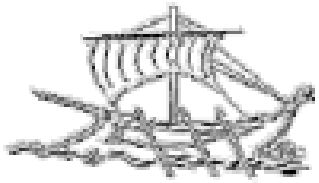


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΝΟΜΟ ΑΜΑΞΙΔΙΟ ANDROID

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

Εισηγητής: Δρ. Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΔΕΚΕΜΒΡΙΟΣ 2015

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΝΟΜΟ ΑΜΑΞΙΔΙΟ ANDROID

Χρήστος Λ. Κατσάρης
Α.Μ. 42463
Ηλίας Γ. Γαλανόπουλος
Α.Μ. 42070

Εισηγητής:

Δρ. Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

_____
_____
_____

Ημερομηνία εξέτασης:_____

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση αυτής της πτυχιακής θα ήταν αδύνατη χωρίς την πολύτιμη συνεισφορά του καθηγητή μας, Δρ. Ιωάννη Έλληνα, ο οποίος, όποτε χρειαστήκαμε βοήθεια, ήταν εκεί και έλυνε άμεσα τις όποιες απορίες μας. Επίσης, ένα μεγάλο ευχαριστώ στους καθηγητές του τμήματός μας που, τέσσερα χρόνια τώρα, μέσω των γνώσεων τους μας δίδαξαν πολλά, μας έκαναν να αγαπήσουμε το αντικείμενο των σπουδών μας και εξέλιξαν μέσα μας την έννοια της συνεργασίας, προϊόν της οποίας είναι και η παρούσα πτυχιακή. Ακόμα, ευχαριστούμε πολύ όλους τους συμφοιτητές μας, για την συνεργασία μας όλα αυτά τα χρόνια.

Τέλος, ένα θερμό ευχαριστώ, ο ένας στον άλλον φυσικά, για την άψογη συνεργασία και τη δυνατή φιλία που χαρακτηρίζει τη σχέση μας, τους γονείς μας, καθώς και τους Φένια Πυργερή και Νικόλαο Χάψα που υπήρξαν ένα πολύτιμο στήριγμα και μας βοήθησαν σημαντικά.

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται την ανάπτυξη και την υλοποίηση ενός αυτόνομου αμαξιδίου, το οποίο, μέσω της κάμερας ενός τηλεφώνου Android, ακολουθεί μια συγκεκριμένη διαδρομή.

Στο πλαίσιο της ανάπτυξης της πτυχιακής εργασίας, παρουσιάζονται τα χαρακτηριστικά των Android και Arduino στα οποία βασίστηκε η εργασία, καθώς και του Android Studio, πάνω στο οποίο εργαστήκαμε. Τέλος, παρατίθεται ο κώδικας της εργασίας όπως επίσης και κάποιες δυνατότητες.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη εφαρμογών Android, Arduino

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αυτοματισμός, Android, Arduino, OpenCV, Bluetooth

ABSTRACT

The present assignment examines the development and the materialization of an autonomous car which follows a specific path via the camera of an Android smartphone.

Furthermore, during the development of this assignment the characteristics of Android, Arduino and Android Studio are being presented, in which our project was based. Still, the mechanical parts of the project are being presented too. Finally, the coding of the project is being explained and some capabilities for extension and improvement are being stated.

SCIENTIFIC AREA: Android, Arduino application programming

Key Words: Automation, Android, Arduino, OpenCV, Bluetooth

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

A.S. – ANDROID STUDIO

SDK – SOFTAWARE DEVELOPMENT KIT

OS – OPERATE SYSTEM

JAVA SE – JAVA STANDARD EDITION

JDK – JAVA DEVELOPMENT KIT

RAM – RANDOM ACCESS MEMORY

GB – GIGA BYTE

MB – MEGA BYTE

API – APPLICATION PROGRAMMING INTERFACE

AVD – ANDROID VIRTUAL DEVICE

GPU – GRAPHICS PROCESSING UNIT

ADB – ANDROID DEBUG BRIDGE

HSV – HUE SATURATION VALUE

RGB – RED GREEN BLUE

BT - BLUETOOTH

OHA - OPEN HANDSET ALLIANCE

LED - LIGHT EMITTING DIODE

NFC - NEAR FIELD COMMUNICATION

VoIP - VOICE OVER IP

IP - INTERNET PROTOCOL

ICS - ICE CREAM SANDWICH

UI - USER INTERFACE

Η/Υ - ΗΛΕΚΤΡΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΤΗΣ

I/O – INPUT/OUTPUT

USB – UNIVERSAL SERIAL BUS

IDE – INTEGRATED DEVELOPMENT ENVIRONMENT

V – VOLTAGE

WI-FI - WIRELESS FIDELITY

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

Αρχικά σκεφτήκαμε να δημιουργήσουμε μια εφαρμογή android η οποία δεν έχει κάποιο tools να αγοράσεις, με αποτέλεσμα το μόνο εφόδιο που θα πρέπει να θυσιάσουμε να είναι μόνο ο χρόνος είτε για εκμάθηση της java γλώσσας είτε της εκμάθησης του τρόπου λειτουργίας του Android Studio.

Η επόμενη κίνηση είναι να βρούμε το θέμα της εργασίας. Θέλαμε μια καινοτόμα ιδέα να υλοποιήσουμε έτσι ώστε να μπορούν στην πορεία να την εξελίξουν και για να είναι μια πτυχιακή η οποία δεν θα είναι απλά στην άκρη ενός γραφείου, αλλά μια πτυχιακή ενδιαφέρουσα και επεκτάσιμη.

1.1 Αντικείμενο της πτυχιακής εργασίας

Η καινοτόμα ιδέα στην οποία καταλήξαμε είναι η δημιουργία μιας εφαρμογής η οποία θα μπορεί να κινήσει ένα αυτοκινητάκι / ρομπότ να κινείται προς μια κατεύθυνση χωρίς κάποιος να αναλαμβάνει το χειρισμό αυτού. Απλά να του λέει ακολούθα αυτό και εκείνο να το πράττει.

Με λίγα λόγια, η συγκεκριμένη πτυχιακή αναφέρεται στην παρουσίαση των android εφαρμογών, καθώς και των δυνατοτήτων που έχουν. Σκοπός της εργασίας μας είναι να δημιουργήσουμε μία εφαρμογή Android, κάνοντας χρήση του Android Studio, η οποία θα εκκινεί κάθε φορά που εκτελείται την κάμερα ενός κινητού με Android λογισμικό. Από την κάμερα θα κάνουμε ανίχνευση στον χώρο και αυτό που θα ανιχνεύεται από αυτήν, με τις κατάλληλες εντολές μέσω προγραμματισμού θα κινείται το αμαξίδιο.

Το αυτοκινητάκι κινείται εξαιτίας των τεσσάρων κινητήρων που έχει πάνω του(ένας για κάθε ρόδα). Αυτοί οι κινητήρες συνδέονται με ένα Arduino Uno το οποίο είναι προγραμματισμένο να ξεκινάει ή να σταματάει τους κινητήρες προκειμένου να έχει την δυνατότητα το αυτοκινητάκι να κάνει κίνηση προς τα εμπρός, προς τα αριστερά, προς τα δεξιά και να σταματά.

Σε αυτό το σημείο λειτουργεί η Android εφαρμογή μας. Μέσω της κάμερας του κινητού ανιχνεύει τον χώρο και σε περίπτωση που ανιχνεύσει κάτι, τότε με τις κατάλληλες εντολές προγραμματισμού, επικοινωνεί με το Arduino Που έχει το

αυτοκινητάκι μέσω ενός Bluetooth module και έτσι παράγεται η κίνηση του αυτοκινήτου.

1.2 Οργάνωση του κειμένου

Αρχικά έχουμε τις ευχαριστίες στην συνέχεια την περίληψη και στα ελληνικά και στα αγγλικά. Τον κατάλογο σχημάτων ,τον κατάλογο πινάκων και τέλος τις συντομογραφίες.

Στο κεφάλαιο 1 κάνουμε μια εισαγωγή της εργασίας μας. Το πώς σκεφτήκαμε αυτήν την καινοτόμα ιδέα και ποιο είναι ακριβώς το αντικείμενο αυτής της πτυχιακής. Τώρα αναφέρουμε κάποιες έννοιες της εργασίας που θα εξηγηθούν αναλυτικά στα παρακάτω κεφάλαια.

Στο κεφάλαιο 2 κάνουμε μια πρώτη γνωριμία του αναγνώστη της πτυχιακής με το τι είναι το Android. Λίγο πολύ όλοι κάποια στιγμή στην ζωή μας έχουμε χρησιμοποιήσει μια εφαρμογή Android. Σας παρουσιάζουμε την ιστορική εξέλιξη του Android από τότε που δημιουργήθηκε μέχρι σήμερα. Λίγα λόγια για την κατανόηση της έννοιας android.

Στο κεφάλαιο 3 σας παρουσιάζουμε τις πλατφόρμες που μπορείς να εργαστείς για την δημιουργία μιας android εφαρμογής. Αποδεικνύουμε με στοιχεία ποια από τις δύο τελικά επιλέξαμε και στην συνέχεια περνάμε στην σύγκριση των android / ios / windows applications.

Στο κεφάλαιο 4 δείχνουμε βήμα-βήμα όλα τα απαραίτητα βήματα που πρέπει να κάνει κάθε προγραμματιστής android εφαρμογών προκειμένου να έχει ένα σωστό android environment. Παρουσιάζουμε όλα τα kit που θα πρέπει να εγκαταστήσετε καθώς και το από πού θα τα κατεβάσετε. Επίσης δείχνουμε την δημιουργία ενός emulator ο οποίος είναι πολύ σημαντικός.

Στο κεφάλαιο 5 σας παρουσιάζουμε την βιβλιοθήκη OpenCV3.0.0 η οποία είναι υπεύθυνη για την ανίχνευση των αντικειμένων στον χώρο. Είναι μια βιβλιοθήκη του Android Studio η οποία είναι απαραίτητη για την χρήση της κάμερας του κινητού.

Στο κεφάλαιο 6 κάνουμε μια αναφορά στα Arduino. Είναι μικροεπεξεργαστές οι οποίοι λύνουν κυριολεκτικά τα χέρια μας και βοηθάει στην επικοινωνία της εφαρμογής του κινητού με το αμαξίδιο.

Στο κεφάλαιο 7 σας παρουσιάζουμε κάποιες εικόνες για να πάρετε μια ιδέα για το πώς είναι κατασκευασμένο το αυτοκινητάκι της εργασίας μας.

Στο κεφάλαιο 8 κάνουμε μια πιο εκτενή αναφορά στην εργασία μας, εξηγώντας όλες τις πτυχές της εργασίας. Δημιουργούμαι το project βήμα-βήμα από το 0 και εξηγούμε όλες τις ενέργειες που έγιναν προκειμένου να υλοποιηθεί. Στην συνέχεια κάνουμε ανάλυση των στοιχείων της εφαρμογής και επεξήγηση όλου του κώδικα.

Στο κεφάλαιο 9 αναλύουμε της προοπτικές της εργασίας και πως αυτή μπορεί να εξελιχθεί και κάνουμε μια σύνοψη της εργασίας.

Στο ΠΑΡΑΡΤΗΜΑ Α’ έχουμε ολόκληρο τον κώδικα της εργασίας μας.

Στο ΠΑΡΑΡΤΗΜΑ Β’ έχουμε τον κώδικα του Arduino.

Και τέλος έχουμε την βιβλιογραφία.

ΚΕΦΑΛΑΙΟ 2 – ANDROID

2.1 Εισαγωγή Android

Το android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα το οποίο χρησιμοποιείται κυρίως σε συσκευές με οθόνη αφής όπως τα smartphones και τα tablets αλλά τελευταία κάνει την εμφάνισή του και σε άλλες συσκευές όπως τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto), ρολόγια χειρός (Android Wear), κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές ακόμα και σε συνηθισμένους Η/Υ (π.χ. το HP Slate 21). Το android το οποίο παρουσιάστηκε το 2007 για πρώτη φορά αναπτύχτηκε αρχικά από την Google και εν συνεχεία από την Open Handset Alliance και βασίζεται στο λειτουργικό σύστημα linux. Η Google έχει αναπτύξει ειδικές βιβλιοθήκες λογισμικού που περιλαμβάνουν εντολές ελέγχου των συσκευών και των δυνατοτήτων τους (όπως κάμερα, Bluetooth κ.α.) μέσω των οποίων επιτρέπεται στους προγραμματιστές να συνθέσουν εφαρμογές σε μια προσαρμοσμένη έκδοση της γλώσσας προγραμματισμού Java οι οποίες κάνουν τις συσκευές πιο λειτουργικές και τις φέρνουν πιο κοντά στις ανάγκες του κάθε χρήστη. Το λογότυπο του είναι ένα ρομπότ πράσινου χρώματος.

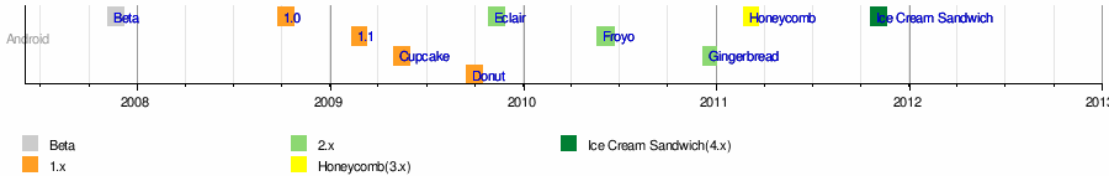
Εικόνα 2.1: Το λογότυπο του Android

2.2 Ιστορική εξέλιξη

Φέτος το android κλείνοντας 8 χρόνια κυκλοφορίας είναι το πιο γρήγορα αναπτυσσόμενο λειτουργικό σύστημα στον κόσμο. Λίγο πριν την επίσημη κυκλοφορία του υπήρξαν τουλάχιστον 2 ανεπίσημες εκδόσεις στο εσωτερικό των Google και OHA. Στη συνέχεια και συγκεκριμένα στις 5 Νοεμβρίου του 2007 παρουσιάστηκε η πρώτη έκδοση -Beta- και τότε ¨γεννήθηκε¨ επίσημα το android. Το android 1.0/1.1 κυκλοφόρησε στις 23 Σεπτεμβρίου του 2008 και το πρώτο κινητό με λειτουργικό android ήταν το G1 της T-mobile.

Εικόνα 2.2: T-mobile G1

Λόγω της open source φύσης του παρουσίασε ραγδαία εξέλιξη και αυτό φαίνεται από το γεγονός ότι 7 πολύ σημαντικές εκδόσεις του κυκλοφόρησαν μέσα σε 2.5 χρόνια.



Εικόνα 2.3: Χρονοδιάγραμμα Εκδόσεων του Android OS

2.3 Εκδόσεις Android

Αν και οι εκδόσεις 1.0 και 1.1 δεν είχαν κάποια κωδικοποιημένη ονομασία στη συνέχεια οι εκδόσεις του απέκτησαν ονόματα γλυκών ξεκινώντας από την έκδοση 1.5 Cupcake και το γράμμα “C”. Με αυτό τον τρόπο κάθε νέα έκδοση έχει ένα όνομα γλυκού με αρχικό γράμμα το επόμενο της αλφαβήτου (π.χ. Donut 1.6, Eclair 2.0-2.1).

2.3.1 Android 1.5 Cupcake

Εικόνα 2.4: Το λογότυπο του Android 1.5 Cupcake

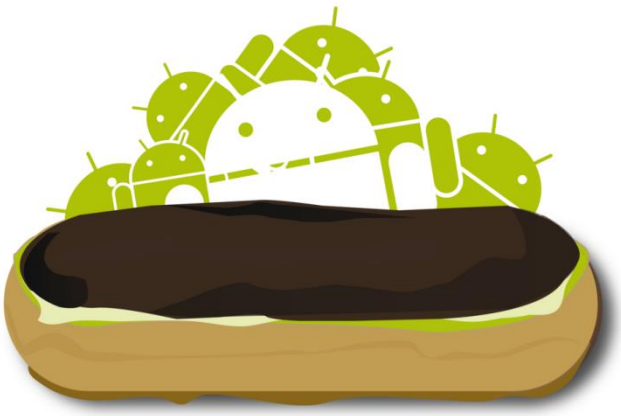
Η έκδοση Android 1.5 Cupcake έκανε την εμφάνιση της στις 30 Απριλίου 2009 και είναι μια από τις σημαντικότερες αναβαθμίσεις του καθώς σε αυτήν εμφανίζεται νέο ψηφιακό πληκτρολόγιο το οποίο έφερε ριζική αλλαγή στο σχεδιασμό των νέων smartphones αφαιρώντας το φυσικό πληκτρολόγιο QWERTY από αυτά. Επίσης, προστέθηκαν νέες λειτουργίες όπως η καταγραφή βίντεο από την κάμερα του κινητού, η παρακολούθησή τους καθώς και η άμεση μεταφόρτωση τους στο Youtube.

2.3.2 Android 1.6 Donut

Εικόνα 2.5: Το λογότυπο του Android 1.6 Donut

Η έκδοση Android 1.6 Donut παρουσιάστηκε στις 15 Σεπτεμβρίου 2009. Μετά από αυτή την έκδοση τα android μπορούν πλέον να υποστηριχτούν σε διαφορετικές αναλύσεις οθόνης ανεξάρτητα από την πυκνότητα των pixels. Χάρη σε αυτό το γεγονός το λογισμικό αυτό το βλέπουμε σε κινητά τηλέφωνα διαφορετικών κατηγοριών. Σε αυτήν την έκδοση ξεκινά να υποστηρίζεται η επιλογή πολλαπλών αρχείων ταυτόχρονα κάτι που πλέον θεωρούμε δεδομένο. Επιπλέον, πρωτοεμφανίστηκε η μπάρα αναζήτησης της Google στην αρχική οθόνη του κινητού και επανασχεδιάστηκε το android market.

2.3.3 Android 2.0 - 2.1 Eclair



Εικόνα 2.6: Το λογότυπο του Android 2.0 Eclair

Η έκδοση Android 2.0 Eclair εμφανίστηκε στις 26 Οκτωβρίου 2009 και επανεκδόθηκε σε Android 2.1 Eclair τον Ιανουάριο του 2010. Σε αυτήν την έκδοση βελτιώθηκε σημαντικά η λειτουργία της κάμερας με λειτουργίες όπως την ενσωμάτωση LED flash, χρωματικών εφέ, λειτουργίας σκηνών, ρυθμίσεων ισορροπίας λευκού, ψηφιακό ζουμ κ.α.. Επίσης, σημαντικές αλλαγές υπήρξαν και στους χάρτες Google με δυνατότητα πλέον φωνητικής καθοδήγησης και οδηγιών στροφή-στροφή. Τέλος βελτιώθηκε η δυνατότητα διαμόρφωσης της αρχικής οθόνης.

2.3.4 Android 2.2 Froyo

Εικόνα 2.7: Το λογότυπο του Android 2.2 Froyo

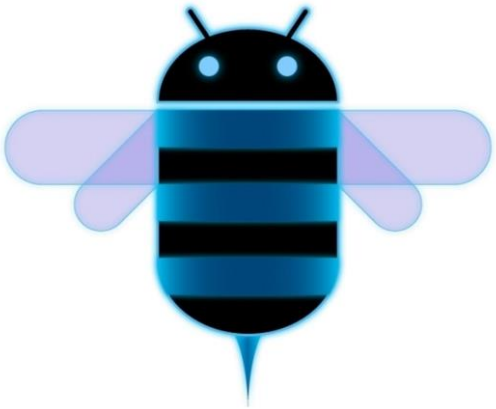
Η έκδοση Android 2.2 Froyo κυκλοφόρησε στις 20 Μαΐου 2010 μια επίσης πολύ σημαντική ενημέρωση λόγω της αλλαγής μεταγλωττιστή και την εμφάνιση του Dalvic που έφερε καίριες βελτιώσεις σε απόδοση και ταχύτητα. Σύμφωνα με την Google οι συσκευές android έπειτα από αυτό γίνονται 2 έως 5 φορές ταχύτερες. Επιπρόσθετα, ενσωματώνεται η λειτουργία Wi-Fi Hotspot, ο έλεγχος χρήσης πακέτων δεδομένων του παρόχου κινητής τηλεφωνίας από το χρήστη. Τέλος σημαντική αναβάθμιση είναι η δυνατότητα μεταφοράς των εφαρμογών από τον χώρο αποθήκευσης της συσκευής στην κάρτα μνήμης.

2.3.5 Android 2.3 - 2.3.3 Gingerbread

Εικόνα 2.8: Το λογότυπο του Android 2.3 Gingerbread

Η έκδοση Android 2.3 Gingerbread εμφανίστηκε στις 6 Δεκεμβρίου 2010 και επανεκδόθηκε σε Android 2.3.3 Gingerbread στις 9 Φεβρουαρίου του 2011. Είναι μια από τις δημοφιλέστερες εκδόσεις android με σημαντικότερη αναβάθμιση την προσθήκη αισθητήρων (γυροσκόπιο, βαρόμετρο κ.α.) κάτι που έδωσε τεράστιες δυνατότητες ανάπτυξης εφαρμογών και παιχνιδιών για το κινητό μας. Επίσης, αναβαθμίστηκε ο έλεγχος της μπαταρίας και εμφανίστηκαν σημαντικές λειτουργίες όπως το NFC και η δυνατότητα κλήσεων μέσω Internet (VoIP).

2.3.6 Android 3.0 Honeycomb



Εικόνα 2.9: Το λογότυπο του Android 3.0 Honeycomb

Η έκδοση Android 3.0 Honeycomb παρουσιάστηκε στις 9 Μαΐου του 2011 και ήταν μια έκδοση η οποία δημιουργήθηκε κυρίως για tablets. Κύριες αλλαγές η βελτίωση του περιβάλλοντος χρήστη και του multitasking καθώς και η υποστήριξη επεξεργαστών δυο και τεσσάρων πυρήνων.

2.3.7 Android 4.0 Ice Cream Sandwich

Εικόνα 2.10: Το λογότυπο του Android 4.0 ICS

Η έκδοση Android 4.0 Ice Cream Sandwich κυκλοφόρησε στις 18 Οκτωβρίου 2011. Το ICS ήταν μια αναβάθμιση που έκανε το android ένα ολοκληρωμένο λειτουργικό σύστημα οπότε η αναβάθμισή του άρχισε σταδιακά να επιβραδύνεται. Βασικό κομμάτι σε αυτήν την ενημέρωση είναι η αλλαγή του περιβάλλοντος χρήστη με την σχεδίαση Holo UI. Επίσης, έχουμε προσθήκη της αναγνώρισης προσώπου στην ασφάλεια της συσκευής. Τέλος, βελτιώθηκε η λειτουργία NFC με τη χρήση του Android Beam καθώς και η κάμερα με την δυνατότητα εγγραφής βίντεο σε ανάλυση 1080p.

2.3.8 Android 4.1 - 4.2 - 4.3 Jelly Bean

Εικόνα 2.11: Το λογότυπο του Android 4.1 Jelly Bean

Όπως είπαμε μετά την έκδοση ICS η ανάπτυξη του συστήματος άρχισε να επιβραδύνεται και αυτό είναι εμφανές ακόμα και από την αμέσως επόμενη έκδοση που είναι το Jelly Bean η οποία έμεινε στην αγορά για περισσότερο από 15 μήνες μέχρι να έρθει η επόμενη(KitKat). Η έκδοση Android Jelly Bean εμφανίστηκε για πρώτη φορά ως 4.1 στις 9 Ιουλίου 2012, επανεκδόθηκε στις 13 Νοεμβρίου 2012 ως 4.2 και η τελευταία της αναβάθμιση έγινε στις 24 Ιουλίου 2013 ως 4.3. Συνοπτικά στις τρεις αναβαθμίσεις του έχουμε αλλαγές στην απόδοση και την ταχύτητα του λογισμικού, αλλαγή στην εφαρμογή ρολογιού με ενσωματωμένο παγκόσμιο ρολόι και χρονόμετρο και αλλαγές στις ειδοποιήσεις με λειτουργία άμεσης ενέργειας από το χρήστη στο χώρο των ειδοποιήσεων. Οι πιο σημαντικές συνεισφορές του στο Android όμως είναι άλλες. Στο Android Jelly Bean βλέπουμε για πρώτη φορά την δυνατότητα ύπαρξης πολλών διαφορετικών χρηστών σε μια συσκευή ο καθένας με το δικό του περιβάλλον χρήσης και ρυθμίσεις όπως αυτό υπάρχει στους Η/Υ. Τέλος, η νέα δυνατότητα που ξεχωρίζει είναι το Google Now ένας ψηφιακός βοηθός μέσω του οποίου η Google παρέχει στο χρήστη άμεσες απαντήσεις σε ότι ρωτήσει μέσω του διαδικτυακού ιστού, πληροφορίες σχετικά με

τον καιρό καθώς και νέα και ειδήσεις ανάλογα με τα ενδιαφέροντα του χρήστη και την περιοχή στην οποία βρίσκεται.

2.3.9 Android 4.4 KitKat

Εικόνα 2.12: Το λογότυπο του Android 4.4 KitKat

Η έκδοση Android 4.4 KitKat έκανε την εμφάνισή της στις 31 Οκτωβρίου 2013. Υπήρξαν ριζικές αλλαγές στο περιβάλλον χρήστη και στις επιδόσεις του λογισμικού με σκοπό το Android να μπορεί να κυκλοφορήσει χωρίς προβλήματα και σε συσκευές χαμηλού κόστους. Και το πέτυχε καθώς ακόμα και συσκευές με 512mb RAM μπορούσαν να “αντέξουν” τα Android KitKat. Επίσης, σε αυτή την έκδοση εξελίχτηκε ριζικά το Google Now με την καινούργια λειτουργία OK Google που επέτρεπε στο χρήστη να κάνει πράγματα στο κινητό του χωρίς καν να αγγίξει τη συσκευή, απλά με φωνητικές εντολές. Τέλος, προστέθηκε και η λειτουργία της ασύρματης εκτύπωσης.

2.3.10 Android 5.0 Lollipop

Εικόνα 2.13: Το λογότυπο του Android 5.0 Lollipop

Η έκδοση Android 5.0 Lollipop κυκλοφόρησε στις 12 Νοεμβρίου 2014. Για άλλη μια φορά η Google προχώρησε σε μια ριζική αλλαγή στο περιβάλλον χρήστη και διαμόρφωσε το Material Design, μια επίπεδη σχεδίαση που έδωσε απλότητα και περισσότερη ευκολία στην χρήση του Android. Επίσης, προστέθηκε η δυνατότητα να συνδέονται άμεσα μεταξύ τους όλες οι συσκευές του χρήστη που έχουν Android Lollipop (τηλέφωνο, ταμπλέτα, ρολόι, Android τηλεόραση κ.α.) με τη λειτουργία Multiscreen και να μοιράζονται από αρχεία, ρυθμίσεις μέχρι και το ιστορικό αναζήτησης. Τέλος, από αυτήν την έκδοση ξεκίνησαν να υποστηρίζονται επεξεργαστές 64-bit για τις συσκευές.

2.3.11 Android 6.0 Marshmallow

Εικόνα 2.14: Το λογότυπο του Android 6.0 Marshmallow

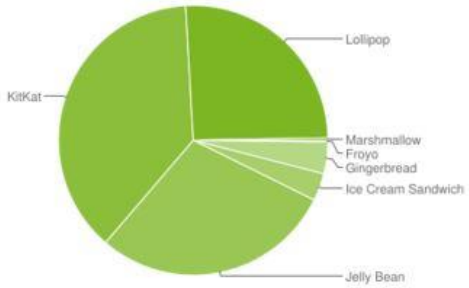
Φτάσαμε στην τελευταία μέχρι σήμερα έκδοση του λογισμικού Android με την Android 6.0 Marshmallow να κυκλοφορεί στις 5 Οκτωβρίου 2015 και να παρουσιάζεται ως η πιο πλήρης έκδοση Android που έχουμε δει μέχρι σήμερα. Έγιναν μεγάλες και ριζικές αλλαγές που δίνουν νέες δυνατότητες στο χρήστη και ανεβάζει το επίπεδο των κινητών συσκευών. Αναβαθμίστηκε εκ νέου το Google Now με τη λειτουργία “Now on tap”. Έτσι ο χρήστης, μπορεί σε οποιαδήποτε εφαρμογή και αν είναι, κρατώντας πατημένο το κεντρικό κουμπί της συσκευής, να εμφανίσει το Google Now με άμεσα διαθέσιμες πληροφορίες σχετικά με αυτό που έβλεπε χωρίς να βγει από την εφαρμογή. Μεγάλη αλλαγή επίσης έχει γίνει και με τις άδειες που ζητούν οι εφαρμογές που κατεβάζουμε από το Play Store. Παλαιότερα η κάθε εφαρμογή, προκειμένου να τρέξει στη συσκευή, ζητούσε τις άδειες που χρειαζόταν όλες μαζί, ίσως και περισσότερες. Τώρα πλέον, ο ίδιος ο χρήστης επιλέγει τι άδειες θα δώσει σε κάθε εφαρμογή και μπορεί να περιορίσει έτσι την πρόσβασή της σε όποια σημεία της συσκευής του επιθυμεί. Προστέθηκε επίσης η λειτουργία Android Pay μέσω της οποίας μπορεί πλέον ο χρήστης να κάνει πληρωμές μέσω κινητού και με ασφάλεια, καθώς λειτουργεί με πιστοποίηση δακτυλικού αποτυπώματος. Άλλη μια κύρια λειτουργία είναι το Doze. Το Doze

είναι μια νέα τεχνολογία που μέσω των αισθητήρων “καταλαβαίνει” πότε είναι ανενεργή η συσκευή για ορισμένο χρόνο και “κοιμίζει” τις εφαρμογές παρασκηνίου επιτυγχάνοντας έτσι, μεγάλη εξοικονόμηση ενέργειας στη μπαταρία της συσκευής. Τέλος, αλλαγές έχουμε και στο περιβάλλον χρήσης με κάθετη πλέον λίστα εφαρμογών, υποστήριξη του νέου usb type-c καθώς και νέες καρτέλες στις ρυθμίσεις, οι οποίες δίνουν πολλές δυνατότητες στον χρήστη για τη διαχείριση της συσκευής του.

2.4 Προγραμματισμός σε Android

Για να φτιάξει κάποιος την δική του εφαρμογή χρειάζεται κάποια εργαλεία και κάποιες πληροφορίες. Η Google δίνει μεγάλη βοήθεια σε όποιον ασχολείται με την ανάπτυξη εφαρμογών android και μάλιστα έχει φτιάξει μια ιστοσελίδα για τους προγραμματιστές android. Η ιστοσελίδα (developer.android.com) παρέχει πολλές πληροφορίες για το πώς κάποιος μπορεί να ξεκινήσει από το μηδέν να φτιάχνει μια εφαρμογή, δίνονται όλα τα απαραίτητα εργαλεία όπως το Android Studio και τα αρχεία που θα χρειαστεί στα οποία θα αναφερθούμε παρακάτω. Επίσης, κάθε μήνα η Google εφοδιάζει τους προγραμματιστές της με πληροφορίες σχετικά με την έκδοση android, το μέγεθος οθόνης κ.α. που χρησιμοποιούνται περισσότερο ώστε να ξέρει ο προγραμματιστής και να επιλέξει το κοινό που θα έχει απήχηση η εφαρμογή του. Κάτι τέτοιο φαίνεται και στην εικόνα που ακολουθεί και δίνει την δυνατότητα στον προγραμματιστή να δει ποιές εκδόσεις χρησιμοποιούνται ανά τον κόσμο και σε τι ποσοστό. Έτσι μπορεί να επιλέξει να φτιάξει, για παράδειγμα, μια εφαρμογή η οποία να μπορεί να “τρέχει” σε Android Jelly Bean ή μεταγενέστερα και όχι σε ICS ή προηγούμενα που έχουν μικρά ποσοστά πια. Και αυτό είναι πολύ σημαντικό, καθώς, κάποιος που φτιάχνει μια εφαρμογή μόνο για την τελευταία έκδοση Android θα χάσει μεγάλο ποσοστό συσκευών, οι οποίες θα αδυνατούν να κατεβάσουν την εφαρμογή του.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	3.3%
4.1.x	Jelly Bean	16	11.0%
4.2.x		17	13.9%
4.3		18	4.1%
4.4	KitKat	19	37.8%
5.0	Lollipop	21	15.5%
5.1		22	10.1%
6.0	Marshmallow	23	0.3%



Εικόνα 2.15: Ποσοστό χρήσης εκδόσεων Android (Νοέμβριος 2015)

ΚΕΦΑΛΑΙΟ 3 - ΕΠΙΛΟΓΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΑΝΑΠΤΥΞΗΣ

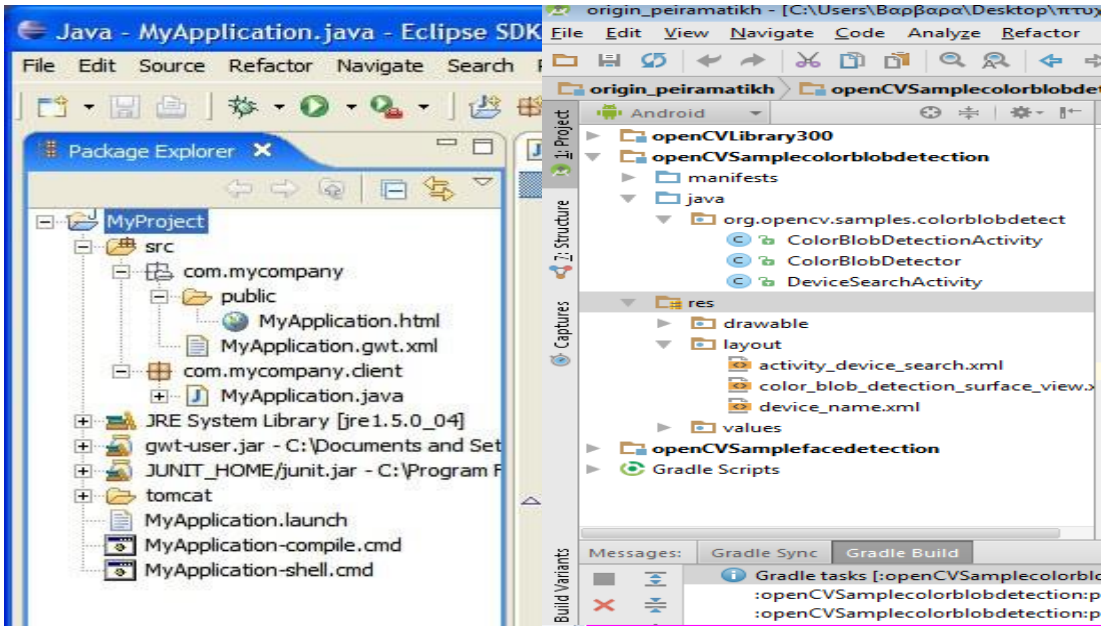
Για την δημιουργία android εφαρμογών(apps) χρειαζόμαστε το κατάλληλο περιβάλλον, έτσι, ώστε να μπορούμε να δουλέψουμε με ευχέρεια και να αποφύγουμε, όσο μπορούμε, απρόσμενες αρνητικές καταστάσεις. Για την δημιουργία ενός τέτοιου περιβάλλοντος θα πρέπει να ακολουθήσουμε τα εξής βήματα.

Θα πρέπει να αποφασίσουμε αρχικά, σε τι περιβάλλον ανάπτυξης θα ξεκινήσουμε να δημιουργούμε τις εφαρμογές μας. Μπορούμε να δουλέψουμε είτε σε **Android Studio** είτε σε **Eclipse**.

