Benutzerdokumentation
Big Data Analytics

Summer School: Not A Drone

Zeitraum:	16.09.2025 - 26.09.2025
Abgabedatum:	05.10.2025
Dozent:	Olga Willner

Projektmitglieder:	Jonathan, Ferdy (s0577931)
	Seitzhanov, Nurzhan (s0593753)
	Wilmerstaedt, Lukas (s0568280)

Inhaltsverzeichnis

1. Einleitung 3

2. Projektzweck und Anwendungsbereich 3

2.1 Problemstellung 3

2.2 Geplanter Nutzen und Mehrwert 4

2.3 Zielgruppe 4

3. Technische Umsetzung 5

3.1 Ausgangsbasis und Board-Konfiguration 5

3.2 Entwicklungsumgebung und Bibliotheken 6

3.3 Kamera und Bildverarbeitung 6

3.4 Motorsteuerung 7

3.5 LED-Signalisierung 8

3.6 Das Zusammenspiel der Komponenten 9

4. Bauanleitung für den Prototypen 9

5. Realisierung 12

6. Fazit und Ausblick 13

7. Reflexion 14

1. Einleitung

Waldbrände stellen in vielen Regionen Europas ein wachsendes Risiko dar. Besonders in Südeuropa, etwa in Portugal oder Spanien, begünstigen extreme Wetterlagen wie starke Winde oder längere Windstille Phasen eine schnelle und unkontrollierte Ausbreitung von Bränden. Um die Folgen für Umwelt, Infrastruktur und Bevölkerung möglichst gering zu halten, spielt die frühzeitige Erkennung von Brandherden eine entscheidende Rolle.

Im Rahmen dieses Projekts wurde ein Prototyp entwickelt, der eine einfache, kostengünstige und flexible Möglichkeit zur Früherkennung von Waldbränden bieten soll. Die Grundidee besteht darin, Ballone oder Drachen als Trägerplattform einzusetzen, an denen ein Kamerasystem mit Auswertungseinheit befestigt wird. Das System nimmt in festgelegten Zeitabständen Bilder der Umgebung auf, wertet diese direkt vor Ort aus und sendet im Falle einer erkannten Brandgefahr automatisch eine Meldung mit den entsprechenden Positionsdaten an die zuständige Feuerwehr.

Die vorliegende Dokumentation beschreibt zunächst den Zweck und die Einsatzmöglichkeiten des Projekts. Im Anschluss wird die technische Umsetzung erläutert, bevor auf praktische Schwierigkeiten und Grenzen des Ansatzes eingegangen wird. Abschließend erfolgt eine Bewertung der bisherigen Ergebnisse sowie ein Ausblick auf mögliche Verbesserungen und Weiterentwicklungen.

2. Projektzweck und Anwendungsbereich

2.1 Problemstellung

Die Bekämpfung von Waldbränden wird oftmals dadurch erschwert, dass Meldungen zu spät erfolgen oder vorhandene Überwachungssysteme nicht ausreichend Abdeckung bieten. Klassische Ansätze wie die Satellitenüberwachung oder fest installierte Kamerasysteme sind entweder sehr kostenintensiv, reagieren zu träge oder lassen sich nur eingeschränkt flexibel einsetzen. Daraus ergibt sich der Bedarf nach einer mobilen,

einfach einsetzbaren und gleichzeitig robusten Lösung, die eine schnellere und verlässliche Früherkennung ermöglicht.

2.2 Geplanter Nutzen und Mehrwert

Das Projekt zielt darauf ab, ein Frühwarnsystem für Waldbrände bereitzustellen, das:

- flexibel in gefährdeten Gebieten installiert werden kann,
- einen Überwachungsradius von bis zu 2 km abdeckt,
- kostengünstig betrieben und gewartet werden kann,
- durch modulare Stromversorgung (Solarmodul oder externe Batterie) über Wochen bis Monate hinweg funktionsfähig bleibt,
- leicht wiederverwendbar ist, indem Ballons nach der Feuergefahr einfach abgelassen und verstaut werden können.

Damit wird ein Beitrag geleistet, Feuerwehren und Katastrophenschutz frühzeitig zu alarmieren, sodass Einsätze effizienter geplant und Schäden deutlich reduziert werden können.

