# Σειριακή επικοινωνία

Τα Arduino μέσω της κλάσης Serial που διαθέτουν, μπορούν να επικοινωνούν με τον υπολογιστή και άλλες εξωτερικές συσκευές όπως bluetooth modules. Στο Arduino Uno, για την επικοινωνία αυτή χρησιμοποιούνται τα pins RX και TX, για λήψη και μετάδοση δεδομένων αντίστοιχα.

**Βασικές functions:**

- **Serial.begin(long speed):**

Με αυτή τη μέθοδο θέτουμε το baud rate με το οποίο θα επικοινωνούμε με την συσκευή την οποία θέλουμε. Την τοποθετούμε μέσα στη void setup. Για το serial monitor, δηλαδή τη κονσόλα που διαθέτει ο υπολογιστής για επικοινωνία με το Arduino, χρησιμοποιούμε ένα τυπικό baud rate των 9600 baud, ενώ άλλες συσκευές έχουν επίσης προκαθορισμένο baud rate.

Παράδειγμα: *Serial.begin(9600); //Ξεκινούμε σειριακή επικοινωνία με ταχύτητα 9600 baud.*

- **Serial.println(value, format):**

Με αυτή τη μέθοδο τυπώνουμε στη σειριακή θύρα με την οποία είμαστε συνδεδεμένοι.
Η τιμή value μπορει να είναι οποιοσδήποτε τύπος δεδομένων.

Η παράμετρος format είναι προαιρετική και μπορούμε να δώσουμε είτε μία αριθμητική βάση (πχ DEC για δεκαδικό, HEX για δεκαεξαδικό) όταν θέλουμε να κωδικοποιήσουμε κάποιον ακέραιο αριθμό, είτε έναν αριθμό δεκαδικών ψηφίων που θέλουμε να τυπώσουμε, όταν εκτυπώνουμε αριθμούς float και double.

Παράδειγμα: *Serial.println(“Hello World!”); //Εκτυπώνουμε το string “Hello World!” στη σειριακή θύρα*

- **int Serial.read():**

Διαβάζουμε το πρώτο byte δεδομένων στη σειριακή θύρα μας. Επιστρέφει -1 αν δεν υπάρχουν δεδομένα προς ανάγνωση. Υπάρχουν επίσης και οι **readString και readFloat** για τους εκάστοτε τύπους δεδομένων που αναμένουμε.

- **int Serial.available():**

Επιστρέφει τον αριθμό των διαθέσιμων bytes προς ανάγνωση στη σειριακή θύρα. Η τυπική χρήση αυτής της μεθόδου είναι μέσα στη void loop. Την τοποθετούμε μέσα σε έναν έλεγχο if για να ελέγξουμε εάν έχουν καταφτάσει δεδομένα προς ανάγνωση στη θύρα μας και να τα επεξεργαστούμε.

Παράδειγμα: *if( Serial.available() > 0) { Serial.read(); } //Ελέγχουμε αν έχουμε νέα δεδομένα στη σειριακή θύρα και αν ναι τα διαβάζουμε*

Για παραπάνω πληροφορίες και επιπλέον μεθόδους που αφορούν τη σειριακή επικονωνία επισκευθείτε: https://www.arduino.cc/reference/en/language/functions/communication/serial/

## State Change / Interrupts

Το Arduino και οι πιο πολλοί μικροελεγκτές διαθέτουν εισόδους που μπορούν να προκαλέσουν interrupts, με την αλλαγή τάσης σε κάποια/ες είσοδο/ους τους. Συγκεκριμένα για το Arduino Uno τα pins 2 και 3 μπορούν να χρησιμοποιηθούν για αυτό το σκοπό. Καλώντας την `void attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)` ορίζουμε μία function να καλείται με την αλλαγή κατάστασης ενός pin, συγκεκριμένα:

- **pin** : Το pin που ελέγχεται
- **ISR** : Η function που καλείται (interrupt service routine)
- **mode** : Μπορεί να πάρει τις τιμές RISING, FALLING, CHANGE προκαλώντας διακοπή όταν η τάση στο pin αλλάζει με τρόπο LOW->HIGH, HIGH->LOW, HIGH<->LOW αντίστοιχα.

Κάθε μεταβλητή που καλείται εντός της ISR πρέπει να είναι ορίζεται ως **volatile**.
Παράδειγμα:

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

Αντίστοιχη λειτουργία επιτυγχάνεται και ελέγχοντας στην loop την κατάσταση μιας εισόδου και ανιχνεύοντας κάποια αλλαγή στην κατάσταση της ως εξής.

```
#define button 2

int buttonState = 0;
int lastButtonState = 0;

void setup(){
  pinMode(button, INPUT);
```

```
}

void loop() {
  buttonState = digitalRead(button);
  if (buttonState != lastButtonState) {
    // code to run
  }
  lastButtonState = buttonState;
}
```

## Χρονοπρογραμματισμός σε μικροελεγκτές

Στους ηλεκτρονικούς υπολογιστές απολαμβάνουμε εδώ και δεκαετίες τις ευκολίες του χρονοπρογραμματισμού. Πολλές διεργασίες εκτελούνται σε πολλαπλούς πυρήνες και το λειτουργικό σύστημα φροντίζει να γίνονται όλα φαινομενικά παράλληλα. Στους μικροελεγκτές δεν υπάρχει η πολυτέλεια ούτε του λειτουργικού συστήματος, των διεργασιών και των πολλαπλών πυρήνων. Ο προγραμματιστής, ο οποίος είναι υπεύθυνος για την υλοποίηση ολόκληρου του λογισμικού του μικροελεγκτή (firmware), αναλαμβάνει και να συντονίσει τις πολλαπλές λειτουργίες του μικροελεγκτή.

**Delay**

Η απλούστερη τεχνική που χρησιμοποιείται για τον χρονισμό είναι η χρονοκαθυστέρηση (delay). Με αυτή προγραμματίζουμε τον μικροελεγκτή να παύσει την εκτέλεση εντολών για συγκεκριμένο χρονικό διάστημα. Το σημαντικότερο μειονέκτημα της μεθόδου είναι πως χάνονται πολλοί κύκλοι της CPU καθώς και η αδράνεια αυτής σημαίνει πως δεν μπορούμε να αξιοποιήσουμε τις περιφερειακές συσκευές. Δέχεται μία παράμετρο, η οποία είναι ο χρόνος που θέλουμε να αδρανοποιηθεί το Arduino σε milliseconds.

*Παράδειγμα: delay(500); //αδρανοποίηση του Arduino για μισό δευτερόλεπτο*

**Millis**

Ο κάθε μικροελεγκτής διαθέτει έναν ή παραπάνω χρονομετρητές (timer - counter). Αυτοί, εν συντομία, μεταβάλλουν τις τιμές τους με βάση τους κύκλους του ρολογιού. Το περιβάλλον προγραμματισμού του Arduino μας προσφέρει τη function **unsigned long millis();** η οποία επιστρέφει τον αριθμό των χιλιοστών του δευτερολέπτου από την έναρξη λειτουργίας του μικροελεγκτή. Η μέγιστη τιμή που μπορεί να αποθηκεύσει ο 32-bit καταχωρητής φτάνει τη $2^{32}$-1 πράγμα που τον κάνει να υπερχειλίζει μετά από περίπου 49 μέρες.

**Παράδειγμα**:

```
int lastMillis = 0; //Εδώ αποθηκεύουμε την τελευταία φορά που εκτελέστηκε η εντολή
int currentMillis = 0;

void loop() {
  currentMillis = millis();
  if(currentMillis - lastMillis  >= 1000) {
    lastMillis = currentMillis;
   //ΕΝΤΟΛΗ
  }
}
```

Ας υποθέσουμε ότι θέλουμε να εκτελέσουμε μια εντολή ανά 1 δευτερόλεπτο, χωρίς να αδρανοποιούμε το Arduino στο ενδιάμεσο.

Μία προσέγγιση με τη millis() είναι να την καλούμε συνεχώς μέσα στη loop και να εκτελούμε έναν απλό έλεγχο που λέει ότι: Αν ο χρόνος που έχει περάσει από την τελευταία φορά που εκτελέστηκε η εντολή είναι ένα δευτερόλεπτο (1000ms)  ή παραπάνω, τότε αποθήκευσε την τωρινή χρονική στιγμή στην μεταβλητή lastMillis και εκτέλεσε την εντολή.

Πρόσθετο υλικό:
https://www.best-microcontroller-projects.com/arduino-millis.html

Για προχωρημένους:
Σε πιο χαμηλού επίπεδου προγραμματισμό, για να χρονίσουμε τις λειτουργίες του μικροελεγκτή μας μπορούμε να αλλάξουμε τις τιμές των Timer Registers και να δημιουργούμε interrupts.
https://circuitdigest.com/microcontroller-projects/arduino-timer-tutorial