ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

# Ενσωματωμένα Συστήματα

**Ενότητα:** ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ Beagleboard SBC (single board computer)- Blink a LED & Read a Key

**Απαιτούμενος Αρ. Εργατοωρών:** 2

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

**Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών**

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

# Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

# Περιεχόμενα



Σημείωση: Σε περίπτωση που συναντήσετε πρόβλημα στη δημιουργία του device tree blob από το device source tree, μπορείτε να κατεβάσετε την έτοιμη έκδοση (ΜΟΝΟ σε περίπτωση προβλήματος): http://arch.icte.uowm.gr/quickshare/files/1508493064b6c03a14afa064374edb9d0547f0edc9/omap3-beagle-xm.zip

# 1. Σκοπός της άσκησης

Σε αυτό το εργαστήριο θα χρησιμοποιήσουμε τις γενικές διεπαφές εισόδου εξόδου (General Purpose Input Output) της αναπτυξιακής πλακέτας Beagleboard-XM για να ανάψουμε ένα LED και να διαβάσουμε την είσοδο από ένα κουμπί.

Ο οδηγός αυτός περιέχει αρκετές τεχνικές λεπτομέρειες για λόγους πληρότητας και εμβάθυνσης σε θέματα αρχιτεκτονικής του επεξεργαστή, για την καλύτερη κατανόηση του τι συμβαίνει και γιατί κάνουμε κάποιες ενέργειες.

Υποθέτουμε ότι σε αυτό το σημείο έχετε ένα Beagleboard-xm που εκτελεί ένα σύγχρονο πυρήνα Ubuntu ή Debian, ρυθμισμένο να συνδέεται στο διαδίκτυο, πλήρως ενημερωμένο, και ένα σειριακό καλώδιο για τη σύνδεση του Host υπολογιστή σας, στην αναπτυξιακή πλακέτα.

Να σημειωθεί ότι ενώ σε άλλες πλακέτες είτε έχουν λειτουργικό σύστημα (π.χ. raspberry) ή δεν έχουν (arduino) η διαδικασία ενεργοποίησης ενός LED είναι σχετικά απλή υπόθεση, στο Beagleboard-XM, απαιτεί αρκετές ρυθμίσεις (που γίνονται μια φορά) εξαιτίας του αρκετά σύνθετου επεξεργαστή TI – OMAP3. Στο διαδίκτυο υπάρχουν κάποιες πληροφορίες, αλλά δεν ισχύουν για τους σημερινούς πυρήνες, επειδή ο πυρήνας Linux του λειτουργικού συστήματος έχει υποστεί αρκετές αλλαγές και συγκεκριμένα έχει απομακρυνθεί η δυνατότητα χρήσης της ειδικής συσκευής `/sys/kernel/debug/omap_mux/` για να ρυθμιστεί η παράμετρο πολύπλεξης MUX (mode=04 αν θέλουμε GPIO). Αυτό επιτυγχάνεται σήμερα με τη χρήση ενός ειδικού αρχείου που ονομάζεται «Δένδρο συσκευών» ή Device Tree (DT). Σε μορφή κειμένου κάποιος περιγράφει το δένδρο συσκευών (πηγαίο αρχείο device tree source) και το μεταγλωττίζει με το `dtc` σε δυαδικό (device tree binary) το οποίο είναι ανεξαρτήτου αρχιτεκτονικής και μπορεί να δημιουργηθεί σε x86 και να αντιγραφεί σε ARM (π.χ. στο beagleboard).

# 2. Προαπαιτούμενα

Όπως αναφέρθηκε, αν και στο διαδίκτυο υπάρχουν αρκετές πληροφορίες για άλλες αρχιτεκτονικές, για την αναπτυξιακή πλακέτα beagleboard-xm δεν υπάρχει σύγχρονη βιβλιογραφία και οι μόνες πληροφορίες που αναφέρονται είναι παρωχημένες από το 2010-2013 και δεν ισχύουν. Ο διδάσκοντας για να καλύψει αυτό το κενό μελέτησε πλήθος εγγράφων που θα αναφερθούν στη συνέχεια, και τοποθετούνται ως αναφορές για κάποιον που θα θελήσει να εμβαθύνει σε αρκετά σημαντικό βαθμό σε αυτή την αρχιτεκτονική.

- Το πιο σημαντικό έγγραφο είναι η πλήρης και αναλυτική αρχιτεκτονική του επεξεργαστή που φέρει το beagleboard-xm. Το έγγραφο αυτό είναι το «OMAP35x Applications Processor Technical Reference Manual (TRM)» http://www.ti.com/lit/ug/spruf98y/spruf98y.pdf Μέσα σε 3492 σελίδες

περιγράφεται πλήρως η αρχιτεκτονική του επεξεργαστή. Κάποια σημεία κλειδιά θα τα αναφέρουμε στη συνέχεια.

- Σημαντικό επίσης έγγραφο είναι το «BeagleBoard-xM Rev C System Reference Manual» http://beagleboard.org/static/BBxMSRM_latest.pdf Μέσα σε 164 σελίδες περιγράφεται το board και οι δυνατότητες του.
- Σχετικά με τον επεξεργαστή και ιδιαίτερα τα ηλεκτρονικά χαρακτηριστικά υπάρχουν πληροφορίες στο DataSheet “Sitara™ AM335x ARM® Cortex™-A8 Microprocessors (MPUs)» http://www.ti.com/lit/ds/sprs717j/sprs717j.pdf
- Το κείμενο για τα GPIO του Linux στη διεύθυνση https://www.kernel.org/doc/Documentation/gpio/gpio.txt
- Πληροφορίες για το muxing του board στη σελίδα (αν και αναφέρεται σε παλαιότερη έκδοση Linux και δεν ισχύουν όλα): https://elinux.org/BeagleBoardPinMux
- Πληροφορίες για cross compile του Linux για ARM σε ένα υπολογιστή αρχιτεκτονικής x86 στη σελίδα: https://raspberrypi.stackexchange.com/questions/192/how-do-i-cross-compile-the-kernel-on-a-ubuntu-host
- Η λίστα μηνυμάτων beagleboard στα Google groups https://groups.google.com/forum/#!categories/beagleboard

Ο επεξεργαστής TI OMAP που φέρει το BBxM έχει πολυπλεγμένες εξόδους, επειδή οι ακροδέκτες του είναι πολύ λιγότεροι από τους ακροδέκτες όλων των υποσυστημάτων. Έχει 7 modes λειτουργίας, οπότε αρκούν 3 bit για να διεθυνσιοδοτήσουν τα 7 modes από 000 έως 111. Η προεπιλογή ορίζεται στο έγγραφο TRM, αλλά ο πυρήνας του Linux τροποποιεί και συνήθως τοποθετεί τους ακροδέκτες στο mode 7, που είναι υψηλής αντίστασης και άρα αποσυνδεδεμένοι.

