

Vorbemerkung

Über Unsere Firma

WayinTop, Your Top Way to Inspiration, ist ein professioneller Hersteller von über 2.000 Open Source-Motherboards, -Modulen und -Komponenten. WayinTop hat sich zum Ziel gesetzt, die wunderbare Welt der eingebetteten Elektronik zu erforschen und zu entmystifizieren, einschließlich, aber nicht beschränkt auf Arduino und Raspberry Pi. Wir sind bestrebt, die am besten gestalteten Produkte für Hersteller aller Altersgruppen und Könnensstufen herzustellen. Unabhängig von Ihrer Vision oder Ihrem Kenntnisstand sind unsere Produkte und Ressourcen darauf ausgelegt, die Elektronik besser zugänglich zu machen. WayinTop wurde 2013 gegründet und ist mittlerweile auf über 100 Mitarbeiter und eine über 50.000 Quadratmeter große Fabrik in China angewachsen. Mit unseren unermüdlichen Bemühungen haben wir auch das Angebot um Werkzeuge, Ausrüstungen, Verbindungssätze und verschiedene DIY-Produkte erweitert, die wir sorgfältig ausgewählt und getestet haben.

US Amazon Store Homepage:

<https://www.amazon.com/shops/A22PZZC3JNHS9L>

CA Amazon Store Homepage:

<https://www.amazon.ca/shops/A22PZZC3JNHS9L>

UK Amazon Store Homepage:

<https://www.amazon.co.uk/shops/A3F8F97TMOROPI>

DE Amazon Store Homepage:

<https://www.amazon.de/shops/A3F8F97TMOROPI>

FR Amazon Store Homepage:

<https://www.amazon.fr/shops/A3F8F97TMOROPI>

IT Amazon Store Homepage:

<https://www.amazon.it/shops/A3F8F97TMOROPI>

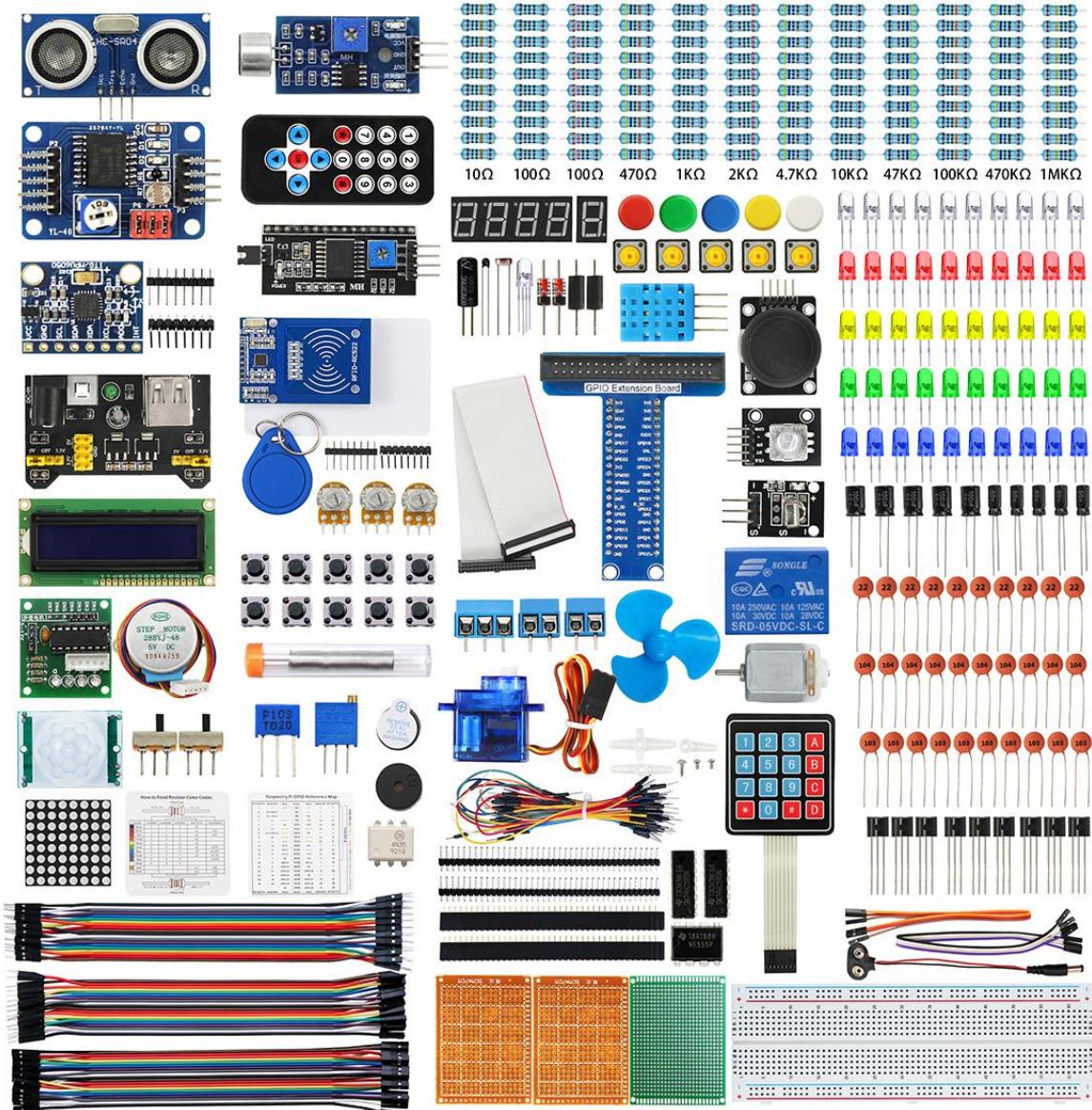
ES Amazon Store Homepage:

<https://www.amazon.es/shops/A3F8F97TMOROPI>

JP Amazon Store Homepage:

<https://www.amazon.co.jp/shops/A1F5OUAXY2TP0K>

WayinTop Ultimate Starter Kit für Raspberry Pi 4 B 3 B+



Vorwort

Dieses Tutorial geignet für das vollständigste ultimative Starter Kit für Raspberry Pi. Wenn Sie unsere C- und Python Tutorials gelernt haben oder grundlegende elektronische Schaltkreise und Programmierung gelernt haben, können Sie mit dem Erlernen dieses Tutorials beginnen. Andernfalls empfehlen wir Ihnen, zuerst unsere C- und Python Tutorials zu lernen. Die Skizze dieses Tutorials wurde in der Java-Sprache in der Verarbeitungssoftware geschrieben. Dieses Tutorial enthält ähnliche Projekte wie C- und Python Tutorials. Eine grafische Mensch-Maschine-Oberfläche wird hinzugefügt, um eine perfekte Integration von elektronischen Schaltkreisen, Computersoftware, Bildern usw. zu erreichen, sodass die Leser den Spaß am Programmieren und Heimwerken voll erleben können.

In diesem Tutorial wird die Installation und Verwendung von Verarbeitungssoftware auf Raspberry Pi in einigen elektronischen Schaltungsprojekten vorgestellt. Kapitel und Sequenz ähnelt dem C- und Python-Tutorial.

Content

Lektion 0 Installieren Processing Software.....	- 5 -
Lektion 1 LED.....	- 8 -
Lektion 2 Maussteuerungs-LED.....	- 14 -
Lektion 3 LED Bar Graph.....	- 16 -
Lektion 4 PWM.....	- 20 -
Lektion 5 RGBLED.....	- 26 -
Lektion 6 Aktiver Summer.....	- 34 -
Lektion 7 PCF8591.....	- 38 -
Lektion 8 ADDA&LED.....	- 43 -
Lektion 9 Fotowiderstand.....	- 47 -
Lektion 10 Thermistor.....	- 51 -
Lektion 11 74HC595 & LED.....	- 56 -
Lektion 12 74HC595 & Sieben-Segment-Anzeige.....	- 62 -
Lektion 13 74HC595 & 4-Bits Sieben-Segment-Anzeige.....	- 68 -
Lektion 14 74HC595 & LED Matrix.....	- 74 -
Lektion 15 I2C-LCD1602.....	- 82 -
Lektion 16 Joystick.....	- 89 -
Lektion 17 Relay & Motor.....	- 94 -
Lektion 18 Schrittmotor.....	- 100 -
Lektion 19 Matrixtastatur.....	- 108 -
Lektion 20 Oszilloskop.....	- 114 -
Lektion 21 Kontroll Grafiken.....	- 121 -
Lektion 22 PingPong Spiel.....	- 126 -
Lektion 23 Schlangenspiel.....	- 133 -
Lektion 24 Tetris Game.....	- 139 -

Lektion 0 Installieren Processing Software

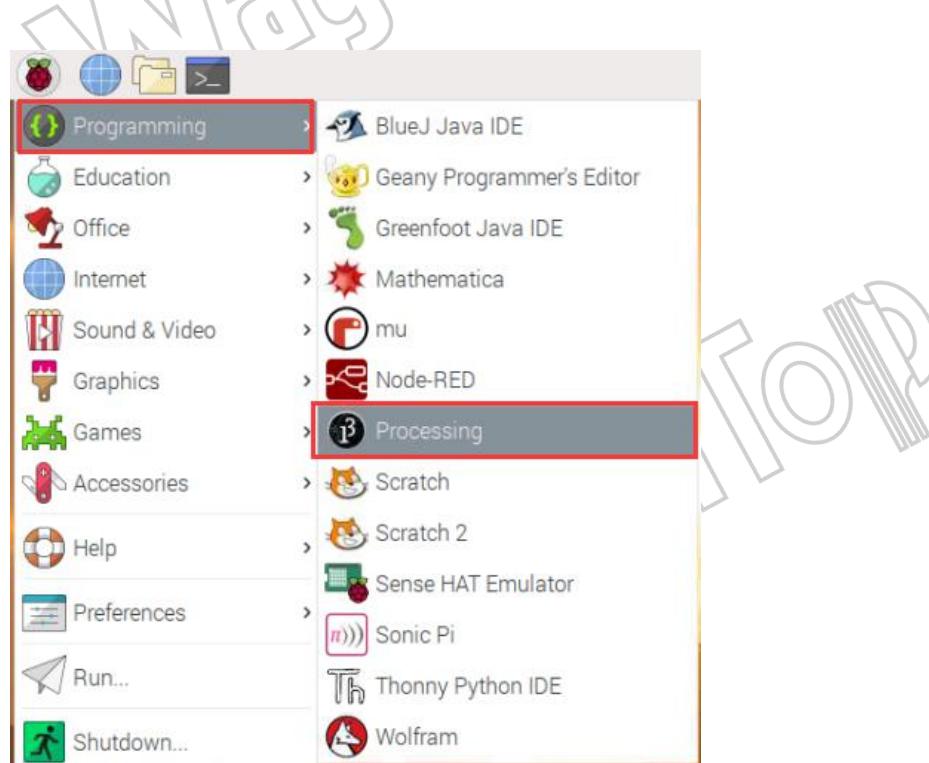
Wir müssen zuerst die Programmiersoftware 'Processing' auf dem Raspberry Pi installieren. Diese Software macht die Programmierung sehr einfach und die Installationsmethode ist auch sehr einfach.

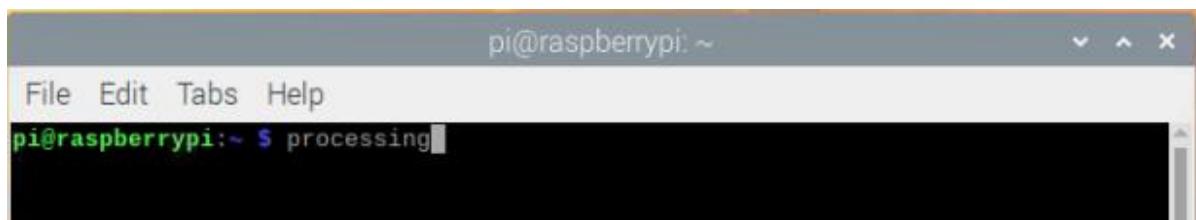
Wie installiere ich es?

Geben Sie im Raspberry Pi Terminal den Befehl 'curl <https://processing.org/download/install-arm.sh> | sudo sh'

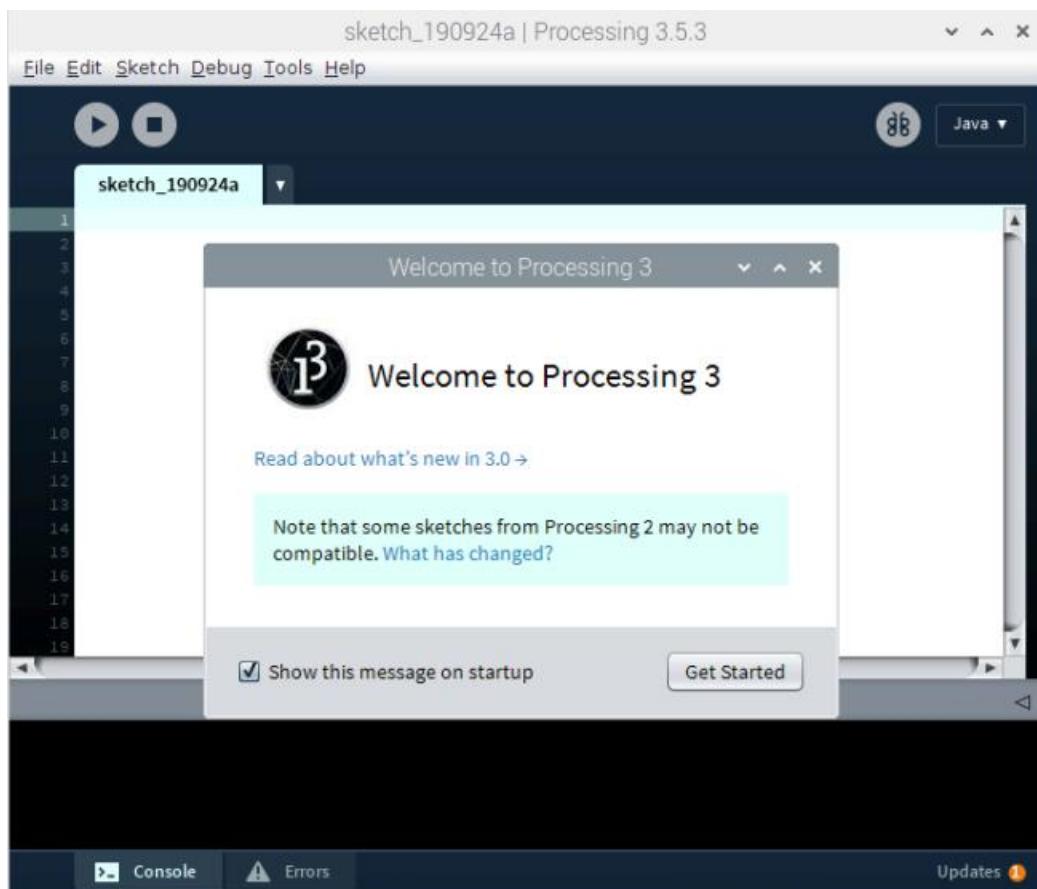
Stellen Sie sicher, dass Ihr RPi während des Installationsvorgangs immer über das Internet verfügt. Sie können die Software auch herunterladen und installieren, indem Sie die offizielle Website besuchen: <https://processing.org/>.

Nach Abschluss der Installation können Sie die "processing" eingeben, um mit dieser Software in einem beliebigen Verzeichnis des Raspberry Pi Terminals zu beginnen, oder sie im Startmenü des Systems öffnen, wie unten gezeigt:





Die Oberfläche der Verarbeitungssoftware ist unten dargestellt:



Geben Sie im Editor "Ellipse (50, 50, 90, 90)" ein. Diese Codezeile bedeutet "Zeichnen Sie eine Ellipse mit einer Mitte von 50 Pixel von links und 50 Pixel von oben nach unten mit einer Breite und Höhe von 80 Pixel. " Klicken Sie auf die Schaltfläche Ausführen (die Dreieckschaltfläche in der Symbolleiste). Wie nachfolgend dargestellt:



Wenn Sie alles richtig eingegeben haben, wird auf Ihrem Bildschirm ein Kreis angezeigt.



Klicken Sie im Anzeigefenster auf "Stop" (die Rechteckschaltfläche in der Symbolleiste) oder auf "Close", um die Ausführung des Programms zu beenden.

Wenn Sie es nicht richtig eingegeben haben, wird der Nachrichtenbereich rot und beschwert sich über einen Fehler. Stellen Sie in diesem Fall sicher, dass Sie den Beispielcode genau kopiert haben: Die Zahlen sollten in Klammern stehen und zwischen ihnen Kommas stehen, und die Zeile muss mit einem Semikolon enden.



Sie können diese Skizze in eine Anwendung exportieren, um sie direkt auszuführen, ohne die "processing" zu öffnen. Um die Skizze in die Anwendung zu exportieren, müssen Sie sie zuerst speichern.

Bisher haben wir das Studium der Installation und Verwendung der "processing" software abgeschlossen und dann begonnen, die Software für die Erstellung von Projekten zu verwenden.

Lektion 1 LED

Überblick

In dieser Lektion lernen Sie, wie Sie LED blinken und einige häufig verwendete Funktionen von Verarbeitungssoftware verwenden. In diesem Projekt werden wir eine Blink-LED erstellen. Gleichzeitig wird ein simuliertes LED-Blinkfenster auf dem Bildschirm angezeigt.

Erforderliche Teile:

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

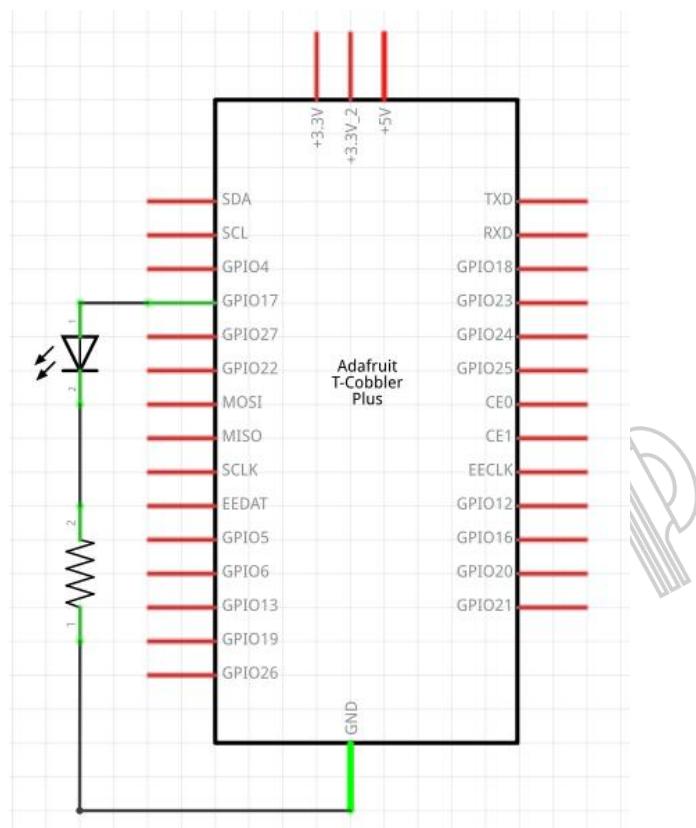
1 x Steckbrett

1 x LED

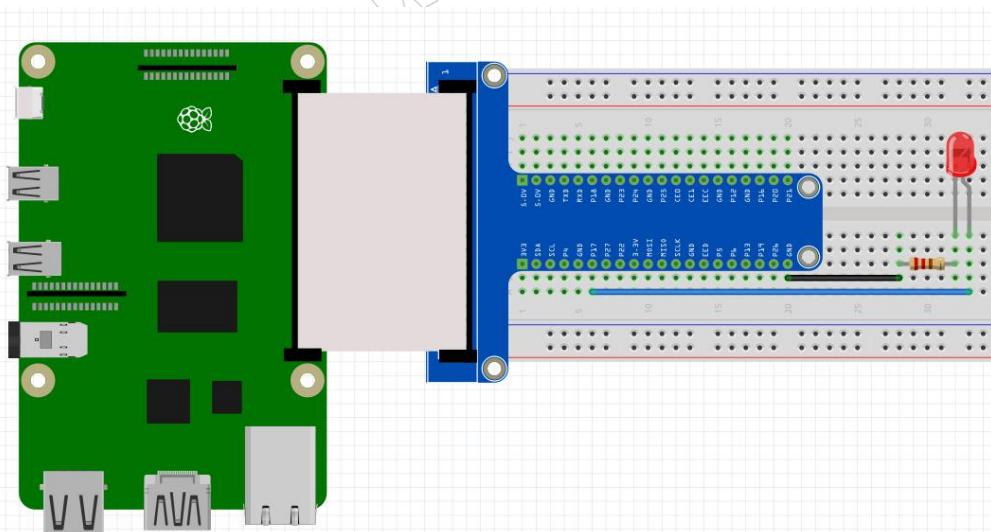
1 x 220 Ohm Widerstand

Einige Jumper Drähte

Verbindung Schema



Schaltplan

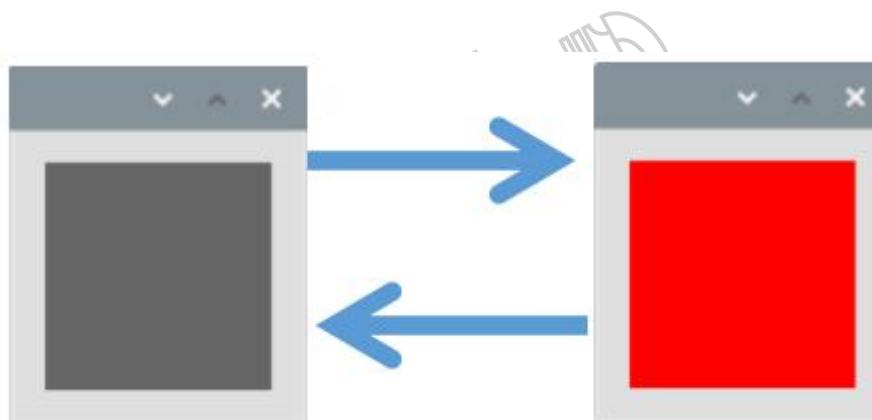


Da die Nummerierung des GPIO Extension Shield mit der des RPi GPIO identisch ist, zeigt das letztere Hardware-Verbindungsdiagramm nur den Teil des Steckbretts und des GPIO Extension Shield.

Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 1.LED / LED / LED.pde](#) ein, um den Code zu öffnen.
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen.
3. Nachdem das Programm ausgeführt wurde, beginnt die LED zu blinken und der Hintergrund des Anzeigefensters ändert sich mit der Änderung des LED-Status. Wie nachfolgend dargestellt:



Folgendes ist Programmcode:

```
import processing.io.*;  
  
int ledPin = 17;      //define ledPin  
boolean ledState = false;    //define ledState  
  
void setup() {  
    size(100, 100);  
    frameRate(1);          //set frame rate  
    GPIO.pinMode(ledPin, GPIO.OUTPUT);    //set the ledPin to output mode  
}  
  
void draw() {  
    ledState = !ledState;  
  
    if (ledState) {  
        GPIO.digitalWrite(ledPin, GPIO.HIGH);    //led on  
    } else {  
        GPIO.digitalWrite(ledPin, GPIO.LOW);    //led off  
    }  
}
```

```
    background(255, 0, 0); //set the fill color of led on
} else {
    GPIO.digitalWrite(ledPin, GPIO.LOW);      //led off
    background(102); //set the fill color of led off
}
}
```

Code Interpretation

```
void setup() {
    size(100, 100);
    frameRate(1);           //set frame rate
    GPIO.pinMode(ledPin, GPIO.OUTPUT); //set the ledPin to output mode
}
```

Processing codes haben normalerweise zwei Funktionen: "setup ()" und "draw ()", wobei die Funktion "setup ()" nur einmal ausgeführt wird, die Funktion "draw ()" jedoch in einer Schleife ausgeführt wird. In der Funktion "setup ()" gibt "size (100, 100)" die Größe des Anzeigefensters auf "100x100pixel" an. FrameRate (1) gibt die Aktualisierungsrate des Anzeigefensters auf einmal pro Sekunde an, dh die Funktion "draw ()" wird einmal pro Sekunde ausgeführt. Mit "GPIO.pinMode(ledPin , GPIO.OUTPUT)" wird "ledPin" in den Ausgabemodus versetzt.

```
void draw() {
    ledState = !ledState;
    if (ledState) {
        GPIO.digitalWrite(ledPin, GPIO.HIGH); //led on
        background(255, 0, 0); //set the fill color of led on
    } else {
        GPIO.digitalWrite(ledPin, GPIO.LOW); //led off
        background(102); //set the fill color of led off
    }
}
```

In der Funktion "draw ()" wird bei jeder Ausführung die Variable "ledState" invertiert. Wenn "ledState" "true" ist, leuchtet die LED und die Hintergrundfarbe des Anzeigefensters wird auf rot gesetzt. Und wenn der "ledState" "false" ist, wird die LED ausgeschaltet und die Hintergrundfarbe des Anzeigefensters auf grau gesetzt. Da die Funktion draw () einmal pro Sekunde ausgeführt wird, ändern sich auch die Hintergrundfarbe des Anzeigefensters und der Status der LED einmal pro Sekunde. Eine solche Schleife wiederholt sich, um den Effekt des Blinkens zu erzielen.

Funktion Einführung

“Setup ()” : Die Funktion setup () wird beim Programmstart einmal ausgeführt.

“Draw ()” : Die Funktion draw () wird direkt nach setup () aufgerufen und führt die in ihrem Block enthaltenen Codezeilen kontinuierlich aus, bis das Programm gestoppt oder noLoop () aufgerufen wird. draw () wird automatisch aufgerufen.

“Size ()”: Definieren Sie die Abmessung der Breite und Höhe des Anzeigefensters in Pixeleinheiten.

“Framerate ()” : Geben Sie die Anzahl der Frames an, die jede Sekunde angezeigt werden sollen.

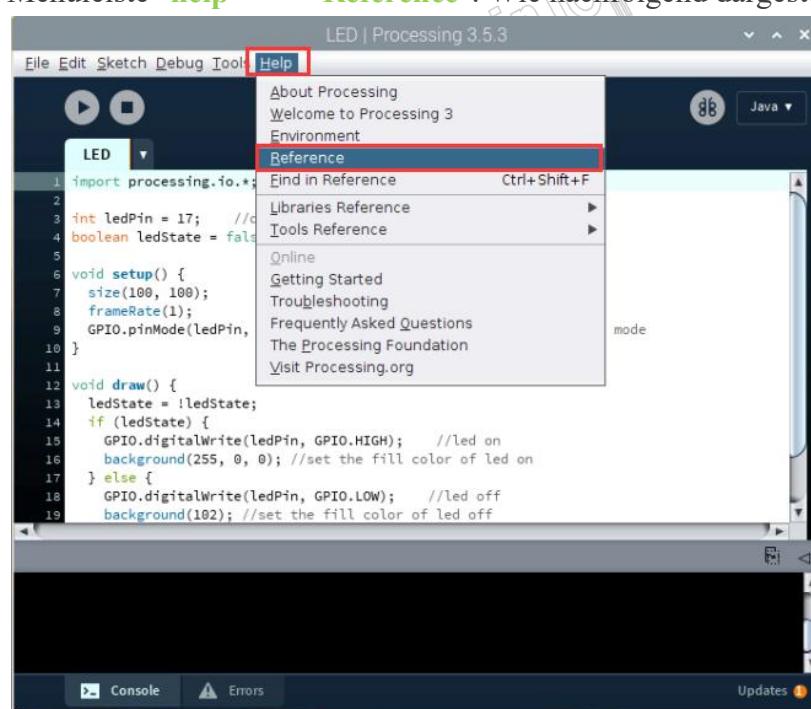
“Hintergrund ()” : Legen Sie die Farbe fest, die für den Hintergrund des Verarbeitungsfensters verwendet wird.

“Hintergrund ()”: Stellen Sie die Farbe ein, die für den Hintergrund des Processing Fensters verwendet wird.

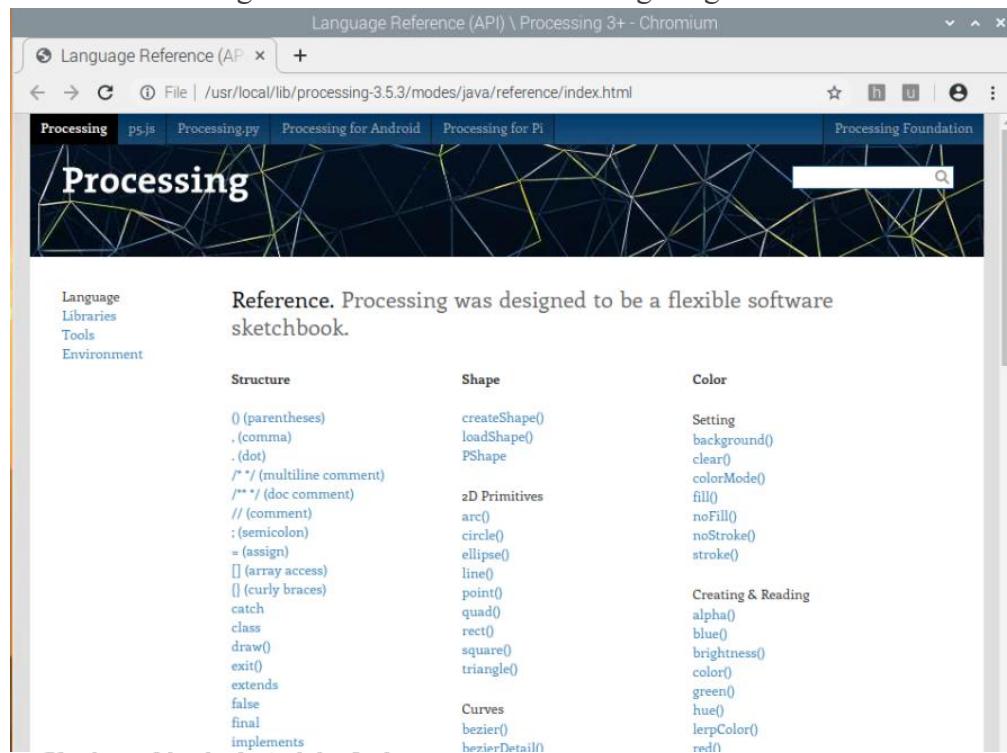
“GPIO.pinMode ()” : Konfigurieren Sie einen Pin als Eingang oder Ausgang.

„GPIO.digitalWrite () “: Stellen Sie einen Ausgangspin auf High oder Low ein.

Alle in diesem Code verwendeten Funktionen finden Sie in der Referenz der Verarbeitungssoftware, in der die integrierten Funktionen ausführlich beschrieben werden, und es gibt einige Beispielprogramme. Anfängern wird empfohlen, mehr Verwendung und Funktionen der Funktion anzusehen. Die Lokalisierung der Referenz kann durch die folgenden Schritte geöffnet werden: Klicken Sie auf die Menüleiste "help" -> “Reference”. Wie nachfolgend dargestellt:



Dann wird die folgende Seite im Webbrowser angezeigt:



The screenshot shows a browser window titled "Language Reference (API) \ Processing 3+ - Chromium". The main content is the Processing Language Reference, which includes a navigation bar with tabs for "Processing", "ps.js", "Processing.py", "Processing for Android", and "Processing for Pi". Below the navigation bar is a search bar. The main area displays the Processing reference documentation, which is organized into sections: "Reference", "Structure", "Shape", "Color", and "Creating & Reading". The "Reference" section states: "Processing was designed to be a flexible software sketchbook.". The "Structure" section lists various language constructs like parentheses, commas, dots, comments, semicolons, assignment, arrays, braces, catch, class, draw, exit, extends, false, final, and implements. The "Shape" section lists 2D primitives: createShape(), loadShape(), PShape, arc(), circle(), ellipse(), line(), point(), quad(), rect(), square(), and triangle(). The "Color" section lists settings: background(), clear(), colorMode(), fill(), noFill(), noStroke(), and stroke(). The "Creating & Reading" section lists alpha(), blue(), brightness(), color(), green(), hue(), lerpColor(), and red().

Oder greifen Sie direkt auf die offizielle Webseite zu: <http://processing.org/reference/>

Lektion 2 Maussteuerungs-LED

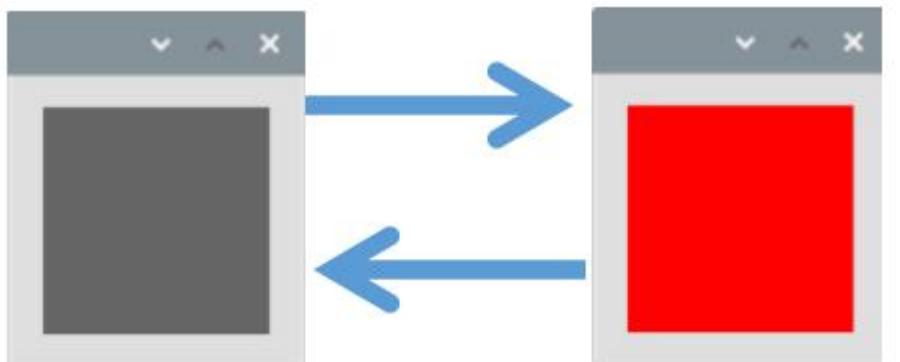
Überblick

In dieser Lektion ändern wir den Code basierend auf der Hardware des LED-Kurses und steuern den Status der LED mit der Maus.

Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 2.MouseLED / MouseLED / MouseLED.pde](#) ein, um den Code zu öffnen.
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen.
3. Nachdem das Programm ausgeführt wurde, ist die LED ausgeschaltet und die Hintergrundfarbe des Anzeigefensters ist grau. Klicken Sie mit der Maus auf das Anzeigefenster, dann wird die LED eingeschaltet und die Hintergrundfarbe des Anzeigefenster wird rot. Klicken Sie erneut auf das Anzeigefenster, dann wird die LED ausgeschaltet und die Hintergrundfarbe wird grau, wie unten gezeigt.



Das Folgende ist der Code:

```
import processing.io.*;  
  
int ledPin = 17;  
boolean ledState = false;  
void setup() {  
    size(100, 100);  
    GPIO.pinMode(ledPin, GPIO.OUTPUT);  
    background(102);
```

```
}
```

```
void draw() {if (ledState) {

    GPIO.digitalWrite(ledPin, GPIO.HIGH);
    background(255,0,0);
} else {
    GPIO.digitalWrite(ledPin, GPIO.LOW);
    background(102);
}
}

void mouseClicked() { //if the mouse Clicked
    ledState = !ledState; //Change the led State
}
```

Code Interpretation

```
void mouseClicked() { //if the mouse Clicked
    ledState = !ledState; //Change the led State
}
```

In diesem Code wird die Funktion "mouseClicked ()" verwendet. Die Funktion wird verwendet, um die Mausklickereignisse zu erfassen, die ausgeführt werden, wenn die Maus angeklickt wird. In dieser Funktion können wir den Status der Variablen „ledState“ ändern, um die Steuerung der LED durch Klicken mit der Maus zu realisieren.



