ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ
ΔΙΚΤΥΩΝ

Απομακρυσμένος έλεγχος ρομποτικής διάταξης με την χρήση αναγνώρισης κίνησης

Διπλωματική Εργασία

Ντελής Γ. Γιώργος

Επιβλέποντες : Νικόλαος Μπέλλας
Αναπληρωτής Καθηγητής

Χρήστος Δ. Αντωνόπουλος
Επίκουρος Καθηγητής

Βόλος, Ιούλιος 2012

Στην οικογένεια & στους φίλους μου

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας και φτάνοντας στο τέλος των προπτυχιακών σπουδών μου, θα ήθελα να ευχαριστήσω όλους τους ανθρώπους που με βοήθησαν να φτάσω ως εδώ. Την οικογένεια μου για την αμέριστη συμπαράσταση και την ανεκτίμητη βοήθεια καθ' όλη τη διάρκεια των σπουδών μου. Τους επιβλέποντες καθηγητές μου, κ. Νικόλαο Μπέλλα και κ. Χρήστο Αντωνόπουλο για την άριστη συνεργασία, την καθοδήγηση και τις υποδείξεις, κατά την εκπόνηση της διπλωματικής εργασίας. Επίσης, τον μεταπτυχιακό φοιτητή Αριστείδη Ιακώβου για τη συνεργασία που είχαμε στη διπλωματική εργασία μου. Τέλος τους φίλους και συμφοιτητές που ήταν δίπλα μου όλα αυτά τα χρόνια.

Γιώργος Ντελής
Βόλος, 2012

Πίνακας περιεχομένων

1. Εισαγωγή

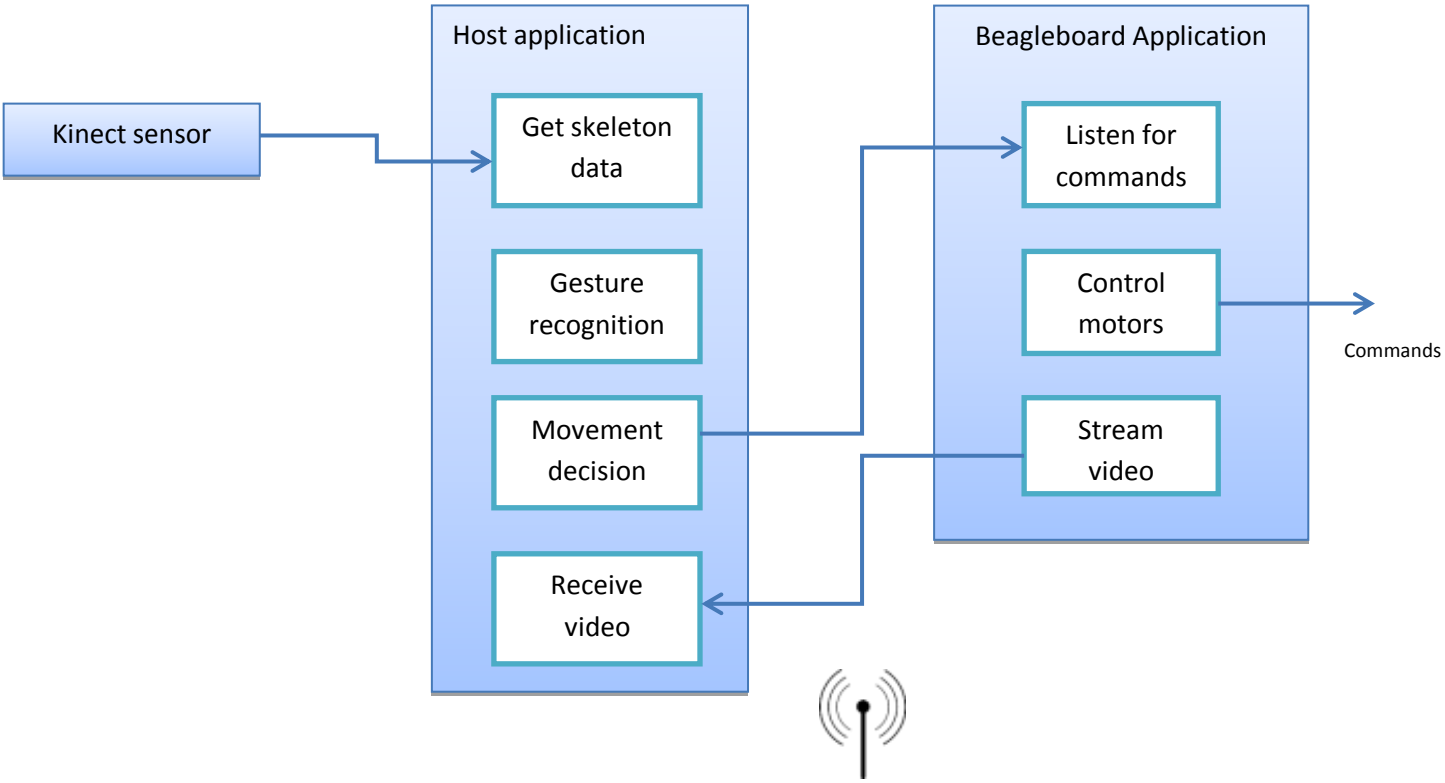
1.1 Περιγραφή του προβλήματος και συμβολή της εργασίας

Η ολοένα και αυξανόμενη χρήση ρομποτικών διατάξεων στην καθημερινότητα, έχει οδηγήσει σε μία σημαντική αύξηση του ερευνητικού ενδιαφέροντος στον τομέα της ρομποτικής. Η αλληλεπίδραση ανθρώπου μηχανής ή αλλιώς Human – Robot interaction (HRI) είναι ο κλάδος που μελετά αυτήν την αλληλεπίδραση μεταξύ ανθρώπου και ρομπότ. Η επικοινωνία μεταξύ χρήστη και ρομπότ είναι δυνατή μέσω του ανάλογου λογισμικού και υλικού, με τη βοήθεια συνήθως ενός πληκτρολογίου, ποντικιού ή joystick ως μέσα για την είσοδο πληροφορίας στο σύστημα. Η ανάγκη εξοικείωσης του ανθρώπου με τις μηχανές, έχει αποτελέσει τη βάση για την ανάπτυξη νέων μεθόδων επικοινωνίας, πιο ανθρώπινων και φιλικών στον τελικό χρήστη. Η παρουσίαση και υλοποίηση μιας τέτοιας μεθόδου αποτέλεσε ένα μεγάλο μέρος της εργασίας.

Η διπλωματική εργασία χωρίζεται σε δύο βασικούς τομείς. Ο πρώτος αφορά το σχεδιασμό και την κατασκευή μιας ρομποτικής διάταξης με την χρήση του Beagleboard. Το Beagleboard είναι μία Embedded πλατφόρμα που αναπτύχθηκε ως εκπαιδευτική πλατφόρμα για να μυήσει σπουδαστές, στον κόσμο του opensource υλικού και λογισμικού. Η πλατφόρμα αυτή, χρησιμοποιήθηκε ως κεντρική μονάδα επεξεργασίας του ρομπότ και επιλέχθηκε για τους εξής λόγους: είναι μικρή σε διαστάσεις (μόλις 3x3”), έχει επιδόσεις που αγγίζουν αυτές ενός σύγχρονου netbook και επίσης έχει μικρή κατανάλωση ενέργειας (περίπου 2W). Ο δεύτερος τομέας της διπλωματικής εργασίας, αφορά την ανάπτυξη μιας εφαρμογής για τον απομακρυσμένο χειρισμό της διάταξης, ξεφεύγοντας ωστόσο από τα συνηθισμένα πρότυπα. Για το σκοπό αυτό χρησιμοποιήθηκε ένας σύγχρονος τρόπος αλληλεπίδρασης μεταξύ ρομπότ και χρήστη, με τη βοήθεια ενός αισθητήρα που αντιλαμβάνεται κινήσεις και στάσεις σώματος του χρήστη. Ο αισθητήρας Kinect της εταιρίας Microsoft που χρησιμοποιήθηκε για τους σκοπούς της εργασίας, έδωσε πρόσφατα την ευκαιρία της αλληλεπίδρασης με τις μηχανές με έναν πιο φυσικό τρόπο, ανοίγοντας νέες προοπτικές για την επικοινωνία μεταξύ ανθρώπου και μηχανής. Ο αισθητήρας αυτός είναι σε θέση να εντοπίζει ανθρώπους αναγνωρίζοντας χειρονομίες, διάφορα σημεία του σώματος και φωνητικές εντολές.

Η τελική υλοποίηση αφορά τον απομακρυσμένο έλεγχο της ρομποτικής διάταξης που κατασκευάστηκε, με χρήση κινήσεων του χρήστη ως μέθοδο χειρισμού. Η εφαρμογή που αναπτύχθηκε προσπελαύνει τον αισθητήρα Kinect, λαμβάνοντας πληροφορίες σχετικά με τη στάση του σώματος του χρηστή. Στη συνέχεια επεξεργάζεται αυτές τις πληροφορίες με σκοπό να αντιστοιχήσει συγκεκριμένες χειρονομίες, οι οποίες έχουν οριστεί κατάλληλα, σε εντολές κίνησης για το ρομπότ. Οι εντολές αυτές έπειτα μεταδίδονται ασύρματα μέσω Wi-fi στο ρομπότ, το οποίο είναι σε θέση να τις εκτελεί και επιπλέον μπορεί να στέλνει ροή βίντεο πίσω στο χρήστη. Για την

επικοινωνία μεταξύ τους έγιναν δύο προσπάθειες: α) Μέσω δικτύου Ad Hoc και β) μέσω διαδικτύου. Και στις δύο περιπτώσεις χρησιμοποιήθηκαν TCP/IP sockets και το μοντέλο client-server. Ως τελική λύση επιλέχθηκε η δεύτερη για λόγους που εξηγούνται στο αντίστοιχο κεφάλαιο. Στο παρακάτω σχήμα παρουσιάζεται αφαιρετικά ο τρόπος λειτουργίας του συστήματος:



1.2 Διάρθρωση Διπλωματικής εργασίας

Στο κεφάλαιο 2 γίνεται η εισαγωγή στην πλατφόρμα Beagleboard και παρουσιάζονται τα βήματα για τη δημιουργία ενός ολοκληρωμένου συστήματος, όπως η εγκατάσταση του λειτουργικού συστήματος και η επικοινωνία με τα διάφορα περιφερειακά.

Στο κεφάλαιο 3 περιγράφεται η διαδικασία κατασκευής της ρομποτικής διάταξης και ο τρόπος οργάνωσής της, ενώ στο 4ο κεφάλαιο περιγράφεται η ανάπτυξη της εφαρμογής για την πλατφόρμα Beagleboard.

Το 5ο κεφάλαιο ασχολείται με την ανάπτυξη της εφαρμογής για τον απομακρυσμένο έλεγχο του ρομπότ, η οποία χρησιμοποιεί τον αισθητήρα Kinect. Τέλος κεφάλαιο 6 παρουσιάζονται κάποιες μελλοντικές προεκτάσεις της εργασίας.

2. Εισαγωγή στο BeagleBoard

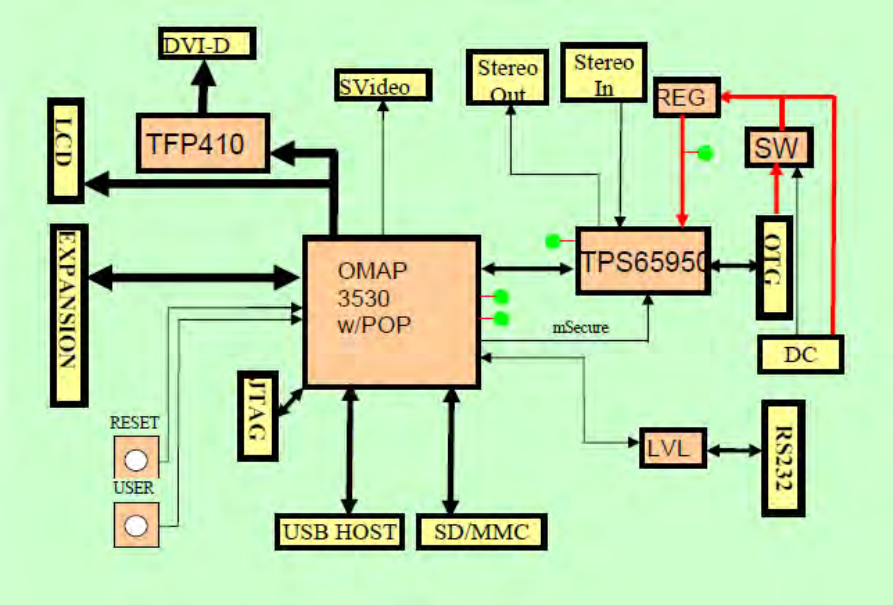
Το **Beagleboard** είναι ένα χαμηλού κόστους, fan-less, single board κομπιούτερ διαστάσεων 3x3", το οποίο κατασκευάζεται από την εταιρία Texas Instruments και βασίζεται στην οικογένεια των επεξεργαστών OMAP, με όλες τις δυνατότητες επέκτασης των σύγχρονων desktop. Ο OMAP περιλαμβάνει έναν ARM Cortex-A8 επεξεργαστή. Το Beagleboard αναπτύχθηκε από μία μικρή ομάδα μηχανικών, με σκοπό να χρησιμοποιηθεί ως educational board σε πανεπιστήμια, ώστε να διδάξει τους σπουδαστές τις δυνατότητες του opensource hardware και του opensource λογισμικού. Πίσω από το Beagleboard υπάρχει μία ομάδα κόσμου που ασχολείται με το Development πάνω σε αυτήν την πλατφόρμα, προσφέροντας μία ευρεία γκάμα από opensource εργαλεία, κάτι που αποτελεί ένα σημαντικό λόγο για την απόκτηση του. Στη συνέχεια το Beagleboard θα αναφέρεται ως **BB** προς χάριν συντομίας. Το BB είναι μία εξελισσόμενη πλατφόρμα, πράγμα που σημαίνει υπάρχουν διάφορες εκδόσεις του, με διαφορετικά χαρακτηριστικά η κάθε μία. Το μοντέλο που χρησιμοποιήθηκε στα πλαίσια της πτυχιακής εργασίας είναι το **Beagleboard C4** τα χαρακτηριστικά του οποίου φαίνονται στον ακόλουθο πίνακα:

Εικόνα 1 – Η πλατφόρμα Beagleboard

Εικόνα 2 – Χαρακτηριστικά Beagleboard

	Feature	
Processor	OMAP3530DCBB72 720MHz	
POP Memory	Micron	
	2Gb NAND (256MB)	2Gb MDDR SDRAM (256MB)
PMIC TPS65950	Power Regulators	
	Audio CODEC	
	Reset	
	USB OTG PHY	
Debug Support	14-pin JTAG	GPIO Pins
	UART	LEDs
PCB	3.1" x 3.0" (78.74 x 76.2mm)	6 layers
Indicators	Power	2-User Controllable
	PMU	
HS USB 2.0 OTG Port	Mini AB USB connector	
	TPS65950 I/F	
	MiniAB	
HS USB Host Port	Single USB HS Port	Up to 500ma Power
Audio Connectors	3.5mm	3.5mm
	L+R out	L+R Stereo In
SD/MMC Connector	6 in 1 SD/MMC/SDIO	4/8 bit support, Dual voltage
User Interface	1-User defined button	Reset Button
Video	DVI-D	S-Video
Power Connector	USB Power	DC Power
Expansion Connector (Not Populated)	Power (5V & 1.8V)	UART
	McBSP	McSPI
	I2C	GPIO
	MMC	PWM
2 LCD Connectors	Access to all of the LCD control signals plus I2C	3.3V, 5V, 1.8V

Στο BB μπορούν να φορτωθούν διάφορα λειτουργικά συστήματα (Linux, Risc OS, Symbian, Android), με την καλύτερη επιλογή να είναι κάποια διανομή Linux καθώς υπάρχει ένας σημαντικός αριθμός από developers που υποστηρίζουν το BB, προσφέροντας κατάλληλους drivers για το hardware του BB, ένα καλό tool chain καθώς και πολλά έτοιμα πακέτα έτοιμα για εγκατάσταση. Παρακάτω φαίνεται ένα high level block diagram του BB, με την θέση των διαφόρων components να ανταποκρίνεται στην πραγματική τους θέση.



Εικόνα 3 – BeagleBoard schematic

2.1 Πρώτα βήματα στο BB

Προτού ξεκινήσει κανείς να δουλεύει με το BB θα πρέπει να κάνει το setup της πλατφόρμας. Στη συσκευασία του BB δεν εμπεριέχονται καλώδια, επομένως για την οργάνωση του συστήματος χρειάζονται κάποια επιπλέον αντικείμενα. Για το σύστημα που υλοποιήθηκε χρειάστηκαν τελικά :

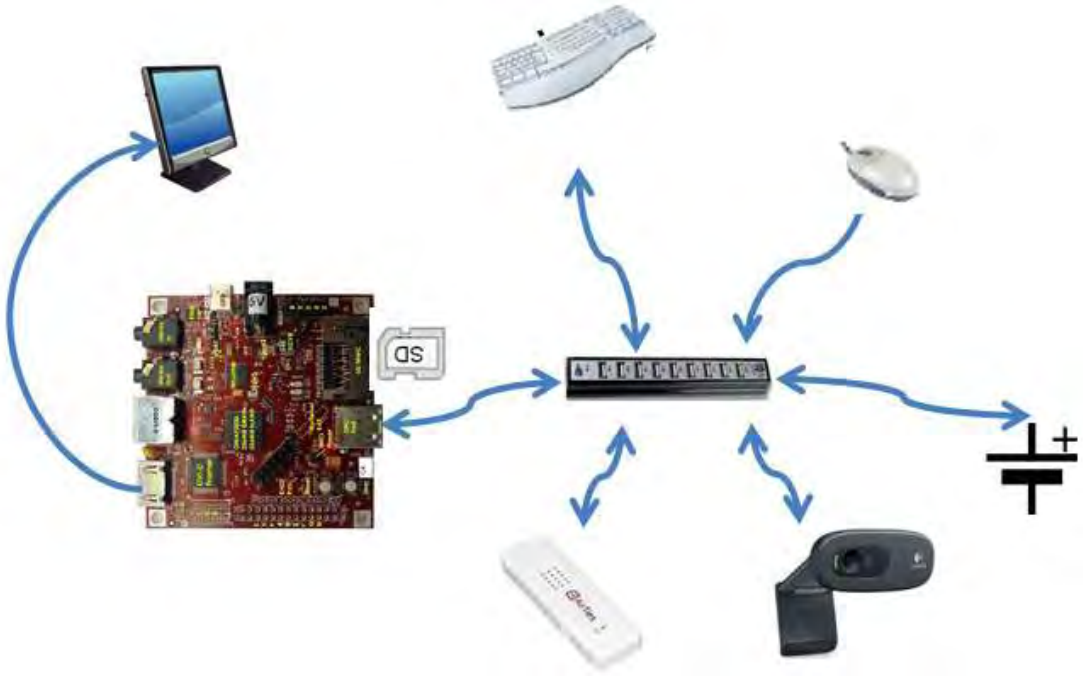
- Ένα αυτοτροφοδοτούμενο hub
- Πληκτρολόγιο και ποντίκι
- Μία κάρτα SDHC 16G
- USB Wifi stick Airties wus-201
- Logitech c270 web camera
- HDMI καλώδιο για τη σύνδεση με την οθόνη
- Serial Adaptor & serial RS232 cable

Εικόνα 4 – USB to 5.5mm

- USB to 5.5mm cable

Εικόνα 5 – Serial adaptor

Η τροφοδοσία που θα συνδεθεί στο BB θα πρέπει να είναι 5V και θα πρέπει να έχει δυνατότητα παροχής τουλάχιστον 500mA ρεύματος. Υπάρχουν δύο τρόποι για να προσφερθεί. Μπορεί να χρησιμοποιηθεί είτε ένα καλώδιο USB standard-A to mini-A USB το οποίο συνδέεται στη θύρα USB-OTG, είτε να συνδέσουμε 5V DC στην DC παροχή του BB. Η λύση που επιλέχτηκε ήταν η δεύτερη, με τη χρήση ενός καλωδίου USB to 5.5mm το οποίο συνδέεται στο τροφοδοτούμενο HUB. Η χρήση ενός αυτοτροφοδοτούμενου HUB και όχι ενός απλού είναι επιβεβλημένη καθώς η USB έξοδος του BB μπορεί να δώσει μόνο 100mA ρεύματος πράγμα που σημαίνει ότι δεν είναι ικανή να οδηγήσει ένα ποντίκι ή κάποιο πληκτρολόγιο. Στη DC παροχή του BB θα πρέπει να συνδεθεί τροφοδοσία ακριβώς **5V**, καθώς σε άλλη περίπτωση υπάρχει περίπτωση βλάβης της πλατφόρμας. Στη συνέχεια παρουσιάζεται σχηματικά η οργάνωση του συστήματος:



Εικόνα 6 – Οργάνωση του συστήματος

2.2 Εγκατάσταση του λειτουργικού συστήματος στο BB

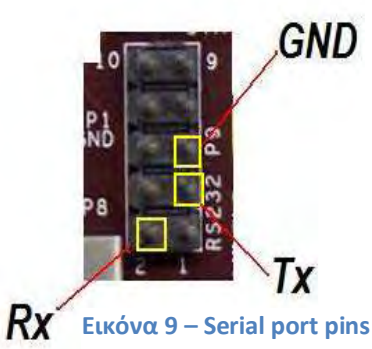
Όπως αναφέρθηκε και αρχικά, στην πλατφόρμα BB υπάρχει η δυνατότητα εγκατάστασης διαφόρων λειτουργικών συστημάτων, μεταξύ των οποίων τα Angstrom, Ubuntu, Maemo και Android είναι τα περισσότερο δημοφιλή. Στην περίπτωση της παρούσας εργασίας έγινε δοκιμή με τη γνωστή διανομή Ubuntu (10.10 Maverick netbook release) όσο και με τη διανομή Angstrom η οποία είναι μια φιλική στο χρήση και stable διανομή ιδανική για Embedded συσκευές κυρίως. Αν και το Ubuntu αποτελεί ίσως την πιο ολοκληρωμένη επιλογή, προσφέροντας την καλύτερη υποστήριξη για τα περιφερειακά που έχουν συνδεθεί πάνω στο BB, δεν επιλέχθηκε τελικά λόγω

