8051 BASED DINO GAME

by

MIDHUN DHASS D 24BLC1037
LINGA RAJA R 24BVD1110

A project report submitted to

Dr. Abraham Sudharson Ponraj

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfillment of the requirements for the course of

BECE204L – MICROPROCESSORS AND MICROCONTROLLERS

in

B. Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

Vandalur – Kelambakkam Road

Chennai – 600127

November 2024

BONAFIDE CERTIFICATE

Certified that this project report entitled "**8051 BASED DINO GAME"** is a bonafide work of **MIDHUN DHASS D-24BLC1037 and LINGA RAJA R -24BVD1110 who** carried out the Project work under my supervision and guidance for **BECE204L – MICROPROCESSORS AND MICROCONTROLLERS.**

Dr. Abraham Sudharson Ponraj
Associate Professor
School of Electronics Engineering (SENSE),
VIT, Chennai
Chennai – 600 127.

ABSTRACT

This project details the design and implementation of an interactive **Dino Runner Game** using the classic **8051 Microcontroller** (AT89C51) as the core processing unit. The primary objective was to successfully implement the game logic and demonstrate hardware interfacing across **two distinct display technologies** while incorporating advanced user features.

The work is split into two complementary systems. The first system interfaces the 8051 with a **JHD204A 20x4 LCD module** to display the game using custom character generation. The second system utilizes a **0.96" OLED display** for a higher-resolution graphical implementation, employing the 8051 for visual output and control. User interaction, essential for game control (like 'jump') and feature selection, is managed via a **4-button input module**, demonstrating effective **bidirectional I/O operations** where a single port (Port 2) is used for both output (to the OLED) and input (from the buttons).

The final system successfully delivers a comprehensive embedded gaming experience, featuring:

- **Dual-Display Capability:** Implementation on both the character-based LCD and the pixel-based OLED.
- **Dual Game Modes:** Offering different play styles or difficulty levels.
- **Adjustable Brightness:** Providing user control over display output.

This project serves as a robust demonstration of the **8051 microcontroller's** versatility and capability in handling real-time data processing, graphical output, and complex I/O management within a resource-constrained embedded system environment.

TABLE OF CONTENTS

SERIAL NO.		TITLE	PAGE NO.
		ABSTRACT	
		ACKNOWLEDGEMENT	
1		INTRODUCTION	5
	1.1	OBJECTIVES AND GOALS	5
	1.2	APPLICATIONS	5
	1.3	FEATURES	5
2		DESIGN	6
	2.1	BLOCK DIAGRAM	6
	2.2	HARDWARE ANALYSIS	8
	2.3	(SNAPSHOTS-PROJECT , TEAM, RESULTS)	9
3	3.1	SOFTWARE –CODING AND ANALYSIS	13
		(SNAPSHOTS OF CODING AND RESULTS)	23
4		CONCLUSION AND FUTURE WORK	24
	4.1	RESULT, CONCLUSION AND INFERENCE	24
	4.2	FUTURE WORK COST	24
5		REFERENCES	27
6		PHOTO GRAPH OF THE PROJECT ALONG WITH THE TEAM MEMBERS	27

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

- Implement the classic "Dino Runner" game logic using the 8051 Microcontroller (specifically address the resource constraints of the 8051).
- Demonstrate game functionality on two distinct display technologies: LCD 20x4 and 0.96" OLED.
- Incorporate advanced features like two distinct game modes and a user-adjustable brightness control (for the OLED or LCD backlight, depending on implementation).

1.2 APPLICATIONS

- Educational Tool: Demonstrating real-time I/O, display interfacing, and game logic development on a core microcontroller.
- Low-Cost Gaming: Creating a simple, standalone entertainment device.

