ΕΝΟΤΗΤΑ 1: ΤΑ ΠΡΩΤΑ ΠΡΟΓΡΑΜΜΑΤΑ

ΣΤΟΧΟΙ

Η δημιουργία αρχικών προγραμμάτων θα σας βοηθήσουν να δουλέψετε εύκολα και γρήγορα με ψηφιακές εισόδους και εξόδους (I/O).

Θα δούμε προγράμματα που έχουν γραφτεί σε γλώσσα προγραμματισμού για επεξεργαστές Arduino ώστε στη συνέχεια να καταφέρετε να γράψετε και μόνοι σας. Είναι μια εξελιγμένη γλώσσα με τους δικούς της κανόνες, δηλώσεις και συντακτικό. Πρώτα θα δούμε τις βασικές δηλώσεις που χρειάζονται για να χρησιμοποιήσετε τις ψηφιακές εισόδους και εξόδους.

ΕΝΟΤΗΤΑ ΘΕΩΡΙΑΣ

- ΠΩΣ ΜΟΙΑΖΕΙ ΕΝΑ ΠΡΟΓΡΑΜΜΑ
  - Τμήμα σχολίων
  - Τμήμα δήλωσης μεταβλητών και συναρτήσεων
  - Τμήμα εργασιών διαμόρφωσης
  - Τμήμα βασικού μέρους του προγράμματος
  - Τμήμα μηνυμάτων
- ΔΗΛΩΣΕΙΣ ΠΡΟΓΡΑΜΜΑΤΟΣ
  - Η συνάρτηση setup()
  - Η συνάρτηση loop()
  - Η συνάρτηση pinMode()
  - Η συνάρτηση digitalRead()
  - Η συνάρτηση digitalWrite()
- ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ
  - Ο τελεστής NOT
  - Ο τελεστής AND
  - Ο τελεστής OR
  - Συνθέτοντας τελεστές

ΕΝΟΤΗΤΑ ΕΞΑΣΚΗΣΗΣ

- ΠΑΡΑΔΕΙΓΜΑ 1: Ενεργοποίηση ενός LED 1 volt
- ΠΑΡΑΔΕΙΓΜΑ 2: Φωτισμός ενός LED 2 volt
- ΠΑΡΑΔΕΙΓΜΑ 3: Φωτισμός ενός LED 3 volt
- ΠΑΡΑΔΕΙΓΜΑ 4: Αντίστροφος φωτισμός ενός LED
- ΠΑΡΑΔΕΙΓΜΑ 5: Έλεγχος δύο κουμπιών
- ΠΑΡΑΔΕΙΓΜΑ 6: Ατομική δουλειά

ΑΠΑΙΤΟΥΜΕΝΑ ΥΛΙΚΑ

- Σταθερός ή φορητός υπολογιστής
- Περιβάλλον εργασίας Arduino IDE: αυτό πρέπει να περιέχει το συμπληρωματικό υλικό εγκατεστημένο και ρυθμισμένο
- Ελεγκτής Arduino UNO
- Ένα USB καλώδιο

Περιεχόμενα

ΕΝΟΤΗΤΑ ΘΕΩΡΙΑΣ

1. ΠΩΣ ΜΟΙΑΖΕΙ ΕΝΑ ΠΡΟΓΡΑΜΜΑ

Α. Τμήμα σχολίων

Όλα τα προγράμματα θα πρέπει να ξεκινούν παρέχοντας συγκεκριμένες πληροφορίες. Σε αυτή τη περίπτωση, για παράδειγμα, αυτές θα είναι το όνομα του μαθήματος, η ημερομηνία, ο συγγραφέας, η εταιρία κτλ. Θα είναι ενδιαφέρον να διαβάσετε πιο αναλυτικά από τι αποτελείτε ένα πρόγραμμα και το τι κάνει.

