

Smart DeskBin

Introducere

Prezentarea pe scurt a proiectului:

- ce face: este un cos de birou cu deschidere automata si interactiune user-friendly
- care este scopul lui: utilizarea atractiva a unui accesoriu de birou
- care a fost ideea de la care ați pornit: pe internet exista varianta unui cos cu deschidere automata implementat cu Arduino, caruia am hotarat sa ii adaug mai multe feature-uri
- de ce credeți că este util pentru alții și pentru voi: este un obiect folosit in mod frecvent si caruia i se pot aduce multe imbunatatiri

Descriere generală

In aceasta sectiune voi descrie modul de functionare al DeskBin-ului, alaturi de o schema bloc. In schema voi prezenta interactiunea placutei Arduino cu toate componentele secundare din proiect

Acest DeskBin are drept functionalitate principala deschiderea automata. Acest lucru se realizeaza in felul urmator:

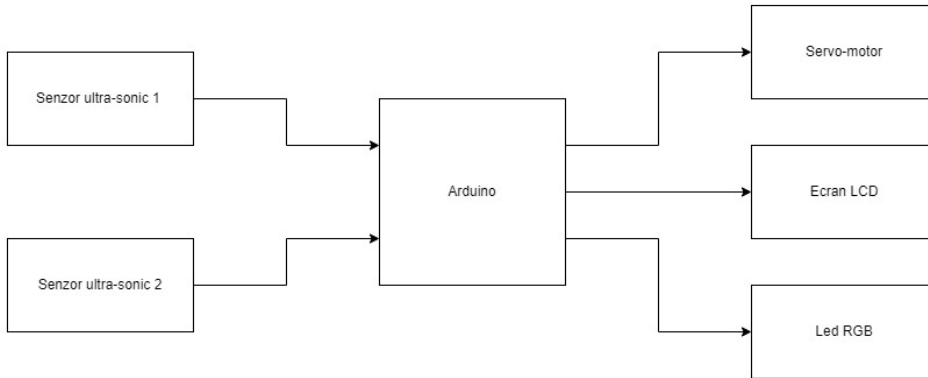
- primul senzor ultra-sonic detecteaza mana aflata la o distanta de 5-20 cm
- acest lucru activeaza servo-motorul care va ridica, actionand printr-un brat de plastic, capacul de la cos
- capacul sta deschis timp de 5s, apoi se inchide

A doua functionalitate consta in raportarea nivelului de umplere. Aceste date sunt preluate de cel de-al doilea senzor ultra-sonic positionat in capacul cosului, fiind legat la un led RGB. Astfel, daca DeskBin-ul nu e plin, led-ul RGB va sta aprins pe verde, respectiv daca este plin, va sta aprins pe rosu, dezactivand senzorul de deschidere pana va fi golit.

A treia functionalitate este cea a ecranului LCD. In functie de nivelul de umplere, va afisa emoji-uri cu diferite stari:

- smiley - cand nu este plin
- satiated - cand este plin
- eating - cand se deschide cosul

