Variante 2: Programmierung mit Unterstützung

1. Überblick & Zielsetzung

Ziel:
Die Schüler entwickeln ein eigenes Programm für den Bordcomputer. Dieser soll:

- Sensordaten als Datenlogger aufzeichnen
- Bei Bedarf den Fallschirm auslösen

2. Komponenten verstehen

Verschaffe dir einen Überblick über alle Bauteile des Moduls. Nutze dazu den vereinfachten Schaltplan.

3. Rechercheaufgabe

In der Informatik kommunizieren Bauteile über sogenannte Schnittstellen (Interfaces). Um herauszufinden, wie die Sensoren funktionieren und wie man sie ansteuert, hilft eine gezielte Online-Recherche (z. B. mit den Begriffen „ESP32“ oder „Arduino“).

Fülle die folgende Tabelle aus, indem du die Funktionen und Eigenschaften der Bauteile recherchierst:

Bauteil	Funktion	Schnittstelle Steuerung	Betriebsspannung	Besonderheiten
ESP32				
ICM-20948				
BME280				
Servo				
Schalter Taster				
LCD – Display				

Tipp: Oft hilft es, bei der Recherche Begriffe wie „ESP32“ oder „Arduino“ zu verwenden.

4. Schnittstellen & Spannungen verstehen

Initialisierung von Bauteilen – Warum ist das wichtig?

Bevor ein Mikrocontroller wie der ESP32 mit angeschlossenen Bauteilen arbeiten kann (z. B. Sensoren, Servos oder Schalter), müssen diese im Programm initialisiert werden. Das bedeutet:

- Die Verbindung zwischen Mikrocontroller und Bauteil wird im Code festgelegt.
- Die Kommunikationseinstellungen (z. B. über welche Schnittstelle kommuniziert wird) werden definiert.
- Es wird geprüft, ob das Bauteil erkannt und funktionsfähig ist.

Warum ist das notwendig?
Ohne diese Initialisierung „weiß" der ESP32 nicht, welches Gerät angeschlossen ist oder wie er mit diesem kommunizieren soll. Es ist vergleichbar mit einem Gerät, das angeschlossen wurde, aber nicht eingeschaltet oder konfiguriert wurde.

Was sind Schnittstellen?
Schnittstellen (auch „Interfaces" genannt) sind digitale Verbindungsmöglichkeiten, über die zwei Geräte Daten austauschen können.

Wichtige Schnittstellen auf dem ESP32:

- I^2C: Eine Bus-Schnittstelle für viele Sensoren über nur zwei Leitungen (SDA & SCL)
- SPI: Schnelle serielle Schnittstelle, ideal für SD-Karten oder Displays
- GPIO: Allgemeine Ein-/Ausgänge, z. B. für Taster, LEDs oder einfache Sensoren
- PWM: Pulsweitenmodulation – wird z. B. für Servos oder Helligkeitssteuerung genutzt

Beispiel: Der Sensor BME280 nutzt I^2C – dafür muss man im Code angeben, an welchen Pins SDA (Datenleitung) und SCL (Taktleitung) angeschlossen sind.

Typische Spannungen:

- 3,3 V: Die meisten Sensoren für den ESP32
- 5,0 V: Häufig bei Arduino Uno oder bei Servos

Achtung: Der ESP32 arbeitet intern mit 3,3 V. Bauteile, die mit 5 V kommunizieren, brauchen oft einen Pegelwandler!

Zusammengefasst:

- Initialisierung: Der ESP32 muss wissen, welches Gerät da ist und wie man es anspricht.

- Schnittstellen: Jede Bauteilart spricht eine eigene Sprache (z. B. I^2C, SPI, PWM…)
- Spannung: Zu hohe Spannung zerstört Bauteile – zu wenig Spannung verhindert Funktion

5. Programmaufbau verstehen

Der ESP32 führt Programme in einer festen Reihenfolge aus.

Er startet immer mit:

```
void setup() {

}
```

Dann läuft dauerhaft:

```
void loop() {

}
```

Zur Erleichterung sind bereits Bereiche vordefiniert, die zu bestimmten Zeitpunkten aufgerufen werden. (Sind mit Kommentaren markiert)

Typischer Aufbau:

Programmbereich	Aufgabe
setup()	Einmalige Aufgaben Komponenten initialisieren (.init() Methoden)
loop()	Wiederkehrende Aufgaben Daten aufzeichnen, Fallschirm-Bedingung prüfen

6. Klassendiagramm & Methoden

Alle Sensoren sind als Klassen implementiert. Die Methoden zur Initialisierung und Abfrage sind standardisiert.

