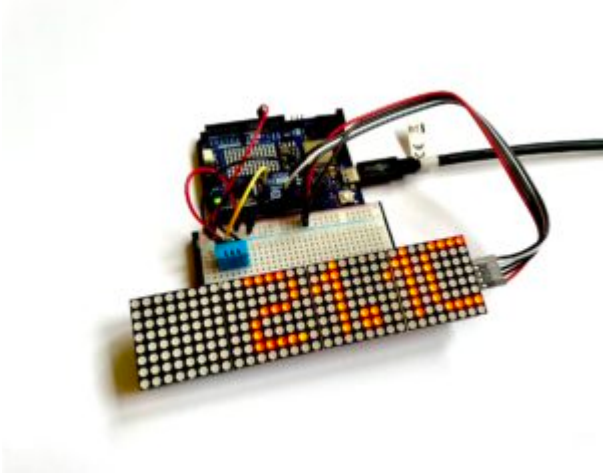
So erzeugst du Laufschrift auf einer LED-Matrix



Eine LED-Matrix mit Laufschrift ist ein tolles Projekt für alle, die sich mit Displays und Effekten am Arduino beschäftigen möchten. Mit nur wenigen Bauteilen lässt sich ein beeindruckender Effekt erzielen. In diesem Projekt lernst du zunächst, wie du einen Text als **Laufschrift über die Matrix wandern** lässt. Danach machst du das Projekt etwas dynamischer: Du kannst den **Text über den Seriellen Monitor aktualisieren.** Und zuletzt lässt du die LED-Matrix aktuelle Daten vom Temperatur- und Luftfeuchtigkeitsensor DHT11 anzeigen.

Was ist eine LED-Matrix?

Eine LED-Matrix ist ein Display aus vielen kleinen Leuchtdioden (LEDs), die in Reihen und Spalten angeordnet sind – meist in 8×8 Feldern. Die Ansteuerung übernimmt ein spezieller Treiberchip, z. B. der **MAX7219**.

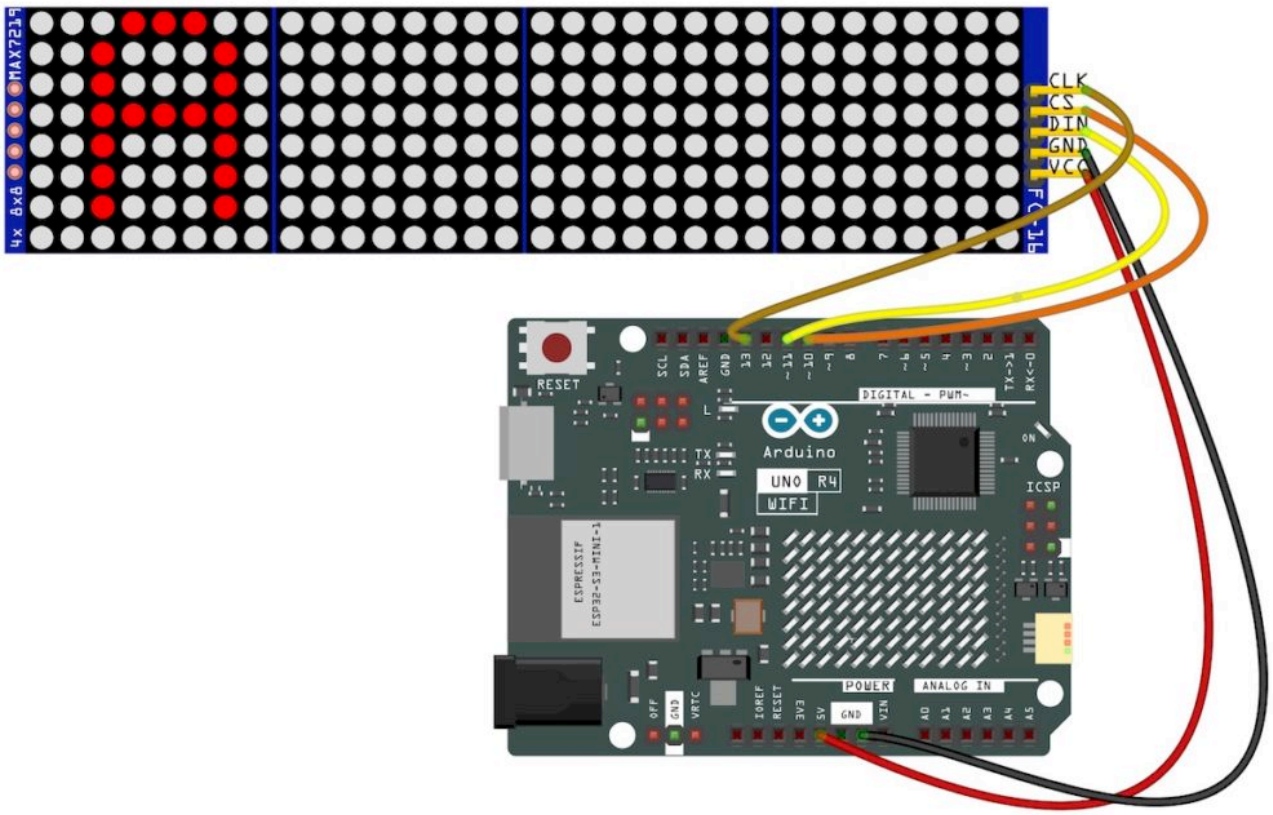
Diese Matrizen gibt es einzeln (also mit den erwähnten 8×8 Feldern) zu kaufen. Ein Projekt, das diese Matrix verwendet ist z.B. das Babyphone. Damit wäre eine Laufschrift allerdings sehr schwer zu lesen. Besser eignen sich hierfür Module, bei denen mehrere dieser Quadrate aneinandergereiht sind – im Folgenden sind das vier.

