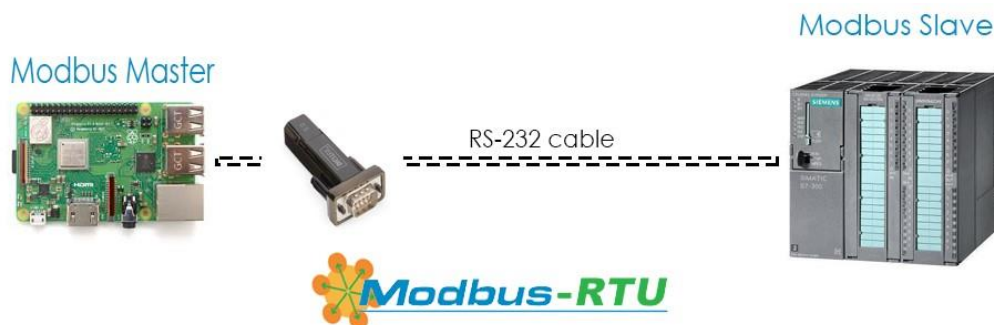


---

## Επικοινωνία PLC Siemens S7-300 και Raspberry Pi με τη χρήση Modbus RTU

---



Η επικοινωνία μεταξύ του PLC Siemens S7-300 και του Raspberry pi μπορεί να γίνει με 2 τρόπους . Είτε σειριακά , είτε μέσω TCP/IP . Τα πρωτόκολλα που εφαρμόζονται είναι πολυάριθμα με πιο γνωστά και διαδεδομένα το PROFIBUS , το PROFINET αλλά και το MODBUS .

### PROFIBUS

Το PROFIBUS ( Process Field Bus ) είναι ένα πρωτόκολλο για fieldbus επικοινωνία στο χώρο των αυτοματισμών . Πρωτοεμφανίστηκε το 1989 από την BMBF και μετά χρησιμοποιήθηκε από την Siemens όντας τμήμα του προτύπου IEC 61158 της Διεθνούς Ηλεκτροτεχνικής Επιτροπής .



Συγκεκριμένα αποτελεί ένα πρωτόκολλο για ντετερμινιστική επικοινωνία μεταξύ PROFIBUS Masters και I/O slaves . Υπάρχουν 2 είδη του πρωτοκόλλου PROFIBUS σήμερα , το *PROFIBUS DP* και το *PROFIBUS PA* με το πρώτο να αποτελεί το πιο σύνθητες στο χώρο της βιομηχανίας .

- **PROFIBUS DP** (Decentralised Peripherals) : Προτιμάται για υψηλές ταχύτητες και φθηνές συνδέσεις σταθμών. Αυτή η έκδοση του PROFIBUS σχεδιάστηκε κυρίως για επικοινωνία μεταξύ των συστημάτων αυτοματισμού και των περιφερειακών εισόδων/εξόδων (I/O) στο επίπεδο μηχανής (device level). Μπορεί να χρησιμοποιηθεί για να αντικαθιστά παράλληλες μεταδόσεις σημάτων με 24 V ή 0 έως 20 mA.
- **PROFIBUS PA** (Process Automation) : Σχεδιάστηκε ειδικά για την βελτίωση και την αντικατάσταση συμβατικών συστημάτων όπως τα συστήματα " 4-20 mA " και τα συστήματα που χρησιμοποιούν το πρωτόκολλο HART (Highway Addressable Remote Transducer Protocol).

### Τι είναι όμως το πρωτόκολλο HART :

Πρόκειται για ένα πρωτόκολλο μέσω του οποίου έχουμε τη δυνατότητα να στείλουμε ψηφιακά και αναλογικά δεδομένα ταυτόχρονα από το ίδιο ζεύγος καλωδίων . Πιο συγκεκριμένα



μπορούμε να στείλουμε τη μέτρηση των αναλογικών αισθητήρων παράλληλα με ψηφιακά δεδομένα όπως το Tagname, ρυθμίσεις βαθμονόμησης ή διαγνωστικά των αισθητήρων .

Αυτό μας οδηγεί στο συμπέρασμα ότι η χρήση του PROFIBUS PA δεν ταιριάζει στη διάταξή μας καθώς πρόκειται για μια διάταξη που χρησιμοποιεί αισθητήρες – διακόπτες (sensors switches) και όχι αισθητήρες – μετρητές (sensors meters).

### Πως λειτουργεί το PROFIBUS

Στο δίκτυο PROFIBUS υπάρχει διαχωρισμός σε κύριους (master) και εξαρτημένους (slave) σταθμούς. Οι κύριοι σταθμοί ορίζουν την επικοινωνία δεδομένων στο δίαυλο και μπορούν να στείλουν μηνύματα χωρίς την ανάγκη εξωτερικών αιτήσεων (request) όταν έχουν τα δικαιώματα πρόσβασης στο δίαυλο (κουπόνι) . Στο πρωτόκολλο PROFIBUS οι κύριοι σταθμοί καλούνται εναλλακτικά και ενεργοί σταθμοί (active stations). Κατά την επικοινωνία των κύριων σταθμών χρησιμοποιείται η διαδικασία ανταλλαγής κουπονιού. Οι εξαρτημένοι σταθμοί είναι περιφερειακές συσκευές οι οποίες περιλαμβάνουν PLCs, I/O συσκευές ,βαλβίδες οδηγούς και μεταδότες μέτρησης .

Η ταχύτητα μετάδοσης είναι επιλέξιμη μεταξύ 9.6 Kbits/sec και 12 Mbits/sec.

Όσον αφορά την εγκατάσταση και εφαρμογή ,όλες οι συσκευές με μέγιστο αριθμό 32,(κύριοι και εξαρτημένοι) συνδέονται σε μορφή διαύλου (γραμμής). Σε περίπτωση που έχουμε περισσότερους από 32 σταθμούς χρησιμοποιούμε επαναλήπτες για τη συνένωση τμημάτων του δικτύου . Το μέγιστο μήκος καλωδίου εξαρτάται από την ταχύτητα μετάδοσης .

Το πρωτόκολλο PROFIBUS έχει σχεδιαστεί για να εξασφαλίζει δύο πρωταρχικές απαιτήσεις του MAC :

- Κατά την διάρκεια μιας επικοινωνίας δύο σταθμών (master) πρέπει να διασφαλίζεται ότι κάθε ένας εξ αυτών θα έχει αρκετό χρόνο για να εκτελέσει το επικοινωνιακό του έργο εντός ενός ακριβώς καθορισμένου χρονικού διαστήματος .
- Η κυκλική και σε πραγματικό χρόνο μετάδοση δεδομένων πρέπει να υλοποιείται όσο το δυνατόν πιο απλά και πιο γρήγορα κατά την επικοινωνία ενός σύνθετου προγραμματιζόμενου λογικού ελεγκτή (PLC) με τις I/O συσκευές (slaves).

Για αυτόν τον λόγο το πρωτόκολλο PROFIBUS περιλαμβάνει τη διαδικασία μεταγωγής κουπονιού το οποίο χρησιμοποιείται από κύριους σταθμούς (master) για την μεταξύ τους επικοινωνία και τη διαδικασία κυρίου-εξαρτημένου (master-slave) η οποία χρησιμοποιείται από κύριους σταθμούς για επικοινωνία με τις απλές I/O συσκευές . Η διαδικασία περάσματος κουπονιού εγγυάται ότι το δικαίωμα πρόσβασης στο δίαυλο (κουπόνι) παρέχεται σε κάθε ένα σταθμό (master) για προκαθορισμένη χρονική διάρκεια . Το κουπόνι, ένα ειδικό πλαίσιο δεδομένων για παροχή δικαιωμάτων πρόσβασης ,περιφέρεται περιμετρικά του λογικού δακτυλίου με μια προκαθορισμένη μέγιστη ταχύτητα περιστροφής ( και προκαθορισμένη από πριν σειρά)

Το PROFIBUS σχεδιάστηκε για επικοινωνία σε υψηλές ταχύτητες στο χαμηλότερο επίπεδο ,το επίπεδο μηχανής (device level).Στο επίπεδο αυτό η επικοινωνία μεταξύ κεντρικών ελεγκτών (π.χ PLCs/PCs) και καταναμημένων συσκευών πεδίου (I/O,οδηγών, βαλβίδων κ.λ.π) γίνεται μέσω μιας σειριακής σύνδεσης υψηλής ταχύτητας . Το

μεγαλύτερο μέρος της επικοινωνίας με αυτές τις συσκευές γίνεται με την κυκλική μέθοδο. Πέρα από την εκτέλεση των κυκλικών διαδικασιών, απαιτούνται και λειτουργίες ασύγχρονης επικοινωνίας για έξυπνες συσκευές πεδίου ώστε να επιτρέπεται η διαμόρφωση, ο έλεγχος και η διάγνωση λαθών όπως και ο χειρισμός σημάτων συναγερμού .

## **PROFINET IO**

Το PROFINET IO είναι πολύ παρόμοιο με το PROFIBUS . Χρησιμοποιεί τρία διαφορετικά κανάλια επικοινωνίας για την ανταλλαγή δεδομένων μεταξύ Προγραμματιζόμενων Λογικών Ελεγκτών ( PLCs) και άλλων συσκευών . Το κανάλι TCP/IP χρησιμοποιείται για παραμετροποίηση , ρύθμιση και μη κυκλική ανάγνωση και εγγραφή δεδομένων. Το RT ή Real Time κανάλι χρησιμοποιείται για κυκλική μεταφορά δεδομένων και χειρισμό σημάτων συναγερμού . Οι RT επικοινωνίες παρακάμπτουν το πρωτόκολλο TCP/IP για να επισπεύσουν την ανταλλαγή δεδομένων μεταξύ των Ελεγκτών. Το τρίτο κανάλι με ονομασία *Isochronous Real Time (IRT)* είναι ένα κανάλι υψηλών ταχυτήτων που χρησιμοποιείται για εφαρμογές Ελέγχου Κίνησης ( Motion Control ) .



Παραδοσιακά , το Ethernet είχε μικρή αποδοχή στο χώρο του Βιομηχανικού Αυτοματισμού . Μέχρι πρόσφατα , το κόστος , η έλλειψη εξελιγμένων switches και routers και η κυριαρχία μεγάλων προμηθευτών με ιδιόκτητα πρωτόκολλα , εμπόδισαν την ευρεία αποδοχή του Ethernet στη βιομηχανία . Τώρα με την πτώση των τιμών , την ύπαρξη υπολογιστών με ενσωματωμένες δυνατότητες Ethernet, έξυπνων Switches και Routers, το Ethernet κερδίζει την αποδοχή της βιομηχανικής κοινότητας . Μόνο η έλλειψη ενός ευρέως αποδεκτού , ευέλικτου application layer που στοχεύει στο Βιομηχανικό Αυτοματισμό εμπόδισε την πλήρη αποδοχή του .

