RDL
Technologies Pvt. Ltd.

ESP32 Microcontroller Board

SNAKE GAME USING OLED

Aim

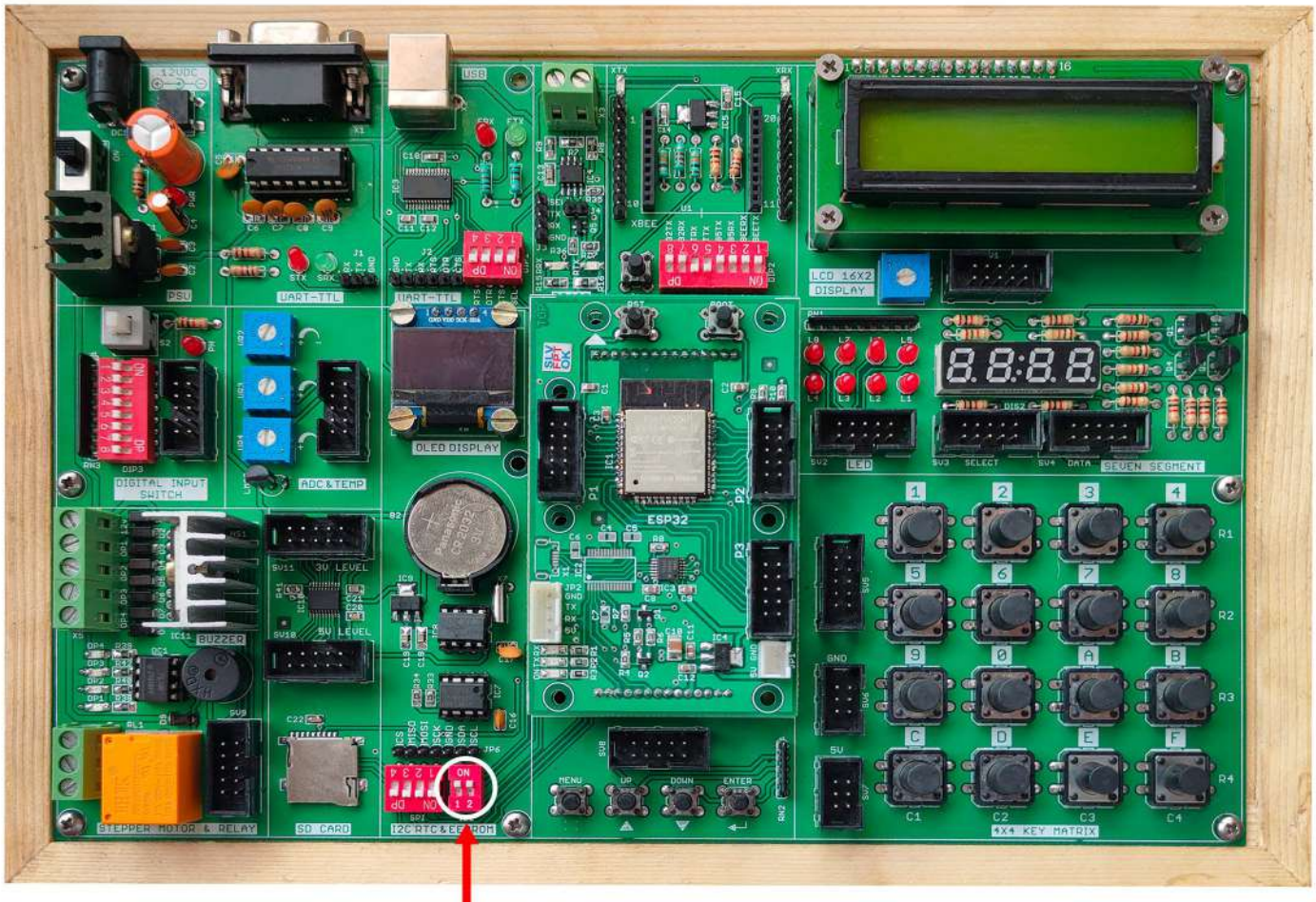
To Interface snake game using OLED Display with ESP32-Microcontroller Board.

Description

To display snake game on OLED screen.

Hardware required

ESP32-Microcontroller Development board.



Procedure:

1. Connect the USB cable to the board.
2. Open Arduino IDE .Select DOIT ESP32 DEVKIT V1 in boards and select COM port.
3. Write the program, verify and Upload it.
4. Now you can see the output displaying the message on OLED of ESP32 microcontroller board.

Program:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET 4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0X3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
const byte buttonPins[] = {27,14,13,12}; // LEFT, UP, RIGHT, DOWN
typedef enum {
 START,
 RUNNING,
 GAMEOVER
} State;
typedef enum {
 LEFT,
 UP,
 RIGHT,
 DOWN
} Direction;
```

```
#define SNAKE_PIECE_SIZE 3
#define MAX_SANKE_LENGTH 165
#define MAP_SIZE_X 20
#define MAP_SIZE_Y 20
#define STARTING_SNAKE_SIZE 2
#define SNAKE_MOVE_DELAY 20

State gameState;

int8_t snake[MAX_SANKE_LENGTH][2];
uint8_t snake_length;
Direction dir;
Direction newDir;
int8_t fruit[2];
void setup() {
 Serial.begin(9600);
 if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
 Serial.println(F("SSD1306 allocation failed"));
 for(;;);
 }
for (byte i = 0; i < 4; i++) {
 pinMode(buttonPins[i], INPUT_PULLUP);
 }
 randomSeed(analogRead(A0));
```

