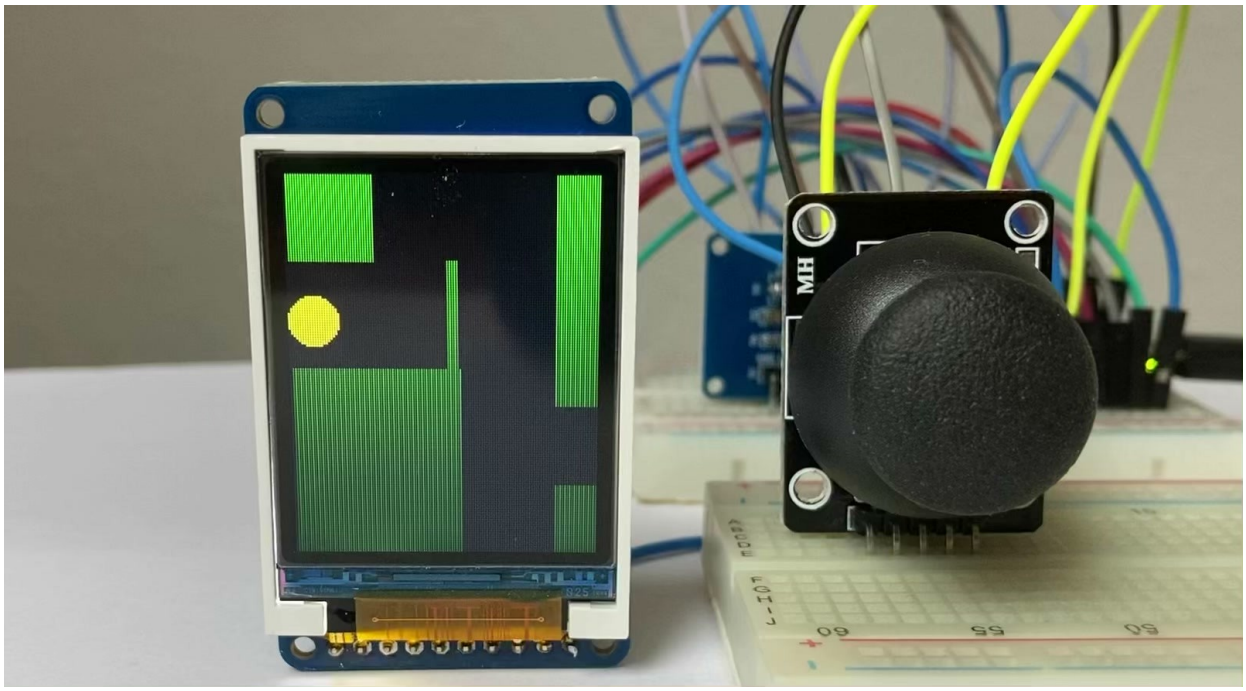
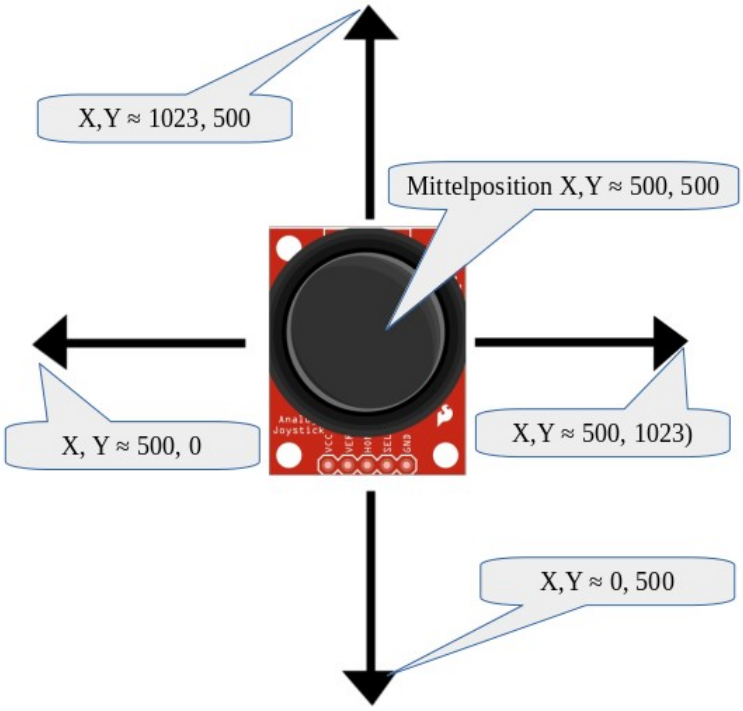
Mit Hilfe eines Joystick wird auf einem TFT-Monitor ein Ball durch ein kleines Labyrinth bewegt. Am Ziel wird die Zeit gemessen.