Der Anschluss am Arduino

Für dieses Projekt brauchst du diese Hardware:

- 1× Arduino UNO oder ein kompatibles Board
- 1× LED-Matrix mit MAX7219 (4 Module)
- Breadboard und Kabel
- (optional) Temperatursensor DHT11

Die LED-Matrix verbindest du über die SPI-Schnittstelle des Arduinos – deshalb musst du zwingend die entsprechenden Pins 10, 11 und 13 verwenden. Orientiere dich beim Aufbau an dieser Skizze:



Und hier noch einmal als Tabelle:

Matrix-Pin	Arduino-Pin
VCC	5 V
GND	GND
DIN	11
CS	10
CLK	13

Die benötigten Bibliotheken

Für die Steuerung der LED-Matrix und um die Laufschrift zu erzeugen benötigst du diese beiden Bibliotheken:

- **MD_Parola** – für Textanimationen und Effekte
- **MD_MAX72xx** – für die Steuerung der Matrix selbst

Beide kannst du bequem über den Bibliotheksverwalter in der Arduino IDE installieren. Normalerweise wirst du bereits bei der Installation der Bibliothek **MD_Parola** gefragt, ob du **MD_MAX72xx** gleich mit installieren möchtest. Bestätige das mit einfach mit **Ja.**

Der Sketch für die Laufschrift

Der folgende Sketch zeigt, wie einfach man eine Laufschrift anzeigt. Hier läuft der Text **Pollux Labs** kontinuierlich über die Anzeige.

```
// ---------------------------------------------------------
// Einfache Laufschrift mit MAX7219-LED-Matrix
// polluxlabs.net
// ---------------------------------------------------------
```

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define DATA_PIN 11
#define CLK_PIN  13
#define CS_PIN   10

MD_Parola P = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

void setup() {
  P.begin();
  P.setIntensity(5);
  P.displayClear();
    P.displayText("Pollux Labs", PA_CENTER, 50, 0, PA_SCROLL_LEFT, PA_SCROLL_LEFT);
}