```
setupGame();
}

void setupGame() {
 gameState = START;
 dir = RIGHT;
 newDir = RIGHT;
 resetSnake();
 generateFruit();
 display.clearDisplay();
 drawMap();
 drawScore();
 drawPressToStart();
 display.display();
}
void resetSnake() {
 snake_length = STARTING_SNAKE_SIZE;
 for(int i = 0; i < snake_length; i++) {
 snake[i][0] = MAP_SIZE_X / 2 - i;
 snake[i][1] = MAP_SIZE_Y / 2;
 }
}
int moveTime = 0;
void loop() {
 switch(gameState) {
 case START:
 if(buttonPress()) gameState = RUNNING;
 break;
```

```
case RUNNING:
 moveTime++;
 readDirection();
 if(moveTime >= SNAKE_MOVE_DELAY) {
 dir = newDir;
 display.clearDisplay();
 if(moveSnake()) {
 gameState = GAMEOVER;
 drawGameover();
 delay(3000);
 }
drawMap();
 drawScore();
 display.display();
 checkFruit();
 moveTime = 0;
 }
 break;
case GAMEOVER:
 if(buttonPress()) {
 delay(500);
 setupGame();
 gameState = START;
 }
 break;
 }
```

```
delay(10);
}
bool buttonPress() {
 for (byte i = 0; i < 4; i++) {
 byte buttonPin = buttonPins[i];
 if (digitalRead(buttonPin) == LOW) {
 return true;
 }
 }
 return false;
}
void readDirection() {
 for (byte i = 0; i < 4; i++) {
 byte buttonPin = buttonPins[i];
 if (digitalRead(buttonPin) == LOW && i != ((int)dir + 2) % 4) {
 newDir = (Direction)i;
 return;
 }
 }
}
bool moveSnake() {
 int8_t x = snake[0][0];
 int8_t y = snake[0][1];
 switch(dir) {
 case LEFT:
 x -= 1;
```

```
break;
 case UP:
 y -= 1;
 break;
 case RIGHT:
 x += 1;
 break;
 case DOWN:
 y += 1;
 break;
 }
if(collisionCheck(x, y))
 return true;
 for(int i = snake_length - 1; i > 0; i--) {
 snake[i][0] = snake[i - 1][0];
 snake[i][1] = snake[i - 1][1];
 }
 snake[0][0] = x;
 snake[0][1] = y;
 return false;
}
void checkFruit() {
 if(fruit[0] == snake[0][0] && fruit[1] == snake[0][1])
 {
 if(snake_length + 1 <= MAX_SANKE_LENGTH)
 snake_length++;
generateFruit();
 }
}
```

```
void generateFruit() {
 bool b = false;
 do {
 b = false;
 fruit[0] = random(0, MAP_SIZE_X);
 fruit[1] = random(0, MAP_SIZE_Y);
 for(int i = 0; i < snake_length; i++) {
 if(fruit[0] == snake[i][0] && fruit[1] == snake[i][1]) {
 b = true;
 continue;
 }
 }
 } while(b);
}
bool collisionCheck(int8_t x, int8_t y) {
 for(int i = 1; i < snake_length; i++) {
 if(x == snake[i][0] && y == snake[i][1]) return true;
 }
 if(x < 0 || y < 0 || x >= MAP_SIZE_X || y >= MAP_SIZE_Y) return true;
 return false;
}
void drawMap() {
 int offsetMapX = SCREEN_WIDTH - SNAKE_PIECE_SIZE * MAP_SIZE_X - 2;
 int offsetMapY = 2;
```

```
display.drawRect(fruit[0] * SNAKE_PIECE_SIZE + offsetMapX, fruit[1] *
SNAKE_PIECE_SIZE + offsetMapY, SNAKE_PIECE_SIZE, SNAKE_PIECE_SIZE,
SSD1306_INVERSE);
 display.drawRect(offsetMapX - 2, 0, SNAKE_PIECE_SIZE * MAP_SIZE_X + 4,
SNAKE_PIECE_SIZE * MAP_SIZE_Y + 4, SSD1306_WHITE);
 for(int i = 0; i < snake_length; i++) {
 display.fillRect(snake[i][0] * SNAKE_PIECE_SIZE + offsetMapX, snake[i][1] *
SNAKE_PIECE_SIZE + offsetMapY, SNAKE_PIECE_SIZE, SNAKE_PIECE_SIZE,
SSD1306_WHITE);
 }
}

void drawScore() {
 display.setTextSize(1);
 display.setTextColor(SSD1306_WHITE);
 display.setCursor(2,2);
 display.print(F("Score:"));
 display.println(snake_length - STARTING_SNAKE_SIZE);

}

void drawPressToStart() {
 display.setTextSize(1);
 display.setTextColor(SSD1306_WHITE);
 display.setCursor(2,20);
 display.print(F("Press a\n button to\n start the\n game!"));

}
void drawGameover() {
 display.setTextSize(1);
```

```
display.setTextColor(SSD1306_WHITE);
 display.setCursor(2,50);
 display.println(F("GAMEOVER"));

}
```

Flow Chart: