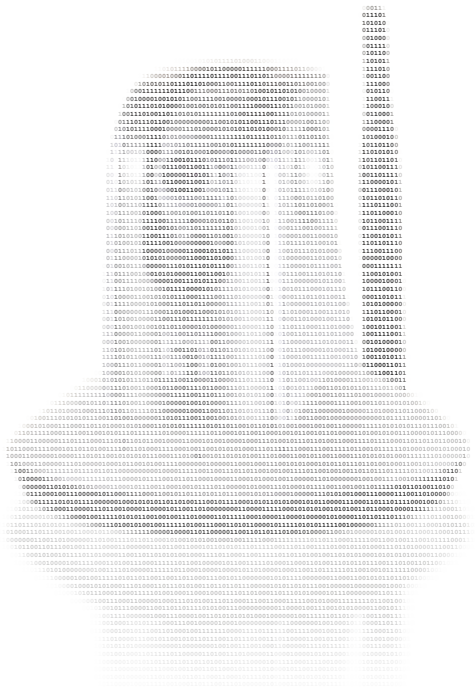
ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

Α.Τ.Ε.Ι. Θεσσαλονίκης
Τμήμα Ηλεκτρονικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ



“Υλοποίηση συστήματος περιμετρικής ανίχνευσης εξωτερικού χώρου με χρήση ασύρματης κάμερας και τηλεειδοποίηση”

ΣΤΑΜΑΤΟΥΛΗΣ ΙΩΑΝΝΗΣ
Κ.Α.Σ.: 503063

Ευχαριστίες

“Θα ήθελα να ευχαριστήσω τον πατέρα μου που επένδυσε τόσα πολλά για την μόρφωση μου και πίστεψε σε μένα...”

Περίληψη

Θέμα της πτυχιακής εργασίας είναι η υλοποίηση ενός συστήματος περιμετρικής ανίχνευσης εξωτερικού χώρου με την χρήση μιας ασύρματης κάμερας και ύπαρξη τηλεειδοποίησης σε πιθανή εισβολή.

Θα γίνεται περιμετρικός έλεγχος του εξωτερικού χώρου από την κάμερα η οποία θα είναι συνδεδεμένη ασύρματα με έναν ηλεκτρικό υπολογιστή στον οποίο θα τρέχει το πρόγραμμα. Αν ανιχνευτεί κίνηση από το πρόγραμμα τότε θα εντοπίζεται ο εισβολέας, η κάμερα θα τον ακολουθεί όπου πάει και ταυτόχρονα θα σώζει στον υπολογιστή φωτογραφίες με τις κινήσεις του εισβολέα. Μετά θα ειδοποιείται ο ιδιοκτήτης ότι έγινε πιθανή παραβίαση του χώρου με email και θα λαμβάνει και φωτογραφία του χώρου.

Abstract

The subject of this project is focusing on the concretisation of a system that provides a perimetric detection of exterior space with the use of wireless IP Camera and mail notification.

The system will perimetrically control the exterior space through a wireless IP Camera which is connected to a computer. If any movement is detected the camera will immediately detect the intruder and follow his every movement. It will also save the intruder's photos on the computer and automatically notify the owner about the possible violation by sending him an email with an attached photo of the space.

Περιεχόμενα

Κεφάλαιο

1

1 Προγραμματισμός

Προγραμματισμός είναι η διαδικασία συγγραφής, δοκιμής, αντιμετώπισης προβλημάτων και συντήρησης του πηγαίου κώδικα των προγραμμάτων των ηλεκτρονικών υπολογιστών. Ο πηγαίος κώδικας είναι γραμμένος σε μία γλώσσα προγραμματισμού. Ο κώδικας μπορεί να είναι μία βελτιωμένη έκδοση ενός ήδη υπάρχοντος προγράμματος ή κάτι εντελώς καινούργιο γραμμένο από την αρχή.

Ο σκοπός του προγραμματισμού είναι η δημιουργία προγραμμάτων τα οποία συμπεριφέρονται όπως ακριβώς θέλουμε εμείς. Η διαδικασία εγγραφής του πηγαίου κώδικα χρειάζεται εξειδίκευση σε πολλούς διαφορετικούς τομείς, οι οποίοι συμπεριλαμβάνουν γνώσεις δομών δεδομένων, γραφής αλγόριθμων κτλ.

Η πρώτη μορφή προγραμματισμού μπορεί να θεωρηθεί ότι εμφανίστηκε πριν αρκετά δεκάδες χρόνια, τον 19ο αιώνα όταν έκαναν την εμφάνισή τους τα μουσικά κουτιά και τα πιάνο τα οποία έπαιζαν από μόνα τους κάποια συγκεκριμένα μουσικά κομμάτια. Αυτά τα κομμάτια ήταν αποτυπωμένα πάνω σε κυλινδρικά τύμπανα με ακίδες, και όπως γύριζαν ακουμπούσαν κάποια μεταλλικά ελάσματα που το κάθε έλασμα ήταν και μία νότα, και έτσι σιγά σιγά γινόταν η αναπαραγωγή του τραγουδιού.

1.1 Η γλώσσα

Γλώσσα προγραμματισμού είναι μία τεχνητή γλώσσα που χρησιμοποιείται για να ελέγξουμε την συμπεριφορά ενός μηχανήματος, και πιο συγκεκριμένα ενός υπολογιστή. Οι γλώσσες προγραμματισμού ορίζονται από συντακτικούς και σημασιολογικούς κανόνες, οι οποίοι περιγράφουν την δομή και το νόημα αντίστοιχα. Πολλές γλώσσες προγραμματισμού έχουν κάποια γραμμένα και προκαθορισμένα τεχνικά χαρακτηριστικά όσων αφορά την σύνταξη τους και την σημασιολογία τους, κάποιες άλλες πάλι ορίζονται μόνο από μία επίσημη εφαρμογή.

Οι γλώσσες προγραμματισμού χρησιμοποιούνται για να διευκολύνουν την επικοινωνία σχετικά με τις εργασίες οργάνωσης και ελέγχου των πληροφοριών και για να εκφράζουν με ακρίβεια διάφορους αλγορίθμους. Μερικοί υποστηρίζουν ότι ο όρος γλώσσα προγραμματισμού αφορά μόνο και μόνο όσες γλώσσες μπορούν να εκφράσουν πλήρως όλους τους πιθανούς αλγορίθμους που υπάρχουν.

Έχουν δημιουργηθεί χιλιάδες διαφορετικές γλώσσες προγραμματισμού και νέες γλώσσες δημιουργούνται και προστίθενται κάθε χρόνο. Είναι δύσκολο να πούμε ποια γλώσσα προγραμματισμού είναι η πιο ευρέως διαδεδομένη και γνωστή. Μία γλώσσα μπορεί να απασχολεί περισσότερες εργατοώρες, μία άλλη να έχει περισσότερες γραμμές πηγαίου κώδικα και μία άλλη να χρησιμοποιεί περισσότερη υπολογιστική δύναμη του υπολογιστή μας.

Μερικές γλώσσες είναι πολύ δημοφιλείς για κάποιο συγκεκριμένο είδος εφαρμογών. Για παράδειγμα, η COBOL είναι ακόμα πολύ διαδεδομένη για συγκεκριμένα είδη εφαρμογών, όπως σε κέντρα δεδομένων αλλά και σε υπολογιστές mainframe. Η FORTRAN για εφαρμογές για μηχανικούς, η C σε ενσωματωμένες εφαρμογές και σε λειτουργικά συστήματα, και άλλες τόσες γλώσσες για άλλες τόσες διαφορετικές χρήσεις και εφαρμογές.

Υπάρχουν αρκετές μέθοδοι για να μπορέσουμε να βρούμε το ποια είναι η πιο δημοφιλής γλώσσα προγραμματισμού, η κάθε μία βασίζεται σε διαφορετική βάση, ανάλογα με τον τρόπο που θα αξιολογηθεί.

Έτσι μερικοί τρόποι που έχουν προταθεί είναι οι εξής:

- Μετρώντας τις αγγελίες για εύρεση προγραμματιστών στη συγκεκριμένη γλώσσα.
- Τον αριθμό των βιβλίων που έχουν πωληθεί που περιγράφουν ή προσπαθούν να διδάξουν στον αναγνώστη την γλώσσα.
- Τον υπάρχον αριθμό γραμμών πηγαίου κώδικα που έχουν γραφτεί στην συγκεκριμένη γλώσσα προγραμματισμού.
- Τον αριθμό των αναφορών μιας γλώσσας που βρίσκονται δια μέσου μιας μηχανής αναζήτησης στο διαδίκτυο.

Γύρω στο 1950 έκαναν την εμφάνιση τους οι πρώτες γλώσσες προγραμματισμού, η FORTRAN, η LISP και η COBOL, των οποίων, κάποιες μεταγενέστερες εκδόσεις τους χρησιμοποιούνται μέχρι και σήμερα για κάποιες συγκεκριμένες εφαρμογές.

Στα τέλη της δεκαετίας του 1960 και αρχές της δεκαετίας του 1970, ιδιαίτερα διαδεδομένη ήταν η γλώσσα C η οποία ακόμα παραμένει ιδιαίτερα δημοφιλής. Το 1980 έγινε στροφή και υιοθετήθηκαν οι αντικειμενοστραφείς γλώσσες προγραμματισμού όπως η C++. Αργότερα το 1990 μέχρι και σήμερα, μαζί με την άνθηση του διαδικτύου, δημιουργήθηκαν νέες γλώσσες που είχαν σαν σκοπό την διαδικτυακή ανάπτυξη εφαρμογών για χρήση μέσα στις σελίδες των διάφορων ιστοτόπων.

1.1.1 C

Η **C** είναι μια γενικής χρήσης διαδικαστική γλώσσα προγραμματισμού η οποία αναπτύχθηκε στις αρχές της δεκαετίας 1970-1980 από τον Dennis Richie στα εργαστήρια Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Σύμφωνα με τον Dennis Richie η πιο δημιουργική περίοδος ήταν το 1972. Ονομάστηκε C γιατί πολλά από τα χαρακτηριστικά της προέρχονται από μία παλαιότερη γλώσσα

προγραμματισμού που λεγόταν Β.

Από τότε χρησιμοποιείται ευρύτατα, και ιδιαίτερα για ανάπτυξη προγραμμάτων συστήματος (system software) αλλά και για απλές εφαρμογές. Οι λόγοι της ραγδαίας ανάπτυξης της συγκεκριμένης γλώσσας προγραμματισμού είναι η ταχύτητα της, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα.

Η C όπως είπαμε είναι μια διαδικαστική γλώσσα προγραμματισμού. Ανάμεσα στους σχεδιαστικούς στόχους που έπρεπε να καλύψει η γλώσσα περιλαμβανόταν το ότι θα μπορούσε να μεταγλωττιστεί άμεσα με τη χρήση single-pass compiler — με άλλα λόγια, ότι θα απαιτούνταν μόνο ένας μικρός αριθμός από εντολές σε γλώσσα μηχανής για κάθε βασικό στοιχείο της, χωρίς εκτεταμένη run-time υποστήριξη. Ως αποτέλεσμα, είναι δυνατό να γραφτεί κώδικας σε C σε low level επίπεδο προγραμματισμού με ακρίβεια ανάλογη της συμβολικής γλώσσας, στην πραγματικότητα η C ορισμένες φορές αποκαλείται "high-level assembly" ή "portable assembly".

Το 1978 ο Dennis Richie και ο Brian Kernighan κυκλοφόρησαν ένα βιβλίο με τίτλο "Η γλώσσα προγραμματισμού C". Αυτό το βιβλίο για σειρά ετών ήταν κάτι σαν ένα ανεπίσημο εγχειρίδιο της γλώσσας το οποίο είχε υιοθετηθεί από μεγάλη μερίδα προγραμματισμών. Αργότερα, το 1989 κυκλοφόρησε η δεύτερη έκδοση του βιβλίου, η οποία έγινε γνωστή και ως ANSI C ή ως Standard C. Ένα χρόνο μετά, το 1990 ο Διεθνής Οργανισμός Προτυποποίησης, με κάποιες μικρές αλλαγές, υιοθέτησε το ANSI C ως το στάνταρ για την γλώσσα C.

Ένα τυπικό πρόγραμμα στην C είναι το παρακάτω:

```
#include <stdio.h>

int main(void)
{
   printf("Καλησπέρα!!!\n");
   return 0;
}
```

Η πρώτη γραμμή είναι μία οδηγία προς τον μεταφραστή, με την οποία δίνεται εντολή να ενσωματωθεί στο πρόγραμμά μας το αρχείο stdio.h. Αυτό

το αρχείο περιέχει τις βασικές ρουτίνες εισόδου και εξόδου για τα προγράμματά μας. Τα **<>** στα οποία ανάμεσα βρίσκεται γραμμένο το όνομα του αρχείου ενημερώνουν τον μεταφραστή για το που βρίσκεται το αρχείο που θέλουμε να συμπεριλάβουμε.

Με την επόμενη γραμμή ορίζουμε μία συνάρτηση με το όνομα main. Το int μπροστά από τον ορισμό της συνάρτησης μας σημαίνει ότι η τιμή που θα επιστραφεί μετά την εκτέλεση της συνάρτησης θα είναι ένας ακέραιος αριθμός (integer). Η λέξη void σημαίνει ότι η συνάρτηση main δεν θα πάρει καμία παράμετρος.

Ανοίγοντας με την αγκύλη (**{**) σημαίνει ότι αρχίζει η συνάρτηση μας.

Η εντολή **printf**, η οποία περιέχεται μέσα στο αρχείο που συμπεριλάβαμε στο πρόγραμμα μας, stdio.h, παίρνει ένα και μοναδικό όρισμα, το οποίο στην περίπτωση μας είναι κείμενο το οποίο κατά την εκτέλεση θα εμφανιστεί στην οθόνη μας. Με το **\n** δίνουμε εντολή ο κέρσορας αμέσως μετά την εκτύπωση του κειμένου να πάει στην από κάτω γραμμή στην αρχή της (new line). Τέλος κάθε εντολή τερματίζεται με ελληνικό ερωτηματικό (**;)**.

Μετά επιστρέφοντας την τιμή 0 (**return 0;**), σημαίνει ότι το πρόγραμμα μας εκτελέστηκε επιτυχώς και το σύστημα μπορεί να σταματήσει την εκτέλεση του.

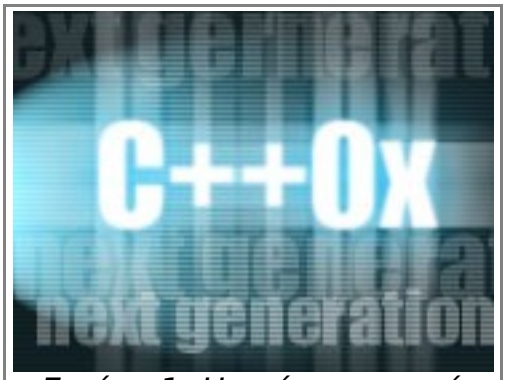
Τέλος, κλείνουμε την αγκύλη (**}**) που ανοίξαμε και μαζί της κλείνει και η συνάρτηση main που ορίσαμε και εκτελέστηκε ήδη.

1.1.2 C++

Η C++ (C Plus Plus) είναι μια γενικού σκοπού και υψηλού επιπέδου γλώσσα προγραμματισμού ηλεκτρονικών υπολογιστών, η οποία ανήκει στην κατηγορία των αντικειμενοστραφών γλωσσών προγραμματισμού. Η C++ αναφέρεται ως μέσου επιπέδου γλώσσα, διότι εμπεριέχει και χαμηλού αλλά και υψηλού επιπέδου γλωσσικά χαρακτηριστικά.

Ο Bjarne Stroustrup ανάπτυξε την C++ το 1979 στα εργαστήρια της Bell

Labs, σαν μία βελτιωμένη έκδοση της C, με το όνομα “C με κλάσεις”. Το 1983 πήρε το σημερινό της όνομα, C++. Βελτιώσεις άρχισαν με την προσθήκη των κλάσεων, οι οποίες ακολουθήθηκαν και από πολλά άλλα νέα χαρακτηριστικά.



Εικόνα 1: Η επόμενη γενιά της C++

Η προτυποποίηση της C++ έγινε το 1998 από τον διεθνή οργανισμό προτυποποίησης. Το 2003 έγινε προτυποποίηση μίας βελτιωμένης έκδοσης, και σήμερα αναπτύσσετε μία πιο βελτιωμένη έκδοση με την κωδική ονομασία C++0X.

1.1.3 Visual C++

Όσο περνούσαν τα χρόνια, και από τα λειτουργικά σύστημα που στηριζόταν στην γραμμή εντολών περνούσαμε σιγά σιγά στα παραθυριακά περιβάλλοντα, δημιουργήθηκε η ανάγκη για την ανάπτυξη εφαρμογών στην μορφή που ξέρουμε σήμερα.

Έτσι, στηριζόμενη στην C++, δημιουργήθηκε η Visual C++ με σκοπό την ανάπτυξη παραθυριακών εφαρμογών και ως γλώσσα προγραμματισμού την C++. Η πρώτη έκδοση της Visual C++ δημιουργήθηκε το 1992 από την Microsoft και έδινε την δυνατότητα στους προγραμματιστές να δημιουργούν διάφορες εφαρμογές συμβατές με την τελευταία έκδοση των Windows.

1.2 Visual Studio

Το σύστημα ανάπτυξης λογισμικού Microsoft Visual Studio είναι μία σουίτα από εργαλεία που βοηθά τον προγραμματιστή, ανεξάρτητα από το επίπεδο γνώσεων του, να δημιουργήσει προγράμματα και λύσεις για διάφορους τομείς. Μερικοί από τους σημαντικούς παράγοντες που κάνει το

Microsoft Visual Studio επιτυχημένο είναι:

Παραγωγικότητα: το Microsoft Visual Studio προάγει συνεχώς καλύτερους τρόπους έτσι ώστε οι προγραμματιστές να ξοδεύουν λιγότερη ενέργεια στην δημιουργία του προγράμματος. Με διάφορα εργαλεία που βοηθούν στο κώδικα, "μάγους" και δυνατότητα χρήσης πολλαπλών γλωσσών για την δημιουργία προγραμμάτων μέσα στο ίδιο προγραμματιστικό περιβάλλον η δημιουργία μιας εφαρμογής επιταχύνετε σημαντικά.

Ενιαίο: με το Visual Studio, οι προγραμματιστές μπορούν μέσω ενός ενιαίου και ενσωματωμένου περιβάλλοντος να τεστάρουν σε πραγματικό χρόνο με άλλα προγράμματα της Microsoft, όπως πχ η δημιουργία εφαρμογών για το Microsoft Office κτλ.

Περιεκτικότητα: Το Visual Studio περιλαμβάνει εργαλεία για όλες τις φάσεις του προγραμματισμού, δοκιμής, αλλαγής, για κάθε είδους προγραμματιστή, οποιοδήποτε επιπέδου.

Ευελιξία: Το Visual Studio έχει σχεδιαστεί για να είναι συμβατό, ασφαλές, λειτουργικό και ανεξάρτητο. Προσφέρει ένα συνδυασμό από στοιχεία που το καθιστούν όσο πιο ευέλικτο γίνεται.

1.2.1 Εκδόσεις και καινοτομίες του Visual Studio

Visual Studio 97

Η πρώτη επίσημη παρουσίαση του Visual Studio από την Microsoft έγινε το 1997 που περιελάμβανε πολλά προγραμματιστικά εργαλεία για πρώτη φορά μαζί. Το Visual Studio 97 είχε δύο εκδόσεις, την Professional και την Enterprise. Συμπεριελάμβανε την Visual Basic 5.0 και την Visual C++ 5.0, που προοριζόταν για προγραμματισμό στα Windows, την Visual J++ 1.1 για εφαρμογές γραμμένες σε Java, και το Visual FoxPro 5.0 που ήταν για εφαρμογές με βάσεις δεδομένων. Ακόμα είχε το Visual InterDev που αφορούσε την δημιουργία δυναμικών ιστοσελίδων χρησιμοποιώντας active server pages (ASP). Το Visual Studio 97 ήταν η πρώτη προσπάθεια της Microsoft

χρησιμοποιώντας μία ενιαία πλατφόρμας ανάπτυξης για πολλαπλές γλώσσες. Τέλος περιελάμβανε την βιβλιοθήκη MSDN που περιείχε πολλαπλούς τρόπους παροχής βοήθειας για τους προγραμματιστές.

Visual Studio 6.0

Η επόμενη έκδοση του Visual Studio ήταν η 6.0 η οποία παρουσιάστηκε τον Ιούνιο του 1998 και ήταν η τελευταία που έτρεχε στην πλατφόρμα των Windows 9X. Όλα τα συμπεριλαμβανόμενα υποπρογράμματα πήγαν στην έκδοση 6.0. Αυτή η έκδοση ήταν η βάση για τους προγραμματιστές της Microsoft για τα επόμενα τέσσερα χρόνια μια και άρχισε η εστίαση στα πακέτα .net. Το Visual Studio 6 ήταν η τελευταία έκδοση που περιελάμβανε την Visual Basic όπως την γνώριζαν οι περισσότεροι προγραμματιστές. Οι επόμενες εκδόσεις θα περιελάμβαναν το πακέτο .net που θα άλλαζε ριζικά το περιβάλλον. Επίσης ήταν η τελευταία έκδοση που περιελάμβανε το Visual J++, το οποίο ήταν ασύμβατο με την Java της Sun. Αυτό έκανε την Sun να μήνυση την Microsoft. Αποτέλεσμα όλου αυτού του δικαστικού παιχνιδιού ήταν η Microsoft να σταματήσει να πουλάει προγραμματιστικά εργαλεία που θα επικεντρωνόταν στο Java Virtual Machine.

Visual Studio .NET (2002)

Η Microsoft παρουσίασε το Visual Studio .NET, τον Φεβρουάριο του 2002 (ενώ η beta έκδοση ήταν διαθέσιμη από το 2001). Η μεγαλύτερη αλλαγή ήταν η παρουσίαση ενός περιβάλλοντος που χρησιμοποιούσε την τεχνολογία .net. Η ανάπτυξη προγραμμάτων χρησιμοποιώντας .NET δεν ήταν πια περίπλοκη αλλά είχε γίνει ευκολότερη. Ήταν η πρώτη έκδοση του Visual Studio το οποίο απαιτούσα πυρήνα NT για να εκτελεστεί, άρα δεν μπορούσε να γίνει εγκατάσταση στα windows 95/98/Me. Μία νέα γλώσσα παρουσιάστηκε, η C#, η οποία επικεντρωνόταν στην νέα πλατφόρμα .net. Η Visual Basic άλλαξε δραματικά και μετονομάστηκε σε Visual Basic .NET. Όλες οι γλώσσες είχαν ένα κοινό περιβάλλον ανάπτυξης το οποίο ήταν εμφανές πιο παραμετροποιήσιμο και έξυπνο. Η εσωτερική αριθμοδότηση της σουίτας ήταν

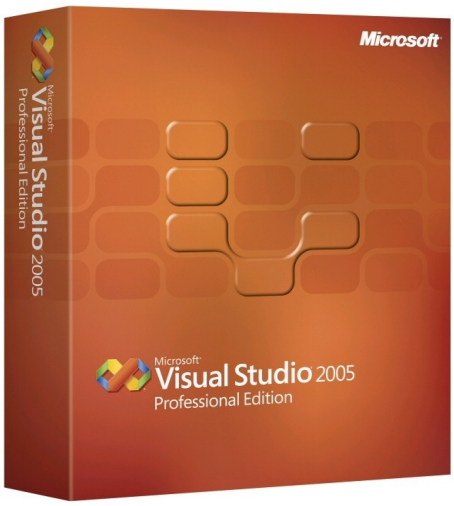
η έβδομη.