## **Πλεονεκτήματα**

Το PROFINET είναι ένα ξεχωριστό “industrial Ethernet application layer” όπου προσφέρει πολλά πλεονεκτήματα σε σχέση με τα ανταγωνιστικά application layers :

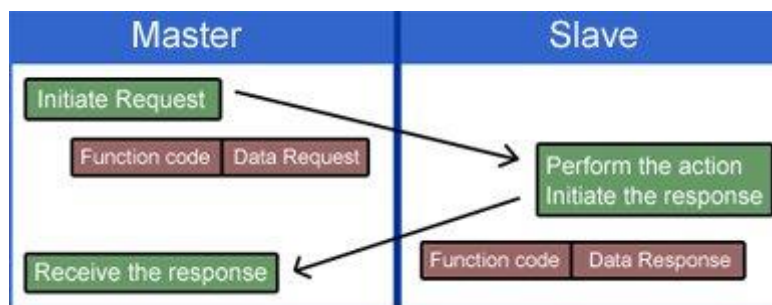
- Υψηλές ταχύτητες λειτουργίας (100Mbps)
- Πανομοιότυπος τρόπος εγκατάστασης με το πρωτόκολλο PROFIBUS
- Υποστήριξη εφαρμογών Ελέγχου Κίνησης κρίσιμου χρόνου
- Ευκολία εγκατάστασης
- Ελάχιστος χρόνος ενεργοποίησης και τεχνική υποστήριξη

## **Διαφορές μεταξύ PROFIBUS και PROFINET**

Επειδή και τα δύο πρωτόκολλα έχουν δημιουργηθεί και στηρίζονται από τον ίδιο οργανισμό, υπάρχουν ομοιότητες όπως η χρήση GSD αρχείων για τον προσδιορισμό του hardware κάθε συσκευής. Τα GSD αρχεία για τον PROFIBUS είναι ASCII αρχεία ενώ για το PROFINET είναι XML αρχεία . Πέρα από τη διαφορά της ταχύτητας , το PROFIBUS μπορεί να μεταδώσει δεδομένα ασύρματα μέσω κατάλληλων συσκευών ενώ το PROFINET μπορεί να χρησιμοποιήσει εύκολα είτε Wi-Fi είτε Bluetooth . Τέλος το πρωτόκολλο PROFINET παρέχει πολύ μεγαλύτερη ασφάλεια σε σχέση με το PROFIBUS . Όμως όλα αυτά τα πλεονεκτήματα του PROFINET έχουν αντίκτυπο στην τιμή καθώς το κόστος μιας κάρτας επικοινωνίας PROFINET είναι πολύ μεγαλύτερο σε σχέση με το κόστος μιας κάρτας PROFIBUS .

## MODBUS RTU

Το πρωτόκολλο MODBUS RTU είναι ένα ανοικτό , σειριακό ( RS-232 ή RS-485 ) πρωτόκολλο που εφαρμόζει την αρχιτεκτονική Κύριου(Master)/Εξαρτημένου(Slave). Είναι ένα ευρέως αποδεκτό πρωτόκολλο λόγω της ευκολίας χρήσης του και της αξιοπιστίας του .Οι εξαρτημένοι δεν μεταδίδουν ποτέ δεδομένα χωρίς να λαμβάνουν αίτημα (request) από τον Κύριο . Ο Κύριος κόμβος μπορεί να εκκινεί ταυτόχρονα μόνο ένα κανάλι MODBUS .



Υπάρχουν 4 τύποι καταχωρητών που χρησιμοποιούνται στο MODBUS :

- **Coils ( Λογικά πηνία )** : Ξεκινάνε από τη διεύθυνση 0001 και αφορούν θέσεις μνήμης ενός bit ( είτε 0 είτε 1 ) π.χ. βοηθητικά ρελέ
- **Input Bits** : Ξεκινάνε από τη διεύθυνση 1001 και αφορούν της εισόδους ( Μόνο ανάγνωση )
- **Input Registers** : Ξεκινάνε από τη διεύθυνση 3001 και αφορούν τις αναλογικές εισόδους ( αισθητήρες )
- **Holding Registers** : Ξεκινάνε από τη διεύθυνση 4001 και αφορούν αναλογικές παραμέτρους οι οποίες μπορούν να αλλάξουν

Για την εφαρμογή οποιοδήποτε πρωτοκόλλου είναι απαραίτητη η χρήση (συγκεκριμένης για κάθε πρωτόκολλο) μονάδας επικοινωνίας . Η μόνη μονάδα επικοινωνίας που είχαμε στη διάθεση μας ήταν η **CP-340 (RS 232C / 340-1AH02-0AE0 )** .



Εικόνα 1 - Siemens CP340

Οπότε , το μόνο πρωτόκολλο που μπορούσαμε να χρησιμοποιήσουμε είναι το MODBUS RTU .

Ο **Tibor Hajnal** έχει δημιουργήσει πακέτο λογισμικού που υλοποιεί το πρωτόκολλο GOULD MODBUS RTU ( τόσο για τη πλευρά του Master όσο και για τη πλευρά του Slave ) στο PLC SIMATIC S7-300/400 με μονάδα επικοινωνίας την CP340 .Στη συγκεκριμένη περίπτωση το PLC λειτουργεί ως Modbus Slave .

### **MBSL340 Software Package**

Σε αυτή τη παράγραφο θα αναφέρουμε συνοπτικά το τρόπο λειτουργίας του πακέτου . Αναλυτική περιγραφή του βρίσκεται στο [forum της Siemens](#) στο manual **MBSL340.pdf** που έχει γράψει ο Tibor Hajnal .

Ο software driver υποστηρίζει τα παρακάτω function codes :

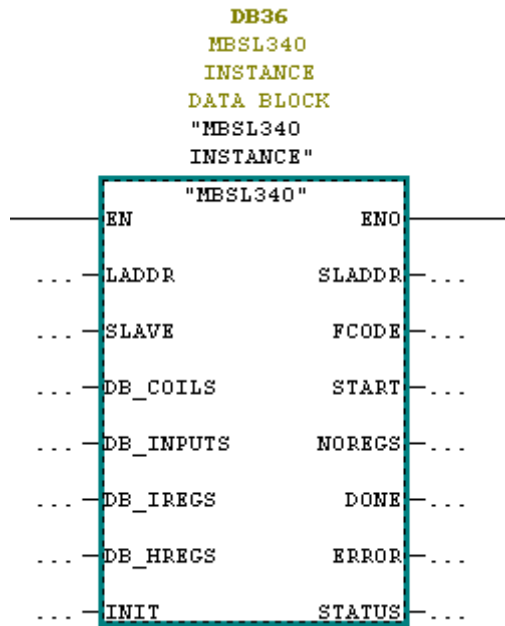
- Function Code 01 – Read Coil Status
- Function Code 02 – Read Input Status
- Function Code 03 – Read Holding Registers
- Function Code 04 – Read Input Registers
- Function Code 05 – Force Single Coil
- Function Code 06 – Preset Single Register
- Function Code 15 – Force Multiple Coils
- Function Code 16 – Preset Multiple Registers

Αποτελεί ένα project ( MBSL340.ZIP) το οποίο ανοίγει με τη χρήση του SIMATIC Manager . Το project αποτελείται συνολικά από 3 Function Blocks

- FB36 “MBSL340”            Modbus Slave Communication function block
- FB3 “P\_SEND”            Siemens standard send function block for CP340
- FC2 “P\_RCV”            Siemens standard receive function block for CP340

### **FB36 “MBSL340” function block**

Είναι το μοναδικό function block που καλείται από το πρόγραμμα του χρήστη . Τα άλλα function blocks “**P\_SEND**” και “**P\_RCV**” για τη μονάδα CP340 καλούνται εντός του “**MBSL340**” function block . Οι ξεχωριστές περιοχές των MODBUS διευθύνσεων καθορίζονται ορίζοντας αντίστοιχα Data Block για κάθε περιοχή διευθύνσεων



Οι παράμετροι εισόδου του Function Block “MBSL340” είναι :

Όνομα	Τύπος	Default	Περιγραφή
LADDR	INT	256	Ορίζει την I/O διεύθυνση του communication Processor (CP) . Αυτή η διεύθυνση πρέπει να ταιριάζει με τη διεύθυνση συ υλικού που έχει δηλωθεί ( HWConfig )
SLAVE	INT	1	Η διεύθυνση του Modbus Slave . Πιθανές τιμές είναι από 1 έως 255
DB_COILS	BLOCK_DB		Data block για τα coils
DB_INPUTS	BLOCK_DB		Data block για τα inputs
DB_IREGS	BLOCK_DB		Data block για τους input registers
DB_HREGS	BLOCK_DB		Data block για τους holding registers

Οι παράμετροι εξόδου είναι :

Όνομα	Τύπος	Default	Περιγραφή
SLADDR	INT		Παρακολούθηση της εξόδου για τη ληφθείσα διεύθυνση του Slave
FCODE	INT		Παρακολούθηση της εξόδου για το ληφθέν function code
START	INT		Παρακολούθηση της εξόδου για τη ληφθείσα start address
NOREGS	INT		Παρακολούθηση της εξόδου για το ληφθέν αριθμό των διευθύνσεων των καταχωρητών
DONE	BOOL		Η ληφθείσα function εκτελέστηκε με επιτυχία χωρίς κανένα σφάλμα
ERROR	BOOL		Ένα σφάλμα συνέβη κατά τη διάρκεια της λειψήσας function
STATUS	WORD		Λεπτομερή πληροφόρηση για την κατάσταση του σφάλματος

Οι παράμετροι εισόδου/εξόδου είναι :

Όνομα	Τύπος	Default	Περιγραφή
INIT	BOOL		Αίτημα αρχικοποίησης. Απαιτείται για τον εκ νέου υπολογισμό των περιοχών διεύθυνσης σε περίπτωση αλλαγής μήκους μπλοκ δεδομένων κατά τη διάρκεια εκτέλεσης

## **Ορισμό PLC ως MODBUS RTU Slave**

Με χρήση της παραπάνω βιβλιοθήκης και του SIMATIC Manager θα ορίσουμε το PLC ως MODBUS Slave . Το υλικό και το λογισμικό που χρησιμοποιήθηκε είναι :

### Hardware:

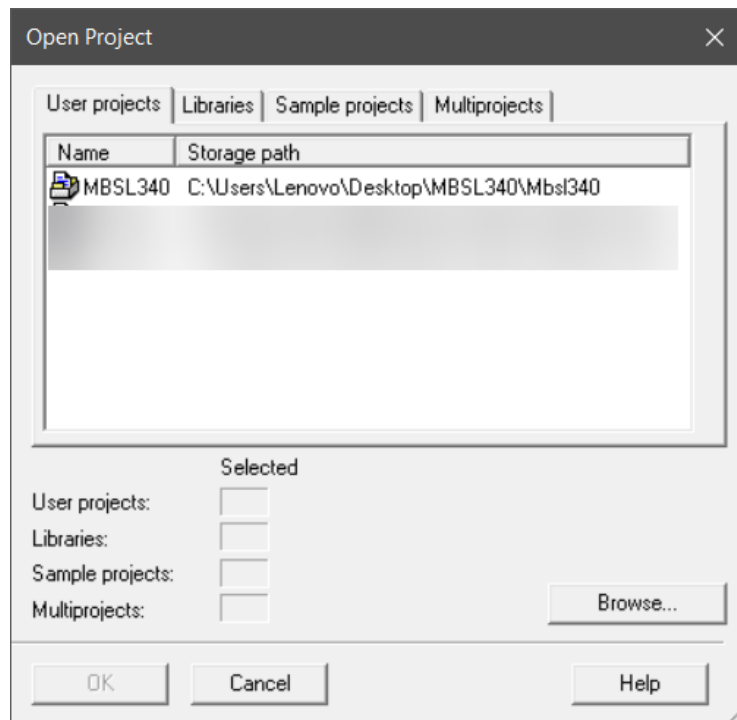
1. Τροφοδοτικό DC POWER SUPPLY HY3005D ( Ρυθμισμένο στα 24 V )
2. Siemens PLC SIMATIC S7-300
  - 2.1. CPU 312C (312-5BD01-0AB0)
  - 2.2. DI 16XDC24V (321-1BH02-0AA0)
  - 2.3. DI8/DO8XDC24V (323-1BH01-0AA0)
  - 2.4. CP340 RS232C (340-1AH02-0AE0)
3. S7-300 PC Adapter