κάποιων σημαντικών λόγων. Καταρχήν το γεγονός ότι το Ubuntu είναι μία αρκετά φορτωμένη διανομή Linux, με πλούσιο γραφικό περιβάλλον, είχε ως αποτέλεσμα να είναι αργή η απόκριση του συστήματος, ακόμη και μετά την απενεργοποίηση του γραφικού περιβάλλοντος. Ο δεύτερος πολύ σημαντικός λόγος για τον οποίο εγκαταλείφθηκε αυτή η επιλογή είναι ότι στο Ubuntu δεν υπάρχει κάποιος native compiler για τον επεξεργαστή ARM που διαθέτει το BB, απαιτώντας να γίνεται cross-compile οποιουδήποτε κώδικα, κάνοντας έτσι τη διαδικασία του development πιο δύσκολη. Έτσι επιλέχθηκε η διανομή Angstrom, η οποία αναιρεί όλα τα προηγούμενα μειονεκτήματα, ωστόσο αμέσως εμφανίστηκαν προβλήματα με κάποια περιφερειακά όπως το usb wifi-stick, λεπτομέρειες για το οποίο θα αναφέρω στη συνέχεια. Στις παρακάτω εικόνες φαίνονται δύο στιγμιότυπα από τις δύο διανομές ενώ τρέχουν στο BB:

Εικόνα 8 – Ubuntu 10.10 Netbook-release

Εικόνα 7 - Angstrom

Πρώτο βήμα ξεκινώντας κανείς να δουλεύει με το BB είναι η εγκαθίδρυση επικοινωνίας με τον υπολογιστή, μέσω της σειριακής θύρας που αυτό διαθέτει. Στο σημείο αυτό θα πρέπει να είναι κανείς προσεκτικός ώστε να κατευθύνει στο σειριακό καλώδιο τις σωστές εξόδους από το RS232 header του BB, δηλαδή τα pins Rx, Tx και GND όπως φαίνονται και στη φωτογραφία. Η έλλειψη σειριακής θύρας στον υπολογιστή, μπορεί να αντιμετωπιστεί με τη χρήση ενός USB-to-Serial καλωδίου. Στη συνέχεια περιγράφονται τα βήματα που ακολουθηθήκαν για την εγκαθίδρυση της επικοινωνίας μεταξύ των δύο συσκευών:



Εικόνα 9 – Serial port pins

-- Αρχικά πρέπει να γίνει εγκατάσταση (εάν δεν υπάρχει) του προγράμματος **minicom** στο σύστημα:

```
$ sudo apt-get install minicom
```

-- Τρέξιμο του minicom ως root

```
$ sudo minicom -s
```

```
           +-----[configuration]------+
           | Filenames and paths      |
           | File transfer protocols  |
           | Serial port setup        |
           | Modem and dialing        |
           | Screen and keyboard      |
           | Save setup as dfl        |
           | Save setup as..          |
           | Exit                     |
           | Exit from Minicom        |
           +--------------------------+
```

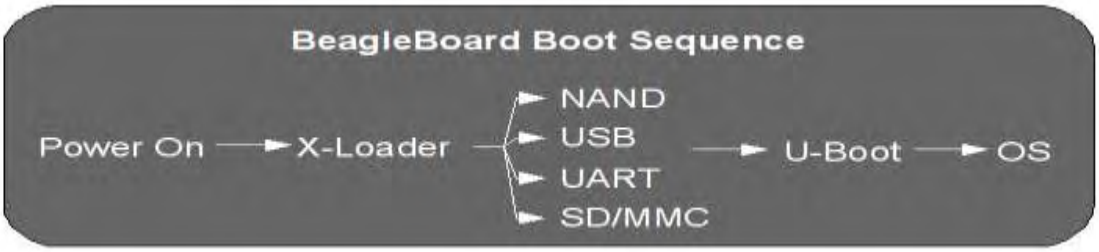
-- Στο setup της σειριακής θύρας θα πρέπει να αλλαχθεί το serial device σε /dev/ttyUSB0, η τιμή /Bps/Par/Bits σε 115200 8N1, να απενεργοποιηθεί το flow control και τέλος να σωθούν οι αλλαγές:

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```

-- Ενεργοποιώντας το BB θα πρέπει να εμφανιστεί η παρακάτω έξοδος, ένδειξη του ότι η όλη διαδικασία έγινε σωστά.

```
Welcome to minicom 2.2

OPTIONS:
Compiled on Sep  8 2008, 17:03:34.
Port /dev/ttyS0

              Press CTRL-A Z for help on special keys


Texas Instruments X-Loader 1.41
Starting OS Bootloader...

U-Boot 1.3.3 (Jul 10 2008 - 16:33:09)

OMAP3530-GP rev 2, CPU-OPP2 L3-165MHz
OMAP3 Beagle Board + LPDDR/NAND
DRAM:  128 MB
NAND:  256 MiB
In:    serial
Out:   serial
Err:   serial
Audio Tone on Speakers  ... complete
OMAP3 beagleboard.org #
```

Προτού προχωρήσουμε στην εγκατάσταση του λειτουργικού στο BB, ας δούμε με ποιον τρόπο γίνεται η φόρτωση του λειτουργικού στην πλατφόρμα. Αυτό δεν γίνεται αμέσως με το που τροφοδοτηθεί το BB αλλά με έναν τρόπο που περιγράφεται στην παρακάτω εικόνα, και είναι γνωστός ως multi-stage boot.



Εικόνα 10 – Boot sequence

Ο boot-loader πρώτου σταδίου ονομάζεται X-Loader και πρέπει να είναι μικρός ώστε να χωράει στην on-chip μνήμη του OMAP3530, μιας και σε αυτό το σημείο η εξωτερική μνήμη δεν έχει ακόμη αρχικοποιηθεί. Ένα αντίγραφο του X-Loader που ονομάζεται MLO είναι το πρώτο αρχείο που αντιγράφεται στην SD κάρτα, και είναι αυτό που ψάχνει να βρει η πλατφόρμα όταν ενεργοποιείται. Ο boot loader αυτός αρχικοποιεί την εξωτερική μνήμη, αντιγράφει τον boot-loader δεύτερου σταδίου στη μνήμη και τον εκτελεί. Ο δεύτερος boot-loader ονομάζεται U-Boot και δουλειά του είναι να φορτώσει τον Linux kernel καθώς και το root File system.

2.2.1 Προετοιμασία της SD κάρτας μνήμης

Ο κύριος αποθηκευτικός χώρος του BB είναι μία κάρτα SDHC 16G και σε αυτή εγκαθίσταται και το λειτουργικό, επομένως δεύτερο βήμα ήταν η προετοιμασία της κάρτας μνήμης. Για να φιλοξενηθεί το Linux χρειάστηκε να δημιουργηθούν δύο partitions με τη βοήθεια του εργαλείου ***fdisk*** που υπάρχει στο Linux. Το πρώτο θα είναι bootable και θα περιέχει τρία αρχεία που ονομάζονται: u-boot.bin, MLO και uImage, ενώ το δεύτερο θα περιέχει το File System. Στη συνέχεια παρουσιάζονται αναλυτικά οι οδηγίες βήμα προς βήμα:

--Καθαρισμός του partition table

```
==================================================================
$ sudo fdisk /dev/sdb

Command (m for help): o
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected
by w(rite)
==================================================================
```

--Πληροφορίες της κάρτας, τα νούμερα θα χρειαστούν αργότερα

```
==================================================================
Command (m for help): p

Disk /dev/sdb: 16 GB, 16005464064 bytes
....
==================================================================
```

--Έπειτα γίνεται επιλογή του “expert mode”

```
==================================================================
Command (m for help): x
==================================================================
```

--Τώρα πρέπει να τεθεί η γεωμετρία σε 255 heads, 63 sectors και να υπολογιστεί ο αριθμός των κυλίνδρων (C) για τη συγκεκριμένη κάρτα SD, ο οποίος είναι $\mathbf{C = B/255/63/512}$, όπου B είναι ο αριθμός των bytes της SD. Η στρογγυλοποίηση θα πρέπει να γίνει προς τα κάτω, αν δηλαδή το αποτέλεσμα είναι 108,8 θα πρέπει να γραφτεί 108.

```
==================================================================
Expert command (m for help): c
Number of cylinders (1-1048576, default 1011): 1945
==================================================================
```

--Στη συνέχεια θα πρέπει να δημιουργηθεί ένα FAT32 partition το οποίο χρησιμοποιείται κατά το booting της συσκευής:

```
================================================================
Expert command (m for help): r
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-245, default 1): (press Enter)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-245, default 245): +50

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
Partition number (1-4): 1
================================================================
```

--Ακολουθεί το Linux partition για το File System:

```
================================================================
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-245, default 52): (press Enter)
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (52-245, default
245):(press Enter)
Using default value 245
================================================================
```

--Πληροφορίες κάρτας και σώσιμο των αλλαγών:

```
================================================================
Command (m for help): p

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1          51      409626    c  W95 FAT32
(LBA)
/dev/sdc2              52         245     1558305   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device
or resource busy. The kernel still uses the old table. The new table
will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x partitions,
please see the fdisk manual page for additional information.
Syncing disks.
================================================================
```

--Μετά την επιτυχή ολοκλήρωση των παραπάνω θα πρέπει να διαμορφωθούν και τα δύο partitions:

```
=====================================================================
$ sudo mkfs.msdos -F 32 /dev/sdc1 -n boot
mkfs.msdos 2.11 (12 Mar 2005)

$ sudo mkfs.ext3 /dev/sdc2 -L Angstrom
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:
=====================================================================
```

2.2.2 Προετοιμασία του Angstrom λειτουργικού

Έχοντας πλέον έτοιμο για χρήση τον αποθηκευτικό χώρο, θα πρέπει να δημιουργηθούν τα boot αρχεία καθώς και το Angstrom File System. Αυτό μπορεί να γίνει μεταβαίνοντας στο site http://narcissus.angstrom-distribution.org/ που υπάρχει το Narcissus, ένα online tool που δίνει τη δυνατότητα δημιουργίας του λειτουργικού και των αρχείων για το στάδιο του booting. Η διαδικασία είναι απλή και αφού συμπληρωθούν τα πεδία κατάλληλα όπως φαίνεται στην παρακάτω εικόνα, επιλέγεται το πλήκτρο build και το αποτέλεσμα μετά από λίγα λεπτά είναι ένα συμπιεσμένο αρχείο tar.gz . Ωστόσο στην τελευταία επιλογή ***Additional packages selection*** θα πρέπει να συμπεριληφθούν στην εγκατάσταση τα εξής πακέτα:

- Native (on-target) SDK
- I2C-tools
- Wireless-tools
- Bootloader Files (x-load/u-boot/scripts)

Base settings:

Select the machine you want to build your rootfs image for:

beagleboard

Choose your image name.
This is used in the filename offered for download, makes it easier to distinguish between rootfs images after downloading.

Angstrom-Beagleboard-demo-image

Choose the complexity of the options below.
simple will hide the options most users don't need to care about and *advanced* will give you lots of options to fiddle with.

advanced

Advanced settings:

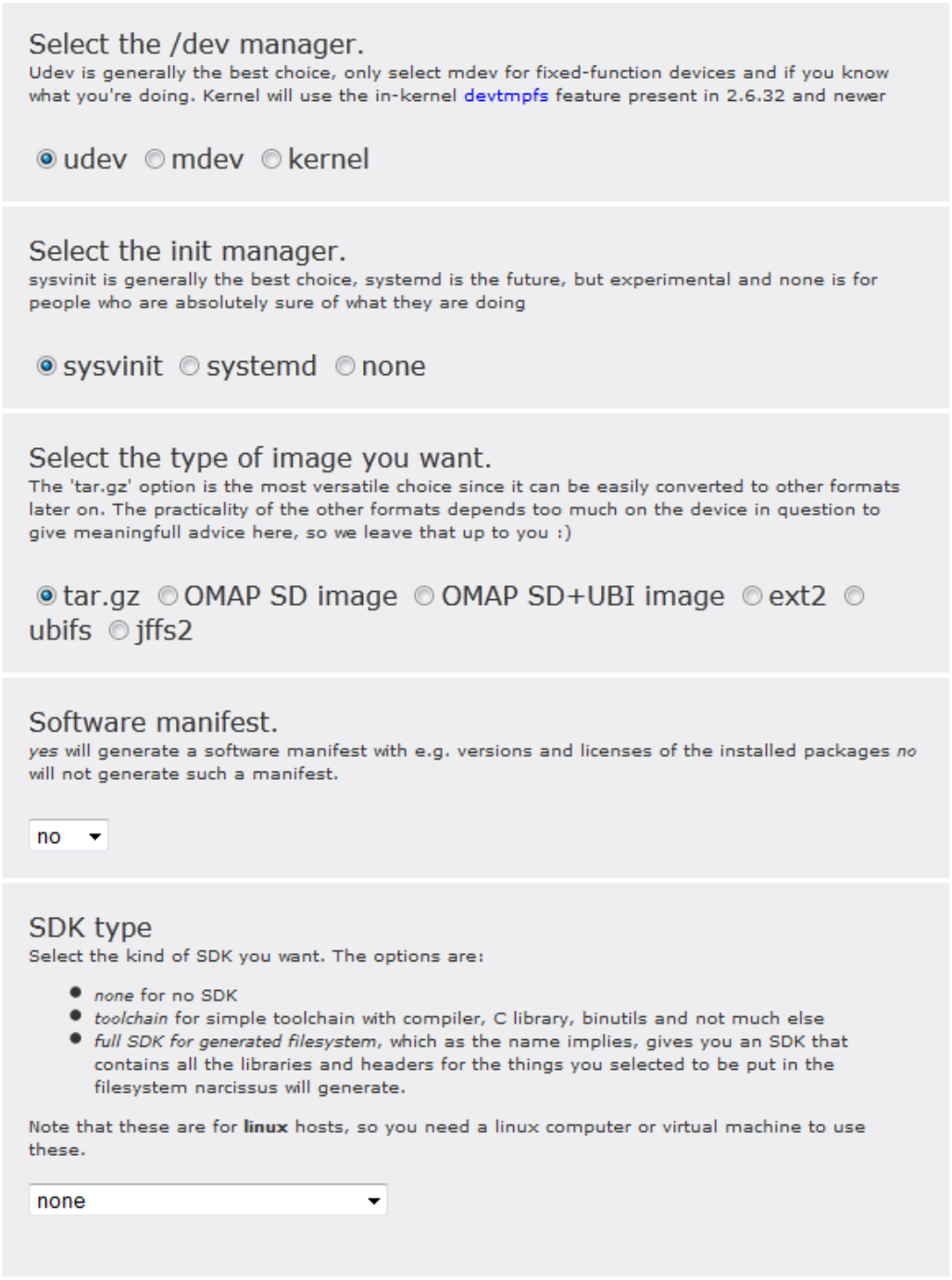
Select the release you want to base your rootfs image on.
The 'stable' option will give you a working system, but will not have the latest versions of packages. The 'unstable' option will give you access to all the latest packages the developers have uploaded, but is known to break every now and then. The 'next' option will give you the bleeding edge, but it's incomplete and only intended for angstrom developers

2011.03

Base system
Each entry down is a superset of the one above it. Busybox will give you only busybox, usefull for e.g. small ramdisks. Task-boot will give you the minimal set of drivers and packages you need to boot. Task-base will give you drivers for non-essential features of your system, e.g. bluetooth. Options below that will include even more drivers for a smoother experience with USB based devices.

- bare bones (busybox)
- small (task-boot)
- regular (task-base)
- extended (task-base-extended)

Εικόνα 11 – Narcissus Tool chain



Στο συμπιεσμένο αρχείο που δημιουργήθηκε, πρέπει να εντοπιστούν τα αρχεία **MLO**, **u-boot.bin**, **uImage** και να αντιγραφούν στο πρώτο partition FAT32 της SD κάρτας. Το αρχείο MLO πρέπει να γραφτεί *πρώτο* στην κάρτα:

```
$ cp MLO /media/boot
$ cp u-boot.bin /media/boot
$ cp uImage /media/boot
```

Ολόκληρο το tar.gz αρχείο μεταφέρεται στο ext3 partition. Το αρχείο αυτό θα πρέπει να γίνει untar στην κάρτα και όχι στο host μηχάνημα. Αφού ολοκληρωθεί η αποσυμπίεση, το tar.gz αρχείο διαγράφεται και τέλος η κάρτα γίνεται unmount.

```
$ sudo cp Angstrom-Beagleboard-demo-image /media/Angstrom
$ cd /media/Angstrom
$ sudo tar -jxvf Angstrom-Beagleboard-demo-image
$ sudo rm Angstrom-Beagleboard-demo-image
$ sudo umount /media/boot
$ sudo umount /media/Angstrom
```

Με το πέρας της παραπάνω διαδικασίας η κάρτα είναι έτοιμη να μπει στο BB και να μπουτάρει για πρώτη φορά. Ένδειξη του ότι όλα τα βήματα έγιναν σωστά είναι η παρακάτω έξοδος στο minicom:

```
.-------.
|       |                    .-.
|   |   |-----.-----.-----.  |  |   .----..-----.-----.
|       |     | __  |  ---'| '--.|   .-'|     |     |
|   |   |  |  |     |---  ||  --'|   |  |  '  |  |  |  |
'---'---'--'--'--.  |-----''----''--'  '-----'-'-'-'-'
                -'  |
                '---'

The Angstrom Distribution beagleboard ttyUSB0

Angstrom 2009.X-stable beagleboard ttyUSB0

beagleboard login:
```

2.3 Υποστήριξη ασύρματης δικτύωσης

Το βήμα αυτό είναι ένα βασικό στάδιο για την υλοποίηση, μιας και το ρομπότ το οποίο θα χρησιμοποιεί ως κεντρική μονάδα το Beagleboard, θα ελέγχεται ασύρματα μέσω δικτύου. Η πλατφόρμα Beagleboard δεν διαθέτει από μόνη της κάποιον προσαρμογέα για ασύρματη σύνδεση στο δίκτυο, έτσι ως καλύτερη επιλογή κρίθηκε η αγορά ενός wifi usb-stick το οποίο προσαρτήθηκε στο τροφοδοτούμενο hub που διαθέτει το Beagleboard. Έτσι λοιπόν με τη βοήθεια του site http://linuxwireless.org , το οποίο είναι ένα από τα βασικότερα site για την ασύρματη δικτύωση πάνω σε λειτουργικό Linux, έγινε αγορά του TP-LINK TL-WN722N. Κάπου εδώ ξεκίνησαν τα προβλήματα που παρουσιαστήκαν με την διανομή Angstrom η οποία είχε εγκατασταθεί στο BB. Το συγκεκριμένο wifi usb-stick χρησιμοποιεί το chipset της Atheros ar9271 και χρειάζεται τους drivers ath9k_htc. Ωστόσο λόγω έκδοσης του kernel δεν υπήρχαν οι απαιτούμενοι drivers. Έτσι έγινε προσπάθεια εγκατάστασης των απαιτούμενων drivers οι οποίοι παρέχονται από το Linux wireless που αναφέρθηκε αρχικά, με δύο τεχνικές που αναφέρονται στη σελίδα αυτή, είτε χρησιμοποιώντας το wireless git tree απευθείας, πραγματοποιώντας τις απαιτούμενες αλλαγές στο σύστημα, είτε με τη βοήθεια ενός εργαλείου, του compact wireless το οποίο και τελικά προτιμήθηκε. Επίσης έγινε προσπάθεια να γίνουν οι drivers cross-compile χωρίς επιτυχία ωστόσο. Εκτός των άλλων, οι οδηγοί αυτοί απαιτούν και κάποιο firmware το οποίο παρέχεται αλλά δεν μπορεί να χρησιμοποιηθεί καθώς δεν είναι σχεδιασμένο για τον επεξεργαστή ARM που

χρησιμοποιεί το BB. Έτσι η μόνη επιλογή ήταν να γίνει λήψη του από το Ubuntu στο οποίο το wifi usb δούλευε κανονικά, αλλά αυτό δεν είχε αποτέλεσμα στο Angstrom. Λόγω των προβλημάτων που ανέκυπταν συνεχώς επειδή το usb wifi είναι σχετικά νέο και ο kernel παλιός, αποφασίστηκε η απόκτηση του USB wifi Airties wus-201 το οποίο χρησιμοποιεί τους drivers *zd1211rw* οι οποίοι εγκαταστάθηκαν και δεν είχαν προβλήματα συμβατότητας λόγω του ότι είναι παλαιότεροι. Έτσι προστέθηκε τελικά η ζητούμενη ασύρματη επικοινωνία.

2.4 Video Streaming από το BB

Η ρομποτική διάταξη που θα κατασκευαστεί θα πρέπει να είναι σε θέση να κάνει streaming βίντεο πίσω στον host υπολογιστή. Για τον σκοπό αυτό χρησιμοποιήθηκε η web camera *Logitech C270* η οποία είναι μία UVC κάμερα. Η κατηγορία **UVC** ή αλλιώς USB video device class περιλαμβάνει συσκευές οι οποίες είναι ικανές να κάνουν streaming βίντεο και λειτουργούν χωρίς ανάγκη κάποιου επιπρόσθετου driver, αλλά χρησιμοποιούν αυτόν που παρέχεται από το εκάστοτε σύστημα. Σε Linux αυτές οι συσκευές υποστηρίζονται από τον Linux UVC driver, που ονομάζεται ***uvcvideo***. Ένα ακόμη μεγάλο πλεονέκτημα των συσκευών που ανήκουν σε αυτήν την κατηγορία, είναι το γεγονός ότι η κάμερα αναλαμβάνει να κάνει όλο το compression και έτσι δεν επιβαρύνεται η CPU με αυτή τη λειτουργία.

Η εφαρμογή που χρησιμοποιήθηκε για το streaming ονομάζεται ***MJPG Streamer***. Μπορεί να χρησιμοποιηθεί για να κάνει κανείς streaming JPEG εικόνες πάνω σε ένα IP-based δίκτυο από μία web camera, σε κάποιον περιηγητή όπως ο Firefox ή ο Chrome. Η εφαρμογή αυτή έχει γραφτεί ειδικά για embedded συσκευές οι οποίες συνήθως διαθέτουν περιορισμένους πόρους όσον αφορά κυρίως τη μνήμη RAM και τη CPU. Οι πηγαίοι κώδικες της εφαρμογής μπορούν να βρεθούν στο link http://sourceforge.net/projects/mjpg-streamer . Μετά τη λήψη του tar.gz αρχείο, οι παρακάτω εντολές μπορούν να χρησιμοποιηθούν για να γίνει compile ο κώδικας:

```
# tar xzvf mjpg-streamer.tgz
# cd mjpg-streamer
# make clean all
```

Η εκτέλεση του προγράμματος πραγματοποιείται με την εξής εντολή:

```
#./mjpg_streamer -i "./input_uvc.so  -d /dev/video0  -y  -f 10 " -o "./output_http.so -w ./www"
```

Η default θύρα που χρησιμοποιεί η εφαρμογή είναι 8080, ωστόσο αυτό μπορεί να αλλάξει όπως επίσης η ανάλυση της εικόνας και τα frames per second. Η προβολή του streaming σε κάποιον

περιηγητή μπορεί να γίνει γράφοντας στο χώρο των διευθύνσεων τα παρακάτω. Εννοείται ότι το πρώτο μέρος της διεύθυνσης είναι η διεύθυνση IP της συσκευής που κάνει το streaming.