2.3 Zielgruppe

Von einer frühzeitigen Waldbrand Detektion können ganz unterschiedliche Akteure profitieren. An erster Stelle stehen natürlich die Feuerwehren und Katastrophenschutzbehörden, die mit einer schnellen Alarmierung ihre Einsätze gezielter planen und koordinieren können. Auch Forstverwaltungen und Umweltbehörden hätten einen klaren Nutzen, da sie besser in der Lage wären, Waldflächen, Artenvielfalt und natürliche Ressourcen zu schützen.

Darüber hinaus profitieren auch Gemeinden im Wald nahe Regionen, da sich Risiken für die Bevölkerung und Infrastruktur frühzeitig einschätzen lassen. Versicherungen sowie Betreiber kritischer Infrastrukturen, etwa Stromleitungen, Bahnstrecken oder wichtige Verkehrswege, könnten durch eine frühere Warnung Schäden und Ausfälle reduzieren. Schließlich ist das Thema auch für Forschungs- und Entwicklungsinstitutionen

interessant, die an neuen Verfahren zur Brandfrüherkennung oder an allgemeinen Umweltmonitoring Systemen arbeiten.

3. Technische Umsetzung

Das Projekt ***Not a Drone*** basiert auf einem Mikrocontroller des Typs ESP32-S3 und verbindet mehrere Hardwarekomponenten durch eine zentrale Softwaresteuerung. Die Kamera dient als Sensor, der in regelmäßigen Abständen Bilder aufnimmt, die anschließend auf feuerähnliche Farben untersucht werden. Die LED übernimmt die Rolle eines visuellen Signals, das den aktuellen Status für den Benutzer sichtbar macht. Ergänzt wird dies durch einen Motor, der nach einem festgelegten Zyklus betrieben wird und zusätzliche Bewegungen ermöglicht.

3.1 Ausgangsbasis und Board-Konfiguration

Für die Entwicklung der Kamerasteuerung wurde zunächst ein funktionierendes Beispiel aus der Arduino-IDE als Grundlage gewählt. Hierbei handelt es sich um das Programm **CameraWebServer**, das unter **Datei → Beispiele → ESP32 → Camera** verfügbar ist. Dieses Beispiel enthält alle notwendigen Funktionen für die Initialisierung und Nutzung eines ESP32-Kameramoduls und stellt damit eine solide Basis dar, auf der eigene Anpassungen vorgenommen werden konnten.

Da verschiedene ESP32-Kameraboards unterschiedliche Pinbelegungen und Konfigurationen verwenden, musste in der Konfigurationsdatei **board_config.h** das richtige Modell aktiviert werden. Für das eingesetzte Seeed Studio XIAO ESP32-S3 wurde der Eintrag ***#define CAMERA_MODEL_XIAO_ESP32S3*** gesetzt. Alle anderen Modelle wurden auskommentiert. Diese Definition sorgt dafür, dass die Pins für Datenleitungen, Takt und Steuerung korrekt zugeordnet sind.

In der Arduino-IDE wurde zusätzlich das passende Boardmodell über **Tools → Board** ausgewählt. Damit wird sichergestellt, dass der Quellcode für die spezifische Hardware übersetzt wird und die benötigten Treiberfunktionen verfügbar sind. Nach dieser

Konfiguration war die Grundlage geschaffen, um die eigene Softwarestruktur für Bildaufnahme, Motorsteuerung und LED-Signalisierung zu implementieren.

3.2 Entwicklungsumgebung und Bibliotheken

Die Entwicklung erfolgte in der Arduino-IDE, die eine benutzerfreundliche Schnittstelle für die Programmierung des ESP32 bietet. Die Datei **Arduino.h** stellt die Grundfunktionen wie Zeitsteuerung und Pinverwaltung bereit. Über **esp_camera.h** wurde der Kameratreiber eingebunden, der die Steuerung und Initialisierung des Sensors ermöglicht. Mit **img_converters.h** können die aufgenommenen Bilder aus dem JPEG-Format in ein für die Analyse besser geeignetes RGB-Format überführt werden. In **board_config.h** wird das korrekte Kameramodell definiert, sodass die Pinbelegung stimmt.

Darüber hinaus nutzt die Software das Betriebssystem **FreeRTOS**. Es sorgt dafür, dass verschiedene Aufgaben wie Motorsteuerung, LED-Signalisierung und Bildanalyse gleichzeitig ausgeführt werden können. Auf diese Weise wird verhindert, dass eine einzelne Komponente das gesamte System blockiert. Über **driver/ledc.h** wird der PWM-Treiber angesprochen, mit dem die Drehbewegungen des Motors präzise gesteuert werden.

