



**Hochschule für Technik
und Wirtschaft Berlin**

University of Applied Sciences

Projektdokumentation

“Klassifizierung von Geräuschquellen im urbanen Raum mit TinyML”

Im Modul Wissensmanagement

Geleitet von Prof. Dr. Willner

Im Sommersemester 2023

Von

Alexander Bohm – 586446

&

Cosimo Jacker – 586447

https://github.com/BZion/WM_CityLAB

Inhaltsverzeichnis

Inhaltsverzeichnis.....	1
Projektziele.....	2
Hardware und Software.....	2
Fachkonzeption.....	3
Was ist die Idee.....	3
Für wen in welcher Situation ist die Idee?.....	4
Was soll durch die Idee ausgelöst werden?.....	4
Projektvorgehen.....	5
1. Prototyp – Auswahl der Hardware.....	5
Installation Arduino.....	6
Arduino IDE.....	6
Arduino CLI.....	6
Test des 1. Prototypen.....	7
Einrichten von Edge Impulse.....	8
Software Installation.....	8
Weboberfläche.....	9
Flashen der Firmware.....	9
Sammeln von Sound Samples.....	10
Erster Prototyp.....	11
Zweiter Prototyp.....	11
Training.....	12
Deployment.....	13
Projektergebnis.....	15
Ausblick.....	16
Quellen.....	17

Projektziele

Ziel dieses Projektes ist es, einen Prototypen zu entwickeln und dadurch die folgende Frage zu beantworten: Können wir Geräuschquellen in einer urbanen Umgebung mit TinyML auf Mikrocontrollern klassifizieren?

Im Rahmen der Summer School 2023 unter dem Motto “Stressfaktor Lautstärke – von der Berliner Löwin zum Schweigefuchs” sahen wir uns mit der folgenden Problemstellung konfrontiert.

Wir können uns schützen, wenn uns Temperaturanzeigen besondere Hitze oder Kälte anzeigen. Lichtsignale sorgen für Sicherheit und machen es allen Verkehrsteilnehmern einfacher, durch den Stadtverkehr zu navigieren. Aber was bedeutet die Lautstärke? Übermäßige Lautstärke schädigt das Hören und führt zu dauerhaftem Stress. Wir fragen uns, wie eine Lärmanzeige für den öffentlichen Raum aussehen, funktionieren und sogar akzeptiert werden kann, obwohl teure Technologien wie Lärmblitzer im Straßenverkehr besonders laute Autos identifizieren sollen. Wir suchen offene Lösungen für Datenvisualisierungen und Freifunk-Anwendungen.

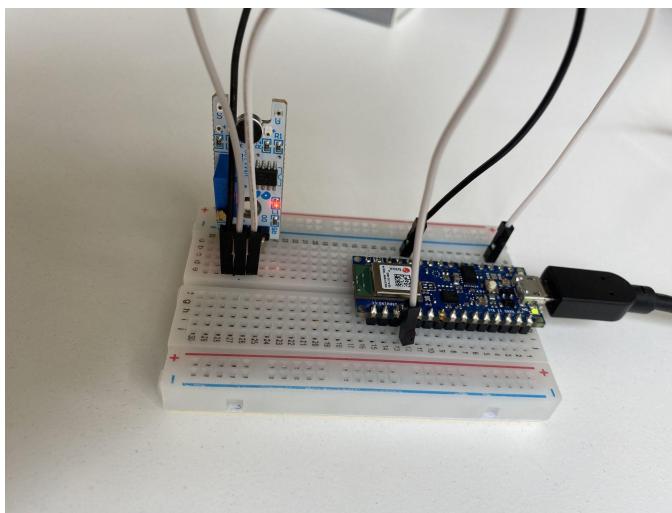
Hardware und Software

Im Folgenden wird beschrieben, welche Hard- und Software während der Umsetzung des Projektes genutzt wurde. Es hat sich herausgestellt, dass der erste Prototyp nicht wie erhofft funktioniert, daher wurde die Entwicklung gestoppt und mit dem zweiten Prototyp fortgesetzt. Eine detaillierte Beschreibung der Ursachen folgt im Kapitel Projektvorgehen.

1. Prototyp

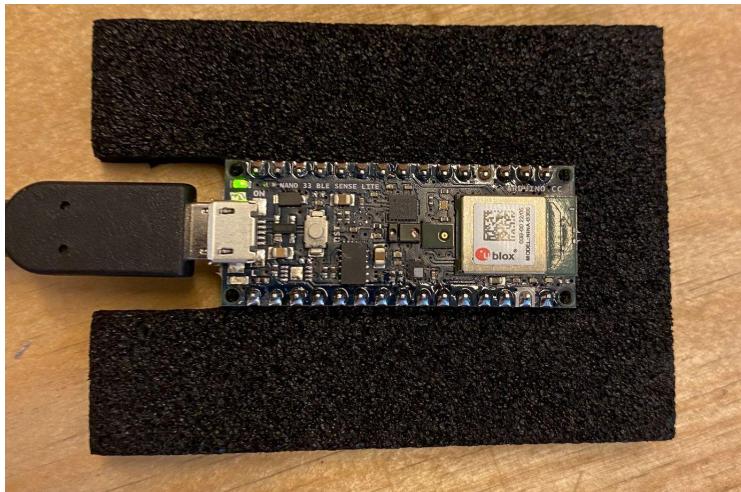
Arduino Nano 33 BLE

Whadda WPS 309 Schall Sensor



2. Prototyp

Arduino Nano 33 BLE Sense Lite



Die folgende Tabelle enthält die verwendete Software inklusive der Version sowie den zu diesem Zeitpunkt gültigen Link zum Download.

