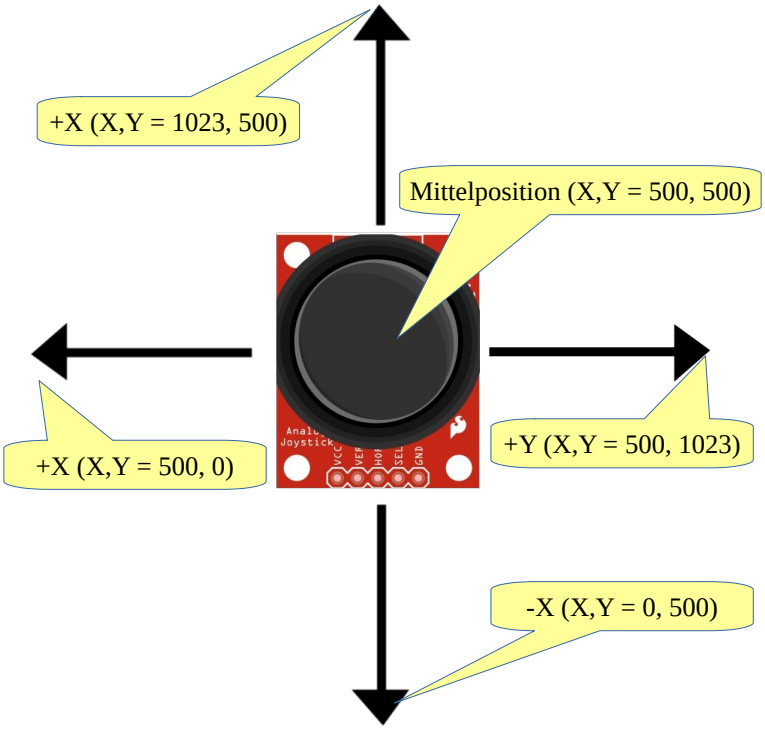
Labyrinthspiel mit Joystick und TFT-Modul

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.

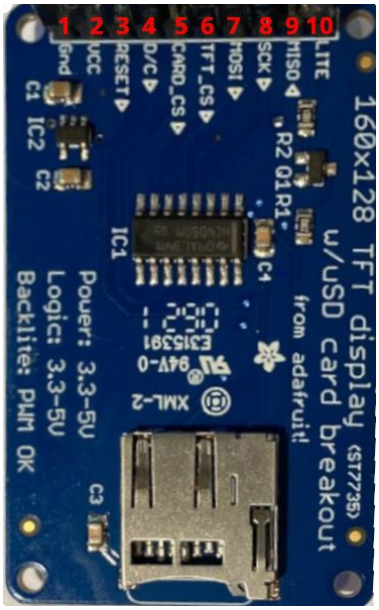
Die Beschriftung und die Reihenfolge der Pins können sich je nach Joystick unterscheiden:

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:

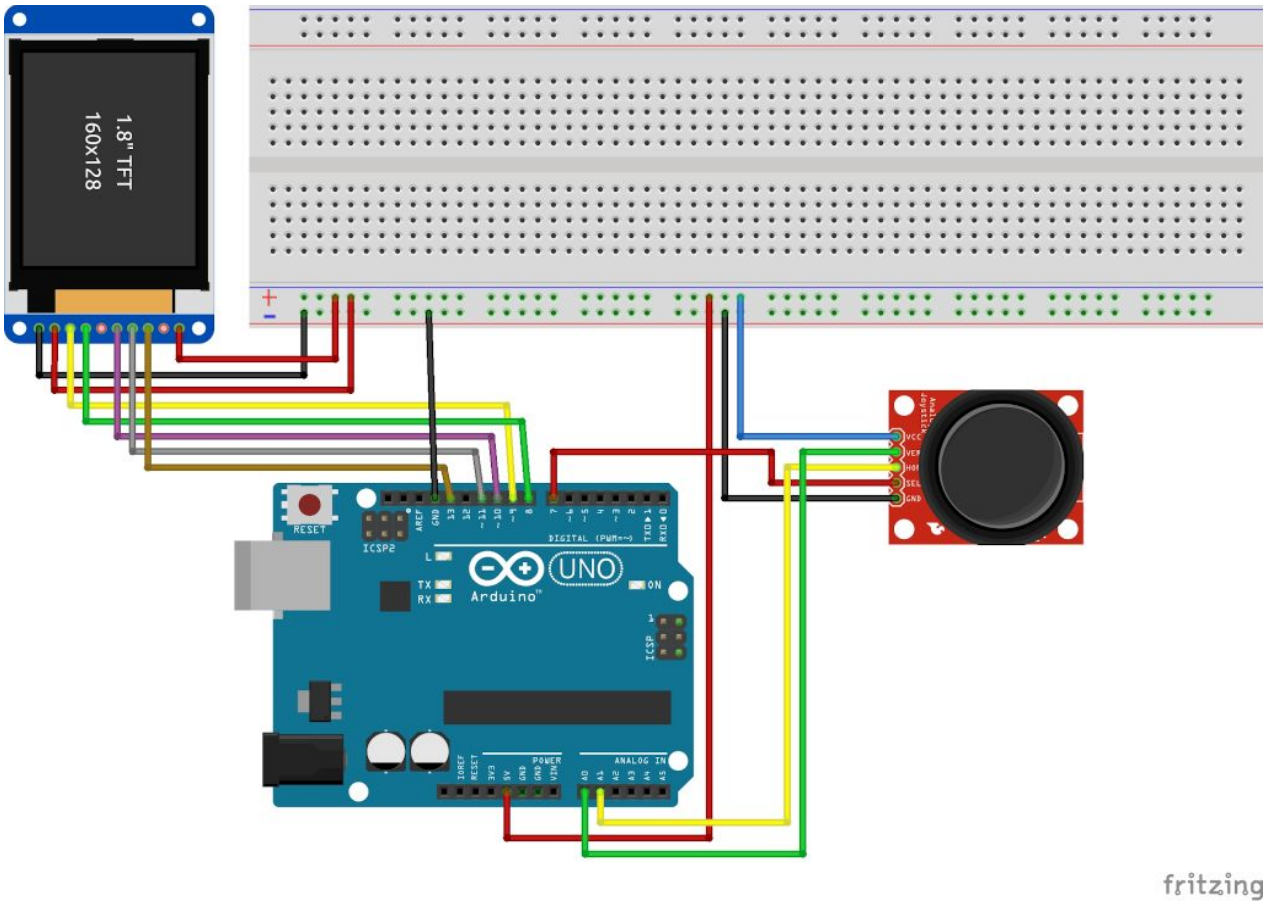
- Joystick
- Adafruit 1,8 Zoll TFT ST7735
- Leitungsdrähte



