Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

PROIECT ARHITECTURA SI ORGANIZAREA CALCULATOARELOR

GameBoy

Profesor: Carp Marius Catalin
Student: Neagu Lucian-Alexandru
Grupa: 4LF291
Specializarea: Electronica Aplicata
Anul de studiu: 2021-2022

Cuprins

Introducere

Denumirea proiectului:

- GameBoy

Alegerea proiectului:

Am ales acest proiect deoarece de mic copil am fost pasionat de jocurile video , acestea influențându-mi viața de-a lungul anilor . O data cu trecere timpului am căpătat o pasiune pentru programare si un respect profund pentru jocurile retro si jocurile făcut în stil retro . Așa că am ales această tema de proiect ca un omagiu adus veteranilor industriei care au creat primele jocuri si respectiv primele console fără de care nu s-ar fi putut realiza nimic.

Proiectul consta într-un joculeț simplu (2D pixel-at) pe un ecran LCD 16x2 în care protagonistul (o mașinuță) are de parcurs o cursă cu obstacole alcătuită din blocuri generate random peste care trebuie să sară fără a atinge obstacolul .

Stadiul actual al domeniului

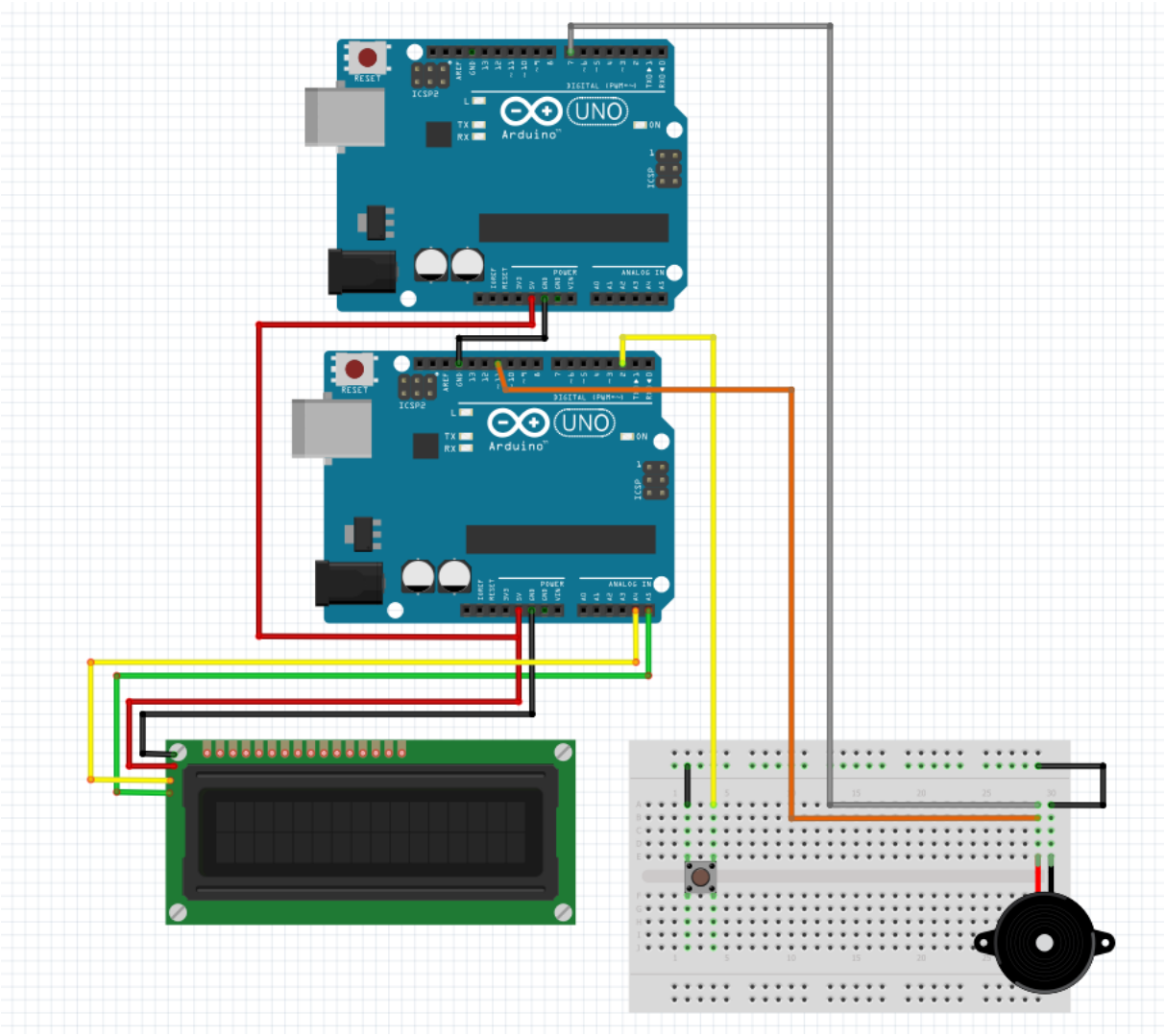
Domeniul gaming-ului se împarte în momentul actual în trei categorii având în vedere tehnologiile hardware si software existente: PC gaming , console gaming și VR gaming .

În cea ce ține de console cele mai performante disponibile la ora actuală pe piață sunt PS5 și Xbox Series X . PS5 fiind echipat cu un CPU AMD Ryzen Zen 8 cores , un GPU AMD Radeon RDNA 2 cu Ray Tracing Acceleration , care poate produce o rezoluție 4K la 120fps și o memorie RAM de 16GB GDDR6/256-bit , iar Xbox Series X deși are componente similare în cea ce ține de CPU și GPU are o memorie RAM mult mai rapidă și posibilitatea de a ajunge la rezoluția de 8K sacrificând jumătate din numărul de fps-uri .

Cât despre PC-uri , cele mai bune CPU-uri achiziționabile pentru gaming sunt Intel Core i9-12900KS și AMD Ryzen 7 5800X3D , iar în materie de GPU cea mai performantă placă este Nvidia GeForce RTX 3090 Ti .

În materie de VR avem HTC Vive Pro 2 care are cea mai bună rezoluție , dar și cel mai mare preț . Domeniul VR este încă la început așa că se așteaptă în viitor multe îmbunătățiri și optimizări .

Arhitectura sistemului



Plăcuțele Arduino UNO sunt programate și alimentate de la laptop .

Pe plăcuța A rulează codul jocului și este conectată la ecranul LCD (care afiseaza grafica jocului) , la buton (controlează acțiunile caracterului) si la buzzer (output pentru sunetul de acțiune în joc) , precum si la plăcuța B .

Pe plăcuța B rulează codul muzicii de fundal , este alimentata de la plăcuța A și la buzzer , care este output pentru muzică .

