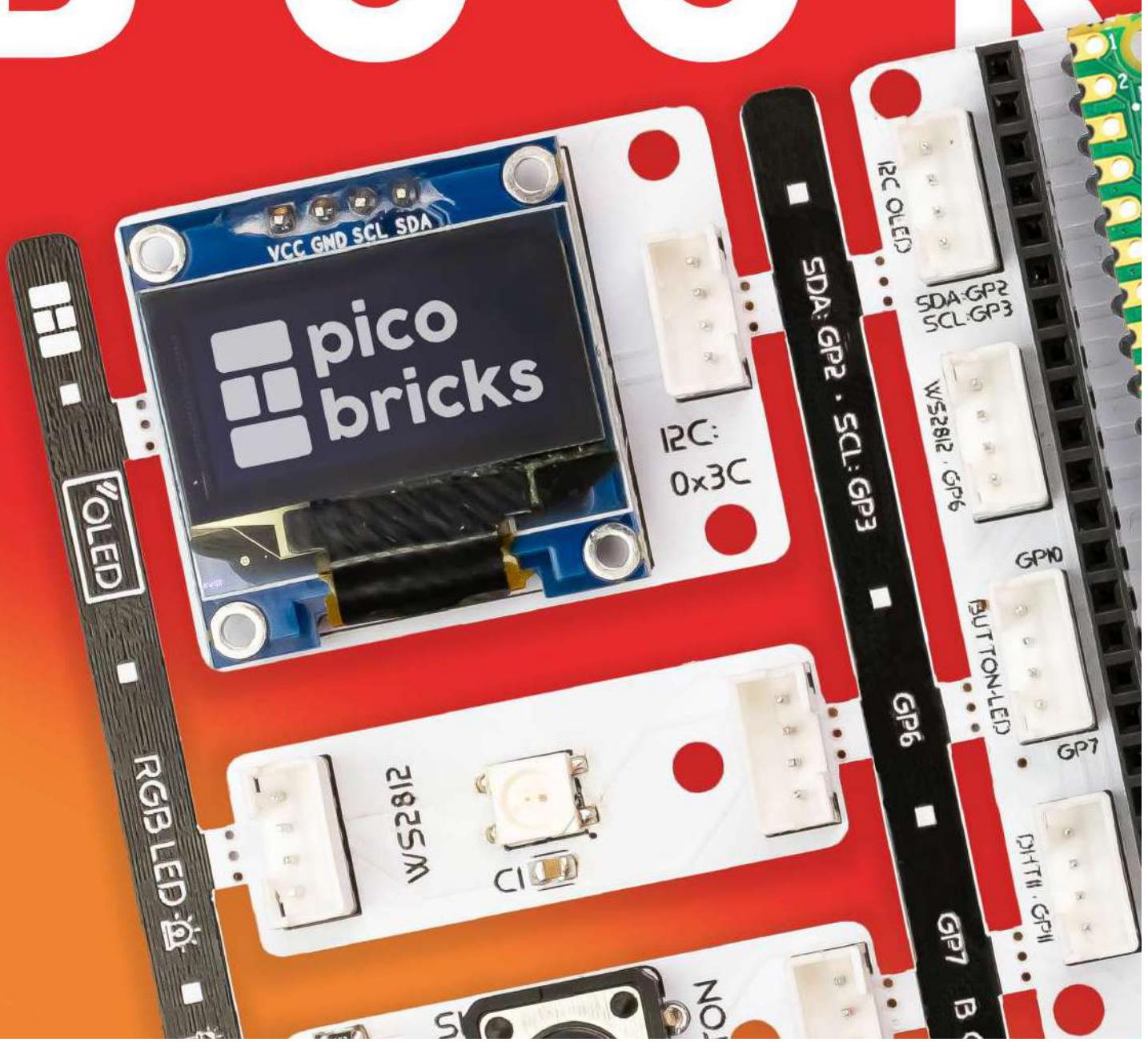


# Project Book





Copyright © 2022 Robotistan

Alle Rechte vorbehalten. Es ist strengstens untersagt, den Text, Fotografien und andere Inhalte dieses Buches, ganz oder teilweise, ohne Genehmigung zu kopieren, zu reproduzieren, zu verwenden, zu veröffentlichen und zu verbreiten, außer für den persönlichen Gebrauch.

Inhalt: Mustafa Kemal Avcı, Abdullah Kaya

Übersetzung: Naze Gizem Özer

**Design:** Ahmet Gürsu

Pico Bricks Entwicklerteam

Yasir Çiçek - Projektmanager

Yusuf Gündoğdu - Softwareentwickler

Mehmet Suat Morkan - Hauptentwickler

Mehmet Ali Dağ - Hardwareentwickler



Powered by



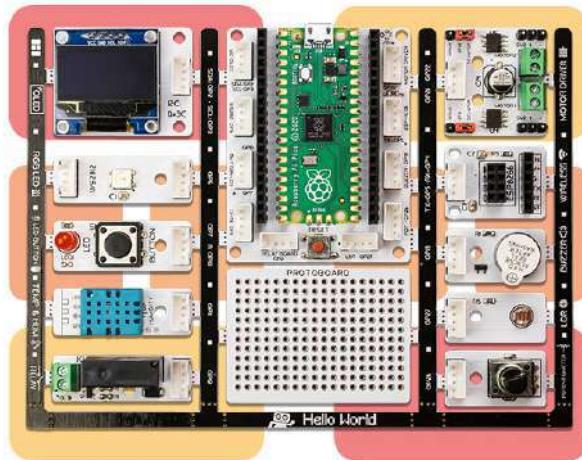
John Maloney · Turgut Güneysu · Kathy Giori · Bernat Romagosa

# INHALT

Was sind Pico Bricks?	4
<b>1. Entwicklungsumgebungen</b>	<b>6</b>
1.1. MicroBlocks Block-Programmiersprache	6
1.1.1. Schnittstellen Einführung	7
1.1.2. MicroBlocks-Picobricks Verbindung und Betrieb	9
1.2. Thonny (MicroPython) IDE für Anfänger	13
1.2.1 Thonny IDE Einrichtung	13
1.2.2. Thonny IDE-Schnittstelle	13
1.2.3. MicroPython-Firmware auf Raspberry Pi Pico hochladen	14
1.2.4. Installation und Ausführung von Code auf Raspberry Pi Pico	15
1.3. Arduino IDE	17
1.3.1. Schreiben und Ausführen von Code mit Arduino IDE	18
1.3.2. Wie fügt man eine Arduino-Bibliothek hinzu?	20
<b>2. PROJEKTE</b>	<b>22</b>
2.1. Blinken	23
2.2. Aktion - Reaktion	27
2.3. Autonome Beleuchtung	31
2.4. Thermometer	38
2.5. Grafikmonitor	44
2.6. Den Rhythmus dominieren	49
2.7. Zeige deine Reaktion	60
2.8. Mein Timer	68
2.9. Wecker	78
2.10. Kenne deine Farbe	85
2.11. Zauberlampe	97
2.12. Smarter Kühler	101
2.13. Buzz Wire Spiel	106
2.14. Dinosaurier-Spiel	115
2.15. Nacht und Tag	120
2.16. Sprachgesteuertes Roboterauto	132
2.17. Zwei-Achsen-Roboterarm	140
2.18. Smartes Haus	154
2.19. Schlemmer-Spardose	161
2.20. Bestätigungstür	168
2.21. Automatischer Mülleimer	182
2.22. Digitaler Lineal	188
2.23. Luftklavier	196
2.24. Labyrinth-Lösungsroboter	205
2.25. Intelligentes Gewächshaus	213
3. Bibliographie	233

## Was sind Pico Bricks?

Pico Bricks ist ein elektronisches Entwicklungsboard + Software, das für die Verwendung in Maker-Projekten entwickelt wurde. Mit zehn abnehmbaren Modulen kann Pico Bricks verwendet werden, um eine Vielzahl von Projekten zu erstellen. Es enthält auch ein Prototypenbrett, das Sie verwenden können, um eigene Module hinzuzufügen!



Pico Bricks richtet sich an alle, die sich für Elektronik und Programmierung interessieren. Anfänger ohne vorherige Erfahrung werden dank des modularen Hardware-Designs, der Scratch-ähnlichen Blockprogrammierumgebung und des Simulators leicht starten können. Erfahrene Benutzer können tiefer in die Elektronik eintauchen oder das Programmieren in Python erkunden. Und selbst die erfahrensten Maker werden zu schätzen wissen, wie schnell sie Ideen erkunden und Prototypen mit

Pico Bricks erstellen können.

Im Gegensatz zu anderen Boards bietet Pico Bricks eine incredible Menge an Flexibilität für jede Maker-Stufe! Die Bricks IDE hat Beispielcodes für verschiedene Szenarien.

Erlernen Sie das Programmieren von Grund auf mit MicroBlocks oder dem Drag-and-Drop-Block-Programmier-Builder von Pico Bricks. MicroBlocks ist die einfachste Programmiererfahrung, die je geschaffen wurde und in der Maker-Industrie weit bekannt ist.



KIDS



STUDENTS



HOBBYISTS



EXPERT

Haben Sie eine Frage? Weitere Informationen finden Sie [here](#)



POWERED BY  
**MicroBlocks**

# **ENTWICKLUNG UMGEBUNG**

# 1. Entwicklungsumgebungen

In der Software-, Web- und mobilen Anwendungsentwicklung ist die Entwicklungs umgebung ein Arbeitsbereich mit einer Reihe von Prozessen und Programmierwerkzeugen, die verwendet werden, um den Quellcode für eine Anwendung oder Softwareprodukt zu entwickeln. Entwicklungsumgebungen ermöglichen es Entwicklern, zu erstellen und zu innovieren, ohne etwas in einer Live-Umgebung zu beschädigen. Sie können Picobricks sowohl mit textbasierten als auch mit blockbasierten Editoren programmieren. MicroBlocks ist ein leistungsstarker Editor, mit dem Sie Picobricks mit Blöcken programmieren können.

Der Thonny-Editor ist eine kostenlose, dedizierte IDE (integrierte Entwicklungsumgebung) für Python, die für Anfänger entwickelt wurde und eine der besten Wahl für diejenigen ist, die gerade anfangen, MicroPython zu lernen. Egal, ob Sie nur MicroPython programmieren oder eine elektronische Schaltung entwerfen, Thonny bietet Ihnen viel Unterstützung.

Die Arduino IDE ist eine Open-Source-Software, die von Arduino.cc entwickelt wurde und hauptsächlich zum Schreiben, Kompilieren und Hochladen von Code auf fast alle Arduino-Module verwendet wird. Es ist eine offizielle Arduino-Software, die die Codekompilierung so einfach macht, dass sogar eine gewöhnliche Person ohne technische Vorkenntnisse mit dem Lernprozess beginnen kann. Wenn Sie in der Arduino C-Sprache versiert sind oder die C-Sprache lernen möchten, sind Picobricks und die Arduino IDE sehr gute Wahl für Sie.

## 1.1. MicroBlocks Block-Programmiersprache

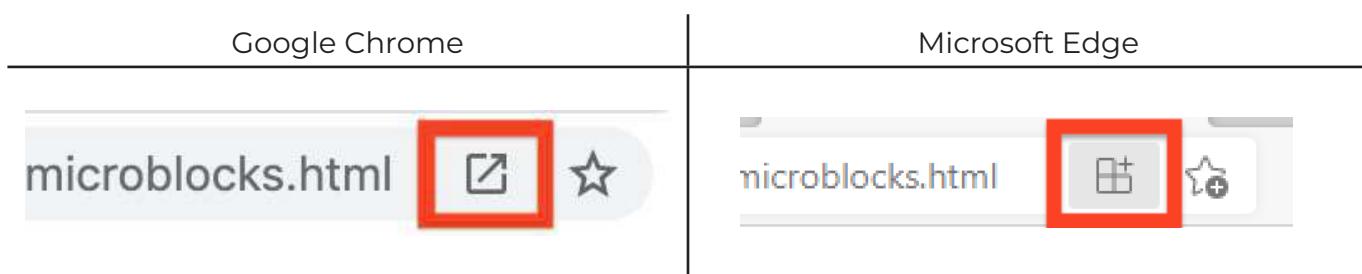
MicroBlocks ist eine kostenlose, Scratch-ähnliche Block-Programmiersprache zum Lernen von physikalischem Computing mit Bildungs-Mikrocontroller-Boards wie dem micro:bit, Adafruit Circuit Playground Express und vielen anderen. MicroBlocks ist eine Live-Umgebung. Klicken Sie auf einen Block und er wird sofort auf dem Board ausgeführt. Probieren Sie Befehle aus. Sehen und grafisch darstellen von Sensorwerten in Echtzeit. Kein Warten mehr auf das Kompilieren und Herunterladen von Code. Möchten Sie eine Animation anzeigen, während Sie einen Motor steuern? Kein Problem! MicroBlocks ermöglicht es Ihnen, separate Skripte für jede Aufgabe zu schreiben und sie gleichzeitig auszuführen. Ihr Code ist einfacher zu schreiben und leichter zu verstehen. MicroBlocks läuft auf vielen verschiedenen Boards, aber Ihre Skripte sind portabel. Tasten, Sensoren und Anzeige-Blöcke verhalten sich auf allen Boards mit der entsprechenden Hardware gleich. Sobald Sie die Codes in MicroBlocks ausführen, können Sie das USB-Kabel trennen und die Picobricks mit einer anderen Stromquelle versorgen. Die Codes auf der Karte funktionieren automatisch. Sie können Thonny IDE und Arduino IDE verwenden, um Picobricks textbasiert zu programmieren.

Um Picobricks mit MicroBlocks zu programmieren, öffnen Sie <https://microblocks.fun/> im Browser (Google Chrome und Edge-Browser werden empfohlen).



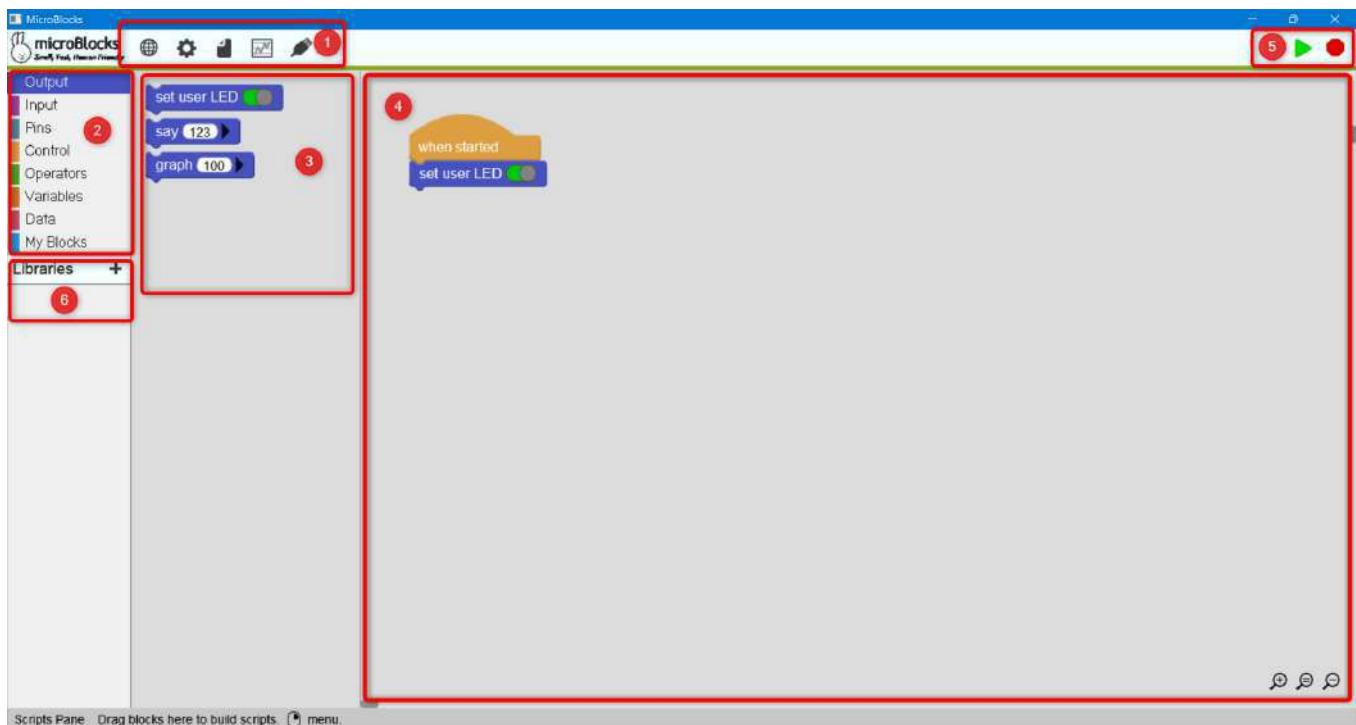
Sie müssen nichts installieren, um MicroBlocks in einem Chrome- oder Edge-Browser auszuführen; Sie können den Online-Editor ausführen, indem Sie auf die Schaltfläche Ausführen im Menü oben rechts auf dem Bildschirm klicken. Durch Klicken auf die Schaltfläche **Download** können Sie die für Ihr Betriebssystem geeignete Version herunterladen und auf Ihrem Computer installieren.

Sie können den MicroBlocks Web-Editor in Ihrem Browser speichern und ohne Internetzugang verwenden. Führen Sie MicroBlocks in Ihrem Browser aus, um die MicroBlocks Web-App zu registrieren, und klicken Sie dann auf die **Installieren**-Schaltfläche in der oberen rechten Ecke der URL-Leiste Ihres Browsers.



### 1.1.1. Schnittstellen Einführung

Wenn Sie das MicroBlocks-Programm öffnen, wird Sie eine Benutzeroberfläche wie im Bild begrüßen. Sie können die detaillierte Erklärung der Programmoberfläche unten einsehen.



1. Menüleiste ( ): In diesem Abschnitt befindet sich der erste Button von links nach rechts ermöglicht es uns, die Spracheinstellung des Programms zu ändern. Der zweite Button ist das Menü, in dem wir die erweiterten MicroBlocks-Code-Optionen sehen können, und das Firmware-Update wird durchgeführt, während der dritte Button die Speicheroptionen anbietet. Der vierte Button öffnet ein Grafikfenster, das vom Grafikblock verwendet wird, um die Daten darzustellen, während der fünfte rechte Button verwendet wird, um eine Verbindung zu Picobricks herzustellen.
2. Blockkategorien: Dieses Feld enthält die Kategorien von Blöcken, die für Programmierung in MicroBlocks verwendet werden. Die Kategorien sind mit verschiedenen Farben für jede Kategorie gruppiert. Wenn die Kategorien ausgewählt werden, werden die relevanten Blöcke in dieser Kategorie im Block 3-Feld in der Palette aufgelistet.
3. Blockpalette: Wenn Auswahl im Feld der Blockkategorien getroffen wird, werden Blöcke mit speziellen Funktionen in diesem Feld aufgelistet. Codes werden erstellt, indem die Blöcke in diesem Bereich in den Skriptbereich Nummer 4 gezogen und abgelegt werden.
4. Skriptbereich: Dies ist der Bereich, in dem alle Programmieraktivitäten stattfinden. Benutzer ziehen und legen Blöcke in diesen Bereich, um Skripte und benutzerdefinierte Blöcke (Funktionen) zu erstellen.
5. Start/Stopp-Tasten ( ) Dieser Bereich enthält zwei Symbole, Start und Stopp, die verwendet werden, um die MicroBlocks-Programme zu steuern.
6. Bibliotheksliste ( Libraries ) In diesem Bereich gibt es Bibliotheken, die geladen werden abhängig von den Anforderungen der Benutzerskripte und Mikrovorrichtungen.

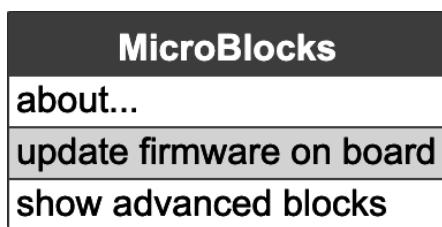
## 1.1.2. MicroBlocks-Picobricks Verbindung und Betrieb

Verbinden Sie die Platine mit Ihrem Computer, während Sie die weiße BOOTSEL

Taste gedrückt halten.



Wählen Sie im MicroBlocks-Menü (Zahnrad-Symbol) die Option Firmware auf dem Board aktualisieren.

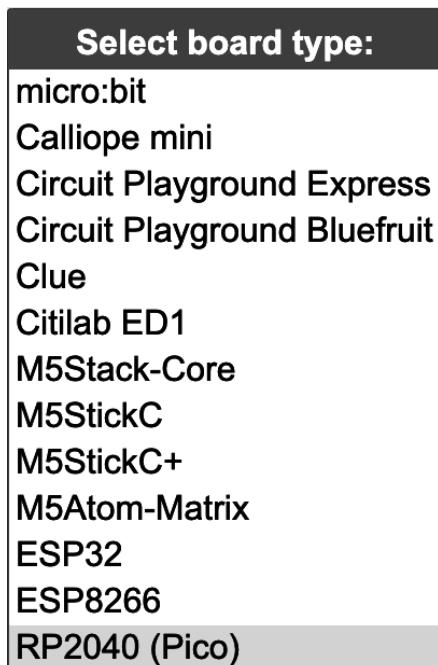


Die Firmware-Installation dauert nur wenige Sekunden. Wenn Sie die MicroBlocks App ausführen, wird MicroBlocks automatisch eine Verbindung zum Board herstellen, wenn dies abgeschlossen ist.

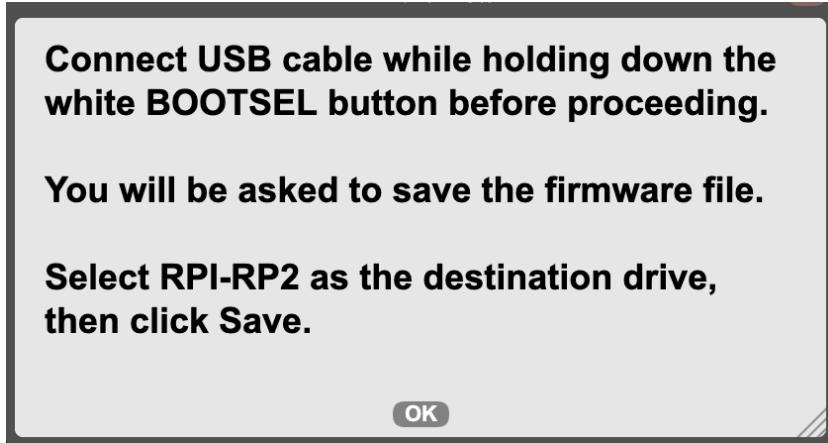
Zusätzliche Schritte im Browser

Wenn Sie MicroBlocks im Browser oder als Web-App ausführen, müssen Sie dem Browser helfen. Aus Sicherheitsgründen kann der Browser nicht auf das USB-Laufwerk des Boards zugreifen, ohne den Benutzer zu fragen.

Wählen Sie zunächst Ihren Boardtyp aus dem Menü aus.



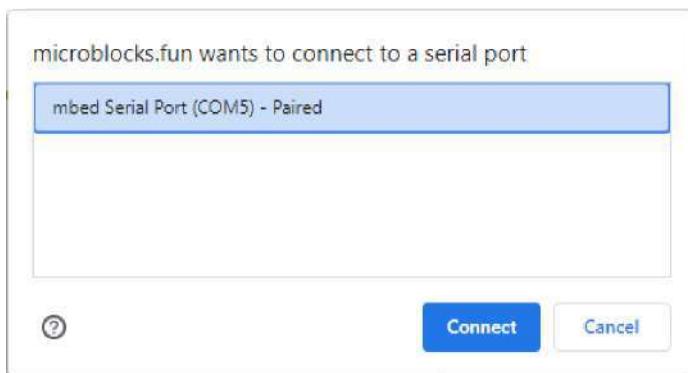
Sie werden aufgefordert, das USB-Laufwerk für das Board im Datei-Speichern-Dialog des Browsers auszuwählen.



Befolgen Sie die Anweisungen, um die Firmware-Datei auf Ihrem Board zu speichern. Wenn die Datei gespeichert ist (gerade ein paar Sekunden), klicken Sie auf das USB-Symbol, um eine Verbindung herzustellen.

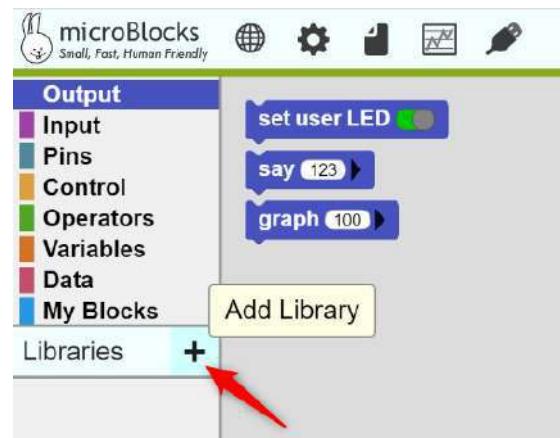


🔌 ➡️ Ein Klick auf die Verbinden-Schaltfläche zeigt die USB-Ports des Systems an, an denen die Mikrogeräte angeschlossen sind. In diesem Fenster können Sie Picobricks mit MicroBlocks verbinden, indem Sie zuerst das Pico-Gerät auswählen und dann auf die Schaltflächen „Verbinden“ klicken. Wenn die Verbindung erfolgreich ist, erscheint ein grüner Kreis hinter dem USB-Symbol.

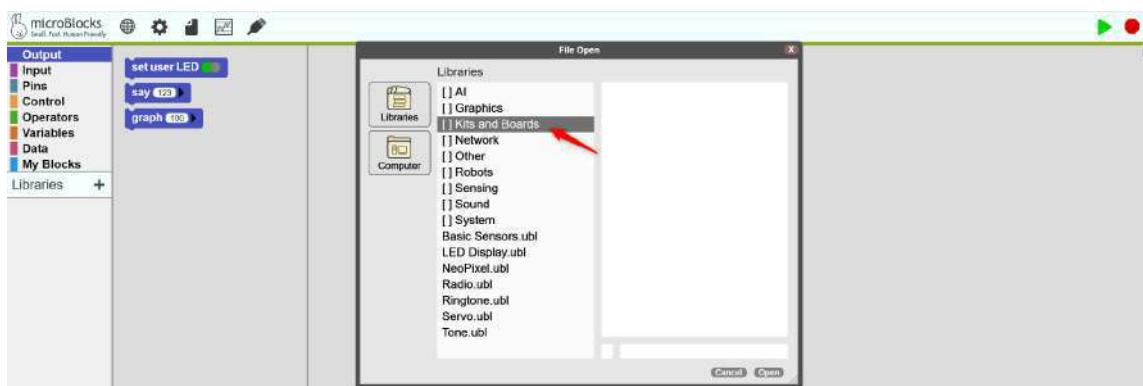


## MicroBlocks ist ein Echtzeit-Coding-Editor.

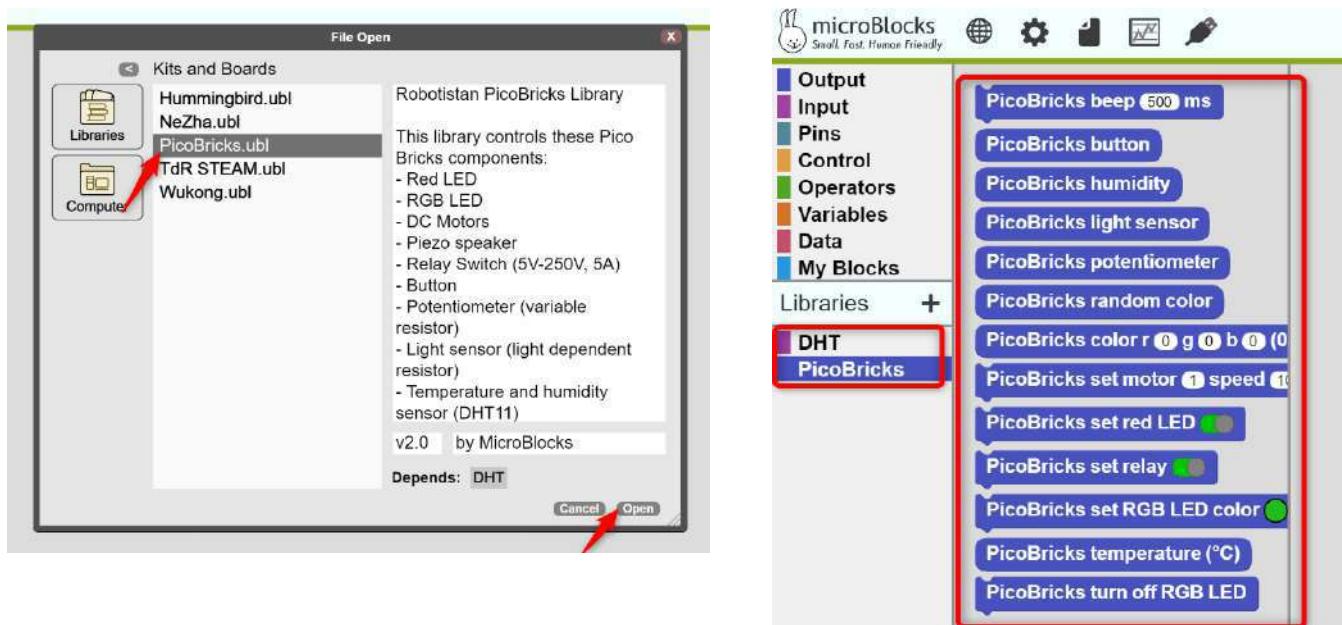
Es gibt keinen Prozess zum Kompilieren und Hochladen der Codes auf die Karte, nachdem sie geschrieben wurden. Wenn Sie auf die Codeblöcke klicken, werden die Codes ausgeführt. Zuerst müssen Sie die Bibliothek von Picobricks in den Microblocks-Editor importieren. Dazu müssen Sie auf die Bibliothek hinzufügen-Schaltfläche klicken.



Im Fenster Datei öffnen, das sich öffnet, klicken Sie auf die Schaltfläche Kits und Boards, um die Liste von Geräten zu öffnen, die Sie mit Microblocks programmieren können.



Klicken Sie auf **PicoBricks.ubl** aus der Dropdown-Liste und klicken Sie dann auf die Öffnen Schaltfläche.



Wenn alles gut gelaufen ist, werden die PicoBricks-Bibliothek und die Codeblöcke im Codeblocks-Bereich angezeigt.

Jetzt lassen Sie uns unseren ersten Code ausführen. Zuerst ziehen Sie den Block „wenn gestartet“ aus dem Steuerungsmenü in den Code-Schreibbereich. Ziehen Sie dann den Block „PicoBricks Set rote LED“ aus der Kategorie PicoBricks und fügen Sie ihn unter dem Block „wenn gestartet“ hinzu. Wenn Sie die Starttaste drücken, sehen Sie, wie die rote LED auf den PicoBricks leuchtet.

 Nachdem Sie Ihre Codes in MicroBlocks bearbeitet haben, werden Ihre Codes installiert und ausgeführt, wenn Sie auf die Starttaste klicken.

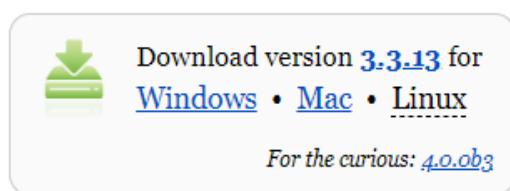
 Die Stopptaste stoppt die Ausführung der Codes. Aber die auf die PicoBricks hochgeladenen Codes werden nicht gelöscht. Sie können das USB-Kabel trennen und die PicoBricks mit einer externen Stromversorgung betreiben.

Wenn Sie zuvor die erforderliche Firmware-Datei hochgeladen haben, um die PicoBricks mit MicroBlocks zu kodieren, können Sie sich verbinden, indem Sie auf das USB-Symbol klicken. Wenn Sie MicroBlocks PicoBricks zum ersten Mal verbinden möchten, können Sie die Schritte im Abschnitt 1.2 befolgen.

Für detaillierte Informationen zur Verwendung des Microblocks-Editors besuchen Sie:

<https://wiki.microblocks.fun/ide>

## 1.2. Thonny (MicroPython) IDE für Anfänger

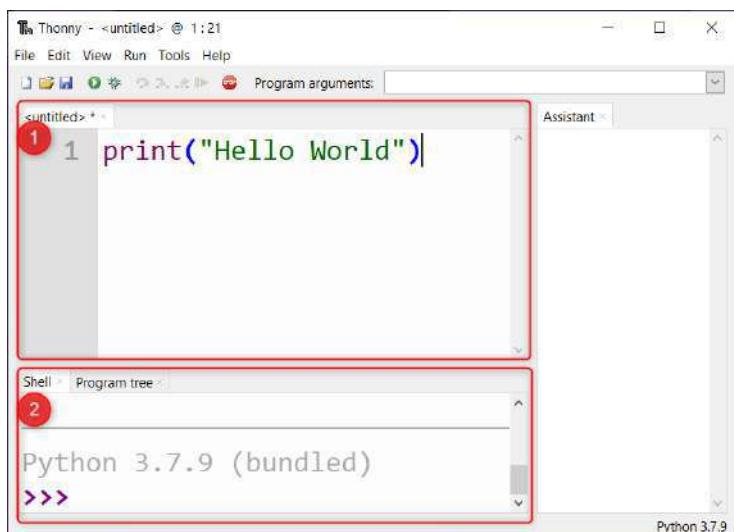


Im Herzen der PicoBricks steht der Raspberry Pi Pico. Der Thonny Raspberry Pi ist eine großartige Wahl für das Programmieren von Pico und damit von Picobricks.

### 1.2.1 Thonny IDE Einrichtung

Besuchen Sie <https://thonny.org/>. Wählen Sie die für Ihr System geeignete Version aus und laden Sie sie auf Ihren Computer herunter. Führen Sie dann die Installation durch. Sie können die Thonny IDE auch mit dem Befehl "\$ pip install thonny" installieren.

### 1.2.2. Thonny IDE Benutzeroberfläche



Wenn Sie Thonny starten, sehen Sie ein Fenster wie das untenstehende. Wir werden unsere Codes im Teil 1 schreiben. Im Teil 2 werden wir die Ausgaben unserer Codes sehen.



A: Öffnet eine leere Skriptdatei.

B: Ermöglicht Ihnen das Öffnen einer vorhandenen Code-Datei.

C: Ermöglicht Ihnen, Änderungen an der Code-Datei, an der Sie arbeiten, zu speichern.

D: Führt den Code aus, den Sie in der von Ihnen angegebenen Interpreter-Umgebung geschrieben haben.

E: Ermöglicht Ihnen, nach Fehlern in Ihrem Code zu suchen.

F: Ermöglicht es Ihnen, Codezeilen auszuführen, um Fehler zu beheben.

**G:** Ermöglicht es Ihnen, während des Debuggens durch die Befehle in der Codezeile zu navigieren.

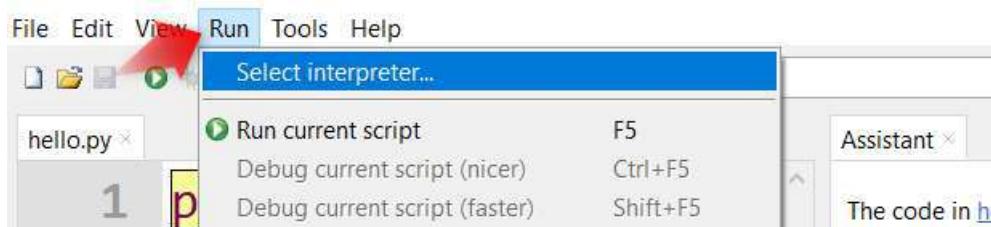
H: Ermöglicht Ihnen, das Debugging zu beenden.

I: Ermöglicht Ihnen, vom Debug-Modus in den Ausführungsmodus zu wechseln.

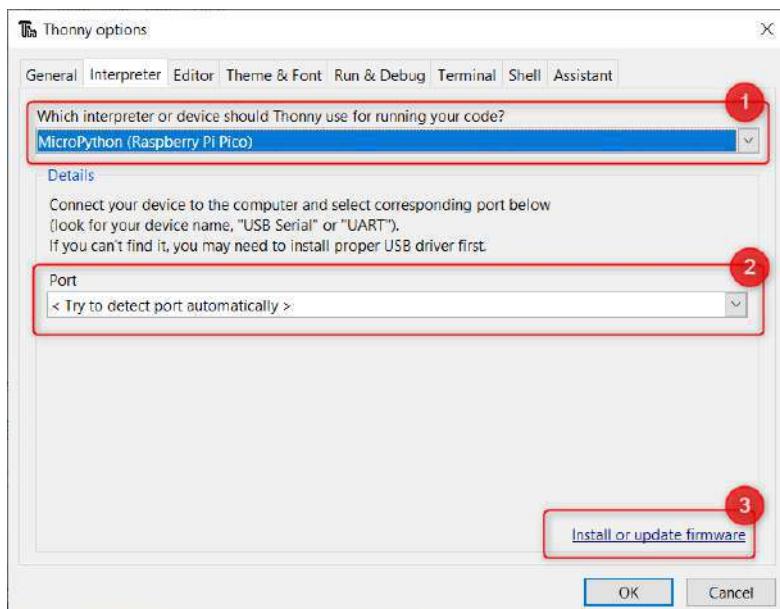
**J:** Stoppt die Ausführung des Codes.

### 1.2.3. MicroPython-Firmware auf Raspberry Pi Pico hochladen

Damit der Raspberry Pi Pico die MicroPython-Codes, die wir schreiben werden, versteht, müssen wir ein spezielles Betriebssystem für ihn installieren. Wir nennen dies Firmware. Öffnen Sie den Thonny Editor und klicken Sie im Menü Ausführen auf Interpreter auswählen.



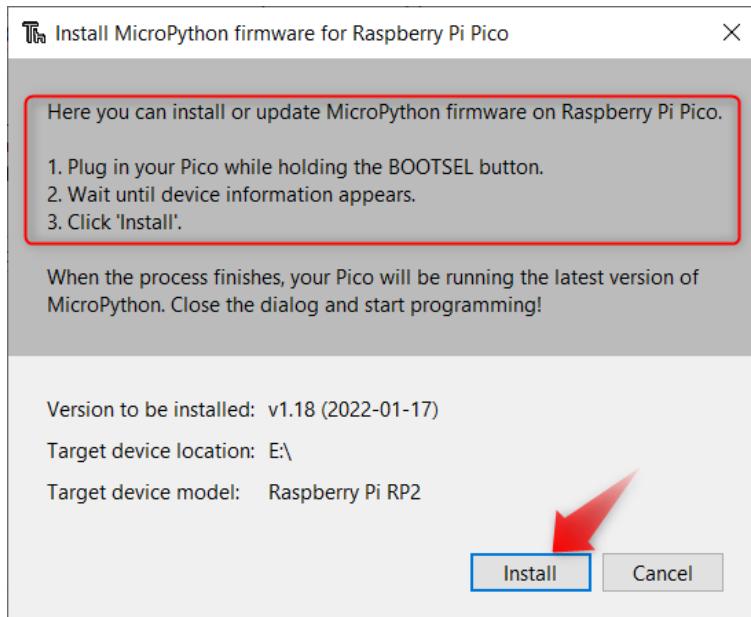
Wählen Sie den **Raspberry Pi Pico** aus der Dropdown-Liste im Bereich 1 aus. Lassen Sie den 2. Bereich wie im Bild und klicken Sie auf den 3. Bereich.



Verbinden Sie den Pico mit dem USB-Port Ihres Computers mit einem Kabel, während Sie die weiße bootsel-Taste darauf gedrückt halten.



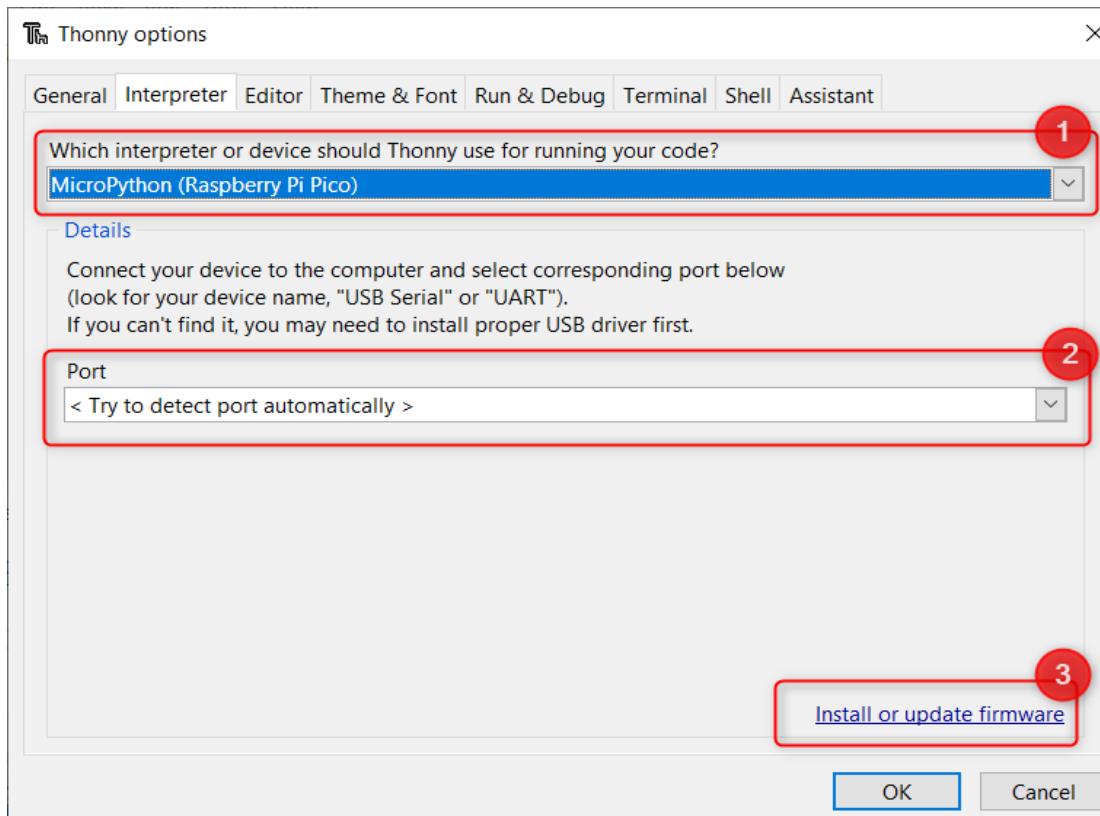
Nachdem die Installationsschaltfläche aktiviert ist, können Sie die Taste loslassen. Drücken Sie die Installations-Schaltfläche und warten Sie, bis die Firmware geladen ist.

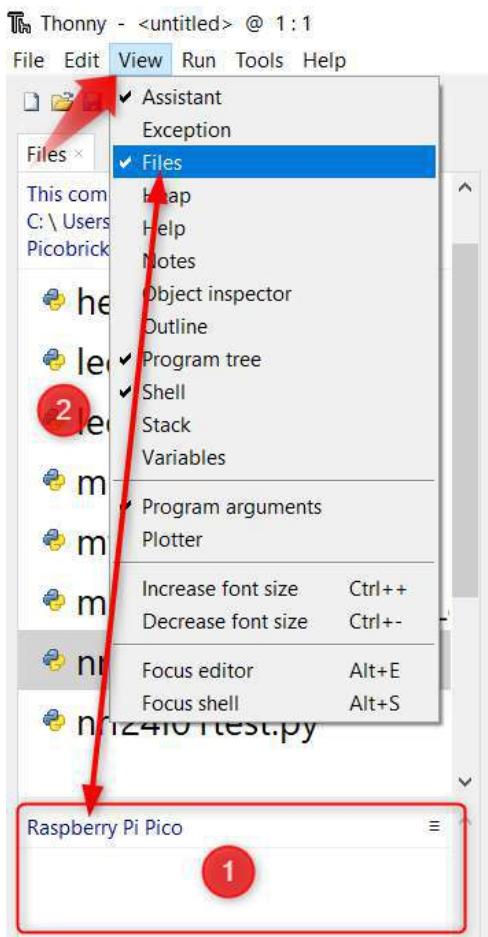


Nachdem die Installation abgeschlossen ist, klicken Sie auf die Schaltfläche Schließen, um die Installation abzuschließen.

#### 1.2.4. Installation und Ausführung von Code auf Raspberry Pi Pico

Stecken Sie das Kabel des Pico direkt in den USB-Port des Computers. Sie müssen nicht die Bootsel-Taste gedrückt halten. Wählen Sie die Interpreter auswählen Option im Ausführen Menü in Thonny. Stellen Sie sicher, dass Raspberry Pi Pico in Abschnitt 1 ausgewählt ist. Klicken Sie auf die **OK** Schaltfläche, um das Fenster zu schließen.

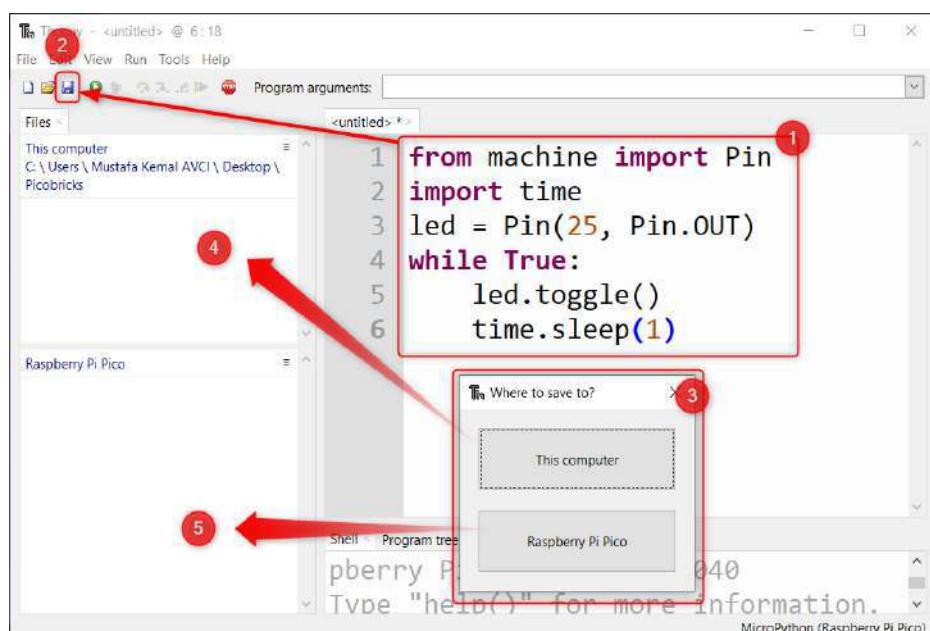




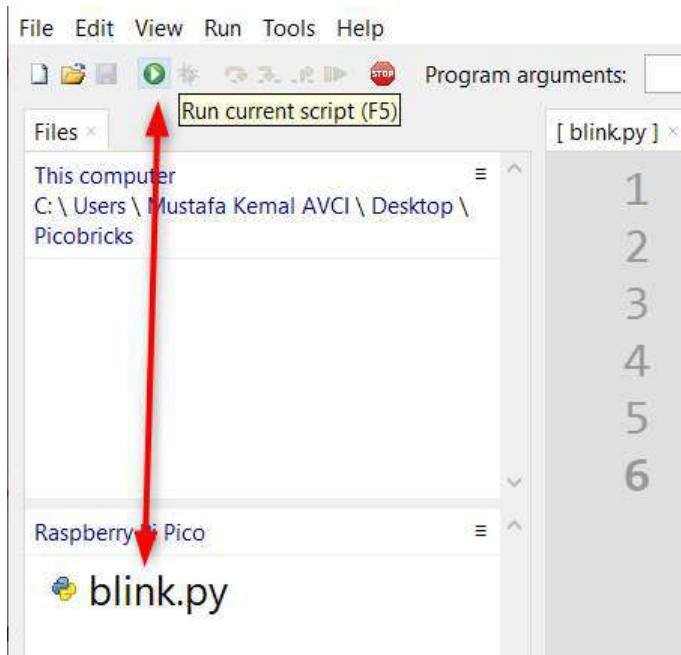
Aktivieren Sie die Dateien Option im Ansicht Menü. Ein langer Datei-Explorer-Tab wird auf der linken Seite des Bildschirms platziert. Wenn Sie Raspberry Pi Pico in Abschnitt 1 sehen, bedeutet das, dass es ohne Probleme mit Thonny Pico verbunden ist, und Sie bereit sind, Ihren Code zu schreiben, zu speichern und auszuführen. Die Teilenummer 2 hinter dem Menü ist der Bereich des Datei-Explorers, der das Arbeitsverzeichnis auf Ihrem Computer anzeigt.

Die MicroPython-Codes, die Sie in Thonny geschrieben haben, bestehen aus Bibliotheken, die für Raspberry Pi Pico und ähnliche Mikrocontrollerkarten angeordnet sind und werden MicroPython genannt. Die Syntax und fast alle Bibliotheken arbeiten gleich wie MicroPython.

Die „Hello World“ Anwendung der Softwarewelt ist die „Blink“ Anwendung in der physischen Programmierung. Schreiben Sie den im Feld 1 angezeigten Code auf. Klicken Sie auf die Speichern Schaltfläche im Bereich 2. Thonny wird Sie im Fenster in Bereich 3 fragen, ob Sie Ihren Code im Arbeitsverzeichnis auf Ihrem Computer oder im Onboard-Speicher von Pico speichern möchten. Wenn Sie Ihren Computer wählen, wird die resultierende Datei im Feld 4 angezeigt, und wenn Sie Pico wählen, wird die resultierende Datei im Feld 5 angezeigt.



Wählen Sie Raspberry Pi Pico im Speichern-Fenster aus, geben Sie „blink.py“ im Dateiname Feld ein und klicken Sie auf die OK-Schaltfläche.



Nachdem Sie die „blink.py“ Datei im Datei-Explorer von Pico gesehen haben, drücken Sie die F5-Taste auf der Tastatur oder die grüne Ausführen-Schaltfläche in der Symbolleiste, und die Code-Datei wird von Pico ausgeführt. Wenn Sie die interne LED auf dem Pico im Intervall von 1 Sekunde blinken sehen, haben Sie erfolgreich Ihren ersten Code geschrieben und ausgeführt. Herzlichen Glückwunsch :)

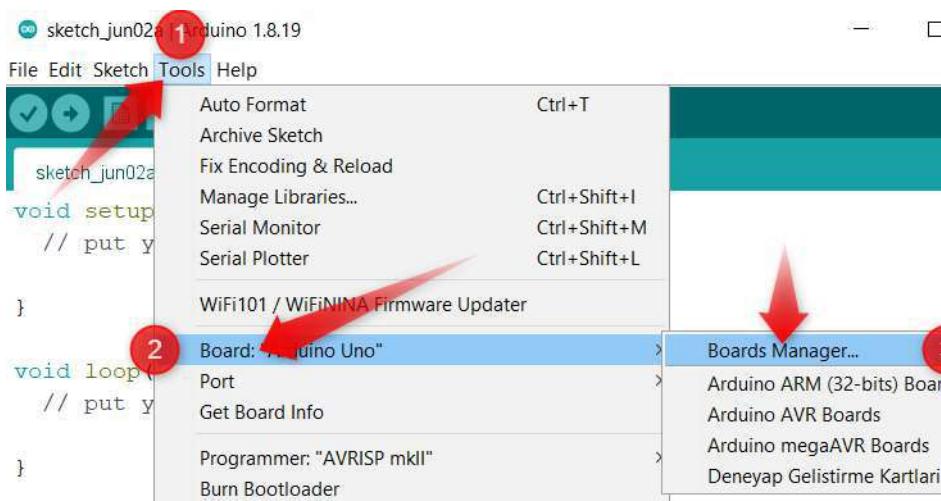
Eine wichtige Anmerkung: Wenn Sie möchten, dass der Code, den Sie geschrieben haben, sofort ausgeführt wird, wenn Pico geöffnet wird, ohne einen Ausführungsbefehl zu geben, müssen Sie Ihren Code im Hauptverzeichnis von Pico unter dem Namen „main.py“ speichern.

