Snake spielen auf dem Arduino (und dem ESP32)

Der Spieleklassiker **Snake** begleitet uns schon lange und auf ganz unterschiedlichen Geräten – auf dem PC, auf Taschenrechnern von Texas Instruments oder auch auf dem ehrwürdigen Nokia 3210. Dank des einfachen Prinzips ist das Spiel fast überall dort umzusetzen, wo etwas Prozessorleistung vorhanden ist. Also warum nicht auch auf einem Arduino?

In diesem Projekt baust du dir zunächst eine Snake-Version auf einem Arduino UNO R4. Als Display verwendest du hierbei ein **OLED-Display** mit 128×64 Pixeln. Steuern wirst du das Spiel mit einem kleinen **Joystick**. Da das Spiel zwar auf dem Arduino UNO läuft, für geschickte Spieler jedoch vielleicht etwas zu langsam ist, kommt später noch ein **ESP32** zum Zug. Dieser verfügt über mehr Leistung und sorgt für ein flüssigeres Spielvergnügen.

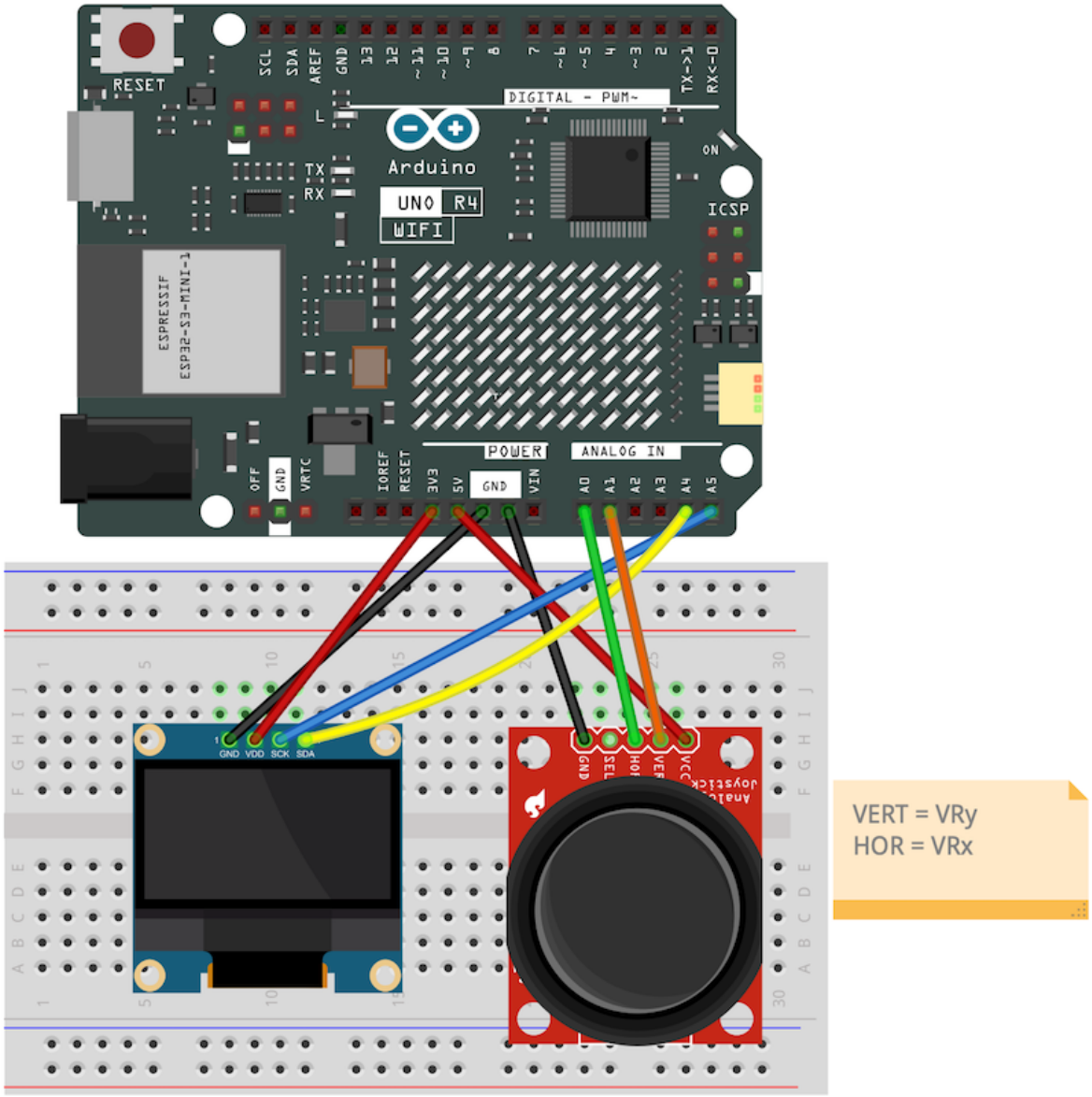
Für dieses Projekt benötigst du:

- Arduino UNO R4 und/oder ESP32*
- OLED-Display* mit 128×64 Pixeln
- Joystick*

Snake auf dem Arduino UNO

Zunächst also die Version auf dem Arduino UNO. Hier eignet sich der „neue“ UNO R4, da dieser deutlich mehr Leistung als sein Vorgänger R3 besitzt. Ob du die WiFi- oder die Minima-Version verwendest, ist für dieses Projekt egal. Falls du jedoch noch überlegst, dir einen R4 zuzulegen, empfehle ich dir auf jeden Fall erstere, da der R4 WiFi nur ein paar Euro mehr kostet und du damit mit deinen Projekten ins Internet kannst.

Nun aber zum Anschluss der Bauteile: Verbinde das OLED-Display und den den Joystick folgendermaßen mit dem Arduino UNO:



Je nachdem, welchen Joystick du verwendest, kann die Beschriftung der Pins variieren. **So kann zum Beispiel statt VERT auch VRy zu lesen sein.** Mehr Infos hierüber findest du im Tutorial zum Arduino Joystick. Wenn du alles verkabelt hast, kann es direkt mit dem Sketch weitergehen.

Die benötigten Bibliotheken

Damit du am Arduino dein OLED-Display verwenden kannst, benötigst du zwei Bibliotheken. Öffne in der Arduino IDE den Bibliotheksverwalter, suche und installiere dort diese beiden Bibliotheken von Adafruit:

Adafruit SSD1306
Adafruit GFX

Im Sketch wirst du gleich noch eine dritte Bibliothek sehen, **Wire.h** – diese ist jedoch bereits vorinstalliert, sodass du dich um diese nicht kümmern musst.