Visual Studio .NET 2003

Τον Απρίλιο του 2003, η Microsoft παρουσίασε το νέο Visual Studio το οποίο ονομαζόταν Visual Studio .NET 2003. Τα Visual Studio .NET 2003 κυκλοφόρησαν σε τέσσερις εκδόσεις, την Academic, την Professional, την Enterprise Developer και την Enterprise Architect.

Visual Studio 2005

Το Visual Studio 2005, παρουσιάστηκε στο διαδίκτυο τον Οκτώβριο του 2005 και έκανε την εμφάνιση του στα καταστήματα μερικές εβδομάδες αργότερα. Παρόλο που η Microsoft αφαίρεσε το ακρωνύμιο ".NET" από την σουίτα, το Visual Studio 2005 είχε ως στόχο την τεχνολογία .NET η οποία αναβαθμίστηκε στην δεύτερη έκδοσή της. Ήταν η τελευταία έκδοση που ήταν συμβατή με τα Windows 2000 και η εσωτερική αριθμοδότηση των γλωσσών ήταν η 8.0. Μετά από περίπου από ένα χρόνο η Microsoft παρουσίασε το Service Pack 1 για το Visual Studio 2005. Το Service Pack 1 ήταν ένα σύνολο αναβαθμίσεων που έδινε συμβατότητα για τα Windows Vista που παρουσιάστηκαν λίγους μήνες αργότερα. Το Visual Studio 2005 αναβαθμίστηκε για να υποστηρίζει όλες τις νέες βελτιώσεις του .NET Framework 2.0. Το Visual Studio 2005 επίσης περιελάμβανε και έναν τοπικό εξυπηρετητή σελίδων που μπορούσε να χρησιμοποιηθεί για προσομοίωση διαδικτυακών εφαρμογών. Ακόμα περιείχε μία μικρή έκδοση από τον Microsoft SQL SERVER 2005 που έχει σχέση με την διαχείριση των βάσεων δεδομένων, δημιουργία χρηστών με δικαιώματα για την κάθε εφαρμογή κτλ.



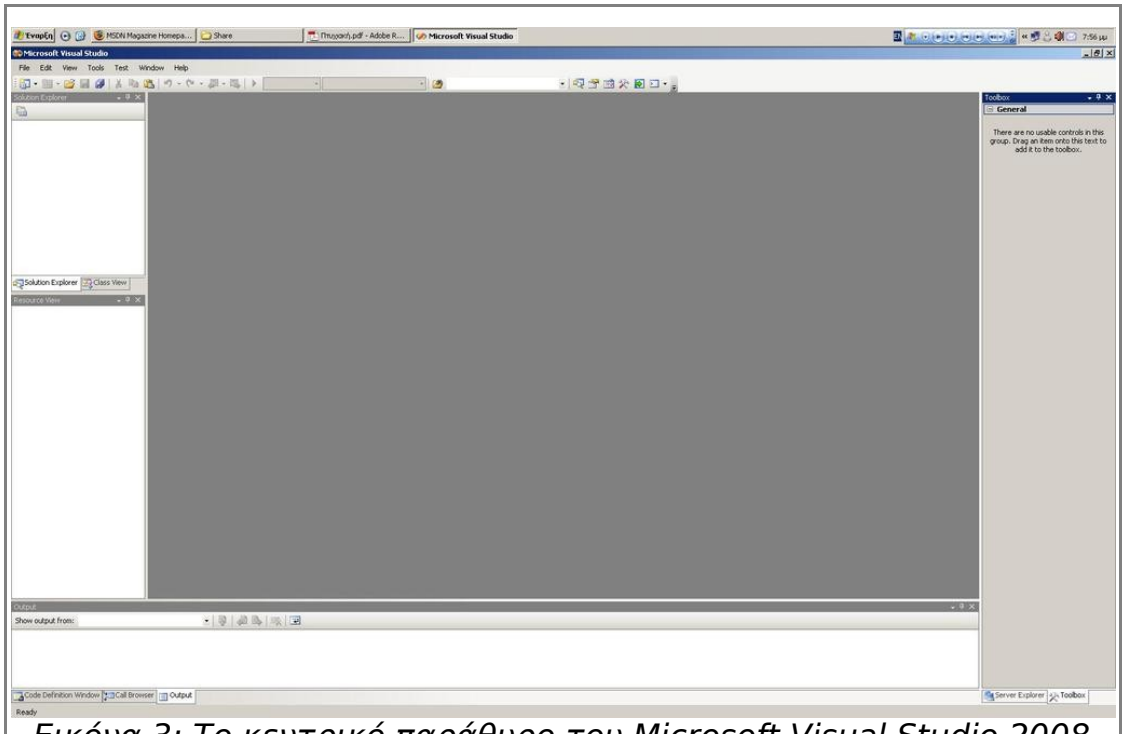
Εικόνα 2: Το σύστημα ανάπτυξης λογισμικού Visual Studio 2005

Visual Studio 2008

Ο διάδοχος του Visual Studio 2005 είναι το Visual Studio 2008 το οποίο παρουσιάστηκε επίσημα στις 19 Νοεμβρίου του 2007 μαζί με την έκδοση 3.5 του .net. Για πρώτη φορά ο πηγαίος κώδικας του Visual Studio θα είναι διαθέσιμος κάτω από μία ειδική άδεια σε μερικούς συνεργάτες της Microsoft.

Το Visual Studio 2008 έχει επικεντρωθεί για την ανάπτυξη εφαρμογών για τα Windows Vista, το Office 2007 αλλά και για εφαρμογές για το διαδίκτυο. Μαζί με όλα τα άλλα, υπάρχουν νέες εκδόσεις της C#, της Visual Basic, αλλά έχουμε και παρουσίαση μιας νέας γλώσσας, της LINQ. Υπάρχει ακόμα ένας βελτιωμένος html editor που ονομάζετε Microsoft Expression Web.

Παρόλο που το Visual Studio 2008 είναι σχεδιασμένο για την 3.5 έκδοση του .net αυτό δεν αναιρεί την ανάπτυξη εφαρμογών και για τις παλαιότερες εκδόσεις του .net. Ακόμα στην νέα έκδοση βρίσκουμε και την νέα έκδοση της MFC η οποία προσθέτει τα οπτικά στυλ των Windows Vista στις εφαρμογές μας. Αυτή η έκδοση επίσης επικεντρώνεται στην ευκολότερη εκσφαλμάτωση των εφαρμογών που τρέχουν με ταυτόχρονες διεργασίες (multi-threading).



Εικόνα 3: Το κεντρικό παράθυρο του Microsoft Visual Studio 2008

Το Visual Studio 2008 έρχεται σε 4 εκδόσεις (Visual Studio 2008 Professional Edition, Visual Studio Team System 2008 Team Suite, Visual Studio Team System 2008 Team Foundation Server και Visual Studio Team System 2008 Test Load Agent) τις οποίες μπορεί κάποιος να κατεβάσει και να δοκιμάσει για 90 μέρες χωρίς καμία υποχρέωση, ενώ για τους πιο ερασιτέχνες και φοιτητές προγραμματιστές υπάρχουν και οι δωρεάν εκδόσεις του Visual Studio 2008 με το όνομα Express Edition.

Γενικότερα το νέο Visual Studio 2008 φαίνεται αρκετά ανανεωμένο, ενώ η Microsoft δείχνει να αλλάζει νοοτροπία και να κάνει όλο και πιο πολλά βήματα προς τα web standards (και στον browser της αλλά και στα development tools της), μιας και το Visual Studio δείχνει να κάνει αρκετά βήματα προς τα εκεί.

1.3 MFC

Η MFC (Microsoft Foundation Class), παρουσιάστηκε το 1992, στην έκδοση 7 της Microsoft C για χρήση με τις 16μπιτες εκδόσεις των windows. Είναι μία βιβλιοθήκη που ενσωματώνει διάφορες λειτουργίες των Windows και έτσι είναι εύκολη η χρήση τους από τους προγραμματιστές. Έτσι ενσωματώνονται κάποια αντικείμενα και κάποια παράθυρα που είναι κοινά σε όλες τις εφαρμογές των Windows.

Η MFC 8.0 παρουσιάστηκε με το Visual Studio 2005, ενώ η 9η έκδοση με το Visual Studio 2008. Παρόλα αυτά η MFC δεν περιλαμβάνετε στην δωρεάν έκδοση του Visual C++ 2005/2008 Express. Το ενδιαφέρον της Microsoft για την MFC μειώθηκε με την εμφάνιση του .NET Framework, παρόλα αυτά, λόγω του ότι είναι μία ευρέως διαδεδομένη τεχνολογία, παραμένει να χρησιμοποιείτε ευρέως.

Παρακάτω παρουσιάζετε ένας πίνακας με τις εκδόσεις της MFC και σε ποια αντίστοιχη έκδοση της Visual C παρουσιάστηκαν.

Έκδοση της Visual C	Έκδοση της MFC
Microsoft C/C++ 7.0	MFC 1.0
Visual C++ 1.0	MFC 2.0
Visual C++ 2.0	MFC 3.0
Visual C++ 4.0	MFC 4.0 (Windows 95)
Visual C++ 4.1	MFC 4.1
Visual C++ 4.2	MFC 4.2 (Windows 98)
Visual C++ 5.0	MFC 4.21
Visual C++ 6.0	MFC 6.0
Visual C++ .NET 2002	MFC 7.0
Visual C++ .NET 2003	MFC 7.1
Visual C++ 2005	MFC 8.0
Visual C++ 2008	MFC 9.0.21022

Κεφάλαιο

2

2 Δίκτυα Υπολογιστών

Δίκτυα υπολογιστών είναι μία διασύνδεση μεταξύ μίας ομάδας υπολογιστών μεταξύ τους με στόχο την κοινή χρήση δεδομένων και περιφερειακών συσκευών, με λίγα λόγια την κοινή χρήση πόρων.

Πριν την άνθηση των υπολογιστών, τα δίκτυα χρησιμοποιούνταν κατά κύριο λόγω στις τηλεπικοινωνίες, για την επικοινωνία μεταξύ μηχανημάτων και διάφορων τηλεπικοινωνιακών συστημάτων για την δρομολόγηση κλήσεων.

Το 1964 μία ερευνητική ομάδα στο MIT που υποστηριζόταν από την General Electric και την Bell Labs χρησιμοποίησε έναν υπολογιστή για την δρομολόγηση και την οργάνωση των τηλεφωνικών κλήσεων. Το 1965 ο Thomas Merrill και ο Lawrence Robert δημιούργησαν το πρώτο δίκτυο ευρείας περιοχής. Το 1969 τρία αμερικάνικα πανεπιστήμια συνδέθηκαν μεταξύ τους δημιουργώντας το γνωστό ARPANet χρησιμοποιώντας κυκλώματα που λειτουργούσαν στα 50kbps. Η αρχή είχε γίνει και σιγά σιγά έφτασαν στην σημερινή μορφή τους.

Τα δίκτυα υπολογιστών μπορούν να χωριστούν σε κατηγορίες με διάφορα κριτήρια. Ένας τυπικός διαχωρισμός που γίνεται είναι με βάση τον τρόπο σύνδεσης. Έτσι έχουμε τα τοπικά δίκτυα και τα δίκτυα ευρείας περιοχής, LAN και WAN αντίστοιχα.

Ένας άλλος διαχωρισμός των δικτύων υπολογιστών είναι με βάση την τοπολογία σύνδεσης. Έτσι έχουμε τις παρακάτω τοπολογίες σύνδεσης: Διαύλου, Αστέρα, Δακτυλίου, Πλέγματος, Κορμού, Πλέγματος

2.1 Ενσύρματα δίκτυα (Ethernet)

Ethernet είναι μία τεχνολογία στην δικτύωση υπολογιστών για τα τοπικά δίκτυα (LAN). Το όνομα προέρχεται από τον αιθέρα. Η προτυποποίηση του ethernet είναι γνωστή ως IEEE 802.3

Το ethernet αρχικά αναπτύχθηκε από την Xerox κατά το 1973-1975. Η πειραματική ταχύτητα του ethernet, όπως αναφερόταν στα χαρτιά ήταν στα 3Mbit/s. Πολύ αργότερα, το 1980 έγινε η προτυποποίηση του 10Mbits/sec το οποίο διαγωνιζόταν δύο μεγάλα συστήματα δικτύωσης, το token ring και το arcnet, αλλά σύντομα το ethernet **επικράτησε.**

Εικόνα 4: Το ανεστραμμένο ζεύγος καλωδίων ethernet

Το ανεστραμμένο ζεύγος καλωδίων ethernet αναπτύχθηκε από τα μέσα της δεκαετίας του 1980 ξεκινώντας με την ονομασία StarLAN, αλλά σύντομα έγινε ευρέως γνωστό σαν 10BASE-T. Ο συνδυασμός του καλωδίου ανεστραμμένου ζεύγους για την σύνδεση των υπολογιστών στο δίκτυο, καθώς και οι οπτικές ίνες για τις συνδέσεις κορμού είναι η πιο συνηθισμένη και ευρέως γνωστή τεχνολογία ενσύρματης δικτύωσης. Χρησιμοποιείτε από το 1980 μέχρι σήμερα αντικαθιστώντας παλαιότερες τεχνολογίες.

Το ethernet αρχικά βασίστηκε στην ιδέα να επικοινωνούν οι υπολογιστές δια μέσου ενός καλωδίου. Η μέθοδος που υιοθετήθηκε είχε πολλές ομοιότητες με τα συστήματα του ραδιοφώνου με κάποιες διαφορές βέβαια, όπως ότι τα προβλήματα στην μετάδοση είναι αρκετά ποιο εύκολο να εντοπιστούν σε ένα καλώδιο παρά σε μία γραμμή ασύρματης μετάδοσης.

Με την εδραίωση του ethernet και την αντικατάσταση του ομοαξονικού

καλωδίου από αυτό, αμέσως το κόστος έπεσε, αυξήθηκε η αξιοπιστία και μειώθηκε ο χρόνος εύρεσης και αποκατάστασης βλαβών.

Λόγω της ευκολίας που προσέφερε το ethernet, η εκθετικά αυξανόμενη αγορά των υπολογιστών έπρεπε να το υποστηρίξει, έτσι σήμερα όλες οι μητρικές πλακέτες των ηλεκτρονικών υπολογιστών ενσωματώνουν μία θύρα ethernet ανεστραμμένου ζεύγους, γλυτώνοντας το τελικό χρήστη από το κόστος αγοράς, καθώς και το χώρο και χρόνο εγκατάστασης μία πρόσθετης κάρτας ενσύρματης δικτύωσης.

2.2 Ασύρματα δίκτυα (WLAN 802.11)

Ασύρματο δίκτυο (Wireless Local Area Network, WLAN) είναι ένα δίκτυο το οποίο συνδέει δύο ή περισσότερους υπολογιστές ασύρματα χωρίς να χρησιμοποιούνται καθόλου καλώδια. Το wlan χρησιμοποιεί μία τεχνολογία διαμόρφωσης η οποία στηρίζεται στα ραδιοκύματα έτσι ώστε να επιτρέψει την σύνδεση και την επικοινωνία ανάμεσα σε συσκευές σε περιορισμένο χώρο.

Το 1970 στο Πανεπιστήμιο της Χαβάης, δημιουργήθηκε το πρώτο ασύρματο δίκτυο υπολογιστών στο κόσμο χρησιμοποιώντας φτηνά εξαρτήματα σαν αυτά των ραδιοερασιτεχνών το οποίο ονομάστηκε alohanet. Η τοπολογία που χρησιμοποιήθηκε ήταν η τοπολογία αστέρα και περιελάμβανε εφτά σταθμούς εργασίας σε τέσσερα νησιά που επικοινωνούσαν με το κεντρικό υπολογιστή χωρίς την χρήση τηλεφωνικών γραμμών. Από το 1979 ξεκίνησε η δημοσίευση διαφόρων εργασιών με διάφορες παραλλαγές για ασύρματες συνδέσεις υπολογιστών.

Τα ασύρματα δίκτυα, με την μορφή που τα γνωρίζουμε σήμερα, ξεκίνησαν μετά το 1996. Από εκεί και μετά άρχισε η παραγωγή ασύρματων συστημάτων που είχαν σαν σκοπό την χρήση τους σε νοσοκομεία, τράπεζες και άλλες επιχειρήσεις και κτήρια.

Μέχρι τότε όμως τα ασύρματα δίκτυα ήταν πολύ ακριβά για τον απλό χρήστη. Το 1999 όμως, στην έκθεση MacWorld στην Νέα Υόρκη, έγινε η επίσημη παρουσίαση ενός iBook το οποίο συνδεόταν στο διαδίκτυο ασύρματα,

χωρίς την χρήση καλωδίων. Η όλη εφαρμογή εκπέμπει στα 2,4GHz καθώς και κάποιες παραλλαγές του στα 5GHz.

Όπως σε κάθε τι που υπάρχει στην γη, έτσι και τα ασύρματα δίκτυα έχουν πλεονεκτήματα και μειονεκτήματα. Ας δούμε μερικά από αυτά.

Ως πλεονέκτημα μπορούμε να ονομάσουμε την φορητότητα που μας δίνει το ασύρματο δίκτυο, έτσι μαζί με αυτήν έχουμε μεγάλη αύξηση των πωλήσεων των φορητών υπολογιστών.

Ακόμα ένα μεγάλο πλεονέκτημα είναι ότι με μία συσκευή που εκπέμπει, μπορούν να συνδεθούν παραπάνω από ένα μηχάνημα, πράγμα που στα ενσύρματα δίκτυα ήταν αδύνατο.

Ακόμα, αν και άμεσα δεν είναι εμφανές, το κόστος του ασύρματου δικτύου είναι πολύ μικρότερο από αυτό των ενσύρματων δικτύων. Γιατί έχει ένα αρχικό κόστος εγκατάστασης και τίποτα άλλο. Στα ενσύρματα δίκτυα, στην πάροδο του χρόνου θα χρειαστεί συντήρηση των καλωδίων κτλ που σε βάθος χρόνου ανεβάζει το κόστος.

Ένα από τα μεγαλύτερα μειονεκτήματα των ασύρματων δικτύων είναι η ασφάλεια. Επειδή τα πακέτα των δεδομένων ταξιδεύουν στον χώρο, κάποιος εύκολα μπορεί να υποκλέψει αυτά τα πακέτα και να μας αποσπάσει σημαντικές πληροφορίες. Βέβαια όπως θα δούμε παρακάτω υπάρχουν διάφορες κωδικοποιήσεις στα δεδομένα που αποστέλλονται και έτσι αυξάνετε κατά πολύ το πρόβλημα της ασφάλειας.

Σημαντικό μειονέκτημα ακόμα είναι η ακτίνα εκπομπής του ασύρματου δικτύου η οποία συνήθως περιορίζετε σε μερικές δεκάδες μέτρα, που συνήθως όμως είναι μια χαρά για ένα σπίτι, αλλά για μία μεγάλη εταιρία με πολλά γραφεία δεν είναι και τόσο καλό. Για να μεγαλώσει η ακτίνα εκπομπής του ασύρματου δικτύου υπάρχουν συσκευές που ονομάζονται repetears που μπορούν να μεγαλώσουν την ακτίνα.

Τα ασύρματα δίκτυα μειονεκτούν ακόμα και στην ταχύτητα σε σχέση με τα ενσύρματα. Η μέγιστη ταχύτητα που μπορεί να επιτευχθεί ασύρματα τυπικά είναι μέχρι 108Mbps τα οποία αν συγκριθούν με τα 100Mbps ή τα 1000Mbps φαίνονται λίγα. Διότι είναι πολύ δύσκολο να επιτευχθεί η

ταχύτητα των 108Mbps λόγω διάφορων παραμέτρων όπως πχ φυσικά εμπόδια κτλ. Βέβαια σιγά σιγά, όπως θα δούμε και παρακάτω, υπάρχουν σχέδια για επέκταση των ταχυτήτων στα 100-200Mbps.

Το 1997 υιοθετήθηκε ένα σύνολο από standar για τα ασύρματα τοπικά δίκτυα, το IEEE 802.11, το οποίο περιλαμβάνει για όλα τα πρωτόκολλα του ασύρματου δικτύου όλα τα χαρακτηριστικά που πρέπει να έχουν οι συσκευές.

Αργότερα υπήρξαν κάποιες προσθήκες στο 802.11, το 802.11a,802.11b κτλ, τα οποία είναι κατά κάποιο τρόπο βελτιωμένες εκδόσεις του κλασσικού 802.11 ως προς την ταχύτητα, την ακτίνα εκπομπής, τις συχνότητες λειτουργίας, καθώς και την ασφάλεια στην μεταφορά δεδομένων.

Υπάρχουν τρεις τύποι σύνδεσης για τα ασύρματα δίκτυα.

1. ***Peer-to-Peer***: σε αυτή την συνδεσμολογία οι σταθμοί εργασίας συνδέονται απευθείας ο ένας με τον άλλον χωρίς να συνδέονται σε κάποιο κεντρικό σταθμό εργασίας.
2. *Bridge*: αυτή η συνδεσμολογία χρησιμοποιείτε για να μετατραπεί μία συσκευή ενσύρματη σε ασύρματη και έτσι να συνδεθεί σε ένα ασύρματο δίκτυο.
3. *Access Point*: εδώ, οι σταθμοί εργασίας συνδέονται όλοι στο access point το οποίο οργανώνει και τις υπηρεσίες και λειτουργεί ως κεντρικός υπολογιστής.

Τα τελευταία χρόνια τα ασύρματα δίκτυα δεν περιορίζονται μόνο για τις συνδέσεις μεταξύ ηλεκτρονικών υπολογιστών, αλλά και σε άλλες συσκευές, όπως κινητά τηλέφωνα, PDA, PNA κτλ. Έτσι βασικές, και όχι μόνο, εργασίες του χρήστη στο διαδίκτυο μπορούν να γίνουν από την φορητή συσκευή του χρήστη γρήγορα και εύκολα.

2.2.1 Ταχύτητα & Εμβέλεια ασύρματων δικτύων

Η πρώτη έκδοση του προτύπου 802.11 που παρουσιάστηκε το 1997 υποστήριζε ταχύτητες μέχρι 2 Mbps και η ακτίνα του ήταν 20 μέτρα για

εσωτερικούς χώρους και 100 μέτρα για τους εξωτερικούς χώρους.

Το 1999 υιοθετήθηκε το 802.11a που έχει κάποιες αλλαγές από το κλασσικό 802.11. Η σημαντικότερη αλλαγή ήταν η συχνότητα εκπομπής που ήταν τα 5GHz έναντι των 2,4GHz του 802.11. Η ακτίνα του βελτιώθηκε στα 35 μέτρα για εσωτερικό χώρο και 120 μέτρα για εξωτερικό χώρο. Η ταχύτητα είχε φτάσει τα 54Mbps.