Στο παρελθόν το BBxM χρησιμοποιούσε το Linux Angstrom, που είχε παλαιό Linux kernel και έτσι με τη χρήση της ειδικής συσκευής `omap_mux` μπορούσε κάποιος να θέσει το επιθυμητό mode (https://groups.google.com/forum/#!topic/beagleboard/yKST0vHGOzA και http://labs.isee.biz/index.php/Mux_instructions), οπότε η χρήση των GPIO ήταν αρκετά εύκολη (https://elinux.org/EBC_Exercise_10_Flashing_an_LED).

Το δύσκολο στοιχείο στους νέους πυρήνες είναι η τοποθέτηση των κατάλληλων GPIO ακροδεκτών στο mode που θέλουμε.

Το BBxM έχει έναν μεγάλο 28pin ακροδέκτη στην άκρη που ονομάζεται “Expansion Connector” P9 (Table 22. Expansion Connector Signals σελίδα 108/164) και επίσης 3 μικρούς που φέρουν τις ετικέτες P11 (Table 24. P11 GPIO Signals 110/164), P13 (Table 25. P13 GPIO Signals 111/164), P17 (Table 26. P17 Auxiliary Expansion Signals 112/164). Όπως μπορεί να δει κάποιος στους πίνακες, οι ακροδέκτες σε αυτούς τους συνδετήρες έχουν πολλαπλές λειτουργίες, και ορίζονται από το mode λειτουργίας. Ο κάθε ακροδέκτης μπορεί να τεθεί σε διαφορετικό mode λειτουργίας με το να τεθεί μια τιμή σε έναν καταχωρητή. Κάθε καταχωρητής κατάστασης είναι 32bit και χρησιμοποιείται για 2 ακροδέκτες (16bit ανά ακροδέκτη). Κάθε καταχωρητής έχει μια δικιά του διεύθυνση. Π.χ. 0x48002168 [31:16] για τον ακροδέκτη MMC2_DAT7 και 0x48002168 [15:0] για τον ακροδέκτη MMC2_DAT6. Οι διευθύνσεις αναγράφονται

στον πίνακα Table 7-4. Core Control Module Pad Configuration Register Fields στη σελίδα 778/3492 του OMAP35x TRM.

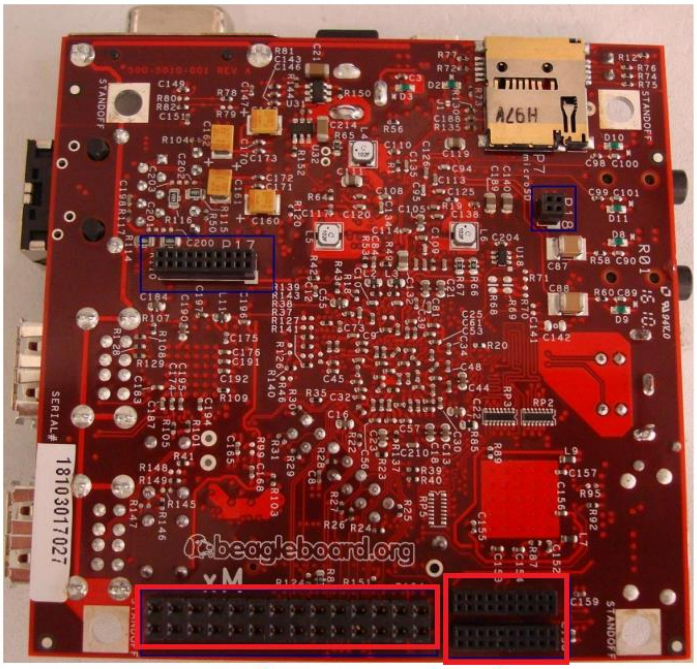



**Figure 82. BeagleBoard-xM Expansion Headers**

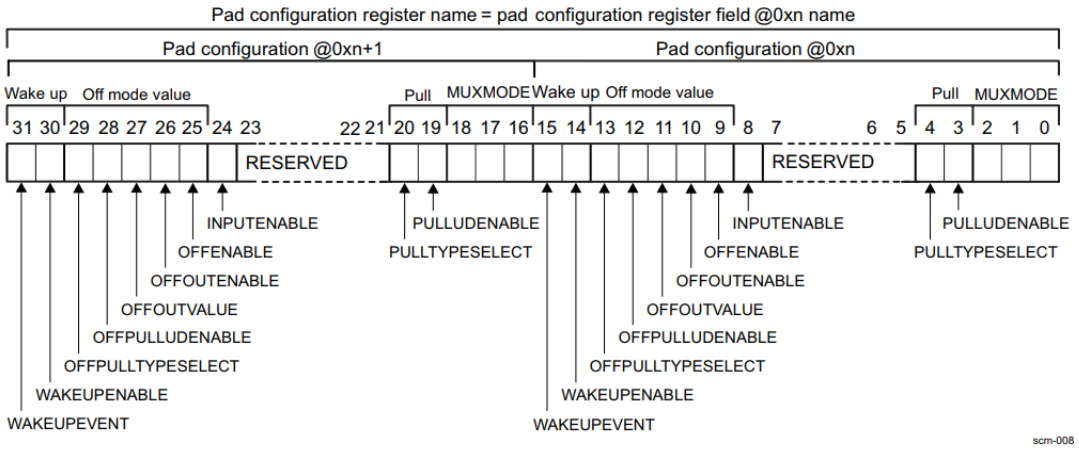
Έστω επιλέγουμε να χρησιμοποιήσουμε το P9 Expansion Connector (Table 22), και συγκεκριμένα να θέσουμε κάποιους ακροδέκτες σε κατάσταση 4 (mode 4). Επιλέγουμε το GPIO_139 στο pin3 (όνομα MMC2_DAT7), GPIO_138 στο pin5 (όνομα MMC2_DAT6). Οι ακροδέκτες φέρουν το όνομα της λειτουργίας που αντιστοιχεί στο mode 0, και στο TRM τοποθετείται από μπροστά το `CONTROL_PADCONF_`

**Table 22. Expansion Connector Signals**

| EXP | Processor | 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 | | | | | | | |

Το `CONTROL_PADCONF_????` είναι ο καταχωρητής που καθορίζει την κατάσταση κάθε ακροδέκτη στα expansion headers. Όπως αναφέρθηκε, τα χαμηλά 16bit καθορίζουν έναν ακροδέκτη και τα υψηλά 16bit έναν άλλο. Ο πίνακας 7-7 Pad Configuration Register Functionality στη σελίδα 769/3492 μας εμφανίζει την ανάλυση του κάθε bit σε συνδυασμό με την επόμενη σελίδα. Για παράδειγμα αν έχουμε την 16bit τιμή 0x4 τότε σημαίνει ότι έχουμε μια έξοδο (inputenable=0), σε mode 4 (muxmode=100), χωρίς pull (pulldenable, pulltypeselect=00), και απενεργοποιημένες τις υπόλοιπες καταστάσεις όπως OFFXXX kai WAKEXXX. Τους αριθμούς δε χρειάζεται να τους θυμόμαστε, παρά μόνο για επιβέβαιωση ότι η κατάσταση που έχουμε ζητήσει έχει ενεργοποιηθεί στον πυρήνα του Linux, όπως θα δούμε προς το τέλος αυτού του φυλλαδίου.

