Dokumentation des Dezibot$_4$

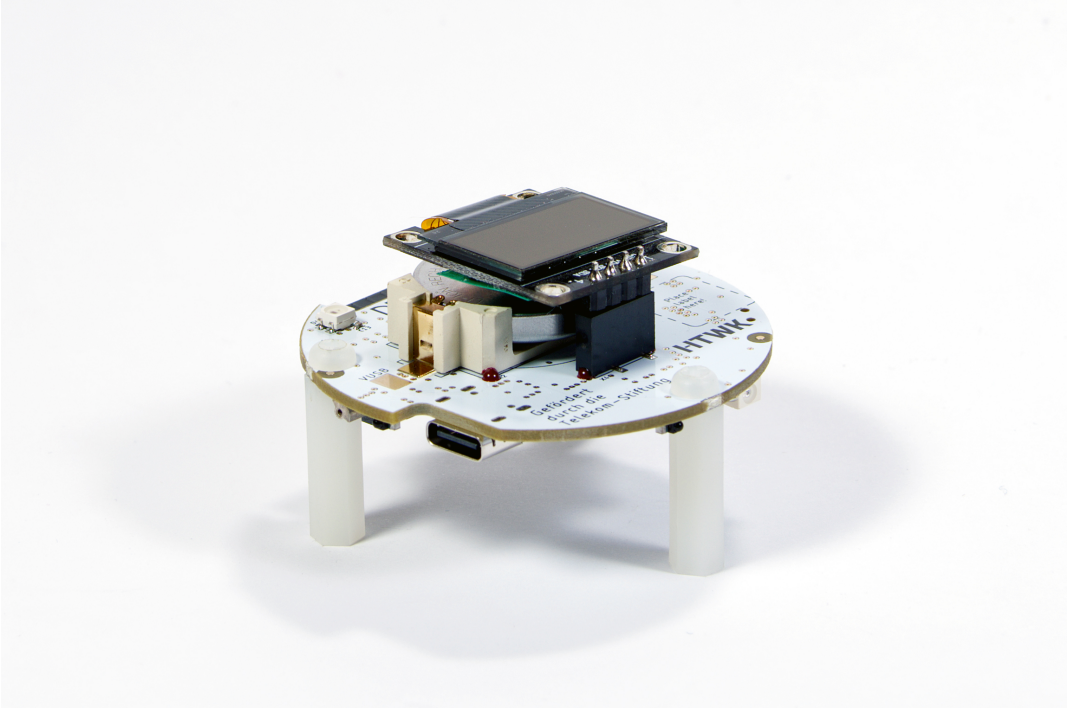
Chris-Dominik Fingerle, Estelle Bianca Wilfer

26. Dezember 2024

Zusammenfassung

Dieses Dokument richtet sich an die Anwenderinnen und Anwender des Bildungsroboters Dezibot$_4$, einem Lehr- und Lernroboter, entwickelt in einem Forschungsprojekt der Hochschule für Technik, Wirtschaft und Kultur Leipzig. Das Projekt wurde durch die Deutsche Telekom Stiftung gefördert.

Die vorliegende Dokumentation soll ein grundlegendes Verständnis für den Dezibot$_4$ schaffen sowie einen einführenden Einblick in Programmierprinzipien ermöglichen. Nachdem zu Beginn auf allgemeine Hinweise und Sicherheitsanweisungen eingegangen wird, erfolgt in Kapitel 3 eine detaillierte Beschreibung der auf dem Dezibot$_4$ integrierten Hardware. Im Anschluss wird die erste Inbetriebnahme des Roboters sowie die Einrichtung der erforderlichen Programmierumgebung vorgestellt. Für ein besseres Verständnis während des Programmierens erfolgt in Kapitel 6 eine kleine Einführung in Programmierprinzipien. Schließlich werden diese Prinzipien anhand unterschiedlicher Beispielprogramme nähergebracht sowie die Außerbetriebnahme und Entsorgung des Dezibot$_4$ beschrieben.



Inhaltsverzeichnis

1 Vorbemerkungen

Herzlich Willkommen zum Dezibot$_4$, den von der Hochschule für Technik, Wirtschaft und Kultur Leipzig entwickeltem Lehr- und Lernroboter in der vierten Generation. Wir freuen uns, Ihnen einen Einblick in die faszinierende Welt des Programmierens zu ermöglichen.

1.1 Vorstellung des Dezibot$_4$

Der Dezibot$_4$ ist ein kleiner Roboter, der speziell für den Einsatz in Schulen entwickelt wurde. Die grundlegende Idee ist, Lernenden das Programmieren auf anschauliche und praxisnahe Weise zu vermitteln. Durch die interaktive Nutzung des Dezibot$_4$ haben Schülerinnen und Schüler nicht nur die Gelegenheit, Kenntnisse in der Robotik zu erlangen, sondern auch ihre technologischen Fähigkeiten zu erweitern.

Das Konzept „Eine Schülerin – ein Roboter“ bzw. „Ein Schüler – ein Roboter“ ermöglicht jedem, seinen eigenen Roboter zu haben und somit individuelle Programmiererfahrungen zu sammeln. Unter Verwendung konstruktivistischer Prinzipien haben die Lernenden die Möglichkeit, ihre Robotik-Projekte eigenständig zu gestalten und dabei ihre eigenen Interessen und kreativen Ideen zu verfolgen. Dies fördert nicht nur die Programmierkompetenz, sondern auch die Selbstständigkeit und Eigeninitiative der Schülerinnen und Schüler im informatischen Bereich.

1.2 Beitrag leisten

Wir heißen jederzeit gerne neue Teammitglieder herzlich willkommen, die unser Dezibot$_4$-Projekt weiter voranbringen möchten! Unabhängig von Ihrer Erfahrung in den Bereichen Robotik, Programmierung oder im Bildungswesen schätzen wir jede Form der Unterstützung.

Falls Sie Interesse daran haben, sich aktiv einzubringen, treten Sie unserem Discord-Server bei (`https://discord.gg/AcbhG3FSqY`) oder kontaktieren Sie uns gerne unter:

`info@dezibot.de`

1.3 Zweckmäßige Verwendung

Bevor wir uns der Programmierung zuwenden, ist es wichtig, einige grundlegende Sicherheitshinweise zu beachten. Der Dezibot$_4$ wurde mit dem Fokus auf sicheres Lernen entwickelt. Bitte nehmen Sie jedoch die folgenden grundlegenden Sicherheitshinweise zur Kenntnis:

- **Sicherheit an erster Stelle:** Der Dezibot$_4$ darf nur unter Aufsicht verwendet werden. Verwenden Sie den Roboter nicht, wenn seine Funktion potenzielle Gefahren darstellen könnte.
- **Schutz von Werten und Leben:** Verwenden Sie den Dezibot$_4$ nicht in Situationen, in denen finanzielle Werte gefährdet werden könnten oder wenn die Gesundheit oder das Leben von Tieren oder Menschen davon abhängt.
- **Nicht für technische Prozesse verwenden:** Der Dezibot$_4$ sollte nicht zur Steuerung oder Überwachung von technischen Prozessen eingesetzt werden.
- **Nicht unbeaufsichtigt betreiben:** Lassen Sie den Dezibot$_4$ nicht unbeaufsichtigt und betreiben Sie ihn nicht permanent. Dies gewährleistet, dass Sie stets die Kontrolle über den Roboter behalten.

1.4 Haftungsausschluss

Dieser Haftungsausschluss wurde gemäß den Bestimmungen der **GNU General Public License (GPL)** [1], insbesondere unter Berücksichtigung der §§ 15, 16 und 17 erstellt. Bitte lesen Sie die nachfolgenden Bestimmungen sorgfältig durch, um ein vollständiges Verständnis für Ihre Rechte und Pflichten im Zusammenhang mit der Verwendung dieses Roboters zu erhalten.

Bitte beachten Sie, dass dieser Haftungsausschluss keine rechtliche Beratung darstellt. Im Zweifelsfall wird empfohlen, Rechtsbeistand in Anspruch zu nehmen, um sicherzustellen, dass alle rechtlichen Anforderungen erfüllt sind.

§ 15 Gewährleistungsausschluss

Es besteht keinerlei Gewährleistung für den Roboter sowie der mitgelieferten Software, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Urheberrechtsinhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit einschließlich – aber nicht begrenzt auf – die implizite Gewährleistung der Marktreife oder der Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

§ 16 Haftungsbegrenzung

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Urheberrechtsinhaber oder irgendein Dritter, der das Programm wie oben erlaubt, modifiziert oder übertragen hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Urheberrechtsinhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

§ 17 Interpretation von §§ 15 und 16

Sollten der o. a. Gewährleistungsausschluss und die o. a. Haftungsbegrenzung aufgrund ihrer Bedingungen gemäß lokalem Recht unwirksam sein, sollen Bewertungsgerichte dasjenige lokale Recht anwenden, das einer absoluten Aufhebung jeglicher zivilen Haftung in Zusammenhang mit dem Programm am nächsten kommt, es sei denn, dem Programm lag eine entgeltliche Garantieerklärung oder Haftungsübernahme bei.

Weitere Hinweise

Darüber hinaus ist zu beachten, dass die Leistungs- und Lichtintensität durch die aktuelle Firmware reduziert wird. Bei einer Manipulation der Firmware kann die zugesicherte geminderte Leistungs- und Lichtintensität nicht gewährleistet werden. Infolgedessen könnte die Lichtintensität bei einem geringen Betrachtungsabstand für das menschliche Auge zu hoch sein.

2 Vorsorgemaßnahmen und Sicherheitsanweisungen

Der Dezibot$_4$ muss während seiner Arbeit stets beaufsichtigt werden und darf nicht autonom arbeiten. Leitfähige Gegenstände (z. B. Büroklammern) und Flüssigkeiten sollten vom Roboter stets ferngehalten werden.

Der Dezibot$_4$ entspricht der europäischen CE-Norm und wurde von einem unabhängigen Prüflabor bestätigt. Der Roboter ist frei von gefährlichen Substanzen gemäß der Europäischen RoHS-Norm (Restriction of Hazardous Substances Directive 2002/95/EC).

2.1 Bauteile

Auf der Oberfläche des Dezibot$_4$ befinden sich empfindliche Bauteile. Er ist vor jeder Benutzung in Augenschein zu nehmen. Beschädigte oder veränderte Roboter müssen ersetzt werden und entsprechen nicht mehr den Anforderungen für eine sichere Verwendung.

Er ist nicht für Kinder unter 12 Jahren geeignet, da er verschluckbare Kleinteile enthält. Im Falle des Verschluckens von Teilen ist umgehend eine Ärztin oder ein Arzt zu konsultieren.

Der Dezibot$_4$ enthält an der Oberfläche Schwermetalle wie Kupfer, Silber, Gold und Zinn. Im Falle von Allergien gegen diese Stoffe sollte er nicht berührt werden.

2.2 Lithiumpolymerakku

Als Energiequelle benutzt der Dezibot$_4$ einen im Handel erhältlichen Lithium-Polymer-Akku vom Typ **LIR2450**. Der Roboter ist nur für diesen Akkutyp geeignet. Bitte setzen Sie keine Akkus anderer Norm oder Batterien der gleichen Größe ein (z. B. CR2450)!