EXAMPLE_4_1

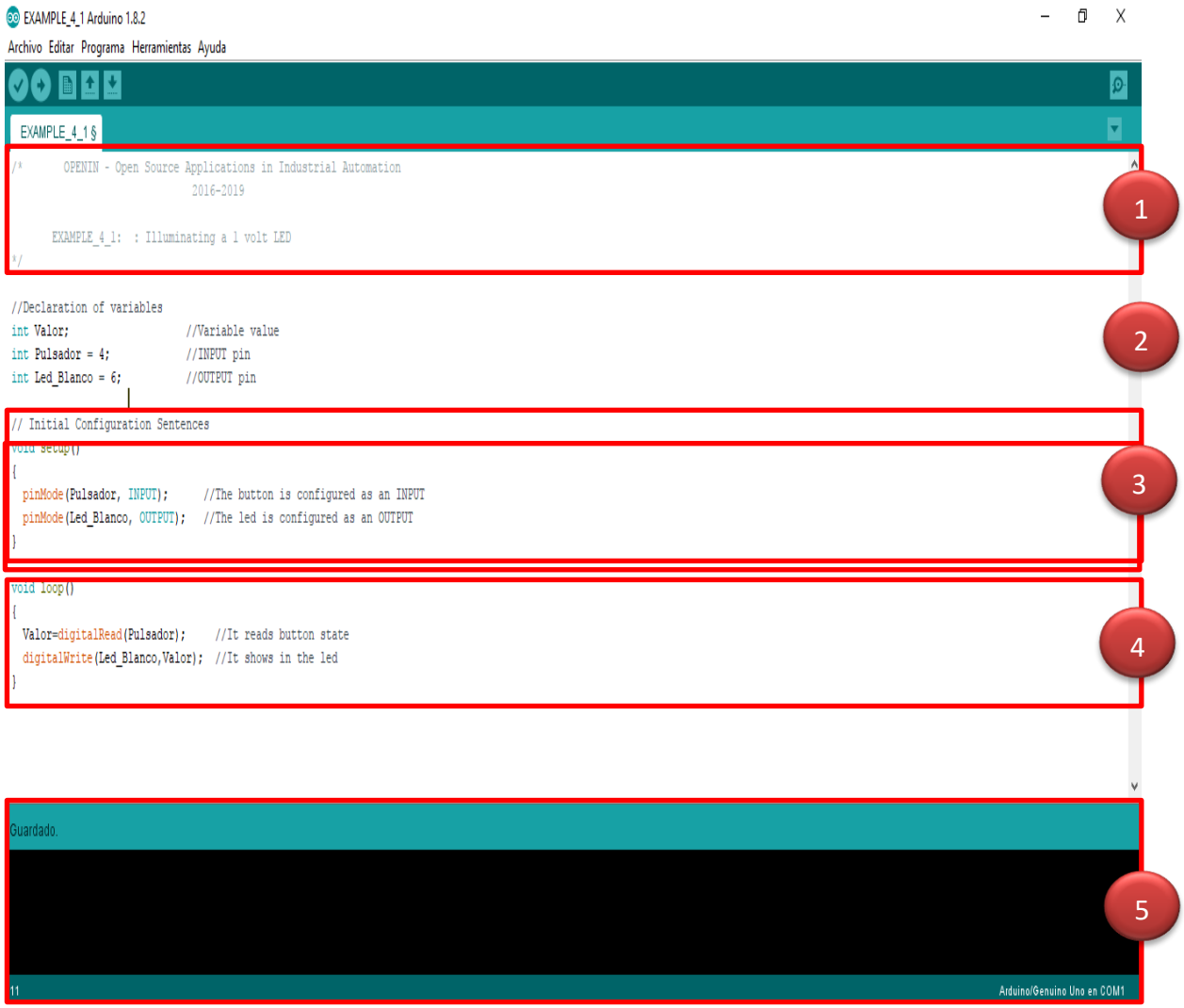
```
/*       OPENIN - Open Source Applications in Industrial Automation
                                2016-2019

       EXAMPLE_4_1:  : Illuminating a 1 volt LED
*/
```

1

Αυτές οι πληροφορίες λέγονται “Σχόλια επικεφαλίδας”. Μπορείτε να προσθέσετε όσα σχόλια θέλετε όπου θέλετε, αρκεί αυτά να είναι ανάμεσα στα σύμβολα “/*” και “*/”. Κοιτάξτε το παραπάνω παράδειγμα.

Μπορείτε επίσης να προσθέσετε πιο απλά σχόλια σε μια γραμμή απλώς βάζοντας το σύμβολο “//” πριν από αυτά. Αυτά τα σχόλια είναι τα πιο συχνά. Συνηθίστε να βάζετε σχόλια στον κώδικά σας. Μπορεί να τα χρειαστείτε στο μέλλον για να θυμηθείτε τι κάνατε, πως το κάνατε ή γιατί το κάνατε.



B. Τμήμα δήλωσης μεταβλητών και συναρτήσεων

Σε αυτό το τμήμα θα δηλώσετε τις μεταβλητές και τις συναρτήσεις που χρησιμοποιείτε στο πρόγραμμα σας.

Σκεφτείτε μια μεταβλητή ως ένα είδος κουτιού ή δοχείου που αντιστοιχίζετε ένα όνομα και πιθανώς μια τιμή. Την αποθηκεύετε στη μνήμη RAM του ελεγκτή (Arduino) και μπορείτε να την χρησιμοποιήσετε στη συνέχεια όσες φορές θέλετε.

Μια συνάρτηση, από την άλλη, περιλαμβάνει έναν αριθμό από άλλες δηλώσεις, εντολές ή συναρτήσεις. Δίνετε ένα όνομα σε αυτό το σύνολο και μπορείτε να το χρησιμοποιήσετε όσες φορές θέλετε.

```
//Declaration of variables
int Valor;                    //Variable value
int Pulsador = 4;             //INPUT pin
int Led_Blanco = 6;           //OUTPUT pin
```

2

Στο παράδειγμα της παραπάνω εικόνας έχουν δημιουργηθεί τρεις μεταβλητές η “Valor” (Value), η ”Pulsador” (Button) και η “Led_Blanco” (White_LED). Η τιμή της ψηφιακής εισόδου θα αποθηκευτεί στην πρώτη. Στη δεύτερη δηλαδή την “ Pulsador ”, δίνετε η τιμή 4 που αντιστοιχεί στον αριθμό της εισόδων (pin) που έχει τοποθετηθεί το κουμπί. Πρέπει να διαβάσουμε την κατάσταση του κουμπιού. Δίνουμε την τιμή 6 στη μεταβλητή “ Led_Blanco ”. Αυτό αντιστοιχεί στην έξοδο (pin) που έχει τοποθετηθεί το LED. Το Led Blanco (white led) είναι ένα από τα BASIC I/O της πειραματικής πλακέτας.

Γενικά, οι μεταβλητές και οι συναρτήσεις πρέπει να δηλώνονται ΠΡΙΝ χρησιμοποιηθούν στο πρόγραμμα.