## 1.3. Arduino IDE

Picobricks bietet uns die Möglichkeit, mit Arduino C zu programmieren. Der Einstieg in die Programmierung des Raspberry Pi Pico, der im Herzen von Picobricks steckt, mit der weit verbreiteten Arduino IDE ist ziemlich einfach.

Laden Sie die Installationsdatei der Arduino IDE 1.8.x von <https://www.arduino.cc/en/software> auf Ihren Computer herunter und installieren Sie sie.

Zuerst müssen Sie den Raspberry Pi Pico zur Arduino IDE hinzufügen. Starten Sie die Arduino IDE. Gehen Sie dann zu Werkzeuge>Board>Boardverwalter.



Schreiben Sie „Raspberry Pi Pico“ in Feld 1. Warten Sie eine Weile und klicken Sie auf die **Arduino Mbed OS RP2040 Boards**-Option und klicken Sie auf die **Installieren**-Schaltfläche in Feld 2.



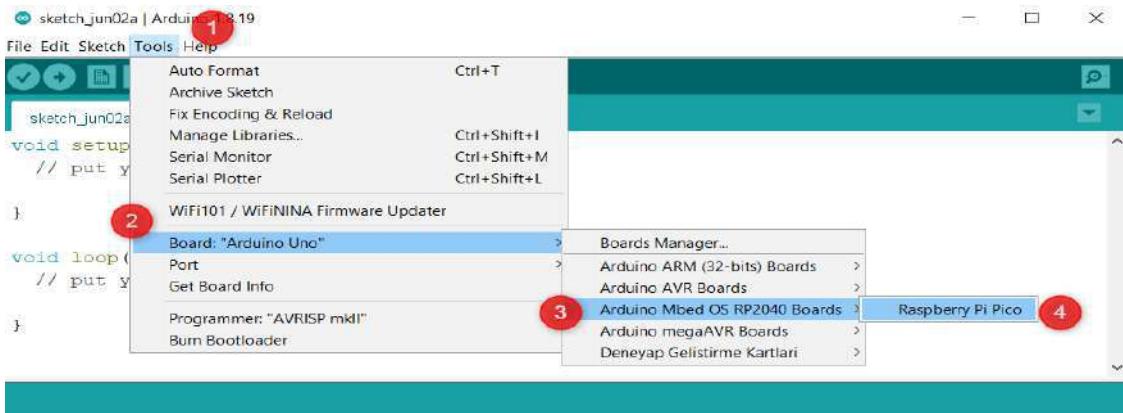
Während all dieser Installationen müssen Sie die Genehmigungen akzeptieren, die Sie gefragt werden. Wenn die Installation abgeschlossen ist und Sie auf die Schaltfläche Schließen klicken, haben Sie Pico zur Arduino IDE hinzugefügt.

### 1.3.1. Schreiben und Ausführen von Code mit der Arduino IDE

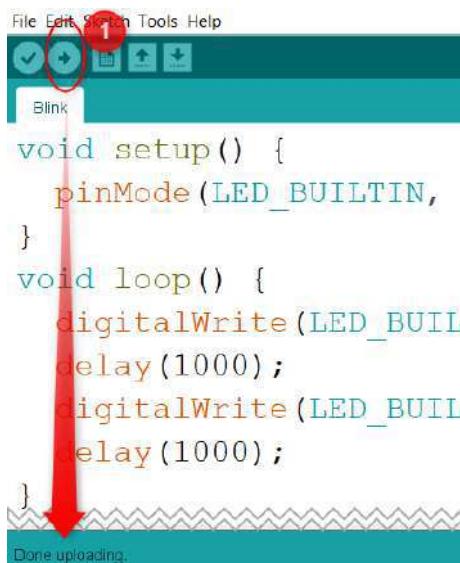
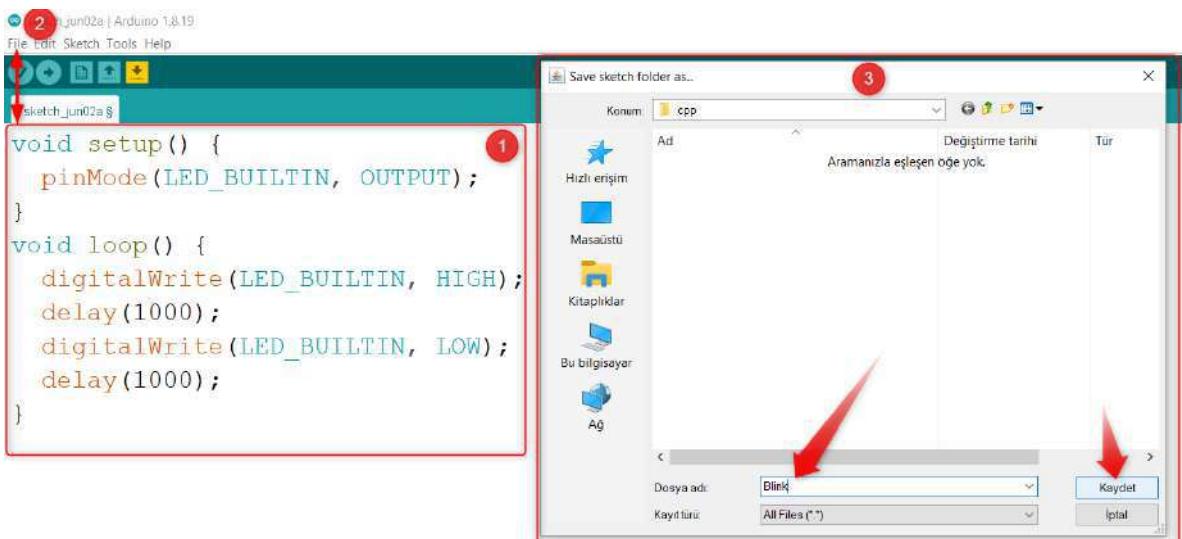
Wenn Sie Pico mit der Arduino IDE programmieren möchten, müssen Sie es nur einmalig mit Ihrem Computer verbinden, indem Sie die BOOTSEL-Taste gedrückt halten.



Auf diese Weise wird Pico im Bootloader-Modus verbunden und von Ihrem Computer als externen Speicher erkannt. Verbinden Sie Pico mit Ihrem Computer, indem Sie die Bootsel-Taste gedrückt halten. Nachdem Sie Pico als Flash-Speicher des Computers gesehen haben, aktivieren Sie Ihre Karte, indem Sie zu Werkzeuge>Board>Arduino Mbed OS RP2040 Boards> Raspberry Pi Pico gehen.



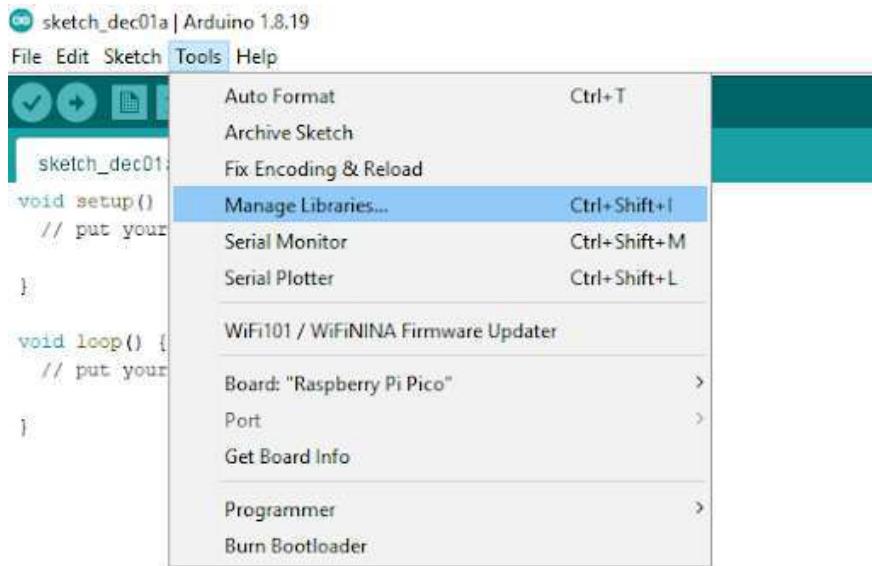
Schreiben Sie den Code in das Feld Nummer 1 unten und folgen Sie dem Datei>Speichern-Pfad und speichern Sie ihn irgendwo auf Ihrem Computer unter dem Namen „Blink“.



Nach dem Speichervorgang müssen wir auf die Schaltfläche „Hochladen“ klicken, um den Code zu kompilieren und ihn in Pico zu speichern. Wenn wir „Hochladen abgeschlossen“ am unteren Rand sehen, wird unser Code in Pico ausgeführt und die integrierte LED wird mit Intervallen von 1 Sekunde blinken. Wichtiger Hinweis: Während dem Programmieren von Picobricks mit der Arduino IDE verbinden Sie es mit Ihrem Computer, indem Sie die BOOTSEL-Taste beim ersten Mal von Micropython oder Microblocks Firmware drücken. Sie müssen die BOOTSEL-Taste für nachfolgende Code-Uplands nicht drücken. Viel Spaß bei den Projekten :)

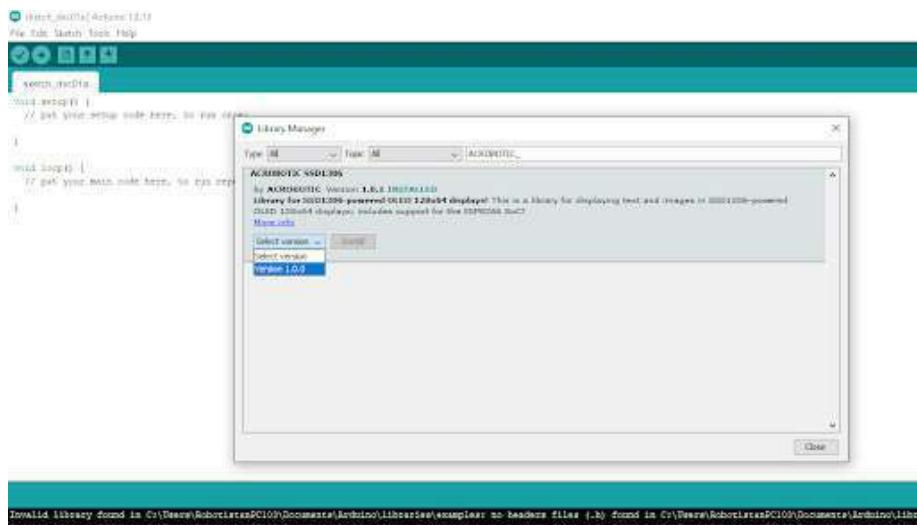
### 1.3.2. Wie fügt man eine Arduino-Bibliothek hinzu?

Um eine neue Bibliothek in Ihre Arduino IDE zu installieren, können Sie den Bibliotheksmanager verwenden. Öffnen Sie die IDE und klicken Sie auf das „Werkzeuge“ Menü und dann auf Werkzeuge > Bibliotheken verwalten.



Dann öffnet sich der Bibliotheksmanager und Sie finden eine Liste von Bibliotheken, die bereits installiert oder zur Installation bereit sind.

Durch Eingabe des Namens der Bibliothek, die Sie installieren möchten, können Sie nach der Bibliothek suchen und dann die Version der Bibliothek auswählen. Klicken Sie schließlich auf die Schaltfläche „installieren“ und warten Sie, bis die Installation abgeschlossen ist.



Die Installation der Bibliothek hängt von Ihrer Verbindungsgeschwindigkeit ab. Wenn die Installation abgeschlossen ist, sehen Sie „INSTALLIERT“ neben der Bibliothek. Auf diese Weise können Sie einfach die Bibliotheken installieren, die Sie gemäß den Codes, die Sie geschrieben haben, oder dem Projekt, das Sie erstellt haben, benötigen.

Wir definieren unsere Bibliothek wie unten gezeigt.

sketch\_dec01a | Arduino 1.8.19

File Edit Sketch Tools Help



sketch\_dec01a §

```
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"
|
#define TRIGGER_PIN 15
#define ECHO_PIN     14
#define MAX_DISTANCE 400
```

# PROJEKTE

## 2.1. Blinken

Im wirklichen Leben übernimmt der Mitarbeiter, der gerade anfängt, den Job zu lernen, zunächst die grundlegendste Aufgabe. Der Reinigungskraft lernt zuerst, den Besen zu benutzen, der Koch lernt, die Küchenutensilien zu verwenden, der Kellner, ein Tablett zu tragen. Wir können diese Beispiele erhöhen. Der erste Code, den Neulinge in der Softwareentwicklung schreiben, ist als „Hallo Welt“ bekannt. Das Drucken von „Hallo Welt“ sofort, wenn das Programm auf dem Bildschirm oder Konsolenfenster in der verwendeten Sprache startet, ist der erste Schritt in der Programmierung. Wie ein Baby, das anfängt zu krabbeln... Der erste Schritt zum Roboter-Coding, auch bekannt als physikalisches Programmieren, ist die Blink-Anwendung. Es bedeutet, dem Roboter-Coding zuzuwinkeln. Indem man einfach eine LED an die Platine anschließt, wird der Code so geschrieben, dass die LED kontinuierlich blinkt. Fragen Sie Menschen, die sich im Bereich des Roboter-Codings weiterentwickelt haben, wie sie zu diesem Niveau gekommen sind. Die Antwort, die sie Ihnen geben werden, beginnt so: Es begann alles mit einer blinkenden LED!

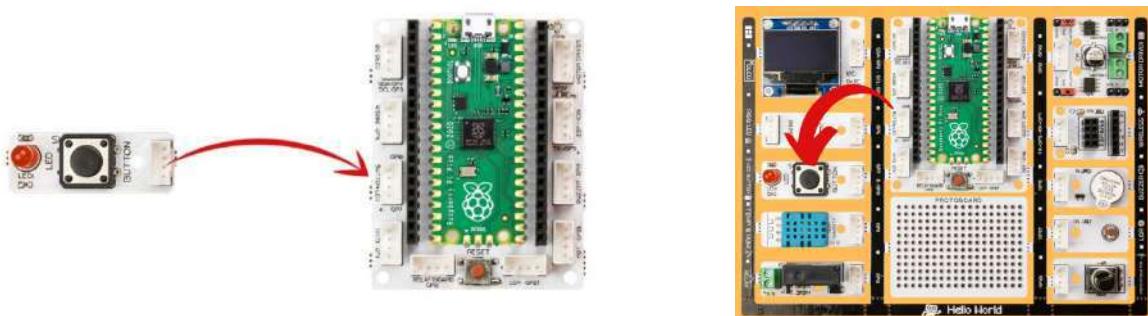
LEDs sind die Sprache elektronischer Geräte. Dank der LEDs informiert der Programmierer die Benutzer darüber, in welcher Phase der Aufgabe sich das Gerät befindet, welches Problem vorliegt, falls vorhanden, und welche Optionen aktiv sind. In diesem Projekt lernen Sie die Arten von LEDs kennen, die darauf mit Picobricks vorhanden sind, und erfahren, wie man sie programmiert.

### 2.1.1. Projektdetails und Algorithmus

Es gibt 1 x 5mm rote LED und 1 x WS2812B RGB LED auf Picobricks. Während normale LEDs in einer Farbe leuchten können, können RGB-Farben in verschiedenen Farben leuchten, sowohl Primär- als auch Sekundärfarben. In diesem Projekt werden wir die rote LED auf Picobricks verwenden.

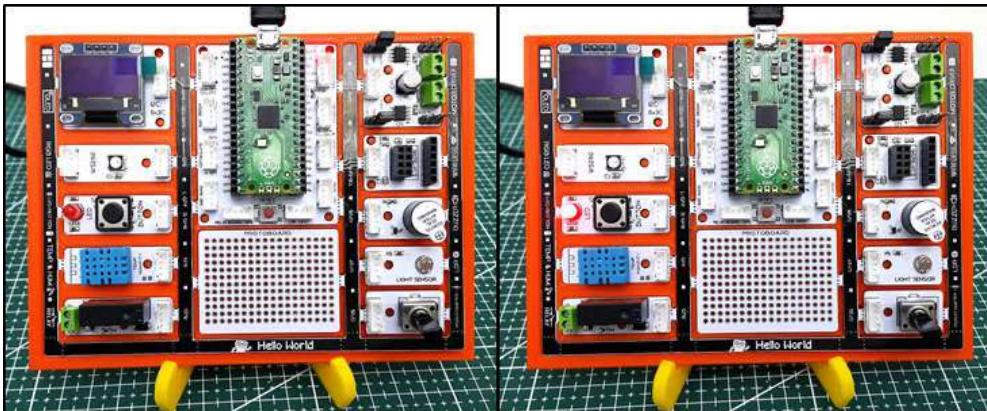
Im Projekt werden wir die notwendigen Codes schreiben, um die rote LED auf Picobricks einzuschalten, sie nach einer bestimmten Zeit auszuschalten, sie nach einer bestimmten Zeit wieder einzuschalten und diese Prozesse kontinuierlich zu wiederholen.

### 2.1.2. Schaltplan



Sie können die Module von Picobricks ohne Verkabelung programmieren und ausführen. Wenn Sie die Module von der Platine trennen möchten, sollten Sie die Modulverbindungen mit Grove-Kabeln herstellen.

## 2.1.3. Projektbild



## 2.1.4. Projektvorschlag

Können wir die LED mit unterschiedlichen Zeitintervallen zum Leuchten bringen? Zum Beispiel; das Blinken der LED mehrmals pro Sekunde, mehrmals alle halbe Sekunde.

## 2.1.5. Programmierung des Projekts mit MicroBlocks

Wenn Sie die MicroBlocks-Picobricks-Verbindung und die Bibliotheksinstallation durchgeführt haben, sind die Schritte, die Sie für das erste Projekt befolgen müssen, in der folgenden Tabelle detailliert beschrieben.

1	Ziehe den "when started"-Block in den Code-Schreibbereich im Steuerungsmenü, damit der Code, den du geschrieben hast, wenn Picobricks startet, zuerst ausgeführt wird.	
2	Ziehe dann den "forever"-Block aus dem Steuerungsmenü und füge ihn unter den "when started"-Block hinzu, so dass die Codes, die du schreibst, kontinuierlich ausgeführt werden, solange Picobricks läuft.	
3	Ziehe den "PicoBricks set red LED"-Block aus den Code-Blöcken in der Picobricks-Bibliothek und lege ihn in den "forever"-Block, damit die rote LED leuchtet. Teste, ob die rote LED leuchtet, indem du die Starttaste drückst.	

4	<p>Um die rote LED auszuschalten, klicke einmal auf das Kontrollkästchen im "PicoBricks set red LED"-Block, um das Kontrollkästchen auf rot, also aus, zu setzen, und teste, ob die LED erlischt, indem du die Starttaste erneut drückst.</p>	
5	<p>Nachdem wir die rote LED mit dem Codeblock blitzen lassen haben, werden wir die notwendigen Codes schreiben, damit die LED in bestimmten Zeitintervallen selbst blitzt. Ziehen Sie den Block „Warte 500 Millisekunden“ aus der Kategorie „Steuerung“ und fügen Sie ihn unter dem Block „PicoBricks setze rotes LED“ hinzu.</p>	
6	<p>Fügen Sie jetzt den roten LED-Block des Picobricks-Sets wieder unter den Block "warte 500 Millisekunden" hinzu und deaktivieren Sie das Kontrollkästchen. Fügen Sie dann den Block "warte 500 Millisekunden" wieder unten hinzu. Wenn Sie die Starttaste drücken, werden Sie sehen, dass die rote LED auf den Picobricks in Intervallen von 500 Millisekunden blinkt. Die Zahl 500 im Block "warte 500 Millisekunden" steht für Millisekunden. Sie können diese Zahl nach Belieben ändern. Wenn sie 1000 erreicht, blinkt die rote LED in 1000 Millisekunden, d.h. in Intervallen von 1 Sekunde.</p>	

[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

### 2.1.6. MicroPython-Codes des Projekts

```
from machine import Pin #um auf die Hardware
```

auf dem Pico zuzugreifen

```
import utime #Zeitbibliothek
```

```
led = Pin(7, Pin.OUT) #digitalen Pin 7 als Ausgang für LED initialisieren
```

```
while True: #while-Schleife
```

```
    led.toggle() #LED ein&aus Status
```

```
    utime.sleep(0.5) #warte eine halbe Sekunde
```

## 2.1.7. Arduino C-Codes des Projekts

```
void setup() {  
    // Fügen Sie hier Ihren Setup-Code ein, der einmal ausgeführt wird:  
    pinMode(7, OUTPUT); // initialisieren Sie den digitalen Pin 7 als Ausgang  
}  
  
void loop() {  
    // Fügen Sie hier Ihren Hauptcode ein, der wiederholt ausgeführt wird:  
    digitalWrite(7, HIGH); // Schalten Sie die LED ein, indem Sie die Spannung auf HIGH setzen  
    delay(500); // warten Sie eine halbe Sekunde  
    digitalWrite(7, LOW); // Schalten Sie die LED aus, indem Sie die Spannung auf LOW setzen  
    delay(500); // warten Sie eine halbe Sekunde  
}
```

GitHub Blink-Projektseite



<http://rbt.ist/link>

## 2.2. Aktion - Reaktion

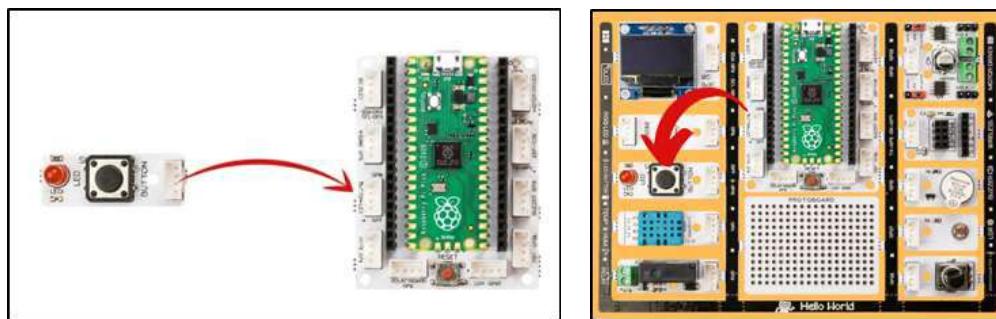
Wie Newton in seinen Bewegungsgesetzen erklärte, erfolgt eine Reaktion auf jede Aktion. Elektronische Systeme erhalten Befehle von Benutzern und führen ihre Aufgaben aus. Üblicherweise wird dafür ein Tastenfeld, ein Touchscreen oder ein Knopf verwendet. Elektronische Geräte reagieren verbal, schriftlich oder visuell, um den Benutzer darüber zu informieren, dass ihre Aufgabe beendet ist und was während der Aufgabe geschieht. Neben der Information des Benutzers über diese Reaktionen kann es helfen zu verstehen, wo der Fehler bei einer möglichen Fehlfunktion liegen könnte. In diesem Projekt lernen Sie, wie Sie einen Befehl vom Benutzer in Ihren Projekten empfangen und darauf reagieren, indem Sie das Knopf-LED-Modul von Picobricks programmieren.

### 2.2.1. Projektdetails und Algorithmus

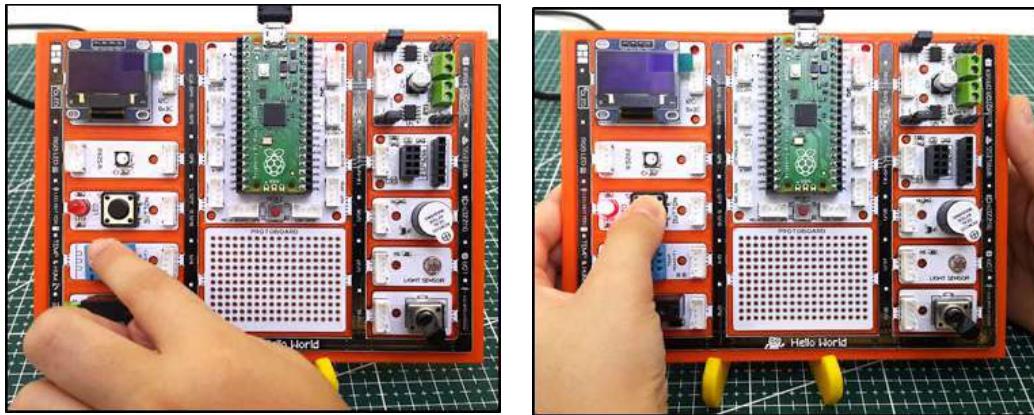
In elektronischen Systemen werden verschiedene Arten von Tasten verwendet. Verriegelte Tasten, Drucktasten, Schalter... Auf Picobricks gibt es 1 Drucktaste. Sie funktionieren wie ein Schalter, sie leiten Strom, wenn sie gedrückt werden, und leiten keinen Strom, wenn sie losgelassen werden. Im Projekt werden wir den Druckstatus verstehen, indem wir überprüfen, ob die Taste Strom leitet oder nicht. Wenn sie gedrückt ist, leuchtet die LED, wenn sie nicht gedrückt ist, schalten wir die LED aus.

### 2.2.2. Schaltplan

Sie können die Module von Picobricks ohne Verkabelung programmieren und ausführen. Wenn Sie die Module von der Platine trennen möchten, sollten Sie die Modulverbindungen mit Grove-Kabeln herstellen.



## 2.2.3. Projektbild

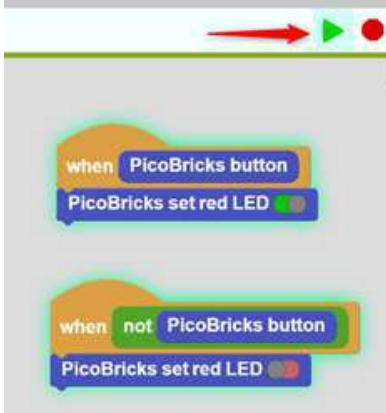


## 2.2.4. Projektvorschlag

In diesem Projekt leuchtet die LED, wenn die Taste gedrückt wird, und die LED erlischt, wenn die Taste losgelassen wird. Sie können die notwendigen Codes schreiben, damit die LED leuchtet, wenn die Taste einmal gedrückt wird, und die LED erlischt, wenn sie erneut gedrückt wird.

## 2.2.5. Programmierung des Projekts mit MicroBlocks

1	Wir ziehen den „when“-Block aus der Kategorie Steuerung, da es sich um einen Befehl handelt, den wir im Falle eines Tastendrucks ausführen müssen. Dieser Block überprüft ständig die Bedingung, die wir festgelegt haben, und wenn die Bedingung wahr ist, führt er den Befehl darunter aus.	
2	Platziere den PicoBricks-Buttonblock in der Picobricks-Kategorie im when-Block, da unsere Bedingung ist, dass der PicoBricks-Button gedrückt wird.	
3	Wir möchten, dass die rote LED in Picobricks leuchtet, wenn die Bedingung erfüllt ist. Platziere also den PicoBricks-Set roten LED-Block in der Picobricks-Kategorie unter dem when-Block.	

4	<p>Wir möchten, dass die rote LED aus bleibt, wenn der Knopf nicht gedrückt ist. Ziehe also den zweiten when-Block aus der Steuerungskategorie. Im Bedingungsfeld platziere den Block aus der Operatoren-Kategorie, um den Ausdruck zu erstellen, wenn der Knopf nicht gedrückt ist.</p>	
5	<p>Platziere den PicoBricks-Buttonblock aus der Picobricks-Kategorie im not-Block, um den Ausdruck zu erstellen, wenn der PicoBricks-Button nicht gedrückt ist.</p>	
6	<p>Wir möchten, dass die rote LED aus bleibt, wann immer der Knopf nicht gedrückt ist. Platziere also den PicoBricks-Set rot LED-Block aus der Picobricks-Kategorie unter dem when-Block und schalte den Schalter aus.</p>	
7	<p>Wenn du die Starttaste von MicroBlocks drückst, wird der Code in Echtzeit ausgeführt. Wenn du den Knopf auf Picobricks drückst, wird die rote LED eingeschaltet, und wenn du ihn loslässt, wird sie ausgeschaltet.</p>	

[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.2.6. MicroPython-Codes des Projekts

```
from machine import Pin #um auf die Hardware von picobricks zuzugreifen
led = Pin(7, Pin.OUT) #digitalen Pin als Ausgang für LED initialisieren
push_button = Pin(10, Pin.IN,Pin.PULL_DOWN) #digitalen
```

Pin 10 als Eingang initialisieren

while True: #while-Schleife

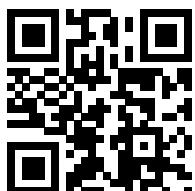
```
logic_state = push_button.value() #Status des Tasters ein/aus
```

```
if logic_state == True: #prüfe den Knopf und ob er eingeschaltet ist  
    led.value(1) #schalte die LED ein  
else:  
    led.value(0) #schalte die LED aus
```

## 2.2.7. Arduino C-Codes des Projekts

```
void setup() {  
    // füge deinen Setup-Code hier ein, um ihn einmal auszuführen:  
    pinMode(7, OUTPUT); //initialisiere digitalen Pin 7 als Ausgang  
    pinMode(10, INPUT); //initialisiere digitalen Pin 10 als Eingang  
  
}  
  
void loop() {  
    // Fügen Sie hier Ihren Hauptcode ein, der wiederholt ausgeführt wird:  
    if (digitalRead(10) == 1){ //prüfe den Knopf und ob er eingeschaltet ist  
  
        digitalWrite(7, HIGH); //schalte die LED ein, indem du die Spannung auf HIGH setzt  
    }  
    else{  
        digitalWrite(7, LOW); //schalte die LED aus, indem du die Spannung auf LOW setzt  
    }  
    delay(10); //warte eine halbe Sekunde  
}
```

## GitHub Action - Reaktionsprojektseite



<http://rbt.ist/actionreaction>

## 2.3. Autonome Beleuchtung

Es wird als autonomer Zustand bezeichnet, wenn elektronische Systeme eine Entscheidung basierend auf den gesammelten Daten treffen und die gegebene Aufgabe automatisch ausführen. Die Komponenten, die es elektronischen Systemen ermöglichen, Daten aus ihrer Umgebung zu sammeln, werden Sensoren genannt. Viele Daten wie der Lichtpegel in der Umgebung, wie viele Grad die Lufttemperatur beträgt, wie viele lt/min der Wasserfluss ist, wie laut das Geräusch ist, werden von den Sensoren gesammelt und als elektrische Signale an PicoBricks übertragen, das sind Daten. Es gibt viele Sensoren in Picobricks. Zu wissen, wie man Daten von Sensoren erhält und wie man diese Daten interpretiert und nutzt, wird die Projektideen verbessern, wie das Lesen eines Buches den Wortschatz verbessert. In diesem Projekt werden wir mit PicoBricks ermöglichen, dass die LED eingeschaltet wird, wenn die Lichtmenge abnimmt, um die Funktionsweise der Systeme zu verstehen, bei denen die Beleuchtung automatisch eingeschaltet wird, wenn es dunkel wird.

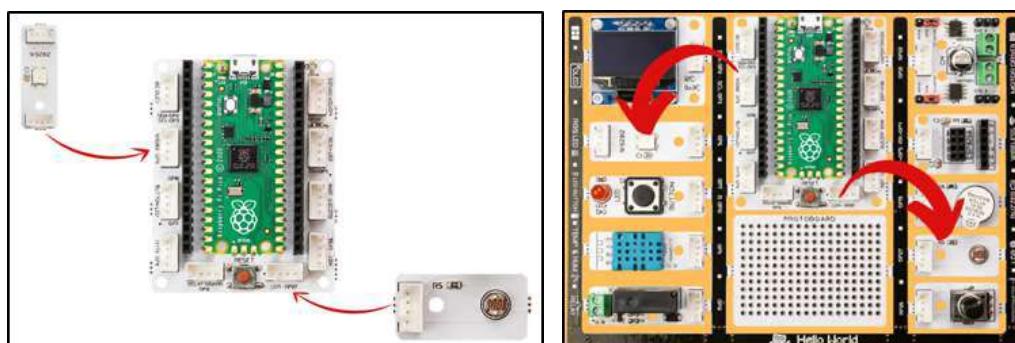
### 2.3.1. Projektdetails und Algorithmus

Sensoren sind elektronische Komponenten, die Daten in externen Umgebungen erfassen und Daten an Mikrocontroller senden. Der LDR-Sensor erkennt ebenfalls die Lichtmenge in der Umgebung und sendet analoge Werte. In unserem Projekt werden wir zuerst die eingehenden Daten überprüfen, wenn die Umgebung hell und dunkel ist, indem wir die LDR-Sensorwerte lesen, dann werden wir eine Grenze gemäß diesen Daten festlegen, und wenn die Lichtmenge unter dieser Grenze liegt, werden wir die RGB-LED von Picobricks ausschalten, andernfalls werden wir die

LED ausschalten.

### 2.3.2. Schaltplan

Sie können die Module von Picobricks ohne Verkabelung programmieren und ausführen. Wenn Sie die Module von der Platine trennen möchten, sollten Sie die Modulverbindungen mit Grove-Kabeln herstellen.



### 2.3.3. Projektbild

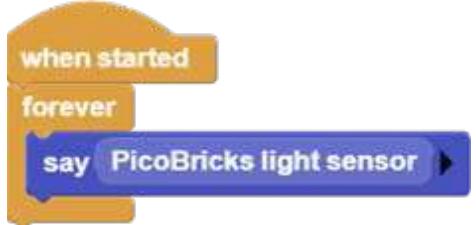


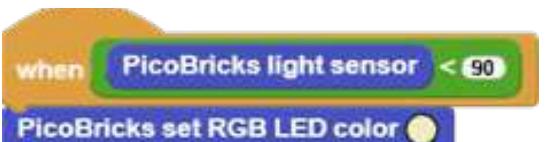
### Projektvorschlag

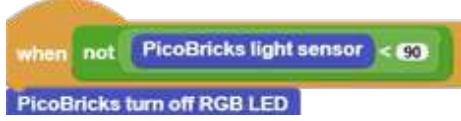
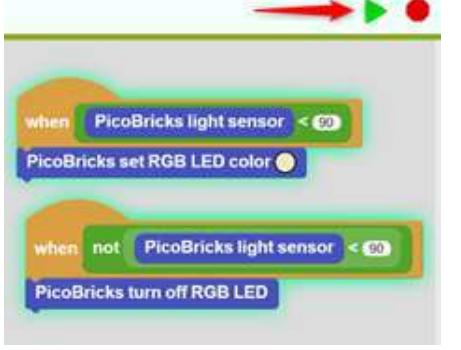
In diesem Projekt haben wir die LED am LDR-Sensor-Daten auf Picobricks eingeschaltet, wenn die Umgebung dunkel war, und die LED ausgeschaltet, wenn es hell war. Durch die Verarbeitung der LDR-Sensordaten können Sie eine Nachtlampe oder Tischlampe in Ihrem Zuhause programmieren, die sich automatisch im Dunkeln einschaltet. Sie können dafür das Relais auf Picobricks verwenden.

### 2.3.5. Programmierung des Projekts mit MicroBlocks

1	Zuerst ziehen Sie den Block „wenn gestartet“ aus dem Menü „Steuerung“ und lassen ihn im Code-Schreibbereich fallen, damit der Code, den Sie geschrieben haben, beim Starten von Picobricks ausgeführt wird.	
2	Dann ziehen Sie den Block „für immer“ aus dem Menü „Steuerung“ und fügen ihn unter den Block „wenn gestartet“ hinzu, damit der Code, den Sie geschrieben haben, kontinuierlich ausgeführt wird, solange Picobricks läuft.	
3	Um eine autonome Beleuchtung bereitzustellen, müssen wir zuerst die Werte vom LDR-Sensor sehen, wenn die Umgebung hell und dunkel ist, und wir müssen entsprechend auf diese Werte reagieren. Ziehen Sie dazu den Block „say123“ aus der Kategorie „Ausgabe“ in den „für immer“-Block.	

4	<p>Ziehen Sie dann den PicoBricks Lichtsensorblock aus der Kategorie Picobricks und lassen Sie ihn in den Kreis, der 123 im „say“-Block sagt. Sehen Sie sich die Werte vom Sensor an, indem Sie die Starttaste drücken. Der Lichtsensorblock gibt den Umgebungslichtpegel als Prozentsatz (%) an. Sie sollten einen Wert von 100 bei vollem Licht sehen, 90 und darunter, wenn Sie es mit Ihrer Hand abdecken, und Werte nahe 0 in völliger Dunkelheit. Sie können die Codes löschen, nachdem Sie die Werte gesehen haben.</p>	
5	<p>Jetzt, wenn die Umgebung dunkel ist, ziehen Sie den „wenn“-Block aus der Kategorie „Steuerung“ in den Code-Schreibbereich, so dass die RGB-LED leuchtet. Im Gegensatz zum Block „wenn gestartet“ überprüft dieser Block ständig die Bedingung, die wir festgelegt haben, und wenn die Bedingung wahr ist, führt er den Befehl darunter aus. Der Block „wenn gestartet“ führt die Blöcke darunter ab dem Moment aus, in dem Sie die Starttaste drücken.</p>	
6	<p>Um dann die Bedingung im „wenn“-Block zu definieren, ziehen Sie den Block „3&lt;4“ aus der Kategorie „Operatoren“ und lassen ihn in den runden Teil des „wenn“-Blocks fallen. Der Block „3&lt;4“ führt Größen-Kontrollen in Bedingungsoperationen durch. Sie können die Blöcke, die Sie steuern möchten, in die Felder 3 und 4 im Block einfügen. In der Programmierung werden häufig verschiedene Operatoren wie gleich, ungleich, größer als, kleiner als verwendet. In diesem Projekt werden wir überprüfen, ob der Wert vom Sensor kleiner als 90 ist.</p>	
7	<p>Jetzt ziehen Sie den PicoBricks-Lichtsensor block aus der Kategorie Picobricks und lassen ihn in den Kreis, der 3 im „3&lt;4“-Operator im „wenn“-Block sagt.</p>	

8	<p>Um die Bedingung im when-Block zu vervollständigen, löschen Sie die Zahl 4 und geben Sie 90 von der Tastatur ein. Auf diese Weise wird das Programm überprüfen, ob die Werte des Picobricks-Sensors weniger als 90 % sind und die Codes im when-Block ausführen, wenn sie weniger als 90 % sind.</p>	
9	<p>Ziehen Sie den Picobricks-Satz RGB-LED-Block aus der Picobricks-Kategorie in den when-Block, damit die LED leuchten kann, wenn die Umgebung dunkel ist, das heißt, wenn der LDR-Sensorwert weniger als 90 beträgt. Sie können die Farbe, die Sie möchten, aus der Farbpalette auswählen, die sich öffnet, wenn Sie auf den grünen Kreis klicken.</p>	
10	<p>Bis zu diesem Punkt haben wir die notwendigen Codes geschrieben, damit die LED leuchtet, wenn die Umgebung dunkel ist. Jetzt müssen wir die Operationen zum Programm hinzufügen, wenn die Umgebung hell ist, das heißt, wenn die Bedingung nicht erfüllt ist. Dafür nehmen Sie den when-Block aus der Kategorie Steuerung und lassen ihn im Code-Schreibbereich.</p>	
11	<p>Im when-Block platzieren Sie die Kategorie der Operatoren block im Bedingungsfeld. Auf diese Weise können wir die Bedingung erstellen, wenn der LDR-Sensorwert nicht weniger als 90 beträgt.</p>	
12	<p>Ziehe erneut den 3&lt;4 Operator aus der Kategorie der Operatoren in den Nicht-Block.</p>	
13	<p>Ziehe den Picobricks-Lichtsensor-Block aus der Picobricks-Kategorie und platziere ihn im Kreis, der die Zahl 3 im 3&lt;4 Operator anzeigt, und lösche die Zahl 4 und schreibe 90.</p>	

14	Ziehe den PicoBricks RGB LED ausschalten Block aus der Picobricks-Kategorie unter den Wenn-Block, sodass die RGB LED ausgeht, wenn der LDR-Sensorwert nicht kleiner als 90 ist.	
15	Teste deinen Code, indem du die Starttaste drückst. Wenn alles gut gelaufen ist, wird die RGB LED in der von dir angegebenen Farbe leuchten, wenn du den Picobricks LDR-Sensor mit deiner Hand schließt, und sie wird ausgehen, wenn du deine Hand hebst.	

[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

### 2.3.6. MicroPython-Codes des Projekts

```

import time
from machine import Pin, ADC
from picobricks import WS2812
#define die Bibliothek

ldr = ADC(Pin(27))
ws = WS2812(6, brightness=0.4)
#define die Eingangs- und Ausgangspins

#define Farben
ROT = (255, 0, 0)
GRÜN = (0, 255, 0)
BLAU = (0, 0, 255)

FARBEN = (ROT, GRÜN, BLAU)
#RGB Farbcode

while True:#while Schleife
print(ldr.read_u16()) #gibt den Wert des LDR-Sensors auf dem Bildschirm aus.

if(ldr.read_u16()>10000):#lassen Sie uns den LDR-Sensor überprüfen
    for farbe in FARBEN:

```

```

#LDR einschalten
    ws.pixels_fill(farbe)
    ws.pixels_show()

else:
ws.pixels_fill((0,0,0)) #RGB ausschalten
    ws.pixels_show()

```

### 2.3.7. Arduino C Codes des Projekts

```

#include <Adafruit_NeoPixel.h>
#define PIN      6
#define ANZAHLLEDS  1
#define LIGHT_SENSOR_PIN 27
Adafruit_NeoPixel leds = Adafruit_NeoPixel(NUMLEDS, PIN, NEO_GRB + NEO_KHZ800);
//Bibliotheken definieren
int delayval = 250; // Verzögerung für eine halbe Sekunde

void setup()
{
    leds.begin();
}

void loop() {
int analogValue = analogRead(LIGHT_SENSOR_PIN);
for(int i=0;i < NUMLEDS;i++)
{
    if (analogValue > 200) {
// pixels.Color nimmt RGB-Werte, von 0,0,0 bis 255,255,255
        leds.setPixelColor(i, leds.Color(255,255,255));
    leds.show(); // Dies sendet die aktualisierte Pixel-Farbe an die Hardware.
        delay(delayval);
    }
sonst {
    leds.setPixelColor(i, leds.Color(0,0,0)); // weißer Farbcode
    leds.show(); // Dies sendet die aktualisierte Pixel-Farbe an die Hardware.
}

}
delay(10);

```

}

GitHub Autonomes Beleuchtungsprojekt Seite



<http://rbt.ist/autonomouslighting>

## 2.4. Thermometer

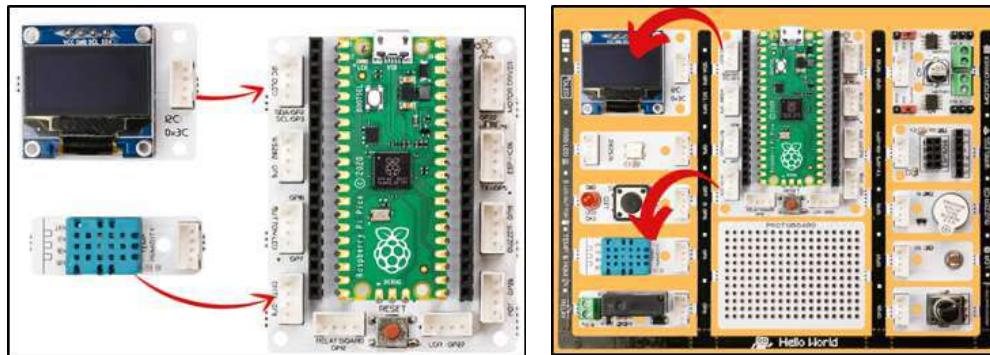
Sensoren sind die Sinnesorgane elektronischer Systeme. Wir nutzen unsere Haut zum Fühlen, unsere Augen zum Sehen, unsere Ohren zum Hören, unsere Zunge zum Schmecken und unsere Nase zum Riechen. Es gibt bereits viele Sinnesorgane (Sensoren) in den Picobrix. Auch neue können hinzugefügt werden. Sie können mit der Umgebung interagieren, indem Sie Feuchtigkeit, Temperatur, Licht und viele weitere Sensoren verwenden. Picobricks können die Umgebungstemperatur messen, ohne dass weitere Umweltkomponenten erforderlich sind.

Die Umgebungstemperatur wird in Gewächshäusern, Brutkästen und in Umgebungen verwendet, die für die Transport von Medikamenten genutzt werden, kurz gesagt in Situationen, in denen die Temperaturänderung ständig überwacht werden muss. Wenn Sie in Ihren Projekten eine Operation zur Temperaturänderung durchführen möchten, sollten Sie wissen, wie man die Umgebungstemperatur misst. In diesem Projekt werden Sie ein Thermometer mit Picobricks vorbereiten, das die Umgebungstemperatur auf dem OLED-Bildschirm anzeigt.

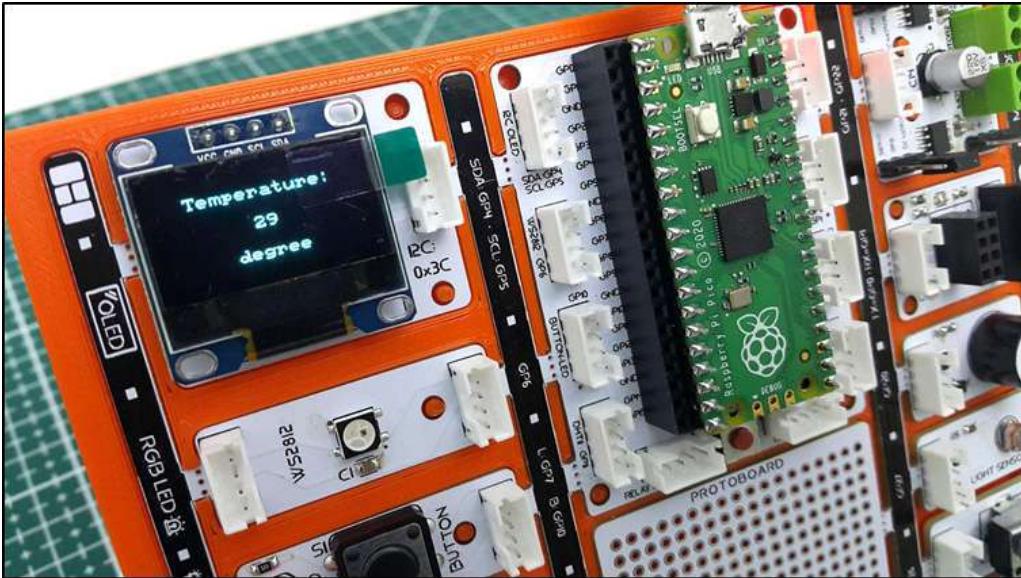
### 2.4.1. Projektdetails und Algorithmus

Picobricks verfügt über ein DHT11-Modul. Dieses Modul kann die Temperatur und Luftfeuchtigkeit in der Umgebung messen und Daten an den Mikrocontroller senden. In diesem Projekt werden wir die notwendigen Codes schreiben, um die Temperaturwerte, die vom DHT11-Temperatur- und Feuchtigkeitssensor gemessen werden, auf dem OLED-Bildschirm anzuzeigen.

### 2.4.2. Schaltplan



## 2.4.3. Bauphasen des Projekts

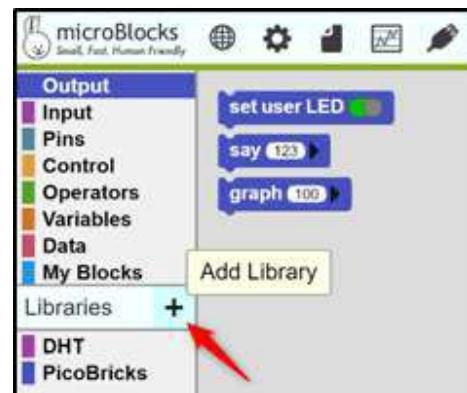


## 2.4.4. Projektvorschlag

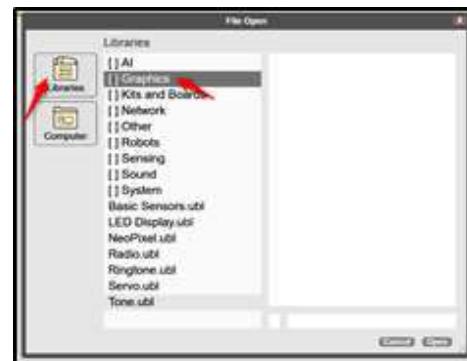
Um Ihr Projekt weiterzuentwickeln, können Sie die rote LED zum Leuchten bringen und einen Warnhinweis auf dem Bildschirm anzeigen, wenn die Temperatur in der Umgebung über 30 Grad steigt.

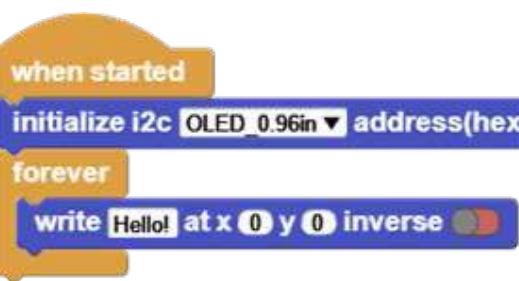
## 2.4.5. Programmierung des Projekts mit Microblocks

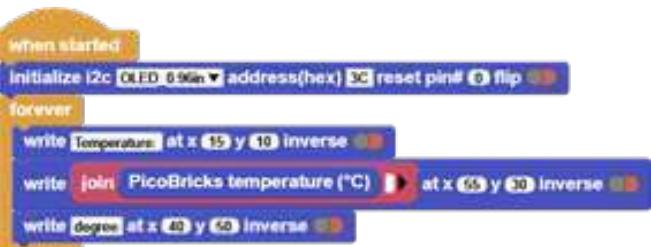
- 1 Zuerst müssen Sie die OLED-Bibliothek in den Microblocks-Editor importieren. Sie müssen dafür auf die Schaltfläche Bibliothek hinzufügen klicken.



- 2 Im sich öffnenden Fenster Datei öffnen klicken Sie auf Bibliothek und dann auf Grafiken.



3	<p>Klicken Sie auf die OLED Graphics.ulb-Bibliothek aus den Grafikbibliotheken und klicken Sie auf die Schaltfläche Öffnen in der unteren rechten Ecke des Fensters. Sie werden sehen, dass die OLED-Bibliothek zur Liste der Codekategorien hinzugefügt wurde.</p>	
4	<p>Nachdem Sie die OLED-Bibliothek hinzugefügt haben, ziehen Sie den Block „wenn gestartet“ aus dem Menü Steuerung und lassen Sie ihn im Code-Schreibbereich fallen, damit die Codes, die Sie geschrieben haben, ausgeführt werden, wenn Picobricks zu laufen beginnt.</p>	
5	<p>Dann müssen Sie den Identifikationsblock des OLED-Displays hinzufügen. Dazu ziehen Sie den Block „i2c OLED initialisieren“ aus der OLED-Kategorie und fügen ihn unter den Block „wenn gestartet“ hinzu.</p>	
6	<p>Ziehen Sie dann den Block „für immer“ aus dem Menü Steuerung an das Ende des OLED-Definitionsblocks, so dass die Codes, die Sie schreiben, kontinuierlich ausgeführt werden, solange Picobricks läuft.</p>	
7	<p>Um Text auf dem OLED-Bildschirm anzuzeigen, ziehen Sie den Block „Hallo! bei x0 y0 umgekehrt schreiben“ aus der OLED-Kategorie in den „für immer“-Block. Wenn Sie die Starttaste drücken, sehen Sie „Hallo!“, was Sie geschrieben haben. Hallo hier! Sie können den Text löschen und den gewünschten Text schreiben, so dass er auf dem Bildschirm angezeigt wird.</p>	
8	<p>Jetzt löschen Sie im Schreibblock den Text „Hallo!“ und schreiben „Temperatur: „ und ändern die x-Position auf 15 und die y-Position auf 10.</p>	

9	<p>Wieder nehmen Sie den Schreibblock aus der Kategorie OLED-Grafiken und ziehen und lassen ihn in den „für immer“-Block fallen und setzen die x-Position auf 55 und die y-Position auf 30. Um numerische Werte auf dem OLED-Bildschirm anzuzeigen, müssen Sie diese numerischen Werte zuerst in textuelle Ausdrücke umwandeln. Dazu ziehen Sie den Block aus der Datenkategorie, „Hallo!“ in den Schreibblock und lassen ihn in den runden Bereich fallen, der sagt. Ziehen Sie dann den Block „PicoBricks Temperatur (C)“ aus der Picobricks-Kategorie und platzieren Sie ihn im Mikro-Kreis im Verbindungsblock und löschen Sie den Text des Blocks.</p>	
10	<p>Schließlich ziehen Sie den Schreibblock aus der Kategorie OLED-Grafiken in den „für immer“-Block. Hallo! Löschen Sie den Text, schreiben Sie die Grad und ändern Sie die x-Position auf 40 und die y-Position auf 50. Durch Drücken des Startknopfes können Sie die Umgebungstemperatur, die von Picobricks' internem Temperatursensor auf dem OLED-Bildschirm gemessen wird, anzeigen.</p>	

[Klicken](#) Sie, um auf die Microblocks-Codes des Projekts zuzugreifen.