3.1 Eclipse VS. Android

##1 Workspace

Eclipse **Android Studio**



Εικόνα 3.1 Eclipse VS Android

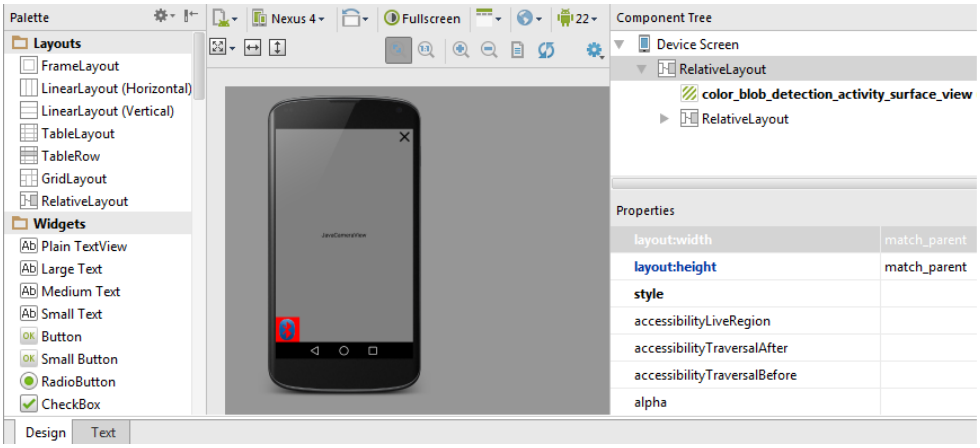
Το workspace διαφέρει πολύ στα δύο αυτά προγράμματα ανάπτυξης κώδικα. Το eclipse κάνει χρήση βιβλιοθηκών και modules από το αρχείο με την κατάληξη .jar. Το android studio, όπως βλέπουμε παραπάνω, δεν χρειάζεται το αρχείο .jar καθώς καταλαβαίνει από μόνο του τα modules και τις βιβλιοθήκες.

##2 run,tested,debugged(Ενότητες)

Οι παραπάνω ενότητες είναι παρόμοιες μεταξύ των δύο προγραμμάτων αλλά με μικρές σημαντικές διαφορές. Στο android studio, η κάθε ενότητα έχει ένα δικό της Gradle (δημιουργείται αυτόματα και βοηθάει στην αποσφαλμάτωση του κώδικα και του σωστού τρόπου εκτέλεσης της εφαρμογής). Επίσης, το αρχείο αυτό

περιέχει σημαντικές πληροφορίες, όπως, ποια σειρά από εκδόσεις Android υποστηρίζει το έργο μας, καθώς και εξαρτήσεις των δεδομένων του.

##3 Design

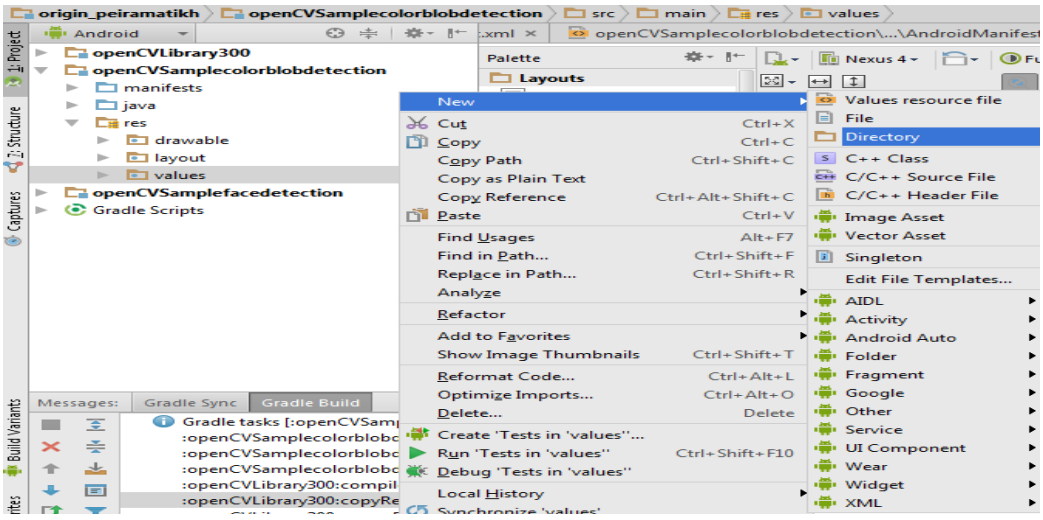


Εικόνα 3.2 Design

Το Android Studio διαθέτει ένα νέο και βελτιωμένο design, όπου μπορούμε να δούμε το περιβάλλον στο οποίο εργαζόμαστε, τι έχουμε φτιάξει μέχρι στιγμής και το πώς θέλουμε να εμφανίζεται η εφαρμογή μας στο κινητό. Το Eclipse έχει και αυτό παρόμοιο design αλλά όχι τόσο καλά εξοπλισμένο και όχι τόσο εύχρηστο.

##4 Προσθήκη μιας βιβλιοθήκης

Στο Eclipse για να εισάγουμε μια νέα βιβλιοθήκη θα πρέπει να πάμε στο .jar και να πατήσουμε <<Add As library>> και μετά να φροντίσουμε να συγχρονίσουμε βιβλιοθήκη και project. Ενώ, στο Android Studio, απλά πάμε στο values->new->directory->(όνομα directory)libs, έτσι φτιάχνουμε βιβλιοθήκη και μετά την γεμίζουμε με ό,τι θέλουμε. Στην συνέχεια, κάνουμε ένα κλικ στο <<Sync Gradle>> προκειμένου το έργο μας να συγχρονιστεί με αυτήν την βιβλιοθήκη.



Εικόνα 3.3 Προσθήκη μιας βιβλιοθήκης

##5 Εξαρτήσεις μεταξύ των Ενοτήτων

Ακριβώς όπως και στο Eclipse, έτσι και στο Android Studio μία μονάδα μπορεί να εξαρτάται από μια άλλη ενότητα. Στο eclipse κάνουμε συγκεκριμένες ενέργειες έτσι ώστε να το πετύχουμε αυτό, ενώ, αντίθετα, στο Android Studio δημιουργείται αυτόματα το απαραίτητο **Gradle** που χρειαζόμαστε.

##6 Manifest Destiny

Μια σημαντική αλλαγή από το Eclipse στο Android Studio είναι η προσθήκη του Android Manifest. Το Android Manifest είναι ο χώρος όπου δηλώνουμε το τι θέλουμε να ισχύει στο πρόγραμμά μας (π.χ. “debuggable=true”-> δυνατότητα να κάνουμε debug στον κώδικα). Παρόλα αυτά, αυτή η διαδικασία γίνεται αυτόματα στο Manifest και απλά αλλάζουμε την τιμή από <<true>> σε <<false>> αναλόγως αν θέλουμε να συμβαίνει κάτι ή όχι.

##7 Η μετάβαση από Android Studio σε Eclipse

Εδώ τα πράγματα είναι λίγο πιο δύσκολα. Κύριος λόγος είναι ότι, το gradle αρχειο δεν μεταφέρεται στο eclipse και είναι αυτό που έχει όλες τις πληροφορίες της εφαρμογής μας. Θα πρέπει να γίνονται όλα χειροκίνητα. Αυτά που θα πρέπει να γνωρίζουμε είναι:

- Πολλές πληροφορίες του έργου είναι στο settings.gradle
- Το settings.gradle είναι ένα αρχείο το οποίο ενημερώνει αυτόματα όλες τις κλάσεις του συστήματος όταν εισάγουμε κάτι καινούργιο
- Κάθε ενότητα στο Android Studio έχει το δικό της build.gradle

3.2 Ποια η εκτίμηση μας

Παρακάτω σας παρουσιάζουμε την εκτίμηση μας, με παραδείγματα, σχετικά με το πρόγραμμα που θα επιλέξουμε.

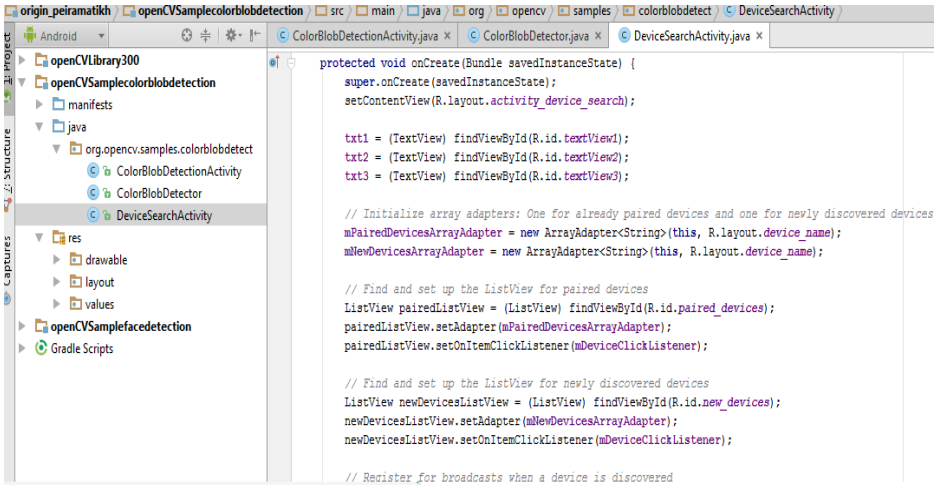
##1 Drag and Drop Gui (Σχεδιαστής)

Το Android Studio στο τμήμα του σχεδιασμού έχει το πλέον γνωστό σε όλους **<<drag and drop>>.** Δηλαδή έτοιμα σχέδια, σχήματα και ιδιότητες που για να τις ενσωματώσουμε, απλά τις σέρνουμε από την παλέτα. Στο eclipse τις φτιάχνουμε μόνοι μας. Βέβαια, τώρα μιλάμε για υπόθεση 2 λεπτών παραπάνω. Οπότε, αν δεν θέλουμε κάτι ιδιαίτερο από πλευρά design σε γρήγορο χρόνο, τότε δεν έχουν και μεγάλη διαφορά, καθώς σε μία σοβαρή εφαρμογή το μεγαλύτερο κομμάτι του χρόνου μας το δαπανούμε στον κώδικα.

Εκτίμηση: Ισοπαλία

##2 Οργάνωση και ανάπτυξη κώδικα, βοήθεια σε error και συμπλήρωση κώδικα

Το Android Studio έχει ολοκλήρωση κώδικα. Με αυτόν τον όρο, εννοούμε την οργάνωση και την ανάπτυξη κώδικα, δηλαδή, την βοήθεια που χρειαζόμαστε, ώστε να πάμε στο επόμενο βήμα, όταν βλέπουμε κάποιο σφάλμα(error).Σε περίπτωση που έχουμε κάποιο συντακτικό λάθος στον κώδικά μας υπάρχει η αυτόματη μετατροπή του λάθους σε σωστό. Επίσης, αν χρησιμοποιήσουμε μια μεταβλητή που δεν έχει δηλωθεί με πάτημα 2 πλήκτρων **<<alt + enter>>** , δείχνει τις επιλογές που έχουμε, ώστε να τρέχει κανονικά. Το σημαντικότερο από όλα είναι, η επιλογή objects ή classes όταν θέλουμε να αναφερθούμε σε εκείνη την συγκεκριμένη κλάση. Δεν είμαστε υποχρεωμένοι να βρούμε το ακριβές όνομα της κλάσης.



Εικόνα 3.4 Ολοκλήρωση κώδικα

Εκτίμηση: Android Studio

##3 Gradle

Στο Android studio, όπως αναφέραμε, είναι ένα πολύτιμο εργαλείο, καθώς, είναι υπεύθυνο για το debug και τον συγχρονισμό ολόκληρου του προγράμματος, απλά με ένα πάτημα. Το καλύτερο είναι ότι, δεν χρειάζεται να το φτιάξουμε και να γράψουμε κάτι μέσα, διότι δημιουργείται και γράφεται αυτόματα με την δημιουργία του project και των αλλαγών που κάνουμε στην πορεία. Ενώ, στο Eclipse πρέπει να το δημιουργήσουμε και όταν αλλάζουμε κάτι στον κώδικά μας, να πηγαίνουμε και εκεί να κάνουμε τις απαραίτητες αλλαγές. Με αυτόν τον τρόπο, υλοποιούμε εύκολα και γρήγορα το έργο μας.

Εκτίμηση: Android Studio

##4 Cloud platform

Το Android Studio υποστηρίζει την **cloud platform της Google,** η οποία μας παρέχει την δυνατότητα να χρήσης της **<<Google App Engine>>.** Το **Google App Engine** είναι το εργαλείο που μας επιτρέπει τη δημιουργία και την εκτέλεση

εφαρμογών πάνω στην Google. Τα πλεονεκτήματα που έχουμε από την χρήση ενός τέτοιου εργαλείου είναι:

- Η εύκολη δημιουργία project
- Η εύκολη συντήρηση του
- Γρήγορη αναβάθμιση στις ανάγκες της σύγχρονης εποχής

Επίσης, αξίζει να σημειωθεί ότι, με την χρήση App Engine δεν χρειάζονται οι servers για να διατηρηθεί.

Όλα αυτά που αναφέραμε, μπορούν να γίνουν και στο Eclipse με το εργαλείο Google Plugin για Eclipse.

Εκτίμηση: Ισοπαλία

##5 Περιβάλλον χρήστη

Το Eclipse είναι μεγαλύτερο σαν πρόγραμμα και ήταν δημοφιλές πριν από μια δεκαετία. Όμως, τα τελευταία χρόνια δείχνει να έχει ξεπερασθεί για την ανάπτυξη Android εφαρμογών. Επίσης, χτίστηκε για να είναι ένα εργαλείο IDE, για οποιαδήποτε χρήση και σε οποιαδήποτε γλώσσα (αν και η java είναι η κύρια). Ενώ, το Android Studio είναι ένα νέο πρόγραμμα, το οποίο δημιουργήθηκε τα τελευταία χρόνια και συγκεκριμένα, για να είναι ένα οικείο περιβάλλον έτσι ώστε, να βοηθάει τον προγραμματιστή να δημιουργεί android applications με μεγαλύτερη ευκολία και με μεγαλύτερη γκάμα εργαλείων.

Εκτίμηση: Android Studio

Τελική εκτίμηση: Android Studio

3.3 Σύγκριση Android / ios / Windows

Το κάθε λειτουργικό κινητής συσκευής έχει την αντίστοιχη επίσημη υποδομή (IDE & SDK) για την δημιουργία εγγενών εφαρμογών για αυτό.

ANDROID: Για το Android υπάρχει το πακέτο λογισμικού **Android Development Tools** το οποίο περιλαμβάνει:

- **Android SDK**
- Γλώσσα προγραμματισμού **JAVA**
- Μια τροποποιημένη έκδοση του προγράμματος **Eclipse** ως **IDE**
- **Λειτουργεί σε όλα τα γνωστά λειτουργικά συστήματα**(Windows,Linux,Mac OS X)

iOS: Για το iOS υπάρχει το πακέτο λογισμικού <<Εργαλεία Προγραμματιστή iOS>> (iOS Developer Toolset) το οποίο περιλαμβάνει:

- **iOS SDK**
- Γλώσσα προγραμματισμού **Objective-C**(υπερσύνολο της C)
- Το πρόγραμμα **XCode** ως **IDE**
- Λειτουργεί **μόνο** στα πλαίσια του λειτουργικού **Mac OS X**

WINDOWS PHONE: Για το Windows Phone υπάρχει το πακέτο λογισμικού **Windows Phone SDK** το οποίο περιλαμβάνει:

- **Windows Phone SDK**
- Γλώσσα προγραμματισμού **οποιαδήποτε** υποστηρίζετε από το πλαίσιο λογισμικού**(Software Framework) .NET**
- Το πρόγραμμα **Visual Studio Express** ως **IDE**
- Λειτουργεί **μόνο** στα πλαίσια του λειτουργικού **Microsoft Windows**

Επίσης, κανένα πακέτο λογισμικού δεν μοιράζεται την ίδια γλώσσα προγραμματισμού με τα άλλα, επομένως το έργο των προγραμματιστών γίνεται ακόμα πιο δύσκολο.

Επομένως, η δημιουργία μιας εγγενούς Android εφαρμογής γίνεται με μηδενικό κόστος, καθώς οι εκδόσεις Linux είναι δωρεάν.

Σε πλήρη αντίθεση με τις Android εφαρμογές, είναι οι εφαρμογές σε iOS και Windows Phone. Και αυτό γιατί, οι εγγενής εφαρμογές iOS προϋποθέτουν την ύπαρξη ενός Apple υπολογιστή, έτσι ώστε, να έχει στην κατοχή του το λειτουργικό Mac OS X. Έτσι, και οι εγγενής εφαρμογές Windows Phone προϋποθέτουν την κατοχή άδειας του λειτουργικού Microsoft Windows. Το αποτέλεσμα είναι ότι, αυτές οι δύο στοιχίζουν. Επίσης, το κόστος δημοσίευσης μιας εφαρμογής Android είναι κατά πολύ πιο οικονομική από τις άλλες δύο. Στην συνέχεια, θα σας δείξουμε έναν συγκριτικό πίνακα εργαλείων ανάπτυξης κινητών εφαρμογών.

	Android SDK	iOS SDK	Windows Phone SDK	Titanium SDK	JQuery Mobile + Apache Cordova
Εγγενής Εφαρμογές	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ
Άδεια Χρήσης	ΔΩΡΕΑΝ	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ
Γλώσσα	Java	Objective-C	C#, C++/ CLI, F#, VB.NET	JavaScript	HTML5 + JavaScript + CSS3
IDE	ΔΩΡΕΑΝ	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ*	ΔΩΡΕΑΝ
Έλεγχος Εφαρμογών (App Review)	ΟΧΙ	ΝΑΙ	ΝΑΙ	Εξαρτάται από τις αγορές δημοσίευσης	Εξαρτάται από τις αγορές δημοσίευσης
Κόστος δημοσίευσης	25$ εφάπαξ	99$/ έτος	99$/έτος	Εξαρτάται από τις αγορές δημοσίευσης	Εξαρτάται από τις αγορές δημοσίευσης
Μέγεθος Οικοσυστή-ματος	ΠΟΛΥ ΜΕΓΑΛΟ	ΠΟΛΥ ΜΕΓΑΛΟ	ΠΟΛΥ ΜΕΓΑΛΟ	ΜΙΚΡΟ	ΜΕΣΑΙΟ
Τεκμηρίωση (Documenta-tion)	ΠΟΛΥ ΚΑΛΗ	ΠΟΛΥ ΚΑΛΗ	ΠΟΛΥ ΚΑΛΗ	ΚΑΚΗ	ΚΑΛΗ
Ταχύτητα εφαρμογής/ Αποκρισιμό-τητα	ΠΟΛΥ ΚΑΛΗ	ΠΟΛΥ ΚΑΛΗ	ΠΟΛΥ ΚΑΛΗ	ΚΑΛΗ	ΚΑΚΗ
Υποστηριζό-μενα Λειτουργικά	Android	iOS	Windows Phone	Android, iOS	Android, iOS, Windows Phone
Μερίδιο Αγοράς	> 70%	21-22%%	< 5%	Εξαρτάται από τις αγορές δημοσίευσης	Εξαρτάται από τις αγορές δημοσίευσης

Πίνακας 3.1:Συγκριτικός πίνακας εργαλείων ανάπτυξης κινητών εφαρμογών

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΚΕΦΑΛΑΙΟ 4 - ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ANDROID DEVELOPMENT ENVIRONMENT

Για να μπορέσουμε να ξεκινήσουμε την ανάπτυξη Android εφαρμογών, θα πρέπει να έχουμε δημιουργήσει το κατάλληλο περιβάλλον για την εργασία μας. Ξεκινώντας από μηδενική βάση, το πρώτο βήμα που πρέπει να πραγματοποιήσουμε είναι η εγκατάσταση στον υπολογιστή μας του Android Studio (Latest Version) μαζί με όλα τα απαραίτητα εργαλεία που θα χρειαστούμε (π.χ. βιβλιοθήκες, πλατφόρμες κτλ.)

Η ανάπτυξη Android εφαρμογών στην πτυχιακή μας βασίζεται σε Windows λειτουργικό σύστημα. Επομένως, όλα τα εργαλεία που θα κατεβάσουμε και θα εγκαταστήσουμε θα είναι σε εκδόσεις που ανταποκρίνονται σε Windows λειτουργικό σύστημα. Στην περίπτωση που θα χρησιμοποιούσαμε κάποιο άλλο λειτουργικό σύστημα, οι αλλαγές θα ήταν ελάχιστες και πιο συγκεκριμένα θα ακολουθούσαμε ακριβώς τα ίδια βήματα αλλά, αντί να κατεβάζαμε τα εργαλεία για Windows θα τα κατεβάζαμε για το δικό μας λειτουργικό σύστημα.

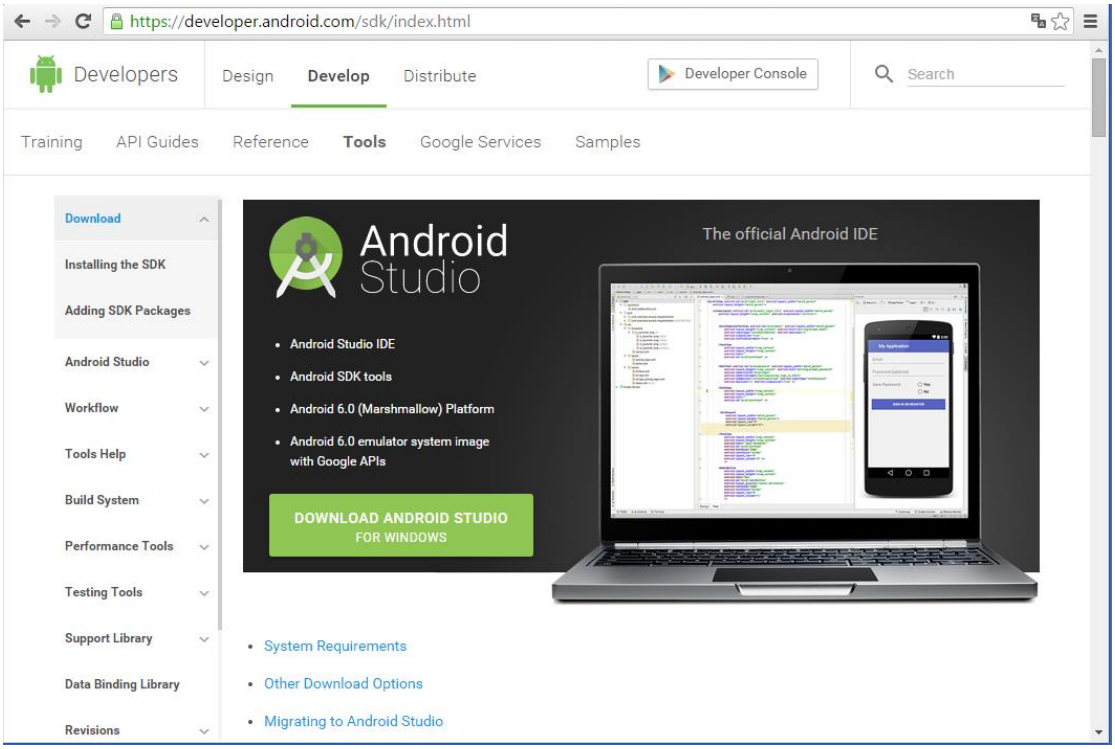
Στο περιβάλλον ανάπτυξης Android εφαρμογών έχουμε τρία πολύ βασικά εργαλεία, τα οποία παρέχονται και τα τρία, δωρεάν. Αυτά είναι τα java, eclipse, Android Studio.

Όλες οι Android εφαρμογές και η ανάπτυξη τους στηρίζονται στην standard version, που είναι η **<<Java Platform Standard Edition – Java SE>>** δηλαδή, γλώσσα προγραμματισμού java. Στα παρακάτω βήματα θα δείτε με ποιο τρόπο θα κατεβάσουμε αυτήν την version από την **Oracle**.

4.1 Download Android Studio

Αρχικά, ανοίγουμε τον browser που έχουμε στον υπολογιστή μας και πληκτρολογούμε <<free download android studio for windows>>, στην συνέχεια, στην πρώτη επιλογή που εμφανίζεται κάνουμε κλικ ώστε, να μπούμε στην σελίδα.

Έπειτα, θα μεταβούμε στη σελίδα, η οποία θα είναι σαν την εικόνα 4.1 που ακολουθεί.



Εικόνα 4.1 Android Studio

Link for download Android studio -> https://developer.android.com/sdk/index.html

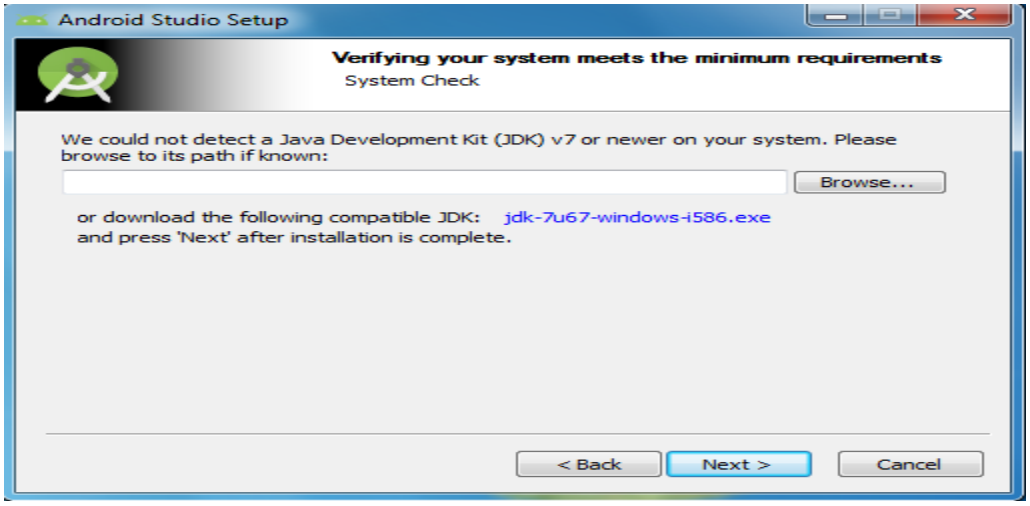
Στην συνέχεια πατάμε το πράσινο κουμπί που γράφει <<DOWNLOAD ANDROID STUDIO FOR WINDOWS>>.

4.2 Εγκατάσταση Android Studio

Εφόσον έχουμε διαβάσει και αποδεχθεί τους όρους και τις πολιτικές, επιλέγουμε το κουτάκι και είμαστε έτοιμοι για να κατεβάσουμε το περιβάλλον ανάπτυξης της εφαρμογής μας.

Όταν πλέον έχει κατέβει το αρχείο μας, το ανοίγουμε και πατάμε το κουμπί <<εκτέλεση>>.

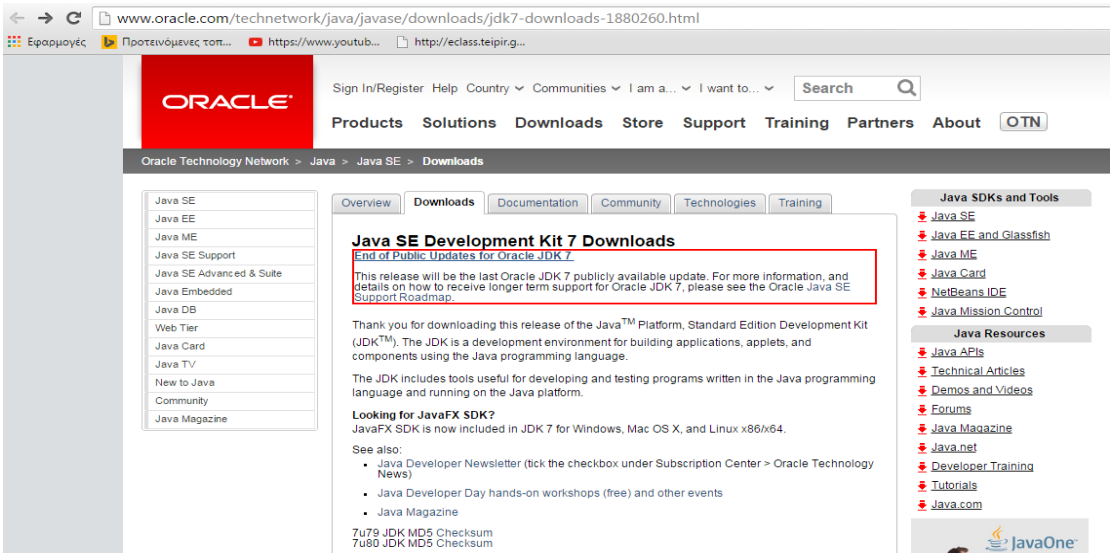
Στην συνέχεια προχωράμε σε ένα από τα πιο σημαντικά βήματα, εφόσον έχουμε πατήσει το κουμπί <<Next>> και μεταφερθεί στην εικόνα 4.2.



Εικόνα 4.2 JDK

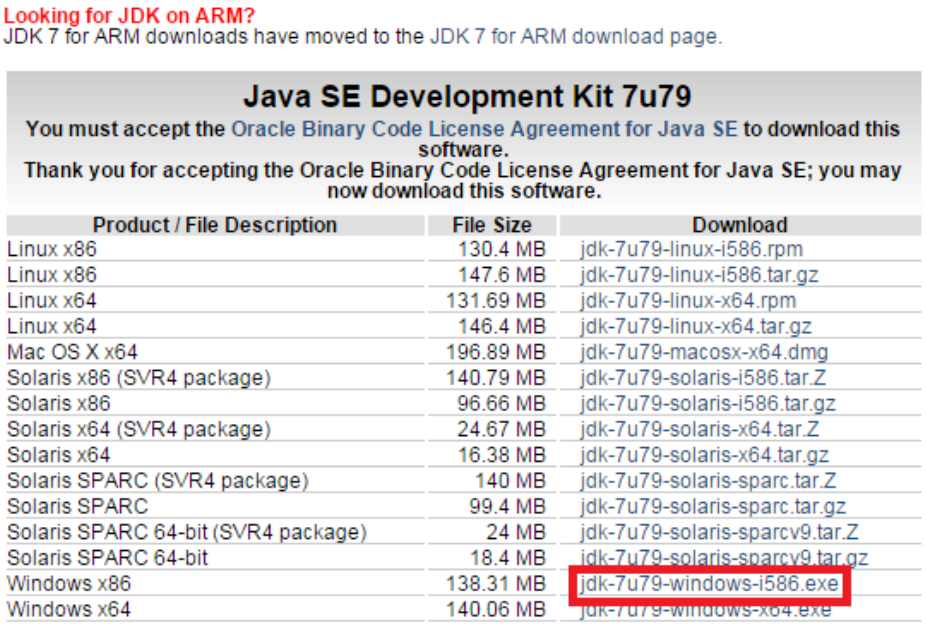
4.3 Download JDK

Σε αυτό το σημείο, θα σταθούμε λίγο παραπάνω διότι, εδώ μας λέει ότι, απαραίτητη προϋπόθεση, για να συνεχιστεί η εγκατάσταση του προγράμματος, είναι η ύπαρξη του εργαλείου Java Development Kit(JDK) v7. Δηλαδή, η ύπαρξη μιας πλατφόρμας java στον υπολογιστή μας, γιατί, όπως προείπαμε, το Android Studio θέλει και την java platform για να λειτουργήσει. Έχουμε, λοιπόν, δύο επιλογές. Αν έχουμε εργαστεί ξανά σε περιβάλλον java, θα έχουμε σίγουρα μια java platform, έτσι, το μόνο που έχουμε να κάνουμε είναι να πατήσουμε <<browse..>> και να ψάξουμε πού έχει αποθηκευθεί στον υπολογιστή μας. Αν δεν έχουμε ξαναδουλέψει σε java, δεν έχουμε και την απαραίτητη πλατφόρμα. Τα βήματα είναι απλά και εύκολα. Αρχικά, πατάμε πάνω στα μπλε γράμματα που λένε **<< jdk-7u67-windows-i586.exe >>** και αυτά αυτόματα μας οδηγούν στη σελίδα που υπάρχει το εργαλείο, το οποίο χρειαζόμαστε για το Android Studio. (βλέπε εικόνα 4.3)



Εικόνα 4.3 Download Kit

Σε αυτήν την σελίδα θα βρούμε το Java SE Development Kit, το οποίο χρειάζεται το περιβάλλον ανάπτυξης της εφαρμογής μας. Πρέπει, το αρχείο που θα κατεβάσουμε να είναι συμβατό με τον επεξεργαστή του υπολογιστή μας (32-bit/64-bit). Έπειτα, κάνουμε κλικ στο Accept License Agreement και κατεβάζουμε το αρχείο που θέλουμε(εικόνα 4.4).



Looking for JDK on ARM?
JDK 7 for ARM downloads have moved to the JDK 7 for ARM download page.

Java SE Development Kit 7u79

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

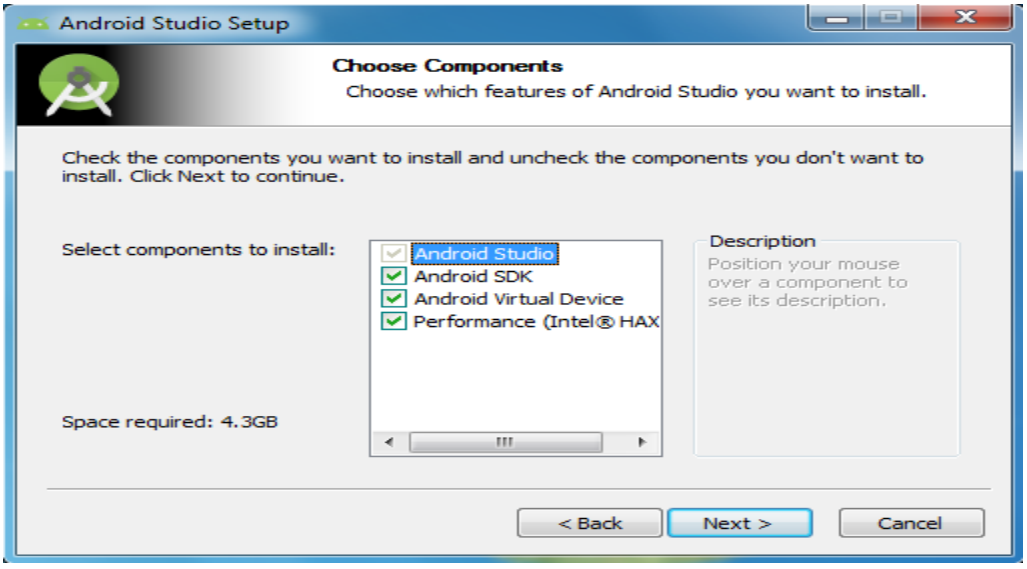
Product / File Description	File Size	Download
Linux x86	130.4 MB	jdk-7u79-linux-i586.rpm
Linux x86	147.6 MB	jdk-7u79-linux-i586.tar.gz
Linux x64	131.69 MB	jdk-7u79-linux-x64.rpm
Linux x64	146.4 MB	jdk-7u79-linux-x64.tar.gz
Mac OS X x64	196.89 MB	jdk-7u79-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.79 MB	jdk-7u79-solaris-i586.tar.Z
Solaris x86	96.66 MB	jdk-7u79-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.67 MB	jdk-7u79-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u79-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	140 MB	jdk-7u79-solaris-sparc.tar.Z
Solaris SPARC	99.4 MB	jdk-7u79-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	24 MB	jdk-7u79-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.4 MB	jdk-7u79-solaris-sparcv9.tar.gz
Windows x86	138.31 MB	jdk-7u79-windows-i586.exe
Windows x64	140.06 MB	jdk-7u79-windows-x64.exe

Εικόνα 4.4 Download this file

Τώρα εγκαθιστούμε το Kit που κατεβάσαμε στον υπολογιστή μας. Πατάμε Next, το πρόγραμμα εγκαθίσταται και πατάμε το κουμπί close.