Lektion 3 LED Bar Graph

Überblick

In this Lektion, we will use the mouse to control the LED Bar Graph.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

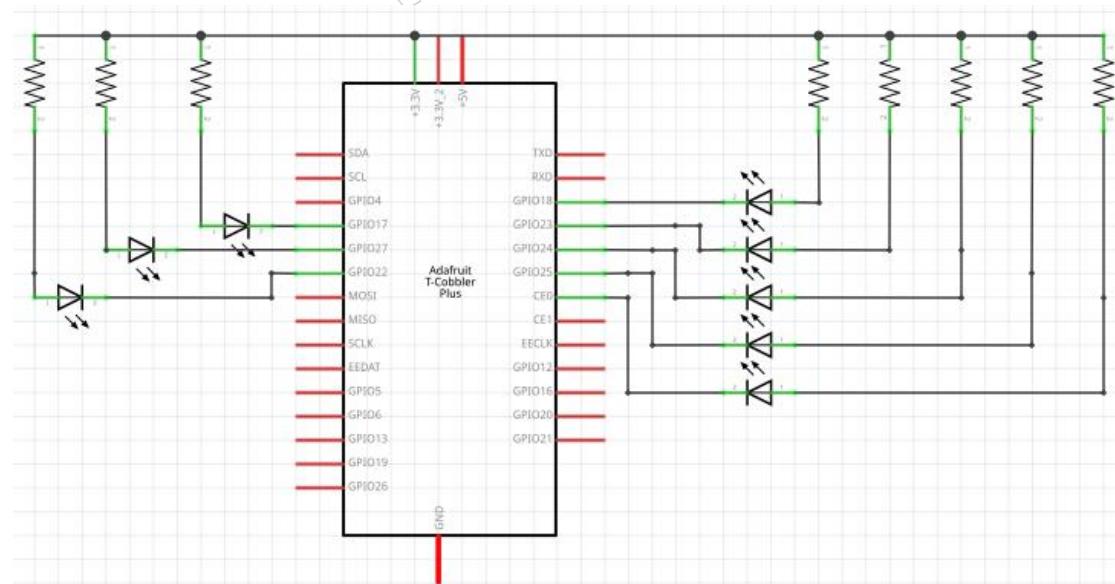
1 x Steckbrett

8 x LED

8 x 220 Ohm Widerstand

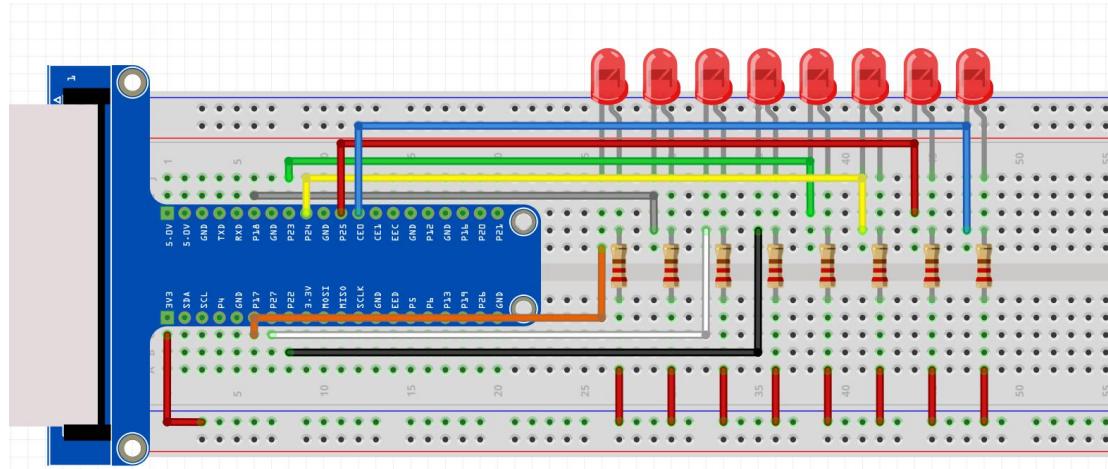
Einige Jumper Drähte

Schaltplan





Verdrahtung Diagramm

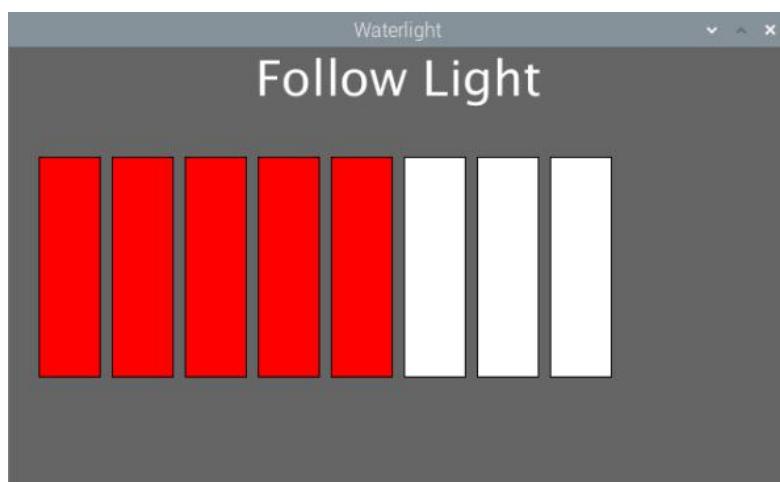


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 3.Waterlight / Waterlight / Waterlight.pde](#) ein, um den Code zu öffnen.
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen.

Nachdem das Programm ausgeführt wurde, ist die LED ausgeschaltet und die Hintergrundfarbe des Anzeigefensters ist grau. Klicken Sie mit der Maus auf das Anzeigefenster, dann wird die LED eingeschaltet und die Hintergrundfarbe des Anzeigefensters wird rot. Klicken Sie erneut auf das Anzeigefenster, dann wird die LED ausgeschaltet und die Hintergrundfarbe wird grau, wie unten gezeigt.



Das Folgende ist der Programmcode:

```
import processing.io.*;  
  
int leds[] = {17, 18, 27, 22, 23, 24, 25, 8};  
  
void setup() {  
    size(640, 360);  
    for (int i=0; i<8; i++) {  
        GPIO.pinMode(leds[i], GPIO.OUTPUT);  
    }  
    background(102);  
    textAlign(CENTER);  
    textSize(40);  
    text("Follow Light", width / 2, 40);  
    textSize(16);  
    text("www.keeyees.com", width / 2, height - 20);  
}  
  
void draw() {  
  
    for (int i=0; i<8; i++) {  
        if (mouseX > (25+60*i)) {  
            fill(255, 0, 0);  
  
            GPIO.digitalWrite(leds[i], GPIO.LOW);  
        } else {  
            fill(255, 255, 255);  
            GPIO.digitalWrite(leds[i], GPIO.HIGH);  
        }  
    }  
}
```

```
        }
        rect(25+60*i, 90, 50, 180);
    }
}
```

Code Interpretation

```
void draw() {
    for (int i=0; i<8; i++) {
        if (mouseX>(25+60*i)) {
            fill(255, 0, 0);
            GPIO.digitalWrite(leds[i], GPIO.LOW);
        } else {
            fill(255, 255, 255);

            GPIO.digitalWrite(leds[i], GPIO.HIGH);
        }
        rect(25+60*i, 90, 50, 180);
    }
}
```

In der Funktion "draw ()" zeichnen wir 8 Rechtecke, um 8 LEDs des LED-Balkendiagramms darzustellen. Wir machen Rechtecke auf der linken Seite der Maus, die entsprechenden LEDs mit roten gefüllt sind. Auf diese Weise, wenn Sie die Maus nach rechts schieben; Je mehr LEDs links von der Maus leuchten. Wenn links, ist das Gegenteil der Fall.

Lektion 4 PWM

Überblick

In dieser Lektion lernen wir, wie man eine „atmende LED“ erstellt, und das Anzeigefenster zeigt ein LED-Farbverlaufsmuster an. Wie steuere ich die Helligkeit der LED? Wir werden PWM verwenden, um dieses Ziel zu erreichen.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

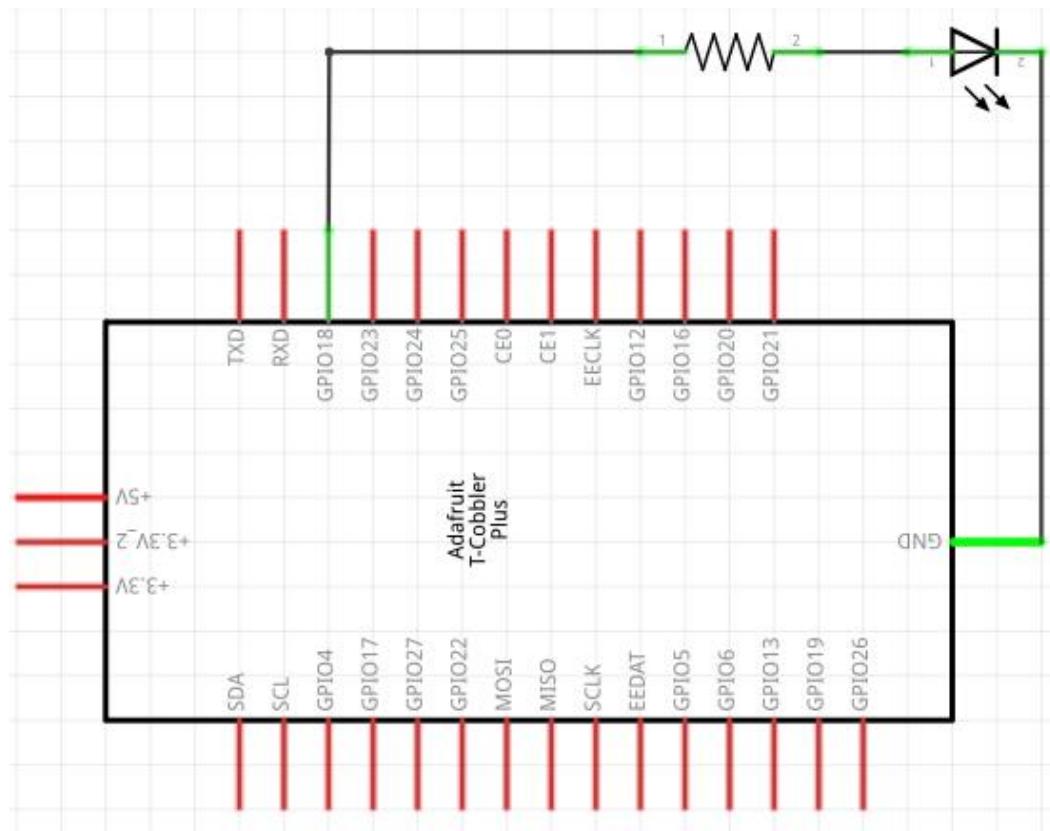
1 x LED

1 x 220 Ohm Widerstand

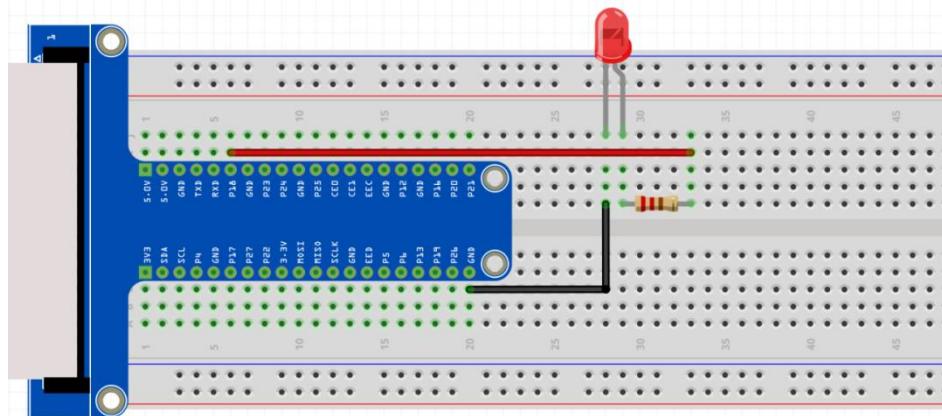
Einige Jumper Drähte



Schaltplan



Verdrahtung Diagramm

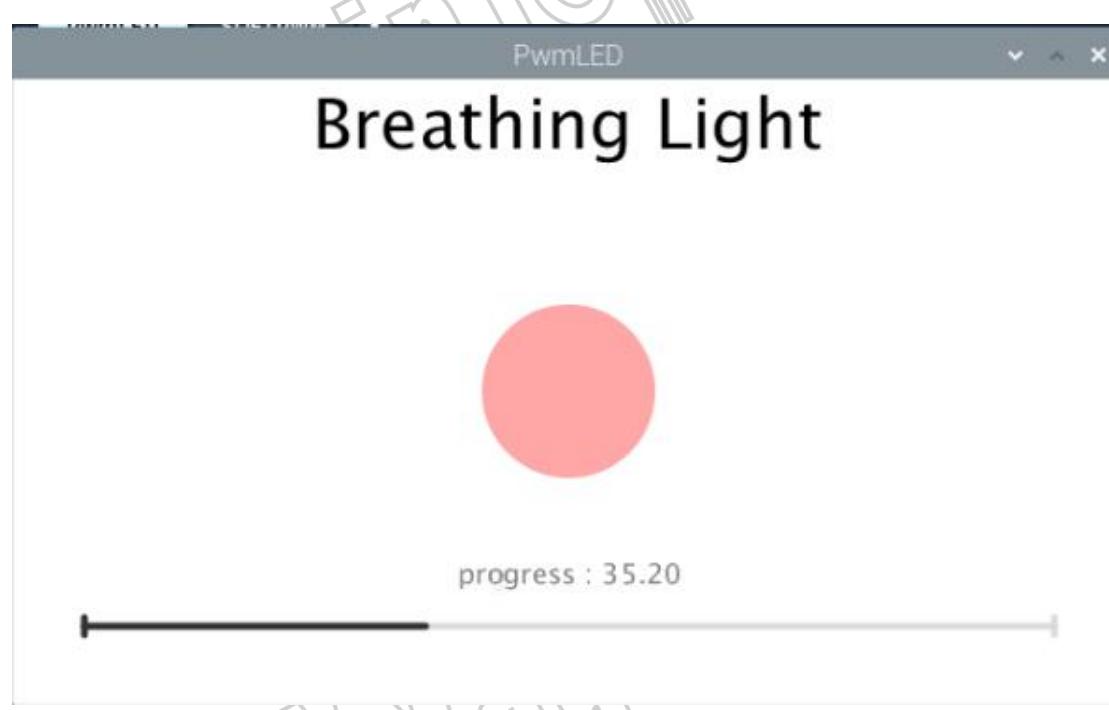


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 4.PWMLED / PwmLED / PwmLED.pde](#) ein, um den Code zu öffnen.
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, wird die LED im Schaltkreis allmählich aufgehellt und gleichzeitig die Farbe des LED-Musters im Anzeigefenster allmählich vertieft. Der Fortschrittsbalken unter dem Muster zeigt den Fertigstellungsgrad an, und durch Klicken mit der Maus auf die Oberfläche des Fensters kann der Fortschritt geändert werden. Wie nachfolgend dargestellt:



Das Folgende ist der Code:

```
import processing.io.*;  
  
int ledPin = 18;  
int borderSize = 40;  
float t = 0.0;  
float tStep = 0.004;  
SOFTPWM p = new SOFTPWM(ledPin, 10, 100);
```

```
void setup() {  
  
    size(640, 360);  
    strokeWeight(4);  
}  
  
void draw() {  
    if (mousePressed) {  
        int a = constrain(mouseX, borderSize, width - borderSize);  
        t = map(a, borderSize, width - borderSize, 0.0, 1.0);  
    } else {  
        t += tStep;  
        if (t > 1.0) t = 0.0;  
    }  
    p.softPwmWrite((int)(t*100));  
    background(255);  
    titleAndSiteInfo();  
  
    fill(255, 255-t*255, 255-t*255);  
    ellipse(width/2, height/2, 100, 100);  
  
    pushMatrix();  
    translate(borderSize, height - 45);  
    int barLength = width - 2*borderSize;  
  
    barBgStyle();  
    line(0, 0, barLength, 0);  
    line(barLength, -5, barLength, 5);  
  
    barStyle();  
    line(0, -5, 0, 5);  
    line(0, 0, t*barLength, 0);  
  
    barLabelStyle();  
    text("progress : "+nf(t*100,2,2),barLength/2,-25);  
    popMatrix();  
}  
  
void titleAndSiteInfo() {  
    fill(0);  
    textAlign(CENTER);  
    textSize(40);  
    text("Breathing Light", width / 2, 40);
```

```
textSize(16);

text("www.keeyees.com", width / 2, height - 20);
}

void barBgStyle() {
    stroke(220);
    noFill();
}

void barStyle() {
    stroke(50);
    noFill();
}

void barLabelStyle() {
    noStroke();
    fill(120);
}
```

Code Interpretation

```
float t = 0.0;
float tStep = 0.004;
SOFTPWM p = new SOFTPWM(ledPin, 10, 100);
```

Verwenden Sie zunächst SOFTPWM, um einen PWM Pin zu erstellen, mit dem die Helligkeit der LED gesteuert wird. Definieren Sie dann eine Variable "t" und eine Variable "tStep", um den PWM-Arbeitszyklus und die Selbstbeschleunigungsrate zu steuern.

```
if (mousePressed) {
    int a = constrain(mouseX, borderSize, width - borderSize);
    t = map(a, borderSize, width - borderSize, 0.0, 1.0);
} else {
    t += tStep;
    if (t > 1.0) t = 0.0;
}
```

Bei der Funktionszeichnung wird bei einem Klick die Koordinate in X-Richtung der Maus auf das Tastverhältnis "t" abgebildet, andernfalls wird das Tastverhältnis "t" schrittweise erhöht. Geben Sie dann PWM mit dem Arbeitszyklus aus.

```
fill(255, 255-t*255, 255-t*255);
ellipse(width/2, height/2, 100, 100);
```

Der nächste Code dient zum Zeichnen von kreisförmigen Farben mit unterschiedlicher Tiefe gemäß dem Wert „t“, mit dem die LED mit unterschiedlicher Helligkeit simuliert wird.

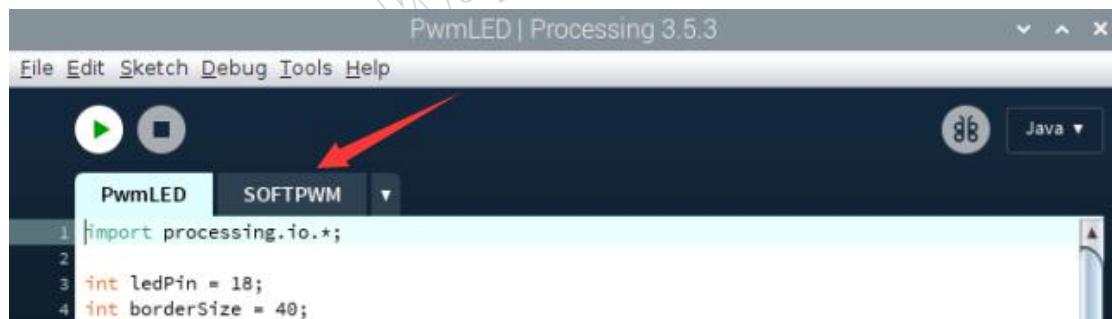
```
barBgStyle();
line(0, 0, barLength, 0);
line(barLength, -5, barLength, 5);

barStyle();
line(0, -5, 0, 5);
line(0, 0, t*barLength, 0);

barLabelStyle();
text("progress : "+nf(t*100,2,2),barLength/2,-25);
```

Der letzte Code dient zum Zeichnen des Fortschrittsbalkens und des Prozentsatzes des Fortschritts.

In der Verarbeitungssoftware wird zusätzlich zum obigen Code eine Tag-Seite "SOFTPWM" angezeigt.



Die Datei enthält einige Informationen zum SOFTPWM.

```
public SOFTPWM(int iPin, int dc, int pwmRange):
```

Es wird verwendet, um einen PWM-Pin zu erstellen, den pwmRange und den anfänglichen Arbeitszyklus festzulegen. Das Minimum von pwmRange beträgt 0,1 ms. PwmRange = 100 bedeutet also, dass der PWM-Zyklus $0,1\text{ ms} * 100 = 10\text{ ms}$ beträgt.

```
public SOFTPWM(int iPin, int dc, int pwmRange):
```

Stellen Sie den PMW-Arbeitszyklus ein.

```
public void softPwmStop()
```

Stoppen Sie die PWM-Ausgabe.

Lektion 5 RGBLED

Überblick

In dieser Lektion lernen Sie, wie Sie RGB LED verwenden und eine bunte LED erstellen. Verwenden Sie also die Verarbeitung, um die Farbe der RGB-LED zu steuern.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

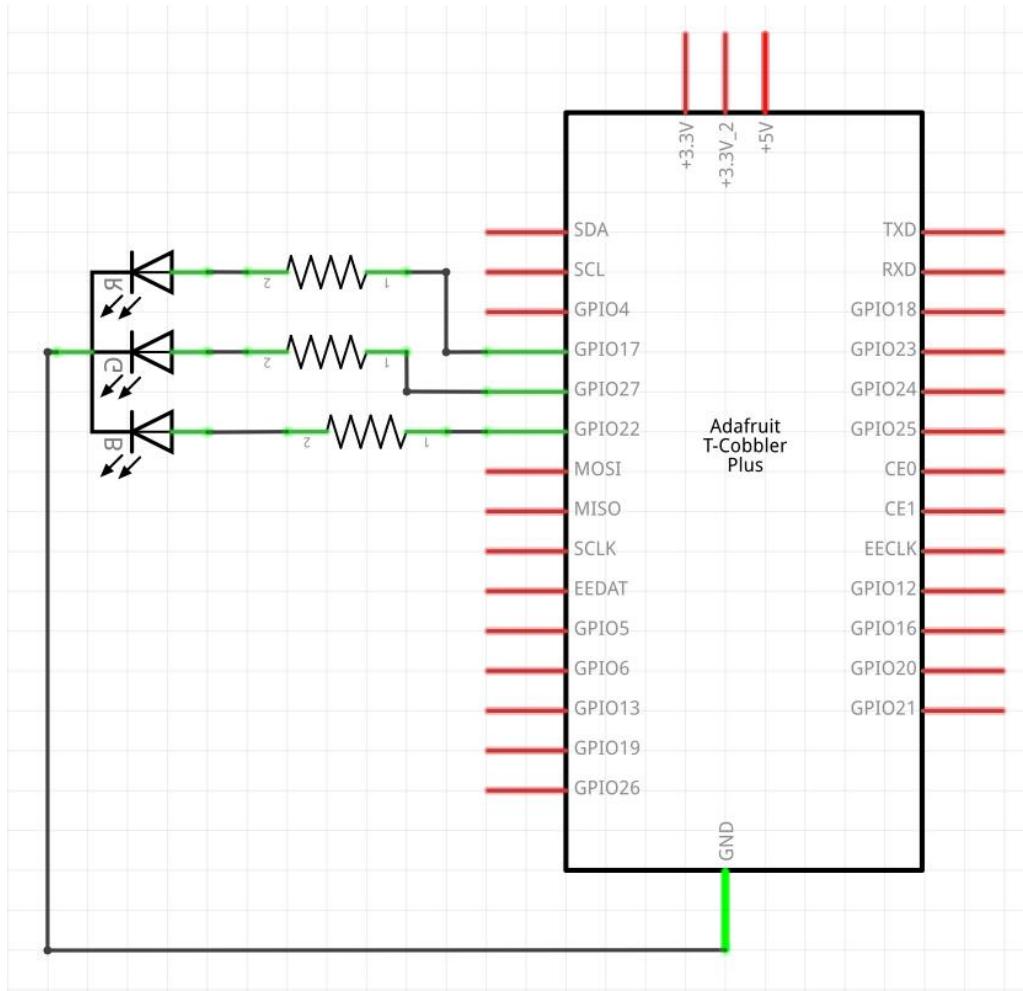
1 x RGBLED

3 x 220 Ohm Widerstände

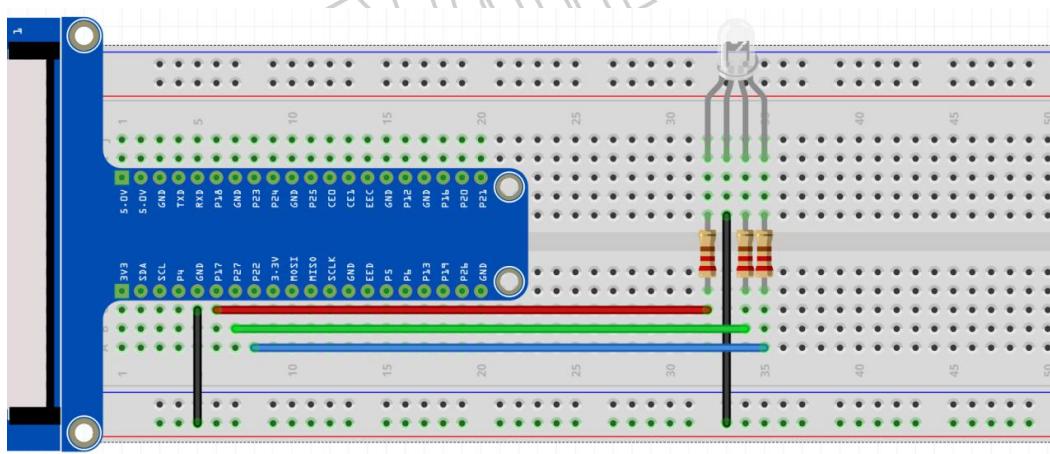
Einige Jumper Drähte



Schaltplan



Verdrahtung Diagramm

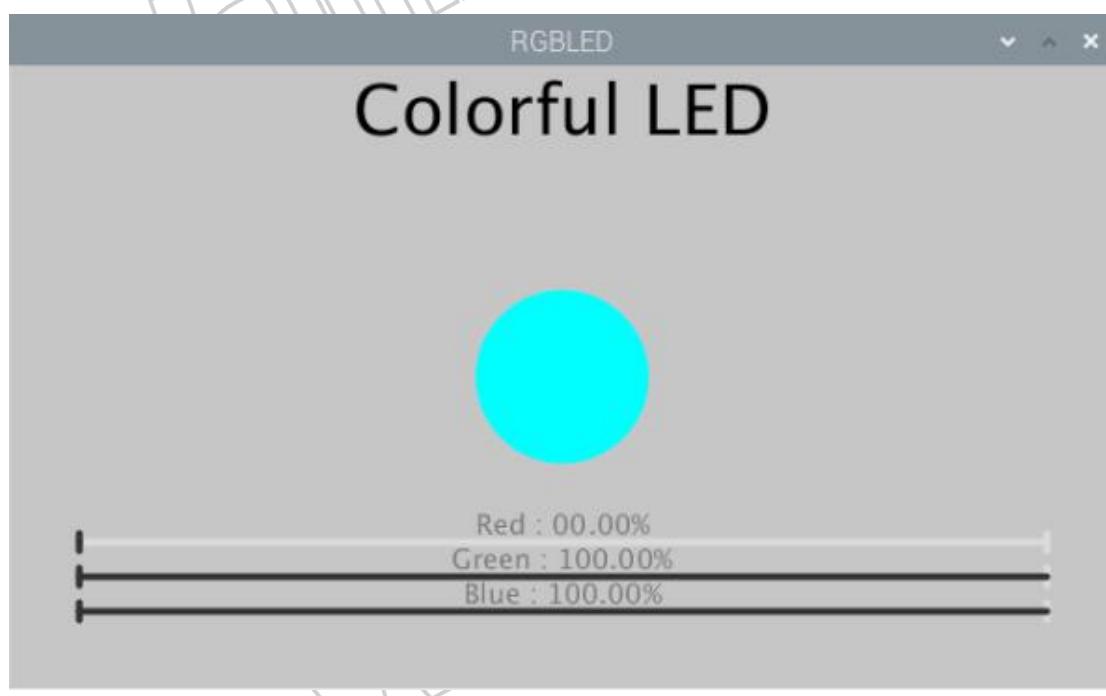


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 3.Waterlight / Waterlight / Waterlight.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, ist die RGB-LED ausgeschaltet. Im Anzeigefenster ist das zur Simulation der LED verwendete Muster schwarz. Rot, Grün und Blau sind in Null. Wenn Sie mit der Maus auf einen Fortschrittsbalken klicken und diesen ziehen, können Sie den PWM-Arbeitszyklus der Farbkanäle einstellen. Die in der Schaltung verwendete RGB-LED zeigt dann die entsprechende Farbe an. Gleichzeitig zeigt das Muster im Anzeigefenster dieselbe Farbe. Wie nachfolgend dargestellt:



Dieses Projekt enthält viele Codedateien, und der Kerncode ist in der Datei „RGB LED“ enthalten. Die anderen Dateien enthalten nur einige benutzerdefinierte Inhalte.



Das Folgende ist der Code:

```
import processing.io.*;  
  
int bluePin = 17; //blue Pin  
int greenPin = 27; //green Pin  
int redPin = 22; //red Pin  
int borderSize = 40; //picture border size  
  
SOFTPWM pRed = new SOFTPWM(redPin, 100, 100);  
SOFTPWM pGreen = new SOFTPWM(greenPin, 100, 100);  
SOFTPWM pBlue = new SOFTPWM(bluePin, 100, 100);  
  
ProgressBar rBar, gBar, bBar;  
boolean rMouse = true, gMouse = true, bMouse = true;  
void setup() {  
    size(640, 360); //display window size  
    strokeWeight(4); //stroke Weight  
    int barLength = width - 2*borderSize;  
    rBar = new ProgressBar(borderSize, height - 85, barLength);  
    gBar = new ProgressBar(borderSize, height - 65, barLength);  
    bBar = new ProgressBar(borderSize, height - 45, barLength);  
    //Set ProgressBar's title  
    rBar.setTitle("Red");gBar.setTitle("Green");bBar.setTitle("Blue");  
}  
  
void draw() {  
    background(200);  
    titleAndSiteInfo();
```

```
fill(rBar.progress*255, gBar.progress*255, bBar.progress*255);

ellipse(width/2, height/2, 100, 100);
rBar.create();
gBar.create();
bBar.create();
}

void mousePressed() {
if ( (mouseY< rBar.y+5) && (mouseY>rBar.y-5) ) {
    rMouse = true;
} else if ( (mouseY< gBar.y+5) && (mouseY>gBar.y-5) ) {
    gMouse = true;
} else if ( (mouseY< bBar.y+5) && (mouseY>bBar.y-5) ) {
    bMouse = true;
}
}

void mouseReleased() {
rMouse = false;
bMouse = false;
gMouse = false;
}

void mouseDragged() {
int a = constrain(mouseX, borderSize, width - borderSize);
float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
if (rMouse) {
    pRed.softPwmWrite((int)(100-t*100));
    rBar.setProgress(t);
} else if (gMouse) {
    pGreen.softPwmWrite((int)(100-t*100));
    gBar.setProgress(t);
} else if (bMouse) {
    pBlue.softPwmWrite((int)(100-t*100));
    bBar.setProgress(t);
}
}

void titleAndSiteInfo() {
fill(0);
textAlign(CENTER);
textSize(40);          //set text size
text("Colorful LED", width / 2, 40);      //title
textSize(16);
```

{}

Code Interpretation

```
SOFTPWM pRed = new SOFTPWM(redPin, 100, 100);
SOFTPWM pGreen = new SOFTPWM(greenPin, 100, 100);
SOFTPWM pBlue = new SOFTPWM(bluePin, 100, 100);
```

```
ProgressBar rBar, gBar, bBar;
```

Erstellen Sie im Code zunächst drei PWM-Pins und drei Fortschrittsbalken, um die RGB-LED zu steuern.

```
void setup() {
    size(640, 360); //display window size
    strokeWeight(4); //stroke Weight
    int barLength = width - 2*borderSize;
    rBar = new ProgressBar(borderSize, height - 85, barLength);
    gBar = new ProgressBar(borderSize, height - 65, barLength);
    bBar = new ProgressBar(borderSize, height - 45, barLength);
    //Set ProgressBar's title
    rBar.setTitle("Red");gBar.setTitle("Green");bBar.setTitle("Blue");
}
```

Definieren Sie dann in function "setup ()" Position und Länge des Fortschrittsbalkens entsprechend der Größe des Anzeigefensters und legen Sie den Namen jedes Fortschrittsbalkens fest.

```
void draw() {
    background(200);
    titleAndSiteInfo();

    fill(rBar.progress*255, gBar.progress*255, bBar.progress*255);
    ellipse(width/2, height/2, 100, 100);

    rBar.create();
    gBar.create();
    bBar.create();
}
```

Stellen Sie in der Funktion "draw ()" zuerst Hintergrund, Header und andere grundlegende Informationen ein. Zeichnen Sie dann einen Kreis und stellen Sie seine Farbe entsprechend dem Arbeitszyklus von drei RGB-Kanälen ein. Zeichnen Sie abschließend drei Fortschrittsbalken.

```
void mousePressed() {  
    if ( (mouseY < rBar.y+5) && (mouseY > rBar.y-5) ) {  
        rMouse = true;  
    } else if ( (mouseY < gBar.y+5) && (mouseY > gBar.y-5) ) {  
        gMouse = true;  
    } else if ( (mouseY < bBar.y+5) && (mouseY > bBar.y-5) ) {  
        bMouse = true;  
    }  
}  
void mouseReleased() {  
    rMouse = false;  
    bMouse = false;  
    gMouse = false;  
}  
void mouseDragged() {  
    int a = constrain(mouseX, borderSize, width - borderSize);  
    float t = map(a, borderSize, width - borderSize, 0.0, 1.0);  
    if (rMouse) {  
        pRed.softPwmWrite((int)(100-t*100));  
        rBar.setProgress(t);  
    } else if (gMouse) {  
        pGreen.softPwmWrite((int)(100-t*100));  
        gBar.setProgress(t);  
    } else if (bMouse) {  
        pBlue.softPwmWrite((int)(100-t*100));  
        bBar.setProgress(t);  
    }  
}
```

Mit den Systemfunktionen `mousePressed ()`, `mouseReleased ()` und `mouseDragged ()` können Sie bestimmen, ob die Maus den Fortschrittsbalken zieht oder nicht, und den Zeitplan festlegen. Wenn die Maustaste in einem Fortschrittsbalken gedrückt wird, wird mit der Maustaste () der Fortschritt eines Flags * Mouse auf "true" gesetzt, `mouseDragged (mouseX)`, in dem gleichzeitig festgelegten Mapping-Fortschrittswert der Fortschritt des entsprechenden Zeitplans und PWM. Wenn die Maus losgelassen wird, leeren Sie alle Flags in `mouseReleased ()`.

Über die Klasse ProgressBar:

```
class ProgressBar
```

Dies ist eine benutzerdefinierte Klasse, mit der ein Fortschrittsbalken erstellt wird.

```
public ProgressBar(int ix, int iy, int barlen)
```

Die Konstruktionsfunktion verwendet "create ProgressBar", die Parameter für die Koordinaten X, Y und die Länge von "ProgressBar".

```
public void setTitle(String str)
```

Dient zum Festlegen des Namens des Fortschrittsbalkens, der in der Mitte des Fortschrittsbalkens angezeigt wird.

```
public void setProgress(float pgress)
```

Dient zum Festlegen des Fortschritts des Fortschrittsbalkens. Parameter: 0 <Fortschritt <1,0.

```
public void create() & public void create(float pgress)
```

Wird zum Zeichnen des Fortschrittsbalkens verwendet.

