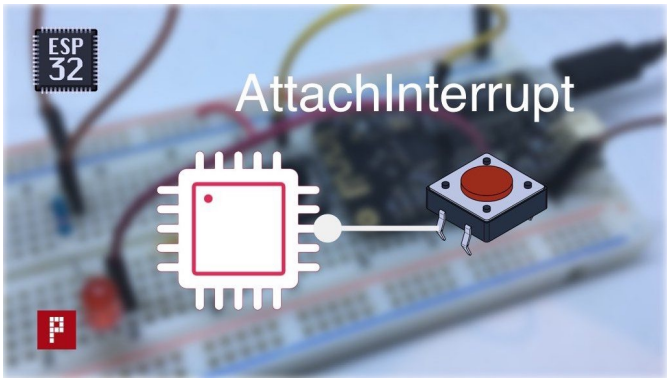
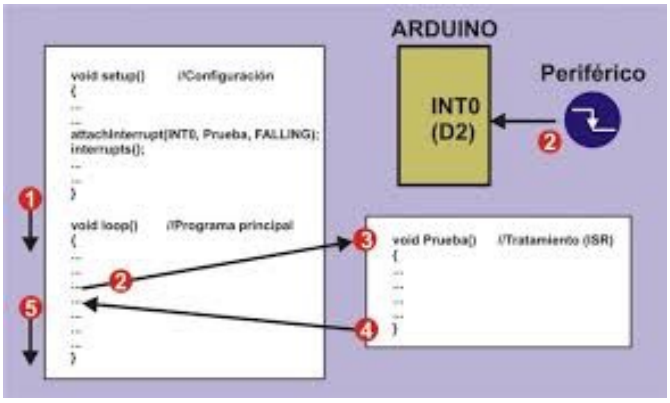
Ο μικροελεγκτής «Arduino Uno» και τα εξωτερικά «Interrupt»





Ο μικροελεγκτής «Arduino Uno» και τα εξωτερικά «Interrupt»

Η υπολογιστική πλατφόρμα Arduino Uno χρησιμοποιεί μέχρι και δύο εξωτερικά interrupt, τα **INT0** και **INT1**, τα οποία εκ κατασκευής είναι ορισμένα στις **ακίδες** (PIN) **2 και 3** αντίστοιχα.

Κάθε φορά που ενεργοποιείται ένα εξωτερικό interrupt, καλείται και εκτελείται άμεσα η συνάρτηση του interrupt (**interrupt service routine**) η οποία δεν χρειάζεται κάποια παράμετρο και δεν επιστρέφει τίποτα.

Μέσα στη συνάρτηση του interrupt η συνάρτηση **delay()** δεν λειτουργεί ενώ οι τιμές της εντολής **millis()** δεν αυξάνονται.

Οι μεταβλητές που **χρησιμοποιούνται μέσα και έξω** από τη συνάρτηση του interrupt πρέπει οριστούν ως **volatile** , που σημαίνει η ανάγνωση της θα γίνει πάντα από την RAM και όχι από τους καταχωρητές του προγράμματος..

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	all digital pins

Επισύναψη interrupt στον κώδικα

attachInterrupt (interrupt, function_name, mode)

- Στη θέση του **interrupt** επιλέγουμε ανάμεσα από το **INT0 (pin2)** και **INT1 (pin3)**
- Στη θέση του **function_name** γράφουμε **το όνομα** της συνάρτησης του interrupt **(Interrupt Service Routine, ISR)**
- Όσον αφορά τον τρόπο λειτουργίας **mode** του interrupt υπάρχουν τέσσερις επιλογές οι οποίες είναι:

  LOW : το interrupt ενεργοποιείται όταν στην ακίδα (PIN) που είναι συνδεδεμένο το interrupt υπάρξει σήμα LOW.

  CHANGE : το interrupt ενεργοποιείται όταν στην ακίδα του interrupt υπάρξει αλλαγή στο σήμα οποιαδήποτε και αν είναι αυτή.

  RISING : το interrupt ενεργοποιείται όταν έχουμε ανερχόμενο μέτωπο δηλαδή, όταν στην ακίδα του interrupt υπάρξει σήμα που μετατρέπεται από LOW σε HIGH.

  FALLING: το interrupt ενεργοποιείται όταν έχουμε κατερχόμενο μέτωπο δηλαδή, στην ακίδα του interrupt υπάρξει σήμα που μετατρέπεται από HIGH σε LOW

Τέλος, κάθε φορά που ενεργοποιείται η ISR interrupt, δεν χρειάζεται κάποιο όρισμα εισόδου ενώ η ρουτίνα δεν επιστρέφει τίποτα απλά κάνει αλλαγές στις μεταβλητές του κώδικα.

