Autopilot für Microcruiser RC Flugzeug

Schule:	**TEKO Technische Fachschule Bern**
Verfasser:	**Jarno Linder**
	[redacted]
Studiengang:	**Elektrotechnik Start 2022**
Eingereicht bei:	[redacted]
Abgabe:	**28. Oktober 2025**

Management Summary

Diese Projektarbeit befasst sich mit der Entwicklung, Integration und Erprobung eines autonomen Navigations- und Landessystems für ein ferngesteuertes Modellflugzeug (Microcruiser). Ziel war es, das *RC-Flugzeug* so auszurüsten, dass es selbstständig *GPS*-basierte Missionen abfliegen und präzise landen kann, ohne manuelle Eingriffe während kritischer Flugphasen.

Kern des Systems bildet der *Pixracer R15 Autopilot* mit der Firmware *ArduPlane*, ergänzt durch ein Raspberry Pi *Compute Module 4* zur Bildverarbeitung. Mit *GPS* (u-blox *NEO-M8N*), *LIDAR* (TF-Luna) und einer *Pi-Kamera* wurde ein System geschaffen, das autonome Navigation durch eine *Sensorfusion* ermöglicht.

Die Hauptziele umfassten: autonome Navigation per *GPS*, automatische Landung mittels *LIDAR* sowie *Failsafe*-Priorität für manuelle *RC-Steuerung*. Erweiterte Ziele waren die visuelle Landebahnerkennung mit *OpenCV* und die Kommunikation via *MAVLink*. Ergänzend erfolgte das *Logging* relevanter Flugdaten.

Nach umfangreicher Parametrierung des *Pixracers* und *Pythonprogrammierung* auf dem *CM4* wurden zunächst *Simulationen* mit *JSBSim* und eigenem aerodynamischen Modell durchgeführt. Diese bestätigten die Stabilität und Missionslogik. Folgende Realflugtests unter Sichtbedingungen zeigten ein stabiles Flugverhalten und erfolgreiche autonome Missionen. Beim Erststart kam es auf Grund von zu schwachem Abwurf zu einem Strömungsabriss und beim letzten Flug kam es infolge eines Spannungseinbruchs im Landeanflug zu einer Bruchlandung, was die Notwendigkeit einer verbesserten Vorflugkontrolle verdeutlichte.

Alle definierten *SMART-Ziele* wurden erreicht. Das Projekt belegt die technische Machbarkeit autonomer Flug- und Landefunktionen im RC-Modellflug und lieferte wertvolle Erkenntnisse für weiterführende Entwickliungen.
Empfohlen werden künftig eine *Telemetrieanbindung*, Checklisten sowie eine Trägerplattform mit höherer Nutzlast.

Inhalt

1. Glossar

Erstellt mit Hilfe von ChatGPT, Modell 5 (ChatGPT (GPT5) kein Datum).

ArduPilot

Open-Source-Flugsteuerungssoftware für autonome Flugzeuge, Multicopter und Fahrzeuge. Bietet umfangreiche Funktionen wie Missionsplanung, Stabilisierung und Failsafes.

Attitude Estimation

Lageschätzung eines Flugzeugs im Raum durch Kombination mehrerer Sensordaten (IMU, Magnetometer, GPS).

Autonomer Flug

Flugmodus, bei dem das UAV eine geplante Mission ohne direkte manuelle Eingriffe abfliegt.

Autopilot

Elektronisches System (z. B. Pixracer R15), das Flugbewegungen automatisch steuert und stabilisiert.

BAKOM

Bundesamt für Kommunikation, regelt die Nutzung von Funkfrequenzen in der Schweiz.

BAZL

Bundesamt für Zivilluftfahrt, zuständig für die Regulierung und Bewilligung von Drohnen- und Flugoperationen in der Schweiz.

Barometer (MS5611)

Sensor zur Messung des Luftdrucks; ermöglicht in Kombination mit GPS eine präzise Höhenbestimmung.

BEC (Battery Eliminator Circuit)

Spannungswandler, der Empfänger, Servos oder Controller mit stabiler Spannung versorgt.

BVLOS (Beyond Visual Line of Sight)

Flug ausserhalb der Sichtweite; in der Schweiz nur mit spezieller Bewilligung erlaubt.

Carrier Board

Trägerplatine, welche die Anschlüsse für das Raspberry Compute Module bereitstellt.

CE-Kennzeichnung

Europäische Konformitätsmarkierung, bestätigt die Einhaltung gesetzlicher Anforderungen.

CL / CD

Aerodynamische Kennwerte: CL = Auftriebsbeiwert, CD = Widerstandsbeiwert.

CM4 (Compute Module 4)

Kompakter Einplatinencomputer von Raspberry Pi, dient als Recheneinheit für Bildverarbeitung und Sensorfusion.

CSI-Port

Camera Serial Interface; Anschluss für Raspberry Pi Kameramodule.

DoF (Degrees of Freedom)

Freiheitsgrade eines Sensors. 6-DoF misst lineare Beschleunigungen und Drehraten, 9-DoF zusätzlich das Magnetfeld.

Drone / UAV

Unmanned Aerial Vehicle; unbemanntes Fluggerät, ferngesteuert oder autonom betrieben.

EDF (Electric Ducted Fan)

Impeller-Antriebssystem, das Schub durch einen elektrischen Gebläsekanal erzeugt.

ESC (Electronic Speed Controller)

Elektronischer Drehzahlregler für Elektromotoren, steuert Leistung und Drehrichtung.

eMMC

Integrierter Flash-Speicher des Compute Module 4, dient als Systemlaufwerk.

Failsafe

Sicherheitsfunktion, die bei Signalverlust oder Fehler automatisch eine definierte Aktion (z. B. Notlandung) ausführt.

FFC-Kabel

Flaches Verbindungskabel für Kameraanschlüsse und Signalleitungen.

Firmware

Eingebettete Software zur Steuerung der Hardware, z. B. ArduPlane auf dem Pixracer R15.

Flächenbelastung

Quotient aus Gewicht und Flügelfläche; beeinflusst die Fluggeschwindigkeit und das Stallverhalten.

Gantt-Diagramm

Diagramm zur Zeitplanung und Visualisierung von Arbeitspaketen und Meilensteinen.

GHz (Gigahertz)

Frequenzeinheit; RC-Systeme operieren typischerweise im 2.4 GHz-Band.

GNSS (Global Navigation Satellite System)

Sammelbegriff für globale Satellitennavigationssysteme (GPS, GLONASS, Galileo, BeiDou).

GPS (Global Positioning System)

Satellitengestütztes Navigationssystem zur Positionsbestimmung.

Ground Control Station (GCS)

Bodenstation zur Missionsplanung, Überwachung und Steuerung des UAV (z. B. QGroundControl).

IMU (Inertial Measurement Unit)

Sensor zur Erfassung von Beschleunigungen und Drehraten; Grundlage für Lageschätzung.

I²C / UART / CAN

Kommunikationsprotokolle für die Datenübertragung zwischen Sensoren und Controllern.

JSBSim

Open-Source-Flugsimulator für physikalisch basierte Simulationen von Flugzeugen.

Kamera (PiCam3)

12-Megapixel-Kameramodul für Raspberry Pi; erkennt Landebahnen und unterstützt visuelle Navigation.

k-Faktor

Induzierter Widerstandsbeiwertfaktor, beschreibt Effizienzverluste durch Auftriebserzeugung.

LIDAR (Light Detection and Ranging)

Sensor zur Distanzmessung mittels Laserimpulsen; dient zur präzisen Höhenmessung beim Landen.

LiPo-Akku (Lithium-Polymer-Akku)

Leichter, leistungsfähiger Energiespeicher; empfindlich gegen Tiefentladung und mechanische Beschädigung.

Logging

Aufzeichnung von Mess- und Flugdaten (z. B. GPS, LIDAR, Kamera) zur Analyse und Dokumentation.

Magnetometer (LIS3MDL)

Sensor zur Bestimmung der Flugrichtung (Heading) anhand des Erdmagnetfelds.

MAVLink

Kommunikationsprotokoll zwischen Flugcontroller (z. B. Pixracer) und externer Software (z. B. QGroundControl).

Mission Planning

Definition von Wegpunkten, Flugrouten und Landezielen für autonome Flüge.

MTOW (Maximum Take-Off Weight)

Maximales Startgewicht, das ein Flugzeug erreichen darf.

NEO-M8N (u-blox)

GNSS-Empfänger mit Mehrsystemfähigkeit (GPS, GLONASS, Galileo, BeiDou).

OpenCV

Open-Source-Bibliothek für Bildverarbeitung und Objekterkennung.

Oswald-Faktor (e)

Effizienzfaktor für die Auftriebsverteilung über die Spannweite; beeinflusst den induzierten Widerstand.

Parametrierung

Einstellung der Softwareparameter im Flugcontroller (z. B. ArduPlane-Parameter).

Pixracer R15

Flugcontroller (Autopilot), steuert Sensoren, Servos und Flugmodi; Basis für autonome Funktionen.

Python

Programmiersprache zur Entwicklung von Skripten, Logik und Bildverarbeitungsalgorithmen.

QGroundControl

Software zur Missionsplanung, Parametrierung und Live-Überwachung von UAVs.

Raspberry Pi OS

Offizielles Betriebssystem für Raspberry Pi Geräte, basiert auf Debian Linux.

RC (Remote Control)

Fernsteuerungssystem für Modellflugzeuge; ermöglicht manuelle Eingriffe.

Redundanz

Mehrfache Auslegung von Systemen oder Sensoren zur Erhöhung der Betriebssicherheit.

SBUS

Digitales RC-Kommunikationsprotokoll mit serieller Datenübertragung (z. B. von Spektrum oder Futaba).

Sensorfusion

Kombination mehrerer Sensordaten zur präzisen Zustands- und Lagebestimmung.

Simulation (SITL)

Software-in-the-Loop-Simulation zur Testung von Fluglogik ohne physisches UAV.

SMART-Ziele

Zieldefinition nach den Kriterien Spezifisch, Messbar, Erreichbar, Relevant, Zeitgebunden.

TF-Luna V1.2

LIDAR-Sensor zur präzisen Distanzmessung auf Basis des Time-of-Flight-Prinzips.

TOF (Time of Flight)

Messprinzip, bei dem die Laufzeit eines Lichtimpulses zur Distanzbestimmung genutzt wird.

Trimmung

Feineinstellung der Steuerflächen für stabilen Geradeausflug.

UART (Universal Asynchronous Receiver Transmitter)

Serielle Schnittstelle zur Datenübertragung zwischen Geräten.

UBEC (Universal Battery Eliminator Circuit)

Externer Spannungswandler zur stabilen Stromversorgung von Elektronikkomponenten.

UAV (Unmanned Aerial Vehicle)

Unbemanntes Luftfahrzeug; kann ferngesteuert oder autonom betrieben werden.

VLOS (Visual Line of Sight)

Flug innerhalb der Sichtweite; gesetzliche Anforderung im Modellflug.

Widerstandsbeiwert (CD)

Mass für den Luftwiderstand eines Körpers im Flug.

WLAN

Drahtlosnetzwerk zur Datenübertragung.

2. Einleitung

2.1 Ausgangslage und Problemstellung

Für das Projekt wurde ein Modellflugzeug gebaut, welches eine Cruise Missile nachbildet. Das Modell kann bei aerojtp (aerojtp.com kein Datum) erworben werden. Ziel war es zu untersuchen, ob es nicht möglich ist, das Flugzeug mittels Autopiloten automatisch steuern und landen zu lassen.
Das manuelle Steuern und das sichere Landen eines *Remote Controlled (RC)* Modellflugzeugs stellen hohe Anforderungen an die Reaktionsfähigkeit, Flugpraxis und Aufmerksamkeit des Piloten. Daraus ergibt sich die zentrale Fragestellung: Kann ein solches Modellflugzeug mithilfe autonomer Systeme in der Lage sein, *Global Positioning System (GPS)* Fixpunkte selbstständig abzufliegen und präzise zu landen, ohne direkten menschlichen Eingriff während der kritischen Flugphasen? Besondere Bedeutung kommt der automatisierten Landung zu. Herkömmliches *GPS* bietet für diesen Zweck meist nicht die notwendige Genauigkeit, um eine sichere und wiederholbare Landung zu gewährleisten. Deshalb soll eine erweiterte *Sensorfusion* bestehend aus *GPS*, *Light Detection and Ranging (LIDAR)* und *Kamera* zum Einsatz kommen. Diese Kombination soll sowohl die exakte Positionsbestimmung als auch die Erkennung der Landefläche in Echtzeit ermöglichen. Ziel ist es, eine robuste, autonome Erkennungstechnologie für Modellflugzeuge zu entwickeln, die auch unter unstabilen *GPS*-Bedingungen oder unebenen Landeflächen zuverlässig funktioniert.
Diese Problemstellung ist nicht nur aus persönlichem Interesse relevant, sondern spiegelt aktuelle Entwicklungen im Bereich autonomer Flugsysteme wider. Während autonome Drohnensysteme zunehmend zum Standard in Forschung und Industrie gehören, ist der Einsatz solcher Technologien im klassischen Modellflugbereich bislang kaum verbreitet. Die Herausforderung besteht daher darin, bestehende Ansätze aus der unbemannten Luftfahrt auf kleinere, leichtere und kostengünstige RC-Modelle zu übertragen und so auch im Hobbybereich neue Massstäbe in Zugänglichkeit zu setzen.

2.2 Zielsetzung

2.2.1 MUSS-Ziele:

1. Autonome Navigation per GPS mit ArduPilot (SMART)

Spezifisch: Das *Unmanned Aerial Vehice (UAV)* soll in der Lage sein, mithilfe des *Pixracer R15* und *GPS* mindestens zwei vorab definierte Wegpunkte automatisch anzufliegen.
Messbar: Erfolgreiches Anfliegen von 2 *GPS*-Wegpunkten während eines Flugtests.
Erreichbar: Mit der vorhandenen Hardware und *Ardupilot* vollständig realisierbar.
Relevant: Grundlage für autonome Flugnavigation und Voraussetzung für die Landefunktion.
Zeitgebunden: Umsetzung und Test bis spätestens 06.10.2025.

2. Autonome Landung über Lidar (SMART)

Spezifisch: Nach Erreichen des letzten *GPS*-Wegpunkts soll das *UAV* mithilfe des *LIDAR*-Sensors automatisch landen.
Messbar: Erfolgreiches Absinken aus einer Höhe >5m bis auf den Boden mit Erkennung <0.3 m Höhe.
Erreichbar: Durch Integration von *TF-Luna* und LAND-Modus in *ArduPilot* direkt möglich.
Relevant: Wesentlicher Teil für den vollständig autonomen Flug.
Zeitgebunden: Umsetzung inklusive Flugtest bis 13.10.2025.

3. Failsafe-Priorität für RC-Steuerung (SMART)

Spezifisch: Im Falle manueller *RC*-Eingaben muss das *UAV* sofort aus dem autonomen Modus in manuelle Steuerung wechseln.
Messbar: Umschaltung innerhalb von 1 Sekunde nach Empfang eines RC-Signals.
Erreichbar: Konfiguration via *ArduPilot* (Failsafe, RC_OVERRIDE) ist technisch vorgesehen.
Relevant: Zentrale Anforderung des *Bundesamtes für Zivilluftfahrt (BAZL)* und höchste Sicherheitspriorität.
Zeitgebunden: Bis spätestens 11.10.2025 vollständig implementiert und demonstriert.

2.2.2 SOLL-Ziele:

1. Visuelle Bahnidentifikation mit PiCam (SMART)

Spezifisch: Die Kamera erkennt die Landebahn durch Farb- und Kontrastanalyse im Kamerabild.
Messbar: Mindestens 80 % Erkennungsrate bei Testdurchläufen mit Bildmaterial.
Erreichbar: Mit *OpenCV2* auf dem *Compute Module 4 (CM4)* (Raspberry Pi CM4 2012) realistisch umsetzbar.
Relevant: Ergänzt die GPS-Navigation und erlaubt perspektivische Präzisierung.
Zeitgebunden: Prototyp bis 18.10.2025.

2. Interaktion Pi → Pixracer via MAVLink (SMART)

Spezifisch: Der Raspberry Pi soll bei Bahn- oder Landezielerkennung via *Micro Air Vehicle Link (MAVLink)* (MAVLink 2022) einen Steuerbefehl an *Pixracer R15* senden.
Messbar: Erfolgreiche Modusumschaltung (z. B. auf LAND) aus *Python* via *pymavlink*.
Erreichbar: Kommunikation via *Universal Asynchronous Receiver Transmitter (UART)* realistisch testbar.
Relevant: Ermöglicht automatisierte Übergabe von Kameralogik an den Flugcontroller.
Zeitgebunden: Implementierung bis 13.10.2025.

2.2.3 KANN-Ziele:

1. Logging von GPS, LIDAR und Kamera auf dem Pi (SMART)

Spezifisch: Das System soll während des Flugs GPS-Positionen, *LIDAR*-Höhenwerte und Kamerabilder aufzeichnen.
Messbar: Je ein Logfile und gegebenenfalls Videostream pro Flugmission mit Timestamps.
Erreichbar: *Python*-Skripte mit *OpenCV2* und Dateiausgabe auf dem Pi ermöglichen dies.
Relevant: Erleichtert Fehleranalyse und Projektdokumentation.
Zeitgebunden: Umsetzung testweise bis 20.10.2025.

2.3 Methodisches Vorgehen

Das Projekt wurde im Entwicklungsansatz iterativ behandelt. In einer ersten Phase erfolgte die theoretische Simulation und daraus die Analyse des Flugverhaltens mit JSBSim. Auf diesen Erkenntnissen wurde die Hardware integriert und die Parameter des Pixracer R15 angepasst. Nach diversen Groundtests und Software-Validierungen folgte Phase 2 mit schrittweisen Flugversuchen unter definierten Bedingungen. Die Ergebnisse der Daten aus den Tests wurden ausgewertet und dienten der Abschätzung von Verbesserungen der Parameter und der Pythonskriptbasis.

2.4 Zeitplanung

Die Zeitplanung wurde im Rahmen der Vorprojektphase umgesetzt, um einen konkreten Anhaltspunkt zu den benötigten Stunden zu haben.
Alle Arbeiten wurden in Vier Phasen unterteilt. Diese Phasen wiederum wurden in Arbeitspakete unterteilt. Diese wurden wiederum in einzelne Tasks unterteilt.

Es wurde ein Ganttdiagramm erstellt, welches die Arbeitspakete darstellt, sowie die Meilensteine enthält. Dieses Diagramm war darauf ausgelegt, bei einer normalen Arbeitsbelastung zu einem positiven Ergebnis zu gelangen.

Es gilt hier jedoch zu beachten, dass das Ganttdiagramm nur als Anhaltspunkt für die Meilensteine diente, da bewusst einige aufeinander aufbauende Arbeitspakete demselben Task zugeordnet werden können. Ausgeführt wurden die Arbeiten abhängig von der verfügbaren Freizeit, was zu variablen Arbeitszeiten geführt hat.

Im Vorfeld dieser Projektarbeit wurde ein *Pythonskript* erstellt, welches die aufgewendeten Stunden erfasst. So war zu jeder Zeit die Übersicht über die aufgewendeten Stunden je Task gegeben.

Eine Detaillierte Übersicht über alle Planungen ist im Anhang Kapitel D zu finden.

3. Einbau und Parametrierung / Programmierung des Systems

3.1 Vorgaben und Risiken

Bei der Realisierung dieses Projekts sind verschiedene Vorgaben einzuhalten und Risiken zu bedenken. Einerseits gibt es strenge Vorgaben der Regierung bezüglich Luftfahrt und andererseits sind die Projektziele zu erfüllen.
Dabei sind die verschiedenen Risiken abzuwägen und wenn möglich zu minimieren.
Aus diesem Grund, und um dem Leser die Nachverfolgbarkeit der Entscheidungsfindung zu erleichtern, wird in diesem Kapitel auf Vorgaben, sowie Risiken eingegangen und diese beleuchtet.

3.1.1 Vorgaben

Bundesamt für Zivilluftfahrt (BAZL):

Das Projekt fällt unter die offene Kategorie und dort in die A3 Unterkategorie. Die Vorgaben sind sehr detailliert auf der Homepage des Bundesamtes für Zivilluftfahrt zu finden. Hier sind nur die jeweils auf das Projekt direkt einflussnehmenden aufgeführt.

Die Regeln besagen folgendes:

1) Kennzeichnung Die Drohne muss über eine Conformité Européenne (CE)- Kennzeichnung verfügen oder eine Klassenmarkierung enthalten. Ist einer dieser Punkte nicht erfüllt, gelten die Regeln der Übergangskategorie.
2) Die Drohne muss auf dem Onlineportal registriert werden.
3) Wenn in dieser Kategorie geflogen werden will, muss ein Zertifikat über den Betrieb einer Drohne erworben werden.
4) Der Flug darf nur in Sichtweite, Visual Line Of Sight (VLOS) erfolgen.
5) Die Drohne darf maximal Fünfundzwanzig Kilogramm wiegen.
6) Die maximale Einsatzhöhe beträgt Hundertzwanzig Meter.
7) Die Funkanlagenregulierung des Bundesamtes für Kommunikation muss eingehalten werden.

(Bundesamt für Zivilluftfahrt kein Datum)

Bundesamt für Kommunikation (BAKOM):

Gemäss *BAKOM*, darf Modellflug nur in ein paar wenigen Frequenzbereichen stattfinden. Das wird in diesem Projekt jedoch nicht zu einem Problem führen, da nur Handelsübliche Geräte zur Fernsteuerung und als Empfangsanlage verwendet werden, welche auf dem 2.4 Gigahertz (GHz)-Band operieren.
(Bundesamt für Kommunikation kein Datum)

Zielvorgaben der Projektarbeit:

Die detaillierten Projektziele sind in Kapitel 2.2 beschrieben.
Jedoch soll gezeigt werden, welche Ziele erfüllt wurden. Die Details sind in den folgenden Kapiteln zu finden.

3.1.2 Risiken

1. Einleitung

Diese Risikoanalyse bezieht sich auf das autonome Flugprojekt auf Basis eines MicroCruiser-Flugzeugs, gesteuert durch einen Pixhawk R15 Flugcontroller und einem Raspberry Pi CM4 für Vision und *Logging*. Besondere Beachtung finden rechtliche Aspekte gemäss den Richtlinien des *BAZL*, sicherheitstechnische Risiken sowie technische Schwachstellen.

2. Regulatorische Risiken (gemäss BAZL)

Tabelle 1 Regulatorische Risiken

Risiko	Beschreibung	Auswirkung	Gegenmassnahme
Sichtkontakt	Gemäss BAZL muss der Pilot jederzeit direkten Sichtkontakt zum Flugobjekt haben.	Hohe rechtliche Relevanz, Projekt könnte als Beyond Visual Line of Sight (BVLOS) gelten.	Flug nur in Sichtweite, Helfer mit Fernglas als visuelle Unterstützung.
Autonomer Betrieb	Autonomes Fliegen ohne sofortige manuelle Übernahme ist verboten.	Illegaler Betrieb, Gefahr für Bewilligung.	Priorisierung der RC-Steuerung sicherstellen. Jeder RC-Befehl muss Autopilot sofort überschreiben.
Gewicht & Flughöhe	Drohnen über 500 g unterliegen strikteren Regeln.	Registrierung, Versicherungspflicht, Flugverbotszonen.	Registrierung und Versicherung einhalten, Flugzonen lokal prüfen.

3. Technische Risiken

Tabelle 2 Technische Risiken

Risiko	Beschreibung	Auswirkung	Gegenmassnahme
Verlust von GPS	Signalverlust durch Abschattung oder Störungen	Navigationsverlust, mögliche Desorientierung	Fail-Safe Routine, visuelle Navigation unterstützen
Kamerafehler / Latenz	Verzögerte oder fehlerhafte Bildverarbeitung auf dem Pi	Fehlerhafte Landung oder Kursabweichung	Simpler visueller Algorithmus mit Schwellenwerten, Logging zur Analyse
Verbindungsverlust RC	Kein Empfang durch Entfernung oder Interferenzen	Verlust der Steuerung	Failsafe-Signal am Pixhawk R15 aktivieren (z. B. automatisches Kreisfliegen, Notlandung)
Stromversorgung instabil	CM4 oder Sensoren fallen bei Spannungseinbruch aus	Teilweiser Funktionsverlust	Universal Battery Eliminator Circuit (UBEC) mit ausreichender Leistung und Kondensatoren verwenden

4. Sicherheitsrisiken für Personen und Umwelt

Tabelle 3 Sicherheitsrisiken Personen / Umwelt

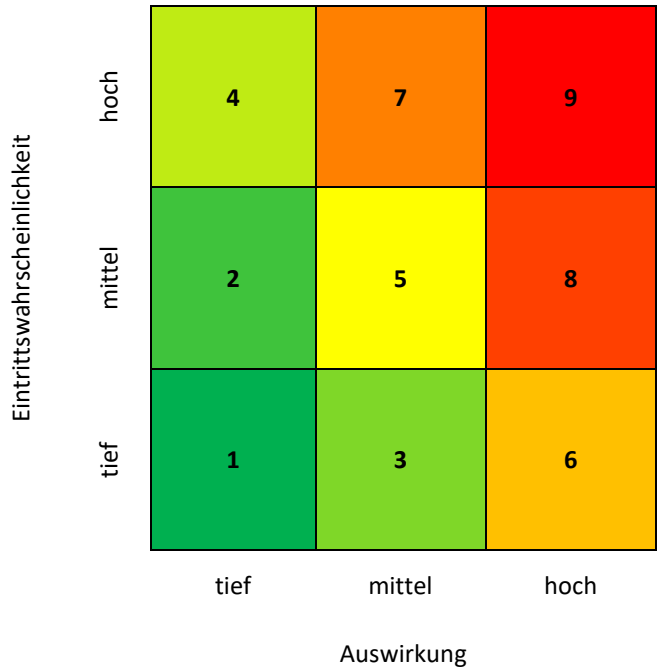
Risiko	Beschreibung	Auswirkung	Gegenmassnahme
Unkontrollierte Landung	Bei Systemfehlern / Autopilotversagen	Verletzungsgefahr, Sachschäden	Autopilot nur über leerem Feld, Mindestabstand zu Menschen einhalten
Mechanischer Defekt (Servo/Flügel)	Materialermüdung, schlechte Wartung	Absturz	Vorflugcheck, präventiver Austausch kritischer Teile
Brandgefahr LiPo-Akku	Kurzschluss, mechanischer Schaden	Feuer an Bord / nach Absturz	Sicherung des Akkus, Verwendung von Akku-Taschen und Soft-Mounts

5. Bewertung der Risiken (qualitativ)

Tabelle 4 Risikobewertung

Risiko	Eintrittswahrscheinlichkeit	Auswirkung	Gesamtrisiko	Indexierungsnummer Matrix
Verlust von Sichtkontakt	Mittel	Hoch	**Hoch**	9
Autonomer Betrieb ohne RC-Kontrolle	Niedrig	Hoch	**Mittel**	6
GPS-Ausfall	Niedrig	Mittel	**Tief**	3
Kamerafehler / Verarbeitungsverzögerung	Mittel	Niedrig	**Tief**	2
Stromausfall am Raspberry Pi	Niedrig	Hoch	**Mittel**	6
Verlust RC-Verbindung	Mittel	Hoch	**Hoch**	8
Unkontrollierte Landung	Niedrig	Hoch	**Mittel**	6
Mechanischer Defekt	Niedrig	Hoch	**Mittel**	6

Tabelle 5 Risikomatrix



6. Massnahmen zur Risikominimierung

- Manuelle Steuerung immer priorisieren, z. B. durch Überwachung der RC-Eingänge per Firmware / Lua-Skript
- Pixhawk-*Failsafes* aktivieren (Loss of RC, Low Battery, GPS Loss)
- Nur in Sichtweite operieren, keine Hindernisse oder bewohnte Gebiete überfliegen
- *Redundanz* bei Stromversorgung (*UBEC* + ggf. Kondensatoren)
- Flug nur bei stabilem Wetter und guter Sicht
- Gewichtsgrenze von vierhundert Gramm einhalten, was auch dem *Maximum Takeoff Weight (MTOW)* des *UAV* entspricht.

7. Fazit

Bei Einhaltung der *BAZL*-Richtlinien, solider technischer Umsetzung und geeigneter Flugumgebung ist das Projekt aus regulatorischer und technischer Sicht realisierbar. Der Betrieb muss jederzeit manuell übersteuerbar bleiben und visuell überwacht werden.

3.2 Gesamtaufbau des Systems

Das Gesamtsystem besteht aus mehreren modular aufgebauten Teilsystemen, die über standardisierte Schnittstellen miteinander verbunden sind. Abbildung 1 (Blockdiagramm) zeigt den logischen Aufbau sowie den Signalfluss zwischen den Sensoren, dem Flugcontroller und den Steuerungselementen.

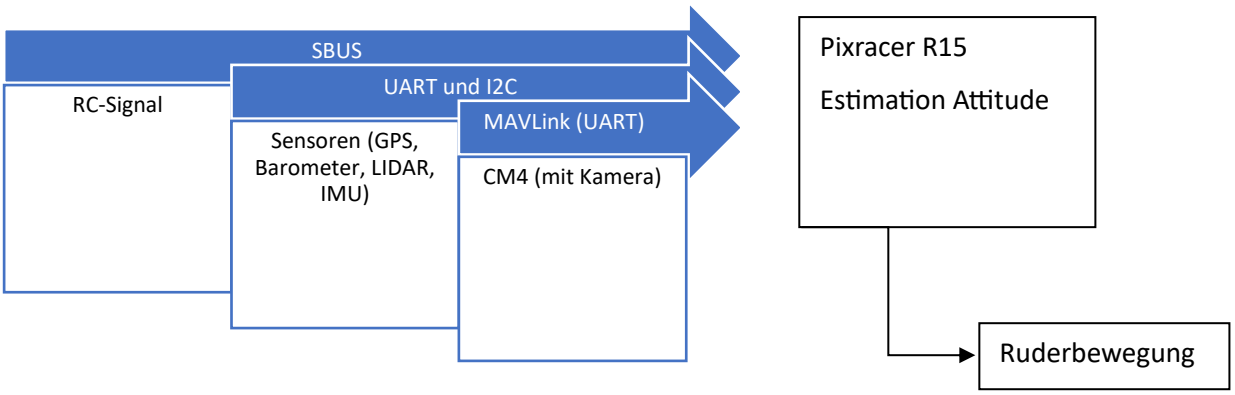


Abbildung 1 Blockdiagramm

3.3 Pixracer R15

Für dieses Projekt wurde der *Pixracer R15* eingesetzt, Nachbauende müssen jedoch auf den Pixracer Pro zurückgreifen, da der R15 nicht mehr produziert wird

Der *Pixracer R15* (Pixracer 2023) ist das zentrale Steuerelement des Projekts. Als Autopilot bildet er die Grundlage für die autonomen Flug- und Lande-Funktionen und trägt massgeblich zur Umsetzung aller MUSS-, sowie Teilen der SOLL- und KANN-Ziele bei.

Der *Pixracer R15* wurde gewählt, weil das Gerät bereits vorrätig war und er alle Aspekte, die gefordert sind, abdeckt. Ausserdem ist das Modul mit seiner kompakten Bauform und dem geringen Gewicht gut geeignet bei einem Leichtbau UAV. Die Technischen Daten sind auch im Anhang unter Kapitel D zu finden.
(Pixracer 2023)

Abbildung 2 Pixracer R15 (nach Aliexpress Pixracer kein Datum)

Die verwendete Firmware ist *ArduPlane* (ArduPilot, 2024).

3.3.1 Übersicht

Wie die verschiedenen Sensoren mit dem Pixracer interagieren, ist in Abbildung 2 (Diagramm Interaktionen) zu sehen. Das Diagramm wurde mit Mermaid (Meramid live editor kein Datum) erstellt.

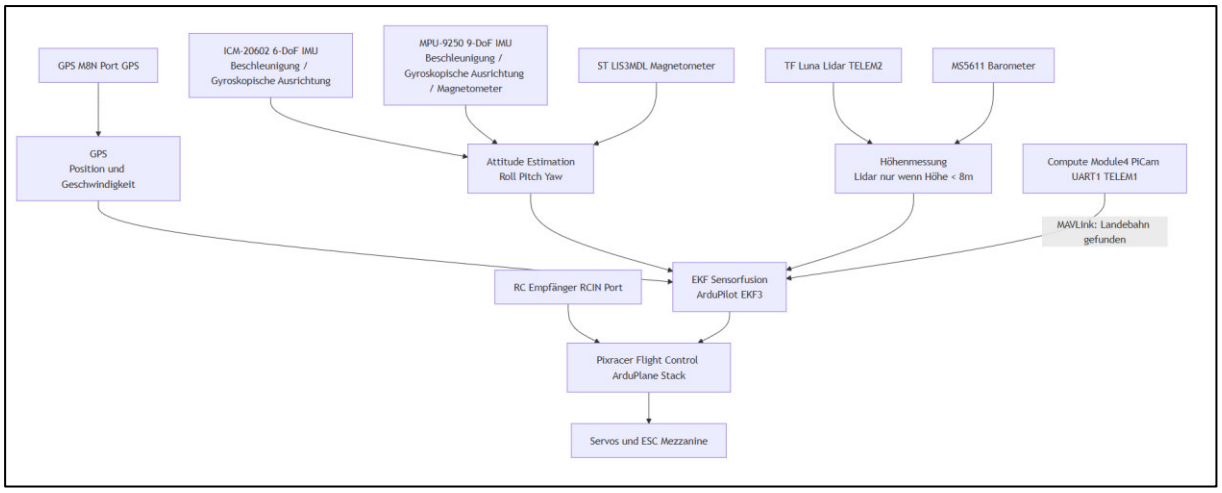


Abbildung 3 Diagramm Interaktion (Meramid live editor kein Datum)

Auf die jeweiligen Sensoren wird im nächsten Kapitel eingegangen.