Lektion 6 Aktiver Summer

Überblick

In diesem Kurs lernen Sie den Umgang mit aktivem Summer.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

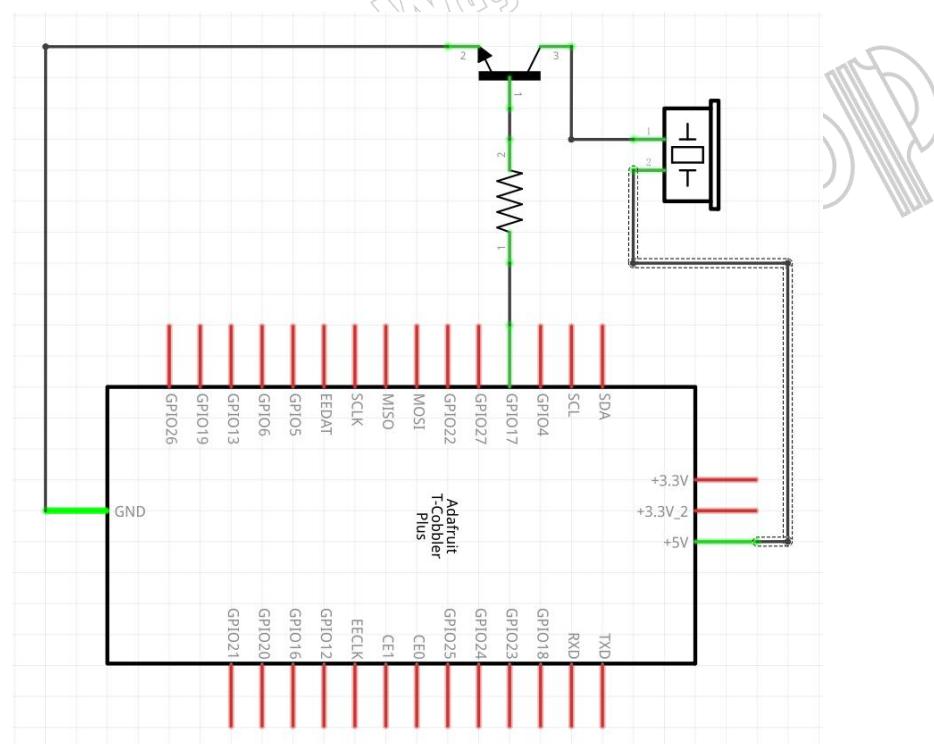
1 x Aktiver Summer

1 x 1K Ohm Widerstand

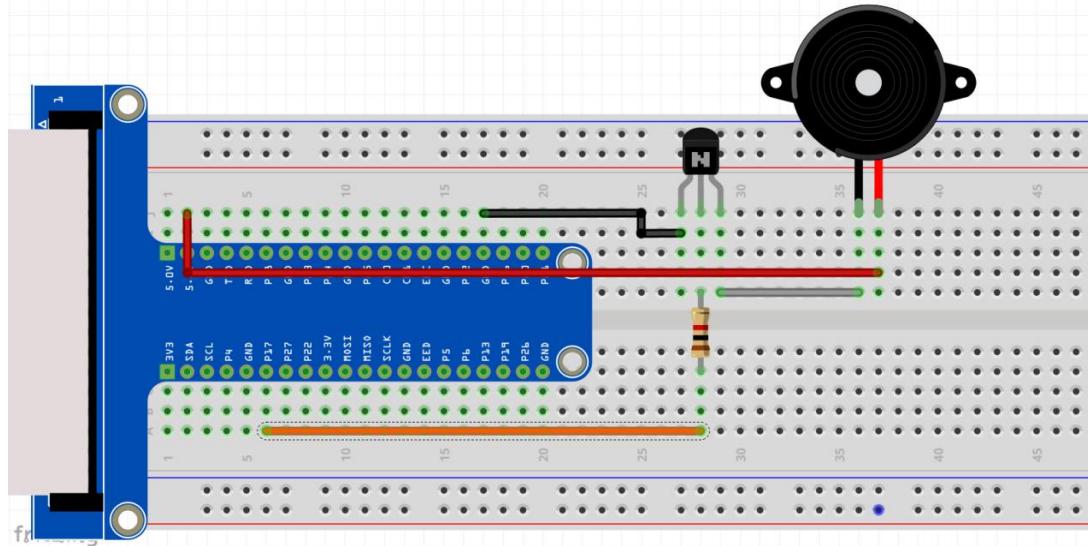
1 x S8050 Transistor

Einige Jumper Drähte

Schematische Darstellung



Schaltplan

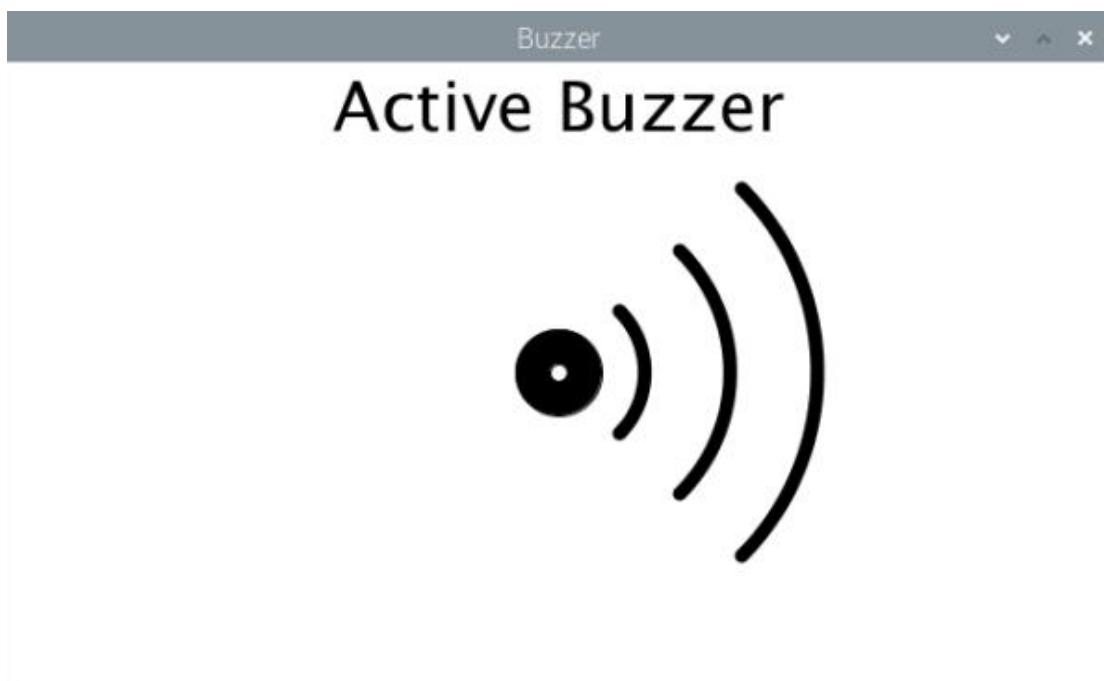


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal `processing code / Java / 6.Buzzer / Buzzer / Buzzer.pde` ein, um den Code zu öffnen;
 2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "**RUN**", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, klicken Sie mit der Maus auf eine beliebige Position des Anzeigefensters. Dann ertönt der aktive Summer und neben dem Summermuster im Anzeigefenster werden Bogengrafiken (Schema des Klangs) angezeigt. Klicken Sie erneut mit der Maus, und dann hört Active Buzzer auf zu ertönen und die Bogengrafiken verschwinden, wie nachfolgend dargestellt:



Das Folgende ist der Code:

```
import processing.io.*;  
  
int buzzerPin = 17;  
boolean buzzerState = false;  
void setup() {  
    size(640, 360);  
    GPIO.pinMode(buzzerPin, GPIO.OUTPUT);  
}  
  
void draw() {  
    background(255);  
    titleAndSiteInfo(); //title and site information  
    drawBuzzer(); //buzzer img  
    if (buzzerState) {  
        GPIO.digitalWrite(buzzerPin, GPIO.HIGH);  
        drawArc(); //Sounds waves img  
    } else {  
        GPIO.digitalWrite(buzzerPin, GPIO.LOW);  
    }  
}  
  
void mouseClicked() {
```

```
buzzerState = !buzzerState; //Change the buzzer State
}
void drawBuzzer() {
    strokeWeight(1);
    fill(0);
    ellipse(width/2, height/2, 50, 50);
    fill(255);
    ellipse(width/2, height/2, 10, 10);
}
void drawArc() {
    noFill();
    strokeWeight(8);
    for (int i=0; i<3; i++) {
        arc(width/2, height/2, 100*(1+i), 100*(1+i), -PI/4, PI/4, OPEN);
    }
}
void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("Active Buzzer", width / 2, 40); //title
    textSize(16);
}
```

Code Interpretation

Der Code in diesem Projekt basiert auf einer ähnlichen Logik wie in der vorherigen "Lektion 2 Maussteuerungs-LED". Und der Unterschied besteht darin, dass dieses Projekt das Summermuster und die Bogengrafiken zeichnen muss, nachdem der Summer ertönt ist.

Lektion 7 PCF8591

Überblick

In dieser Lektion lernen Sie, wie Sie das AD/DA PCF8591 Modul verwenden.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

1 x 220 Ohm Widerstand

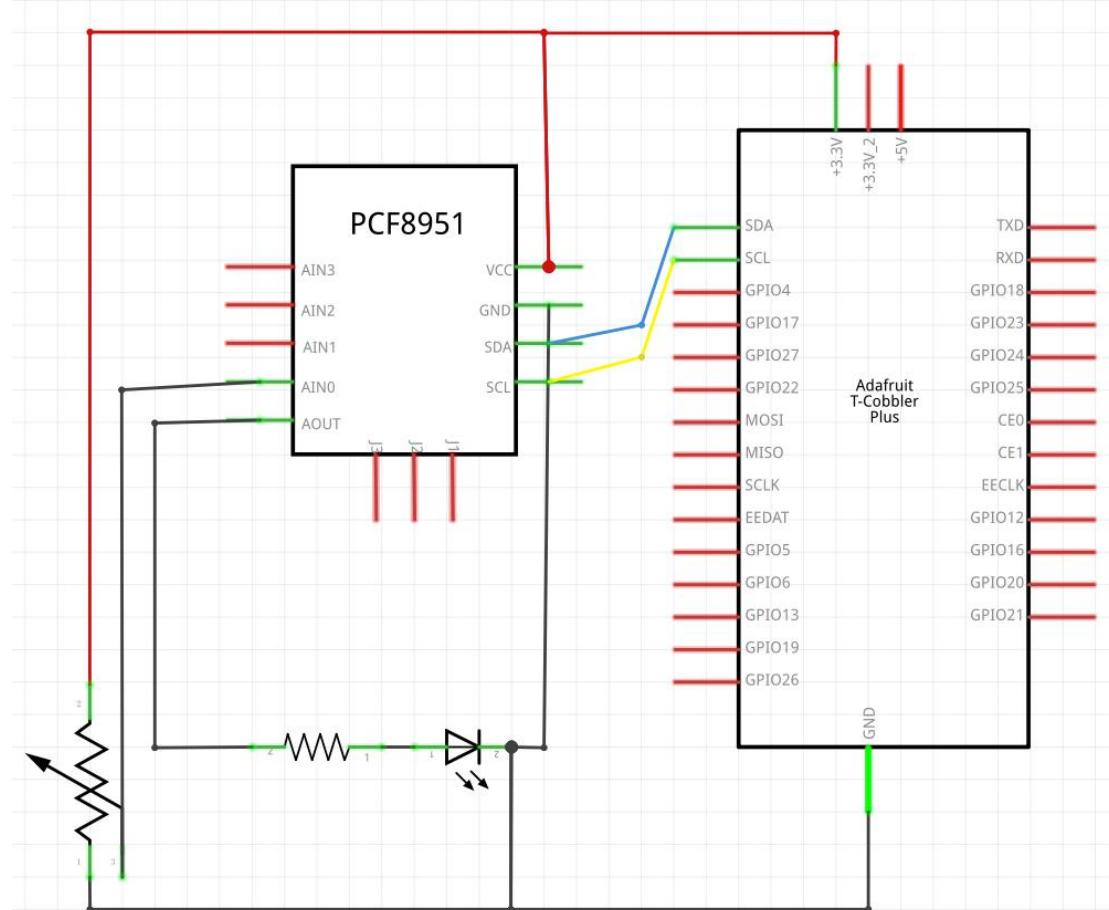
1 x PCF8591 Modul

1 x Potentiometer

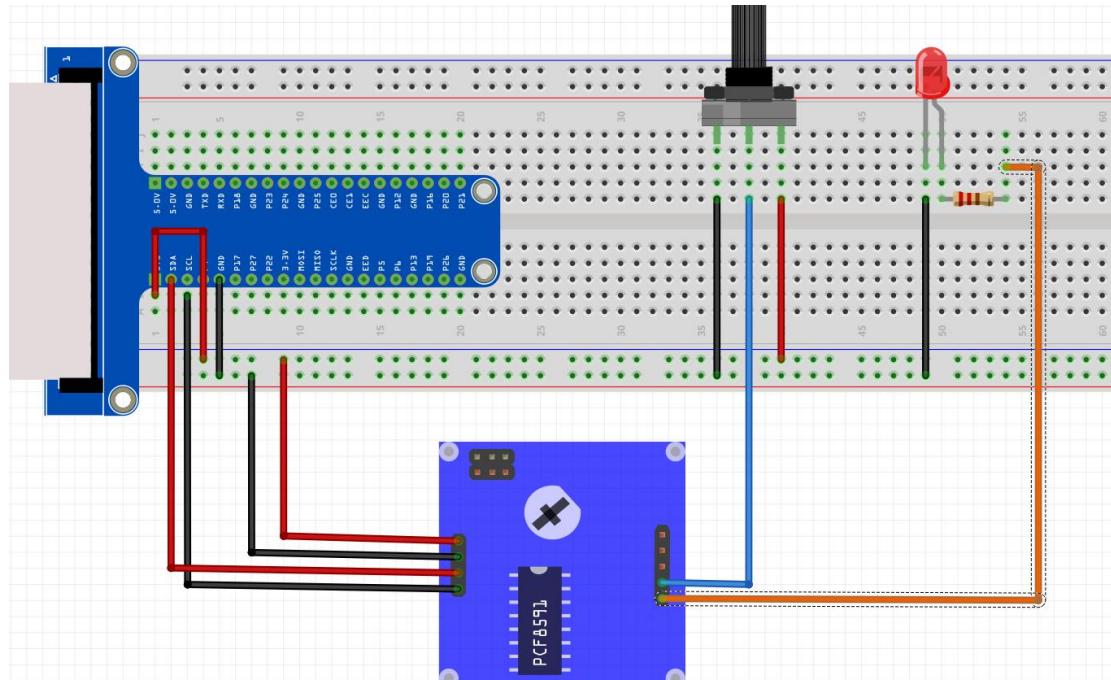
1 x LED

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

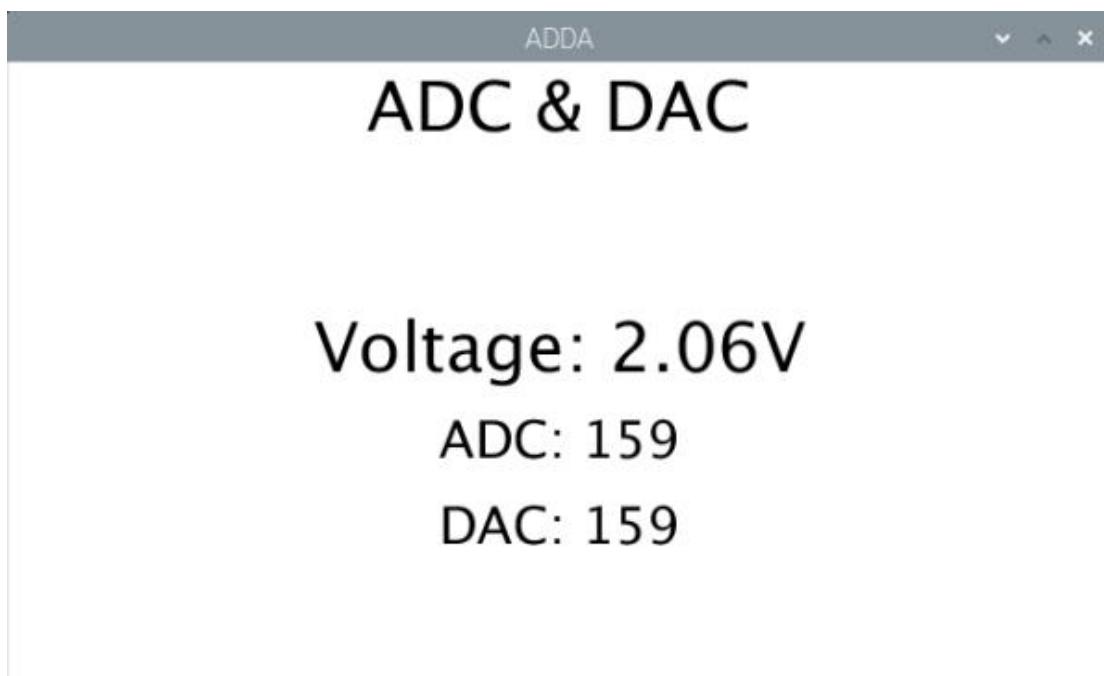


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 7.ADDA / ADDA / ADDA.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "**RUN**", um den Code auszuführen;

Nach Ausführung des Programms zeigt das Anzeigefenster den Spannungswert des Potentiometers, den ADC- und den DAC Wert an. Der ADC-Wert und der DAC-Wert sind gleich. Drehen Sie das Potentiometer, um den Spannungsausgang zu ändern. Da die Einschaltspannung für die rote LED etwa 1,6V beträgt, ist die LED ausgeschaltet, wenn die Spannung weniger als 1,6V beträgt. Wenn die Spannung größer als 1,6 V ist, leuchtet die LED. Erhöhen Sie den DAC-Wert weiter, und die LED-Helligkeit erhöht sich entsprechend.



Das Folgende ist der Code:

```
import processing.io.*;
PCF8591 pcf = new PCF8591(0x48);
void setup() {
    size(640, 360);
}
void draw() {
    int adc = pcf.analogRead(0),  
        //Read the ADC value of channel 0
    float volt = adc*3.3/255.0;  
        //calculate the voltage
    pcf.analogWrite(adc);  
        //Write the DAC
    background(255);
    titleAndSiteInfo();

    fill(0);
    textAlign(CENTER);      //set the text centered
    textSize(30);
    text("ADC: "+nf(adc, 3, 0), width / 2, height/2+50);
    textSize(30);
```

```
text("DAC: "+nf(adc, 3, 0), width / 2, height/2+100);
textSize(40);           //set text size
text("Voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2);
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);      //set the text centered
    textSize(40);           //set text size
    text("ADC & DAC", width / 2, 40);    //title
    textSize(16);

}
```

Code Interpretation

```
int adc = pcf.analogRead(0);
float volt = adc*3.3/255.0;
pcf.analogWrite(addr);
```

Der Projektcode verwendet hauptsächlich die Funktionen `analogRead()` und `analogWrite()`, um ADC zu lesen und DAC zu schreiben.

Über die Klasse PCF8591:

```
class PCF8591
```

Dies ist eine benutzerdefinierte Klasse, mit der ADC und DAC von PCF8591 betrieben werden.

```
public PCF8591(int addr)
```

Der Konstruktor, der zum Erstellen des PCF8591 Klassenobjekts verwendet wird, ist der Parameter der I2C PCF8591 Geräteadresse.

```
public int analogRead(int chn)
```

Zum Lesen des ADC Werts eines Kanals von PCF8591 verwendet, gibt der Parameter CHN die Kanalnummer an: 0,1,2,3.

```
public byte[] analogRead()
```

Lesen Sie die ADC Werte aller Kanäle von PCF8591.

```
public void analogWrite(int data)
```

Schreiben Sie einen DAC Wert in PCF8591.

Lektion 8 ADDA&LED

Überblick

In dieser Lektion lernen Sie, wie Sie ADC und PWM kombinieren, um die Helligkeit der LED zu steuern.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

1 x 220 Ohm Widerstand

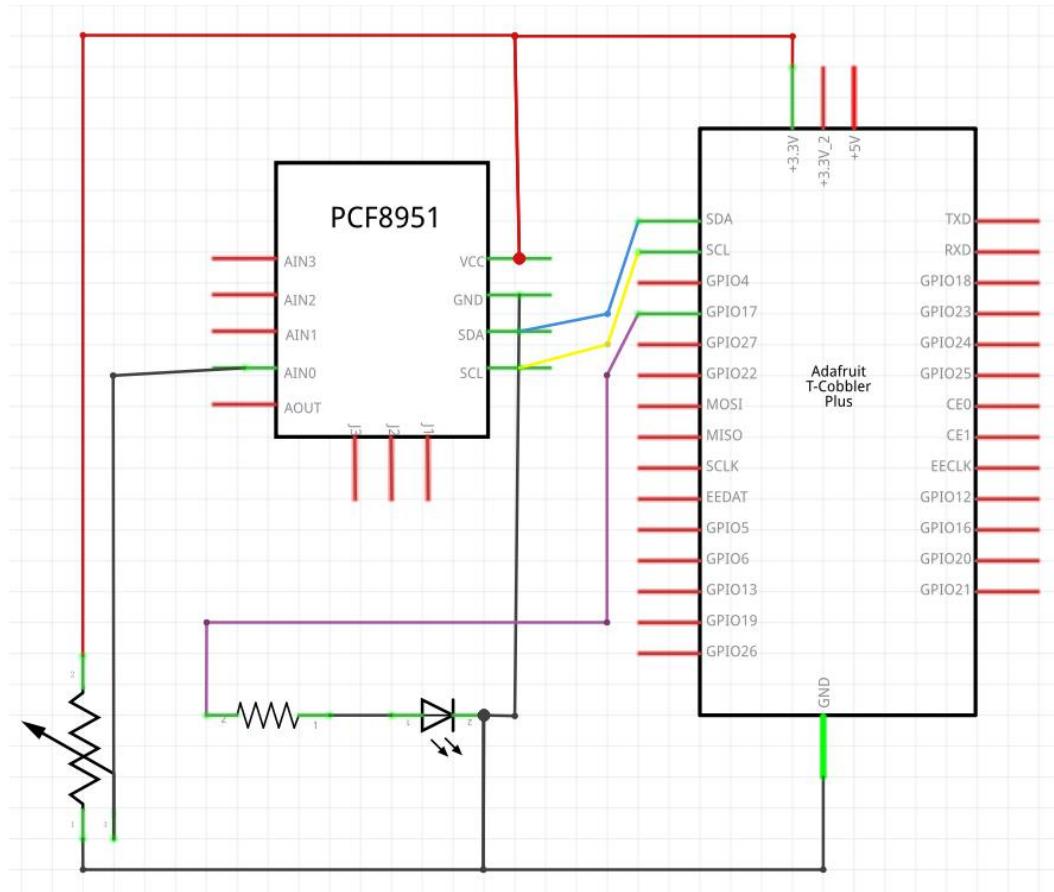
1 x PCF8591 Modul

1 x Potentiometer

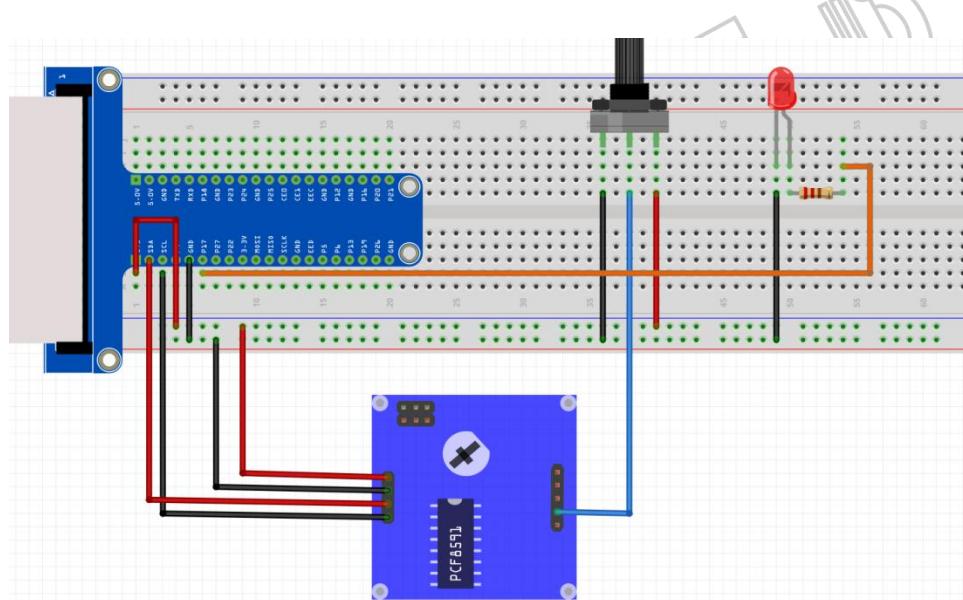
1 x LED

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagram

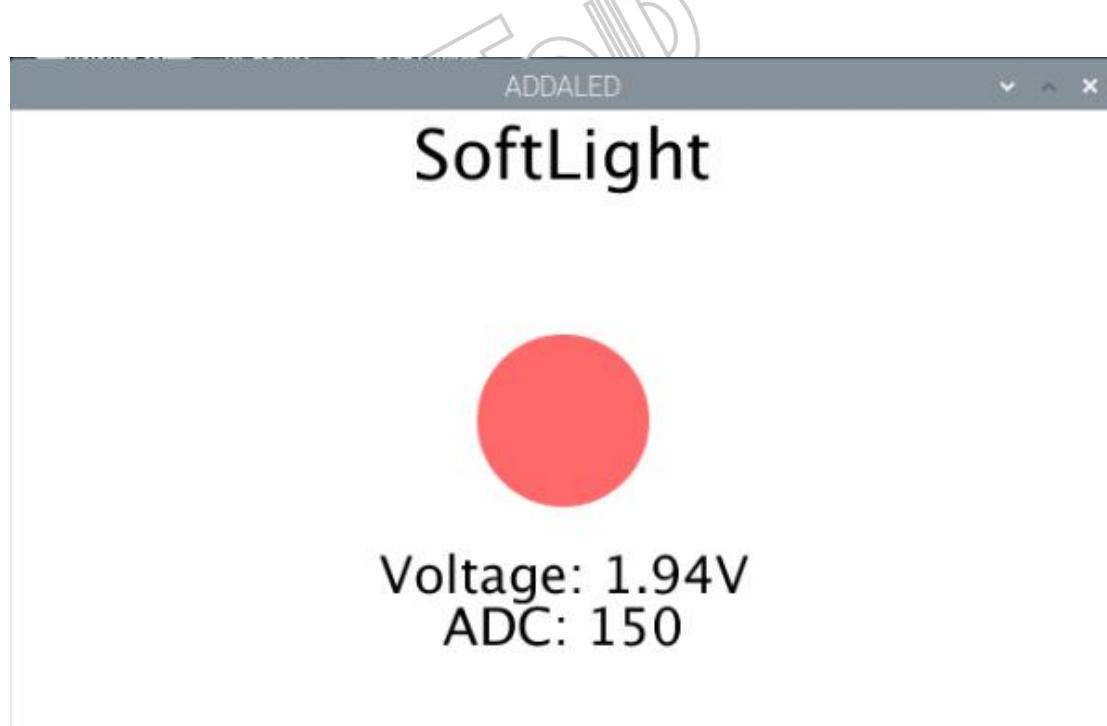


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 8.ADDALED / ADDALED.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nach Ausführung des Programms zeigt das Anzeigefenster den Spannungswert des Potentiometers, den ADC Wert und ein LED Muster an. Drehen Sie das Potentiometer, um den Spannungswert und die Helligkeit der LED zu ändern.



Das Folgende ist der Code:

```
import processing.io.*;  
  
int ledPin = 17;  
PCF8591 pcf = new PCF8591(0x48);  
SOFTPWM p = new SOFTPWM(ledPin, 0, 100);  
void setup() {  
    size(640, 360);  
}  
void draw() {
```

```
int adc = pcf.analogRead(0);      //Read the ADC value of channel 0

float volt = adc*3.3/255.0;      //calculate the voltage
float dt = adc/255.0;
p.softPwmWrite((int)(dt*100));   //output the pwm
background(255);
titleAndSiteInfo();

fill(255, 255-dt*255, 255-dt*255); //cycle
noStroke(); //no border
ellipse(width/2, height/2, 100, 100);

fill(0);
textAlign(CENTER); //set the text centered
textSize(30);
text("ADC: "+nfadc, 3, 0), width / 2, height/2+130);
text("Voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2+100);
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("SoftLight", width / 2, 40);
    textSize(16);

}
```

Code Interpretation

```
int adc = pcf.analogRead(0);
float volt = adc*3.3/255.0;
float dt = adc/255.0;
p.softPwmWrite((int)(dt*100));
```

Ermitteln Sie in diesem Projektcode den ADC-Wert des Potentiometers und ordnen Sie ihn dann dem PWM-Arbeitszyklus der LED zu, um deren Helligkeit zu steuern. Im Anzeigefenster ändert sich die im LED-Modus ausgefüllte Farbe zu simulieren Sie die Helligkeitsänderung der LED.

Lektion 9 Fotowiderstand

Überblick

In this course, you will learn how to use Photo Widerstand, used to intense ambient light, to make a Night Lamp. When the ambient light get dark, LED brightness will be enhanced automatically. Conversely, the LED brightness will be weakened automatically.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

1 x 10 K Widerstand

1 x 220 Ohm Widerstand

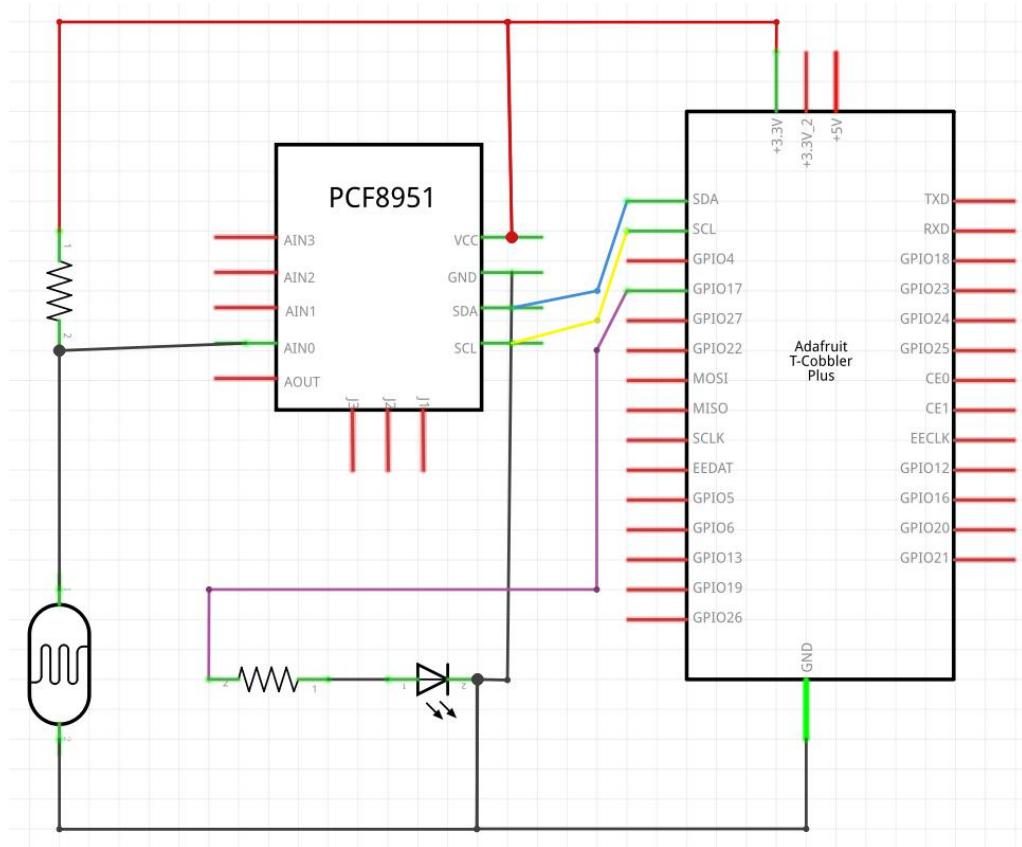
1 x PCF8591 Modul

1 x Photo Widerstand

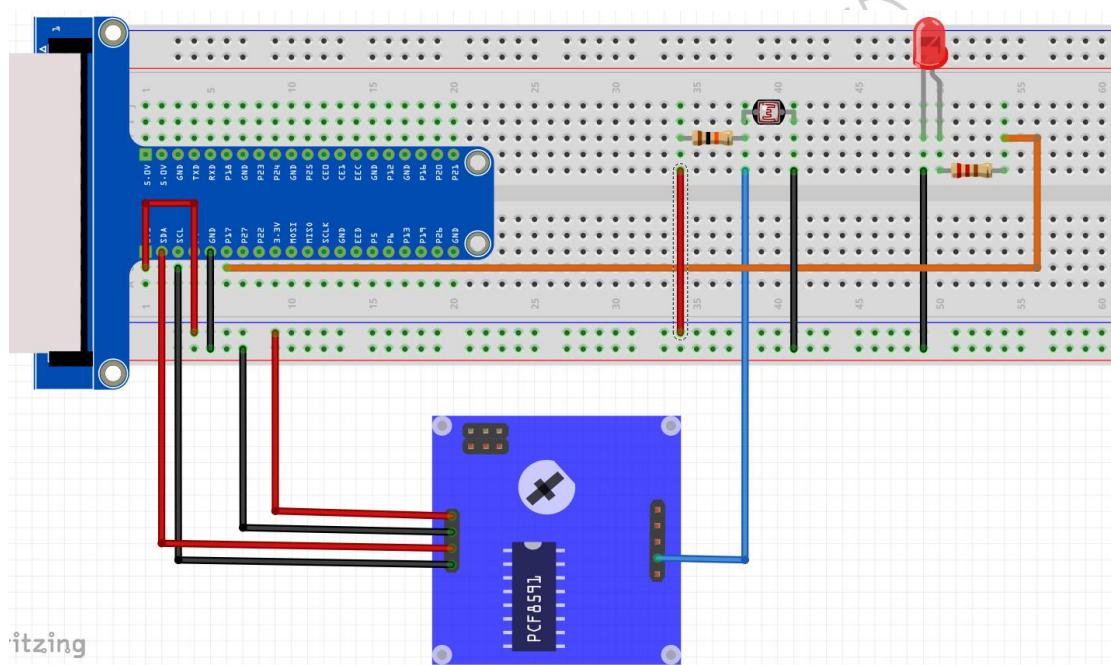
1 x LED

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagram

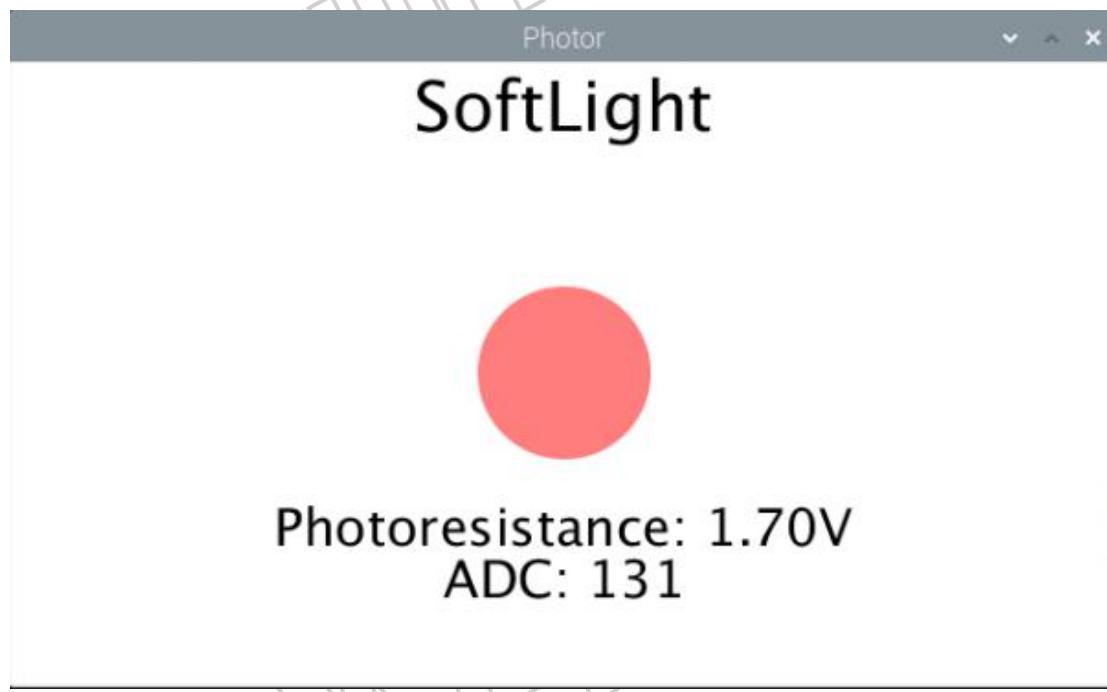


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 9.Photoresistance / Photor / Photor.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, zeigt das Anzeigefenster den Spannungswert des Fotowiderstands, den ADC Wert und ein LED Muster an. Beleuchten Sie den Fotowiderstand mit Licht, um den Spannungswert und die Helligkeit der LED zu ändern.