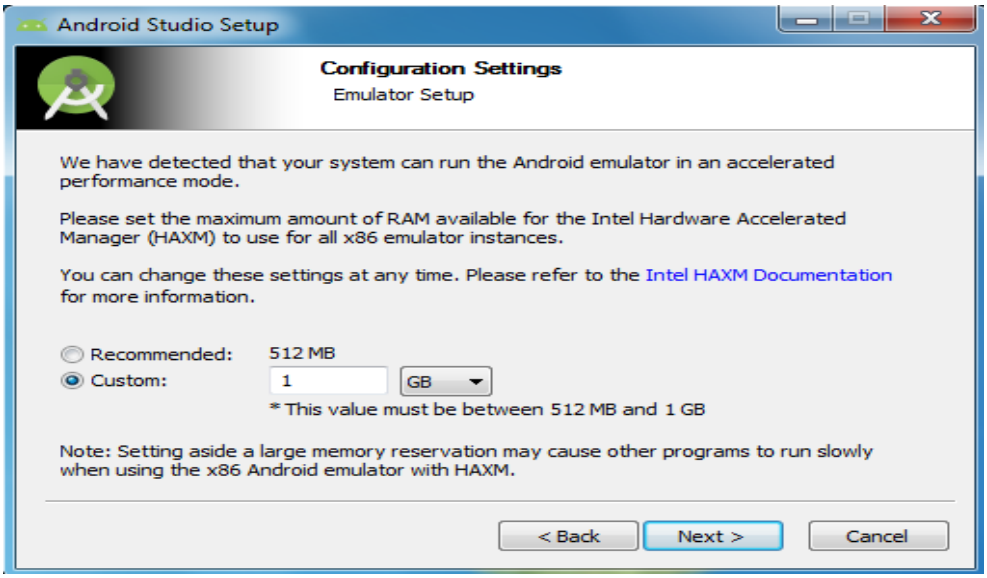
4.4 Εγκατάσταση Android Studio(συνέχεια)

Στην πορεία μεταφερόμαστε στο Android Studio στο εξής παράθυρο. (εικόνα 4.5)



Εικόνα 4.5 Choose Components

Στην συνέχεια, πατάμε agree και μετά επιλέγουμε πού θέλουμε να αποθηκευτεί το Android studio στον δίσκο μας. Στο επόμενο βήμα, επιλέγουμε πόσα MB της μνήμης RAM θα δεσμεύονται όταν ανοίγει το Android Studio. Αυτό που προτείνουμε είναι να επιλέξουμε όσον το δυνατόν μεγαλύτερη τιμή μπορούμε, χωρίς όμως να υπερβούμε το 50% της RAM του Η/Υ μας. Μια καλή ποσότητα μνήμης είναι κοντά στο 1GB με 1.5GB για έναν Η/Υ με 4GB RAM, για να μπορούμε να τρέχουμε μεγάλα προγράμματα, χωρίς αυτό να “κολλάει” και να μας καθυστερεί (εικόνα 4.6).



Εικόνα 4.6 Configuration Settings

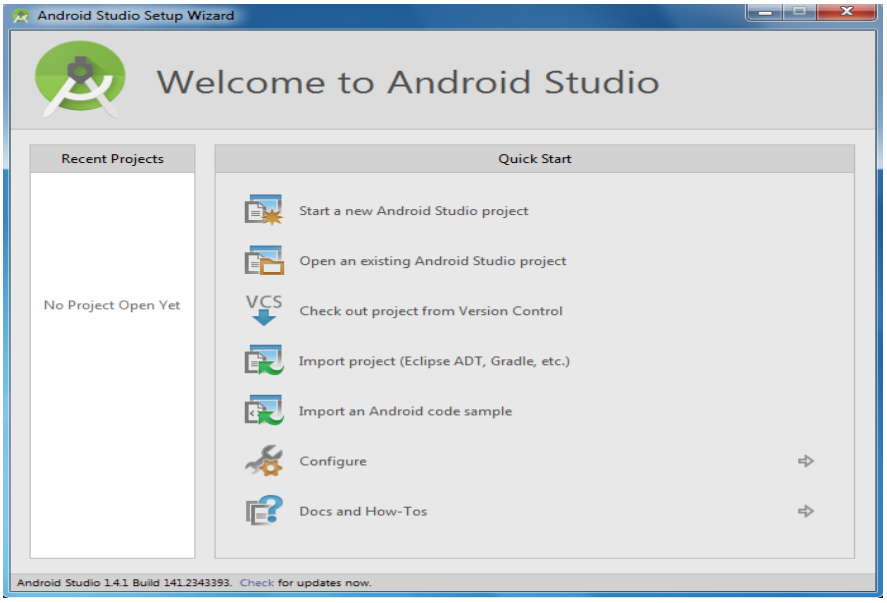
Μετά το Next έρχεται το install του προγράμματος .

Το πρόγραμμα κάνει install με επιτυχία και ρωτάει, αν έχουμε κάποια παλαιότερη έκδοση .

4.5 Δημιουργία ενός απλού προγράμματος Android

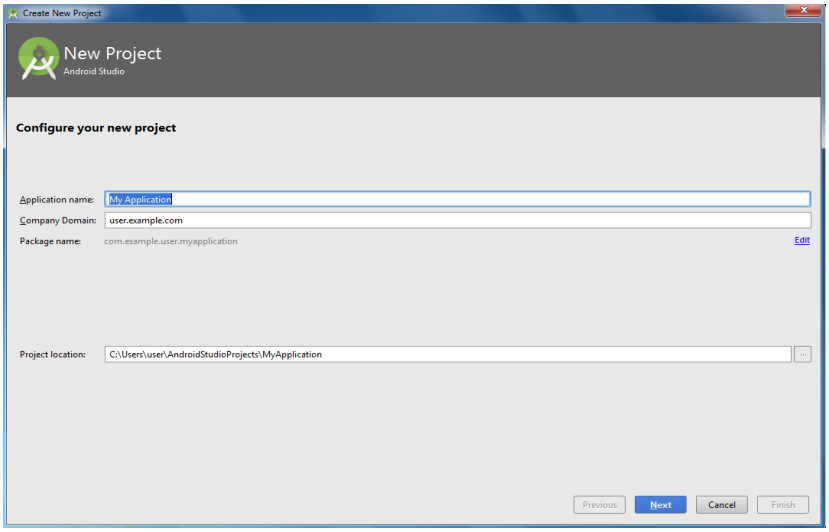
Βήμα 1 -> Δημιουργία ενός νέου έργου

Ανοίγοντας για πρώτη φορά το Android Studio, θα βρεθούμε μπροστά σε ένα παράθυρο. Προετοιμάζεται το Android Studio για άνοιγμα. Έπειτα, ακολουθεί ένα παράθυρο που μας δείχνει ποιες δυνατότητες έχουμε και περιμένει να επιλέξουμε τι ακριβώς επιθυμούμε να κάνουμε. (εικόνα 4.7).



Εικόνα 4.7 Start a new Project

Επιλέγουμε το <<Start a new Studio Android Project>> διότι, ανοίγουμε πρώτη φορά project. Μετά, θα πρέπει να δώσουμε ένα όνομα στην main class που θα έχει το project. Το πρόγραμμα μας βοηθά, για αρχή, δίνοντας ένα προαιρετικό όνομα στην κλάση(εικόνα 4.8).



Εικόνα 4.8 Application name

Βημα 2 -> Επιλογή επιπέδου API

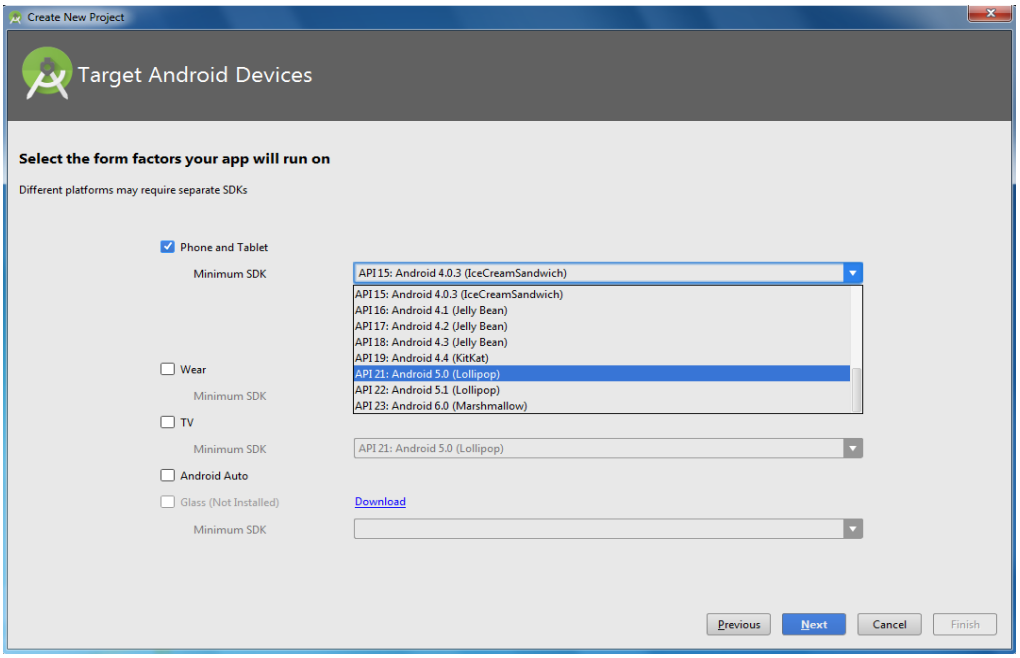
Στο Application name δίνουμε το όνομα που θέλουμε να έχει η κλάση στο project. Με το Company Domain, αποτελούν σημαντικό κομμάτι στις Android εφαρμογές. Σημαντικό είναι, να βάλουμε κάτι μοναδικό, έτσι ώστε αν θέλουμε να δημοσιεύσουμε την εφαρμογή μας στο google play, να είναι μοναδικό αυτό το πακέτο. Στο επόμενο παράθυρο, μας επιτρέπεται η επιλογή των παραγόντων για την μορφή που υποστηρίζεται από την εφαρμογή μας(π.χ. τηλέφωνο, tablet,

τηλεόραση). Για την κάθε μορφή που επιλέγεται έχουμε την δυνατότητα να επιλογής επιπέδου API(εικόνα 4.10) για αυτήν την εφαρμογή (εικόνα 4.11).

Το επίπεδο API δείχνει την κατανομή των κινητών συσκευών, οι οποίες τρέχουν σε κάθε έκδοση android, όπως φαίνεται στην εικόνα 4.9. Κάνοντας κλικ σε ένα επίπεδο API έχουμε τη δυνατότητα να δούμε τα χαρακτηριστικά του συγκεκριμένου επιπέδου που έχουμε επιλέξει. Αυτό μας διευκολύνει στο να ξέρουμε, έστω και στο ελάχιστο, ποιες δυνατότητες και ποια χαρακτηριστικά έχει η εφαρμογή μας.

Code name	Version	API level
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(no code name)	1.1	API level 2
(no code name)	1.0	API level 1

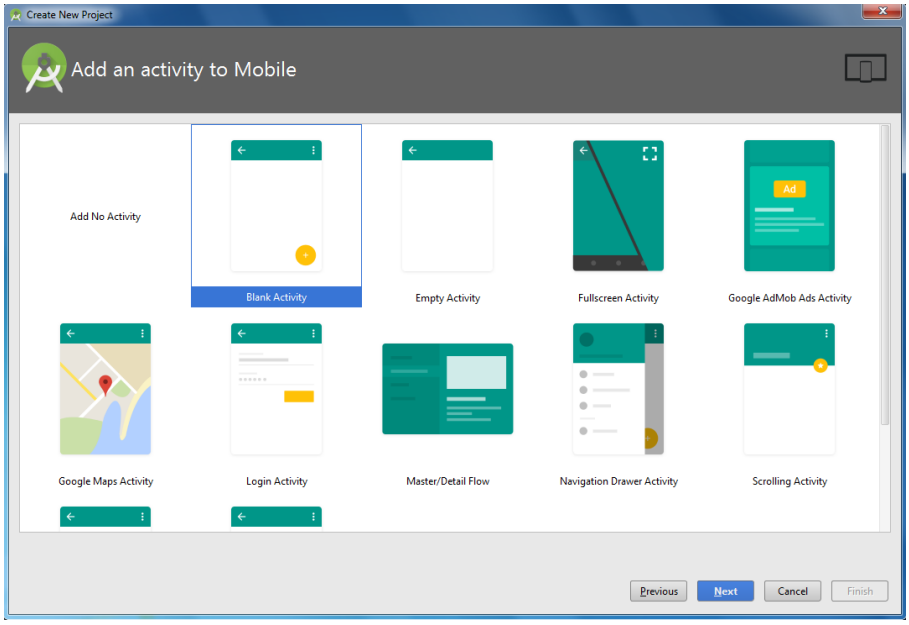
Εικόνα 4.9 Επίπεδα API



Εικόνα 4.10 Target Android Devices

Βήμα 3 -> Προσθέτουμε μια δραστηριότητα(Activity)

Στο επόμενο παράθυρο, μας παρέχεται η δυνατότητα να επιλέξουμε τον τύπο δραστηριότητας που θέλουμε να προσθέσουμε στην εφαρμογή μας (εικόνα 4.11). Αυτή η οθόνη εμφανίζει ένα μεγάλο σύνολο από διαφορετικά activities για την κάθε μια μορφή που επιλέξαμε από το προηγούμενο βήμα.



Εικόνα 4.11 Επιλογή δραστηριότητας

Σημείωση: Σε περίπτωση που έχουμε επιλέξει <<Add No Activity>> μετά κάνουμε κλικ στο κουμπί <<Finish>>. Ενώ, αν έχουμε κάνει κλικ πάνω σε κάποιο άλλο activity, ακολουθούμε τα βήματα που εμφανίζονται ώστε να δημιουργηθεί με επιτυχία.

4.6 Κατέβασμα βιβλιοθηκών και εργασιών

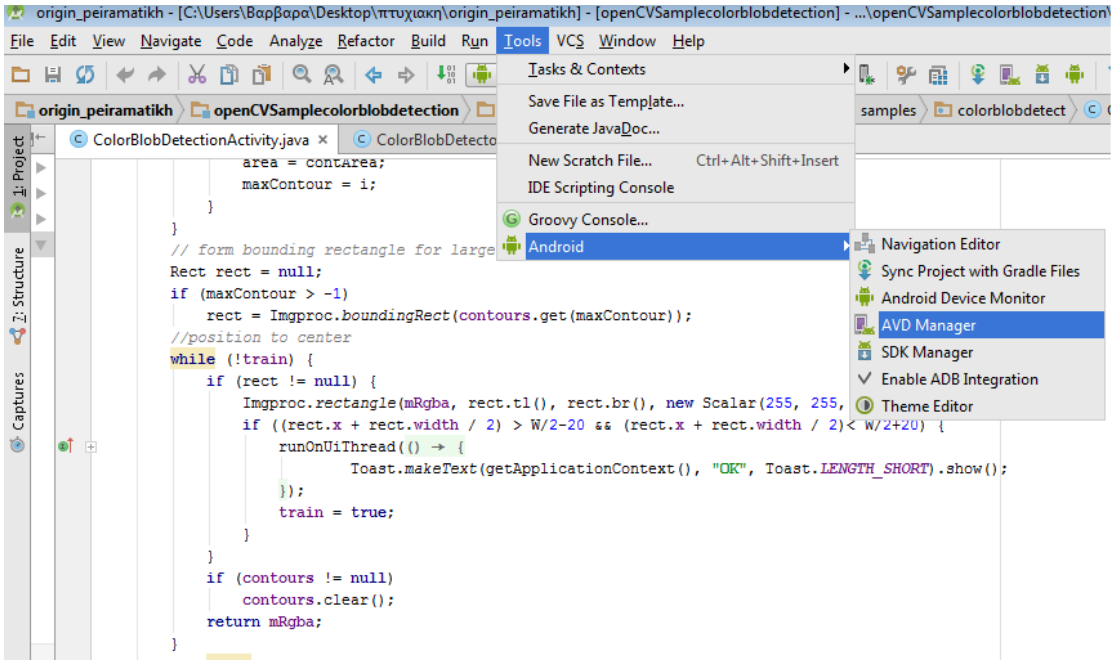
4.6.1 AVD

Μία εικονική συσκευή ή AVD είναι ένας εξομοιωτής (emulator) , ο οποίος δίνει την δυνατότητα να τρέξουμε και να δοκιμάσουμε την εφαρμογή Android σε ένα ευρύ φάσμα συσκευών με μεγάλη ευκολία και το κυριότερο είναι πώς, μπορούμε να το κάνουμε αυτό χωρίς να έχουμε αυτές τις συσκευές. Επίσης, στο AVD έχουμε την δυνατότητα να καθορίσουμε το υλικό και το λογισμικό που θέλουμε να μιμηθεί ο Emulator AVD. Είναι πολύ σημαντικό, να δημιουργήσουμε ένα τέτοιο emulator διότι, θα μας γλιτώσει από πολλές ώρες δουλείας και ένα θετικό που έχει ακόμα, είναι ότι, ξέρουμε πάντα τί έχουμε φτιάξει και πού τρέχει αυτό.

Ο προτιμώμενος τρόπος για να δημιουργηθεί μια τέτοια εικονική συσκευή εξομοιωτή(AVD) είναι η ακόλουθη.

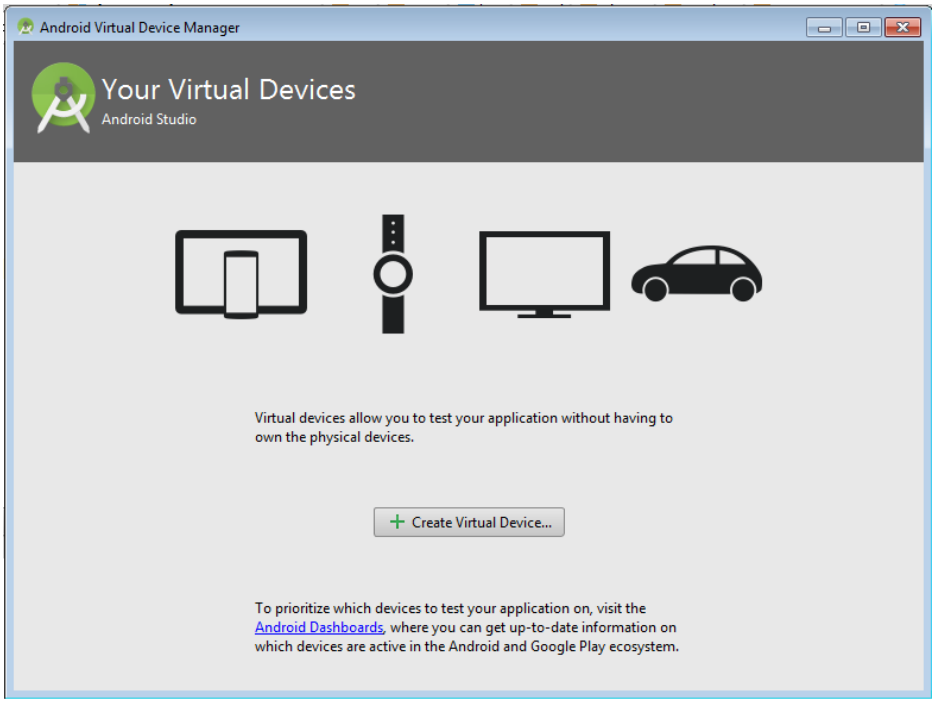
##1 Στο **Android Studio** πάμε στην μπάρα εργαλείων και επιλέγουμε:

Tools->Android->AVD Manager (εικόνα 4.12)



Εικόνα 4.12 Δημιουργία AVD

Τώρα εμφανίζεται στην οθόνη το ακόλουθο παράθυρο (εικόνα 4.13) και επιλέγουμε το <<Create Virtual Device>>.

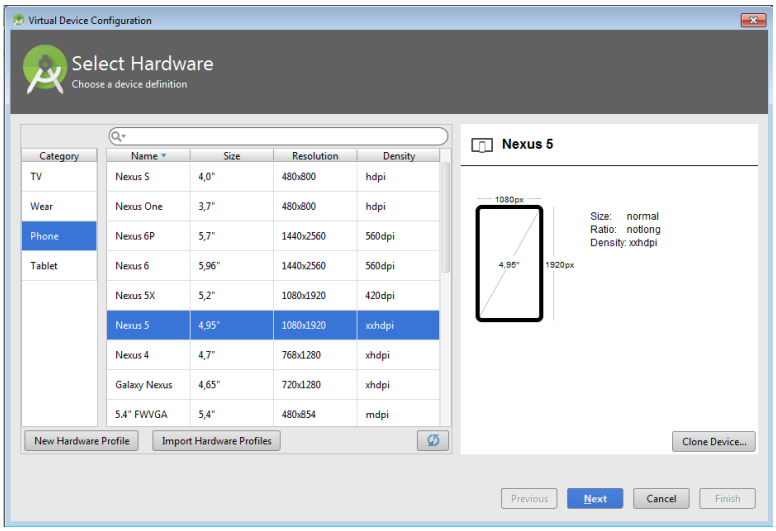


Εικόνα 4.13 Παράθυρο δημιουργίας AVD

##**2** Επιλογή μορφής (πχ. κινητό , tablet) και επιλογή συσκευής

Εδώ υπάρχει η δυνατότητα να διαλέξουμε από την στήλη **<<Category>>** την μορφή συσκευής στην οποία θέλουμε να τρέχει η εφαρμογή μας. Η εργασία μας είναι σε smartphone με Android λειτουργικό οπότε επιλέγουμε **Phone**.

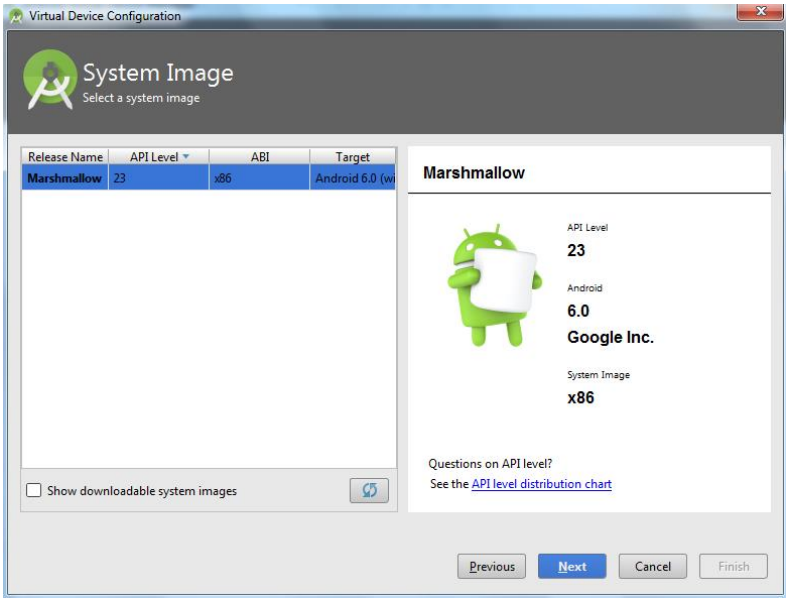
Στην συνέχεια, εμφανίζεται μια γκάμα από συσκευές τις οποίες μπορούμε να επιλέξουμε, έτσι ώστε η εφαρμογή μας να τρέξει εικονικά στην συσκευή της επιλογής μας. Εμείς επιλέξαμε **Nexus 5** και μετά **Next** (εικόνα 4.14).



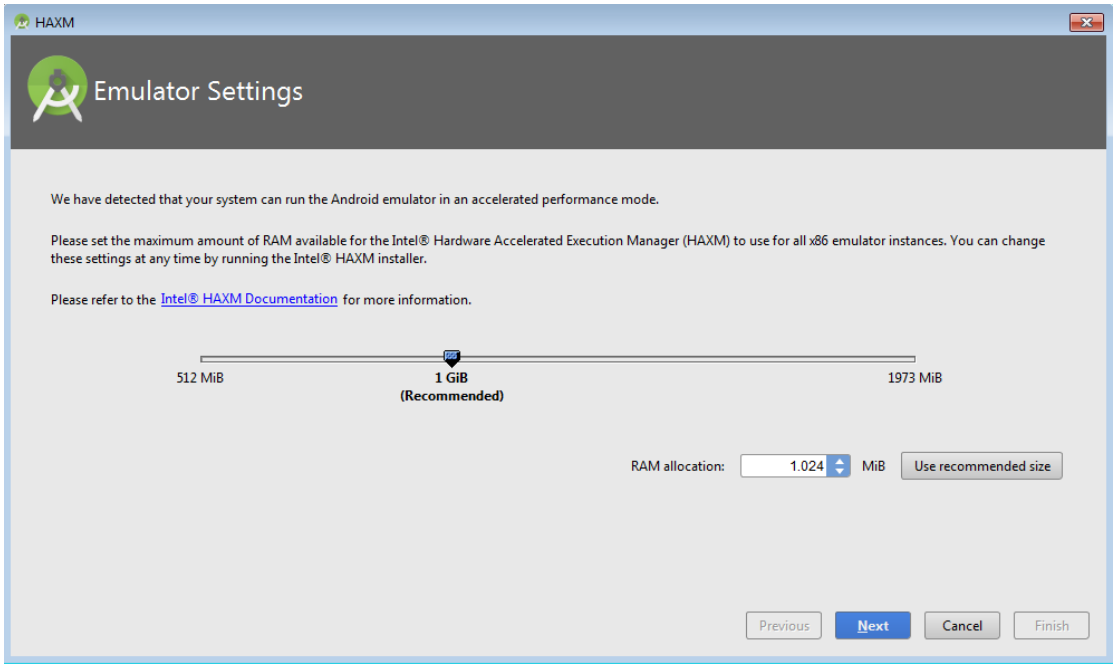
Εικόνα 4.14 Επιλογή εικονικής συσκευής

##**3** System image & emulator settings

Το System image (εικόνα 4.15) μας δίνει το λειτουργικό για το επίπεδο API που έχει η εικονική συσκευή καθώς και την ονομασία αυτού του επιπέδου. Στο Emulator Settings (εικόνα 4.16) επιλέγουμε τις ρυθμίσεις που θα έχει ο emulator σύμφωνα πάντα με τις δυνατότητες του υπολογιστή μας.



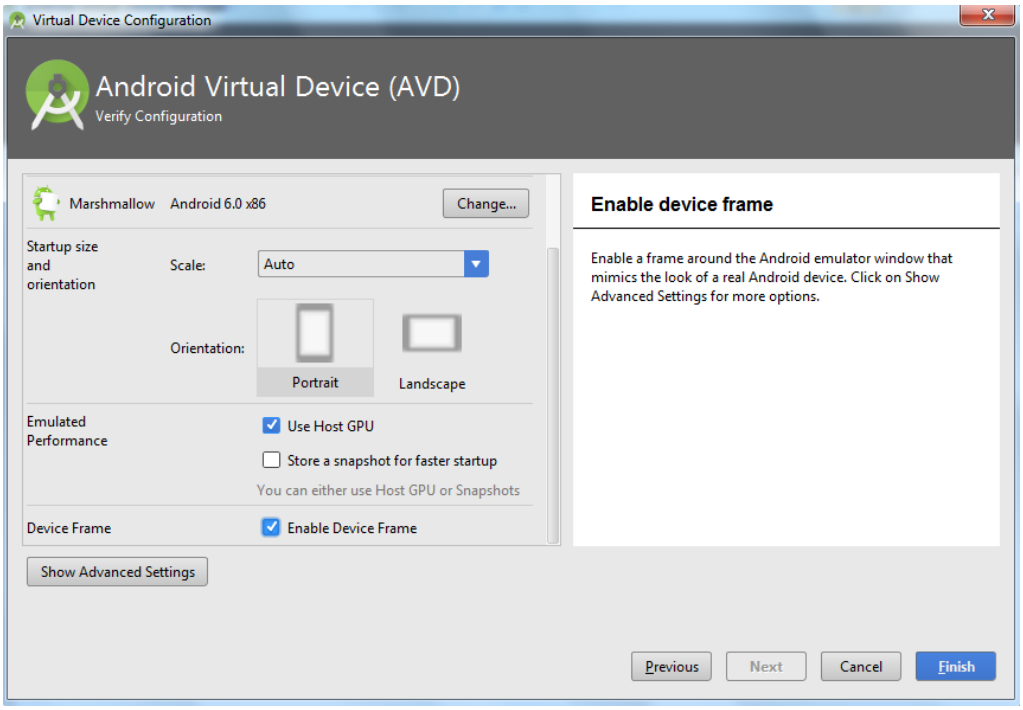
Εικόνα 4.15 System Image



Εικόνα 4.16 Emulator Settings

##**4** Android Virtual Device (AVD)

Αυτό το παράθυρο είναι πολύ σημαντικό διότι εδώ καθορίζονται πολλές ρυθμίσεις του emulator AVD (εικόνα 4.17).



Εικόνα 4.17 Android Virtual Device (AVD)

Ας δούμε αναλυτικά τι ορίζει κάθε πεδίο.

Scale : Μας δίνει την δυνατότητα δοκιμής της εφαρμογής μας σε μια οθόνη η οποία χρησιμοποιεί μια ανάλυση που δεν υποστηρίζεται από τον **AVD Manager**. Εδώ δημιουργούμε μια καινούργια κλίμακα έτσι ώστε, να έχουμε την επιθυμητή ανάλυση.

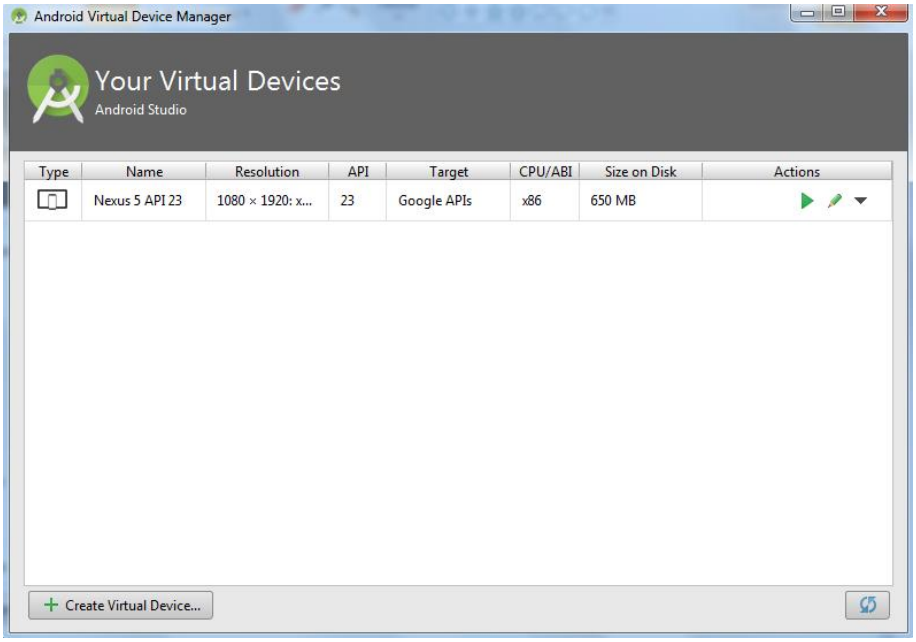
Κλικ στο <<**Use Host GPU**>> : Δίνουμε την δυνατότητα στον emulator που δημιουργούμε, ώστε να τρέχει πιο γρήγορα, να τρέξει με την κάρτα γραφικών του υπολογιστή μας.

Κλικ στο <<**Store a snapshot for faster startup**>> : Συμβάλει στην βελτίωση της επίδοσης του εξομοιωτή. Ξεκινάμε τον **AVD Emulator** από τον **AVD Manager** και κάνουμε κάθε φορά εκκίνηση από το στιγμιότυπο που έχουμε ορίσει. Με αυτόν τον τρόπο όταν ο **AVD** κλείσει, αυτό κρατάει ένα στιγμιότυπο και την επόμενη φορά που θα θέλουμε να τρέξουμε θα ξεκινάει από αυτό το στιγμιότυπο.

Σημείωση: Το κουμπί <<**Use Host GPU**>>, δεν μπορούμε να το κάνουμε επιλεγμένο όταν είναι επιλεγμένο το κουμπί <<**Store a snapshot faster Startup**>> και αντίστροφα. Αυτό γιατί, πρέπει να επιλέξουμε αν θα τρέχει από τα γραφικά του υπολογιστή μας ή από το στιγμιότυπο.

Κλικ στο <<**Enable Device Frame**>> : Ενεργοποιεί ένα πλαίσιο γύρω από το Android emulator window, το οποίο μιμείται την εμφάνιση μιας πραγματικής συσκευής Android.

Αν έχουμε ακολουθήσει τα βήματα σωστά και ο υπολογιστής μας μπορεί να αντεπεξέλθει σε επίπεδο hardware, τότε θα εμφανιστεί το ακόλουθο παράθυρο (εικόνα 4.18).



Εικόνα 4.18 Success Android Virtual Device (AVD)

4.6.2 SDK

SDK σημαίνει "**Software Development Kit**" και είναι ένα σύνολο εργαλείων ανάπτυξης της Google για τους προγραμματιστές που θέλουν να δημιουργήσουν μια εφαρμογή Android.

Το **SDK** είναι το βασικό εργαλείο για:

- Δημιουργία εφαρμογών
- Κατασκευή Custom Rom
- Μεταγλώττιση πυρήνα για την συσκευή μας
- Hacking

Μέσω αυτού του εργαλείου, μπορούμε να χρησιμοποιήσουμε πολλά εργαλεία. Τέτοια είναι το **ADB**, το οποίο είναι υπεύθυνο για την μεταφορά αρχείων σε χώρους που κανονικά δεν επιτρέπεται και το **FastBoot** που μας επιτρέπει να εγκαθιστούμε **custom recovery images** και να ξεκλειδώνουμε τον bootloader της συσκευής μας.