#### 2.4.6. Micropython-Codes des Projekts

```
from machine import Pin,I2C,ADC #um auf die Hardware picobricks zuzugreifen
from picobricks import SSD1306_I2C, DHT11 #oled-Bibliothek
import utime #Zeitbibliothek
#um auf die Hardware picobricks zuzugreifen
WIDTH=128
HEIGHT=64
#definiere das Gewicht und die Höhe der picobricks

sda=machine.Pin(4)
scl=machine.Pin(5)
#wir definieren die sda- und scl-Pins für die internen Kommunikationspfade
```

```

i2c=machine.I2C(0, sda=sda, scl=scl, freq=2000000)#bestimme die Frequenzwerte
oled=SSD1306_I2C(WIDTH, HEIGHT, i2c)
pico_temp=DHT11(Pin(11))
current_time=utime.time()
while True:
    if(utime.time() - current_time > 2):
        current_time = utime.time()
        pico_temp.measure()
    oled.fill(0)#OLED löschen
    oled.show()
    temperature=pico_temp.temperature
    humidity=pico_temp.humidity
    oled.text("Temperatur: ",15,10)#"Temperatur: " auf dem OLED bei x=15 y=10 anzeigen
    oled.text(str(int(temperature)),55,25)
    oled.text("Luftfeuchtigkeit: ", 30,40)
    oled.text(str(int(humidity)),55,55)
    oled.show()#auf OLED anzeigen
    utime.sleep(0.5)#eine halbe Sekunde warten

```

## 2.4.7. Arduino C Codes des Projekts

```

#include <Wire.h>
#include <DHT.h>
#include "ACROBOTIC_SSD1306.h"

#define DHTPIN 11
#define DHTTYPE DHT11
//Bibliothek definieren

DHT dht(DHTPIN, DHTTYPE);
float Temperatur;
//definiere die Temperaturvariable

void setup() {
//definiere den DHT-Sensor und den OLED-Bildschirm
    Serial.begin(115200);
    dht.begin();
    Wire.begin();
    oled.init();

```

```
oled.clearDisplay();
}

void loop() {
temperature = dht.readTemperature();
Serial.print("Temp: ");
Serial.println(temperature);
oled.setTextXY(3,1);
oled.putString("Temperatur: ");
//drucke "Temperatur: " auf dem OLED bei x=3 y=1
oled.setTextXY(4,3);
oled.putString(String(temperature));
//drucke den Wert vom Temperatursensor auf dem OLED-Bildschirm bei x=4 y=3
Serial.println(temperature);
delay(100);
}
```

GitHub Thermometer Projektseite



<http://rbt.ist/thermometer>

## 2.5. Grafischer Monitor

Wenn wir uns die elektronischen Geräte um uns herum ansehen, wird uns klar, dass sie viele wechselbare Funktionen haben und von Ingenieuren so gestaltet wurden, dass sie für den Benutzer am nützlichsten sind. Zum Beispiel Beleuchtungssysteme, Kochsysteme, Soundsysteme, Reinigungssysteme. Die Funktionsweise, die Menge, die Methode usw. können von vielen Systembenutzern programmiert werden, um sich zu ändern.

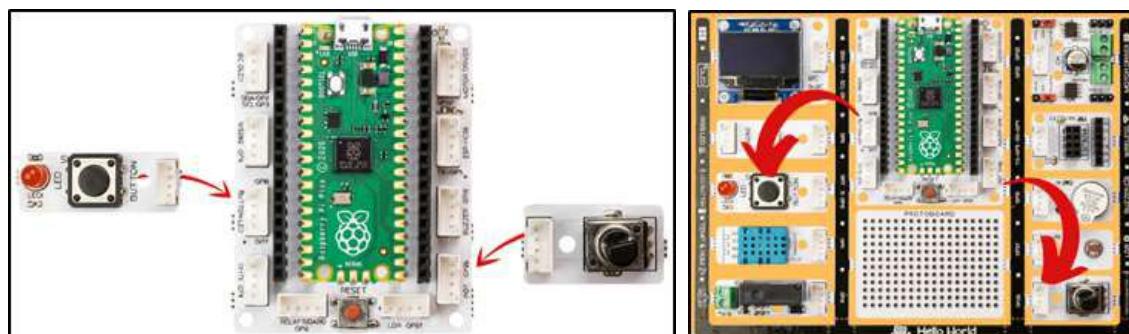
In Robotikprojekten wird bei den Prozessen der Änderung des Lautstärkepegels, der Änderung der Motordrehzahl, der Änderung der Helligkeit des Lichts die elektrische Spannung so gesendet, dass sie einen niedrigeren oder höheren Effekt erzeugt. Durch Verringerung der Frequenz des elektrischen Signals zum Bauteil kann es auf einem niedrigeren Niveau betrieben werden, und durch Erhöhung der Frequenz der ausgehenden elektrischen Signale kann es auf einem höheren Niveau betrieben werden.

In Systemen ohne Bildschirm werden Echtzeit-Grafikmonitore verwendet, um einige Sensoren und Variablen, die am Betrieb des Systems beteiligt sind, zu überwachen. Grafikmonitore erleichtern die Fehlersuche erheblich.

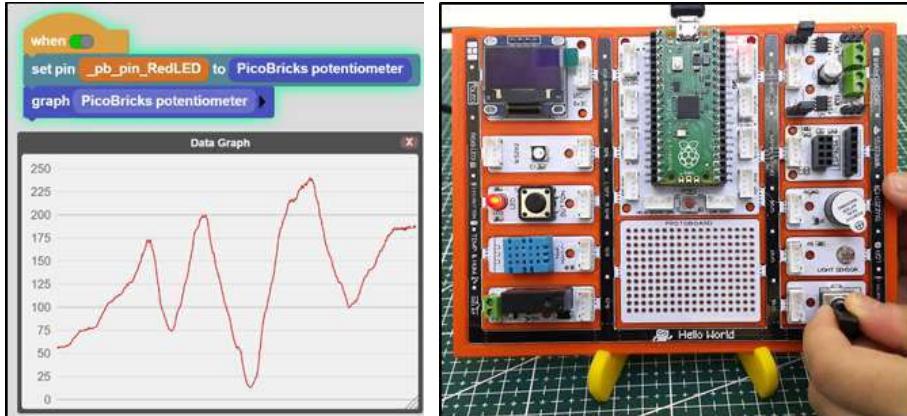
### 2.5.1. Projektdetails und Algorithmus

In diesem Projekt werden wir ein Projekt vorbereiten, in dem wir die Helligkeit der roten LED mit einem Potentiometer erhöhen oder verringern. Darüber hinaus werden wir gleichzeitig die elektrische Veränderung, die während dieses Prozesses auf dem Microblocks-Grafikmonitor auftritt, überwachen. Wenn die Picobricks starten, wird der Wert des Potentiometers kontinuierlich gelesen und der Helligkeitswert der LED wird angepasst. Anwendungen, bei denen der Effekt des elektrischen Signals durch Änderung der Frequenz verringert wird, werden PWM genannt. Wir werden die analogen Werte, die wir vom Potentiometer lesen, als PWM-Signale an die rote LED senden und wir werden in der Lage sein, die Beleuchtungsintensität anzupassen.

### 2.5.2. Schaltplan



## 2.5.3. Projektbild



## Projektvorschlag

Wenn Sie den Potentiometer drehen, können Sie ein Projekt vorbereiten, das die Lautstärke des aus dem Summer kommenden Tons ändert und die fließenden Daten auf dem Grafikmonitor anzeigt.

## 2.5.5. Programmierung des Projekts mit Microblocks

1	Um die Codes vorzubereiten, die so lange laufen, wie Picobricks eingeschaltet ist, ziehen Sie zuerst den „when“-Block aus der Kategorie Steuerung.	
2	Um die Helligkeit der LED anzupassen, können wir ihr eine Spannung von 3,3 V senden, indem wir sie in 1024 Teile aufteilen. Dazu ziehen Sie den Block „set pin 1 to 1023“ aus der Kategorie Pins an das Ende des „when“-Blocks. Anstelle von 1 sollten Sie die GPIO-Nummer schreiben, an die die rote LED auf den Picobricks angeschlossen ist.	
3	Klicken Sie auf das Symbol, um die Pin-Nummer der roten LED in Picobricks hinzuzufügen. Wählen Sie „erweiterte Blöcke anzeigen“ aus dem Dropdown-Menü.	

4	<p>Sie werden die bereitgestellten Variablen in der Kategorie Variablen sehen. Sie können die Pin-Nummern aller PicoBricks-Module mit dem Präfix „_pb_pin_“ finden.</p> <p>Ziehen Sie den Block _pb_pin_RedLED in das Feld mit der Bezeichnung 1, um den Pin darzustellen, an den die rote LED angeschlossen ist.</p>	
5	<p>Der Bereich, der 1023 sagt, ist der Bereich, in dem wir den Strompegel festlegen, der an die rote LED gesendet werden soll. Platzieren Sie hier den PicoBricks-Potentiometerblock aus der Kategorie Picobricks, da wir diesen Wert im Potentiometer erhalten werden.</p>	
6	<p>Ziehen Sie den Block „graph 100“ aus der Kategorie Ausgabe, um die numerischen Daten, die vom Potentiometer gesendet werden, im Diagramm anzuzeigen.</p>	
7	<p>Wenn wir den PicoBricks Potentiometerblock auf den Graphen block legen, schreiben wir den notwendigen Code, damit die Werte auf dem grafischen Monitor erscheinen und das Projekt abgeschlossen wird. Sie können den Wert des Potentiometers ändern, indem Sie Ihren Code ausführen und auf das grafische Symbol klicken.</p> 	

[Klicken](#) Sie, um auf die Microblocks-Codes des Projekts zuzugreifen.

## 2.5.6. Micropython-Codes des Projekts

```
from machine import Pin,ADC,PWM
from utime import sleep
#define libraries
led=PWM(Pin(7))
pot=ADC(Pin(26,Pin.IN))
```

```
#define den Wert, den wir von der LED und dem Potentiometer erhalten  
led.freq(1000)  
  
while True: #while-Schleife  
    led.duty_u16(int((pot.read_u16())))  
  
    print(str(int((pot.read_u16())))) #Schalten Sie die LED entsprechend dem Wert vom  
    Potentiometer ein  
    sleep(0.1) #Verzögerung
```

## 2.5.7. Arduino C-Codes des Projekts

```
void setup() {  
    // Fügen Sie hier Ihren Setup-Code ein, der einmal ausgeführt wird:  
    pinMode (7,OUTPUT); //digitalen Pin 7 als Ausgang initialisieren  
    pinMode (26,INPUT); //digitalen Pin 26 als Eingang initialisieren Serial.begin(9600); //starten  
    serielle Kommunikation  
  
}  
  
void loop() {  
    // Fügen Sie hier Ihren Hauptcode ein, um ihn wiederholt auszuführen:  
    int pot_val = analogRead(26);  
    int led_val = map(pot_val, 0, 1023, 0, 255);  
    digitalWrite(7, led_val);  
    Serial.println(led_val);  
    //Schalte die LED entsprechend dem Wert des Potentiometers ein  
  
    delay(100); //warte  
}
```

---

## GitHub Grafikmonitor-Projektseite



<http://rbt.ist/monitor>

## 2.6. Den Rhythmus beherrschen

Viele Ereignisse in unserem Leben wurden digitalisiert. Eines davon sind Klänge. Der Ton und die Intensität des Klangs können elektrisch verarbeitet werden. So können wir Noten elektronisch extrahieren. Die kleinste Einheit von Klängen, die Musik ausmachen, wird als Note bezeichnet. Jede Note hat eine Frequenz und Intensität. Mit den Codes, die wir schreiben werden, können wir einstellen, welche Note gespielt werden soll und wie lange sie dauern soll, indem wir Frequenz und

Intensität anwenden.

In diesem Projekt werden wir ein Musiksystem vorbereiten, das die Melodie eines Liedes mit dem Buzzer-Modul spielt und den Rhythmus mit dem Potentiometer-Modul mit Picobricks anpasst. Sie werden auch den Umgang mit Variablen lernen, die einen wichtigen Platz in der Programmierterminologie einnehmen, in diesem Projekt.

### 2.6.1. Projektdetails und Algorithmus

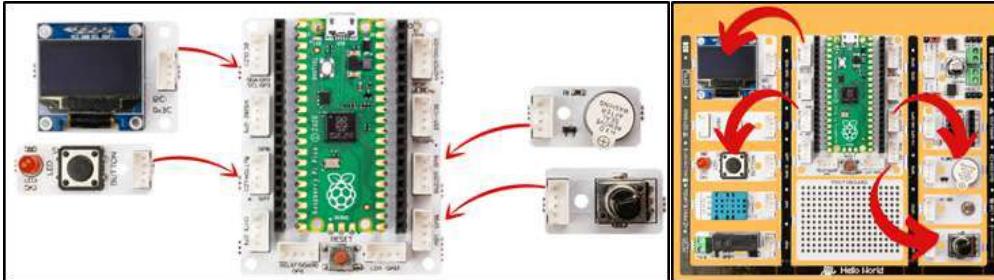
Mit Picobricks können Sie jedes Lied spielen, dessen Noten wir kennen. Wir werden das Button-LED-Modul verwenden, um das Lied zu starten, das Potentiometer-Modul, um die Geschwindigkeit des Liedes anzupassen, und das Buzzer-Modul, um die Noten zu spielen.

Das Potentiometer ist ein analoges Eingabemodul. Es ist ein variabler Widerstand. Wenn die Menge an Strom, die durch ihn fließt, gedreht wird, erhöht und verringert sie sich wie das Öffnen und Schließen eines Wasserhahns. Wir werden die Geschwindigkeit des Liedes steuern, indem wir diese Menge an Strom mit Codes kontrollieren. Buzzer ändern die Lautstärke je nach Intensität des Stroms, der über sie fließt, und die Klangtöne je nach Spannungsfrequenz.

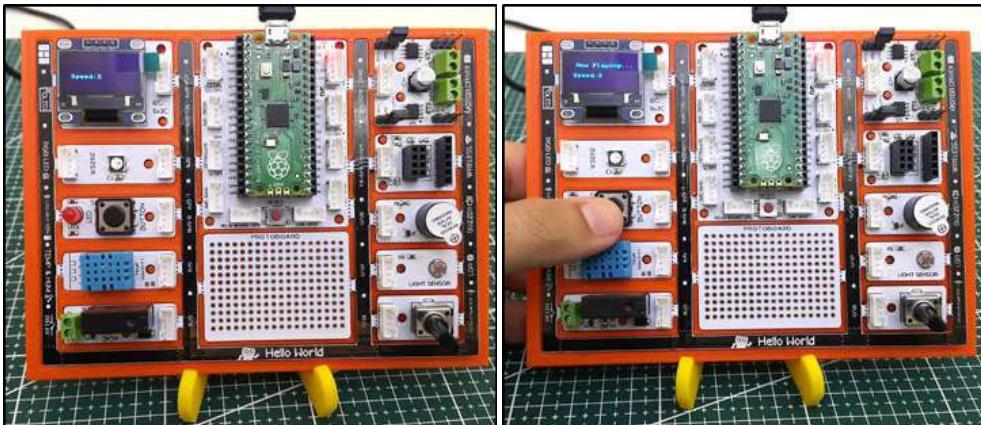
Mit Microblocks können wir die Noten, die wir vom Buzzer-Modul wollen, einfach codieren, indem wir ihre Töne und Dauer anpassen.

Wir werden den Status des Button-Drucks im Projekt überprüfen. Die Melodie wird zu spielen beginnen, wenn der Button gedrückt wird. Während das Lied spielt, werden wir eine Variable namens rthm verwenden, um die Spielzeiten der Noten im gleichen Verhältnis zu erhöhen oder zu verringern. Nachdem Picobricks gestartet ist, werden wir dem Benutzer ermöglichen, die rthm-Variable mit dem Potentiometer anzupassen, entweder während dem Spielen der Melodie oder davor. Solange Picobricks eingeschaltet ist, werden wir den Wert des Potentiometers (0-1023) durch 128 teilen und ihm die rthm-Variable zuweisen. Variablen sind Datenstrukturen, die wir verwenden, wenn wir Werte verwenden möchten, die vom Benutzer oder von Sensoren in unseren Codes geändert werden können. Wenn der Benutzer den Button drückt, um das Lied zu starten, werden wir die Notencodes vorbereiten, die es ermöglichen, die Noten für die Dauer zu spielen, die gemäß der rthm-Variable berechnet wurde.

## 2.6.2. Schaltplan



## 2.6.3. Projektbild

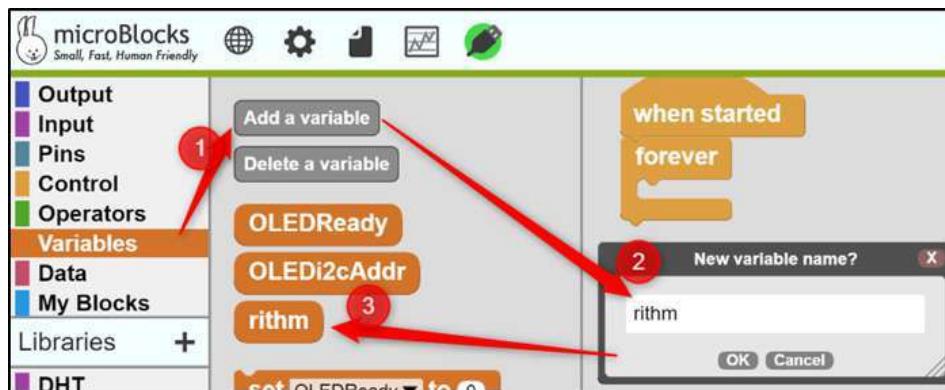


## 2.6.4. Projektvorschlag

Um Ihr Projekt visuell ansprechender zu gestalten, können Sie eine LED in einer anderen Farbe entsprechend der gespielten Note leuchten lassen, die Notennamen und die Spielgeschwindigkeit auf dem OLED-Bildschirm anzeigen.

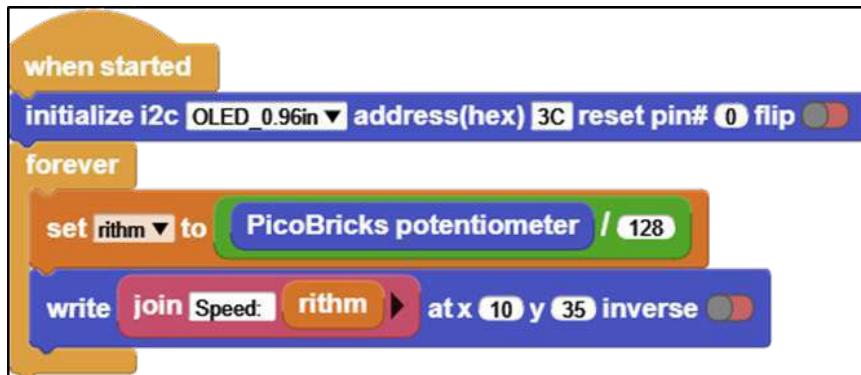
## 2.6.5. Codierung des Projekts mit Microblocks

Ziehen Sie das „Wenn gestartet“-Block aus der Kategorie Steuerung und den „für immer“-Block darunter, um Picobricks den Code kontinuierlich auszuführen, sobald es startet. Fügen Sie die OLED-Display-Bibliothek hinzu und platzieren Sie den Code, der das Display initialisiert, vor dem für immer-Block, damit Sie die Geschwindigkeitseinstellung des Benutzers auf dem Bildschirm anzeigen können. Klicken Sie auf die Schaltfläche Variable hinzufügen in der Kategorie Variablen. Wenn Sie rithm in dem Dialog eingeben und bestätigen, der sich öffnet, wird die rithm-Variable erstellt.



Wählen Sie den Variablennamen als rithm aus, indem Sie auf das schwarze Dreieck im Block „OledReady to 0“ in der Kategorie Variablen klicken. Legen Sie ihn dann in den „für immer“-Block.

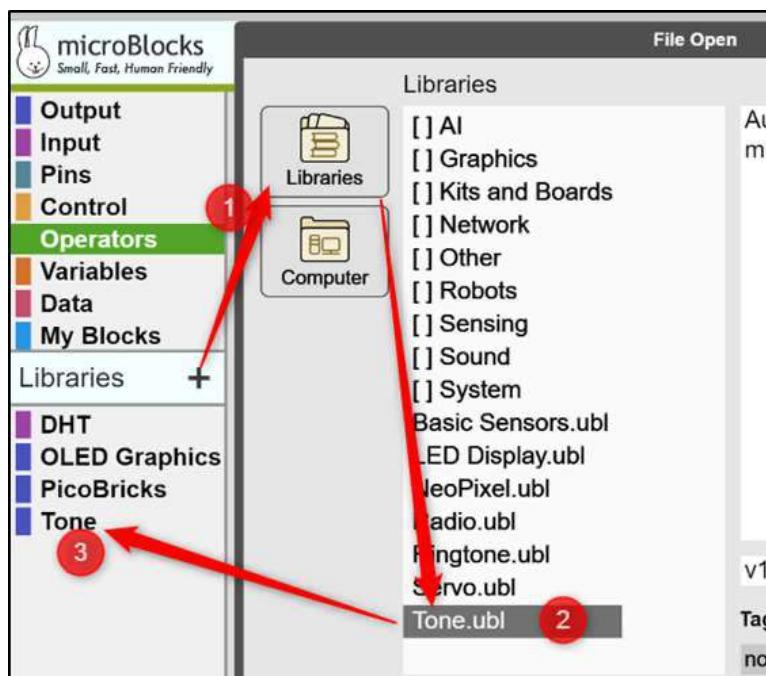
Wir werden den Wert, der vom Potentiometer gelesen wird, durch 128 teilen und ihn verwenden, um die Dauer der Note zu berechnen. Platzieren Sie den 10/2-Block aus der Kategorie Operatoren in den Variablenblock und setzen Sie den Nenner auf 128. Um das Potentiometer zu lesen, ziehen Sie den PicoBricks-Potentiometerblock aus der PicoBricks-Kategorie in das 10-Feld im Teilungsblock. Nehmen Sie den Schreibblock aus der OLED-Kategorie. Im Hello-Feld platzieren Sie den Join-Block aus der Datenkategorie und setzen die rithm-Variable ein.



Ziehen Sie den when-Block aus der Kategorie Steuerung für die kontinuierliche Überwachung des Status des Knopfdrucks auf den Picobricks. Platzieren Sie den Picobricks-Buttonblock darin.



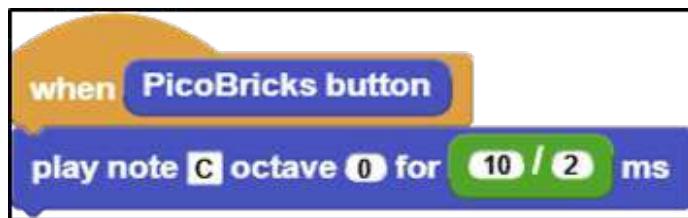
Fügen Sie die Tonbibliothek zum Arbeitsbereich hinzu.



Fügen Sie einen Block

play note C octave 0 for 500 ms

unter dem when-Block hinzufügen. C-Notenname, 0 ermöglicht es Ihnen, den Notenton einzustellen, und 500 ermöglicht es Ihnen, die Spielzeit festzulegen. Wir werden die Spielzeiten der Noten bestimmen, indem wir sie durch die rithm-Variable teilen. Ziehen Sie den 10/2 Block aus der Kategorie Operatoren in das 500-Feld des Spielnotenblocks.



Die rithm-Variable hat den niedrigsten Wert von 0. Wir werden eins zum rithm-Wert hinzufügen, da 0 ein Quotient einer Zahl unendlich sein wird. Ziehen Sie den „10+2“-Summenblock aus der Kategorie Operatoren in das Nenner(2)-Feld der Division. Setzen Sie die rithm-Variable in das Feld 10 und ändern Sie die Zahl 2 in 1. Duplizieren Sie die Blöcke so oft wie die Anzahl der Noten im Lied, indem Sie mit der rechten Maustaste auf den Spielnotenblock klicken und die Optionen Duplizieren und Alle duplizieren auswählen.

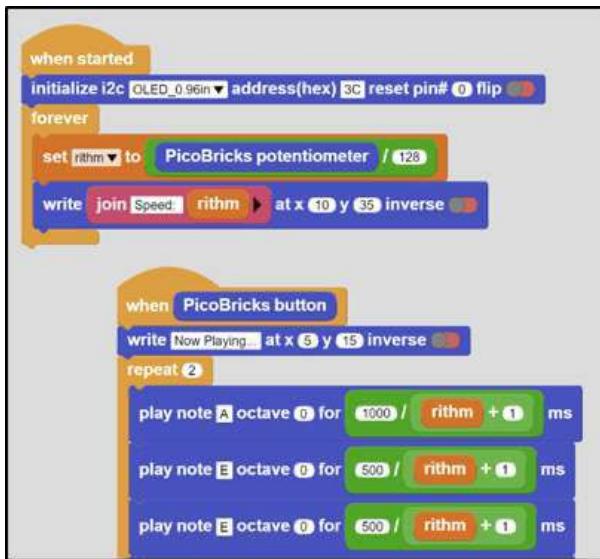
Schreiben Sie die Namen der Noten in der richtigen Reihenfolge. Schreiben Sie die Standard-Schlagzeiten für jede Note in das Feld 10 des Divisionsoperators. Ein Wert von 1000 steht für einen vollen Schlag, 500 steht für einen halben Schlag und 250 steht für einen viertel Schlag. Indem wir den Schreibblock aus der OLED-Kategorie direkt nach dem when-Block hinzufügen, zeigen wir dem Benutzer „Jetzt spielen“. Wenn die Melodie beendet ist, werden wir den Bildschirm löschen.



Diese Noten sind der Refrainteil der Melodie. Daher müssen wir es so programmieren, dass es zweimal wiederholt wird und dann den Bildschirm löscht. Nehmen Sie einfach den Wiederholen-10-Block aus der Kategorie Steuerung und platzieren Sie ihn darin. Setzen Sie die Anzahl der Wiederholungen auf 2. Fügen Sie nach der Schleife den Löschcode aus der OLED-Kategorie hinzu.



Wenn Sie die Codes ausführen, stellen Sie die Melodiegeschwindigkeit mit dem Potentiometer auf den Picobricks ein. Steuern Sie das Tempo der Melodie, indem Sie das Potentiometer während oder vor dem Spielen Ihrer Melodie drehen.



[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.6.6. Micropython-Codes des Projekts

```
from machine import Pin,PWM,ADC,I2C #um auf die Hardware picobricks zuzugreifen
from utime import sleep #Zeitbibliothek
from picobricks import SSD1306_I2C
import utime

WIDTH=128
HEIGHT=64
#definiere das Gewicht und die Höhe der picobricks

sda=machine.Pin(4)
scl=machine.Pin(5)
#wir definieren die sda- und scl-Pins für die internen Kommunikationspfade
i2c=machine.I2C(0, sda=sda, scl=scl, freq=2000000)#bestimme die Frequenzwerte
oled=SSD1306_I2C(WIDTH, HEIGHT, i2c)

button= Pin(10,Pin.IN,Pin.PULL_DOWN)
pot=ADC(Pin(26))
buzzer= PWM(Pin(20))
#bestimme unsere Eingangs- und Ausgangspins
```

```
pressed = False
rithm = 0

tones = {
    "A3": 220,
    "D4": 294,
    "E4": 330,
    "F4": 349
}
#define die Töne

mysong = ["A3","E4","E4","E4","E4","E4","E4","F4","E4","D4","F4","E4"]#lass uns die
töne, die für unser Lied benötigt werden, in der richtigen Reihenfolge in eine Sequenz definieren
noteTime = [1,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1]#definiere Wartezeiten zwischen den Tönen
in einem Array
```

```
def playtone(frequency):
    buzzer.duty_u16(6000)
    buzzer.freq(frequency)
#define die Frequenzen des Buzzers
def playsong(pin):
    global pressed
    pressed = True
#define die Töne mit den richtigen Abkühlzeiten
#Und schließlich müssen wir den Pins sagen, wann sie auslösen sollen, und die Funktion, die aufgerufen werden soll, wenn
sie ein Ereignis erkennen:
button.irq(trigger=Pin.IRQ_RISING, handler=playsong)
note_count = 9999
played_time = 0
while True:
    current_time = utime.ticks_ms()
    oled.show()
    oled.text("Drücken Sie die Taste",0,0)

    if (note_count < len(mysong)):
        oled.fill(0)
        oled.text("Dominieren ",30,10)
        oled.text("den ",45,25)
        oled.text("Rhythmus ",35,40)
        rithm=((pot.read_u16()/65535.0)*20) +1
        if (current_time - played_time)/1000.0 >= noteTime[note_count]/rithm:
```

```
    played_time = utime.ticks_ms()
    playtone(tones[mysong[note_count]])
    note_count += 1
else:
    buzzer.duty_u16(0)

if pressed:

    note_count = 0
    pressed = False
```

## 2.6.7. Arduino C-Codes des Projekts

```
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"

int buzzer = 20;
int pot = 26;
int button = 10;
//definiere den Summer, Potentiometer und Knopf

int Re = 294;
int Mi = 330;
int Fa = 349;
int La = 440;
//definiere die Töne

void setup()
{
    Wire.begin();
    oled.init();
    oled.clearDisplay();

    pinMode(buzzer, OUTPUT);
    pinMode(26, INPUT);
    pinMode(button, INPUT);
//bestimme unsere Eingangs- und Ausgangspins
```

```
}

void loop()
{
int rithm = (analogRead(pot))/146;
String char_rithm = String(rithm);
oled.setTextXY(3,4);
oled.putString("Speed: ");
oled.setTextXY(3,10);
oled.putString(char_rithm);

//drucke "Speed: " und den Geschwindigkeitswert auf dem OLED bei x=3 y=4

delay(10);

if (digitalRead(button) == 1){

    oled.clearDisplay();
    oled.setTextXY(3,2);
    oled.putString("Now playing...");
    //drucke "Speed: " und den Geschwindigkeitswert auf dem OLED bei x=3 y=4
    tone(buzzer, La); delay (1000/(rithm+1));
    tone(buzzer, Mi); delay (500/(rithm+1));

    tone(buzzer, Mi); delay (500/(rithm+1));
    tone(buzzer, Re); delay (500/(rithm+1));
    tone(buzzer, Fa); delay (500/(rithm+1));
    tone(buzzer, Mi); delay (1000/(rithm+1));
    //spiele die Noten in der richtigen Reihenfolge und Zeit, wenn der Knopf gedrückt wird

    oled.clearDisplay(); //lösche den Bildschirm
}
noTone(buzzer); //stoppe den Summer
```

---

## GitHub Dominate the Rhythm Projektseite



<http://rbt.ist/rhythm>

## 2.7 Zeige deine Reaktion

Jetzt werden wir ein Spiel vorbereiten, das Aufmerksamkeit und Reflexe entwickelt. Schnell zu bewegen und in der Lage zu sein, lange Zeit Aufmerksamkeit zu schenken, sind wichtige Entwicklungsmerkmale von Kindern. Vorschul- und Grundschulkinder führen Aktivitäten durch, die ihre Aufmerksamkeitsspanne und Reflexe erhöhen, da sie von ihren Eltern und Lehrern gemocht werden. Das elektronische System, das wir vorbereiten werden, wird ein Spiel sein, das die Aufmerksamkeit erhöht und die Reflexe entwickelt. Nach Abschluss des Projekts kannst du mit deinen Freunden konkurrieren. :)

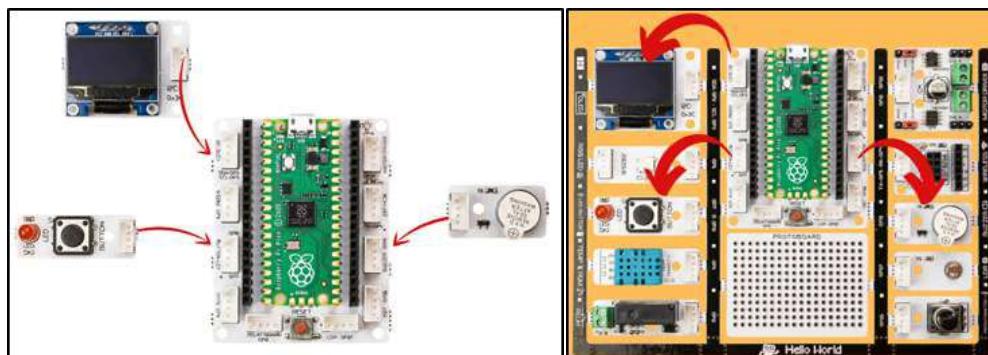
In diesem Projekt wirst du über die Zufälligkeit lernen, die in jeder Programmiersprache verwendet wird. Mit Picobricks werden wir ein elektronisches System mit OLED-Display, Button-LED und Buzzer-Modul entwickeln.

### 2.7.1. Projektdetails und Algorithmus

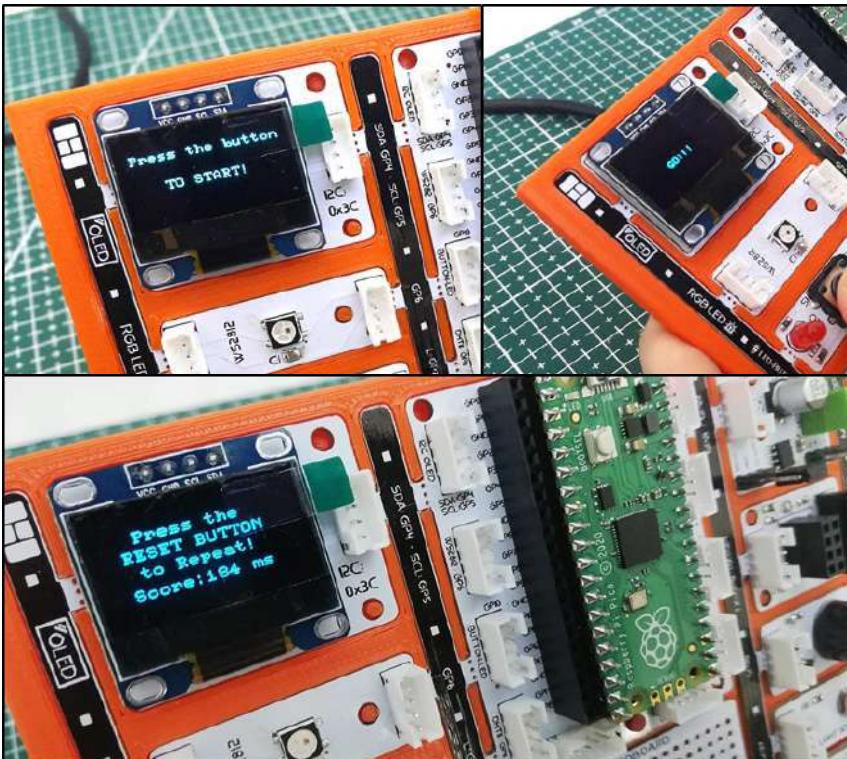
Ein Timer beginnt zu laufen, sobald die Picobricks eingeschaltet werden. Mit diesem Timer können wir 1 Tausendstel einer Sekunde messen. Ein Tausendstel einer Sekunde wird als Millisekunde bezeichnet. Timer werden in vielen elektronischen Systemen im täglichen Leben verwendet. Zeitgesteuertes Licht, Öfen, Bügeleisen, Küchenmaschinen...

Wenn unser Projekt zu arbeiten beginnt, werden wir eine Willkommensnachricht auf dem OLED-Bildschirm anzeigen. Dann werden wir auf dem Bildschirm drucken, was der Benutzer tun muss, um das Spiel zu starten. Um das Spiel zu starten, werden wir den Spieler bitten, sich vorzubereiten, indem er rückwärts von 3 auf dem Bildschirm zählt, nachdem die Taste gedrückt wurde. Nach dem Ende des Countdowns wird die rote LED in einer zufälligen Zeit zwischen 2-10 Sekunden aufleuchten. Wir werden den Timer sofort zurücksetzen, nachdem die rote LED aufleuchtet. Wir werden den Timer messen, sobald die Taste erneut gedrückt wird. Dieser Wert, den wir erhalten, wird in Millisekunden sein. Wir werden diesen Wert auf dem Bildschirm als Reaktionszeit des Spielers anzeigen.

### 2.7.2. Schaltplan



## 2.7.3. Projektbild

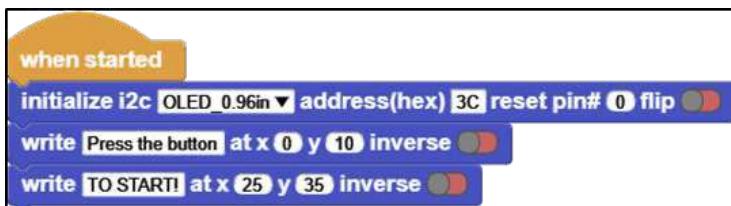


## 2.7.4. Projektvorschlag

Picobricks muss zurückgesetzt werden, um das Spiel neu zu starten. Sie können Ihr Projekt entwickeln, indem Sie den Knopf erneut drücken, um das Spiel wieder zu starten. Sie können auch die höchste Punktzahl und den letzten Punktesammler am Ende des Spiels auf dem Bildschirm anzeigen lassen.

## 2.7.5. Programmierung des Projekts mit Microblocks

Sobald Picobricks nach dem Hinzufügen der OLED-Display-Bibliothek startet, lassen Sie uns Platz für die Anweisungen auf dem Bildschirm schaffen.



Wenn der Benutzer den Knopf von Picobricks drückt, nehmen wir die Schleife for i in 10 aus der Kategorie Steuerung und setzen die Zahl 10 auf 3, damit sie als 3,2,1 herunterzählt. Diese Schleife wird die Variable i bei jedem Durchlauf von 1 auf 3 erhöhen und den Code 3 Mal ausführen. Um den Bildschirm zu löschen und 3...,2...,1... jede Sekunde anzuzeigen, müssen Sie die OLED-Bildschirmblöcke und den Warteblock innerhalb des for-Blocks wie im Bild platzieren.

```
when PicoBricks button pressed
for [i v] in [1 to 3]
  clear
  write [Hello! at x: 25 y: 35 inverse]
  wait [1000] millisecs
```

Aus der Kategorie Daten platzieren Sie den A Schreibblock, löschen Sie den Mikrotext und platzieren Sie den Subtraktionsoperator im Blockfeld. Machen Sie 4 - als Subtraktion.

join micro blocks ➔

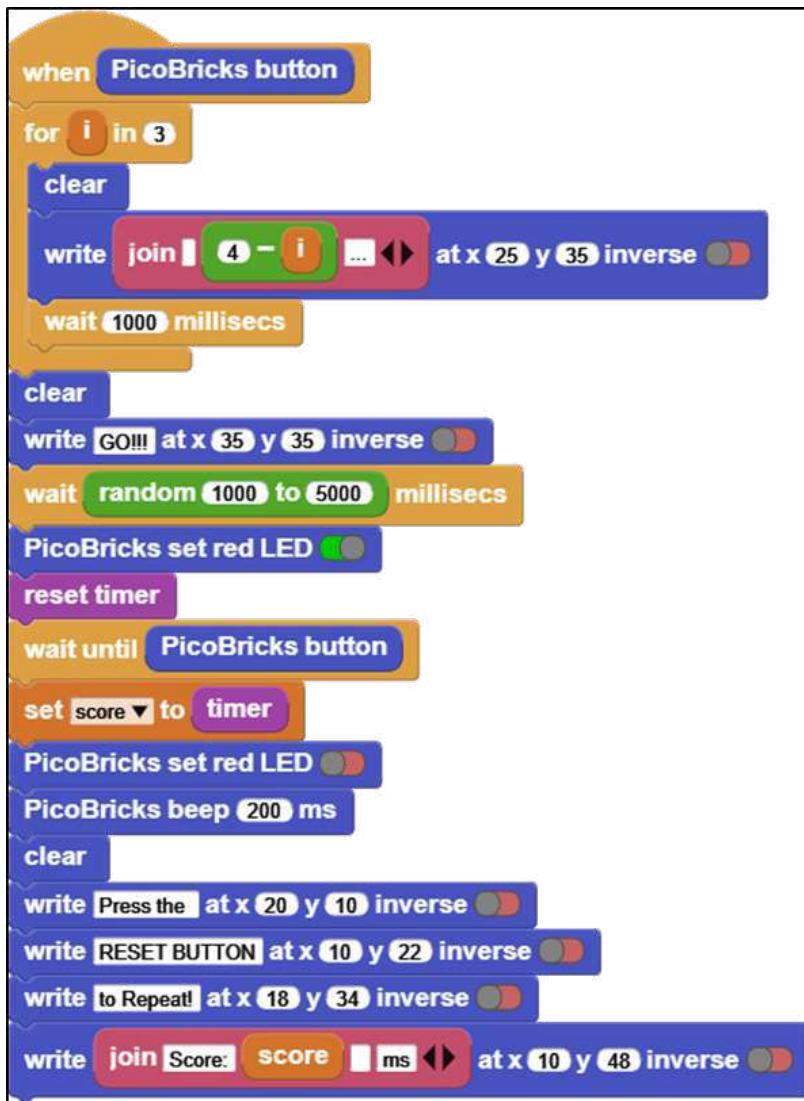
Block im Hallo-Feld im

```
when PicoBricks button pressed
for [i v] in [1 to 3]
  clear
  write [join "4 - " i at x: 25 y: 35 inverse]
  wait [1000] millisecs
```

Direkt unter der for-Schleife fügen wir den Block hinzu, der den Bildschirm löscht und den Spielstart ankündigt. Um die zufällige Wartezeit festzulegen, platzieren Sie den Block zufällig 1 bis 10 im Warteblock. Schreiben Sie 1000 in das erste Feld und 5000 in das zweite Feld. Dieser Befehl wird eine zufällige Zeit im Bereich von 1 Sekunde bis 5 Sekunden erzeugen. Fügen Sie den Code hinzu, der die rote LED sofort danach einschaltet. Wir setzen den Timer mit dem Reset-Timer-Block aus der Kategorie Eingabe zurück. Indem wir den PicoBricks-Knopfblock zum Warte-bis-Block in der Kategorie Steuerung hinzufügen, lassen wir ihn warten, bis der Knopf gedrückt wird.

```
clear
write [GO!!! at x: 35 y: 35 inverse]
wait [random (1000) to (5000)] millisecs
PicoBricks set red LED
reset timer
repeat until [PicoBricks button pressed]
```

Erstellen Sie eine Punktzahlvariable, um den Timerwert zu messen, wenn der Knopf gedrückt wird. Platzieren Sie den Timerwertblock in der Punktzahlvariable unter dem Warte-bis-Block. Schalten Sie dann die rote LED aus und erzeugen Sie einen Piepton von 200 ms. Schließlich reinigen Sie den Bildschirm. Wir drücken den Reset-Knopf auf dem Picoboard, um das Spiel neu zu starten und die Reaktionszeit auf dem Bildschirm anzuzeigen. Viel Spaß beim Spielen.



[Klicken](#) Sie, um auf die Microblocks-Codes des Projekts zuzugreifen.

## 2.7.6. Micropython-Codes des Projekts

```
from machine import Pin, I2C,Timer
from picobricks import SSD1306_I2C
import utime
import urandom
#define die Bibliothek
WIDTH=128
HEIGHT=64
#define die Breite und Höhe Werte
sda=machine.Pin(4)
scl=machine.Pin(5)
i2c=machine.I2C(0,sda=sda, scl=scl, freq=2000000)
oled= SSD1306_I2C(WIDTH, HEIGHT, i2c)
```

```

button = Pin(10,Pin.IN,Pin.PULL_DOWN)
led=Pin(7,Pin.OUT)
#define unsere Eingangs- und Ausgangspins
while True:
    led.value(0)
    oled.fill(0)
    oled.text("drücke die Taste",0,10)
    oled.text("UM ZU STARTEN!",25,25)
    oled.show()
#drücke "Drücke die Taste" und "UM ZU STARTEN!" auf dem OLED-Bildschirm
    while button.value()==0:
        pass
        oled.fill(0)
        oled.text("Warten auf LED", 15, 30)
        oled.show()
# Schreibe "Warten auf LED" auf den Bildschirm, wenn die Taste gedrückt wird
        utime.sleep(urandom.uniform(1, 5))
        led.value(1)
        timer_start = utime.ticks_ms()
# Warte eine zufällige Sekunde und schalte die LED ein
        while button.value() == 0:
            pass
            timer_reaction=utime.ticks_diff(utime.ticks_ms(), timer_start)
            pressed=True
            oled.fill(0)
            oled.text("Ihre Zeit", 25, 25)
            oled.text(str(timer_reaction), 50, 50)
            oled.show()
            led.value(0)
            utime.sleep(1.5)
# Drucke die Punktzahl und "Ihre Zeit" auf den Bildschirm, wenn die Taste gedrückt wird.

```

## 2.7.7. Arduino C-Codes des Projekts

```

#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"
// Definiere die Bibliothek

int buzzer = 20;
int button = 10;

```

```
int led = 7;
int La = 440;

int old_time = 0;
int now_time = 0;
int score = 0;
String string_score;
//definiere unsere Integer- und String-Variablen

void setup(){
// füge deinen Setup-Code hier ein, um ihn einmal auszuführen:
Wire.begin();
oled.init();
oled.clearDisplay();

pinMode(led,OUTPUT);
pinMode(buzzer,OUTPUT);
pinMode(button,INPUT);
Serial.begin(9600);
//definiere die Eingangs- und Ausgangspins
}

void loop(){
// put your main code here, to run repeatedly:
oled.setTextXY(3,0);
oled.putString("Drücken Sie die Taste");
oled.setTextXY(5,4);
oled.putString("UM ZU STARTEN");

if (digitalRead(button) == 1){

for (int i=3;i>0;i--){

String string_i = String(i);
oled.clearDisplay();
oled.setTextXY(4,8);
oled.putString(string_i);
delay(1000);

}
//zähle rückwärts von drei
```

```
oled.clearDisplay();
oled.setTextXY(4,6);
oled.putString("LOS!!!");
//drucke "LOS!!" auf dem OLED bei x=4 y=6

int random_wait = random(1000, 5000);
delay(random_wait);
//warte eine zufällige Sekunde zwischen 1 und 5

digitalWrite(led, HIGH);
old_time=millis();
//schalte LED ein

while(!(digitalRead(button) == 1)){

now_time=millis();

score = now_time-old_time;
string_score = String(score);
//score als String beim Drücken des Buttons speichern
}

digitalWrite(led, HIGH);
tone(buzzer, La);
delay (200);
noTone(buzzer);
//LED und Buzzer einschalten

oled.clearDisplay();
oled.setTextXY(1,4);
oled.putString("Drücken Sie die");
// "Drücken Sie die" auf dem OLED bei x=1 Y=4 drucken
oled.setTextXY(2,3);
oled.putString("RESET-TASTE");
// "RESET BUTTON" auf dem OLED bei X=2 Y=3 drucken
oled.setTextXY(3,3);
oled.putString("um zu wiederholen!");
// "Um zu wiederholen!" auf dem OLED bei X=3 Y=3 drucken
oled.setTextXY(6,3);
oled.putString("Punktestand: ");
oled.setTextXY(6,9);
```

```
oled.putString(string_score);
oled.setTextXY(6,13);
oled.putString(" ms");
Serial.println(score);

//Punktestandwert auf dem Bildschirm drucken

delay(10000);
oled.clearDisplay();
//warte zehn Sekunden und lösche den Bildschirm
}

}
```

GitHub Zeige Deine Reaktion Projektseite



<http://rbt.ist/reaction>

## 2.8. Mein Timer

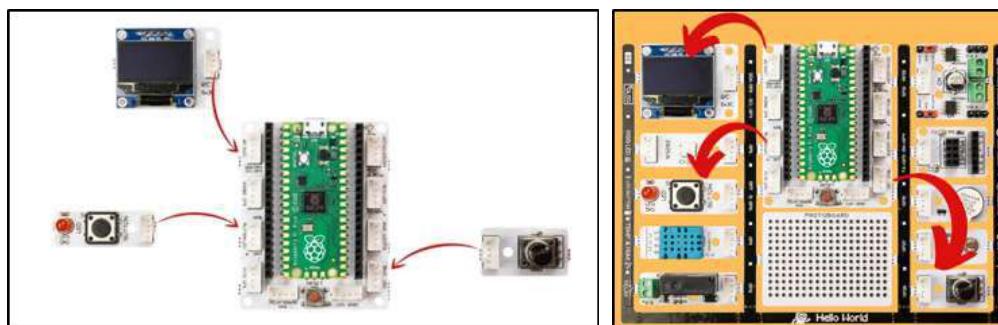
Die Zeitmessung ist eine einfache, aber wichtige Aufgabe, die wir in unserem täglichen Leben tun, ohne es zu merken. Ein Chirurg bei einer Operation, ein Geschäftsmann, der versucht, ein Meeting zu erreichen, ein Sportler, der versucht zu gewinnen, ein Student, der versucht, eine Prüfung oder ein Schachspiel zu beenden... Smarte Armbanduhren, Telefone und sogar professionelle Chronometer werden verwendet, um die Zeit zu messen. Zeit ist eine Variable, die in elektronischen Systemen sehr genau verwendet werden sollte. Zum Beispiel eine Waschmaschine; wie lange sich die Trommel im Uhrzeigersinn dreht, wie viel gegen den Uhrzeigersinn, wie viele Sekunden Wasser fließen muss, um das Waschmittel aufzulösen, sind Aufgaben, die durch Zeitmessung erledigt werden. Um Projekte zu entwickeln, bei denen Zeit von entscheidender Bedeutung ist, müssen Sie wissen, wie man sie nutzt.

In diesem Projekt werden Sie Ihr eigenes Zeitmessgerät mit Picobricks, OLED-Display, Tasten- und Potentiometer-Modulen erstellen. Ein Timer...

