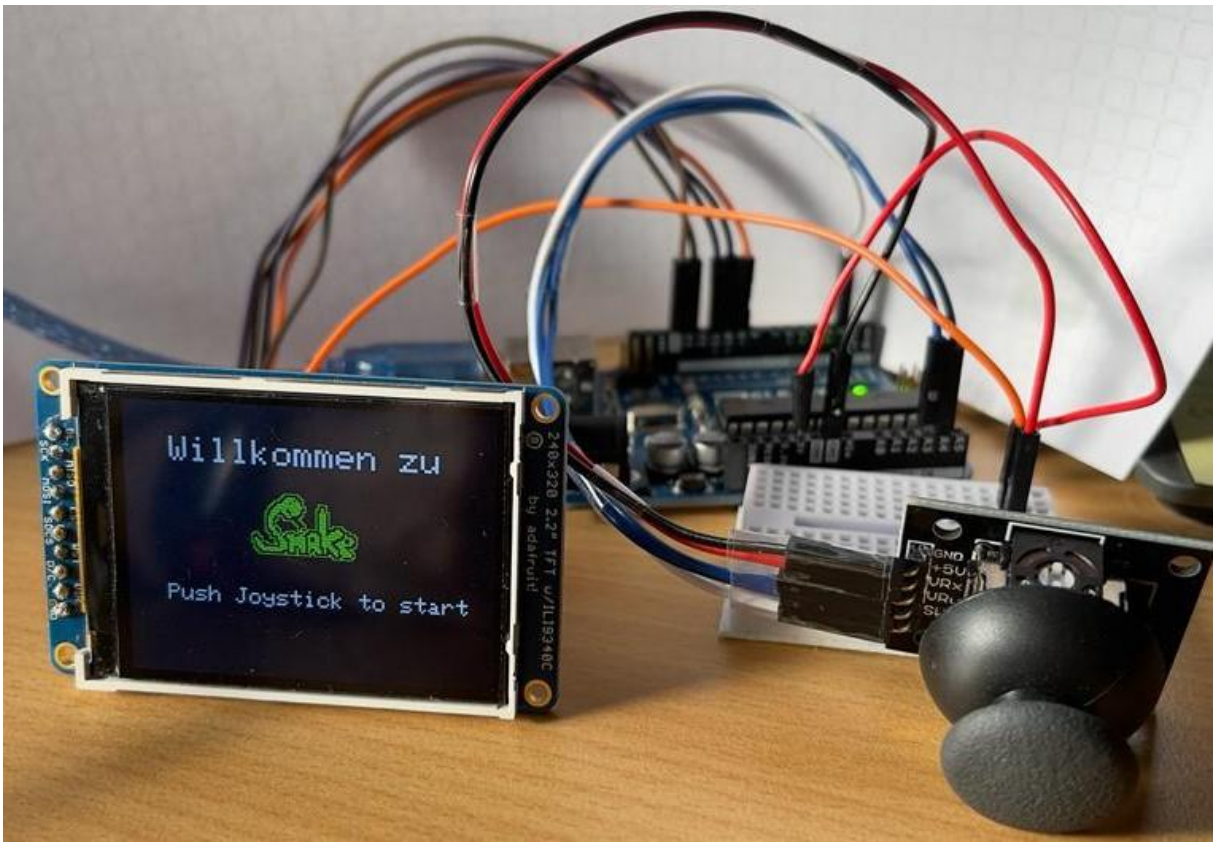
Arduino Snake Game

Dokumentation



Inhaltsverzeichnis

Projekteinleitung

Diese Dokumentation beschreibt die Prozesse, die benötigt wurden, um das Spiel Snake mit dem Einplatinencomputer Arduino Uno zu erstellen.

Snake ist ein Computerspielklassiker, bei dem eine sich gerade oder rechtwinklig bewegende Schlange durch ein Spielfeld gesteuert wird. Ziel des Spieles ist, die als Futter angebotenen zufällig erscheinenden „Happen“ aufzunehmen und Hindernissen, einschließlich des eigenen Schlangenkörpers, auszuweichen.

Ein 2,2“ Display wird als Anzeige verwendet, um die Schlange in einem definierten Spielfeld laufen zu lassen. Mit Hilfe eines Joysticks wird die Bewegung der Schlange kontrolliert und das Spiel gestartet. Ein Score wird über dem Spielfeld angezeigt.

Verwendete Hardware und Software

Diese Hardware wird für das Projekt benötigt:

Arduino Uno Rev. 3	https://store.arduino.cc/products/arduino-uno-rev3
Adafruit 2.2“ TFT LCD-Display	https://www.exp-tech.de/displays/tft/9192/adafruit-2.2-18-bit-color-tft-lcd-display-with-microsd-card-breakout-ili9340
Barrel Jack Power Switch	https://www.exp-tech.de/en/accessories/buttonsswitches/4991/barrel-jack-power-switch-m-f-7-62cm
5 Wege Joystick	https://www.marotronics.de/JoyStick-Dual-Achse-Modul-PS2-Spielsteuerung-Game-Controller-fuer-Arduino-Pi
Jumper Kabel	https://www.amazon.de/Female-Female-Male-Female-Male-Male-Steckbrücken-Drahtbrücken-bunt/dp/B01EV70C78
9V Batterie	https://www.conrad.de/de/p/varta-longlife-9v-bli-1-9-v-block-batterie-alkali-mangan-565-mah-9-v-1-st-650709.html
Connector	https://www.christians-shop.de/9V-Batterie-Power-Stecker-10cm-Kabel-perfekt-fuer-Arduino

Anstatt der 9V Batterie und des Connectors kann auch das USB-Kabel welches zum verbinden des Arduino mit einem Computer verwendet wird, verwendet werden, um zum Beispiel mit einer Powerbank nur mit Strom versorgt zu werden.