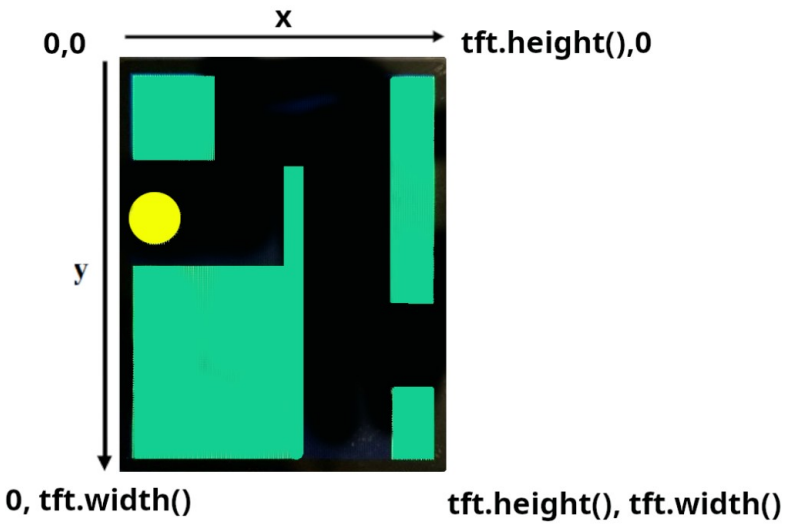


Der Joystick besteht aus zwei Potentiometern, jeweils einer für die X-Achse und einer für die Y-Achse. Beide lesen bei den Bewegungen die Spannung und liefern dem Arduino jeweils einen analogen Wert, der zwischen 0 und 1023 liegt.
Der Knopf funktioniert durch Drücken auch gleichzeitig als Schalter.

Die Werte können je nach Joystick leicht nach oben oder unten abweichen.

Die Beschriftung und die Reihenfolge der Pins können sich je nach Joystick unterscheiden:

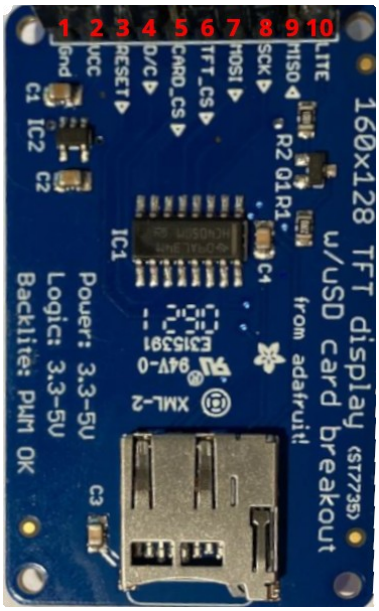
Achse	Bezeichnung	Anschluss
X-Achse	VRx/VER	A0
Y-Achse	VRy/HOR	A1
Button	SW/SEL	7



Das verwendete TFT-Modul von Adafruit hat eine Bildschirmdiagonale von 1,8 Zoll und eine Bildschirmauflösung von 160x128 Pixeln.

Benötigte Bauteile:

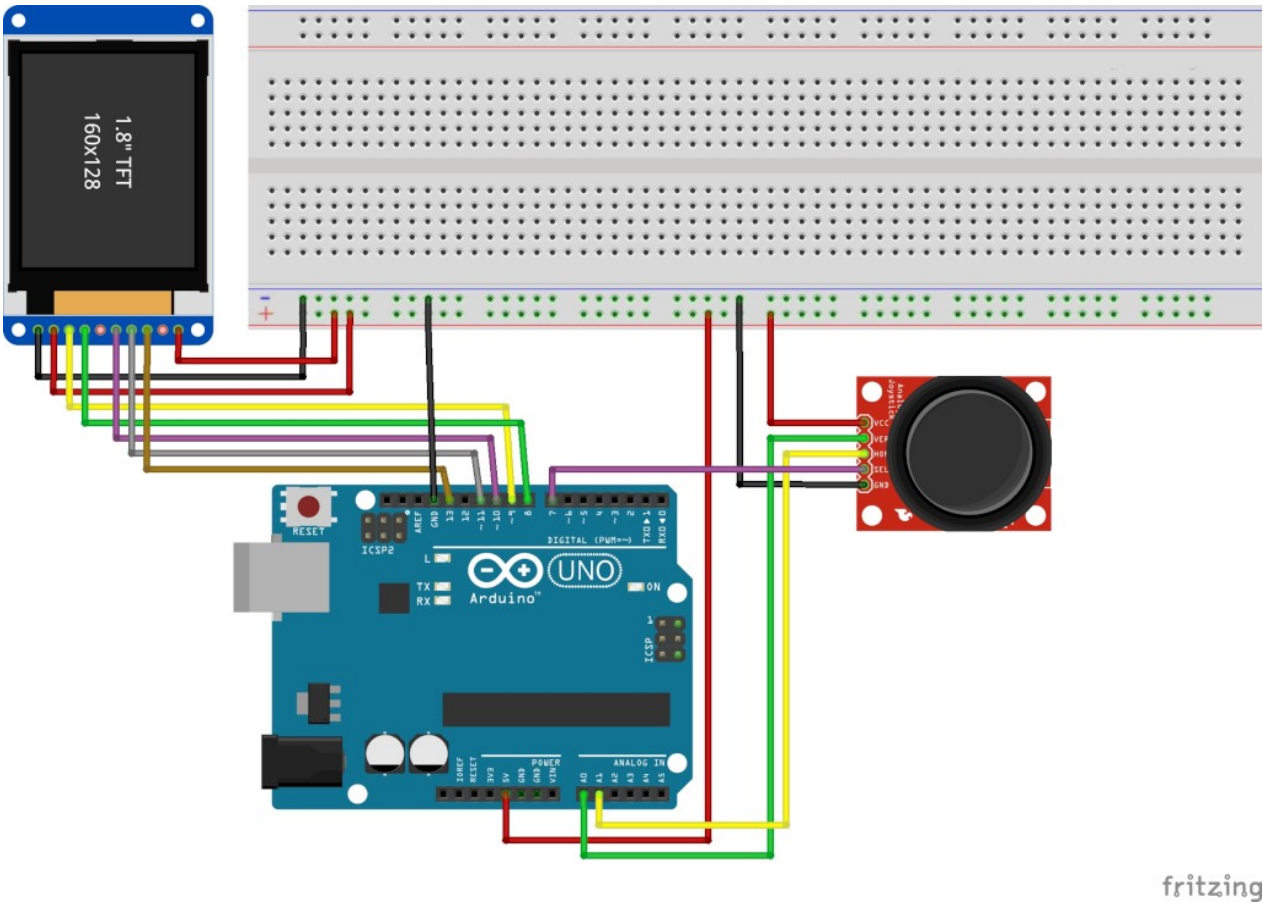
- Joystick
- Adafruit 1,8 Zoll TFT ST7735
- Leitungsdrähte



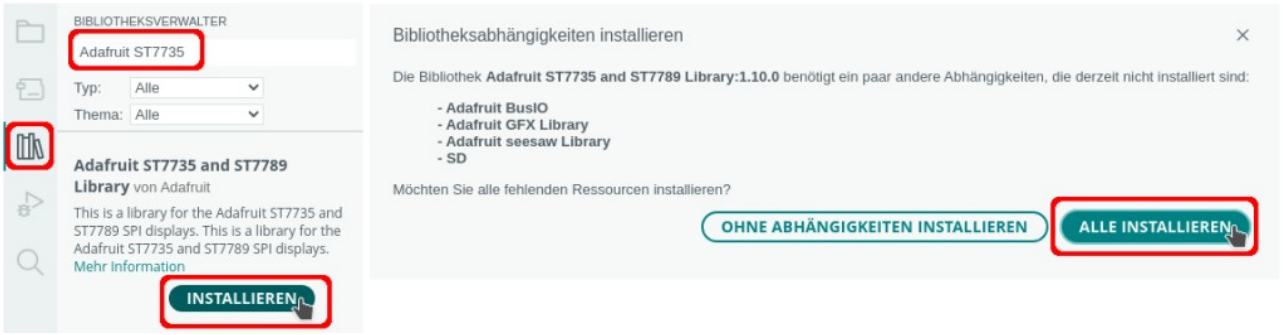
Pinbelegung Adafruit 1,8 Zoll TFT

1 -> Gnd -> GND
2 -> VCC -> 5V
3 -> RESET -> D9
4 -> D/C -> D8
5 -> CARD_CS (nicht angeschlossen)
6 -> TFT_CS -> D10
7 -> SDO -> D11
8 -> SCK -> D13
9 -> SDI (nicht angeschlossen)
10 →5V

Baue die Schaltung auf.