Der Akku ist nach einem Jahr oder 1000 Ladezyklen bzw. nach Herstellerangaben zu ersetzen. Er muss mindestens einmal alle 6 Monate vollständig geladen oder ersetzt werden, da sonst die Gefahr der Beschädigung durch Tiefenentladung besteht. Der Akku ist nach der Benutzung wieder vollständig über ein USB-C-Kabel zu laden (siehe Nr. 15 in Abb. 2).

Bei falscher Handhabung geht von allen Lithiumakkumulatoren eine Brandgefahr aus.

2.3 Versand und Lagerung

Beim Postversand ist die Verpackung durch die Gefahrgutkennzeichnung für Lithium-Akkumulatoren zu kennzeichnen. Gemäß den örtlichen Vorschriften sind bei der Lagerung größerer Mengen von Robotern entsprechende Vorkehrungen für die Lagerung von Gefahrgut zu treffen.

3 Hardware des Dezibot$_4$

In diesem Kapitel wird die auf dem Dezibot$_4$ integrierte Hardware vorgestellt.

Um einen tieferen Einblick in die einzelnen Bauteile zu erhalten, finden Sie in jedem Abschnitt weiterführende Literatur sowie verlinkte Datenblätter, welche die technischen Spezifikationen der Komponenten beschreiben.

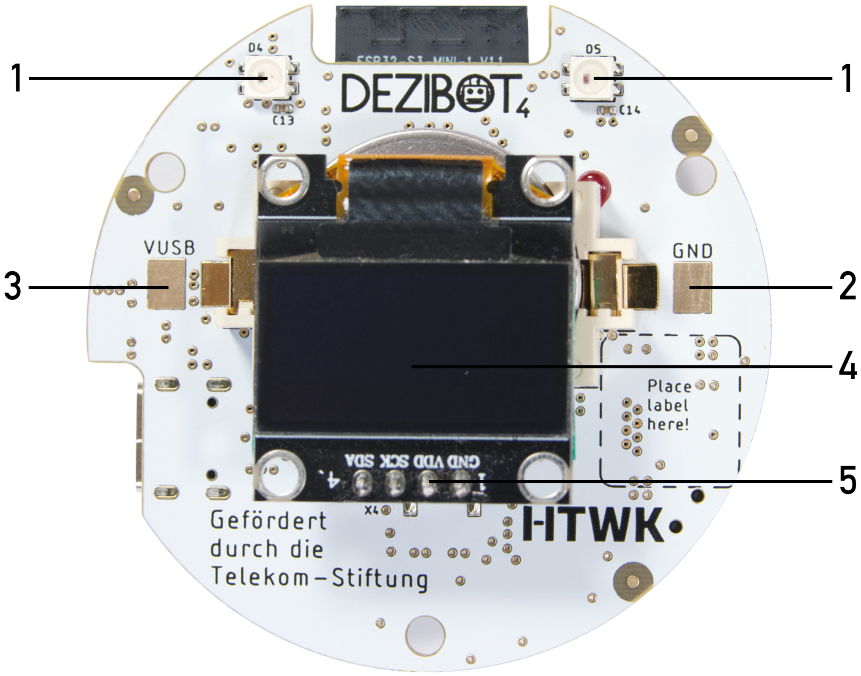


Abbildung 1: Ansicht des Dezibot$_4$ von oben

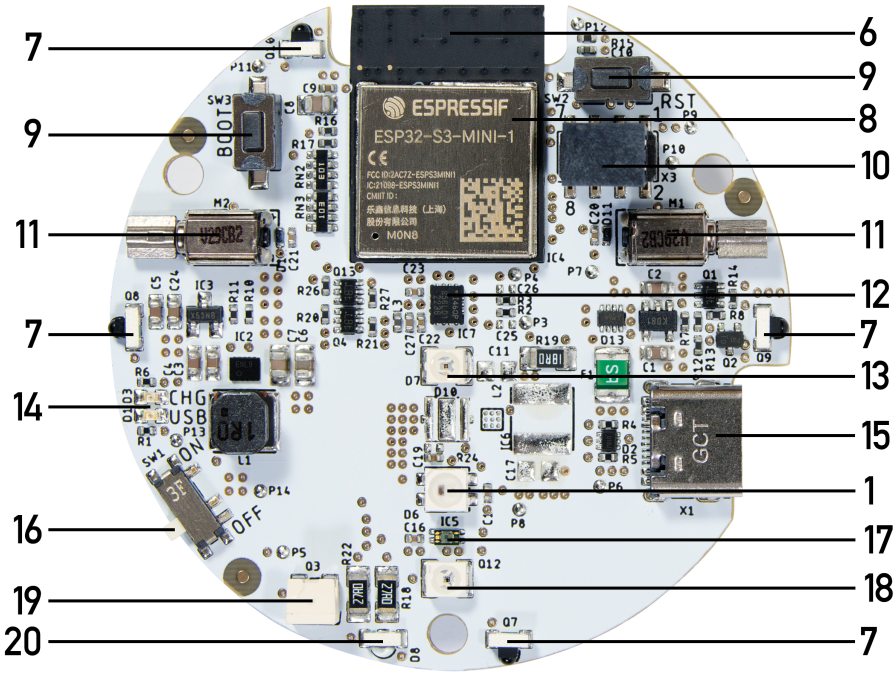


Abbildung 2: Ansicht des Dezibot$_4$ von unten (ohne Beine)

1	RGB-LEDs	**11**	Motoren
2	GND-Pin	**12**	IMU-Sensor
3	5 V-Pin	**13**	IR-LED 1
4	OLED-Display	**14**	Batteriestatus
5	I^2C-Anschluss	**15**	USB-C-Anschluss
6	WLAN-Antenne	**16**	An- und Ausschalter
7	IR-Sensoren	**17**	RGBW-Farbsensor
8	ESP32-S3-MINI-1	**18**	Tageslichtsensor 1
9	BOOT- und RST-Taster	**19**	Tageslichtsensor 2
10	Erweiterungsschnittstelle	**20**	IR-LED 2

3.1 Mikrocontroller

Ein Mikrocontroller [2] ist ein integrierter Schaltkreis, der neben einem Prozessor mindestens Speicher- und Peripheriebaugruppen enthält. Er vereint auf kleinstem Raum eine Vielzahl von Funktionen, die die Ausführung von Programmen und die Kontrolle von Peripheriegeräten ermöglichen. Der Kern eines Mikrocontrollers besteht in der Regel aus einem zentralen Prozessor, bekannt als Central Processing Unit (CPU), welcher von Speicher- sowie Input- und Output-Komponenten begleitet wird.

Mikrocontroller finden Anwendung in zahlreichen elektronischen Geräten, von Haushaltsgeräten über Unterhaltungselektronik bis hin zu industriellen Steuerungssystemen. Sie sind in der Lage, komplexe Aufgaben zu bewältigen und reagieren auf externe Signale, um entsprechende Aktionen auszuführen. Durch ihre geringe Größe und niedrigen Energieanforderungen eignen sich Mikrocontroller besonders gut für batteriebetriebene oder mobile Anwendungen.

Die Programmierung von Mikrocontrollern erfolgt in der Regel mithilfe von speziellen Entwicklungsumgebungen und Programmiersprachen. Hierbei werden Befehle festgelegt, die vom Prozessor interpretiert und ausgeführt werden, um die gewünschten Funktionen zu realisieren.

3.1.1 ESP32-S3-MINI-1

Der auf dem Dezibot_4 integrierte ESP32-S3-MINI-1 [3] (siehe Nr. 8 in Abb. 2) ist ein leistungsstarker und vielseitiger Mikrocontroller. Er wurde von Espressif Systems entwickelt und basiert auf der Xtensa Dual-Core 32-Bit LX7 CPU, die mit bis zu 240 MHz getaktet werden kann. Die Dual-Core-Architektur ermöglicht die parallele Ausführung von Aufgaben und erhöht die Gesamtleistung des Mikrocontrollers.

Ein herausragendes Merkmal des ESP32-S3-MINI-1 ist seine integrierte Unterstützung für drahtlose Kommunikationsstandards wie Wi-Fi und Bluetooth. Zusätzlich zu den drahtlosen Funktionen verfügt der ESP32-S3-MINI-1 über eine breite Palette von Peripheriegeräten und Schnittstellen, darunter GPIO-Pins [4], UART [5], I^2C [6], SPI [7] und mehr. Dies macht ihn äußerst flexibel und anpassungsfähig für unterschiedlichste Anwendungen.

Übersicht der wichtigsten Spezifikationen

- Xtensa Dual-Core 32-Bit LX7 CPU, bis zu 2,4 GHz
- Speicher: 384 KB ROM, 512 KB SRAM, bis zu 8 MB Flash
- Serielle Schnittstellen: ADC, GPIO, I^2C, I^2S, SPI, PWM, UART, USB
- Betriebsspannung: 3 $V \sim$ 3,6 V
- Bluetooth, Wi-Fi

3.2 Optik

3.2.1 RGB-Leuchtdioden (LED)

RGB-LEDs [8] sind vielseitige Leuchtdioden, die Licht in verschiedenen Farben erzeugen können. Jede dieser LEDs integriert drei separate Lichtquellen in einem Gehäuse, eine rote, grüne und blaue. Diese Farben bilden die Grundfarben im RGB-Farbraum. Durch präzise Steuerung der Helligkeit jeder dieser LEDs lassen sich unterschiedliche Farbmischungen erzielen.

WS2812B-Mini-V2

Auf dem Dezibot$_4$ wurden drei WS2812B-MINI-V2 [9] LEDs eingebaut (siehe Nr. 1 in Abb. 1 und Abb. 2). Diese integrieren jeweils drei LEDs in verschiedenen Farben sowie einen Controller in einem einzigen Gehäuse. Dadurch wird ermöglicht, dass jede LED individuell mit einer eigenen Pulsweitenmodulation [10] angesteuert werden kann. Da der LED-Chip einen eigenen Controller besitzt, kostet diese Pulsweitenmodulation keine Prozessorleistung.

Das bedeutet jedoch auch, dass die LEDs nicht automatisch ausgeschaltet werden, wenn sie für eine spezifische Aufgabe programmiert sind und gleichzeitig ein anderes, nicht mit den LEDs verknüpftes Programm auf dem Roboter läuft. Um diese explizit zu deaktivieren, müssen sie entweder erneut direkt angesprochen oder der Dezibot$_4$ muss durch Ein- und Ausschalten zurückgesetzt werden (siehe Nr. 16 in Abb. 2).

Darüber hinaus ist zu beachten, dass die Leistungs- und Lichtintensität durch die aktuelle Firmware reduziert wird. Bei einer Manipulation der Firmware kann die zugesicherte geminderte Leistungs- und Lichtintensität nicht gewährleistet werden. Infolgedessen könnte die Lichtintensität bei einem geringen Betrachtungsabstand für das menschliche Auge zu hoch sein.

3.2.2 RGBW-Farbsensor

Ein RGBW-Farbsensor [11] ist ein Sensor, der dazu dient, Farbinformationen zu erfassen, jedoch im Vergleich zu herkömmlichen RGB-Sensoren eine zusätzliche Komponente für Weißlicht integriert. Die Abkürzung RGBW steht für die Grundfarben Rot, Grün, Blau und Weiß. Typischerweise sind RGB-Sensoren darauf ausgelegt, Licht in den drei Hauptfarben Rot, Grün und Blau zu messen und auf dieser Grundlage eine Vielzahl von Farben zu reproduzieren.