Hinweis: Initialisierungen geben einen booleschen Wert zurück (true bei Erfolg).

Beispielmethoden:

- *bool initSensor()*
- *float getAltitude()*
- *bool initWebserver()*

7. Erweiterung: Webinterface

Optional kann der ESP32 eine Webseite bereitstellen, um Live-Daten anzuzeigen. Dafür wird die Methode initWebserver() verwendet.

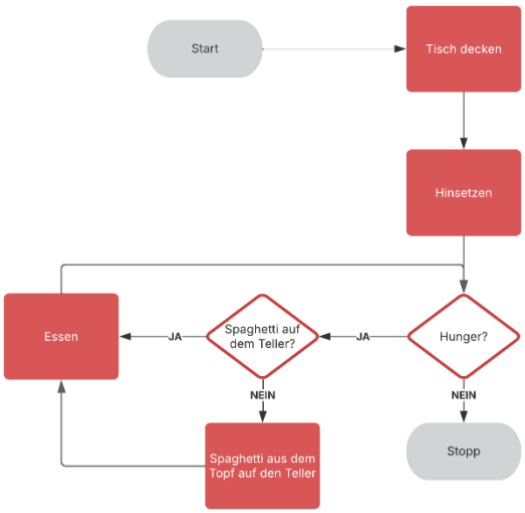
8. Programmablauf planen

Erstelle ein Flussdiagramm, das den Ablauf deines Programms beschreibt.

Dein Auftrag:

- Entwickle allein ein Flussdiagramm zum Ablauf eures Raketenprogramms.
- Tauscht euch danach in der Gruppe aus.
- Diskutiert Vor- und Nachteile eurer Lösungen.

Als Beispiel für ein Flussdiagramm hier ein potenzieller Ablauf von Spaghetti essen.



9. Diskussionspunkte

- Was passiert, wenn ein Sensor nicht erkannt wird?
- Was, wenn die SD-Karte fehlt oder voll ist?
- Wie könnt ihr Fehler erkennen und melden?

10. Umsetzung auf dem ESP32

Lade das Programm über PlatformIO auf den ESP32. Arbeite nur in der Datei main.cpp (im src-Ordner). Verwende ausschließlich die vordefinierten Methoden. (s. letztes Blatt)

11. Wie wird der Fallschirm ausgelöst?

Entscheide dich für eine oder mehrere Bedingungen:

- Höhenabfall größer als X m?
- Hohe Beschleunigung?
- Drehbewegung (Gyroskop)?

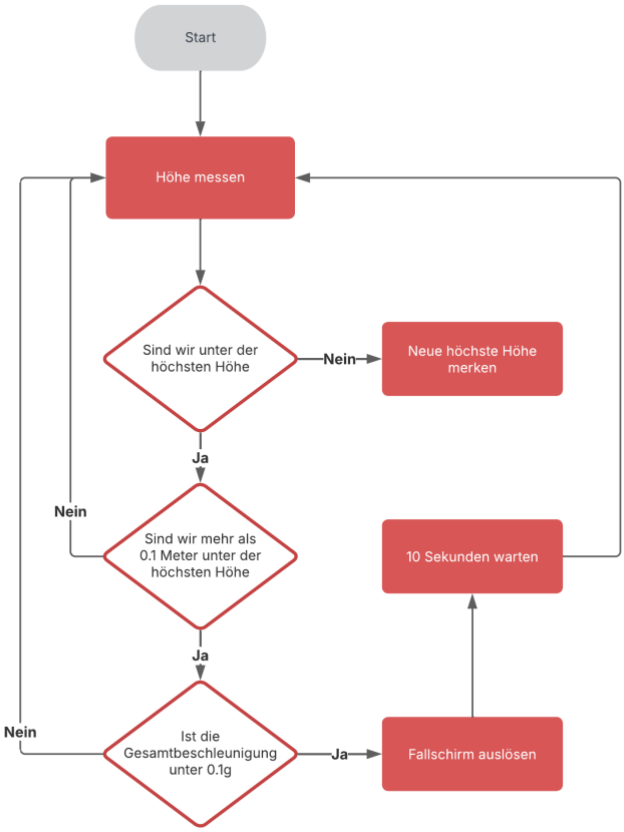
→ entwickle ein detailliertes Flussdiagramm für die Fallschirmauslösung

(Vorschlag auf der nächsten Seite)

Vergleicht euer Ergebnis mit dem Vorschlag. Stellt Vor- Nachteile eure Lösung im Vergleich zum Vorschlag dar.

