Arduino – senzor de umiditate pentru plante

In acest tutorial vei descoperi cum se poate programa o placa Arduino sa citeasca si sa afiseze, printr-o matrice de led-uri, nivelul de umiditate din solul unei plante. Astfel, poti afla rapid daca planta are nevoie de apa sau nu.

Daca planta are nevoie de apa, atunci matricea de led-uri va desena o expresie umana care indica tristetea, iar daca planta are suficienta apa atunci expresia umana va fi fericita.

In ultima instanta, daca planta duce lipsa majora de apa, matricea va desena o figura moarta.

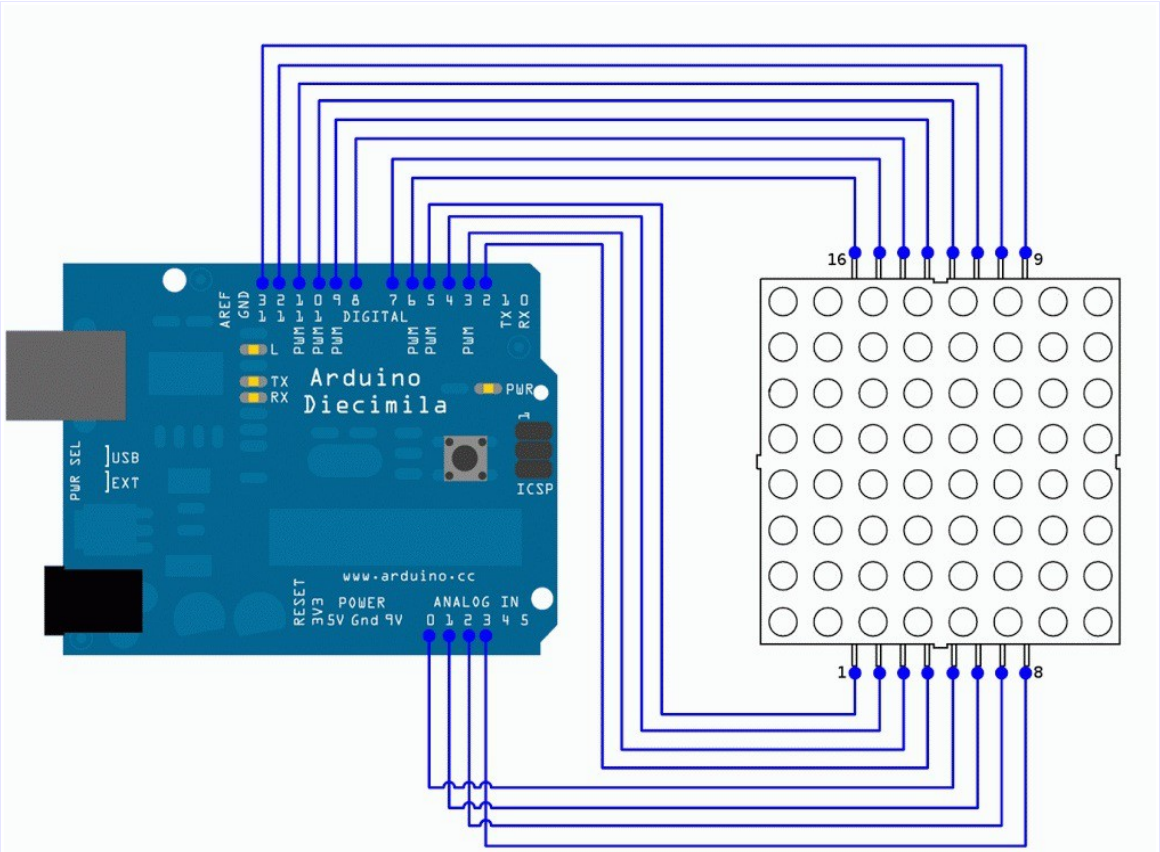
Pentru a realiza acest montaj vei avea nevoie de urmatoarele componente:

- O placa Arduino - http://www.robofun.ro/arduino
- O matrice de led-uri 8x8 - http://www.robofun.ro/electronice/led/matrice-leduri-8x8
- Un senzor de umiditate pe care il vei construi singur.
- Fire de conexiune diverse - http://www.robofun.ro/cabluri

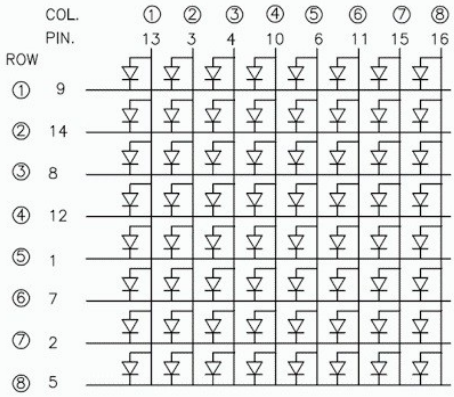
Cum se conecteaza matricea de led-uri ?

Matricea de led-uri se conecteaza direct la porturile placii Arduino. Iti recomand totusi sa conectezi cate un rezistor de 330 de ohmi in serie cu fiecare coloana a matricei. In felul asta vei proteja porturile placii in eventualitatea unui scurt circuit.

Urmeaza diagrama de mai jos.



Column and row anode and cathode wiring to pins in matrix



Cum se construieste senzorul de umiditate ?

Principiul de functionare al senzorului este simplu. Vei avea nevoie de 2 conductoare pe care le vei infige direct in pamantul din ghiveci. Conductoarele formeaza un rezistor a carei rezistenta variaza in functie de umiditatea pamantului.

Ca si conductoare poti folosi o placuta de cablaj dublu strat. Fiecare strat de cupru se comporta ca un electrod.

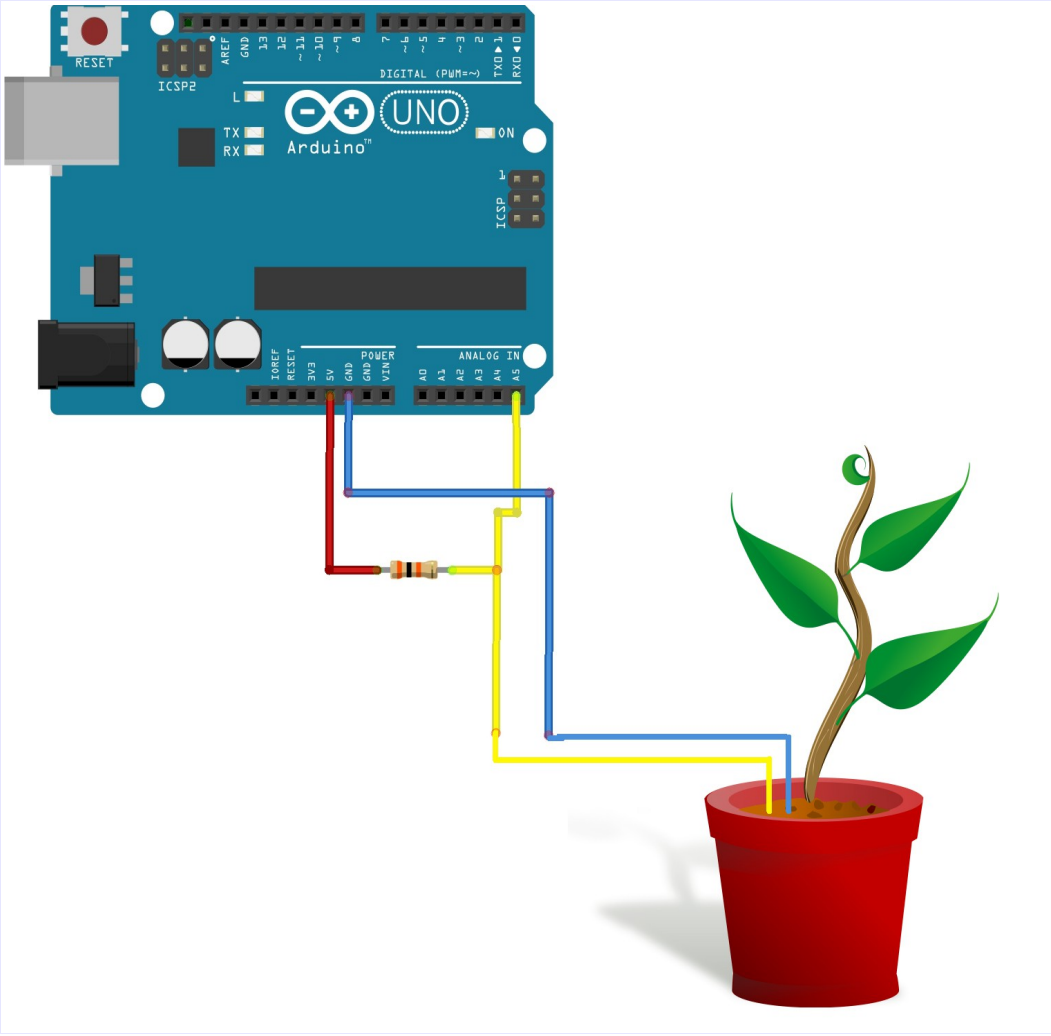
Pe langa acest rezistor vei mai avea nevoie de un rezistor secundar pe care il vei determina si va avea o valoare fixa. Vei conecta cele 2 rezistoare in configuratie de divizor de tensiune.

