

Gesture to Emoji

Sebastian Stigler sebastian.stigler@hs-aalen.de
Hochschule Aalen - Technik und Wirtschaft

Was wird in diesem Aufgabenblatt alles gemacht

- Bibliothek für die IMU¹ wird installiert.
- Daten der IMU werden visualisiert.
- Trainingsdaten aufgezeichnet.
- Das Machine Learning Modell wird trainiert.
- Die Machine Learning Applikation wird mit dem trainierten Modell implementiert.
- Die Applikation wird so angepasst, dass Emojis für die erkannten Bewegungen gesendet werden.
- Experimentieren mit den Sensorwerten.

1 Projekte laden und Bibliothek installieren

Starten Sie VSCode und fügen Sie die drei Projekte (**Emoji_Keyboard**, **IMU_Capture** und **IMU_Classifier**) aus dem **platformio** Verzeichnis zum aktuellen Arbeitsbereich hinzu (siehe Abb. 1).

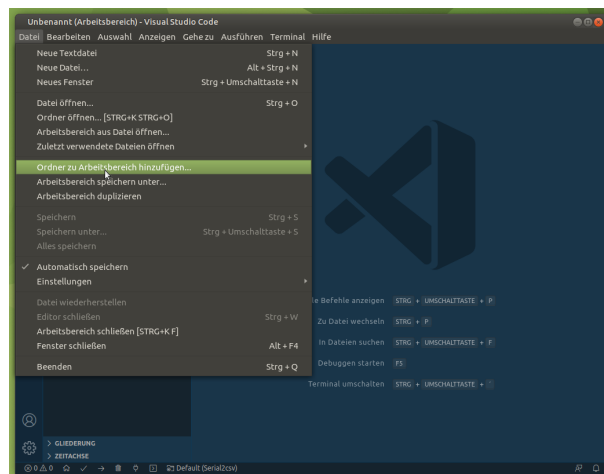


Abbildung 1: Ordner zum aktuellen Arbeitsbereich hinzufügen.

Anschließend installieren Sie die Bibliothek **Arduino_LSM9D1** für die IMU in die Projekte **IMU_Capture** und **IMU_Classifier** (siehe Abb. 2-5). Für **Emoji_Keyboard** benötigen Sie die Gestensensor-Bibliothek **Arduino_APDS9960**.

¹Inertia Measuring Unit

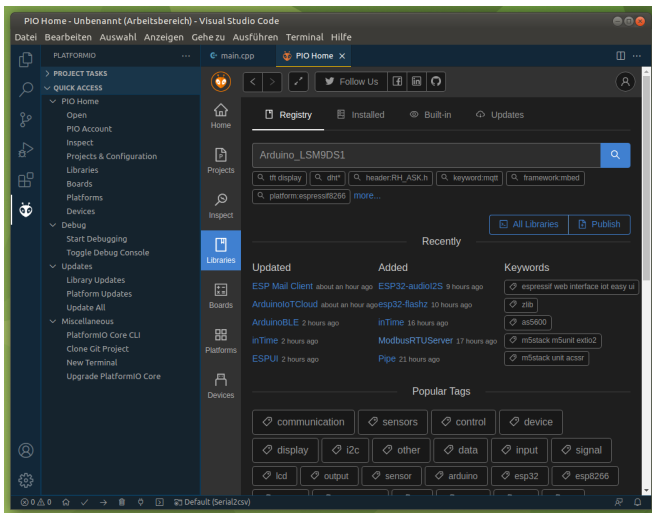


Abbildung 2: Auf das PIO Symbol an der Seite klicken und den Punkt Libraries auswählen.

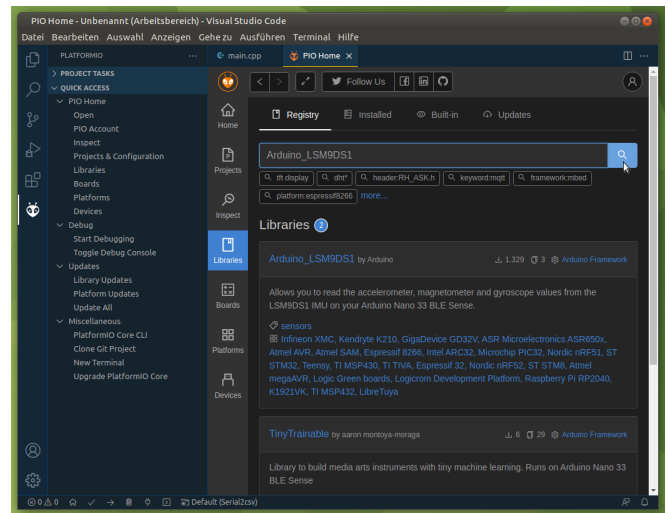


Abbildung 3: Den Namen der Bibliothek ins Suchfeld eintragen.

