ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κίνηση δίποδου ρομπότ Arduino μέσω εφαρμογής Android

Γώγος Σταμάτης

Παυλίδης Ηλίας

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κίνηση Δίποδου Ρομπότ με Arduino και Εφαρμογή Android

Γώγος Σταμάτης
Α.Μ. 42723

Παυλίδης Ηλίας
Α.Μ. 43207

Εισηγητής:

Δρ. Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Δρ. Ιωάννης Έλληνας, Καθηγητής
Ιωάννης Αμοργίνος, Λέκτορας
Αναστασία Βελώνη, Καθ. Εφαρμογών

Ημερομηνία εξέτασης:

__/06/2020

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονη προσπάθεια από τους φοιτητές Παυλίδη Ηλία και Γώγο Σταμάτη. Θα θέλαμε να ευχαριστήσουμε τον καθηγητή Δρ. Ιωάννη Έλληνα που ανέλαβε την εισήγηση και επίβλεψη της πτυχιακής μας εργασίας και την μεγάλη υπομονή που έκανε καθ'όλη τη διάρκεια αυτής. Επίσης θα θέλαμε να ευχαριστήσουμε ξεχωριστά τις οικογένειες μας που μας πίεσαν να ολοκληρώσουμε τις σπουδές μας το συντομότερο δυνατό.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με το πώς ένας χρήστης μπορεί χωρίς ιδιαίτερες γνώσεις να καταφέρει να χειρίζεται (απλά) ένα δίποδο ρομπότ μέσω της κινητής συσκευής του, αρκεί να είναι smartphone. Πιο συγκεκριμένα με τη βοήθεια ενός αρντουίνο (Arduino), μιας εφαρμογής Android σε γλώσσα Java για τη συσκευή, και ενός module bluetooth για την επικοινωνία των δύο, καταφέρνουμε να κινήσουμε το ρομπότ μας τη στιγμή που επιθυμούμε.

The present thesis covers how a user can control a biped robot from their smartphone without the need for serious technical knowledge on the subject. Specifically, with the help of an arduino, a Java android application and a bluetooth module to connect the two of them, the control of a biped is accomplished.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ρομποτική, Μηχατρονική, Μικροελεγκτές, Προγραμματισμός.
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Arduino UNO, Biped robot, Android application, Bluetooth.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Η ΠΛΑΤΦΟΡΜΑ ARDUINO....11
1.1. Γενικές πληροφορίες για το Arduino....11
1.1.1. Υλικό....13
1.1.2. Shields....13
1.2. Εισαγωγή στο Arduino UNO....15
1.2.1. Χαρακτηριστικά Arduino Uno....15
1.2.2. Ο μικροελεγκτής ATMega328p....16
1.2.3. Ακροδέκτες Arduino Uno....16
1.2.4. Τροφοδοσία....18
1.2.5. Μνήμη....19
1.3. Το περιβάλλον ανάπτυξής Arduino IDE....20
1.3.1. Προγραμματισμός Arduino....22
1.3.2. Ρυθμίσεις....25
2. ΤΟ ΛΟΓΙΣΜΙΚΟ ANDROID....27
2.1. Εισαγωγή στο λογισμικό Android....27
2.2. Βασικά χαρακτηριστικά και πλεονεκτήματα....28
2.3. Οι εκδόσεις Android....29
2.4. Αρχιτεκτονική Android....31
2.4.1. Επίπεδο εφαρμογών (Applications)....32
2.4.2. Επίπεδο πλαισίου εφαρμογών (Application Framework) 32
2.4.3. Επίπεδο βιβλιοθηκών....33
2.4.4. Επίπεδο χρόνου εκτέλεσης (Android Runtime)....34
2.4.5. Επίπεδο πυρήνα Linux (Linux Kernel)....34
2.5. Οι εφαρμογές Android....35
2.5.1. Το αρχείο AndroidManifest.xml....37
2.5.2. Οι φάκελοι src & res....37
2.5.3. Κύκλος ζωής μιας εφαρμογής Android....38
2.6. Η ασφάλεια στο Android....40
2.7. Το Android SDK....41

2.8. Εισαγωγή στο Android Studio....42
2.8.1. Το περιβάλλον Android Studio....43
3. ΡΟΜΠΟΤ ΚΑΙ ΡΟΜΠΟΤΙΚΗ....55
3.1. Εισαγωγή στη ρομποτική....55
3.2. Ιστορία του ρομπότ και της ρομποτικής....56
3.3. Ο ορισμός του ρομπότ....57
3.3.1. Τα βασικά χαρακτηριστικά ενός ρομπότ....57
3.3.2. Κατηγορίες και είδη ρομπότ....58
3.4. Το δίποδο ρομπότ....63
3.5. Ρομπότ στη κοινωνία....63
4. ΤΟ BLUETOOTH....65
4.1. Εισαγωγή στην επικοινωνία bluetooth....65
4.1.1. Τρόπος λειτουργίας Bluetooth....65
4.1.2. Η δομή του Bluetooth....66
4.2. Το Bluetooth στις Android εφαρμογές....66
5. ΚΩΔΙΚΑΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ....67
5.1. Γενική λειτουργία και εισαγωγή στο project....67
5.2. Κώδικας Arduino....68
5.3. Κώδικας Android....74
5.3.1. Αρχείο activity_main.xml....74
5.3.2. Αρχείο AndroidManifest.xml....75
5.3.3. Αρχείο MainActivity.java....76
5.4. Επικοινωνία Android & Arduino με Bluetooth module HC-06....79
6. ΒΙΒΛΙΟΓΡΑΦΙΑ....81

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1.1: Γνωστές πλακέτες Arduino.....12
Πίνακας 1.2: Διάφορα Arduino Shields.....14
Πίνακας 1.3: Λειτουργίες ανάπτυξής Arduino IDE.....22
Πίνακας 2.1: Η αρχιτεκτονική του Android.....31

1. Η ΠΛΑΤΦΟΡΜΑ ARDUINO

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοικτού (open source) κώδικα και σχεδιασμού που ξεκίνησε το 2005 στην Ιταλία και βασίζεται σε ένα ευέλικτο και εύκολο στη χρήση λογισμικό και υλικό. Προορίζεται για οποιαδήποτε μορφή πρότζεκτ ηλεκτρονικών κυκλωμάτων και ιδίως για καλλιτέχνες, σχεδιαστές, την υλοποίηση χόμπι και δραστηριοτήτων, και γενικότερα για οποιονδήποτε ενδιαφέρεται να δημιουργήσει αλληλεπιδραστικά αντικείμενα ή περιβάλλοντα. Αυτό που έκανε το Arduino να αναδειχθεί και να ξεχωρίσει είναι πως όλο το κύκλωμα της πλακέτας διατίθεται με άδεια χρήσης Creative Commons, δηλαδή είναι “open source”, το οποίο σημαίνει πως ο καθένας μπορεί να κατασκευάσει και να σχεδιάσει τη δική του πλακέτα, όπως και εαν αυτός επιθυμεί.

1.1 Γενικές πληροφορίες για το Arduino

Μία πλακέτα Arduino αποτελείται από ένα κύκλωμα που χρησιμοποιεί μικροελεγκτή και διαθέτει έναν αριθμό εισόδων και εξόδων που αντιδρούν βάση του προγραμματισμού που κάνει ο χρήστης. Αυτές τις εισόδους ή εξόδους μπορούμε να τις διαχειριστούμε γράφοντας κώδικα σε γλώσσα προγραμματισμού Wiring (ουσιαστικά πρόκειται για γλώσσα C++ με κάποιες μετατροπές), στο περιβάλλον προγραμματισμού Arduino IDE (Integrated Development Environment).

Εικόνα 1.1: Λογότυπο Android

Οι δυνατότητες που προσφέρει το Arduino είναι πάρα πολλές, καθώς μπορεί να χρησιμοποιηθεί σε εφαρμογές ρομποτικής και γενικότερα σε αυτοματισμούς καταφέρνοντας έτσι τον έλεγχο των κινήσεων servo, stepper και DC κινητήρων, τη

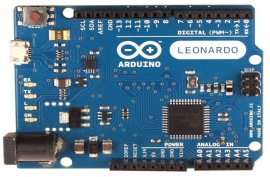
λήψη πληροφοριών από διάφορους αισθητήρες θερμοκρασίας, υπερύθρων κ.α., την αμφίδρομη σειριακή επικοινωνία μεταξύ μιας πλακέτας Arduino και ενός ηλεκτρονικού υπολογιστή χρησιμοποιώντας γλώσσες προγραμματισμού.

Υπάρχουν αρκετές πλακέτες Arduino με πιο γνωστή και διαδεδομένη την Arduino Uno.

Άλλες εξίσου γνωστές πλακέτες Arduino είναι:

- Arduino Nano
- Arduino Mega
- Arduino Leonardo
- Arduino Diecimila
- Arduino Duemilanove
- Arduino Uno

Οι παραπάνω πλακέτες απεικονίζονται στον ακόλουθο πίνακα:

Γνωστές πλακέτες Arduino στην αγορά		
Arduino Nano 	Arduino Mega 	Arduino Leonardo 
Arduino Diecimila 	Arduino Duemilanove 	Arduino Uno 

Πίνακας 1.1: Γνωστές πλακέτες Arduino

Στη συγκεκριμένη πτυχιακή εργασία, επιλέχθηκε το μοντέλο Arduino UNO μιας και καλύπτει όλες τις ανάγκες της εργασίας, καθώς τα pins που διαθέτει επαρκούν για όλες τις λειτουργίες, όπως και η μνήμη του μικροελεγκτή είναι αρκετή για να αποθηκεύσει τον κώδικα που αναπτύχθηκε.

1.1.1 Υλικό

Μια πλακέτα Arduino αποτελείται από ένα μικροελεγκτή (Atmel AVR) και τα απαραίτητα συμπληρωματικά εξαρτήματα τα οποία διευκολύνουν του χρήστη στον προγραμματισμό και την ενσωμάτωση της πλακετας σε άλλα κυκλώματα καθώς και την ομαλή λειτουργία.

Όλες οι πλακέτες Arduino περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V (voltage regulator) και έναν κρυσταλλικό ταλαντωτή 16 MHz (crystal oscillator). Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα bootloader έτσι ώστε να μη χρειάζεται εξωτερικός προγραμματιστής. Τα Arduino συνδέονται σειριακά μέσω ενός απλού level shifter κυκλώματος για τη μετατροπή του σήματος RS-232 σε TTL. Τα σημερινά Arduino προγραμματίζονται με USB μέσω της εφαρμογής προσαρμογέων USB-to-Serial.

Τέλος, μια πλακέτα διαθέτει microcontroller I/O pins για χρήση από άλλα κυκλώματα. Τα Diecimila, Duemilanove και το τρέχον Uno παρέχουν 14 ψηφιακά I/O pins, 6 από τα οποία μπορούν να παράγουν σήματα PWM, και 6 αναλογικά δεδομένα.

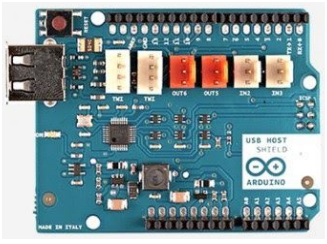
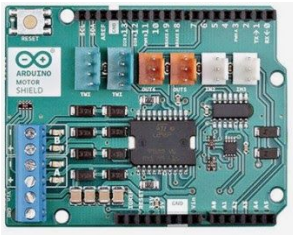
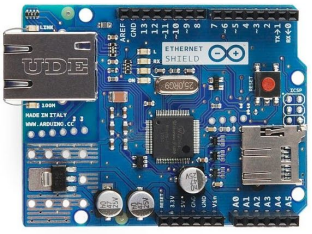
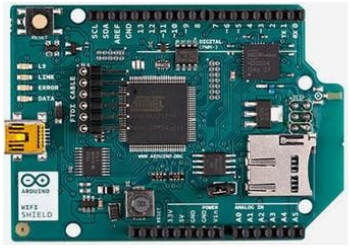
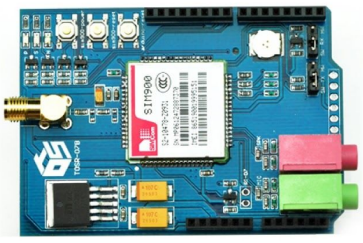
1.1.2 Shields

Ένα από τα πλεονεκτήματα του λογισμικού Arduino είναι πως στην πλακέτα στην οποία αναπτύσσεται μια εφαρμογή, είναι εύκολο να τοποθετήσουμε και άλλες πλακέτες Arduino που έχουν ενσωματωμένα κυκλώματα τα οποία συνδέονται στα pin επεκτείνοντας έτσι τις δυνατότητες της πλακέτας. Αυτές οι πλακέτες ονομάζονται ασπίδες (Shields) και πραγματοποιούν προκαθορισμένες εργασίες. Πιο γνωστά Arduino Shield είναι:

- USB – Host Shield: Το USB Host Shield καθιστά δυνατή τη σύνδεση του Arduino με οποιαδήποτε συσκευή USB μέσω του MAX-3421, το οποίο βρίσκεται στο shield, και χειρίζεται το πρωτόκολλο USB. Πρέπει να σημειωθεί πως το USB - Host Shield για να λειτουργήσει πρέπει να τοποθετηθεί ειδική βιβλιοθήκη μαζί με τις άλλες βιβλιοθήκες του περιβάλλοντος ανάπτυξης.

- Motor Shield: Το Motor Shield καθιστά δυνατή την οδήγηση ρελέ, κινητήρων συνεχούς ρεύματος (DC) και βηματικών κινητήρων (stepper). Η λειτουργιά του βασίζεται στο L298, το οποίο είναι ένας διπλός οδηγός πλήρους γέφυρας (dual full-bridge driver) και μπορεί να οδηγήσει δυο κινητήρες DC με έλεγχο της ταχύτητας και της κατεύθυνσης περιστροφής.
- Ethernet Shield: Το Ethernet Shield καθιστά δυνατή τη σύνδεση του Arduino με τον παγκόσμιο ιστό.
- WiFi shield: Το WiFi Shield καθιστά δυνατή την ασύρματη σύνδεση του Arduino με τον παγκόσμιο ιστό.
- GSM/GPRS Shield: Το GSM/GPRS Shield καθιστά δυνατή τη σύνδεση του Arduino μέσω GSM/GPRS 850/900/1800/1900MHz, για συνδέσεις, φωνής, SMS, Data και Fax. Η λειτουργία του προγραμματίζεται μέσω εντολών AT.
- Bluetooth shield: Το Bluetooth Shield καθιστά δυνατή την ασύρματη σύνδεση του Arduino μέσω του πρωτοκόλλου Bluetooth με άλλες συσκευές.

Στον πίνακα που ακολουθεί απεικονίζονται τα shields που αναφέρθηκαν παραπάνω:

Διάφορα Arduino Shields		
USB – Host Shield 	Motor Shield 	Ethernet Shield 
WiFi shield 	GSM/GPRS Shield 	Bluetooth shield 

Πίνακας 1.2: Διάφορα Arduino Shields

1.2 Εισαγωγή στο Arduino UNO

Το Arduino – Uno είναι ο πιο κλασικός εκπρόσωπος του ανοικτού λογισμικού Arduino, το οποίο αποτελείται από μια πλακέτα με έναν μικροελεγκτή χρονισμένο στους 16 MHz, στην οποία μπορούν να προσαρμοστούν shields.

1.2.1 Χαρακτηριστικά Arduino UNO

Η πλακέτα Arduino – Uno αποτελείται από:

- Ένα μικροελεγκτή ATmega328P
- ATMega16U2 ως μετατροπέα USB-to-Serial για σύνδεση με υπολογιστή. Μπορεί να λειτουργήσει με την τροφοδοσία του USB για ρεύμα έως 500 mA.
- Ενσωματωμένο LED το οποίο είναι συνδεδεμένο στο pin 13
- Είσοδος AREF μέσω της οποίας μπορεί να δοθεί εξωτερική τάση αναφοράς στον ενσωματωμένο A/D converter
- Κουμπί RESET
- Δυο LED (RX και TX) τα οποία αναβοσβήνουν όταν υπάρχει επικοινωνία με τον υπολογιστή μέσω USB, καθώς και οποιαδήποτε άλλη σειριακή επικοινωνία.
- Εξωτερική τροφοδοσία σε ειδική είσοδο 7-12 V
- Τάση λειτουργίας +5V η οποία παρέχεται σε pin για τροφοδοσία των shields
- Παρεχόμενη τάση εξόδου +3.3V με ρεύμα έως 50mA για τροφοδοσία shields

Εικόνα 1.2: Το Arduino Uno

1.2.2 Ο μικροελεγκτής ATmega328P

Ο μικροελεγκτής ATmega328P έχει τα εξής χαρακτηριστικά:

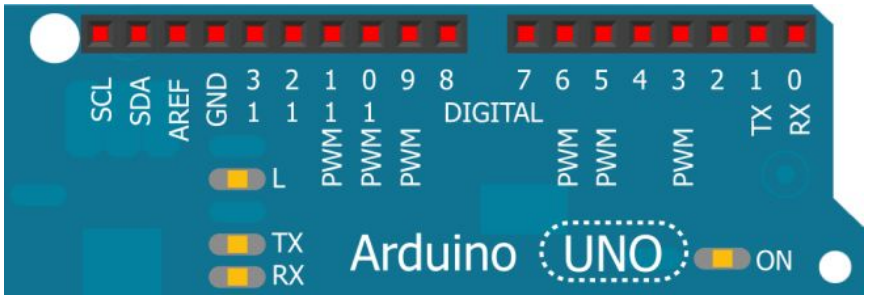
- 14 ψηφιακές εισόδους/εξόδους από τις οποίες οι 6 μπορούν να προγραμματιστούν ως έξοδοι παλμών μεταβλητού Duty Cycle (PWM).
- 6 αναλογικές εισόδους με ανάλυση στα 10 bits.
- Συνεχές ρεύμα 40 mA σε κάθε είσοδο/έξοδο σε λειτουργία 5V και 50mA σε λειτουργία 3,3V.
- Μνήμη προγράμματος 32KB (Flash ROM) από την οποία 0.5KB είναι ο ενσωματωμένος bootloader, με τον οποίο είναι δυνατή η μεταφορά του κώδικα της εφαρμογής από τον υπολογιστή στον επεξεργαστή μέσω USB.
- Μνήμη δεδομένων SRAM 2KB
- Μνήμη EEPROM 1KB
- Ταχύτητα ρολογιού 16MHz
- Πρωτόκολλα επικοινωνίας UART, SPI, I2C
- Εξωτερική τροφοδοσία 7-12V με όριά τάσης εισόδου 6-20V

Εικόνα 1.3: Ο μικροελεγκτής ATmega328P

1.2.3 Ακροδέκτες Arduino UNO

Κάθε μικροελεγκτής Arduino διαθέτει εισόδους και εξόδους για την αλληλεπίδραση με το περιβάλλον του και τα συμπληρωματικά εξαρτήματα. Κάθε ακροδέκτης (pin) λειτουργεί σαν είσοδος αλλα και σαν έξοδος. Το Arduino Uno διαθέτει 20 ακροδέκτες, από τους οποίους 14 είναι ψηφιακοί και 6 είναι αναλογικοί. Στη συνέχεια παρουσιάζονται μέσα από εικόνες και αναλύονται οι ιδιότητές τους.

- Pin 0 και 1: Λειτουργούν ως RX και TX της σειριακής θύρας, όταν το πρόγραμμα ενεργοποιεί τη σειριακή θύρα. Έτσι, όταν το πρόγραμμα στέλνει δεδομένα στη σειριακή θύρα, αυτά προωθούνται και στη θύρα USB μέσω του ελεγκτή Serial-Over-USB, αλλά και στον ακροδέκτη 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή. Αυτό φυσικά σημαίνει, ότι αν στο πρόγραμμά ενεργοποιηθεί το σειριακό interface, χάνει 2 ψηφιακές εισόδους/εξόδους η πλατφόρμα.
- Pin 2 και 3: Λειτουργούν και ως εξωτερικές διακοπές (interrupt 0 και interrupt 1 αντίστοιχα). Ρυθμίζονται μέσα από το πρόγραμμα, ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι, στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει άμεσα και εκτελείται μια συγκεκριμένη συνάρτηση. Οι εξωτερικές διακοπές είναι ιδιαίτερα χρήσιμες σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- Pin 3, 5, 6, 9, 10 και 11: Μπορούν να λειτουργήσουν ως αναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation).
- Pin 4, 7 και 8: Μπορούν να λειτουργήσουν ως ψηφιακοί είσοδοι/έξοδοι γενικού σκοπού.
- Pin 10(SS), 11(MOSI), 12(MISO) και 13(SCK): Χρησιμοποιούνται για το πρωτόκολλο επικοινωνίας SPI. Τα pin 11 και 12 βρίσκονται στον ICSP header, ο οποίος χρησιμοποιείται για τον προγραμματισμό του μικροελεγκτή.
- Pin 13: Το pin 13 είναι συνδεδεμένο με ένα Led πάνω στη πλακέτα με την σήμανση L, αποτελεί στην ουσία κομμάτι του πρώτου πειραματισμού με το Arduino.