VEML6040

Auf dem Dezibot$_4$ wurde ein VEML6040 [12] RGBW-Farbsensor verbaut (siehe Nr. 17 in Abb. 2). Dieser erfasst rotes, grünes, blaues sowie weißes Licht und vereint Fotodioden, Verstärker und analoge bzw. digitale Schaltungen in einem einzigen Chip im CMOS-Verfahren [13].

3.2.3 Infrarot (IR) LED

IR-LEDs [14] sind Leuchtdioden, die Infrarotlicht erzeugen. Im Gegensatz zu sichtbarem Licht liegt die Wellenlänge des von IR-LEDs erzeugten Lichts außerhalb des für den Menschen sichtbaren Bereichs.

Infrarot-LEDs finden in der Robotik vielfältige Anwendungen, insbesondere im Bereich der Sensortechnologie und Navigation. Mögliche Einsatzgebiete sind:

- **Kommunikation**: IR-LEDs können für die drahtlose Kommunikation zwischen Robotern oder zwischen einem Roboter und einer Basisstation verwendet werden. Dies kann für kooperative Aufgaben oder den Datenaustausch zwischen Robotern in einem Netzwerk relevant sein.
- **Abstandssensoren**: IR-LEDs werden in Kombination mit IR-Empfängern eingesetzt, um die Entfernung zwischen dem Roboter und einem Hindernis zu messen. Durch die Auswertung der reflektierten Infrarotstrahlen kann der Roboter Hindernissen ausweichen oder sie erkennen.
- **Erkennung der relativen Position der Roboter zueinander:** Roboter können mit IR-LEDs und IR-Empfängern ausgestattet sein, um die Reflexion von Infrarotlicht an Oberflächen zu erkennen. Wenn ein Roboter Infrarotlicht auf eine Oberfläche sendet und das reflektierte Licht von einem anderen Roboter erfasst wird, kann die relative Position geschätzt werden.

Die Verwendung von Infrarot-LEDs in der Robotik ermöglicht es, Informationen über die Umgebung zu sammeln, ohne auf sichtbares Licht angewiesen zu sein.

VSMY3850X01

Auf dem Dezibot$_4$ wurde eine VSMY3850X01 [15] IR-LED verbaut (siehe Nr. 13 in Abb. 2). Die nach unten gerichtete IR-LED strahlt einen Infrarot-Lichtstrahl aus, der von der Bodenoberfläche reflektiert wird. Das reflektierte Licht wird gleichmäßig in alle Richtungen verteilt, außer in den Bereichen, die im Schatten der Beine des Roboters liegen. Dies ermöglicht die Selbstortung des Dezibot$_4$ oder die Lokalisierung anderer Roboter in seiner Umgebung.

B2141IR–A1C000373U1930

Zudem wurde auf dem Dezibot$_4$ eine B2141IR–A1C000373U1930 [16] IR-LED eingebaut (siehe Nr. 20 in Abb. 2). Die nach vorne gerichtete IR-LED arbeitet in Verbindung mit dem Tageslichtsensor (siehe Nr. 19 in Abb. 2 sowie Kapitel 3.2.4) und dient der Messung des Abstands zu potenziellen Objekten in unmittelbarer Umgebung.

3.2.4 Fototransistoren

Fototransistoren sind elektronische Schalter, die leitfähig werden, wenn Licht auf sie fällt. Sie bestehen im Wesentlichen aus einer Fotodiode [17], die Licht in elektrische Ladung umwandelt, und einem Transistor [18], der diese Ladung in einen elektrischen Strom verstärkt.

Fototransistoren können vielseitig eingesetzt werden, entweder als Schalter mit zwei Zuständen oder zur Erfassung kontinuierlicher Werte. Auf dem Dezibot$_4$ gibt es drei unterschiedliche Fototransistortypen, die alle einer eigenen Funktion nachgehen.

PT26-51B

Der Dezibot$_4$ integriert vier PT26-51B [19] IR-Sensoren (siehe Nr. 7 in Abb. 2). Sie erfassen Infrarotlicht mit einer Wellenlänge zwischen 730 und 1100 nm in jede der vier Himmelsrichtungen. Durch diese Sensoren ist es dem Dezibot$_4$ möglich, die Position von anderen Robotern wahrzunehmen, die durch die nach unten gerichtete IR-LED Licht streuen (siehe Nr. 13 in Abb. 2 sowie Kapitel 3.2.3).

SFH320

Des Weiteren wurde auf dem Dezibot$_4$ ein nach unten gerichteter SFH320 [20] Tageslichtsensor eingebaut (siehe Nr. 18 in Abb. 2). Er erfasst Licht mit einer Wellenlänge von 450 bis 1150 nm.

SFH325

Zudem besitzt der Roboter einen nach vorne gerichteten SFH325 [21] Tageslichtsensor (siehe Nr. 19 in Abb. 2). Dieser erfasst ebenfalls Licht mit einer Wellenlänge von 450 bis 1120 nm. In Kombination mit der eingebauten IR-LED, die parallel zum Sensor ausgerichtet ist (siehe Nr. 20 in Abb. 2 sowie Kapitel 3.2.3), besteht die Möglichkeit, Abstände in diese Richtung wahrzunehmen und zu messen.

3.2.5 OLED-Display

Ein Organic Light Emitting Diode (OLED) Display [22] ist eine Bildschirmtechnologie, die organische Verbindungen für die Lichtemission verwendet. Das heißt, es sind spezielle Moleküle oder Polymere in dünnen Schichten auf dem Display angeordnet. Diese Verbindungen emittieren Licht, wenn sie elektrischer Spannung ausgesetzt sind. Im Gegensatz zu herkömmlichen LC-Displays benötigen OLED-Displays keine Hintergrundbeleuchtung, da sie selbst leuchten, wenn Strom durch sie hindurch fließt.

OLED 128×64 0.96” I2C

Es wird empfohlen, den Dezibot$_4$ in Kombination mit einem OLED 128×64 0.96” I^{2}C [23] Display mit einer Größe von 128×64 Pixeln und einer Diagonale von 0,96 Zoll (siehe Nr. 4 in Abb. 1) zu verwenden. Dieses Display lässt sich über die I^{2}C-Schnittstelle anschließen (siehe Nr. 5 in Abb. 1 sowie Kapitel 3.4.2). Dabei muss jedoch beachtet werden, dass die Reihenfolge der Anschlüsse der Schnittstelle GND – VCC – SCL – SDA ist, welche auch von der Display-Komponente eingehalten werden muss. Da sich das Display direkt über dem Akku befinden würde, kann zur zusätzlichen Sicherheit auf der Rückseite des Displays ein Klebestreifen angebracht werden, um einen Kurzschluss zu vermeiden. Im Normalfall sollte dieser Klebestreifen jedoch nicht notwendig seinie , da die I^{2}C-Schnittstelle weit über dem Akku endet.

3.3 Kinematik

Die Kinematik beschäftigt sich mit der Analyse von Bewegungen und Positionen von Robotern. Ihr Fokus liegt darauf, wie die verschiedenen Komponenten eines Roboters interagieren, um ihn von einem Ort zum anderen zu bewegen. Kurz gesagt, die Kinematik umfasst alle Baugruppen, die sich entweder selbst bewegen oder die Bewegung des Roboters ermöglichen. Die Beweglichkeit und damit das Verrichten von Arbeit stellt den entscheidenden Unterschied zwischen Computern und Robotern dar. Der Dezibot$_4$ verfügt über zwei Vibrationsmotoren als einzige derartige Baugruppe.

Zusätzlich integriert der Roboter einen Inertial Measurement Unit (IMU) Sensor, der eine unterstützende Rolle bei der Steuerung übernimmt und mit den Motoren interagiert. Aus diesem Grund ist der IMU-Sensor nicht unmittelbar Bestandteil der Kinematik, wurde aber dennoch aus Gründen der Vollständigkeit in diesem Kapitel mit aufgeführt.

3.3.1 Motor

Unwucht- bzw. Vibrationsmotoren [24] sind spezielle Motoren, die ursprünglich dazu entwickelt wurden, Benutzern von Mobiltelefonen lautlose Nachrichten zu geben. Auf dem Dezibot$_4$ dienen die Motoren der Bewegung des Roboters. Der Mechanismus basiert auf dem Prinzip, dass eine nicht gleichmäßige Gewichtsverteilung zu einer Unwucht führt, wenn der Motor rotiert. Durch die Erzeugung von Unwuchten entstehen Vibrationen und der Motor sowie der Roboter werden in Bewegung versetzt.

VM-2702A2.7-SMD

Auf dem Dezibot$_4$ wurden zwei VM-2702A2.7-SMD [25] Motoren eingebaut (siehe Nr. 11 in Abb. 2). Durch präzise Steuerung der Vibrationen per Software bzw. Programmierung kann der Roboter gesteuert und gelenkt werden.

3.3.2 Inertiale Messeinheit

Der Inertial Measurement Unit (IMU) Sensor [26] ist ein hoch entwickeltes Motion-Tracking-Gerät, das speziell für die Erfassung und Messung von Bewegungen in sechs verschiedenen Richtungen entwickelt wurde. Dabei kombiniert es ein 3-Achsen-Gyroskop, das Rotationsbewegungen um drei verschiedene Achsen erkennt, mit einem 3-Achsen-Beschleunigungsmesser, der lineare Bewegungen und Beschleunigungen in drei verschiedenen Richtungen erfasst. Dies bedeutet, dass das Gerät in der Lage ist, Drehbewegungen und lineare Bewegungen in alle Richtungen zu verfolgen. Das ist besonders nützlich in Anwendungen, bei denen es wichtig ist, die genaue Bewegung und Ausrichtung eines Objekts oder Systems zu überwachen.

ICM-42670-P

Der Dezibot$_4$ ist mit einem ICM-42670-P [27] IMU-Sensor ausgestattet (siehe Nr. 12 in Abb. 2). Dieser zeichnet sich durch seine kompakte Bauweise, hohe Geschwindigkeit sowie kostengünstige Verfügbarkeit aus und bietet eine Vielzahl von Funktionen für die Erfassung von Bewegungsdaten. Es ist jedoch zu beachten, dass der Sensor kein Magnetometer enthält.

Trotz seiner komplexen Programmierung wurde diese bereits durch eine benutzerfreundliche Firmwarelösung vereinfacht.

3.4 Schnittstellen

Der Dezibot$_4$ lässt sich per USB-C laden sowie programmieren. Für weitere Programmieroptionen verfügt der Roboter über einen I^{2}C-Anschluss und eine Erweiterungsschnittstelle.

3.4.1 USB-C-Anschluss

Informationen zum Ladevorgang können Sie Kapitel 4.1 entnehmen.

Für die Programmierung wird der Dezibot$_4$ über den USB-C-Anschluss (siehe Nr. 15 in Abb. 2) mit einem USB-C-Kabel und einem Computer verbunden. Anschließend kann er in der Arduino IDE 2 mit einem Programm geflasht werden, wie ab Kapitel 5 beschrieben wird.

3.4.2 I^{2}C-Anschluss

Der Inter Integrated Circuit (I^{2}C) Anschluss [6] (siehe Nr. 5 in Abb. 1) ist ein serieller Datenbus, der es ermöglicht, mehrere elektronische Komponenten wie Sensoren, Aktuatoren oder Displays miteinander zu verbinden. Der Bus besteht aus einer Daten- und einer Taktleitung.