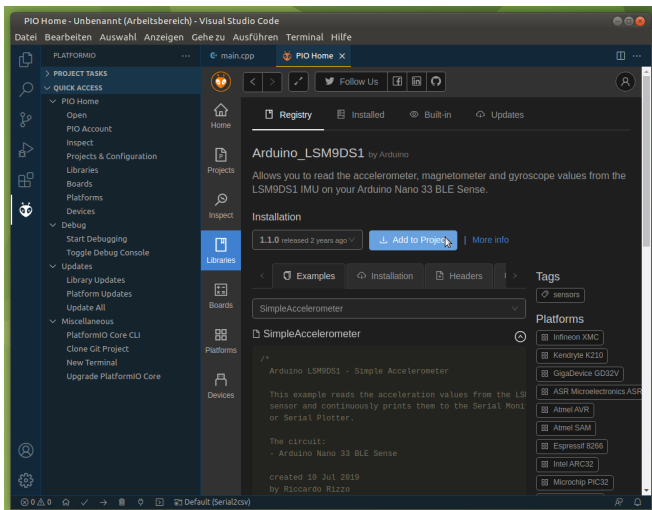


Abbildung 4: Die passende Bibliothek auswählen und auf Add to Project klicken.

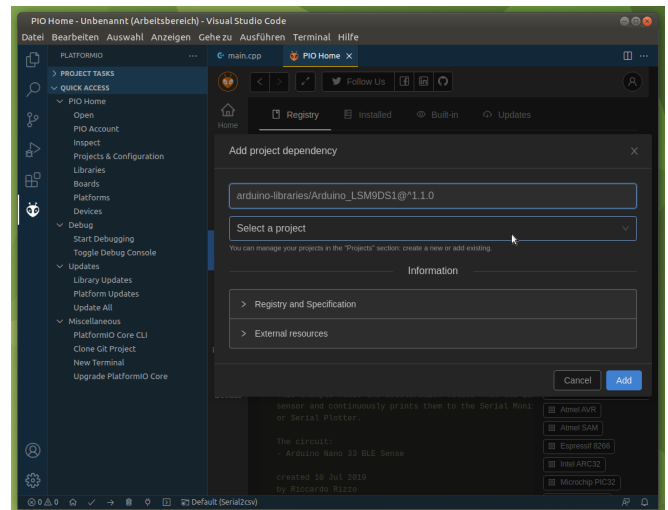


Abbildung 5: Das Projekt im Dropdown Menü auswählen.

2 Visualisieren Sie die Daten der IMU

Wechseln Sie das PIO Projekt Environment (siehe Abb. 6) auf IMU_Capture.

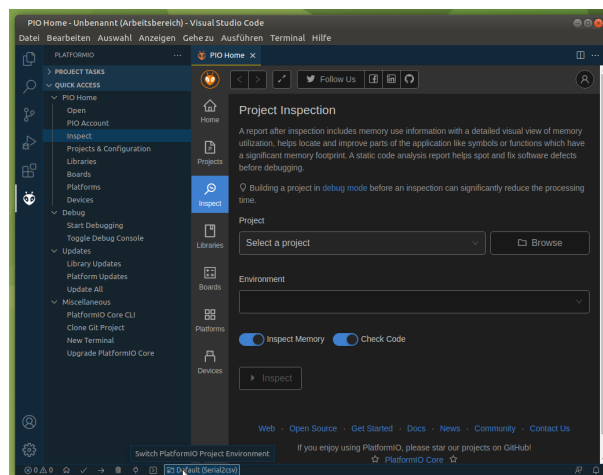


Abbildung 6: Klicken Sie in der Fußzeile auf Switch to PlatformIO Project Environment um das aktuelle Projekt festzulegen.