Das Folgende ist der Code:

```
import processing.io.*;  
  
int ledPin = 17;  
PCF8591 pcf = new PCF8591(0x48);  
SOFTPWM p = new SOFTPWM(ledPin, 0, 100);  
void setup() {
```

```

        size(640, 360);
    }
void draw() {
    int adc = pcf.analogRead(0);      //Read the ADC value of channel 0

    float volt = adc*3.3/255.0;      //calculate the voltage
    float dt = adc/255.0;

    p.softPwmWrite((int)(dt*100));   //output the pwm
    background(255);
    titleAndSiteInfo();

    fill(255, 255-dt*255, 255-dt*255); //cycle
    noStroke(); //no border
    ellipse(width/2, height/2, 100, 100);

    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(30);
    text("ADC: "+nf(adc, 3, 0), width / 2, height/2+130);
    text("Photoresistance: "+nf(volt, 0, 2)+"V", width / 2, height/2+100);
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("SoftLight", width / 2, 40);
    textSize(16);

}

```

Code Interpretation

```

int adc = pcf.analogRead(0);
float volt = adc*3.3/255.0;
float dt = adc/255.0;
p.softPwmWrite((int)(dt*100));

```

Der Projektcode entspricht dem Code "Lektion 8 ADDA&LED". Der einzige Unterschied besteht darin, dass das Eingangssignal des Pins "AIN0" des "PCF8591" von einem Potentiometer in eine Kombination aus einem Fotowiderstand und einem Widerstand geändert wird.

Lektion 10 Thermistor

Überblick

In dieser Lektion lernen wir, wie man mit einem Thermistor ein Thermometer herstellt.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

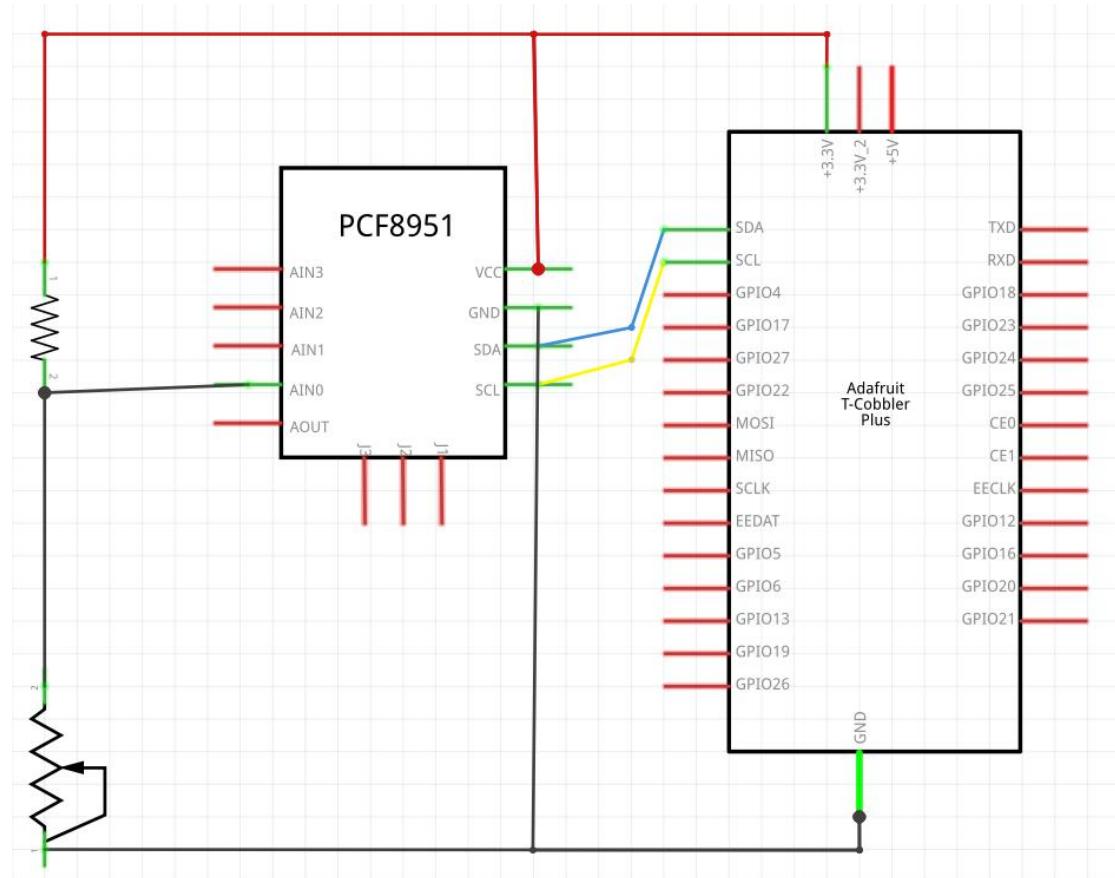
1 x 10K Widerstand

1 x PCF8591 Modul

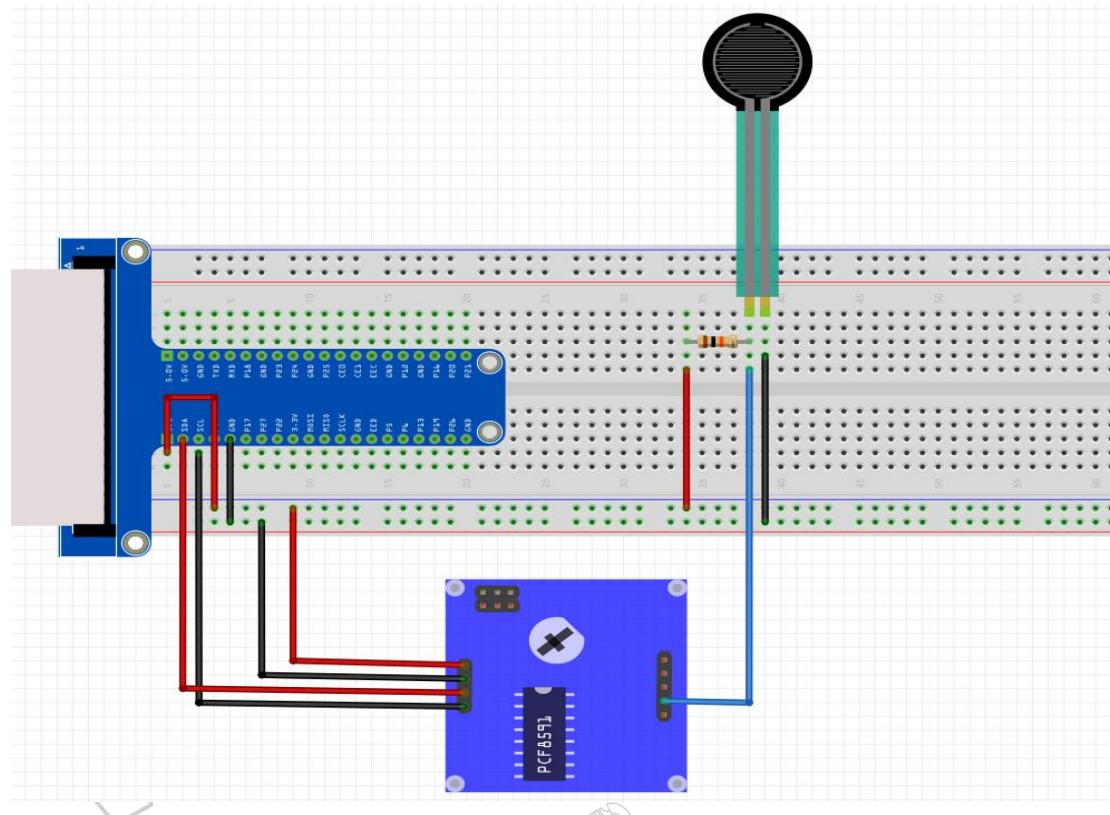
1 x Thermistor

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

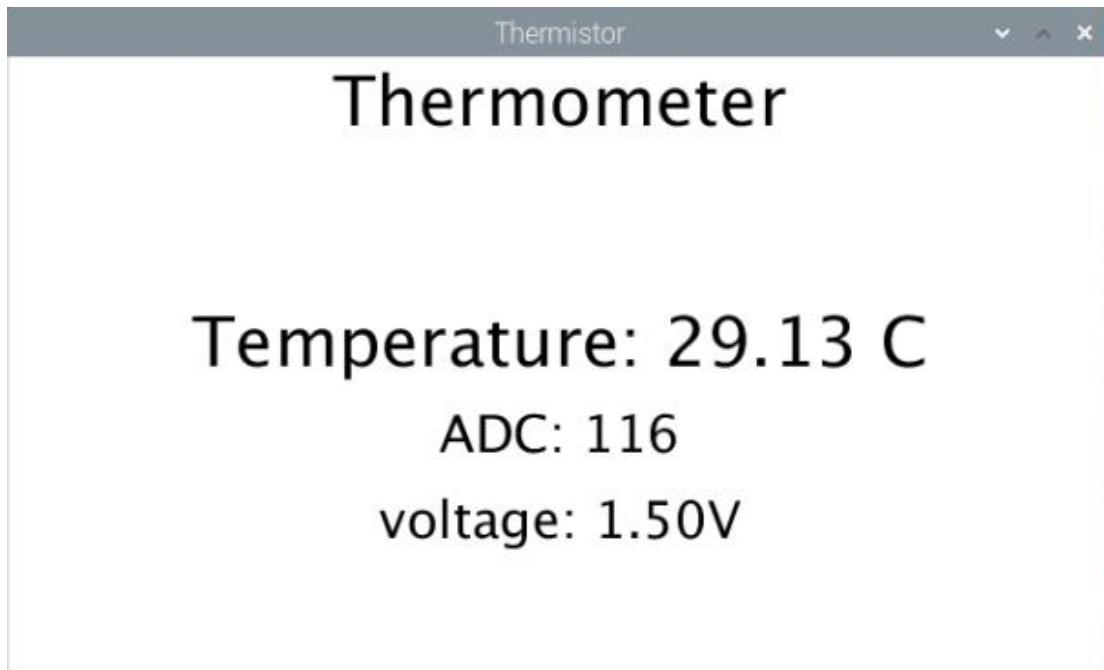


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 10.Thermistor / Thermistor.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nach Ausführung des Programms zeigt das Anzeigefenster die aktuelle Temperatur, den ADC-Wert und den Spannungswert an.



Das Folgende ist der Code:

```
import processing.io.*;
PCF8591 pcf = new PCF8591(0x48);
void setup() {
    size(640, 360);
}
void draw() {
    int adc = pcf.analogRead(0);
    float volt = adc*3.3/255.0;
    float tempK,tempC,Rt;
    Rt = 10*volt / (3.3-volt);
    tempK = 1/(1/(273.15+25) + log(Rt/10)/3950);
    tempC = tempK - 273.15;

    background(255);
    titleAndSiteInfo();

    fill(0);
    textAlign(CENTER);
    textSize(30);
    text("ADC: "+nf(adc, 0, 0), width / 2, height/2+50);
    textSize(30);
```

```
text("voltage: "+nf(volt, 0, 2)+"V", width / 2, height/2+100);
  textSize(40);
  text("Temperature: "+nf(tempC, 0, 2)+" C", width / 2, height/2);
}
void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);
  textSize(40);
  text("Thermometer", width / 2, 40);
  textSize(16);

}
```

Code Interpretation

```
int adc = pcf.analogRead(0);
float volt = adc*3.3/255.0;
float tempK,tempC,Rt;
Rt = 10*volt / (3.3-volt);
tempK = 1/(1/(273.15+25) + log(Rt/10)/3950);
tempC = tempK - 273.15;
```

Lesen Sie in diesem Projektcode zuerst ADC und berechnen Sie dann die aktuelle Temperatur gemäß dem zuvor erwähnten Ohmschen Gesetz und der Temperaturformel. Zeigen Sie sie schließlich im Anzeigefenster an.

Lektion 11 74HC595 & LED

Überblick

In dieser Lektion erfahren Sie, wie Sie 74HC595 verwenden und wie Sie damit LEDs steuern.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

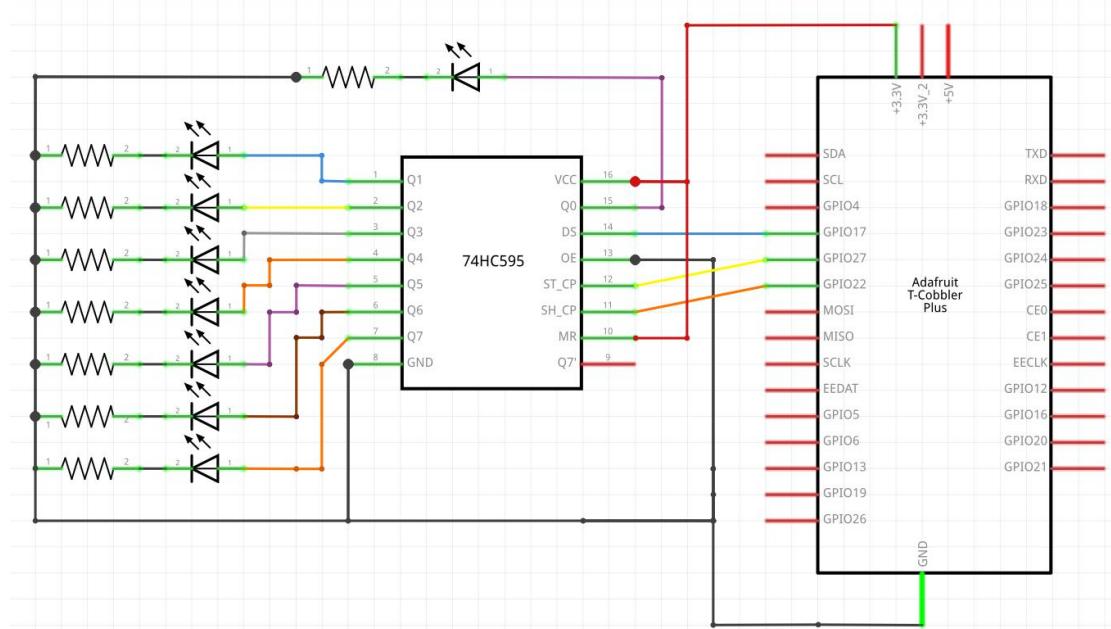
8 x 220 Ohm Widerstand

8 x LED

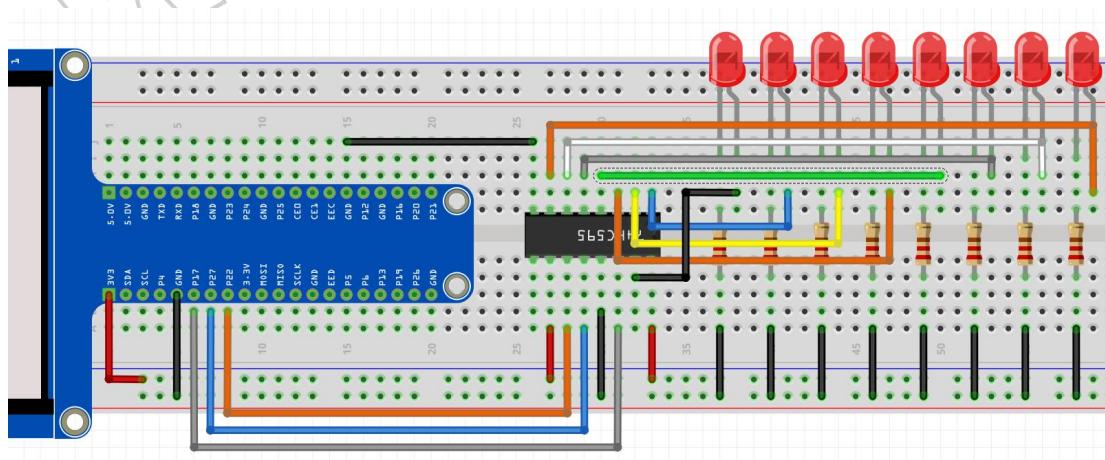
1 x 74HC595 Chip

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagram

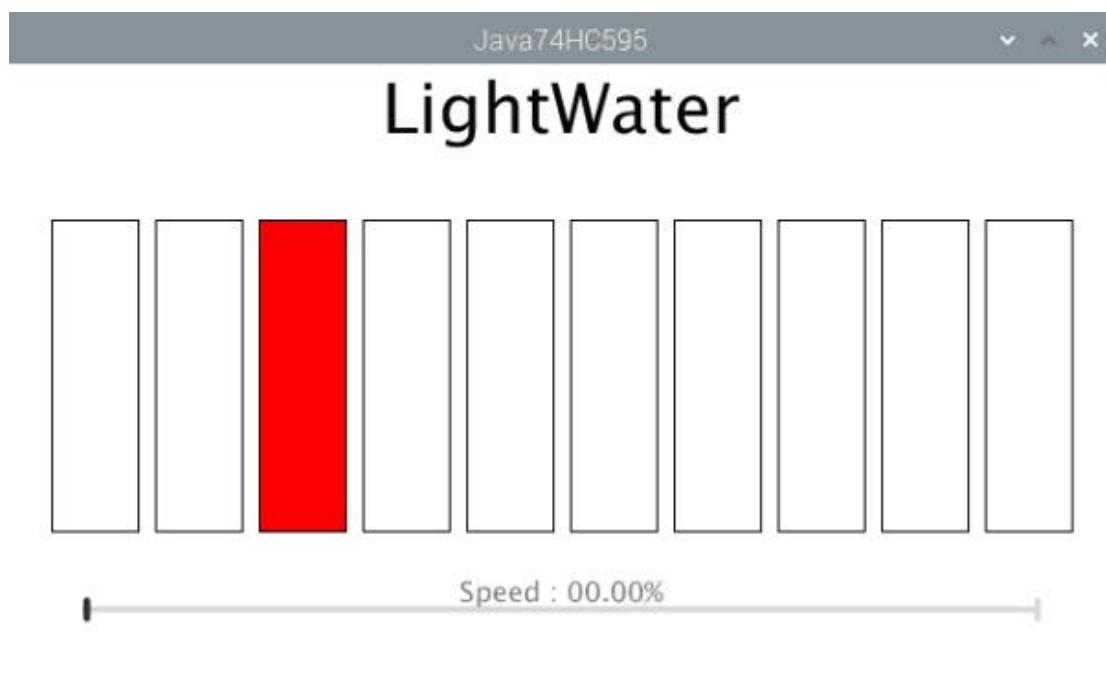


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal `processing` / `Java` / `12.Java74HC595LED` / `Java74HC595` / `Java74HC595.pde` ein, um den Code zu öffnen;
 2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "**RUN**", um den Code auszuführen;

Nach dem Ausführen des Programms zeigt das Anzeigefenster ein virtuelles LEDBar Graph an, das mit der gleichen Geschwindigkeit und Weise wie das LEDBar Graph aufleuchtet. Ziehen Sie den Fortschrittsbalken, um die Durchflussrate von leichtem Wasser anzupassen. Wie im Folgenden gezeigt:



Das Folgende ist der Code:

```
import processing.io.*;  
  
int dataPin = 17;  
int latchPin = 27;  
int clockPin = 22;  
final int borderSize = 45;  
ProgressBar mBar;  
IC74HC595 ic;  
boolean mMouse = false;  
int leds = 0x01;  
int lastMoveTime = 0;  
void setup() {  
    size(640, 360);  
    mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);  
    mBar.setTitle("Speed");  
    ic = new IC74HC595(dataPin, latchPin, clockPin);  
}
```

```
void draw() {
    background(255);
    titleAndSiteInfo();
    strokeWeight(4);
    mBar.create();

    if (millis() - lastMoveTime > 50/(0.05+mBar.progress)) {
        lastMoveTime = millis();
        leds<<=1;
        if (leds == 0x100)
            leds = 0x01;
    }
    ic.write(ic.LSBFIRST, leds);

    stroke(0);
    strokeWeight(1);
    for (int i=0; i<10; i++) {
        if (leds == (1<<i)) {
            fill(255, 0, 0);
        } else {
            fill(255, 255, 255);
        }
        rect(25+60*i, 90, 50, 180);
    }
}

void mousePressed() {
    if ( (mouseY< mBar.y+5) && (mouseY>mBar.y-5) ) {
        mMouse = true;
    }
}
void mouseReleased() {
    mMouse = false;
}
void mouseDragged() {
    int a = constrain(mouseX, borderSize, width - borderSize);
    float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
    if (mMouse) {
        mBar.setProgress(t);
    }
}
void titleAndSiteInfo() {
```

```
fill(0);
  textAlign(CENTER);
  textSize(40);
  text("LightWater", width / 2, 40);
  textSize(16);
}
```

Code Interpretation

```
int dataPin = 17;
int latchPin = 27;
int clockPin = 22;
final int borderSize = 45;
ProgressBar mBar;
IC74HC595 ic;
boolean mMouse = false;
int leds = 0x01;
int lastMoveTime = 0;
```

Definieren Sie zuerst den an 74HC595 angeschlossenen GPIO Pin, das ProgressBar-Klassenobjekt, IC74HC595 Klassenobjekt und einige Variablen.

```
mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);
mBar.setTitle("Speed");
ic = new IC74HC595(dataPin, latchPin, clockPin);
```

Instanziieren Sie in der Funktion setup () das ProgressBar Klassenobjekt und das IC74HC595-Klassenobjekt.

```
background(255);
titleAndSiteInfo();
strokeWeight(4);
mBar.create();
```

Stellen Sie in der Funktion draw () den Hintergrund, den Text und andere Informationen ein und zeichnen Sie den Fortschrittsbalken.

```
if (millis() - lastMoveTime > 50/(0.05+mBar.progress)) {
  lastMoveTime = millis();
  leds<<=1;
  if (leds == 0x100)
    leds = 0x01;
}
```

```
ic.write(ic.LSBFIRST, leds);
```

Dann entsprechend der Geschwindigkeit des Atemlichts, berechnen Sie die Daten "LEDs" für 74HC595, und machen Sie es auf 74HC595 geschrieben, dann wird LEDBar Graph eingeschaltet.

```
stroke(0);
strokeWeight(1);
for (int i=0; i<10; i++) {
    if (leds == (1<<i)) {
        fill(255, 0, 0);
    } else {
        fill(255, 255, 255);
    }
    rect(25+60*i, 90, 50, 180);
}
```

Entsprechend den Variable LED, zeichnen Sie das virtuelle LEDBar Diagramm im Anzeigefenster.

Über die Klasse IC74HC595:

```
class IC74HC595
```

Dies ist eine benutzerdefinierte Klasse, die wird verwendet, um die integrierte Schaltung 74HC595 zu betreiben.

```
public IC74HC595(int dPin, int lPin, int cPin)
```

Konstruktion Funktion, der Parameter für den an 74HC595 angeschlossenen GPIO Pin.

```
public void write(int order,int value)
```

Wird zum Schreiben von Daten auf 74HC595 verwendet, und der 74HC595-Ausgangsport gibt diese Daten sofort aus.

Lektion 12 74HC595 & Sieben-Segment-Anzeige

Überblick

In dieser Lektion lernen wir eine neue Komponente, die Ein Bit Sieben Segment Anzeige (SSD).

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

1 x 220 Ohm Widerstand

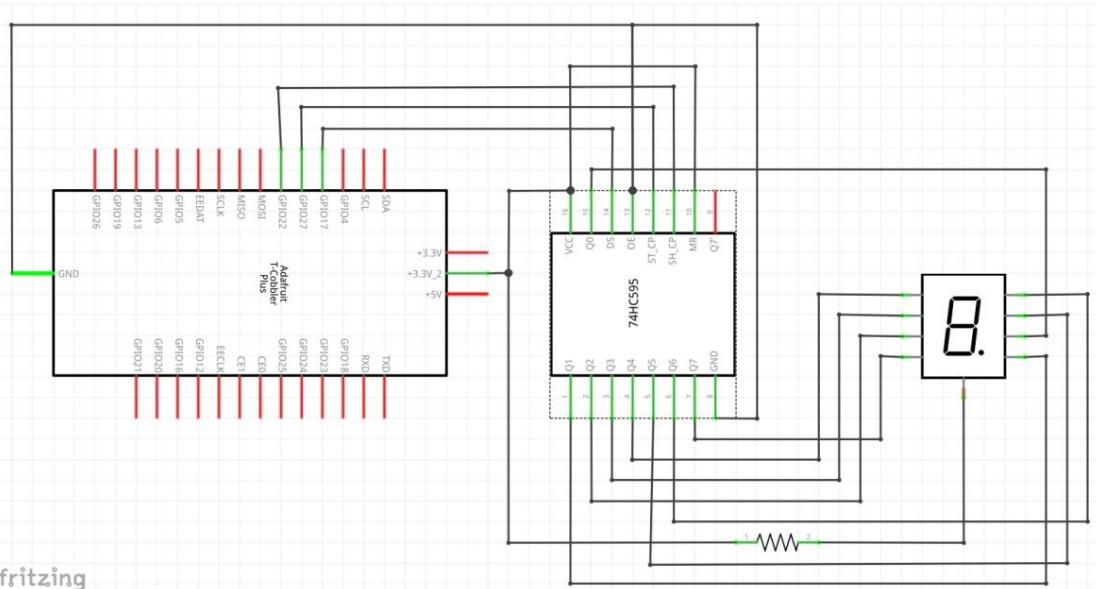
1 x Ein Bit Sieben-Segment-Anzeige

1 x 74HC595 Chip

Einige Jumper Drähte



Schaltplan



Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 12.Segment1 / Segment1 / Segment1.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, sowohl Anzeigefenster als auch SSD in der Schaltung zeige die gleiche Nummer. Und sie haben die gleiche Selbstbeschleunigungsrate, um die Nummer "0-9" ständig anzuzeigen. Durch Ziehen des Fortschrittsbalkens kann die Selbstbeschleunigungsrate angepasst werden.



```
import processing.io.*;
```

```
int dataPin = 17;  
int latchPin = 27;  
int clockPin = 22;  
final int borderSize = 45;
```

```
ProgressBar mBar;
IC74HC595 ic;
boolean mMouse = false;
int index = 0;

int lastMoveTime = 0;
final int[] numCode = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};

PFont mFont;

void setup() {
    size(640, 360);
    mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);
    mBar.setTitle("Speed");
    ic = new IC74HC595(dataPin, latchPin, clockPin);
    mFont = loadFont("DigifaceWide-100.vlw");
}

void draw() {
    background(255);
    titleAndSiteInfo();
    strokeWeight(4);
    mBar.create();
    if (millis() - lastMoveTime > 50/(0.05*mBar.progress)) {
        lastMoveTime = millis();
        index++;
        if (index > 9) {
            index = 0;
        }
    }
    ic.write(ic.MSBFIRST, numCode[index]);
    showNum(index);
}

void showNum(int num) {
    fill(0);
    textSize(100);
    textFont(mFont);
    textAlign(CENTER, CENTER);
    text(num, width/2, height/2);
}

void mousePressed() {
    if ( (mouseY< mBar.y+5) && (mouseY>mBar.y-5) ) {
        mMouse = true;
    }
}
```

```

    }
    void mouseReleased() {
        mMouse = false;
    }

    void mouseDragged() {

        int a = constrain(mouseX, borderSize, width - borderSize);
        float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
        if (mMouse) {
            mBar.setProgress(t);
        }
    }
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);
    textSize(16);
    text("Seven-segment Display", width / 2, 40);
    textSize(40);
}
}

```

Code Interpretation

```
final int[] numCode = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};
```

Der Projektcode ähnelt dem letzten Kapitel. Der Unterschied besteht darin, dass in diesem Projekt die von 74HC595 ausgegebenen Daten die festen Codierungsinformationen der SSD sind. Zunächst wird das Zeichen "0-9" als Code der SSD mit gemeinsamer Anode definiert.

```

if (millis() - lastMoveTime > 50/(0.05+mBar.progress)) {
    lastMoveTime = millis();
    index++;
    if (index > 9) {
        index = 0;
    }
}
ic.write(ic.MSBFIRST, numCode[index]);
showNum(index);

```

In der Funktion draw () werden die Daten mit einer bestimmten Geschwindigkeit ausgegeben. Gleichzeitig gibt das Anzeigefenster das gleiche Zeichen aus.

```
PFont mFont;  
  
mFont = loadFont("DigifaceWide-100.vlw");
```

Durch Erstellen der Schriftart "mFont" können wir die Schriftart der Zeichen im Anzeigefenster ändern. Die Schriftart ".vlw" wird durch Klicken auf "Schriftart erstellen" in der Menüleiste erstellt, die im Datenordner des aktuellen "Segment1" gespeichert wird.

```
textFont(createFont("", 100));
```

Durch Erstellen einer leeren Schriftart können Sie die Schriftart auf die Standardschriftart zurücksetzen.

Weitere Informationen zu loadFont () finden Sie auf der offiziellen Website:

https://processing.org/reference/loadFont_.html

Lektion 13 74HC595 & 4-Bits

Sieben-Segment-Anzeige

Überblick

In dieser Lektion lernen wir, wie man eine 4-Bits Sieben-Segment-Anzeige verwendet.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

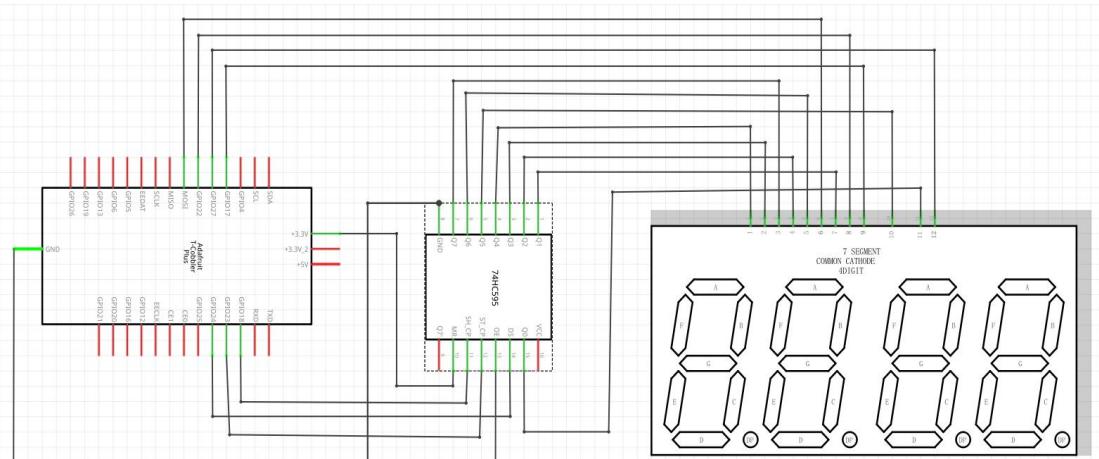
1 x Steckbrett

1x 4-Bits Sieben-Segment-Anzeige

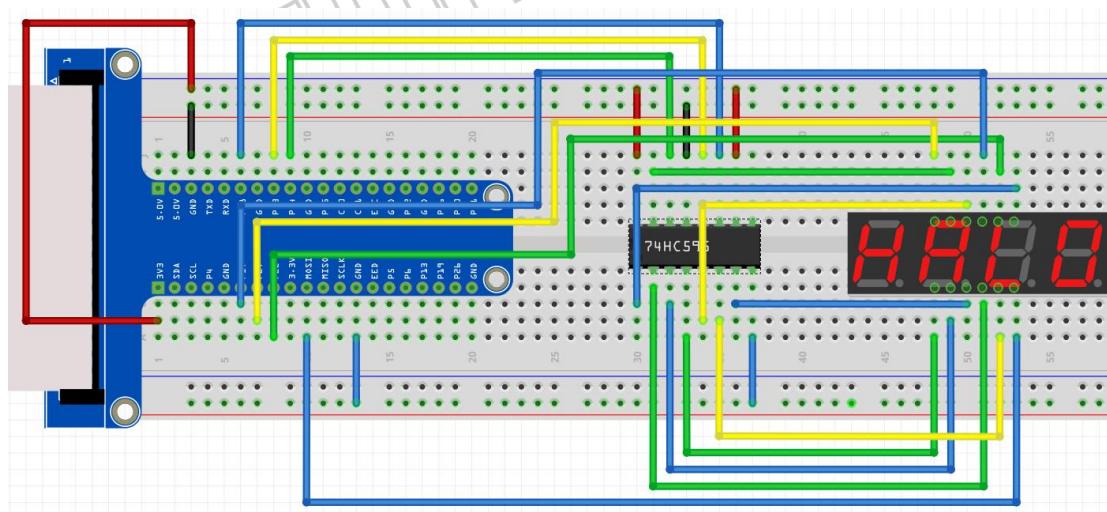
1 x 74HC595

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagram



Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal **processing code / Java / 13.Segment4 / Segment4 / Segment4.pde** ein, um den Code zu öffnen;
 2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "**RUN**", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, zeigen Anzeigefenster und FDSSD in der Schaltung dieselben Zahlen an und sie haben dieselbe Add-Beschleunigungsrate. Sie

zeigen ständig kreisförmig die Nummer "0-9999" an. Durch Ziehen des Fortschrittsbalkens kann die Rate geändert werden.