Wie bereits in Kapitel 3.2.5 erwähnt, kann das OLED-Display (siehe Nr. 4 in Abb. 1) über den I^{2}C-Anschluss angeschlossen und gesteuert werden. Dabei muss jedoch beachtet werden, dass die Reihenfolge der Anschlüsse der Schnittstelle GND – VCC – SCL – SDA ist, welche auch von der Display-Komponente eingehalten werden muss. Darüber hinaus ist es möglich, Peripheriegeräte wie beispielsweise Kameras, Mikrofone oder Lautsprecher mit dem Dezibot$_4$ zu verbinden. Dies eröffnet vielfältige Optionen für die Erweiterung und Anpassung der Funktionalitäten des Roboters.

3.4.3 Erweiterungsschnittstelle

Auf der unteren Seite des Dezibot$_4$ ist eine weitere serielle Schnittstelle [28] (siehe Nr. 10 in Abb. 2) zu finden. An dieser gibt es einen 3 V-Pin (Pin 1), zwei GND-Pins (Pin 7 und 8) und einen VBAT-Pin (Pin 2) für den Anschluss der Versorgungsspannung des eingebauten Akkus. Die anderen vier Anschlüsse sind universelle Ein- und Ausgangspins (Pin 3 bis 6), sogenannte General Purpose Input/Output (GPIO) Pins. Über sie können digitale Signale gesendet oder empfangen werden.

3.5 Kommunikation

Kommunikation ist ein wesentlicher Bestandteil der Steuerung und Interaktion mit dem Dezibot$_4$.

3.5.1 Bluetooth und WLAN

Für die Kommunikation mit dem Dezibot$_4$ und zwischen den Robotern untereinander besteht die Möglichkeit der Verbindung über Bluetooth [29] oder mit WLAN [30]. Die WLAN-Antenne ist nach hinten gerichtet (siehe Nr. 6 in Abb. 2). Wenn die Antenne mit den Händen oder Fingern abgedeckt wird, kann die WLAN-Verbindung des Roboters beeinträchtigt werden. Bitte berücksichtigen Sie dies bei der drahtlosen Kommunikation mit dem Roboter. Darüber hinausgehend verfügt der ESP32-S3-MINI-1 Mikrocontroller über die Fähigkeit, als Access-Point zu agieren. Dies geht jedoch mit einem hohen Energieverbrauch einher.

Es sei an dieser Stelle darauf hingewiesen, dass die Verwendung von Bluetooth mithilfe des Roboters nicht in der CE-Zertifizierung enthalten ist, weshalb sich auch keine derartigen Methoden in der von uns bereitgestellten Bibliothek finden werden. Verwendet der Nutzer dennoch das Bluetooth des Roboters, so trägt der Nutzer selbst das volle Risiko.

3.6 Weiteres

Wenn Sie sich tiefer mit dem Dezibot$_4$ beschäftigen möchten, stehen Ihnen zusätzliche nützliche Funktionen zur Verfügung, die im Folgenden vorgestellt werden.

3.6.1 GND- und 5 V-Pin

Auf der Oberseite des Dezibot$_4$ finden Sie zwei Kontakte, einen für die 5 V-Versorgungsspannung (siehe Nr. 3 in Abb. 1) und einen GND-Anschluss (siehe Nr. 2 in Abb. 1). Die 5 V liegen nur an, wenn der Roboter per USB-C-Kabel verbunden ist. Mit diesen könnte man zum Beispiel den Akku mit einer externen Spannungsquelle aufladen.

3.6.2 P-Points

Zudem gibt es auf der Unterseite des Dezibot$_4$ verschiedene P-Points (P3 bis P14), die gleichmäßig über die gesamte Fläche der unteren Platine verteilt sind. Diese P-Points erfüllen eine wichtige Funktion bei der Qualitätssicherung während der Herstellung. Da sie über die internen Anschlüsse hinausgehende Zugangsmöglichkeiten zum Dezibot$_4$ bieten, könnten sie für sehr erfahrene Programmierer von Interesse sein.

4 Erste Inbetriebnahme

Der neu ausgelieferte Roboter muss zuerst montiert werden. Dazu werden die Beine, wie auf dem unteren Bild zu sehen ist, angeschraubt. Die drei Beine werden dabei am besten von unten gegen die Bohrungen gehalten. Die Schrauben werden von oben eingesetzt und mit einem Schraubendreher angezogen.

Wenn versucht wird, die Schrauben von Hand zu halten und gleichzeitig die Beine auf der Unterseite zu drehen, besteht die Gefahr, dass die Motoren dauerhaft beschädigt werden.

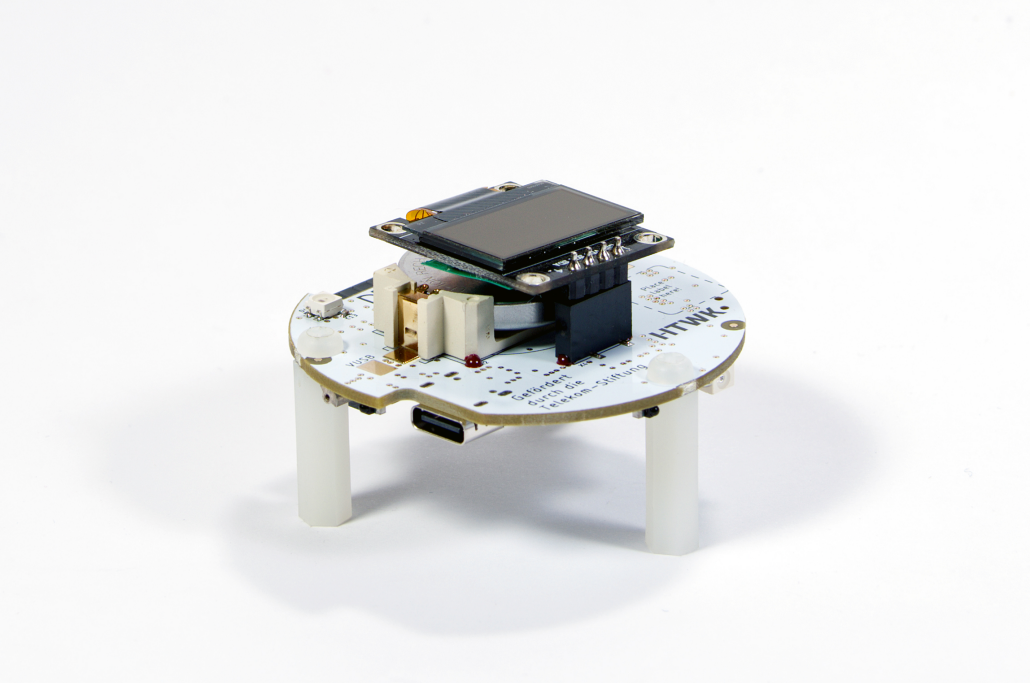


Abbildung 3: Ansicht des fertig montierten Dezibot$_4$ mit OLED-Display

Danach wird der Lithium-Polymer-Akku vom Typ LIR2450 eingesetzt. Der Dezibot$_4$ ist nur für diesen Akkutyp geeignet. Bitte setzen Sie keine Akkus anderer Norm oder Batterien der gleichen Größe ein (z. B. CR2450)!

Optional besteht die Möglichkeit das OLED-Display aufzustecken, wie es in Abb. 3 zu sehen ist.

4.1 Ladevorgang

Anschließend ist der Dezibot$_4$ über den USB-C-Anschluss (siehe Nr. 15 in Abb. 2) mit einem USB-C-Kabel zu laden. Wenn die Verbindung zu einem Steckernetzteil oder Computer hergestellt ist, leuchtet auf der unteren Seite eine grüne LED (siehe Nr. 14 in Abb. 2). Während des Ladevorgangs leuchtet die rote LED neben der grünen LED, um den Fortschritt anzuzeigen. Sobald der Akku vollständig aufgeladen ist, erlischt diese LED. Es ist nicht erforderlich, den Dezibot$_4$ während des Ladevorgangs einzuschalten. Sollte es dazu kommen, dass die rote LED nach dem Anschließen nicht leuchtet, kann es hilfreich sein, den Lithiumakku einmal aus- und wieder einzustecken.

4.2 Programmneustart

Auf der Unterseite des Dezibot$_4$ befinden sich zwei Taster namens BOOT und RST, mit denen verschiedene Funktionen möglich sind (siehe Nr. 9 in Abb. 2). Im Falle eines Programmabsturzes oder fehlerhaften Verhaltens des Programms kann dieses durch einmaligen Drücken von RST neu gestartet werden.

4.3 Bootmodus

Sollte es der Fall sein, dass der Dezibot$_4$ über USB-C nicht mehr kommuniziert, gibt es die Möglichkeit, ihn neu zu starten. Dafür halten Sie zunächst BOOT für zwei Sekunden gedrückt und dann ohne loszulassen für eine Sekunde RST. Lassen Sie zuerst RST, dann BOOT los. Nun sollte sich der Dezibot$_4$ im Bootmodus befinden.

4.4 Bootloader zurücksetzen

Der Bootloader ist die Software, die die Kommunikation des Dezibot$_4$ über den USB-C-Anschluss ermöglicht. Wurde dieser versehentlich überschrieben, besteht die Möglichkeit, ihn zurückzusetzen. Das Überschreiben des Bootloaders erkennt man meist daran, dass der per USB-C-Kabel angeschlossene Roboter am Computer nicht erkannt wird.

Halten Sie zunächst RST für zwei Sekunden gedrückt und dann ohne loszulassen für eine Sekunde BOOT. Lassen Sie zuerst BOOT, dann RST los. Der Bootloader sollte jetzt zurückgesetzt sein.

4.5 Memory reset

Sollte der Upload mnerfach fehlschlagen kann dies meist durch das leeren des Speichers behoben werden. Laden Sie hierzu das Programm `esptool` für Ihr System herunter, anschließend kann mit Listing 1 der Speicher geleert werden.

Listing 1: Memory reset

```
esptool.py --chip esp32s3 --port /dev/cu.usbmodem1301 erase_flash
```

5 Einrichtung der Entwicklungsumgebung

Bevor Sie in die Welt des Programmierens eintauchen, gibt es ein paar Voraussetzungen, die Sie erfüllen müssen. Sie werden in der Arduino Integrated Development Environment (IDE) 2 programmieren. Als vielseitiger Editor ermöglicht die Arduino IDE 2 die direkte Installation von Bibliotheken, die Synchronisierung von Sketches mit der Arduino Cloud, das Debuggen von Code und vieles mehr.
Die Mindestanforderungen für die Installation der Arduino IDE 2 sind:

- Windows - Win 10 und neuere Versionen, 64 Bits
- Linux - 64 Bits
- macOS - Version 10.15: „Catalina“ und neuere Versionen, 64 Bits

Sollten Sie weiterführende Informationen zur Installation benötigen, als die im Folgenden aufgeführten, so finden Sie diese unter folgendem Link:

`Weitere Hinweise zu Download und Installation` [31]

5.1 Installation

5.1.1 Windows

Wählen Sie die passende Version für Ihr Windows-Betriebssystem aus und laden Sie sie herunter:

`Arduino Software` [32]

Nachdem der Download abgeschlossen ist, öffnen Sie die heruntergeladene Datei und folgen Sie den Anweisungen auf dem Bildschirm. Die Installation kann einige Minuten in Anspruch nehmen. Sobald sie abgeschlossen ist, können Sie den nächsten Schritten aus Kapitel 5.2 folgen.