In primul rand vei masura rezistenta dintre cele 2 conductoare atunci cand pamantul este uscat si cand este ud. In cazul de fata s-au masurat valori cuprinse intre 10 si 100 kΩ. Pentru rezistorul fix o valoare de 30 kΩ este ok.

Scopul placii Arduino este sa culeaga tensiunea de pe rezistorul variabil (senzorul de umiditate). Divizorul de tensiune este un senzor a carui iesire este liniara cu marimea de intrare, adica umiditatea.

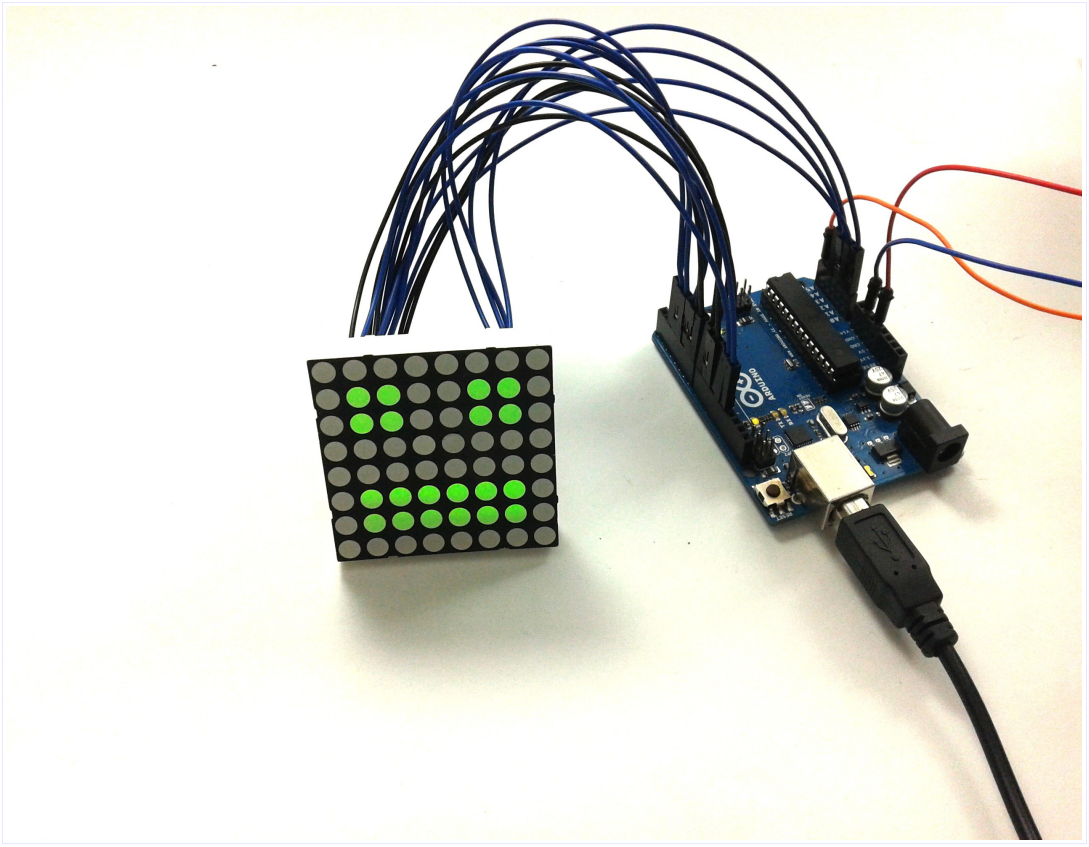
Cu acest avantaj placa Arduino poate determina simplu si rapid, fara calcule necesare, care este nivelul de umiditate din pamant.