Die Verbindung des Displays mit den benötigten Kabeln muss gelötet werden. Dazu werden ein Lötkolben und Lötzinn benötigt.

Lötkolben mit Lötzinn	https://www.conrad.de/de/p/basetech-zd-30b-loetkolben-set-230-v-30-w-bleistiftform-inkl-loetzinn-inkl-ablage-inkl-entloetsaugpumpe-inkl-loetspi-2280536.html

Zum Schreiben des Codes und verwalten der Bibliotheken wird die Entwicklungsumgebung Arduino IDE benötigt.

Arduino IDE	https://www.arduino.cc/en/software

Quellcode

Include

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include "bitmaps.h"
```

Das Display wird per Serial Peripheral Interface Schnittstelle (SPI) mit Hilfe der Bibliothek „SPI.h“ angesteuert, die über die Arduino IDE heruntergeladen und installiert werden kann. Sie ist für die Kommunikation zwischen Arduino und Display verantwortlich.

Um grundlegende Funktionen wie zum Beispiel das Zeichnen eines Kreises zu unterstützen wird die Bibliothek "Adafruit_GFX.h" verwendet. Für erweiterte Funktionen kommt die hardwarespezifische Bibliothek "Adafruit_ILI9341.h" zum Einsatz.

Es werden zwei Bitmaps in diesem Projekt verwendet. Für die Benutzung wird die Datei „bitmaps.h“ eingebunden, um die zwei in der Datei „bitmaps.cpp“ enthaltenen Bitmaps für den Start- und Verlierer-Screen aufrufen zu können. Eine Bitmap ist eine Art Grafik, die anhand von einzelnen Bildpunkten in einem Raster dargestellt wird.

Define

```
#define TFT_DC 9
#define TFT_CS 10

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

#define Joystick_X A4
#define Joystick_Y A3
#define Joystick_Button 2
#define maximum_snake_lenght 300
#define spielfeld_breite 31
#define spielfeld_hoehe 20

#define p2s_x(p_x) (10*p_x+5)
#define p2s_y(p_y) (10*p_y+35)
```

Erläuterung:

Ein Define wird genutzt, um einen Wert fest zu schreiben. Dieser kann im Programm nicht mehr geändert werden. Sie können auch als kleine Funktion genutzt werden.

Fest gelegt werden unter anderem eine feste Pin-Belegung für den Joystick (A4, A3, 2) und das Display (9, 10).

Um das Display im folgenden Code mit tft. Ansprechen zu können muss es mit zwei Pins initialisiert werden. Diese Pins werden hier festgelegt.

Die Variable `maximum_snake_lenght` ist auf 300 byte begrenzt da der Programmspeicher des Arduino limitiert.

Zuletzt werden zwei Defines benutzt, mit denen zwei feste Funktionen geschrieben werden, um feste Umrechnung in Spiel Koordinaten zu erhalten. Diese Funktionen können überall im Programm aufgerufen werden.

Variablen Initialisierungen

```
int JoystickX;
int JoystickY;

int score = 0;
int highscore=0;
int game_status = 0;

int CursorX;
int CursorY;

int dx=1,dy=0;

byte blocked[spielfeld_breite][spielfeld_hoehe];
byte position_x[maximum_snake_lenght];
byte position_y[maximum_snake_lenght];

int head = 0;
int tail = 0;
int counter = 0;
int random_food_x;
int random_food_y;
int random_generator_for_food_x;
int random_generator_for_food_y;
```

Variablen müssen angelegt werden, um Speicherplatz zu reservieren, um darin später Werte speichern zu können. Neben den anderen angelegten Variablen werden drei Arrays angelegt. Das Array „`blocked`“, „`position_x`“ und „`position_y`“.

Funktionen

Erläuterung:

Eine Funktion beinhaltet Programmcode, der in einem Programm mehrmals aufgerufen werden kann. Sie wird benutzt, um das Programm übersichtlicher zu gestalten. Ein weiterer Vorteil ist, dass sollte in dieser Funktion etwas verändert werden müssen, muss dies nur einmal getan werden und nicht in dem gesamten Code in dem Programmteile ohne Funktion mehrfach benutzt werden würden.

Welcome_screen

```
void welcome_screen()
{
  tft.fillScreen(ILI9341_BLACK);
  tft.setTextColor(ILI9341_WHITE); // Farbe des Textes setzen
  tft.setTextSize(3); // Größe des Textes
  tft.setCursor(40, 30); // Koordinaten setzen wo die Schrift anfängt
  tft.print("Willkommen zu"); // Text auf das Display schreiben
  tft.drawBitmap(110, 75, welcome_snake, 92, 56, ILI9341_GREEN);
  tft.setCursor(35, 160); // Koordinaten setzen wo die Schrift anfängt
  tft.setTextSize(2);
  tft.print("Push Joystick to start"); // Text auf das Display schreiben
}
```

Zuerst wird das Display schwarz gezeichnet. Die Standardfarbe ist weiß. Die Schriftfarbe wird auf weiß gestellt. Bevor Text auf das Display geschrieben werden kann, muss der Cursor an die richtigen Koordinaten gesetzt werden. Wird der Cursor nicht gesetzt wird bei den Koordinaten X:0 und Y:0 gezeichnet. Anschließend kann der Text „Willkommen zu“ auf den Bildschirm gezeichnet werden. Mit dem Befehl `tft.drawBitmap` wird die erste der beiden Bitmaps auf dem Display abgebildet. Abschließend wird der Cursor an eine neue Stelle gesetzt, die Größe des Textes gesetzt und erneut Text ausgegeben.