5.1.2 Linux

Um die Arduino IDE 2 unter Linux zu installieren, laden Sie zunächst das `AppImage 64 bits (X86-64)` von der Arduino Software-Seite herunter:

`Arduino Software` [32]

Bevor der Editor gestartet werden kann, muss er zunächst zu einer ausführbaren Datei gemacht werden. Dies erfolgt durch:

1. Rechtsklick auf die heruntergeladene Datei
2. Auswahl von `Eigenschaften`
3. Auswahl der Registerkarte `Zugriffsrechte`
4. Aktivieren des Kontrollkästchens `Datei als Programm ausführen`
5. Schließen des Fensters

Sie können nun auf die Datei doppelklicken, um die Arduino IDE 2 auf Ihrem Linux-Rechner zu starten. Falls Sie die AppImage-Datei nicht ausführen können, stellen Sie sicher, dass FUSE auf Ihrem System installiert ist.

Filesystem in USErspace (FUSE) ist ein Linux-Kernel-Modul, das es ermöglicht, Dateisystem-Treiber aus dem Kernel-Modus in den User-Modus zu verlagern. Es erlaubt auch nicht-privilegierten Benutzern eigene Dateisysteme zu aktivieren („mount“). Da die Dateisystem-Treiber im User-Modus laufen, braucht man sich beim Programmieren nicht mit den Beschränkungen und Besonderheiten des Kernel-Modes auseinanderzusetzen.

Eine Anleitung zur Installation von FUSE für Ihre Linux-Distribution finden Sie hier:

`FUSE Installation` [33]

Um die Verbindung zwischen der Arduino IDE 2 und der seriellen Schnittstelle für das Hochladen von Code auf Ihr Board zu ermöglichen, kann es hilfreich sein, die folgende Regel in der Datei `/etc/udev/rules.d/99-arduino.rules` hinzuzufügen:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="2341", GROUP="plugdev", MODE="0666"
```

Bitte führen Sie nun die folgenden Schritte von Kapitel 5.2 aus.

5.1.3 MacOS

Wählen Sie die passende Version für Ihr macOs-Betriebssystem aus und laden Sie diese herunter:

`Arduino Software` [32]

Nachdem der Download abgeschlossen ist, öffnen Sie die heruntergeladene Datei und ziehen Sie das Programm in Ihren Anwendungs- bzw. Programmordner. Anschließend können Sie den Schritten aus Kapitel 5.2 folgen.

5.2 Setup

Die Arduino IDE 2 kann jetzt durch Doppelklicken geöffnet werden. Nach dem Öffnen werden Sie mit einer leeren Entwicklungsumgebung begrüßt, in der Sie Ihre Arduino-Programme schreiben und hochladen können.

5.2.1 Board

Um mit den Arduino-Boards arbeiten zu können, muss die entsprechende Board-Unterstützung installiert werden. Dazu klicken Sie auf die Registerkarte `Werkzeuge`, dann `Board` und anschließend auf `Boards verwalten`.

Nun befinden Sie sich im Boardmanager der Arduino IDE 2. Suchen Sie die `esp32 von Espressif Systems`-Extension **Version 2.0.15** und installieren Sie diese (siehe die zweite Erweiterung in Abb. 4). Um die Erweiterung zu verwenden, führen Sie bitte die folgenden Schritte aus:

1. Wählen Sie in der Registerkarte `Werkzeuge` aus
2. Dann wählen Sie `Board` und anschließend `esp32`
3. Wählen Sie nun aus der Liste das `ESP32-S3-USB-OTG Board` (siehe Abb. 5)

Die Arduino IDE 2 ist jetzt für das erforderliche Board eingerichtet.

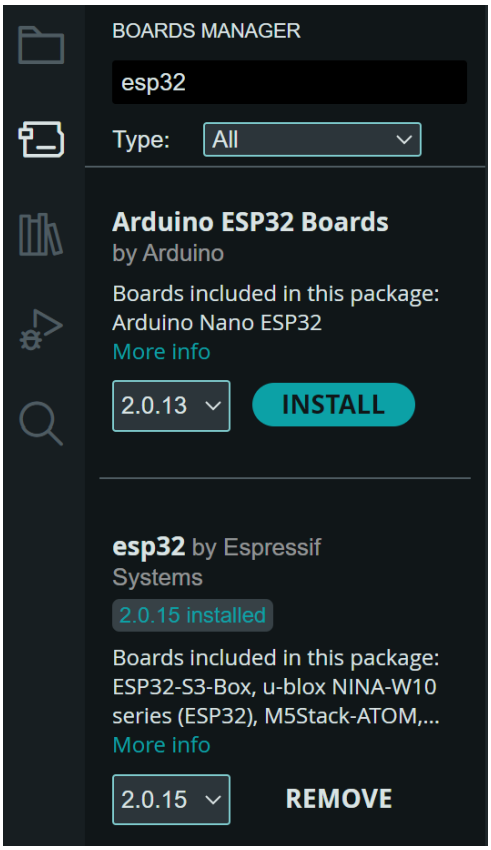


Abbildung 4: Boardmanager und `esp32 von Espressif Systems`

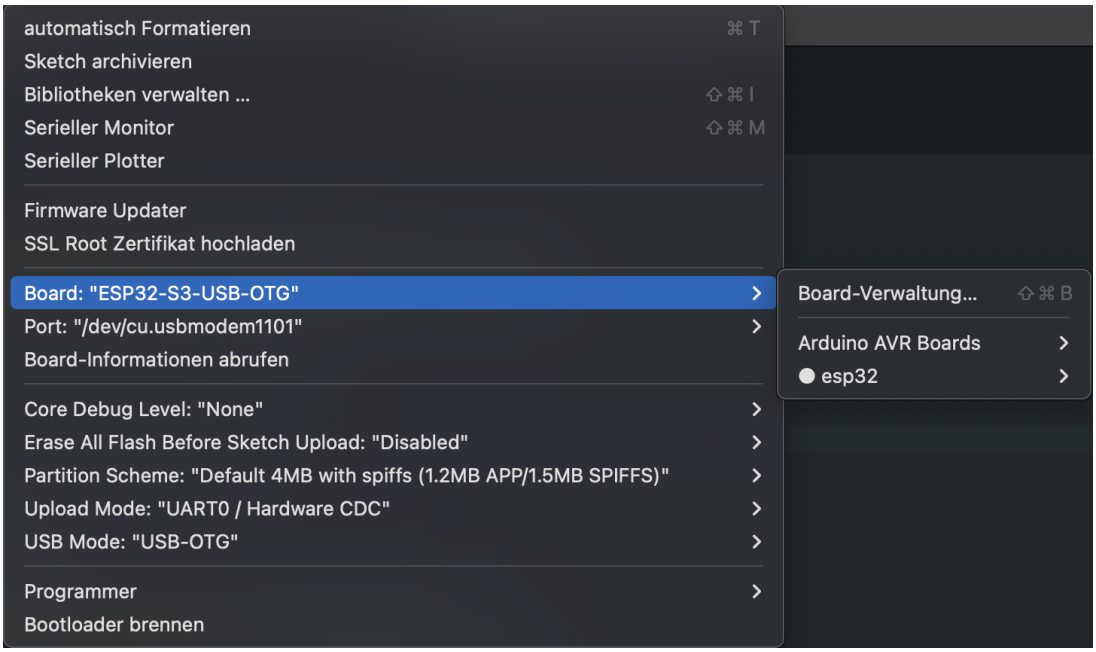


Abbildung 5: Einstellen des erforderlichen Boards `ESP32-S3-USB-OTG`

5.2.2 Bibliotheken

Um die erforderliche Bibliothek für die Programmierung des Dezibot$_4$ installieren zu können, führen Sie bitte folgende Schritte aus:

1. Öffnen von `https://github.com/dezibot/dezibot`
2. Download der Bibliothek als zip-Datei
3. Wählen Sie in in Arduino die Registerkarte `Sketch` aus
4. Dann wählen Sie `Bibliothek einbinden` und anschließend `.ZIP-Bibliothek hinufügen...`
5. Navigieren Sie in den Ordner, in dem die Bibliothek heruntergeladen wurde
6. Doppelklicken Sie auf die zip-Datei, um diese zu installieren

Die Dezibot-Bibliothek enthält bereits Lösungen zu von uns bereit gestellten Arbeitsmaterialien. Wenn die Schülerinnen und Schüler diese Lösungen nicht auf ihren Computern zur Verfügung haben sollen, müssen diese vor dem Einbinden der Bibliothek (also zwischen Schritt 2 und 3) entfernt werden. Dazu muss im Datei-Explorer zunächst in den Ordner, in dem die Bibliothek heruntergeladen wurde, navigiert werden. Anschließend muss in der Zip-Datei in den Ordner `examples` navigiert werden. Nun müssen Sie nur noch den Ordner `advanced` löschen und es kann dann wieder mit Schritt 3 fortgefahren werden.

Weiterhin wird für die Verwendung des Dezibots die Bibliothek `Adafruit NeoPixel von Adafruit` benötigt. Dazu klicken Sie auf die Registerkarte `Sketch`, dann `Bibliothek einbinden` und anschließend auf `Bibliothek verwalten...`.

Sie befinden sich nun im Bibliotheksmanager der Arduino IDE 2. Suchen Sie die `Adafruit NeoPixel von Adafruit`-Bibliothek und installieren Sie diese (siehe Abb. 6).

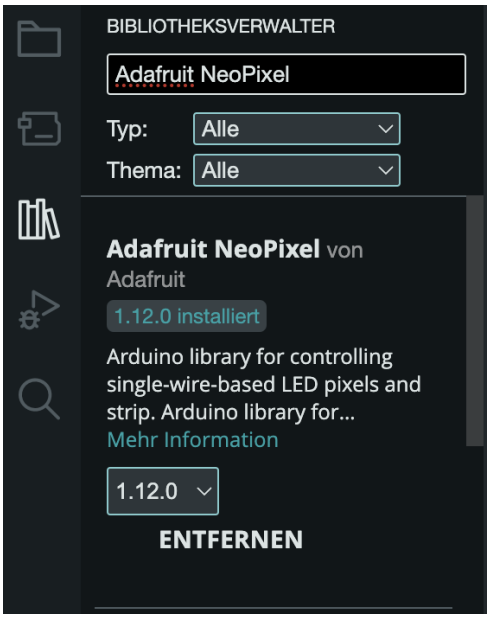


Abbildung 6: Bibliotheksmanager und `Adafruit NeoPixel von Adafruit`

Ebenfalls benötigt wird die Biblothek `Painless Mesh by Coopdis` (Version **1.5.0**), sowie die Biblothek `AsyncTCP by dvarrel` (Version **1.1.4**). Gehen Sie bei der Installation genauso vor, wie für die `Adafruit NeoPixel von Adafruit`-Bibliothek.

Jetzt sind alle erforderlichen Bibliotheken installiert und stehen Ihnen zur weiteren Benutzung zur Verfügung. Eine Umfassende Dokumentation aller in der Bibliothek verfügbaren Funktionen findest du unter: `https://docs.dezibot.de/`.

5.2.3 Upload

Um den Dezibot$_4$ mit Ihrem Programm zu flashen, führen Sie nach der Auswahl der Registerkarte `Werkzeuge` bitte noch folgende Schritte aus:

1. Wählen Sie für den `Upload Mode "UART0/Hardware CDC"` aus
2. Wählen Sie für den `USB Mode "Hardware CDC and JTAG"` aus
3. Anschließend wählen Sie den `Port`, an dem der Dezibot$_4$ an Ihrem PC angeschlossen ist

Sollte der Port nicht angezeigt werden, kann es hilfreich sein, den Dezibot$_4$ aus- und wieder anzuschalten.

6 Eine kleine Einführung in Programmierprinzipien

Für ein besseres Verständnis während des Programmierens werden nun kurz einige grundlegende Prinzipien vorgestellt. Sollten Sie bereits über Programmierkenntnisse verfügen, können Sie dieses Kapitel gerne überspringen.
Einen umfassenderen Einblick in die Prinzipien der Arduino-Sprache finden Sie hier:

`Sprach-Referenz der Arduino-Programmiersprache` [34]

6.1 Grundlegende Syntax

Je nach Programmiersprache gelten unterschiedliche syntaktische Regeln, die beachtet werden müssen. Grundsätzlich ähnelt die Arduino-Programmiersprache der von C++ [35].
Einige wesentliche Syntaxregeln, die es zu beachten gilt, sind:

- Anweisungen enden mit einem Semikolon.
- Blöcke werden durch geschweifte Klammern begrenzt.
- Variablen müssen vor ihrer Verwendung deklariert werden.
- Es wird zwischen Groß- und Kleinschreibung unterschieden.
- Leerzeichen, Tabs und Zeilenumbrüche werden ignoriert.
- Funktionen müssen vor ihrem Aufruf deklariert und definiert werden.
- Es gibt reservierte Schlüsselwörter (z.B. `int, for, return` ...).
- Variablen und Funktionen sind nur innerhalb ihres Gültigkeitsbereichs ansprechbar.

Wichtig ist, dass das Einhalten dieser Regeln entscheidend für die korrekte Funktion und Ausführung von C++-Programmen ist. Weiterführend wird näher auf einige der gerade genannten Regeln eingegangen.

6.2 Variablen und Datentypen

Variablen [36] speichern Werte, die während der Ausführung eines Programms verändert werden können. Die Deklaration erfolgt durch die Angabe eines Datentyps [37], gefolgt von einem Bezeichner, der den Namen der Variablen repräsentiert. Einige Datentypen der Arduino-Programmiersprache sind:

- `bool` für Wahrheitwerte mit `true = 1` und `false = 0`
- `char` für einzelne Zeichenwerte als 8-Bit-Wert (z.B. `char buchstabe = 'A'`)
- `const` für Konstanten (z.B. `const double pi = 3.14`)
- `double` für Gleitkommazahlen als 64-Bit-Wert (z.B. `double a = 4.20`)
- `int` für Ganzzahlen als 16-Bit-Wert
- `string` für Zeichenketten (z.B. `string text = 'Beispieltext'`)
- `unsigned` für vorzeichenlose Variablen (z.B. `unsigned int`)

Es wird darauf hingewiesen, Datentypen so zu verwenden, dass sie eine optimale Speicherauslastung garantieren.

6.3 Funktionen

Funktionen [38] sind eigenständige Codeblöcke, die eine bestimmte Aufgabe ausführen. Sie besitzen einen Namen und können mehrere Übergabeparameter sowie einen Rückgabewert haben.

Beispielhaft wäre die Syntax der Funktionsdeklaration und -definition für die Addition zweier Zahlen `a` und `b` wie folgt:

```
1  int addition(int a, int b) {
2    int ergebnis = a + b;
3    return ergebnis;
4  }
```

Dabei stellt das `int` anfangs in Zeile 1 den Rückgabewert der Funktion dar. Der Funktionsname lautet `addition`. Die Übergabeparameter werden in den runden Klammern festgelegt. In diesem Beispiel handelt es sich um zwei Integerwerte mit dem Namen `a` und `b`.

Der Funktionskörper, der beim Aufruf der Funktion ausgeführt wird, ist durch die geschweiften Klammern begrenzt. Das `return`-Schlüsselwort in Zeile 3 bestimmt, welcher Wert von der Funktion zurückgegeben wird. Dieser muss den gleichen Datentyp besitzen wie der festgelegte Rückgabewert in der Funktionsdeklaration.

Funktionen werden aufgerufen, indem man ihren Namen nennt und die Argumente übergibt, mit denen sie ausgeführt werden soll. Das Funktionsergebnis wird einer Variable zugewiesen, die denselben Datentyp entspricht wie der `return`-Wert dieser Funktion.

Ein Beispiel für den Aufruf der oben vorgestellten Funktion `addition(int a, int b)` wäre:

```
1  void loop() {
2    int ergebnis = addition(4,20);
3  }
```

Die Variable `ergebnis` enthält nach Funktionsaufruf den Wert 24.

Die `void()`-Funktion stellt eine Besonderheit dar, da sie explizit keinen Rückgabewert definiert. Das Schlüsselwort wird ausschließlich in Funktionsdeklarationen verwendet, wie im folgenden Beispiel zu sehen ist:

```
1  void funktion() {
2    // Funktionscode
3  }
```

6.3.1 Bibliotheksfunktionen

Bibliotheksfunktionen [39] sind vordefinierte Funktionen, die in einer externen Bibliothek enthalten sind und zusätzliche Werkzeuge für ein Programm bereitstellen.

Durch Hinzufügen von `#include <Dezibot.h>` kann die Dezibot$_4$-Bibliothek in Ihrem Programm eingebunden werden (siehe Zeile 1 in Listing 7.1). Auf diese Weise stehen Ihnen die Funktionen der Bibliothek zur Verfügung. Eine detaillierte Erläuterung der weiteren Möglichkeiten der Dezibot$_4$-Bibliothek erfolgt in Kapitel 7.

Darüber hinaus ist beispielsweise `delay()` eine Bibliotheksfunktion, die in der Arduino-Plattform ohne zusätzliche Einbindung zur Verfügung steht. Sie dient dazu, eine Pause oder Verzögerung in der Ausführung des Programms zu erzeugen und erwartet Millisekunden als Übergabeparameter. Eine Verzögerung von einer Sekunde könnte dabei mit `delay(1000)` realisiert werden.

6.4 Kontrollstrukturen

Die grundlegenden Kontrollstrukturen [40] in der Programmierung sind `if, else, for` und `while`. Mit ihnen kann man Bedingungen überprüfen und Schleifen erstellen.

6.4.1 `if, else`-Bedingung

Das `if`-Statement checkt, ob eine Bedingung wahr ist und führt dann den Code aus, welcher innerhalb der geschweiften Klammern steht. Anderenfalls wird der Code übersprungen und die Ausführung geht zum nächsten Codeabschnitt nach dem `if`-Block über.

```
1   if(bedingung) {
2     // Anweisung
3   }
```

Optional kann eine Default-`else`-Anweisung verwendet werden, die Code ausführt, wenn alle vorherigen Tests auf `false` evaluiert wurden. Sollten mehr als zwei spezifische Bedingungen geprüft werden, können diese durch `else if`-Anweisungen erfolgen.

```
1   if(bedingung1) {
2     // Erste Anweisung
3   }
4   else if(bedingung2) {
5     // Zweite Anweisung
6   }
7   else {
8     // Andere Anweisung
9   }
```

6.4.2 `for`-Schleife

Eine `for`-Schleife ist eine Struktur, die verwendet wird, um eine Gruppe von Anweisungen wiederholt auszuführen.

```
1   for(int i = 0; i < iterationen; i++) {
2     // Anweisung
3   }
```

Die Anweisung hinter dem `for`-Schlüsselwort gibt die Anzahl an Wiederholungen vor und besteht aus drei Teilen. Dabei stellt `i` die Zählvariable dar, welche in diesem Beispiel als Integer definiert wird und bei Null beginnt. Durch `i++` wird diese in jedem Durchlauf um Eins erhöht bzw. inkrementiert. Die Variable `iterationen` gibt die obere Grenze der Wiederholungen an. In der Regel wird an ihrer Stelle ein Zahlenwert eingesetzt. Der Code innerhalb der geschweiften Klammern wird solange durchlaufen, wie die Bedingung `i < iterationen` wahr ist.

6.4.3 `while`-Schleife

Die `while`-Schleife ist eine andere Art von Schleifenstruktur, welche so lange ausgeführt wird, wie eine Bedingung wahr ist und endet, wenn diese als falsch ausgewertet.

```
1   while(bedingung) {
2     // Anweisung
3   }
```

6.5 Objektorientierte Programmierung

Die objektorientierte Programmierung [41] ist ein Programmierparadigma, das auf dem Konzept von Objekten basiert, die Daten und Methoden zur Manipulation dieser Daten zusammenfassen. Die wichtigsten Prinzipien der objektorientierten Programmierung sind Vererbung, Polymorphismus, Kapselung und Abstraktion. Durch diese Programmierkonzepte wird eine modularere, flexiblere und leichter wartbare Codebasis ermöglicht.

Das Verständnis des Konzepts der objektorientierten Programmierung ist für die Programmierung des Dezibot_4 nicht zwingend erforderlich. Da das Konstrukt bereits durch die Dezibot_4-Bibliothek bereitgestellt wird, ist es lediglich wichtig zu wissen, wie Sie auf die Objekte und Methoden zugreifen können.

6.5.1 Klassen und Objekte

Ein Objekt [42] ist eine Instanz einer Klasse und wird durch Aufrufen des Standardkonstruktors erstellt (sog. Instanziieren). Eine Klasse [43] hingegen ist eine abstrakter Datentyp, der definiert, welche Attribute bzw. Daten und Methoden ein Objekt haben wird.

In der Dezibot_4-Bibliothek wurden mehrere Klassen implementiert, unter anderem `Dezibot`, `Motion` und `MultiColorLight`. Dabei stellt `Dezibot` die übergeordnete Klasse dar, welche die Klassen der einzelnen Baugruppen hält.

Die Instanziierung des `Dezibot`-Objekts und der dazugehörigen Bauteile wird Ihnen im nächsten Kapitel nähergebracht.

6.5.2 Methoden

Methoden [44] sind Funktionen, die einer bestimmten Klasse zugeordnet sind und auf Objekten dieser Klasse aufgerufen werden können. Sie können auf die Attribute und Funktionen des Objekts zugreifen und diese verändern. Methoden können sowohl Parameter als auch Rückgabewerte haben.

Um Zugriff auf die Methoden der jeweiligen Klassen der Dezibot_4-Bibliothek zu erhalten, muss zuerst auf die `Dezibot`-Instanz zugegriffen werden. Anschließend können die untergeordneten Baugruppen-Objekte durch den Punktoperator angesprochen werden. Durch ein weiteres Anhängen des Punktoperators können die Methoden der untergeordneten Instanzen verwendet werden.

Ein Beispiel könnte dabei wie folgt aussehen:

```
1   void loop() {
2     // Roboter bewegt sich fuer eine Sekunde geradeaus
3     dezibot.motion.move();
4     delay(1000);
5     dezibot.motion.stop();
6   }
```

Hier wird über die zuvor erstellte `dezibot`-Instanz auf das Bewegungsobjekt `motion` zugegriffen und anschließend die Methode `move()` auf diesem Objekt aufgerufen. Einen tieferen Einblick dazu erhalten Sie ebenfalls im nächsten Kapitel.

7 Beispielprogramme

Die Dezibot$_4$-Bibliothek verfügt über viele hilfreiche Funktionen, um den Roboter in Bezug auf seine Bewegung und die Steuerung der RGB-LEDs effektiv anzusprechen.

Um mit der Programmierung beginnen zu können, sollte der Dezibot$_4$ über ein USB-C-Kabel mit Ihrem Computer verbunden und eingeschaltet sein. Stellen Sie außerdem sicher, dass der entsprechende Port, an dem der Roboter angeschlossen ist, ausgewählt wird (siehe Schritt 3 in Kapitel 5.2.3). Sollte der Port nicht angezeigt werden, kann es hilfreich sein, den Dezibot$_4$ kurzzeitig aus- und wieder einzuschalten.

7.1 Zu Beginn

Um die Funktionen der Dezibot$_4$-Bibliothek nutzen zu können, müssen Sie diese zu Beginn in Ihr Programm einbinden. Dies geschieht in Zeile 1.

