Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2023-2024

ΑΘΗΝΑ 1 Δεκεμβρίου 2023

8η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Συνδυαστική/Επαναληπτική άσκηση – Εφαρμογή Internet of Things (στο ntuAboard**)**
Εξέταση – Επίδειξη: Τετάρτη 15/12/2023

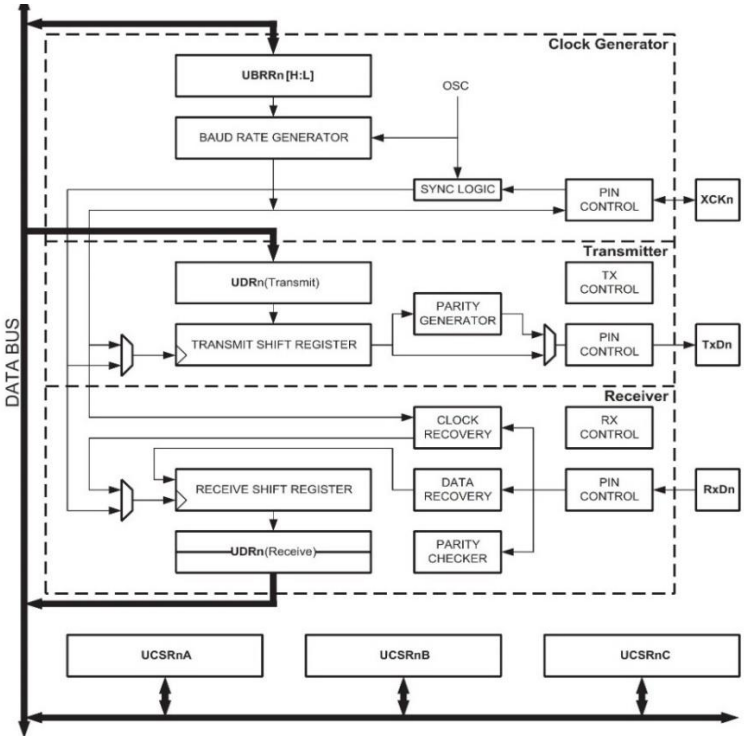
Προθεσμία για παράδοση Έκθεσης: Τρίτη 19/12/2023 23:59

Ασύγχρονη σειριακή επικοινωνία UART

Ο ATmega328PB είναι εξοπλισμένος με δυο μονάδες USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) οι οποίες μπορούν να χρησιμοποιηθούν για Σύγχρονη αλλά και για Ασύγχρονη επικοινωνία. Στο πλαίσιο αυτής της άσκησης θα μελετηθεί ο Ασύγχρονος τρόπος επικοινωνίας (UART) και η μονάδα USART1. Οι δυο μονάδες είναι ισοδύναμες και για αυτό υπάρχει σε όλα τα σχηματικά το γράμμα n, όπου n=0 για την πρώτη μονάδα και n=1 για την δεύτερη.
Η μονάδα UART μπορεί να μεταδώσει 30 συνδυασμούς πλαισίου:

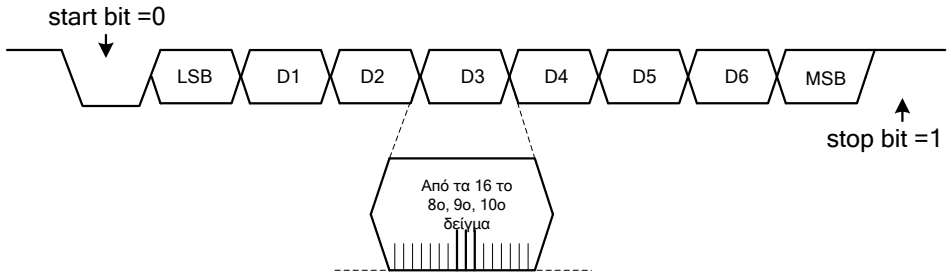
- 1 bit εκκίνησης
- 5,6,7,8 ή 9 bits
- Ένα η δύο bit λήξης
- Καμία, άρτια ή περιττή ισοτιμία (parity)

Η βασική δομή φαίνεται στο παρακάτω σχήμα:



Σχήμα 8.1. Κυκλωματικό διάγραμμα Μονάδας UART.