3.3.2 Sensoren

Der *Pixracer R15* und seine angeschlossenen Sensoren tragen auf unterschiedliche Weise zur Erreichung der definierten MUSS-, SOLL- und KANN-Ziele bei.
Für die MUSS-Ziele der autonomen Navigation spielen insbesondere das *GPS*-Modul (*u-blox NEO-M8N*) (holybro GPS kein Datum), die integrierten *Magnetometer* (im *GPS* sowie im *ST LIS3MDL*), die redundanten IMUs (ICM-20602 und MPU-9250) (Pixracer 2023), das *LIDAR* (TF-Luna V1.2) (TF-Luna LIDAR kein Datum) sowie das barometrische Druckmodul (MS5611) (Pixracer 2023) eine zentrale Rolle. Diese Komponenten liefern die erforderlichen Daten für die präzise Positionsbestimmung, Kursstabilisierung, Lageschätzung und Höhenkontrolle.
Die autonome Landung wird in erster Linie durch den eingesetzten *LIDAR*-Sensor (TF-Luna V1.2) (TF-Luna LIDAR kein Datum) ermöglicht, der in Bodennähe exakte Höheninformationen bereitstellt. Das *Barometer* liefert dabei ergänzend eine grobe Höhenreferenz, sodass eine Kombination beider Sensoren einen zuverlässigen und sicheren Landeanflug erlaubt.
Für das dritte MUSS-Ziel, die *Failsafe*-Priorität der RC-Steuerung, sind die RC-Receiver-Schnittstellen (in meinem Falls *SBUS*) sowie die *Pixracer R15*-Firmware entscheidend. Sie gewährleisten, dass bei manuellen Steuerbefehlen der Autopilot unverzüglich überschrieben wird.
Die SOLL-Ziele werden massgeblich durch die Interaktion mit dem Raspberry Pi CM4 erreicht. Die Kameraeinheit des CM4 übernimmt die visuelle Erkennung der Landebahn. Über die *UART*-Schnittstellen des *Pixracer R15* erfolgt eine *MAVLink*-Kommunikation, sodass das CM4 die identifizierte Bahn- oder Landepunkte direkt an den Flugcontroller übermitteln kann. Dies ermöglicht eine automatisierte Übergabe von Bildinformationen an den Autopiloten und unterstützt eine präzisere Navigation.
Die KANN-Ziele betreffen vor allem die Datenaufzeichnung und Erweiterung der Funktionalität. Die Secure Digital (SD)-Karte des *Pixracer R15* dient der Speicherung von Logfiles, die GPS-, IMU- und *LIDAR*-Messwerte enthalten. Der Raspberry Pi zeichnet darüber hinaus zusätzlich Kamerabilder auf. Diese Aufzeichnungen sind insbesondere für die Projektdokumentation und Fehleranalyse von Bedeutung.

LIDAR:

Entgegen der Projekteingabe wurde im Verlauf der Vorprojektphase gegen einen Radio Detection and Ranging (Radar) Sensor entschieden. Da das UAV nur bei Sichtwetterbedingungen geflogen werden darf und der Abstandsensor zur Landung genutzt werden soll, hat der *LIDAR* mit der hohen Genauigkeit gegenüber einem Radar, der bei schlechten Sichtbedingungen wie zum Beispiel Nebel Verwendung findet, für mich bei der Entscheidungsfindung Vorrang erhalten. Ebenso hat der Umstand, dass es viele *LIDAR* auf dem Markt der Ferngesteuerten Drohnen gibt, welche mittels diverser Bussysteme, unter anderen *I^2C*, *UART* und *CAN*, angebunden werden können dazu geführt. Diese Sensoren sind allesamt einfach im Verbund mit dem Pixracer, wenn mit der entsprechenden Software geflasht, zu verwenden.

Beim eingesetzten *LIDAR* handelt es sich um ein TF-Luna V1.2 single point ranging Sensor (TF-Luna LIDAR kein Datum). Dieser Sensor wurde gewählt, da der Preis moderat ist und die *Arduplane*-Software diesen Sensor bereits kennt, und so nur die Parameter zur Kommunikation und diejenigen zur Verwendung entsprechend angepasst werden müssen.

Abbildung 4 TF- Luna LIDAR (TF-Luna LIDAR kein Datum)

Weitere technische Daten sind im Anhang unter Kapitel D zu finden.

GPS:

Bei dem GPS- Empfänger, der verwendet wurde, handelt es sich um einen *u-blox NEO-M8N GNSS* (holybro GPS kein Datum). Dieser Sensor war im Lieferumfang des Pixracers, weshalb die Wahl auf diesen Sensor fiel.

Was den M8N-Empfänger auszeichnet sind seine *Global Navigation Satellite System (GNSS)* Fähigkeit. Er kann gleichzeitig Signale von mehreren globalen Navigationssatellitensystemen (*GNSS*) empfangen und verarbeiten, darunter *GPS* (USA), *GLONASS* (Russland), *BeiDou* (China) und *Galileo* (Europa). Die meisten *M8N-Module* kommunizieren über gängige Schnittstellen wie *UART* oder I^2C, was die Integration in Flugsteuerungen wie den *Pixracer R15* sehr einfach macht.
Ausserdem zu erwähnen ist, dass der *M8N* ein integriertes *Magnetometer* hat, welches als Backup für den fest integrierten Sensor des *Pixracer R15* dienen kann.
(holybro GPS kein Datum)

Abbildung 5 GPS M8N (nach holybro GPS kein Datum)

Weitere Sensoren:

Der *Pixracer R15* hat mehrere fest verbaute Sensoren. Diese sind:

Invensense/TDK ICM-20602 (6-DoF IMU) Dieser Sensor ist eine Kombination aus einem 3-Achsen-Beschleunigungsmesser und einem 3-Achsen-Gyroskop, wobei hier das 6-*Degrees of Freedom (DoF)* IMU für einen Sensor mit 6 Freiheitgraden steht. Das bedeutet er hat in 3 Achsen eine Beschleunigungsmessung in m/s^2 (X, Y, Z) und erkennt so lineare Bewegungen beziehungsweise Neigungen. Und in 3 Achsen eine Messung der Drehraten in °/s (Gyroskop) was als Rollen, Pitchen und Gieren erkannt wird, also die Drehung um die jeweiligen Flugzeugachsen. Diese Infos gehen ins *Attitude Estimation* (Lageschätzung), das der Pixracer für Stabilisierung, Autopilot-Funktionen und *Sensorfusion* (mit *GPS*, *Barometer*, *Magnetometer*) nutzt.

Beim ***MPU-9250 (9-DoF-IMU)*** handelt es sich um einen Kombinierten Sensor wie beim oben genannten Invensense/TDK ICM-20602 (6-*DoF* IMU). Er hat aber zusätzlich zu den Beschleunigungsmessern und den Gyroskopen noch ein *Magnetometer* integriert. Auch dieser Sensor liefert seine Daten direkt in die *Attitude Estimation* um bei Störungen oder Ausfällen als Backup zu dienen. Die Grösse der *Sensorfusion* kann hierbei in der Software beliebig selber eingestellt und genutzt werden.

Der ***MEAS MS5611*** ist ein extrem präziser barometrischer Drucksensor, der für die Höhenmessung verwendet wird. Dabei wird der aktuelle Luftdruck mit dem beim Start gespeicherten Wert verglichen, um Rückschlüsse auf die Höhe zu ziehen. Zusammen mit dem *GPS* kann so eine sehr präzise Höhenmessung gemacht werden. Ausserdem hilft er gegen Höhendrift bei *GPS*-Schwankungen.

Und der ***ST LIS3MDL*****:** ist ein zusätzlicher 3-Achsen-*Magnetometer*, der für eine Kompassfunktion sorgt. Mit diesem Sensor wird das heading des *UAV* bestimmt, da *GPS* zeigt, wo man ist, aber nicht in welche Richtung man schaut, bezogen auf das Erdmagnetfeld. Ausserdem hilft dieser Sensor die Drift des Gyroskops zu kompensieren.

Ausserdem ist ein ***ESP8266-Modul*** verbaut, um per *Wireless Local Area Network (WLAN)* zu kommunizieren. Das ist zwar nicht ein Sensor im üblichen Sinne, aber spielt im Projekt eine Rolle. Die Sensoren können je nach Baujahr und Serie andere Typen sein, haben aber immer dieselbe Funktion.
(Pixracer 2023)

3.4 System Raspberry Compute Modul 4

Hier geht es um das Zusammenspiel von *CM4* und Kamera, sowie der Kommunikation mit dem *Pixracer R15*.
Die Wahl fiel auf dieses System, weil einerseits einfacher Zugang dazu vorlag, es war bei mir noch vorrätig, und andererseits, weil der Formfaktor perfekt in den Rumpf des *UAV* passt. Ausserdem sind die vier GB RAM auf dem Board genug, um auch anspruchsvolle Videobearbeitungen zu machen.
Im Zuge der Vorprojektphase wurden auch andere Systeme evaluiert. Diese waren das Raspberry Pi 5, welches aber zu gross ist. Oder das NVIDIA Jetson, aber das ist ebenfalls zu gross und das Gewicht ist viel zu hoch. Auch einfachere Mikrokontroller wie das ESP32 wurden evaluiert, aber wegen der fehlenden Möglichkeit der Videobearbeitung und des kleinen Arbeitsspeichers wurden diese verworfen.

3.4.1 Compute Module 4

Das Raspberry *CM4* ist ein kompakter Einplatinencomputer, der auf der gleichen Architektur wie das Raspberry Pi 4B basiert. Herzstück ist ein Broadcom BCM2711 Quad-Core Cortex-A72 Prozessor mit einer Taktfrequenz von 1,5 *GHz* (Raspberry Pi CM4 2012). Ergänzt wird dieser durch 4 GB LPDDR4-3200 SDRAM, wodurch auch rechenintensive Aufgaben wie Bildverarbeitung oder *Logging* zuverlässig ausgeführt werden können.
Das Modul ist wahlweise mit eMMC-Speicher oder nur als Lite-Version erhältlich. Im Projekt wird die eMMC- Variante verwendet.
Die Abmessungen betragen 55 mm × 40 mm bei einem Gewicht von nur 46 g, was das *CM4* besonders für Anwendungen in Gewichtsoptimierten *UAV*s geeignet macht.
Es ist zu beachten, dass die Schnittstellen nur genutzt werden können, wenn ein entsprechendes Carrier- Board genutzt wird.
(Raspberry Pi CM4 2012)

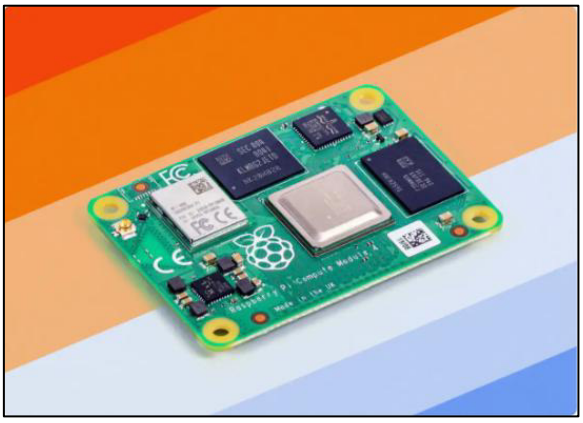


Abbildung 6 CM4 (nach Raspberry Pi CM4 2012)

3.4.2 Carrier Board

Das Board wird benötigt, weil das *CM4* selbst ausser den High-Density Board-to-Board Conectors keine Anschlüsse hat. Wenn man also eine Kamera oder ähnliches nutzen will, muss man die Anschlüsse erst zur Verfügung stellen.

In diesem Projekt wurde das Carier Board NANO-A von waveshare (CM4-NANO-A kein Datum) verwendet, weil dieses Board das kleinste vom Formfaktor her ist das gefunden wurde und es in das *UAV* passt. Ausserdem benötigt das Projekt auf der *CM4* Seite nur einen Kameraanschluss und die Möglichkeit das *CM4* zu flashen, sowie die Möglichkeit die *UART*-Pins zu nutzen, was alles durch das NANO-A abgedeckt ist.
Um das Board zu montieren, muss das *CM4* auf die passenden Steckverbinder gesteckt werden.

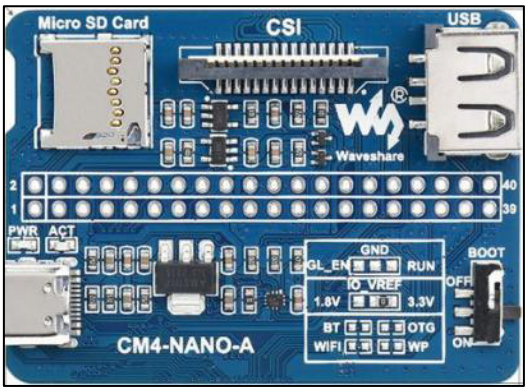


Abbildung 7 Carrier Board (nach CM4-NANO-A kein Datum)

3.4.3 Kamera

Die *Raspberry Pi Camera Module 3* (Raspberry Pi Picam3 2012) ist ein kompaktes CMOS-Kameramodul mit einer Auflösung von zwölf Megapixeln. Der Anschluss an das *CM4* erfolgt über ein fünfzehnpoliges Flat Flexible Cable (FFC) an den Mobile Industry Processor Interface (MIPI) Camera Serial Interface (CSI)-2 Port (Raspberry Pi Picam3 2012).
Das Kameramodul ist in verschiedenen Ausführungen erhältlich. In diesem Projekt wird die Version mit einem Öffnungswinkel von 75 Grad verwendet. Aufgrund des geringen Gewichts, der kleinen Bauform und der vollständigen Kompatibilität mit dem *CM4* ist die *Picam3* für den Einsatz in *UAV*s besonders geeignet.

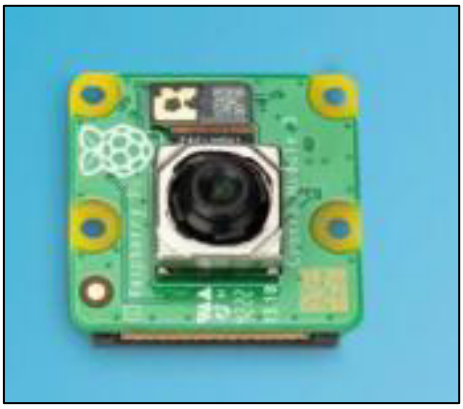


Abbildung 8 Picam3 (nach Raspberry Pi Picam3 2012)

3.5 Softwarecodes und Parametrierungen

In diesem Kapitel ist die Parametrierung des *Pixracer R15* und die Programmierung auf Basis von *Python* des *CM4* zu finden.

3.5.1 Parametrierung des Pixracer R15 via QroundControl

Bevor der *Pixracer R15* verwendet werden kann, muss er zunächst mit einer geeigneten Firmware versehen werden. In dieser Konfiguration wurde die Firmware *ArduPlane* (ArduPilot, 2024) installiert. *ArduPlane* bietet eine umfangreiche Softwarebasis, die sämtliche erforderlichen Parameter und Funktionen zur sicheren und zuverlässigen Steuerung eines Flächenflugzeugs bereitstellt. Das Aufspielen der Firmware, via *QGroundControl* (QGroundControl 2019), ist im Anhang Kapitel B beschrieben.

Nachdem die Firmware aufgespielt ist, können die Parameter per USB- Verbindung in *QGroundControl* mittels Mausklick eingestellt werden. Es gibt aber auch die Möglichkeit, eine vorhandene Parameterdatei direkt zu laden.

Um jegliche Vorgaben einzuhalten sind nur wenige Parameter einzustellen, jedoch reicht das nicht um das *UAV* zu fliegen. Im Anhang Kapitel B ist eine detaillierte Anleitung zur Parametrierung zu finden.

Die Parameter, welche gefordert sind, lauten:

Tabelle 6 wichtige Parameter

Parameter	Wert	Beschreibung
STICK_MIXING	1	RC-Eingaben werden in allen Modi berücksichtigt
FS_LONG_ACTN	0	Keine Aktion bei langanhaltendem Failsafe
FS_GCS_ENABLE	0	Kein Failsafe bei Verlust der GCS-Verbindung
FS_RC_ENABLE	1	RC-Failsafe aktiv, RC-Signal hat Vorrang

Wenn diese Parameter so eingestellt sind, hat in jedem Fall, solange ein *RC*- Signal vorhanden ist, dieses Vorrang. Es ist auch von Vorteil, sich einen Schalter zur Arming- und Disarming- Funktion zu

programmieren, um das *UAV* per Fernsteuerung bequem aktivieren beziehungsweise deaktivieren zu können. (ArduPilot, 2024)

3.5.2 Programmieren des Compute Module 4

Das *CM4* muss mit einem Betriebssystem versehen werden. In diesem Projekt wurde das offizielle 64bit Desktop Operating System (OS), basierend auf Debian 12, von Raspberry Pi (raspberrypi OS kein Datum) vom 13. Mai 2025 verwendet. Eine detaillierte Anleitung zur Inbetriebnahme und Installation der benötigten Abhängigkeiten ist im Anhang Kapitel B zu finden.

Um das Image zu laden, wird der offizielle Raspberry Pi Imager (Raspberry Pi Imager kein Datum) verwendet.

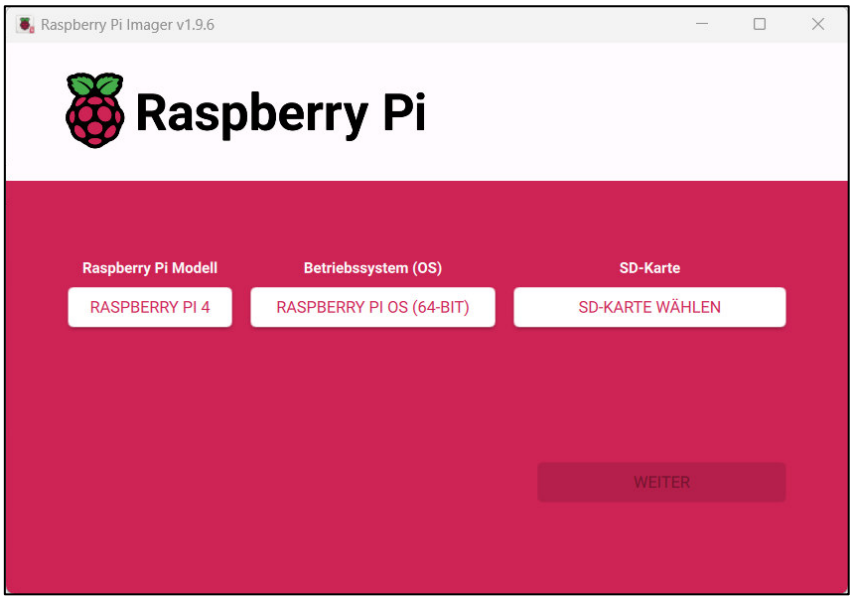


Abbildung 9 Raspberry Pi Imager (eigener Screenshot)

Ausserdem muss ein Tool verwendet werden, um das *CM4* als USB-Drive an einem PC zu betreiben. Das wird benutzt, um das OS zu laden. Bei diesem Projekt wurde der rpiboot (Github usbboot 2024) verwendet.

Sobald der *CM4* ein funktionierendes OS aufgespielt hat, kann das *Python*- Programm, zu finden im Anhang Kapitel B, auf dem *CM4* gespeichert werden. Dieses Skript wurde vollständig eigenständig

entwickelt. Für die Erzeugung eines leeren *Python*-Grundgerüsts (Klassenskelett, Funktionsrahmen) wurde ChatGPT (ChatGPT (GPT5) kein Datum) als Hilfsmittel verwendet. Alle Implementierungen, Kommentare und fachlichen Inhalte stammen vom Autor.

Es wurde *Python* gewählt, da dort alle Funktionen, die gesucht waren, als Bibliotheken vorhanden sind. Ausserdem muss das Programm nicht auf Geschwindigkeit getrimmt sein, da es keine Zeitkritischen Tasks benötigt. Das komplette Skript ist im Anhang Kapitel B zu finden.
Um das Programm zu betreiben, muss eine virtuelle Umgebung erzeugt werden, um dort die erforderlichen Bibliotheken installieren zu können. Die benötigten Bibliotheken sind in Tabelle 7 (Bibliotheken Python 3) aufgelistet.

Tabelle 7 Bibliotheken Python 3

Import / Komponente	Paket / Quelle	Hinweis
os, time, math, datetime	Stdlib	Teil der Python-Standardbibliothek
numpy as np	numpy	apt auf dem Pi meist schneller/stabiler
cv2	opencv	bringt Video-I/O/Codecs sauber mit
pymavlink	pymavlink	typischerweise nicht über apt verfügbar
picamera2	python3-picamera2	offizieller Raspberry-Pi-Kamerastack
libcamera.controls	python3-libcamera	kommt mit Picamera2/libcamera
ffmpeg	Systemtool	Codecs & Container für Video
v4l-utils	Systemtool	Video4Linux Tools, nützlich für Kamera-Debug
python3-venv, python3-pip	Systemtools	stellt venv & pip bereit

Wenn alles installiert ist, kann eine Systemd- Datei erstellt werden, um das Programm bei jedem Systemstart automatisch starten zu lassen.

3.6 Fernsteuerung und Empfänger

Bei der Fernsteuerung nutze ich meine *Spektrum iX12* Anlage als Sender mit einem SPM4649T Empfänger im *UAV*. Ich will hier nicht sehr tief auf diese Geräte eingehen, da jeder seine eigene Vorliebe im Bereich *RC*-Anlagen hat. Die Integration einer X- Beliebigen Anlage, ist mit dem Pixracer R15 (Pixracer 2023) kein Problem. Dementsprechend können die Vorlieben der Nutzer, im Bereich der RC- Anlagen, gewährleistet werden.

Grundsätzlich gilt für alle Sender, dass ein Modell im Speicher erstellt werden muss, welches als Flächenmodell ausgelegt ist. Um die *ArduPlane* Software im *Pixracer R15* richtig zu nutzen, muss unbedingt darauf geachtet werden, dass jegliche Trimmwerte auf null stehen und alle Servowege zu hundert Prozent nutzbar sind. Alle *Trimmungen* und Einstellungen der Servos im *UAV* erfolgen über die Parameter der *ArduPlane* Software.

Bevor der Empfänger im *UAV* verbaut wird, empfiehlt es sich, diesen an den Sender zu Binden.

Beim Einbau im *UAV* darauf achten, dass der Empfänger so weit wie möglich von den Stromführenden Leitern der Motorsteuerung entfernt ist, da diese mit Hochfrequenz EMV- technisch belastet ist.

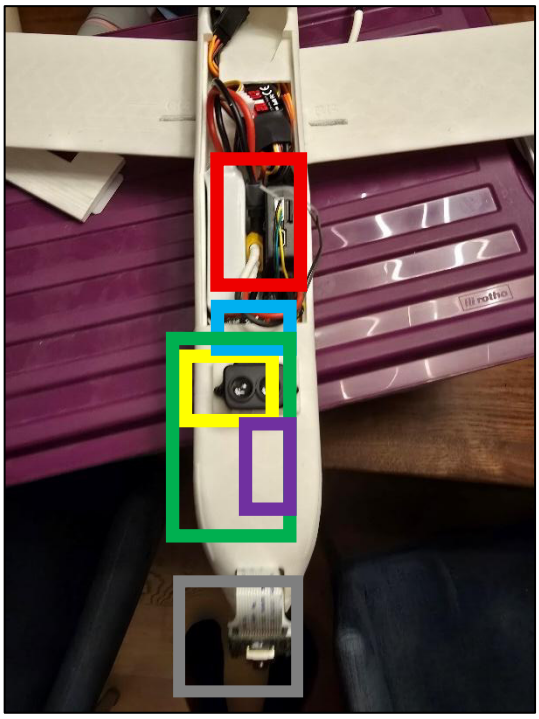
Eine Detaillierte Anleitung zu Vorbereitung, Einbau und den verwendeten Protokollen ist im Anhang Kapitel C zu finden.

3.7 Einbau und Verkabelung des Systems

In diesem Abschnitt werden nur explizit die Ausrüstungsgegenstände, welche sich vom originalen Micro-Cruiser unterscheiden, erwähnt. Alles, was gemäss Manual des Micro- Cruiser verbaut und vorgegeben ist, wird hier aus Urheberrechtlichen Gründen nicht weiter ausgeführt.
Der Einbau der Komponenten muss gemäss Platz und gewählter Ausrüstung erfolgen.
Um das *UAV* so zu bauen, wie in diesem Bericht, müssen die Komponenten gemäss Abbildungen 9 (UAV Topansicht) und 10 (UAV Einbauten) verbaut werden.

Abbildung 10 UAV Topansicht (eigenes Foto)

Hier ist das *M8N– GPS* zu sehen, welches eine Anschlusskabelverlängerung benötigt. Der *GPS*-Empfänger wird auch als Trimmgewicht genutzt, um den Schwerpunkt des *UAV* einzustellen.



Rot: *Pixracer R15*

Blau: RC-Empfänger

Grün: *CM4*

Violett: *UBEC*

Gelb: *LIDAR*

Grau: *Kamera*

Abbildung 11 UAV Einbauten (eigenes Foto)

Im Anhang Kapitel C ist ein vergrössertes Bild zu sehen.

Mit Ausnahme des *Pixracer R15*, des *LIDAR* und der Kamera hat kein Bauteil eine feste Einbaurichtung.

Die Einbaurichtung des *Pixracer R15* ist vorgegeben, da in der Missionsplanungssoftware (in der auch die Sensoren kalibriert werden) nur Ausrichtungen mit 90°-Winkeln in allen Achsen unterstützt werden. Daher ist diese Orientierung zwingend einzuhalten.
Das *LIDAR* muss nach unten ausgerichtet sein, um die Höhe zu messen und die Kamera sollte in einem 45 Grad Winkel nach vorne ausgerichtet sein.
(Pixracer 2023)

Es empfiehlt sich, vor dem Einbau der Komponenten alles zu verdrahten und auf Funktion zu kontrollieren. Das zugehörige Verdrahtungsschema ist im Anhang Kapitel C enthalten. Dieses wurde mit TinyCAD erstellt (Sourceforge 2002).

Der eingesetzte Akku hat eine Kapazität von 1.1 Ah.

Das *UAV* hat im Betrieb, mit Vollgas auf dem Antrieb und Servos mit Vollausschlägen 18.5A, gemäss Messung des *Pixracer R15*, an Strom benötigt, was jedoch nur bei Steigflugphasen stimmt, da zum horizontalen Flug nur circa 70 Prozent des Schubs und bei Sinkflugphasen nur circa 40 Prozent Schub benötigt werden. Aus meiner Erfahrung wurde mit durchschnittlich 70 Prozent gerechnet. Der *CM4* ist für 2.2A davon verantwortlich, sowie der *Pixracer R15* für 1.8A. Ausserdem müssen, um den Akku nicht zu beschädigen, rund 30 Prozent der Ladung verbleiben, damit ein Tiefentladen verhindert werden kann.

Die Flugzeit t lässt sich aus dem Energieverhältnis herleiten:

$$E = P \times t \tag{3.6.1}$$

Mit

$$E = U \times Q \; und \; P = U \times I \tag{3.6.2}$$

Folgt (umgeformt nach t):

$$t = \frac{E}{P} = \frac{U \times Q}{U \times I} = \frac{Q}{I} \tag{3.6.3}$$

Setzt man nun $Q = C$ (Kapazität des Akkus) in Ah ergibt sich:

$$t = \frac{C}{I} \tag{3.6.4}$$

Um die Sicherheitsreserve der Kapazität zu berücksichtigen, wird diese mit dem Faktor von 0.7 verrechnet. Um das Resultat in Minuten zu erhalten, wird dieses mit einem Faktor, welcher das Verhältnis von Minuten pro Stunde beschreibt (60min/h) multipliziert.

$$t_{Flug} = 60_{min/h} \times \frac{C_{Akku} \times 0.7}{I_{CM4} + I_{Pixracer} + I_{Antrieb} \times 0.7} = 60_{min/h} \times \frac{1.1Ah \times 0.7}{2.2A + 1.8A + 14.5A \times 0.7} = 3.256min \tag{3.6.5}$$

Die Berechnung basiert auf einer konstanten Last. In der Praxis reduziert sich die Flugzeit durch:

- Teillast- und Regelverluste des ESC,
- Schwankungen im Schubbedarf (Steig- / Sinkflug),
- Sicherheitsreserve (20 % Restkapazität für Spannungseinbruch).

Damit ergibt sich eine realistische Flugzeit von ca. 3 Minuten.

Diese Werte dienten als Grundlage für die Missionsplanung sowie für den Abgleich in *Simulation* und Flugtest (Kapitel 4).

3.8 Missionsplanung mit QGroundControl

Um eine Mission zu fliegen, muss diese erst in der entsprechenden Missionsplanungssoftware erzeugt werden. Hier wurde *QGroundControl* (QGroundControl 2019) verwendet, da bei den Recherchen zu diesem Projekt mehrfach auf diese verwiesen wurde. Das Erstellen von Missionen ist in unzähligen Tutorials online zu finden, weshalb hier nicht weiter darauf eingegangen wird. Wichtig ist hier nur, dass ein Landepunkt definiert werden muss. Auch wichtig zu erwähnen ist, dass auf die Flugzeit geachtet werden muss.
Sobald eine Mission geplant ist, muss diese in den *Pixracer R15* geladen werden und das *UAV* ist startbereit.
(QGroundControl 2019)

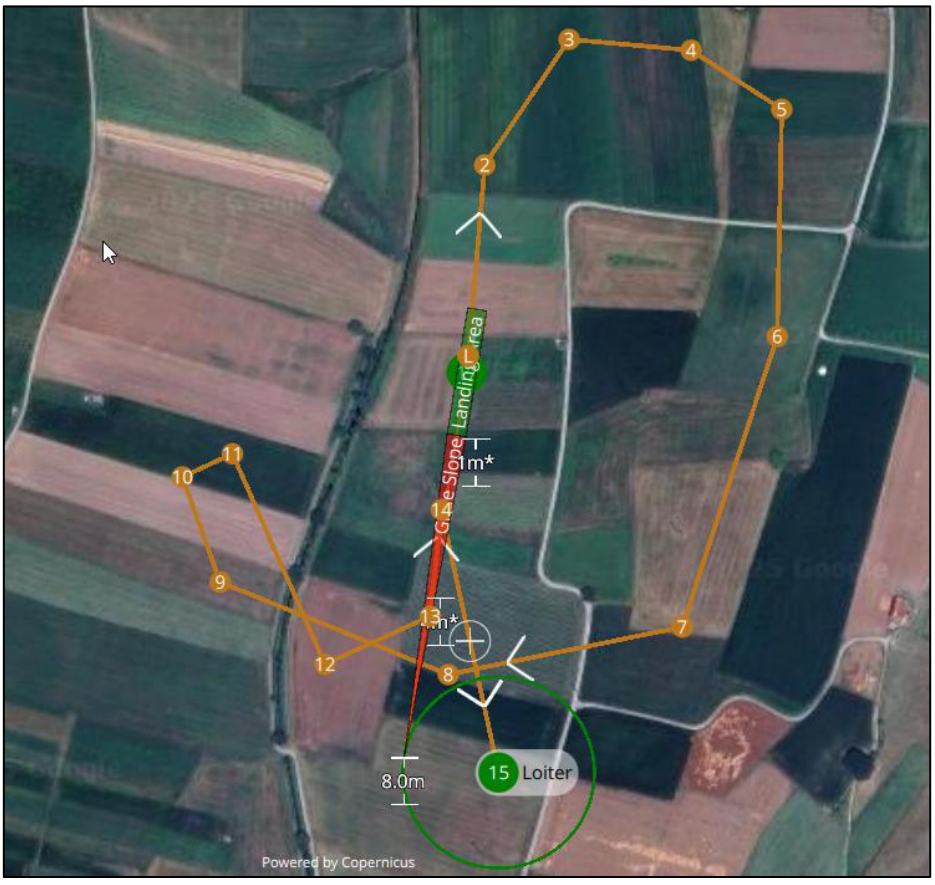


Abbildung 12 Missionsplanung QGroundControl (eigener Screenshot)

4. Ergebnisse

4.1 Systemerprobung durch Simulation

Bevor ich das *UAV* in der Realität fliegen lassen wollte, wurde es zuerst in einer *Simulation* getestet. Ziel war es, das Flugverhalten mit den berechneten Werten zu vergleichen und zu sehen, ob die Einstellungen grundsätzlich passen. So konnten Risiken minimiert und allfällige Fehler im Vorfeld gefunden werden.

Einleitung

Mittels *Simulation* wurde überprüft:

- ob sich das *UAV* mit den berechneten aerodynamischen Daten realistisch verhält,
- ob die Steuerung und Missionsplanung grundsätzlich funktionieren,
- und ob alle wichtigen Flugphasen wie Start, Steigflug, Reiseflug und Landung stabil ablaufen.

So wurde sichergestellt, dass das *UAV* im echten Flug keine bösen Überraschungen bereitet.

Aerodynamische Berechnungen

Bevor die *Simulation* aufgebaut wurde, sind die wichtigsten aerodynamischen Kenngrössen berechnet worden. Diese Werte dienten als Grundlage, um das Flugverhalten realitätsnah im Simulator abzubilden.

Die Berechnungen umfassten:

- die Schub- und Leistungsberechnung des Antriebssystems,
- die Flächenbelastung,
- die Stallgeschwindigkeit,
- die Reisegeschwindigkeit,
- die Steigrate,
- sowie den erforderlichen Flugschub im Horizontal- und Steigflug.

Die Berechnungen wurden nach den Formeln von Klaus-Peter Neitzke, wissenschaftlicher Mitarbeiter der TU-Dresden, (Neitzke 1999) durchgeführt.

Alle Herleitungen, Rechenschritte und Ausgangswerte sind im Anhang Kapitel A zu finden.

Diese Werte wurden anschließend direkt in die XML-Datei des *UAV* übernommen, damit das Modell im Simulator das gleiche Flugverhalten zeigt wie in der Realität.

Simulationsaufbau

Für die *Simulation* wurde *JSBSim* (JSBsim 2025) verwendet. Dieses läuft bei mir im Windows-Subsystem für Linux (WSL).

Abbildung 13 JSBsim gestartet (eigener Screenshot)

Damit die *Simulation* meinem Modell entspricht, wurde ursprünglich geplant, eine eigene XML-Datei komplett neu zu erstellen. Dabei sind jedoch mehrere Probleme aufgetreten. Einige Parameter wurden vom Simulator nicht akzeptiert, und Fehler wurden teilweise erst zur Laufzeit sichtbar.
Um die *Simulation* trotzdem zuverlässig nutzen zu können, wurde schliesslich entschieden, ein bestehendes Flugzeugmodell zu kopieren und meine Werte (Masse, Flügelfläche, Schub, Auftriebs- und Widerstandsbeiwerte usw.) dort einzutragen. So konnten die wichtigsten aerodynamischen Eigenschaften übernommen werden und trotzdem mein *UAV* realitätsnah abgebildet werden.
Dabei wurden unter anderem folgende Werte berücksichtigt:

- die Schub- und Leistungsberechnung des Antriebssystems,
- die Flächenbelastung,
- und die Fluggeschwindigkeiten (Stall-, Reise- und Steiggeschwindigkeit).

Damit das Ganze auch steuerbar ist, nutzte das System eine TCP-Verbindung zwischen *JSBSim* und *QGroundControl* (JSBSim 2025; QGroundControl 2019) eingerichtet.
Da die vorhandenen Antriebe in meinem Repository zu gross und schwer waren, nutzte die Simulation zusätzlich einen Dummy-Antrieb, der mit 2.16 N Schub, als externe physikalische Kraft in der Längsachse, arbeitet. Dieser Wert entspricht dem real gemessenen Schub des Impellers und wird linear mit der Schubvorgabe skaliert.