Name	Link	Version
Python 3	https://www.python.org/downloads/	3.11
Node.js	https://nodejs.org/de/download	18.18.0
Arduino IDE	https://www.arduino.cc/en/software	2.2.1
Arduino CLI	https://arduino.github.io/arduino-cli/0.34/installation/	0.34
Edge-Impulse-CLI	https://docs.edgeimpulse.com/docs/tools/edge-impulse-cli/cli-installation	1.21.1

Fachkonzeption

Im Folgenden wird jeweils die vorangehende Planung und Konzeption zur Erreichung des in Kapitel 1 beschriebenen Projektziels erläutert.

Was ist die Idee

Der erste Schritt wird darin bestehen, Audiodateien aus dem Internet zu sammeln, die verschiedene Geräuschquellen in einer städtischen Umgebung repräsentieren, darunter Sirenen von Einsatzfahrzeugen, Straßenverkehr, Baulärm und Gespräche. Diese Audiodateien werden abgespielt und mit einem Mikrofon aufgenommen, das mit dem Mikrocontroller verbunden ist, um eine umfassende Datenbank mit frischen

Aufnahmen zu erstellen. Im Anschluss daran werden wir diese Datenbank nutzen, um ein Modell mit Hilfe von Edge Impulse zu trainieren. Hierbei werden die gesammelten und verarbeiteten Daten verwendet, um das Modell auf die Klassifizierung verschiedener Geräuschquellen in einer urbanen Umgebung vorzubereiten. Nach erfolgreichem Training des Modells erfolgt die Übertragung auf den Mikrocontroller. Unser Ziel besteht darin, sicherzustellen, dass der Mikrocontroller in der Lage ist, neu aufgenommene Geräusche in einer urbanen Umgebung zu klassifizieren. Dies erfordert die Fähigkeit, die aufgenommenen Geräusche zu analysieren und zu identifizieren, ob es sich beispielsweise um das heulende Geräusch einer Sirene, das Dröhnen eines Flugzeugs, das Bellen eines Hundes oder das Rumpeln eines Müllwagens handelt.

Für wen in welcher Situation ist die Idee?

Dieses Projekt richtet sich an Ingenieure, Entwickler, Forscher und Technologie-Enthusiasten, die an der Entwicklung von Lösungen zur Geräusch-Klassifizierung in urbanen Umgebungen interessiert sind. Die Idee ist relevant in städtischen Umgebungen, wo die Lärmbelastung ein ernsthaftes Problem darstellt. Die Lärmbelastung in Städten kann negative Auswirkungen auf die Lebensqualität der Bewohner haben und sogar zu Gesundheitsproblemen führen. Dieses Projekt ist besonders relevant für:

1. **Stadtplaner und Stadtverwaltungen:** Sie können die Technologie nutzen, um Lärmquellen zu identifizieren und Maßnahmen zur Lärmbekämpfung zu ergreifen.
2. **Umweltorganisationen:** Sie können Daten zur Lärmbelastung sammeln, um Umweltauswirkungen zu bewerten und Maßnahmen zur Reduzierung von Lärm zu fördern.
3. **Bürger und Gemeinschaften:** Die Bürger können die Technologie nutzen, um die Lärmbelastung in ihrer Umgebung zu überwachen und bei Bedarf Maßnahmen zu ergreifen.

Was soll durch die Idee ausgelöst werden?

Das Hauptziel dieses Projekts ist es, einen Prototyp für eine kostengünstige und effiziente Methode zur Klassifizierung von Geräuschquellen in urbanen Umgebungen mithilfe von Tiny Machine Learning (TinyML) auf Mikrocontrollern zu entwickeln. Durch die Umsetzung dieser Idee sollen folgende Ergebnisse erzielt werden:

1. **Lärminderung:** Die Identifizierung und Klassifizierung von Lärmquellen ermöglicht es Städten und Gemeinschaften, gezielte Maßnahmen zur Lärmbekämpfung zu ergreifen. Dies kann zu einer Reduzierung der Lärmbelastung und damit zu einer Verbesserung der Lebensqualität führen.
2. **Umweltüberwachung:** Umweltorganisationen können die gesammelten Daten verwenden, um die Auswirkungen von Lärm auf die Umwelt zu bewerten und geeignete Schutzmaßnahmen zu ergreifen.
3. **Bürgerbeteiligung:** Die Idee ermöglicht es Bürgern, aktiv zur Überwachung der Lärmbelastung in ihrer Umgebung beizutragen und bei Bedarf auf störenden Lärm hinzuweisen.

4. **Technologische Innovation:** Die Entwicklung von TinyML-Anwendungen auf Mikrocontrollern fördert die technologische Innovation und trägt zur Miniaturisierung von Machine Learning-Modellen bei, was in einer Vielzahl von Anwendungen in der IoT-Welt nützlich sein kann.

Projektvorgehen

In diesem Kapitel werden die einzelnen Vorgehensschritte des Projekts in chronologischer Reihenfolge beschrieben und dokumentiert.

Erster Prototyp – Auswahl der Hardware

Bei der Planung und Entwicklung unseres Prototyps war eine gründliche Sichtung und Auswahl der möglichen Schallsensoren von entscheidender Bedeutung. Es ist erwähnenswert, dass die Mehrheit der verfügbaren Sensoren eher dazu neigt, lediglich das Vorhandensein von Schall wahrzunehmen, anstatt präzise Mikrofonfunktionalität zu bieten. Daher ist die Entscheidung für den Whadda WPS 309 Schall Sensor getroffen worden, da dieser laut Hersteller Dokumentation als Mikrofon verwendet werden kann.