Ο ακροδέκτης XCKn (Transfer Clock) χρησιμοποιείται μόνο στο σύγχρονο τρόπο επικοινωνίας.
Στο ρυθμό μετάδοσης υπάρχει διαίρεση με το 16. Ο πομπός χρησιμοποιεί την διαιρεμένη συχνότητα για την αποστολή ενός bit ενώ ο δέκτης χρησιμοποιεί το 16πλάσιο της συχνότητας αυτής ώστε να δειγματοληπτεί το σήμα. Από τα δείγματα που λήφθηκαν χρησιμοποιούνται το 8ο, το 9ο και το 10ο και με τον κανόνα της πλειοψηφίας αποφασίζεται αν λήφθηκε λογικό 1, λογικό 0 ή αν πρόκειται για θόρυβο στην περίπτωση που αναμένεται start bit (που πρέπει οπωσδήποτε να είναι 0).
Όταν αναγνωριστεί το start bit πραγματοποιείται ένας ¨συγχρονισμός¨ του δέκτη με τον πομπό, αφού αναγνωρίζεται η μετάβαση από λογικό 1 σε λογικό 0 και στη συνέχεια το επόμενο bit θα καταφθάσει σε συγκεκριμένο χρόνο (1/BAUD).
Ο ρυθμός μετάδοσης, ο αριθμός των bit που αναμένονται, ο αριθμός των stop bit και το είδος του parity bit πρέπει να είναι όλα γνωστά στον δέκτη για να λειτουργήσει σωστά η επικοινωνία.



Σχήμα 8.2 Δειγματοληψία δεδομένου

Περισσότερες πληροφορίες σχετικά με την λειτουργία των καταχωρητών αλλά και γενικότερα της UART μπορείτε να βρείτε στο datasheet ATmega328PB.

Καταχωρητές

Καταχωρητής δεδομένων (UDRn)

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Από το σχήμα φαίνεται ότι πομπός και δέκτης μοιράζονται τον καταχωρητή δεδομένων UDRn. Στην πραγματικότητα όμως πρόκειται για δύο διαφορετικούς καταχωρητές που έχουν κοινή φυσική διεύθυνση 0xC6 (UDR0). Ο ένας είναι ο καταχωρητής δεδομένων εκπομπής όπου μεταφέρονται τα δεδομένα που γράφουμε στη διεύθυνση 0xC6. Ο άλλος είναι ο καταχωρητής δεδομένων λήψης όπου μεταφέρονται τα δεδομένα που διαβάζουμε από τη διεύθυνση 0xC6. Ο καταχωρητής δεδομένων λήψης είναι δομημένος σε FIFO 2 επιπέδων και η κατάσταση του αλλάζει σε κάθε προσπέλαση. Γι αυτό το λόγω δε πρέπει να χρησιμοποιούνται οι εντολές SBI και CBI και χρειάζεται ιδιαίτερη προσοχή στη χρήση των SBIC και SBIS αφού και αυτές αλλάζουν την κατάσταση του καταχωρητή.

Καταχωρητής κατάστασης (USCRnA)

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

RXCn: Αν είναι λογικό 1 τότε υπάρχουν δεδομένα προς ανάγνωση. Μετά την ανάγνωση τους η τιμή μηδενίζεται.
TXCn: Τίθεται σε λογικό 1 όταν έχει ολοκληρωθεί η ολίσθηση όλων των δεδομένων και δεν υπάρχουν καινούρια δεδομένα στον καταχωρητή UDRn.
UDREn: Αν είναι λογικό 1 ο καταχωρητής αποστολής δεδομένων UDRn είναι έτοιμος να δεχθεί νέα δεδομένα.

Καταχωρητής ελέγχου (USCRnB)

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

RXENn: Αν είναι λογικό 1 ενεργοποιείται το τμήμα λήψης της μονάδας UART
TXENn: Αν είναι λογικό 1 ενεργοποιείται το τμήμα εκπομπής της μονάδας UART

Καταχωρητής ελέγχου (USCRnC)

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

UMSELn1:0: Επιλογή του τρόπου λειτουργίας της USART. 00 για Ασύγχρονη λειτουργία.
UCSZn1:0: Τα bit UCSZn1:0 σε συνδυασμό με το bit UCSZn2 του καταχωρητή UCSRnB καθορίζουν των αριθμό των bit δεδομένων (Character Size) που ανταλλάσσουν πομπός και δέκτης. Συγκεκριμένα:

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Καταχωρητής του ρυθμού μετάδοσης (UBRRn)

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

UBRR11:0: Πρόκειται για έναν καταχωρητή 12-bit που περιέχει το ρυθμό μετάδοσης. Ο καταχωρητής UBRRnH περιέχει τα 4 MSB, και ο UBRRnL περιλαμβάνει τα 8 LSB. Εγγραφή στον UBRRnL ανανεώνει άμεσα τον ρυθμό και θα προκαλέσει αλλοίωση των δεδομένων σε περίπτωση που γίνει κατά τη διάρκεια εκπομπής.

Για τον υπολογισμό του ρυθμού μετάδοσης BAUD από την τιμή του καταχωρητή ισχύει ο τύπος

$$BAUD = \frac{f_{osc}}{16(UBRR + 1)}$$

όπου f_{OSC} η συχνότητα λειτουργίας του μικροελεγκτή. Λύνοντας ως προς UBRR έχουμε:

$$UBRR = \frac{f_{osc}}{16BAUD} - 1$$

Για να έχουμε BAUD=9600 και αφού ο μικροελεγκτής μας λειτουργεί στα 16MHz πρέπει UBRR=103.1667. Στρογγυλοποιούμε στο 103, εισάγοντας ένα σφάλμα 0,16%.