Diagrama bloc

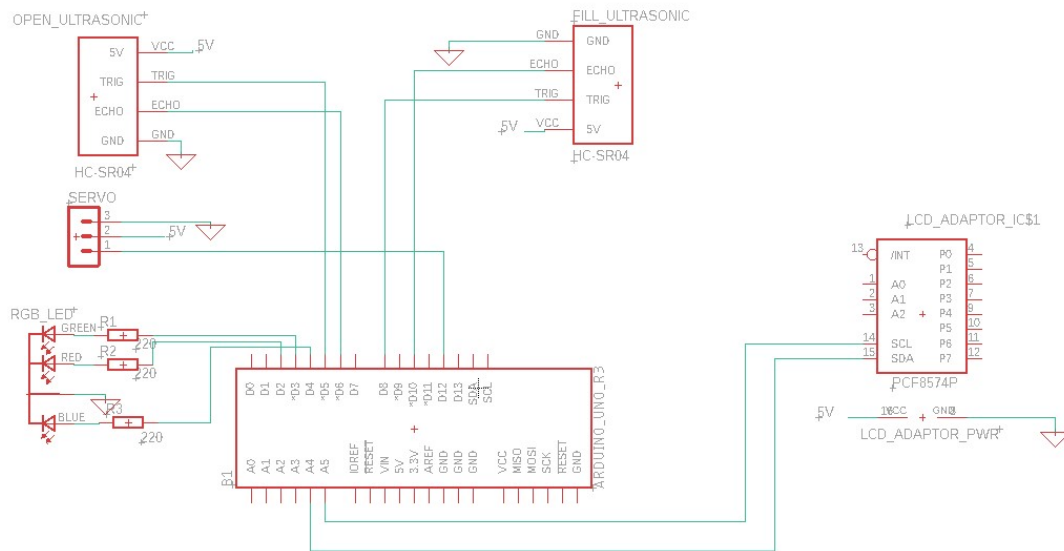


Hardware Design

Lista de componente

Componenta	Cantitate
Arduino UNO R3	1
Servo-motor SG90	1
Senzor ultra-sonic HC-SR04	2
Display LCD 2004A IIC	1
Adaptor display PCF8574	1
Led RGB	1
Fire T-T, M-T	

Schema circuit



In schema am inclus adaptorul, deoarece incorporat fiind in componenta LCD, am gasit potrivit sa reprezint strict partea ce contribuie la legare.

Software Design

Descrierea codului aplicației (firmware):

- mediu de dezvoltare: Arduino IDE 2.1.0
- librării și surse 3rd-party: Servo.h, Wire.h, LiquidCrystal_I2C.h
- pentru generare de emoji-uri: <https://maxpomer.github.io/LCD-Character-Creator/>

Inainte de toate, am definit:

- macro-uri pentru pinii dedicati
- macro-uri pentru constante (distanțe)
- variabile globale (array-uri de bytes pentru emoji-uri, variabilele servo si lcd)

Realizarea emoji-urilor a constat in gruparea blocurilor de pixeli, astfel ganditi, ca laolalta sa genereze expresii.

In functia **setup()** realizez urmatoarele:

- initializarea display-ului
- crearea tuturor blocurilor necesare emoji-urilor
- initializarea servo-motorului
- initializarea pin-urilor pentru INPUT, respectiv OUTPUT

In functia **loop()** se urmeaza acest curs de executie:

- pentru **ultrasonic_deschidere**: resetare + masurat distanta -> activare servo
- pentru **ultrasonic_umplere**: resetare + masurat distanta -> activare LED
- setare emoji in functie de stare

Observatii

In cazul senzorului ultrasonic pentru deschidere, daca distanta este intre 5 si 20 cm (parametru calculat in functie de timp), se va apela **my_servo.write()** pe servomotor.

Iar in cazul senzorului ultrasonic pentru nivelul de umplere, culoarea va fi setata prin **digitalWrite(pin, level)**. Totodata, datorita structurii if din loop, se permite deschiderea automata a cosului doar dca inca este spatiu.

Functia **set_emoji()** este apelata ultima in loop, deoarece ea depinde tot ce a fost executat pana atunci. Practic se reevalueaza conditiile de umplere si deschidere pentru a afisa pe LCD un emoji definit anterior. Altfel, default va exista un emoji smiley :)