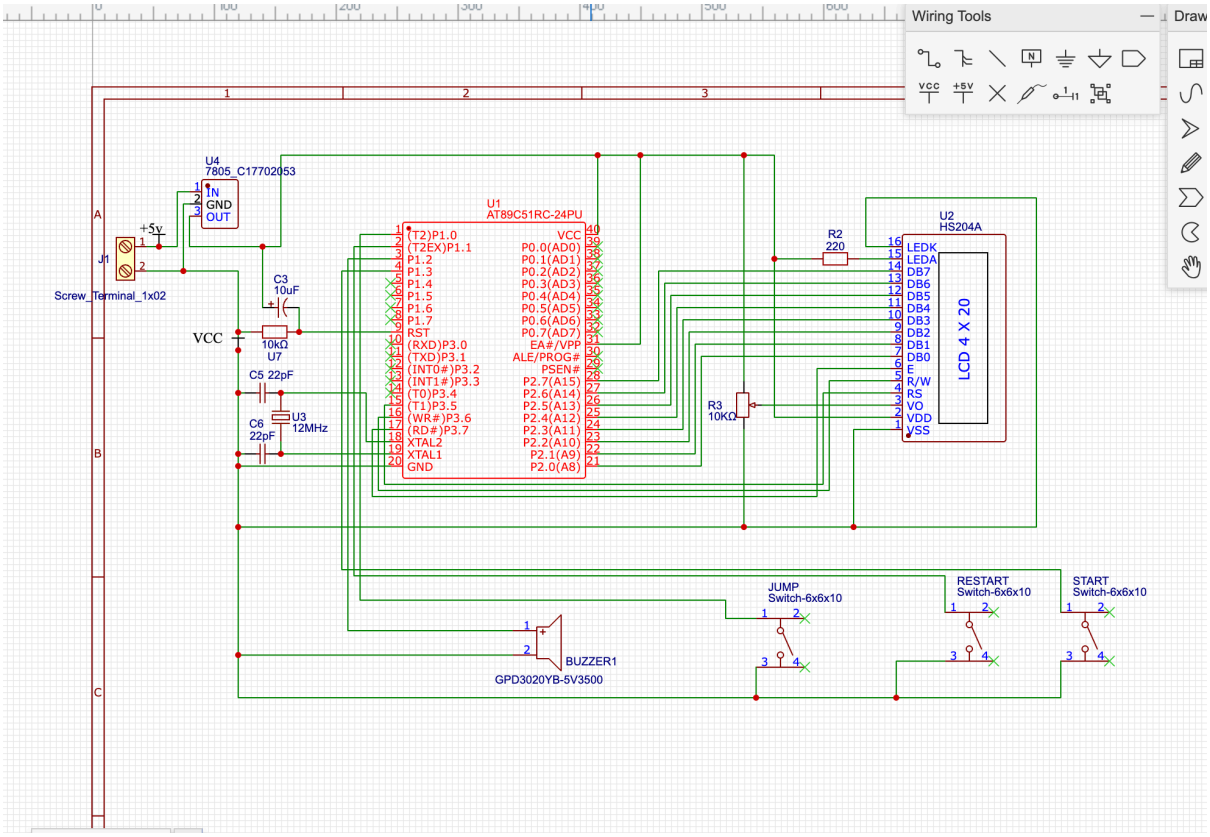
1.3 FEATURES

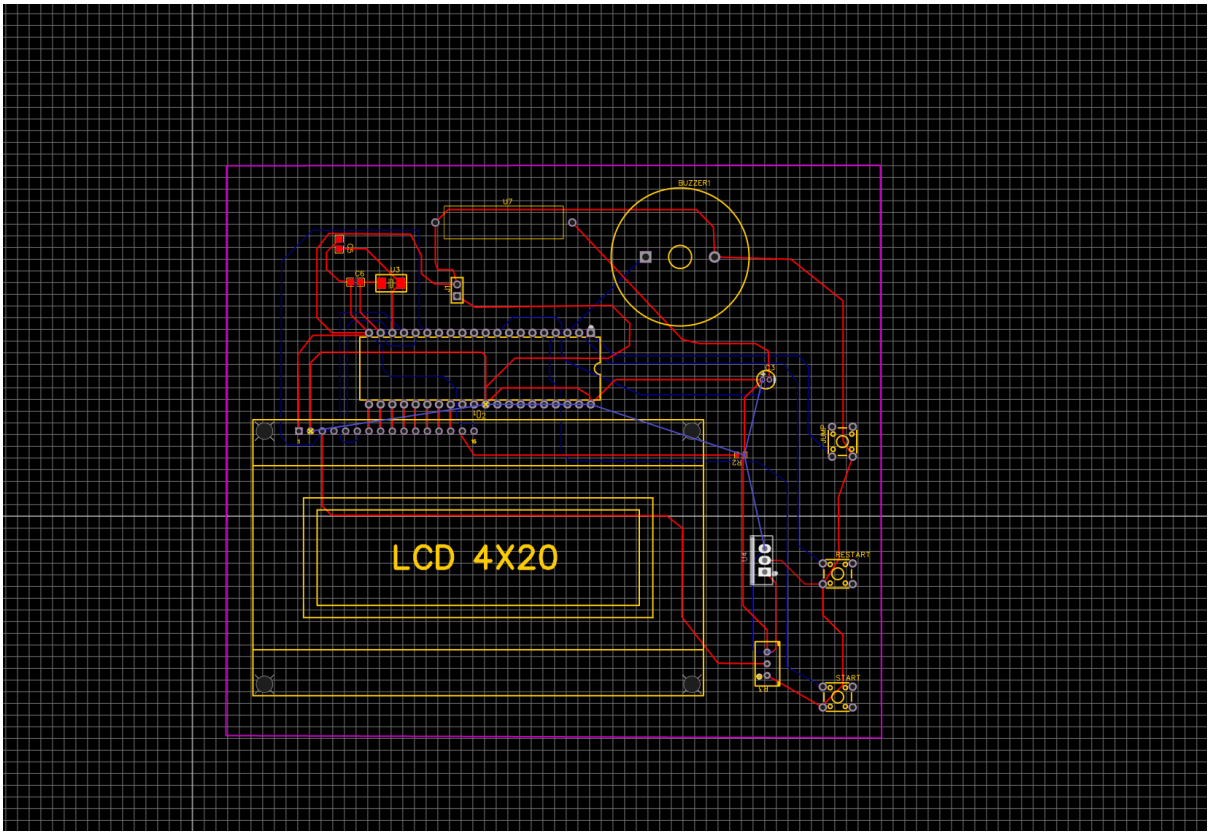
- Game Modes: Implement at least two variations (e.g., Classic Speed and High-Speed/Hardcore mode).
- Brightness Control: Implement adjustable display brightness (likely via PWM for the LCD backlight or software control for the OLED).