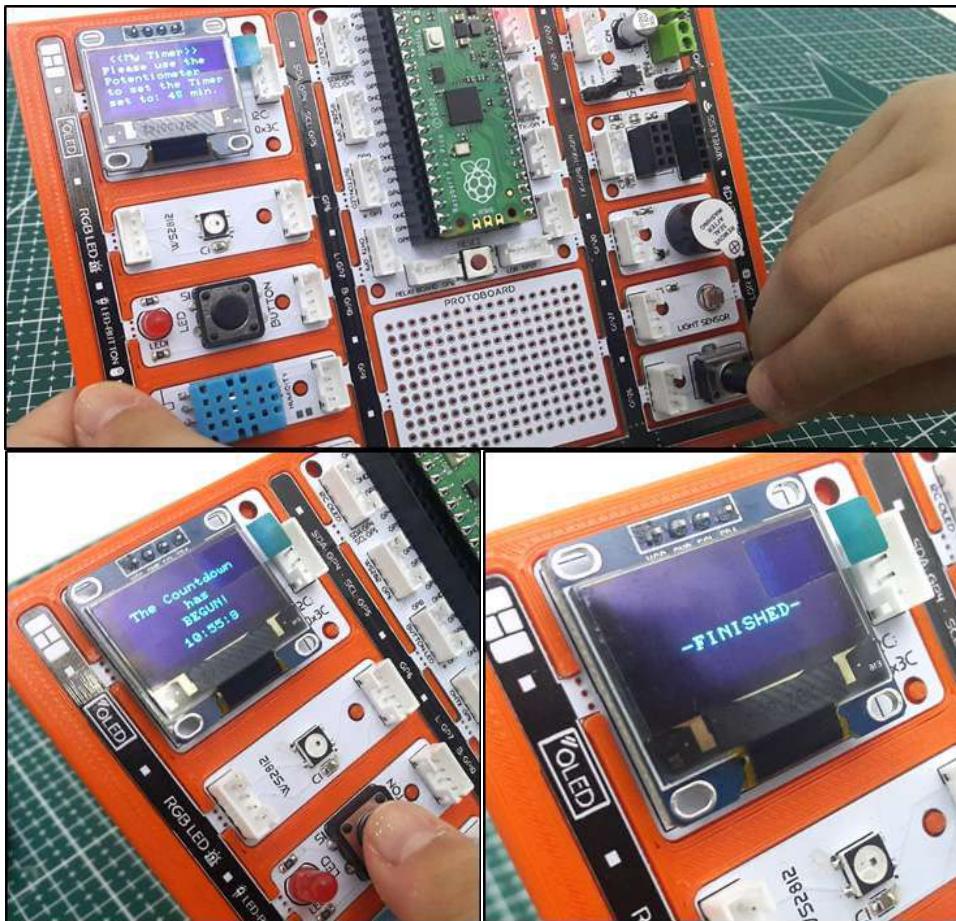
### 2.8.1. Projektdetails und Algorithmus

Wenn Picobricks startet, lassen Sie uns eine Aussage auf dem Bildschirm anzeigen, die das Projekt vorstellt und Anweisungen enthält. Während der Benutzer den Potentiometer dreht, wird er eine Zeit im Bereich von 0-60 Minuten einstellen. Wenn der Benutzer die Taste von Picobricks drückt, nachdem er die Zeit mit dem Potentiometer festgelegt hat, beginnt der Countdown in Minuten und Sekunden auf dem Bildschirm. Wenn die Taste gedrückt wird, während die Zeit rückwärts läuft, stoppt der Timer und zeigt die verbleibende Zeit auf dem Bildschirm an. Wenn der Minuten-, Sekunden- und Sekundenwert null erreicht, ohne die Taste zu drücken, wird eine Benachrichtigung angezeigt, die besagt, dass die Zeit abgelaufen ist, und das Programm wird gestoppt.

### 2.8.2. Verdrahtungsdiagramm



### 2.8.3. Bauphasen des Projekts

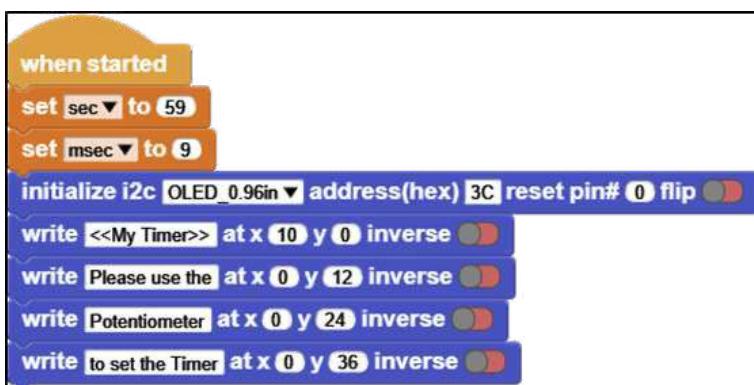


### 2.8.4. Projektvorschlag

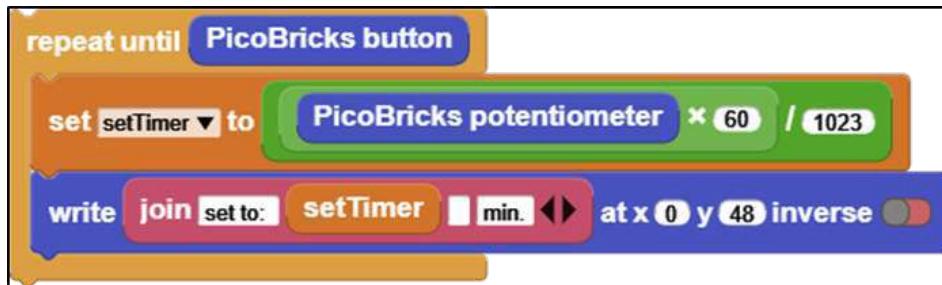
Sie können einen Piepton zu Beginn des Timers hinzufügen. Wenn die Zeit zurückgesetzt wird, können Sie different und hohe Warnsignale mit dem Summer geben und aus der Ferne ankündigen, dass die Zeit abgelaufen ist.

### 2.8.5. Programmierung des Projekts mit MicroBlocks

Erstellen Sie zunächst die Variablen, die im Projekt verwendet werden sollen. setTimer, min, sec, msec setzen die Werte der Variablen sec und msec auf 59 bzw. 9. Fügen Sie dann die Codes für den Splashscreen ein.



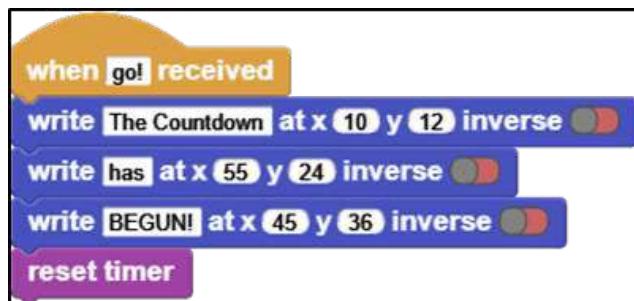
Weisen wir den angepassten Minutenwert der Variable setTimer zu, indem wir den Wert, der vom Potentiometer gelesen wird, mit 60 multiplizieren und durch 1023 teilen, bis die Taste gedrückt wird, und drucken diese Variable auf dem Bildschirm aus.



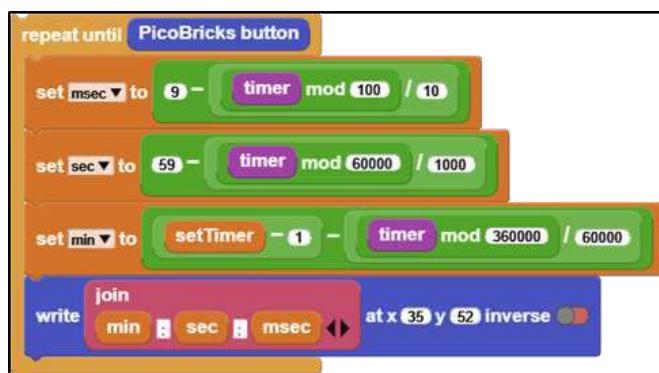
Das Drücken der Taste nach dem Wiederhole-bis-Block löscht den Bildschirm und wartet einen kurzen Moment, dann LOS! Lassen Sie uns senden.



LOS! Lassen Sie uns den Ausdruck platzieren, der besagt, dass der Countdown gestartet ist, wenn die Übertragung empfangen wird. Dann setzen wir den Timer zurück.



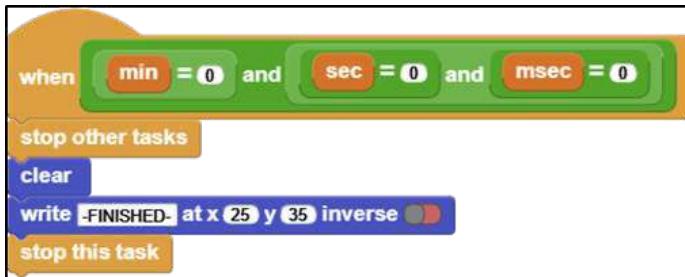
Lassen Sie uns die Werte für msec, sec und min festlegen, indem wir den Timer messen, bis die Taste zum zweiten Mal gedrückt wird. msec steht für Millisekunden (0-9), sec für Sekunden (0-59), min (0-59) Minuten. Dann zeigen wir die Variablen auf dem OLED-Bildschirm an.



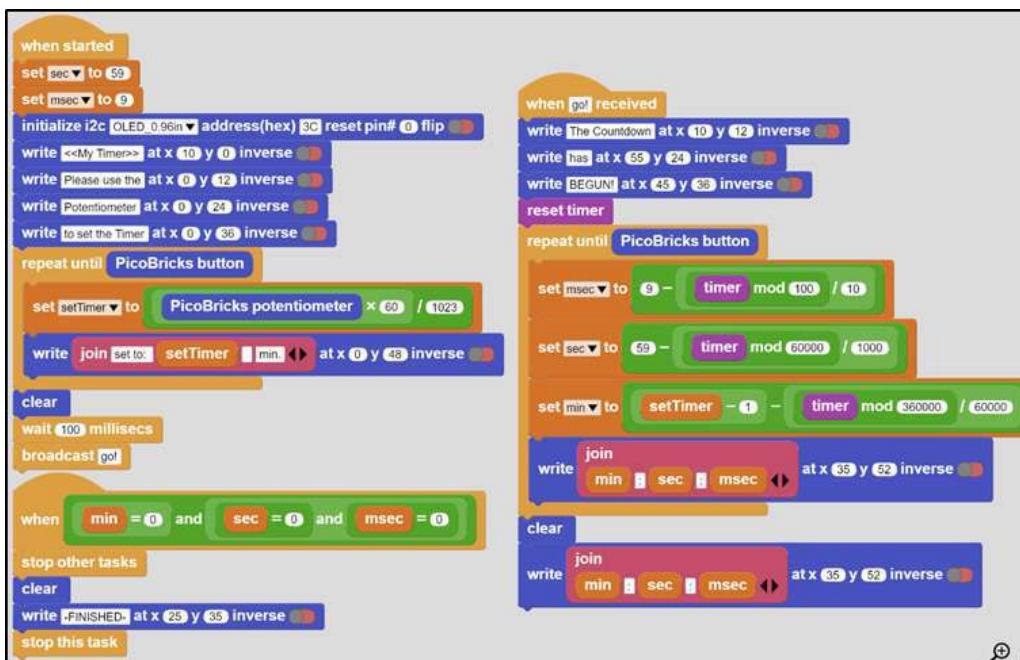
Wenn der Timer vor Ablauf der Zeit beendet werden soll, lassen Sie die verbleibende Zeit auf dem Bildschirm erscheinen, wenn die Taste gedrückt wird.



Überprüfen wir schließlich ständig, ob die Variablen sec, min und msec 0 sind. Wenn alle drei dieser Variablen null sind, bedeutet das, dass die Zeit abgelaufen ist. Lassen Sie uns also alle Codes stoppen, nachdem wir die anderen Codeblöcke gestoppt haben, und die Endnachricht auf dem Bildschirm ausgeben.



Der endgültige Code sollte wie folgt aussehen. Sobald Sie die Starttaste drücken, wird Ihr Projekt ausgeführt.



[Klicken](#) Sie, um auf die Microblocks-Codes des Projekts zuzugreifen.

## 2.8.6. MicroPython-Codes des Projekts

```
from machine import Pin, I2C, ADC, Timer #um auf die Hardware picobricks zuzugreifen
from picobricks import SSD1306_I2C #oled-Bibliothek
import utime #Zeitbibliothek
```

WIDTH = 128

```
HEIGHT = 64
#definieren Sie die Breiten- und Höhenwerte

sda=machine.Pin(4)
scl=machine.Pin(5)
#wir definieren die sda- und scl-Pins für die internen Kommunikationspfade
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)#bestimmen Sie die Frequenzwerte

oled = SSD1306_I2C(128, 64, i2c)
pot = ADC(Pin(26))
button = Pin(10,Pin.IN,Pin.PULL_DOWN)
#bestimmen Sie unsere Eingangs- und Ausgangspins

oled.fill(0)
oled.show()
#Auf OLED anzeigen

time=Timer()
time2=Timer()
time3=Timer()
#define timers

def minute(timer):
    global setTimer
    setTimer -=1

def second(timer):
    global sec
    sec-=1
    if sec== -1:
        sec=59

def msecnd(timer):
    global msec
    msec-=1
    if msec== -1:
        msec=99

#Wir bestimmen die Inkremente der Minuten-, Sekunden- und Millisekundenwerte.
sec=59
msec=99
```

```
global setTimer
```

```
while button.value()==0:  
setTimer=int((pot.read_u16()*60)/65536)+1  
oled.text("Timer einstellen:" + str(setTimer) + " min",0,12)  
oled.show()  
utime.sleep(0.1)  
oled.fill(0)  
oled.show()
```

#Wenn der Knopf nicht gedrückt wird, wird der Wert, der durch den Potentiometer bestimmt wird, auf dem OLED-Bildschirm ausgegeben.

```
setTimer-=1
```

```
time.init(mode=Timer.PERIODIC,period=60000,callback=minute)  
time2.init(mode=Timer.PERIODIC,period=1000,callback=second)  
time3.init(mode=Timer.PERIODIC,period=10,callback=msecond)  
#Wir bestimmen die Perioden von Minuten, Sekunden und Millisekunden.  
utime.sleep(0.2)#warte 0.2 Sekunden
```

```
while button.value()==0:  
    oled.text("min:" + str(setTimer),50,10)  
    oled.text("sec:" + str(sec),50,20)  
    oled.text("ms:" + str(msec),50,30)  
    oled.show()  
    utime.sleep(0.008)  
    oled.fill(0)  
    oled.show()  
if(setTimer==0 and sec==0 and msec==99):  
    utime.sleep(0.1)  
    msec=0  
    break;
```

#Wenn der Knopf gedrückt wird, werden die Min-Sek-Ms-Werte auf dem OLED-Bildschirm an den bestimmten x- und y-Koordinaten ausgegeben.

```
oled.text(str(setTimer),60,10)  
oled.text(str(sec),60,20)  
oled.text(str(msec),60,30)  
oled.text("Die Zeit ist abgelaufen!",10,48)  
oled.show()
```

#Gibt die Minuten, Sekunden, Millisekunden und die Werte „Die Zeit ist abgelaufen“ an den X- und Y-Koordinaten aus

## 2.8.7. Arduino C-Codes des Projekts

```
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"

// Definiere die Bibliothek
int minute;
int second = 59;
int milisecond = 9;
int setTimer;
//Variablen definieren
void setup() {
    // füge deinen Setup-Code hier ein, um ihn einmal auszuführen:
    pinMode(10,INPUT);
    pinMode(26,INPUT);

    Wire.begin();
    oled.init();
    oled.clearDisplay();
    //definiere die Ein- und Ausgangspins und das OLED-Display

}

void loop() {
    // put your main code here, to run repeatedly:
    oled.setTextXY(1,2);
    oled.putString("<<Mein Timer>>");
    oled.setTextXY(3,1);
    oled.putString("Bitte verwenden Sie das");
    oled.setTextXY(4,1);
    oled.putString("Potentiometer");
    oled.setTextXY(5,0);
    oled.putString("um den Timer einzustellen");

    //drucke "<<Mein Timer>>", "Bitte verwenden Sie das", "Potentiometer" und "um den Timer einzustellen."
    //auf die x- und y-Koordinaten, die auf dem OLED-Bildschirm bestimmt sind.
```

```
delay(3000);
oled.clearDisplay();
//wir haben drei Sekunden gewartet und gelöscht

while(!(digitalRead(10) == 1))
{
    setTimer = (analogRead(26)*60)/1023;
    oled.setTextXY(3,1);
    oled.putString("set to:");
    oled.setTextXY(3,8);
    oled.putString(String(setTimer));
    oled.setTextXY(3,11);
    oled.putString("min.");
//Bestimmen Sie den Minimalwert mit dem Potentiometer und drucken Sie ihn auf dem Bildschirm aus, bis die Taste gedrückt wird.
}
oled.clearDisplay();
oled.setTextXY(1,1);
oled.putString("Der Countdown");
oled.setTextXY(2,3);
oled.putString("hat begonnen!");
//Drucken Sie "Der Countdown" und "hat begonnen!" an die auf dem OLED-Bildschirm bestimmten x- und y-Koordinaten
while(!(digitalRead(10) == 1))
{
milisekunde = 9 - (millis() % 100) / 10;
sekunde = 59 - (millis() % 60000) / 1000;
minute = (setTimer - 1) - ((millis() % 360000) / 60000);

oled.setTextXY(5,3);
oled.putString(String(minute));
oled.setTextXY(5,8);
oled.putString(String(second));
oled.setTextXY(5,13);
oled.putString(String(milisecond));
oled.setTextXY(5,6);
oled.putString(":");
oled.setTextXY(5,11);
oled.putString(":");
//Wenn der Knopf gedrückt wird, verringern Sie die Werte für Millisekunden, Sekunden und Minuten
```

und schreiben Sie auf den Bildschirm.

```
}

oled.setTextXY(5,3);
oled.putString(String(minute));
oled.setTextXY(5,8);
oled.putString(String(second));
oled.setTextXY(5,13);
oled.putString(String(milisecond));
oled.setTextXY(5,6);
oled.putString(":");
oled.setTextXY(5,11);
oled.putString(":");
delay(10000);

//Drucken Sie die Werte für Minuten, Sekunden und Millisekunden an den x- und y-Koordinaten
//, die auf dem OLED-Bildschirm bestimmt sind

if (minute==0 & second==0 & milisecond==0){

    oled.setTextXY(5,3);
    oled.putString(String(minute));
    oled.setTextXY(5,8);
    oled.putString(String(second));
    oled.setTextXY(5,13);
    oled.putString(String(milisecond));
    oled.setTextXY(5,6);
    oled.putString(":");
    oled.setTextXY(5,11);
    oled.putString(":");
    oled.putString("-fertig-");
    oled.setTextXY(7,5);
    delay(10000);

    //Geben Sie die Minuten-, Sekunden- und Millisekundenwerte sowie „-fertig-“ an den auf dem
    //OLED-Bildschirm bestimmten x- und y-Koordinaten aus.

}

}
```

---

## GitHub Meine Timer-Projektseite



<http://rbt.ist/mytimer>

## 2.9. Wecker

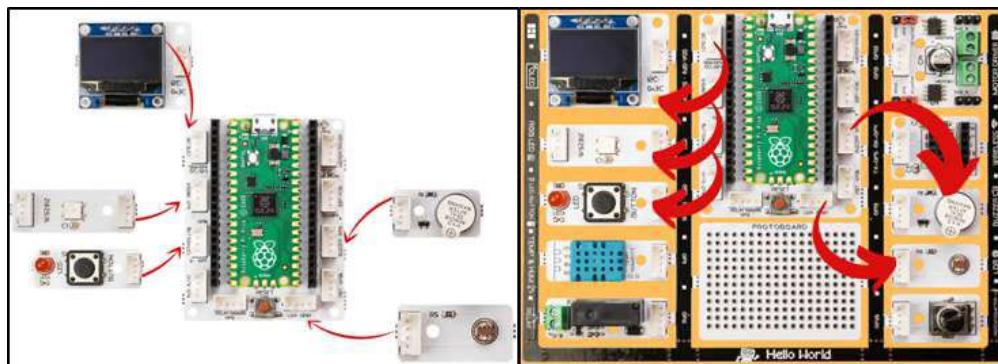
Die globale Erwärmung beeinflusst das Klima unserer Welt jeden Tag schlimmer. Die Länder ergreifen viele Vorsichtsmaßnahmen und unterzeichnen Vereinbarungen, um die Auswirkungen der globalen Erwärmung zu verringern. Die Nutzung erneuerbarer Energiequellen und der effiziente Einsatz von Energie ist ein Thema, das überall Aufmerksamkeit benötigt, von Fabriken bis zu unseren Räumen. Viele Gründe, wie das eingeschaltete Straßen- und Parklicht in Städten aufgrund menschlicher Fehler und die Verwendung von energieintensiven Beleuchtungssystemen, verringern die Energieeffizienz. Viele elektronische und digitale Systeme werden von Ingenieuren entwickelt und programmiert, um die Licht-, Temperatur- und Feuchtigkeitswerte der Umgebung zu messen und sicherzustellen, dass sie nur bei Bedarf und in den richtigen Mengen verwendet werden.

In diesem Projekt werden wir einen Timer-Alarm erstellen, der sich mithilfe des Lichtsensors in Picobricks an die Tageslichtverhältnisse anpasst.

### 2.9.1. Projektdetails und Algorithmus

In diesem Projekt werden wir eine einfache Alarmanwendung erstellen. Das Alarmsystem, das wir entwerfen werden, ist so konzipiert, dass es morgens automatisch ertönt. Dazu werden wir im Projekt einen LDR-Sensor verwenden. Nachts wird der OLED-Bildschirm dem Benutzer eine Gute-Nacht-Nachricht anzeigen, und am Morgen ertönt ein Alarm mit einem Summergeräusch, eine Guten-Morgen-Nachricht wird auf dem Bildschirm angezeigt, und die RGB-LED wird weiß leuchten, um eine Lichtbenachrichtigung zu geben. Der Benutzer muss die Taste von Picobricks drücken, um den Alarm zu stoppen. Nach diesen Prozessen, die fortgesetzt werden, bis der Alarm gestoppt wird, werden beim Drücken der Taste der Summer und die RGB-LED ausgeschaltet, und eine Gute-Tag-Nachricht wird auf dem OLED-Bildschirm angezeigt.

### 2.9.2. Verdrahtungsdiagramm



## 2.9.3. Projektbild



## 2.9.4. Projektvorschlag

Sie können das Projekt verbessern, indem Sie eine Melodie als Alarmton anstelle eines Pieptons hinzufügen. Oder anstelle eines Alarms, der gemäß dem Tageslicht mit dem LDR-Sensor eingestellt ist, können Sie einen Alarm entwickeln, der zu einer bestimmten Zeit ertönt, wobei die Zeitinformationen über die Taste und den OLED-Bildschirm eingestellt werden.

## 2.9.5. Codierung des Projekts mit MicroBlocks

Fügen Sie zunächst die OLED-Grafikbibliothek zum Programm hinzu und fügen Sie die Initialisierung von I2C und die Schreibblöcke aus der OLED-Kategorie unter dem Block „Wenn gestartet“ hinzu, damit „Gute Nacht“ auf dem Bildschirm angezeigt wird, wenn Pico startet. Wenn Sie dieses Projekt Tagsüber durchführen, müssen Sie den LDR-Sensor mit Ihrer Hand abdecken, um das Programm zu testen. Aus diesem Grund warten Sie nach dem Schreibblock 2 Sekunden mit dem Warteblock und decken den LDR-Sensor innerhalb dieser 2 Sekunden mit Ihrer Hand ab.



Im autonomen Beleuchtungsprojekt lesen wir die LDR-Sensordaten mit dem say-Block und sahen Werte über 90, wenn die Umgebung hell war, und unter 90, wenn es dunkel war. In diesem Projekt können wir den Wert 90 verwenden, um den Unterschied zwischen Tag und Nacht zu bestimmen. Mit dem say-Block können Sie den LDR-Sensorwert in Ihrer Umgebung anzeigen und den Wert ändern, bei dem der Alarm ertönt.

Wenn der Wert des LDR-Sensors größer als 90 ist, müssen wir den when Block verwenden und die Bedingung für die Aktivierung des Alarms festlegen. Unter dem when-Block sollten wir den repeat until-Block verwenden, damit der Alarm weiter ertönen kann, bis der Benutzer die Taste drückt, und wir müssen den Block des Drückens der Taste von PicoBricks als Bedingung festlegen. Sie sollten die Codes für die RGB-LED, den Summer und den OLED-Bildschirm innerhalb des repeat until-Blocks platzieren, der die Blöcke darin ausführt, bis die festgelegte Bedingung erfüllt ist. Sie können die Bildschirmtexte, die RGB-LED-Farbe und das Abspielen des Summers ändern.

Zeit.

```
when [90 < PicoBricks light sensor]
  initialize i2c [OLED_0.96in v] address(hex) [3C] reset pin# [0] flip [red]
  repeat until [PicoBricks button]
    write [Good morning] at x [15] y [32] inverse [red]
    PicoBricks set RGB LED color [white]
    PicoBricks beep [500] ms
```

Die bisher geschriebenen Codes sind die Codes, die die Aktionen ausführen, die beim Systemstart und am Morgen durchgeführt werden sollen. Jetzt lassen Sie uns die notwendigen Codes für die Aktionen schreiben, die ausgeführt werden sollen, nachdem der Knopf gedrückt wurde, während der Alarm ertönt.

```
when [93 < PicoBricks light sensor]
  initialize i2c [OLED_0.96in v] address(hex) [3C] reset pin# [0] flip [red]
  repeat until [PicoBricks button]
    write [Good morning] at x [15] y [32] inverse [red]
    PicoBricks set RGB LED color [white]
    PicoBricks beep [500] ms
    write [have a nice day] at x [0] y [32] inverse [red]
    PicoBricks turn off RGB LED
  stop this task
```

Wenn der Knopf gedrückt wird, während der Alarm ertönt, wird das Programm aus dem Wiederhole-bis-Block herausgehen und die Codes in der nächsten Zeile ausführen. Aus diesem Grund sollten Sie die RGB-LED ausschalten und die Bildschirmtextblöcke unter dem Wiederhole-bis-Block schreiben.

Alarm Clock

```
when started
  initialize i2c [OLED_0.96in v] address(hex) [3C] reset pin# [0] flip [red]
  write [Good night] at x [25] y [32] inverse [red]
  wait [2000] millisecs

when [93 < PicoBricks light sensor]
  initialize i2c [OLED_0.96in v] address(hex) [3C] reset pin# [0] flip [red]
  repeat until [PicoBricks button]
    write [Good morning] at x [15] y [32] inverse [red]
    PicoBricks set RGB LED color [white]
    PicoBricks beep [500] ms
    write [have a nice day] at x [0] y [32] inverse [red]
    PicoBricks turn off RGB LED
  stop this task
```

Wenn Sie alle Codes eingeben und die Starttaste drücken, wird „Gute Nacht“ auf dem Bildschirm angezeigt, und wenn es dunkel ist, wartet das Programm, bis es hell wird. Wenn Sie das Projekt in einer hellen Umgebung testen, decken Sie den LDR-Sensor innerhalb von 2 Sekunden nach dem Drücken der Starttaste mit Ihrer Hand ab. Dann, wenn die Umgebung beleuchtet ist, werden Sie sehen, dass der Alarm mit dem OLED-Bildschirm, dem Summer und der RGB-LED funktioniert. Sie können den Alarm durch Drücken der Taste stoppen.

[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.9.6. Micropython-Codes des Projekts

```
from machine import Pin, I2C, ADC, PWM#um auf die Hardware des Pico zuzugreifen
from picobricks import SSD1306_I2C#OLED-Bildschirmbibliothek
import utime
from picobricks import WS2812#ws8212-Bibliothek

#OLED-Bildschirm-Einstellungen
WIDTH = 128
HEIGHT = 64

sda=machine.Pin(4)
scl=machine.Pin(5)
#digitalen Pin 4 und 5 als AUSGANG für die OLED-Kommunikation initialisieren

i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)
neo = WS2812(pin_num=6, num_leds=1, brightness=0.3)#digitalen Pin 6 als
AUSGANG für NeoPixel initialisieren

oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
ldr = ADC(Pin(27))#digitalen Pin 6 als AUSGANG für NeoPixel initialisieren
button = Pin(10,Pin.IN,Pin.PULL_DOWN)#digitalen Pin 10 als EINGANG für
Taste initialisieren
buzzer = PWM(Pin(20, Pin.OUT))#digitalen Pin 20 als AUSGANG für den Summer initialisieren
buzzer.freq(1000)

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
#RGB Schwarz- und Weißfarbcodes
oled.fill(0)
oled.show()
```

```
neo_pixels_fill(BLACK)
neo_pixels_show()

if ldr.read_u16()<4000:
    wakeup = True
else:
    wakeup = False

while True:
    while wakeup==False:
        oled.fill(0)
        oled.show()
        oled.text("Gute Nacht",25,32)
        oled.show()
#Auf OLED anzeigen und "Gute Nacht" drucken
        utime.sleep(1)
        if ldr.read_u16()<4000:
während button.value()==0:
            oled.fill(0)
oled.show()
            oled.text("Guten Morgen",15,32)
            oled.show()
#Drucken Sie die Minuten, Sekunden, Millisekunden und die Werte von "Guten Morgen" an den
X- und Y-Koordinaten, die auf dem OLED-Bildschirm bestimmt sind.
neo_pixels_fill(WEISS)
    neo_pixels_show()
    buzzer.duty_u16(6000)
    utime.sleep(1)
#Warten Sie eine Sekunde
    buzzer.duty_u16(0)
    utime.sleep(0.5)
    #Warten Sie eine halbe Sekunde
    wakeup=True
neo_pixels_fill(SCHWARZ)
    neo_pixels_show()
    oled.fill(0)
    oled.show()
oled.text("Einen schönen Tag noch!",0,32)
#Drucken Sie die Minuten, Sekunden, Millisekunden und die Werte von "Einen schönen Tag noch!" an den X
und Y-Koordinaten, die auf dem OLED-Bildschirm bestimmt sind.
oled.show()
if ldr.read_u16()>40000:
```

```
wakeup= False
```

```
utime.sleep(1)
```

```
#Warten Sie eine Sekunde
```

## 2.9.7. Arduino C Codes des Projekts

```
#include <Adafruit_NeoPixel.h>
#ifndef __AVR__
#include <avr/power.h>
#endif
#define PIN      6

#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"
int button;

void setup() {
  Wire.begin();
  oled.init();
  oled.clearDisplay();

#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  pinMode(10,EINGANG);
  pinMode(27,EINGANG);
  pinMode(20,AUSGANG);

  pixels.begin();
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.show();
}

void loop() {

oled.setTextXY(4,3);
oled.putString("Gute Nacht");

if (analogRead(27)<200){
```

```
while(!(button == 1)){

    button=digitalRead(10);

    oled.setTextXY(4,2);
    oled.putString("Guten Morgen");
    pixels.setPixelColor(0, pixels.Color(255, 255, 255));
    pixels.show();
    tone(20,494);
}

oled.clearDisplay();
oled.setTextXY(4,1);
oled.putString("Einen schönen Tag");
noTone(20);
pixels.setPixelColor(0, pixels.Color(0, 0, 0));
pixels.show();
delay(10000);

}

}
```

## GitHub Alarm Clock Projektseite



<http://rbt.ist/alarm>

## 2.10. Kenne deine Farbe

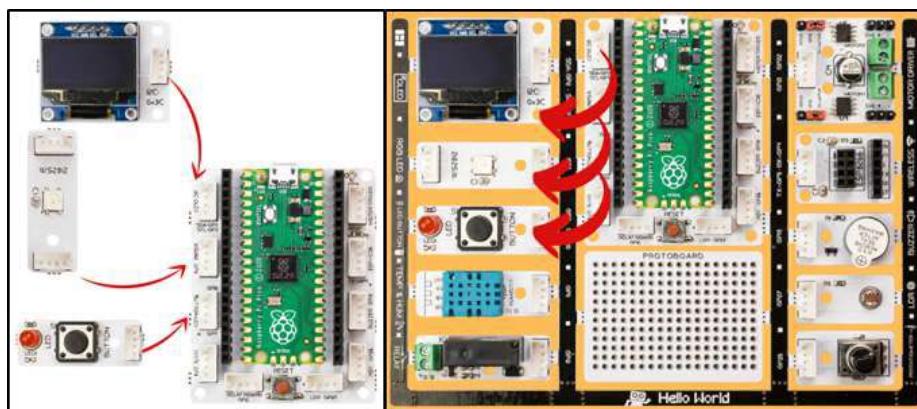
LEDs werden häufig in elektronischen Systemen verwendet. Jeder Knopf kann kleine LEDs neben jeder Option haben. Indem man eine einzelne LED in verschiedenen Farben leuchten lässt, ist es möglich, die Arbeit von mehr als einer LED mit einer einzigen LED zu erledigen. LEDs, die in dieser Art arbeiten, werden RGB-LEDs genannt. Sie verdanken ihren Namen den Anfangsbuchstaben der Farbnamen Rot, Grün, Blau. Ein weiterer Vorteil dieser LED ist, dass sie in Mischungen der 3 Primärfarben leuchten kann. Lila, Türkis, Orange...

In diesem Projekt wirst du über die Zufälligkeit lernen, die in jeder Programmiersprache verwendet wird. Wir werden ein unterhaltsames Spiel mit der RGB-LED, dem OLED-Bildschirm und dem Tastenmodul von Picobricks vorbereiten.

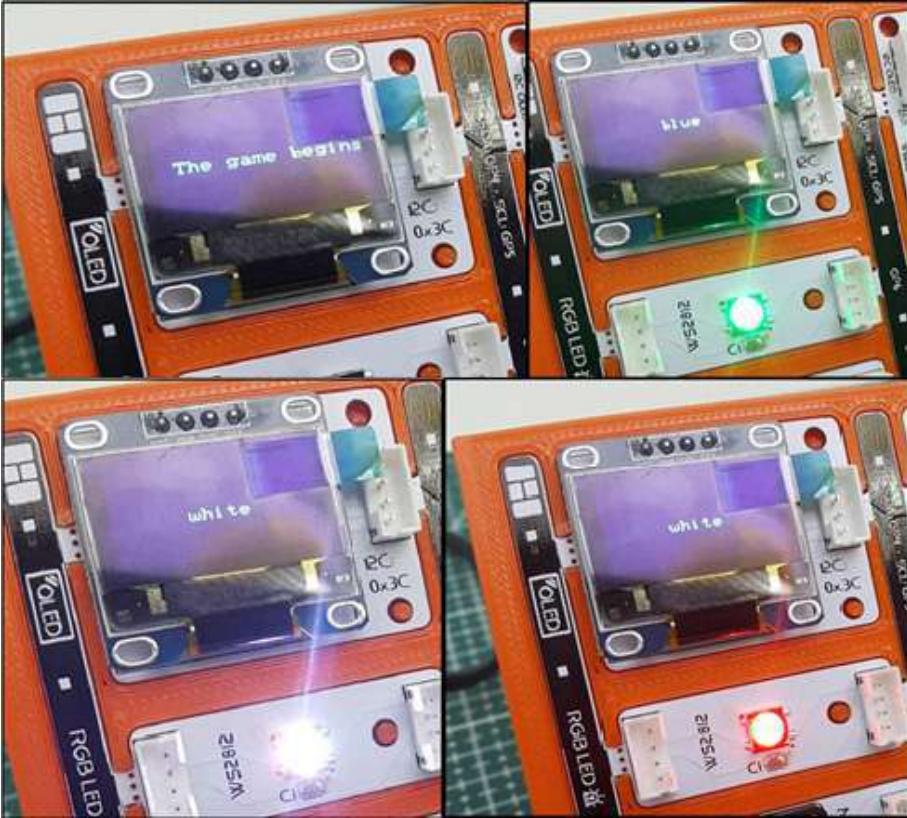
### 2.10.1. Projektdetails und Algorithmus

Das Spiel, das wir im Projekt bauen werden, basiert darauf, dass der Benutzer die Farben richtig oder falsch erkennt. Eine der Farben Rot, Grün, Blau und Weiß wird zufällig auf der RGB-LED von Picobricks leuchten, und der Name einer dieser vier Farben wird gleichzeitig zufällig auf dem OLED-Bildschirm angezeigt. Der Benutzer muss innerhalb von 1,5 Sekunden den Knopf von Picobricks drücken, um sein Antwortrecht zu nutzen. Das Spiel wird 10 Mal wiederholt, jede Wiederholung erhält 10 Punkte, wenn der Benutzer den Knopf drückt, wenn die Farben übereinstimmen, oder wenn der Benutzer den Knopf nicht drückt, wenn sie nicht übereinstimmen. Wenn der Benutzer den Knopf drückt, obwohl die Farben nicht übereinstimmen, verliert er 10 Punkte. Nach zehn Wiederholungen wird die Punktzahl des Benutzers auf dem OLED-Bildschirm angezeigt. Wenn der Benutzer möchte, kann er sein Antwortrecht nicht nutzen, indem er den Knopf nicht drückt.

### 2.10.2. Schaltplan



## 2.10.3. Projektbild

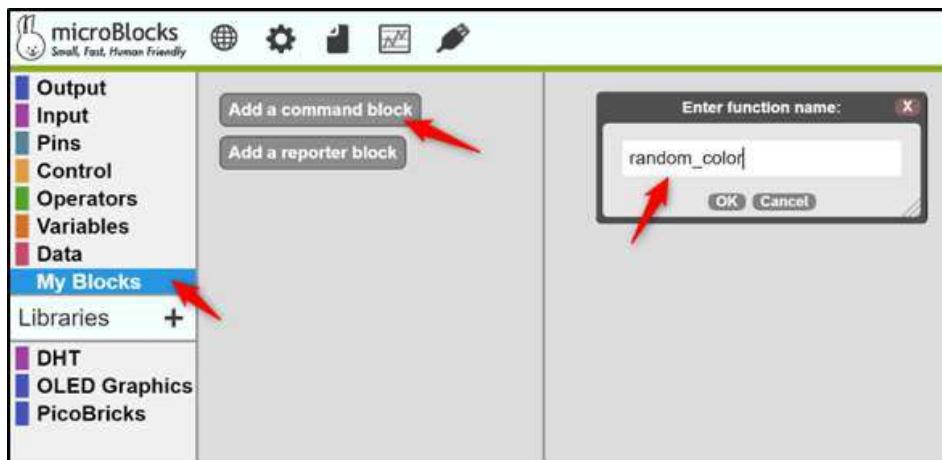


## Projektvorschlag

Sie können das Spiel angenehmer gestalten, indem Sie es ein wenig schwieriger machen. Zum Beispiel können Sie das Spiel beschleunigen, indem Sie die Wiederholungszeit der Farben reduzieren. Oder anstatt Punkte zu verlieren, wenn der Benutzer den Knopf an der falschen Stelle drückt, können Sie das Spiel beenden und es erneut starten.

## 2.10.5. Programmierung des Projekts mit MicroBlocks

Da wir im Projekt ein OLED-Display verwenden werden, müssen Sie zunächst die OLED-Grafikbibliothek zum Programm hinzufügen. Dann müssen Sie zwei Variablen mit den Namen OLED\_color und RGB\_color erstellen, um zufällige Farben auf dem OLED-Bildschirm und der RGB-LED zu erzeugen. Damit das Programm stabiler funktioniert, können wir die zufällige Bestimmung der Farben durchführen, indem wir eine Funktion erstellen und diese Funktion aufrufen. Funktionen werden erstellt, indem Codes, die aus einer oder mehreren Operationen bestehen, als Codeblock strukturiert werden. Sobald Funktionen erstellt sind, können sie überall im Programm nur mit dem Funktionsnamen aufgerufen werden. Um eine Funktion zu erstellen, müssen Sie auf die Schaltfläche Meine Blöcke im Abschnitt Codekategorien klicken, dann auf die Schaltfläche Befehl block hinzufügen klicken und den Namen der Funktion, die Sie erstellen möchten, im Fenster Funktionsnamen eingeben, das sich öffnet. Wir können der Funktion, die wir in diesem Projekt erstellen, den Namen random\_color geben.



Nachdem Sie die Funktion erstellt haben, wird der Block define random\_color in den Code-Schreibbereich kommen. Sie können die Funktionscodes unter diesem Block schreiben. In der random\_color-Funktion weisen wir zunächst der OLED\_color- und RGB\_color-Variablen, die wir zuvor erstellt haben, eine zufällige Zahl zwischen 1 und 4 zu. Dazu sollten Sie den random-Block in der Kategorie Operatoren verwenden. Dann sollten Sie die zufälligen Werte, die den Variablen zugewiesen sind, mit dem if-Block vergleichen und die RGB-LED-Farbe sowie die Texte auf dem OLED-Bildschirm bearbeiten.

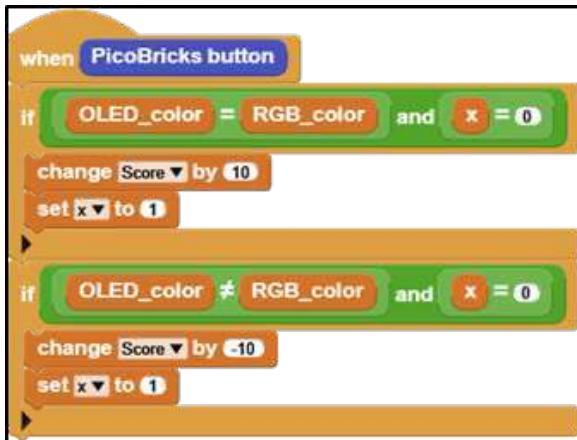
Nachdem die random\_color-Funktion definiert wurde, können wir unser Programm unter dem when started-Block schreiben. In dieser Codegruppe müssen Sie zunächst die Codes schreiben, die ausgeführt werden sollen, wenn das Programm startet. Nach den Startblöcken sollten Sie die random\_color-Funktion, die Sie zuvor erstellt haben, innerhalb des repeat 10-Blocks aufrufen. Nachdem random\_color 10 Mal ausgeführt wurde, sollten Sie eine Weile warten, den Bildschirm löschen und die RGB-LED ausschalten. Sie müssen all diese Schritte 10 Mal wiederholen. Wenn Sie die Codes bis jetzt ausführen, werden Sie sehen, dass 10 Mal zufällige Farben mit einem Intervall von 1,5 Sekunden auf der RGB-LED und dem OLED-Bildschirm erscheinen.

```

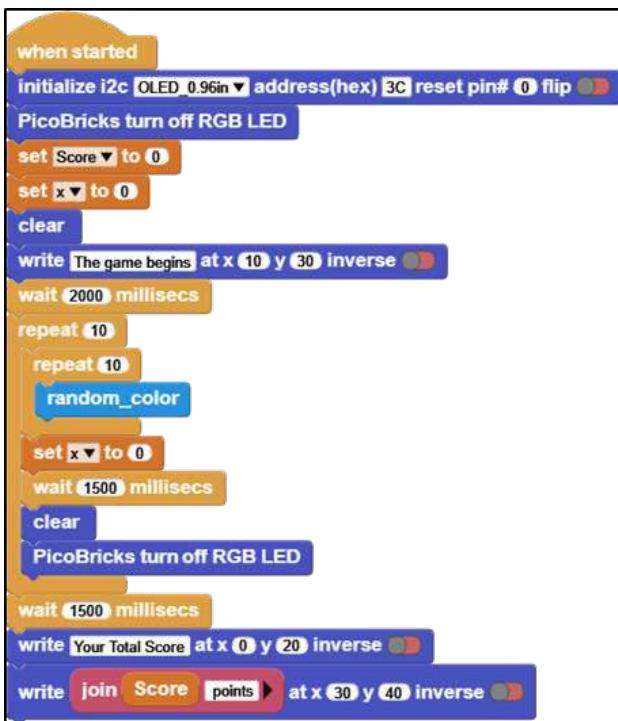
define [random_color]
set [OLED color v] to [random (1) to (4)]
set [RGB_color v] to [random (1) to (4)]
if [RGB_color = 1] then
  PicoBricks set RGB LED color [red v]
end
if [RGB_color = 2] then
  PicoBricks set RGB LED color [green v]
end
if [RGB_color = 3] then
  PicoBricks set RGB LED color [blue v]
end
if [RGB_color = 4] then
  PicoBricks set RGB LED color [white v]
end
if [OLED_color = 1] then
  clear
  write [red] at x [50] y [32] inverse [red]
end
if [OLED_color = 2] then
  clear
  write [green] at x [45] y [32] inverse [green]
end
if [OLED_color = 3] then
  clear
  write [blue] at x [50] y [32] inverse [blue]
end
if [OLED_color = 4] then
  clear
  write [white] at x [45] y [32] inverse [white]
end
wait (20) millisecs
repeat (10)
  repeat (10)
    [random_color v]
    wait (1500) millisecs
    clear
  end
  PicoBricks turn off RGB LED
end

```

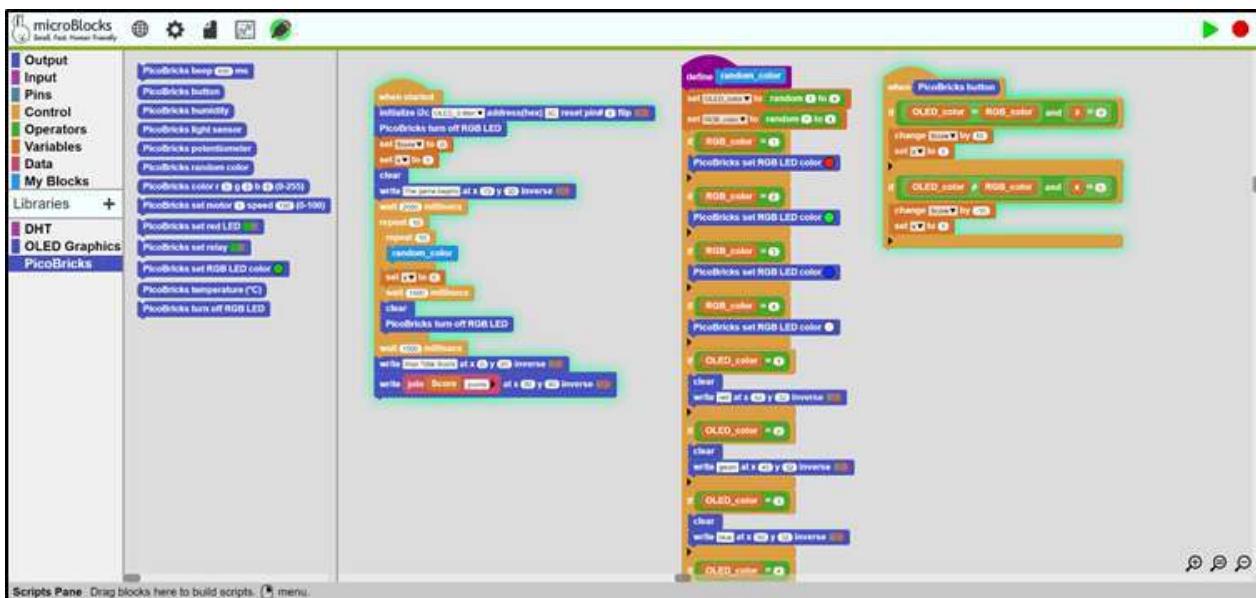
In diesem Stadium, wenn die Farben auf dem OLED-Bildschirm und der RGB-LED gleich sind, erstellen Sie zwei weitere Variablen namens Score und x, damit der Benutzer den Knopf drücken und Punkte verdienen kann. Fügen Sie den Block für das Drücken des Picobricks-Knopfes als Bedingung zum when-Block hinzu. Vergleichen Sie dann die beiden if-Blöcke, um zu sehen, ob die Farbvariablen gleich sind. Wenn die Farben gleich sind, erhöhen Sie die Score-Variable um 10, andernfalls verringern Sie sie um 10. Der Grund, warum wir hier die x-Variable verwenden, besteht darin, den Score nur einmal zu erhöhen. Wenn Sie die Variable x nicht als zweite Bedingung im and-Operator verwenden, wird die Score-Variable während des Drückens des Knopfes kurzfristig um 10 erhöht.



Nachdem Sie die Aktionen, die das Programm ausführen soll, wenn der Knopf gedrückt wird, angeordnet haben, müssen Sie, wenn das Spiel vorbei ist, die notwendigen Codes zum Hauptprogramm hinzufügen, damit der Punktestand des Benutzers auf dem Bildschirm angezeigt wird. Legen Sie die Anfangswerte der Variablen Score und x auf „0“ fest und weisen Sie den Wert der x-Variable bei jeder Iteration innerhalb von 10 Iterationen auf „0“ zu. Auf diese Weise wird jedes Mal, wenn der Knopf im Spiel gedrückt wird, eine einmalige Punktsteigerung erfolgen.



Nachdem der Code-Schreibprozess abgeschlossen ist, starten Sie das Spiel mit der Starttaste und testen Sie es. Wenn alles gut gelaufen ist, erscheinen zufällige Farben auf dem OLED-Bildschirm und RGB-LEDs 10 Mal in Abständen von 1,5 Sekunden. Wenn die Farben gleich sind und die Taste gedrückt wird, erhöht sich die Punktzahl um 10 Punkte, und wenn das Spiel vorbei ist, wird die Punktzahl auf dem Bildschirm angezeigt.