Προσθήκη SDK πακέτων

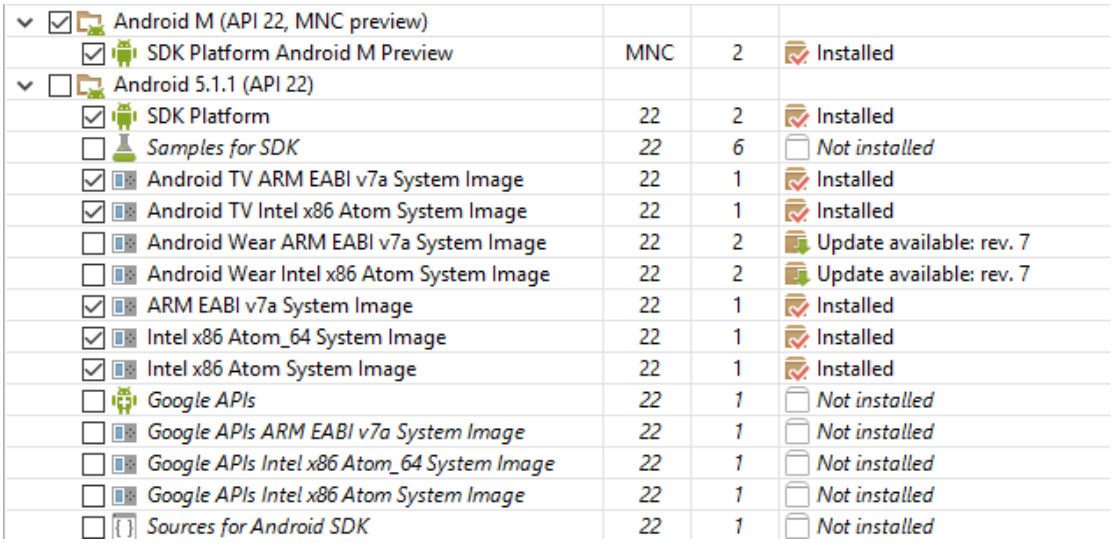
Στο Android studio που δουλέψαμε περιλαμβάνει όλα όσα χρειάζονται ώστε, να ξεκινήσεις την δημιουργία μιας εφαρμογής Android. Το SDK χωρίζει τα εργαλεία, τις πλατφόρμες και άλλα πακέτα που χρειάζεται να κατεβάσουμε με την βοήθεια του **Android SDK Manager**.

Για να προσθέσουμε μερικά απαραίτητα πακέτα(SDK) για το περιβάλλον που θα δουλέψουμε θα πρέπει να κάνουμε τα εξής:

- Ανοίγουμε το Android Studio και κάνουμε κλικ στην γραμμή εργαλείων στο SDK Manager .

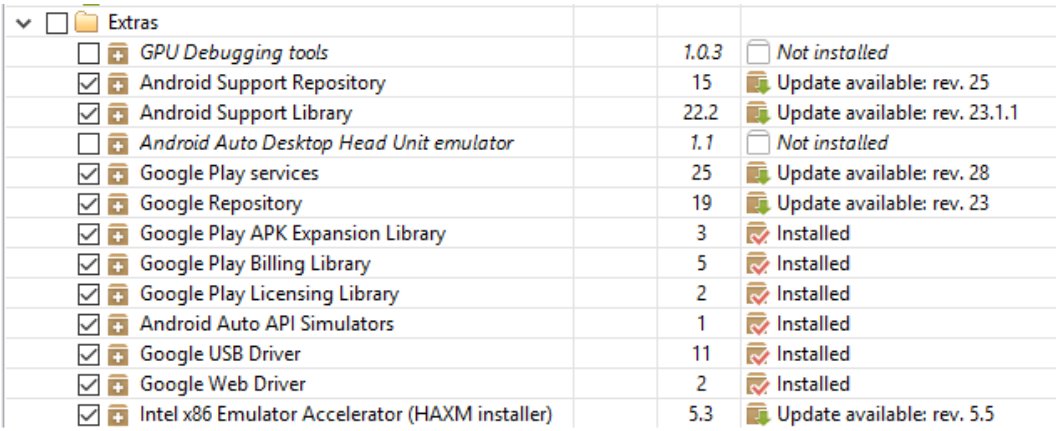
Όταν ανοίγουμε το SDK Manager για πρώτη φορά παρατηρούμε ότι κάποια πακέτα είναι προεπιλεγμένα. Αυτά, είναι τα απαραίτητα εργαλεία που χρειαζόμαστε, ώστε να μπορείς να χρησιμοποιήσεις το εργαλείο του SDK. Επίσης, κατεβάζουμε όποιο άλλο κρίνουμε απαραίτητο.

Στις εικόνες 4.19-4.20 -4.21 παρουσιάζονται τα πακέτα που επιλέξαμε να κατεβάσουμε εμείς για την δικιά μας εργασία.



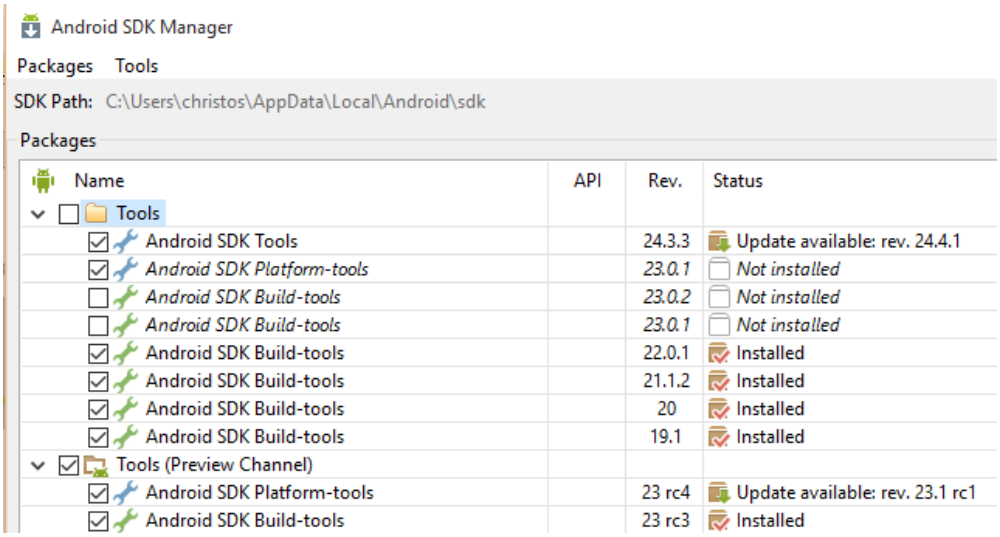
Android M (API 22, MNC preview)			
SDK Platform Android M Preview	MNC	2	Installed
Android 5.1.1 (API 22)			
SDK Platform	22	2	Installed
Samples for SDK	*22*	*6*	*Not installed*
Android TV ARM EABI v7a System Image	22	1	Installed
Android TV Intel x86 Atom System Image	22	1	Installed
Android Wear ARM EABI v7a System Image	22	2	Update available: rev. 7
Android Wear Intel x86 Atom System Image	22	2	Update available: rev. 7
ARM EABI v7a System Image	22	1	Installed
Intel x86 Atom_64 System Image	22	1	Installed
Intel x86 Atom System Image	22	1	Installed
Google APIs	*22*	*1*	*Not installed*
Google APIs ARM EABI v7a System Image	*22*	*1*	*Not installed*
Google APIs Intel x86 Atom_64 System Image	*22*	*1*	*Not installed*
Google APIs Intel x86 Atom System Image	*22*	*1*	*Not installed*
Sources for Android SDK	*22*	*1*	*Not installed*

Εικόνα 4.19 Κατέβασμα πακέτων από SDK Manager part1



Extras			
GPU Debugging tools		*1.0.3*	*Not installed*
Android Support Repository		15	Update available: rev. 25
Android Support Library		22.2	Update available: rev. 23.1.1
Android Auto Desktop Head Unit emulator		*1.1*	*Not installed*
Google Play services		25	Update available: rev. 28
Google Repository		19	Update available: rev. 23
Google Play APK Expansion Library		3	Installed
Google Play Billing Library		5	Installed
Google Play Licensing Library		2	Installed
Android Auto API Simulators		1	Installed
Google USB Driver		11	Installed
Google Web Driver		2	Installed
Intel x86 Emulator Accelerator (HAXM installer)		5.3	Update available: rev. 5.5

Εικόνα 4.20 Κατέβασμα πακέτων από SDK Manager part2



Εικόνα 4.21 Κατέβασμα πακέτων από SDK Manager part3

Στην εικόνα 4.22 κατεβάζουμε την πλατφόρμα του SDK για API 22(Android 5.1.1) καθώς και εργαλεία που χρειάζεται για να συγχρονιστεί η εφαρμογή μας με το Android 5.1.1.

Στην εικόνα 4.23 κατεβάζουμε όλα τα περεταίρω που χρειαζόμαστε για τη εργασία μας. Αυτά είναι, οι υποστηρικτές φακέλων και βιβλιοθηκών. Επίσης, κατεβάζουμε τους usb-drivers, για να μπορεί η εφαρμογή να περαστεί στο κινητό μας, και ό,τι άλλο είναι προεπιλεγμένο. Στην εικόνα 4.24 κατεβάζουμε τα εργαλεία του Android SDK, που απαιτούνται.

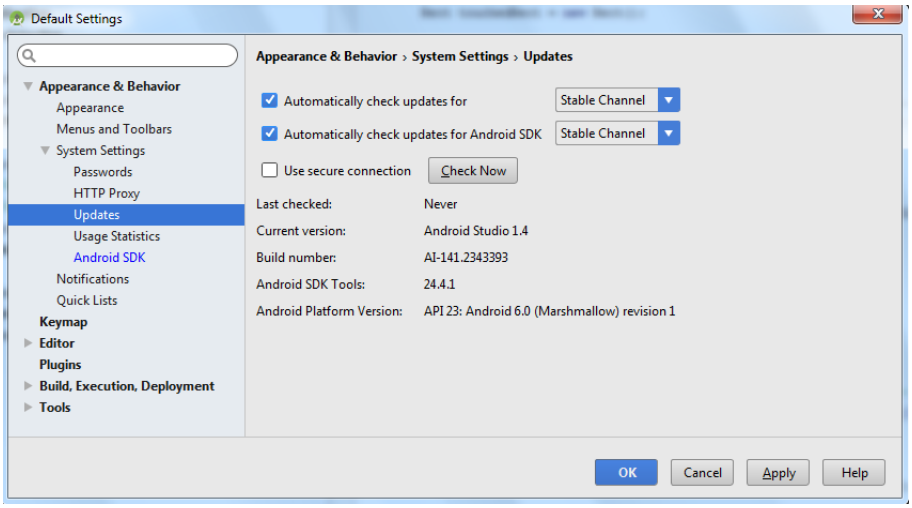
Σημείωση: Τα προεπιλεγμένα πακέτα δεν είναι πάντα οι τελευταίες εκδόσεις. Οπότε καλό θα ήταν, να ελέγξουμε, αν υπάρχει μια πιο πρόσφατη έκδοση των πακέτων που θέλουμε να κατεβάσουμε και να κατεβάσουμε αυτά.

Σημείωση: Παρατηρούμε ότι μερικά πακέτα απ' αυτά που έχουμε κατεβάσει χρειάζονται αναβάθμιση. Καλό είναι να κρατάμε ενημερωμένο το Android Studio και τον SDK Manager, διαφορετικά μπορεί να αντιμετωπίσουμε προβλήματα στις εφαρμογές που αναπτύσσουμε.

Μια άλλη λύση θα ήταν, να ρυθμίσουμε τον SDK Manager να ενημερώνεται αυτόματα, όπως φαίνεται παρακάτω.

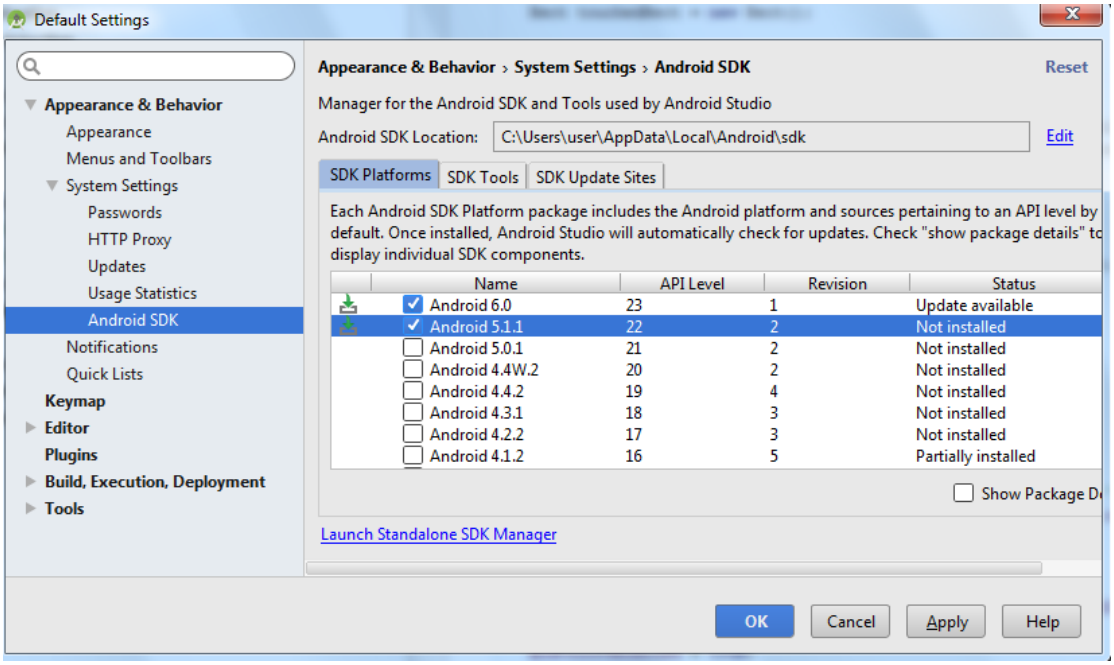
Το Android Studio δεν έχει την αυτόματη ενημέρωση του Android SDK ως προεπιλογή. Για να την ενεργοποιήσουμε θα πρέπει να κάνουμε τα εξής (εικόνα 1.34):

- **File**>**Settings**>**Appearance & Behavior**>**System Settings**>**Updates**
- Επιλέγουμε το <<**Automatically check updates for android SDK**>>
- Κάνουμε κλικ στο **OK** ή **Apply** για να ενεργοποιηθεί η επιλογή



Εικόνα 4.22 Automatically check for updates

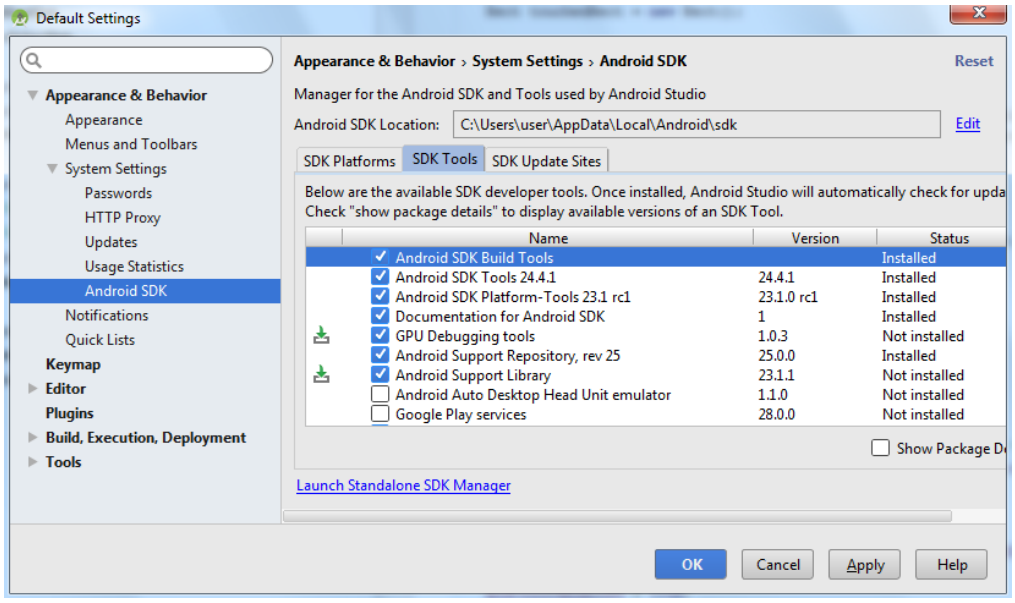
Στην γραμμή εργαλείων, κάνοντας κλικ στο SDK Manager (εικόνα) και στην συνέχεια **Appearance & Behavior**>**System Settings**>**Android SDK** φτάνουμε σε ένα παράθυρο όπως η εικόνα 4.23.



Εικόνα 4.23 SDK Platforms

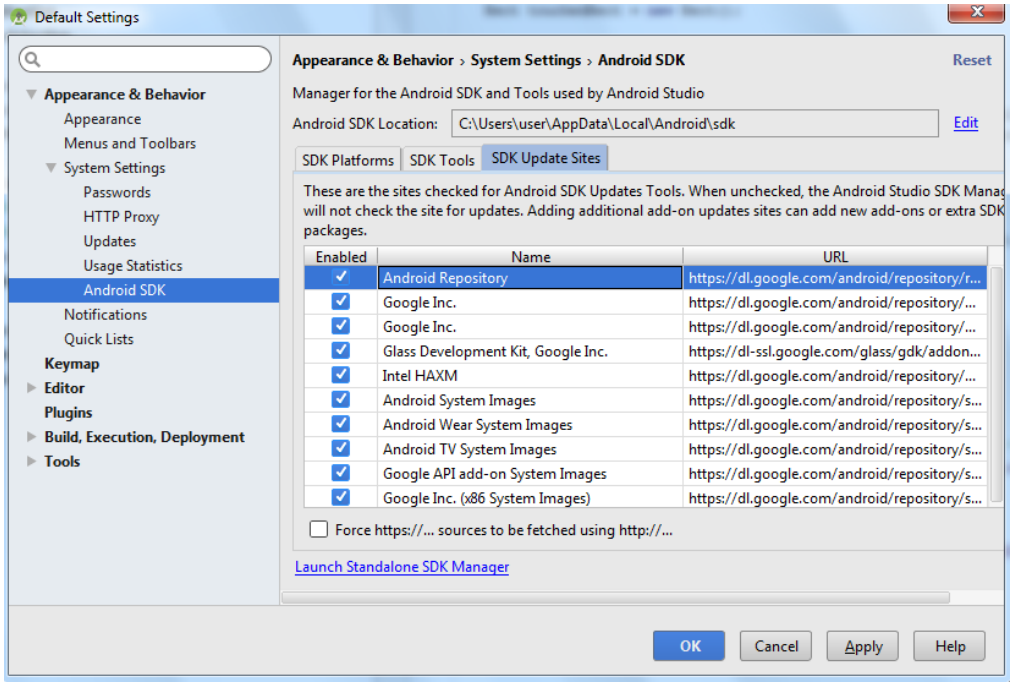
Στο πεδίο SDK Platforms έχουμε την δυνατότητα να παρατηρήσουμε ποιες πλατφόρμες υποστηρίζει το Android Studio στο οποίο δουλεύουμε. Εμείς έχουμε επιλέξει Android 6.0 (API 23) και Android 5.1.1(API 22).

Στο πεδίο SDK Tools παρατηρούμε τα εργαλεία που χρειαζόμαστε, ώστε να έχουμε την δυνατότητα να κάνουμε στην εφαρμογή μας debug, Support library και support repository(εικόνα 4.24).



Εικόνα 4.24 SDK Tools

Στο πεδίο SDK Updates Sites έχουμε την δυνατότητα να επιλέξουμε από ποιες ιστοσελίδες θέλουμε να ψάχνει το SDK για ενημερώσεις (εικόνα 4.25).



Εικόνα 4.25 SDK Update Sites

Εφόσον έχουν γίνει όλα τα παραπάνω βήματα, το Android Studio είναι έτοιμο για χρήση ώστε να χτίσουμε τις δικές μας εφαρμογές.

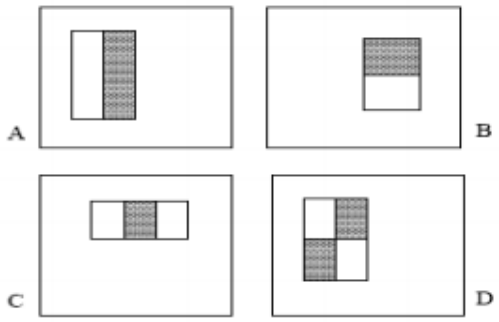
ΚΕΦΑΛΑΙΟ 5 – OPENCV

Η OpenCV είναι μια βιβλιοθήκη ελεύθερου λογισμικού, η οποία αναπτύχθηκε από την Intel και επικεντρώνεται στην επεξεργασία εικόνας. Η OpenCV περιέχει περίπου 300 συναρτήσεις γραμμένες σε C και μερικές κλάσεις γραμμένες σε C++. Η βιβλιοθήκη αυτή δημιουργήθηκε κυρίως για να ανιχνεύει και να αναγνωρίζει πρόσωπα, να κατανοεί και να παρακολουθεί την κίνηση ενός αντικειμένου ,απομόνωση και ανάγνωση αντικειμένων και για να επικοινωνεί ο υπολογιστής με τον άνθρωπο.

Οι αλγόριθμοι της OpenCV είναι βελτιστοποιημένοι για επεξεργαστές αρχιτεκτονικής όπως είναι οι Intel Pentium (MMx ,Pro , 3, 4). Επομένως, η δημιουργία της βιβλιοθήκης αυτής αποσκοπεί και στην δημιουργία μιας κοινότητας ανοιχτού λογισμικού, σχετικά με την μηχανική όραση, η οποία θα αναπτύσσει μεθόδους επεξεργασίας εικόνας όλο και πιο σύγχρονες σε ένα περιβάλλον που συνεχώς εξελίσσεται.

5.1 OpenCV και ανίχνευση αντικειμένων

Στην OpenCV η ανίχνευση των αντικειμένων βασίζεται σε ένα σύστημα μηχανικής μάθησης, το οποίο έχει την δυνατότητα να επεξεργάζεται μια εικόνα σε πολύ μικρό χρονικό διάστημα πετυχαίνοντας πολύ υψηλά ποσοστά ανίχνευσης και ακρίβειας. Όταν χρειάζεται να ανιχνεύσουμε χαρακτηριστικά (features) είναι πολύ πιο εύκολο και πιο αποδοτικό καθώς αυτά κωδικοποιούν πληροφορία σχετικά με το αντικείμενο που πρόκειται να ανιχνευτεί. Τα χαρακτηριστικά αυτά χρησιμοποιούνται σε σύνολα και κωδικοποιούν πληροφορία με βάση την αντίθεση της φωτεινότητας ανάμεσα σε 2 περιοχές της εικόνας, το πού βρίσκεται αυτή η αντίθεση και την χωρητικότητα που έχει το αντικείμενο.



Εικόνα 5.1 Παραδείγματα χαρακτηριστικών

Τα A, B είναι δυ-πολυγωνικά, το C είναι τρι-πολυγωνικό και το D είναι τετρα-πολυγωνικό.

Η βιβλιοθήκη που αναφέραμε κατά την διαδικασία της ανίχνευσης ενός αντικειμένου έχει 3 τρόπους που το πετυχαίνει. Υπάρχουν τρία είδη

χαρακτηριστικών ανίχνευσης, τα δυ-πολυγωνικά, τα τρι-πολυγωνικά και τα τετρα-πολυγωνικά (εικόνα 1.χ). Η τιμή των δυ-πολυγωνικών χαρακτηριστικών είναι η διαφορά ανάμεσα στο σύνολο των pixels που βρίσκονται μέσα σε αυτές τις δύο πολυγωνικές περιοχές. Η τιμή των των τρι-πολυγωνικών χαρακτηριστικών υπολογίζεται από την αφαιρώντας από το άθροισμα των pixels των εξωτερικών πολυγώνων, το άθροισμα των pixels του εσωτερικού πολυγώνου. Τέλος, η τιμή των τέτρα-πολυγωνικών χαρακτηριστικών υπολογίζεται από την διαφορά των pixels ανάμεσα στα διαγώνια μέρη του πολυγώνου.

Για να υπολογίζονται πιο γρήγορα τα χαρακτηριστικά της εικόνας, σε οποιαδήποτε περιοχή ή κλίμακα, η βιβλιοθήκη δεν χρησιμοποιεί κατευθείαν τις τιμές φωτεινότητας της εικόνας αλλά μια ενδιάμεση αναπαράσταση της, γνωστή και ως πλήρης εικόνα. Η πλήρης εικόνα μας δίνει την δυνατότητα να την υπολογίσουμε πολύ εύκολα από την αρχική με χρήση απλών πράξεων στις τιμές των pixels της. Συγκεκριμένα, ο τύπος που μας δίνει την πλήρη εικόνα είναι (εικόνα 5.2).

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

Εικόνα 5.2 Τύπος πλήρης εικόνας

Στο παραπάνω τύπο το ii(x,y) είναι η τιμή της πλήρους εικόνας στο σημείο (x,y) και i(x,y) η τιμή της πραγματικής εικόνας στο σημείο (x,y). Η πλήρης εικόνα μπορεί να υπολογιστεί μόνο με ένα πέρασμα πάνω από την πραγματική εικόνα.

5.2 Διαδικασία εκμάθησης

Η βιβλιοθήκη OpenCV χρησιμοποιεί μια διαδικασία ανίχνευσης, η οποία αρχικά προτάθηκε από τον Paul Viola και υλοποιήθηκε από τον Rainer Lienhart. Αρχικά, έχουμε ένα ταξινομητή, ο οποίος εκπαιδεύεται με έναν μεγάλο αριθμό θετικών δειγμάτων (εικόνες που περιέχουν το αντικείμενο προς ανίχνευση και έχουν όλες ίδιο μέγεθος) και με αρνητικά δείγματα (τυχαίες εικόνες που δεν περιέχουν το αντικείμενο αλλά είναι ίδιου μήκους με αυτές με τα θετικά δείγματα).

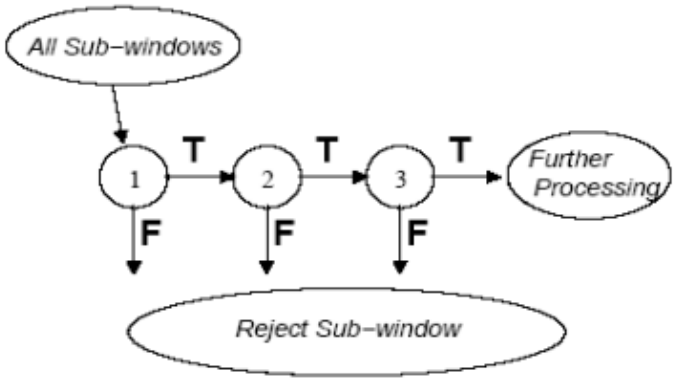
Ας δούμε τώρα, πιο αναλυτικά, τι συμβαίνει όταν έχουμε θετικά δείγματα και τι όταν έχουμε αρνητικά. Από κάθε δείγμα εξάγεται ένας πολύ μεγάλος αριθμός χαρακτηριστικών. Πειραματικά είναι αποδεδειγμένο ότι, ένα πολύ μικρό σύνολο αυτών των χαρακτηριστικών είναι αρκετό για την δημιουργία ενός αρκετά αποδοτικού ταξινομητή. Για να μπορέσουμε να πάρουμε αυτά τα χαρακτηριστικά και να τα επεξεργαστούμε χρησιμοποιείται ένας αλγόριθμος μάθησης, ο οποίος έχει σχεδιαστεί ώστε να επιλέγει μόνο τα πολυγωνικά χαρακτηριστικά. Τα πολυγωνικά χαρακτηριστικά ανταποκρίνονται πολύ καλύτερα στα θετικά και στα αρνητικά δείγματα.

Ο αλγόριθμος μάθησης που χρησιμοποιήθηκε είναι ο AdaBoost σε συνδυασμό με naïve Bayes. Ο AdaBoost είναι ένας αλγόριθμος που προσαρμόζεται με την εικόνα, με την έννοια ότι, οι μεταγενέστεροι ταξινομητές που δημιουργεί ανταποκρίνονται καλύτερα στα θετικά και αρνητικά δείγματα σε σχέση με τους προγενέστερους. Ο αλγόριθμος κάνει επανάληψη για μια σειρά κύκλων. Σε κάθε κύκλο η κατανομή βαρών των χαρακτηριστικών ανανεώνεται έτσι ώστε να ξεχωρίσουν τα χαρακτηριστικά με μεγαλύτερο βάρος από τα άλλα με μικρότερο, κατά την ταξινόμηση.

5.3 Διαδικασία ανίχνευσης

Εφόσον, έχουν δημιουργηθεί οι ταξινομητές, τότε μπορεί να ξεκινήσει η διαδικασία ανίχνευσης αντικειμένων σε μια εικόνα. Η ανίχνευση αντικειμένων γίνεται με την εφαρμογή των ταξινομητών στην εικόνα. Οι ταξινομητές εφαρμόζονται πάνω στην εικόνα και βγάζουν στην έξοδο τους 1 αν η περιοχή όπου ελέγχθηκε περιέχει το αντικείμενο και 0 αν δεν το έχει. Για την ανίχνευση ολόκληρης της εικόνας ο ταξινομητής εξετάζει την εικόνα σειριακά με ένα κυλιόμενο παράθυρο. Επίσης, ο ταξινομητής έχει σχεδιαστεί έτσι ώστε να μπορεί να μεταβάλλει το μέγεθος του παραθύρου του για να μπορεί να ανίχνευση αντικείμενα μεγάλου μήκους. Υπάρχει από πίσω και ένα υπό παράθυρο, το οποίο συνεχώς μεταβάλει το μέγεθος του με συγκεκριμένο ρυθμό. Ο ρυθμός αυτός της μεταβολής του υπό παραθύρου ορίζεται από τον προγραμματιστή ή από τον χρήστη.

Η διαδικασία της ανίχνευσης σε μια εικόνα χωρίζεται σε στάδια. Αρχικά στην εικόνα που θέλουμε να ανιχνεύσουμε εφαρμόζουμε τους πιο απλούς ταξινομητές ώστε να απορριφθεί η πλειονότητα των υπό παραθύρων της εικόνας, τα οποία δεν περιέχουν το αντικείμενο προς ανίχνευση. Στην συνέχεια εφαρμόζονται οι πιο περίπλοκοι ταξινομητές, που έχουν σκοπό την μείωση του ποσοστού λάθους ανίχνευσης. Όταν κάποιο υπό παράθυρο της εικόνας απορριφθεί κατά την διαδικασία της ανίχνευσης δεν επανεξετάζεται από τους πιο περίπλοκους ταξινομητές. Στην εικόνα 1.χ2 φαίνεται η διαδικασία ανίχνευσης αντικειμένου.



Εικόνα 5.3 Διαδικασία ανίχνευσης αντικειμένου

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

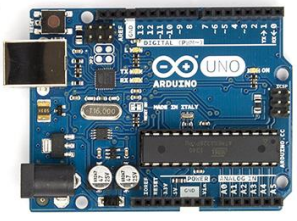
ΚΕΦΑΛΑΙΟ 6 – ARDUINO

6.1 Γενικά για το Arduino

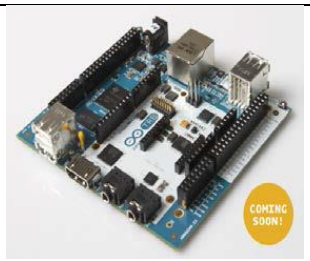
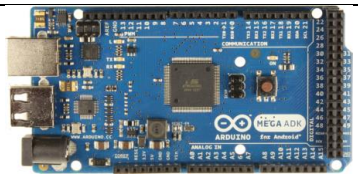
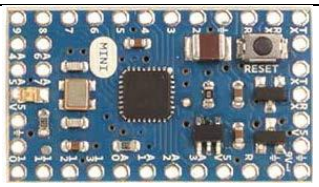
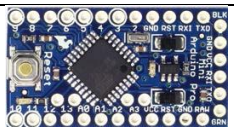
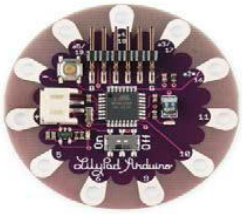
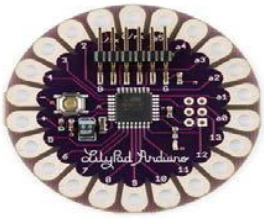
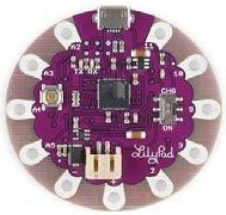
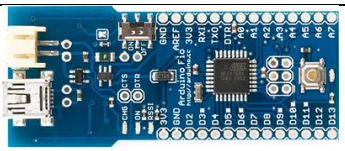
Το Arduino είναι ένας single-board μικροελεγκτής. Πιο αναλυτικά είναι μια πλακέτα ανοιχτού κώδικα που αποτελείται από έναν μικροελεγκτή της ATMEL και εισόδους/εξόδους. Με αυτό μπορεί κάποιος να φτιάξει κάποιο διαδραστικό αντικείμενο και μέσω αυτού μπορεί κάποιος να ελέγξει με τη μορφή κώδικα κάποιες λειτουργίες (π.χ. την κίνηση κινητήρων κ.α.). Προγραμματίζεται εύκολα σε Η/Υ μέσω USB με το πρόγραμμα Arduino IDE που παρέχεται δωρεάν μέσω της επίσημης ιστοσελίδας του (www.arduino.cc), το οποίο είναι συμβατό με όλα τα κύρια λειτουργικά συστήματα. Η γλώσσα προγραμματισμού του είναι μια παραλλαγή της γλώσσας C++ και πιο συγκεκριμένα η Wiring. Γενικότερα, πρόκειται για συσκευή με ευκολία στην χρήση και τον προγραμματισμό και για αυτό ενδείκνυται ακόμα και για αρχάριους. Τέλος, έχουμε πολλές επιλογές παροχής ενεργείας. Πιο συγκεκριμένα, το Arduino μπορεί να τροφοδοτηθεί μέσω της θύρας USB από Η/Υ, μέσω εξωτερικού τροφοδοτικού (7-12V συνήθως) ή ακόμα και από μια μπαταρία 9V.

6.2 Προϊόντα Arduino

Παρακάτω βλέπουμε τα μοντέλα τα όποια κυκλοφορούν στην αγορά, παρόλο που, εάν κάποιος το επιθυμεί, μπορεί να φτιάξει το δικό του Arduino με όλες τις πληροφορίες που χρειάζεται να υπάρχουν στην ιστοσελίδα του.

		
Arduino Uno	**Arduino Leonardo**	**Arduino Mega 2560**

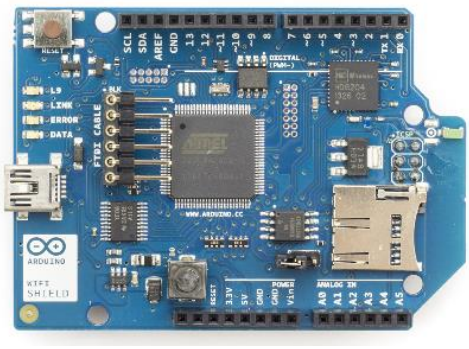
Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

 **Arduino Due**	 **Arduino Yun**	 **Arduino Tre**
 **Arduino Zero**	 **Arduino Micro**	 **Arduino Esplora**
 **Arduino Ethernet**	 **Arduino Mega ADK**	 **Arduino Robot**
 **Arduino Mini**	 **Arduino Nano**	 **Arduino Pro Mini**
 **Lilypad Arduino Simple**	 **Lilypad Arduino**	 **Lilypad Arduino USB**
 **Arduino Fio**	 **Arduino Pro**	

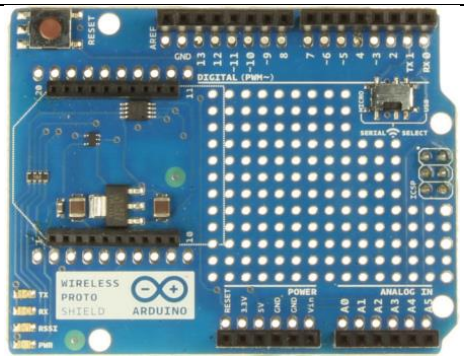
Πίνακας 6.1: Προϊόντα Arduino

6.3 Arduino Shields

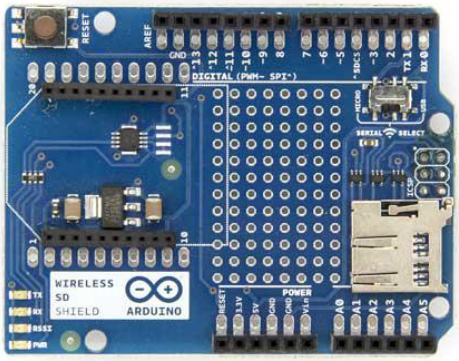
Τα Arduino shields είναι πλακέτες-επεκτάσεις οι οποίες εφαρμόζουν στην επάνω όψη του Arduino και του προσφέρουν διάφορες ιδιότητες και λειτουργίες όπως(Bluetooth,WI-FI,ETHERNET κ.α.). Μπορούν επίσης να συνδυαστούν μεταξύ τους και να τοποθετηθούν πάνω στην πλακέτα. Παρακάτω βλέπουμε κάποια από αυτά.