Das Folgende ist der Code:

```
import processing.io.*;  
  
int dataPin = 24;  
int latchPin = 23;  
int clockPin = 18;  
int[] digitPin = {17, 27, 22, 10};  
final int borderSize = 45;  
ProgressBar mBar;  
IC74HC595 ic;  
boolean mMouse = false;  
int index = 0;  
int lastMoveTime = 0;  
final int[] numCode = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};  
PFont mFont;  
  
void setup() {  
    size(640, 360);  
    for (int i = 0; i < 4; i++) {
```

```
    GPIO.pinMode(digitPin[i], GPIO.OUTPUT);
}
mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);

mBar.setTitle("Speed");
ic = new IC74HC595(dataPin, latchPin, clockPin);
mFont = loadFont("DigifaceWide-100.vlw");
thread("displaySSD");
}

void draw() {
background(255);
titleAndSiteInfo();
strokeWeight(4);
mBar.create();
if (millis() - lastMoveTime > 50/(0.05+mBar.progress)) {
    lastMoveTime = millis();
    index++;
    if (index > 9999) {
        index = 0;
    }
}
showNum(index);
}
void showNum(int num) {
fill(0);
textSize(100);
textFont(mFont);
textAlign(CENTER, CENTER);
text(nf(num,4,0), width/2, height/2);
}

void displaySSD() {
while (true) {
    display(index);
}
}

void selectDigit(int digit) {
    GPIO.digitalWrite(digitPin[0], ((digit&0x08) != 0x08) ? GPIO.LOW : GPIO.HIGH);
    GPIO.digitalWrite(digitPin[1], ((digit&0x04) != 0x04) ? GPIO.LOW : GPIO.HIGH);
    GPIO.digitalWrite(digitPin[2], ((digit&0x02) != 0x02) ? GPIO.LOW :
```

```
GPIO.HIGH);
    GPIO.digitalWrite(digitPin[3], ((digit&0x01) != 0x01) ? GPIO.LOW : GPIO.HIGH);

}

void display(int dec) {
    selectDigit(0x00);
    ic.write(ic.MSBFIRST, numCode[dec%10]);
    selectDigit(0x01);
    delay(1);
    selectDigit(0x00);
    ic.write(ic.MSBFIRST, numCode[dec%100/10]);
    selectDigit(0x02);
    delay(1);
    selectDigit(0x00);
    ic.write(ic.MSBFIRST, numCode[dec%1000/100]);
    selectDigit(0x04);
    delay(1);
    selectDigit(0x00);
    ic.write(ic.MSBFIRST, numCode[dec%10000/1000]);
    selectDigit(0x08);
    delay(1);
}
void mousePressed() {
    if ( (mouseY < mBar.y+5) && (mouseY > mBar.y-5) ) {
        mMouse = true;
    }
}
void mouseReleased() {
    mMouse = false;
}
void mouseDragged() {
    int a = constrain(mouseX, borderSize, width - borderSize);
    float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
    if (mMouse) {
        mBar.setProgress(t);
    }
}
void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);
    textSize(40);
    font = createFont("", 100);
}
```

```
    text("4-Digit 7-Segment Display", width / 2, 40);
    textSize(16);
}
```

Code Interpretation

```
int[] digitPin = {17, 27, 22, 10};
```

Der Projektcode ähnelt dem letzten Kapitel. Der Unterschied besteht darin, dass das Projekt vier Digitalanzeigen steuern muss. Die vier Koplanare der vierstelligen Anzeigen werden von vier GPIOs über vier Transistoren gesteuert. Zunächst werden vier GPIOs definiert.

```
thread("displaySSD");
void displaySSD() {
    while (true) {
        display(index);
    }
}
```

In einem separaten Thread, lassen Sie die FDSSD die Zahlen im Scanmodus anzeigen. Mit der Unterfunktionsanzeige () macht FDSSD zeigt eine vierstellige Nummer an.

Lektion 14 74HC595 & LED Matrix

Überblick

In dieser Lektion lernen Sie, wie Sie den 74HC595 weiterhin zur Steuerung weiterer LEDs verwenden, das heißt LED Matrix. Wir verwenden zwei 74HC595 zur Steuerung einer monochromen LEDMatrix (8*8), um einige Grafiken und Zeichen anzuzeigen.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-type Erweiterungskarte

1 x Steckbrett

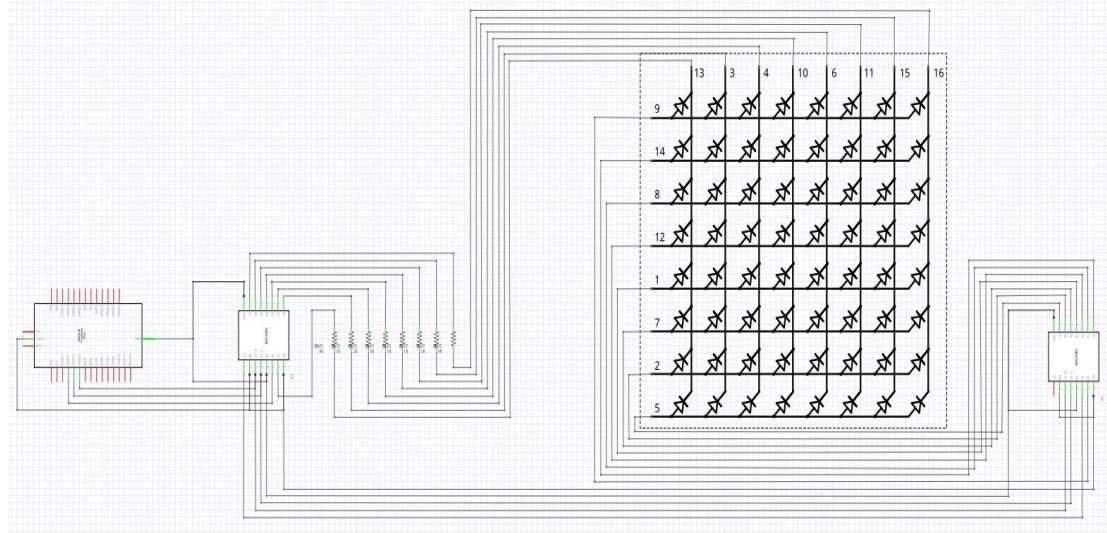
2 x 74HC595

8 x 220Ω Widerstand

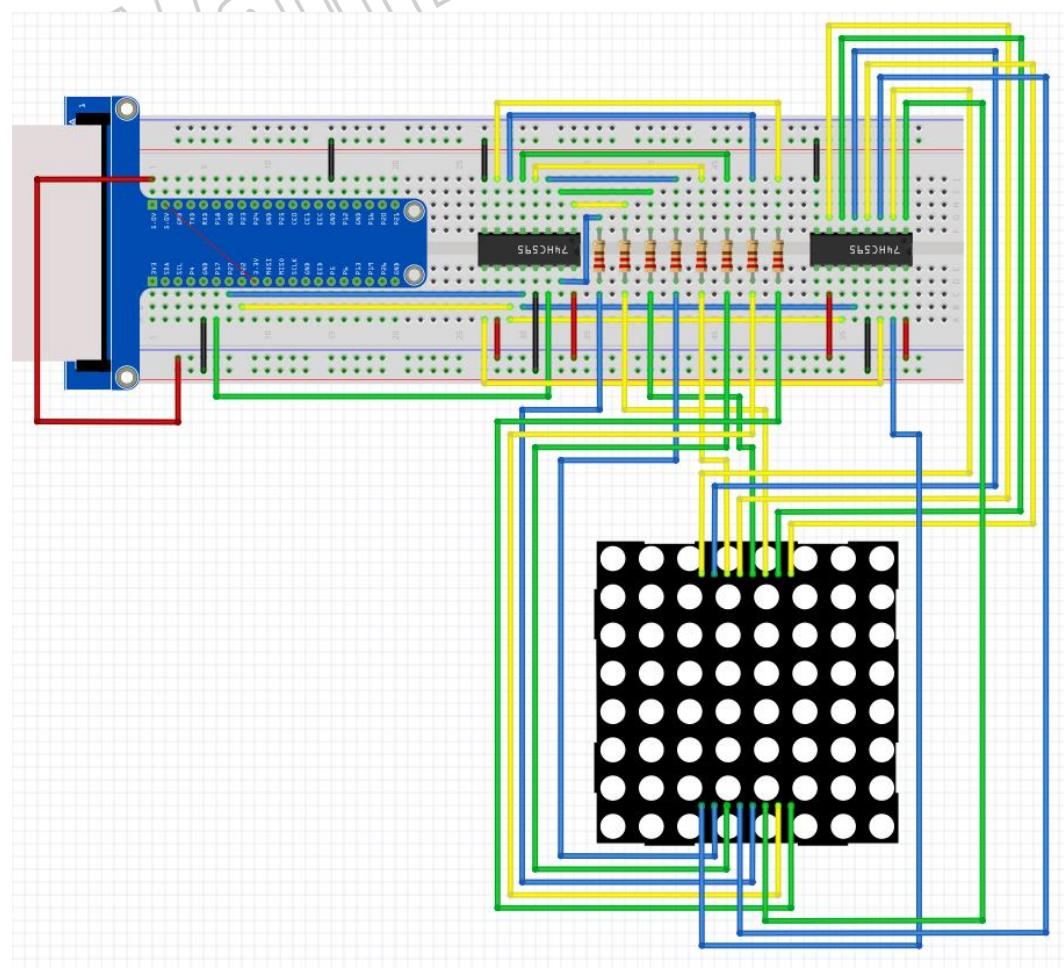
1 x LED Matrix (8*8)

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

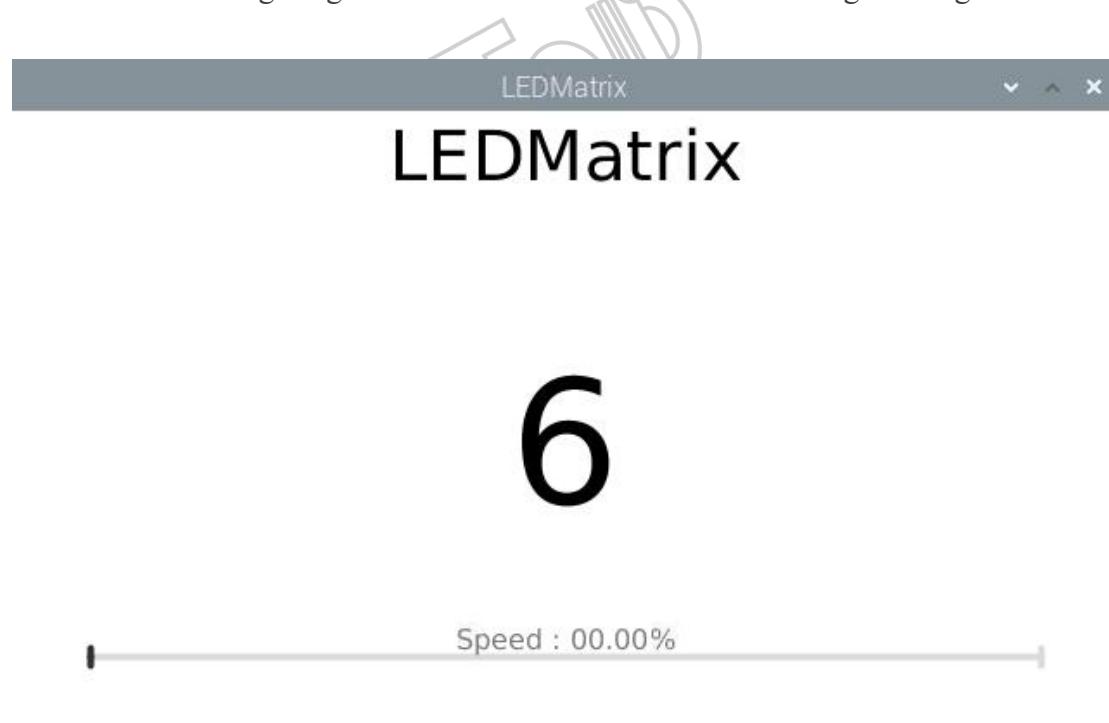


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 14.LEDMatrix / LEDMatrix / LEDMatrix.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, beginnt die LED-Matrix mit der Anzeige von Grafiken und Zeichen, und das Anzeigefenster ändert sich mit dem Status der von der LED-Matrix angezeigten Grafiken und Zeichen. Wie nachfolgend dargestellt:



Folgendes ist Programmcode:

```
import processing.io.*;  
  
int dataPin = 17;      //connect to the 74HC595  
int latchPin = 27;  
int clockPin = 22;  
final int borderSize = 45;    //border size  
ProgressBar mBar;        //ProgressBar Object  
IC74HC595 ic;          //IC74HC595 Object  
boolean mMouse = false;   //determined whether a mouse click the ProgressBar
```

```
int index = 0;          // index of number

//encoding for smile face
final int[] pic = {0x1c, 0x22, 0x51, 0x45, 0x45, 0x51, 0x22, 0x1c};

//encoding for character 0-9 of ledmatrix
final int[] numCode={

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "
    0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, // "0"
    0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, // "1"
    0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, // "2"
    0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, // "3"
    0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, // "4"
    0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, // "5"
    0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, // "6"
    0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, // "7"
    0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, // "8"
    0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, // "9"
    0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, // "A"
    0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, // "B"
    0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, // "C"
    0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, // "D"
    0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, // "E"
    0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00, // "F"
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "

};

myThread t = new myThread();      //create a new thread for ledmatrix
void setup() {
    size(640, 360);
    mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);
    mBar.setTitle("Speed");    //set the ProgressBar's title
    ic = new IC74HC595(dataPin, latchPin, clockPin);
    t.start();    //thread start
}

void draw() {
    background(255);
    titleAndSiteInfo(); //title and site information
    strokeWeight(4);   //border weight
    mBar.create();     //create the ProgressBar
    displayNum(hex(index, 1)); //show the number in dispaly window
}

class myThread extends Thread {
    public void run() {
```

```
while (true) {  
  
    showMatrix();      //show smile picture  
    showNum();        //show the character "0-F"  
}  
}  
}  
}  
void showMatrix() {  
    for (int j=0; j<100; j++) {      //picture show time  
        int x=0x80;  
        for (int i=0; i<8; i++) {      //display a frame picture  
            GPIO.digitalWrite(latchPin, GPIO.LOW);  
            ic.shiftOut(ic.MSBFIRST, pic[i]);  
            ic.shiftOut(ic.MSBFIRST, ~x);  
            GPIO.digitalWrite(latchPin, GPIO.HIGH);  
            x>>=1;  
        }  
    }  
}  
}  
void showNum() {  
    for (int j=0; j<numCode.length-8; j++) { //where to start showing  
        index = j/8;  
        for (int k =0; k<10*(1.2-mBar.progress); k++) {      //speed  
            int x=0x80;  
            for (int i=0; i<8; i++) {      //display a frame picture  
                GPIO.digitalWrite(latchPin, GPIO.LOW);  
                ic.shiftOut(ic.MSBFIRST, numCode[j+i]);  
                ic.shiftOut(ic.MSBFIRST, ~x);  
                GPIO.digitalWrite(latchPin, GPIO.HIGH);  
                x>>=1;  
            }  
        }  
    }  
}  
}  
void displayNum(String num) {  
    fill(0);  
    textSize(100);  
    textAlign(CENTER, CENTER);  
    text(num, width/2, height/2);  
}  
}  
void mousePressed() {  
    if ( (mouseY<mBar.y+5) && (mouseY>mBar.y-5) ) {  
        mMouse = true;      //the mouse click the progressBar
```

```

        }

    void mouseReleased() {
        mMouse = false;
    }

    void mouseDragged() {
        int a = constrain(mouseX, borderSize, width - borderSize);
        float t = map(a, borderSize, width - borderSize, 0.0, 1.0);
        if (mMouse) {
            mBar.setProgress(t);
        }
    }

    void titleAndSiteInfo() {
        fill(0);
        textAlign(CENTER);      //set the text centered
        textSize(40);          //set text size
        text("LEDMatrix", width / 2, 40);    //title
        textSize(16);
    }
}

```

Code Interpretation

```

final int[] pic = {0x1c, 0x22, 0x51, 0x45, 0x45, 0x51, 0x22, 0x1c};
//encoding for character 0-9 of ledmatrix
final int[] numCode={

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "
    0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, // "0"
    0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, // "1"
    0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, // "2"
    0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, // "3"
    0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, // "4"
    0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, // "5"
    0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, // "6"
    0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, // "7"
    0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, // "8"
    0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, // "9"
    0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, // "A"
    0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, // "B"
    0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, // "C"
    0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, // "D"
    0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, // "E"
    0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00, // "F"
}

```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " "
};


```

Definieren Sie im Code zunächst die Daten des lächelnden Gesichts und der Zeichen "0-F".

```
myThread t = new myThread();
.....
class myThread extends Thread {
    public void run() {
        while (true) {
            showMatrix();      //show smile picture
            showNum();        //show the character "0-F"
        }
    }
}


```

Dann erstelle einen neuen **Thread t**. Der LEDMatrix Scan Anzeige Code wird in **run()** dieses Threads ausgeführt.

```
void setup() {
    size(640, 360);
    mBar = new ProgressBar(borderSize, height-borderSize, width-borderSize*2);
    mBar.setTitle("Speed");    //set the ProgressBar's title
    ic = new IC74HC595(dataPin, latchPin, clockPin);
    t.start();    //thread start
}


```

Die Funktion **setup()** definiert die Größe des Anzeigefensters, der **ProgressBar** Klassenobjekte und des **IC75HC595** Klassenobjekts und startet den **Thread t**.

```
void draw() {
    background(255);
    titleAndSiteInfo(); //title and site information
    strokeWeight(4);   //border weight
    mBar.create();     //create the ProgressBar
    displayNum(hex(index, 1)); //show the number in dispaly window
}


```

Zeichnen Sie in **draw()** die relevanten Informationen und die aktuell anzuzeigende Nummer.

```
void showMatrix() {
```

```
for (int j=0; j<100; j++) {      //picture show time
    int x=0x80;
    for (int i=0; i<8; i++) {      //display a frame picture

        GPIO.digitalWrite(latchPin, GPIO.LOW);
        ic.shiftOut(ic.MSBFIRST, pic[i]);
        ic.shiftOut(ic.MSBFIRST, ~x);
        GPIO.digitalWrite(latchPin, GPIO.HIGH);
        x>>=1;
    }
}

}
```

Die Unterfunktion `showMatrix()` bewirkt, dass die LED Matrix ein lächelndes Gesichtsmuster anzeigt und eine gewisse Zeit hält.

```
void showNum() {
    for (int j=0; j<numCode.length-8; j++) { //where to start showing
        index = j/8;
        for (int k =0; k<10*(1.2-mBar.progress); k++) {      //speed
            int x=0x80;
            for (int i=0; i<8; i++) {      //display a frame picture
                GPIO.digitalWrite(latchPin, GPIO.LOW);
                ic.shiftOut(ic.MSBFIRST, numCode[j+i]);
                ic.shiftOut(ic.MSBFIRST, ~x);
                GPIO.digitalWrite(latchPin, GPIO.HIGH);
                x>>=1;
            }
        }
    }
}
```

Mit der Unterfunktion `showNum()` wird die LEDMatrix Bildlaufanzeige mit dem Zeichen "0-F" angezeigt, wobei die Variable `k` zum Einstellen der Bildlaufgeschwindigkeit verwendet wird.

Lektion 15 I2C-LCD1602

Überblick

In dieser Lektion lernen Sie den LCD1602 Anzeige. Die aktuelle Uhrzeit und das aktuelle Datum werden auf dem LCD1602 und im Anzeigefenster angezeigt.

Erforderliche Teile

1 x Raspberry Pi

1 x GPIO T-type Erweiterungskarte

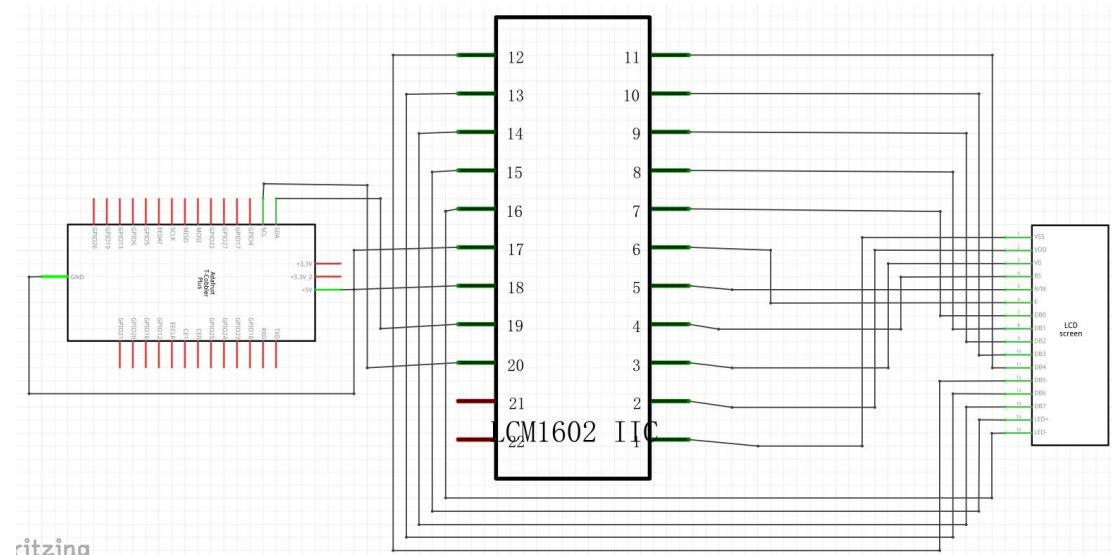
1 x Steckbrett

1 x LCD1602

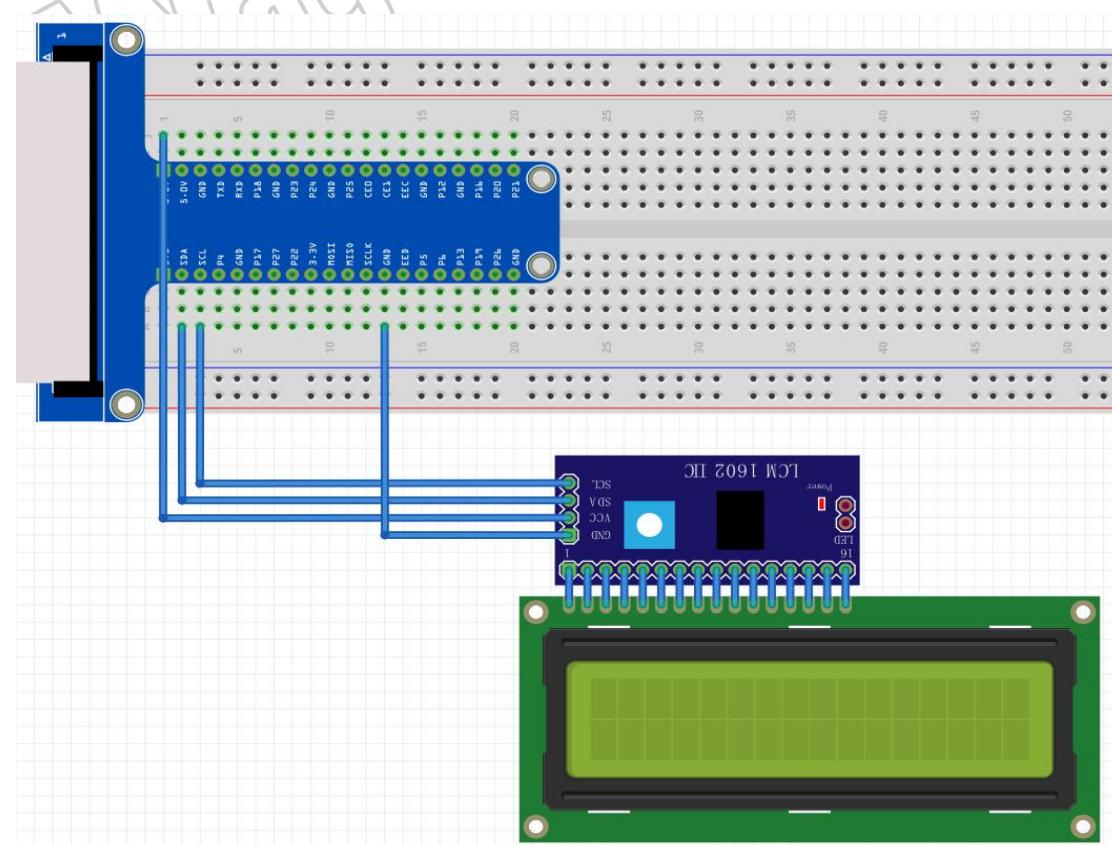
1 x PCF8574

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

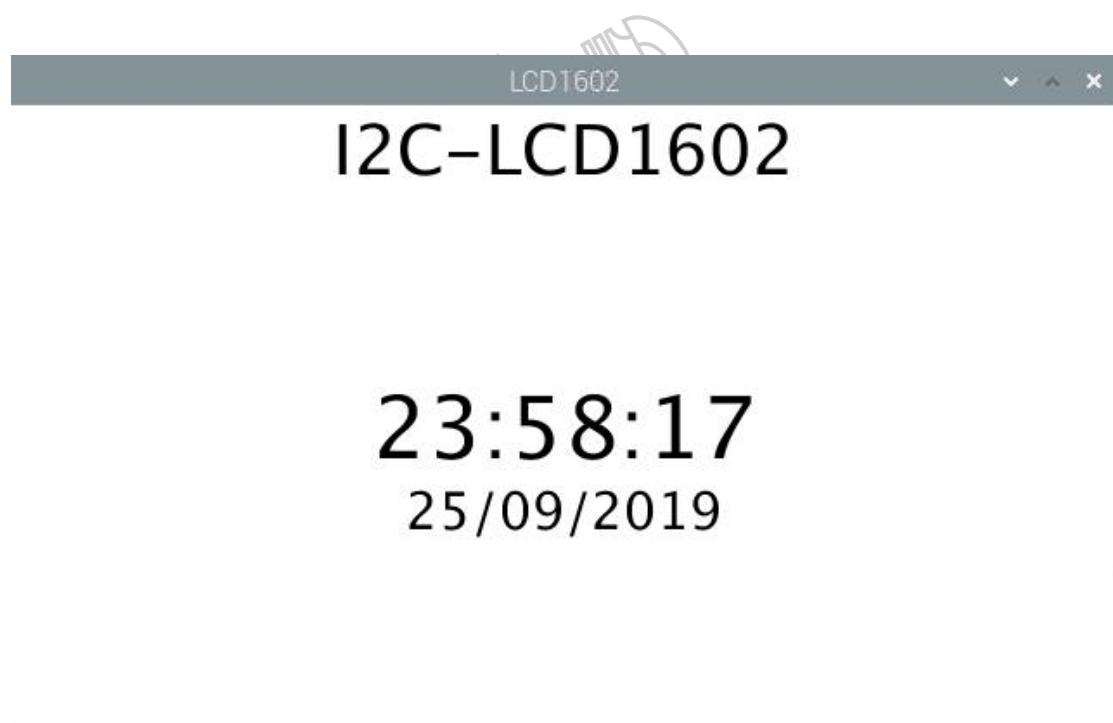


Code

First observe the results of the code, and then analyze the code.

1. Geben Sie in das Terminal [processing code / Java / 15.LCD1602 / LCD1602 / LCD1602.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "[RUN](#)", um den Code auszuführen;

Nach dem Ausführen des Programms zeigt das LCD1602 Uhrzeit und Datum an, und das Anzeigefenster zeigt auch den Inhalt des LCD1602 an. Wie nachfolgend dargestellt:



Das Folgende ist der Programmcode:

```
import processing.io.*;  
//Create a object of class PCF8574  
PCF8574 pcf = new PCF8574(0x27);  
LCD_LCD1602 lcd; //Create a lcd object  
String time = "";  
String date = "";  
void setup() {  
    size(640, 360);  
    lcd = new LCD_LCD1602(pcf);  
    frameRate(2); //set display window frame rate for 2 HZ
```

```
}

void draw() {
    background(255);
    titleAndSiteInfo();
    //get current time
    time = nf(hour(), 2, 0) + ":" + nf(minute(), 2, 0) + ":" + nf(second(), 2, 0);
    //get current date
    date = nf(day(), 2, 0) + "/" + nf(month(), 2, 0) + "/" + nf(year(), 2, 0);
    lcd.position(4, 0); //show time on the lcd display
    lcd.puts(time);
    lcd.position(3, 1); //show date on the lcd display
    lcd.puts(date);
    showTime(time, date); //show time/date on the display window
}

void showTime(String time, String date) {
    fill(0);
    textAlign(CENTER, CENTER);
    textSize(50);
    text(time, width/2, height/2);
    textSize(30);
    text(date, width/2, height/2+50);
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER); //set the text centered
    textSize(40); //set text size
    text("I2C-LCD1602", width / 2, 40); //title
    textSize(16);
}
```

Code Interpretation

```
PCF8574 pcf = new PCF8574(0x27);
LCD_LCD1602 lcd; //Create a lcd object
String time = "";
String date = "";
void setup() {
    size(640, 360);
    lcd = new LCD_LCD1602(pcf);
    frameRate(2); //set display window frame rate for 2 HZ
}
```

Erstellen Sie zuerst ein PCF8574 Klassenobjekt "pcf" und nehmen Sie "pcf" als Parameter, um ein LCD1602 Klassenobjekt zu erstellen. Definieren Sie dann die variable zum Speichern von Datum und Uhrzeit. Das Anzeigefenster muss nicht häufig aktualisiert werden. Daher kann die Bildrate auf 2 Hz eingestellt werden.