Verbinden Sie den Arduino Nano 33 BLE Sense mit Ihrem Rechner und klicken Sie in der Fußzeile von VSCode auf den Pfeil nach Rechts, um das Projekt zu übersetzen und auf den Mikrocontroller zu schreiben.

Verwenden Sie das Programm **SerialPlot** um die serielle Ausgabe zu visualisieren.

Auf der Seite [Hackaday.io: SerialPlot - Realtime Plotting Software](https://hackaday.io/project/161142-serialplot) gibt es für Linux ein [AppImage](#)² und für Windows eine [Setup.exe](#).

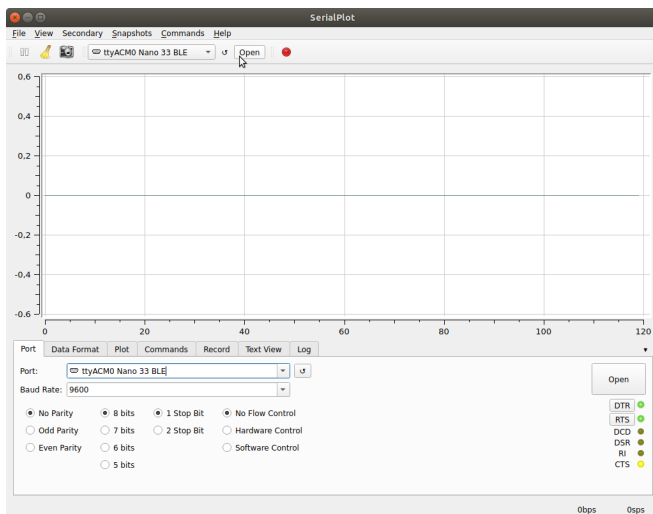


Abbildung 7: Wählen Sie die serielle Schnittstelle aus und stellen Sie die Baud Rate auf 9600 und klicken Sie auf Open. Stellen Sie sicher, dass der Mikrocontroller am Rechner angeschlossen ist. Drücken Sie den Refresh Button neben dem Dropdownmenü, falls die Schnittstelle nicht angezeigt wird.

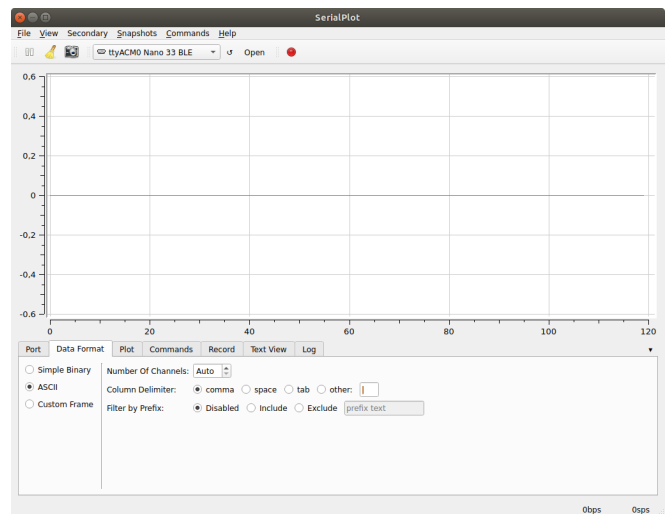


Abbildung 8: Wählen Sie im Data Format Reiter den **ASCII** Modus, den Column Delimiter auf **comma** und Filter by Prefix auf **Disabled**.

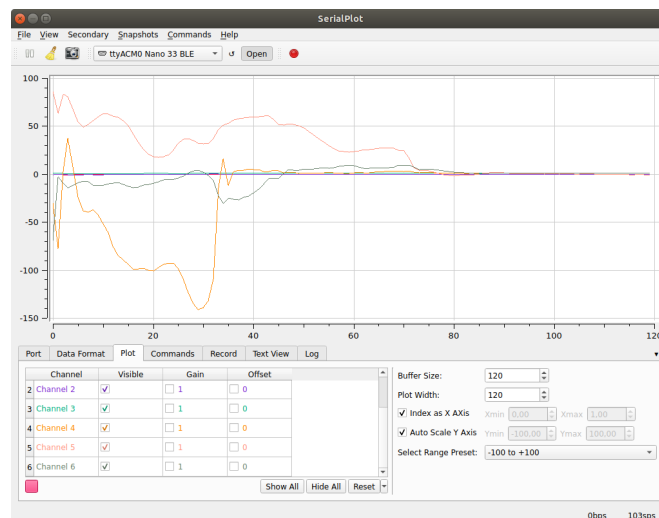


Abbildung 9: Im Plot Reiter setzen Sie die Buffer Size und Plot Width auf **120**. Setzen Sie den Hacken bei Index as X Axis und Auto Scale Y Axis.