void loop() {
  if (P.displayAnimate()) {
    P.displayReset();
  }
}
```

So funktioniert der Sketch

Lass uns einen kurzen Blick auf die wichtigsten Teile des Sketchs werfen:

- Die Bibliothek MD_Parola übernimmt das Anzeigen und Animieren des Textes.
- Mit dem Befehl P.setIntensity(5); stellst du die Helligkeit der LED-Matrix ein.
- Die Zeile P.displayText("Pollux Labs", PA_CENTER, 50, 0,

PA_SCROLL_LEFT, PA_SCROLL_LEFT); steuert, wie und wo der Text auf dem Display angezeigt wird. Der erste Parameter ist der Text selbst, PA_CENTER richtet ihn zentriert aus, 50 bestimmt die Scroll-Geschwindigkeit (je kleiner, desto schneller), 0 ist die Pause vor dem erneuten Start, und die beiden PA_SCROLL_LEFT definieren den Ein- und Ausblendeeffekt (hier beides nach links).

- Die Funktion displayAnimate() wird im Loop aufgerufen und steuert den Ablauf.
- Mit displayReset() startet die Animation nach dem Durchlauf erneut.

___STEADY_PAYWALL___

So änderst du die Laufrichtung des Texts

Wenn du etwas mit dem Sketch und der LED-Matrix experimentieren möchtest, probiere einmal, die Laufrichtung des Textes zu ändern. Die Befehle hierfür lauten **PA_SCROLL_RIGHT**, **PA_SCROLL_UP** und **PA_SCROLL_DOWN**.

Aktualisiere die Laufschrift über den Seriellen Monitor

Deine Laufschrift funktioniert schon einmal – allerdings ist der Text fest im Sketch integriert. Für einen neuen Text müsstest du den Sketch also anpassen und neu auf den Arduino laden. Eleganter geht das mit dem Seriellen Monitor. Der folgende Sketch startet wieder mit dem Schriftzug **Pollux Labs** – wenn du allerdings den Seriellen Monitor in der Arduino IDE öffnest und im Feld **Nachricht** einen neuen Text eingibst sowie Enter drückst, erscheint dieser auf deiner LED-Matrix. Achte hierfür darauf, dass im Seriellen Monitor die gleiche Baudrate (9600) wie im Sketch eingestellt ist.

```
// -------------------------------------------------------
```

```
// Laufschrift mit MAX7219-LED-Matrix und Texteingabe per Serial
// ---------------------------------------------------------
// Bibliotheken
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Hardware-Einstellungen
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define DATA_PIN 11
#define CLK_PIN  13
#define CS_PIN   10

// Parola-Objekt erstellen
MD_Parola P = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

// Variablen
String inputText = "Pollux Labs";  // Standardtext beim Start
bool newText = false;

void setup() {
  Serial.begin(9600);
  Serial.println("Gib einen Text ein und drücke ENTER:");
  P.begin();
  P.setIntensity(5);
  P.displayClear();
    P.displayText(inputText.c_str(), PA_CENTER, 50, 0, PA_SCROLL_LEFT, PA_SCROLL_LEFT);
}

void loop() {
  // Prüfen, ob neuer Text vom Serial Monitor vorliegt
  if (Serial.available() > 0) {
    inputText = Serial.readStringUntil('\n');  // Eingabe bis Zeilenende lesen
    inputText.trim();                          // Entfernt evtl. Leerzeichen
    if (inputText.length() > 0) {
```

```
      newText = true;
    }
  }

  // Wenn eine Animation abgeschlossen ist, neu starten oder
neuen Text setzen
  if (P.displayAnimate()) {
    if (newText) {
      P.displayClear();
        P.displayText(inputText.c_str(), PA_CENTER, 50, 0,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
      newText = false;
    }
    P.displayReset();
  }
}
```

Der Trick besteht darin, dass das Programm im loop() ständig überprüft, ob neue Daten über die serielle Schnittstelle eintreffen. Wenn der Benutzer im seriellen Monitor eine Zeile eingibt und mit der Eingabetaste bestätigt, erkennt das Programm dies über den Befehl Serial.available(). Anschließend liest es die eingegebene Zeile mit Serial.readStringUntil('\n') ein, also bis zum Zeilenendezeichen. Danach wird der Text mithilfe von displayText() auf der LED-Matrix angezeigt, sodass jede neue Eingabe sofort als Laufschrift sichtbar wird.

Ein Problem gibt es jedoch: Sobald du deinen Arduino neu startest, erscheint wieder die im Sketch hinterlegte Laufschrift **Pollux Labs**. Das behebst du, indem du deinen Text im EEPROM deines Arduinos speicherst.

Neuen Text im EEPROM speichern

Der folgende Sketch unterscheidet sich von der vorigen Version

in mehreren Punkten und sorgt dafür, dass der eingegebene Text dauerhaft gespeichert bleibt und sich weiterhin über den Seriellen Monitor ändern lässt.

Zunächst wird der eingegebene Text jetzt im **EEPROM** gespeichert – also in einem kleinen, nichtflüchtigen Speicherbereich des Arduinos. Dadurch bleibt der Text auch nach einem **Neustart oder Stromausfall** erhalten. Beim Einschalten liest der Sketch den zuletzt gespeicherten Text wieder aus dem EEPROM und zeigt ihn automatisch auf der LED-Matrix an. Ist der Speicher leer oder enthält ungültige Daten, wird automatisch der Standardtext **Pollux Labs** geladen und gespeichert.

Außerdem überprüft der Code im loop(), ob über die serielle Schnittstelle neue Eingaben vorliegen. Wenn du im Seriellen Monitor eine Zeile eingibst und mit Enter bestätigst, wird der neue Text sofort angezeigt und gleichzeitig dauerhaft im EEPROM abgelegt. Damit das funktioniert, musst du im Seriellen Monitor wie gehabt die **Baudrate auf 9600** einstellen und bei den **Zeilenende-Optionen Beides CR/LF** oder **Neue Zeile** wählen. Nur so erkennt der Sketch das Ende deiner Eingabe korrekt.