Γ. Τμήμα εργασιών διαμόρφωσης

Συνήθως τα προγράμματα που είναι γραμμένα στη γλώσσα προγραμματισμού Arduino, ξεκινάνε με την εκτέλεση μερικών οδηγιών ή συναρτήσεων. Αυτές προσδιορίζουν ποια pin θα χρησιμοποιηθούν ως είσοδοι και ποια ως έξοδοι.

```
// Initial Configuration Sentences
void setup()
{
  pinMode(Pulsador, INPUT);      //The button is configured as an INPUT
  pinMode(Led_Blanco, OUTPUT);   //The led is configured as an OUTPUT
}
```

3

Η μεταβλητή “ Pulsador” (Button) στην εικόνα 2, που προηγουμένως της είχε δοθεί η τιμή 4, έχει οριστεί ως ΕΙΣΟΔΟΣ.

Η μεταβλητή " Led_Blanco " (White_LED) , έχει οριστεί ως ΕΞΟΔΟΣ.

Γενικά, οι δηλώσεις ή οι συναρτήσεις διαμόρφωσης εκτελούνται μόνο όταν γίνει RESET στο σύστημα ή το συνδέσετε σε μια πηγή ενέργειας.

Δ. Τμήμα βασικού μέρους του προγράμματος

Πρέπει να γράψετε όλες τις οδηγίες, δηλώσεις και συναρτήσεις που αποτελούν το πρόγραμμα σε αυτήν την ενότητα. Σε αυτό το παράδειγμα είναι μόνο δύο συναρτήσεις που έχουμε να ασχοληθούμε προς το παρόν, αλλά έχετε στο μυαλό σας ότι ένα πρόγραμμα μπορεί να περιέχει εκατοντάδες ή ακόμα και χιλιάδες.

```
void loop()
{
  Valor=digitalRead(Pulsador);     //It reads button state
  digitalWrite(Led_Blanco,Valor);  //It shows in the led
}
```

4

Η πρώτη συνάρτηση, "Valor=digitalRead(Pulsador);", διαβάζει τη ψηφιακή κατάσταση του κουμπιού που είναι συνδεδεμένο στο pin 4, αυτό δηλαδή που πριν είχε οριστεί ως είσοδος. Η κατάσταση του κουμπιού έχει αποθηκευτεί στη μεταβλητή " Valor " (Value) στη μνήμη RAM.

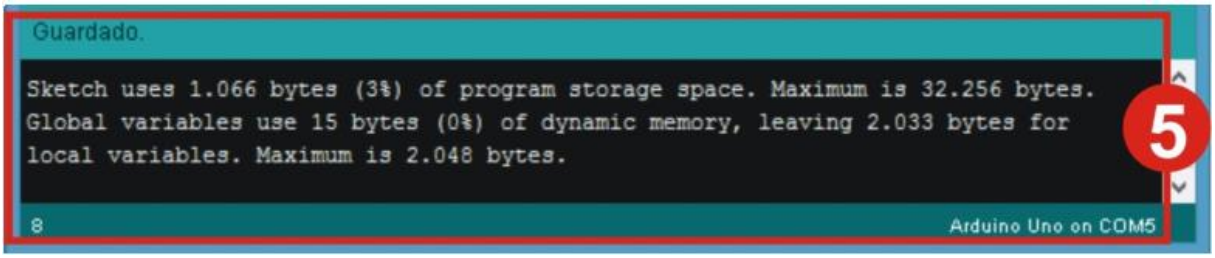
Η δεύτερη συνάρτηση, "digitalWrite(Led_Blanco,Valor);", γράφει τα περιεχόμενα της μεταβλητής "Valor" στο pin 6. Θυμηθείτε ότι αυτό το pin ορίστηκε πριν ως έξοδος .

Ο ελεγκτής εκτελεί όλες τις συναρτήσεις που αποτελούν το βασικό μέρος του προγράμματος όσο πιο γρήγορα γίνετε. Τις εκτελεί συνεχώς και επ' αόριστων από την πρώτη ως την τελευταία.

Σημαντικό: Όταν δηλώνονται οι μεταβλητές, οι παραμετροποιήσεις και οι συναρτήσεις του προγράμματος, σιγουρευτείτε ότι πάντα τελειώνουν με το σύμβολο ";".

Ε. Τμήμα μηνυμάτων

Στο τελευταίο τμήμα, το περιβάλλον εργασίας Arduino IDE θα παρουσιάσει μια σειρά από μηνύματα. Θα σας ενημερώσει για το πότε κάνετε αποθήκευση, εκτέλεση ή εγγραφή ενός προγράμματος στη μνήμη του ελεγκτή. Επίσης θα σας ενημερώσει αν υπάρχουν λάθη στην εκτέλεση του προγράμματος, συντακτικά λάθη και από που προέρχονται.



Όπως βλέπουμε η παραπάνω άσκηση δούλεψε τέλεια. Μας λέει ότι τα πρόγραμμα καταλαμβάνει συνολικά 1,066 bytes από τα 32,256 bytes που είναι τα διαθέσιμα της μνήμης FLASH. 15 από τα 2,048 διαθέσιμα bytes της μνήμης RAM έχουν χρησιμοποιηθεί για δεδομένα (λόγω των μεταβλητών που δημιουργήθηκαν). Έτσι έμειναν 2,033 bytes ελεύθερα.

Θυμηθείτε: "Compile" σε ένα πρόγραμμα σημαίνει να μεταφραστεί αυτό που γράφατε σε γλώσσα Arduino σε binary κώδικα (γνωστό σαν γλώσσα μηχανής): αυτό ουσιαστικά είναι αυτό που αποθηκεύετε στη FLASH μνήμη του ελεγκτή. Αυτό είναι μια αυτοματοποιημένη διαδικασία.