```
tft.drawBitmap(110, 75, welcome_snake, 92, 56, ILI9341_GREEN);
```

110	X Koordinate
75	Y Koordinate
welcome_snake	Name der Bitmap in der eingefügten Datei
92	Größe nach rechts ab der X Koordinate
56	Größe nach unten ab der Y Koordinate
ILI9341_GREEN	Farbe der Bitmap

Game_screen

```
void game_screen()
{
  tft.fillScreen(ILI9341_BLACK);
  tft.drawRect(0, 30, 320, 210, ILI9341_PURPLE);
  tft.drawRect(1, 31, 318, 208, ILI9341_PURPLE);
  tft.drawRect(2, 32, 316, 206, ILI9341_PURPLE);
  tft.drawRect(3, 33, 314, 204, ILI9341_PURPLE);
  tft.drawRect(4, 34, 312, 202, ILI9341_PURPLE);

  tft.setTextColor(ILI9341_RED);
  tft.setCursor(65, 4);
  tft.print("Score:");
  tft.print(score);
}
```

In dieser Funktion wird das Spielfeld, in dem die Schlange sich bewegen darf, gezeichnet. Die Spielfeldumrandung ist an jeder Seite fünf Pixel breit/hoch. Die Spielfeldumrandung fängt erst 30 Pixel unterhalb des oberen Bildschirmrandes an um über dem Spielfeld den aktuellen Punktestand ausgegeben zu können. Für die Ausgabe wird der Cursor neu gesetzt und der Text „Score“ mit dem aktuellem Punktestand ausgegeben.

```
tft.drawRect(0, 30, 320, 210, ILI9341_PURPLE);
```

0	X Koordinate
30	Y Koordinate
320	Größe nach rechts ab der X Koordinate
210	Größe nach unten ab der Y Koordinate
ILI9341_PURPLE	Farbe der Bitmap

Loser_screen

```
void loser_screen()
{
  tft.fillScreen(ILI9341_BLACK);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(3);
  tft.setCursor(50, 10);
  tft.print("! YOU LOST !");
  tft.drawBitmap(100, 55, dead_snake, 124, 38, ILI9341_RED);
  tft.setCursor(45, 110);
  tft.setTextSize(3);
  tft.print("Learn to play");
  tft.setCursor(65, 140);
  tft.setTextColor(ILI9341_RED);
  tft.print("Score:");
  tft.print(score);
  tft.setCursor(40, 170);
  tft.print("Highscore:");
  tft.print(highscore);
  tft.setTextSize(2);
  tft.setTextColor(ILI9341_WHITE);
  tft.setCursor(12, 210);
  tft.print("Play again? Push Joystick");
}
```

Die Funktion „`looser_screen`“ wird aufgerufen, sobald das Spiel verloren ist.

Es werden unter anderem der Text „You Lost“, die zweite Bitmap einer toten Schlange und erreichte Score sowie der Highscore angezeigt.

Food_generator

```
void food_generator()
{
    random_generator_for_food_x = random(0,31);
    random_generator_for_food_y = random(0,20);

    random_food_x = p2s_x(random_generator_for_food_x);
    random_food_y = p2s_y(random_generator_for_food_y);
}
```

In dieser Funktion werden die X- und Y-Koordinate für das Essen der Schlange (Essen wird als rotes Rechteck dargestellt) berechnet. Zuerst wird mit Hilfe der integrierten Funktion `random` jeweils eine zufällig ausgesuchte Zahl zwischen 0 und 31 für die X-Koordinate und zwischen 0 und 20 für die Y-Koordinate festgelegt. Diese zwei Koordinaten können so noch nicht verwendet werden, sie müssen zuerst mit Hilfe der zwei zuvor erstellten define Funktionen `p2x_x` und `p2y_y` verarbeitet werden:

```
#define p2s_x(p_x) (10*p_x+5)
```

Die zufällig generierte Zahl wird mit zehn multipliziert, um ein Ergebnis zu erhalten das immer durch fünf teilbar ist. Um zu verhindern, dass der Apfel innerhalb der Spielfeldbegrenzung gezeichnet wird, an der die Schlange sterben würde, wird die errechnete Zahl mit fünf, aufgrund einer Breite von 5 Pixeln der Spielfeldbegrenzung addiert.

```
#define p2s_y(p_y) (10*p_y+35)
```

Es wird das gleiche durchgeführt wie oberhalb beschrieben. Der Unterschied ist, dass hier nicht 5, sondern 35 auf die errechnete Zahl addiert wird. Das ist notwendig, da das Spielfeld der Schlange erst 35 Pixel unterhalb des oberen Bildschirmrandes anfängt, weil darüber der aktuelle Punktestand ausgegeben wird.

Move_tail

```
void move_tail()
{
    if(counter > 0)
    {
      counter-=1;
    }
    else
    {
      tft.drawRect(p2s_x(position_x[tail]), p2s_y(position_y[tail]), 10,
      10, ILI9341_BLACK);
      blocked[position_x[tail]][position_y[tail]] = 0;
      tail++;
      if(tail >= maximum_snake_length)
      {
       tail = 0;
      }
    }
}
```

Diese Funktion ist für die Regulierung der Schlangenlänge verantwortlich. In der IF-Schleife wird überprüft, ob die Variable `counter`, welche in dem Loop des Programms mit 4 initialisiert wird, größer als null ist. Wenn dies der Fall ist, wird die Variable `counter` -1 gerechnet. Die Else-Schleife tritt somit nicht ein. Währenddessen wird innerhalb der Loop-Schleife ein grünes Rechteck gezeichnet, das nicht wieder schwarz gezeichnet wird, weil die Funktion `move_tail` nur einmal pro Durchgang des Loops aufgerufen wird. Da die Variable `counter` mit vier initialisiert wurde, wird die If-Schleife fünf Mal aufgerufen. Das Zählen in C++ beginnt immer bei 0 -> 0, 1, 2, 3, 4, 5. So erreicht die Schlange eine Länge von fünf Einheiten. Ist die Variable nicht mehr größer als null tritt die Else-Schleife ein. Um das Rechteck wieder schwarz zeichnen zu können wird erneut mit den Funktionen p2x_x und p2y_y und den Koordinaten des Schlangenendes an der Stelle `tail` die in den Arrays `position_x` und `position_y` gespeichert wurden, die Koordinaten errechnet, an denen das Rechteck schwarz gezeichnet werden muss.

In dem Array `blocked` werden die Koordinaten gespeichert, an denen die Schlange sich aktuell befindet. Um eine Kollision an einer Stelle zu verhindern an der die Schlange sich aktuell nicht befindet, müssen die Koordinaten des Rechtecks, welches einen Schritt davor wieder schwarz gezeichnet wurde, in dem Array „`blocked`“ mit einer null vermerkt werden. Als vorletzter Schritt wird die Variable `tail` eins hochgezählt, um zu verhindern, dass Werte in den Arrays `position_x` und `position_y` wieder an die gleiche Stelle geschrieben werden. Da in den angesprochenen Arrays nur 300 Werte gespeichert werden dürfen, wird anhand einer IF-Schleife überprüft, ob die Variable `tail` den Wert 300 erreicht hat. Erreicht sie diesen sind die Arrays voll und müssen „geleert“ werden um neue Werte ohne Fehler speichern zu können.

Print_score

