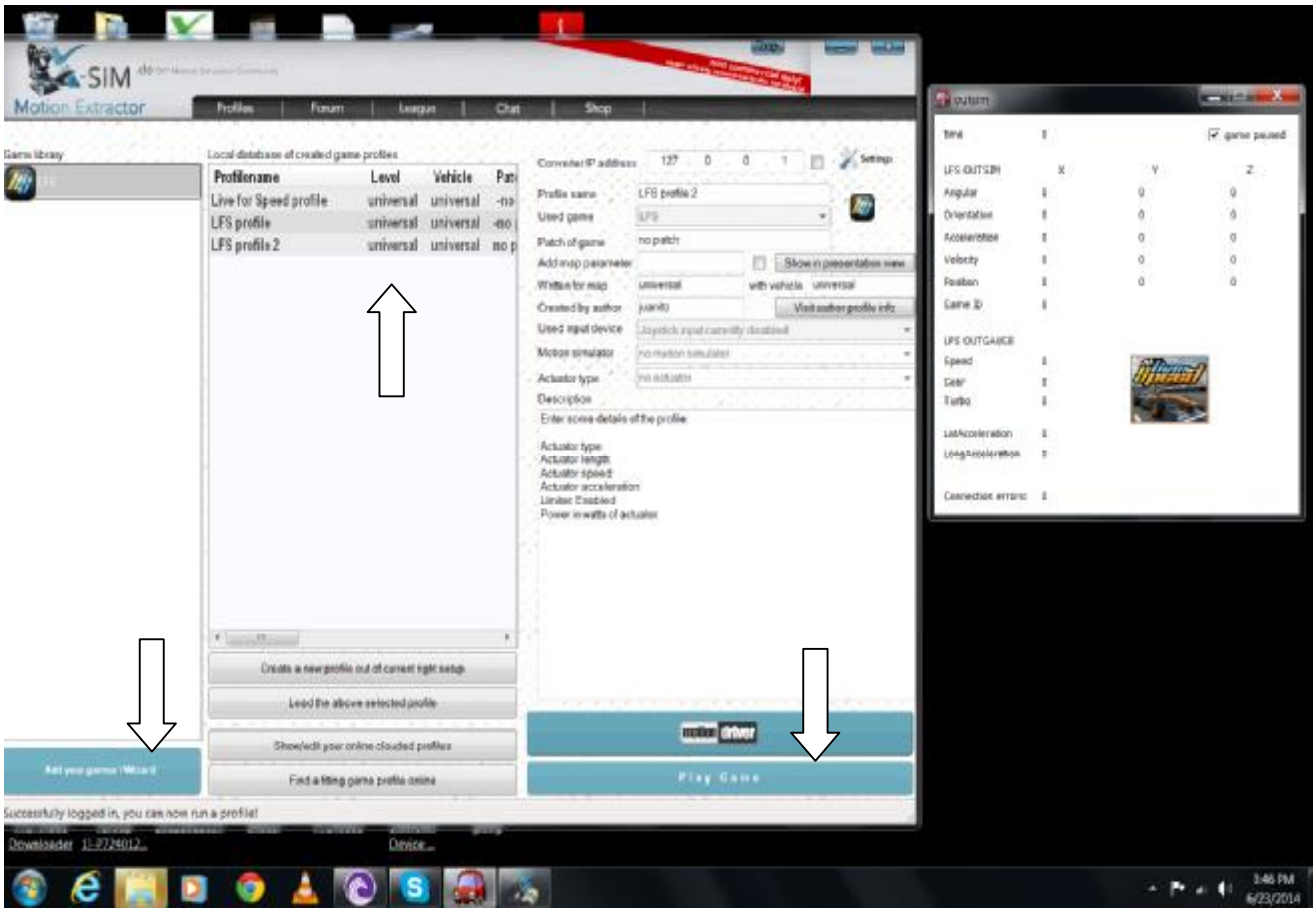
Το κοντέρ περιέχει και άλλο ένα μοτέρ αυτό της θερμοκρασίας. Επειδή το outsim και το παιχνίδι προσομοίωσης δεν μας δίνει στο περιβάλλον έξοδο , δεδομένα για τις τιμές της θερμοκρασίας που τρέχουν το χρησιμοποιώ σαν turbo timer δεδομένο ότι έχω έξοδο για αύτη τη χρήση. Η διαδικασία του προγραμματισμού είναι ακριβώς η ιδία με αύτη της βενζίνης με την μόνη διάφορα ότι το αποτέλεσμα του map που υπολογίζουμε απ τις εντολές το στέλνουμε σε άλλο pwn pin. (pin παλμού)

ΚΕΦΑΛΑΙΟ 2 X-SIM SIMULATION

2.1 ΕΦΑΡΜΟΓΗ X-SIM

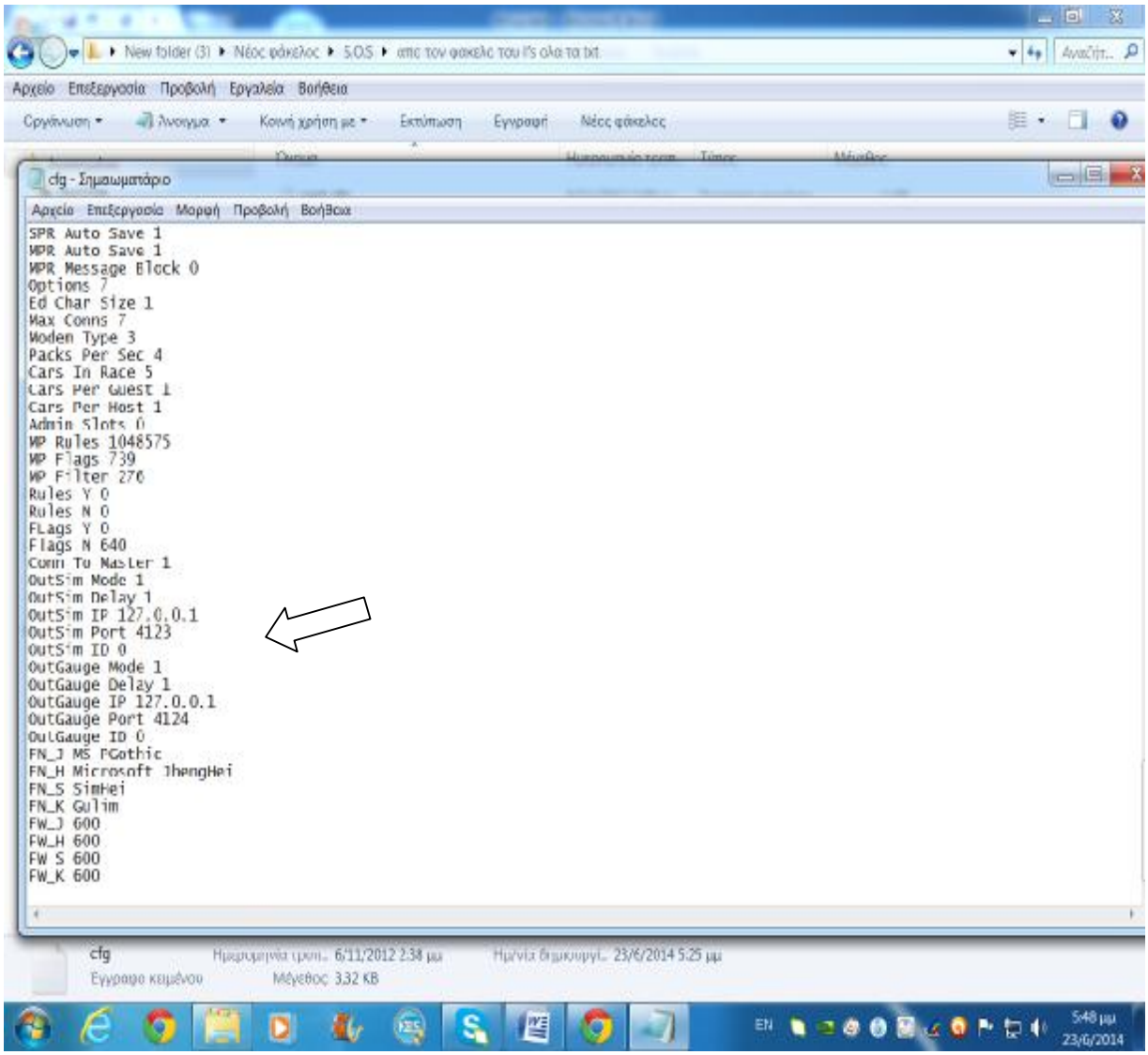
Το x-sim αρχικά είναι ο δαυλός επικοινωνίας μεταξύ του παιχνιδιού προσομοίωσης και του κώδικα προσομοίωσης που έχουμε κατεβάσει στον επεξεργαστή της Arduino mega R32. Το x-sim είναι μια εφαρμογή που λειτούργει online και χωρίζετε σε υποεφαρμογες. Η πρώτη υποεφαρμογη είναι η εφαρμογή exe που συνδυάζει τις εξόδους και στέλνει δεδομένα στο x-sim converter που θα αναφέρω ποιο κάτω τα όποια χρειάζονται ώστε να πραγματοποιηθεί η πλήρης λειτουργιά του ηλεκτρονικού κοντέρ και ονομάζετε x-sim extractor. Το δεύτερο μέρος είναι άλλη μια εφαρμογή exe που συνδυάζει τις εξόδους διαβάζοντας ‘τες σαν άξονες και ονομάζετε x-sim converter . Ποιο συγκεκριμένα το x-sim τραβάει τα δεδομένα από το παιχνίδι προσομοίωσης τα μεταφράζει έτσι ώστε να τα πάρει ο επεξεργαστής της Arduino mega ο όποιος με την σειρά του θα στείλει ανάλογα με τις εντολές που έχει μέσα τις κατάλληλες τάσεις τα κατάλληλα σήματα και τους κατάλληλους παλμούς έτσι ώστε να λειτουργήσει το κοντέρ.