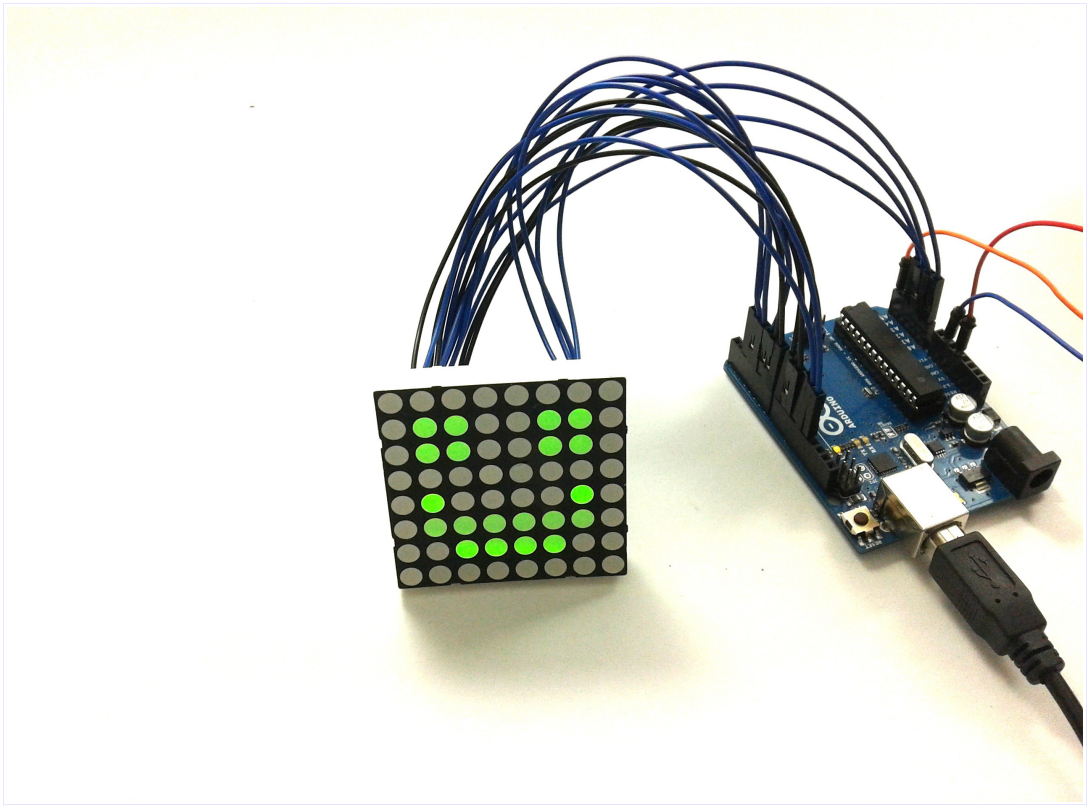
Mai jos este diagrama senzorului de umiditate.