```
void print_score()
{
    tft.fillRect(173, 4, 250, 24, ILI9341_BLACK);.
    tft.setTextColor(ILI9341_RED);
    tft.setTextSize(3);
    tft.setCursor(65, 4);
    tft.print("Score:");
    tft.print(score);
}
```

Ausgabe des Scores. Diese Funktion wird innerhalb der Loops zum Aktualisieren des Scores aufgerufen.

Void Setup

```
void setup()
{
  Serial.begin(9600);
  pinMode(Joystick_Button, INPUT_PULLUP);
  tft.begin();
  tft.setRotation(3);
}
```

Erläuterung:

Die Programmstruktur des Arduino ist in zwei Abschnitte aufgeteilt. Der Setup Teil und der Loop Teil. Programmcode, der in dem Setup Teil steht, wird nur zu Beginn des Programms einmal ausgeführt. Programmcode innerhalb des Loop Teils wird so lange ausgeführt bis der Arduino von der Stromquelle getrennt wird.

Serial.begin(9600)	Verbindung für die Anzeige des seriellen Monitors
pinMode(Joystick_Button, INPUT_PULLUP)	Pin Modus festlegen
tft.begin()	Display initialisieren
tft.setRotation(3)	Display Richtung bestimmen.

Void Loop

```
void loop(void)
{
  int JoystickX = analogRead(Joystick_X);
  int JoystickY = analogRead(Joystick_Y);
```

Um die Richtung des Joysticks zu jeder Zeit richtig zu erkennen, werden dauerhaft die Position des Joysticks auf der X und Y-Achse ausgelesen und in den Variablen `JoystickX` und `JoystickY` gespeichert.

Erläuterung: Der Programmteil des Loops wurde mit einer Switch-Case Anweisung aufgebaut.

```
  switch(game_status)
  {
    case 0:
          welcome_screen();
          game_status=1;
          break;
```

In Case 0 wird der `Welcome_screen` angezeigt und die Variable `game_status` auf 1 gesetzt. Somit springt das Programm in Case 1. Ein break bedeutet, dass der Case an der Stelle fertig ist und das Programm in den nächsten Case springen kann.

```
    case 1:
      if(digitalRead(Joystick_Button)==LOW)
      {
```

Nachdem das Programm in Case 1 gesprungen ist, wird auf eine Benutzereingabe gewartet. Mit einer If-Abfrage wird überprüft, ob der Benutzer den Button des Joysticks drückt, um das Programm fortzusetzten. Der `welcome_screen` wird so lange angezeigt bis der Button gedrückt wurde.

```
        game_screen();
        game_status = 2;

        // -----------------------------------
        CursorX = 0; dx=1;
        CursorY = 0; dy=0;
        head=0;
        tail=0;
        counter=4;
        score=0;
        print_score();
        // -----------------------------------
```

Es wird die Funktion `game_screen` gestartet und Variablen mit neuen Werten initialisiert. Der `game_status` wird auf 2 gesetzt. Die Variable `counter` bestimmt die Anfangslänge der Schlange wie bereits in der Beschreibung der Funktion `move_tail` beschrieben.

```
        for (int i=0;i<31;i++) {
          for (int j=0;j<20;j++) {
            blocked[i][j]=0;
          }
        }
        food_generator();
      }

      break;
```

Diese zwei For-Schleifen löschen bei jedem Start eines Spiels die bereits gespeicherten Werte aus dem Array `blocked`. Würden dieser Werte nicht gelöscht werden würden bei einem erneutem Spiel Fehler auftreten. Die Koordinaten für das Essen der Schlange werden mit der Funktion `food_generator` generiert.

```
case 2:

  if(blocked[CursorX][CursorY]==1
  {
    game_status=3;
    if(score > highscore)
    {
      highscore = score;
    }
  }
```

In Case 2 findet der Hauptteil des Spieles statt. Um zu überprüfen, ob eine Kollision mit dem Körper der Schlange stattgefunden hat, wird mit einer If-Abfrage überprüft, ob die aktuellen Werte von `CursorX` und `CursorY` in dem Array „`blocked`“ mit einer eins vermerkt sind. Wenn ja, wird der game_status auf 3 gesetzt und überprüft, ob der erreichte Score höher ist als der aktuelle Highscore. Sollte er höher sein, wird er in der Variable Highscore gespeichert.