Mai jos atasez codul:

```
#include <Servo.h>

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define SERVO 12
#define ECHO_PIN_DIST 5
#define TRIGGER_PIN_DIST 6
#define ECHO_PIN_FULL 10
#define TRIGGER_PIN_FULL 8
#define RED_PIN 2
#define BLUE_PIN 3
#define GREEN_PIN 4
#define FULL 15

#define MIN_DIST 5
#define MAX_DIST 20

// servo
Servo my_servo;

// display
LiquidCrystal_I2C lcd(0x27,20,4); // set the LCD address to 0x27 for a 16
chars and 2 line display
```

```
byte eyeLeftUp[] = {
    0x0F,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10
};

byte eyeLeftDown[] = {
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x0F
};

byte eyeRightUp[] = {
    0x1E,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01
};

byte eyeRightDown[] = {
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x01,
    0x1E
};

byte eyeVerticalLeft[] = {
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
    0x10,
```

```
    0x10,  
    0x10  
};  
  
byte eyeVerticalRight[] = {  
    0x01,  
    0x01,  
    0x01,  
    0x01,  
    0x01,  
    0x01,  
    0x01,  
    0x01  
};  
  
byte eyeHorizontalUp[] = {  
    0x1F,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00  
};  
  
byte eyeHorizontalDown[] = {  
    0x00,  
    0x00,  
    0x1F,  
    0x1F,  
    0x1F,  
    0x1F,  
    0x1F,  
    0x1F  
};  
  
byte smileyMouth[] = {  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x10,  
    0x11,  
    0x0E  
};  
  
byte eatingMouth[] = {  
    0x00,
```

```
    0x00,  
    0x0E,  
    0x11,  
    0x11,  
    0x11,  
    0x11,  
    0x0E  
};  
  
byte sleepyEye[] = {  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x1F,  
    0x1F  
};  
  
byte satiatedMouth[] = {  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00,  
    0x00  
};  
  
// Reset trigger to LOW  
void reset_trigger(int trigger_pin)  
{  
    digitalWrite(trigger_pin, LOW);  
    delayMicroseconds(5000);  
    digitalWrite(trigger_pin, HIGH);  
    delayMicroseconds(5000);  
    digitalWrite(trigger_pin, LOW);  
}  
  
void createSatiatedMouth() {  
    lcd.createChar(12, satiatedMouth);  
}  
  
void printSatiatedMouth() {  
    lcd.setCursor(9, 3);  
    lcd.write(12);  
}  
  
void createSmileyMouth() {
```

```
lcd.createChar(9, smileyMouth);

}

void printSmileyMouth() {
    lcd.setCursor(9, 3);
    lcd.write(9);
}

void createOpenMouth() {
    lcd.createChar(10, eatingMouth);
}

void printOpenMouth() {
    lcd.setCursor(9, 3);
    lcd.write(10);
}

void createSleepyEye() {
    lcd.createChar(11, sleepyEye);
}

void printSleepyEye(int val) {

    int i = val;

    lcd.setCursor(i + 3, 2);
    lcd.write(11);
    lcd.setCursor(i + 4, 2);
    lcd.write(11);
}

void printSleepyEyes() {
    printSleepyEye(0);
    printSleepyEye(12);
}

// construct every component of emoji
void createNormalEye() {

    lcd.createChar(1, eyeLeftUp);

    lcd.createChar(2, eyeLeftDown);

    lcd.createChar(3, eyeRightUp);

    lcd.createChar(4, eyeRightDown);
```



```
    lcd.createChar(5, eyeVerticalLeft);  
    lcd.createChar(6, eyeVerticalRight);  
    lcd.createChar(7, eyeHorizontalUp);  
    lcd.createChar(8, eyeHorizontalDown);  
  
}  
  
void printNormalEye(int val) {  
    int i = val;  
  
    lcd.setCursor(i + 2, 0);  
    lcd.write(1);  
  