Ein paar weiter Diskussionspunkte:

- Warum mehr als 0.1 Meter unter höchster Höhe
- Warum 10 Sekunden warten?
- Warum die Gesamt-beschleunigung?
- Wo würde man einfügen die geflogene Höhe auf dem Display anzuzeigen?



12. Anschluss & Verdrahtung

Nutze den Schaltplan, um die Pinbelegung nachzuvollziehen.

Die richtige Verdrahtung ist Voraussetzung für eine funktionierende Programmierung.

Testen des Moduls

Nehmt euch das fertige Modul und überprüft, ob es wie gewünscht funktioniert. Beobachtet dabei insbesondere den Moment der Auslösung: Öffnet sich der Fallschirm genau dann, wenn ihr es erwartet habt? Überlegt euch mögliche Ursachen, wenn das Modul zu früh oder zu spät auslöst. Mögliche Fehlerquellen könnten sein:

- Ungenaue Sensordaten (z. B. Rauschen oder Verzögerungen)
- Fehlerhafte Berechnung der Höhe oder des Höhenabfalls
- Falsche Referenzwerte im Code
- Verzögerungen durch den Servo

Testet außerdem die Datenaufzeichnung auf der SD-Karte:

- Ist das Format korrekt? (z. B. CSV mit Kommas oder Semikolon getrennt)
- Stimmen die Einheiten? (z. B. Meter für Höhe, °C für Temperatur, hPa für Luftdruck)
- Sind die Messwerte in einer sinnvollen Größenordnung? (z. B. keine negativen Höhenwerte oder unrealistisch hohe Temperaturen)

Versucht, die aufgezeichneten Daten zu validieren:

- Ist der Verlauf der Höhe plausibel? Gibt es z. B. einen klar erkennbaren Anstieg beim Start und einen Abfall nach der höchsten Stelle?
- Entsprechen die Temperaturwerte etwa der Umgebungstemperatur?
- Verhält sich der Luftdruck physikalisch korrekt (z. B. sinkender Druck mit zunehmender Höhe)?

Diese Tests helfen euch, mögliche Fehler frühzeitig zu erkennen und die Funktion eures Systems zu verbessern.

Aufzeichnen der Flugdaten

Für die Test empfehlen wir die Finnen 1 und die ogive Spitze. Die Flasche sollte mit 500ml Wasser gefüllt sein und bis 5 bar Luftdruck.

Jetzt wird’s spannend: Deine Rakete ist bereit für den Start – und dein Messmodul soll alle wichtigen Daten aufzeichnen! Höhe, Luftdruck, Temperatur und Bewegung geben dir spannende Einblicke in den Flugverlauf. Mit diesen Daten kannst du später genau analysieren, wann der höchste Punkt erreicht wurde, wie schnell die Rakete steigt – und ob der Fallschirm im richtigen Moment ausgelöst wurde.

Genaue Startanleitung in Plug-and-Play.

Hier ein Tutorial, wie man den Fallschirm richtig faltet, dass er zuverlässig aufgeht.

https://www.youtube.com/shorts/kGTpw6yevmc?feature=share

Sicherheitshinweise

Bevor du mit dem Start beginnst, beachte folgende Punkte:

- Luftpumpe prüfen: Achte darauf, dass die Luftpumpe einen genau Anzeige hat. Die 5 Bar dürfen nicht überschritten werden.
- Zustand der Flasche prüfen: nach einigen Flügen können sich Risse am Hals der Flasche bilden. Außerdem wird die Flasche zunehmend dünner. Wechsle die Flasche nach 10 Starts aus.
- Sicherheitsabstand einhalten: Alle Zuschauer und Beteiligten müssen beim Start mindestens 5 m Abstand halten.
- Schutzbrille tragen: Bei Arbeiten mit Druckluft ist eine Schutzbrille Pflicht.
- Modul richtig befestigen: Das Modul muss mit der Flasche stabil verklebt sein, da diese sonst getrennt herunterfallen könnten.
- Startrampe nur auf freiem Gelände verwenden: Kein Start in der Nähe von Gebäuden, Straßen oder Stromleitungen. Empfehlung hierzu ist einer der kurzen Seiten eines Sportplatzes und leicht in die andere Richtung ausrichten. (maximal 20 Grad von der Senkrechten)
- Nie auf Menschen oder Tiere zielen: Die Rakete ist kein Spielzeug!
- Niemals die Rakete versuchen zu fangen.