Die Abbildungen 7-9 zeigen, wie man **SerialPlot** konfiguriert.

Eine Ausgabe wie in Abb. 9 erhalten Sie sobald der Mikrocontroller eine Beschleunigung erfährt. Der Mikrocontroller sendet dann 119 Datensätze und eine Leerzeile. Wenn Sie bei den Werten der letzten drei Kanäle den Gain auf **0.001** setzen, dann sind die Werte der Ausgaben des Beschleunigungssensors und des Gyroskops in einem ähnlichen Bereich. Alternativ können Sie im View Menü die **Multi Plot** Option auswählen, in der jeder Kanal einen eigenen Plot bekommt.

²Einfach herunterladen und ausführbar machen. Die Datei kann dann direkt gestartet werden.

3 Trainingsdaten aufzeichnen

Das Aufzeichnen der Trainingsdaten wird im Terminal in VSCode gemacht.

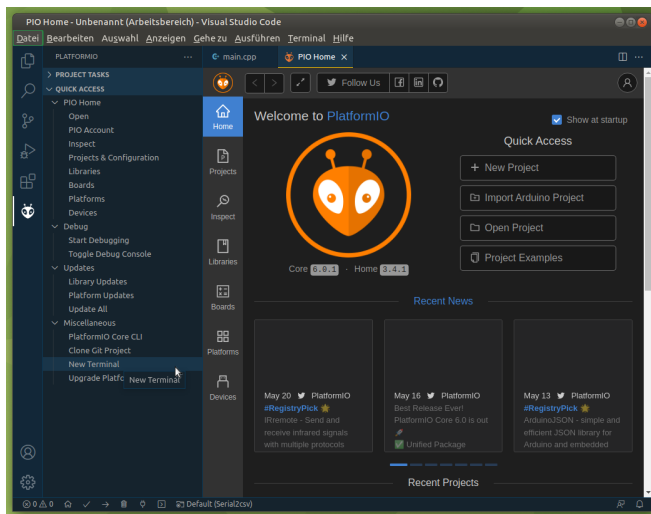


Abbildung 10: Klicken Sie an der linken Seite des Bildschirms auf das PIO Symbol und klicken Sie unter QUICK ACCESS → Miscellaneous → New Terminal.

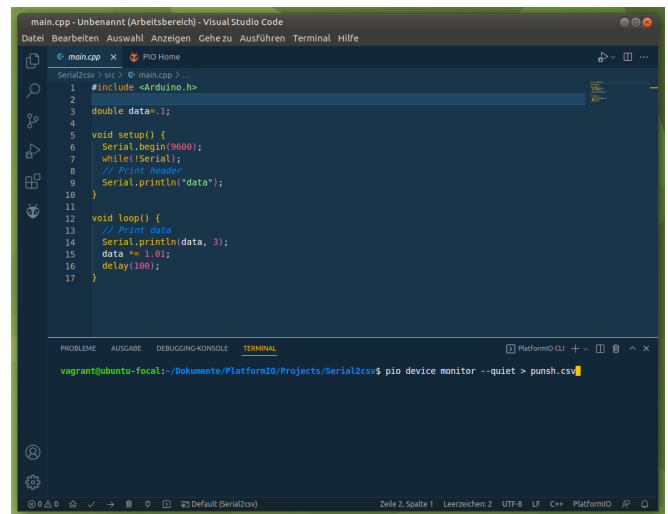


Abbildung 11: Um Trainingsdaten für den Punsh aufzuzeichnen, rufen Sie im Terminal den folgende Befehl auf: `pio device monitor --quiet > punsh.csv`

In Abb. 10 sehen Sie wie man ein Terminal öffnet, so dass das `pio` Kommando verfügbar ist.