Παράδειγμα χρήσης των εξωτερικών Interrupt («Interrupt Service Routine»)

Η χρήση των εξωτερικών Interrupt («Interrupt Service Routine») γίνεται αντιληπτή με τα παρακάτω παραδείγματα, με τα οποία το LED 13 που βρίσκεται πάνω στην **υπολογιστική πλατφόρμα του Arduino** εναλλάσσεται η κατάσταση του («HIGH-LOW») κάθε φορά που ενεργοποιείται το Interrupt 0 (INT0) .

Παράδειγμα 1:

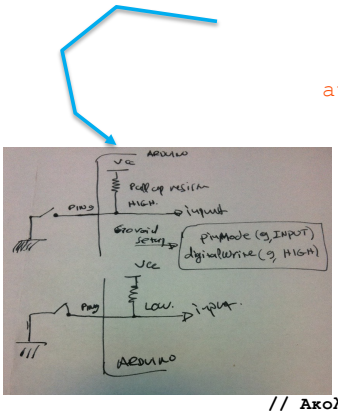
```
// Ακολουθούν  η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
int pin = 13; //δήλωση του LED
volatile int state = LOW;//δήλωση της μεταβλητής state

// Ακολουθεί η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
void setup(){
      pinMode(pin, OUTPUT); //δήλωση του LED ως έξοδο
      attachInterrupt(0, blink, CHANGE); //δήλωση της συνάρτησης του
          }                                //interrupt (INT0)

// Ακολουθεί η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
void loop(){
      digitalWrite(pin, state); //εγγραφή της κατάστασης του LED
          }
// Ακολουθεί η Ρουτίνα εξυπηρέτησης της διακοπής
void blink() {
        state = !state; //αναστροφή της κατάστασης του LED
          }
```

Παράδειγμα 2: έναυση και σβήσιμο Led ελεγχόμενο από εξωτερικό button με την διαδικασία Interrupt (Η διαφορά με την προηγούμενη περίπτωση είναι ότι πρίν η ενεργοποίηση του LED γινόταν μέτα το loop ενώ τώρα άμεσα με την ενεργοποίηση της διακοπής)

```
const int LED = 13; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα
                    //LED και της δίνουμε την τιμή 13 με σκοπό να την
                    //χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που
                    //θα συνδέσουμε το led.
const int BUTTON = 9; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα
                      // BUTTON και της δίνουμε την τιμή 9 με σκοπό να την
                      // χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε
                      // που θα συνδέσουμε το button.
// Ακολουθεί η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
void setup () {
        pinMode (LED, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή LED,
                               // ότι το pin 13 θα λειτουργεί σαν ψηφιακή
                               // έξοδος 0-5V
```




```
        pinMode (BUTTON, INPUT);
    digitalWrite (BUTTON, HIGH); //ενεργοποιώ την internal pull-up resistor
                                 // Ορίζουμε χρησιμοποιώντας την μεταβλητή
                                 // BUTTON, ότι το pin 9
                                 // θα λειτουργεί σαν ψηφιακή είσοδος 0-5V
attachInterrupt (0, ISRpinChange, CHANGE);// Ενεργοποίηση της ρουτίνας
                                          // που θα εξυπηρετεί την
                                          // διακοπή. Η πρώτη παράμετρος
                                          // με τιμή 0 (0 => INT0=> pin 2)μας
                                          // ορίζει την πηγή της εξωτερικής
                                          //διακοπής (για τιμή 1=> INT1=> pin 3).
                                          //Η δεύτερη παράμετρος με
                                          // τιμή ISRpinChange μας ορίζει το
                                          // όνομα της ρουτίνα εξυπηρέτησης
                                          // της διακοπής ISR που δημιουργήσαμε
                                          // παραπάνω. Η τρίτη παράμετρος με
                                          // τιμές LOW, RISING, FALLING, CHANGE μας ορίζει το
                                          // είδος της αλλαγής λογικής στάθμης που ενεργοποιεί
                                          // την διακοπή.
}
// Ακολουθείται η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
```



```
void loop () {
              // Δεν εκτελείται καμία εντολή αρά δεν γίνεται τίποτα και
             // περιμένουμε να συμβεί μία  διακοπή
            }

// Ακολουθεί η Ρουτίνα εξυπηρέτησης της διακοπής
// Interrupt Service Routine (ISR)
// Ακολουθεί η δημιουργία ρουτίνας εξυπηρέτησης διακοπής με όνομα pinChange (μπορεί να
δηλωθεί μετά τις μεταβλητές και πριν την ρουτίνα αρχικών ρυθμίσεων του μικροελεγκτή ή
και μετά την συνάρτηση void loop() όπως κάνουμε εδώ)

void ISRpinChange(){
          noInterrupts (); // Απενεργοποίηση των υπόλοιπων διακοπών με σκοπό
                           // να μην συμβεί κάποια άλλη διακοπή για όσο χρόνο
                           // εξυπηρετείται η παρούσα διακοπή. Αν έχουμε μόνο
                           // μία ISR τότε δεν απαιτείται να γράψουμε αυτήν την
                           // εντολή. Επίσης μέσα στις ρουτίνες ISR δέν
                           //επιτρέπεται η χρήση της συνάρτησης delay().
      if (digitalRead (BUTTON) == HIGH) // Διαβάζουμε την ψηφιακή τιμή
                                        // που έχει το Pin 9 (εκεί που
                                        // συνδέεται το button) και
                                        // ελέγχουμε αν το button είναι
                                        // πατημένο, αν ναι τότε
         digitalWrite (LED, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13,
                                   // δηλαδή άναψε το led
      else digitalWrite (LED, LOW);//Αλλιώς: Βγάλε λογικό 0
                                   //(0V) στο pin 13, δηλαδή σβήσε το led
       interrupts (); // Ενεργοποίηση των διακοπών
           }
```

