

# Tom the Musical and Talking Robot

## Introducere

- Autor: Oana Maria Băcăran
- Grupa: 334CB

### \* **Ce face?**

Tom este o jucărie interactivă, care știe să imite ceea ce spui, poate să redea cântecele și poate realiza efecte vizuale, redând mici animații pe un ecran OLED.

### \* **Care este scopul lui?**

Scopul acestui proiect este acela de a implementa o jucărie amuzantă, folosind și materiale reciclabile.

### \* **Care a fost ideea de la care ați pornit?**

Inspirația acestui proiect a venit gândindu-mă la o jucărie asemănătoare din copilărie, un hamster vorbitor. A fost o jucărie unică, pe care a apreciat-o toată familia.

### \* **De ce credeți că este util pentru alții și pentru voi?**

Acest proiect este util și interesant deoarece nu numai că poate fi folosit ca o modalitate distractivă pentru copii de a-și petrece timpul, dar poate fi folosit și pe post de mini music player, cu o înfățișare hazlie și unică.

## Descriere generală

Mod de funcționare:

Roboțelul are două moduri de utilizare:

1. Modul ascultare și redare, în care acesta primește un input audio de la utilizator, apăsând pe butonul de rec (înregistrare) și redă ulterior ceea ce a înregistrat, cu ajutorul unui difuzor. Butoanele necesare se află pe modulul audio ISD1820.
2. Modul ascultare muzică, unde utilizatorul poate folosi butoanele de play/pause, next și previous ca să navigheze printre fișierele stocate pe un card sd; se folosește un card reader compatibil SPI pentru a reda melodiile.

Pentru a adăuga un strop de culoare și dinamsim, în ambele moduri, pe ecranul OLED se vor afișa mici animații, care vor alterna constant și vor fi randomizate în modul music player.