x-sim extractor



2.2 OUTSIM KAI CFG FILE

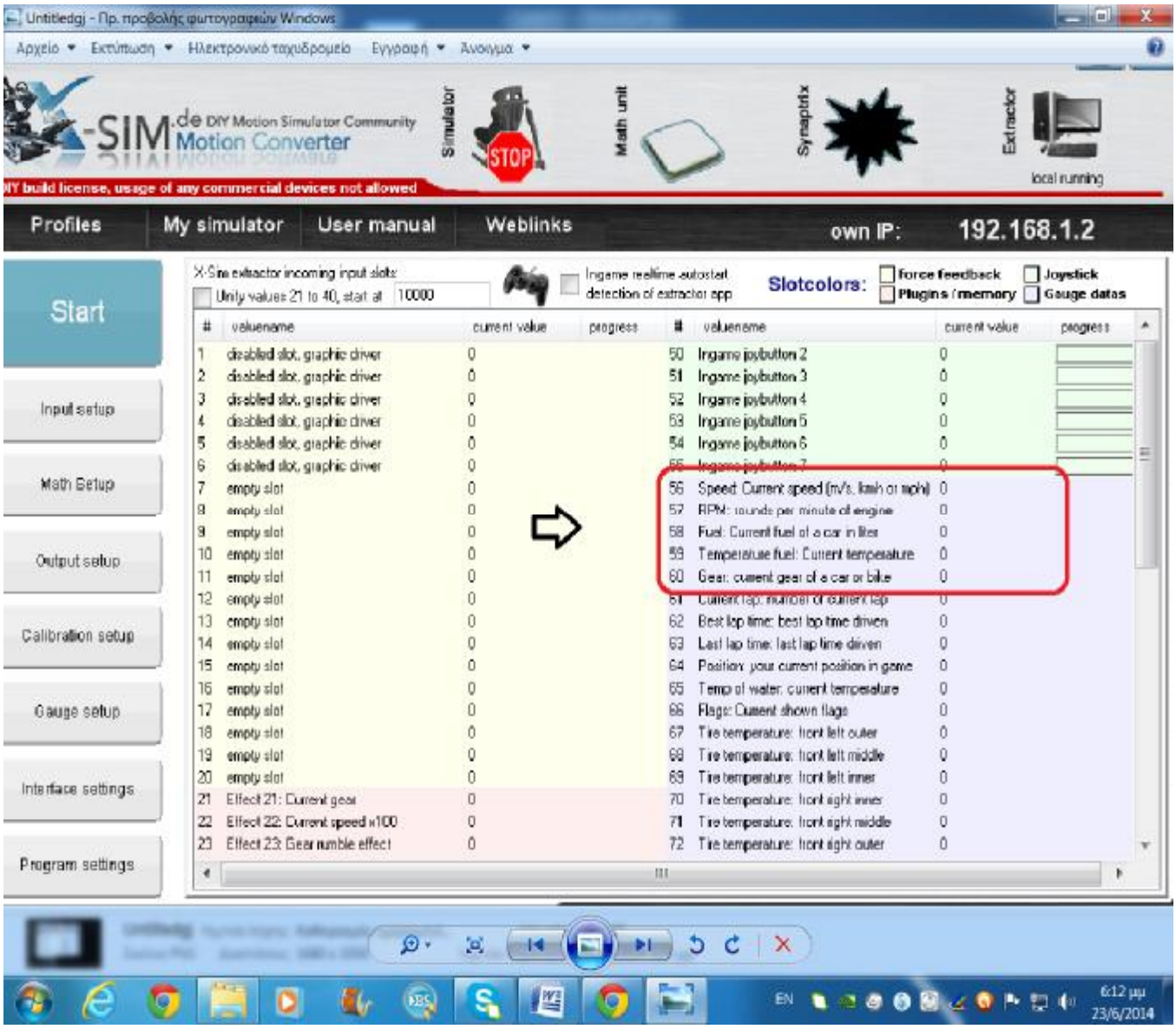
Πριν μεταβούμε στο x-sim converter θα πρέπει να επέμβουμε στα αρχεία του παιχνιδιού προσομοίωσης και να αλλάξουμε τις τιμές των εξόδων out gauge. Με αυτό επιτυγχάνουμε τις πληροφορίες που θέλουμε, να τις εισάγουμε στο x-sim converter και x-sim extractor δηλαδή να είναι οπτικά στο χρηστή εμφανίσιμες. Στον φάκελο του live for speed βρίσκουμε ένα .txt αρχείο(cfg file) με κάποια δεδομένα και εμείς αλλάζουμε τις τιμές των Out gauges όπως φαίνετε παρακάτω.