```
void draw() {  
    background(255);  
    titleAndSiteInfo();  
    //get current time  
    time = nf(hour(), 2, 0) + ":" + nf(minute(), 2, 0) + ":" + nf(second(), 2, 0);  
    //get current date  
    date = nf(day(), 2, 0) + "/" + nf(month(), 2, 0) + "/" + nf(year(), 2, 0);  
    lcd.position(4, 0); //show time on the lcd display  
    lcd.puts(time);  
    lcd.position(3, 1); //show date on the led display  
    lcd.puts(date);  
    showTime(time, date); //show time/date on the display window  
}
```

Rufen Sie in der Funktion draw () die aktuelle Uhrzeit und das aktuelle Datum ab und zeigen Sie sie auf dem LCD1602 und im Anzeigefenster an.

Über die Klasse PCF8574:

Dies ist eine benutzerdefinierte Klasse, die zur Steuerung der integrierten Schaltung PCF8574 verwendet wird.

public PCF8574(int addr)

Konstruktor Funktion, die zum Erstellen eines PCF8574 Klasse Objekts verwendet wird, Parameter für die I2C-Geräteadresse von PCF8574.

public int digitalRead(int pin)

Dient zum Lesen des Werts (HIGH / LOW) eines der Ports.

public int readByte()

Dient zum Lesen der Werte aller Ports.

public void digitalWrite(int pin, int val)

Schreiben Sie Daten (HIGH / LOW) in einen Port.

public void writeByte(int data)

Schreiben Sie Daten in alle Ports.

Über die Klasse LCD_LCD:

Dies ist eine benutzerdefinierte Klasse, die derzeit nur zur Steuerung des an PCF8574 angeschlossenen I2C-LCD1602 verwendet wird.

Public LCD_LCD1602(PCF8574 ipcf)

Konstruktions Funktion, die zum Erstellen des LCD1602 Klassenobjekts verwendet wird, dem Parameter für das PCF8574 Klassenobjekt.

public void putChar(char data)

Schreiben Sie ein Zeichen auf den LCD-Bildschirm.

public void puts(String str)

Schreiben Sie eine Zeichenfolge auf den LCD-Bildschirm.

public void display(boolean state)

LCD ein- / ausschalten.

public void lcdCursor(boolean state)

Cursor ein- / ausschalten.

public void cursorBlink(boolean state)

Cursor Blink ein- / ausschalten.

public void position(int x, int y)

Stellen Sie die Position des Cursors auf.

public void home()

Stellen Sie den Cursor auf home auf.

public void lcdClear()

Löschen Sie den Bildschirm.

public void backLightON() & public void backLightOFF()

Schalten Sie die die Hintergrundbeleuchtung ein / aus.

public void scrollDisplayLeft() & public void scrollDisplayRight()

Verschieben Sie den Bildschirm einer Einheit nach links / rechts.

public void leftToRight() & public void rightToLeft()

Stellen Sie die Textrichtung als Formular von links nach rechts / von rechts nach links ein.

public void autoScroll() & public void noAutoScroll()

Automatischer Schaltbildschirm / Schalten Sie den automatischen Schaltbildschirm aus.

Lektion 16 Joystick

Überblick

In dieser Lektion lernen Sie den Umgang mit dem Drehpotentiometer. Auf dem Bildschirm werden die Koordinaten und die Position des Joysticks angezeigt.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

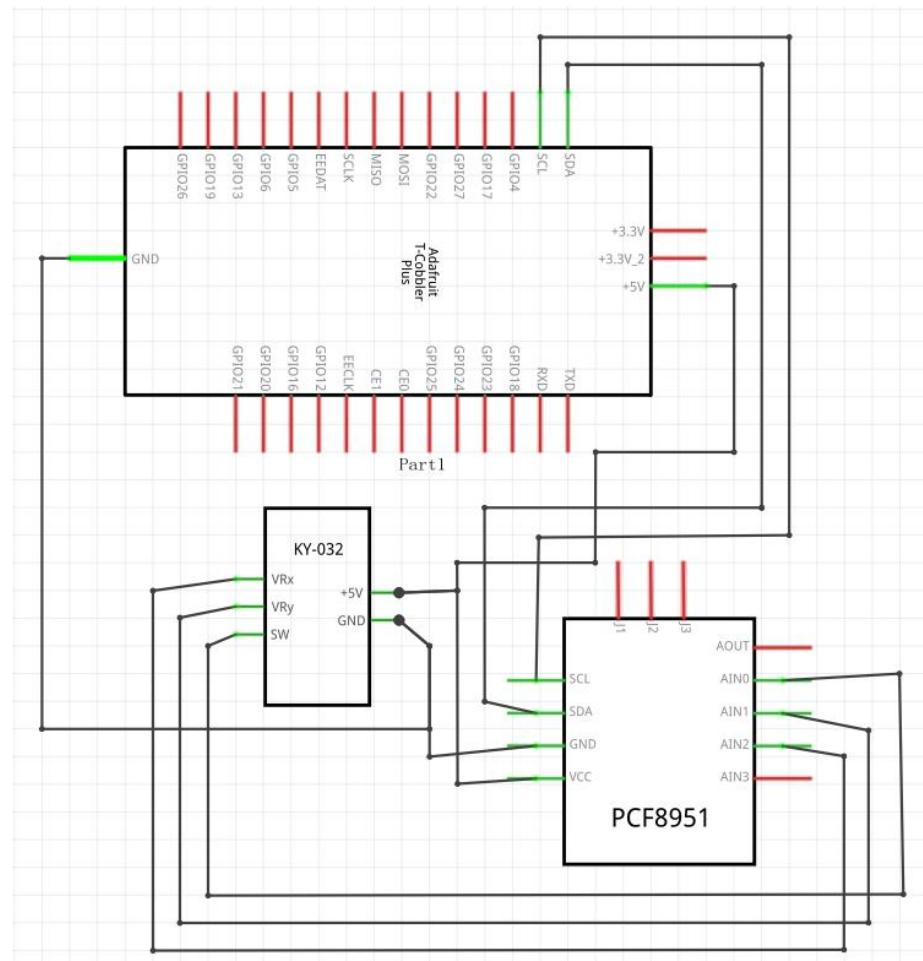
1 x Steckbrett

1 x PCF8591 Modul

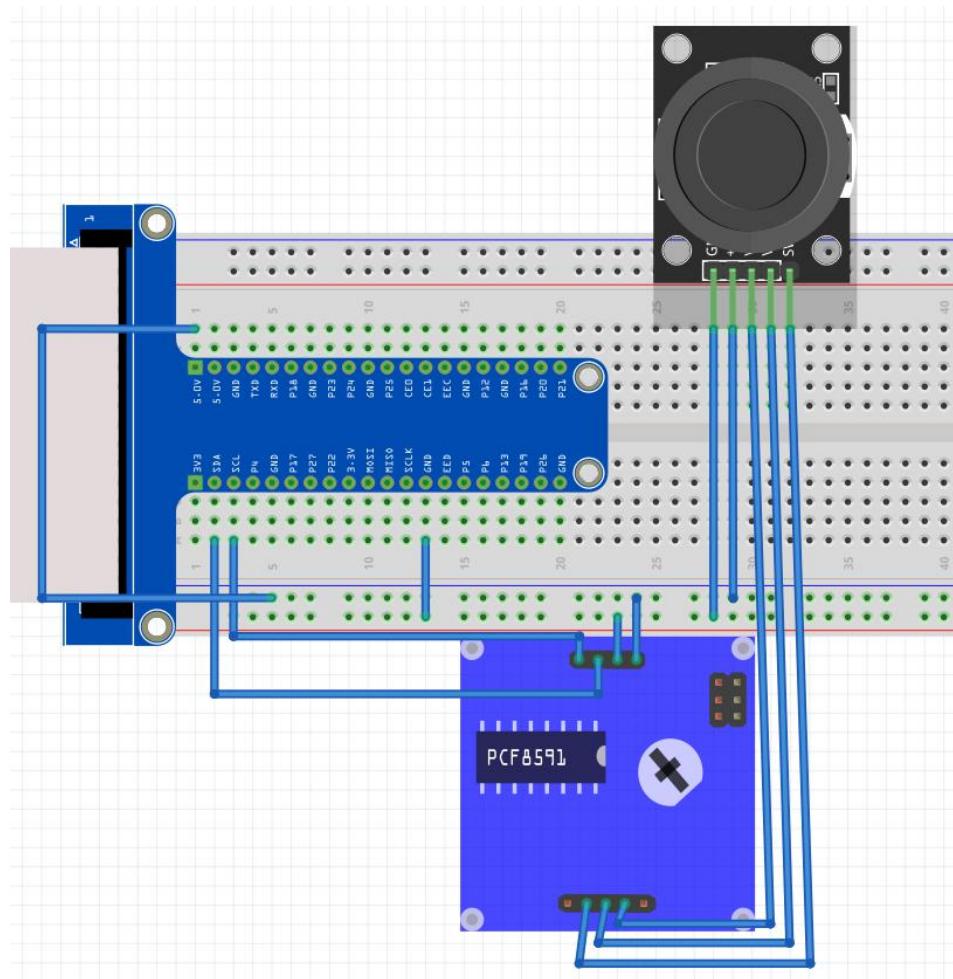
1 x Doppelachsen-Joystick

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

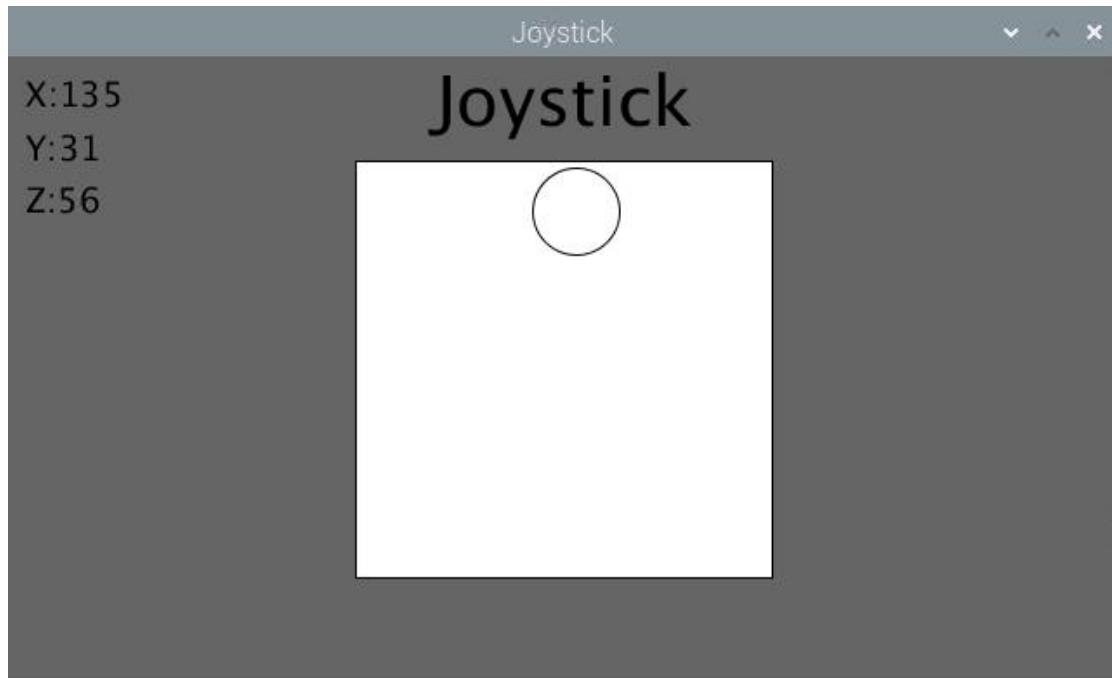


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 16.Joystick / Joystick.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, zeigt das Drehen des Joysticks, dass sich die Kreisposition der seriellen Schnittstelle mit dem Drehen des Joysticks ändert, und das Anzeigefenster zeigt die Koordinaten des Joysticks an. Der Lauffeffekt ist in der folgenden Abbildung dargestellt:



Das Folgende ist der Programmcode:

```
import processing.io.*;
//Create a object of class PCF8591
PCF8591 pcf = new PCF8591(0x48);
int cx, cy, cd, cr;      //define the center point,side length & half.

void setup() {
    size(640, 360);
    cx = width/2;      //center of the display window
    cy = height/2;     //
    cd = (int)(height/1.5);
    cr = cd /2;
}
void draw() {
    int x=0, y=0, z=0;
    x = pcf.analogRead(2); //read the ADC of joystick
    y = pcf.analogRead(1); //
    z = pcf.analogRead(0);
    background(102);
    titleAndSiteInfo();
    fill(0);
    textSize(20);
```

```

textAlign(LEFT, TOP);

text("X:" + x + "\nY:" + y + "\nZ:" + z, 10, 10);

fill(255);      //wall color
rect(cx-cr, cy-cr, cd, cd);
fill(constrain(z, 255, 0));    //joystick color
ellipse(map(x, 0, 255, cx-cr, cx+cr), map(y, 0, 255, cy-cr, cy+cr), 50, 50);
}

void titleAndSiteInfo() {
  fill(0);
  textAlign(CENTER);    //set the text centered
  textSize(40);         //set text size
  text("Joystick", width / 2, 40); //title
  textSize(16);
}

```

Code Interpretation

```

void draw() {
  int x=0, y=0, z=0;
  x = pcf.analogRead(2); //read the ADC of joystick
  y = pcf.analogRead(1); //
  z = pcf.analogRead(0);
  background(102);
  titleAndSiteInfo();
  fill(0);
  textSize(20);
  textAlign(LEFT, TOP);
  text("X:" + x + "\nY:" + y + "\nZ:" + z, 10, 10);

  fill(255);      //wall color
  rect(cx-cr, cy-cr, cd, cd);
  fill(constrain(z, 255, 0));    //joystick color
  ellipse(map(x, 0, 255, cx-cr, cx+cr), map(y, 0, 255, cy-cr, cy+cr), 50, 50);
}

```

In der Funktion draw () wird der ADC Wert des Joysticks mit drei Achsen gelesen. Der ADC Wert der X- und Y-Richtung wird auf die Position des Kreises abgebildet, und der ADC Wert der Z Achse wird auf die gefüllte Farbe des Kreises abgebildet.

Lektion 17 Relay & Motor

Überblick

In diesem Kurs lernen Sie eine Art Spezialschalter Modul - Relais Modul. Wir werden einen Druckknopf verwenden, um ein Relais zu steuern und den Motor anzutreiben.

Erforderliche Teile

1 x Raspberry Pi

1 x GPIO T-type Erweiterungskarte

1 x Steckbrett

1 x $1k\Omega$ Widerstand

1 x 220Ω Widerstand

1 x LED

1 x Taste

1 x Relais

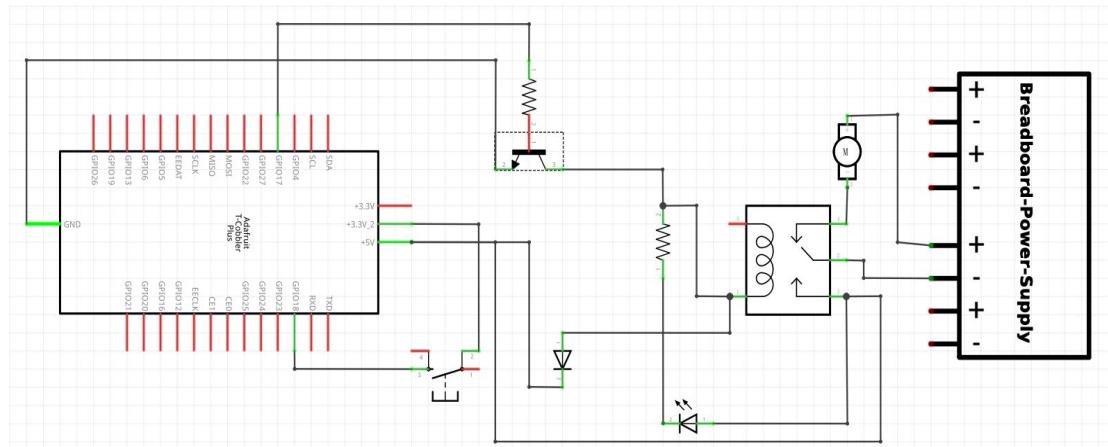
1 x Motor

1 x NPN Transistor

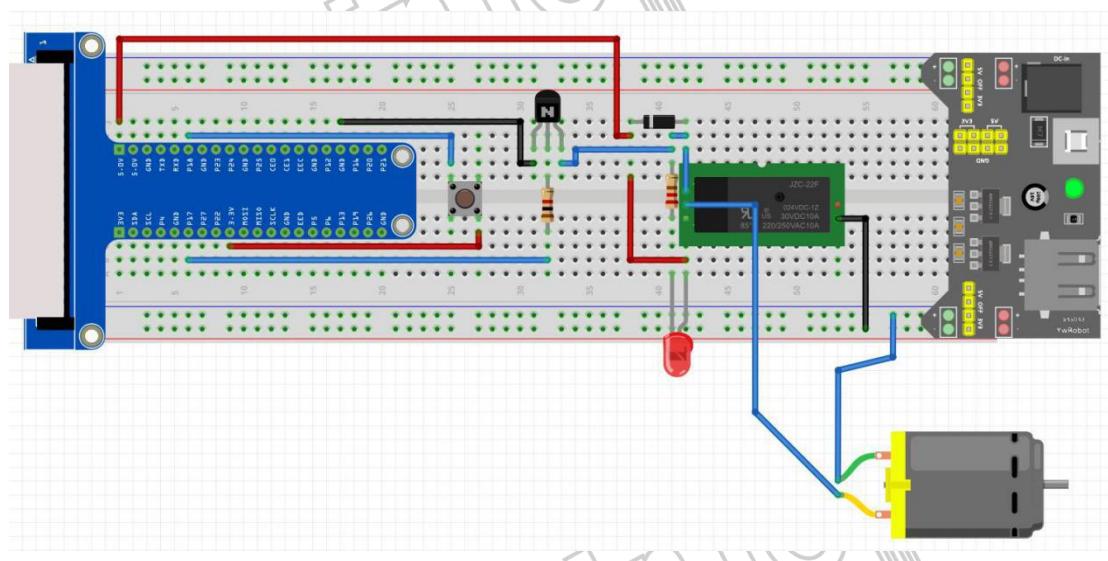
1 x Diode

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm



Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 17.Relay / Relay.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, drücken Sie die Taste einmal, der Motor beginnt sich zu drehen und der Lüfter auf dem Bildschirm dreht sich mit. Der Betriebseffekt ist in der folgenden Abbildung dargestellt:

Relay

Relay & Motor

RelayState: ON**Das Folgendes ist Programmcode:**

```
import processing.io.*;  
  
BUTTON btn;  
int relayPin = 17; //define relayPin  
int buttonPin = 18;  
boolean ledState = false; //define ledState  
int count=0;  
int fign=0;  
float rotaSpeed = 0.02 * PI; //virtual fan's rotating speed,  
float rotaPosition = 0; //motor position  
void setup() {  
    size(640, 360);  
    GPIO.pinMode(relayPin, GPIO.OUTPUT); //set the ledPin to output mode  
    GPIO.pinMode(buttonPin, GPIO.INPUT);  
    btn = new BUTTON(90, height - 90, 50, 30); //define the button  
    btn.setBgColor(0, 255, 0); //set button color  
    btn.setText("OFF"); //set button text  
}  
void draw() {
```

```
background(255, 255, 255);
titleAndSiteInfo();

if(GPIO.digitalRead(buttonPin)==GPIO.HIGH && fign==0)
{
    delay(20);
    if(GPIO.digitalRead(buttonPin)==GPIO.HIGH)
    {
        count++;
    }
    count=count%2;
    if(count==1)
        ledState=true;
    else
        ledState=false;
    fign=1;
}
else if(GPIO.digitalRead(buttonPin)==GPIO.LOW)
{
    fign=0;
}
textAlign(RIGHT, CENTER);
text("RelayState: ", btn.x, btn.y+btn.h/2);
btn.create(); //create the button
if (ledState) {
    GPIO.digitalWrite(relayPin, GPIO.HIGH); //led on
    rotaPosition += rotaSpeed;
    btn.setBgColor(0, 255, 0);
    btn.setText("ON");
} else {
    GPIO.digitalWrite(relayPin, GPIO.LOW); //led off
    btn.setBgColor(255, 0, 0);
    btn.setText("OFF");
}
if (rotaPosition >= 2*PI) {
    rotaPosition = 0;
}
drawFan(rotaPosition);
}

void drawFan(float angle) {
    constrain(angle, 0, 2*PI);
    fill(0);
    for (int i=0; i<3; i++) {
```

```
arc(width/2, height/2, 200, 200, 2*i*PI/3+angle, (2*i+0.3)*PI/3+angle, PIE);
}

fill(0);
ellipse(width/2, height/2, 30, 30);
fill(128);
ellipse(width/2, height/2, 15, 15);
}
void titleAndSiteInfo() {
fill(0);
textAlign(CENTER); //set the text centered
textSize(40); //set text size
text("Relay & Motor", width / 2, 40); //title
textSize(16);
}
```

Code Interpretation

```
void setup() {
size(640, 360);
GPIO.pinMode(relayPin, GPIO.OUTPUT); //set the ledPin to output mode
GPIO.pinMode(buttonPin, GPIO.INPUT);
btn = new BUTTON(90, height - 90, 50, 30); //define the button
btn.setBgColor(0, 255, 0); //set button color
btn.setText("OFF"); //set button text
}
```

Im Funktionssetup () werden Fenster Anzeige und virtuelle Schaltfläche initialisiert.

```
void draw() {
background(255, 255, 255);
titleAndSiteInfo();
if(GPIO.digitalRead(buttonPin)==GPIO.HIGH && fign==0)
{
delay(20);
if(GPIO.digitalRead(buttonPin)==GPIO.HIGH)
{
count++;
}
count=count%2;
if(count==1)
ledState=true;
else
```

```
    ledState=false;
    fign=1;
}

else if(GPIO.digitalRead(buttonPin)==GPIO.LOW)
{
    fign=0;
}
textAlign(RIGHT, CENTER);
text("RelayState: ", btn.x, btn.y+btn.h/2);
btn.create(); //create the button
if (ledState) {
    GPIO.digitalWrite(relayPin, GPIO.HIGH); //led on
    rotaPosition += rotaSpeed;
    btn.setBgColor(0, 255, 0);
    btn.setText("ON");
} else {
    GPIO.digitalWrite(relayPin, GPIO.LOW); //led off
    btn.setBgColor(255, 0, 0);
    btn.setText("OFF");
}
if (rotaPosition >= 2*PI) {
    rotaPosition = 0;
}
drawFan(rotaPosition);

}
```

In der Funktion draw (), ob die Scan-Taste gedrückt wird. Wenn die Taste gedrückt wird, wird beurteilt, ob das Relais geöffnet oder geschlossen werden soll, und der Status des Relais und der virtuellen Taste wird geändert. Zeichnen Sie dann virtuelle Knöpfe und Lüfterblätter.

Lektion 18 Schrittmotor

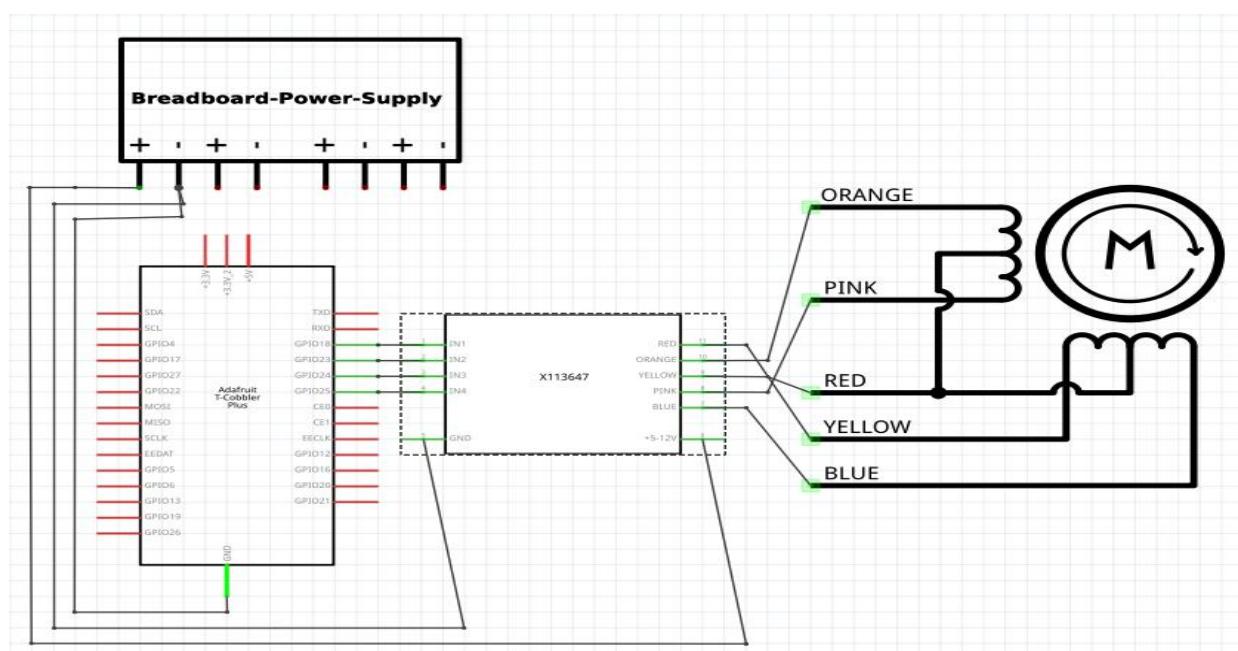
Überblick

In dieser Lektion lernen Sie, wie man die Drehung eines Schrittmotors mit Raspberry pi steuert, und wie man den Drehwinkel eines Schrittmotors steuert. Wir haben DC-Motor und Servo schon einmal gelernt: Der DC-Motor kann sich ständig drehen, aber wir können es nicht in einen bestimmten Winkel drehen lassen. Im Gegenteil, das gewöhnliche Servo kann sich bis zu einem bestimmten Winkel drehen, kann sich aber nicht ständig drehen. In diesem Kapitel lernen wir einen Motor kennen, der sich nicht nur ständig drehen kann, sondern auch zu einem bestimmten Winkel Schrittmotor. Die Verwendung eines Schrittmotors kann leicht zu einer höheren Genauigkeit der mechanischen Bewegung führen.

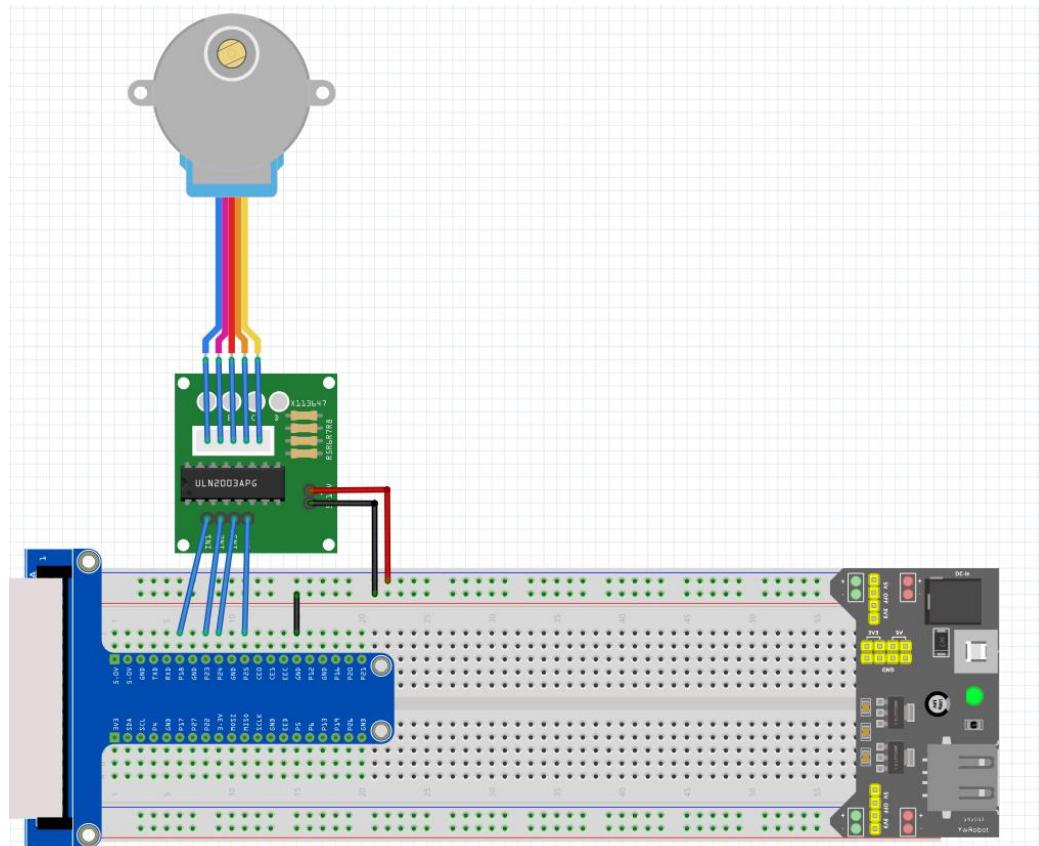
Erforderliche Teile

- 1 x Raspberry Pi
- 1 x GPIO T-Typ Erweiterungskarte
- 1 x Steckbrett
- 1 x Schrittmotor
- 1 x ULN2003 Schrittmotor Treiber
- Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

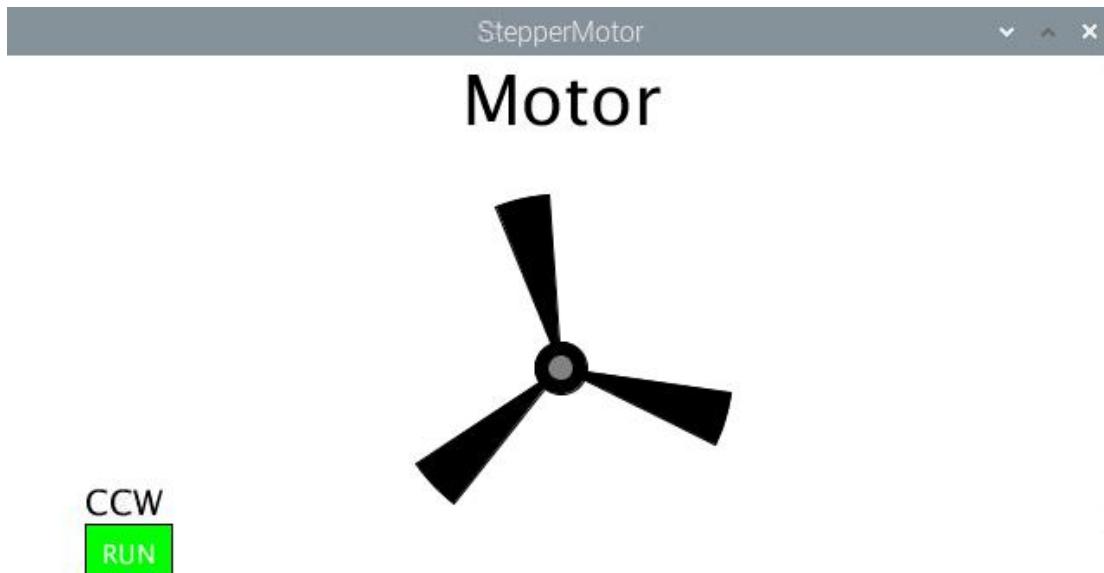


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal `processing code / Java / 18.StepMotor / StepperMotor / StepperMotor.pde` ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, zeigt das Anzeigefenster ein Muster zur Simulation des Motors und eine Schaltfläche zur Steuerung des RUN / STOP Zustands des Schrittmotors. Der Schrittmotor in der Schaltung und der virtuelle Motor im Anzeigefenster beginnen sich gleichzeitig zu drehen.



Das Folgende ist der Programmcode:

```
import processing.io.*;

int[] pins = {18, 23, 24, 25}; //connect to motor phase A,B,C,D pins
BUTTON btn; //BUTTON Object, For controlling the direction of motor
SteppingMotor m = new SteppingMotor(pins);
float rotaSpeed = 0, rotaPosition = 0; //motor speed
boolean isMotorRun = true; //motor run/stop flag

void setup() {
    size(640, 360);
    btn = new BUTTON(45, height - 90, 50, 30); //define the button
    btn.setBgColor(0, 255, 0); //set button color
    btn.setText("RUN"); //set button text
    m.motorStart(); //start motor thread
    rotaSpeed = 0.002 * PI; //virtual fan's rotating speed
}

void draw() {
    background(255);
    titleAndSiteInfo(); //title and site information
}
```

```
btn.create();           //create the button
if (isMotorRun) {      //motor is runnig

fill(0);
textAlign(LEFT,BOTTOM);
textSize(20);
if (m.dir == m.CW) {
    text("CW",btn.x,btn.y);      //text "CW "
    rotaPosition+=rotaSpeed;
    if (rotaPosition>=TWO_PI) {
        rotaPosition = 0;
    }
} else if (m.dir == m.CCW) {
    text("CCW",btn.x,btn.y);    //text "CCW"
    rotaPosition-=rotaSpeed;
    if (rotaPosition<=0) {
        rotaPosition = TWO_PI;
    }
}
if(m.steps<=0) {          //if motor has stopped,
    if(m.dir == m.CCW) {      //change the direction ,restart.
        m.moveSteps(m.CW, 1, 512);
    } else if (m.dir == m.CW) {
        m.moveSteps(m.CCW, 1, 512);
    }
}
drawFan(rotaPosition);    //show the virtual fan in Display window
}

//Draw a clover fan according to the stating angle
void drawFan(float angle) {
    constrain(angle, 0, 2*PI);
    fill(0);
    for (int i=0; i<3; i++) {
        arc(width/2, height/2, 200, 200, 2*i*PI/3+angle, (2*i+0.3)*PI/3+angle, PIE);
    }
    fill(0);
    ellipse(width/2, height/2, 30, 30);
    fill(128);
    ellipse(width/2, height/2, 15, 15);
}

void exit() {
```

```
m.motorStop();
println("exit");

System.exit(0);
}

void mousePressed() {
    if ((mouseY< btn.y+btn.h) && (mouseY>btn.y)
        && (mouseX< btn.x+btn.w) && (mouseX>btn.x)) { // the mouse click the
button
        if (isMotorRun) {
            isMotorRun = false;
            btn.setBgColor(255, 0, 0);
            btn.setText("STOP");
            m.motorStop();
        } else {
            isMotorRun = true;
            btn.setBgColor(0, 255, 0);
            btn.setText("RUN");
            m.motorRestart();
        }
    }
}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);      //set the text centered
    textSize(40);          //set text size
    text("Motor", width / 2, 40); //title
    textSize(16);
}
```

Code Interpretation

```
int[] pins = {18, 23, 24, 25};      //connect to motor phase A,B,C,D pins
BUTTON btn;           //BUTTON Object, For controlling the direction of motor
SteppingMotor m = new SteppingMotor(pins);
```

Definieren Sie in diesem Code zunächst 4 GPIO, die mit dem Motor verbunden sind, das "BUTTON" Klassenobjekt und das "SteppingMotor" Klassenobjekt.

```
void setup() {
    size(640, 360);
    btn = new BUTTON(45, height - 90, 50, 30); //define the button
    btn.setBgColor(0, 255, 0); //set button color
```

```

btn.setText("RUN");           //set button text
m.motorStart();             //start motor thread
rotaSpeed = 0.002 * PI;    //virtual fan's rotating speed

}

```

Im Funktionssetup (), initialisieren Sie den Button, starten Sie den Thread des Schrittmotors, und stellen Sie die Drehzahl des virtuellen Motors ein.

```

background(255);
titleAndSiteInfo(); //title and site information
btn.create(); //create the button
if (isMotorRun) { //motor is runnig
  fill(0);
  textAlign(LEFT,BOTTOM);
  textSize(20);
  if (m.dir == m.CW) {
    text("CW",btn.x,btn.y); //text "CW "
    rotaPosition+=rotaSpeed;
    if (rotaPosition>=TWO_PI) {
      rotaPosition = 0;
    }
  } else if (m.dir == m.CCW) {
    text("CCW",btn.x,btn.y); //text "CCW"
    rotaPosition-=rotaSpeed;
    if (rotaPosition<=0) {
      rotaPosition = TWO_PI;
    }
  }
}

```

In der Funktion draw (), zeichnen Sie zuerst die Taste, berechnen Sie die Position des virtuellen Motors und zeigen Sie die aktuelle Drehrichtung an.

