ΕΝΟΤΗΤΑ 2: ΨΗΦΙΑΚΕΣ ΕΙΣΟΔΟΙ/ΕΞΟΔΟΙ ΚΑΙ ΔΙΑΚΟΠΕΣ

ΣΤΟΧΟΙ

Δεν θα μιλήσουμε για το τι είναι ψηφιακές είσοδοι και έξοδοι. Θα υποθέσουμε ότι είστε ήδη εξοικειωμένοι και τις έχετε χρησιμοποιήσει. Αυτό που θα κάνουμε είναι να δούμε μερικές ιδέες για τον έλεγχο ψηφιακών συσκευών και να εξηγήσουμε τι είναι οι pull-up και pull-down αντιστάσεις.

Σε αυτή την ενότητα θα δούμε πιο προσεκτικά το θέμα των διακοπών. Το Arduino είναι σε θέση να αναστείλει ένα πρόγραμμα που βρίσκεται σε εξέλιξη, για να τρέξει ένα άλλο ή να εκτελέσει μια άλλη διεργασία. Μόλις αυτή πραγματοποιηθεί, το Arduino επιστρέφει στο αρχικό πρόγραμμα. Αυτό μπορεί να συμβεί όταν τα pin εισόδου εντοπίσουν συγκεκριμένα σήματα. Ας δούμε ποια είναι αυτά τα σήματα.

ΕΝΟΤΗΤΑ ΘΕΩΡΙΑΣ

- ΨΗΦΙΑΚΕΣ ΕΞΟΔΟΙ; ΕΝΕΡΓΑ ΥΨΗΛΑ ΛΟΓΙΚΑ ΕΠΙΠΕΔΑ
- ΨΗΦΙΑΚΕΣ ΕΙΔΟΔΟΙ; ΑΝΤΙΣΤΑΣΕΙΣ PULL-UP ΚΑΙ PULL-DOWN
- ΔΙΑΚΟΠΕΣ

ΕΝΟΤΗΤΑ ΕΞΑΣΚΗΣΗΣ

- ΠΑΡΑΔΕΙΓΜΑ 1: Ενεργοποίηση LEDs
- ΠΑΡΑΔΕΙΓΜΑ 2: Παρακολούθηση εισόδων
- ΠΑΡΑΔΕΙΓΜΑ 3: Παρακολούθηση εισόδων με pull-up αντιστάσεις
- ΠΑΡΑΔΕΙΓΜΑ 4: Παρακολούθηση εισόδων χωρίς διακοπές
- ΠΑΡΑΔΕΙΓΜΑ 5: Παρακολούθηση εισόδων με διακοπές
- ΠΑΡΑΔΕΙΓΜΑ 6: Έλεγχος δυο διακοπών

ΥΛΙΚΟ ΕΞΑΣΚΗΣΗΣ

- Η/Υ φορητός ή επιτραπέζιος

-Περιβάλλον εργασίας Arduino IDE. Αυτό θα πρέπει να περιλαμβάνει το συμπληρωματικό υλικό που έχει ήδη εγκατασταθεί και διαμορφωθεί.

-Ελεγκτής Arduino UNO

-Καλώδιο USB

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΝΟΤΗΤΑ ΘΕΩΡΙΑΣ

1. ΨΗΦΙΑΚΟΙ ΕΞΟΔΟΙ, ΕΝΕΡΓΑ ΥΨΗΛΑ ΛΟΓΙΚΑ ΕΠΙΠΕΔΑ

Μπορεί να φαίνεται λίγο δύσκολο αλλά στη πραγματικότητα δεν είναι και τόσο, για να καταλάβουμε για το τι θα συζητήσουμε σε αυτό το κομμάτι. Υποθέτουμε ότι έχετε επαφή με συναρτήσεις που σχετίζονται με τις λειτουργίες ψηφιακών εξόδων. Ας τις ξαναδούμε λίγο:

- **pinMode(n,OUTPUT)**: Μας επιτρέπει να διαμορφώσουμε ένα pin D0:D13 ως έξοδο.
- **digitalWrite(n,LOW)**: Στέλνει ένα λογικό επίπεδο "0" από 0 V μέσω του pin n που υποδεικνύεται.
- **digitalWrite(n,HIGH)**: Στέλνει ένα λογικό επίπεδο «1» από +5 volt μέσω του pin n που υποδεικνύεται.

Τώρα φανταστείτε ότι θέλετε να ελέγξετε τον φωτισμό ενός LED που είναι συνδεδεμένο στην ψηφιακή έξοδο pin D4. Σίγουρα θα γράψετε μια σειρά από συναρτήσεις κάπως έτσι:

```
pinMode(4,OUTPUT);          // Θέσε το pin D4 ως έξοδο

digitalWrite(4,HIGH);       // στείλε το λογικό επίπεδο “1” μέσω του pin D4
```

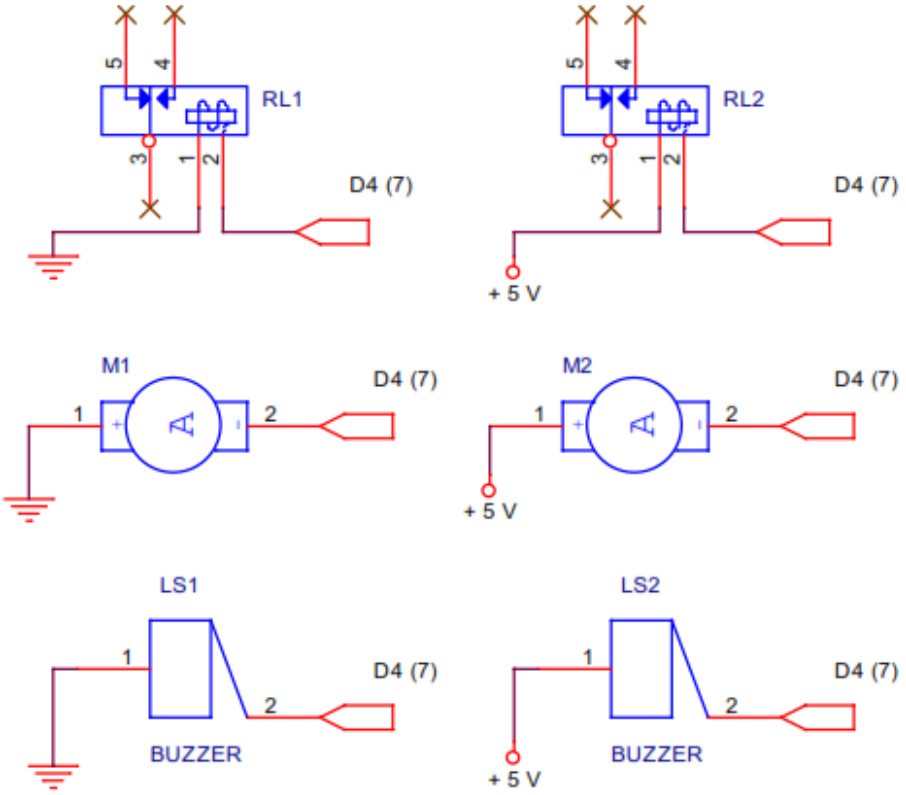
Γιατί πάντα λέμε να ορίσουμε ένα επίπεδο «1» (+5 V) κάθε φορά που θέλουμε να ενεργοποιήσουμε κάτι; Χρησιμοποιούμε το επίπεδο «0» (0 V) όταν θέλουμε μόνο να απενεργοποιήσουμε κάτι ; Δεν μπορούμε να το κάνουμε αντίστροφα; Και όμως μπορούμε ! Και τα δύο επίπεδα «1» και «0» είναι εξίσου αντιπροσωπευτικά. Ρίξτε μια ματιά στα διαγράμματα της Εικόνας 1. Τι είδους λογικό επίπεδο θα έπρεπε να στείλουμε μέσω του σήματος D4 (pin 7) για να ενεργοποιήσουμε το LED στο κύκλωμα στα αριστερά;; Και τι γίνεται με το LED στα δεξιά;

Εικόνα 1

Γνωρίζοντας ότι η άνοδος πρέπει να είναι θετική σε σχέση με την κάθοδο για να ανάψει το LED, πρέπει να στείλουμε «1» (+5 V) μέσω του D4 (pin 7) στην άνοδο του κυκλώματος στα αριστερά. Αυτό συμβαίνει γιατί η κάθοδος είναι συνδεδεμένη στο GND (0 V) μέσω της αντίστασης απορρόφησης. Παρ 'όλα αυτά θα πρέπει να στείλουμε «0» (0 V) μέσω του D4 στην κάθοδο του κυκλώματος στα δεξιά, επειδή η άνοδος είναι συνδεδεμένη στα +5 V μέσω της αντίστασης.

Μπορεί να έχουμε να κάνουμε με ένα LED , αλλά το ίδιο ισχύει και για άλλες συσκευές όπως πηνία, κινητήρες, buzzers κ.τ.λ. Ρίξτε μια προσεκτική ματιά στα διαγράμματα που ακολουθούν.

Υπάρχει ένα πράγμα που θα θέλαμε να επισημάνουμε εδώ: μπορεί να προσέξατε ότι η κατεύθυνση περιστροφής του κινητήρα αλλάζει όταν τον ενεργοποιείτε με επίπεδο «1» αντί με επίπεδο «0».