Την ίδια εποχή παρουσιάστηκε και το 802.11b που μπορεί να υστερούσε σε ταχύτητα σε σχέση με το 802.11a αλλά είχε βελτιωθεί αρκετά η εμβέλεια του. Τέσσερα χρόνια μετά έκανε την εμφάνισή του το 802.11g το οποίο βελτίωνε την ταχύτητα στα 54Mbps και η ακτίνα δράσης του ήταν 38 και 140 μέτρα για εσωτερικό και εξωτερικό χώρο αντίστοιχα.

Εικόνα 5: Το επίσημο σήμα πιστοποίησης για χώρους με ασύρματη πρόσβαση

Τέλος έχει προταθεί για προτυποποίηση το 802.11n το οποίο λειτουργεί και στα 5GHz αλλά και στα 2,4GHz και η ταχύτητα μπορεί να φτάσει τα 300Mbps και η ακτίνα του για εσωτερικό χώρο τα 70 μέτρα.

Στην τρέχουσα εφαρμογή, επιλέξαμε ως ταχύτητα σύνδεσης τα 54Mbps (802.11g) για να μην έχουμε προβλήματα με το δίκτυο με τις επιμέρους χρήσεις του ασυρμάτου δικτύου. Αν επιλέγαμε πιο αργή ταχύτητα ίσως αντιμετωπίζαμε προβλήματα, χάνονταν πακέτα και δεν μεταδιδόταν οι εντολές από και προς την κάμερα. Επίσης με το 802.11g έχουμε εμβέλεια του δικτύου σε εξωτερικούς χώρους γύρω στα 140 μέτρα που είναι υπεραρκετή για την συγκεκριμένη εφαρμογή.

2.2.2 Ασφάλεια στα ασύρματα δίκτυα

Ένα μεγάλο θέμα στα ασύρματα δίκτυα είναι το θέμα της ασφάλειας και η ανάγκη για αυτήν. Πολλά παλαιότερα μοντέλα των AP, δεν μπορούσαν να

προσφέρουν ελεγχόμενη πρόσβαση στο δίκτυο. Παρόλο που αυτό το πρόβλημα κατά κάποιο τρόπο υπήρχε και στα ενσύρματα δίκτυα, όμως δεν αποτέλεσε ποτέ σημαντικό πρόβλημα για αυτά, διότι συνήθως πολλοί οργανισμοί είχαν πολύ καλή ασφάλεια σε φυσικό επίπεδο. Όμως το γεγονός ότι το ασύρματο σήμα εκπέμπεται και εκτός από ένα κτήριο, και γενικά εκτός της ιδιοκτησίας μας δημιουργεί ένα θέμα ασφάλειας.

Οποιοσδήποτε μέσα στην εμβέλεια ενός ανοιχτού, μη κρυπτογραφημένου ασύρματου δικτύου, μπορεί να συνδεθεί και να πάρει μη εξουσιοδοτημένη πρόσβαση στο εσωτερικό μας δίκτυο και να αποκτήσει πρόσβαση σε πόρους του δικτύου μας, όπως το ίντερνετ και γενικά να το χρησιμοποιήσει για παράνομες πράξεις με αποτέλεσμα η παράνομη πράξη να βαραίνει τον ιδιοκτήτη του δικτύου, που σε επίπεδο δικτύων εταιριών είναι σημαντικό θέμα.

Υπάρχουν αρκετοί μέθοδοι για να ασφαλιστεί ένα ασύρματο δίκτυο. Ένας από αυτούς είναι να υπάρχει ελεγχόμενη πρόσβαση στο δίκτυο μέσω της MAC Address του κάθε υπολογιστή. Έτσι να μπορούν να συνδέονται μόνο όσοι υπολογιστές είναι εξουσιοδοτημένοι και να χρησιμοποιούν τους πόρους του δικτύου.

Ένας άλλος τρόπος για να θωρακίσουμε ένα ασύρματο δίκτυο είναι το κλείδωμα του με ένα κωδικό. Υπάρχουν τρεις διαφορετικοί τρόποι που μπορούμε να το κάνουμε αυτό, ανάλογα με το επίπεδο ασφάλειας που θέλουμε να έχουμε. Έτσι έχουμε το WEP, το οποίο τείνει να ξεπεραστεί λόγω τον πολύ σοβαρών κενών ασφαλείας, το WPA και το WPA2 τα οποία σχεδιάστηκαν για να αντιμετωπίσουν τα προβλήματα του WEP και ειδικά το WPA2 τα καταφέρνει αρκετά καλά.

2.2.3 Σημεία πρόσβασης (Access Point)

Ασύρματο σημείο πρόσβασης, ή Wireless Access Point, είναι η συσκευή η οποία συνδέει ασύρματα συσκευές που υποστηρίζουν ασύρματη δικτύωση έτσι ώστε μαζί να συγκροτήσουν ένα ασύρματο δίκτυο.

Το Access Point συνήθως συνδέετε ενσύρματα σε ένα δίκτυο ethernet μέσω του οποίου ανταλλάσσονται δεδομένα μεταξύ όλων των συσκευών του δικτύου. Επίσης υπάρχει η δυνατότητα να συνδεθούν μεταξύ τους πολλά Access Point και να συγκροτήσουν ένα μεγάλο δίκτυο ασύρματης δικτύωσης σε μία μεγάλη περιοχή.

Εικόνα 6: Το Access Point που θα χρησιμοποιηθεί στην εφαρμογή

Λόγω του χαμηλού τους κόστους και της εύκολης δικτύωσης τους τα Access Point έγιναν ευρέως διαδεδομένα από τις αρχές του 2000. Η χρήση τους γίνεται από οικιακούς χρήστες αλλά και από επιχειρήσεις λόγω του ότι προσφέρει εύκολη και χαμηλού κόστους δικτύωση.

Κάποιοι περιορισμοί των Access Point είναι ότι μπορούν ασύρματα να συνδεθούν συνήθως μέχρι 30 πελάτες σε μία ακτίνα 100 μέτρων. Η απόσταση αυτή όμως εξαρτάται από τον περιβάλλοντα χώρο καθώς και το αν η χρήση γίνεται σε κλειστό ή ανοιχτό χώρο.

Ένας άλλος περιορισμός αφορά το εύρος ζώνης του δικτύου που στηρίζεται πάνω στα Access Point. Τα πρώτα Access Point είχαν εύρος ζώνης 11MBps, τα σημερινά υποστηρίζουν είτε 54Mbps είτε 108MBps και υπάρχουν σχέδια για περαιτέρω αύξησή του.

Κεφάλαιο

3

3 Ψηφιακή Επεξεργασία Εικόνας

Επεξεργασία εικόνας ονομάζετε η οποιαδήποτε μορφής επεξεργασία σήματος η οποία έχει σαν είσοδο μία εικόνα, όπως φωτογραφίες, ή ένα καρέ από ένα βίντεο. Η έξοδος μίας επεξεργασίας εικόνας μπορεί να είναι είτε μία εικόνα είτε κάποιο σύνολο από χαρακτηριστικά ή παραμέτρους που σχετίζονται με την εικόνα. Οι περισσότερες τεχνικές επεξεργασίας εικόνας προϋποθέτουν τον χειρισμό της εικόνας σαν σήμα δύο διαστάσεων εφαρμόζοντας της κλασσικές μεθόδους επεξεργασίας εικόνας σε αυτήν.

Εικόνα 7: Η ψηφιακή επεξεργασία βρίσκει εφαρμογές σε πολλούς τομείς

Ψηφιακή επεξεργασία εικόνας ονομάζεται η χρήση υπολογιστικών αλγορίθμων για την εκτέλεση επεξεργασίας εικόνας σε ψηφιακές εικόνες. Σαν υποκατηγορία των ψηφιακών σημάτων επεξεργασίας. Η ψηφιακή επεξεργασία εικόνας έχει πολλά προτερήματα σε σχέση με την αναλογική επεξεργασία εικόνας. Μπορεί να εισάγει πολλούς περισσότερους αλγορίθμους στην είσοδο και μπορεί να εξαλείψει προβλήματα όπως ο θόρυβος και η παραμόρφωση σήματος κατά την επεξεργασία στην έξοδο.

3.1 Σύντομη ιστορία

Πολλές από τις τεχνικές της ψηφιακής επεξεργασίας εικόνας, αναπτύχθηκαν την δεκαετία του 1960 στο MIT, στα Bell Labs, στο Maryland κτλ οι οποίες είχαν σκοπό να εφαρμοστούν στις δορυφορικές εικόνες, ιατρικές εικόνες, βιντεοτηλέφωνα, αναγνώριση χαρακτήρων καθώς και στην επεξεργασία εικόνας για αισθητικούς λόγους. Λόγω όμως του υψηλού κόστους των υπολογιστών της εποχής καθώς και της μειωμένης υπολογιστικής ισχύς την καθιστούσαν απαγορευτική.

Την δεκαετία του 1970, η ψηφιακή επεξεργασία της εικόνας παρουσίασε αλματώδη άνοδο λόγω του ότι εμφανίστηκαν φτηνότεροι ηλεκτρονικοί υπολογιστές και αποκλειστικοί διακομιστές εμφανίστηκαν για αυτό το λόγο. Οι εικόνες πλέον μπορούσαν να επεξεργάζονται σε πραγματικό χρόνο. Όσο οι προσωπικοί ηλεκτρονικοί υπολογιστές άρχισαν να γίνονται όλο και πιο γρήγοροι οι αποκλειστικοί διακομιστές σιγά σιγά παραμερίστηκαν.

Σήμερα, με της ταχύτητες των σημερινών επεξεργαστών, η ψηφιακή επεξεργασία εικόνας έχει γίνει η πιο κοινή μορφή επεξεργασία εικόνας και χρησιμοποιείτε όχι μόνη επειδή είναι η πιο πετυχημένη μέθοδος, αλλά επίσης και η πιο φτηνή.

Πριν την εμφάνιση των ηλεκτρονικών υπολογιστών, η επεξεργασία εικόνας γινόταν με μελάνι, μπογιές, διπλοεκθέσεις, κομμάτιασμό φωτογραφιών καθώς και ένωση αρνητικών στους σκοτεινούς θαλάμους.

3.2 Εύρεση διαφορών ανάμεσα σε δύο εικόνες & Υπολογισμός κέντρου βάρους

Για την εύρεση διαφορών ανάμεσα σε δύο εικόνες υπάρχουν αρκετοί τρόποι. Εμείς χρησιμοποιούμε μία μέθοδο που έχει ως σκοπό την ταχύτητα αλλά και ταυτόχρονα τον υπολογισμό του κέντρου βάρους της εικόνας. Έτσι

μπορούμε, ξέροντας το κέντρο βάρος της διαφοράς των εικόνων, να δώσουμε και την κατάλληλη εντολή στην κάμερα για να κινηθεί και να κεντράρει το θέμα μας.

Αρχικά έχουμε δύο εικόνες, μεγέθους 640X480 πίξελ. Επειδή είναι πολύ δύσκολο και χρονοβόρο να σκανάρουμε και να συγκρίνουμε 307200 πίξελ, κάνουμε αλλαγή του μεγέθους των εικόνων, σε 3X3 πίξελ.

Έτσι τώρα έχουμε δύο εικόνες με 9 συνολικά πίξελ η κάθε μία. Παίρνουμε τις τιμές για το κάθε βασικό χρώμα (Κόκκινο, Πράσινο, Μπλε) RGB και τις αφαιρούμε και παίρνουμε την απόλυτη τιμή της αφαίρεσης. Άρα έχουμε σε έναν πίνακα 3X3 στοιχείων, τις διαφορές στα χρώματα ανάμεσα στις δύο αρχικές εικόνας.

Μέσα από μία απλή δομή ακολουθίας ελέγχουμε τους αριθμούς των διαφορών, αν είναι μικρότερα ή μεγαλύτερα από κάποιο προκαθορισμένο αριθμό που έχουμε θέσει ως βάση για την εύρεση διαφοράς. Αφού σκαναριστεί όλη η εικόνα, έχουμε κρατήσει το στοιχείο του πίνακα με την μεγαλύτερη διαφορά και άρα έχουμε και το κέντρο βάρους, που ανάλογα τις συντεταγμένες του, δίνουμε και την εντολή στην κάμερα για να γυρίσει και στους δύο άξονες και να κεντράρει το θέμα μας.

Κεφάλαιο

4

4 IP Κάμερες

Η παρακολούθηση ενός χώρου μέσω IP βίντεο μπορεί να οριστεί ως το σύστημα που χρησιμοποιεί τα ελεύθερα πρωτόκολλα του internet και τα στάνταρ με σκοπό την εγγραφή και την παρακολούθηση ενός χώρου. Οι IP κάμερες, δεν πρέπει να συγχέονται με τις κανονικές κάμερες, διότι ο σκοπός τους δεν είναι η ποιοτική καταγραφή και απεικόνιση ενός χώρου, αλλά ο έλεγχος του.

Μερικά από τα πολλά πλεονεκτήματα των συστημάτων ελέγχου και ανίχνευσης που στηρίζονται σε IP κάμερες είναι τα εξής:

- Μείωση του κόστος διότι χρησιμοποιούνται οι υπάρχουσες δικτυακές εγκαταστάσεις για να δουλέψουν οι IP κάμερες.
- Μείωση του κόστους πάλι λόγω της χρήσης του CAT5 καλωδίου έναντι του RG-58 ομοαξονικού καλωδίου.
- Λόγω της ανοιχτής πλατφόρμας υπάρχουν στην αγορά πολλά δωρεάν προγράμματα διαχείρισης IP καμερών.

Εικόνα 8: Μία τυπική IP κάμερα εξωτερικού χώρου

- Ευελιξία όσο αφορά τα στάνταρ της εικόνας, δεν υπάρχει εδώ το πρόβλημα με τις αναλύσεις των κλασικών καμερών που είναι είτε NTSC

είτε PAL είτε SECAM.

- Οι εντολές μετακίνησης του μηχανισμού της κάμερας (άξονες XY & Ζουμ) μεταφέρονται μέσα από το ίδιο καλώδιο και δεν χρειάζονται ξεχωριστά κυκλώματα ελέγχου και μετάδοσης της εντολής.
- Πολλές IP κάμερες περιέχουν δικό τους λογισμικό που μεταφέρει τις "ύποπτες" εικόνες μέσω ηλεκτρονικού ταχυδρομείου αυτόματα και άμεσα.
- Υποστήριξη διαφόρων φορμά εικόνας και ήχου, για την κάλυψη όλων των αναγκών σε εύρος ζώνης δικτύου αλλά και σε αποθηκευτικό χώρο.
- Ενσωμάτωση προγραμμάτων ανίχνευσης κίνησης και αναγνώρισης/καταμέτρησης σχημάτων που εφαρμόζεται σε αντικείμενα, ανθρώπους και οχήματα.
- Συνεργασία του όλου συστήματος και με άλλα υποσυστήματα που ελέγχουν την εξουσιοδότηση εισόδου σε κτήρια, συναγερμούς, διαχείριση κίνησης στους δρόμους κτλ.
- Και τέλος, λόγω της δυνατότητας για αναβάθμιση του προγράμματος της κάμερας μέσω δικτύου, μελλοντικά δεν αποκλείονται και νέες εφαρμογές και δυνατότητες.

Σήμερα υπάρχουν αρκετοί κατασκευαστές IP καμερών καθώς και αμέτρητες εταιρίες κατασκευής λογισμικού για εγγραφή και έλεγχο των παραπάνω συστημάτων.

4.1 Λίγη Ιστορία

Η πρώτη IP κάμερα παρουσιάστηκε το 1996 από την Axis Communications, και χρησιμοποιούσε μία πλατφόρμα βασισμένη σε διανομή Linux που έτρεχε πάνω στην κάμερα. Η Axis επίσης παρουσίασε μαζί και ένα API με το όνομα "VAPIX" το οποίο βασιζόταν πάνω στα ελεύθερα στάνταρ του http και του rtsp. Αυτή η κίνηση παρακίνησε και τρίτες εταιρίες λογισμικού να αναπτύξουν προγράμματα για την διαχείριση και την εγγραφή βίντεο από τις IP κάμερες, πράγμα το οποίο και έγινε και μέσα σε ελάχιστα χρόνια

είχαμε εκθετική αύξηση των κλειστών κυκλωμάτων τηλεόρασης που βασίζοντας πάνω σε αυτές τις IP κάμερες.

4.2 Κινήσεις κάμερας

Μία σύγχρονη IP κάμερα μπορεί να κάνει μέχρι και 3 κινήσεις.

- Pan: κίνηση του όλου κορμού της κάμερας δεξιόστροφα ή αριστερόστροφα συνήθως +-175 μοίρες από την αρχική θέση. Οι δέκα μοίρες που φαινομενικά φαίνονται να είναι νεκρό σημείο στο οπτικό πεδίο της κάμερας στην πραγματικότητα καλύπτονται από τις δύο ακραίες θέσεις, απλά για λόγους κατασκευαστικούς δεν γίνεται εύκολα η κάμερα να μετακινείτε 360 μοίρες.

- Tilt: κίνηση της κάμερας πάνω κάτω. Συνήθως η ωφέλιμη γωνία κίνησης είναι από 90 μοίρες και πάνω, έτσι ώστε να μπορεί μία κάμερα να έχει μεγάλο οπτικό πεδίο.

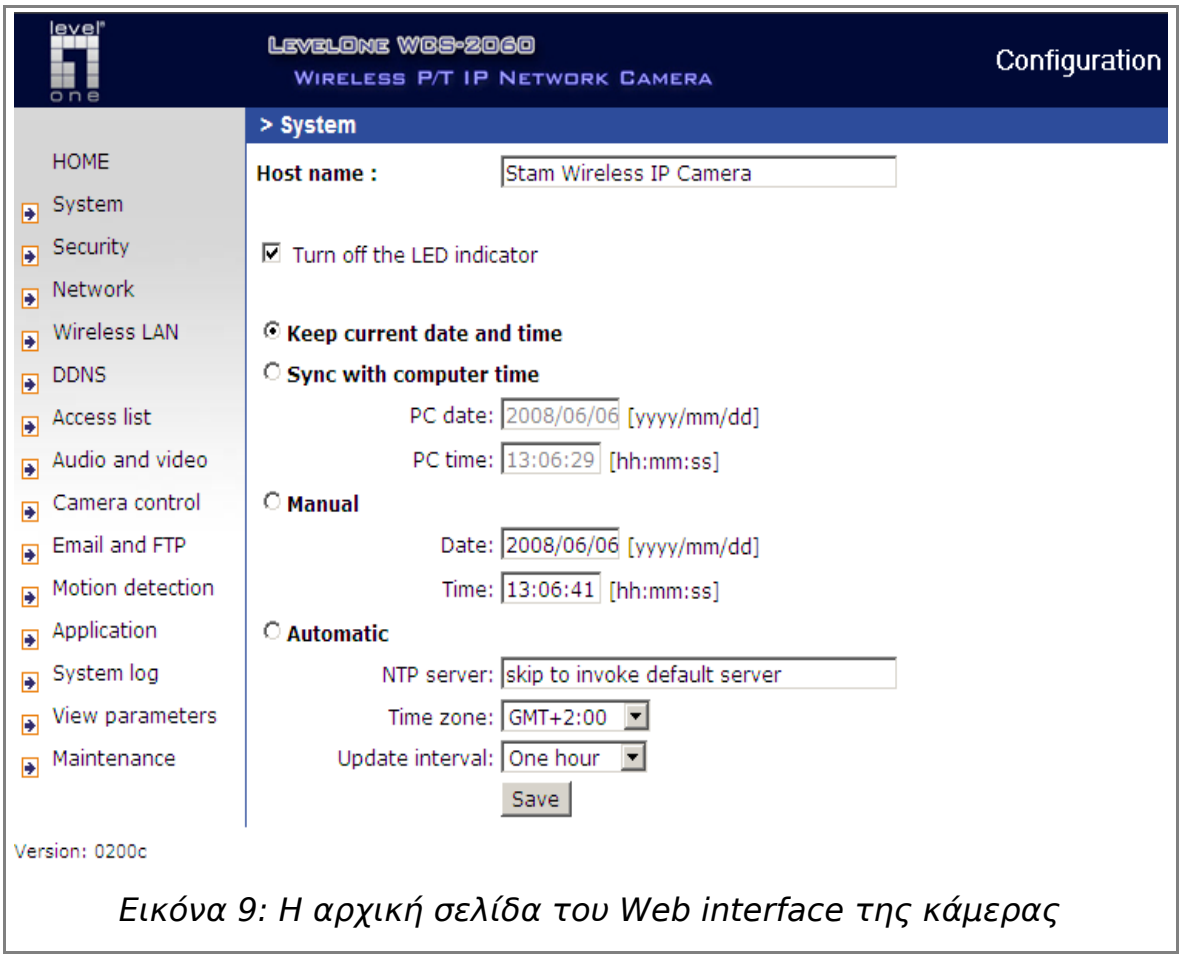
- Zoom: είτε ψηφιακό είτε αναλογικό, με το αναλογικό να είναι πολύ καλύτερο σε ποιότητα, μπορεί να γίνει κίνηση του φακού της κάμερας και έτσι να αλλάξει στο εστιακό σημείο της και να επικεντρωθεί σε ένα συγκεκριμένο μέρος για καλύτερη παρατήρηση και καταγραφή.

4.3 Web Interface

Ένα σημαντικό χαρακτηριστικό της κάμερας, που κάνει την χρήση της, και ειδικότερα την ρύθμιση της εύκολη, είναι το web interface που ενσωματώνει. Μέσω αυτού μπορούμε με μερικά μόνο κλικ από τον internet browser του υπολογιστή μας, ανεξάρτητα από το λειτουργικό σύστημα που έχουμε (Windows, Mac, Linux) να την ρυθμίσουμε καθώς και να την βάλουμε σε λειτουργία.

Μερικά από τα πράγματα που μπορούμε να ρυθμίσουμε είναι η ώρα της

κάμερας, η ασφάλεια της με την δημιουργία χρηστών για την πρόσβαση στην χρήση της κάμερας. Ακόμα μπορούμε να ρυθμίσουμε τις διευθύνσεις IP που θα παίρνει η κάμερα, το αν θα συνδέεται ασύρματα σε κάποιο access point, την ποιότητα και πολλές άλλες ρυθμίσεις στην εικόνα και τον ήχο, την κίνηση της κάμερας καθώς και ο ορισμός κάποιων θέσεων που θεωρούμε σημαντικές. Επίσης ρύθμιση για τον mailserver, ενσωματωμένο ανιχνευτή κίνησης, ημερολογιακή οργάνωση για την λήψη φωτογραφιών καθώς και κάποιες ρυθμίσεις στο firmware της κάμερας.



Εικόνα 9: Η αρχική σελίδα του Web interface της κάμερας

4.4 Level One WCS-2060

Η IP Camera που χρησιμοποιήθηκε για την πτυχιακή είναι η WCS-2060 της Level One. Είναι μια wireless ip camera που επιτρέπει κινήσεις pan & tilt και προσφέρει υψηλής ποιότητας εικόνα και έχει βελτιστοποιημένο εύρος

ζώνης μέσω της υπάρχουσας σύνδεσης δικτύου.

Η παραπάνω κάμερα προορίζεται κυρίως για εσωτερική χρήση, ή για χρήση σε χώρους που σε περίπτωση βροχής δεν θα υπάρχει πρόσβαση του νερού. Βέβαια με την ειδική βάση/κουβούκλιο που πουλάει η ίδια εταιρία, μπορεί να γίνει αδιάβροχη, και να τοποθετηθεί σε οποιοδήποτε εξωτερικό σημείο θέλουμε να επιβλέψουμε τον χώρο.