```
// ---------------------------------------------------------
// Laufschrift mit MAX7219-LED-Matrix, Texteingabe per Serial
// und Speicherung im EEPROM (robuste Version für Arduino UNO/Nano)
// ---------------------------------------------------------

#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>
#include <EEPROM.h>

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define DATA_PIN 11
#define CLK_PIN  13
#define CS_PIN   10

MD_Parola P = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN,
```

```
CS_PIN, MAX_DEVICES);

String inputText = "Pollux Labs";
bool newText = false;

// --- Text aus EEPROM lesen ---
void readEEPROMString(String &text) {
  char c;
  text = "";
  for (int i = 0; i < 100; i++) {
    c = EEPROM.read(i);
    if (c == '\0' || c == 0xFF) break;  // Ende oder leerer
Speicher
    text += c;
  }
}

// --- Text ins EEPROM schreiben ---
void writeEEPROMString(const String &text) {
  int len = text.length();
  for (int i = 0; i < len; i++) {
    EEPROM.write(i, text[i]);
  }
  EEPROM.write(len, '\0');  // Nullterminator
}

void setup() {
  Serial.begin(9600);
  delay(200);
  Serial.println("Gib einen Text ein und drücke ENTER:");
  Serial.println("(Zeilenende: 'Neue Zeile' oder 'Beides NL &
CR')");

  readEEPROMString(inputText);

  // Wenn EEPROM leer oder ungültig, Standardtext schreiben
  if (inputText.length() == 0) {
    inputText = "Pollux Labs";
    writeEEPROMString(inputText);
  }
```

```
  P.begin();
  P.setIntensity(5);
  P.displayClear();
    P.displayText(inputText.c_str(),  PA_CENTER,  50,  0,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
}

void loop() {
  // Prüfen, ob neuer Text über Serial kommt
  if (Serial.available() > 0) {
    inputText = Serial.readStringUntil('\n');
    inputText.trim();

    if (inputText.length() > 0) {
      writeEEPROMString(inputText);
      newText = true;
    }
  }

  if (P.displayAnimate()) {
    if (newText) {
      P.displayClear();
        P.displayText(inputText.c_str(), PA_CENTER, 50, 0,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
      newText = false;
    }
    P.displayReset();
  }
}
```

Die wichtigsten Funktionen in diesem Sketch

Lass uns kurz einen Blick darauf werfen, wie hier das Speichern und Lesen von Text mit dem EEPROM funktioniert.

1. Beim Start ruft der Sketch readEEPROMString() auf →

liest gespeicherten Text aus.
2. Ist der EEPROM leer, wird der Standardtext **Pollux Labs** gespeichert.
3. Wenn du im Seriellen Monitor etwas Neues eintippst und ENTER drückst,
   wird der Text mit writeEEPROMString() gespeichert.
4. Beim nächsten Einschalten liest der Arduino diesen Text wieder aus – und zeigt ihn automatisch an.

```
c = EEPROM.read(i);
```

Hier wird Zeichen für Zeichen aus dem Speicher gelesen, bis ein Ende-Zeichen ('\0') gefunden wird. So entsteht wieder der komplette Text, den du vorher gespeichert hattest.