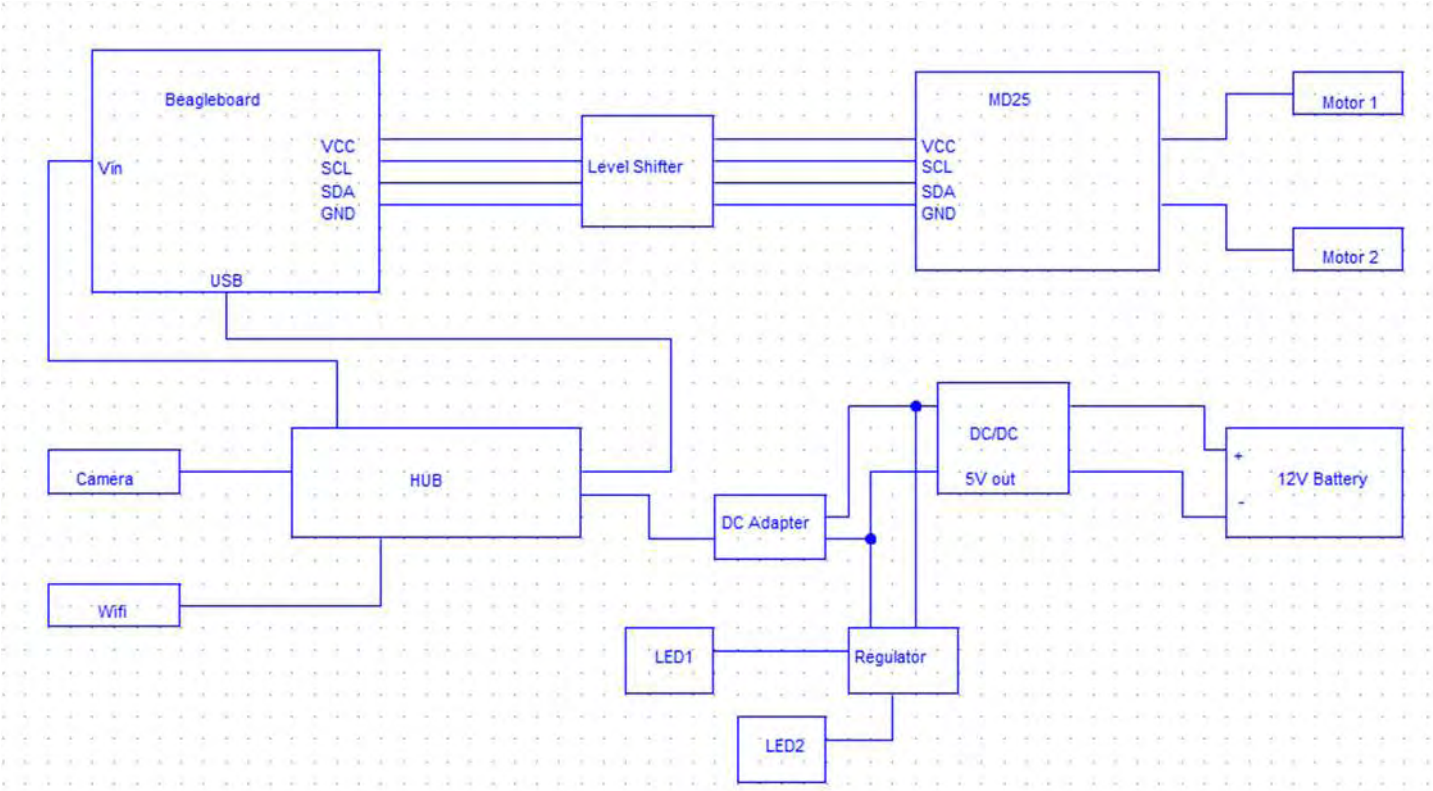
http://127.0.0.1:8080/?action=stream

3. Κατασκευή ρομποτικής διάταξης

Μετά την επιτυχή ολοκλήρωση της προετοιμασίας του BB, ξεκίνησε η διαδικασία κατασκευή της ρομποτικής διάταξης που θα χρησιμοποιηθεί τελικά στην εργασία. Προτού ξεκινήσει οποιοδήποτε κατασκευαστικό στάδιο, έπρεπε να υπάρχει βεβαιότητα ότι το BB επικοινωνεί σωστά με όλα τα περιφερειακά. Ο expansion connector του BB όπως έρχεται στη συσκευασία δεν διαθέτει κάποιο header για τις απαιτούμενες συνδέσεις. Έτσι αρχικά απαιτήθηκε η συγκόλληση ενός τέτοιου header που μου προσέφερε πρόσβαση στα pins του expansion connector. Μπορούν να χρησιμοποιηθούν είτε male είτε female headers και μπορούν να τοποθετηθούν ως προς όποια κατεύθυνση θέλει κανείς. Αφού έγινε η αγορά των Break away male headers προσκολλήθηκαν στο BB με τον τρόπο που φαίνεται στην εικόνα παρακάτω :

Εικόνα 12 – Τοποθέτηση του header

Ο μηχανισμός παροχής κίνησης που χρησιμοποιεί το ρομπότ είναι ο MD25-12V 2.8A Dual H-Bridge, ο οποίος ελέγχει την κίνηση των δύο τροχών που διαθέτει η διάταξη. Έτσι πρώτο βήμα ήταν η επικοινωνία του controller με το BB. Στο παρακάτω schematic που δημιουργήθηκε με την εφαρμογή PSpice, φαίνεται ένα high level schematic με τις συνδέσεις των διαφόρων συστατικών μερών της ρομποτικής διάταξης. Λεπτομέρειες περιγράφονται στη συνέχεια του κεφαλαίου.

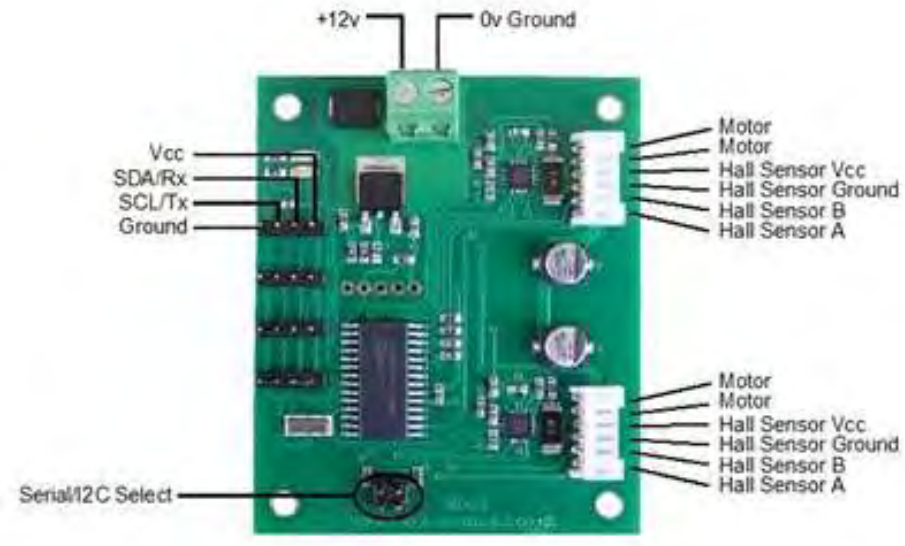


Εικόνα 13 – Robot Schematic

3.1 Επικοινωνία μεταξύ MD25 και BB

Ο ελεγκτής MD25 που φαίνεται στη διπλανή φωτογραφία, κινεί δύο step motors του τύπου EMG30–GearMotor, ανεξάρτητα μεταξύ τους είτε με συνδυασμένο έλεγχο. Μαζί αποτελούν το σύστημα οδήγησης RD02. Για την λειτουργία του απαιτείται η παροχή 12V, ενώ ένας onboard regulator μπορεί να παρέχει 5V (μέχρι 1A peak και 300mA συνεχόμενα σε εξωτερικό κύκλωμα), κάτι χρήσιμο όπως θα φανεί και στη συνέχεια. Ο ελεγκτής προσφέρει δύο επιλογές για τη μεταφορά εντολών· είτε με σειριακή μεταφορά δεδομένων (rs232), είτε με τη χρήση του πρωτοκόλλου I2C. Η επιλογή ήταν τελικά αυτή του I2C και στηρίχθηκε σε αρκετούς λόγους. Καταρχήν η ύπαρξη του είναι γνωστή εδώ και 20 χρόνια και έχει γίνει παγκόσμιο στάνταρ καθώς βρίσκεται σε πολλές συσκευές σήμερα όπως κινητά, PDA's, DVDs κλπ. και έχει υιοθετηθεί από γίγαντες της βιομηχανίας όπως η HP, IBM, Intel, Nokia και πολλές άλλες. Επίσης πάνω στο I2C bus μπορούν να συνδεθούν πολλές συσκευές (σε περίπτωση που μελλοντικά συνδεθούν άλλοι αισθητήρες με το BB) και στο Angstrom υπάρχουν υλοποιήσεις για τον έλεγχο του I2C Bus του BB. Στην ακόλουθη εικόνα φαίνονται αναλυτικά οι συνδέσεις του ελεγκτή. Για να χρησιμοποιηθεί το πρωτόκολλο I2C πρέπει να αφαιρεθούν τα δύο jumpers που υπάρχουν κάτω αριστερά.

Εικόνα 14 –Σύστημα οδήγησης RD02



Εικόνα 15 – MD25 pins

Με τη βοήθεια του I2C είναι δυνατό να γραφτούν ή να διαβαστούν οι διάφοροι registers του ελεγκτή. Καθένας από αυτούς υλοποιεί μία λειτουργία που φαίνεται στον ακόλουθο πίνακα. Οι καταχωρητές που παρουσιάζουν το μεγαλύτερο ενδιαφέρον είναι οι: 0, 1, 10, 14 και 15.

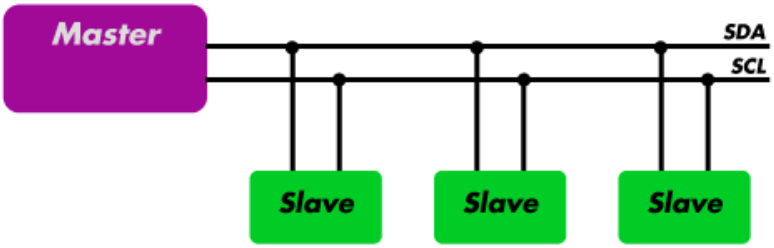
Register	Name	Read/Write	
0	Speed 1	R/W	Motor1 speed (mode 0,1) or speed (mode 2,3)
1	Speed 2/Turn	R/W	Motor2 speed (mode 0,1) or turn (mode 2,3)
2	Enc1a	Read only	Encoder 1 position, 1st byte (highest), capture count when read
3	Enc1b	Read only	Encoder 1 position, 2nd byte
4	Enc1c	Read only	Encoder 1 position, 3rd byte
5	Enc1d	Read only	Encoder 1 position, 4th (lowest byte)
6	Enc2a	Read only	Encoder 2 position, 1st byte (highest), capture count when read
7	Enc2b	Read only	Encoder 2 position, 2nd byte
8	Enc2c	Read only	Encoder 2 position, 3rd byte
9	Enc2d	Read only	Encoder 2 position, 4th byte (lowest byte)
10	Battery volts	Read only	The supply battery voltage
11	Motor 1 current	Read only	The current through motor 1
12	Motor 2 current	Read only	The current through motor 2
13	Software Revision	Read only	Software Revision Number
14	Acceleration rate	R/W	Optional Acceleration register
15	Mode	R/W	Mode of operation (see below)
16	Command	R/W	Used for reset of encoder counts and module address changes

Mode: Υπάρχουν τέσσερις επιλογές για την κίνηση ανάλογα με το αν υπάρχει ανάγκη κίνησης των δύο τροχών σε συνδυασμό ή ανεξάρτητα. Οι επιλογές για τις διάφορες τιμές του καταχωρητή mode είναι:

- ***0***: (Default) Η ταχύτητα του κάθε κινητήρα προσαρμόζεται από τον κάθε αντίστοιχο SPEED καταχωρητή. Οι τιμές της ταχύτητα κυμαίνονται από το 0 έως το 255 όπου 0 (full reverse), 128(stop), 255(full forward).
- ***1*** Ίδιο με την επιλογή 0, εκτός του ότι το εύρος της ταχύτητας κυμαίνεται ως προσημασμένες τιμές όπου -128(full reverse), 0(stop), 127(full forward).
- ***2***: Ο καταχωρητής SPEED1 ελέγχει και τους 2 κινητήρες και ο SPEED2/TURN αντιστοιχεί στη στροφή. Οι ταχύτητες κυμαίνονται από το 0-255.
- ***3***: Όπως το 2, με τη διαφορά ότι το εύρος των τιμών των καταχωρητών κυμαίνεται από το -128 έως το 127.

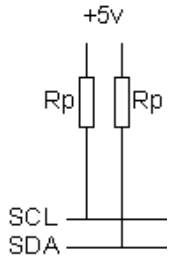
3.1.1 Λίγα λόγια για το I2C Bus

Η Philips Semiconductors ανέπτυξε το I^2C Bus πριν από περίπου 20 χρόνια ως έναν τρόπο επικοινωνίας μεταξύ IC's (Integrated Circuits) χρησιμοποιώντας τον ελάχιστο αριθμό από pins. Είναι ένα απλό bi-directional, 2-wire serial interface bus, το οποίο χρειάζεται μόνο δύο PCB tracks κάτι που έχει ως αποτέλεσμα μικρότερα και φθηνότερα PCBs. Το bus όπως αναφέρθηκε έχει μόνο δύο γραμμές· την **serial data line (SDA)** για τα δεδομένα και την **serial clock line (SCL)** για το ρολόι. Μία απλή περιγραφή του φαίνεται στην ακόλουθη εικόνα.



Εικόνα 16 – I2C Bus

Κάθε συσκευή (slave) που συνδέεται στο bus έχει τη δική της διεύθυνση και έτσι υλοποιείται μία απλή σχέση μεταξύ του master και κάθε slave. Λόγω του ότι η επικοινωνία είναι bidirectional ο master μπορεί να στέλνει αλλά και να λαμβάνει δεδομένα από τις συνδεδεμένες συσκευές. Οι μεταφορές των δεδομένων μπορούν να γίνουν με ταχύτητες έως 100 kbits/s σε standard-mode, μέχρι 400kbits/s σε Fast-mode και μέχρι 3.4 Mbits/s σε High-speed mode. Θα πρέπει επίσης να σημειωθεί ότι για την ορθή λειτουργία του bus θα πρέπει να προστεθούν pull-up resistors μεταξύ της τροφοδοσίας και των γραμμών SDA και SCL όπως φαίνεται αριστερά. Οι τιμές των αντιστάσεων δεν παίζουν κάποιο σημαντικό ρόλο, με τις πιο συνηθισμένες τιμές να είναι 1k8, 4k7 και 10k. Οι pull-up resistors χρειάζονται ώστε οι γραμμές να μπορούν να γίνουν high καθώς σε περίπτωση που δεν υπήρχαν, οι γραμμές SDA και SCL θα ήταν πάντα low κοντά στα 0 volts.



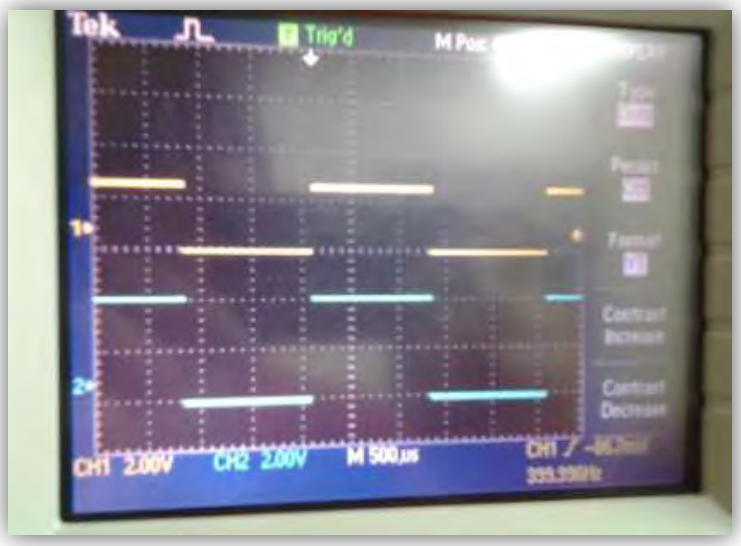
Εικόνα 17 – Pull-up resistors

Στη περίπτωση που εξετάζεται, το ρόλο του master έχει το BB ενώ ο MD25 συνδέεται ως slave πάνω στο bus. Ωστόσο εδώ παρουσιάστηκε ένα βασικό πρόβλημα στην επικοινωνία. Αυτό πηγάζει από το γεγονός υπάρχει ασυμφωνία στα σήματα που ανταλλάσσονται μεταξύ των δύο συσκευών. Τα σήματα SCL και SDA που φεύγουν από το BB είναι πλάτους 1.8V , ενώ αυτά από τον MD25 είναι πλάτους 5V. Αυτό σημαίνει ότι ο MD25 δεν μπορεί να ¨ακούσει¨ το BB και ότι ο MD25 μπορεί να προκαλέσει ζημιά στο BB αν συνδεθεί απευθείας στα pins του expansion header. Πρέπει έτσι τα δύο σήματα με κάποιον τρόπο να έρθουν σε συμφωνία. Αυτό μπορεί να γίνει με τη χρήση ενός ***Logic Level Converter*** το οποίο είναι μία συσκευή που μπορεί να κάνει με μεγάλη ασφάλεια step down ένα σήμα από τα 5V στα 1.8V αλλά και το αντίθετο. Το σημείο αυτό υπήρξε ιδιαίτερα χρονοβόρο μιας και οι πρώτες προσπάθειες που έγιναν με τη βοήθεια του PCA9306 Level Translator απέβησαν άκαρπες. Η λύση τελικά δόθηκε με την αγορά ενός Logic Level

Εικόνα 18 – Logic Level Converter

Converter της εταιρίας Sparkfun με κωδικό BOB-08745 που φαίνεται παραπάνω. Στο converter, χρησιμοποιήθηκαν οι διεπαφές TX καθώς είναι bidirectional όπως ορίζει το πρωτόκολλο I2C και επίσης διαθέτουν τους απαιτούμενους pull-up resistors και στις δύο πλευρές επικοινωνίας.

Η σύνδεση είναι απλή αν και εδώ θα πρέπει κάποιος να είναι εξοικειωμένος με τη συγκόλληση ώστε να προσθέσει τα απαραίτητα headers. Στην πλευρά Low Voltage αρχικά συνδέονται στις επαφές Tx τα σήματα SCL και SDA (pins 24,23 αντίστοιχα από το expansion header του BB) καθώς και η DC 1.8V τάση και η γείωση (pins 1, 28 αντίστοιχα). Με την ίδια λογική στην πλευρά High voltage συνδέονται τα SCL και SDA από τον MD25 καθώς και η γείωση και η τροφοδοσία των 5V που προσφέρεται από τον MD25. Προτού συνδεθεί τον converter στη διάταξη έγιναν δοκιμές στο εργαστήριο, έτσι ώστε να διαπιστωθεί ότι ο converter κάνει αυτό που απαιτείται. Χρησιμοποιώντας την γεννήτρια σήματος έγινε παραγωγή ενός τετραγωνικού σήματος πλάτους 1.8V το οποίο οδηγήθηκε στην πλευρά Low voltage. Στην άλλη πλευρά συνδέθηκε τροφοδοσία >1.8V ρυθμίζοντας σωστά την γεννήτρια τάσης του εργαστηρίου και έγινε παρακολούθηση της εξόδου στον παλμογράφο. Ένα από τα αποτελέσματα φαίνεται στην ακόλουθη εικόνα, όπου το σήμα που φαίνεται πρώτο είναι το σήμα εισόδου και το δεύτερο το ενισχυμένο, με πλάτος ίσο με την τάση high voltage.



Εικόνα 19 – Ενίσχυση σήματος με τη χρήση του converter

Ωστόσο, αν σε αυτό το σημείο προσπαθήσει κανείς να κινήσει τους τροχούς, δεν θα δει κάποιο αποτέλεσμα. Αυτό συμβαίνει για δύο πολύ σημαντικούς λόγους:

- Το BB διαθέτει τρία I2C buses. Το bus που βρίσκεται στο expansion port και χρησιμοποιήθηκε στην υλοποίηση, είναι το δεύτερο από τα τρία, με την default τιμή του ρολογιού SCL να είναι 400khz (Fast Mode). Το πρόβλημα πηγάζει από το γεγονός ότι ο

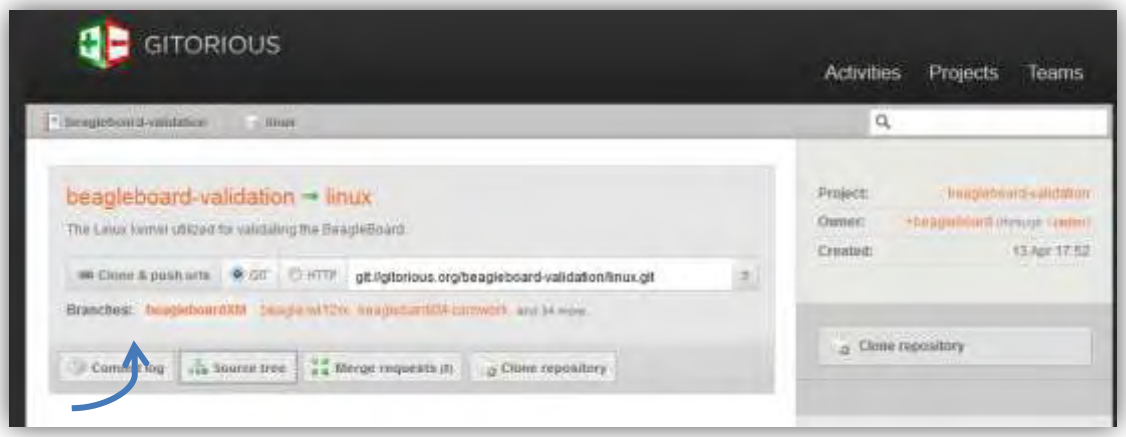
ελεγκτής MD25 απαιτεί ρολόι συχνότητας μέχρι 100khz το πολύ, επομένως με την παρούσα ρύθμιση δεν μπορεί να υπάρξει επικοινωνία.

- Ο δεύτερος λόγος είναι ότι τα 28 pins του expansion port υλοποιούν πολλαπλές λειτουργίες το καθένα. Αυτό γίνεται μέσω ενός πολυπλέκτη 8 σε 1, ο οποίος καθορίζει ποια σήματα θα περάσουν στις εξόδους. Θα πρέπει έτσι να είναι κανείς σίγουρος ώστε στα pins 23 και 24 να περάσουν τα σήματα SDA και SCL αντίστοιχα.

Στη συνέχεια περιγράφονται τα βήματα που πρέπει να ακολουθηθούν, έτσι ώστε να ξεπεραστούν τα παραπάνω προβλήματα.