Analyse der Daten

Aufgabe: Auswertung der Flugdaten mit Excel

Für die Auswertung der Daten könnt euch für das Programm eure Wahl entscheiden. Bei Vorerfahrung kann Python mit dem Modul MathPlotLib eine Möglichkeit sein. Für den ersten Umgang mit Daten empfiehlt sich Excel.

Auswertung mit Excel:

1. CSV-Datei öffnen

- Starte Excel
- Öffne deine log_XXXX.csv-Datei
- Spaltenüberschriften umbenennen:
- Zeit [s], Höhe [m], Beschleunigung [g], Luftdruck [hPa], Temperatur [°C] etc
- Achte auf richtige Schreibweise (Excel verwendet Kommas für die Nachkommastellen, in der CSV Datei sind es Punkte – das Tool Suchen und Ersetzten kann helfen)

2. Höhe über Zeit – Diagramm erstellen

- Spalten „Zeit [s]“ und „Höhe [m]“ markieren
- Einfügen → Liniendiagramm
- Flugverlauf sichtbar machen

Arbeite aus den Daten folgende Wert heraus.

- Höchste Beschleunigung
- Apogäum
- Flugdauer
- Maximale Geschwindigkeit
- Sinkgeschwindigkeit
- Kann man daraus die Formel für den Senkrechten Wurf ausarbeiten?
- Bestimme den c_w Wert des Fallschirms (Masse, Luftdichte und Fläche gegeben)
- Bestimme die Schubkurve Wasserrakete – wie unterscheidet sie schon von großen Raketen bzw. Modellraketen mit Schwarzpulvermotor

Daten präsentieren

Nachdem du deine Flugdaten ausgewertet hast, geht es nun darum, die Ergebnisse klar und verständlich darzustellen. Eine gute Präsentation hilft dabei, deine Messungen nachvollziehbar zu machen und besondere Erkenntnisse hervorzuheben.

Übersichtliche Darstellung

- Erstelle eine strukturierte Auswertung in Excel oder PowerPoint.
- Nutze Tabellen für Werte wie Apogäum, Flugdauer, Maximalgeschwindigkeit etc.
- Verwende Diagramme, z. B.:
    - Höhe über Zeit (für Flugkurve)
    - Geschwindigkeit über Zeit
    - Schubkurve
    - Beschleunigung über Zeit

Hebe Besonderheiten hervor

- Zeige deutlich, wo das Apogäum liegt
- Benenne Stellen mit besonderer Beschleunigung oder auffälligem Flugverhalten.
- Wenn der Fallschirm ausgelöst wurde: Markiere diesen Zeitpunkt (z. B. durch eine vertikale Linie im Diagramm).
- Was ist bei euren Daten besonders
- Gab es Probleme bei der Auswertung

Wenn jede Gruppe die gleichen Daten vorstellt, kann es schnell langweilig werden. Konzentriert euch auf die Erfolge oder Probleme, die ihr das Projekt hinweg hattet.

Interpretation

Erkläre mit eigenen Worten:

- Was passiert wann?
- Warum sieht die Kurve so aus?
- Welche Phase des Flugs erkennst du (Start, Steigflug, Apogäum, Sinkflug)?
- Vergleiche berechnete Werte mit theoretischen Erwartungen (z. B. Höhe beim senkrechten Wurf).

Kreative Präsentationsform

- Erstelle ein kurzes Poster, ein Video mit Flug- und Datenanalyse oder einen digitalen Bericht.
- Baue Screenshots deiner Diagramme ein.
- Ergänze ein Fazit: Was würdest du beim nächsten Mal anders machen?

Ziel ist nicht nur, die Daten zu zeigen – sondern auch zu erklären, was sie bedeuten und was du daraus gelernt hast.

Methoden

Methoden	Funktion
init()	Starte die Protokolle für SD-Karte und Sensoren
initWebserver()	Starte Webserver und WLAN
initRecording()	Legt die CSV - Datei an
record()	Legt die aktuellen Werte auf der SD-Karte ab
check_parachute()	Gibt true zurück, wenn der Fallschirm auslösen soll
deploy_parachute()	Löst den Pin für Fallschirm aus Sollte wieder eingefahren werden
stop_parachute()	Fährt den Pin des Fallschirms wieder aus
reset()	Setzt die Daten für die Fallschirmauslösung zurück

Alle Komponenten müssen mit .init() aufgerufen werden und deren Status in die dazugehörige Variable gespeichert werden.

Bsp:

bme_active = bme.init();

Dies gilt für:

- icm
- bme
- lipo
- oled
- servo