Για να αρχικοποιήσουμε την USART μπορούμε να χρησιμοποιήσουμε την εξής ρουτίνα:

```
/* Routine: usart_init
 Description:
 This routine initializes the
 usart as shown below.
 ------- INITIALIZATIONS -------

 Baud rate: 9600 (Fck= 8MH)
 Asynchronous mode
 Transmitter on
 Reciever on
 Communication parameters: 8 Data ,1 Stop, no Parity
 --------------------------------
 parameters: ubrr to control the BAUD.
 return value: None.*/

void usart_init(unsigned int ubrr){
         UCSR0A=0;
         UCSR0B=(1<<RXEN0)|(1<<TXEN0);
         UBRR0H=(unsigned char)(ubrr>>8);
         UBRR0L=(unsigned char)ubrr;
         UCSR0C=(3 << UCSZ00);
         return;
}

/* Routine: usart_transmit
 Description:
 This routine sends a byte of data
```

```
 using usart.
 parameters:
 data: the byte to be transmitted
 return value: None. */

void usart_transmit(uint8_t data){
       while(!(UCSR0A&(1<<UDRE0)));
       UDR0=data;
}


/* Routine: usart_receive
 Description:
 This routine receives a byte of data
 from usart.
 parameters: None.
 return value: the received byte */

uint8_t usart_receive(){
       while(!(UCSR0A&(1<<RXC0)));
       return UDR0;
}
```

Σύντομη εισαγωγή στο Internet of Things

Η ραγδαία ανάπτυξη του Internet of Things (IoT) έχει επιφέρει σημαντική αύξηση των συσκευών που είναι συνδεδεμένες στο διαδίκτυο. Ο αριθμός αυτός εξακολουθεί να αυξάνεται με ταχύτατους ρυθμούς. Συνέπεια αυτού είναι η παραγωγή τεράστιου όγκου πληροφοριών προς επεξεργασία.

Τι είναι οι Συσκευές IoT:

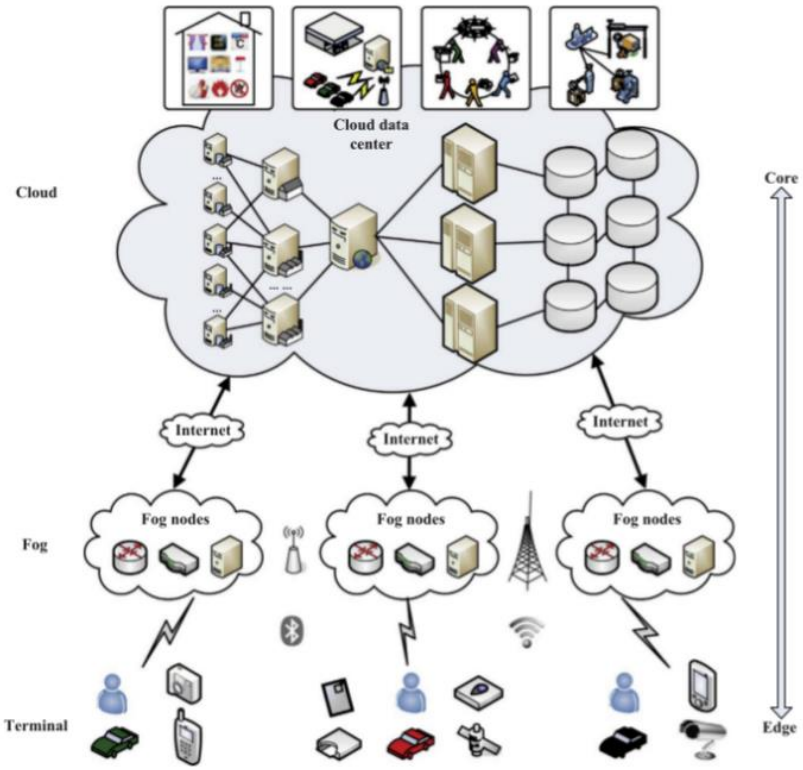
Ως συσκευές IoT ορίζουμε κάθε είδους φυσική συσκευή, οχήματα, οικιακές συσκευές και άλλα αντικείμενα, στα οποία υπάρχουν ενσωματωμένα αισθητήρες, λογισμικό, ενεργοποιητές και σύνδεση στο διαδίκτυο, που τους επιτρέπει να επικοινωνούν και να ανταλλάσσουν δεδομένα είτε μεταξύ τους, είτε με το υπόλοιπο διαδίκτυο. Αυτές οι συσκευές έχουν περιορισμένους πόρους σε ό,τι αφορά την cpu, τη μνήμη και τις δικτυακές τους ικανότητες.