    lcd.setCursor(i + 2, 2);  
    lcd.write(2);  
  
    lcd.setCursor(i + 5, 0);  
    lcd.write(3);  
  
    lcd.setCursor(i + 5, 2);  
    lcd.write(4);  
  
    lcd.setCursor(i + 2, 1);  
    lcd.write(5);  
  
    lcd.setCursor(i + 5, 1);  
    lcd.write(6);  
  
    lcd.setCursor(i + 3, 0);  
    lcd.write(7);  
    lcd.setCursor(i + 4, 0);  
    lcd.write(7);  
  
    lcd.setCursor(i + 3, 2);  
    lcd.write(8);  
    lcd.setCursor(i + 4, 2);
```

```
    lcd.write(8);
}

void printNormalEyes() {
    printNormalEye(0);
    printNormalEye(12);
}

/* print emojis functions */
void printEatingEmoji() {
    printNormalEyes();
    printOpenMouth();
}

void printSleepingEmoji() {
    printSleepyEyes();
    printOpenMouth();
}

void printSmileyEmoji() {
    printNormalEyes();
    printSmileyMouth();
}

void printSatiatedEmoji() {
    printNormalEyes();
    printSatiatedMouth();
}

void setup()
{
    // initialize the lcd
    lcd.init();
    lcd.backlight();
    lcd.clear();

    // init face components
    createNormalEye();
    createSleepyEye();
    createSmileyMouth();
    createOpenMouth();
    createSatiatedMouth();

    // init servo
    my_servo.attach(SERV0);

    // init open ultrasonic sensor
```

```
pinMode(TRIGGER_PIN_DIST, OUTPUT);
pinMode(ECHO_PIN_DIST, INPUT);

// init full ultrasonic
pinMode(TRIGGER_PIN_FULL, OUTPUT);
pinMode(ECHO_PIN_FULL, INPUT);

// init rgb led
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);

delay(100);
Serial.begin(9600);
}

float get_distance(int echo_pin)
{
    float dist;

    // set pin
    pinMode(echo_pin, INPUT);

    // get distance to hand
    long duration = pulseIn(echo_pin, HIGH);
    // distance in cm (converted from time)
    dist = duration / 58.82;
    return dist;
}

void open_trash_by_hand(float dist)
{
    int pos;

    if (dist <= MAX_DIST && dist >= MIN_DIST) {
        pos = 90;
        my_servo.write(pos);
        delay(50);
    }
    else {
        pos = 0;
        my_servo.write(pos);
    }
}

// check if deskbin is full
void check_content(float dist_full) {
    if (dist_full <= FULL) {
        // turn red led on
        digitalWrite(RED_PIN, HIGH);
        digitalWrite(GREEN_PIN, LOW);
    }
}
```

```
}
else {
    // turn green led on
    digitalWrite(RED_PIN, LOW);
    digitalWrite(GREEN_PIN, HIGH);
    delay(1000);
}
}

void set_emoji(int dist_hand, int dist_full) {
    if (dist_full <= FULL) {
        printSatiatedEmoji();
        delay(100);
    } else if (dist_hand <= MAX_DIST && dist_hand >= MIN_DIST) {
        printEatingEmoji();
        delay(500);
    } else {
        printSmileyEmoji();
    }
}

void loop()
{
    /* full ultrasonic */
    // reset
    reset_trigger(TRIGGER_PIN_FULL);

    // get dist(sensor, garbage)
    float dist_full = get_distance(ECHO_PIN_FULL);

    // activate led depending on status
    check_content(dist_full);

    float dist_hand;

    if (dist_full <= FULL) {
        // is full
    } else {
        /* open sensor*/
        // reset
        reset_trigger(TRIGGER_PIN_DIST);

        // get dist(sensor, hand)
        dist_hand = get_distance(ECHO_PIN_DIST);

        // activate servo if dist_hand in range(MIN_DIST, MAX_DIST)
        open_trash_by_hand(dist_hand);
    }

    /* set emoji */
}
```

```
set_emoji(dist_hand, dist_full);  
}
```