### Software:

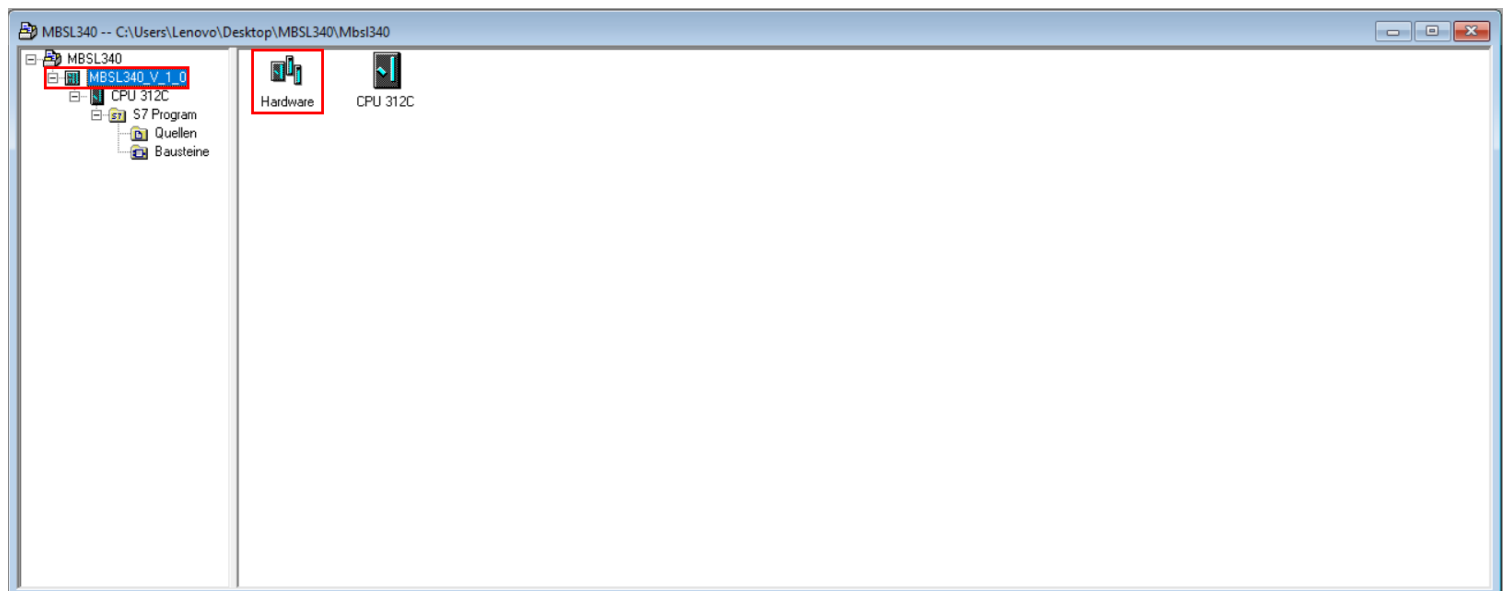
1. STEP7 V5.6 (Simatic Manager)
2. MDBUS ( Modbus Simulator Software )

## **STEP7 HW-Config**

Αφ' ότου έχουμε εγκαταστήσει τη μονάδα CP340 στη βάση στήριξης και ανάπτυξης του υλικού , ανοίγουμε το Simatic Manager . Στη συνέχεια πατάμε File > Open και ανοίγουμε το project MBSL340.s7p που κατεβάσαμε

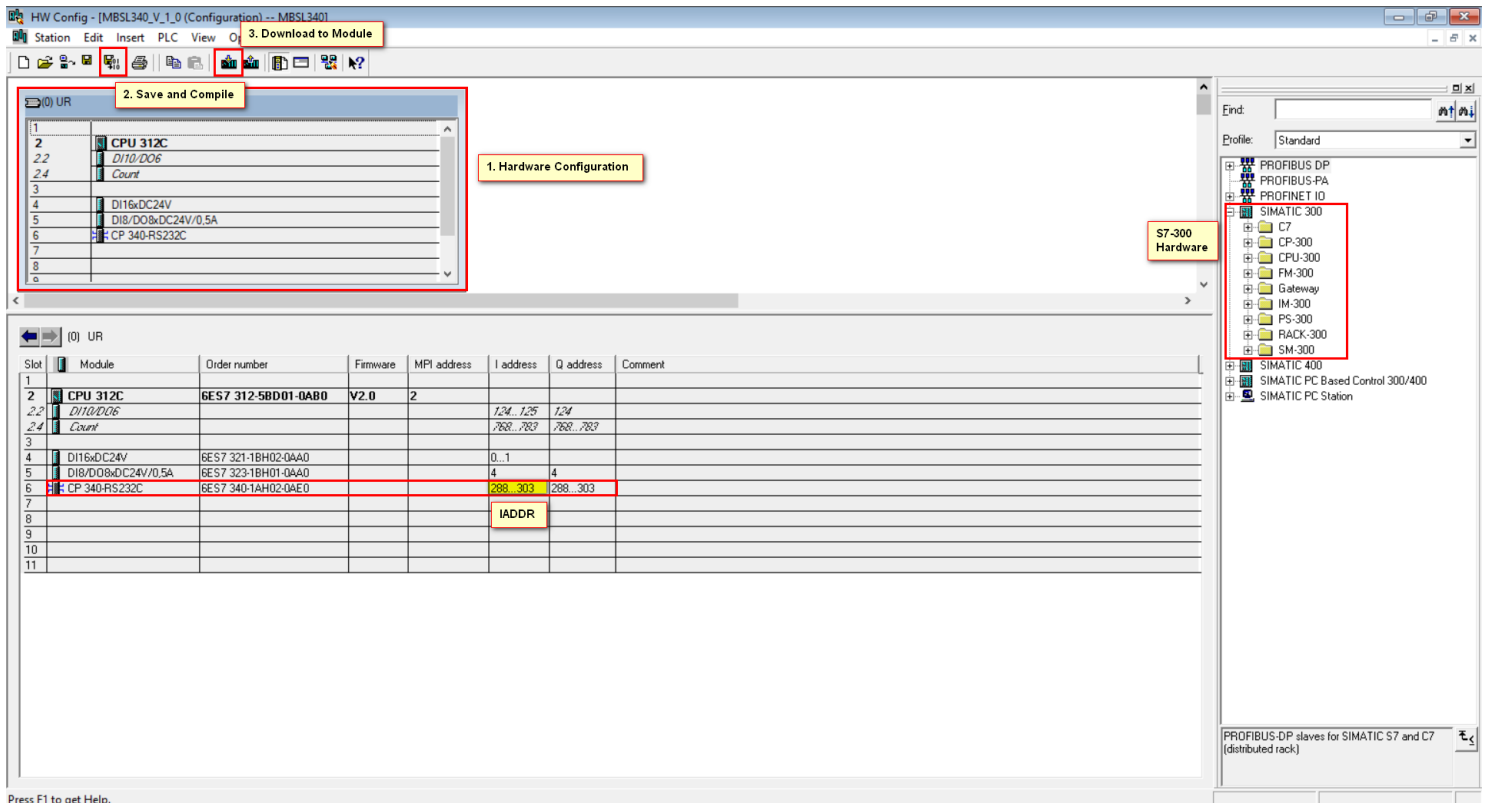


Ανοίγοντας το project εμφανίζεται ένα παράθυρο .Αρχικά , το δηλωμένο υλικό του project είναι το PLC S7-300 με κεντρική μονάδα επεξεργασίας την CPU 315-DP . Επομένως θα πρέπει να δηλώσουμε το υλικό σύμφωνα με το δικό μας PLC . Αυτό γίνεται πατώντας **MBSL340\_V\_1\_0** και στη συνέχεια το εικονίδιο **Hardware** .



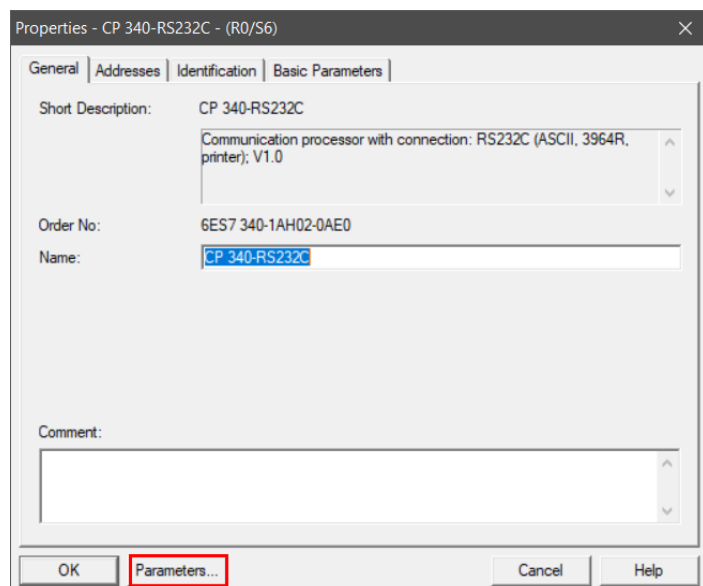


Έτσι οδηγούμαστε σε ένα νέο παράθυρο στο οποίο θα γίνει η δήλωση του υλικού . Επειδή όπως προαναφέρθηκε το project έχει εξ' αρχής δηλωμένο υλικό , αρκεί εμείς να το αντικαταστήσουμε με το δικό μας υλικό . Αυτό γίνεται πατώντας σε κάθε βαθμίδα του υλικού δεξί κλικ και στη συνέχεια **Replace Object** .

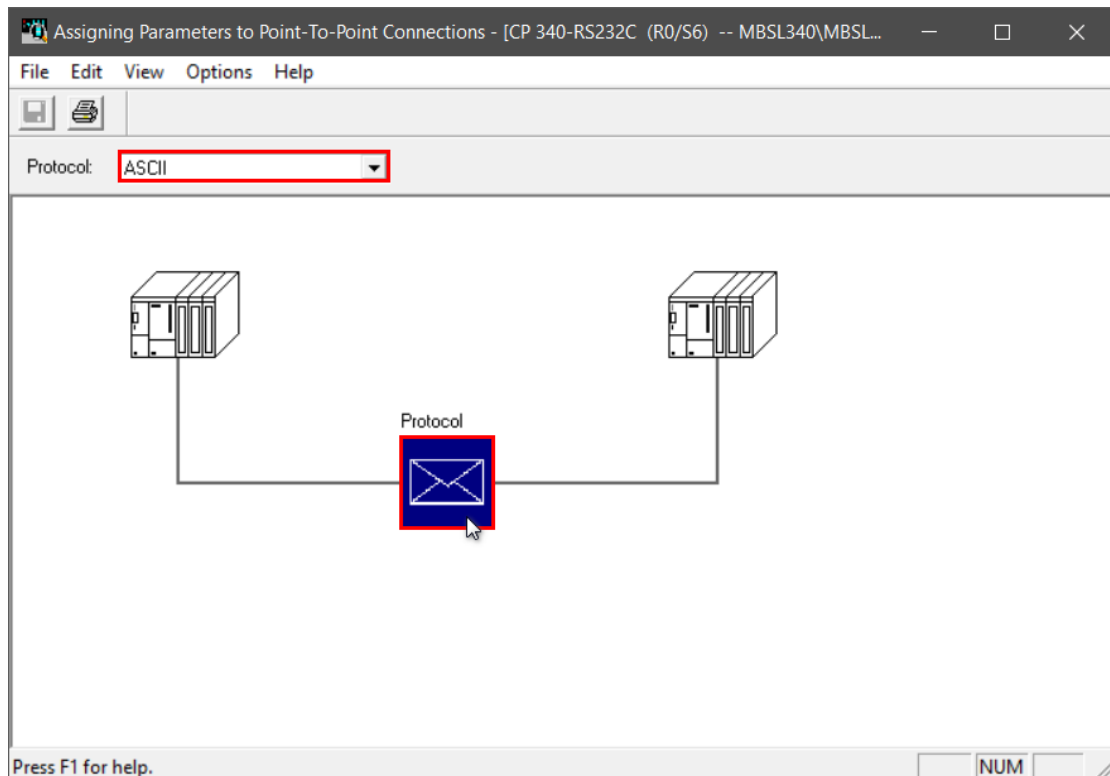



**Σημείωση :** Η **I address** της μονάδας επικοινωνίας θα ζητηθεί ως παράμετρος εισόδου LADDR στο Function Block FB36 “MBSL340” Modbus Slave Communication function block.

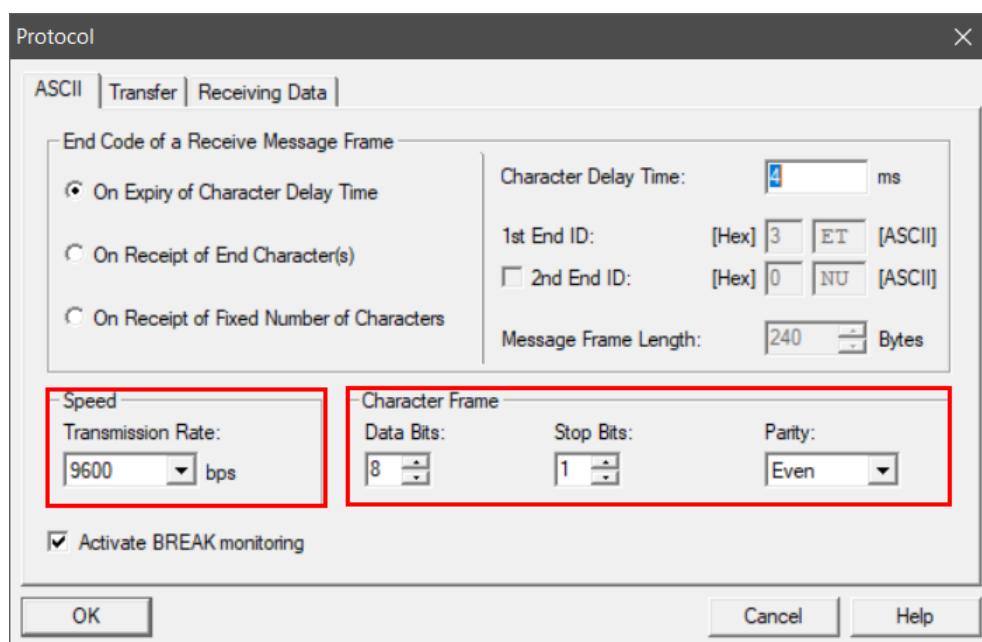
Στη συνέχεια, έχοντας επιλέξει τη μονάδα επικοινωνίας , πατάμε δεξί κλικ και έπειτα **Object Properties** . Έγτερα , αφ' ότου εμφανιστεί το παράθυρο ιδιοτήτων της βαθμίδας επικοινωνίας , πατάμε το κουμπί **Parameters** .



Στις παραμέτρους ,ως πρωτόκολλο πρέπει να επιλέξουμε το πρωτόκολλο ASCII προκειμένου να προσαρμόσει το πρωτόκολλο MODBUS RTU .



Στη συνέχεια πατώντας στο εικονίδιο  ρυθμίζουμε τις παραμέτρους του πρωτοκόλλου ( Baud Rate , Parity , Data Bits,Stop Bits ) . Οι πρώτες δύο παράμετροι πρέπει να ταιριάζουν με τις παραμέτρους του Master ( στη προκειμένη περίπτωση **Baud Rate : 9600** και **Parity : Even** ) .Τα **Data Bits** πρέπει να είναι **8** και τα **Stop Bits= 1** .

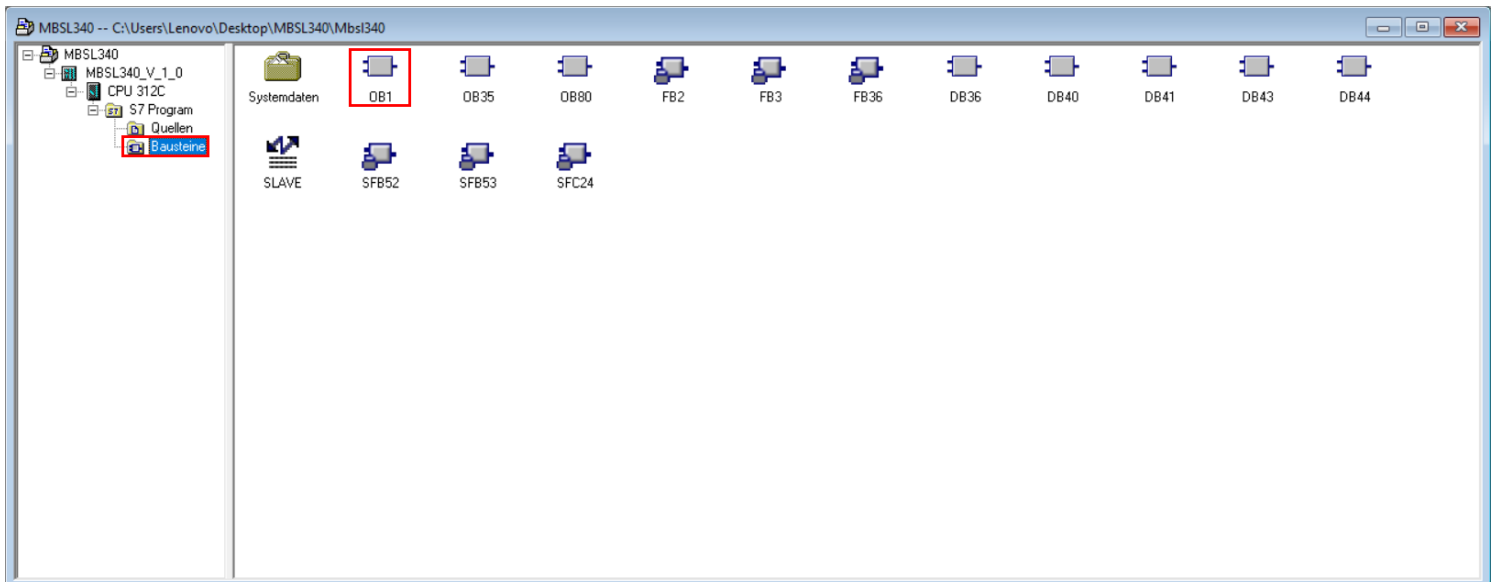


Έπειτα πατάμε **OK** και **Save** . Αφ' ότου έχουμε αντικαταστήσει το υπάρχον hardware με το δικό μας και έχουμε παραμετροποιήσει κατάλληλα τη μονάδα επικοινωνίας , πατάμε **Save and Compile** και στη συνέχεια **Download to Module** .

Θα παρατηρήσουμε ότι πλέον ο φάκελος S7 program του project δεν βρίσκεται εντός της CPU αλλά εκτός . Αυτό που έχουμε να κάνουμε είναι να επιλέξουμε όλο το φάκελο, να πατήσουμε δεξί κλικ και **Cut** και στην συνέχεια να κάνουμε επικόλληση ( **Paste**) στη CPU . Έτσι πλέον θα παρατηρήσουμε ότι ο φάκελος S7 Program βρίσκεται εντός της CPU όπως φαίνεται στην παρακάτω εικόνα



Αυτό που μας απομένει είναι να εισάγουμε τις δικές μας παραμέτρους στο πρόγραμμα ( *αλλάζοντας μόνο τις παραμέτρους LADDR και SLAVE* ) . Για να το κάνουμε αυτό επιλέγουμε το φάκελο **Bausteine** και στη συνέχεια ανοίγουμε το Program Block **OB1** .



Ανοίγοντας το Program Block OB1 θα παρατηρήσουμε ένα πρόγραμμα γραμμένο σε *Structured Text* ( γλώσσα του προτύπου IEC 61131-3 ) . Όπως αναφέρθηκε και προηγουμένως , οι μόνες παράμετροι που καθορίζουμε εμείς είναι οι παράμετροι **LADDR** και **SLAVE** . Στη προκειμένη περίπτωση η I address της μονάδας επικοινωνίας μας είναι **288** ενώ το Unit Id του Slave που έχουμε επιλέξει είναι **1** .

Network 1: MODBUS RTU SLAVE COMMUNICATION FUNCTION BLOCK

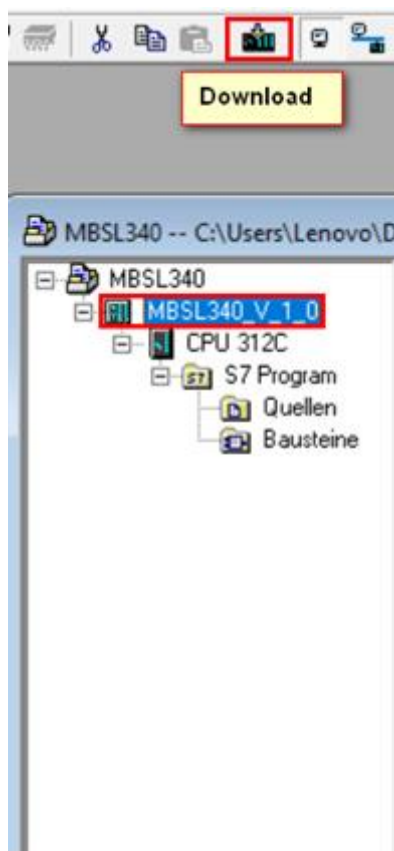
```
CALL "MBSL340", "MBSL340 INSTANCE"    FB36 / DB36    -- MODBUS RTU SLAVE COMMUNICATION FUNCTION BLOCK / MBSL340 INSTANCE DATA BLOCK
LADDR    :=288
SLAVE    :=1
DB_COILS :="COILS"                      DB40          -- MODBUS ADDRESS AREA 0
DB_INPUTS:= "INPUTS"                   DB41          -- MODBUS ADDRESS AREA 1
DB_IREGS := "INPUT REGISTERS"          DB43          -- MODBUS ADDRESS AREA 3
DB_HREGS := "HOLDING REGISTERS"        DB44          -- MODBUS ADDRESS AREA 4
SLADDR   := "SLADDR"                  MW100         -- MONITOR - RECEIVED SLAVE ADDRESS FROM MASTER
FCODE    := "FCODE"                   MW102         -- MONITOR - RECEIVED FUNCTION CODE FROM MASTER
START    := "START"                   MW104         -- MONITOR - RECEIVED START ADDRESS FROM MASTER
NOREGS   := "NOREGS"                  MW106         -- MONITOR - RECEIVED NUMBER OF REGISTER FROM MASTER
DONE     := "DONE_340"                M108.0        -- MBSL340 - DONE, WITHOUT FAILURE
ERROR    := "ERROR_340"               M108.1        -- MBSL340 - DONE WITH FAILURE, SEE STATUS WORD
STATUS   := "STATUS_340"              MW110         -- MBSL340 - STATUS WORD
INIT     := "INIT_340"                M112.0        -- MBSL340 - INITIALISATION REQUEST

A        "DONE_340"                   M108.0        -- MBSL340 - DONE, WITHOUT FAILURE
CU        C        1
A        "ERROR_340"                 M108.1        -- MBSL340 - DONE WITH FAILURE, SEE STATUS WORD
CU        C        2

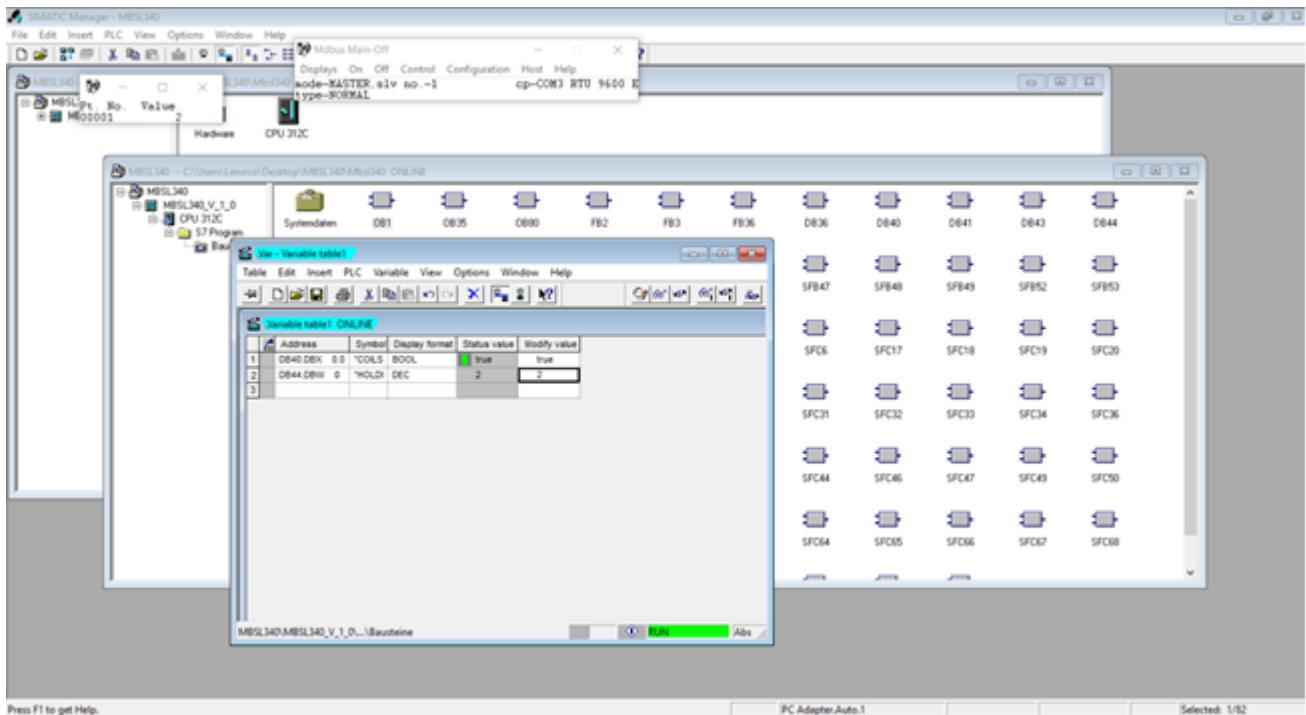
AN        "ERROR_340"                 M108.1        -- MBSL340 - DONE WITH FAILURE, SEE STATUS WORD
JC        M000
L        "STATUS_340"                 MW110         -- MBSL340 - STATUS WORD
T        MW        114
```

M000: BEU

Μετά την αλλαγή των 2 αυτών παραμέτρων και την Αποθήκευση του προγράμματος , το μόνο που μας μένει είναι να φορτώσουμε το πρόγραμμα στο PLC . Αυτό γίνεται πατώντας το **MBSL340\_V\_1\_0** και μετά **Download** .



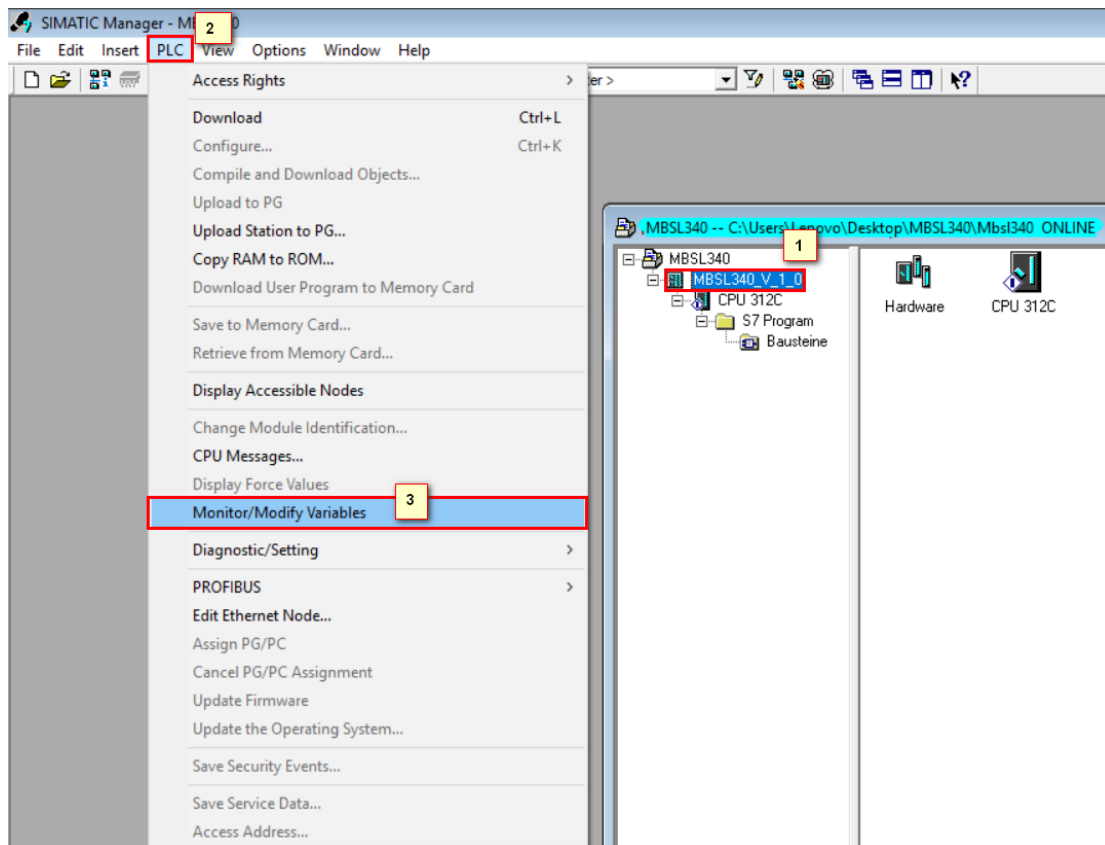
Τη στιγμή που τελειώνει το κατέβασμα του προγράμματος στο PLC , το Siemens S7-300 λειτουργεί πλέον ως Modbus RTU Slave . Για να δούμε τη σωστή λειτουργία του PLC ως MODBUS RTU Slave χρησιμοποιήσαμε **MODBUS Simulator** . Ο υπολογιστής ρυθμίστηκε ως MODBUS Master ,ο οποίος ζητούσε την ανάγνωση ενός Holding Register.



Για την αλλαγή των τιμών των 4 τύπων καταχωρητών ( Coils , Inputs , Input Registers , Holding Registers ) αρχικά πατάμε το κουμπί **Online** .



Πατώντας το , εμφανίζεται ένα νέο παράθυρο το οποίο μας δείχνει τα αρχεία του PLC .Στη συνέχεια επιλέγουμε το **MBSL340\_V\_1\_0** και στο Menu επιλέγουμε με τη σειρά **PLC > Monitor Modify Variables** .



Έτσι ανοίγει το παράθυρο και ανάλογα τον καταχωρητή που θέλουμε να μεταβάλλουμε την τιμή του πληκτρολογούμε την ανάλογη διεύθυνση .Στη συνέχεια πατάμε **Monitor Variable** και αφ' ότου κάνουμε την αλλαγή στη τιμή πατάμε **Activate modify values** .



Τώρα το μόνο που μας μένει είναι να ορίσουμε το Raspberry Pi ως Modbus Master

## Raspberry Pi as Modbus RTU Master

Για να ορίσουμε το Raspberry Pi ως MODBUS Master θα χρησιμοποιήσουμε το λογισμικό **CODESYS**.

Το CODESYS (ακρωνύμιο του Controller Development System) είναι ένα περιβάλλον ανάπτυξης εφαρμογών για ελεγκτές σύμφωνα με το πρότυπο IEC 61131-3.



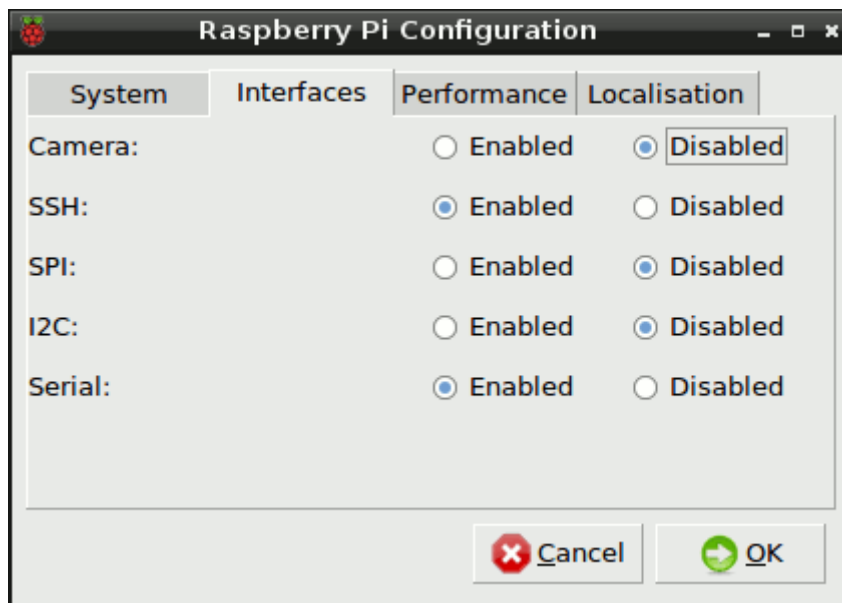
Ενσωματωμένοι μεταγλωττιστές (Integrated Compilers) μετατρέπουν το πρόγραμμα του χρήστη σε γλώσσα μηχανής η οποία μεταφορτώνεται στον εκάστοτε ελεγκτή. Οι πιο σημαντικές 16 και 32-bit οικογένειες επεξεργαστών που υποστηρίζονται είναι οι C166, TriCore, 80x86, ARM/Cortex, Power Architecture, SH, MIPS, BlackFin.

### Διαδικασία

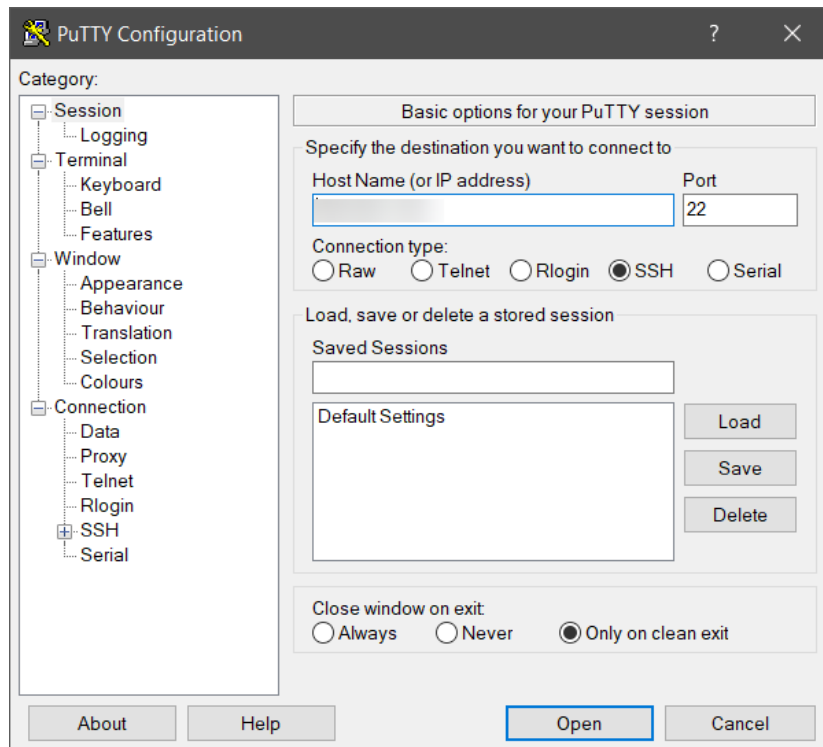
Για να ορίσουμε το Raspberry Pi ως Modbus Master χρειαζόμαστε :

- CODESYS Development System Version 3.5.13.20
- Raspbian Stretch Full / Release date: 2018-11-13 / Kernel version: 4.14
- USB to RS-232 Adapter

Αρχικά, αφ' ότου εγκαταστήσουμε το Raspbian στο Raspberry ενεργοποιούμε το πρωτόκολλο **SSH** πατώντας το **Raspberry Pi Configuration** από το **Menu**.