Αλλάζοντας αυτές τις τιμές επιτρέπουμε στο παιχνίδι προσομοίωσης να μας δώσει τις εξόδους που χρειάζεται και η arduino και το x-sim. Μετά την αλλαγή αύτη με το άνοιγμα του x-sim extractor θα ανοίγει και ένα plug-in εξόδων (outsim) το οποίο περιέχει τις εξόδους.

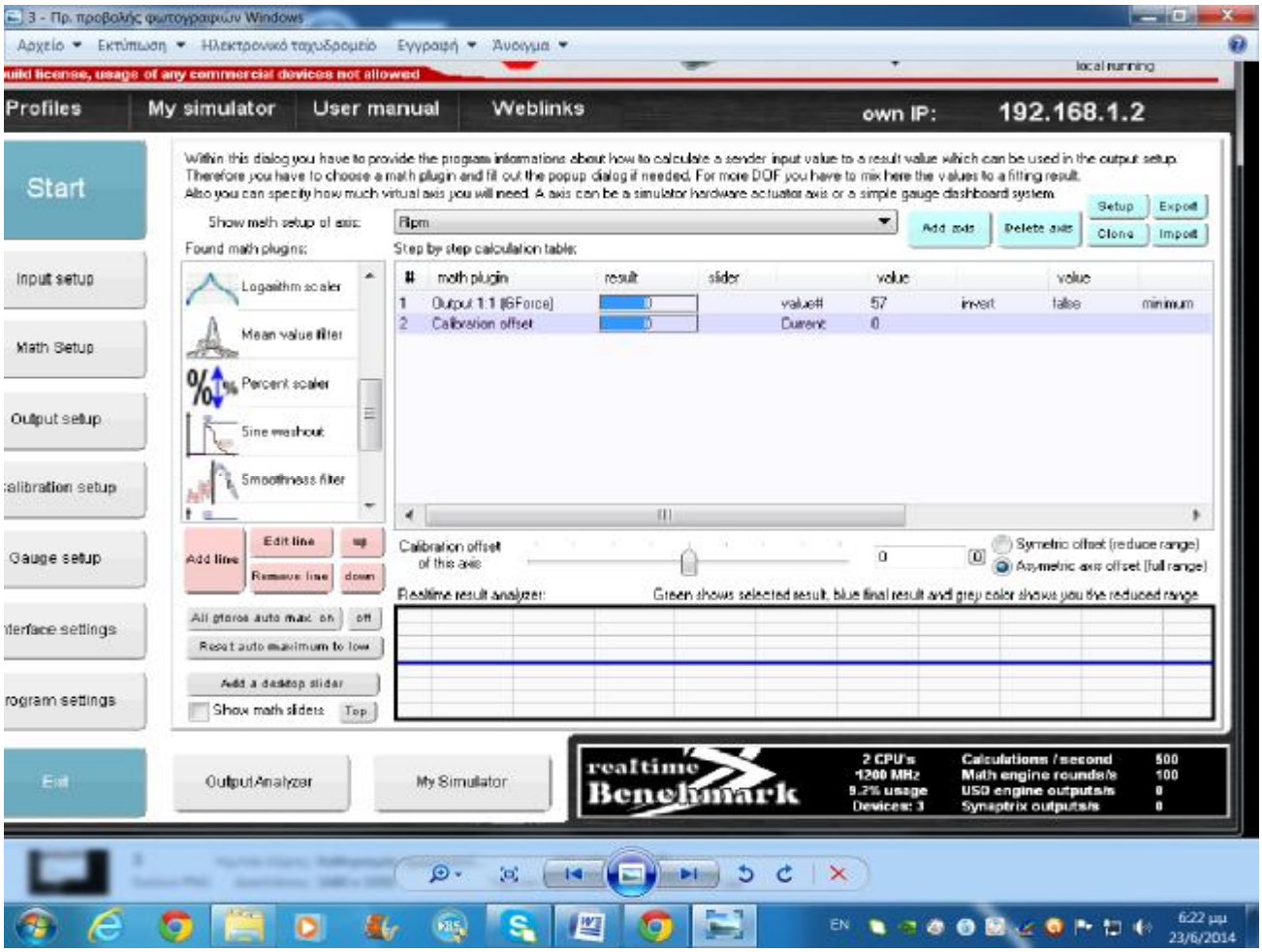
2.3 X-SIM CONVERTER

Έχοντας αυτές τις τιμές ανοίγουμε το x-sim converter το οποίο περιέχει 7 καρτέλες. Στην πρώτη καρτέλα (input setup) βλέπουμε πως το x-sim converter αντιλαμβάνεται τις εξόδου που πήραμε.



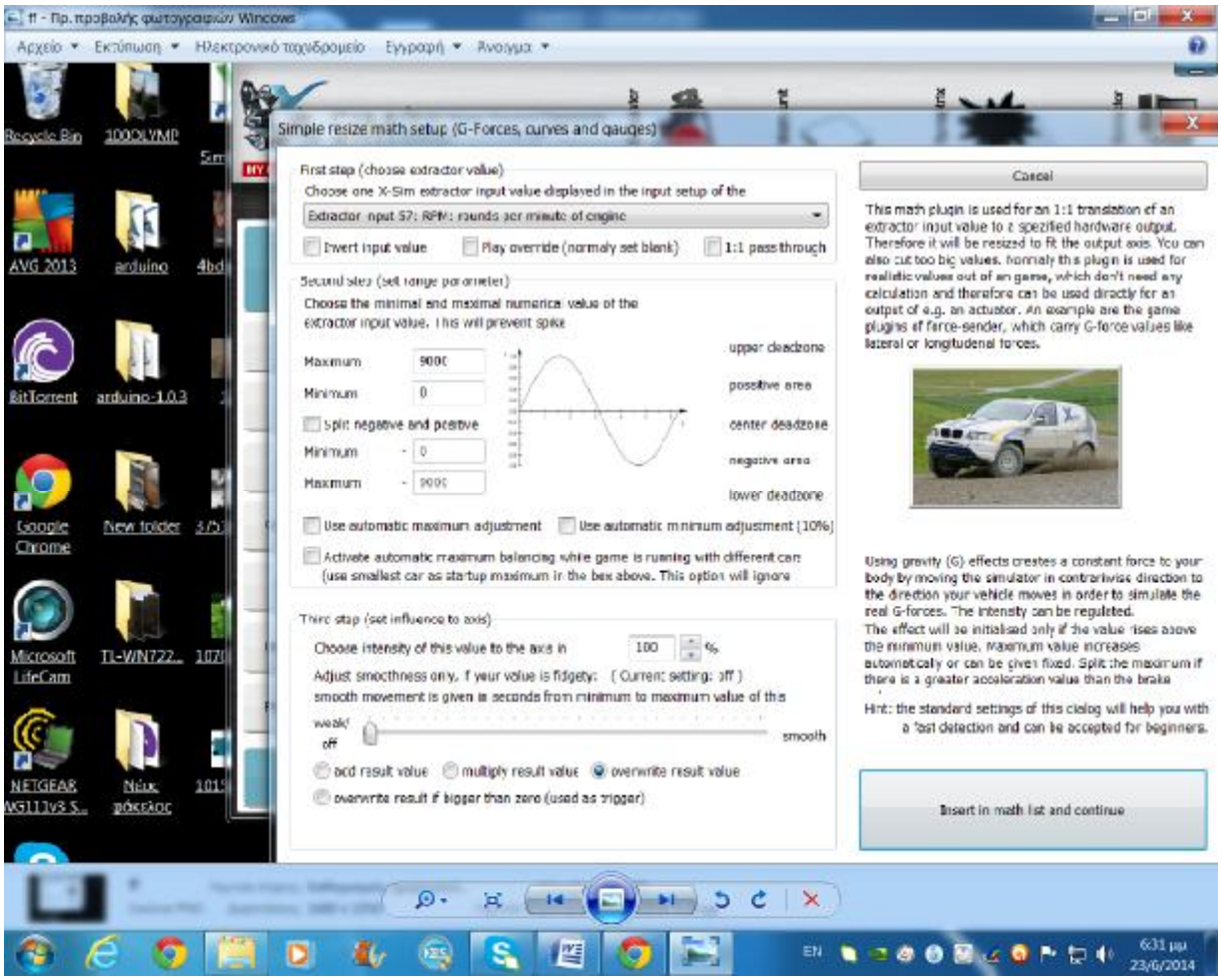
Παρατηρούμε τις εξοδους οι όποιες με το start θα πάρουν τιμές ανάλογες.