Componenta Software

Cod Arduino plăcuța A (jocul)

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

#define PIN_BUTTON 2
#define PIN_AUTOPLAY 1
#define SPRITE_RUN1 1  //Car sprite
#define SPRITE_RUN2 2
#define SPRITE_JUMP 3
#define SPRITE_JUMP_UPPER '.'
#define SPRITE_JUMP_LOWER 4
#define SPRITE_TERRAIN_EMPTY ' '
#define SPRITE_TERRAIN_SOLID 5
#define SPRITE_TERRAIN_SOLID_RIGHT 6
#define SPRITE_TERRAIN_SOLID_LEFT 7
#define CAR_HORIZONTAL_POSITION 1    // Horizontal position of CAR on screen
#define TERRAIN_WIDTH 16
#define TERRAIN_EMPTY 0
#define TERRAIN_LOWER_BLOCK 1
#define TERRAIN_UPPER_BLOCK 2
#define CAR_POSITION_OFF 0        // CAR is invisible
#define CAR_POSITION_RUN_LOWER_1 1  // CAR is running on lower row (pose 1)
#define CAR_POSITION_RUN_LOWER_2 2  //                      (pose 2)
#define CAR_POSITION_JUMP_1 3      // Starting a jump
#define CAR_POSITION_JUMP_2 4      // Half-way up
#define CAR_POSITION_JUMP_3 5      // Jump is on upper row
#define CAR_POSITION_JUMP_4 6      // Jump is on upper row
#define CAR_POSITION_JUMP_5 7      // Jump is on upper row
#define CAR_POSITION_JUMP_6 8      // Jump is on upper row
#define CAR_POSITION_JUMP_7 9      // Half-way down
#define CAR_POSITION_JUMP_8 10     // About to land
#define CAR_POSITION_RUN_UPPER_1 11 // CAR is running on upper row (pose 1)
#define CAR_POSITION_RUN_UPPER_2 12 //                      (pose 2)
LiquidCrystal_I2C lcd(0x27, 16, 2);
static char terrainUpper[TERRAIN_WIDTH + 1];
static char terrainLower[TERRAIN_WIDTH + 1];
static bool buttonPushed = false;
void initializeGraphics() {
  static byte graphics[] = {
    // Run position 1
    0b00000,
    0b00000,
```

```
0b00100,
0b11110,
0b11110,
0b11111,
0b01001,
0b00000,
// Run position 2
0b00000,
0b00000,
0b00100,
0b11110,
0b11110,
0b11111,
0b01001,
0b00000,
// Jump
0b00000,
0b00000,
0b00100,
0b11110,
0b11110,
0b11111,
0b01001,
0b00000,
// Jump lower
0b00000,
0b00000,
0b00100,
0b11110,
0b11110,
0b11111,
0b01001,
0b00000,
// Ground
0b01110,
0b11111,
0b10101,
0b11111,
0b10101,
0b11111,
0b10101,
0b11111,
// Ground right
0b01110,
0b11111,
0b10101,
0b11111,
```

```
    0b10101,
    0b11111,
    0b10101,
    0b11111,
    // Ground left
    0b01110,
    0b11111,
    0b10101,
    0b11111,
    0b10101,
    0b11111,
    0b10101,
    0b11111,
  };
  int i;
  // Skip using character 0, this allows lcd.print() to be used to
  // quickly draw multiple characters
  for (i = 0; i < 7; ++i) {
    lcd.createChar(i + 1, &graphics[i * 8]);
  }
  for (i = 0; i < TERRAIN_WIDTH; ++i) {
    terrainUpper[i] = SPRITE_TERRAIN_EMPTY;
    terrainLower[i] = SPRITE_TERRAIN_EMPTY;
  }
}

// Slide the terrain to the left in half-character increments
//
void advanceTerrain(char* terrain, byte newTerrain) {
  for (int i = 0; i < TERRAIN_WIDTH; ++i) {
    char current = terrain[i];
    char next = (i == TERRAIN_WIDTH - 1) ? newTerrain : terrain[i + 1];
    switch (current) {
      case SPRITE_TERRAIN_EMPTY:
        terrain[i] = (next == SPRITE_TERRAIN_SOLID) ? SPRITE_TERRAIN_SOLID_RIGHT :
SPRITE_TERRAIN_EMPTY;
        break;
      case SPRITE_TERRAIN_SOLID:
        terrain[i] = (next == SPRITE_TERRAIN_EMPTY) ? SPRITE_TERRAIN_SOLID_LEFT :
SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_RIGHT:
        terrain[i] = SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_LEFT:
        terrain[i] = SPRITE_TERRAIN_EMPTY;
        break;
```

```
   }
  }
}

bool drawCAR(byte position, char* terrainUpper, char* terrainLower, unsigned int score) {
  bool collide = false;
  char upperSave = terrainUpper[CAR_HORIZONTAL_POSITION];
  char lowerSave = terrainLower[CAR_HORIZONTAL_POSITION];
  byte upper, lower;
  switch (position) {
    case CAR_POSITION_OFF:
      upper = lower = SPRITE_TERRAIN_EMPTY;
      break;
    case CAR_POSITION_RUN_LOWER_1:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN1;
      break;
    case CAR_POSITION_RUN_LOWER_2:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN2;
      break;
    case CAR_POSITION_JUMP_1:
    case CAR_POSITION_JUMP_8:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_JUMP;
      break;
    case CAR_POSITION_JUMP_2:
    case CAR_POSITION_JUMP_7:
      upper = SPRITE_JUMP_UPPER;
      lower = SPRITE_JUMP_LOWER;
      break;
    case CAR_POSITION_JUMP_3:
    case CAR_POSITION_JUMP_4:
    case CAR_POSITION_JUMP_5:
    case CAR_POSITION_JUMP_6:
      upper = SPRITE_JUMP;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case CAR_POSITION_RUN_UPPER_1:
      upper = SPRITE_RUN1;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case CAR_POSITION_RUN_UPPER_2:
      upper = SPRITE_RUN2;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
  }
```

```
  if (upper != ' ') {
   terrainUpper[CAR_HORIZONTAL_POSITION] = upper;
   collide = (upperSave == SPRITE_TERRAIN_EMPTY) ? false : true;
  }
  if (lower != ' ') {
   terrainLower[CAR_HORIZONTAL_POSITION] = lower;
   collide |= (lowerSave == SPRITE_TERRAIN_EMPTY) ? false : true;
  }

  byte digits = (score > 9999) ? 5 : (score > 999) ? 4 : (score > 99) ? 3 : (score > 9) ? 2 : 1;

  // Draw the scene
  terrainUpper[TERRAIN_WIDTH] = '\0';
  terrainLower[TERRAIN_WIDTH] = '\0';
  char temp = terrainUpper[16 - digits];
  terrainUpper[16 - digits] = '\0';
  lcd.setCursor(0, 0);
  lcd.print(terrainUpper);
  terrainUpper[16 - digits] = temp;
  lcd.setCursor(0, 1);
  lcd.print(terrainLower);

  lcd.setCursor(16 - digits, 0);
  lcd.print(score);

  terrainUpper[CAR_HORIZONTAL_POSITION] = upperSave;
  terrainLower[CAR_HORIZONTAL_POSITION] = lowerSave;
  return collide;
}

int tempo = 80;
const int buzzer = 11;