Durchführung

Die Testmission wurde in *QGroundControl* erstellt. Sie bestand aus mehreren Wegpunkten und einem definierten Landeziel.

Damit das *UAV* immer am gleichen Ort startet, wurde zusätzlich eine Reset-XML-Datei angelegt. Darin sind, unter anderem, die Startposition (Koordinaten) und die Höhe über Meer definiert.

Beim Test habe ich vor allem auf folgende Punkte geachtet:

- wie sich das Modell um die drei Achsen verhält (Roll, Nick, Gier),
- ob es die geplante Route sauber abfliegt,
- und ob die Übergänge zwischen den Flugphasen (z. B. Reiseflug → Landung) stabil sind.

Die Steuerung über *QGroundControl* funktionierte dabei genau gleich wie im realen Betrieb.

Eine Einschränkung war, dass kein *RC*-System an der *Simulation* angeschlossen war.

Auswertung

JSBSim erstellt wie der echte *Pixracer R15* Logdaten im .bin-Format.

Diese wurde mit einem kleinen *Python*-Skript (generiert mit Unterstützung von *ChatGPT*, OpenAI, Modell GPT-5) in *HTML*-Dateien umgewandelt. So konnten die Daten übersichtlich im Browser angeschaut und analysiert werden.

Vor allem angeschaut:

- die Höhe über der Zeit,
- die Geschwindigkeit,
- den Kursverlauf,
- und ob Soll- und Ist-Werte zueinander passen.

In Abbildung 13 (Höhenprofil Barometrisch *Simulation*) ist die aufgezeichnete Flughöhe über der Zeit dargestellt.

Zu erkennen ist, dass der Simulierte Steigflug gleichmässig erfolgt ist.

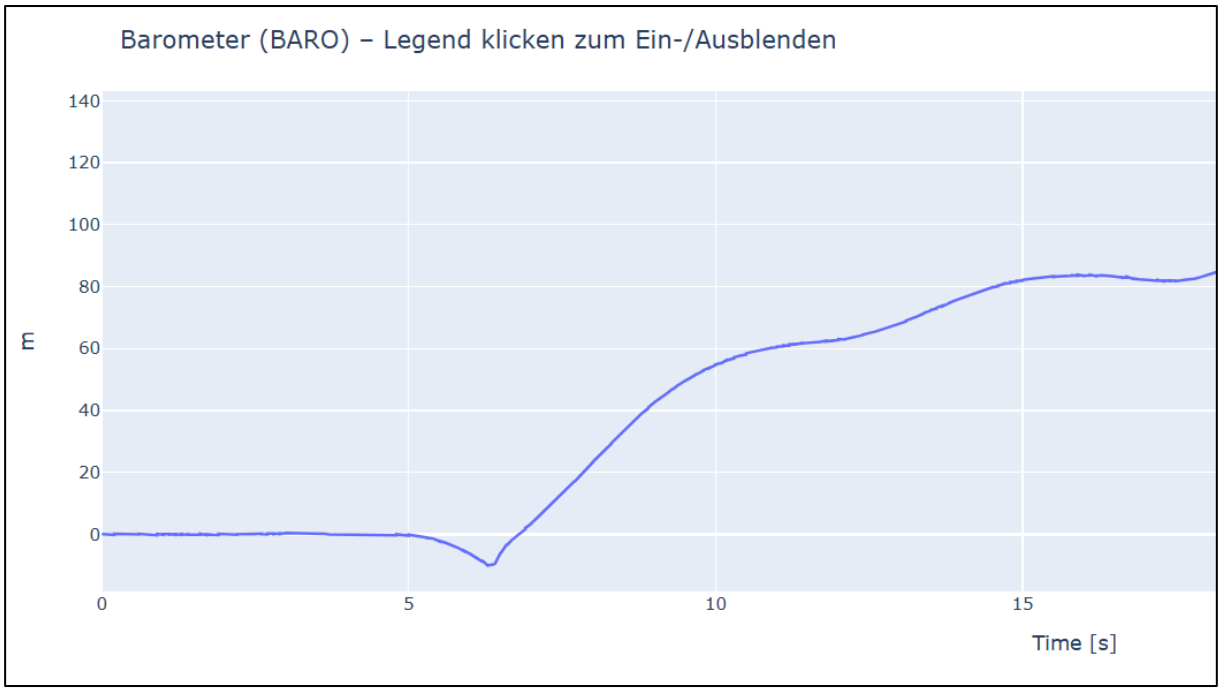


Abbildung 14 Höhenprofil Barometrisch Simulation (eigener Screenshot)

In Abbildung 14 (Steuerflächen *Simulation*) sind die Bewegungen der Steuerflächen während desselben Fluges zu sehen. Es zeigt sich hier ein ruhiges Steuerverhalten, ohne Oszillationen.

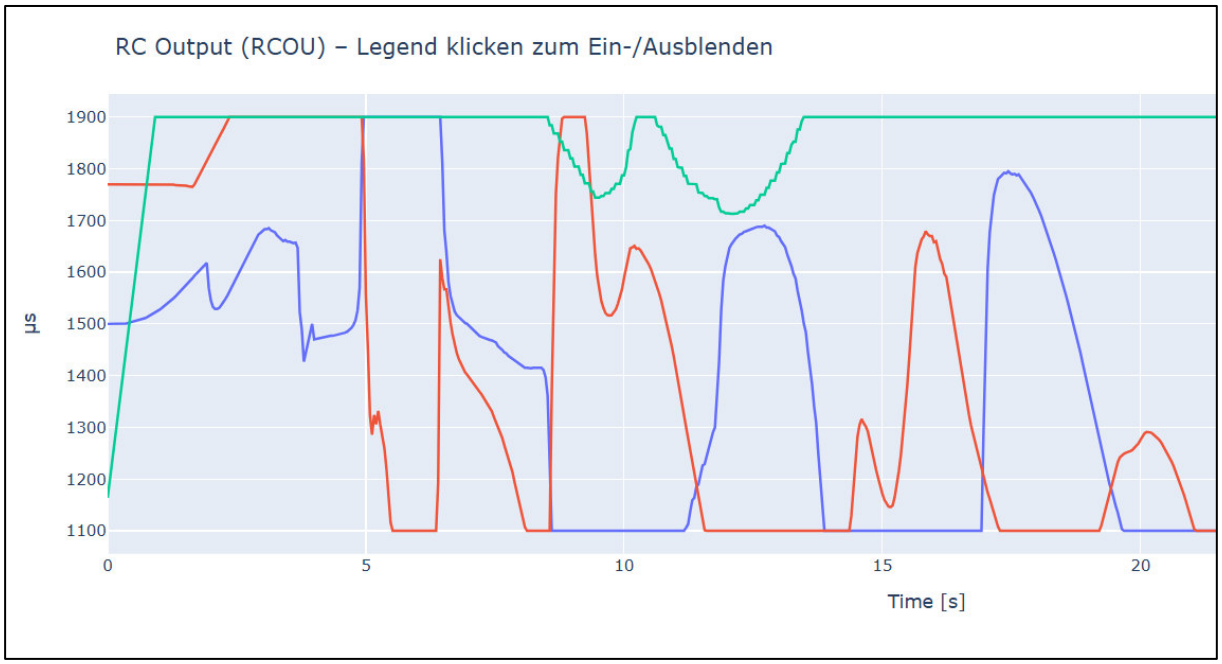


Abbildung 15 Steuerflächen Simulation (eigener Screenshot)

Dabei zeigte sich, dass das *UAV* die Wegpunkte präzise abfliegt, stabil bleibt und sich insgesamt wie erwartet verhält. Auch die zuvor berechneten Werte für Schub, Geschwindigkeit und Steigrate haben gut gepasst.

Erkenntnisse und Abweichungen

Am meisten Zeit habe ich nicht beim Fliegen der *Simulation*, sondern beim XML-Modell verloren. Mein Ziel war, das *UAV* von Grund auf sauber zu parametrisieren. In der Praxis hat das nicht zuverlässig funktioniert: einzelne Bezeichner/Parameter wurden von *JSBSim* nicht akzeptiert, Einheiten und Referenzpunkte waren schnell vertauscht (z. B. Schwerpunkt vs. Achsursprung), der Syntax bei Berechnungen war genau einzuhalten, und die Zuordnung der Ruderflächen (Aileron/Elevator/Rudder inkl. Ausschlagrichtungen) war fehleranfällig. Dazu kamen Fehlermeldungen erst zur Laufzeit, was die Fehlersuche erschwert hat.

Am Ende habe ich deshalb ein bestehendes Flugzeugmodell kopiert und dort meine Daten eingetragen (Masse, Flügelfläche, AR, CD_0/k, Schub usw.). Das lief stabil, hatte aber mehrere Auswirkungen:

- Modelltreue: Einige aerodynamische Annahmen des Basis-Modells bleiben erhalten (z. B. Verteilungsannahmen, Trägheitsmomente, Steuerkennlinien). Das erklärt Abweichungen bei Geschwindigkeit, Höhenprofilen und insbesondere Flugzeit zwischen Soll (Berechnung) und Ist (*Simulation*).
- Antriebsabbildung: In Kombination mit dem vereinfachten Dummy-Antrieb (linear auf 2.16 N skaliert) werden Teillast-Effekte und Wirkungsgradkurven nicht realitätsnah nachgebildet. Dadurch verschiebt sich die Reichweiten-/Flugzeitabschätzung.
- Sensorausfälle und *Failsafe*: In der *Simulation* liessen sich bewusst, Fehler von verschiedenen Sensoren Simulieren. Um das korrekte Verhalten des *UAV*, bei Steuerung via Autopiloten, zu testen, wurden verschieden Szenarien simuliert. Dabei zeigte sich eine stabile Failsaferoutine.

Trotzdem war die *Simulation* für meine Ziele sehr hilfreich: Stabilität, Navigationsverhalten und Missionslogik liessen sich prüfen. Das exakte Verhalten wurde bewusst im realen Flugtest (Kap. 4.2) verifiziert.

Fazit der Simulation

Die *Simulation* hat gezeigt, dass das Flugmodell richtig parametrisiert ist und die Flugphysik realistisch abgebildet wird.

Das *UAV* flog stabil, reagierte nachvollziehbar auf Steuerbefehle des Autopiloten, konnte Sensorausfälle handeln, und erreichte die geplanten Wegpunkte wie vorgesehen.

Damit war die Grundlage gelegt, um im nächsten Schritt mit gutem Gefühl in die realen Flugtests (Kapitel 4.2) zu gehen.

4.2 Systemerprobung durch Flugtests

Nach der erfolgreichen *Simulation* wurde das *UAV* in mehreren realen Flugtests erprobt. Ziel war es, das in der *Simulation* bestätigte Flugverhalten unter realen Bedingungen zu überprüfen und mögliche Schwachstellen im praktischen Betrieb zu erkennen.
Jeder Flugtest wurde anhand definierter Kriterien bewertet: Stabilität der Fluglage, GPS-Genauigkeit, Reaktionszeit des Failsafe-Systems und Präzision der Landung. Die Auswertung erfolgte anhand der Pixracer-Logdaten (.bin) sowie Python-Analyseskripten.
Alle Tests fanden auf unserem Flugfeld in Seftigen (BE) unter Sichtflugbedingungen (VLOS) statt.

Einleitung

Die Flugversuche sollten zeigen,

- ob das *UAV* stabil fliegt,
- ob die Regelung korrekt arbeitet,
- wie stark sich das reale Verhalten von der *Simulation* unterscheidet.
- Ob das Projekt ein Erfolg ist.

Als Energiespeicher kam derselbe 3S-*LiPo* mit 1.1 Ah zum Einsatz wie in der Berechnung (Kap. 3.6). Zwischen den Flügen wurde der Akku jeweils geladen. Die jeweiligen Flüge sind Tabellarisch in der Tabelle 9 Flüge zusammengefasst.

Abbildung 16 Flugtag (eigenes Foto)

Übersicht der Flüge

Tabelle 8 Flüge

Flug	Beschreibung	Ergebnis
1	Start misslungen, UAV nach ca. 2 s Bodenkontakt	Bruch Nase (kleiner Schaden)
2	Erfolgreicher Handstart, stabiler Flug, Manualmode	OK
3	Stabiler Flug, Missionsprofil abgeflogen	OK
4	Wiederholungsflug, Vergleich mit Simulation	OK
5	Längerer Flug mit GPS-Loss und Unterspannung, Bruchlandung	->Analyse

Flug 5 Ablauf und Beobachtungen

Der fünfte Flug wurde mit derselben Mission wie in Flug 4 geflogen. Der Start und der Steigflug verliefen unauffällig. Im Landeanflug, etwa 1 m über der Landebahn, trat ein merkbar steilerer Sinkflug auf. Kurz davor bzw. zeitgleich zeigte sich im später ausgelesenen Log ein deutlicher Spannungsabfall am Antriebsakku, gefolgt von *GPS*-Loss.

Da keine *Telemetrie* implementiert war, gab es während des Flugs keine Live-Meldungen oder Warnanzeigen. Die Ursachenanalyse erfolgte ausschliesslich anhand des Logfiles, das nach dem Flug vom *Pixracer R15* ausgelesen und in *HTML* konvertiert wurde.

Das UAV setzte hart auf und erlitt dabei einen strukturellen Schaden am Pendelruder, der eine weitere Inbetriebnahme ohne Reparatur unmöglich machte. Der Testflug konnte somit nicht wiederholt werden.

Auswertung (Loganalyse)

Die Auswertung basiert auf dem Log des fünften Fluges (nachträglich aus dem *Pixracer R15* gelesen).

Spannung & Strom

- Der Spannungsverlauf fällt gegen Ende des Fluges unter die Warnschwelle; kurz vor dem Aufsetzen zeigt sich ein rapider Einbruch.
- Die Stromspitzen liegen im Bereich der erwarteten Last; im Landeanflug sinkt der Mittelstrom, der Spannungseinbruch ist daher nicht durch eine Lastspitze, sondern durch geringe Restkapazität/Innenwiderstand erklärbar.

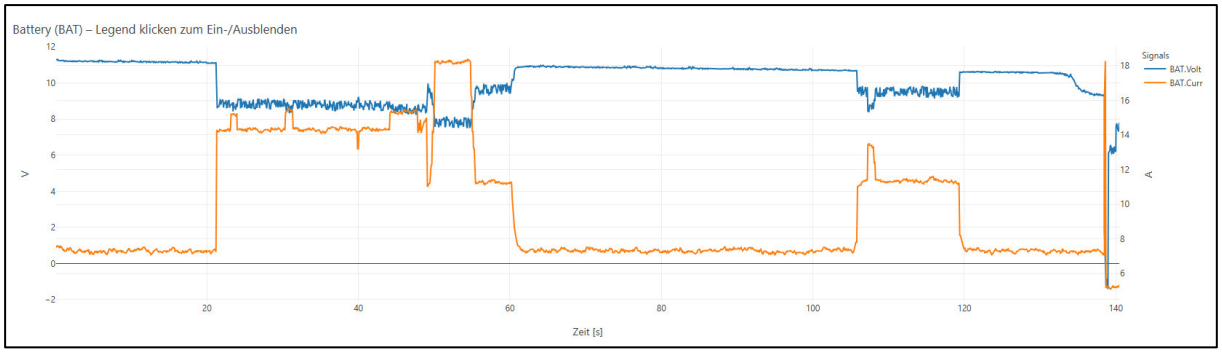


Abbildung 17 5. Flug (Mission 4) Strom / Spannung log (eigener Screenshot)

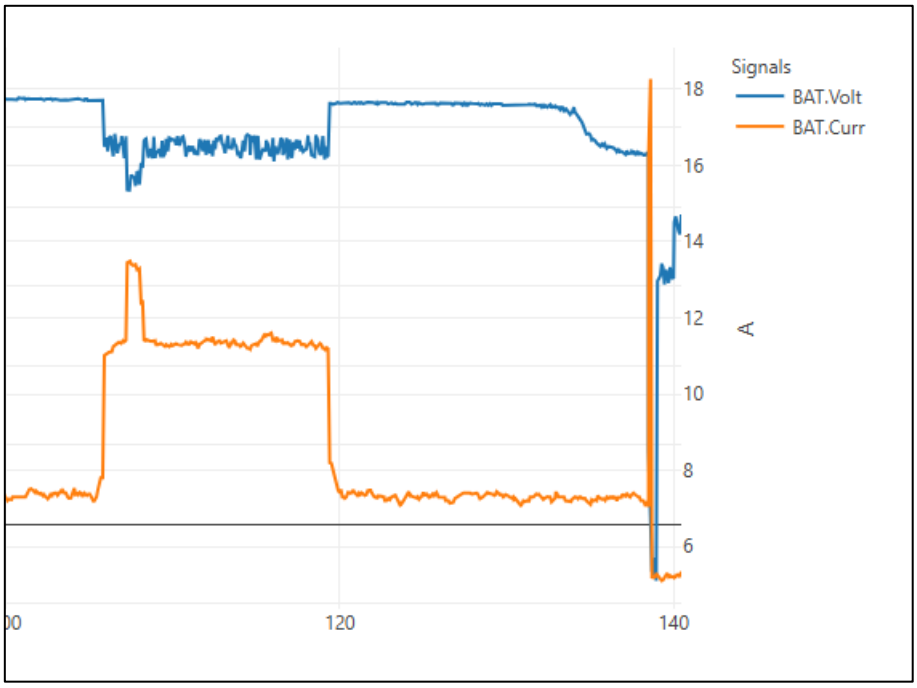


Abbildung 18 5. Flug (Mission 4) Strom / Spannung log vergrössert (eigener Screenshot)

GPS-Status

- Zeitgleich mit dem Spannungsminimum verliert der Empfänger den *GPS*-Fix (Satellitenzahl/Fix-Status bricht ein).
- Ohne *Telemetrie* wurden diese Ereignisse nicht live angezeigt; sie sind nur im Log sichtbar.

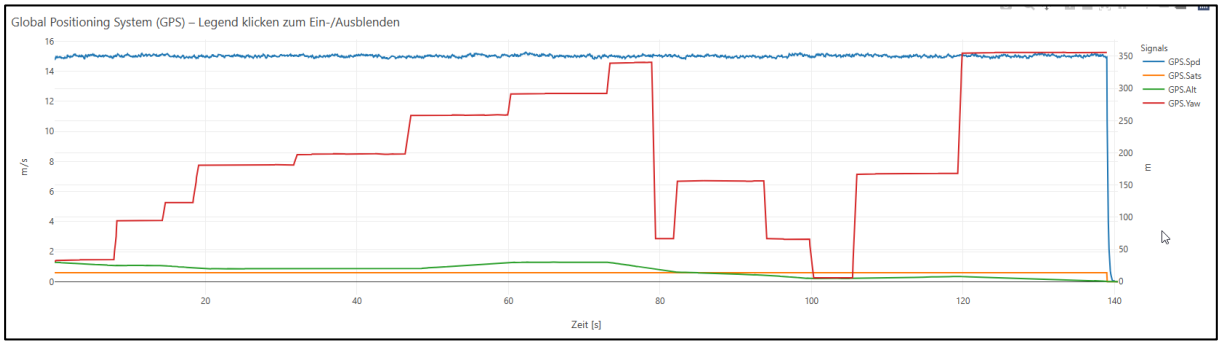


Abbildung 19 5. Flug (Mission 4) GPS log (eigener Screenshot)

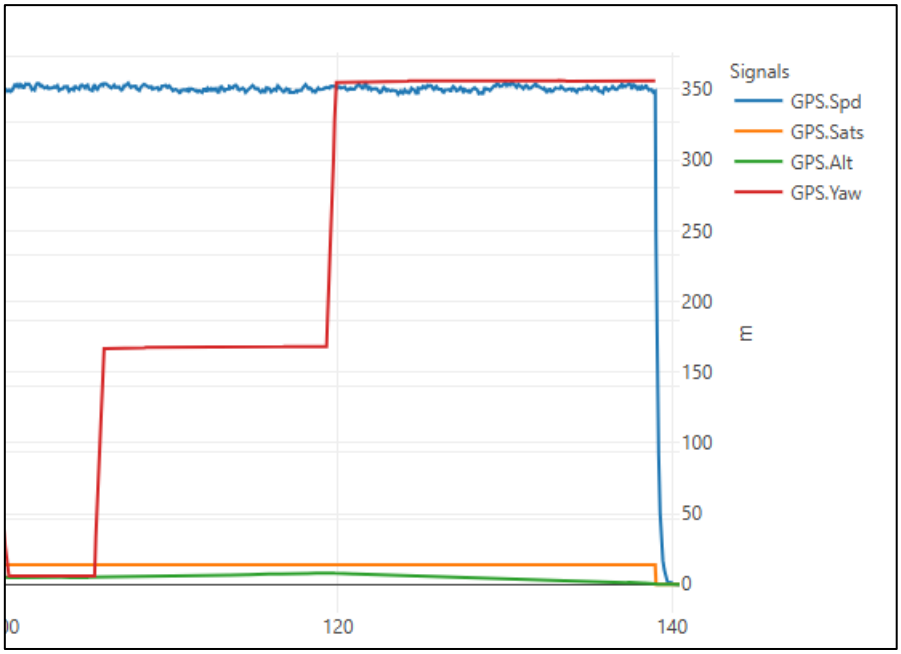


Abbildung 20 5. Flug (Mission 4) GPS log vergrössert (eigener Screenshot)

Interpretation

Der Spannungseinbruch im Landeanflug führte zu:

- Leistungsverlust am Antrieb → steiler Sinkflug kurz vor der Bahn,
- Unterversorgung des *GPS*-Empfängers → *GPS*-Loss unmittelbar vor dem Aufsetzen,
- dadurch fehlte dem Autopiloten die Positionsreferenz in der kritischsten Phase.

Die Beobachtung am Platz (steiler Sinkflug ~1 m über Bahn) deckt sich 1:1 mit den Logdaten (Spannungsminimum + *GPS*-Loss).

Erkenntnisse

- Das *UAV* zeigt insgesamt stabiles Flugverhalten; kritischer Engpass ist die Energieversorgung.
- *Failsafe* bei *GPS* loss greift (stabile Landung, nicht genau am Startpunkt wegen Trägheitsnavigation).

Fazit der Flugtests

Die Flugtests bestätigten das aus der *Simulation* erwartete Grundverhalten. Der fünfte Flug war besonders lehrreich: Er zeigte, wie kritisch die Versorgungsspannung in der finalen Phase ist und wie schnell *GPS*-Funktionen bei Spannungseinbruch ausfallen können.

Durch die Loganalyse liess sich die Ursache eindeutig nachvollziehen.

Tabelle 9 Ergebnistabelle

Ziel	Testmethode	Ergebnis	Bewertung
GPS-Waypoints autonom abfliegen	Flüge 2 – 5	Erfolgreich	Erfüllt
Autoland	Flüge 4 – 5	Erfolgreich, Pythonskript hat Fehldetektionen	Erfüllt. Restriktionen vom BAZL sind angerechnet
RC-Failsafe	Groundtests	Erfolgreich	Erfüllt
Sensorfehler Failsafe	Groundtests	Funktion gemäss Vorgabe	Erfüllt

5. Diskussion

In diesem Kapitel werden die Ziele, definiert in Kapitel 2.2, mit den erzielten Ergebnissen verglichen. Im Anschluss werden die Erfahrungen, Fehlerquellen und mögliche Verbesserungspotentiale diskutiert.

5.1 Vergleich Berechnung-Simulation-Realflug

Die theoretisch berechnete Flugzeit lag bei rund 3 Minuten 20 Sekunden.
In der *Simulation* ergaben sich, je nach Missionsprofil, kürzere Flugzeiten von etwa 2 Minuten 50 Sekunden bis 3 Minuten.
Die realen Flüge dauerten zwischen 3 und 3.5 Minuten.
Die Abweichung von bis zu 30 Sekunden erklärt sich durch:

- das höhere Startgewicht (Realflug 413 g statt 400 g *MTOW*),
- aerodynamische Verluste und reale Luftwiderstände,
- sowie den geringeren Wirkungsgrad des realen Antriebssystems.

Damit zeigte die *Simulation* den grundsätzlichen Trend, war jedoch optimistischer in Bezug auf Reichweite und Flugzeit.

5.2 Bewertung der Simulation

Die *Simulation* erwies sich als wertvolles Werkzeug, um das Steuerverhalten und die Missionslogik vor dem Erstflug zu prüfen.
Besonders das Stabilitätsverhalten und die Reaktion auf Steuerbefehle stimmten gut mit der Realität überein.
Einschränkungen ergaben sich durch:

- das vereinfachte Antriebsmodell (Dummy-Thrust),
- fehlende Massen- und Trägheitsverteilung,
- sowie eine nicht exakt abgebildete Rudercharakteristik.
- Fehlende *RC*- Anlage

Trotz dieser Vereinfachungen bot die *Simulation* eine solide Grundlage, um Parameter realitätsnah einzustellen und Risiken im Erstflug deutlich zu reduzieren.
Die Simulation bildete das Flugverhalten grundsätzlich korrekt ab, jedoch wurden aerodynamische Einflüsse wie Bodeneffekt oder Windböen nicht berücksichtigt. Diese Limitierung erklärt den Unterschied zum Realflug und sollte bei zukünftigen Simulationen durch Erweiterung der Modellparameter reduziert werden können.

5.3 Bewertung des Flugtests

Bewertung aus technischer Sicht:

Alle definierten MUSS-, SOLL- und KANN-Ziele aus Kapitel 2.2 wurden erreicht.

Das *UAV* war flugfähig, stabil, führte Missionen selbstständig aus und zeichnete alle relevanten Daten auf.

Besonderheiten aus dem praktischen Betrieb:

- *UBEC*-Nachrüstung für das *CM4*, da das *ESC*-BEC nicht genügend Leistung bot.
- Das zusätzliche Gewicht verschob die Abrissgeschwindigkeit leicht nach oben; das *UAV* wirkte im Flug träge („wie eine bleierne Ente“).
- Der fünfte Flug zeigte, wie kritisch die Energieversorgung ist: Unterspannung führte zu *GPS*-Ausfall und Bruchlandung.

Die Ursache der Unterspannung war ein nicht geladener Akku. Der Ladevorgang wurde wegen fehlender Eingangsleistung abgebrochen, aber nicht bemerkt.

Für zukünftige Versuche wäre eine andere Trägerplattform, z. B. *FMS PC-21 (1400 mm)*, mit grösserer Tragfläche und höherer Nutzlast von Vorteil.

Bewertung aus regulatorischer Sicht (BAZL):

Aus Sicht des BAZL unterliegt das UAV den Vorgaben der *offenen Kategorie*, Unterkategorie A3. Gemäss diesen Vorschriften muss der Betrieb stets innerhalb der Sichtweite (VLOS) erfolgen und der Pilot muss jederzeit manuell eingreifen können. Ein vollständig autonomer Flug, insbesondere während kritischer Phasen wie Start oder Landung, ist in dieser Kategorie nicht zulässig, da er als autonomer Betrieb ohne unmittelbare Steuerungsmöglichkeit gilt.

Das in diesem Projekt verwendete System zeigte zwar, dass ein Modellflugzeug mit *GPS*, *LIDAR* und *Kamera* technisch in der Lage ist, Wegpunkte autonom abzufliegen und selbstständig zu landen. Aus regulatorischer Sicht wäre ein solcher Betrieb jedoch nur im Rahmen einer speziellen Bewilligung (zum Beispiel nach *SORA-Light* oder in der speziellen Kategorie) erlaubt, da die automatische Landung eine sicherheitskritische Flugphase darstellt.

Positiv hervorzuheben ist, dass der *Autopilot* über eine Failsafe-Logik verfügt, die bei *RC*-Eingriff sofort in den manuellen Modus wechselt, womit eine zentrale Forderung des BAZL erfüllt wird.

Dennoch ersetzt dies nicht die vorgeschriebene ständige Überwachung durch den Piloten, und der autonome Betrieb darf nicht ohne Aufsicht erfolgen.
Insgesamt zeigt das Projekt aus regulatorischer Sicht, dass die technische Machbarkeit eines autonomen UAV zwar belegt ist, der praktische Einsatz aber rechtlich stark eingeschränkt bleibt. Ein Betrieb ausserhalb der Sichtweite oder ohne direkte Eingriffsmöglichkeit würde eine behördliche Bewilligung und weiterführende Sicherheitsnachweise erfordern. Das System ist somit als Testplattform unter Aufsicht zulässig, erfüllt aber noch nicht die Voraussetzungen für einen regulären Betrieb im offenen Luftraum (Bundesamt für Zivilluftfahrt kein Datum).

Die regulatorische Bewertung verdeutlicht, dass neben der technischen Leistungsfähigkeit auch die rechtlichen Rahmenbedingungen eine entscheidende Rolle spielen. Während die erzielten Ergebnisse den autonomen Betrieb grundsätzlich bestätigen, bleibt die praktische Umsetzung durch die Vorgaben des BAZL begrenzt. Für zukünftige Projekte ist es daher wichtig, technische Weiterentwicklungen stets im Kontext der geltenden Vorschriften zu betrachten. Im folgenden Kapitel werden die dabei gewonnenen technischen Erfahrungen und aufgetretenen Fehlerquellen detailliert analysiert.

5.4 Technische Erfahrungen und Fehlerquellen

Während der Entwicklung traten mehrere praxisrelevante Herausforderungen auf:

- Stromversorgung: ESC-BEC reichte nicht aus → zusätzlicher *UBEC* erforderlich.
- Startvorgang: Handstart benötigt ausreichend Geschwindigkeit, sonst Strömungsabriss.
- Keine *Telemetrie*: Fehler konnten nur im Logfile erkannt werden.
- Kameradateien: MP4-Dateien ohne Header bei abruptem Abschalten → gelöst durch automatisches 15 Sekunden Segmentieren und Dateischliessung bei Disarm.
- Programmierung: Erstes Projekt in objektorientiertem *Python*, anfangs schwierig, später wertvolle Erfahrung.
- Datenvalidierung: Tests durch Spaziergänge mit aktivem System, um *Logging* und Sensorik zu prüfen.

Diese Punkte verdeutlichen, dass Stromversorgung, Dateihandling und Softwarelogik entscheidend für die Systemzuverlässigkeit sind.

5.5 Verbesserungspotential

- Energie-Management: Checkliste zur Kontrolle der Akkuspannung erstellen.
- *Telemetrie*-Link: Live-Warnungen für Spannung und *GPS*-Status.
- Verfeinertes *Simulationsmodell*: realer Schub, Trägheitswerte, Ruderkennlinien.
- Robustere Plattform: z. B. *FMS PC-21 (1400 mm)* mit mehr Platz und Tragfähigkeit.

Im momentanen Weltgeschehen ist der Einsatz autonomer UAVs leider vor allem in Kriegseinsätzen präsent. Aus diesen Berichten, seien es Zeitungsartikel oder Telenews, gehen viele mögliche Verbesserungen ein. Die KI als Unterstützung ist hier nur als mögliches Beispiel zu nennen. In Anbetracht dieser Lage in der Welt möchte ich nicht zu sehr auf mögliche Verbesserungen eingehen, da die Erfahrung zeigt, dass alles immer als Kriegswaffe genutzt werden wird.

5.6 Lessons Learned

Das Projekt verdeutlichte, wie viele Subsysteme bei einem *UAV* zusammenspielen und wie kleine Details, etwa ein unvollständig geladener Akku, grosse Auswirkungen haben.
Die Kombination aus Berechnung, *Simulation* und Realtest erwies sich als effektiv, um das System schrittweise zu validieren.
Zudem bot das Projekt die Gelegenheit, erstmals objektorientiert in *Python* zu programmieren, eine anspruchsvolle, aber sehr lehrreiche Erfahrung.
Fazit: *Simulation* und Praxis ergänzten sich gut. Die gewonnenen Erkenntnisse bilden eine solide Basis für zukünftige Projekte mit verbessertem Energiemanagement, *Telemetrie* und einer grösseren, robusteren Plattform.

5.7 Zeitplanung und Abweichung

Die detaillierte Zeitplanung wurde bereits im Kapitel 2.4 beschrieben.
Dort sind die vier Hauptphasen, Arbeitspakete sowie das verwendete *Gantt-Diagramm* und das Zeiterfassungs-Tool (*Python*-Skript) erläutert.
Im Folgenden wird die tatsächliche Umsetzung dieser Planung bewertet und mit den Soll-Vorgaben verglichen. Dabei werden sowohl positive als auch negative Abweichungen dargestellt und die Ursachen sowie Erkenntnisse für künftige Projekte abgeleitet.

Positive Abweichungen (Zeitgewinn)
Viele Aufgaben konnten schneller als geplant abgeschlossen werden, insbesondere dank guter Vorbereitung und laufender Dokumentation.

Negative Abweichungen (Verzögerungen)
Andere Arbeitspakete erforderten mehr Zeit aufgrund technischer Hürden, falscher Kalkulation oder Lernphasen:

- Task Dokumentation: Aufwand unterschätzt. Soll 20h ist 39
- Task Phase 1 Pixracer Firmware, *ArduPlane*-Setup: Erstellen von Parameterdateien schwierig, Wechsel auf Desktopbedienerführung. Soll 2h ist 3.38h.
- Task Phase 2 Manuelle Groundtests: Testen der verschiedenen Einstellungen und Funktionen. Soll 5h ist 10.53h.
- Task Phase 4 SITL-Setup: Erstellen einer eigenen XML- Datei. Soll 5h ist 14.48h.

Die grössten Verzögerungen entstanden durch die Integration verschiedener Systeme sowie Unerfahrenheit mit neuen Frameworks.
Auch äussere Faktoren wie Wetterbedingungen für die Flugtests beeinflussten den Zeitplan.
Insgesamt verkürzte sich der Projektzeitraum um etwa 50 Stunden, wobei alle definierten Ziele erreicht wurden.
Für zukünftige Projekte empfiehlt sich:

- grösserer Zeitpuffer (25–30 %) für Integration und Tests
- frühzeitige Identifikation kritischer Pfade

6. Empfehlungen und Ausblick

Aus meinen Erfahrungen mit diesem Projekt ergeben sich für mich folgende Empfehlungen und Weiterentwicklungen.