Στην επομένη καρτέλα math setup κάθε μια απ τις εξοδους διαβάζετε σαν άξονας όπως προείπα (axis) . Πατάμε add axis και φτιάχνουμε μια εσωτερική καρτέλα για κάθε έξοδο. Στο ορθογώνιο μονόμετρε (real-time result analyzer) βλέπουμε αφού πατήσουμε στο start πως συμπεριφέρεται κάθε μια απ τις εξόδους σε χρόνο 33ms



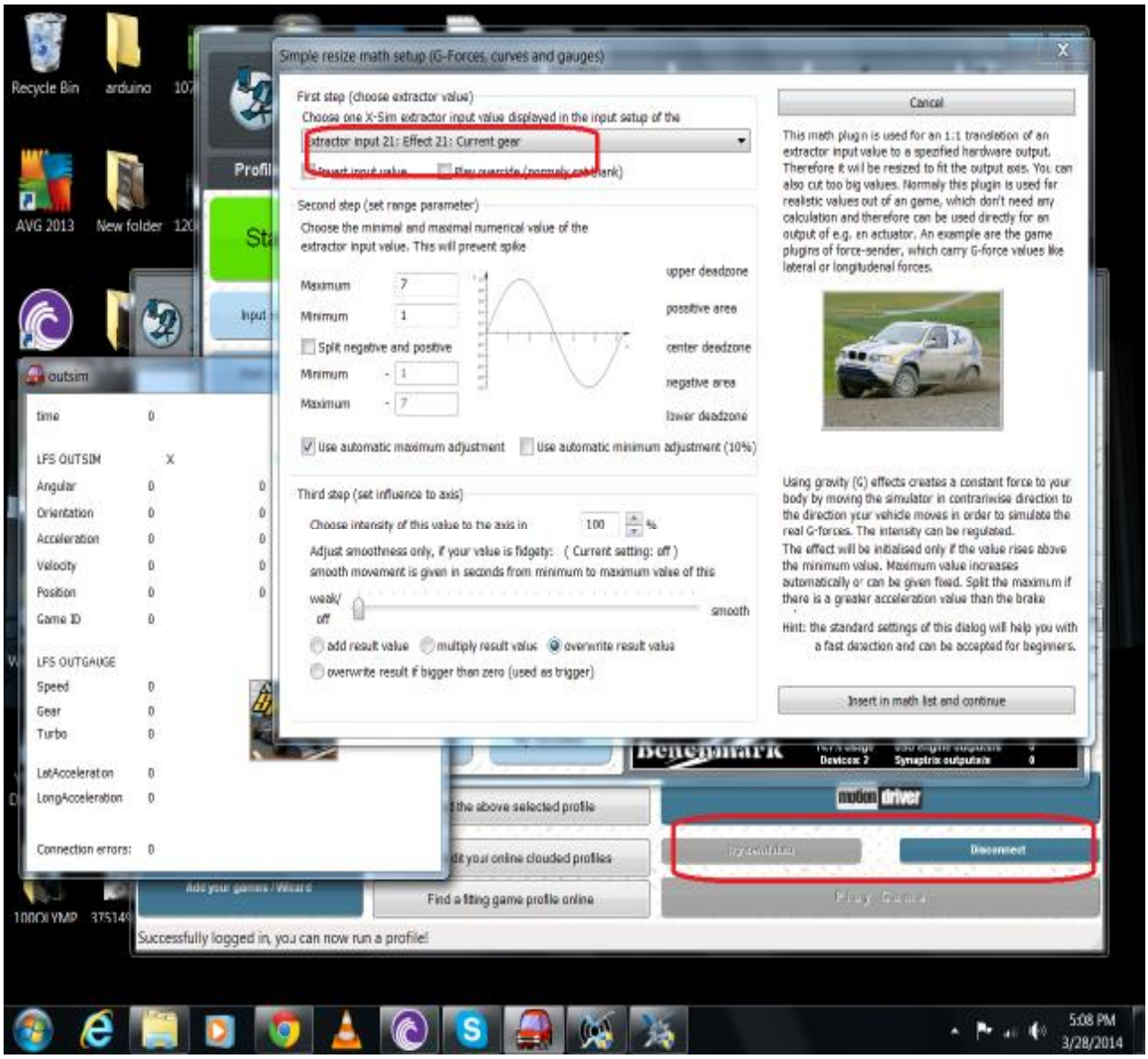
Έπειτα στα αριστερά επιλεγούμε math plug-in σε κάθε απ την μια έξοδο. Το plug-in είναι ίδιο για όλες τις εξόδους και είναι το Output 1:1. Έπεισες μπορούμε να δούμε στην μπάρα του plug-in την τιμή του value η όποια είναι το 57 όπως και στην πρώτη καρτέλα (input setup) Η συγκεκριμένη περίπτωση είναι για την έξοδο των στροφών (RPM)