LOW

Παρατηρήσεις κατά την συγγραφή συναρτήσεων Interrupt Service Routine (ISR):

- Φρόντισε η ISR να είναι μικρή σε μέγεθος
- Μην χρησιμοποιείτε την συνάρτηση delay()
- Μην χρησιμοποιείτε την συνάρτηση Serial.print()
- Κάνε τις μεταβλητές ορατές από τον κύριο κώδικα της void loop ορίζοντάς αυτές ως **volatile**

http://gammon.com.au/interrupts

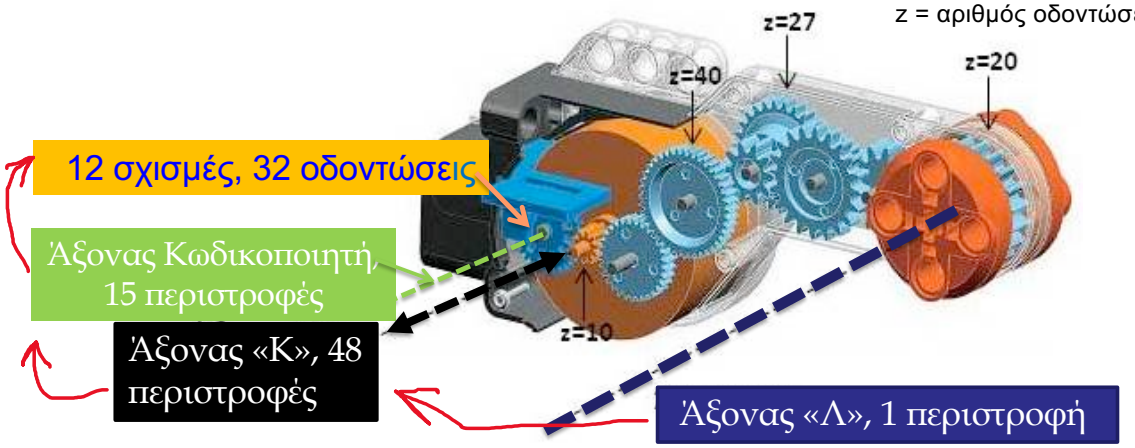
https://www.arduino.cc/en/Reference/AttachInterrupt

85

Εργαστηριακό Μάθημα – Interrupts – (πειραματικό μέρος)

(2 ώρες)

Ο DC κινητήρας της Lego



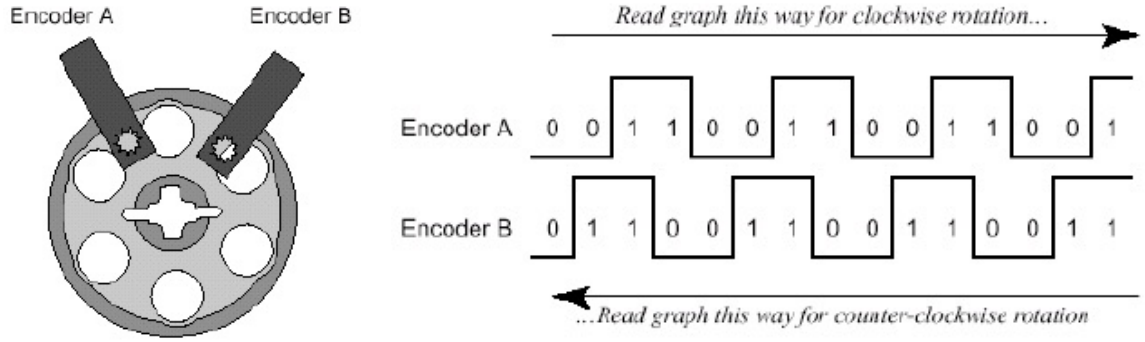
Η **σχέση μετάδοσης** μεταξύ του γραναζιού στο οποίο βρίσκεται ο κωδικοποιητής (z=32) και του γραναζιού που βρίσκεται πάνω στον άξονα «Κ» (z=10) είναι 10:32, άρα για μία περιστροφή του άξονα «Λ»(hub) ο κωδικοποιητής κάνει 48*10:32=**15 περιστροφές.**