Empfehlungen:

Empfehlenswert ist die Erstellung von Checklisten für Vorflugkontrolle und Nachflugkontrolle.
In der praktischen Erprobung hat sich gezeigt, dass menschliches Versagen der Hauptgrund für Fehler ist. Mit Checklisten können solche Versäumnisse relativ einfach vermieden werden.
Darüber hinaus wäre eine *Telemetrieanbindung* an eine Bodenstation oder an die Fernsteuerung von Vorteil. So liessen sich Werte wie Spannung und Strom und, je nach Art der *Telemetrie*, auch komplexe Flugdaten während eines Fluges auslesen und könnten auf das Lagebild des Piloten Einfluss nehmen und den Entscheidungsprozess vereinfachen.
In Anbetracht der Flugleistungen sollten leichter Komponenten eingesetzt werden, welche das maximale Abfluggewicht des *UAV* nicht über die Herstellerangabe kommen lassen. Ich würde auch empfehlen eine andere Trägerplattform zu nutzen, die ein höheres Abfluggewicht hat um das Mehrgewicht des Autopiloten nicht als limitierenden Faktor zu haben. In meinem Fall werde ich den PC21 von FMS als zusätzliche, weitere Plattform testen.
Es sollten noch mehr Testflüge stattfinden, in welchen unter verschiedenen Lichtbedingungen und Sichtbedingungen das System der Landebahnerkennung getestet werden.
Aus regulatorischer Sicht wird es, Stand 2025, keine Möglichkeit geben, das Projekt in Sinne der Autonomie ohne spezielle Bewilligungen weiterzuentwickeln. Für eine Umsetzung im regulären Betrieb, wäre daher eine Zulassung in der speziellen Kategorie erforderlich. Das bedingt jedoch auch redundante Systeme an Bord, was dem Punkt des zu hohen Gewichts widerspricht.

Ausblick:

Ich gedenke, dass *UAV* mit einer automatischen Flügelausklappung zu versehen, damit ich das von einem Trägerflugzeug in der Luft aus starten kann. So könnte das *UAV* selbstständig zum Landepunkt zurückkehren.

Ausserdem würde ich gerne eine künstliche Intelligenz für die Landebahnerkennung nutzen oder ein maschinelles Lernen einsetzen, um nicht nur meinen Heimflugplatz sondern jeden, per Satellitenbild zu erkennenden Modellflugplatz zu nutzen. Das jedoch bedingt wohl mehr Rechenleistung als das *CM4* bringt und somit auch Mehrgewicht.

Zusammenfassend bestätigen die Ergebnisse die technische Machbarkeit der autonomen Flug- und Landefunktionen im Modellflug. Trotz kleinerer Abweichungen zwischen Simulation und Realität konnten alle Ziele erreicht werden. Die gewonnenen Erkenntnisse bieten eine solide Grundlage für zukünftige UAV-Projekte im Semi-autonomen Bereich

7. Eigenständigkeitserklärung

Ich bestätige, dass ich die vorliegende Diplomarbeit selbstständig verfasst und alle benutzten Quellen gekennzeichnet habe. Diese Arbeit wurde weder in gleicher noch in ähnlicher Form bereits einer Prüfungskommission vorgelegt.

Name / Vorname:

Jarno Linder

Ort / Datum / Unterschrift:

8. Quellen und Hilfsmittel

8.1 Verzeichnisse

Quellenverzeichnis

aerojtp.com. kein Datum. https://aerojtp.com/s/aero-jtp/:Micro_Cruisers (Zugriff am 13. Januar 2025).

AliExpress Antrieb. kein Datum. https://de.aliexpress.com/item/1005006079992481.html?spm=a2g0o.order_list.order_list_main.30.47645c5frDvKXC&gatewayAdapt=glo2deu (Zugriff am 24. September 2025).

ArduPilot,. *ARDUPILOT.* 2024. https://firmware.ardupilot.org/Plane/stable-4.6.2/Pixracer/ (Zugriff am 12. September 2025).

Bundesamt für Kommunikation. kein Datum. https://www.bakom.admin.ch/de/drohnen-und-flugmodelle (Zugriff am 12. September 2025).

Bundesamt für Zivilluftfahrt. kein Datum. https://www.bazl.admin.ch/bazl/de/home/drohnen.html (Zugriff am 12. September 2025).

ChatGPT (GPT5). kein Datum. https://chatgpt.com (Zugriff am 28. September 2025).

CM4-NANO-A. kein Datum. https://www.waveshare.com/wiki/CM4-NANO-A (Zugriff am 12. September 2025).

Github usbboot. 2024. https://github.com/raspberrypi/usbboot (Zugriff am 12. September 2025).

holybro GPS. kein Datum. https://docs.px4.io/main/en/gps_compass/gps_holybro_m8n_m9n.html (Zugriff am 23. September 2025).

Inalhan, Mirac Aksugur / Gokhan. *researchgate.net.* 1. Februar 2009. https://www.researchgate.net/publication/225938050_Design_Methodology_of_a_Hybrid_Propulsion_Driven_Electric_Powered_Miniature_Tailsitter_Unmanned_Aerial_Vehicle (Zugriff am 07. Oktober 2025).

JSBsim. Mai 2025. https://jsbsim.sourceforge.net/ (Zugriff am 20. September 2025).

Linder, Jarno. *Github/Autopilot-Microcruiser-RC-Flugzeug.* 5. Oktober 2025. https://github.com/plexgen/Autopilot-Microcruiser-RC-Flugzeug (Zugriff am 5. Oktober 2025).

MAVLink. *MAVLink Micro Air Vehicle Message Marshalling Library.* 14. März 2022. https://github.com/ArduPilot/mavlink/blob/master/message_definitions/v1.0/common.xml#L1008 (Zugriff am 14. September 2025).

Meramid live editor. kein Datum. https://mermaid.live (Zugriff am 15. September 2025).

nach Aliexpress Pixracer. kein Datum. https://de.aliexpress.com/i/1005001666309617.html (Zugriff am 5. Oktober 2025).

Neitzke, Klaus-Peter. *TU-Dresden.* 1999. https://tu-dresden.de/ing/maschinenwesen/ilr/ressourcen/dateien/tfd/studium/dateien/Flugmechanik_U.pdf?lang=de (Zugriff am 23. September 2025).

Pixracer, R15 mRo. *Mrobotics User Guide.* 1. Januar 2023. https://docs.mrobotics.io/discontinued/pixracer-r15.html (Zugriff am 12. September 2025).

QGroundControl. 2019. https://qgroundcontrol.com/ (Zugriff am 12. September 2025).

Raspberry Pi CM4. 2012. https://www.raspberrypi.com/products/compute-module-4/?variant=raspberry-pi-cm4001000 (Zugriff am 23. September 2025).

Raspberry Pi Imager. kein Datum. https://www.raspberrypi.com/software/ (Zugriff am 12. September 2025).

Raspberry Pi Picam3. 2012. https://www.raspberrypi.com/products/camera-module-3/ (Zugriff am 19. September 2025).

raspberrypi OS. kein Datum. https://www.raspberrypi.com/software/operating-systems/ (Zugriff am 12. September 2025).

Sourceforge. 27. Februar 2002. https://www.tinycad.net/ (Zugriff am 16. September 2025).

TF-Luna LIDAR. kein Datum. https://www.waveshare.com/wiki/TF-Luna_LiDAR_Range_Sensor (Zugriff am 17. September 2025).

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Tabelle 10 Formelverzeichnis

Nr.	Formel	Beschreibung	Seite
(3.6.1)	$E = P \times t$	Grundformel Energie	35
(3.6.2)	$E = U \times Q \; und \; P = U \times I$	Elektrische Energie und Leistung in Abhängigkeit von Spannung, Ladung und Strom	35
(3.6.3)	$E = P \times t$	Energiebilanz	35
(3.6.4)	$C = I \times t$	Kapazität aus Strom und Zeit	35
(3.6.5)	$C = I \times t$	Angewandte Formel der Kapazitätsberechnung	35
(A-1.1)	$T = D + W \times sin(\gamma)$	Kraftgleichgewicht in Flugrichtung	A-1
(A-1.2)	$L = W \times cos(\gamma)$	Auftrieb im stationären Steigflug	A-1
(A-1.3)	$D = q \times S \times C_D$	Aerodynamischer Gesamtwiderstand	A-1
(A-2.1)	$C_L = \frac{W}{q \times S}$	Auftriebsgleichgewicht im stationären Flug	A-2
(A-2.2)	$C_{L,MD} = \sqrt{\frac{C_{D0}}{k}}$	Auftriebsbeiwert bei minimalem Widerstand	A-2
(A-2.3)	$D_{min} = 2 \times W \times \sqrt{C_{D0} \times k}$	Minimaler aerodynamischer Gesamtwiderstand	A-2
(A-2.4)	$(\frac{L}{D})_{max} = \frac{1}{2 \times \sqrt{C_{D0} \times k}}$	Maximales Gleitverhältnis	A-2
(A-3.1)	$\sqrt{C_{D0} \times k}$	Hilfsrechnung für Minimalwiderstand und beste Gleitzahl	A-3
(A-3.2)	$D_{min} = 2 \times W \times \sqrt{C_{D0} \times k}$	Berechnung des Minimalwiderstands für den MicroCruiser	A-3
(A-3.3)	$(\frac{L}{D})_{max} = \frac{1}{2 \times \sqrt{C_{D0} \times k}}$	Berechnung des maximalen Gleitverhältnisses	A-3
(A-3.4)	$T = D + W \times sin(\gamma)$	Berechnung des minimalen Schubs	A-3
(A-3.5)	$T_{erf} = D + W \times \sin(\gamma)$	Erforderlicher Schub im Steigflug (5°)	A-3
(A-3.6)	$T_{fl} = 0.6 \times T_{Impeller}$	Berechnung des verfügbaren Schubs im Flug	A-3
(A-3.7)	$V_S = \sqrt{\frac{2 \times W}{\rho \times S \times C_{Lmax}}}$	Überziehgeschwindigkeit im Horizontalflug	A-3
(A-3.8)	$\mathrm{P_{excess}} = (\mathrm{T} - \mathrm{D}) \times \mathrm{V}$	Leistungsüberschuss bei 15m/s	A-3
(A-3.9)	$ROC = \frac{P_{excess}}{W}$	Berechnung der Steigrate aus der Überschussleistung	A-3

8.2 Hilfsmittel

Hardware:

Tabelle 11 Hilfsmittel Hardware

Kategorie	**Bezeichnung / Typ / Hersteller**	**Verwendungszweck**
Montagematerial	Doppelseitiges Klebeband Tesa	Befestigung Pixracer und GPS
Fertigung Additiv	3d-Drucker Bambu Lab P1S	Herstellung MicroCruiser
Filament	PLA-Aero Bambu Lab	Herstellung MicroCruiser
Werkzeug	Schraubenzieher Kreutzschlitz / Schlitz Grösse 0 und 00	Montage Servos und Anlenkungen
Werkzeug	Lötstation	Motor / UBEC / ESC / Steckersysteme
Werkzeug	Cuttermesser	
Klebstoff	Pattex	Zusammenbau
Steckverbinder	JST-GH / JST-ZH	Anschluss Peripherie Pixracer
T-Litzen	2.5mm^2 / 0.34mm^2 div. Farben	Verdrahtung gemäss Schema
Kabel	USB-C Powerkabel mit offenem Ende	Speisung CM4
Computer	Notebook	Dokumentation / Groundstation / Programmierung

Software:

Tabelle 12 Hilfsmittel Software

Kategorie	Bezeichnung / Typ / Hersteller	Verwendungszweck	Version
CAD	Fusion 360 / Autodesk	Erstellen Halter LIDAR und Kamera	v.2604.0.316
CAD	TinyCAD /	Schema zeichnen	3.00.03
Texteditor	Notepad++	Coding	8.8.5 64bit
Programmierumgebung	VS Code	Coding	1.104.3
Officeprogramme	Office 365		
Missionsplanungssoftware	QGroundControl	Setup Pixracer und Missionsplanung / Datendownload	V5.0.7 64bit
Simulation	JSBsim	Simulationssoftware	V1.2.3
Repository	Github	Repository Anhang B	

Sonstige Hilfsmittel:

Tabelle 13 Hilfsmittel sonstige

Kategorie	Bezeichnung / Typ / Hersteller	Verwendungszweck
Taschenrechner	Texas Instruments TI-30XIIS	Berechnungen
Schreibwaren	Kugelschreiber / Papierblock	Div. Notizen

A) Anhang A Technische Daten & Berechnungen

Technische Daten Microcruiser und Berechnungen der Aerodynamik

Ausgangsdaten:

Tabelle 14 Ausgangsdaten (eigene Tabelle)

Grösse	Symbol	Wert	Einheit	Anmerkung
Spannweite	b	0.46	m	
Flügelfläche	S	0.03	m²	
Masse (Abfluggewicht)	m	0.413	kg	Flugfertig
Gewichtskraft	W = m·g	4.05	N	
Luftdichte	ρ	1.225	kg/m³	Standardatmosphäre
Profilwiderstandsbeiwert	C_D0	0.06		konservativ für Mikro-Modelle
Oswald-Faktor	e	0.8		
Streckung	AR = b²/S	7.05		
Induzierter Faktor	k = 1/(π·e·AR)	0.056		
Max. Auftriebsbeiwert	C_Lmax	1.0		kleine Reynolds-Zahl
EDF-Daten		QF1611-7000KV @ 3S		laut Datenblatt
Standschub	T_stat	2.16	N	= 220 g
Flugschub (≈60%)	T_fl	1.3	N	konservative Näherung aus Literatur
Eingangsleistung	P	124	W	laut Datenblatt

Quellen: (AliExpress Antrieb kein Datum) , (Neitzke 1999)

Aerodynamische Grundlagen:

Für den stationären Horizontal- oder Steigflug gilt (Neitzke 1999):

Kraftgleichgewichtsgleichung

$$T = D + W \times sin(\gamma) \qquad \text{(A-1.1)}$$

Auftrieb im stationären Steigflug

$$L = W \times cos(\gamma) \qquad \text{(A-1.2)}$$

Aerodynamischer Gesamtwiderstand

$$D = q \times S \times C_D, q = \frac{1}{2} \times \rho \times V^2, C_D = C_{D0} + kC_L^2 \qquad \text{(A-1.3)}$$

Mit:

$$C_L = \frac{W}{q \times S} \quad \text{(A-2.1)}$$

Mit folgenden Einheiten:

Tabelle 15 Einheiten

Symbol	Bedeutung	Einheit
T	Schubkraft	N
D	Widerstandskraft	N
L	Auftriebskraft	N
W	Gewichtskraft ((W = m \cdot g))	N
γ	Flugbahnwinkel	° oder rad
q	Staudruck	Pa = N/m²
S	Flügelfläche	m²
ρ	Luftdichte	kg/m³
V	Geschwindigkeit	m/s
C_D	Widerstandsbeiwert	–
C_{D0}	parasitärer Widerstandsbeiwert	–
k	induzierter Widerstandsbeiwertfaktor	–
C_L	Auftriebsbeiwert	–

Bestes Gleiten / minimaler Widerstand:

Minimaler Widerstand bei:

$$C_{L,MD} = \sqrt{\frac{C_{D0}}{k}} \quad \text{(A-2.2)}$$

Gesamtwiderstand im Minimum:

$$D_{min} = 2 \times W \times \sqrt{C_{D0} \times k} \quad \text{(A-2.3)}$$

Gleitverhältnis:

$$\left(\frac{L}{D}\right)_{max} = \frac{1}{2 \times \sqrt{C_{D0} \times k}} \quad \text{(A-2.4)}$$

Einsetzen und Berechnen mit konservativen Werten gemäss Quelle:

$$\sqrt{C_{D0} \times k} = \sqrt{0.06 \times 0.056} = 0.0579 \quad \text{(A-3.1)}$$

$$D_{min} = 2 \times 4.05N \times 0.0579 = 0.47N \quad \text{(A-3.2)}$$

$$(\frac{L}{D})_{max} \approx 8.6 \quad \text{(A-3.3)}$$

Für Horizontalflug wird ein Schub von ca.

$$\boldsymbol{T_{min} = 0.47N} \quad \text{(A-3.4)}$$

Benötigt.

Für das Steigen im 5° Winkel gilt:

$$T_{erf} = D + W \times \sin(\gamma) = 0.47N + 4.05N \times \sin(5°) = 0.82N \quad \text{(A-3.5)}$$

Dabei ist zu beachten, dass der Schub im Flug abnimmt, da die Rotorblätter des Impellers immer schneller angeströmt werden. Im Flug verfügbar (Inalhan 2009):

$$T_{fl} = 0.6 \times 2.16N = 1.30N \quad \text{(A-3.6)}$$

Da Tfl > Terf, ist ein Steigflug mit 5 bis 10° realistisch bei sehr konservativen 60%.

Stallgeschwindigkeit:

$$V_S = \sqrt{\frac{2 \times W}{\rho \times S \times C_{Lmax}}} = \sqrt{\frac{2 \times 4.05N}{1.225kg/m^3 \times 0.03m^2 \times 1.0}} = 14.8\,m/s \quad \text{(A-3.7)}$$

Die Horizontalgeschwindigkeit muss die Stallgeschwindigkeit auf jeden Fall übertreffen.

Dabei ist eine Steigrate gemäss nachfolgender Berechnung zu erwarten:

Leistungsüberschuss:

$$\mathrm{P_{excess}} = (\mathrm{T} - \mathrm{D}) \times \mathrm{V} = (1.30 - 0.47)\mathrm{N} \times 15\,\mathrm{m/s} = 12.45\mathrm{W} \quad \text{(A-3.8)}$$

Steigrate:

$$ROC = \frac{P_{excess}}{W} = \frac{12.45W}{4.05N} = 3.1\,m/s \quad \text{(A-3.9)}$$

Die weiteren Werte für die XML- Datei wurden durch Testen bzw. Kopieren bestehender Daten ermittelt.

B) Anhang B Software, Parametrierungen, Testprotokolle & Skripte

Dieser Anhang dokumentiert die zur Umsetzung des autonomen Landeverfahrens verwendete Softwareumgebung. Er umfasst die zentralen Python-Skripte, Systemdienste (systemd), Parameterdateien des Flugcontrollers sowie ergänzende Hilfsprogramme (Batch- und Auswertungsskripte).

Zur Nachvollziehbarkeit und Reproduzierbarkeit ist das gesamte Projekt als öffentliches GitHub-Repository verfügbar. Dort sind sämtliche im Anhang beschriebenen Dateien strukturiert abgelegt und versioniert:

GitHub-Repository https://github.com/plexgen/Autopilot-Microcruiser-RC-Flugzeug

- 📁 **scripts/** – Hauptskripte zur Bildverarbeitung, MAVLink-Kommunikation und Zeiterfassung
- - autoland_trigger_on_land.py – Hauptskript zur autonomen Triggerung des Landemodus bei erkannter Bahn
- - Sensordaten_plott_und_excel.py – Auswertungsskript (BIN → HTML + Excel-Export)
- - zeiterfassung.py – Skript zur Zeiterfassung aus der Vorprojektphase (Erstellt mit Unterstützung eines KI-Hilfsmittels, GPT-5)
- 📁 **batch/** – Datentransfer
- - Datendownload.bat – Automatischer Video- und Log-Download vom Raspberry Pi
- 📁 **configs/** – Konfigurationen
- - microcruiser.xml – Fahrzeugkonfiguration
- - reset_microcruiser.xml – Reset-Konfiguration
- 📁 **params/** – Parameterdateien
- - pixracer_autoland.param – Pixracer-Parameterdatei (ArduPlane 4.6.2)
- 📁 **docs/** – Begleitdokumentation
- - Anhang_B_Autopilot_Microcruiser_final.pdf – Begleitdokumentation (dieser Anhang)

Systemübersicht:

Trägerplattform: Microcruiser RC-Flugzeug (JTPaero – Microcruiser)

- Spannweite ≈ 460 mm, klassisches Starrflügelmodell
- Antrieb: Elektromotor (Impeller)
- Steuerung: Quer-, Höhenruder, Gas
- Missionsprofil: Wegpunktflug, autonomes Landeverfahren

Autopilot: Pixracer R15 (ArduPlane 4.6.2)

- Fluglagenregelung, Missionsmanagement und Sensorfusion
- Kommunikation über MAVLink (UART) mit dem CM4
- Unterstützte Kommandos: SET_MODE, DO_LAND_START, SET_POSITION_TARGET_GLOBAL_INT
- Firmwarebasis: ArduPlane v4.6.2

Compute Module 4 (CM4):

- Raspberry Pi Compute Module 4
- CSI-Kamera (IMX708)
- Eigenes UBEC (5 V) zur Versorgung
- Laufzeitumgebung: python3-venv mit pymavlink, picamera2, opencv
- Systemd-Service autoland.service für automatischen Start beim Boot

Funktion:

- Visuelle Bahnerkennung in Echtzeit
- Automatischer Wechsel in den Landemodus
- Log- und Videoerstellung zur Nachweisführung

CM4 Setup (Kurzüberblick):

1. System aktualisieren:

   sudo apt update && sudo apt upgrade -y

2. Kamera aktivieren & Tools installieren:

   sudo apt install -y libcamera-apps python3-picamera2 python3-opencv python3-numpy python3-simplejpeg

3. Virtuelle Umgebung einrichten:

   python3 -m venv ~/mavenv --system-site-packages

   source ~/mavenv/bin/activate

   pip install pymavlink pyserial

4. Autostart (systemd):

   Service-Datei /etc/systemd/system/autoland.service mit ExecStart=/home/pi/mavenv/bin/python3 /home/pi/autoland_trigger_on_land.py

5. Service aktivieren:

   sudo systemctl daemon-reload

   sudo systemctl enable autoland.service

   sudo systemctl start autoland.service

Parameterdatei (Pixracer):

Die vollständige ArduPlane-Parameterdatei (pixracer_autoland.param) ist im Ordner params/ des GitHub-Repositories abgelegt. Sie enthält alle für das Projekt relevanten Einstellungen (u. a. Flugmodi, Sensorik, EKF3-Konfiguration, Landemodi).

Dokumentation & Nachweise:

Alle Nachweise (Video, Logdaten, Screenshots) sowie das vollständige Anhangsdokument: docs/Anhang_B_Autopilot_Microcruiser_final.pdf

GitHub: https://github.com/plexgen/Autopilot-Microcruiser-RC-Flugzeug

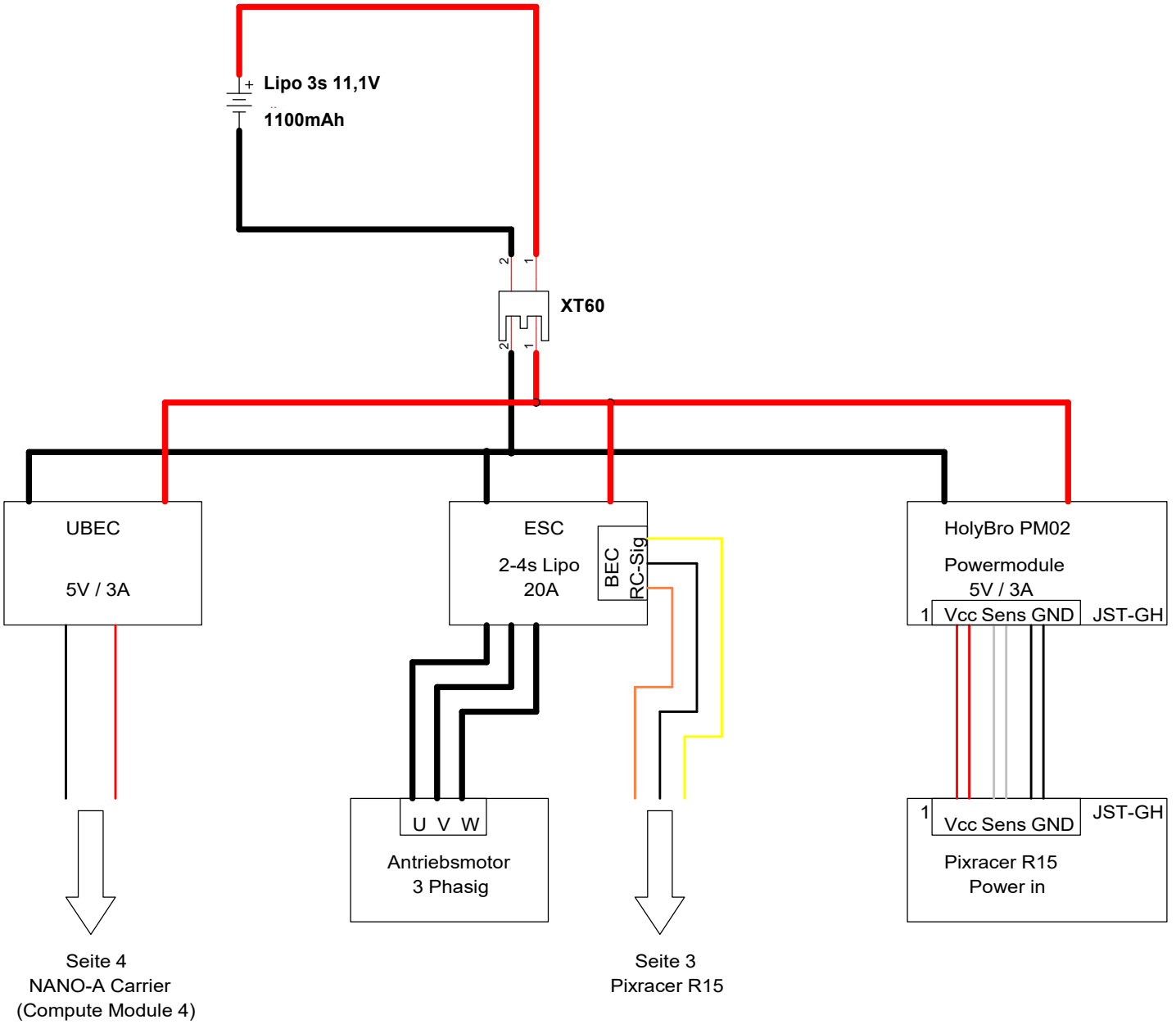
C) Anhang C Aufbau und Verdrahtung

Stückliste zu Schema:

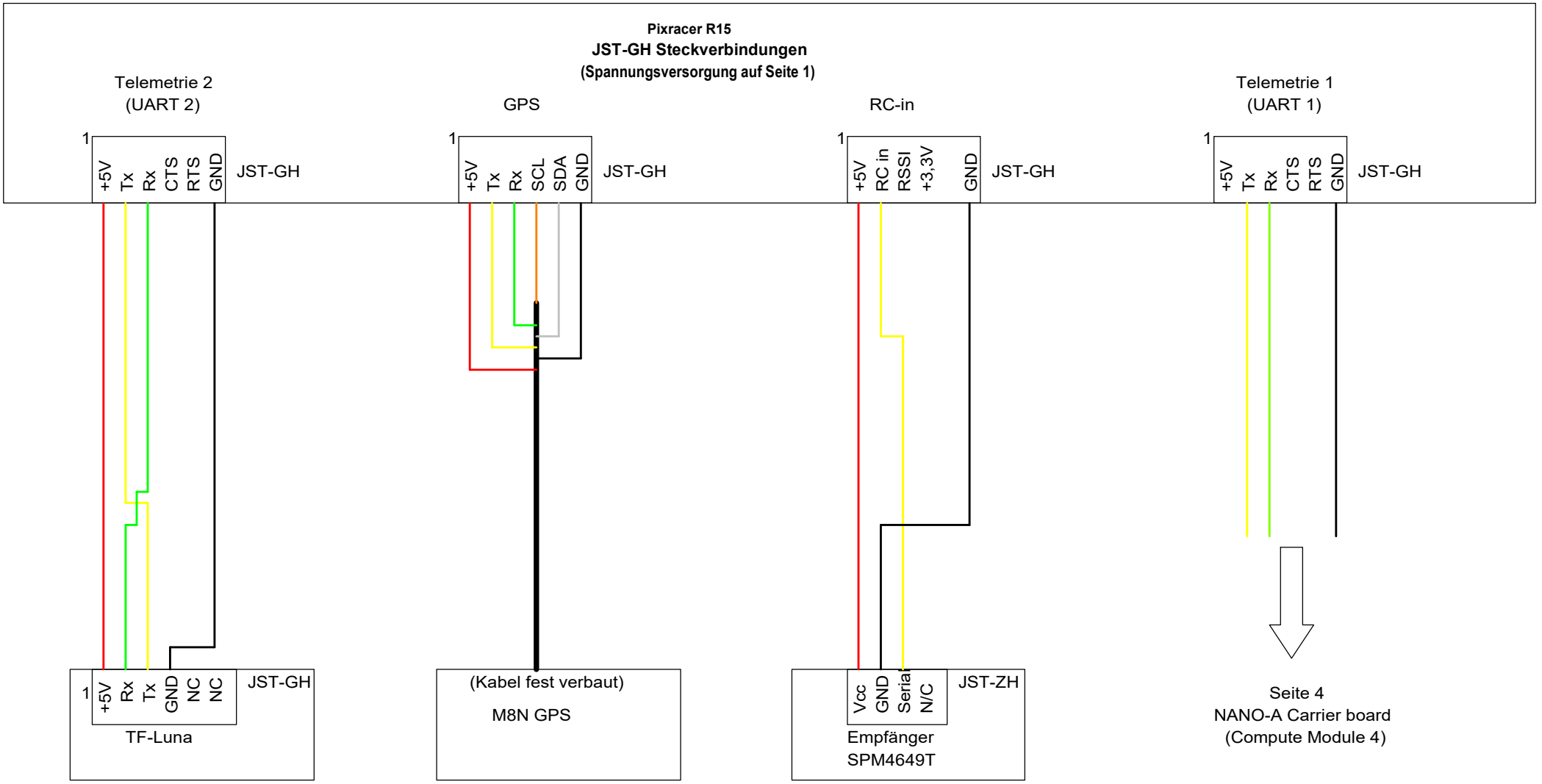
Tabelle 16 Stückliste

Artikel	Anzahl	Lieferant	Zusätzliche Infos
Pixracer R15	1	Aliexpress	Unbedingt mit Spannungsversorungsmodul und M8N GPS bestellen
CM4	1	Pi-shop.ch	Artikelnummer 11304
Carrier Board	1	Pi-shop.ch	Artikelnummer 11856
Picam3	1	Pi-shop.ch	Artikelnummer 11995
UBEC	1	Galaxus.ch	Artikelnummer 21608404
LiPo Akku	1	Swaytronic.ch	Artikelnummer 7640159360322
MicroCruiser	1	Aero-jtp.com	MC-01F30
JST-Verbinder	1	Amazon.de	Nach JST-GH suchen und von elechawk das Set beziehen
XT60-Steckverbinder	1	Galaxus.ch	Artikelnummer 6050498
USB-C Powerkabel	1	Digitec.ch	Artikelnummer 44876134
Halter zu Picam3	1	Autor	Kann im Github repo geladen werden
Halter zu LIDAR	1	Autor	Kann im Github repo geladen werden
LIDAR TF-Luna	1	Digitec.ch	Artikelnummer 19179757
Empfänger	1		Je nach RC-System, BUS-Anbindung nötig

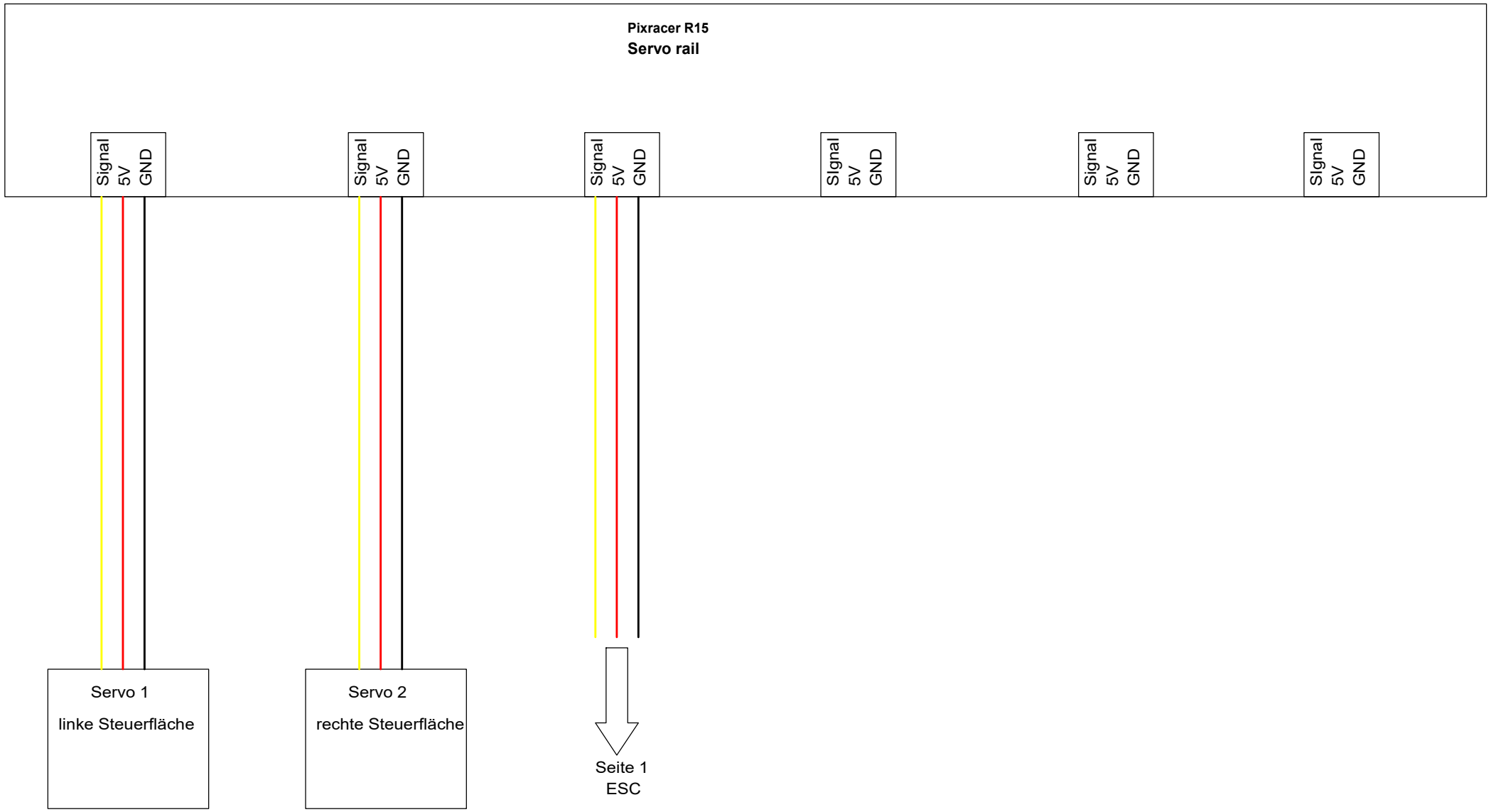
Auf den nächsten Seiten ist das Schema, zur Verdrahtung des UAV. Weitere nicht genannte Bauteile sind in den Unterlagen des MicroCruiser zu finden.