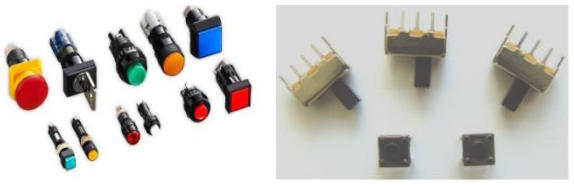


Εικόνα 2

Συνοψίζοντας: υπάρχουν περιφερειακά εξόδου που ενεργοποιούνται όταν δώσουμε επίπεδο «1» σε αυτά και λέγονται «ενεργά υψηλού σήματος». Υπάρχουν επίσης περιφερειακά εξόδου που ενεργοποιούνται όταν δώσουμε επίπεδο «0» σε αυτά. Θα συναντήσετε και τους δύο τύπους. Εξαρτάται κυρίως από το πως τα συγκεκριμένα περιφερειακά είναι συνδεδεμένα (Εικόνα 2).

2. ΨΗΦΙΑΚΟΙ ΕΙΣΟΔΟΙ, PULL-UP ΚΑΙ PULL-DOWN ΑΝΤΙΣΤΑΣΕΙΣ

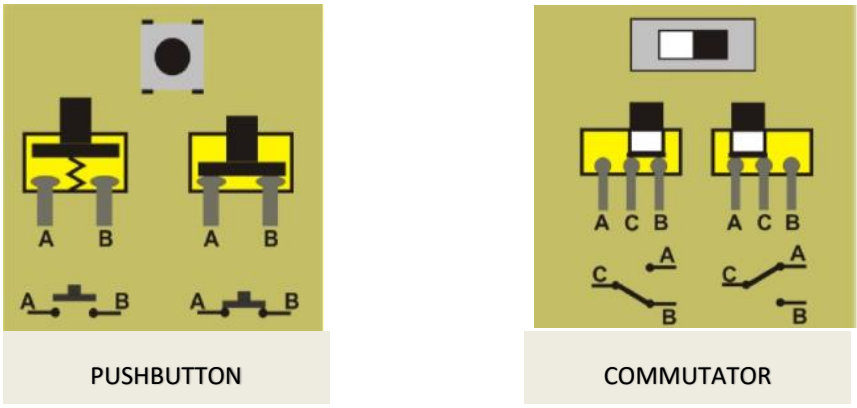
Μπορεί να νομίζετε ότι γνωρίζετε τα πάντα για τις ψηφιακές εισόδους, αλλά θα εξετάσουμε ένα πρόβλημα που ίσως αντιμετωπίσατε, αλλά δεν γνωρίζετε γιατί. Για αρχή, να σας υπενθυμίσουμε ότι τα πιο βασικά και οικονομικά περιφερειακά εισόδου είναι τα απλά κουμπιά και οι διακόπτες. Θα τα συναντήσετε σε όλα τα μεγέθη και σχήματα (Εικόνα 3). Κάποια είναι σχεδιασμένα για βιομηχανική χρήση σε μηχανήματα, σε πίνακες ελέγχου κ.τ.λ



Εικόνα 3

Υπάρχουν άλλα που είναι πολύ πιο απλά και πιο οικονομικά. Χρησιμοποιούνται ευρέως στα σπίτια, στην εκπαίδευση, για να κολλήσουν υλικά σε τυπωμένα κυκλώματα, σε μικροεφαρμογές κ.τ.λ. Θα χρησιμοποιήσουμε μερικά μικρά κουμπιά πίεσης όπως αυτά που βλέπετε στην Εικόνα 3.

Σε κάθε περίπτωση, δεν έχει σημασία για ποιο λόγο χρησιμοποιούνται διότι ο τρόπος που λειτουργούν είναι πολύ απλός. Όταν ενεργοποιούνται ένα μεταλλικό έλασμα ανοίγει και κλείνει μια ηλεκτρική επαφή ανάμεσα σε δύο ή και περισσότερα pins μέσα στη συσκευή. Λέμε ότι ένας διακόπτης ή ένας μεταγωγέας είναι "ενσωματωμένος". Όταν τον ενεργοποιούμε , ο διακόπτης μένει εκεί μέχρι να τον μετακινήσουμε ξανά. Οι διακόπτες φωτισμού λειτουργούν έτσι αλλά τα κουμπιά πίεσης που θα δούμε δεν λειτουργούν με αυτό τον τρόπο. Τα κουμπιά πίεσης ανοίγουν και κλείνουν ένα ή περισσότερα κυκλώματα μόνο όταν είναι ενεργά. Έπειτα, όταν τα αφήσουμε επιστρέφουν στην αρχική τους θέση, όπως για παράδειγμα το κουδούνι της πόρτας στα σπίτια. Μπορείτε να δείτε ένα διάγραμμα ενός συρόμενου μεταγωγέα που αλλάζει θέση και ενός κουμπιού πίεσης στην Εικόνα 4. Μπορείτε επίσης να δείτε και τα αντίστοιχα ηλεκτρολογικά σύμβολα τους.



Εικόνα 4

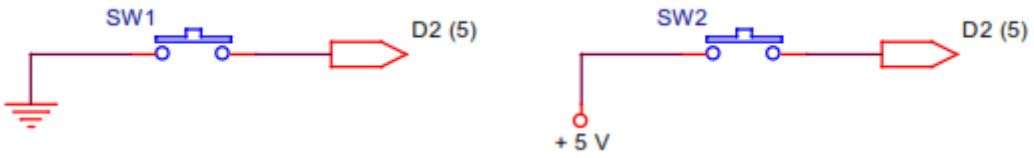
Ο μεταγωγέας στην Εικόνα 4 έχει τρία pins, με ένα από αυτά να είναι κοινό με τα άλλα δύο. Όταν σύρουμε τον διακόπτη στα δεξιά το έλασμα κλείνει το κύκλωμα μεταξύ των pins B και C. Το κύκλωμα μεταξύ των pins C-A παραμένει ανοικτό αλλά όχι συνδεδεμένο. Αν σύρουμε το διακόπτη στα αριστερά το κύκλωμα C-A κλείνει και το C-B παραμένει ανοικτό.

Το κουμπί πίεσης έχει δύο pins , τα A και B. Λέμε ότι όταν βρίσκεται στην θέση ηρεμίας είναι συνήθως ανοικτό. Τα pins δεν είναι συνδεδεμένα. Όταν το πιέσουμε κλείνει το κύκλωμα και τα δύο pins είναι συνδεδεμένα. Όταν το αφήσουμε ένα ελατήριο σπρώχνει πάνω το μεταλλικό έλασμα στη θέση ηρεμίας. Πληροφορικά να αναφέρουμε ότι υπάρχουν κουμπιά πίεσης που είναι κλειστά και ανοίγουν το κύκλωμα μόνο όταν τα πιέσουμε. (Εικόνα 4).

Σίγουρα γνωρίζετε τις παρακάτω συναρτήσεις για να ελέγχετε τις ψηφιακές εισόδους:

- **pinMode(n,INPUT)**: Μας επιτρέπει να ρυθμίσουμε οποιοδήποτε από τα pins D0:D13 ως εισόδους. Στην πραγματικότητα, δεν χρειάζεται πραγματικά να κάνετε τίποτα: είτε συνδέετε την παροχή ρεύματος είτε πατάτε το κουμπί RESET, κάθε φορά που εκκινείτε το σύστημα, όλα τα pins τίθενται ως είσοδοι.
- **digitalRead(n)**: Διαβάζει το λογικό επίπεδο του pin n.

Παρ 'όλα αυτά, δείτε τα παρακάτω διαγράμματα κυκλωμάτων (Εικόνα 5). Ένα κουμπί πίεσης έχει συνδεθεί στο D2 (pin 5) το οποίο έχει προφανώς ρυθμιστεί ως είσοδος.



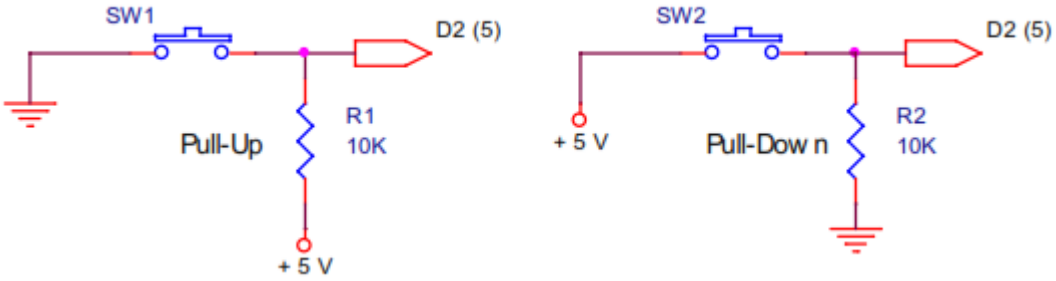
Εικόνα 4

Είναι ξεκάθαρο ότι κάθε φορά που πιέζουμε το κουμπί στο κύκλωμα στα αριστερά, το pin D2 συνδέεται με την GND (0 V) και επομένως είναι στο επίπεδο «0». Το διάγραμμα στα δεξιά είναι ακριβώς το αντίθετο. Όταν πιέσουμε το κουμπί συνδέεται με τα +5 V και επομένως είναι στο επίπεδο «1».

Τι συμβαίνει όμως στα pins αν δεν πιέσουμε κανένα από τα κουμπιά στα δύο κυκλώματα ; Πιθανόν να λέγατε «το αντίθετο από αυτό που γίνεται όταν τα πιέζουμε». Με άλλα λόγια, αν δεν πιέσουμε το κουμπί στο κύκλωμα στα αριστερά, το pin D2 παραμένει στο επίπεδο «1» και στο κύκλωμα στα δεξιά παραμένει στο επίπεδο «0».