3.3 Kamera und Bildverarbeitung

Die Kamera stellt das Herzstück des Systems dar. Ihre Aufgabe ist es, regelmäßig Bilder aus der Umgebung aufzunehmen und auf Anzeichen von Feuer zu prüfen. Direkt nach dem Start führt die Funktion **init_camera()** die Initialisierung des Sensors durch. Hierbei werden die benötigten Parameter wie Auflösung, Bildformat und Taktfrequenz gesetzt. Bei unserem Aufbau wurde eine Auflösung von 320×240 Pixeln gewählt, da diese ein ausgewogenes Verhältnis zwischen Rechenaufwand und Bildqualität darstellt. Die Bilder werden zunächst im JPEG-Format aufgenommen, da dieses Speicherplatz spart und vom Treiber effizient verarbeitet werden kann.

Da JPEG komprimiert ist und sich für eine Pixel-für-Pixel-Analyse nicht eignet, erfolgt anschließend eine Umwandlung ins **RGB-Format**. Jeder Bildpunkt wird dadurch durch drei Werte beschrieben: den Anteil von Rot, Grün und Blau. Damit lassen sich die Farbwerte besser vergleichen und in eine zweite Darstellung, das **HSV-Farbmodell**, übertragen. Dieses Modell unterscheidet Farbton, Sättigung und Helligkeit und eignet sich damit hervorragend, um typische Feuerfarben wie Rot, Orange und Gelb gezielt zu erkennen.

Die Software prüft für jeden Pixel, ob er innerhalb dieser Farbbereiche liegt und gleichzeitig ausreichend hell und gesättigt ist. Wird ein Pixel als „heiß“ klassifiziert, erhöht sich der Zähler für auffällige Bildpunkte. Am Ende der Analyse wird das Verhältnis auffälliger Pixel zur Gesamtzahl berechnet - die sogenannte Hot Ratio. Dieser Wert wird genutzt, um eine Entscheidung zu treffen: Liegt er unterhalb eines bestimmten Schwellenwerts, wird kein Feuer angenommen. Steigt er über einen mittleren Wert, zeigt das System einen möglichen Brand an. Überschreitet er einen höheren Grenzwert, geht die Software von einem wahrscheinlichen Brand aus.

Um Fehlalarme zu vermeiden, wird die Hot Ratio über mehrere Bilder hinweg geglättet. Einzelne Störungen oder helle Reflexionen reichen so nicht aus, um sofort ein Signal auszulösen. Erst wenn sich ein Muster über mehrere Aufnahmen bestätigt, wird ein eindeutiger Status gemeldet. Die Ergebnisse werden im seriellen Monitor ausgegeben und gleichzeitig an die LED-Steuerung übergeben.

3.4 Motorsteuerung

Der Motor ist ein ergänzendes Element, das in einem festen Muster betrieben wird. Seine Aufgabe besteht darin, regelmäßige Bewegungen auszuführen, die beispielsweise genutzt werden können, um den Blickwinkel der Kamera zu verändern oder zusätzliche mechanische Effekte zu erzeugen.

Damit der Motor zuverlässig arbeitet, wurde eine *Startverzögerung* eingebaut. Nach dem Einschalten des Systems wartet der Motor zunächst für eine Minute, bevor er sich das

erste Mal bewegt. In dieser Zeit können sich die anderen Komponenten stabilisieren. Danach beginnt der Motor mit einem wiederkehrenden Bewegungsmuster: Er fährt seine Leistung langsam hoch, läuft für eine bestimmte Zeit, wird dann wieder abgebremst und macht eine Pause. Nach mehreren dieser Schritte folgt eine längere Ruhepause, bevor der Zyklus von Neuem beginnt.

Die Motorsteuerung basiert auf dem Prinzip der **Pulsweitenmodulation**. Das bedeutet, dass der Motor nicht nur ein einfaches Ein/Aus-Signal erhält, sondern die Stärke des Signals fein abgestuft geregelt werden kann. Dadurch lassen sich weiche Übergänge beim Anlaufen und Abbremsen umsetzen. Diese Rampe schont die Mechanik und sorgt für einen gleichmäßigeren Lauf.

Im Code ist festgelegt, wie lange die einzelnen Phasen dauern, mit welcher Leistung der Motor betrieben wird und wann Pausen eingelegt werden. Auf diese Weise ist das Bewegungsmuster genau reproduzierbar.