Κάνοντας διπλό κλικ πάνω στο plug-in έχουμε



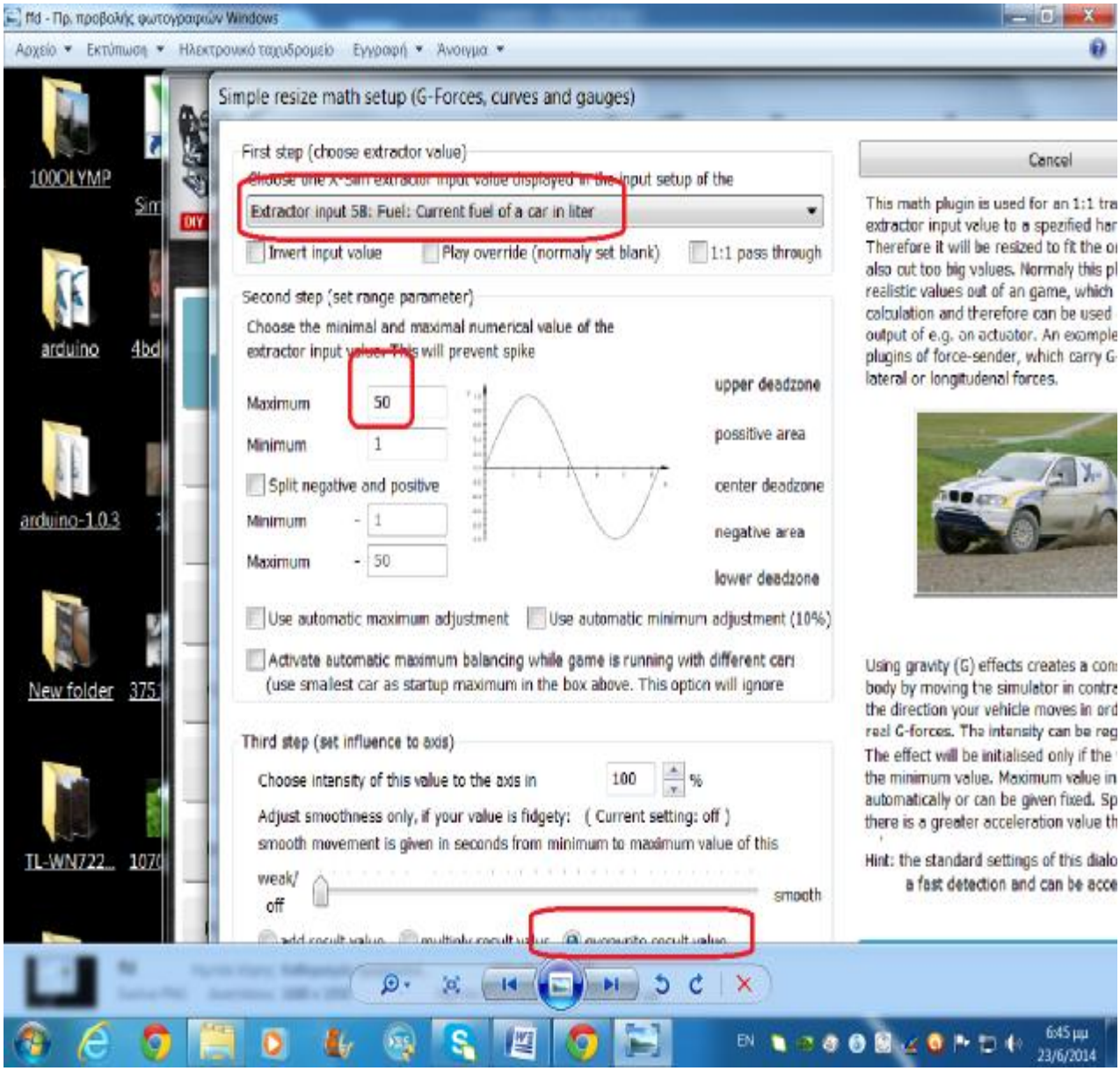
Πατάμε στο βελάκι και επιλεγούμε input value ανάλογα με την έξοδο. Η συγκεκριμένη περίπτωση είναι για extractor input 57 (rpm) . Στη συνεχεία δίνουμε μεγίστη τιμή 9000 διότι έχουμε 9000 στροφές. Βέβαια αυτό είναι εικονικό, μπορούμε να επιλέξουμε το use automatic maximum adjustment δίνοντας το x-sim μόνο του την μεγίστη τιμή. Στην συγκεκριμένη περίπτωση 9000 rpm είναι ο (κόφτης) του στροφόμετρου. Την ιδία διαδικασία κάνουμε και στις υπόλοιπες εξοδους βρίσκοντας πάντα το σωστό input value για κάθε περίπτωση.

2.4 CURRENT GEAR



η περίπτωση του gear indication το οποίο μας δείχνει σε display τον αριθμό της ταχύτητας κάθε φορά

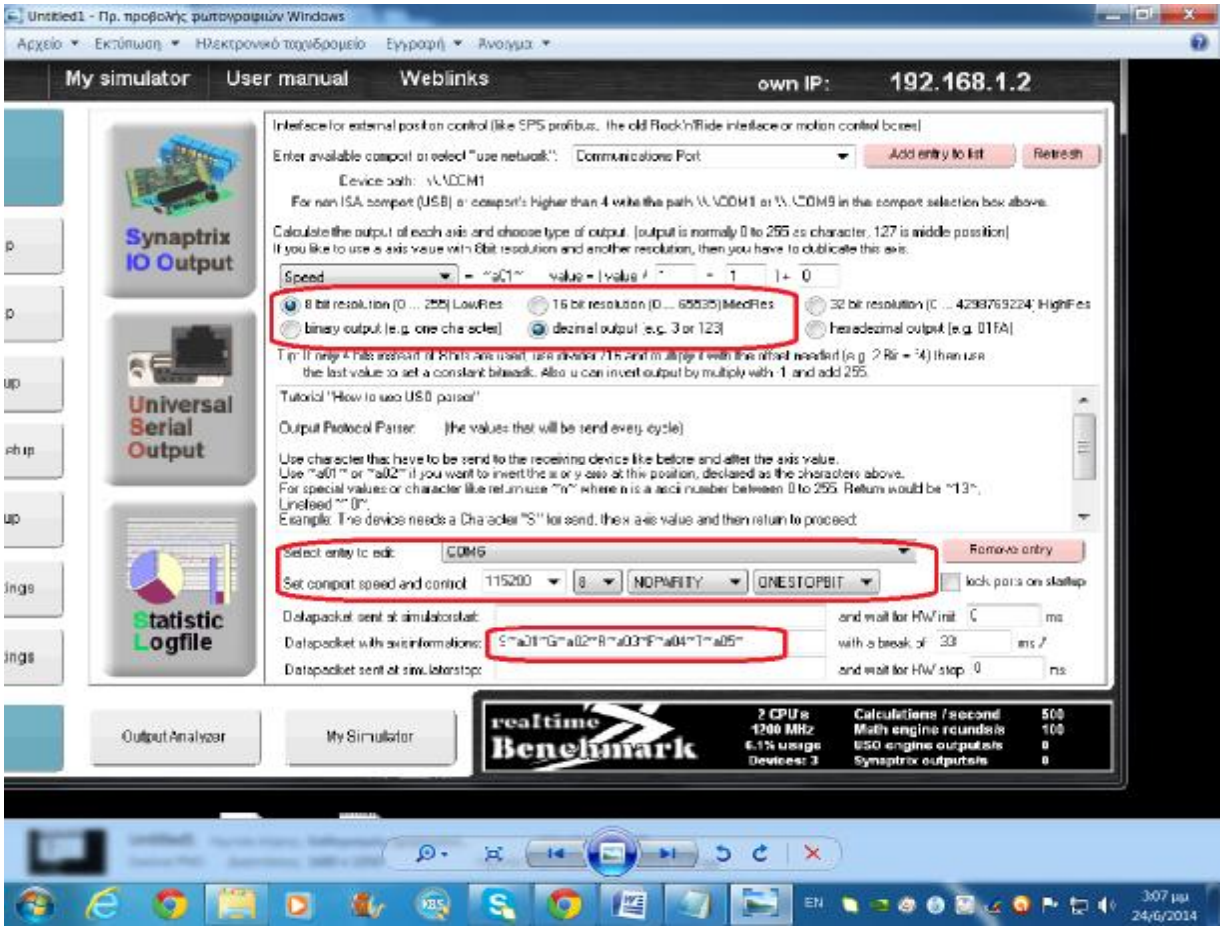
2.5 CURRENT FUEL



Η τιμή του use automatic maximum adjustment είναι επιλεγμένη από το LFS.