Εικόνα 1.4: Οι ψηφιακοί ακροδέκτες του Arduino UNO

Μετά τα 14 ψηφιακά pin ακολουθούν οι ακροδέκτες GDN και AREF των οποίων η λειτουργίες περιγράφονται παρακάτω:

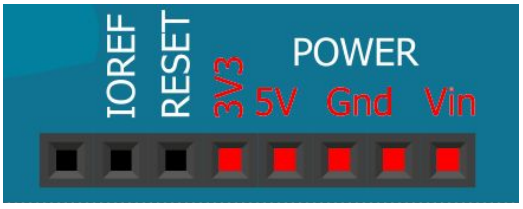
- GND: Είσοδος γείωσης
- AREF: Τάση αναφοράς

Στην άλλη πλευρά του Arduino, με τη σήμανση ANALOG IN όπως φαίνεται και στην εικόνα 4, υπάρχει μια ακόμη σειρά από 6 pin, αριθμημένα από το 0 ως το 5 τα οποία λειτουργούν ως αναλογικές είσοδοι. Η μέτρηση της τάσης γίνεται στο εύρος απο 0V έως 5V και διαβάζεται από το arduino ως τιμή από το 0 (που ισούται με 0V) έως 1023 (που ισούται με 5V). Αυτοί οι ακροδέκτες μπορούν επίσης να μετατραπούν σε ψηφιακές εξόδους αλλά τότε θα μετονομάζονται σε pin 14-19.

Εικόνα 1.5: Οι αναλογικοί ακροδέκτες του Arduino UNO

1.2.4 Τροφοδοσία

Το Arduino, πρέπει να τροφοδοτηθεί με ρεύμα, είτε από τον υπολογιστή μέσω σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm που βρίσκεται στην κάτω αριστερή γωνία της πλακέτας. Για την αποφυγή προβλημάτων, η εξωτερική τροφοδοσία θα πρέπει να είναι από 7-12V. Η παρακάτω εικόνα παρουσιάζει τις εισόδους και τις εξόδους τροφοδοσίας του Arduino UNO.



Εικόνα 1.6: Είσοδοι/Έξοδοι τροφοδοσίας Μικροελεγκτή Arduino

Οι ακροδέκτες τροφοδοσίας είναι:

- Vin: Η τάση εισόδου της πλακέτας, όταν χρησιμοποιεί εξωτερική πηγή ενέργειας. Η τροφοδοσία τάσης γίνεται μέσω αυτού του ακροδέκτη.
- 5V: Η τάση που χρησιμοποιείται από τα διάφορα μέρη της πλακέτας και από τον μικροελεγκτή. Η τάση αυτή, την οποία δίνει αυτός ο ακροδέκτης, είναι είτε η τάση 5V που δίνει η σύνδεση με USB, είτε η ρυθμισμένη τάση που δίνεται μέσω του Vin.
- 3.3V: Η τάση αυτή παράγεται από το ολοκληρωμένο FTDI. Το όριο άντλησης ρεύματος είναι 50mA.
- GND: Είσοδοι γείωσης.

Αριστερά από τα pin τροφοδοσίας υπάρχουν άλλα δυο pin, το RESET και το IOREF όπου η λειτουργία τους περιγράφεται παρακάτω:

- RESET: Κάνει επανεκκίνηση στο Arduino όταν συνδεθεί (μέσω καλωδίου) με οποιοδήποτε από τους 3 ακροδέκτες με ένδειξη ground κάνει επανεκκίνηση το arduino.
- IOREF: Λειτουργεί ως είσοδος/έξοδος αναφοράς. Παρέχει την τάση εισόδου με βάση την οποία λειτουργεί ο μικροελεγκτης.

1.2.5 Μνήμη

Ο μικροεπεξεργαστής ATmega328, έχει τρεις ομάδες μνήμης. Διαθέτει flash memory, στην οποία αποθηκεύονται τα Arduino sketch, SRAM (static random access memory), στην οποία δημιουργείται το sketch και χρησιμοποιεί τις μεταβλητές όταν τρέχει, και EEPROM, η οποία χρησιμοποιείται από τους προγραμματιστές για την αποθήκευση μακροχρόνιων πληροφοριών. Πιο συγκεκριμένα, η μνήμη του ATmega328 αποτελείται από:

- 2KB μνήμης SRAM: Η ωφέλιμη μνήμη, που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κλπ. Η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή πατηθεί το κουμπί επανεκκίνησης.

- 1KB μνήμης EEPROM: Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τα προγράμματα. Σε αντίθεση με την SRAM, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.
- 32KB μνήμης Flash: Από τα 32, τα 2 KB χρησιμοποιούνται από το firmware του Arduino, που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware, είναι αναγκαίο για την εγκατάσταση προγραμμάτων στο μικροελεγκτή μέσω της θύρας USB. Τα υπόλοιπα 30KB της μνήμης Flash, χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

1.3 Το περιβάλλον ανάπτυξής Arduino IDE

Το Arduino IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment), που περιέχει έναν επεξεργαστή πηγαίου κώδικα, έναν μεταγλωττιστή, εργαλεία αυτόματης παραγωγής κώδικα, αποσφαλμάτωση, σύστημα ελέγχου εκδόσεων και εργαλεία κατασκευής γραφικών διασυνδέσεων χρήστη για τις υπό ανάπτυξη εφαρμογές. Συνδέεται με το hardware μέρος του Arduino για να φορτώσει προγράμματα και να επικοινωνεί μαζί του.

Ο κώδικας που έχει γραφτεί για το Arduino ονομάζεται sketch και η γλώσσα που χρησιμοποιείται για την συγγραφή του κώδικα είναι η Wiring, μια παραλλαγή της C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C, καθώς και μερικά χαρακτηριστικά της C++.
Το περιβάλλον ανάπτυξής παρουσιάζεται στην παρακάτω εικόνα.

Εικόνα 1.7: Το περιβάλλον ανάπτυξής Arduino IDE

Όπως φαίνεται στην εικόνα 1.7, στο περιβάλλον ανάπτυξής, σε κάθε sketch υπάρχουν οι εξής δύο συναρτήσεις:

1. setup(): Η συνάρτηση αυτή καλείται κατά την έναρξη ενός sketch. Χρησιμοποιείται για την αρχικοποίηση των μεταβλητών, για τον ορισμό των ακροδεκτών και για την εισαγωγή βιβλιοθηκών. Η συνάρτηση αυτή εκτελείται μόνο μια φορά, κατά την παροχή τροφοδοσίας ή μετά από reset.

2. loop(): Μετά την setup() εκτελείται η loop() συνάρτηση η οποία αποτελεί το κυρίως πρόγραμμα που εκτελείται συνεχώς.

Έτσι, η βασική λειτουργία του Arduino είναι ότι τρέχει η συνάρτηση setup() μία φορά στην αρχή και ακολούθως η loop() ξανά και ξανά μέχρι να το κλείσουμε (να μην τροφοδοτείται με ρεύμα) ή να πατήσουμε το πλήκτρο reset. Στην περίπτωση του reset ξαναεκτελείται η setup() μία φορά στην αρχή και ακολούθως η loop() σε συνέχεια όπως αρχικά ενεργοποιείται με ρεύμα ο μικροελεγκτής.

Στον ακόλουθο πίνακα παρουσιάζονται οι λειτουργίες του περιβάλλοντος ανάπτυξης, υπό μορφή κουμπιών.

ΛΕΙΤΟΥΡΓΙΕΣ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΑΝΑΠΤΥΞΗΣ	
Λειτουργία	Περιγραφή
	Φόρτωση: Μεταγλώττιση και φόρτωση κώδικά στο Arduino. Αν ο κώδικάς μας εμφανίσει σφάλματα τότε δεν μπορεί να γίνει η φόρτωση.
	Επαλήθευση: Έλεγχος συντακτικών λαθών κωδικά.
	Νέο: Δημιουργία νέου αρχείου (sketch).
	Άνοιγμα: Άνοιγμα αρχείου (sketch).
	Αποθήκευση: Αποθήκευση αρχείου (sketch).
	Σειριακή Οθόνη: Εμφάνισή σειριακής οθόνης.

Πίνακας 1.3: Λειτουργίες ανάπτυξής Arduino IDE

1.3.1 Προγραμματισμός Arduino

Όπως αναφέραμε, για τον προγραμματισμό του Arduino χρησιμοποιούμε τη γλώσσα Wiring η οποία βασίζεται στη γλώσσα προγραμματισμού C.

Λόγω της καταγωγής της από την C, στην γλώσσα Wiring μπορούμε να χρησιμοποιούμε ουσιαστικά τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπους δεδομένων και τους ίδιους τελεστές όπως και στην C.

Υπάρχουν όμως κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino οι οποίες επεξηγούνται παρακάτω:

- LOW: Ακέραια σταθερά η οποία έχει την τιμή 0 ή False
- HIGH: Ακέραια σταθερά η οποία έχει την τιμή 1 ή True
- INPUT: Ακέραια σταθερά η οποία έχει την τιμή 0 ή False
- OUTPUT: Ακέραια σταθερά η οποία έχει την τιμή 1 ή True
- pinMode(pin, mode): καθορίζει εάν το συγκεκριμένο ψηφιακό pin θα λειτουργεί σαν είσοδος (INPUT) ή σαν έξοδος (OUTPUT) ανάλογα με την τιμή που δίνεται στη παράμετρο mode.
- digitalWrite(pin, pinstatus): καθορίζει εάν το συγκεκριμένο ψηφιακό pin θα είναι σε κατάσταση HIGH (ή 1 ή 5V ή Vcc) ή σε κατάσταση LOW (ή 0 ή 0V ή GND) ανάλογα με την τιμή που δίνεται στη παράμετρο pinstatus.
- digitalRead(pin): Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού pin (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.
- analogReference(type): Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο type για να καθορίσει την τάση αναφοράς (Vref) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα).
- analogRead(pin): Επιστρέφει έναν ακέραιο από 0 εώς 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο pin αναλογικής εισόδου στην κλίμακα 0 ως Vref.

- analogWrite(pin, value): Θέτει το συγκεκριμένο ψηφιακό pin σε κατάσταση αναλογικής εξόδου (PWM). Η παράμετρος value καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με value 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).

- millis(): Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος.

- delay(time): Σταματά προσωρινά την ροή του προγράμματος για time ms. Παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.

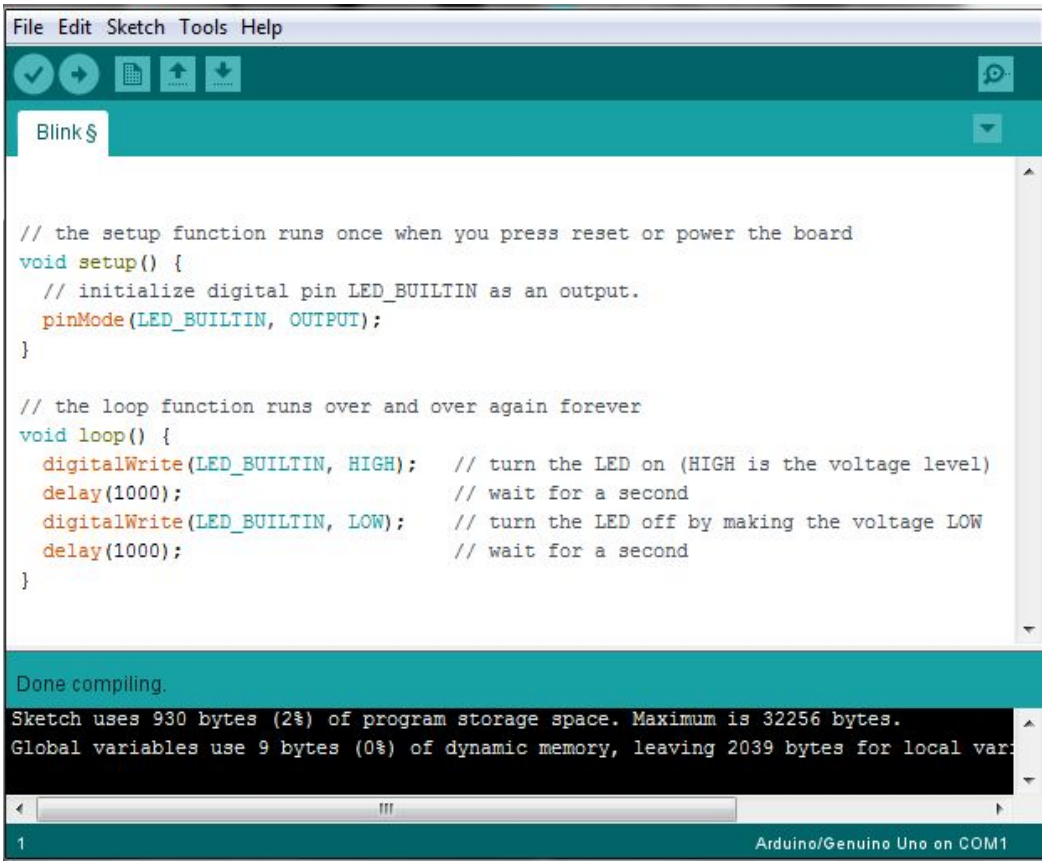
- attachInterrupt(interrupt, function, triggermode): Θέτει σε λειτουργία το συγκεκριμένο interrupt, ώστε να ενεργοποιεί την συνάρτηση function, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο triggermode.
    1. LOW: ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW
    2. RISING: όταν απο LOW γινει HIGH
    3. FALLING: οταν απο HIGH γινει LOW
    4. CHANGE: όταν αλλάξει κατάσταση γενικά
- detachInterrupt(interrupt): Απενεργοποιεί το συγκεκριμένο interrupt.

- noInterupts(): Σταματά προσωρινά την λειτουργία όλων των interrupt

- interrupts(): Επαναφέρει την λειτουργία των interrupt που διακόπηκαν προσωρινά από μια εντολή noInterrupts.

- Serial.begin(datarate): Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud).

- Serial.println(data): Διοχετεύει τα δεδομένα data για αποστολή μέσω του σειριακού interface. Η παράμετρος data μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.



Εικόνα 1.8: Παράδειγμα προγραμματισμού σε Arduino IDE

1.3.2 Ρυθμίσεις περιβάλλοντος ανάπτυξης Arduino IDE

Η βασικές ρυθμίσεις που πρέπει να κάνουμε από την στιγμή που ενώσουμε το Arduino στο σύστημα μας είναι:

- Επιλογή πλακέτας. Από το μενού Tools | Board επιλέγουμε την πλακέτα που έχουμε. Στο συγκεκριμένο παράδειγμα θα χρησιμοποιήσουμε το Arduino UNO, οπότε επιλέγουμε το “Arduino/Genuino UNO”.
- Επιλογή σειριακής θύρας. Από το μενού Tools | Serial Port επιλέγουμε την σειριακή θύρα ή θύρα USB που έχουμε συνδεδεμένο το Arduino.

- Ρυθμίσεις που αφορούν το μέγεθος του κειμένου, τον φάκελο αποθήκευσης, χρήση εξωτερικού κειμενογράφου βρίσκονται στη καρτέλα Preferences (File | Preferences).

2. ΤΟ ΛΟΓΙΣΜΙΚΟ ANDROID

Το Android είναι ένα λειτουργικό σύστημα που αναπτύχθηκε από τη Google, το οποίο ενσωματώνεται σε συσκευές κινητής τηλεφωνίας, οι οποίες διαθέτουν οθόνη αφής (smartphone, tablets), και τρέχει τον πυρήνα (kernel) του λειτουργικού συστήματος Linux. Το λογισμικό Android δίνει την δυνατότητα στους κατασκευαστές λογισμικού να αναπτύξουν κωδικά χρησιμοποιώντας τη γλώσσα προγραμματισμού Java, ελέγχοντας τη συσκευή μέσω των βιβλιοθηκών λογισμικού οι οποίες αναπτύχθηκαν από τη Google.

2.1 Εισαγωγή στο λογισμικό Android

Εικόνα 2.1: Το λογότυπο Android

Το λογισμικό Android παρουσιάστηκε πρώτη φορά το 2007 από την Open Handset Alliance, μια κοινοπραξία 48 τηλεπικοινωνιακών εταιρειών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

Στις μέρες μας, υπάρχουν πάρα πολλές συσκευές με Android, όπου βέβαια η κάθε μία έχει τα δικά της διαφορετικά χαρακτηριστικά και κατασκευάζονται από διαφορετικές εταιρείες όπου η κάθε μια δίνει διαφορετική βαρύτητα σε διαφορετικά χαρακτηριστικά. Μερικές από τις εταιρίες που χρησιμοποιούν το λειτουργικό Android για τα smartphones τους είναι η LG, Samsung, Xiaomi, Sony Ericsson, Motorola.

Το Android έχει μια μεγάλη κοινότητα προγραμματιστών που γράφουν εφαρμογές οι οποίες επεκτείνουν τη λειτουργικότητα των συσκευών. Μέσω του Google Play Store μπορεί να γίνει άμεση εγκατάσταση εφαρμογών χωρίς τη χρήση υπολογιστή. Το Google Play Store προσφέρει πάνω από 2.000.000 εφαρμογές.

2.2 Βασικά χαρακτηριστικά και πλεονεκτήματα

Το Android είναι μια μοναδική πλατφόρμα που επιτρέπει την ανάπτυξη εφαρμογών λογισμικού, το οποίο εκμεταλλεύεται πλήρως τις δυνατότητές μιας συσκευής, χρησιμοποιώντας οποιαδήποτε από τις βασικές λειτουργίες των συσκευών όπως τις τηλεφωνικές κλήσεις, τη λήψη βίντεο και φωτογραφιών, το GPS κτλ. Είναι πλατφόρμα multi tasking, πράγμα που σημαίνει ότι κάθε εφαρμογή μπορεί να τρέχει παράλληλα με άλλες εφαρμογές χωρίς να επηρεαστεί η απόδοσή τους.

Το Android είναι πλατφόρμα ανοιχτού κώδικα, το οποίο σημαίνει πως μπορεί τροποποιηθεί και να επεκταθεί εύκολα ώστε να συμβαδίζει με τις τελευταίες τεχνολογικές εξελίξεις. Το γεγονός ότι είναι πλατφόρμα ανοικτού κώδικα διασφαλίζει τη συνεχή πρόοδο και εξέλιξη.

Οι εφαρμογές που δημιουργούνται και προστίθενται από τρίτους προγραμματιστές δεν διαφέρουν σε τίποτα από τις ήδη ενσωματωμένες στο τηλέφωνο και έχουν την ίδια πρόσβαση σε όλες τις κύριες λειτουργίες της συσκευής. Αυτό, επιτρέπει στους τελικούς χρήστες να απολαμβάνουν ένα μεγάλο φάσμα εφαρμογών που μπορούν να χρησιμοποιηθούν για σχεδόν απεριόριστους σκοπούς. Οι συσκευές που έχουν χτιστεί στην πλατφόρμα Android, δίνουν τη δυνατότητα στους χρήστες να προσαρμόσουν πλήρως τη συσκευή τους ανάλογα με τις ανάγκες τους και να έχουν ένα πιο εξατομικευμένο εργαλείο.

Η πλατφόρμα ενθαρρύνει πολλους νεους προγραμματιστές και μή, να ασχοληθουν με την ανάπτυξη εφαρμογών καθώς παρέχει μια μεγάλη ποικιλία από

βιβλιοθήκες και χρήσιμα εργαλεία που μπορούν να χρησιμοποιηθούν για την υλοποίηση πολλών εφαρμογών. Αυτή η δυνατότητα παροχής έτοιμων εργαλείων αυξάνει την παραγωγικότητα των προγραμματιστών και τους βοηθά να δημιουργήσουν πλούσιο λογισμικό γρηγορότερα και με λιγότερα λάθη.

Εικόνα 2.2: Λογότυπο και το εικονίδιο του Play Store

2.3 Οι εκδόσεις Android

Ξεκινώντας από το 2008 με την κυκλοφορία του πρώτου Android smartphone μέχρι σήμερα έχουν βγεί αρκετές εκδόσεις όπως είναι φυσικό και υπάρχει το χαρακτηριστικό γνώρισμα ότι όλες έχουν πάντα ονόματα κάποιου γλυκού με εξαίρεση τις εκδόσεις 1.0 και 1.1.