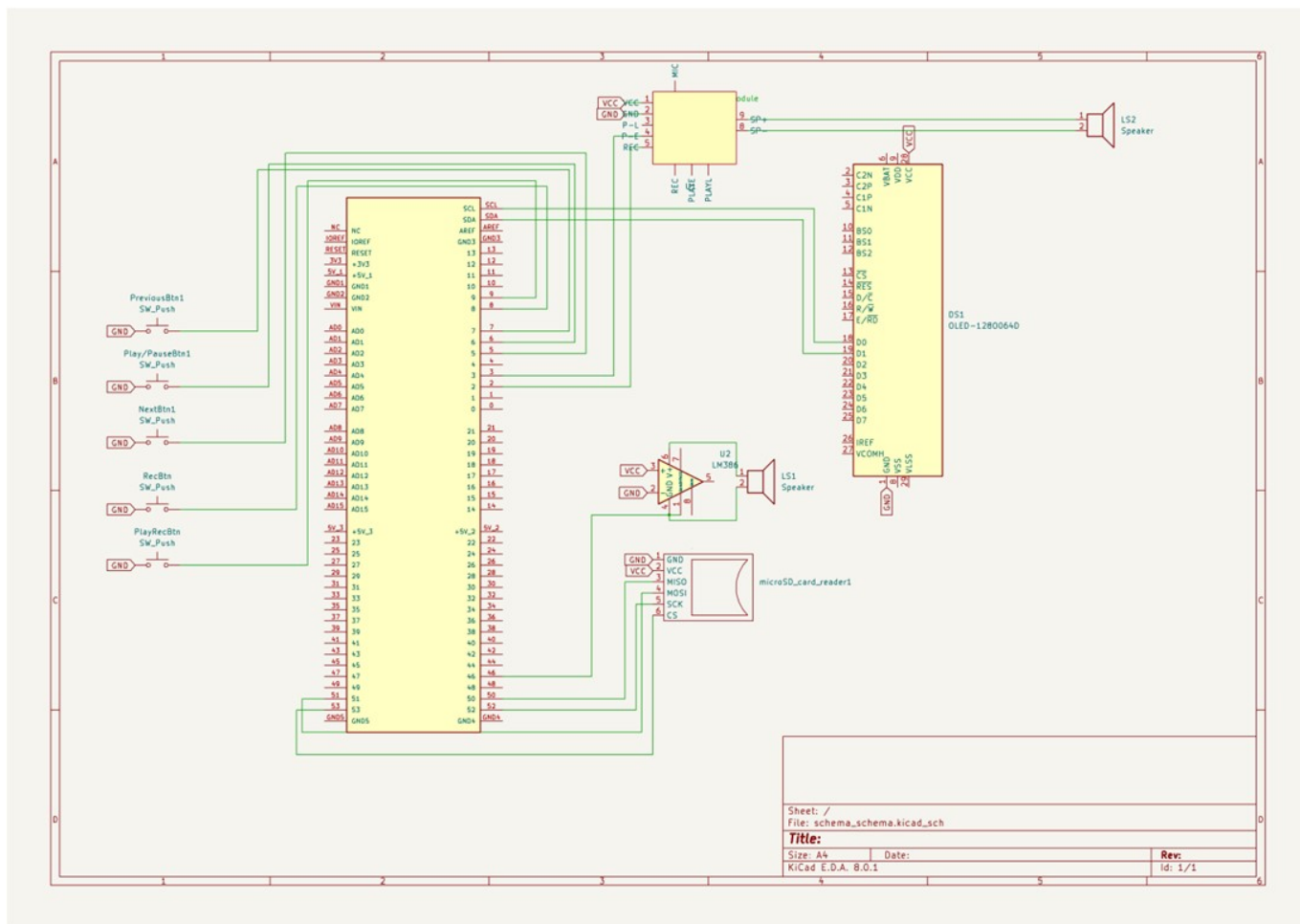


# Hardware Design

## Lista piese:

- Arduino ATmega2560
- Modul inregistrare redare audio ISD1820
- Modul Amplificator Audio LM386
- 2 x Difuzor
- Display grafic OLED
- Placa de stocare Micro SD TF Card reader Shield
- Card SD
- 5 x push buttons
- fire conectare