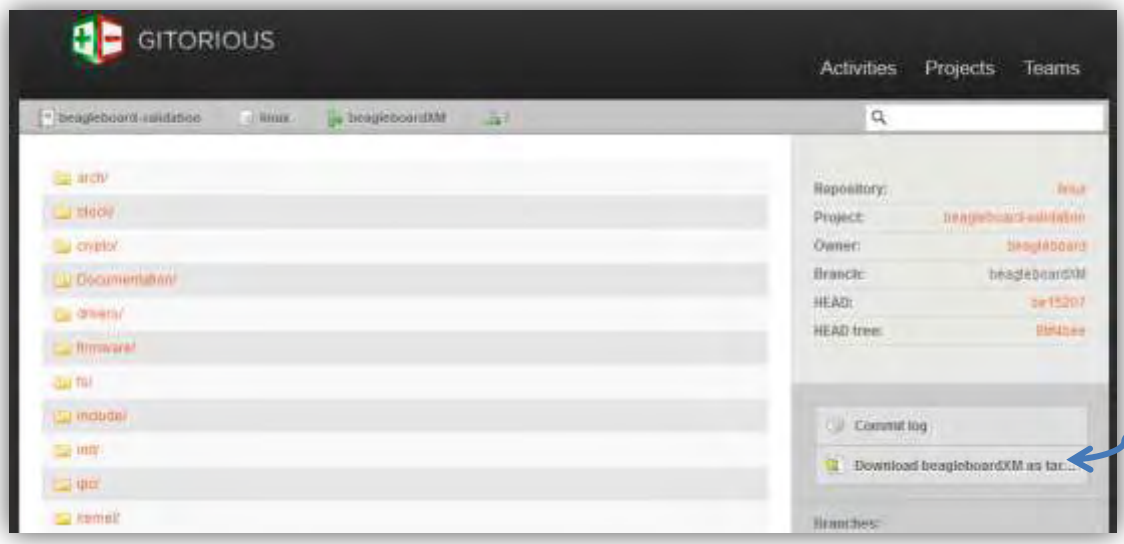
3.1.2 Αλλαγή ταχύτητας του I2C Bus

Ο μόνος τρόπος να αλλαχθεί η ταχύτητα στο I2C Bus, είναι να τροποποιηθεί μία γραμμή σε ένα αρχείο .c στον source κώδικα του Linux Kernel. Αυτό πρακτικά σημαίνει ότι ο kernel έπρεπε να γίνει recompile. Το πρόβλημα που παρουσιάστηκε ήταν ότι ο τρόπος που χρησιμοποιήθηκε για να χτιστεί ο kernel και το File System, δηλαδή το εργαλείο Narcissus, δεν προσφέρει τη δυνατότητα επεξεργασίας του πηγαίου κώδικα του Kernel. Έτσι σε αυτό το σημείο ξεκίνησε ένα μεγάλο ταξίδι ώστε να καταστεί δυνατή η εύρεση του κατάλληλου πηγαίο κώδικα του kernel. Το σημείο αυτό ήταν χρονοβόρο μιας και λόγω των πολλών διαφορετικών εκδόσεων BB που υπάρχουν στην αγορά, έπρεπε με δοκιμές να βρεθεί τελικά ο kernel που θα υποστήριζε με τον καλύτερο τρόπο τη συσκευή BB. Τα διάφορα sources τα υπάρχουν στην ιστοσελίδα gitorious.org, ένα χώρο στον οποίο δραστηριοποιούνται πολλοί developers του BB. Η λύση που επιλέχθηκε ήταν του project beagleboard-validation που βρίσκεται στο link: http://gitorious.org/beagleboard-validation/linux και φαίνεται παρακάτω:



Εικόνα 20 – Gitorious Beagleboard validation

Κάνοντας κλικ στην επιλογή Source Tree γίνεται μετάβαση στην ακόλουθη σελίδα όπου βρίσκεται το συμπιεσμένο αρχείο tar.gz με τον πηγαίο κώδικα.



Μετά την ολοκλήρωση της λήψης του αρχείου πρέπει να βρεθεί το αρχείο */arch/arm/mach-omap2/board-omap3beagle.c* και να αλλαχθεί η γραμμή:

```
omap_register_i2c_bus(2, 400, beagle_i2c2_boardinfo,ARRAY_SIZE(beagle_i2c2_boardinfo));
```

Η τιμή που πρέπει να αλλαχτεί στην κλήση της συνάρτησης είναι η δεύτερη και η τιμή που επιλέχθηκε ήταν αυτή των 50khz, η οποία είναι αρκετή για τη μετάδοση των πληροφοριών που πρέπει να σταλούν πάνω στο Bus. Για την κατασκευή του αρχείο uImage, του Linux Kernel δηλαδή, είναι απαραίτητη η εγκατάσταση κάποιου **Cross Compiler** στο σύστημα. Υπάρχουν αρκετοί τέτοιοι compilers ωστόσο επιλέχθηκε αυτός της εταιρίας CodeSourcery.

Η διαδικασία που ακολουθήθηκε για να γίνει cross compile ο kernel φαίνεται στη συνέχεια και διήρκησε κάποια λεπτά:

```
$ ARCH=arm
$ export ARCH
$ make CROSS_COMPILE=/path/to/arm-angstrom-linux-gnueabi- distclean
$ make CROSS_COMPILE=/path/to/arm-angstrom-linux-gnueabi-
omap3_beagle_defconfig
$make CROSS_COMPILE=/path/to/arm-angstrom-linux-gnueabi- uImage
```

Τελευταίο βήμα είναι η αντιγραφή του αρχείου uImage στην SD κάρτα τόσο στο πρώτο partition όσο και στο δεύτερο στη θέση /Angstrom/boot και έτσι τελικά με τον νέο kernel το I2C Bus τρέχει στα 50khz .

3.1.3 Ρύθμιση των σημάτων στο Expansion port του BB

Όπως αναφέρθηκε και προηγουμένως, τα pins του expansion port μπορούν να υλοποιούν πολλές λειτουργίες, μέσω ενός πολυπλέκτη 8 σε 1, ο οποίος καθορίζει τη δρομολόγηση των σημάτων στις εξόδους. Στον επόμενο πίνακα παρουσιάζονται αναλυτικά οι λειτουργίες που επιτελεί κάθε pin εξόδου στον expansion connector, ανάλογα με την τιμή του πολυπλέκτη.

EXP	OMAP	0	1	2	3	4	5	6	7
1		VIO_1V8							
2		DC_5V							
3	AE3	MMC2_DAT7	*	*	*	GPIO_139	*	*	Z
4	AB26	UART2_CTS	McBSP3_DX	GPT9_PWMEVT	X	GPIO_144	X	X	Z
5	AF3	MMC2_DAT6	*	*	*	GPIO_138	*	X	Z
6	AA25	UART2_TX	McBSP3_CLKX	GPT11_PWMEVT	X	GPIO_146	X	X	Z
7	AH3	MMC2_DAT5	*	*	*	GPIO_137	*	X	Z
8	AE5	McBSP3_FSX	UART2_RX	X	X	GPIO_143	*	X	Z
9	AE4	MMC2_DAT4	*	X	*	GPIO_136	X	X	Z
10	AB25	UART2_RTS	McBSP3_DR	GPT10_PWMEVT	X	GPIO_145	X	X	Z
11	AF4	MMC2_DAT3	McSPI3_CS0	X	X	GPIO_135	X	X	Z
12	V21	McBSP1_DX	McSPI4_SIMO	McBSP3_DX	X	GPIO_158	X	X	Z
13	AG4	MMC2_DAT2	McSPI3_CS1	X	X	GPIO_134	X	X	Z
14	W21	McBSP1_CLKX	X	McBSP3_CLKX	X	GPIO_162	X	X	Z
15	AH4	MMC2_DAT1	X	X	X	GPIO_133	X	X	Z
16	K26	McBSP1_FSX	McSPI4_CS0	McBSP3_FSX	x	GPIO_161	X	X	Z
17	AH5	MMC2_DAT0	McSPI3_SOMI	X	X	GPIO_132	X	X	Z
18	U21	McBSP1_DR	McSPI4_SOMI	McBSP3_DR	X	GPIO_159	X	X	Z
19	AG5	MMC2_CMD	McSPI3_SIMO	X	X	GPIO_131	X	X	Z
20	Y21	McBSP1_CLKR	McSPI4_CLK	X	X	GPIO_156	X	X	Z
21	AE2	MMC2_CLKO	McSPI3_CLK	X	X	GPIO_130	X	X	Z
22	AA21	McBSP1_FSR	X	*	Z	GPIO_157	X	X	Z
23	AE15	I2C2_SDA	X	X	X	GPIO_183	X	X	Z
24	AF15	I2C2_SCL	X	X	X	GPIO_168	X	X	Z
25	25	REGEN							
26	26	nRESET							
27	27	GND							
28	28	GND							

Εικόνα 21 – Expansion Connector Signals

Οι τιμές των πολυπλεκτών που καθορίζουν ποιο σήμα θα περάσει σε κάθε pin καθορίζονται στο αρχείο **u-boot.bin** που χρησιμοποιείται κατά το booting της συσκευής και ρυθμίζει διάφορες παραμέτρους του συστήματος. Όπως και στην περίπτωση του kernel, έτσι και εδώ δεν μπορούν να γίνουν αλλαγές μέσω του εργαλείου Narcissus και πρέπει να βρεθούν οι πηγαίοι κώδικες που δημιουργούν το αρχείο u-boot και να γίνουν cross compile. Ο κώδικες που χρησιμοποιήθηκαν βρίσκονται στο link http://gitorious.org/beagleboard-validation/u-boot. Για να ενεργοποιηθεί το I2C στον expansion connector πρέπει οι τιμές του πολυπλέκτη για τα pins 23 και 24 να είναι σωστά

ρυθμισμένες. Οι τιμές αυτές ορίζονται στο αρχείο *board/ti/beagle/beagle.h* στις ακόλουθες γραμμές:

```
MUX_VAL(CP(I2C2_SCL),           (IEN  | PTU | DIS | M0)) /*I2C2_SCL*/
MUX_VAL(CP(I2C2_SDA),           (IEN  | PTU | DIS | M0)) /*I2C2_SDA*/
```

Όπως φαίνεται και στον προηγούμενο πίνακα, το όρισμα M0 αφήνει να περάσει στον πολυπλέκτη το σήμα για το I2C. Αφού διαπιστώθηκε η σωστή ρύθμιση των εξόδων, ακολούθησε η διαδικασία του cross compile, η οποία δημιούργησε το νέο αρχείο u-boot.bin:

```
$ make CROSS_COMPILE=/path/to/arm-none-linux-gnueabi- mrproper
$ make CROSS_COMPILE=/path/to/arm-none-linux-gnueabi- omap3_beagle_config
$ make CROSS_COMPILE=/path/to/arm-none-linux-gnueabi-
```

Σε αυτό το σημείο έγινε δοκιμή για το αν επικοινωνούν τελικά μεταξύ τους οι δύο συσκευές. Ξεκινώντας θα πρέπει κανείς να βεβαιωθεί ότι είναι εγκατεστημένο στο σύστημα το πακέτο ***i2c-tools***. Σε περίπτωση που κάτι τέτοιο δεν συμβαίνει, μπορεί να εγκατασταθεί τρέχοντας από την κονσόλα την εξής εντολή:

```
$ opkg install i2c-tools
```