Τι είναι IoT Gateway:

Καθώς δισεκατομύρια συσκευές συνδέονται πλέον στο δίκτυο, το «Gateway» χρειάζεται προκειμένου να συνδέει υπο-δίκτυα διαφορετικών αρχιτεκτονικών και σε διαφορετικά περιβάλλοντα. Στην κλασική δομή του IoT χρησιμοποιείται ως διαμεσολαβητής/γέφυρα μεταξύ των τελικών συσκευών και του Cloud διαμορφώνοντας κατάλληλα τα δεδομένα που στέλνονται και λαμβάνονται. Τα Gateways συνδέουν επίσης στο Cloud υπο-δίκτυα που στο εσωτερικό τους μπορεί να χρησιμοποιούν διαφορετικά πρωτόκολλα επικοινωνίας. Τα Gateways εποπτεύουν την διαδικασία επικοινωνίας εντός και εκτός του υπο-δικτύου στο οποίο βρίσκονται. Μπορούν να εκτελούν διεργασίες ελέγχου, ασφάλειας, φιλτραρίσματος των δεδομένων καθώς επίσης να κάνουν σε χαμηλότερο επίπεδο επεξεργασία των δεδομένων που λαμβάνουν (πιο κοντά στις συσκευές και όχι στο Cloud). Τέλος, μεγάλος αριθμός δεδομένων που λαμβάνονται από τους αισθητήρες των υπο-δικτύων IoT, χρειάζεται το Gateway ως σύστημα συντονισμού και ελέγχου της μεταφοράς τους όχι μόνο από μία συσκευή στο Cloud αλλά και από συσκευή σε συσκευή.

Edge / Fog Computing:

Ωστόσο, τα τελευταία χρόνια, λόγω του γεγονότος ότι οι υποδομές του Cloud είναι γεωγραφικά κεντρικοποιημένες και απομακρυσμένες από τους χρήστες, οι εφαρμογές που απαιτούν επεξεργασία σε πραγματικό χρόνο, με χαμηλό latency αδυνατούν να εκτελεστούν επιτυχώς. Αυτό συμβαίνει λόγω μεγάλου χρόνου μεταφοράς των δεδομένων, συμφόρησης του δικτύου και υποβάθμισης της ποιότητας. Επιπλέον, σε εφαρμογές που απαιτείται ιδιωτικότητα των δεδομένων (π.χ. ιατρικές εφαρμογές), αυτή δε μπορεί να διασφαλιστεί απόλυτα σε απομακρυσμένες υποδομές. Αυτοί οι λόγοι οδήγησαν στη δημιουργία του Edge και του Fog Computing, όπου οι απαιτούμενοι υπολογισμοί γίνονται κοντά στις IoT συσκευές. Το σχήμα της αρχιτεκτονικής αυτής παρουσιάζεται στην Εικόνα 1. Έτσι οι χρήστες ανεξαρτητοποιούνται από τις υποδομές του Cloud και, ταυτόχρονα, ενισχύεται η προστασία των προσωπικών τους δεδομένων.



Εικόνα 1 Αρχιτεκτονική Fog και Edge Computing[1]

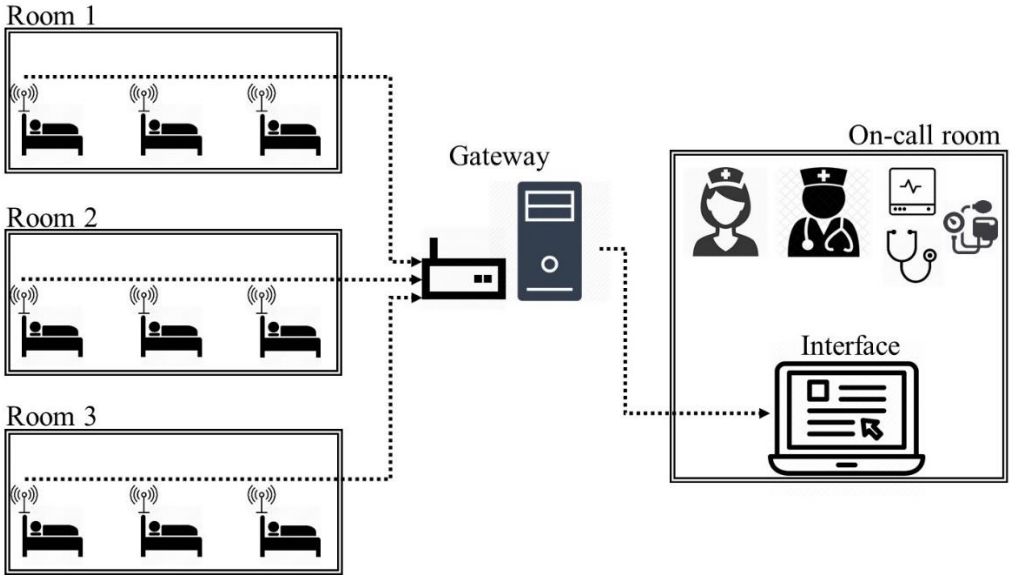
Από τη μία πλευρά, οι συσκευές IoT βρίσκονται κοντά στους καθημερινούς χρήστες και πιο συγκεκριμένα, στο terminal layer της αρχιτεκτονικής που φαίνεται στην Εικόνα 1. Από την άλλη πλευρά ο Edge Node (Gateways του Edge Computing, τα οποία συνεισφέρουν στην ανάπτυξη υπηρεσιών Edge και στη παροχή υπολογιστικών, αποθηκευτικών και δικτυακών πόρων στις συσκευές IoT) βρίσκεται στη διεπαφή του Fog layer και του terminal layer. Επιπροσθέτως, οι συσκευές IoT είναι συνδεδεμένες, είτε ενσύρματα, είτα ασύρματα, με το Gateway κόμβο και μπορούν να επικοινωνούν και να ανταλλάσσουν δεδομένα με εκείνον.