2. ΔΗΛΩΣΕΙΣ ΠΡΟΓΡΑΜΑΤΟΣ

Τώρα που είδατε πως μοιάζει ένα πρόγραμμα στη θεωρία τουλάχιστον, θα δούμε στη συνέχεια και τη χρήση των συναρτήσεων. Κάθε γλώσσα προγραμματισμού έχει τις δικές της συναρτήσεις, τη δικιά τους σύνταξη και τους δικούς της κανόνες προγραμματισμού. Εμείς θα ασχοληθούμε με τη γλώσσα προγραμματισμού Arduino. Έχουμε ήδη χρησιμοποιήσει πέντε διαφορετικές εντολές ή συναρτήσεις, ας τις δούμε λοιπόν πιο αναλυτικά.

Α. Η συνάρτηση setup()

Αυτή η συνάρτηση και όλες οι άλλες συναρτήσεις που περιέχει εκτελούνται όταν το σύστημα κάνει επανεκκίνηση. Συμβαίνει κάθε φορά που πιέζετε το κουμπί RESET ή ενεργοποιείτε το σύστημα.

Συνήθως υπάρχει ένας αριθμός από άλλες συναρτήσεις μέσα σε αυτήν: καθορισμός pins ως είσοδοι και έξοδοι, συγκεκριμένες ρυθμίσεις μεταβλητών, βιβλιοθήκες κτλ. Γενικά, μπορείτε να βάλετε ότι δηλώσεις και συναρτήσεις θέλετε μέσα στη συνάρτηση **setup()** όσο αυτές είναι κλεισμένες μέσα σε άγκιστρα "{…}". Πάντα πρέπει να δημιουργείτε μια συνάρτηση **setup()** κι ας μην περιέχει τίποτα στο εσωτερικό της. Απλά αφήστε τα άγκιστρα κενά.

Είναι σημαντικό να θυμάστε ότι όλες οι συναρτήσεις που περιέχονται στη συνάρτηση **setup()** εκτελούνται μόνο μια φορά: όταν κάνετε επανεκκίνηση.

Σύνταξη:

Void setup()

{

.....

.....

}

Παράδειγμα

```
Void setup()
{
pinMode(INPUT, Button);          //To button ορίζεται ως είσοδος
pinMode(White_LED,OUTPUT);       //To button ορίζεται ως έξοδος
}
```

Β. Η συνάρτηση loop()

Εδώ θα βάλετε όλες τις οδηγίες και τις συναρτήσεις σας μέσα σε άγκιστρα “{...}”. Είναι αυτό που λέμε το κύριο σώμα του προγράμματος.

Σύνταξη:

void loop()

{

.....

.....

}

Παράδειγμα

Void loop()

{

Value=digitalRead(Button); //Διαβάζει την κατάσταση του button

digitalWrite(White_LED,Value); //Ενεργοποίηση του LED

}

Η κύρια συνάρτηση loop() του προγράμματος μπορεί να περιλαμβάνει δεκάδες, εκατοντάδες ή και χιλιάδες άλλες συναρτήσεις. Το παραπάνω παράδειγμα περιλαμβάνει μόνο δύο. Η συνάρτηση loop() μπορεί επίσης να είναι κενή αλλά πρέπει πάντα να υπάρχει.

Όλες οι άλλες συναρτήσεις μέσα στη συνάρτηση loop() εκτελούνται συνέχεια από την πρώτη μέχρι την τελευταία, η μια μετά την άλλη μέχρι το σύστημα να αποσυνδεθεί.

Γ. Η συνάρτηση pinMode()

Αυτή η συνάρτηση καθορίζει μια από τις υποδοχές του Arduino ως είσοδο ή έξοδο. Συνήθως την συναντάμε στην αρχή του προγράμματος και εμπεριέχεται στη συνάρτηση setup().

Σύνταξη:

pinMode;

pin: Αυτός είναι ο αριθμός του pin που θα οριστεί ως είσοδος ή έξοδος και στο Arduino UNO πρέπει να είναι 0- 13.

Mode: Αυτό ορίζει αν το pin θα λειτουργεί ως είσοδος ή ως έξοδος.

Παραδείγματα

int Button=4;

int White_LED=6;

pinMode(White_LED, OUTPUT); //Το pin 6 ορίζεται ως έξοδος

pinMode(9 OUTPUT); //Το pin 9 ορίζεται ως έξοδος

pinMode(Button, INPUT); //Το pin 4 ορίζεται ως είσοδος

pinMode(12, INPUT); //Το pin 12 ορίζεται ως είσοδος

Όταν κάνετε επανεκκίνηση του συστήματος όλα τα pins αυτόματα ορίζονται ως είσοδοι.

Δ. Η συνάρτηση digitalRead()

Αυτή η συνάρτηση διαβάζει και επιστρέφει την δυαδική λογική κατάσταση (“1” ή “0”, “HIGH” ή “LOW”) από κάθε pin του ελεγκτή Arduino.

Σύνταξη:

digitalRead(pin);

pin: Αυτό εμφανίζει τον αριθμό του pin που πρόκειται να διαβάσουμε. Στο Arduino UNO πρέπει να είναι 0-13.

Παραδείγματα

int Button=4;

Value=DigitalRead(Button); //Διαβάζει την κατάσταση του pin 4 και την αποθηκεύει στην μεταβλητή “Value”

X=digitalRead(8); // Διαβάζει την κατάσταση του pin 8 και την αποθηκεύει στην μεταβλητή “X”

Ε. Η συνάρτηση digitalWrite()

Γράφει ή ορίζει την δυαδική τιμή (“1” or “0”, “HIGH” or “LOW”) μέσω ενός pin εξόδου.