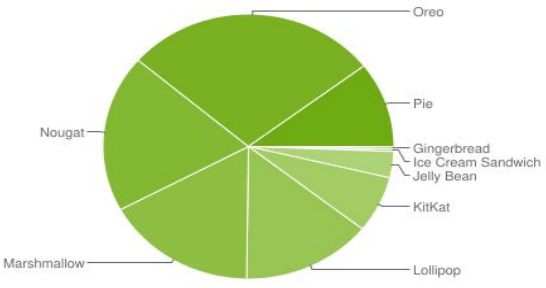
Στη παρακάτω λίστα παρουσιάζονται όλες οι εκδόσεις Android που έχουν βγεί σε κυκλοφορία μέχρι και σήμερα:

- Android 1.0
- Android 1.1
- Android 1.5 (Cupcake)
- Android 1.6 (Donut)
- Android 2.0 / 2.1 (Eclair)
- Android 2.2 (Froyo)
- Android 2.3 (Gingerbread)
- Android 3.0 / 3.2 (Honeycomb)
- Android 4.0 / 4.0.2 (Ice Cream Sandwich)
- Android 4.1 / 4.2 και 4.3 (Jelly Bean)

- Android 4.4 (KitKat)
- Android 5.0 (Lollipop)
- Android 6.0 / 6.0.1 (Marshmallow)
- Android 7.0 / 7.1.2 (Nougat)
- Android 8.0 / 8.1 (Oreo)
- Android 9.0 (Pie)

Στη παρακάτω εικόνα βλέπουμε το ποσοστό τον συσκευών που τρέχουν κάθε έκδοση Android από την έκδοση Gingerbread και έπειτα, για τον Μάιο του 2019:

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

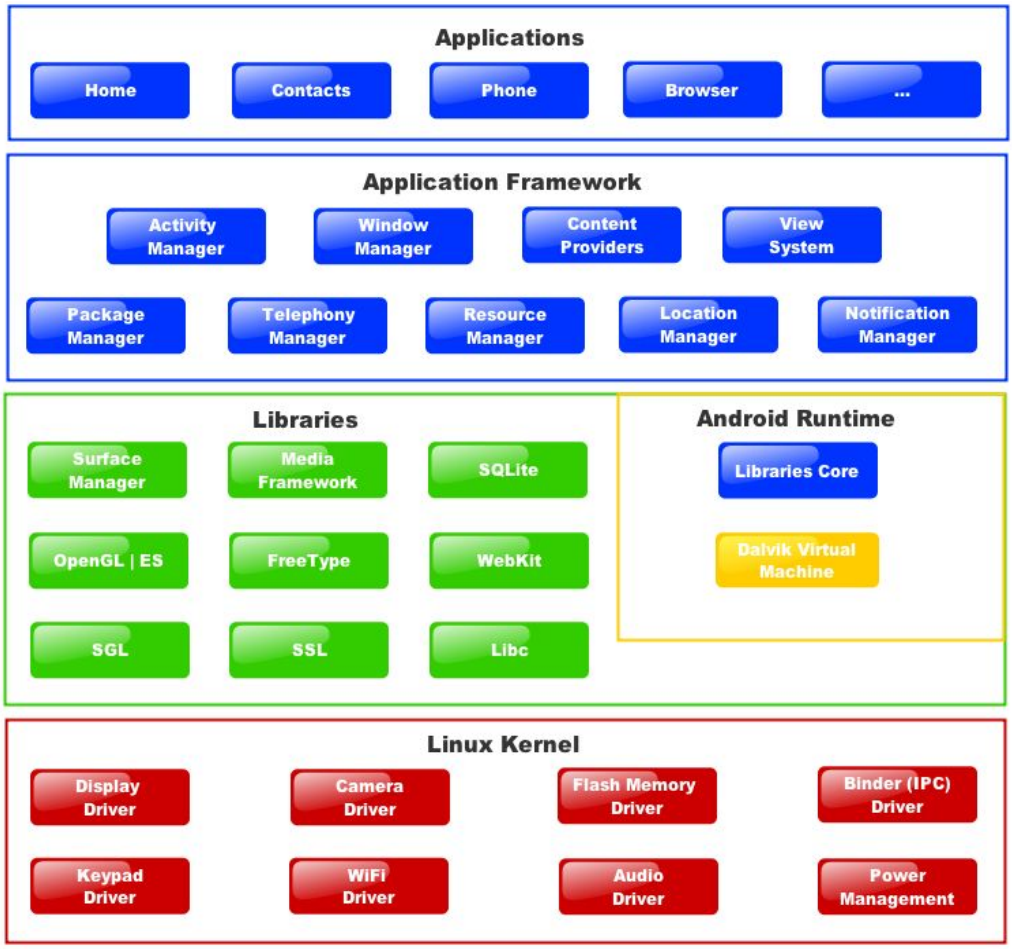


Data collected during a 7-day period ending on May 7, 2019
Any versions with less than 0.1% distribution are not shown.

Εικόνα 2.3: Ποσοστό τον συσκευών που τρέχουν κάθε έκδοση Android από την έκδοση Gingerbread και μετά, για τον Μάιο του 2019

2.4 Αρχιτεκτονική Android

Το Android δεν είναι μόνο ένα λειτουργικό σύστημά. Είναι μια στοίβα λογισμικού η οποία αποτελείται από το λειτουργικό σύστημα, τις υπηρεσίες διασύνδεσης με τις εφαρμογές (middleware) και τέλος από τις κύριες (core) εφαρμογές. Στο επίσημο σχεδιάγραμμα που ακολουθεί θα δούμε οπτικά την αρχιτεκτονική αυτή.



Πίνακας 2.1: Η αρχιτεκτονική του Android

Τα επίπεδα πρόσβασης της στοίβας Android παρουσιάζονται παρακάτω από το υψηλότερο προς το χαμηλότερο:

- Επίπεδο εφαρμογών (Applications)
- Επίπεδο πλαισίου εφαρμογών (Application framework)
- Επίπεδο βιβλιοθηκών (Libraries)

- Επίπεδο χρόνου εκτέλεσης (Android runtime)
- Επίπεδο πυρήνα Linux (LInux Kernel)

2.4.1 Επίπεδο εφαρμογών (Applications)

Σε αυτό το επίπεδο περιλαμβάνεται ένα σύνολο από βασικές εφαρμογές μερικές από τις οποίες είναι το e-mail client, πρόγραμμα sms, ημερολόγιο, χάρτες, browser, επαφές κ.α. Όλες οι εφαρμογές είναι γραμμένες με χρήση της γλώσσας προγραμματισμού Java.

2.4.2 Επίπεδο πλαισίου εφαρμογών (Application framework)

Οι προγραμματιστές εφαρμογών Android έχουν τις δυνατότητες μιας ανοικτού κώδικα πλατφόρμας ανάπτυξης εφαρμογών έτσι ώστε να είναι ικανοί να αναπτύξουν καινοτόμες εφαρμογές. Το επίπεδο πλαισίου των εφαρμογών της αρχιτεκτονικής Android, αποτελείται από ένα σύνολο συστημάτων και εφαρμογών. Παρακάτω παρουσιάζονται τα σημαντικότερα δομικά στοιχεία του πλαισίου εφαρμογών:

- Σύστημα προβολών (View System): Αποτελεί ένα μεγάλο σύνολο από αντικείμενα GUI τα οποία μπορούν να χρησιμοποιηθούν στο σχεδιασμό μιας εφαρμογής. Παράδειγμα τέτοιων προβολών είναι οι λίστες (listView), το πλέγμα (GridView), πεδία εισαγωγής κειμένου, κλπ.
- Πάροχος Περιεχομένου (Content Provider): Δίνει τη δυνατότητα στις εφαρμογές να μοιράζονται ή να ανταλλάσσουν δεδομένα μιας συγκεκριμένης μορφής η οποία ορίζεται από τον πάροχο. Παραδείγματα τέτοιων δεδομένων είναι οι επαφές χρηστών και οι βάσεις δεδομένων των εφαρμογών.
- Διαχειριστής Πόρων (Resource Manager): Παρέχει πρόσβαση σε υλικό όπως εικόνες, αρχεία xml, πίνακες χαρακτήρων κλπ.
- Διαχειριστής Ειδοποιήσεων (Notification Manager): Δίνει στις εφαρμογές πρόσβαση στις υπηρεσίες ειδοποιήσεων χρήστη όπως είναι οι ειδοποιήσεις στη notification bar, η δόνηση του κινητού και ενεργοποίηση της οθόνης, κλπ.

- Διαχειριστής Δραστηριοτήτων (Activity Manager): Διαχειρίζεται τον κύκλο της ζωής των δραστηριοτήτων και παρέχει δυνατότητα εναλλαγής από δραστηριότητα σε δραστηριότητα κρατώντας αποθηκευμένη στη μνήμη τη σειρά εκτέλεσης τους.
- Διαχειριστής Τοποθεσίας (Location Manager): Παρέχει πληροφορίες για τη γεωγραφική θέση της συσκευής έτσι ώστε ο χρήστης να γνωρίζει που βρίσκεται κάθε στιγμή.

2.4.3 Επίπεδο βιβλιοθηκών

Σε αυτό το επίπεδο της στοίβας έχουμε τις βιβλιοθήκες που χρησιμοποιεί το Android και περιλαμβάνει ένα σύνολο από βιβλιοθήκες γραμμένες σε C και C++. Ουσιαστικά αποτελούν τα APIs που είναι διαθέσιμα στους προγραμματιστές για την ανάπτυξη των εφαρμογών. Οι δυνατότητες των βιβλιοθηκών γίνονται εμφανείς στους προγραμματιστές από τη στοίβα του πλαισίου εφαρμογής. Μερικές από τις κύριες βιβλιοθήκες του Android είναι:

- System C library: Μια ενσωμάτωση της standard βιβλιοθήκης συστήματος της C (libc) τροποποιημένη για κινητές συσκευές βασισμένες στο Linux.
- Βιβλιοθήκες πολυμέσων: Υποστηρίζει την αναπαραγωγή και εγγραφή μέσων ήχου και εικόνας, όπως MPEG4, H.264, MP3,JPG, PNG.
- Surface Manager: Διαχειρίζεται το υποσύστημα προβολής και συνθέτει δισδιάστατα και τρισδιάστατα επίπεδα γραφικών τα οποία προέρχονται από πολλαπλές εφαρμογές.
- LIbWebCore: Μηχανή υποστήριξης πλοήγησης στο διαδίκτυο η οποία χρησιμοποιείται και από τον ενσωματωμένο browser του Android αλλά και από τις WebViews που ενσωματώνονται στις εφαρμογές.
- SGL: Μηχανή δισδιάστατων γραφικών.
- Βιβλιοθήκες 3D: Μια υλοποίηση βασισμένη στα APIs του OpenGL ES1. Οι βιβλιοθήκες αυτές χρησιμοποιούν είτε τρισδιάστατη επιτάχυνση υλικού, όταν αυτή είναι διαθέσιμη, είτε μια βελτιωμένη τρισδιάστατη επιτάχυνση λογισμικού σε περίπτωση που η πρώτη δεν είναι διαθέσιμη.

- FreeType: Παρέχει ευκρίνεια γραφικών στα bitmaps και τις γραμματοσειρές των εφαρμογών του συστήματος.
- SQLite: Μια πανίσχυρη και ελαφριά σχεσιακή βάση δεδομένων.

2.4.4 Επίπεδο χρόνου εκτέλεσης (Android Runtime)

Αυτό το επίπεδο αποτελείται από ένα σύνολο από βασικές βιβλιοθήκες και από την Dalvik Virtual Machine. Η Dalvik Virtual Machine είναι υλοποίηση μιας εικονικής μηχανής JAVA για φορητές συσκευές που δημιούργησε η Google και έχει αναπτυχθεί με τέτοιο τρόπο έτσι ώστε μια συσκευή να μπορεί να διαχειρίζεται πολλές εικονικές συσκευές με μεγάλη απόδοσή.

2.4.5 Επίπεδο πυρήνα Linux (Linux Kernel)

Η βάση της στοίβας λογισμικού του Android είναι ο πυρήνας Linux. Βασίζεται στο πυρήνα LInux 2.6 του Linux Kernel, ο οποίος υποστηρίζει λειτουργίες του λειτουργικού συστήματος που αφορούν διαχείριση μνήμης, διαχείριση διεργασιών, λειτουργίες δικτύου, ασφάλεια του λειτουργικού και ένα σύνολο οδηγών υλικού (device drivers). Οι οδηγοί αυτοί είναι υπεύθυνοι για την επικοινωνία του software με το hardware της συσκευής.

Εικόνα 2.4: Το UI και εφαρμογές μιας κινητής συσκευής με λογισμικό Android

2.5 Οι εφαρμογές Android

Κάθε εφαρμογή Android αποτελείται από ένα σύνολο αρχείων και φακέλων δομημένα σε μορφή project, τα οποία αφού γίνει η επαλήθευση τους μέσω του Android SDK μας δίνουν το αρχείο .apk. Το αρχείο αυτό αποτελεί την εφαρμογή και μπορούμε να εγκαταστήσουμε στις συσκευές που επιθυμούμε. Σε κάθε εφαρμογή Android υπάρχουν 4 διαφορετικά είδη συστατικών μερών ή ένας συνδυασμός αυτών. Κάθε είδος εξυπηρετεί συγκεκριμένο σκοπό και έχει συγκεκριμένο κύκλο ζωής που ορίζει πότε το συστατικό θα δημιουργηθεί και θα καταστραφεί. Την απόφαση για το ποια συστατικά μέρη θα χρησιμοποιηθούν θα την πάρει ο προγραμματιστής και θα πρέπει να τα ορίσει στο αρχείο AndroidManifest.xml. Τα τέσσερα συστατικά μέρη είναι τα: Activities, Services, Content Providers, Broadcast receivers και αναλύονται παρακάτω.

1. Activities (Δραστηριότητες)

   Αποτελούν το κύριο στοιχείο της εφαρμογής. Είναι οθόνες διεπαφής χρήστη (UI) η οποία προβάλλουν πληροφορίες καθώς περιέχουν εικόνες, κείμενα, κουμπιά και αλληλεπιδρούν με γεγονότα. Κάθε εφαρμογή έχει τόσες δραστηριότητες όσες και οι διαφορετικές οθόνες οι οποίες προβάλλονται στο

χρήστη. Οι δραστηριότητες μπορούν και συνεργάζονται μεταξύ τους ώστε να υπάρχει ομοιομορφία στη δομή της εφαρμογής. Ενδεικτικά, η μετάβαση από οθόνη σε οθόνη υλοποιείται με την έναρξη μιας νέας δραστηριότητας, η οποία μπορεί να επιστρέφει κάποια τιμή σε προηγούμενη, όπως για παράδειγμα η δραστηριότητα που εμφανίζει τις επαφές του τηλεφώνου, επιστρέφει το προφίλ της επιλεγμένης επαφής στη δραστηριότητα που την κάλεσε.

2. Services (Υπηρεσίες)

   Πρόκειται για τις λειτουργίες που είναι σχεδιασμένες να επεξεργάζονται δεδομένα στο παρασκήνιο για μεγάλο χρονικό διάστημα χωρίς διεπαφή του χρήστη. Μπορούν να εμφανίζουν αποτελέσματα και πληροφορίες ακόμα και όταν η εφαρμογή δεν είναι στο προσκήνιο. Για παράδειγμα, μια υπηρεσία μπορεί να παίζει μουσική στο παρασκήνιο ενώ ο χρήστης έχει ανοιχτή μια άλλη εφαρμογή. Η διάρκεια και ο κύκλος ζωής μιας υπηρεσίας διαφέρει από αυτή της δραστηριότητας διότι θα συνεχίσει να τρέχει μέχρι να τη σταματήσει ο χρήστης ή τη σταματήσει το σύστημα έτσι ώστε να εξοικονομήσει μνήμη.

3. Content Providers (Πάροχοι Περιεχομένου)

   Διαχειρίζονται αποθηκευτικούς χώρους για δεδομένα που μπορεί να χρησιμοποιήσει οποιαδήποτε εφαρμογή και επιτρέπουν στις εφαρμογές να επεξεργαστούν δεδομένα συγκεκριμένου τύπου.

4. Broadcast Receivers (Δέκτες Μετάδοσης)

   Πρόκειται για ένα είδους υπηρεσίας η οποία αντιλαμβάνεται κάποια γεγονότα του συστήματος και αναλαμβάνει να ενημερώσει το σύστημα ή τις υπόλοιπες εφαρμογές. Ο σκοπός τους είναι διπλός καθότι μπορούν και να ενημερωθούν για κάποιο συμβάν από άλλες εφαρμογές αλλά και να ειδοποιήσουν τις υπόλοιπες εφαρμογές και το σύστημα για κάποιο συμβάν που τις ενεργοποίησε. Δεν έχουν γραφικό περιβάλλον αλλά μπορούν να προβάλουν ειδοποίηση στον χρήστη μέσω της μπάρας ειδοποιήσεων.

Σε ορισμένες περιπτώσεις οι προγραμματιστές έκτος από τα παραπάνω είδη συστατικών μερών μιας εφαρμογης Android, προσθέτουν και τα Intents (Προθέσεις) τα οποία ουσιαστικά εξασφαλίζουν την μετάβαση από μια δραστηριότητα σε μια άλλη και χρησιμοποιούνται για ανταλλαγή δεδομένων

2.5.1 Το αρχείο AndroidManifest.xml

Κάθε project αναπτυξης Android εφαρμογής, περιέχει ένα αρχείο στο οποίο βρίσκονται καταχωρημένες ολες οι σημαντικές πληροφορίες της εφαρμογής σε ένα αρχείο που ονομάζεται AndroidManifest.xml. Πρόκειται για ένα αρχείο xml μέσα στο οποίο ο προγραμματιστής καταχωρεί τις σημαντικές πληροφορίες της εφαρμογής για τη χρήση από το λειτουργικό σύστημά. Αυτές οι πληροφορίες μπορεί να είναι:

- Το όνομα του πακέτου της εφαρμογής
- Το κανονικό όνομα της εφαρμογης που φαίνεται στον χρήστη
- Η έκδοση των API που χρησιμοποιούνται
- Ο αριθμός έκδοσης της εφαρμογής
- Οι άδειες χρήσης που ζητάει η εφαρμογή
- Όλες οι δραστηριότητες, πάροχοι περιεχομένου, υπηρεσίες, κλπ. που περιέχει και χρησιμοποιεί η εφαρμογή.

2.5.2 Οι φακελοι src & res

Ο φάκελος src (source) περιέχει τα αρχεία κλάσης της Java, όλων των Activities, Services, Content providers κλπ. Ο φάκελος αυτος αποτελεί τον μοναδικό φάκελο του project στον οποίο αποθηκεύονται τα αρχεία του κώδικα της εφαρμογής.

Ο φάκελος res (resources) περιέχει όλα τα αρχεία εικόνας, ήχου και οτιδήποτε σχετικό με την εμφάνιση της εφαρμογής που χρησιμοποιούν οι διαδικασίες που βρίσκονται στον φάκελο src.

Εκτός βέβαια από τους παραπάνω βασικούς φακέλους, υπάρχουν και κάποιοι φάκελοι οι οποίοι μπορεί να θεωρηθούν και περιττοί αναλόγως τις ανάγκες της εκάστοτε εφαρμογής. Υπάρχει φάκελος με τα διαθέσιμα APIs αναλόγως την έκδοση που εχει επιλέξει ο προγραμματιστής, φάκελος με τις διαθέσιμες βιβλιοθήκες που χρησιμοποιεί το project κ.α.

2.5.3 Κύκλος ζωής μιας εφαρμογής Android

Όπως αναφέραμε και παραπάνω, τα Activities (δραστηριότητες) μιας εφαρμογής Android αποτελούν το κύριο στοιχείο της εφαρμογής καθώς αλληλεπιδρούν μεταξύ τους αλλά και με τον χρήστη δίνοντας έτσι μια συνολική εμπειρία χρήσης της εφαρμογής.

Οι δραστηριότητες του συστήματος διαχειρίζονται σαν μια στοίβα από δραστηριότητες, αυτό σημαίνει πως οταν μια δραστηριότητα ξεκινήσει, τότε τοποθετείται στην κορυφή της στοίβας ενω οι υπόλοιπες περιμένουν κάτω στην στοίβα, η οποίες θα έρθουν στο προσκήνιο μόνο αν η τρέχουσα δραστηριότητα τελειώσει.