Η τροφοδοσία της γίνεται μέσω ενός μετασχηματιστή που παρέχετε με την συσκευασία και μετασχηματίζει το ρεύμα της πρίζας του δικτύου από 220V/50HZ σε 12VDC. Η λειτουργία της με 12VDC την κάνει ιδιαίτερα ευέλικτη διότι δίνεται να λειτουργήσει σε οποιοδήποτε περιβάλλον με ένα απλό κύκλωμα σταθεροποιητή τάσης το οποίο θα περιέχει ως κύριο ολοκληρωμένο το 7812 το οποίο στην έξοδο του παρέχει 12VDC & 2A.

Τα χαρακτηριστικά της κάμερας είναι τα παρακάτω:

- **Πρότυπα: IEEE 802.3 10BaseT Ethernet. IEEE 802.3u 100Base-TX Fast Ethernet. IEEE 802.11 b/g.**
- **Θύρες:** 1 x RJ-45 10/100 Mbps.
- **Συχνότητα λειτουργίας:** 2.4~ 2.4835GHz.
- **Gain κεραίας:** 2 dBi.
- **Ισχύς εξόδου RF:** 64QAM: 14dBm; 16QAM: 15dBm; QPSK: 16dBm; BPSK: 17dBm; DBPSK, DQPSK, CCK: 17dBm.
- **Πρωτόκολλα:** UPnP, TCP/IP, RTSP/RTP/RTCP, HTTP, SMTP, FTP, Telnet, NTP, DNS, DHCP, DDNS και PPPoE.
- **Αλγόριθμοι βίντεο:** MPEG4 (Απλό προφίλ) για streaming βίντεο. JPEG για φωτογραφίες.
- **Χαρακτηριστικά εικόνας:** Προσαρμοζόμενο μέγεθος εικόνας, ποιότητα και bit rate. Timestamp και text overlay. 3 motion detection windows. Flip & mirror.
- **Ανάλυση βίντεο:** Μέχρι και 30/25 frames σε 160x120. Μέχρι και 30/25 frames σε 176x144. Μέχρι και 30/25 frames σε 320x240. Μέχρι και 15/12 frames σε 640x480.
- **Κάμερα/ Φακός:** Έγχρωμος αισθητήρας CCD 1/4 ιντσών. Τυπικό: 1.0 Lux / F2.0. AGC, AWB, AES. Ηλεκτρονικό shutter: 1/60 to 1/15,000

sec. Εστιακό μήκος: 4.0mm. Aperture: F2.0.

- **Αλγόριθμος ήχου:** GSM-AMR/ MPEG4 AAC.
- **Μικρόφωνο:** Omni-directional. Συχνότητα: 50Hz έως 16,000Hz. S/N ratio: πάνω από 60dB.
- **Pan/Tilt:** Pan: πεδίο 350 (+175~-175). Tilt: πεδίο 125 (+90~-35). Λειτουργία Auto pan/Auto patrol.
- **Ασφάλεια:** ID/Password authentication. Ασφάλεια Wireless: 64/128-bit WEP; WPA-PSK.
- **Μνήμη:** RAM: 32MB SDRAM. ROM: 4MB Flash ROM.
- **Ενδείξεις LED:** System power και status. System activity και Network link. Κατάστασης μικροφώνου.
- **Τροφοδοσία:** 12VDC, 1.5A.
- **Διαστάσεις:** (L x W x H) 138 x 88.9 x 57.2mm.
- **Βάρος:** 222g.

Ακόμα μπορεί να εκτελεί κάποιες προκαθορισμένες εργασίες, όπως να παίρνει φωτογραφίες κάθε όσο χρόνο της ορίσουμε και είτε να τις ανεβάζει σε έναν εξυπηρετητή αρχείων (FTP) είτε να τις αποστέλλει μέσω ηλεκτρονικού ταχυδρομείου.

Για μελλοντικές εφαρμογές μπορεί να γίνει και εκμετάλλευση του εσωτερικού μικροφώνου που περιέχει η κάμερα έτσι ώστε να ανιχνεύεται ως πιθανή είσοδος εισβολέα και να αρχίζει τον περιμετρικό έλεγχο του χώρου για την εύρεση του εισβολέα.

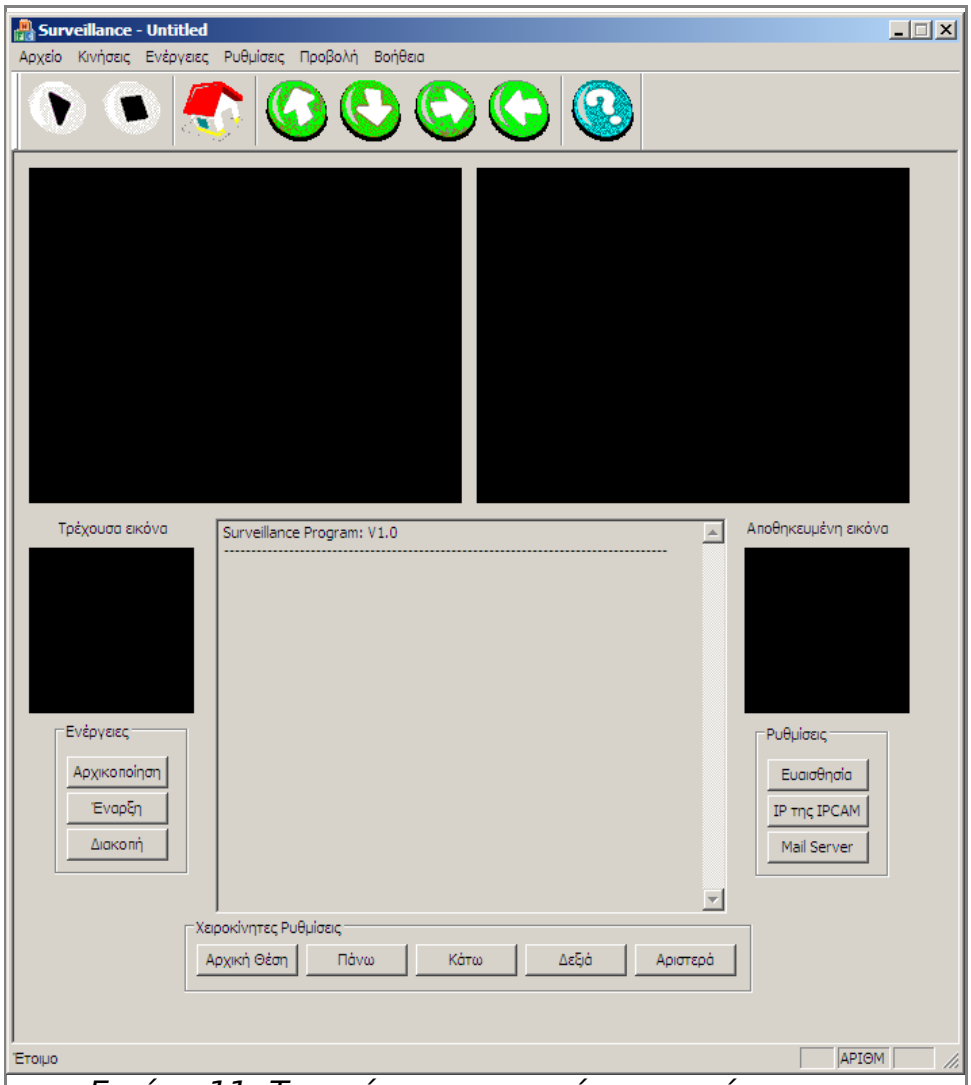
Εικόνα 10: Η IP κάμερα της εφαρμογής μας

Κεφάλαιο

5

5 Το πρόγραμμα

Σκοπός του προγράμματος είναι ο έλεγχος ενός εξωτερικού χώρου για εισβολέα και η ειδοποίηση του ιδιοκτήτη μέσω ηλεκτρονικού μηνύματος.

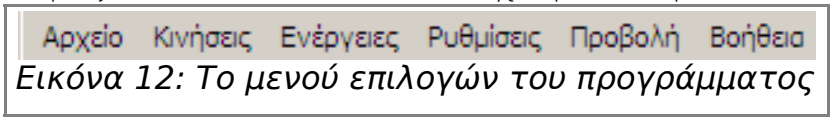


Εικόνα 11: Το πρόγραμμα κατά την εκκίνηση του

5.1 Παρουσίαση του προγράμματος

Το πρόγραμμα είναι γραμμένο σε Visual C++, χρησιμοποιήθηκαν επίσης οι κλάσεις MFC της Microsoft και το interface του προγράμματος είναι SDI (Single Document Interface).

Αρχίζοντας από πάνω προς τα κάτω, πάνω πάνω έχουμε ένα μενού, που έχει την κλασσική δομή που έχουν όλα τα μενού των εφαρμογών στα windows. Υπάρχουν εντολές για όλες τις λειτουργίες του προγράμματος καθώς και συντομεύσεις από το πληκτρολόγιο για άμεση εκτέλεση αυτών. Ποιο αναλυτικά έχουμε της εξής εντολές:



Εικόνα 12: Το μενού επιλογών του προγράμματος

Μενού	**Υπομενού**	**Εντολή**	**Συντόμευση**
Αρχείο	------->	Έξοδος	F12
Κινήσεις	------->	Αρχική Θέση	Ctrl+Shift+Home
Κινήσεις	Οριζόντιος άξονας	Δεξιά	Ctrl+Shift+Δεξιά
Κινήσεις	Οριζόντιος άξονας	Αριστερά	Ctrl+Shift+Αριστερά
Κινήσεις	Κάθετος άξονας	Πάνω	Ctrl+Shift+Πάνω
Κινήσεις	Κάθετος άξονας	Κάτω	Ctrl+Shift+Κάτω
Ενέργειες	------->	Αρχικοποίηση	
Ενέργειες	------->	Έναρξη	
Ενέργειες	------->	Διακοπή	
Ρυθμίσεις	------->	Ρύθμιση της ευαισθησίας	F4
Ρυθμίσεις	------->	Ρύθμιση της IP της κάμερας	F5
Ρυθμίσεις	------->	Ρύθμιση MailServer	F6
Προβολή	------->	Γραμμή εργαλείων	
Προβολή	------->	Γραμμή κατάστασης	
Βοήθεια	------->	Θέματα στη βοήθεια	F1
Βοήθεια	------->	Σχετικά με το Surveillance	Shift+F1

Ακριβώς από κάτω έχουμε μία μπάρα εργαλείων που έχει κουμπιά με εικόνες για την έναρξη/διακοπή της παρακολούθησης, κουμπιά για τον έλεγχο των κινήσεων της κάμερας, καθώς και ένα κουμπί για την επιστροφή της κάμερας στην αρχική θέση της.

Εικόνα 13: Η μπάρα εργαλείων

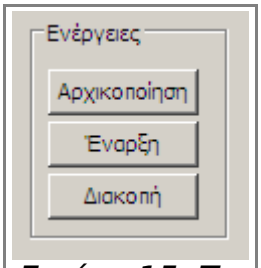
Πιο αναλυτικά, από αριστερά προς δεξιά υπάρχουν τα κουμπιά: Έναρξη, Διακοπή, Αρχική θέση, Πάνω, Κάτω, Δεξιά, Αριστερά, Σχετικά με το Surveillance.

Κάτω κάτω έχουμε μία γραμμή κατάστασης που εμφανίζει μηνύματα του προγράμματος ανάλογα με την κατάσταση του, καθώς και επεξηγήσεις για τις λειτουργίες του μενού και της μπάρας εργαλείων. Στην δεξιά μεριά της

Εικόνα 14: Η γραμμή κατάστασης

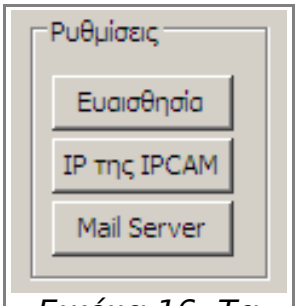
υπάρχουν ενδείξεις για το αν είναι αναμμένα τα Numlock, Capslock και Scrolllock.

Στο κύριο παράθυρο του προγράμματος αριστερά και δεξιά έχουμε δύο ομάδες κουμπιών.



Εικόνα 15: Τα κουμπιά των βασικών ενεργειών

Η πρώτη ομάδα ονομάζεται Ενέργειες και περιέχει τρία κουμπιά, Αρχικοποίηση, Έναρξη, Διακοπή. Από εκεί μπορούμε να κάνουμε αρχικοποίηση του προγράμματος, να αρχίσουμε την παρακολούθηση αλλά και να την διακόψουμε.

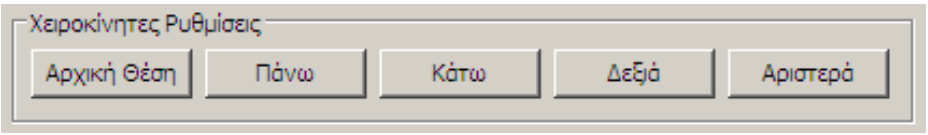


Εικόνα 16: Τα κουμπιά για τις ρυθμίσεις του προγράμματος

Η δεύτερη ομάδα ονομάζεται ρυθμίσεις και περιέχει και αυτή τρία κουμπιά, Ευαισθησία, IP της IPCAM και MailServer. Πατώντας στο καθένα από αυτά, τότε ανοίγει ένα παράθυρο για την αντίστοιχη ρύθμιση.

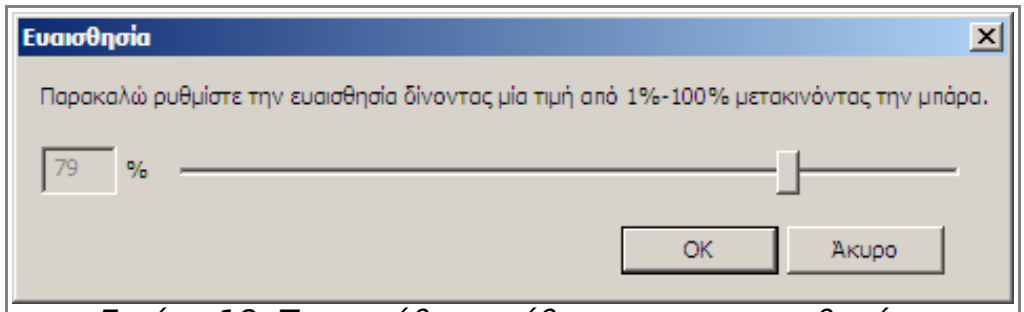
Τέλος υπάρχει άλλη μία ομάδα κουμπιών, η οποία αφορά της χειροκίνητες ρυθμίσεις της κάμερας. Αυτές αφορούν τον οριζόντιο και τον κάθετο άξονα. Πιο αναλυτικά τα πέντε κουμπιά είναι τα: Αρχική Θέση, Πάνω,

Κάτω, Δεξιά, Αριστερά.



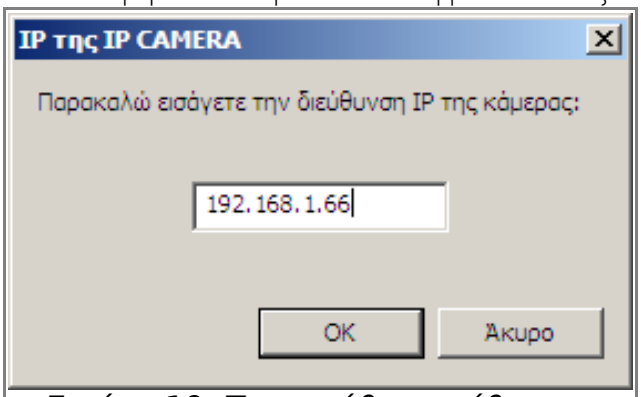
Εικόνα 17: Το κουμπιά για χειροκίνητη κίνηση της κάμερας

Τώρα θα δούμε αναλυτικά τα παράθυρα των ρυθμίσεων. Το πρώτο παράθυρο αφορά την ευαισθησία του προγράμματος όσον αφορά τον εντοπισμό πιθανού εισβολέα στο χώρο. Πατώντας στο κουμπί ευαισθησία, ή πατώντας F4, ανοίγει το παράθυρο. Σε αυτό το παράθυρο υπάρχει ένα slider, το οποίο το κινούμε αριστερά και δεξιά για να αυξομειώσουμε την ευαισθησία. Όσο πιο δεξιά τον μετακινούμε τόσο πιο ευαίσθητο γίνεται το σύστημα ενώ όσο πιο αριστερά τόσο λιγότερο ευαίσθητο στις αλλαγές της εικόνας. Το νούμερο της ευαισθησίας φαίνεται δίπλα από τον slider στην αριστερή του μεριά και εκφράζεται σε επί τοις εκατό ποσοστό.



Εικόνα 18: Τα παράθυρο ρύθμισης της ευαισθησίας

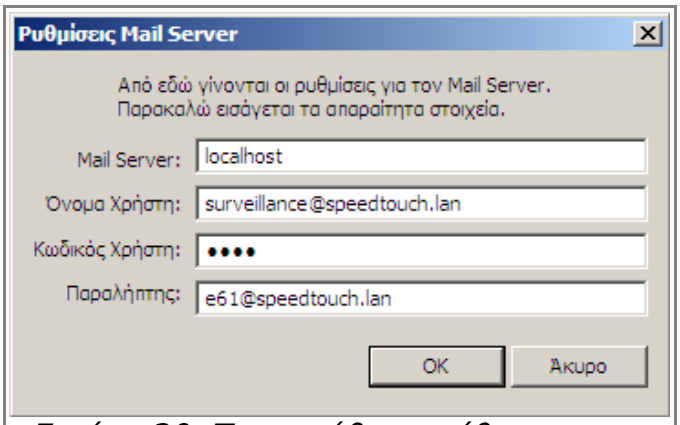
Το δεύτερο παράθυρο ρυθμίσεων αφορά την IP της κάμερας. Μπορούμε να το εμφανίσουμε είτε πηγαίνοντας από το μενού Ρυθμίσεις -> Ρυθμίσεις IP της IPCAM, είτε πατώντας F5. Μόλις ανοίξει το παράθυρο υπάρχει ένα χώρος στον οποίο μπορούμε να γράψουμε εμείς την διεύθυνση IP της κάμερας. Το πρόγραμμα με την έναρξη του, φορτώνει αυτόματα μία IP, αλλά αν χρειαστεί να την αλλάζουμε πρέπει να το κάνουμε μέσα από αυτό το παράθυρο.



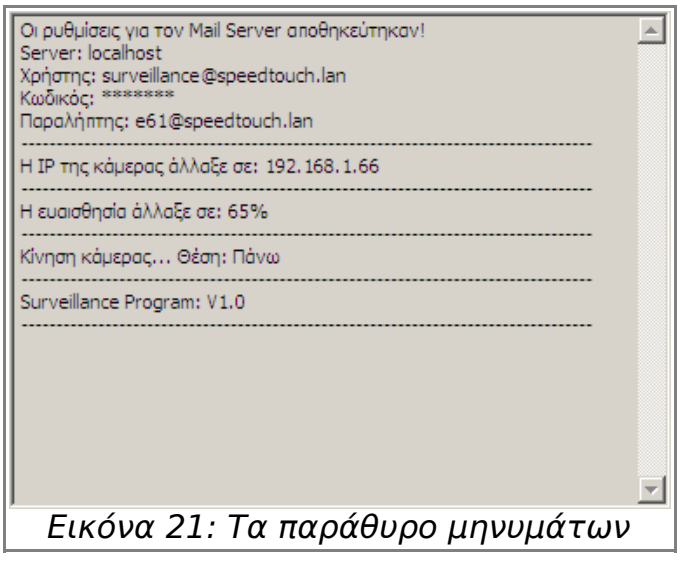
Εικόνα 19: Τα παράθυρο ρύθμισης διεύθυνσης IP της κάμερας

Το τρίτο και τελευταίο παράθυρο των ρυθμίσεων αφορά τις ρυθμίσεις που πρέπει να γίνουν όσων αφορά τον MailServer. Το πρόγραμμα για να μπορέσει να στείλει e-mail στον ιδιοκτήτη για να τον ειδοποιήσει για μία πιθανή εισβολή πρέπει να συνδεθεί με έναν MailServer. Έτσι πατάμε Ρυθμίσεις-> Ρυθμίσεις MailServer, ή F6 και ανοίγει το παράθυρο για να εισάγουμε της απαραίτητες πληροφορίες.

Πιο αναλυτικά στο πρώτο πεδίο εισάγουμε την διεύθυνση του διακομιστή ηλεκτρονικού ταχυδρομείου. Μετά στα δύο επόμενα πεδία εισάγουμε το όνομα χρήστη και τον κωδικό που έχουμε για τον συγκεκριμένο διακομιστή ηλεκτρονικού ταχυδρομείου έτσι ώστε να μπορέσει το πρόγραμμα να συνδεθεί στον διακομιστεί και να στείλει το ηλεκτρονικό μήνυμα. Τέλος στο τελευταίο πεδίο εισάγουμε την διεύθυνση που θέλουμε να πηγαίνουν τα ηλεκτρονικά μηνύματα όταν έχουμε πιθανή παραβίαση του χώρου.



Εικόνα 20: Τα παράθυρο ρύθμισης για τον MailServer



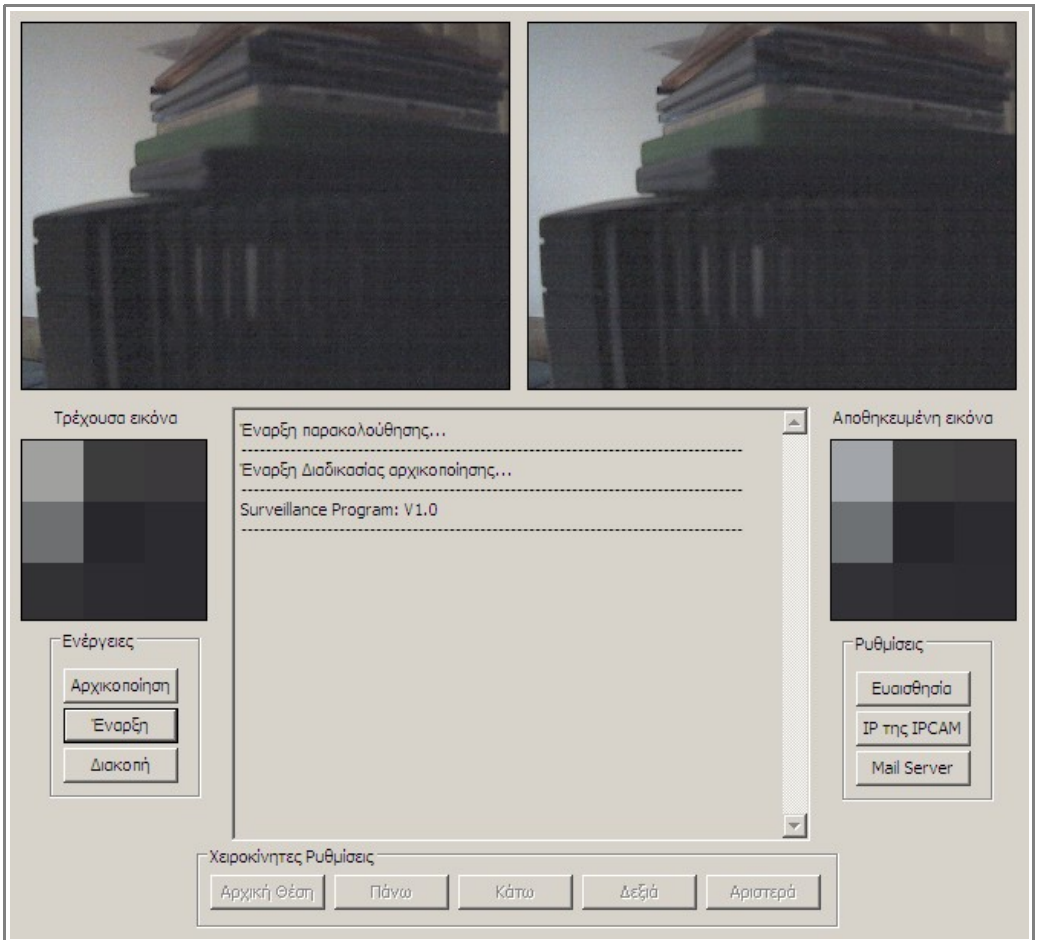
Εικόνα 21: Τα παράθυρο μηνυμάτων

Τώρα ανάμεσα στις δύο ομάδες κουμπιών, στο κεντρικό παράθυρο του προγράμματος, υπάρχει ένας χώρος, στο οποίο εμφανίζονται διάφορα μηνύματα που αφορούν διάφορες ενέργειες που έχουν γίνει στο πρόγραμμα, όπως αλλαγές ρυθμίσεων, κινήσεις της κάμερας κτλ.

Τώρα στο κύριο παράθυρο του προγράμματος επίσης παρατηρούμε τέσσερις μαύρους χώρους σχεδίασης. Αυτοί οι χώροι είναι για τον σχεδιασμό

και την απεικόνιση του χώρου που φωτογραφίζει η κάμερα.

Πιο συγκεκριμένα οι χώροι αυτοί μπορούν να χωριστούν σε δύο ομάδες. Η δεξιά ομάδα, απεικονίζει τις εικόνες που έχει αποθηκεύσει το πρόγραμμα στον υπολογιστή, από την διαδικασία της αρχικοποίησης που εκτελούμε κάθε φορά που ξεκινάμε το πρόγραμμα. Αντίστοιχα, στην αριστερή ομάδα απεικονίζονται οι εικόνες που στοχεύει αυτή την στιγμή η κάμερα.



Εικόνα 22: Το πρόγραμμα σε πλήρη λειτουργία

Τώρα μπορούμε αυτούς τους τέσσερις χώρους σχεδίασης να τους χωρίσουμε σε άλλες δύο ομάδες. Οι δύο πάνω εικόνες, απεικονίζουν της εικόνες στο φυσικό τους μέγεθος. Το μέγεθος συγκεκριμένα είναι 640X480 εικονοστοιχεία. Οι δύο κάτω εικόνες απεικονίζουν τις δύο πάνω εικόνες αντίστοιχα, αλλά σε μικρότερο μέγεθος, 3X3 εικονοστοιχεία.

5.2 Λειτουργία του προγράμματος

Το πρόγραμμα κατά την εκκίνηση του βρίσκεται σε κατάσταση αναμονής, η κάμερα δεν λειτουργεί, ούτε είναι έτοιμη για παρακολούθηση.

Πρώτη κίνηση που πρέπει να κάνουμε είναι να πατήσουμε στο κουμπί αρχικοποίηση.

Τότε η κάμερα θα γυρίσει στην τέρμα αριστερά θέση στον οριζόντιο άξονα, και στην πάνω θέση στον κατακόρυφο άξονα. Μετά διαδοχικά η κάμερα θα κάνει 13 κινήσεις μέχρι την τέρμα δεξιά θέση τραβώντας ταυτόχρονα και φωτογραφίες του χώρου. Μετά αμέσως θα γυρίσει στην αρχική θέση και θα έρθει στην κανονική θέση στον κάθετο άξονα και θα συνεχίσει πάλι την περιμετρική φωτογράφιση του χώρου. Τέλος η ίδια διαδικασία θα γίνει για να φωτογραφηθεί και ο χώρος στην τέρμα κάτω θέση στον κατακόρυφο άξονα. Έτσι με λίγα λόγια με την αρχικοποίηση έχουμε φωτογραφίες όλου του χώρου.

Μετά για να αρχίσει ο έλεγχος πρέπει να πατήσουμε έναρξη. Με το που πατάμε έναρξη αρχίζει η διαδικασία ελέγχου του χώρου περιμετρικά. Η κάμερα γυρνάει διαδοχικά ανάμεσα στις δύο ακραίες θέσεις, τραβώντας νέες φωτογραφίες και τις συγκρίνει με τις φωτογραφίες της αρχικοποίησης. Για όσο δεν διαπιστώνει το πρόγραμμα σοβαρή διαφορά η κάμερα συνεχίζει τον περιμετρικό έλεγχο.

Μόλις διαπιστωθεί σημαντική διαφορά ανάλογα στις δύο φωτογραφίες, αυτόματα υπολογίζετε το κέντρο βάρους του εισβολέα στην εικόνα, και δίνεται η αντίστοιχη εντολή στην κάμερα για το κεντράρισμα του εισβολέα στην εικόνα. Ταυτόχρονα, για όσο χρόνο το πρόγραμμα βρίσκεται σε κατάσταση συναγερμού, σώζονται όλες οι φωτογραφίες στο σκληρό δίσκο του υπολογιστή.

Μόλις τελειώσει η παρακολούθηση του εισβολέα, το πρόγραμμα στέλνει αμέσως μήνυμα μέσω ηλεκτρονικού ταχυδρομείου με κάποιο μήνυμα, που προειδοποιεί τον ιδιοκτήτη ότι υπάρχει πιθανή παραβίαση του χώρου και επισυνάπτει και μερικές φωτογραφίες.

Τέλος για να σταματήσει η όλη διαδικασία του ελέγχου του χώρου

πρέπει να πατήσουμε στο κουμπί διακοπή. Υπάρχουν ακόμα διάφορες ρυθμίσεις που γίνονται μέσα από παράθυρα, καθώς και κουμπιά για χειροκίνητο χειρισμό της κάμερας.

5.3 Ρουτίνες

Εδώ θα δούμε αναλυτικά την χρησιμότητα καθώς και την λειτουργία της κάθε ρουτίνας που περιλαμβάνει το πρόγραμμα. Ο κώδικας της κάθε ρουτίνας καθώς και ο πλήρες κώδικας παρατίθεται στο τέλος.

5.3.1 OnInitialUpdate()

Αυτή είναι η πρώτη ρουτίνα του προγράμματος που εκτελείται με την έναρξη του.

Πρώτα απόλα δημιουργούμε τους τέσσερις χώρους που θα εμφανίζονται οι μικρές αλλά και οι μεγάλες εικόνες. Η δημιουργία γίνεται δημιουργώντας τέσσερα παράθυρα με τις διαστάσεις που θέλουμε καλώντας την συνάρτηση Create. Μετά δημιουργούμε τέσσερα συμβατά DC πάνω στα οποία θα εμφανίσουμε τις εικόνες που επιθυμούμε.

5.3.2 MoveCamera()

Αυτή η ρουτίνα μας βοηθάει να κινήσουμε την κάμερα προς την κατεύθυνση που επιθυμούμε. Ανάλογα με το που θέλουμε να κινηθεί η κάμερα, έχουμε δύο μεταβλητές, την posx και την posy, οι οποίες παίρνουν τις αντίστοιχες τιμές. Αυτές οι τιμές είναι από 1 έως 13 για την posx και από 0 έως 2 για την posy.

Μετά καλώντας την διεύθυνση από το interface τις κάμερας, μαζί με την

κατάλληλη παράμετρος γίνεται η κίνηση της κάμερας προς την επιθυμητή θέση.

Για να αποφευχθεί η κίνηση της κάμερας στις ακραίες θέσεις και για προστασία των βηματικών μοτέρ της, γίνεται ένας έλεγχος και για τον κάθετο και για τον οριζόντιο άξονα, ώστε αν η κάμερα βρίσκεται ήδη σε μία από τις τέσσερις ακραίες θέσεις τότε η ρουτίνα σταματάει να εκτελείτε χωρίς να γίνει κάποια επιβλαβή κίνηση της κάμερας.

5.3.3 SetCameraOption()

Εδώ η ρουτίνα SetCameraOption μας βοηθάει να περάσουμε κάποιες ρυθμίσεις στον επεξεργαστή της κάμερας. Κάποιες από αυτές τις ρυθμίσεις αφορούν το μέγεθος των εικόνων που μας επιστρέφει, την ποιότητα της εικόνας, τον φωτισμό, καθώς και για το αν θα είναι έγχρωμες ή ασπρόμαυρες.

Τέλος η κίνηση της κάμερας προς κάποιες προκαθορισμένες θέσεις από εμάς γίνεται από αυτή τη ρουτίνα και όχι από την MoveCamera. Έτσι αν θέλουμε η κάμερα να κινηθεί τέρμα δεξιά, θα δοθεί η αντίστοιχη εντολή και θα ενημερωθούν και οι μεταβλητές θέσεις που έχουμε posx και posy.

5.3.4 GetBitmap()

Μέσω της GetBitmap μπορούμε να πάρουμε την τρέχουσα εικόνα από την κάμερα, από μία διεύθυνση του web interface της κάμερας, να την αποθηκεύσουμε προσωρινά σε έναν καταχωρητή, και αν αποθηκευτεί σωστά στην προσωρινή μνήμη, τότε να την πάρουμε από εκεί και να την αντιγράψουμε σε ένα αρχείο στον υπολογιστή μας.

Το όνομα του αρχείου που θα γίνει η αποθήκευση του αρχείου ορίζεται ως παράμετρος στην συνάρτηση καθώς και η διαδρομή που θα γίνει αποθήκευση, πχ GetBitmap(“C:\folder\image.jpg”).

5.3.5 InitBitmaps()

Αυτή είναι η ρουτίνα που εκτελείται κατά την αρχικοποίηση του προγράμματος. Υπάρχουν δύο διαδοχικές δομές ακολουθίας, η μία από 0 έως 2 και η άλλη από 1 έως 13.

Κατά την εκτέλεση της ρουτίνας, η κάμερα μετακινείτε από την μία ακραία οριζόντια θέση διαδοχικά μέχρι την άλλη ακραία οριζόντια θέση και φωτογραφίζει τον χώρο καλώντας την ρουτίνα GetBitmap. Αφού φωτογραφίσει περιμετρικά των χώρο για ένα ύψος, τότε η διαδικασία επαναλαμβάνεται και για τις άλλες δύο κάθετες θέσεις.

Ως όνομα για τις εικόνες δίνεται το "Screenshot-Y-X.jpg" όπου Y είναι μία τιμή από 0 έως 2 και X μία τιμή από 1 έως 13. Οι τιμές αυτές αντιπροσωπεύουν την θέση της κάμερας για την κάθε φωτογραφία.

Ανάμεσα στην κάθε κίνηση της κάμερας, υπάρχει μία χρονοκαθυστέρηση της τάξης του 1,5 δευτερολέπτου, για να προλαβαίνει η κάμερα να μετακινείτε πριν γίνεται η αποθήκευση της φωτογραφίας, καθώς και για να προσαρμόζεται ο φακός στις τρέχουσες συνθήκες φωτισμού του χώρου.

Μετά το τέλος της αρχικοποίησης η κάμερα επιστρέφει στην μία ακραία θέση για να είναι έτοιμη για την έναρξη της παρακολούθησης.

5.3.6 LoadPosBitmap()

Ο σκοπός αυτής της ρουτίνας είναι το φόρτωμα της αποθηκευμένης εικόνας από το σκληρό μας δίσκο από την τρέχουσα θέση της κάμερας. Την τρέχουσα θέση της κάμερας την έχουμε αποθηκευμένη στις μεταβλητές (posx, posy).

Ταυτόχρονα φορτώνουμε σε δύο μεταβλητές με ονόματα cx και cy τις

διαστάσεις της αποθηκευμένης εικόνας και τέλος πηγαίνουμε και "ζωγραφίζουμε" την επιφάνεια dc με την αποθηκευμένη εικόνα από την τρέχουσα θέση της κάμερας.

5.3.7 Watch()

Αυτή η ρουτίνα εκτελείται για να γίνεται η περιμετρική κίνηση της κάμερας κατά την παρακολούθηση. Έχουμε μία μεταβλητή, την watch_pos, η οποία κάθε φορά αυξάνεται κατά ένα.

Για όσο η τιμή της μεταβλητής μας είναι μικρότερη ή και ίση με 12, τότε η κάμερα μας κινείτε αριστερόστροφα. Όταν η τιμή της μεταβλητής ξεπεράσει την τιμή 12 τότε η κάμερα κινείτε δεξιόστροφα. Τέλος αν πάρει την τιμή 24, η οποία αντιστοιχεί στις 360 μοίρες (περίπου) τότε μηδενίζεται η τιμή της μεταβλητής για να ξεκινήσει η περιμετρική κίνηση της κάμερας από την αρχή.

5.3.8 PrintMsg()

Η λειτουργία αυτής της ρουτίνας είναι να ενημερώνει με μηνύματα το χώρο ανάμεσα στις δύο μικρές εικόνες με μηνύματα ενημερωτικά για τον χρήστη.

Παίρνει σαν παράμετρος κείμενο, το οποίο το εμφανίζει με αντίστροφη σειρά (LIFO). Δηλαδή το κείμενο που εμφανίστηκε στην αρχή του προγράμματος είναι κάτω κάτω ενώ το κείμενο που στείλαμε για εμφάνιση στο τέλος εμφανίζεται πάνω πάνω.

5.3.9 OnBnClickedStart()

Με το που πατήσουμε το κουμπί έναρξη πρώτα απόλα κινούμε την κάμερα στην ακραία θέση και μηδενίζουμε την μεταβλητή watch_pos. Τότε ενεργοποιούμε μία χρονοκαθυστέρηση της τάξης των τεσσάρων δευτερολέπτων για να προλάβει η κάμερα να μεταφερθεί στην ακραία θέση.

Με την έναρξη επίσης απενεργοποιούμε τα κουμπιά για την χειροκίνητη κίνηση της κάμερας έτσι ώστε να αποφευχθούν λανθασμένες κινήσεις της κάμερας από εντολές που θα δώσει ο χρήστης ταυτόχρονα με την ρουτίνα κίνησης. Ακόμα δημιουργούμε την επιφάνεια dc πάνω στην οποία θα αποθηκεύουμε τα bitmap με τις εικόνες μας.

Τέλος ενεργοποιούμε έναν χρονομετρητή, ο οποίος θα εκτελείται κάθε δυόμιση δευτερόλεπτα και θα εκτελεί τον έλεγχο ανάμεσα στις δύο εικόνες. Ο χρόνος αυτός επιλέχτηκε μετά από χρονομέτρηση της όλης λειτουργίας του προγράμματος έτσι ώστε το πρόγραμμα να προλαβαίνει να στέλνει την εντολή της κίνησης, να παίρνει φωτογραφία από την τρέχουσα θέση, να κάνει την σύγκριση με την αποθηκευμένη εικόνα και να επιστρέφει για το αν υπάρχει εισβολέας ή όχι.

5.3.10 OnTimer()

Η εκτέλεση αυτής της ρουτίνας όπως είπαμε παραπάνω γίνεται κάθε δυόμιση δευτερόλεπτα. Πρώτα απόλα δημιουργούμε έξι dc τα οποία θα μας χρησιμεύσουν για την απεικόνιση αλλά και για την επεξεργασία των εικόνων. Αυτές οι εικόνες είναι τέσσερις που θα εμφανιστούν στο πρόγραμμα μας και δύο που θα υπάρχουν στην μνήμη του υπολογιστή. Τα δύο dc έχουν διαστάσεις 640X480 και τα άλλα δύο 120X120.

Αφού τα δημιουργήσουμε πάμε και "ζωγραφίζουμε" πάνω τους την εικόνα από το σκληρό δίσκο της τρέχουσας θέσης καθώς και την τρέχουσα εικόνα από την μνήμη του υπολογιστή της τρέχουσας θέσης πάνω στις επιφάνειες dc3 και dc4 στις διαστάσεις που έχουν τα παράθυρα που έχουμε δημιουργήσει.

Την ίδια δουλειά κάνουμε και για τα dc5 και dc6, στα οποία όμως εμφανίζουμε της δύο παραπάνω εικόνες όχι σε πλήρες μέγεθως, αλλά σμικρυμένες σε 3X3 εικονοστοιχεία.

Μετά ορίζουμε κάποιες βοηθητικές μεταβλητές που θα μας βοηθήσουν στον έλεγχο των εικόνων. Μερικές από αυτές είναι η dif, για την επιθυμητή τιμή της διαφοράς που θεωρούμε ως ανοχή, τις red,blue,green, οι οποίες θα κρατούν τις τιμές για το κάθε χρώμα, τις x1 και y1, οι οποίες θα κρατάνε την θέση προς τα που πρέπει να κινηθεί η κάμερα, την μεταβλητή intruder, η οποία παίρνει δύο τιμές και μπορεί να είναι είτε αληθείς είτε ψευδείς και τέλος η μεταβλητή sum, στην οποία θα αποθηκεύουμε την συνολική τιμή των διαφορών των τριών χρωμάτων των δύο εικόνων, αυτή από το σκληρό δίσκο και αυτή από την κάμερα.

Μετά αρχίζουμε τον έλεγχο για το κάθε εικονοστοιχείο των δύο εικόνων. Έτσι για κάθε X και Y από ένα σύνολο 3X3, αποθηκεύουμε στις μεταβλητές red, blue και green την απόλυτη τιμή της διαφοράς για κάθε ένα από τα τρία βασικά χρώματα ανάμεσα στην αποθηκευμένη εικόνα στον υπολογιστή και στην τρέχουσα εικόνα της κάμερας.

Μετά προσθέτουμε τις μεταβλητές red, blue και green στην μεταβλητή sum και την συγκρίνουμε με την τιμή της μεταβλητής dif που είναι η ανοχή που έχουμε δηλώσει για τις διαφορές. Αν η μεταβλητή sum είναι μεγαλύτερη, τότε σημαίνει ότι κάτι έχει αλλάξει στην εικόνα, αποθηκεύουμε την τρέχουσα θέση της κάμερας στις μεταβλητές x1,y1 και δηλώνουμε την μεταβλητή intruder ως αληθείς.

Τώρα αν η μεταβλητή intruder είναι ψευδείς, δίνουμε εντολή να κινηθεί η κάμερα στην επόμενη θέση παρακολούθησης. Αν είναι αληθείς, ανάλογα με την τιμή των x1,y1 δίνουμε εντολή για κατάλληλη κίνηση της κάμερας. Επίσης γίνεται ένας έλεγχος για να μην πάει η κάμερα να κινηθεί εκτός των προκαθορισμένων θέσεων.

Τέλος καταστρέφουμε όλες τις επιφάνειες dc που δημιουργήσαμε έτσι ώστε να μην σπαταλάμε μνήμη από το λειτουργικό μας σύστημα και κολλήσει αργότερα και δεν ανταποκρίνεται.

5.3.11 OnBnClickedStop()

Με το πάτημα του κουμπιού "Διακοπή" διακόπτουμε τον χρονομετρητή από το να μετράει και να εκτελεί την ρουτίνα OnTimer κάθε δυόμιση δευτερόλεπτα. Μετά επιστρέφουμε την κάμερα στην ακραία θέση και ενεργοποιούμε την χρήση των κουμπιών χειροκίνητου χειρισμού της κάμερας. Έπειτα μηδενίζουμε πάλι την μεταβλητή θέσης της κάμερας και τυπώνουμε και ένα μήνυμα ότι έγινε διακοπή της παρακολούθησης.

5.3.12 MailIt()

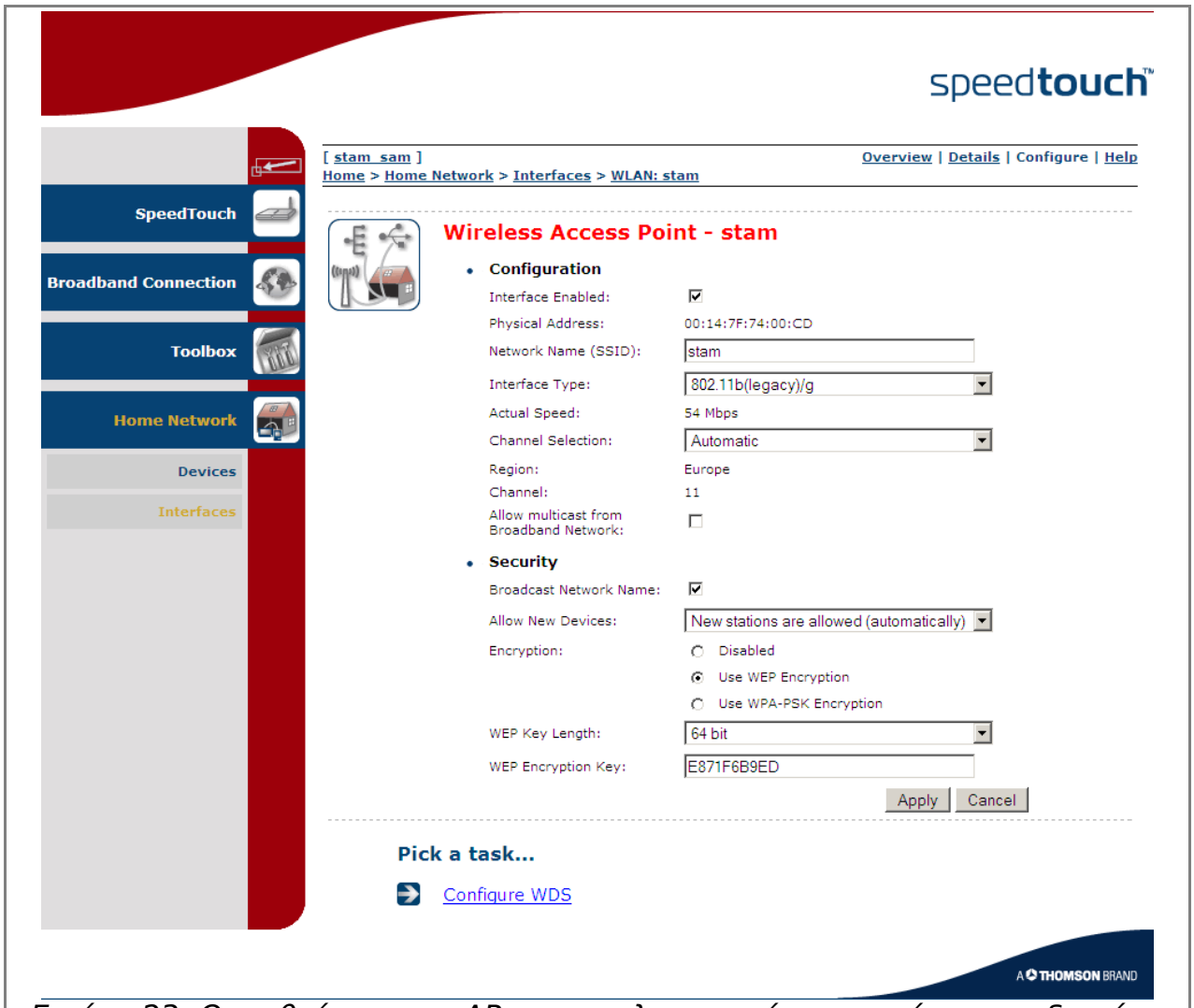
Όταν βρεθεί κάποιος εισβολέας, τότε καλείτε αυτή η ρουτίνα που έχει ως σκοπό να ενημερώνει τον ιδιοκτήτη του χώρου ότι υπάρχει πιθανή εισβολή. Με το που καλούμε αυτή την ρουτίνα, το πρόγραμμα σώζει την τρέχουσα εικόνα από την κάμερα και καλεί το πρόγραμμα commandLineEmailer.exe.