## Schema electrică:

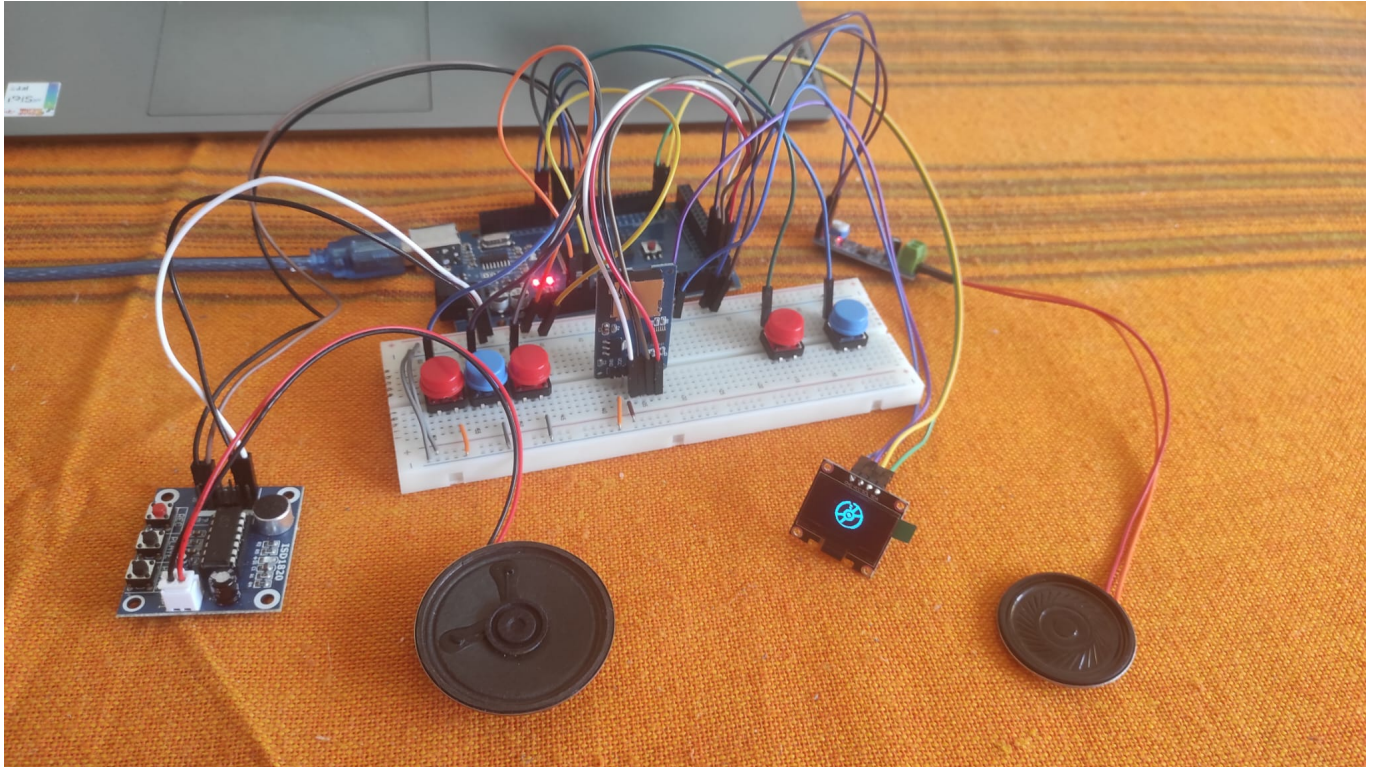


## Descriere pini:

- Ecran OLED: conectat la pinii corespunzatori protocolului I2C, SCL(PD0) si SDA(PD1), conform datasheet-ului
- Modul ISD1820: conectat la pinii GPIO 2 si 3
- MicroSD Card Reader: foloseste protocolul SPI, iar pinii corespunzatori acestuia sunt: D53 pentru CS, D52 pentru SCK, D51 pentru MOSI, D50 pentru MISO

- LM385: acesta are nevoie de un pin de input pentru speaker si foloseste pinul D46
- Butoanele: folosesc pinii digitali 5,6,7,8,9

### Componentele conectate:



## Software Design

### Mediu de dezvoltare:

- dezvoltare cod: Arduino IDE
- animații: Wokwi animations
- realizare schemă bloc: draw.io
- realizare schemă electrică: Kicad

### Biblioteci:

- SD.h - ofera suport pentru initializarea card reader-ului si citirea cardului microSD
- TMRpcm.h - folosita pentru redarea, intreruperea audio-urilor
- SPI.h - folosita pentru cititorul de card SD
- Adafruit\_GFX.h - ofera primitive pentru grafica
- Adafruit\_SSD1306.h - folosita pentru manipularea animatiilor de pe display-ul OLED, specifica pentru tipul de ecran OLED
- Animations.h - stochez frame-urile animatiilor

```
#include <SD.h>
#include <TMRpcm.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
#include <SPI.h>
#include "Animations.h"
```

### Configurari si variabile globale:

Pentru display, am initializat o instanta a clasei Adafruit\_SSD1306, aceasta este variabila care controleaza display-ul. De asemenea, am setat si variabilele de measurements, precum viteza de redare a frame-urilor, latimea si lungimea animatiei in ecran si o variabila currentAnimation, care va stoca o referinta la o animatie aleasa random, pentru partea de music player.

```
#define SCREEN_I2C_ADDR 0x3C // or 0x3C
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RST_PIN -1 // Reset pin (-1 if not available)

Adafruit_SSD1306 display(128, 64, &Wire, OLED_RST_PIN);

#define FRAME_DELAY (42)
#define FRAME_WIDTH (48)
#define FRAME_HEIGHT (48)
#define FRAME_COUNT (sizeof(recording) / sizeof(recording[0]))
const byte (*currentAnimation)[288];

int frame = 0;
int frame_music = 0;
```