Λάθος! Ας το δούμε πιο προσεκτικά. Αν δεν πιέσουμε το κουμπί σε κανένα από τα δύο κυκλώματα, ποια είναι η διαφορά όσον αναφορά το pin D2; Απολύτως καμία. Το D2 παραμένει αποσυνδεδεμένο και στις δυο περιπτώσεις. Λέμε ότι «επιπλέει». Ποιο κερδίζει; Το επίπεδο «1» ή το επίπεδο «0»; Δεν υπάρχει σαφής απάντηση, κάποιες φορές επικρατεί το «1» και κάποιες φορές το «0». Είναι κατανοητό ότι δεν πρόκειται για ιδανική κατάσταση.

Το καλύτερο που μπορεί να γίνει είναι να προσθέσουμε μια αντίσταση στο κύκλωμα. Αν είναι συνδεδεμένη στο θετικό άκρο +5V λέγεται αντίσταση «***PULL-UP***», ενώ αν είναι συνδεδεμένη στο GND ή 0V λέγεται αντίσταση «***PULL-DOWN***». Δείτε τα παρακάτω κυκλώματα (Εικόνα 6).



Εικόνα 5

Αυτό στα αριστερά, με την αντίσταση pull-up R1 κρατά το σήμα του D2 (pin 5) στο επίπεδο «1» (+5 V) όταν το κουμπί SW1 δεν πιέζεται, ή με άλλα λόγια, είναι ανοικτό. Όταν το pin D2 πιέζεται, πηγαίνει στο επίπεδο «0». Στο κύκλωμα στα δεξιά, η pull-down αντίσταση R2 κρατά το pin στο επίπεδο «0» (GND) όταν το κουμπί SW2 δεν πιέζεται. Όταν το D2 pin πιέζεται τότε πηγαίνει στο επίπεδο «1».

Εντάξει, αλλά τι τιμή βάζουμε στις αντιστάσεις; Δεν έχει τόσο σημασία. Θυμηθείτε ότι κάθε φορά που πιέζετε το κουμπί, υπάρχει ροή ηλεκτρονίων ανάμεσα στην GND και το άκρο με τα +5 V, ή, με άλλα λόγια, ρεύμα (I) διαρρέει το κύκλωμα. Πρέπει να προσπαθούμε να ελαχιστοποιούμε το ρεύμα ή την κατανάλωση. Για παράδειγμα, αν βάλουμε αντίσταση 100Ω στην R1 ή στην R2, η κατανάλωση θα είναι ως εξής:

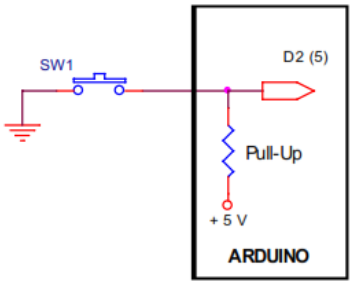
$$I = \frac{V}{R} = \frac{5}{100} = 0.05\,A = 50\,mA$$

Αν διαλέξουμε αντίσταση 10ΚΩ, όπως και στο παράδειγμα, η κατανάλωση θα είναι ως εξής:

$$I = \frac{V}{R} = \frac{5}{10000} = 0.0005\,A = 0,5\,mA$$

Αλλά, να είστε προσεκτικοί, δεν πρέπει να το παρακάνετε. Μπορεί να μπείτε στον πειρασμό να χρησιμοποιήσετε μια πολύ μεγάλη αντίσταση για να μειώσετε στο ελάχιστο την κατανάλωση. Αυτό που θα συμβεί όμως διαλέγοντας μια υψηλή τιμή, είναι ότι το pin D2 θα απομονωθεί από το άκρο των +5V (στην περίπτωση της αντίστασης pull-up) ή από την GND (στην περίπτωση της αντίστασης pull-down) και δεν θα είναι σταθερό. Αν αυτό συμβεί είστε εκεί που ξεκινήσατε, επομένως είναι καλύτερα να μην χρησιμοποιούσατε αντίσταση καθόλου. Συνήθεις τιμές των αντιστάσεων κυμαίνονται μεταξύ 4700 Ω και 10 ΚΩ.

Για άλλη μια φορά θα πρέπει να πούμε ότι το ενεργό επίπεδο όταν πιέζουμε το κουμπί δεν θα πρέπει απαραίτητα να είναι «1», μπορεί να είναι «0». Εξαρτάται από το πως θα συνδέσετε το κουμπί πίεσης.



Εικόνα 6

Πως μας βοηθά το Arduino; Μας παρέχει την αντίσταση pull-up και δεν χρειάζεται να την εγκαταστήσουμε εξωτερικά στα κυκλώματά μας. Δείτε στην Εικόνα 7 ότι η αντίσταση είναι ενσωματωμένη στον ελεγκτή Arduino.

Μπορείτε να ρυθμίσετε αυτή την επιλογή με τη χρήση αυτής της συνάρτησης:

- **pinMode (n, INPUT_PULLUP)**: Όπου το n είναι το pin εισόδου που θέλουμε να συνδέσουμε με την αντίστοιχη αντίσταση pull-up.

Μπορείτε να το κάνετε αυτό με οποιοδήποτε pin από τα D0:D13 που πρόκειται να ρυθμιστούν ως είσοδοι. Δεν έχει νόημα να το κάνουμε αυτό για τις εξόδους και επίσης δεν έχετε τη δυνατότητα να ρυθμίσετε μια είσοδο ως αντίσταση pull-down.

Ίσως να νομίζετε ότι δεν είναι σημαντικό να εξοικονομήσετε μια αντίσταση εισόδου μιας και κοστίζουν μόλις μερικά cents. Αλλά σκεφτείτε το λίγο καλύτερα, αν το έργο σας σας χρειάζεται πολλές pull-up αντιστάσεις όχι μόνο θα θέλατε να τις εξοικονομήσετε αλλά επίσης:

- Θα εξοικονομήσετε χώρο αφού δεν θα χρειάζεται να της αποθηκεύετε κάπου.
- Θα έχετε εξοικονόμηση στο μέγεθος των τυπωμένων πλακετών των κυκλωμάτων. Πωλούνται με το τετραγωνικό εκατοστό και όσο περισσότερα συστατικά μέρη υπάρχουν τόσο μεγαλύτερη είναι η πλακέτα και επομένως τόσο περισσότερα πληρώνετε.
- Θα εξοικονομήσετε χρόνο στην σχεδίαση των κομματιών για την τυπωμένη πλακέτα του κυκλώματος. Όσα περισσότερα υλικά χρησιμοποιείτε τόσο περισσότερο χρόνο θέλετε για την σχεδίαση και όπως είναι γνωστό ο χρόνος είναι χρήμα.
- Θα εξοικονομήσετε χρόνο για την συναρμολόγηση την πλακέτας του κυκλώματος. Όσο περισσότερα υλικά χρησιμοποιείτε, τόσο περισσότερο χρόνο θα χρειαστείτε για την συναρμολόγηση.

Αν πολλαπλασιάσετε όλα αυτά με τον αριθμό των πλακετών κυκλώματος ή των kits που θέλετε να πουλήσετε η εξοικονόμηση θα μπορούσε να είναι δεκάδες η εκατοντάδες ευρώ.

Τώρα που ξέρετε πως να χρησιμοποιήσετε την συνάρτηση pull-up, κάντε το όποτε θεωρείτε ότι είναι απαραίτητο.

3. ΔΙΑΚΟΠΕΣ (INTERRUPTS)

Τώρα ας κοιτάξουμε ένα άλλο θέμα. τις διακοπές. Με αυτόν τον όρο ίσως νομίζετε πως θα διακοπεί η λειτουργία του ελεγκτή και η παύση του θα οδηγήσει στον διακοπή εκτέλεσης του προγράμματος, αλλά στην πραγματικότητα δεν έχει καμία σχέση με αυτό.

Φανταστείτε ένα μηχανικό εργαλείο που φτιάχνει εξαρτήματα. Ξαφνικά χτυπά συναγερμός για έναν από τους παρακάτω λόγους :

- Ένα κομμάτι δεν έχει τοποθετηθεί σωστά και ως αποτέλεσμα το τελικό προϊόν θα είναι ελλαττωματικό.
- Ο κινητήρας που οδηγεί την ταινία μεταφοράς έχει σπάσει και τα κομμάτια συσσωρεύονται.
- Ένας αισθητήρας ανιχνεύει ότι ένα εξάρτημα της μηχανής υπερθερμαίνεται.
- Ένας εργάτης τοποθετεί το χέρι του μέσα στην μηχανή.

Τι πρέπει να κάνει ο ελεγκτής ώστε να σταματήσει μια τέτοια δυσλειτουργία;

α) Τίποτα το ιδιαίτερο, να συνεχίσει να εκτελεί ως συνήθως το πρόγραμμα σαν να μην είχε συμβεί τίποτα.

β) Να κάνει παύση της λειτουργίας και να σταματήσει να εκτελεί το πρόγραμμα αμέσως.

γ) Να σταματήσει το εκτελούμενο πρόγραμμα και να εκτελέσει ένα άλλο που να αντιμετωπίζει τις αιτίες του προβλήματος που ενεργοποίησαν τον συναγερμό, εξετάζοντας τις προειδοποιήσεις και να προβαίνει: στην αφαίρεση ελλαττωματικών προϊόντων, στον τερματισμό της κατασκευής εξαρτημάτων, την ενεργοποίηση ενός συστήματος ψύξης, να δώσει στον εργάτη ένα προειδοποιητικό σήμα, κ.τ.λ.

Η απάντηση είναι προφανής, σωστά; Μια διακοπή δεν σημαίνει απαραίτητα ότι ο ελεγκτής θα σταματήσει εντελώς, απλά σταματά κάποια διεργασία που τρέχει εκείνη την στιγμή για να εκτελέσει μια άλλη.