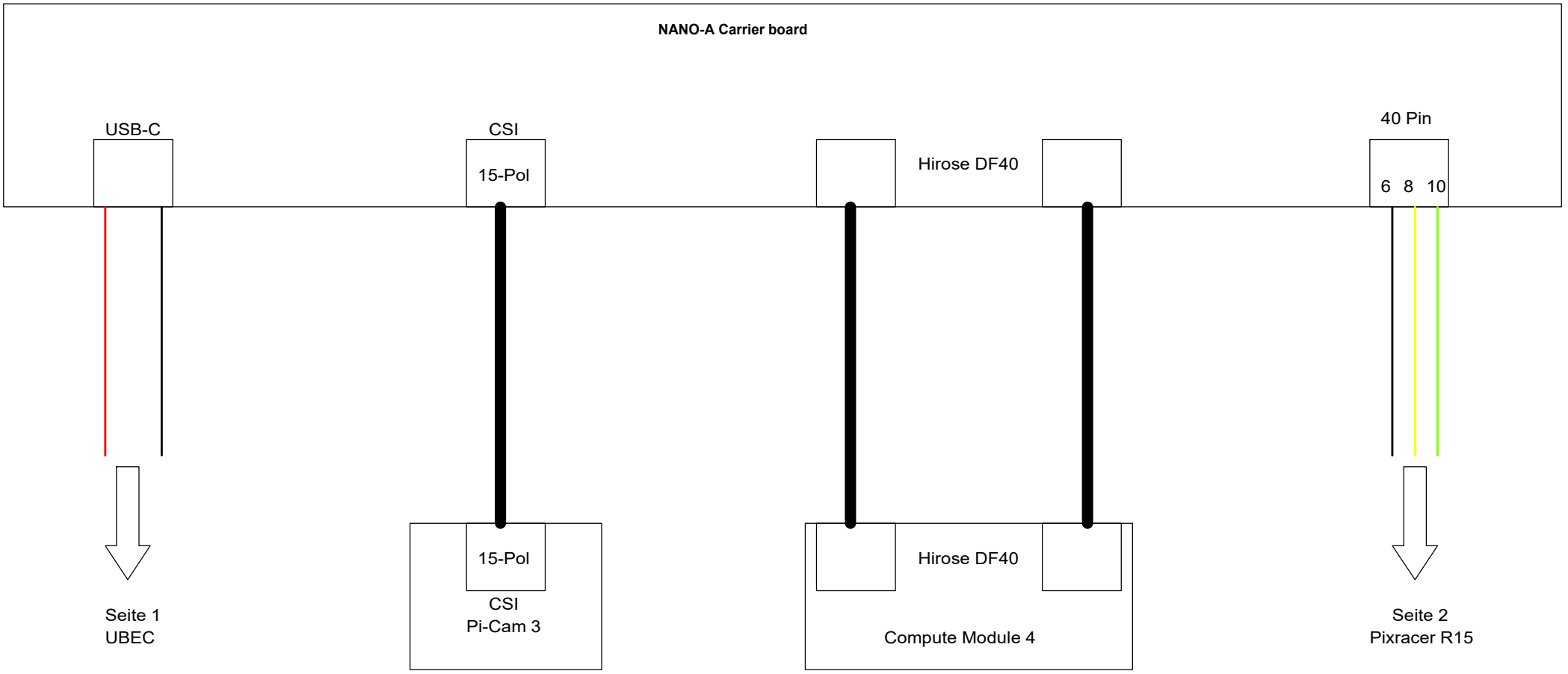
Title Micro Cruiser UAV Spannungsversorgung		
Author Jarno Linder Teko Bern		
File C:\ ... Micro Cruiser Spannungsversorgung.dsn		Document
Revision 1.0	Date 16.09.2025	Sheets 1 / 4



Title Micro Cruiser UAV Pixracer R15		
Author Jarno Linder Teko Bern		
File C:\ ... Micro Cruiser Spannungsversorgung.dsn		Document
Revision 1.0	Date 16.09.2025	Sheets 2 / 4



Title Micro Cruiser UAV Pixracer R15		
Author Jarno Linder Teko Bern		
File C:\ ... Micro Cruiser Spannungsversorgung.dsn		Document
Revision 1.0	Date 16.09.2025	Sheets 3 / 4

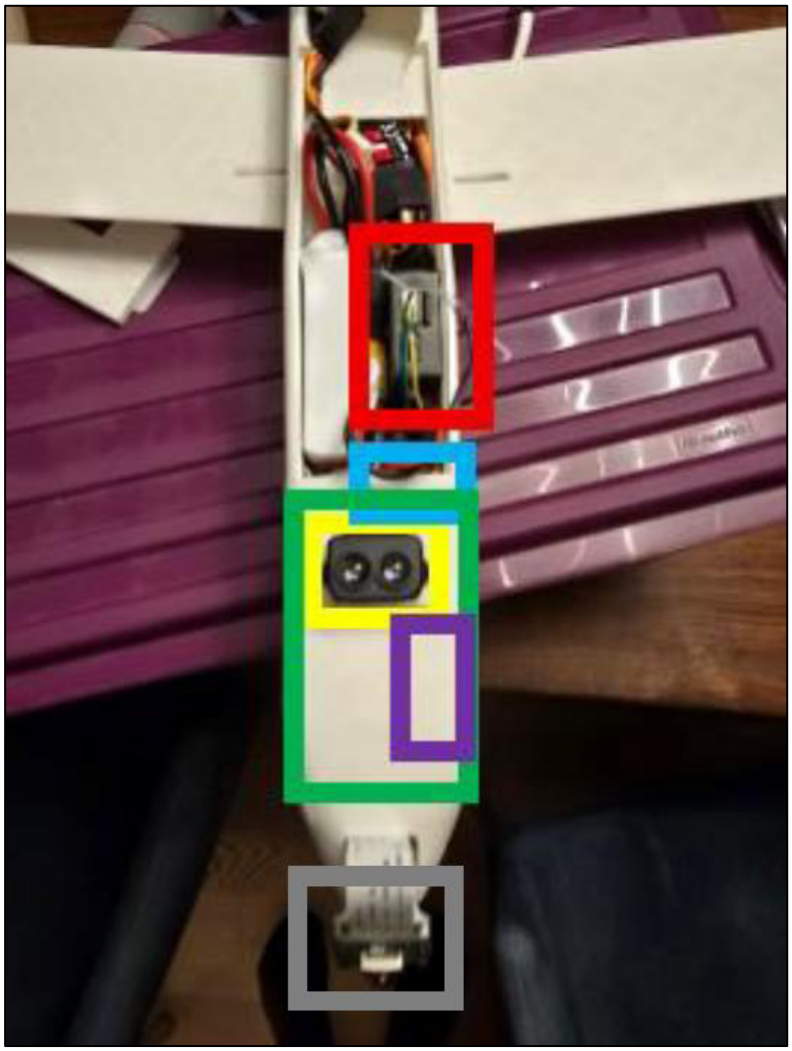


Title Micro Cruiser UAV NANO-A Carrier Board (Compute Module 4)		
Author Jarno Linder Teko Bern		
File C:\ ... Micro Cruiser Spannungsversorgung.dsn		Document
Revision 1.0	Date 16.09.2025	Sheets 4 / 4

Aufbau:

Der Aufbau ist gemäss Anleitung MicroCruiser zu erledigen, mit Ausnahme der Schritte, Servo zur Flächenaktivierung und RC-Einbau. Bis zum Abgabetermin dieses Projekts wurde der Klappmechanismus der Tragflächen nicht umgesetzt. Der Einbau der restlichen Komponenten, zu sehen in Abbildung 21 (Einbau vergrössert), ist so zu realisieren, dass der CM4 mit dem UBEC in der Rumpfnase geklemmt wird, und der Pixracer R15 mit doppelseitigem Klebeband an der Rumpfwand befestigt wird, USB-Anschluss sichtbar. Der Akku wird durch den Deckel gehalten.

Hier das vergrösserte Bild zu Seite 33 [Rücksprung].



Rot: *Pixracer R15*

Blau: RC-Empfänger

Grün: *CM4*

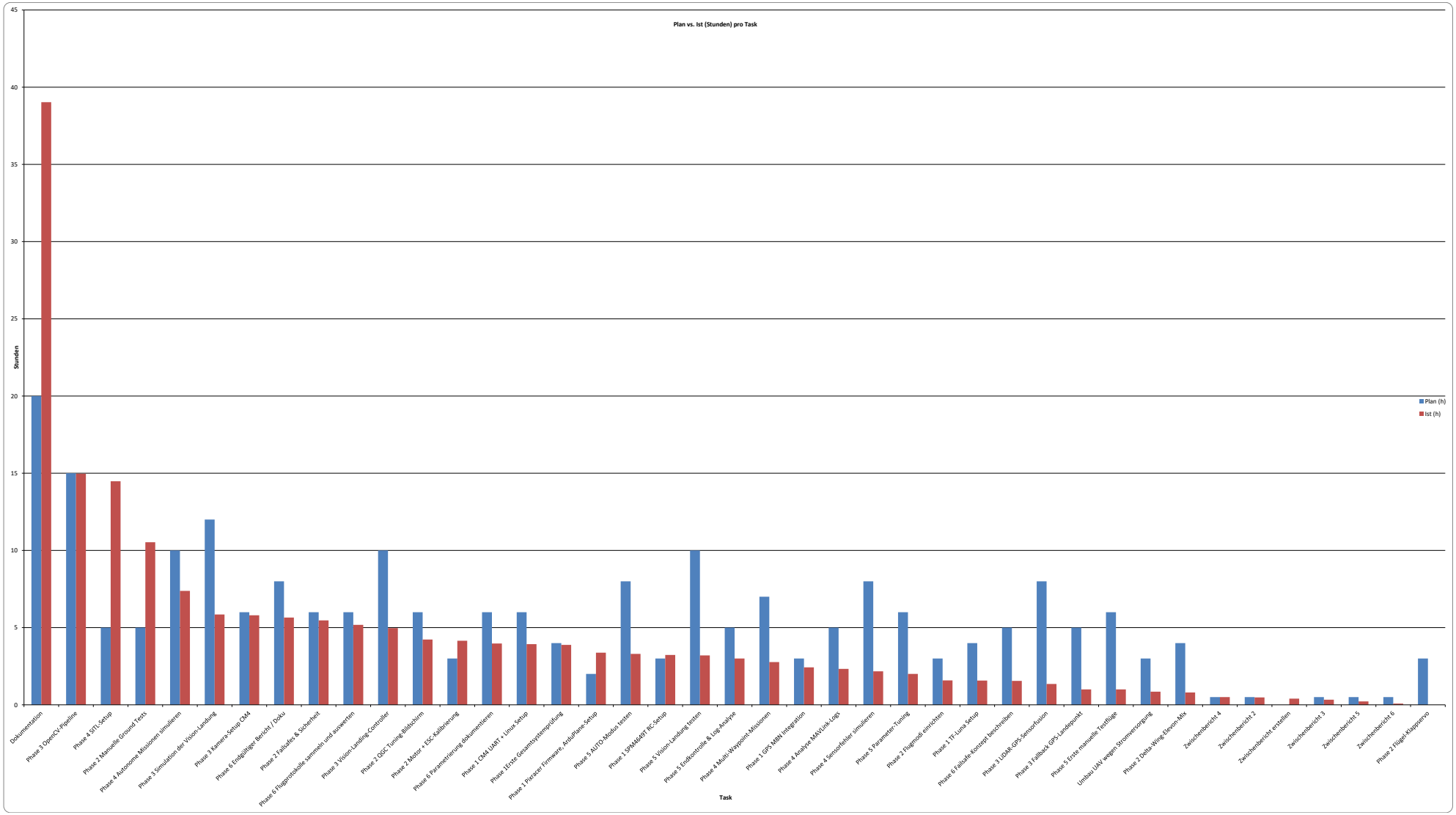
Violett: *UBEC*

Gelb: *LIDAR* auf Halter

Grau: *PiCam3* auf Halter

Abbildung 21 Einbau vergrössert (eigenes Foto)

D) Anhang D – Weitere Unterlagen und Referenzdokumente


Plan vs. Ist (Stunden) pro Task
Stunden
Task
Plan (h)
Ist (h)
0
5
10
15
20
25
30
35
40
45
Dokumentation
Phase 3 OpenCV-Pipeline
Phase 4 STL-Setup
Phase 2 Manuelle Ground Tests
Phase 4 Autonome Missionen simulieren
Phase 3 Simulation der Vision-Landung
Phase 3 Kamera-Setup CM4
Phase 6 Endgültiger Bericht / Doku
Phase 2 Failsafes & Sicherheit
Phase 6 Flugprotokolle sammeln und auswerten
Phase 3 Vision-Landing-Controller
Phase 2 QGC Tuning Bildschirm
Phase 2 Motor + ESC Kalibrierung
Phase 6 Parametrierung dokumentieren
Phase 1 CM4 UART + Linux Setup
Phase 1Erste Gesamtsystemprüfung
Phase 1 Pixracer Firmware, ArduPlane Setup
Phase 5 AUTO-Modus testen
Phase 1 SPM4645T RC-Setup
Phase 5 Vision-Landung testen
Phase 5 Endkontrolle & Log-Analyse
Phase 4 Multi-Waypoint-Missionen
Phase 1 GPS M8N Integration
Phase 4 Analyse MAVLink-Logs
Phase 4 Sensorfehler simulieren
Phase 5 Parameter-Tuning
Phase 2 Flugmodi einrichten
Phase 1 TF-Luna Setup
Phase 6 Failsafe-Konzept beschreiben
Phase 3 LIDAR-GPS-Sensorfusion
Phase 3 Fallback GPS-Landeport
Phase 5 Erste manuelle Testflüge
Umbau UAV wegen Stromversorgung
Phase 2 Delta-Wing Elevon-Mix
Zwischenbericht 4
Zwischenbericht 2
Zwischenbericht erstellen
Zwischenbericht 3
Zwischenbericht 5
Zwischenbericht 6
Phase 2 Flügel Klappservo

mRo Pixracer R15

PixRacer has all the capabilities of the original Pixhawk (including version 2) and even more! It is optimized in size and has just about the right amount of inputs/outputs that allow you to switch from a full-enriched autopilot stack system (with auto landing and full navigation) to a high-performance racing platform.

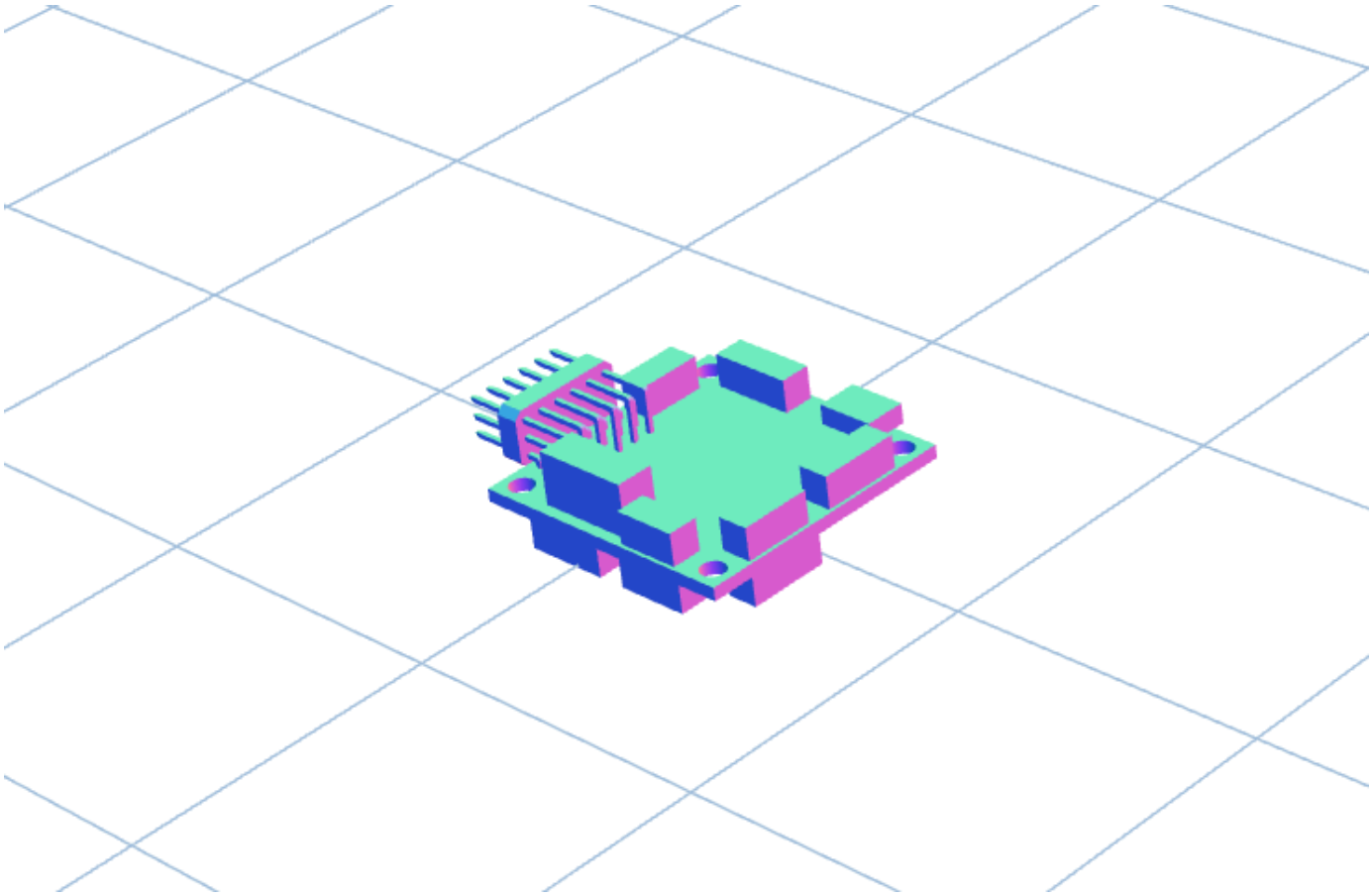
R15 has an updated accelerometer/gyro, magnetometer, and is ROHS (Lead Free). It also includes an ESP8266 for easy WiFi updates and comes with the latest **Ardupilot ESP8266 firmware (http://firmware.ardupilot.org/Tools/MAVESP8266/latest/)** developed by Andrew Tridge. This new firmware has a nice web interface, MavLink2 support, and an easier way to do future updates via the web interface.

Specifications

Specifications	mRo PixRacer R15
Main Processor	32-bit STM32F427 Cortex M4 core with FPU rev.3 168 MHz
IO Processor	No
RAM	256 KB RAM
Flash	2 MB FRAM
Crypto / Hash Processor	No
Accelerometers / Gyros / Mags	2 / 2 / 2
Sensors	Invensense/TDK ICM-20602 (6DOF) Invensense/TDK MPU-9250 (9DOF)
Sensors – Dampened	None
Internal Magnetometer	AK8963 inside MPU-9250 and ST LIS3MDL
Barometer	MEAS MS5611
Interfaces and Protocols	5x UART (serial ports)[2x with HW flow control and GPS+I2C®]. 1x PPM sum input signal 6x PWM outputs 1x RSSI (PWM or voltage) input

	1x SPI 1x CAN 1x JTAG (Debuging & programming interface) 8x OneShot PWM output (Configurable) 1x External microUSB port Dronecode Debug connector. WiFi Telemetry & firmware update via ESP8266 (Included). JST-GH connectors using Dronecode connector standard. Supported RC input protocols: Spektrum DSM / DSM2 / DSM-X® Satellite compatible input up to DX9 and above. Futaba S.BUS® & S.BUS2® compatible input. FRSky Telemetry port output. Graupner SUMD.Yuneec ST24.
Connectors	-JST GH series connectors -Servo Header -Onboard MicroUSB -2x 5 header (Esp-01)
Pin Headers	Yes
Conformal Coating	Available
Extended Testing and Burn In	No
Custom Carrier Board Support	No
LED	Yes
Dimensions	Width: 36mm (1.42″) Length: 36mm (1.42″)
Weight	10.54g (.37 oz)
Mounting Holes	30mm x 30mm (1.18″x1.18″)
Protector Case	Optional
Typical Platforms	-Multirotor -Rover -Fixed-Wing -Boats -Submarines -VTOL -Automatic Tractors -Others

3D Model



Firmware

The mRo Pixracer R15 is compatible with the following firmware:

ArduPilot

- ArduCopter 4.x
- ArduPlane 4.x
- ArduRover 4.x

PX4

- PX4 V1.8 (and further versions)

Normal Usage Guide

All connectors follow the Dronecode connector standard. Unless noted otherwise, all connectors are JST-GH.

Pinouts

Telemetry Ports

Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	TX (OUT)	+3.3V
3 (blk)	RX (IN)	+3.3V
4 (blk)	CTS (IN)	+3.3V
5 (blk)	RTS (OUT)	+3.3V
6 (blk)	GND	GND

Basic GPS Port

Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	TX (OUT)	+3.3V
3 (blk)	RX (IN)	+3.3V
4 (blk)	I2C1 SCL	+3.3V
5 (blk)	I2C1 SDA	+3.3V
6 (blk)	GND	GND

CAN Port

1 (red)	VCC	+5V
2 (blk)	CAN_H	CAN high
3 (blk)	CAN_L	CAN low
4 (blk)	GND	GND

I²C Port

Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	I2C SCL (1.5K pullup on autopilot)	+3.3V
3 (blk)	I2C SDA (1.5K pullup on autopilot)	+3.3V
4 (blk)	GND	GND

SPI Port

Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	SPI_EXT_SCK	+3.3
3 (blk)	SPI_EXT_MISO	+3.3
4 (blk)	SPI_EXT_MOSI	+3.3
5 (blk)	!SPI_SS1	+3.3
6 (blk)	!SPI_SS2	+3.3
7 (blk)	GND	GND

Analog Power Port

1 (red)	VCC	+5.3V
2 (blk)	VCC	+5.3V
3 (blk)	CURRENT	+3.3V
4 (blk)	VOLTAGE	+3.3V
5 (blk)	GND	GND
6 (blk)	GND	GND

Tutorials

- First Time Setup (https://ardupilot.org/copter/docs/initial-setup.html)
- CAN Bus Setup (https://ardupilot.org/copter/docs/common-canbus-setup-advanced.html)
- UAVCAN Setup (https://ardupilot.org/copter/docs/common-uavca~n-setup-advanced.html)
- Advanced Configuration (https://ardupilot.org/copter/docs/common-advanced-configuration.html)

Downloads

3D Model (https://mrobotics.io/wp-content/uploads/files/products/Autopilot/Pixracer/mRo_PixRacer.stl)

Last Updated: 1.3.2023, 12:27:00

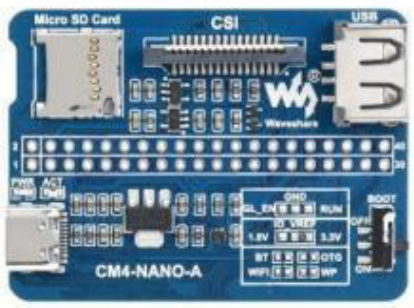
Contributors: Leonardo Garcia

CM4-NANO-A

From Waveshare Wiki
Jump to: navigation, search

Introduction

CM4-NANO-A-4



(https://www.waveshare.com/cm4-nano-a.htm)

Description

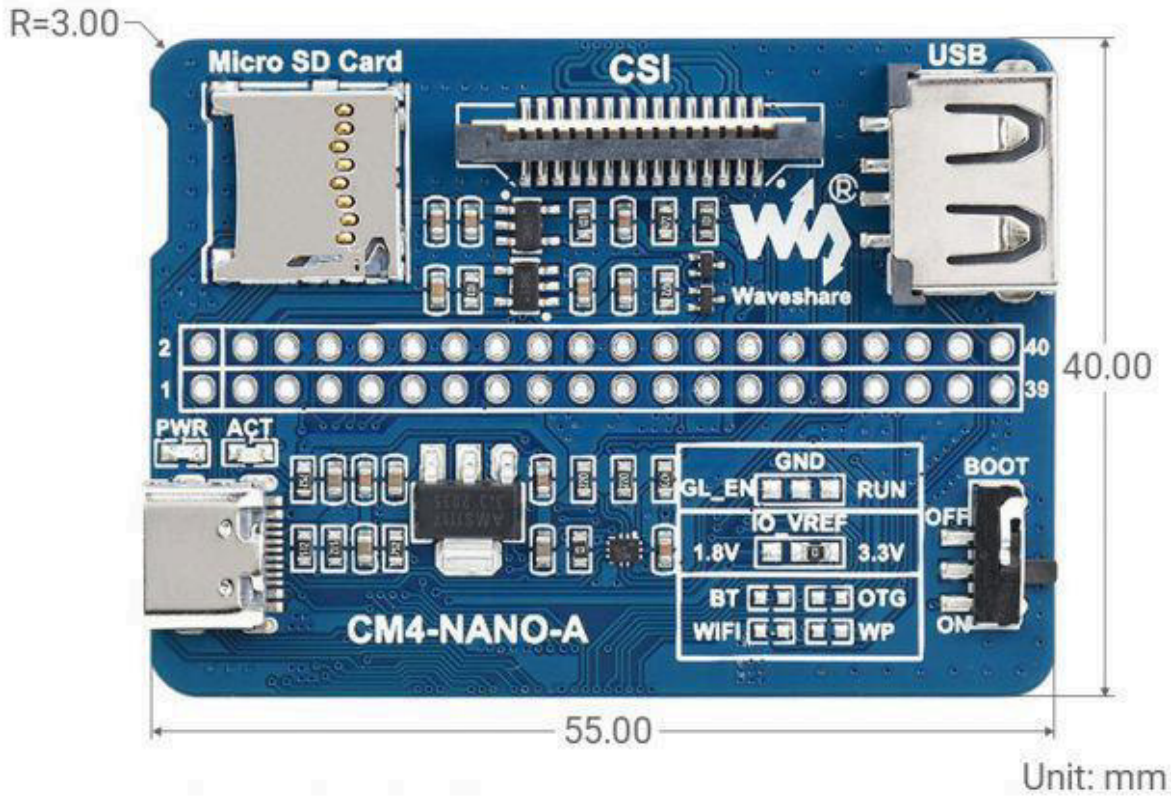
CM4-NANO-A is the mini board of Raspberry Pi Compute Module 4, which is a baseboard of Raspberry Pi Compute Module 4 with a 5V/2.5A USB Type C interface.

Precautions

1. DO NOT plug and unplug any device other than USB while it is powered on.
2. The Type C interface can be used as a power supply or as a USB SLAVE interface to flash the image.
3. In order to ensure the normal power supply of CM4, please do not connect other devices when using the Type C interface to flash the image.
4. When CM4 is in normal use, it needs to provide 5V 2A power supply for CM4. Otherwise, there may be problems such as shutdown, frequency reduction, and so on.
5. Since the module does not have any protection circuit, please do not short-circuit the power supply.

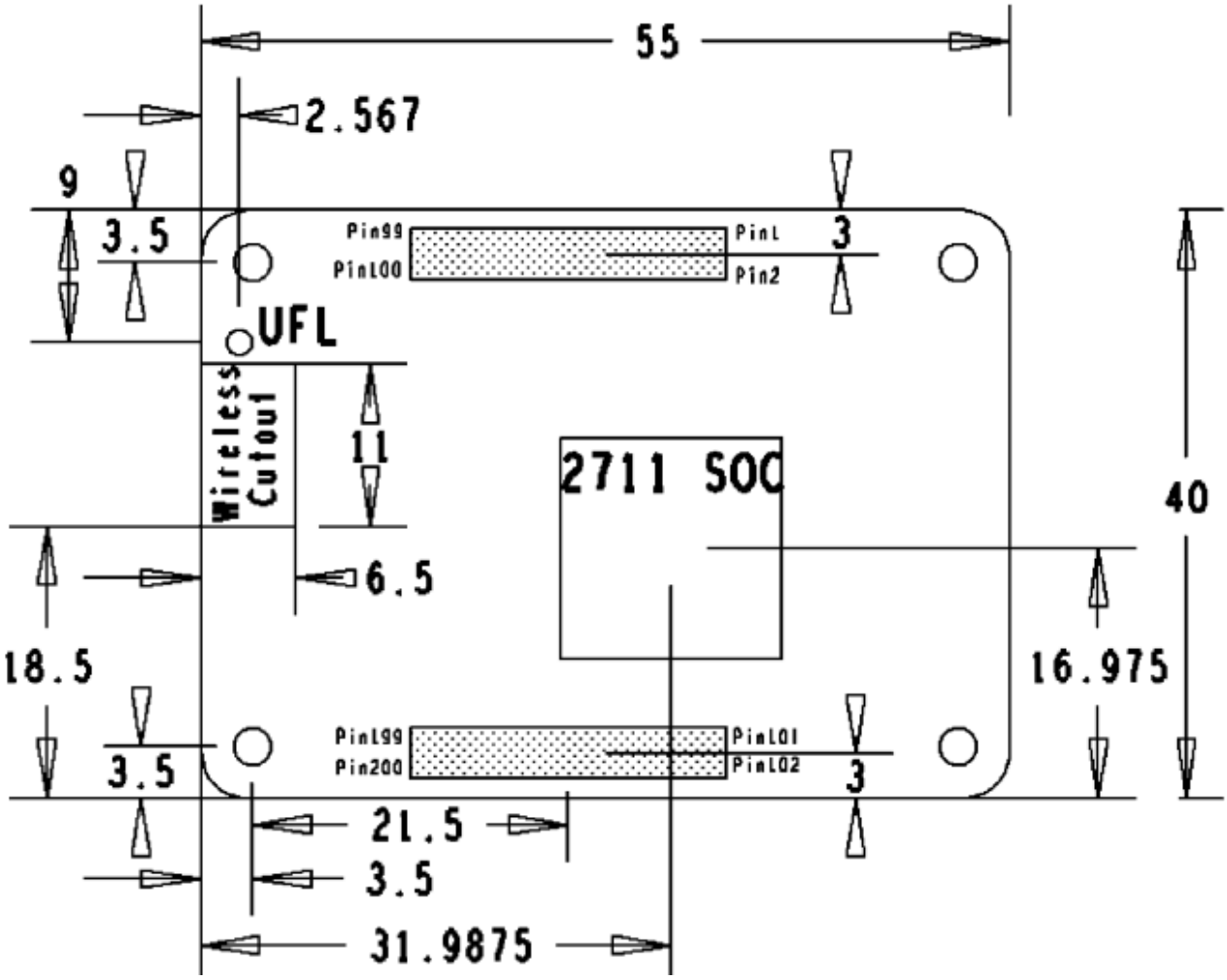
6. USB2.0 is disabled by default, if you want to open it, you need to add the line *dtoverlay=dwc2,dr_mode=host* to the config.txt file.

Dimension



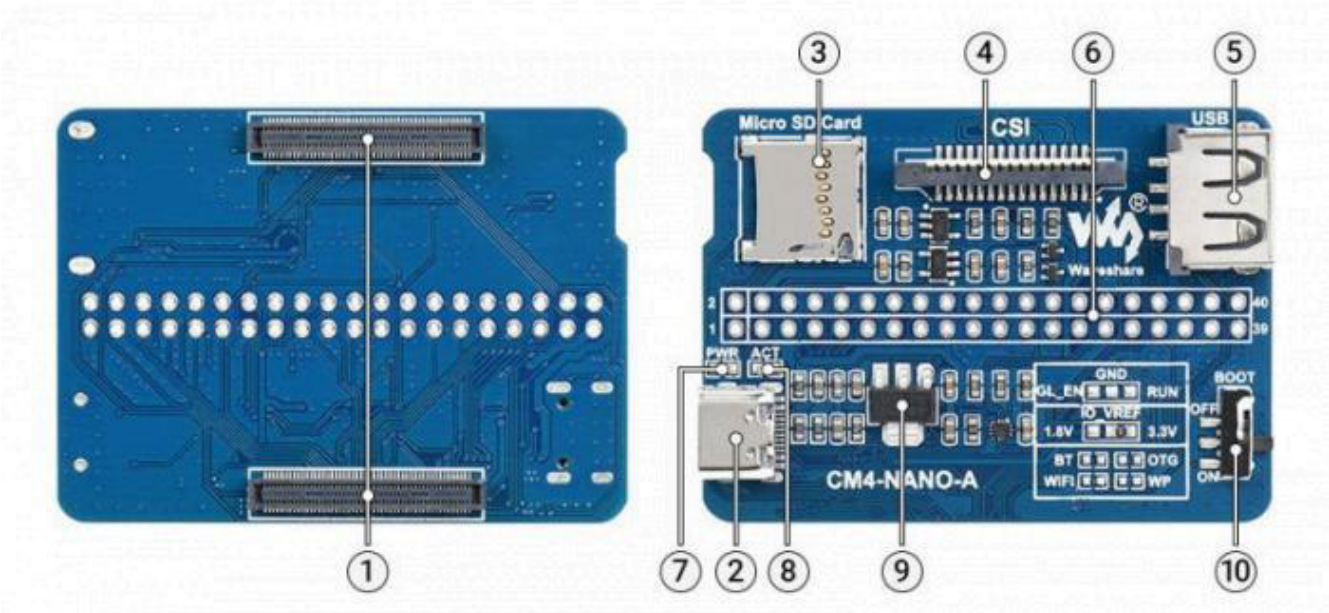
(/wiki/File:CM4-NANO-A-details-intro.jpg)

Compute_Module 4 Core board



(/wiki/File:Compute_Module_4_IO_Board_5.png)

Onboard Resources



(/wiki/File:CM4-NANO-A-details-size.jpg)

Label	Name	Description
1	CM4 socket	suitable for all variants of Compute Module 4
2	Power supply/flashing interface	5V/2A power supply, can also be used as an eMMC flashing interface

3	Micro SD card interface	For inserting a Micro SD card with the system, only for the Lite version
4	CSI Interface	Single MIPI CSI Camera Interface
5	USB 2.0 Interface	USB 2.0 interface, support various USB device insertion
6	40PIN GPIO Header	Easy access to various HAT modules
7	PWR Indicator	Indicates the power status of the Raspberry Pi
8	ACT Indicator	Indicates the working status of the Raspberry Pi
9	AMS1117-3.3V	Supply voltage for CSI and 40Pin
10	BOOT	ON: Switch the USB to type C interface, and enter the download mode when powered on (configured as a large-capacity disk through RPI boot), OFF: Switch the USB to TYPE A interface, it will not enter the download when powered on (start from eMMC or SD card)

Introduction

Precautions

Do not plug or unplug any device while it is powered on.