[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.10.6. MicroPython-Codes des Projekts

```

from machine import Pin, I2C
from picobricks import SSD1306_I2C
import utime
import urandom
import _thread
from picobricks import WS2812

WIDTH = 128
HEIGHT = 64
sda=machine.Pin(4)
scl=machine.Pin(5)
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)
ws = WS2812(pin_num=6, num_leds=1, brightness=0.3)

oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)

button = Pin(10,Pin.IN,Pin.PULL_DOWN)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
WHITE = (255, 255, 255)

```

```
BLACK = (0, 0, 0)
```

```
oled.fill(0)  
oled.show()
```

```
ws.pixels_fill(SCHWARZ)  
ws.pixels_show()
```

```
global button_pressed  
punktstand=0  
button_pressed = False
```

```
def random_rgb():  
    global ledcolor  
    ledcolor=int(urandom.uniform(1,4))  
    if ledcolor == 1:  
        ws.pixels_fill(ROT)  
            ws.pixels_show()  
    elif ledcolor == 2:  
        ws.pixels_fill(GRÜN)  
            ws.pixels_show()  
    elif ledcolor == 3:  
        ws.pixels_fill(BLAU)  
            ws.pixels_show()  
    elif ledcolor == 4:  
        ws.pixels_fill(WEISS)  
            ws.pixels_show()
```

```
def random_text():  
    global oledtext  
    oledtext=int(urandom.uniform(1,4))  
    if oledtext == 1:  
        oled.fill(0)  
            oled.show()  
            oled.text("ROT",45,32)  
            oled.show()  
    elif oledtext == 2:  
        oled.fill(0)  
        oled.show()  
            oled.text("GRÜN",45,32)  
            oled.show()
```

```
elif oledtext == 3:  
    oled.fill(0)  
    oled.show()  
    oled.text("BLAU",45,32)  
    oled.show()  
elif oledtext == 4:  
    oled.fill(0)  
    oled.show()  
    oled.text("WEISS",45,32)  
    oled.show()  
  
def button_reader_thread():  
    while True:  
        global button_pressed  
        if button_pressed == False:  
            if button.value() == 1:  
                button_pressed = True  
                global score  
                global oledtext  
                global ledcolor  
                if ledcolor == oledtext:  
                    score += 10  
                else:  
                    score -= 10  
                utime.sleep(0.01)  
  
    _thread.start_new_thread(button_reader_thread, ())  
  
oled.text("Das Spiel beginnt",0,10)  
oled.show()  
utime.sleep(2)  
  
for i in range(10):  
    random_text()  
    random_rgb()  
    button_pressed=False  
    utime.sleep(1.5)  
    oled.fill(0)  
    oled.show()  
    ws.pixels_fill(BLACK)  
    ws.pixels_show()
```

```
utime.sleep(1.5)
oled.fill(0)
oled.show()
oled.text("Ihr Gesamtergebnis:", 0, 20)
oled.text(str(score), 30, 40)
oled.show()
```

## 2.10.7. Arduino C-Codes des Projekts

```
#include <Adafruit_NeoPixel.h>
#define PIN      6
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
#define DELAYVAL 500
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h" //Bibliotheken definieren
int OLED_color;
int RGB_color;
int score = 0;
int button = 0;

void setup() {
// Fügen Sie hier Ihren Setup-Code ein, der einmal ausgeführt wird:
    Wire.begin();
    oled.init();
    oled.clearDisplay();

    pixels.begin();
    pixels.clear();
    randomSeed(analogRead(27));

}

void loop() {
// Fügen Sie hier Ihren Hauptcode ein, der wiederholt ausgeführt wird:
    oled.clearDisplay();
    oled.setTextXY(3,1);
```

```
oled.putString("Das Spiel beginnt");
pixels.setPixelColor(0, pixels.Color(0, 0, 0));
pixels.show();
delay(2000);
oled.clearDisplay();

for (int i=0;i<10;i++){
    button = digitalRead(10);
    random_color();
    pixels.show();
    unsigned long start_time = millis();
    while (button == 0) {
        button = digitalRead(10);
        if (millis() - start_time > 2000)
            break;
    }
    if (button == 1){

        if(OLED_color==RGB_color){
            score=score+10;
        }
        if(OLED_color!=RGB_color){
            score=score-10;
        }
        delay(200);
    }
    oled.clearDisplay();
    pixels.setPixelColor(0, pixels.Color(0, 0, 0));
    pixels.show();
}

String string_scrore=String(score);
oled.clearDisplay();
oled.setTextXY(2,5);
oled.putString("Score: ");
oled.setTextXY(4,7);
oled.putString(string_scrore);
oled.setTextXY(6,5);
oled.putString("points");
// print final score on OLED screen
```

```
delay(10000);
}

void random_color(){

OLED_color = random(1,5);
RGB_color = random(1,5);
// generate numbers between 1 and 5 randomly and print them on the screen
if (OLED_color == 1){
    oled.setTextXY(3,7);
    oled.putString("red");
}
if (OLED_color == 2){
    oled.setTextXY(3,6);
    oled.putString("grün");
}
if (OLED_color == 3){
    oled.setTextXY(3,6);
    oled.putString("blau");
}
if (OLED_color == 4){
    oled.setTextXY(3,6);
    oled.putString("weiß");
}
if (RGB_color == 1){
    pixels.setPixelColor(0, pixels.Color(255, 0, 0));
}
if (RGB_color == 2){
    pixels.setPixelColor(0, pixels.Color(0, 255, 0));
}
if (RGB_color == 3){
    pixels.setPixelColor(0, pixels.Color(0, 0, 255));
}
if (RGB_color == 4){
    pixels.setPixelColor(0, pixels.Color(255, 255, 255));
}
}
```

---

GitHub Wissen Sie Ihr Farbprojektseite



<http://rbt.ist/color>

## 2.11. Magische Lampe

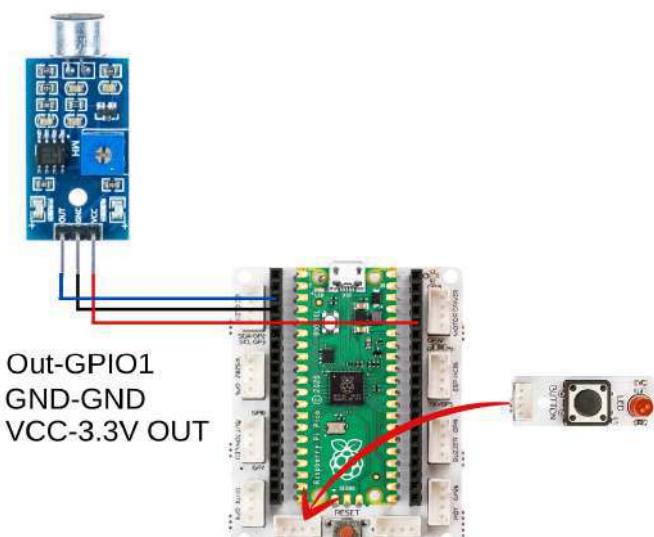
Projektautor: Abdullah KAYA

Die meisten von uns haben in Filmen Lampen gesehen, die magisch blitzen, oder Türen, die sich mit dem Geräusch von Applaus öffnen und schließen. Es gibt Set-Assistenten, die diese Türen schließen und die Lampen während der Dreharbeiten ausschalten. Was wäre, wenn wir das automatisch machen könnten? Es gibt Sensoren, die die erwartete Veränderung der Schallintensität in der Umgebung in ein elektrisches Signal umwandeln. Diese werden als Schallsensoren bezeichnet.

### 2.12.1. Projektdetails und Algorithmus

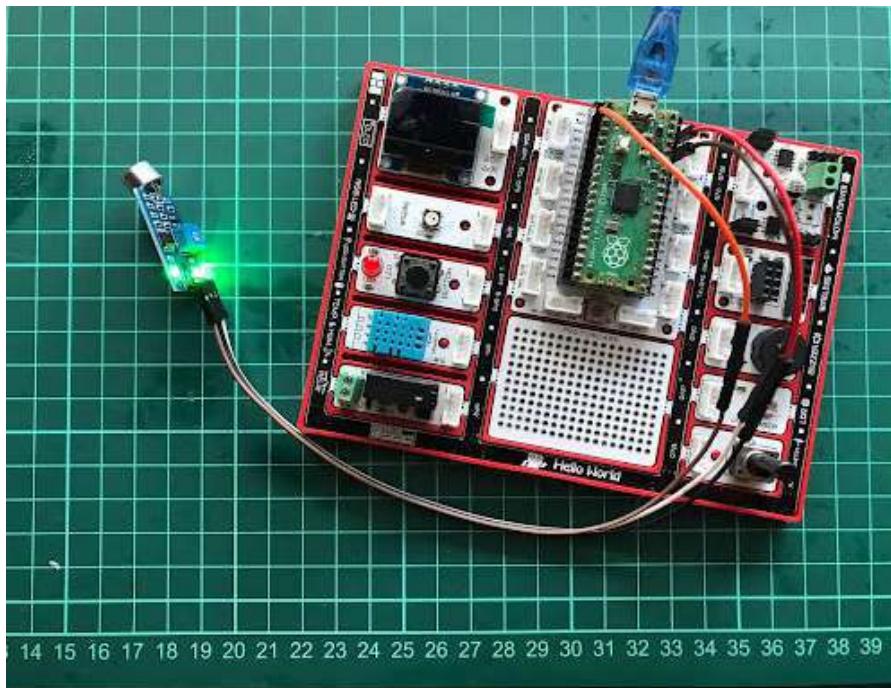
In diesem Projekt werden wir das LED-Modul auf der Picobricks-Platine mit dem Geräusch ein- und ausschalten. In unserem Projekt, das wir mit dem Picobricks-Schallpegel-Sensor aufbauen werden, führen wir die Ein- und Ausschaltvorgänge durch, indem wir ein Klatschgeräusch erzeugen. Wie in früheren Projekten wird es, bevor wir mit dem Schreiben des Codes beginnen, einfacher sein, zu sehen, welche Werte der Sensor bei den Operationen sendet, die wir durchführen möchten, indem wir den Sensor einfach laufen lassen, und dann den Code des Projekts basierend auf diesen Werten zu schreiben.

### 2.12.2. Schaltplan



### 2.11.3. Bauphasen des Projekts

Während des Baus des Projekts wurden zwei Drahtsteckdosen und Steckdosen verwendet. Die beiden Enden, die durch das Schneiden des Phasenkabels abgeschnitten wurden, wurden mit dem Relais verbunden. Sie sollten auf die Isolierung mit Isolierband achten, damit keine gefährliche Situation entsteht, wenn Sie den anderen Draht schneiden. Wenn Sie eine Dreidrahtsteckdose verwenden, müssen Sie den braunen Draht mit dem Phasenkabel schneiden und ihn mit dem Relais verbinden.



#### 2.11.4. Projektvorschlag

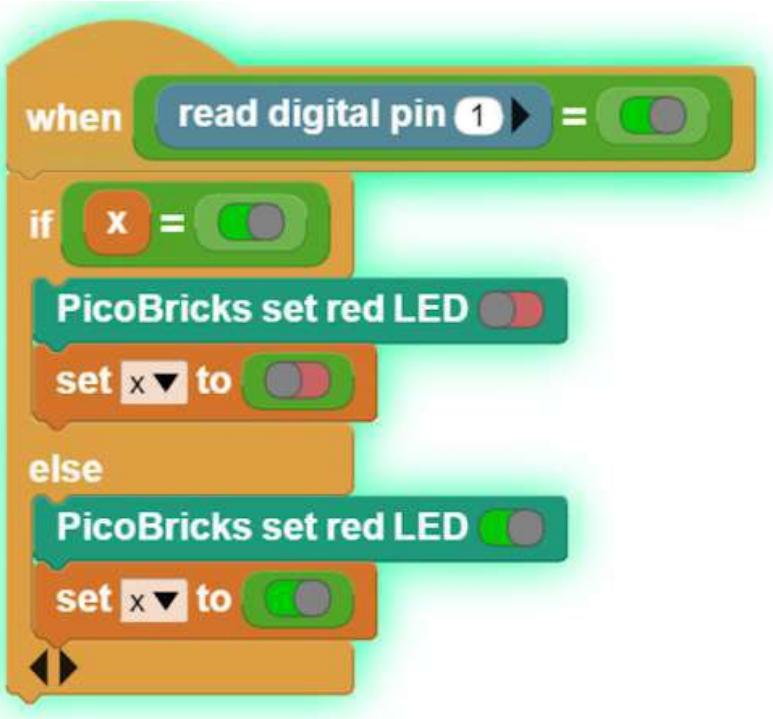
Sie können dem Spieler Anweisungen und Benachrichtigungen auf dem OLED-Bildschirm präsentieren. Darüber hinaus können Sie ein spannenderes Spiel vorbereiten, indem Sie auf dem OLED-Bildschirm anzeigen, wie viele Millisekunden nach Spielbeginn das Spiel beendet ist.

#### 2.11.5. Programmierung des Projekts mit MicroBlocks

Beim Schreiben der Microblocks-Codes erstellen wir zunächst eine Variable namens x und setzen den Anfangswert sowohl der Variablen als auch des Relais auf falsch. Wir werden die x-Variable verwenden, um die Lampe auszuschalten, wenn sie eingeschaltet ist, und einzuschalten, wenn sie ausgeschaltet ist.



Der digitale Schallsensor sendet uns einen Wert von „0“, wenn über den digitalen Pin, an den er angeschlossen ist, ein Geräusch erkannt wird, und „1“, wenn er im Leerlauf ist. MicroBlocks interpretiert diese Werte als „Wahr“ und „Falsch“. In den von uns geschriebenen Codes wird, wenn der Wert, der vom digitalen Pin 16 kommt, „0“ ist, also „Falsch“, das Relais aktiviert. Wenn das Relais aktiviert wird, wird der Wert der x-Variable überprüft. Wenn x falsch ist, wird das Relais eingeschaltet und der Wert der x-Variable wird auf „Wahr“ geändert, andernfalls wird das Relais geschlossen und der Wert der x-Variable wird wieder auf „Falsch“ gesetzt. Diese Codes sorgen dafür, dass die Lampe angeht, wenn wir klatschen, und die Lampe ausgeht, wenn wir erneut klatschen.



[Klicken](#) Sie, um auf die MicroBlock-Codes des Projekts zuzugreifen.

## 2.11.6. MicroPython-Codes des Projekts

```
from machine import Pin
sensor=Pin(16,Pin.IN)
relay=Pin(12,Pin.OUT)
x=0
while True:
  if sensor.value()==0:
    if x==0:
      relay.value(1)
      x=1
    else:
      relay.value(0)
      x=0
```

## 2.11.7. Arduino C Codes des Projekts

```
void setup() {
```

```
// Fügen Sie hier Ihren Setup-Code ein, der einmal ausgeführt wird:  
pinMode(1,INPUT);  
pinMode(7,OUTPUT);  
//definiere die Eingangs- und Ausgangspins  
}  
  
void loop() {  
// Fügen Sie hier Ihren Hauptcode ein, der wiederholt ausgeführt wird:  
  
Serial.println(digitalRead(1));  
  
if(digitalRead(1)==1){  
  digitalWrite(7,HIGH);  
  delay(3000);  
}  
else{  
  digitalWrite(7,LOW);  
  delay(1000);  
}  
}
```

GitHub Magic Lamp Projektseite



<http://rbt.ist/lamp>

## 2.12. Intelligenter Kühler

Projektautor: Abdullah KAYA

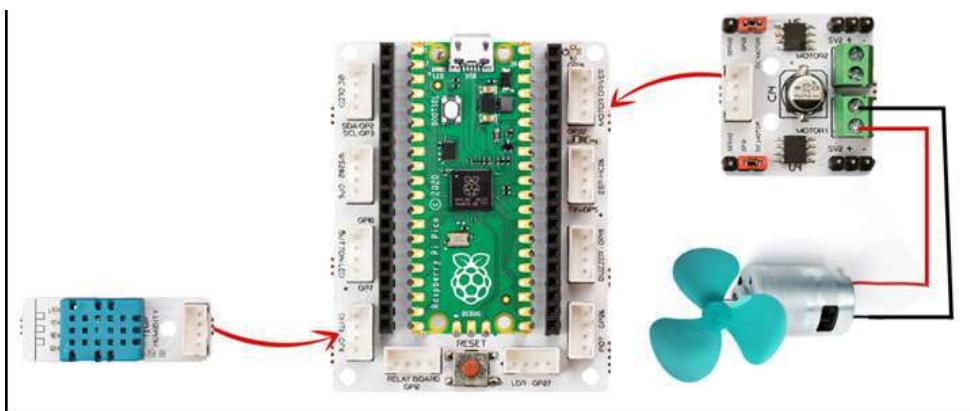
Klimaanlagen werden im Sommer zum Kühlen und im Winter zum Heizen verwendet. Klimaanlagen passen den Grad der Heizung und Kühlung entsprechend der Temperatur der Umgebung an. Während des Kochens versuchen die Öfen, die vom Benutzer festgelegte Temperatur zu erreichen und aufrechtzuerhalten. Diese beiden elektronischen Geräte verwenden spezielle Temperatursensoren zur Steuerung der Temperatur. Darüber hinaus werden Temperatur und Luftfeuchtigkeit in Gewächshäusern zusammen gemessen. Um diese beiden Werte im gewünschten Gleichgewicht zu halten, wird versucht, mit dem Ventilator einen Luftstrom bereitzustellen.

In Picobricks können Sie Temperatur und Luftfeuchtigkeit separat messen und mit diesen Messungen mit der Umgebung interagieren. In diesem Projekt werden wir ein Kühlssystem vorbereiten, das die Ventilatorgeschwindigkeit automatisch entsprechend der Temperatur mit Picobricks anpasst. Auf diese Weise lernen Sie das Betriebssystem des Gleichstrommotors und die Regelung der Motorgeschwindigkeit kennen.

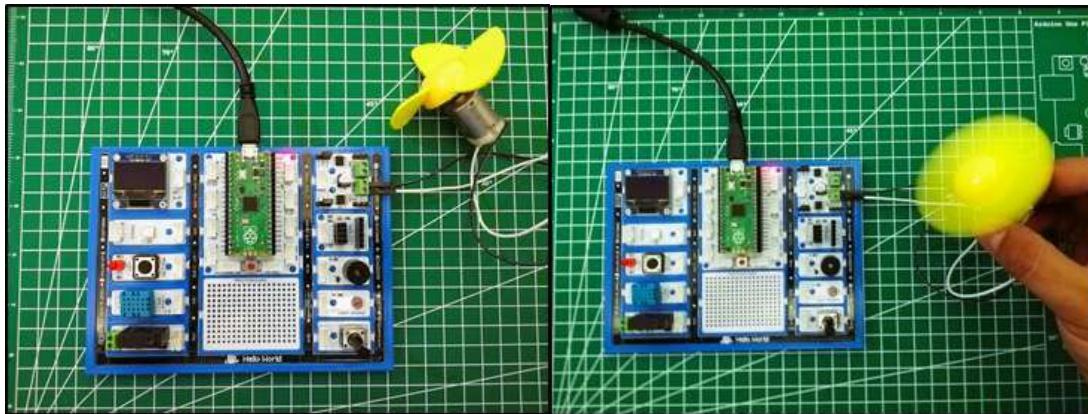
### 2.12.1. Projektdetails und Algorithmus

In unserem Projekt werden wir zunächst die Temperaturwerte, die vom DHT11-Temperatur- und Feuchtigkeitssensor auf Picobricks gemessen werden, anzeigen. Dann werden wir eine Temperaturgrenze definieren und die notwendigen Codes schreiben, damit der an Picobricks angeschlossene Gleichstrommotor zu rotieren beginnt, wenn der Temperaturwert des DHT11-Moduls diese Grenze erreicht, und damit der Gleichstrommotor stoppt, wenn der Temperaturwert unter die von uns festgelegte Grenze fällt.

### 2.12.2. Schaltplan



### 2.12.3. Projektbild



### 2.12.4. Projektvorschlag

Mit dem OLED-Bildschirm auf Picobricks können Sie die Temperatur auf dem Bildschirm anzeigen und die Temperatur verfolgen, bei der der Ventilator aktiviert wird.

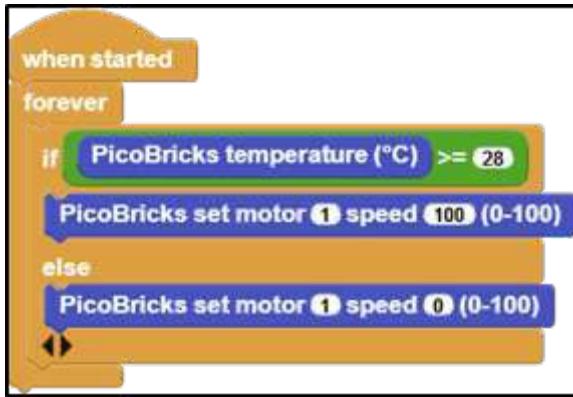
Picobricks hat eine modulare Struktur, Module können durch Brechen getrennt werden und können durch Anschließen an das Pico-Board mit Grove-Kabeln verwendet werden. Durch die Montage des smart cooling circuits, den wir in unserem Projekt erstellt haben, am Chassis des Roboters können Sie ein Projekt entwickeln, das autonom in Ihrer Umgebung navigiert und gleichzeitig die Umgebung kühlt.

### 2.12.5. Programmierung des Projekts mit MicroBlocks



Um zu entscheiden, wann der Ventilator starten und arbeiten soll, müssen wir zuerst die Werte vom DHT11-Sensor sehen und entsprechend diesen Werten handeln. Sie können den say123-Block in der Kategorie Ausgabe dafür verwenden. Ziehen Sie dann den PicoBricks-Temperaturblock aus der Picobricks-Kategorie in den Kreis, der 123 im say-Block ist. Sehen Sie sich die Werte vom Sensor an, indem Sie die Starttaste drücken. Sie sollten Werte rund 25 Grad bei Raumtemperatur sehen. Halten Sie eine Weile den DHT11-Modul auf den Picobricks. Sie werden sehen, dass der Temperaturwert vom Sensor aufgrund der Wärme auf dem Finger des DHT11 ansteigt. Nachdem Sie die Werte gesehen haben, können Sie die say- und Temperaturblöcke löschen.

Nachdem Sie einen Temperaturwert für die Aktivierung des Ventilators entsprechend Ihrer Umgebung festgelegt haben, können Sie das Projekt durchführen, indem Sie den Picobricks-Motorblock im if-else-Block verwenden. Sie können die Ventilatorgeschwindigkeit zwischen 0 und 100 ändern.



[Klicken](#) Sie, um auf die MicroBlock-Codes des Projekts zuzugreifen.

## 2.12.6. MicroPython-Codes des Projekts

Codes, die die aktuelle Temperatur im Shell-Fenster ausgeben:

```
from machine import Pin  
from dht import DHT11  
from utime import sleep  
dht_sensor = DHT11(11)
```

```
while True:  
    sleep(1) # Es wurde verwendet, damit der DHT11 messen kann.  
    dht_sensor.measure() # Verwenden Sie den sleep() Befehl vor dieser Zeile.  
    temp=dht_sensor.temperature()  
    print(temp)
```

Projektcodes:

```
from machine import Pin  
from picobricks import DHT11  
import utime  
  
LIMIT_TEMPERATURE = 20 #definiere die Grenztemperatur  
  
dht_sensor = DHT11(Pin(11, Pin.IN, Pin.PULL_DOWN))  
m1 = Pin(21, Pin.OUT)  
m1.low()  
dht_read_time = utime.time()
```

```
#definiere Eingangs- und Ausgangspins

while True:
    if utime.time() - dht_read_time >= 3:
        dht_read_time = utime.time()
        dht_sensor.measure()
        temp= dht_sensor.temperature
        print(temp)
        if temp >= LIMIT_TEMPERATURE:
            m1.high()
            #operate if the room temperature is higher than the limit temperature
        else:
            m1.low()
```

## 2.12.7. Arduino C Codes of the Project

```
#include <DHT.h>

#define LIMIT_TEMPERATURE 27
#define DHTPIN 11
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
float temperature;

void setup() {
    // Fügen Sie hier Ihren Setup-Code ein, der einmal ausgeführt wird:
    Serial.begin(115200);
    dht.begin();
    pinMode(21,OUTPUT);

}

void loop() {
    // Fügen Sie hier Ihren Hauptcode ein, der wiederholt ausgeführt wird:
    delay(100);
    temperature = dht.readTemperature();
    Serial.print("Temp: ");
    Serial.println(temperature);
    if(temperature > LIMIT_TEMPERATURE){
```

```
    digitalWrite(21,HIGH);
} else{
    digitalWrite(21,LOW);
}

}
```

## GitHub Smart Cooler Projektseite



<http://rbt.ist/coolerm>

## 2.13. Buzz Wire Spiel

Projekte müssen nicht immer darauf abzielen, Probleme zu lösen und Dinge einfacher zu machen. Sie können auch Projekte vorbereiten, um Spaß zu haben und sich selbst weiterzuentwickeln. Aufmerksamkeit und Konzentration sind Eigenschaften, die viele Menschen entwickeln möchten. Die Anwendungen, die wir damit machen können, sind ziemlich interessant. Wie wäre es, ein Buzz Wire Spiel mit Picobricks zu machen?

Sie haben sicherlich den Ausdruck gehört, dass Computer mit 0 und 1 arbeiten. 0 stellt das Fehlen von Elektrizität dar und 1 stellt deren Vorhandensein dar. 0 und 1 kommen mit einer bestimmten Anzahl und Reihenfolge von Kombinationen zusammen, um bedeutungsvolle Daten zu bilden. In elektronischen Systemen können 0 und 1 verwendet werden, um eine Situation direkt zu steuern. Ist die Tür geschlossen oder nicht? Ist das Licht an oder aus? Ist das Bewässerungssystem an oder nicht? Um solche Informationen zu erhalten, wird eine Statusüberprüfung durchgeführt.

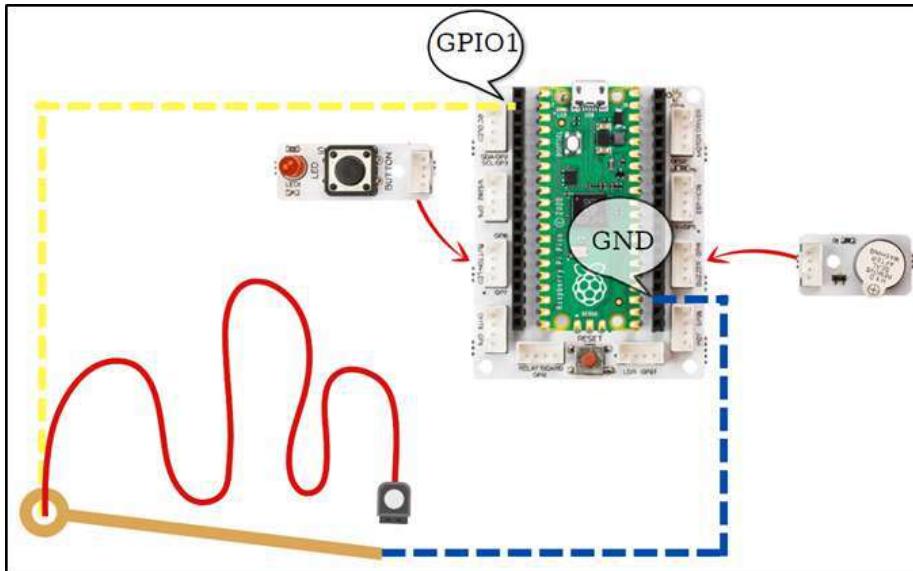
In diesem Projekt werden wir das Aufmerksamkeit und Konzentration fördernde Buzz Wire Spiel elektronisch mit Hilfe eines Leiterdrahts unter Verwendung des Buzzers und LED-Moduls mit Picobricks vorbereiten. Während Sie dieses Projekt vorbereiten, werden Sie eine eingabetechnische Methode lernen, die kein Knopf ist, aber wie ein Knopf verwendet wird.

### 2.13.1. Projektdetails und Algorithmus

Um das Projekt vorzubereiten, benötigen Sie 2 männliche Jumperkabel und einen 15 cm langen biegsamen Leiterdraht. Wenn der Spieler bereit ist, wird er aufgefordert, den Knopf zu drücken, um das Spiel zu starten. Wenn das Jumperkabel den Leiterdraht in der Hand des Spielers berührt, während der Knopf gedrückt wird, wird Picobricks dies erkennen und eine hörbare und schriftliche Warnung ausgeben. Die Zeit vom Beginn des Spiels bis zum Ende wird ebenfalls auf dem OLED-Bildschirm angezeigt.

Wir setzen den Timer zurück, nachdem der Benutzer die Taste drückt. Dann geben wir eine Spannung von 3,3 V an den Leitungsdraht, der mit dem GPIO1-Pin von Picobricks verbunden ist. Ein Ende des Kabels, das der Spieler hält, wird mit dem GND-Pin auf den Picobricks verbunden. Wenn der Spieler das Jumperkabel in seiner Hand mit dem leitenden Draht berührt, fällt der GPIO1-Pin in die Passive/Aus/0-Position. Dann wird angekündigt, dass das Spiel vorbei ist, und es wird Licht-, schriftliches und akustisches Feedback geben, danach wird die verstrichene Zeit auf dem OLED-Bildschirm in Millisekunden angezeigt. Nach 5 Sekunden wird der Spieler aufgefordert, die Taste zu drücken, um neu zu starten.

## 2.13.2. Verdrahtungsdiagramm



## 2.13.3. Projektvorschlag

Sie können physische und softwaretechnische Verbesserungen am Projekt vornehmen. Indem Sie die Start- und Endpunkte mit Isolierband abdecken, können Sie verhindern, dass der Spieler Probleme beim Starten und Beenden des Spiels hat. In Bezug auf die Software, wenn der Spieler das Kabel zum anderen Ende bringt, ohne den Draht zu berühren, drücken Sie die Taste und Sie können den Punktestand auf dem OLED-Bildschirm sehen.

## 2.13.4. Programmierung des Projekts mit MicroBlocks

Wenn PicoBricks startet, öffnen wir sofort eine Endlosschleife, nachdem der OLED-Bildschirm gestartet ist. Denn wenn das Spiel vorbei ist, wird es zum Anfang zurückkehren. Innerhalb der Endlosschleife schalten wir zunächst die rote LED aus und platzieren die Startnachricht auf dem OLED-Bildschirm. Dann platzieren wir den Block "Warten bis" aus der Kategorie Steuerung und warten in der Schleife, bis die Taste von Picobricks gedrückt wird.

Nachdem die Taste gedrückt wurde, fügen wir einen Ausdruck auf dem OLED-Bildschirm hinzu, dass das Spiel mit dem Fortsetzen des Codes beginnt. Dann öffnen wir den GPIO1-Pin und geben eine Spannung von 3,3 V. Nach dem Zurücksetzen des Timers warten wir, bis der GPIO1-Pin 0 im "Warten bis"-Block geschlossen ist. Hier wird der Zustand des Spielers überprüft, der den Draht in der Hand mit dem Draht berührt. Wenn eine Berührung erkannt wird, setzt sich der Zyklus von dort fort, wo er aufgehört hat, und die notwendigen beleuchteten, schriftlichen und akustischen Benachrichtigungen werden präsentiert. Schließlich warten wir 5 Sekunden und kehren zum Anfang der Schleife zurück.

```

when started
  initialize i2c [OLED_0.96in v] address(hex) [3C]
  reset pin# [0] flip [red]
forever
  PicoBricks set red LED [red]
  write <BUZZ WIRE GAME> at x [0] y [0] inverse [red]
  write Press the Button at x [0] y [17] inverse [red]
  write TO START! at x [25] y [35] inverse [red]
  wait until PicoBricks button = [green]
  clear
  write GAME at x [25] y [35] inverse [red]
  write STARTED! at x [25] y [45] inverse [red]
  set digital pin [1] to [green]
  reset timer
  wait until read digital pin [1] = [red]
  clear
  set time ▾ to [timer]
  say [time] ▶
  write GAME OVER! at x [25] y [35] inverse [red]
  write join [time] [ms] ▶ at x [25] y [45] inverse [red]
  PicoBricks set red LED [green]
  PicoBricks beep [500] ms
  wait [5000] millisecs
  clear

```

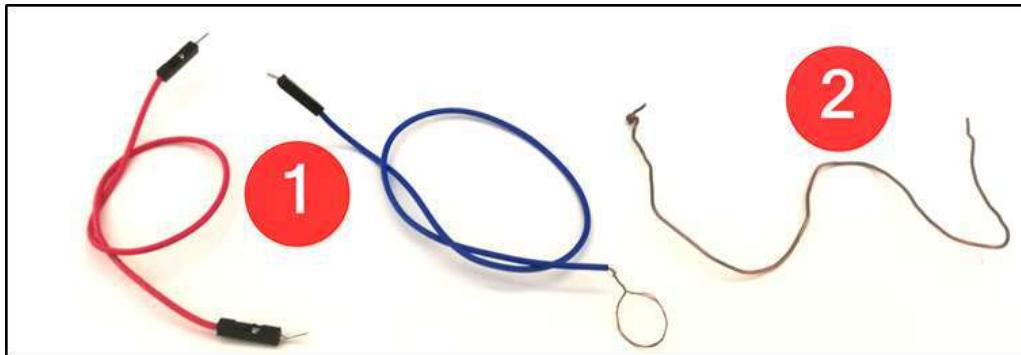
[Klicken](#) Sie, um auf die Codes des Projekts zuzugreifen.

## 2.13.5. Bauphasen des Projekts

Zusammen mit dem PicoBricks-Basiskit,

1: 2 20 cm männliche-männliche Jumperkabel. Ein Ende des Kabels, das an den GND angeschlossen werden soll, wird 4-5 cm abisoliert und zu einem Ring geformt.

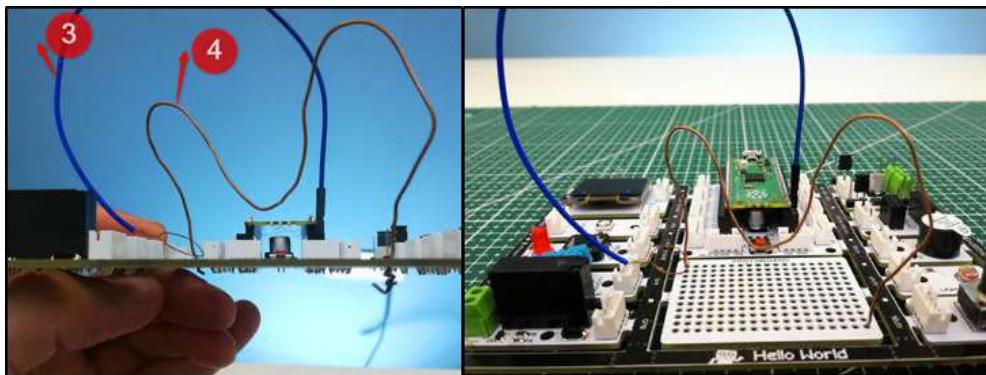
2: 15-20 cm leitfähiger Draht mit einer Dicke von 0,8 mm. Bereiten Sie Ihre Materialien vor.



Biegen Sie den Leitungsdraht auf dem Prototypenbrett nach Belieben und führen Sie ihn durch die Löcher. Bevor Sie ein Ende durchstecken, müssen Sie das männliche Ende durchstecken, das mit dem GND-Pin auf dem PicoBoard verbunden ist. Das andere Ende des Kabels haben Sie zu einem Ring geformt.

### 3: Leitungsdraht

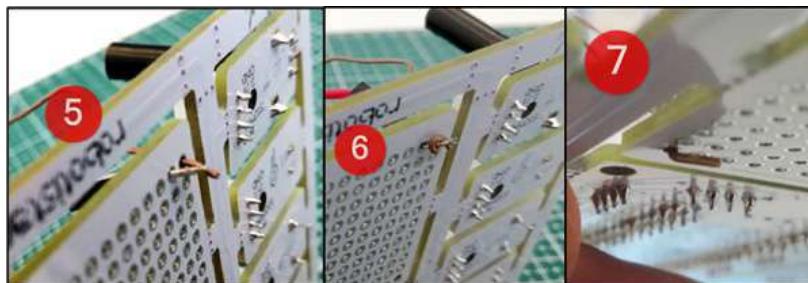
4: Jumper-Kabel mit einem Ende, das mit dem GND-Pin verbunden ist und einem geschlungenen Ende.



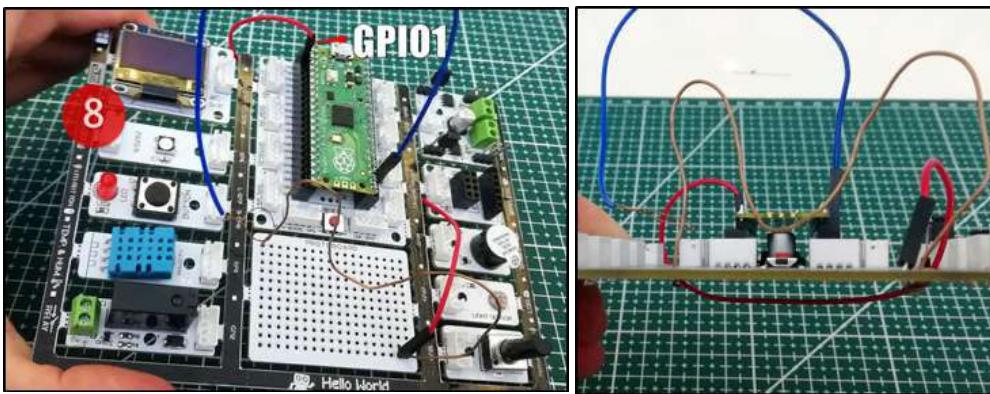
5: Stecken Sie ein Ende des Jumper-Kabels, das beide männliche Enden hat, in das Loch direkt neben dem Ende des Leitungsdräts, den Sie auf dem Prototypenbrett platziert haben.

6: Verdrehen Sie das Ende des Jumper-Drahts und das Ende des Leitungsdräts zusammen unter dem Prototypenbrett.

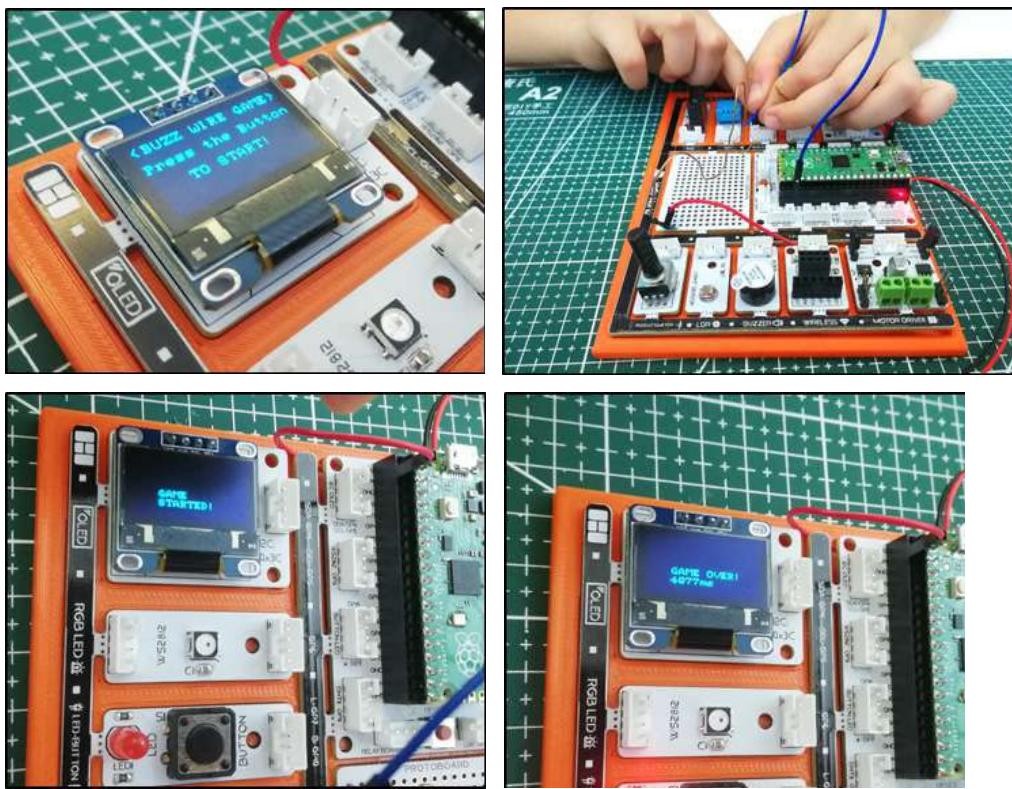
7: Biegen Sie das andere Ende des Leitungsdräts, das auf dem Prototypenbrett platziert ist, so, dass es nicht herauskommt.



8: Verbinden Sie das andere männliche Ende des Jumper-Kabels, das Sie um das Ende des Leitungsdräts in Schritt 6 gewickelt haben, mit dem Pin Nr. GPIO1 auf dem Picoboard.



Wenn Sie die Installation abgeschlossen haben, können Sie das Spiel nach der Installation der Codes starten. Viel Spaß. :)



## 2.13.6. MicroPython-Codes des Projekts

```
from machine import Pin, I2C, Timer #um auf die Hardware des Pico zuzugreifen
from picobricks import SSD1306_I2C #OLED-Bildschirmbibliothek
from utime import sleep # Zeitbibliothek
```

```
#OLED-Bildschirmeinstellungen
BREITE = 128
```

HEIGHT = 64

```
sda=machine.Pin(4)#digitalen Pin 4 und 5 als AUSGANG für OLED  
Kommunikation initialisieren  
scl=machine.Pin(5)  
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)  
oled = SSD1306_I2C(BREITE, HÖHE, i2c)  
  
wire=Pin(1,Pin.OUT)#digitalen Pin 1 als AUSGANG initialisieren  
led = Pin(7,Pin.OUT)#digitalen Pin 7 und 5 als AUSGANG für LED initialisieren  
buzzer=Pin(20, Pin.OUT)#digitalen Pin 20 als AUSGANG für Buzzer initialisieren  
button=Pin(10,Pin.IN,Pin.PULL_DOWN)#digitalen Pin 10 als EINGANG für den Knopf initialisieren  
endtime=0
```

```
while True:  
    led.low()  
    oled.fill(0)  
    oled.show()  
    oled.text("<BUZZ WIRE GAME>",0,0)  
    oled.text("Drücke den Knopf",0,17)  
    oled.text("UM ZU STARTEN!",25,35)  
    oled.show()  
#Wenn der Knopf '0' ist, sagt OLED 'SPIEL GESTARTET'  
    while button.value()==0:  
        print("drücke den Knopf")  
        oled.fill(0)  
        oled.show()  
        oled.text("SPIEL",25,35)  
        oled.text("GESTARTET",25,45)  
        oled.show()  
        wire.high()  
    timer_start=utime.ticks_ms()  
    #Wenn wire '1' ist, sagt OLED 'GAME OVER'  
    während wire.value()==1:  
        print("Started")  
    endtime=utime.ticks_diff(utime.ticks_ms(), timer_start)  
    print(endtime)  
    oled.fill(0)  
    oled.show()  
    oled.text("GAME OVER!",25,35)  
    oled.text(endtime + " ms" ,25,45)
```

```
oled.show()
led.high()#LED Ein
buzzer.high()#Buzzer Ein
sleep(5)#Verzögerung
```

## 2.13.7. Arduino C Codes des Projekts

```
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"

int Time=0;
unsigned long Old_Time=0;

void setup() {

pinMode(20,OUTPUT);
pinMode(7,OUTPUT);
pinMode(1,OUTPUT);
pinMode(10,INPUT);

Wire.begin();
oled.init();
oled.clearDisplay();

#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
clock_prescale_set(clock_div_1);
#endif
}

void loop() {

digitalWrite(7,LOW);

oled.setTextXY(2,1);
oled.putString("BUZZ WIRE GAME");
oled.setTextXY(4,2);
oled.putString("Drücken Sie die Taste");
oled.setTextXY(5,3);
oled.putString("UM ZU STARTEN!");
```

```
while (!(digitalRead(10)==1)){  
  
}  
  
oled.clearDisplay();  
oled.setTextXY(3,6);  
oled.putString("SPIEL");  
oled.setTextXY(5,4);  
oled.putString("GESTARTET");  
  
digitalWrite(1,HIGH);  
Old_Time=millis();  
  
while(!(digitalRead(1)==0)){  
  
    Time=millis()-Old_Time;  
}  
  
String(String_Time)=String(Time);  
  
oled.clearDisplay();  
oled.setTextXY(3,4);  
oled.putString("GAME OVER");  
oled.setTextXY(5,4);  
oled.putString(String_Time);  
oled.setTextXY(5,10);  
oled.putString("ms");  
  
digitalWrite(7,HIGH);  
digitalWrite(20,HIGH);  
delay(500);  
digitalWrite(20,LOW);  
delay(5000);  
  
Time=0;  
Old_Time=0;  
oled.clearDisplay();  
}
```

---

## GitHub Buzz Wire Spiel Projektseite



<http://rbt.ist/buzzwire>

## 2.14. Dinosaurier Spiel

Wenn die zu entwickelnden elektronischen Systeme ihre Aufgaben durch Drücken, Ziehen, Drehen, Heben, Senken usw. erfüllen sollen, werden pneumatische Systeme oder elektrische Motorsysteme als Aktuatoren im Projekt verwendet. Picobricks unterstützt zwei verschiedene Motorarten, damit Sie Systeme produzieren können, die die Codes aktivieren, die Sie in Ihren Projekten schreiben. Gleichstrommotoren und Servomotoren, bei denen die Bewegungen der Gleichstrommotoren elektronisch geregelt werden. Servomotoren sind Motoren, die sich auf den angegebenen Winkel drehen, wenn der Drehwinkelwert gegeben wird. In RC-Booten werden Servomotoren mit derselben Logik verwendet, um die Richtung des Fahrzeugs zu ändern. Darüber hinaus werden auch fortschrittliche Servomotoren, die als intelligente kontinuierliche Servos bekannt sind und sich vollständig drehen können, in den Rädern der intelligenten Staubsauger verwendet, die wir in unseren Haushalten nutzen.

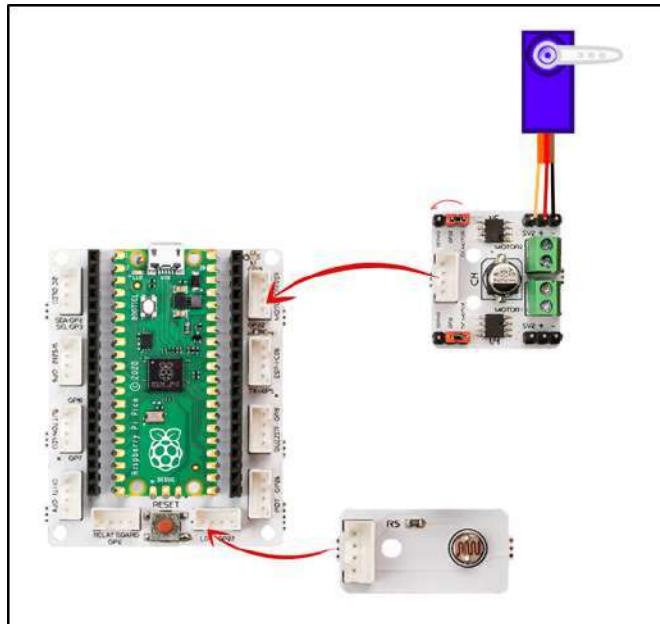
In diesem Projekt lernen Sie, wie man Servomotoren mit PicoBricks steuert.

### 2.14.1. Projektdetails und Algorithmus

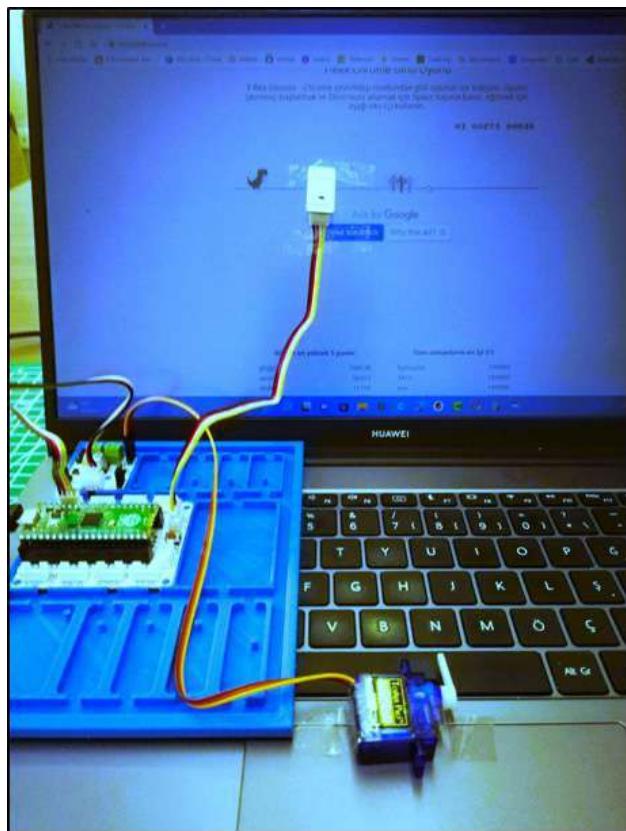
Beim Schreiben des Projektcodes werden wir zunächst den LDR-Sensor auf dem Computerbildschirm befestigen und die Sensordaten auf dem weißen und schwarzen Hintergrund lesen, dann die notwendigen Codes für den Servomotor schreiben, um sich entsprechend diesen Daten zu bewegen. In diesem Projekt werden wir das Offline-Dinosaurierspiel von Google Chrome automatisch mit Picobricks spielen. Im Spiel wird Picobricks die Bewegungen des Dinosauriers automatisch steuern, indem es Hindernisse erkennt. Wir werden den LDR-Sensor von Picobricks verwenden, um die Hindernisse vor dem Dinosaurier während des Spiels zu erkennen. Der LDR kann analoge Signale senden, indem er die Menge des Lichts misst, das die Sensoroberfläche berührt. Indem wir den Sensor auf dem Computerbildschirm befestigen, können wir erkennen, ob sich ein Hindernis vor dem Dinosaurier befindet, indem wir den Unterschied in der Lichtmenge zwischen den weißen und schwarzen Farben nutzen. Wenn ein Hindernis erkannt wird, können wir einen Servomotor verwenden, um automatisch die Leertaste auf der Tastatur zu drücken. Auf diese Weise wird der Dinosaurier die Hindernisse leicht überwinden. Beim Schreiben des Projektcodes werden wir zunächst den LDR-Sensor auf dem Computerbildschirm befestigen und die Sensordaten auf dem weißen und schwarzen Hintergrund lesen, dann die notwendigen Codes für den Servomotor schreiben, um sich entsprechend diesen Daten zu bewegen.

## 2.14.2. Schaltplan

Hinweis: Es gibt dreifache Pins auf der rechten und linken Seite des Motorsteuerungskabels, und diese Pins sind mit 2 Jumpers kurzgeschlossen. Wenn ein Gleichstrommotor verwendet wird, sollte der Jumper, der auf der Gleichstrommotorseite angebracht werden sollte, entfernt werden, wenn ein Servomotor verwendet wird, und an die Servoseite angebracht werden.



## 2.14.3. Projektbild



## 2.14.4. Projektvorschlag

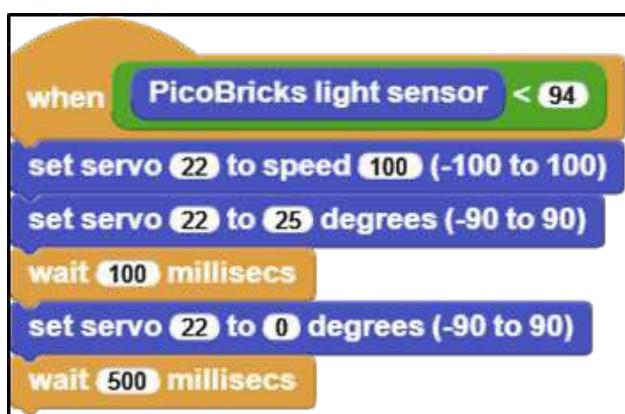
Zu Beginn des Spiels ist die Bodenfarbe weiß und die Figuren sind schwarz. Nach einer bestimmten Phase werden die Farben umgekehrt. Aus diesem Grund ändern sich die LDR-Sensordaten. Um dieses Problem zu lösen, können Sie Variablen und Funktionen verwenden, um eine Codegruppe auszuführen, wenn das Spiel auf einem weißen Hintergrund ist, eine andere Codegruppe, wenn es auf einem schwarzen Hintergrund ist, oder Sie können einen zweiten LDR-Sensor installieren, um diesen Unterschied zu erkennen.

Picobricks und seine Module ermöglichen es uns, viele Projekte von einfach bis komplex zu entwickeln. Sie können es auch in verschiedenen Spielen wie Minecraft verwenden, indem Sie dieses Projekt entwickeln, das wir automatisch mit Picobricks spielen, ein Computerspiel, das wir im Alltag spielen.

## 2.14.5. Codierung des Projekts mit MicroBlocks

Um das Projekt zum Laufen zu bringen, müssen Sie zunächst die LDR-Sensorwerte lesen, die sich je nach Umgebung ändern. Sie können dafür den say123-Block verwenden. Öffnen Sie das Offline-Dinosaurier-Spiel in Chrome. Befestigen Sie den Sensor 3-4 cm rechts vom Dinosaurier und direkt oberhalb der Straßenlinie mit Hilfe von Klebeband. Stellen Sie sicher, dass der Sensor den Bildschirm berührt, und lesen Sie die Sensorwerte. Die Werte auf dem weißen Boden werden von den Werten abweichen, wenn ein Hindernis kommt. Wandeln Sie den Grenzwert, den Sie durch Beobachtung des Unterschieds bestimmen, mit dem when-Block in den Code um.

Wenn der LDR-Sensorwert kleiner ist als der von Ihnen festgelegte Wert, schreiben Sie die notwendigen Codes, um den Winkel des Servos um 25 Grad zu ändern und in seine ursprüngliche Position zurückzukehren, und befestigen Sie den Servomotor an Ihrer Tastatur, sodass er automatisch die Leertaste drückt.



[Klicken](#) Sie, um auf die Codes des Projekts zuzugreifen.