Der gesamte Sketch

Hier nun der vollständige Sketch für Snake. Kopiere dir den folgenden Code, erstelle einen neuen Sketch und lade ihn auf deinen Arduino hoch.

```
// Snake spielen auf dem Arduino
// Pollux Labs

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>

// OLED Display Konfiguration
#define SCREEN_WIDTH 128  // Breite des OLED-Displays
#define SCREEN_HEIGHT 64  // Höhe des OLED-Displays
#define OLED_RESET    -1  // OLED-Reset-Pin (nicht verwendet)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Joystick Konfiguration
#define VRX_PIN A0  // Joystick X-Achsen-Pin (analoger Pin A0 des Arduino Uno)
#define VRY_PIN A1  // Joystick Y-Achsen-Pin (analoger Pin A1 des Arduino Uno)
#define SW_PIN  7   // Joystick Taster-Pin (digitaler Pin 7 des Arduino Uno)

// Eigenschaften der Schlange
#define SNAKE_SIZE 4  // Größe jedes Segments der Schlange
#define MAX_SNAKE_LENGTH 50  // Maximale Länge der Schlange (reduziert für Arduino Uno, um Speicher zu sparen)
int snakeX[MAX_SNAKE_LENGTH], snakeY[MAX_SNAKE_LENGTH];  // Arrays zur Speicherung der Segmentpositionen der Schlange
```

```
int snakeLength = 5;  // Anfangslänge der Schlange
int directionX = 1, directionY = 0;  // Anfangsrichtung der Bewegung (nach rechts)

// Eigenschaften des Futters
int foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;  // X-Koordinate des Futters (nicht direkt am Rand)
int foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;  // Y-Koordinate des Futters (Spielbereich unterhalb des Rahmens, nicht direkt am Rand)

// Score
int score = 0;

void setup() {
  Serial.begin(9600);  // Initialisiere serielle Kommunikation mit niedrigerer Baudrate für Arduino Uno

  // Initialisiere Display
   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {   // Initialisiere das OLED-Display mit der I2C-Adresse 0x3C
    Serial.println(F("SSD1306 allocation failed"));
     for (;;);  // Stoppe, falls die Display-Initialisierung fehlschlägt
  }
  display.clearDisplay();  // Leere den Display-Puffer
  display.display();  // Zeige den geleerten Puffer an

  // Initialisiere Joystick
  pinMode(VRX_PIN, INPUT);  // Setze Joystick X-Achsen-Pin als Eingang
  pinMode(VRY_PIN, INPUT);  // Setze Joystick Y-Achsen-Pin als Eingang
  pinMode(SW_PIN, INPUT_PULLUP);  // Setze Joystick-Taster-Pin als Eingang mit internem Pull-up-Widerstand

  // Initialisiere Position der Schlange
  for (int i = 0; i < snakeLength; i++) {
    snakeX[i] = SCREEN_WIDTH / 2 - (i * SNAKE_SIZE);  // Setze
```

```
anfängliche X-Koordinaten der Schlangensegmente
    snakeY[i] = SCREEN_HEIGHT / 2;  // Setze anfängliche Y-
Koordinate der Schlangensegmente
  }
  Serial.println("Setup abgeschlossen");
}

void loop() {
  // Lese Joystick-Werte
  int xValue = analogRead(VRX_PIN);  // Lese X-Achsen-Wert vom
Joystick
  int yValue = analogRead(VRY_PIN);  // Lese Y-Achsen-Wert vom
Joystick
  Serial.print("Joystick X: ");
  Serial.print(xValue);
  Serial.print(" Y: ");
  Serial.println(yValue);

   // Setze Richtung basierend auf Joystick-Eingaben,
verhindere diagonale Bewegung
  if (xValue < 300 && directionX == 0) {  // Bewegung nach
links, wenn derzeit nicht horizontal bewegt wird
    directionX = -1;
    directionY = 0;
    Serial.println("Richtung: Links");
  } else if (xValue > 700 && directionX == 0) {  // Bewegung
nach rechts, wenn derzeit nicht horizontal bewegt wird
    directionX = 1;
    directionY = 0;
    Serial.println("Richtung: Rechts");
  } else if (yValue < 300 && directionY == 0) {  // Bewegung
nach oben, wenn derzeit nicht vertikal bewegt wird
    directionX = 0;
    directionY = -1;
    Serial.println("Richtung: Oben");
  } else if (yValue > 700 && directionY == 0) {  // Bewegung
nach unten, wenn derzeit nicht vertikal bewegt wird
    directionX = 0;
    directionY = 1;
    Serial.println("Richtung: Unten");
  }
```

```
  // Aktualisiere Position der Schlange
  for (int i = snakeLength - 1; i > 0; i--) {  // Bewege jedes
Segment zur Position des vorherigen Segments
    snakeX[i] = snakeX[i - 1];
    snakeY[i] = snakeY[i - 1];
  }
   snakeX[0] += directionX * SNAKE_SIZE;   // Aktualisiere
Kopfposition in X-Richtung
   snakeY[0] += directionY * SNAKE_SIZE;   // Aktualisiere
Kopfposition in Y-Richtung
  Serial.print("Schlangenkopf X: ");
  Serial.print(snakeX[0]);
  Serial.print(" Y: ");
  Serial.println(snakeY[0]);

  // Prüfe auf Kollision mit dem Futter
  if (snakeX[0] == foodX && snakeY[0] == foodY) {  // Wenn der
Schlangenkopf das Futter erreicht
    if (snakeLength < MAX_SNAKE_LENGTH) {
      snakeLength++;  // Erhöhe die Schlangenlänge
      score++;  // Erhöhe den Score
       Serial.println("Futter gegessen, Schlangenlänge: " +
String(snakeLength));
    }
    // Generiere neue Futterposition (nicht direkt am Rand)
     foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) /
SNAKE_SIZE) * SNAKE_SIZE;
     foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 *
SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
    Serial.print("Neues Futter bei X: ");
    Serial.print(foodX);
    Serial.print(" Y: ");
    Serial.println(foodY);
  }

  // Prüfe auf Kollision mit den Wänden
  if (snakeX[0] < SNAKE_SIZE || snakeX[0] >= SCREEN_WIDTH -
SNAKE_SIZE || snakeY[0] < 10 || snakeY[0] >= SCREEN_HEIGHT -
SNAKE_SIZE) {
     Serial.println("Kollision mit der Wand, Spiel wird
zurückgesetzt");
```

```
    resetGame();  // Setze das Spiel zurück, wenn die Schlange die Wand trifft
  }

  // Prüfe auf Kollision mit sich selbst
  for (int i = 1; i < snakeLength; i++) {
    if (snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i]) { // Wenn der Schlangenkopf mit ihrem eigenen Körper kollidiert
      Serial.println("Kollision mit sich selbst, Spiel wird zurückgesetzt");
      resetGame();  // Setze das Spiel zurück
    }
  }

  // Zeichne alles
  display.clearDisplay();  // Leere den Display-Puffer

  // Zeichne den Score
  display.setTextSize(1);  // Setze Textgröße
  display.setTextColor(SSD1306_WHITE);  // Setze Textfarbe
  display.setCursor(0, 0);  // Setze Cursor für Score
  display.print("Score: ");
  display.print(score);  // Zeige aktuellen Score an

  // Zeichne den Rand um das Spielfeld
  display.drawRect(0, 10, SCREEN_WIDTH, SCREEN_HEIGHT - 10, SSD1306_WHITE);  // Zeichne einen weißen Rahmen um das Spielfeld

  // Zeichne die Schlange
  for (int i = 0; i < snakeLength; i++) {
    display.fillRect(snakeX[i], snakeY[i], SNAKE_SIZE, SNAKE_SIZE, SSD1306_WHITE);  // Zeichne jedes Segment der Schlange
  }

  // Zeichne das Futter
  display.fillRect(foodX, foodY, SNAKE_SIZE, SNAKE_SIZE, SSD1306_WHITE);  // Zeichne das Futter

  display.display();  // Zeige den aktualisierten Puffer an
```

```
   delay(150);   // Verzögerung zur Steuerung der Geschwindigkeit der Schlange
}

void resetGame() {
  // Setze Eigenschaften der Schlange zurück
  snakeLength = 5;  // Setze die Schlangenlänge zurück
  directionX = 1;  // Setze die Richtung nach rechts zurück
  directionY = 0;
  score = 0;  // Setze den Score zurück
  for (int i = 0; i < snakeLength; i++) {
    snakeX[i] = SCREEN_WIDTH / 2 - (i * SNAKE_SIZE);  // Setze die Position der Schlange zurück auf die Mitte
    snakeY[i] = SCREEN_HEIGHT / 2;
  }
  // Generiere neue Futterposition (nicht direkt am Rand)
   foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
   foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
  Serial.println("Spiel zurückgesetzt");
  Serial.print("Neues Futter bei X: ");
  Serial.print(foodX);
  Serial.print(" Y: ");
  Serial.println(foodY);
}
```