Σύνταξη:

digitalWrite(pin, value);

pin: Αυτό μας δίνει τον αριθμό του pin που πρόκειται να χρησιμοποιήσουμε για να δώσουμε την τιμή. Πρέπει να είναι ανάμεσα στη 0 και 13 στο Arduino UNO.

Value: Προσδιορίζει την τιμή που θα δοθεί (“1” ή “0”, “HIGH” ή “LOW”).

Παραδείγματα

int White_LED=6;

DigitalWrite(White_LED,Value);	**//Θέτει το περιεχόμενο της “*Value*” (“1” or “0” μέσω του pin 6**
DigitalWrite(11, LOW);	**//Θέτει επίπεδο “0” μέσω του pin εξόδου 11**
DigitalWrite(7,1);	**// Θέτει επίπεδο “1” μέσω του pin εξόδου 7**

3. ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ

Δεν μπορούμε να προχωρήσουμε παρακάτω χωρίς να μιλήσουμε για τους τρεις τύπους τελεστών που είναι ικανοί να συσχετίζουν σε διάφορες συναρτήσεις, λειτουργίες, εκφράσεις κτλ. Είναι γνωστοί ως λογικοί τελεστές και χρησιμοποιούνται πολύ στα ψηφιακά συστήματα. Θα γίνει μια προσπάθεια να τους εξηγήσουμε χωρίς να αναφερθούν πολλοί τεχνικοί όροι.

Α. Ο τελεστής NOT

Αυτός ο τελεστής εκφράζει την άρνηση (NOT) και αναπαρίσταται από το σύμβολο (!). Είναι πιο εύκολο να το καταλάβουμε δίνοντας ένα παράδειγμα. Κοιτάξτε τη συνάρτηση παρακάτω που σίγουρα θα καταλάβετε.

Value=digitalRead(Button);

Η μεταβλητή "Value" είναι ίση με το "1" όταν το "Button" είναι επίσης στο "1". Αν δεν συμβαίνει αυτό τότε η "Value" είναι ίση με το "0". Τώρα ας δούμε την παρακάτω συνάρτηση:

Value = ! digitalRead(Button);

Η μεταβλητή "Value" είναι ίση με το "1" όταν το "Button" διαβάζεται και **ΔΕΝ** είναι (**NOT)** στο "1", ή, με άλλα λόγια είναι στο "0". Αντίστοιχα, αν το "Button" είναι στο "1", η μεταβλητή "Value" θα είναι στο "0". Δείτε τα παραδείγματα από κάτω και απαντήστε:

Value = ! digitalRead(12);

"*Value*" είναι στο "1" ή "*true*" εάν: _____

"*Value*" είναι στο "0" ή "*false*" εάν: _____

Β. Ο τελεστής AND

Αυτός ο τελεστής δημιουργεί ένα "1" ή όπως είναι γνωστό ως "true", όταν ΟΛΑ τα στοιχεία που σχετίζονται μεταξύ τους είναι στο "1" ή "true". Παρουσιάζετε με τα σύμβολα "&&". Κοιτάξτε το παρακάτω παράδειγμα:

Value =digitalRead(4) && **digitalRead(12);**

Η μεταβλητή "Value" είναι ίση με το "1", "*true*", όταν οι είσοδοι 4 AND 12 είναι και οι δυο στο "1". Αν κάποια από τις εισόδους είναι στο "0" το αποτέλεσμα για τη μεταβλητή "Value" είναι επίσης στο "0" or "*false*". Κοιτάξτε τα παρακάτω παραδείγματα και απαντήστε:

Value =digitalRead(4) && **digitalRead(8)** && **digitalRead(12);**

"*Value*" είναι στο "1" ή "*true*" εάν: _____

"*Value*" είναι στο "0" ή "*false*" εάν: _____

Γ. Ο τελεστής OR

Αυτός ο τελεστής δημιουργεί ένα "1", επίσης γνωστό ως "*true*", όταν ΚΑΠΟΙΟ από τα στοιχεία που σχετίζονται είναι επίσης στο "1" ή είναι αληθές. Παρουσιάζεται από τα σύμβολα: "||". Κοιτάξτε το παρακάτω παράδειγμα:

Value =digitalRead(4) || digitalRead(12);

Η μεταβλητή "Value" είναι ίση με το "1", "*true*", όταν η είσοδος 4 Η (OR) ή είσοδος 12 Η (OR) και οι δύο είσοδοι είναι στο"1". Αν όλες οι είσοδοι είναι στο "0" το αποτέλεσμα της μεταβλητής "Value" είναι επίσης στο"0" ή "*false*". Κοιτάξτε το παρακάτω παράδειγμα και απαντήστε:

Value =digitalRead(4) || digitalRead(8) || digitalRead(12);

"*Value*" " είναι στο "1" ή "*true*" εάν:_____

"*Value*" είναι στο "0" ή "*false*" εάν: _____

Δ. Συνδυασμός τελεστών

Μπορείτε βέβαια να συνδυάσετε διάφορους λογικούς τελεστές στην ίδια συνάρτηση. Πρέπει όμως να χρησιμοποιήσετε παρενθέσεις για να ορίσετε τη σειρά που θα υπολογιστούν. Κοιτάξτε το παρακάτω παράδειγμα:

Value = ! digitalRead(4) && **digitalRead(7);**

Η μεταβλητή "Value" είναι ίση με το "1" or "*true*" όταν η είσοδος 4 είναι **NOT** στο "1", **ΑΛΛΑ** η είσοδος 7 είναι "1". Κοιτάξτε το παρακάτω παράδειγμα και απαντήστε:

Value = (digitalRead(8) && **! digitalRead(12)) || digitalRead(4));**

"*Value*" είναι στο "1" ή "true" εάν: _____

"*Value*" είναι στο "0" ή "false" εάν: _____

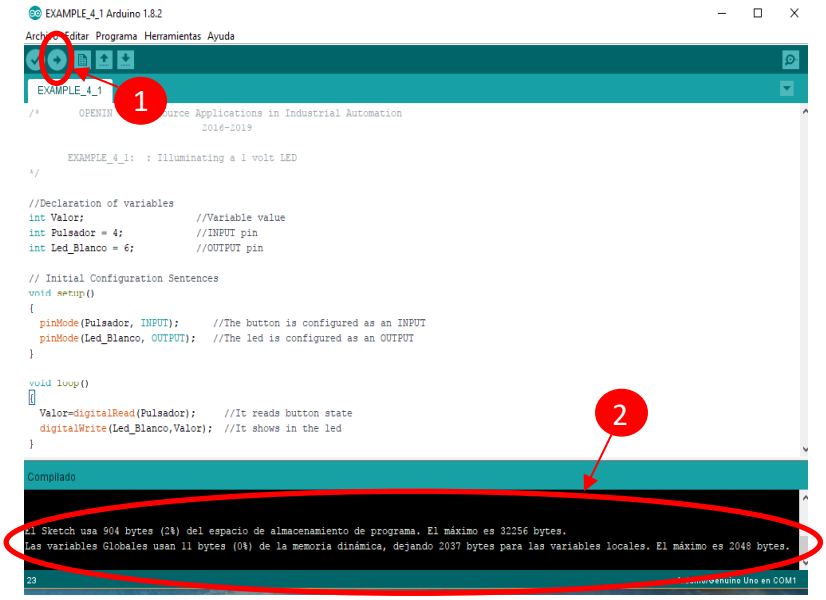
ΕΝΟΤΗΤΑ ΕΞΑΣΚΗΣΗΣ: ΠΑΡΑΔΕΙΓΜΑΤΑ

1. Παράδειγμα 1: Φωτισμός LED 1 volt

Αυτό είναι το παράδειγμα που μελετήσατε στην Ενότητα Θεωρίας και έτσι δεν θα εξηγήσουμε ξανά πως δουλεύει. Απλά θα σας θυμίσουμε ότι αυτό που κάνει είναι ότι διαβάζει την κατάσταση του κουμπιού που είναι συνδεδεμένο στον ακροδέκτη 4 και το αντιστοιχούμε στο white LED στα BASIC I/O της πλακέτας που είναι συνδεδεμένο στον ακροδέκτη 6 στον ελεγκτή.

- **Μεταγλώττιση και εγγραφή**

Όπως βλέπετε στην εικόνα, αυτό που έχετε να κάνετε είναι να πατήσετε το κουμπί που είναι μαρκαρισμένο με 1. Το Arduino IDE μεταγλωττίζει το πρόγραμμα για εσάς μεταφράζοντάς το σε γλώσσα μηχανής ή δυαδικό κώδικα. Όλη η διαδικασία είναι αρκετά γρήγορη και τελείως αυτόματη. Δεν είναι αναγκαστικό να γνωρίζετε πως ακριβώς συμβαίνει.



Αν το πρόγραμμα είναι γραμμένο σωστά και δεν έχει λάθη, θα δείτε μηνύματα όπως αυτά μαρκαρισμένα με 2. Τα έχουμε δει και πιο πριν: θα λένε πόσο χώρο καταλαμβάνει το πρόγραμμα στη FLASH μνήμη του ελεγκτή όπως επίσης πόση μνήμη RAM καταναλώθηκε από τις μεταβλητές που χρησιμοποιήθηκαν. Επίσης αν το πρόγραμμα είναι σωστά γραμμένο θα γίνει εγγραφή αμέσως στη μνήμη FLASH του ελεγκτή. Αν υπάρχουν λάθη θα εμφανιστούν επίσης στο ίδιο παράθυρο. Θα δείτε τι λάθη γίναν όπως και που γίναν μέσα στο πρόγραμμα. Διορθώστε τα και μεταγλωττίστε το πρόγραμμα ξανά.

- **Έλεγχος**

Ακόμα και αν το περιβάλλον Arduino IDE δεν σας ενημερώσει για λάθη όταν τελειώσετε τη μεταγλώττιση δεν σημαίνει απαραίτητα ότι το πρόγραμμά σας θα δουλεύει σωστά. Το περιβάλλον ανιχνεύει μόνο συντακτικά λάθη όπως: μια συνάρτηση που δεν έχει γραφτεί σωστά ή δεν υπάρχει καθόλου, μεταβλητές που δεν έχουν οριστεί, δεδομένα που δεν είναι σωστά κτλ.

Η δύσκολη δοκιμή είναι όταν ελέγχετε ότι ο ελεγκτής κάνει ακριβώς ότι τον έχετε προγραμματίσει να κάνει. Αυτό το παράδειγμα είναι πολύ απλό: Θεωρήστε ότι τα BASIC I/O στη πλακέτα έχουν συνδεθεί σωστά στον ελεγκτή Arduino, αυτό που έχετε να κάνετε είναι να πατήσετε το κουμπί D4: αν το white LED που είναι συνδεδεμένο στον ακροδέκτη D6 φωτίσει, τότε όλα είναι καλά, αν όχι θα μείνει σβηστό.