Πως προκαλούμε μια διακοπή; Υπάρχουν πολλά διαφορετικά συμβάντα που μπορούν να πυροδοτήσουν μια διακοπή. Εξαρτάται από το μοντέλο του ελεγκτή που εξετάζουμε. Ο πιο κοινός τρόπος είναι μέσω εξωτερικών περιφερειακών που στέλνουν σήματα μέσω συγκεκριμένων pin του ελεγκτή. Στην δική μας περίπτωση, το Arduino UNO έχει δύο διαφορετικά pins ως πηγές διακοπών, τα D2/INT0 και D3/INT1.

Σίγουρα θυμόσαστε ότι έχετε χρησιμοποιήσει αυτά τα pins σε μερικές περιπτώσεις και ποτέ δεν προκάλεσαν καμιά διακοπή. Στην πραγματικότητα είναι ψηφιακοί έξοδοι/είσοδοι όπως όλα τα υπόλοιπα pins D0:D13. Το θέμα είναι ότι τα pins D2 και D3 για να δουλέψουν ως INT0 και INT1 είσοδοι διακοπής, πρέπει να ρυθμιστούν κατάλληλα. Η εκκίνηση μιας διακοπής δεν είναι απλά μια λεπτομέρεια. Και αυτό ακριβώς θα μάθετε σε αυτό το κεφάλαιο.

Τι συμβαίνει όταν ένας ελεγκτής λάβει μια διακοπή; Ρίξτε μια ματιά στην Εικόνα 8, δείχνει πως δουλεύει η διαδικασία:

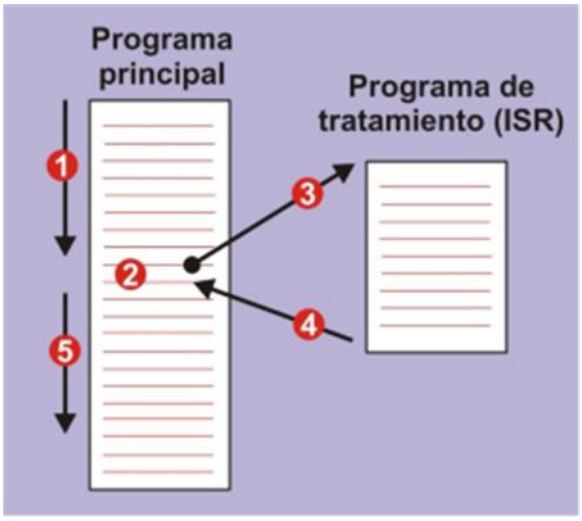
1. Ο ελεγκτής εκτελεί ως συνήθως το κυρίως πρόγραμμα.

2. Σε κάποιο σημείο η αίτηση ενός περιφερειακού προκαλεί μια διακοπή.

3. Ο ελεγκτής αδρανοποιεί το κυρίως πρόγραμμα και εκτελεί ένα πρόγραμμα που αντιμετωπίζει το γεγονός, γνωστό και ως ISR (Interrupt Service Routine).

4. Μόλις τελειώσει η εκτέλεση του ISR, ο ελεγκτής επιστρέφει στην εκτέλεση του κυρίως προγράμματος.

5. Η εκτέλεση συνεχίζεται από το σημείο που σταμάτησε.



Εικόνα 8

Για την διαχείριση και την χρήση των διακοπών του Arduino UNO, υπάρχουν τέσσερις συναρτήσεις στη διάθεσή σας:

Συνάρτηση attachInterrupt()

Ρυθμίζει την λειτουργία μιας διακοπής.

Σύνταξη:

attachInterrupt(pin, ISR, modo);

pin: Αναπαριστά το pin που θα ρυθμιστεί ως είσοδος διακοπής. Σε αυτή την περίπτωση του Arduino UNO μπορεί να είναι το INT0(D2) ή το INT1(D3).

ISR: Αυτό είναι το όνομα της συνάρτησης ρουτίνας που πρέπει να εκτελείται κάθε φορά που συμβαίνει μια διακοπή.

mode: Είναι για το πότε θα έπρεπε να ενεργοποιηθεί μια διακοπή.

LOW: όταν το pin δώσει επίπεδο «0».

CHANGE: όταν εντοπιστεί μια αλλαγή της κατάστασης του pin.

RISING: όταν ένα pin εντοπίσει αύξηση («0» -> «1») .

FALLING: όταν ένα pin εντοπίσει πτώση («1» -> «0»).

Συνάρτηση detachInterrupt()

Απενεργοποιεί μια διακοπή.

Σύνταξη:

detachInterrupt(pin);

pin: Αναπαριστά το pin διακοπής που θα απενεργοποιηθεί. Στην περίπτωση του ARDUINO UNO μπορεί να είναι το INT0 (D2) ή το INT1 (D3).

Συνάρτηση interrupts()

Ενεργοποιεί τις διακοπές. Σκεφτείτε το σαν γενική εξουσιοδότηση για τις διακοπές που έχουν προηγουμένως ρυθμιστεί μέσω της attachInterrupt().

Σύνταξη:

interrupts();

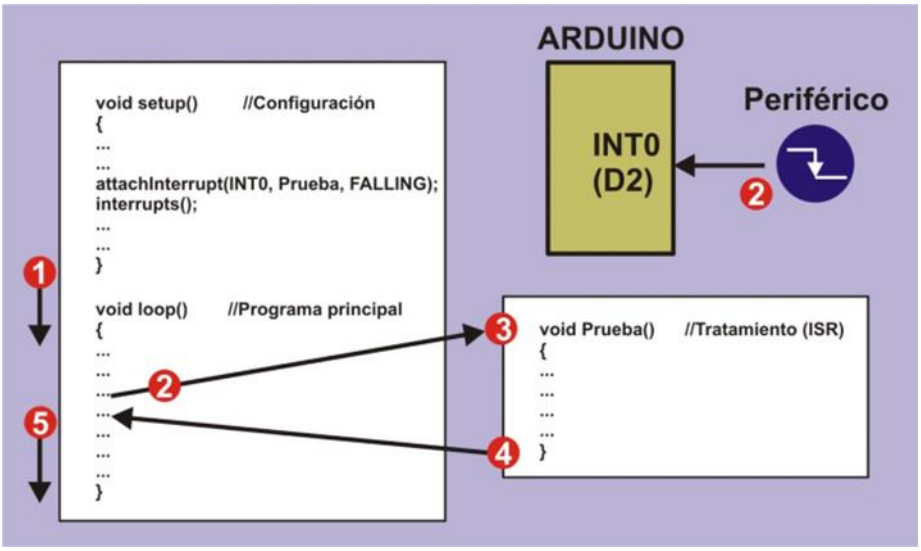
Συνάρτηση noInterrupts()

Απενεργοποιεί όλες τις διακοπές. Σκεφτείτε το κάτι σαν γενική απαγόρευση που σταματά όλες τις διακοπές από το να τεθούν σε λειτουργία.

Σύνταξη:

noInterrupts();

Η Εικόνα 9 μπορεί να σας δώσει μια ιδέα για το πως μοιάζει ένα πρόγραμμα που χρησιμοποιεί διακοπές.



Εικόνα 9

Κάθε φορά που ανιχνεύεται πτώση άκρου, το pin D2 (INT0) ρυθμίζει μια διακοπή μέσω της συνάρτησης setup() με τη χρήση της attachInterrupt(). Αν αυτό συμβεί, θα εκτελεστεί η συνάρτηση Test() που αντιστοιχεί στο ISR. Σχεδόν αμέσως η interrupts() δίνει γενική εξουσιοδότηση. Συμβαίνουν τα εξής:

1. Αρχίζει να εκτελεί το κυρίως πρόγραμμα **loop()**.

2. Σε κάποιο σημείο ένα περιφερειακό αναφέρει μια αίτηση διακοπής με μια πτώση άκρου μέσω του pin INT0 (D2). Ο ελεγκτής σταματά την εκτέλεση του προγράμματος.

3. Ο ελεγκτής εκτελεί την συνάρτηση Test() (Interrupt Service Routine).

4. Μόλις η συνάρτηση Test() έχει εκτελεστεί ο ελεγκτής επιστρέφει στο κυρίως πρόγραμμα.

5. Ο ελεγκτής συνεχίζει την εκτέλεση του προγράμματος από εκεί που σταμάτησε.

Περιορισμοί

Εξαιτίας της φύσης του Arduino ,το σύστημα διακοπών που έχει είναι κάπως περιορισμένο. Για παράδειγμα οι γνωστές συναρτήσεις delay() και millis() δεν λειτουργούν αν βρίσκονται ήδη σε ρουτίνα διακοπής. Αυτό συμβαίνει γιατί αυτές οι συναρτήσεις έχουν δικά τους εσωτερικά συστήματα διακοπών και το Arduino δεν μπορεί να ανταποκριθεί σε διακοπή από άλλο σύστημα.

Δεν μπορείτε να μεταφέρετε παραμέτρους σε μια ρουτίνα διακοπής και το ISR δεν μπορεί να επιστρέψει τίποτα. Αν όμως πρέπει να το κάνετε, θα πρέπει να χρησιμοποιήσετε global μεταβλητές για να μεταφέρετε τα δεδομένα που χρησιμοποιούνται από το κυρίως πρόγραμμα και την συνάρτηση ρουτίνας διακοπής.

Να θυμάστε ότι μια διακοπή είναι ένα γεγονός που μπορεί ή δεν μπορεί να συμβεί στη πάροδο του χρόνου. Αυτές οι καταστάσεις συναγερμού είτε μπορεί να συμβούν είτε όχι. Ένα περιφερειακό μπορεί να στείλει ένα σήμα διακοπής, αλλά αυτό δεν είναι απαραίτητο ότι θα συμβεί.