Υπάρχουν τέσσερις καταστάσεις στις οποίες μπορεί να βρεθεί μια δραστηριότητα:

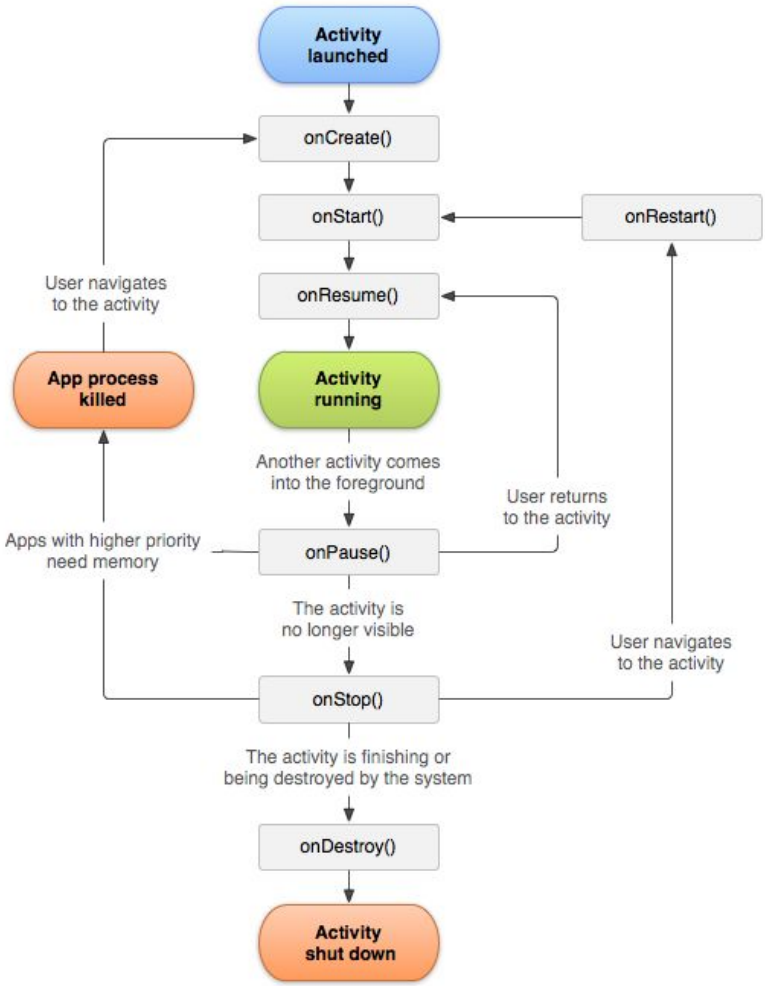
- Αν μια δραστηριότητα βρίσκεται στο προσκήνιο (δηλαδή εμφανίζεται στην οθόνη και βρίσκεται στην κορυφή της στοίβας), τότε βρίσκεται σε κατάσταση active ή running.
- Αν μια δραστηριότητα δεν είναι focused αλλά είναι ακόμα ορατή (δηλαδή μια νέα δραστηριότητα που δεν είναι full-sized είναι focused πάνω από την πρώτη δραστηριότητα), τότε βρίσκεται στην κατάσταση paused. Σε αυτή την κατάσταση η δραστηριότητα είναι ζωντανή (διατηρεί όλη την κατάσταση), αλλά μπορεί να κλείσει από το σύστημα σε περίπτωση πολύ χαμηλής μνήμης.
- Αν μια δραστηριότητα είναι πλήρως κρυμμένη από άλλη δραστηριοτητα, τότε βρίσκεται στην κατάσταση stopped. Διατηρεί όλη την κατάσταση και τις πληροφορίες, ωστόσο δεν είναι πλέον ορατή στον χρήστη, έτσι το παράθυρό της είναι κρυμμένο και συνήθως κλείνει από το σύστημα όταν υπάρχει ανάγκη για μνήμη.
- Αν μια δραστηριότητα είναι στην κατάσταση paused ή stopped, το σύστημα μπορεί να την αφαιρέσει από την μνήμη, είτε ρωτώντας την να την σταματήσει, είτε απλώς κλείνοντας την διεργασία της. Όταν εμφανίζεται ξανά στο χρήστη, πρέπει να ξεκινήσει πάλι και να αποκαταστήσει την προηγούμενη της κατάσταση.

Στην εικόνα 2.5 που παρουσιάζεται παρακάτω, υπάρχει ένα διάγραμμα το οποίο αναπαριστά τα μονοπάτια των διαφόρων καταστάσεων μιας δραστηριότητας. Όπως βλέπουμε υπάρχουν τρία loops τα οποία ένας προγραμματιστής μπορεί να διαχειριστεί μέσα σε μια δραστηριοτητα.

1. entire lifetime loop: Διενεργείται μεταξύ της πρώτης κλήσης onCreate() μέχρι μιας τελικής κλήσης onDestroy(). Μια δραστηριοτητα θα κάνει όλες τις αρχικοποιήσεις στην κλήση onCreate(), και θα απελευθερώσει όλους τους πόρους στην onDestroy().

2. visible lifetime loop: Διενεργείται μεταξύ μιας κλήσης onStart() και της αντίστοιχης κλήσης onStop(). Κατά την διάρκεια αυτού του loop ο χρήστης μπορεί να δει την δραστηριότητα στην οθόνη, παρόλο που μπορεί να μην είναι στο προσκήνιο και να αλληλεπιδρά με τον χρήστη. Μεταξύ των δύο αυτών μεθόδων μπορούν να διατηρούνται οι πόροι που χρειάζονται για να δείχνουν την δραστηριότητα στο χρήστη. Αυτές οι δύο μέθοδοι μπορούν να καλούνται πολλές φορές, καθώς μια δραστηριότητα εμφανίζεται ή είναι κρυμμένη στο χρήστη.

3. foreground lifetime loop: Διενεργείται μεταξύ μιας κλήσης onResume() και της αντίστοιχης κλήσης onPause(). Κατά τη διάρκεια αυτή, η δραστηριότητα βρίσκεται μπροστά από όλες τις υπόλοιπες δραστηριότητες και αλληλεπιδρά με τον χρήστη. Μια δραστηριότητα μπορεί συχνά να μεταβεί μεταξύ των δύο αυτών μεθόδων, για παράδειγμα όταν η συσκευή τίθεται σε sleep mode όταν φτάνει ένα αποτέλεσμα μιας δραστηριότητας μέσω ενός intent. Για αυτό το λόγο ο κώδικας σε αυτές τις μεθόδους πρέπει να είναι αρκετά ελαφρύς.



Εικόνα 2.5: Ο κύκλος ζωής μιας εφαρμογής Android

2.6 Η ασφάλεια στο Android

Απο τη στιγμή που μια εφαρμογή εγκαθίσταται σε μια συσκευή, λειτουργεί αποκλειστικά στη δική της εικονική μηχανή η οποία αποτελεί το πλαίσιο ασφάλειας της εφαρμογής (sandbox). Η ασφάλεια στα Android συστήματα επιτυγχάνεται καθώς:

- Κάθε εφαρμογή αντιμετωπίζεται σαν ενας διαφορετικός χρήστης
- Το λογισμικό δίνει ένα μοναδικό αριθμό ID στις εφαρμογές ο οποίος είναι άγνωστος σε κάθε εφαρμογή. Το σύστημα αναθέτει συγκεκριμένες άδειες χρήσης (permissions) στα αρχεία της εφαρμογής, και μονό η εφαρμογή με το σωστό ID μπορεί να έχει πρόσβαση σε αυτά.

- Κάθε εφαρμογή τρέχει στη δική της εικονική μηχανή (VM), ξεχωριστα από τις υπόλοιπες εικονικές μηχανές τον άλλων εφαρμογών. Η κάθε εικονική μηχανή ξεκινάει μόλις ζητηθεί από το σύστημα και κλείνει είτε επειδή δεν χρησιμοποιείται πλέον, είτε επειδή το σύστημά θέλει να ελευθερώσει πόρους της μνήμης για τη χρήση τους από άλλη εφαρμογή.

Με αυτό τον τρόπο το Android χρησιμοποιεί την αρχή των ελάχιστων δικαιωμάτων. Έτσι κάθε εφαρμογή έχει πρόσβαση, μέσω του AndroidManifest, μόνο σε όσους πόρους του συστήματος χρειάζεται και όχι σε περισσότερους. Κατά την εγκατάσταση μιας εφαρμογής γίνονται γνωστά στον χρήστη τα δικαιώματα και οι πόροι που χρειάζεται η εφαρμογή για να λειτουργήσει έτσι ώστε ο χρήστης να έχει την επιλογή να δεχτεί ή να απορρίψει το αίτημα της εφαρμογής να έχει πρόσβαση στους πόρους που ζητάει.

2.7 Το Android SDK

Το Android SDK (Software Development Kit) είναι το επίσημο εργαλείο της Google για την ανάπτυξη εφαρμογών με τη πλατφόρμα Android. Περιέχει μια μεγαλη συλλογή από εργαλεία και βιβλιοθήκες, απαραίτητα για τους προγραμματιστές. Περιλαμβάνει παραδείγματα εφαρμογών με τον πηγαίο τους κώδικα, βοηθήματα, καθώς και εικονική μηχανή για τον έλεγχο της εφαρμογής που αναπτύσσεται. Ακόμη, αναλαμβάνει μεταγλωττιστή ώστε να μπορεί να τρέχει στην εικονική μηχανή Dalvik. Το Android SDK περιλαμβάνει μια εικονική μηχανή που χρησιμοποιεί το Android Virtual Device (AVD). Το AVD επιτρέπει τη δημιουργία μιας μιμούμενης συσκευής ώστε η εφαρμογή να δοκιμάζεται σε ποικιλία παραλλαγών και να διασφαλίζεται η ομαλή λειτουργία της.

Για την υποστήριξη προηγούμενων εκδόσεων λειτουργικού, το SDK δίνει τη δυνατότητα στον προγραμματιστή να επιλέξει το ελάχιστο επίπεδο API που θα στοχεύει η εφαρμογή. Δηλαδή, είναι εφικτή η επιλογή του εύρους των εκδόσεων που θα τρέχει η εφαρμογή, επιλέγοντας την παλαιότερη και την νεότερη. Αυτό είναι αναγκαίο επειδή πολλοί χρήστες κατέχουν συσκευές με παλαιότερες εκδόσεις

Android και ο κατασκευαστής δεν έχει ή δεν πρόκειται να βγάλει ενημέρωση για τις συσκευές αυτές.

2.8 Εισαγωγή στο Android Studio

Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment) για την ανάπτυξη εφαρμογών στην πλατφόρμα Android.

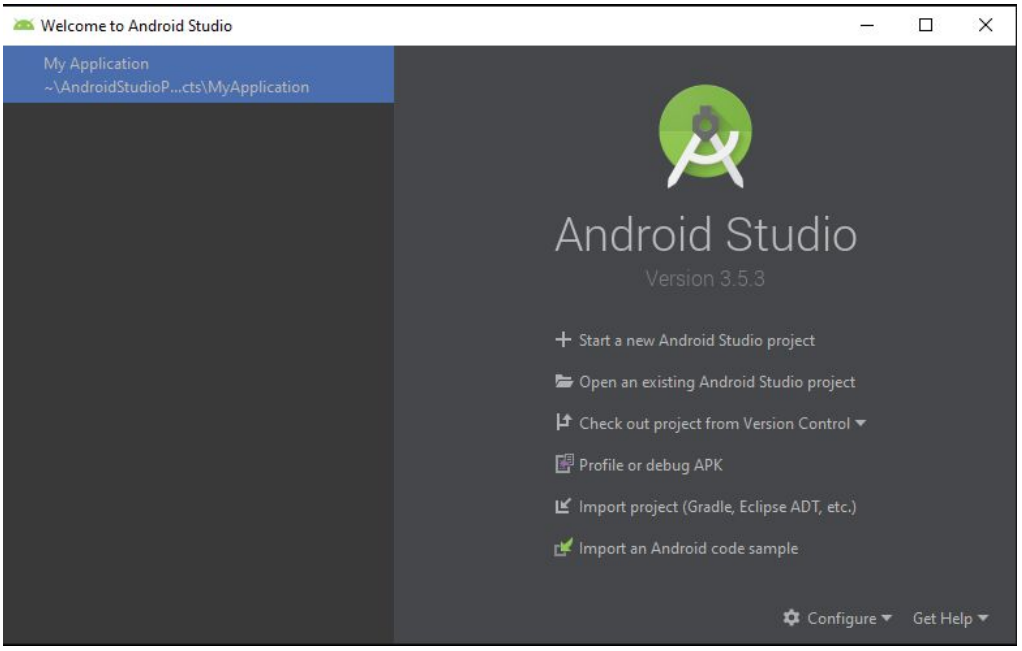
Εικόνα 2.6: Το λογότυπο του Android Studio

Βασισμένο στο λογισμικό της JetBrains IntelliJ IDEA, το Android Studio σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux, και αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

2.8.1 Το περιβάλλον Android Studio

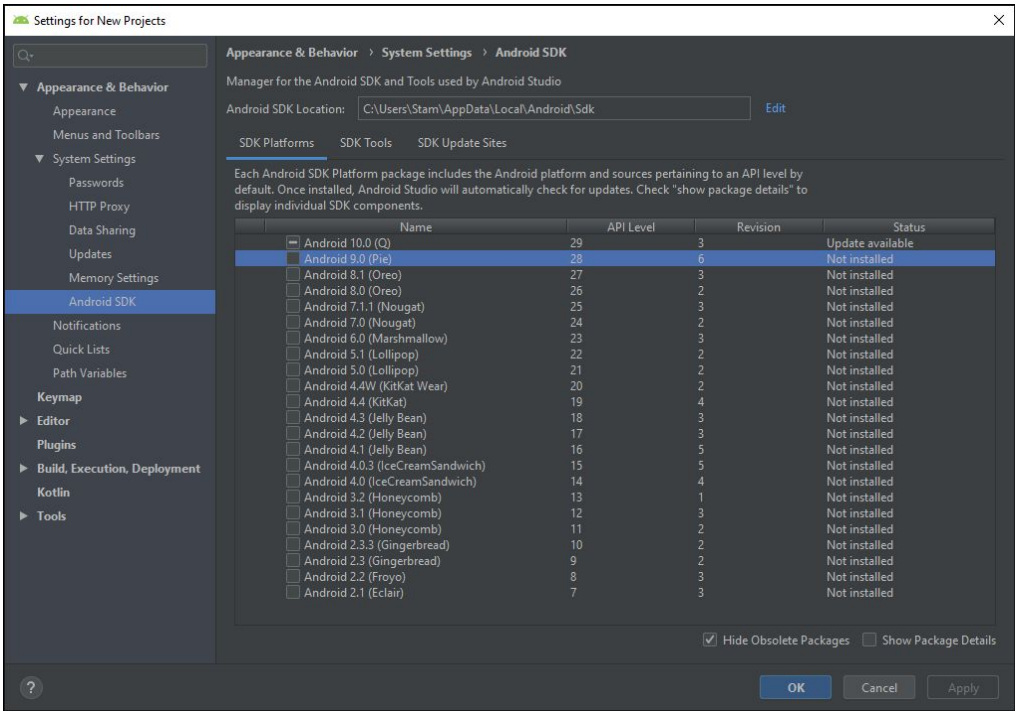
Αφού έχουμε κατεβάσει και εγκαταστήσει το Android Studio στον υπολογιστή μας, είμαστε σε θέση να δημιουργήσουμε την πρώτη μας εφαρμογή για smartphones. Στη συνέχεια παρουσιάζεται το περιβάλλον του Android Studio και μερικές απο τις δυνατότητες του μεσα απο το παράδειγμα δημιουργίας μιας απλής εφαρμογής.

Κάνοντας εκκίνηση του Android Studio, στον υπολογιστή μας εμφανίζεται ένα παράθυρο καλωσορίσματος το οποίο μας δίνει τη δυνατότητα να ξεκινήσουμε ένα καινούργιο project ή να επεξεργαστούμε ήδη υπάρχοντα projects:



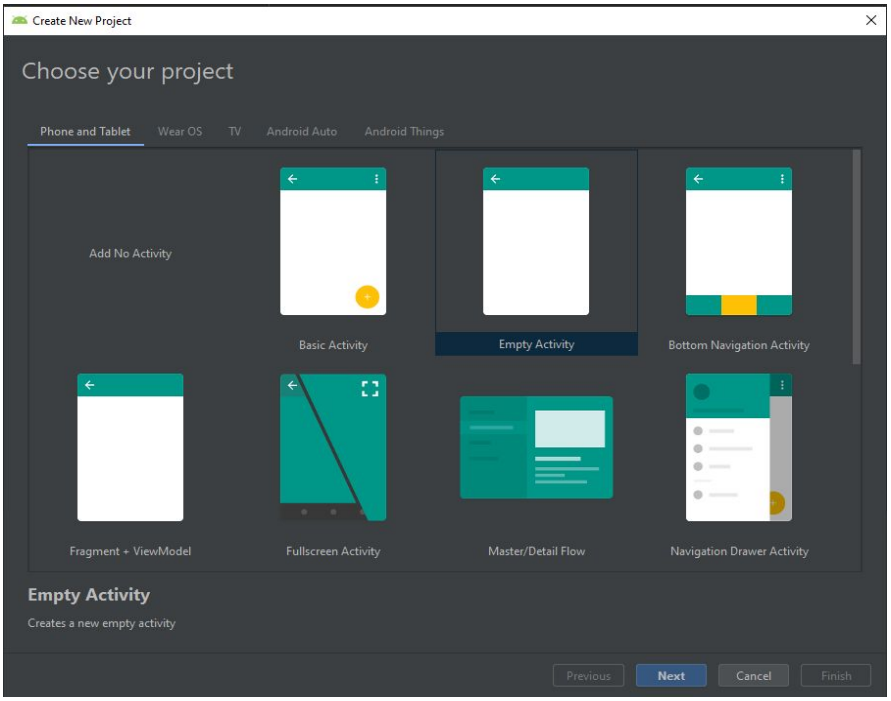
Εικόνα 2.7: Παράθυρο καλωσορίσματος Android Studio

Για να κάνουμε διαμόρφωση του SDK, κάνουμε κλικ στο Configure όπου μας εμφανίζεται η παρακάτω οθόνη στην οποία μπορούμε να επεξεργαστούμε τα επιθυμητά SDKs.

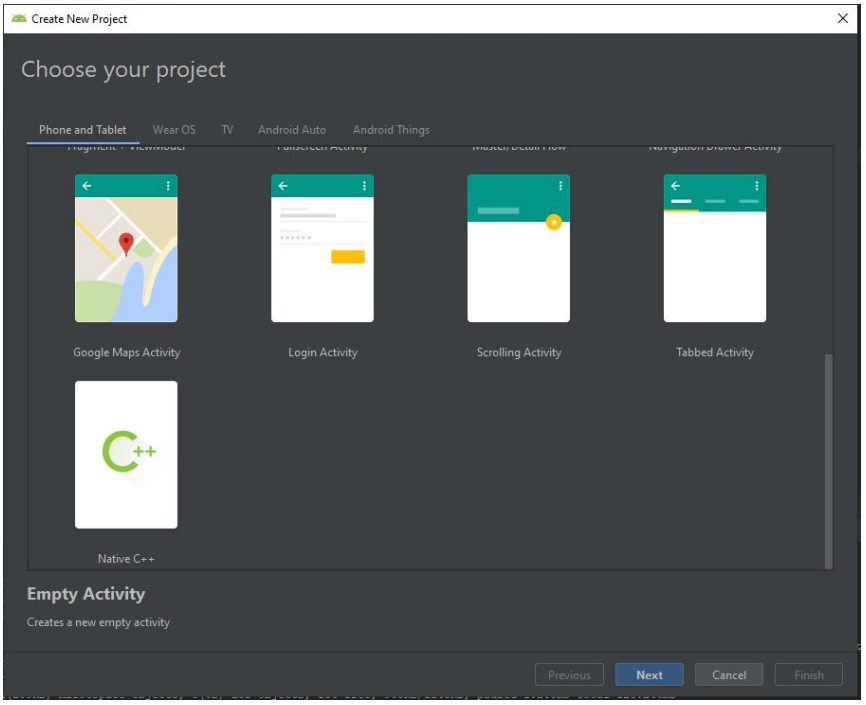


Εικόνα 2.8: Παράθυρο διαμόρφωσης του SDK

Για να ξεκινήσουμε τη δημιουργία ενος καινούργιου project, επιλέγουμε Start a new Android project στο παράθυρο καλωσορίσματος. Στη συνέχεια μας εμφανίζεται η παρακάτω οθόνη στην οποία θα επιλέξουμε τη βασική μορφή που θα έχει το Activity του καινούργιου μας project. Επίσης μπορούμε να επιλέξουμε το είδος της συσκευής που μπορεί να τρέξει η καινούργια μας εφαρμογή (Phone and Tablet, Wear OS, TV, Android Auto).