Die sorgfältige Auswahl und Abwägung der verfügbaren Optionen führte schließlich zur Entscheidung, einen Mikrocontroller zu verwenden, der bereits in früheren Projekten erfolgreich eingesetzt wurde. Diese Entscheidung basierte auf den guten Erfahrungen, die vorherige Summer Schools mit diesem Modell gemacht haben. Wir sind zuversichtlich, von den Kenntnissen profitieren zu können. Daher ist die Entscheidung auf den Arduino Nano 33 BLE gefallen. Diese Version des Mikrocontrollers verfügt nicht über ein integriertes Mikrofon, sondern soll den externen Schallsensor nutzen.

Um den WPS 309 Schall Sensor mit dem Arduino Nano 33 BLE zu verbinden, waren drei Kabel und ein Steckbrett nötig (siehe Kapitel Hardware und Software). Die folgende Tabelle dokumentiert, welche Pins zusammengeschlossen wurden.

Microcontroller	Schallsensor	Beschreibung
GND	GND	Masse
3.3V	+	Spannungseingang
A7	AO	Daten

Damit war der Hardwareaufbau beendet und wir haben mit der Installation der nötigen Software begonnen.

Installation Arduino

Im folgenden Kapitel wird beschrieben, welche Software wir installiert haben, um den Mikrocontroller anzusprechen, sowie ein Machine Learning Modell mit Edge Impulse zu trainieren. Die beiden Prototypen unterscheiden sich nur in der verwendeten Hardware, die Software bleibt unverändert.

Arduino IDE

Der erste Schritt in unserem Entwicklungsprozess erforderte die Verwendung der Arduino IDE, um sicherzustellen, dass der Hardwareaufbau ordnungsgemäß funktioniert und alle Komponenten einwandfrei arbeiten. Um dies zu erreichen, verwendeten wir den in der Tabelle bereitgestellten Link, um die Arduino IDE herunterzuladen und zu installieren.

Die Installation selbst gestaltete sich reibungslos, und wir stießen dabei auf keinerlei Schwierigkeiten. Der heruntergeladene Installer führte uns durch den Installationsprozess und sorgte dafür, dass alle erforderlichen Dateien und Einstellungen korrekt eingerichtet wurden. Anschließend kann die Arduino IDE geöffnet werden. Für die Mikrocontroller Arduino Nano BLE 33 und BLE 33 Sense wird ein zusätzlicher Board Manager benötigt, dieser wurde wie folgt installiert:

In der Suchleiste vom Board Manager "Mbed OS Nano" eingeben, anschließend bei Arduino Mbed OS Nano Boards auf Installieren klicken. Wir haben die Version 4.0.6 verwendet.

Arduino CLI

Zusätzlich zur Arduino IDE wird auch das Programm Arduino CLI benötigt. Das Akronym "CLI" steht für Command Line Interface, und dieses Programm wird benötigt, um Edge Impulse entwickeln zu können.

Zur Installation des Befehls `arduino-cli` steht ein .sh Installations-Skript zur Verfügung. Dieses kann unter Mac und Linux ausgeführt werden. Auf Windows Computern wird dafür die zusätzliche Software Git Bash benötigt. Mit dem folgenden Befehl wird das Skript heruntergeladen und ausgeführt, anschließend wird `arduino-cli` in dem folgendem Ordner installiert: `$PWD/bin`

```
curl -fsSL https://raw.githubusercontent.com/arduino/arduino-cli/master/install.sh | sh
```

Damit der Befehl `arduino-cli` von jedem Ordner ausgeführt werden kann, sollte der Installationsordner der `arduino-cli` zur PATH Variable hinzugefügt werden.

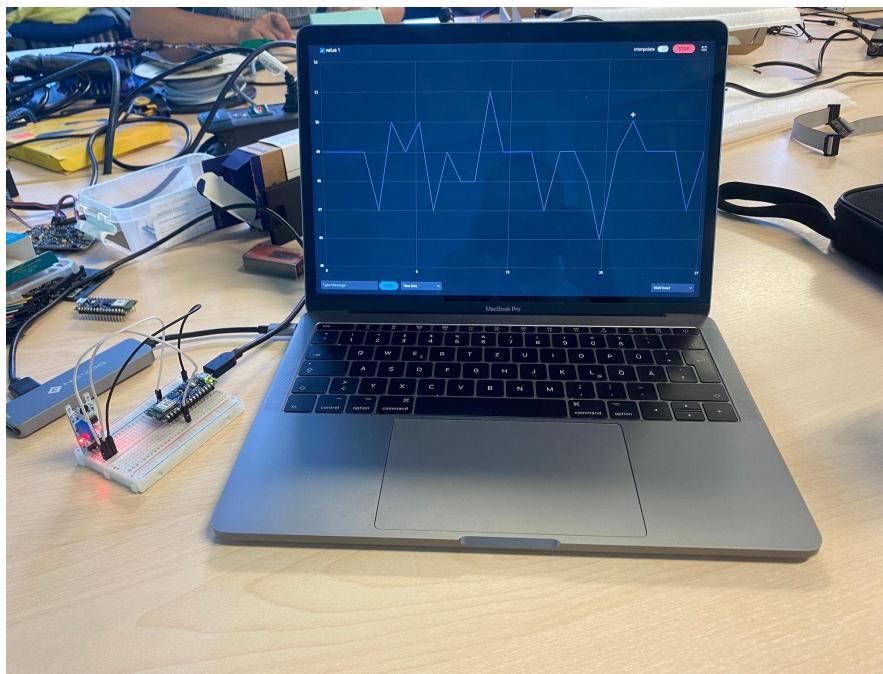
Test des ersten Prototypen

Nach diesen Schritten ist die nötige Software installiert, um den Aufbau und die Funktion des 1. Prototypen zu testen. Dazu haben wir ein kleines Programm geschrieben, welches den Input vom Mikrofon ausliest und das Ergebnis auf der Konsole ausgibt. Das Lesen und Ausgeben wird alle 500 Millisekunden wiederholt. Dadurch können wir erkennen, ob das Mikrofon korrekt angeschlossen ist und