Είναι καλή ιδέα να εκτελείτε μια ρουτίνα διακοπής όσο πιο γρήγορα γίνεται. Να θυμάστε ότι κατά τη διάρκεια εκτέλεσης μιας συνάρτησης ρουτίνας διακοπής , ο ελεγκτής αναστέλλει την εκτέλεση του κυρίως προγράμματος.

Πλεονεκτήματα

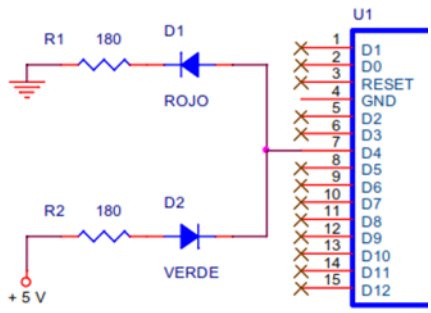
Υπάρχουν βέβαια και πλεονεκτήματα. Γενικότερα, ένα πρόγραμμα που χρησιμοποιεί διακοπές είναι πολύ πιο αποτελεσματικό. Θα πρέπει όμως να προσέχετε πως τις χρησιμοποιείτε. Φανταστείτε ότι κάνετε μια εργασία και πρέπει να φέρετε εις πέρας μια συγκεκριμένη διεργασία κάθε φορά που ένα περιφερειακό στέλνει ένα σήμα. Έχετε δύο επιλογές: η μία είναι μέσω της συνάρτησης digitalRead(), που θα διαβάσει και θα περιμένει για ένα σήμα για να ενεργοποιηθεί. Να θυμάστε όμως ότι όσο περιμένετε για το σήμα δεν μπορείτε να κάνετε τίποτα άλλο.

Η άλλη λύση -που είναι και η καλύτερη- είναι να κάνετε χρήση μιας διακοπής. Το κυρίως πρόγραμμα σας μπορεί να κάνει ότι θέλετε, αλλά μόλις το σήμα διακοπής φτάσει τότε και μόνο τότε ο ελεγκτής ανταποκρίνεται στην διακοπή και εκτελεί την ανάλογη διεργασία. Αυτό σημαίνει ότι δεν υπάρχει χαμένος χρόνος διότι ο ελεγκτής πάντα θα κάνει κάτι χρήσιμο σαν να έκανε δυο διεργασίες ταυτόχρονα.

ΕΝΟΤΗΤΑ ΕΞΑΣΚΗΣΗΣ

4. ΠΑΡΑΔΕΙΓΜΑ 1: ΕΝΕΡΓΟΠΟΙΗΣΗ LEDS

Δεν πρόκειται να μάθετε κάτι καινούριο σχετικά με προγραμματισμό μέσω αυτού του παραδείγματος. Μέχρι τώρα ξέρετε πως να ανάβετε και να σβήνετε τα LEDs. Αυτό που θα μάθετε εδώ είναι πως θα κάνετε την ηλεκτρονική εγκατάσταση να ελέγχει ότι ένα LED ή άλλα περιφερειακά μπορούν να ενεργοποιηθούν αφού λάβουν ένα λογικό επίπεδο «0» ή ένα λογικό επίπεδο «1». Δείτε το διάγραμμα στην Εικόνα 10.



Εικόνα 10

Το ίδιο pin εξόδου D4 θα δρα και στα δύο LEDs. Παρατηρείστε ότι είναι ή άνοδος του κόκκινου LED (ROJO) με την κάθοδο να συνδέεται στην γείωση (GND).

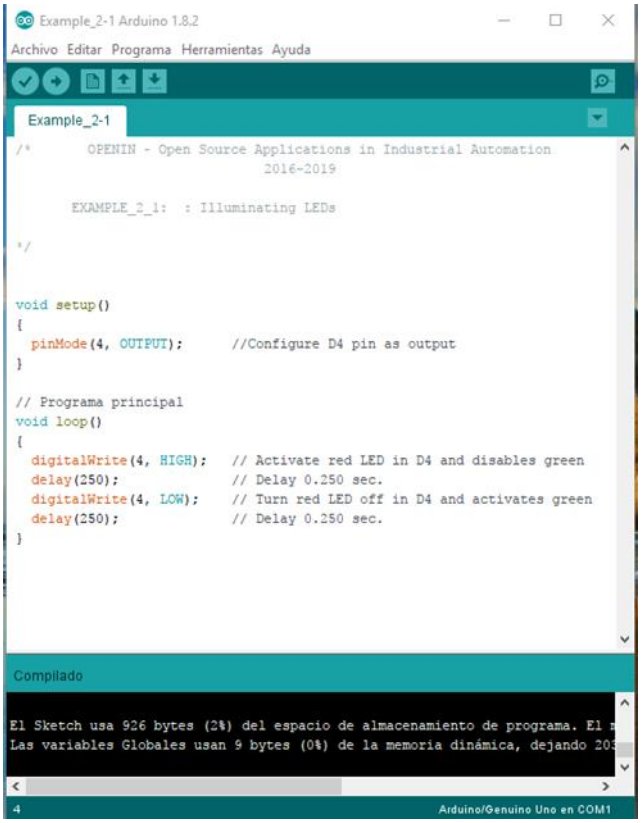
Οπότε αυτό το LED θα ανάψει όταν το D4=«1». Από την άλλη, το D4 είναι συνδεδεμένο στην κάθοδο του πράσινου LED (VERDE) με την άνοδο να είναι στα +5 V, οπότε αυτό το LED θα ανάψει όταν το D4 = «0».

Το πρόγραμμα είναι πολύ απλό (Εικόνα 11). Στην συνάρτηση setup() το pin D4 είναι ρυθμισμένο ως έξοδος.

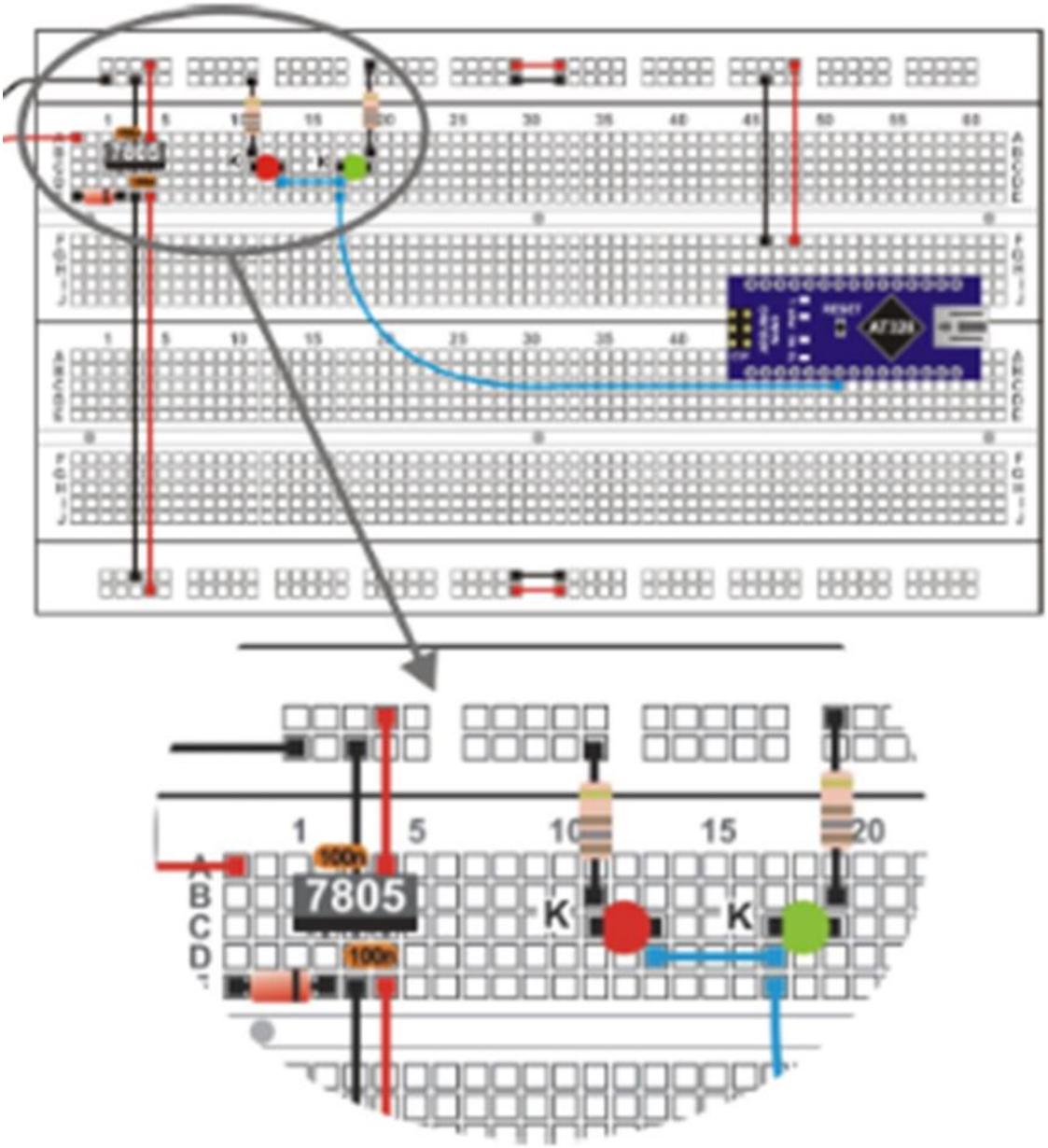
Το κυρίως πρόγραμμα είναι μια συνεχής επανάληψη. Το pin D4 είναι στο επίπεδο «1». Το κόκκινο LED (ROJO) ανάβει, ενώ το πράσινο LED (VERDE) είναι ανενεργό.