```
1   #include <Dezibot.h>
2
3   Dezibot dezibot = Dezibot();
4
5   void setup() {
6     dezibot.begin();
7   }
8
9   void loop() {
10    //Fuegen Sie hier eigenen Code ein
11  }
```

Listing 2: Einbindung der Bibliothek, Initialisierung des Dezibot$_4$-Objekts und Funktionen

Anschließend wird eine Instanz der Klasse `Dezibot` mit dem Namen `dezibot` erstellt, ein sogenanntes Objekt. Dies geschieht durch eine Variablendeklaration vom Typ `Dezibot` und der sofortigen Initialisierung dieser Variablen mit dem Aufruf des Standardkonstruktors `Dezibot()`, wie in Zeile 3 zu sehen ist.

Die `setup()`-Funktion in Zeile 5 bis 7 wird immer aufgerufen, wenn der Sketch startet. Sie wird benutzt, um Variablen, Pinmodi, Bibliotheken usw. zu initialisieren. Diese Funktion wird nur einmal ausgeführt – jedes Mal, wenn das Board gestartet oder zurückgesetzt wird.

In Zeile 6 werden die grundlegenden Einstellungen für den Dezibot$_4$ festgelegt. Durch `dezibot.begin()` wird es möglich, die Instanzen und Methoden der verschiedenen Baugruppen zu nutzen, z. B. die `MultiColorLight`-Klasse für die RGB-LEDs oder die `move()`-Methode für die Motoren.

Die Funktion `loop()` in Zeile 9 bis 11 ist eine Endlosschleife, die nach jedem Durchlauf erneut aufgerufen wird. Dadurch kann das Programm Variablen verändern, Daten lesen oder darauf reagieren. Diese Funktion wird verwendet, um das Arduino-Board aktiv zu steuern.

Nachdem Sie die Zeilen 1 bis 11 in Ihrem Programm eingefügt haben, können Sie die vielfältigen Funktionen der Dezibot$_4$-Bibliothek auf das zuvor erstellte `dezibot`-Objekt nutzen und Ihre eigenen Programme auf den Roboter übertragen. Um möglichst schnell mit diesem Ausgangsprogramm starten zu können, gehen Sie in Arduino auf die Registerkarte `Datei`, wählen sie `Beispiele` und anschließend `Dezibot` aus. Drücken Sie dann auf `start` und das Startprogramm wird automatisch geladen. Unter den Beispielen gibt es neben `start` noch weitere Programmbeispiele, die die Verwendung der einzelnen Komponenten des Dezibots veranschaulichen.

Um Ihnen einen Einstieg in die Methoden der Bibliothek zu ermöglichen, finden Sie in den folgenden Kapiteln einige Beispielprogramme. Bitte beachten Sie, dass die Zeilen 1 bis 8 aus Gründen der Übersichtlichkeit in den Beispielprogrammen ausgelassen wurden. Es ist jedoch wichtig, dass diese Zeilen immer vor der `loop()`-Funktion stehen.

7.2 Beispielprogramm 1

Wie im Optik-Kapitel schon vorgestellt wurde, befinden sich auf dem Dezibot$_4$ drei RGB-LEDs (siehe Nr. 1 in Abb. 1 und Abb. 2). Zwei dieser LEDs befinden sich auf der Oberseite und eine auf der Unterseite des Roboters. Sie lassen sich über die `MultiColorLight`-Klasse ansprechen, welche die Methoden `setLed(), setTopLeds(), blink()` und `turnOffLed()` zur Verfügung stellt:

- `setLed()` setzt die angegebenen LEDs auf den übergebenen Farbwert.
- `setTopLeds()` weist den zwei oberen LEDs eine Farbe zu.
- `blink()` lässt die angegebene LED in einer bestimmten Farbe für eine vorgegebene Zeitdauer in der gewünschten Anzahl an Wiederholungen blinken. Hierbei ist zu beachten, dass die Farbe als Hexadezimalwert angegeben werden muss. Das Format lautet `0x00RRGGBB`, wobei `R` für Rot, `G` für Grün und `B` für Blau steht. Der Wertebereich jeder Farbe beträgt 0 bis 100, dies entspricht in hexadezimal `0x00` bis `0x64`. So kann zum Beispiel für Rot der Wert `0x00640000` angegeben werden. Wird kein Parameter an die Methode übergeben, beträgt der Defaultwert für die Farbe `GREEN`, für die LEDs `TOP` und das Zeitintervall ist eine Sekunde lang. Ein Beispiel für den Aufruf dieser Methode können sie im Beispielprogramm 2 finden.
- `turnOffLed()` schaltet die angegebenen LEDs wieder aus. Wenn keine spezifische LED angegeben wird, werden alle ausgeschalten.

Die LEDs können individuell oder zusammen mit `TOP_LEFT, TOP_RIGHT, BOTTOM, TOP` und `ALL` angesprochen werden. Die Übergabeparameter der Methoden bestehen dann aus der oder den angesprochenen LED/s und dem gewünschten Farbwert.

```
1   void loop() {
2     dezibot.multiColorLight.setLed(BOTTOM, RED);
3     dezibot.multiColorLight.setLed(TOP, RED);
4     delay(2000);
5
6     dezibot.multiColorLight.setLed(BOTTOM, GREEN);
7     dezibot.multiColorLight.setLed(TOP, GREEN);
8     delay(2000);
9
10    dezibot.multiColorLight.setLed(BOTTOM, BLUE);
11    dezibot.multiColorLight.setLed(TOP, BLUE);
12    delay(2000);
13
14    dezibot.multiColorLight.turnOffLed();
15    delay(2000);
16  }
```

Listing 3: Beispielprogramm 1

Beispielprogramm 1 stellt nacheinander für alle LEDs die Farben Rot, Grün und Blau ein. Zuerst wird die Farbe Rot (`RED`) in Zeile 2 für die untere und dann in Zeile 3 für beide oberen RGB-LEDs eingestellt. Der `delay(2000)` in Zeile 4 stellt sicher, dass die LEDs für 2000 ms, also zwei Sekunden, leuchten. Dies wird für die Farben Grün (`GREEN`) und Blau (`BLUE`) in den Zeilen 6 bis 12 wiederholt.

In Zeile 14 und 15 werden die LEDs für zwei Sekunden ausgeschaltet.

Durch die `loop()`-Funktion beginnt das Programm danach wieder von vorne.

7.3 Beispielprogramm 2

Da Sie nun eine grundlegende Einführung in die Methoden der RGB-LEDs erhalten haben, wird im Folgenden ein LED-Programm mit Kontrollstrukturen vorgestellt.

```
1  void loop() {
2   for(int i = 0; i < 100; i++){
3    if(i % 4 == 0){
4     dezibot.multiColorLight.setLed(TOP_LEFT, GREEN);
5     dezibot.multiColorLight.setLed(TOP_RIGHT, BLUE);
6     dezibot.multiColorLight.setLed(BOTTOM, RED);
7     // Gruen
8     dezibot.multiColorLight.blink(2, 0x00006400, TOP_LEFT, 200);
9     dezibot.multiColorLight.setLed(TOP_LEFT, GREEN);
10    // Blau
11    dezibot.multiColorLight.blink(2, 0x00000064, TOP_RIGHT, 200);
12    dezibot.multiColorLight.setLed(TOP_RIGHT, BLUE);
13    // Rot
14    dezibot.multiColorLight.blink(2, 0x00640000, BOTTOM, 200);
15   }
16   else if(i % 4 == 1){
17    dezibot.multiColorLight.setLed(TOP_LEFT, RED);
18    dezibot.multiColorLight.setLed(TOP_RIGHT, GREEN);
19    dezibot.multiColorLight.setLed(BOTTOM, BLUE);
20    // Rot
21    dezibot.multiColorLight.blink(2, 0x00640000, TOP_LEFT, 200);
22    dezibot.multiColorLight.setLed(TOP_LEFT, RED);
23    // Gruen
24    dezibot.multiColorLight.blink(2, 0x00006400, TOP_RIGHT, 200);
25    dezibot.multiColorLight.setLed(TOP_RIGHT, GREEN);
26    // Blau
27    dezibot.multiColorLight.blink(2, 0x00000064, BOTTOM, 200);
28   }
29   else if(i % 4 == 2){
30    dezibot.multiColorLight.setLed(TOP_LEFT, BLUE);
31    dezibot.multiColorLight.setLed(TOP_RIGHT, RED);
32    dezibot.multiColorLight.setLed(BOTTOM, GREEN);
33    // Blau
34    dezibot.multiColorLight.blink(2, 0x00000064, TOP_LEFT, 200);
35    dezibot.multiColorLight.setLed(TOP_LEFT, BLUE);
36    // Rot
37    dezibot.multiColorLight.blink(2, 0x00640000, TOP_RIGHT, 200);
38    dezibot.multiColorLight.setLed(TOP_RIGHT, RED);
39    // Gruen
40    dezibot.multiColorLight.blink(2, 0x00006400, BOTTOM, 200);
41   }
42   else {
43    dezibot.multiColorLight.blink(2);
44   }
45  }
46 }
```

Listing 4: Beispielprogramm 2

Beispielprogramm 2 lässt die Farben der LEDs nacheinander rotieren und blinken. Innerhalb der `loop()`-Funktion wird in den Zeilen 2 bis 45 hundertmal eine `for`-Schleife durchlaufen.

In Zeile 3 wird durch eine `if`-Bedingung geprüft, ob die Zählvariable `i` geteilt durch vier den Rest null ergibt. Das Prozentzeichen stellt dabei die Modulo-Operation dar, welche einer Division mit Rest entspricht. Trifft dies zu, wird der Codeblock aus den Zeilen 4 bis 15 ausgeführt. In den Zeilen 4 bis 6 wird zuerst der linken oberen LED die Farbe grün, dann der rechten oberen LED die Farbe blau und anschließend der unteren LED die Farbe rot zugewiesen.

Daraufhin wird in Zeile 7 die `blink()`-Methode für die obere linke LED aufgerufen. Der erste Übergabeparameter, hier die `2`, gibt an, wie oft die LED blinken soll. Danach wird die Farbe als Hexadezimalwert übergeben. Die obere linke LED soll grün leuchten. Zuletzt muss die gewünschte LED und das Zeitintervall, also wie lange die LED leuchtet, angegeben werden.

In Zeile 9 wird der oberen linken LED erneut die Farbe Grün zugewiesen, da die LED nach Abschluss der `blink()`-Methode ausgeschalten wird.

Dies wird in den Zeilen 10 bis 14 für die obere, rechte und untere LED wiederholt. Da danach eine neue Iteration der `for`-Schleife beginnt, muss die untere LED nicht erneut explizit angeschalten werden.

Wenn die Bedingung aus Zeile 3 nicht zutrifft, wird zuerst Zeile 16 und anschließend Zeile 29 geprüft. In den Zeilen 17 bis 28 sowie 30 bis 41 werden den LEDs lediglich andere Farbwerte in einer unterschiedlichen Reihenfolge zugewiesen. Das Prinzip entspricht den soeben vorgestellten Zeilen 3 bis 15.

Trifft keine der drei geprüften Bedingungen zu, wird der Code in Zeile 43 ausgeführt. Hier wird `blink()` mit Defaultwerten aufgerufen. Aus diesem Grund muss lediglich die Anzahl der Wiederholungen an die Methode übergeben werden. Welche Werte standardmäßig übergeben werden, wurde bereits in Kapitel 7.2 beschrieben.

Durch die `loop()`-Funktion beginnt das Programm danach wieder von vorne.

7.4 Beispielprogramm 3

Die Unwuchtmotoren des Dezibot$_4$ (siehe Nr. 11 in Abb. 2), welche bereits im Kinematik-Kapitel beschrieben wurden, lassen sich mit dem `motion`-Objekt ansprechen. Dieses Objekt enthält die Methoden `move()`, `rotateClockwise()` und `rotateAnticlockwise()`, welche den Roboter geradeaus, nach rechts oder links bewegen. Der Übergabeparameter dieser Methoden wird in Millisekunden angegeben.