Nach dem Upload sollte es auf dem Display direkt losgehen: In der Mitte erscheint, die zunächst 5 Pixel lange Schlange und bewegt sich. Ein Stück Futter erscheint zufällig platziert auf dem Spielfeld. Wie du sicherlich weißt, ist dein Ziel, so viel zu fressen wie möglich – ohne mit dem Spielfeldrand oder dir selbst zu kollidieren.

Oben links findest du deine Punktzahl. Für jede Verlängerung deiner Schlange erhöht sich dein Score um 1.

So funktioniert der Sketch

Lass uns ein paar wichtige Stellen des Sketchs anschauen.

Setup

Die setup()-Funktion führt alle notwendigen Initialisierungen durch:

- Serielle Kommunikation wird gestartet, um Debugging-Informationen auszugeben.
- Das OLED-Display wird initialisiert. Falls die Initialisierung fehlschlägt, wird eine Fehlermeldung ausgegeben und das Programm angehalten.

```
void setup() {
  Serial.begin(9600);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  display.clearDisplay();
  display.display();
}
```

- Der Joystick wird initialisiert, indem seine Pins als Eingänge konfiguriert werden.
- Die Position der Schlange wird initial in der Mitte des Displays gesetzt.

Loop

Die loop()-Funktion läuft kontinuierlich und behandelt alle Aspekte des Spiels, wie die Steuerung, Bewegung, Kollisionserkennung und die Ausgabe auf dem Display.

Joystick-Steuerung

Der Joystick wird verwendet, um die Bewegungsrichtung der Schlange zu ändern. Die analogen Eingänge des Joysticks liefern Werte, die bestimmen, in welche Richtung sich die Schlange bewegt:

- Wenn der Joystick nach links bewegt wird, wird die Richtung auf links gesetzt, solange die Schlange sich nicht bereits nach rechts bewegt.
- Gleiches gilt für die anderen Richtungen.

```
if (xValue < 300 && directionX == 0) {
  directionX = -1;
  directionY = 0;
  Serial.println("Richtung: Links");
}
```

Bewegung der Schlange

Die Position der Schlange wird durch eine Schleife aktualisiert, in der jedes Segment der Schlange zur Position des vorherigen Segments bewegt wird. Der Kopf der Schlange wird in die gewählte Richtung verschoben.

```
for (int i = snakeLength - 1; i > 0; i--) {
  snakeX[i] = snakeX[i - 1];
  snakeY[i] = snakeY[i - 1];
}
snakeX[0] += directionX * SNAKE_SIZE;
snakeY[0] += directionY * SNAKE_SIZE;
```