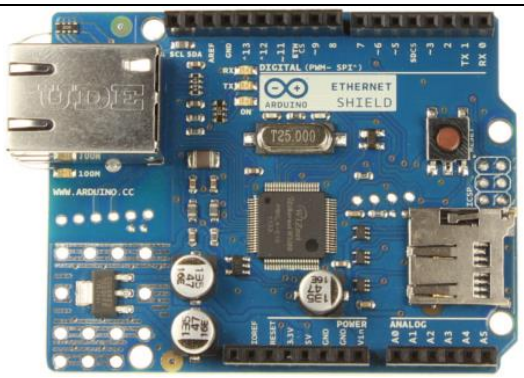
Arduino Wifi Shield



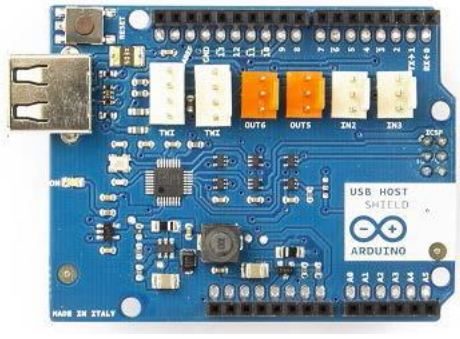
Arduino Wireless Proto Shield



Arduino Wireless SD Shield

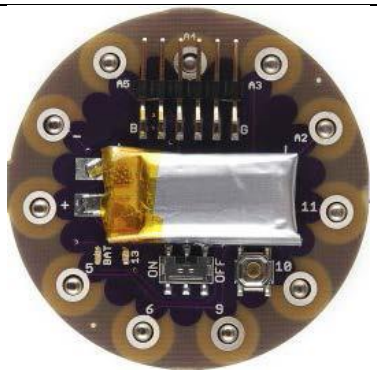
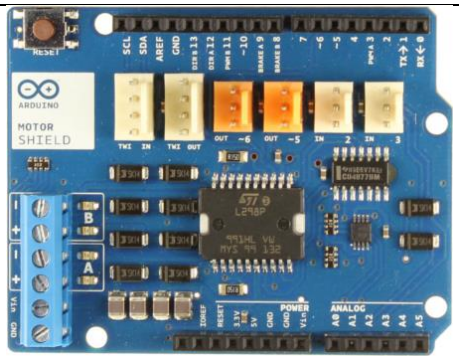


Arduino Ethernet Shield



Arduino USB Host Shield

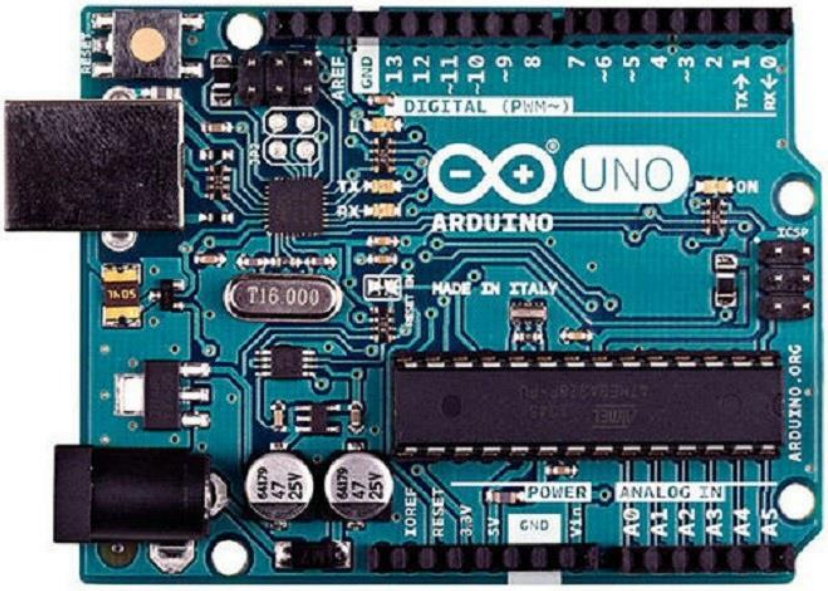
Arduino GSM Shield

	
LilyPad Arduino SimpleSnap	**Arduino Motor Shield**

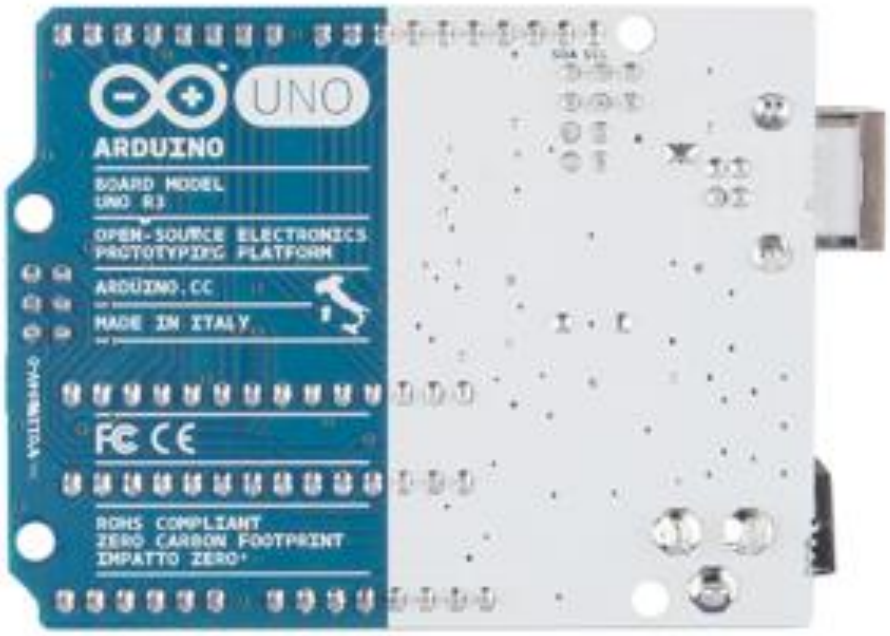
Πίνακας 6.2: Arduino Shields

6.4 Arduino Uno

Στην παρούσα εργασία χρησιμοποιήθηκε το Arduino UNO και παρακάτω παρουσιάζονται συνοπτικά τα χαρακτηριστικά του όπως αυτά παρουσιάζονται στην επίσημη ιστοσελίδα του. Το shield που χρησιμοποιήθηκε είναι κατασκευασμένο από τον καθηγητή μας Δρ. Ι. Έλληνα και δίνει στο arduino μας υποδοχή για το Bluetooth που χρειαζόμαστε καθώς και κλέμες για τα καλώδια που έρχονται από τους κινητήρες.

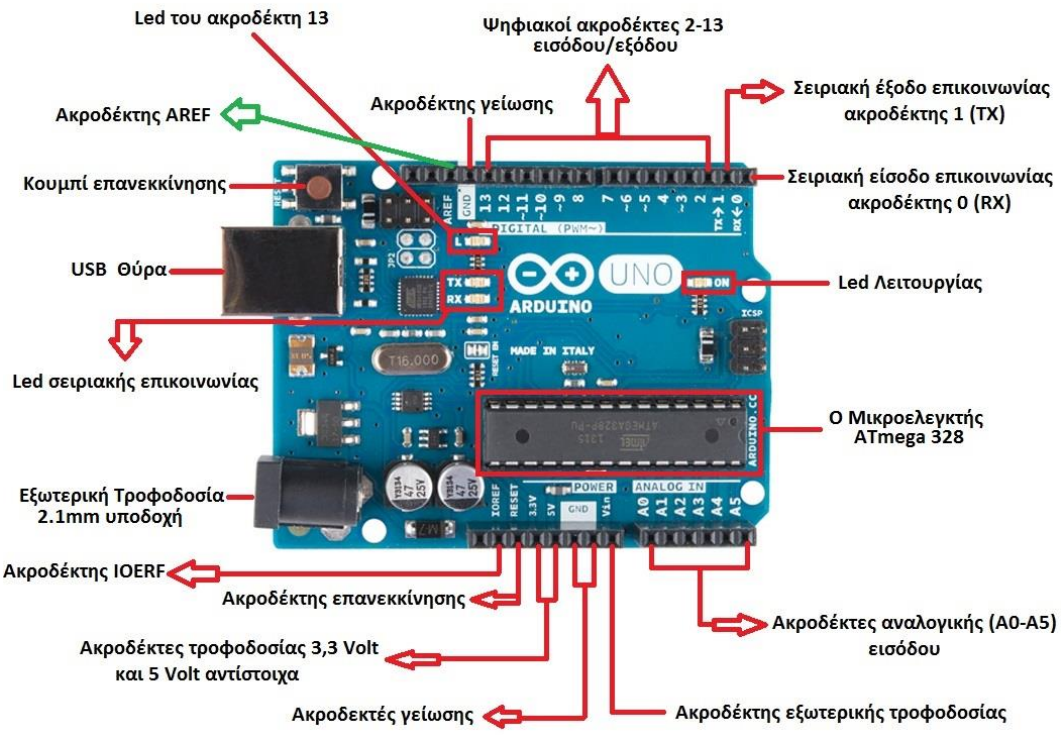


Εικόνα 6.1: Εμπρόσθια όψη Arduino Uno



Εικόνα 6.2: Οπίσθια όψη Arduino Uno

6.5 Χαρακτηριστικά Arduino Uno

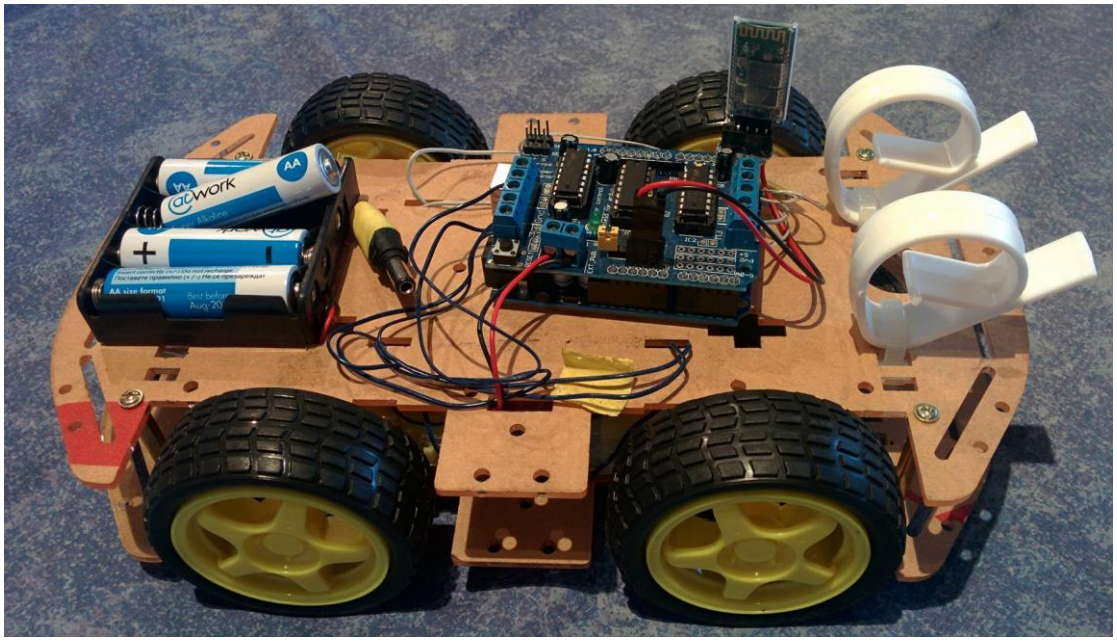


Εικόνα 6.3: Χαρακτηριστικά Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Πίνακας 6.3: Χαρακτηριστικά Arduino Uno

ΚΕΦΑΛΑΙΟ 7 – HARDWARE / ΑΜΑΞΙΔΙΟ ΕΡΓΑΣΙΑΣ

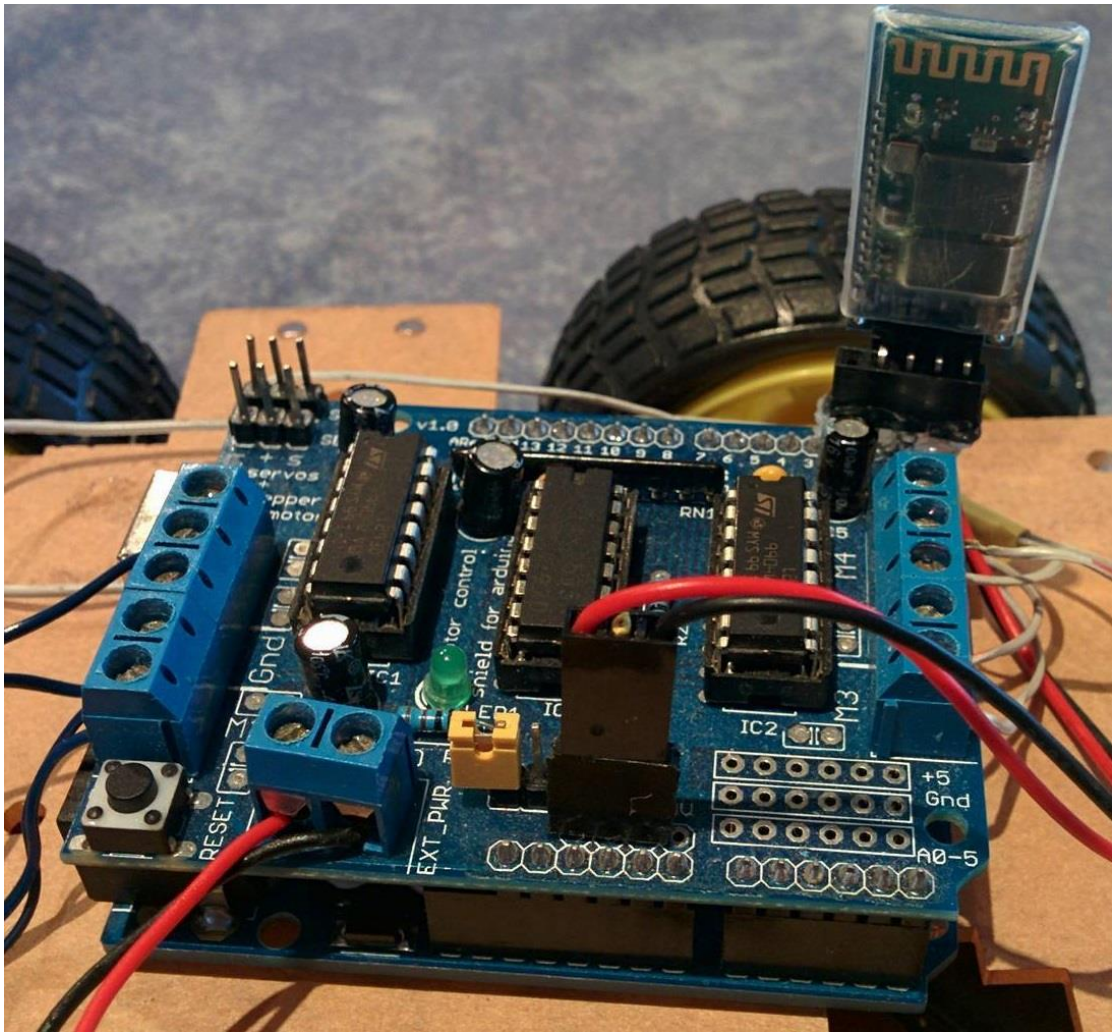


Εικόνα 7.1: Αμαξίδιο εργασίας

Στην παραπάνω εικόνα βλέπουμε το αυτοκινητάκι που είχαμε στη διάθεση μας για την εκπόνηση της εργασία μας. Πρόκειται για ένα αυτοσχέδιο αυτοκινητάκι το οποίο έχει 4 ρόδες με ευθεία μόνο κίνηση. Οι ρόδες του ελέγχονται, η καθεμία ξεχωριστά, από 4 κινητήρες οι όποιοι βρίσκονται στο κάτω μέρος του αμαξιδίου όπως φαίνεται στην επομένη εικόνα.

Εικόνα 7.2: Κινητήρες αμαξιδίου

Στο μπροστινό μέρος, στα λευκά μανταλάκια τοποθετείται το τηλέφωνο το οποίο θα “τρέχει” την Android εφαρμογή στην οποία βασίζεται η εργασία και μέσω της κάμερας του θα καθοδηγεί το αμαξάκι. Το δεύτερο βασικό εξάρτημα της εργασίας μας βρίσκεται πάνω στο αμαξάκι και είναι το Arduino Uno.



Εικόνα 7.3: Arduino, Arduino Shield, Bluetooth

Στη φωτογραφία βλέπουμε το Arduino Uno πάνω στο όποιο έχει τοποθετηθεί και ένα επιπλέον εξάρτημα(Arduino Shield), το οποίο έχει υποδοχές σύνδεσης για τα καλώδια που έρχονται από τους κινητήρες και για το Bluetooth(επάνω δεξιά γωνία της φωτογραφίας), που συνδέει το κινητό με το Arduino. Μέσω του Arduino, λοιπόν, ελέγχονται και οι 4 κινητήρες βάσει της κάμερας του κινητού και της εφαρμογής. Η τροφοδοσία του συστήματος αυτού φαίνεται στην παρακάτω εικόνα.

Εικόνα 7.4: Τροφοδοσία Αμαξιδίου

Πρόκειται για 4 μπαταριές ΑΑ, οι οποίες δίνουν ενεργεία στους κινητήρες και για μια 9v μπαταρία, που τροφοδοτεί το Arduino.

Χρήστος Λ. Κατσάρης
Ηλίας Γ. Γαλανόπουλος

ΚΕΦΑΛΑΙΟ 8 – Η ΕΡΓΑΣΙΑ ΜΑΣ

8.1 Τι περιλαμβάνει;

Η εφαρμογή μας περιλαμβάνει ένα **αυτοκινητάκι με 4 κινητήρες**(έναν για κάθε ρόδα), ένα **Arduino UNO** (το οποίο είναι υπεύθυνο για την σωστή λειτουργία των τροχών, περισσότερη αναφορά έχει γίνει στο κεφάλαιο 7 –Hardware/αμαξίδιο εργασίας), ένα **module Bluetooth**(για την επικοινωνία του arduino με το κινητό) και μια **εφαρμογή Android** χτισμένη πάνω στο Android Studio στην οποία απορροφάμε όλα τα στοιχεία από την **κάμερα του κινητού** και τα επεξεργαζόμαστε έτσι ώστε να βγει το τελικό επιθυμητό αποτέλεσμα.

8.1.1 Lunar3

Η εφαρμογή Lunar3 είναι μια εφαρμογή, η οποία χρησιμοποιεί την κάμερα του κινητού ως κύριο μέσο. Όταν την ανοίγουμε αυτό που εμφανίζεται στην οθόνη της εφαρμογής είναι το activity,το οποίο έχουμε επιλέξει να παρουσιάζεται πρώτο, με ένα ορθογώνιο πλαίσιο στην μέση. Αυτό το ορθογώνιο στην οθόνη υπάρχει με σκοπό να ανιχνεύει τις 2 γραμμές, που έχουμε , και να τις ακολουθεί, μένοντας στο κέντρο των γραμμών. *Στην συ*νέχεια, έχουμε ένα κουμπί, το οποίο ενεργοποιεί τα Bluetooth του κινητού και κάνει αναζήτηση στον χώρο για άλλες συσκευές Bluetooth. Μόλις ανιχνεύσει το module από το αυτοκινητάκι, δίνουμε εντολή να συνδεθεί με αυτό ώστε να μπορούμε να στέλνουμε τις εντολές στο Arduino του αυτοκινήτου. Σε περίπτωση που δεν συνδεθούμε με το Module, το αρχικό activity της εφαρμογής ξαναεμφανίζεται αλλά δεν κάνει τίποτα. Αυτό είναι ένα φίλτρο που τοποθετήσαμε, ώστε να μην κρασάρει η εφαρμογή. Επίσης, υπάρχει και ένα button X, το οποίο σε οποιαδήποτε φάση της εφαρμογής και να βρισκόμαστε, κλείνει την εφαρμογή.

8.1.2 Lunar1

Η εφαρμογή Lunar1 είναι μια εφαρμογή, η οποία χρησιμοποιεί και αυτή την κάμερα του κινητού ως κύριο μέσο. Η λειτουργία αυτής της εφαρμογής είναι, να ακολουθεί ένα αντικείμενο οποιουδήποτε χρώματος μέσα στον χώρο. Όταν την ανοίγουμε αυτό που εμφανίζεται στην οθόνη της εφαρμογής είναι το activity που έχουμε επιλέξει να βγαίνει πρώτο στην εφαρμογή μας. Στην συνέχεια, έχουμε ένα κουμπί, το οποίο ενεργοποιεί τα Bluetooth του κινητού και κάνει αναζήτηση στον χώρο για άλλες συσκευές Bluetooth. Μόλις ανιχνεύσει το module από το αυτοκινητάκι, το συνδέουμε με αυτό για να μπορούμε να στέλνουμε τις εντολές στο Arduino του αυτοκινήτου. Σε περίπτωση που, δεν συνδεθούμε με το Module,το αρχικό activity της ξαναεμφανίζεται αλλά δεν κάνει τίποτα. Αυτό είναι ένα φίλτρο που βάλαμε εμείς έτσι ώστε να μην κρασάρει η εφαρμογή. Επίσης, υπάρχει και ένα button X το οποίο σε οποιαδήποτε φάση της εφαρμογής και να είμαστε, κλείνει την εφαρμογή.

8.1.3 Επιλογή της εργασίας

Επιλέξαμε ως τελική εφαρμογή για την παρουσίαση της πτυχιακής μας εργασίας την εφαρμογή Lunar1. Αυτό δεν σημαίνει ότι δεν θα εξηγήσουμε και την Lunar3 αλλά θα επεκταθούμε περισσότερο στην Lunar1.

Τι μας οδήγησε σε αυτήν την απόφαση;

Η απόφαση δεν ήταν κάτι εύκολο. Έπρεπε να κοιτάξουμε καλά και τις δύο εφαρμογές και να ζυγίσουμε τα θετικά και τα αρνητικά της καθεμιάς. Ένα μεγάλο αρνητικό της Lunar3 είναι ότι αν και ο κώδικας λειτουργεί σωστά, σε ένα αμαξάκι με μεγαλύτερη δύναμη στους κινητήρες και με ρόδες που να στρίβουν, θα λειτουργούσε πολύ καλά. Με αυτό το αυτοκινητάκι λοιπόν, αποφασίσαμε να φτιάξουμε μια εφαρμογή στα μέτρα του. Στην εφαρμογή Lunar1, εμπλουτίσαμε τον κώδικα έτσι ώστε να ακολουθάει ένα αντικείμενο οποιουδήποτε χρώματος στον χώρο. Το ακολουθάει όμως, απλά πηγαίνοντας ευθεία, όσο το αντικείμενο αυτό βρίσκεται στον χώρο που ανιχνεύει η κάμερα. Μόλις αυτό χαθεί τότε και το αυτοκινητάκι σταματάει και αν πατήσεις ένα άλλο αντικείμενο στον χώρο ακολουθεί αυτό.

8.2 Δημιουργία της Εφαρμογής Lunar1

Και για τις 2 εφαρμογές που έχουμε φτιάξει η διαδικασία δημιουργίας είναι ακριβώς η ίδια. Οπότε θα σταθούμε στην υλοποίηση της Lunar1 όπου είναι και η βασική μας εφαρμογή.

Για την δημιουργία της εφαρμογής θα χρειαστούμε κάποια απαραίτητα εργαλεία (κυρίως βιβλιοθήκες) που θα μας βοηθήσουν με την επεξεργασία των στοιχείων της κάμερας του κινητού. Ένα τέτοιο εργαλείο είναι το OpenCV για Android, το οποίο είναι ένα εργαλείο, που περιέχει παραδείγματα όσον αφορά στον τρόπο επεξεργασίας της κάμερα ενός κινητού, το οποίο βασίζεται σε Android λειτουργικό.

8.2.1 Download OpenCV 3.0.0

Αρχικά, κάνουμε αναζήτηση στο internet για το OpenCV (αν χρειαζόμαστε περισσότερες λεπτομέρειες για την OpenCV βιβλιοθήκη μπορούμε να τις βρούμε στο κεφάλαιο OpenCV). Εδώ, θα κατεβάσουμε την OpenCV3.0.0, θα την κάνουμε “import” στο Android Studio και θα δείξουμε τον τρόπο με τον οποίο χτίσαμε την εφαρμογή μας βήμα-βήμα.

Κατεβάζουμε την **OpenCV300** από εδώ: http://opencv.org/downloads.html

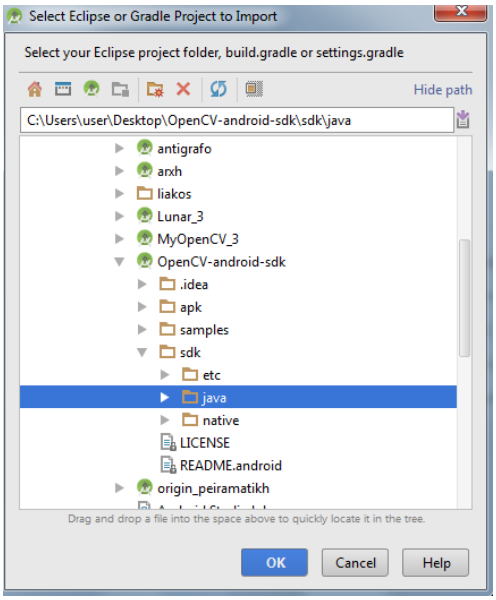
Στην συνέχεια αποθηκεύουμε ως συμπιεσμένο αρχείο (.rar) εκεί που του έχουμε ορίσει, με το όνομα **OpenCV-3.0.0-android-sdk-1.**

Κάνουμε αποσυμπίεση σε μια τοποθεσία που προτιμούμε (π.χ. στα έγγραφα μου).

8.2.2 Δημιουργία Module από την OpenCV

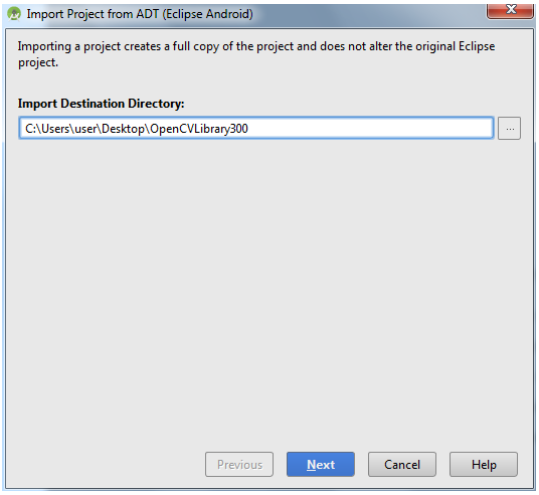
Στην συνέχεια θα πρέπει να δημιουργήσουμε ένα **module** για να μπορέσουμε να το ενσωματώσουμε στην εφαρμογή μας. Τα βήματα για το Android Studio είναι τα εξής:

- ✓ Ανοίγουμε το Android Studio
- ✓ File→New→Import Project και πάμε εκεί που αποθηκεύσαμε τον φάκελο που κάναμε unzip και επιλέγουμε τον φάκελο sdk/java και στην συνέχεια OK (εικόνα 8.1)



Εικόνα 8.1 Import project

- ✓ Δίνουμε ένα όνομα στο project που δημιουργήσαμε για να ξέρουμε τι έχουμε κάνει μέχρι στιγμής. Δώσαμε το όνομα OpenCVLibrary300 και μετά Finish.(εικόνα 8.2)



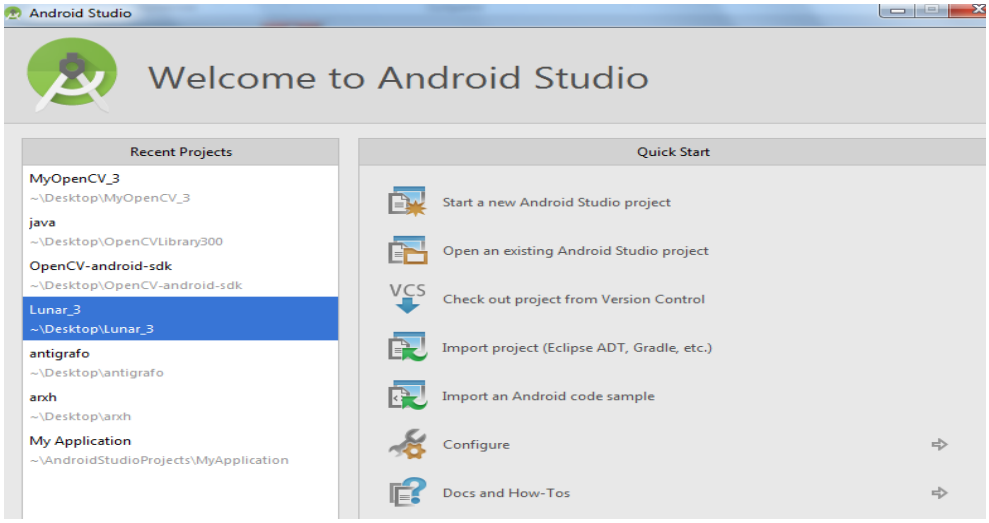
Εικόνα 8.2 Αλλαγή ονόματος

- ✓ File→Close Project

8.2.3 Δημιουργία project MyOpenCV_3

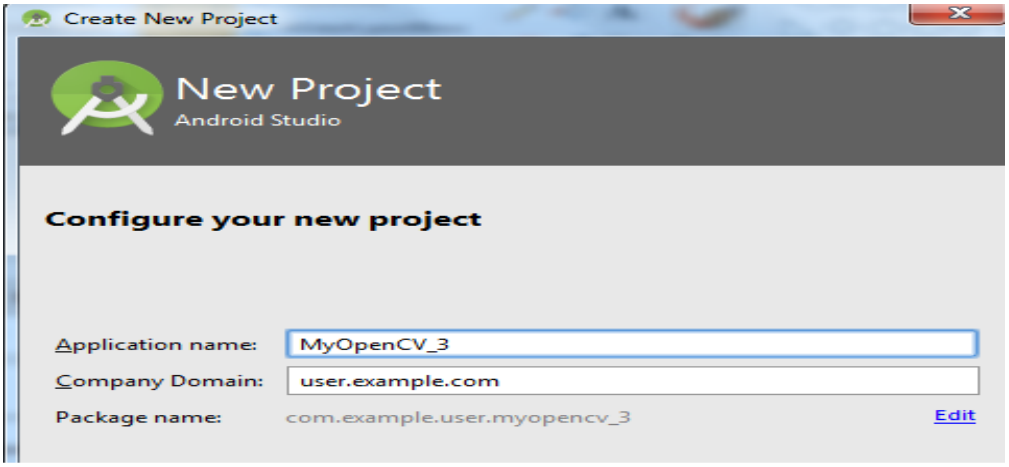
Στην συνέχεια δημιουργούμε ένα project για την εφαρμογή μας με στόχο την επεξεργασία των εικόνων που δίνει η κάμερα του κινητού. Τα βήματα είναι τα ακόλουθα:

- ✓ Επιλέγουμε το <<Start a new Android Studio project>> (εικόνα 8.3)



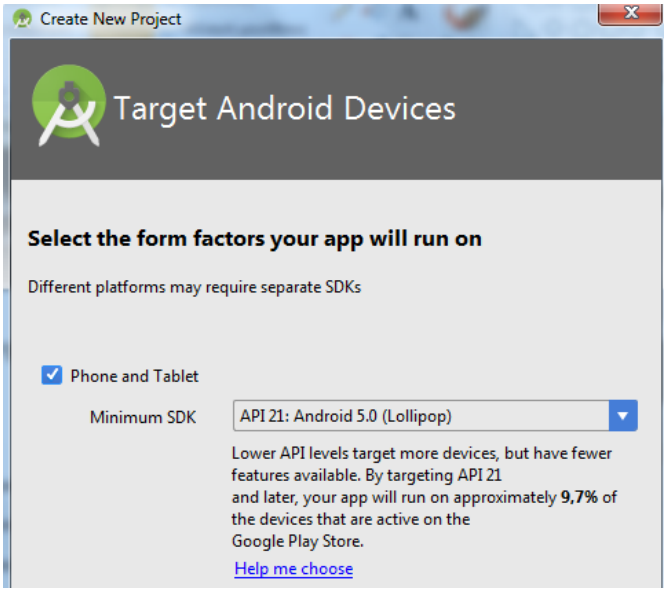
Εικόνα 8.3 Start a new Android Studio Project

- ✓ Του δίνουμε το όνομα MyOpenCV_3 και πατάμε Next. (εικόνα 8.4)



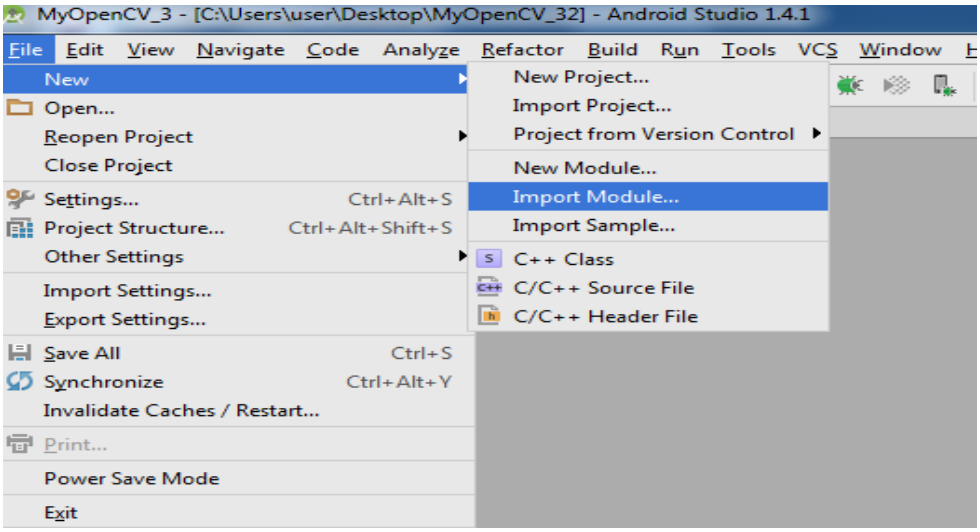
Εικόνα 8.4 Application Name

- ✓ Στην συνέχεια επιλέγουμε <<Phone and Tablet>> για Android 5.0 (Lollipop). Αυτό είναι το χαμηλότερο επίπεδο API που τρέχει η εφαρμογή μας. (εικόνα 8.5)