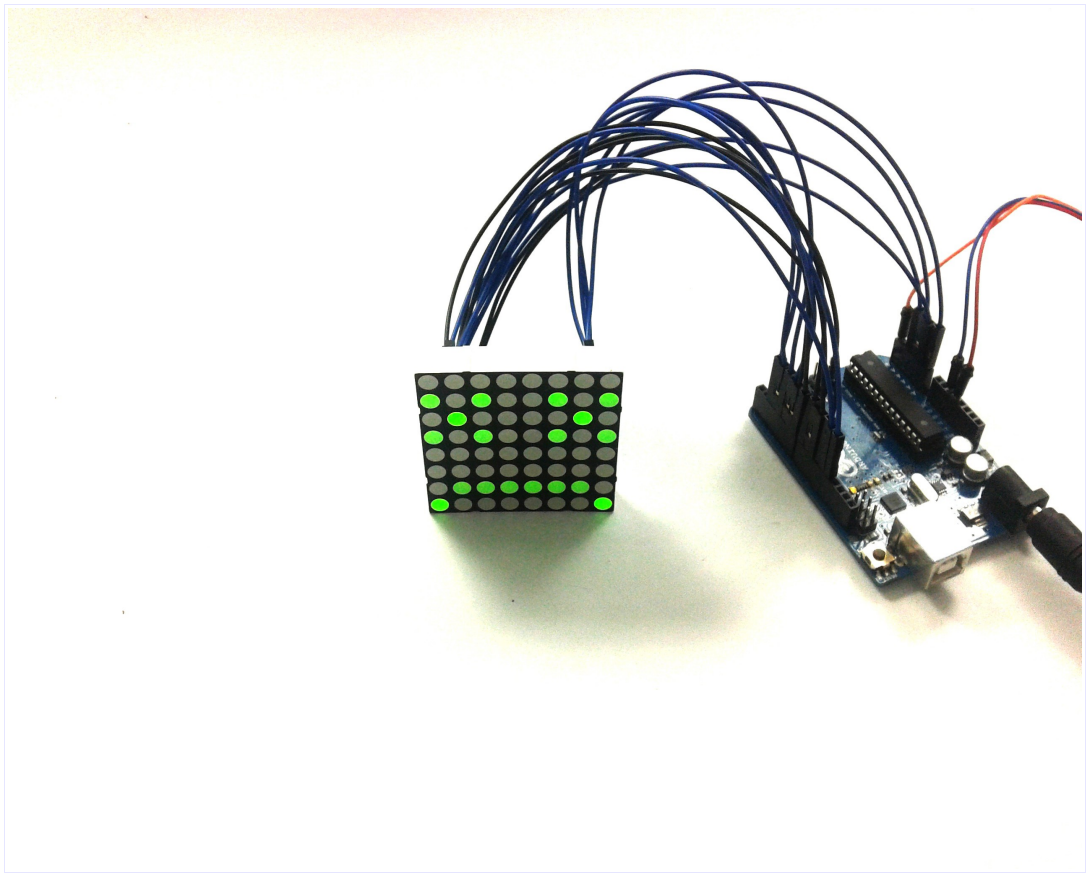


Sketch-ul pentru afisarea umiditatii ?