Writing Image

- Write Image for Compute Module Boards eMMC version (/wiki/Write_Image_for_Compute_Module_Boards_eMMC_version)
- Write Image for Compute Module Boards Lite version (/wiki/Wrote_Image_for_Compute_Module_Boards_Lite_version)

USB2.0

The USB port is disabled by default on the CM4 to save power. If you need to start, you need to add the following to the config.txt file:

```
dtoverlay=dwc2,dr_mode=host
```

After restarting:

If you use the latest Raspberry Pi OS (image after October 30, 2021) USB2.0 is OTG mode by default, CM4 will report an error:

```
config failed, hub doesn't have any ports! (err -19)
```

However, USB can still be used. If you want to remove this error, remove otg_mode=1 in [cm4] of config.txt, and add dtoverlay=dwc2, dr_mode=host (USB cannot be recognized without adding it).

```
[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
#otg_mode=1

dtoverlay=dwc2,dr_mode=host
```

(/wiki/File:CM4_Burn_EMMC_12.png)

CSI

Configuration

- Due to CSI and DSI being disabled by default, we need to load the device tree to enable, and I2C-10, I2C-11, and I2C-0 will be occupied when using the CSI camera and DSI screen.
- Enter the following command:

```
sudo apt-get install p7zip-full -y
wget https://files.waveshare.com/upload/4/41/CM4_dt_blob.7z
7z x CM4_dt_blob.7z -O./CM4_dt_blob
sudo chmod 777 -R CM4_dt_blob
cd CM4_dt_blob/
# If using two cameras and DSI1, execute
sudo  dtc -I dts -O dtb -o /boot/dt-blob.bin dt-blob-disp1-double_cam.dts
#When using any DSI, HDMI1 has no image output, even if you do not connect the DSI scree
n, as long as the corresponding file is compiled, then HDMI1 will not output.
#If you need to restore, delete the corresponding dt-blob.bin: sudo rm -rf /boot/dt-blob.
bin
# After execution, turn off the power and restart the CM4.
```

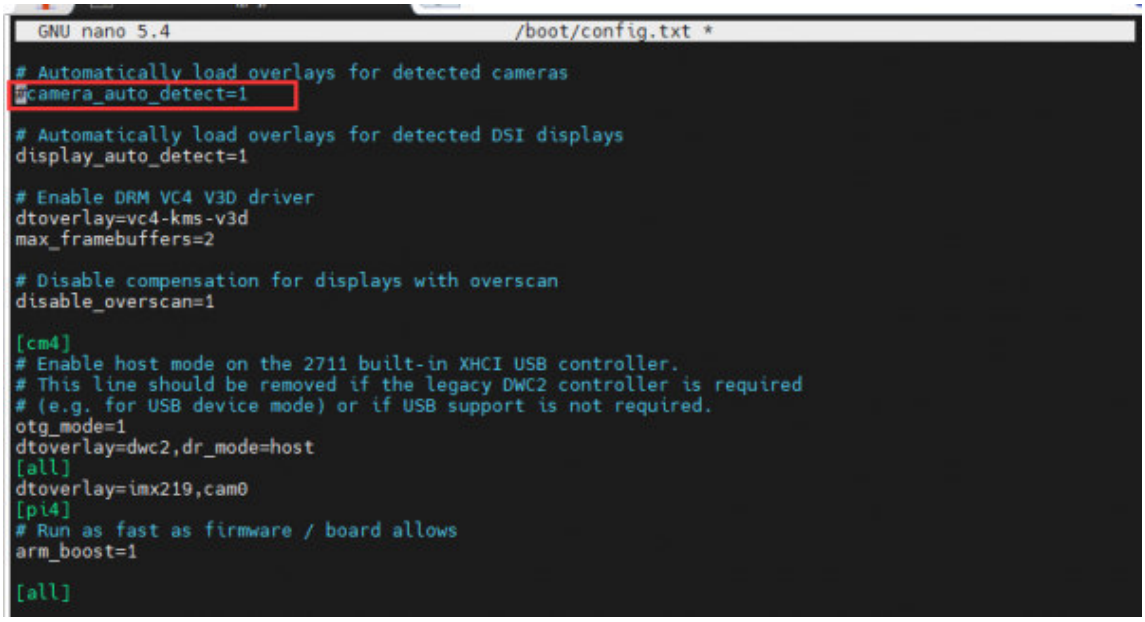
New Version System (Bullseye)

Camera Config

1. Execute the following commands to edit the "/boot/config.txt" file.

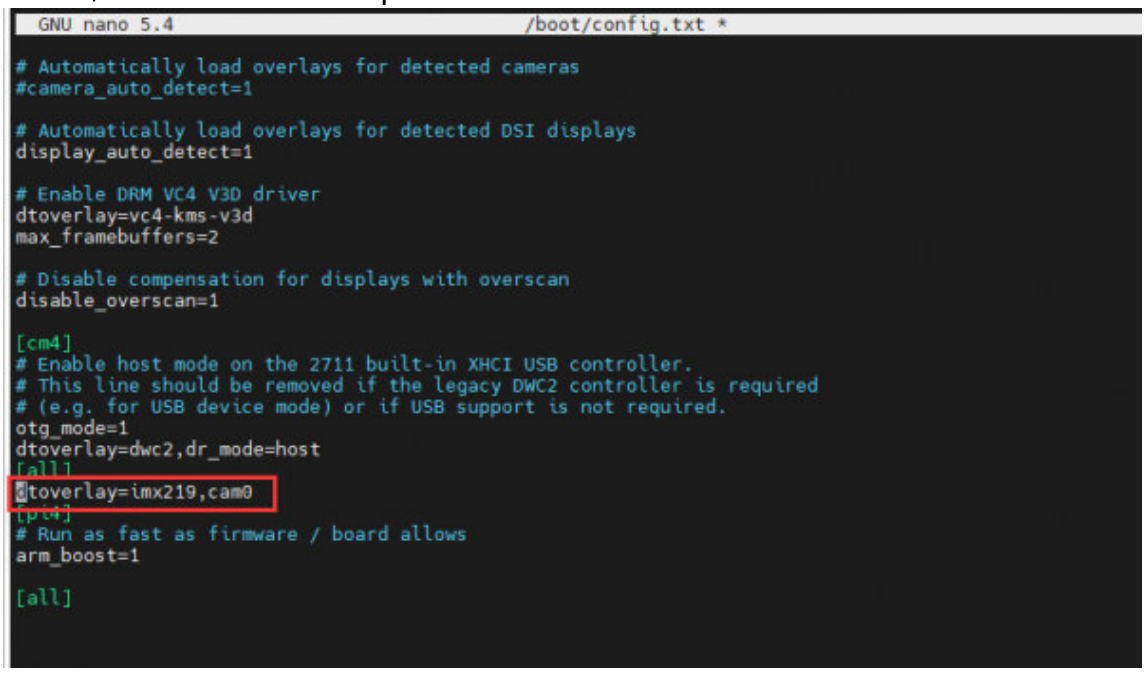
   ```
   sudo nano /boot/config.txt
   ```

2. Block or delete the camera auto-detect sentence.



(/wiki/File:CM4-NANO-B02.png)

3. Add the driver of the camera you used. Here I take IMX219 as an example, connect to CAM0, and attach the adapter.



(/wiki/File:CM4-NANO-B03.png)

Model	CAM0 Setting Sentence	CAM1 Setting Sentence
OV9281	dtoverlay=ov9281, cam0	dtoverlay=ov9281,cam1
IMX290/IMX327	dtoverlay=imx290, clock-frequency=37125000, cam0	dtoverlay=imx290, clock-frequency=37125000, cam1
IMX378	dtoverlay=imx378, cam0	dtoverlay=imx378, cam1
IMX219	dtoverlay=imx219, cam0	dtoverlay=imx219, cam1
IMX477	dtoverlay=imx477, cam0	dtoverlay=imx477, cam1

- If the camera you are using is the official Raspberry Pi camera and there only is one camera connected, you do not need to set up the config file.
- CM4-NANO only uses CAM0, so you only need to add "dtoverlay=imx219,cam0".

4. Ctrl+o to save the file and press Enter.

(/wiki/File:CM4-NANO-B04.png)

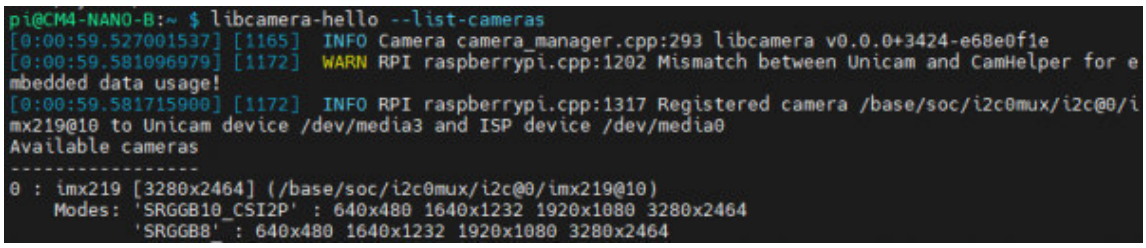
5. Ctrl+x to exit the editor.
6. Reboot CM4.

```
sudo reboot
```

Test Camera

1. Input the camera detecting commands, and you can see the camera is detected.

```
libcamera-hello --list-cameras
```



(/wiki/File:CM4-NANO-B05.png)

2. Display the camera pictures on the desktop:

```
libcamera-hello -t
```

3. Take a photo:

```
libcamera-jpeg -o test.jpg
```

4. Record a 10s video:

```
libcamera-vid -t 10000 -o test.h264
```

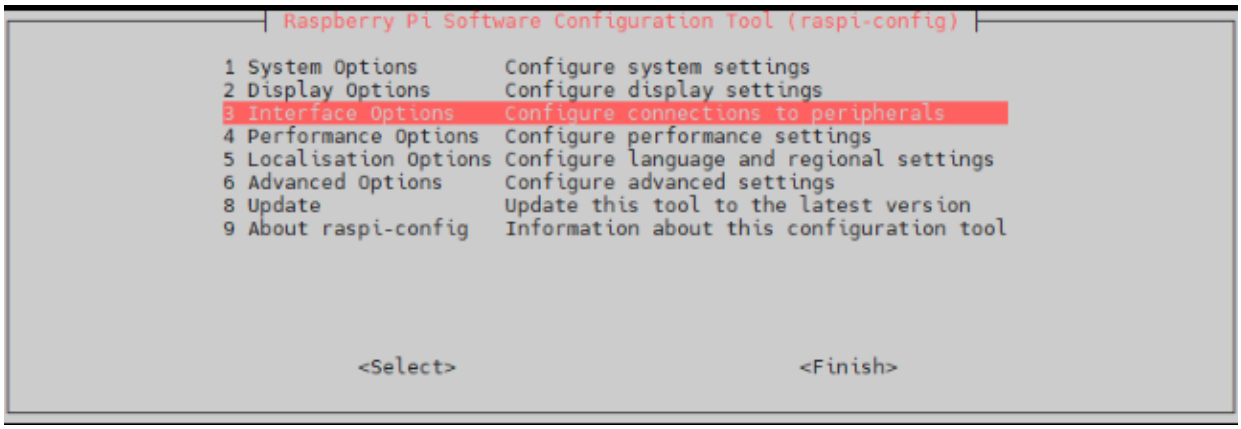
Old Version (Buster)

Configure Camera

1. Execute the following command to enter the Raspberry Pi configuration.

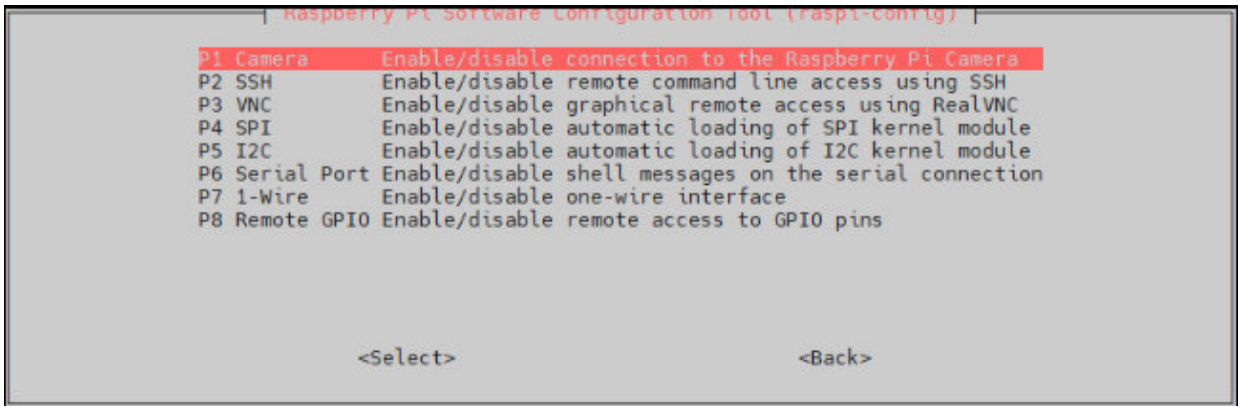
```
sudo raspi-config
```

2. Choose Interfacing Options and enter.



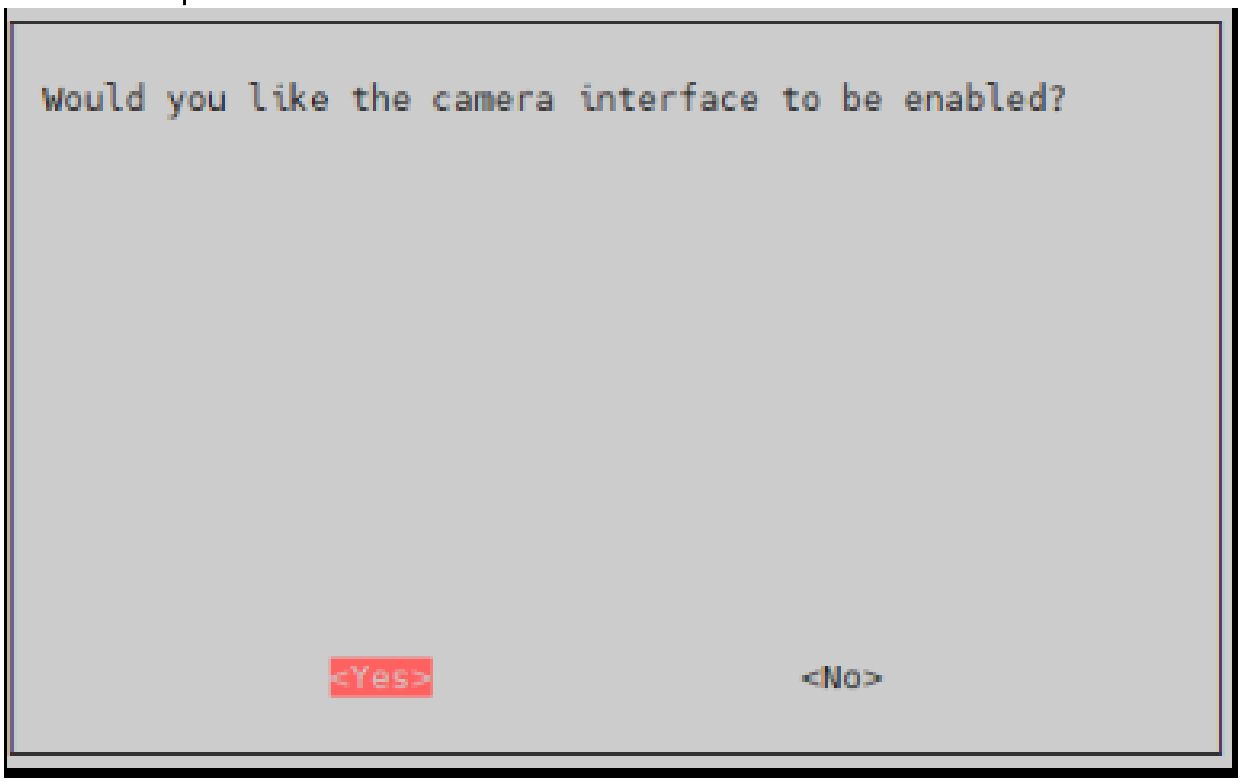
(/wiki/File:Interface.png)

3. Choose a camera.



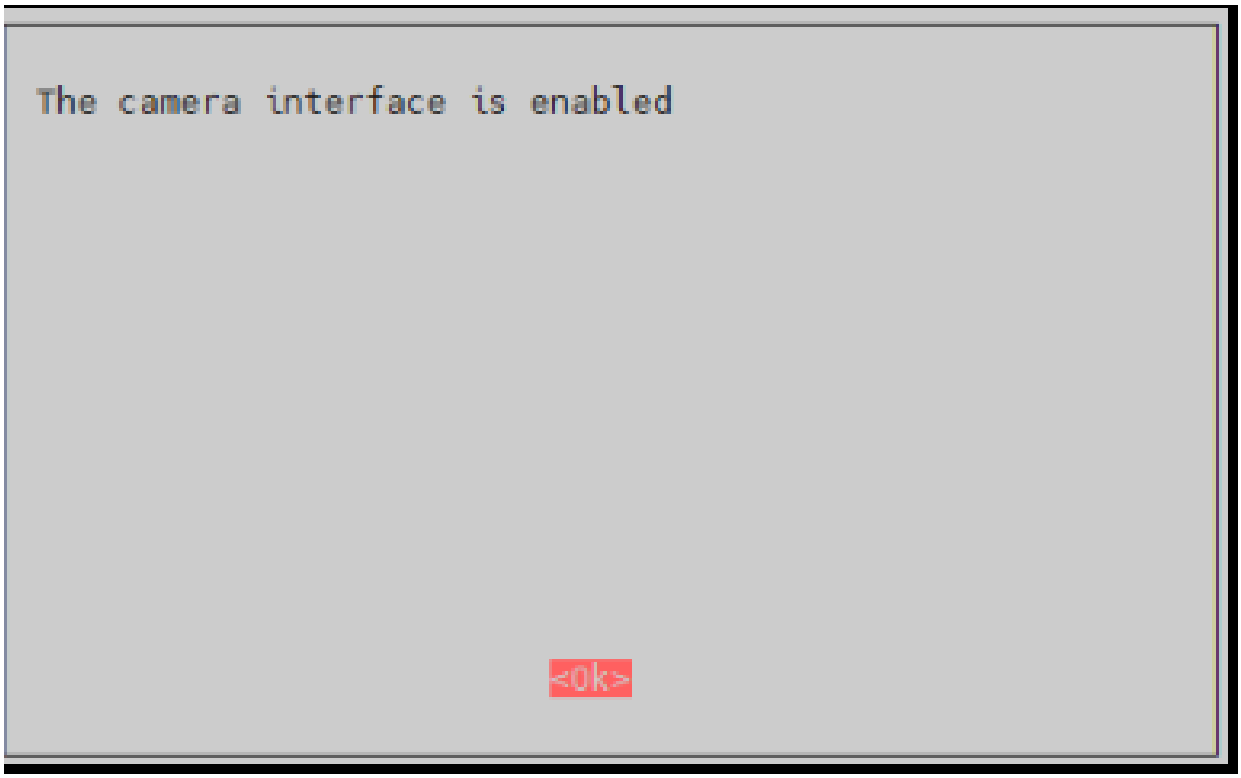
(/wiki/File:Camera.png)

4. Choose to open the camera interface.



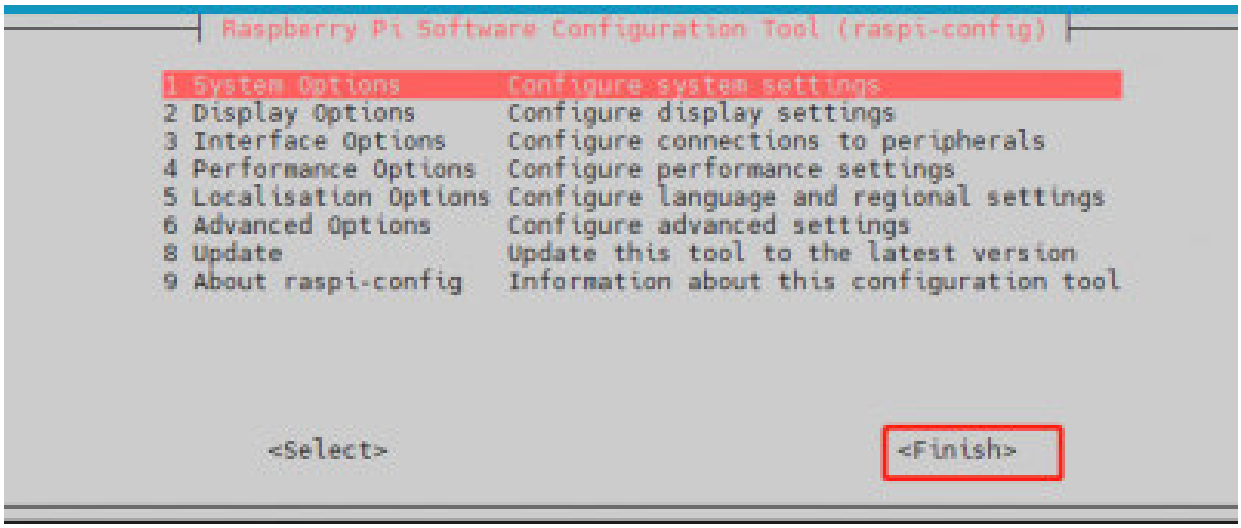
(/wiki/File:Camera2.png)

5. The system prompts as follows:



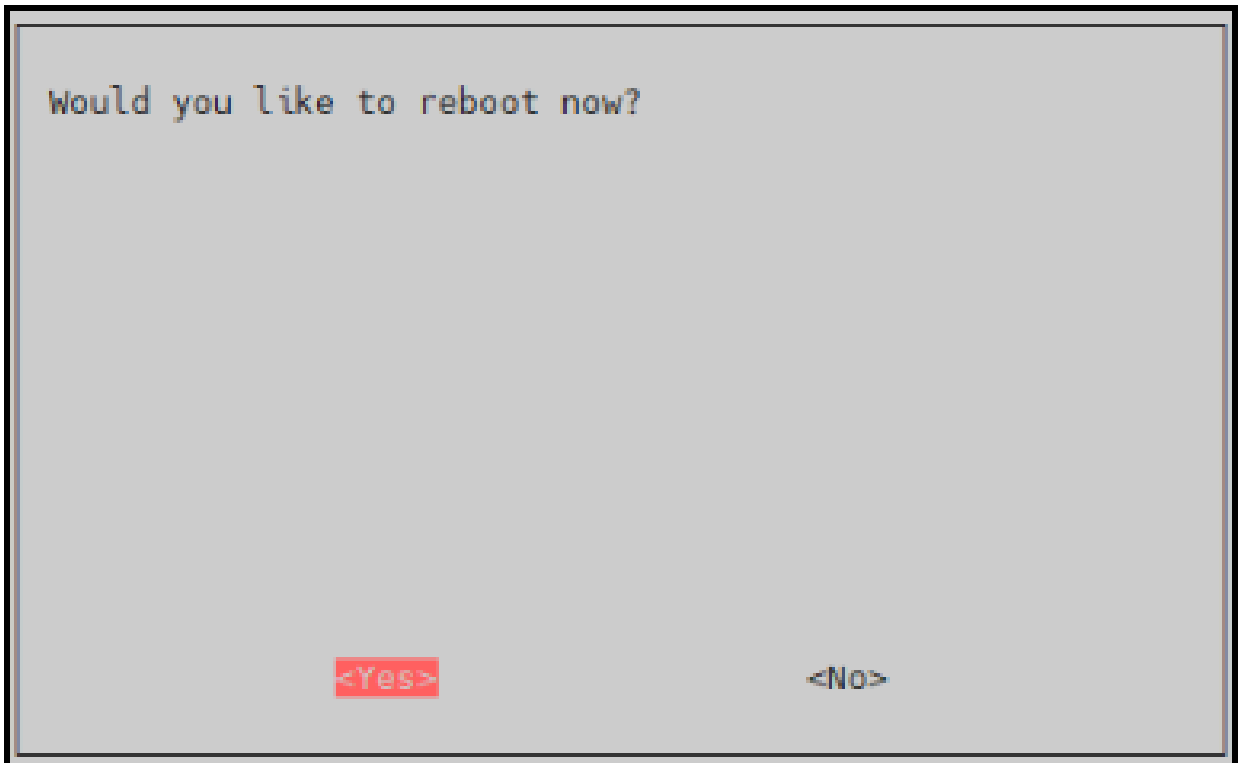
(/wiki/File:Prompt.png)

6. Back to the main interface, select Finish.



(/wiki/File:Finish.png)

7. Reboot the system.



(/wiki/File:Finish2.png)

Camera Test

- Take a photo:

```
raspistill -o image.jpg
```

- Test the recording function:

```
raspivid -o video.h264 -t 10000
```

- Where -t 10000 means recording for 10 seconds, users can adjust according to their own needs.
- Please refer to CSI (https://www.raspberrypi.com/documentation/computers/compute-module.html).

Resources

Official Manual

- Raspberry Pi Compute Module 4 IO Board Brief (https://datasheets.raspberrypi.org/cm4io/cm4io-product-brief.pdf)
- Raspberry Pi Compute Module 4 IO Board Datasheet (https://datasheets.raspberrypi.org/cm4io/cm4io-datasheet.pdf)

- CSI Camera Reference (https://www.raspberrypi.org/documentation/hardware/computemodule/cmio-camera.md)

Schematic

- Schematic (https://files.waveshare.com/upload/6/68/CM4-NANO-A_SchDoc.pdf)

3D document

- 3D Drawing (https://files.waveshare.com/upload/1/10/CM4-NANO-A-3D.zip)

Software

- Panasonic_SDFormatter-Formatting SD card (https://files.waveshare.com/upload/d/d7/Panasonic_SDFormatter.zip)
- Win32DiskImager (https://files.waveshare.com/upload/7/76/Win32DiskImager.zip)
- Putty (https://files.waveshare.com/upload/5/56/Putty.zip)
- RPiboot_Setup (https://files.waveshare.com/upload/f/f3/Rpiboot_setup.zip)

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now (https://service.waveshare.com/)

TF-Luna LiDAR module (Short-range distance sensor)

Product specification SJ-GU-TF-Luna A05

北醒 Benewake

TF-Luna is a single-point ranging LiDAR, based on ToF principle. Mainly used for stable, accuracy and high-frame rate range detection.

The product is built with algorithms adapted to various application environments and adopts multiple adjustable configurations and parameters so as to offer excellent distance measurement performances in complex application fields and scenarios.

Main product features

- Small size
- Light weight
- Low power consumption
- Low cost

Main application scenarios

- Auxiliary focus
- Elevator projection
- Intrusion detection
- Level measurement

SPECIFICATIONS

Description		Parameter value
Product performance	Operating range	0.2m~8m(90%reflectivity indoor 0klux)[1] 0.2m~2.5m(10%reflectivity indoor 0klux) [2] 0.2m~8m(90%reflectivity outdoor 90klux) 0.2m~2.5m(10%reflectivity outdoor 90klux)
	Accuracy	±6cm@(0.2m-3m)[3] ±2%@(3m-8m)
	Distance resolution	1cm
	Frame rate	1-250Hz[4]
	Ambient light immunity	70Klux
	Operation temperature	-10℃~60℃
	Enclose rating	/
Optical parameters	Light source	VCSEL
	Central wavelength	850nm
	Photobiological safety	Class1 (IEC60825)
	FOV	2°[5]
Electrical parameters	Supply voltage	3.7V-5.2V
	Average current	≤70mA

[1] Range based on the indoor test with the standard white board (90% reflectivity) at 25℃ as the detection object;

[2] Range based on the indoor test with the standard white board (90% reflectivity) at 25℃ as the detection object;

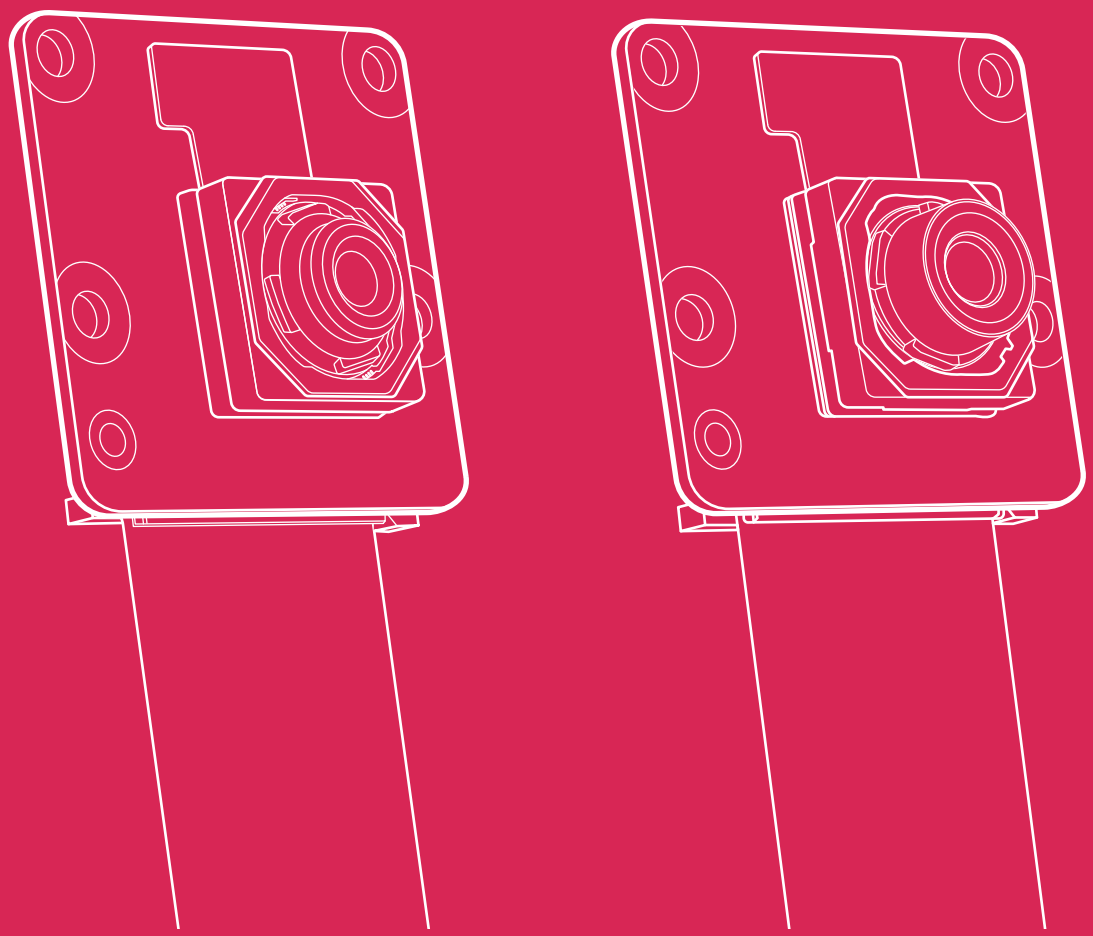
[3] Accuracy based on the indoor test with the standard white board (90% reflectivity) at 25℃ as the detection object;

[4] The Highest frame rate is 250Hz, the default frame rate is 100Hz. The customized update rate should be calculated by the formula: 500/n (n is more than 2),;

[5] This is a theoretical reference value.

Raspberry Pi Camera Module 3
Standard
NoIR
Wide
NoIR Wide

Published June 2024



Raspberry Pi Ltd

Raspberry Pi Compute Module 4

A Raspberry Pi for deeply embedded applications

Colophon

build-date: 2023-11-24
build-version: githash: 3ee4166-dirty

Legal disclaimer notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (“RESOURCES”) ARE PROVIDED BY RASPBERRY PI LTD (“RPL”) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage (“High Risk Activities”). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL’s Standard Terms. RPL’s provision of the RESOURCES does not expand or otherwise modify RPL’s Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Table of contents

Chapter 1. Introduction

1.1. Introduction

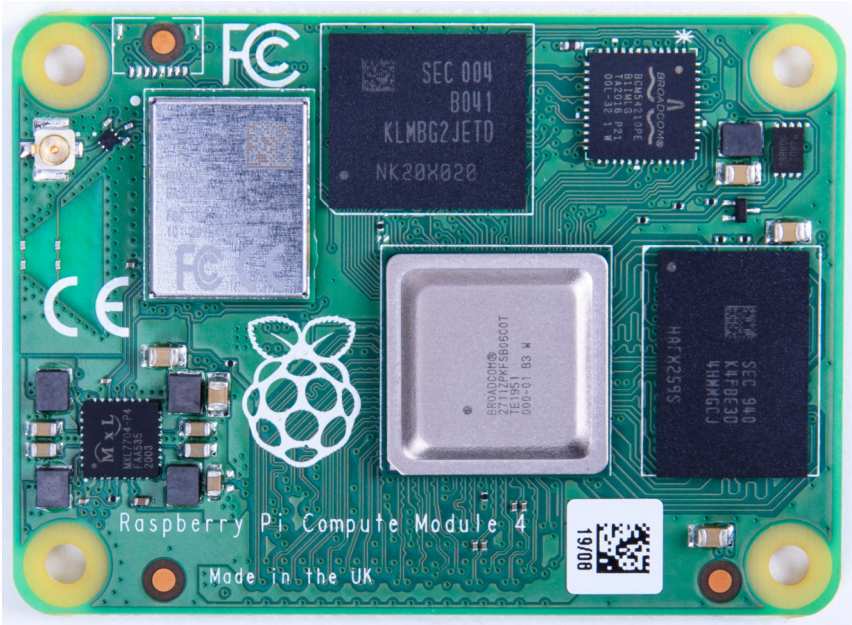


Figure 1. The Raspberry Pi Compute Module 4 (CM4).

The Raspberry Pi Compute Module 4 (CM4) is a System on Module (SoM) containing processor, memory, eMMC Flash, and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition, these modules have extra IO interfaces over and above what is available on the Raspberry Pi boards, opening up more options for the designer.

The design of the CM4 is loosely based on the Raspberry Pi 4 Model B, and for cost-sensitive applications it can be supplied without the eMMC fitted; this version is called the Raspberry Pi Compute Module 4 Lite (CM4Lite).

While previous generations of the Compute Module have all shared the same DDR2-SODIMM-mechanically-compatible form factor, the new CM4 and CM4Lite are different. The electrical interface of the CM4 is via two 100-pin high density connectors, and the new physical form factor has a smaller footprint overall when the connectors are taken into account.

This change is due to the addition of new interfaces: an additional second HDMI, PCIe, and Ethernet. The addition of these new interfaces, especially PCIe, would not have been possible while preserving the previous form factor.

NOTE

Unless otherwise stated, for this document CM4 also refers to CM4Lite.