## 2.14.6. MicroPython-Codes des Projekts

```
from machine import Pin, ADC, PWM #um auf die Hardware auf dem Pico zuzugreifen  
from utime import sleep #Zeitbibliothek
```

```
ldr=ADC(27)      #digitalen Pin 27 für LDR initialisieren
```

```
servo=PWM(Pin(21)) #digitalen PWM-Pin 27 für Servomotor initialisieren  
servo.freq(50)  
  
while True:      #Wenn LDR-Daten höher als 40000  
    sleep(0.01)  
    if ldr.read_u16()>4000:  
        servo.duty_u16(2000) #setzt die Position auf 180 Grad  
        sleep(0.1)          #Verzögerung  
        servo.duty_u16(1350) #setzt die Position auf 0 Grad  
        sleep(0.5)          #Verzögerung
```

## 2.14.7. Arduino C-Codes des Projekts

```
#include <Servo.h>  
Servo myservo;  
  
void setup() {  
  myservo.attach(22);  
  myservo.write(20);  
  pinMode(27,INPUT);  
}  
  
void loop() {      // put your main code here, to run repeatedly:  
  
  int light_sensor=analogRead(27);  
  
  if(light_sensor>100){  
  
    int x=45;  
    int y=20;  
  
    myservo.write(x);  
    delay(100);  
    myservo.write(y);  
    delay(500);  
  }  
}
```

---

## GitHub Dinosaurier-Spiel Projektseite



<http://rbt.ist/dinosaur>

## 2.15. Nacht und Tag

Wie wäre es, das Spiel Nacht und Tag, das du in der Schule gespielt hast, elektronisch zu spielen? Das Spiel Nacht und Tag ist ein Spiel, bei dem wir unseren Kopf auf den Tisch legen, wenn unser Lehrer Nacht sagt, und unsere Köpfe heben, wenn unser Lehrer Tag sagt. Dieses Spiel wird ein Spiel sein, bei dem du deine Aufmerksamkeit und Reflexe einsetzen musst. In diesem Projekt werden wir ein 0,96" 128x64 Pixel I2C OLED-Display verwenden. Da OLED-Bildschirme als künstliche Lichtquelle genutzt werden können, kannst du die Zeichen auf dem Bildschirm mit Linsen und Spiegeln vergrößern und sie auf die gewünschte Fläche reflektieren. Systeme, die Informationen, Straßen- und Verkehrsinfos auf Smart Glasses und Autofenstern reflektieren können, können mit OLED-Bildschirmen hergestellt werden.

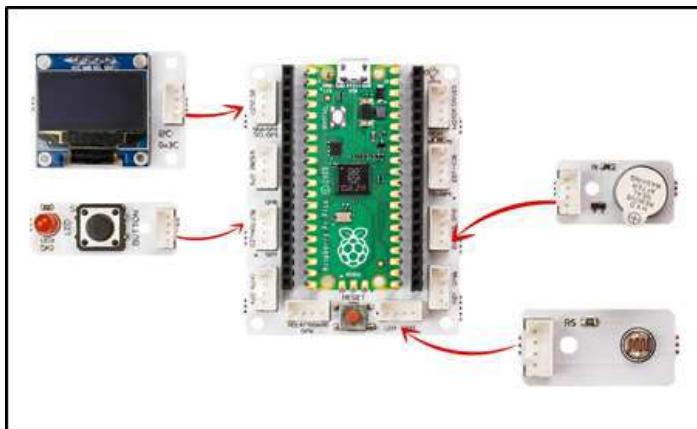
Lichtsensoren sind Sensoren, die die Lichtniveaus der Umgebung, in der sie sich befinden, messen können, auch als Photodioden bezeichnet. Die elektrische Leitfähigkeit des dem Licht ausgesetzten Sensors ändert sich. Wir können den Lichtsensor durch Programmierung steuern und elektronische Systeme entwickeln, die die Lichtmenge beeinflussen.



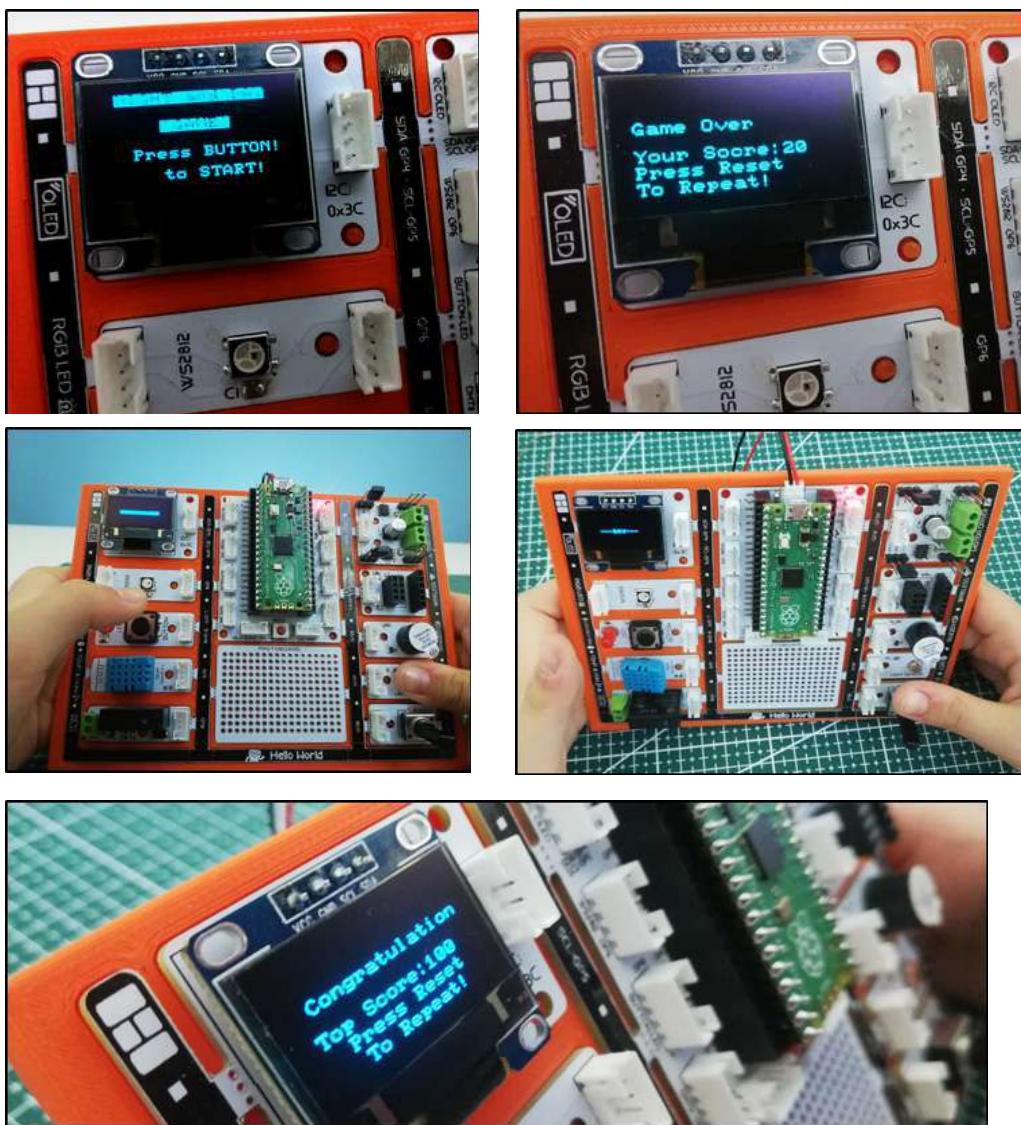
### 2.15.1. Projektdetails und Algorithmus

Zuerst werden wir den Spieler bitten, die Taste zu drücken, um das Spiel zu starten. Dann lassen wir den OLED-Bildschirm von PicoBricks abwechselnd für jeweils 2 Sekunden die Wörter NACHT und TAG anzeigen. Der Spieler sollte den LDR-Sensor innerhalb von 2 Sekunden mit seiner Hand abdecken, wenn das Wort NACHT auf dem OLED-Bildschirm geschrieben steht, und wenn das Wort TAG auf dem OLED-Bildschirm geschrieben steht, sollte der Spieler seine Hand über den LDR-Sensor heben. Jede korrekte Antwort des Spielers bringt 10 Punkte. Bei einer falschen Antwort ist das Spiel vorbei, und es wird eine schriftliche Mitteilung auf dem Bildschirm angezeigt, die das Ende des Spiels besagt, ein anderer Ton wird vom Summer ertönen, und die Punktzahl wird auf dem OLED-Bildschirm angezeigt. Wenn der Spieler insgesamt 10 korrekte Antworten gibt und 100 Punkte erreicht, wird der Satz „Herzlichen Glückwunsch“ auf dem OLED-Bildschirm angezeigt, und der Summer spielt Noten in verschiedenen Tönen.

## 2.15.2. Schaltplan



## 2.15.3. Projektbild



## 2.15.4. Projektvorschlag

Sie können das Projekt entwickeln, indem Sie die Werte, die der LDR-Sensor an das Projekt sendet, je nach Umgebung, in der Sie sich befinden, erfassen und automatisch die Grenze bestimmen, die im Spiel gemäß dem Sensorwert verarbeitet werden soll, das heißt, indem Sie Kalibrierungscodes für den LDR-Sensor hinzufügen. Sie können dem Spiel einen Schwierigkeitsgrad hinzufügen. Mit dem Potentiometer kann der Schwierigkeitsgrad als leicht, mittel und schwer ausgewählt werden.

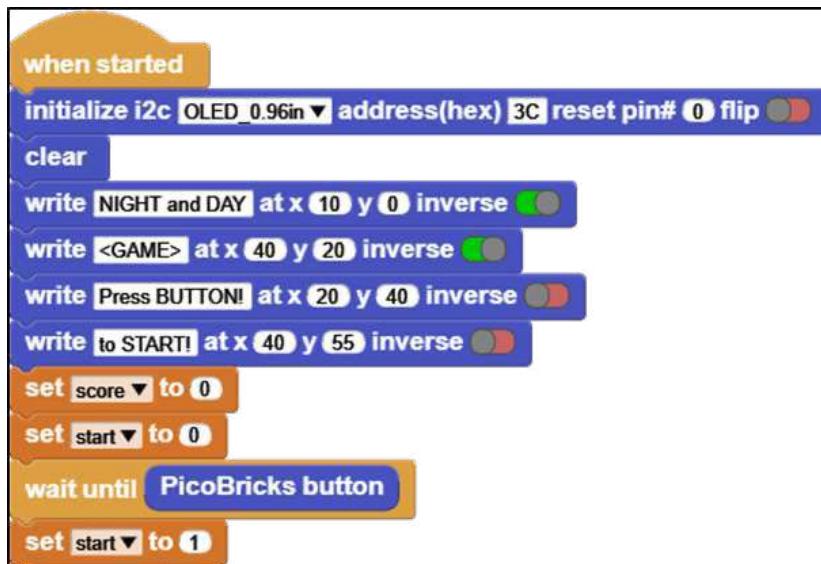
Wenn leicht ausgewählt ist, kann die Änderungszeit für die Wörter 2 Sekunden betragen, 1,5 Sekunden, wenn mittel ausgewählt ist, und 1 Sekunde, wenn schwer ausgewählt ist.

## 2.15.5. Programmierung des Projekts mit MicroBlocks

Wenn Picobricks startet, definieren wir den OLED-Bildschirm und drucken die Startbildschirmnachrichten.

Dann erstellen wir Variablen mit den Namen score, start, nightorday und gamerReaction.

Da das Spiel nach dem Drücken des Knopfes startet, wird die Startvariable nach dem Warten auf den Block auf 1 gesetzt.



Ein „Ändere Wort“ wird alle zwei Sekunden gesendet, wenn das Spiel beginnt. Dank dieser Nachricht wird der Ausdruck TAG oder NACHT auf dem Bildschirm angezeigt. Während der 2 Sekunden, in denen das Wort auf dem Bildschirm bleibt, wird die Reaktion des Spielers der Variablen gamerReaction entsprechend dem vom LDR-Sensor gelesenen Wert zugewiesen. Wenn der LDR-Wert größer als 80 ist, ist die Oberseite des Sensors nicht geöffnet, andernfalls ist der Sensor geschlossen. Sie können den Wert von 80 für Ihre eigenen Betriebsbedingungen ändern.

Am Ende der letzten zwei Sekunden werden die Sendungen „Korrekt“ und „Falsch“ gemacht, indem das zufällig bestimmte Wort (nightorday) mit gamerReaction verglichen wird. Diese Sendungen werden verwendet, um Punkte im Spiel zu verdienen und das Spiel zu beenden.

```

when start = 1
broadcast [change word v]
reset timer
repeat (20)
  if [PicoBricks light sensor] < [80]
    set [gamerReaction v] to [0]
  else
    set [gamerReaction v] to [1]
  end
  PicoBricks beep [100] ms
  if [nightorday] = [gamerReaction]
    broadcast [Correct v]
  else
    broadcast [Wrong v]
  end
end

```

Wenn eine Ändere-Wort-Nachricht in zwei Sekunden Intervallen empfangen wird, wird der Bildschirm gelöscht und der Wert 0 oder 1 wird zufällig der nightorday-Variablen zugewiesen. Wenn der Wert 0 ist, wird NACHT angezeigt, wenn er nicht 0 ist, wird TAG auf dem Bildschirm angezeigt. Damit der Broadcast-Block einmal ausgeführt wird, wird diese Befehlssequenz am Ende mit dem Befehl „Diese Aufgabe stoppen“ gestoppt.

```

when [change word] received
clear
set [nightorday v] to [random (0) to (1)]
if [nightorday] = [0]
  write [—NIGHT—] at x [20] y [30] inverse
else
  write [—DAY—] at x [25] y [30] inverse
stop this task

```

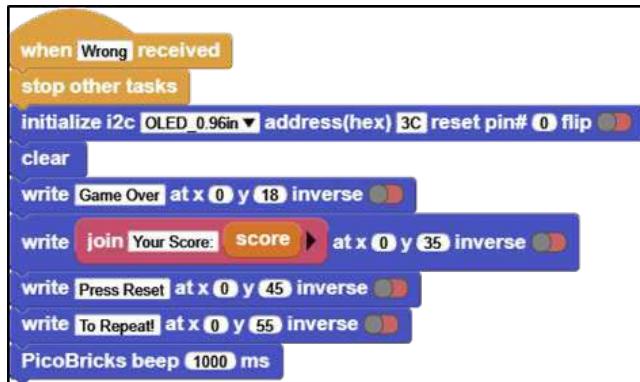
Der Codeblock, der die Codes enthält, die die score-Variable einmal um 10 erhöhen, wenn die „Korrekt“-Nachricht gesendet wird, wenn der Spieler korrekt reagiert, ist wie folgt.

```

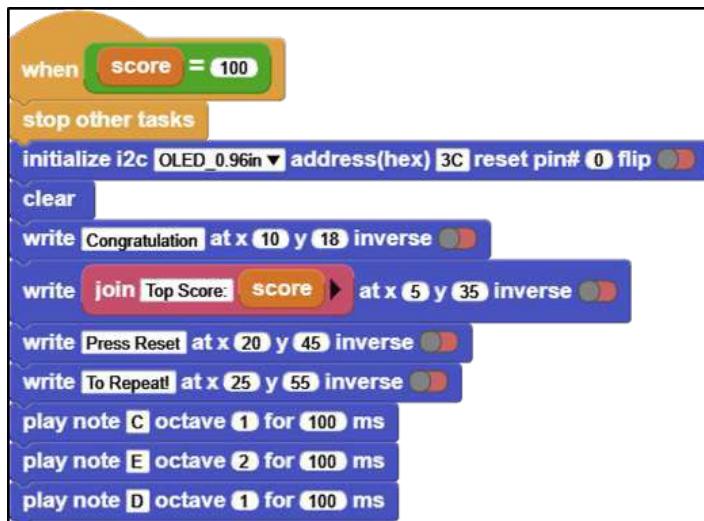
when [Correct] received
change [score v] by [10]
stop this task

```

Der Codeblock, der die Codes enthält, die alle anderen Codes für die „Falsch“-Nachricht stoppen, die gesendet wird, wenn der Spieler falsch reagiert, und den Text über das Ende des Spiels und den Punktwert auf dem Bildschirm ausgibt, ist wie folgt.



Wenn der Spieler das Spiel ohne Fehler abschließt, wird die score-Variable den Wert 100 annehmen. Der Codeblock, der dies erfasst und den Spieler gratuliert und das Spiel stoppt, ist wie folgt.



[Klicken](#) Sie, um auf die Codes des Projekts zuzugreifen.

## 2.15.6. MicroPython-Codes des Projekts

```
from machine import Pin, I2C, Timer, ADC, PWM
from picobricks import SSD1306_I2C
import utime
import urandom
#define die Bibliotheken
WIDTH = 128
HEIGHT = 64
#OLED-Bildschirmeinstellungen
```

```
sda=machine.Pin(4)
scl=machine.Pin(5)
#digitalen Pin 4 und 5 als AUSGANG für die OLED-Kommunikation initialisieren
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)
oled = SSD1306_I2C(BREITE, HÖHE, i2c)
buzzer = PWM(Pin(20))
buzzer.freq(440)
ldr=ADC(Pin(27))
button=Pin(10,Pin.IN,Pin.PULL_DOWN)
#define die Eingangs- und Ausgangspins
oled.text("NACHT und TAG", 10, 0)
oled.text("<SPIEL>", 40, 20)
oled.text("Drücke den Knopf", 0, 40)
oled.text("um ZU STARTEN!", 40, 55)
oled.show()
#OLED-Bildschirmtext-Einstellungen
def changeWord():
global nachtundtag
    oled.fill(0)
    oled.show()
nachtundtag=round(urandom.uniform(0,1))

    #wenn die Daten '0' sind, zeigt der OLED-Text NACHT an
    if nachtundtag==0:
        oled.text("---NACHT---", 20, 30)
        oled.show()
    else:
        oled.text("---TAG---", 20, 30)
        oled.show()
#wartet darauf, dass der Knopf gedrückt wird, um zu aktivieren

while button.value()==0:
print("Drücke den Knopf")
sleep(0.01)

oled.fill(0)
oled.show()
start=1
global score
score=0
while start==1:
    global gamerReaction
```

```
global score
changeWord()
startTime=utime.ticks_ms()

#wenn die Daten des LDR größer als 2000 sind, Gamerreaktion '0'
while utime.ticks_diff(utime.ticks_ms(), startTime)<=2000:
    if ldr.read_u16()>20000:
        gamerReaction=0
#wenn die LDR-Daten unter 2000 liegen, Spielerreaktion '1'
    sonst:
        gamerReaction=1
        sleep(0.01)

#Buzzer funktioniert
    buzzer.duty_u16(2000)
    sleep(0.05)
    buzzer.duty_u16(0)

    wenn gamerReaction==nightorday:
        score += 10

#wenn der Punktestand 10 ist, sagt OLED 'Game Over'
    sonst:
        oled.fill(0)
        oled.show()
        oled.text("Game Over", 0, 18, 1)
        oled.text("Ihr Punktestand " + str(score), 0,35)
        oled.text("Drücken Sie RESET",0, 45)
        oled.text("Um zu WIEDERHOLEN",0,55)
        oled.show()
        buzzer.duty_u16(2000)
        sleep(0.05)
        buzzer.duty_u16(0)
        break;

    if score==100:
#wenn der Punktestand 10 ist, sagt das OLED 'You Won'
        oled.fill(0)
        oled.show()
        oled.text("Herzlichen Glückwunsch", 10, 10)
        oled.text("Höchster Punktestand: 100", 5, 35)
        oled.text("Drücken Sie Zurücksetzen", 20, 45)
        oled.text("Um ZU WIEDERHOLEN", 25,55)
        oled.show()
        buzzer.duty_u16(2000)
        sleep(0.1)
```

```
buzzer.duty_u16(0)
sleep(0.1)
buzzer.duty_u16(2000)
sleep(0.1)
buzzer.duty_u16(0)

break;
```

## 2.15.7. Arduino C Codes des Projekts

```
#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"

// Definiere die Bibliothek

#define RANDOM_SEED_PIN    28
int Gamer_Reaction = 0;
int Night_or_Day = 0;
int Score = 0;
int counter=0;

double currentTime = 0;
double lastTime = 0;
double getLastTime(){
    return currentTime = millis()/1000.0 - lastTime;
}

void _delay(float seconds) {
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime) _loop();
}

void _loop() {
}

void loop() {
    _loop();
}

void setup() {

// put your setup code here, to run once
```

```
pinMode(10,Eingang);
pinMode(27,Eingang);
pinMode(20,Ausgang);
randomSeed(RANDOM_SEED_PIN);
Wire.begin();
oled.init();
oled.clearDisplay();
```

```
oled.clearDisplay();
oled.setTextXY(1,3);
oled.putString("NACHT und TAG");
oled.setTextXY(2,7);
oled.putString("SPIEL");
oled.setTextXY(5,2);
oled.putString("Drücke die TASTE!");
oled.setTextXY(6,4);
oled.putString("to START!");
```

```
Score = 0;
```

```
while(!(digitalRead(10) == 1))
{
    _loop();
}
_delay(0.2);
```

```
while(1){ //while loop
    if (counter==0){

        delay(500);
        Change_Word();
        lastTime = millis()/1000.0;
```

```
while(!(getLastTime() > 2))
{
    Serial.println(analogRead(27));
    if(analogRead(27) > 500){
        Gamer_Reaction = 0;
    }else{
```

```
Gamer_Reaction = 1;
}
}

//bestimme die Reaktion des Spielers basierend auf dem Wert des LDR-Sensors
digitalWrite(20,HIGH); //schalte den Summer ein
delay(250);
digitalWrite(20,LOW); //Buzzer ausschalten

if(Night_or_Day == Gamer_Reaction){
    Correct();
}

}else{
    Wrong();
}

} _loop();

if(Score==100){
    oled.clearDisplay();
    oled.setTextXY(1,1);
    oled.putString("Herzlichen Glückwunsch");
    oled.setTextXY(3,1);
    oled.putString("Ihr Punktestand: ");
    oled.setTextXY(3,13);
    String String_Score=String(Score);
    oled.putString(String_Score);
    oled.setTextXY(5,3);
    oled.putString("Drücken Sie Zurücksetzen");
    oled.setTextXY(6,3);
    oled.putString("Um Wiederholen!");

    // Schreiben Sie die "Herzlichen Glückwunsch, Ihr Punktestand, drücken Sie Zurücksetzen, um zu wiederholen!" und die Punktestand-Variable
    // an den auf dem OLED-Bildschirm bestimmten x- und y-Koordinaten

    for(int i=0;i<3;i++){
        digitalWrite(20,HIGH);
        delay(500);
        digitalWrite(20,LOW);
        delay(500);
    } counter=1; // Schalten Sie den Summer drei Mal ein und aus
}
}
}
}
```

```
void Correct (){
    Score += 10;
    oled.clearDisplay();
    oled.setTextXY(3,4);
    oled.putString("10 Punkte");
    //Erhöhe den Punktestand um 10, wenn der Spieler richtig antwortet
}

void Change_Word (){
    oled.clearDisplay();
    Night_or_Day=random(0,2);

    if (Night_or_Day==0){
        oled.setTextXY(3,6);
        oled.putString("NACHT");
    }else{
        oled.setTextXY(3,7);
        oled.putString("TAG");
    }
}
//Schreibe "NACHT" oder "TAG" auf den zufälligen OLED-Bildschirm
void Wrong (){
    oled.clearDisplay();
    oled.setTextXY(1,3);
    oled.putString("Spiel Vorbei");
    oled.setTextXY(3,1);
    oled.putString("Ihr Punktestand: ");
    oled.setTextXY(3,13);
    String String_Score=String(Score);
    oled.putString(String_Score);
    oled.setTextXY(5,3);
    oled.putString("Drücken Sie Zurücksetzen");
    oled.setTextXY(6,3);
    oled.putString("Um Wiederholen!");
}

// Schreiben Sie die Punktzahlvariable und die Ausdrücke in Anführungszeichen an die Koordinaten, die auf dem OLED-Bildschirm festgelegt sind.
digitalWrite(20,HIGH);
delay(1000);
digitalWrite(20,LOW);
counter=1;
}
```

---

## GitHub Nacht- und Tag-Projektseite



<http://rbt.ist/nightday>

## 2.16. Sprachgesteuertes Robotauto

Die Entwicklung und kontinuierliche Entwicklung von Anwendungen der künstlichen Intelligenz, die menschliche Eigenschaften erkennen, lernen und versuchen, sich wie Menschen zu verhalten. Wir können künstliche Intelligenz in ihrer kürzesten Form als Software ausdrücken, die lernen kann. Manchmal lernt sie das Bild, manchmal den Klang und manchmal, indem sie die Daten nutzt, die sie von den Sensoren sammelt. Dies geschieht dank der von den Entwicklern festgelegten Algorithmen, und sie hilft in den Entscheidungsprozessen in den Bereichen, in denen sie eingesetzt wird, basierend auf den Ergebnissen, die sie erzielt hat. Kurz gesagt, Anwendungen der künstlichen Intelligenz werden jetzt in Situationen eingesetzt, in denen der Entscheidungsprozess schnell und fehlerfrei durchgeführt werden muss. Vom Marketingbereich bis zur Verteidigungsindustrie, von der Bildung bis zur Gesundheit, von der Wirtschaft bis zur Unterhaltung erhöht die künstliche Intelligenz die Effizienz und senkt die Kosten.

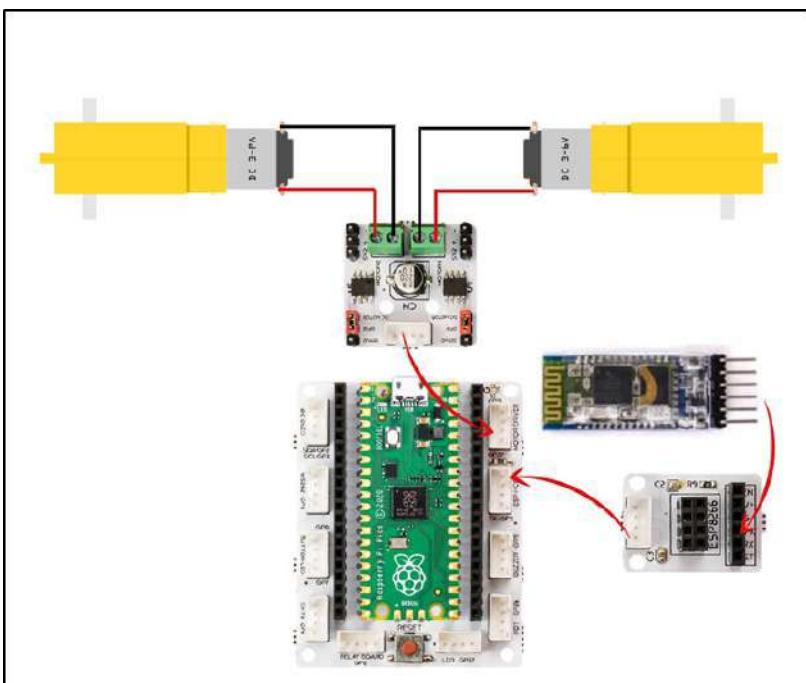
In diesem Projekt, das wir mit PicoBricks durchführen werden, werden wir ein 2WD-Auto bauen, das Sie durch Sprechen steuern können. PicoBricks ermöglicht es Ihnen, drahtlos mit 2 6V DC

Motoren und Bluetooth zu kommunizieren.

### 2.16.1. Projektdetails und Algorithmus

Im Projekt wird das Roboterauto-Kit, das aus dem Set stammt, zusammengebaut und über das Mobiltelefon gesteuert. Das HC05-Bluetooth-Modul ist ein Modul, das es uns ermöglicht, drahtlos zwischen PicoBricks und einem Mobiltelefon zu kommunizieren. Dank der Mobilanwendung, die im Projekt auf dem Mobiltelefon installiert ist, werden die Befehle, die vom Telefon gesendet werden, über das HC05-Modul an die PicoBricks übertragen, und das Roboterauto wird sich gemäß diesen Daten bewegen. Wir können das Roboterauto mit den Vorwärts-, Rückwärts-, Rechts- und Links-Tasten vom Mobiltelefon steuern und auch Daten mit Sprachbefehlen an die PicoBricks senden. Im Projekt werden wir Sprachbefehle geben, um die Bewegungen des Roboterautos zu steuern.

### 2.16.2. Verdrahtungsdiagramm



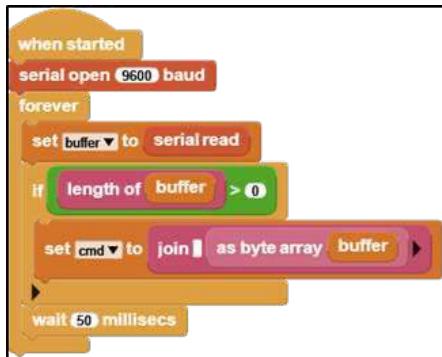
## 2.16.3. Projektvorschlag

In diesem Projekt haben wir das Roboterauto gesteuert, indem wir Sprachbefehle über die mobile Anwendung gegeben haben, die wir auf dem Mobiltelefon installiert haben. Sie können den Mechanismus mit Sprachbefehlen oder Tasten steuern, indem Sie das HC05-Bluetooth-Modul mit dem Pan-Tilt-Mechanismus im Zwei-Achsen-Roboterarm-Projekt verbinden. Ebenso können Sie eine mobile Anwendung ausprobieren, mit der Sie das Roboterauto in diesem Projekt mit Tasten anstelle von Sprachbefehlen steuern können, oder Sie können eine mobile Anwendung speziell für Ihr Projekt mit dem MIT Appinventor-Editor entwickeln.

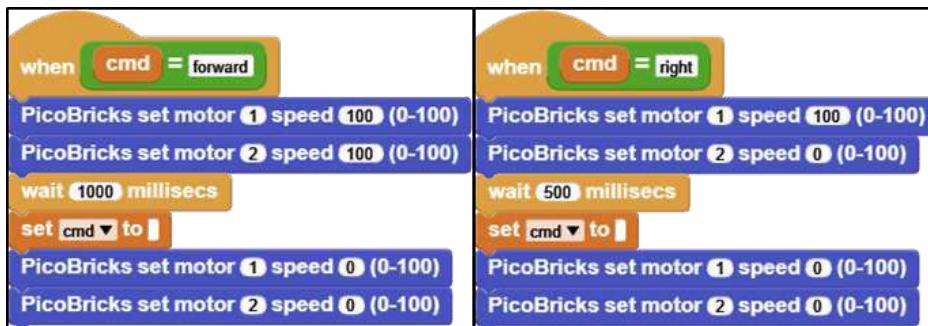
Mit dem HC05-Bluetooth-Modul können Sie nicht nur den Motorantrieb und Motor steuern, sondern auch andere Module auf PicoBricks. Zum Beispiel können Sie die RGB LED in jeder gewünschten Farbe über die mobile Anwendung einschalten, die Temperatur- und Feuchtigkeitswerte vom DHT11-Modul, die Lichtwerte am LDR-Sensor ablesen und Texte auf dem OLED-Bildschirm ausgeben. Es gibt eine mobile Anwendung, die speziell für diese Prozesse mit dem MIT Appinventor-Editor geschrieben wurde, sowie fertige Codes, die mit Microblocks geschrieben wurden, um die von der Anwendung kommenden Daten automatisch auszuführen. Sie können alle diese Funktionen nutzen, indem Sie die Microblocks-Datei von dem untenstehenden Link herunterladen und ausführen und die Android-APK-Datei herunterladen und auf Ihrem Telefon installieren.

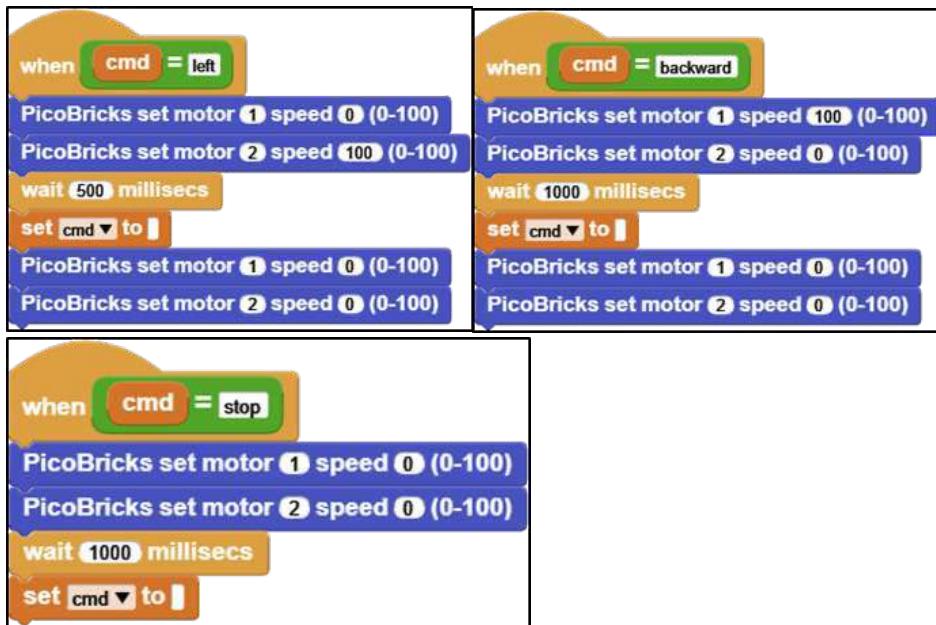
### Download-Link

## 2.16.4. Programmierung des Projekts mit MicroBlocks



Nachdem wir die Montage des Projekts abgeschlossen haben, erstellen wir zuerst zwei Variablen namens cmd und buffer im Code-Teil und lesen den seriellen Port, der auf 9600 Baudrate definiert ist, mit der buffer-Variablen und übertragen die eingehenden Informationen in die cmd-Variablen. Wenn Sie alle Codes geschrieben haben und keine Kommunikation zwischen dem Telefon und PicoBricks besteht, können Sie versuchen, die Baudrate von 9600 auf 115200 zu ändern.



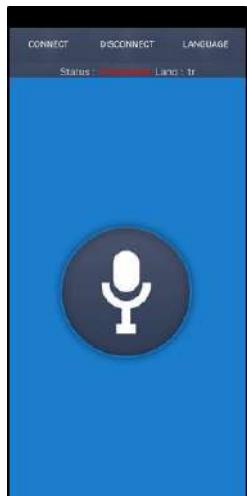


[Klicken Sie](#) hier, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

Mit dem HC05-Modul führen wir Vergleichsoperationen mit der cmd-Variablen durch, die die Informationen über den seriellen Port speichert, und wenn Blöcke. Wenn die eingehenden Informationen „vorwärts“ sind, müssen wir die motor1- und motor2-Blöcke zwischen 0 und 100 Werten ausführen, damit das Roboterauto vorwärts fahren kann. Die Geschwindigkeit Ihres Fahrzeugs ändert sich proportional zur Füllung Ihrer Batterie. Ebenso schreiben wir die notwendigen Codes, damit das Fahrzeug nach rechts abbiegt, wenn die über Bluetooth empfangenen Informationen „rechts“ sind, nach links abbiegt, wenn es „links“ ist, zurückfährt, wenn es „rückwärts“ ist, und das Fahrzeug stoppt, wenn es „stop“ ist. In diesen Codes wird dem Fahrzeug ermöglicht, Bewegungen für einen bestimmten Zeitraum auszuführen. Sie können die Fristen verlängern oder entfernen, wenn Sie möchten.

Nachdem wir die Codes geschrieben und auf PicoBricks ausgeführt haben, laden wir die mobile Anwendung für Android-Geräte von dem untenstehenden Link herunter und installieren sie auf dem Telefon.

[Download-Link](#)

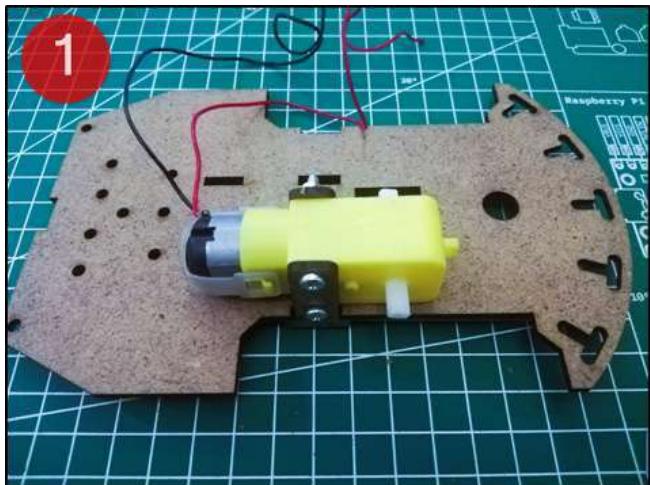


Sie können die Spracheinstellungen mit der Schaltfläche Sprache öffnen und die Sprache auf Englisch mit der Option „en“ und die Sprache auf Türkisch mit der Option „tr“ ändern.

## 2.16.5. Bauphasen des Projekts

1. Schrauben Sie den ersten Motor an das Chassis des 2WD-Roboterautos, das aus dem Set kommt

und befestigen Sie ihn.



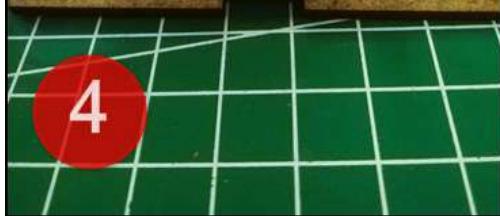
2. Befestigen Sie den zweiten Motor, indem Sie ihn am Chassis festschrauben.

3. Bringen Sie die Räder an den Motoren an.



4. Befestigen Sie das Rad unter dem Chassis mit Abstandshaltern.

5. Fixieren Sie den Abstandshalter mit der Mutter von oben am Chassis.



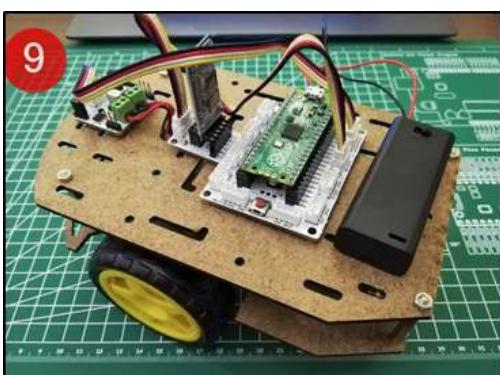
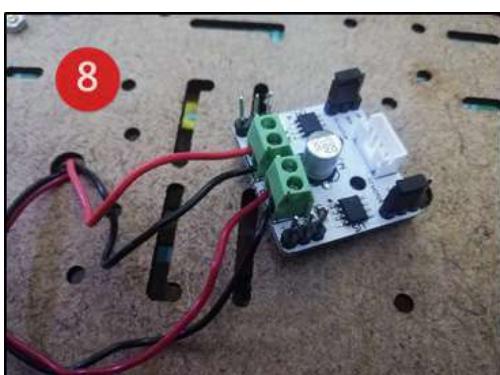
**6.** Fixieren Sie 4 Abstandshalter an den vier Ecken des unteren Chassis.

**7.** Fixieren Sie das obere Chassis mit Stecker und Muttern.



**8.** Verbinden Sie die Kabel der Motoren mit den Anschlüssen am Motorsteuergerät.

**9.** Fixieren Sie das Motorsteuergerät, das Bluetooth-Modul, die Picobricks-Platine und die Batteriekiste am Chassis mit heißem Silikon.



## 2.16.6. MicroPython-Codes des Projekts

```
from machine import Pin, UART
from utime import sleep

uart = UART(0,9600) #Wenn die Verbindung nicht hergestellt werden kann, versuchen Sie 115200.
m1 = Pin(21, Pin.OUT)
m2 = Pin(22, Pin.OUT)

m1.low()
m2.low()

while True:
    sleep(0.05)
    if uart.any():
        cmd = uart.readline()
    if cmd==b'F':
        m1.high()
        m2.high()
    elif cmd==b'R':
        m1.high()
        m2.low()
    elif cmd==b'L':
        m1.low()
        m2.high()
    elif cmd==b'S':
        m1.low()
        m2.low()
    cmd=""
```

## 2.16.7. Arduino C-Code des Projekts

```
void setup() {
  Serial1.begin(9600);
}

void loop() {
  if (Serial1.available() > 0) {

    char sread = Serial1.read();
    Serial.println(sread);
```

```
if (sread == 'f') {
    Forward();
} else if(sread == 'r'){
    Turn_Right();
} else if(sread == 'l'){
    Turn_Left();
} else if(sread == 's'){
    Stop();
}

void Forward(){
    digitalWrite(21,HIGH);
    digitalWrite(22,HIGH);
    delay(1000);
    digitalWrite(21,LOW);
    digitalWrite(22,LOW);
}

void Turn_Left(){
    digitalWrite(21,LOW);
    digitalWrite(22,HIGH);
    delay(500);
    digitalWrite(21,LOW);
    digitalWrite(22,LOW);
}

void Turn_Right(){
    digitalWrite(21,HIGH);
    digitalWrite(22,LOW);
    delay(500);
    digitalWrite(21,LOW);
    digitalWrite(22,LOW);
}

void Stop(){
    digitalWrite(21,LOW);
    digitalWrite(22,LOW);
    delay(1000);
}
```

Nachdem Sie die Arduino-Codes hochgeladen haben, laden Sie die Android-Anwendung von dem Link unten herunter, öffnen Sie den Terminalmodus und verwenden Sie die Buchstaben f, b, r, l, s für die Bewegungen des Fahrzeugs.

[Link](#)

GitHub-Seite des Sprachgesteuerten Roboters



<http://rbt.ist/voicecar>

## 2.17. Zwei-Achsen-Roboterarm

Roboterarme haben die menschliche Kraft im industriellen Bereich ersetzt. In Fabriken übernehmen Roboterarme die Aufgaben des Tragens und Drehens von Lasten, die von einem Menschen nicht getragen werden können. Die Fähigkeit, mit einer Präzision von einem Tausendstel Millimeter positioniert zu werden, übersteigt die Sensibilität, die eine menschliche Hand zeigen kann. Wenn Sie die Produktionsvideos von Automobilfabriken ansehen, werden Sie sehen, wie wichtig die Roboterarme sind. Der Grund, warum sie Roboter genannt werden, ist, dass sie programmiert werden können, um die gleiche Arbeit mit endlosen Wiederholungen zu verrichten. Der Grund, warum es einen Arm genannt wird, ist, weil es eine gegliederte Struktur wie unsere Arme hat. Wie viele verschiedene Richtungen ein Roboterarm drehen und bewegen kann, wird als Achsen ausgedrückt. Roboterarme werden auch zum Schnitzen und Formen von Aluminium und verschiedenen Metallen verwendet. Diese Geräte, die als 7-Achsen-CNC-Router bezeichnet werden, können Metalle formen, wie ein Bildhauer Ton formt.

Je nach Verwendungszweck in Roboterarmen werden Schrittmotoren und Servomotoren, die eine Art Elektromotor sind, verwendet. PicoBricks ermöglicht es Ihnen, Projekte mit Servomotoren zu erstellen.

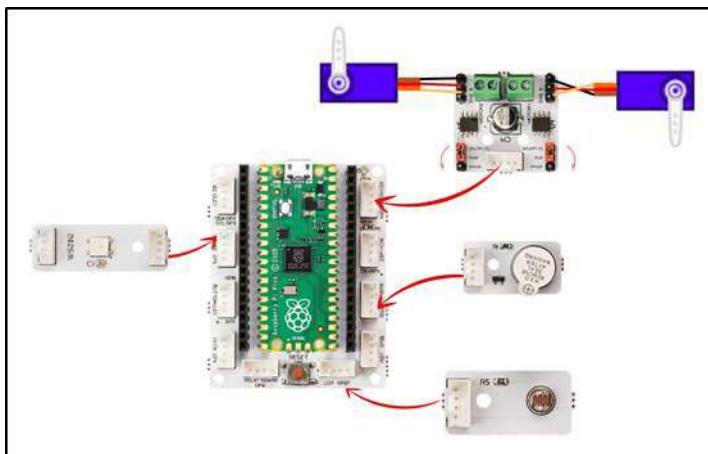
### 2.17.1. Projektdetails und Algorithmus

Zur Vorbereitung der Installation werden wir zunächst die Codes schreiben und hochladen, um die Servomotoren auf 0 Grad einzustellen. Wenn ein Objekt auf den LDR-Sensor gelegt wird, wird der Roboterarm herunterbeugen und seinen offenen Greifer schließen. Nachdem der Greifer geschlossen ist, wird der Roboterarm wieder ansteigen. Bei jeder Bewegung des Roboterarms wird ein kurzer Piepton vom Summer zu hören sein. Die RGB-LED leuchtet rot, wenn ein Objekt auf dem LDR-Sensor platziert wird. Wenn das Objekt vom Roboterarm gehalten und in die Luft gehoben wird, wird die RGB-LED grün leuchten.

Die Bewegungen des Servomotors sind sehr schnell. Um die Bewegung zu verlangsamen, werden wir die Servomotoren mit insgesamt 90 Grad Bewegung codieren, jeweils 2 Grad in 30 Millisekunden-Intervallen. Dies werden wir nicht für das Schließen des Greifers tun.

Damit der Servomotor seine Halte- und Freigabefunktion ausführen kann, drucken und montieren Sie die notwendigen Teile mit dem [Link hier.](#)

## 2.17.2. Schaltplan



## 2.17.3. Projektvorschlag

Durch das Hinzufügen des HC05-Moduls zum 2-Achsen-Roboterarm-Projekt können Sie es entwickeln, indem Sie es von Ihrem Mobiltelefon mit der mobilen Anwendung steuern.

## 2.17.4. Bauphasen des Projekts

Bereiten Sie die Teile des Pan-Tilt-Kits vor, um das Projekt vorzubereiten. Nehmen Sie Ihre 3D-gedruckten Teile, Abfallkartonstücke, heiße Silikonkleber und Scheren mit.



1. Zunächst werden wir den festen Arm des Roboterarms vorbereiten. Machen Sie einen 8 cm hohen Kartonzyylinder in den runden Teil von Teil D. Platzieren Sie ihn auf dem Teil D und kleben Sie ihn mit Silikon fest.



2. Platzieren Sie den Kopf, der aus dem Servomotor-Paket stammt, auf dem Teil C, indem Sie ihn ein wenig kürzen. Befestigen Sie ihn mit den kleinsten Schrauben aus dem Pan-Tilt-Kit.

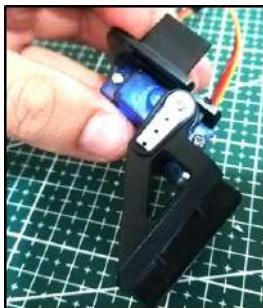


3. Befestigen Sie die Teile A und C mit 2 spitzen Schrauben zusammen.

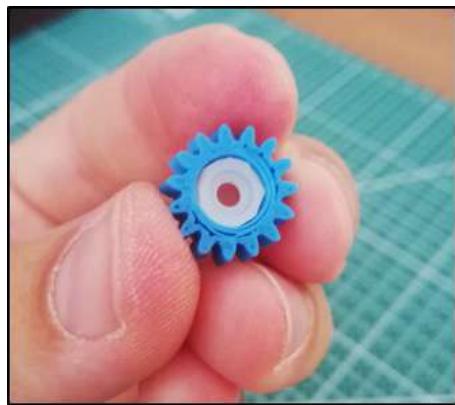


4. Befestigen Sie den Servomotor intern an Teil C. Platzieren Sie dann den Servomotor auf Teil B

und schrauben Sie ihn fest.



5. Für den Halter schneiden Sie einen der Servomotorenköpfe in der Mitte des Zahnradteils , das Sie mit dem 3D-Drucker gedruckt haben, und setzen Sie ihn in das Zahnrad ein. Schrauben Sie ihn dann an den Servomotor.



6. Kleben Sie das 3D-gedruckte Linearzahnrad und den Griff mit starkem Kleber zusammen.



7. Platzieren Sie den Servomotor im 3D-Druckhalter und fixieren Sie ihn. Dies können Sie mit heißem Silikon oder durch Schrauben tun. Achten Sie darauf, dass das Servozahnrad beim Platzieren auf dem Linearzahnrad vollständig geöffnet ist.



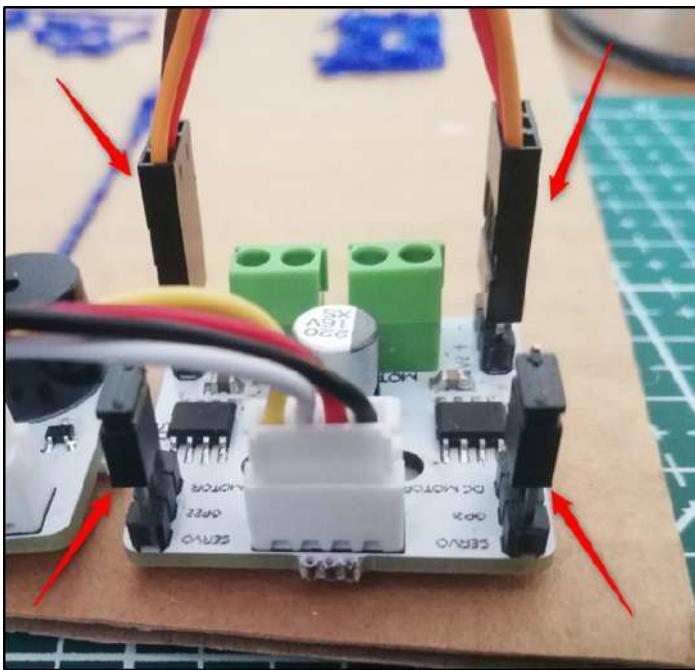
8. Kleben Sie das Haltesystem des Servos mit Silikon an Teil B.



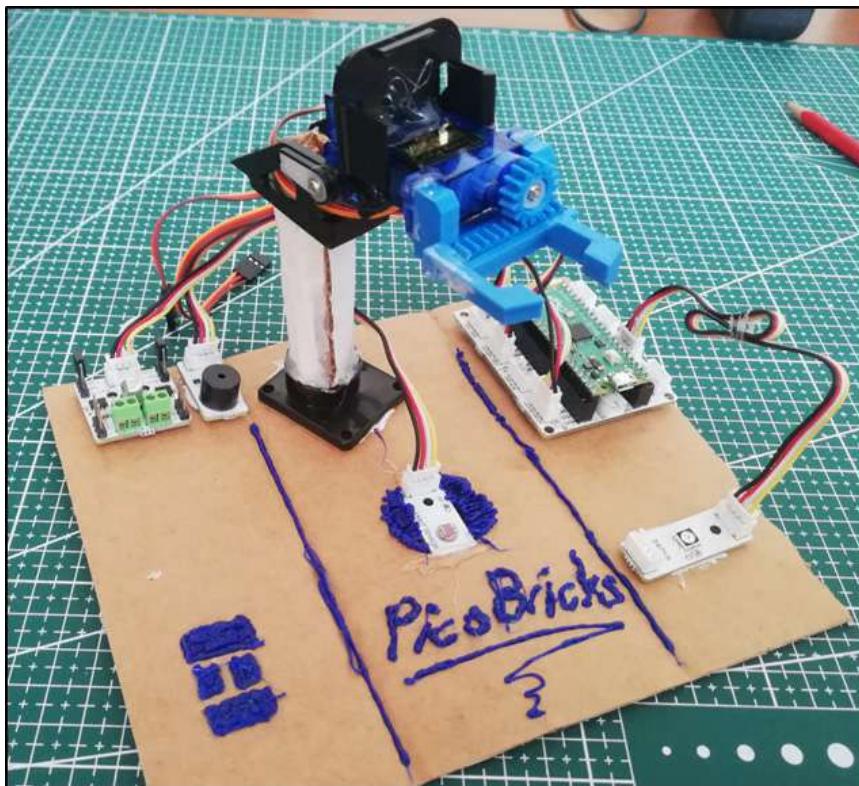
9. Führen Sie das Stück, das wir in Schritt 3 vorbereitet haben, über den Zylinder, den wir im ersten Schritt aus Pappe vorbereitet haben, und fixieren Sie es mit Silikon.