funktioniert. Durch das Erzeugen eines lauten Geräusches wie z.B. das Schnipsen mit den Fingern erwarten wir einen Anstieg der Werte in der Ausgabe. Der Code sieht wie folgt aus:

```
int pin = A7;  
int value;  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
    pinMode(pin, INPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
    value = analogRead(pin);  
    delay(500);  
    Serial.println(value);  
}
```

Das folgende Bild zeigt links neben dem Laptop den ersten Prototypen sowie auf dem Laptop die Ausgabe des Arduino IDE Serial Plotters. Der Serial Plotter nimmt als Input die Ausgabe der Konsole aus der Arduino IDE und erstellt daraus eine fortlaufende Grafik.



Mit diesem Test wurde sichergestellt, dass die Hardware korrekt aufgebaut und angeschlossen wurde und dass die Hardware funktionsfähig ist. Durch die Höhen und Tiefen wird außerdem deutlich, dass verschiedene hohe und niedrige Zahlen je nach der Umgebungslautstärke vom Mikrofon an den Mikrocontroller gesendet werden. Trotz intensiver Recherchen im Internet und insbesondere durch die gründliche Prüfung der Herstelleranleitung fanden wir keine unmittelbare Möglichkeit, aus diesem Integer-Wert eine Sounddatei zu erzeugen. Das zuvor genannte Problem birgt das Potenzial, den gesamten

Prototypen unbrauchbar zu machen. Diese Schwierigkeit könnte die gesamte Funktionalität des Prototyps beeinträchtigen und stellt somit eine erhebliche Herausforderung für unser Projekt dar. Wir haben dennoch entschieden, mit der Entwicklung fortzufahren, denn wir wollten noch eine weitere Lösungsmöglichkeit testen.

Einrichten von Edge Impulse

In dem folgenden Kapitel werden wir im Detail darauf eingehen, wie wir die Einrichtung und Konfiguration von Edge Impulse durchgeführt haben, um sicherzustellen, dass die Plattform optimal für unsere Bedürfnisse funktioniert und wir die gewünschten Ergebnisse erzielen können.

Software Installation

Im ersten Schritt wird die folgende Software installiert:

- Python 3
- Node.js

Zur Installation kann der Link im Kapitel Hardware und Software verwendet werden.

Im zweiten Schritt wird Edge Impulse CLI installiert. Die Edge Impulse CLI wird benötigt, um lokale Geräte zu steuern, als Proxy zu fungieren, um Daten für Geräte zu synchronisieren, die nicht über eine Internetverbindung verfügen, und um lokale Dateien hochzuladen und zu konvertieren.

Unter Windows reicht der folgende Befehl zur Installation:

```
npm install -g edge-impulse-cli --force
```

Anschließend sollte `edge-impulse-cli` im PATH zur Verfügung stehen.

Unter Mac OS und Linux wurden die folgenden Befehle ausgeführt um sicherzustellen, dass Edge Impulse CLI dem PATH hinzugefügt wird:

```
mkdir ~/.npm-global  
npm config set prefix '~/.npm-global'  
echo 'export PATH=~/\.npm-global/bin:$PATH' >> ~/.profile
```

Die Installation wird mit dem folgendem Befehl abgeschlossen:

```
npm install -g edge-impulse-cli --force
```

Weboberfläche

Um Edge Impulse nutzen zu können, wird ein kostenfreier Account benötigt. Dieser wurde unter dem folgenden Link erstellt: <https://studio.edgeimpulse.com/signup>

Nach der erfolgreichen Registrierung kann ein neues Projekt erstellt werden. Beim Anlegen eines neuen Projekts wird ein Wizard angezeigt, um den Typ der Daten festzulegen und die passenden Einstellungen automatisch vorzunehmen. Das möchten wir in unserem Fall nicht und wählen daher die Option am Ende der Seite „I know what I'm doing, hide this wizard!“ aus. Bevor wir mit Edge Impulse fortfahren können, ist es nötig, die Firmware auf dem Arduino zu flashen. Nur so kann sich Edge Impulse mit Arduino verbinden.