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

Pinbelegung Adafruit 1,8 Zoll TFT

Baue die Schaltung auf.



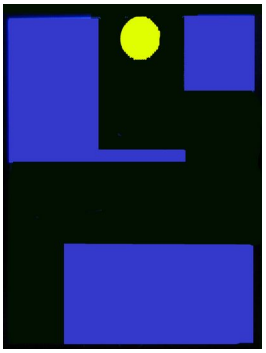
Es stehen mehrere Bibliotheken zur Auswahl, die sich aber in der Anwendung nicht unterscheiden. Der Einfachheit halber wird die eingebaute Bibliothek TFT verwendet.
Alternativ kannst du die Bibliotheken Adafruit_GFX und Adafruit_ST7735 installieren.

Überblick über die Methoden der Bibliotheken:

Methode	**Anweisung**	**Parameter**
TFT starten	begin();	
Farbschema bestimmen	initR(INITR_*TAB);	BLACKTAB GREENTAB REDTAB
Bildschirm ausrichten	setRotation(Richtung);	Richtung = 0 → nicht drehen Richtung = 1 → 90° drehen Richtung = 2 → 180° drehen
Bildschirmhintergrund füllen	fillScreen(Farbe)	
Linie zeichnen	drawLine(StartX, StartY, EndeX, EndeY, Farbe);	
horizontale Linie zeichnen	drawFastHLine(StartX, StartY, Länge, Farbe);	
vertikale Linie zeichnen	drawFastVLine(StartX, StartY, Länge, Farbe);	
abgerundetes Rechteck zeichnen	drawRoundRect(StartX, StartY, Breite, Höhe, Eckenradius, Farbe);	
ausgefülltes Rechteck zeichnen	fill.Rect(StartX, StartY, Breite, Höhe, Farbe);	
Kreis zeichnen	drawCircle(MittelpunkX, MittelpunktY, Radius, Farbe);	
Ausgefüllten Kreis zeichnen	fillCircle(MittelpunkX, MittelpunktY, Radius, Füllfarbe);	
Cursor setzen	setCursor(x, y);	
Textgröße setzen	setTextSize(Textgröße);	Textgröße: 1 - 4
Textfarbe bestimmen	setTextColor(Farbe);	
Text schreiben	print("Text"); println("Text");	
Zeilenumbruch	setTextWrap(true/false);	false → Text fließt über den Rand des TFTs hinaus true → Text wird am Ende umgebrochen

Beispiel mit den Grafik- und Textfunktionen

Das Labyrinth



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

```
# define TFT_CS        10
# define TFT_RST        9
# define TFT_DC         8

// TFT-Bibliothek
# include <TFT.h>
# include <SPI.h>

TFT tft = TFT(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     0xFFE0 // dezimal 65504
# define WEISS    0xFFFF // dezimal 65535
# define BRAUN    0xFC00 // dezimal 64512
# define GRAU     0xF7F0 // dezimal 63472

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

// Joystick Knopf
int JoystickButton = 7;

// Zustand des Buttons
int ButtonLesen;
```

```
// Spiel starten, nachdem der Knopf gedrückt wurde
bool SpielStart = false;

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

// Bildschirm Höhe/Breite
const int Radius = 10;
const int BildschirmBreite = tft.width();
const int BildschirmHoehe = tft.height();

// Geschwindigkeit der Bewegung des Joysticks
// je höher, dest langsamer
const int Geschwindigkeit = 50;

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

// Startposition des Kreises
int CursorX = Radius;
int CursorY = BildschirmHoehe / 2;

// 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);
}
```