10. Setzen Sie die Motorantriebsbrücken auf die Servo-Pins. Verbinden Sie das Kabel des Halteservos mit GPIO21 und das Kabel des Neigservos mit GPIO22.



11. Platzieren Sie den Motorcontroller, den Summer, den LDR und das RGB-LED-Modul auf einer Plattform und platzieren Sie den Roboterarm entsprechend auf der Plattform. Mit dem 3D-Stift können Sie Ihr Projekt nach Belieben anpassen.



12. Sie können den Roboterarm betreiben, wenn Sie Picobricks über USB oder 3 Stiftbatterien von der Strombuchse am Picoboard speisen.

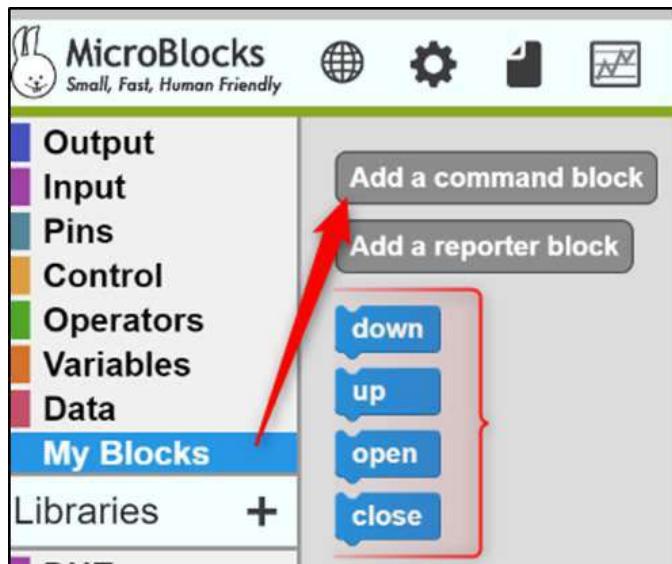
## 2.17.5. Codierung des Projekts mit MicroBlocks

Öffnen Sie die Microblocks und verbinden Sie sie mit den Picobricks. Fügen Sie die Servo-Bibliothek hinzu. Schließen Sie den

Haltmotor an Pin 21 und den Biegemotor an Pin 22 an. Setzen Sie den Servowert des Griffendes des Roboterarms auf -90. Setzen Sie den Winkel des Neigermotors auf 0. Wenn Sie die unten Blöcke vorbereiten und darauf klicken, werden die Codes ausgeführt und die Servos bewegen sich auf den von Ihnen festgelegten Winkel.

```
set servo 22 to 0 degrees (-90 to 90)
set servo 21 to -90 degrees (-90 to 90)
```

Wir werden die Bewegungen des Servomotors in separaten Blöcken vorbereiten. Erstellen Sie dazu vier Befehlsblöcke mit den Namen hoch, runter, öffnen, schließen aus der Kategorie Meine Blöcke.



Für die Armbewegung nach oben und unten codieren Sie die hoch- und runter-Blöcke für 45 Wiederholungen, um jeweils 2 Grad in 30 Millisekunden zu rotieren. Um den Winkelwert zu ändern, erstellen Sie eine Variable namens angleupdown. Im runter-Block gibt es eine Änderung von -2 Grad und im hoch-Block von 2 Grad.

```
define up
repeat (45)
  change angleupdown by 2
  set servo 22 to angleupdown degrees (-90 to 90)
  wait (30) millisecs
end
play note C octave 1 for 100 ms

define down
repeat (45)
  change angleupdown by -2
  set servo 22 to angleupdown degrees (-90 to 90)
  wait (30) millisecs
end
play note C octave 1 for 100 ms
```

Um den Greifer-Servo einzuschalten, codieren Sie seinen Winkel auf 90, um ihn auszuschalten, codieren Sie seinen Winkel auf -60. Fügen Sie einen Ton Ihrer Wahl hinzu, indem Sie die Tone-Bibliothek am Ende aller Servobewegungen hinzufügen. 100 ms Klingeln sind ausreichend.

```
define open
set servo 21 to 90 degrees (-90 to 90)
play note D octave 2 for 100 ms

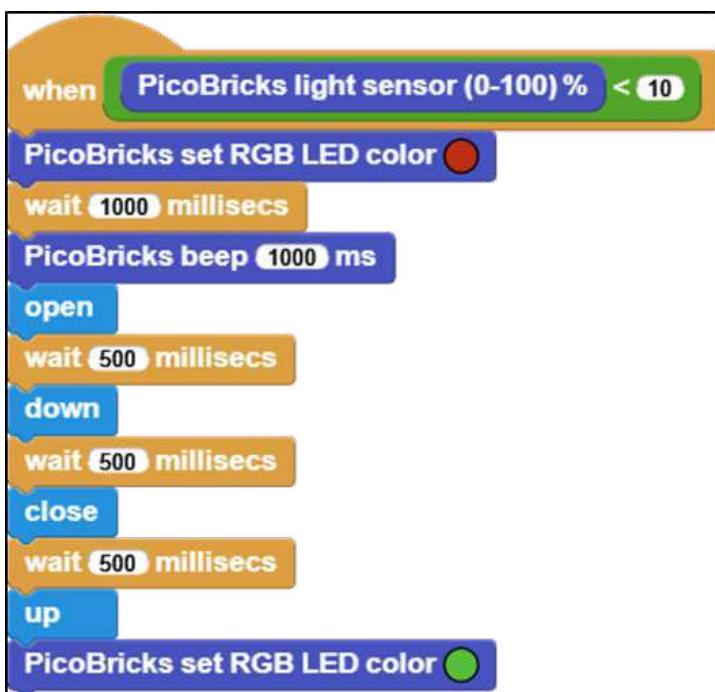
define close
set servo 21 to -60 degrees (-90 to 90)
play note D octave 2 for 100 ms
```

Wenn die PicoBricks starten, sollte der Greifer-Servo eingeschaltet werden, die angleupdown Variable sollte auf ihren Anfangswert von 90 gesetzt werden, und der Roboterarm sollte sich um 90 Grad nach oben drehen. Die RGB-LED sollte ausgehen.



Wir erkennen, dass sich ein Objekt auf dem LDR-Sensor befindet, wenn sein Wert unter 10 fällt. Dieser Wert kann für Ihr Projekt unterschiedlich sein. Sie können herausfinden, wie viele Messungen durchgeführt wurden, indem Sie auf den Block klicken. Zuerst leuchtet die RGB-LED rot. Um das Objekt zu halten und anzuheben, wird der Haltmotor geöffnet, die Abwärtsbewegung wird durchgeführt, der Haltmotor wird geschlossen und die Aufwärtsbewegung wird realisiert. Schließlich leuchtet die RGB-LED grün.  
Erleichtern Sie die Beobachtung der Bewegung, indem Sie zwischen jeder

Bewegung eine Pause von 500 ms einlegen.



[Klicken](#) Sie, um auf die Codes des Projekts zuzugreifen.

## 2.17.6. MicroPython-Codes des Projekts

Codes, die Servomotoren kalibrieren:

```
from machine import Pin, PWM,  
servo1=PWM(Pin(21))  
servo2=PWM(Pin(22))
```

```
servo1.freq(50)  
servo2.freq(50)
```

```
servo1.duty_u16(8200) # 180 Grad  
servo2.duty_u16(4770) # 90 Grad
```

Projektcodes:

```
from machine import Pin, PWM, ADC  
from utime import sleep  
from picobricks import WS2812  
#define libraries  
  
ws = WS2812(6, brightness=0.3)  
ldr=ADC(27)  
buzzer=PWM(Pin(20, Pin.OUT))  
servo1=PWM(Pin(21))  
servo2=PWM(Pin(22))  
# define LDR, Buzzer und Servomotoren Pins  
  
servo1.freq(50)  
servo2.freq(50)  
buzzer.freq(440)  
# define frequencies of servo motors and buzzer  
  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
BLACK = (0, 0, 0) # RGB-Farbeinstellungen  
angleupdown=4770  
angleupdown2=8200  
  
def up():  
    global angleupdown
```

```
for i in range (45):
    angleupdown +=76
    servo2.duty_u16(angleupdown)
    sleep(0.03)
    buzzer.duty_u16(2000)
    sleep(0.1)
    buzzer.duty_u16(0)
    # servo2 geht in festgelegten Intervallen nach oben
def down():
    global angleupdown
    for i in range (45):
        angleupdown -=76
        servo2.duty_u16(angleupdown)
        sleep(0.03)
        buzzer.duty_u16(2000)
        sleep(0.1)
        buzzer.duty_u16(0)
    # servo2 geht in festgelegten Intervallen nach unten

def open():
    global angleupdown2
    for i in range (45):
        angleupdown2 +=500
        servol.duty_u16(angleupdown2)
        sleep(0.03)
        buzzer.duty_u16(2000)
        sleep(0.1)
        buzzer.duty_u16(0)
    # servol arbeitet zum Öffnen der Klemmen
def close():
    global angleupdown2
    for i in range (45):
        angleupdown2 -=500
        servol.duty_u16(angleupdown2)
        sleep(0.03)
        buzzer.duty_u16(2000)
        sleep(0.1)
        buzzer.duty_u16(0)
    # servol arbeitet zum Schließen der Klemmen
open()
servo2.duty_u16(angleupdown)
```

```

ws.pixels_fill(BLACK)
ws.pixels_show()
while True:
    if ldr.read_u16()>20000:
        ws.pixels_fill(ROT)
        ws.pixels_show()
        sleep(1)
        buzzer.duty_u16(2000)
        sleep(1)
        buzzer.duty_u16(0)
        open()
        sleep(0.5)
        down()
        sleep(0.5)
        close()
        sleep(0.5)
        up()
        ws.pixels_fill(GRÜN)
        ws.pixels_show()
        sleep(0.5)
# Laut den von LDR empfangenen Daten leuchten die RGB-LEDs rot und grün und
bewegen sich die Servomotoren

```

## 2.16.7. Arduino C-Codes des Projekts

```

##include <Adafruit_NeoPixel.h>
#ifndef __AVR__
#include <avr/power.h>
#endif
#define PIN      6
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
#define DELAYVAL 500
// erforderliche Bibliotheken definieren
#include <Servo.h>
Servo myservo1;
Servo myservo2;

int angleupdown;

void setup() {

```

```
pinMode(20,OUTPUT);
pinMode(27,INPUT);
// Eingangs- und Ausgangspins definieren

pixels.begin();
pixels.clear();

myservo1.attach(21);
myservo2.attach(22); // Servomotor-Pins definieren
Open();
angleupdown=180;
myservo2.write(angleupdown);

}

void loop() {
if(analogRead(27)>150){

pixels.setPixelColor(0, pixels.Color(255, 0, 0));
pixels.show();
delay(1000);
tone(20,700);
delay(1000);
noTone(20);

Open();
delay(500);
Down();
delay(500);
Close();
delay(500);
Up();
pixels.setPixelColor(0, pixels.Color(0, 255, 0));
pixels.show();
delay(10000);
pixels.setPixelColor(0, pixels.Color(0, 0, 0));
pixels.show();
Öffnen();
angleupdown=180;
myservo2.write(angleupdown);

// Wenn die LDR-Daten den festgelegten Grenzwert überschreiten, wird der Summer ertönen, die RGB
wird rot und die Servomotoren werden arbeiten
}
```

```
// Die RGB wird grün, wenn die Bewegung abgeschlossen ist
```

```
    }  
}
```

```
void Öffnen(){  
    myservo1.write(180);  
}
```

```
void Schließen(){  
    myservo1.write(30);  
}
```

```
void Hoch(){
```

```
    for (int i=0;i<45;i++){  
  
        angleupdown = angleupdown+2;  
        myservo2.write(angleupdown);  
        delay(30);  
    }  
}
```

```
void Down(){
```

```
    for (int i=0;i<45;i++){  
  
        angleupdown = angleupdown-2;  
        myservo2.write(angleupdown);  
        delay(30);  
    }  
    delay(30);  
}
```

```
}
```

---

GitHub Zwei-Achsen-Roboterarm-Projektseite



<http://rbt.ist/robotarm>

## 2.18. Smart House

Arbeitsplätze, Fabriken, Wohnungen und sogar Tierheime... Es gibt verschiedene elektronische Systeme, die verwendet werden können, um unsere Lebensräume vor Eindringlingen zu schützen.

Diese Systeme werden als Sicherheitslösungen für Zuhause und am Arbeitsplatz produziert und vermarktet. Es gibt Systeme, bei denen die von Sicherheitskameras erzeugten Bilder verarbeitet und interpretiert werden, sowie Sicherheitssysteme, die den menschlichen Körper und seine Bewegungen mit Sensoren erkennen und Maßnahmen ergreifen. Sicherheitssysteme sind wie eine Art Alarmanlage eingerichtet und geben akustische und visuelle Warnungen aus, wenn eine nicht identifizierte Aktivität in dem festgelegten Zeitbereich erkannt wird. Es benachrichtigt das Unternehmen oder den Hauseigentümer und kann auch automatische Benachrichtigungen an die Sicherheitsdienste senden.

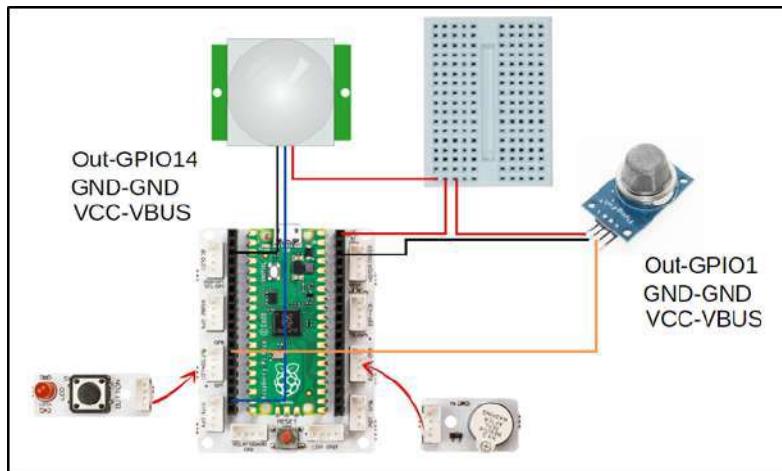
Bei Gaslecks, Bränden usw. werden in solchen Fällen Gassensoren in Wohnungen und am Arbeitsplatz eingesetzt, um Vergiftungen zu verhindern. In einer negativen Situation werden die Menschen in der Umgebung mit einem lauten Alarm gewarnt.

Wir werden ein Modellprojekt für ein intelligentes Zuhause mit PicoBricks unter Verwendung des HC-SR501 und des MQ-2-Gassensors vorbereiten. Dieser Sensor HC-SR501, auch bekannt als PIR-Sensor, erkennt Bewegungen, indem er die Veränderungen der von dem menschlichen Körper reflektierten Infrarotwellen erfasst.

### 2.18.1. Projektdetails und Algorithmus

Wenn der HC-SR501 PIR-Sensor Bewegung erkennt, gibt er für 3 Sekunden ein digitales Signal aus. Wir werden im Projekt ein Picoboard, einen Summer und ein LED-Modul verwenden. Alle Teile müssen in dem Modell enthalten sein. Wenn Picobricks startet, muss der Knopf gedrückt werden, um das Alarmsystem zu aktivieren. Nach dem Drücken des Knopfes müssen wir 3 Sekunden warten, bis die Hand aus dem Modell gezogen wird. Am Ende der 3 Sekunden leuchtet die rote LED und das Alarmsystem wird aktiviert. Wenn das Alarmsystem eine Bewegung erkennt, beginnt die rote LED zu blinken und der Summer gibt einen Alarmton von sich. Um ihn zu stummschalten, muss Picobricks neugestartet werden. Der MQ-2-Sensor ist immer eingeschaltet. Wenn er ein giftiges Gas erkennt, wird er mit einem Summer und einer roten LED warnen.

### 2.18.2. Schaltplan

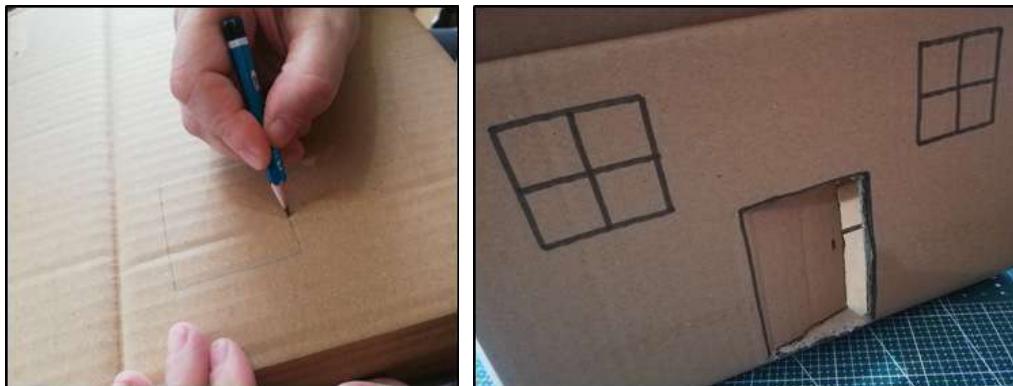


## 2.18.3. Projektvorschlag

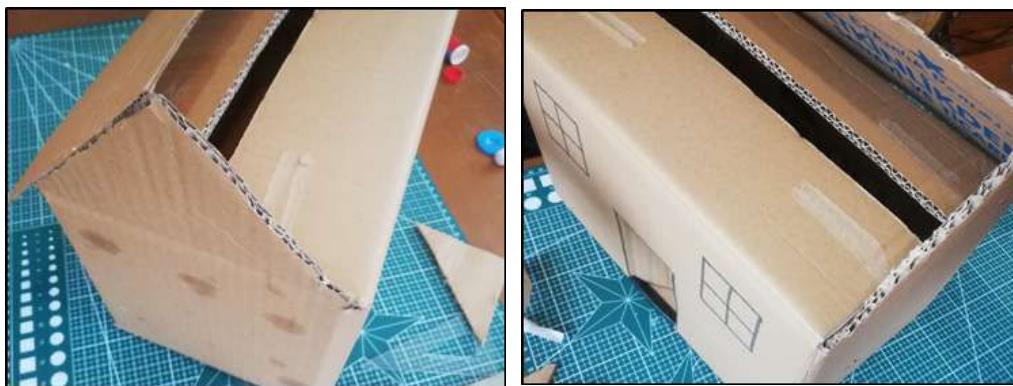
Nach dem 25. Projekt, dem smarten Gewächshaus, können Sie durch Hinzufügen des ESP8266-Moduls zum Einbruchmeldesystem eine Benachrichtigung an das Telefon des Hausbesitzers senden, wenn ein Dieb ins Haus eindringt, und das Projekt in ein IoT-Projekt verwandeln. Sie können Löschrohre an der Decke des Hauses mit einer Tauchpumpe installieren, sodass Sie im Falle eines Feuers im Haus automatisch löschen können.

## 2.18.4. Bauphasen des Projekts

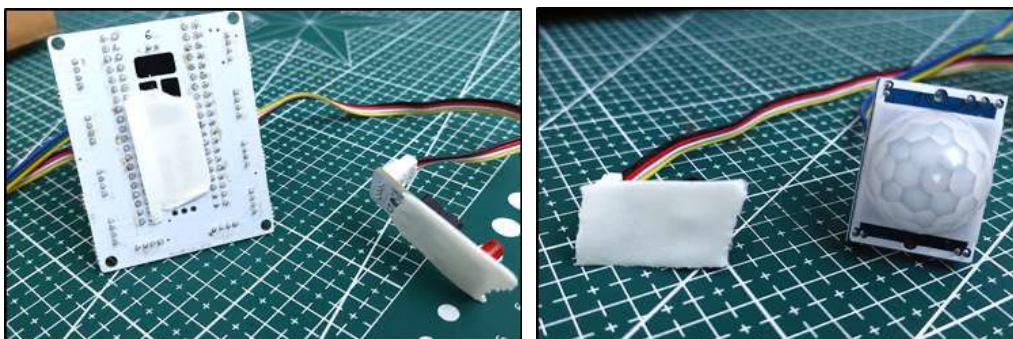
Um das Projekt auszuführen, müssen Sie einen Karton in ein Modellhaus verwandeln. Sie benötigen Scheren, Bleistifte, Klebeband, Kleber und ein Cuttermesser. Zeichnen Sie Fenster und Türen mit einem Bleistift auf den Karton. Schneiden Sie den Türbereich mit einem Cuttermesser aus.



Sie können einen anderen Karton verwenden, um den Dachbereich zu erstellen.

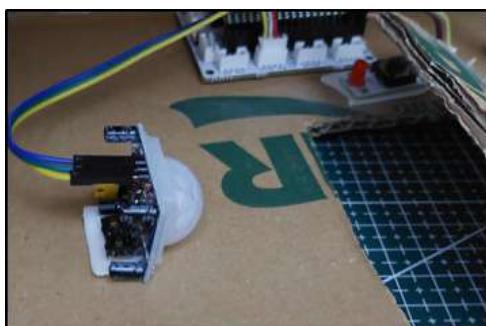


Bringen Sie doppelseitiges Schaumstoffband unter den Picobricks-Stücken an.

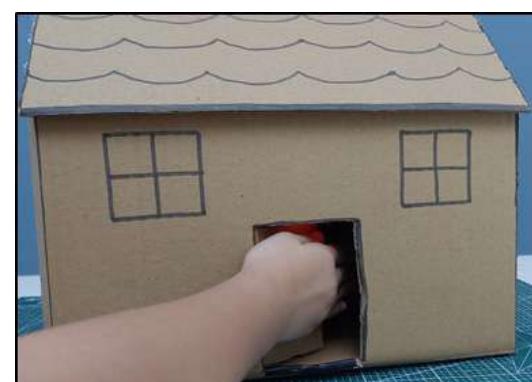
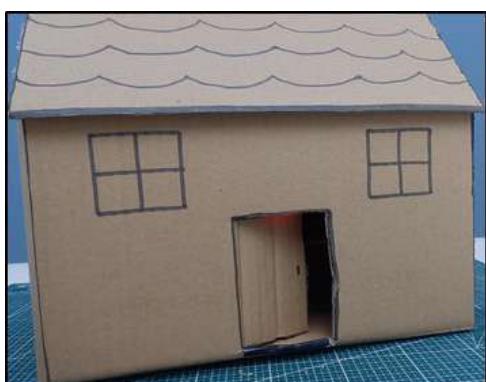


Platzieren Sie Stücke von Picobricks im Modellhaus. Positionieren Sie den PIR-Sensor so, dass er die

Tür direkt von innen sieht. Kleben Sie das Tastenmodul direkt über die Tür von der Innenseite.



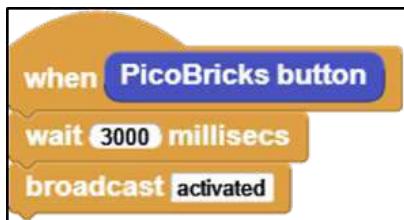
Wenn Sie das Batteriefach mit dem Picoboard verbinden und es öffnen, beginnen die Codes zu laufen. 3 Sekunden nach dem Drücken der Taste wird das Alarmsystem aktiviert und die rote LED leuchtet auf. Sobald Sie Ihre Hand in die Tür stecken, beginnt der Summer zu läuten.



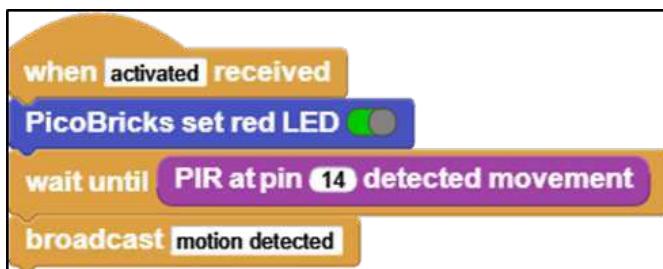
Wenn Sie das Feuerzeuggas im Haus halten, wird erwartet, dass das Alarmsystem nochmals aktiviert wird.

## 2.18.5. Programmierung des Projekts mit MicroBlocks

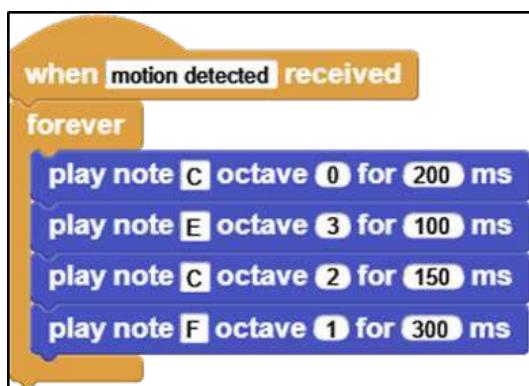
Es wartet darauf, dass die Taste gedrückt wird, um das Alarmsystem zu aktivieren. Die aktivierte Übertragung erfolgt 3 Sekunden nach dem Drücken der Taste.



Zeigen Sie an, dass das Alarmsystem aktiviert ist, indem Sie die rote LED einschalten. Gehen Sie zu Bibliothek>Sensing>PIR.ulb und fügen Sie den Block des PIR-Sensors zu MicroBlocks hinzu. Ändern Sie die Pin-Nummer auf 14. Halten Sie die Codeausführung an, bis Bewegung erkannt wird. Übertragen Sie Bewegung erkannt, wenn Bewegung erkannt wird.



Wenn die Übertragung „Bewegung erkannt“ empfangen wird, wird der Summer kontinuierlich einen Alarm geben. Sie können den Alarmton nach Belieben anpassen, indem Sie den Ton hinzufügen. ubl-Bibliothek.



[Klicken](#) Sie, um auf die Codes des Projekts zuzugreifen.

## 2.18.6. MicroPython-Codes des Projekts

```
from machine import Pin, PWM
from utime import sleep
# define libraries
PIR=Pin(14, Pin.IN)
MQ2=Pin(1,Pin.IN)
buzzer=PWM(Pin(20,Pin.OUT))
```

```
redLed=Pin(7,Pin.OUT)
button=Pin(10,Pin.IN,Pin.PULL_DOWN)
# define output and input pins

activated=0
gas=0

while True:
    if button.value()==1:
        activated=1
        gas=0
        sleep(3)
        redLed.value(1)
        buzzer.duty_u16(0)
    if MQ2.value()==1:
        gas=1
    if activated==1:
        if PIR.value()==1:
            buzzer.duty_u16(6000)
            buzzer.freq(440)
            sleep(0.2)
            buzzer.freq(330)
            sleep(0.1)
            buzzer.freq(494)
            sleep(0.15)
            buzzer.freq(523)
            sleep(0.3)
    if gas==1:
        buzzer.duty_u16(6000)
        buzzer.freq(330)
        sleep(0.5)
        redLed.value(1)
        buzzer.freq(523)
        sleep(0.5)
        redLed.value(0)

# LED wird leuchten und der Buzzer wird ertönen, wenn der PIR Bewegung erkennt oder MQ2
toxisches Gas erkennt
```

## 2.18.7. Arduino C Codes des Projekts

```
void actived (){
    digitalWrite(7,1);
    while(!(digitalRead(14) == 1))
    {
        _loop();
    }
    motion_detected();
}

void motion_detected (){
    while(1) {
// Buzzer-Einstellungen
        tone(20,262,0.25*1000);
        delay(0.25*1000);
        tone(20,330,0.25*1000);
        delay(0.25*1000);
        tone(20,262,0.25*1000);
        delay(0.25*1000);
        tone(20,349,0.25*1000);
        delay(0.25*1000);
    }
    // Den Buzzer ertönen lassen, wenn PIR eine Bewegung erkennt
    _loop();
}
}

void _delay(float seconds) {
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime) _loop();
}

void _loop() {
}

void loop() {
    _loop();
}

void setup() {
```

```
pinMode(10,EINGANG);
pinMode(1,EINGANG);
pinMode(20,AUSGANG);
pinMode(7,AUSGANG);
pinMode(14,EINGANG);
// Eingangs- und Ausgangspins definieren

while(1) {
    if(digitalRead(10) == 1){
        _delay(3);
        actived();
    }
    if(digitalRead(1) == 1){
        while(!(digitalRead(10) == 1))
        {
            _loop();
            tone(20,349,0.5*1000);
            delay(0.5*1000);
            digitalWrite(7,1);
            _delay(0.5);
            tone(20,392,0.5*1000);
            delay(0.5*1000);
            digitalWrite(7,0);
            _delay(0.5);
        }
        _loop();
    }
}
```

GitHub Smart House Project Page



<http://rbt.ist/smarthouse>

## 2.19. Sparschwein

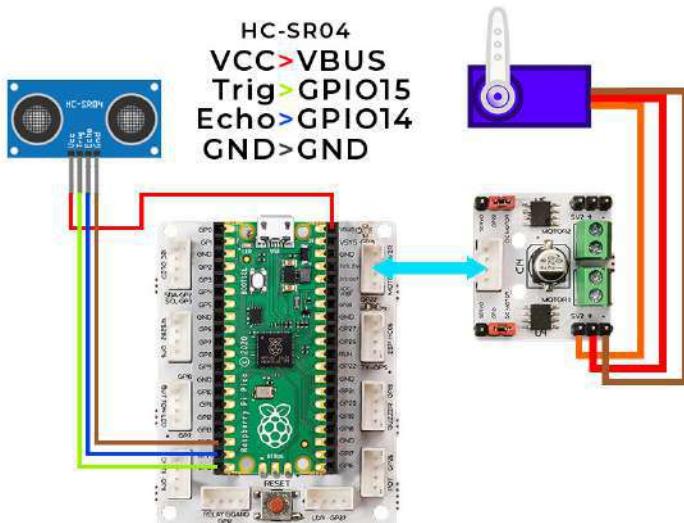
Ultraschallsensoren sind Sensoren, die elektrische Veränderungen zeigen, indem sie von Schallwellen beeinflusst werden. Diese Sensoren senden Schallwellen mit einer Frequenz, die unser Ohr nicht vernehmen kann, und erzeugen Entfernungsinformationen, indem sie die Rücklaufzeit der reflektierten Schallwellen berechnen. Wir, die Programmierer, entwickeln Projekte, indem wir die gemessene Entfernung und die Veränderungen in der Entfernung sinnvoll nutzen. Parksensoren an der Vorder- und Rückseite von Autos sind die Orte, an denen Ultraschallsensoren im täglichen Leben am häufigsten vorkommen. Weißt du, welches Lebewesen sich in der Natur mit dieser Methode orientiert? Da Fledermäuse blind sind, finden sie ihren Weg durch die Reflexionen der Geräusche, die sie erzeugen. [Es könnte ein Bild gezeigt werden, das die Schallwellen darstellt]

Viele von uns sparen gerne Geld. Es ist ein sehr schönes Gefühl, dass das Geld, das wir nach und nach sparen, nützlich ist, wenn wir es brauchen. In diesem Projekt werden Sie sich eine sehr angenehme und niedliche Spardose basteln. Sie werden beim Bau der Spardose den Servomotor und den Ultraschallsensor verwenden.

### 2.19.1. Projektdetails und Algorithmus

Im diesem Projekt werden der HC-SR04 Ultraschall-Abstandssensor und der SG90 Servomotor verwendet. Wenn der Benutzer Geld in den Schlitz der Spardose legt, wird der Abstandssensor die Nähe erkennen und an die Picobricks senden. Basierend auf diesen Informationen werden die Picobricks einen Servomotor betätigen, den Arm anheben, das Geld in die Spardose werfen und der Arm wird wieder nach unten gehen.

### 2.19.2. Schaltplan



## 2.19.3. Projektvorschlag

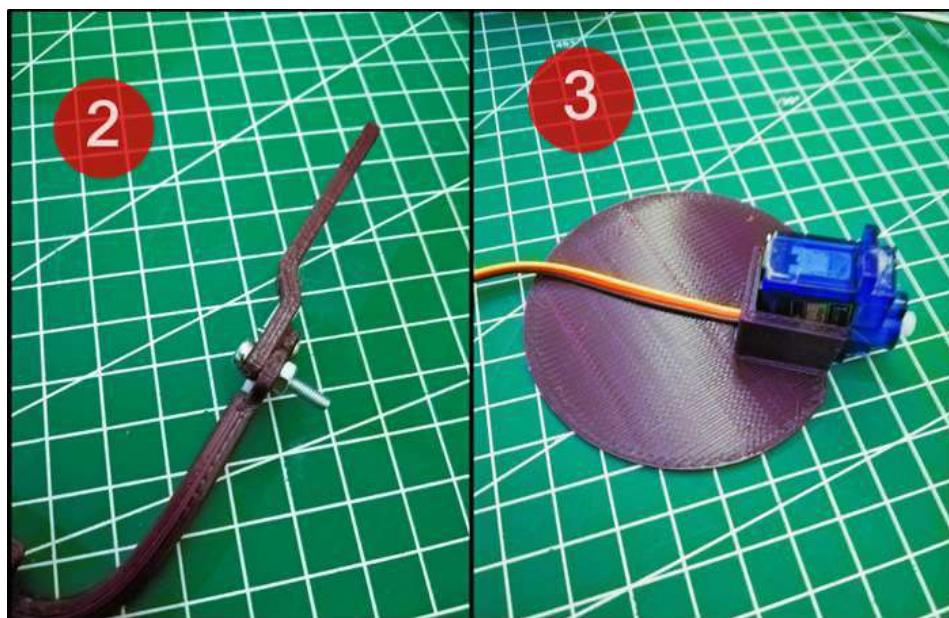
Durch das Hinzufügen eines RGB-LED-Moduls zum Schlemmer-Spardosenprojekt kann das Licht jedes Mal in der gewünschten Farbe leuchten, wenn eine Münze geworfen wird. Sie können auch einen Summer hinzufügen und jedes Mal einen Ton erzeugen, wenn eine Münze geworfen wird. Außerdem können Sie die Anzahl der Münzwürfe auf dem Bildschirm anzeigen, indem Sie einen OLED-Bildschirm hinzufügen.

## 2.19.4. Bauphasen des Projekts

Sie können auf die Originaldateien und Bauphasen des Projekts zugreifen, indem Sie hier [klicken](#). Im Gegensatz zu [dem Projekt in diesem Link](#) werden wir den HC-SR04 Ultraschall-Abstandssensor verwenden. Sie können die aktualisierten 3D-Zeichnungsdateien gemäß dem HC-SR04 Ultraschall-Abstandssensor von [diesem Link](#) herunterladen und 3D-Druck anfertigen.

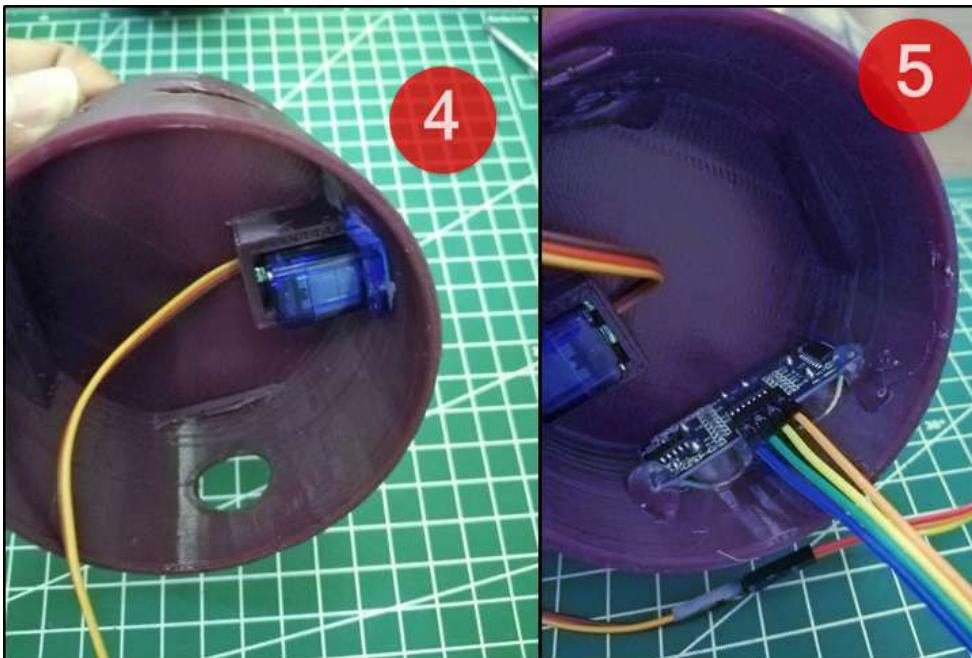


1: Befestigen Sie das Kunststoffgerät des Servomotors mit 2 Schrauben am Arm der Spardose.



2: Befestigen Sie den zweiten Teil des Spardosenarms mit der M3-Schraube und der Mutter am ersten Teil, wo sich der Schlitz befindet.

3: Führen Sie das Servomotorkabel durch und platzieren Sie es in seinem Slot.



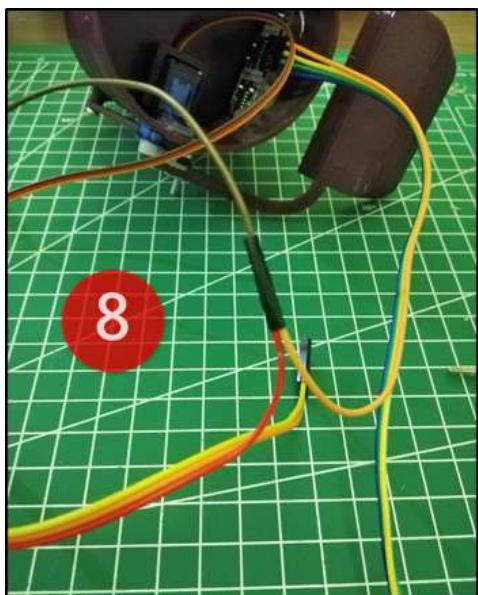
4: Platzieren Sie den Servomotor und sein Gehäuse auf dem Körper der Spardose. Sie können hier Heißkleber verwenden.

5: Platzieren Sie den Ultraschall-Abstandssensor im Körper der Spardose und fixieren Sie ihn mit Heißkleber.



6: Befestigen Sie den Spardosenarm am Servomotor und fixieren Sie ihn mit M3-Schrauben am oberen Deckel.

7: Befestigen Sie den Spardosenarm mit einer M2-Schraube am Körper.



8: Stecken Sie die Kabel des Servomotors und des Ultraschall-Abstandssensors ein und verbinden Sie die Stromkabel.

9: Verbinden Sie gemäß dem Schaltplan die Kabel des Servomotors und des Ultraschall-Abstandssensors mit dem Pico.



10: Stecken Sie das USB-Kabel des Pico ein, setzen Sie die Kabel wieder zusammen und befestigen Sie den Bodendeckel.  
Das ist alles.

## 2.19.5. Programmierung des Projekts mit MicroBlocks

Beim Schreiben des Codes müssen Sie zuerst den Winkel des Servomotors bearbeiten. Um den Deckel der Spardose zu schließen, müssen Sie die Winkelwerte in den Servomotor eingeben und den geeigneten Wert bestimmen. Dann müssen Sie den Servomotorwinkel in der offenen Position der Spardose finden. Wenn die Spardose zum ersten Mal gestartet wird, sollte der Deckel in der geschlossenen Position sein. Wenn der Wert vom Ultraschall-Abstandssensor

Der Sensor ist weniger als 5 cm entfernt, warten Sie 2 Sekunden, der Servomotor sollte arbeiten, heben Sie die Abdeckung an, und nach 300 Millisekunden sollte er erneut arbeiten und die Abdeckung in die geschlossene Position bringen.



[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.19.6. MicroPython-Codes des Projekts

```
from machine import Pin, PWM
import utime
#definiere die Bibliotheken

servo=PWM(Pin(21,Pin.OUT))
trigger = Pin(15, Pin.OUT)
echo = Pin(14, Pin.IN)
#definiere die Eingangs- und Ausgangspins

servo.freq(50)
servo.duty_u16(6750)

def getDistance():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value() == 0:
        signaloff = utime.ticks_us()
    while echo.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
```

```

distance = (timepassed * 0.0343) / 2
print("Die Entfernung zum Objekt beträgt ",distance,"cm")
return distance
#berechne die Entfernung
while True:
    utime.sleep(0.01)
if int(getDistance())<=5: #wenn die Distanzvariable kleiner als 5 ist
    servo.duty_u16(4010)
utime.sleep(0.3) #warte
    servo.duty_u16(6750)

```

## 2.19.7. Arduino C Codes des Projekts

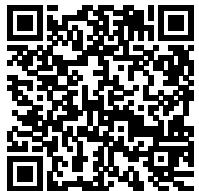
```

#include <Servo.h>
#define trigPin 15
#define echoPin 14
//Bibliotheken definieren
Servo servo;
void setup() {
    Serial.begin (9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    //Eingangs- und Ausgangspins definieren
    servo.attach(21); //Servo-Pin definieren
}
void loop() {
    long duration, distance;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;
    //Distanz berechnen
    if(distance < 5){ //wenn die Distanzvariable kleiner als 5 ist
        Serial.print(distance);
        Serial.println(" cm");
        servo.write(179);
    }
    else if (distance>5){ // wenn die Distanzvariable größer als 5 ist

```

```
Serial.print(distance);
Serial.println(" cm");
servo.write(100);
}
}
```

GitHub Piggy Bank Projektseite



<http://rbt.ist/bank>

## 2.20. NFC Smart Door

Sicherheitssysteme umfassen Technologien, die Berechtigungen an Eingängen von Gebäuden und Räumen steuern können. Karteneingangssysteme, bei denen nur autorisierte Personen die Operationssäle von Krankenhäusern betreten können, sind eines der ersten Beispiele, die einem in den Sinn kommen. Darüber hinaus sind die Eingangstüren von Bereichen, die von Personen oder Personal aller Ebenen in militärischen Sicherheitszentren nicht betreten werden dürfen, mit Karten- und Passwort-Eingangstechnologien ausgestattet. Diese elektronischen Systeme, die an Eingängen von Gebäuden und Räumen verwendet werden, verhindern nicht nur den Zutritt unbefugter Personen, sondern sorgen auch dafür, dass Ein- und Aus-Informationen aufgezeichnet werden. Passwort-Eingabe, Karten-Eingabe, Fingerabdruck-Scannen, Gesichtserkennung, Retina-Scannen und Sprachverarbeitungstechnologien sind die Authentifizierungsmethoden, die in elektronischen Zugangssystemen verwendet werden.

Systeme wie RFID und NFC sind die grundlegenden Formen der kontaktlosen Zahlungstechnologien heute. Obwohl die kontaktlose Zahlungstechnologie in Kreditkarten technisch unterschiedlich ist, ist die Arbeitslogik die gleiche. Der maximale Abstand zwischen dem Leser und der Karte ist eines der Merkmale, das die verwendeten Technologien voneinander unterscheidet. Beim Verlassen der Geschäfte, insbesondere in Bekleidungsgeschäften, werden NFC-Tags an den Produkten piepen, wenn sie von den Lesegeräten am Eingang erkannt werden. Eine Art von RFID-Technologie wird in diesen Systemen verwendet.

In diesem Projekt werden wir ein Karteneingangssystem an einem Modellhaus vorbereiten. Die elektronischen Komponenten, die wir verwenden werden, sind der MFRC522 RFID-Leser und 13,56 MHz Karten.

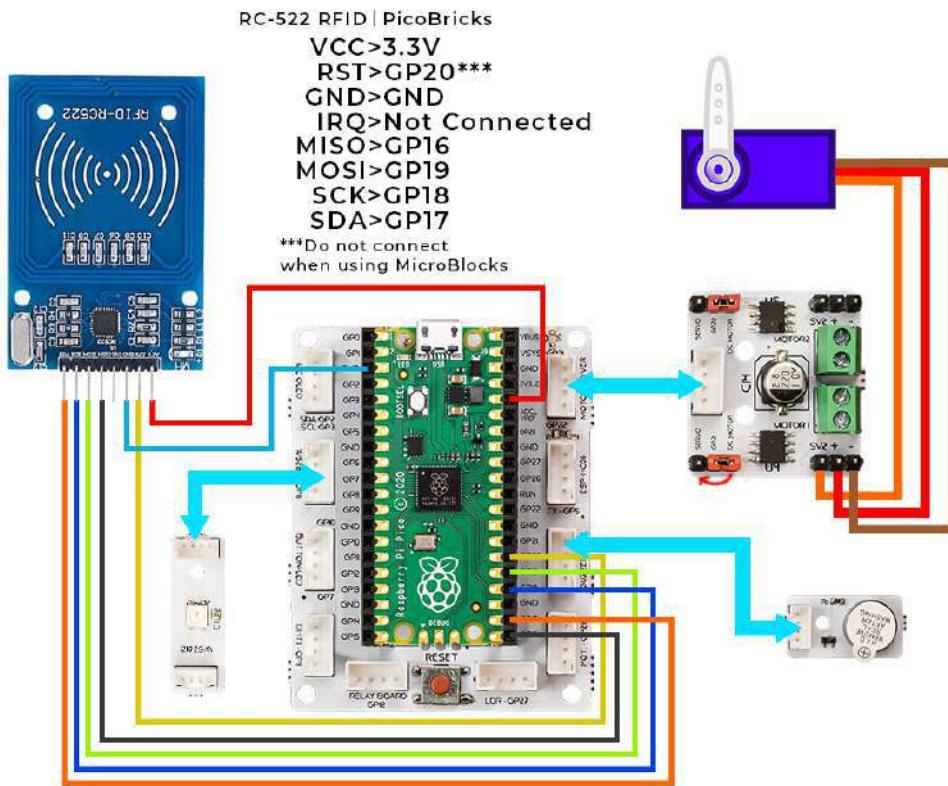
### 2.20.1. Projektdetails und Algorithmus

Platzieren Sie den MFRC522-Leser in der Nähe der Tür des Modells, sodass er von außen sichtbar ist. Platzieren Sie die RGB-LED und den Summer an der Wand, wo die Tür von außen sichtbar ist. Das Picoboard kann im Modell bleiben. Die Eingangstür des Modells sollte mit der Tür des Servos verbunden sein, während das Servo auf 0 Grad eingestellt ist, sollte die Tür geschlossen sein. Sie sollten die Seriennummer des RFID / NFC Tags bestimmen, der die Tür öffnen wird, die Variable für den Hausbesitzer erstellen und

die Seriennummer dieser Variablen zuweisen.

Setzen Sie die Tür in die geschlossene Position, wenn Picobricks startet. Lassen Sie den Summer piepen, wenn eine Karte dem RFID-Leser gezeigt wird. Wenn die Seriennummer der gelesenen Karte mit der Seriennummer in der Hausbesitzer-Variablen übereinstimmt, schalten Sie die RGB-LED auf grün. Dann lassen Sie die Tür öffnen. Stellen Sie sicher, dass die Tür 3 Sekunden nach dem Öffnen der Tür geschlossen ist. Wenn die Seriennummer der gelesenen Karte nicht mit der Hausbesitzer-Variable übereinstimmt, leuchtet die RGB-LED rot. Ein anderer Ton ertönt vom Summer.

## 2.20.2. Verdrahtungsdiagramm



Stellen Sie sicher, dass die Kabelverbindungen des RC522 RFID-Kartenlesermoduls

entsprechend der untenstehenden Tabelle hergestellt werden.

VCC	3.3V
RST	GP20 ***
GND	GND
IRQ	Nicht verbunden
MISO	GP16
MOSI	GP19
SCK	GP18
SDA	GP17

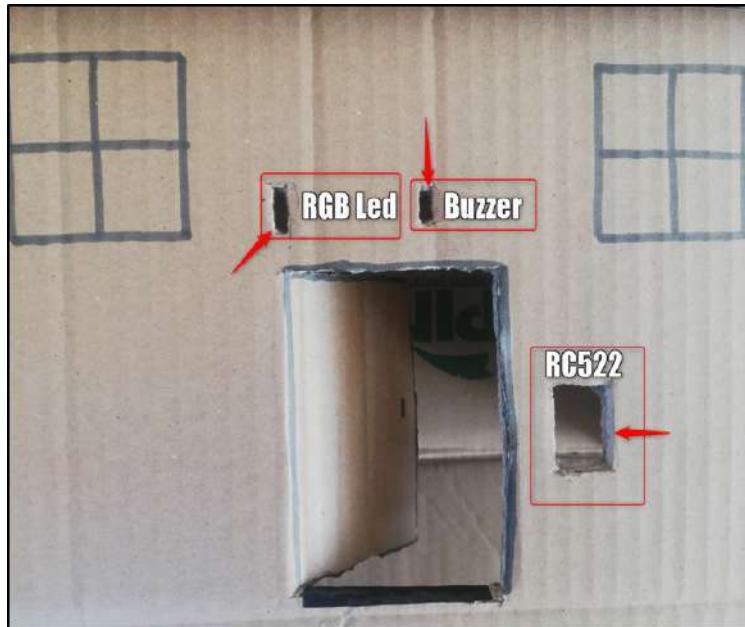
\*\*\* Verbindet sich nicht bei Verwendung von MicroBlocks

## 2.20.3. Projektvorschlag

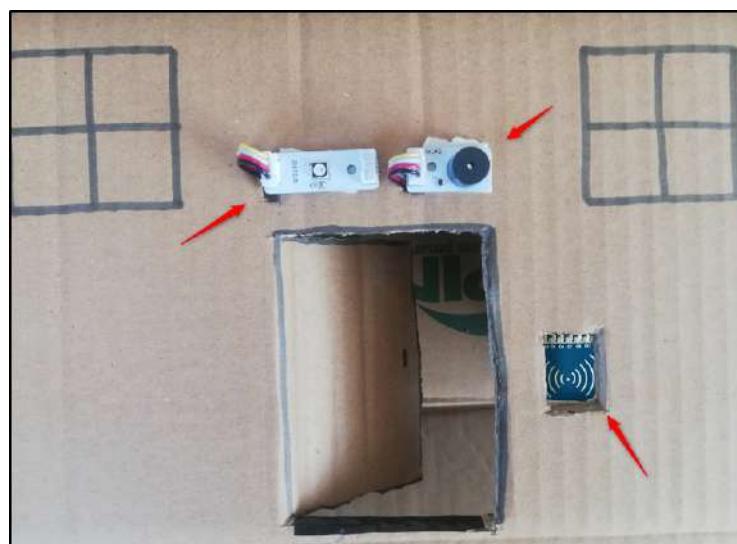
Im automatischen Türprojekt können Sie RFID-Karten Personennamen zuweisen. Durch das Hinzufügen eines OLED-Bildschirms zum Projekt können Sie den Namen der Person, die die Karte gelesen hat, auf dem OLED-Bildschirm anzeigen, wenn die Karte gelesen wird.

## 2.20.4. Bauphasen des Projekts

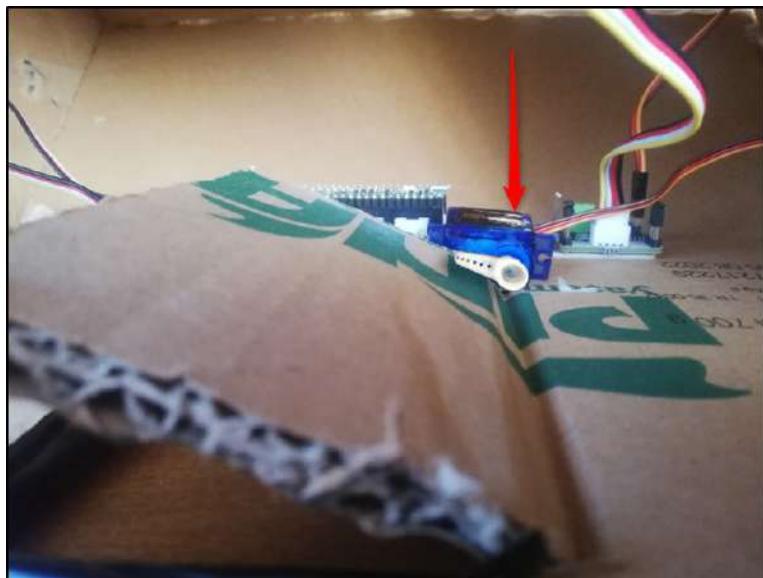
Wir werden das Projekt mit dem Hausmodell durchführen, das Sie im Smart Home-Projekt Nummer 18 verwendet haben. Bohren Sie Löcher für die RGB-LED, den Summer und den RC522 RFID-Leser in das Hausmodell.