1.2. Features

Key features of the CM4 are as follows:

- Broadcom BCM2711, quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Small Footprint 55mm × 40mm × 4.7mm module
  - 4 × M2.5 mounting holes

- H.265 (HEVC) (upto 4Kp60 decode), H.264 (upto 1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM with ECC (see Appendix B)
- Options for 0GB **(CM4Lite)**, 8GB, 16GB, or 32GB eMMC flash memory (see Appendix B)
  - Peak eMMC bandwidth 100MBps (four times faster than previous Compute Modules)
- Option (see Appendix B) for certified radio module with:
  - 2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless
  - Bluetooth 5.0, BLE
  - On board electronic switch to select between PCB trace or external antenna
- Gigabit Ethernet PHY supporting IEEE 1588
- 1 × PCIe 1-lane Host, Gen 2 (5Gbps)
- 1 × USB 2.0 port (high speed)
- 28 × GPIO supporting either 1.8V or 3.3V signalling and peripheral options:
  - Up to 5 × UART
  - Up to 5 × I2C
  - Up to 5 × SPI
  - 1 × SDIO interface
  - 1 × DPI (parallel RGB display)
  - 1 × PCM
  - Up to 2× PWM channels
  - Up to 3× GPCLK outputs
- 2 × HDMI 2.0 ports (up to 4Kp60 supported)
- MIPI DSI:
  - 1 × 2-lane MIPI DSI display port
  - 1 × 4-lane MIPI DSI display port
- MIPI CSI-2:
  - 1 × 2-lane MIPI CSI camera port
  - 1 × 4-lane MIPI CSI camera port
- 1 × SDIO 2.0 **(CM4Lite)**
- Single +5V PSU input.

Chapter 2. Interfaces

2.1. Wireless

The CM4 can be supplied with an on-board wireless module based on the Cypress CYW43455 supporting both:

- 2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless
- Bluetooth 5.0, BLE

These wireless interfaces can be individually enabled or disabled as required. For instance, in the case of a kiosk application, a service engineer could enable wireless operation and then disable it once finished.

The CM4 has an on-board antenna. If used it should be positioned in the product such that it is not surrounded by metal, including any ground plane (see Chapter 3 for further details). Alternatively there is a standard U.FL connector on the module, see Figure 1, so that an external antenna can be used.

Raspberry Pi Ltd has an antenna kit which is certified to be used with the CM4. If a different antenna is used then separate certification will be required.

WARNING

Raspberry Pi Ltd will not be able to assist with certification for third-party antennas.

The selection of internal or external antenna is done at boot time using the `config.txt` file, and can not be changed during operation. The `config.txt` options are `dtparam=ant1` to select the internal antenna, or `dtparam=ant2` for the external antenna.

2.1.1. `WL_nDisable`

This pin serves a number of functions;

1. It can be used to monitor the enable/disable state of wireless networking. A logic high means the wireless networking module is powered up.
2. When driven or tied low it prevents the wireless network module from powering up. This is useful to reduce power consumption or in applications where it is required to physically ensure the wireless networking is disabled. If the interface is enabled after being disabled, the wireless interface driver needs reinitialised.

NOTE

On CM4 modules without wireless, this pin is reserved.

2.1.2. `BT_nDisable`

This pin serves a number of functions;

1. It can be used to monitor the enable/disable state of Bluetooth. A logic high means the Bluetooth module is powered up.
2. When driven, or tied low, it prevents the Bluetooth module from powering up. This is useful to reduce power consumption, or in applications where it is required to physically ensure the Bluetooth is disabled. If the interface is enabled after being disabled, the Bluetooth interface driver needs reinitialised.

NOTE

On CM4 modules without wireless, this pin is reserved.

2.2. Ethernet

The CM4 has an on-board Gigabit Ethernet PHY — the Broadcom BCM54210PE — some of the major features of this PHY include;

- IEEE 1588-2008 compliant
- MDI crossover, pair skew and pair polarity correction

A standard 1:1 RJ45 MagJack is all that is necessary to provide an Ethernet connection to the CM4. Typical wiring of a MagJack supporting PoE, and with added ESD protection, can be seen in Figure 2.

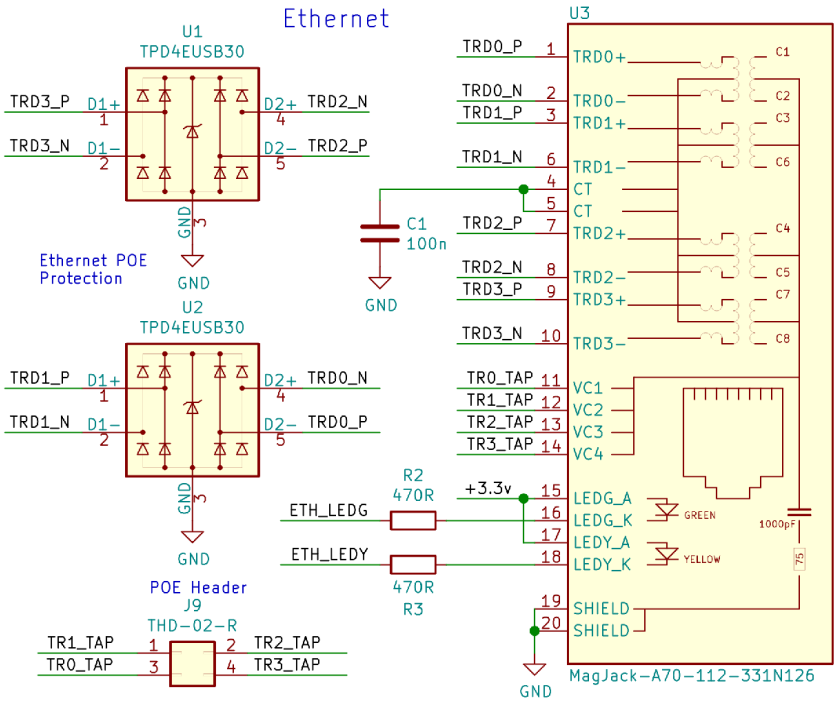


Figure 2. Ethernet schematic interface for the Raspberry Pi Compute Module 4 supporting PoE, and with added ESD protection.

The differential Ethernet signals should be routed as 100Ω differential pairs, with suitable clearances. Length matching between pairs should be better than 50mm, so in the typical case no length matching is required. However the signals within a pair need to be length matched, ideally to better than 0.15mm.

The PHY also supports up to 3 LEDs to give user status feedback, these are low active. These LEDs can have a range of functions, and you should consult your OS driver to see which functions are supported by your driver.

The PHY also provides `SYNC_IN` and `SYNC_OUT` at 3.3V signalling to support IEEE 1588-2008.

2.3. PCIe (Gen2 x1)

The CM4 has an internal PCIe 2.0 x1 host controller. While on the Raspberry Pi 4 Model B this has been connected to a USB 3 host controller (using the Via Labs VLI805), on the CM4 the product designer is free to choose how the interface is used.

WARNING

You should ensure that there is a suitable OS driver for any host controller that is chosen before proceeding to a prototype.

NOTE

The on-board PCIe Host controller doesn't support 64-bit accesses from the ARM, they must be split up into two 32-bit accesses.

Connecting a PCIe device follows the standard PCIe convention. The CM4 has on-board AC coupling capacitors for `CLK` and `PCIe_TX` signals. However the `PCIe_RX` signals need external coupling capacitors close to the driving source (the device `TX`), if you are using an external PCIe/NVMe card these capacitors will be on-board. The PCIe convention is that if you are wiring directly to an IC then the TX and RX pairs need to be swapped (i.e. TX → RX, RX → TX). If you are wiring to a connector then this is typically labelled from the host point of view and so TX/RX swaps aren't required. Additionally the `PCIe_CLK_nREQ` must be connected to ensure the CM4 produces a clock signal, and the `PCIe_nRST` should also be connected to ensure the device is correctly reset when required.

The differential PCIe signals should be routed as 90Ω differential pairs, with suitable clearances. There is no need to match the lengths between pairs, only the signals within a Pair need to be length matched ideally to better than 0.1mm.

TIP

5.10 kernels and newer have had support for MSI-X added. There is a limit of upto 32 IRQs available. If the device has problems with interrupts then adding `pci=nomsi` to `cmdline.txt` (and rebooting) often fixes the issue.

2.4. USB 2.0 (high speed)

The USB 2.0 interface supports up to 480Mbps signalling. The differential pair should be routed as a 90Ω differential pair. The length of the P/N signals should ideally be matched to better than 0.15mm.

TIP

The firmware disables the USB interface by default to save power. In recent versions of Raspberry Pi OS (Bullseye) it is automatically enabled by the `otg_mode=1` setting in the config.txt file. If you are using a different OS, or an older version of Raspberry Pi OS, you will need to add this to config.txt to enable the USB interface.

NOTE

The port is capable of being used as a true USB On-The-Go (OTG) port. While there is no official documentation, some users have had success making this work. The `USB_OTG_ID` pin is used to select between USB host and device that is typically wired to the ID pin of a Micro USB connector. To use this functionality it must be enabled in the OS. If using either as a fixed slave or fixed master, please tie the `USB_OTG_ID` pin to ground.

2.5. GPIO

There are 28 pins available for general purpose I/O (GPIO), which correspond to the GPIO pins on the Raspberry Pi 4 Model B 40-pin header. These pins have access to internal peripherals: SMI, DPI, I2C, PWM, SPI, and UART. The BCM2711 ARM peripherals book describes these features in detail, along with the multiplexing options available. The drive strength and slew rate should ideally be set as low as possible to reduce any EMC issues. GPIO2 and GPIO3 have 1.8kΩ pull up resistors.

The BCM2711 GPIO bank is powered by `GPIO_VREF`, this can either be connected to +1.8V for 1.8V signalling GPIO, or

+3.3V for 3.3V signalling. You should keep the load on the 28 GPIO pins to below 50mA in total. `GPIO_VREF` must be powered for the CM4 to start up correctly.

2.5.1. Alternative function assignments

Up to six alternative functions are available. The BCM2711 ARM peripherals book describes these features in detail. The table below gives a quick overview.

Table 1. GPIO pins alternative function assignment

GPIO	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
GPIO1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
GPIO2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
GPIO3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
GPIO4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
GPIO5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
GPIO6	High	GPCLK2	SOE_N / SE	DPI_D2	SPI4_MOSI	CTS3	SDA4
GPIO7	High	SPI0_CE1_N	SWE_N / SRW_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
GPIO8	High	SPI0_CE0_N	SD0	DPI_D4	BSCSL / CE_N	TXD4	SDA4
GPIO9	Low	SPI0_MISO	SD1	DPI_D5	BSCSL / MISO	RXD4	SCL4
GPIO10	Low	SPI0_MOSI	SD2	DPI_D6	BSCSL SDA / MOSI	CTS4	SDA5
GPIO11	Low	SPI0_SCLK	SD3	DPI_D7	BSCSL SCL / SCLK	RTS4	SCL5
GPIO12	Low	PWM0_0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
GPIO13	Low	PWM0_1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
GPIO14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
GPIO15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
GPIO16	Low	<reserved>	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
GPIO17	Low	<reserved>	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
GPIO18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0_0
GPIO19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM0_1
GPIO20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
GPIO21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
GPIO22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
GPIO23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
GPIO24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
GPIO25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
GPIO26	Low	SD0_DAT2	<reserved>	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
GPIO27	Low	SD0_DAT3	<reserved>	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

GPIO	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO44	-	GPCLK1	SDA0	SDA1	<reserved>	SPI0_CE1_N	SD_CARD_VOLT
GPIO45	-	PWM0_1	SCL0	SCL1	<reserved>	SPI0_CE2_N	SD_CARD_PWR0

Special function legend:

Table 2. GPIO pins alternative function legend

Name	Function
SDA0	BSC master 0 data line[a]
SCL0	BSC master 0 clock line
SDAx	BSC master 1,3,4,5,6 data line[b]
SCLx	BSC master 1,3,4,5,6 clock line
GPCLKx	General purpose clock 0,1,2
SPIx_CE2_N	SPI 0,3,4,5,6 chip select 2
SPIx_CE1_N	SPI 0,3,4,5,6 chip select 1
SPIx_CE0_N	SPI 0,3,4,5,6 chip select 0
SPIx_MISO	SPI 0,3,4,5,6 MISO
SPIx_MOSI	SPI 0,3,4,5,6 MOSI
SPIx_SCLK	SPI 0,3,4,5,6 serial clock
PWMx_0	PWM 0,1 channel 0
PWMx_1	PWM 0,1 channel 1
TXDx	UART 0,2,3,4,5 transmit data
RXDx	UART 0,2,3,4,5 receive data
CTSx	UART 0,2,3,4,5 clear to send
RTSx	UART 0,2,3,4,5 request to send
PCM_CLK	PCM clock
PCM_FS	PCM frame sync
PCM_DIN	PCM data in
PCM_DOUT	PCM data out
SAx	Secondary mem address bus
SOE_N / SE	Secondary mem controls
SWE_N / SRW_N	Secondary mem controls
SDx	Secondary mem data bus
BSCSL SDA / MOSI	BSC slave data, SPI slave MOSI
BSCSL SCL / SCLK	BSC slave clock, SPI slave clock
BSCSL - / MISO	BSC <not used>, SPI MISO
BSCSL - / CE_N	BSC <not used>, SPI CSn
SPI1_CE2_N	SPI 1 chip select 2 [c]

Name	Function
SPI1_CE1_N	SPI 1 chip select 1
SPI1_CE0_N	SPI 1 chip select 0
SPI1_MISO	SPI 1 MISO
SPI1_MOSI	SPI 1 MOSI
SPI1_SCLK	SPI 1 serial clock
TXD1	UART 1 transmit data
RXD1	UART 1 receive data
CTS1	UART 1 clear to send
RTS1	UART 1 request To send
ARM_TRST	ARM JTAG reset
ARM_RTCK	ARM JTAG return clock
ARM_TDO	ARM JTAG data out
ARM_TCK	ARM JTAG clock
ARM_TDI	ARM JTAG data in
ARM_TMS	ARM JTAG mode select
PCLK	Display parallel interface
DE	Display parallel interface
LCD_VSYNC	Display parallel interface
LCD_HSYNC	Display parallel interface
DPI_Dx	Display parallel interface

[a] The Broadcom serial control bus is a proprietary bus compliant with the Philips® I2C bus/interface.
[b] BSC master 2 & 7 are not user-accessible.
[c] SPI 2 is not user-accessible.

2.6. Dual HDMI 2.0

The CM4 supports two HDMI 2.0 interfaces, each one capable of driving 4K images. If both HDMI outputs are used then each can be driven upto 4Kp30, however if only HDMI0 interface is being used then images up to 4Kp60 are possible.

HDMI signals should be routed as 100Ω differential pairs. Each signal within a pair should ideally be matched to better than 0.15mm. Pairs don't typically need any extra matching, as they only have to be matched to 25mm.

CEC is also supported; an internal 27kΩ pullup resistor is included in the CM4.

Basic on-board ESD protection is provided for the I2C EDID signals and the CEC signals; internal pullup and pulldown resistors are also provided. On the Raspberry Pi 4 Model B the HDMI signals don't have any extra ESD protection. Depending on the application, extra ESD protection may be required.

2.7. CSI-2 (MIPI serial camera)

The CM4 supports two camera ports: `CAM0` (2 lanes) and `CAM1` (4 lanes). CSI signals should be routed as 100Ω differential pairs. Each signal within a pair should ideally be matched to better than 0.15mm.

The documentation around the CSI interface can be found on the Raspberry Pi website, while Linux kernel drivers can be found on GitHub.

NOTE

The official Raspberry Pi firmware supports the OmniVision OV5647, Sony IMX219, Sony IMX296, Sony IMX477 and Sony IMX708 camera sensors. No security device is required on Compute Module devices in order to use these camera sensors.

2.8. DSI (MIPI serial display)

The CM4 supports two display ports: `DISP0` (2 lanes) and `DISP1` (4 lanes). Each lane supports a maximum data rate per lane of 1Gbps.

Although Linux kernel drivers are available, the DSI interface is not currently documented. Only DSI displays supported by the official Raspberry Pi firmware are supported. DSI signals should be routed as 100Ω differential pairs; each signal within a pair should ideally be matched to better than 0.15mm.

NOTE

While only official DSI displays are supported, other displays can be added using the parallel DPI interface which is available as a GPIO alternative function. The CM4 supports up to three displays of any type (HDMI, DSI, DPI) at any one time.

2.9. I2C (SDA0 SCL0)

This internal I2C bus is normally allocated to the CSI1 and DSI1, as these devices are controlled by the firmware. It can be used as a general I2C bus if the CSI1 ad DSI1 interfaces aren't being used, or are being controlled by the firmware. For example libcamera runs on the ARM and doesn't use the firmware, so in this case you may use CSI1 and this I2C bus. SDA0 is connected to GPIO44 on the BCM2711 and SCL0 is connected to GPIO45.

2.10. I2C (ID_SD ID_SC)

This I2C bus is normally used for identifying HATs and controlling CSI0 and DSI0 devices. If the firmware isn't using the I2C bus e.g. CSI0 and DSI0 aren't being used then these pins may be used as GPIO 0 and GPIO 1 if required.

NOTE

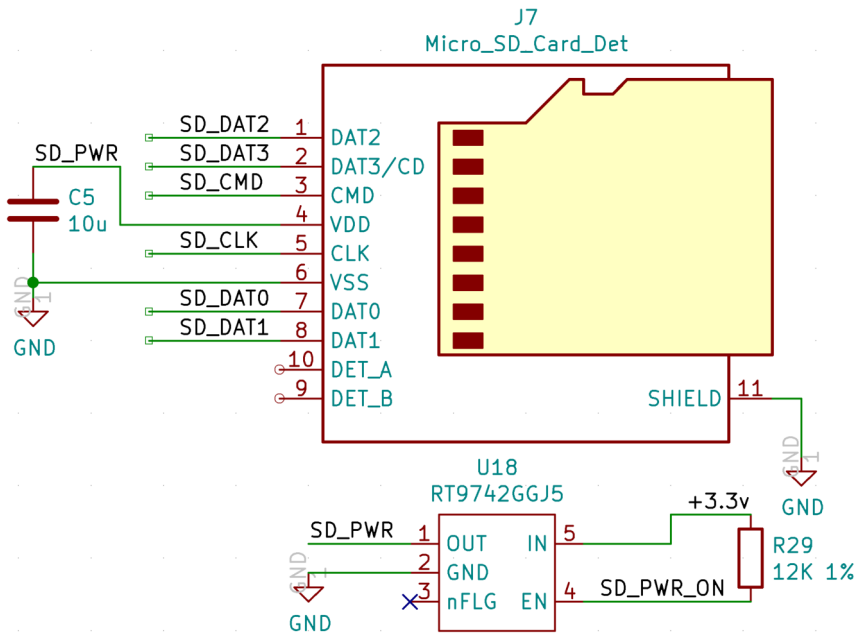
If these pins are used as GPIO pins, then to prevent the firmware from checking to see if there is a HAT EEPROM available, add `force_eeprom_read=0` and `disable_poe_fan=1` to the config.txt file.

2.11. SDIO/eMMC (CM4Lite only)

The CM4Lite does not have on-board eMMC. The eMMC signals are available on the connector so that an external eMMC or SD card can be used.

The SD_PWR_ON signal is used to enable an external power-switch to turn on power to the SD card; for eMMC it typically isn't used. If booting from SD card is required, then a pullup resistor must also be fitted to default the power-switch to be on. When SD_VDD_OVERRIDE is high (3.3V), this forces 1.8V signalling on the SDIO interface. Typically this is used with eMMC memory.

Figure 3. CM4Lite SD card interface.



2.12. Analog IP0/IP1

These are the two spare inputs on the MXL7704. The MXL7704 datasheet should be consulted if these pins are to be used. On-board filtering is provided by a 100nF capacitor to ground for each signal. On the Raspberry Pi 4 Model B these are connected to the USB C connector CC1 and CC2 pins.

2.13. Global_EN

Pulling this pin low puts the CM4 in the lowest possible power-down state. After software shutdown, Global_EN needs to be pulled low for > 1ms to restart the power system on the CM4.

TIP

It is recommended to only pull this pin low once the OS has shut down.

2.14. RUN_PG

This pin when high signals that the CM4 has started. Driving this pin low resets the module. This should be done with caution; if files on a filesystem are open they will not be closed.

2.15. nRPI_BOOT

During boot if this pin is low, booting from eMMC will be stopped and booting will be transferred to rpi boot which is via USB.

2.16. LED_nACT

This pin is designed to drive an LED to replicate the green LED on the Raspberry Pi 4 Model B. Under Linux this pin will flash to signify eMMC access. If any error occurs during booting, then this LED will flash an error pattern which can be decoded using the look up table on the Raspberry Pi website.

2.17. LED_nPWR

This pin needs to be buffered to drive an LED. The signal is designed to replicate the red power LED on the Raspberry Pi 4 Model B.

2.18. EEPROM_nWP

It is recommended that final products pull this pin low to prevent the end users changing the contents of the on-board EEPROM. See the Raspberry Pi 4 Model B documentation for instructions on the software settings required to support EEPROM write protection.

Chapter 3. Electrical and mechanical

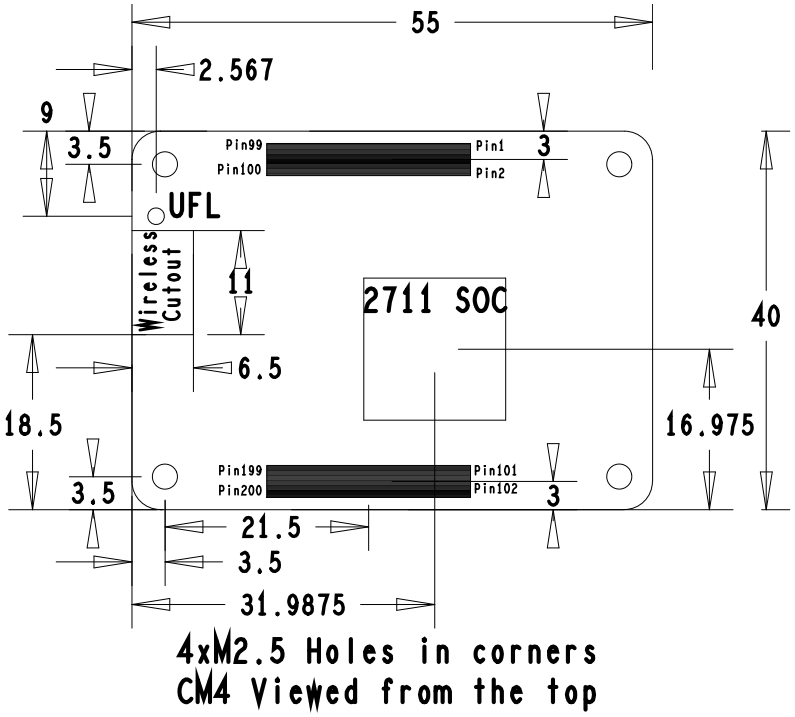
3.1. Mechanical

The CM4 is a compact 40mm × 55mm module. The Module is 4.7mm deep, but when connected the height will be 5.078mm or 6.578mm depending on the stacking height chosen.

1. 4 × M2.5 mounting holes (inset 3.5mm from module edge)
2. PCB thickness 1.2mm ± 10%
3. BCM2711 SoC height including solder balls 2.378 ± 0.11mm
4. Stacking height either:
    a. 1.5mm with mating connector (clearance under CM4 0mm): DF40C-100DS-0.4v
    b. 3.0mm with mating connector (clearance under CM4 1.5mm): DF40HC(3.0)-100DS-0.4v

If the on-board wireless antenna is used (see Section 2.1) it must be orientated towards the edge of the plastic enclosure and any nearby metal must have cut-outs or the wireless performance will be degraded. It is suggested that there is at least 10mm clearance around the PCB antenna, but the designer must check the performance.

Figure 4. Mechanical specification of the Raspberry Pi Compute Module 4



There must not be any metal, including ground planes, under the antenna. The ground plane cutout must be a minimum of 6.5mm × 11mm, but ideally at least 8mm × 15mm. If these requirements can't be met wireless performance may be degraded, especially in the 2.4GHz spectrum. It is recommended that the external antenna is used where possible.

NOTE

The location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however the maximum component heights and PCB thickness will be kept as specified.

A step file of the CM4 is available as part of the CM4 design data package. This is for guidance only and is subject to changes over time due to revisions.

3.2. Thermal

The CM4 dissipates less power than the Raspberry Pi 4 Model B. The CM4 also contains less metal in the PCB and fewer connectors, which means that it has less passive heat sinking than the Raspberry Pi 4 Model B. Despite it consuming less power, it may run warmer than the Raspberry Pi 4 Model B.

The BCM2711 will reduce the clock rate to try and keep its internal temperature below 85°C. So in high ambient temperatures it is possible that the clock will also be automatically throttled back. If the BCM2711 is unable to lower its internal clocks enough to bring the temperature down, its case temperature will rise above 85°C. It is important that any thermal solution chosen keeps the ambient temperature for the other silicon devices on the CM4 within the operating temperature range.

Operating temperature range: -20°C - +85°C non-condensing. NB Optimal RF wireless performance is between -20°C and +75°C.

3.3. Electrical specification

WARNING

Stresses above those listed in Table 3 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 3. Absolute maximum ratings

Symbol	Parameter	Minimum	Maximum	Unit
V_{IN}	5V Input Voltage	-0.5	6.0	V
V_{GPIO_VREF}	GPIO Voltage	-0.5	3.6	V
V_{gpio}	GPIO Input voltage	-0.5	V_{GPIO_VREF} + 0.5	V

NOTE

V_{GPIO_VREF} is the GPIO bank voltage, which must be tied to either the 3.3V or the 1.8V rail of the CM4.

Table 4. DC characteristics

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
$V_{IL(gpio)}$	Input low voltage	V_{GPIO_VREF} = 3.3V	0	-	0.8	V
$V_{IH(gpio)}$	Input high voltage	V_{GPIO_VREF} = 3.3V	2.0	-	V_{GPIO_VREF}	V

$V_{IL(gpio)}$	Input low voltage	V_{GPIO_VREF} = 1.8V	0	-	0.35	V
$V_{IH(gpio)}$	Input high voltage	V_{GPIO_VREF} = 1.8V	0.65	-	V_{GPIO_VREF}	V
$I_{IL(gpio)}$	Input leakage current	-	-	-	10	µA
$V_{OL(gpio)}$	Output low voltage	-	-	-	0.4	V
$V_{OH(gpio)}$	Output high voltage	-	V_{GPIO_VREF} - 0.4	-	-	V
$I_{O(gpio)}$	Output current	1mA	0.87	1.3	-	mA
$I_{O(gpio)}$	Output current	2mA	1.75	2.6	-	mA
$I_{O(gpio)}$	Output current	3mA	2.63	3.9	-	mA
$I_{O(gpio)}$	Output current	4mA **default**	3.5	5.3	-	mA
$I_{O(gpio)}$	Output current	5mA	4.39	6.6	-	mA
$I_{O(gpio)}$	Output current	6mA	5.27	7.9	-	mA
$I_{O(gpio)}$	Output current	7mA	6.15	9.2	-	mA
$I_{O(gpio)}$	Output current	8mA	7.02	10.5	-	mA
$R_{PU(gpio)}$	Pullup resistor	V_{GPIO_VREF} = 3.3V	33	47	73	kΩ
$R_{PD(gpio)}$	Pulldown resistor	V_{GPIO_VREF} = 3.3V	33	47	73	kΩ
$R_{PU(gpio)}$	Pullup resistor	V_{GPIO_VREF} = 1.8V	18	47	73	kΩ
$R_{PD(gpio)}$	Pulldown resistor	V_{GPIO_VREF} = 1.8V	18	47	73	kΩ

Refer to interface specifications (see Chapter 2) for electrical details of other interfaces.

Table 5. Power consumption

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
$I_{shutdown}$	Shutdown current	`GLOBAL_EN` = 0V	-	15	-	µA
$I_{shutdown}$	Shutdown current	`GLOBAL_EN` > 2V	-	8	-	mA
I_{idle}	Idle current	`GLOBAL_EN` > 2V	-	400	-	mA
I_{load}	Operation current	`GLOBAL_EN` > 2V	-	1400	-	mA

NOTE

The figures in Table 5 greatly depend on the end application.

Chapter 4. Pinout

Table 6. Pinout for the Raspberry Pi Compute Module 4

Pin	Signal	Description
1	GND	Ground (0V)
2	GND	Ground (0V)
3	Ethernet_Pair3_P	Ethernet pair 3 positive (connect to transformer or MagJack)
4	Ethernet_Pair1_P	Ethernet pair 1 positive (connect to transformer or MagJack)
5	Ethernet_Pair3_N	Ethernet pair 3 negative (connect to transformer or MagJack)
6	Ethernet_Pair1_N	Ethernet pair 1 negative (connect to transformer or MagJack)
7	GND	Ground (0V)
8	GND	Ground (0V)
9	Ethernet_Pair2_N	Ethernet pair 2 negative (connect to transformer or MagJack)
10	Ethernet_Pair0_N	Ethernet pair 0 negative (connect to transformer or MagJack)
11	Ethernet_Pair2_P	Ethernet pair 2 positive (connect to transformer or MagJack)
12	Ethernet_Pair0_P	Ethernet pair 0 positive (connect to transformer or MagJack)
13	GND	Ground (0V)
14	GND	Ground (0V)
15	Ethernet_nLED3	Active-low Ethernet activity indicator (`CM4_3.3V` signal): typically a green LED is connected to this pin. I_{OL} = 8mA @ V_{OL} < 0.4V
16	Ethernet_SYNC_IN	IEEE1588 SYNC Input pin (`CM4_3.3V` signal: I_{OL} = 8mA @ V_{OL} < 0.4V)
17	Ethernet_nLED2	Active-low Ethernet speed indicator (`CM4_3.3V`signal): typically a yellow LED is connected to this pin. A low state indicates the 1Gbit or 100Mbit link: I_{OL} = 8mA @ V_{OL} < 0.4V
18	Ethernet_SYNC_OUT	IEEE1588 SYNC Output pin (`CM4_3.3V` signal: I_{OL} = 8mA @ V_{OL} < 0.4V)
19	Ethernet_nLED1	Active-low Ethernet speed indicator (`CM4_3.3V` signal): typically a yellow LED is connected to this pin. A low state indicates the 1Gbit or 10Mbit link: I_{OL} = 8mA @ V_{OL} < 0.4V
20	EEPROM_nWP	Leave floating NB internally pulled up to `CM4_3.3V` via 100kΩ (V_{IL} < 0.8V), but can be grounded to prevent writing to the on-board EEPROM which stores the bootcode
21	Pi_nLED_Activity	Active-low Pi activity LED. 20mA Max, 5V tolerant (V_{OL} < 0.4V). (this is the signal that drives the green LED on the Raspberry Pi 4 Model B)
22	GND	Ground (0V)
23	GND	Ground (0V)
24	GPIO26	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to `CM4_1.8V`
25	GPIO21	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to `CM4_1.8V`
26	GPIO19	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to `CM4_1.8V`
27	GPIO20	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to `CM4_1.8V`

28	GPIO13	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
29	GPIO16	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
30	GPIO6	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
31	GPIO12	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
32	GND	Ground (0V)
33	GND	Ground (0V)
34	GPIO5	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
35	ID_SC	(BCM2711 GPIO 1) GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
36	ID_SD	(BCM2711 GPIO 0) GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
37	GPIO7	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
38	GPIO11	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
39	GPIO8	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
40	GPIO9	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
41	GPIO25	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
42	GND	Ground (0V)
43	GND	Ground (0V)
44	GPIO10	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
45	GPIO24	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
46	GPIO22	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
47	GPIO23	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
48	GPIO27	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
49	GPIO18	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
50	GPIO17	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
51	GPIO15	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
52	GND	Ground (0V)

53	GND	Ground (0V)
54	GPIO4	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
55	GPIO14	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V
56	GPIO3	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V. Internal 1.8kΩ pull up to GPIO_VREF
57	SD_CLK	SD card clock signal (only available on CM4Lite)
58	GPIO2	GPIO: typically a 3.3V signal, but can be a 1.8V signal by connecting GPIO_VREF to CM4_1.8V. Internal 1.8kΩ pull up to GPIO_VREF
59	GND	Ground (0V)
60	GND	Ground (0V)
61	SD_DAT3	SD card/eMMC Data3 signal (only available on CM4Lite)
62	SD_CMD	SD card/eMMC Command signal (only available on CM4Lite)
63	SD_DAT0	SD card/eMMC Data0 signal (only available on CM4Lite)
64	SD_DAT5	SD card/eMMC Data5 signal (only available on CM4Lite)
65	GND	Ground (0V)
66	GND	Ground (0V)
67	SD_DAT1	SD card/eMMC Data1 signal (only available on CM4Lite)
68	SD_DAT4	SD card/eMMC Data4 signal (only available on CM4Lite)
69	SD_DAT2	SD card/eMMC Data2 signal (only available on CM4Lite)
70	SD_DAT7	SD card/eMMC Data7 signal (only available on CM4Lite)
71	GND	Ground (0V)
72	SD_DAT6	SD card/eMMC Data6 signal (only available on CM4Lite)
73	SD_VDD_OVERRIDE	Connect to CM4_3.3V to force SD card/eMMC interface to 1.8V signalling instead of 3.3V, otherwise leave unconnected. Typically only used if external eMMC is connected.
74	GND	Ground (0V)
75	SD_PWR_ON	Output to power-switch for the SD card. The CM4 sets this pin high (3.3V) to signal that power to the SD card should be turned on. If booting from the SD card is required then a pullup should also be fitted so the power-switch defaults to on. (only available on CM4Lite)
76	Reserved	Do not connect anything to this pin.
77	+5V (Input)	4.75V-5.25V. Main power input
78	GPIO_VREF	Must be connected to CM4_3.3V (pins 84 and 86) for 3.3V GPIO or CM4_1.8V (pins 88 and 90) for 1.8V GPIO. This pin cannot be floating or connected to ground.
79	+5V (Input)	4.75V-5.25V. Main power input
80	SCL0	I2C clock pin (BCM2711 GPIO45): typically used for Camera and Display. Internal 1.8kΩ pull up to CM4_3.3V
81	+5V (Input)	4.75V-5.25V. Main power input
82	SDA0	I2C Data pin (BCM2711 GPIO44): typically used for Camera and Display. Internal 1.8kΩ pull up to CM4_3.3V