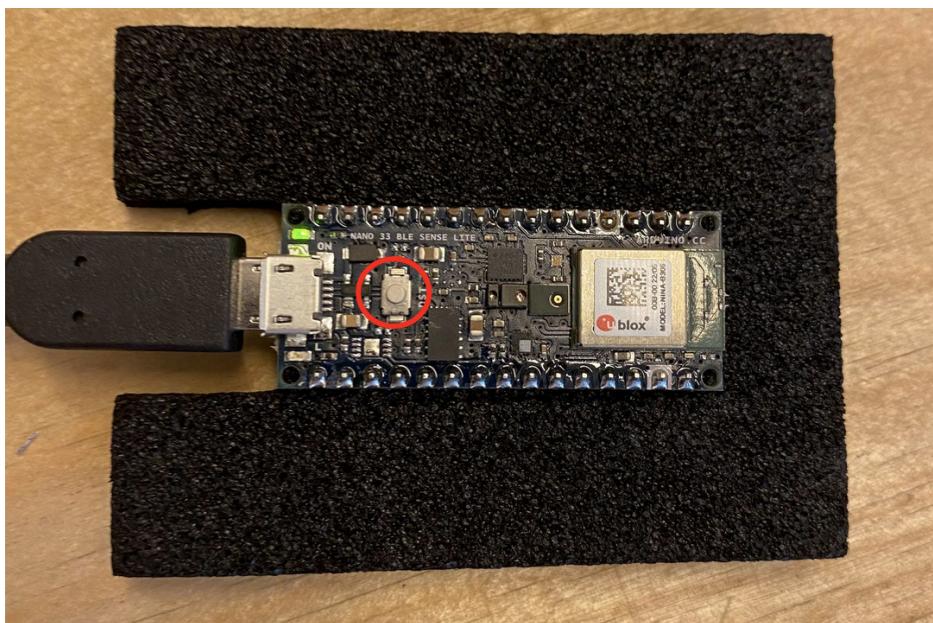
Flashen der Firmware

Die Firmware wird von Arduino auf Github zur Verfügung gestellt und kann unter dem folgendem Link heruntergeladen werden:

<https://github.com/edgeimpulse/firmware-arduino-nano-33-ble-sense>

Anschließend sind die folgenden Schritte auszuführen:

- Mikrocontroller per USB mit der Computer verbinden
- Die Reset Taste auf dem Arduino zweimal drücken, die LED sollte anfangen zu blinken



- Je nach Betriebssystem das entsprechende Firmware Skript ausführen z.B. flash_mac.command

```

[MacBook-Pro-2:arduino-nano-33-ble-sense alexanderbohm$ ./flash_mac.command
You're using an untested version of Arduino CLI, this might cause issues (found: 0.34.2, expected: 0.18.x)
Finding Arduino Mbed core...
Finding Arduino Mbed OK
Finding Arduino Nano 33 BLE...
Finding Arduino Nano 33 BLE OK
Flashing board...
Device      : nRF52840-QIAA
Version     : Arduino Bootloader (SAM-BA extended) 2.0 [Arduino:IKXYZ]
Address     : 0x0
Pages       : 256
Page Size   : 4096 bytes
Total Size  : 1024KB
Planes      : 1
Lock Regions: 0
Locked      : none
Security    : false
Erase flash

Done in 0.001 seconds
Write 352560 bytes to flash (87 pages)
[=====] 100% (87/87 pages)
Done in 13.886 seconds
New upload port: /dev/cu.usbmodem143301 (serial)

Flashed your Arduino Nano 33 BLE development board.
To set up your development with Edge Impulse, run 'edge-impulse-daemon'
To run your impulse on your development board, run 'edge-impulse-run-impulse'
MacBook-Pro-2:arduino-nano-33-ble-sense alexanderbohm$ ]

```

- Verbindung herstellen mit dem Befehl `edge-impulse-daemon`. Anschließend den Nutzernamen und Passwort des Edge Impulse Nutzers eingeben, der Mikrocontroller erhält einen Namen, in diesem Fall ‘Otto’, damit wir ihn in der Weboberfläche wiederfinden können.

```

PS C:\Users\cosim> edge-impulse-daemon
Edge Impulse serial daemon v1.21.1
Endpoints:
  WebSocket: wss://remote-mgmt.edgeimpulse.com
  API: https://studio.edgeimpulse.com
  Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to COM3
[SER] Serial is connected, trying to read config...
Failed to parse snapshot line [ ]
[SER] Retrieved configuration
[SER] Device is running AT command version 1.7.0

Configuring API key in device... OK
Failed to parse snapshot line [ ]
Failed to parse snapshot line [ ]
[SER] Device is not connected to remote management API, will use daemon
[WS] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS] Connected to wss://remote-mgmt.edgeimpulse.com
? What name do you want to give this device? otto
[WS] Device "otto" is now connected to project "Y_lo-project-1". To connect to another project, run `edge-impulse-daemon --clean`.
[WS] Go to https://studio.edgeimpulse.com/studio/281040/acquisition/training to build your machine learning model!

```

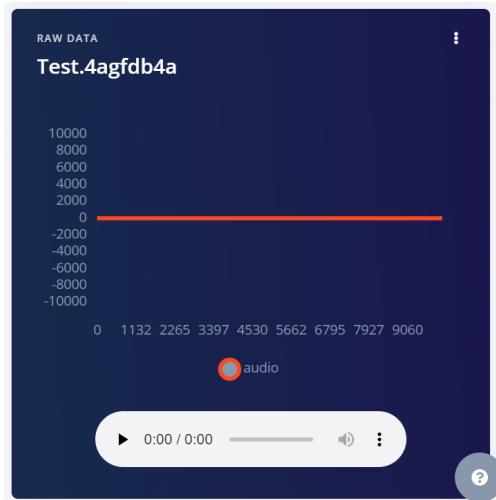
Damit ist der Mikrocontroller erfolgreich eingerichtet und kann mit Edge Impulse z.B. zum Sammeln von Sound Sample verwendet werden.

Sammeln von Sound Samples

Unsere Idee ist es, Sounds von Straßenverkehr, Martinshörnern und Baustellen im Internet zu sammeln und für optimale Trainingsbedingungen vom Mikrofon des Mikrocontrollers aufnehmen zu lassen. Mit diesen Sound Samples möchten wir dann unser Modell trainieren.

Erster Prototyp

Über den Tab Data Acquisition gelangen wir in Edge Impulse zur Webseite, in der wir Sound aufnehmen und speichern können. Das haben wir mit dem ersten Prototypen versucht. Die aufgenommene Datei ist zwar aufgenommen worden, doch leider ist nichts zu hören und auch auf dem Vorschaubild sind keine Ausschläge zu erkennen.