Zeichnen Sie nun die Daten für die Punsh und Flexbewegungen auf:

1. Drücken Sie die **Reset Taste** auf dem Arduino Nano 33 BLE Sense.
2. Rufen Sie `pio device monitor --quiet > punsh.csv` im Terminal von VSCode auf (siehe Abb. 11).
3. Führen Sie 10 mal eine Schlagbewegung (Punsh) mit dem Mikrocontroller in der Hand aus.
4. Brechen Sie die Aufzeichnung im Terminal mit `Strg+c` ab.
5. Drücken Sie die **Reset Taste** auf dem Arduino Nano 33 BLE Sense.
6. Rufen Sie `pio device monitor --quiet > flex.csv` im Terminal von VSCode auf.
7. Führen Sie 10 mal eine Beugebewegung Ihres Arms (Flex) mit dem Mikrocontroller in der Hand aus.
8. Brechen Sie die Aufzeichnung im Terminal mit `Strg+c` ab.

Die Dateien `punsh.csv` und `flex.csv` werden in den nächsten Teilaufgabe benutzt.

4 Das Machine Learning Modell trainieren

Öffnen Sie die Colab (<https://colab.research.google.com>) und melden Sie sich mit Ihrem Googleaccount an.

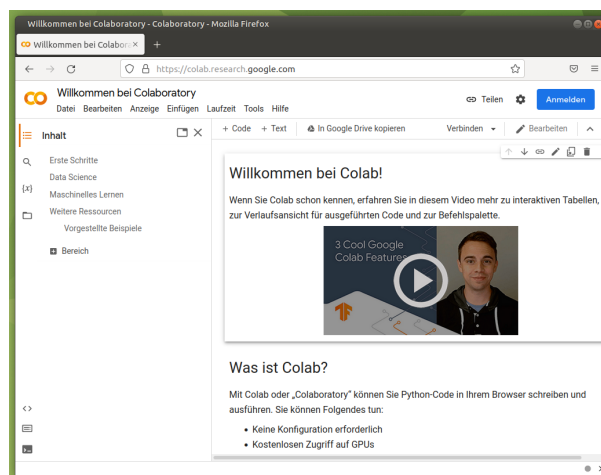


Abbildung 12: Shreenshot der Goole Colab Webseite

Klicken Sie dort auf Datei → Notebook hochladen und laden Sie das Jupyter Notebook `colab/training_guestures.ipynb` hoch. Folgen Sie nun den Anweisungen im Notebooks.

Das Notebook erzeugt die `model.h` Datei, die Sie in der nächsten Teilaufgabe benutzen werden.

5 Implementieren der Applikation mit dem trainierten Modell

Kopieren Sie die `model.h` Datei in das `src` Verzeichnis unter `platformio/IMU_Classifier`, übersetzen Sie das Projekt und laden Sie auf den Mikrocontroller.

Starten Sie nun in VSCode den seriellen Monitor (Steckersymbol in der Fußleiste). Wenn Sie jetzt mit dem Mikrocontroller eine Punsh oder Flex Bewegung ausführen, dann sollten Sie im seriellen Monitor sehen, dass der Mikrocontroller diese beiden Bewegungen unterscheiden kann.

6 Sende Emojis statt Text

Wechseln Sie nun in VSCode zum Projekt `Emoji_Keyboard`. Übersetzen Sie das Programm und laden Sie es auf den Mikrocontroller.

Gehen Sie nun im Browser auf die Seite https://image.informatik.htw-aalen.de/~stigler/jufo_display/ und klicken Sie einmal in die Mitte der Seite.

Sie können jetzt mit dem Gestensensor auf dem Mikrocontroller die Bewegungen Ihrer Hand in vier Gesten übersetzen und das Ergebnis als Emoji anzeigen lassen. Streichen Sie dafür im Abstand von weniger als 10cm von Oben nach Unten, von Links nach Rechts, von Unten nach Oben und von Rechts nach Links über die Öffnung im Mikrocontrollergehäuse.

Passen Sie nun das Projekt `IMU_Classifier` derart an, dass Sie **statt** der seriellen Ausgabe die Elemente aus dem Quellcode `Emoji_Keyboard` benutzen, um die Ergebnisse der Klassifikation als Emojis auszugeben.