Rezultate Obținute

Smart Deskbin-ul proiectat este unul complet functional, care, pentru o functionare foarte lina, responsive, ar avea nevoie de un surplus de tensiune pe anumite componente, deoarece alimentandu-le de la acelasi 5V, pot aparea intarzieri de reactie.

Cand exista spatiu si este in asteptare, apare **smiley_emoji**, iar ledul este **verde**



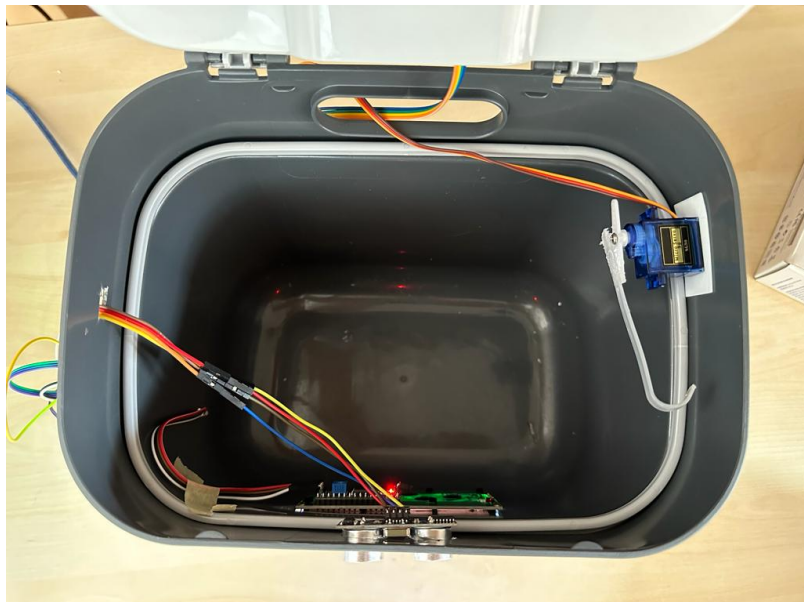
Cand este deschis, apare **eating_emoji**

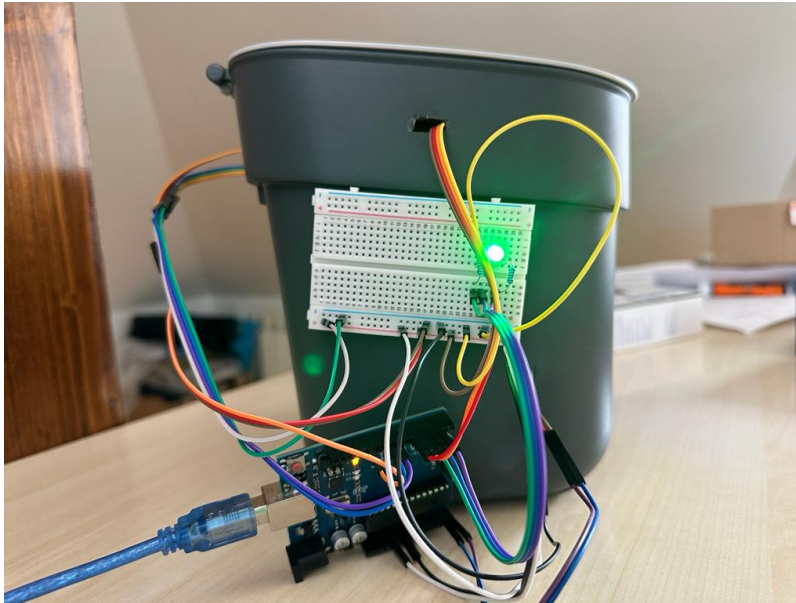


Cand nu mai exista spatiu, apare **satiated_emoji**, iar ledul devine **rosu**



Iar aici, alte detalii





Concluzii

Realizarea acestui proiect a fost o experienta frumoasa si utila. A fost primul meu proiect cu micro-controller si pot spune ca este un sentiment rewarding in final.

Principala dificultate pe care am intampinat-o a fost gasirea pe internet a modului de legare pentru un LCD 2.8 inch ILI9341, fara pin CS. Drept urmare, am gasit alternativa LCD 2004A IIC, care s-a pretat foarte bine genului de aplicatie al proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite:

- emoji blocks: <https://maxpromer.github.io/LCD-Character-Creator/>
- legare LCD: https://www.youtube.com/watch?v=F9IVtKa8C7Q&ab_channel=educ8s.tv
- alte detalii: https://www.youtube.com/watch?v=9yrP1CZN3Ds&ab_channel=IndianLifeHacker

[Export to PDF](#)

From:
<http://ocw.cs.pub.ro/courses/> - **CS Open CourseWare**

Permanent link:
<http://ocw.cs.pub.ro/courses/pm/prj2023/ncaroi/smart-deskbin>



Last update: **2023/05/27 10:36**