Pentru partea de redare a audio-urilor, am setat variabilele necesare butoanelor, pentru controlarea audio-ului (setarea calitatii audiolui, a volumului, pornirea si oprirea acestuia), a directorului in care se afla fisierele .wav si doua variabile care stocheaza numarul de audio-uri si indexul melodiei curente, incepand de la 1.

```
#define SD_CS_PIN 53
#define SPEAKER_PIN 46
#define PREV_PIN 7
#define PLAY_PIN 6
#define NEXT_PIN 5
#define FILENAME_SIZE 20

TMRpcm audio;
File root;

int numSongs = 0;
int songIdx = 1;
```

Alte variabile initializate au legatura cu partea de inregistrare a vocii, de la butoanele de control, pana la doua variabile bool care blocheaza folosirea inermitemta a celor doua moduri de functionare.

```
int rec=2;
int play=3;
int button_rec=8;
int button_play=9;
```

```
bool music_player_on = false;
bool recording_on = false;
```

## Setup:

```
void setup() {

    // MUSIC PLAYER PART
    Serial.begin(9600);

    // initialize buttons
    pinMode(PREV_PIN, INPUT_PULLUP);
    pinMode(PLAY_PIN, INPUT_PULLUP);
    pinMode(NEXT_PIN, INPUT_PULLUP);

    Serial.print(F("Initializing SD card..."));

    // check if card is initialized correctly
    if (!SD.begin(SD_CS_PIN)) {
        Serial.println(F("Failed to initialize SD card"));
        while(true); // stay here.
    }

    Serial.println("SD card is OK!");

    // open the directory and count the number of .wav/.WAV files
    root = SD.open("/");
    Serial.print(F("Number of .wav files: "));
    numSongs = numberAudios(root);
    Serial.println(numSongs);

    // set audio
    audio.speakerPin = SPEAKER_PIN;
    audio.setVolume(6);
    audio.quality(0);
    audio.stopPlayback();

    // ISD + DISPLAY PART
    pinMode(rec,OUTPUT);
    pinMode(play,OUTPUT);
    pinMode(button_rec,INPUT_PULLUP);
    pinMode(button_play,INPUT_PULLUP);

    // init display
    display.begin(SSD1306_SWITCHCAPVCC, SCREEN_I2C_ADDR);
    display.clearDisplay();
    display.display();

    // init isd
    digitalWrite(rec,LOW);
```

```
digitalWrite(play,LOW);

// check if any button is pressed at start point,
// wait for them to not be pressed, helps for debouncing
if (digitalRead(button_rec) == LOW) {
  while (digitalRead(button_rec) == LOW);
}
if (digitalRead(button_play) == LOW) {
  while (digitalRead(button_play) == LOW);
}

if (digitalRead(PREV_PIN) == LOW) {
  while (digitalRead(PREV_PIN) == LOW);
}
if (digitalRead(PLAY_PIN) == LOW) {
  while (digitalRead(PLAY_PIN) == LOW);
}

if (digitalRead(NEXT_PIN) == LOW) {
  while (digitalRead(NEXT_PIN) == LOW);
}

// for random animations
randomSeed(analogRead(0));

}
```

In functia de setup, se initializeaza butoanele la INPUT\_PULLUP, este deschis fisierul ce stocheaza audio-urile, este initializat isd-ul si display-ul si se verifica daca nu cumva un buton este apasat in timpul setup-ului. Pentru cititorul de card SD, am formatat cardul de 1GB in format FAT, iar fisierele le-am convertit in format .wav cu setarile: Samples Per second (Hz): 16000, Channel: Mono, Bits Per Sample: 8PCM, format: PCM unsigned 8-bit.