// Handle the button push as an interrupt
void buttonPush() {
  buttonPushed = true;

}


void setup() {
  pinMode(PIN_BUTTON, INPUT);
  digitalWrite(PIN_BUTTON, HIGH);
  pinMode(PIN_AUTOPLAY, OUTPUT);
  digitalWrite(PIN_AUTOPLAY, HIGH);

  // Digital pin 2 maps to interrupt 0
```

```
  attachInterrupt(0/*PIN_BUTTON*/, buttonPush, FALLING);

  initializeGraphics();

  lcd.init();
  lcd.backlight();

  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);

}

void loop() {

  static byte CARPos = CAR_POSITION_RUN_LOWER_1;
  static byte newTerrainType = TERRAIN_EMPTY;
  static byte newTerrainDuration = 1;
  static bool playing = false;
  static bool blink = false;
  static unsigned int distance = 0;


    int buttonState = digitalRead(PIN_BUTTON); // read new state

  if (buttonState == LOW) {
    digitalWrite(buzzer, HIGH); // turn on

  }
  else
  if (buttonState == HIGH) {
    digitalWrite(buzzer, LOW);  // turn off
  }

  if (!playing) {
    drawCAR((blink) ? CAR_POSITION_OFF : CARPos, terrainUpper, terrainLower, distance >>
3);
    if (blink) {
      lcd.setCursor(0, 0);
      lcd.print("Press Start");

    }
    delay(100);
    blink = !blink;
    if (buttonPushed) {
      initializeGraphics();
      CARPos = CAR_POSITION_RUN_LOWER_1;
      playing = true;
```

```
    buttonPushed = false;
    distance = 0;

   }
   return;
  }


  // Shift the terrain to the left
  advanceTerrain(terrainLower, newTerrainType == TERRAIN_LOWER_BLOCK ?
SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);
  advanceTerrain(terrainUpper, newTerrainType == TERRAIN_UPPER_BLOCK ?
SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);

  // Make new terrain to enter on the right
  if (--newTerrainDuration == 0) {
   if (newTerrainType == TERRAIN_EMPTY) {
    newTerrainType = (random(3) == 0) ? TERRAIN_UPPER_BLOCK :
TERRAIN_LOWER_BLOCK;
    newTerrainDuration = 10 + random(6);
   } else {
    newTerrainType = TERRAIN_EMPTY;
    newTerrainDuration = 10 + random(6);
   }
  }

  if (buttonPushed) {
   if (CARPos <= CAR_POSITION_RUN_LOWER_2) CARPos = CAR_POSITION_JUMP_1;
   buttonPushed = false;
  }

  if (drawCAR(CARPos, terrainUpper, terrainLower, distance >> 3)) {
   playing = false; // The CAR collided with something. Too bad.
   for (int i = 0; i <= 2; i++) {
   }
  } else {
   if (CARPos == CAR_POSITION_RUN_LOWER_2 || CARPos == CAR_POSITION_JUMP_8) {
    CARPos = CAR_POSITION_RUN_LOWER_1;
   } else if ((CARPos >= CAR_POSITION_JUMP_3 && CARPos <= CAR_POSITION_JUMP_5) &&
terrainLower[CAR_HORIZONTAL_POSITION] != SPRITE_TERRAIN_EMPTY) {
    CARPos = CAR_POSITION_RUN_UPPER_1;
   } else if (CARPos >= CAR_POSITION_RUN_UPPER_1 &&
terrainLower[CAR_HORIZONTAL_POSITION] == SPRITE_TERRAIN_EMPTY) {
    CARPos = CAR_POSITION_JUMP_5;
   } else if (CARPos == CAR_POSITION_RUN_UPPER_2) {
    CARPos = CAR_POSITION_RUN_UPPER_1;
   } else {
```

```
    ++CARPos;
   }
   ++distance;

   digitalWrite(PIN_AUTOPLAY, terrainLower[CAR_HORIZONTAL_POSITION + 2] ==
SPRITE_TERRAIN_EMPTY ? HIGH : LOW);
  }

}
```

Cod Arduino plăcuța B (muzica de fundal)

```
/*
  Fur Elise
*/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
```

```
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
```

```
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST     0


// change this to make the song slower or faster
int tempo = 80;

// change this to whichever pin you want to use

int buzzer = 7;


// notes of the moledy followed by the duration.
// a 4 means a quarter note, 8 an eighteenth , 16 sixteenth, so on
// !!negative numbers are used to represent dotted notes,
// so -4 means a dotted quarter note, that is, a quarter plus an eighteenth!!
const int melody[] PROGMEM = {

 // Fur Elise - Ludwig van Beethovem
 // Score available at https://musescore.com/user/28149610/scores/5281944

 //starts from 1 ending on 9
 NOTE_E5, 16, NOTE_DS5, 16, //1
 NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
 NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
 NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
 NOTE_C5, 8,  REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,

 NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,//6
 NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
 NOTE_B4, -8, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
 NOTE_A4 , 4, REST, 8, //9 - 1st ending

 //repaets from 1 ending on 10
 NOTE_E5, 16, NOTE_DS5, 16, //1
```

```
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
NOTE_C5, 8,  REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,

NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,//6
NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, -8, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
NOTE_A4, 8, REST, 16, NOTE_B4, 16, NOTE_C5, 16, NOTE_D5, 16, //10 - 2nd ending
//continues from 11
NOTE_E5, -8, NOTE_G4, 16, NOTE_F5, 16, NOTE_E5, 16,
NOTE_D5, -8, NOTE_F4, 16, NOTE_E5, 16, NOTE_D5, 16, //12

NOTE_C5, -8, NOTE_E4, 16, NOTE_D5, 16, NOTE_C5, 16, //13
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, REST, 16,
REST, 16, NOTE_E5, 16, NOTE_E6, 16, REST, 16, REST, 16, NOTE_DS5, 16,
NOTE_E5, 16, REST, 16, REST, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,

NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16, //19
NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
NOTE_A4, 8, REST, 16, NOTE_B4, 16, NOTE_C5, 16, NOTE_D5, 16, //24 (1st ending)

//repeats from 11
NOTE_E5, -8, NOTE_G4, 16, NOTE_F5, 16, NOTE_E5, 16,
NOTE_D5, -8, NOTE_F4, 16, NOTE_E5, 16, NOTE_D5, 16, //12

NOTE_C5, -8, NOTE_E4, 16, NOTE_D5, 16, NOTE_C5, 16, //13
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, REST, 16,
REST, 16, NOTE_E5, 16, NOTE_E6, 16, REST, 16, REST, 16, NOTE_DS5, 16,
NOTE_E5, 16, REST, 16, REST, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,

NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16, //19
NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
NOTE_A4, 8, REST, 16, NOTE_C5, 16, NOTE_C5, 16, NOTE_C5, 16, //25 - 2nd ending