Στη παρακάτω εικόνα βρίσκετε η διαδικασία της επικοινωνίας του arduino με το x-sim . Βρισκόμαστε στην καρτέλα output setup και στην υπό καρτέλα UNIVERSAL SERIAL OUTUT . Αναφέραμε πως οι πληροφορίες που στέλνονται απ το x-sim είναι στα 8 bit. Σε αύτη την υπό

καρτέλα γίνονται οι διαδικασίες επικοινωνίας δηλώνοντας την serial comport που έχει διαβάσει ο δαυλός usb της arduino (COM 6) και πατώντας add entry to list βλέπουμε πως η comport έχει διαβαστεί βλ. στο select entry to edit. Ακριβώς από κάτω στο set comport speed and control έχουμε τον αριθμό 115200 τιμή του serial begin. Έδω ρυθμίζουμε την ταχύτητα με την όποια θα στέλνονται τα δεδομένα η όποια θα πρέπει να είναι ιδία με την αρχική τιμή στον κώδικα .Στην γραμμή των data packet with axis information γραφούμε τον κάθε ένα απ τους άξονες όπως τους αντιλαμβάνεται το x-sim. Κάθε άξονας ανάλογα με την σειρά που τον περνάμε στο xsim έχει και δικό του κωδικό. Στην συγκεκριμένη εικόνα ο άξονας του speed είναι ο πρώτος ο όποιος ισούται με ~ a01~ κωδικός για την αναγνώριση του συγκεκριμένου άξονα και καθώς προσθέτουμε άξονες αλλάζει και ο αριθμός του κωδικού. Τον κάθε άξονα αντιπροσωπεύει ένα νούμερο ανάλογα με την σειρά και ένα γράμμα ανάλογα με το είδος του. Speed--> S~ a01~

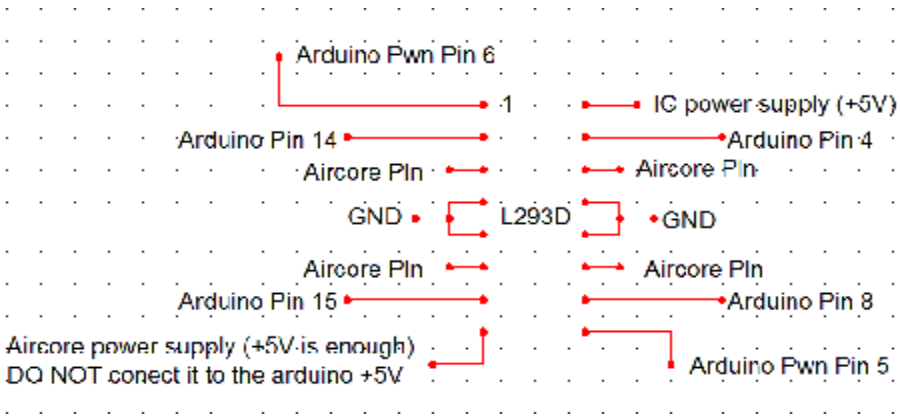


ΚΕΦΑΛΑΙΟ 3 - ΗΛΕΚΤΡΟΝΙΚΟ ΜΕΡΟΣ

Στο προηγούμενο κεφάλαιο ανέφερα την διαδικασία επικοινωνίας arduino - xsim και πως καταφέρνω και περνώ τις εξόδους για κάθε άξονα. Σε αυτό το κεφάλαιο θα αναφερθώ στο ηλεκτρονικό κομμάτι. Το κοντέρ περιέχει 4 μοτέρ .Τα 2 απ αυτά είναι air cores των τεσσάρων Pin και υπόλοιπα είναι με 2 Pin. Το κάθε ένα μοτέρ χεριάζετε ένα ολοκληρωμένο με αναλογικές πύλες για την πλήρη λειτουργιά του.

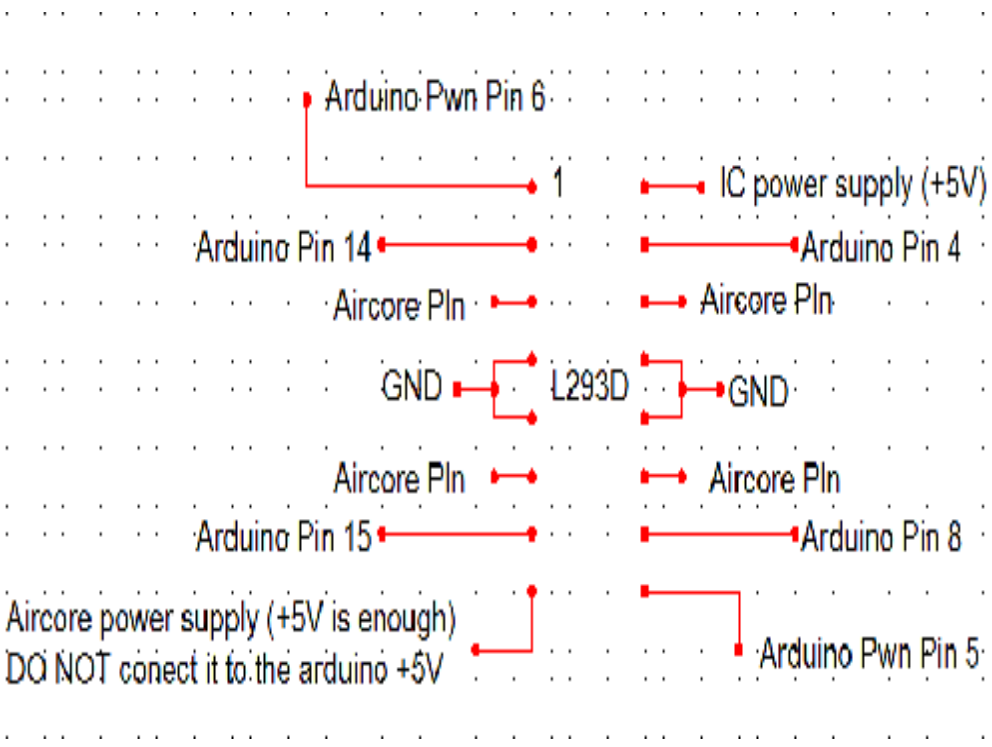
3.1 *ΛΕΙΤΟΥΡΓΙΑ ΣΤΡΟΦΟΜΕΤΡΟΥ*

Για την λειτουργιά του στροφόμετρου χρησιμοποιώ το ολοκληρωμένο *L293D.* Ανήκει στην κατηγορία H-BRIGHES και η εργασία που κάνει είναι οι γρήγορες εναλλαγές πολικότητας πράγμα που ο επεξεργαστής του arduino δεν μπορεί να κάνει. Στην παρακάτω εικόνα φαίνετε η ακριβής συνδεσμολογία του ολοκληρωμένου με την Arduino. Στους ακροδέκτες Air core pln συνδέουμε τα 4 pin του στροφόμετρου. Στα pin 14,15,4,5 συνδέουμε τα ψηφιακά pin από την arduino (40.41.42.43) digital pins ενώ τα 5,6 συνδέονται στα pin παλμού της arduino. Η τροφοδοσία του ολοκληρωμένου βρίσκετε σε 2 σημεία . Το ένα τροφοδοτείτε απευθείας από την arduino ενώ το άλλο από εξωτερική τροφοδοσία ανεξάρτητη από την πλακέτα προγραμματισμού. Στους δυο βραχυκυκλωμένους ακροδέκτες έχουμε την γείωση του ολοκληρωμένου οπού στα αριστερά θα συνδεθεί από την εξωτερική τάση ενώ απ τα δεξιά θα συνδεθεί σε γείωση της arduino.



