BSc Computational and Data Science

CDS205 Computer Science

Dozentin: Ana Petrus

# PWM-Dashboard

## Ein simples Dashboard zur grafischen Anzeige eines PWM-Signals für die Verwendung in Ferngesteuerten Modellautos, Booten und Flieger

Silvan Joos*

[1]*Fachhochschule Graubünden*

**E-Mail-Adresse: silvan.joos@stud.fhgr.ch*

10. Januar 2025

**Abstract**

Ein simples, kostengünstiges Dashboard zur Visualisierung von PWM-Signalen bietet Modellbaufanatiker die Möglichkeit, ihre Projekte möglichst detailreich und realitätsgetreu umzusetzen. Kommerzielle Lösungen sind mehrheitlich teuer und nur mit einer begrenzten Auswahl an Modellen kompatibel. Diese Arbeit beschreibt die Entwicklung eines kompakten und universellen Dashboards, das Empfängerdaten mittels einem Raspberry Pi verarbeitet und anzeigt.

Die Resultate weisen darauf hin, dass die Verwendung von CircuitPython auf einem Adafruit QT Py RP2040 eine schnelle Signalverarbeitung und Visualisierung ermöglicht. Dabei überzeugt das Dashboard insbesondere durch seine Kosteneffizienz. Mit Materialkosten von knapp 20 US-Dollar ist es rund fünfmal günstiger als vergleichbare kommerzielle Lösungen. Trotz einiger Einschränkungen bietet das entwickelte Dashboard eine solide Grundlage für zukünftige Projekte.

# 1 Einleitung

Für Modellbaufanatiker ist das massstabtreue Nachbauen eines Modells die Königsdisziplin. Flugzeuge, Boote oder auch Autos werden mit viel Handarbeit sorgfältig angefertigt und gebaut, sodass jedes Detail möglichst dem Original entspricht. Auch die Funktionalität der Modelle ist von hoher Priorität.

Funktionale Fahrwerke, Scheinwerfer und Positionslichter sind beliebte Ergänzungen. Bereiche mit hoher Detailgenauigkeit, wie Cockpits, werden dabei häufig vernachlässigt, da kleine Anzeigen und Lichter bisher kommerziell kaum erhältlich waren. Hier setzt diese Arbeit an, um eine kostengünstige und universelle Lösung für digitale Dashboards zu entwickeln.

Unternehmen wie MACH-13 nutzen diese Marktlücke und bieten dank hochauflösender Displays realistische digitale Dashboard an, die Echtzeitdaten des Modells auswerten. (MACH-13, o. J.)

Diese Lösungen sind jedoch kostspielig und aufgrund der engen Marktnische nur für eine begrenzte Anzahl von Modellen verfügbar.

# 2 Forschungsfrage und Methodik

Aufgrund der hohen Kosten verfügbarer Lösungen und der fehlenden Modellunterstützung ergibt sich folgende Fragestellung:

*Wie kann ein platzsparendes und universelles Dashboard zur Verarbeitung und Anzeige von Empfängerdaten mit einem Raspberry Pi umgesetzt werden?*

**Wahl des Mikrokontrollers**

Der Raspberry Pi 4B ist Aufgrund seiner Grösse und Gewicht ungeeignet, weshalb auf kleinere Alternativen gesetzt wird. Der Adafruit QT Py RP2040 erfüllt mit seinem 32bit «RP2040» Raspberry Pi Prozessor und rund 22mm auf 18mm Grösse die Anforderungen vollumfänglich (Kostenpunkt: $9.95).*(Adafruit, o. J.)* Dies ermöglicht ebenfalls die Verwendung in Flugzeugen und Modellen mit wenig Platz.

Als Betriebssystem dient CircuitPython 9.2.2, eine open source Version von Python für Mikrocontroller. CircuitPython eignet sich besonders zum Erlernen von Programmierung, da es die Interaktion zwischen Code und Realität vereinfacht. (*CircuitPython*, 2025)

Als IDE und zur Kommunikation mit dem QT Py dient Thonny (Version 4.1.6).

**Datenfluss und Signalverarbeitung**

Um Daten auf dem Dashboard anzeigen zu können muss zuerst Zugang zu diesen verschafft werden. Der Datenfluss vom Modell kann direkt über einen Pin, hier *Pin A1* auf dem QT Py, mittels integriertem CircuitPython Modul *digitalio* ausgelesen werden. Ein 10kΩ Widerstand zwischen *Pin A1* und dem Signalkabel verhindert ein Übersteuern und dämpft somit das Signal.

Der Empfänger eines Modells übermittelt das Signal mit Pulsweitenmodulation (PWM) an die betroffenen Komponenten wie ein Servo oder Fahrtenregler. Bei PWM, einer quadratischen Welle, pulsiert das Signal zwischen Masse (GND oder *low)* und +5 Volt (*high*). Die Dauer des *high* Signals in Relation zum *low* Signal über eine Zeitspanne bestimmt somit den

Tastgrad (Duty Cycle)[1].(*Pulse Width Modulation (PWM)*, 2024)

Die einfachste Methode für eine zuverlässige Auslesung ist eine Messung der Signaldauer im Zustand *high*. *(CustomRaspi, 2021)*

Um die Signale nutzbar zu machen, werden folgende Funktionen implementiert:

Die Funktion *pwm_pulse* erfasst den Zeitstempel mittels dem Modulm *time*, sobald ein *high* Signal erkannt wird und berechnet bei einem folgenden *low* das Delta der beiden Zeitstempel. Besteht über die Dauer einer Sekunde ein durchgehendes *high* oder *low* wird das als Timeout erkannt.

Die Funktion *pwm_percentage* normalisiert die gemessene Zeit auf einen Bereich zwischen *0* und *100%*. Dafür müssen zuvor das Minimum und Maximum im Setup definiert werden. Diese Werte können direkt auf dem Display mittels der Funktion *pwm_debug_menu* ausgelesen werden, welches die gemessenen Werte aus *pwm_pulse* zeigt. Für das Wechseln mehrerer Menus sorgt ein Kippschalter zwischen *Pin A0* und Masse (GND).

**Display und grafische Anzeigen**

Das Display ist ein monochromes 0.66-Zoll OLED Display von seeed studio mit einer Auflösung von 64*48 Pixel und ist gerade mal 20mm*20mm gross (Kostenpunkt: $5.50).(*Seeed Studio*, o. J.) Es wird über die Pins *SCL* und *SDA* mittels dem Protokoll *I2C* angesteuert. Das Protokoll ist Teil des CircuitPython integrierte Moduls *busio* und erlabut serielle Kommunikation zwischen verschiedenen Geräten oder Controller. (*CircuitPython*, 2025)

Mittels dem Modul *adafruit_ssd1306* 2.12.18, ein Treiber für Displays erlaubt das Zeichnen von Pixel, Text und Linien, sowie die Steuerung sämtlicher Aspekte des Displays wie Helligkeit und Kontrast.

Die Funktion *draw_circle* implementiert den Bresenham-Algorithmus zum Zeichnen eines Kreises.(*Bresenham's Circle Drawing Algorithm*, 2024) Gewählt wurde der Algorithmus aufgrund des einfachen Aufbaus und Implementation.

Mit der Funktion *draw_speedometer* bildet das Display mittels modifiziertem Bresenham-Algorithmus nur die obere Hälfte eines Kreises ab. Zuvor findet eine Umrechnung des Werts aus *pwm_percentage* statt. Dies zeichnet schluss-endlich vom Mittelpunkt eine Linie zum korrespondierenden Punkt auf dem Halbkreis. Als letzter Schritt wird die Funktion *draw_circle* aufgerufen, die einen Kreis rund um das Dashboard abbildet.

**Ablauf vom Programm**

Der Ablauf des Programmes ist folgender: Während der Schalter auf der Position *0* ist, wird das Signal ausgelesen, verarbeitet und in einer Variable gespeichert. Das Dashboard wird zur Reduktion von Artefakten nur alle drei Zyklen gezeichnet. Die aktuelle Prozentzahl wird unterhalb des Dashboards hingeschrieben. Während der Schalter auf Position *1* ist, wird das Debug-Menu aufgerufen. Der Status vom Schalter wird bei jedem Durchlauf geprüft. Bei einem Programmabbruch zeigt das Display den Fehlertext „Error“

[1] Ausserhalb des Umfangs dieser Arbeit. Mehr Infos dazu in folgendem Artikel: https://www.geeksforgeeks.org/duty-cycle/

# 3 Resultate

Im Rahmen dieser Arbeit wurde ein platzsparendes (ca. 4.4cm$^2$) und universell einsetzbares digitales Dashboard[2] zur Anzeige von Empfängerdaten entwickelt. Das Dashboard wird auf einem kompakten OLED-Display gezeigt, welches von einem Adafruit QT Py RP2040 betrieben wird.

Die implementierten Module und Funktionen ermöglichen eine schnelle Auslesung und Darstellung des PWM-Signals in Form eines Tachometers Diese Lösung erwies sich als effektiv und zuverlässig. Zudem ermöglicht das integrierte Debugging-Menü eine einfache Kalibrierung des Dashboards auf PWM-fähige Empfänger, was eine universelle Einsetzbarkeit in verschiedenen Modelltypen gewährleistet.