Ένας τελευταίος έλεγχος: όταν γράφετε στη FLASH μνήμη ένα πρόγραμμα, μένει εκεί για πάντα ακόμη και αν απενεργοποιήσετε τον ελεγκτή. Το πρόγραμμα διαγράφεται μόνο όταν γράψετε ένα άλλο. Μια λεπτομέρεια: όταν το πρόγραμμα γραφτεί, το Arduino δεν βασίζεται στον υπολογιστή με κανένα τρόπο. Υπάρχει ένας πολύ εύκολος τρόπος για να το διαπιστώσετε: βγάλτε το καλώδιο USB από το Arduino UNO και συνδέστε σε μια συνηθισμένη πηγή ενέργειας ή σε μια μπαταρία όπως στην εικόνα. Θα διαπιστώσετε ότι όλα λειτουργούν όπως λειτουργούσαν και πριν.

Ίσως πιστεύετε ότι όλα αυτά δεν μετράνε πολύ και δεν χρειάζεται ένας ελεγκτής για να ανάβει ένα LED με τη χρήση ενός κουμπιού. Είστε λάθος. Κάνατε το πρώτο βήμα: Αν καταλάβατε την έννοια ενός προγράμματος και τις συναρτήσεις του, σίγουρα θα μπορέσετε να κάνετε πιο πολύπλοκα και ενδιαφέροντα πράγματα. Πρέπει όμως να αρχίσετε σιγά σιγά.

2. Παράδειγμα 2: Φωτισμός ενός LED 2 volt

Εδώ είναι μια άλλη εκδοχή του προηγούμενου παραδείγματος. Ελέγχει το white LED που είναι συνδεδεμένο στον ακροδέκτη 6 με το κουμπί που είναι συνδεδεμένο στον ακροδέκτη 4 στην ίδια πλακέτα. Παρατηρείστε την εικόνα και συγκρίνετε με το προηγούμενο παράδειγμα.

Η κύρια διαφορά είναι ότι δεν έχουμε χρησιμοποιήσει μεταβλητές για να ορίσουμε ποιος ακροδέκτης είναι συνδεδεμένος στο κουμπί και ποιος στο LED.

Ο ακροδέκτης που αναφέρετε στους αριθμούς 4 και 6 υπάρχει μέσα στις συναρτήσεις: **pinMode()**, **digitalRead()**, **digitalWrite()**.

Σε αυτή τη περίπτωση, οι αριθμοί των ακροδεκτών λέγονται “constants” (σταθερές) και όχι “variables” (μεταβλητές).

Όπως κάνατε και στο προηγούμενο παράδειγμα, μεταγλωττίστε και γράψτε το πρόγραμμα, μετά σιγουρευτείτε ότι δουλεύει σωστά.

- **Συμπεράσματα**

Η χρήση μεταβλητών ή σταθερών για τον ορισμό ακροδεκτών είναι στη κρίση σας. Για παράδειγμα, αν το κουμπί θα είναι μόνιμα συνδεδεμένο στον ακροδέκτη 4, τότε χρησιμοποιήστε αυτόν τον αριθμό σαν σταθερά όπως ακριβώς σε αυτό το παράδειγμα. Θα κάνετε οικονομία στη μνήμη RAM με αυτόν τον τρόπο.

Από την άλλη όμως, αν υπάρχει πιθανότητα να αλλαχτεί η συνδεσμολογία, τότε είναι καλύτερο να οριστούν ως μεταβλητές: για παράδειγμα “**intButton=4;**” .

Με αυτόν τον τρόπο αν το κουμπί συνδεθεί στον ακροδέκτη 12 στο μέλλον, το μόνο που θα χρειαστεί να αλλάξετε είναι η μεταβλητή “**intButton=12;**” και δε θα χρειαστεί να αλλάξετε τις συναρτήσεις.

3. Παράδειγμα 3: Φωτισμός ενός LED 3 volt

Εδώ είναι ακόμη ένα και τελευταίο παράδειγμα του προγράμματος για το έλεγχο ενός LED με ένα κουμπί.

Το πρώτο πράγμα που θα κάνετε είναι να μεταγλωττίσετε και να γράψετε το πρόγραμμα στον ελεγκτή. Σιγουρευτείτε ότι δουλεύει σωστά.

- **Συμπεράσματα**

Αν μελετήσατε το πρόγραμμα προσεκτικά θα δείτε ποιες είναι οι διαφορές ανάμεσα σε αυτό το παράδειγμα και τα παραδείγματα 4_1 και 4_2. Καταλήγουμε στα παρακάτω συμπεράσματα:

1. Καμία μεταβλητή δεν χρησιμοποιήθηκε, αυτό σημαίνει ότι αποδεσμεύσαμε χώρο από τη μνήμη RAM. Σε πολλές περιπτώσεις η χρήση μεταβλητών και σταθερών εξαρτάται από το τι θέλετε να κάνετε.

2. Οι ακροδέκτες που θα χρησιμοποιηθούν ως ψηφιακές είσοδοι δεν χρειάζεται να ρυθμιστούν από τη συνάρτηση **pinMode()**. Όλοι οι ακροδέκτες είναι αυτόματα ρυθμισμένοι ως είσοδοι.

3. Το αποτέλεσμα μιας συνάρτησης μπορεί να χρησιμοποιηθεί ως είσοδος σε μια άλλη συνάρτηση χωρίς να χρειάζεται μια ενδιάμεση μεταβλητή “Value” όπως στα άλλα παραδείγματα. Κοιτάχτε την παρακάτω έκφραση:

digitalWrite(6,digitalRead(4));

Υπάρχει μια συνάρτηση μέσα σε μια άλλη. Η εσωτερική συνάρτηση εκτελείτε πρώτη: **digitalRead(4)**. Το αποτέλεσμά της περνάει στην επόμενη **digitalWrite(,...);**.