Αφού ενεργοποιήθηκε ο ελεγκτής MD25 χρησιμοποιήθηκε η εντολή η εντολή i2cdetect, η οποία ανήκει στο πακέτο i2ctools και σκανάρει το Bus, για να διαπιστωθεί αν η συσκευή εμφανίζεται στο χώρο των διευθύνσεων του διαύλου.

```
root@beagleboard:~# i2cdetect -r 2
WARNING! This program can confuse your I2C bus, cause data loss and
worse!
I will probe file /dev/i2c-2 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n]
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- 58 -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Στο datasheet του ελεγκτή MD25 αναφέρεται ότι η base address είναι η 0xB0, ωστόσο δεν είναι αυτή η τιμή που εμφανίζεται παραπάνω. Αυτό συμβαίνει διότι το πιο σημαντικό bit σε μία διεύθυνση I2C υποδεικνύει αν είναι μία read (1) ή write (0) διεύθυνση, έτσι για να βρεθεί η διεύθυνση που θα εμφανίσει η εντολή **i2cdetect** πρέπει να γίνει αριστερή ολίσθηση της διεύθυνσης που αναφέρει ο κατασκευαστής κατά μία θέση αριστερά. Δηλαδή 0xB0 << 1 = 0x3A = $(58)_{10}$. Για τις αρχικές δοκιμές έγινε χρήση δύο απλών εντολών, της **i2cset** και της **i2cget**, με τις οποίες μπορεί κανείς να γράψει και να διαβάσει αντίστοιχα πάνω στο Bus.

3.2 Κατασκευαστικό στάδιο

Μετά τη σωστή σύνδεση όλων των περιφερειακών, ξεκίνησε η διαδικασία κατασκευής της ρομποτικής διάταξης. Η υλοποίηση βασίστηκε σε ένα παλαιότερο project μου που έγινε στα πλαίσια του μαθήματος των Ενσωματωμένων Συστημάτων, τον E-vlampio ++, ένα ρομπότ που ήταν σε θέση να αποφεύγει εμπόδια και φαίνεται στη διπλανή εικόνα. Για τον σκοπό αυτό δημιουργήθηκε μία εντελώς νέα βάση από πλεξιγκλάς ωστόσο ξανά τύπου ¨σάντουιτς¨ ώστε να μπορέσουν να τοποθετηθούν η μπαταρία ,το μεγαλύτερο μέρος της καλωδίωσης της συσκευής και το Hub ενώ στο επάνω μέρος το BB, ο ελεγκτής MD25 και η κάμερα. Η σχεδίαση ξεκίνησε από λευκό χαρτί και σε πρώτη φάση δημιουργηθήκανε κάποια πρότυπα από χοντρό χαρτόνι τα οποία χρησιμοποιήθηκαν για να κοπεί σωστά το πλεξιγκλάς. Με τις φωτογραφίες που επισυνάπτονται φαίνεται αναλυτικά η όλη διαδικασία.

Εικόνα 22 – Project E-vlampios++

Στη συνέχεια χρησιμοποιήθηκαν τα πρότυπα από χαρτόνι για την κοπή του θα σα

το

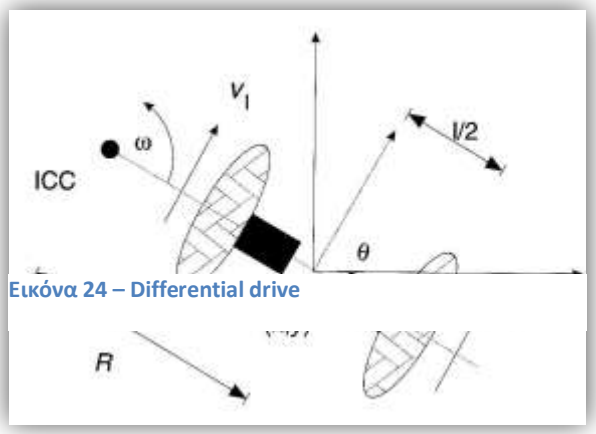
Για την στερέωση των δύο επιπέδων επιλέχθηκαν 4 βίδες των 6 χιλιοστών οι οποίες αφού κοπήκαν στο επιθυμητό μέγεθος, τοποθετήθηκαν σε τοπολογία

Εικόνα 23 – Ball caster

σταυρού ώστε να δημιουργηθεί μία όσο το δυνατόν πιο στιβαρή κατασκευή. Στη συνέχεια έγιναν όλες οι απαιτούμενες τρύπες με χρήση τρυπανιού. Σε όλη τη διαδικασία δόθηκε ιδιαίτερη προσοχή, με αποτέλεσμα μία πολύ σωστή δουλειά. Το προηγούμενο project διέθετε τέσσερις τροχούς οι οποίοι ελέγχονταν από δύο ελεγκτές MD25. Ωστόσο στην παρούσα κατασκευή επιλέχτηκε η χρήση ενός μόνο σετ τροχών για λόγους βάρους, χώρου, ευκολίας στους ελιγμούς και το κυριότερο, χαμηλότερης κατανάλωσης ενέργειας, μιας και ήδη λόγω των πολλών περιφερειακών υπάρχει αυξημένη κατανάλωση. Αργότερα γίνεται αναφορά αναλυτικά στον τομέα της ενέργειας. Οι δύο τροχοί τοποθετήθηκαν στη μέση της κατασκευής με ζητούμενο την καλύτερη ισορροπία και τη δυνατότητα ελιγμού του ρομπότ γύρω από τον άξονα του. Έμενε λοιπόν να βρεθεί μία λύση για την στήριξη του εμπρός και πίσω μέρους του ρομπότ. Αυτό που επιλέχθηκε ήταν ένα set από **Ball Caster** της εταιρίας Sparkfun και φαίνονται δίπλα. Κάπου εδώ τελείωσε η κατασκευή βασικού πλαισίου, με πολλές προσθήκες ωστόσο να γίνονται στη συνέχεια και θα περιγραφούν παρακάτω. Το αποτέλεσμα που επιτεύχθηκε μετά από δύο εξαντλητικές ημέρες φαίνεται στη συνέχεια.

3.3 Κινηματική

Η επιλογή να προστεθούν δύο τροχοί στη ρομποτική διάταξη, σημαίνει ότι θα χρησιμοποιηθεί το διαφορικό σύστημα οδήγησης ως μηχανισμός κίνησης. Αποτελείται από δύο τροχούς που βρίσκονται πάνω σε έναν κοινό άξονα, όπως συμβαίνει και στην κατασκευή, με τον κάθε τροχό να μπορεί να οδηγηθεί ανεξάρτητα από τον άλλο είτε εμπρός είτε πίσω. Προκειμένου να μπορεί το ρομπότ να εκτελέσει κυκλική κίνηση θα πρέπει να περιστρέφεται γύρω από ένα σημείο που βρίσκεται κατά μήκος του κοινού άξονα των δύο τροχών. Το σημείο αυτό είναι γνωστό ως ICC ή αλλιώς Instantaneous Center of Curvature φαίνεται στη διπλανή εικόνα. Μεταβάλλοντας τις ταχύτητες των ρομπότ, μεταβάλλονται και οι τροχιές του ρομπότ. Επειδή η γωνιακή ταχύτητα ω πρέπει να ίδια και για τους δύο τροχούς γύρω από το ICC, προκύπτουν οι σχέσεις:



Εικόνα 24 – Differential drive

$$\omega\left(R+\frac{l}{2}\right)=V_\tau \ (1) \ \kappa\alpha\iota \ \omega\left(R-\frac{l}{2}\right)=V_l \ (2)$$

όπου l είναι η απόσταση μεταξύ των κέντρων των δύο τροχών, v_τ , v_l οι ταχύτητες του δεξιού και αριστερού τροχού αντίστοιχα κατά μήκος του επιπέδου και R η απόσταση από το ICC μέχρι το μέσο του άξονα. Σε κάθε χρονική στιγμή αν λύσω το παραπάνω σύστημα ως R ή ω, παίρνω:

$$R=\frac{\frac{l}{2}(V_l-V_\tau)}{V_\tau-V_l} \ \ \kappa\alpha\iota \ \ \omega=\ (V_\tau-V_l)/l$$

Από τις παραπάνω σχέσεις μπορώ να διακρίνω εύκολα τις τρεις περιπτώσεις που παρουσιάζουν ενδιαφέρον για την περίπτωση του ρομπότ που κατασκευάστηκε:

- Αν $V_l=V_\tau$, τότε θα εκτελεί το ρομπότ προς τα εμπρός είτε προς τα πίσω, γραμμική κίνηση πάνω σε ευθεία γραμμή. Το R θα τείνει στο άπειρο και θα υπάρχει σχεδόν μηδενική περιστροφή- το ω θα είναι μηδέν.
- Αν $V_l=-V_\tau$ τότε το R=0 και θα υπάρχει περιστροφή γύρω από το μέσο του διαστήματος μεταξύ άξονα τροχών- επί τόπου περιστροφή.

3.4 Power Supply για την κατασκευή

Η κατασκευή της τροφοδοσίας του ρομπότ αποτέλεσε ένα σημαντικό στάδιο κατά την κατασκευή. Ζητούμενο ήταν η κατασκευή να μπορεί να είναι αυτόνομη λειτουργώντας με μπαταρία για ένα εύλογο χρονικό διάστημα. Η μπαταρία που επιλέχθηκε τελικά για τον σκοπό αυτό ήταν μία **Sealed Lead Acid** μπαταρία μολύβδου 12V 7Ah. Στο κύκλωμα που θα κατασκευαζόταν θα έπρεπε να υπάρχουν τρεις διαφορετικές τιμές τάσεως, 12V για τον ελεγκτή MD25 και τους τροχούς, 5V για το BB και τα περιφερειακά του καθώς και 2.2V για τα LEDs. Για τη λειτουργία του BB και των περιφερειακών έπρεπε να τροφοδοτηθεί με την κατάλληλη τάση το HUB.

Εικόνα 25 – SLA battery 12V 7Ah

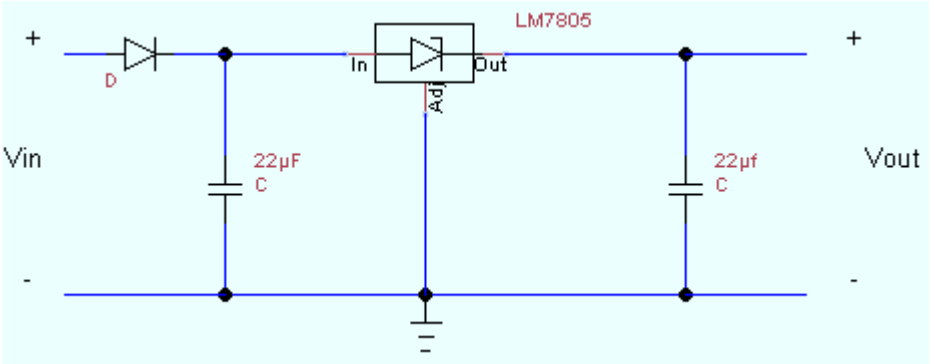
Όσον αφορά τον τρόπο που θα γινόταν η τροφοδοσία υπήρχαν δύο επιλογές. Η μία ήταν η χρήση απλών Linear Voltage Regulators και η δεύτερη ήταν η χρήση κάποιου DC/DC converter η οποία αποτελεί μία πιο κομψή λύση.

Το κύκλωμα θα πρέπει να είναι σε θέση να προσφέρει το απαραίτητο ρεύμα που χρειάζεται η ρομποτική διάταξη, το οποίο ανέρχεται σε ***1.7 A*** στη χειρότερη περίπτωση. Στης συνέχεια φαίνεται αναλυτικά μία εκτίμηση του ρεύματος που χρειάζεται κάθε συσκευή για τη λειτουργία της:

- <=500mA για το BB
- Περίπου 200mA για την webcam
- Περίπου 200mA για την ασύρματη σύνδεση
- 40mA για τον φωτισμό
- < 1 A για τους τροχούς


Εικόνα 26 – LM7805 circuit 1

- Στα πρώτα στάδια έγινε προσπάθεια τροφοδοσίας της κατασκευής με τη χρήση του linear voltage regulator LM7805 των 5V και σχεδιάστηκε το παρακάτω κύκλωμα:



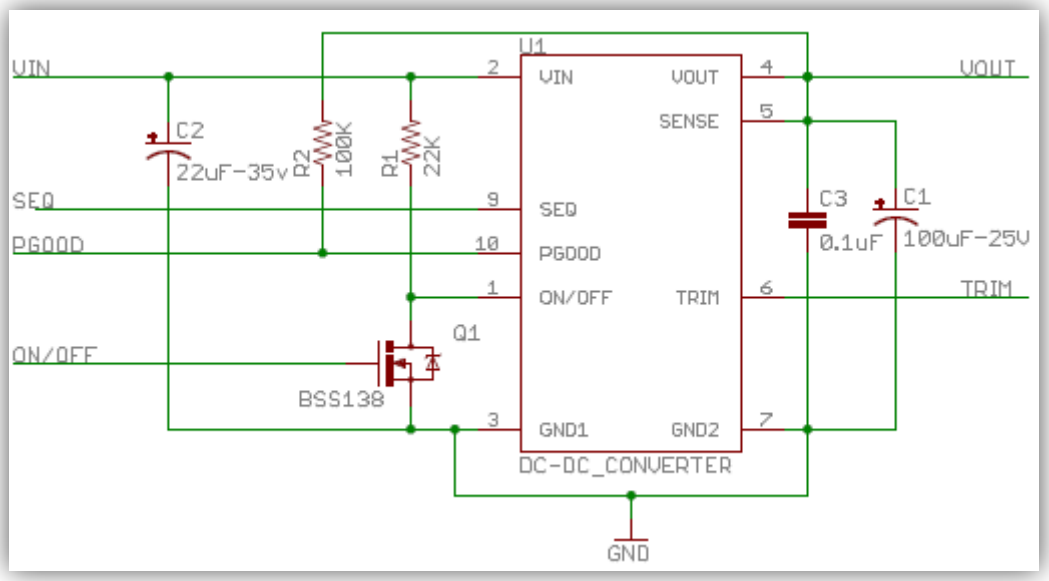
Εικόνα 27 - LM7805 circuit 2

Όπως είναι εμφανές και από το παραπάνω

κύκλωμα, η χρήση των Linear Voltage regulators είναι πολύ απλή, ωστόσο απορρίφθηκε αμέσως καθώς το μέγιστο ρεύμα που μπορεί να προσφέρει στο κύκλωμα μου είναι της τάξης του 1A, ενώ η κατασκευή χρειάζεται από 1A και πάνω. Επίσης αυτοί οι regulators δεν είναι καθόλου αποδοτικοί, ειδικά για μεγάλα ρεύματα, μιας και αυτό που κάνουν είναι να μετατρέπουν την επιπλέον τάση στην είσοδο σε θερμότητα. Η *θερμότητα* που παράγεται μπορεί να υπολογιστεί με τη σχέση: $(Vin - Vout) * I$ το οποίο στην περίπτωση του μέγιστου ρεύματος που μπορεί να δώσει ισούται με (12V -5V) * 1 = 7Watts. Σύμφωνα με το data sheet του LM7805 η θερμοκρασία του regulator αυξάνεται κατά 65^o για κάθε Watt, το οποίο σημαίνει ότι η θερμοκρασία λειτουργίας ανέρχεται σε $7W * \frac{65^o}{W} + 25^o = 480^o$, όπου 25^o είναι η θερμοκρασία δωματίου. Η τιμή αυτή είναι απαγορευτική καθώς όπως αναφέρεται στο data sheet, η μέγιστη θερμοκρασία είναι αυτή των 125^o. Επίσης η *απόδοση* ενός τέτοιου regulator ισούται περίπου με $Vout/Vin = 5/12 = \mathbf{42}\%$, μία τιμή υπερβολικά χαμηλή αφού το μεγαλύτερο μέρος της ενέργειας που προσφέρει η μπαταρία μετατρέπεται σε θερμότητα.

- Ως τελική λύση επιλέχθηκε η χρήση ενός **DC/DC Converter**.Αρχικά έγινε προσπάθεια να κατασκευαστεί από τον εαυτό μου το κύκλωμα του Converter, ωστόσο λόγω δυσκολίας στην εύρεση των συστατικών μερών αποφασίστηκε η αγορά του **converter BOB-09370** της εταιρίας Sparkfun, που φαίνεται στη διπλανή φωτογραφία. Το module αυτό είναι σε θέση να μετατρέψει οποιαδήποτε DC τάση εισόδου μεταξύ **4.5 -14VDC** σε κάθε επιθυμητή τάση μεταξύ **0.59 και 5.5VDC** και μπορεί να προσφέρει μέχρι **6A** ρεύματος. Για να τεθεί η επιθυμητή τάση εξόδου, μία αντίσταση (RTrim όπως αποκαλείται) θα πρέπει να τοποθετηθεί μεταξύ των pin TRIM και GND και το pin ON να γίνει high. Αυτό βεβαίως προϋποθέτει ότι πρέπει και εδώ να συγκολληθούν male headers ώστε να καταστεί δυνατή η σύνδεση. Στο ακόλουθο schematic φαίνεται η εσωτερική δομή της συσκευής

Εικόνα 28 - BOB-09370 DC/DC Converter



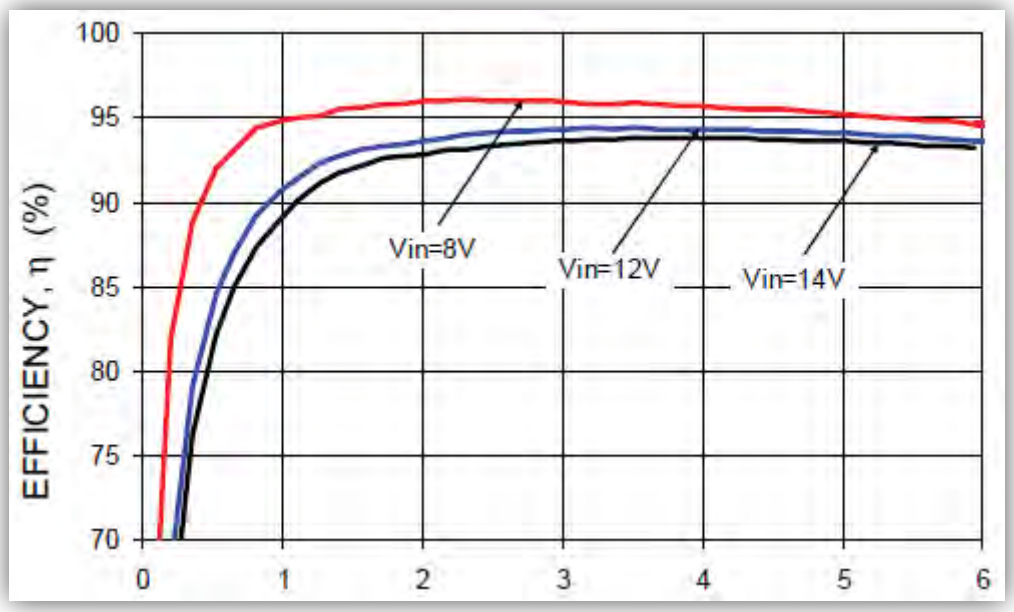
Εικόνα 29 – DC/DC converter schematic

$V_{O,set}$ (V)	*Rtrim* (KΩ)
0.6	656.7
1.0	14.45
1.2	9.704
1.5	6.502
1.8	4.888
2.5	3.096
3.3	2.182
5.0	1.340

Εικόνα 30 – Rtrim values

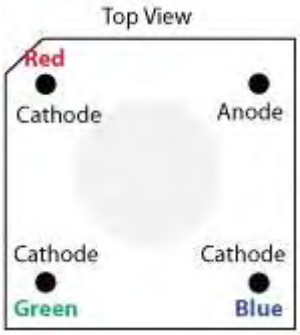
Αν δεν υπάρχει κάποια αντίσταση συνδεδεμένη μεταξύ TRIM και GND τότε η Vout ισούται με 0.59VDC. Για να υπολογιστεί η τιμή της αντίστασης RTrim μπορεί να χρησιμοποιηθεί ο τύπος: $\boldsymbol{RTrim = [5.91/(Vo - 0.591)]k\Omega}$. Ο πίνακας που παρουσιάζεται στη συνέχεια και προέρχεται από το data sheet του converter παρουσιάζει αναλυτικά ποιες πρέπει να είναι οι τιμές των αντιστάσεων ανάλογα με την τάση εξόδου. Για την τιμή των 5V που χρειάζεται, πρέπει να χρησιμοποιηθεί μια RTrim των 1,340kΩ. Για την επίτευξη αυτής της τιμή ακριβώς, συνδέθηκαν σε σειρά μία αντίσταση 1k και ένα ποτενσιόμετρο για μεγαλύτερη ακρίβεια.

Το μεγαλύτερο πλεονέκτημα των Switched Mode Regulators είναι το γεγονός ότι είναι πάρα πολύ αποδοτικοί, μέσα σε ένα εύρος από 85% μέχρι 95%. Αυτό σημαίνει ότι σε σύγκριση με την περίπτωση των Linear Regulators υπάρχει μία τρομακτική διαφορά στην αυτονομία της μπαταρίας, μιας και ένα πολύ μικρό ποσοστό μετατρέπεται σε θερμότητα και επίσης είναι σε θέση να παρέχουν την τιμή του ρεύματος που απαιτείται. Στην επόμενη γραφική παράσταση παρουσιάζεται η αποδοτικότητα (Efficiency) σε σχέση με το ρεύμα εξόδου I_o , για διαφορετικές τιμές στην τάση εισόδου. Εύκολα βλέπει κανείς ότι για την τάση εισόδου των 12V και για ρεύμα >1A που χρειάζεται η κατασκευή, η αποδοτικότητα βρίσκεται πάνω από το ***90%***.



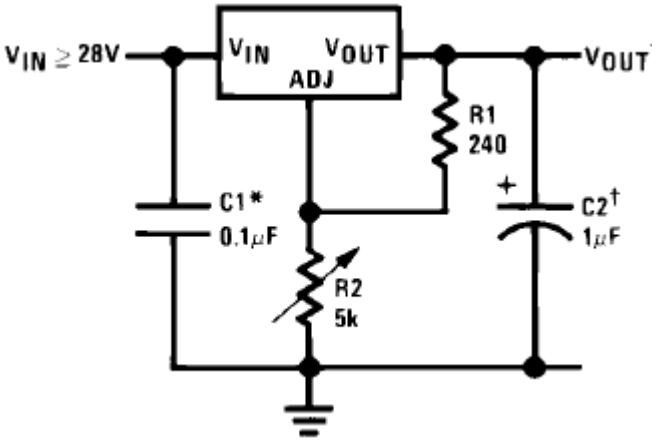
Εικόνα 31 – DC/DC converter efficiency

Επόμενο βήμα στην εργασία ήταν η κατασκευή ενός κυκλώματος για την τροφοδοσία των LEDs που χρησιμοποιήθηκαν για τον φωτισμό της κατασκευής. Για τον σκοπό αυτό χρησιμοποιήθηκαν δύο 5mm RBG Super Flux LEDs τα οποία είναι πολύ φωτεινά και είναι σε θέση να εκπέμπουν τρία χρώματα ανάλογα με τη συνδεσμολογία. Γι αυτό το λόγο διαθέτουν και τέσσερις διεπαφές. Κάθε χρώμα απαιτεί την δική του τάση τροφοδοσίας, όπως φαίνεται και στον ακόλουθο πίνακα:



Forward Voltage	Value
Red	2.0-2.2V
Green	3.0-3.2V
Blue	3.0-3.2V

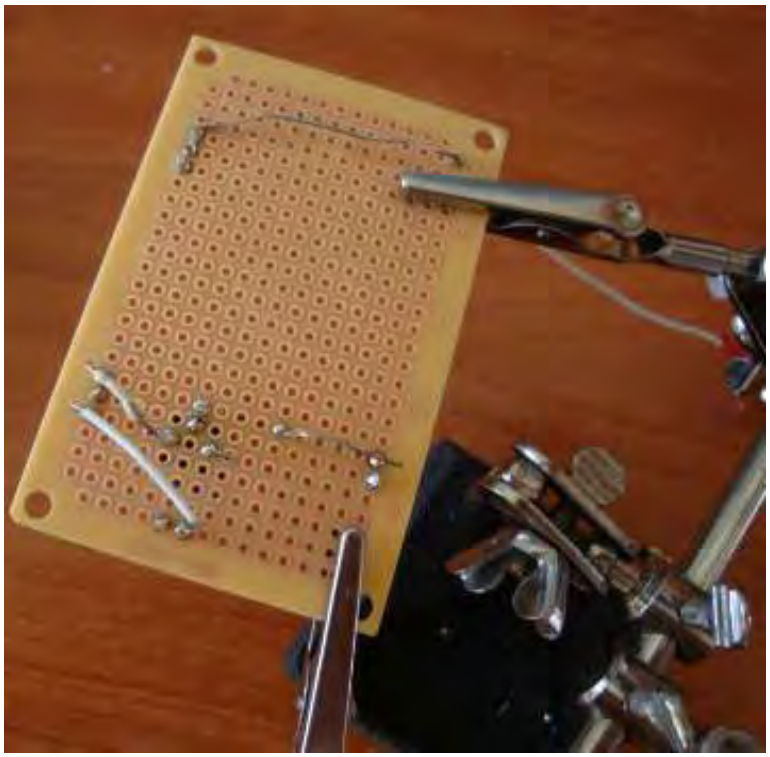
Για τη δημιουργία του κυκλώματος χρησιμοποιήθηκε ο Adjustable Regulator LM317 ο οποίος είναι σε θέση να βγάζει στην έξοδο διάφορες τιμές τάσεων. Το data sheet της συσκευής προσφέρει όλες τις απαραίτητες οδηγίες για την κατασκευή του κυκλώματος, ένα παράδειγμα του οποίου φαίνεται στη διπλανή εικόνα. Η τάση εξόδου μπορεί να αλλάξει μεταβάλλοντας την τιμή της αντίστασης R2. Ως είσοδος επιλέχτηκε η τάση των 5V που δημιουργήθηκε προηγουμένως, με



Εικόνα 32 – LM317 circuit

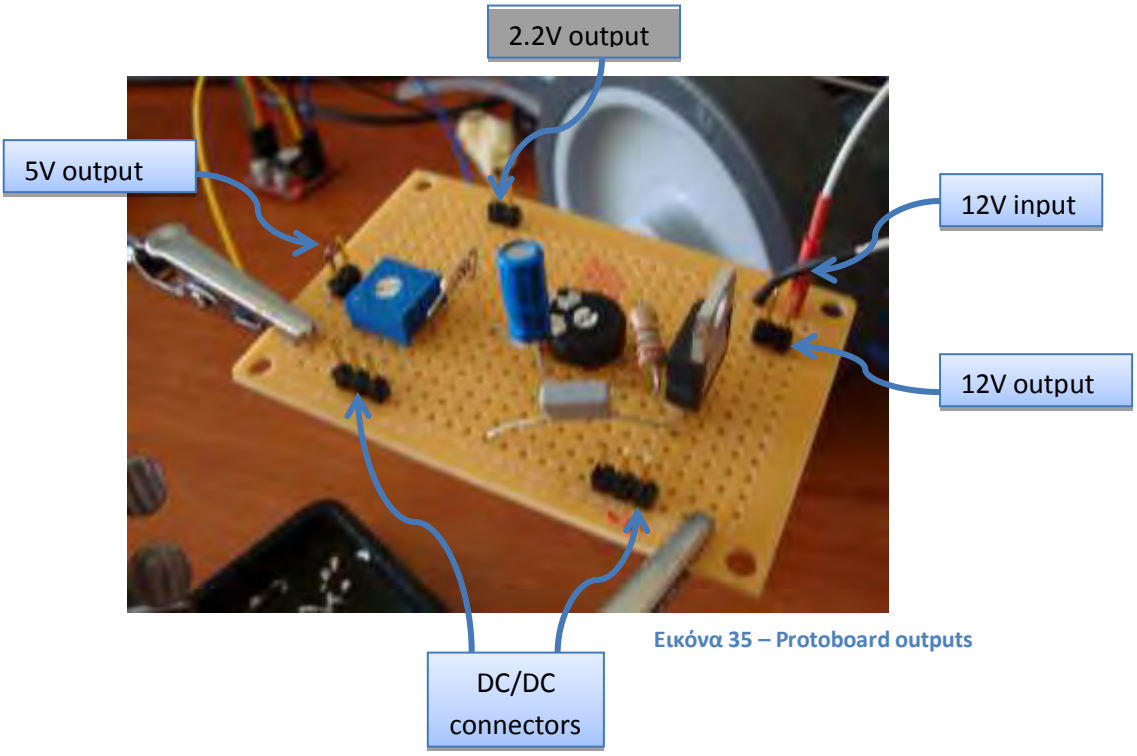
στόχο την καλύτερη δυνατή αποδοτικότητα, μιας και όπως εξηγήθηκε και προηγουμένως, όσο μικρότερη είναι η διαφορά μεταξύ εισόδου και εξόδου, τόσο καλύτερη είναι και η αποδοτικότητα.

Η τελική υλοποίηση του συστήματος τροφοδοσίας έγινε με τη βοήθεια ενός protoboard πάνω στο οποίο σχεδιάστηκαν τα παραπάνω κυκλώματα, ενώ προστέθηκαν και οι ανάλογοι διακόπτες. Ακολουθούν στιγμιότυπα από την κατασκευή και την τελική τοποθέτηση πάνω στο ρομπότ.



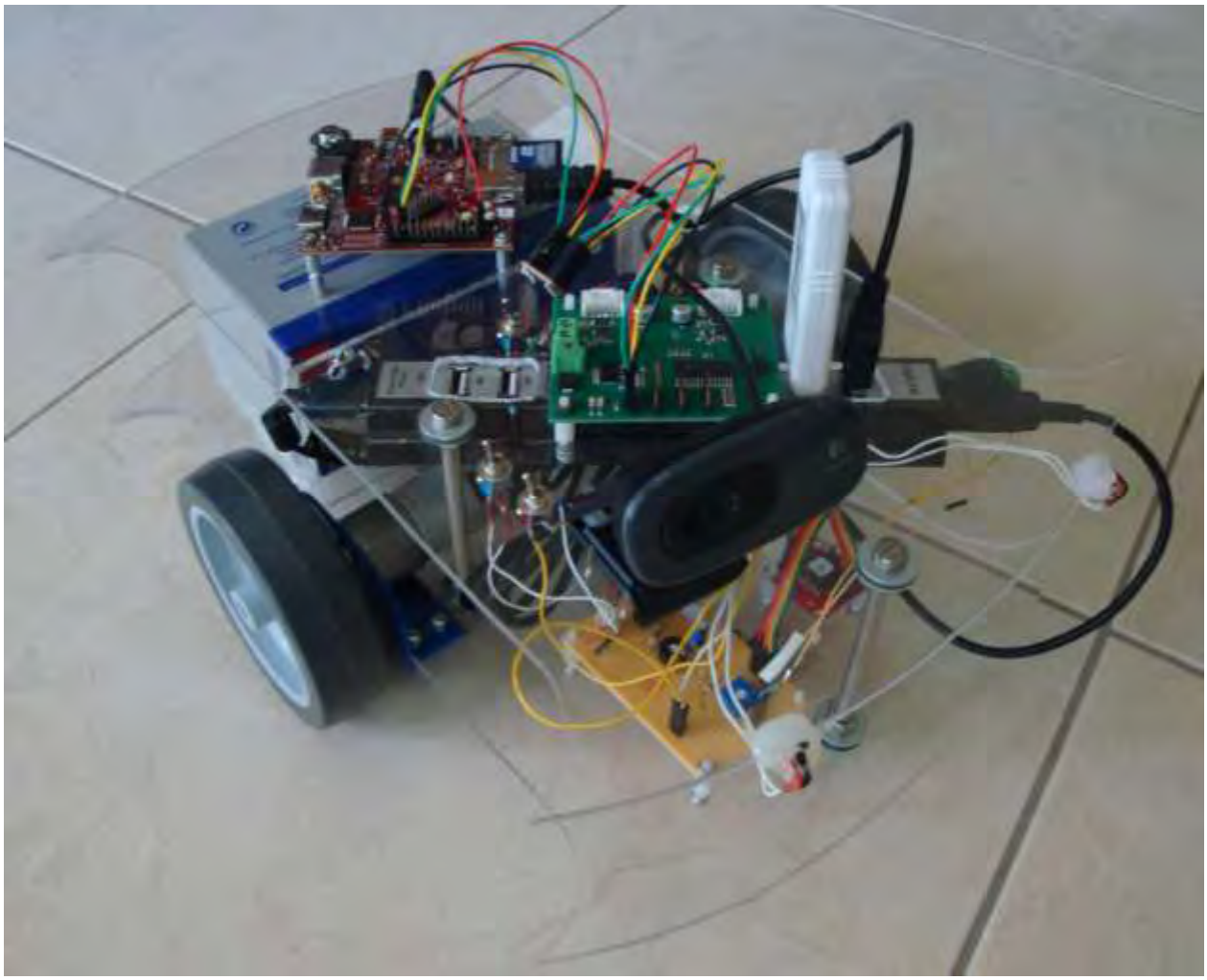
Εικόνα 34 – Protoboard soldering

Εικόνα 33 – Complete power supply



Εικόνα 35 – Protoboard outputs

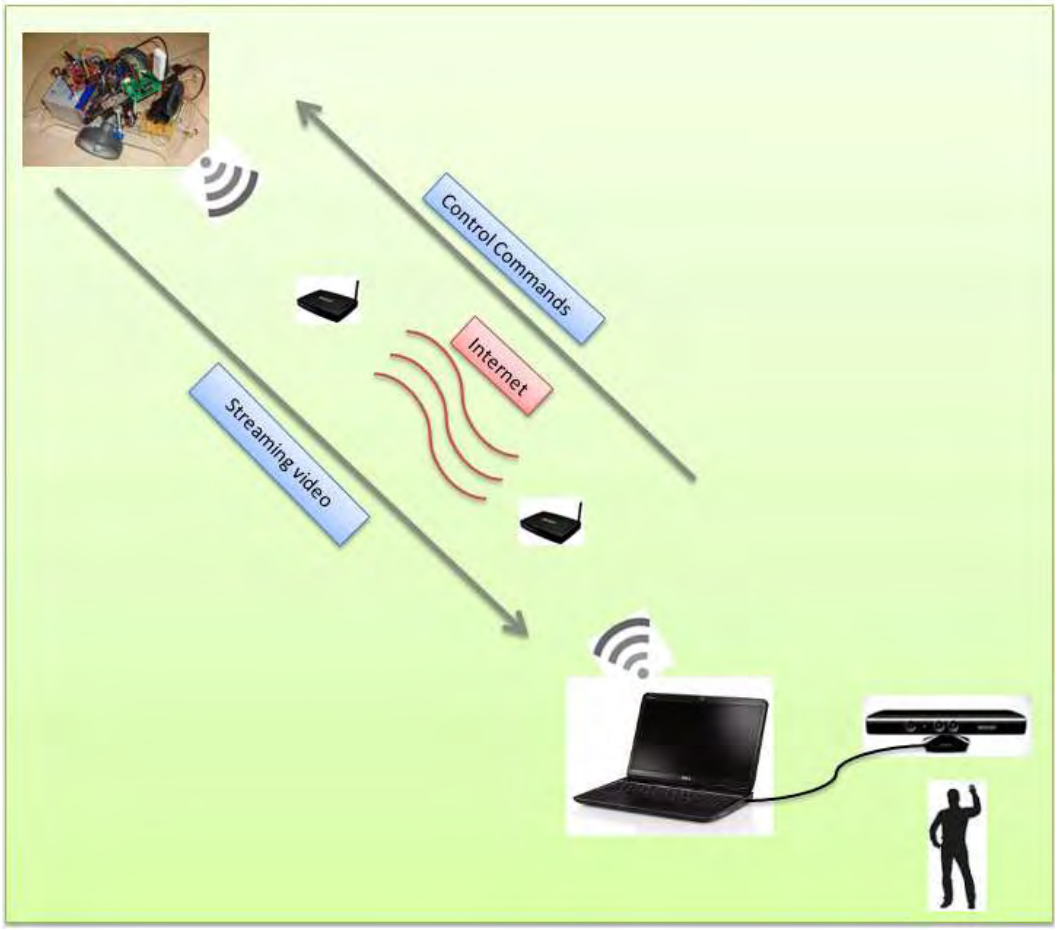
Στη συνέχεια φαίνεται ολοκληρωμένη η ρομποτική διάταξη που κατασκευάστηκε.



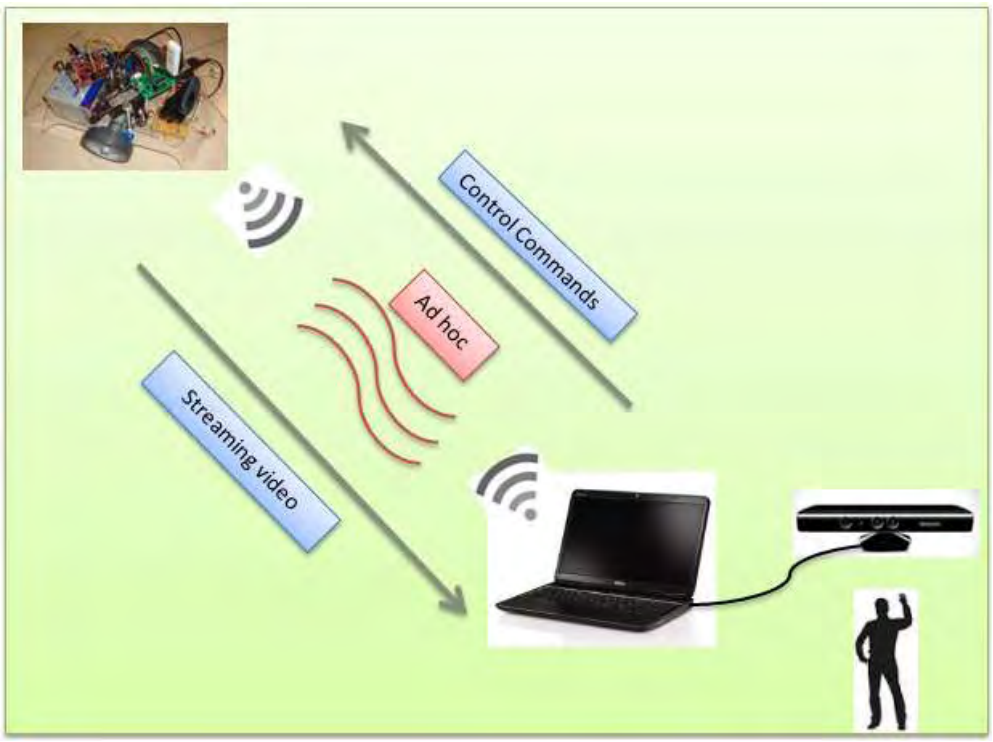
Εικόνα 36 – Η ρομποτική διάταξη ολοκληρωμένη

4. BeagleBoard Software

Μετά την επιτυχή ολοκλήρωση του κατασκευαστικού σταδίου και της επιτυχής οργάνωσης όλου του συστήματος, ακολούθησε ο προγραμματισμός των δύο εφαρμογών του project, μία για την πλατφόρμα BB και μία στον host υπολογιστή με τη χρήση του Kinect. Η εφαρμογή στον υπολογιστή πρέπει να είναι σε θέση να ανιχνεύει με τη βοήθεια του Kinect τις κινήσεις του χρήστη και να τις μεταφράζει σε ανάλογες εντολές κίνησης, τις οποίες στέλνει μέσω δικτύου στο BB. Από την πλευρά του, το BB μόλις λάβει τις εντολές και τις αποκωδικοποιήσει σωστά, κινεί τους τροχούς του ρομπότ. Επίσης το BB αναλαμβάνει να κάνει streaming video και να στέλνει διάφορες τιμές που μπορεί να ζητά η απομακρυσμένη εφαρμογή. Στο κεφάλαιο αυτό θα παρουσιαστεί η διαδικασία προγραμματισμού του BB, ενώ ο έλεγχος του ρομπότ με τη χρήση του Kinect θα ακολουθήσει στο επόμενο κεφάλαιο. Στα πλαίσια της εργασίας, δοκιμάστηκε επικοινωνία μεταξύ των δύο συσκευών μέσω του Internet, αλλά και μέσω δικτύου ad-hoc που στήθηκε μεταξύ τους. Ωστόσο λόγω προφανώς των διαφορετικών λειτουργικών συστημάτων, η επιλογή του ad hoc δεν είναι αξιόπιστη, καθώς ενώ αρχικά όλα δουλεύουν σωστά, έπειτα από λίγο ο host υπολογιστής δεν μπορεί πλέον να ανιχνεύσει το δίκτυο. Στις παρακάτω εικόνες φαίνονται συνοπτικά τα δύο σενάρια λειτουργίας που εξέτασα.



Εικόνα 37 – Απομακρυσμένος έλεγχος ρομπότ μέσω Internet



Εικόνα 38 – Έλεγχος ρομπότ μέσω δικτύου Ad Hoc

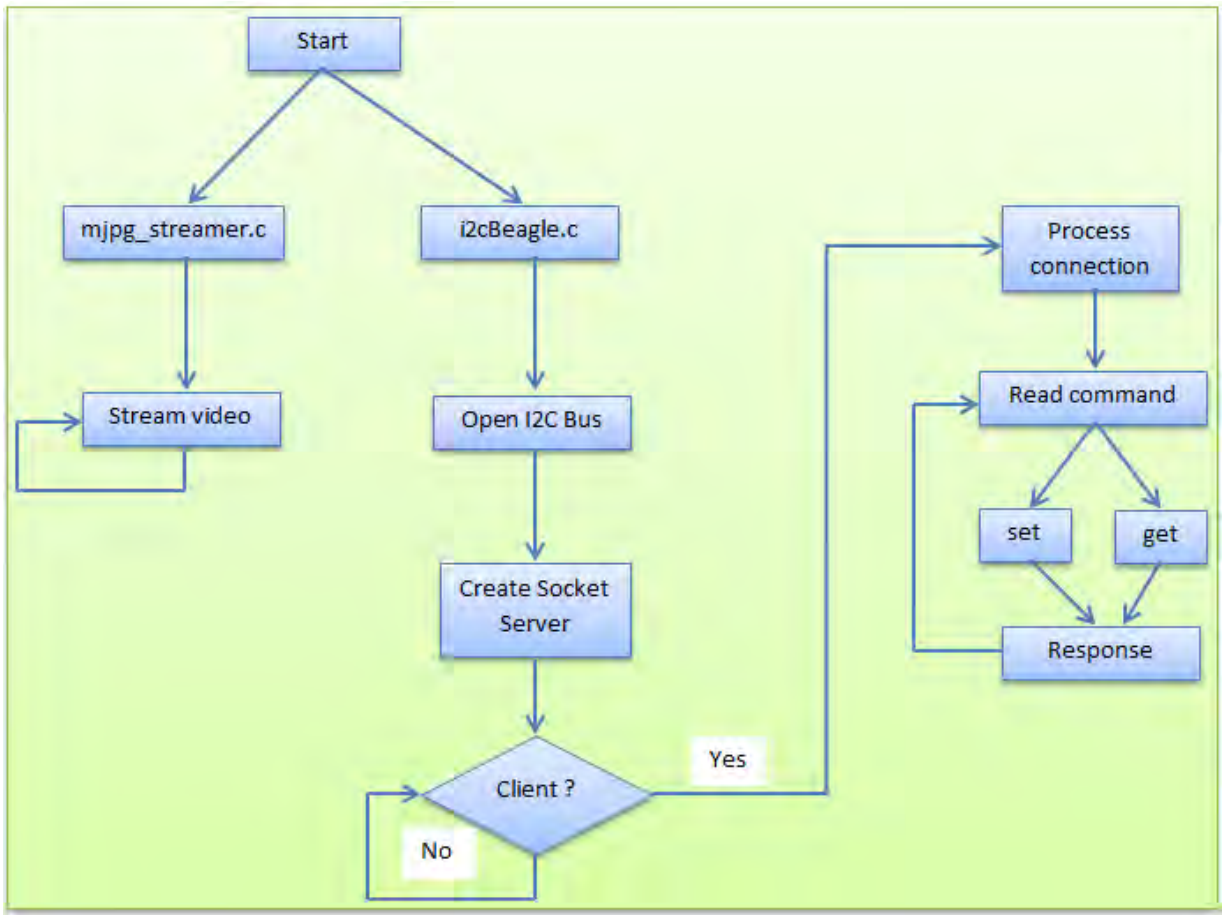
Ο κώδικας και στις δύο περιπτώσεις είναι κοινός μιας και το μόνο που διαφέρει είναι το μέσο επικοινωνίας, δηλαδή το δίκτυο. Αυτό συμβαίνει γιατί ο τρόπος επικοινωνίας και στις δύο περιπτώσεις είναι ο ίδιος (TCP Sockets).

4.1 Ανάπτυξη εφαρμογής στο Beagleboard

Όπως αναφέρθηκε και αρχικά, η επιλογή της διανομής Angstrom ως το λειτουργικό σύστημα του BB, έχει ένα μεγάλο πλεονέκτημα, την ύπαρξη ενός native gcc compiler. Αυτό σημαίνει το development μπορεί να γίνει πιο εύκολο μιας και μπορεί να γίνει απευθείας πάνω στη συσκευή χωρίς τη διαμεσολάβηση κάποιου Cross-compiler.Η συγγραφή κώδικα έγινε σε **γλώσσα C**, ωστόσο θα πρέπει να υπάρχει εγκατεστημένο στη σύστημα το πακέτο **task-native-sdk** που περιλαμβάνει τον gcc compiler. Σε περίπτωση που αυτό δε συμβαίνει, μπορεί εύκολα να εγκατασταθεί με την ακόλουθη εντολή από την κονσόλα:

```
opkg install task-native-sdk
```

Στη συνέχεια φαίνεται το διάγραμμα ροής που αναπαριστά τις λειτουργίες που επιτελεί το BB:



Εικόνα 39 – Διάγραμμα ροής για το Beagleboard

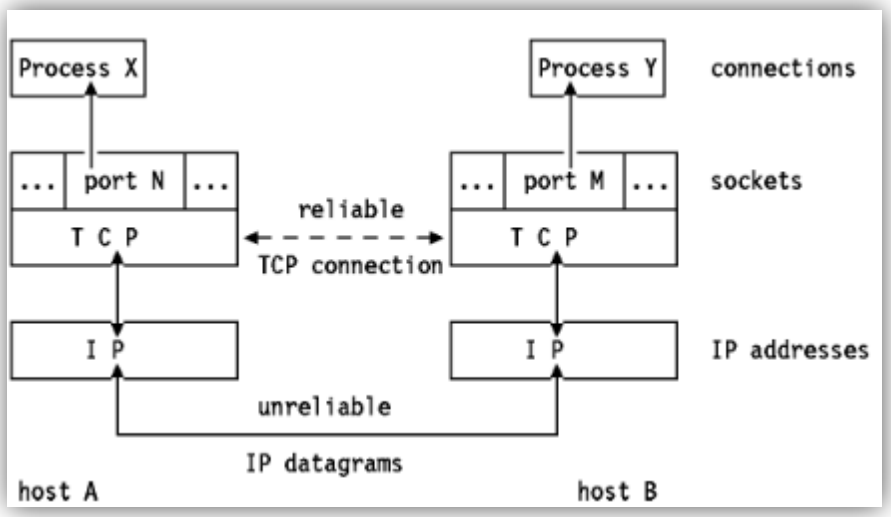
- **Open I2C Bus:** Κάθε συσκευή που συνδέεται πάνω στο δίαυλο I2C φέρει ένα μοναδικό αριθμό (τη διεύθυνση του), επομένως προτού ξεκινήσει κανείς να δουλεύει με το I2C Bus θα πρέπει να γνωρίζει το αναγνωριστικό της συσκευής. Αυτό μπορεί να γίνει με τον τρόπο που περιέγραψα στο κεφάλαιο 3 χρησιμοποιώντας την εντολή i2cdetect, η οποία παρουσιάζει όλες τις συνδεδεμένες συσκευές πάνω στο δίαυλο. Η i2cdetect αποτελεί μέρος του πακέτου i2c-tools που έχει εγκατασταθεί στο σύστημα. Ας δούμε στη συνέχεια πως μπορεί να προσπελαστεί μία i2c συσκευή από ένα πρόγραμμα C. Η πρώτη κίνηση είναι να προστεθεί στο πρόγραμμα μου μία αναφορά στο αρχείο <linux/i2c-dev.h".

```
#include <linux/i2c-dev.h>
```

Για την προσπέλαση μιας συσκευής θα χρειαστεί το I2C device file το οποίο είναι ένα character device file και συνήθως αποκαλείται “i2c-%d” (i2c-0, i2c-1, …). Όπως έχει αναφερθεί και αρχικά, το BB διαθέτει τρεις διαύλους i2c και εγώ χρησιμοποιώ τον δεύτερο, δηλαδή τον **i2c-2**. Το άνοιγμα ενός τέτοιου device file γίνεται ως εξής:

```
char devicePath[20];
int i2c_file;
int bus = 2;


sprintf(devicePath,"/dev/i2c-%d", bus);
printf("Opening i2c device %s\n", devicePath);
if((i2c_file = open(devicePath, O_RDWR)) < 0)
{
    printf("Failed to open the bus!\n");
    exit(1);
}
```

- **Create Socket Server:** Για την προσθήκη της ασύρματης λειτουργικότητας στο ρομπότ, δημιουργήθηκε μία επικοινωνία client-server μεταξύ του host υπολογιστή και του BB αντίστοιχα. Στην πλευρά του BB αναπτύχθηκε κώδικας για τη δημιουργία ενός Server, με χρήση TCP/IP sockets. Η επικοινωνία δύο ή περισσότερων μηχανών με σκοπό την ανταλλαγή δεδομένων, παίζει πάρα πολύ σημαντικό ρόλο στην ανάπτυξη Software, με το TCP/IP να είναι ο πιο ευρέως διαδεδομένος τρόπος που έχει υιοθετηθεί για αυτήν την επικοινωνία. Ο λόγος που επιλέχθηκε έναντι του UDP, είναι το γεγονός ότι το TCP πρωτόκολλο προσφέρει error correction, το οποίο σημαίνει δεν υπάρχει περίπτωση να χαθούν πακέτα δεδομένων στο δίκτυο. Το μοντέλο client-server που χρησιμοποιήθηκε αποτελεί το πιο σύνηθες προγραμματιστικό σενάριο όταν χρησιμοποιούνται TCP sockets. Η συνήθης λειτουργία ενός Server είναι ότι αποτελεί την master διεργασία η οποία ξοδεύει τον περισσότερο χρόνο της ακούγοντας clients. Όταν κάποιος client συνδέεται σε έναν server κάνει κάποιο request και αναμένει κάποια απάντηση. Είναι σημαντικό ότι ένας client θα πρέπει να γνωρίζει από πριν τη διεύθυνση του server που επιθυμεί να επικοινωνήσει κάτι που δεν ισχύει για τον server. Στο ακόλουθο σχήμα δίνεται μία αφαιρετική άποψη αυτού του τρόπου επικοινωνίας. Και οι δύο πλευρές κατά τη δημιουργία της σύνδεσης, χρησιμοποιούν μία πολύ βασική δομή, το socket. Κάθε άκρο επικοινωνίας διαθέτει το δικό του socket με δική του διεύθυνση το καθένα, η οποία ορίζεται με βάση την 32-bit IP address σε συνδυασμό με έναν αριθμό θύρας (port number) 16-bit. Τα sockets είναι bidirectional, δηλαδή κάθε άκρο μπορεί να στέλνει αλλά και να λαμβάνει δεδομένα. Επίσης τα sockets δεν χρησιμοποιούνται μόνο για επικοινωνία πάνω σε δίκτυο, αλλά μπορούν να χρησιμοποιηθούν και για επικοινωνία μεταξύ διεργασιών στο ίδιο σύστημα.

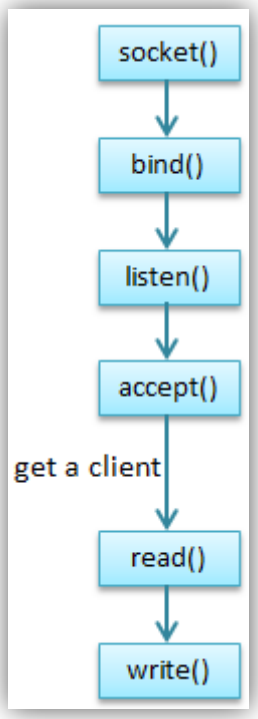


Εικόνα 40 – TCP/IP communication

Για την υποστήριξη των sockets, θα πρέπει να προστεθούν τα ακόλουθα στοιχεία στο πρόγραμμα:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/tcp.h>
```

Ας δούμε συνοπτικά τη διαδικασία που ακολουθήθηκε για την κατασκευή του Server που τρέχει στο BB παραθέτοντας κάποια τμήματα κώδικα από την υλοποίηση. Στο σχήμα που φαίνεται δίπλα, φαίνονται τα βασικά βήματα που ακλουθεί μία εφαρμογή server.



Εικόνα 41 –TCP Server basic functions

- Αρχικά πρέπει να δημιουργηθεί ένα socket. Για να γίνει αυτό θα πρέπει να κληθεί η συνάρτηση socket(), η οποία δημιουργεί ένα un-named socket στον kernel και επιστρέφει μία ακέραια τιμή που είναι γνωστή ως socket descriptor. Αν η επιστρεφόμενη τιμή είναι -1, τότε απέτυχε η δημιουργία του socket. Το πρώτο όρισμα καθορίζει την address family (π.χ. για internet family IPv4 διευθύνσεων, βάζω AF_INET), το δεύτερο καθορίζει τον τύπο του socket που θα δημιουργηθεί και το τρίτο το πρωτόκολλο που θα χρησιμοποιηθεί.

```
int sock = socket(localAddress->ai_family, localAddress->ai_socktype,
                  localAddress->ai_protocol)
```

- Τα sockets που δημιουργούνται με την συνάρτηση socket(), είναι αρχικά unnamed. Για το λόγο αυτό μετά εκτελείται η συνάρτηση bind(), η οποία αναθέτει μία τοπική socket address στο socket που δείχνει ο προηγούμενος socket descriptor.

```
bind(sock, localAddress->ai_addr, localAddress->ai_addrlen)
```

- Μετά την επιτυχή δημιουργία του socket, ο server εκτελώντας τη συνάρτηση listen(), δημιουργεί ένα πλήρως λειτουργικό listening socket. Σαν πρώτο όρισμα δέχεται το socket descriptor, ενώ σαν δεύτερο τον μέγιστο αριθμό συνδέσεων.

```
res = listen(sockt, 1);
```

- Μετά τη συνάρτηση listen(), ακολουθεί η accept(), η οποία βάζει τον server σε sleep mode. Περιμένει μέχρι να γίνει κάποιο request από κάποιον client και αφού ολοκληρωθεί το TCP handshake, επιστρέφεται ένας descriptor που αναπαριστά τον client που μόλις συνδέθηκε.

```
con = accept(sockt, (struct sockaddr *) &client_address,
             &client_address_size);
```

Από τη στιγμή που ολοκληρώνεται η παραπάνω διαδικασία και έχει επιτευχθεί η επικοινωνία μεταξύ BB και host υπολογιστή, ο server αναμένει για εντολές, τις οποίες θα μεταφέρει κατάλληλα στον ελεγκτή MD25. Οι εντολές που στέλνονται από τον host υπολογιστή και λαμβάνει το BB είναι οι εξής:

"set address register value"
"get address register"

Η πρώτη μπορεί να χρησιμοποιηθεί για να τεθούν τιμές στον ελεγκτή MD25, ενώ με τη δεύτερη ο host μπορεί να ζητήσει την τιμή κάποιου register από τον MD25 (π.χ. την κατάσταση της μπαταρίας). Στις παραπάνω εντολές, η παράμετρος address περιέχει τη διεύθυνση του MD25 πάνω στο I2C Bus,η δεύτερη παράμετρος αφορά τον register που θέλω να γραφτεί ή να διαβαστεί και η τελευταία είναι η τιμή που τίθεται στον register. Στο πρόγραμμα που σχεδιάστηκε, αρχικά υπάρχει μία συνάρτηση **read_line** η οποία διαβάζει κάθε φορά μία εντολή από το δίκτυο και την αποθηκεύει σε ένα string που ονομάζεται request. Στη συνέχεια ο κώδικας συγκρίνει τα τρία πρώτα γράμματα του string για να καταλάβει ποια από τις δύο εντολές είναι και αναλόγως το αποτέλεσμα καλεί την αντίστοιχη συνάρτηση:

```
read_line(&reader, request, sizeof(request));
if( strncmp("get", request, 3) == 0)
{
    get_command(request, i2c_file, response, sizeof(response));
}
else if( strncmp("set", request, 3) == 0)
{
    set_command(request, i2c_file, response, sizeof(response));
}
else
{
    printf("ERROR:Unknown command\n");
    strcpy(response, "ERROR:\r\n");
}
```

Στις δύο συναρτήσεις get_command και set_command που καλούνται παραπάνω, η εντολή ¨σπάει¨ στα συστατικά της μέρη με τη βοήθεια της sscanf όπως φαίνεται στη συνέχεια για την εντολή set και τα αποτελέσματα αποθηκεύονται σε τρεις μεταβλητές address, reg και value αντίστοιχα:

```
int n = sscanf(command, "set %hhd %hhd %hhd",&address, &reg, &value);
```

Έπειτα, αναλόγως με τον τύπο της εντολής γίνεται είτε read, είτε write στο I2C Bus. Και στις δύο περιπτώσεις θα πρέπει να τεθεί αρχικά η διεύθυνση του ελεγκτή ως slave address με τον εξής τρόπο:

```
if(ioctl(file, I2C_SLAVE, address) < 0)
{
    printf("ERROR: Failed setting slave address!!\n");
    return -1;
}
```

Τέλος, διακρίνω τις δύο περιπτώσεις:

- Αν γράφω στο bus εκτελώ:

```
__s32 r = i2c_smbus_write_byte_data(file, reg, value);
```

- Αν διαβάζω, εκτελώ:

```
__ s32 r = i2c_smbus_read_byte_data(file, reg);
```

5. Έλεγχος ρομπότ με χρήση του Kinect

Στο κεφάλαιο αυτό θα γίνει περιγραφή των βημάτων που ακολουθήθηκαν για την ανάπτυξη μιας εφαρμογής η οποία χρησιμοποιεί τον αισθητήρα Kinect. Η εφαρμογή αυτή βασιζόμενη στις κινήσεις του χρήστη μεταδίδει στη ρομποτική διάταξη τις ανάλογες εντολές κίνησης μέσω δικτύου. Επίσης αναλαμβάνει να παρουσιάσει το streaming βίντεο που στέλνει το ρομπότ. Για την ανάπτυξη της εφαρμογής υπήρχαν δύο επιλογές: α) Σε περιβάλλον Linux με χρήση open source οδηγών και β) στο περιβάλλον των Windows 7 χρησιμοποιώντας το official SDK της Microsoft. Έγιναν δοκιμές και στα δύο λειτουργικά συστήματα ωστόσο λόγω καλύτερης υποστήριξης, τελικά επιλέχθηκε το official SDK. Ωστόσο αυτό για εμένα σήμαινε ότι θα έπρεπε σε σύντομο χρόνο να μάθω μία νέα γλώσσα προγραμματισμού, τη **C#**. Ο προγραμματισμός τελικά έγινε με χρήση του Microsoft Visual C# 2010 Express.