Η αρχιτεκτονική του συστήματος που θα μελετήσουμε στα πλαίσια της παρούσας εργασίας αποτελείται από ένα σύνολο n IoT συσκευών (ntuAboard πλακέτες) και μίας συσκευής Gateway. Δεδομένα από αισθητήρες συνδεδεμένους στις συσκευές μπορούν να σταλούν προς επεξεργασία στο Gateway, ενώ τον Gateway μπορεί να στέλνει δεδομένα πίσω στις συσκευές.

[1] P. Hu, S. Dhelim, H. Ning, and T. Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. Journal of Network and Computer Applications, 2017

Περιγραφή της εφαρμογής

Στην παρούσα άσκηση θα μελετηθεί μια εφαρμογή νοσοκομείου. Η Εικόνα 2 δίνει το γενικό σχήμα της εφαρμογής. Το κάθε κρεβάτι (IoT node) έχει συνδεδεμένους αισθητήρες στον ασθενή, τα δεδομένα των οποίων στέλνονται στο Gateway. Το Gateway επεξεργάζεται τα δεδομένα και είτε στέλνει κάποια απάντηση στο IoT node προκειμένου να γίνει κάποια αυτόματη λειτουργία, είτε στέλνει τα δεδομένα στο δωμάτιο των νοσηλευτών και των γιατρών. Τα δεδομένα παρουσιάζονται σε μια οθόνη (Interface) ούτως ώστε να μελετηθούν από το προσωπικό του νοσοκομείου προκειμένου να προβεί στις κινήσεις που απαιτούνται.



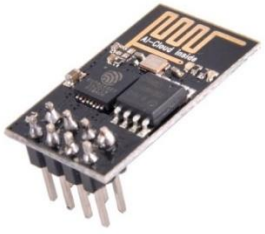
Εικόνα 2 Η εφαρμογή IoT που θα μελετηθεί στην παρούσα άσκηση

Η κάθε συσκευή IoT που βρίσκεται σε κάθε κρεβάτι θα είναι μια πλακέτα ntuAboard, το πρόγραμμα της οποίας θα γραφτεί από την κάθε ομάδα. Ο αριθμός της ομάδας θα είναι ο αριθμός του κρεβατιού. Για λόγους απλότητας ο ένας αισθητήρας που θα χρησιμοποιήσετε θα είναι ο αισθητήρας θερμοκρασίας DS18B20 που μελετήθηκε σε προηγούμενη Εργαστηριακή άσκηση. Για τον δεύτερο αισθητήρα θα χρησιμοποιηθεί ένα ποτενσιόμετρο για να προσομοιώσει την λειτουργία ενός αισθητήρα κεντρικής φλεβικής πίεσης 0-20 cm H20.

Το Gateway θα είναι ήδη προγραμματισμένο και το Interface του δωματίου προσωπικού του νοσοκομείου θα παρουσιάζεται στον προβολέα του εργαστηρίου. Αν κάποια ομάδα το επιθυμεί, υπάρχει δυνατότητα να αναπτύξει δικό της πρόγραμμα για το Gateway.

Ο πομποδέκτης WiFi ESP8266:

Ο πομποδέκτης Wi-Fi που θα χρησιμοποιήσουμε για την παραπάνω εφαρμογή είναι ο ESP8266 (Εικόνα 3). Πρόκειται για έναν προγραμματίσιμο πομποδέκτη ο οποίος έχει προγραμματιστεί να δέχεται εντολές μέσω σειριακής και να στέλνει αντίστοιχη απάντηση.