```
  tft.drawRect(random_food_x,random_food_y, 10, 10, ILI9341_RED);
  tft.drawRect(p2s_x(CursorX), p2s_y(CursorY), 10, 10, ILI9341_GREEN);
  tft.drawRect(random_food_x,random_food_y, 10, 10, ILI9341_RED);
  blocked[CursorX][CursorY]=1;
```

In diesem Abschnitt werden das Essen (der Apfel) und die Schlange gezeichnet. Mit den Funktionen `p2s_x` und `p2s_y` werden die Koordinaten für das nächste Rechteck der Schlange berechnet. Es ist möglich das der Apfel auf der Schlange gezeichnet wird. Damit der Apfel nachdem laufen der Schlange nicht verschwindet, da die Schlange ein neues Rechteck über das Rechteck des Essens zeichnet, wird der Apfel mit denselben Koordinaten noch einmal gezeichnet. Die aktuelle Position des Kopfes der Schlange wird in dem Array `blocked` gespeichert und mit einer 1 vermerkt. Dies wird für die Kollisionsberechnung benötigt.

```
  delay(125);
```

Mit dem delay wird die Spielgeschwindigkeit bestimmt. Ein höherer Wert als 125 sollte nicht gewählt werden. Sollte ein geringeres Delay gewählt werden stimmen die Reaktionsgeschwindigkeit der Schlange mit dem Joystick nicht mehr überein. Der Joystick müsste länger an einer Position gehalten werden, bis das Programm merkt, dass die Schlange in eine andere Richtung laufen soll. Ein flüssiges Spielerlebnis ist so nicht möglich. Ein geringerer delay als 125 lässt die Schlange schneller laufen.

```
position_x[head] = CursorX;
position_y[head] = CursorY;
head++;

if(head >= maximum_snake_lenght)
{
  head=0;
}
```

In den Arrays `position_x` und `position_y` werden die Positionen gespeichert, an denen die Schlange gerade war. Um die Werte nicht zu überschreiben, wird das Array nach jedem Speichern mit Hilfe der Variable `head` um eins hochgezählt. Aufgrund der maximalen Länge von 300 Einheiten wird mit Hilfe der If-Schleife überprüft, ob die Anzahl erreicht wurde. Wurde sie erreicht wird die Variable `head` auf 0 gesetzt. Somit werden Werte wieder an Stelle 1 der Arrays geschrieben.

```
move_tail();
```

Nachdem die Schlange gelaufen ist, wird die Funktion `move_tail` aufgerufen.

```
CursorX+=dx;
CursorY+=dy;
```

Um eine Kollision im nächsten Schritt erkennen zu können, wird mit jedem Schritt der Schlange die Variable `CursorX` und `CursorY` je nach Laufrichtung der Schlange hoch oder runtergezählt.

```
if(CursorX == 31)
{
  game_status = 3;
  if(score > highscore)
  {
    highscore = score;
  }
}
if(CursorX < 0)
{
  game_status = 3;
  if(score > highscore)
  {
    highscore = score;
  }
}
if(CursorY == -1)
{
  game_status = 3;
  if(score > highscore)
  {
    highscore = score;
  }
}
if(CursorY == 20)
{
  game_status = 3;
  if(score > highscore)
  {
    highscore = score;
  }
}
```

Diese If-Schleifen überprüfen anhand der Variable `CursorX` und `CursorY` ob eine Kollision mit der linken, rechten, unteren oder oberen Bande des Spielfeldes stattgefunden hat.

```
if(random_generator_for_food_x == CursorX && random_genera
tor_for_food_y == CursorY)
{
  food_generator();
  counter=1;
  score+=10;
  print_score();
}
```

Um die Schlange wachsen zu lassen muss noch überprüft werden, ob die Schlange gegessen hat. Stimmen die Koordinaten in der If-Schleife überein wird die Funktion `food_generator` aufgerufen, um neue Koordinaten zum Zeichnen eines neuen Apfels zu erhalten. Die Variable `counter` wird um 1 erhöht um die Schlange einen Block länger werden zu lassen. Die Variable Score wird um 10 erhöht und die Funktion `print_score` aufgerufen, um den richtigen Punktestand auf dem Display anzuzeigen

```
if(JoystickX>900) // laufe nach rechts
{
  dx=1;dy=0;
 }
if(JoystickX<300) // laufe nach links
{
  dx=-1;dy=0;
}
if(JoystickY>900) // laufe nach unten
{
  dx=0;dy=1;
}
if(JoystickY<300) // laufe nach oben
{
  dx=0;dy=-1;
}

break;
```

Die letzten 4 If-Schleifen in Case 2 überprüfen die aktuelle Richtung des Joysticks. Bei jeweiliger Richtung werden die Variablen dx und dy angepasst.

```
  case 3:
    {
      loser_screen();
      game_status=1;
      break;
    }
  }
}
```

Der letzte Case führt bei einem verlorenen Spiel die Funktion `loser_screen` aus. Der `game_status` wird auf 1 gesetzt. Das break signalisiert das Ende des Case. Da der `game_status` auf 1 gesetzt wurde, springt das Programm wieder in Case 1. Aufgrund der If-Schleife in Case 1 die zuerst auf die Benutzereingabe, dem drücken des Joysticks wartet, bleibt der `looser_screen` so lange bestehen bis der Spieler ein neues Spiel startet.

Struktogramme

Define

define

```
include "SPI.h"
include "Adafruit_GFX.h"
include "Adafruit_ILI9341.h"
include "bitmaps.h"

define TFT_DC 9
define TFT_CS 10

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

define Joystick_X A4
define Joystick_Y A3
define Joystick_Button 2
define maximum_snake_lenght 300
define spielfeld_breite 31
define spielfeld_hoehe 20