Χρονισμένο στα 0.25 δευτερόλεπτα, το pin D4 είναι ρυθμισμένο στο επίπεδο «0». Το κόκκινο LED (ROJO) θα απενεργοποιηθεί και το πράσινο LED (VERDE) θα ανάψει. Μετά από ακόμα 0.25 δευτερόλεπτα ο κύκλος επαναλαμβάνεται.

Παρακάτω στην Εικόνα 12 φαίνεται πως είναι η συναρμολόγηση σε ένα bread board. Δώστε προσοχή στις διάταξη και την θέση των εξαρτημάτων.



Εικόνα 11



Εικόνα 12

Άσκηση

Το πρόγραμμα είναι πολύ απλό και δεν θα χρειαστεί να το αλλάξετε. Αυτό που μπορείτε να κάνετε, είναι να αλλάξετε την τιμή της αντίστασης απορρόφησης R2 του πράσινου LED. Η τιμή της είναι 180Ω οπότε αλλάξτε την σε 10ΚΩ.

Τι παρατηρείτε ;

-

Εξηγείστε γιατί

-

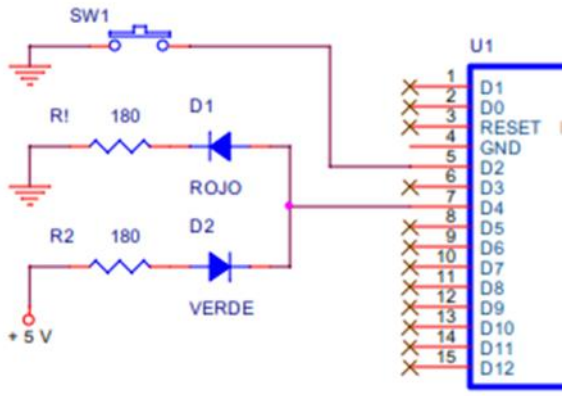
Προσπαθήστε να βρείτε πόσο ρεύμα περνά από το πράσινο LED :

$$I = \frac{V}{R} = \frac{V - V_{AK}}{R} = \frac{5 - 1.5}{10000} = 0.00035\ A = 0,35\ mA$$

Βλέπετε ότι τώρα το ρεύμα που περνά από το πράσινο LED είναι 0.35 mA, ενώ ο κατασκευαστής προτείνει να είναι περίπου 20mA. Τώρα ξέρετε άλλον έναν τρόπο να ρυθμίσετε την φωτεινότητα ενός LED. Βάλτε την αντίσταση R2 στην αρχική της τιμή στα 180Ω.

5. ΠΑΡΑΔΕΙΓΜΑ 2: ΠΑΡΑΚΟΛΟΘΗΣΗ ΕΙΣΟΔΩΝ

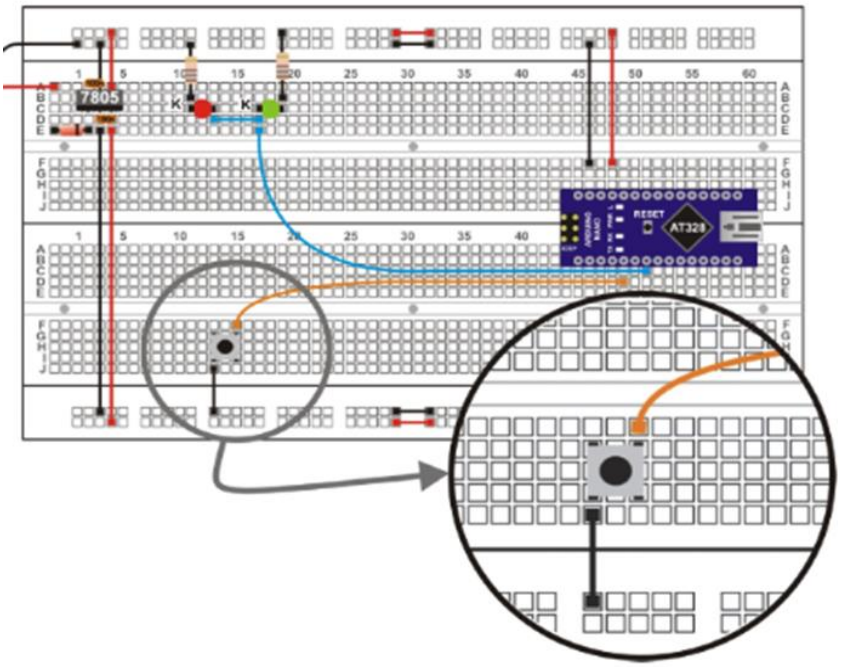
Το παρακάτω διάγραμμά (Εικόνα 13) είναι το ίδιο με του προηγούμενου παραδείγματος με τη διαφορά ότι στο pin D2 είναι συνδεδεμένο ένας κουμπί πίεσης.



Δεν υπάρχει κάτι διαφορετικό από προγραμματιστικής άποψης. Αυτό που πρέπει να κάνετε είναι να διαβάσετε το λογικό επίπεδο του pin D2 και να το εφαρμόσετε στα LED. Αν η είσοδος είναι στο «1» το κόκκινο LED ανάβει και αν είναι στο «0» τότε ανάβει το πράσινο LED. Και τα δύο LED ελέγχονται από το pin εξόδου D4.

Εικόνα 13

Σας προτείνουμε να χρησιμοποιήσετε ένα αμβλύ αντικείμενο για να ισιώσετε τα pins που κουμπιού πίεσης ώστε να προσαρμοστεί σωστά στο bread board. Τα pins θα πρέπει να είναι εντελώς κάθετα στο σώμα του κουμπιού. Η Εικόνα 14 δείχνει πως πρέπει να είναι η συναρμολόγηση. Θα πρέπει σταδιακά να εξοικειωθείτε με την συναρμολόγηση κυκλωμάτων πάνω στο bread board.

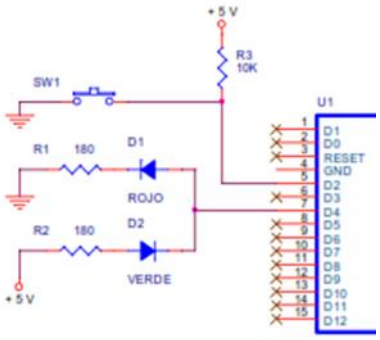


Εικόνα 14

Άσκηση

Όπως και πριν δεν υπάρχει κάτι στο πρόγραμμα του παραδείγματος 2, που δεν γνωρίζετε. Αυτό που έχετε να κάνετε είναι τα το γράψετε στο Arduino UNO και να ελέγξετε ότι δουλεύει σωστά.

Είναι προφανές ότι όταν πιέζετε το κουμπί στέλνετε το επίπεδο «0» μέσω του pin D2 και το πράσινο LED ανάβει. Αλλά αν δεν το πιέσετε τι γίνετε; Και κάτι ακόμα. Αν ακουμπήσετε με τα δάκτυλά σας το καλώδιο του κουμπιού ή το καλώδιο του LED ή ακόμα και τα pin του Arduino UNO, τι συμβαίνει;

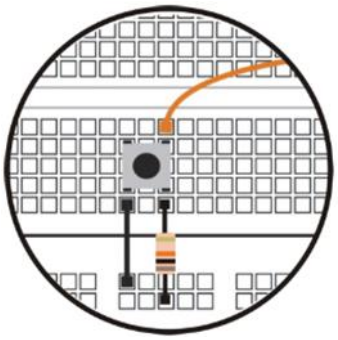


Εικόνα 15

Φαίνεται ότι τα LED ανάβουν τυχαία. Αυτό συμβαίνει διότι όταν δεν πιέζουμε το SW1, η είσοδος D2 «επιπλέει» και δεν έχει ένα ξεκάθαρο λογικό επίπεδο. Δείτε ξανά το αντίστοιχο θεωρητικό τμήμα.

Αλλάξτε το ηλεκτρικό κύκλωμα, προσθέτοντας μια αντίσταση 10ΚΩ όπως φαίνεται στο θεωρητικό διάγραμμα (Εικόνα 15) και σε αυτό της ηλεκτρονικής εγκατάστασης (Εικόνα 16). Βλέπουμε ότι πρόκειται για μια αντίσταση pull-up. Αυτή η αντίσταση εξασφαλίζει ότι το pin D2 θα μένει στο επίπεδο «1» όσο δεν πιέζουμε το κουμπί.

Ελέγξτε ότι το παράδειγμα 2 λειτουργεί σωστά χωρίς αλλαγές και ότι τα LEDs ανάβουν κανονικά



Εικόνα 16

6. ΠΑΡΑΔΕΙΓΜΑ 3: ΠΑΡΑΚΟΛΟΘΗΣΗ ΕΙΣΟΔΩΝ ΜΕ PULL-UP ΑΝΤΙΣΤΑΣΕΙΣ

Αυτό το παράδειγμα παρέχει μια οριστική λύση για τις αντιστάσεις pull-up. Το pin D2 είναι ρυθμισμένο ως είσοδος με μια αντίσταση pull-up οπότε δεν χρειάζεται να προσθέσετε άλλη. Και τώρα το κύκλωμα μοιάζει ακριβώς όπως και αυτό του πρώτου διαγράμματος του προηγούμενου παραδείγματος. Η μόνη διαφορά εδώ είναι στο πρόγραμμα (Εικόνα 17).

```
void setup()
{
  pinMode(4, OUTPUT);      //Configure D4 pin as output
  pinMode(2, INPUT_PULLUP);
}
```

Εικόνα 17

Το pin D2 είναι ρυθμισμένο να λειτουργεί ως είσοδος με μια αντίσταση pull-up, μέσω της συνάρτησης pinMode(2, INPUT_PULLUP) που βρίσκεται στην setup().

