ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Ενσωματωμένα Συστήματα

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ: CIRC-12

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

1. Σκοπός της άσκησης

- Χρήση του αναπτυξιακού περιβάλλοντος του Arduino[1] για τη συγγραφή και τη μεταφόρτωση προγραμμάτων στην πλακέτα του Arduino Uno.
- Δημιουργία κυκλώματος ελέγχου ενός RGB LED.
- Έλεγχος ενός Servo με τη χρήση φωτοαντιστάτη.

2. Παραδοτέα

- Ένα αρχείο zip με τα project **CIRC_12**, **CIRC_12_c1** που θα δημιουργήσετε.
- Ένα screenshot από το arduino IDE που δείχνει ότι η μεταγλώττιση (compile) έγινε με επιτυχία και ταυτόχρονα εμφανίζει το μέγεθος του δυαδικού σχεδίου για κάθε κύκλωμα.
- Ένα video 5-10 sec επίδειξης του κυκλώματος μαζί με ηχητική περιγραφή για κάθε κύκλωμα.

Χρόνος ολοκλήρωσης εργαστηρίου: 30 λεπτά.

Εκτός από τα LEDs που έχουμε δει ως τώρα, τα οποία είναι μονόχρωμα, υπάρχουν και LEDs τα οποία μπορούν να παράγουν πολλά χρώματα. Ονομάζονται RGB LEDs και είναι τυπικά τρία LEDs (κόκκινο, πράσινο και μπλε) σε ένα. Όταν ενεργοποιούνται ταυτόχρονα και τα τρία LEDs τα φώτα τους ενώνονται και προκύπτουν διάφορα χρώματα. Τα χρώματα που προκύπτουν εξαρτώνται από τη φωτεινότητα κάθε LED ξεχωριστά. Η φωτεινότητα ελέγχεται με διαμόρφωση πλάτους παλμού (PWM), η λειτουργία της οποίας έχει αναλυθεί στο εργαστήριο 3.

3. Κατασκευή του κυκλώματος CIRC-12

Για την εκπόνηση του κυκλώματος CIRC-12 απαιτούνται τα εξής μέρη:

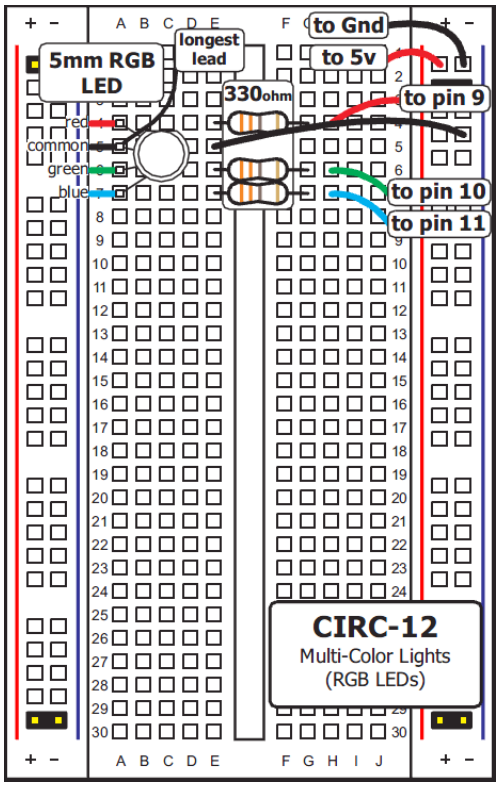
Στοιχείο	Περιγραφή και ποσότητα
	**1x** RGB LED
	**6x** Καλώδια
	**3x** Αντιστάσεις 330 Ohm ***(Πορτοκαλί – Πορτοκαλί – Καφέ)***

1 Χρησιμοποιήθηκε υλικό από το SparkFun Inventors Kit for Arduino

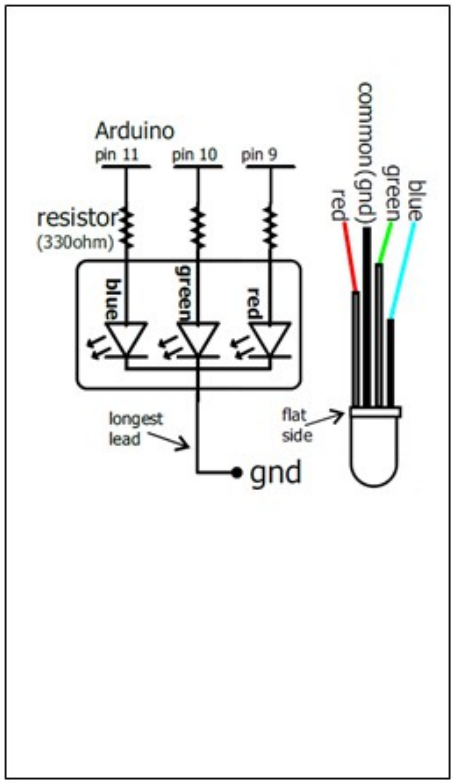
Τα φυλλάδια δεδομένων (datasheets) των υλικών βρίσκονται παρακάτω:

- RGB LED
- Αντίσταση 330 Ohm

Κατασκευάστε το κύκλωμα, σύμφωνα με τις παρακάτω σχηματικές αναπαραστάσεις:



Σχήμα 1



Σχήμα 2

Τοποθετήστε το RGB LED όπως φαίνεται στο *Σχήμα 1*.