2. DESIGN

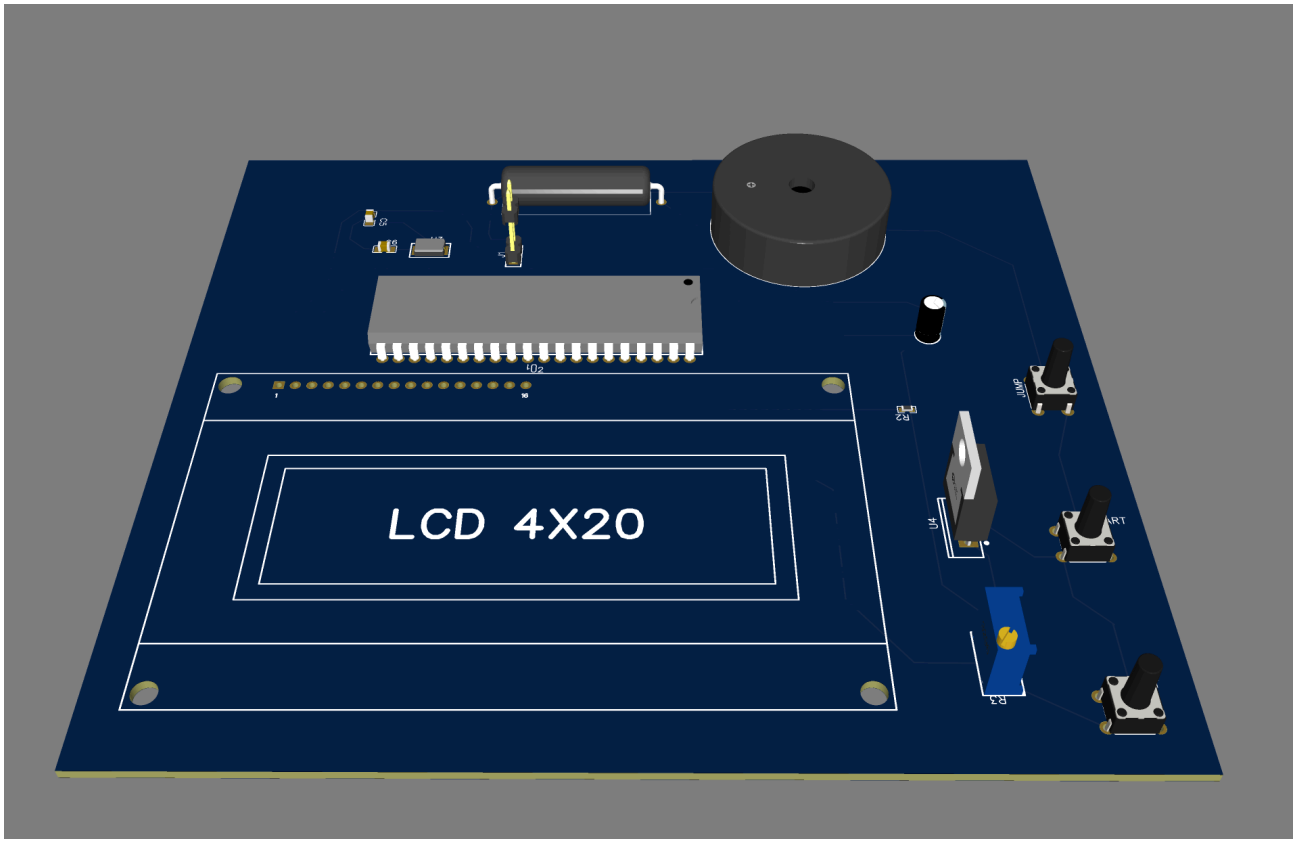
2.1 BLOCK DIAGRAM

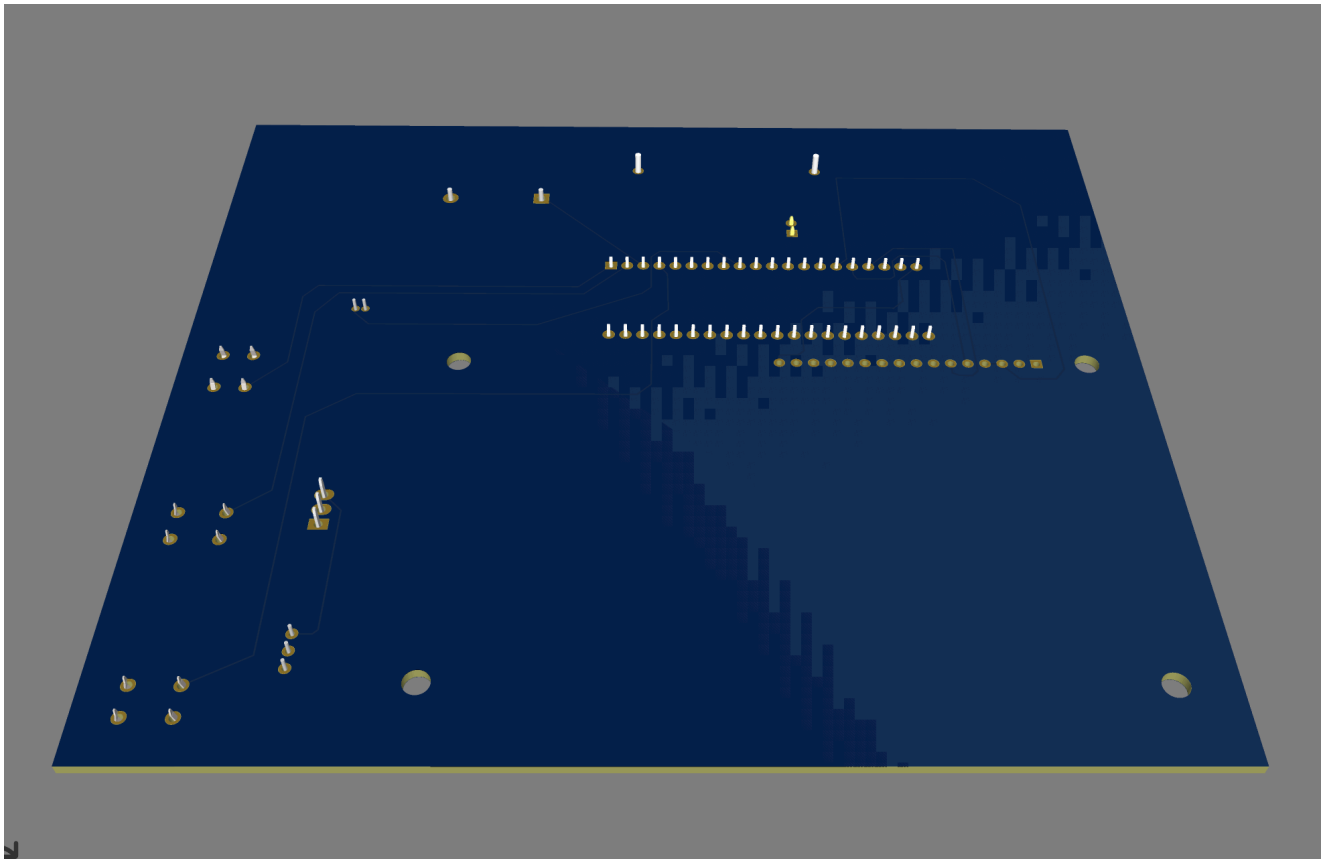
For LCD





PCB BOAED DESIGN:





2.2 HARWARE ANALYSIS

1)For LCD

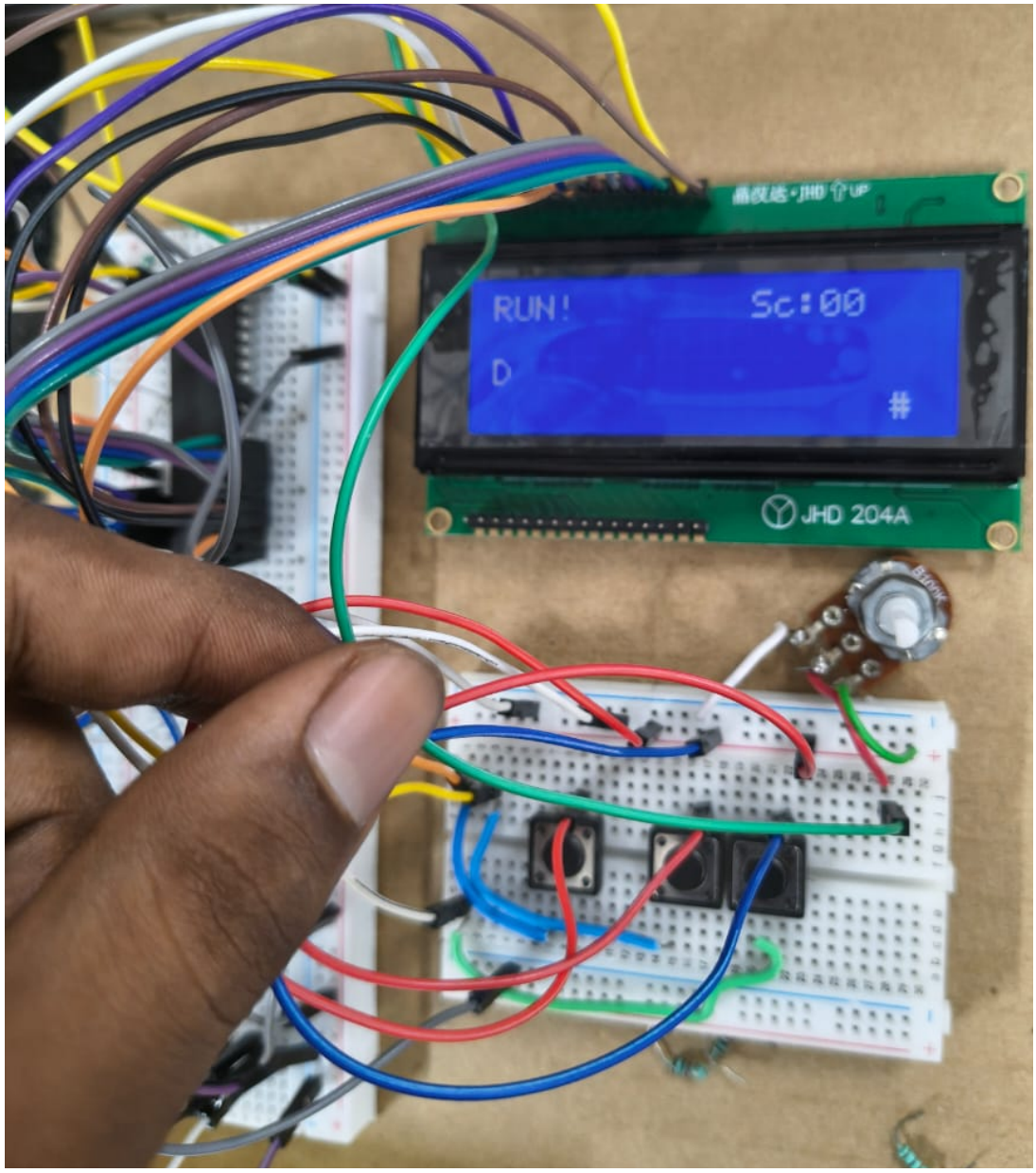
S.No	Component	Quantity	Description
1	**AT89C51 Microcontroller**	1	8-bit 8051 family MCU (40-pin DIP)
2	**JHD204A 20×4 LCD Module**	1	20 characters × 4 lines, HD44780 controller
3	**12 MHz Crystal**	1	Provides clock signal for MCU
4	**33 pF Capacitors**	2	Used with the crystal for stability
5	**10 µF Capacitor**	1	Reset circuit
6	**10 kΩ Resistor**	1	Reset pull-down resistor
7	**10 kΩ Potentiometer**	1	Adjusts LCD contrast
8	**100 Ω Resistor**	1	Limits LCD backlight current
9	**Push Button**	1	Manual reset switch
10	**7805 Voltage Regulator**	1	Regulates 5 V from external supply
11	**Breadboard / PCB**	1	For assembling circuit
12	**Jumper Wires**	—	For connections
13	**Power Supply (9–12 V DC)**	1	External adapter or battery