Benötigte Bibliotheken:

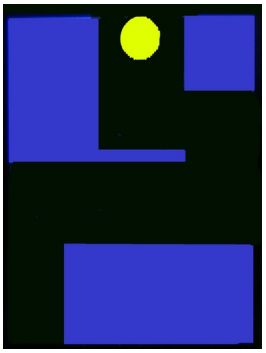


Überblick über die Funktionen der Bibliothek Adafruit ST7735/TFT:

Schlüsselwort	Parameter	Aktion
begin();		TFT starten
initR(INITR_*TAB);	BLACKTAB GREENTAB REDTAB	Farbschema bestimmen
setRotation(Richtung);	Richtung = 0 → nicht drehen Richtung = 1 → 90° drehen Richtung = 2 → 180° drehen	Bildschirm ausrichten
fillScreen(Farbe)		Bildschirmhintergrund
drawLine(StartX, StartY, EndeX, EndeY, Farbe);		Linie zeichnen
drawFastHLine(StartX, StartY, Länge, Farbe);		horizontale Linie zeichnen
drawFastVLine(StartX, StartY, Länge, Farbe);		vertikale Linie zeichnen
drawRect(StartX, StartY,, Breite, Höhe, Farbe);		Rechteck zeichnen
drawRoundRect(StartX, StartY, Breite, Höhe, Eckenradius, Farbe);		abgerundetes Rechteck zeichnen
fill.Rect(StartX, StartY, Breite, Höhe, Farbe);		ausgefülltes Rechteck zeichnen
drawCircle(MittelpunkX, MittelpunktY, Radius, Farbe);		Kreis zeichnen
fillCircle(MittelpunkX, MittelpunktY, Radius, Füllfarbe);		Ausgefüllten Kreis zeichnen
setCursor(x, y);		Cursor setzen
setTextSize(Textgröße);	Textgröße: 1 - 4	Textgröße bestimmen
setTextColor(Farbe);		Textfarbe setzen
print("Text"); println("Text");		Text schreiben

Beispiel mit den Grafik- und Textfunktionen

Das Labyrinth



Binde die benötigen Bibliotheken ein und definiere die Variablen. Beachte die Kommentare.

```
/*
  Pinbelegung:
  GND      (1) - GND
  VCC      (2) - 5V
  RESET    (3) - D9
  D/C      (4) - D8
  CARD-CS  (5) -
  TFT-CS   (6) - D10
  MOSI     (7) - D11
  SCK      (8) - D13
  MISO     (9) -
  LITE    (10) - 5V
*/
# include <Adafruit_GFX.h>
# include <Adafruit_ST7735.h>

// Pins zuordnen
# define TFT_CS        10
# define TFT_RST        9
# define TFT_DC         8

// Name des TFTs und die zugeordneten Pins
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

/*
  Farben als hexadezimal definiert
  alternativ:
  int SCHWARZ = 0;
  int BLAU = 15;
  . . .
*/
# define SCHWARZ    0x0000 // dezimal 0
# define BLAU       0x000F // dezimal 15
# define ROT        0xF800 // dezimal 406664
# define GRUEN      0x0E81 // dezimal 3713
# define CYAN       0x07FF // dezimal 2047
# define MAGENTA    0xF81F // dezimal 63519
# define GELB       0xAFE5 // dezimal 65504
# define WEISS      0xFFFF // dezimal 65535
# define BRAUN      0xFC00 // dezimal 64512
# define GRAU       0xF7F0 // dezimal 63472
# define GRUENGELB  0xAFE5 // dezimal 45029
# define DUNKELCYAN 0x03EF // dezimal 1007
# define ORANGE     0xFD20 // dezimal 64800
# define PINK       0xFC18 // dezimal 64536

// Farbe der Blöcke
# define FARBE GRUEN

// Farbe des Kreises
# define KREISFARBE GELB
```

```
// Farbe der Schrift
# define SCHRIFTFARBE WEISS

// Joystick
// analoge Pins
int XAchse = A0;
int YAchse = A1;

// Button/Knopf
int JoystickButton = 7;

// Zustand des Buttons
int ButtonLesen;

// Spiel starten wenn * gedrückt wurde
bool SpielStart = false;

// Radius des Kreises
const int Radius = 10;

// Abstand zu den Rändern
const int Abstand = Radius * 2;

// je höher, dest langsamer
const int Geschwindigkeit = 100;

// Bewegung des Kreises in Pixeln
const int Bewegung = 5;

// Startposition des Kreises
int CursorX = Radius;
int CursorY = tft.height() / 2 – Abstand;

// Variablen für die Auswertung der Bewegung des Joysticks
int PositionX;
int PositionY;

// Variable für die Zeitmessung
long Start;
```