(Τα pins του LED αναλύονται στο Σχήμα 2).

Στη συνέχεια συνδέστε από μία **αντίσταση 330 Ohm** σε **κάθε pin που αντιστοιχεί σε κάποιο χρώμα**, και έπειτα κάθε αντίσταση με το ανάλογο pin του Arduino. Επίσης συνδέστε και το pin της γείωσης.

4. Προγραμματισμός του κυκλώματος CIRC-12

Αντιγράψτε τον παρακάτω κώδικα στο προγραμματιστικό περιβάλλον του Arduino.

(Εναλλακτικά κατεβάστε τον από εδώ)

Κώδικας προγραμματισμού του κυκλώματος

```
/*

RGB_LED_Color_Fade_Cycle.pde

Cycles through the colors of a RGB LED

Written for SparkFun Arduino Inventor's Kit CIRC-RGB

*/

// LED leads connected to PWM pins
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;

// Used to store the current intensity level of the individual LEDs
int redIntensity = 0;
int greenIntensity = 0;
int blueIntensity = 0;

// Length of time we spend showing each color
const int DISPLAY_TIME = 100; // In milliseconds


void setup() {
 // No setup required.
}

void loop() {
// Cycle color from red through to green
// (In this loop we move from 100% red, 0% green to 0% red, 100% green)
 for (greenIntensity = 0; greenIntensity <= 255; greenIntensity+=5){
  redIntensity = 255-greenIntensity;
  analogWrite(GREEN_LED_PIN, greenIntensity);
  analogWrite(RED_LED_PIN, redIntensity);
  delay(DISPLAY_TIME);
 }

// Cycle color from green through to blue
// (In this loop we move from 100% green, 0% blue to 0% green, 100% blue)
 for (blueIntensity = 0; blueIntensity <= 255; blueIntensity+=5){
  greenIntensity = 255-blueIntensity;
  analogWrite(BLUE_LED_PIN, blueIntensity);
  analogWrite(GREEN_LED_PIN, greenIntensity);
  delay(DISPLAY_TIME);
}

// Cycle cycle from blue through to red
  // (In this loop we move from 100% blue, 0% red to 0% blue, 100% red)
  for (redIntensity = 0; redIntensity <= 255; redIntensity+=5) {
  blueIntensity = 255-redIntensity;
  analogWrite(RED_LED_PIN, redIntensity);
  analogWrite(BLUE_LED_PIN, blueIntensity);
  delay(DISPLAY_TIME);
}
}
```

Αποθηκεύστε το ως CIRC_12 και στη συνέχεια φορτώστε το πρόγραμμα στο Arduino.

Σε περίπτωση που το LED δεν ανάβει, ή δείχνει λάθος χρώματα, ελέγξτε αν το κύκλωμά σας είναι σωστό.

5. Παραμετροποίηση του κυκλώματος CIRC-12

5.1 Μείωση της υπερβολικής παρουσίας του κόκκινου χρώματος

Ενδέχεται το κόκκινο χρώμα του LED να είναι εντονότερο από τα άλλα και αυτό να έχει ως συνέπεια τα χρώματα που προκύπτουν να μην είναι τόσο ισορροπημένα. Αν συμβαίνει κάτι τέτοιο, μπορείτε να το διορθώσετε είτε βάζοντας μια αντίσταση περισσότερων Ohm στο pin του κόκκινου χρώματος, είτε αλλάζοντας την παρακάτω γραμμή στο πρόγραμμά σας, από:

```
analogWrite(RED_LED_PIN, redIntensity);
```

σε:

```
analogWrite(RED_LED_PIN, redIntensity/3);
```

Αποθηκεύστε (Ctrl+s) και φορτώστε ξανά το πρόγραμμα στο Arduino, για να επιβεβαιώσετε ότι το πρόβλημα λύθηκε.

5.2 Χρήση δεκαεξαδικών τριάδων για την απεικόνιση των χρωμάτων

Εάν έχετε δουλέψει με HTML ή CSS, πιθανότατα θα σας βολεύει περισσότερο να χρησιμοποιείτε δεκαεξαδικούς για να καθορίσετε το χρώμα που επιθυμείτε.