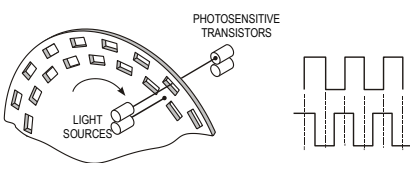
Μία περιστροφή τού άξονα «Λ» αντιστοιχεί σε 48 περιστροφές του άξονα «Κ» του κινητήρα και 15 περιστροφές του άξονα του κωδικοποιητή (encoder). Ο κωδικοποιητής του **NXT** έχει **12 σχισμές** ανά κανάλι, άρα έχουμε **15 * 12 = 180** παλμούς ανα κανάλι για κάθε μία περιστροφή του άξονα «Λ».

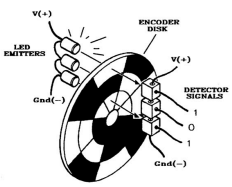
Με την αξιοποίηση των δύο καναλιών μπορούμε να έχουμε **τετραπεριοδική διεύθυνση του κωδικοποιητή («quadrature encoder») και βελτίωση της διακριτοποίησης** με αποτέλεσμα ο μέγιστος αριθμός παλμών που μπορούμε να έχουμε είναι **4 * 180 = 720** παλμοί ανά περιστροφή δηλαδή ακρίβεια γωνιακής θέσης 360deg/720 = **0.5deg**.

Εφαρμογή των interrupt στους κωδικοποιητες Προσαύξησης

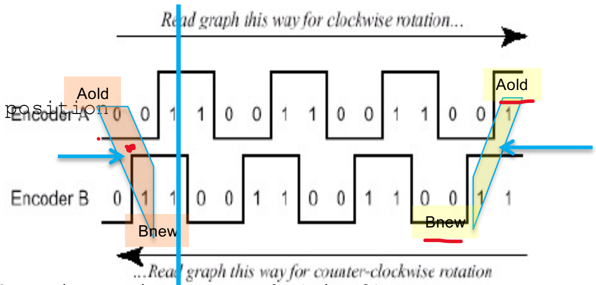
- Η φορα περιστροφης μπορει να ανιχνευθει με τη βοηθεια ενος κωδικοποιητη που εχει 2 φωτοκυτταρα ανιχνευσης στον ιδιο δισκο. Τα φωτοκυτταρα ειναι τοποθετημενα κατα τετοιο τροπο ωστε οι εξοδοι να εχουν **διαφορα φασης 90°** η μια σε σχεση με την αλλη. Η φορα περιστροφης προσδιοριζεται με τη βοηθεια καταλληλου λογικου κυκλωματος το οποιο δεχεται σαν εισοδους τις δυο ακολουθιες παλμων.







int =>16-bit => -32768 μέχρι 32767 => unsigned int : (2^16-1) = 65.535




```
#define encoder0PinA 2
#define encoder0PinB 3
volatile unsigned int encoder0Pos = 32768;
         unsigned int tmp = 32768; //temporary position
         unsigned int Aold = 0;
         unsigned int Bnew = 0;
volatile double angle = 0; // motor's angle
void setup()            {
        pinMode(encoder0PinA, INPUT);
        pinMode(encoder0PinB, INPUT);
attachInterrupt(0, doEncoderA, CHANGE); // encoder pin on interrupt 0 (pin 2)
attachInterrupt(1, doEncoderB, CHANGE); // encoder pin on interrupt 1 (pin 3)
  Serial.begin (115200);}

void loop(){//Check each changes in position
            if (encoder0Pos != tmp) {
                angle = (360.0/720.0)*(encoder0Pos-32768);
                Serial.println(angle); Serial.print("\t");Serial.println(encoder0Pos,DEC);
                tmp = encoder0Pos;    }
 delay(10);}

void doEncoderB(){// Interrupt on B changing state
  Bnew=digitalRead(encoder0PinB);
  Bnew^Aold ? encoder0Pos++:encoder0Pos--;}
//Ternary operator (condition ? Evaluated_when_true : evaluated_when_false)

void doEncoderA(){ // Interrupt on A changing state
Bnew^Aold ? encoder0Pos++ : encoder0Pos--;
Aold=digitalRead(encoder0PinA);
                  }
```

Τελεστής Bitwise ή XOR (^)

XORing the actual PinB state with the previous PinA state, give us the increase or decrease of the encoder count. Few code lines, full encoder precision and achieve the arduino max aquisition rate.

89

http://playground.arduino.cc/Main/RotaryEncoders#Example3