# Circuit Documentation

## Summary

This circuit is designed around an Arduino Uno R3 microcontroller and includes an 8x8 LED matrix, four pushbuttons, and four green LEDs. The primary function of the circuit appears to be a simple game, likely a version of the classic "Snake" game, where the 8x8 LED matrix serves as the display. The pushbuttons are used to control the direction of the snake, and the green LEDs provide visual feedback for button presses.

## Component List

### Arduino Uno R3

* Microcontroller board based on the ATmega328P
* Digital I/O Pins: 14 (of which 6 provide PWM output)
* Analog Input Pins: 6
* Operating Voltage: 5V
* Features: USB connection, power jack, ICSP header, and a reset button

### 8x8 LED Matrix

* An 8x8 grid of LEDs, typically used for displaying characters, symbols, or custom graphics
* Requires three control signals: data input (DIN), clock (CLK), and chip select (CS)

### Pushbuttons (x4)

* Simple tactile switches that close the circuit when pressed
* Each button has two input pins and two output pins

### LEDs: Two Pin (green) (x4)

* Basic green light-emitting diodes with an anode and cathode for indicating statuses or events

## Wiring Details

### Arduino Uno R3

* 5V connected to the 8x8 LED matrix VCC
* GND connected to the cathodes of all four green LEDs, GND of the 8x8 LED matrix, and Pin 1 (in) of all four pushbuttons
* D2 connected to Pin 4 (out) of the first pushbutton
* D3 connected to Pin 4 (out) of the second pushbutton
* D4 connected to Pin 4 (out) of the third pushbutton
* D5 connected to Pin 4 (out) of the fourth pushbutton
* D7 connected to the anode of the first green LED
* D8 connected to the anode of the second green LED
* D9 connected to the anode of the third green LED
* D10 connected to the anode of the fourth green LED
* D11 connected to the CS of the 8x8 LED matrix
* D12 connected to the DIN of the 8x8 LED matrix
* D13 connected to the CLK of the 8x8 LED matrix

### 8x8 LED Matrix

* VCC connected to the 5V of the Arduino Uno R3
* GND connected to the GND of the Arduino Uno R3
* DIN connected to D12 of the Arduino Uno R3
* CLK connected to D13 of the Arduino Uno R3
* CS connected to D11 of the Arduino Uno R3

### Pushbuttons (x4)

* Pin 1 (in) connected to GND of the Arduino Uno R3
* Pin 4 (out) of each pushbutton connected to D2, D3, D4, and D5 of the Arduino Uno R3 respectively

### LEDs: Two Pin (green) (x4)

* Cathode connected to GND of the Arduino Uno R3
* Anode of each LED connected to D7, D8, D9, and D10 of the Arduino Uno R3 respectively

## Documented Code

#include <LedControl.h>  
  
// Define Snake structure  
struct Snake {  
 int head[2];  
 int body[40][2];  
 int len;  
 int dir[2];  
};  
  
struct Apple {  
 int rPos;  
 int cPos;  
};  
  
// MAX7219 LED Matrix Pins  
const int DIN = 12;  
const int CS = 11;  
const int CLK = 13;  
LedControl lc = LedControl(DIN, CLK, CS, 1);  
  
// Push Button Pins  
const int upButton = 2;  
const int downButton = 3;  
const int leftButton = 4;  
const int rightButton = 5;  
  
// LED Pins for each button  
const int ledUp = 7;  
const int ledDown = 8;  
const int ledLeft = 9;  
const int ledRight = 10;  
  
// Game Variables  
byte pic[8] = {0, 0, 0, 0, 0, 0, 0, 0};   
Snake snake = {{1, 5}, {{0, 5}, {1, 5}}, 2, {1, 0}};  
Apple apple;  
  
float oldTime = 0, timer = 0, updateRate = 3;  
  
void setup() {  
 lc.shutdown(0, false);  
 lc.setIntensity(0, 0);  
 lc.clearDisplay(0);  
  
 pinMode(upButton, INPUT\_PULLUP);  
 pinMode(downButton, INPUT\_PULLUP);  
 pinMode(leftButton, INPUT\_PULLUP);  
 pinMode(rightButton, INPUT\_PULLUP);  
  
 pinMode(ledUp, OUTPUT);  
 pinMode(ledDown, OUTPUT);  
 pinMode(ledLeft, OUTPUT);  
 pinMode(ledRight, OUTPUT);  
  
 randomSeed(analogRead(0));  
 generateApple();  
}  
  
void loop() {  
 float deltaTime = millis() - oldTime;  
 oldTime = millis();  
 timer += deltaTime;  
  
 checkInput();  
  