**Figure 7-7. Pad Configuration Register Functionality**



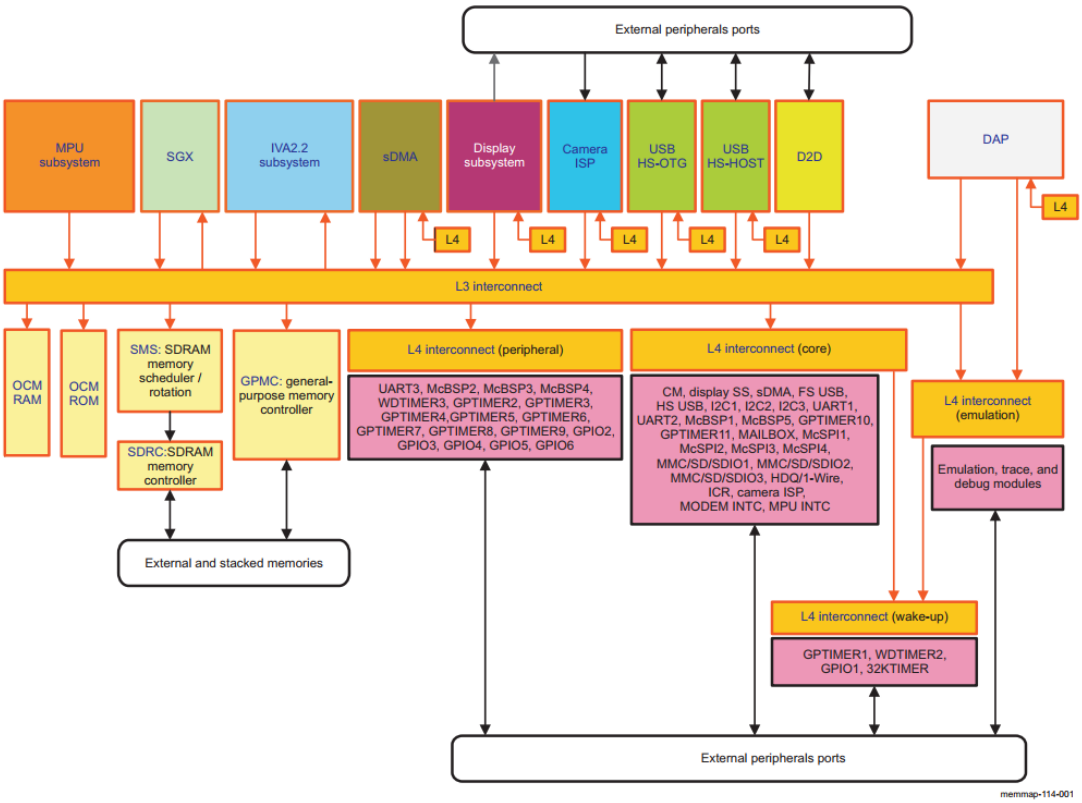
# 3. Το Device Tree στο Linux

Μια σημαντική αλλαγή στον πυρήνα του Linux έγινε το 2013/2014 όπου απομακρύνθηκαν κάποιες ρυθμίσεις από τον ίδιο τον πυρήνα (που απαιτούσαν το χρονοβόρο re-compilation αν απαιτούσε να αλλάξει κάποια ρύθμιση) και τοποθετήθηκαν σε ένα επιπρόσθετο αρχείο που ονομάζεται Device Tree. Αυτό χρησιμοποιείται στα ενσωματωμένα συστήματα που έχουν πολλαπλές αρχιτεκτονικές σε αντίθεση με τους τυπικούς υπολογιστές που έχουν τυποποιημένες αρχιτεκτονικές. Το Device Tree απεικονίζει τις συνδέσεις των στοιχείων και την αρχιτεκτονική της πλατφόρμας. Στην αρχιτεκτονική ορίζονται διευθύνσεις επικοινωνίας, θύρες εισόδου, εξόδου και που συνδέονται. Επίσης, κάθε στοιχείο που τοποθετείται στο device tree, συνήθως είναι συμβατό με κάποιο άλλο, οπότε δε χρειάζονται να τοποθετούνται όλες οι ρυθμίσεις για τη νέα αρχιτεκτονική. Π.χ. σε μια αρχιτεκτονική μπορούμε να αναφέρουμε ότι το XXX στοιχείο είναι UART compatible και άρα δε θα δώσουμε όλες τις παραμέτρους για αυτό, αφού θα αντιγραφούν οι default παράμετροι από το βασικό στοιχείο της βιβλιοθήκης UART.

Στον πηγαίο κώδικα του linux υπάρχουν αρκετές περιγραφές device trees (στη διαδρομή **`arch/arm/boot/dts`**) για διάφορες αναπτυξιακές πλακέτες (και για beagleboard). Η προεπιλογή για το BBxM (αρχείο **`omap3-beagle-xm.dts`**) είναι να μη χρησιμοποιούνται τα GPIO (δεν ορίζονται καθόλου), οπότε θα πρέπει εμείς να επεξεργαστούμε το default Device Tree για το BBxM, και να προσθέσουμε ένα block που να καθορίζει τα gpio που θα χρησιμοποιήσουμε. Το block αυτό όμως δε μπορούμε να το προσθέσουμε σε οποιοδήποτε σημείο του Device Tree. Θα πρέπει να εξετάσουμε την αρχιτεκτονική του TI OMAP3 και να το συνδέσουμε στο κατάλληλο σημείο στο δένδρο.

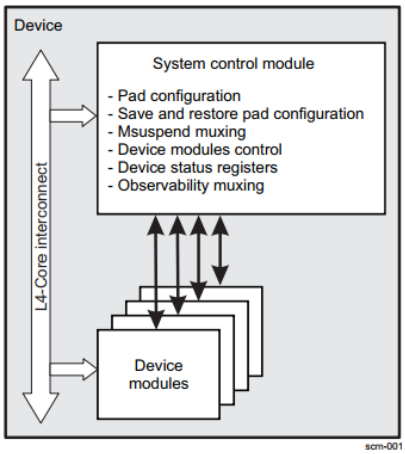
Το σχήμα 2-1 Interconnect Overview στη σελίδα 196/3492 (OMAP3x TRM) μας δείχνει ότι τα GPIO2 έως και GPIO6 συνδέονται στο L4 interconnect (peripheral), ενώ το GPIO1 στο L4 interconnect (wake-up).

**Figure 2-1. Interconnect Overview**



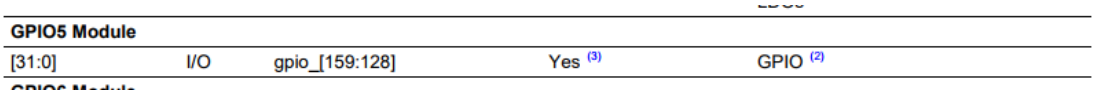

Από το σχήμα βλέπουμε επίσης ότι όλα τα modules συνδέονται στο External Peripherals Ports, που είναι μια συστοιχία από πολυπλέκτες και διαχειρίζεται από το SCM (System Control Module). Το σχήμα 7.1 SCM Overview (σελίδα 761/3492) δείχνει τις λειτουργίες του SCM.

**Figure 7-1. SCM Overview**




Σε αυτή την άσκηση επιλέγουμε να χρησιμοποιήσουμε τα pins: **GPIO_133, GPIO_136, GPIO_137, GPIO_138, GPIO_139**. Θα πρέπει να βρούμε σε ποιο GPIO controller αντιστοιχούν από τους 6 που έχουμε στον επεξεργαστή. Για να το βρούμε αυτό θα χρησιμοποιήσουμε τον πίνακα 24-5 στο GPIO Channel Description σελίδα 3371/3492 του OMAP3x TRM. Παρατηρούμε ότι τα συγκεκριμένα GPIO βρίσκονται όλα στο

GPIO5 module. Επίσης σε εκείνη τη σελίδα μας δίνονται και άλλες πληροφορίες, όπως π.χ. κάποια GPIO που είναι υποχρεωτικά input ή output (σε άλλα GPIO controllers, όχι στον GPIO5).



| GPIO5 Module | | | | |
|---|---|---|---|---|
| [31:0] | I/O | gpio_[159:128] | Yes [3] | GPIO [2] |

Έχοντας αυτή τη γνώση μπορούμε να τροποποιήσουμε το Linux Device Source Tree.

Κατά τη διαδικασία εκκίνησης της πλακέτας μας στη σειριακή οθόνη, μας αναφέρεται το αρχείο dtb που διαβάζεται μαζί με τον πυρήνα. Στο BBxM είναι το omap3-beagle-xm.dtb . Θα πρέπει να δημιουργήσουμε ένα τέτοιο αρχείο και να το τοποθετήσουμε στο /boot διαμέρισμα της SD κάρτας σβήνοντας το παλαιό (**το ίδιο να γίνει και στο USB flash drive, αν χρησιμοποιούμε root partition σε USB**).

Για να γινεί αυτό θα πρέπει να εγκαταστήσουμε κάποια προγράμματα για τη μεταγλώττιση (ίσως να μη χρειαστεί, δοκιμάστε να κάνετε τα επόμενα βήματα πρώτα, και αν εμφανιστεί πρόβλημα τότε μπορείτε να εγκαταστήσετε αυτά τα πακέτα).

```
sudo apt-get -y install bc curl gcc git libncurses5-dev lzop
make u-boot-tools ncurses-dev make gcc-arm-linux-gnueabi
```

(Σημείωση: αναλόγως της διανομής linux που έχετε, ίσως το `gcc-arm-linux-gnueabi` να αναφέρεται `gcc-arm-none-eabi` ή `gcc-arm-none-gnueabi`. Σε κάθε περίπτωση, μπορείτε να δώσετε `apt-cache search gcc-arm` και να δείτε πως ονομάζεται το πακέτο για τη διανομή σας).

στη συνέχεια θα φέρουμε στον τοπικό δίσκο (επιβεβαιώστε ότι έχετε 4-5GB ελεύθερα στον τρέχοντα κατάλογο) το δένδρο που αντιστοιχεί στον πυρήνα που χρησιμοποιούμε, π.χ. για το Ubuntu 16, ο πυρήνας φέρει το όνομα xenial. Αλλάζουμε τις μεταβλητές που καθορίζουν ότι θα προβούμε σε μεταγλώττιση για επεξεργαστή ARM, ενεργοποιούμε τη δυνατότητα να αντιγραφεί ένα οποιοδήποτε default αρχείο για ένα board για επεξεργαστή ARM (δε μας ενδιαφέρει ποιο, αρκεί να δημιουργήσει ένα αρχείο **`.config`** για να μην παραπονεθεί το επόμενο βήμα). Στη συνέχεια κάνουμε compile μόνο τα dtbs (device tree blob source) και όχι όλο τον πυρήνα του Linux. Μόλις ολοκληρωθεί με επιτυχία, τότε θα είμαστε σίγουροι ότι μπορούμε να κάνουμε compile dts σε dtb και έτσι θα μεταβούμε στην τροποποίηση του default configuration για το DTS του BBxM.

```
1) git clone git://kernel.ubuntu.com/ubuntu/ubuntu-xenial.git
2) cd ubuntu-xenial
3) export ARCH=arm; export CROSS_COMPILE=/usr/bin/arm-linux-gnueabi-
   (η παύλα στο τέλος δεν είναι τυπογραφικό λάθος)
4) make omap2plus_defconfig
5) make dtbs
```

Εναλλακτικά, μπορούμε να κάνουμε το εξής (δεν προτιμάται όπως θα εξηγηθεί παρακάτω). Να κάνουμε decompile το dtb που χρησιμοποιεί το board μας σε dts με μια εντολή παρόμοια με αυτή:

```
dtc -I dtb -O dts /boot/dtbs/4.12.5-armv7-x3/omap3-beagle-xm.dtb
> omap3-beagle-xm.dts
```

να επεξεργαστούμε το αρχείο που θα δημιουργηθεί: omap3-beagle-xm.dts και στη συνέχεια να το κάνουμε πάλι compile με:

```
dtc -I dts -O dtb -o /boot/dtbs/4.12.5-armv7-x3/omap3-beagle-
xm.dtb omap3-beagle-xm.dts
```

Αυτό δε προτιμάται, επειδή κατά το decompile απομακρύνονται κάποια labels και δε μπορούμε να χρησιμοποιήσουμε defines, όπως PIN_OUTPUT και έτσι δεν είναι τόσο απλό, τουλάχιστον όχι από κάποιον που δεν έχει μεγάλη εμπειρία. Στο decompiled αρχείο μας εμφανίζονται οι τιμές των καταχωρητών εξόδου με τη δεκαεξαδική τιμή (π.χ. 0x4) και όχι με τις ετικέτες που έχουμε δώσει στο αρχικό πηγαίο αρχείο (π.χ. PIN_OUTPUT | MUX_MODE4). Μπορούμε βέβαια τη δεκαεξαδική τιμή που μας εμφανίζεται να την αποκωδικοποιήσουμε (και να την τροποποιήσουμε) σύμφωνα με τον πίνακα 7-7 Pad Configuration Register Functionality στη σελίδα 769/3492 που εμφανίσαμε προηγουμένως.

---

Πριν τροποποιήσουμε το device tree καλό είναι να κοιτάξετε λίγο το **/sys/kernel/debug/pinctrl/** και να δείτε σε ποιο pinmux βρίσκονται οι διευθύνσεις που θέλετε να τροποποιήσετε με την παρακάτω διαδικασία. Θέλουμε να βρούμε το pin GPIO_139 που έχει διεύθυνση 0x48002168. Εισερχόμαστε σε κάθε κατάλογο που φέρει το όνομα pinmux (`48002030.pinmux`, `480025a0.pinmux`, `48002a00.pinmux`) και διαβάζουμε το αρχείο `pins`. Σε ένα από αυτά θα υπάρχει η συγκεκριμένη διεύθυνση, π.χ.

```
pin 156 (PIN156) 48002168 ..... pinctrl-single
```

Επίσης, στο αρχείο pinmux-pins θα δούμε ότι το συγκεκριμένο PIN αναγράφεται ως

```
pin 156 (PIN156): (MUX UNCLAIMED) (GPIO UNCLAIMED)
```

οπότε μπορούμε να το τροποποιήσουμε. Αν ήταν δεσμευμένο κάπου, θα έπρεπε να αποδεσμεύσουμε τη σύνδεση (να σβήσουμε την καταχώρηση από το device tree).

Επίσης, όταν βρούμε το pinmux που φέρει αυτό το PIN (στη δική μας περίπτωση είναι το `48002030.pinmux`) μπορούμε να δούμε και να σημειώσουμε τι άλλα modules φέρει στο αρχείο pinmux-functions (στη δικιά μας περίπτωση `pinmux_hsusb2_pins`, `pinmux_uart3_pins`, `pinmux_dss_dpi_pins2`, `pinmux_twl4030_pins`). Αυτά τα ονόματα θα μας βοηθήσουν στην επεξεργασία του device tree, επειδή θα προσθέσουμε το δικό μας block στο σημείο που βρίσκονται όλα αυτά μαζί αποφεύγοντας ένα πιθανό λάθος σύνδεσης στην ιεραρχία των modules.

Σε αυτό το σημείο να αναφέρουμε ότι το `48002030.pinmux` δεν είναι τυχαίο. Σύμφωνα με τη σελίδα 769/3492 OMAP3x TRM:

"*Some pad configuration registers control the configuration of pads in the CORE power domain. These registers are instantiated in the CORE power domain of the SCM (core control module, physical addresses 0x4800 2030 to 0x4800 2260). Pad configuration registers also control the configuration of pads in the WKUP power domain. These registers are instantiated in the WKUP power domain of the SCM (wake-up control module, physical addresses 0x4800 2A00 to 0x4800 2A4C).*"

Δηλαδή το `48002030.pinmux` είναι το Core power domain του SCM, ενώ το `48002A00.pinmux`, είναι το Wake Up Power Domain του SCM.

Επεξεργαζόμαστε το αρχείο `arch/arm/boot/dts/omap3-beagle-xm.dts` μέσα στον πηγαίο κατάλογο του linux kernel που έχουμε φέρει με git.

Βρίσκουμε τα modules που είχαμε σημειώσει πιο πριν και παρατηρούμε ότι είναι μέσα στο device tree entry **`&omap3_pmx_core { … }`** . Αυτό το & σημαίνει ότι θα εμπλουτίσουμε το default configuration του omap3_pmx_core με κάποια επιπρόσθετα modules. Ανάμεσα σε αυτά τα modules θα προσθέσουμε ένα block που θα το ονομάσουμε **gpio5_pins**, επειδή πρόκειται να αλλάξουμε κάποια pins που ανήκουν στο gpio5 (όπως είχαμε βρει πιο πριν). Για κάθε pin θα πρέπει να υπολογίσουμε το offset μέσα στον controller. Αυτό βρίσκεται από τον πίνακα 7-66 στη σελίδα 861/3492, όπου παρατηρούμε για το pin CONTROL_PADCONF_**MMC2_DAT6** που αντιστοιχεί στο GPIO_138 και βρίσκεται στη διεύθυνση 48002168 [15:0] (όπως είχαμε υπολογίσει προηγουμένως) είναι **`0x138`**. Στην περίπτωση που θέλουμε το pin GPIO_139 που φέρει το όνομα MMC2_DAT7 δε θα βρούμε το CONTROL_PADCONF_**MMC2_DAT7** γιατί υπενθυμίζουμε ότι κάθε 2 ακροδέκτες έχουν ένα CONTROL_PADCONF των 32bit. Οπότε το GPIO_139 που βρίσκεται στη διεύθυνση 48002168 [31:16] χρησιμοποιεί τον καταχωρητή CONTROL_PADCONF_**MMC2_DAT6** αφού όμως προσθέσουμε +2Byte, οπότε η διεύθυνση του είναι **`0x138+2=0x13A`**.

Υπολογίζουμε όλες τις μετατοπίσεις για τα pins που θέλουμε να χρησιμοποιήσουμε και δημιουργούμε ένα block `pinmux_gpio5_pins` στο οποίο τοποθετούμε το offset και μέσα σε παρένθεση τις σημαίες λειτουργίας χωρισμένες με την πράξη OR (αφού έχουν άλλη βαρύτητα). Πιθανές σημαίες: `PIN_OUTPUT_PULLUP`, `PIN_OUTPUT`, `PIN_INPUT_PULLUP`, `PIN_INPUT`, `MUX_MODE4` και άλλες που συνδέονται με τις υπόλοιπες δυνατότητες. Αυτές οι λέξεις θα μετατραπούν στην αντίστοιχη δεκαεξαδική λέξη, π.χ. `0x4`. Κάποιος μπορεί να γράψει αντί για αυτές τις λέξεις την δεκαδεξαδική λέξη των σημαιών (όπως φαίνεται αν κάνει κάποιος decompile το dtb σε dts). Π.χ. αν στο τέλος υπάρχει η λέξη 0x4 τότε σε 16bit είναι 000000000000100 που σύμφωνα με τον πίνακα 7-7 (σελίδα 7 του φυλλαδίου) σημαίνει ότι είναι mode=4 χωρίς pull up και pull –down και είναι έξοδος. Αν ήταν 0x11c, δηλαδή σε 16bit δυαδικό: 0000000100011100 σημαίνει mode=4, με pull up mode =11 και ρύθμιση ως είσοδο.

**Table 7-66. PADCONFS Register Summary (continued)**

| Register Name | Type | Register Width (Bits) | Address Offset | Physical Address |
|---|---|---|---|---|
| CONTROL_PADCONF_CAM_D9 | RW | 32 | 0x0000 00F8 | 0x4800 2128 |
| CONTROL_PADCONF_CAM_D11 | RW | 32 | 0x0000 00FC | 0x4800 212C |
| CONTROL_PADCONF_CAM_WEN | RW | 32 | 0x0000 0100 | 0x4800 2130 |
| CONTROL_PADCONF_CSI2_DX0 | RW | 32 | 0x0000 0104 | 0x4800 2134 |
| CONTROL_PADCONF_CSI2_DX1 | RW | 32 | 0x0000 0108 | 0x4800 2138 |
| CONTROL_PADCONF_MCBSP2_FSX | RW | 32 | 0x0000 010C | 0x4800 213C |
| CONTROL_PADCONF_MCBSP2_DR | RW | 32 | 0x0000 0110 | 0x4800 2140 |
| CONTROL_PADCONF_MMC1_CLK | RW | 32 | 0x0000 0114 | 0x4800 2144 |
| CONTROL_PADCONF_MMC1_DAT0 | RW | 32 | 0x0000 0118 | 0x4800 2148 |
| CONTROL_PADCONF_MMC1_DAT2 | RW | 32 | 0x0000 011C | 0x4800 214C |
| CONTROL_PADCONF_MMC1_DAT4 | RW | 32 | 0x0000 0120 | 0x4800 2150 |
| CONTROL_PADCONF_MMC1_DAT6 | RW | 32 | 0x0000 0124 | 0x4800 2154 |
| CONTROL_PADCONF_MMC2_CLK | RW | 32 | 0x0000 0128 | 0x4800 2158 |
| CONTROL_PADCONF_MMC2_DAT0 | RW | 32 | 0x0000 012C | 0x4800 215C |
| CONTROL_PADCONF_MMC2_DAT2 | RW | 32 | 0x0000 0130 | 0x4800 2160 |
| CONTROL_PADCONF_MMC2_DAT4 | RW | 32 | 0x0000 0134 | 0x4800 2164 |
| CONTROL_PADCONF_MMC2_DAT6 | RW | 32 | 0x0000 0138 | 0x4800 2168 |
| CONTROL_PADCONF_MCBSP3_DX | RW | 32 | 0x0000 013C | 0x4800 216C |
| CONTROL_PADCONF_MCBSP3_CLKX | RW | 32 | 0x0000 0140 | 0x4800 2170 |
| CONTROL_PADCONF_UART2_CTS | RW | 32 | 0x0000 0144 | 0x4800 2174 |
| CONTROL_PADCONF_UART2_TX | RW | 32 | 0x0000 0148 | 0x4800 2178 |
| CONTROL_PADCONF_UART1_TX | RW | 32 | 0x0000 014C | 0x4800 217C |
| CONTROL_PADCONF_UART1_CTS | RW | 32 | 0x0000 0150 | 0x4800 2180 |
| CONTROL_PADCONF_MCBSP4_CLKX | RW | 32 | 0x0000 0154 | 0x4800 2184 |
| CONTROL_PADCONF_MCBSP4_DX | RW | 32 | 0x0000 0158 | 0x4800 2188 |
| CONTROL_PADCONF_MCBSP1_CLKR | RW | 32 | 0x0000 015C | 0x4800 218C |
| CONTROL_PADCONF_MCBSP1_DX | RW | 32 | 0x0000 0160 | 0x4800 2190 |
| CONTROL_PADCONF_MCBSP_CLKS | RW | 32 | 0x0000 0164 | 0x4800 2194 |
| CONTROL_PADCONF_MCBSP1_CLKX | RW | 32 | 0x0000 0168 | 0x4800 2198 |

Θα προσθέσουμε το παρακάτω block (**στο κατάλληλο σημείο**) που ορίζει τις σημαίες στα pins.

```
gpio5_pins: pinmux_gpio5_pins {

                pinctrl-single,pins = <

                        0x12C (PIN_INPUT_PULLUP | MUX_MODE4) /* sdmmc2_dat1.gpio_133 */

                        0x134 (PIN_INPUT_PULLUP | MUX_MODE4) /* sdmmc2_dat4.gpio_136 */

                        0x136 (PIN_INPUT_PULLUP | MUX_MODE4) /* sdmmc2_dat5.gpio_137 */

                        0x138 (PIN_OUTPUT_PULLUP | MUX_MODE4) /*sdmmc2_dat6.gpio_138 */

                        0x13A (PIN_OUTPUT | MUX_MODE4) /* sdmmc2_dat7.gpio_139 */

                        >;

};
```

Τέλος, πρέπει να τοποθετήσουμε εκτός του block αυτού (κάτω από το `&gpio1` κατά προτίμηση), ένα block ενεργοποίησης των pins με την παρακάτω μορφή:

```
&gpio5 {

    pinctrl-names = "default";

    pinctrl-0 = <&gpio5_pins>;

    };
```

Κάνουμε compile με **`make dtbs`** και αν δεν έχουμε κάνει κάποιο λάθος θα δημιουργηθεί το .dtb αρχείο. Αυτό θα το αντιγράψουμε στην SD card και στο USB (αν χρησιμοποιείται) στο διαμέρισμα που βρίσκεται ο κατάλογος /boot, στη θέση που αναγράφεται κατά τη διαδικασία εκκίνησης (π.χ. στο **`/dev/sda2`** **`/boot/dtbs/έκδοση_kernel/`** και στο **`/dev/mmc0blkp2`** ίδιο υποκατάλογο).

## 4. Δοκιμές εισόδου/εξόδου GPIO

Ως αυτό το σημείο έχουμε τροποποιήσει το device tree, το έχουμε αντιγράψει στη SD κάρτα στο 2ο διαμέρισμα που βρίσκεται το root file system, στην κατάλληλη τοποθεσία **`/boot/dtbs/έκδοση_kernel/`** (και αν χρησιμοποιούμε USB FlashDrive στην αντίστοιχη τοποθεσία εκεί), και είμαστε έτοιμοι να χρησιμοποιήσουμε την είσοδο έξοδο.

Σύμφωνα με τις προδιαγραφές του επεξεργαστή υπάρχουν περιορισμοί στην ένταση που μπορεί να δώσει ένα pin (ονομάζεται source current) ή να λάβει ως επιστροφή (ονομάζεται sink current). Σύμφωνα με τον πίνακα 4-1. Pin Attributes (σελίδα 22), η δύναμη οδήγησης των GPIO είναι 6mA.