4. Ένα πρόγραμμα μπορεί να γραφτεί με πάρα πολλούς τρόπους. Αυτή τη στιγμή πρέπει να ανησυχείτε για το αν το πρόγραμμα δουλεύει σωστά ή όχι, αυτό είναι το πιο σημαντικό. Με τον καιρό θα έχετε τη δυνατότητα να ελαχιστοποιήσετε τον χώρο

που θα χρησιμοποιείτε στη FLASH μνήμη και στη μνήμη RAM κι έτσι θα κάνετε το πρόγραμμά σας πιο αποτελεσματικό.

Μπορείτε να κάνετε έναν πολύ απλό έλεγχο από μόνοι σας. Μεταγλωττίστε και γράψτε πάλι τα παραδείγματα 4_1, 4_2 και 4_3. Παρατηρείστε πόση FLASH μνήμη και μνήμη RAM χρησιμοποιείτε. Θα παίρνετε αυτή τη πληροφορία από το περιβάλλον Arduino IDE κάθε φορά που θα γράφετε ένα πρόγραμμα, θα φαίνεται στο κάτω μέρος. Συμπληρώστε τον παρακάτω πίνακα:

ΠΑΡΑΔΕΙΓΜΑ	FLASH (Μνήμη Προγραμμάτων)	RAM (Μνήμη Δεδομένων)
ΠΑΡΑΔΕΙΓΜΑ_4_1		
ΠΑΡΑΔΕΙΓΜΑ _4_2		
ΠΑΡΑΔΕΙΓΜΑ _4_3		

4. Παράδειγμα 4: Αντίστροφος φωτισμός ενός LED

Η ιδέα είναι να χρησιμοποιήσουμε τον τελεστή NOT έτσι ώστε να κάνουμε ένα πρόγραμμα να φωτίζει το λευκό LED που είναι συνδεδεμένο στον ακροδέκτη 6 κάθε φορά που το κουμπί που είναι συνδεδεμένο στον ακροδέκτη 4 ΔΕΝ είναι πατημένο.

Το κύριο σώμα του προγράμματος βρίσκετε στη συνάρτηση **loop()** που είναι μαρκαρισμένη με κίτρινο χρώμα.

Μοιάζει πολύ με το προηγούμενο παράδειγμα. Η μόνη διαφορά είναι ότι η συνάρτηση NO προβάλλεται με το σύμβολο “**!**”, αντιστρέφει το διάβασμα του κουμπιού που είναι συνδεδεμένο στον ακροδέκτη 4.

Αυτό το παράδειγμα προσομοιάζει μια μηχανή με ένα μοτέρ που τρέχει συνεχώς εκτός και αν κάποιο κουμπί έκτακτης ανάγκης πατηθεί.

5. ΠΑΡΑΔΕΙΓΜΑ 5: Χειρισμός δύο κουμπιών

Φανταστείτε ότι θέλετε να ελέγξετε ένα συναγερμό με δυο κουμπιά που βρίσκονται μακριά το ένα από το άλλο. Ο συναγερμός πρέπει να κλείνει ή το ένα κουμπί ή το άλλο ή και τα δυο. Ο συναγερμός προσομοιάζεται με ένα κόκκινο LED στη πλακέτα BASIC I/O συνδεδεμένο στον ακροδέκτη 11. Τα δυο κουμπιά είναι συνδεδεμένα στους ακροδέκτες 4 και 12.

Έτσι φαίνεται το τελικό πρόγραμμα, ρίξτε του μια ματιά:

6. ΠΑΡΑΔΕΙΓΜΑ 6: Ατομική δουλειά

Χρησιμοποιήστε το παράδειγμα **EXAMPLE 4 6** ως μοντέλο για το πρόγραμμα (ή αλλιώς "template"). Κάντε τις απαραίτητες αλλαγές στο μοντέλο για να κάνετε τις ασκήσεις που ακολουθούν. Προσπαθήστε να χρησιμοποιήσετε όλα όσα είδαμε, μάθετε περισσότερα και περάστε καλά.

Όσο μαθαίνετε περισσότερα για τον προγραμματισμό, θα παρατηρήσετε ότι ο ελεγκτής είναι ικανός να κάνει πολύ περισσότερα από το απλά να φωτίζει ένα LED. Αυτό ήταν μόνο η αρχή!

Exercise	Description
1	The amber LED should go on when buttons 7 and 12 are pushed.
2	The green LED goes on when buttons 12 and 8 are pushed but not button 4.
3	The white LED goes on when button 4 is pushed but not button 7. The red LED goes on when buttons 8 or 12 are pushed.
4	The green LED goes on when button 4 is pushed but not button 7, and when button 8 is pushed but not button 12.

5	The red LED goes on when buttons 4, 7 and 8 are pushed or, on the other hand, button 12 is.

Θυμηθείτε το ακόλουθο: πρέπει να γράψετε ότι αλλαγή κάνετε στο περιβάλλον Arduino IDE.

Γράψτε τα στον ελεγκτή και σιγουρευτείτε ότι δουλεύουν σωστά σύμφωνα με τις οδηγίες τις κάθε άσκησης.

Συμπληρώστε τον ακόλουθο πίνακα.

Exercise	Function	Your program's instructions
1	**setup()**	
	loop()	
2	**setup()**	
	loop()	
3	**setup()**	
	loop()	
4	**setup()**	
	loop()	
5	**setup()**	
	loop()	

ΠΑΡΑΠΟΜΠΕΣ

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