3.5 LED-Signalisierung

Die LED dient als visuelles Warnsignal und zeigt jederzeit den aktuellen Zustand der Auswertung an. Direkt nach dem Einschalten führt sie einen kurzen Selbsttest durch, indem sie zweimal blinkt. Anschließend übernimmt die Software die Steuerung vollständig auf Basis der Bildanalyse.

Es gibt drei mögliche Zustände. Wird kein Feuer erkannt, bleibt die LED ausgeschaltet. Wird ein möglicher Brand festgestellt, beginnt die LED in einem festgelegten Rhythmus zu blinken. Wird ein wahrscheinlicher Brand erkannt, leuchtet die LED dauerhaft. Auf diese Weise erhält der Benutzer eine sofortige visuelle Rückmeldung über die Bewertung der Kamera.

Auch die LED läuft in einem eigenen Task. Damit reagiert sie sofort auf Änderungen des Analyseergebnisses und wird nicht durch die Verarbeitung von Kamerabildern oder Motorsteuerungen verzögert.

3.6 Das Zusammenspiel der Komponenten

Die Software ist so aufgebaut, dass Kamera, Motor und LED klar voneinander getrennt arbeiten, aber dennoch über gemeinsame Variablen miteinander verbunden sind. Die Kamera analysiert alle zwei Sekunden ein Bild und berechnet die Hot Ratio. Auf Basis dieses Werts entscheidet die Software über den Zustand der LED, die das Ergebnis für den Benutzer sichtbar macht. Der Motor arbeitet unabhängig in seinen Bewegungszyklen und wird durch die Analyse nicht beeinflusst.

Durch die Nutzung von FreeRTOS laufen alle drei Komponenten gleichzeitig. Das bedeutet, dass die LED sofort reagieren kann, auch wenn die Kamera gerade ein Bild verarbeitet. Gleichzeitig bleibt der Motor in seinem Takt, ohne durch die anderen Prozesse gestört zu werden. Diese klare Trennung und parallele Ausführung sorgen dafür, dass das System stabil und zuverlässig arbeitet. Die Architektur ist außerdem modular aufgebaut, sodass einzelne Komponenten bei Bedarf angepasst oder erweitert werden können, ohne die gesamte Struktur zu verändern.

4. Bauanleitung für den Prototypen

Mindestteile:

- ESP32
- ESP32-Kameramodul
- Akku (3,7v)
- Motor (JYCL0610R2540 DigiKey)
- Propeller
- MOSFET (2N 7000 Vg)
- Diode (SS14 oder 1N5819)
- Widerstand (100 kOhm)
- Verbinder/Motorhalterung
- LED
- Lötset mit Lötkolben, Lötzinn und Lötöl

- Heißkleber
- Kabel
- Messer/Abisolierzange
- Ballons
- Helium
- Angelschnur

Um die Nicht-Drohne zu bauen, benötigt man ein ESP32 mit Kameramodul, einen Motor-Driver, einen Propeller, einen Motor und einen Akku. Wichtig ist, dass der Motor-Driver zum Motor passt und dass die Spannungswerte von Akku und Motor kompatibel sind.

Im Folgenden beschreibe ich, wie wir vorgegangen sind. Es ist jedoch nicht zwingend notwendig, jeden Schritt exakt zu übernehmen. Bei der Verwendung eines anderen Motors oder Akkus müssen die restlichen Bauteile entsprechend angepasst werden, damit die Schaltung weiterhin mit dem ESP32 funktioniert.

Je nach gewähltem Modell oder Anbringungsmöglichkeit wird zusätzlich ein Ballon oder ein Drachen benötigt, der das Projekt trägt. Wir haben uns für Folienballons entschieden, die mit Helium gefüllt und mit Angelschnur verbunden wurden.

Zur Befestigung der Komponenten benötigt man ein Lötset, etwas Heißkleber und in unserem Fall die Angelschnur, um die Konstruktion am Ballon zu befestigen.

Der Motor wurde aufgrund seines geringen Gewichts und der verfügbaren Optionen ausgewählt. Es handelt sich um einen kleinen Vibrationsmotor (wie er z. B. in Handys verwendet wird), bei dem der Vibrationskopf durch einen Plastikstecker ersetzt wurde. Dies erleichtert das Anbringen eines Propellers.

Der Propeller sowie die Motorhalterung sind 3D-gedruckte Open-Source-Modelle. Solche Modelle findet man z. B. auf Printables oder Makersworld. Wir haben unseren Propeller von Printables bezogen. Dort sind die meisten Modelle frei verfügbar, dürfen jedoch nicht für kommerzielle Zwecke genutzt oder als eigene Modelle weiterverbreitet werden.

