VIDYA JYOTHI
INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION

Joystick Game with LCD

A Project Report
Submitted
by

B.Laxman - 21911A3506
B.Mahesh- 21911A3507
B.Mallesh - 21911A3508

Department of Artificial Intelligence

Title : Joystick Game with LCD

Project Abstract : This project involves designing and implementing a simplified version of the Super Mario game using a 16x2 LCD display. The game features basic functionalities, such as character movement and simple obstacles, controlled via input buttons. The objective is to provide a minimalistic yet functional version of the classic game on a small-scale display.

Introduction :

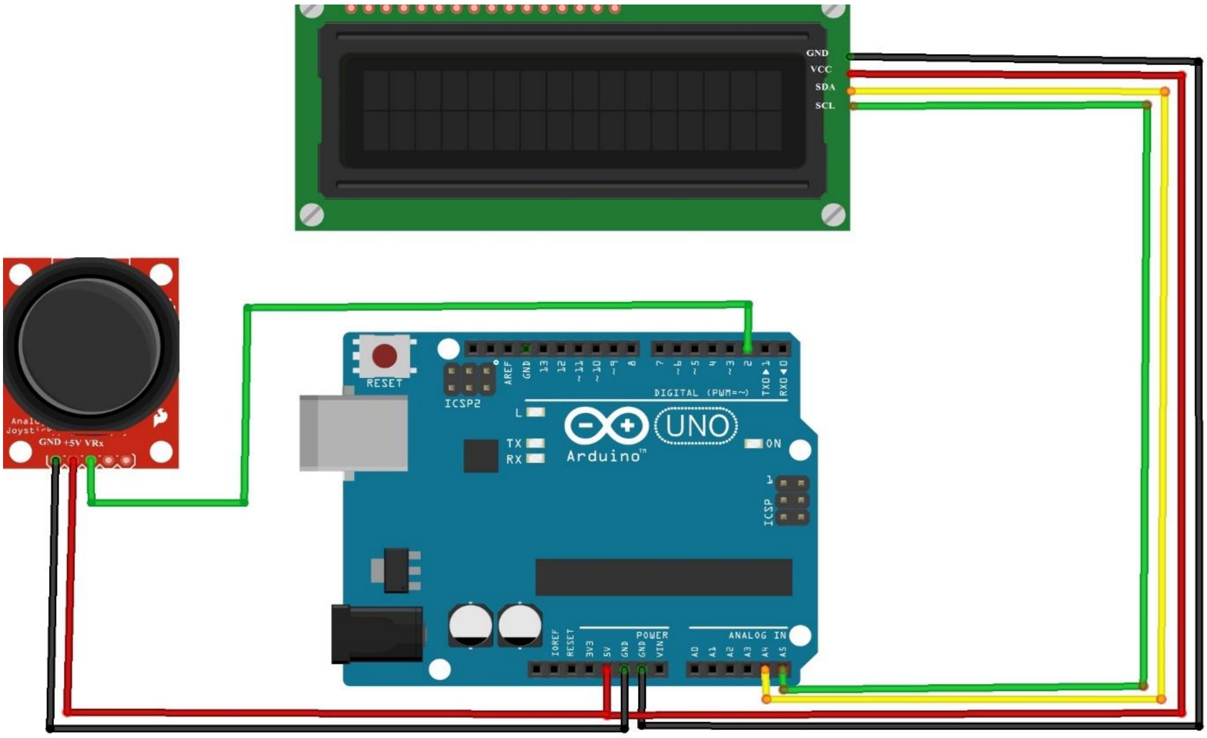
In this project, we are making a jumping jack game using Arduino Uno with a 16*2 LCD Display. In the sketch, a basic mini-program of a retro video game replicating the Super Mario Bros arcade game is a good code to explore and study if you are into programming.

Components Required :

1. Arduino UNO
2. Jumper Wires
3. Standard LCD - 16x2 with I2C Connector
4. Analog joystick

Description of the Project : The project involves creating a simple Super Mario game where the character can move horizontally across the 16x2 LCD screen, avoiding obstacles represented by specific characters. The game is controlled using push buttons that allow the player to move Mario left or right. The LCD screen is used to display the game state, including the position of Mario and obstacles.