```

if (m.steps<=0) { //if motor has stopped,
  if (m.dir == m.CCW) { //change the direction ,restart.
    m.moveSteps(m.CW, 1, 512);
  } else if (m.dir == m.CW) {
    m.moveSteps(m.CCW, 1, 512);
  }
}
drawFan(rotaPosition); //show the virtual fan in Display window

```

Und dann bestimmen ob sich der Schrittmotor im Stoppzustand befindet nach dem Wert von "m.steps". Wenn dies zutrifft, ändern Sie die Drehrichtung des Motors, und den Motor antreiben, um einen Kreis zu drehen. Endlich zeichnet der virtuelle Lüfter.

Über Klasse Schrittmotor:

Dies ist eine benutzerdefinierte Klasse, die einige Methoden zum Antreiben des Vierphasen-Schrittmotors definiert.

`public SteppingMotor(int[] mPins)`

Konstruktion Funktion, der Parameter für den GPIO-Pin, der an den Schrittmotor angeschlossen ist.

`public void motorStart()`

Starten Sie einen Schrittmotor-Thread, und dann befindet sich der Thread im Wartezustand, warten auf eine Benachrichtigung, um aufzuwachen.

`public void moveSteps(int idir, int ims, int isteps)`

Wird verwendet, um die Drehung des Schrittmotors anzutreiben, der Parameter "idir" für die Richtung, die als CW / CCW eingestellt werden kann. Der Parameter "ims" ist die Verzögerung (mit der Einheit ms) zwischen jeweils zwei Schritten des Schrittmotors, Je höher der Wert von "ims" ist, desto niedriger ist die Geschwindigkeit des Schrittmotors. Der Parameter „isteps“ gibt die Anzahl der Drehschritte des Schrittmotors an. Was den Vierphasen-Schrittmotor betrifft, vier Schritte pro Zyklus, wenn gesetzt isteps = 1, das heißt, geben Sie den Schrittmotor an, der vier Schritte drehen soll.

`public void motorStop()`

Schrittmotor anhalten.

`public void motorRestart()`

Starten Sie den Schrittmotor neu.



Lektion 19 Matrixtastatur

Überblick

In dieser Lektion erfahren Sie, wie man die Matrix Tastatur verwendet.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

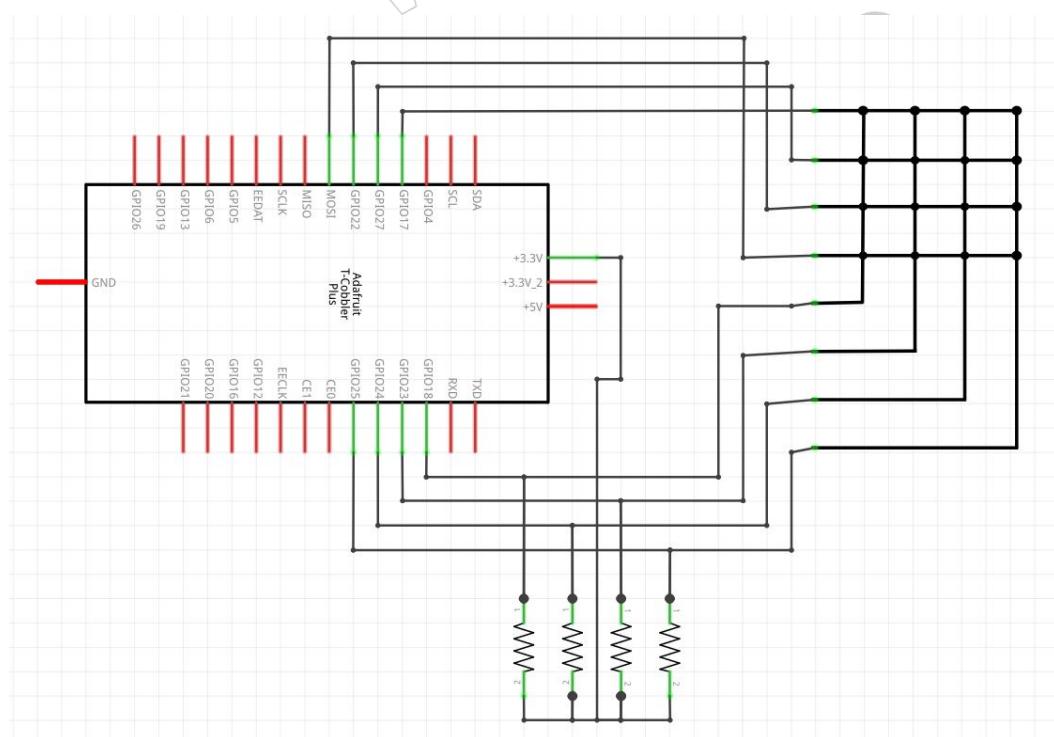
1 x Steckbrett

1 x Matrixtastatur

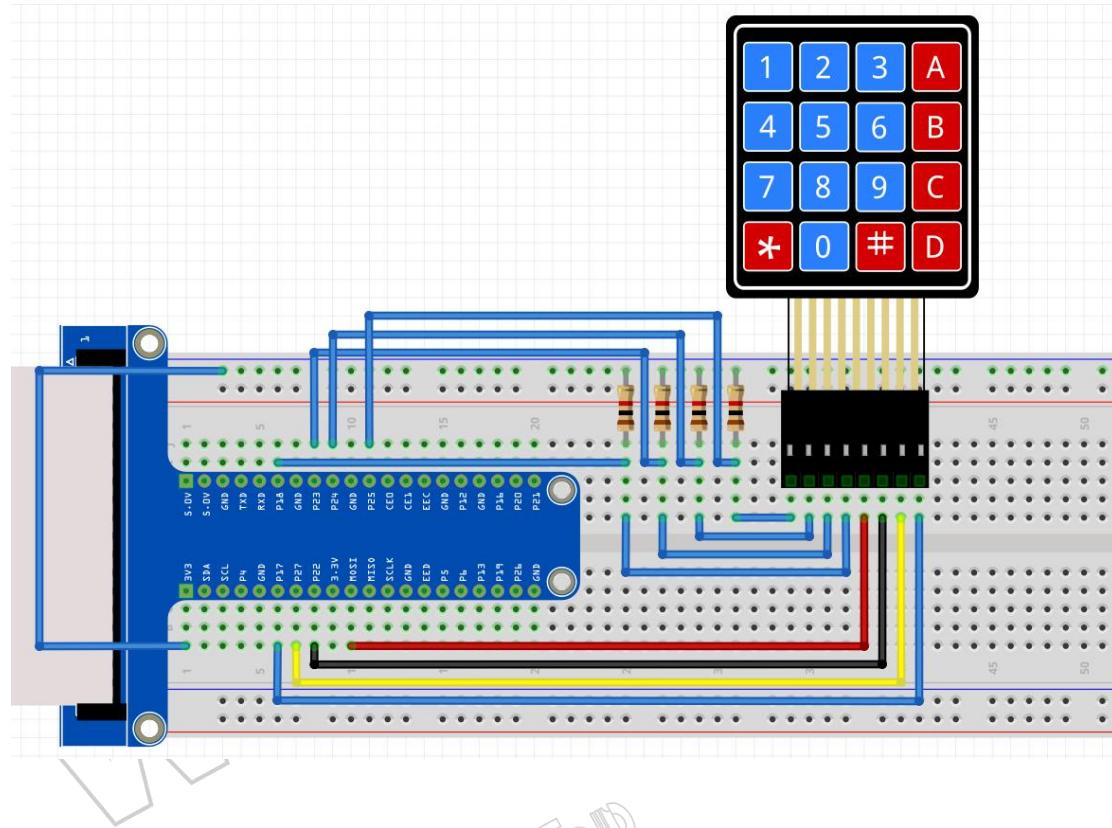
4x 10K Widerstand

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

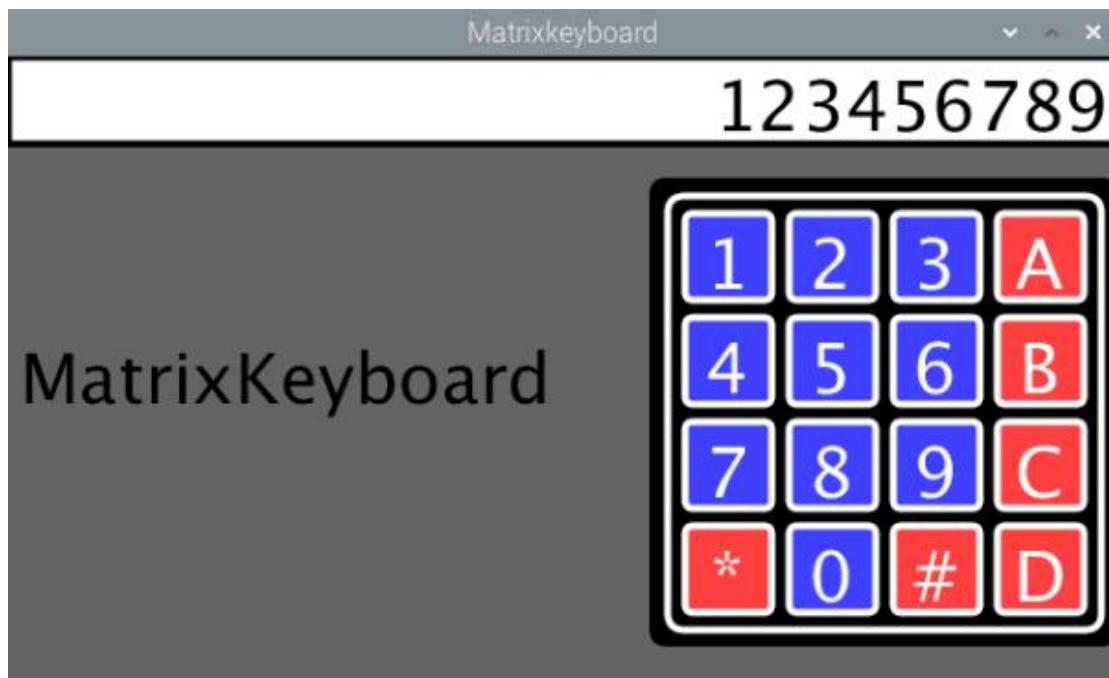


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 19.Matrixkeyboard / Matrixkeyboard / Matrixkeyboard.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, zeigt das Anzeigefenster nach Drücken der Zifferntasten auf der Matrixtastatur die Nummer der von Ihnen gedrückten Taste an.



Das Folgende ist der Programmcode:

```

import processing.io.*;

final static char[]  keys = {  //key code
    '1', '2', '3', 'A',
    '4', '5', '6', 'B',
    '7', '8', '9', 'C',
    '*', '0', '#', 'D'  };
final int[] rowsPins = {18, 23, 24, 25};  //Connect to the row pinouts of the keypad
final int[] colsPins = {10, 22, 27, 17};  //Connect to the column pinouts of the keypad
Keypad kp = new Keypad(keys, rowsPins, colsPins);  //class Object

Calculator cc = new Calculator(kp);  //class Object
void setup() {

size(640, 360);
}
void draw() {
background(102);
titleAndSiteInfo();  //Tile and site information
cc.process();  //Get key and processing
drawDisplay(cc.contentStr);  //Draw display area and content
drawKeypad(width-kpSize, 70);  //draw virtual Keypad
}

```

```

}

void titleAndSiteInfo() {
    fill(0);
    textAlign(CENTER);      //set the text centered
    textSize(40);           //set text size
    text("MatrixKeyboard", width / 4, 200);     //title
    textSize(20);
}

```

Code Interpretation

```

final static char[]  keys = {  //key code
    '1', '2', '3', 'A',
    '4', '5', '6', 'B',
    '7', '8', '9', 'C',
    '*', '0', '#', 'D'  };
final int[] rowsPins = {18, 23, 24, 25}; //Connect to the row pinouts of the keypad
final int[] colsPins = {10, 22, 27, 17}; //Connect to the column pinouts of the keypad
Keypad kp = new Keypad(keys, rowsPins, colsPins); //class Object
Calculator cc = new Calculator(kp); //class Object

```

Definieren Sie im Code zuerst den Schlüsselcode der Tastatur und den mit der Tastatur verbundenen GPIO. Erstellen Sie dann ein Keypad Klassenobjekt basierend auf den Informationen und erstellen Sie schließlich ein Calculator Klassenobjekt gemäß dem Keypad Klassenobjekt.

```

void draw() {
    background(102);
    titleAndSiteInfo(); //Tile and site information
    cc.process(); //Get key and processing
    drawDisplay(cc.contentStr); //Draw display area and content
    drawKeypad(width-kpSize, 70); //draw virtual Keypad
}

```

Verwenden Sie in draw () cc.process (), um den Schlüsselcode der Tastatur abzurufen und zu verarbeiten. Zeichnen Sie dann den Anzeigebereich und die virtuelle Tastatur.

Über die Funktion drawDisplay () und drawKeypad ():

void drawKeypad(int x, int y)

Wird verwendet, um eine Tastatur mit (x, y) für die obere linke Ecke zu zeichnen.

void drawDisplay(String content)

Diese Funktion zeichnet einen Taschenrechner-Anzegebereich am oberen Rand des Fensters und richtet den Anzeigehinhalt rechts vom Bereich aus.

Über die Klasse Schlüssel:

Dies ist eine benutzerdefinierte Klasse, die das zugehörige Attribut definiert, das einem Schlüssel gehört. In dieser Klasse gibt es nur einige Elementvariablen und eine Konstruktionsfunktion.

Über die Klasse Tastatur:

Dies ist eine benutzerdefinierte Klasse, die die Methoden zur Verwendung der Tastatur definiert.

public Keypad(char[] usrKeyMap, int[] row_Pins, int[] col_Pins)

Konstruktions Funktion, die Parameter sind: key code of keyboard, row pins, column pins.

public char getKey()

Holen Sie sich den Schlüsselcode der gedrückten Taste. Wenn keine Taste gedrückt wird, lautet der Rückgabewert '\ 0'.

public void setDebounceTime(int ms)

Stellen Sie die Entprellzeit ein. Die Standardzeit beträgt 10 ms.

public void setHoldTime(int ms)

Stellen Sie die Zeit ein, zu der die Taste nach dem Drücken den stabilen Zustand beibehält.

public boolean isPressed(char keyChar)

Beurteilen Sie, ob die Taste mit dem Code "keyChar" gedrückt wurde.

public char waitForKey()

Warten Sie, bis eine Taste gedrückt wurde, und geben Sie den Tastencode der gedrückten Taste zurück.

public int getState()

Status der Schlüssel abrufen.

boolean keyStateChanged()

Beurteilen Sie, ob sich der Schlüsselstatus geändert hat, und geben Sie dann True oder False zurück.

Über Klasse Taschenrechner:

Dies ist eine benutzerdefinierte Klasse, die die Regeln und Berechnungsmethoden des Rechners definiert.

String contentStr = "";

Member variable, die die aktuellen Verarbeitungsergebnisse des Rechners speichert, die direkt im Anzeigebereich angezeigt werden.

public Calculator(Keypad kp)

Konstruktionsfunktion, der Parameter für das Keypad-Klassenobjekt.

public void process()

Ruft den Schlüsselcode des Schlüssels ab und nimmt die entsprechende Beurteilung und Verarbeitung vor. Die Verarbeitungsergebnisse werden in der Mitgliedsvariablen „contentStr“ gespeichert.

public double parse(String content)

Dies ist der Kern des Rechners. Seine Funktion besteht darin, eine Folge von vier Grundoperationen zu analysieren und einen doppelten Werttyp zurückzugeben. Geben Sie beispielsweise eine Zeichenfolge "1 + 2-3 * 4/5" ein und geben Sie den Wert 0,6 zurück.

Lektion 20 Oszilloskop

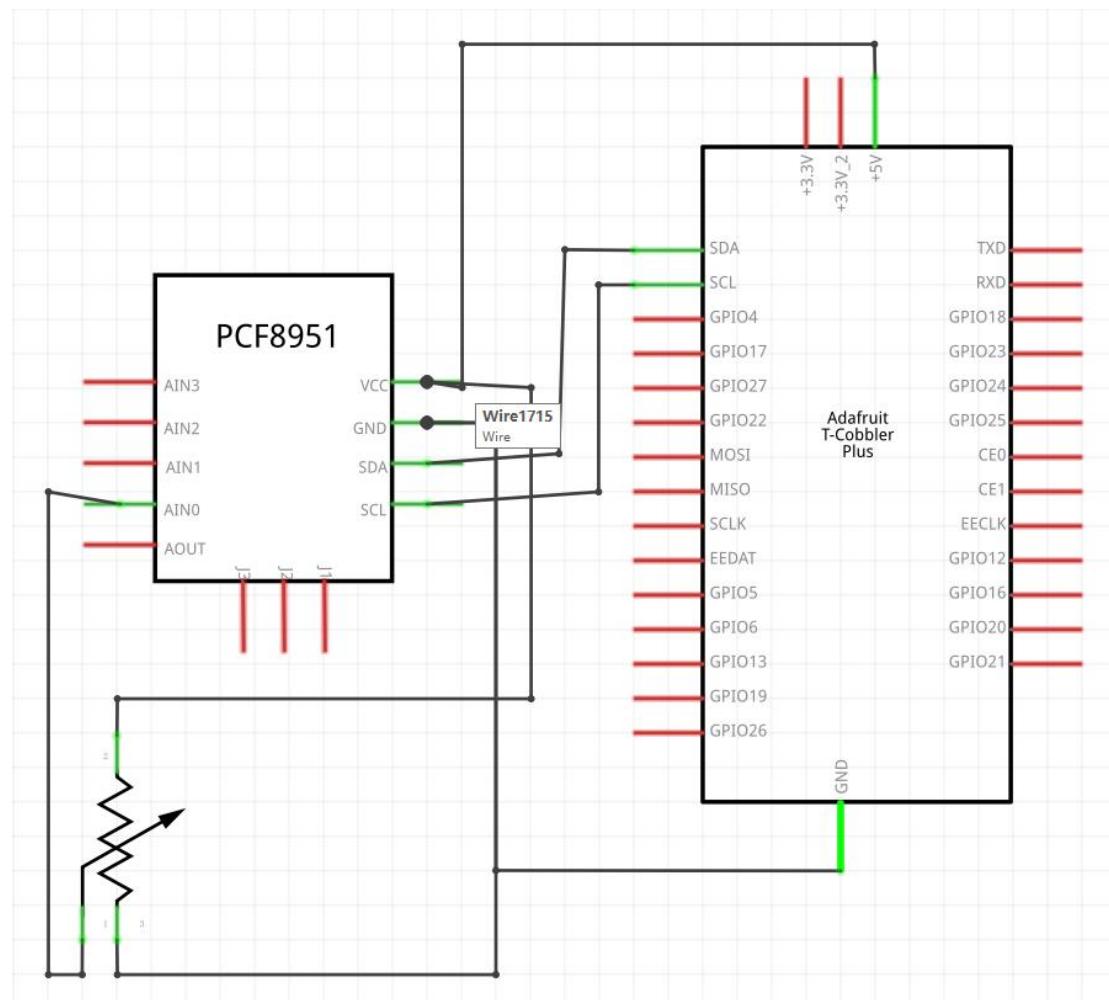
Überblick

Wir haben den PCF8591 verwendet, um die Spannung des Potentiometers abzulesen und die Funktion des Voltmeters vorher zu realisieren. In diesem Kapitel werden wir ein komplexeres Oszilloskop für virtuelle Instrumente erstellen. Das Oszilloskop ist ein weit verbreitetes elektronisches Messgerät. Es kann die nicht direkt beobachteten elektrischen Signale in ein sichtbares Bild bringen, um die Analyse und Untersuchung verschiedener Änderungsprozesse der elektrischen Signale zu erleichtern.

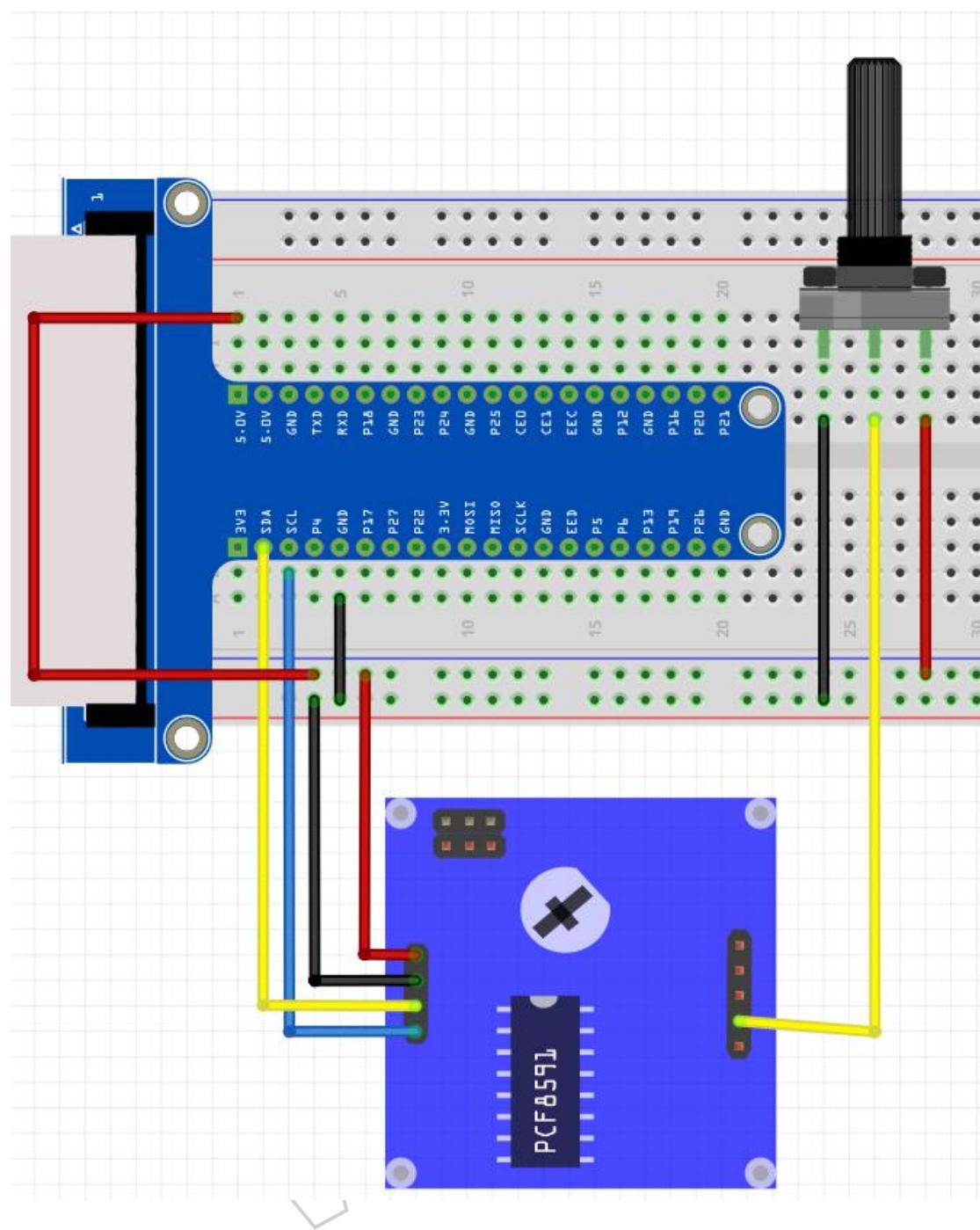
Erforderliche Teile

- 1 x Raspberry Pi
- 1 x Raspberry Pi GPIO Erweiterungskarte
- 1 x Steckbrett
- 1 x Potentiometer
- 1 x PCF8591 Modul
- Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

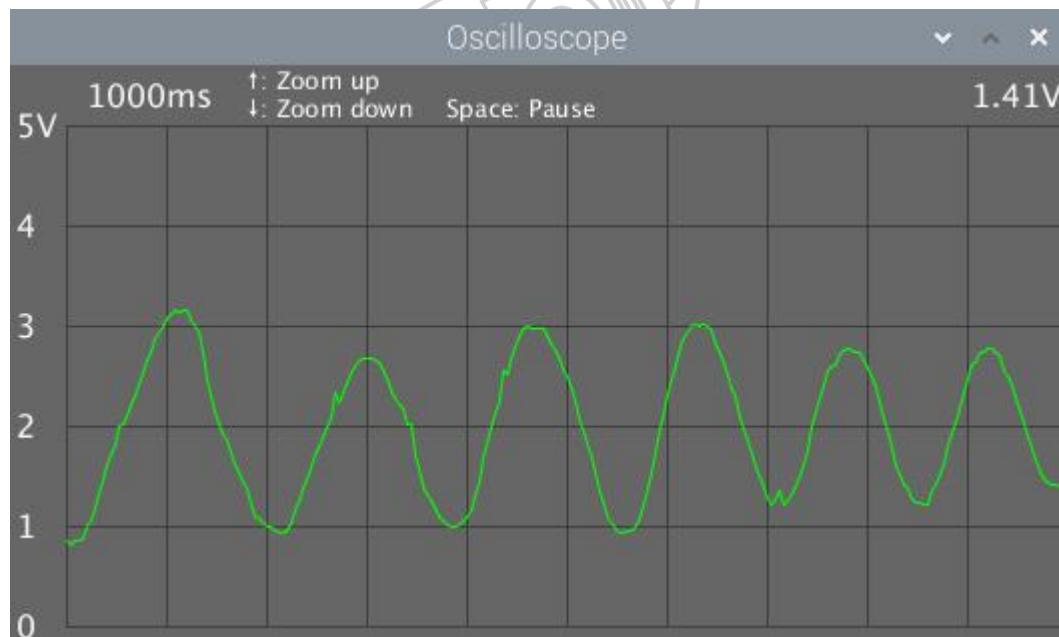


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 20.Oscilloscope / Oscilloscope / Oscilloscope.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Drehen Sie nach Ausführung des Programms das Potentiometer. Die Wellenform des Anzeigefensters ändert sich, wenn sich das Potentiometer dreht, wie nachfolgend dargestellt:



Die linke Seite des Fenster ist eine Spannungsskala, die zur Anzeige der Spannung der Wellenform verwendet wird.

Die "1000ms" in der oberen linken Ecke ist ein Zeitraum, und Sie können die Tasten "↑" und "↓" auf der Tastatur drücken, um sie anzupassen.

Die "1,41 V" in der oberen rechten Ecke ist der Spannungswert des Stromsignals.

Sie können die Leertaste auf der Tastatur drücken, um die Anzeige-Wellenform anzuhalten, die einfach anzuzeigen und zu analysieren ist.

Das Folgende ist der Programmcode:

```
import processing.io.*;  
  
PCF8591 pcf = new PCF8591(0x48);  
int[] analogs; // Analog data send from serial device  
int analogsCount; // Length of analogs[] array  
int voltage = 0; // Voltage  
int hMult = 1; // Horizontal zoom ratio, relative to 1 second  
boolean pause = false; // Storage is suspended display  
  
void setup()  
{  
    size(530, 290);  
    background(102);  
    textAlign(CENTER, CENTER);  
    textSize(64);  
    text("Starting...", width / 2, (height - 40) / 2);  
    textSize(16);  
    textAlign(LEFT, CENTER);  
  
    analogsCount = width / 2;  
    analogs = new int[analogCount];  
    for (int i = 0; i < analogsCount; i++)  
        analogs[i] = -1;  
}  
  
void draw()  
{  
  
    int analog = pcf.analogRead(0); //serialDevice.requestAnalog();  
    if (analog != -1)  
    {  
        // GUI  
        background(102);  
        textSize(12);  
        text("↑: Zoom up", 120, 6);  
        text("↓: Zoom down", 120, 20);  
        text("Space: Pause", 220, 20);  
        textSize(16);  
        // Voltage scale text  
        for (int i = 5; i >= 0; i--)  
        {  
            text(i, 5, 280 - i * 50 - 2);  
    
```

```
if (i == 5)
    text("V", 15, 280 - i * 50 - 2);

}

// Horizontal line
stroke(64, 64, 64);
for (int i = 0; i < 6; i++)
    line(30, 30 + i * 50, width, 30 + i * 50);
// Vertical line time text
text(1000 / hMult + "ms", 40, 15 - 2);
// Vertical line
for (int i = 0; i < (width - 30) / 50 + 1; i++)
    line(30 + i * 50, 30, 30 + i * 50, height - 10);

if (!pause)
{
    // Prepare wave data
    for (int i = 0; i < analogsCount - 1; i++)
        analogs[i] = analogs[i + 1];
    analogs[analogCount - 1] = height - 10 - analog * (height - 10 - 30) / 255;
    // Voltage text
    voltage = analog * 500 / 255;
}
String sVoltage = voltage / 100 + "," + voltage / 10 % 10 + voltage % 10;
text(sVoltage + "V", width - 48, 15 - 2);
// Wave line
stroke(0, 255, 0);
for (int i = width; i > 30; i -= hMult * width / analogsCount)
{
    int a = i / hMult + width * (hMult - 1) / hMult;
    a = a * analogsCount / width - 1;
    if (analog[a] >= 0 && analog[a - 1] >= 0)
        line(i, analogs[a], i - hMult * width / analogsCount, analogs[a - 1]);
}
}

void keyPressed()
{
    if (key == CODED)
    {
        if (keyCode == UP)
```

```
{  
    if (hMult == 1)  
  
        hMult = 2;  
    else if (hMult == 2)  
        hMult = 5;  
    else if (hMult == 5)  
        hMult = 10;  
}  
}  
else if (keyCode == DOWN)  
{  
    if (hMult == 10)  
        hMult = 5;  
    else if (hMult == 5)  
        hMult = 2;  
    else if (hMult == 2)  
        hMult = 1;  
}  
}  
else  
{  
    if(key == ' ')  
    {  
        pause = !pause;  
        if (!pause)  
        {  
            for (int i = 0; i < analogsCount; i++)  
                analogs[i] = -1;  
        }  
    }  
}  
}
```

Lektion 21 Kontroll Grafiken

Überblick

In dieser Lektion lernen wir, den Graphen mit einem Potentiometer zu ändern.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

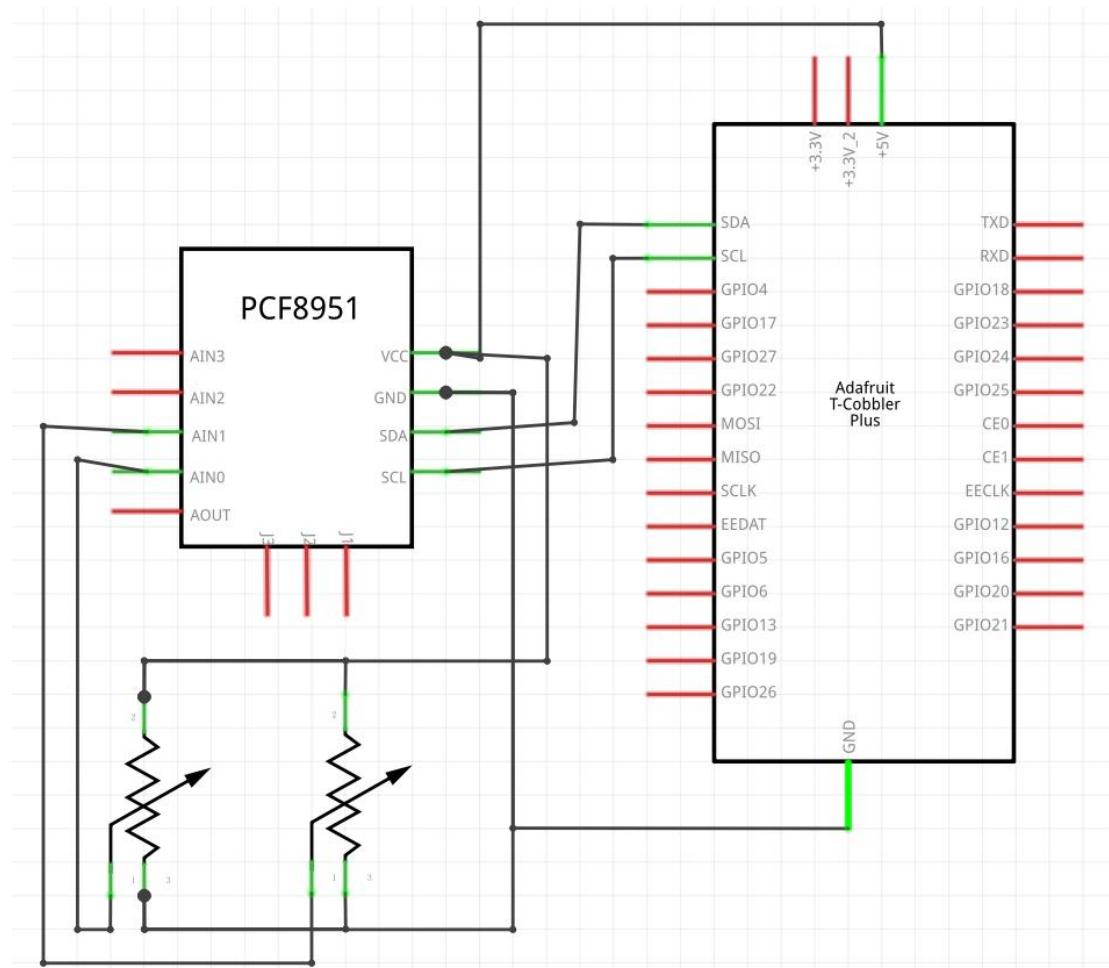
1 x Steckbrett

2x Potentiometers

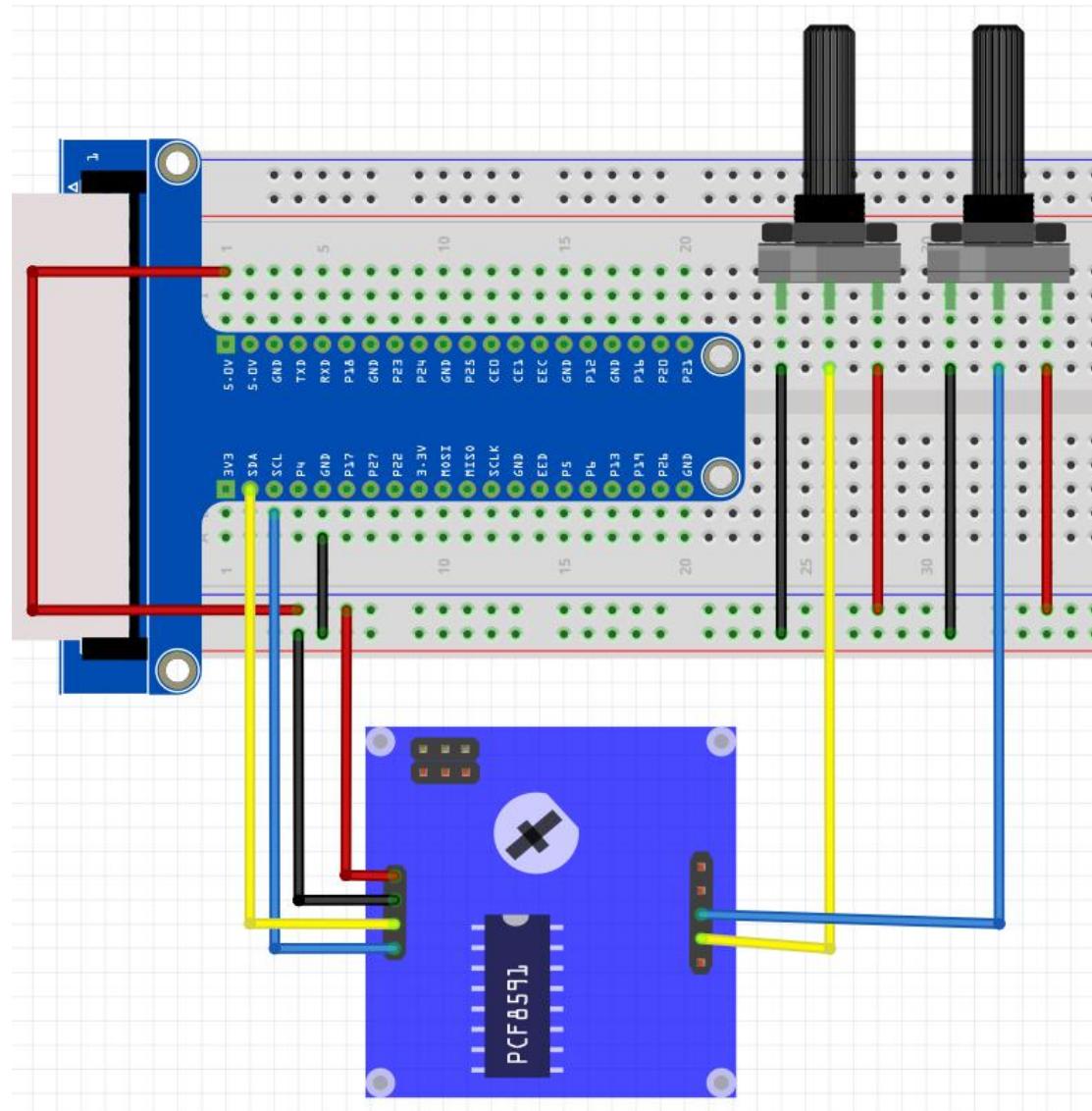
1 x PCF8591 Modul

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm

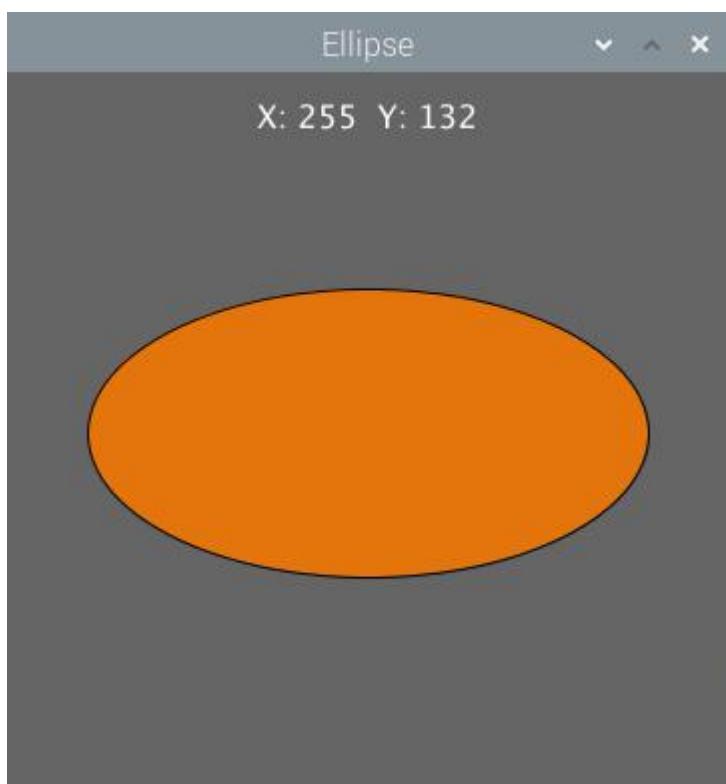


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal `processing code / Java / 21.Ellipse / Ellipse / Ellipse.pde` ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nach Ausführung des Programms kann ein rotierendes Potentiometer die Form und Größe der Ellipse ändern. Wenn die Spannungen der beiden Positionierer gleich sind, wird ein Kreis angezeigt, und wenn nicht, wird eine Ellipse angezeigt.



Das Folgende ist der Programmcode:

```
import processing.io.*;
PCF8591 pcf = new PCF8591(0x48);
void setup()
{
    size(360, 360);
    background(102);
    textAlign(CENTER, CENTER);
    textSize(64);
    text("Starting...", width / 2, (height - 40) / 2);
    textSize(16);
}
```

```
void draw()
{
    int[] analogs = new int[2];

    analogs[0] = pcf.analogRead(0);
    analogs[1] = pcf.analogRead(1);
    if (analogs != null)
    {
        background(102);
        drawEllipse(analogs[0], analogs[1]);
    }
}

void drawEllipse(int x, int y)
{
    int maxDiameter = 280;

    fill(255, 255, 255);
    textAlign(CENTER, CENTER);
    textSize(16);
    text("X: " + x, width / 2 - 30, 20);
    text("Y: " + y, width / 2 + 30, 20);

    x = x * maxDiameter / 255;
    y = y * maxDiameter / 255;
    fill(227, 118, 12);
    ellipse(width / 2, height / 2, x, y);
}
```

Lektion 22 PingPong Spiel

Überblick

In dieser Lektion lernen wir, ein PingPong Spiel mit einem Drehpotentiometer auf der Display Oberfläche zu spielen.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

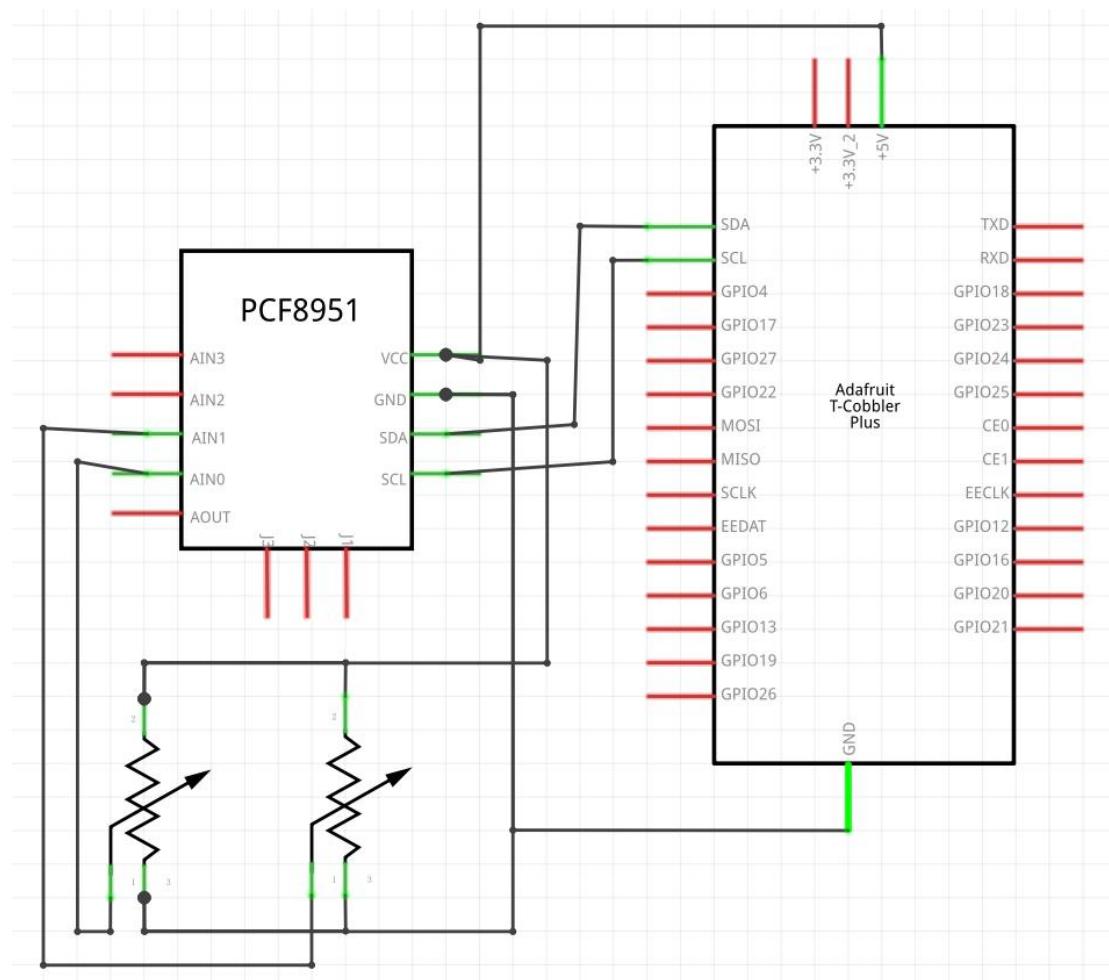
1 x Steckbrett

2x Potentiometers

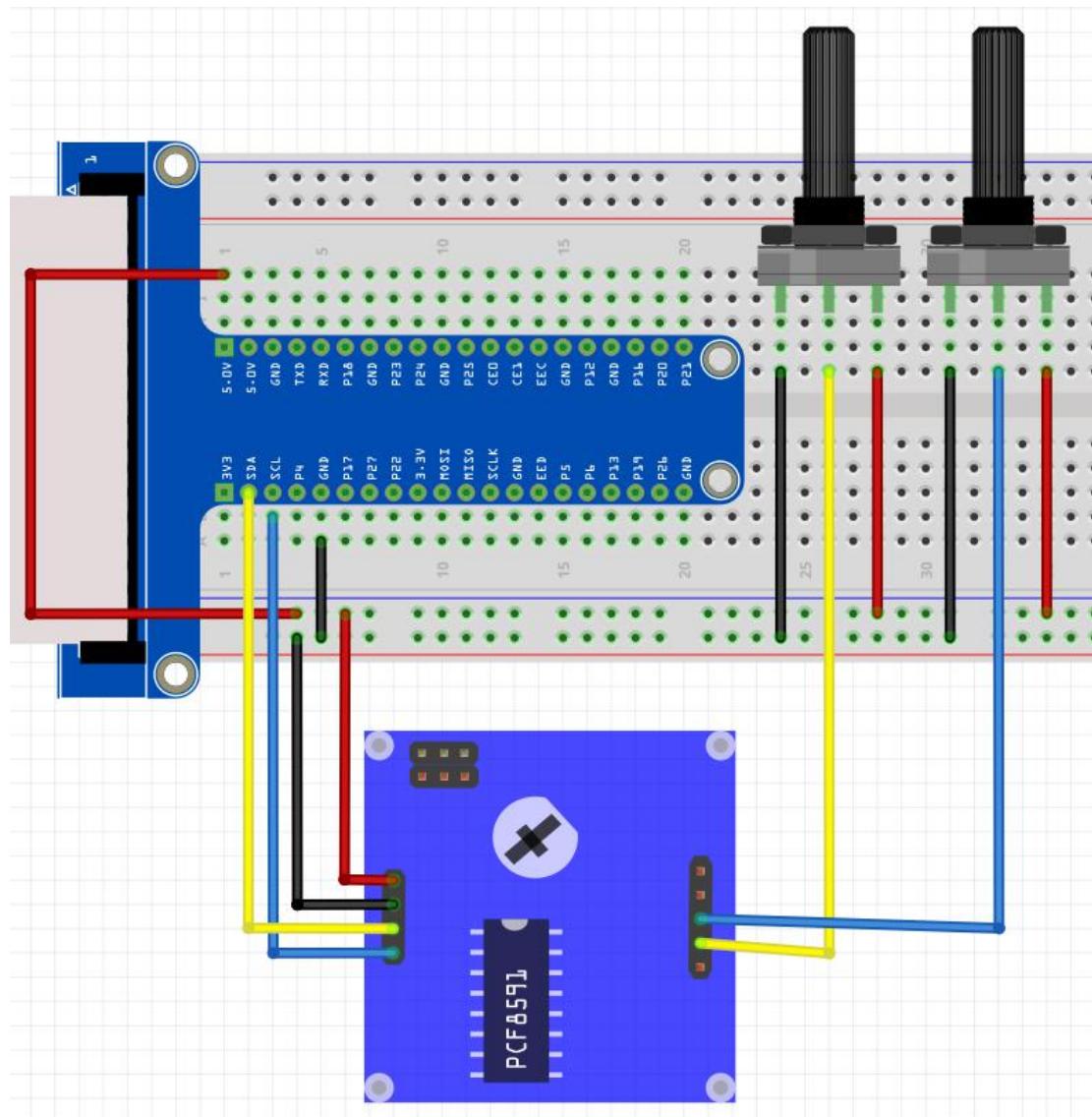
1 x PCF8591 Modul

Einige Jumper Drähte

Schaltplan



Verdrahtung Diagramm



Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 22.Pong_Game / Pong_Game / Pong_Game.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Durch Drücken der Leertaste kann das Spiel gestartet werden. Dann können Sie versuchen, das Potentiometer zu drehen, um die Bewegung des Paddels zu steuern. Verwenden Sie ein Potentiometer, um die Bewegung des Paddels zu steuern und den Ball zurückzudrehen. Die Regeln sind die gleichen wie beim klassischen PingPong Spiel.



Das Folgende ist der Programmcode:

```
import processing.io.*;  
  
PCF8591 pcf = new PCF8591(0x48);  
  
int winScore = 3;  
float acceleration = 0.5;  
float deviate = 1;  
/* Private variables */  
  
Ball ball;  
Paddle lPaddle, rPaddle;  
int gameState = GameState.WELCOME;  
int lScore, rScore;  
  
void setup() {  
    size(640, 360);  
    background(102);
```

```
textAlign(CENTER, CENTER);
textSize(64);
text("Starting...", width / 2, (height - 40) / 2);
textSize(16);

ball = new Ball(10);
lPaddle = new Paddle(new Size(12, 80), 12);
rPaddle = new Paddle(new Size(12, 80), width - 12);

}

void draw() {
    int[] analogs = new int[2];
    analogs[0] = pcf.analogRead(0);
    analogs[1] = pcf.analogRead(1);
    if (analogs != null)
    {
        lPaddle.position.y = analogs[0] * height / 255;
        rPaddle.position.y = analogs[1] * height / 255;
    }

    background(102);
    if (gameState == GameState.WELCOME)
    {
        showGUI();
        lPaddle.display();
        rPaddle.display();
        showInfo("Pong Game");
    }
    else if (gameState == GameState.PLAYING)
    {
        ball.update();
        calculateGame();
        showGUI();
        ball.display();
        lPaddle.display();
        rPaddle.display();
    }
    else if (gameState == GameState.PLAYER1WIN)
    {
        showGUI();
        lPaddle.display();
```

```
rPaddle.display();
    showInfo("Player 1 win!");
}
else if (gameState == GameState.PLAYER2WIN)
{
    showGUI();
    lPaddle.display();
    rPaddle.display();
    showInfo("Player 2 win!");
}

void showInfo(String info)
{
    rectMode(CENTER);
    stroke(0, 0, 0);
    fill(0, 0, 0, 50);
    rect(width / 2, height / 2, width / 2, height / 3);
    fill(255, 255, 255);
    textSize(24);
    textAlign(CENTER, CENTER);
    text(info, width / 2, height / 2 - 24);
    text("Press Space to start", width / 2, height / 2 + 24);
}

void calculateGame()
{
    if (ball.position.x - ball.radius < lPaddle.position.x + lPaddle.size.width / 2)
    {
        if (ball.position.y < lPaddle.position.y - lPaddle.size.height / 2 - ball.radius ||
            ball.position.y > lPaddle.position.y + lPaddle.size.height / 2 + ball.radius)
        {
            rScore++;
            ball.reset();
        }
    }
    else
    {
        ball.speed.getSpeed();
        ball.speed.speed += acceleration;
        ball.speed.getXYSpeed((ball.position.y - lPaddle.position.y) /
(IPaddle.size.height / 2) * deviate);
    }
}
```

```
        }

    }

    if (ball.position.x + ball.radius > rPaddle.position.x - rPaddle.size.width / 2)
    {
        if (ball.position.y < rPaddle.position.y - rPaddle.size.height / 2 - ball.radius ||
            ball.position.y > rPaddle.position.y + rPaddle.size.height / 2 + ball.radius)
        {
            lScore++;
            ball.reset();
        }
        else
        {
            ball.speed.getSpeed();
            ball.speed.speed += acceleration;
            ball.speed.getXYSpeed((ball.position.y - rPaddle.position.y) / (rPaddle.size.height / 2) * deviate);
            ball.speed.x = - ball.speed.x;
        }
    }

    if (lScore == winScore)
        gameState = GameState.PLAYER1WIN;
    if (rScore == winScore)
        gameState = GameState.PLAYER2WIN;
}

void showGUI()
{
    fill(255, 255, 255);
    textSize(16);
    textAlign(CENTER, CENTER);
    text("Press Space to restart game", width * 3 / 4, height - 20);
    text("Player 1: " + lScore, width / 4, 20);
    text("Player 2: " + rScore, width * 3 / 4, 20);

    rectMode(CENTER);
    noStroke();
    fill(144, 144, 144);
    rect(width / 2, height / 2, 4, height);
}
```

```
void keyPressed() {  
    if (key == ' ')  
    {  
        lScore = 0;  
        rScore = 0;  
        ball.reset();  
        gameState = GameState.PLAYING;  
    }  
}
```

Lektion 23 Schlangenspiel

Überblick

In dieser Lektion lernen wir, das Schlangenspiel mit den Tasten auf der Display Oberfläche zu spielen.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

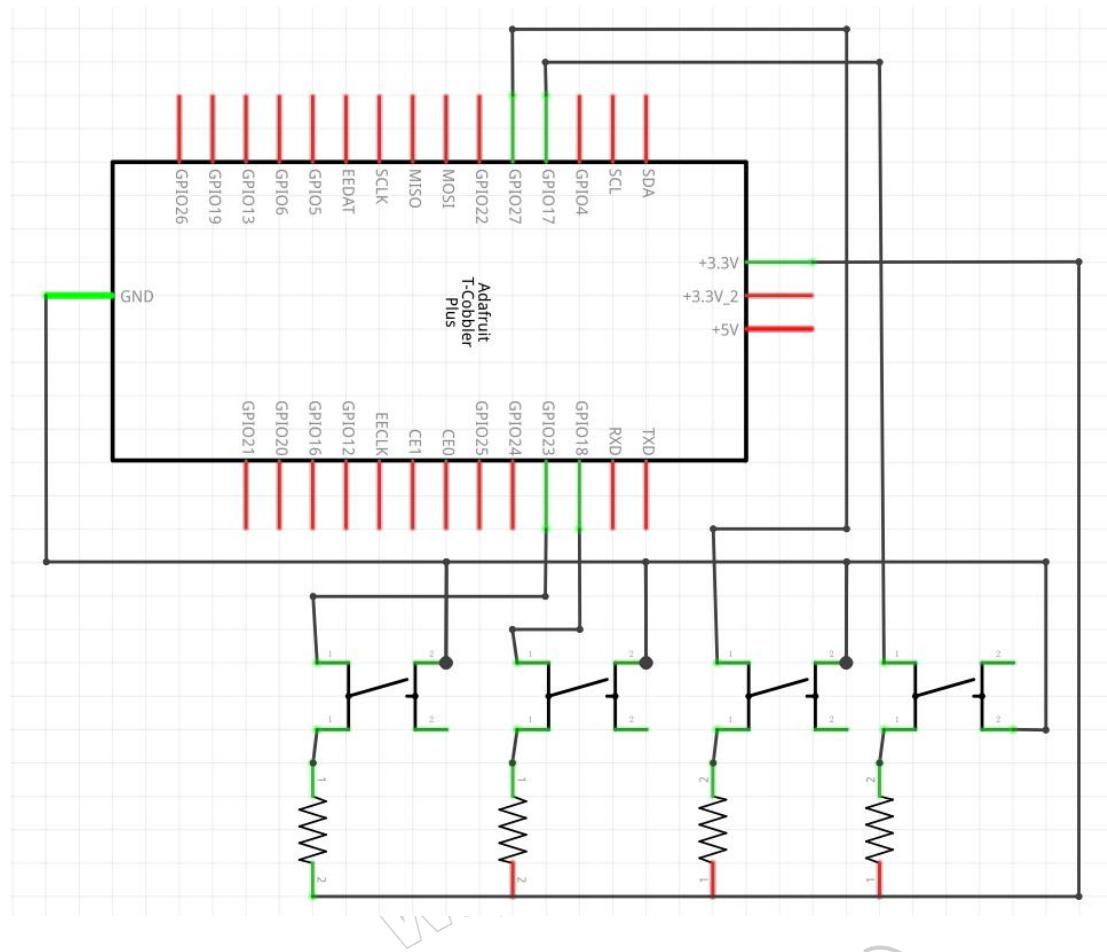
4x 10K Widerstand

4 x Tasten

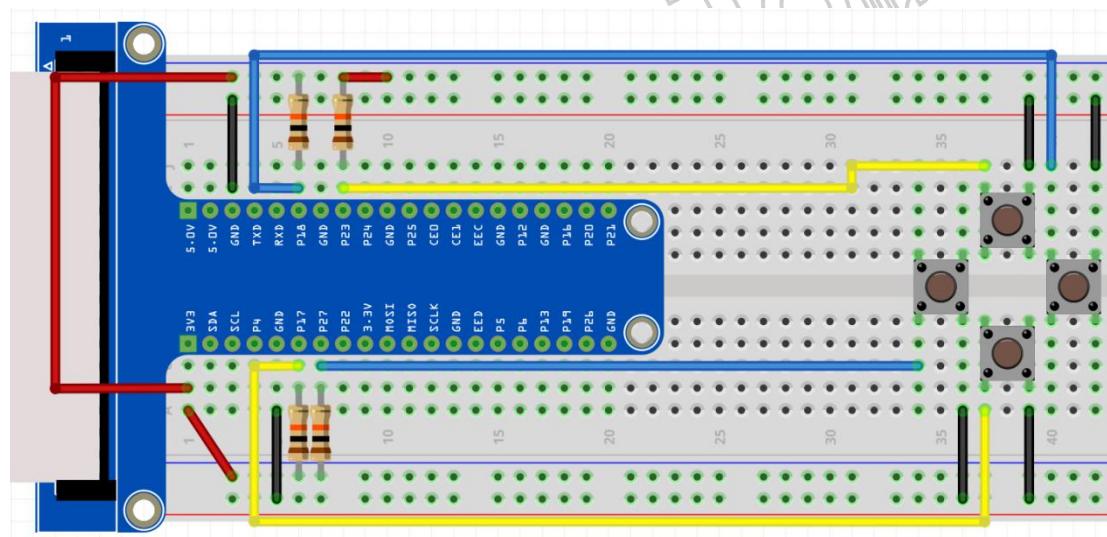
Einige Jumper Drähte



Schaltplan



Verdrahtung Diagramm



Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 23.Snake_Game / Snake_Game / Snake_Game.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Nachdem das Programm ausgeführt wurde, können Sie durch Drücken der Leertaste das Spiel starten. Sie können die Bewegungsrichtung der Schlange über die vier Tasten im Schaltkreis oder die vier Pfeiltasten auf der Tastatur steuern.



Das Folgende ist der Programmcode:

```
import processing.io.*;  
int threshold = 400;  
  
KeyPad keyUp = new KeyPad(23);  
KeyPad keyDown = new KeyPad(17);  
KeyPad keyLeft = new KeyPad(22);  
KeyPad keyRight = new KeyPad(18);
```

```
Snake snake;
Food food;

void setup() {
    print("Starting ... \n");
    size(640, 360);
    background(102);
    textAlign(CENTER, CENTER);
    textSize(64);
    text("Starting...", width / 2, (height - 40) / 2);
    textSize(16);

    food = new Food(new GridMap(new Size(width, height), 20, 2));
    snake = new Snake(new GridMap(new Size(width, height), 20, 2));
    thread("keypadDetect");
}

void draw() {
    background(102);
    if(snake.gameState == GameState.WELCOME)
    {
        rectMode(CENTER);
        stroke(0, 0, 0);
        fill(0, 0, 0, 50);
        rect(width / 2, height / 2, width / 2, height / 3);
        fill(255, 255, 255);
        textSize(24);
        textAlign(CENTER, CENTER);
        text("Snake Game", width / 2, height / 2 - 24);
        text("Press Space to start", width / 2, height / 2 + 24);
    } else if (snake.gameState == GameState.PLAYING)
    {

        if (snake.body[0].x == food.position.x && snake.body[0].y == food.position.y)
        {
            snake.grow();
            food.generate(snake.body, snake.length);
            snake.speedUp();
        }
        snake.step();
        showGame();
    } else if (snake.gameState == GameState.LOSE)
}
```

```
{  
    showGame();  
    rectMode(CENTER);  
    stroke(0, 0, 0);  
    fill(0, 0, 0, 50);  
    rect(width / 2, height / 2, width / 2, height / 3);  
    fill(255, 255, 255);  
    textSize(24);  
    textAlign(CENTER, CENTER);  
    text("You lose!", width / 2, height / 2 - 24);  
    text("Press Space to start", width / 2, height / 2 + 24);  
}  
}  
  
void showGame()  
{  
    snake.display();  
    food.display();  
  
    fill(255, 255, 255);  
    textSize(16);  
    textAlign(LEFT, CENTER);  
    textAlign(RIGHT, CENTER);  
    text("Press Space to restart game", width - 20, height - 20);  
    textAlign(LEFT, CENTER);  
    text("Score: " + (snake.length - 3), 20, 20);  
    textAlign(RIGHT, CENTER);  
    text("Speed: " + ((snake.initSpeed - snake.speed) / 5 + 1), width - 20, 20);  
}  
  
void keyPressed() {  
    if ((key == CODED) || (keyValue != -1))  
    {  
        if ((keyCode == UP) || ((keyValue == keyUp.pin)))  
        {  
            if (snake.direction != Direction.DOWN)  
                snake.nextDirection = Direction.UP;  
        } else if ((keyCode == DOWN) || ((keyValue == keyDown.pin))) {  
            if (snake.direction != Direction.UP)  
                snake.nextDirection = Direction.DOWN;  
        } else if ((keyCode == LEFT) || ((keyValue == keyLeft.pin))) {  
            if (snake.direction != Direction.RIGHT)
```

```
snake.nextDirection = Direction.LEFT;
} else if ((keyCode == RIGHT)||((keyValue ==  keyRight.pin))) {
    if (snake.direction != Direction.LEFT)
        snake.nextDirection = Direction.RIGHT;
}
//keyValue = -1;
println(keyValue);
} else
{
    if (key == ' ')
    {
        snake.reset();
        food.generate(snake.body, snake.length);
        snake.gameState = GameState.PLAYING;
    }
}
void keypadDetect() {
    while (true) {
        keyUp.keyScan();
        keyDown.keyScan();
        keyLeft.keyScan();
        keyRight.keyScan();
        transAction();
        try {
            Thread.sleep(10);
        }
        catch(Exception e) {
        }
    }
}
void transAction() {
    if ((keyValue != -1))
    {
        if (keyValue ==  keyUp.pin)
        {
            if (snake.direction != Direction.DOWN)
                snake.nextDirection = Direction.UP;
        } else if (((keyValue ==  keyDown.pin))) {
            if (snake.direction != Direction.UP)
                snake.nextDirection = Direction.DOWN;
        } else if (((keyValue ==  keyLeft.pin))) {
```

```
if (snake.direction != Direction.RIGHT)
    snake.nextDirection = Direction.LEFT;
} else if (((keyValue == keyRight.pin))) {
    if (snake.direction != Direction.LEFT)
        snake.nextDirection = Direction.RIGHT;
}
keyValue = -1;
}
```

Lektion 24 Tetris Game

Überblick

In dieser Lektion lernen wir, das Tetris Spiel mit den Tasten auf der Benutzeroberfläche zu spielen.

Erforderliche Teile

1 x Raspberry Pi

1 x Raspberry Pi GPIO Erweiterungskarte

1 x Steckbrett

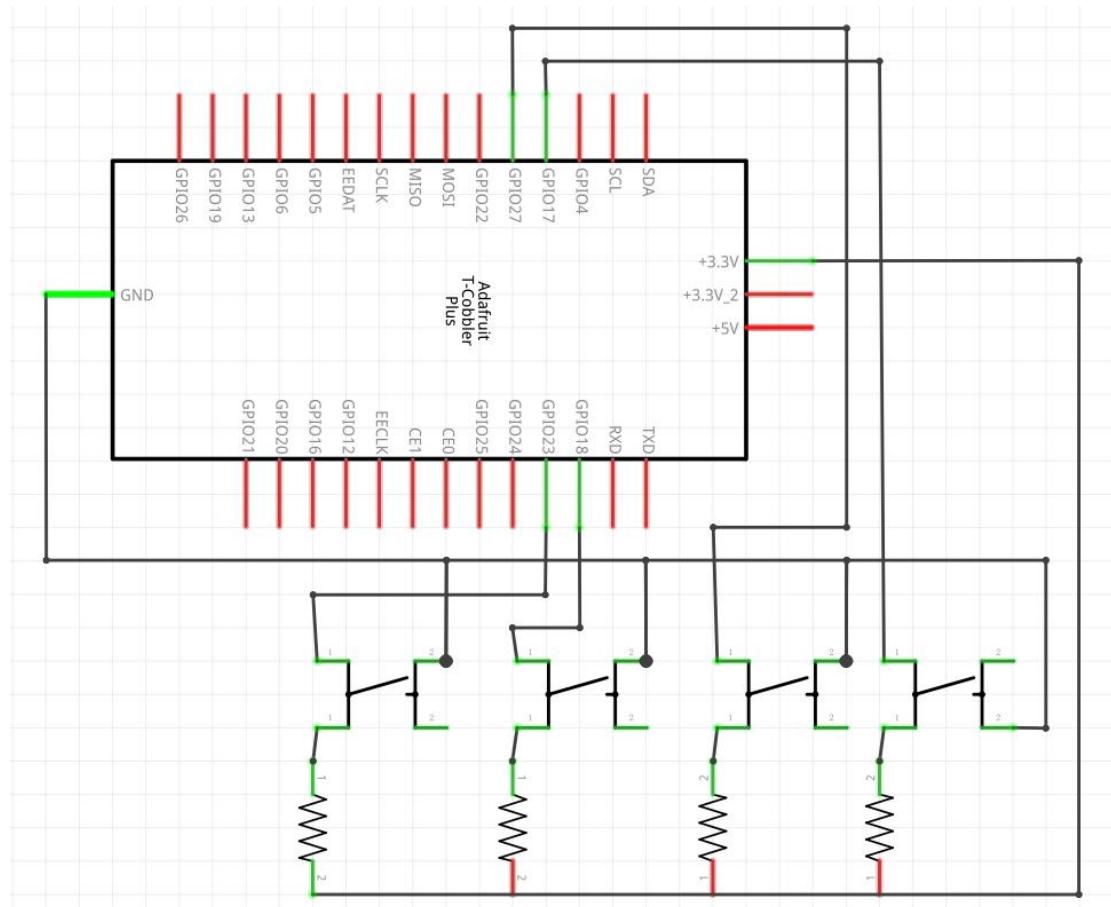
4x 10K Widerstand

4 x Tasten

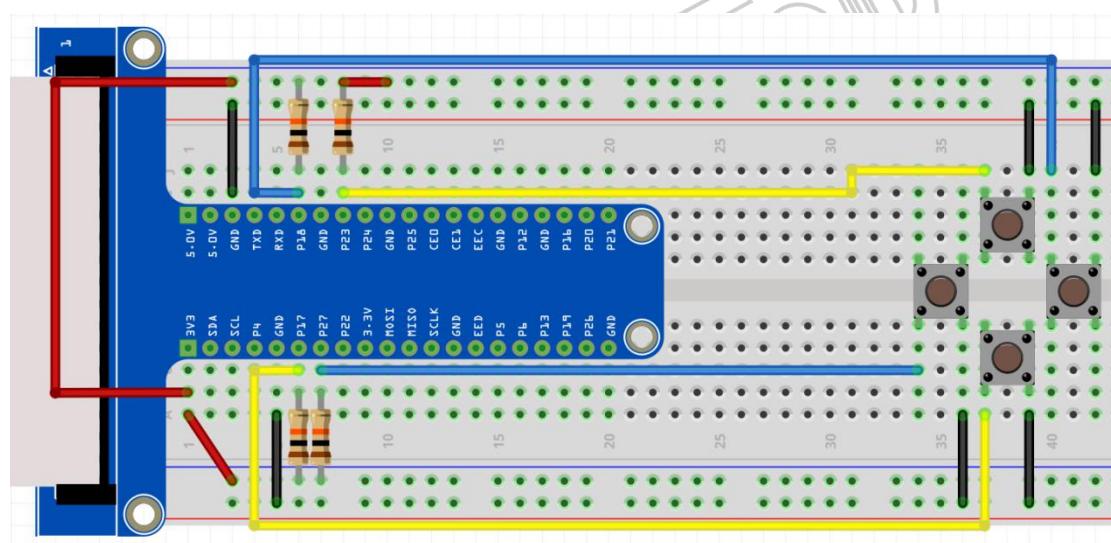
Einige Jumper Drähte



Schaltplan



Verdrahtung Diagramm

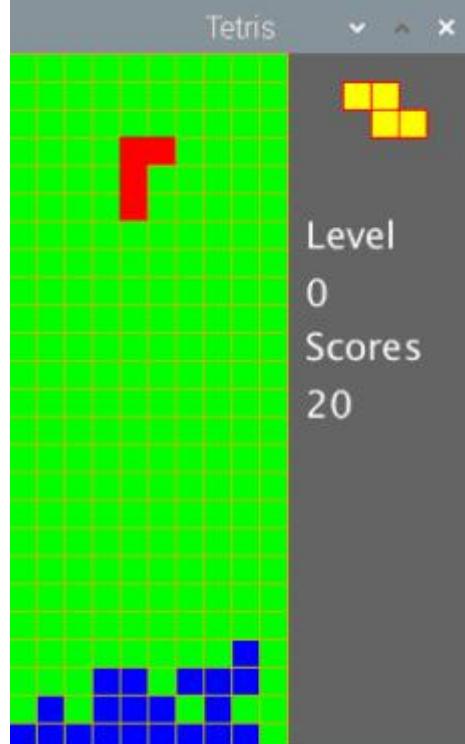


Code

Beobachten Sie zuerst das laufende Ergebnis der Skizze und analysieren Sie dann den Code.

1. Geben Sie in das Terminal [processing code / Java / 24.Tetris / Tetris.pde](#) ein, um den Code zu öffnen;
2. Klicken Sie in der Popup-Software "processing" auf die Schaltfläche "RUN", um den Code auszuführen;

Drücken Sie nach dem Ausführen des Programms die Leertaste des Computers, um das Spiel zu starten.



Die linke und rechte Taste in der Schaltung können die Bewegung des fallenden Blocks nach links oder rechts steuern. Und der Knopf unten kann das Herunterfallen des Blocks beschleunigen. Die obige Schaltfläche dient zum Drehen des Tetris Blocks. Vier Richtungstasten auf der Tastatur können auch zum Spielen des Spiels verwendet werden.

Während des Spiels kann das Drücken der Leertaste auf der Tastatur das Spiel unterbrechen. Die rechte Seite des Anzeigefensters zeigt den nächsten bevorstehenden Tetris, die aktuelle Spielgeschwindigkeit und die aktuelle Punktzahl. Je mehr Zeilen Sie einmal entfernen, desto höher sind die Punktzahlen. Wenn sich die Tetris

außerhalb des Bildschirms befinden, ist das Spiel beendet. Drücken Sie nach dem Spiel die Leertaste, um ein neues Spiel zu starten.

Das Folgende ist der Programmcode:

```
import processing.io.*;  
  
static final int w = 10; // 4  
static final int h = 25; // 60  
static final int framesInSecond = 30;  
static float gameInitSpeed = 10;  
static float gameSpeed = 10;  
static final int BlockScale = 15;  
static final int sizeWidth = w*BlockScale+100;  
static final int sizeHeight = h*BlockScale;  
  
KeyPad keyUp = new KeyPad(23);  
KeyPad keyDown = new KeyPad(17);  
KeyPad keyLeft = new KeyPad(22);  
KeyPad keyRight = new KeyPad(18);  
  
boolean isPaused = false;  
boolean keyAllow = true;  
float updatingThreshold = 0;  
Game game;  
  
float recalculateUpdatingThreshold(float threshold) {  
    return threshold + 1;  
}  
void settings() {  
    size(sizeWidth, sizeHeight);  
}  
void setup() {  
    game = new Game(w, h);  
    generateRandomBlock(game);  
    frameRate(framesInSecond);  
    thread("keypadDetect");  
}
```

```
void draw() {  
  
    background(102);  
    Game newGame = game;  
  
    updatingThreshold = recalculateUpdatingThreshold(updatingThreshold);  
    if (updatingThreshold > gameSpeed) {  
        if (isGameOver(newGame)) {  
        } else if (isPaused) {  
        } else {  
            newGame = updateGameState(game);  
        }  
        updatingThreshold = 0;  
    }  
    drawGameState(newGame);  
    if (!isGameOver(newGame)&& (isPaused)) { //pause  
        textSize(40);  
        fill(0);  
        text("Pause", BlockSeale*2, 150);  
        keyAllow = false;  
    } else if (isGameOver(newGame)&& (isPaused)) { //restart game  
        game = new Game(w, h);  
        generateRandomBlock(game);  
        isPaused = false;  
        keyAllow = false;  
    } else if (isGameOver(newGame)) { //game over  
        textSize(40);  
        fill(0);  
        text("Game \nOver", BlockScale*2, 150);  
        keyAllow = false;  
    } else { //playing  
        keyAllow = true;  
    }  
  
    //level,score information  
    pushMatrix();  
    translate(w*BlockScale, 0);  
    fill(255);  
    textSize(20);  
    text("Level\n"+game.level, 10, BlockScale*7);  
    text("Scores\n"+game.score, 10, BlockScale*11);  
    textSize(12);
```



```
    drawNextBlock(game.nextBlock, BlockScale*2, BlockScale*1);
    popMatrix();
}

void keyPressed() {

    if (key == CODED) {
        if (keyAllow) {
            switch (keyCode) {
                case LEFT:
                    moveBlock(game, MoveLeft);
                    break;
                case RIGHT:
                    moveBlock(game, MoveRight);
                    break;
                case DOWN:
                    makeBlockFall(game);
                    break;
                case UP:
                    rotateBlock(game);
                    break;
            }
        }
    } else if (key == ' ') { // SPACE
        isPaused =! isPaused;
    }
}

void keypadDetect() {
    while (true) {
        keyUp.keyScan();
        keyDown.keyScan();
        keyLeft.keyScan();
        keyRight.keyScan();
        transAction();
        try {
            Thread.sleep(10);
        }
        catch(Exception e) {
        }
    }
}

void transAction() {
    if ((keyValue != -1))
```

```
{  
    if (keyAllow) {  
        if (keyValue == keyLeft.pin) {  
            moveBlock(game, MoveLeft);  
        } else if (keyValue == keyRight.pin) {  
  
            moveBlock(game, MoveRight);  
        } else if (keyValue == keyDown.pin) {  
            makeBlockFall(game);  
        } else if (keyValue == keyUp.pin) {  
            rotateBlock(game);  
        }  
    }  
    try {  
        Thread.sleep(50);  
    }  
    catch(Exception e){  
    }  
    keyValue = -1;  
}
```