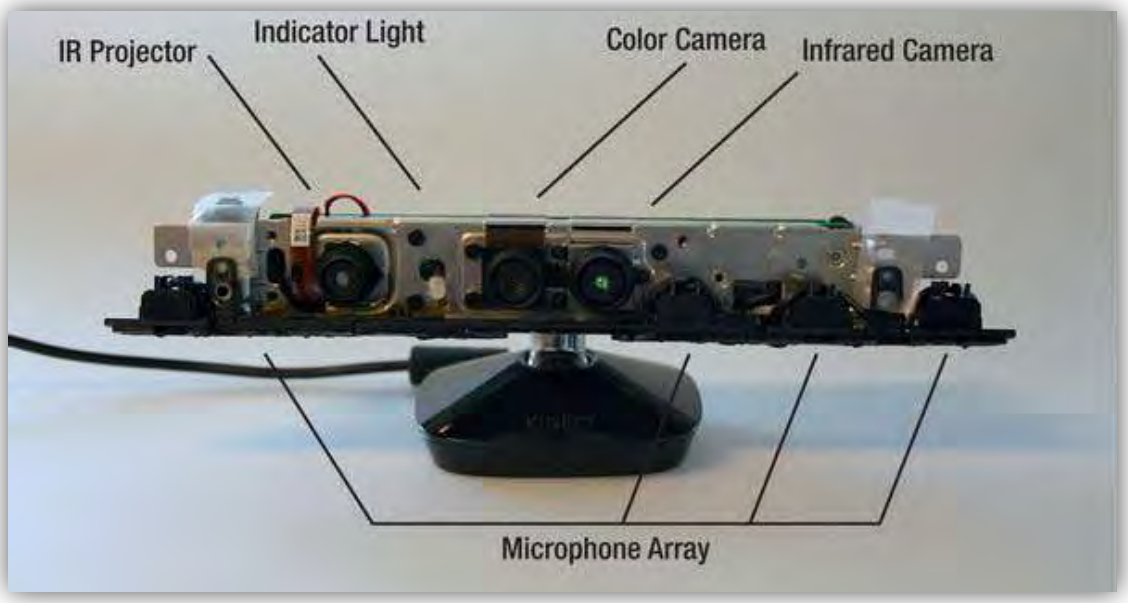
5.1 Εισαγωγή στον αισθητήρα Kinect

Πριν αρχίσει η παρουσίαση της διαδικασίας του προγραμματισμού, θα γίνει μία σύντομη παρουσίαση του αισθητήρα Kinect, για να γνωρίσουμε καλύτερα το hardware με το οποίο δούλεψα. Καταρχήν τι είναι το Kinect? Είναι μία motion sensing USB (Universal Serial Bus) συσκευή η οποία παρουσιάστηκε από την εταιρία Microsoft το Νοέμβριο του 2010 για να συνοδεύσει την παιχνιδομηχανή XBOX360. Η συσκευή είναι σε θέση να αναγνωρίζει φωνητικές εντολές, χειρονομίες, κινήσεις σώματος ακόμη και πρόσωπα. Οι οπτικές και ακουστικές πληροφορίες χρησιμοποιούνται ως εντολές αλληλεπίδρασης με το ψηφιακό περιεχόμενο των παιχνιδιών ή άλλων εφαρμογών. Αν και η άφιξη του Kinect στην αγορά δημιούργησε ανάμεικτα συναισθήματα, αυτό δεν το εμπόδισε ώστε να πουλάει 133 χιλιάδες συσκευές την ημέρα, τις πρώτες 60 ημέρες μετά την κυκλοφορία του, κάνοντας νέο ρεκόρ Guinness. Ο αρχικός προορισμός του Kinect ήταν αποκλειστικά για χρήση μαζί με τη συσκευή XBOX360 προφέροντας νέες δυνατότητες ψυχαγωγίας. Ωστόσο το Νοέμβριο του 2010 η εταιρία Adafruit Industries αποφάσισε να προσφέρει ένα χρηματικό ποσό σε όποιον κατάφερνε να δημιουργήσει έναν open source driver για το Kinect. Η Microsoft παρά τις αρχικές της αντιδράσεις, τελικά αναθεώρησε τη στάση της, μιας και ο open source driver δεν αποτελούσε προϊόν hacking αφού όπως η ίδια δήλωσε η σύνδεση USB είχε μείνει ανοιχτεί από το στάδιο σχεδιασμού της συσκευής. Στις 10 Νοεμβρίου, η Adafruit ανακήρυξε νικητή τον Héctor Martín, ο οποίος είχε παράγει έναν Linux driver που επέτρεπε την πλήρη χρήση της συσκευής. Μετά από αυτήν την επιτυχία υπήρξε ένα ολοένα και αυξανόμενο ενδιαφέρον από Developers μιας και η αλληλεπίδραση του Kinect με τους υπολογιστές αποτέλεσε τη βάση για

Εικόνα 42 – Ο αισθητήρας Kinect

τη δημιουργία πολλών πρωτότυπων και ριζοσπαστικών project. Το γεγονός αυτό οδήγησε τη Microsoft στο να ανακοινώσει το non-commercial Kinect Software Development Kit (SDK) τον Ιούλιο του 2011.

Το Kinect SDK για Windows είναι στην πραγματικότητα μία σειρά από βιβλιοθήκες, που μας επιτρέπουν να προγραμματίσουμε εφαρμογές χρησιμοποιώντας τον αισθητήρα Kinect ως είσοδο. Για να μπορέσει κανείς να προγραμματίσει χρησιμοποιώντας τη συσκευή, πρώτα θα πρέπει να γνωρίσει το Hardware του Kinect και πως αυτό δουλεύει. Στην επόμενη φωτογραφία, έχει αφαιρεθεί το κάλυμμα της συσκευής ώστε να είναι εμφανή τα μέρη που την απαρτίζουν.



Εικόνα 43 – Συστατικά μέρη του αισθητήρα

Όπως βλέπουμε η συσκευή αποτελείται βασικά από έναν infrared projector, μία infrared camera, μία color camera και από μία συστοιχία μικροφώνων. Οι συσκευές αυτές λειτουργώντας όλες μαζί προσδίδουν στο Kinect την ιδιαίτερη λειτουργικότητα του:

- **Color RGB camera:** Αυτή η κάμερα στοχεύει στην αναγνώριση διάφορων features, όπως για παράδειγμα την αναγνώριση προσώπων. Η RGB κάμερα προσφέρει βίντεο σε διαφορετικές αναλύσεις:
  - 12 FPS: 1280X960 RGB
  - 15 FPS: Raw YUV 640x480
  - 30 FPS: 640x480

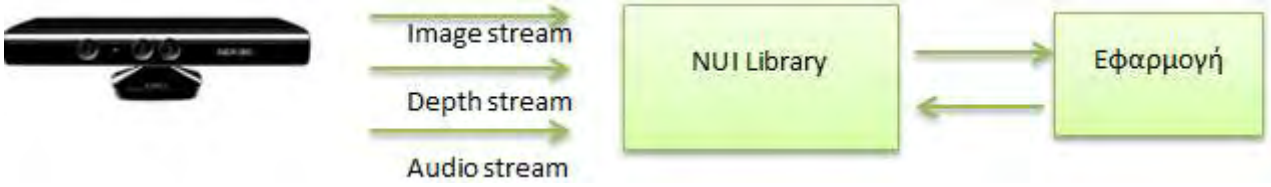
- **Depth Sensor:** Ο infrared projector σε συνδυασμό με την infrared camera (ένας monochrome CMOS sensor) συνιστούν τον Depth sensor της συσκευής.

  Ο αισθητήρας αυτός είναι ικανός να συλλαμβάνει 3D video data ανεξάρτητα από τις συνθήκες φωτισμού που επικρατούν. Είναι σε θέση να προσφέρει βίντεο ανάλυσης

640x480 pixels με 11-bit depth στα 30FPS, προσφέροντας έτσι 2.048 επίπεδα ευαισθησίας όσον αφορά το βάθος. Στη συνέχεια φαίνεται η δυνατότητα ανίχνευσης του Kinect, σε συνάρτηση με την απόσταση από τη συσκευή:

- **Microphone Array:** Στη βάση του αισθητήρα υπάρχουν τέσσερα μικρόφωνα όπως φαίνεται και στην προηγούμενη εικόνα. Συγκρίνοντας το πότε κάθε μικρόφωνο λαμβάνει τον ίδιο ήχο, η συσκευή είναι σε θέση να αναγνωρίσει την κατεύθυνση από την οποία έρχεται ο ήχος. Αυτή η τεχνική μπορεί επίσης να χρησιμοποιηθεί για να μπορεί το Kinect να εστιάζει περισσότερο σε κάποια πηγή ήχου. Επίσης αλγόριθμοι μπορούν να χρησιμοποιηθούν ώστε να αφαιρεθούν τυχόν θόρυβοι από το παρασκήνιο. Αυτή η εξελιγμένη αλληλεπίδραση μεταξύ hardware και software επιτρέπει τη χρήση φωνητικών εντολών ακόμη και σε μεγάλους χώρους, με τον χρήστη να βρίσκεται μερικά μέτρα μακριά.

Στη συνέχεια φαίνεται συνοπτικά η αλληλεπίδραση μεταξύ Kinect και εφαρμογής. Η εφαρμογή προσπελαύνει τον αισθητήρα ο οποίος είναι συνδεδεμένος στον υπολογιστή, και λαμβάνει πίσω μία επεξεργασμένη εικόνα βασισμένη στα image και data streams με τη βοήθεια των βιβλιοθηκών του SDK.



Εικόνα 44 – Αλληλεπίδραση υλικού - εφαρμογής

Πολύ σημαντικό για την εφαρμογή που αναπτύχθηκε είναι το Skeleton API, το οποίο προσφέρει πληροφορίες για τους χρήστες που μπορεί να στέκονται μπροστά από τον αισθητήρα Kinect, με λεπτομέρειες σε τρεις διαστάσεις. Τα δεδομένα που παρέχονται στην εφαρμογή είναι ένα σύνολο είκοσι σημείων που αναπαριστούν είκοσι διαφορετικά σημεία πάνω στο σώμα του χρήστη. Αναλυτικά τα σημεία αυτά φαίνονται στην παρακάτω εικόνα:

Εικόνα 45 – Skeleton joints

Πρώτο βήμα είναι η εγκατάσταση του SDK στον προσωπικό υπολογιστή. Αυτό μπορεί να γίνει μεταβαίνοντας στην επίσημη ιστοσελίδα http://www.microsoft.com/en-us/kinectforwindows/ και κατεβάζοντας το ανάλογο αρχείο. Οι ελάχιστες απαιτήσεις του συστήματος είναι:

- Windows 7, Windows Embedded Standard 7
- 32-bit (x86) or 64-bit (x64) processor
- Dual-core 2.66-GHz or faster processor
- Dedicated USB 2.0 bus
- 2 GB RAM
- A Microsoft Kinect for Windows sensor

Το Kinect SDK προσφέρει στον developer δυνατότητα προγραμματισμού σε τρεις γλώσσες (C#, C++ και Visual Basic). Η επιλογή μου τελικά ήταν να προγραμματίσω σε C#, επομένως χρειάστηκε να γίνει εγκατάσταση της εφαρμογής Microsoft Visual C# 2010 Express , η οποία και αυτή μπορεί να βρεθεί από το επίσημο site της Microsoft. Αυτή τη στιγμή η Microsoft έχει κυκλοφορήσει τη νέα έκδοση 1.5 του SDK, ωστόσο όταν ξεκίνησε η εργασία ήταν διαθέσιμη η έκδοση 1.0 και αυτή χρησιμοποιήθηκε

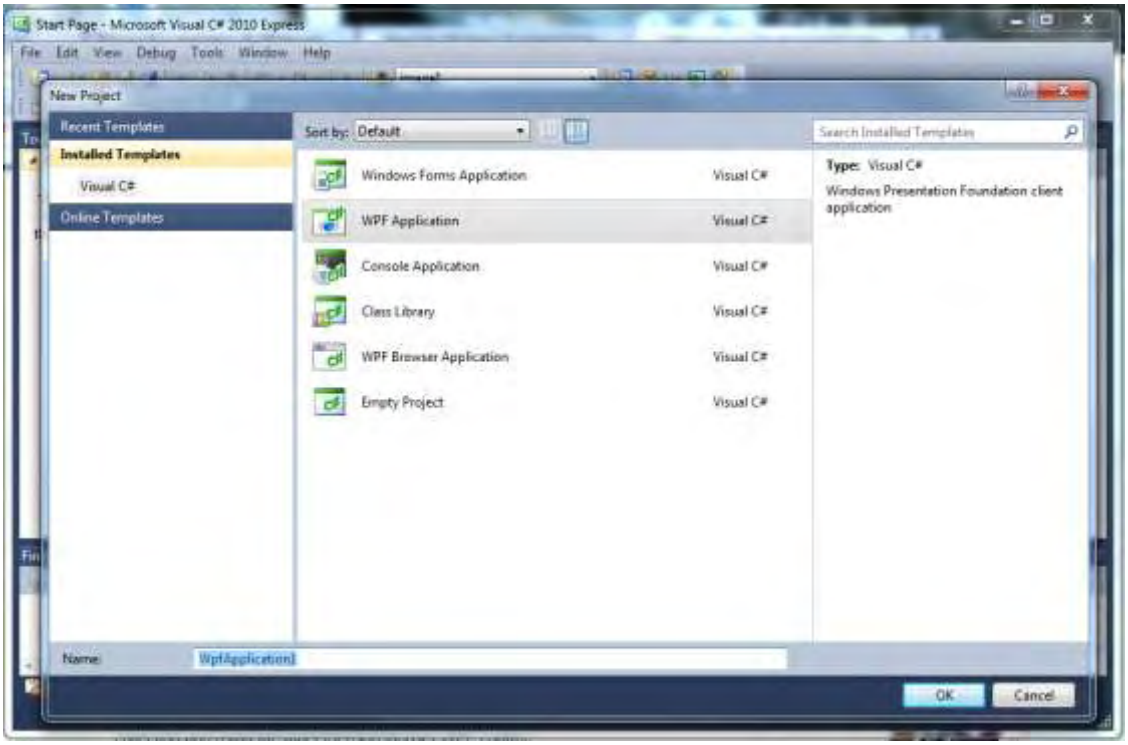
5.2 Ξεκινώντας τον προγραμματισμό με χρήση του Kinect

Το μοντέλο προγραμματισμού του Kinect διαφέρει από τα γνωστά, με την έννοια ότι πρέπει να συνηθίσει κανείς το γεγονός ότι το κυρίως UI (User Interface) των .NET προγραμμάτων δεν χρησιμοποιείται ως είσοδος όπως στις περισσότερες εφαρμογές. Αντίθετα τα παράθυρα συνήθως αναπαριστούν πληροφορίες, ενώ η είσοδος προέρχεται από τον αισθητήρα Kinect. Η είσοδος μιας εφαρμογής που χρησιμοποιεί το Kinect είναι μία συνεχής ροή πληροφοριών η οποία αλλάζει διαρκώς. Μία τέτοια εφαρμογή δεν περιμένει να συμβεί κάποιο συγκεκριμένο event, όπως για παράδειγμα το πάτημα ενός κουμπιού. Αντίθετα επεξεργάζεται συνεχώς τα δεδομένα που έρχονται την RGB κάμερα, τον αισθητήρα βάθους και αναλόγως μεταβάλει το UI. Το Kinect SDK υποστηρίζει τριών ειδών εφαρμογές: Console application, WPF application και Windows Forms applications.

Για τις ανάγκες της εργασίας, έγινε ανάπτυξη μίας WPF εφαρμογής. Το WPF ή αλλιώς Windows Presentation Foundation, εμφανίστηκε με τα Windows Vista ως μέρος του .NET framework 3.0. Το εργαλείο αυτό επιτρέπει τη δημιουργία διεπαφών χρήστη (UI) που περιέχουν αρχεία, πολυμέσα, 2D και 3D γραφικά, animations, web-like χαρακτηριστικά και πολλά άλλα. Το WPF δίνει τη δυνατότητα στους σχεδιαστές (designers) να δουλεύουν μαζί με τους developers. Για να μπορεί αυτό να είναι δυνατό, το WPF εισάγει μία νέα γλώσσα, την eXtensible Application Markup Language (XAML). Η γλώσσα αυτή ορίζει ένα σύνολο από XAML στοιχεία, όπως *Button*, *Textblock*, *Label* και άλλα, που καθορίζουν την εμφάνιση ενός user interface. Κάθε XAML στοιχείο αντιστοιχεί σε μία WPF κλάση και κάθε γνώρισμα (attribute) του στοιχείου, αντιστοιχεί σε κάποια ιδιότητα ή

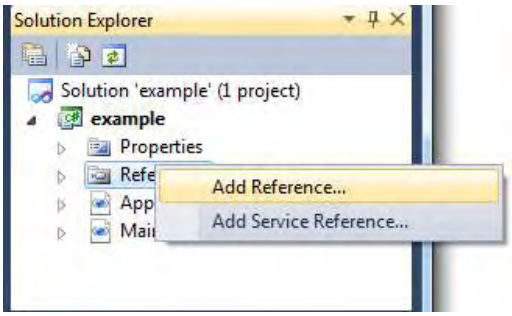
κάποιο event στην κλάση αυτή. Στην περίπτωση της εργασίας, η οι κλάσεις ορίζονται σε αρχεία με κατάληξη .cs, δηλαδή αρχεία με κώδικα γραμμένο σε γλώσσα C# .

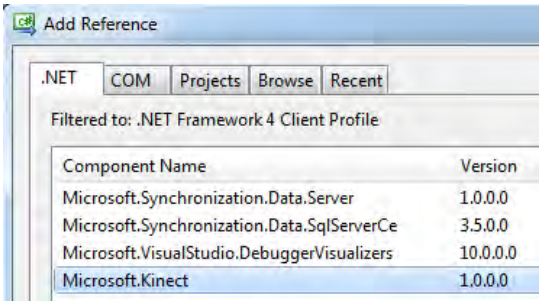
Τα πρώτα βήματα στην διαδικασία ανάπτυξης της εφαρμογής, αφορούν τη σωστή προετοιμασία του προγραμματιστικού περιβάλλοντος. Ανοίγοντας το Visual C# Express 2010, στην αρχική οθόνη επιλέγουμε New Project και δημιουργούμε μία WPF εφαρμογή δίνοντας ένα όνομα στο κατάλληλο πεδίο.



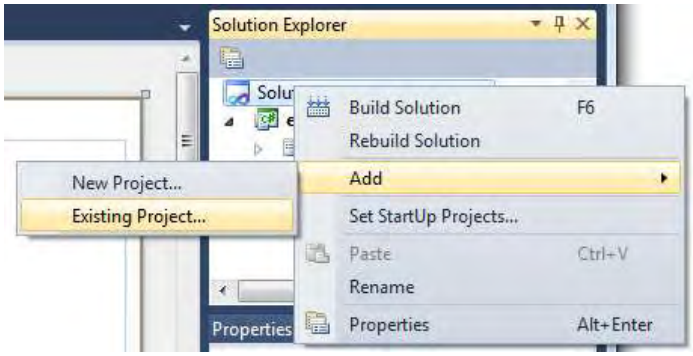
Εικόνα 46 – Δημιουργία νέας WPF εφαρμογής

Αυτό που πρέπει να γίνει αρχικά, είναι να προστεθεί μία αναφορά στο Microsoft.Kinect το οποίο είναι η βιβλιοθήκη για τις API κλήσεις. Αυτό γίνεται κάνοντας δεξί κλικ στην επιλογή References στην καρτέλα Solution Explorer και επιλέγοντας Add Reference. Στο παράθυρο που ανοίγει, στην καρτέλα .NET επιλέγεται η αναφορά Microsoft.Kinect.