Fressen des Futters

Wenn die Schlange das Futter erreicht, wird ihre Länge erhöht, und ein neuer Punktestand wird berechnet. Danach wird das

Futter an einer neuen Position generiert, die nicht direkt am Rand liegt, um das Spiel einfacher zu machen.

```
if (snakeX[0] == foodX && snakeY[0] == foodY) {
  if (snakeLength < MAX_SNAKE_LENGTH) {
    snakeLength++;
    score++;
  }
   foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
   foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
}
```

Kollisionserkennung

Die loop()-Funktion prüft auch, ob die Schlange mit den Rändern des Spielfeldes oder mit sich selbst kollidiert:

- Wenn die Schlange mit einer Wand kollidiert, wird das Spiel zurückgesetzt.
- Wenn die Schlange ihren eigenen Körper berührt, wird ebenfalls das Spiel zurückgesetzt.

```
if (snakeX[0] < SNAKE_SIZE || snakeX[0] >= SCREEN_WIDTH - SNAKE_SIZE || snakeY[0] < 10 || snakeY[0] >= SCREEN_HEIGHT - SNAKE_SIZE) {
  resetGame();
}
```

Anzeige auf dem OLED-Display

Das OLED-Display wird in jeder Schleife aktualisiert:

- Der Punktestand wird oben links angezeigt.

- Ein weißer Rahmen wird um das Spielfeld gezeichnet, um die Grenzen des Spielfeldes anzuzeigen.
- Die Schlange und das Futter werden auf dem Display gezeichnet.

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.print("Score: ");
display.print(score);

display.drawRect(0, 10, SCREEN_WIDTH, SCREEN_HEIGHT - 10, SSD1306_WHITE);

for (int i = 0; i < snakeLength; i++) {
    display.fillRect(snakeX[i], snakeY[i], SNAKE_SIZE, SNAKE_SIZE, SSD1306_WHITE);
}
display.fillRect(foodX, foodY, SNAKE_SIZE, SNAKE_SIZE, SSD1306_WHITE);
display.display();
```

Das Spiel zurücksetzen

Die resetGame()-Funktion setzt die Schlange zurück auf ihre Ausgangsposition und -länge, wenn eine Kollision erkannt wird. Auch der Punktestand wird zurückgesetzt, und eine neue Position für das Futter wird generiert.

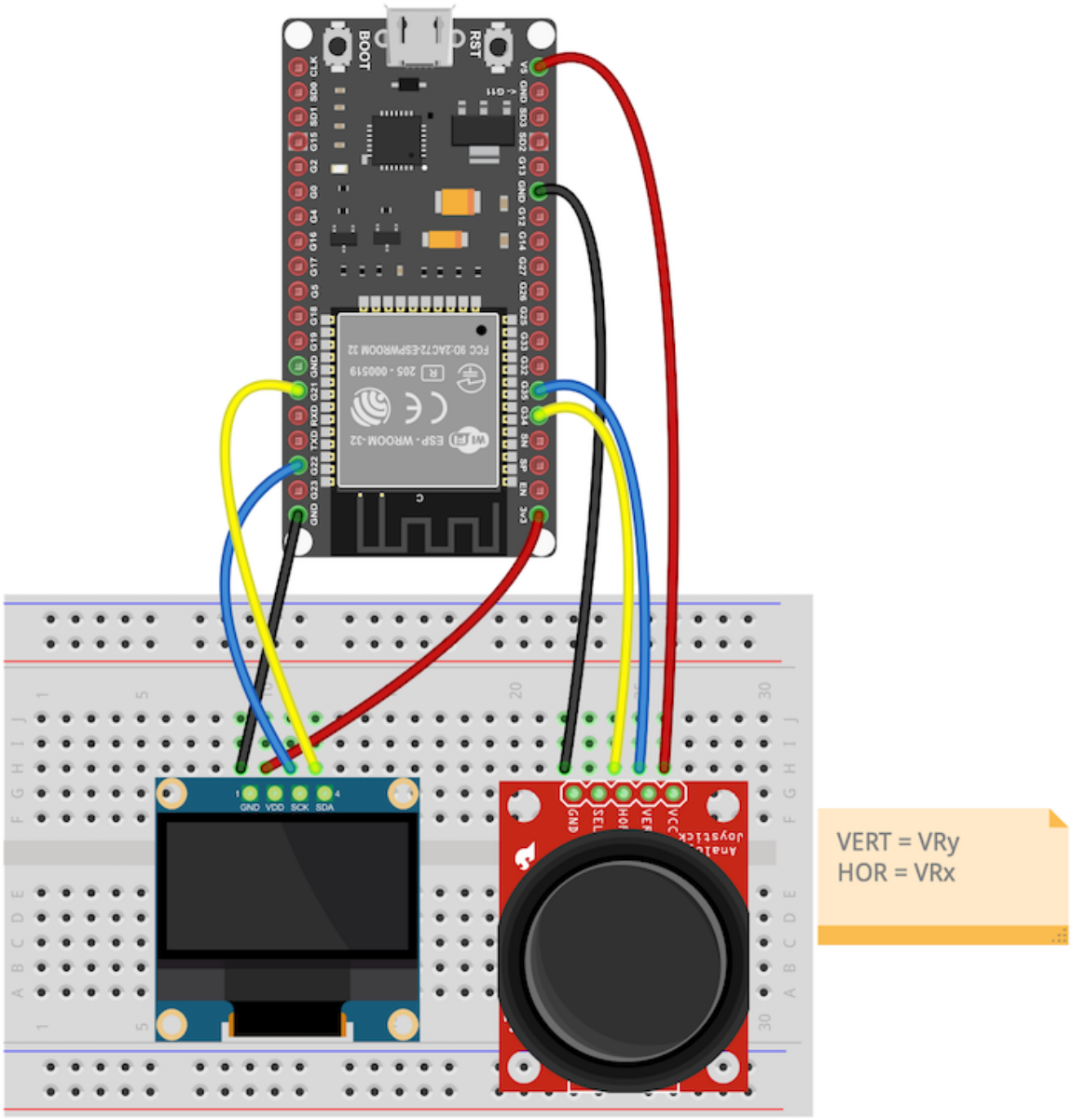
```
void resetGame() {
  snakeLength = 5;
  directionX = 1;
  directionY = 0;
  score = 0;
  for (int i = 0; i < snakeLength; i++) {
    snakeX[i] = SCREEN_WIDTH / 2 - (i * SNAKE_SIZE);
    snakeY[i] = SCREEN_HEIGHT / 2;
```