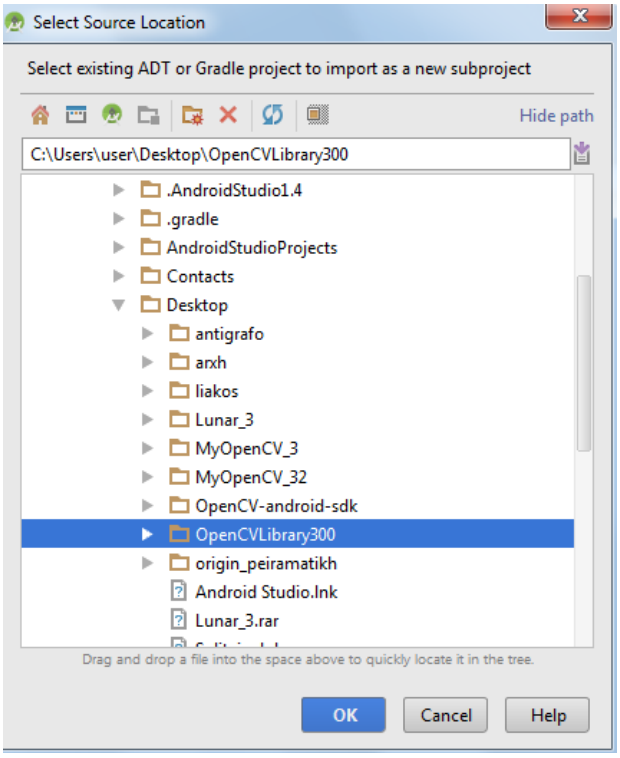


Εικόνα 8.5 Target Android Devices

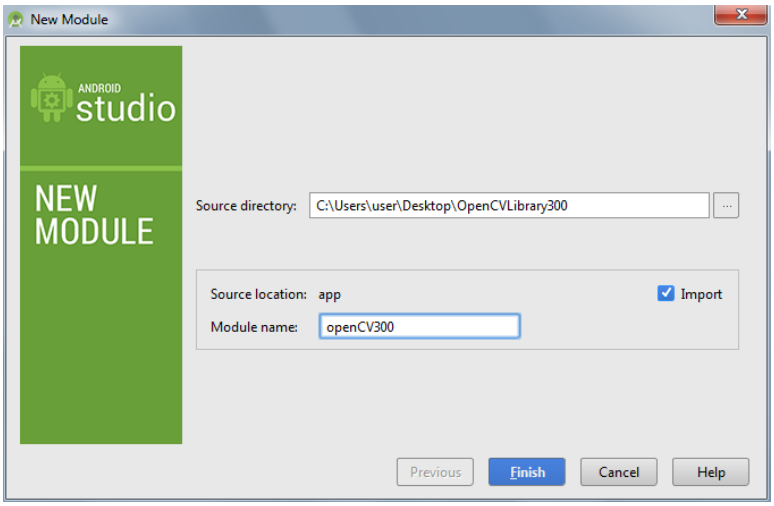
- ✓ Add No Activity για να έχουμε ένα καθαρό background.
- ✓ Στην συνέχεια βλέπουμε πως έχουμε ανοίξει την εφαρμογή. Κάνουμε File→New→Import Module. Με το πλήκτρο browse μετακινούμαστε μέσα στον υπολογιστή μας και ψάχνουμε να βρούμε το module που έχουμε δημιουργήσει. Εμείς το αποθηκεύσαμε σαν OpenCVLibrary300 και πατάμε OK. Κάνουμε κλικ στο κουτάκι Import και επιλέγουμε ένα όνομα όπως openCV300.(εικόνα8.6 & εικόνα 8.7 & εικόνα 8.8)



Εικόνα 8.6 Import Module

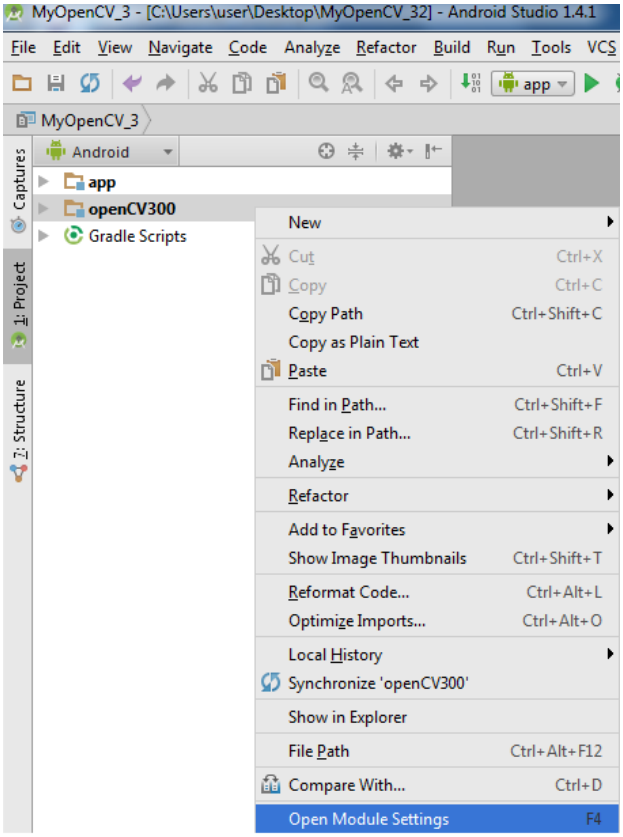


Εικόνα 8.7 Import project OpenCVLibrary300

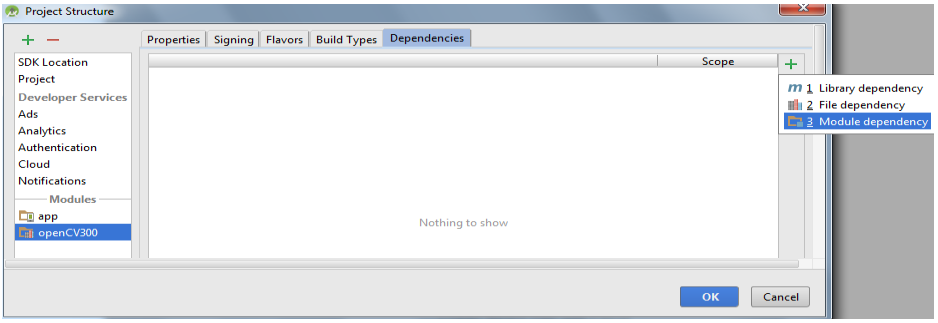


Εικόνα 8.8 Rename σε openCV300

- ✓ Πατάμε στο project→δεξί κλικ→Open Module settings (εικόνα 8.9). Στην συνέχεια επιλέγουμε από το Modules→openCV300 και από τα πεδία το Dependencies. Πατάμε + και επιλέγουμε Module Dependency και μετά Οκ (εικόνα 8.10).



Εικόνα 8.9 Open Module Settings



Εικόνα 8.10 Add Module Dependency

Τώρα έχουμε φτάσει στο σημείο όπου έχει γίνει αυτόματα Rebuild και η OpenCV300 είναι πλέον ενσωματωμένη στο project μας.

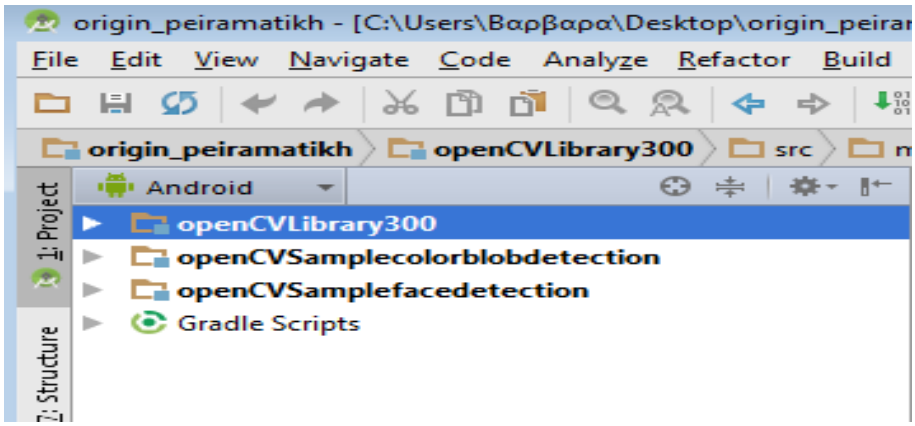
8.2.4 Δημιουργία κλάσεων στο MyOpenCV_3

Η βιβλιοθήκη OpenCV300 έχει νέους τρόπους διαχείρισης της κάμερας του κινητού, προφανώς για τα νέα επίπεδα API. Έτσι, αποφασίσαμε να πάρουμε τις κλάσεις που υπάρχουν σαν samples στην βιβλιοθήκη και να περάσουμε όσες χρειαζόμαστε στην δικιά μας εφαρμογή.

- ✓ Κάνουμε import από τα samples της OpenCV το color-blob-detection. Κάνουμε copy paste τα περιεχόμενα του αρχείου manifest.xml, του activity_main και του αρχείου java class (ColorBlobDetectionActivity.

- ✓ Κάνουμε import από τα samples της OpenCV το face_detection. Κάνουμε copy paste τα περιεχόμενα του αρχείου manifest.xml, του activity_main και του αρχείου java class (FdActivity) .
- ✓ Από τους κώδικες που έχουμε περάσει σβήνουμε ό,τι δεν χρειαζόμαστε και φτιάχνουμε την εφαρμογή μας με τέτοιον τρόπο,ώστε να λαμβάνει τα frames της κάμερας και να παράγει (μια ασπρόμαυρη) εικόνα στην οποία μετά θα κάνουμε επεξεργασία για την τελική κατάσταση της εφαρμογής.

Έπειτα από τις αλλαγές που προαναφέραμε στο project αλλά και στους κώδικες(θα αναφερθούμε στο επόμενο κεφάλαιο εκτενέστερα), η τελική μορφή της εργασίας μας είναι κάπως έτσι (εικόνα 8.11).



Εικόνα 8.11 Τελική

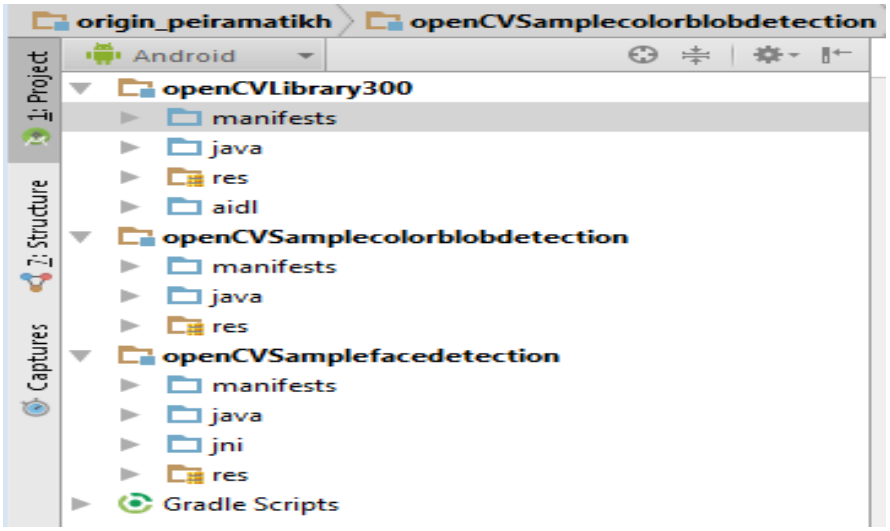
8.3 Ανάλυση της εφαρμογής Lunar1

Σε αυτό το σημείο θα αναλύσουμε τα σημεία της εφαρμογής μας. Δηλαδή, θα δούμε ξεχωριστά τη λειτουργία κάθε κλάσης, ποιες εντολές έχουμε προσθέσει από το αρχικό πλάνο, τη λειτουργία αυτών των εντολών και την επεξήγηση άλλων αρχείων που δημιουργούνται αυτόματα.

Μετά το τέλος αυτού του κεφαλαίου θα έχουμε μια πιο ολοκληρωμένη εικόνα για το Android Studio και της εφαρμογής μας που θα αναλυθεί.

8.3.1 Τι είναι το AndroidManifest.xml;

Ανοίγοντας την εφαρμογή, παρατηρούμε το AndroidManifest.xml (εικόνα 8.12). Αυτό γίνεται διότι, βλέπουμε πως και τα τρία αρχεία που έχουμε στην εφαρμογή μας έχουν το δικό τους manifest (όπου κρύβεται το androidManifest.xml αρχείο).



Εικόνα 8.12 Manifest

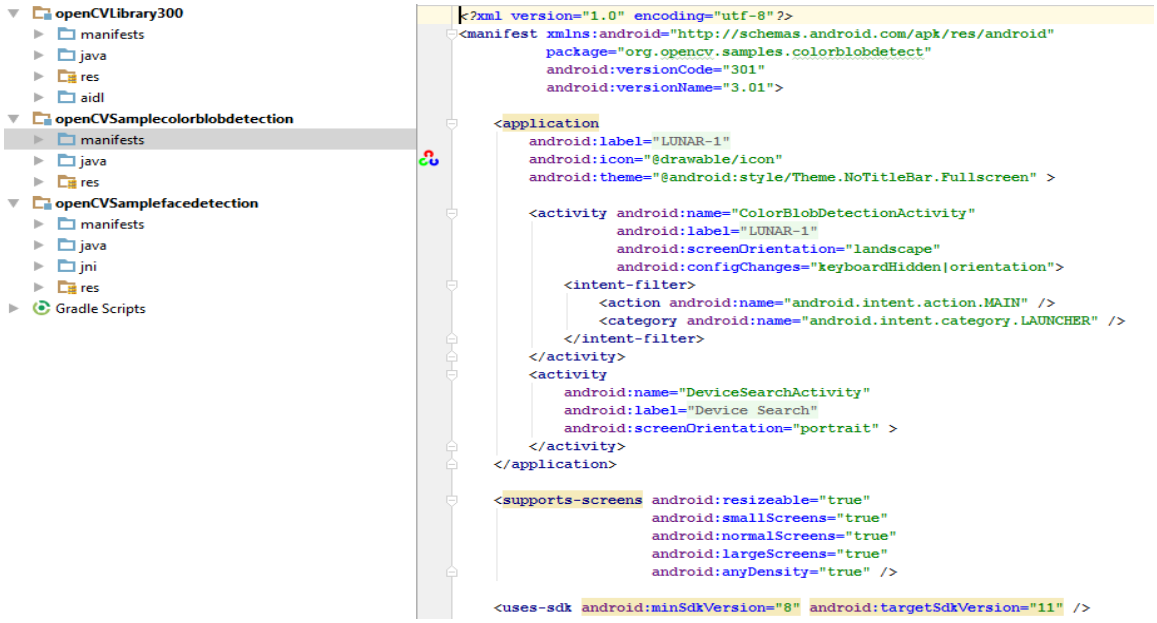
Το androidmanifest.xml είναι το σημαντικότερο αρχείο που χρειάζεται κάθε εφαρμογή android για να ξεκινήσει. Από την κατάληξη του αρχείου καταλαβαίνουμε ότι το περιεχόμενο του είναι γραμμένο σε μορφή xml. Πριν όμως περάσουμε στο εσωτερικό του αρχείου αυτού, καλό θα ήταν, να γνωρίζουμε το λόγο ύπαρξης και τη λειτουργία του.

Όπως καταλαβαίνουμε, το αρχείο αυτό είναι, αυτό που αναζητά πρώτα το Android Studio, για να ξέρει πώς θα εκτελέσει την εφαρμογή, μαζί με όλα τα στοιχεία που την αποτελούν.

Οι υπηρεσίες που προσφέρει αυτό το αρχείο είναι:

- Αναγνωρίζει και δίνει συγκεκριμένο όνομα στα java πακέτα της εφαρμογής, στα οποία μέσα βρίσκονται οι κλάσεις
- Κάνει περιγραφή των στοιχείων, από τα οποία αποτελείται μια εφαρμογή (activities, services κτλ)
- Προσδιορίζει τα processes, που χρειάζονται για την υποστήριξη της εφαρμογής
- Προσδιορίζει τα δικαιώματα, τα οποία πρέπει να έχει η εφαρμογή για να μπορεί να αλληλεπιδρά με άλλες
- Ορίζει τα δικαιώματα, που οι άλλες εφαρμογές πρέπει να έχουν για να χρησιμοποιήσουν την δική μας
- Κάνει δήλωση του κατώτερου επιπέδου API, που υποστηρίζει η εφαρμογή μας
- Προσθέτει τις βιβλιοθήκες, που χρειάζεται η εφαρμογή για να τρέξει
- Καλεί και κάνει χρήση των Instrumentation κλάσεων, οι οποίες έχουν πληροφορίες για την σωστή εκτέλεση της εφαρμογής.

Κάνοντας διπλό κλικ πάνω στο manifest ανοίγει το αρχείο androidmanifest.xml και έχει την παρακάτω μορφή (εικόνα 8.13).



Εικόνα 8.13 anroidmanifest.xml

Ανοίγοντας αυτό το αρχείο σε μια εφαρμογή, όπως η δικιά μας, θα δούμε πάρα πολλά elements(εικόνα 8.14). Τώρα, θα προσπαθήσουμε να τα διαλευκάνουμε. Αρχικά, θα πρέπει να ξέρουμε, πως από όλα αυτά τα elements, δύο είναι τα πιο σημαντικά, το **manifest(κόκκινο)** και το **application(μπλέ)**. Αυτά τα elements θα πρέπει να υπάρχουν σε όλα τα manifest αρχεία και θα πρέπει να εμφανίζονται μόνο μια φορά. Όλα τα άλλα elements μπορούν να είναι γραμμένα πολλές φορές στο ίδια αρχείο.

Εικόνα 8.14 Παρουσίαση manifest-application

Τι είναι λοιπόν το manifest;

Είναι το root element του αρχείου AndroidManifest.xml και πρέπει να περιέχει ένα application element. Ορίζει και δύο attributes: **xmlns:android** - **package:**

Xmlns:android (εικόνα 8.15)

Ορίζει το Namespace και πρέπει πάντα να δείχνει στο http://schemas.android.com/apk/res/android.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

Εικόνα 8.15 xmlns:android

Package: (εικόνα 8.16)

Είναι το java πακέτο μέσα στο οποίο, βρίσκεται η εκτελέσιμη java class της εφαρμογής μας.

```
package="org.opencv.samples.colorblobdetect"
```

Εικόνα 8.16 package

Android:versionCode (εικόνα 8.17)

Αυτός είναι ο εσωτερικός αριθμός, που δείχνει σε ποια έκδοση της εφαρμογής βρισκόμαστε. Το **android:versionCode** δεν εμφανίζεται στους χρήστες. Όσο πιο μεγάλος είναι ο αριθμός, τόσο πιο πρόσφατη είναι η έκδοση της εφαρμογής μας. Επίσης, ο αριθμός πρέπει πάντα να είναι ακέραιος.

```
android:versionCode="301"
```

Εικόνα 8.17 android:versionCode

Android:versionName (εικόνα 8.18)

Αυτός ο αριθμός εμφανίζεται στους χρήστες. Ο λόγος που υπάρχει αυτή η εντολή είναι απλά για να δείχνει στους χρήστες σε ποια έκδοση τρέχει η εφαρμογή.

```
android:versionName="3.01"
```

Εικόνα 8.18 android:versionName

Uses-sdk (εικόνα 8.19)

Ο λόγος που υπάρχει το uses-sdk είναι για να τηρεί την συμβατότητα της εφαρμογής, της οποίας αναπτύσσουμε με μια ή περισσότερες εκδόσεις της πλατφόρμας του Android μέσω του αριθμού του επιπέδου API. Δηλαδή, αυτό το στοιχείο χρησιμοποιείται για να καθορίσει το επίπεδο API και όχι τον αριθμό έκδοσης του SDK. Στην εικόνα 4.8 φαίνεται ο αριθμός του API που είναι 8. Αυτό δηλώνει ότι, η εφαρμογή υποστηρίζει από Android με λειτουργικό 2.2 και πάνω.

```
<uses-sdk android:minSdkVersion="8"
```

Εικόνα 8.19 uses-sdk

Android:minSdkVersion (εικόνα 8.20)

Είναι ο ακέραιος αριθμός που ορίζει το ελάχιστο επίπεδο API που απαιτείται για να τρέξει η εφαρμογή. Απαγορεύει στον χρήστη να εγκαταστήσει οποιαδήποτε εφαρμογή έχει μικρότερο API από αυτό που δηλώνεται εδώ.

```
<uses-sdk android:minSdkVersion="8"
```

Εικόνα 8.20 android:minSdkVersion

Android:targetSdkVersion (εικόνα 8.21)

Ορίζει το επίπεδο API στο οποίο στοχεύει η εφαρμογή. Αν δεν το έχουμε ορίσει παίρνει αυτόματα την τιμή του mhnSdkVersion.

```
android:targetSdkVersion="11"
```

Εικόνα 8.21 targetSdkVersion

Το επόμενο xml αρχείο που βλέπουμε είναι το element application και συντάσσεται όπως στην εικόνα 8.22.

```
<application
    android:label="LUNAR-1"
    android:icon="@drawable/icon"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >

    <activity android:name="ColorBlobDetectionActivity"
              android:label="LUNAR-1"
              android:screenOrientation="landscape"
              android:configChanges="keyboardHidden|orientation">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="DeviceSearchActivity"
        android:label="Device Search"
        android:screenOrientation="portrait" >
    </activity>
</application>
```

Εικόνα 8.22 elements-applications

Android:icon (εικόνα 8.23)

Εδώ, ορίζεται το Path του φακέλου από το οποίο θα προέλθει η εικόνα της εφαρμογής. Αν δεν ορίσουμε κάτι εμείς τότε το android Studio θα βάλει μια εικόνα που έχει by default.

```
android:icon="@drawable/icon"
```

Εικόνα 8.23 android:icon

Android:label (εικόνα 8.24)

Εδώ ορίζεται το όνομα με το οποίο θα εμφανιστεί η εφαρμογή. Το όνομα προέρχεται από το strings.xml αρχείο που βρίσκεται στο φάκελο values.

```
android:label="LUNAR-1"
```

Εικόνα 8.24 android:label

Android:theme (εικόνα 8.25)

Αναφέρεται στο default θέμα της εφαρμογής, το οποίο προέρχεται από το αρχείο styles.xml από τον φάκελο values.

```
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
```

Εικόνα 8.25 android:theme

Το επόμενο στοιχείο, που θα αναλύσουμε είναι το activity. Η σύνταξη αυτού είναι κάπως έτσι(εικόνα 8.26).

```
<activity android:name="ColorBlobDetectionActivity"
          android:label="LUNAR-1"
          android:screenOrientation="landscape"
          android:configChanges="keyboardHidden|orientation">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name="DeviceSearchActivity"
    android:label="Device Search"
    android:screenOrientation="portrait" >
</activity>
```

Εικόνα 8.26 elements-activity

Το activity ορίζει το ποια είναι java activity καθώς και το που βρίσκεται στην εφαρμογή. Επίσης είναι υπεύθυνη για την λειτουργία της εφαρμογής και για την δημιουργία visual interface που θα εμφανιστεί στην οθόνη της συσκευής. Δηλαδή, το AndroidManifest είναι το αρχείο, στο οποίο αναζητεί αρχικά η Android πλατφόρμα για να μπορέσει να βρεί την activity κλάση που περιέχει το java πρόγραμμα εκκίνησης της εφαρμογής και να εμφανιστεί στην οθόνη μας. Δύο είναι οι παράμετροι, που είναι απαραίτητο να υπάρχουν μέσα στο activity. Το android:name – android:label.

Android:name

Αυτό είναι το όνομα της κλάσης, μέσα στην οποία υλοποιείται το activity. Το όνομα πρέπει να είναι σε “fully qualified class” μορφή δείχνοντας ακριβώς την τοποθεσία της.

Andoid:label

Είναι το όνομα με το οποίο θα εμφανιστεί στην οθόνη η εφαρμογή.

Τέλος, φτάσαμε στο τελευταίο element που συναντάμε. Το intent-filter (εικόνα 8.27). Μέσα σε αυτό ορίζουμε όλες τις πράξεις στις οποίες ανταποκρίνεται η εφαρμογή μας. Δηλαδή, ορίζουμε τις δυνατότητος της εφαρμογή φιλτράροντας και αφήνοντας έξω εκείνες τις πράξεις που δεν έχουν ουσία. Βασικά το Intent filter “διαφημίζει” τι είδους πράξεις μπορεί να εκτελέσει. Μέσα σε αυτό το element πρέπει απαραίτητα να υπάρχει το element action ενώ αντίθετα το category είναι προαιρετικό.

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Εικόνα 8.27 Intent-filter

Action (εικόνα 8.28)

Το action προσθέτει μία ενέργεια στο intent-filter και κατά συνέπεια στην εφαρμογή.

```
<action android:name="android.intent.action.MAIN" />
```

Εικόνα 8.28 action

Android:name (εικόνα 8.29)

Ορίζουμε το όνομα της πράξης. Ο τρόπος, με τον οποίο γίνεται αυτό είναι να ορίσουμε το name μετά το action, για παράδειγμα ACTION_MAIN. Αυτός ο ορισμός σημαίνει ότι η κλάση ορίζεται ως σημείο έναρξης της εφαρμογής και δεν περιμένει να λάβει καμία πληροφορία data. Αυτό σημαίνει ότι δεν έχει input και δεν βγάζει output.

```
<action android:name="android.intent.action.MAIN" />
```

Εικόνα 8.29 action android:name

Category (εικόνα 8.30)

Προσθέτει μια κατηγορία στο intent-filter.

```
<category android:name="android.intent.category.LAUNCHER" />
```

Εικόνα 8.30 category

Android:name (εικόνα 8.31)

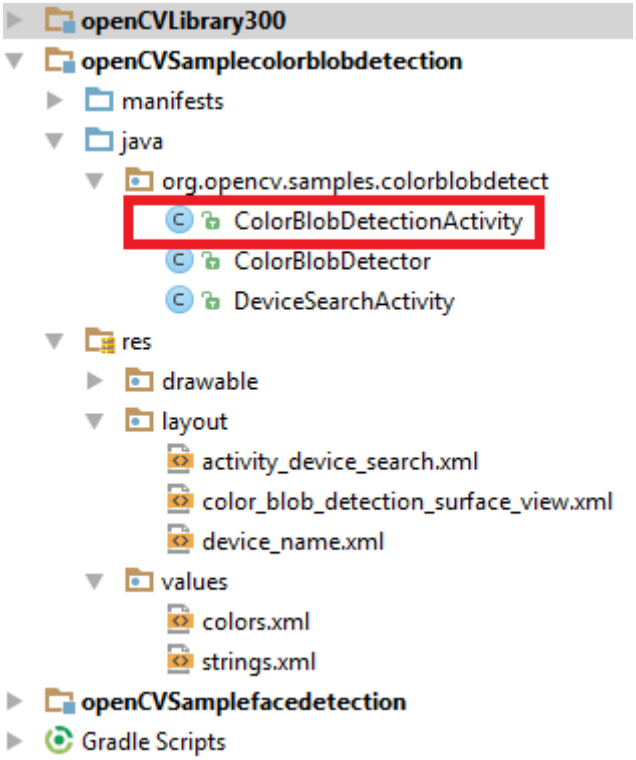
Το activity θα είναι το πρώτο activity που θα τρέξει όταν ξεκινάει η εφαρμογή.

```
<category android:name="android.intent.category.LAUNCHER" />
```

Εικόνα 8.31 category action:name

8.3.2 Επεξήγηση των φακέλων στο project

Στην εικόνα 8.32 βλέπουμε τους φακέλους ένος project σε Android Studio.



Εικόνα 8.32 Παρουσίαση των φακέλων του project

Αρχικά, κάνοντας διπλό κλικ στον φάκελο manifest ανοίγει το androidmanifest.xml που είπαμε προηγουμένως και είδαμε την μορφή του. Στον φάκελο java υπάρχει ένας άλλος φάκελος, ο οποίος αναγράφει το πακέτο του project και μέσα σε αυτόν είναι οι κλάσεις μας. Η μια είναι η κύρια κλάση. Στην εφαρμογή μας είναι αυτή στο κόκκινο πλαίσιο. Στον φάκελο res έχουμε τρία διαφορετικά μονοπάτια.

res/drawable→ Σε αυτό το σημείο έχουμε αποθηκεύσει φωτογραφίες της εφαρμογής

res/layout→ Σε αυτό το σημείο φτιάχνουμε τα xml αρχεία από τα οποία αντλούν πληροφορίες τα activity που καλούμε και παρουσιάζεται στον χρήστη η οθόνη της εφαρμογής

res/values→ Σε αυτό το σημείο αποθηκεύουμε αρχεία της μορφής xml, τα οποία έχουν πληροφορίες για τα χρώματα της εφαρμογής, το background, το theme και τα χρώματα που έχουμε χρησιμοποιήσει.

Στην συνέχεια βλέπουμε ένα αρχείο **Gradle**. Το gradle είναι μια κατασκευή αυτοματοποιημένης βιβλιοθήκης, που επιτρέπει την ταξινόμηση των έργων που συνυπάρχουν σε ένα project μέσω μιας σειράς αρχείων ρύθμισης. Σε αυτό το αρχείο περιλαμβάνεται, ο καθορισμός του έργου και πως αυτός πρέπει να δημιουργηθεί, τι πρέπει να πληρούν οι εξαρτήσεις για την επιτυχή κατασκευή του έργου. Το αρχείο αυτό εγγυάται την ευελιξία που θα έχουμε στην κατασκευή του έργου μας. Επίσης, το αρχείο αυτό δημιουργείται αυτόματα με το import ενός project.

8.3.3 Activities

Με τον όρο activity περιγράφουμε το κάθε γραφικό περιβάλλον μιας εφαρμογής, όπως για παράδειγμα η αρχική σελίδα μιας εφαρμογής. Μέσω αυτού του γραφικού περιβάλλοντος μας δίνονται επιλογές να μεταβούμε σε άλλα activities τα οποία δίνουν περισσότερες δυνατότητες στην εφαρμογή.

Κάθε εφαρμογή έχει τουλάχιστον ένα activity ενώ, επαγγελματικές εφαρμογές έχουν παραπάνω. Η δικιά μας εφαρμογή έχει πολλά (εικόνα 8.33). Στις εφαρμογές με πολλά activities, πάντα ένα είναι το κύριο, το οποίο παρουσιάζεται στον χρήστη όταν ξεκινάει η εφαρμογή. Όταν κάποιο άλλο activity θέλουμε να βγει στην εφαρμογή μας, τότε θα πρέπει να το καλέσουμε για να πάρει την θέση του άλλου.

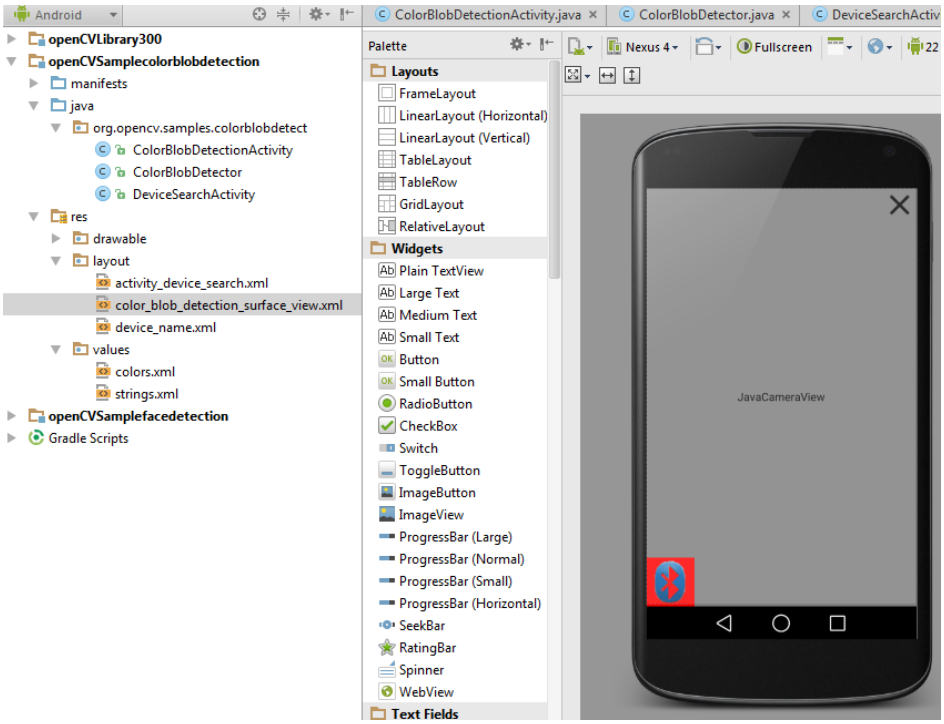
Κάθε φορά που θέλουμε ένα νέο activity, το παλιό μπαίνει σε μια λίστα (stack), όπου αυτή έχει την λογική της λειτουργίας LIFO(last input first output). Αυτό σημαίνει ότι, όταν εμείς πατήσουμε το κουμπί back, αυτό τραβάει από την λίστα το τελευταίο activity που μπήκε και το εμφανίζει στην οθόνη της εφαρμογής μας.

Σε περίπτωση που, θέλουμε να εξαναγκάσουμε ένα activity να σταματήσει διότι έρχεται ένα άλλο στο προσκήνιο, τότε το παλιό activity ενημερώνεται για την αλλαγή της κατάστασης του μέσα από μια σειρά μεθόδων που ονομάζονται «lifecycle callback methods» . Υπάρχουν αρκετές μέθοδοι από τις οποίες ένα activity περνάει από τη στιγμή της δημιουργίας του μέχρι την στιγμή της καταστροφής του. Αυτές οι μέθοδοι είναι γνωστές και ως «stages» μέσα από τις οποίες περνάει ένα activity κατά την διάρκεια της ζωής του. Για έναν προγραμματιστή Android εφαρμογών είναι σημαντικό να γνωρίζει αυτή την διαδικασία, ώστε να μπορεί να επέμβει σωστά στο πρόγραμμα και να χρησιμοποιεί αυτές τις φάσεις του activity υπέρ του. Ένα τέτοιο παράδειγμα είναι, όταν ένα activity σταματάει μπορούμε εκείνη την στιγμή να ελευθερώσουμε όλα τα connections που έχουμε πάνω σε μια βάση. Αυτό ισχύει κυρίως για εφαρμογές που χρειάζονται μια βάση για να τραβήξουν δεδομένα. Η συγκεκριμένη εφαρμογή ανήκει σε αυτές.

Στην παρακάτω εικόνα 8.33 παρουσιάζεται το σημείο του κώδικα μας που κάνει extends το activity.

Εικόνα 8.33 Παρουσίαση του extends Activity

Για τη δημιουργία ενός activity χρειάζεται να φτιάξουμε την δική μας κύρια κλάση, η οποία θα κάνει extends(κληρονομεί) από την κλάση activity. Το activity που έχουμε δημιουργήσει στην δικιά μας εργασία, βρίσκεται στο φάκελο res/layout και φορτώνει τα γραφικά του στοιχεία κάνοντας χρήση xml αρχείου (εικόνα 8.34).



Εικόνα 8.34 Παρουσίαση του activity που εκτελείται πρώτο

Στην εικόνα 8.35 βλέπουμε με ποία εντολή στην κλάση όπου κάνουμε το extends activity δηλώνουμε ότι αυτό το activity θα πρέπει να εκτελεστεί τώρα.

```
setContentView(R.layout.color_blob_detection_surface_view);
```

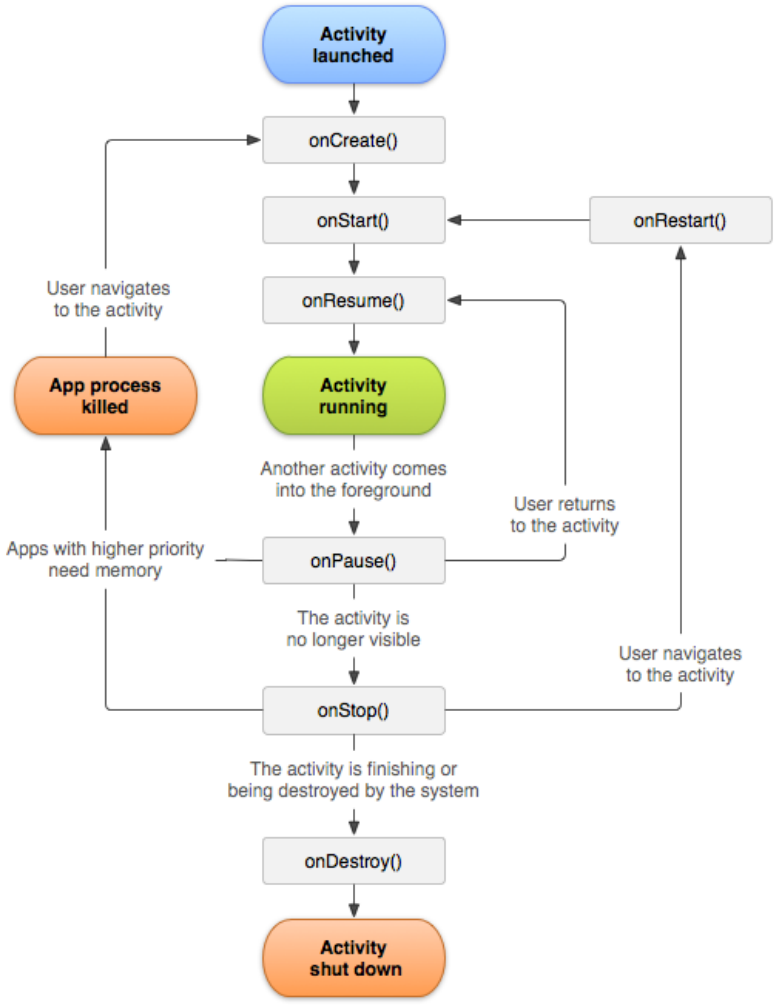
Εικόνα 8.35 Δήλωση του activity που θέλω τώρα να εκτελεστεί

Τώρα, φορτώνεται από τον φάκελο res/layout το “color_blob_detection_surface_view.xml” το οποίο έχει το design που φαίνεται στην εικόνα 8.36.

Κάθε activity κλάση ορίζεται από μια σειρά από events που καθορίζουν το lifecycle του activity. Πιο συγκεκριμένα, το κάθε activity έχει τα εξής events:

- onCreate() → καλείται όταν ξεκινάει το activity για πρώτη φορά
- onStart() → καλείται όταν το activity γίνεται ορατό στον χρήστη
- onResume() → καλείται όταν το activity χρησιμοποιείται από τον χρήστη
- onPause() → καλείται όταν το τρέχον activity είναι σε κατάσταση pause και ένα προηγούμενο που ήταν σε αδράνεια ήρθε στο προσκήνιο
- onStop() → καλείται όταν το activity δεν είναι πλέον ορατό στον χρήστη
- onDestroy() → καλείται όταν το activity έχει διαγραφεί από την stack
- onRestart() → καλείται όταν το activity έχει σταματήσει και ξεκινάει πάλι

Εξ ορισμού, το activity που έχουμε δημιουργήσει περιέχει το onCreate() event μέσα στο οποίο έχουμε γράψει τον κώδικα για το design. Η εικόνα 8.36 μας κάνει μια περιγραφή από τα events που περνάει ένα activity.



Εικόνα 8.36 Τα events ενός activity

Παρακάτω βλέπουμε τα events του activity στην εφαρμογή μας. Στην εικόνα 8.37 βλέπουμε το event **onCreate().** Εδώ, καλείται το activity που θα εμφανιστεί πρώτη φορά στην οθόνη της εφαρμογής. Η εντολή με μπλε χρώμα είναι υπεύθυνη για τις διαστάσεις της οθόνης και να εμφανίζεται η εφαρμογή μας στο κινητό σε πλήρες παράθυρο. Η εντολή με κόκκινο διατηρεί το πρώτο activity, ώστε στο χρήστη να φαίνεται κάτι. Χρησιμοποιείται κυρίως σε παιχνίδια για να αλλάζουν γρήγορα στο μάτι του χρήστη τα activity. Με την εντολή σε πράσινο πλαίσιο δηλώνουμε που θα βρουμε το πρώτο activity που θα έχει η εφαρμογή μας. Είναι στο φάκελο res/layout. Το ibtn είναι το button που κάνει connect με χρήση Bluetooth και το cross είναι το button που κλείνει την εφαρμογή μας.

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    Log.i(TAG, "called onCreate");
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    setContentView(R.layout.color_blob_detection_surface_view);

    ibtn=(ImageButton) findViewById(R.id.imageButton1);
    cross=(ImageView) findViewById(R.id.imageView);
    cross.setOnClickListener((view) -> {
        if (mConnectedThread == null) {
            finish();
            System.exit(0);
        }
        else {
            stopCar();
            finish();

            System.exit(0);
        }

    });
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.color_blob_detection_activity_surface_view);
    mOpenCvCameraView.setCvCameraViewListener(this);
}
```

Εικόνα 8.37 onCreate()

Στην εικόνα 8.38 βλέπουμε το **onPause()**. Εδώ όσο η κάμερα δεν ανιχνεύει κάτι στο χώρο, κάνει νέα αναζήτηση.

```
@Override
public void onPause()
{
    super.onPause();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}
```

Εικόνα 8.38 onPause()

Στην εικόνα 8.39 βλέπουμε το **onResume()**. Εδώ κάνει τον έλεγχο αν έχουμε στο κινητό μας το OpenCV Manager.Αν δεν το έχουμε μας εμφανίζει καινούργιο Activity στο PlayStore και μας ζητάει να κατεβάσουμε την εφαρμογή. Αν δεχτούμε

γυρνάμε πίσω στην εφαρμογή αλλιώς η εφαρμογή κλείνει.

```
@Override
public void onResume()
{
    super.onResume();
    if (!OpenCVLoader.initDebug()) {
        Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0, this, mLoaderCallback);
    } else {
        Log.d(TAG, "OpenCV library found inside package. Using it!");
        mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
    }
}
```

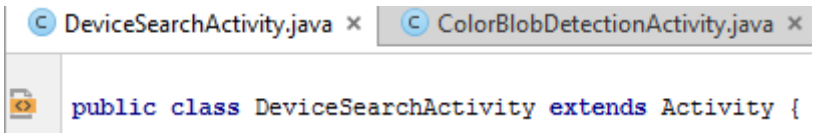
Εικόνα 8.39 onResume()

Στην εικόνα 8.40 βλέπουμε το **onDestroy().** Εδώ το activity ξανά καλείται από την stack.

```
public void onDestroy() {
    super.onDestroy();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}
```

Εικόνα 8.40 onDestroy

Δεν έχουμε όμως ένα activity όπως αναφέραμε και προηγουμένως. Το δεύτερο activity βρίσκεται μέσα στην κλάση “DeviceSearchActivity” (εικόνα 8.41).



Εικόνα 8.41Το δεύτερο activity

Στην εικόνα 8.42 βλέπουμε το event **onCreate().** Στο μπλε πλαίσιο εξαναγκάζει το νέο activity να ξεκινήσει και πετάει το παλιό στη stack και παίρνει το νέο activity από τον φάκελο res/layout με όνομα “activity_device_search”. Στην συνέχεια δηλώνονται τα τρία text. Στο κόκκινο πλαίσιο δίνει από ένα όνομα στο πρώτο στοιχείο του πίνακα «pairedDevices” και «nesDevices” αντίστοιχα. Στο πρώτο πράσινο πλαίσιο βρίσκει και τοποθετεί στην ListView τις pairedDevices. Στο δεύτερο πράσινο πλαίσιο βρίσκει και τοποθετεί στην ListView για τα καινούργια Discovered Devices. Στο πρώτο μοβ πλαίσιο κάνει εγγραφή όταν βρίσκει μια συσκευή με ενεργά Bluetooth. Στο δεύτερο μοβ πλαίσιο κάνει εγγραφή της συσκευής όταν η αναζήτηση τελειώσει. Τέλος, παίρνει την διεύθυνση του Bluetooth adapter και την αποθηκεύει στο mBluetoothAdapter και κάνει έλεγχο. Αν είναι μη διαθέσιμο εμφανίζεται το μήνυμα “Bluetooth not supported..”.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_device_search);
    txt1 = (TextView) findViewById(R.id.textView1);
    txt2 = (TextView) findViewById(R.id.textView2);
    txt3 = (TextView) findViewById(R.id.textView3);
    // Initialize array adapters: One for already paired devices and one for newly discovered devices
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
    mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
    // Find and set up the ListView for paired devices
    ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);
    // Find and set up the ListView for newly discovered devices
    ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
    newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
    newDevicesListView.setOnItemClickListener(mDeviceClickListener);
    // Register for broadcasts when a device is discovered
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);
    // Register for broadcasts when discovery has finished
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);
    // Get local Bluetooth adapter
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    // If adapter is null, then Bluetooth is not supported
    if (mBluetoothAdapter == null) {
        Toast.makeText(getBaseContext(), "Bluetooth not supported...", Toast.LENGTH_LONG).show();
        finish();
        return;
    }
}
```

Εικόνα 8.42 onCreate() στο δεύτερο activity

Στην εικόνα 8.43 θα δούμε το onStart(). Εδώ, βλέπουμε ότι, ρωτάει αν υπάρχει σύνδεση με το Bluetooth. Αν όχι πρέπει να κάνουμε πάλι αναζήτηση.

```
@Override
protected void onStart() {
    super.onStart();
    // If BT is not ON, request to enable it
    // setupCommand() will then be called during onActivityResult
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
    }
    else {
      discoverBT();
    }
}
```

Εικόνα 8.43 onStart() στο δεύτερο activity

Στην επόμενη εικόνα 8.44 βλέπουμε το onResume(). Διατηρεί το activity αυτό.

```
@Override
protected void onResume() {
    super.onResume();
}
```

Εικόνα 8.44 onResume() στο δεύτερο activity

Στην εικόνα 8.45 βλέπουμε το onDestroy(). Καλείται όταν εξασφαλίζει τη σύνδεση με το Bluetooth.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    // Make sure we're not doing discovery anymore
    if (mBluetoothAdapter != null) {
        mBluetoothAdapter.cancelDiscovery();
    }

    // Unregister broadcast listeners
    this.unregisterReceiver(mReceiver);
}
```

Εικόνα 8.45 onDestroy() στο δεύτερο activity

Το επόμενο activity είναι στην κλάση “FdActivity”. Στην εικόνα 8.46 βλέπουμε το onCreate(). Η μόνη διαφορά με πριν είναι ότι τώρα φτιάχνει το mOpenCvCameraView ,με κόκκινο χρώμα, παίρνοντας τώρα το activity του “fd_activity_surface_view” που βρίσκεται στο φάκελο res/layout του OpenCVSampleFaceDetection.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    Log.i(TAG, "called onCreate");
    super.onCreate(savedInstanceState);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    setContentView(R.layout.face_detect_surface_view);

    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surface_view);
    mOpenCvCameraView.setCvCameraViewListener(this);
}
```

Εικόνα 8.46 onCreate() στο τρίτο activity

Στην εικόνα 8.47 βλέπουμε το onPause(). Εδώ, σταματάει το activity όταν ανιχνεύσει κάτι στον χώρο.

```
@Override
public void onPause()
{
    super.onPause();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}
```

Εικόνα 8.47 onPause() στο τρίτο activity

Στην εικόνα 8.48 βλέπουμε το onResume(). Ελέγχει αν έχει το OpenCV Manager.Αν όχι το κατεβάζει από PlayStore και συνεχίζει.

```
@Override
public void onResume()
{
    super.onResume();
    if (!OpenCVLoader.initDebug()) {
        Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0, this, mLoaderCallback);
    } else {
        Log.d(TAG, "OpenCV library found inside package. Using it!");
        mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
    }
}
```

Εικόνα 8.48 onResume() στο τρίτο activity

Στην εικόνα 8.49 βλέπουμε το onDestroy(). Με το που ανιχνεύσει κάτι η κάμερα διαγράφεται από την stack.

```
public void onDestroy() {
    super.onDestroy();
    mOpenCvCameraView.disableView();
}
```

Εικόνα 8.49 onDestroy () στο τρίτο activity

8.3.4 Εξήγηση κώδικα Lunar1

Αρχικά, πηγαίνουμε στην κύρια κλάση μας που είναι η “ColorBlobDetectionActivity.class”. Εδώ αξίζει να αναφέρουμε στον χώρο των δηλώσεων αυτό που εμφανίζεται στην εικόνα 8.50.

```
private static final byte FORWARD = 0x5;
private static final byte BACKWARD = 0x6;
private static final byte LEFT = 0x2;
private static final byte RIGHT = 0x3;
private static final byte STOP = 0x9;
```

Εικόνα 8.50 Εντολές για να καταλαβαίνει και το arduino

Με αυτές τις εντολές δηλώνουμε στο Arduino ότι όταν θα βλέπει την κατάλληλη λέξη θα ανοίγει την κατάλληλη πόρτα(θα την ενεργοποιεί).

Στην εικόνα 8.51 βλέπουμε την κάμερα να αρχικοποιεί τις μεταβλητές που φαίνονται στην εικόνα. Επίσης, φαίνεται ότι θέλει να είναι όλα στο ίδιο φάσμα και για αυτό τους δηλώνει φάσμα από 0-255.

```
public void onCameraViewStarted(int width, int height) {
    mRgba = new Mat(height, width, CvType.CV_8UC4);
    mDetector = new ColorBlobDetector();
    mSpectrum = new Mat();
    mBlobColorRgba = new Scalar(255);
    mBlobColorHsv = new Scalar(255);
    SPECTRUM_SIZE = new Size(200, 64);
    CONTOUR_COLOR = new Scalar(255,0,0,255);
}
```

Εικόνα 8.51 Δήλωση νέου φάσματος

Στην εικόνα 8.52 έχουμε τον κώδικα που ενεργοποιείται όταν αγγίζουμε την οθόνη του κινητού. Σε γενικές γραμμές, αυτός ο κώδικας παίρνει μια εικόνα από τις συντεταγμένες που σχεδιάζονται με ένα άγγιγμα από τον χρήστη. Στη συνέχεια μετατρέπει το χρωματικό χώρο και αλλάζει το μέγεθος με το μέγεθος του φάσματος.

Στο κόκκινο πλαίσιο είναι όλες αυτές οι εντολές, που χρειαζόμαστε για να πάρουμε την ανάλυση της οθόνης. Με την if φιλτράρουμε ουσιαστικά τον κώδικα επιστρέφοντας λάθος σε περίπτωση που το αντικείμενο είναι έξω από τον χώρο που βλέπει το κινητό ή αν δεν είναι ενεργοποιημένο το Bluetooth (σε αυτό το σημείο δεν κάνει τίποτα).

Στο επόμενο πράσινο πλαίσιο έχουμε τις εντολές που χρειάζονται για να καταλάβει τον χώρο που λαμβάνει το αντικείμενο ,δηλαδή το μέγεθος του.

Στην συνέχεια, στο μοβ πλαίσιο μετατρέπει σε HSV color. Σε αυτό το σημείο θα επεκταθούμε λίγο.

Συνήθως, για την ανίχνευση αντικειμένων με συγκεκριμένο χρώμα, θα πρέπει πρώτα να μετατρέψουμε τον χώρο αο RGB -> HSV. Το HSV είναι ένας χρωματικός χώρος όπου το V συμβολίζει το φωτισμό, έτσι ώστε τα χρώματα να αντιπροσωπεύονται με τις πραγματικές τους τιμές που αντανακλούν από την πηγή .Το H,S βοηθάει στην συγκρίσει των χρωμάτων ωστε να καταλάβουμε πιο συγκεκριμένο χρώμα είναι.

Στην συνέχεια, στο μπλε πλαίσιο υπολογίζεται το μέσο χρώμα από την περιοχή που πατήθηκε και κάνει convert από Hsv→RGBA.

Στο κόκκινο πλαίσιο κάνει resize της εικόνας με το νέο φάσμα της εικόνας. Τέλος, καθαρίζει όλους τους πίνακες.

```
public boolean onTouch(View v, MotionEvent event) {
    int cols = mRgba.cols();
    int rows = mRgba.rows();

    int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;
    int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;

    int x = (int)event.getX() - xOffset;
    int y = (int)event.getY() - yOffset;

    Log.i(TAG, "Touch image coordinates: (" + x + ", " + y + ")");

    if ((x < 0) || (y < 0) || (x > cols) || (y > rows) || (mConnectedThread == null)) return false;

    Rect touchedRect = new Rect();

    touchedRect.x = (x>4) ? x-4 : 0;
    touchedRect.y = (y>4) ? y-4 : 0;

    touchedRect.width = (x+4 < cols) ? x + 4 - touchedRect.x : cols - touchedRect.x;
    touchedRect.height = (y+4 < rows) ? y + 4 - touchedRect.y : rows - touchedRect.y;

    Mat touchedRegionRgba = mRgba.submat(touchedRect);

    Mat touchedRegionHsv = new Mat();
    Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv, Imgproc.COLOR_RGB2HSV_FULL);

    // Calculate average color of touched region
    mBlobColorHsv = Core.sumElems(touchedRegionHsv);
    int pointCount = touchedRect.width*touchedRect.height;
    for (int i = 0; i < mBlobColorHsv.val.length; i++)
        mBlobColorHsv.val[i] /= pointCount;

    mBlobColorRgba = converScalarHsv2Rgba(mBlobColorHsv);

    Log.i(TAG, "Touched rgba color: (" + mBlobColorRgba.val[0] + ", " + mBlobColorRgba.val[1] +
            ", " + mBlobColorRgba.val[2] + ", " + mBlobColorRgba.val[3] + ")");

    mDetector.setHsvColor(mBlobColorHsv);

    Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
    mIsColorSelected = true;
    touchedRegionRgba.release();
    touchedRegionHsv.release();
    return false; // don't need subsequent touch events
}
```

Εικόνα 8.52 Επεξήγηση κώδικα onTouch

Στην εικόνας 8.53 βλέπουμε το τμήμα του κώδικα στο οποίο η κάμερα σε frame με τις κατάλληλες εντολές μας επιστρέφει το mRgba.

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
    mRgba = inputFrame.rgba();

    if (mIsColorSelected) {
        mDetector.process(mRgba);
        List<MatOfPoint> contours = mDetector.getContours();
        Log.e(TAG, "Contours count: " + contours.size());
        Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);

        Mat colorLabel = mRgba.submat(4, 68, 4, 68);
        colorLabel.setTo(mBlobColorRgba);

        Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 + mSpectrum.cols());
        mSpectrum.copyTo(spectrumLabel);
    }

    return mRgba;
}
```

Εικόνα 8.53 Frame εικόνας και επιστροφή mRgba

Στην εικόνα 8.54 βλέπουμε ότι, θέλει να πετύχει την επικοινωνία με το Bluetooth χρησιμοποιώντας κάποιο ελεύθερο thread. Αρχικά, δέχεται την max address της συσκευής, η οποία κοιτάει αν είναι ίδια με αυτήν που του έχουμε ορίσει εμείς ότι είναι η σωστή. Αν ναι, παίρνει το object της Bluetooth Device. Στην συνέχεια κοιτάει αν κάποιο νήμα είναι ελεύθερο. Αν όχι, τότε του λέει ελευθερώσου και αυτό το thread ξεκινάει την μεταφορά των δεδομένων της συσκευής Bluetooth που έχουμε στην εργασία μας.

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQUEST_CONNECT_DEVICE:
            if (resultCode == Activity.RESULT_OK) {
                // Get the device MAC address
                String address = data.getExtras().getString(DEVICE_ADDRESS);
                // Get the BLuetoothDevice object
                BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
                // Cancel any thread currently running a connection
                if (mConnectedThread != null) {
                    mConnectedThread.cancel();
                    mConnectedThread = null;
                }
                // Attempt to connect to the device
                mConnectThread = new ConnectThread(device);
                mConnectThread.start();
            }
    }
}
```

Εικόνα 8.54 OnActivityResults

Στην εικόνα 8.55 βλέπουμε τον κώδικα που είναι υπεύθυνος για την μετατροπή του φάσματος της εικόνας από HSV → RGBA.

```
private Scalar converScalarHsv2Rgba(Scalar hsvColor) {
    Mat pointMatRgba = new Mat();
    Mat pointMatHsv = new Mat(1, 1, CvType.CV_8UC3, hsvColor);
    Imgproc.cvtColor(pointMatHsv, pointMatRgba, Imgproc.COLOR_HSV2RGB_FULL, 4);

    return new Scalar(pointMatRgba.get(0, 0));
}
```

Εικόνα 8.55 HSV→RGBA

Στην εικόνα 8.56 βλέπουμε τον κώδικα που κάνει διαχείριση των δεδομένων που λαμβάνει κατά τη σύνδεση του module με το κινητό. Στο κόκκινο πλαίσιο ακυρώνει το νήμα, το οποίο έφερε εις πέρας την σύνδεση. Στην συνέχεια ,στο πράσινο πλαίσιο κάνει ακύρωση οποιουδήποτε νήματος πρόσφατα έτρεξε μια σύνδεση. Στο μπλε πλαίσιο κάνει εκκίνηση του νήματος για να διαχειριστεί την σύνδεση και εκτελεί την μετάδοση. Στο μοβ πλαίσιο στέλνει το όνομα της συσκευής που συνδέθηκε με το Bluetooth πίσω στο activity.

```
// Management of BT connection through ConnectedThread
public void connected(BluetoothSocket socket, BluetoothDevice device) {

    // Cancel the thread that completed the connection
    if (mConnectThread != null) {
        mConnectThread.cancel();
        mConnectThread = null;
    }
    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
    }
    // Start the thread to manage the connection and perform transmissions
    mConnectedThread = new ConnectedThread(socket);
    mConnectedThread.start();
    // Send the name of the connected device back to the UI Activity
    Message msg = mHandler.obtainMessage(MESSAGE_DEVICE_NAME);
    Bundle bundle = new Bundle();
    bundle.putString(DEVICE_NAME, device.getName());
    msg.setData(bundle);
    mHandler.sendMessage(msg);
}
```

Εικόνα 8.56 Management of connection

Στην εικόνα 8.57 φτιάχνουμε την void stop, η οποία κάνει stop μόλις κληθεί σε όλα τα νήματα. Αυτά που έτρεχαν πριν μια διασύνδεση και αυτά που έτρεχαν τώρα.

```
// Stop all threads
public void stop() {

    if (mConnectThread != null) {
        mConnectThread.cancel();
        mConnectThread = null;
    }

    if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
    }
}
```

Εικόνα 8.57 Stop all threads

Στην εικόνα 8.58 βλέπουμε με ποιες εντολές εμφανίζουμε το μήνυμα “Unable to connect device”, όταν η σύνδεση με το Bluetooth χαθεί.

```
// Connection attempt failed
private void connectionFailed() {

    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(MESSAGE_FIELD);
    Bundle bundle = new Bundle();
    bundle.putString(FIELD, "Unable to connect device");
    msg.setData(bundle);
    mHandler.sendMessage(msg);
}
```

Εικόνα 8.58 Connection attempt Failed

Στην εικόνα 8.59 βλέπουμε την δημιουργία της void forward(). Αυτή η void είναι υπεύθυνη για την κίνηση του αυτοκινήτου προς τα εμπρός. Λειτουργεί βάζοντας στον πίνακα buffer στην πρώτη θέση την πόρτα που καταλαβαίνει το Arduino για να κινηθεί προς τα εμπρός. Ο ίδιος κώδικας είναι και για το backward(), left() ,right() και stopCar().

```
public static void forward() {
    byte[] buffer = new byte[1];
    buffer[0] = FORWARD;
    mConnectedThread.write(buffer);
}
```

Εικόνα 8.59 Forward

Στην εικόνα 8.60 βλέπουμε τον κώδικα στο οποίο το connectThread κληρονομεί τα thread. Στο κόκκινο πλαίσιο δηλώνουμε το socket & το device και παίρνουμε τιμές σε αυτά όταν πετύχουμε την επικοινωνία με το Bluetooth. Στο πράσινο πλαίσιο αρχικά ακυρώνει συνέχεια την αναζήτηση, για να μην έχουμε μια αργή σύνδεση. Στην συνέχεια, δημιουργεί μια σύνδεση με το Bluetooth και με την χρήση της εντολής try μας επιστρέφει μόνο όταν βρει κάποια σύνδεση. Αν δεν βρει, πάει συνεχώς στο exception και αναζητάει νέα σύνδεση. Στο μπλε πλαίσιο κάνει reset το connectThread και ξεκινάει την σύνδεση των νημάτων.

```
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    public ConnectThread(BluetoothDevice device) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
        }
        mmSocket = tmp;
    }
    public void run() {
        // Always cancel discovery because it will slow down a connection
        mBluetoothAdapter.cancelDiscovery();
        // Make a connection to the BluetoothSocket
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            mmSocket.connect();
        } catch (IOException e) {
            connectionFailed();
            // Close the socket
            try {
                mmSocket.close();
            } catch (IOException e2) { }
            return;
        }
        // Reset the ConnectThread-connection ok
        synchronized (ColorBlobDetectionActivity.this) {
            mConnectThread = null;
        }
        // Start the connected thread
        connected(mmSocket, mmDevice);
    }
```

Εικόνα 8.60 Connect Tread

Στην εικόνα 8.61 βλέπουμε να γίνεται η σύνδεση με το Bluetooth και με ποίες εντολές μπορούμε να πάρουμε τα input-output data και να τα διαχειριστούμε. Στο κόκκινο πλαίσιο είναι οι εντολές που χρειαζόμαστε, για να πάρουμε τα Input-output data. Στο πράσινο πλαίσιο παρατηρούμε ότι, για όσο είναι συνδεδεμένο με το Bluetooth(true), θα διαβάζει τα input και θα στέλνει τα λαμβανόμενα bytes στο activity.

```
// Manage connection with BT device - in and out data
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;
    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;
        // Get the BluetoothSocket input and output streams
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
        }
        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }
    public void run() {
        byte[] buffer = new byte[1024];
        int bytes;
        // Keep listening to the InputStream while connected
        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);
                // Send the obtained bytes to the UI Activity
                mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer).sendToTarget();
            } catch (IOException e) {
                connectionLost();
                break;
            }
        }
    }
```

Εικόνα 8.61 input-output data part 1

Στην εικόνα 8.62 είναι το δεύτερο μέρος των input-output data. Στο κόκκινο πλαίσιο στέλνουμε μέσα του output το μήνυμα, που θέλουμε να εμφανίσουμε. Στο πράσινο πλαίσιο πραγματοποιούμε την ακύρωση της σύνδεσης.

```
    // Write to the connected OutStream
    public void write(byte[] buffer) {
        try {
            mmOutStream.write(buffer);

            // Share the sent message back to the UI Activity
            mHandler.obtainMessage(MESSAGE_WRITE, buffer.length, -1, buffer).sendToTarget();
        } catch (IOException e) { }
    }

    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}
```

Εικόνα 8.62 input-output data part 2

Στην εικόνα 8.63 βλέπουμε την δημιουργία του Handler. Σκοπός της δημιουργίας ενός τέτοιου handler είναι να πάρει πίσω τις πληροφορίες που θέλουμε και να τις πετάξει στην οθόνη της εφαρμογής μας. Στο πράσινο πλαίσιο κάνει το Button, που υπάρχει για την σύνδεση με το Bluetooth, να αλλάξει χρώμα μόλις συνδεθεί και να γίνει πράσινο.

```
// The Handler that gets information back
private final Handler mHandler = handleMessage(msg) -> {
        switch (msg.what) {
            case MESSAGE_WRITE:
                break;
            case MESSAGE_READ:
                break;
            case MESSAGE_DEVICE_NAME:
                // save the connected device's name
               // txt1.setText(msg.getData().getString(DEVICE_NAME));
                int bgrnd = getResources().getColor(R.color.green);
                ibtn.setBackgroundColor(bgrnd);
                break;
            case MESSAGE_FIELD:
                Toast.makeText(ColorBlobDetectionActivity.this, msg.getData().getString(FIELD), Toast.LENGTH_SHORT).show();
                break;
        }
};
```

Εικόνα 8.63 Handler-button connect change color

Στην συνέχεια πάμε στην ColorBlobDetection.class. Τα κομμάτια κώδικα που θα αναλύσουμε θα είναι μέσα από αυτήν. Στην εικόνα 8.64 βλέπουμε το setColorRadius, το οποίο είναι το εύρος ελέγχου σε HSV χώρο.

```
public void setColorRadius(Scalar radius) {
    mColorRadius = radius;
}
```

Εικόνα 8.64 SetColorRadius

Στην εικόνα 8.65βλέπουμε την δημιουργία σε HSV χώρο των χρωμάτων. Παίρνει μέγιστη και ελάχιστη τιμή. Τοποθετεί στους δύο πίνακες με κόκκινο πλαίσιο από 4 στοιχεία ξεκινώντας από min,max και καταλήγοντας από 0-255. Επίσης, ορίζει το νέο φάσμα στο πράσινο πλαίσιο.

```
public void setHsvColor(Scalar hsvColor) {
    double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ? hsvColor.val[0]-mColorRadius.val[0] : 0;
    double maxH = (hsvColor.val[0]+mColorRadius.val[0] <= 255) ? hsvColor.val[0]+mColorRadius.val[0] : 255;

    mLowerBound.val[0] = minH;
    mUpperBound.val[0] = maxH;

    mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];
    mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];

    mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];
    mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];

    mLowerBound.val[3] = 0;
    mUpperBound.val[3] = 255;

    Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);

    for (int j = 0; j < maxH-minH; j++) {
        byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};
        spectrumHsv.put(0, j, tmp);
    }

    Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL, 4);
}
```

Εικόνα 8.65 setHSVColor

Στην εικόνα 8.66 βλέπουμε την δημιουργία void process. Στο πράσινο πλαίσιο βλέπουμε ότι, μικραίνουν τα frame της εικόνας σε διαίρεση με το 4 και αυτό γίνεται γιατί έτσι μειώνεται ο χρόνος που χρειάζεται η προεργασία, γιατί αντί να ελέγξουμε έναν τεράστιο πίνακα, ελέγχουμε το ¼ αυτού. Επομένως, γίνεται πιο γρήγορη. Στο κόκκινο πλαίσιο κάνουμε δημιουργία και εύρεση του κατάλογου των περιγραμμάτων. Στο μπλε πλαίσιο βρίσκει την max contour area.Στο μοβ πλαίσιο φιλτράρει Την contours περιοχή και δίνει νέες διαστάσεις στην εικόνα.

```
public void process(Mat rgbaImage) {
    Imgproc.pyrDown(rgbaImage, mPyrDownMat);
    Imgproc.pyrDown(mPyrDownMat, mPyrDownMat);

    Imgproc.cvtColor(mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);

    Core.inRange(mHsvMat, mLowerBound, mUpperBound, mMask);
    Imgproc.dilate(mMask, mDilatedMask, new Mat());
    List<MatOfPoint> contours = new ArrayList<>();

    Imgproc.findContours(mDilatedMask, contours, mHierarchy, Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
    ColorBlobDetectionActivity.stopCar();
    // Find max contour area
    double maxArea = 0;
    Iterator<MatOfPoint> each = contours.iterator();
    while (each.hasNext()) {
        MatOfPoint wrapper = each.next();
        double area = Imgproc.contourArea(wrapper);
        if (area > maxArea)
            maxArea = area;
    }
    // Filter contours by area and resize to fit the original image size
    mContours.clear();
    each = contours.iterator();
    while (each.hasNext()) {
        MatOfPoint contour = each.next();
        if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {
            Core.multiply(contour, new Scalar(4,4), contour);
            ColorBlobDetectionActivity.forward();
            mContours.add(contour);
        }
    }

}
```

Εικόνα 8.66 Process και διαίρεση της εικόνας δια 4

Στην συνέχεια θα επεξηγήσουμε τον κώδικα της κλάσης “DeviceSearchActivity.class”. Σε αυτήν την κλάση έχουμε ό,τι έχει σχέση με την επικοινωνία με το Module. Ας τα δούμε αναλυτικότερα. Στην εικόνα 8.67 βλέπουμε την onActivityResult. Στο πράσινο πλαίσιο έχουμε την case η οποία μας επιστρέφει την απάντηση για την αίτηση να ανοίξουν τα Bluetooth του κινητού. Αν μπει στο πρώτο if τότε σημαίνει ότι η αίτηση για άνοιγμα του Bluetooth είναι enable. Αν μπει στο else τότε δεν έγινε αποδεκτή η αίτηση για άνοιγμα του Bluetooth.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
    case REQUEST_ENABLE_BT:
        // When the request to enable Bluetooth returns
        if (resultCode == Activity.RESULT_OK) {
            // Bluetooth is now enabled, so discover BT devices
            discoverBT();
        } else {
            // User did not enable Bluetooth or an error occured
            Toast.makeText(this, "Bluetooth not enabled...", Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

Εικόνα 8.67 Request enable BT

Στην εικόνα 8.68 στο πράσινο πλαίσιο παίρνει μια πρόσφατη συσκευή που έχει ξανακάνει connect. Αν ήδη υπάρχουν παλιές συσκευές στην ArrayAdapter πρόσθεσε άλλη μια αλλιώς βγάλε μήνυμα “No paired Devices”(κόκκινο πλαίσιο).

```
public void discoverBT() {
    // Get a set of currently paired devices
    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
    // If there are paired devices, add each one to the ArrayAdapter
    if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        String noDevices = "No paired devices".toString();
        mPairedDevicesArrayAdapter.add(noDevices);
    }
}
```

Εικόνα 8.68 DiscoverBT

Στην εικόνα 8.69 κάνει το startScan. Στο πράσινο πλαίσιο αν έχει ήδη κάνει discovering τότε σταμάτα. Στο κόκκινο πλαίσιο εμφανίζει το “scanning for devices..” και δίνει ένα συγκεκριμένο χρώμα που του έχουμε ορίσει στον scan_background.xml το οποίο είναι στον φάκελο res/values.

```
public void startScan(View v) {
    v.setVisibility(View.GONE);
    // If we're already discovering, stop it
    if (mBluetoothAdapter.isDiscovering()) {
        mBluetoothAdapter.cancelDiscovery();
    }
    txt1.setText("scanning for devices...");
    txt1.setBackgroundResource(R.color.scan_background);
    // Request discover from BluetoothAdapter
    mBluetoothAdapter.startDiscovery();
}
```