### Implementare:

Functia loop():

In functia loop, se afla tratarea cazurilor in care apasam unul din cele 5 butoane. Pentru music player, daca apasam unul din butoanele previous sau next, cu functia playAudio ne vom muta la anteriorul, respectiv urmatorul audio, scazand sau adunand 1 la indexul audio-ului curent. Audio-urile pot fi accesate in loop (cand ultima melodie va fi redata ne putem intoarce la prima) si vom selecta de fiecare data cand apasam aceste butoane o animatie random cu functia selectRandomAnimation(). Apasam butonul play pentru a da start primei melodii dupa setup, dar si pentru a intrerupe melodia. Cand melodia se termina, ecranul se va opri de asemenea. Pentru a da replay, apasam tot butonul play. Pentru a realiza efectul de debounce, am dat un mic delay. Cat timp o melodie este in desfasurare, nu ne putem inregistra vocea, si viceversa.

Pentru partea de inregistrare si redare a ceea ce spunem, apasam butonul rec cat timp vorbim. In timp ce vorbim, o animatie se va derula. Apasam butonul play recording pentru derularea inregistrarii. O animatie de 4 secunde se va declansa cand vom apasa acest buton.

```
void loop() {
  // put your main code here, to run repeatedly:
```

```
int prevState = digitalRead(PREV_PIN);
int playState = digitalRead(PLAY_PIN);
int nextState = digitalRead(NEXT_PIN);

// MUSIC PLAYER BUTTONS
if (recording_on == false) {
  if (prevState == LOW) {
    // debounce
    delay(100);

    if (songIdx == 1) {
      playAudio(numSongs);
    } else {
      playAudio(songIdx - 1);
    }
    music_player_on = true;
    selectRandomAnimation();
    updateAnimation();
  } else if (playState == LOW) {
    // debounce
    delay(100);

    if (audio.isPlaying()) {
      if(music_player_on) {
        audio.pause();
        music_player_on = false;
        stopAnimation();
      } else {
        audio.pause();
        music_player_on = true;
        updateAnimation();
      }
    } else {
      playAudio(songIdx);
      music_player_on = true;
      selectRandomAnimation();
      updateAnimation();
    }
  } else if (nextState == LOW) {
    // debounce
    delay(100);

    if (songIdx == numSongs) {
      playAudio(1);
    } else {
      playAudio(songIdx + 1);
    }
    music_player_on = true;
    selectRandomAnimation();
  }
}
```

```
        updateAnimation();
    }
}

// ISD BUTTONS
if (music_player_on == false) {
    if (digitalRead(button_rec) == LOW) {

        digitalWrite(rec, HIGH);
        recording_on = true;

        while (digitalRead(button_rec) == LOW) {
            display.clearDisplay();
            display.drawBitmap(40, 8, recording[frame], FRAME_WIDTH,
FRAME_HEIGHT, 1);
            display.display();
            frame = (frame + 1) % FRAME_COUNT;
            delay(FRAME_DELAY);
        }

        digitalWrite(rec, LOW);
        recording_on = false;
        delay(50); // Debounce delay
        display.clearDisplay();
        display.display();
    }

    else if (digitalRead(button_play) == LOW) {

        digitalWrite(play, HIGH);
        while (digitalRead(button_play) == LOW) {
            delay(10);
        }

        unsigned long startTime = millis();
        unsigned long playbackDuration = 4000;

        while (millis() - startTime < playbackDuration) {
            display.clearDisplay();
            display.drawBitmap(40, 8, talking[frame], FRAME_WIDTH,
FRAME_HEIGHT, 1);
            display.display();
            frame = (frame + 1) % FRAME_COUNT;
            delay(FRAME_DELAY);
        }

        digitalWrite(play, LOW);
        display.clearDisplay();
        display.display();
    }
}
```

```

}

// stop animation when audio stops
if (music_player_on && !audio.isPlaying()) {
    music_player_on = false;
    stopAnimation();
}
// play animation while playing music
else if (music_player_on && audio.isPlaying()) {
    updateAnimation();
} else if (!music_player_on) {
    display.clearDisplay();
    display.display();
}
}
}