- https://www.printables.com/model/28547-flying-propeller-toy

Falls erforderlich, muss der Propelleranschluss an die eigenen Bedürfnisse angepasst werden. Ich habe dies mit Fusion 360 erledigt, das als kostenlose Studentenversion verfügbar ist. Alternativ gibt es zahlreiche frei erhältliche CAD-Programme, die dafür geeignet sind.

Die Halterung des Motors am ESP32 erfolgte über einen dünnen Plastiksteg. Alternativ kann man auch einen stabilen, leichten Gegenstand wie einen Holzzahnstocher verwenden. Der Motor wurde per Press-Fit in die Halterung eingesetzt; ein Verkleben wäre ebenfalls möglich. Auch der Propeller wurde per Press-Fit angebracht.

Der Motor wird mit dem Motor-Driver verbunden. Sollten keine Kabel beiliegen, müssen diese selbst beschafft und am Motor angelötet werden.

Für den Motor-Driver werden ein MOSFET, eine Diode und ein Widerstand benötigt. Die Bauteile müssen zueinander passen, um ein Durchbrennen des Motors zu vermeiden.

Verschaltung:

- Am linken Pin des MOSFET wird ein Kabel angelötet und mit Ground des ESP sowie mit einer Seite des Widerstands verbunden.
- Der Widerstand wird mit dem mittleren Pin des MOSFET sowie mit dem Steuerpin des ESP verbunden (bei uns war es D9, andere Pins sind ebenfalls möglich).
- Der rechte Pin des MOSFET wird mit der Kathode der Diode sowie mit dem schwarzen Kabel des Motors (Minuspol) verbunden.
- Die Anode der Diode wird mit dem roten Kabel des Motors und mit einem zusätzlichen roten Kabel verbunden, das wiederum zum Pluspol der Batterie führt.

Damit ist der Motor-Driver aufgebaut.

- Die LED mit dem Grund Fuß am GND vom ESP löten und den anderen Fuß an einen Freien Anschluss z.B. D10.

Nun wird das zusätzliche rote Kabel mit der Batterie verbunden. Dazu entfernt man vorsichtig die Isolierung am roten Batteriekabel und lötet das Kabel an. Die Batteriekabel müssen außerdem an die Batteriepole des ESP32 angelötet werden. Dabei ist darauf zu achten, den ESP nicht durch zu große Hitze zu beschädigen.

Die Verbindungen sollten anschließend isoliert und verstärkt werden. Wir haben dafür Heißkleber verwendet, der die Anschlüsse schützt und das Abreißen der Kabel verhindert.

Im letzten Schritt wird die Motor-Propeller-Einheit am ESP angebracht. Auch hier haben wir Heißkleber verwendet. Wichtig ist, dass die gesamte Konstruktion ausbalanciert ist. Wir haben die Batterie auf der gegenüberliegenden Seite herunterhängen lassen und durch mehrere Versuche ein stabiles Gleichgewicht erreicht.

Zum Schluss wird die Angelschnur durch den Schwerpunkt der Konstruktion geführt und am Ballon befestigt – damit ist der Prototyp einsatzbereit.

5. Realisierung

Der Prototyp wurde in mehreren Schritten aufgebaut. Zunächst erfolgte die Auswahl der zentralen Hardwarekomponenten (ESP32-S3 mit Kamera Modul, LED, Vibrationsmotor mit Propeller, Akku). Für die Steuerung des Motors wurde ein einfacher Treiber auf Basis eines MOSFETs aufgebaut, der um eine Schutzdiode und einen Widerstand ergänzt wurde.

Im Anschluss wurden die Komponenten auf einer kleinen Trägerkonstruktion montiert und mit dem Ballon verbunden. Dabei zeigte sich, dass insbesondere das Gewicht und die Balance des Systems eine zentrale Rolle spielten. Durch die Platzierung der Batterie auf der Gegenseite des Motors konnte ein stabiles Gleichgewicht hergestellt werden.

Die Programmierung und Inbetriebnahme erfolgte iterativ: Zunächst wurden Kamera, LED und Motor einzeln getestet, danach in das Gesamtsystem integriert. Besonders die Synchronisation von Motorbewegung und Kameraaufnahme stellte sich als Herausforderung heraus, da Bildaufnahme und Bewegung teilweise gleichzeitig liefen. Durch Anpassung der Zeitsteuerung und die Nutzung von FreeRTOS konnte dieses Problem gelöst werden.