Πριν γράψετε το πρόγραμμα αφαιρέστε την αντίσταση pull-up 10ΚΩ που βάλαμε στο προηγούμενο παράδειγμα. Δεν πρέπει να υπάρχει πάνω από μία αντίσταση και σε αυτή την περίπτωση την παρέχει ήδη το Arduino.

Τώρα γράψτε το πρόγραμμα και βεβαιωθείτε ότι όλα λειτουργούν σωστά. Τα LEDs που είναι συνδεδεμένα στο pin D4 παρακολουθούν την λογική κατάσταση του κουμπιού που είναι συνδεδεμένο στο pin D2. Το κόκκινο LED ανάβει όσο δεν πιέζουμε το κουμπί (επίπεδο «1» λόγω της αντίστασης pull-up) και το πράσινο LED ανάβει όταν πιέσουμε το κουμπί (επίπεδο «0»).

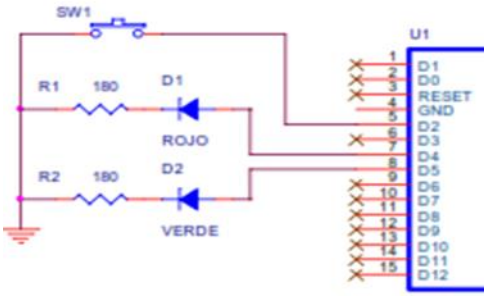
Συνοψίζοντας τα τρία πρώτα παραδείγματα μπορούμε να βγάλουμε τα εξής συμπεράσματα:

- Οποιοδήποτε περιφερειακό εξόδου μπορεί να ενεργοποιηθεί με τη χρήση επιπέδου «1» ή «0», ανάλογα με το πως είναι συνδεδεμένο.
- Με την ίδια λογική τα περιφερειακά μπορούν να παράγουν επίπεδο «1» ή «0» ανάλογα με το πως είναι συνδεδεμένα.
- Το επίπεδο «0» έχει ίδια αξία με το επίπεδο «1».
- Κάποια περιφερειακά μπορεί να έχουν ή να μην έχουν μία αντίσταση pull-up. Αν δεν έχουν θα πρέπει να βάλετε εσείς μια ή να χρησιμοποιήσετε αυτή του Arduino UNO.

7. ΠΑΡΑΔΕΙΓΜΑ 4: ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΕΙΣΟΔΩΝ ΧΩΡΙΣ ΔΙΑΚΟΠΕΣ.

Και πάλι σε αυτό το παράδειγμα δεν θα δείτε κάτι που δεν ξέρετε όσο αναφορά το πρόγραμμα αλλά θα μπορείτε να δείτε αν είναι καλή ιδέα να χρησιμοποιείτε διακοπές ή όχι.

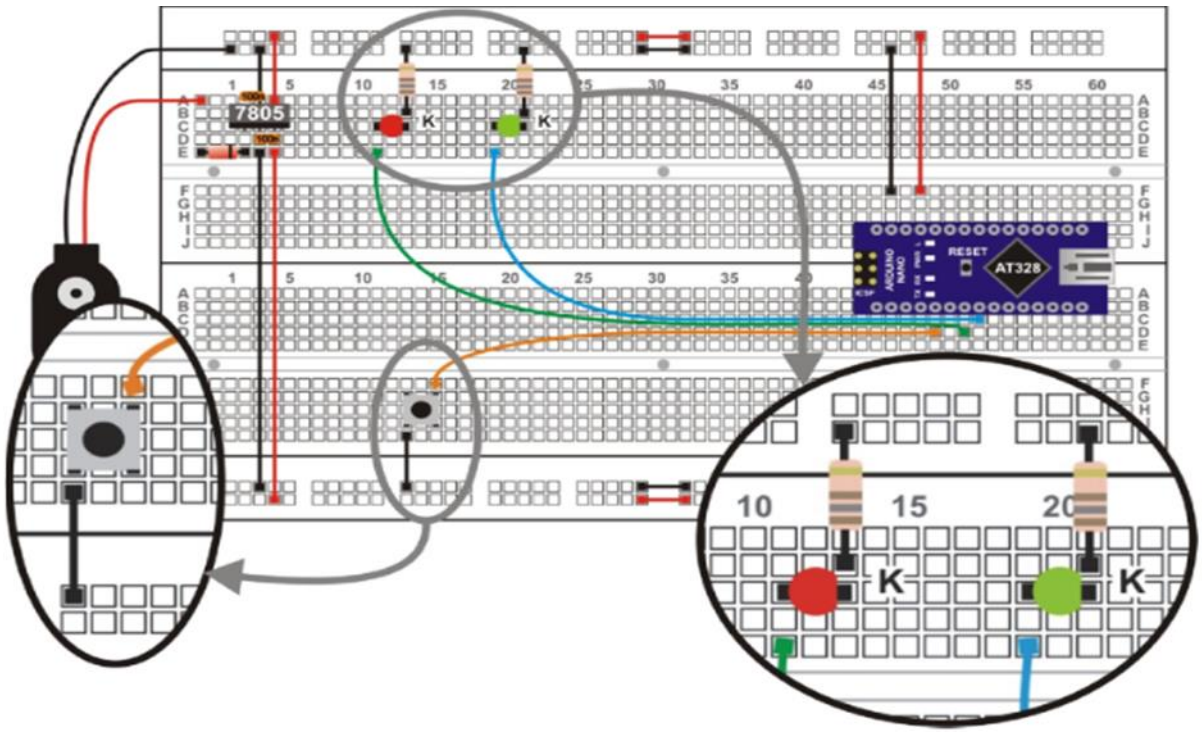
Ένα κόκκινο LED είναι συνδεδεμένο στο pin D4 και θα παρακολουθεί συνεχώς την κατάσταση της εισόδου που είναι στο pin D2. Η ιδέα είναι ότι το πράσινο LED που είναι στο pin D5 θα αναβοσβήνει κάθε 5 δευτερόλεπτα.



Στην εικόνα 18 φαίνεται το διάγραμμα του κυκλώματος. Το κόκκινο LED (ROJO) που είναι στο D4, παρακολουθεί την κατάσταση του κουμπιού που είναι συνδεδεμένο στο pin D2. Το πράσινο LED (VERDE) ελέγχεται από το pin D5 και αναβοσβήνει κάθε 5 δευτερόλεπτα. Προσέξτε ότι και τα δύο LEDs ανάβουν στο επίπεδο «1».

Εικόνα 18

Αυτή είναι η άσκηση που θα κάνετε στο bread board (Εικόνα 19).



Εικόνα 19

Όταν όλα είναι έτοιμα και έχετε γράψει το πρόγραμμα, βεβαιωθείτε ότι λειτουργεί σωστά.

Το κόκκινο LED που παρακολουθεί την κατάσταση του κουμπιού μοιάζει να έχει καθυστέρηση. Και κάποιες φορές αν πιέσετε το κουμπί πολύ γρήγορα το κόκκινο LED δεν το «αντιλαμβάνεται». Γιατί συμβαίνει αυτό;

Όσο ο ελεγκτής εκτελεί μια από τις συναρτήσεις delay(500) δεν μπορεί να εκτελέσει την συνάρτηση digitalWrite(4, digitalRead(2)) την ίδια στιγμή. Με άλλα λόγια δεν προλαβαίνει να «δει» τι συμβαίνει με την είσοδο D2 του κουμπιού. Επομένως δεν μπορεί να γνωρίζει και την κατάσταση στην D4.

Στην πραγματικότητα, ο ελεγκτής παρακολουθεί μόνο κάθε φορά που ολοκληρώνεται ένας κύκλος ενεργοποίησης και απενεργοποίησης από την έξοδο D5, κάτι το οποίο συμβαίνει κάθε 1 δευτερόλεπτο. Ας προσπαθήσουμε να λύσουμε το πρόβλημα με τη χρήση μιας διακοπής.

8. ΠΑΡΑΔΕΙΓΜΑ 5: ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΕΙΣΟΔΩΝ ΜΕ ΔΙΑΚΟΠΕΣ

Αυτό παράδειγμα κάνει ακριβώς το ίδιο με το προηγούμενο, αλλά σωστά αυτή τη φορά. Το LED που είναι συνδεδεμένο στην έξοδο D5 ανάβει για μισό δευτερόλεπτο και σβήνει για άλλο τόσο. Την ίδια στιγμή η έξοδος D4 παρακολουθεί την λογική κατάσταση της εισόδου D2.