Εικόνα 8.69 startScan

Στην εικόνα 8.70 βλέπουμε το on-click listener for all devices in the ListViews. Στο πράσινο πλαίσιο έχουμε την ακύρωση της αναζήτησης συσκευής για να συνδεθεί διότι, είναι πλέον η ώρα της αναζήτησης. Στο κόκκινο πλαίσιο παίρνει την mac address της συσκευής που έχει συνδεθεί η οποία είναι τα 17 τελευταία ψηφία του View.

```
// The on-click listener for all devices in the ListViews
private OnItemClickListener mDeviceClickListener = (av, v, arg2, arg3) -> {
        // Cancel discovery because it is time to connect
        mBluetoothAdapter.cancelDiscovery();

        // Get the device MAC address, which is the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);
        Intent intent = new Intent();
        intent.putExtra(ColorBlobDetectionActivity.DEVICE_ADDRESS, address);
        setResult(Activity.RESULT_OK, intent);
        finish();
};
```

Εικόνα 8.70 OnItemClickListener

Στην εικόνα 8.71 βλέπουμε δέχεται listensγια αναζήτηση συσκευών και αλλάζει τον τίτλο όταν η αναζήτηση φτάσει στο τέλος. Στο πράσινο πλαίσιο όταν η αναζήτηση βρει μια συσκευή τότε παίρνει το object από την μεταβλητή intent. Εάν όλα αυτά έχουν γίνει τότε τα αγνοούμε. Στο κόκκινο πλαίσιο όταν η αναζήτηση έχει τελειώσει τότε άλλαζουμε το activity title.

```
// The BroadcastReceiver that listens for discovered devices and
// changes the title when discovery is finished
public final BroadcastReceiver mReceiver = (context, intent) -> {
        String action = intent.getAction();

        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // If it's already paired, skip it, because it's been listed already
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        // When discovery is finished, change the Activity title
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            txt1.setBackgroundResource(R.color.select_background);
            txt1.setText("select a device to connect");
            if (mNewDevicesArrayAdapter.getCount() == 0) {
                String noDevices = "No devices found".toString();
                mNewDevicesArrayAdapter.add(noDevices);
            }
        }
};
```

Εικόνα 8.71 BroadcastReceiver

8.3.5 Επεξήγηση κώδικα Lunar3

Ο κώδικας του Lunar3 είναι περίπου ίδιος με τον κώδικα του Lunar1 αφού υπάρχουν μερικές σημαντικές αλλαγές. Αυτές θα δείξουμε παρακάτω. Στην εικόνα 8.72 υπάρχουν διαφορές. Είναι η κλάση του “ColorBlobDetectionActivity.class” η public Boolean onTouch. Πάμε να δούμε τις παρακάτω εικόνες και θα καταλάβετε. Στο πράσινο πλαίσιο παίρνουμε την ανάλυση της οθόνης. Στο κόκκινο πλαίσιο αν το αντικείμενο βρίσκεται εκτός ορίων της κάμερας ή δεν έχει συνδεθεί με το Bluetooth, επιστρέφει λάθος .Δηλαδή δεν κάνει τίποτα. Στο μπλε πλαίσιο δημιουργούμε την περιοχή που πατήθηκε πάνω στην οθόνη. Στο μοβ πλαίσιο μετατρέπει σε HSV περιοχή. Στο πράσινο το μέσο χρώμα από την περιοχή που πατήθηκε. Στο πράσινο πλαίσιο μετατρέπει την περιοχή από HSV→RGBA. Στο κόκκινο πλαίσιο κάνουμε resize του φάσματος και απελευθερώνουμε τους mats.

```
public boolean onTouch(View v, MotionEvent event) {
    int cols = mRgba.cols();  //get resolution on display(αναλυση)
    int rows = mRgba.rows();  //get resolution on display(αναλυση)
    int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;  //get resolution on display(αναλυση)
    int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;  //get resolution on display(αναλυση
    int x = (int)event.getX() - xOffset;  //get resolution on display(αναλυση)
    int y = (int)event.getY() - yOffset;  //get resolution on display(αναλυση)

    Log.i(TAG, "Touch image coordinates: (" + x + ", " + y + ")");

    if ((x < 0) || (y < 0) || (x > cols) || (y > rows) || (mConnectedThread == null)) return false

    Rect touchedRect = new Rect();
    touchedRect.x = (x>4) ? x-4 : 0;
    touchedRect.y = (y>4) ? y-4 : 0;
    touchedRect.width = (x+4 < cols) ? x + 4 - touchedRect.x : cols - touchedRect.x;
    touchedRect.height = (y+4 < rows) ? y + 4 - touchedRect.y : rows - touchedRect.y;
    // create a touched regionmat from the image created from the touches
    Mat touchedRegionRgba = mRgba.submat(touchedRect);
    //Convert the new mat to HSV colour space
    //μετατρεπει σε HSV color
    Mat touchedRegionHsv = new Mat();
    Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv, Imgproc.COLOR_RGB2HSV_FULL);
    // Calculate average color of touched region
    mBlobColorHsv = Core.sumElems(touchedRegionHsv);
    int pointCount = touchedRect.width*touchedRect.height;
    for (int i = 0; i < mBlobColorHsv.val.length; i++) {
        mBlobColorHsv.val[i] /= pointCount;
    }
    //converts scalar to hsv to RGB
    mBlobColorRgba = converScalarHsv2Rgba(mBlobColorHsv);//convert the fasma hsv->rgba
    Log.i(TAG, "Touched rgba color: (" + mBlobColorRgba.val[0] + ", " + mBlobColorRgba.val[1] +
            ", " + mBlobColorRgba.val[2] + ", " + mBlobColorRgba.val[3] + ")");
    mDetector.setHsvColor(mBlobColorHsv);
    // Resize the image to specture size
    Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
    mIsColorSelected = true;
    // Release all mats
    touchedRegionRgba.release();
    touchedRegionHsv.release();
    return false; // don't need subsequent touch events
}
```

Εικόνα 8.72 OnTouch

Στην εικόνα 8.73 βλέπουμε το onCameraFrame. Στο πράσινο πλαίσιο κάνει μετατροπή σε gray scale. Στο κόκκινο πλαίσιο φτιάχνει την δυαδική εικόνα και remove pixel στο erode με ένα νέο φάσμα 9X9. Στο μπλε πλαίσιο φτιάχνει ένα νέο παράθυρο. Στο μοβ πλαίσιο βρίσκουμε το contours και το maxContours.

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
    mRgba = inputFrame.rgba();
    // convert to Gray scale
    Imgproc.cvtColor(mRgba, img_gray, Imgproc.COLOR_BGR2GRAY);
    // make it binary with threshold=100
    Imgproc.threshold(img_gray, erd, 100, 255, Imgproc.THRESH_OTSU);
    // remove pixel noise by "erode" with structuring element of 9X9
    Mat erode = Imgproc.getStructuringElement(Imgproc.MORPH_ERODE, new Size(9, 9));
    Imgproc.erode(erd, tgt, erode);
    // apply "dilation" to enlarge object
    Mat dilate = Imgproc.getStructuringElement(Imgproc.MORPH_DILATE, new Size(9, 9));
    Imgproc.dilate(tgt, erd, dilate);
    //take a window
    Size s=erd.size();
    int W = (int) s.width;
    int H = (int) s.height;
    Rect r=new Rect(0,H/2-100, W,200);
    Mat mask= new Mat(s, CvType.CV_8UC1, new Scalar(0,0,0));
    rectangle(mask, r.tl(), r.br(),new Scalar(255, 255, 255),-1);
    erd.copyTo(window, mask);
    // find the contours
    Imgproc.findContours(window, contours, dest, 0, 2);
    // find largest contour
    int maxContour = -1;
    double area = 0;
    for (int i = 0; i<contours.size(); i++) {
        double contArea = Imgproc.contourArea(contours.get(i));
        if (contArea > area) {
            area = contArea;
            maxContour = i;
        }
    }
```

Εικόνα 8.73 OnCameraFrame Part 1

Στην εικόνα 8.74 βλέπουμε στο πράσινο πλαίσιο ψάχνουμε να βρούμε την θέση στο κέντρο της εικόνας. Στο κόκκινο πλαίσιο ελέγχουμε την κατεύθυνση της κίνησης και τις εντολές που δίνουμε για να πάει right, left, forward. Στο μπλε φαίνεται το stopmoving δηλαδή να σταματήσει να κινείται.

```
Rect rect = null;
if (maxContour > -1)
    rect = Imgproc.boundingRect(contours.get(maxContour));
//position to center
while (!train) {
    if (rect != null) {
        Imgproc.rectangle(mRgba, rect.tl(), rect.br(), new Scalar(255, 255, 255), 5);
        if ((rect.x + rect.width / 2) > W/2-20 && (rect.x + rect.width / 2)< W/2+20) {
            runOnUiThread(() -> {
                    Toast.makeText(getApplicationContext(), "OK", Toast.LENGTH_SHORT).show()
            });
            train = true;
        }
    }
    if (contours != null)
        contours.clear();
    return mRgba;

}
if (train) {
    if (rect != null) {
        Imgproc.rectangle(mRgba, rect.tl(), rect.br(), new Scalar(255, 255, 255), 5);
        // direction of movement
        int thr = 10;
        if ((rect.x + rect.width / 2) < (W / 2 - thr)) {
            // move to the RIGHT
            uHandler.obtainMessage(ColorBlobDetectionActivity.RIGHT).sendToTarget();
        } else {
            if ((rect.x + rect.width / 2) > (W / 2 + thr)) {
                uHandler.obtainMessage(ColorBlobDetectionActivity.LEFT).sendToTarget();
            } else {
                uHandler.obtainMessage(ColorBlobDetectionActivity.FORWARD).sendToTarget();
            }
        }
    }
    else {
        // stop moving
        uHandler.obtainMessage(ColorBlobDetectionActivity.STOP).sendToTarget();
    }
}
if (contours != null)
    contours.clear();
return mRgba;
```

Εικόνα 8.74 OnCameraFrame Part 2

ΚΕΦΑΛΑΙΟ 9 - ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

Η απόφαση για την δημιουργία μιας τέτοιας εργασίας πάρθηκε μετά από πολύ σκέψη διότι προσπαθούσαμε να βρούμε μια καινοτόμα ιδέα, η οποία μπορεί να αναπτυχθεί περεταίρω και να χρησιμοποιηθεί στην καθημερινότητα μας με αποτέλεσμα να κάνει την ζωή μας πιο εύκολη.

Σκεφτήκαμε, πως μια τέτοια καινοτόμα ιδέα μπορεί να βρει ανταπόκριση στις μέρες μας.

Ένα αυτόνομο αμαξίδιο αν αναπτυχθεί περεταίρω μπορεί να χρησιμοποιηθεί ακόμα και σε τομείς εξερεύνησης ενός νέου πλανήτη καθώς αν εντοπίσουμε κάτι νέο που δεν έχουμε ξαναδεί, το αυτοκινητάκι μπορεί πολύ εύκολα να το ακολουθήσει και να καταγράψει όλες του τις κινήσεις.

Ένας άλλος τρόπος χρήσης είναι η προσαρμογή της εφαρμογής μας πάνω σε ένα αυτοκίνητο το οποίο κινείται μόνο του, χωρίς οδηγό. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι το λεωφορείο που έκανε την πρώτη εμφάνισή του στην Ελλάδα, στα Τρίκαλα και εκτέλεση μιας συγκεκριμένης διαδρομής χωρίς οδηγό αποφεύγοντας, μάλιστα, τα εμπόδια με χρήση συγκεκριμένης ρύθμισης. (εικόνα 9.1).

Εικόνα 9.1 Λεωφορείο χωρίς οδηγό

Όπως παρατηρούμε είναι μια ωραία και καινοτόμα ιδέα από την οποία μπορούμε να δημιουργήσουμε κάτι νέο εμπεριέχοντας όμως και δικά μας στοιχεία.

ΠΑΡΑΡΤΗΜΑ Α': ΚΩΔΙΚΑΣ ANDROID LUNAR1

"ColorBlobDetectionActivity.class"

```
package org.opencv.samples.colorblobdetect;

import android.app.Activity;

public class ColorBlobDetectionActivity extends Activity implements OnTouchListener,
CvCameraViewListener2 {

    private static final String  TAG              = "OCVSample::Activity";

    private ImageButton ibtn;

    private ImageView cross;

    //private TextView txt1;

    private boolean              mIsColorSelected = false;

    private Mat                  mRgba;

    private Scalar               mBlobColorRgba;

    private Scalar               mBlobColorHsv;

    private ColorBlobDetector    mDetector;

    private Mat                  mSpectrum;

    private Size                 SPECTRUM_SIZE;

    private Scalar               CONTOUR_COLOR;

    private CameraBridgeViewBase mOpenCvCameraView;

    private static final byte FORWARD = 0x5;

    private static final byte BACKWARD = 0x6;

    private static final byte LEFT = 0x2;

    private static final byte RIGHT = 0x3;

    private static final byte STOP = 0x9;

    // return Intent extra

    public static String DEVICE_ADDRESS = "device_address";

    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    private ConnectThread mConnectThread;

    private static ConnectedThread mConnectedThread;

    // Intent request codes

    private static final int REQUEST_CONNECT_DEVICE=1;
```

```
    private BluetoothAdapter mBluetoothAdapter=null;
    // Messages sent to Handler from Threads
    public static final int MESSAGE_DEVICE_NAME = 2;
    public static final int MESSAGE_FIELD = 3;
    public static final int MESSAGE_READ = 4;
    public static final int MESSAGE_WRITE = 5;
    // Key names to Handler from Threads
    public static final String DEVICE_NAME = "device_name";
    public static final String FIELD = "field";
    public static final String READ = "read";
    public static final String WRITE = "write";
    private BaseLoaderCallback  mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS:
                {
                    Log.i(TAG, "OpenCV loaded successfully");
                    mOpenCvCameraView.enableView();
mOpenCvCameraView.setOnTouchListener(ColorBlobDetectionActivity.this);
                } break;
                default:
                {
                        super.onManagerConnected(status);
                } break;
            }
        }
    };

    public ColorBlobDetectionActivity() {
        Log.i(TAG, "Instantiated new " + this.getClass());
    }
```

```
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        setContentView(R.layout.color_blob_detection_surface_view);
        ibtn=(ImageButton) findViewById(R.id.imageButton1);
        cross=(ImageView) findViewById(R.id.imageView);
        cross.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (mConnectedThread == null) {
                    finish();
                    System.exit(0);
                }
                else {
                    stopCar();
                    finish();
                    System.exit(0);
                }
            }
        });
        // txt1=(TextView) findViewById(R.id.textView1);
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.color_blob_detection_activity_surface_view);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }
    @Override
    public void onPause()
```

```
    {
        super.onPause();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }
    public void search(View view) {
        Intent intent = new Intent(this,DeviceSearchActivity.class);
        startActivityForResult(intent, REQUEST_CONNECT_DEVICE);
    }
    @Override
    public void onResume()
    {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
            Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0, this, mLoaderCallback);
        } else {
            Log.d(TAG, "OpenCV library found inside package. Using it!");
mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
    }
    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }
    public void onCameraViewStarted(int width, int height) {
        mRgba = new Mat(height, width, CvType.CV_8UC4);
        mDetector = new ColorBlobDetector();
        mSpectrum = new Mat();
        mBlobColorRgba = new Scalar(255);
        mBlobColorHsv = new Scalar(255);
        SPECTRUM_SIZE = new Size(200, 64);
```

```
        CONTOUR_COLOR = new Scalar(255,0,0,255);
    }
    public void onCameraViewStopped() {
        mRgba.release();
    }
    //when a motion event happens (someone touches the device)
    public boolean onTouch(View v, MotionEvent event) {
        int cols = mRgba.cols();  //get resolution on display(αναλυση)
        int rows = mRgba.rows();  //get resolution on display(αναλυση)
        int xOffset = (mOpenCvCameraView.getWidth() - cols) / 2;  //get resolution on display(αναλυση)
        int yOffset = (mOpenCvCameraView.getHeight() - rows) / 2;  //get resolution on display(αναλυση)
        int x = (int)event.getX() - xOffset;  //get resolution on display(αναλυση)
        int y = (int)event.getY() - yOffset;  //get resolution on display(αναλυση)
        Log.i(TAG, "Touch image coordinates: (" + x + ", " + y + ")");
        if ((x < 0) || (y < 0) || (x > cols) || (y > rows) || (mConnectedThread == null)) return false;
        Rect touchedRect = new Rect();
        touchedRect.x = (x>4) ? x-4 : 0;
        touchedRect.y = (y>4) ? y-4 : 0;
        touchedRect.width = (x+4 < cols) ? x + 4 - touchedRect.x : cols - touchedRect.x;
        touchedRect.height = (y+4 < rows) ? y + 4 - touchedRect.y : rows - touchedRect.y;
        // create a touched regionmat from the image created from the touches
        Mat touchedRegionRgba = mRgba.submat(touchedRect);
        //Convert the new mat to HSV colour space
        //μετατρεπει σε HSV color
        Mat touchedRegionHsv = new Mat();
        Imgproc.cvtColor(touchedRegionRgba, touchedRegionHsv, Imgproc.COLOR_RGB2HSV_FULL);
        // Calculate average color of touched region
        mBlobColorHsv = Core.sumElems(touchedRegionHsv);
        int pointCount = touchedRect.width*touchedRect.height;
        for (int i = 0; i < mBlobColorHsv.val.length; i++) {
            mBlobColorHsv.val[i] /= pointCount;
        }
```

```
        //converts scalar to hsv to RGB
        mBlobColorRgba = converScalarHsv2Rgba(mBlobColorHsv);//convert the fasma hsv->rgba
        Log.i(TAG, "Touched rgba color: (" + mBlobColorRgba.val[0] + ", " + mBlobColorRgba.val[1] +
                ", " + mBlobColorRgba.val[2] + ", " + mBlobColorRgba.val[3] + ")");
        mDetector.setHsvColor(mBlobColorHsv);
        // Resize the image to specture size
        Imgproc.resize(mDetector.getSpectrum(), mSpectrum, SPECTRUM_SIZE);
        mIsColorSelected = true;
        // Release all mats
        touchedRegionRgba.release();
        touchedRegionHsv.release();
        return false; // don't need subsequent touch events
    }
    //Σε γενικές γραμμές αυτός ο κώδικας παίρνει μια εικόνα από τις συντεταγμένες που σχεδιάζονται με ένα άγγιγμα απο τον χρήστη. Στη συνέχεια μετατρέπει το χρωματικό χώρο και αλλάζει το μέγεθος με το μέγεθος του φάσματος
    public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
        mRgba = inputFrame.rgba();
        if (mIsColorSelected) {
            mDetector.process(mRgba);
            List<MatOfPoint> contours = mDetector.getContours();
            Log.e(TAG, "Contours count: " + contours.size());
            Imgproc.drawContours(mRgba, contours, -1, CONTOUR_COLOR);
            Mat colorLabel = mRgba.submat(4, 68, 4, 68);
            colorLabel.setTo(mBlobColorRgba);
            Mat spectrumLabel = mRgba.submat(4, 4 + mSpectrum.rows(), 70, 70 + mSpectrum.cols());
            mSpectrum.copyTo(spectrumLabel);
        }
        return mRgba;
    }
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode) {
            case REQUEST_CONNECT_DEVICE:
```

```
            if (resultCode == Activity.RESULT_OK) {
                // Get the device MAC address
                String address = data.getExtras().getString(DEVICE_ADDRESS);
                // Get the BLuetoothDevice object
                BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
                // Cancel any thread currently running a connection
                if (mConnectedThread != null) {
                    mConnectedThread.cancel();
                    mConnectedThread = null;
                }
                // Attempt to connect to the device
                mConnectThread = new ConnectThread(device);
                mConnectThread.start();
            }
        }
    }
    //final conversion(τελευταια μετατροπη)
    private Scalar converScalarHsv2Rgba(Scalar hsvColor) {
        Mat pointMatRgba = new Mat();
        Mat pointMatHsv = new Mat(1, 1, CvType.CV_8UC3, hsvColor);
        Imgproc.cvtColor(pointMatHsv, pointMatRgba, Imgproc.COLOR_HSV2RGB_FULL, 4);
        return new Scalar(pointMatRgba.get(0, 0));
    }
    // Management of BT connection through ConnectedThread
    public void connected(BluetoothSocket socket, BluetoothDevice device) {
        // Cancel the thread that completed the connection
        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }
        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {
```

```
            mConnectedThread.cancel();
            mConnectedThread = null;
        }
        // Start the thread to manage the connection and perform transmissions
        mConnectedThread = new ConnectedThread(socket);
        mConnectedThread.start();
        // Send the name of the connected device back to the UI Activity
        Message msg = mHandler.obtainMessage(MESSAGE_DEVICE_NAME);
        Bundle bundle = new Bundle();
        bundle.putString(DEVICE_NAME, device.getName());
        msg.setData(bundle);
        mHandler.sendMessage(msg);
    }
    // Stop all threads
    public void stop() {
        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }
        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }
    }
    // Connection attempt failed
    private void connectionFailed() {
        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(MESSAGE_FIELD);
        Bundle bundle = new Bundle();
        bundle.putString(FIELD, "Unable to connect device");
        msg.setData(bundle);
        mHandler.sendMessage(msg);
```

```
    }
    public static void forward() {
        byte[] buffer = new byte[1];
        buffer[0] = FORWARD;
        mConnectedThread.write(buffer);
    }
    public static void backward() {
        byte[] buffer = new byte[1];
        buffer[0] = BACKWARD;
        mConnectedThread.write(buffer);
    }
    public static void left() {
        byte[] buffer = new byte[1];
        buffer[0] = LEFT;
        mConnectedThread.write(buffer);
    }
    public static void right() {
        byte[] buffer = new byte[1];
        buffer[0] = RIGHT;
        mConnectedThread.write(buffer);
    }
    public static void stopCar() {
        byte[] buffer = new byte[1];
        buffer[0] = STOP;
        mConnectedThread.write(buffer);
    }
    // Connection was lost
    private void connectionLost() {
        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(MESSAGE_FIELD);
        Bundle bundle = new Bundle();
        bundle.putString(FIELD, "Device connection was lost");
```

```
        msg.setData(bundle);
        mHandler.sendMessage(msg);
    }
    // Thread for connection with remote BT device
    private class ConnectThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final BluetoothDevice mmDevice;
        public ConnectThread(BluetoothDevice device) {
            mmDevice = device;
            BluetoothSocket tmp = null;
            // Get a BluetoothSocket for a connection with the
            // given BluetoothDevice
            try {
                tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
            } catch (IOException e) {
            }
            mmSocket = tmp;
        }
        public void run() {
            // Always cancel discovery because it will slow down a connection
            mBluetoothAdapter.cancelDiscovery();
            // Make a connection to the BluetoothSocket
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                    mmSocket.connect();
            } catch (IOException e) {
                connectionFailed();
                // Close the socket
                try {
                    mmSocket.close();
                } catch (IOException e2) { }
```

```
            return;
        }
        // Reset the ConnectThread-connection ok
        synchronized (ColorBlobDetectionActivity.this) {
            mConnectThread = null;
        }
        // Start the connected thread
        connected(mmSocket, mmDevice);
    }
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
        }
    }
}
// Manage connection with BT device - in and out data
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;
    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;
        // Get the BluetoothSocket input and output streams
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
        }
        mmInStream = tmpIn;
```

```
        mmOutStream = tmpOut;
    }
    public void run() {
        byte[] buffer = new byte[1024];
        int bytes;
        // Keep listening to the InputStream while connected
        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);
                // Send the obtained bytes to the UI Activity
                mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer).sendToTarget();
            } catch (IOException e) {
                connectionLost();
                break;
            }
        }
    }
    // Write to the connected OutStream
    public void write(byte[] buffer) {
        try {
            mmOutStream.write(buffer);
            // Share the sent message back to the UI Activity
            mHandler.obtainMessage(MESSAGE_WRITE, buffer.length, -1, buffer).sendToTarget();
        } catch (IOException e) { }
    }
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}
```

```
    // The Handler that gets information back
    private final Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MESSAGE_WRITE:
                    break;
                case MESSAGE_READ:
                    break;
                case MESSAGE_DEVICE_NAME:
                    // save the connected device's name
                    // txt1.setText(msg.getData().getString(DEVICE_NAME));
                    int bgrnd = getResources().getColor(R.color.green);
                    ibtn.setBackgroundColor(bgrnd);
                    break;
                case MESSAGE_FIELD:
                    Toast.makeText(ColorBlobDetectionActivity.this, msg.getData().getString(FIELD),
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
    private void doInfo() {
        new AlertDialog.Builder(this).
                setTitle( getString(R.string.title_info)).
                setMessage(" Constructor ==> ΗΛΙΑΣ ΓΑΛΑΝΟΠΟΥΛΟΣ / ΧΡΗΣΤΟΣ ΚΑΤΣΑΡΗΣ\n"
                        + " Institution ==> TEI of Piraeus\n"
                        + " Laboratory ==> Applied Information Systems\n"
                        + " Project Name ==> ColorBlobDetector\n"
                        + " Application ==> Follow the target\n").
                show();
    }
```

}

“ColorBlobDetection.class”

```
package org.opencv.samples.colorblobdetect;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;

public class ColorBlobDetector {
    // Lower and Upper bounds for range checking in HSV color space
    private Scalar mLowerBound = new Scalar(0);
    private Scalar mUpperBound = new Scalar(0);
    // Minimum contour area in percent for contours filtering
    private static double mMinContourArea = 0.1;
    // Color radius for range checking in HSV color space
    private Scalar mColorRadius = new Scalar(25,50,50,0);
    private Mat mSpectrum = new Mat();
    private List<MatOfPoint> mContours = new ArrayList<MatOfPoint>();
    // Cache
    Mat mPyrDownMat = new Mat();
    Mat mHsvMat = new Mat();
    Mat mMask = new Mat();
    Mat mDilatedMask = new Mat();
```

```
    Mat mHierarchy = new Mat();

    public void setColorRadius(Scalar radius) {

        mColorRadius = radius;

    }

    public void setHsvColor(Scalar hsvColor) {

        double minH = (hsvColor.val[0] >= mColorRadius.val[0]) ? hsvColor.val[0]-mColorRadius.val[0] : 0;

        double maxH = (hsvColor.val[0]+mColorRadius.val[0] <= 255) ? hsvColor.val[0]+mColorRadius.val[0] :
255;

        mLowerBound.val[0] = minH;

        mUpperBound.val[0] = maxH;

        mLowerBound.val[1] = hsvColor.val[1] - mColorRadius.val[1];

        mUpperBound.val[1] = hsvColor.val[1] + mColorRadius.val[1];

        mLowerBound.val[2] = hsvColor.val[2] - mColorRadius.val[2];

        mUpperBound.val[2] = hsvColor.val[2] + mColorRadius.val[2];

        mLowerBound.val[3] = 0;

        mUpperBound.val[3] = 255;

        Mat spectrumHsv = new Mat(1, (int)(maxH-minH), CvType.CV_8UC3);


        for (int j = 0; j < maxH-minH; j++) {

            byte[] tmp = {(byte)(minH+j), (byte)255, (byte)255};

            spectrumHsv.put(0, j, tmp);

        }

        Imgproc.cvtColor(spectrumHsv, mSpectrum, Imgproc.COLOR_HSV2RGB_FULL, 4);

    }

    public Mat getSpectrum() {

        return mSpectrum;

    }

    public void setMinContourArea(double area) {

        mMinContourArea = area;

    }

    public void process(Mat rgbaImage) {

        Imgproc.pyrDown(rgbaImage, mPyrDownMat);  //divide length and height by 2
```

```
        Imgproc.pyrDown(mPyrDownMat, mPyrDownMat); //divide length and height by 2

        //In these lines the frame length and height is divided by four, this will reduce the process time needed.

        // To put the contour back on the original frame they have to resize length and height by 4


        Imgproc.cvtColor(mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);

        Core.inRange(mHsvMat, mLowerBound, mUpperBound, mMask);

        Imgproc.dilate(mMask, mDilatedMask, new Mat());

        // katalogos perigramatwn

        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();

        // eyresh perigrammatos

        Imgproc.findContours(mDilatedMask, contours, mHierarchy, Imgproc.RETR_EXTERNAL,
Imgproc.CHAIN_APPROX_SIMPLE);

        ColorBlobDetectionActivity.stopCar();

        // Find max contour area

        double maxArea = 0;

        Iterator<MatOfPoint> each = contours.iterator();

        while (each.hasNext()) {

            MatOfPoint wrapper = each.next();

            double area = Imgproc.contourArea(wrapper);

            if (area > maxArea)

                maxArea = area;

        }

        // Filter contours by area and resize to fit the original image size

        mContours.clear();

        each = contours.iterator();

        while (each.hasNext()) {

            MatOfPoint contour = each.next();

            if (Imgproc.contourArea(contour) > mMinContourArea*maxArea) {

                Core.multiply(contour, new Scalar(4,4), contour);

                //(περιγραμα,νεο φασμα,περιγραμμα)

                ColorBlobDetectionActivity.forward();

                mContours.add(contour);
```

```
            }
        }
    }
    public List<MatOfPoint> getContours() {
        return mContours;
    }
}
```

“DeviceSearchActivity.class”

```
package org.opencv.samples.colorblobdetect;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
```

```
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

public class DeviceSearchActivity extends Activity {
        TextView txt1, txt2, txt3;
    // Return Intent extra
        private static final int REQUEST_ENABLE_BT = 1;
        private ArrayAdapter<String> mPairedDevicesArrayAdapter;
        private ArrayAdapter<String> mNewDevicesArrayAdapter;
        private BluetoothAdapter mBluetoothAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_search);
        txt1 = (TextView) findViewById(R.id.textView1);
        txt2 = (TextView) findViewById(R.id.textView2);
        txt3 = (TextView) findViewById(R.id.textView3);
        // Initialize array adapters: One for already paired devices and one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);
        // Find and set up the ListView for newly discovered devices
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
```

```
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);
        // Register for broadcasts when a device is discovered
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);
        // Register for broadcasts when discovery has finished
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);
        // Get local Bluetooth adapter
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        // If adapter is null, then Bluetooth is not supported
        if (mBluetoothAdapter == null) {
            Toast.makeText(getBaseContext(), "Bluetooth not supported...", Toast.LENGTH_LONG).show();
           finish();
           return;
        }
    }
            @Override
            protected void onStart() {
                        super.onStart();
                        // If BT is not ON, request to enable it
        // setupCommand() will then be called during onActivityResult
                        if (!mBluetoothAdapter.isEnabled()) {
                                    Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                                    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
                        }
                        else {
                          discoverBT();
                        }
            }
            @Override
```

```
        protected void onResume() {
                super.onResume();
        }

        @Override
        protected void onDestroy() {
                super.onDestroy();
    // Make sure we're not doing discovery anymore
    if (mBluetoothAdapter != null) {
        mBluetoothAdapter.cancelDiscovery();
    }
    // Unregister broadcast listeners
    this.unregisterReceiver(mReceiver);
        }
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
    case REQUEST_ENABLE_BT:
      // When the request to enable Bluetooth returns
      if (resultCode == Activity.RESULT_OK) {
        // Bluetooth is now enabled, so discover BT devices
        discoverBT();
      } else {
        // User did not enable Bluetooth or an error occured
        Toast.makeText(this, "Bluetooth not enabled...", Toast.LENGTH_SHORT).show();
        finish();
      }
    }
  }
  public void discoverBT() {
                // Get a set of currently paired devices
    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
```

```
        // If there are paired devices, add each one to the ArrayAdapter
        if (pairedDevices.size() > 0) {
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        } else {
              String noDevices = getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }
    public void startScan(View v) {
              v.setVisibility(View.GONE);
        // If we're already discovering, stop it
        if (mBluetoothAdapter.isDiscovering()) {
              mBluetoothAdapter.cancelDiscovery();
        }
        txt1.setText(R.string.scanning);
        txt1.setBackgroundResource(R.color.scan_background);
        // Request discover from BluetoothAdapter
        mBluetoothAdapter.startDiscovery();
    }
    // The on-click listener for all devices in the ListViews
    private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
            // Cancel discovery because it is time to connect
              mBluetoothAdapter.cancelDiscovery();
            // Get the device MAC address, which is the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);
              Intent intent = new Intent();
              intent.putExtra(ColorBlobDetectionActivity.DEVICE_ADDRESS, address);
              setResult(Activity.RESULT_OK, intent);
```

```
            finish();
        }
    };
    // The BroadcastReceiver that listens for discovered devices and
    // changes the title when discovery is finished
    public final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            // When discovery finds a device
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // If it's already paired, skip it, because it's been listed already
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
                }
            // When discovery is finished, change the Activity title
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
              txt1.setBackgroundResource(R.color.select_background);
              txt1.setText(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
                    String noDevices = getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };
}
```

“Activity_device_search.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/select_device"
        android:background="@color/select_background"
        android:textColor="@color/white"
        android:paddingLeft="5dp" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:background="@color/select_background"
        android:paddingLeft="5dp"
        android:text="@string/title_paired_devices"
        android:textColor="@color/white" />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1" />
    <TextView android:id="@+id/textView3"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:background="@color/select_background"
        android:textColor="@color/white"
        android:paddingLeft="5dp" />
    <ListView android:id="@+id/new_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2" />
    <Button android:id="@+id/button_scan"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
        android:onClick="startScan" />
</LinearLayout>
```

“color_blob_detection_surface_view.xml”

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <org.opencv.android.JavaCameraView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/color_blob_detection_activity_surface_view" />
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <ImageButton
```

```
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/red"
        android:onClick="search"
        android:src="@drawable/bt"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true" />
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/close"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true" />
  </RelativeLayout>
</RelativeLayout>
```

“device_name.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textSize="18sp"
  android:padding="5dp"/>
```

ΠΑΡΑΡΤΗΜΑ Β': ΚΩΔΙΚΑΣ ARDUINO

```
#include <AFMotor.h>
#define FOR 0x5
#define BAC 0x6
#define LEF 0x2
#define RIG 0x3
#define STO 0x9
AF_DCMotor motor1(1, MOTOR12_64KHZ);
AF_DCMotor motor2(2, MOTOR12_64KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
void setup() {
  Serial.begin(57600);
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
  Serial.flush();
}
void loop() {
  Serial.flush();
  // ***Receiving***
  if (Serial.available()>0){
    int command = Serial.read();
    Serial.println(command);
    switch (command) {
      case FOR: {
```

```
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
      } break;
      case BAC: {
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
      } break;
      case LEF: {
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
      } break;
      case RIG: {
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
      } break;
      case STO: {
        motor1.run(RELEASE);
        motor2.run(RELEASE);
        motor3.run(RELEASE);
        motor4.run(RELEASE);
      } break;
    }
  }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) “Εισαγωγή στον προγραμματισμό Android”, Ι. Έλληνας – Ν. Έλληνας, Εκδόσεις Τζιόλα
2) https://www.android.com/history/
3) https://en.wikipedia.org/wiki/Android_(operating_system)
4) https://en.wikipedia.org/wiki/Android_version_history
5) http://techmaniacs.gr/android-cupcake-to-lollipop/
6) http://www.techgear.gr/android-6-0-marshmallow-micro-site-101905/
7) https://el.wikipedia.org/wiki/Arduino
8) https://www.arduino.cc/
9) http://osarena.net/android/aguides/mathenontas-to-android-androidmanifest-xml-is-all-about-mathimata-4-5-6.html
10) http://insights.dice.com/2014/03/19/googles-android-studio-vs-eclipse-fits-needs/
11) http://docs.appcelerator.com/platform/latest/#!/guide/Maintaining_a_Custom_AndroidManifest.xml
12) http://www.kassapoglou.com/category/programming/java/
13) https://developer.android.com/intl/vi/index.html
14) http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/
15) http://l.facebook.com/l.php?u=http%3A%2F%2Fdocs.nvidia.com%2Fgameworks%2Fcontent%2Ftechnologies%2Fmobile%2Fopencv_tutorial_camera_preview.htm&h=8AQHc_JqU&s=1