Με αυτό το πρόγραμμα, εφόσον δώσουμε κάποιες παραμέτρους που έχουμε βάλει στο παράθυρο Ρυθμίσεις Mailserver, αποστέλλετε το μήνυμα, μαζί με την φωτογραφία του χώρου, στην διεύθυνση email που έχουμε καθορίσει.

5.4 Ρυθμίσεις του Access Point και της κάμερας

Για την σωστή συνεργασία μεταξύ της κάμερας και του ασύρματου

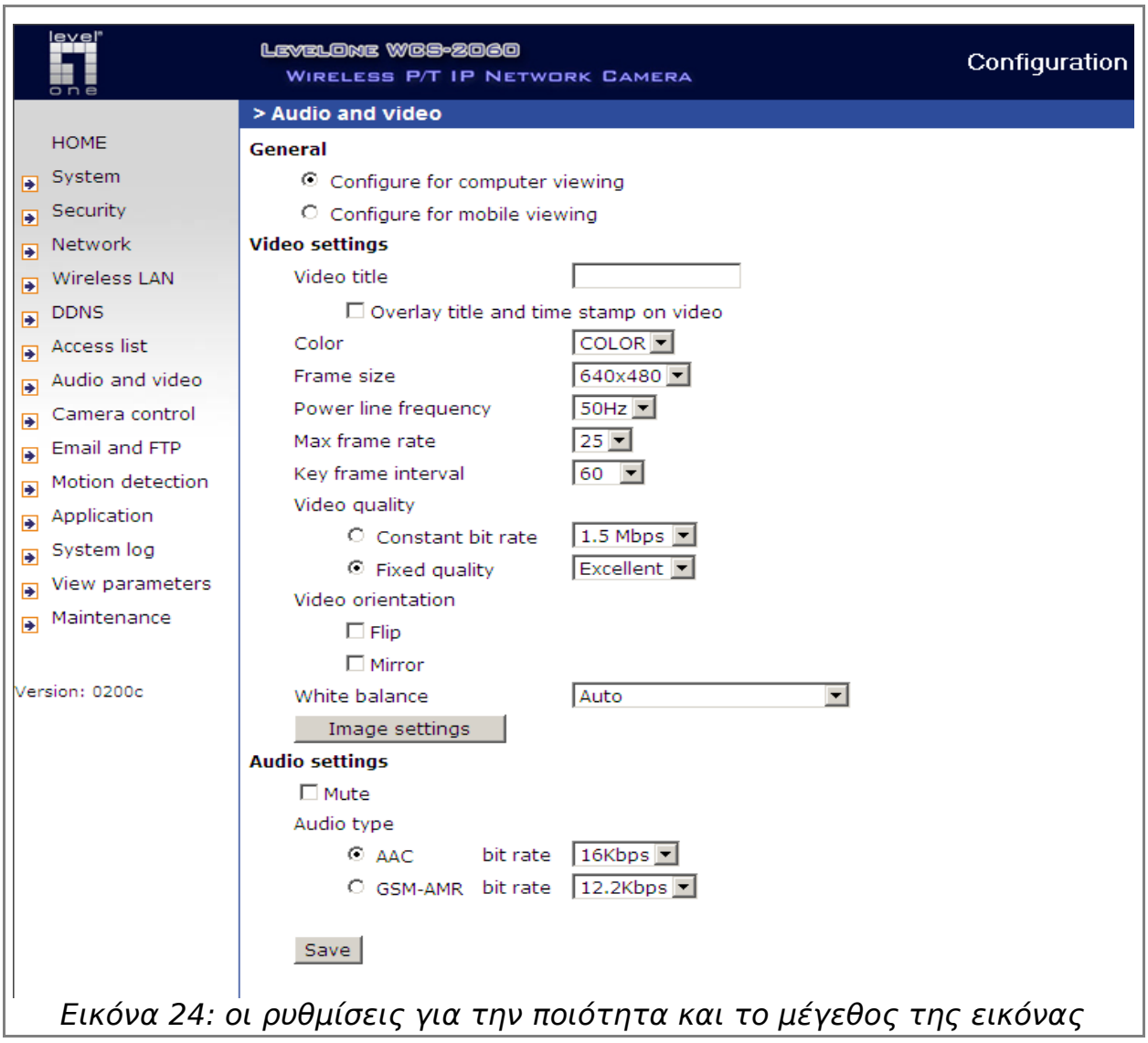
Access Point έγιναν κάποιες ρυθμίσεις. Αυτές αφορούσαν την σύνδεση με το ίντερνετ, έτσι ώστε το AP, που είναι και DSL μόντεμ, να μπορεί να συνδέεται κανονικά στο διαδίκτυο, απόδοση της ίδιας IP στην κάμερα, για να υπάρχει συμβατότητα με το πρόγραμμα, καθώς και ενεργοποίηση του ασύρματου δικτύου, με χρήση ασφάλειας WEP για ασφάλεια αλλά και ρύθμιση της



Εικόνα 23: Οι ρυθμίσεις στο AP για την λειτουργία του ασύρματου δικτύου καθώς και ο ορισμός του επίπεδου ασφαλείας

ταχύτητας του ασύρματου δικτύου στα 54Mbps.

Οι ρυθμίσεις για την κάμερα αφορούσαν την ρύθμιση για την απόδοση IP για συμβατότητα με το πρόγραμμα και το AP, ρύθμιση του μεγέθους των εικόνων σε 640X480, έγχρωμες και με φυσιολογική φωτεινότητα και αντίθεση, κλείδωμα του Web Interface έτσι ώστε να μην είναι προσβάσιμο από μη εξουσιοδοτημένους χρήστες, καθώς και ορισμός δύο θέσεων ως ακραίες θέσεις με τα ονόματα end_left και end_right.



Εικόνα 24: οι ρυθμίσεις για την ποιότητα και το μέγεθος της εικόνας

Βιβλιογραφικές πηγές

Για την συγγραφή της παρούσας πτυχιακής εργασίας έγινε χρήση διάφορων πηγών, είτε έντυπων είτε στο διαδίκτυο. Πιο αναλυτικά:

- Sams Teach Yourself Visual C++ in 21 Days , Davis Chapman, Macmillan Computer Publishing , USA, August 1998
- http://www.leveone.com
- http://www.functuonx.com/visualc/
- http://www.codeguru.com/cpp/
- http://www.codeproject.com
- http://en.wikipedia.org
- http://www.twmn.net
- http://www.ieee.org
- http://wi-fi.org/
- http://msdn.microsoft.com/

Φύλλα δεδομένων

Παρακάτω επισυνάπτονται τα φύλλα δεδομένων (data sheets) της ασύρματης IP κάμερας που χρησιμοποιήθηκε.

CamCon_WCS-2060

one world_one brand_one level_

11g Wireless P/T IP Network Camera

LevelOne WCS-2060 is a Pan, Tilt and Zoom wireless IP network camera, offering superior image quality and optimized bandwidth efficiency through existing network connection. It is able to broadcast live images to your handheld mobile phone or PDA and supports 3GPP/ISMA RSTP format via a 2.5G/3G mobile phone*.

Easy Access and Connection
In order to provide various flexible connection possibilities, this network camera can access control data and transmit video streaming either through a 54Mbps wireless network or an Ethernet connection. This camera is also easy to access as simple as surfing Web. Users can immediately use any of the features of this product simply by attaching the cables and turning on the power.

Pan/Tilt Network Camera
The FCS-1060 contains a CMOS sensor to record brilliant images up to 640 x 480 pixels and the camera presents users with a 4x digital zoom capability. The Auto Pan function allows the user to define individual cameras automatic panning movements thus providing the capability to survey the widest range of targets over a pan range of 350-degrees and a tilt range of 125-degrees. The Patrol mode allows the user to preset up to 20 spots for sequential auto-monitoring.

Remotely, Advanced Monitoring Features
The motion detection feature provides operational flexibility to record only when there is motion thus saving record storage. With a built-in web server, the WCS-2060 provides an easy user interface for remote access to receive video signals from anywhere anytime over the Internet with the Internet Explorer browser.

Easy Maintenance and Configuration
With LevelOne's free bundled software, administrators are able to monitor up to 32 channels simultaneously. The WCS-2060 also provides recording capabilities for synchronized video & audio when captured by event-driven or schedule recording modes. For full-featured surveillance solutions to your camera needs, our LevelOne WCS-2060 absolutely provides optimum solutions when building a digital surveillance environment.

** This feature depends on the local mobile telecom system and the mobile phone model. LevelOne does not guarantee that this feature will work in all countries.*

Related Products

FCS-1020
10/100 Mbps IP Web Camera

WCS-2010
11g Wireless P/T IP Network Camera

FCS-1060
10/100Mbps P/T IP Network Camera

Key Features

- Motorized wild-range Pan/Tilt.
- Built-in 1/4" CMOS sensor and high sensitive microphone to deliver high quality streaming video
- 802.11g Wireless Connection with WEP & WPA-PSK
- Provides 4x digital zoom
- Supports MPEG4 (Short header mode) compression for streaming video and JPEG for still Image
- Supports 3GPP/ISMA RTSP (Real Time Streaming Protocol)
- Intelligent Motion Detection
- Free-Bundled powerful 32-Channel Recording Software
- Built-in Web server for remote accessing image from general web browser anywhere
- Supports UPnP & DDNS

CamCon_WCS-2060

one world_one brand_one level_

Technical Specification

Networking

- **Standards Compliance**
  IEEE 802.3 10BaseT Ethernet
  IEEE 802.3u 100Base-TX Fast Ethernet
  IEEE 802.11 b/g
- **Ports**
  1 x RJ-45 10/100 Mbps port
- **Frequency**
  2.4~ 2.4835GHz
- **Antenna**
  Gain Value: 2 dBi
  RF Output Power: 64QAM: 14dBm; 16QAM: 15dBm; QPSK: 16dBm; BPSK: 17dBm; DBPSK, DQPSK, CCK: **17dBm**
- **Protocols**
  UPnP, TCP/IP, RTSP/RTP/RTCP, HTTP, SMTP, FTP, Telnet, NTP, DNS, DHCP, DDNS and PPPoE

Video Specification

- **Video Algorithm Support**
  MPEG4 (Simple Profile) for streaming video
  JPEG for still image
- **Features**
  Adjustable image size, quality and bit rate
  Timestamp and text overlay
  3 motion detection windows
  Flip & mirror
- **Video Resolution:**
  Up to 30/25 frames at 160x120
  Up to 30/25 frames at 176x144
  Up to 30/25 frames at 320x240
  Up to 30/25 frames at 640x480

Camera Specification

- **Camera/ Lens**
  1/4 inch color CMOS sensor
  Typical: 1.0 Lux / F2.0
  AGC, AWB, AES
  Electronic shutter: 1/60 to 1/15,000 sec
  Fix length: 4.0mm
  Aperture: F2.0

Audio

- **Algorithm**
  GSM-AMR/ MPEG4 AAC
- **Audio Mute**
  Supported

Microphone

Omni-directional
Frequency: 50 – 16,000Hz
S/N ratio: more than 60dB

Pan/Tilt

Pan: range 350° (+175°~-175°)
Tilt: range 125° (+90°~-35°)
Auto pan mode/ Auto patrol mode

Security

ID/Password authentication
Wireless Security: 64/128-bit WEP; WPA-PSK

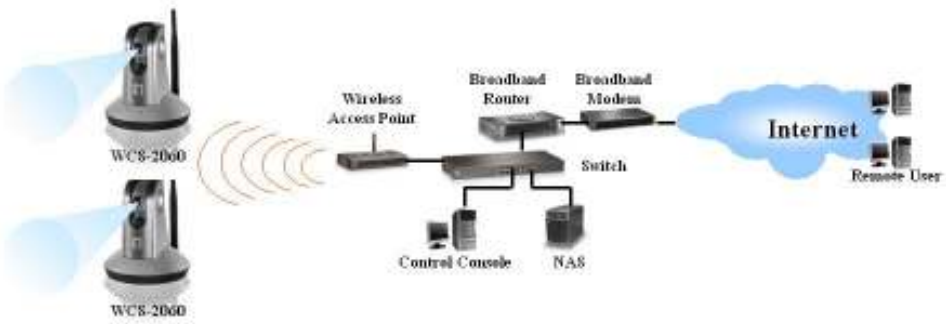
Hardware

- **System**
  RAM: 32MB SDRAM
  ROM: 4MB Flash ROM
- **LEDs**
  System power and status indicator
  System activity and Network link indicator
  Microphone status indicator
- **Power**
  Adapter output: 12VDC, 1.5A
- **Temperature**
  Operating: 0°C -40°C (32°F -102°F)
- **Humidity**
  Operating: 85% (RH)
- **Dimension**
  121.4mm(L) * 110mm(W) * 103mm(H)
- **Weight**
  222g
- **EMI and Safety**
  FCC, CE

Viewing System Requirements

- **OS**
  Microsoft Windows 98SE/ ME/ 2000/ XP
- **Browser**
  IE5.0 or above
- **3GPP Player**
  Real Player 10.5
  Quick Time 6.5
  Packet Video Player 3.0