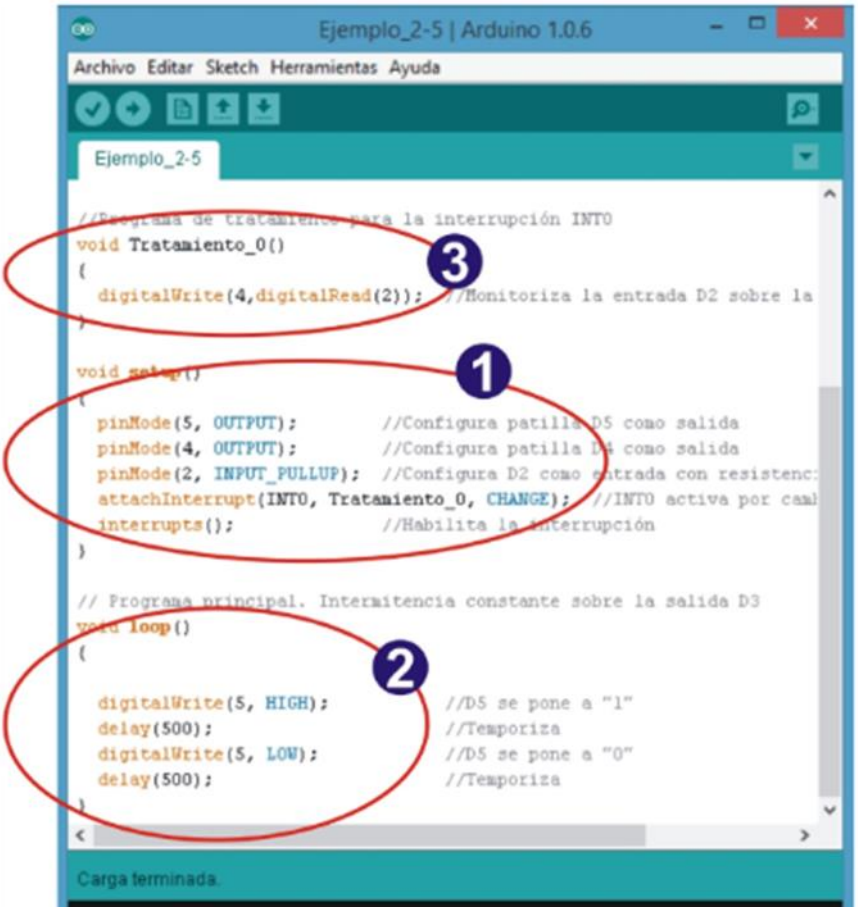
Χρησιμοποιήσετε το ίδιο κύκλωμα και αλλάξτε το πρόγραμμα ώστε να έχει τρία διαφορετικά τμήματα. Δείτε την Εικόνα 20:

Setup():

Τα pins D4 και D5 είναι ρυθμισμένα ως έξοδοι. Το pin D2 είναι ρυθμισμένο ως είσοδος με μια αντίσταση pull-up. Η διακοπή ρυθμίζεται με την χρήση της συνάρτησης attachInterrupt(). Αυτή συσχετίζεται με το pin INT0 (D2), που ενεργοποιείται όταν εντοπιστεί μια αλλαγή στο pin. Κάθε φορά που συμβαίνει αυτό εκτελείται η συνάρτηση Treatment_0(). Η συνάρτηση interrupts() ενεργοποιεί και εξουσιοδοτεί τις διακοπές που σε αυτή την περίπτωση είναι η INT0.

Loop():

Είναι το κυρίως πρόγραμμα. Προκαλεί παύσεις στην έξοδο D5. Το LED ανάβει για μισό δευτερόλεπτο και σβήνει για άλλο τόσο.



Εικόνα 20

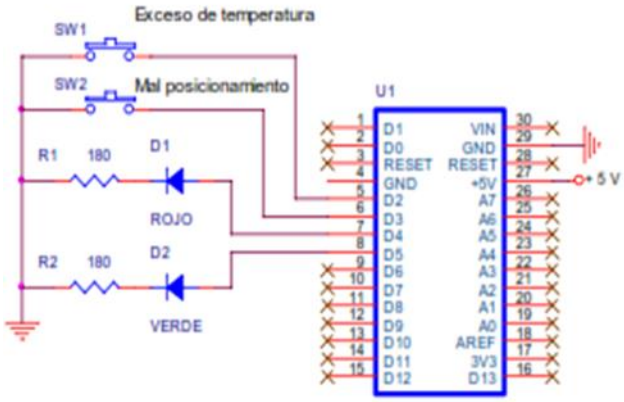
Treatment_0():

Αυτό είναι το πρόγραμμα που χειρίζεται τη ρουτίνα υπηρεσίας διακοπής ή ISR. Κάθε φορά που εντοπίζεται μια αλλαγή στην κατάσταση του INT0 (D2), ο ελεγκτής σταματά ότι κάνει και εκτελεί την συνάρτηση Treatment_0(). Αυτή η συνάρτηση διαβάζει την κατάσταση της εισόδου στο D2 και παρακολουθεί την έξοδο στο D4. Όταν τελειώσει επιστρέφει στο κυρίως πρόγραμμα και συνεχίζει από εκεί που είχε σταματήσει.

Όταν γράψετε το πρόγραμμα θα μπορείτε να διαπιστώσετε ότι ο ελεγκτής παρακολουθεί στιγμιαία το κόκκινο LED που καταγράφει την κατάσταση εισόδου. Παρόλα αυτά το πράσινο LED συνεχίζει να αναβοσβήνει σαν να μην είχε γίνει τίποτα. Θυμηθείτε ότι το πρόγραμμα Treatment_0() (αυτό που παρακολουθεί την κατάσταση εισόδων) εκτελείται σε μερικά μικρό-δευτερόλεπτα και δεν επηρεάζει καθόλου το κυρίως πρόγραμμα.

9. ΠΑΡΑΔΕΙΓΜΑ 6: ΕΛΕΓΧΟΣ ΔΥΟ ΔΙΑΚΟΠΩΝ

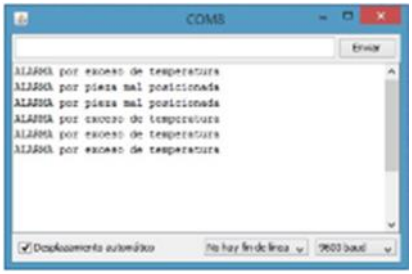
Τελειώνουμε την ενότητα 2 με ένα παράδειγμα που χρησιμοποιεί τις δύο διακοπές που είναι διαθέσιμες στο ARDUINO UNO: την INT0 (D2) και INT1(D3). Φανταστείτε μια μηχανή που είναι υπεύθυνη για τον έλεγχο δύο εξόδων , D4 και D5 που ακολουθούν μια συγκεκριμένη ακολουθία.



Δύο αισθητήρες ανιχνεύουν δύο πιθανούς συναγερμούς: υπερθέρμανση και λάθος θέση εξαρτήματος. Οι αισθητήρες που οδηγούνται από τα κουμπιά SW1 και SW2 είναι συνδεδεμένοι στα pin D2 και D3 και ενεργοποιούν τις διακοπές INT0 και INT1 αντίστοιχα. Δείτε το διάγραμμα της Εικόνας 21.

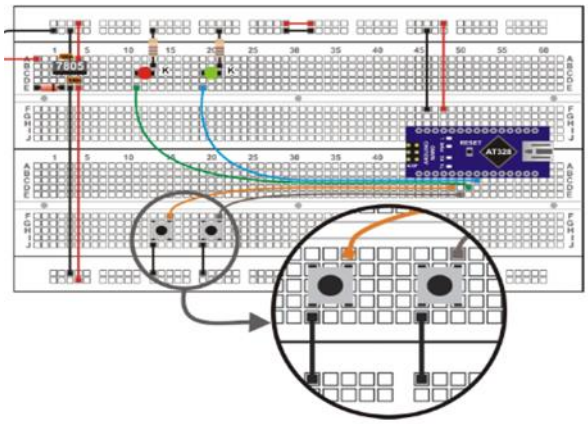
Εικόνα 21

Το πρόγραμμα ανάβει και σβήνει το κόκκινο και το πράσινο LED με τη σειρά. Όταν μια από της διακοπές εντοπιστεί τότε το κατάλληλο πρόγραμμα Treatment_0() μεταδίδει μια σειριακή προειδοποίηση σε μορφή μηνύματος όπως αυτά που βλέπετε στην Εικόνα 22.



Εικόνα 22

Και οι δύο διακοπές έχουν ρυθμιστεί να ενεργοποιούνται με την πτώση του άκρου . Αυτό συμβαίνει κάθε φορά που πιέζετε το αντίστοιχο κουμπί. Θυμηθείτε ότι τα pin D2 (INT0) και D3 (INT1) είναι ρυθμισμένα ως είσοδοι με αντιστάσεις pull-up. Όταν το κουμπί δεν πιέζεται τα pins έχουν επίπεδο «1». Εδώ μπορείτε να δείτε την συναρμολόγηση για την άσκηση (Εικόνα 23). Είναι παρόμοια με αυτή του προηγούμενου παραδείγματος. Έχει προστεθεί το κουμπί SW2 που είναι συνδεδεμένο με το D3 (INT1).



Εικόνα 23

Όταν γράψετε το πρόγραμμα ελέγξτε ότι λειτουργεί σωστά. Ανοίξτε την σειριακή παρακολούθηση. Το κυρίως πρόγραμμα περιορίζεται στο να αναβοσβήνει το κόκκινο και πράσινο LED χωρίς να πιέζεται κανένα κουμπί. Αν θέλετε μπορείτε να το αλλάξετε αυτό.

Αν κάποιο από τα κουμπιά πιεστεί, θα δείτε το κατάλληλο μήνυμα στο παράθυρο σειριακής παρακολούθησης.

Βλέπουμε ότι κατά διαστήματα, όταν κάποια από τις δύο διακοπές ενεργοποιηθεί το σύστημα κλείνει. Πρέπει να κάνετε επανεκκίνηση πιέζοντας το RESET. Αυτό συμβαίνει εξαιτίας των δύο κουμπιών. Ακόμα και αν το πατήσετε μία φορά μπορεί να διαβαστούν μερικές μαζεμένες διακοπές και το Arduino δεν μπορεί να τις διαχειριστεί σωστά. Είδαμε επίσης στο εργαστήριο ότι όταν τα σήματα διακοπών παράγονται από γεννήτρια δεν έχουμε τέτοια αποτελέσματα και το σύστημα δεν κλείνει. Ενεργοποιήσαμε 10 διακοπές ανά δευτερόλεπτο για 5 λεπτά και το σύστημα πάντα τα κατάφερνε.

Πειραματιστείτε και με άλλα περιφερειακά εκτός από κουμπιά. Ώρα για δουλειά !

ΑΝΑΦΟΡΕΣ

ΒΙΒΛΙΑ

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

ΙΣΤΟΣΕΛΙΔΕΣ

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com