```
EEPROM.write(i, text[i]);
```

Hier werden alle Zeichen deines Textes nacheinander im EEPROM abgelegt.

Hinweis: Der EEPROM hat eine **begrenzte Lebensdauer** (typisch etwa **100.000 Schreibvorgänge pro Speicherzelle**). Das heißt, du solltest ihn nicht ständig in jeder Loop-Schleife beschreiben – aber für Texteingaben ist das völlig unproblematisch.

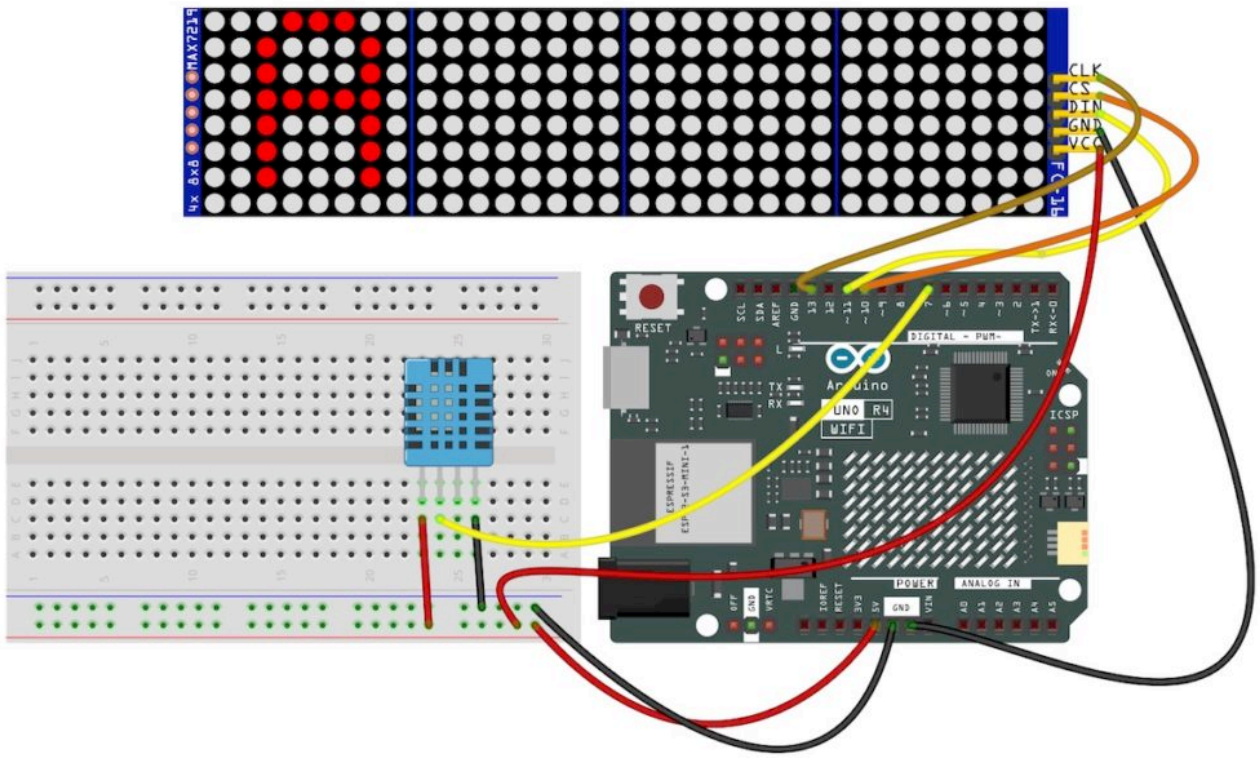
```
EEPROM.write(len, '\0');
```

Am Ende des Schreibvorgangs wird ein sogenannter **Nullterminator** gespeichert ('\0'). Das markiert das Ende des Textes. Beim Auslesen weiß der Sketch dann, wo der gespeicherte String aufhört.

Wetterdaten als Laufschrift

Lass uns zuletzt noch schnell ein praktisches Projekt bauen: Eine kleine Wetterstation. Der Sensor DHT11 ermittelt hier die aktuelle Temperatur und Luftfeuchtigkeit und dein Arduino zeigt beides als Laufschrift auf der LED-Matrix an.

Ergänze den DHT11 auf deinem Breadboard folgendermaßen:



Neben der Stromversorgung mit 5V schließt du hier den DHT11 an den Digitalpin 7 deines Arduinos an.

Der Sketch für die Wetterstation

Und hier nun der passende Sketch. Falls du bisher noch nicht mit dem DHT11 gearbeitet hast, wirf einen Blick in dieses Tutorial. Dort erfährst du, wie du die benötigten Bibliotheken installierst und wie du die aktuellen Wetterdaten ausliest.

Lade anschließend den folgenden Sketch auf deinen Arduino:

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
```

```
#include <SPI.h>
#include <DHT.h>

// --- Display Setup ---
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4  // Anzahl der Module (4 x 8x8 = 32 Spalten)
#define CS_PIN 10      // Chip Select (LOAD)

MD_Parola P = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

// --- DHT11 Setup ---
#define DHTPIN 7
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// --- Ablauf-Variablen ---
uint8_t state = 0;          // 0="Temp" Wort, 1=Temp Wert, 2="Luft" Wort, 3=Luft Wert
bool stageActive = false;  // true wenn current displayText bereits gesetzt wurde

const uint8_t speed = 25;                  // Scroll-Geschwindigkeit (kleiner = schneller)
const unsigned long pauseWord = 2000;   // 2s Pause für das Wort
const unsigned long pauseValue = 5000;  // 5s Pause für den Wert

void setup() {
  Serial.begin(9600);

  // Display init
  P.begin();
  P.setIntensity(3);  // Helligkeit 0-15
  P.displayClear();

  // DHT init