Product Diagram



Ordering Information

- WCS-2060: 11g Wireless P/T IP Network Camera

www.level1.com

Κώδικας

Παρακάτω επισυνάπτονται τα αρχεία με το κώδικα γραμμένο σε visual c++ με σχόλια σε σημεία που κρίνεται απαραίτητο για την εύκολη κατανόηση του.

```
// SurveillanceView.cpp : implementation of the CSurveillanceView class
//

// Εδώ συμπεριλαμβάνουμε όλα τα απαραίτητα αρχεία που χρείαζεται
// το πρόγραμμα για να λειτουργήσει.
#include "stdafx.h"
#include <afxinet.h>
#include <atlimage.h>
#include "Surveillance.h"
#include "MailSettingsDlg.h"
#include "SensivityDlg.h"
#include "CameraIPDlg.h"
#include "DisplayWnd.h"
#include "DisplayWnd2.h"
#include "SurveillanceDoc.h"
#include "SurveillanceView.h"



#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CSurveillanceView

IMPLEMENT_DYNCREATE(CSurveillanceView, CFormView)

// Ο πίνακας με τα μηνύματα που λαμβάνει το πρόγραμμα και αντιδρά σε αυτά
BEGIN_MESSAGE_MAP(CSurveillanceView, CFormView)
	ON_BN_CLICKED(IDC_HOME, &CSurveillanceView::OnBnClickedHome)
	ON_BN_CLICKED(IDC_UP, &CSurveillanceView::OnBnClickedUp)
	ON_BN_CLICKED(IDC_RIGHT, &CSurveillanceView::OnBnClickedRight)
	ON_BN_CLICKED(IDC_DOWN, &CSurveillanceView::OnBnClickedDown)
	ON_BN_CLICKED(IDC_LEFT, &CSurveillanceView::OnBnClickedLeft)
	ON_BN_CLICKED(IDC_INITPICTURES,
&CSurveillanceView::OnBnClickedInitpictures)
	ON_BN_CLICKED(IDC_START, &CSurveillanceView::OnBnClickedStart)
	ON_BN_CLICKED(IDC_STOP, &CSurveillanceView::OnBnClickedStop)
	ON_BN_CLICKED(IDC_MAILSERVER,
&CSurveillanceView::OnBnClickedMailserver)
	ON_BN_CLICKED(IDC_SENSIVITY, &CSurveillanceView::OnBnClickedSensivity)
	ON_BN_CLICKED(IDC_CAMERAIP, &CSurveillanceView::OnBnClickedCameraip)
	ON_COMMAND(ID_SETTINGS_SENSIVITY,
&CSurveillanceView::OnSettingsSensivity)
	ON_COMMAND(ID_SETTINGS_CAMERAIP,
&CSurveillanceView::OnSettingsCameraip)
	ON_COMMAND(ID_SETTINGS_MAILSERVER,
&CSurveillanceView::OnSettingsMailserver)
	ON_COMMAND(ID_MOVES_RIGHT, &CSurveillanceView::OnMovesRight)
	ON_COMMAND(ID_MOVES_LEFT, &CSurveillanceView::OnMovesLeft)
	ON_COMMAND(ID_MOVES_UP, &CSurveillanceView::OnMovesUp)
	ON_COMMAND(ID_MOVES_DOWN, &CSurveillanceView::OnMovesDown)
	ON_COMMAND(ID_MOVES_HOME, &CSurveillanceView::OnMovesHome)
	ON_WM_TIMER()
	ON_COMMAND(ID_ACTIONS_INITPICTURES,
&CSurveillanceView::OnActionsInitpictures)
	ON_COMMAND(ID_ACTIONS_START, &CSurveillanceView::OnActionsStart)
	ON_COMMAND(ID_ACTIONS_STOP, &CSurveillanceView::OnActionsStop)
	ON_BN_CLICKED(IDC_FOLLOWME, &CSurveillanceView::OnBnClickedFollowme)
END_MESSAGE_MAP()



CSurveillanceView::CSurveillanceView()
	: CFormView(CSurveillanceView::IDD)
```

```
	, m_smessage(_T(""))
	, m_sInformations(_T(""))
	, m_sMessage(_T(""))
{
	// Εδώ δίνουμε κάποιες αρχικές τιμές σε μερικές μεταβλητές
	//m_sMessage=CString("Surveillance Program: V1.0");
	m_smessage=CString("");
	m_sInformations=CString("");
}

//Επιφάνειες που χρειαζόμαστε
HDC dc, dc1, dc2, dc3, dc4, dc5, dc6;
HBITMAP bit, bit1, bit2, bit3, bit4, bit5, bit6;

CSurveillanceView::~CSurveillanceView()
{

	// Παρακάτω διαγράφουμε όλες τις επιφάνειες dc
	// για να απελευθερώσουμε handler και να μην
	// κολλήσει όλο το λειτουργικό μας
	bit3=(HBITMAP)SelectObject(dc3,bit3);
	DeleteDC(dc3);
	DeleteObject(bit3);

	bit4=(HBITMAP)SelectObject(dc4,bit4);
	DeleteDC(dc4);
	DeleteObject(bit4);

	bit5=(HBITMAP)SelectObject(dc5,bit5);
	DeleteDC(dc5);
	DeleteObject(bit5);

	bit6=(HBITMAP)SelectObject(dc6,bit6);
	DeleteDC(dc6);
	DeleteObject(bit6);

	bit1=(HBITMAP)SelectObject(dc1,bit1);
	DeleteDC(dc1);
	DeleteObject(bit1);

	bit=(HBITMAP)SelectObject(dc,bit);
	DeleteDC(dc);
	DeleteObject(bit);

}

void CSurveillanceView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_HOME, m_cHome);
	DDX_Control(pDX, IDC_RIGHT, m_cRight);
	DDX_Control(pDX, IDC_LEFT, m_cLeft);
	DDX_Control(pDX, IDC_DOWN, m_cDown);
	DDX_Control(pDX, IDC_UP, m_cUp);
//	DDX_Control(pDX, IDC_INFORMATIONS, m_ctrl_informations);
}

BOOL CSurveillanceView::PreCreateWindow(CREATESTRUCT& cs)
{
	return CFormView::PreCreateWindow(cs);
}

// Εδώ δημιουργούμε τα αντικείμενα w,w2,win_small, win_small2, τα οποία
// θα μας χρησιμεύσουν ως χώροι για να εμφανίσουμε τις εικόνες
CDisplayWnd w;
CDisplayWnd2 w2;
```

```
CDisplayWnd win_small;
CDisplayWnd2 win_small2;
//Τρέχουσα θέση της κάμερας
int posx = 0;
int posy = 0;
//Μεγεθος εικόνων
int cx = 640;
int cy = 480;

// Εδώ βρίσκεται η ρουτίνα που εκτελείτε μόλις αρχίσει το πρόγραμμα μας
void CSurveillanceView::OnInitialUpdate()
{
	CFormView::OnInitialUpdate();
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit();

	// Απόδοση αρχικών τιμών στις απαραίτητες μεταβλητές για την λειτουργία
του mail server
	m_sMailServer=CString("localhost");
	m_sMailUser=CString("surveillance@speedtouch.lan");
	m_sMailPassword=CString("pass");
	m_sMailReciept=CString("e61@speedtouch.lan");

	//Απόδοση αρχικής IP για την κάμερα
	m_IP=CString("192.168.1.66");

	// Εμφάνιση ενός αρχικού μηνύματος
	PrintMsg(CString("Surveillance Program: V1.0\r\n"));

	// Τα παρακάτω τέσσερα σετ εντολών δημιουργούν τα 4 παράθυρα στα οποία
	// θα εμφανίσουμε τις εικόνες μας
	RECT r;
	GetDlgItem(IDC_STATICDISPLAY)->GetWindowRect(&r);
	ScreenToClient(&r);
	w.Create(NULL, NULL, WS_CHILD | WS_VISIBLE | WS_BORDER, r, this ,1001);

	RECT r2;
	GetDlgItem(IDC_STATICDISPLAY2)->GetWindowRect(&r2);
	ScreenToClient(&r2);
	w2.Create(NULL, NULL, WS_CHILD | WS_VISIBLE | WS_BORDER, r2, this ,
1002);

	RECT r3;
	GetDlgItem(IDC_STATICDISPLAY_SMALL)->GetWindowRect(&r3);
	ScreenToClient(&r3);
	win_small.Create(NULL, NULL, WS_CHILD | WS_VISIBLE | WS_BORDER, r3,
this ,1003);

	RECT r4;
	GetDlgItem(IDC_STATICDISPLAY_SMALL2)->GetWindowRect(&r4);
	ScreenToClient(&r4);
	win_small2.Create(NULL, NULL, WS_CHILD | WS_VISIBLE | WS_BORDER, r4,
this ,1004);

	// Παίρνουμε το dc της οθόνης
	HDC dc0 = ::GetDC(0);

	// Δημιουργούμε τα 4 dc για τους 4 χώρους εμφάνισης των εικόνων
	w.m_dc = CreateCompatibleDC(dc0);
	w2.m_dc2 = CreateCompatibleDC(dc0);
	win_small.m_dc_small = CreateCompatibleDC(dc0);
	win_small2.m_dc_small2 = CreateCompatibleDC(dc0);
	// Δημιουργούμε τις 4 επιφάνειες για τους 4 χώρους για την εμφάνιση των
εικόνων
	w.m_Bit = CreateCompatibleBitmap(dc0, 320, 240);
	w2.m_Bit2 = CreateCompatibleBitmap(dc0, 320, 240);
```

```
	win_small.m_Bit_small = CreateCompatibleBitmap(dc0, 120, 120);
	win_small2.m_Bit_small2 = CreateCompatibleBitmap(dc0, 120, 120);

	// Δημιουργούμε τα υπόλοιπα dc που θα χρειαστούμε στο πρόγραμμα
	dc = CreateCompatibleDC(dc0);
	bit = CreateCompatibleBitmap(dc0, cx, cy);
	dc1 = CreateCompatibleDC(dc0);
	bit1 = CreateCompatibleBitmap(dc0, cx, cy);

	dc3 = CreateCompatibleDC(dc0);
	bit3 = CreateCompatibleBitmap(dc0, 640, 480);
	dc4 = CreateCompatibleDC(dc0);
	bit4 = CreateCompatibleBitmap(dc0, 640, 480);
	dc5 = CreateCompatibleDC(dc0);
	bit5 = CreateCompatibleBitmap(dc0, 3, 3);
	dc6 = CreateCompatibleDC(dc0);
	bit6 = CreateCompatibleBitmap(dc0, 3, 3);

	// Απελευθερόνουμε το dc της οθόνης
	::ReleaseDC(0, dc0);

	bit = (HBITMAP)SelectObject(dc, bit);
	bit1 = (HBITMAP)SelectObject(dc1, bit1);
	bit3 = (HBITMAP)SelectObject(dc3, bit3);
	bit4 = (HBITMAP)SelectObject(dc4, bit4);
	bit5 = (HBITMAP)SelectObject(dc5, bit5);
	bit6 = (HBITMAP)SelectObject(dc6, bit6);
	w.m_Bit = (HBITMAP)SelectObject(w.m_dc, w.m_Bit);
	w2.m_Bit2 = (HBITMAP)SelectObject(w2.m_dc2, w2.m_Bit2);
	win_small.m_Bit_small = (HBITMAP)SelectObject(win_small.m_dc_small,
win_small.m_Bit_small);
	win_small2.m_Bit_small2 = (HBITMAP)SelectObject(win_small2.m_dc_small2,
win_small2.m_Bit_small2);

	// Κάνουμε τις ρυθμίσεις για το κάθε dc για να μπορούμε να αλλάζουμε το
μέγεθος των εικόνων
	SetStretchBltMode(dc3, HALFTONE);
	SetStretchBltMode(dc4, HALFTONE);
	SetStretchBltMode(dc5, HALFTONE);
	SetStretchBltMode(dc6, HALFTONE);
	SetStretchBltMode(w.m_dc,HALFTONE);
	SetStretchBltMode(w2.m_dc2,HALFTONE);
	SetStretchBltMode(win_small.m_dc_small,HALFTONE);
	SetStretchBltMode(win_small2.m_dc_small2,HALFTONE);
}


// CSurveillanceView diagnostics

#ifdef _DEBUG
void CSurveillanceView::AssertValid() const
{
	CFormView::AssertValid();
}

void CSurveillanceView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CSurveillanceDoc* CSurveillanceView::GetDocument() const // non-debug version
is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSurveillanceDoc)));
	return (CSurveillanceDoc*)m_pDocument;
}
```

```
#endif //_DEBUG

// Η ρουτίνα που γυρνάει διαδοχικά δεξιά και αριστερά την κάμερα κατά την διάρκεια
// της παρακολούθησης. Ορίζουμε και την μεταβλητή watch_pos για να ελέγχουμε την κίνηση.
int watch_pos=0;
void CSurveillanceView::Watch()
{
	// Αυξάνουμε κατά ένα την μεταβλητή θέσης
	watch_pos++;
	// Αν η μεταβλητή θέσης είναι πάνω από 12, τότε κινούμαστε αριστερά
	if(watch_pos > 12) MoveCamera(CString("left"));
	// Αν η μεταβλητή θέσης είναι μικρότερη ή ίση του 12, τότε κινούμαστε δεξιά
	if(watch_pos <= 12) MoveCamera(CString("right"));
		// Αν έχει φτάσει στον αριθμό 24, σημαίνει ότι έχει γίνει μία πλήρη περιστροφή 360 μοιρών
		// οπότε μηδενήζουμε την μεταβλητή θέσης
		if(watch_pos == 24) watch_pos = 0;
}

// Μεταβλητή για το αν έχουμε εισβολέα.
// Ως αρχική τιμή βάζουμε ότι δεν έχουμε.
bool intruder=FALSE;

// Η ρουτίνα που εκτελείτε όταν μετρήσει ο χρονομετρητής το χρόνο που του ορίσουμε
void CSurveillanceView::OnTimer(UINT_PTR nIDEvent)
{
	HDC dc0 = ::GetDC(0);

	// Καλούμε την συνάρτηση για να φορτώσουμε την αποθηκευμένη εικόνα
	// απο την τρέχουσα θέση της κάμερας, η θέση είναι γνωστή μέσω των
	// μεταβλητών posx, posy.
	LoadPosBitmap();
	// Μέσω της παρακάτω συνάρτησης αποθηκεύουμε στην θέση c:\now.jpg
	// την τρέχουσα εικόνα της κάμερας
	GetBitmap(_T("c:\\now.jpg"));
	CImage img;
	// Φορτώνουμε την εικόνα σε μία μεταβλητή CImage
	img.Load(_T("c:\\now.jpg"));
	// Εμφανίζουμε την εικόνα πάνω στην επιφάνεια dc1
	img.BitBlt(dc1, 0, 0, SRCCOPY);


	////////////////////////////////////////////////////////////////
	// Πίνακας για την αντιστοίχιση των επιφανειών dc με τις εικόνας//
	////////////////////////////////////////////////////////////////
	// Τρέχουσα εικόνα                  Κανονική          Μικρή
		//
	//          dc1                                  dc4               dc6
	                          //
	////////////////////////////////////////////////////////////////
	// Αποθηκευμένη εικόνα        Κανονική          Μικρή
	//
	//          dc                                   dc3               dc5
	                          //
	////////////////////////////////////////////////////////////////

	// Παιρνάμε την κάθε εικόνα στο κατάλληλο dc
	StretchBlt(dc3, 0, 0, 640, 480, dc, 0, 0, 640,480, SRCCOPY);
	StretchBlt(dc4, 0, 0, 640, 480, dc1, 0, 0, 640,480, SRCCOPY);
	StretchBlt(w.m_dc,0,0,320,240,dc3,0,0,640,480, SRCCOPY);
	StretchBlt(w2.m_dc2,0,0,320,240,dc4,0,0,640,480, SRCCOPY);
	StretchBlt(dc5, 0, 0, 3, 3, dc3, 0, 0, 640,480, SRCCOPY);
```

```
	StretchBlt(win_small.m_dc_small,0,0,120,120,dc5,0,0,3,3, SRCCOPY);
	StretchBlt(dc6, 0, 0, 3, 3, dc4, 0, 0, 640,480, SRCCOPY);
	StretchBlt(win_small2.m_dc_small2,0,0,120,120,dc6,0,0,3,3, SRCCOPY);

	// Εμφανίζουμε όλες τις εικόνες στο πρόγραμμα
	InvalidateRect(NULL);


	// Μεταβλητές για την ρουτίνα της σύγκρισης
	// Τρεις μεταβλητές για το καθένα από τα τρία βασικά χρώματα
	int red=0,blue=0,green=0;
	// Μεταβλητή που αθροίζονται οι τρεις παραπάνω μεταβλητές
	int sum=0;
	// Ορίζουμε την τιμή της ευαισθησίας για την εύρεση των διαφορών
	int dif=3*40;
	// Σε αυτές τις δύο μεταβλητές αποθηκεύονται οι συντεταγμένες για
	// την διαφορά που βρέθηκε, βάζουμε την τιμή 1, σε περίπτωση που
	// δεν βρεθεί διαφορά να μην μείνουν οι τελικές τιμές των παρακάτω
	// δομών ακολουθίας
	int y1 = 1;
	int x1 = 1;
	// Μεταβλητή με την οποία γνωρίζουμε αν έχει εντοπιστεί εισβολέας ή όχι
	intruder=FALSE;

	// Πρώτη δομή ακολουθίας για το Y
	for (int y=0; y<3; y++) {
		// Δεύτερη δομή ακολουθίας για το X
		for (int x=0; x<3; x++) {
			// Στις τρεις παρακάτω γραμμές παίρνουμε την τιμή για κάθε
πίξελ
			// για κάθε χρώμα για τις δύο εικόνες για την συγκεκριμένη
θέση
			// και τις αφαιρούμε μεταξύ τους, παίρνουμε την απόλυτη
τιμή
			// του αποτελέσματος και την αποθηκεύουμε στην μεταβλητή
που
			// αντιστοιχή στο κάθε χρώμα
			red=abs(GetRValue(GetPixel(dc3,x,y))-
GetRValue(GetPixel(dc4,x,y)));
			blue=abs(GetBValue(GetPixel(dc3,x,y))-
GetBValue(GetPixel(dc4,x,y)));
			green=abs(GetGValue(GetPixel(dc3,x,y))-
GetGValue(GetPixel(dc4,x,y)));
			// Αθροίζουμε τις τρεις παραπάνω τιμές σε μία μεταβλητή
			sum=red+green+blue;
			// Ελέγχουμε αν το άθροισμα των διαφορών είναι μεγαλύτερο
από την ευαισθησία
			// που έχουμε καθορίσει εμείς
			if(abs(sum) > dif)
			{
				// Αν ισχύει η παραπάνω συνθήκη τότε αποθηκεύουμε τις
				// τρέχουσες συντεταγμένες.
				x1 = x;
				y1 = y;
				// Και ορίζουμε την μεταβλητή intruder ίσον αληθείς,
				// δηλαδή ότι υπάρχει εισβολέας
				intruder=TRUE;
			}
		}
	}

	CString intrudermsg;
	intrudermsg.Format(CString("X=%2d|Diff:%d@(%d,
%d)\r\n"),posx,sum,x1,y1);
	PrintMsg(CString(intrudermsg));
```

```
// Ελέγχουμε αν υπάρχει εισβολέας μέσω της μεταβλητής intruder
if(!intruder)
{
	// Αν δεν υπάρχει εισβολέας συνεχίζουμε την περιμετρική κίνηση
	Watch();
	PrintMsg(CString("Δεν υπάρχει εισβολέας\r\n"));
} else {
	// Αν υπάρχει εισβολέας
	MessageBeep(-1);
	MailIt();
	// Πριν κουνήσουμε την κάμερα ελέγχουμε σε πια θέση βρίσκεται
	// έτσι ώστε αν είναι σε μία από τις ακραίες θέσεις να μην
	// δώσουμε εντολή να κινηθεί και άλλο και δημιουργήσει πρόβλημα
	// στον μηχανισμό της αλλά και στο πρόγραμμα
	if (posx>1)
		{
			// Ελέγχουμε την τιμή της τεταγμένης, και αν είναι
			// ίση με δύο, τότε κινούμαστε δεξιά
			if(x1==2)
			{
				// Αυξάνουμε κατά ένα το μετρητή θέσης
				watch_pos++;
				// Μετακινούμε την κάμερα δεξιά
				MoveCamera(CString("right"));
				PrintMsg(CString("Δεξιά\r\n"));
			}
		}
	if (posx<13)
		{
			// Ελέγχουμε την τιμή της τεταγμένης, και αν είναι
			// ίση με μηδέν, τότε κινούμαστε αριστερά
			if(x1==0)
			{
				// Αυξάνουμε κατά ένα το μετρητή θέσης
				watch_pos++;
				// Μετακινούμε την κάμερα αριστερά
				MoveCamera(CString("left"));
				PrintMsg(CString("Αριστερά\r\n"));
			}
		}

	if (posy>0)
		{
			// Ελέγχουμε την τιμή της τετμημένης, και αν είναι
			// ίση με δύο, τότε κινούμαστε αριστερά
			if(y1==0)
			{
				// Μετακινούμε την κάμερα πάνω
				MoveCamera(CString("up"));
				Sleep(500);
				PrintMsg(CString("Πάνω\r\n"));
			}
		}
	if (posy<4)
		{
			// Ελέγχουμε την τιμή της τετμημένης, και αν είναι
			// ίση με μηδέν, τότε κινούμαστε αριστερά
			if(y1==2)
			{
				// Μετακινούμε την κάμερα κάτω
				MoveCamera(CString("down"));
				Sleep(500);
				PrintMsg(CString("Κάτω\r\n"));
			}
		}
}
```

```
	// Απελευθερώνουμε το dc της οθόνης
	::ReleaseDC(0,dc0);
	sum=0;
	CFormView::OnTimer(nIDEvent);
}


// Η ρουτίνα που κινεί την κάμερα.
// Παίρνει ως όρισμα το που θα κίνηθεί
// σε ένα CString
void CSurveillanceView::MoveCamera(CString sWhere)
{
	//Κρατάμε τη θέση της κάμερας στις μεταβλητές posx, posy.
	// Ανάλογα με το που θα κινηθεί αυξάνουμε ή μειώνουμε.
	if(!sWhere.Compare(_T("left"))) posx--;
	if(!sWhere.Compare(_T("right"))) posx++;
	if(!sWhere.Compare(_T("up"))) posy--;
	if(!sWhere.Compare(_T("down"))) posy++;
	if(!sWhere.Compare(_T("home"))) {posx = 7; posy = 2;};


	// Εδώ κάνουμε τέσσερις ελέγχους για να προστατέψουμε την κάμερα
	// αλλά και το πρόγραμμα, από το να κινηθεί η κάμερα σε μη επιτρεπτεί
	// θέση, με αποτέλεσμα να κρασάρει το πρόγραμμα
	if(posx < 1)
	{
		posx = 1;
		return;
	}
	if(posx > 13)
	{
		posx = 13;
		return;
	}

	if(posy < 0)
	{
		posy = 0;
		return;
	}
	if(posy > 2)
	{
		posy = 2;
		return;
	}
	//Μετακίνηση της κάμερας
	const TCHAR szHeaders[] = _T("Accept: text/*\r\nUser-Agent:
Surveillance\r\n");
	CInternetSession session(_T("Camera"));
	CHttpConnection* pServer = NULL;
	CHttpFile* pFile = NULL;
	pServer = session.GetHttpConnection(m_IP);
	pFile = pServer->OpenRequest(CHttpConnection::HTTP_VERB_GET,
CString("cgi-bin/camctrl.cgi?move=") + sWhere);
	pFile->AddRequestHeaders(szHeaders);
	pFile->SendRequest();
	DWORD dwRet;
	pFile->QueryInfoStatusCode(dwRet);
	delete pFile;
	delete pServer;
	session.Close();
}


// Σκοπός αυτής της ρουτίνας είναι να μπορούμε να αλλάζουμε
// κάποιες ρυθμίσεις στην κάμερα και να την κινούμε σε προκαθορισμένες
```

```
// από εμάς θέσεις.
void CSurveillanceView::SetCameraOption(CString sWhat)
{
	//Ρύθμιση της κάμερας πχ sWhat="camstrl.cgi?move=left"
	const TCHAR szHeaders[] = _T("Accept: text/*\r\nUser-Agent:
Surveillance\r\n");
	CInternetSession session(_T("Camera"));
	CHttpConnection* pServer = NULL;
	CHttpFile* pFile = NULL;
	pServer = session.GetHttpConnection(m_IP);
	pFile = pServer->OpenRequest(CHttpConnection::HTTP_VERB_GET,
CString("cgi-bin/") + sWhat);
	pFile->AddRequestHeaders(szHeaders);
	pFile->SendRequest();
	DWORD dwRet;
	pFile->QueryInfoStatusCode(dwRet);
	delete pFile;
	delete pServer;
	session.Close();
	// Αν δώσουμε εντολή να κινηθεί η κάμερα σε μία από τις δύο ακραίες
θέσεις τότε
	// δίνουμε τις σωστές τιμές για την νέα θέση της κάμερας.
	if(!sWhat.Compare(_T("recall.cgi?recall=End_Right"))) {posx = 1; posy =
2;};
}


// Παίρνουμε μία εικόνα από την κάμερα και την αποθηκεύουμε στο Name.
// Το Name είναι το fullpath, όχι μόνο το όνομα αρχείου, πχ
// C:\MyFolder\image.jpg
void CSurveillanceView::GetBitmap(CString Name)
{
	const TCHAR szHeaders[] = _T("Accept: text/*\r\nUser-Agent:
Surveillance\r\n");
	char szBuff[100000];
	CInternetSession session(_T("Camera"));
	CHttpConnection* pServer = NULL;
	CHttpFile* pFile = NULL;
	pServer = session.GetHttpConnection(m_IP);
	pFile = pServer->OpenRequest(CHttpConnection::HTTP_VERB_GET, _T("cgi-
bin/video.jpg"));
	pFile->AddRequestHeaders(szHeaders);
	pFile->SendRequest();
	DWORD dwRet;
	pFile->QueryInfoStatusCode(dwRet);

	// Εάν πήραμε σωστά την εικόνα την αποθηκεύουμε...
	if (dwRet == HTTP_STATUS_OK)
	{
		CFile f;
		f.Open(Name, CFile::modeCreate|CFile::modeWrite);
		UINT nRead = pFile->Read(szBuff, 100000);
		while (nRead > 0)
		{
			f.Write(szBuff, nRead);
		nRead = pFile->Read(szBuff, 100000);
		}
		f.Close();
		}

		delete pFile;
		delete pServer;
		session.Close();
}
```

```
// Η ρουτίνα που σώζει όλες τις αρχικές εικόνας
void CSurveillanceView::InitBitmaps()
{
	// Μία δομή ακολουθίας για την κάθετη κίνηση
	for(int j = 1; j < 4; j++)
	{
		// Κινούμαστε στην αρχική θέση
		SetCameraOption(CString("recall.cgi?recall=End_Right"));
		// Χρονοκαθυστέρηση για να προλάβει η κάμερα να πάει εκεί
		Sleep(5000);
		// Δύο έλεγχοι που ανάλογα με την τιμή του j
		// κινούν την κάμερα είτε πάνω είτε κάτω
		if(j == 1) MoveCamera(CString("up"));
		if(j == 3) MoveCamera(CString("down"));
		// Χρονοκαθυστέρηση για να κινηθεί η κάμερα
		Sleep(500);
		// Άλλη μία δομή ακολουθίας για την οριζόντια κίνηση
		for(int i = 1; i < 14; i++)
		{
			CString name;
			// Δημιουργούμε το string για το όνομα και την διαδρομή του
αρχείου
			name.Format(_T("C:\\Screenshot-%i-%i.jpg"), j, i);
			// Και παίρνουμε και αποθηκεύουμε την εικόνα
			GetBitmap(name);
			// Η όλη διαδικασία γίνεται αριστερόστροφα
			if(i < 13) MoveCamera(CString("right"));
			// Χρονοκαθυστέρηση για να προλάβει η κάμερα να κινηθεί
			Sleep(1500);
		}
	}
	// Στο τέλος επιστροφή στην θέση εκκίνησης
	SetCameraOption(CString("recall.cgi?recall=End_Right"));
}


// Φορτώνουμε την αντίστοιχη αποθηκευμένη εικόνα από την τρέχουσα θέση της
κάμερας
void CSurveillanceView::LoadPosBitmap()
{

	CString s;
	// Δημιουργούμε το string για να βρούμε την εικόνα
	s.Format(CString("C:\\Screenshot-%i-%i.jpg"), posy, posx);
    CImage img;
	// Φορτώνουμε την εικόνα
	img.Load(s);
	// Παίρνουμε της διαστάσεις τις εικόνας
	cx = img.GetWidth();
	cy = img.GetHeight();
	// Την βάζουμε στην επιφάνεια dc
	img.BitBlt(dc, 0, 0, SRCCOPY);
}

void CSurveillanceView::MailIt()
{
	GetBitmap(CString("stam.jpg"));
	CString mail;
	mail.Format(CString("/smtp:%s /f:surveillance@speedtouch.lan
/to:test@speedtouch.lan;%s /s:Πιθανή εισβολή στον χώρο /b:Αυτή την στιγμή
υπάρχει πιθανή εισβολή στον χώρο που παρακολουθείτε... /a:stam.jpg"),
m_sMailServer, m_sMailReciept);
	ShellExecute(NULL, CString("open"), CString("CommandLineEmailer.exe"),
mail, NULL, SW_HIDE);
```

```
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Αρχικοποίηση
void CSurveillanceView::OnBnClickedInitpictures()
{
	// Στέλνουμε για εμφάνιση το παρακάτω μήνυμα
	PrintMsg(CString("Έναρξη Διαδικασίας αρχικοποίησης...\r\n"));
	// Καλούμε την συνάρτηση που φωτογραφίζει περιμετρικά τον χώρο
	InitBitmaps();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Έναρξη
void CSurveillanceView::OnBnClickedStart()
{
	// Ορίζουμε την μεταβλητή θέσης ίση με μηδέν
	watch_pos=0;
	// Στέλνουμε για εμφάνιση το παρακάτω μήνυμα
	PrintMsg(CString("Έναρξη παρακολούθησης...\r\n"));

	HDC dc0 = ::GetDC(0);
	dc = CreateCompatibleDC(dc0);
	bit = CreateCompatibleBitmap(dc0, 640, 480);
	bit = (HBITMAP)SelectObject(dc, bit);
	::ReleaseDC(0, dc0);

	// Απενεργοποίηση χειριστηρίων της κάμερας
	m_cUp.EnableWindow(FALSE);
	m_cDown.EnableWindow(FALSE);
	m_cHome.EnableWindow(FALSE);
	m_cRight.EnableWindow(FALSE);
	m_cLeft.EnableWindow(FALSE);

	// Κίνηση της κάμερας στην ακραία θέση
	SetCameraOption(CString("recall.cgi?recall=End_Right"));
	// Χρονοκαθυστέρηση για να προλάβει η κάμερα να πάει στην ακραία θέση
	Sleep(4000);
	// Ορίζουμε ότι ο χρονομετρητής θα εκτελείτε κάθε 2,5 δευτερόλεπτα
	SetTimer(ID_MOVES_TIMER,2500,NULL);

}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Διακοπή
void CSurveillanceView::OnBnClickedStop()
{
	// Σταματάμε το χρονομετρητή
	KillTimer(ID_MOVES_TIMER);
	// Κινούμε την κάμερα στην ακραία θέση
	SetCameraOption(CString("recall.cgi?recall=End_Right"));
	// Μηδενίζουμε την μεταβλητή θέσης