//continues from 26
NOTE_C5 , 4, NOTE_F5, -16, NOTE_E5, 32, //26
```

NOTE_E5, 8, NOTE_D5, 8, NOTE_AS5, -16, NOTE_A5, 32,
NOTE_A5, 16, NOTE_G5, 16, NOTE_F5, 16, NOTE_E5, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_AS4, 8, NOTE_A4, 8, NOTE_A4, 32, NOTE_G4, 32, NOTE_A4, 32, NOTE_B4, 32,
NOTE_C5 , 4, NOTE_D5, 16, NOTE_DS5, 16,
NOTE_E5, -8, NOTE_E5, 16, NOTE_F5, 16, NOTE_A4, 16,
NOTE_C5 , 4, NOTE_D5, -16, NOTE_B4, 32,

NOTE_C5, 32, NOTE_G5, 32, NOTE_G4, 32, NOTE_G5, 32, NOTE_A4, 32, NOTE_G5, 32,
NOTE_B4, 32, NOTE_G5, 32, NOTE_C5, 32, NOTE_G5, 32, NOTE_D5, 32, NOTE_G5, 32, //33
NOTE_E5, 32, NOTE_G5, 32, NOTE_C6, 32, NOTE_B5, 32, NOTE_A5, 32, NOTE_G5, 32,
NOTE_F5, 32, NOTE_E5, 32, NOTE_D5, 32, NOTE_G5, 32, NOTE_F5, 32, NOTE_D5, 32,
NOTE_C5, 32, NOTE_G5, 32, NOTE_G4, 32, NOTE_G5, 32, NOTE_A4, 32, NOTE_G5, 32,
NOTE_B4, 32, NOTE_G5, 32, NOTE_C5, 32, NOTE_G5, 32, NOTE_D5, 32, NOTE_G5, 32,

NOTE_E5, 32, NOTE_G5, 32, NOTE_C6, 32, NOTE_B5, 32, NOTE_A5, 32, NOTE_G5, 32,
NOTE_F5, 32, NOTE_E5, 32, NOTE_D5, 32, NOTE_G5, 32, NOTE_F5, 32, NOTE_D5, 32, //36
NOTE_E5, 32, NOTE_F5, 32, NOTE_E5, 32, NOTE_DS5, 32, NOTE_E5, 32, NOTE_B4, 32,
NOTE_E5, 32, NOTE_DS5, 32, NOTE_E5, 32, NOTE_B4, 32, NOTE_E5, 32, NOTE_DS5, 32,
NOTE_E5, -8, NOTE_B4, 16, NOTE_E5, 16, NOTE_DS5, 16,
NOTE_E5, -8, NOTE_B4, 16, NOTE_E5, 16, REST, 16,

REST, 16, NOTE_DS5, 16, NOTE_E5, 16, REST, 16, REST, 16, NOTE_DS5, 16, //40
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,

NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16, //46
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
NOTE_A4, 8, REST, 16, NOTE_B4, 16, NOTE_C5, 16, NOTE_D5, 16,
NOTE_E5, -8, NOTE_G4, 16, NOTE_F5, 16, NOTE_E5, 16,
NOTE_D5, -8, NOTE_F4, 16, NOTE_E5, 16, NOTE_D5, 16,
NOTE_C5, -8, NOTE_E4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, REST, 16,
REST, 16, NOTE_E5, 16, NOTE_E6, 16, REST, 16, REST, 16, NOTE_DS5, 16,

NOTE_E5, 16, REST, 16, REST, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_D5, 16, //54
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, NOTE_DS5, 16,
NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,

NOTE_A4, 8, REST, 16, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16, //60
NOTE_B4, 8, REST, 16, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,

```
  NOTE_A4, 8, REST, 16, REST, 16, REST, 8,
  NOTE_CS5 , -4,
  NOTE_D5 , 4, NOTE_E5, 16, NOTE_F5, 16,
  NOTE_F5 , 4, NOTE_F5, 8,
  NOTE_E5 , -4,
  NOTE_D5 , 4, NOTE_C5, 16, NOTE_B4, 16,
  NOTE_A4 , 4, NOTE_A4, 8,
  NOTE_A4, 8, NOTE_C5, 8, NOTE_B4, 8,
  NOTE_A4 , -4,
  NOTE_CS5 , -4,

  NOTE_D5 , 4, NOTE_E5, 16, NOTE_F5, 16, //72
  NOTE_F5 , 4, NOTE_F5, 8,
  NOTE_F5 , -4,
  NOTE_DS5 , 4, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_AS4 , 4, NOTE_A4, 8,
  NOTE_GS4 , 4, NOTE_G4, 8,
  NOTE_A4 , -4,
  NOTE_B4 , 4, REST, 8,
  NOTE_A3, -32, NOTE_C4, -32, NOTE_E4, -32, NOTE_A4, -32, NOTE_C5, -32, NOTE_E5, -32,
NOTE_D5, -32, NOTE_C5, -32, NOTE_B4, -32,

  NOTE_A4, -32, NOTE_C5, -32, NOTE_E5, -32, NOTE_A5, -32, NOTE_C6, -32, NOTE_E6, -32,
NOTE_D6, -32, NOTE_C6, -32, NOTE_B5, -32, //80
  NOTE_A4, -32, NOTE_C5, -32, NOTE_E5, -32, NOTE_A5, -32, NOTE_C6, -32, NOTE_E6, -32,
NOTE_D6, -32, NOTE_C6, -32, NOTE_B5, -32,
  NOTE_AS5, -32, NOTE_A5, -32, NOTE_GS5, -32, NOTE_G5, -32, NOTE_FS5, -32, NOTE_F5, -
32, NOTE_E5, -32, NOTE_DS5, -32, NOTE_D5, -32,

  NOTE_CS5, -32, NOTE_C5, -32, NOTE_B4, -32, NOTE_AS4, -32, NOTE_A4, -32, NOTE_GS4, -
32, NOTE_G4, -32, NOTE_FS4, -32, NOTE_F4, -32, //84
  NOTE_E4, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,

  NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, NOTE_DS5, 16, //88
  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
  NOTE_A4, -8, REST, -8,
  REST, -8, NOTE_G4, 16, NOTE_F5, 16, NOTE_E5, 16,
  NOTE_D5 , 4, REST, 8,
  REST, -8, NOTE_E4, 16, NOTE_D5, 16, NOTE_C5, 16,

  NOTE_B4, -8, NOTE_E4, 16, NOTE_E5, 8, //96
  NOTE_E5, 8, NOTE_E6, -8, NOTE_DS5, 16,
  NOTE_E5, 16, REST, 16, REST, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_DS5, 16,
```

```
  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,

  NOTE_C5, 8, REST, 16, NOTE_E4, 16, NOTE_E5, 16, NOTE_DS5, 16, //102
  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
  NOTE_A4 , -4,
};

// sizeof gives the number of bytes, each int value is composed of two bytes (16 bits)
// there are two values per note (pitch and duration), so for each note there are four bytes
int notes = sizeof(melody) / sizeof(melody[0]) / 2;

// this calculates the duration of a whole note in ms
int wholenote = (60000 * 4) / tempo;

int divider = 0, noteDuration = 0;
void setup() {

  // iterate over the notes of the melody.
  // Remember, the array is twice the number of notes (notes + durations)