Εικόνα 3 Ο πομποδέκτης WiFi ESP8266

Συγκεκριμένα, κάποιες από τις εντολές που δέχεται φαίνονται στον παρακάτω πίνακα:

command	usage
baudrate: "Your baudrate"	Resets the Serial to the required speed
configuration	Reports your current configuration
connect	Uses the provided SSID and Password to connect to the WiFi network
debug: "true or false"	Every command will send extra debug information if set to true
help	Lists all commands
password: "Your Password"	Password to be used for the connection
payload: [Your payload]	The payload to post to the Host. Must be in brackets
restart	Restarts the ESP
ssid: "Your SSID"	SSID to be used for the connection
transmit	Transmits the specified payload to the url
url: "Your url"	The url to transmit to. Example: http://192.168.1.250:5000/data"

Προκειμένου το ESP8266 να αναγνωρίσει ότι πρέπει να διαβάσει τα δεδομένα της σειριακής, οτιδήποτε στείλετε πρέπει να ξεκινάει με ESP: (π.χ. για σύνδεση θα στείλετε ESP:connect).
Το ESP μπορεί να συνδεθεί στον server σαν client και να στέλνει αιτήματα στον server για να γίνει ανταλλαγή δεδομένων.

Δώστε προσοχή στα εξής:

1) Σε όποια εντολή υπάρχουν τα "" πρέπει να τα στέλνετε. Κάθε εντολή που στέλνετε καθώς και κάθε απάντηση που λαμβάνετε θα έχει στο τέλος τον χαρακτήρα αλλαγής γραμμής '\n'.
2) Μερικές φορές οι client αποσυνδέονται οπότε να ελέγχετε κάθε φορά με την εντολή connect και μπορείτε σε περίπτωση που αποτύχει να δοκιμάζετε άλλη μια φορά (σε περίπτωση αποτυχίας υπάρχει timeout 20sec όπου το ESP δεν θα αποκρίνεται).
3) Μπορείτε στην αρχή του προγράμματος σας να στέλνετε την εντολή restart για να σιγουρευτείτε ότι το ESP βρίσκεται στις default επιλογές του. Μετά την επανεκκίνηση το ESP τυπώνει μια γραμμή με κάποιες πληροφορίες και την γραμμή ESP8266: Waiting for command.
4) Το payload που δέχεται ο server πρέπει να είναι σε μορφή json και συγκεκριμένα η εντολή πρέπει να είναι στην μορφή:

```
payload:
[{"name": "temperature","value": "36.0"},{"name": "pressure","value": "60.0"},{"name": "team","va
lue": "5"},{"name": "status","value": "OK"}]
```

5) Η μεταβλητή team είναι απαραίτητη στο payload και σε περίπτωση που δεν την συμπεριλάβετε θα λάβετε ειδικό μήνυμα λάθους από το gateway. Σε περίπτωση που λείπει κάποια από τις άλλες μεταβλητές η έχετε κάνει τυπογραφικό θα συμπληρωθεί με None.
6) Τα ονόματα των μεταβλητών στο payload είναι αυστηρά. Σε όποια εντολή υπάρχουν τα "" πρέπει να τα στέλνετε. Κάθε εντολή πρέπει να τελειώνει με τον χαρακτήρα αλλαγής γραμμής \n. Τα ssid, password και baudrate(9600) **έχουν ήδη ρυθμιστεί για εσάς**.

Default τιμές:

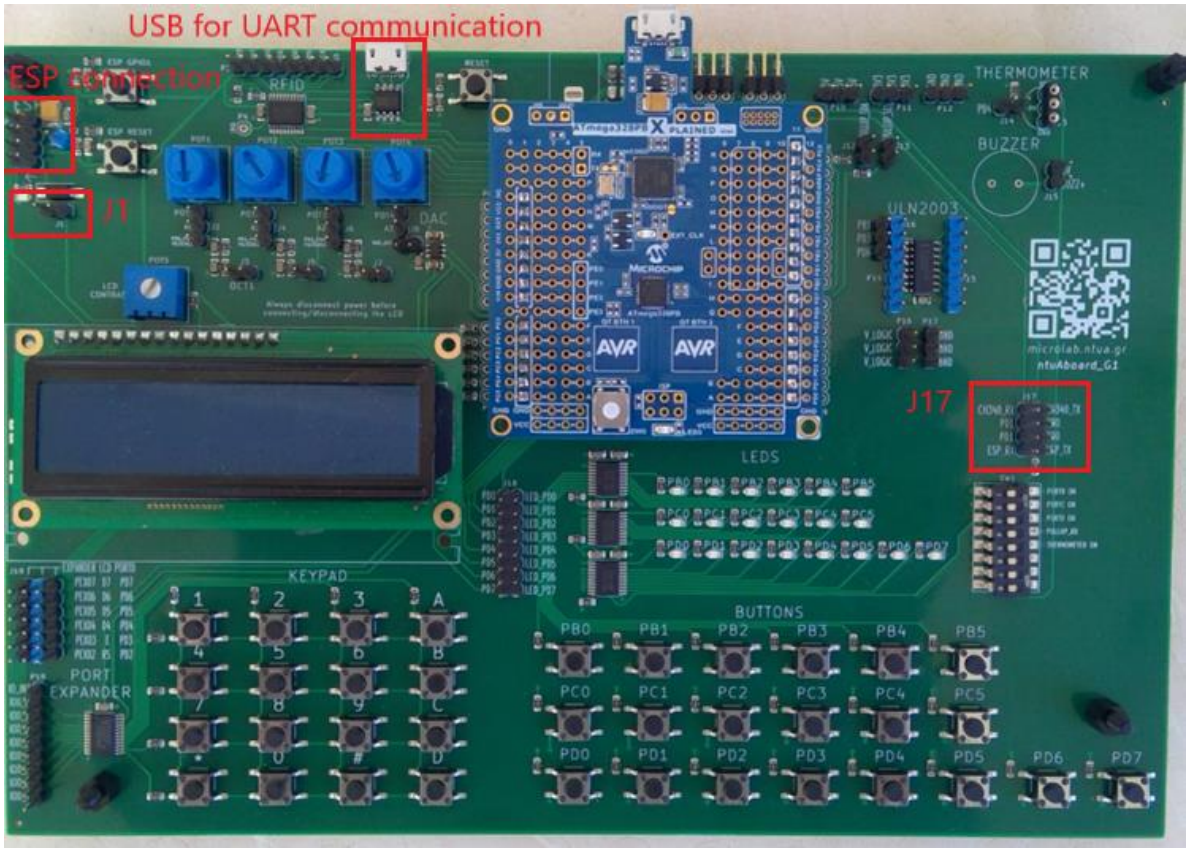
baudrate	9600
debug	false
password	Microlab_IoT
ssid	Micro_IoT

Σας δίνεται παράδειγμα επικοινωνίας με το ESP:

SENDER	MESSAGE
AVR	ESP:connect
ESP	"Success"
AVR	ESP:url:"http://192.168.1.250:5000/data"
ESP	"Success"
AVR	ESP:payload:[{"name": "team","value": "5"}]
ESP	"Success"
AVR	ESP:transmit
ESP	200 OK

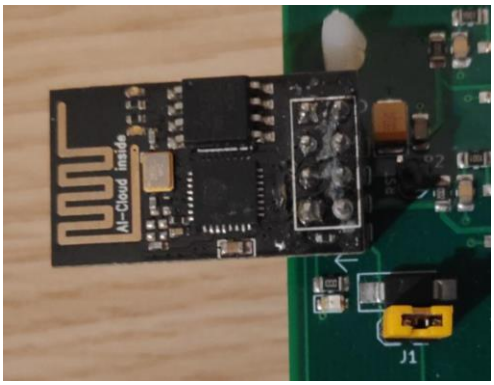
Jumpers

Η UART μπορεί μέσω του J17 να συνδεθεί είτε με το ESP8266 είτε με το CH340N το οποίο είναι ένα ολοκληρωμένο υπεύθυνο για την μετατροπή των σημάτων του διαύλου USB σε σήματα σειριακής επικοινωνίας και το αντίστροφο. Αυτό δίνει την δυνατότητα στον υπολογιστή σας να επικοινωνήσει με τη θύρα UART του ATmega328 αν συνδέσετε ένα επιπλέον καλώδιο USB από τον υπολογιστή σας στο αριστερό μέρος του ntuAboard που φαίνεται στην εικόνα. Για την σύνδεση με το ESP8266 χρειάζεται να συνδέσετε το ESP8266 στο ntuAboard, τα jumper στα J17 και J1 όπως φαίνονται στις παρακάτω εικόνες.
ΠΡΟΣΟΧΗ: Συνδέστε το ESP8266 μόνο με τη φορά που φαίνεται στην εικόνα!



Εικόνα 4 ESP8266 και UART στο ntuAboard

Εικόνα 5 Jumper στο J17 για επικοινωνία με ESP8266 (αριστερά) ή το CH340 (δεξιά)



Εικόνα 6 Σύνδεση ESP8266 και J1 jumper

Μπορείτε να χρησιμοποιήσετε την UART και να προσομοιώνετε εσείς τις απαντήσεις του ESP8266 από τον υπολογιστή για να σιγουρευτείτε ότι λειτουργεί σωστά η σειριακή επικοινωνία. Για την επικοινωνία με τον υπολογιστή σας μπορείτε να χρησιμοποιήσετε οποιοδήποτε πρόγραμμα σειριακής επικοινωνίας επιθυμείτε. Ένα πρόγραμμα είναι το Arduino IDE 2.0.3 που μπορείτε να κατεβάσετε στον υπολογιστή σας από εδώ (https://www.arduino.cc/en/software) και αφού το ανοίξετε επιλέξτε Tools->Port και επιλέξτε τη θύρα που είναι συνδεδεμένο το USB σειριακής επικοινωνίας (το αριστερά) και στη συνέχεια Tools->Serial Monitor. Επιλέξτε το σωστό baud από δεξιά κάτω.