```

Functia numberAudios(): Pentru a numara cate audio-uri avem in directorul root, parcurgem fiecare fisier din director si verificam daca string-ul format din numele fisierului se termina in extensia .wav .Daca da, marim contorul.

```

int numberAudios(File dir) {
    int count = 0;

    for (File file = dir.openNextFile(); file; file = dir.openNextFile()) {
        if (!file.isDirectory()) {
            String name = file.name();
            if (name.endsWith(".wav") || name.endsWith(".WAV")) {
                count++;
            }
        }
        file.close();
    }

    return count;
}

```

Functia playAudio(): Reda audio-ul cu index-ul curent, oprind orice alt audio care s-ar derula in acel moment.

```

void playAudio(int index) {
    audio.stopPlayback();

    // make the name of the file
    songIdx = index;
    char fileName[FILENAME_SIZE];
    snprintf(fileName, FILENAME_SIZE, "song_%d.wav", index);

    audio.play(fileName);
}

```

Functia selectRandomAnimation(): Alege un numar din 3 si, in functie de numarul ales, reda o

animatie.

```
void selectRandomAnimation() {
    int animationChoice = random(3);
    switch (animationChoice) {
        case 0:
            currentAnimation = cd;

            break;
        case 1:
            currentAnimation = tape;

            break;
        case 2:
            currentAnimation = music;

            break;
    }
}
```

Functia `updateAnimation()`: Aceasta șterge ecranul, desenează cadrul curent al animației, actualizează ecranul pentru a afișa noul cadru, avansează la următorul cadru și introduce o pauză scurtă pentru a controla viteza animației. `% FRAME_COUNT` asigură că, dacă `frame_music` ajunge la `FRAME_COUNT`, acesta revine la 0 (creând o animație în buclă).

```
void updateAnimation() {
    display.clearDisplay();
    display.drawBitmap(40, 8, currentAnimation[frame_music], FRAME_WIDTH,
FRAME_HEIGHT, 1);
    display.display();
    frame_music = (frame_music + 1) % FRAME_COUNT;
    delay(FRAME_DELAY);
}
```

Functia `stopAnimation()`:

```
void stopAnimation() {
    display.clearDisplay();
    display.display();
}
```


### Optimizări:

Am mutat variabilele constante care stocau animatiile in PROGMEM.