Στη συνέχεια πρέπει να προστεθεί στο project , ένα ήδη υπάρχον project από το επίσημο SDK της Microsoft, το Microsoft.Samples.Kinect.WpfViewers. Αυτό γίνεται όπως φαίνεται παρακάτω:



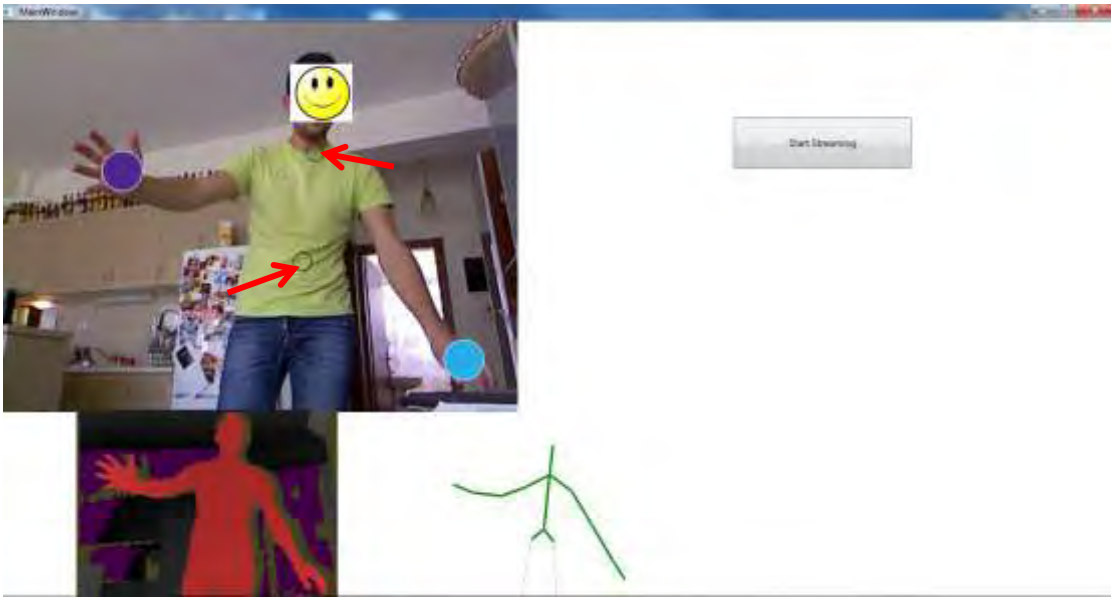
Εικόνα 47 – Προσθήκη του Microsoft.Samples.Kinect.WpfViewers

Τέλος πρέπει να προστεθεί μία αναφορά στη βιβλιοθήκη Coding4Fun. Αυτό γίνεται με τον τρόπο που περιγράφηκε προηγουμένως, όμως επιλέγεται η καρτέλα Browse και επιλέγεται η βιβλιοθήκη Coding4Fun.Kinect.Wpf.dll στη θέση που είναι αποθηκευμένη.

Όπως έχει περιγραφεί και σε προηγούμενες ενότητες, η εφαρμογή θα πρέπει να εκτελεί τρεις βασικές λειτουργίες:

- Αναγνώριση των κινήσεων του χρήστη
- Κατάλληλη μετάφρασή τους σε εντολές κίνησης
- Παρουσίαση του βίντεο που στέλνει το ρομπότ

Ξεκινώντας θα γίνει μία παρουσίαση του UI της εφαρμογής. Όπως ήδη αναφέρθηκε αυτό γίνεται στη γλώσσα XAML. Στην επόμενη εικόνα φαίνεται ένα snapshot της εφαρμογής κατά τη διάρκεια της εκτέλεσης.



Εικόνα 48 – Application UI

Το παραθυρικό περιβάλλον, περιλαμβάνει μία ροή βίντεο από την RGB κάμερα του αισθητήρα, ένα βίντεο ανάλυσης 320x240 pixels από τον depth sensor καθώς και ένα skeleton viewer. Οι τρεις προηγούμενοι viewers που προσπελαύνουν το hardware του Kinect, περιλαμβάνονται στο project

Microsoft.Samples.Kinect.WpfViewers και γι αυτό το λόγο έγινε η εισαγωγή του στην αρχή. Στην RGB εικόνα υπάρχει αντιστοίχηση των πέντε skeleton joints που χρησιμοποιούνται στον κώδικα, στους αντίστοιχους κύκλους και στην εικόνα με το χαμογελαστό πρόσωπο. Επίσης αν και στο παραπάνω στιγμιότυπο δεν φαίνεται, έχει εισαχθεί και ένας controller για το Kinect (kinectSensorChooser1) ο οποίος επιτρέπει τον έλεγχο της κατάστασης του Kinect χωρίς να καταρρέει η εφαρμογή, όταν για παράδειγμα αποσυνδεθεί το Kinect ενώ η εφαρμογή τρέχει. Ακόμη, υπάρχει ένα κουμπί Start Streaming, το οποίο όταν πατηθεί ανοίγει ένα νέο παράθυρο στο οποίο προβάλλεται το video που στέλνει το BB. Παρακάτω φαίνεται ο κώδικας XAML που ορίζει το παράθυρο της εφαρμογής, το οποίο ονομάζεται MainWindow:

```
<Window x:Class="SkeletalTracking.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="768" Width="1280" Loaded="Window_Loaded"
        xmlns:my="clr-namespace:Microsoft.Samples.Kinect.WpfViewers;assembly=Microsoft.Samples.Kinect.WpfViewers"
        Closing="Window_Closing" WindowState="Maximized">
    <Canvas Name="MainCanvas">
        <my:KinectColorViewer Canvas.Left="-8" Canvas.Top="0" Width="640" Height="480"  Name="kinectColorViewer1"

                              Kinect="{Binding ElementName=kinectSensorChooser1, Path=Kinect}" />
        <Ellipse Canvas.Left="0" Canvas.Top="0" Height="50" Name="leftEllipse"  Width="50" Fill="#FF4D298D" Opacity="1"
Stroke="White" />
        <Ellipse Canvas.Left="100" Canvas.Top="0" Fill="#FF2CACE3" Height="50" Name="rightEllipse" Width="50" Opacity="1"
Stroke="White" />
        <my:KinectSensorChooser Canvas.Left="472" Canvas.Top="255" Name="kinectSensorChooser1" Width="328" />
        <my:KinectSkeletonViewer Canvas.Left="480" Canvas.Top="477" Name="kinectSkeletonViewer1" Width="320" Height="240"
Kinect="{Binding ElementName=kinectSensorChooser1, Path=Kinect}" />
        <my:KinectDepthViewer Canvas.Left="91" Canvas.Top="477" Name="kinectDepthViewer1" Width="320" Height="240"
Kinect="{Binding ElementName=kinectSensorChooser1, Path=Kinect}" />
        <Image Canvas.Left="168" Canvas.Top="134" Height="71" Name="image1" Stretch="Fill" Width="76"
Source="/SkeletalTracking;component/Images/face21.png" />
        <Button Canvas.Left="896" Canvas.Top="118" Content="Start Streaming" Height="63" Name="button1" Width="219"
Click="startStreaming"/>
        <Label Foreground="White" Content="{Binding HypothesizedText}" Height="55" Width="965" FontSize="32"
Canvas.Left="115" Canvas.Top="1025" />
        <Ellipse Canvas.Left="115" Canvas.Top="81" Height="24" Name="hip_center" Stroke="Black" Width="23" />
        <Ellipse Canvas.Left="12" Canvas.Top="81" Height="23" Name="shoulder_center" Stroke="Black" Width="23" />
        <TextBlock Canvas.Left="910" Canvas.Top="577" Height="72" Name="textBlock1" Width="223" TextWrapping="Wrap"/>
    </Canvas>
</Window>
```

Στον ακόλουθο πίνακα φαίνονται όλες οι πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί ο αισθητήρας. Ο controller kinectSensorChooser1 που χρησιμοποιώ είναι σε θέση να αντιμετωπίσει οποιαδήποτε από τις παρακάτω καταστάσεις κατάλληλα.

KinectStatus	What it means
Undefined	The status of the attached device cannot be determined
Connected	The device is attached and is capable of producing data from its streams.
DeviceNotGenuine	DeviceNotGenuine The attached device is not an authentic Kinect sensor.
Disconnected	Disconnected The USB connection with the device has been broken.

Error	Error Communication with the device produces errors.
Initializing	The device is attached to the computer, and is going through the process of connecting.
InsufficientBandwidth	Kinect cannot initialize, because the USB connector does not have the necessary bandwidth required to operate the device.
NotPowered	Kinect is not fully powered. The power provided by a USB connection is not sufficient to power the Kinect hardware. An additional power adapter is required.
NotReady	Kinect is attached, but is yet to enter the Connected state.

Στο αρχείο MainWindow.xaml.cs που βρίσκεται ο κώδικας C#, θα πρέπει αρχικά να γίνει αναφορά στα: Microsoft.Kinect, Coding4Fun.Kinect.Wpf και Microsoft.Samples.Kinect.WpfViewers, δηλώνοντας τα με τον εξής τρόπο:

```
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Microsoft.Samples.Kinect.WpfViewers;
```

Η εφαρμογή ξεκινάει με τον event handler Window_Loaded (εικόνα) που περιέχει λεπτομέρειες σχετικά με το τι πρέπει να γίνει όταν ανοίξει το παράθυρο της εφαρμογής. Όπως φαίνεται και στον κώδικα που παρατίθεται, υπάρχουν δύο βασικές εργασίες που επιτελούνται. Καταρχήν ο αισθητήρας Kinect αρχικοποιείται με τη βοήθεια του controller kinectSensorChooser1 και της συνάρτησης kinectSensorChooser1_KinectSensorChanged και κατά δεύτερον στήνεται το δίκτυο για τη μεταφορά της ροής video από τη ρομποτική διάταξη.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    kinectSensorChooser1.KinectSensorChanged +=
        new DependencyPropertyChangedEventHandler
            (kinectSensorChooser1_KinectSensorChanged);
    SetupNetwork();

}
```

Στη συνάρτηση kinectSensorChooser1_KinectSensorChanged ανακαλύπτεται ο αισθητήρας Kinect που είναι συνδεδεμένος στον υπολογιστή και ρυθμίζονται κατάλληλα οι παράμετροι του, προτού είναι έτοιμος να ξεκινήσει τη λειτουργία του. Η μεταβλητή `KinectSensor` χρησιμοποιείται για την αποθήκευση των πληροφοριών του εκάστοτε αισθητήρα. Ένας αισθητήρας για να αρχικοποιηθεί θα πρέπει να βρίσκεται στην κατάσταση *connected*. Στη συνέχεια ενεργοποιούνται οι ροές δεδομένων: DepthStream, ColorStream και SkletonStream που επιτρέπουν την προσπέλαση της RGB κάμερας, του Depth sensor και την κατασκευή του σκελετού του χρήστη. Επίσης προστίθεται το event AllFramesReady το οποίο πυροδοτείται όταν όλων των ειδών τα frames είναι έτοιμα. Πρέπει σε αυτό το σημείο να δοθεί προσοχή, ώστε οι αναλύσεις των δύο καμερών όπως ορίζονται, να είναι ίδιες. Επίσης μπορούν να οριστούν και κάποιες παράμετροι smoothing, που εισάγουν

ωστόσο latency στις κινήσεις κάτι που δεν πρέπει να υφίσταται στην εφαρμογή, μιας και πρέπει να υπάρχει όσο το δυνατόν πιο άμεση ανταπόκριση στις κινήσεις του χρήστη. Τέλος με τη μέθοδο sensor.Start() δίνεται εντολή στο Kinect να ξεκινήσει.

```
void kinectSensorChooser1_KinectSensorChanged(object sender, DependencyPropertyChangedEventArgs e)
{
        KinectSensor old = (KinectSensor)e.OldValue;

        StopKinect(old);

        KinectSensor sensor = (KinectSensor)e.NewValue;

        if (sensor == null)
        {
                return;
        }

        sensor.SkeletonStream.Enable();

        sensor.AllFramesReady += new EventHandler<AllFramesReadyEventArgs>(sensor_AllFramesReady);
        sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
        sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

        try
        {
                sensor.Start();
        }
        catch (System.IO.IOException)
        {
                kinectSensorChooser1.AppConflictOccurred();
        }

}
```

Όσον αφορά την επικοινωνία πάνω στο δίκτυο, χρησιμοποιήθηκαν TCP sockets, με την εφαρμογή μου να λαμβάνει το ρόλο του client. Για την υλοποίηση της σύνδεσης είναι απαραίτητη γνώση της IP διεύθυνσης και τη θύρας (port) του server, δηλαδή του BB. Θα πρέπει επίσης να δηλωθούν να namespace's:

```
using System.Net;
using System.Net.Sockets;
```

Ο κώδικας φαίνεται παρακάτω:

```
void SetupNetwork()
{

    IPAddress host = IPAddress.Parse("192.168.1.1");
    IPEndPoint hostep = new IPEndPoint(host, port);
    sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

    try
    {
        sock.Connect(hostep);
    }
    catch (SocketException e)
    {
        Console.WriteLine("Problem connecting to host!");
        Console.WriteLine(e.ToString());
        sock.Close();
        return;
    }

    try
    {
        sock.Send(Encoding.ASCII.GetBytes("set 89 15 0 \n"));
    }
    catch (SocketException e)
    {
        Console.WriteLine("Problem setting mode!");
        Console.WriteLine(e.ToString());
        sock.Close();
        return;
    }
}
```

Πριν γίνει η επεξεργασία των δεδομένων που έρχονται από τον αισθητήρα, έπρεπε να οριστεί κατάλληλα ο handler στην περίπτωση κλεισίματος του παραθύρου (της εφαρμογής δηλαδή). Στην περίπτωση αυτή πρέπει να απενεργοποιηθεί ο αισθητήρας και να κλείσει η επικοινωνία μεταξύ host υπολογιστή και BB ως εξής:

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    closing = true;
    StopKinect(kinectSensorChooser1.Kinect);
    sock.Close();
}
private void StopKinect(KinectSensor sensor)
{
    if (sensor != null)
    {
        if (sensor.IsRunning)
        {
            //stop sensor
            sensor.Stop();

            //stop audio
            if (sensor.AudioSource != null)
            {
                sensor.AudioSource.Stop();
            }
        }
    }
}
```

Ο event handler sensor_AllFramesReady προσφέρει τη βασική λειτουργικότητα της εφαρμογής, περιμένοντας το event που τον ειδοποιεί ότι όλα τα frames είναι έτοιμα. Όταν αυτό το event πυροδοτείται τελικά, σημαίνει ότι το Kinect έχει λάβει ένα Depth data frame, ένα Color data frame και ένα skeleton frame ταυτόχρονα. Στην περίπτωση αυτή ο driver του Kinect αποθηκεύει τα δεδομένα σε πίνακες, έναν για κάθε τύπο frame. Στον handler sensor_AllFramesReady υλοποιούνται οι εξής βασικές λειτουργίες:

- Ανίχνευση του πρώτου σκελετού που βρίσκει ο αισθητήρας
- Αντιστοίχηση (map) των joints από το skeleton frame στο depth frame
- Αντιστοίχηση των joints από το depth frame στο color frame
- Έλεγχος της θέσης των joints και αποστολή της ανάλογης κίνησης στο ρομπότ

Η διαδικασία του mapping που περιγράφεται προηγουμένως μπορεί να αναπαρασταθεί απλοϊκά ως εξής:

Εικόνα 49 – Αντιστοίχηση των joints στα διάφορα frames

Κάθε joint έχει συντεταγμένες (x,y,z). Στο πρώτο βήμα γίνεται αντιστοίχηση αυτών των συντεταγμένων στο depth frame. Ο λόγος που δεν γίνεται απλώς αντιγραφή των συντεταγμένων από το depth frame στο color frame, είναι ότι δεν είναι ακριβώς ίδιες. Αυτό συμβαίνει επειδή πάνω στο Kinect οι δύο κάμερες έχουν μία μικρή απόσταση μεταξύ τους επομένως δεν κάνουν λήψη της ίδιας ακριβώς εικόνας. Για το λόγο αυτό πρέπει να γίνει μία αντιστοίχηση των σημείων. Στη συνέχεια φαίνεται ο τρόπος που μπορούν να γίνουν αυτές οι δύο διαδικασίες, έστω για το κεφάλι.

```
//Map a joint location to a point on the depth map
//head
DepthImagePoint headDepthPoint =
     depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);

//Map a depth point to a point on the color image
//head
ColorImagePoint headColorPoint =
     depth.MapToColorImagePoint(headDepthPoint.X, headDepthPoint.Y,
     ColorImageFormat.RgbResolution640x480Fps30);
```

Μετά την παραπάνω διαδικασία, εξετάζεται η θέση των πέντε σημείων του σκελετού(head, shoulder_center, hip_center, right hand, left hand) και αναλόγως αποφασίζεται η κίνηση που πρέπει να εκτελέσει το ρομπότ. Στις παρακάτω εικόνες φαίνονται στιγμιότυπα από την εκτέλεση της εφαρμογής και περιγράφεται η κίνηση του ρομπότ που επιτελείται:

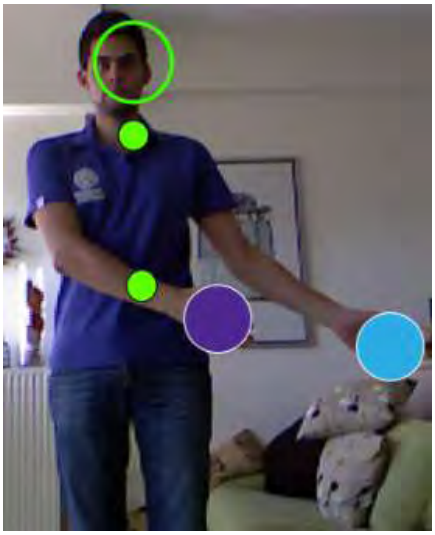
Εικόνα 51 – Κίνηση εμπρός

Εικόνα 50 – Κίνηση πίσω

Εικόνα 52 - Initialize

Εικόνα 53 – Τερματισμός εφαρμογής

Εικόνα 54 – Στροφή αριστερόστροφα



Εικόνα 55 – Στροφή δεξιόστροφα

Τελευταίο βήμα ήταν η κατάλληλη διαχείριση του βίντεο που στέλνει το BB. Για τη δημιουργία μιας πιο ολοκληρωμένης εφαρμογής, έγινε προσπάθεια ενσωμάτωσης του βίντεο στην WPF

εφαρμογή, έτσι ώστε να μην χρειάζεται να υπάρχει ανοιχτός παράλληλα και κάποιος web browser. Για το λόγο αυτό χρησιμοποιήθηκε η εφαρμογή MJPEG Decoder (http://mjpeg.codeplex.com/), η οποία στην ουσία κάνει πολλαπλά http requests σε μία συγκεκριμένη URL και λαμβάνει ένα stream από JPEG εικόνες τις οποίες παρουσιάζει στην WPF εφαρμογή. Για να χρησιμοποιηθεί η εφαρμογή αυτή πρέπει να γίνουν τα εξής:

1. Αναφορά στη βιβλιοθήκη **MjpegProcessor.dll**
2. Δημιουργία ενός νέου αντικειμένου **MjpegDecoder**
3. Εισαγωγή ενός event **FrameReady**
4. Στον event handler, πρέπει να γίνει λήψη της εικόνας **Bitmap** και να ανατεθεί στον image display control
5. Κλήση της μεθόδου **ParseStream** με τον τρόπο που φαίνεται στη συνέχεια, χρησιμοποιώντας την URL που γίνεται το streaming.

Στο παράθυρο της εφαρμογής έχει εισαχθεί ένα κουμπί “Start Streaming”, το οποίο όταν πατηθεί πυροδοτεί ένα event που έχει ως αποτέλεσμα το άνοιγμα ενός νέου παραθύρου, στο οποίο προβάλλεται το βίντεο από το BB. Το κλείσιμο αυτού του παραθύρου δε σημαίνει και τον τερματισμό της εφαρμογής. Τμήμα του κώδικα που χρησιμοποιήθηκε φαίνεται παρακάτω:

```
public partial class Window1 : Window
    {
        MjpegDecoder _mjpeg;

              public Window1()
              {
                     InitializeComponent();
                     _mjpeg = new MjpegDecoder();
                     _mjpeg.FrameReady += mjpeg_FrameReady;
              }

              private void Start_Stream(object sender, RoutedEventArgs e)
              {
                     _mjpeg.ParseStream(new Uri("http://192.168.1.1:8080/?action=streamworks"));
              }

              private void mjpeg_FrameReady(object sender, FrameReadyEventArgs e)
              {
                     image1.Source = e.BitmapImage;
              }
}
```

6. Επίλογος – Μελλοντικές επεκτάσεις

Στη συγκεκριμένη εργασία παρουσιάστηκε μία νέα μέθοδος αλληλεπίδρασης μεταξύ ανθρώπου και ρομπότ. Πιο συγκεκριμένα υλοποιήθηκε ο απομακρυσμένος έλεγχος μιας ρομποτικής διάταξης βασιζόμενος στις κινήσεις του χρήστη. Για το σκοπό αυτό αναπτύχθηκε μία εφαρμογή που κάνει χρήση του αισθητήρα Kinect, με στόχο να αναγνωρίζει συγκεκριμένες χειρονομίες του χρήστη και τις ανάγει σε αντίστοιχες εντολές κίνησης. Οι εντολές αυτές μεταδίδονται ασύρματα, μέσω Wi-fi σε μία ρομποτική διάταξη που κατασκευάστηκε στα πλαίσια της εργασίας. Έγινε αναλυτική περιγραφή όλων των σταδίων της κατασκευής καθώς και του προγραμματισμού των εφαρμογών που υλοποιήθηκαν για την υποστήριξη των διαφόρων λειτουργιών.

Πιθανές επεκτάσεις στην εργασία θα μπορούσαν να είναι οι εξής:

- Η προσθήκη ultra sonic αισθητήρων περιφερειακά της ρομποτικής διάταξης, με σκοπό την αξιόπιστη αναγνώριση και αποφυγή εμποδίων που ο χρήστης μπορεί να αγνοήσει στην παρούσα υλοποίηση.
- Προσθήκη κάποιου σερβομηχανισμού στη βάση της κάμερας του ρομπότ, που θα επιτρέπει την περιστροφή της.
- Προσθήκη τρισδιάστατης όρασης (3D-Vision) στο ρομπότ και ακόμη τη δυνατότητα mapping του χώρου που κινείται το ρομπότ σε τρεις διαστάσεις.

7. Βιβλιογραφία – Πηγές

[1] Beagleboard Rev C4 Official documentation: http://beagleboard.org/static/BBSRM_latest.pdf

[2] Beagleboard Wiki: http://elinux.org/BeagleBoard

[3] OMAP35x Applications Processor Technical Reference Manual:
http://www.ti.com/lit/ug/spruf98x/spruf98x.pdf

[4] MJPG-streamer: http://sourceforge.net/projects/mjpg-streamer/

[5] Yet Another Hacker's Blog: Beaglebot – A BeagleBoard based robot:
http://yetanotherhackersblog.wordpress.com/

[6] Surachai Panich, *Development of Mobile Robot Based on I2C Bus System*

[7] MD-25 H-Bridge Motor Drive documentation:
http://www.robotelectronics.co.uk/htm/md25tech.htm

[8] Gitorious: http://gitorious.org/

[9] LM317 Regulator datasheet: https://www.national.com/ds/LM/LM117.pdf

[10] LM7805 Regulator datasheet: http://www.sparkfun.com/datasheets/Components/LM7805.pdf

[11] Gregory Dudek and Michael Jenkin, *Computational Principles of Mobile Robotics*

[12] Sparkfun Electronics: http://www.sparkfun.com/

[13] Jarret Webb and James Ashley (2012), *Beginning Kinect Programming with the Microsoft Kinect SDK*

[14] Kinect Documentation & API Resources:
http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx

[15] MJPEG Decoder: http://channel9.msdn.com/coding4fun/articles/MJPEG-Decoder

[16] Gerard Marull Paretas, Introduction to Beagleboard, Universitat Politècnica de Catalunya, September 2009: http://www.teslabs.com/wp-content/uploads/2010/02/beagleboard.pdf