

**Big Data Analytics - Summer School 2025**

# **In The Woods**

**Eingereicht von** Thi Trung Anh Nguyen  
Mauriz Weymann  
**Studiengang** Wirtschaftsinformatik

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>ii</b>
<b>1 Projektbeschreibung</b>	<b>1</b>
<b>2 Maschine Learning: Datensatz, Training und Deployment (Edge Impulse)</b>	<b>2</b>
<b>3 Einrichtung und Implementierung (Arduino-IDE)</b>	<b>8</b>
3.1 Konfiguration der Entwicklungsumgebung . . . . .	8
3.2 Struktur und Funktionsweise des Steuerungsprogramms . . . . .	9
<b>4 Prototypenaufbau</b>	<b>11</b>
4.1 Beschreibung des Prototyps . . . . .	11
4.2 Elektronisches Signal- und Steuersystem . . . . .	12
4.3 Testphase und Beobachtung . . . . .	13
4.4 Gründe für das Fehlverhalten . . . . .	14
4.5 Neuer, funktionsfähiger Aufbau . . . . .	15
4.6 Test des neuen Aufbaus und Simulationsergebnisse . . . . .	16
<b>5 Fazit und Ausblick</b>	<b>18</b>

# Abbildungsverzeichnis

2.1	Edge-Impulse-Datensatz mit »Baum«-Annotationen . . . . .	3
2.2	Impulse-Konfiguration: Bildverarbeitung mit Objekterkennung . . . .	4
2.3	Trainingskonfiguration . . . . .	5
2.4	Leistungsübersicht der Objekterkennung . . . . .	6
2.5	Deployment-Konfiguration für Arduino . . . . .	7
3.1	Boards Manager und Portauswahl in der Arduino IDE . . . . .	8
3.2	PSRAM einstellen und ZIP-Bibliothek einbinden . . . . .	9
3.3	Serieller Monitor: Inferenz-Ausgaben und Richtungsentscheidung . . .	10
4.1	Prototypaufbau des Projekts „ <i>In The Woods</i> “ . . . . .	12
4.2	Alter, nicht funktionsfähiger Aufbau . . . . .	13
4.3	Funktionsfähiger Aufbau des Systems . . . . .	16

# 1 Projektbeschreibung

Das Projekt *In The Woods* wurde im Rahmen der Summer School 2025 im Bereich Big Data Analysis von Masterstudierenden der HTW Berlin in Kooperation mit dem CityLAB Berlin entwickelt.

Die Motivation und Inspiration resultierten aus verschiedenen Workshops, Anforderungen und Anregungen, die im Rahmen der Summer School präsentiert wurden. Das übergeordnete Thema des Jahres 2025 lautete „Der Wald“. Als Herausforderungen und Projektaufgaben wurden verschiedene Challenges angeboten; unsere Entscheidung fiel auf die Entwicklung eines fliegenden Objekts, das ausdrücklich keine Drohne darstellen sollte. Der Titel dieser Challenge lautete „NOT-A-DRONE“. In weiteren Workshops wurden Themen des Maschinellen Lernens (ML) und der Edge Devices bzw. Microcontroller behandelt. Darüber hinaus wurden von Technik inspirierte Kunstprojekte vorgestellt und Einblicke in Themen wie Prototyping, Datenvisualisierung und Forschungsinitiativen gegeben.

Aus den Themengebieten Wald, Machine Learning (ML), Microcontroller und Elektronik entstand die Idee, „*das Leben im Wald mithilfe von ML und einem autarken Flugobjekt zu erforschen*“, bzw. mit minimalen Ressourcen eine Form der Naturbeobachtung oder des Umweltmonitorings zu realisieren. Als Einstiegspunkt wurde die autonome Navigation mittels Echtzeit-Hinderniserkennung durch KI-gestützte Bilderkennung gewählt. Der Umsetzungsplan gliederte sich in mehrere Teilprojekte:

1. Erstellung eines trainierten Modells auf Basis geeigneter Trainingsdaten
2. Entwicklung einer lauffähigen Anwendung, welche das trainierte Modell ausführt und daraus Steuersignale für LEDs und Motoren ableitet
3. Integration der Hardwarekomponenten zu einem funktionalen Prototyp

## 2 Maschine Learning: Datensatz, Training und Deployment (Edge Impulse)

Als Grundlage für die bildbasierte Navigation dient ein mithilfe von Edge Impulse trainiertes Machine-Learning-Modell.

Edge Impulse ist eine Plattform (SaaS) für TinyML. Mithilfe von Edge Impulse lassen sich alle Schritte – von der Datenerfassung über das Training bis hin zum Deployment – in einer durchgängigen End-to-End-Umgebung umsetzen. Ein wesentlicher Vorteil der Plattform liegt in den integrierten AutoML-Werkzeugen für Klassifikation, Objekterkennung und Anomalieerkennung. Zusätzlich besteht die Möglichkeit, Modelle zu testen und bei Bedarf erneut zu trainieren. Nach dem Training lässt sich das optimierte Modell für eine gewünschte Zielplattform exportieren.

Für das Machine Learning werden gelabelte Trainingsdaten benötigt. Zu diesem Zweck wurde ein künstlicher Mischwald mit zwölf Nadel- und Laubbäumen aus grünem Karton erstellt. Der erstellte Datensatz besteht ausschließlich aus Bildern von Bäumen, die jeweils mit dem Label „Baum“ versehen wurden (vgl. Abbildung 2.1). Weitere Klassen wurden nicht definiert, da die Unterscheidung zwischen „links“ und „rechts“ nicht über unterschiedliche Labels erfolgt, sondern erst in einer späteren Auswertung anhand der Position des erkannten Objekts im Bild berechnet wird. Der Datensatz umfasst insgesamt 508 Bilder, die automatisch in Trainings- und Testdaten aufgeteilt wurden, um eine robuste Modellvalidierung sicherzustellen.

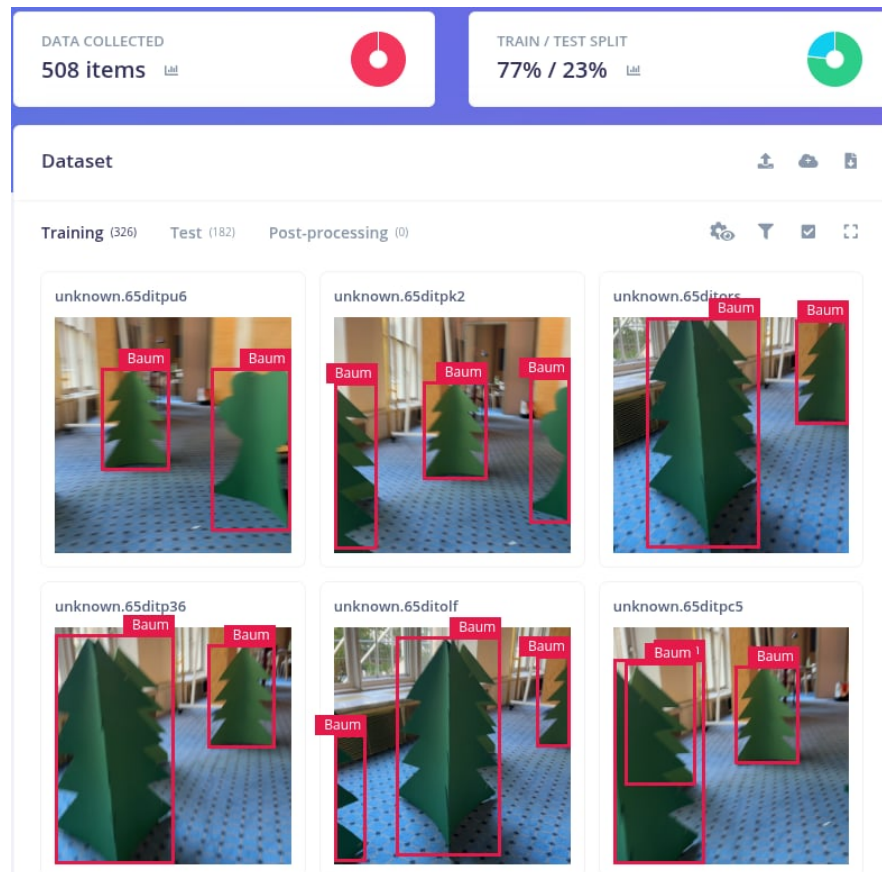


Abbildung 2.1: Edge-Impulse-Datensatz mit »Baum«-Annotationen

Im Edge Impulse Studio wurde ein sogenannter Impulse konfiguriert, bei dem die Eingangsdaten auf eine feste Bildgröße von  $96 \times 96$  Pixeln skaliert wurden. Diese Reduktion der Auflösung ist notwendig, da der verwendete XIAO ESP32S3 Sense nur über begrenzte Rechenleistung und Speicherkapazität verfügt. Durch die kleinere Bildgröße kann das Modell dennoch in Echtzeit auf dem Mikrocontroller ausgeführt werden, ohne die verfügbaren Ressourcen zu überlasten.

Anschließend wurden die Bilddaten über einen Image Processing Block vorverarbeitet und an einen Learning Block für Object Detection übergeben. Als Ausgabe war ausschließlich die Klasse „Baum“ definiert, sodass das Modell neben der Klassifikation auch die Bounding-Box-Koordinaten der erkannten Objekte bereitstellt (vgl. Abbildung 2.2).

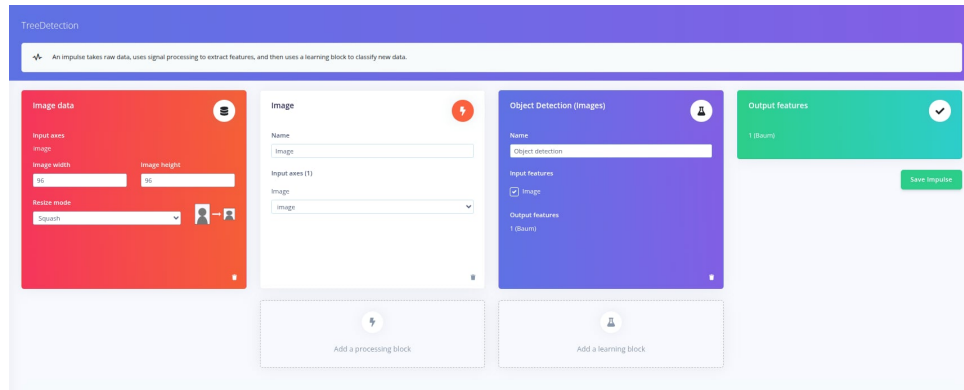


Abbildung 2.2: Impulse-Konfiguration: Bildverarbeitung mit Objekterkennung

Als Modellarchitektur kam FOMO (Faster Objects, More Objects) auf Basis von MobileNetV2 0.35 zum Einsatz (vgl. Abbildung 2.3). Diese Architektur ist speziell für ressourcenbeschränkte Geräte wie Microcontroller optimiert. Sie ermöglicht eine schnelle und speichereffiziente Inferenz und eignet sich daher ideal für den Einsatz auf dem XIAO ESP32S3 Sense.

Das Training des Modells erfolgte über 60 Epochen bei einer Lernrate von 0,1. Während des Trainings wurde der Datensatz automatisch in Trainings- und Validierungsdaten aufgeteilt, um eine objektive Bewertung der Modelleistung zu gewährleisten. Jede Epoche umfasste mehrere Trainingsdurchläufe, bei denen die Modellparameter iterativ angepasst wurden, um den Klassifikationsfehler zu minimieren.

**Neural Network settings**

**Training settings**

- Number of training cycles: 60
- Use learned optimizer: ☐
- Learning rate: 0.1
- Training processor: CPU
- Data augmentation: ☐

**Advanced training settings**

- Validation set size: 20 %
- Split train/validation set on metadata key:
- Batch size: 32
- Profile int8 model: ☒

**Neural network architecture**

Input layer (27,648 features)

FOMO (Faster Objects, More Objects) MobileNetV2 0.35

Choose a different model

Output layer (1 classes)

Save & train

Abbildung 2.3: Trainingskonfiguration

Das mit Edge Impulse auf Basis der FOMO-Architektur trainierte Erkennungsmodell erreicht im Validierungssatz einen F1-Score von 81,7 % bei sehr hoher Präzision (0,92) und gutem Recall (0,74). In der Konfusionsmatrix bleiben Hintergrundbeispiele vollständig als Hintergrund klassifiziert (100 %), während die Zielklasse BAUM in 73,7 % der Fälle korrekt erkannt wird, was eine zuverlässige Auslösung der Erkennung bei zugleich geringer Fehlalarmrate sicherstellt. Die Inferenz erfolgt vollständig on-device auf dem XIAO ESP32S3 in etwa 1013 ms pro Bild (quantisierte INT8-Ausführung mit EON-Compiler, RAM-optimiert) und benötigt lediglich rund 137,7 KB Spitzen-RAM sowie etwa 81,3 KB Flash (vgl. Abbildung 2.4).

Insgesamt verbindet das Modell robuste Erkennungsleistung mit einem effizienten Embedded-Footprint und stellt stabil die zur Steuerlogik benötigte Positionsinformation bereit.



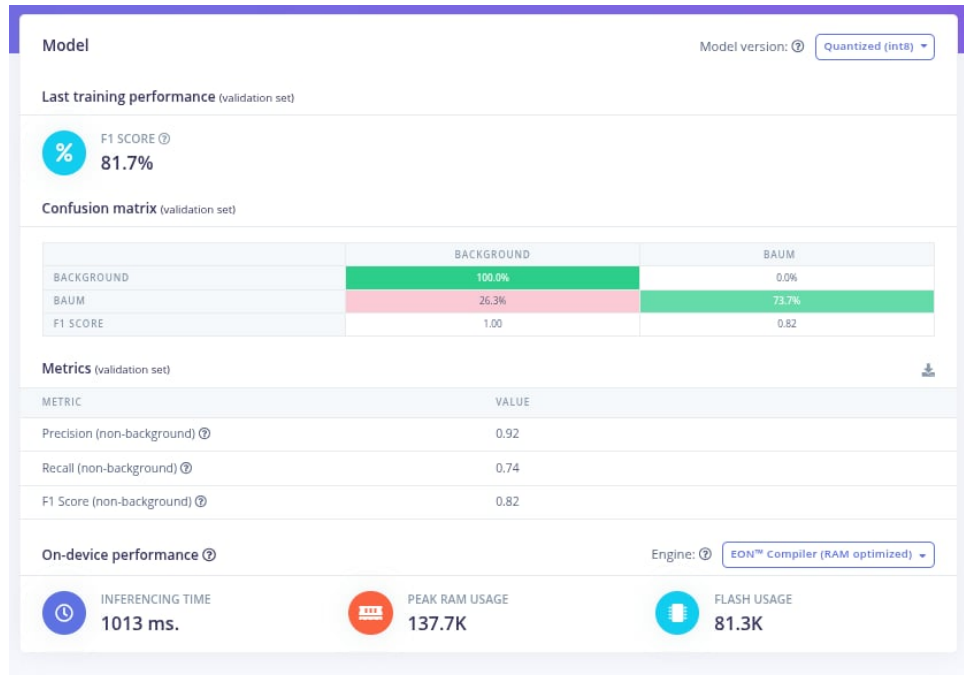


Abbildung 2.4: Leistungsübersicht der Objekterkennung


Nach Abschluss des Trainings wurde das Modell aus Edge Impulse als Arduino Library exportiert. Diese Bibliothek enthält alle notwendigen Funktionen für die Inferenz auf dem Mikrocontroller und kann direkt in der Arduino-IDE eingebunden werden. Im Projekt wurde sie in den Code integriert, sodass die Kamera des XIAO ESP32S3 Sense fortlaufend Bilder aufnimmt und diese unmittelbar an das Modell übergibt.

Die Verwendung der Arduino Library bietet den Vorteil, dass die gesamte Verarbeitung direkt auf dem Gerät erfolgt. Eine Internetverbindung oder externe Rechenressourcen sind nicht erforderlich, wodurch das System unabhängig und in Echtzeit betrieben werden kann (vgl. Abbildung 2.5).

### Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

Arduino library




SELECTED DEPLOYMENT

**Arduino library**

An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.



TensorFlow Lite

Quantized (int8)

Selected ✓

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	11 ms.	1,013 ms.	1,024 ms.
RAM	4.0K	283.1K	283.1K
FLASH	-	110.6K	-
ACCURACY			-

Unoptimized (float32)

Select

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	11 ms.	7,022 ms.	7,033 ms.
RAM	4.0K	1.0M	1.0M
FLASH	-	137.9K	-
ACCURACY			-

To compare model accuracy, run model testing for all available optimizations.

[Run model testing](#)

Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) - [Change target](#)

Build

Abbildung 2.5: Deployment-Konfiguration für Arduino

7

## 3 Einrichtung und Implementierung (Arduino-IDE)

### 3.1 Konfiguration der Entwicklungsumgebung

Um das trainierte Modell auf dem XIAO ESP32S3 Sense auszuführen, muss in der Arduino IDE das passende Board installiert werden. Dazu öffnet man den Boardverwalter über Werkzeuge → Board → Boardverwalter, sucht nach ESP32 und installiert das Paket ESP32 by Espressif Systems. Nach der Installation wird im Menü Werkzeuge → Board das Board Seeed Studio XIAO ESP32S3 Sense ausgewählt (vgl. Abbildung 3.1).

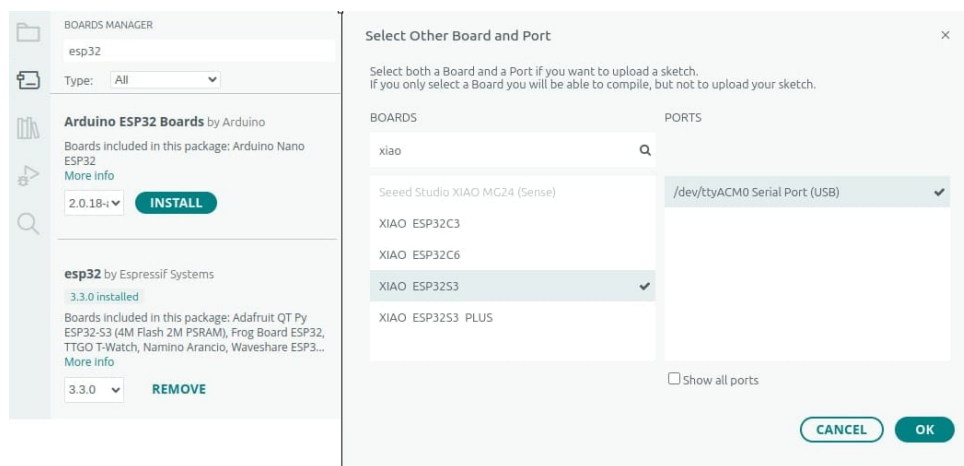


Abbildung 3.1: Boards Manager und Portauswahl in der Arduino IDE

Da die Bilderkennung eine größere Datenmenge verarbeitet, ist es erforderlich, den PSRAM (Pseudo Static RAM) des XIAO ESP32S3 Sense zu aktivieren. Nur mit aktivem PSRAM steht genügend Speicher zur Verfügung, um das Modell stabil auszuführen. Dies geschieht über die Board-Einstellungen in der Arduino IDE: Unter Werkzeuge → PSRAM muss die Option Enabled ausgewählt werden (vgl. Abbildung 3.2).

Anschließend wird die aus Edge Impulse exportierte Modell-Bibliothek in das Projekt eingebunden. Dies geschieht über Sketch → Bibliothek einbinden → ZIP-Bibliothek hinzufügen, wobei die zuvor heruntergeladene ZIP-Datei ausgewählt wird. Nach dem Import stehen alle Funktionen des Modells (Initialisierung, Inferenz, Auswertung) im eigenen Arduino-Sketch zur Verfügung und kann im Code

eingebunden werden. Damit ist die Entwicklungsumgebung vollständig eingerichtet, und das Modell kann direkt auf dem XIAO ESP32S3 ausgeführt werden (vgl. Abbildung 3.2).

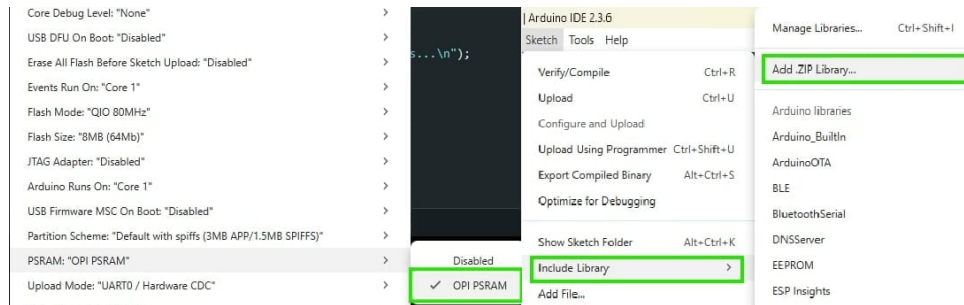


Abbildung 3.2: PSRAM einstellen und ZIP-Bibliothek einbinden

## 3.2 Struktur und Funktionsweise des Steuerungsprogramms

Das Steuerungsprogramm wurde in der Arduino IDE implementiert und steuert den Ablauf der Bildaufnahme, Objekterkennung und Aktorenregelung. Die Software basiert auf der aus Edge Impulse exportierten Arduino-Bibliothek und verwendet zusätzlich die Kamera-Bibliothek des XIAO ESP32S3 Sense. Nach dem Einbinden der Header-Dateien werden die Hardware-Pins für LEDs und Propeller definiert sowie die Kamera des XIAO ESP32S3 Sense initialisiert.

In der Funktion **setup()** erfolgt die Initialisierung der seriellen Schnittstelle, die Konfiguration der Kamera und die Festlegung der Ein- und Ausgänge. Anschließend beginnt der kontinuierliche Inferenzprozess, der in der Funktion **loop()** ausgeführt wird.

Die Hauptschleife **loop()** umfasst folgende Verarbeitungsschritte:

1. Erfassung eines Kamerabildes.
2. Übergabe der Bilddaten an das Edge-Impulse-Modell.
3. Ausführung der Inferenz und Ermittlung der Bounding-Box-Koordinaten für das erkannte Objekt.
4. Berechnung der Objektposition im Bildfeld (links, rechts oder zentral).
5. Ansteuerung der Ausgabekomponenten (LEDs und Propeller) entsprechend der erkannten Position.

Die Steuerungslogik ist wie folgt definiert:

- Wird kein Objekt erkannt (Konfidenz  $\neq 0,5$ ), bleiben beide LEDs aus, und beide Propeller laufen mit mittlerer Drehzahl.

- Wird ein Baum im linken Bildbereich erkannt, leuchtet die blaue LED, und der rechte Propeller wird aktiviert, um eine Ausweichbewegung nach rechts einzuleiten.
- Wird ein Baum im rechten Bildbereich erkannt, leuchtet die rote LED, und der linke Propeller wird aktiviert, um eine Bewegung nach links auszuführen.
- Wird das Objekt zentral erkannt, bleiben beide LEDs aus, und beide Propeller werden gestoppt.

Die Ausführungsergebnisse der Inferenz (Bounding-Box-Koordinaten, Klassifikationswert, Berechnungszeiten) werden über den seriellen Monitor ausgegeben (vgl. Abbildung 3.3).

```
Baum (0.781250) [ x:8, y:48, w:16, h:16 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.734375) [ x:8, y:24, w:8, h:8 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.906250) [ x:8, y:40, w:16, h:24 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
No object (score=0.50) -> straight ahead
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
No object (score=0.00) -> straight ahead
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
No object (score=0.00) -> straight ahead
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.964844) [ x:8, y:40, w:8, h:24 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.906250) [ x:8, y:48, w:16, h:16 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
No object (score=0.00) -> straight ahead
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.625000) [ x:56, y:24, w:8, h:8 ] -> zentral
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.625000) [ x:56, y:24, w:8, h:8 ] -> zentral
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.855469) [ x:8, y:48, w:8, h:8 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
Baum (0.625000) [ x:16, y:56, w:8, h:8 ] -> links
Predictions (DSP: 4 ms., Classification: 155 ms., Anomaly: 0 ms.):
No object (score=0.50) -> straight ahead
```

Abbildung 3.3: Serieller Monitor: Inferenz-Ausgaben und Richtungsentscheidung

Der Programmaufbau ist modular gestaltet, sodass Kamera, Inferenz und Steuerung getrennt initialisiert und gewartet werden können. Alle Prozesse laufen lokal auf dem Mikrocontroller, wodurch eine Echtzeitreaktion ohne externe Rechenressourcen gewährleistet ist.

# 4 Prototypenaufbau

## 4.1 Beschreibung des Prototyps

Der Prototyp besteht aus einem Mikrocontroller, einer integrierten Kamera, zwei Motoren mit Propellern, mehreren elektronischen Komponenten sowie einer Batterie. Alle Hardwareelemente sind kompakt auf einer leichten Trägerstruktur montiert, die von mehreren Heliumballons getragen wird. Dadurch kann sich das System frei im Raum bewegen und autonom auf erkannte Hindernisse reagieren.

Als zentrale Steuereinheit wird ein XIAO ESP32S3 Sense eingesetzt. Dieses kompakte Mikrocontroller-Board verfügt über eine integrierte Kamera und bietet trotz seiner geringen Größe ausreichend Rechenleistung, um ein mit Edge Impulse trainiertes Machine-Learning-Modell direkt auszuführen. Die Kamera erfasst kontinuierlich Bilddaten, welche in Echtzeit vom Modell analysiert und klassifiziert werden.

Die Erkennungsergebnisse werden unmittelbar in Steuersignale umgesetzt. Erkennt das Modell ein Objekt im linken oder rechten Bildbereich, so aktiviert der Mikrocontroller das entsprechende Ausgangssignal: Zwei LEDs visualisieren den Erkennungsstatus – die linke (blau) und rechte (rot) LED zeigen an, auf welcher Seite sich ein Hindernis befindet. Gleichzeitig steuern zwei Motoren mit Propellern die Bewegungsrichtung, indem sie gezielt die linke oder rechte Seite des Systems ansteuern.

Durch das Zusammenspiel von Kamera, ML-Modell, LED-Signalgebung und Motorsteuerung entsteht ein leichtgewichtiges, autonomes System, das Umgebungsobjekte in Echtzeit erkennt, visuell rückmeldet und selbstständig ausweicht (vgl. Abbildung 4.1).

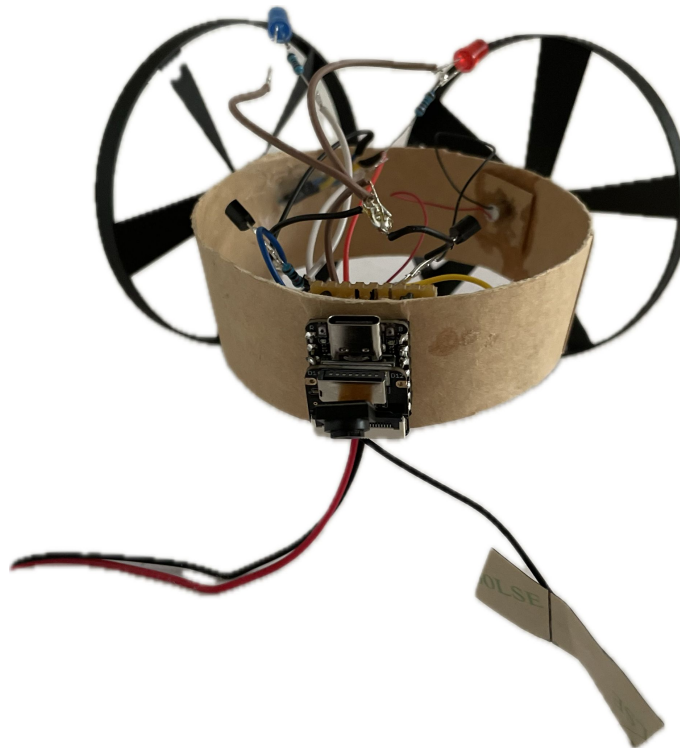


Abbildung 4.1: Prototypaufbau des Projekts „In The Woods“

## 4.2 Elektronisches Signal- und Steuersystem

Das System des Prototyps gliedert sich in zwei funktionale Teilsysteme: ein Signal-System zur optischen Richtungsanzeige und ein Motorantriebs-Subsystem zur Bewegungskontrolle (vgl. Abbildung 4.2).

Im Signal-System dienen zwei LEDs der visuellen Anzeige der erkannten Bewegungsrichtung. Die linke, blaue LED ist über den Pin D0, die rechte, rote LED über den Pin D1 mit dem Mikrocontroller verbunden. Die Kathode (kurzes Bein) jeder LED ist über einen Vorwiderstand mit dem jeweiligen Steuerpin verbunden, während die Anode (langes Bein) auf Masse liegt. Diese Beschaltung ermöglicht eine zuverlässige und eindeutige Signalisierung: Leuchtet die blaue LED, wurde ein Objekt auf der linken Seite erkannt; leuchtet die rote LED, befindet sich das Objekt auf der rechten Seite.

Das Motorantriebs-Subsystem steuert die Bewegung des Systems. Der Minuspol jedes Gleichstrommotors ist mit dem Drain eines 2N7000-MOSFETs verbunden. Gate und Source des Transistors sind über einen Widerstand gekoppelt, wobei das Gate das Steuersignal des Mikrocontrollers erhält (Pin D3 für den linken, Pin D8 für den rechten Motor). Die Source liegt auf Masse. Der Drain jedes Transistors ist zusätzlich mit der Anode einer Freilaufdiode verbunden, deren Kathode am Pluspol des jeweiligen Motors liegt. Die Kathoden beider Dioden sind miteinander verbunden und bilden einen gemeinsamen Knoten.

In der ursprünglichen Version des Aufbaus fehlte jedoch die direkte Verbindung dieses Knotens zur positiven Versorgungsspannung (3,3 V / 5 V). Dadurch war der Stromkreis unvollständig, sodass die Motoren keine Versorgung erhielten und nicht anlaufen konnten.

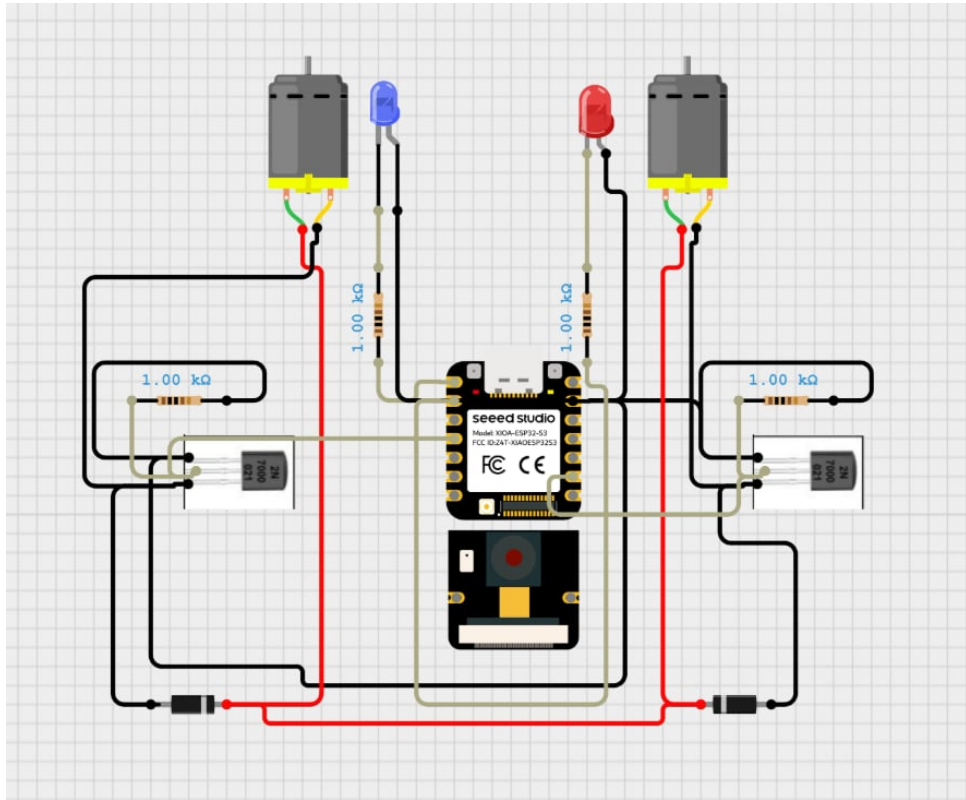


Abbildung 4.2: Alter, nicht funktionsfähiger Aufbau

## 4.3 Testphase und Beobachtung

Für die Tests wurden künstliche Bäume in einem Innenraum aufgestellt, um verschiedene Szenarien zu simulieren und die Funktionalität des Prototyps zu überprüfen. Die Testreihen dienten der Bewertung der Erkennungsleistung des Modells, der Reaktionsgeschwindigkeit des Systems sowie des Zusammenspiels zwischen Sensorik und Aktorik.

**Erkennung und Klassifikation:** Das in Edge Impulse trainierte FOMO-Modell erkannte Objekte mit dem Label „Baum“ zuverlässig, solange diese vollständig im Kamerabild erfasst und ausreichend beleuchtet waren. Die Klassifikation erfolgte stabil mit durchschnittlichen Konfidenzwerten zwischen 0,70 und 0,90. In Situationen mit ungleichmäßiger Beleuchtung oder geringer Bildschärfe kam es vereinzelt zu Fehlklassifikationen, was auf die begrenzte Trainingsdatenmenge und die geringe Auflösung ( $96 \times 96$  Pixel) zurückzuführen ist.



**Reaktionsverhalten:** Die Signalausgabe über die LEDs funktionierte einwandfrei und reagierte unmittelbar auf die Ergebnisse der Objekterkennung.

- Bei Objekten im rechten Bildbereich leuchtete die rote LED, um „Baum rechts“ anzuzeigen.
- Bei Objekten im linken Bildbereich leuchtete die blaue LED, um „Baum links“ anzuzeigen.
- Wurde kein Objekt erkannt, blieben beide LEDs ausgeschaltet.

Die Propeller reagierten dagegen nicht wie vorgesehen. Trotz korrekter Softwarelogik und Ausgangssignale an den Pins D3 und D8 kam es zu keiner Motorbewegung. Die Tests zeigten, dass die Signalübertragung vom Mikrocontroller zwar funktionierte, die elektrische Ansteuerung der Motoren jedoch fehlerhaft war.

**Systemverhalten:** Der Prototyp arbeitete in den LED-basierten Tests stabil und zeigte ein konsistentes Reaktionsverhalten. Die Kamera und das Modell liefen über längere Zeiträume ohne Fehlermeldungen, sofern der PSRAM aktiviert war. Das System reagierte in Echtzeit auf Änderungen im Bildfeld, wodurch die korrekte Funktion des KI-Modells bestätigt wurde.

**Bewertung:** Insgesamt konnte die Objekterkennung erfolgreich nachgewiesen werden. Das visuelle Feedback über die LEDs ermöglichte eine klare Darstellung der Systemreaktion. Der mechanische Antrieb über die Propeller war hingegen nicht funktionsfähig, was auf eine fehlerhafte Stromversorgung bzw. unzureichende Leistungssteuerung der Motorstufe zurückzuführen ist. Trotz dieses Defizits erfüllt der Prototyp die Kernanforderung der Echtzeit-Objekterkennung und reagiert konsistent auf erkannte Objekte.

## 4.4 Gründe für das Fehlverhalten

Die Funktionsstörung betraf ausschließlich den Motorantrieb. Ursache war die fehlende Anbindung des gemeinsamen Diodenknotens an die Plus-Versorgung. Dadurch entstand kein geschlossener Strompfad „+V → Motor → MOSFET → Masse“. Selbst bei aktivem Steuersignal blieb der Motor stromlos. Hinzu kam, dass die Freilaufdioden nicht parallel zu den Motoren, sondern in Serie geschaltet waren und somit keine Schutz- oder Schaltfunktion übernehmen konnten. Zusätzlich erwies sich der 2N7000-MOSFET bei 3,3 V Gate-Spannung als zu leistungsschwach, um die Motoren vollständig durchzuschalten. Folglich erreichten die Motoren nicht die erforderliche Spannung zum Anlaufen.

Das Signal-System mit den LEDs funktionierte dagegen einwandfrei und zeigte die erkannte Bewegungsrichtung korrekt an. Der Fehler lag somit ausschließlich in der Strompfad- und Leistungsdimensionierung des Motorantriebs.

## 4.5 Neuer, funktionsfähiger Aufbau

Auf Grundlage der Fehleranalyse wurde der ursprüngliche Aufbau überarbeitet und funktional optimiert. Der neue Aufbau gliedert sich – analog zum Zielsystem – in zwei funktionale Teilsysteme: ein Signal-System zur optischen Richtungsanzeige und ein Motorantriebs-Subsystem zur Bewegungskontrolle (vgl. Abbildung 4.3).

Das Signal-System visualisiert die erkannte Objektposition (z. B. Baum links oder rechts). Die linke blaue LED und die rechte rote LED werden jeweils direkt vom Mikrocontroller angesteuert. Ihre Anoden sind über separate Vorwiderstände zur Strombegrenzung mit den Pins D0 (links/blau) bzw. D1 (rechts/rot) verbunden, während die Kathoden auf Masse liegen. Leuchtet die blaue LED, signalisiert dies „Objekt links“; leuchtet die rote LED, „Objekt rechts“. Diese Beschaltung realisiert ein eindeutiges HIGH  $\rightarrow$  EIN-Verhalten, und der LED-Strom bleibt innerhalb der sicheren Grenzen für die GPIO-Pins. Widerstandswerte zwischen  $680\,\Omega$  und  $1\,\text{k}\Omega$  sind für 3,3-V-Logik und typische LED-Ströme von 2–5 mA geeignet.

Das Motorantriebs-Subsystem versorgt jeden Gleichstrommotor aus der Spannungsquelle des Controllers. Der Pluspol liegt an 3V3 (nur für sehr kleine Motoren) oder – sofern verfügbar – an 5 V/VBUS. Parallel zum Motor ist eine Freilaufdiode installiert, deren Kathode (Markierungsring) an Motor-Plus und deren Anode an Motor-Minus angeschlossen ist. Der Minuspol jedes Motors wird low-side über einen 2N2222-Transistor geschaltet: Der Kollektor liegt an Motor-Minus, der Emitter an GND, und die Basis erhält ihr Steuersignal über einen Serien-Basiswiderstand von den Pins D3 (linker Motor) bzw. D8 (rechter Motor). Der Widerstand begrenzt den Basisstrom und gewährleistet ein sauberes Schaltverhalten.

Wird der Ausgang des Mikrocontrollers auf HIGH gesetzt, schaltet der Transistor durch (Sättigungsbereich), und der Motorstrom kann fließen. Bei LOW wird der Strom unterbrochen, und die parallel geschaltete Diode begrenzt die induktive Spannungsspitze, wodurch sowohl Transistor als auch Mikrocontroller geschützt werden. Für eine stabile Funktion ist eine gemeinsame Masseführung zwischen Logik- und Motorstromkreis erforderlich. Die Leitungen zur Diode sollten möglichst kurz gehalten werden, um Störimpulse zu vermeiden. Wird eine PWM-Ansteuerung zur Drehzahlregelung gewünscht, kann diese direkt auf D3 bzw. D8 erfolgen, wobei der mittlere Motorstrom innerhalb der zulässigen Grenzwerte des 2N2222 liegen muss.

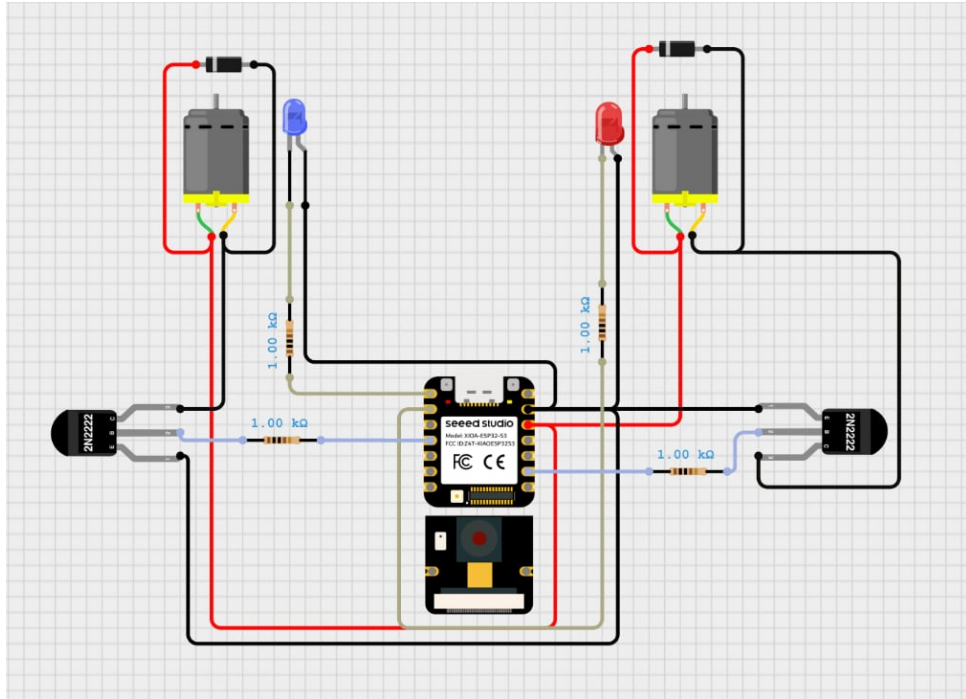


Abbildung 4.3: Funktionsfähiger Aufbau des Systems

## 4.6 Test des neuen Aufbaus und Simulationsergebnisse

Der überarbeitete Aufbau wurde zunächst in einer Simulation getestet, um die Funktionalität der Steuerlogik ohne reale Motoransteuerung zu überprüfen. Ziel war es, das Zusammenspiel von Kameradaten, Objekterkennung und Signalsteuerung zu validieren.

Im Testbetrieb erkannte das FOMO-Modell weiterhin zuverlässig Objekte mit dem Label „Baum“. Die LED-Ansteuerung reagierte konsistent auf die erkannten Positionen:

- Objekt links → blaue LED aktiv, rechte LED aus.
- Objekt rechts → rote LED aktiv, blaue LED aus.
- Kein Objekt erkannt → beide LEDs aus.

Die Signale an den Pins D3 und D8, die in der realen Hardware zur Motoransteuerung vorgesehen sind, wurden in der Simulation aufgezeichnet und überprüft. Dabei zeigte sich, dass die PWM-Ausgabe korrekt erzeugt und entsprechend der Erkennung angepasst wurde. Somit konnte die Funktionsfähigkeit der gesamten Softwarelogik, einschließlich Inferenz, Signalverarbeitung und Ausgangssteuerung, bestätigt werden.

Die Simulation diente zugleich als Nachweis, dass die Steuerarchitektur prinzipiell funktionsfähig ist und nach Integration geeigneter Treiberbausteine auch mit realen Motoren betrieben werden kann. Das System zeigte ein stabiles Laufverhalten, keine Speicherfehler und eine gleichmäßige Reaktionszeit im Bereich von wenigen Hundert Millisekunden.

Insgesamt belegt der simulierte Testaufbau, dass die Steuerlogik, das KI-Modell und die Signalverarbeitung korrekt miteinander interagieren. Die Verwendung der LEDs als Ersatzindikatoren für die Propellerbewegung erwies sich als praktikable Lösung, um die Entscheidungsprozesse des Systems sichtbar und nachvollziehbar zu machen.

## 5 Fazit und Ausblick

Mit dem Projekt *In The Woods* wurde erfolgreich ein funktionsfähiger Prototyp zur KI-gestützten Objekterkennung und Richtungssteuerung auf Basis eines XIAO ESP32S3 CAM entwickelt. Das System ist in der Lage, Bäume im Sichtfeld der Kamera zu erkennen, ihre Position im Bildfeld zu bestimmen und daraus in Echtzeit Steuerbefehle für LEDs und Propeller abzuleiten. Die Kombination aus Edge-Impulse-Modell, kompaktem Hardwareaufbau und klar strukturierter Steuerlogik führte zu einem vollständig autonomen, leichtgewichtigen System, das ohne externe Rechenleistung arbeitet. Die Tests zeigten, dass das Modell Bäume zuverlässig erkennt und die Reaktionszeiten des Systems für Echtzeitbetrieb ausreichend sind. Die gewählte Architektur auf Basis des FOMO-Ansatzes mit MobileNetV2 0.35 hat sich als praxistauglich erwiesen und bietet ein gutes Verhältnis zwischen Genauigkeit und Rechenaufwand. Das Konzept belegt, dass sich maschinelles Lernen auch auf sehr leistungsschwachen Mikrocontrollern erfolgreich einsetzen lässt.

Für zukünftige Weiterentwicklungen bieten sich mehrere Optimierungsmöglichkeiten an. Durch eine Erweiterung des Datensatzes – insbesondere mit Bildern unter variierenden Licht- und Wetterbedingungen – kann die Erkennungsgenauigkeit weiter verbessert werden. Ebenso wäre der Einsatz stärkerer Treiberstufen oder Motorcontroller sinnvoll, um größere Propeller oder alternative Antriebe sicher zu betreiben. Eine Erweiterung um drahtlose Kommunikation (z. B. Wi-Fi oder Bluetooth) würde zudem eine Fernüberwachung und Datenerfassung während des Flugs ermöglichen. Insgesamt zeigt das Projekt, dass KI-basierte Objekterkennung und autonome Steuerung auch mit einfacher, kostengünstiger Hardware realisierbar sind. *In The Woods* verbindet Elemente aus Big Data Analysis, eingebetteten Systemen und Umwelttechnologie und demonstriert damit eindrucksvoll, wie moderne Sensortechnik und maschinelles Lernen zur praktischen Erkundung und Interaktion mit der natürlichen Umgebung eingesetzt werden können.