**Table 4-1. Pin Attributes (ZCE and ZCZ Packages) (continued)**

| ZCE BALL NUMBER [1] | ZCZ BALL NUMBER [1] | PIN NAME [2] | SIGNAL NAME [3] | MODE [4] | TYPE [5] | BALL RESET STATE [6][23] | BALL RESET REL. STATE [7] | RESET REL. MODE [8] | ZCE POWER / ZCZ POWER [9] | HYS [10] | BUFFER STRENGTH (mA) [11] | PULLUP /DOWN TYPE [12] | I/O CELL [13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NA | V16 | GPMC_A8 | gpmc_a8 | 0 | O | L | L | 7 | NA / VDDSHV3 | Yes | 6 | PU/PD | LVCMOS |
| | | | gmii2_rxd3 | 1 | I | | | | | | | | |
| | | | rgmii2_rd3 | 2 | I | | | | | | | | |
| | | | mmc2_dat6 | 3 | I/O | | | | | | | | |
| | | | gpmc_a24 | 4 | O | | | | | | | | |
| | | | pr1_mii1_rxd0 | 5 | I | | | | | | | | |
| | | | mcasp0_aclkx | 6 | I/O | | | | | | | | |
| | | | gpio1_24 | 7 | I/O | | | | | | | | |

Αν συνδέσουμε το LED σε ένα GPIO (1.8 Volt), τότε μας αρκεί μια αντίσταση 330 Ωhm ως 680 Ωhm για να περιορίσει το ρεύμα σε κάποια mA. Να σημειώσετε ότι ένα κόκκινο LED απαιτεί 2Volt για να ανάψει, οπότε το GPIO οριακά μπορεί να το οδηγήσει. Αν χρησιμοποιήσετε άλλο χρώμα (π.χ. μπλε 430nm) τότε αυτό απαιτεί μεγαλύτερη τάση και δε θα μπορέσει να ενεργοποιηθεί.

Δημιουργούμε ένα κύκλωμα (2 καλώδια Dupont, 1 αντίσταση, 1 κόκκινο LED) στο breadboard ως εξής:

1. 1 καλώδιο Dupont συνδέεται στο breadboard με τη μια άκρη στη θετική άκρη του LED (με το μεγαλύτερο μήκος). Η άλλη άκρη του καλωδίου την αφήνουμε στον αέρα και την ονομάζουμε V_in.
2. Το LED βρίσκεται τοποθετημένο στο breadboard προσέχοντας την πόλωση (μεγάλη άκρη / μικρή άκρη).
3. 1 αντίσταση τοποθετείται στο breadboard και συνδέεται στον αρνητικό ακροδέκτη του LED.
4. 1 άλλο καλώδιο συνδέεται στην άλλη άκρη της αντίστασης, και το τοποθετούμε στον ακροδέκτη της γείωσης GND, δηλαδή στη θέση 27 ή 28 του P9.

Έχουμε συνδεδεμένο το σειριακό καλώδιο στο BBxM και στον υπολογιστή μας, και έχουμε ενεργοποιήσει τη σύνδεση με το πρόγραμμα σειριακής επικοινωνίας (putty ή cu ή minicom ή screen).

Κάνουμε power-on το BBxM και μόλις μας βγει η προτροπή "Hit any key to stop autoboot" σταματάμε τη διαδικασία εκκίνησης με ESC.

## 4.1 Δοκιμή Α – Έλεγχος συνδεσιμότητας του breadboard

Συνδέουμε την ελεύθερη άκρη του καλωδίου (που ονομάζουμε V_in) στο pin 1 του ακροδέκτη επέκτασης P9 που φέρει το όνομα VIO_1V8, δηλαδή ότι είναι 1.8 Volt. Θα πρέπει το LED να ανάψει. Αν ανάψει τότε ο έλεγχος αυτός έχει περάσει με επιτυχία. Αν δεν ανάψει, υπάρχει πρόβλημα στο κύκλωμα. Επίσης, μπορείτε να χρησιμοποιήσετε το pin 2 του ακροδέκτη επέκτασης P9 που φέρει το όνομα DC_5V και φέρει 5Volt και το LED θα ανάψει πολύ πιο έντονα, αλλά μην το διατηρήσετε πολύ ώρα, γιατί θα δημιουργηθεί μεγαλύτερη ένταση ρεύματος που θα επιστρέψει στη γείωση (και θα απαιτούσε μεγαλύτερη αντίσταση για να περιορίσει το ρεύμα στους ακροδέκτες). Αν δεν ανάψει το LED το πιο πιθανό είναι ότι έχουμε ανάποδη τη πόλωση, ή ότι δεν κάνει επαφή κάποιο καλώδιο ή ότι είναι καμένο το LED. Επίσης, μπορούμε με ένα πολύμετρο να μετρήσουμε τη διαφορά τάσης που έχουμε στο PIN2 και PIN27, όπου θα πρέπει να είναι 5V.

## 4.2 Δοκιμή Β – Έλεγχος επαφής GPIO

Ενώ βρισκόμαστε στην προτροπή του boot loader αφού έχουμε διακόψει τη διαδικασία boot θα επιβεβαιώσουμε ότι προγραμματιστικά λειτουργεί ο ακροδέκτης που έχουμε επιλέξει. Τοποθετούμε τη θετική άκρη του καλωδίου στο PIN3 (GPIO_139) και δίνουμε την ίδια εντολή αρκετές φορές στη σειριακή οθόνη του beagleboard-xm (εντολή προς το Boot loader):

```
gpio toggle 139
```

κάθε φορά που θα το δίνουμε θα βλέπουμε να αλλάζει η κατάσταση του LED. Αν είναι κλειστό θα ανάψει, αν είναι ανοιχτό θα σβήσει. Επαναλάβετε την ίδια άσκηση με το GPIO_161 (σύμφωνα με τον πίνακα 22, βρίσκεται στο PIN16). Αν ανάβει το LED τότε προγραμματιστικά ο ελεγκτής GPIO είναι λειτουργικά σωστός (δεν έχει καεί π.χ. κάποιος ακροδέκτης) και μπορείτε να ξεκινήσετε τη διαδικασία boot για τον τελικό έλεγχο.

Επίσης, ενδέχεται το uboot να υποστηρίζει είσοδο. Το GPIO_137 θα το χρησιμοποιήσουμε ως είσοδο. Με τη βοήθεια του διδάσκοντα συνδέστε ένα καλώδιο στη γείωση διαβάστε την τιμή του ακροδέκτη με gpio input 137 στη συνέχεια συνδέστε τον ακροδέκτη στη γείωση (ή στο 1V8 ακροδέκτη) και διαβάστε ξανά την τιμή. Θα πρέπει να έχει αλλάξει. Θα μπορούσατε να χρησιμοποιήσετε ένα script uboot όπως το παρακάτω για να αλλάζετε ένα led αν έχετε πατημένο ένα πλήκτρο:

```
if gpio input 137; then
    gpio toggle 139; "
    sleep 1;
```

```
    gpio toggle 139;

    sleep 1;

fi;
```

Δώστε **boot** για να ξεκινήσει το boot του Linux.