### Probleme intampinate:

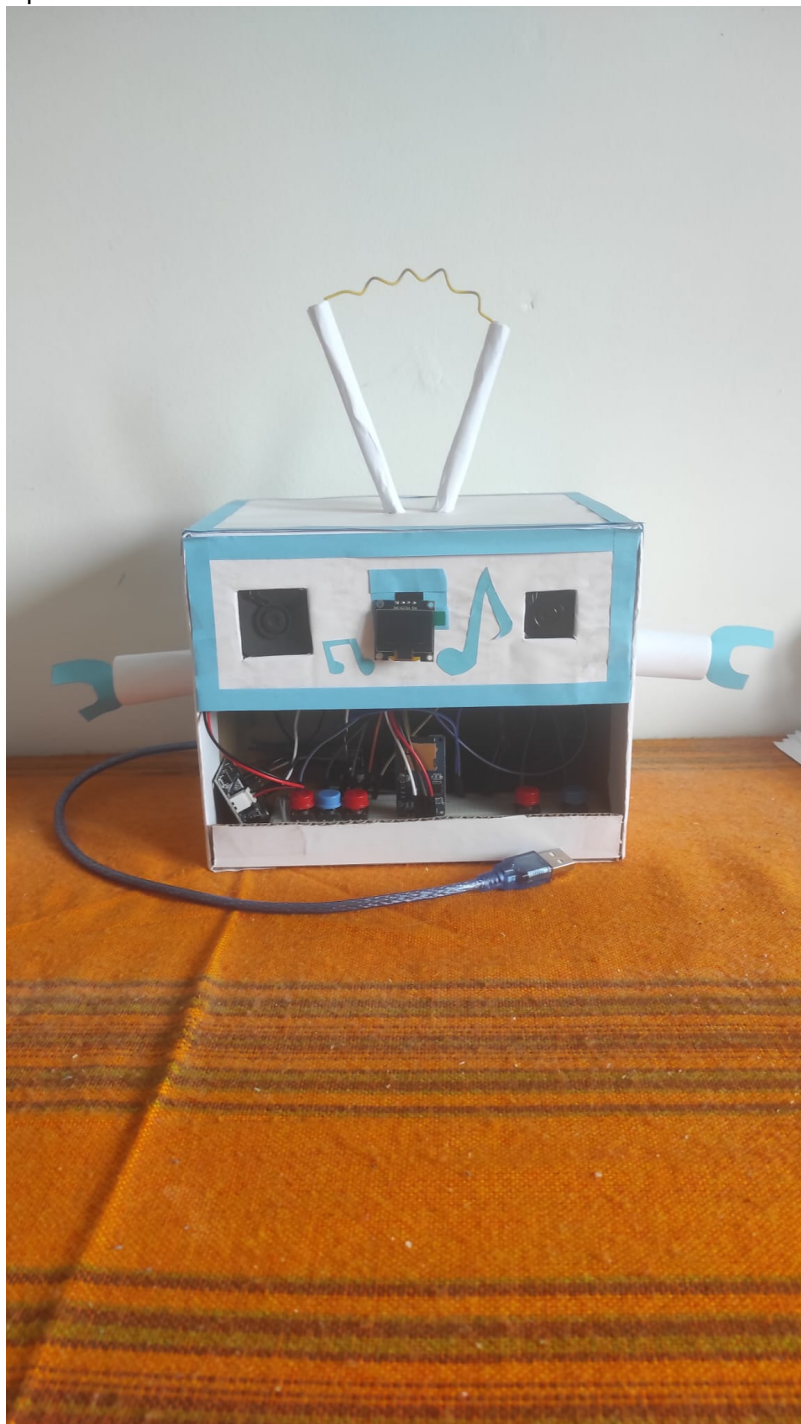
Codul a fost prea mare in ciuda optimizarilor, nu am mai avut loc pe memoria flash pe placuta Arduino Uno. Solutia cea mai smart si rapida a fost sa schimb placuta cu Arduino Mega, unde totul a mers smooth.

## Rezultate Obținute

Am reușit să realizez o jucărie dragută pentru copii, care poate să cânte și să repete tot ce spui. De asemenea, animațiile variate adaugă dinamism proiectului. Tot ce mai trebuie terminat este carcasa din carton a robotului (voi pune poze, lipiciul durează cam mult să se usuce ).

[https://youtu.be/oykXTy-1w\\_U](https://youtu.be/oykXTy-1w_U)

Update:



## Concluzii

A fost un proiect interesant, aparte de temele pe care le-am realizat pana acum. Desi acest proiect a venit si cu stresul constant ca piesele mi se pot defecta sau ca pot veni deja defecte (inca mai am aceasta spaima pana voi prezenta la PM fair), m-am distrat cel mai mult la partea de scris cod si de implementat functionalitati diverse, iar sentimentul pe care l-am avut la final, cand am vazut ca merge ce mi-am propus, a fost unul satisfactor.

## Download

[tom.zip](#)

## Jurnal

02-03 mai - realizare descriere generala a proiectului, listarea componentelor necesare, diagrama generala si schema electrica

06 mai - achizitionare piese

09-10 mai - primire piese

14-16 mai - realizare asamblare hardware

23 mai - achizitionarea unor elemente noi

21-25 mai - scrierea codului

26 mai - realizarea robotelului de carton

## Bibliografie/Resurse

### Datasheet:

[Arduino ATmega2560 Datasheet](#)

### Resurse Software:

[Animations](#)

[Music player inspiration](#)

### Resurse Hardware:

[CardSD setup](#)

[OLED setup](#)

[ISD setup](#)

[Export to PDF](#)

From:

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

Permanent link:

[http://ocw.cs.pub.ro/courses/pm/prj2024/azamfir/oana\\_maria.bacaran](http://ocw.cs.pub.ro/courses/pm/prj2024/azamfir/oana_maria.bacaran)



Last update: **2024/05/27 12:28**