}

void loop() {
  // no need to repeat the melody.
   for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
    // calculates the duration of each note
    divider = pgm_read_word_near(melody+thisNote + 1);
    if (divider > 0) {
      // regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted notes
    }
    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, pgm_read_word_near(melody+thisNote), noteDuration * 0.9);
    // Wait for the specief duration before playing the next note.
    delay(noteDuration);
    // stop the waveform generation before the next note.
    noTone(buzzer);
  }
}
```

Componenta Hardware

Lista componente:

- 2 Arduino UNO;
- 11 fire mamă-tată;
- 1 LCD I2C 16x2;
- 1 Push Button;
- 1 Buzzer;
- 1 Breadboard;
- 9 fire jumper.

Caracteristicile componentelor hardware:

Arduino UNO

Tip	Placă de dezvoltare cu microcontroler
Data lansării	24 septembrie 2010
CPU	Atmega328P @ 16MHz
Memorie	32KB Flash din care 0.5KB este ocupat de bootloader, 2KB SRAM, 1KB EEPROM

Arduino UNO constituie o platforma de procesare tip open-source, bazata pe un software si hardware flexibil construita in jurul unui microcontroler ATMEGA 328P-PU capabila de a prelua date printr-o serie de senzori conectati la pinii placii si de a actiuna asupra altor dispozitive ca LED-uri, motoarelor, servomotoare, sau alte tipuri de dispozitive mecanice pe baza unor comenzi cuprinse in codul scris intr-un limbaj de programare, similar cu limbajul C++ incarcat in memoria microcontrolerului.Placa se constituie ca o platforma de referinta pentru cei de la Arduino si se poate achizitiona la preturi intre 80 si 110 RON dar puteti achizitiona o clona chinezeasca perfect functionala la preturi in jur de 65 RON. Aspectul placii este aratat in imaginea de mai sus.

Alimentarea

Placa Arduino Uno poate fi alimentata ce laportul USB al calculatoriului sau de la o sursa externa.Selectia surselor se face automat.Sursa externa poate fi un adaptor AC/DC sau baterii. Adaptorul este un jack de 2.1 mm, avand plusul pe centru. Firele de la baterie pot fi conectate fie prin intermediul aceluiasi port sau pot fi conectate la pinii header GND respectiv Vin ai conectorului POWER.Placa poate sa functioneze cu tensiuni intre 6 si 20 volti dar valorile de tensiune recomandate sunt in gama 7 – 12 volti.Pinii de alimentare sunt urmatorii:

VIN. Intrare pentru alimentare cu tensiune externa pentru situatia cand nu se foloseste conectarea la portul de USB al calculatorului care ofera si el o tensiune de 5 volti.Se poate alimenta prin acest pin (cu 7-12V) sau se poate accesa tensiunea de intrare prin acest pin. **5V**. Acest pin furnizeaza o tensiune stabilizata de 5V obtinuta din stabilizatorul intern al placii.Alimentarea cu tensiune exterioara prin pinii 5V sau 3.3V poate distruge placa. **3V3**. Acest pin furnizeaza o tensiune de 3.3 la un curent maxim de 50 mA generata de un stabilizator intern. Tensiunea poate fi utilizata pentru aplicatii care necesita alimentarea la3,3 volti.**GND**. Pini de masa**IOREF**. Acest pin genereaza o tensiune de eferinta cu care microcontrolerul poate opera.

Memoria

Microcontrolerul ATmega328 are 32 KB de memorie din care 0.5 KB sunt utilizati pentru bootloader.Contine de asemena 2 KB de SRAM si 1 KB de memorie EEPROM

Input and Output

Fiecare din cei 14 pini digitali al lui Arduino Uno pot fi utilizati ca input sau output, utilizand functiunile pinMode(), digitalWrite() si digitalRead().Pinii functioneaza la 5 volti si pot furniza sau absorbi un curent de maximum 40 mA si datorita unui pull-up resistor, care sunt deconectate by default avand valoarea de 20-50 khms.O pare din pini au functii speciale: **Serial**: 0 (RX) si 1 (TX). Sunt utilizati pentru receptia (RX) si transmisia (TX) datelor seriale TTL. Acesti pini sunt conectati la pinii corespunzatori ai ATmega8U2, care are rolul de convertor USB/TTL Serial chip.**External Interrupts**: 2 si 3. Acesti pini pot fi configurati ca pini pentru intreruperi externe.**PWM**: 3, 5, 6, 9, 10, si 11. Sunt iesiri cu functii PWM pe 8-bit prin functia analogWrite().**SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Sunt pini care asigura comunicatiaSPI prin utilizarea SPI library.**LED**: 13. Este singurul LED conectat pe portul digital 13. Atunci cand acest pin este HIGH atunci LED-ul este ON iar cand valoarea lui este LOW atunci acest LED este OFF.

Placa Arduino Uno are **6 intrari analogice**, numerotate de la A0 la A5, fiecare permit o rezolutie pe 10 bits, asta insemnand un numar de 1024 valori diferite. In mod normal permit masuratori a unor valori de tensiune in gama 0- 5 volti, dar este posibila modificarea prin folosirea unei tensiuni diferite la pinul AREF (analogReference).**TWI**: A4 sau SDA pin si A5 sau SCL pin, permit comunicatie TWI, cu sprijinul librariilor Wire library.**AREF**. Furnizeaza o tensiune de referinta pentru intrarile analogice. prin utilizarea functiei analogReference(). Reset. Punerea acestui pin in LOW are loc un RESET al microcontrolerului. In mod normal este utilizat la conectarea unui buton de RESET

Comunicarea

Arduino Uno permite comunicarea cu PC-ul, cu un alta placa Arduino sau cu alte microcontrolere. ATmega328 asigura comunicatii seriale UART TTL (5V), pentru care sunt prevazuti pinii digitali **D0 (RX) si D1 (TX)**. Placa este prevazuta si cu un ATmega16U2 care asigura o comunicatie seriala over USB care apare ca si port virtual pentru software-ul calculatorului. Firmware-ul din ATmega16U2 utilizeaza drivere USB standard si nu are nevoie de drivere externe ci doar de un fisier .inf.Arduino software are inclus si un serial monitor care permite vizualizarea datelor text transferate. Doua LED-uri, unul montat pe RX si altul

pe TX vor clipi atunci cand datele circula via USB la calculator nu insa si pentru indicarea comunicatiei seriale de pe pinii D0 si D1. ATmega328 suporta si comunicatii I2C (TWI) sau SPI. Software-ul de la Arduino include si o librarie Wire library ce simplifica comunicatia pe bus-ul I2C. Comunicatia SPI utilizeaza la randul ei o librarie SPI.

Programarea

Microcontrolerul de pe placa Arduino poate fi programat prin mediul Arduino, daca se selecteaza Arduino Uno din menu-ul Tools, alegand corect tipul microcontrolerului de pe placa. De remarcat este ca microcontrolerul ATmega328 de pe Arduino Uno, vine incarcat cu un bootloader care ofera posibilitatea ca sa se poata incarca noul cod, fara utilizarea unui dispozitiv hardware suplimentar. Comunicatia are loc folosind protocolul STK500. Daca se doreste programarea fara folosirea bootloader-ului, se poat folosi pinii ICSP (In-Circuit Serial Programming). Aceasta metoda ofera avantajul de a castiga 0,5 kB din memorie, care in mod normal este ocupata de bootloader, insa ere dezavantajul de a avea nevoie de un programator extern.

LCD I2C

- Material PCB + plastic
- Tip ecran LCD
- Dimensiune ecran 2.6 inch
- Rezoluție 80 x 16
- Tensiune de lucru 4.5 ~ 5.5V
- Curent de lucru 80mA
- Dimensiuni 3,15 in x 1,42 in x 0,71 in (8 cm x 3,6 cm x 1,8 cm)
- Greutate 34 g

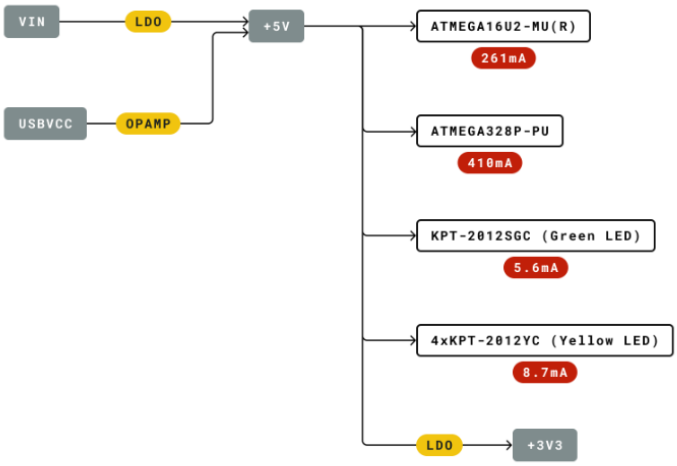
Acest ecran LCD 1602 cu modul IIC/I2C integrat, este compatibil cu Arduino Uno R3 / Arduino Mega 2560 și poate fi folosit pentru afișarea informațiilor, primite de la diferiți senzori de temperatura, umiditate sau orice fel de mesaje programate.

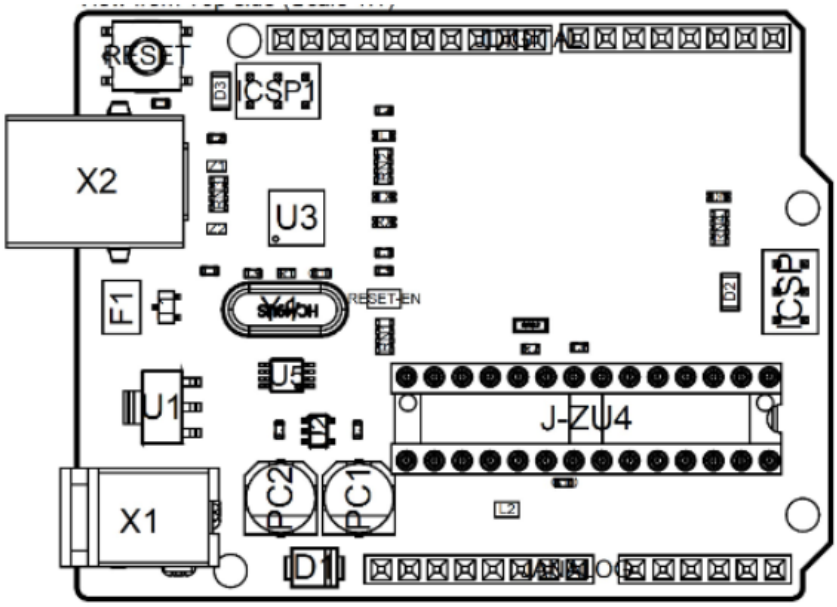
Acest LCD este folosit deobicei pentru proiecte în care nu avem foarte mulți pini disponibili de la microcontroller, datorită adaptorului pentru interfață I2C, ce are nevoi de de doar două conexiuni, SDA/SCL și conexiunea la masă. LCD-ul are contrast ajustabil și vă poate ajuta să citiți clar ecranul într-un mediu întunecat. Culoarea caracterelor este alba, iar backlightul este albastru.

Buzzer

Modulul constă într-un buzzer piezoelectric pasiv, care poate genera tonuri între 1,5 și 2,5 kHz prin comutarea și oprirea la frecvențe diferite, fie folosind întârzieri sau PWM.Acesta poate produce o gamă variată de sunete în funcție de frecvența de intrare.

Performanțe





Board topology

Ref.	Description	Ref.	Description
X1	Power jack 2.1x5.5mm	U1	SPX1117M3-L-5 Regulator
X2	USB B Connector	U3	ATMEGA16U2 Module
PC1	EEE-1EA470WP 25V SMD Capacitor	U5	LMV358LIST-A.9 IC
PC2	EEE-1EA470WP 25V SMD Capacitor	F1	Chip Capacitor, High Density
D1	CGRA4007-G Rectifier	ICSP	Pin header connector (through hole 6)
J-ZU4	ATMEGA328P Module	ICSP1	Pin header connector (through hole 6)
Y1	ECS-160-20-4X-DU Oscillator		

Concluzii

Costurile au fost foarte reduse pentru realizarea acestui proiect deoarece unele componente le-am achiziționat anterior pentru realizarea altor proiecte și le-am reutilizat , iar celelalte componente le-am împrumutat de la colegii din cămin.

Dificultatea principală întâmpinată a fost implementarea muzicii de fundal deoarece nu am reușit să programez microcontroller-ul să ruleze codul pentru joc și codul pentru coloana sonoră în același timp . Așa că am decis să integrez un al doilea modul Arduino UNO pe care l-am conectat la primul si am implementat pe acesta codul pentru muzică.

Bibliografie

1. Arduino datasheet
https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

2. LCD I2C 16x2 datasheet
http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf

3. Buzzer datasheet
https://www.farnell.com/datasheets/2171929.pdf

4. Surse de inspirație proiect
https://www.hackster.io/bruno_opaiva/car-game-with-arduino-and-i2c-lcd-display-938b6e
https://github.com/robsoncouto/arduino-songs/blob/master/furelise/furelise.ino

Anexa

song

```
// notes of the moledy followed by the duration.
// a 4 means a quarter note, 8 an eighteenth , 16 sixteenth, so on
// !!negative numbers are used to represent dotted notes,
// so -4 means a dotted quarter note, that is, a quarter plus an eighteenth!!
const int melody[] PROGMEM = {

  // Fur Elise - Ludwig van Beethovem
  // Score available at https://musescore.com/user/28149610/scores/5281944

  //starts from 1 ending on 9
  NOTE_E5, 16, NOTE_DS5, 16, //1
  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
  NOTE_C5, 8,  REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,

  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,//6
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_C5, 16, NOTE_B4, 16,
  NOTE_A4 , 4, REST, 8, //9 - 1st ending

  //repaets from 1 ending on 10
  NOTE_E5, 16, NOTE_DS5, 16, //1
  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
  NOTE_B4, -8, NOTE_E4, 16, NOTE_GS4, 16, NOTE_B4, 16,
  NOTE_C5, 8,  REST, 16, NOTE_E4, 16, NOTE_E5, 16,  NOTE_DS5, 16,

  NOTE_E5, 16, NOTE_DS5, 16, NOTE_E5, 16, NOTE_B4, 16, NOTE_D5, 16, NOTE_C5, 16,//6
  NOTE_A4, -8, NOTE_C4, 16, NOTE_E4, 16, NOTE_A4, 16,
```

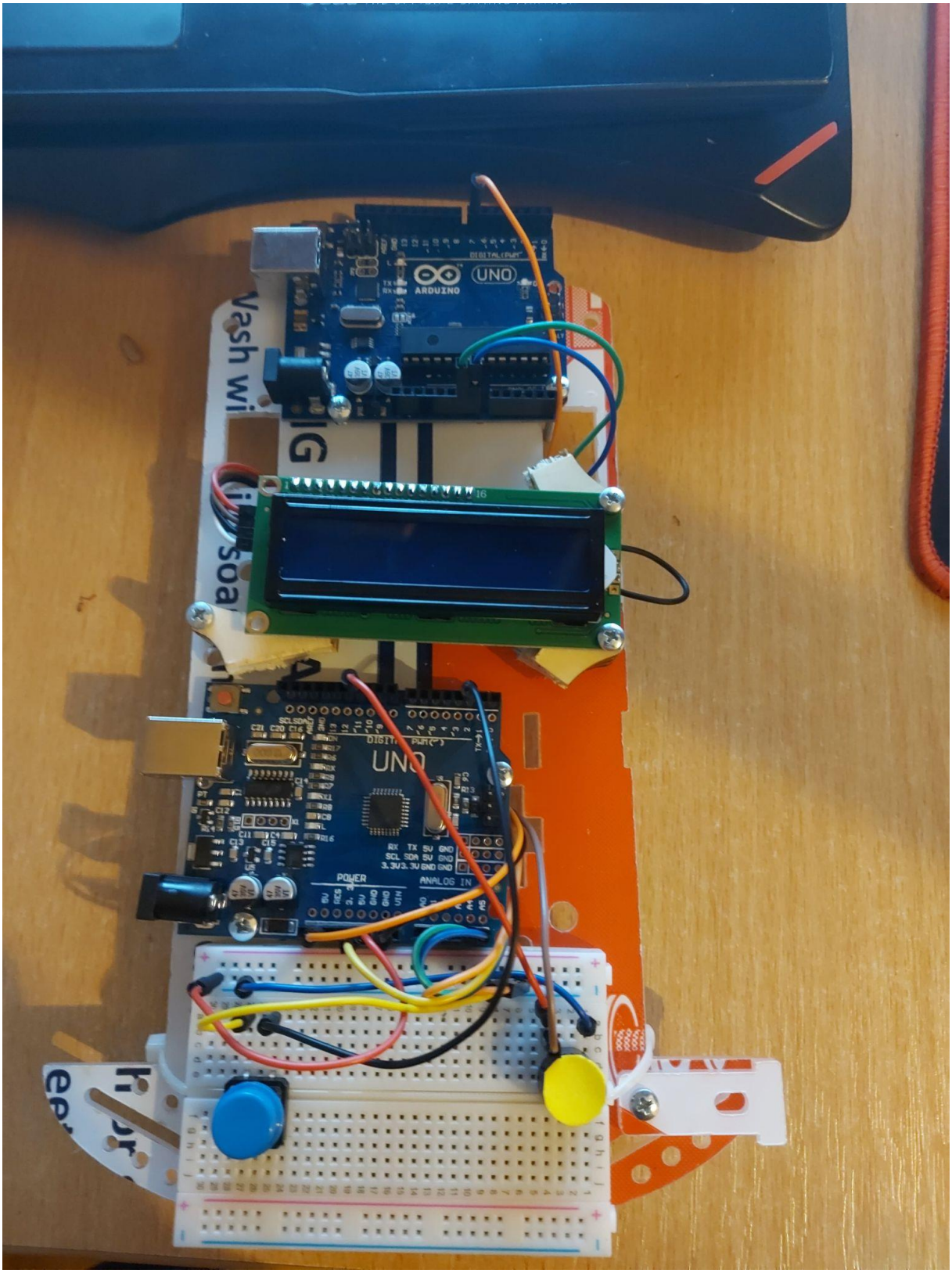
Done uploading.

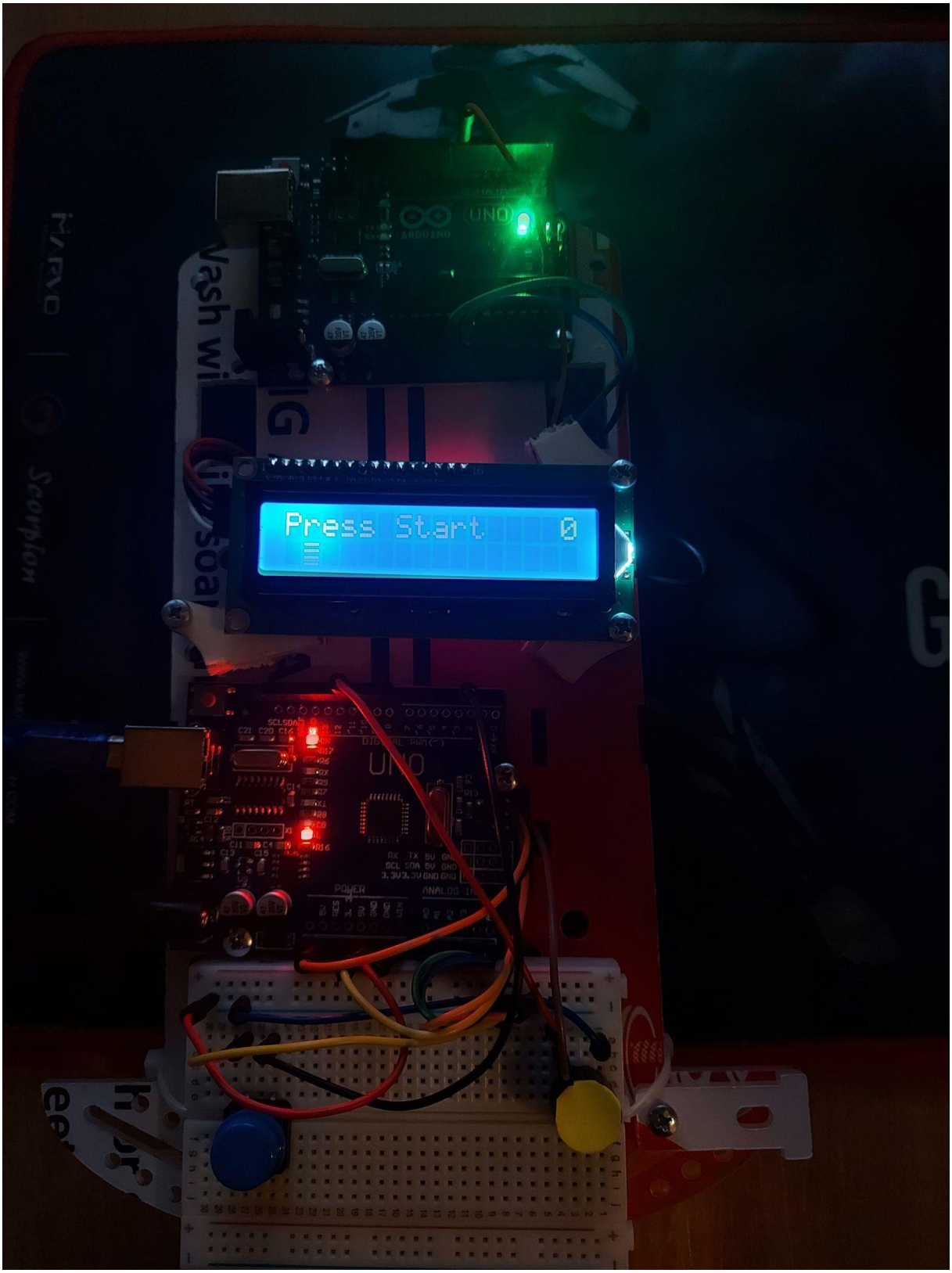
```
Sketch uses 5714 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 32 bytes (1%) of dynamic memory, leaving 2016 bytes for local variables. Maximum is 2048 bytes.
```

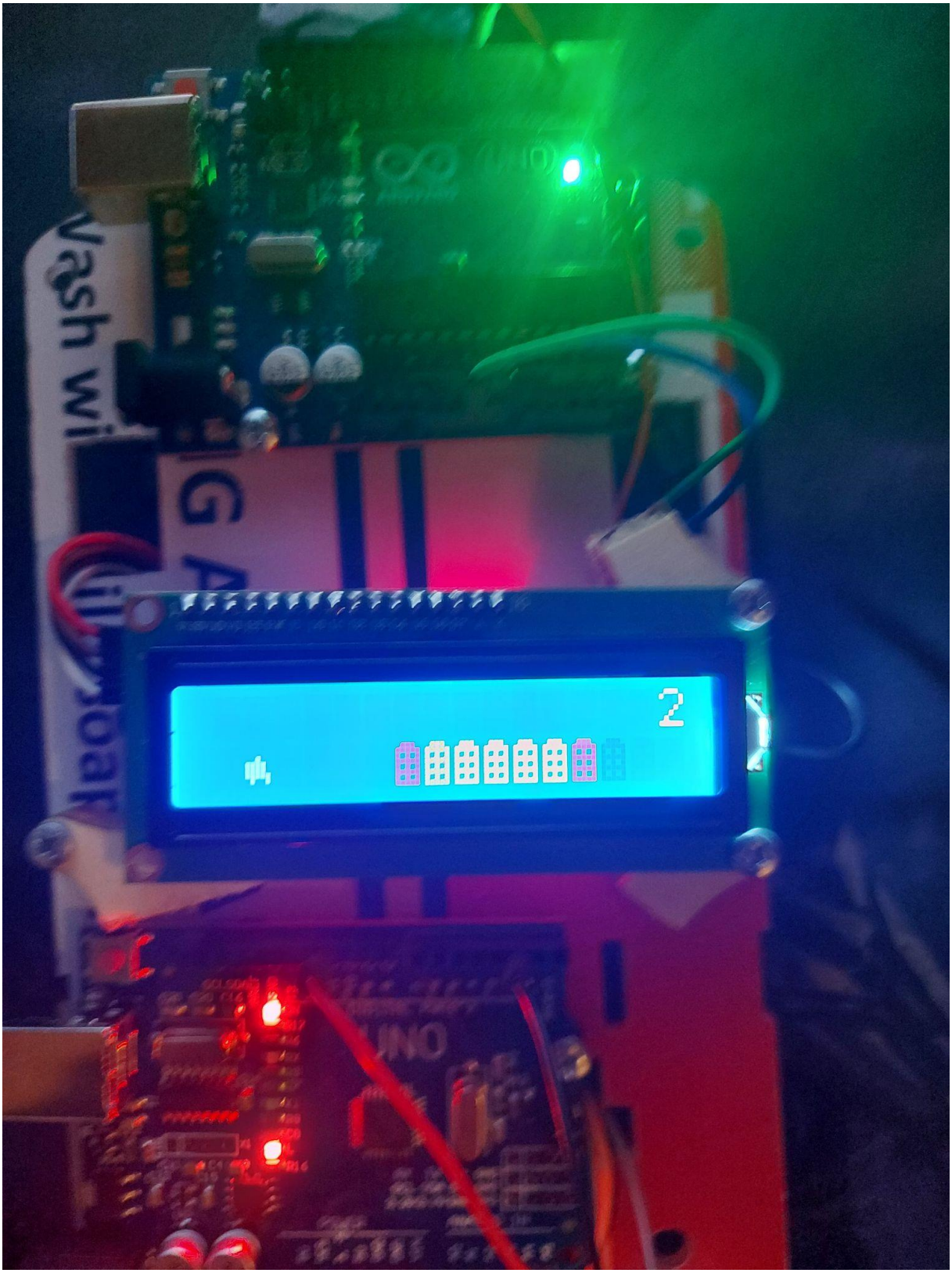
game_boy2.0

