PROJECT REPORT

Nijat Sadikhov

August 17, 2021

Micro-controllers Programming

Handheld Game Console

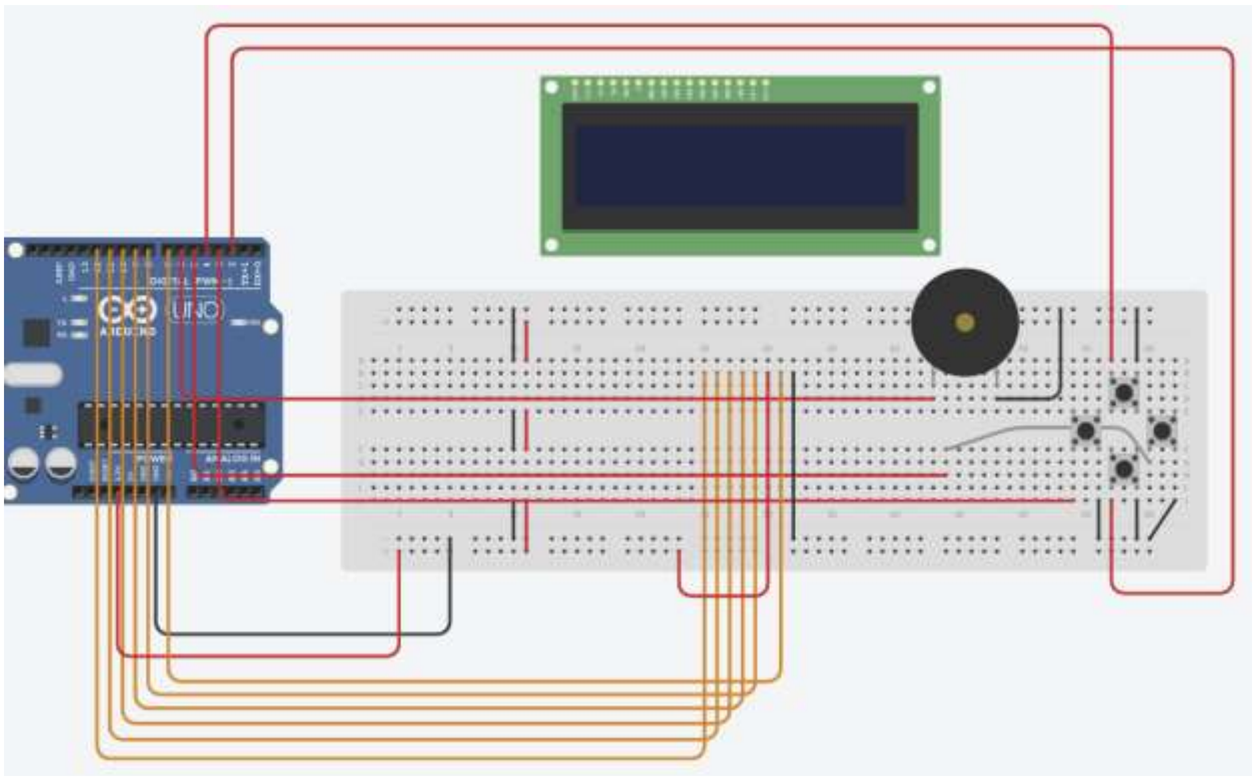
Description

This project is about a small, portable game console with a screen, speaker and 4 buttons. The console has built-in configuration options, along with game options. Retro style games can be written and added to the console.

Components used

- Arduino Nano x1.
- Push Button x4
- Piezo Buzzer x1
- Breadboard x1
- Nokia 5110 LCD display x1
- Jumper Wires

Hardware / Schema



Details

As it is mentioned in the description, more games can be coded and imported to the console. The console currently has a built-in game named “Snake”.

When the console gets turned on, first, an animation with a song shows up, and then comes the “Menu” page. In this page, there are 2 options (Games and Settings). From one option to another, up and down buttons are used to navigate.

Games option has 2 games inside: 1-Snake and 2-Ping Pong. The game “Ping Pong” is still in development process.

In Settings, there are 3 configuration selections.

1. Contrast level (default value is 70)
2. Background light (default: OFF)
3. Reset

Any changes done in settings are being saved and applied to console instantly. Reset option can be used to set the settings options back to their default values.

Snake:

The Game consists of a score header, rectangular shape frame, snake itself and food.