Tot ce iti ramane este sa incarci sketch-ul listat mai jos. Il copiezi cu Copy/Paste si il incarci direct in placa Arduino. Dupa cateva momente, matricea va afisa starea plantei in functie de nivelul de umiditate din sol. Daca nivelul este suficient de mare, matricea iti va indica aceasta stare printr-o expresie fericita. Daca nivelul este mediu sau scazut, expresiile se vor schimba.







```
#include <FrequencyTimer2.h>

#define SAD { \
    {0, 0, 0, 0, 0, 0, 0, 0},  \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 1, 1, 1, 1, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0} \
}

#define HAPPY { \
    {0, 0, 0, 0, 0, 0, 0, 0},  \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 0, 1, 1, 1, 1, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0} \
}
```

```
#define MEH { \
    {0, 0, 0, 0, 0, 0, 0, 0},  \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 1, 1, 0, 0, 1, 1, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0} \
}

#define DEAD { \
    {0, 0, 0, 0, 0, 0, 0, 0},  \
    {1, 0, 1, 0, 0, 1, 0, 1}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {1, 0, 1, 0, 0, 1, 0, 1}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {1, 0, 0, 0, 0, 0, 0, 1} \
}

byte col = 0;
byte leds[8][8];

// pin[xx] on led matrix connected to nn on Arduino (-1 is dummy to
make array start at pos 1)
int pins[17]= {-1, 5, 4, 3, 2, 14, 15, 16, 17, 13, 12, 11, 10, 9,
8, 7, 6};

// col[xx] of leds = pin yy on led matrix
int cols[8] = {pins[13], pins[3], pins[4], pins[10], pins[06],
pins[11], pins[15], pins[16]};

// row[xx] of leds = pin yy on led matrix
int rows[8] = {pins[9], pins[14], pins[8], pins[12], pins[1],
pins[7], pins[2], pins[5]};

const int numPatterns = 6;
byte patterns[numPatterns][8][8] = {
  HAPPY, MEH, SAD, DEAD
};

int pattern = 0;
```

```
void setup() {
  Serial.begin(9600);
  // sets the pins as output
  for (int i = 1; i <= 16; i++) {
    pinMode(pins[i], OUTPUT);
  }

  // set up cols and rows
  for (int i = 1; i <= 8; i++) {
    digitalWrite(cols[i - 1], LOW);
  }

  for (int i = 1; i <= 8; i++) {
    digitalWrite(rows[i - 1], LOW);
  }

  clearLeds();

  // Turn off toggling of pin 11
  FrequencyTimer2::disable();
  // Set refresh rate (interrupt timeout period)
  FrequencyTimer2::setPeriod(2000);
  // Set interrupt routine to be called
  FrequencyTimer2::setOnOverflow(display);

  setPattern(pattern);
}

void loop() {
    int pragUmiditateSuperior = 700;
    int pragUmiditateMediu = 500;
    int pragUmiditateInferior = 300;

    int nivelUmiditate = analogRead(5);


    Serial.println(nivelUmiditate);
    if (nivelUmiditate > pragUmiditateSuperior) {
       setPattern(0); // happy face
    } else if (nivelUmiditate < pragUmiditateSuperior &&
nivelUmiditate > pragUmiditateMediu) {
      setPattern(1); // meh face
    } else if (nivelUmiditate < pragUmiditateMediu &&
nivelUmiditate > pragUmiditateInferior) {
      setPattern(2); // sad face
    } else if (nivelUmiditate < pragUmiditateInferior) {
      setPattern(3); // dead face
    }
```

```
    delay(1000);
}

void clearLeds() {
  // Clear display array
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      leds[i][j] = 0;
    }
  }
}

void setPattern(int pattern) {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      leds[i][j] = patterns[pattern][i][j];
    }
  }
}

// Interrupt routine
void display() {
  digitalWrite(cols[col], LOW);  // Turn whole previous column off
  col++;
  if (col == 8) {
    col = 0;
  }
  for (int row = 0; row < 8; row++) {
    if (leds[col][7 - row] == 1) {
      digitalWrite(rows[row], LOW);  // Turn on this led
    }
    else {
      digitalWrite(rows[row], HIGH); // Turn off this led
    }
  }
  digitalWrite(cols[col], HIGH); // Turn whole column on at once
(for equal lighting times)
}
```