Εικόνα 2.9: Επιλογή επιθυμητού Activity (α)



Εικόνα 2.10: Επιλογή επιθυμητού Activity (β)

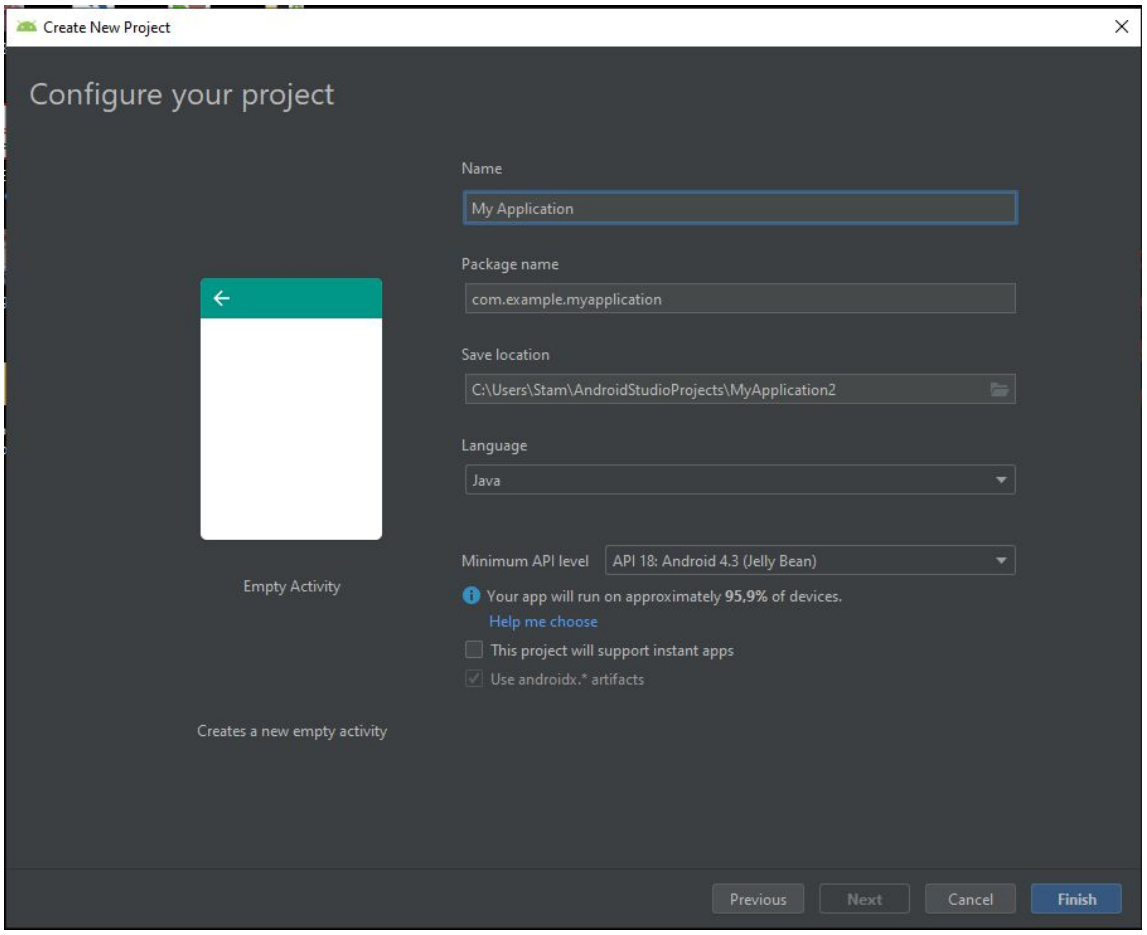
Είδη activities:

- Basic Activity - Δημιουργεί μία blank activity με μία app bar.
- Empty Activity - Δημιουργεί μία κενή δραστηριότητα.
- Bottom navigation Activity - Δημιουργεί μια navigation bar στο κάτω μέρος της οθόνης.
- Fullscreen Activity - Η full-screen mode εναλλάσσεται με την ορατότητα της διεπαφής χρήστη όποτε ο χρήστης αγγίζει την οθόνη.
- Master/Detail Flow - Χωρίζει την οθόνη σε 2 περιοχές: Στην αριστερή ένα μενού και οι λεπτομέρειες του επιλεγέντος αντικειμένου στη δεξιά.
- Navigation Drawer Activity - Δημιουργεί μια νέα δραστηριότητα με ενα πάνελ στο αριστερό μέρος της οθόνης στο οποίο επιτρέπει στους χρήστες να διαλέξουν επιθυμητές ενέργειες.
- Google Maps Activity - Δημιουργεί μια καινούργια δραστηριότητα με Google Maps.
- Login Activity - Επιτρέπει στους χρήστες να κάνουν login ή εγγραφή με ένα e-mail και ένα κωδικό πρόσβασης.
- Scrolling Activity - Δημιουργεί μια δραστηριότητα για σκρολάρισμα.
- Tabbed Activity - Δημιουργεί μια κενη δραστηριότητα με tabs.
- Native C++ - Δημιουργεί ένα νέο project με μια κενή δραστηριότητα, διαμορφωμένη για JNI.

Για το συγκεκριμένο παράδειγμα θα επιλέξουμε μια empty Activity και θα δημιουργήσουμε μια εφαρμογή για smartphones και tablet.

Έχοντας επιλέξει το είδος της δραστηριότητας, μας εμφανίζεται το παρακάτω παράθυρο στο οποίο θέτουμε το όνομα του project (Name), το όνομα πακέτου (Package name) για το μοναδικό αναγνωριστικό της εφαρμογής, τη τοποθεσία αποθήκευσης στον υπολογιστή (Save Location), τη γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε, καθώς και το επίπεδο API που θα εργαστούμε.

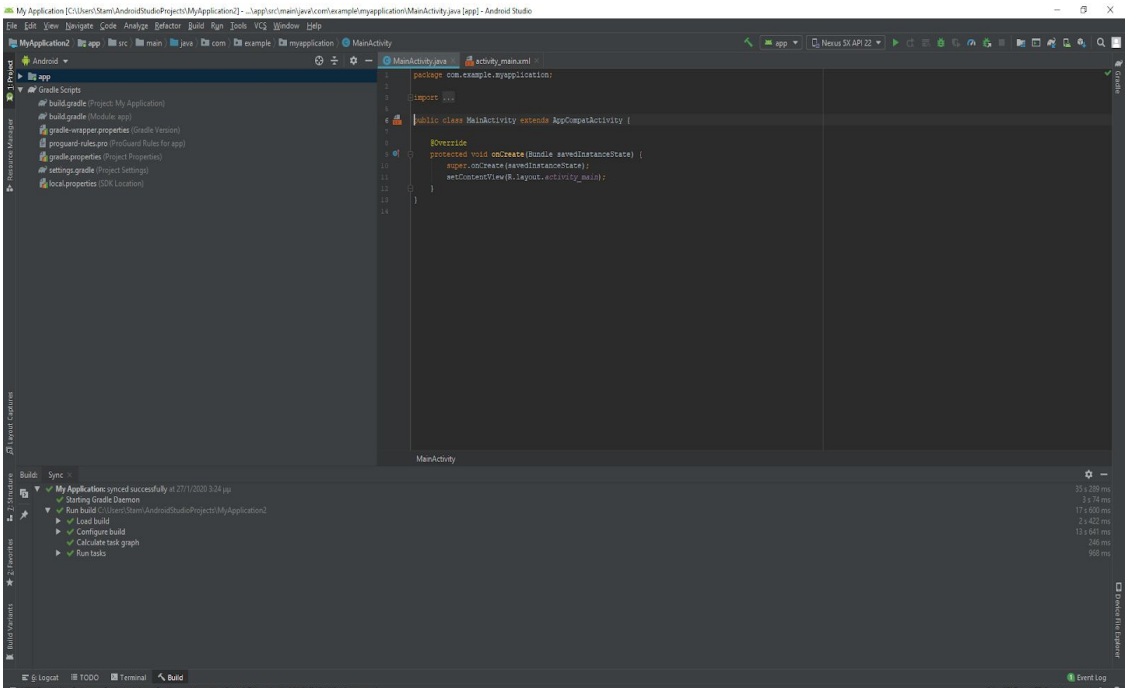
Για το παράδειγμα μας θα χρησιμοποιήσουμε τη γλώσσα Java και API 4.3 Jelly Bean



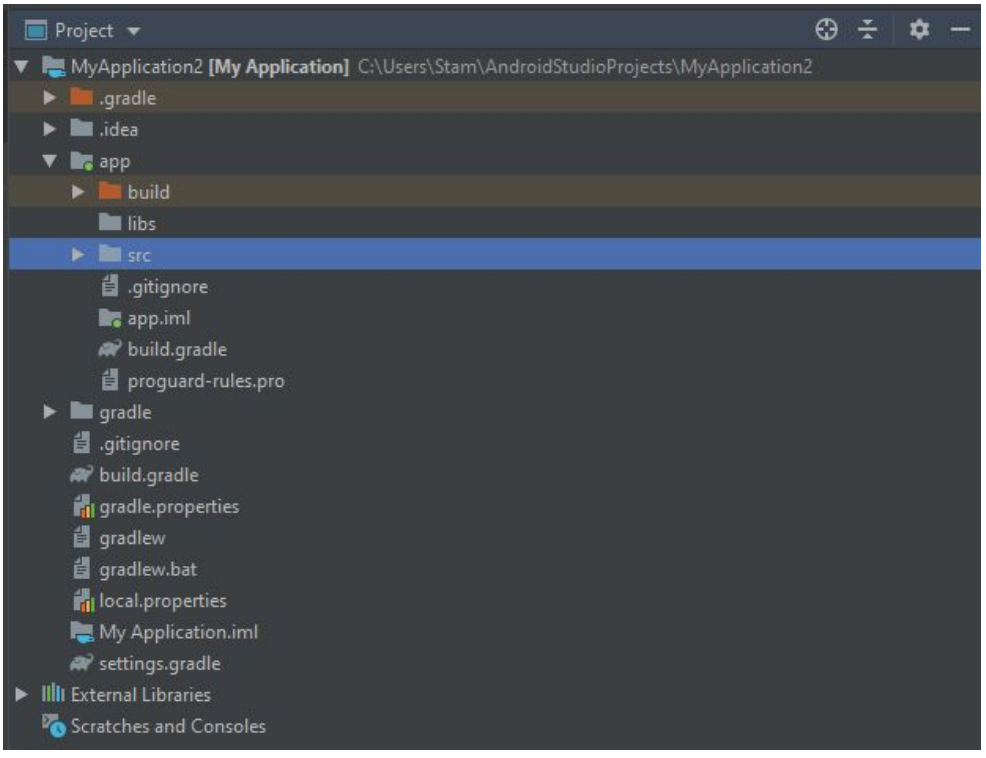
Εικόνα 2.11: Επεξεργασία στοιχείων του καινούργιου project.

Στην Εικόνα 2.12 βλέπουμε το περιβάλλον εργασίας του Android Studio.

Η δομή του νέου μας project μαζί με όλους τους υποφακέλους του και τα συσχετιζόμενα αρχεία (αρχείο AndroidManifest.xml) βρίσκεται στο αριστερό πλαίσιο του περιβάλλοντος εργασίας. Στο κεντρικό πλαίσιο βλέπουμε τον code editor του Android Studio.



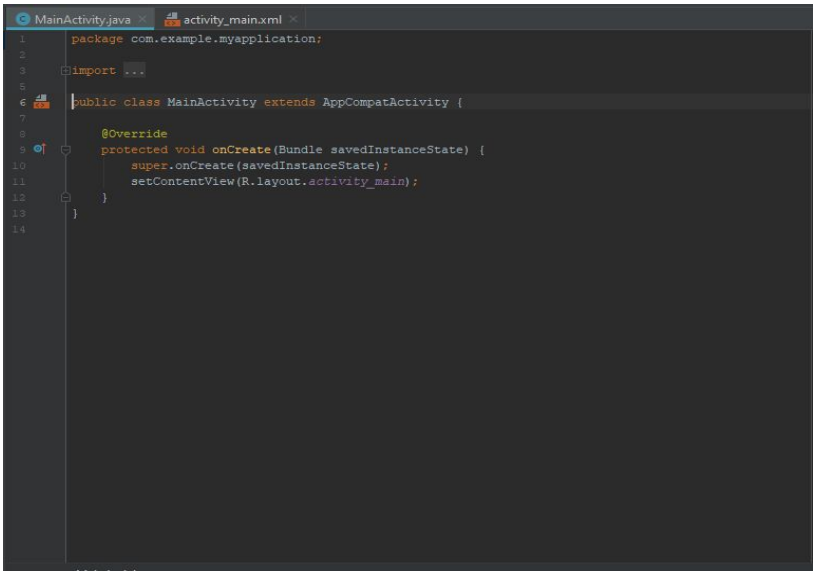
Εικόνα 2.12: Το περιβάλλον εργασίας του Android Studio.



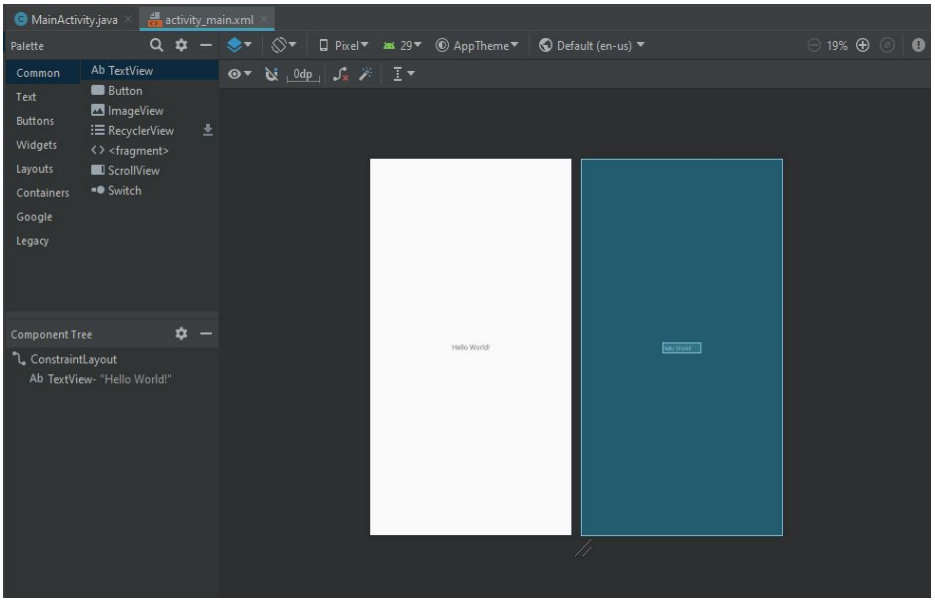
Εικόνα 2.13: Δομή και φάκελοι του project.

Στον code editor βλέπουμε πως υπάρχουν δύο ανοιχτά αρχεία. Το MainActivity.java είναι το αρχείο του project στο οποίο βρίσκεται ο πηγαίος κώδικας

της εφαρμογής μας, ενω το activity_main.xml είναι το αρχείο με βάση το οποίο μπορούμε να επεξεργαστούμε το γραφικό περιβάλλον της εφαρμογής μας.



Εικόνα 2.14: Το αρχείο MainActivity.java.



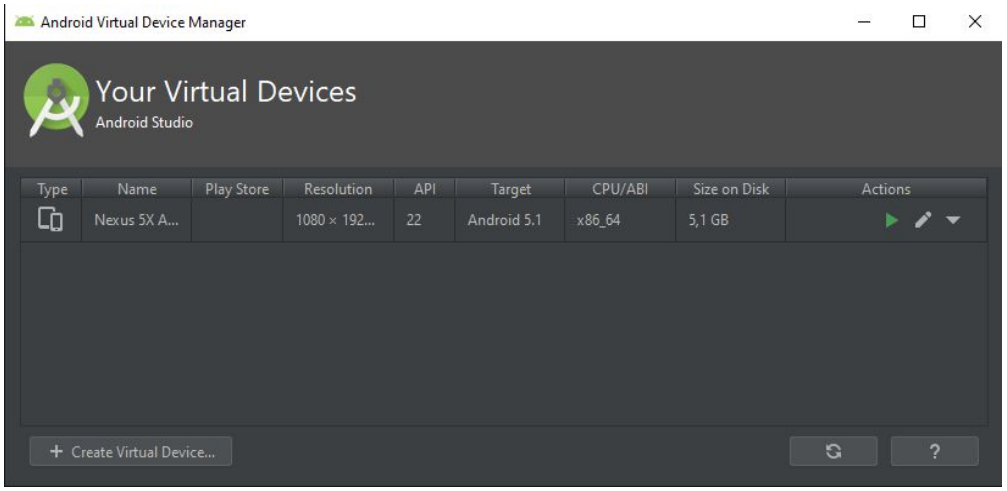
Εικόνα 2.15: Το αρχείο activity_main.xml.

Στο activity_main.xml, το Android Studio μας δίνει τη δυνατότητα να μορφοποιήσουμε την εφαρμογή μας, μέσω μιας παλέτας από εργαλεία:

- Text: Πεδία όπου ο χρήστης μπορεί να πληκτρολογήσει κείμενο. Η διαφορά μεταξύ τους είναι ο τύπος κειμένου που ο χρήστης μπορεί να εισάγει.
- Buttons: Η κατηγορία αυτή περιέχει είδη κουμπιών, check box, radio buttons, switch buttons κ.α.
- Widgets: Η κατηγορία αυτή περιέχει τα πιο κοινά συστατικά των περισσότερων layouts: Buttons, checkboxes, text views, switches, image views, progress bars, spinners, web views.
- Layouts: Ένα layout είναι ένα αντικείμενο container για την κατανομή των συστατικών στην οθόνη. Το αρχικό στοιχείο (root element) μιας διεπαφής χρήστη είναι ένα layout object, αλλά τα layouts μπορούν επίσης να περιέχουν περισσότερα layouts, δημιουργώντας έτσι μία ιεραρχία συστατικών δομημένη σε layouts.
- Containers: Τα containers ομαδοποιούν συστατικά με κοινή συμπεριφορά: list views, scroll views, tab hosts κ.α.
- Google: Περιέχει το AdView και το MapView της Google.

Το Android Studio μας δινει την δυνατότητα μέσω μιας εικονικής μηχανής/συσκευής να κάνουμε δοκιμές πάνω στην εφαρμογή μας με τη χρήση του AVD Manager (Android Virtual Device Manager).

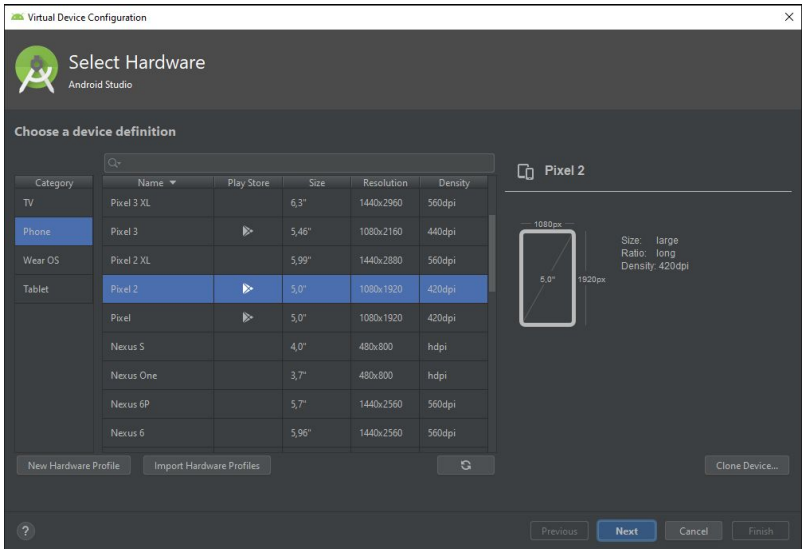
Για να επιλέξουμε το είδος της εικονικής μηχανής/συσκευής στο οποίο θέλουμε να τρέξουμε την εφαρμογή μας, επιλέγουμε από την γραμμή εργαλείων Tools/AVD Manager (Android Virtual Device Manager) και στη συνέχεια μας εμφανίζεται το παρακάτω παράθυρο:



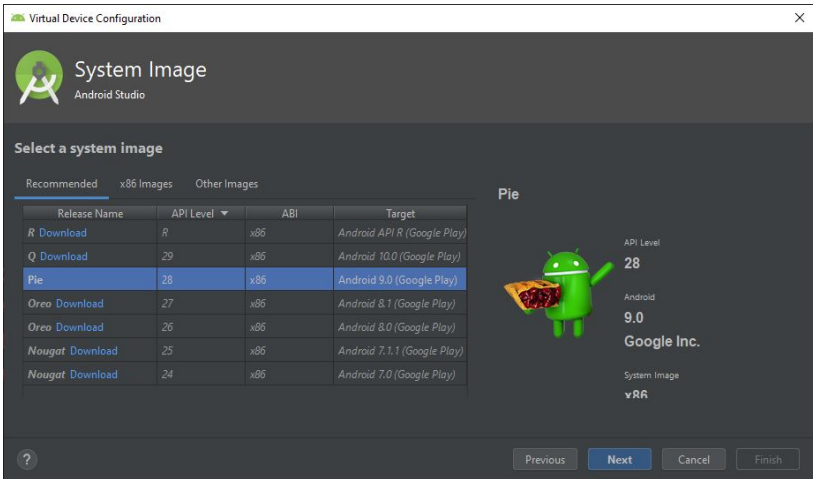
Εικόνα 2.16: Παράθυρο δημιουργίας νέας εικονικής μηχανής.

Ο AVD Manager εμφανίζει την λίστα των διαμορφωμένων εικονικών συσκευών. Αν δεν υπάρχουν τοτε πατώντας το Create Virtual Device, που βρίσκεται κάτω αριστερά στο παράθυρο του AVD Manager, μας δίνεται η δυνατότητα να δημιουργήσουμε μια νέα εικονική συσκευή.