Για τα πρώτα ερωτήματα θα χρησιμοποιήσετε την UART και θα προσομοιώνεται εσείς τις απαντήσεις του ESP8266 και μόνο στο τελευταίο ερώτημα θα συνδέσετε το ESP8266 στο ntuAboard.

Τα ζητούμενα της 8ης εργαστηριακής άσκησης

Ζήτημα 8.1 Να γραφεί πρόγραμμα σε C για τον ATmega328 στο ntuAboard το οποίο να στέλνει στην UART κατάλληλη εντολή για να συνδεθεί το ESP8266 στο δίκτυο. Στη συνέχεια, θα διαβάζει από την UART και θα περιμένει την απάντηση του ESP8266 και αν αυτή είναι "Success" θα απεικονίζει στην LCD το μήνυμα 1.Success ενώ αν είναι "Fail" θα τυπώνει στην LCD το μήνυμα 1.Fail . Επαναλάβετε το ίδιο για την εντολή url ώστε να θέσετε το url σε http://192.168.1.250:5000/data και αντίστοιχα να εμφανίζεται μηνύματα 2.Success και 2.Fail. Μπορείτε να προσθέσετε μια καθυστέρηση πριν την αποστολή της δεύτερης εντολής για να προλάβετε να δείτε τα μηνύματα στην LCD.

Ζήτημα 8.2 Επεκτείνετε το παραπάνω πρόγραμμα ώστε μετά την εκτύπωση του 2ου μηνύματος για επαρκή χρόνο

(α) να διαβάζει μια μέτρηση από τον αισθητήρα θερμοκρασίας DS18B20 με σκοπό να ενσωματώσετε την τιμή στο payload που θα στείλετε. Προσθέστε στην τιμή του αισθητήρα μια τιμή ώστε το αποτέλεσμα σας να είναι σε συνθήκες θερμοκρασίας δωματίου κοντά στο 36 για να προσομοιάζει πραγματική θερμοκρασία ασθενούς.

(β) να διαβάζει μια μέτρηση από το POT0 και να την μετατρέπετε σε κλίμακα 0-20 cm H20 για να συμπεριφέρεται σαν αισθητήρας κεντρικής φλεβικής πίεσης. Και αυτή η τιμή πρέπει να ενσωματωθεί στο payload.

(γ) να διαμορφώνει το status ως εξής:

1) Αν πατηθεί στο πληκτρολόγιο το πλήκτρο που αντιστοιχεί στο τελευταίο ψηφίο της ομάδας σας τότε το status να γίνεται NURSE CALL. Αν πατηθεί στη συνέχεια η δίεση # τότε το status να γίνεται OK εκτός αν συμβαίνει κάτι από τα 2 η 3.

2) Αν η πίεση είναι πάνω από 12 η κάτω από 4 τότε το status να γίνεται CHECK PRESSURE.

3) Αν η θερμοκρασία είναι κάτω από 34 η πάνω από 37 τότε το status να γίνεται CHECK TEMP.

4) Αν δεν έχει συμβεί κανένα από τα παραπάνω τότε το status να γίνεται OK.

Την τελική τιμή του status ενσωματώστε την στο payload.

Εμφανίστε τις τιμές της θερμοκρασίας και του αισθητήρα κεντρικής φλεβικής πίεσης στην 1η γραμμή της LCD και το status στην 2η γραμμή για επαρκή χρόνο.

Ζήτημα 8.3 Επεκτείνετε το παραπάνω πρόγραμμα ώστε να στέλνει στην UART κατάλληλες εντολές για την αποστολή του payload στον server. Αυτές είναι η εντολή payload για την οποία αντίστοιχα να εμφανίζεται μηνύματα 3.Success και 3.Fail και η εντολή transmit. Για την transmit δεν στέλνει το ESP Success η Fail αλλά στέλνει κατευθείαν την απάντηση του Server. Να τυπώνετε στην οθόνη την απάντηση του server στη μορφή 4. Απάντηση server. Αν όλα έχουν γίνει σωστά πρέπει να λάβετε 200 OK και να δείτε τα στοιχεία που στείλατε στον προβολέα του εργαστηρίου.

Το πρόγραμμα πρέπει να επαναλαμβάνει την λειτουργία του από το 8.1-8.3.