Wir stehen damit vor demselben Problem, dass wir zwar ein Mikrofon angeschlossen haben, aber keinerlei Dokumentation oder Beispiele finden, wie wir damit eine Sound-Datei aufnehmen können. Nach Rücksprache mit unseren Dozenten haben wir uns daher entschieden, den Mikrocontroller Arduino Nano BLE 33 Sense zu verwenden. Auf dieser Version des Mikrocontrollers ist ein Mikrofon integriert und wurde bereits mit Edge-Impulse getestet.

Zweiter Prototyp

Alle zuvor beschriebenen Schritte wurden bis zu dem Punkt sammeln der Sound Samples in Edge-Impulse erneut ausgeführt, um den zweiten Prototypen verwenden zu können. Mit dem zweiten Prototypen war der Test, einen Sound aufzunehmen erfolgreich, wir konnten über das Mikrofon einen Sound in Edge-Impulse speichern und wiedergeben. Wir können daher mit dieser Hardware fortfahren.

Edge Impulse bietet die Möglichkeit, direkt aus dem Dashboard heraus Geräusche über das Mikrofon eines verbundenen Mikrocontrollers aufzunehmen und in den Datensatz zu übertragen. Dazu muss nur im Reiter Data Acquisition das richtige Gerät ausgewählt, das Label für das ML-Training vergeben und die Aufnahmedauer ausgewählt werden. Mit dem Nano BLE Sense 33 war uns es möglich, maximal 19 Sekunden pro Aufnahme zu erfassen. In dieser Zeit können mehrere Geräusche zum passenden Label aufgenommen werden. Nach automatischer Übertragung der Audiodatei vom Mikrokontroller zu Edge Impulse besteht dort die Option, eine Audiodatei in mehrere kleinere Dateien zu trennen.

Dataset Data explorer Data sources | CSV Wizard

DATA COLLECTED
5m 10s

TRAIN / TEST SPLIT
75% / 25%

Dataset

Training (36) Test (12)

SAMPLE NAME	LABEL	ADDED	LENGTH
Verkehr.4aj1k0fi.s1	Verkehr	Yesterday, 12...	6s
Hintergrund.4aj2j...	Hintergrund	Yesterday, 12...	6s
Baustelle.4aj0goe...	Baustelle	Yesterday, 12...	9s
Hintergrund.4aj1...	Hintergrund	Yesterday, 12...	6s

Collect data

Device ?
Helmut

Label
Hintergrund

Sample length (ms.)
19000

Sensor
Built-in microphone

Frequency
16000Hz

Start sampling

Unser Datensatz umfasste am Ende insgesamt Audiodateien mit einer Gesamtlänge von 5 Minuten und 10 Sekunden. Aufgeteilt in einen Trainings- und einen Testdatensatz sowie über die vier Klassen "Martinshorn", "Baustelle", "Verkehr" und "Hintergrund" standen uns somit rund 60 Sekunden pro Label an Daten zur Verfügung, die wir zum Training des Modells nutzen konnten.

Training

Für das Training muss zunächst ein Impuls erstellt werden. Bei diesem werden die grundsätzlichen Rahmenbedingungen des Trainings festgelegt. Für unser Projekt haben wir einen Datensatz bestehend aus Audio-Dateien benutzt. Dementsprechend haben wir den Audio MFE processing block genutzt. Dieser extrahiert Zeit- und Frequenz Informationen aus den Audiosignalen, die für das Training verwendet werden können, und eignet sich besonders gut für Projekte, die keine Spracherkennung erzielen wollen. Darauf wird ein Klassifikator aufgesetzt, welcher die vom MFE extrahierten Merkmale als Input nutzt, um die vier, von uns definierten Klassen vorherzusagen.

Time series data

Input axes
audio

Window size
1000 ms.

Window increase
500 ms.

Frequency (Hz)
16000

Zero-pad data

Audio (MFE)

Name
MFE

Input axes (1)
audio

Classification

Name
Classifier

Input features
MFE

Output features
4 (Baustelle, Hintergrund, Martinshorn, Verkehr)

Output features

4 (Baustelle, Hintergrund, Martinshorn, Verkehr)

Save Impulse

Add a processing block

Add a learning block

Mit Erstellung des Impuls ist der Trainingsprozess noch nicht angestoßen, sondern nur definiert. Es werden zwei neue Reiter erzeugt, mit den beim processing und learning block angegebenen Namen. Bei uns sind dies "MFE" und "Classifier". Bei "MFE" muss zunächst die Extrahierung der Merkmale aus dem Datensatz durchgeführt werden. Anschließend kann bei "Classifier" das Modell trainiert werden. Mit Abschluss des Trainings wird das Ergebnis des Models auf dem Validierungsdatensatz angezeigt. Da dieses nur bedingt aussagekräftig ist, kann über den Reiter "Model testing" die Performance des Models auf dem Testdatensatz überprüft werden. Mit unserem Modell konnten wir eine Genauigkeit von 81,29 Prozent erreichen.



