jumper game

Prezentarea pe scurt a proiectului vostru:

- a game where you have to jump and make a high score

Descriere generală

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

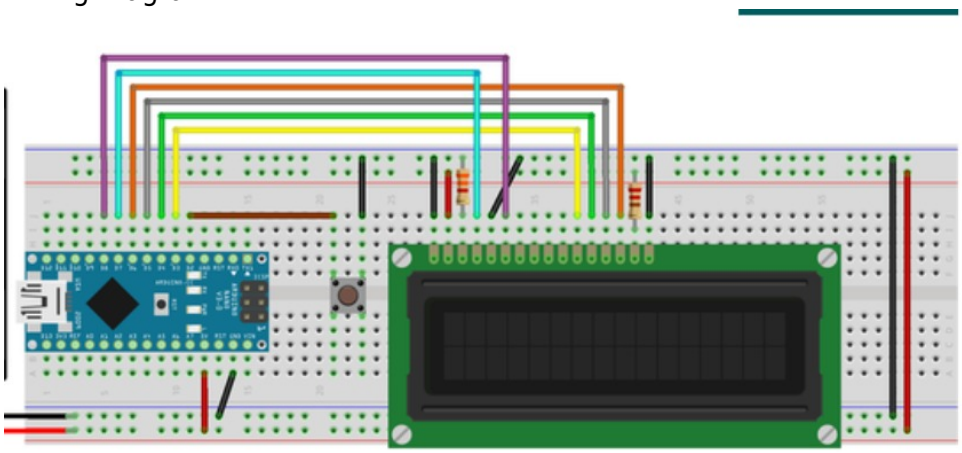
Hardware Design

Aici puneţi tot ce ţine de hardware design:

- LCD corespund cu astia de pe arduino:

rs = 12, en = 11, d4 = 6, d5 = 5, d6 = 4, d7 = 3

Wiring Diagram :



Software Design

```
#include <LiquidCrystal.h>

#define PIN_BUTTON 2

#define SPRITE_RUN1 1 #define SPRITE_RUN2 2 #define SPRITE_JUMP 3 #define SPRITE_JUMP_UPPER '.'  Use the '.' character for the head #define SPRITE_JUMP_LOWER 4 #define SPRITE_TERRAIN_EMPTY
```

' ' User the ' ' character #define SPRITE_TERRAIN_SOLID 5 #define SPRITE_TERRAIN_SOLID_RIGHT 6 #define SPRITE_TERRAIN_SOLID_LEFT 7

#define HERO_HORIZONTAL_POSITION 1 *Horizontal position of hero on screen #define TERRAIN_WIDTH 16 #define TERRAIN_EMPTY 0 #define TERRAIN_LOWER_BLOCK 1 #define TERRAIN_UPPER_BLOCK 2 #define HERO_POSITION_OFF 0* Hero is invisible #define HERO_POSITION_RUN_LOWER_1 1 *Hero is running on lower row (pose 1) #define HERO_POSITION_RUN_LOWER_2 2* (pose 2)

#define HERO_POSITION_JUMP_1 3 *Starting a jump #define HERO_POSITION_JUMP_2 4* Half-way up #define HERO_POSITION_JUMP_3 5 *Jump is on upper row #define HERO_POSITION_JUMP_4 6* Jump is on upper row #define HERO_POSITION_JUMP_5 7 *Jump is on upper row #define HERO_POSITION_JUMP_6 8* Jump is on upper row #define HERO_POSITION_JUMP_7 9 *Half-way down #define HERO_POSITION_JUMP_8 10* About to land

#define HERO_POSITION_RUN_UPPER_1 11 *Hero is running on upper row (pose 1) #define HERO_POSITION_RUN_UPPER_2 12* (pose 2)

DB7 const int rs = 12, en = 11, d4 = 6, d5 = 5, d6 = 4, d7 = 3; LiquidCrystal lcd(rs, en, d4, d5, d6, d7); char terrainUpper[TERRAIN_WIDTH + 1]; char terrainLower[TERRAIN_WIDTH + 1]; volatile boolean buttonPushed = false; void initializeGraphics() { byte graphics[] = { Run position 1

```
      B01100,
      B01100,
      B00000,
      B01110,
      B11100,
      B01100,
      B11010,
      B10011,
      // Run position 2
      B01100,
      B01100,
      B00000,
      B01100,
      B01100,
      B01100,
      B01100,
      B01110,
      // Jump
      B01100,
      B01100,
      B00000,
      B11110,
      B01101,
      B11111,
      B10000,
      B00000,
      // Jump lower
      B11110,
      B01101,
```

```
      B11111,
      B10000,
      B00000,
      B00000,
      B00000,
      B00000,
      // Ground
      B11111,
      B11111,
      B11111,
      B11111,
      B11111,
      B11111,
      B11111,
      B11111,
      B11111,
      // Ground right
      B00011,
      B00011,
      B00011,
      B00011,
      B00011,
      B00011,
      B00011,
      B00011,
      B00011,
      // Ground left
      B11000,
      B11000,
      B11000,
      B11000,
      B11000,
      B11000,
      B11000,
      B11000,
      B11000,
  };
```