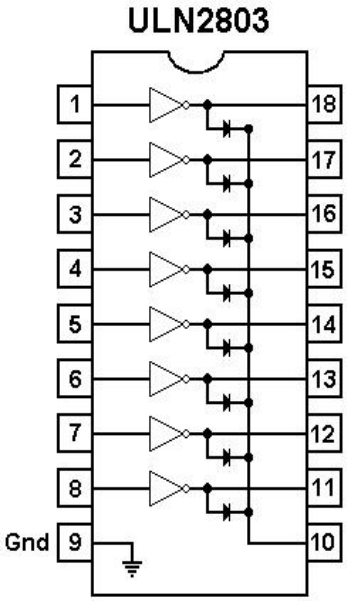
3.2 ***ΛΕΙΤΟΥΡΓΙΑ ΟΔΟΜΕΤΡΟΥ***

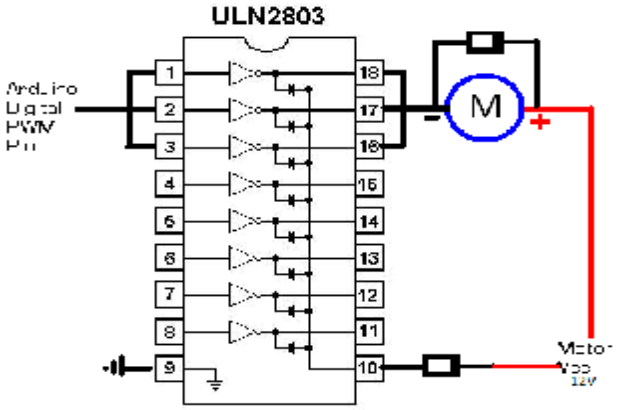
Για την λειτουργιά του οδομέτρου χρησιμοποιώ το ολοκληρωμένο ***L293D.*** Στους ακροδέκτες Air core pln συνδέουμε τα 4 pin του στροφόμετρου. Στα pin 14,15,4,5 συνδέουμε τα ψηφιακά pin από την arduino (44.45.46.47) digital pins ενώ τα 5,6 συνδέονται στα pin παλμού της arduino. Η τροφοδοσία του ολοκληρωμένου βρίσκετε σε 2 σημεία . Το ένα τροφοδοτείτε απευθείας από την arduino ενώ το άλλο από εξωτερική τροφοδοσία ανεξάρτητη από την πλακέτα προγραμματισμού. Στους δυο βραχυκυκλωμένους ακροδέκτες έχουμε την γείωση του ολοκληρωμένου οπού στα αριστερά θα συνδεθεί από την εξωτερική τάση ενώ απ τα δεξιά θα συνδεθεί σε γείωση της arduino.



3.3 Λειτουργια δεικτη βενζινης

Το μοτέρ του δείκτη βενζίνης όπως προείπα έχει 2 Pins πράγμα που δεν μου επιτρέπει να χρησιμοποιήσω το L293D . Για την λειτουργιά του χρησιμοποιώ το ULN2803A . Ολοκληρωμένο 8 bit με κάθε μια είσοδο και παράλληλα έξοδο. Περιέχει αναλογικές πύλες NOT που ανάλογα με την είσοδο δίνουν την κατάλληλη έξοδο. Η ονομαστική του τάση είναι στα 12V Dc με 1A. Στην θέση 1 που βλέπουμε στην εικόνα συνάδουμε ένα pwm παλμό από ένα pin της arduino. Στην Έξοδο 18 συνδέουμε το - του μοτέρ . Το + θα συνδεθεί μαζί με το + της τροφοδοσίας στο σημείο 10 και απ την γείωση του ολοκληρωμένου συνάδουμε άλλο ένα καλώδιο σε μια γείωση της arduino.



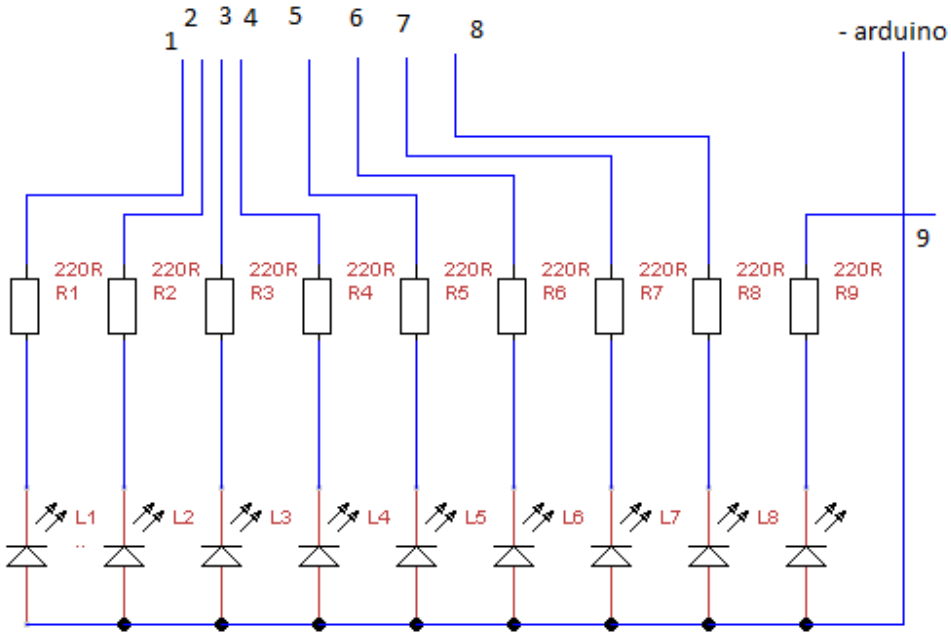


3.4Λειτουργια δεικτη Turbo

Με ένα δεύτερο ULN2803 A πραγματοποιούμε την ιδία διαδικασία με αύτη του δείκτη βενζίνης. Καθώς πραγματοποίησα δόκιμες στο κατασκευαστικό κομμάτι παρατήρησα πως δεν μπορώ με μια τροφοδοτική διάταξη των 12 V να οδηγήσω 2 βελόνες δηλαδή και της βενζίνης και του turbo γι αυτό χρησιμοποίησα ένα δεύτερο ίδιο ολοκληρωμένο ακλουθώντας την ιδία διαδικασία πάλι στα 12 V.

3.5 ΛΕΙΤΟΥΡΓΙΑ RPM SHIFTER LIGHTS

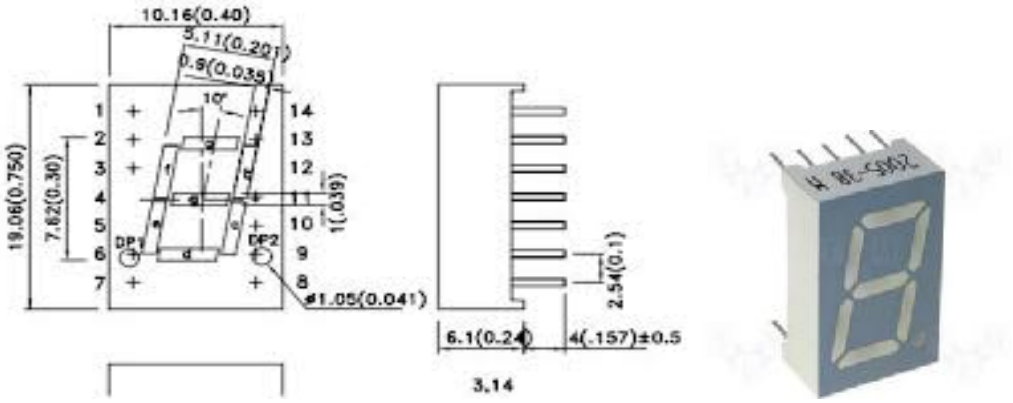
Το RPM SHIFTER είναι μια σειρά από λαμπάκια των 5 V συνδεδεμένα όλα μεταξύ τους. Τα + συνδέονται όλα μεταξύ τους και καταλήγουν σε μια γείωση της arduino. Τα - των λαμπτήρων συνδέονται το κάθε ένα ξεχωριστά με μια ψηφιακή πόρτα της arduino. Στην εικόνα που ακολουθεί φαίνετε η συνδεσμολογία.