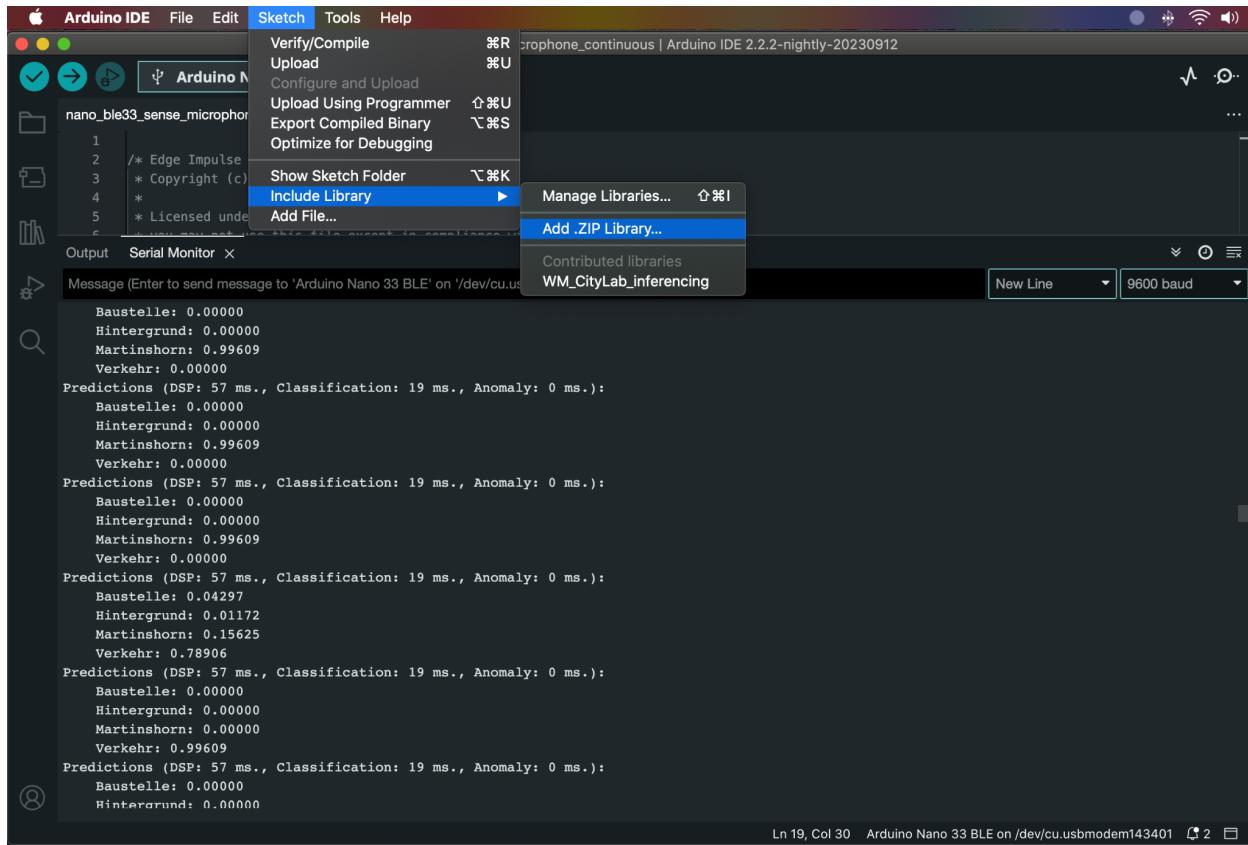
	BAUSTELLE	HINTERGRU	MARTINSHC	VERKEHR	UNCERTAIN
BAUSTELLE	52.6%	0%	21.1%	26.3%	0%
HINTERGRUND	0%	100%	0%	0%	0%
MARTINSHORN	0%	8.8%	91.2%	0%	0%
VERKEHR	14.7%	0%	0%	85.3%	0%
F1 SCORE	0.63	0.96	0.85	0.79	

Mit einem Blick auf die Konfusionsmatrix, bei der die vorhergesagten Label in den Spalten stehen und die tatsächlichen in den Zeilen, ist zu erkennen, dass das Modell die Klassen Martinshorn und Hintergrund sehr gut abgrenzen kann. Besonders bei Baustellen stößt das Problem jedoch an eine Grenze. Hier werden nur rund die Hälfte der Testdateien richtig als Baustelle klassifiziert. Dies ist jedoch mit den Trainingsdaten erklärbar. Bei vielen Audiodateien für die Klasse Baustelle, sind auch Motorengeräusche ähnlich dem Straßenverkehr zu hören. Für eine bessere Unterscheidung zwischen diesen Klassen ist die Datenmenge vermutlich nicht ausreichend.

Deployment

Nach erfolgreichem Training muss das Modell noch auf den Mikrocontroller gebracht werden. Dazu kann eine Library aus Edge Impulse exportiert werden, die anschließend in beliebigen Skripten eingebunden werden kann. Da wir mit dem Arduino Nano BLE Sense und der Arduino IDE gearbeitet haben, haben wir das Modell direkt als Arduino Library exportiert. Um diese anschließend nutzen zu können, muss die Library zuerst in der Arduino IDE hinzugefügt werden. Um die Library der Arduino IDE hinzuzufügen, bitte wie im Screenshot zu sehen vorgehen: Sketch → Include Library → Add .ZIP Library... und

anschließend über den Explorer die Datei auswählen. Unter dem Titel Contributed libraries sollte nun die neue Library auftauchen.



Im nächsten Schritt kann der Sourcecode auf den Mikrocontroller geladen werden. In der Arduino IDE wird der Code kompiliert, dies kann beim ersten Mal etwas länger dauern. Danach kann der Code auf den Mikrocontroller geladen werden, dazu den Upload Button drücken. Der folgende Screenshot zeigt den Upload am Beispiel.