```
1   void loop() {
2     for(int i = 0; i < 15; i++) {
3       if(i < 5) {
4         dezibot.motion.move();
5         delay(1000);
6       }
7       else if(i >= 5 && i < 10) {
8         dezibot.motion.rotateClockwise();
9         delay(1000);
10      }
11      else {
12        dezibot.motion.rotateAnticlockwise();
13        delay(1000);
14      }
15    }
16  }
```

Listing 5: Beispielprogramm 3

Beispielprogramm 3 verwendet eine `for`-Schleife in den Zeilen 2 bis 15, um den Dezibot$_4$ zuerst geradeaus, dann nach rechts und schließlich nach links zu bewegen. Innerhalb der Schleife prüfen drei `if`- bzw. `else if`-Bedingungen, ob die Zählvariable `i` größer oder kleiner einem vorgegebenen Wert ist.

In Zeile 3 wird überprüft, ob die Zählvariable kleiner als fünf ist. Trifft dies zu, soll der Roboter geradeaus gehen, wie es in Zeile 4 beschrieben wird.

Liegt `i` zwischen fünf und zehn, bewegt sich der Dezibot$_4$ nach rechts. Dies kann den Zeilen 7 bis 10 entnommen werden.

Gemäß den Zeilen 11 bis 14 soll sich der Roboter in allen anderen Fällen nach links bewegen.

Nach jeder Bewegung wurde ein `delay(1000)` von einer Sekunde eingefügt, um so zu gewährleisten, dass die Bewegung jeweils eine Sekunde lang durchgeführt wird.

Durch die `loop()`-Funktion beginnt das Programm danach wieder von vorne.

8 Außerbetriebnahme und Entsorgung

Zur Außerbetriebnahme ist der Akkumulator zu entfernen und zum Beispiel mit einem Klebeband aus Kunststoff oder einer Plastiktüte zu isolieren.

Danach ist der Akku entsprechend den lokalen Vorschriften zur Batterierücknahme in Recyclinghöfen, im Einzelhandel oder bei spezialisierten Firmen zu entsorgen.

Der Dezibot$_4$ ohne Akkumulator sollte über die Elektronikschrottrücknahme für Kleingeräte (z.B. im Einzelhandel) oder bei einer Recyclingfirma fachgerecht entsorgt werden.

Sowohl der Dezibot$_4$ als auch der Akkumulator gehören nicht in den Hausmüll und dürfen keinesfalls verbrannt werden oder in die Umwelt gelangen. Bitte entsorge sie ordnungsgemäß entsprechend den Vorschriften.

Literatur

[1] Inc. Free Software Foundation. *GNU General Public License*. 2017. URL: http://www.gnu.de/documents/gpl.de.html.

[2] Wikipedia – Die freie Enzyklopädie. *Mikrocontroller*. Feb. 2024. URL: https://de.wikipedia.org/wiki/Mikrocontroller.

[3] Espressif Systems. *ESP32-S3-MINI-1 Datenblatt*. März 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3-mini-1_mini-1u_datasheet_en.pdf.

[4] Wikipedia – Die freie Enzyklopädie. *GPIO*. Mai 2023. URL: https://de.wikipedia.org/wiki/GPIO.

[5] Wikipedia – Die freie Enzyklopädie. *Universal Asynchronous Receiver Transmitter*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Universal_Asynchronous_Receiver_Transmitter.

[6] Wikipedia – Die freie Enzyklopädie. *I²C*. Sep. 2023. URL: https://de.wikipedia.org/wiki/I%C2%B2C.

[7] Wikipedia – Die freie Enzyklopädie. *Serial Peripheral Interface*. März 2023. URL: https://de.wikipedia.org/wiki/Serial_Peripheral_Interface.

[8] Wikipedia – Die freie Enzyklopädie. *Leuchtdiode*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Leuchtdiode.

[9] Limited Worldsemi Co. *WS2812B-Mini-V2 Datenblatt*. Aug. 2018. URL: https://www.led-stuebchen.de/download/WS2812B-Mini-V2_V1.0_EN.pdf.

[10] Wikipedia – Die freie Enzyklopädie. *Pulsdauermodulation*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Pulsdauermodulation.

[11] Wikipedia – Die freie Enzyklopädie. *Farbsensor*. Feb. 2023. URL: https://de.wikipedia.org/wiki/Farbsensor.

[12] Inc. Vishay Intertechnology. *VEML6040 Datenblatt*. Jan. 2023. URL: https://www.vishay.com/docs/84276/veml6040.pdf.

[13] Wikipedia – Die freie Enzyklopädie. *Complementary metal-oxide-semiconductor*. Dez. 2022. URL: https://de.wikipedia.org/wiki/Complementary_metal-oxide-semiconductor.

[14] Wikipedia – Die freie Enzyklopädie. *Infrarot-LED*. Feb. 2023. URL: https://de.wikipedia.org/wiki/Infrarot-LED.

[15] Inc. Vishay Intertechnology. *VSMY3850X01 Datenblatt*. Nov. 2019. URL: https://www.vishay.com/doc/?80225=.

[16] Harvatek Corporation. *B2141IR–A1C000373U1930 Datenblatt*. Feb. 2020. URL: https://www.harvatek.com/files/B2141IR.pdf.

[17] Wikipedia – Die freie Enzyklopädie. *Photodiode*. Sep. 2023. URL: https://de.wikipedia.org/wiki/Photodiode.

[18] Wikipedia – Die freie Enzyklopädie. *Transistor*. Dez. 2023. URL: https://de.wikipedia.org/wiki/Transistor.

[19] Ltd. Everlight Electronics Co. *PT26-51B Datenblatt*. Juni 2007. URL: https://www.mouser.de/datasheet/2/143/PT2651B-1166171.pdf.

[20] OSRAM Opto Semiconductors GmbH. *SFH320 Datenblatt*. Mai 2007. URL: https://cdn-reichelt.de/documents/datenblatt/A501/SFH320_Pb_free.pdf.

[21] OSRAM Opto Semiconductors GmbH. *SFH325 Datenblatt*. Nov. 2021. URL: https://dammedia.osram.info/media/resource/hires/osram-dam-5891706/SFH%20325_EN.pdf.

[22] Wikipedia – Die freie Enzyklopädie. *Organische Leuchtdiode*. Dez. 2023. URL: https://de.wikipedia.org/wiki/Organische_Leuchtdiode.

[23] DigiKey. *OLED 128x64 0.96" I2C Datenblatt*. URL: https://universal-solder.ca/downloads/canaduino_oled_128x64_i2c.pdf.

[24] Wikipedia – Die freie Enzyklopädie. *Unwuchtmotor*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Unwuchtmotor.

[25] Elektrotechnik Karl-Heinz Mauz GmbH. *VM-2702A2.7-SMD Datenblatt*. Okt. 2021. URL: https://www.reichelt.de/index.html?ACTION=7&LA=3&OPEN=0&INDEX=0&FILENAME=C900%2F860271VM-2702A2.7-SMD%285%29.pdf.

[26] Wikipedia – Die freie Enzyklopädie. *Inertiale Messeinheit*. Dez. 2023. URL: https://de.wikipedia.org/wiki/Inertiale_Messeinheit.

[27] InvenSense. *ICM-42670-P Datenblatt*. Apr. 2021. URL: https://product.tdk.com/system/files/dam/doc/product/sensor/mortion-inertial/imu/data_sheet/ds-000451-icm-42670-p.pdf.

[28] Wikipedia – Die freie Enzyklopädie. *Serielle Schnittstelle*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Serielle_Schnittstelle.

[29] Wikipedia – Die freie Enzyklopädie. *Bluetooth*. Dez. 2023. URL: https://de.wikipedia.org/wiki/Bluetooth.

[30] Wikipedia – Die freie Enzyklopädie. *Wireless Local Area Network*. Dez. 2023. URL: https://de.wikipedia.org/wiki/Wireless_Local_Area_Network.

[31] Arduino. *Herunterladen und Installieren der Arduino IDE 2*. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing/.

[32] Arduino. *Downloads*. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing/.

[33] GitHub. *FUSE*. URL: https://github.com/AppImage/AppImageKit/wiki/FUSE.

[34] Arduino. *Sprach-Referenz*. URL: https://www.arduino.cc/reference/de/.

[35] Wikipedia – Die freie Enzyklopädie. *C++*. März 2024. URL: https://de.wikipedia.org/wiki/C%2B%2B#:~:text=C%2B%2B%20ist%20eine%20von,eine%20Programmierung%20auf%20hohem%20Abstraktionsniveau..

[36] Wikipedia – Die freie Enzyklopädie. *Variable (Programmierung)*. Jan. 2024. URL: https://de.wikipedia.org/wiki/Variable_(Programmierung).

[37] Wikipedia – Die freie Enzyklopädie. *Datentyp*. Jan. 2024. URL: https://de.wikipedia.org/wiki/Datentyp.

[38] Wikipedia – Die freie Enzyklopädie. *Funktion (Programmierung)*. Okt. 2022. URL: https://de.wikipedia.org/wiki/Funktion_(Programmierung).

[39] Wikipedia – Die freie Enzyklopädie. *Programmbibliothek*. Feb. 2024. URL: https://de.wikipedia.org/wiki/Programmbibliothek.

[40] Wikipedia – Die freie Enzyklopädie. *Kontrollstruktur*. Aug. 2023. URL: https://de.wikipedia.org/wiki/Kontrollstruktur.

[41] Wikipedia – Die freie Enzyklopädie. *Objektorientierte Programmierung*. Okt. 2022. URL: https://de.wikipedia.org/wiki/Objektorientierte_Programmierung.

[42] Wikipedia – Die freie Enzyklopädie. *Objekt (Programmierung)*. Okt. 2023. URL: https://de.wikipedia.org/wiki/Objekt_(Programmierung).

[43] Wikipedia – Die freie Enzyklopädie. *Klasse (Objektorientierung)*. Feb. 2024. URL: https://de.wikipedia.org/wiki/Klasse_(Objektorientierung).

[44] Wikipedia – Die freie Enzyklopädie. *Methode (Programmierung)*. März 2024. URL: https://de.wikipedia.org/wiki/Methode_(Programmierung).