- Score Header: it starts by default from 1 and increments by 1, every time the snake eats food.
- Snake: in the beginning of the game, its size is 1 which is its head. For every eaten food, its size and speed increases. A player loses the game when the snake hits its tail or the frame. An indication shows up and the page redirects to the game options page.
- Food: its location is randomly generated inside of a frame. When it is eaten by the snake, on other coordinates, it appears.

With the usage of buzzer, some melodies are also implemented in this game. it occurs when the food is eaten, or the game is lost.

Code (Software)

The software is written in C++ programming language. To make use of the Nokia LCD display, a library called “LCD5110” is used.

Main Functions:

drawInitialPage () – it is used to demonstrate an initial layout.

printPageOptions () – it prints out the options between pages.

checkButtonState(button) – this function is responsible for checking states of buttons, whether there are pressed or not.

startGame () – it clears the screen and shows a small animation before starting the game.

initSnakeBoard () – it is used to initialize the snake and its surroundings.

moveSnake () – with the usage of this function, snake moves on the screen according to the pressed button.

Code:

```
void initSnakeBoard() {
  snake = Snake_OBJ((LCD_WIDTH) / 2, LCD_HEIGHT - 10, 3, 250);
  snakeBoard = SnakeBoard(snake.sizePerLen, 10, LCD_WIDTH - 1 - 2, LCD_HEIGHT - 1, 2);

  lcd.print("SCORE: " + String(snakeBoard.score), CENTER, 0);

  // drawing borders
  lcd.drawRect(snakeBoard.borderLeftX, snakeBoard.borderTopY, snakeBoard.borderRightX, snakeBoard.borderBottomY);

  // drawing snake
  for (byte x = snake.positions[0].x; x < snake.positions[0].x + snake.sizePerLen; x++) {
    for (byte y = snake.positions[0].y; y < snake.positions[0].y + snake.sizePerLen; y++) {
      lcd.setPixel(x, y);
    }
  }

  drawSnakeFood();
  isGameInitialized = true;
}
```

```
void checkButtonState(Button &button) {
  byte currentButtonState = digitalRead(button.pin);

  if (currentButtonState != button.lastButtonState) {
    // reseting time
    button.lastDebounceTime = millis();
  }

  if ((millis() - button.lastDebounceTime) > button.debounceDelay) {
    if (currentButtonState != button.buttonState) {
      button.buttonState = currentButtonState;
      if (currentButtonState == ACTIVATED) {
        switch (button.pin)
        {
          case buttonDownPin: buttonDownPressed();
            break;
          case buttonLeftPin: buttonLeftPressed();
            break;
          case buttonUpPin: buttonUpPressed();
            break;
          case buttonRightPin: buttonRightPressed();
            break;
        }
      }
    }
  }

  button.lastButtonState = currentButtonState;
}
```

```
struct Button {
  byte pin;
  volatile byte buttonState;
  volatile byte lastButtonState;

  volatile unsigned long lastDebounceTime = 0;
  unsigned long debounceDelay;

  Button(byte _pin, unsigned long _debounceDelay = BTN_DEBOUNCE_DELAY) {
    pin = _pin;
    debounceDelay = _debounceDelay;
  }
};
```

```
struct Snake_OBJ {
  volatile unsigned short maxSize = 20;
  Position* positions = new Position[maxSize];
  byte sizePerLen;
  // speed as milliseconds
  volatile unsigned short snakeSpeed;
  MoveDirections moveDirection = MoveDirections::up;
  // length of snake
  volatile short len = 1;
  volatile unsigned long lastTimeUpdated = 0;

  Snake_OBJ() {
  }

  Snake_OBJ(byte x, byte y, byte _sizePerLen = 1, unsigned short _snakeSpeed = 250) {
    positions[0] = Position(x, y);
    sizePerLen = _sizePerLen;
    snakeSpeed = _snakeSpeed;
  }

  void snakeSetPos(byte index, byte x, byte y) {
    positions[index] = Position(x, y);
  }

  void eatFood() {
    len++;
    snakeSpeed -= 5;
    if (len + 1 == maxSize) {
      maxSize += 20;
      Position* temp = positions;
      positions = new Position[maxSize];
      for (byte i = 0; i < len; ++i) {
        positions[i] = temp[i];
      }
      delete [] temp;
    }

    positions[len - 1] = Position(positions[len - 2].x, positions[len - 2].y);
  }
};
```