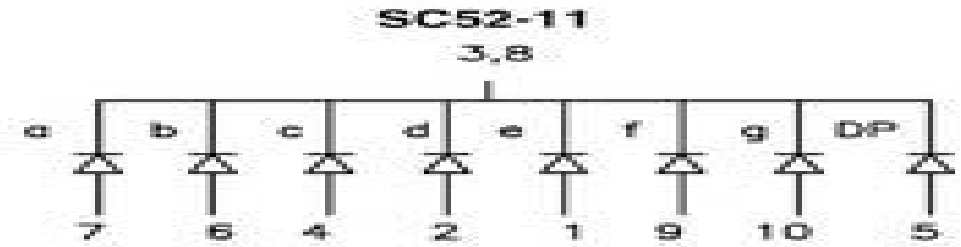


3.6 ***ΛΕΙΤΟΥΡΓΙΑ*** *7 SEGMENT DISPLAY*

Το συγκεκριμένο display είναι το ανόδου το οποίο σημαίνει πως λειτούργει με 5 V. Έχει 10 ακροδέκτες εκ των οποίων οι 2 είναι για την τροφοδοσία (3,8). Από τον ακροδέκτη 1 έως τον 10 συνδέουμε καλώδια τα όποια πέρανε από μια αντίσταση των 180 Ω και στην συνεχεία καταλήγουν στις ψηφιακές εξόδους του arduino

.





3.7 ΛΕΙΤΟΥΡΓΙΑ ΛΑΜΠΤΗΡΩΝ ΚΑΙ ΠΕΡΙΦΕΡΙΑΚΩΝ

Για την λειτουργιά των λαμπτήρων του κοντέρ έχω επέμβει στο κοντέρ κάνοντας κολλήσεις με χαλκό ξεχωριστά σε κάθε λαμπτήρα. Οι λαμπτήρες των φλας που υπήρχαν μέσα στο κοντέρ αφαιρεθήκαν και τοποθέτησαν εκνέου καινούργιοι λαμπτήρες με ρύθμιση εσωτερική on-off των 3 V συνδεδεμένοι στα 12V με μια αντίσταση των 690Ω στον θετικό ακροδέκτη των λαμπτήρων ξεχωριστά. Σαν διακόπτη χρησιμοποιώ τους δείκτες των φλας από κανονικό αυτοκίνητο και παράλληλα τους συνδέουμε με τέτοιο τρόπο έτσι ώστε πατώντας το κουμπί των αλάρμ να αναβοσβήνουν ταυτόχρονα. Στο κατασκευαστικό κομμάτι έχει τοποθετηθεί χειρόφρενο στο οποίο υπάρχουν 2 τερματικοί διακόπτες .Ο ένας διακόπτης είναι συνδεδεμένος σε σειρά με μια πηγή τάσης 12V μαζί με τον λαμπτήρα χειρόφρενου. Ο άλλος τερματικός διακόπτης παρεμβάλλεται στο ηλεκτρονικό κύκλωμα της LOGITECT τιμονιέρας και ουσιαστικά όταν ενεργοποιείτε είναι σαν να ενεργοποιείτε κάποιο button την τιμονιέρας. Η δεύτερη συνδεσμολογία γίνετε για την λειτουργιά του χειρόφρενου στο πραγματικό περιβάλλον του παιχνιδιού προσομοίωσης. Στη συνεχεία στο κατασκευαστικό μέρος έχει τοποθετηθεί προέκταση τιμονιού με μια κολόνα από το golf 3 φέρνοντας στον χρηστή την ρεαλιστικότατα στην προσομοίωση. Η κολόνα έχει μίζα η όποια λειτούργει με τον ίδιο τρόπο όπως το χειρόφρενο. Η Logitech μου έδωσε την δυνατότητα μέσα από μια πλακέτα της η όποια έστελνε σε κάποια button χειρισμού διαφορές εντολές. Πήρα αυτά τα καλώδια και ενώ βρισκόμουν μέσα στις ρυθμίσεις του παιχνιδιού προσομοίωσης τα έφερνα σε επαφή μεταξύ τους παρατηρώντας πως κάποιο ζευγάρι καλωδίων θα αναγνωρίζονταν σε κάποιο button της τιμονιέρας. Όποτε στις ρυθμίσεις επέλεξα το button 22 για την μίζα συνδέοντας το στα καλώδιο που υπήρχαν πάνω στην μίζα της κολόνας.

ΕΠΙΛΟΓΟΣ

Ο σκοπός της συγκεκριμένης μελέτης είναι η λειτουργιά του κοντέρ σε μορφή προσομοίωσης. Με βασικό στοιχειό τον επεξεργαστή της arduino και ένα από τα εγκεκριμένα παιχνίδια προσομοίωσης που μου παρέχουν τις ανάλογες πληροφορίες επεξεργασμένες από το x-sim πρόγραμμα εξομοίωσης καταφέρνουμε τις πληροφορίες αυτές να τις φέρουμε στο περιβάλλον με οπτικό αποτέλεσμα τη κίνηση της κάθε μια βελόνας ξεχωριστά. Με την μορφή γλώσσα C# προγραμματίζουμε αυτές τις πληροφορίες που περνούμε. Η σωστή αλληλουχία και σειρά για την λειτουργιά αυτή βρίσκετε στα δεδομένα που περνούμε από την αλλαγή αυτών των τιμών μέσα στα αρχεία του παιχνιδιού. Ανοίγοντας το πρώτο υπό πρόγραμμα x-sim extractor ενεργοποιούμε το παιχνίδι μέσα από αυτό. Με αυτή την κίνηση όλες οι πληροφορίες και τα δεδομένα στέλνονται στο άλλο υπό πρόγραμμα x-sim converter. Σε αυτό το υπό πρόγραμμα δημιουργούμε ένα προφίλ το όποιο περιέχει κάποια υπό προφίλ ξεχωριστά για κάθε ένα από τα μοτέρ του κοντέρ όπως rpm, speed, fuel. Συνδυάζοντας τις ρυθμίσεις του converter να είναι ίδιες με τις αρχικοποιήσεις στον κώδικα του επεξεργαστή όπως serial begin 115200 έτσι ώστε να γίνει η σωστή επικοινωνία μεταξύ τους ενεργοποιούμε το converter πατώντας start. Το x-sim extractor στέλνει στο converter και αυτό με την σειρά του αλληλοστέλνει δεδομένα με τον επεξεργαστή περνώντας αυτά τα δεδομένα (έξοδοι) μέσα από μια σειρά κώδικα. Το arduino με την σειρά του στέλνει στις εξόδους του (Pins) όλα τα σήματα και τις τάσεις οι όποιες με την σειρά τους καταλήγουν στο κοντέρ δημιουργώντας κίνηση.