Eine zu häufige Aktualisierung der Anzeige bei führt zu Interferenzen, die sich in Form von unerwarteten Zeigerausschlägen auf dem Display äussern.(Silvan Joos, 2025)

Eine grössere Skalierung des Dashboards führt zu längeren Reaktionszeiten aufgrund des Zeichnens einzelner Pixel. Dies führt zu spürbar verminderter Leistung. Zukünftig könnten auch weitere Menus oder Displays somit aufgrund der gestiegenen Komplexität zu einer Beeinträchtigung der Leistung führen. Dies fordert entweder bessere Hardware oder eine effizientere Signalverarbeitung.

Im Bezug auf die Kosten befindet sich das entwickelte Dashboard mit Materialkosten von etwa $20 eine kostengünstige Alternative zu kommerziellen Lösungen. Das preisgünstigste Angebot von MACH-13 liegt bei knapp $100.(MACH-13, o. J.)

# 4 Diskussion

Eine Analyse der Resultate zeigt, dass das entwickelte Dashboard die Rahmen-bedingungen dieses Projektes erfüllt. und eine kostengünstige Lösung für die Anzeige von Empfängerdaten in Modellbau-anwendungen bietet. Besonders hervorzuheben ist die Kosteneffizienz: Mit Materialkosten von etwa $20 liegt das Dashboard deutlich unter den Preisen der Mitbewerber, wie zum Beispiel MACH-13. Jedoch ist das Dashboard in der aktuellen Form noch sehr einfach und weist gewisse Einschränkungen auf. Das Display ist monochrom und hat aktuell nur eine Anzeige zur Auswahl (ohne Debug-Menu).

Eine erhöhte Wiederholrate des Displays zeigt sporadische Ausschläge vom Zeiger, was nicht nur störend ist, sondern auch die Genauigkeit vom Dashboard reduziert. Ursache dafür scheint die Taktfrequenz des Displays zu sein. Als temporäre Lösung dient eine künstliche Verlangsamung des Displays, was eine drastische Verbesserung zeigt, jedoch auch die Reaktionszeit beeinträchtigt. Filtern der Messdaten wäre ebenfalls ein Lösungsansatz. Eine genauere Analyse ist jedoch noch notwendig.

Weiterfahrend ist das Projekt ausbau-freundlich aufgebaut, sodass ein Wechsel auf andere Displays durchaus möglich wäre. Auch die Integration von zusätzlichen Sensoren oder seriellen Schnittstellen könnte die Unterstützung von Flugkontroller neuster Generationen sicherstellen. All diese Änderungen fordern jedoch eine umfassende Überarbeitung des bestehenden Projekts. Der QT Py könnte als zuverlässiger Mikrokontroller auch in zukünftigen Revisionen verwendet werden.

[2] Für eine Visualisierung siehe Abschnitt “Links”.

.

# Literatur


Adafruit. (o. J.). *Adafruit QT Py RP2040*. Abgerufen 10. Januar 2025, von https://www.adafruit.com/product/4900

*Bresenham's circle drawing algorithm*. (2024, Oktober). GeeksforGeeks. https://www.geeksforgeeks.org/bresenhams-circle-drawing-algorithm/

*CircuitPython*. (2025, Januar). https://docs.circuitpython.org/en/latest/README.html

CustomRaspi (Regisseur). (2021, Juni 10). *2.4 GHz Remote Controlled Raspberry Pi: PWM input via GPIO (realtime linux!)* [Video recording]. https://www.youtube.com/watch?v=bxdPWrEhbto

MACH-13. (o. J.). *Functional LED Displays*. Abgerufen 9. Januar 2025, von https://www.mach-13.com/en/displays-cockpits

*Pulse Width Modulation (PWM)*. (2024, Mai). GeeksforGeeks. https://www.geeksforgeeks.org/pulse-width-modulation-pwm/

*Seeed studio*. (o. J.). Abgerufen 10. Januar 2025, von https://www.seeedstudio.com/Grove-OLED-Display-0-66-SSD1306-v1-0-p-5096.html

Silvan Joos (Regisseur). (2025, Januar 10). *SimplePWMDash Demo* [Video recording]. https://www.youtube.com/watch?v=J47dhDQEdjE


# Links

GitHub-Link zum Projekt für weitere Einblicke, Anleitung zur Installation und Fotos: https://github.com/SilvanJoos/SimplePWMDash

Demo-Video zur Veranschaulichung der Funktionalität: https://youtu.be/J47dhDQEdjE?si=tdxn_z5K1aN5YqZO