## 4.3 Δοκιμή Γ – Έλεγχος του GPIO μέσω λειτουργικού συστήματος-Έξοδος

Αρχικά πρέπει να σημειωθεί ότι το Linux βλέπει τους ακροδέκτες του GPIO με μια μετατόπιση +32. Οπότε αν θέλουμε να τροποποιήσουμε τον ακροδέκτη GPIO_161, το Linux το βλέπει ως 161+32 = 193 (αυτό οφείλεται σε εσωτερική αρίθμηση του πυρήνα του linux και το ακριβές αίτιο είναι άγνωστο σε όλους, εκτός από αυτούς που προγραμματίζουν τον πυρήνα Linux). Θα πρέπει να επιβεβαιώσετε ότι στο αρχείο που απεικονίζονται οι συνδέσεις των ακροδεκτών (κατάλογος: **/sys/kernel/debug/pinctrl/**48002030.pinmux/), υπάρχει το block gpio5_pins που έχουμε ήδη δημιουργήσει. Αν δεν υπάρχει αυτό, τότε σημαίνει ότι δεν έχουμε αντιγράψει το Device Tree που δημιουργήσαμε στην κατάλληλη τοποθεσία.

Αφού επιβεβαιώσουμε ότι υπάρχει η αντιστοίχηση με τους ακροδέκτες, ενεργοποιήστε το GPIO με μια εντολή εγγραφής της τιμής του GPIO στο ειδικό αρχείο **export** ως εξής (π.χ. για τον ακροδέκτη GPIO_138 της πλακέτας, που είναι το 170 για το Linux)

```
echo 170 > /sys/class/gpio/export
```

Με την εγγραφή αυτή, δημιουργείται ένας κατάλογος ελέγχου του GPIO με όνομα **/sys/class/gpio/gpio170/** . Στη συνέχεια, δηλώστε τον ακροδέκτη ως έξοδο (σε αντιστοιχία με το Device Tree που και εκεί το είχατε δηλώσει ως έξοδο).

```
echo out > /sys/class/gpio/gpio170/direction
```

Μπορείτε να θέσετε τιμή 0 ή τιμή 1 με τις παρακάτω εντολές αντίστοιχα (και να δείτε το LED να σβήνει ή να ανάβει):

```
echo 0 > /sys/class/gpio/gpio170/value

echo 1 > /sys/class/gpio/gpio170/value
```

Αν με τις παρακάτω εντολές τροποποιείτε την κατάσταση του LED, η δοκιμή έχει στεφθεί με επιτυχία. Να σημειώστε ότι στο device tree του φυλλαδίου, θέσαμε ως μια έξοδο χωρίς pullup και μια με pullup. Τοποθετήστε την V_in άκρη του καλωδίου είτε στον ένα ακροδέκτη είτε στον άλλο και δείτε τη διαφορά (σε έναν ακροδέκτη θα δείτε κάποιο πολύ μικρό φωτισμό επειδή δεν έχει pull up αντίσταση, ενώ στον άλλο θα είναι τελείως σβηστό).

## 4.4 Δοκιμή Δ – Έλεγχος του GPIO μέσω λειτουργικού συστήματος-Είσοδος

Στην είσοδο θα χρησιμοποιήσουμε ένα κουμπί (αν δεν έχετε μπορείτε να χρησιμοποιήσετε και ένα καλώδιο). Η είσοδος μπορεί να έχει pullup αντίσταση (όπως το έχουμε ορίσει στο Device Tree για το συγκεκριμένο ακροδέκτη) προς το Vcc, οπότε

αν διαβαστεί η τιμή χωρίς να έχουμε συνδεδεμένο καλώδιο θα φέρει τιμή ‘1’. Σε περίπτωση που έχει pullup, εμείς θέλουμε να αλλάξουμε την τιμή σε ‘0’, οπότε αυτό επιτυγχάνεται με το να κλείσει ένα κύκλωμα προς το GND. Στο παράδειγμά μας, ο ακροδέκτης GPIO_137 (δηλαδή 137 + 32 = 169 στο Linux) έχει Input_Pullup.

Ενεργοποιούμε τον ακροδέκτη, θέτουμε την κατάσταση σε είσοδο, και διαβάζουμε την τρέχουσα τιμή ως εξής:

```
echo 169 > /sys/class/gpio/export

echo out > /sys/class/gpio/gpio169/direction

cat /sys/class/gpio/gpio169/value
```

Μπορούμε να δημιουργήσουμε ένα script ή μια εντολή όπως η παρακάτω για να διαβάζει συνεχώς την τιμή και να μας την εμφανίζει:

```
while [ 1 -eq 1 ] ; do cat /sys/class/gpio/gpio169/value ; sleep 1 ; done
```

[για να το σταματήσουμε πατάμε Control+C].

Αν δε λειτουργεί το GPIO_137 δοκιμάστε κάποιον άλλον ακροδέκτη από αυτούς που έχετε δηλώσει ως είσοδο, και επαναλάβετε τις παραπάνω εντολές.

Ενώ εκτελείται το script παίρνουμε ένα καλώδιο ή κουμπί και από τη μια πλευρά το τοποθετούμε στη γείωση GND της πλακέτας και την άλλη άκρη την κρατάμε στο χέρι. Αν ακουμπήσουμε με αυτή την άκρη τον ακροδέκτη GPIO_137 τότε η τιμή εκεί που εμφανίζονταν με ‘1’ θα γίνει ‘0’. Αντί για καλώδιο ασφαλώς μπορούμε να τοποθετήσουμε κουμπί που συνδέεται στη γείωση και η λογική θα είναι ακριβώς η ίδια.

**ΆΣΚΗΣΗ: Να τροποποιήσετε το Device Tree και να προσθέσετε μια νέα GPIO ως input_pulldown, και να επιβεβαιώστε την παρακάτω συνδεσμολογία.**

Σε περίπτωση που έχουμε είσοδο με pull_down, τότε σημαίνει ότι η είσοδος είναι συνδεδεμένη με μια αντίσταση στο GND και το παραπάνω script θα μας εμφανίζει τιμή ‘0’. Για να το ενεργοποιήσουμε, συνδέουμε ένα καλώδιο στο 1.8V, τοποθετούμε για λόγους προστασίας (για να μη δημιουργηθούν πολλά μA και κάψουν τον ακροδέκτη) μια αντίσταση περίπου 680Ωhm και συνδέουμε την άλλη άκρη προσωρινά στο κατάλληλο GPIO. Θα δούμε την τιμή από ‘0’ που ήταν να γίνει ‘1’.

Σε περίπτωση που μπορέσατε να δείτε την αλλαγή, έχει ολοκληρωθεί η βασική επικοινωνία με GPIO στο BBxM.

Προχωρημένα θέματα αναπτυξιακής πλακέτας:

- Χρήση Interrupt για είσοδο σε GPIO ώστε να μη γίνεται polling όπως τώρα.
- Χρήση αναλογικής εισόδου (Analog to Digital).
- Σύνδεση αισθητηρίου I2C
- Σύνδεση αισθητηρίου SPI