```
  }
}
```

Snake auf dem ESP32 spielen

Wenn du an dieser Stelle bereits Snake auf dem Arduino UNO gespielt hast, wirst du sicherlich bemerkt haben, dass das Spiel sehr gemächlich – um nicht zu sagen „ruckelig“ – abläuft. Das liegt an der eher begrenzten Leistung, die auch beim Modell R4 nicht ausreichend ist, alle Berechnungen in der Loop-Funktion schnell genug durchzuführen.

Falls du jedoch auch einen ESP32 dein Eigen nennst, kannst du das Spiel (fast) ganz einfach umziehen und ihm so mehr Geschwindigkeit einhauchen. Schließe das OLED-Display und den Joystick folgendermaßen am ESP32 an:



Der Sketch

Damit Snake auf dem ESP32 läuft, sind nur ein paar kleinere Adaptionen nötig. So hinterlegst du im Sketch zunächst natürlich andere Pins. Aber auch die maximale Länge der Schlange kannst du hier nach oben setzen – von 50 für den Arduino auf 100 für den ESP32:

```
#define MAX_SNAKE_LENGTH 100  // Maximale Länge der Schlange
```

Auch die Schwellenwerte für den Joystick musst du beim ESP32 ändern, weil der dieser eine höhere Auflösung für die analogen Eingänge hat als der Arduino UNO. Der Arduino Uno verwendet eine 10-Bit-Auflösung, was bedeutet, dass die analogen Eingänge Werte von 0 bis 1023 zurückgeben. Der ESP32 hingegen verwendet eine 12-Bit-Auflösung, was einen Bereich von 0 bis 4095 ergibt.

Für eine Bewegung nach links ist das zum Beispiel ein Wert von 1000. Im Sketch für den Arduino UNO lag dieser Wert bei 300:

```
if (xValue < 1000 && directionX == 0) { //Schwellenwert für den ESP32
```

Mit einem Delay steuerst du die Geschwindigkeit der Schlange. Da dir auf dem ESP32 mehr Leistung zur Verfügung steht, setzt du diesen Wert hier in Zeile 147 des Sketchs auf 100 (statt 150 für den Arduino UNO).

Hier nun der gesamte Sketch für den ESP32:

```
// Snake spielen auf dem ESP32
// Pollux Labs

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>

// OLED Display Konfiguration
#define SCREEN_WIDTH 128  // Breite des OLED-Displays
#define SCREEN_HEIGHT 64  // Höhe des OLED-Displays
#define OLED_RESET    -1  // OLED-Reset-Pin (nicht verwendet)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Joystick Konfiguration
#define VRX_PIN 34  // Joystick X-Achsen-Pin
```