The screenshot shows the Arduino IDE interface. The title bar reads "nano_ble33_sense_microphone_continuous | Arduino IDE 2.2.2-nightly-20230912". The main window displays the code for "nano_ble33_sense_microphone_continuous.ino". Below the code editor is the "Output" tab, which shows the serial monitor output. The output consists of a series of progress bars indicating the upload of a file, with each bar followed by its percentage and page count (e.g., "60% (26/43 pages)", "62% (27/43 pages)", etc.). At the bottom of the output window, it says "Done in 6.848 seconds". A progress bar at the bottom right of the output window shows "Uploading..." with a green progress bar. The status bar at the bottom of the IDE shows "Ln 19, Col 30" and "Arduino Nano 33 BLE on /dev/cu.usbmodem143401".

Projektergebnis

Im folgenden Kapitel wollen wir unser Projektergebnis vorstellen. Unser Ziel war es, Geräuschquellen in einer urbanen Umgebung mit TinyML auf Mikrocontrollern zu klassifizieren. Mit unserem Prototypen ist es uns gelungen, die Klassen Martinshorn, Baustelle, Verkehr und Hintergrund relativ präzise vorherzusagen. Einmal alle 50 Millisekunden nimmt der Mikrocontroller über sein Mikrofon Geräusche auf und berechnet die wahrscheinlichste Klassenzugehörigkeit, die Ergebnisse werden in der Konsole ausgegeben. Der folgende Screenshot zeigt, wie die Klasse Martinshorn vorhergesagt wurde als ein Krankenwagen vorbei fuhr.

The screenshot shows the Arduino Serial Monitor window. The title bar says "Serial Monitor". The message area displays several lines of text representing sound classification predictions. The text includes:

```

Output Serial Monitor X
Message (Enter to send message to 'Arduino Nano 33 BLE' on '/dev/cu.usbmodem143401')
New Line 9600 baud

Baustraße: 0.00000
Hintergrund: 0.00000
Martinshorn: 0.99609
Verkehr: 0.00000
Predictions (DSP: 57 ms., Classification: 19 ms., Anomaly: 0 ms.):
Baustraße: 0.00000
Hintergrund: 0.00000
Martinshorn: 0.99609
Verkehr: 0.00000
Predictions (DSP: 57 ms., Classification: 19 ms., Anomaly: 0 ms.):
Baustraße: 0.00000
Hintergrund: 0.00000
Martinshorn: 0.99609
Verkehr: 0.00000
Predictions (DSP: 57 ms., Classification: 19 ms., Anomaly: 0 ms.):
Baustraße: 0.04297
Hintergrund: 0.01172
Martinshorn: 0.15625
Verkehr: 0.78906
Predictions (DSP: 57 ms., Classification: 19 ms., Anomaly: 0 ms.):
Baustraße: 0.00000
Hintergrund: 0.00000
Martinshorn: 0.00000
Verkehr: 0.99609
Predictions (DSP: 57 ms., Classification: 19 ms., Anomaly: 0 ms.):
Baustraße: 0.00000
Hintergrund: 0.00000

```

At the bottom right of the monitor window, it says "Ln 19, Col 30 Arduino Nano 33 BLE on /dev/cu.usbmodem143401".

Zusätzlich zum trainierten Modell sind auch die von uns gesammelten Daten Teil unseres Projektergebnisses. Die Sound Dateien sind in unserem Git zu finden und können für das Training oder Testen in anderen Projekten verwendet werden.

Unser Source Code sowie die erstellten Sound Daten sind unter dem folgenden Git Repository verfügbar:

https://github.com/BZion/WM_CityLAB

Ausblick

In zukünftigen Forschungen kann dieses Projekt als Grundlage genommen werden, mit dem Ziel, die Qualität des Modells zu verbessern. Ein möglicher Gegenstand dabei könnte sein, unterschiedliche Mikrofone zu verwenden, um die Auswirkung derer Differenzen auf das Modell zu ermitteln. Insbesondere stellt sich die Frage, ob es einen merklichen Unterschied macht, ob die aufgezeichneten Klänge mit dem im Controller eingebauten Mikrofon oder einem externen Mikrofon aufgenommen wurden. Auch interessant ist die Frage, ob ein Modell Daten, die mit einem anderen Mikrofon aufgenommen wurden als die Trainingsdaten, genauso gut klassifizieren kann.

Da wir bisher nur mit Audiodateien aus dem Internet gearbeitet haben, fehlt uns ein Vergleich zu direkt auf der Straße aufgenommenen Sounds. Daher wäre ein Live-Test sinnvoll, um die Performance des Modells unter realen Bedingungen zu evaluieren. Gegebenenfalls ergibt sich daraus die Notwendigkeit, die Trainingsdaten an dem Ort aufzunehmen, an dem das Gerät zum Einsatz kommen soll. Dabei kann auch zwischen einem internen und einem externen Mikrofon verglichen werden.

Durch eine solch systematische Herangehensweise können wertvolle Einblicke für die Verbesserung des Modells gewonnen werden, um schlussendlich eine möglichst präzise Vorhersage über die Geräuschquellen zu ermöglichen.

Quellen

ARDUINO SA. “Installation.” Installation - Arduino CLI, 2020, arduino.github.io/arduino-cli/0.33/installation/.

Edge Impulse. “Installation.” Installation - Edge Impulse Documentation, Apr. 2023, docs.edgeimpulse.com/docs/tools/edge-impulse-cli/cli-installation.

EdgeImpulse Inc. “Edge Impulse Dashboard.” *WM_CityLab*, 2023, studio.edgeimpulse.com.

Reichert, Sarah, and Reni Safitri. “Offene Werkstatt: Du Verstehst Mich Nie – Ich Probier’s Mal Mit KI!” TU Berlin, 2023.