define p2s_x(p_x) (10*p_x+5)
define p2s_y(p_y) (10*p_y+35)
```

Initialisierung

Initialisierungen

Deklaration: JoystickX, JoystickY als Ganzzahl
Deklaration: score als Ganzzahl
Deklaration: highscore als Ganzzahl
Deklaration: game_staus als Ganzzahl
Deklaration: CursorX, CursorY als Ganzzahl
Deklaration: dx, dy als Ganzzahl
Deklaration: head als Ganzzahl
Deklaration: tail als Ganzzahl
Deklaration: counter als Ganzzahl
Deklaration: random_food_x, random_food_y als Ganzzahl
Deklaration: random_generator_for_food_x, random_generator_for_food_y als Ganzzahl

Deklaration: blocked[spielfeld_breite][spielfeld_hoehe] als **byte**
Deklaration: position_x[maximum_snake_lenght] als **byte**
Deklaration: position_y[maximum_snake_lenght] als **byte**

Initialisierung: score ← 0
Initialisierung: highscore ← 0
Initialisierung: game_status ← 0
Initialisierung: dx ← 1, dy ← 0
Initialisierung: head ← 0
Initialisierung: tail ← 0
Initialisierung: counter ← 5

Welcome_screen

welcome_screen

Fülle das Display mit der Farbe schwarz aus
Setze die Textfarbe auf weiß
Setze die Textgröße auf 3
Setze den Cursor an die Koordinaten **X**:40, Y:30
Gebe den Text "Willkommen zu" aus
Zeichne die Bitmap wellcome_snake an den Koordinaten **X**:110, Y:75 mit einer Größe von 92 Pixeln in der Breite und 56 in der Höhe in der Farbe Grün
Setze den Cursor an die Koordinaten **X**:35, Y:160
Setze die Textgröße auf 2
Gebe den Text "Push Joystick to start" aus

Game_screen

game_screen

Fülle das Display mit der Farbe schwarz aus
Zeichne ein Rechteck an den Koordinaten **X**:0, Y:30 mit einer Breite von 320 und einer Höhe von 210 Pixeln in der Farbe Lila
Zeichne ein Rechteck an den Koordinaten **X**:1, Y:31 mit einer Breite von 318 und einer Höhe von 208 Pixeln in der Farbe Lila
Zeichne ein Rechteck an den Koordinaten **X**:2, Y:32 mit einer Breite von 316 und einer Höhe von 206 Pixeln in der Farbe Lila
Zeichne ein Rechteck an den Koordinaten **X**:3, Y:33 mit einer Breite von 314 und einer Höhe von 204 Pixeln in der Farbe Lila
Zeichne ein Rechteck an den Koordinaten **X**:0, Y:34 mit einer Breite von 312 und einer Höhe von 202 Pixeln in der Farbe Lila
Setze die Textfarbe auf Rot
Setze die Textgröße auf 3
Setze den Cursor an die Koordinaten **X**:65, Y:4
Gebe den Text "Score:" aus
Gebe den Score aus

Loser_screen

looser_screen

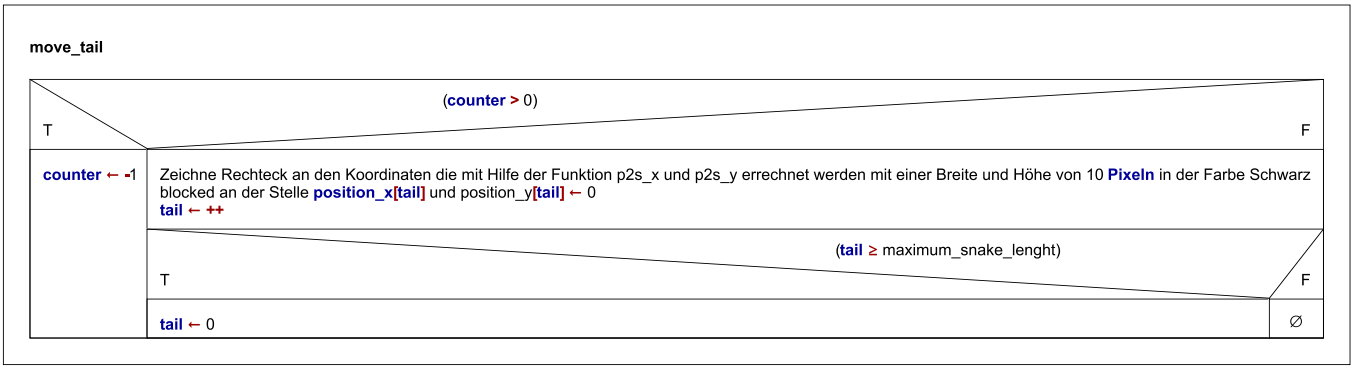