Εφόσον θέλουμε να δημιουργήσουμε μια νέα εικονική συσκευή πρέπει πρώτα να επιλέξουμε το είδος της συσκευής που θέλουμε να τρέξει η εφαρμογή μας (TV, Phone, Wear OS, Tablet) αλλά και το επίπεδο API και επιθυμητή έκδοση Android.



Εικόνα 2.17: Επιλογή εικονικού μοντέλου και επίπεδο API της εικονικής μηχανής (α).

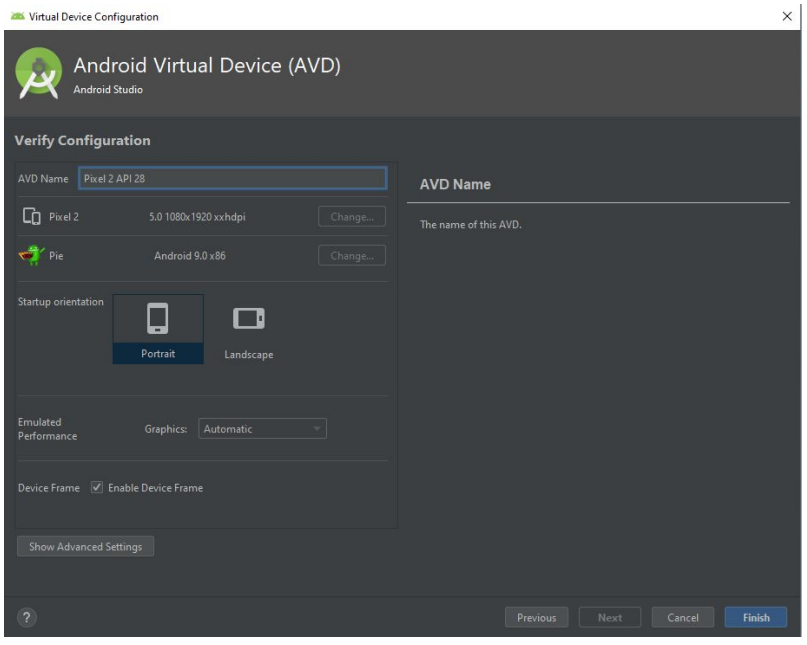


Εικόνα 2.17: Επιλογή εικονικού μοντέλου και επίπεδο API της εικονικής μηχανής (β).

Στη συνέχεια μας εμφανίζεται το παρακάτω παράθυρο στο οποίο μπορούμε να ονομάσουμε την εικονική συσκευή μας αλλά και να κάνουμε εξατομίκευση και παραμετροποίηση όπως ποιά κάμερα θα χρησιμοποιεί ο emulator (Front Camera, Front Camera), διαμόρφωση μνήμης, αν θα είναι σε κατάσταση portrait ή landscape κ.α.

Κάποιες από τις παραπάνω επιλογές φαίνονται όταν πατήσουμε το κουμπί “Show Advanced Settings”.

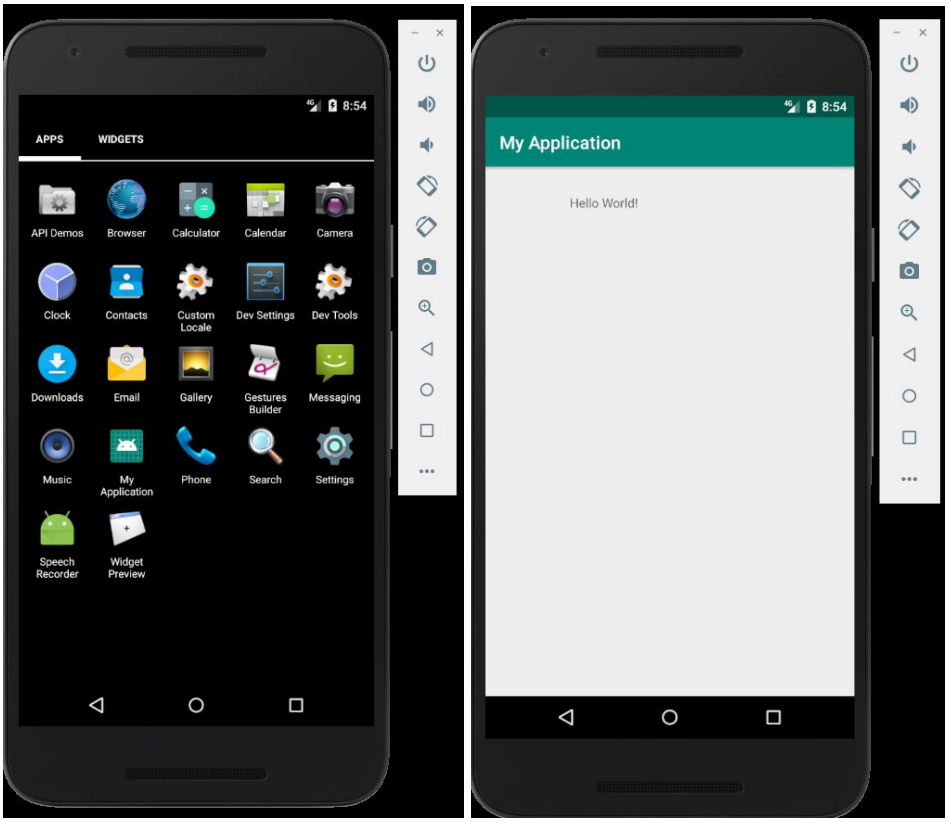
Κάνοντας κλικ στο finish ολοκληρώνουμε τη διαδικασία της δημιουργίας της εικονικής μας συσκευής.



Εικόνα 2.18: Παράθυρο επεξεργασίας στοιχείων εικονικής μηχανής

Εφόσον έχουμε φτιάξει και την εικονική μας συσκευή είμαστε έτοιμοι να δοκιμάσουμε την εφαρμογή μας και να ελέγξουμε πως τρέχει σε ένα περιβάλλον Android. Για να τρέξουμε την εφαρμογή μας κάνουμε κλικ στο Run απο τη γραμμή εργαλείων και στη συνέχεια Run app.

Αφού πρώτα γίνει η αποσφαλμάτωση για τον έλεγχο λαθών στον κώδικα της εφαρμογής μας , θα ξεκινήσει και ο emulator που θα μας εμφανίσει την εικονική μηχανή που δημιουργήσαμε και την εφαρμογή που αναπτύξαμε.



Εικόνα 2.19: Η εικονική μηχανή τρέχει την εφαρμογή μας.

3 ΡΟΜΠΟΤ ΚΑΙ ΡΟΜΠΟΤΙΚΗ

3.1 Εισαγωγή στη ρομποτική

Η Ρομποτική είναι ο σύγχρονος τεχνολογικός κλάδος της αυτοματοποίησης που ασχολείται με τη σύλληψη, τη σχεδίαση, την κατασκευή, τη λειτουργία, τη θεωρία και τις εφαρμογές των ρομπότ στην καθημερινότητα του ανθρώπου. Θεμέλια της ρομποτικής τεχνολογίας αποτελούν οι τεχνολογίες τού ψηφιακού ελέγχου και της τηλεχειρικής (tele- cherics) η οποία είναι μέθοδος χρήσης μηχανικών βραχιόνων για την εξ αποστάσεως εκτέλεση λεπτών χειρωνακτικών χειρισμών σε επικίνδυνα αντικείμενα ή σε επικίνδυνους για τον άνθρωπο χώρους.

Τα ρομπότ είναι μηχανές που δεν έχουν τη μορφή ή τη συμπεριφορά του ανθρώπου, αλλά μπορούν να εκτελούν εργασίες που μπορεί να κάνει και ένας άνθρωπος. Η χρήση τους λοιπόν έχει σκοπό την αντικατάσταση του ανθρώπου στην εκτέλεση εργασίας, η οποία αφορά τόσο το φυσικό επίπεδο, όσο και το επίπεδο λήψης αποφάσεων.

Εικόνα 3.1: Ρομποτικός βραχίονας

3.2 Ιστορία του ρομπότ και της ρομποτικής

Αναζητώντας κανείς τις ρίζες της ρομποτικής θα οδηγηθεί αρκετά χρόνια πίσω στην ιστορία της ανθρωπότητας. Στη μυθολογία γίνεται αναφορά σε ένα ον το οποίο είχε κατασκευάσει ο Ήφαιστος με το όνομα Τάλων, που ήταν ένας μεταλλικός γίγαντας από χαλκό, που ήταν φύλακας του νησιού της Κρήτης. Επίσης στην Ιλιάδα αναφέρεται η ύπαρξη κάποιων «ανθρωπόμορφων χρυσών σκλάβων» που βοηθούσαν τον Ήφαιστο, ο οποίος ήταν κουτσός, να περπατήσει. Ακόμη, ο Ήρων ο Αλεξανδρεύς, Έλληνας σοφός τού 1ου αιώνα, κατασκεύασε μεγάλο αριθμό αυτόματων μηχανισμών που βασιζόταν στις αρχές της στατικής και δυναμικής των ρευστών και στην κλασική μηχανική.

Η κατασκευή του πρώτου ρομπότ αποδίδεται στον μαθηματικό Αρχύτα τον Ταραντίνο (428-347 π.Χ.), ο οποίος αναφέρεται πως κατασκεύασε μια ιπτάμενη μηχανή. Η μηχανή κινούνταν με ατμό, μπορούσε να διανύσει μέχρι και 200 μέτρα και ονομαζόταν “περιστερά” ή “πετομηχανή”. Για αρκετούς αιώνες όμως δεν υπήρξαν αλλά σημάδια πρώιμης ρομποτικής.

Η λέξη ρομπότ εμφανίστηκε για πρώτη φορά σε ένα θεατρικό έργου επιστημονικής φαντασίας από τον Τσέχο σκηνοθέτη Karel Capek 1921 και προέρχεται από τη σλαβική λέξη robota που σημαίνει «καταναγκαστική εργασία» ή «αγγαρεία».

Το 1942 ο συγγραφέας επιστημονικής φαντασίας Isaac Asimov, στο χρονογράφημά του Runaround, εισάγει τη λέξη Robotics και ορίζει ως ρομπότ ένα αυτόματο με εμφάνιση ανθρώπου αλλά απαλλαγμένο από συναισθήματα.

Η μεγαλύτερη συνεισφορά του Asimov, πέρα από την εισαγωγή της έννοιας της Ρομποτικής, είναι η δημιουργία των τριών βασικών νόμων των ρομπότ:

- Νόμος 1: Ένα ρομπότ δεν πρέπει ποτέ να βλάψει έναν άνθρωπο ή λόγω αδράνειάς του να αφήσει ένα άνθρωπο να πάθει κακό.

- Νόμος 2: Ένα ρομπότ πρέπει πάντοτε να υπακούει τις εντολές που δίνονται από τους ανθρώπους εκτός και αν συγκρούονται με ανώτερο νόμο.

- Νόμος 3: Ένα ρομπότ πρέπει να προστατέψει την ύπαρξη του εκτός και αν αυτό συγκρούεται με ένα ανώτερο νόμο.

Ίσως το πιό δημοφιλείς ρομπότ αναπτύχθηκε το 1960 με όνομα “Unimate”, και ήταν ικανό να κινήσει το ένα του χέρι πρός διάφορες κατευθύνσεις και να ανοιγοκλείνει την παλάμη του.

Εννέα χρόνια μετά αναπτύχθηκε από σπουδαστή μηχανικής ο ρομποτικός βραχίονας με όνομα “Stanford” το οποίο ήταν το πρώτο ρομπότ ελεγχόμενο από ηλεκτρονικό υπολογιστή.

Το 1972, το Ιαπωνικό πανεπιστήμιο Waseda ολοκλήρωσε την ανάπτυξη του πρώτου ανθρωποειδές ρομπότ με όνομα “WABOT-1” και είχε την ικανότητα να περπατάει με τα κάτω άκρα του, να κρατάει και να μεταφέρει αντικείμενα με τα χέρια του, να υπολογίζει αποστάσεις και κατευθύνσεις μεταξύ αντικειμένων καθώς και να συμμετάσχει σε μια συνομιλία με ένα άτομο στα Ιαπωνικά.

Στη σημερινή ημέρα συναντάμε ρομπότ παντού στη καθημερινή μας ζωή. Ενα απο τα πιό δημοφιλή ρομπότ με πάνω από δύο εκατομμύρια πωλήσεις είναι το Roomba, το οποίο κινείται αυτόνομα και σκουπίζει από όπου περνά.

3.3 Ο ορισμός του ρομπότ

Ο Διεθνής Οργανισμός Προτύπων (International Standards Organization - ISO) δίνει τον εξής ορισμό για το ρομπότ: Είναι ένας αυτόματος, σερβοελεγχόμενος, ελεύθερα προγραμματιζόμενος, πολλών εφαρμογών χειριστής, με αρκετούς άξονες, για τη διαχείριση αντικειμένων, εργαλείων ή ειδικών συσκευών. Μεταβλητά προγραμματιζόμενες ενέργειες καθιστούν δυνατή την εκτέλεση πολλαπλών έργων.

3.3.1 Τα βασικά χαρακτηριστικά ενός ρομπότ

Υπάρχουν πολλοί και διάφοροι τύποι ρομπότ με διαφορετικά χαρακτηριστικά μεταξύ τους, όμως συνήθως έχουν τα εξής τρία βασικά χαρακτηριστικά:

- Ένα μηχανολογικό υποσύστημα, το οποίο δίνει τη δυνατότητα στο ρομπότ για εκτέλεση ενος έργου. Το υποσύστημα αυτό περιλαμβάνει μηχανισμούς που επιτρέπουν στο ρομπότ να κινείται όπως αρθρώσεις, κινητήρες, οδηγούς κλπ.

- Ένα υποσύστημα αισθητήρων, μέσω του οποίου το ρομπότ συγκεντρώνει πληροφορίες και δεδομενα για την κατάσταση στην οποία βρίσκονται τόσο το ίδιο όσο και το περιβάλλον. Το υποσύστημα αυτό είναι υπεύθυνο για την αποδοχή των εξωτερικών εντολών, την επεξεργασία τους, τη μετάφρασή τους σε ηλεκτρική ισχύ που θα δοθεί στους κινητήρες του ρομπότ, καθώς επίσης και για την παραγωγή σημάτων εξόδου τα οποία θα παρέχουν πληροφορίες για την κατάσταση του συστήματος. Στο υποσύστημα αισθητήρων περιλαμβάνονται όργανα μετρήσεως, αισθητήρες, ηλεκτρονικά στοιχεία κλπ.

- Ένα υποσύστημα επεξεργασίας/ελέγχου, που είναι το υποσύστημα στο οποίο γίνεται ο συνδυασμός των άλλων δύο υποσυστημάτων μαζί με διάφορες άλλες εντολές και δίνουν την δυνατότητα στο ρομπότ να είναι αποτελεσματικό στην λειτουργικότητα του.

3.3.2 Κατηγορίες και είδη ρομπότ

Οι σπουδαιότερες κατηγορίες ρομπότ είναι οι παρακάτω:

- Ρομπότ Σταθερής Βάσης: Αποτελούνται από ενώσεις που συνδέονται μέσω αρθρώσεων σχηματίζοντας μία κινηματική αλυσίδα, η οποία έχει το ένα άκρο της (βάση) σταθερά συνδεδεμένο με κάποιο σημείο του περιβάλλοντος χώρου. Η μορφή αυτή ρομπότ είναι η μορφή ενός βιομηχανικού ρομποτικού βραχίονα, και περιλαμβάνει το βραχίονα, τον καρπό και το εργαλείο.

- Κινούμενα Ρομπότ: ως κινούμενα ρομπότ χαρακτηρίζονται όλα εκείνα τα ρομπότ που έχουν τη δυνατότητα να μετακινήσουν όλα τα σημεία του μηχανισμού τους. Αυτό επιτυγχάνεται μέσω κάποιων ειδικών συστημάτων προώθησης, τα οποία μπορεί να είναι είτε απλά (όπως τροχοί) είτε

πολύπλοκα (μηχανικά πόδια). Ανάλογα με το βαθμό αυτονομίας τους, τα κινούμενα ρομπότ διακρίνονται στις εξής κατηγορίες:

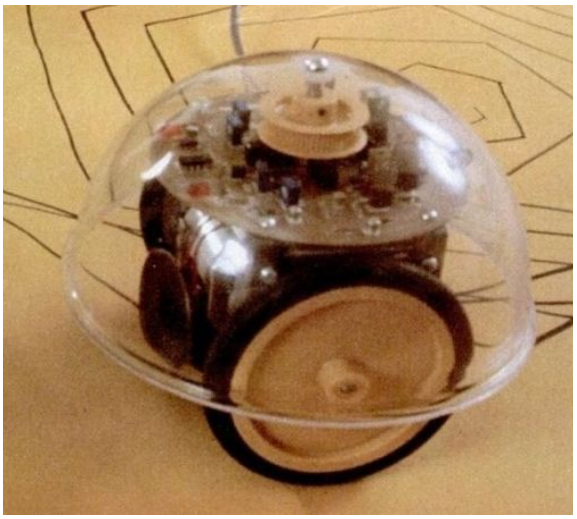
- AGVs: τα AGVs (Automatic Guided Vehicles) έχουν περιορισμένη αυτονομία κίνησης, δεδομένου ότι η τροχιά τους είναι προκαθορισμένη μέσω καλωδίων στο έδαφος ή πομπών στον περιβάλλοντα χώρο

- Αυτόνομα Έντροχα Ρομπότ: τα ρομπότ αυτά λειτουργούν με αρκετά υψηλό βαθμό αυτονομίας. Πιο συγκεκριμένα μπορούν και να λειτουργούν χωρίς κάποια συνεχή εξωτερική επίβλεψη και είναι ικανά να εκτελούν εργασίες αυτόνομα δεχόμενα μόνο ορισμένες υψηλού επιπέδου εντολές.

- Βαδίζοντα Ρομπότ: τα ρομπότ αυτά χρησιμοποιούν μηχανικά πόδια για την κίνησή τους. Με αυτό το τρόπο επιτυγχάνεται η δυνατότητα αποφυγής εμποδίων και η ικανότητα αναρρίχησης σε ανώμαλα εδάφη και μη επίπεδες επιφάνειες. Από τα πιο συνηθισμένα ρομπότ αυτής της κατηγορίας είναι τα δίποδα καθώς και εφαρμογές με περισσότερα από δύο πόδια.

Άλλα είδη ρομπότ που αξίζει να αναφερθούν είναι τα εξης:

- Εκπαιδευτικά ρομπότ (Educational robot): Είναι ρομπότ που χρησιμοποιούνται ως εκπαιδευτικοί βοηθοί σε σχολεία. Από τα πιο γνωστά είναι τα turtles τα οποία είχαν χρησιμοποιηθεί σε σχολεία και προγραμματίζονταν χρησιμοποιώντας τη γλώσσα Logo.



Εικόνα 3.1: Educational Robot

- Αρθρωτά ρομπότ (Modular robot): Τα αρθρωτά ρομπότ είναι μια νέα γενιά ρομπότ που η σχεδίαση τους έχει βασιστεί στη modularizing αρχιτεκτονική και είναι περισσότερο λειτουργικά και αποτελεσματικά σε σχέση με τα συμβατικά ρομπότ. Η αρχιτεκτονική τους δομή επιτρέπει να σχεδιάζονται με περισσότερους από 8 βαθμούς ελευθερίας ενώ προγραμματισμός τους είναι πιο περίπλοκος από ότι στα παραδοσιακά.

Εικόνα 3.3: Modular Robot

- Συνεργατικά ή συλλογικά ρομπότ (Collaborative Robots): Ένα συλλογικό ρομπότ ή cobot είναι ένα ρομπότ που μπορεί να

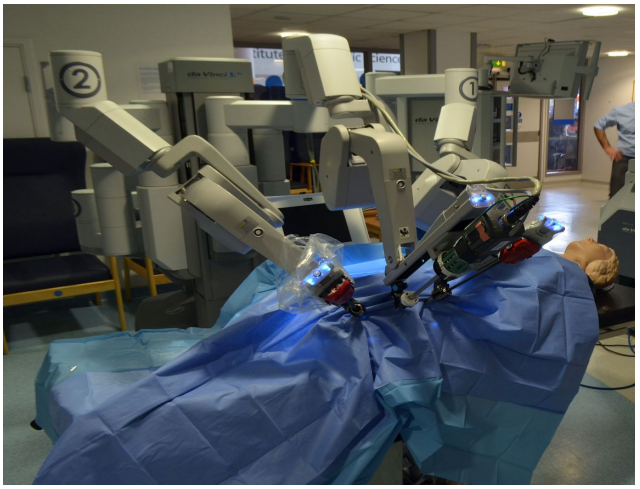
αλληλοεπιδράσει με το ανθρώπινο δυναμικό έτσι ώστε επιτευχθεί με ασφάλεια και αποτελεσματικότητα η εκτέλεση απλών αλλά και πιο πολύπλοκων βιομηχανικών εργασιών. Ωστόσο, απολήξεις και άλλες περιβαλλοντικές συνθήκες ενδέχεται να δημιουργήσουν κινδύνους, έτσι η αξιολόγηση του κινδύνου πρέπει να γίνεται πριν από τη κάθε χρήση της εφαρμογή όπως έλεγχος ορθής κίνησης.

Εικόνα 3.4: Collaborative Robot

- Στρατιωτικά ρομπότ (Military Robots): Τα στρατιωτικά ρομπότ είναι αυτόνομες ή τηλεχειριζόμενες συσκευές που έχουν σχεδιαστεί για στρατιωτικές εφαρμογές ή υπηρεσίες. Τέτοια ρομπότ χρησιμοποιούνται από πολλές στρατιωτικές υπηρεσίες παγκοσμίως.

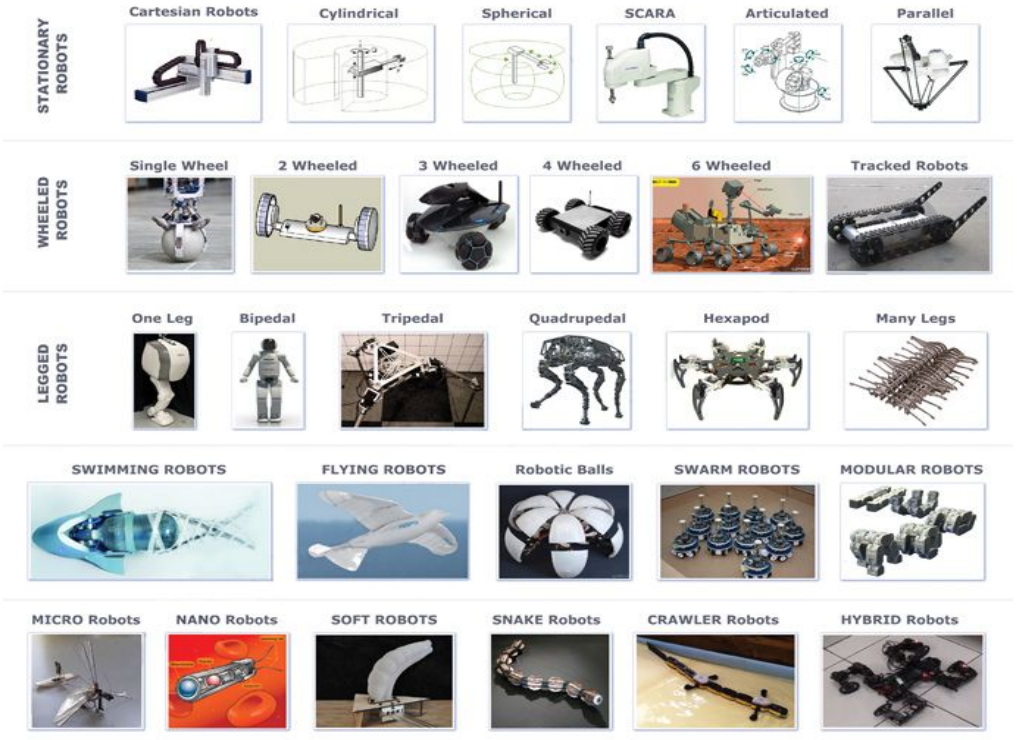
Εικόνα 3.5: Military Robot

- Ρομπότ στην υγεία και την ιατρική (Healthcare and Medical Robots): Τα ρομπότ στον τομέα της υγείας έχουν δύο βασικές λειτουργίες, εκείνα που βοηθούν ένα άτομο, όπως ένα πάσχοντα από μια ασθένεια και εκείνα που βοηθούν σε συνολικά συστήματα όπως τα φαρμακεία και τα νοσοκομεία.



Εικόνα 3.6: Medical Robot

Στη παρακάτω εικόνα βλέπουμε διάφορα είδη ρομπότ:



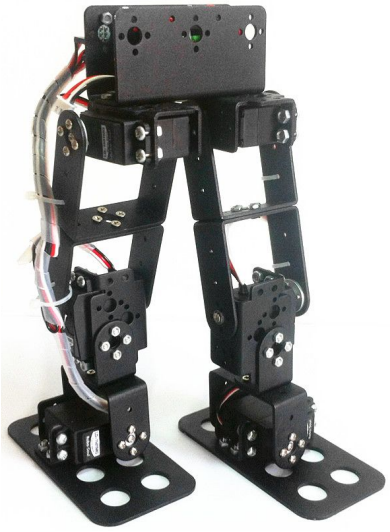
Εικόνα 3.7: Είδη ρομπότ

3.4 Το δίποδο ρομπότ

Στην εργασία αυτή, εστιάζουμε στα δίποδα ρομπότ και πώς μπορεί κάποιος με ευκολία να αναπτύξει μια βασική εφαρμογή αυτού. Όπως έχει αναφερθεί και προηγουμένως, υπάρχουν τρία βασικά χαρακτηριστικά ενός ρομπότ, δεν εξαιρούνται φυσικά και τα δίποδα τα οποία μεταξύ άλλων έχουν αισθητήρες, μικροελεγκτή και μηχανικό τμήμα.

Εδώ δεν έχουμε αισθητήρες αφού ο χειρισμός γίνεται από χρήστη με εφαρμογή android και όχι αυτόνομα από το ρομπότ, για μικροελεγκτή έχουμε ένα arduino uno, η οποία είναι η πιό δημοφιλείς και προσβάσιμη πλατφόρμα για μικροελεγκτές σε ένα πρότζεκτ και τα μηχανικά εξαρτήματα καθώς και οι σερβοκινητήρες είναι από ένα κίτ για ανθρωποειδή ρομπότ.

Οι βαθμοί ελευθερίας στο δικό μας ρομπότ είναι οκτώ και βρίσκονται στον γοφό με κίνηση εμπρός και πίσω, στο γόνατο εμπρός και πίσω, και στον αστράγαλο πάλι εμπρός και πίσω, καθώς και δεξιά - αριστερά.



Εικόνα 3.8: Ενα δίποδο (biped) ρομπότ

3.5 Ρομπότ στη κοινωνία

Στη σημερινή κοινωνία υπάρχουν ανησυχίες σχετικά με την αυξανόμενη χρήση

των ρομπότ. Τα ρομπότ έχουν κατηγορηθεί για την αύξηση της ανεργίας, καθώς χρησιμοποιούνται στην αντικατάσταση των εργαζομένων σε όλο και μεγαλύτερο αριθμό εργασιών. Επίσης η χρήση των ρομπότ για στρατιωτικούς σκοπούς έχει δημιουργήσει ηθικά ζητήματα.

4 BLUETOOTH

Το Bluetooth είναι μια ασύρματη τηλεπικοινωνιακή τεχνολογία μικρών αποστάσεων, η οποία μπορεί να μεταδώσει σήματα μέσω μικροκυμάτων σε ψηφιακές συσκευές.

4.1 Εισαγωγή στην επικοινωνία Bluetooth

Το Bluetooth είναι ένα πρωτόκολλο το οποίο παρέχει προτυποποιημένη, ασύρματη επικοινωνία ανάμεσα σε PDA (personal digital assistant), κινητά τηλέφωνα, φορητούς υπολογιστές, προσωπικούς υπολογιστές, εκτυπωτές, καθώς και ψηφιακές φωτογραφικές μηχανές ή ψηφιακές κάμερες, μέσω μιας ασφαλούς, φθηνής και παγκοσμίως διαθέσιμης χωρίς ειδική άδεια ραδιοσυχνότητας μικρής εμβέλειας. Από τεχνικής άποψης το Bluetooth είναι ένα πρωτόκολλο ασύρματης δικτύωσης σε φυσικό επίπεδο, υποεπίπεδο MAC και, προαιρετικά, υποεπίπεδο LLC.

Εικόνα 4.1: Το λογότυπο του Bluetooth

4.1.1 Τρόπος λειτουργίας Bluetooth

Για να μπορέσουν δύο συσκευές bluetooth να ανταλλάξουν μεταξύ τους δεδομένα, απαιτείται πρώτα να δημιουργηθεί ένα κανάλι επικοινωνίας μέσω της διαδικασίας pairing. Η μία συσκευή που βρίσκεται σε ανιχνεύσιμη λειτουργία είναι διαθέσιμη πρός εισερχόμενα αιτήματα σύνδεσης.

Στη συνέχεια μια δεύτερη συσκευή βρίσκει την πρώτη μέσω της διαδικασίας ανίχνευσης υπηρεσιών και μόλις η πρώτη συσκευή αποδεχθεί το αίτημα σύνδεσης της δεύτερης, τότε δημιουργείται το λεγόμενο bonding process των δύο συσκευών

στο οποίο ανταλλάσουν κλειδιά ασφάλειας (security keys). Τα συγκεκριμένα κλειδιά αποθηκεύονται στη μνήμη cache για μετέπειτα χρήση. Πλέον οι δύο συσκευές μπορούν να ανταλλάξουν πληροφορίες και δεδομένα.

Όταν οι δύο συσκευές λήξουν την επικοινωνία τους, η δεύτερη συσκευή απελευθερώνει το κανάλι στο οποίο έγινε η σύνδεση με την πρώτη συσκευή και παραμένουν bonded ώστε να μπορούν να συνδεθούν αυτόματα μελλοντικά σε περίπτωση που χρειαστεί να ανταλλάξουν πάλι δεδομένα.

4.1.2 Η δομή του Bluetooth

Η τεχνολογία Bluetooth, είναι ένα πρωτόκολλο με βάση τα πακέτα και αρχιτεκτονική Master/Slave. Ένα Master μπορεί να συνδεθεί με έως και επτά (7) Slaves σε ένα piconet (ασύρματο δίκτυο που συνδέει συσκευές bluetooth μεταξύ τους) και κάθε συσκευή που βρίσκεται μέσα σε αυτό το piconet χρησιμοποιούν το clock που δίνεται από το Master για την ανταλλαγή πακέτων.

4.2 Το Bluetooth στις Android εφαρμογές

Η πλατφόρμα Android υποστηρίζει την τεχνολογία bluetooth, η οποία δίνει την δυνατότητα σε μια συσκευή να ανταλλάσει δεδομένα με άλλες συσκευές bluetooth. Αναπτύσσεται και είναι διαθέσιμο μέσω διαφόρων Bluetooth API’s με τα οποία γίνεται εύκολη η ασύρματη επικοινωνία μιας εφαρμογής με συσκευές.

Με τη χρήση αυτών των API μπορεί εύκολα μια εφαρμογή να πραγματοποιήσει τις ακόλουθες ενέργειες: σάρωση συσκευών bluetooth, ερώτηση προς τον bluetooth adapter για τυχόν συνδυασμένες συσκευές (device pairing), καθιέρωση RFCOMM channels, σύνδεση με άλλες συσκευές, αμφίδρομη μεταφορά δεδομένων μεταξύ συνδεδεμένων συσκευών καθώς και διαχείριση πολλαπλών συνδέσεων.

5 ΚΩΔΙΚΑΣ & ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

5.1 Γενική λειτουργία και εισαγωγή στο project

Το project αυτό, αποτελείται από τρία τμήματα, το ρομπότ, το arduino με το HC-06, και την εφαρμογή android.

Το ρομπότ μας, αποτελείται από 8 servo-κινητήρες, 4 για κάθε “πόδι”. Αυτό δίνει στο ρομπότ ουσιαστικά 8 βαθμούς ελευθερίας (degrees of freedom). Κάθε κινητήρας έχει τρία καλώδια, ένα για την τροφοδοσία (+), ένα για την γείωση (-) και ένα για την κίνηση περιστροφής το οποίο δέχεται παλμούς με εναλλασόμενο duty cycle, ή αλλιώς PWM (pulse width modulation).

Με το PWM ανάλογα με το πόσο duty cycle έχουμε, δηλαδή σε μία περίοδο πόσο ποσοστό βρίσκεται σε logic high και πόσο σε logic low, αλλάζει και η τοποθεσία του άξονα του κινητήρα. Εάν αυτή η εναλλαγή του PWM γίνει σε αρκετά γρήγορη συχνότητα, τότε έχουμε ομαλή κίνηση του κινητήρα. Συνδυάζοντας λοιπόν τους οκτώ κινητήρες στο ρομπότ με ένα arduino και τις εξόδους PWM αυτού, μπορούμε πλέον να ελέγξουμε πλήρως ένα ρομπότ με 8-DOF.

Σε αυτή τη φάση όμως συναντάμε ένα πρόβλημα, ο μόνος τρόπος να έχουμε εκκίνηση του ρομπότ και λήξη αυτής (περπατώ - σταματώ) είναι μέσω σειριακής επικοινωνίας με τον υπολογιστή μας. Αυτό περιορίζει τον χρήστη και για αυτόν τον λόγο επιλέγουμε να χρησιμοποιήσουμε μια ασύρματη τεχνολογία επικοινωνίας με την οποία θα μπορεί ο χρήστης να ελέγχει το ρομπότ από απόσταση και χωρίς να είναι συνδεδεμένο το arduino (άρα και κατ επέκταση και το ρομπότ μας) στον υπολογιστή. Ο πιο εύκολος, φθηνός, και προσβάσιμος τρόπος υλοποίησης αυτού, είναι η bluetooth επικοινωνία.

Έτσι, συνδέοντας ένα bluetooth module HC-06 με το arduino στις σειριακές του θύρες, μπορούμε να στέλνουμε δεδομένα ασύρματα στο arduino μας.

Ο τρόπος με τον οποίο θα στέλνουμε τα δεδομένα αυτά ασύρματα είναι με τη βοήθεια εφαρμογής android η οποία θα στέλνει μέσω bluetooth τα δεδομένα που χρειάζεται το arduino ώστε να δώσει και αυτό με τη σειρά του εντολές στους κινητήρες του ρομπότ ώστε να έχουμε την κίνηση που επιθυμούμε.

Επειδή μιλάμε για οκτώ κινητήρες servo, αρα και αρκετή ανάγκη για Amper, ήταν αναγκαία η εξωτερική πηγή τροφοδοσίας για τους κινητήρες. Επιλέξαμε ένα τροφοδοτικό στα 5V και 18A ώστε να έχουμε επαρκείς ρεύμα και ισχύς για τους κινητήρες. Για λόγους ευκολίας και οργάνωσης, αναπτύχθηκε μια αυτοσχέδια ασπίδα (shield) για το Arduino η οποία είναι φτιαγμένη πάνω σε μια διάτρητη πλακέτα και ουσιαστικά συνδέει τον θετικό (+) πόλο της τροφοδοσίας με κάθε ακροδέκτη απο τους κινητήρες, τις αντίστοιχες γειώσεις (-) αυτών αλλά και ταυτόχρονα τη γείωση του Arduino, καθώς και τους ακροδέκτες απο το Arduino που ευθύνονται για την αποστολή PWM για να κινηθούν. Το Bluetooth module HC-06 συνδέεται απευθείας πάνω στο Arduino.

5.2 Κώδικας Arduino

Εδώ θα αναλύσουμε τον κώδικα του Arduino και της γενικότερης κίνησης του ρομπότ.

Αρχικά, κάνουμε include το αρχείο Servo.h, πράγμα που θα μας επιτρέπει να κάνουμε χρήση των κλάσεων και συναρτήσεων αυτού του αρχείου. Στη περίπτωση μας, ώστε να ενεργοποιήσουμε και να κινήσουμε σερβο-κινητήρες.

Στη συνέχεια, ορίζουμε ενα <<instance>> της κλάσης Servo για κάθε κινητήρα, δίνοντας ένα αντιπροσωπευτικό όνομα σε καθεμία από αυτές. Έπειτα, ορίζουμε κάποιες Global μεταβλητές, τις οποίες θα αναλύσουμε παρακάτω ως προς τη χρήση τους.

```
#include <Servo.h>
Servo LeftTopServo;   //left top
Servo LeftMidlServo;  //left midl
Servo LeftMid2Servo;  //left mid2
Servo LeftBotServo;   //left bot

Servo RightTopServo;  //right top
Servo RightMidlServo; //right midl
Servo RightMid2Servo; //right mid2
Servo RightBotServo;  //right bot

int LeftTopPos, LeftMidlPos, LeftMid2Pos, LeftBotPos, RightTopPos, RightMidlPos, RightMid2Pos, RightBotPos = 0;

int LED = 13;
int userInput = 0;
int interrupt = 0;
bool walkingState = false;
```

Εικόνα 5.1: Servo.h include, Class instances και Global μεταβλητές

- Η συνάρτηση void setup() - Σε αυτή τη βασική για πρόγραμμα Arduino συνάρτηση, θα κάνουμε όλες τις αρχικοποιήσεις και τους ορισμούς περιφερειακών.
    1. *Serial.begin(9600)* - Ενεργοποιούμε την σειριακή επικοινωνία με συχνότητα 9600 baud.
    2. *servo.attach()* - Με τις αντιστοιχίες των ονομάτων που ορίσαμε προηγουμένως για κάθε κινητήρα, τωρα θα αντιστοιχήσουμε και κάποια pins στα οποία θα στέλνει το πρόγραμμα μας PWM για να κινήσουμε κάθε κινητήρα.
    3. *pinMode(LED, OUTPUT)* - Ορίζουμε το Pin με όνομα LED ώς έξοδος ώστε να μπορούμε να το αναβοσβήνουμε.
    4. Ακολουθεί ενα block κώδικα το οποίο είναι για να ορίσουμε ενα timer interrupt με συχνότητα 8kHz. Το interrupt θα χρησιμοποιείται για να διαβάζει το πρόγραμμα μας χωρίς κάποια ενόχληση ότι έρχεται στην σειριακή θύρα του Arduino.
    5. Τέλος έχουμε την εκτέλεση μιας συνάρτησης που θα δούμε παρακάτω, που δίνει αρχικές τιμές στους κινητήρες μας.

```
19 void setup() {
20   Serial.begin(9600);
21   LeftTopServo.attach(11); //left top
22   LeftMid1Servo.attach(10); //left mid1
23   LeftMid2Servo.attach(9); //left mid2
24   LeftBotServo.attach(8); //left bottom
25
26   RightTopServo.attach(7); //right top
27   RightMid1Servo.attach(6); //right mid1
28   RightMid2Servo.attach(5); //right mid2
29   RightBotServo.attach(4); //right bottom
30
31   pinMode(LED, OUTPUT);
32
33
34   cli();//stop interrupts
35
36   //set timer2 interrupt at 8kHz
37   TCCR2A = 0;// set entire TCCR2A register to 0
38   TCCR2B = 0;// same for TCCR2B
39   TCNT2  = 0;//initialize counter value to 0
40   // set compare match register for 8khz increments
41   OCR2A = 249;// = (16*10^6) / (8000*8) - 1 (must be <256)
42   // turn on CTC mode
43   TCCR2A |= (1 << WGM21);
44   // Set CS21 bit for 8 prescaler
45   TCCR2B |= (1 << CS21);
46   // enable timer compare interrupt
47   TIMSK2 |= (1 << OCIE2A);
48
49   sei();//allow interrupts
50
51 //init position
52   writeToServos(90, 90, 90, 90, 90, 90, 90, 90);
53 } //end setup
```

Εικόνα 5.2: void setup()

- *ISR(TIMER2_COMPA_vect)* - Εδώ έχουμε το τμήμα κώδικα που θα εκτελείται με συχνότητα 8kHz, σύμφωνα με το πως ορίσαμε το timer interrupt μας.
  Το πρόγραμμα ελέγχει αρχικά εάν είναι διαθέσιμη η σειριακή επικοινωνία, και αν είναι τότε θα εκτελεστούν και τα επόμενα. Σε αυτό το τμήμα ότι λαμβάνουμε στη σειριακή θύρα αποθηκεύεται σε μια μεταβλητή ‘interrupt’. Στη περίπτωση που αντιστοιχεί η τιμή της μεταβλητής με κάποια από τις επιθυμητές τιμές τότε δίνουμε αντίστοιχα τιμές στη μεταβλητή walkingState και userInput καθώς και ανάβουμε ή σβήνουμε το LED. Στο τέλος ότι τιμή έχουμε λάβει, την εκτυπώνουμε στην σειριακή κονσόλα του Arduino IDE.

```
57 ISR(TIMER2_COMPA_vect) { //timer1 interrupt 8kHz toggles pin 9
58   //generates pulse wave of frequency 8kHz/2 = 4kHz (takes two cycles for full wave- toggle high then toggle low)
59   if (Serial.available() > 0) { //checks if serial port is available for reading
60     interrupt = Serial.read(); //reads the serial input and stores in the 'interrupt' variable
61     if (interrupt == '0') { //if we receive a '0' on the serial input set the relevant variables to the reset state
62       walkingState = false;
63       userInput = 0;
64       digitalWrite(LED, HIGH); //LED ON for visibility/debugging so we know which state we are currently on
65     } else if (interrupt == '1') { //if we receive a '1' on the serial input set the relevant variables to the walking state
66       walkingState = true;
67       userInput = 1;
68       digitalWrite(LED, LOW); //LED OFF for visibility/debugging so we know which state we are currently on
69     }
70     Serial.print(userInput); //print 0 or 1 on the serial console for more visibility
71     Serial.print("\n");
72   }
73 }
```

Εικόνα 5.3: ISR(TIMER2_COMPA_vect)

- Η συνάρτηση void loop() - Εδώ έχουμε το κύριο μέρος του προγράμματός μας, που τρέχει σε ατέρμονη λούπα. Μέσα σε αυτή τη λούπα έχουμε ένα Switch - case, το οποίο ελέγχει την τιμή της μεταβλητής userInput και εάν είναι μια απο τις 2 επιθυμητές τιμές, θα εκτελεί αντίστοιχα τμήματα κώδικα. Για την εφαρμογή αυτή έχουμε δύο περιπτώσεις: κίνηση και παύση, άρα θα θέσουμε το 0 και το 1 ως περιπτώσεις (case).
    - *case 0:* Σε αυτή τη περίπτωση, έχουμε την κίνηση του ρομπότ μας. Αρχικά στέλνουμε στη συνάρτηση walk() τις συντεταγμένες (πρακτικά, μοίρες) που θέλουμε να δώσουμε στους κινητήρες για να μπει στην αρχική κατάσταση της κίνησης.

      Έπειτα μπαίνουμε σε μια ακόμη λούπα, η χρήση της οποίας είναι σε περίπτωση που καθ'όλη τη διάρκεια της κίνησης λάβουμε αλλαγή κατάστασης, να μπορέσουμε να σπάσουμε απο τη λούπα και να προχωρήσει το πρόγραμμα στην επιθυμητή κατάσταση.

      Μέσα σε αυτή τη λούπα γίνεται η κλήση της συνάρτησης η οποία αναλαμβάνει την κίνηση των κινητήρων ώστε να περπατήσει το ρομπότ μας.
    - *case 1:* Σε αυτή τη περίπτωση, έχουμε όπως και προηγουμένως μια λούπα που ελέγχει την αλλαγή κατάστασης, και μέσα σε αυτή μια χρονοκαθυστέρηση ενός δευτερολέπτου για παύση και ύστερα γίνεται

κλήση της συνάρτησης που θα φέρει τους κινητήρες, αρα και το ρομπότ, στην αρχική τους θέση.

```
75 void loop() {
76   switch (userInput) { //checks user input that arrives from the interrupt algorithm, which itself comes from the serial input
77     case 0: //robot is walking
78       walk(130, 170, 50, 90, 50, 10, 130, 90); //init position
79       while (!walkingState) { //this is for escaping the current state if interrupted
80         sillyWalk(); //walking algorithm
81       } break; //end while loop and break
82     case 1: //robot pauses for 1 second and resets it's motors to init position
83       while (walkingState) { //this is for escaping the current state if interrupted
84         delay(1000);
85         resetPosition(); //reset algorithm
86       } break;
87   }//switch case
88 } //main loop
```

Εικόνα 5.4: void loop()

- Η συνάρτηση void writeToServos() - Η συνάρτηση αυτή δέχεται 8 τιμές και τις στέλνει αντίστοιχα στους ανάλογους κινητήρες.

```
94 void writeToServos(int LeftTopPos, int LeftMidlPos, int LeftMid2Pos, int LeftBotPos, int RightTopPos, int RightMidlPos, int RightMid2Pos, int RightBotPos) {
95   LeftTopServo.write(LeftTopPos);
96   LeftMidlServo.write(LeftMidlPos);
97   LeftMid2Servo.write(LeftMid2Pos);
98   LeftBotServo.write(LeftBotPos);
99
100  RightTopServo.write(RightTopPos);
101  RightMidlServo.write(RightMidlPos);
102  RightMid2Servo.write(RightMid2Pos);
103  RightBotServo.write(RightBotPos);
104 }
```

Εικόνα 5.5: void writeToServos()

- Η συνάρτηση void readServoPosition() - Η συνάρτηση αυτή διαβάζει την τελευταία θέση των κινητήρων σύμφωνα με τις εντολές που έχουν δοθεί, και τις αποθηκεύει στις ανάλογες μεταβλητές.

```
106 void readServoPosition() {
107   LeftTopPos    =   LeftTopServo.read();
108   LeftMidlPos   =   LeftMidlServo.read();
109   LeftMid2Pos   =   LeftMid2Servo.read();
110   LeftBotPos    =   LeftBotServo.read();
111
112   RightTopPos   =   RightTopServo.read();
113   RightMidlPos  =   RightMidlServo.read();
114   RightMid2Pos  =   RightMid2Servo.read();
115   RightBotPos   =   RightBotServo.read();
116 }
```

Εικόνα 5.6: void readServoPosition()

- Η συνάρτηση void walk() - Η συνάρτηση αυτή δέχεται 8 μεταβλητές, μια για κάθε κινητήρα και αντιπροσωπεύει τις επιθυμητές μοίρες στις οποίες θέλουμε

ο κινητήρας μας να μετατοπιστεί. Διαβάζει την προηγούμενη θέση των κινητήρων με τη συνάρτηση readServoPosition() και εισερχόμαστε σε μια λούπα που τρέχει μέχρι να φτάσουν όλοι οι κινητήρες στην θέση που έχουμε ορίσει. Μέσα στη λούπα αυτή, γίνεται έλεγχος της προηγούμενης θέσης κάθε κινητήρα και αναλόγως προσθέτει ή αφαιρεί στην μεταβλητή που αντιστοιχεί σε κάθε κινητήρα.

Τέλος, με τη βοήθεια της συνάρτησης writeToServos(), στέλνουμε αυτές τις τιμές στους κινητήρες μαζί και μια χρονοκαθυστέρηση 20ms ώστε να γίνει η κίνηση από τη προηγούμενη τιμή προς την καινούργια ομαλά και όχι άμεσα.

```
127 void walk(int a, int b, int c, int d, int e, int f, int g, int h) {
128   readServoPosition();
129   while ((LeftTopPos != a || LeftMidlPos != b || LeftMid2Pos != c    || LeftBotPos != d ||
130           RightTopPos != e || RightMidlPos != f || RightMid2Pos != g  || RightBotPos != h)) {
131     if (LeftTopPos < a)LeftTopPos++;
132     else if (LeftTopPos > a)LeftTopPos--;
133     if (LeftMidlPos < b)LeftMidlPos++;
134     else if (LeftMidlPos > b)LeftMidlPos--;
135     if (LeftMid2Pos < c)LeftMid2Pos++;
136     else if (LeftMid2Pos > c)LeftMid2Pos--;
137     if (LeftBotPos < d)LeftBotPos++;
138     else if (LeftBotPos > d)LeftBotPos--;
139
140     if (RightTopPos < e)RightTopPos++;
141     else if (RightTopPos > e)RightTopPos--;
142     if (RightMidlPos < f)RightMidlPos++;
143     else if (RightMidlPos > f)RightMidlPos--;
144     if (RightMid2Pos < g)RightMid2Pos++;
145     else if (RightMid2Pos > g)RightMid2Pos--;
146     if (RightBotPos < h)RightBotPos++;
147     else if (RightBotPos > h)RightBotPos--;
148
149     writeToServos(LeftTopPos, LeftMidlPos, LeftMid2Pos, LeftBotPos, RightTopPos, RightMidlPos, RightMid2Pos, RightBotPos);
150     delay(10);
151   }
152 }
```

Εικόνα 5.7: void walk()

- Η συνάρτηση void resetPosition() και void sillyWalk() - αυτές οι δύο συναρτήσεις ευθύνονται για την θέση παύσης και την συνεχόμενη κίνηση του ρομπότ αντίστοιχα.

```
118 void resetPosition() {
119   walk(90, 90, 90, 90, 90, 90, 90, 90);
120 }
121 void sillyWalk() {
122   walk(130, 150, 50, 60, 30, 1, 130, 90);
123   walk(150, 169, 50, 90, 50, 20, 130, 120);
124 }
```

Εικόνα 5.8: void resetPosition() και void sillyWalk()

5.3 Κώδικας εφαρμογής Android

Σε αυτό το κεφάλαιο θα δούμε αναλυτικά τον κώδικα της εφαρμογής Android με βάση την οποία πετυχαίνουμε την κίνηση του ρομπότ.

5.3.1 Αρχείο activity_main.xml:

Σε αυτό το αρχείο καλύπτουμε το design κομμάτι της εφαρμογής, στο οποίο δημιουργούμε 3 κουμπιά και τους δίνουμε ονομασίες, default κείμενο καθώς και συγκεκριμένες διαστάσεις/μέγεθος.

Ο κώδικας του:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    tools:context=".MainActivity"
    android:paddingTop="200dp"
    android:paddingBottom="10dp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"
    android:orientation="vertical"
   >

    <Button
        android:id="@+id/device_connect"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/connect_to_device" />

    <Button
        android:id="@+id/start"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/start" />

    <Button
        android:id="@+id/stop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/stop" />

</LinearLayout>
```

Εικόνα 5.9: Αρχείο activity_main.xml

Και το design πλαίσιο του:

Εικόνα 5.10: Design/GUI της εφαρμογής του πρότζεκτ

5.3.2 Αρχείο AndroidManifest.xml:

Σε αυτό το αρχείο η μόνη αλλαγή που έχει γίνει στο ήδη παραγόμενο αρχείο είναι η προσθήκη των εντολών:

<uses-permission android:name="android.permission.BLUETOOTH" />

<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

με σκοπό την εκχώρηση άδειας στην εφαρμογή να χρησιμοποιήσει Bluetooth.

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

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BluetoothRobot"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Εικόνα 5.11: Αρχείο AndroidManifest.xml

5.3.3 Αρχείο MainActivity.java:

Αυτό είναι το κύριο μέρος της εφαρμογής, και χωρίζεται σε διάφορα τμήματα:

Αρχικά έχουμε όλα τα imports, δηλαδή εισάγουμε βιβλιοθήκες οι οποίες περιέχουν συναρτήσεις και κλάσεις. Έπειτα δημιουργούμε την βασική κλάση μας, MainActivity, που εκεί μέσα γίνονται όλες οι βασικές λειτουργίες της εφαρμογής. Ξεκινάμε με δημιουργία μεταβλητών, κλάσεων (class instance) και των κουμπιών (objects) και στη συνέχεια προχωράμε με δημιουργία τεσσάρων βασικών συναρτήσεων.

Συνάρτηση onCreate(): Σε αυτή τη συνάρτηση κάνουμε κάποιες αρχικοποιήσεις και θέτουμε την βασική λειτουργία. Αρχικά εάν το bluetooth δεν είναι διαθέσιμο, μας επιστρέφει αντίστοιχο μήνυμα στην οθόνη. Παρακάτω έχουμε την διαχείριση του πρώτου κουμπιού (device_connect) με το περιεχόμενο του να αλλάζει δυναμικά ανάλογα με την κατάσταση σύνδεσης του Bluetooth. Στη συνέχεια έχουμε ένα onClick Listener για το ίδιο κουμπί το οποίο εάν πατηθεί από τον χρήστη, ανάλογα με την τρέχουσα κατάσταση του bluetooth, είτε θα αποσυνδεθεί είτε θα δείξει μια λίστα με διαθέσιμες συσκευές. Αντίστοιχα, για τα δύο υπόλοιπα κουμπιά (start, stop)

έχουμε πάλι onClick Listener τα οποία μόλις πατηθούν από τον χρήστη, στέλνουν στη συνδεδεμένη συσκευή την τιμή της μεταβλητής που έχουμε ορίσει (στην περίπτωσή μας, 0 ή 1).

Συνάρτηση onStart(): Εδώ, γίνεται έλεγχος αν το bluetooth είναι ενεργό, σε περίπτωση που δεν είναι ενεργό, μέσω λούπας περιμένει να ενεργοποιηθεί και ύστερα συνεχίζει παρακάτω. Γίνεται για τελευταία φορά έλεγχος του αν το bluetooth είναι ενεργό, και όπως προηγουμένως εάν δεν είναι, το ενεργοποιούμε, και εάν είναι ήδη Συνάρτηση onActivityResult(): Μετά από την εντολή μας για επιλογή συσκευής προς σύνδεση (στη συνάρτηση onCreate) εδώ γίνεται η διαχείριση της απάντησης από αυτό το “ερώτημα” για το εάν η συσκευή που επιλέξαμε να συνδεθούμε είναι όντως διαθέσιμη και εάν καταφέραμε να συνδεθούμε με αυτή, ώστε να ανταλλάξουμε δεδομένα.

Συνάρτηση onDestroy(): Εδώ η μόνη δραστηριότητα είναι η διακοπή του bluetooth όταν το πρόγραμμά μας τερματίσει την λειτουργία του.

```
package com.example.bluetoothrobot;

import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import app.akexorcist.bluetotohspp.library.BluetoothSPP;
import app.akexorcist.bluetotohspp.library.BluetoothState;
import app.akexorcist.bluetotohspp.library.DeviceList;

public class MainActivity extends AppCompatActivity {

    final String START = "0";
    final String STOP = "1";

    BluetoothSPP bluetooth; //create instance of BluetoothSPP class

    Button device_connect;
    Button start;
    Button stop;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); //default
        setContentView(R.layout.activity_main); //default

        bluetooth = new BluetoothSPP( context: this); //new instance of class

        device_connect = findViewById(R.id.device_connect); //link buttons
        start = findViewById(R.id.start);
        stop = findViewById(R.id.stop);
```

Εικόνα 5.12: Αρχείο MainActivity.java (α)

```
37
38          //if bluetooth is not available, view quick msg indicating so.
39          if (!bluetooth.isBluetoothAvailable()) {
40              Toast.makeText(getApplicationContext(), text: "Bluetooth is not available", Toast.LENGTH_
41              finish();
42          }
43
44          //bluetooth listener actions
45          bluetooth.setBluetoothConnectionListener(new BluetoothSPP.BluetoothConnectionListener() {
46              public void onDeviceConnected(String name, String address) { //on device connection chan
47                  device_connect.setText("Connected to " + name);
48              }
49
50              public void onDeviceDisconnected() { //on device disconnection change button text to con
51                  device_connect.setText("Connection lost");
52              }
53
54              public void onDeviceConnectionFailed() { //on device connection failure change button te
55                  device_connect.setText("Unable to connect");
56              }
57          });
58
59
60          //actions when user "clicks" (taps) on device connect button
61          device_connect.setOnClickListener(new View.OnClickListener() {
62              @Override
63              public void onClick(View v) { //if bluetooth state was already connected and user presse
64                  if (bluetooth.getServiceState() == BluetoothState.STATE_CONNECTED) {
65                      bluetooth.disconnect();
66                  } else { //else list devices, choose which to connect and request a connection with
67                      Intent intent = new Intent(getApplicationContext(), DeviceList.class);
68                      startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
69                  }
70              }
71          });
```

Εικόνα 5.13: Αρχείο MainActivity.java (β)

```
73          //action when user presses the start button
74          start.setOnClickListener(new View.OnClickListener() {
75              @Override
76              public void onClick(View v) {
77                  bluetooth.send(START, CRLF: true); //send "0" when start button is pressed.
78              }
79          });
80
81
82          //action when user presses the stop button
83          stop.setOnClickListener(new View.OnClickListener() {
84              @Override
85              public void onClick(View v) {
86                  bluetooth.send(STOP, CRLF: true); //send "1" when stop button is pressed
87              }
88          });
89
90
91
92      }
93
94      public void onStart() {
95          //Toast.makeText(getApplicationContext(), "onStart function", Toast.LENGTH_SHORT).show();
96          super.onStart();
97          while(!bluetooth.isBluetoothEnabled()){ //waits for bluetooth to be enabled before continuing,
98              bluetooth.enable(); // this makes sure that application does not crash if started with bluetooth turned off
99          }
100         if (!bluetooth.isBluetoothEnabled()) { //if bluetooth is not enabled on device, enable.
101             bluetooth.enable();
102             Toast.makeText(getApplicationContext(), text: "bluetooth enabling function", Toast.LENGTH_SHORT).show();
103         } else { //else if bluetooth is enabled, setup and start bluetooth service
104             if (!bluetooth.isServiceAvailable()) {
105                 bluetooth.setupService();
106                 Toast.makeText(getApplicationContext(), text: "setting up bluetooth", Toast.LENGTH_SHORT).show();
107                 bluetooth.startService(BluetoothState.DEVICE_OTHER);
108                 Toast.makeText(getApplicationContext(), text: "starting bluetooth service", Toast.LENGTH_SHORT).show();
109             }
110         }
```

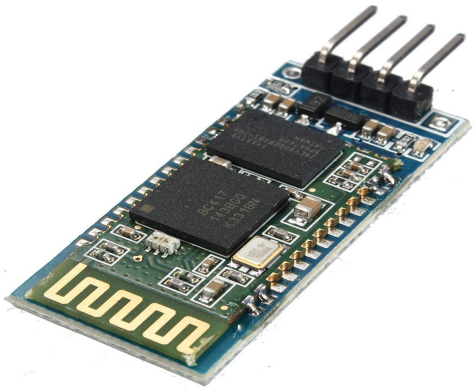
Εικόνα 5.14: Αρχείο MainActivity.java (γ)

```
110         }
111     }
112
113     public void onDestroy() {
114         super.onDestroy();
115         bluetooth.stopService();
116     }
117
118     //check results from intent to choose device activity
119     public void onActivityResult(int requestCode, int resultCode, Intent data) {
120         if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
121             if (resultCode == Activity.RESULT_OK)
122                 bluetooth.connect(data);
123         } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
124             if (resultCode == Activity.RESULT_OK) {
125                 bluetooth.setupService();
126             } else {
127                 Toast.makeText(getApplicationContext()
128                         , text: "Error; Bluetooth was not enabled, you failed to choose a device."
129                         , Toast.LENGTH_SHORT).show();
130                 //finish();
131             }
132         }
133     }
134 }
135
```

Εικόνα 5.15: Αρχείο MainActivity.java (δ)

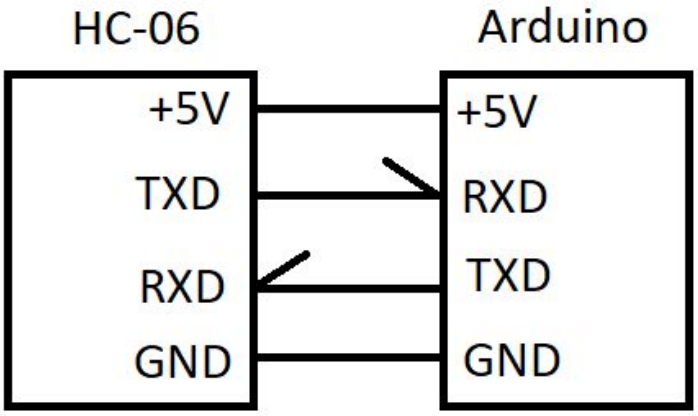
5.4 Επικοινωνία Android και Arduino με Bluetooth module HC-06

Το module HC-06, είναι μια arduino-συμβατή συσκευή bluetooth για την σειριακή επικοινωνία. Έχει 4 pins: Vcc, GND, TXD και RXD. Vcc και GND είναι η τροφοδοσία και η γείωση αντίστοιχα και τα δύο υπόλοιπα pins είναι για την σειριακή επικοινωνία, transfer και receive αντίστοιχα. Έτσι, μπορούμε με αυτή τη συσκευή να έχουμε σειριακή επικοινωνία μεταξύ του arduino και της android συσκευής μας, ασύρματα.



Εικόνα 5.16: Το module HC-06

Η σύνδεση με το Arduino γίνεται ως εξής, οι ακροδέκτες Vcc και GND που είναι για την τροφοδοσία και την γείωση αντίστοιχα, θα συνδεθούν στους ακροδέκτες του Arduino +5V και GND. Ο ακροδέκτης TXD θα συνδεθεί με τον ακροδέκτη Rx του Arduino και ο ακροδέκτης RXD του HC-06 θα συνδεθεί στον ακροδέκτη Tx του Arduino.



Εικόνα 5.17: Συνδεσμολογία με Bluetooth module HC-06 και Arduino

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://en.wikipedia.org/wiki/Arduino
2. https://www.arduino.cc/
3. https://en.wikipedia.org/wiki/Android_(operating_system)
4. https://developer.android.com/
5. https://en.wikipedia.org/wiki/Robotics
6. https://robots.ieee.org/
7. https://en.wikipedia.org/wiki/Bluetooth
8. Ρομποτική για μηχανικούς - Δρ. Σταμάτης Αλατσαθιανός & Ειρήνη Αλατσαθιανού
9. Εισαγωγή στη μηχατρονική και στα ενσωματωμένα συστήματα - Δρ. Σταμάτης Αλατσαθιανός