Sie benötigen dafür die Header `PluggableUSBHID.h` und `USBKeyboard.h` die bereits im Framework für den Mikrocontroller enthalten sind. Ferner benötigen Sie die `printString` Funktion sowie die Strings für die Emojis.

Die Strings sind dabei wie folgt aufgebaut:

- Er startet mit einem Großen **U**.
- Es folgen genau sechs hexadezimale Zahlen (Groß-/Kleinschreibung egal), die dem Code des Emojis entsprechen. Wenn der Code weniger als sechs Stellen hat, dann fügen Sie führende Nullen hinzu.

Beispiel

Gegeben ist folgender Code für das Emoji:

Code	Emoji
U+1F600	😄

Dann verwenden Sie für den Keyboardstring den Wert `"U01F600"`.

Unter <https://www.unicode.org/emoji/charts/full-emoji-list.html> finden Sie eine Liste aller Emojis. Benutzen Sie die Browser Spalte um zu sehen, wie die Emojis dargestellt werden.

7 Experimentieren mit den Sensorwerten

In diesem Abschnitt können Sie mit den Trainingsdaten ein wenig Experimentieren.

7.1 Untersuchen Sie, welche Komponente des IMU Sensors für die Erkennung der Geste wichtig ist

Im Jupyter Notebook in Colab können Sie im Abschnitt Train Neural Network → Parse and prepare data im Quellcode entscheiden ob nur Beschleunigungswerte (aX-aZ) oder nur Gyroskopwerte (gX-gZ) fürs Training benutzt werden.

Kommentieren Sie einfach die nicht verwendeten Komponenten aus und trainieren Sie dann das Modell.

Beispiel

Es sollen nur die Gyroskopwerte verwendet werden:

```
...
# normalize the input data, between 0 to 1:
# - acceleration is between: -4 to +4
# - gyroscope is between: -2000 to +2000
tensor += [
    # (df['aX'][index] + 4) / 8,
    # (df['aY'][index] + 4) / 8,
    # (df['aZ'][index] + 4) / 8,
    (df['gX'][index] + 2000) / 4000,
    (df['gY'][index] + 2000) / 4000,
    (df['gZ'][index] + 2000) / 4000
]
...
```

Im Quellcode vom IMU_Classifier müssen Sie dann ebenfalls den Code anpassen:

Fortsetzung des Beispiels

```
...
// normalize the IMU data between 0 to 1 and store in the model's
// input tensor
/*
tflInputTensor->data.f[samplesRead * 6 + 0] = (aX + 4.0) / 8.0;
tflInputTensor->data.f[samplesRead * 6 + 1] = (aY + 4.0) / 8.0;
tflInputTensor->data.f[samplesRead * 6 + 2] = (aZ + 4.0) / 8.0;
*/
tflInputTensor->data.f[samplesRead * 3 + 0] = (gX + 2000.0) / 4000.0;
tflInputTensor->data.f[samplesRead * 3 + 1] = (gY + 2000.0) / 4000.0;
tflInputTensor->data.f[samplesRead * 3 + 2] = (gZ + 2000.0) / 4000.0;
...
```

Hinweis: Beachten Sie, dass sich die Indizes ebenfalls geändert haben, da jedes Sample jetzt nur noch drei statt sechs Werte hat.

7.2 Fügen Sie ein oder zwei neue Bewegungen hinzu

Hinweis: Benutzen Sie hier wieder sowohl die Beschleunigungs- als auch die Gyroskopwerte!

Zeichnen Sie ein oder zwei neue Bewegungen auf. Achten Sie darauf das Sie in etwa die selbe anzahl von Trainingsbewegungen aufzeichnen.

Auch hier muss im Jupyter Notebook und im Quellcode eine Anpassung durchgeführt werden.

Im Jupyter Notebook muss im Abschnitt Train Neural Network → Parse and prepare data die GESTURES Liste um die neuen Gesten erweitert werden.

Auch im IMU_Classifier muss das GESTURES Array analog angepasst werden. Achten Sie darauf das die Reihenfolge im Notebook und in C++ Code übereinstimmen.

7.3 Verbessern Sie die Erkennungsrate

Erhöhen Sie die Anzahl der Trainingsbewegungen für jede Geste. Halten Sie dabei auch den Mikrocontroller in unterschiedlichen Positionen.