  dht.begin();
  delay(1000);  // kurzer Warmup
```

```
  // Start mit "Temp"
   stageActive = false;  // initial wird im loop der erste
Schritt gesetzt
}

void loop() {
  // Wenn noch kein Text für den aktuellen Step gesetzt wurde
-> setze ihn
  if (!stageActive) {
    char buf[20];

    switch (state) {
      case 0:  // Wort "Temp" linksbündig, scroll in/out, 1s
Pause
        strcpy(buf, "Temp");
          P.displayText(buf, PA_LEFT, speed, pauseWord,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
        stageActive = true;
        break;

      case 1:
        {  // Temperaturwert rechtsbündig, scroll in/out, 2s
Pause
          float t = dht.readTemperature();
          if (isnan(t)) {
            strcpy(buf, "Err");
          } else {
            dtostrf(t, 4, 1, buf);  // Format: z.B. "23.5"
            strcat(buf, "C");
          }
            P.displayText(buf, PA_RIGHT, speed, pauseValue,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
          stageActive = true;
          break;
        }

      case 2:  // Wort "Luft" linksbündig, scroll in/out, 1s
Pause
        strcpy(buf, "Luft");
          P.displayText(buf, PA_LEFT, speed, pauseWord,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
```

```
        stageActive = true;
        break;

      case 3:
         {  // Luftfeuchte rechtsbündig, scroll in/out, 2s
Pause
          float h = dht.readHumidity();
          if (isnan(h)) {
            strcpy(buf, "Err");
          } else {
            dtostrf(h, 3, 0, buf);  // Format: z.B. "45"
            strcat(buf, "%");
          }
            P.displayText(buf, PA_RIGHT, speed, pauseValue,
PA_SCROLL_LEFT, PA_SCROLL_LEFT);
          stageActive = true;
          break;
        }
    }
  }

  // Animation durchlaufen lassen; wenn fertig, auf nächsten
Schritt wechseln
  if (P.displayAnimate()) {
    stageActive = false;
    state = (state + 1) % 4;  // Zyklus 0..3
  }
}
```

Anschließend sollten die aktuelle Temperatur und Luftfeuchtigkeit im Wechsel durch das Display laufen. Die Geschwindigkeit bzw. die Zeit, wie lange der Text (**Temp** und **Luft**) und der Wert stehen bleiben, stellst du hier im Sketch ein:

```
const unsigned long pauseWord = 2000;   // 2s Pause für das
Wort
const unsigned long pauseValue = 5000;  // 5s Pause für den
Wert
```

Und das soll es erst einmal gewesen sein. Viel Erfolg und Spaß mit deiner LED-Matrix!