83	+5V (Input)	4.75V-5.25V. Main power input
84	CM4_3.3V (Output)	3.3V ± 2.5%. Power Output max 300mA per pin for a total of 600mA. This will be powered down during power-off or GLOBAL_EN being set low
85	+5V (Input)	4.75V-5.25V. Main power input
86	CM4_3.3V (Output)	3.3V ± 2.5%. Power Output max 300mA per pin for a total of 600mA. This will be powered down during power-off or GLOBAL_EN being set low
87	+5V (Input)	4.75V-5.25V. Main power input
88	CM4_1.8V (Output)	1.8V ± 2.5%. Power Output max 300mA per pin for a total of 600mA. This will be powered down during power-off or GLOBAL_EN being set low
89	WL_nDisable	Can be left floating; if driven low the wireless interface will be disabled. Internally pulled up via 1.8kΩ to `CM4_3.3V`
90	CM4_1.8V (Output)	1.8V ± 2.5%. Power Output max 300mA per pin for a total of 600mA. This will be powered down during power-off or GLOBAL_EN being set low
91	BT_nDisable	Can be left floating; if driven low the Bluetooth interface will be disabled. Internally pulled up via 1.8kΩ to `CM4_3.3V`
92	RUN_PG	Bidirectional pin. Can be driven low (via a 220Ω resistor) to reset the CM4 CPU. As an output, a high signals that power is good and CPU is running. Internally pulled up to +3.3V via 10kΩ
93	nRPIBOOT	A low on this pin forces booting from an RPI server (e.g. PC or a Raspberry Pi); if not used leave floating. Internally pulled up via 10kΩ to +3.3V
94	AnalogIP1	Analogue input of the MXL7704: typically connected to CC pin of Type C power connector
95	PI_LED_nPWR	Active-low output to drive Power On LED. This signal needs to be buffered.
96	AnalogIP0	Analogue input of the MXL7704: typically connected to CC pin of Type C power connector
97	Camera_GPIO	Typically used to shut down the camera to reduce power. Reassigning this pin to another function isn't recommended. `CM4_3.3V` signalling
98	GND	Ground (0V)
99	GLOBAL_EN	Input. Drive low to power off CM4. Internally pulled up with a 100kΩ to +5V
100	nEXTRST	Output. Driven low during reset; Driven high (`CM4_3.3V`) once CM4 CPU has started to boot
101	USB_OTG_ID	Input (3.3V signal) USB OTG Pin. Internally pulled up. When grounded the CM4 becomes a USB host but the correct OS driver also needs to be used
102	PCIe_CLK_nREQ	Input (3.3V signal) PCIe clock request pin (low to request PCI clock). Internally pulled up
103	USB_N	USB D-
104	Reserved	Do not connect anything to this pin.
105	USB_P	USB D+
106	Reserved	Do not connect anything to this pin.
107	GND	Ground (0V)
108	GND	Ground (0V)
109	PCIe_nRST	Output (+3.3V signal) PCIe reset active-low
110	PCIe_CLK_P	PCIe clock Out positive (100MHz) NB AC coupling capacitor included on CM4
111	VDAC_COMP	Video DAC output (TV OUT)

112	PCIe_CLK_N	PCIe clock Out negative (100MHz) NB AC coupling capacitor included on CM4
113	GND	Ground (0V)
114	GND	Ground (0V)
115	CAM1_D0_N	Input Camera1 D0 negative
116	PCIe_RX_P	Input PCIe GEN 2 RX positive NB external AC coupling capacitor required
117	CAM1_D0_P	Input Camera1 D0 positive
118	PCIe_RX_N	Input PCIe GEN 2 RX negative NB external AC coupling capacitor required
119	GND	Ground (0V)
120	GND	Ground (0V)
121	CAM1_D1_N	Input Camera1 D1 negative
122	PCIe_TX_P	Output PCIe GEN 2 TX positive NB AC coupling capacitor included on CM4
123	CAM1_D1_P	Input Camera1 D1 positive
124	PCIe_TX_N	Output PCIe GEN 2 TX positive NB AC coupling capacitor included on CM4
125	GND	Ground (0V)
126	GND	Ground (0V)
127	CAM1_C_N	Input Camera1 clock negative
128	CAM0_D0_N	Input Camera0 D0 negative
129	CAM1_C_P	Input Camera1 clock positive
130	CAM0_D0_P	Input Camera0 D0 positive
131	GND	Ground (0V)
132	GND	Ground (0V)
133	CAM1_D2_N	Input Camera1 D2 negative
134	CAM0_D1_N	Input Camera0 D1 negative
135	CAM1_D2_P	Input Camera1 D2 positive
136	CAM0_D1_P	Input Camera0 D1 positive
137	GND	Ground (0V)
138	GND	Ground (0V)
139	CAM1_D3_N	Input Camera1 D3 negative
140	CAM0_C_N	Input Camera0 clock negative
141	CAM1_D3_P	Input Camera1 D3 positive
142	CAM0_C_P	Input Camera0 clock positive
143	HDMI1_HOTPLUG	Input HDMI1 hotplug. Internally pulled down with a 100kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
144	GND	Ground (0V)
145	HDMI1_SDA	Bidirectional HDMI1 SDA. Internally pulled up with a 1.8kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)

146	HDMI1_TX2_P	Output HDMI1 TX2 positive
147	HDMI1_SCL	Bidirectional HDMI1 SCL. Internally pulled up with a 1.8kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
148	HDMI1_TX2_N	Output HDMI1 TX2 negative
149	HDMI1_CEC	Input HDMI1 CEC. Internally pulled up with a 27kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
150	GND	Ground (0V)
151	HDMI0_CEC	Input HDMI0 CEC. Internally pulled up with a 27kΩ. 5V tolerant (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
152	HDMI1_TX1_P	Output HDMI1 TX1 positive
153	HDMI0_HOTPLUG	Input HDMI0 hotplug. Internally pulled down 100kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
154	HDMI1_TX1_N	Output HDMI1 TX1 negative
155	GND	Ground (0V)
156	GND	Ground (0V)
157	DSI0_D0_N	Output Display0 D0 negative
158	HDMI1_TX0_P	Output HDMI1 TX0 positive
159	DSI0_D0_P	Output Display0 D0 positive
160	HDMI1_TX0_N	Output HDMI1 TX0 negative
161	GND	Ground (0V)
162	GND	Ground (0V)
163	DSI0_D1_N	Output Display0 D1 negative
164	HDMI1_CLK_P	Output HDMI1 clock positive
165	DSI0_D1_P	Output Display0 D1 positive
166	HDMI1_CLK_N	Output HDMI1 clock negative
167	GND	Ground (0V)
168	GND	Ground (0V)
169	DSI0_C_N	Output Display0 clock negative
170	HDMI0_TX2_P	Output HDMI0 TX2 positive
171	DSI0_C_P	Output Display0 clock positive
172	HDMI0_TX2_N	Output HDMI0 TX2 negative
173	GND	Ground (0V)
174	GND	Ground (0V)
175	DSI1_D0_N	Output Display1 D0 negative
176	HDMI0_TX1_P	Output HDMI0 TX1 positive

177	DSI1_D0_P	Output Display1 D0 positive
178	HDMI0_TX1_N	Output HDMI0 TX1 negative
179	GND	Ground (0V)
180	GND	Ground (0V)
181	DSI1_D1_N	Output Display1 D1 negative
182	HDMI0_TX0_P	Output HDMI0 TX0 positive
183	DSI1_D1_P	Output Display1 D1 positive
184	HDMI0_TX0_N	Output HDMI0 TX0 negative
185	GND	Ground (0V)
186	GND	Ground (0V)
187	DSI1_C_N	Output Display1 clock negative
188	HDMI0_CLK_P	Output HDMI0 clock positive
189	DSI1_C_P	Output Display1 clock positive
190	HDMI0_CLK_N	Output HDMI0 clock negative
191	GND	Ground (0V)
192	GND	Ground (0V)
193	DSI1_D2_N	Output Display1 D2 negative
194	DSI1_D3_N	Output Display1 D3 negative
195	DSI1_D2_P	Output Display1 D2 positive
196	DSI1_D3_P	Output Display1 D3 positive
197	GND	Ground (0V)
198	GND	Ground (0V)
199	HDMI0_SDA	Bidirectional HDMI0 SDA. Internally pulled up with a 1.8kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)
200	HDMI0_SCL	Bidirectional HDMI0 SCL. Internally pulled up with a 1.8kΩ. 5V tolerant. (It can be connected directly to a HDMI connector; a small amount of ESD protection is provided on the CM4 by an on-board HDMI05-CL02F3)

All ground pins should be connected. If none of the signals on the second connector (pins 101 to 200) are used, then you may omit the connector to reduce costs, but mechanical stability needs to be considered.

The voltage on GPIO pins 0-27 must not exceed `CM4_3.3V` if +3.3V signalling is used or `CM4_1.8V` if +1.8V signalling is used. These pins are the same as on the 40-pin connector on the Raspberry Pi 4 Model B.

If the `CM4_1.8V` rail is used to power other devices other than the `GPIO_VREF` then you should ensure that in case of surprise power removal (e.g.the +5V pin goes below +4.5V) from the CM4, the load on the `CM4_1.8V` must go to zero.

Similarly if the `CM4_3.3V` rail is used to power other devices other than the `GPIO_VREF`, then you should ensure that in the case of surprise power removal the `CM4_3.3V` rail never falls below the `CM4_1.8V` rail. This is the typical case, but you should check this in your design. In the case where it does fall below the `CM4_1.8V` rail, then extra circuitry is required to disconnect the `CM4_3.3V` load.

No reverse voltage must be applied to any pin, or power-up may be prevented; i.e. during power-down/off no pin may have external voltage applied, otherwise this may prevent a subsequent power-up.

4.1. Differential pairs

It is recommended that P/N signals within a pair are matched to better than 0.15mm. Often, matching between pairs is not so critical: e.g. HDMI pair-to-pair matching should be better than 25mm, so on a typical board no extra matching is required.

4.1.1. 100Ω differential pair signal lengths

On the CM4 all differential pairs are matched to better than 0.05mm (P/N signals).

NOTE

It is recommended that pairs are also matched on the interface board.

On the CM4, pair-to-pairs are not always matched, as many interfaces do not require very accurate matching between pairs. Table 7 documents the CM4 track-length difference within each group. (A non-zero value represents how much longer in mm that track is, when compared to the signal with zero length difference.)

Table 7. 100Ω differential pair signal lengths

Signal	Length
CAM0_C_N	0.02
CAM0_C_P	0.02
CAM0_D0_N	0.06
CAM0_D0_P	0.07
CAM0_D1_N	0
CAM0_D1_P	0.01
CAM1_C_N	0.78
CAM1_C_P	0.78
CAM1_D0_N	0.02
CAM1_D0_P	0.01
CAM1_D1_N	0.4
CAM1_D1_P	0.4
CAM1_D2_N	0.05
CAM1_D2_P	0.04
CAM1_D3_N	0.01
CAM1_D3_P	0
DSI0_C_N	0
DSI0_C_P	0
DSI0_D0_N	0
DSI0_D0_P	0
DSI0_D1_N	0.01
DSI0_D1_P	0.01

DSI1_C_N	1.28
DSI1_C_P	1.28
DSI1_D0_N	0
DSI1_D0_P	0.01
DSI1_D1_N	1.06
DSI1_D1_P	1.06
DSI1_D2_N	0.83
DSI1_D2_P	0.84
DSI1_D3_N	3.78
DSI1_D3_P	3.79
HDMI0_CLK_N	3.25
HDMI0_CLK_P	3.24
HDMI0_TX0_N	1.76
HDMI0_TX0_P	1.76
HDMI0_TX1_N	0.62
HDMI0_TX1_P	0.62
HDMI0_TX2_N	0
HDMI0_TX2_P	0
HDMI1_CLK_N	2.47
HDMI1_CLK_P	2.46
HDMI1_TX0_N	1.51
HDMI1_TX0_P	1.51
HDMI1_TX1_N	1
HDMI1_TX1_P	1
HDMI1_TX2_N	0
HDMI1_TX2_P	0.01
Ethernet_Pair0_P	5.23
Ethernet_Pair0_N	5.23
Ethernet_Pair1_P	0
Ethernet_Pair1_N	0
Ethernet_Pair2_P	3.82
Ethernet_Pair2_N	3.82
Ethernet_Pair3_P	4.29
Ethernet_Pair3_N	4.29

4.1.2. 90Ω differential pair signal lengths

On the CM4 all differential pairs are matched to better than 0.05mm (P/N signals).

NOTE

It is recommended that pairs are also matched on the interface board.

Pair-to-pairs aren't always matched as many interfaces don't require very accurate matching between pairs. Table 8 documents the CM4 track-length difference within each group. (A non-zero value represents how much longer in mm that track is, when compared to the signal with zero length difference.)

Table 8. 90Ω differential pair signal lengths

Signal	Length
PCIe_CLK_P	0.65
PCIe_CLK_N	0.65
PCIe_TX_P	0
PCIe_TX_N	0
PCIe_RX_P	0.23
PCIe_RX_N	0.23
USB2_P	0
USB2_N	0

Chapter 5. Power

5.1. Power-up sequencing

The CM4 requires a single +5V supply, and can supply up to 600mA at +3.3V and +1.8V to peripherals.

All pins should not have any power applied to them before the +5V rail is applied.

If the EEPROM is to be write-protected, then the `EEPROM_nWP` should be low before power-up.

If the CM4 is to be booted using USB then `RPI_nBOOT` needs to be low within 2ms of +5V rising.

+5V should rise monotonically to 4.75V and stay above 4.75V for the entire operation of the CM4.

The power-up sequence will start when both +5V rail is above 4.75V and `GLOBAL_EN` rises. `GLOBAL_EN` has internal RC delay so that it rises after +5V has risen. The order of events is as follows

1. +5V rises
2. `GLOBAL_EN` rises
3. +3.3V rises
4. +1.8V rises at least 1ms after +3.3V
5. `RUN_PG` rises at least 10ms after +1.8V
6. `EXT_nRESET` rises at least 1s after `RUN_PG`

5.2. Power-down sequencing

The operating system should be shut down before the power is removed, to ensure that the file system remains consistent. If this can't be achieved, then a filesystem like `btrfs`, `f2fs` or `overlayfs` (use `raspi-config` to enable this) should be considered.

Once the operating system has shut down, the +5V rail can be removed or the `GLOBAL_EN` pin can be taken low to put the CM4 into the lowest power mode.

During the shutdown sequence the +1.8V will be discharged before the +3.3V rail.

5.3. Power consumption

The exact power consumption of the CM4 will greatly depend on the tasks being run on the CM4. The lowest shutdown power consumption mode is with the `GLOBAL_EN` driven low, typically is 15µA. With `GLOBAL_EN` high but software shut down, the typical consumption is 8mA. Idle power consumption is typically 400mA, but this varies considerably depending on the operating system. Operating power consumption is typically around 1.4A; again, this greatly depends on the operating system and the tasks being executed.

5.4. Regulator outputs

To make it easier to interface to the CM4 the on-board regulators (+3.3V and +1.8V) can each supply 600mA to devices connected to the CM4. The loads on these outputs isn't taken into account in the power consumption figures.

Appendix A: Troubleshooting

The CM4 has a number of stages of power-up before the CPU starts. If there is an error at any of the stages, power-up will be halted.

Hardware checklist

1. Is the +5V supply good? Check this by pulling `GLOBAL_EN` low and apply an external 2A load to the +5V supply. Does it stay > +4.75V including noise? Ideally it should remain > +4.9V including any noise.
2. Remove external 2A load, but keep `GLOBAL_EN` pulled low.
3. Check the CM4 +3.3V rail is < 200mV. If this is not the case there is an external power path back-feeding the CM4, either directly or indirectly. This could also occur via the digital pins, e.g Ethernet.
4. Still with `GLOBAL_EN` pulled low check the CM4 +1.8V rail is < 200mV. Again if the +1.8V rail is above 200mV then there is an external path back-feeding the 1.8V rail. (If nothing is connected to these pins you can ignore this check.)
5. Remove the pull down on `GLOBAL_EN`.
6. Check `GLOBAL_EN` now goes high (it is internally pulled up on the CM4)
7. Check the +3.3V supply rises to > +3.15V. If it does not, this suggests there is too much load on the +3.3V rail.
8. Check the +1.8V rail gets to > +1.71V. If it does not, this suggests there is too much load on the +1.8V rail.
9. Check `RUN_PG` goes high
10. Check `ACT_LED` starts to oscillate to indicate booting; check it isn't flashing an error code.

Bootloader

1. Connect a HDMI cable to see if the HDMI diagnostics screen appears.
2. Connect a USB serial cable to GPIO pins 14 and 15.
    a. See https://www.raspberrypi.com/documentation/computers/configuration.html#configuring-uarts for details.
3. Short the `nRPIBOOT` pin to ground to force USB boot mode. The CM4IO board has a jumper for `nRPIBOOT` This can be used to enable different boot modes (e.g. network) and enable UART logging.
    a. See https://www.raspberrypi.com/documentation/computers/compute-module.html#flashing-the-compute-module-emmc

rpi-eeprom-update

1. CM4 will not run `recovery.bin` from from the EMMC (or SD Card on CM4Lite). Therefore, the only way to update the bootloader EEPROM is via `usbboot` or self-update.

EEPROM write-protect

The on-board EEPROM can be write-protected by shorting `EEPROM_nWP` to ground. The CM4IO board has a jumper for

`EEPROM_nWP`.

1. See https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-4-bootloader-configuration

Firmware

1. A 5.4 or newer kernel and the latest firmware release is required. These can be updated by using usbboot to mount the EMMC as a USB MSD device.
2. Nightly OS images are now available which contain `rpi-update` master firmware + kernel. Bug fixes for CM4 will normally be provided via these images except where a test/patch binary is required.
    a. See http://downloads.raspberrypi.org/nightlies/

Kernel

1. The updated OS images use the new Raspberry Pi Compute Module 4 device tree file. If that is not found then the Raspberry Pi 4 Model B device tree file will be used.
    a. See https://github.com/raspberrypi/linux/blob/rpi-5.4.y/arch/arm/boot/dts/bcm2711-rpi-cm4.dts

Appendix B: Availability

Support

For documentation please see the Compute Module Hardware documentation section of the Raspberry Pi website. Support questions can be posted to the Raspberry Pi forum.

Ordering codes

Table 9. Part number options

Model	Wireless	RAM LPDDR4	eMMC Storage
CM4	0 = No	01 = 1GB	000 = 0GB (Lite)
	1 = Yes	02 = 2GB	008 = 8GB
		04 = 4GB	016 = 16GB
		08 = 8GB	032 = 32GB
Example Part Number			
CM4	1	02	032

Table 10. Ordering options

Wireless	RAM LPDDR4	Storage eMMC	RPL #	Part Number	Order Multiple	RRP
-	1GB	Lite	SC0695B	CM4001000	1+ / Bulk	$ 30.00
-	1GB	8GB	SC0696B	CM4001008	1+ / Bulk	$ 35.00
-	1GB	16GB	SC0697B	CM4001016	1+ / Bulk	$ 40.00
-	1GB	32GB	SC0698B	CM4001032	1+ / Bulk	$ 45.00
Yes	1GB	Lite	SC0691B	CM4101000	1+ / Bulk	$ 35.00
Yes	1GB	8GB	SC0692B	CM4101008	1+ / Bulk	$ 40.00
Yes	1GB	16GB	SC0693B	CM4101016	1+ / Bulk	$ 45.00
Yes	1GB	32GB	SC0694B	CM4101032	1+ / Bulk	$ 50.00
-	2GB	Lite	SC0679B	CM4002000	1+ / Bulk	$ 35.00
-	2GB	8GB	SC0680B	CM4002008	1+ / Bulk	$ 40.00
-	2GB	16GB	SC0681B	CM4002016	1+ / Bulk	$ 45.00
-	2GB	32GB	SC0682B	CM4002032	1+ / Bulk	$ 50.00
Yes	2GB	Lite	SC0667B	CM4102000	1+ / Bulk	$ 40.00
Yes	2GB	8GB	SC0668B	CM4102008	1+ / Bulk	$ 45.00
Yes	2GB	16GB	SC0669B	CM4102016	1+ / Bulk	$ 50.00
Yes	2GB	32GB	SC0670B	CM4102032	1+ / Bulk	$ 55.00
-	4GB	Lite	SC0683B	CM4004000	1+ / Bulk	$ 50.00
-	4GB	8GB	SC0684B	CM4004008	1+ / Bulk	$ 55.00

-	4GB	16GB	SC0685B	CM4004016	1+ / Bulk	$ 60.00
-	4GB	32GB	SC0686B	CM4004032	1+ / Bulk	$ 65.00
Yes	4GB	Lite	SC0671B	CM4104000	1+ / Bulk	$ 55.00
Yes	4GB	8GB	SC0672B	CM4104008	1+ / Bulk	$ 60.00
Yes	4GB	16GB	SC0673B	CM4104016	1+ / Bulk	$ 65.00
Yes	4GB	32GB	SC0674B	CM4104032	1+ / Bulk	$ 70.00
-	8GB	Lite	SC0687B	CM4008000	1+ / Bulk	$ 75.00
-	8GB	8GB	SC0688B	CM4008008	1+ / Bulk	$ 80.00
-	8GB	16GB	SC0689B	CM4008016	1+ / Bulk	$ 85.00
-	8GB	32GB	SC0690B	CM4008032	1+ / Bulk	$ 90.00
Yes	8GB	Lite	SC0675B	CM4108000	1+ / Bulk	$ 80.00
Yes	8GB	8GB	SC0676B	CM4108008	1+ / Bulk	$ 85.00
Yes	8GB	16GB	SC0677B	CM4108016	1+ / Bulk	$ 90.00
Yes	8GB	32GB	SC0678B	CM4108032	1+ / Bulk	$ 95.00

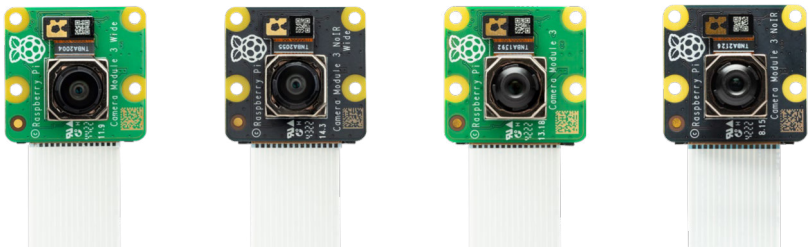
NOTE

RRP was correct at time of publication and excludes taxes.

Packaging

Small quantities are supplied in individual cardboard boxes. These have an internal ESD coating so that a separate ESD bag isn't required. This packaging is recyclable and reduces waste.

Overview



Raspberry Pi Camera Module 3 is a compact camera from Raspberry Pi. It offers an IMX708 12-megapixel sensor with HDR, and features phase detection autofocus. Camera Module 3 is available in standard and wide-angle variants, both of which are available with or without an infrared cut filter.

Camera Module 3 can be used to take full HD video as well as stills photographs, and features an HDR mode up to 3 megapixels. Its operation is fully supported by the libcamera library, including Camera Module 3's rapid autofocus feature: this makes it easy for beginners to use, while offering plenty for advanced users. Camera Module 3 is compatible with all Raspberry Pi computers.[1]

The PCB size and mounting holes remain the same as for Camera Module 2. The Z dimension differs: due to the improved optics, Camera Module 3 is several millimetres taller than Camera Module 2.

All variants of Camera Module 3 feature:

- Back-illuminated and stacked CMOS 12-megapixel image sensor (Sony IMX708)
- High signal-to-noise ratio (SNR)
- Built-in 2D Dynamic Defect Pixel Correction (DPC)
- Phase Detection Autofocus (PDAF) for rapid autofocus
- QBC Re-mosaic function
- HDR mode (up to 3 megapixel output)
- CSI-2 serial data output
- 2-wire serial communication (supports I2C fast mode and fast-mode plus)
- 2-wire serial control of focus mechanism

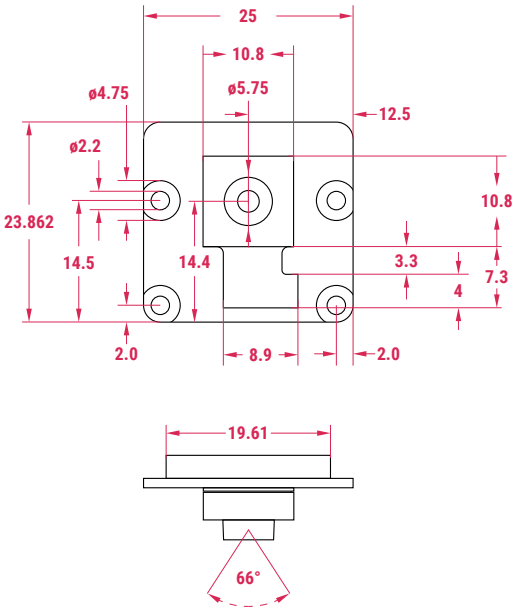
[1] Excluding early Raspberry Pi Zero models, which lack the necessary FPC connector. Later Raspberry Pi Zero models require an adapter FPC, sold separately.

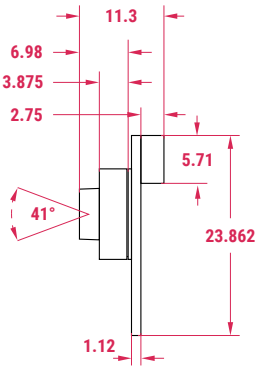
Specification

Sensor:	Sony IMX708
Resolution:	11.9 megapixels
Sensor size:	7.4mm sensor diagonal
Pixel size:	1.4µm × 1.4µm
Horizontal/vertical:	4608 × 2592 pixels
Common video modes:	1080p50, 720p100, 480p120
Output:	RAW10
IR cut filter:	Integrated in standard variants; not present in NoIR variants
Autofocus system:	Phase Detection Autofocus
Dimensions:	25 × 24 × 11.5mm (12.4mm height for Wide variants)
Ribbon cable length:	200mm
Cable connector:	15 × 1mm FPC
Operating temperature:	0°C to 50°C
Compliance:	FCC 47 CFR Part 15, Subpart B, Class B Digital Device Electromagnetic Compatibility Directive (EMC) 2014/30/EU Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU
Production lifetime:	Raspberry Pi Camera Module 3 will remain in production until at least January 2030

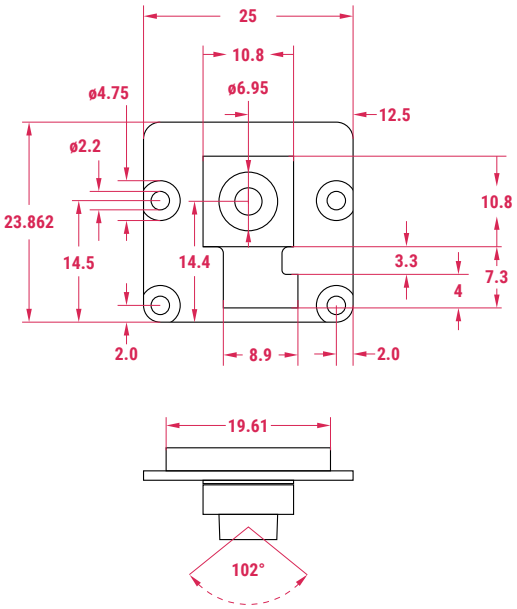
Physical specification

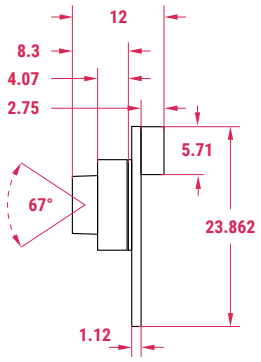
Standard lens





Wide lens





Note: all dimensions in mm
tolerances are accurate to 0.2mm

Variants

	Camera Module 3	Camera Module 3 NoIR	Camera Module 3 Wide	Camera Module 3 Wide NoIR
Focus range	10cm–∞	10cm–∞	5cm–∞	5cm–∞
Focal length	4.74mm	4.74mm	2.75mm	2.75mm
Diagonal field of view	75 degrees	75 degrees	120 degrees	120 degrees
Horizontal field of view	66 degrees	66 degrees	102 degrees	102 degrees
Vertical field of view	41 degrees	41 degrees	67 degrees	67 degrees
Focal ratio (F-stop)	F1.8	F1.8	F2.2	F2.2
Infrared-sensitive	No	Yes	No	Yes

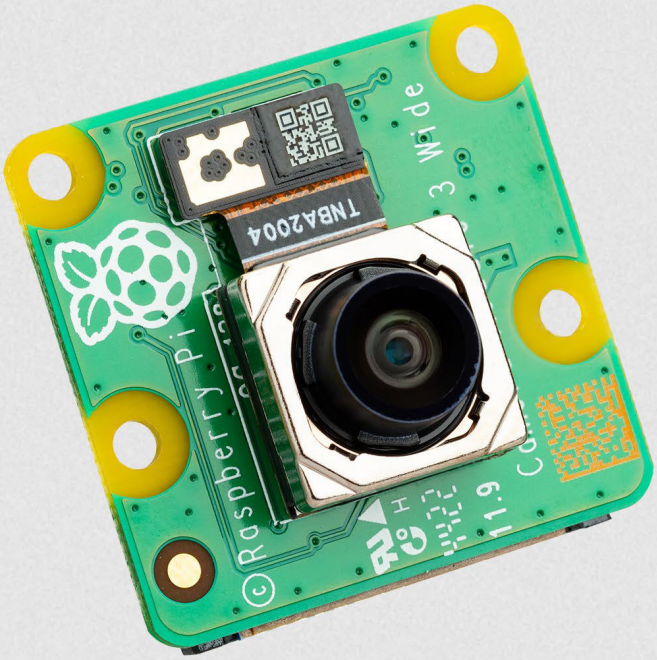
WARNINGS

- This product should be operated in a well ventilated environment, and if used inside a case, the case should not be covered.
- Whilst in use, this product should be firmly secured or should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Camera Module 3 may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

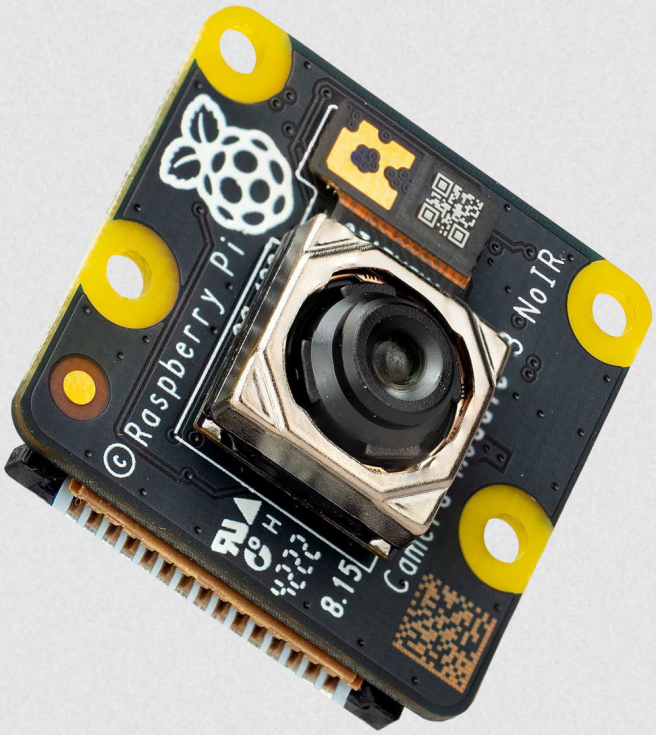
SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Important: Before connecting this device, shut down your Raspberry Pi computer and disconnect it from external power.
- If the cable becomes detached, first pull forward the locking mechanism on the connector, then insert the ribbon cable ensuring that the metal contacts face towards the circuit board, and finally push the locking mechanism back into place.
- This device should be operated in a dry environment at 0–50°C.
- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; Raspberry Pi Camera Module 3 is designed for reliable operation at normal ambient temperatures.
- Store in a cool, dry location.
- Avoid rapid changes of temperature, which can cause moisture to build up in the device, affecting image quality.
- Take care not to fold or strain the ribbon cable.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimise the risk of electrostatic discharge damage.


TNBA2004
Raspberry Pi
Camera Module 3 Wide


Raspberry Pi
Camera Module 3 NoIR Wide


Raspberry Pi
NoIR
Camera Module 3


TNBA1392
Raspberry Pi
Camera Module 3

	Power consumption	≤0.35W
	Peak current	150mA
	Communication level	LVTTL(3.3V)
	Communication interface	UART, I²C, I/O
Others	Dimension	35mm*21.25mm*12.5mm (L*W*H)
	Housing	ABS+PC
	Storage temperature	-20℃~75℃
	Weight	<5g

■ DIMENSIONS

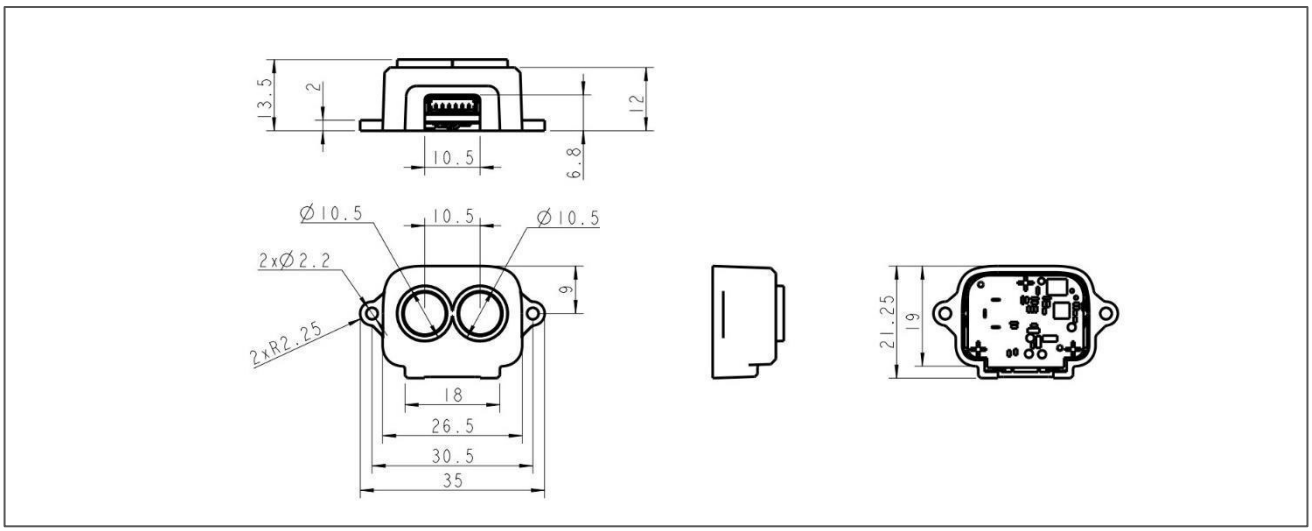


Figure 1 TF-Luna dimensions（Unit: mm）

■ COMMUNICATION INTERFACE

Table 1 Communication Interface—UART

Default baud rate	115200（adjustable）
Data bit	8
Stop bit	1
Parity	None

Table 2 Communication Interface--I2C

Max transmission rate	400kbps
Master/Slave mode	Slave
Default address	0x10
Address range	0x08~0x77