Fülle das Display mit der Farbe Schwarz aus
Setze die Textfarbe auf Weiß
Setze die Textgröße auf 3
Setze den Cursor an die Koordianten **X**:50, Y:10
Gebe den Text "! You Lost !" aus
Zeichne die Bitmap dead_snake an den Koordinaten **X**:100, Y:55 mit einer Größe von 124 Pixeln in der Breite und 38 in der Höhe in der Farbe Rot
Setze den Cursor an die Koordinaten **X**:45, Y:110
Setze die Textgröße auf 3
Gebe den Text "Learn to play" aus
Setze den Cursor an die Koordinaten **X**:65, Y:140
Setze die Textfarbe auf Rot
Gebe den Text "Score:" aus
Gebe den score aus
Setze den Cursor an die Koordinaten **X**:40, Y:170
Gebe den Text "Highscore:" aus
Gebe den Highscore aus
Setze die Textgröße auf 2
Setze die Textfarbe auf Weiß
Setze den Cursor an die Koordinaten **X**:12, Y:210
Gebe den Text "Play again? Push Joystick" aus

Food_generator

food_generator

random_generator_for_food_x ← Zufallszahl zwischen 0 und 31
random_generator_for_food_y ← Zufallszahl zwischen 0 und 20
random_food_x ← p2s_x(**random_generator_for_food_x**)
random_food_y ← p2s_y(**random_generator_for_food_y**)

Move_tail



Print_score

print_score

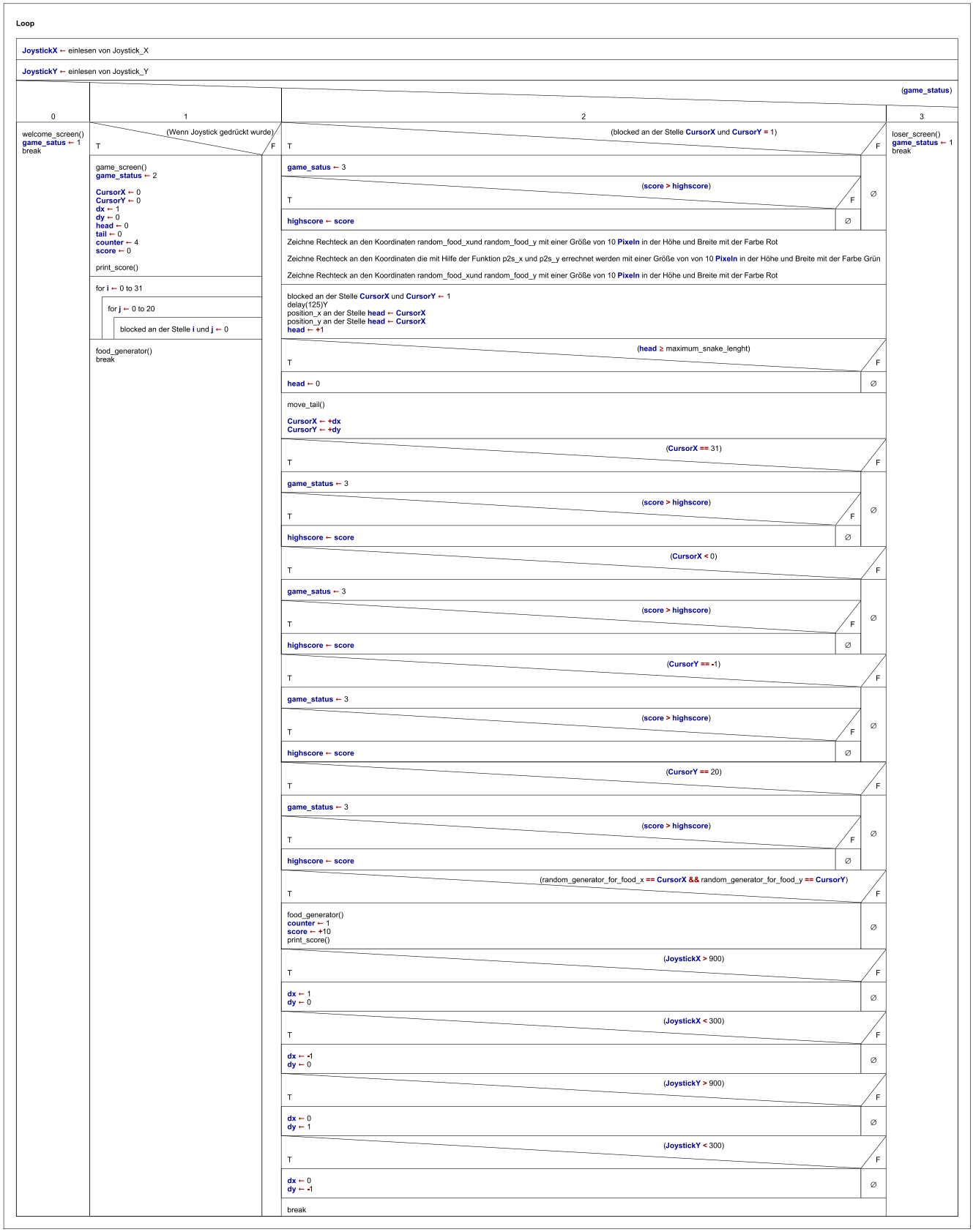
Fülle das Display mit der Farbe Schwarz aus
Setze die Textfarbe auf Rot
Setze die Textgröße auf 3
Setze den Cursor an die Koordinaten X:65, Y:4
Gebe den Text "Score:" aus
Gebe den Score aus

Setup

setup

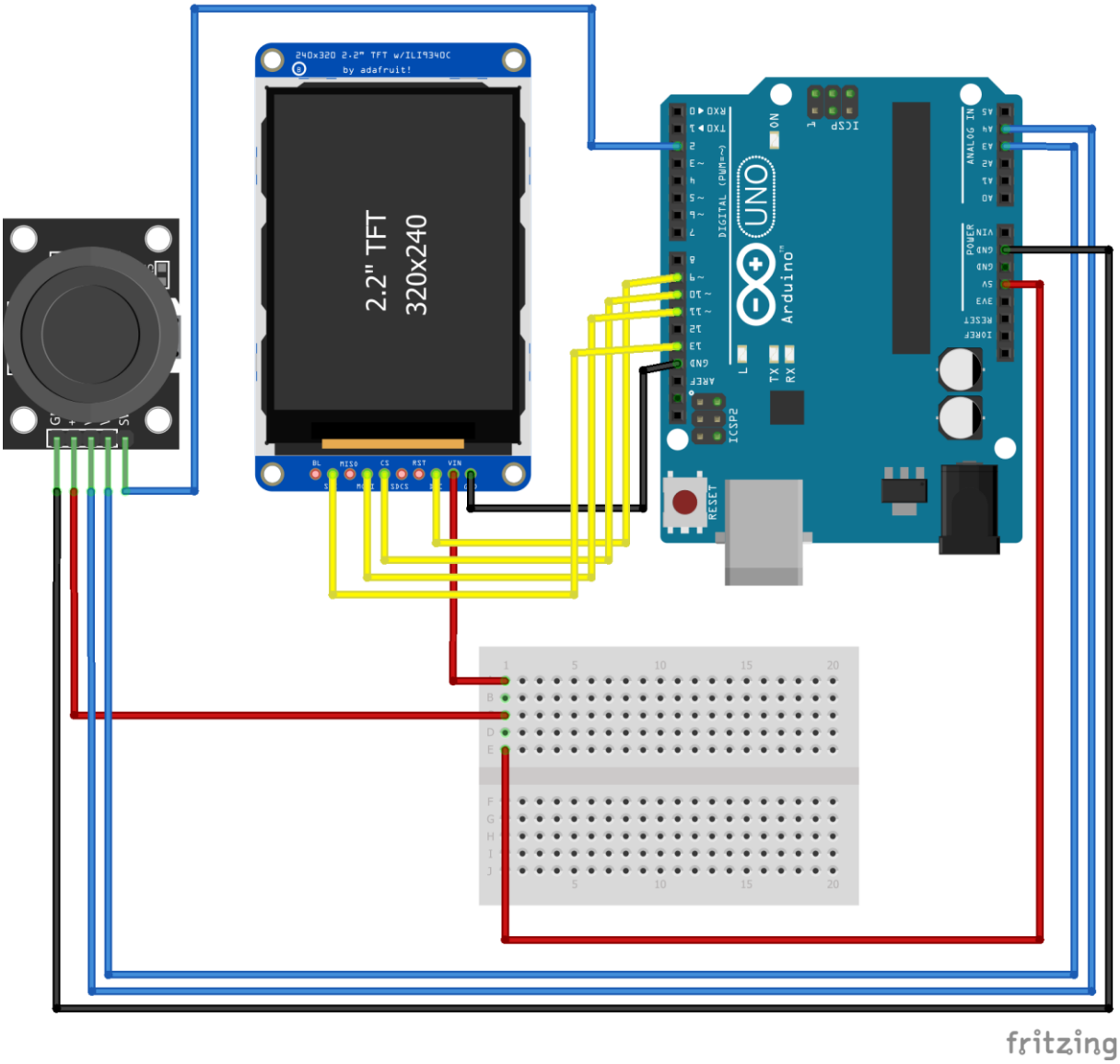
Stelle den PinMode des Joystick_Button auf Input_Pullup
Initialisiere das Display
Setze die Bildschirm Rotation auf 3

Loop


Loop
JoystickX ← einlesen von Joystick_X
JoystickY ← einlesen von Joystick_Y
(game_status)
0
1
2
3
welcome_screen()
game_satus ← 1
break
(Wenn Joystick gedrückt wurde)
T
F
game_screen()
game_status ← 2
CursorX ← 0
CursorY ← 0
dx ← 1
dy ← 0
head ← 0
tail ← 0
counter ← 4
score ← 0
print_score()
for i ← 0 to 31
for j ← 0 to 20
blocked an der Stelle i und j ← 0
food_generator()
break
(blocked an der Stelle CursorX und CursorY = 1)
T
F
game_satus ← 3
(score > highscore)
T
F
highscore ← score
∅
Zeichne Rechteck an den Koordinaten random_food_xund random_food_y mit einer Größe von 10 PixeIn in der Höhe und Breite mit der Farbe Rot
Zeichne Rechteck an den Koordinaten die mit Hilfe der Funktion p2s_x und p2s_y errechnet werden mit einer Größe von von 10 PixeIn in der Höhe und Breite mit der Farbe Grün