```
#define VRY_PIN 35  // Joystick Y-Achsen-Pin
#define SW_PIN  32  // Joystick Taster-Pin

// Eigenschaften der Schlange
#define SNAKE_SIZE 4  // Größe jedes Segments der Schlange
#define MAX_SNAKE_LENGTH 100  // Maximale Länge der Schlange
int snakeX[MAX_SNAKE_LENGTH], snakeY[MAX_SNAKE_LENGTH];  // Arrays zur Speicherung der Segmentpositionen der Schlange
int snakeLength = 5;  // Anfangslänge der Schlange
int directionX = 1, directionY = 0;  // Anfangsrichtung der Bewegung (nach rechts)

// Eigenschaften des Futters
int foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;  // X-Koordinate des Futters (nicht direkt am Rand)
int foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;  // Y-Koordinate des Futters (Spielbereich unterhalb des Rahmens, nicht direkt am Rand)

// Score
int score = 0;

void setup() {
   Serial.begin(115200);   // Initialisiere serielle Kommunikation

  // Initialisiere Display
   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {   // Initialisiere das OLED-Display mit der I2C-Adresse 0x3C
    Serial.println(F("SSD1306 allocation failed"));
     for (;;);  // Stoppe, falls die Display-Initialisierung fehlschlägt
  }
  display.clearDisplay();  // Leere den Display-Puffer
  display.display();  // Zeige den geleerten Puffer an

  // Initialisiere Joystick
  pinMode(VRX_PIN, INPUT);  // Setze Joystick X-Achsen-Pin als Eingang
```

```
  pinMode(VRY_PIN, INPUT);  // Setze Joystick Y-Achsen-Pin als Eingang
  pinMode(SW_PIN, INPUT_PULLUP);  // Setze Joystick-Taster-Pin als Eingang mit internem Pull-up-Widerstand

  // Initialisiere Position der Schlange
  for (int i = 0; i < snakeLength; i++) {
    snakeX[i] = SCREEN_WIDTH / 2 - (i * SNAKE_SIZE);  // Setze anfängliche X-Koordinaten der Schlangensegmente
    snakeY[i] = SCREEN_HEIGHT / 2;  // Setze anfängliche Y-Koordinate der Schlangensegmente
  }
  Serial.println("Setup abgeschlossen");
}

void loop() {
  // Lese Joystick-Werte
  int xValue = analogRead(VRX_PIN);  // Lese X-Achsen-Wert vom Joystick
  int yValue = analogRead(VRY_PIN);  // Lese Y-Achsen-Wert vom Joystick
  Serial.print("Joystick X: ");
  Serial.print(xValue);
  Serial.print(" Y: ");
  Serial.println(yValue);

   // Setze Richtung basierend auf Joystick-Eingaben, verhindere diagonale Bewegung
  if (xValue < 1000 && directionX == 0) {  // Bewegung nach links, wenn derzeit nicht horizontal bewegt wird
    directionX = -1;
    directionY = 0;
    Serial.println("Richtung: Links");
  } else if (xValue > 3000 && directionX == 0) {  // Bewegung nach rechts, wenn derzeit nicht horizontal bewegt wird
    directionX = 1;
    directionY = 0;
    Serial.println("Richtung: Rechts");
  } else if (yValue < 1000 && directionY == 0) {  // Bewegung nach oben, wenn derzeit nicht vertikal bewegt wird
    directionX = 0;
```

```
    directionY = -1;
    Serial.println("Richtung: Oben");
  } else if (yValue > 3000 && directionY == 0) {  // Bewegung
nach unten, wenn derzeit nicht vertikal bewegt wird
    directionX = 0;
    directionY = 1;
    Serial.println("Richtung: Unten");
  }

  // Aktualisiere Position der Schlange
  for (int i = snakeLength - 1; i > 0; i--) {  // Bewege jedes
Segment zur Position des vorherigen Segments
    snakeX[i] = snakeX[i - 1];
    snakeY[i] = snakeY[i - 1];
  }
   snakeX[0] += directionX * SNAKE_SIZE;  // Aktualisiere
Kopfposition in X-Richtung
   snakeY[0] += directionY * SNAKE_SIZE;  // Aktualisiere
Kopfposition in Y-Richtung
  Serial.print("Schlangenkopf X: ");
  Serial.print(snakeX[0]);
  Serial.print(" Y: ");
  Serial.println(snakeY[0]);

  // Prüfe auf Kollision mit dem Futter
  if (snakeX[0] == foodX && snakeY[0] == foodY) {  // Wenn der
Schlangenkopf das Futter erreicht
    if (snakeLength < MAX_SNAKE_LENGTH) {
      snakeLength++;  // Erhöhe die Schlangenlänge
      score++;  // Erhöhe den Score
       Serial.println("Futter gegessen, Schlangenlänge: " +
String(snakeLength));
    }
    // Generiere neue Futterposition (nicht direkt am Rand)
     foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) /
SNAKE_SIZE) * SNAKE_SIZE;
     foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 *
SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
    Serial.print("Neues Futter bei X: ");
    Serial.print(foodX);
    Serial.print(" Y: ");
```

```
    Serial.println(foodY);
  }

  // Prüfe auf Kollision mit den Wänden
  if (snakeX[0] < SNAKE_SIZE || snakeX[0] >= SCREEN_WIDTH - SNAKE_SIZE || snakeY[0] < 10 || snakeY[0] >= SCREEN_HEIGHT - SNAKE_SIZE) {
     Serial.println("Kollision mit der Wand, Spiel wird zurückgesetzt");
    resetGame();  // Setze das Spiel zurück, wenn die Schlange die Wand trifft
  }

  // Prüfe auf Kollision mit sich selbst
  for (int i = 1; i < snakeLength; i++) {
    if (snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i]) { // Wenn der Schlangenkopf mit ihrem eigenen Körper kollidiert
       Serial.println("Kollision mit sich selbst, Spiel wird zurückgesetzt");
      resetGame();  // Setze das Spiel zurück
    }
  }

  // Zeichne alles
  display.clearDisplay();  // Leere den Display-Puffer

  // Zeichne den Score
  display.setTextSize(1);  // Setze Textgröße
  display.setTextColor(SSD1306_WHITE);  // Setze Textfarbe
  display.setCursor(0, 0);  // Setze Cursor für Score
  display.print("Score: ");
  display.print(score);  // Zeige aktuellen Score an

  // Zeichne den Rand um das Spielfeld
  display.drawRect(0, 10, SCREEN_WIDTH, SCREEN_HEIGHT - 10, SSD1306_WHITE);  // Zeichne einen weißen Rahmen um das Spielfeld

  // Zeichne die Schlange
  for (int i = 0; i < snakeLength; i++) {
      display.fillRect(snakeX[i], snakeY[i], SNAKE_SIZE,
```

```
SNAKE_SIZE, SSD1306_WHITE);  // Zeichne jedes Segment der Schlange
  }

  // Zeichne das Futter
   display.fillRect(foodX, foodY, SNAKE_SIZE, SNAKE_SIZE, SSD1306_WHITE);  // Zeichne das Futter

  display.display();  // Zeige den aktualisierten Puffer an

   delay(100);   // Verzögerung zur Steuerung der Geschwindigkeit der Schlange
}

void resetGame() {
  // Setze Eigenschaften der Schlange zurück
  snakeLength = 5;  // Setze die Schlangenlänge zurück
  directionX = 1;  // Setze die Richtung nach rechts zurück
  directionY = 0;
  score = 0;  // Setze den Score zurück
  for (int i = 0; i < snakeLength; i++) {
    snakeX[i] = SCREEN_WIDTH / 2 - (i * SNAKE_SIZE);  // Setze die Position der Schlange zurück auf die Mitte
    snakeY[i] = SCREEN_HEIGHT / 2;
  }
  // Generiere neue Futterposition (nicht direkt am Rand)
   foodX = random(2, (SCREEN_WIDTH - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
   foodY = random(12 / SNAKE_SIZE, (SCREEN_HEIGHT - 2 * SNAKE_SIZE) / SNAKE_SIZE) * SNAKE_SIZE;
  Serial.println("Spiel zurückgesetzt");
  Serial.print("Neues Futter bei X: ");
  Serial.print(foodX);
  Serial.print(" Y: ");
  Serial.println(foodY);
}
```

Wie geht es weiter?

Du hast nun eine spielbare Version von Snake samt einer Anzeige deiner Punkte. Wie könntest du das Spiel noch erweitern oder verbessern? Eine Idee könnte zum Beispiel ein Highscore sein. Der aktuelle Halter dieses Highscores könnte mit dem Joystick seine Initialen eintragen, die neben der Punktezahl im Dateisystem des ESP32 gespeichert werden und auf dem Display angezeigt werden – ein guter Ansporn für eine weitere Partie Snake!