 if (timer > 1000 / updateRate) {  
 timer = 0;  
 updateGame();  
 }  
  
 renderGame();  
}  
  
// Function to check button input  
void checkInput() {  
 static unsigned long lastInputTime = 0;  
 if (millis() - lastInputTime < 150) return;  
  
 if (digitalRead(upButton) == LOW && snake.dir[0] == 0) {  
 snake.dir[0] = -1; snake.dir[1] = 0;  
 flashLED(ledUp);  
 }  
 if (digitalRead(downButton) == LOW && snake.dir[0] == 0) {  
 snake.dir[0] = 1; snake.dir[1] = 0;  
 flashLED(ledDown);  
 }  
 if (digitalRead(leftButton) == LOW && snake.dir[1] == 0) {  
 snake.dir[0] = 0; snake.dir[1] = -1;  
 flashLED(ledLeft);  
 }  
 if (digitalRead(rightButton) == LOW && snake.dir[1] == 0) {  
 snake.dir[0] = 0; snake.dir[1] = 1;  
 flashLED(ledRight);  
 }  
  
 lastInputTime = millis();  
}  
  
// LED Feedback Function  
void flashLED(int ledPin) {  
 digitalWrite(ledPin, HIGH);  
 delay(200);  
 digitalWrite(ledPin, LOW);  
}  
  
// Function to generate a new apple position  
void generateApple() {  
 bool validPos;  
 do {  
 apple.rPos = random(0, 8);  
 apple.cPos = random(0, 8);  
 validPos = true;  
  
 for (int i = 0; i < snake.len; i++) {  
 if (snake.body[i][0] == apple.rPos && snake.body[i][1] == apple.cPos) {  
 validPos = false;  
 break;  
 }  
 }  
 } while (!validPos);  
}  
  
// Function to update the game logic  
void updateGame() {  
 resetMatrix();  
  
 int newHead[2] = {snake.head[0] + snake.dir[0], snake.head[1] + snake.dir[1]};  
  
 // Handle border wrapping  
 if (newHead[0] == 8) newHead[0] = 0;  
 if (newHead[0] == -1) newHead[0] = 7;  
 if (newHead[1] == 8) newHead[1] = 0;  
 if (newHead[1] == -1) newHead[1] = 7;  
  
 // Check collision with itself  
 for (int j = 0; j < snake.len; j++) {  
 if (snake.body[j][0] == newHead[0] && snake.body[j][1] == newHead[1]) {  
 delay(1000);  
 snake = {{1, 5}, {{0, 5}, {1, 5}}, 2, {1, 0}};  
 generateApple();  
 return;  
 }  
 }  
  
 // Check if the snake ate the apple  
 if (newHead[0] == apple.rPos && newHead[1] == apple.cPos) {  
 snake.len++;  
 generateApple();  
 } else {  
 removeTail();  
 }  
  
 snake.body[snake.len - 1][0] = newHead[0];  
 snake.body[snake.len - 1][1] = newHead[1];  
 snake.head[0] = newHead[0];  
 snake.head[1] = newHead[1];  
  
 for (int j = 0; j < snake.len; j++) {  
 pic[snake.body[j][0]] |= 128 >> snake.body[j][1];  
 }  
 pic[apple.rPos] |= 128 >> apple.cPos;  
}  
  
// Function to render the game on the LED matrix  
void renderGame() {  
 for (int i = 0; i < 8; i++) {  
 lc.setRow(0, i, pic[i]);  
 }  
}  
  
// Functionto reset the LED matrix

void resetMatrix() {

for (int j = 0; j < 8; j++) {

pic[j] = 0;

}

}

// Function to remove the snake's tail (shift body left)

void removeTail() {

for (int j = 1; j < snake.len; j++) {

snake.body[j - 1][0] = snake.body[j][0];

snake.body[j - 1][1] = snake.body[j][1];

}

}

SUMMARY:

This Snake Game project uses an Arduino Uno, a MAX7219 LED matrix, and four LEDs for visual feedback. The game allows players to control the snake’s movement using push buttons, with each button press triggering a corresponding LED to indicate the chosen direction. The snake navigates the LED matrix, growing when it eats an apple and resetting if it collides with itself. The game also features boundary wrapping, ensuring the snake reappears on the opposite side if it crosses the screen’s edge. The LED matrix is connected to pins 12 (DIN), 11 (CS), and 13 (CLK), while the push buttons are wired to pins 2, 3, 4, and 5. The directional LEDs are connected to pins 7, 8, 9, and 10. For successful operation, players should ensure proper wiring, especially for the MAX7219 display and LEDs. If issues arise, testing individual components like LEDs and button functionality can help diagnose problems. This project offers a fun and interactive way to explore Arduino programming and LED matrix control.