Zeichne Rechteck an den Koordinaten random_food_xund random_food_y mit einer Größe von 10 PixeIn in der Höhe und Breite mit der Farbe Rot
blocked an der Stelle CursorX und CursorY ← 1
delay(125)Y
position_x an der Stelle head ← CursorX
position_y an der Stelle head ← CursorX
head ← +1
(head ≥ maximum_snake_lenght)
head ← 0
move_tail()
CursorX ← +dx
CursorY ← +dy
(CursorX == 31)
(CursorX < 0)
(CursorY == -1)
(CursorY == 20)
(random_generator_for_food_x == CursorX && random_generator_for_food_y == CursorY)
food_generator()
counter ← 1
score ← +10
print_score()
(JoystickX > 900)
dx ← 1
dy ← 0
(JoystickX < 300)
dx ← -1
dy ← 0
(JoystickY > 900)
dx ← 0
dy ← 1
(JoystickY < 300)
dx ← 0
dy ← -1
break
loser_screen()
game_status ← 1
break

Fritzing

Die Abbildung zeigt die nötige Verkabelung des Display und Joystick um richtig an den Arduino Uno angeschlossen zu werden.



Quellen

GitHub - Stiju/arduino_snake (Stand 2.02.2022) – Bitmaps und Code Beispiel für Snake	Mit Hilfe des Codes konnte ich die Bitmaps in meinen Code integrieren und einen ersten Überblick erhalten, wie mein Snake Spiel aussehen könnte.	https://github.com/Stiju/arduino_snake
2.2 18-bit color TFT LCD display with microSD card breakout - Adafruit Industries, Unique & fun DIY electronics and kits (Stand 15.11.2021) – Technische Details, Grafik Bibliothek	Hier sind alle Informationen über das Display niedergeschrieben. Eine wichtige Information war, ob das Display einen integrierten „level shifter“ hat.	https://www.adafruit.com/product/1480
Nr.22 - Joystick Modul am Arduino | Funduino - Kits und Anleitungen für Arduino (Stand 10.10.2021) – Beispiel Code für Joystick	Anhand dieser Webseite habe ich die Informationen erhalten, wie ich die aktuelle Position des Joysticks auswerte und mit dem Arduino weiterverarbeite.	https://funduino.de/nr-22-joystick-modul
Arduino Code | 2.2" TFT Display | Adafruit Learning System (Stand 20.12.2021) – How to use Display	Diese Webseite fast zusammen wie das Display mit der Entwicklungsumgebung Arduino IDE richtig angesteuert und wird.	https://learn.adafruit.com/2-2-tft-display/arduino-code
Arduino 2.2" TFT Display Overview (Stand Ende November 2021) – Overview 2.2“ Display	Diese Webseite fast zusammen, wie das Display funktioniert, was die einzelnen Pinouts bedeuten und welche eine feste Belegung am Arduino haben.	https://learn.adafruit.com/2-2-tft-display?view=all