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; } } } boolean drawHero(byte position, char* terrainUpper, char* terrainLower, unsigned int score) { boolean collide = false; char upperSave = terrainUpper[HERO_HORIZONTAL_POSITION]; char lowerSave = terrainLower[HERO_HORIZONTAL_POSITION]; byte upper, lower; switch (position) { case HERO_POSITION_OFF: upper = lower = SPRITE_TERRAIN_EMPTY; break; case HERO_POSITION_RUN_LOWER_1: upper = SPRITE_TERRAIN_EMPTY; lower = SPRITE_RUN1; break; case HERO_POSITION_RUN_LOWER_2: upper = SPRITE_TERRAIN_EMPTY; lower = SPRITE_RUN2; break; case HERO_POSITION_JUMP_1: case HERO_POSITION_JUMP_8: upper = SPRITE_TERRAIN_EMPTY; lower = SPRITE_JUMP; break; case HERO_POSITION_JUMP_2: case HERO_POSITION_JUMP_7: upper = SPRITE_JUMP_UPPER; lower = SPRITE_JUMP_LOWER; break; case HERO_POSITION_JUMP_3: case HERO_POSITION_JUMP_4: case HERO_POSITION_JUMP_5: case HERO_POSITION_JUMP_6: upper = SPRITE_JUMP; lower = SPRITE_TERRAIN_EMPTY; break; case HERO_POSITION_RUN_UPPER_1: upper = SPRITE_RUN1; lower = SPRITE_TERRAIN_EMPTY; break; case HERO_POSITION_RUN_UPPER_2: upper = SPRITE_RUN2; lower = SPRITE_TERRAIN_EMPTY; break; } if (upper != ' ') { terrainUpper[HERO_HORIZONTAL_POSITION] = upper; collide = (upperSave == SPRITE_TERRAIN_EMPTY) ? false : true; } if (lower != ' ') { terrainLower[HERO_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[HERO_HORIZONTAL_POSITION] = upperSave;

```
  terrainLower[HERO_HORIZONTAL_POSITION] = lowerSave;
```

return collide; }

Handle the button push as an interrupt void buttonPush() { buttonPushed = true; } void setup() { pinMode(PIN_BUTTON, INPUT_PULLUP); Digital pin 2 maps to interrupt 0

```
  attachInterrupt(0, buttonPush, FALLING);
```

initializeGraphics();

lcd.begin(16, 2); }

void loop(){

```
  static byte heroPos = HERO_POSITION_RUN_LOWER_1;
  static byte newTerrainType = TERRAIN_EMPTY;
  static byte newTerrainDuration = 1;
  static boolean playing = false;
  static boolean blink = false;
  static unsigned int distance = 0;
```

if (!playing) {

```
      drawHero((blink) ? HERO_POSITION_OFF : heroPos, terrainUpper,
terrainLower, distance >> 3);
```

if (blink) {

```
      lcd.setCursor(0,0);
      lcd.print("Press Start");
      }
```

delay(250);

```
      blink = !blink;
```

if (buttonPushed) {

```
          initializeGraphics();
          heroPos = HERO_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 = 2 + random(10);
      } else {
          newTerrainType = TERRAIN_EMPTY;
          newTerrainDuration = 10 + random(10);
      }
  }
```

if (buttonPushed) {

```
      if (heroPos <= HERO_POSITION_RUN_LOWER_2)
          heroPos = HERO_POSITION_JUMP_1;
```

buttonPushed = false;

```
  }
```

if (drawHero(heroPos, terrainUpper, terrainLower, distance » 3)) {

```
      playing = false; // The hero collided with something. Too bad.
  } else {
      if (heroPos == HERO_POSITION_RUN_LOWER_2 || heroPos ==
HERO_POSITION_JUMP_8) {
          heroPos = HERO_POSITION_RUN_LOWER_1;
      } else if ((heroPos >= HERO_POSITION_JUMP_3 && heroPos <=
HERO_POSITION_JUMP_5) && terrainLower[HERO_HORIZONTAL_POSITION] !=
SPRITE_TERRAIN_EMPTY) {
          heroPos = HERO_POSITION_RUN_UPPER_1;
      } else if (heroPos >= HERO_POSITION_RUN_UPPER_1 &&
terrainLower[HERO_HORIZONTAL_POSITION] == SPRITE_TERRAIN_EMPTY) {
          heroPos = HERO_POSITION_JUMP_5;
      } else if (heroPos == HERO_POSITION_RUN_UPPER_2) {
          heroPos = HERO_POSITION_RUN_UPPER_1;
      } else {
          ++heroPos;
      }
```

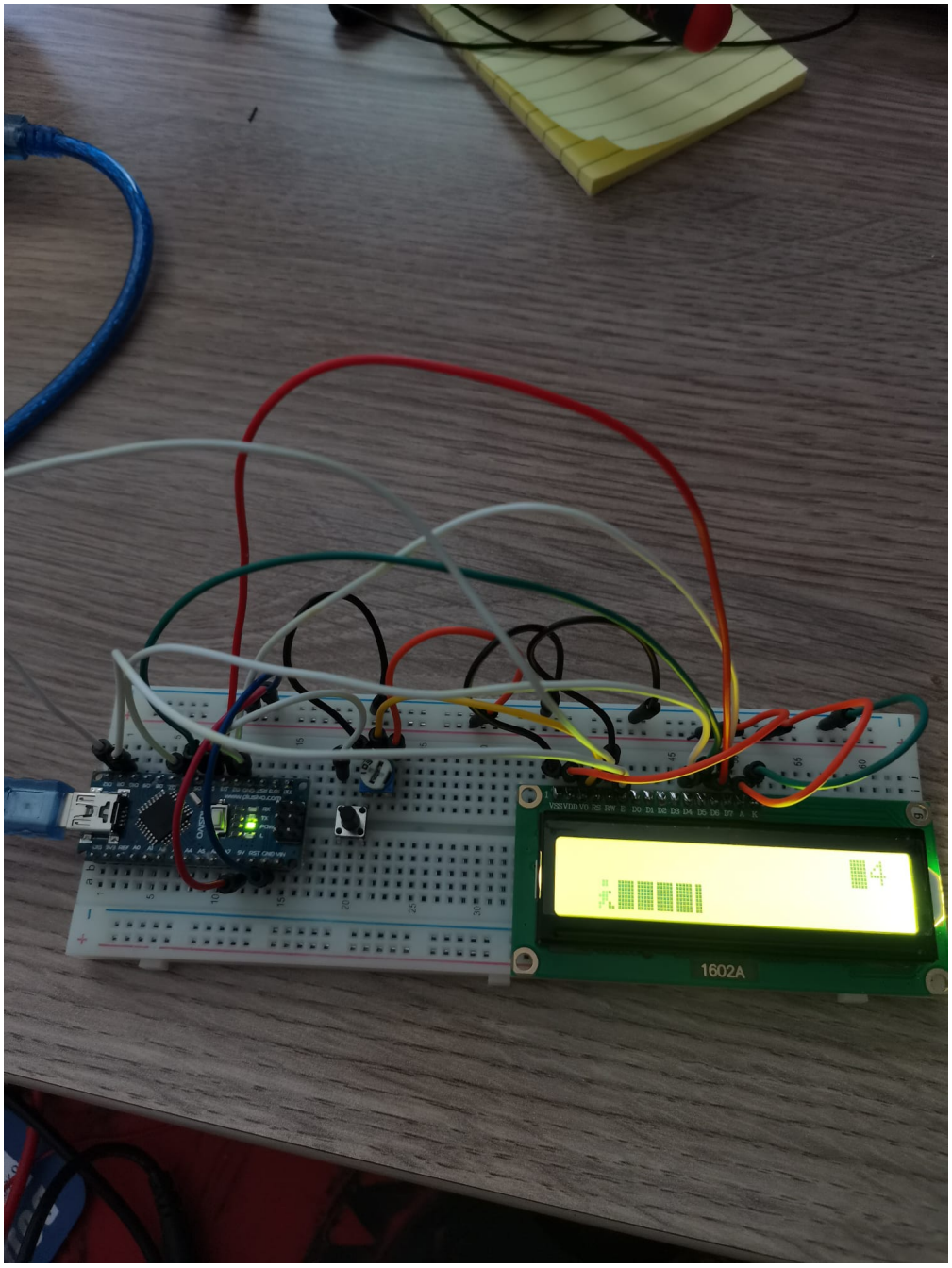
++distance;

```
  }
```

delay(100); }

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.



Concluzii
Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună 😉.

Fişierele se încarcă pe wiki folosind facilitatea *Add Images or other files. Namespace-ul în care se încarcă fişierele este de tipul *:pm:prj20??:c?* sau *:pm:prj20??:c?:nume_student** (dacă este cazul). *Exemplu:* Dumitru Alin, 331CC → *:pm:prj2009:cc:dumitru_alin*.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe *Resurse Software* şi *Resurse Hardware*.

Export to PDF