Circuit Diagram :



Procedure:

1. **Setup the Hardware:**
   - Connect the 16x2 LCD display to the microcontroller using the appropriate pins (usually using 4-bit or 8-bit mode).
   - Connect the push buttons to the microcontroller for user inputs.
   - Use a breadboard to organize connections and ensure a stable setup.
   - Adjust the LCD contrast using a potentiometer.
2. **Initialize the LCD:**
   - Write the code to initialize the LCD and configure it for displaying characters.
   - Test the LCD by displaying simple text messages to ensure proper connections and functionality.
3. **Design the Game Logic:**
   - Create variables and functions to handle the game state, player position, obstacles, and score.
   - Implement the main game loop that updates the game state and refreshes the LCD display accordingly.
   - Handle user inputs through push buttons to control Mario's movements.
4. **Implement the Display Update:**

- Write functions to update the LCD display based on the game state.
- Use custom characters if needed to represent Mario and obstacles within the limited resolution of the LCD.

5. **Test and Debug:**
    - Test the game thoroughly to ensure it runs smoothly.
    - Debug any issues related to display updates, input handling, and game logic.

Code :

```
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

from backside of LCD

#define SPRITE_TERRAIN_EMPTY ' '

#define SPRITE_TERRAIN_SOLID 5

#define SPRITE_TERRAIN_SOLID_RIGHT 6

#define SPRITE_TERRAIN_SOLID_LEFT 7

#define Joy_X_axis 2

#define Autostartpin 1

#define PIN_READWRITE 10

#define PIN_CONTRAST 12


#define SPRITE_RUN1 1

#define SPRITE_RUN2 2

#define SPRITE_JUMP 3

#define SPRITE_JUMP_UPPER '.'
#define SPRITE_JUMP_LOWER 4


#define BOY_HORIZONTAL_POSITION 1

#define TERRAIN_WIDTH 16
#define TERRAIN_EMPTY 0
#define TERRAIN_LOWER_BLOCK 1
#define TERRAIN_UPPER_BLOCK 2
```

```
#define BOY_POSITION_OFF 0          // boy is invisible
#define BOY_POSITION_RUN_LOWER_1 1  // boy is running on lower row
(pose 1)
#define BOY_POSITION_RUN_LOWER_2 2  //(pose 2)

#define BOY_POSITION_JUMP_1 3       // Starting a jump
#define BOY_POSITION_JUMP_2 4       // Half-way up
#define BOY_POSITION_JUMP_3 5       // Jump is on upper row
#define BOY_POSITION_JUMP_4 6       // Jump is on upper row
#define BOY_POSITION_JUMP_5 7       // Jump is on upper row
#define BOY_POSITION_JUMP_6 8       // Jump is on upper row
#define BOY_POSITION_JUMP_7 9       // Half-way down
#define BOY_POSITION_JUMP_8 10      // About to land

#define BOY_POSITION_RUN_UPPER_1 11 // boy is running on upper row
(pose 1)
#define BOY_POSITION_RUN_UPPER_2 12 // (pose 2)

//LiquidCrystal lcd(11, 9, 6, 5, 4, 3);
static char terrainUpper[TERRAIN_WIDTH + 1];
static char terrainLower[TERRAIN_WIDTH + 1];
static bool buttonPushed = false;

void initializeGraphics(){
  static 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,
    // Ground right
    B00011,
    B00011,
    B00011,
    B00011,
    B00011,
    B00011,
    B00011,
    B00011,
    // Ground left
    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;
    }
  }
}

bool drawBoy(byte position, char* terrainUpper, char* terrainLower,
unsigned int score) {
  bool collide = false;
  char upperSave = terrainUpper[BOY_HORIZONTAL_POSITION];
  char lowerSave = terrainLower[BOY_HORIZONTAL_POSITION];
  byte upper, lower;
  switch (position) {
    case BOY_POSITION_OFF:
      upper = lower = SPRITE_TERRAIN_EMPTY;
      break;
    case BOY_POSITION_RUN_LOWER_1:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN1;
      break;
    case BOY_POSITION_RUN_LOWER_2:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_RUN2;
      break;
    case BOY_POSITION_JUMP_1:
    case BOY_POSITION_JUMP_8:
      upper = SPRITE_TERRAIN_EMPTY;
      lower = SPRITE_JUMP;
      break;
    case BOY_POSITION_JUMP_2:
    case BOY_POSITION_JUMP_7:
      upper = SPRITE_JUMP_UPPER;
      lower = SPRITE_JUMP_LOWER;
      break;
    case BOY_POSITION_JUMP_3:
    case BOY_POSITION_JUMP_4:
    case BOY_POSITION_JUMP_5:
    case BOY_POSITION_JUMP_6:
      upper = SPRITE_JUMP;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case BOY_POSITION_RUN_UPPER_1:
      upper = SPRITE_RUN1;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
    case BOY_POSITION_RUN_UPPER_2:
      upper = SPRITE_RUN2;
      lower = SPRITE_TERRAIN_EMPTY;
      break;
  }
  if (upper != ' ') {
    terrainUpper[BOY_HORIZONTAL_POSITION] = upper;
    collide = (upperSave == SPRITE_TERRAIN_EMPTY) ? false : true;
  }
  if (lower != ' ') {
    terrainLower[BOY_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[BOY_HORIZONTAL_POSITION] = upperSave;
  terrainLower[BOY_HORIZONTAL_POSITION] = lowerSave;
  return collide;
}

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

void setup(){
  pinMode(PIN_READWRITE, OUTPUT);
  digitalWrite(PIN_READWRITE, LOW);
  pinMode(PIN_CONTRAST, OUTPUT);
  digitalWrite(PIN_CONTRAST, LOW);
  pinMode(Joy_X_axis, INPUT);
  digitalWrite(Joy_X_axis, HIGH);
  pinMode(Autostartpin, OUTPUT);
  digitalWrite(Autostartpin, HIGH);

  // Digital pin 2 maps to interrupt 0
  attachInterrupt(0/*PIN_BUTTON*/, buttonPush, FALLING);

  initializeGraphics();

  lcd.begin(16, 2);
}

void loop(){
  static byte boyPos = BOY_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;
  if (!playing) {
    drawBoy((blink) ? BOY_POSITION_OFF : boyPos, terrainUpper,
terrainLower, distance >> 3);
    if (blink) {
      lcd.setCursor(0,0);
      lcd.print("Push to  Start");
    }
    delay(100);
    blink = !blink;
    if (buttonPushed) {
      initializeGraphics();
      boyPos = BOY_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 (boyPos <= BOY_POSITION_RUN_LOWER_2) boyPos =
BOY_POSITION_JUMP_1;
    buttonPushed = false;
  }

  if (drawBoy(boyPos, terrainUpper, terrainLower, distance >> 3)) {
    playing = false; // The boy collided with something. Too bad.
  } else {
    if (boyPos == BOY_POSITION_RUN_LOWER_2 || boyPos ==
BOY_POSITION_JUMP_8) {
      boyPos = BOY_POSITION_RUN_LOWER_1;
    } else if ((boyPos >= BOY_POSITION_JUMP_3 && boyPos <=
BOY_POSITION_JUMP_5) && terrainLower[BOY_HORIZONTAL_POSITION] !=
SPRITE_TERRAIN_EMPTY) {
      boyPos = BOY_POSITION_RUN_UPPER_1;
    } else if (boyPos >= BOY_POSITION_RUN_UPPER_1 &&
terrainLower[BOY_HORIZONTAL_POSITION] == SPRITE_TERRAIN_EMPTY) {
      boyPos = BOY_POSITION_JUMP_5;
    } else if (boyPos == BOY_POSITION_RUN_UPPER_2) {
      boyPos = BOY_POSITION_RUN_UPPER_1;
    } else {
      ++boyPos;
    }
    ++distance;

    digitalWrite(Autostartpin, terrainLower[BOY_HORIZONTAL_POSITION +
2] == SPRITE_TERRAIN_EMPTY ? HIGH : LOW);
  }
  delay(100);

}
```

Result : When running the project, the 16x2 LCD display shows Mario as he moves based on user input. Obstacles move from right to left across the screen. If Mario collides with an obstacle, the game displays "Game Over" and resets.

Conclusion : This project successfully demonstrates how a simple version of the Super Mario game can be implemented on a 16x2 LCD display. It highlights the challenges and creative solutions required to adapt complex games for small-scale, low-resolution displays. This project serves as a foundation for more advanced game development on embedded systems.