Die endgültige Version des Prototyps konnte Bilder aufnehmen, auf feuerähnliche Farben prüfen und die LED-Signalisierung entsprechend ansteuern.

6. Fazit und Ausblick

Im Rahmen des Projekts ***Not a Drone*** konnte ein funktionsfähiger Prototyp zur Früherkennung von Waldbränden entwickelt werden. Ziel war es, eine kostengünstige und flexible Alternative zu bestehenden Überwachungssystemen aufzuzeigen. Durch den Einsatz eines Ballons als Trägerplattform und die Kombination von Kamera, Auswertungseinheit und LED-Signalisierung konnte gezeigt werden, dass eine kontinuierliche Überwachung mit einfachen Mitteln möglich ist.

Die Tests haben deutlich gemacht, dass die Grundidee technisch umsetzbar ist, jedoch noch mehrere Herausforderungen bestehen. Dazu zählen insbesondere die Abhängigkeit von Wetterbedingungen, die begrenzte Bildqualität der verwendeten Kamera sowie die Balance und Stabilität des Aufbaus. Auch die Erkennung von Feuerfarben erwies sich als anfällig für Störeinflüsse wie Reflexionen oder Sonnenlicht.

Trotz dieser Einschränkungen stellt der Prototyp eine solide Basis für weitere Entwicklungen dar. Zukünftig könnte das System durch den Einsatz leistungsfähigerer Kameras oder Infrarotsensoren verbessert werden. Ebenso wäre eine direkte Funkanbindung an bestehende Notrufsysteme sinnvoll, um Warnungen nicht nur lokal, sondern auch zentral weiterzugeben. Langfristig könnte ein Netzwerk solcher mobilen Einheiten aufgebaut werden, die gefährdete Regionen flächendeckend überwachen.

Damit zeigt das Projekt sowohl das Potenzial als auch die Grenzen eines einfachen, mobilen Frühwarnsystems auf und legt den Grundstein für weiterführende Arbeiten in diesem Bereich.

7. Reflexion

Die Durchführung des Projekts war sowohl technisch als auch organisatorisch eine lehrreiche Erfahrung. Besonders die iterative Vorgehensweise machte deutlich, wie wichtig es ist, einzelne Schritte regelmäßig zu testen und flexibel auf neue Probleme zu reagieren.

Herausforderungen traten vor allem bei der praktischen Umsetzung auf: Ein zentrales Problem bestand darin, dass einige elektrische Verbindungen nicht stabil genug waren, sodass zunächst unklar blieb, ob Fehler durch die Hardware selbst oder durch die eigene Programmierung verursacht wurden. Eine weitere Schwierigkeit lag in der Fehlersuche innerhalb des Codes: Da mehrere Komponenten parallel liefen, war es oft komplex nachzuvollziehen, welche Abläufe voneinander abhingen und warum bestimmte Funktionen nicht wie erwartet ausgeführt wurden. Auch der Aufbau des Gesamtsystems erwies sich als anspruchsvoll, weil viele Einzelteile miteinander verbunden werden mussten, bis alle Komponenten zuverlässig zusammenarbeiteten.

Positive Aspekte zeigten sich hingegen vor allem in der Organisation und Zusammenarbeit. Das Team ging strukturiert mit der verfügbaren Zeit um, verteilte die Aufgaben klar und setzte die richtigen Prioritäten, sodass trotz Schwierigkeiten ein funktionierender Prototyp entstehen konnte. Unterschiedliche Vorkenntnisse der Mitglieder wurden effektiv genutzt, wodurch Hardware, Software und Projektmaterialien sinnvoll zusammengeführt werden konnten. Besonders hilfreich war außerdem die offene Kommunikation im Team, die half, Missverständnisse zu vermeiden und Lösungswege gemeinsam zu entwickeln. Neben den technischen Kenntnissen zu Mikrocontrollern, Sensorik und paralleler Programmierung wurden dadurch auch wichtige Erfahrungen im Projektmanagement, in der Problemlösung und Zusammenarbeit gesammelt.

Der vollständige Quellcode sowie weiterführende Dokumentationen sind im GitHub-Repository abrufbar und können als Grundlage für künftige Verbesserungen oder Nachbauten dienen:

GitHub-Repository: https://github.com/NurzhanSeitzhanov/Not-a-Drone_Big-Data