```
//
void advanceTerrain(char* terrain, byte newTerrain) {
  for (int i = 0; i < TERRAIN_WIDTH; ++i) {
    char current = terrain[i];
    char next = (i == TERRAIN_WIDTH - 1) ? newTerrain : terrain[i + 1];
    switch (current) {
      case SPRITE_TERRAIN_EMPTY:
        terrain[i] = (next == SPRITE_TERRAIN_SOLID) ? SPRITE_TERRAIN_SOLID_RIGHT : SPRITE_TERRAIN_EMPTY;
        break;
      case SPRITE_TERRAIN_SOLID:
        terrain[i] = (next == SPRITE_TERRAIN_EMPTY) ? SPRITE_TERRAIN_SOLID_LEFT : SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_RIGHT:
        terrain[i] = SPRITE_TERRAIN_SOLID;
        break;
      case SPRITE_TERRAIN_SOLID_LEFT:
        terrain[i] = SPRITE_TERRAIN_EMPTY;
        break;
    }
  }
}

bool drawCAR(byte position, char* terrainUpper, char* terrainLower, unsigned int score) {
  bool collide = false;
  char upperSave = terrainUpper[CAR_HORIZONTAL_POSITION];
  char lowerSave = terrainLower[CAR_HORIZONTAL_POSITION];
  byte upper, lower;
  switch (position) {
    case CAR_POSITION_OFF:
      upper = lower = SPRITE_TERRAIN_EMPTY;
      break;
    case CAR_POSITION_RUN_LOWER_1:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN1;
      break;
```


ARDUINO
UNO
UNO
POWER
ANALOG IN


Press Start 0


2