Der setup-Teil:

```
void setup()
{
  pinMode(JoystickButton, INPUT_PULLUP);

  // Startbildschirm
  // schwarzes Farbschema horizontale Ausrichtung
  // Cursor setzen, Schriftgröße und -farbe definieren
  tft.initR(INITR_BLACKTAB);
  tft.setRotation(1);
  tft.fillScreen(SCHWARZ);
  tft.setTextSize(2);
  tft.setCursor(1, 10);
  tft.setTextColor(ROT);

  tft.println("Start:");
  tft.print("-> Taste");
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  // Button/Knopf auswerten
  ButtonLesen = digitalRead(JoystickButton);

  if (ButtonLesen == LOW)
  {
    // Spiel wird gestartet
    SpielStart = true;

    // Parcours bauen
    ParcoursBauen();

    // Zeitmessung starten ➡millis()
    Start = millis();
  }

  // wenn der Button gedrückt wurde
  if (SpielStart)
  {
    // Bewegung der X-Achse lesen
    PositionX = analogRead(XAchse);
    // Bewegung X-Achse nach oben
    if (PositionX > 600)
    {
      // Kreis an der aktuellen Position "löschen"
      tft.fillCircle(CursorX, CursorY, Radius, SCHWARZ);

      // wenn der Bildschirmrand noch nicht erreicht wurde
      // vorwärts bewegen
      if (CursorX < BildschirmBreite) CursorX += Bewegung;
      tft.fillCircle(CursorX, CursorY, Radius, GELB);
      delay(Geschwindigkeit);
    }
```

```
    // Bewegung X-Achse nach unten
    if (PositionX < 300)
    {
      tft.fillCircle(CursorX, CursorY, Radius, SCHWARZ);
      if (CursorX > Radius) CursorX -= Bewegung;
      tft.fillCircle(CursorX, CursorY, Radius, GELB);
      delay(Geschwindigkeit);
    }

    // Bewegung der Y-Achse lesen
    PositionY = analogRead(YAchse);

    // Bewegung Y-Achse nach rechts
    if (PositionY > 600)
    {
      tft.fillCircle(CursorX, CursorY, Radius, SCHWARZ);
      if (CursorY < BildschirmHoehe - Radius) CursorY += Bewegung;
      tft.fillCircle(CursorX, CursorY, Radius, GELB);
      delay(Geschwindigkeit);
    }

    // Bewegung Y-Achse nach links
    if (PositionY < 300)
    {
      tft.fillCircle(CursorX, CursorY, Radius, SCHWARZ);
      if (CursorY > Radius) CursorY -= Bewegung;
      tft.fillCircle(CursorX, CursorY, Radius, GELB);
      delay(Geschwindigkeit);
    }

    // unterer Bildschirmrand erreicht -> Spielende
    if (CursorX > BildschirmBreite - Radius)
    {
      ErgebnisZeigen();
    }
  }
}
```

Jetzt fehlen noch die Methoden `ErgebnisZeigen()` und `ParcoursBauen()`:

```
void ErgebnisZeigen()
{
  // Zeit berechnen
  int Sekunden;
  long VerstricheneZeit = millis() - Start;
  Sekunden = int(VerstricheneZeit / 1000);

  // Zeit anzeigen
  tft.fillScreen(SCHWARZ);
  tft.setTextSize(2);
  tft.setCursor(1, 10);
  tft.setTextColor(ROT);
  tft.println(String(Sekunden) + " Sekunden\n\n");
  tft.println("Neustart\nTaste dr\232cken");
  SpielStart = false;

  // Startposition des Kreises zurücksetzen
  CursorX = Radius;
  CursorY = BildschirmHoehe / 2;
}

void ParcoursBauen()
{
  tft.fillScreen(SCHWARZ);

  // Kreis anzeigen
  tft.fillCircle(CursorX, CursorY, Radius, GELB);

  // Parcours "bauen"
  tft.fillRect(65, 35, 5, 45, BLAU);
  tft.fillRect(1, 1, 35, 35, BLAU);
  tft.fillRect(1, 80, 70, 70, BLAU);
  tft.fillRect(110, 1, 70, 95, BLAU);
}
```