Τώρα μπορούμε να επικοινωνήσουμε με το Raspberry μέσω του **PuTTY**.



Αφ' ότου ξεκινήσουμε την επικοινωνία με το Raspberry πρέπει να ελέγξουμε τη σωστή μετάδοση δεδομένων μεταξύ του Raspberry Pi και του PLC . Αυτό γίνεται πληκτρολογώντας τις παρακάτω εντολές :

1. `cd /dev/`
2. `ls`

Εκτελώντας την εντολή ls θα παρατηρήσουμε στο τέλος ότι το ttyUSB0 ανιχνεύτηκε εντός του /dev/

```

pi@raspberrypi:~$ cd /dev
pi@raspberrypi:/dev$ ls
autofs          ram5            tty19           tty44           uhid
bluetooth       loop6          ram6            tty2            tty45           uinput
brfs-control    loop-control   ram7            tty20           tty46           urandom
bus             mapper         ram8            tty21           tty47           vc-cma
cachefiles      mem            ram9            tty22           tty48           vchiq
char            memory_bandwidth random          tty23           tty49           vcio
char_dev        mmchblk0      raw             tty24           tty5           vc-mem
console         mmchblk0p1    rxfill          tty25           tty50           vcs
cpu_dma_latency mmchblk0p2    serial          tty26           tty51           vcs1
cuse            nvme           sda             tty27           tty52           vcs2
disk            net            snd             tty28           tty53           vcs3
fd              network_latency spidev0.0       tty29           tty54           vcs4
fd              network_throughput spidev0.1       tty3            tty55           vcs5
full            null           stderr          tty30           tty56           vcs6
fuse            ppp           stdin           tty31           tty57           vcsa
gpiomem         ptmx          stdout          tty32           tty58           vcsa1
hwrng           rtc           tty             tty33           tty59           vcsa2
i2c-1           ram0          tty0            tty34           tty6           vcsa3
initctl         ram1          tty1            tty35           tty60           vcsa4
input           ram10         tty10           tty36           tty61           vcsa5
kmsg            ram11         tty11           tty37           tty62           vcsa6
log             ram12         tty12           tty38           tty63           vcam
loop0           ram13         tty13           tty39           tty7           vchi
loop1           ram14         tty14           tty4            tty8           xconconsole
loop2           ram15         tty15           tty10           tty9           zero
loop3           ram2          tty16           tty41           ttyAMA0
loop4           ram3          tty17           tty42           ttyprintk
loop5           ram4          tty18           tty43           ttyUSB0

```



Πληκτρολογώντας την εντολή **dmseg | grep tty** επιστρέφει “usb 1-1.3: FTDI USB Serial Device converter now attached to ttyUSB0”

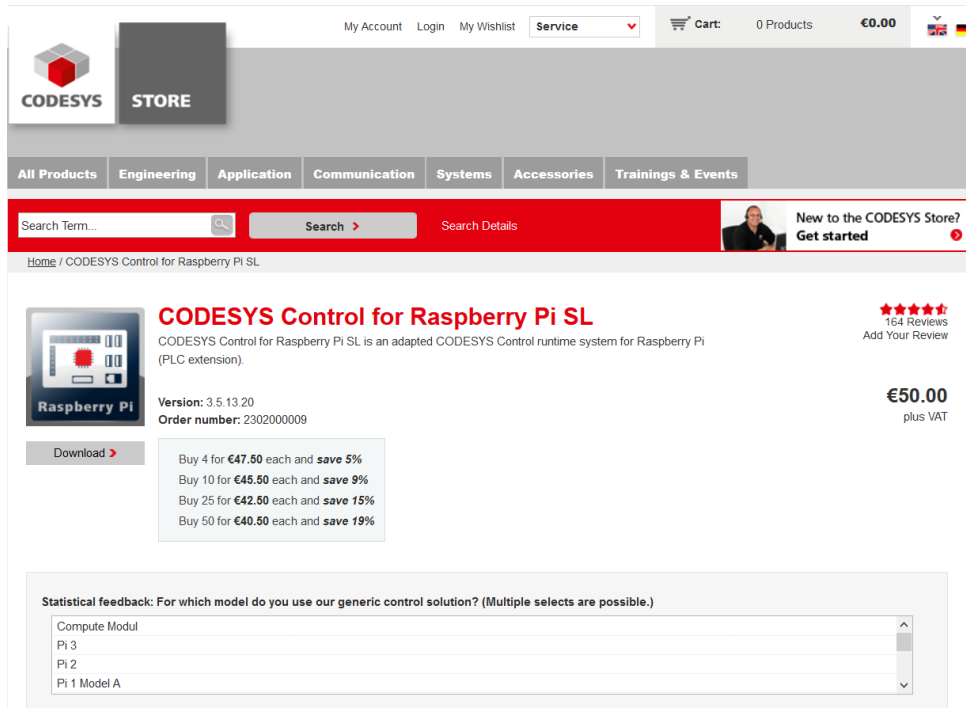
```
pi@raspberrypi: /dev
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 26 16:40:32 2018
pi@raspberrypi:~$ dmesg | grep tty
-bash: dmesg: command not found
pi@raspberrypi:~$ cd /dev/
pi@raspberrypi:/dev$ dmesg |grep tty
[ 0.000000] Kernel command line: 8250.nr_ua... bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2708_fb.fbswap=1 vc_mem.mem_base=0x3ec00000 vc_mem.mem_size=0x40000000 dwc_otg.lpm_enable=0 console=ttyS0,115200 console=tty1 root=PARTUUID=06d3fed3-02 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
[ 0.000282] console [tty1] enabled
[ 0.664652] 3f201000.serial: ttyAMA0 at MMIO 0x3f201000 (irq = 87, base_baud = 0) is a PL011 rev2
[ 4.176151] usb 1-1.3: FTDI USB Serial Device converter now attached to ttyUSB0
pi@raspberrypi:/dev$ usb 1-1.3: FTDI USB Serial Device converter now attached to ttyUSB0^C
pi@raspberrypi:/dev$
```

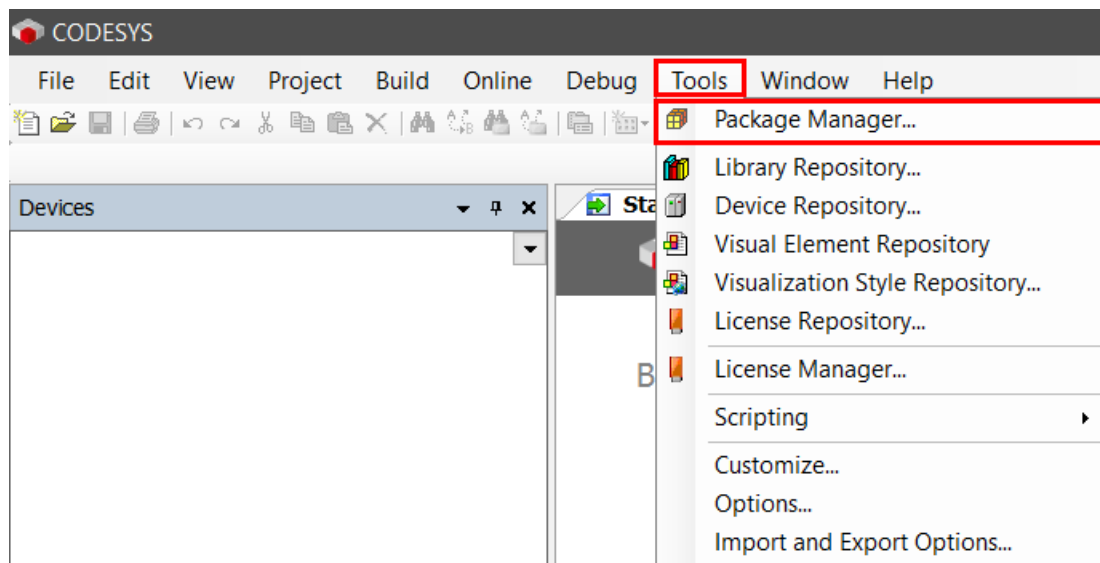
Τώρα , μπορούμε να χρησιμοποιήσουμε το CODESYS . Αρχικά πρέπει να κατεβάσουμε το CODESYS Development System από το **Codesys Store**.

The screenshot shows the CODESYS Store website. At the top, there's a navigation bar with links for 'My Account', 'Log Out', 'My Wishlist', a 'Service' dropdown, a shopping cart icon showing '0 Products' for '€0.00', and a language selector. Below this is a category bar with 'All Products', 'Engineering', 'Application', 'Communication', 'Systems', 'Accessories', and 'Trainings & Events'. A search bar is present with a 'Search' button and a 'Search Details' link. A promotional banner for 'New to the CODESYS Store? Get started' is also visible. The main product listing for 'CODESYS Development System V3' includes a product image, a 5-star rating with '2 Reviews', and a price of '€0.00 plus VAT'. There are buttons for 'Download 32 Bit' and 'Download 64 Bit', as well as links for 'Add to My Wishlist' and 'Recommend Product'. Below the product listing, there's a 'Product Description' tab selected, showing details about the system's capabilities and a list of features: Project tree for structuring project configuration, Configurator for integrating and describing various devices and fieldbus systems, Editors for typical application development in all graphical and text-based implementation languages defined by IEC 61131-3, and Compilers for building applications in lean and powerful machine code.

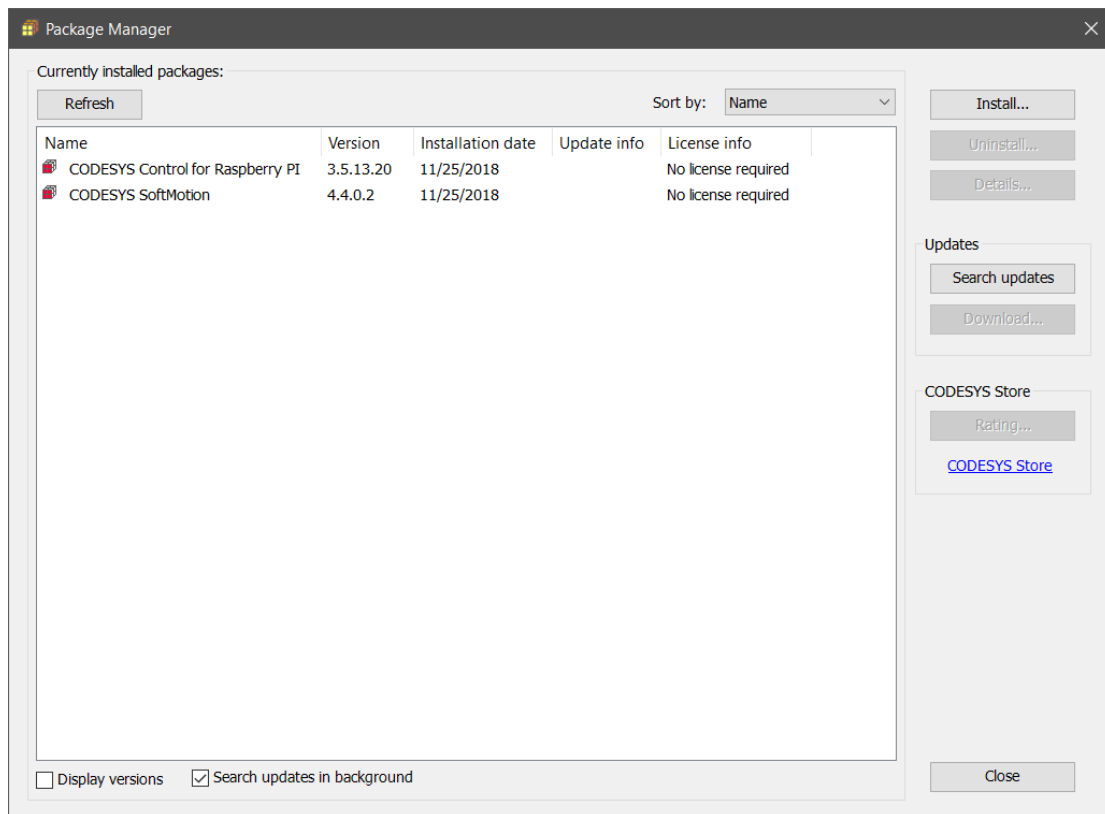
Στη συνέχεια πρέπει να κατεβάσουμε (αγοράζοντάς το) το **Software package** για την εκάστοτε συσκευή . Στη προκειμένη περίπτωση η συσκευή που χρησιμοποιούμε είναι Raspberry Pi οπότε θα κατεβάσουμε το πακέτο για τη συγκεκριμένη συσκευή .