2)For OLE

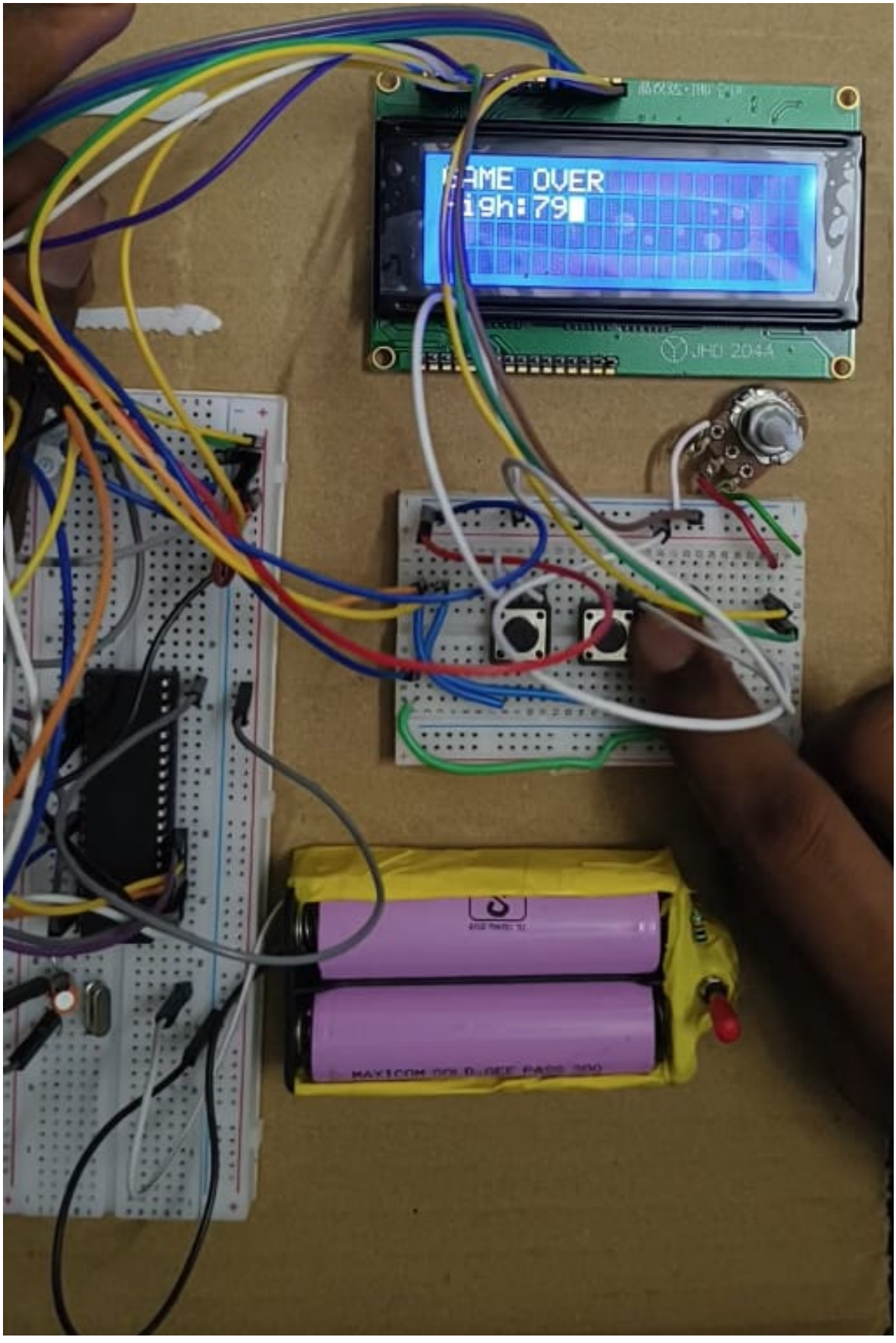
S.No	Component	Quantity	Description
1	**AT89C51 Microcontroller** (8051 Devkit)	1	8-bit MCU with 4 KB/8 KB Flash, 32 I/O pins
2	**0.96" or 1.3" OLED Display**	1	Monochrome (128×64 or 128×32), SPI or I²C interface (SSD1306 controller)
3	**4-Button Input Module**	1	4 tactile push buttons with pull-up resistors (Used for Jump, Game Mode, and Brightness control)
4	**12 MHz Crystal Oscillator**	1	Provides system clock to MCU

S.No	Component	Quantity	Description
5	**33 pF Capacitors**	2	Crystal stabilizing capacitors
6	**10 µF Capacitor**	1	Used in reset circuit
7	**10 kΩ Resistor**	1	Reset pull-down resistor
8	**4.7 kΩ Series Resistors** (Optional)	7	Protect shared lines between OLED and buttons
9	**7805 Voltage**	1	Regulates 5 V supply
10	**Jumper Wires and Breadboard / PCB**	1	For wiring connections
11	**Power Supply (9–12 V DC)**	1	For devkit and display operation

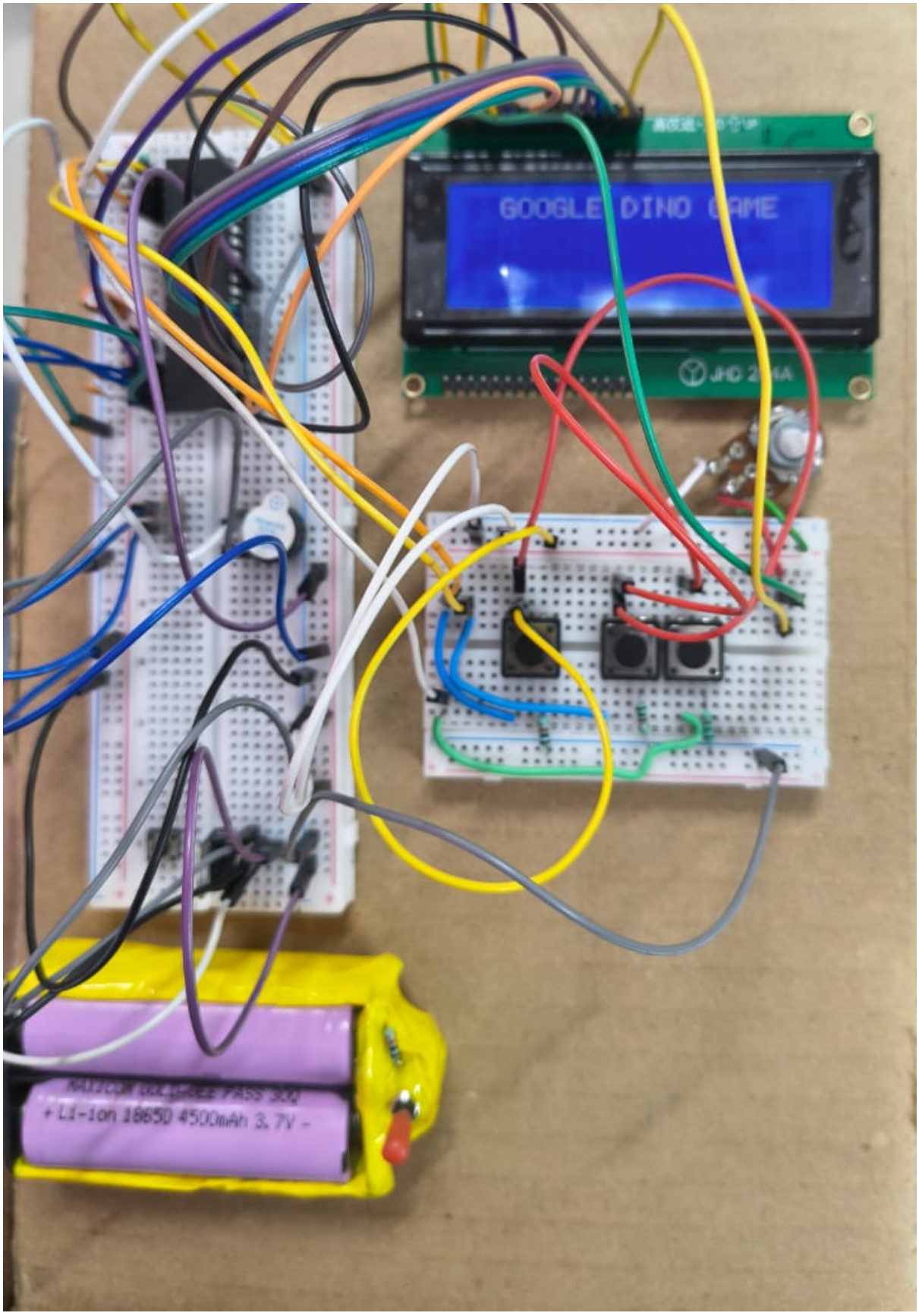
GAME RUNNING SCREEN



GAME END SCREEN



GAME START SCREEN



OLED GAME

3. SOFTWARE

3.1 SOFTWARE –CODING AND ANALYSIS

C Programing (USING KEIL MICROVISION SOFTWARE)

1)For LCD

```
#include <reg51.h>

/* ================== 8051 PIN CONFIG (4-BIT) ================== */
/* LCD 4-Bit Data Port */
#define LCD_DATA_PORT P2    /* Data D4-D7 on P2.4-P2.7 */

/* LCD Control Pins */
sbit RS = P3^5;          /* Register Select */
sbit EN = P3^7;          /* Enable */
/* NOTE: LCD R/W Pin (Pin 5) must be connected to GND */

/* Game Control Pins */
sbit BTN_JUMP    = P1^0;  /* Game 1 / Jump */
sbit BUZZER      = P1^1;  /* Buzzer */
sbit BTN_RESTART = P1^2;  /* Exit to Menu */
sbit BTN_START   = P1^3;  /* Game 2 */

/* ================== TYPEDEFS ================== */
typedef unsigned char byte;
typedef unsigned char bool;
#define true 1
#define false 0

/* ================== GLOBAL STATE ================== */
bool inMenu = true;

/* ====== GAME 1 VARIABLES ====== */
bool dinoOnGround = true;
bool playState = false;
unsigned int highScore = 0;
int dist = 0;
int distTwo = 0;
unsigned int score = 0;
unsigned char prng_seed = 1;

/* ================== CUSTOM CHARS (in Code Memory) ================== */
unsigned char code dino[8] = {
  0x00, 0x07, 0x05, 0x17, 0x1C, 0x1F, 0x0D, 0x0C
};

unsigned char code tree[8] = {
  0x03, 0x1B, 0x1B, 0x1B, 0x1B, 0x1F, 0x0E, 0x0E
};

/* ================== 8051 LCD DRIVER (4-BIT) ================== */
void delay_us(unsigned int us) {
    while(us--) { /* Calibrated for ~1us at 12MHz */ }
}

void delay_ms(unsigned int ms) {
    unsigned int i, j;
```

```
    for (i = 0; i < ms; i++)
        for (j = 0; j < 120; j++);  // ~1ms at 11.0592MHz
}

void lcd_pulse(void) {
    EN = 1;
    delay_us(5);   // 5 microsecond pulse
    EN = 0;
    delay_ms(1);   // Wait for command to process
}

void lcd_cmd(unsigned char cmd) {
    unsigned char temp;

    // Send UPPER nibble (4 bits)
    temp = cmd & 0xF0;
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | temp;
    RS = 0;
    lcd_pulse();

    // Send LOWER nibble (4 bits)
    temp = (cmd << 4) & 0xF0;
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | temp;
    RS = 0;
    lcd_pulse();
}

void lcd_data(unsigned char dat) {
    unsigned char temp;

    // Send UPPER nibble
    temp = dat & 0xF0;
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | temp;
    RS = 1;
    lcd_pulse();

    // Send LOWER nibble
    temp = (dat << 4) & 0xF0;
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | temp;
    RS = 1;
    lcd_pulse();
}

void lcd_init(void) {
    delay_ms(20);

    // --- 4-bit init sequence ---
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | 0x30;
    RS = 0;
    lcd_pulse();
    delay_ms(5);
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | 0x30;
    lcd_pulse();
    delay_ms(1);
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | 0x30;
    lcd_pulse();
    delay_ms(1);
    LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | 0x20;
    lcd_pulse();
    delay_ms(1);

    // --- End of special sequence ---
    lcd_cmd(0x28); // 4-bit mode, 2-line, 5x7 font
```

```
    lcd_cmd(0x0C); // Display ON, Cursor OFF
    lcd_cmd(0x06); // Entry Mode
    lcd_cmd(0x01); // Clear Display
    delay_ms(2);
}

void lcd_clear(void) {
    lcd_cmd(0x01);
}

void lcd_gotoxy(unsigned char row, unsigned char col) {
    unsigned char row_addr[] = {0x80, 0xC0, 0x94, 0xD4};
    lcd_cmd(row_addr[row] + col);
}

void lcd_puts(char *s) {
    while (*s) {
        lcd_data(*s++);
    }
}

/* ================== HELPER FUNCTIONS ================== */

unsigned char rand_8bit() {
    prng_seed ^= prng_seed << 1;
    prng_seed ^= prng_seed >> 1;
    prng_seed ^= prng_seed << 2;
    return prng_seed;
}

int random(int min, int max) {
    return (rand_8bit() % (max - min + 1)) + min;
}

void lcd_print_number(unsigned int n) {
    char buffer[6];
    int i = 4;
    buffer[5] = '\0';

    if (n == 0) {
        lcd_data('0');
        return;
    }

    while (n > 0) {
        buffer[i--] = (n % 10) + '0';
        n /= 10;
    }
    lcd_puts(&buffer[i+1]);
}

void beep(void) {
    BUZZER = 1;
    delay_ms(100);
    BUZZER = 0;
}

void lcd_load_custom_chars() {
    unsigned char i;

    // Load tree to char 6
    lcd_cmd(0x40 + (6 * 8));
    for(i=0; i<8; i++) lcd_data(tree[i]);
```

```
    // Load dino to char 7
    lcd_cmd(0x40 + (7 * 8));
    for(i=0; i<8; i++) lcd_data(dino[i]);
}

/* ================== GAME FUNCTIONS ================== */

void startDinoGame() {
    int i;
    playState = true;
    score = 0;
    lcd_clear();
    lcd_load_custom_chars();

    while (playState) {
        dist = random(4, 9);
        distTwo = random(4, 9);

        for (i = 15; i >= -(dist + distTwo); i--) {
            lcd_gotoxy(0, 13);
            lcd_print_number(score);

            if (BTN_JUMP == 0) {
                beep();
                lcd_gotoxy(0, 1);
                lcd_data(7);      // Write dino (char 7)
                lcd_gotoxy(1, 1);
                lcd_data(' ');    // Clear ground
                dinoOnGround = false;
            } else {
                lcd_gotoxy(1, 1);
                lcd_data(7);      // Write dino (char 7)
                lcd_gotoxy(0, 1);
                lcd_data(' ');    // Clear air
                dinoOnGround = true;
            }

            lcd_gotoxy(1, i);
            lcd_data(6); // Write tree (char 6)
            lcd_gotoxy(1, i + dist);
            lcd_data(6);
            lcd_gotoxy(1, i + dist + distTwo);
            lcd_data(6);

            if ((i == 1 || (i + dist) == 1 || (i + dist + distTwo) == 1) && dinoOnGround) {
                lcd_clear();
                lcd_gotoxy(0, 0);
                lcd_puts("GAME OVER");
                if (score > highScore) highScore = score;
                lcd_gotoxy(1, 0);
                lcd_puts("High:");
                lcd_print_number(highScore);
                delay_ms(2000);
                playState = false;
                inMenu = true;
                lcd_clear();
                return;
            }

            score++;
            delay_ms(300);
```

```
            if (BTN_RESTART == 0) {
                playState = false;
                inMenu = true;
                lcd_clear();
                return;
            }
        }
    }
}

void startTerrainGame() {
    int pos = 0;
    lcd_clear();
    lcd_gotoxy(0, 0);
    lcd_puts("Terrain Game");
    delay_ms(1000);
    lcd_clear();

    while (true) {
        lcd_clear();
        lcd_gotoxy(1, pos);
        lcd_data('#');
        lcd_gotoxy(0, 1);
        lcd_puts("RUN!");
        delay_ms(100);

        pos--;
        if (pos < 0) pos = 15;

        if (BTN_RESTART == 0) {
            inMenu = true;
            lcd_clear();
            return;
        }
    }
}

void showMenu() {
    lcd_clear();
    lcd_gotoxy(0, 0);
    lcd_puts("Select Game:");
    lcd_gotoxy(1, 0);
    lcd_puts("1.Dino  2.Terrain");
    delay_ms(200);

    if (BTN_JUMP == 0) {
        delay_ms(200);
        startDinoGame();
    }
    if (BTN_START == 0) {
        delay_ms(200);
        startTerrainGame();
    }
}

/* ================== MAIN ================== */
void main(void) {
    P1 = 0xFF;      // Set P1 as input (for buttons)
    BUZZER = 0;
    lcd_init();
    lcd_load_custom_chars();
    prng_seed = 1;
```

```
    while (1) {
        if (inMenu) {
            showMenu();
        }
    }
}
```

2)For OLED

```
/* ================== PIN MAP ==================
   OLED I2C:
     P2.1 -> SCL
     P2.2 -> SDA

   Buttons (active LOW):
     P2.3 -> START
     P2.4 -> JUMP
     P2.5 -> RESTART
   ============================================= */
sbit OLED_SCL    = P2^1;
sbit OLED_SDA    = P2^2;
sbit BTN_START   = P2^3;
sbit BTN_JUMP    = P2^4;
sbit BTN_RESTART = P2^5;

/* ========== SSD1306 I2C address (0x3C << 1) ========== */
#define OLED_ADDR  0x78   /* 0x3C shifted left */

/* ========== GAME VARS (stay in DATA) ========== */
bit game_running = 0;
bit dino_up      = 0;
unsigned char jump_timer = 0;
unsigned char obstacle_x = 120;   // start from right
unsigned int  score      = 0;

/* ========== SPRITES IN CODE MEMORY ========== */
/* Put them in code so they don't occupy DATA RAM */
unsigned char code dino8[8] = {
    0x18, 0x3C, 0x7E, 0x5A,
    0x18, 0x18, 0x38, 0x00
};

unsigned char code cactus8[8] = {
    0x10, 0x10, 0x14, 0x14,
    0x7C, 0x10, 0x10, 0x00
};

/* 10 digits × 8 bytes = 80 bytes ? must be in code */
unsigned char code digit8x8[10][8] = {
    {0x3C,0x66,0x6E,0x76,0x66,0x66,0x3C,0x00}, //0
    {0x18,0x38,0x18,0x18,0x18,0x18,0x7E,0x00}, //1
    {0x3C,0x66,0x06,0x0C,0x30,0x60,0x7E,0x00}, //2
```

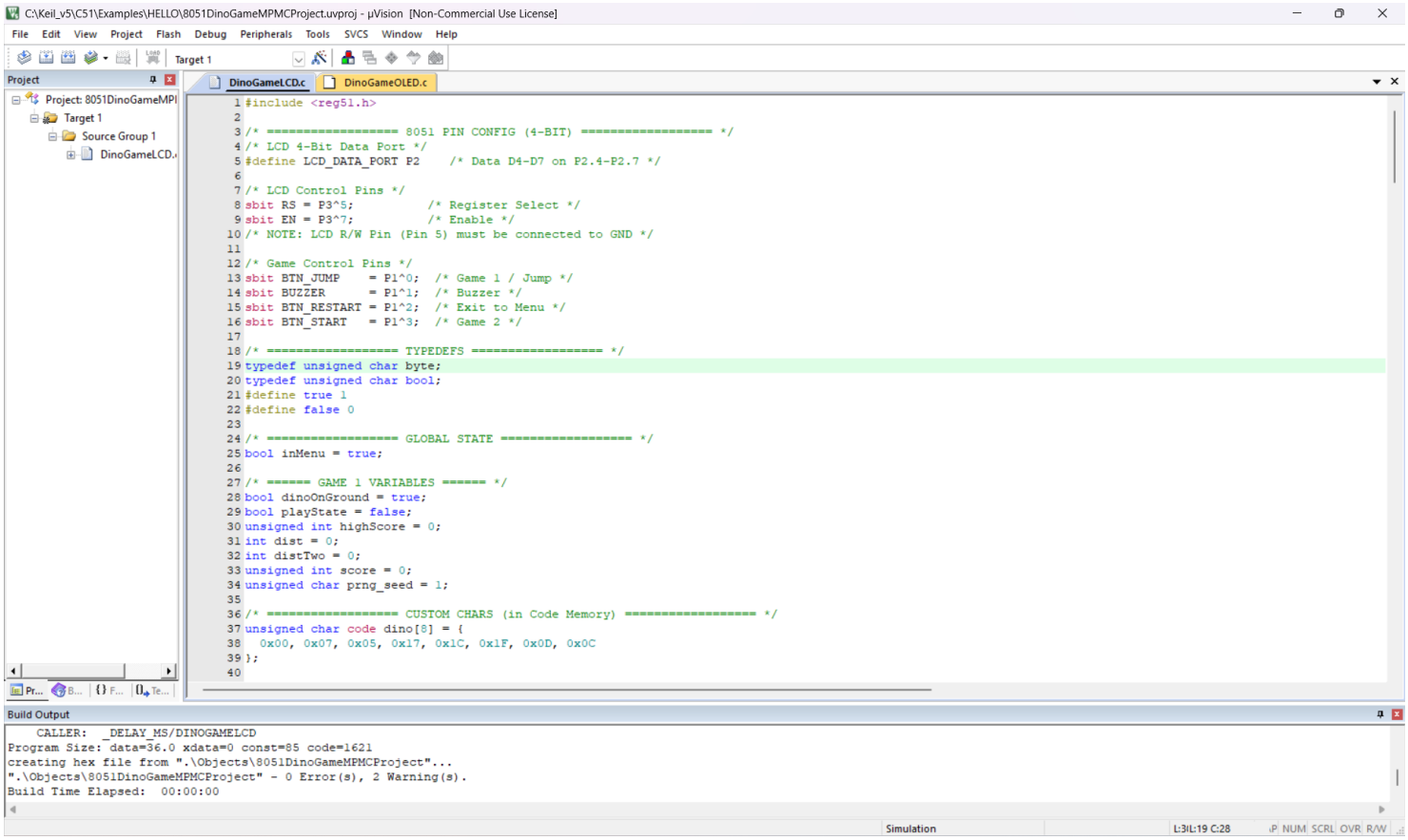
```
    {0x3C,0x66,0x06,0x1C,0x06,0x66,0x3C,0x00}, //3
    {0x0C,0x1C,0x3C,0x6C,0x7E,0x0C,0x0C,0x00}, //4
    {0x7E,0x60,0x7C,0x06,0x06,0x66,0x3C,0x00}, //5
    {0x1C,0x30,0x60,0x7C,0x66,0x66,0x3C,0x00}, //6
    {0x7E,0x06,0x0C,0x18,0x30,0x30,0x30,0x00}, //7
    {0x3C,0x66,0x66,0x3C,0x66,0x66,0x3C,0x00}, //8
    {0x3C,0x66,0x66,0x3E,0x06,0x0C,0x38,0x00}  //9
};

/* ========== UTILS ========== */
void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 120; j++);
}

/* ========== I2C BITBANG ========== */
void i2c_start(void) {
    OLED_SDA = 1; OLED_SCL = 1;
    OLED_SDA = 0;
    OLED_SCL = 0;
}

void i2c_stop(void) {
    OLED_SDA = 0; OLED_SCL = 1;
    OLED_SDA = 1;
}

void i2c_write(unsigned char dat) {
    unsigned char i;
    for (i = 0; i < 8; i++) {
        OLED_SDA = (dat & 0x80) ? 1 : 0;
        OLED_SCL = 1;
        OLED_SCL = 0;
        dat <<= 1;
    }
    /* ACK */
    OLED_SDA = 1;
    OLED_SCL = 1;
    OLED_SCL = 0;
}

/* ========== OLED LOW-LEVEL ========== */
void oled_cmd(unsigned char cmd) {
    i2c_start();
    i2c_write(OLED_ADDR);
    i2c_write(0x00);   // command
    i2c_write(cmd);
    i2c_stop();
}

/* renamed to avoid 'data' keyword */
void oled_write_data(unsigned char val) {
    i2c_start();
    i2c_write(OLED_ADDR);
```

```
    i2c_write(0x40);   // data
    i2c_write(val);
    i2c_stop();
}

void oled_setpos(unsigned char x, unsigned char page) {
    oled_cmd(0xB0 | (page & 0x07));
    oled_cmd(0x00 | (x & 0x0F));
    oled_cmd(0x10 | ((x >> 4) & 0x0F));
}

void oled_clear(void) {
    unsigned char page, col;
    for (page = 0; page < 8; page++) {
        oled_setpos(0, page);
        for (col = 0; col < 128; col++) {
            oled_write_data(0x00);
        }
    }
}
void oled_init(void) {
    delay_ms(100);
    oled_cmd(0xAE);
    oled_cmd(0x20); oled_cmd(0x00); // horizontal addressing
    oled_cmd(0xB0);
    oled_cmd(0xC8);
    oled_cmd(0x00);
    oled_cmd(0x10);
    oled_cmd(0x40);
    oled_cmd(0x81); oled_cmd(0x7F);
    oled_cmd(0xA1);
    oled_cmd(0xA6);
    oled_cmd(0xA8); oled_cmd(0x3F);
    oled_cmd(0xA4);
    oled_cmd(0xD3); oled_cmd(0x00);
    oled_cmd(0xD5); oled_cmd(0xF0);
    oled_cmd(0xD9); oled_cmd(0x22);
    oled_cmd(0xDA); oled_cmd(0x12);
    oled_cmd(0xDB); oled_cmd(0x20);
    oled_cmd(0x8D); oled_cmd(0x14);
    oled_cmd(0xAF);
    oled_clear();
}
/* draw an 8x8 sprite from CODE memory */
void oled_draw_sprite(unsigned char x, unsigned char page, unsigned char code *spr) {
    unsigned char i;
    oled_setpos(x, page);
    for (i = 0; i < 8; i++)
        oled_write_data(spr[i]);   // Keil will read from code
}
/* draw one digit 0-9 at x,page from CODE memory */
void oled_draw_digit(unsigned char x, unsigned char page, unsigned char d) {
   unsigned char i;
    if (d > 9) d = 0;
    oled_setpos(x, page);
```

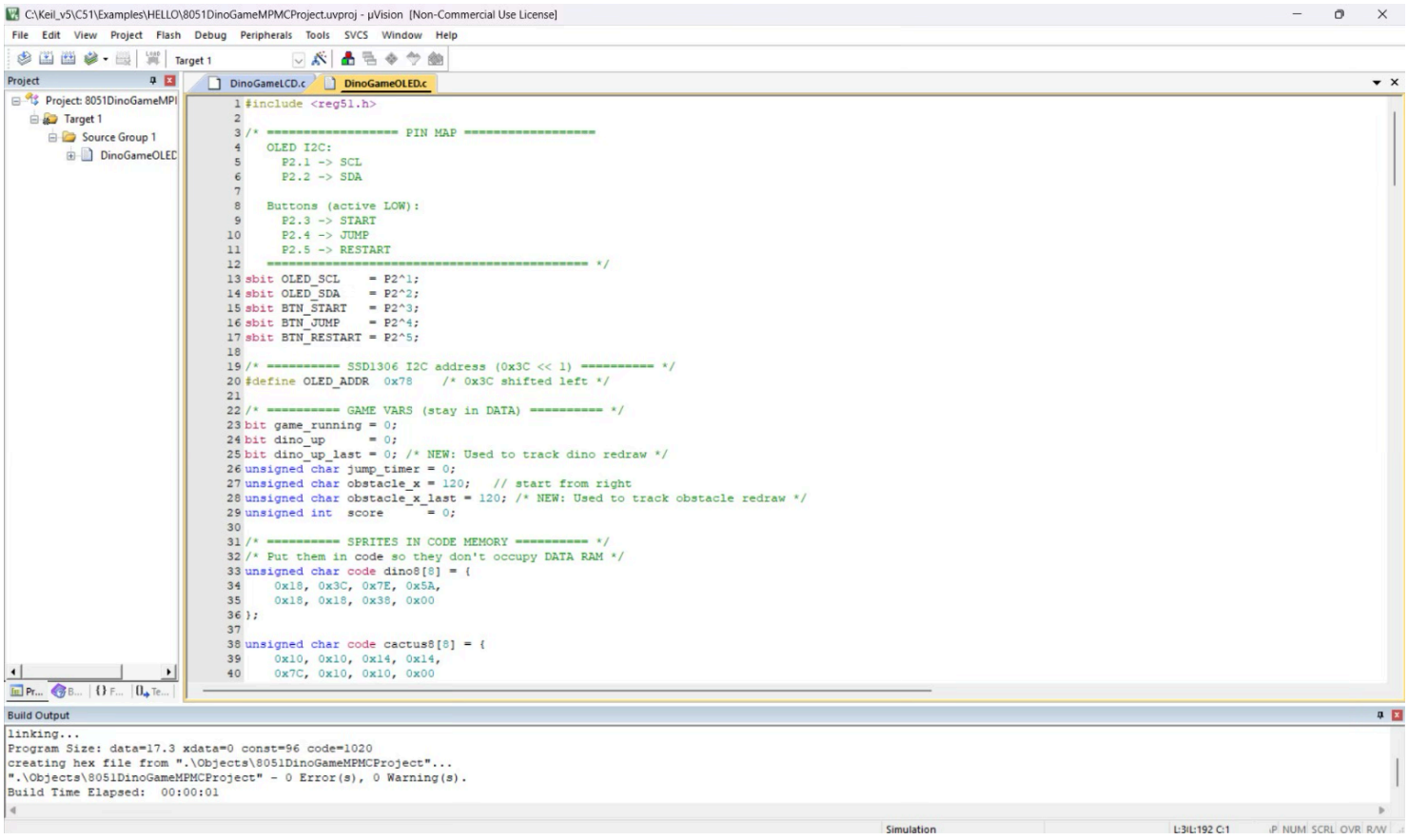
```
    for (i = 0; i < 8; i++)
        oled_write_data(digit8x8[d][i]);  // read from code
}

/* show score as 4 digits at top (page 0) */
void oled_draw_score(unsigned int s) {
    unsigned char d0, d1, d2, d3;
    unsigned char c;

    d0 = (s / 1000) % 10;
    d1 = (s / 100)  % 10;
    d2 = (s / 10)   % 10;
    d3 = s % 10;

    /* clear first 40px of page 0 before writing score */
    oled_setpos(0,0);
    for (c=0;c<40;c++)
        oled_write_data(0x00);

    oled_draw_digit(0,  0, d0);
    oled_draw_digit(10, 0, d1);
    oled_draw_digit(20, 0, d2);
    oled_draw_digit(30, 0, d3);
}

/* ========== MAIN ========== */
void main(void)
{
    unsigned char c;

    /* set P2.3, P2.4, P2.5 as inputs (high) */
    P2 |= 0x38;  // bits 3,4,5 = 1
    oled_init();
    oled_clear();
    game_running = 0;
    score = 0;
    while (1) {
        /* WAIT FOR START (P2.3 LOW) */
        if (!game_running) {
            if (BTN_START == 0) {
                delay_ms(40);
                if (BTN_START == 0) {
                    game_running = 1;
                    score = 0;
                    obstacle_x = 120;
                    dino_up = 0;
                    oled_clear();
                    oled_draw_score(score);
                }
            }
            continue;
        }

        /* RESTART (P2.5 LOW) */
        if (BTN_RESTART == 0) {
```

```
            delay_ms(40);
            if (BTN_RESTART == 0) {
                game_running = 0;
                oled_clear();
                continue;
            }
        }
        /* JUMP (P2.4 LOW) */
        if (BTN_JUMP == 0 && dino_up == 0) {
            delay_ms(20);
            if (BTN_JUMP == 0) {
                dino_up = 1;
                jump_timer = 5;
            }
        }
        /* CLEAR game pages (page 5 and 6) */
        oled_setpos(0,5);
        for (c=0;c<128;c++) oled_write_data(0x00);
        oled_setpos(0,6);
        for (c=0;c<128;c++) oled_write_data(0x00);
        /* MOVE OBSTACLE */
        if (obstacle_x > 0) obstacle_x--;
        else obstacle_x = 120;
        /* HANDLE JUMP TIMER */
        if (dino_up) {
            if (jump_timer > 0) jump_timer--;
            else dino_up = 0;
        }
       /* DRAW DINO */
        if (dino_up)
            oled_draw_sprite(0, 5, dino8);   // jump row
        else
            oled_draw_sprite(0, 6, dino8);   // ground row
        /* DRAW OBSTACLE */
        oled_draw_sprite(obstacle_x, 6, cactus8);

        /* COLLISION: dino on ground and obstacle_x overlaps 0..7 */
        if (!dino_up) {
            if (obstacle_x <= 6) {
                oled_clear();
                oled_draw_score(score);   // show final score
                game_running = 0;
                continue;
            }
        }
        score++;
        if (score > 9999) score = 0;
        oled_draw_score(score);

        /* GAME SPEED */
        delay_ms(80);
    }
}
```

LCD CODE



OLED CODE



4. CONCLUSION AND FUTURE WORK

4.1 RESULT, CONCLUSION AND INFERENCE

For LCD

- Integrate sensors (temperature, humidity, etc.) to display live readings.
- Add keypad input for user interaction.
- Use I²C-based LCD interface to reduce pin usage.
- Combine with serial or wireless modules for IoT applications.

For OLED

- Add *interrupt-based button handling* instead of polling.
- Implement *menu navigation* and *dynamic graphics* on OLED.
- Interface *sensors* to display live data (temperature, light, etc.).
- Use *I²C-based OLED* to free more I/O pins.
- Introduce *EEPROM storage* for saving settings

4.2 FUTURE WORK COST

The overall cost of the project mainly depends on the components used, fabrication quality, and system integration. Since this project uses basic and reusable hardware such as the 8051 Development Kit, OLED display, and push buttons, the total cost is low to moderate for prototype development.

Estimated Component Costs (Prototype Level)

S.No	Component	Quantity	Unit Price (INR)	Total (INR)
1	8051 Development Board	1	₹600.00	₹600.00

S.No	Component	Quantity	Unit Price (INR)	Total (INR)
2	OLED Display (0.96” or 1.3”)	1	₹250.00	₹250.00
3	4-Button Module	1	₹100.00	₹100.00
4	Power Supply (5 V Adapter / USB)	1	₹150.00	₹150.00
5	Connecting Wires & Breadboard	1 set	₹100.00	₹100.00
—	Total Estimated Cost	—	—	₹1,200 – ₹1,300

Future Development Cost

If this system is scaled or converted into a final product, the cost per unit can be greatly reduced through printed circuit board (PCB) design and component optimization:

Stage	Description	Expected Cost Range (INR)
Prototype (Current)	Using development boards, jumper wires, and modules	₹1,200 – ₹1,300
Custom PCB Version	Integrated circuit with SMD components, PCB fabrication, and housing	₹400 – ₹600
Mass Production (≥100 units)	Automated assembly, bulk purchase of components	₹250 – ₹350 per unit

Remarks

Using the 8051 Dev Kit simplifies development and debugging but increases initial cost slightly.With future optimization (such as microcontroller integration on PCB and direct keypad interface), the system can become more cost-effective and commercially viable

Future Cost Estimate — bare 8051 IC + JHD204A (20×4 LCD)

1) Prototype (one-off hobby build — breadboard / perfboard)

S.No	Item	Qty	Unit price (INR)	Notes / source
1	AT89C51 / compatible 8051 DIP-40 IC	1	₹160 – ₹470	Typical retail range for AT89C51 / equivalents. Prayog India
2	JHD204A 20×4 character LCD	1	₹280 – ₹480	Common retail listings (JHD204A / 20×4 character modules). Robocraze
3	12 MHz crystal (or 11.0592 MHz)	1	₹6 – ₹30	generic component
4	33 pF capacitors	2	₹2 – ₹15	pair
5	10 μF capacitor (reset)	1	₹2 – ₹10	
6	10 kΩ resistor / pot (LCD contrast)	1	₹10 – ₹40	pot
7	100 Ω resistor (backlight)	1	₹1 – ₹5	
8	7805 regulator + filter caps (or small SMPS module)	1	₹30 – ₹150	depends on choice
9	Reset push-button, header, jumper wires, perfboard	—	₹80 – ₹200	miscellaneous
10	Programmer (one-time, to program the bare IC)	1	₹250 – ₹6,500	USBASP ~₹200–₹900 (cheap), Universal programmer (TL866 etc.) ₹5k–₹10k. Robu

Prototype total (estimate): ₹820

Lower end assumes cheap 8051 IC (₹160), cheap 20×4 LCD (₹280) and a low-cost USBASP programmer (~₹250).

Higher end includes buying a universal TL866 programmer or pricier dev components.

VIDEO LINK

5.REFERENCES

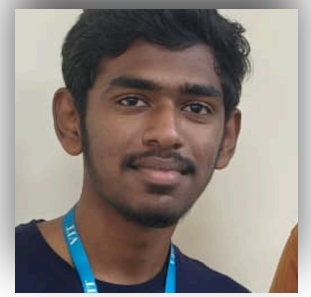
LIST OF PUBLICATIONS (If applicable)

INTERNATIONAL JOURNALS

1. [1] Atmel Corporation, AT89C51 8-bit Microcontroller with 4K Bytes Flash Datasheet, Rev. 4368E–8051–08/08, 2008. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/doc0265.pdf
2. [2] Microchip Technology Inc., 8051 Microcontroller Family User Manual. [Online]. Available: https://www.microchip.com/en-us/product/AT89C51
3. [3] JHD Display, JHD204A 20x4 Alphanumeric LCD Module Datasheet, 2019. [Online]. Available: https://www.electronicscomp.com/datasheet/jhd204a.pdf
4. [4] Texas Instruments, Interfacing Character LCD Modules with Microcontrollers, Application Note, 2017. [Online]. Available: https://www.ti.com/lit/an/snoa749/snoa749.pdf
5. [5] M. A. Mazidi, J. G. Mazidi, and R. D. McKinlay, The 8051 Microcontroller and Embedded Systems, 2nd ed. Upper Saddle River, NJ, USA: Pearson Education, 2008.
6. [6] Arduino.cc, Using Push Buttons with Microcontrollers, Learning Resources, 2023. [Online]. Available: https://docs.arduino.cc/
7. [7] Robu.in, 20x4 LCD (JHD204A) Product Specification and Interface Guide. [Online]. Available: https://robu.in/product/20x4-character-lcd-display-module/
8. [8] Electronics Hub, Interfacing 20x4 LCD with 8051 Microcontroller – Circuit and Code Explanation, 2022. [Online]. Available: https://www.electronicshub.org/interfacing-lcd-with-8051/
9. [9] TutorialsPoint, 8051 Microcontroller Architecture and Programming Basics, 2023. [Online]. Available: https://www.tutorialspoint.com/8051_microcontroller/
10. [10] Circuit Digest, 8051 Microcontroller Projects and Interfacing Tutorials, 2023. [Online]. Available: https://circuitdigest.com/microcontroller-projects/8051-projects

6.PHOTO GRAPH OF THE PROJECT ALONG WITH THE TEAM MEMBERS

BIODATA



Name :LINGA RAJA R

Mobile Number :9344394970

E-mail :lingaraja.r2024@vitstudnet.ac.in

Permanent Address: 3/168,North Street, Tharumathupatti ,Kovilpatti .

Name :MIDHUN DHASS D

Mobile Number :9363926322

E-mail :midhundhass.d2024@vitstudent.ac.in

Permanent Address: 84/5 , Vinayaka nagar, K.R Nagar P.O ,Kovilpatti .