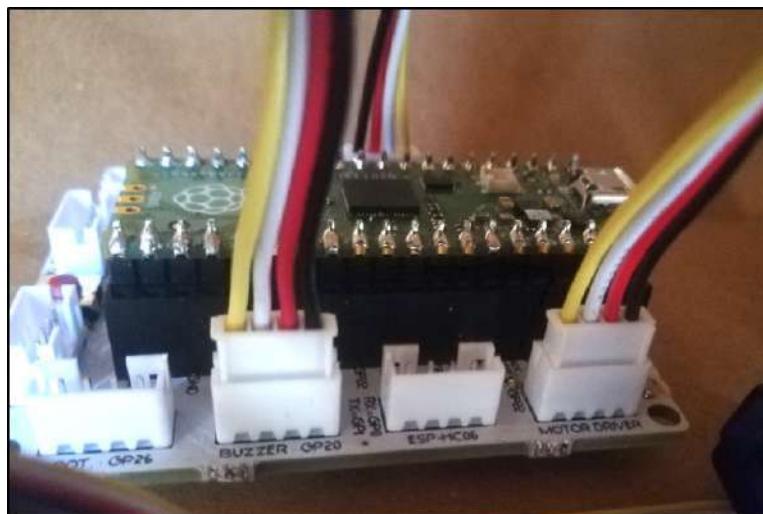
Bringen Sie doppelseitiges Schaumstoffband auf der Rückseite der RGB-LED und des Summers an und kleben Sie es auf die Box. Platzieren Sie den RC522 im Modell wie im Bild.



Befestigen Sie den Servomotor mit doppelseitigem Klebeband im Inneren des Modells als Scharnier in der oberen linken Ecke der Tür. Befestigen Sie den Servo-Kopf mit Heißkleber oder Flüssigkleber an der Tür.

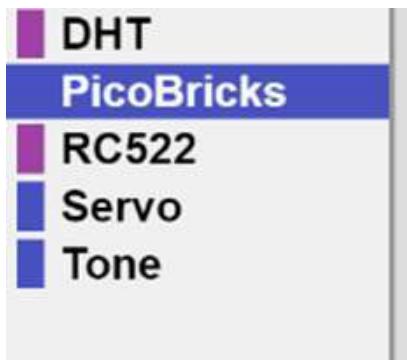


Platzieren Sie schließlich die Pico-Platine und die 2-Tasten-Batteriekiste im Inneren des Modellhauses und schließen Sie die Kabelverbindungen ab. Nach den abschließenden Überprüfungen Ihres Projekts ist es bereit zur Nutzung.



## 2.20.5. Programmierung des Projekts mit MicroBlocks

Zuerst müssen wir die aktuellen Karteninformationen des Benutzers lesen. Dann werden wir die Codes des Projekts vorbereiten. Erstellen Sie eine Variable namens homeowner. Fügen Sie die Bibliotheken RC522 und Servo hinzu.



Wenn Picobricks startet, geben Sie den SDA-Pin an und identifizieren Sie das Modul mit dem Block „RC522\_initialize“. Platzieren Sie die if-else-Struktur innerhalb der Forever-Schleife. Wenn die Karte gelesen wird, drucken wir die Seriennummer der Karte auf dem Bildschirm aus. Der Block „RC522 card UID“ ist eine Liste mit 4 Elementen. Um diesen Wert anzuzeigen oder zu vergleichen, sollten Sie den Block „Elemente der Liste verbinden“ in der Kategorie „Daten“ verwenden.



[Klicken Sie](#) für die MicroBlocks-Codes, die erforderlich sind, um die Seriennummer der Karte zu lesen. Zeigen Sie Ihre Karte dem RC522, wenn Sie den Code ausführen. Weisen Sie die Seriennummer der Karte über den Codeblöcken der homeowner-Variable mit dem Block „Liste“ in der Kategorie Daten zu. Die Seriennummer Ihrer Karte ist anders als die hier angegebene.



Jetzt können wir die Codes des Algorithmus des Projekts schreiben. Definieren Sie RC522, wenn Picobricks startet, definieren Sie die homeowner-Variable und setzen Sie den Servowinkel.



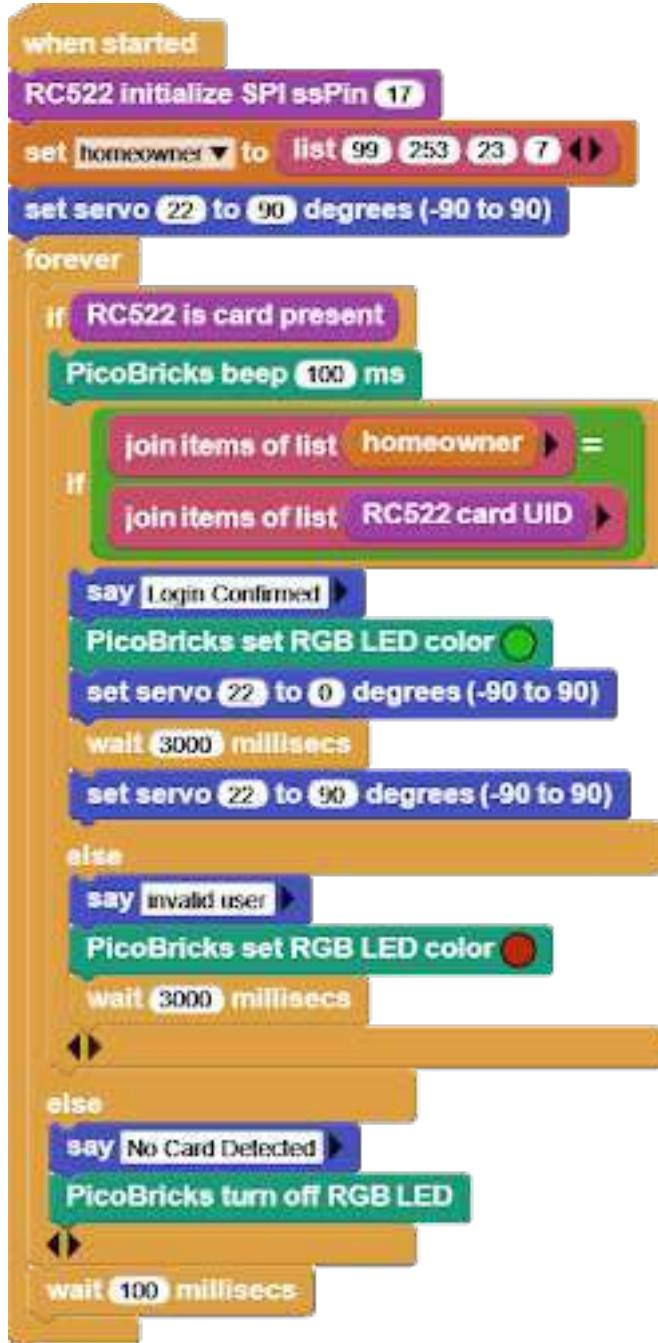
Der Summer wird piepen, wenn die Karte gezeigt wird. In Fällen, in denen die Karte nicht gezeigt wird, wird die Aussage „Keine Karte erkannt“ ausgedruckt. Bereiten Sie die erforderlichen Codes wie folgt vor, indem Sie eine if-else-Struktur innerhalb der Forever-Schleife platzieren.



Bereiten Sie die „if else“-Struktur vor, die die Variable des Hausbesitzers mit der Seriennummer der Karte vergleicht, die direkt nach dem Piepton gelesen wird. Drucken Sie „Login Bestätigt“ aus, wenn die Seriennummern übereinstimmen. Drucken Sie „ungültiger Benutzer“ aus, wenn die Seriennummern nicht übereinstimmen. Ihr Code sollte wie unten aussehen.



Ihr Servo sollte sich bewegen und die RGB-LED sollte grün leuchten, wenn die richtige Karte gezeigt wird. Warten Sie 3 Sekunden und das Servo sollte in seinen ursprünglichen Winkel zurückkehren. Die RGB-LED sollte rot leuchten, wenn die falsche Karte gezeigt wird. Warten Sie 3 Sekunden. In Fällen, wo die Karte nicht gelesen wird, sollte die RGB-LED ausgeschaltet werden. Die fertigen Codes des Projekts sollten wie folgt aussehen.



[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.20.6. MicroPython-Codes des Projekts

Der auszuführende Code zum Lernen der Karten-ID:

```
from machine import Pin, SPI  
from mfrc522 import MFRC522
```

```

import utime
#Bibliotheken definieren
sck = Pin(18, Pin.OUT)
mosi = Pin(19, Pin.OUT)
miso = Pin(16, Pin.OUT)
sda = Pin(17, Pin.OUT)
rst = Pin(15, Pin.OUT)
spi = SPI(0, baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)
rdr = MFRC522(spi, sda, rst)
#define MFRC522 pins

while True:
    (stat, tag_type) = rdr.request(rdr.REQIDL)
    if stat == rdr.OK:
        (stat, raw_uid) = rdr.anticoll()
        if stat == rdr.OK:
            uid = ("0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1], raw_uid[2], raw_
            uid[3]))
            print(uid)
            utime.sleep(1)
            #read the card and give the serial number of the card

```

Project Codes:

```

from machine import I2C, Pin, SPI, PWM
from mfrc522 import MFRC522
from ws2812 import NeoPixel
from utime import sleep

servo = PWM(Pin(21))
servo.freq(50)
servo.duty_u16(1350) #servo set 0 angle 8200 for 180.

buzzer = PWM(Pin(20, Pin.OUT))
buzzer.freq(440)

neo = NeoPixel(6, n=1, brightness=0.3, autowrite=False)
ROT = (255, 0, 0)
GRÜN = (0, 255, 0)
SCHWARZ = (0, 0, 0)

sck = Pin(18, Pin.OUT)
mosi = Pin(19, Pin.OUT)
miso = Pin(16, Pin.OUT)

```

```
sda = Pin(17, Pin.OUT)
rst = Pin(15, Pin.OUT)
spi = SPI(0, baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)
hauseigentümer = "0x734762a3"
rdr = MFRC522(spi, sda, rst)
```

while True:

```
(stat, tag_type) = rdr.request(rdr.REQIDL)
if stat == rdr.OK:
    (stat, raw_uid) = rdr.anticoll()
    if stat == rdr.OK:
        buzzer.duty_u16(3000)
        sleep(0.05)
        buzzer.duty_u16(0)
        uid = ("0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1], raw_uid[2], raw_
uid[3]))
        print(uid)
        sleep(1)
if (uid==hauseigentümer):
    neo.fill(GREEN)
    neo.show()
    servo.duty_u16(6000)
    sleep(3)
    servo.duty_u16(1350)
    neo.fill(BLACK)
    neo.show()

else:
    neo.fill(RED)
    neo.show()
    sleep(3)
    neo.fill(BLACK)
    neo.show()
    servo.duty_u16(1350)
```

## 2.20.7. Arduino C Codes des Projekts

Der auszuführende Code zum Lernen der Karten-ID:

```
#include <SPI.h>
#include <MFRC522.h>
//Bibliotheken definieren

int RST_PIN = 26;
```

```
int SS_PIN = 17;
//Pins definieren

MFRC522 rfid(SS_PIN, RST_PIN);

void setup()
{
    Serial.begin(9600);
    SPI.begin();
    rfid.PCD_Init();
}

void loop() {

if (!rfid.PICC_IsNewCardPresent())
    return;
if (!rfid.PICC_ReadCardSerial())
    return;
rfid.uid.uidByte[0] ;
rfid.uid.uidByte[1] ;
rfid.uid.uidByte[2] ;
rfid.uid.uidByte[3] ;
printid();
rfid.PICC_HaltA();
//Lesen Sie Ihre ID.
}
void printid()
{
Serial.print("Ihre ID: ");
for (int x = 0; x < 4; x++) {
    Serial.print(rfid.uid.uidByte[x]);
    Serial.print(" ");
}
Serial.println("");
}
```

Projektcodes:

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
#include <Adafruit_NeoPixel.h>
```

//Bibliotheken definieren.

```
#define RST_PIN 26
#define SS_PIN 17
#define servoPin 22
#define PIN 6
#define NUMPIXELS 1
#define buzzer 20
//define pins of servo,buzzer,neopixel and rfid.

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
Servo motor;
MFRC522 rfid(SS_PIN, RST_PIN);

byte ID[4] = {"Write your own ID."};

void setup() {
    pixels.begin();
    motor.attach(servoPin);
    Serial.begin(9600);
    SPI.begin();
    rfid.PCD_Init();
    pinMode(buzzer, OUTPUT);

}

void loop()
{
    pixels.clear();

    if ( ! rfid.PICC_IsNewCardPresent())
        return;
    if ( ! rfid.PICC_ReadCardSerial())
        return;

    if
    (
        rfid.uid.uidByte[0] == ID[0] &&
        rfid.uid.uidByte[1] == ID[1] &&
        rfid.uid.uidByte[2] == ID[2] &&
        rfid.uid.uidByte[3] == ID[3] )
    {
        Serial.println("Tür geöffnet.");
    }
}
```

```
printid();
tone(buzzer,523);
delay(200);
noTone(buzzer);
delay(100);
tone(buzzer,523);
delay(200);
noTone(buzzer);
pixels.setPixelColor(0, pixels.Color(0, 250, 0));
delay(200);
pixels.show();
pixels.setPixelColor(0, pixels.Color(0, 0, 0));
delay(200);
pixels.show();
motor.write(180);
delay(2000);
motor.write(0);
delay(1000);

//RGB-LED wird grün und die Tür öffnet sich dank des Servomotors, wenn die richtige Karte vom Sensor gelesen wird.

}

else
{
Serial.println("Unbekannte Karte.");
printid();
tone(buzzer,494);
delay(200);
noTone(buzzer);
delay(100);
tone(buzzer,494);
delay(200);
noTone(buzzer);
pixels.setPixelColor(0, pixels.Color(250, 0, 0));
delay(100);
pixels.show();
pixels.setPixelColor(0, pixels.Color(0, 0, 0));
delay(100);
pixels.show();

//RGB LED wird rot und die Tür öffnet sich nicht, wenn die falsche Karte am Sensor gelesen wird

}
rfid.PICC_HaltA();

}

void printid()
```

```
{  
Serial.print("ID Nummer: ");  
for(int x = 0; x < 4; x++){  
    Serial.print(rfid.uid.uidByte[x]);  
    Serial.print(" ");  
}  
Serial.println("");  
}
```

GitHub NFC Smart Door Projektseite



<http://rbt.ist/door>

## 2.21. Automatischer Mülleimer

Die Covid-19-Pandemie hat die täglichen Routinen der Menschen in vielen Bereichen verändert. In vielen Bereichen wie Reinigung, Arbeit, Einkaufen und Sozialleben wurden die Menschen mit einer Reihe neuer Regeln konfrontiert, die sie einhalten mussten. Covid-19 hat zur Entwicklung neuer Geschäftsfelder sowie einiger Produkte geführt, die sich abheben. In einer Zeit, in der Handhygiene sehr wichtig war, wollte niemand den Deckel des Mülleimers berühren, um seinen Müll wegzwerfen.

Die Mülleimer, deren Deckel sich automatisch öffnen, wenn man sich nähert, und die bei Vollheit bereit sind, die Tüten wegzwerfen, fanden Käufer zu Preisen, die weit über ihren Kosten lagen. Darüber hinaus boten automatische Desinfektionsmittelmaschinen kontaktlose Hygiene, indem sie eine bestimmte Menge Flüssigkeit in unsere Hände gossen, wenn wir sie unter unsere Hände hielten. Automatische Desinfektionsmittelmaschinen fanden sich zu Preisen, die weit über ihren Kosten lagen, im Regal. Diese beiden Produkte haben in Bezug auf das Funktionssystem Ähnlichkeiten. In automatischen Desinfektionsmittelmaschinen überträgt eine Pumpe mit einem Elektromotor die Flüssigkeit direkt, und einige Modelle verfügen über Geräte, die auf dem Pumpensystem mit der Kraft des Servomotors basieren. In automatischen Mülleimern wurde ein Servomotor verwendet, der den Deckel öffnet, und Infrarot- oder

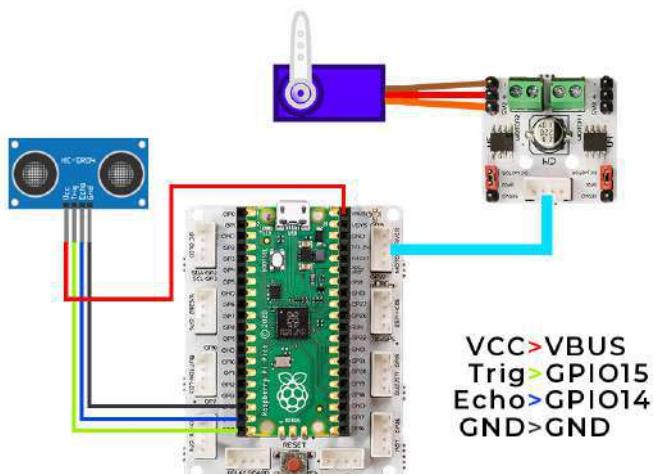
Ultraschallsensoren wurden verwendet, um die Handbewegung zu erkennen.

In diesem Projekt werden Sie einen mobilen und automatischen stilvollen Mülleimer für Ihr Zimmer mit einem Ultraschallsensor und einem Servomotor mit PicoBricks herstellen.

### 2.21.1. Projektdetails und Algorithmus

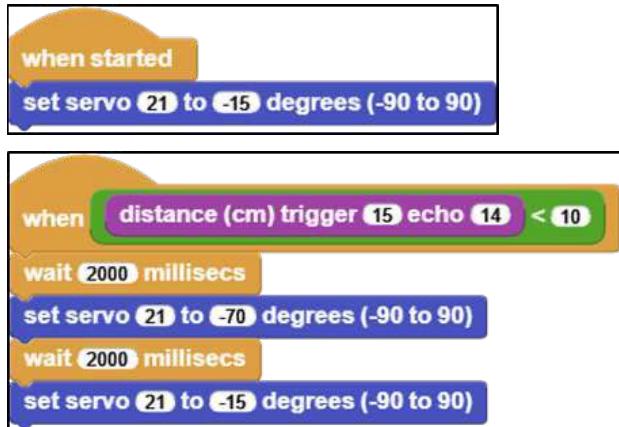
Im diesem Projekt werden der HC-SR04 Ultraschall-Abstandssensor und der SG90 Servomotor verwendet. Wenn der Benutzer seine Hand vor den Deckel des Mülleimers hält, erkennt der Abstandssensor die Nähe und sendet diese an die PicoBricks. Basierend auf diesen Informationen wird PicoBricks den Deckel des Mülleimers öffnen, indem er einen Servomotor betreibt, und ihn nach kurzer Zeit wieder schließen.

### 2.21.2. Schaltplan



## 2.21.3. Programmierung des Projekts mit MicroBlocks

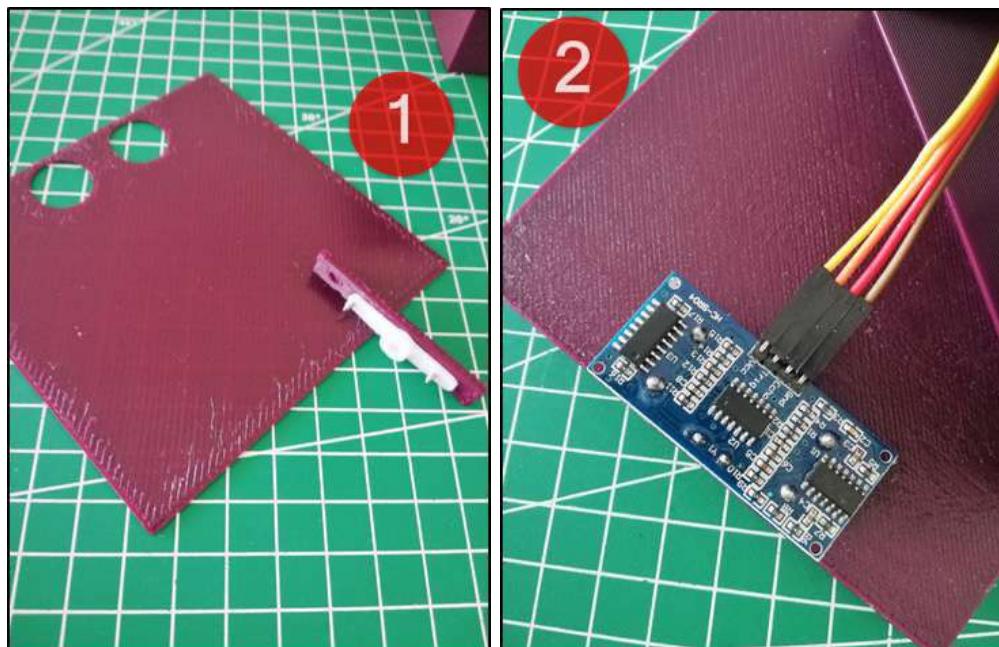
Beim Schreiben des Codes müssen Sie zunächst den Winkel des Servomotors einstellen. Um den Deckel des Mülleimers in die geschlossene Position zu bringen, müssen Sie die Winkelwerte in den Servomotor eingeben und den geeigneten Wert bestimmen. Dann müssen Sie den Winkel des Servomotors in der offenen Position des Mülleimers finden. Der Deckel muss in der geschlossenen Position sein, wenn der Mülleimer zum ersten Mal gestartet wird. Wenn der Wert des Ultraschallsensors weniger als 10 cm beträgt, warten Sie 2 Sekunden, der Servomotor sollte arbeiten, den Deckel anheben, und nach 2 Sekunden sollte er erneut arbeiten und den Deckel in die geschlossene Position bringen.



[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

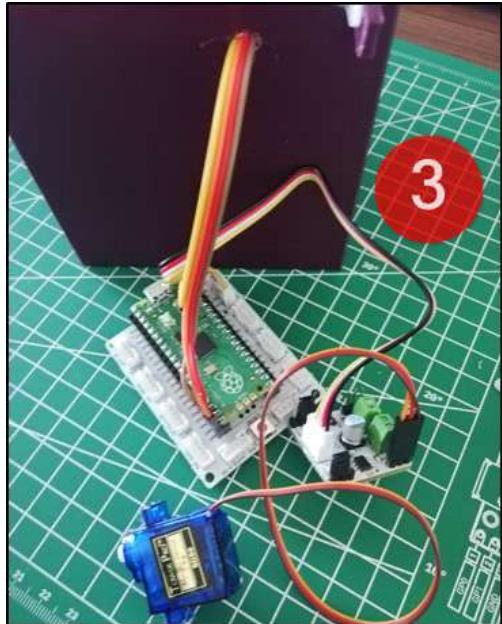
## 2.21.4. Bauphasen des Projekts

Sie können die 3D-Zeichnungsdateien des Projekts von [diesem Link herunterladen](#) und 3D drucken.



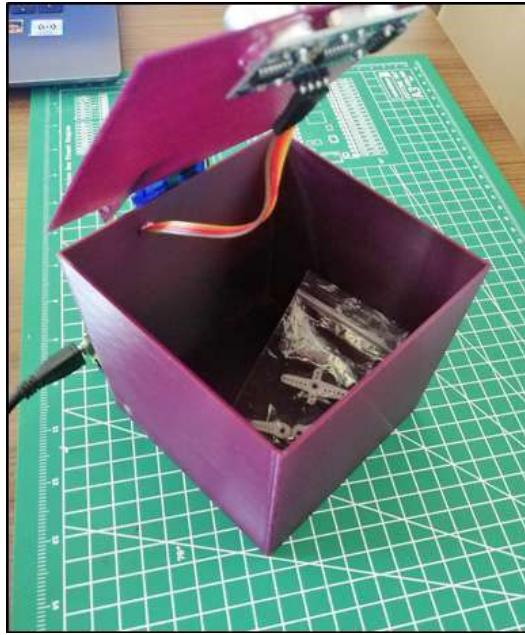
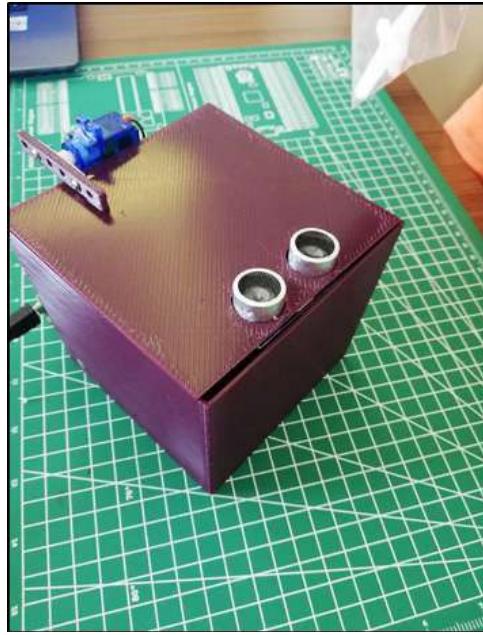
1: Befestigen Sie es, indem Sie es an den Deckel des Mülltonnenbehälters des Servomotorapparats schrauben.

2: Befestigen Sie den Ultraschall-Abstandssensor mit Heißkleber am Deckel der Mülltonne.



3: Führen Sie die Kabel des Ultraschall-Abstandssensors durch das Loch in der Box und verbinden Sie sie mit den Pins, die im Schaltplan der Picobricks dargestellt sind, und stellen Sie die Verbindungen des Servomotors und des Motorsteuergeräts her.

4: Befestigen Sie die Teile des Servomotors, der Picobricks und des Motorsteuergeräts mit Heißkleber an der Box.



Wenn alles gut gelaufen ist, öffnet sich der Deckel des Behälters, wenn Sie Ihre Hand in die Nähe der Mülltonne bringen, und schließt sich wieder, nachdem Sie den Müll weggeworfen haben.

## 2.21.5. Projektvorschlag

Wie in diesem Projekt, in dem wir eine Mülltonne in unserem Haus mit Hilfe von Sensoren und Motoren automatisieren, können Sie auch eine Schublade oder eine Schranktür automatisch mit Hilfe von Sensoren und Motoren öffnen.

## 2.21.6. MicroPython-Codes des Projekts

```
from machine import Pin, PWM
from utime import sleep

servo=PWM(Pin(21,Pin.OUT))
trigger = Pin(15, Pin.OUT)
echo = Pin(14, Pin.IN)

servo.freq(50)
servo.duty_u16(1920) #15 Grad

def getDistance():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value() == 0:
        signaloff = utime.ticks_us()
    while echo.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    print("Die Entfernung zum Objekt beträgt ",distance,"cm")
    return distance

while True:
    sleep(0.01)
    if int(getDistance())<=10:
        servo.duty_u16(4010) #70 Grad
        utime.sleep(0.3)
        servo.duty_u16(1920)
```

## 2.21.7. Arduino C Codes des Projekts

```
#include <Servo.h>
#define trigPin 14
#define echoPin 15
Servo servo;
void setup() {
    Serial.begin (9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    servo.attach(21);
}

void loop() {
lange Dauer, Entfernung;

delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
Dauer = pulseIn(echoPin, HIGH);
Entfernung = (Dauer/2) / 29.1;
wenn (Entfernung < 80) {
    Serial.print(distance);
    Serial.println(" cm");
    servo.write(179);
}

else if (distance<180) {
    Serial.print(distance);
    Serial.println(" cm");
    servo.write(100);
}

}
```

---

# GitHub Automatisches Mülltonnenprojekt Seite



<http://rbt.ist/bin>

## 2.22. Digitaler Maßstab

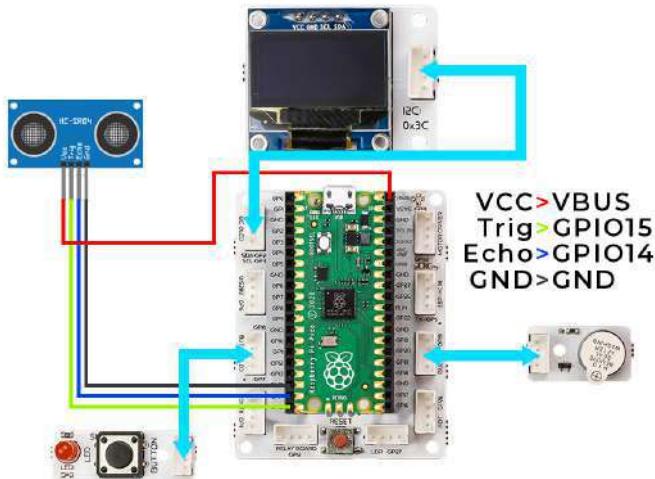
Viele Werkzeuge werden verwendet, um Längen zu messen. An erster Stelle dieser Werkzeuge stehen Maßstäbe. Unser Messinstrument unterscheidet sich je nach Ort und Größe, die gemessen werden sollen. Maßbänder werden im Bauwesen und in der Architektur verwendet, und Schieblehren werden für kleine Objekte eingesetzt, die eine Millimeterpräzision erfordern. Darüber hinaus, wenn es gewünscht ist, eine Fläche zu messen, die sowohl groß als auch präzise sein muss, werden Distanzmesser verwendet, die mit Laser- und Infrarotsystemen arbeiten. Ultraschallgeräte, die im Gesundheitssektor eingesetzt werden, arbeiten ebenfalls nach demselben Prinzip, wandeln jedoch ihre Messungen in visuelle Darstellungen um.

In unserem Projekt werden wir PicoBricks und einen Ultraschallsensor verwenden, um einen digitalen Maßstab vorzubereiten, der den Abstandswert auf dem OLED-Bildschirm anzeigt, wenn die Taste gedrückt wird. Ultraschallsensoren erkennen den Abstand anhand der Rücklaufzeiten der von ihnen ausgesendeten Schallwellen.

### 2.22.1. Projektdetails und Algorithmus

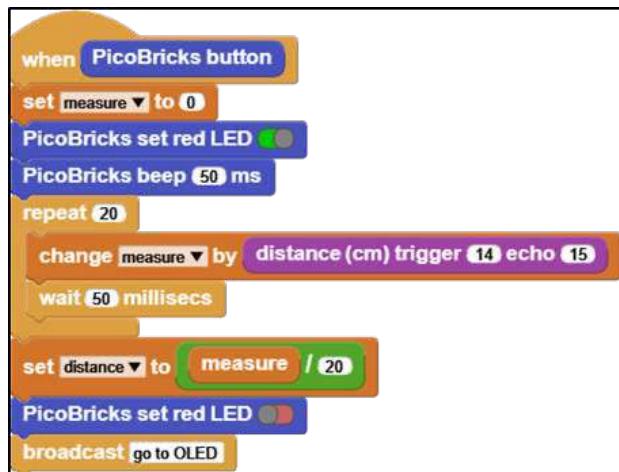
Wenn Picobricks startet, werden Anweisungen auf dem OLED-Bildschirm angezeigt. Nachdem der Benutzer die Taste gedrückt hat, werden 20 Messungen in Intervallen von 50 Millisekunden für 1 Sekunde durchgeführt und der Durchschnitt wird berechnet. Die rote LED bleibt während der Messung eingeschaltet, und die rote LED schaltet sich aus, wenn die Messung abgeschlossen ist. Der Durchschnittswert wird zum Abstand von der Spitze des Sensors zur Rückseite der Box hinzugefügt. Der letzte Abstandswert wird auf dem OLED-Display angezeigt.

### 2.22.2. Schaltplan

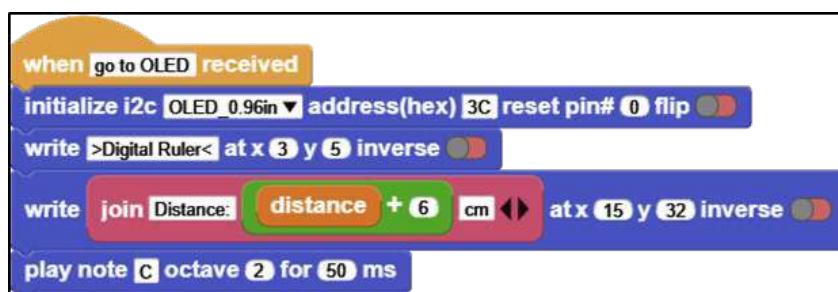


## 2.22.3. Projekt mit MicroBlocks codieren

Starten Sie MicroBlocks und verbinden Sie Picobricks. Fügen Sie die Bibliotheken Tone, Distance, OLED Graphics hinzu. Erstellen Sie Variablen mit dem Namen distance, um die Entfernung zu berechnen, und measure, um den Messwert zu mitteln. Schalten Sie die rote LED ein, um anzuzeigen, dass die Messung begonnen hat, wenn die Taste gedrückt wird. Nach einem kurzen Piepton führen Sie 20 Messungen in Abständen von 50 Millisekunden durch und weisen den Durchschnitt dem Wert der distance-Variable zu. Senden Sie das Signal go an das OLED, um die rote LED auszuschalten und das Ergebnis auf dem OLED-Bildschirm anzuzeigen.



Fügen Sie die Größe Ihrer Box zu dem vom Sensor gemessenen Wert hinzu, während Sie die distance-Variable auf dem Bildschirm anzeigen. Da die Dicke der Box, die wir in unserem Projekt verwendet haben, 6 cm beträgt, haben wir 6 cm zur distance-Variable hinzugefügt und auf dem Bildschirm angezeigt. Erzeugen Sie einen tiefen Ton, um den Benutzer darüber zu informieren, dass die Messung abgeschlossen ist.

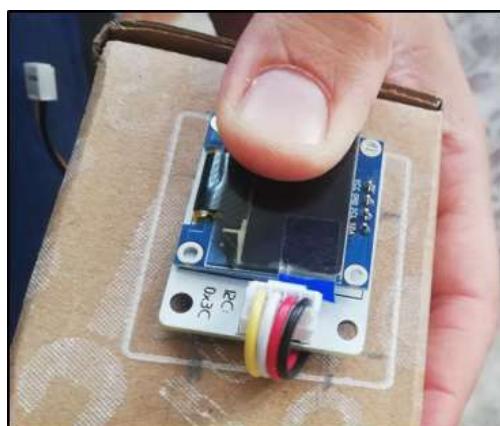
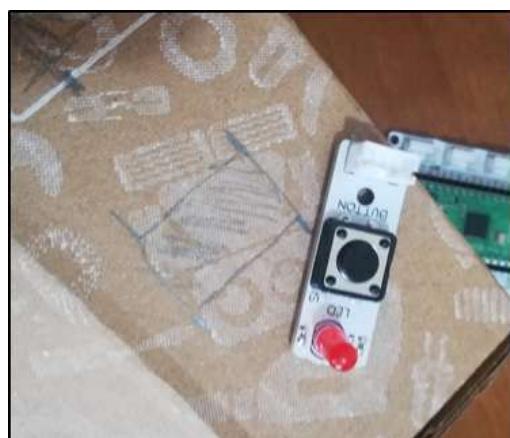


[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.22.4. Bauphasen des Projekts

Um das Projekt vorzubereiten, benötigen Sie doppelseitiges Schaumstoffband, ein Cuttermesser und einen Abfallkarton von etwa 15x10x5 cm.

1. Schneiden Sie mit einem Cuttermesser die Löcher für den Ultraschallsensor, den OLED-Bildschirm, das Tasten-LED-Modul, den Summer und die Batteriebox, um die Kabel hindurchzuführen.



2. Hängen Sie alle Kabel in die Box und kleben Sie die Rückseite der Module mit doppelseitigem Schaumstoffband an die Box. Verbinden Sie den Trig-Pin des Ultraschallsensors mit dem GPIO14-Pin und den Echo-Pin mit dem GPIO15-Pin. Sie sollten den VCC-Pin mit dem VBUS-Pin auf dem Picoboard verbinden.



3. Nachdem Sie die Kabelverbindungen aller Module abgeschlossen haben, können Sie die 2-Batteriebox in die Strombuchse des Picoboard einsetzen und den Schalter einschalten. Das war's für das digitale Linealprojekt!



## 2.22.5. Projektvorschlag

Sie können das digitale Lineal in ein Höhenmessgerät verwandeln, indem Sie es an der Wand oder Decke befestigen. Da der Höhenmesser an der Wand oder Decke befestigt wird, ist es nicht möglich, durch Drücken der Taste zu messen. Dafür können Sie das HC05-Bluetooth-Modul zum Projekt hinzufügen und es messen lassen, wenn ein Befehl von der mobilen Anwendung empfangen wird.

## 2.22.6. MicroPython-Codes des Projekts

```
from machine import Pin, PWM, I2C
from utime import sleep
from picobricks import SSD1306_I2C
import utime
#define die Bibliotheken
redLed=Pin(7,Pin.OUT)
button=Pin(10,Pin.IN,Pin.PULL_DOWN)
buzzer=PWM(Pin(20,Pin.OUT))
buzzer.freq(392)
trigger = Pin(15, Pin.OUT)
echo = Pin(14, Pin.IN)
#define Eingangs- und Ausgangspins
WIDTH = 128
HEIGHT = 64
#OLED-Bildschirmeinstellungen
sda=machine.Pin(4)
scl=machine.Pin(5)
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)
#initialisiere digitalen Pin 4 und 5 als AUSGANG für die OLED-Kommunikation
oled = SSD1306_I2C(128, 64, i2c)
measure=0
finalDistance=0

def getDistance():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value() == 0:
        signaloff = utime.ticks_us()
```

```

while echo.value() == 1:
    signalon = utime.ticks_us()
timepassed = signalon - signaloff
distance = (timepassed * 0.0343) / 2
return distance
#Berechne die Distanz
def getMeasure(pin):
    global measure
    global finalDistance
    redLed.value(1)
    for i in range(20):
        measure += getDistance()
        sleep(0.05)
    redLed.value(0)
    finalDistance = (measure/20) + 1
    oled.fill(0)
    oled.show()
    oled.text(">Digitaler Lineal<", 2,5)
    oled.text("Distanz " + str(round(finalDistance)) +" cm", 0, 32)
    oled.show()
#Gib die angegebene Distanz an den angegebenen x- und y-Koordinaten auf dem OLED-Bildschirm
aus
print(finalDistance)
buzzer.duty_u16(4000)
sleep(0.05)
buzzer.duty_u16(0)
measure=0
finalDistance=0
#den Buzzer betätigen
button.irq(trigger=machine.Pin.IRQ_RISING, handler=getMeasure)

```

## 2.22.7. Arduino C-Codes des Projekts

```

#include <Wire.h>
#include "ACROBOTIC_SSD1306.h"
#include <NewPing.h>
// Bibliotheken definieren
#define TRIGGER_PIN 15
#define ECHO_PIN 14
#define MAX_DISTANCE 400

```

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

#define T_B 493

int distance = 0;
int total = 0;

void setup() {
  pinMode(7,OUTPUT);
  pinMode(20,OUTPUT);
  pinMode(10,INPUT);
  // Eingangs- und Ausgangspins definieren
  Wire.begin();
  oled.init();
  oled.clearDisplay();

}

void loop() {

delay(50);
if(digitalRead(10) == 1){

  int measure=0;
  digitalWrite(7,HIGH);
  tone(20,T_B);
  delay(500);
  noTone(20);

  for (int i=0;i<20;i++){

    measure=sonar.ping_cm();
    total=total+measure;
    delay(50);
  }

  distance = total/20+6; // berechne die Distanz
  digitalWrite(7,LOW);

  delay(1000);

}
```

```
oled.clearDisplay();
oled.setTextXY(2,1);
oled.putString(">Digitaler Lineal<");
oled.setTextXY(5,1);
oled.putString("Distanz: ");
oled.setTextXY(5,10);
String string_distance=String(distance);
oled.putString(string_distance);
oled.setTextXY(5,12);
oled.putString("cm"); // drucke die berechnete Distanz auf dem OLED
Bildschirm

measure=0;
distance=0;
total=0;
}
}
```

GitHub Digital Ruler Projektseite



<http://rbt.ist/ruler>

## 2.23. Luftklavier

Mit der Entwicklung der Elektroniktechnologie wurden Musikanstrumente, die schwer zu produzieren, teuer und von hoher Klangqualität sind, digitalisiert. Klaviere gehören zu diesen Instrumenten. Jede Taste von digitalen Klavieren erzeugt elektrische Signale mit unterschiedlichen Frequenzen. So kann es 88 verschiedene Töne über seine Lautsprecher spielen. Faktoren wie die Verzögerungszeit der Tasten von digitalen Instrumenten, die Qualität des Lautsprechers und die Auflösung des Klangs sind Faktoren, die die Qualität beeinflussen. Bei E-Gitarren werden die Vibrationen der Saiten anstelle von Tasten digitalisiert. Auf der anderen Seite können bei Blasinstrumenten die gespielten Töne dank der hochauflösenden Mikrofone, die mit dem Audioausgang verbunden sind, in elektrische Signale umgewandelt und aufgezeichnet werden. Diese Entwicklung in der Elektroniktechnologie hat den Zugang zu hochpreisigen Musikanstrumenten erleichtert, die Musikbildung hat eine größere Vielfalt gewonnen und sich an ein breiteres Publikum verbreitet.

In diesem Projekt werden wir ein einfaches Klavier bauen, das 8 Töne mit PicoBricks spielen kann. Der Lautsprecher dieses Klaviers wird der Summer sein. Der

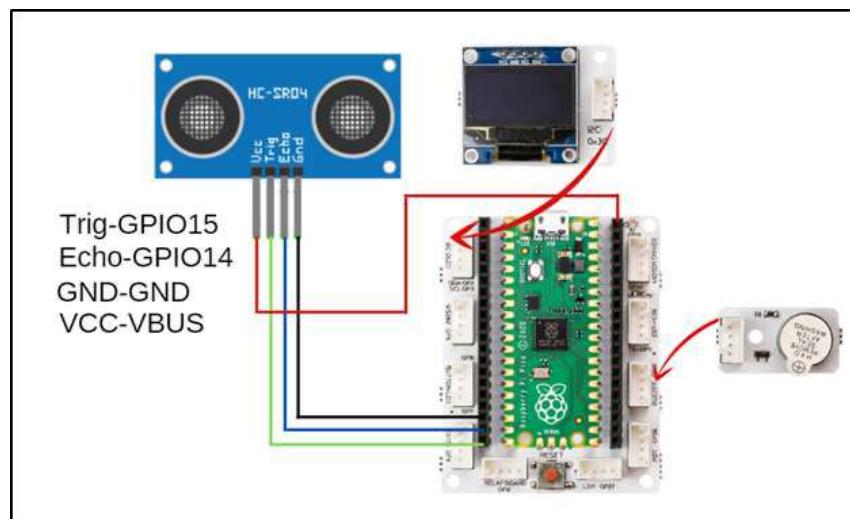
ultraschall Sensor wird als die Tasten des Klaviers fungieren.

### 2.23.1. Projektdetails und Algorithmus

In diesem Projekt werden wir eine Klavieranwendung mit dem HC-SR04 Ultraschall-Abstandssensor und dem Summermodul auf PicoBricks erstellen. Wir werden den Summer verschiedene Töne spielen lassen, je nach den Werten, die vom Abstandssensor kommen, und wir werden Melodien erzeugen, indem wir unsere Hand nahe an den Sensor und wieder weg bewegen. Darüber hinaus werden wir die

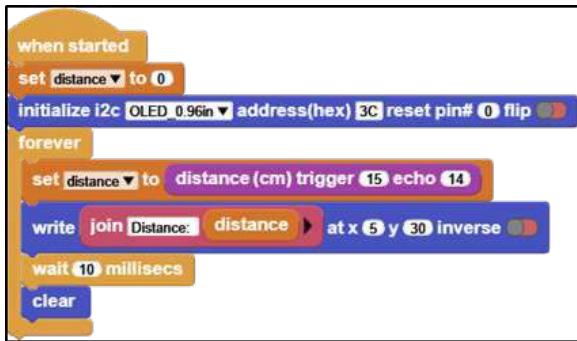
Informationen über den gespielten Ton sofort auf dem OLED-Bildschirm ausdrucken.

### 2.23.2. Schaltplan



### 2.23.3. Programmierung des Projekts mit MicroBlocks

Wenn wir beginnen, den Code des Projekts mit MicroBlocks zu schreiben, müssen wir zuerst die Tone-Bibliothek, die OLED-Grafikbibliothek in der Kategorie Grafik und die Abstandsbibliotheken in der Kategorie Sensorik importieren, indem wir auf die Schaltfläche Bibliothek hinzufügen klicken. Nach dem Hinzufügen der Bibliotheken müssen wir die Pins für das HC-SR04-Modul definieren und eine Variable namens Abstand erstellen, um die Daten vom Sensor zu empfangen und zu verarbeiten, und die notwendigen Codes schreiben, um die Sensordaten an die Variable zu übertragen und sie auf dem OLED- Bildschirm anzuzeigen.



Jede Note hat ein entsprechendes Buchstabenäquivalent. Die Tonbibliothek enthält auch Blöcke, die wir ausführen können, indem wir die Buchstabenäquivalente der Noten eingeben.



Wir sollten die Buchstabenäquivalente der Noten, die wir spielen möchten, im Feld im Play-Block schreiben. Durch den Vergleich der Distanzwerte in der Distanzvariablen können Sie das Projekt ausführen, indem Sie den Oktavwert im Play-Block nach Belieben ändern, nachdem Sie die notwendigen Codes für die Noten geschrieben haben, die sich in Abständen von 5 cm nacheinander ändern.

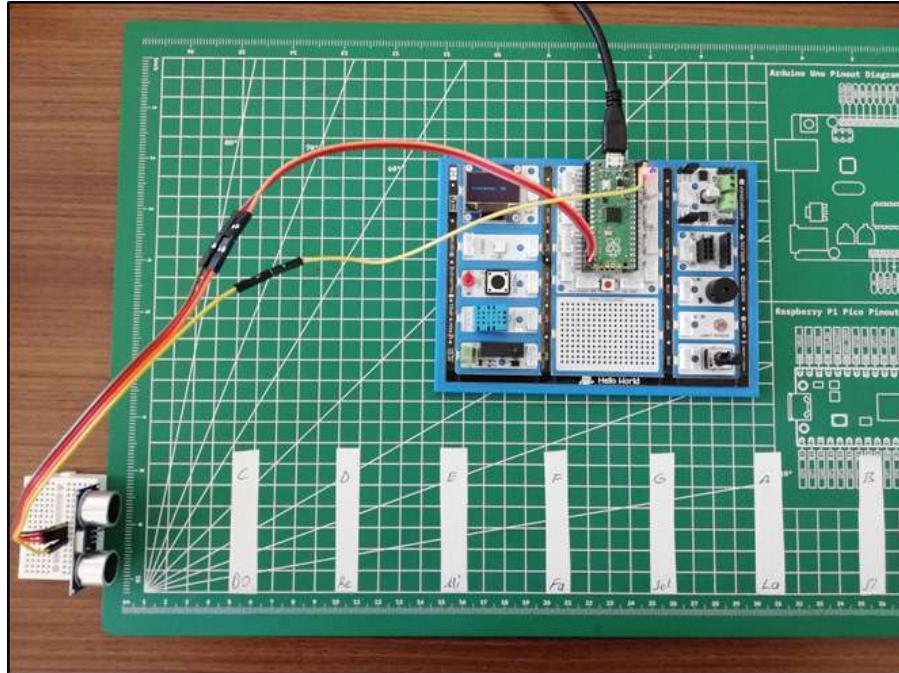
```

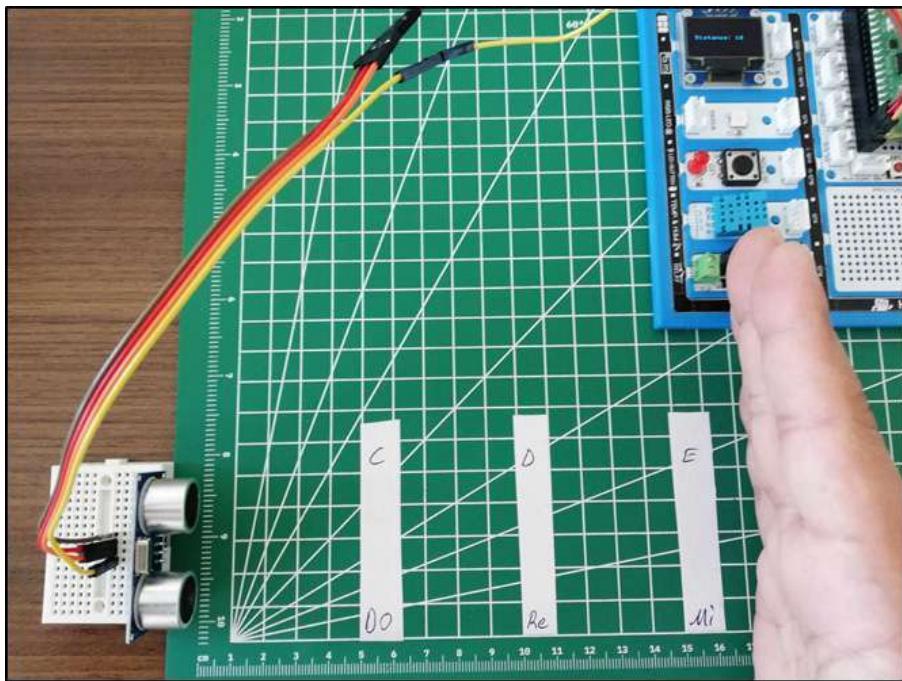
when started
forever
if distance > 5 and distance < 11
    play note C octave 1 for 400 ms
else if distance > 10 and distance < 16
    play note D octave 1 for 400 ms
else if distance > 15 and distance < 21
    play note E octave 1 for 400 ms
else if distance > 20 and distance < 26
    play note F octave 1 for 400 ms
else if distance > 25 and distance < 31
    play note G octave 1 for 400 ms
else if distance > 30 and distance < 36
    play note A octave 1 for 400 ms
else if distance > 35 and distance < 41
    play note B octave 1 for 400 ms
else
    //<>

```

[Klicken](#) Sie, um auf die MicroBlocks-Codes des Projekts zuzugreifen.

## 2.23.4. Bauphasen des Projekts





## 2.23.5. Projektvorschlag

Es gibt einen Sofortknopf auf PicoBricks. Indem Sie 7 Tasten an Pico anschließen, können Sie unterschiedliche Noten spielen, wenn jede Taste gedrückt wird. Sie können Tasten für den Oktavwert und die Schlagzeiten verwenden und Ihr Klavierprojekt entwickeln.

## 2.23.6. MicroPython-Codes des Projekts

```

from machine import Pin, PWM, I2C
from utime import sleep
import utime
from picobricks import SSD1306_I2C
import _thread
#definiere die Bibliotheken

buzzer=PWM(Pin(20,Pin.OUT))
trigger = Pin(15, Pin.OUT)
echo = Pin(14, Pin.IN)
#definiere die Eingangs- und Ausgangspins

WIDTH = 128
HEIGHT = 64
#OLED-Bildschirmeinstellungen

sda=machine.Pin(4)

```

```
scl=machine.Pin(5)
i2c=machine.I2C(0,sda=sda, scl=scl, freq=1000000)
#initialisiere digitale Pins 4 und 5 als AUSGANG für die OLED-Kommunikation

oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)

measure=0

def getDistance():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value() == 0:
        signaloff = utime.ticks_us()
    while echo.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    return distance
#calculate distance

def airPiano():
    while True:
        global measure

        if measure>5 and measure<11:
            buzzer.duty_u16(4000)
            buzzer.freq(262)
            sleep(0.4)

        elif measure>10 and measure<16:
            buzzer.duty_u16(4000)
            buzzer.freq(294)
            sleep(0.4)

        elif measure>15 and measure<21:
            buzzer.duty_u16(4000)
            buzzer.freq(330)
            sleep(0.4)
```