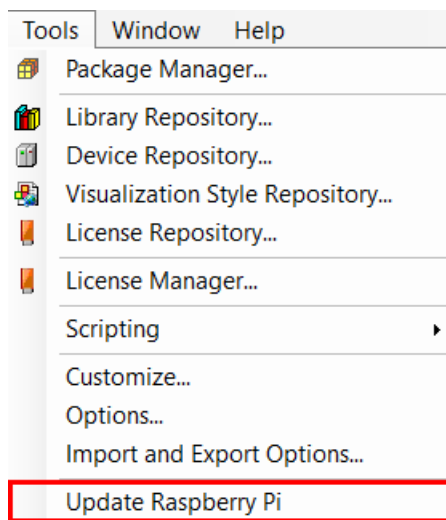
Στη συνέχεια αφ' ότου ανοίξουμε το CODESYS επιλέγουμε **Tools** και στη συνέχεια **Package Manager** .



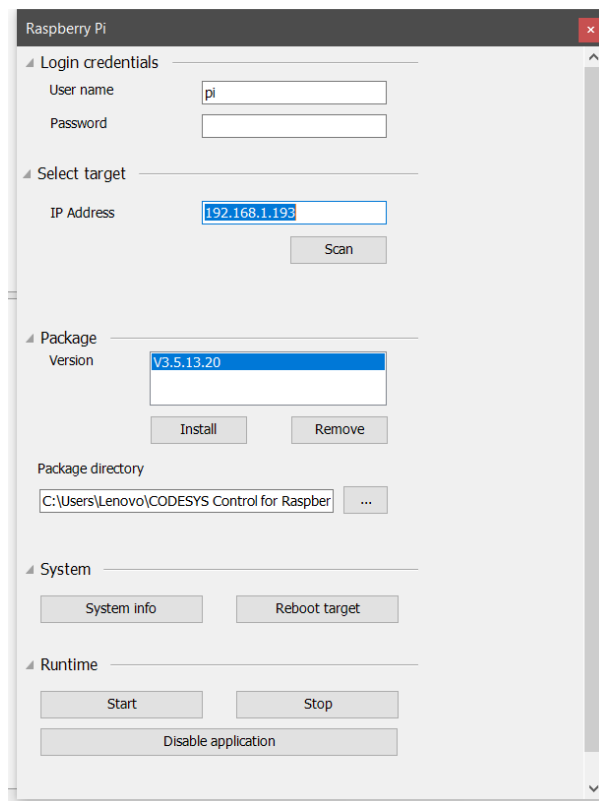
Για να εγκαταστήσουμε το πακέτο για το Raspberry πατάμε Install και στη συνέχεια επιλέγουμε το πακέτο.



Αφ' ότου έχουμε εγκαταστήσει το πακέτο τώρα μπορούμε να εγκαταστήσουμε τα απαραίτητα αρχεία στο Raspberry Pi. Για να γίνει αυτό πατάμε **Tools** και στη συνέχεια **Update Raspberry Pi**.



Πληκτρολογούμε το **username** και το **password** του Raspberry, την **IP διεύθυνση** και στη συνέχεια πατάμε **Install** προκειμένου να εγκαταστήσουμε τα απαραίτητα αρχεία στο Raspberry Pi. Αφ' ότου γίνει η εγκατάσταση πατάμε **Start**.



Όπως προτείνεται και στο forum της CODESYS , όταν χρησιμοποιείται USB to Serial Adapter , είναι αναγκαίο να εισάγουμε στο αρχείο **CODESYSControl.cfg** τις ακόλουθες γραμμές :

**[SysCom]**

**Linux.Devicefile=/dev/ttyUSB**

**portnum := COM.SysCom.SYS\_COMPORT1;**

Πληκτρολογώντας την εντολή **sudo nano /etc/CODESYSControl.cfg**

```
pi@raspberrypi: ~
GNU nano 2.2.6      File: /etc/CODESYSControl.cfg      Modified
[ CmpApp ]
Bootproject.RetainMismatch.Init=1
Application.1=Application

[ SysMem ]
Linux.Memlock=0

[ CmpCodeMeter ]
InitLicenseFile.0=3SLicense.wbb

[ SysEthernet ]
Linux.ProtocolFilter=3

[ SysCom ]
Linux.Devicefile=/dev/ttyUSB
portnum := COM.SysCom.SYS_COMPORT1;
```

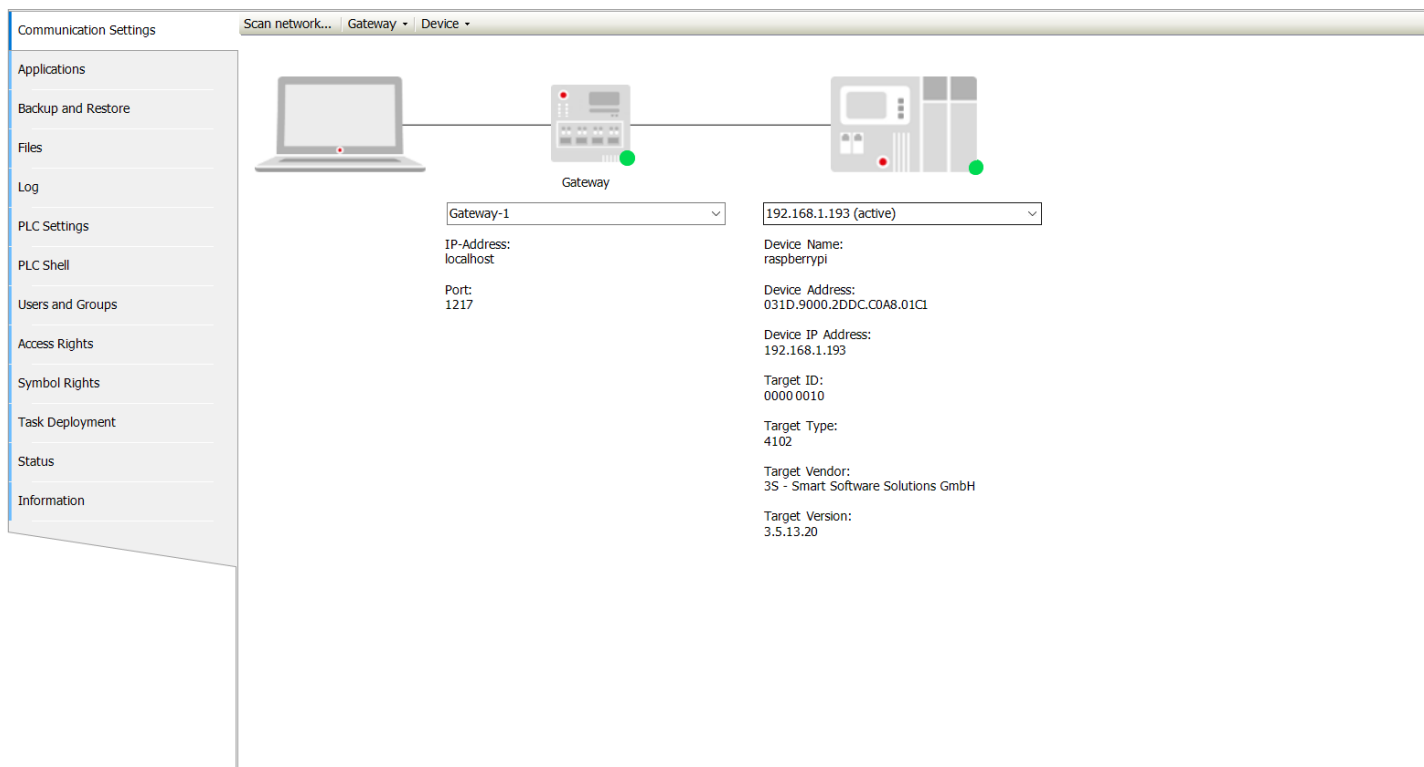
Η επανεκκίνηση είναι υποχρεωτική , αλλιώς πρέπει να σταματήσουμε και να ξαναξεκινήσουμε το Codesys Runtime Service με τις ακόλουθες εντολές

**sudo /etc/init.d/codesyscontrol stop**

και μετά

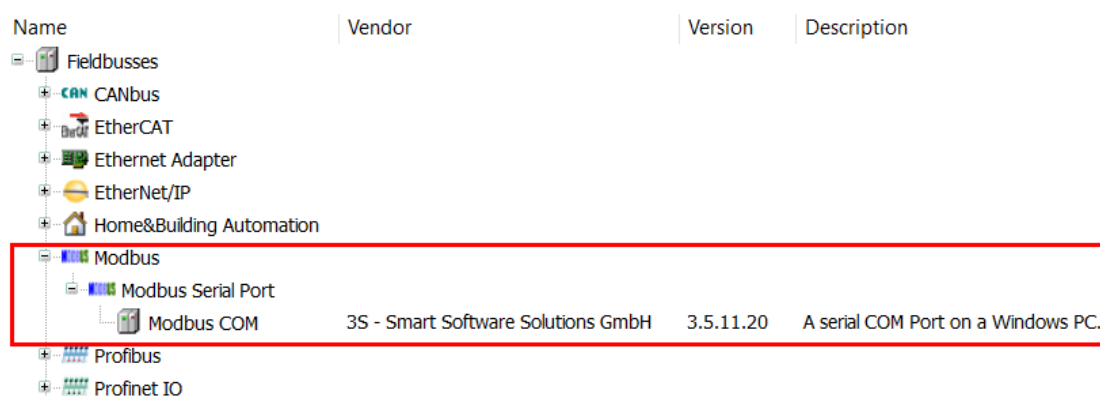
**sudo /etc/init.d/codesyscontrol start**

Στη συνέχεια δημιουργούμε ένα νέο **Codesys Standard Project** . Το πρώτο πράγμα που πρέπει να κάνουμε είναι να συνδεθούμε με το Raspberry Pi . Αυτό γίνεται πατώντας διπλό κλικ πάνω στο **Device** οπότε εμφανίζεται το παρακάτω παράθυρο.



Πληκτρολογούμε την **IP διεύθυνση** της συσκευής που θέλουμε να συνδεθούμε ( στη προκειμένη περίπτωση 192.168.1.193) και πατώντας **ENTER** γίνεται ανίχνευση στο δίκτυο . Η σύνδεση επιτυγχάνεται όταν δίπλα στην παρένθεση γράφεται **active** .