	watch_pos=0;
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Διακοπή παρακολούθησης...\r\n"));
	// Ενεργοποιούμαι ξανά τα χειριστήρια της κάμερας
	m_cUp.EnableWindow(TRUE);
	m_cDown.EnableWindow(TRUE);
	m_cHome.EnableWindow(TRUE);
	m_cRight.EnableWindow(TRUE);
	m_cLeft.EnableWindow(TRUE);



}
// Η ρουτίνα μέσω της οποίας εμφανίζουμε μηνύματα στο πρόγραμμα,
// παίρνει σαν όρισμα ένα CString το οποίο και εκτυπώνη με
// αντίστροφη σειρά (LIFO).
void CSurveillanceView::PrintMsg(CString Message)
```

```
{
	// Το νέο μήνυμα θα είναι το καινούργιο μήνυμα και μετά προσθέτουμε και
το παλιό
	CString
a=CString("----------------------------------------------------------------------
----------------\r\n");
	m_sMessage = Message + a + CString(m_sMessage);
	// Στέλνουμε την μεταβλητή m_sMessage με το μήνυμα να εμφανιστή στο
IDC_INFORMATIONS
	GetDlgItem(IDC_INFORMATIONS)->SetWindowText(CString(m_sMessage));

}




// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Αρχική Θέση
void CSurveillanceView::OnBnClickedHome()
{
	// Μετακινούμε την κάμερα στην αρχική θέση
	MoveCamera(CString("home"));
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Κίνηση κάμερας... Θέση: Αρχική\r\n"));
	ShowImg2Wnd();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Πάνω
void CSurveillanceView::OnBnClickedUp()
{
	// Μετακινούμε την κάμερα πάνω
	MoveCamera(CString("up"));
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Κίνηση κάμερας... Θέση: Πάνω\r\n"));
	ShowImg2Wnd();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Δεξιά
void CSurveillanceView::OnBnClickedRight()
{
	// Αυξάνουμε τον μετρητή θέσης κατά ένα
	watch_pos++;
	// Μετακινούμε την κάμερα δεξιά
	MoveCamera(CString("right"));
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Κίνηση κάμερας... Θέση: Δεξιά\r\n"));
	ShowImg2Wnd();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Κάτω
void CSurveillanceView::OnBnClickedDown()
{
	// Μετακινούμε την κάμερα κάτω
	MoveCamera(CString("down"));
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Κίνηση κάμερας... Θέση: Κάτω\r\n"));
	ShowImg2Wnd();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Αριστερά
void CSurveillanceView::OnBnClickedLeft()
{
	// Αυξάνουμε τον μετρητή θέσης κατά ένα
	watch_pos++;
	// Μετακινούμε την κάμερα αριστερά
	MoveCamera(CString("left"));
```

```
	// Στέλνουμε για εκτύπωση το παρακάτω μήνυμα
	PrintMsg(CString("Κίνηση κάμερας... Θέση: Αριστερά\r\n"));
	ShowImg2Wnd();
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Mail server
void CSurveillanceView::OnBnClickedMailserver()
{

	CMailSettingsDlg dlg;
	dlg.m_sMailServer=m_sMailServer;
	dlg.m_sMailPassword=m_sMailPassword;
	dlg.m_sMailReciept=m_sMailReciept;
	dlg.m_sMailUser=m_sMailUser;
	if(dlg.DoModal() == IDOK)
	{
		m_sMailServer=dlg.m_sMailServer;
		m_sMailPassword=dlg.m_sMailPassword;
		m_sMailReciept=dlg.m_sMailReciept;
		m_sMailUser=dlg.m_sMailUser;
		CString a;
		a.Format(CString("Οι ρυθμίσεις για τον Mail Server
αποθηκεύτηκαν!\r\nServer: %s\r\nΧρήστης: %s\r\nΚωδικός: *******\r\
nΠαραλήπτης: %s\r\n"),m_sMailServer,m_sMailUser,m_sMailReciept);
		PrintMsg(CString(a));
	}
}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί Ευαισθησία
void CSurveillanceView::OnBnClickedSensivity()
{
	// Εμφανίζουμε σε πρώτο πλάνο το παράθυρο με την ρύθμιση της
ευαισθησίας

	CSensivityDlg dlg;
	if(dlg.DoModal() == IDOK)
	{
		CString a;
		a.Format(CString("Η ευαισθησία άλλαξε σε: %s%
%\r\n"),dlg.m_SliderValue);
		PrintMsg(a);
	}


}

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί IP της IPCAM
void CSurveillanceView::OnBnClickedCameraip()
{
	// Εμφανίζουμε σε πρώτο πλάνο το παράθυρο με την ρύθμιση της IP της
κάμερας
	CCameraIPDlg dlg;
	dlg.m_IP=m_IP;
	if(dlg.DoModal() == IDOK)
	{
		m_IP=dlg.m_IP;
		CString a;
		a.Format(CString("Η IP της κάμερας άλλαξε σε: %s\r\n"),m_IP);
		PrintMsg(CString(a));
	}
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ρυθμίσεις-> Ευαισθησία
void CSurveillanceView::OnSettingsSensivity()
{
```

```
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedSensivity();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ρυθμίσεις->IP της IPCAM
void CSurveillanceView::OnSettingsCameraip()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedCameraip();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ρυθμίσεις-> Mail Server
void CSurveillanceView::OnSettingsMailserver()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedMailserver();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή Κινήσεις-
>Οριζόντιος Άξονας->Δεξιά
void CSurveillanceView::OnMovesRight()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedRight();
}
void CSurveillanceView::OnBnClickedFollowme()
{
	CSensivityDlg dlg;
	CString test;
	test.Format(CString("test: %s"), dlg.m_SliderValue);
}
// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή Κινήσεις-
>Οριζόντιος Άξονας->Αριστερά
void CSurveillanceView::OnMovesLeft()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedLeft();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή Κινήσεις-
>Κάθετος Άξονας->Πάνω
void CSurveillanceView::OnMovesUp()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedUp();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή Κινήσεις-
>Κάθετος Άξονας->Κάτω
void CSurveillanceView::OnMovesDown()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedDown();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή Κινήσεις-
> Αρχική θέση
```

```
void CSurveillanceView::OnMovesHome()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedHome();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ενέργειες-> Αρχικοποίηση
void CSurveillanceView::OnActionsInitpictures()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedInitpictures();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ενέργειες-> Έναρξη
void CSurveillanceView::OnActionsStart()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedStart();
}

// Η ρουτίνα που εκτελείτε όταν επιλέξουμε από το μενού την επιλογή
Ενέργειες-> Διακοπή
void CSurveillanceView::OnActionsStop()
{
	// Καλούμε την προαναφερθήσα συνάρτηση που κάνει την ίδια
	// διαδικασία πατώντας αντί για το μενού το αντίστοιχο κουμπί
	OnBnClickedStop();
}

void CSurveillanceView::ShowImg2Wnd()
{
	// Δημιουργούμε ένα αντικείμενο CImage
	CImage img;
	// Φορτώνουμε την εικόνα σε μία μεταβλητή CImage
	img.Load(_T("c:\\now.jpg"));
	// Εμφανίζουμε την εικόνα πάνω στις κατάλληλες επιφάνειες
	img.BitBlt(dc1, 0, 0, SRCCOPY);
	StretchBlt(w2.m_dc2,0,0,320,240,dc1,0,0,640,480, SRCCOPY);
	StretchBlt(dc6, 0, 0, 3, 3, dc1, 0, 0, 640,480, SRCCOPY);
	StretchBlt(win_small2.m_dc_small2,0,0,120,120,dc6,0,0,3,3, SRCCOPY);
	// Ενημερώνουμε το παράθυρο
	w2.InvalidateRect(NULL);
	win_small2.InvalidateRect(NULL);
	InvalidateRect(NULL);
}
```

```
// SurveillanceView.h : interface of the CSurveillanceView class
//

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για το κεντρικό παράθυρο του προγράμματος

#pragma once
#include "mailsettingsdlg.h"
#include "sensivitydlg.h"
#include "cameraipdlg.h"
#include "afxwin.h"


class CSurveillanceView : public CFormView
{
protected: // create from serialization only
	CSurveillanceView();
	DECLARE_DYNCREATE(CSurveillanceView)

public:
	CString m_IP;
	CString m_sMailServer;
	CString m_sMailUser;
	CString m_sMailPassword;
	CString m_sMailReciept;
	enum{ IDD = IDD_SURVEILLANCE_FORM };
	void MailIt();
	void MoveCamera(CString sWhere);
	void SetCameraOption(CString sWhat);
	void InitBitmaps();
	void Watch();
	void LoadPosBitmap();
	void GetBitmap(CString Name);
	void ShowImg2Wnd();

// Attributes
public:
	CSurveillanceDoc* GetDocument() const;

// Operations
public:

// Overrides
public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	virtual void OnInitialUpdate(); // called first time after construct

// Implementation
public:
	virtual ~CSurveillanceView();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClickedButton4();
	afx_msg void OnBnClickedHome();
	afx_msg void OnBnClickedUp();
```

```
	afx_msg void OnBnClickedRight();
	afx_msg void OnBnClickedDown();
	afx_msg void OnBnClickedLeft();
	afx_msg void OnBnClickedInitpictures();
	afx_msg void OnBnClickedStart();
private:
	CMailSettingsDlg m_dMailSettings;
	CSensivityDlg m_dSensivityDlg;
	CCameraIPDlg m_dCameraIPDlg;
	CBitmap m_Bit;
public:

	afx_msg void OnBnClickedMailserver();
	afx_msg void OnBnClickedSensivity();
	afx_msg void OnBnClickedCameraip();
	afx_msg void OnSettingsSensivity();
	afx_msg void OnSettingsCameraip();
	afx_msg void OnSettingsMailserver();
	afx_msg void OnMovesRight();
	afx_msg void OnMovesLeft();
	afx_msg void OnMovesUp();
	afx_msg void OnMovesDown();
	afx_msg void OnMovesHome();
	afx_msg void OnBnClickedStop();
	afx_msg void OnTimer(UINT_PTR nIDEvent);
	CButton m_cHome;
	CButton m_cRight;
	CButton m_cLeft;
	CButton m_cDown;
	CButton m_cUp;
	afx_msg void OnActionsInitpictures();
	afx_msg void OnActionsStart();
	afx_msg void OnActionsStop();
	CString m_smessage;
	afx_msg void OnBnClickedFollowme();
	CString m_sInformations;
	CString m_sMessage;
	void PrintMsg(CString Message);
};

#ifndef _DEBUG  // debug version in SurveillanceView.cpp
inline CSurveillanceDoc* CSurveillanceView::GetDocument() const
   { return reinterpret_cast<CSurveillanceDoc*>(m_pDocument); }
#endif
```

```
// SurveillanceDoc.cpp : implementation of the CSurveillanceDoc class
//

// Σε αυτό το αρχείο υπάρχουν κάποιες ρουτίνες που δημιουργούνται αυτόματα
// από το Visual Studio λόγω του ότι επιλέξαμε SDI project

#include "stdafx.h"
#include "Surveillance.h"
#include "DisplayWnd.h"
#include "SurveillanceDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CSurveillanceDoc

IMPLEMENT_DYNCREATE(CSurveillanceDoc, CDocument)

BEGIN_MESSAGE_MAP(CSurveillanceDoc, CDocument)
	ON_COMMAND(ID_FILE_SEND_MAIL, &CSurveillanceDoc::OnFileSendMail)
	ON_UPDATE_COMMAND_UI(ID_FILE_SEND_MAIL,
&CSurveillanceDoc::OnUpdateFileSendMail)
END_MESSAGE_MAP()


// CSurveillanceDoc construction/destruction

CSurveillanceDoc::CSurveillanceDoc()
{
	// TODO: add one-time construction code here

}

CSurveillanceDoc::~CSurveillanceDoc()
{
}

BOOL CSurveillanceDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)

	return TRUE;
}




// CSurveillanceDoc serialization

void CSurveillanceDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: add storing code here
	}
	else
	{
		// TODO: add loading code here
	}
}
```

```
// CSurveillanceDoc diagnostics

#ifdef _DEBUG
void CSurveillanceDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CSurveillanceDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG


// CSurveillanceDoc commands
```

```
// SurveillanceDoc.h : interface of the CSurveillanceDoc class
//


#pragma once


class CSurveillanceDoc : public CDocument
{
protected: // create from serialization only
	CSurveillanceDoc();
	DECLARE_DYNCREATE(CSurveillanceDoc)

// Attributes
public:

// Operations
public:

// Overrides
public:
	virtual BOOL OnNewDocument();
	virtual void Serialize(CArchive& ar);

// Implementation
public:
	virtual ~CSurveillanceDoc();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
	DECLARE_MESSAGE_MAP()
};
```

```
// MailSettingsDlg.cpp : implementation file
//

// Σε αυτό το αρχείο υπάρχουν κάποιες ρουτίνες που αφορούν το
// παράθυρο για την ρύθμιση της ευαισθησίας

// Εδώ συμπεριλαμβάνουμε όλα τα απαραίτητα αρχεία που χρείαζεται
// το παράθυρο για να λειτουργήσει.
#include "stdafx.h"
#include "Surveillance.h"
#include "MailSettingsDlg.h"


// CMailSettingsDlg dialog

IMPLEMENT_DYNAMIC(CMailSettingsDlg, CDialog)

CMailSettingsDlg::CMailSettingsDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMailSettingsDlg::IDD, pParent)
	// Εδώ αρχικοποιούμε κάποιες τιμές σχετικά με τις ρυθμίσεις
	// για την λειτουργία του mail server
{

}


CMailSettingsDlg::~CMailSettingsDlg()
{
}

void CMailSettingsDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_MAIL_SERVER, m_sMailServer);
	DDX_Text(pDX, IDC_MAIL_USER, m_sMailUser);
	DDX_Text(pDX, IDC_MAIL_PASSWORD, m_sMailPassword);
	DDX_Text(pDX, IDC_MAIL_RECIEPT, m_sMailReciept);
}

BEGIN_MESSAGE_MAP(CMailSettingsDlg, CDialog)
	ON_BN_CLICKED(IDOK, &CMailSettingsDlg::OnBnClickedOk)
END_MESSAGE_MAP()


// CMailSettingsDlg message handlers

// Η ρουτίνα που εκτελείτε όταν πατήσουμε το κουμπί OK
void CMailSettingsDlg::OnBnClickedOk()
{
	// Κλείνει το παράθυρο
	OnOK();
}
```

```
#pragma once

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για το παράθυρο για την ρύθμιση του MailServer

// CMailSettings Dlg dialog

class CMailSettingsDlg : public CDialog
{
	DECLARE_DYNAMIC(CMailSettingsDlg)

public:
	CMailSettingsDlg(CWnd* pParent = NULL);   // standard constructor
	virtual ~CMailSettingsDlg();

// Dialog Data
	enum { IDD = IDD_MAILSETTINGS };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

	DECLARE_MESSAGE_MAP()
public:
	CString m_sMailServer;
	CString m_sMailUser;
	CString m_sMailPassword;
	CString m_sMailReciept;
	afx_msg void OnBnClickedOk();
};
```

```
// SensivityDlg.cpp : implementation file
//

// Σε αυτό το αρχείο υπάρχουν κάποιες ρουτίνες που αφορούν το
// παράθυρο για την ρύθμιση της ευαισθησίας

// Εδώ συμπεριλαμβάνουμε όλα τα απαραίτητα αρχεία που χρείαζεται
// το παράθυρο για να λειτουργήσει.
#include "stdafx.h"
#include "Surveillance.h"
#include "SensivityDlg.h"


// CSensivityDlg dialog

IMPLEMENT_DYNAMIC(CSensivityDlg, CDialog)

CSensivityDlg::CSensivityDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSensivityDlg::IDD, pParent)
	, m_SliderValue(_T("0"))
{

}

CSensivityDlg::~CSensivityDlg()
{
}

void CSensivityDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_SLIDER, m_Slider);
	DDX_Text(pDX, IDC_SLIDER_VALUE, m_SliderValue);
}


BEGIN_MESSAGE_MAP(CSensivityDlg, CDialog)

	ON_WM_HSCROLL()
	ON_BN_CLICKED(IDOK, &CSensivityDlg::OnBnClickedOk)
//	ON_EN_CHANGE(IDC_SLIDER_VALUE, &CSensivityDlg::OnEnChangeSliderValue)
ON_WM_CREATE()
ON_WM_ACTIVATE()
END_MESSAGE_MAP()


// CSensivityDlg message handlers


void CSensivityDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar)
{
	CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
	// TODO: Add your message handler code here and/or call default
	if(nSBCode & (SB_THUMBPOSITION|SB_THUMBTRACK))
	m_SliderValue.Format(CString("%ld"), nPos);
	UpdateData(false);



}

void CSensivityDlg::OnBnClickedOk()
{
	// TODO: Add your control notification handler code here
	OnOK();
```

```
}
int CSensivityDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CDialog::OnCreate(lpCreateStruct) == -1)
		return -1;

	// TODO:  Add your specialized creation code here

	return 0;
}

bool ini = false;

void CSensivityDlg::OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized)
{
	CDialog::OnActivate(nState, pWndOther, bMinimized);
	if(!ini)
	((CSliderCtrl*)GetDlgItem(IDC_SLIDER))->SetRange(0, 150);
	ini = true;
}
```

```
#pragma once
#include "afxcmn.h"

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για το παράθυρο για την ρύθμιση της ευαισθησίας

// CSensivityDlg dialog

class CSensivityDlg : public CDialog
{
	DECLARE_DYNAMIC(CSensivityDlg)

public:
	CSensivityDlg(CWnd* pParent = NULL);   // standard constructor
	virtual ~CSensivityDlg();

// Dialog Data
	enum { IDD = IDD_SENSIVITY };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

	DECLARE_MESSAGE_MAP()
public:
	CSliderCtrl m_Slider;
	CString m_SliderValue;
	afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar);
	afx_msg void OnBnClickedOk();
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	afx_msg void OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized);
};
```

```cpp
// CameraIPDlg.cpp : implementation file
//

// Σε αυτό το αρχείο υπάρχουν κάποιες ρουτίνες που αφορούν το
// παράθυρο για την ρύθμιση της ευαισθησίας

// Εδώ συμπεριλαμβάνουμε όλα τα απαραίτητα αρχεία που χρείαζεται
// το παράθυρο για να λειτουργήσει.
#include "stdafx.h"
#include "Surveillance.h"
#include "CameraIPDlg.h"


// CCameraIPDlg dialog

IMPLEMENT_DYNAMIC(CCameraIPDlg, CDialog)

CCameraIPDlg::CCameraIPDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CCameraIPDlg::IDD, pParent)
	, m_IP(_T("192.168.1.66"))
{

}

CCameraIPDlg::~CCameraIPDlg()
{
}

void CCameraIPDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_IP, m_IP);
}


BEGIN_MESSAGE_MAP(CCameraIPDlg, CDialog)
	ON_BN_CLICKED(IDOK, &CCameraIPDlg::OnBnClickedOk)
END_MESSAGE_MAP()


// CCameraIPDlg message handlers

void CCameraIPDlg::OnBnClickedOk()
{
	// TODO: Add your control notification handler code here
	OnOK();
}
```

```
#pragma once

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για το παράθυρο για την ρύθμιση της IP της κάμερας

// CCameraIPDlg dialog

class CCameraIPDlg : public CDialog
{
	DECLARE_DYNAMIC(CCameraIPDlg)

public:
	CCameraIPDlg(CWnd* pParent = NULL);   // standard constructor
	virtual ~CCameraIPDlg();

// Dialog Data
	enum { IDD = IDD_CAMERAIP };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

	DECLARE_MESSAGE_MAP()
public:
	CString m_IP;
	afx_msg void OnBnClickedOk();
};
```

```
// DisplayWnd.cpp : implementation file
//

// Το αρχείο που μας χρειάζεται για την δημιουργία των δύο
// των δύο παραθύρων για την εμφάνιση των εικόνων

#include "stdafx.h"
#include "Surveillance.h"
#include "DisplayWnd.h"

// CDisplayWnd

IMPLEMENT_DYNAMIC(CDisplayWnd, CWnd)

CDisplayWnd::CDisplayWnd()
{

}

CDisplayWnd::~CDisplayWnd()
{
}


BEGIN_MESSAGE_MAP(CDisplayWnd, CWnd)
	ON_WM_PAINT()
END_MESSAGE_MAP()



// CDisplayWnd message handlers

void CDisplayWnd::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	// TODO: Add your message handler code here
	// Do not call CWnd::OnPaint() for painting messages
	// Ντι σι???
	BitBlt(dc.m_hDC, 0, 0, 320, 240, m_dc, 0, 0, SRCCOPY);
	BitBlt(dc.m_hDC, 0, 0, 240, 240, m_dc_small, 0, 0, SRCCOPY);
}
```

```cpp
#pragma once

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για τα δύο παράθυρα για την εμφάνιση των εικόνων

// CDisplayWnd

class CDisplayWnd : public CWnd
{
	DECLARE_DYNAMIC(CDisplayWnd)

public:
	CDisplayWnd();
	virtual ~CDisplayWnd();
	HDC m_dc;
	HBITMAP m_Bit;
	HDC m_dc_small;
	HBITMAP m_Bit_small;

protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnPaint();
};
```

```
// DisplayWnd2.cpp : implementation file
//

// Το αρχείο που μας χρειάζεται για την δημιουργία των δύο
// των δύο παραθύρων για την εμφάνιση των εικόνων

#include "stdafx.h"
#include "Surveillance.h"
#include "DisplayWnd2.h"

// CDisplayWnd2

IMPLEMENT_DYNAMIC(CDisplayWnd2, CWnd)

CDisplayWnd2::CDisplayWnd2()
{

}

CDisplayWnd2::~CDisplayWnd2()
{
}


BEGIN_MESSAGE_MAP(CDisplayWnd2, CWnd)
	ON_WM_PAINT()
END_MESSAGE_MAP()



// CDisplayWnd2 message handlers

void CDisplayWnd2::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	// TODO: Add your message handler code here
	// Do not call CWnd::OnPaint() for painting messages
	// Ντι σι???
	BitBlt(dc.m_hDC, 0, 0, 320, 240, m_dc2, 0, 0, SRCCOPY);
	BitBlt(dc.m_hDC, 0, 0, 240, 240, m_dc_small2, 0, 0, SRCCOPY);
}
```

```cpp
#pragma once

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για τα δύο παράθυρα για την εμφάνιση των εικόνων

// CDisplayWnd2

class CDisplayWnd2 : public CWnd
{
	DECLARE_DYNAMIC(CDisplayWnd2)

public:
	CDisplayWnd2();
	virtual ~CDisplayWnd2();
	HDC m_dc2;
	HBITMAP m_Bit2;
	HDC m_dc_small2;
	HBITMAP m_Bit_small2;

protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnPaint();
};
```

```cpp
// MainFrm.cpp : implementation of the CMainFrame class
//

// Σε αυτό το αρχείο υπάρχουν κάποιες ρουτίνες που δημιουργούνται αυτόματα
// από το Visual Studio λόγω του ότι επιλέξαμε SDI project

#include "stdafx.h"
#include "Surveillance.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	ON_WM_CREATE()
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // status line indicator
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};


// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
	// TODO: add member initialization code here
}

CMainFrame::~CMainFrame()
{
}


int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

	if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE |
CBRS_TOP
		| CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC)
||
		!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
	{
		TRACE0("Failed to create toolbar\n");
		return -1;      // fail to create
	}

	if (!m_wndStatusBar.Create(this) ||
		!m_wndStatusBar.SetIndicators(indicators,
		  sizeof(indicators)/sizeof(UINT)))
	{
		TRACE0("Failed to create status bar\n");
		return -1;      // fail to create
```

```
	}

	// TODO: Delete these three lines if you don't want the toolbar to be
dockable
	m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
	EnableDocking(CBRS_ALIGN_ANY);
	DockControlBar(&m_wndToolBar);

	return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
	if( !CFrameWnd::PreCreateWindow(cs) )
		return FALSE;
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	cs.style = WS_OVERLAPPED | WS_CAPTION | FWS_ADDTOTITLE
		| WS_THICKFRAME | WS_MINIMIZEBOX | WS_SYSMENU;

	return TRUE;
}


// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}

#endif //_DEBUG


// CMainFrame message handlers
```

```
// MainFrm.h : interface of the CMainFrame class
//

// Αρχείο με όλες τις δηλώσεις μεταβλητών και ενεργειών
// για την γραμμή εργαλείων καθώς και την γραμμή κατάστασης

#pragma once

class CMainFrame : public CFrameWnd
{

protected: // create from serialization only
	CMainFrame();
	DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// Implementation
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
	CStatusBar  m_wndStatusBar;
	CToolBar    m_wndToolBar;

// Generated message map functions
protected:
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	DECLARE_MESSAGE_MAP()
};
```