Σε κάθε περίπτωση πάντως μπορείτε να επισκεφθείτε το σχετικό λήμμα στη Wikipedia για περισσότερες πληροφορίες.

Για να μπορείτε να χρησιμοποιήσετε κι εδώ το ίδιο σύστημα, δημιουργήστε ένα καινούριο Arduino Sketch και αντιγράψτε τον παρακάτω κώδικα

(Εναλλακτικά κατεβάστε τον από εδώ)

Κώδικας προγραμματισμού του κυκλώματος

```
/*
Set_RGB_Color.pde
Shows how to use HTML-style hex triplet color codes to specify RGB LED colors
Written for SparkFun Arduino Inventor's Kit CIRC-RGB
*/

// LED leads connected to PWM pins
const int RED_LED_PIN = 9;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 11;

// HTML-style hex triplet color code values from Wikipedia
const unsigned long ORANGE = 0xFF7F00;
const unsigned long TEAL = 0x008080;
const unsigned long AUBERGINE = 0x614051;
```

Κώδικας προγραμματισμού του κυκλώματος

```
// Length of time we spend showing each color
const int DISPLAY_TIME = 2000; // In milliseconds

void setup() {
// No setup required.
}

void loop() {
// Cycle through our awesome colors

setColor(ORANGE);
delay(DISPLAY_TIME);

setColor(TEAL);
delay(DISPLAY_TIME);

setColor(AUBERGINE);
delay(DISPLAY_TIME);
}

void setColor(unsigned long color) {
/*
Sets the color of the RGB LED to the color specified by the
HTML-style hex triplet color value supplied to the function.

The color value supplied should be an unsigned long number of the form:

0xRRGGBB

e.g. 0xFF7F00 is orange

where:
 0x specifies the value is a hexadecimal number
 RR is a byte specifying the red intensity
 GG is a byte specifying the green intensity
 BB is a byte specifying the blue intensity

 How it works (you don't need to understand this to use the function):

 The supplied value of the form 0xRRGGBB is an unsigned long which can
 store four bytes. If we view the number in bit form the bits of the
 unsigned long look like this:

 iiiiiiiirrrrrrrrggggggggbbbbbbbb

where:

 i are bits that are ignored (because we only need three bytes for the value)
 r are bits specifying the red intensity
 g are bits specifying the green intensity
 b are bits specifying the blue intensity

 We use bit shifting and masking to extract the individual values.

 Bit shifting moves the bits in a number along in one direction to produce
 a new number. For example ">> n" means shift the bits n positions to the
 right.

 e.g. iiiiiiiirrrrrrrrggggggggbbbbbbbb >> 8 gives:

 00000000iiiiiiiirrrrrrrrgggggggg

 The number 0xFF represents the bit mask:

 00000000000000000000000011111111
```

Κώδικας προγραμματισμού του κυκλώματος

```
 The "&" character means the two values are "and"ed together--for each bit
 in the two numbers being "and"ed together the resulting bit is only set in
 the result if the same bits are set in both the input numbers.

 So, continuing our example:

 00000000iiiiiiiirrrrrrrrggggggggg && 0xFF gives:

 00000000iiiiiiiirrrrrrrrggggggggg &&
 00000000000000000000000011111111 =
 000000000000000000000000ggggggggg

And we now have extracted only the green bits from our hex triplet.

More details on bit math can be found here:
<http://www.arduino.cc/playground/Code/BitMath>

*/

// Extract the intensity values from the hex triplet
byte redIntensity = (color >> 16) & 0xFF;
byte greenIntensity = (color >> 8) & 0xFF;
byte blueIntensity = color & 0xFF;

// Display the requested color
 analogWrite(RED_LED_PIN, redIntensity);
 analogWrite(GREEN_LED_PIN, greenIntensity);
 analogWrite(BLUE_LED_PIN, blueIntensity);
}
```

Εκτός από τα παραπάνω, χρησιμοποιήστε τη συνάρτηση δημιουργίας τυχαίων αριθμών για την παραγωγή τυχαίων χρωμάτων.

Ορίστε μια νέα μεταβλητή: **long randNumber;**

Αρχικοποιήστε την συνάρτηση **randomSeed(analogRead(0));**

και λάβετε μια τυχαία τιμή: **randNumber = random(1300000);**

την οποία θα την εμφανίσετε με **setColor();**

Στη συνέχεια αποθηκεύστε το ως CIRC_12_c1 και φορτώστε το στο Arduino για να επιβεβαιώσετε την ορθή λειτουργία του. Εφόσον λειτουργεί σωστά μπορείτε να πειραματιστείτε δίνοντας τις δικές σας τιμές για τα χρώματα με έναν, ίσως, πιο γνώριμο για εσάς τρόπο.