Θα πρέπει τώρα να συμπεριλάβουμε το πρωτόκολλο MODBUS στο project . Αυτό γίνεται πατώντας δεξί κλικ στο **Device** και επιλέγοντας το **Add Device** .Στη συνέχεια επιλέγουμε το **Modbus COM** και μετά **Add Device**.



Επίσης πρέπει να προσθέσουμε ένα Modbus RTU Master και ένα Modbus Slave. Αυτό γίνεται πατώντας δεξί κλικ στο **Modbus COM** και επιλέγοντας και **Modbus RTU Master** αλλά και **Modbus RTU Slave** .Τώρα θα πρέπει να ρυθμίσουμε και τις 2 πλευρές .Όσον αφορά το Modbus Master , οι μόνες ρυθμίσεις που πρέπει να γίνουν είναι οι παρακάτω.

General
ModbusGenericSerialMaster I/O Mapping
ModbusGenericSerialMaster IEC Objects
Status
Information

Modbus-RTU/ASCII

MODBUS

Transmission Mode
☒ RTU
☐ ASCII

Response Timeout (ms)

Time between Frames (ms)

☒ Auto-restart Communication

Ενώ για τον Modbus RTU Slave θα πρέπει να κάνουμε τις παρακάτω ρυθμίσεις

General
Modbus Slave Channel
Modbus Slave Init
ModbusGenericSerialSlave I/O Mapping
ModbusGenericSerialSlave IEC Objects
Status
Information

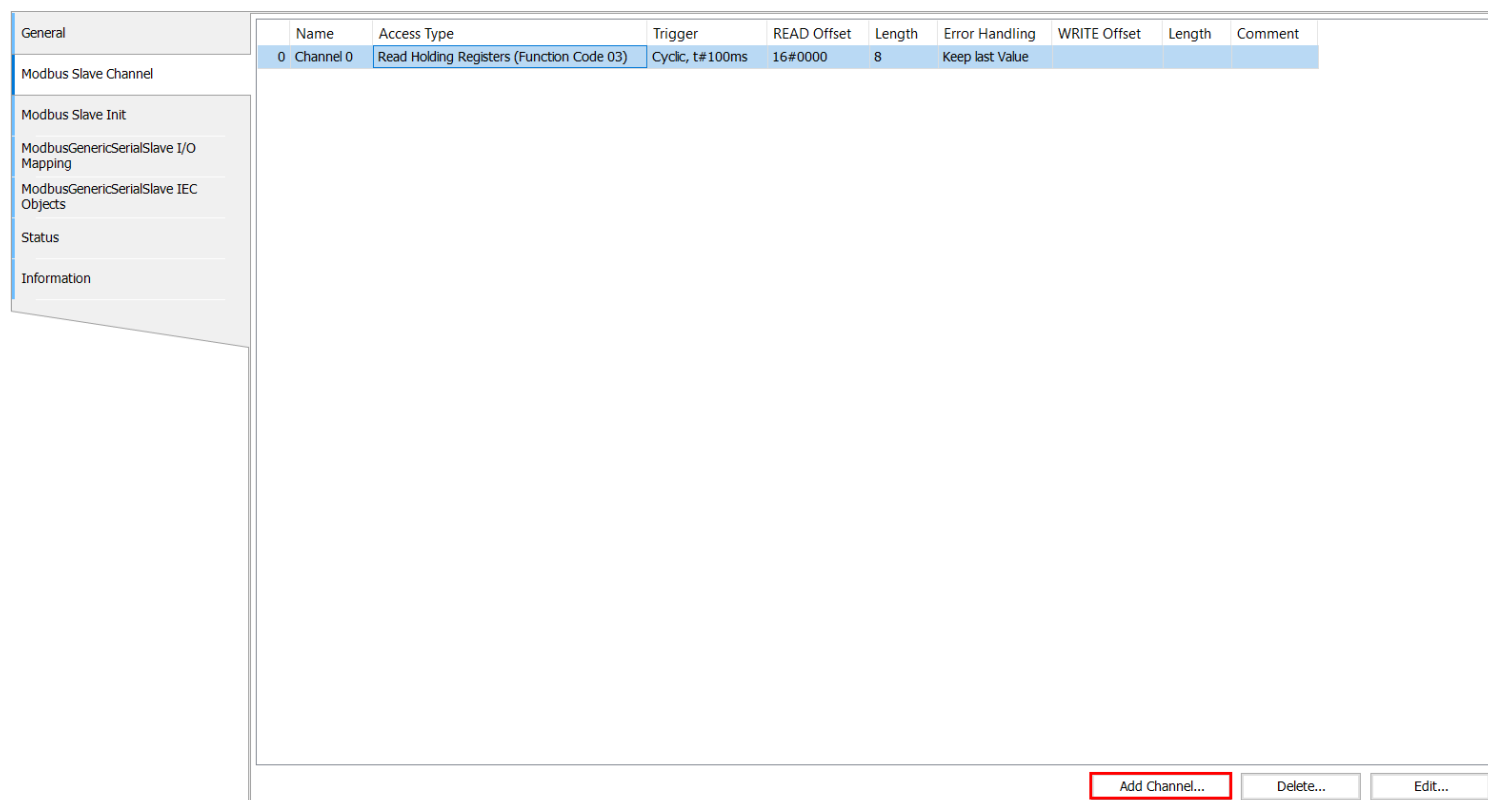
Modbus-RTU/ASCII

MODBUS

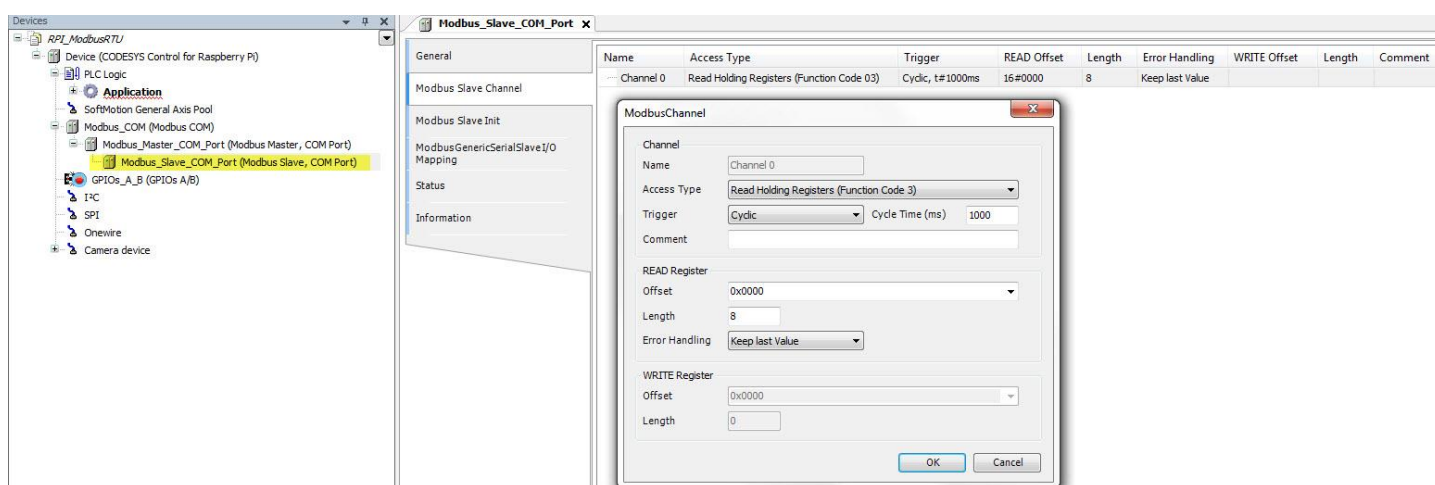
Slave Address [1...247]

Response Timeout [ms]

Επίσης θα πρέπει να εισάγουμε ένα κανάλι. Αυτό γίνεται πατώντας αριστερά στην επιλογή **Modbus Slave Channel** και στη συνέχεια **Add channel**.



Με αποτέλεσμα να μας εμφανιστεί το παρακάτω παράθυρο



Στη προκειμένη περίπτωση εμείς θέλουμε να διαβάσουμε Holding Registers οπότε στο **Access Type** επιλέγουμε **Read Holding Registers (Function Code 3)**. Επίσης θέλουμε να διαβάζουμε την τιμή 8 Holding Registers , οπότε στο **Length** πληκτρολογούμε το **8** . Αφ' ότου έχουμε ρυθμίσει το Modbus Master και το Modbus Slave μπορούμε να γράψουμε τον κώδικα του Raspberry Pi. Όπως έχει αναφερθεί προηγουμένως , οι γλώσσες προγραμματισμού στις οποίες μπορούμε να προγραμματίσουμε ανήκουν στο πρότυπο IEC 61131-3 . Στη προκειμένη περίπτωση θα προγραμματίσουμε στη γλώσσα **Structured Text** λόγω του ότι για το συγκεκριμένο σκοπό είναι πολύ πιο βολική σε σχέση με τις άλλες γλώσσες ( LAD , STL , FBD, SFC ).



```
1 PROGRAM PLC_PRG
2 VAR
3   Register4001:WORD;
4   Register4002:WORD;
5   Register4003:WORD;
6   Register4004:WORD;
7   Register4005:WORD;
8   Register4006:WORD;
9   Register4007:WORD;
10  Register4008:WORD;
11 END_VAR

1 Register4001:%IW0;
2 Register4002:%IW1;
3 Register4003:%IW2;
4 Register4004:%IW3;
5 Register4005:%IW4;
6 Register4006:%IW5;
7 Register4007:%IW6;
8 Register4008:%IW7;
```

Οι επιλογή των αριθμών στους καταχωρητές δεν είναι τυχαία καθώς όπως έχει αναφερθεί προηγουμένως οι Holding Registers ξεκινάνε από την Modbus διεύθυνση 4001. Τώρα έχοντας τελειώσει και με τη γραφή του προγράμματος , είμαστε έτοιμοι να το περάσουμε στο Raspberry Pi ούτως ώστε να λειτουργήσει ως Modbus RTU Master . Πατάμε **Online** και στη συνέχεια **Login** . Αν είναι η πρώτη φορά που κάνουμε Login , θα μας ζητηθεί να γίνει **download** του προγράμματος στο Raspberry Pi. Ύστερα από το κατέβασμα του προγράμματος , είμαστε σε θέση να κάνουμε Login . Στη συνέχεια πατάμε **Create boot application** και στη συνέχεια **Start** . Τώρα πλέον το Raspberry Pi λειτουργεί ως Modbus Master .

## Προσοχή !!

Μπορεί να υπάρξει η πιθανότητα ο Modbus RTU Slave να μην τρέχει ( Bug : *ttyUSB0: usb\_serial\_generic\_read\_bulk\_callback - urb stopped: -32* ) γεγονός που δεν πρέπει να μας ανησυχήσει . Για την επίλυση του προβλήματος υπάρχουν οι εξής τρόποι :

- Να κάνουμε **Disable** τη σειριακή επικοινωνία ( **Serial** ) από τα Interfaces στο **Raspberry Pi Configuration**

Σε περίπτωση που δεν λειτουργήσει ο παραπάνω τρόπος πρέπει να κάνουμε τα εξής :

- Προσθέτουμε "**max\_usb\_current=1**" στο **/boot/config.txt**
- Αλλάζουμε το αρχείο **/boot/cmdline.txt** σε : "**dwc\_otg.lpm\_enable=0 console=tty1 root=PARTUUID=965d45c8-02 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait dwc\_otg.fiq\_fsm\_mask=0x0 dwc\_otg.speed=1**"

Επίσης , καλό είναι η διακοπή της σειριακής επικοινωνίας να γίνεται ομαλά ( σταμάτημα του Raspberry Pi , Log Out και κλείσιμο του Project ) και όχι απλά βγάζοντας το καλώδιο καθώς υπάρχει η πιθανότητα conflict